aws

開發人員指南

AWS Elastic Beanstalk

AWS Elastic Beanstalk: 開發人員指南

Table of Contents

什麼是 AWS Elastic Beanstalk？

Amazon Web Services (AWS) 包含一百種以上的服務，每種服務均有各自的功能區域。雖然各種服務讓您能夠靈活管理 AWS 基礎設施，但如何選擇應使用的服務並加以佈建，則可能十分困難。

Elastic Beanstalk 讓您能夠在 AWS 雲端快速部署和管理應用程式，而無需了解執行那些應用程式的基礎設施。Elastic Beanstalk 可降低管理複雜性而不會限制選擇或控制。您只需上傳應用程式，Elastic Beanstalk 將自動處理容量佈建、負載平衡、擴展和應用程式運作狀態監控的細節。

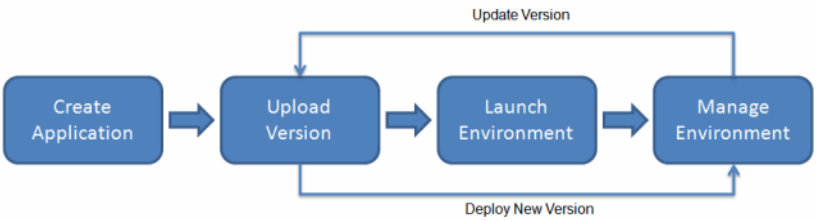
Elastic Beanstalk 支援以 Go、Java、.NET、Node.js、PHP、Python 及 Ruby 開發的應用程式。當您部署應用程式時，Elastic Beanstalk 會建置所選取的支援平台版本，並佈建一或多個 AWS 資源 (例如 Amazon EC2 執行個體)，以執行您的應用程式。

您可使用 Elastic Beanstalk 主控台、AWS Command Line Interface (AWS CLI) 或 eb (專門針對 Elastic Beanstalk 設計的高階 CLI)，藉此與 Elastic Beanstalk 互動。

若要進一步了解如何使用 Elastic Beanstalk 部署範例 Web 應用程式，請參閱 AWS 入門：部署 Web 應用程式。

您亦可執行大部分的部署任務，例如直接透過 Elastic Beanstalk Web 介面 (主控台) 變更 Amazon EC2 執行個體集群的大小或監控您的應用程式。

若要使用 Elastic Beanstalk，您須建立應用程式、將應用程式版本 (採用諸如 Java .war 檔案之應用程式原始碼套件的形式) 上傳至 Elastic Beanstalk，然後提供應用程式的部分資訊。Elastic Beanstalk 會自動啟動環境，並建立及設定執行您程式碼所需的 AWS 資源。環境啟動後，即可管理您的環境並部署新的應用程式版本。下圖說明 Elastic Beanstalk 的工作流程。



建立並部署您的應用程式後，應用程式的相關資訊 (包括指標、事件和環境狀態) 均可透過 Elastic Beanstalk 主控台、API 或命令列界面 (包括統一的 AWS CLI) 取得。

定價

使用 Elastic Beanstalk 並不收取其他費用。您只需針對應用程式使用的基礎 AWS 資源付費。如需定價的詳細資訊，請參閱 Elastic Beanstalk 服務詳細資訊頁面。

後續作業

本指南提供 Elastic Beanstalk Web 服務的概念性資訊，以及有關如何使用此服務來部署 Web 應用程式的資訊。個別章節說明如何使用 Elastic Beanstalk 主控台、命令列界面 (CLI) 工具和 API 來部署和管理您的 Elastic Beanstalk 環境。本指南亦記載 Elastic Beanstalk 與其他 Amazon Web Services 提供的服務整合情況。

我們建議您先閱讀開始使用 Elastic Beanstalk，了解如何開始使用 Elastic Beanstalk。入門步驟會帶您逐步建立、檢視並更新 Elastic Beanstalk 應用程式，以及編輯和終止您的 Elastic Beanstalk 環境。入門章節亦說明不同的 Elastic Beanstalk 存取方法。

若要進一步了解 Elastic Beanstalk 應用程式及其元件，請參閱下列頁面。

- Elastic Beanstalk 概念
- Elastic Beanstalk 平台詞彙表
- Elastic Beanstalk 平台維護的共同責任模型
- Elastic Beanstalk 平台支援政策

開始使用 Elastic Beanstalk

為了協助您瞭解 AWS Elastic Beanstalk 運作方式，本教學課程將逐步引導您建立、探索、更新和刪除 Elastic Beanstalk 應用程式。一小時內即可完成。

使用 Elastic Beanstalk 沒有任何費用，但是它為本教程創建的 AWS 資源是實時的（並且不會在沙箱中運行）。您需要支付這些資源的標準使用費，直到在本教學結束時終止它們為止。總計費用一般不到 1 美元。如需如何將費用降至最低的資訊，請參閱 AWS 免費方案。

主題

設定：建立帳 AWS 戶

如果您還不是 AWS 客戶，則需要創建一個 AWS 帳戶。註冊使您可以訪問 Elastic Beanstalk 和您需要的其他 AWS 服務。

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」。

步驟 1：建立範例應用程式

在此步驟中，您會從預先存在的範例應用程式開始建立新的應用程式。Elastic Beanstalk 支援各種程式設計語言、應用程式伺服器及 Docker 容器的平台。您可以在建立應用程式時選擇平台。

建立應用程式和環境

您將會使用 Create application (建立應用程式) 主控台精靈建立範例應用程式。它會建立 Elastic Beanstalk 應用程式，並在其中啟動環境。環境是執行應用程式程式碼所需的 AWS 資源集合。

建立範例應用程式

1. 開啟 Elastic Beanstalk 主控台。
2. 選擇建立應用程式。
3. 在 Application name (應用程式名稱) 中，輸入 **`getting-started-app`**。
4. 選擇性新增應用程式標籤。
5. 關於 Platform (平台)，選擇一個平台。
6. 選擇下一步。
7. 畫面上會顯示設定服務存取頁面。
8. 針對服務角色，選擇使用現有服務角色。
9. 接下來，我們會將重點放在 EC2 執行個體設定檔下拉式清單。此下拉式清單中顯示的值可能會有所不同，視您的帳戶是否先前已建立新環境而定。

   根據清單中顯示的值，選擇下列其中一項。

   - 如果下拉式清單中顯示 `aws-elasticbeanstalk-ec2-role`，請從 EC2 執行個體設定檔下拉式清單中選取它。
   - 如果清單中顯示另一個值，而且它是適用於您環境的預設 EC2 執行個體設定檔，請從 EC2 執行個體設定檔下拉式清單中選取該值。
   - 如果 EC2 執行個體設定檔下拉式清單未顯示任何可供選擇的值，請展開隨後的程序，為 EC2 執行個體設定檔建立 IAM 角色。

完成為 EC2 執行個體設定檔建立 IAM 角色中的步驟來建立 IAM 角色，接下來，您可以為 EC2 執行個體設定檔選取該角色。然後，請返回此步驟。

現在您已建立 IAM 角色，重新整理清單，該角色就會在下拉式清單中顯示為選項。從 EC2 執行個體設定檔下拉式清單中選取您剛建立的 IAM 角色。

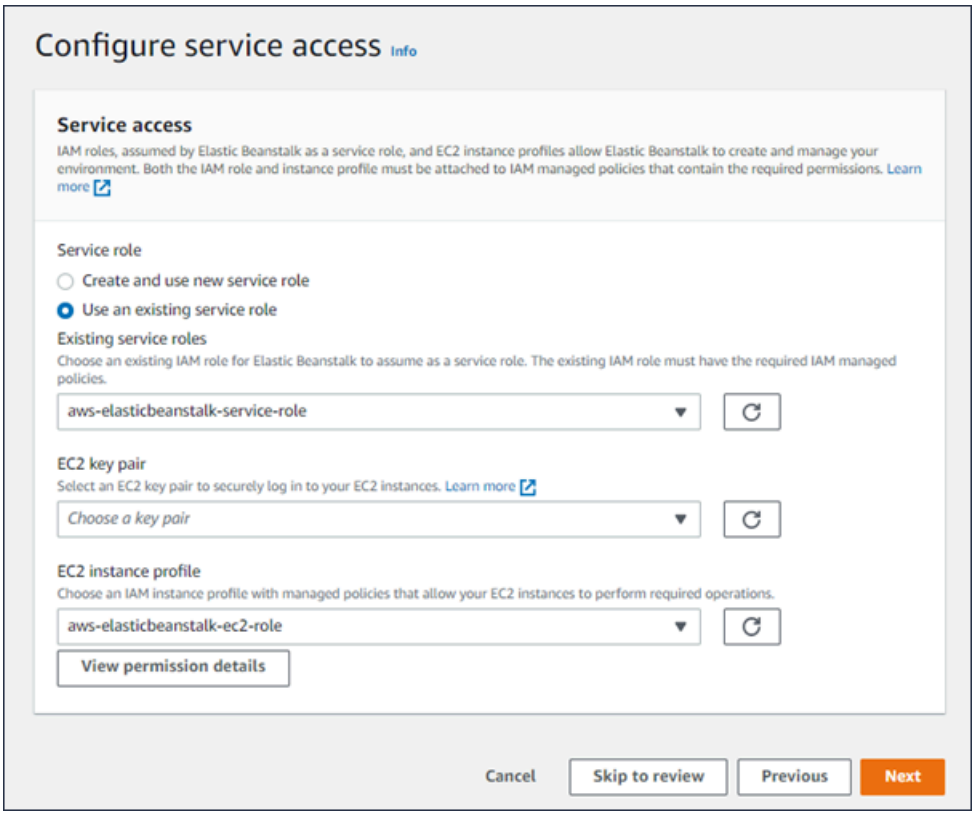
10. 在 Configure service access (設定服務存取)頁面上選擇 Skip to Review (略過以檢閱)。

    這會略過選擇性步驟。

11. Review (檢閱)頁面會顯示您所有選擇的摘要。

    選擇頁面底部的 Submit (提交) 。

為 EC2 執行個體設定檔建立 IAM 角色



若要建立 EC2 執行個體設定檔的 IAM 角色

1. 選擇檢視許可詳細資料。這會顯示在 EC2 執行個體設定檔下拉式清單下。

畫面上會顯示標題為檢視執行個體設定檔許可的模態視窗。此視窗會列出您需要附加至您建立的新 EC2 執行個體設定檔的受管設定檔。它也提供啟動 IAM 主控台的連結。

2. 選擇顯示於視窗頂端的 IAM 主控台連結。
3. 在 IAM 主控台導覽窗格中，選擇 Roles (角色)。
4. 選擇建立角色。
5. 在受信任的實體類型下，選擇 AWS 服務。
6. 在 Use case (使用案例) 下，選擇 EC2。
7. 選擇下一步。
8. 附加合適的受管政策。在檢視執行個體設定檔許可模態視窗中捲動，查看受管政策。這些政策也會列在此處：

   - `AWSElasticBeanstalkWebTier`
   - `AWSElasticBeanstalkWorkerTier`
   - `AWSElasticBeanstalkMulticontainerDocker`
9. 選擇下一步。
10. 輸入角色的名稱。
11. (選用) 附加標籤至角色。
12. 選擇建立角色。
13. 返回開啟的 Elastic Beanstalk 主控台視窗。
14. 關閉檢視執行個體設定檔許可模態視窗

> **Important**
>
> 請勿關閉顯示 Elastic Beanstalk 主控台的瀏覽器頁面。

15. 選擇 EC2 執行個體設定檔下拉式清單旁的

    (重新整理)。

    這會重新整理下拉式清單，讓您剛建立的「角色」會顯示在下拉式清單中。

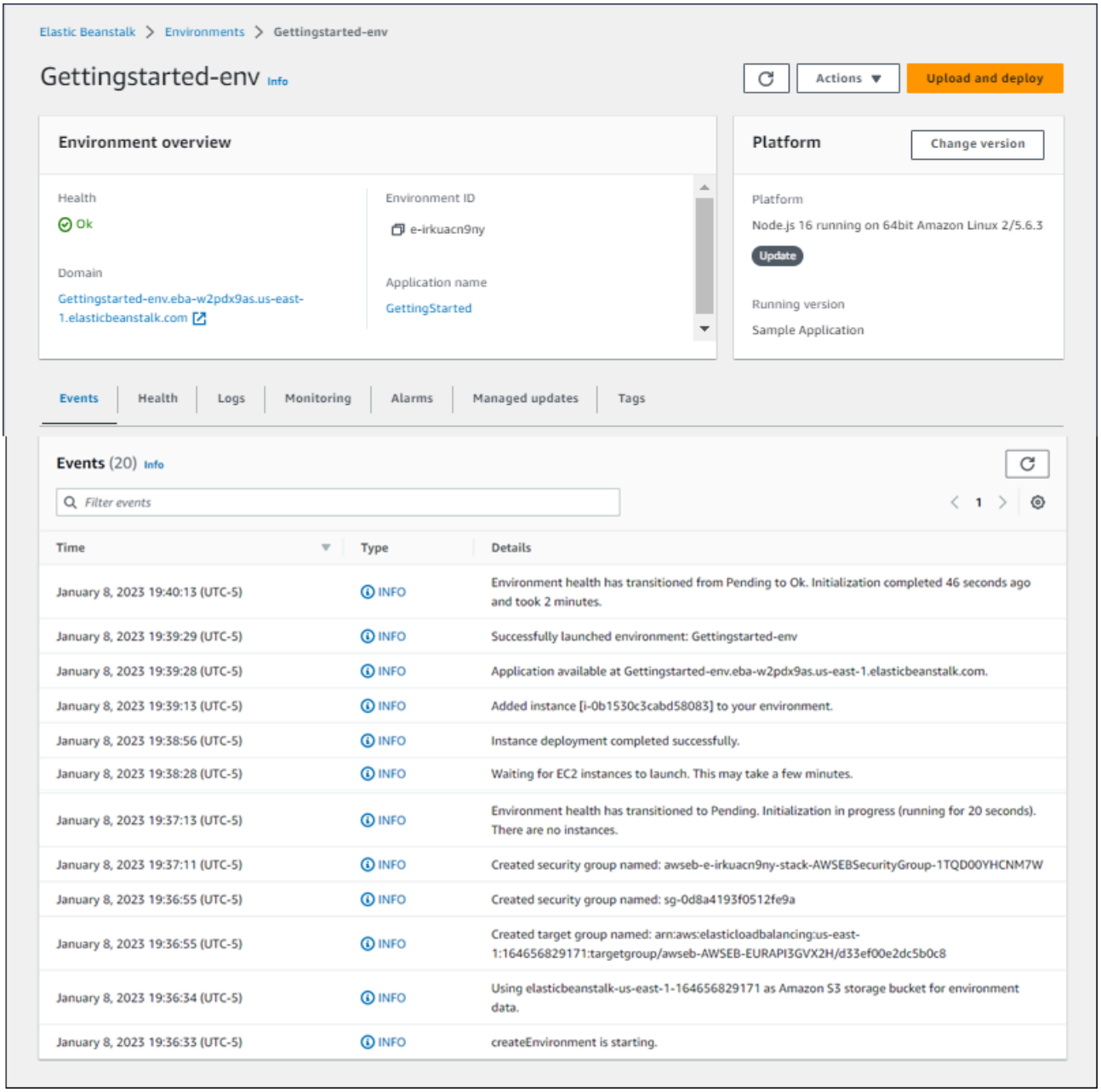
Elastic Beanstalk 工作流程

若要在 AWS 資源上部署和執行範例應用程式，Elastic Beanstalk 會採取下列動作。完成這些動作約需 5 分鐘。

1. 創建一個名為 Elastic Beanstalk 應用程序。getting-started-app
2. 使用以下 AWS 資源啟動名為 GettingStartedApp-env 的環境：
   - 一個 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體 (虛擬機器)
   - 一個 Amazon EC2 安全群組
   - 一個 Amazon Simple Storage Service (Amazon S3) 儲存貯體
   - Amazon CloudWatch 警報
   - 一個 AWS CloudFormation 堆疊
   - 網域名稱

   如需這些 AWS 資源的詳細資訊，請參閱the section called “AWS 為範例應用程式建立的資源”。
3. 建立名為 Sample Application 的新應用程式版本。這是預設的 Elastic Beanstalk 範例應用程式檔案。
4. 將範例應用程式的程式碼部署至 GettingStartedApp-env 環境。

在環境建立過程中，主控台會追蹤進度並顯示事件。



當所有的資源都已啟動，而且執行應用程式的 EC2 執行個體通過運作狀態檢查後，環境的運作狀態就會變更為 0k。您現在可以使用您 Web 應用程式的網站。

AWS 為範例應用程式建立的資源

當您建立範例應用程式時，Elastic Beanstalk 會建立下列 AWS 資源：

- EC2 執行個體 – 設定在您所選平台上執行 Web 應用程式的 Amazon EC2 虛擬機器。

每個平台會執行不同一套軟體、設定檔和指令碼，來支援特定的語言版本、架構、Web 容器或其組合。大多數的平台會使用 Apache 或 nginx 做為反向代理，處理您 Web 應用程式前端的網路流量、向它轉送請求、提供靜態資產，並產生存取和錯誤日誌。

- 執行個體安全群組 – 設定允許從連接埠 80 傳入流量的 Amazon EC2 安全群組。此資源可讓負載平衡器傳來的 HTTP 傳輸資料，到達執行您 Web 應用程式的 EC2 執行個體。在預設情況下，不允許傳輸資料從其他通訊埠傳送。
- Amazon S3 儲存貯體 - 儲存位置，用來儲存當您使用 Elastic Beanstalk 時所建立的原始程式碼、日誌和其他成品。
- Amazon CloudWatch 警示 — 兩個 CloudWatch 警示，用於監控環境中執行個體的負載，並在負載過高或過低時觸發。當警示觸發時，您的 Auto Scaling 群組會擴展或縮減以進行回應。
- AWS CloudFormation 堆疊 — Elastic Beanstalk 用 AWS CloudFormation 來啟動環境中的資源並傳播組態變更。資源定義於範本中，您可在 AWS CloudFormation 主控台中檢視此範本。
- 網域名稱 – 會路由到您 Web 應用程式的網域名稱，其格式為 *subdomain*.*region*.elasticbeanstalk.com。

> **Note**
>
> 為了增強 Elastic Beanstalk 應用程式的安全性，我們會在公共后缀列表 (PSL) 中註冊網域 elasticbeanstalk.com。為了加強安全性，如果您需要在 Elastic Beanstalk 應用程式的預設網域名稱中設定敏感性 Cookie，我們建議您使用具 `__Host-` 前置詞的 Cookie。此做法將有助於保護您的網域免受跨站請求偽造 (CSRF) 攻擊。如需更多資訊，請參閱 Mozilla 開發人員網路中的設定 Cookie 頁面。

步驟 2：探索您的環境

若要查看 Elastic Beanstalk 應用程式環境的概觀，請使用 Elastic Beanstalk 主控台中的 Environment overview (環境概觀) 頁面。

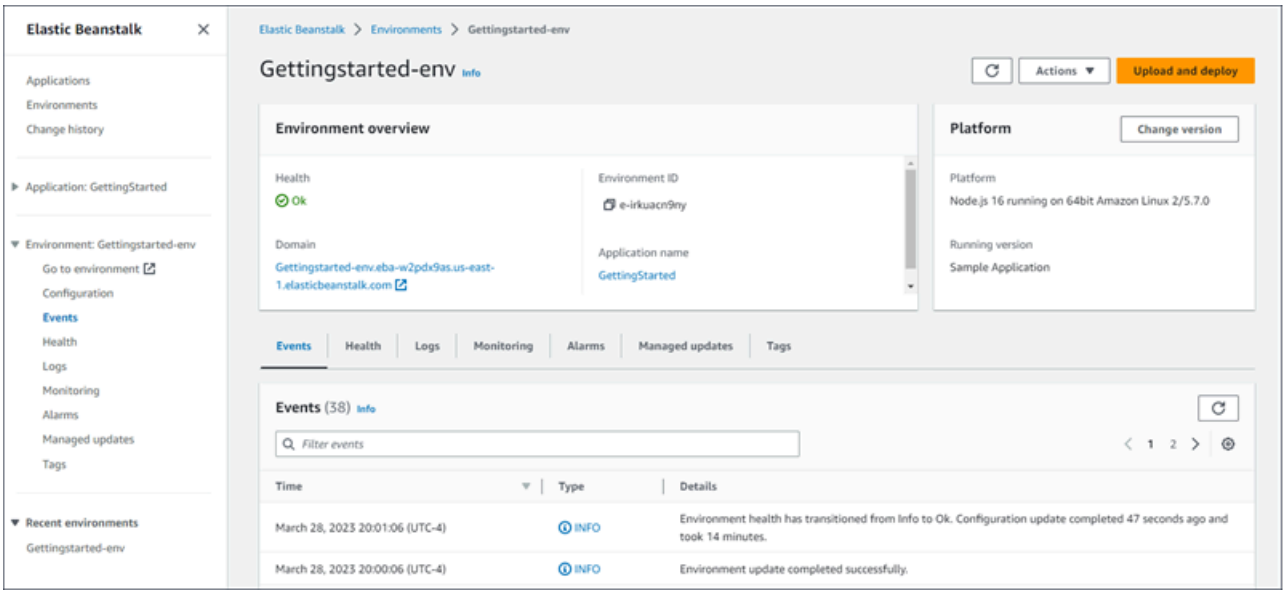
檢視環境概觀

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

Environment overview (環境概觀) 頁面的上部顯示您環境的最上層資訊。包括環境的名稱、網域 URL、目前運作狀態、目前部署的應用程式版本名稱，以及應用程式執行所在的平台版本。在概觀窗格下方，您可以在 Events (事件) 索引標籤中看到最近的環境事件。其他索引標籤顯示與您環境有關的其他主要詳細資訊。

若要進一步了解環境層、平台、應用程式版本和其他 Elastic Beanstalk 概念，請參閱 概念。



Elastic Beanstalk 會建立您的 AWS 資源並啟動應用程式時，環境處於狀態。`Pending`啟動事件的狀態訊息會持續新增到概觀中。

環境的 Domain (網域) 或 URL 位於 Environment overview (環境概觀) 頁面的上部、環境 Health (運作狀態) 下方。這是環境所執行之 Web 應用程式的 URL。選擇此 URL 來取得範例應用程式的「Congratulations (組態)」頁面。左側的瀏覽窗格會列出一個 Go to environment (前往環境) 的連結，可啟動相同的應用程式頁面。

左側導覽窗格中也列出 Configuration (組態)，顯示「組態概觀」頁面。此頁面顯示環境組態選項值的摘要，並依類別分組。

頁面下半部分顯示的索引標籤含有更多關於環境的詳細資訊，並可讓您存取其他功能：

- Events (事件) – 顯示此環境所用資源之 Elastic Beanstalk 服務和其他服務的資訊或錯誤訊息。

- Health (運作狀態) – 顯示執行您應用程式之 Amazon EC2 執行個體的狀態和相關的詳細運作狀態資訊。
- Logs (日誌) – 從您環境中的 Amazon EC2 擷取和下載日誌。您可以擷取完整的日誌或最近的活動。擷取到的日誌的可用時間為 15 分鐘。
- Monitoring (監控) – 顯示環境的統計資料，例如平均延遲和 CPU 使用率。
- Alarms (警示) – 顯示您設定的環境指標警示。您可以在此頁面上新增、修改或刪除警示。
- Managed updates (受管更新) – 顯示即將到來和已完成之受管平台更新與執行個體替換的相關資訊。
- Tags (標籤) – 顯示環境標籤並允許您管理它們。標籤是套用至您環境的鍵/值對。

> **Note**
>
> 主控台左側的導覽窗格列出連結以及與索引標籤相同的名稱。選取其中任何一個連結將會顯示相應索引標籤的內容。

步驟 3：部署新版的應用程式

您可能需要定期部署新版的應用程式。只要您的環境中沒有其他正在進行的更新操作，您隨時都可以部署新版本。

啟動本教學的應用程式版本稱為 Sample Application。

若要更新您的應用程式版本。

1. 下載符合您環境平台的範例應用程式。使用下列應用程式之一。

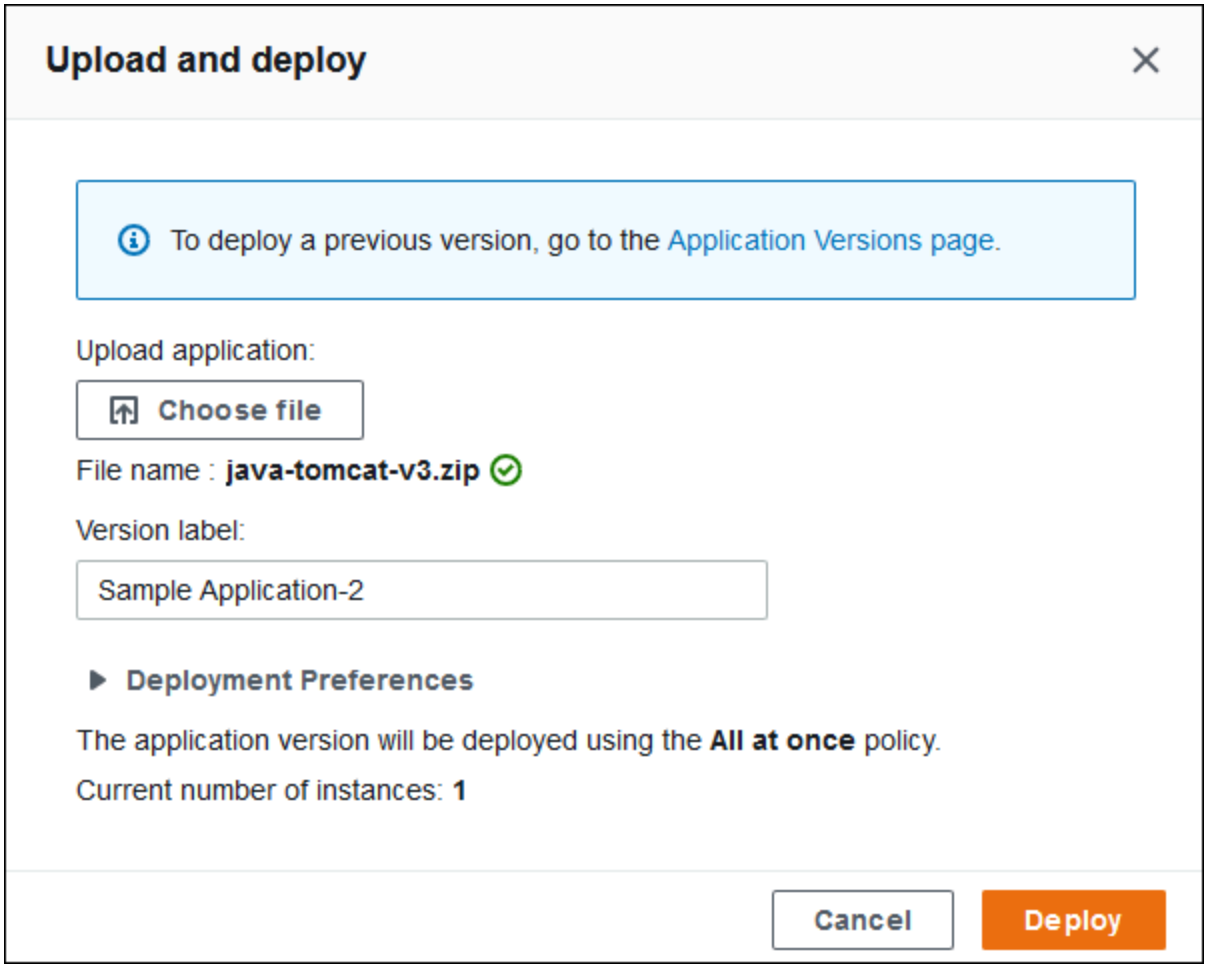
   - Docker – docker.zip
   - 多容器泊塢視窗 — 2.zip docker-multicontainer-v
   - 預配置碼頭工人（玻璃魚) -1.zip docker-glassfish-v
   - Go – go.zip
   - Corretto – corretto.zip
   - Tomcat – tomcat.zip
   - dotnet-core-linux. NET 核心
   - . NET 核心-dotnet-asp-windows. 郵編
   - Node.js – nodejs.zip

- PHP – php.zip
- Python – python.zip
- Ruby – ruby.zip

2. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
3. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

4. 在環境概觀頁面上，選擇 Upload and deploy (上傳和部署)。
5. 選擇 Choose file (選擇檔案)，然後上傳您下載的範例應用程式原始碼套件。



主控台會在 Version label (版本標籤) 中自動填入新的唯一標籤。如果您輸入自己的版本標籤，請確定它是唯一的。

6. 選擇部署。

當 Elastic Beanstalk 將您的檔案部署到您的 Amazon EC2 執行個體時，您可在環境的概觀中檢視部署狀態。Environment Health (環境運作狀態) 的狀態在應用程式版本更新時為灰色。部署完成時，Elastic Beanstalk 會執行應用程式運作狀態檢查。當應用程式回應運作狀態檢查時，會視為狀態良好並回到綠色狀態。環境概觀會顯示新的 Running Version (正在執行的版本) — 這是您提供做為 Version label (版本標籤) 的名稱。

Elastic Beanstalk 也會上傳您的新應用程式版本，並將它新增到應用程式版本的資料表中。若要檢視表格，請getting-started-app在導覽窗格中選擇 [應用程式版本]。

步驟 4：設定您的環境

您可以設定環境，使它更切合您的應用程式。例如，如果您有運算密集型的應用程式，可以針對執行您應用程式的 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體，來變更其類型。為了套用組態變更，Elastic Beanstalk 會執行環境更新。

有些組態變更的操作簡單而且快速。某些變更需要刪除並重新建立 AWS 資源，這可能需要幾分鐘的時間。當您變更組態設定時，Elastic Beanstalk 會警告您可能的應用程式停機時間。

建立組態變更

在此組態變更範例中，您會編輯您環境的容量設定。您設定一個負載平衡、可擴展的環境，在其 Auto Scaling 群組中有介於兩個與四個間的 Amazon EC2 執行個體，然後確認變更已生效。Elastic Beanstalk 會建立額外的 Amazon EC2 執行個體，並新增至最初建立的單一執行個體。然後，Elastic Beanstalk 將這兩個執行個體與環境負載平衡器產生關聯。最後，改善應用程式的回應能力，且時提升可用性。

變更環境的容量

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇組態。

4. 在 Instance traffic and scaling (執行個體流量和擴展) 組態類別中，選擇 Edit (編輯)。
5. 收合 Instances (執行個體) 區段，以便更輕鬆地查看 Capacity (容量) 區段。在 Auto Scaling group (Auto Scaling 群組) 中，將 Environment type (環境類型) 變更為 Load balanced (負載平衡)。
6. 在 Instances (執行個體) 資料列中，將 Max (上限) 變更為 **4**，Min (下限) 變更為 **2**。
7. 若要儲存變更，請選擇頁面底部的儲存變更。
8. 警告會告訴您此更新會取代您目前所有的執行個體。選擇確認。
9. 將會顯示 Environment overview (環境概觀) 頁面，並顯示 Events (事件) 索引標籤。

   環境更新可能需要幾分鐘的時間。若要確認其是否完成，請在事件清單中尋找 Successfully deployed new configuration to environment (新組態已成功部署至環境) 事件。這確認了 Auto Scaling 的執行個體最低計數已設定為 2。Elastic Beanstalk 會自動啟動第二個執行個體。

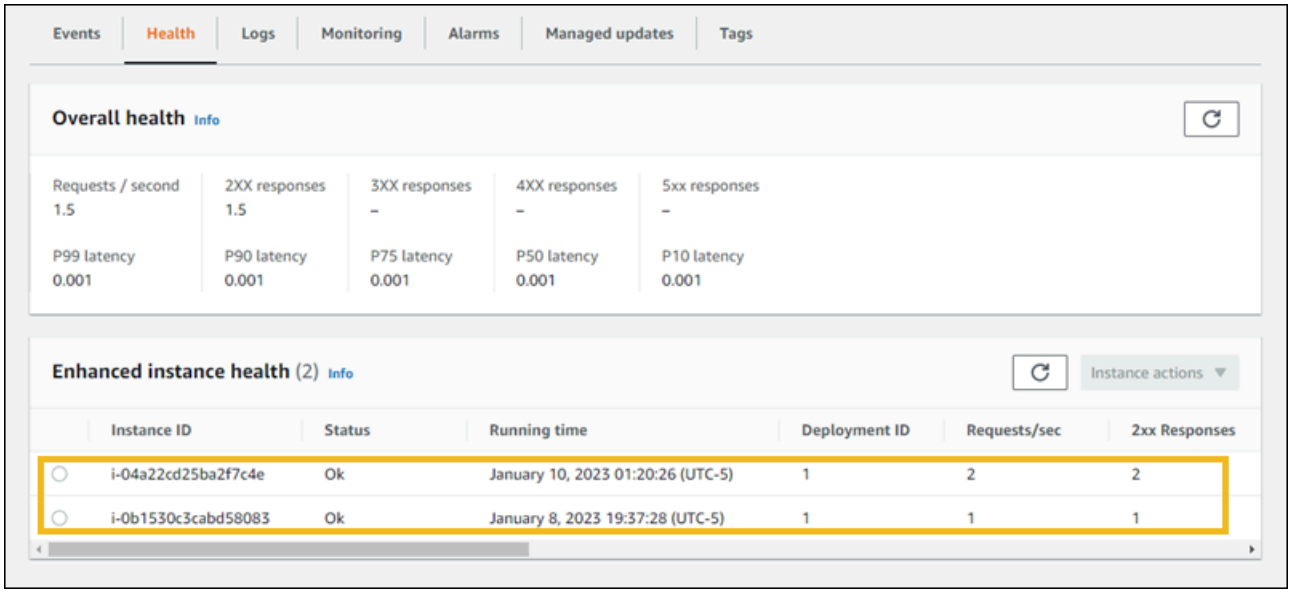
確認組態變更

當環境更新已完成且環境準備就緒時，請確認您的變更。

確認增加的容量

1. 從索引標籤清單或左側導覽窗格中選擇 Health (運作狀態)。
2. 查看 Enhanced instance health (增強型執行個體運作狀態) 區段。

   您可以看到列出兩個 Amazon EC2 執行個體。您的環境容量已提高為兩個執行個體。



步驟 5：清除

恭喜您！您已成功將範例應用程式部署到 AWS 雲端、上傳新版本，並修改其設定以新增第二個 Auto Scaling 執行個體。為確保不會支付任何未使用的服務費用，請刪除所有應用程式版本並終止環境。這也會刪除環境為您建立的 AWS 資源。

刪除應用程式和所有相關聯的資源

1. 刪除所有的應用程式版本。
   a. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
   b. 在功能窗格中，選擇 [應用程式]，然後選擇getting-started-app。
   c. 在導覽窗格中，找到應用程式名稱並選擇 Application versions (應用程式版本)。
   d. 在 Application versions (應用程式版本) 頁面上，選取想要刪除的所有應用程式版本。
   e. 選擇動作，然後選擇刪除。
   f. 打開 Delete versions from Amazon S3 (從 Amazon S3 刪除版本)。
   g. 選擇 Delete (刪除)，然後選擇 Done (完成)。
2. 終止環境。
   a. 在瀏覽窗格中，選擇 getting-started-app，然後在環境清單中選擇 GettingStartedApp-env。
   b. 選擇 Actions (動作)，然後選擇 Terminate Environment (終止環境)。
   c. 輸入環境名稱，確認您要終止 GettingStartedApp-env，然後選擇 [終止]。
3. 刪除 getting-started-app 應用程式。
   a. 在導覽窗格中，選擇getting-started-app。
   b. 選擇 Actions (動作)，然後選擇 Delete application (刪除應用程式)。
   c. 輸入應用程式名稱，確認您要刪除 getting-started-app，然後選擇 [刪除]。

後續步驟

現在您已了解如何建立 Elastic Beanstalk 應用程式和環境，我們建議您閱讀 概念。本主題提供有關 Elastic Beanstalk 元件及架構的資訊，並說明 Elastic Beanstalk 應用程式的重要設計考量。

除了 Elastic Beanstalk 主控台之外，您也可以使用下列工具來建立和管理 Elastic Beanstalk 環境。

EB CLI

EB CLI 是一項命令列工具，可用來建立和管理環境。如需詳細資訊，請參閱 使用 Elastic Beanstalk 命令列界面 (EB CLI)。

AWS SDK for Java

提 AWS SDK for Java 供您可用來建置使用 AWS 基礎結構服務之應用程式的 Java API。使用 AWS SDK for Java，您可以在幾分鐘內開始使用單一、可下載的套件，其中包括 AWS Java 程式庫、程式碼範例和文件。

AWS SDK for Java 需要 J2SE 開發套件 5.0 或更新版本。您可從 http://developers.sun.com/downloads/ 下載最新 Java 軟體。軟體開發套件還需要 Apache Commons (Codec、HTTPClient 和記錄) 和 Saxon-HE 第三方套件，這些套件包含在軟體開發套件的第三方目錄中。

如需詳細資訊，請參閱適用於 Java 的AWS 開發套件。

AWS Toolkit for Eclipse

AWS Toolkit for Eclipse 這是一個開放原始碼外掛程式，適用於日蝕 Java IDE。您可以使用它來建立預先設定的 AWS Java Web 專案 AWS SDK for Java，然後將 Web 應用程式部署到 Elastic Beanstalk。Elastic Beanstalk 外掛程式是建置於 Eclipse Web 工具平台 (WTP) 上。此工具組提供了 Travel Log 範例 Web 應用程式的範本，此範本示範如何使用 Amazon S3 和 Amazon SNS。

為確保您擁有全部的 WTP 相依性，建議您從 Eclipse 的 Java EE 分佈開始。您可以從 http://eclipse.org/downloads/ 下載。

如需使用適用於 Eclipse 的 Elastic Beanstalk 外掛程式詳細資訊，請參閱 AWS Toolkit for Eclipse。若要開始使用 Eclipse 來建立您的 Elastic Beanstalk 應用程式，請參閱在 Elastic Beanstalk 建立和部署 Java 應用程式。

AWS SDK for .NET

AWS SDK for .NET 可讓您建置使用 AWS 基礎結構服務的應用程式。使用 AWS SDK for .NET，您可以在幾分鐘內開始使用單一、可下載的套件，其中包括 AWS .NET 程式庫、程式碼範例和文件。

如需詳細資訊，請參閱適用於 .NET 的AWS 開發套件。如需支援的 .NET Framework 與 Visual studio 版本，請參閱《AWS SDK for .NET 開發人員指南》。

AWS Toolkit for Visual Studio

使用 AWS Toolkit for Visual Studio 外掛程式，您可以將現有的 .NET 應用程式部署到 Elastic Beanstalk。您也可以使用預先配置的 AWS 範本來建立專案。 AWS SDK for .NET

如需必要條件與安裝的資訊，請參閱 AWS Toolkit for Visual Studio。若要開始使用 Visual Studio 來建立您的 Elastic Beanstalk 應用程式，請參閱在 Elastic Beanstalk 上創建和部署 .NET 應用程序。

AWS Node.js JavaScript 中適用的軟體套件

Node.js JavaScript 中的 AWS SDK 可讓您在 AWS 基礎結構服務之上建置應用程式。使用 Node.js JavaScript 中的 AWS SDK，您可以在幾分鐘內使用包含 AWS Node.js 程式庫、程式碼範例和文件的單一可下載套件，在幾分鐘內開始使用。

如需詳細資訊，請參閱 Node.js JavaScript 中的適用AWS 軟體開發套件。

AWS SDK for PHP

AWS SDK for PHP 可讓您在 AWS 基礎結構服務之上建置應用程式。使用 AWS SDK for PHP，您可以在幾分鐘內開始使用單一、可下載的套件，其中包括 AWS PHP 程式庫、程式碼範例和文件。

AWS SDK for PHP 需要 PHP 5.2 或更新版本。如需下載詳細資訊，請參閱 http://php.net/。

如需詳細資訊，請參閱適用於 PHP 的AWS 開發套件。

AWS SDK for Python (Boto)

使用 AWS SDK for Python (Boto)，您可以在幾分鐘內開始使用單一、可下載的套件，其中包括 AWS Python 程式庫、程式碼範例和文件。您可以在 API 上建置 Python 應用程式，讓 web 服務界面的程式碼撰寫不再如此複雜。

該 all-in-one 庫提供了 Python 開發人員友好的 API，它隱藏了許多與 AWS 雲編程相關的較低級任務，包括身份驗證，請求重試和錯誤處理。軟體開發套件提供了 Python 中的實用範例，示範如何使用程式庫來建置應用程式。

如需 Boto、範例程式碼、文件、工具和其他資源的資訊，請參閱 Python 開發人員中心。

AWS SDK for Ruby

您可以在幾分鐘內開始使用單一、可下載的套件完整的 AWS Ruby 程式庫、程式碼範例和文件。您可以在 API 上建置 Ruby 應用程式，讓 Web 服務介面的程式碼撰寫不再如此複雜。

該 all-in-one 庫提供了 Ruby 開發人員友好的 API，它隱藏了許多與 AWS 雲編程相關的較低級任務，包括身份驗證，請求重試和錯誤處理。開發套件提供了 Ruby 中的實用範例，示範如何使用程式庫來建置應用程式。

如需開發套件、範例程式碼、文件、工具和其他資源的資訊，請參閱 Ruby 開發人員中心。

Elastic Beanstalk 概念

AWS Elastic Beanstalk 可讓您將執行應用程式的所有資源當做環境來管理。以下是一些關鍵的 Elastic Beanstalk 概念。

應用

Elastic Beanstalk「應用程式」為 Elastic Beanstalk 元件的邏輯集合，包括「環境」、「版本」和「環境資訊」。在 Elastic Beanstalk 中，應用程式的概念與資料夾類似。

應用程式版本

在 Elastic Beanstalk 中，「應用程式版本」為 Web 應用程式可部署程式碼的特定、具標記的反覆項目。應用程式版本會指向 Amazon Simple Storage Service (Amazon S3) 物件，其中包含諸如 Java WAR 檔案的可部署程式碼。應用程式版本是應用程式的一部分，一個應用程式可以具備多個版本，而各個版本都是唯一的。在執行環境中，您可以部署任何您已經上傳至應用程式的應用程式版本，或上傳新的應用程式版本並立即部署。您可以上傳多個應用程式版本，針對您的 Web 應用程式，測試不同版本的差異。

環境

環境為執行某個應用程式版本的 AWS 資源的集合。每個環境一次只會執行一個應用程式版本，然而，您可以同時在許多環境中執行相同應用程式版本或不同應用程式版本。當您建立環境時，Elastic Beanstalk 會佈建所需資源，以執行您指定的應用程式版本。

環境層

當您啟動 Elastic Beanstalk 環境時，必須先選擇環境層。應用程式層會指定環境執行的應用程式類型，並判斷 Elastic Beanstalk 佈建的資源以支援它。處理 HTTP 請求的應用程式會執行於 Web 伺服器環境層。自 Amazon Simple Queue Service (Amazon SQS) 佇列提取任務的後端環境，則執行於工作者環境層。

環境資訊

環境資訊會針對定義環境及其相關聯資源行為的參數和設定，辨識其集合。當您更新環境資訊設定時，Elastic Beanstalk 會自動將變更套用至現有資源，或加以刪除並部署新的資源 (依變更類型而異)。

已儲存的組態

已儲存的組態為一個範本，可做為建立獨特環境資訊的起點。您可以建立和修改已儲存的組態，並使用 Elastic Beanstalk 主控台、EB CLI、AWS CLI 或 API 將組態套用至環境。API 和 AWS CLI 會參考已儲存的組態，稱為組態範本。

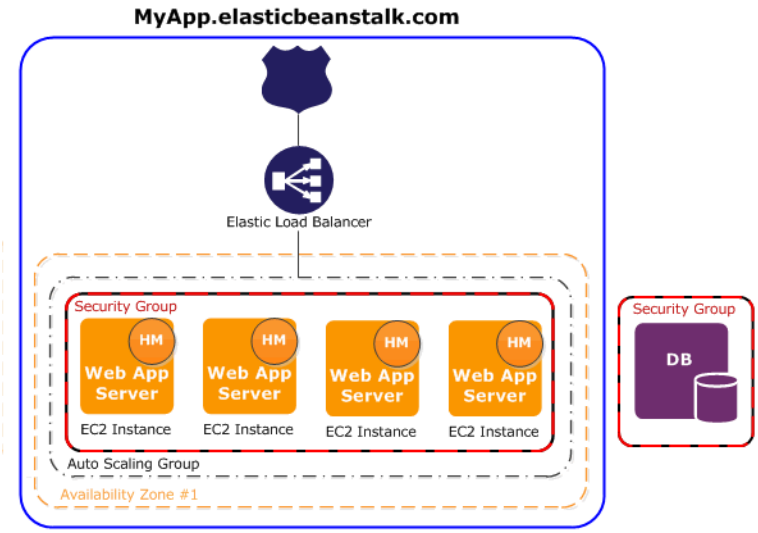
平台

「平台」是作業系統、程式語言執行時間、Web 伺服器、應用程式伺服器及 Elastic Beanstalk 元件的組合。您可以設計 Web 應用程式並將其定位到平台上。Elastic Beanstalk 提供各種平台讓您建置應用程式。

如需詳細資訊，請參閱「Elastic Beanstalk 平台」。

Web 伺服器環境

下圖顯示 Web 伺服器環境層的範例 Elastic Beanstalk 架構，並解釋這類環境層內的元件如何共同運作。



環境是應用程式的核心。在圖表中，環境會顯示在最上方的實線內。當您建立環境時，Elastic Beanstalk 會佈建執行應用程式的所需資源。為環境建立的 AWS 資源包括一個 Elastic Load Balancer (圖中的 ELB)、Auto Scaling 群組及一或多個 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體。

每個環境都具備指向負載平衡器的 CNAME (URL)。本環境即具備 URL (如 `myapp.us-west-2.elasticbeanstalk.com`)。此 URL 會使用 CNAME 記錄，在 Amazon

Route 53 中獲得一個 Elastic Load Balancing URL 別名 (像是 `abcdef-123456.us-west-2.elb.amazonaws.com`)。Amazon Route 53 是一種可用性高、可擴展性強的網域名稱系統 (DNS) Web 服務。其能夠安全可靠的路由至您的基礎設施。您向 DNS 供應商註冊的網域名稱，會將請求轉送至 CNAME。

負載平衡器位於 Amazon EC2 執行個體前方，屬於 Auto Scaling 群組的一部分。Amazon EC2 Auto Scaling 會自動啟動額外的 Amazon EC2 執行個體，以容納您的應用程式增加的負載。若您的應用程式負載減少，Amazon EC2 Auto Scaling 會停止執行個體，但永遠會保留至少一個執行個體繼續執行。

執行於 Amazon EC2 執行個體的軟體堆疊，依容器類型而異。容器類型會定義用於該環境的基礎設施拓撲和軟體堆疊。例如，搭配 Apache Tomcat 容器的 Elastic Beanstalk 環境，會使用 Amazon Linux 作業系統、Apache Web 伺服器和 Apache Tomcat 軟體。如需支援的容器類型清單，請參閱 支援 Elastic Beanstalk 的平台。執行您應用程式的各個 Amazon EC2 執行個體，都會使用這些容器類型之一。此外，名為「主機管理員 (HM)」的軟體元件，會執行於各個 Amazon EC2 執行個體。主機管理員負責下列事項：

- 部署應用程式
- 彙整事件和指標供主控台、API 或命令列進行擷取
- 產生執行個體層級事件
- 監控應用程式日誌檔案是否出現重要錯誤
- 監控應用程式伺服器
- 修補執行個體元件
- 輪動應用程式的日誌檔案，並將其發佈至 Amazon S3

主機管理員會報告指標、錯誤和事件，以及伺服器執行個體狀態，這些可透過 Elastic Beanstalk 主控台、API 和 CLI 取得。

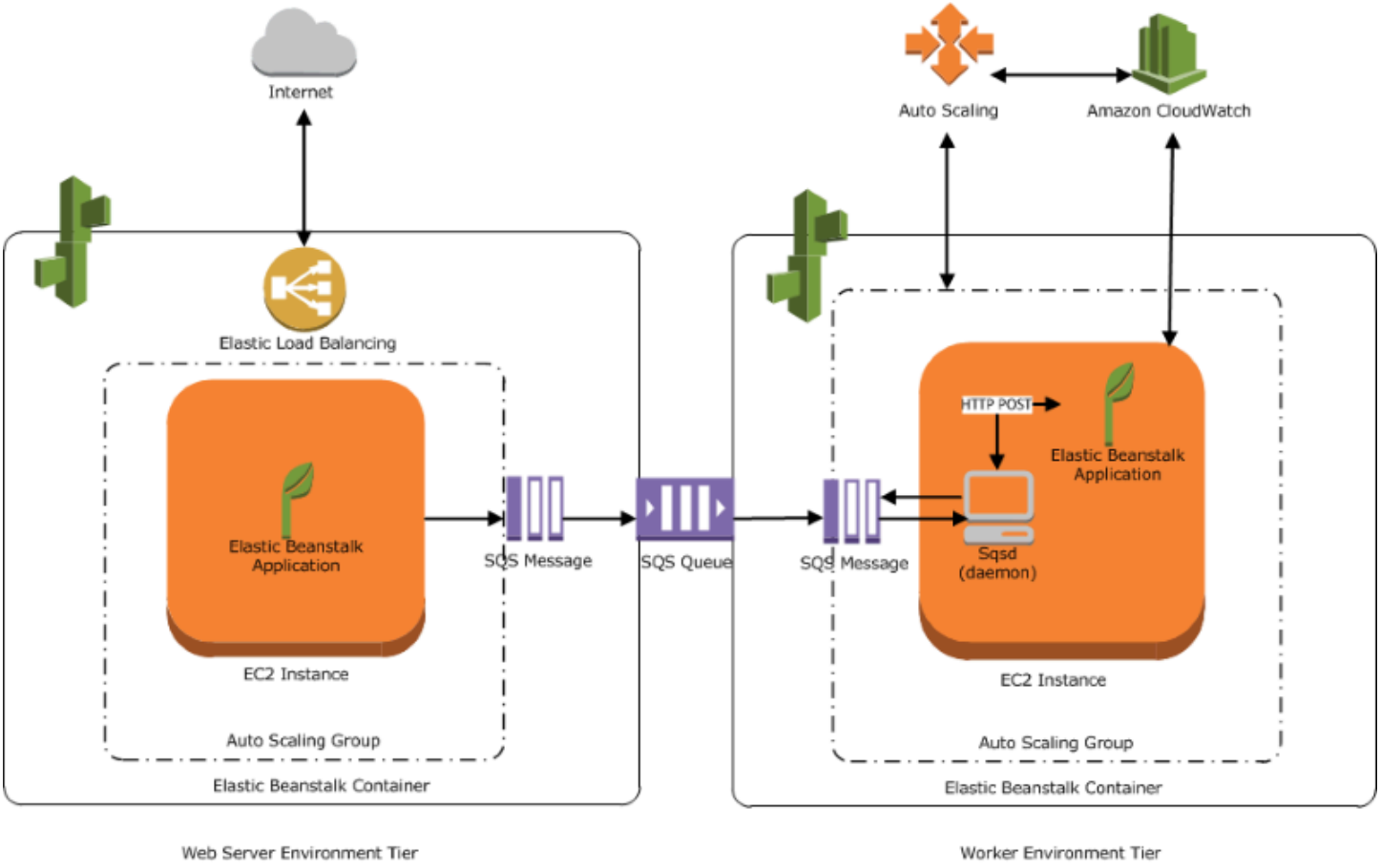
圖中的 Amazon EC2 執行個體均屬於同一「安全群組」。安全群組會定義您的執行個體的防火牆規則。Elastic Beanstalk 預設會定義安全群組，讓所有人都可使用連接埠 80 (HTTP) 進行連線。您可定義多個安全群組。例如，您可為您的資料庫伺服器定義一個安全群組。如需有關 Amazon EC2 安全群組及其設定方式供 Elastic Beanstalk 應用程式使用的詳細資訊，請參閱安全群組。

工作者環境

針對工作者環境層建立的 AWS 資源包括 Auto Scaling 群組、一或多個 Amazon EC2 執行個體和 IAM 角色。對於工作者環境層，Elastic Beanstalk 也會建立和佈建 Amazon SQS 佇列 (如果您還沒有)。

啟動工作者環境層時，Elastic Beanstalk 會依據你所選的程式設計語言安裝必要支援檔案，並在 Auto Scaling 群組的各個 EC2 執行個體安裝協助程式。協助程式會從 Amazon SQS 佇列讀取訊息。協助程式會把從每個訊息讀取的資料，傳送至工作者環境中執行的 Web 應用程式，以進行處理。若您的工作者環境擁有多個執行個體，各個執行個體各自都有協助程式，但一律自相同的 Amazon SQS 佇列讀取資料。

下圖說明橫跨環境和 AWS 服務的不同元件及其互動情況。



Amazon CloudWatch 會用於警示及運作狀態監控。如需詳細資訊，請前往 基礎型運作狀態報告。

如需工作者環境層的運作詳細資訊，請參閱 Elastic Beanstalk 工作者環境。

設計考量

由於要使用在 AWS 雲端端資源執行的 AWS Elastic Beanstalk 來部署應用程式，因此在最佳化應用程式時，必須考量多項組態因素：可擴展性、安全性、持久性儲存、容錯能力、內容交付、軟體更新與修補和連線能力。本主題分別介紹其中的每一項因素。如需 AWS 技術白皮書的完整清單 (涵蓋架構、安全性和成本效益等主題)，請參閱 AWS 雲端運算白皮書。

可擴展性

在物理硬體環境中運作時，與雲端環境相比，您可以透過兩種方法實現可擴展性。您可以透過縱向擴展達成垂直擴展，也可以透過橫向擴展達成水平擴展。垂直擴展方法需要您投資強大的硬體，以支援不斷增長的業務需求。水平擴展方法需要您遵循分散投資模式。因此，您的硬體與應用程式的採購可以更切合目標、資料集能夠整合，而且設計能夠成為服務導向。垂直擴展方法的成本可能極為昂貴，而且也有可能產生需求增長超過容量的風險。在這方面，水平擴展方法通常更有效。不過，使用水平擴展方法時，您必須能夠定期預測需求，並以區塊形式部署基礎設施，以滿足需求。因此，這種方法通常會產生未使用的容量，而且可能需要小心仔細地監控。

只要遷移至雲端，您就可以充分利用雲端的彈性，讓基礎設施與需求順利保持一致。彈性有助於簡化資源的取得和釋出。這樣一來，您的基礎設施可以隨需求的變動快速縮減和擴展。使用時，請將您的 Auto Scaling 設定調整為根據您環境中資源的指標進行擴展或縮減。例如，您可以設定伺服器使用率或網路 I/O 等指標。Auto Scaling 可根據使用量增減，自動新增或移除運算容量。您可以將系統指標 (例如 CPU、記憶體、磁碟 I/O 和網路 I/O) 發佈到 Amazon CloudWatch，然後使用 CloudWatch 設定警示以觸發 Auto Scaling 動作，或根據這些指標傳送通知。如需關於如何設定 Auto Scaling 的指示，請參閱適用於您 Elastic Beanstalk 環境的 Auto Scaling 群組。

此外，我們建議您使用鬆耦合的容錯性元件 (可視需要水平擴展)，盡可能將所有 Elastic Beanstalk 應用程式設計為無狀態。如需深入了解如何設計適用於 AWS 的可擴展應用程式架構，請參閱 AWS Well-Architected 架構。

安全性

AWS 上的安全性是共同的責任。Amazon Web Services 會保護您環境中的實體資源，並確保雲端是一個執行應用程式的安全之處。您需負責進出 Elastic Beanstalk 環境的資料的安全性，和應用程式的安全性。

請設定 SSL 以保護您的應用程式和用戶端之間的資訊。若要設定 SSL，您需要 AWS Certificate Manager (ACM) 提供的免費憑證。如果您已擁有外部憑證授權單位 (CA) 的憑證，您可以使用 ACM 匯入該憑證，也可以使用 AWS CLI 匯入。

如果您的 AWS 區域沒有提供 ACM，您可以從外部 CA 購買憑證，例如 VeriSign 或 Entrust。接著，使用 AWS Command Line Interface (AWS CLI) 將第三方簽署或自我簽署的憑證和私有金鑰上傳到 AWS Identity and Access Management (IAM)。該憑證的公有金鑰會對瀏覽器驗證您的伺服器。它也可用於建立共用工作階段金鑰的基礎，其可為兩方加密資料。關於如何建立、上傳和指派 SSL 憑證至您的環境，相關指示請參閱為您的 Elastic Beanstalk 環境設定 HTTPS。

當您為環境設定 SSL 憑證時，在用戶端與您環境的 Elastic Load Balancing 負載平衡器之間傳送的資料會加密。根據預設，加密動作會終止於負載平衡器，而負載平衡器與 Amazon EC2 執行個體之間的傳輸資料是未加密的。

持久性儲存

Amazon EC2 執行個體上所執行的 Elastic Beanstalk 應用程式，不具備持久性本機儲存功能。當 Amazon EC2 執行個體終止時，不會儲存本機檔案系統。新的 Amazon EC2 執行個體會以預設檔案系統啟動。我們建議您將應用程式設定為將資料儲存於持久的資料來源中。AWS 提供多種持久性儲存服務，可供您的應用程式使用。下表列出了這些版本。

儲存服務	服務文件	Elastic Beanstalk 整合
Amazon S3	Amazon Simple Storage Service 文件	將 Elastic Beanstalk 與 Amazon S3 搭配使用
Amazon Elastic File System	Amazon Elastic File System 文件	搭配 Amazon Elastic File System 使用 Elastic Beanstalk
Amazon Elastic Block Store	Amazon Elastic Block Store 功能指南：Elastic Block Store	
Amazon DynamoDB	Amazon DynamoDB 文件	搭配 Amazon DynamoDB 使用 Elastic Beanstalk
Amazon Relational Database Service (RDS)	Amazon Relational Database Service 文件	搭配 Amazon RDS 使用 Elastic Beanstalk

> **Note**
>
> Elastic Beanstalk 會建立一個 webapp 使用者，您可以將其設定為 EC2 執行個體上應用程式目錄的擁有者。對於 2022 年 2 月 3 日起發行的 Amazon Linux 2 平台版本，Elastic Beanstalk 會在新環境中為 webapp 使用者指派值為 900 的 uid (使用者 ID) 和 gid (群組 ID)。平台版本更新後，對現有環境也會執行相同的操作。此方法可讓 webapp 使用者對永久檔案系統儲存的存取許可保持一致。

雖然不太可能，但如有其他使用者或程序已經使用 900，作業系統會將 webapp 使用者 uid 和 gid 預設為其他值。請在您的 EC2 執行個體上執行 Linux 命令 id webapp，對指派給 webapp 使用者的 uid 和 gid 值進行驗證。

容錯能力

根據經驗，在設計雲端的架構時，您應採用最壞打算的做法。請充分利用架構提供的彈性。在設計，建置和部署時，一律都要能從故障自動復原。針對您的 Amazon EC2 執行個體和 Amazon RDS，使用多個可用區域。可用區域在概念上如同邏輯資料中心。使用 Amazon CloudWatch，來針對您 Elastic Beanstalk 應用程式的健全狀況取得更高的能見度，以及在硬體故障或效能降低時採取適當的動作。請進行您的 Auto Scaling 設定，來將 Amazon EC2 執行個體叢集維持於固定的大小，以讓新的 Amazon EC2 執行個體取代不健全的 Amazon EC2 執行個體。如果您使用的是 Amazon RDS，請設定備份的保留期，讓 Amazon RDS 可以執行自動備份作業。

內容交付

當使用者連線到您的網站，其請求可能會透過眾多的個別網路轉傳。因此，使用者可能會因為高度的延遲而體驗到低落的效能。Amazon CloudFront 可將您的 Web 內容 (例如影像和影片) 分送到世界各地節點網路的各處，進而協助您改善延遲的問題。使用者的請求會轉到最近的邊緣位置，讓內容能夠以最佳效能發佈。CloudFront 可與 Amazon S3 完美搭配運作，後者會持久地儲存您檔案的原始最終版本。如需 Amazon CloudFront 的詳細資訊，請參閱 Amazon CloudFront 開發人員指南。

軟體更新與修補

AWS Elastic Beanstalk 會定期發佈平台更新，來提供修正程式，軟體更新和新功能。Elastic Beanstalk 提供了多種處理平台更新的選擇。受管平台更新可以在排定的維護時段將您的環境自動升級至最新的平台版本，同時讓您的應用程式維持運作。在 2019 年 11 月 25 日或之後使用 Elastic Beanstalk 主控台建立的環境中，會盡可能根據預設啟用受管更新。您也可以使用 Elastic Beanstalk 主控台或 EB CLI 手動啟動更新。

連線能力

Elastic Beanstalk 需要連線到您環境中的執行個體，才能完成部署。當您在 Amazon VPC 中部署 Elastic Beanstalk 應用程式時，啟用連線功能所需的組態，取決於您所建立的 Amazon VPC 環境的類型：

- 對於單一執行個體環境，不需額外進行設定。這是因為在這類環境中，Elastic Beanstalk 會指派公有彈性 IP 地址給每個 Amazon EC2 執行個體，而此等地址可以讓執行個體直接與網際網路進行通訊。

- 對於 Amazon VPC 中有負載平衡且可擴展的環境而言 (同時具備公有與私有子網路)，您必須執行以下操作：
  - 在公有子網路中建立負載平衡器，來轉傳從網際網路到 Amazon EC2 執行個體的傳入流量。
  - 建立網路位址轉譯 (NAT) 裝置，來轉傳從私有子網路中的 Amazon EC2 執行個體到網際網路的傳出流量。
  - 針對私有子網路中的 Amazon EC2 執行個體，建立傳入與傳出的轉傳規則。
  - 如果使用 NAT 執行個體，請設定 NAT 執行個體和 Amazon EC2 執行個體適用的安全群組，以啟用網際網路通訊。
- 對於具備一個公有子網路的 Amazon VPC 中負載平衡的可擴展環境，不需額外進行設定。這是因為在這類環境中，您的 Amazon EC2 執行個體設有公有 IP 地址，而此等地址可以讓執行個體直接與網際網路進行通訊。

如需搭配 Amazon VPC 使用 Elastic Beanstalk 的詳細資訊，請參閱搭配 Amazon VPC 使用 Elastic Beanstalk。

服務角色、執行個體描述檔和使用者政策

當您建立環境時，AWS Elastic Beanstalk 會提示您輸入以下 AWS Identity and Access Management (IAM) 角色：

- 服務角色：Elastic Beanstalk 會擔任服務角色，以代表您使用其他 AWS 服務 服務。
- 執行個體設定檔 Elastic Beanstalk 會將執行個體設定檔套用於環境中的執行個體。它可讓執行個體執行下列操作：
  - 從 Amazon Simple Storage Service (Amazon S3) 擷取應用程式版本。
  - 將日誌上傳到 Amazon S3。
  - 執行視環境類型和平台而異的其他任務。

服務角色

當您使用 Elastic Beanstalk 主控台或透過 Elastic Beanstalk EB CLI 建立環境時，會建立所需的服務角色並分配受管政策。這些策略包括所有必要的許可權限。現在，假設您的帳户已有服務角色，然後您使用 Elastic Beanstalk 主控台或使用 Elastic Beanstalk CLI 創立了一個新環境。在這種情況下，現有的服務角色將自動分配給新環境。

執行個體描述檔

如果您的 AWS 帳戶沒有 EC2 執行個體設定檔，您必須使用 IAM 服務建立一個。然後，您可以將 EC2 執行個體設定檔指派給您建立的新環境。建立環境精靈提供可指導您完成 IAM 服務的資訊，以便您建立具有所需許可的 EC2 執行個體設定檔。建立執行個體設定檔後，您就可以返回主控台，將其選為 EC2 執行個體設定檔，然後繼續建立環境的步驟。

> **Note**
>
> 先前，Elastic Beanstalk 在 AWS 帳戶初次建立環境時，建立了一個名稱為 `aws-elasticbeanstalk-ec2-role` 的預設 EC2 執行個體設定檔。此執行個體設定檔包含預設受管政策。如果您的帳戶已經擁有此執行個體設定檔，您仍然可以將其指派給您的環境。
> 不過，最近的 AWS 安全性準則不允許 AWS 服務自動建立具有信任政策的角色給其他 AWS 服務 (在這種情況下為 EC2)。基於這些安全性準則，Elastic Beanstalk 不再建立預設 `aws-elasticbeanstalk-ec2-role` 執行個體設定檔。

使用者政策

除了指派至您環境的角色，您亦可建立使用者政策，並將其套用於您帳戶的 IAM 使用者和群組。套用使用者政策可讓使用者建立並管理 Elastic Beanstalk 應用程式和環境。Elastic Beanstalk 亦可提供完整存取和唯讀存取的受管政策。如需這些政策的詳細資訊，請參閱the section called “使用者政策”。

其他執行個體設定檔和使用者政策

您可以建立自己的執行個體描述檔和使用者政策，供進階情境使用。若您的執行個體須存取的服務未包含在預設政策中，您可以建立新的政策，或是將其他政策新增至預設政策。若受管政策的許可過於寬鬆，您亦可建立更嚴格的使用者政策。如需 AWS 許可的詳細資訊，請參閱《IAM 使用者指南》。

主題

- Elastic Beanstalk 服務角色
- Elastic Beanstalk 執行個體描述檔
- Elastic Beanstalk 使用者政策

Elastic Beanstalk 服務角色

服務角色是 Elastic Beanstalk 代表您呼叫其他服務時所擔任的 IAM 角色。例如，當 Elastic Beanstalk 呼叫 Amazon Elastic Compute Cloud (Amazon EC2)、Elastic Load Balancing 和 Amazon EC2 Auto Scaling API 時，會使用服務角色來收集資訊。Elastic Beanstalk 使用的服務角色是您在建立 Elastic Beanstalk 環境時指定的服務角色。

有兩個連接至服務角色的受管政策：這些政策提供許可，讓 Elastic Beanstalk 能存取建立和管理環境所需的 AWS 資源。一個受管政策提供用於增強型運作狀態監控的受管政策，另一個提供用於受管平台更新所需的其他許可。

`AWSElasticBeanstalkEnhancedHealth`

此政策授予 Elastic Beanstalk 監控環境運作狀態所需的所有許可。亦包含 Amazon SQS 動作，讓 Elastic Beanstalk 能監控工作者環境的佇列活動。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:DescribeInstanceHealth",
                "elasticloadbalancing:DescribeLoadBalancers",
```

```
                "elasticloadbalancing:DescribeTargetHealth",
                "ec2:DescribeInstances",
                "ec2:DescribeInstanceStatus",
                "ec2:GetConsoleOutput",
                "ec2:AssociateAddress",
                "ec2:DescribeAddresses",
                "ec2:DescribeSecurityGroups",
                "sqs:GetQueueAttributes",
                "sqs:GetQueueUrl",
                "autoscaling:DescribeAutoScalingGroups",
                "autoscaling:DescribeAutoScalingInstances",
                "autoscaling:DescribeScalingActivities",
                "autoscaling:DescribeNotificationConfigurations",
                "sns:Publish"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

AWSElasticBeanstalkManagedUpdatesCustomerRolePolicy

此政策會授予許可，讓 Elastic Beanstalk 能夠代表您更新環境以執行受管平台更新。

服務層級許可分組

此政策會根據提供的許可集分組到陳述式中。

- *ElasticBeanstalkPermissions* – 此許可群組用於呼叫 Elastic Beanstalk 服務動作 (Elastic Beanstalk API)。
- *AllowPassRoleToElasticBeanstalkAndDownstreamServices* – 此許可群組允許將任何角色傳遞給 Elastic Beanstalk 和其他下游服務，如 AWS CloudFormation。
- *ReadOnlyPermissions* – 此許可群組用於收集執行中環境的相關資訊。
- **OperationPermissions* – 具有此命名模式的群組用於呼叫必要的操作來執行平台更新。
- **BroadOperationPermissions* – 具有此命名模式的群組用於呼叫必要的操作來執行平台更新。它們也包含支援舊式環境的廣泛許可。
- **TagResource* – 具有此命名模式的群組適用於使用建立時標記 API 的呼叫，以在 Elastic Beanstalk 環境中建立的資源上附加標籤。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ElasticBeanstalkPermissions",
            "Effect": "Allow",
            "Action": [
                "elasticbeanstalk:*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowPassRoleToElasticBeanstalkAndDownstreamServices",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::*:role/*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "elasticbeanstalk.amazonaws.com",
                        "ec2.amazonaws.com",
                        "ec2.amazonaws.com.cn",
                        "autoscaling.amazonaws.com",
                        "elasticloadbalancing.amazonaws.com",
                        "ecs.amazonaws.com",
                        "cloudformation.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Sid": "ReadOnlyPermissions",
            "Effect": "Allow",
            "Action": [
                "autoscaling:DescribeAccountLimits",
                "autoscaling:DescribeAutoScalingGroups",
                "autoscaling:DescribeAutoScalingInstances",
                "autoscaling:DescribeLaunchConfigurations",
                "autoscaling:DescribeLoadBalancers",
                "autoscaling:DescribeNotificationConfigurations",
                "autoscaling:DescribeScalingActivities",
                "autoscaling:DescribeScheduledActions",
                "ec2:DescribeAccountAttributes",
```

```
                "ec2:DescribeAddresses",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeImages",
                "ec2:DescribeInstanceAttribute",
                "ec2:DescribeInstances",
                "ec2:DescribeKeyPairs",
                "ec2:DescribeLaunchTemplates",
                "ec2:DescribeLaunchTemplateVersions",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSnapshots",
                "ec2:DescribeSpotInstanceRequests",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcClassicLink",
                "ec2:DescribeVpcs",
                "elasticloadbalancing:DescribeInstanceHealth",
                "elasticloadbalancing:DescribeLoadBalancers",
                "elasticloadbalancing:DescribeTargetGroups",
                "elasticloadbalancing:DescribeTargetHealth",
                "logs:DescribeLogGroups",
                "rds:DescribeDBEngineVersions",
                "rds:DescribeDBInstances",
                "rds:DescribeOrderableDBInstanceOptions",
                "sns:ListSubscriptionsByTopic"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "EC2BroadOperationPermissions",
            "Effect": "Allow",
            "Action": [
                "ec2:AllocateAddress",
                "ec2:AssociateAddress",
                "ec2:AuthorizeSecurityGroupEgress",
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:CreateLaunchTemplate",
                "ec2:CreateLaunchTemplateVersion",
                "ec2:CreateSecurityGroup",
                "ec2:DeleteLaunchTemplate",
                "ec2:DeleteLaunchTemplateVersions",
                "ec2:DeleteSecurityGroup",
                "ec2:DisassociateAddress",
                "ec2:ReleaseAddress",
```

```
                "ec2:RevokeSecurityGroupEgress",
                "ec2:RevokeSecurityGroupIngress"
            ],
            "Resource": "*"
        },
        {
            "Sid": "EC2RunInstancesOperationPermissions",
            "Effect": "Allow",
            "Action": "ec2:RunInstances",
            "Resource": "*",
            "Condition": {
                "ArnLike": {
                    "ec2:LaunchTemplate": "arn:aws:ec2:*:*:launch-template/*"
                }
            }
        },
        {
            "Sid": "EC2TerminateInstancesOperationPermissions",
            "Effect": "Allow",
            "Action": [
                "ec2:TerminateInstances"
            ],
            "Resource": "arn:aws:ec2:*:*:instance/*",
            "Condition": {
                "StringLike": {
                    "ec2:ResourceTag/aws:cloudformation:stack-id": [
                        "arn:aws:cloudformation:*:*:stack/awseb-e-*",
                        "arn:aws:cloudformation:*:*:stack/eb-*"
                    ]
                }
            }
        },
        {
            "Sid": "ECSBroadOperationPermissions",
            "Effect": "Allow",
            "Action": [
                "ecs:CreateCluster",
                "ecs:DescribeClusters",
                "ecs:RegisterTaskDefinition"
            ],
            "Resource": "*"
        },
        {
            "Sid": "ECSDeleteClusterOperationPermissions",
```

```
            "Effect": "Allow",
            "Action": "ecs:DeleteCluster",
            "Resource": "arn:aws:ecs:*:*:cluster/awseb-*"
        },
        {
            "Sid": "ASGOperationPermissions",
            "Effect": "Allow",
            "Action": [
                "autoscaling:AttachInstances",
                "autoscaling:CreateAutoScalingGroup",
                "autoscaling:CreateLaunchConfiguration",
                "autoscaling:CreateOrUpdateTags",
                "autoscaling:DeleteLaunchConfiguration",
                "autoscaling:DeleteAutoScalingGroup",
                "autoscaling:DeleteScheduledAction",
                "autoscaling:DetachInstances",
                "autoscaling:DeletePolicy",
                "autoscaling:PutScalingPolicy",
                "autoscaling:PutScheduledUpdateGroupAction",
                "autoscaling:PutNotificationConfiguration",
                "autoscaling:ResumeProcesses",
                "autoscaling:SetDesiredCapacity",
                "autoscaling:SuspendProcesses",
                "autoscaling:TerminateInstanceInAutoScalingGroup",
                "autoscaling:UpdateAutoScalingGroup"
            ],
            "Resource": [
                "arn:aws:autoscaling:*:*:launchConfiguration:*:launchConfigurationName/
awseb-e-*",
                "arn:aws:autoscaling:*:*:launchConfiguration:*:launchConfigurationName/
eb-*",
                "arn:aws:autoscaling:*:*:autoScalingGroup:*:autoScalingGroupName/awseb-
e-*",
                "arn:aws:autoscaling:*:*:autoScalingGroup:*:autoScalingGroupName/eb-*"
            ]
        },
        {
            "Sid": "CFNOperationPermissions",
            "Effect": "Allow",
            "Action": [
                "cloudformation:*"
            ],
            "Resource": [
                "arn:aws:cloudformation:*:*:stack/awseb-*",
```

```
                "arn:aws:cloudformation:*:*:stack/eb-*"
            ]
        },
        {
            "Sid": "ELBOperationPermissions",
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:AddTags",
                "elasticloadbalancing:ApplySecurityGroupsToLoadBalancer",
                "elasticloadbalancing:ConfigureHealthCheck",
                "elasticloadbalancing:CreateLoadBalancer",
                "elasticloadbalancing:DeleteLoadBalancer",
                "elasticloadbalancing:DeregisterInstancesFromLoadBalancer",
                "elasticloadbalancing:DeregisterTargets",
                "elasticloadbalancing:RegisterInstancesWithLoadBalancer",
                "elasticloadbalancing:RegisterTargets"
            ],
            "Resource": [
                "arn:aws:elasticloadbalancing:*:*:targetgroup/awseb-*",
                "arn:aws:elasticloadbalancing:*:*:targetgroup/eb-*",
                "arn:aws:elasticloadbalancing:*:*:loadbalancer/awseb-*",
                "arn:aws:elasticloadbalancing:*:*:loadbalancer/eb-*",
                "arn:aws:elasticloadbalancing:*:*:loadbalancer/*/awseb-*/*",
                "arn:aws:elasticloadbalancing:*:*:loadbalancer/*/eb-*/*"
            ]
        },
        {
            "Sid": "CWLogsOperationPermissions",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:DeleteLogGroup",
                "logs:PutRetentionPolicy"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:/aws/elasticbeanstalk/*"
        },
        {
            "Sid": "S3ObjectOperationPermissions",
            "Effect": "Allow",
            "Action": [
                "s3:DeleteObject",
                "s3:GetObject",
                "s3:GetObjectAcl",
                "s3:GetObjectVersion",
```

```
                "s3:GetObjectVersionAcl",
                "s3:PutObject",
                "s3:PutObjectAcl",
                "s3:PutObjectVersionAcl"
            ],
            "Resource": "arn:aws:s3:::elasticbeanstalk-*/*"
        },
        {
            "Sid": "S3BucketOperationPermissions",
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:GetBucketPolicy",
                "s3:ListBucket",
                "s3:PutBucketPolicy"
            ],
            "Resource": "arn:aws:s3:::elasticbeanstalk-*"
        },
        {
            "Sid": "SNSOperationPermissions",
            "Effect": "Allow",
            "Action": [
                "sns:CreateTopic",
                "sns:GetTopicAttributes",
                "sns:SetTopicAttributes",
                "sns:Subscribe"
            ],
            "Resource": "arn:aws:sns:*:*:ElasticBeanstalkNotifications-*"
        },
        {
            "Sid": "SQSOperationPermissions",
            "Effect": "Allow",
            "Action": [
                "sqs:GetQueueAttributes",
                "sqs:GetQueueUrl"
            ],
            "Resource": [
                "arn:aws:sqs:*:*:awseb-e-*",
                "arn:aws:sqs:*:*:eb-*"
            ]
        },
        {
            "Sid": "CWPutMetricAlarmOperationPermissions",
            "Effect": "Allow",
```

```
            "Action": [
                "cloudwatch:PutMetricAlarm"
            ],
            "Resource": [
                "arn:aws:cloudwatch:*:*:alarm:awseb-*",
                "arn:aws:cloudwatch:*:*:alarm:eb-*"
            ]
        },
        {
            "Sid": "AllowECSTagResource",
            "Effect": "Allow",
            "Action": [
                "ecs:TagResource"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "ecs:CreateAction": [
                        "CreateCluster",
                        "RegisterTaskDefinition"
                    ]
                }
            }
        }
    ]
}
```

您可以使用以下任何方式建立 Elastic Beanstalk 環境。每個部分都會說明該方法如何處理服務角色。

Elastic Beanstalk 主控台

如果您使用 Elastic Beanstalk 主控台建立環境，Elastic Beanstalk 會提示您建立名為 `aws-elasticbeanstalk-service-role` 的服務角色。透過 Elastic Beanstalk 建立時，此角色會包含一個能讓 Elastic Beanstalk 擔任該服務角色的信任政策。本主題中先前提到的兩個受管理政策也會連接至角色。

Elastic Beanstalk 命令列界面 (EB CLI)

您可以使用 Elastic Beanstalk 命令列界面 (EB CLI) 的 the section called “eb create” 命令建立環境。如果您未透過 `--service-role` 選項指定服務角色。Elastic Beanstalk 會建立相同的預設服務角色 `aws-elasticbeanstalk-service-role`。若預設服務角色已存在，Elastic Beanstalk 會將其運用於新環境。透過 Elastic Beanstalk 建立時，此角色會包含一個能讓 Elastic Beanstalk 擔任該服務角色的信任政策。本主題中先前提到的兩個受管理政策也會連接至角色。

Elastic Beanstalk API

您可以使用 Elastic Beanstalk API 的 `CreateEnvironment` 動作建立環境。如果您未指定服務角色，Elastic Beanstalk 會建立一個監控服務連結角色。這是服務角色的獨特類型，由 Elastic Beanstalk 預先定義以納入所有該服務代表您呼叫其他 AWS 服務 所需的許可。服務連結角色會您的帳戶建立關聯。Elastic Beanstalk 只會建立一次此角色，然後在建立其他環境時重複使用。您也可以使用 IAM，預先建立您帳戶的監控服務連結角色。您的帳戶擁有監控服務連結角色時，即可使用 Elastic Beanstalk 主控台、Elastic Beanstalk API 或 EB CLI，將其用於建立環境。如需在 Elastic Beanstalk 環境中使用服務連結角色的說明，請參閱 使用 Elastic Beanstalk 的服務連結角色。

如需服務角色的詳細資訊，請參閱管理 Elastic Beanstalk 服務角色。

Elastic Beanstalk 執行個體描述檔

執行個體設定檔是您 Elastic Beanstalk 環境啟動的 Amazon EC2 執行個體所套用的 IAM 角色。建立 Elastic Beanstalk 環境時，您要指定 EC2 執行個體採取下列動作時會使用的執行個體設定檔：

- 從 Amazon Simple Storage Service (Amazon S3) 擷取應用程式版本
- 將日誌寫入 Amazon S3
- 於 AWS X-Ray 整合式環境將除錯資料上傳至 X-Ray
- 在 Amazon ECS 受管 Docker 環境中，使用 Amazon Elastic Container Service (Amazon ECS) 協調容器部署。
- 在工作者環境中，從 Amazon Simple Queue Service (Amazon SQS) 佇列讀取
- 於工作者環境透過 Amazon DynamoDB 執行領導者選擇
- 於工作者環境將執行個體運作狀態指標發佈至 Amazon CloudWatch

Elastic Beanstalk 提供一組受管政策，可讓您環境中的 EC2 執行個體執行所需的作業。基本使用案例所需的受管政策如下。

- `AWSElasticBeanstalkWebTier`
- `AWSElasticBeanstalkWorkerTier`
- `AWSElasticBeanstalkMulticontainerDocker`

當您在 Elastic Beanstalk 主控台首次啟動環境時，您需要將這些政策附加到您所建立的執行個體設定檔。

若您的 Web 應用程式需要存取其他額外的 AWS 服務，請將陳述式或受管政策新增至能夠存取這些服務的執行個體設定檔。

如需有關執行個體描述檔的詳細資訊，請參閱 管理 Elastic Beanstalk 執行個體描述檔。

Elastic Beanstalk 使用者政策

為使用 Elastic Beanstalk 的所有使用者建立 IAM 使用者，避免使用您的根帳戶或分享登入資料。安全的最佳實務做法是僅授予這些使用者存取所需服務和功能的許可。

Elastic Beanstalk 需要的許可不只用於其 API 動作，亦用於其他數個 AWS 服務。Elastic Beanstalk 在環境中啟動資源時需使用使用者許可。這些資源包括 EC2 執行個體、Elastic Load Balancing 負載平衡器和 Auto Scaling 群組。Elastic Beanstalk 亦透過使用者許可，儲存 Amazon Simple Storage Service (Amazon S3) 的日誌和範本、傳送通知至 Amazon SNS、指派執行個體描述檔並發佈指標至 CloudWatch。Elastic Beanstalk 需要 AWS CloudFormation 許可，以協調資源部署和更新。此外，也需要 Amazon RDS 許可來視需要建立資料庫，以及 Amazon SQS 許可來建立工作者環境的佇列。

如需使用者政策的詳細資訊，請參閱管理 Elastic Beanstalk 使用者政策。

Elastic Beanstalk 平台

AWS Elastic Beanstalk 提供各種平台，您可以在其上建立您的應用程式。您針對這些平台來設計 Web 應用程式，而 Elastic Beanstalk 會將您的程式碼部署到您選擇來建立作用中應用程式環境的平台版本。

Elastic Beanstalk 為各種程式設計語言 、應用程式伺服器及 Docker 容器提供平台。部分平台有多個並行支援的版本。

主題

Elastic Beanstalk 平台詞彙表

以下為與 AWS Elastic Beanstalk 平台及其生命週期相關的關鍵詞彙。

執行時間

執行您應用程式碼所需之程式設計語言特定的執行時間軟體 (架構、程式庫、解譯器、vm 等等)。

Elastic Beanstalk 元件

Elastic Beanstalk 在平台上新增的軟體元件可啟用 Elastic Beanstalk 功能。例如，收集和報告運作狀態資訊需要使用增強型運作狀態代理程式。

平台

作業系統 (OS)、執行時間、Web 伺服器、應用程式伺服器和 Elastic Beanstalk 元件的組合。平台提供可供執行應用程式的元件。

平台版本

特定作業系統 (OS) 版本、執行時間、Web 伺服器、應用程式伺服器和 Elastic Beanstalk 元件的組合。您根據平台版本建立 Elastic Beanstalk 環境，並在此環境中部署應用程式。

平台版本的語意版本編號格式為 X.Y.Z，X 表示主要版本、Y 表示次要版本，Z 則是修補程式版本。

平台版本可以是以下其中一個狀態：

- 支援的 – 完全包含支援的元件的平台版本。所有元件尚未達到其各自供應商 (擁有者 – AWS 或第三方 – 或社群) 指定的生命週期結束 (EOL)。它們會收到供應商的定期修補程式或次要更新。Elastic Beanstalk 提供支援的平台版本讓您建立環境。
- 已淘汰 – 平台版本，其中包含一或多個已達供應商指定的生命週期結束 (EOL) 的已淘汰元件。Elastic Beanstalk 環境不向新或現有的客戶提供淘汰的平台版本。

  如需已淘汰元件的詳細資訊，請參閱 the section called “平台支援政策”。

平台分支

一系列平台版本，會共用某些元件的特定 (通常是主要) 版本，例如作業系統 (OS)、執行時間或 Elastic Beanstalk 元件。例如：在 64 位元 Amazon Linux 上執行的 Python 3.6；在 64 位元 Windows Server 2016 上執行的 IIS 10.0。分支中的每個連續平台版本都是前一個版本的更新。

您可以無條件使用每個平台分支中的最新平台版本來建立環境。分支中先前的平台版本仍然受支援如果您在過去 30 天曾在環境中使用過，則可以根據先前的平台版本來建立環境。但是這些先前的平台版本缺少最新的元件，不建議使用。

平台分支可以是以下其中一個狀態：

- 支援的 – 目前的平台分支。它完全由支援的元件組成。它會持續接收平台更新，建議用於生產環境。如需支援的平台分支清單，請參閱 AWS Elastic Beanstalk 平台指南中的 Elastic Beanstalk 支援的平台。

- Beta – 預覽版，發行前平台分支。這本質上是實驗性的。它可能會持續收到平台更新一段時間，但沒有長期支援。不建議在生產環境中使用 beta 平台分支。僅用於評估。如需 beta 平台分支的清單，請參閱 AWS Elastic Beanstalk 平台指南中的公開 Beta 版中的 Elastic Beanstalk 平台版本。
- 已取代的 – 具有一或多個「已取代的元件」的平台分支。它會接收持續的平台更新，但不建議在生產環境中使用。如需已棄用的平台分支清單，請參閱 AWS Elastic Beanstalk 平台指南中的排定淘汰的 Elastic Beanstalk 平台版本。
- 淘汰 – 具有一或多個「淘汰元件」的平台分支。它不再接收平台更新，不建議在生產環境中使用。AWS Elastic Beanstalk 平台指南中未列出淘汰的平台分支。Elastic Beanstalk 不會將淘汰平台分支的平台版本提供給您建立環境。

支援的元件沒有供應商 (擁有者或社群) 排程的淘汰日期。供應商可能是 AWS 或第三方。已棄用元件具有其供應商排程的淘汰日期。已淘汰的元件已達生命週期結束 (EOL)，且其供應商不再支援。如需已淘汰元件的詳細資訊，請參閱 the section called “平台支援政策”。

如果您的環境使用已棄用或已淘汰的平台分支，建議您將其更新為支援的平台分支中的平台版本。如需詳細資訊，請參閱「the section called “平台更新”」。

平台更新

發行的新平台版本包含某些平台元件更新 — 作業系統、執行時間、web 伺服器、應用程式伺服器和 Elastic Beanstalk 元件。平台更新下列語意版本控制分類，可有數個層級：

- 重大更新 – 變更內容與現有平台版本不相容的更新。您可能需要修改應用程式，才能在新版本中正確執行。重大更新有新的主要平台版本編號。
- 次要更新 – 新增功能與現有平台版本回溯相容的更新。您不需要修改應用程式，就能在新的次要版本中正確執行。次要更新有新的次要平台版本編號。
- 修補程式更新 – 包含之維護版本 (錯誤修正、安全更新和效能改善) 與現有平台版本回溯相容的更新。修補程式更新有新的修補程式平台版本編號。

受管更新

自動套用 Elastic Beanstalk 支援平台版本之作業系統 (OS)、執行時間、web 伺服器、應用程式伺服器和 Elastic Beanstalk 元件修補程式和次要更新的 Elastic Beanstalk 功能。受管更新會將相同平台分支中的較新平台版本套用至您的環境。您可以設定受管更新只套用修補程式更新，或同時套用次要與修補程式更新。您也可以完全停用受管更新。

如需更多詳細資訊，請參閱受管平台更新。

Elastic Beanstalk 平台維護的共同責任模型

AWS 我們的客戶有責任達成高度軟體元件的安全性與合規性。此共同模型可降低您的作業負擔。

如需詳細資訊，請參閱 AWS 共同的責任模型。

AWS Elastic Beanstalk 提供受管理的更新功能，協助您執行共用責任模型的一端。此功能會自動套用 Elastic Beanstalk 支援平台版本的修補程式和次要更新。如果受管更新失敗，Elastic Beanstalk 會通知您以確保您知悉此失敗，從而可立即採取行動。

如需詳細資訊，請參閱 受管平台更新。

此外，Elastic Beanstalk 可以執行以下操作：

- 發佈其平台支援政策以及接下來 12 個月的淘汰排程。
- 發行作業系統 (OS)、執行時間、應用程式伺服器和 web 伺服器元件的修補程式、次要和重大更新，可用性一般為 30 天。Elastic Beanstalk 負責建立出現在其支援平台版本之 Elastic Beanstalk 元件的更新。所有其他更新直接來自其供應商 (擁有者或社群)。

我們會在AWS Elastic Beanstalk 版本備註指南的版本備註中公佈支援平台的所有更新。我們也會在 AWS Elastic Beanstalk 平台指南中提供所有支援平台及其元件的清單，以及平台歷史記錄。如需更多資訊，請參閱支援平台。

您負責下列作業：

- 更新您控制的所有元件 (在「 AWS 共用職責模型」中識別為「客戶」)。這包括確保您應用程式、資料，以及您應用程式所需和您已下載之所有元件的安全性。
- 確保您的 Elastic Beanstalk 環境在支援的平台版本中執行，並將所有在淘汰平台版本中執行的環境遷移至受支援的版本。
- 解決所有因受管更新嘗試所引起的問題，重試更新。
- 如果選擇不使用 Elastic Beanstalk 受管更新，則要自行修補作業系統、執行時間、應用程式伺服器和 web 伺服器。您可手動套用平台更新執行此作業，或直接修補所有相關環境資源中的元件。
- 根據 AWS 共同責任模型，管理您在 Elastic Beanstalk 以外使用的任何 AWS 服務的安全性和合規性。

Elastic Beanstalk 平台支援政策

AWS Elastic Beanstalk 為執行應用程式提供多種平台 AWS。Elastic Beanstalk 支援的平台分支仍會繼續收到其供應商 (擁有者或社群) 提供的次要和修補程式更新。如需相關辭彙的完整定義，請參閱Elastic Beanstalk 平台詞彙表。

已淘汰的平台分支

當其供應商將支援平台分支的元件標示為生命週期結束 (EOL) 時，Elastic Beanstalk 會將平台分支標示為已停用。平台分支的元件包括下列項目：作業系統 (OS)、執行階段語言版本、應用程式伺服器或 Web 伺服器。

一旦平台分支標記為已淘汰，下列原則適用：

- Elastic Beanstalk 停止提供維護更新，包括安全性更新。
- Elastic Beanstalk 不再為已退休的平台分支提供技術支援。
- Elastic Beanstalk 不再使平台分支可供新的 Elastic Beanstalk 客戶用於部署到新環境。對於具有在已淘汰平台分支上執行的作用中環境的現有客戶，自發佈的淘汰日期起，有 90 天寬限期。

> **Note**
>
> 退休的平台分支將無法在 Elastic Beanstalk 控制台中使用。但是，對於擁有以淘汰平台分支為基礎的現有環境的客戶 AWS CLI，可透過 EB CLI 和 EB API 取得此功能。現有客戶也可以使用複製環境和重建環境主控台。

如需排定要淘汰的平台分支清單，請參閱隨後的 Elastic Beanstalk 平台排程主題正在淘汰的平台分支排程中的。

如需有關您環境的平台分支淘汰時的預期情況的詳細資訊，請參閱平台淘汰常見問答集。

超過 90 天寬限期

我們的淘汰平台分支原則不會移除對環境的存取，也不會刪除資源。但是，在已淘汰的平台分支上執行 Elastic Beanstalk 環境的現有客戶應該注意這樣做的風險。由於供應商標示其元件 EOL，Elastic Beanstalk 無法為已淘汰的平台分支提供安全性更新、技術支援或 Hotfix，因此這類環境最終可能會發生無法預測的情況。

例如，在淘汰的平台分支上執行的環境中可能會出現有害且嚴重的安全漏洞。或者，如果 EB API 動作隨著時間的推移變得與 Elastic Beanstalk 服務不相容，則可能會停止在此環境中工作。已淘汰平台分支上的環境保持作用中狀態的時間越長，這些類型的風險的機會就越大。我們強烈建議您將所有 Elastic Beanstalk 環境更新為受支援的平台版本，以便繼續享有元件供應商透過最新版本提供的重要安全性、效能和功能增強好處。

如果您的應用程式在淘汰的平台分支上執行時應該遇到問題，而您無法將其移轉到支援的平台，則需要考慮其他替代方案。解決方法包括將應用程式封裝到 Docker 映像檔以將其作為 Docker 容器執行。這將允許客戶使用我們的任何碼頭解決方案，例如我們的 Elastic Beanstalk 2023/AL2 碼頭平台或其他基於碼頭的服務，例如 Amazon ECS 或 Amazon EKS。非 Docker 替代方案包括我們的 AWS CodeDeploy 服務，它允許您完全自定義所需的運行時間。

Elastic Beanstalk 平台發布時間表

為了確保您的應用程式在受支援且安全的環境中執行，Elastic Beanstalk 會針對其受管理平台提供定期更新，如上一個主題所述。共同責任模式除了每月發行新平台分支版本之外，我們的發行維護還包括以下流程：

- 發行新的平台分支 — 這些分支通常會引入新的主要版本的執行階段語言、作業系統或應用程式伺服器。
- 平台分支的淘汰 — 當平台分支的其中一個元件達到生命週期結束 (EOL) 時，我們必須淘汰平台分支。有關退休分行政策的更多信息，請參閱 Elastic Beanstalk 平台支援政策

主題

規劃資源

除了接下來的排程之外，還有其他資源可協助您規劃在 Elastic Beanstalk 平台上執行的應用程式的維護和支援。如需有關我們平台元件、重要日期和發行公告的詳細資訊，請參閱下列資源：

- AWS Elastic Beanstalk 平台指南 — 本指南提供了我們每個平台分支的詳細組件列表。它還按發布日期提供了具有相同詳細信息的平台歷史記錄。本指南可以在您的平台分支的特定組件更改時通知您。如果您的應用程式開始行為不同，您也可以在平台指南中交互參照發生日期，以查看是否有任何平台變更可能會影響您的應用程式。
- AWS Elastic Beanstalk 版本說明 — 我們的版本說明會公佈我們所有的平台版本，包括次要版本和主要版本。這包括我們每月的平台更新、安全性版本、Hotfix 和淘汰公告。您可以從版本說明文件訂閱我們的 RSS 摘要。

即將發布的平台分支

下表列出了即將推出的 Elastic Beanstalk 平台分支及其目標發布日期。這些日期是暫定的，可能會有所變更。

運行時版本/平台分支	作業系統	目標發行日期
Corretto 21 with Tomcat 10 AL2023	Amazon Linux 2023	2024 年 9 月
PHP 8.3 AL2023	Amazon Linux 2023	2024 年 9 月
Python 3.12 AL2023	Amazon Linux 2023	2024 年 9 月
Ruby 3.3 AL2023	Amazon Linux 2023	2024 年 11 月

正在淘汰的平台分支排程

下表列出了排定退休的 Elastic Beanstalk 平台分支，因為它們的某些元件已達到其生命週期結束 (EOL)。

如需淘汰平台分支 (包含其特定元件) 的詳細清單，請參閱平台指南中的淘汰AWS Elastic Beanstalk 平台版本。

運行時版本/平台分支	作業系統	目標退休日期
Corretto 8 with Tomcat 8.5 AL2	Amazon Linux 2	2024 年 9 月 30 日

運行時版本/平台分支	作業系統	目標退休日期
Corretto 11 with Tomcat 8.5 AL2	Amazon Linux 2	2024 年 9 月 30 日
.NET 6 AL2023	Amazon Linux 2023	2025 年 1 月 31 日
Node.js 14 AL2	Amazon Linux 2	2024 年 9 月 30 日
Node.js 16 AL2	Amazon Linux 2	2024 年 9 月 30 日
Ruby 2.7 AL2	Amazon Linux 2	2024 年 9 月 30 日
Ruby 3.0 AL2	Amazon Linux 2	2024 年 9 月 30 日
PHP 8.0 AL2	Amazon Linux 2	2024 年 9 月 30 日
PHP 8.1 AL2	Amazon Linux 2	2025 年 1 月 31 日
PHP 8.1 AL2023	Amazon Linux 2023	2025 年 1 月 31 日
Python 3.7 AL2	Amazon Linux 2	2024 年 9 月 30 日
Python 3.8 AL2	Amazon Linux 2	2025 年 1 月 31 日

淘汰的平台分支歷史記錄

下表列出了已處於淘汰狀態的 Elastic Beanstalk 平台分支。您可以在平台指南的平台歷史記錄中查看這些平AWS Elastic Beanstalk 台分支及其組件的詳細歷史記錄。

Amazon Linux 2 (AL2)

運行時版本/平台分支	淘汰日期		
Corretto 11 with Tomcat 7 AL2	2022 年 6 月 29 日		
Corretto 8 with Tomcat 7 AL2	2022 年 6 月 29 日		

運行時版本/平台分支	淘汰日期		
Node.js 12 AL2	2022 年 12 月 23 日		
Node.js 10 AL2	2022 年 6 月 29 日		
PHP 7.4 AL2	2023 年 6 月 9 日		
PHP 7.3 AL2	2022 年 6 月 29 日		
PHP 7.2 AL2	2022 年 6 月 29 日		
Ruby 2.6 AL2	2022 年 12 月 23 日		
Ruby 2.5 AL2	2022 年 6 月 29 日		

Amazon Linux AMI (AL1)

運行時版本/平台分支	淘汰日期		
Single Container Docker	2022 年 7 月 18 日		
Multicontainer Docker	2022 年 7 月 18 日		
Preconfigured Docker - GlassFish 5.0 with Java 8	2022 年 7 月 18 日		
Go 1	2022 年 7 月 18 日		
Java 8	2022 年 7 月 18 日		
Java 7	2022 年 7 月 18 日		

運行時版本/平台分支	淘汰日期		
Java 8 with Tomcat 8.5	2022 年 7 月 18 日		
Java 7 with Tomcat 7	2022 年 7 月 18 日		
Node.js	2022 年 7 月 18 日		
PHP 7.2 - 7.3	2022 年 7 月 18 日		
Python 3.6	2022 年 7 月 18 日		
Ruby 2,4, 2.5, 2.6 with Passenger	2022 年 7 月 18 日		
Ruby 2.4, 2.5, 2.6 with Puma	2022 年 7 月 18 日		
Go 1.3–1.10	2020 年 10 月 31 日		
Java 6	2020 年 10 月 31 日		
Node.js 4.x–8.x	2020 年 10 月 31 日		
PHP 5.4–5.6	2020 年 10 月 31 日		
PHP 7.0–7.1	2020 年 10 月 31 日		
Python 2.6、2.7、3.4	2020 年 10 月 31 日		
Ruby 1.9.3	2020 年 10 月 31 日		
Ruby 2.0–2.3	2020 年 10 月 31 日		

> **Note**
>
> 2022 年 7 月 18 日，Elastic Beanstalk 已將所有以 Amazon Linux AMI (AL1) 為基礎的平台分支狀態設為已淘汰。如需詳細資訊，請參閱 平台淘汰常見問答集。

Windows Server

運行時版本/平台分支	淘汰日期		
在 64 位元 Windows Server (和 Core) 2012 R2 0.1.0 版上執行的 IIS 8.5	2022 年 6 月 29 日		
在 64 位元 Windows Server (和 Core) 2012 R2 1.2.0 版上執行的 IIS 8.5	2022 年 6 月 29 日		
在 64 位元 Windows Server 2016 (和 Core) 1.2.0 版上執行的 IIS 10.0	2022 年 6 月 29 日		
在 64 位元 Windows Server 2012 R1 平台分支上執行的 IIS 8	2022 年 6 月 22 日		
在 64 位元 Windows Server 2012 R1 0.1.0 版上執行的 IIS 8	2022 年 6 月 22 日		

運行時版本/平台分支	淘汰日期		
在 64 位元 Windows Server 2012 R1 1.2.0 版上執行的 IIS 8	2022 年 6 月 22 日		

> **Note**
>
> 如需有關 Windows 2012 R2 平台分支機構淘汰的詳細資訊，請參閱AWS Elastic Beanstalk 版本說明中退休的 Windows Server 2012 年 R2 平台分支。

淘汰的伺服器和作業系統歷史

下表提供 Elastic Beanstalk 平台不再支援的作業系統、應用程式伺服器和 Web 伺服器的歷史記錄。所有使用這些元件的平台分支現已淘汰。這些日期反映了最後一個包含元件的 Elastic Beanstalk 平台分支的退休日期。

作業系統

作業系統版本	平台淘汰日期		
Windows Server 2012 R2 running IIS 8.5	2023 年 12 月 4 日		
Windows Server Core 2012 R2 running IIS 8.5	2023 年 12 月 4 日		
Amazon Linux AMI (AL1)	2022 年 7 月 18 日		
Windows Server 2012 R1	2022 年 6 月 22 日		

作業系統版本	平台淘汰日期		
Windows Server 2008 R2	2019 年 10 月 28 日		

應用程式伺服器

應用程式伺服器版本	平台淘汰日期		
Tomcat 7	2022 年 6 月 29 日針對 Amazon Linux 2 (AL2) 平台 2022 年 7 月 18 日針對 Amazon Linux AMI (AL1) 平台		
Tomcat 8	2020 年 10 月 31 日		
Tomcat 6	2020 年 10 月 31 日		

Web 伺服器

Web 伺服器版本	平台淘汰日期		
在 64 位元 Windows Server 上執行的 IIS 8	2022 年 6 月 22 日		
Apache HTTP 伺服器 2.2	2020 年 10 月 31 日		
Nginx 1.12.2	2020 年 10 月 31 日		

支援 Elastic Beanstalk 的平台

AWS Elastic Beanstalk 提供各種平台，您可以在其上建立您的應用程式。您針對這些平台來設計 Web 應用程式，而 Elastic Beanstalk 會將您的程式碼部署到您選擇來建立作用中應用程式環境的平台版本。

Elastic Beanstalk 為程式設計語言 (Go、Java、Node.js、PHP、Python、Ruby)、、應用程式伺服器 (Tomcat、Passenger、Puma) 及 Docker 容器提供平台。部分平台有多個並行支援的版本。

Elastic Beanstalk 會佈建所需的資源 (包括一或多個 Amazon EC2 執行個體)，來執行您的應用程式。Amazon EC2 執行個體上執行的軟體堆疊取決於您為環境選取的特定平台版本。

您可以自訂和設定您應用程式在平台中依賴的軟體。在在 Linux 伺服器上自訂軟體和在 Windows Server 上自訂軟體中進一步了解。詳細的版本備註近期已發行，位於 AWS Elastic Beanstalk 版本備註。

支援平台

「AWS Elastic Beanstalk 平台」指南會在 Elastic Beanstalk 支援的平台區段中列出所有目前的平台分支版本。平台指南還列出了每個平台的平台歷史記錄，其中包括先前的分支平台版本的列表。若要檢視每個平台的平台歷史記錄，請選取下列其中一個連結。

- Docker
- Go
- Java SE
- Tomcat (執行 Java SE)
- Linux 上的 .NET Core
- Windows Server 上的 .NET
- Node.js
- PHP
- Python
- Ruby

平台分支的解決方案堆疊名稱

您可以使用指定平台分支版本的解決方案堆疊名稱，透過 EB CLI、Elastic Beanstalk API 或 CLI 啟動環境。AWS「AWS Elastic Beanstalk 平台」指南會在 Elastic Beanstalk 支援的平台和平台歷史記錄區段中，在平台分支版本下列出解決方案堆疊名稱。

若要擷取可用來建立環境的所有解決方案堆疊名稱，請使用 ListAvailableSolutionStacksAPI 或 AWS CLI `aws elasticbeanstalk list-available-solution-stacks`中的。

Elastic Beanstalk Linux 平台

Elastic Beanstalk 支援的大多數平台均以 Linux 作業系統為基礎。具體來說，這些平台是基於 Amazon Linux，由 AWS. Elastic Beanstalk Linux 平台使用 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體，而這些執行個體執行 Amazon Linux。

Elastic Beanstalk Linux 平台提供許多立即可用的功能。您可以透過多種方式擴充平台以支援您的應用程式。如需詳細資訊，請參閱 the section called “擴充 Linux 平台”。

主題

- 支援的 Amazon Linux 版本
- Elastic Beanstalk Linux 平台的清單
- 擴充 Elastic Beanstalk Linux 平台

支援的 Amazon Linux 版本

AWS Elastic Beanstalk 支持基於 Amazon Linux 2 和 Amazon Linux 2023 的平台。

從 2023 年 10 月 19 日開始，Elastic Beanstalk 支援在 AL2023 平台上使用 Amazon Linux 2 平台也支援的所有程式語言。Beanstalk 也支援 Amazon Linux 2 和 Amazon Linux 2023 上的 Docker 和以 ECS 為基礎的 Docker 平台。

如需 Amazon Linux 2 和 Amazon Linux 2023 的詳細資訊，請參閱下列內容：

- Amazon Linux 2 — Amazon EC2 用戶指南中的亞馬遜 Linux。
- Amazon Linux 2023 –《Amazon Linux 2023 使用者指南》中的什麼是 Amazon Linux 2023？

如需支援平台版本的詳細資訊，請參閱支援 Elastic Beanstalk 的平台。

> **Note**
>
> 您可以將應用程式從 Elastic Beanstalk AL1 或 AL2 平台分支遷移到同等的 AL2023 平台分支。如需詳細資訊，請參閱 將您的 Elastic Beanstalk Linux 應用程式遷移到 Amazon Linux 2023 或 Amazon Linux 2。

Amazon Linux 2023

AWS 宣布 Amazon Linux 2023 在 2023 年三月正式上市。《Amazon Linux 2023 使用者指南》總結說明了 Amazon Linux 2 與 Amazon Linux 2023 間的主要差異。如需詳細資訊，請參閱《使用者指南》中的比較 Amazon Linux 2 及 Amazon Linux 2023。

Elastic Beanstalk Amazon Linux 2 和 Amazon Linux 2023 平台之間具有高度相容性。雖然有部分差異需要注意：

- 執行個體中繼資料服務版本 1 (IMDSv1) – AL2023 平台上的 DisableIMDSv1 選項設定預設為 `true`。AL2 平台上的預設值為 `false`。
- pkg-repo 執行個體工具 — pkg-repo 工具不適用於在 AL2023 平台上執行的環境。但您可以手動將套件和作業系統更新套用至 AL2023 執行個體。如需詳細資訊，請參閱《Amazon Linux 2023 使用者指南》中的管理套件和作業系統更新。
- Apache HTTPd 組態 — AL2023 平台的 Apache `httpd.conf` 檔案具有部分與 AL2 不同的組態設定：
  - 在預設情況下，拒絕存取伺服器的整個檔案系統。Apache 網站安全性提示頁面上的依預設保護伺服器檔案，提供了對此類設定的說明。
  - 避免使用者覆寫您已設定的安全性功能。除特別啟用的目錄外，組態會拒絕存取所有目錄中的 `.htaccess` 設定。Apache 網站安全性提示頁面上的保護系統設定，提供了對此設定的說明。Apache HTTP 伺服器教學課程：.htaccess 檔案頁面指出，這項設定可能有助於改善效能。
  - 拒絕存取具有名稱模式 `.ht*` 的檔案。此設定可防止 Web 用戶端檢視 `.htaccess` 和 `.htpasswd` 檔案。

您可以針對您的環境變更上述任何組態設定。如需詳細資訊，請參閱擴充 Elastic Beanstalk Linux 平台。展開反向代理主題，以查看設定 Apache HTTPD 章節。

Elastic Beanstalk Linux 平台的清單

以下清單列出 Elastic Beanstalk 針對不同程式設計語言和 Docker 容器支援的 Linux 平台。Elastic Beanstalk 為以 Amazon Linux 2 和 Amazon Linux 2023 為基礎的平台提供所有以下支援。如需有關平台的詳細資訊，請選取相應的連結。

- Docker (和 ECS Docker)
- Go
- Tomcat (執行 Java SE)
- Java SE
- Linux 上的 .NET Core
- Node.js
- PHP
- Python
- Ruby

擴充 Elastic Beanstalk Linux 平台

AWS Elastic Beanstalk Linux 平台提供許多立即可用的功能，以支援開發和執行應用程式。您可以視需要透過多種方式擴充平台，以設定選項、安裝軟體、新增檔案和啟動命令、提供建置和執行時間說明，以及新增可在您環境的 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體之各種佈建階段執行的初始化指令碼。

Buildfile 和 Procfile

某些平台可讓您自訂建置或準備應用程式的方式，以及指定執行應用程式的程序。每個單獨的平台主題都特別提到 Buildfile 和/或 Procfile，如果平台支援它們。在 平台 下查詢您的特定平台。

對於所有支援的平台，語法和語意都是相同的，如本頁所述。個別的平台主題提到這些檔案的特定用法，以建置和執行各自語言的應用程式。

Buildfile

若要為應用程式指定自訂建置和組態命令，請將名為 `Buildfile` 的檔案放在應用程式原始碼的根目錄中。檔案名稱區分大小寫。請將以下語法用於 `Buildfile`。

```
<process_name>: <command>
```

Buildfile 中的命令必須符合下列規則表達式：^[A-Za-z0-9_-]+:\s*[^\s].*$。

Elastic Beanstalk 不會監控透過 Buildfile 執行的應用程式。針對短期執行且須在任務完成後終止的命令，請使用 Buildfile。針對長期執行且不應退出的應用程式程序，請使用 Procfile。

Buildfile 內的所有路徑均相對於原始碼套件的根目錄。在下列 Buildfile 的範例中，build.sh 為位於原始碼套件根目錄的 Shell 指令碼。

Example Buildfile

```
make: ./build.sh
```

若要提供自訂建置步驟，建議您將 predeploy 平台勾點用於除了最簡單的命令以外的任何項目，而不是使用 Buildfile。平台勾點允許使用更豐富的指令碼和更完善的錯誤處理方式。下節將說明平台勾點。

Procfile

若要指定可啟動和執行應用程式的自訂命令，請將名為 Procfile 的檔案放在應用程式原始碼的根目錄中。檔案名稱區分大小寫。請將以下語法用於 Procfile。您可以指定一或多個命令。

```
<process_name1>: <command1>
<process_name2>: <command2>
...
```

Procfile 中的每一行必須符合下列規則表達式：^[A-Za-z0-9_-]+:\s*[^\s].*$

針對長期執行且不應退出的應用程式程序，使用 Procfile。Elastic Beanstalk 預期 Procfile 執行的程序會持續執行。Elastic Beanstalk 會監控這些程序，並重新啟動終止的任何程序。若是短期執行的程序，請使用 Buildfile。

Procfile 內的所有路徑均相對於原始碼套件的根目錄。以下範例 Procfile 定義了三個程序。第一個程序 (在此範例中稱為 web) 是主要 Web 應用程式。

Example Procfile

```
web: bin/myserver
cache: bin/mycache
foo: bin/fooapp
```

Elastic Beanstalk 會將代理伺服器設定為將要求轉送至連接埠 5000 上的主要 Web 應用程式，而且您可以設定此連接埠號碼。`Procfile` 的常見用途是將此連接埠號碼作為命令引數傳遞給您的應用程式。如需代理組態的相關資料，請展開本頁面的反向代理組態一節。

Elastic Beanstalk 會從日誌檔案中的 `Procfile` 程序擷取標準輸出和錯誤串流。Elastic Beanstalk 會在執行程序後，命名日誌檔案，並將其存放至 `/var/log`。例如，前述範例的 `web` 程序，會分別為 `web-1.log` 和 `web-1.error.log` 產生名為 `stdout` 和 `stderr` 的日誌。

平台勾點

平台勾點經過專門設計，可擴充環境的平台。這些是您部署為應用程式原始碼一部分的自訂指令碼與其他可執行檔，而 Elastic Beanstalk 會在各個執行個體佈建階段期間執行。

> **Note**
>
> Amazon Linux AMI 平台版本 (前述的 Amazon Linux 2) 不支援平台勾點。

應用程式部署平台勾點

當您提供用於應用程式部署的新原始碼套件，或是當您進行需要終止和重新建立所有環境執行個體的組態變更時，就會發生應用程式部署。

若要提供在應用程式部署期間執行的平台勾點，請將檔案放在原始碼套件中的 `.platform/hooks` 目錄下，位於下列其中一個子目錄中。

- `prebuild` – 此處的檔案會在 Elastic Beanstalk 平台引擎下載並擷取應用程式原始碼套件後執行，以及在完成本身設定並設定應用程式和 Web 伺服器前執行。

  `prebuild` 檔案會在執行任何組態檔案的 commands 區段中所發現的命令後執行，以及在執行 `Buildfile` 命令前執行。
- `predeploy` – 此處的檔案會在 Elastic Beanstalk 平台引擎設定並設定應用程式和 Web 伺服器後執行，以及在將其部署至其最終執行時間位置前執行。

  `predeploy` 檔案會在執行任何組態檔案的 container_commands 區段中所發現的命令後執行，以及在執行 `Procfile` 命令前執行。
- `postdeploy` – 此處的檔案會在 Elastic Beanstalk 平台引擎部署應用程式和代理伺服器後執行。

  這是最後一個部署工作流程步驟。

組態部署平台勾點

當您進行只更新環境執行個體而不重新建立組態變更時，就會發生組態部署。下列選項更新會造成組態更新。

- 環境屬性和平台特定設定
- 靜態檔案
- AWS X-Ray 常駐程式
- 日誌儲存與串流
- 應用程式連接埠 (如需詳細資訊，請展開此頁面的 反向代理主機組態一節)

要提供在組態部署期間執行的勾點，請將它們放置在原始碼套件中的 `.platform/confighooks` 目錄下。與應用程式部署勾點相同的三個子目錄也適用。

更多關於平台勾點

可執行檔可以是二進位檔案，或開頭為 `#!` 行且包含其解譯器路徑 (例如 `#!/bin/bash`) 的指令碼檔案。所有檔案都必須有執行許可。使用 `chmod +x` 在勾點檔案上設定執行權限。對於在 2022 年 4 月 29 日或之後發佈的所有以 Amazon Linux 2023 和 Amazon Linux 2 為基礎的平台版本，Elastic Beanstalk 會自動授予執行許可給所有平台勾點指令碼。在此情況下，您不需要手動授予執行許可。如需這些平台版本的清單，請參閱 AWS Elastic Beanstalk 版本備註指南中的 2022 年 4 月 29 日 Linux 平台版本備註。

Elastic Beanstalk 會依照檔案名稱的辭典編纂方式排序，在這些目錄中逐一執行檔案。所有檔案都會以 `root` 使用者身分執行。平台勾點的目前工作目錄 (cwd) 為應用程式的根目錄。若是 `prebuild` 和 `predeploy` 檔案，這是應用程式暫存目錄，而若是 `postdeploy` 檔案，則是目前的應用程式目錄。如果其中一個檔案失敗 (以非零結束代碼結束)，部署則會中止及失敗。

如果平台勾點文字指令碼含有 Windows 回車/換行 (CRLF) 換行符字元，則可能會失敗。如果檔案儲存在 Windows 主機中，然後傳輸到 Linux 伺服器，就可能包含 Windows CRLF 換行符。對於在 2022 年 12 月 29 日或之後發佈的平台，Elastic Beanstalk 會自動將平台勾點文字檔案中的 Windows CRLF 字元轉換為 Linux 換行 (LF) 換行符字元。如果您的應用程式在此日期之前發佈的任何 Amazon Linux 2 平台上執行，則您需要將 Windows CRLF 字元轉換為 Linux LF 字元。完成此操作的一種方法是在 Linux 主機上建立並保存程式碼檔案。也可以在網際網路上找到轉換這些字元的工具。

執行的檔案可存取您已在應用程式選項中定義的所有環境屬性，並可存取 `HOME`、`PATH` 和 `PORT` 這些系統環境變數。

若要將環境變數和其他組態選項的值擷取至您的平台勾點指令碼中，您可以使用在環境執行個體上 Elastic Beanstalk 提供的 `get-config` 公用程式。如需詳細資訊，請參閱the section called “平台指令碼工具”。

組態檔案

您可以將組態檔案新增至應用程式原始碼的 `.ebextensions` 目錄，以設定您 Elastic Beanstalk 環境的各個層面。此外，組態檔案可讓您自訂軟體和您環境的執行個體上的其他檔案，並在執行個體上執行初始化命令。如需詳細資訊，請參閱the section called “Linux 伺服器”。

您也可以使用組態檔案來設定組態選項。許多選項可控制平台行為，而其中某些選項為平台專用。

對於以 Amazon Linux 2 和 Amazon Linux 2023 為基礎的平台，我們建議您在執行個體佈建期間使用 Buildfile、Procfile 和 platform hooks (平台勾點)，在環境執行個體上設定和執行自訂程式碼。這些機制會在此頁面的先前區段中說明。您仍可在 `.ebextensions` 組態檔案中使用命令和容器命令，但它們並不容易使用。例如，從語法的角度來看，在 YAML 檔案內寫入命令指令碼可能相當困難。您仍須將 `.ebextensions` 組態檔案用於需要參照 AWS CloudFormation 資源的任何指令碼。

反向代理組態

所有 Amazon Linux 2 和 Amazon Linux 2023 平台版本都使用 nginx 作為預設反向代理伺服器。Tomcat、Node.js、PHP 和 Python 平台也支援 Apache HTTPD 作為替代方案。若要在這些平台上選取 Apache，請將 `aws:elasticbeanstalk:environment:proxy` 命名空間中的 `ProxyServer` 選項設定為 `apache`。所有平台都會以統一的方式啟用代理伺服器設定，如本節所述。

> **Note**
>
> 在 Amazon Linux AMI 平台版本 (前述的 Amazon Linux 2) 上，您可能需要以不同的方式設定代理伺服器。您可以在本指南中各自的平台主題下找到這些舊式詳細資訊。

Elastic Beanstalk 會設定您環境執行個體上的代理伺服器，以將環境的根 URL 上的 Web 流量轉送至主要 Web 應用程式；例如 `http://my-env.elasticbeanstalk.com`。

根據預設，Elastic Beanstalk 會設定代理，以將 80 連接埠的傳入請求轉送至連接埠 5000 上的主要 Web 應用程式。您可以在組態檔案中使用 aws:elasticbeanstalk:application:environment 命名空間來設定 `PORT` 環境屬性 (如以下範例所示)，藉此設定此連接埠號碼。

```
option_settings:
  - namespace:  aws:elasticbeanstalk:application:environment
```

```
    option_name:  PORT
    value:  <main_port_number>
```

如需有關應用程式設定環境變數的詳細資訊，請參閱 the section called “選項設定”。

您的應用程式應該會在已於代理中設定的連接埠上進行接聽。如果您使用 PORT 環境屬性變更預設連接埠，程式碼可讀取 PORT 環境變數的值，藉此存取該連接埠。例如，在 Go 中呼叫 `os.Getenv("PORT")`，或在 Java 中呼叫 `System.getenv("PORT")`。如果您設定代理以將流量傳送至多個應用程式程序，您可以設定多個環境屬性，然後同時在代理組態和應用程式程式碼中使用其值。另一個選項是在 `Procfile` 中以命令引數的形式，將連接埠值傳送至程序。如需詳細資料，請展開本頁面的 Buildfile 和 Procfile 區段。

設定 nginx

Elastic Beanstalk 使用 nginx 做為預設的反向代理伺服器，將您的應用程式映射到 Elastic Load Balancing 負載平衡器。Elastic Beanstalk 提供了預設的 nginx 組態，您可以加以擴展，或使用自己的組態將其完全覆寫。

> **Note**
>
> 當您新增或編輯 nginx `.conf` 組態檔案時，請務必將其編碼為 UTF-8 格式。

若要擴充 Elastic Beanstalk 預設 nginx 組態，請將 `.conf` 組態檔案加入您應用程式原始碼套件中名為 `.platform/nginx/conf.d/` 的資料夾。Elastic Beanstalk nginx 組態會自動在此資料夾中加入 `.conf` 檔案。

```
~/workspace/my-app/
|-- .platform
|   `-- nginx
|       `-- conf.d
|           `-- myconf.conf
`-- other source files
```

若要完全覆寫 Elastic Beanstalk 預設 nginx 組態，請在 `.platform/nginx/nginx.conf` 的原始碼套件中加入組態：

```
~/workspace/my-app/
|-- .platform
|   `-- nginx
```

```
|       `-- nginx.conf
`-- other source files
```

如果您覆寫了 Elastic Beanstalk nginx 組態，請在 `nginx.conf` 中新增下列的行，以納入適用於 增強型運作狀態報告與監控、自動應用程式映射和靜態檔案的 Elastic Beanstalk 組態。

```
 include conf.d/elasticbeanstalk/*.conf;
```

設定 Apache HTTPD

Tomcat，Node.js，PHP 和 Python 平台允許您選擇 Apache HTTPD 代理服務器作為連至 nginx 的替代方案。這不是預設值。下列範例設定 Elastic Beanstalk 使用 Apache HTTPD。

Example .ebextensions/httpd-proxy.config

```
option_settings:
  aws:elasticbeanstalk:environment:proxy:
    ProxyServer: apache
```

您可以擴展 Elastic Beanstalk 預設 Apache 組態與您的其他組態檔案。或者，您可以完全覆寫 Elastic Beanstalk 預設 Apache 組態。

若要擴展 Elastic Beanstalk 預設 Apache 組態，請將 `.conf` 組態檔案加入您應用程式原始碼套件中名為 `.platform/httpd/conf.d` 的資料夾。Elastic Beanstalk Apache 組態會自動在此資料夾中加入 `.conf` 檔案。

```
~/workspace/my-app/
|-- .ebextensions
|   -- httpd-proxy.config
|-- .platform
|   -- httpd
|      -- conf.d
|         -- port5000.conf
|         -- ssl.conf
-- index.jsp
```

例如，下列 Apache 2.4 組態會在 5000 埠上新增接聽程式。

Example .platform/httpd/conf.d/port5000.conf

```
listen 5000
```

```
<VirtualHost *:5000>
  <Proxy *>
    Require all granted
  </Proxy>
  ProxyPass / http://localhost:8080/ retry=0
  ProxyPassReverse / http://localhost:8080/
  ProxyPreserveHost on

  ErrorLog /var/log/httpd/elasticbeanstalk-error_log
</VirtualHost>
```

若要完全覆寫 Elastic Beanstalk 預設 Apache 組態，請在 `.platform/httpd/conf/httpd.conf` 的原始碼套件中加入組態。

```
~/workspace/my-app/
|-- .ebextensions
|   -- httpd-proxy.config
|-- .platform
|   `-- httpd
|       `-- conf
|           `-- httpd.conf
`-- index.jsp
```

> **Note**
>
> 如果您覆寫了 Elastic Beanstalk Apache 組態，請在 `httpd.conf` 中新增下列的行，以納入適用於 增強型運作狀態報告與監控、自動應用程式映射和靜態檔案的 Elastic Beanstalk 組態。
>
> ```
> IncludeOptional conf.d/elasticbeanstalk/*.conf
> ```

> **Note**
>
> 如果您要將 Elastic Beanstalk 應用程式遷移至 Amazon Linux 2 或 Amazon Linux 2023 平台，請務必同時閱讀 the section called “遷移至 AL2023/AL2” 中的資訊。

主題

- 具有擴充功能的應用程式範例

- 執行個體部署工作流程
- 執行於 Amazon Linux 2 和更新版本的 ECS 的執行個體部署工作流程
- 平台指令碼工具

具有擴充功能的應用程式範例

以下範例示範具有 Elastic Beanstalk Amazon Linux 2 和 Amazon Linux 2023 平台支援之多項擴充功能的應用程式原始碼套件：`Procfile`、`.ebextensions` 組態檔案、自訂勾點和代理組態檔案。

```
~/my-app/
|-- web.jar
|-- Procfile
|-- readme.md
|-- .ebextensions/
|   |-- options.config        # Option settings
|   `-- cloudwatch.config     # Other .ebextensions sections, for example files and
 container commands
`-- .platform/
    |-- nginx/                # Proxy configuration
    |   |-- nginx.conf
    |   `-- conf.d/
    |       `-- custom.conf
    |-- hooks/                # Application deployment hooks
    |   |-- prebuild/
    |   |   |-- 01_set_secrets.sh
    |   |   `-- 12_update_permissions.sh
    |   |-- predeploy/
    |   |   `-- 01_some_service_stop.sh
    |   `-- postdeploy/
    |       |-- 01_set_tmp_file_permissions.sh
    |       |-- 50_run_something_after_app_deployment.sh
    |       `-- 99_some_service_start.sh
    `-- confighooks/          # Configuration deployment hooks
        |-- prebuild/
        |   `-- 01_set_secrets.sh
        |-- predeploy/
        |   `-- 01_some_service_stop.sh
        `-- postdeploy/
            |-- 01_run_something_after_config_deployment.sh
            `-- 99_some_service_start.sh
```

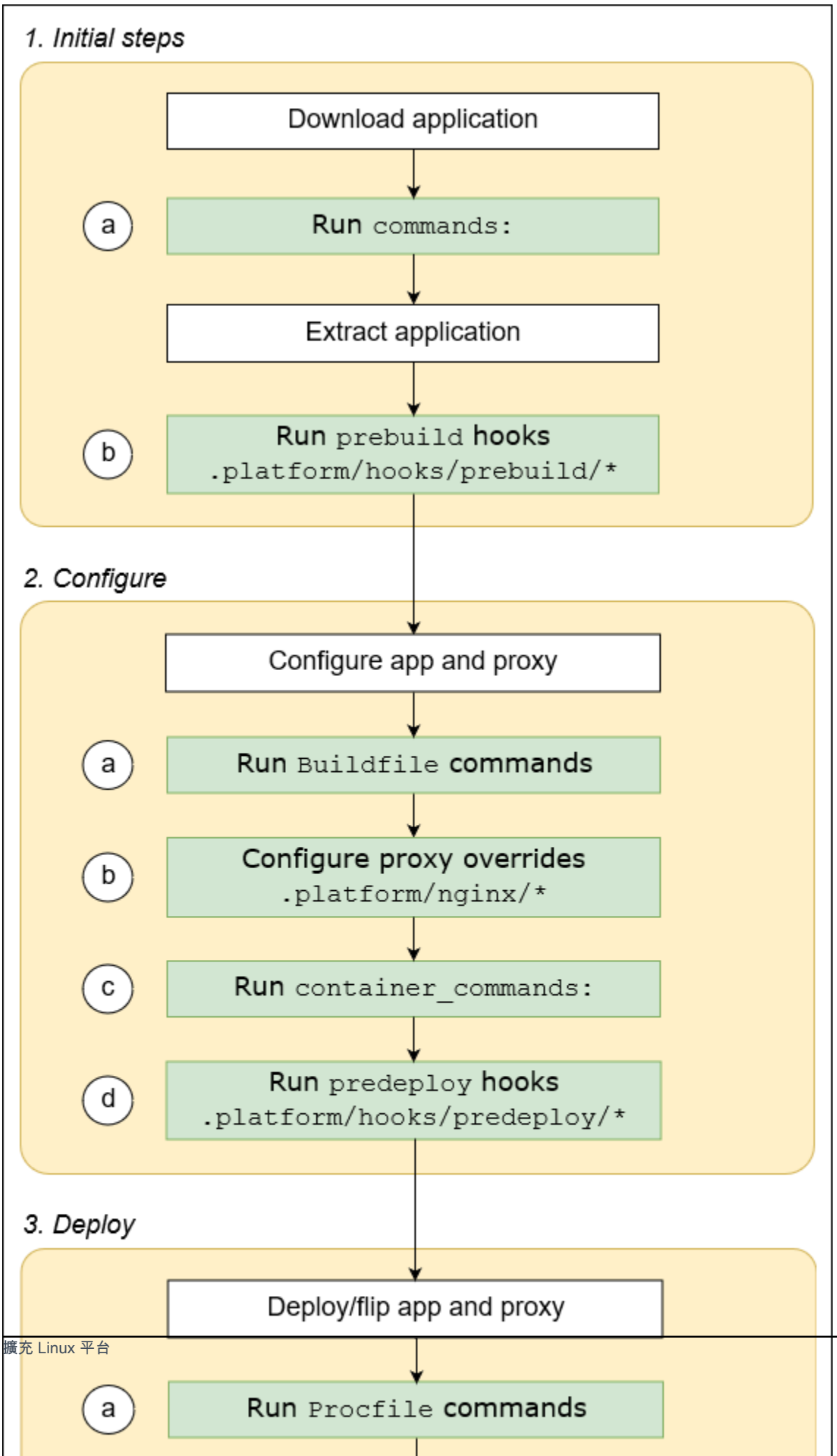
> **Note**
>
> Amazon Linux AMI 平台版本 (前述的 Amazon Linux 2) 不支援某些擴充功能。

執行個體部署工作流程

> **Note**
>
> 本節中的資訊不適用於執行於 Amazon Linux 2 和 Amazon Linux 2023 的 ECS 平台分支。如需詳細資訊，請參閱下一節 執行於 Amazon Linux 2 和更新版本的 ECS 的執行個體部署工作流程。

由於可透過多種方式擴充環境的平台，因此隨時了解 Elastic Beanstalk 佈建執行個體或對執行個體執行部署時所發生的情況會很有幫助。下圖顯示這整個部署流程。它描述了部署作業中的各個不同階段，以及 Elastic Beanstalk 在各個階段中採取的步驟。

> **備註**
>
> - 此圖不代表部署期間 Elastic Beanstalk 在環境執行個體上進行的完整步驟集。此圖表僅供示意，為您提供執行自訂的順序和內容。
> - 為了簡單起見，圖表僅提及 `.platform/hooks/*` 勾點子目錄 (適用於應用程式部署)，而不提及 `.platform/confighooks/*` 勾點子目錄 (適用於組態部署)。後面子目錄中的勾點與圖中對應子目錄中的勾點執行步驟完全相同。


1. Initial steps
Download application
a
Run commands:
Extract application
b
Run prebuild hooks
.platform/hooks/prebuild/*
2. Configure
Configure app and proxy
a
Run Buildfile commands
b
Configure proxy overrides
.platform/nginx/*
c
Run container_commands:
d
Run predeploy hooks
.platform/hooks/predeploy/*
3. Deploy
Deploy/flip app and proxy
a
Run Procfile commands

下列清單詳細說明部署階段和步驟。

1. 初始步驟

   Elastic Beanstalk 會下載並擷取您的應用程式。在完成這些步驟中的每個步驟後，Elastic Beanstalk 會執行其中一個擴充步驟。

   a. 執行任何組態檔案的 commands: 區段中所發現的命令。

   b. 執行原始碼套件 `.platform/hooks/prebuild` 目錄中找到的任何可執行檔案 (`.platform/confighooks/prebuild` 適用於組態部署)。

2. 設定

   Elastic Beanstalk 會設定應用程式和代理伺服器。

   a. 執行原始碼套件的 `Buildfile` 中所發現的命令。

   b. 如果原始碼套件的 `.platform/nginx` 目錄中有任何自訂代理組態檔案，則會將其複製到其執行時間位置。

   c. 執行任何組態檔案的 container_commands: 區段中所發現的命令。

   d. 執行原始碼套件 `.platform/hooks/predeploy` 目錄中找到的任何可執行檔案 (`.platform/confighooks/predeploy` 適用於組態部署)。

3. 部署

   Elastic Beanstalk 會部署並執行應用程式和代理伺服器。

   a. 執行原始碼套件的 `Procfile` 檔案中所發現的命令。

   b. 使用自訂代理組態檔案 (如果有的話)，執行或重新執行代理伺服器。

   c. 執行原始碼套件 `.platform/hooks/postdeploy` 目錄中找到的任何可執行檔案 (`.platform/confighooks/postdeploy` 適用於組態部署)。

執行於 Amazon Linux 2 和更新版本的 ECS 的執行個體部署工作流程

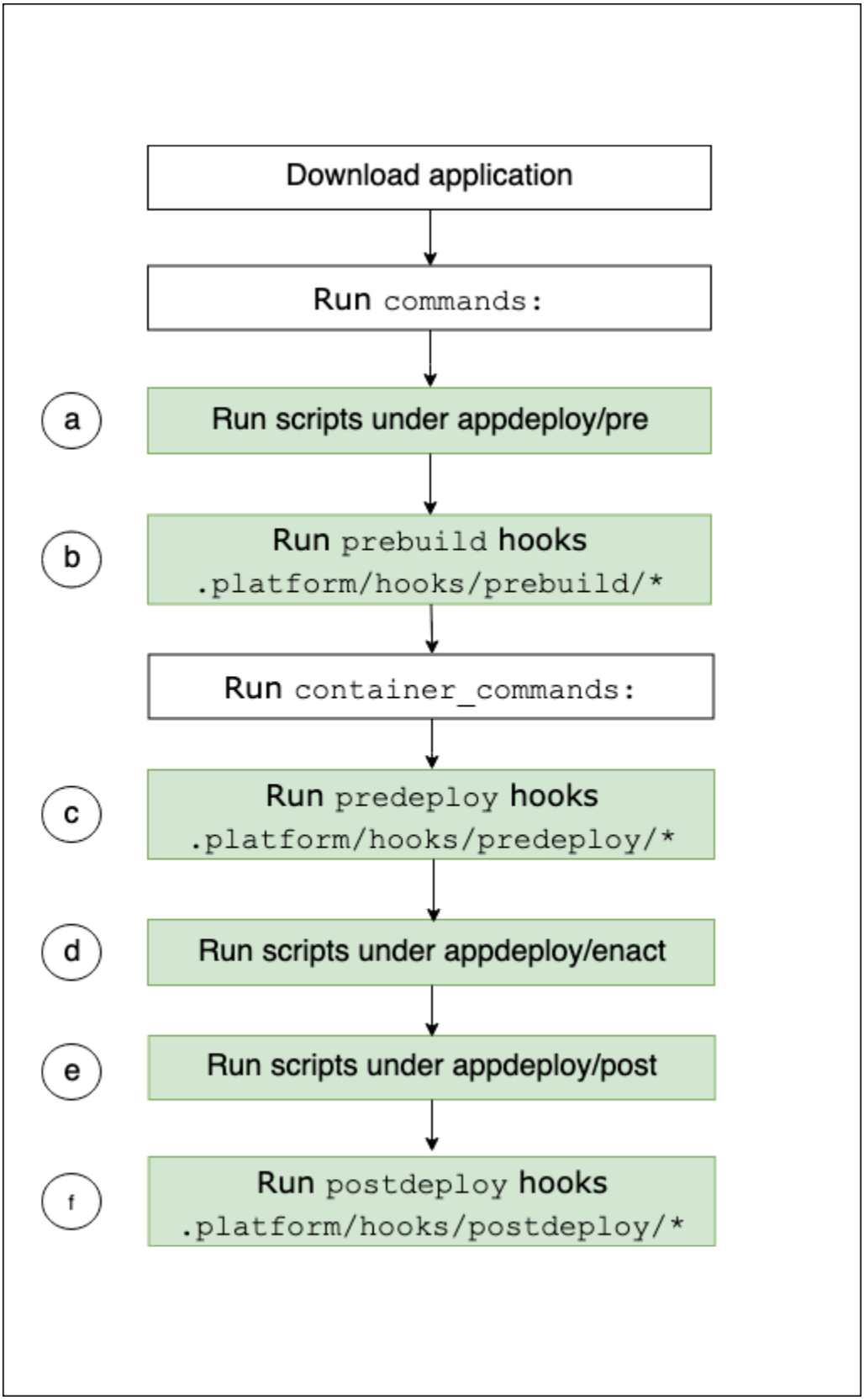
上一節介紹了在應用程式部署工作流程整個階段支援的可擴展性功能。執行於 Amazon Linux 2 和更新版本的 ECS 平台分支的 Docker 具有某些差異。本節介紹這些概念如何運用於此特定平台分支。

由於可透過多種方式擴充環境的平台，因此隨時了解 Elastic Beanstalk 佈建執行個體或對執行個體執行部署時所發生的情況會很有幫助。下圖顯示了以執行於 Amazon Linux 2 的 ECS 和執行於 Amazon Linux 2023 的 ECS 平台分支為基礎的環境的整個部署工作流程。它描述了部署作業中的各個不同階段，以及 Elastic Beanstalk 在各個階段中採取的步驟。

與上一節中描述的工作流程不同，部署組態設定階段不支援以下可擴展性功能：`Buildfile` 命令、`Procfile` 命令、反向代理組態。

備註

- 此圖不代表部署期間 Elastic Beanstalk 在環境執行個體上進行的完整步驟集。此圖表僅供示意，為您提供執行自訂的順序和內容。
- 為了簡單起見，圖表僅提及 `.platform/hooks/*` 勾點子目錄 (適用於應用程式部署)，而不提及 `.platform/confighooks/*` 勾點子目錄 (適用於組態部署)。後面子目錄中的勾點與圖中對應子目錄中的勾點執行步驟完全相同。



下列清單詳細說明了部署工作流程步驟。

a. EBhooksDir 下 appdeploy/pre 目錄中所發現的任何可執行檔。
b. 執行原始碼套件 .platform/hooks/prebuild 目錄中找到的任何可執行檔案 (.platform/confighooks/prebuild 適用於組態部署)。
c. 執行原始碼套件 .platform/hooks/predeploy 目錄中找到的任何可執行檔案 (.platform/confighooks/predeploy 適用於組態部署)。
d. EBhooksDir 下 appdeploy/enact 目錄中所發現的任何可執行檔。
e. EBhooksDir 下 appdeploy/post 目錄中所發現的任何可執行檔。
f. 執行原始碼套件 .platform/hooks/postdeploy 目錄中找到的任何可執行檔案 (.platform/confighooks/postdeploy 適用於組態部署)。

參考 EBhooksDir 表示平台勾點目錄的路徑。若要擷取目錄路徑名稱，請使用環境執行個體命令列上的 get-config 指令碼工具，如下所示：

```
$ /opt/elasticbeanstalk/bin/get-config platformconfig -k EBhooksDir
```

平台指令碼工具

本主題說明為使用 Amazon Linux 平台的環境所 AWS Elastic Beanstalk 提供的工具。這些工具位於 Elastic Beanstalk 環境的 Amazon EC2 執行個體上。

get-config

使用 get-config 工具擷取環境變數值以及其他平台和執行個體資訊。此工具可於 /opt/elasticbeanstalk/bin/get-config 取得。

get-config 命令

每個 get-config 工具命令都會傳回特定類型的資訊。使用下列語法可執行任一工具的命令。

```
$ /opt/elasticbeanstalk/bin/get-config command [ options ]
```

以下範例執行 environment 命令。

```
$ /opt/elasticbeanstalk/bin/get-config environment -k PORT
```

視您選擇的命令和選項而定，工具會傳回含鍵值對或單一值的物件 (JSON 或 YAML)。

您可以使用 SSH 連線至 Elastic Beanstalk 環境中的 EC2 執行個體，測試 get-config。

> **Note**
>
> 當您執行 get-config 測試時，某些命令可能需要根使用者權限才能存取基礎資訊。如果您收到存取許可錯誤，請在 sudo 下重新執行命令。
> 在部署到環境的指令碼中使用工具時，不需要新增 sudo。Elastic Beanstalk 會以根使用者的身分執行所有指令碼。

以下幾節會說明工具的命令。

optionsettings – 組態選項

get-config optionsettings 命令會傳回一個物件，其中列出針對環境所設定並由環境執行個體平台使用的組態選項。這些選項會依命名空間加以組織。

```
$ /opt/elasticbeanstalk/bin/get-config optionsettings
{"aws:elasticbeanstalk:application:environment":
{"JDBC_CONNECTION_STRING":""},"aws:elasticbeanstalk:container:tomcat:jvmoptions":{"JVM
 Options":"","Xms":"256m","Xmx":"256m"},"aws:elasticbeanstalk:environment:proxy":
{"ProxyServer":"nginx","StaticFiles":
[""]},"aws:elasticbeanstalk:healthreporting:system":
{"SystemType":"enhanced"},"aws:elasticbeanstalk:hostmanager":
{"LogPublicationControl":"false"}}
```

若要傳回特定的組態選項，請使用 --namespace (-n) 選項來指定命名空間，並使用 --option-name (-o) 選項指定選項名稱。

```
$ /opt/elasticbeanstalk/bin/get-config optionsettings -
n aws:elasticbeanstalk:container:php:phpini -o memory_limit
256M
```

environment – 環境屬性

get-config environment 命令會傳回包含環境屬性清單的物件，其中包括使用者設定以及由 Elastic Beanstalk 提供的屬性。

```
$ /opt/elasticbeanstalk/bin/get-config environment
{"JDBC_CONNECTION_STRING":"","RDS_PORT":"3306","RDS_HOSTNAME":"anj9aw1b0tbj6b.cijbpanmxz5u.us-
west-2.rds.amazonaws.com","RDS_USERNAME":"testusername","RDS_DB_NAME":"ebdb","RDS_PASSWORD":"te
```

例如，Elastic Beanstalk 會提供環境屬性，以連線至整合式 Amazon RDS 資料庫執行個體 (例如 RDS_HOSTNAME)。這些 RDS 連線屬性會顯示於 `get-config environment` 的輸出內容，但不會出現在 `get-config optionsettings` 的輸出內容。這是因為您並未在組態選項中加以設定。

若要傳回特定環境屬性，請使用 `--key` (`-k`) 選項來指定屬性的鍵。

```
$ /opt/elasticbeanstalk/bin/get-config environment -k TESTPROPERTY
testvalue
```

container – 執行個體組態值

`get-config container` 命令會傳回一個物件，列出環境執行個體的平台和環境組態值。

下列範例會在 Amazon Linux 2 Tomcat 環境中顯示命令的輸出內容。

```
$ /opt/elasticbeanstalk/bin/get-config container
{"common_log_list":["/var/log/eb-engine.log","/var/log/eb-
hooks.log"],"default_log_list":["/var/log/nginx/access.log","/var/log/nginx/
error.log"],"environment_name":"myenv-1da84946","instance_port":"80","log_group_name_prefix":"/
aws/elasticbeanstalk","proxy_server":"nginx","static_files":
[""],"xray_enabled":"false"}
```

要返回特定鍵的值，請使用 `--key` (`-k`) 選項來指定鍵。

```
$ /opt/elasticbeanstalk/bin/get-config container -k environment_name
myenv-1da84946
```

addons – 附加元件組態值

`get-config addons` 命令會傳回內含環境附加元件組態資訊的物件。您可使用此物件來擷取與環境相關聯的 Amazon RDS 資料庫組態。

```
$ /opt/elasticbeanstalk/bin/get-config addons
{"rds":{"Description":"RDS Environment variables","env":
{"RDS_DB_NAME":"ebdb","RDS_HOSTNAME":"ea13k2wimu1dh8i.c18mnpu5rwvg.us-
east-2.rds.amazonaws.com","RDS_PASSWORD":"password","RDS_PORT":"3306","RDS_USERNAME":"user"}}}
```

您可以使用兩種方式限制結果。若要擷取特定附加元件的值，請使用 `--add-on` (`-a`) 選項來指定附加元件名稱。

```
$ /opt/elasticbeanstalk/bin/get-config addons -a rds
```

```
{"Description":"RDS Environment variables","env":
{"RDS_DB_NAME":"ebdb","RDS_HOSTNAME":"ea13k2wimu1dh8i.c18mnpu5rwvg.us-
east-2.rds.amazonaws.com","RDS_PASSWORD":"password","RDS_PORT":"3306","RDS_USERNAME":"user"}}
```

若要傳回附加元件內特定鍵的值，請新增 `--key (-k)` 選項以指定鍵。

```
$ /opt/elasticbeanstalk/bin/get-config addons -a rds -k RDS_DB_NAME
ebdb
```

platformconfig – 常數組態值

`get-config platformconfig` 命令會傳回內含平台組態資訊的物件，該資訊不會因為平台版本而有所改變。在執行相同平台版本的所有環境中，輸出內容相同。命令的輸出物件具有兩個內嵌物件：

- `GeneralConfig` – 包含所有 Amazon Linux 2 和 Amazon Linux 2023 平台分支最新版本中維持不變的資訊。
- `PlatformSpecificConfig` – 包含平台版本中維持不變且專屬於該平台版本的資訊。

下列範例顯示針對使用 Tomcat 8.5 執行 Corretto 11 平台分支的環境，執行命令會產生哪些輸出內容。

```
$ /opt/elasticbeanstalk/bin/get-config platformconfig
{"GeneralConfig":{"AppUser":"webapp","AppDeployDir":"/var/app/
current/","AppStagingDir":"/var/app/
staging/","ProxyServer":"nginx","DefaultInstancePort":"80"},"PlatformSpecificConfig":
{"ApplicationPort":"8080","JavaVersion":"11","TomcatVersion":"8.5"}}
```

要返回特定鍵的值，請使用 `--key (-k)` 選項來指定鍵。這些鍵在兩個嵌入物件中是唯一的鍵。您不需要指定包含鍵的對象。

```
$ /opt/elasticbeanstalk/bin/get-config platformconfig -k AppStagingDir
/var/app/staging/
```

get-config 輸出選項

使用 `--output` 選項來指定輸出物件格式。有效值為 `JSON` (預設) 和 `YAML`。這是全域選項。您必須在命令名稱之前加以指定。

以下範例會傳回 YAML 格式的組態選項值。

```
$ /opt/elasticbeanstalk/bin/get-config --output YAML optionsettings
aws:elasticbeanstalk:application:environment:
  JDBC_CONNECTION_STRING: ""
aws:elasticbeanstalk:container:tomcat:jvmoptions:
  JVM Options: ""
  Xms: 256m
  Xmx: 256m
aws:elasticbeanstalk:environment:proxy:
  ProxyServer: nginx
  StaticFiles:
        - ""
aws:elasticbeanstalk:healthreporting:system:
  SystemType: enhanced
aws:elasticbeanstalk:hostmanager:
  LogPublicationControl: "false"
```

pkg-repo

> **Note**
>
> pkg-repo 工具不適用於以 Amazon Linux 2023 平台為基礎的環境。但您可以手動將套件和作業系統更新套用至 AL2023 執行個體。如需詳細資訊，請參閱《Amazon Linux 2023 使用者指南》中的管理套件和作業系統更新

在某些緊急情況下，您可能需要使用必要 Elastic Beanstalk 平台版本尚未推出的 Amazon Linux 2 安全修補程式來更新 Amazon EC2 執行個體。預設情形下，您無法在 Elastic Beanstalk 環境中執行手動更新。原因在於，平台版本僅能搭配特定版本的 Amazon Linux 2 儲存庫一起使用。藉由這樣的版本鎖定設計，系統能確保執行個體執行的是受支援且一致的軟體版本。發生緊急情況時，如果您需要在環境中安裝 yum 套件，但新版 Elastic Beanstalk 平台尚未發佈該套件時，pkg-repo 工具可允許您在 Amazon Linux 2 上手動更新 yum 套件。

Amazon Linux 2 平台上的 pkg-repo 工具可提供解鎖 yum 套件儲存庫的功能。接著，您就可以手動執行 yum update，取得安全性修補程式。相反地，您也可以使用此工具鎖定 yum 套件儲存庫，以防止系統執行其他更新作業，隨著既定的更新時程更新。透過 Elastic Beanstalk 環境中所有 EC2 執行個體的 /opt/elasticbeanstalk/bin/pkg-repo 目錄，即可取用 pkg-repo 工具。

使用 pkg-repo 工具所做的變更只會在您使用該工具的 EC2 執行個體上生效，不會影響其他執行個體，也不會影響環境日後的更新作業。本主題稍後提供的範例會說明如何透過指令碼和組態檔案呼叫 pkg-repo 命令，一次變更所有執行個體。

> **Warning**
>
> 大部分使用者皆不建議使用這項工具。系統會將您套用至解鎖平台版本的任何手動變更項目視為頻外行為。唯有當使用者在緊急情況下可以接受下列風險時，才可使用此選項：
>
> - 無法保證套件版本在環境中的所有執行個體之間都能保持一致。
> - 使用 pkg-repo 工具修改過的環境不保證能正常運作。這類環境尚未在 Elastic Beanstalk 支援的平台上完成測試和驗證。
>
> 我們強烈建議您採取最佳實務，包括執行測試及退出計畫。為協助採取最佳實務，您可使用 Elastic Beanstalk 主控台和 EB CLI 來複製環境及交換環境 URL。如需執行這些作業的詳細資訊，請參閱本指南中管理環境章節的藍/綠部署一節。

如果您打算手動編輯 yum 儲存庫組態檔案，請先執行 pkg-repo 工具。yum 儲存庫組態檔案經過手動編輯後，pkg-repo 工具可能無法在 Amazon Linux 2 環境中正常運作。這是因為工具可能無法辨識組態的變更項目。

如需有關 Amazon Linux Package 儲存庫的詳細資訊，請參閱 Amazon EC2 使用者指南中的套件儲存庫主題。

pkg-repo 命令

使用下列語法執行 pkg-repo 工具的命令。

```
$ /opt/elasticbeanstalk/bin/pkg-repo command [options]
```

該 pkg-repo 命令如下：

- lock — 將 yum 套件儲存庫鎖定至特定版本
- unlock — 將 yum 套件儲存庫與特定版本解除鎖定
- status — 列出所有 yum 套件儲存庫及其目前的鎖定狀態
- help — 顯示命令的一般說明或說明內容

這些選項對命令的適用情形如下：

- lock、unlock 和 status — 選項：-h、--help 或無 (預設)。

- `help` — 選項：`lock`、`unlock`、`status` 或無 (預設)。

以下範例執行 unlock 命令。

```
$ sudo /opt/elasticbeanstalk/bin/pkg-repo unlock
Amazon Linux 2 core package repo successfully unlocked
Amazon Linux 2 extras package repo successfully unlocked
```

以下範例執行 lock 命令。

```
$ sudo /opt/elasticbeanstalk/bin/pkg-repo lock
Amazon Linux 2 core package repo successfully locked
Amazon Linux 2 extras package repo successfully locked
```

以下範例執行 status 命令。

```
$ sudo /opt/elasticbeanstalk/bin/pkg-repo status
Amazon Linux 2 core package repo is currently UNLOCKED
Amazon Linux 2 extras package repo is currently UNLOCKED
```

下列範例針對 lock 命令執行 help 命令。

```
$ sudo /opt/elasticbeanstalk/bin/pkg-repo help lock
```

下列範例針對 `pkg-repo` 工具執行 help 命令。

```
$ sudo /opt/elasticbeanstalk/bin/pkg-repo help
```

您可以使用 SSH 連線到 Elastic Beanstalk 環境中的執行個體來測試 `pkg-repo`。一種 SSH 選項是 EB CLI eb ssh 命令。

> **Note**
>
> 這項 `pkg-repo` 工具需有根使用者的權限才能執行。如果您收到存取許可錯誤，請在 `sudo` 下重新執行命令。
> 在部署到環境的指令碼或組態檔案中使用工具時，不需要新增 `sudo`。Elastic Beanstalk 會以根使用者的身分執行所有指令碼。

pkg-repo 範例

上一節提供在 Elastic Beanstalk 環境個別 EC2 執行個體上進行測試的命令列範例。就測試作業而言，這種方法相當實用。不過，這種方法一次只能更新一個執行個體，因此不適合將變更項目套用至環境中的所有執行個體。

更務實的方法是使用平台勾點指令碼或 `.ebextensions` 組態檔案，以一致的方式將變更項目套用至所有執行個體。

下列範例呼叫 `.ebextensions` 資料夾中組態檔案的 `pkg-repo`。Elastic Beanstalk 會在您部署應用程式原始碼套件時，執行檔案中的 `update_package.config`。

```
.ebextensions
### update_package.config
```

若要接收最新版本的 docker 套件，此組態會在 yum update 命令中指定 docker 套件。

```
### update_package.config ###

commands:
  update_package:
    command: |
      /opt/elasticbeanstalk/bin/pkg-repo unlock
      yum update docker -y
      /opt/elasticbeanstalk/bin/pkg-repo lock
      yum clean all -y
      rm -rf /var/cache/yum
```

此組態不會在 yum update 命令中指定任何套件，因此所有可用的更新都會一併套用。

```
### update_package.config ###

commands:
  update_package:
    command: |
      /opt/elasticbeanstalk/bin/pkg-repo unlock
      yum update -y
      /opt/elasticbeanstalk/bin/pkg-repo lock
      yum clean all -y
      rm -rf /var/cache/yum
```

下列範例以平台勾點的形式呼叫 bash 指令碼的 pkg-repo。Elastic Beanstalk 執行 prebuild 子目錄中的 update_package.sh 指令碼檔案。

```
.platform
### hooks
    ### prebuild
        ### update_package.sh
```

若要接收最新版本的 docker 套件，此指令碼會在 yum update 命令中指定 docker 套件。如果省略套件名稱，則套用所有可用的更新。前一個組態檔案範例會示範這項操作。

```
### update_package.sh ###

#!/bin/bash

/opt/elasticbeanstalk/bin/pkg-repo unlock
yum update docker -y
/opt/elasticbeanstalk/bin/pkg-repo lock
yum clean all -y
rm -rf /var/cache/yum
```

download-source-bundle (僅限 Amazon Linux AMI)

Elastic Beanstalk 在 Amazon Linux AMI 平台分支 (前述的 Amazon Linux 2) 上提供一項額外工具：download-source-bundle。部署平台時，您可使用這項工具下載應用程式原始碼。此工具可於 /opt/elasticbeanstalk/bin/download-source-bundle 取得。

範例指令碼 00-unzip.sh 位於環境執行個體的 appdeploy/pre 資料夾中。這會示範如何在部署期間使用 download-source-bundle 將應用程式原始碼下載至 /opt/elasticbeanstalk/deploy/appsource 資料夾。

自 Docker 容器部署 Elastic Beanstalk 應用程式

本章介紹如何使用 Elastic Beanstalk 從 Docker 容器部署 Web 應用程序。Docker 容器可獨立自主運作，內含執行 Web 應用程式所需的所有組態資訊和軟體。使用 Docker 容器，您可以定義自己的運行時環境。您也可以選擇自己的程式設計語言和應用程式相依性，例如套件管理員或工具，這些程式通常不受其他 Elastic Beanstalk 平台支援。

請依照中QuickStart 對於碼頭工人的步驟建立 Docker「Hello World」應用程式，並使用 EB CLI 將其部署到 Elastic Beanstalk 環境。

主題

碼頭平台分支

Elastic Beanstalk Docker 平台支援下列平台分支：

執行於 Amazon Linux 2 的 Docker 和執行於 AL2023 的 Docker

Elastic Beanstalk 會將 Docker 容器和原始程式碼部署到 EC2 執行個體並加以管理。這些平台分支提供多容器支援。可以使用 Docker Compose 工具來簡化您的應用程式設定、測試和部署。如需此平台分支的詳細資訊，請參閱 the section called “Docker 平台分支”。

執行於 Amazon Linux 2 的 ECS 和執行於 AL2023 的 ECS

若客戶需要從執行於 (Amazon Linux AMI) 的已淘汰多容器 Docker 平台分支遷移至 AL2023/AL2 之路徑，我們會為客戶提供該分支。最新的平台分支支援已淘汰平台分支的所有功能。無需變更原始程式碼。如需詳細資訊，請參閱 將執行於 Amazon Linux 的多容器 Docker 遷移至執行於 Amazon Linux 2023 的 ECS。如果沒有在基於 ECS 平台分支上執行的 Elastic Beanstalk 環境，建議使用另一個平台分支，即在 64 位元 AL2023 上執行的 Docker。此方法較為簡單，且所需資源較少。

在 Amazon Linux AMI (AL1) 上執行的已淘汰平台分支

2022 年 7 月 18 日，Elastic Beanstalk 已將所有以 Amazon Linux AMI (AL1) 為基礎的平台分支狀態設為已淘汰。展開後文每個章節，可閱讀更多關於每個已淘汰平台分支及遷移至執行於 Amazon Linux 2 或 Amazon Linux 2023 (建議) 最新平台分支之路徑的詳細資訊。

Docker (Amazon Linux AMI)

此平台分支可部署 Docker 映像檔，如 Dockerfile 或 `Dockerrun.aws.json` v1 定義中所述。此平台分支針對每個執行個體只會執行一個容器。後續平台分支 (在 64 位元 AL2023 上執行的 Docker 和在 64 位元 Amazon Linux 2 上執行的 Docker) 支援每個執行個體擁有多個 Docker 容器。

建議使用更新且受支援的在 64 位元 AL2023 上執行的 Docker 平台分支來建立環境。然後，您就可以將應用程式遷移至新建立的環境。如需建立此類環境的詳細資訊，請參閱 the section called “Docker

平台分支”。如需遷移的詳細資訊，請參閱 將您的 Elastic Beanstalk Linux 應用程式遷移到 Amazon Linux 2023 或 Amazon Linux 2。

多容器 Docker (Amazon Linux AMI)

此平台分支使用 Amazon ECS 來協調將多個 Docker 容器部署至 Elastic Beanstalk 環境內的 Amazon ECS 叢集。如果您目前正在使用此已淘汰的平台分支，我們建議您遷移至執行於 Amazon Linux 2023 的最新 ECS 平台分支。最新的平台分支支援此已停產平台分支的所有功能。無需變更原始程式碼。如需詳細資訊，請參閱 將執行於 Amazon Linux 的多容器 Docker 遷移至執行於 Amazon Linux 2023 的 ECS。

預先設定的 Docker 容器

除了前面提到的 Docker 平台之外，還有在 Amazon Linux AMI 操作系統（AL1）上運行的預配置 Docker GlassFish 平台分支。

此平台分支已被在 64 位元 AL2023 上執行的 Docker 和在 64 位元 Amazon Linux 2 上執行的 Docker 平台分支所取代。如需詳細資訊，請參閱將 GlassFish 應用程式部署到 Docker 平台。

使用 Docker 平台分支

AWS Elastic Beanstalk 可以通過構建圖像中描述的圖像`Dockerfile`或提取遠程 Docker 映像來啟動 Docker 環境。若您要部署遠端 Docker 影像，則無須納入 `Dockerfile`。相反地，如果您也使用 Docker Compose，請使用 `docker-compose.yml` 檔案，該檔案指定要使用的映像和其他組態選項。如果您未將 Docker Compose 與 Docker 環境搭配使用，請改用 `Dockerrun.aws.json` 檔案。

主題

- QuickStart：將 Docker 應用程式部署到 Elastic Beanstalk
- Docker 組態
- 設定 Docker 環境

QuickStart：將 Docker 應用程式部署到 Elastic Beanstalk

本 QuickStart 教學課程將引導您完成建立 Docker 應用程式並將其部署到 AWS Elastic Beanstalk 環境的程序。

> **Note**
>
> 本 QuickStart 自學課程主要用於示範目的。請勿將本教學課程中建立的應用程式用於生產流量。

章節

您的 AWS 帳戶

如果您還不是 AWS 客戶，則需要創建一個 AWS 帳戶。註冊使您可以訪問 Elastic Beanstalk 和您需要的其他 AWS 服務。

如果您已經有 AWS 帳戶，則可以轉到必要條件。

創建一個 AWS 帳戶

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」。

必要條件

為了遵循本指南的程序，您需要命令列終端機或 shell 來執行命令。命令清單前會出現提示字元 ($) 及目前的目錄名稱 (如有)。

```
~/eb-project$ this is a command
this is output
```

在 Linux 和 macOS 上，您可以使用偏好的 Shell 和套件軟體管理工具。在視窗上，你可以安裝視窗子系統為 Linux 得到一個視窗集成的 Ubuntu 和 Bash 的版本。

EB CLI

本教學使用 Elastic Beanstalk 命令列界面 (EB CLI)。關於安裝和設定 EB CLI 的詳細資訊，請參閱安裝 EB CLI 和設定 EB CLI。

Docker

要按照本教程進行操作，您需要 Docker 的有效本地安裝。如需詳細資訊，請參閱 Docker 文件網站上的取得 Docker。

通過運行以下命令來驗證 Docker 守護進程是否正在運行。

```
~$ docker info
```

第 1 步：創建一個碼頭應用程序和容器

在此範例中，我們建立範例 Flask 應用程式的 Docker 影像，該影像也在中參考。[將 Flask 應用程式部署至 Elastic Beanstalk]

該應用程序由兩個文件組成：

- app.py— 包含將在容器中執行的程式碼的 Python 檔案。
- Dockerfile-碼頭文件來構建你的形象。

將這兩個檔案放在目錄的根目錄中。

```
~/eb-docker-flask/
|-- Dockerfile
|-- app.py
```

將下列內容新增至您的Dockerfile.

Example **~/eb-docker-flask/Dockerfile**

```
FROM python:3.12
COPY . /app
WORKDIR /app
RUN pip install Flask==3.0.2
EXPOSE 5000
CMD [ "python3", "-m" , "flask", "run", "--host=0.0.0.0"]
```

將以下內容添加到您的app.py文件中。

Example **~/eb-docker-flask/app.py**

```
from flask import Flask
app = Flask(__name__)
@app.route('/')
def hello_world():
    return 'Hello Elastic Beanstalk! This is a Docker application'
```

構建您的 Docker 容器，使eb-docker-flask用標記圖像。

```
~/eb-docker-flask$ docker build -t eb-docker-flask
```

步驟 2：在本機執行應用程式

使用 docker build 命令在本地構建容器映像，並使用標記圖像。eb-docker-flask命令末尾的句號（.）特定路徑是本地目錄。

```
~/eb-docker-flask$ docker run -dp 127.0.0.1:5000:5000 eb-docker-flask .
```

使用 docker 運行命令運行容器。該命令將打印正在運行的容器的 ID。該-d選項在後台模式下運行 docker。-p此選項會在 5000 連接埠公開您的應用程式。默認情況下，Elastic Beanstalk 為碼頭平台上的端口 5000 提供流量。

```
~/eb-docker-flask$ docker run -dp 127.0.0.1:5000:5000 eb-docker-flask container-id
```

在瀏覽器http://127.0.0.1:5000/ 中導覽至。你應該看到文字「你好 Elastic Beanstalk！這是一個碼頭應用程序「。

運行 docker 殺死命令終止容器。

```
~/eb-docker-flask$ docker kill container-id
```

步驟 3：使用 EB CLI 部署您的泊塢視窗應用程式

執行下列命令，為此應用程式建立 Elastic Beanstalk 環境。

若要建立環境並部署您的 Docker 應用程式

1. 透過 eb init 命令初始化您的 EB CLI 儲存庫。

   ```
   ~/eb-docker-flask$ eb init -p docker docker-tutorial us-east-2
   Application docker-tutorial has been created.
   ```

   此命令創建一個名為的應用程序，docker-tutorial並配置您的本地存儲庫以創建具有最新 Docker 平台版本的環境。
2. (選用) 再次執行 eb init 來設定預設金鑰對，藉此使用 SSH 連接至執行您應用程式的 EC2 執行個體：

   ```
   ~/eb-docker-flask$ eb init
   Do you want to set up SSH for your instances?
   ```

```
(y/n): y
Select a keypair.
1) my-keypair
2) [ Create new KeyPair ]
```

若您已有金鑰對，請選擇一對，或依照提示建立金鑰對。若未出現提示，或稍後需要變更設定，請執行 eb init -i。

3. 使用 eb create 建立環境並於其中部署您的應用程式。Elastic Beanstalk 會自動為您的應用程序構建一個 zip 文件，並在端口 5000 上啟動它。

```
~/eb-docker-flask$ eb create docker-tutorial
```

Elastic Beanstalk 大約需要五分鐘的時間來創建您的環境。

第 4 步：在 Elastic Beanstalk 上運行應用程序

當創建環境的過程完成後，打開您的網站eb open。

```
~/eb-docker-flask$ eb open
```

恭喜您！您已經部署了帶有 Elastic Beanstalk 的 Docker 應用程序！這會開啟瀏覽器視窗，並使用為應用程式建立的網域名稱。

步驟 5：清除

您可以在完成應用程式的工作後終止環境。Elastic Beanstalk 會終止與您的環境相關聯的所有 AWS 資源。

若要使用 EB CLI 終止 Elastic Beanstalk 環境，請執行下列命令。

```
~/eb-docker-flask$ eb terminate
```

AWS 您應用程式的資源

您剛剛建立了單一執行個體應用程式。它可作為單一 EC2 執行個體的簡單範例應用程式使用，因此不需要負載平衡或 auto 擴展。對於單個實例應用程序，Elastic Beanstalk 創建以下 AWS 資源：

- EC2 執行個體 – 設定在您所選平台上執行 Web 應用程式的 Amazon EC2 虛擬機器。

每個平台會執行不同一套軟體、設定檔和指令碼，來支援特定的語言版本、架構、Web 容器或其組合。大多數的平台會使用 Apache 或 nginx 做為反向代理，處理您 Web 應用程式前端的網路流量、向它轉送請求、提供靜態資產，並產生存取和錯誤日誌。

- 執行個體安全群組 – 設定允許從連接埠 80 傳入流量的 Amazon EC2 安全群組。此資源可讓負載平衡器傳來的 HTTP 傳輸資料，到達執行您 Web 應用程式的 EC2 執行個體。在預設情況下，不允許傳輸資料從其他通訊埠傳送。
- Amazon S3 儲存貯體 - 儲存位置，用來儲存當您使用 Elastic Beanstalk 時所建立的原始程式碼、日誌和其他成品。
- Amazon CloudWatch 警示 — 兩個 CloudWatch 警示，用於監控環境中執行個體的負載，並在負載過高或過低時觸發。當警示觸發時，您的 Auto Scaling 群組會擴展或縮減以進行回應。
- AWS CloudFormation 堆疊 — Elastic Beanstalk 用 AWS CloudFormation 來啟動環境中的資源並傳播組態變更。資源定義於範本中，您可在 AWS CloudFormation 主控台中檢視此範本。
- 網域名稱 – 會路由到您 Web 應用程式的網域名稱，其格式為 *subdomain*.*region*.elasticbeanstalk.com。

Elastic Beanstalk 會管理所有這些資源。當您終止環境時，Elastic Beanstalk 會終止其中的所有資源。

後續步驟

在您擁有執行應用程式的環境後，可以隨時部署應用程式的新版本或不同的應用程式。部署新的應用程式版本非常快速，因無須佈建或重新啟動 EC2 執行個體。您也可以使用 Elastic Beanstalk 控制台探索您的新環境。如需詳細步驟，請參閱本指南的「入門」一章中的「探索您的環境」。

在您部署一或兩個範例應用程式並準備好開始在本機開發和執行 Docker 應用程式之後，請參閱

使用 Elastic Beanstalk 控制台進行部署

您也可以使用彈性魔豆控制台來啟動範例應用程式。如需詳細步驟，請參閱本指南的「入門」一章中的「建立範例應用程式」。

Docker 組態

本章節說明如何針對 Elastic Beanstalk 的部署準備 Docker 影像和容器。

使用 Docker Compose 的 Docker 環境

本章節說明如何針對 Elastic Beanstalk 的部署準備 Docker 影像和容器。如果您也使用 Docker Compose 工具，在 Docker 環境中部署到 Elastic Beanstalk 的任何 Web 應用程式必須包含 `docker-`

compose.yml 檔案。您可以執行下列其中一個動作，將 Web 應用程式做為容器化服務部署至 Elastic Beanstalk：

- 建立 docker-compose.yml 檔案，以將 Docker 影像從託管儲存庫部署至 Elastic Beanstalk。如果所有部署皆源自於公有儲存庫中的映像，則不需要其他檔案。(如果您的部署必須取用私有儲存庫的映像，則需要包含其他組態檔案以進行身分驗證。如需詳細資訊，請參閱使用私有儲存庫中的映像。) 如需 docker-compose.yml 檔案的詳細資訊，請參閱 Docker 網站上的 Compose 檔案參考。
- 建立 Dockerfile，以擁有 Elastic Beanstalk 組建，並執行自訂映像。根據您的部署需求，此檔案是選用的。如需有關 Dockerfile 的詳細資訊，請參閱 Docker 網站上的 Dockerfile 參考。
- 建立 .zip 檔案，其中納入您的應用程式檔案、任何應用程式檔案依存項目、Dockerfile 及 docker-compose.yml 檔案。如果您使用 EB CLI 來部署應用程式，它會為您建立 .zip 檔案。這兩個檔案務必位於 .zip 封存檔的根目錄或頂層目錄。

  如果您指使用 docker-compose.yml 檔案來部署應用程式，則不需要建立 .zip 檔案。

這個主題是語法參考。如需使用 Elastic Beanstalk 啟動 Docker 環境的詳細程序，請參閱使用 Docker 平台分支。

若要進一步了解 Docker Compose 及如何安裝，請參閱 Docker 網站的 Docker Compose 的概觀和安裝 Docker Compose。

> **Note**
>
> 如果您未使用 Docker Compose 來設定 Docker 環境，那麼您也不應該使用 docker-compose.yml 檔案。相反地，請使用 Dockerrun.aws.json 檔案或 Dockerfile，或兩者。
> 如需更多詳細資訊，請參閱 the section called “Docker 平台的組態 (沒有 Docker Compose)”。

使用私有儲存庫中的映像

Elastic Beanstalk 必須先使用託管私有儲存庫的線上登錄進行驗證，然後才能從私有儲存庫提取和部署映像。我們提供兩個選項的範例，這些選項可儲存和擷取 Elastic Beanstalk 環境的登入資料，以便對儲存庫進行驗證。

- 該 AWS Secrets Manager

- `Dockerrun.aws.json` v3 檔案

使用 AWS Secrets Manager

您可以先設定 Elastic Beanstalk 以登入私有儲存庫，然後開始部署程序。這可讓 Elastic Beanstalk 存取儲存庫的映像，並將這些映像部署至您的 Elastic Beanstalk 環境。

此組態會在 Elastic Beanstalk 部署程序的「預先建置」階段啟動事件。您可以在 .ebextentions 組態目錄中進行設定。組態使用平台勾點指命碼，該指令碼會呼叫 docker login，以針對託管私有儲存庫的線上登錄進行驗證。這些組態步驟的詳細明細如下。

使用 AWS Secrets Manager 設定 Elastic Beanstalk 以針對您的私有儲存庫進行身分驗證

> **Note**
>
> 必須授與特定許可才能完成這些步驟。如需詳細資訊，請參閱如下參考。
>
> - 在步驟 2 中，您需要建立秘密的許可。如需詳細資訊，請參閱 AWS Secrets Manager 使用者指南中的 Example: Permission to retrieve secret (範例：擷取秘密的許可)。
> - 在步驟 3 中，您需要使用 `secretsmanager` 動態參考擷取秘密的許可。如需詳細資訊，請參閱《AWS Secrets Manager 使用者指南》中的 Example: Permission to retrieve secret values (範例：擷取秘密值的許可)。

1. 建立您的 `.ebextensions` 目錄結構，如下所示。

```
### .ebextensions
#   ### env.config
### .platform
#   ### confighooks
#   #   ### prebuild
#   #       ### 01login.sh
#   ### hooks
#       ### prebuild
#           ### 01login.sh
### docker-compose.yml
```

2. 用 AWS Secrets Manager 於儲存私人存放庫的認證，以便 Elastic Beanstalk 可以在需要時擷取您的認證。為此，請運行 Secrets Manager 創建 AWS CLI 密鑰命令。

```
aws secretsmanager create-secret \
            --name MyTestSecret \
            --description "My image repo credentials created with the CLI." \
            --secret-string "{\"USER\":\"EXAMPLE-USERNAME\",\"PASSWD\":\"EXAMPLE-
PASSWORD\"}"
```

3. 建立以下 env.config 檔案，並將其放置在 .ebextensions 目錄中，如前面的目錄結構所示。此組態會使用 aws:elasticbeanstalk:application:environment 命名空間，藉助動態參考將 USER 和 PASSWD Elastic Beanstalk 環境變數初始化為 AWS Secrets Manager。有關secretsmanager動態參考的詳細資訊，請參閱《AWS Secrets Manager 使用指南》中的〈擷取 AWS CloudFormation 資源中的 AWS Secrets Manager 密碼〉。

   > **Note**
   >
   > 指令碼中的 USER 和 PASSWD 必須符合上述 secretsmanager create-secret 命令中使用的相同字串。

```
option_settings:
  aws:elasticbeanstalk:application:environment:
    USER: '{{resolve:secretsmanager:MyTestSecret:SecretString:USER}}'
    PASSWD: '{{resolve:secretsmanager:MyTestSecret:SecretString:PASSWD}}'
```

4. 建立下列 01login.sh 指令碼檔案，並將其放置在以下目錄中 (也如同前面的目錄結構所示)：

   - .platform/confighooks/prebuild
   - .platform/hooks/prebuild

```
### example 01login.sh
#!/bin/bash
USER=/opt/elasticbeanstalk/bin/get-config environment -k USER
/opt/elasticbeanstalk/bin/get-config environment -k PASSWD | docker login -u $USER
 --password-stdin
```

   01login.sh 指令碼會呼叫 get-config 平台指令碼以擷取儲存庫憑證，然後將日誌放入儲存庫中。其將使用者名稱儲存在 USER 指令碼變數中。在下一行中，會檢索密碼。指令碼不會將密碼儲存在指令碼變數中，而是將密碼直接傳送至 stdin 輸入串流中的 docker login 命令。--

`password-stdin` 選項使用輸入串流，因此您不必將密碼儲存在變數中。如需有關使用 Docker 命令列界面進行驗證的詳細資訊，請參閱 Docker 文件網站中的《Docker 登入》。

> **備註**
>
> - 所有指令碼檔案必須有執行許可。使用 chmod +x 在勾點檔案上設定執行權限。對於在 2022 年 4 月 29 日或之後發佈的所有以 Amazon Linux 2 為基礎的平台版本，Elastic Beanstalk 會自動授予執行許可給所有平台勾點指令碼。在此情況下，您不需要手動授予執行許可。如需這些平台版本的清單，請參閱 AWS Elastic Beanstalk 版本備註指南中的 2022 年 4 月 29 日 - Linux 平台版本備註。
> - 可執行檔也可以是二進位檔案，或開頭為 #! 行且包含其解譯器路徑 (例如 #!/bin/bash) 的指令碼檔案。
> - 如需詳細資訊，請參閱 Extending Elastic Beanstalk Linux 平台中的 the section called “平台勾點”。

Elastic Beanstalk 使用託管私有儲存庫的線上登錄進行驗證之後，即可提取和部署您的映像。

使用 `Dockerrun.aws.json v3` 檔案

本節說明另一種對私有儲存庫 Elastic Beanstalk 進行驗證的方法。使用這種方法，您可以使用 Docker 命令產生身分驗證檔案，然後將驗證檔案上傳到 Amazon S3 儲存貯體。您也必須在 `Dockerrun.aws.json v3` 檔案中包含儲存貯體資訊。

若要產生身分驗證檔案並提供給 Elastic Beanstalk

1. 透過 docker login 命令產生身分驗證檔案。若是 Docker Hub 上的儲存庫，請執行 docker login：

   ```
   $ docker login
   ```

   若是其他登錄檔，請納入登錄伺服器的 URL：

   ```
   $ docker login registry-server-url
   ```

> **Note**
>
> 如果您的 Elastic Beanstalk 環境使用 Amazon Linux AMI 平台版本 (Amazon Linux 2 之前的版本)，請閱讀 the section called “Amazon Linux AMI (Amazon Linux 2 之前的版本) 上的 Docker 組態” 中的相關資訊。

如需身分驗證檔案的詳細資訊，請至 Docker 網站參閱 Store images on Docker Hub 和 docker login。

2. 將名為 `.dockercfg` 的身分驗證檔案副本上傳至安全的 Amazon S3 儲存貯體。

   - Amazon S3 儲存貯體必須託管在與使用該儲 AWS 區域 存貯體的環境相同。Elastic Beanstalk 無法從其他區域託管的 Amazon S3 儲存貯體下載檔案。
   - 針對執行個體設定檔中的 IAM 角色，授予 `s3:GetObject` 操作的許可。如需更多詳細資訊，請參閱 管理 Elastic Beanstalk 執行個體描述檔。

3. 將 Amazon S3 儲存貯體資訊納入 `Authentication` 檔案中的 `Dockerrun.aws.json v3` 參數。

   以下是 `Dockerrun.aws.json v3` 檔案的範例。

```
{
  "AWSEBDockerrunVersion": "3",
  "Authentication": {
    "bucket": "DOC-EXAMPLE-BUCKET",
    "key": "mydockercfg"
  }
}
```

> **Note**
>
> `AWSEBDockerrunVersion` 參數表示 `Dockerrun.aws.json` 檔案的版本。
>
> - Docker Amazon Linux 2 平台會針對使用 Docker Compose 的環境使用 `Dockerrun.aws.json v3` 檔案。它會針對未使用 Docker Compose 的環境使用 `Dockerrun.aws.json v1` 檔案。
> - 多容器 Docker Amazon Linux AMI 平台使用 `Dockerrun.aws.json v2` 檔案。

Elastic Beanstalk 可以使用託管私有儲存庫的線上登錄進行驗證之後，即可部署和提取您的映像。

透過 Dockerfile 建置自訂映像

若您尚未於儲存庫內託管現有映像，您需要建立 Dockerfile。

下列程式碼片段為 Dockerfile 的範例。遵循使用 Docker 平台分支中的說明時，可按撰寫內容上傳此 Dockerfile。若您使用此 Dockerfile，Elastic Beanstalk 會執行遊戲 2048。

如需可納入 Dockerfile 之指示的詳細資訊，請參閱 Docker 網站的 Dockerfile 參考。

```
FROM ubuntu:12.04

RUN apt-get update
RUN apt-get install -y nginx zip curl

RUN echo "daemon off;" >> /etc/nginx/nginx.conf
RUN curl -o /usr/share/nginx/www/master.zip -L https://codeload.github.com/
gabrielecirulli/2048/zip/master
RUN cd /usr/share/nginx/www/ && unzip master.zip && mv 2048-master/* . && rm -rf 2048-
master master.zip

EXPOSE 80

CMD ["/usr/sbin/nginx", "-c", "/etc/nginx/nginx.conf"]
```

> **Note**
>
> 您可以從單一 Dockerfile 執行多階段建置，以產生較小的映像，同時顯著降低複雜性。如需詳細資訊，請參閱 Docker 文件網站上的使用多階段建置。

Docker 平台的組態 (沒有 Docker Compose)

如果您的 Elastic Beanstalk Docker 環境未使用 Docker Compose，請閱讀以下章節中的其他資訊。

Docker 平台組態 - 不含 Docker Compose

您部署至 Docker 環境之 Elastic Beanstalk 的任何 Web 應用程式必須包含 Dockerfile 或 Dockerrun.aws.json 檔案。您可執行下列其中一項動作，從 Docker 容器將 Web 應用程式部署至 Elastic Beanstalk：

- 建立 Dockerfile，以擁有 Elastic Beanstalk 組建，並執行自訂映像。

- 建立 Dockerrun.aws.json 檔案，以將 Docker 影像從託管儲存庫部署至 Elastic Beanstalk。
- 建立 .zip 檔案，其中納入您的應用程式檔案、任何應用程式檔案依存項目、Dockerfile 及 Dockerrun.aws.json 檔案。如果您使用 EB CLI 來部署應用程式，它會為您建立 .zip 檔案。

  若您只使用 Dockerfile 或 Dockerrun.aws.json 檔案來部署應用程式，無須建立 .zip 檔案。

這個主題是語法參考。如需啟動 Docker 環境的詳細程序，請參閱使用 Docker 平台分支。

Dockerrun.aws.json v1

Dockerrun.aws.json 檔案說明如何將遠端 Docker 影像部署為 Elastic Beanstalk 應用程式。此 JSON 檔案專屬 Elastic Beanstalk。若您的應用程式執行於託管儲存庫提供的映像上，您可於 Dockerrun.aws.json v1 檔案指定該映像並省略 Dockerfile。

Dockerrun.aws.json v1 檔案的有效金鑰和值包括以下操作：

AWSEBDockerrunVersion

(必要) 將單一容器 Docker 環境的版本編號指定為值 1。

身分驗證

(僅私有儲存庫為必要) 指定存放 .dockercfg 檔案的 Amazon S3 物件。

請參閱 使用私有儲存庫中的映像。

映像

指定將從中建置 Docker 容器之現有 Docker 儲存庫的 Docker 基礎映像名稱。指定 Name (名稱) 金鑰值：Docker Hub 上的映像採用 *<organization>/<image name>* 格式，其他網站則使用 *<site>/<organization name>/<image name>* 的格式。

當您在 Dockerrun.aws.json 檔案中指定映像時，Elastic Beanstalk 環境中的每個執行個體都會執行 docker pull 以執行映像。您亦可選擇納入 Update (更新) 金鑰。預設值為 true，且會指示 Elastic Beanstalk 檢查儲存庫、叫出映像更新並覆寫快取映像。

使用 Dockerfile 時，請勿於 Dockerrun.aws.json 檔案中指定 Image (映像) 金鑰。若 Dockerfile 描述了映像，Elastic Beanstalk 永遠會依此建置並加以使用。

連接埠

(指定 Image (映像) 金鑰時為必要) 列出在 Docker 容器上公開的連接埠。Elastic Beanstalk 使用該ContainerPort值將 Docker 容器連接到主機上運行的反向代理。

您可以指定多個容器連接埠，但 Elastic Beanstalk 只會使用第一個連接埠。其使用此連接埠將您的容器連接至主機的反向代理程式，並路由來自公有網際網路的請求。如果您使用的是`Dockerfile`，則第一個ContainerPort值應與 EXPOSE 清單中`Dockerfile`的第一個項目相符。

或者，您可以在中指定連接埠清單HostPort。HostPort項目會指定ContainerPort值對應至的主機連接埠。如果您沒有指定HostPort值，它會預設為該ContainerPort值。

```
{
  "Image": {
    "Name": "image-name"
  },
  "Ports": [
    {
      "ContainerPort": 8080,
      "HostPort": 8000
    }
  ]
}
```

磁碟區

將磁碟區從 EC2 執行個體對應至您的 Docker 容器。指定欲對應的一個或多個磁碟區陣列。

```
{
  "Volumes": [
    {
      "HostDirectory": "/path/inside/host",
      "ContainerDirectory": "/path/inside/container"
    }
  ]
...
```

日誌

在容器內指定應用程式寫入日誌的目錄。當您請求結尾或套件日誌時，Elastic Beanstalk 會將此目錄的日誌上傳至 Amazon S3。若您將日誌輪換至此目錄內名為 `rotated` 的資料夾，亦可設定 Elastic Beanstalk 將輪換日誌上傳至 Amazon S3 供永久儲存。如需更多詳細資訊，請參閱 [在 Elastic Beanstalk 環境中檢視 Amazon EC2 執行個體的日誌]。

命令

指定要在容器中執行的命令。如果您指定 Entrypoint，然後會將 Command 新增做為對 Entrypoint 的引數。如需詳細資訊，請參閱 Docker 文件中的 CMD。

進入點

指定在容器啟動時要執行的預設命令。如需詳細資訊，請參閱 Docker 文件中的 ENTRYPOINT。

下列程式碼片段範例說明單一容器的 Dockerrun.aws.json 檔案之語法。

```
{
  "AWSEBDockerrunVersion": "1",
  "Image": {
    "Name": "janedoe/image",
    "Update": "true"
  },
  "Ports": [
    {
      "ContainerPort": "1234"
    }
  ],
  "Volumes": [
    {
      "HostDirectory": "/var/app/mydb",
      "ContainerDirectory": "/etc/mysql"
    }
  ],
  "Logging": "/var/log/nginx",
  "Entrypoint": "/app/bin/myapp",
  "Command": "--argument"
}
```

您可只向 Elastic Beanstalk 提供 Dockerrun.aws.json 檔案，或內含 .zip 及 Dockerrun.aws.json 檔案的 Dockerfile 封存檔。在提供這兩份檔案時，Dockerfile 描述 Docker 影像，Dockerrun.aws.json 檔案則提供其他部署資訊，如本章節稍後所述。

> **Note**
>
> 這兩個檔案務必位於 .zip 封存檔的根目錄或頂層目錄。請勿從包含這些檔案的目錄建立封存檔。相反地，請瀏覽至該目錄內並從中建立封存檔。

提供這兩個檔案時，請勿於 Dockerrun.aws.json 檔案內指定映像。Elastic Beanstalk 會建立 Dockerfile 所述的映像並加以使用，而忽略 Dockerrun.aws.json 檔案指定的映像。

使用私有儲存庫中的映像

將內含身分驗證檔案的 Amazon S3 儲存貯體相關資訊，新增至 Authentication 檔案的 Dockerrun.aws.json v1 參數。請確認 Authentication 參數包含有效的 Amazon S3 儲存貯體和金鑰。託管 Amazon S3 儲存貯體的區域，必須與使用所在環境處於相同的 AWS 區域 。Elastic Beanstalk 不會從其他區域託管的 Amazon S3 儲存貯體下載檔案。

如需產生身分驗證檔案並加以上傳的詳細資訊，請參閱 使用私有儲存庫中的映像。

下列範例說明如何使用 mydockercfg 儲存貯體內名為 DOC-EXAMPLE-BUCKET 的身分驗證檔案，以使用第三方登錄檔內的私有映像。

```
{
  "AWSEBDockerrunVersion": "1",
  "Authentication": {
    "Bucket": "DOC-EXAMPLE-BUCKET",
    "Key": "mydockercfg"
  },
  "Image": {
    "Name": "quay.io/johndoe/private-image",
    "Update": "true"
  },
  "Ports": [
    {
      "ContainerPort": "1234"
    }
  ],
  "Volumes": [
    {
      "HostDirectory": "/var/app/mydb",
      "ContainerDirectory": "/etc/mysql"
    }
  ],
  "Logging": "/var/log/nginx"
}
```

設定 Docker 環境

有幾種方法可以配置 Elastic Beanstalk Docker 環境的行為。

> **Note**
>
> 如果您的 Elastic Beanstalk 環境使用 Amazon Linux AMI 平台版本 (Amazon Linux 2 之前的版本)，請閱讀 the section called “Amazon Linux AMI (Amazon Linux 2 之前的版本) 上的 Docker 組態” 中的其他資訊。

章節

在 Docker 環境中設定軟體

您可使用 Elastic Beanstalk 主控台來設定於您環境的執行個體上執行的軟體。

在 Elastic Beanstalk 主控台中設定 Docker 環境

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

   > **Note**
   >
   > 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇組態。

4. 在更新、監控和日誌記錄組態類別中，選擇編輯。
5. 進行必要的組態變更。
6. 若要儲存變更，請選擇頁面底部的儲存變更。

如需有關在任何環境中配置軟體設定的資訊，請參閱 the section called “環境屬性與軟體設定”。以下各節介紹 Docker 的特定資訊。

容器選項

Container options (容器選項) 區段具有平台特定選項。對於 Docker 環境而言，它可讓您選擇您的環境是否包含 NGINX 代理伺服器。

使用 Docker Compose 的環境

如果您使用 Docker Compose 來管理 Docker 環境，則 Elastic Beanstalk 會假定您以容器身分來執行代理伺服器。因此，對於 Proxy server (代理伺服器) 設定，其預設為 None (無)，而且 Elastic Beanstalk 不會提供 NGINX 組態。

> **Note**
>
> 即使您選取 NGINX 做為代理伺服器，在使用 Docker Compose 的環境中也會忽略此設定。Proxy server (代理伺服器) 設定仍預設為 None (無)。

由於使用 Docker Compose 的 Amazon Linux 2 平台上的 Docker 已停用 NGINX Web 伺服器 Proxy，因此，您必須依照指示產生增強型運作狀態報告的日誌。如需更多詳細資訊，請參閱 產生增強型運作狀態報告的日誌 (Docker Compose)。

環境屬性和環境變數

Environment Properties (環境屬性) 的部分可讓您針對執行您應用程式的 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體，來指定其上的環境資訊設定。環境屬性會以金鑰值對的形式傳到應用程式。在 Docker 環境中，Elastic Beanstalk 會將環境屬性傳遞給容器做為環境變數。

您在容器中執行的應用程式的程式碼，可以依名稱參照環境變數並讀取其值。讀取這些環境變數的原始程式碼會因程式語言而有所不同。您可以在個別平台主題中，找到在 Elastic Beanstalk 受管平台支援之程式設計語言中讀取環境變數值的指示。如需這些主題的連結清單，請參閱 the section called “環境屬性與軟體設定”。

使用 Docker Compose 的環境

如果您使用 Docker Compose 來管理 Docker 環境，您必須進行一些額外的設定，以擷取容器中的環境變數。若要讓容器中執行的可執行檔存取這些環境變數，您必須在 `docker-compose.yml` 中參考這些變數。如需詳細資訊，請參閱 在容器中參考環境變數。

在容器中參考環境變數

如果您在 Amazon Linux 2 Docker 平台上使用 Docker Compose 工具，Elastic Beanstalk 會在應用程式專案的根目錄中產生名為 `.env` 的 Docker Compose 環境。此檔案會存放針對 Elastic Beanstalk 設定的環境變數。

> **Note**
>
> 如果您在應用程式套件中加入 `.env` 檔案，Elastic Beanstalk 將不會產生 `.env` 檔案。

為了讓容器參考您在 Elastic Beanstalk 中定義的環境變數，您必須遵循其中一種或兩種設定方法。

- 將 Elastic Beanstalk 產生的 `.env` 檔案新增至 `env_file` 檔案中的 `docker-compose.yml` 組態選項。
- 直接定義 `docker-compose.yml` 檔案中的環境變數。

以下檔案提供範例。範例 `docker-compose.yml` 檔案會示範這兩種方法。

- 如果您定義環境屬性 `DEBUG_LEVEL=1` 和 `LOG_LEVEL=error`，Elastic Beanstalk 會為您產生以下 `.env` 檔案：

```
DEBUG_LEVEL=1
LOG_LEVEL=error
```

- 在此 `docker-compose.yml` 檔案中，`env_file` 組態選項會指向 `.env` 檔案，而且它也會在 `DEBUG=1` 檔案中直接定義環境變數 `docker-compose.yml`。

```
services:
  web:
    build: .
    environment:
      - DEBUG=1
    env_file:
      - .env
```

備註

- 如果您在兩個檔案中設定相同的環境變數，則 `docker-compose.yml` 檔案中定義之變數的優先順序高於 `.env` 檔案中定義的變數。
- 請小心不要在等號 (=) 和指派給變數的值之間留下空格，以防止空格加入字串中。

若要進一步了解 Docker Compose 中的環境變數，請參閱 Compose 中的環境變數

對環境變數使用插補功能 (Docker Compose)

從 2023 年 7 月 28 日的平台發行開始，Docker Amazon Linux 2 平台分支提供 Docker Compose 插補功能。使用此功能，Compose 檔案中的值可由變數設定，並在執行期進行插補。如需有關此功能的詳細資訊，請參閱 Docker 文件網站上的插補。

Important

如果想搭配使用此功能與您的應用程式，請注意，您需要實作使用平台勾點的方法。這是必要的，因為我們在平台引擎中實作了緩解措施。這種緩解措施可讓不知道新插補功能，且擁有使用環境變數搭配 $ 字元之現有應用程式的客戶確保回溯相容性。預設情況下，更新後的平台引擎透過將 $ 字元取代為 $$ 字元來轉義插補。

以下是平台勾點指令碼範例，您可以將其設定為允許使用插補功能。

```
#!/bin/bash

: '
example data format in .env file
key1=value1
key2=value2
'
envfile="/var/app/staging/.env"
tempfile=$(mktemp)

while IFS= read -r line; do
  # split each env var string at '='
  split_str=(${line//=/ })
  if [ ${#split_str[@]} -eq 2 ]; then
    # replace '$$' with '$'
```

```
    replaced_str=${split_str[1]//\$\$/\$}
    # update the value of env var using ${replaced_str}
    line="${split_str[0]}=${replaced_str}"
  fi
  # append the updated env var to the tempfile
  echo "${line}" #"${tempfile}"
done < "${envfile}"
# replace the original .env file with the tempfile
mv "${tempfile}" "${envfile}"
```

將平台勾點放在這兩個目錄下：

- `.platform/confighooks/predeploy/`
- `.platform/hooks/predeploy/`

如需詳細資訊，請參閱本指南的擴充 Linux 平台主題中的 平台勾點。

產生增強型運作狀態報告的日誌 (Docker Compose)

Elastic Beanstalk 運作狀態代理程式提供 Elastic Beanstalk 環境的作業系統和應用程式運作狀態指標。其依賴以特定格式轉送資訊的 Web 伺服器日誌格式。

Elastic Beanstalk 假設您以容器身分來執行 Web 伺服器 Proxy。因此，針對執行 Docker Compose 的 Docker 環境，已停用 NGINX Web 伺服器 Proxy。您必須將伺服器設為以 Elastic Beanstalk 運作狀態代理程式所使用的位置和格式來寫入日誌。這麼做可讓您即使在已停用 Web 伺服器 Proxy 的狀態下，也能充分利用增強型運作狀態報告。

如需如何執行此動作的詳細資訊，請參閱 Web 伺服器日誌組態

Docker 容器自訂記錄 (Docker Compose)

為了有效疑難排解問題，並監控您的容器化服務，您可以透過環境管理主控台或 EB CLI，從 Elastic Beanstalk 要求執行個體日誌。執行個體日誌由套件日誌和結尾日誌組成，經過組合和封裝，可讓您以有效且直接的方式，檢視日誌和最近的事件。

Elastic Beanstalk 會在容器執行個體上建立日誌目錄，針對 `docker-compose.yml` 檔案中定義的每個服務建立一個日誌目錄，位置是 `/var/log/eb-docker/containers/`*`<service name>`*。如果您在 Amazon Linux 2 Docker 平台上使用 Docker Compose 功能，您可以將這些目錄掛載到容器檔案結構中寫入日誌的位置。當您掛載日誌目錄以寫入日誌資料時，Elastic Beanstalk 可從這些目錄收集日誌資料。

如果您的應用程式位於未使用 Docker Compose 的 Docker 平台上，則可以遵循 Docker 容器自訂記錄 (Docker Compose) 中所述的標準程序。

若要將服務的日誌檔案設為可擷取的結尾檔案和套件日誌

1. 編輯 docker-compose.yml 檔案。
2. 在您服務的 volumes 金鑰下，將綁定掛載新增至以下內容：

   "${EB_LOG_BASE_DIR}/*<service name>*:*<log directory inside container>*

   在以下範例 docker-compose.yml 檔案中：

   - nginx-proxy 是 *<####>*
   - /var/log/nginx 是 *<########>*

```
services:
  nginx-proxy:
    image: "nginx"
    volumes:
      - "${EB_LOG_BASE_DIR}/nginx-proxy:/var/log/nginx"
```

- var/log/nginx 目錄包含容器中 nginx-proxy 服務的日誌，其將映射至主機上的 /var/log/eb-docker/containers/nginx-proxy 目錄。
- 此目錄中的所有日誌現在可以透過 Elastic Beanstalk 的請求執行個體日誌功能，以套件和結尾日誌的形式擷取。

> **備註**
>
> - ${EB_LOG_BASE_DIR} 是 Elastic Beanstalk 設定的環境變數，具有數值 /var/log/eb-docker/containers。
> - Elastic Beanstalk 會針對 /var/log/eb-docker/containers/*<service name>* 檔案中的每個服務，自動建立 docker-compose.yml 目錄。

Docker 影像

Elastic Beanstalk 的 Docker 和 ECS 受管 Docker 平台分支可支援使用存放於公有或私有線上映像儲存庫的 Docker 影像。

根據 `Dockerrun.aws.json` 中的名稱來指定映像。請注意這些慣例：

- Docker Hub 上官方儲存庫中的映像，使用的是單一名稱 (例如，`ubuntu` 或 `mongo`)。
- Docker Hub 上的其他儲存庫中的映像要求使用組織名稱 (例如，`amazon/amazon-ecs-agent`)。
- 其他線上儲存庫中的映像更進一步要求使用網域名稱 (例如，`quay.io/assemblyline/ubuntu` 或 *`account-id`*`.dkr.ecr.us-east-2.amazonaws.com/ubuntu:trusty`)。

您也可以在環境建立期間，使用 Dockerfile 建置自己的映像。這僅適用於使用 Docker 平台的環境。如需詳細資訊，請參閱「透過 Dockerfile 建置自訂映像」。多容器 Docker 平台無法支援此功能。

使用 Amazon ECR 儲存庫中的映像

您可以 AWS 使用 Amazon 彈性容器註冊表（Amazon ECR）存儲自定義碼頭映像。若將 Docker 影像存放於 Amazon ECR，Elastic Beanstalk 會自動透過您環境的執行個體描述檔向 Amazon ECR 登錄檔進行驗證，因此您無須產生身分驗證檔案並將其上傳至 Amazon Simple Storage Service (Amazon S3)。

不過，您必須於環境的執行個體設定檔新增許可，讓您的執行個體具備存取 Amazon ECR 儲存庫內映像的許可。您可以將 AmazonEC2 ContainerRegistryReadOnly 受管政策附加到執行個體設定檔，以提供對帳戶中所有 Amazon ECR 儲存庫的唯讀存取權，或使用下列範本建立自訂政策授予單一儲存庫的存取權：

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Sid": "AllowEbAuth",
        "Effect": "Allow",
        "Action": [
          "ecr:GetAuthorizationToken"
        ],
        "Resource": [
          "*"
        ]
      },
```

```
      {
        "Sid": "AllowPull",
        "Effect": "Allow",
        "Resource": [
          "arn:aws:ecr:us-east-2:account-id:repository/repository-name"
        ],
        "Action": [
          "ecr:GetAuthorizationToken",
          "ecr:BatchCheckLayerAvailability",
          "ecr:GetDownloadUrlForLayer",
          "ecr:GetRepositoryPolicy",
          "ecr:DescribeRepositories",
          "ecr:ListImages",
          "ecr:BatchGetImage"
        ]
      }
    ]
  }
```

將上述原則內的 Amazon Resource Name (ARN) 替換為您儲存庫的 ARN。

在您的 Dockerrun.aws.json 檔案，以 URL 引用映像。若為 Docker 平台，URL 會由 Image 定義：

```
  "Image": {
      "Name": "account-id.dkr.ecr.us-east-2.amazonaws.com/repository-name:latest",
      "Update": "true"
    },
```

若為多容器 Docker 平台，請使用容器定義物件中的 image 金鑰。

```
"containerDefinitions": [
        {
        "name": "my-image",
        "image": "account-id.dkr.ecr.us-east-2.amazonaws.com/repository-name:latest",
```

使用私有儲存庫中的映像

欲使用線上登錄檔託管的私有儲存庫內的 Docker 影像，您必須提供身分驗證檔案，內含向登錄檔進行驗證所需的資訊。

透過 docker login 命令產生身分驗證檔案。若是 Docker Hub 上的儲存庫，請執行 docker login：

```
$ docker login
```

若是其他登錄檔，請納入登錄伺服器的 URL：

```
$ docker login registry-server-url
```

> **Note**
>
> 如果您的 Elastic Beanstalk 環境使用 Amazon Linux AMI 平台版本 (Amazon Linux 2 之前的版本)，請閱讀 the section called “Amazon Linux AMI (Amazon Linux 2 之前的版本) 上的 Docker 組態” 中的其他資訊。

將名為 `.dockercfg` 的身分驗證檔案副本上傳至安全的 Amazon S3 儲存貯體。Amazon S3 儲存貯體必須託管在與使用該儲存貯體的環境相同的 AWS 區域。Elastic Beanstalk 無法從其他區域託管的 Amazon S3 儲存貯體下載檔案。針對執行個體設定檔中的 IAM 角色，授予 `s3:GetObject` 操作的許可。如需詳細資訊，請參閱管理 Elastic Beanstalk 執行個體描述檔。

將 Amazon S3 儲存貯體資訊納入 `Authentication` 檔案內的 `authentication` (v1) 或 `Dockerrun.aws.json` (v2) 參數。

如需 Docker 環境的 `Dockerrun.aws.json` 格式的詳細資訊，請參閱 Docker 組態。針對多容器環境，請參閱 ECS 受管 Docker 組態。

如需身分驗證檔案的詳細資訊，請至 Docker 網站參閱 Store images on Docker Hub 和 docker login。

為 Docker 環境設定受管更新

使用受管平台更新，您可以設定環境，根據排程自動更新至平台的最新版本。

如果是 Docker 環境，您可能會要決定自動平台更新是否應跨 Docker 版本實施 — 當新的平台版本包含新的 Docker 版本時。當從執行 2.9.0 版本以上的 Docker 平台版本的環境更新時，Elastic Beanstalk 支援跨 Docker 版本的受管理平台更新。當新的平台版本包含新版本的 Docker 時，Elastic Beanstalk 便會增加小型更新版本的號碼。因此，要允許跨 Docker 版本的受管平台更新，啟用次要和修補程式版本更新的受管平台更新。要避免跨 Docker 版本的受管平台更新，僅啟用修補程式版本更新的受管平台更新。

例如，以下組態檔案可讓受管平台於每個星期二的 9:00 AM UTC 為次要與修補程試版本更新，進而允許跨 Docker 版本的受管更新：

Example .ebextension/. managed-platform-update 配置

```
option_settings:
  aws:elasticbeanstalk:managedactions:
    ManagedActionsEnabled: true
    PreferredStartTime: "Tue:09:00"
  aws:elasticbeanstalk:managedactions:platformupdate:
    UpdateLevel: minor
```

針對執行 2.9.0 或更早版本 Docker 平台的環境，若新的平台版本包含新的 Docker 版本，則 Elastic Beanstalk 永遠不會執行受管平台更新。

Docker 組態命名空間

您可以使用組態檔案來設定組態選項，並在部署期間執行其他的執行個體設定工作。組態選項可由 Elastic Beanstalk 服務或您使用的平台來定義，並且會組織成「命名空間」。

> **Note**
>
> 此資訊僅適用於未執行 Docker Compose 的 Docker 環境。此選項與執行 Docker Compose 的 Docker 環境有不同的行為。如需使用 Docker Compose 之 Proxy 服務的詳細資訊，請參閱 容器選項。

除了適用於所有 Elastic Beanstalk 環境的支援選項之外，Docker 也支援使用下列命名空間的選項：

- `aws:elasticbeanstalk:environment:proxy` – 選擇適用於您環境的代理伺服器。Docker 支援執行 Nginx 或沒有代理伺服器。

下列範例配置檔案會將 Docker 環境設定為在沒有代理伺服器的情況下運作。

Example .ebextensions/docker-settings.config

```
option_settings:
  aws:elasticbeanstalk:environment:proxy:
    ProxyServer: none
```

Amazon Linux AMI (Amazon Linux 2 之前的版本) 上的 Docker 組態

如果您的 Elastic Beanstalk Docker 環境使用 Amazon Linux AMI 平台版本 (之前的 Amazon Linux 2)，請閱讀本節中的其他資訊。

使用私有儲存庫的身分驗證檔案

如果您要使用私有儲存庫的映像，則此資訊與您相關。自 Docker 版本 1.7 起，docker login 命令會變更身分驗證檔案的名稱和檔案格式。Amazon Linux AMI Docker 平台版本 (Amazon Linux 2 之前的版本) 需要較舊的 `~/.dockercfg` 格式組態檔案。

若使用 Docker 版本 1.7 和更新版本，docker login 命令會於 `~/.docker/config.json` 建立身分驗證檔案，格式如下。

```
{
    "auths":{
      "server":{
        "auth":"key"
      }
    }
  }
```

若使用 Docker 版本 1.6.2 和更早版本，docker login 命令會於 `~/.dockercfg` 建立身分驗證檔案，格式如下。

```
{
    "server" :
    {
      "auth" : "auth_token",
      "email" : "email"
    }
  }
```

欲轉換 `config.json` 檔案，請移除外部 `auths` 金鑰、新增 `email` 金鑰並平展 JSON 文件，以符合舊的格式。

在 Amazon Linux 2 Docker 平台版本上，Elastic Beanstalk 會使用較新的身分驗證檔案名稱和格式。如果您使用的是 Amazon Linux 2 Docker 平台版本，則可以使用 docker login 命令建立的身分驗證檔案，而無需進行任何轉換。

設定其他儲存磁碟區

為了提高 Amazon Linux AMI 的效能，Elastic Beanstalk 為您 Docker 環境的 Amazon EC2 執行個體設定兩個 Amazon EBS 儲存磁碟區。除了所有 Elastic Beanstalk 環境佈建的根磁碟區，Docker 環境會佈建第二個名為 `xvdcz` 的 12GB 磁碟區供映像儲存使用。

若您需要更多儲存空間或增加 IOPS 供 Docker 影像使用，您可於 aws:autoscaling:launchconfiguration 命名空間使用 BlockDeviceMapping 組態選項，藉此自訂映像儲存磁碟區。

例如，下列組態檔案會將儲存磁碟區的大小提高為 100 GB，而佈建 IOPS 為 500：

Example .ebextensions/blockdevice-xvdcz.config

```
option_settings:
  aws:autoscaling:launchconfiguration:
    BlockDeviceMappings: /dev/xvdcz=:100::io1:500
```

若您使用 BlockDeviceMappings 選項來設定您應用程式的其他磁碟區，您應該納入 xvdcz 的映射，以確保該磁碟區已建立。下列範例設定兩個磁碟區，一個為使用預設設定的映像儲存磁碟區 xvdcz，另一個則是名為 sdh 的額外 24 GB 應用程式磁碟區：

Example .ebextensions/blockdevice-sdh.config

```
option_settings:
  aws:autoscaling:launchconfiguration:
    BlockDeviceMappings: /dev/xvdcz=:12:true:gp2,/dev/sdh=:24
```

> **Note**
>
> 當於此命名空間變更設定時，Elastic Beanstalk 會將您環境中的所有執行個體，替換為執行新組態的執行個體。如需詳細資訊，請參閱「組態變更」。

使用 Amazon ECS 平台分支

本主題介紹執行於 Amazon Linux 2 的 Amazon ECS 平台分支及其取代的執行於 AL1 的多容器 Docker 平台分支 (也稱為 ECS 受管)。除非另有說明，否則本主題中的所有資訊均適用於兩個平台分支。

> **Note**
>
> 2022 年 7 月 18 日，Elastic Beanstalk 已將所有以 Amazon Linux AMI (AL1) 為基礎的平台分支狀態設為已淘汰。

從 AL1 上的多容器 Docker 遷移

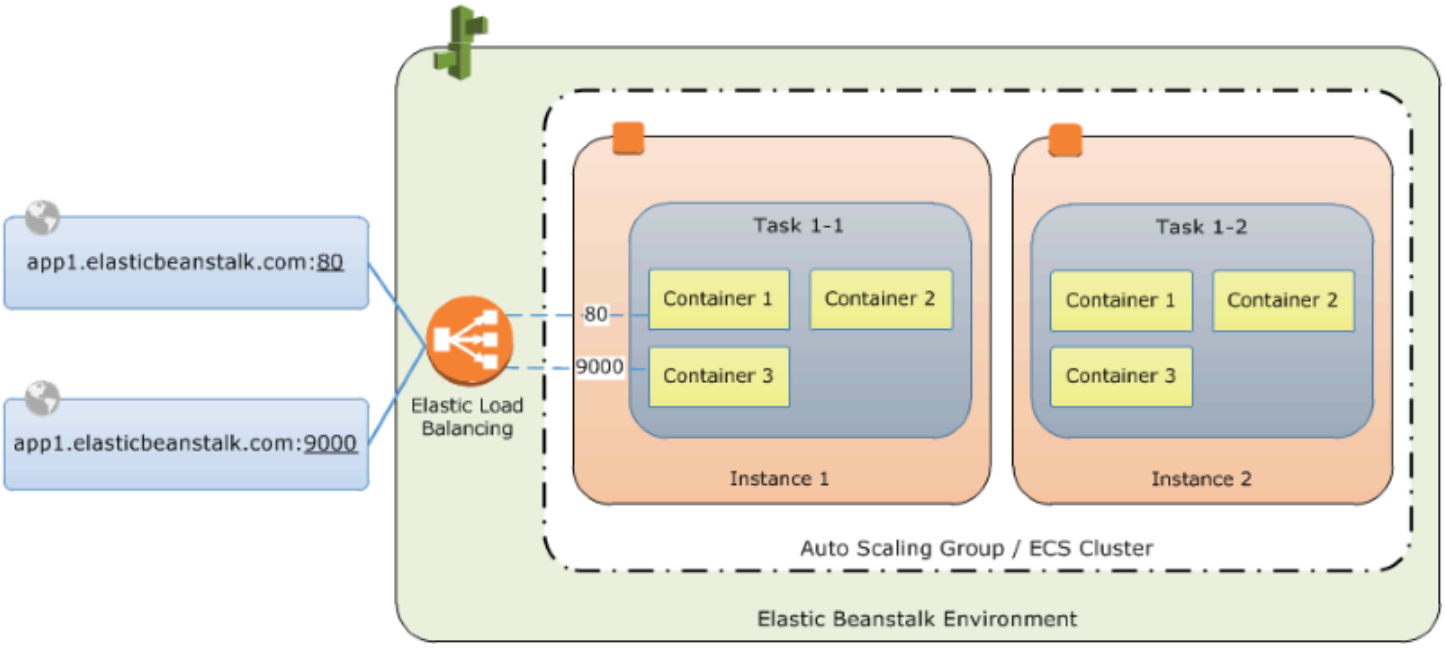
如果您目前正在使用執行於 AL1 的已淘汰多容器 Docker 平台分支，可以遷移至執行於 AL2023 上的最新 ECS 平台分支。最新的平台分支可支援此已停用平台分支的所有功能。無需變更原始程式碼。如需詳細資訊，請參閱將執行於 Amazon Linux 的多容器 Docker 遷移至執行於 Amazon Linux 2023 的 ECS。

主題

ECS 受管 Docker 平台

Elastic Beanstalk 使用 Amazon Elastic Container Service (Amazon ECS) 來協調 ECS 受管 Docker 環境的容器部署。Amazon ECS 提供了工具，來管理執行 Docker 容器的執行個體叢集。Elastic Beanstalk 會負責處理 Amazon ECS 的工作，包括建立叢集、定義工作與執行。環境中的每個執行個體，均須執行由 `Dockerrun.aws.json` v2 檔案定義的相同容器組合。為了充分發揮 Docker 的功用，Elastic Beanstalk 可讓您建立環境，而您的 Amazon EC2 執行個體可在此環境中並列執行多個 Docker 容器。

下圖顯示範例 Elastic Beanstalk 環境，此環境設定了在 Auto Scaling 群組中的每個 Amazon EC2 執行個體上，各執行 3 個 Docker 容器：



> **Note**
>
> Elastic Beanstalk 為其所有平台提供了可擴展性功能，您可以使用這些功能來自訂應用程式的部署和執行。對於執行於 Amazon Linux 2 的 ECS 平台分支，這些功能的執行個體部署工作流程實作與其他平台有所差異。如需詳細資訊，請參閱執行於 Amazon Linux 2 和更新版本的 ECS 的執行個體部署工作流程。

`Dockerrun.aws.json` file

容器執行個體 – 在 Elastic Beanstalk 環境中執行 ECS 受管 Docker 的 Amazon EC2 執行個體 – 需要名稱為 `Dockerrun.aws.json` 的組態檔案。此檔案為 Elastic Beanstalk 專用，可單獨使用，或是與原始碼套件中的原始碼與內容結合，用來在 Docker 平台上建立環境。

> **Note**
>
> 使用第 1 版 `Dockerrun.aws.json` 格式，將單一 Docker 容器啟動至執行於 Amazon Linux AMI 的 Elastic Beanstalk 環境 (Amazon Linux 2 之前的版本)。環境以執行於 64 位元 Amazon Linux 的 Docker 平台分支為基礎，後者將於 2022 年 7 月 18 日淘汰。若要進一步了解 `Dockerrun.aws.json` v1 格式，請參閱 Docker 平台組態 - 不含 Docker Compose。
> 第 2 版 `Dockerrun.aws.json` 格式新增了每個 Amazon EC2 執行個體執行多個容器的支援功能，但只能搭配 ECS 受管 Docker 平台使用。其格式與前一版有顯著差異。

如需關於更新格式的詳細資訊與範例檔案，請參閱 Dockerrun.aws.json v2。

Docker 影像

Elastic Beanstalk 的 ECS 受管 Docker 平台需要預先建置映像，並將映像儲存於公有或私有的線上映像儲存庫。

> **Note**
>
> Elastic Beanstalk 上的 ECS 受管 Docker 平台，並不支援在部署時使用 Dockerfile 來建置自訂映像。請在建立 Elastic Beanstalk 環境之前，先建置您的映像，並將這些映像部署到線上儲存庫中。

根據 Dockerrun.aws.json v2 中的名稱來指定映像。請注意這些慣例：

- Docker Hub 上官方儲存庫中的映像，使用的是單一名稱 (例如，ubuntu 或 mongo)。
- Docker Hub 上的其他儲存庫中的映像要求使用組織名稱 (例如，amazon/amazon-ecs-agent)。
- 其他線上儲存庫中的映像的名稱中，會再加上網域名稱 (例如，quay.io/assemblyline/ubuntu)。

若要設定 Elastic Beanstalk 驗證私有儲存庫，請在您的 Dockerrun.aws.json v2 檔案中加入 authentication 參數。

容器執行個體角色

Elastic Beanstalk 使用 Amazon ECS 最佳化 AMI 搭配在 Docker 容器中執行的 Amazon ECS 容器代理程式。此代理程式會與 Amazon ECS 進行通訊，來協調統籌容器的部署作業。為了與 Amazon ECS 進行通訊，每個 Amazon EC2 執行個體都必須在 IAM 中擁有對應的許可。當您在 Elastic Beanstalk 主控台中建立環境時，這些許可會連接至預設的執行個體設定檔：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ECSAccess",
      "Effect": "Allow",
      "Action": [
        "ecs:Poll",
        "ecs:StartTask",
```

```
        "ecs:StopTask",
        "ecs:DiscoverPollEndpoint",
        "ecs:StartTelemetrySession",
        "ecs:RegisterContainerInstance",
        "ecs:DeregisterContainerInstance",
        "ecs:DescribeContainerInstances",
        "ecs:Submit*"
      ],
      "Resource": "*"
    }
  ]
}
```

如果您建立自己的執行個體設定檔，可以連接 AWSElasticBeanstalkMulticontainerDocker 受管雲則，以確保許可隨時保持於最新狀態。如需在 IAM 中建立原則與角色的相關說明，請參閱 IAM 使用者指南中的建立 IAM 角色。

由 Elastic Beanstalk 建立的 Amazon ECS 資源

當您使用 ECS 受管 Docker 平台來建立環境時，Elastic Beanstalk 會在建置環境的同時，自動建立和設定數個 Amazon Elastic Container Service 資源，同時建置相關環境。執行此操作時，它會在每個 Amazon EC2 執行個體上建立必要的容器。

- Amazon ECS 叢集 – 會將 Amazon ECS 中的容器執行個體整編為叢集。搭配 Elastic Beanstalk 使用時，一律會針對每個 ECS 受管 Docker 環境建立一個叢集。
- Amazon ECS 工作定義 – Elastic Beanstalk 會使用您專案中的 Dockerrun.aws.json v2 來產生 Amazon ECS 工作的定義，此定義可用來設定環境中的容器執行個體。
- Amazon ECS 工作 – Elastic Beanstalk 會與 Amazon ECS 進行通訊，在環境中的每個執行個體上執行工作，來統籌容器的部署。在可擴展環境中，Elastic Beanstalk 會在執行個體每次新增到叢集時，起始新的工作。在極少數的情況中，您可能需要為容器和映像增加預留的空間。請參閱 設定 Docker 環境 一節來進一步了解。
- Amazon ECS 容器代理程式 – 在您環境執行個體上的 Docker 容器中執行的代理程式。代理程式會輪詢 Amazon ECS 服務和等待工作執行。
- Amazon ECS 資料磁碟區 – Elastic Beanstalk 會將磁碟區的定義 (除了您在 Dockerrun.aws.json v2 中所定義的磁碟區以外) 加入工作定義中，以便於收集日誌。

  Elastic Beanstalk 會在容器執行個體上建立日誌磁碟區 (每個容器一個)，路徑為 /var/log/containers/*containername*。這些磁碟區會命名為 awseb-logs-*containername*，並提供給容器來進行掛載。關於掛載這些磁碟區的方法，詳細資訊請參閱 容器定義格式。

使用多個 Elastic Load Balancing 接聽程式

您可以在 ECS 受管 Docker 環境中設定多個 Elastic Load Balancing 接聽程式，針對不透過預設 HTTP 通訊埠執行的代理程式或其他服務，支援傳入的資料流。

在您的原始碼套件中建立 `.ebextensions` 資料夾，然後新增副檔名為 `.config` 的檔案。下列範例顯示用來建立 Elastic Load Balancing 接聽程式的組態檔案，此接聽程式會監聽 8080 埠。

`.ebextensions/elb-listener.config`

```
option_settings:
  aws:elb:listener:8080:
    ListenerProtocol: HTTP
    InstanceProtocol: HTTP
    InstancePort: 8080
```

如果您的環境在您建立的自訂 Amazon Virtual Private Cloud (Amazon VPC) 中執行，則 Elastic Beanstalk 會處理其餘的工作。在預設 VPC 中，您需要設定執行個體的安全群組，來允許從負載平衡器傳入。新增第二個組態檔案，來加入安全群組的傳入規則：

`.ebextensions/elb-ingress.config`

```
Resources:
  port8080SecurityGroupIngress:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: {"Fn::GetAtt" : ["AWSEBSecurityGroup", "GroupId"]}
      IpProtocol: tcp
      ToPort: 8080
      FromPort: 8080
      SourceSecurityGroupName: { "Fn::GetAtt": ["AWSEBLoadBalancer",
 "SourceSecurityGroup.GroupName"] }
```

如需組態檔案格式的詳細資訊，請參閱 新增和自訂 Elastic Beanstalk 環境資源 和 選項設定。

除了在 Elastic Load Balancing 組態中新增接聽程式，以及在安全群組中開啟通訊埠之外，您還需要在 `Dockerrun.aws.json` v2 的 `containerDefinitions` 區段中，將主機執行個體上的通訊埠對應到 Docker 容器上的通訊埠。下列的摘錄顯示了範例：

```
"portMappings": [
  {
    "hostPort": 8080,
```

```
    "containerPort": 8080
  }
]
```

關於 Dockerrun.aws.json v2 檔案格式的詳細資訊，請參閱 Dockerrun.aws.json v2。

失敗的容器部署

如果 Amazon ECS 的工作失敗，則您 Elastic Beanstalk 環境中的一個或多個容器將不會開始執行。因為 Amazon ECS 工作失敗，所以 Elastic Beanstalk 不會還原多容器環境。如果容器無法在您的環境中啟動，請從 Elastic Beanstalk 主控台重新部署目前的版本或先前可正常運作的版本。

若要部署現有的版本

1. 在您環境的區域中開啟 Elastic Beanstalk 主控台。
2. 按一下應用程式名稱右方的 Actions (動作)，然後按一下 View application versions (檢視應用程式版本)。
3. 選擇應用程式的版本，然後按一下 Deploy (部署)。

ECS 受管 Docker 組態

Dockerrun.aws.json 為 Elastic Beanstalk 組態檔案，會說明如何部署於 Elastic Beanstalk 環境之 ECS 叢集中託管的 Docker 容器組。Elastic Beanstalk 平台會建立 ECS 任務定義，其中包含 ECS 容器定義。這些定義會在 Dockerrun.aws.json 組態檔案中說明。

Dockerrun.aws.json 檔案中的容器定義將說明要部署至 ECS 叢集中每個 Amazon EC2 執行個體的容器。在此情況下，Amazon EC2 執行個體也稱為主機容器執行個體，因為其託管 Docker 容器。組態檔案也會針對要掛載的 Docker 容器說明要在主機容器執行個體上建立的資料磁碟區。如需 Elastic Beanstalk 上 ECS 受管 Docker 環境中的元件詳細資訊和圖表，請參閱本章先前所述的 ECS 受管 Docker 平台 內容。

Dockerrun.aws.json 檔案可獨立使用，或與其他原始碼一同壓縮為單一封存檔。與 Dockerrun.aws.json 一同封存的原始碼會部署至 Amazon EC2 容器執行個體，並可於 /var/app/current/ 目錄存取。

主題

- Dockerrun.aws.json v2
- 磁碟區格式
- 容器定義格式

- 驗證格式 — 使用私有儲存庫的影像
- 範例 Dockerrun.aws.json v2

Dockerrun.aws.json v2

Dockerrun.aws.json 檔案包含下列章節：

AWSEBDockerrunVersion

將 ECS 受管 Docker 環境的版本編號指定為值 2。

磁碟區

自 Amazon EC2 容器執行個體的資料夾或已部署至 /var/app/current 的原始碼套件，建立磁碟區。使用 containerDefinitions 章節中的 mountPoints，將這些磁碟區掛載至 Docker 容器內的路徑。

containerDefinitions

容器定義陣列。

驗證 (選用)

內含私有儲存庫驗證資料的 .dockercfg 檔案於 Amazon S3 中的位置。

Dockerrun.aws.json 的 containerDefinitions 與磁碟區區段使用的格式，與 Amazon ECS 任務定義檔案的對應區段相同。如需任務定義格式的詳細資訊及任務定義參數的完整清單，請參閱《Amazon Elastic Container Service 開發人員指南》中的 Amazon ECS 任務定義。

磁碟區格式

磁碟區參數會從 Amazon EC2 容器執行個體中的資料夾，或從原始碼套件 (已部署至 /var/app/current) 建立磁碟區。

磁碟區的指定格式如下：

```
"volumes": [
    {
      "name": "volumename",
      "host": {
        "sourcePath": "/path/on/host/instance"
      }
    }
```

```
  ],
```

使用容器定義中的 `mountPoints`，將這些磁碟區掛載至 Docker 容器內的路徑。

Elastic Beanstalk 會設定日誌的其他磁碟區，每個容器均有一個磁碟區。這些應由 Docker 容器掛載，以將日誌寫入主機執行個體。

如需詳細資訊，請參閱下列容器定義格式章節中的 `mountPoints` 欄位。

容器定義格式

下列範例將說明 containerDefinitions 區段常用參數的子集。另外還有更多選用的參數。

Beanstalk 平台會建立 ECS 任務定義，其中包含 ECS 容器定義。Beanstalk 支援 ECS 容器定義的參數子集。如需詳細資訊，請參閱《Amazon Elastic Container Service 開發人員指南》中的容器定義。

`Dockerrun.aws.json` 檔案內含一個或多個容器定義物件的陣列，其中包含下列欄位：

name

容器的名稱。如需長度上限及允許的字元相關資訊，請參閱標準容器定義參數。

image

從中建置 Docker 容器之線上 Docker 儲存庫的 Docker 影像名稱。請注意這些慣例：

- Docker Hub 上官方儲存庫中的映像，使用的是單一名稱 (例如，`ubuntu` 或 `mongo`)。
- Docker Hub 上的其他儲存庫中的映像要求使用組織名稱 (例如，`amazon/amazon-ecs-agent`)。
- 其他線上儲存庫中的映像更進一步要求使用網域名稱 (例如，`quay.io/assemblyline/ubuntu`)。

environment

傳遞至容器的環境變數陣列。

例如，下列項目會定義名稱為 **`Container`** 且值為 **`PHP`** 的環境變數：

```
"environment": [
  {
    "name": "Container",
    "value": "PHP"
  }
],
```

essential

若容器故障則應停止任務，則此值為 True。非基本容器完成或當機，不會影響執行個體上的其他容器。

memory

在容器執行個體上為容器預留的記憶體容量。為容器定義內 `memory` 或 `memoryReservation` 兩者參數中的其中一項，指定非零的整數。

memoryReservation

為容器保留的記憶體軟性限制 (MiB)。為容器定義內 `memory` 或 `memoryReservation` 兩者參數中的其中一項，指定非零的整數。

mountPoints

欲自 Amazon EC2 容器執行個體掛載的磁碟區，以及 Docker 容器檔案系統掛載磁碟區的位置。當您掛載內含應用程式內容的磁碟區，您的容器可讀取原始碼套件內上傳的資料。當您掛載日誌磁碟區進行日誌資料寫入，Elastic Beanstalk 可自這些磁碟區收集日誌資料。

Elastic Beanstalk 會在容器執行個體上建立日誌磁碟區 (每個 Docker 容器一個)，路徑為 `/var/log/containers/`*`containername`*。這些磁碟區名為 `awseb-logs-`*`containername`*，且掛載位置應位於日誌寫入容器檔案結構的位置。

例如，下列掛載點會將容器內 nginx 日誌的位置，對應至 Elastic Beanstalk 為 `nginx-proxy` 容器產生的磁碟區。

```
{
  "sourceVolume": "awseb-logs-nginx-proxy",
  "containerPath": "/var/log/nginx"
}
```

portMappings

將容器上的網路連接埠對應至主機上的連接埠。

links

連結的容器清單。已連結的容器可互相探索並安全通訊。

volumesFrom

自不同容器掛載所有磁碟區。例如，自名為 `web` 的容器掛載磁碟區：

```
"volumesFrom": [
```

```
  {
    "sourceContainer": "web"
  }
],
```

驗證格式 — 使用私有儲存庫的影像

`authentication` 區段包含私有儲存庫的驗證資料。此為選用項目。

將內含身分驗證檔案的 Amazon S3 儲存貯體相關資訊，新增至 `authentication` 檔案的 `Dockerrun.aws.json` 參數。請確認 `authentication` 參數包含有效的 Amazon S3 儲存貯體和金鑰。Amazon S3 儲存貯體託管的區域，必須與使用其的環境處於相同的區域。Elastic Beanstalk 將不會從其他區域託管的 Amazon S3 儲存貯體下載檔案。

使用下列格式：

```
"authentication": {
    "bucket": "DOC-EXAMPLE-BUCKET",
    "key": "mydockercfg"
  },
```

如需產生及上傳驗證檔案的相關資訊，請參閱本章環境組態主題中的 使用私有儲存庫中的映像。

範例 Dockerrun.aws.json v2

下列程式碼片段範例說明具備兩個容器的執行個體的 `Dockerrun.aws.json` 檔案之語法。

```
{
  "AWSEBDockerrunVersion": 2,
  "volumes": [
    {
      "name": "php-app",
      "host": {
        "sourcePath": "/var/app/current/php-app"
      }
    },
    {
      "name": "nginx-proxy-conf",
      "host": {
        "sourcePath": "/var/app/current/proxy/conf.d"
      }
    }
```

```
  ],
  "containerDefinitions": [
    {
      "name": "php-app",
      "image": "php:fpm",
      "environment": [
        {
          "name": "Container",
          "value": "PHP"
        }
      ],
      "essential": true,
      "memory": 128,
      "mountPoints": [
        {
          "sourceVolume": "php-app",
          "containerPath": "/var/www/html",
          "readOnly": true
        }
      ]
    },
    {
      "name": "nginx-proxy",
      "image": "nginx",
      "essential": true,
      "memory": 128,
      "portMappings": [
        {
          "hostPort": 80,
          "containerPort": 80
        }
      ],
      "links": [
        "php-app"
      ],
      "mountPoints": [
        {
          "sourceVolume": "php-app",
          "containerPath": "/var/www/html",
          "readOnly": true
        },
        {
          "sourceVolume": "nginx-proxy-conf",
          "containerPath": "/etc/nginx/conf.d",
```

```
          "readOnly": true
        },
        {
          "sourceVolume": "awseb-logs-nginx-proxy",
          "containerPath": "/var/log/nginx"
        }
      ]
    }
  ]
}
```

使用 Elastic Beanstalk 主控台來建置 ECS 受管 Docker 環境

您可以使用 Elastic Beanstalk 主控台，在單一執行個體或可擴展的 Elastic Beanstalk 環境中，啟動多容器執行個體的叢集。本教學課程詳細說明使用兩個容器環境的容器組態及原始程式碼準備。

在 Elastic Beanstalk 環境中的每個 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體上，會同時並列執行容器、PHP 應用程式和 nginx 代理。在建立環境和確認應用程式已執行後，您會連線到容器執行個體，來查看這些項目是否能夠順利共同運作。

章節

定義 ECS 受管 Docker 容器

建立新 Docker 環境的第一步，就是為您的應用程式資料建立目錄。此資料夾可以置於本機機器的任意位置，並讓您選擇任意名稱。除了容器組態檔案之外，此資料夾也會包含您將上傳到 Elastic Beanstalk 的內容，並部署到您的環境。

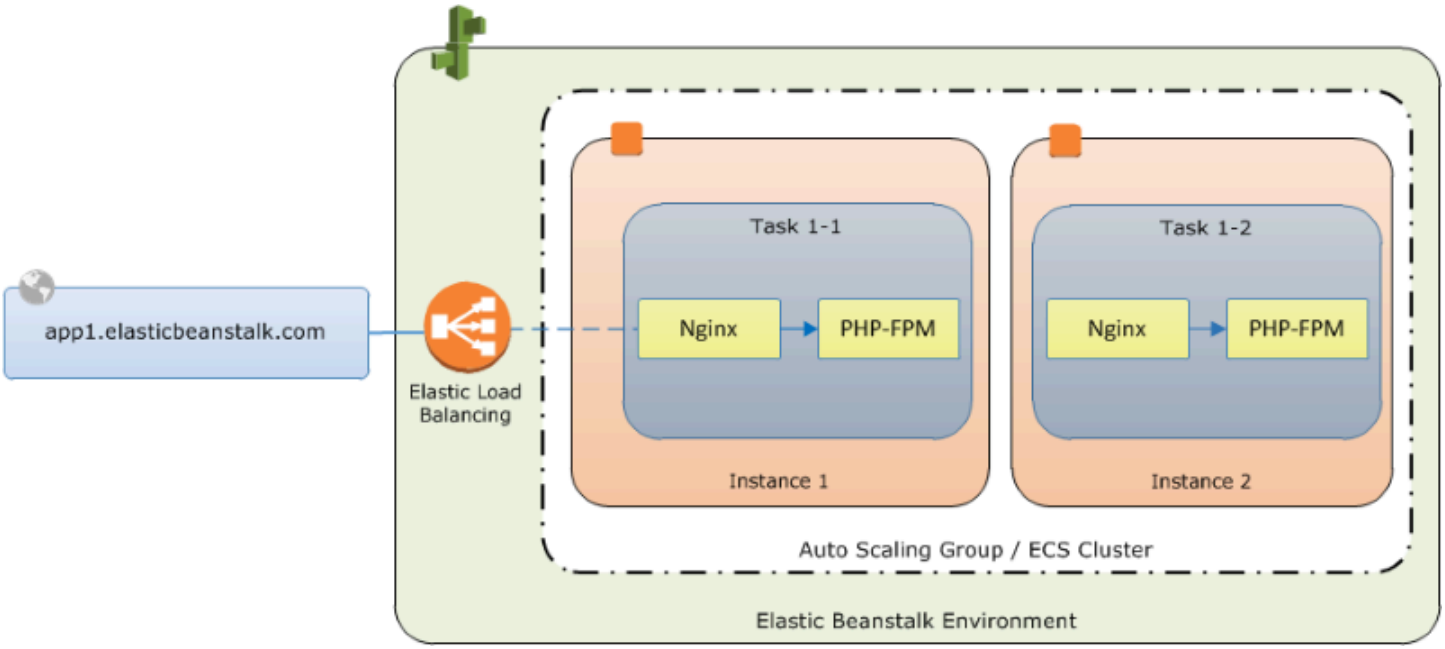
> **Note**
>
> 在 GitHub 上的 awslabs 儲存庫中 (https://github.com/awslabs/eb-docker-nginx-proxy)，提供了此教學課程中的所有程式碼。

Elastic Beanstalk 用來設定 Amazon EC2 執行個體容器的檔案，是名為 Dockerrun.aws.json 的 JSON 格式文字檔。在您應用程式的根目錄中，使用此名稱來建立文字檔案，並在此檔案中加入下列文字內容：

```
{
  "AWSEBDockerrunVersion": 2,
  "volumes": [
    {
      "name": "php-app",
      "host": {
        "sourcePath": "/var/app/current/php-app"
      }
    },
    {
      "name": "nginx-proxy-conf",
      "host": {
        "sourcePath": "/var/app/current/proxy/conf.d"
      }
    }
  ],
  "containerDefinitions": [
    {
      "name": "php-app",
      "image": "php:fpm",
      "essential": true,
      "memory": 128,
      "mountPoints": [
        {
          "sourceVolume": "php-app",
          "containerPath": "/var/www/html",
          "readOnly": true
        }
      ]
    },
    {
      "name": "nginx-proxy",
      "image": "nginx",
      "essential": true,
      "memory": 128,
      "portMappings": [
        {
          "hostPort": 80,
          "containerPort": 80
```

```
        }
      ],
      "links": [
        "php-app"
      ],
      "mountPoints": [
        {
          "sourceVolume": "php-app",
          "containerPath": "/var/www/html",
          "readOnly": true
        },
        {
          "sourceVolume": "nginx-proxy-conf",
          "containerPath": "/etc/nginx/conf.d",
          "readOnly": true
        },
        {
          "sourceVolume": "awseb-logs-nginx-proxy",
          "containerPath": "/var/log/nginx"
        }
      ]
    }
  ]
}
```

此範例組態定義了兩個容器 (前景有 nginx 代理的 PHP 網站)。這兩個容器將會在您 Elastic Beanstalk 環境的每個執行個體上，於 Docker 容器中同時並列執行，存取主機執行個體磁碟區的共用內容 (網站的內容)，該磁碟區也會在此檔案中定義。容器本身是從 Docker Hub 的官方儲存庫中所託管的映像建立的。所產生的環境看起來如下：



組態中所定義的磁碟區，會與您接下來所要建立的內容相符，並隨應用程式來源套件上傳。容器會掛載容器定義 `mountPoints` 區段中的磁碟區，以存取主機上的內容。

關於 `Dockerrun.aws.json` 的格式及其參數，詳細資訊請參閱 容器定義格式。

新增內容

接下來，您將會加入 PHP 網站的一些內容 (要顯示給訪客看的內容)，和 nginx 代理的組態檔案。

php-app/index.php

```
<h1>Hello World!!!</h1>
<h3>PHP Version <pre><?= phpversion()?></pre></h3>
```

php-app/static.html

```
<h1>Hello World!</h1>
<h3>This is a static HTML page.</h3>
```

proxy/conf.d/default.conf

```
server {
  listen 80;
  server_name localhost;
```

```
  root /var/www/html;

  index index.php;

  location ~ [^/]\.php(/|$) {
    fastcgi_split_path_info ^(.+?\.php)(/.*)$;
    if (!-f $document_root$fastcgi_script_name) {
      return 404;
    }

    include fastcgi_params;
    fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
    fastcgi_param PATH_INFO $fastcgi_path_info;
    fastcgi_param PATH_TRANSLATED $document_root$fastcgi_path_info;

    fastcgi_pass php-app:9000;
    fastcgi_index index.php;
  }
}
```

部署到 Elastic Beanstalk

您的應用程式資料夾現在包含下列檔案：

```
### Dockerrun.aws.json
### php-app
#   ### index.php
#   ### static.html
### proxy
    ### conf.d
        ### default.conf
```

您只需建立 Elastic Beanstalk 環境即可。建立前述檔案和資料夾的 `.zip` 封存檔 (不包括最上層專案資料夾)。若要在 Windows 檔案總管中建立封存檔，請選取專案資料夾的內容，在按一下滑鼠右鍵後選擇 Send To (傳送對象)，然後再按一下Compressed (zipped) Folder (壓縮的 (zipped) 資料夾)

> **Note**
>
> 如需所需的檔案架構的相關資訊，以及在其他環境中建立封存檔的說明，請參閱建立應用程式原始碼套件

接著，將原始碼套件上傳到 Elastic Beanstalk，然後建立您的環境。針對 Platform (平台)，請選取 Docker (Docker)。針對 Platform branch (平台分支)，請選取 ECS running on 64bit Amazon Linux 2 (執行於 64 位元 Amazon Linux 的 ECS)。

啟動環境 (主控台)

1. 透過此一預設連結來開啟 Elastic Beanstalk 主控台：console.aws.amazon.com/elasticbeanstalk/home#/newApplication?applicationName=tutorials&environmentType=LoadBalanced
2. 在 Platform (平台)，選取符合您應用程式所使用語言的平台和平台分支，或針對以容器為基礎的應用程式選取 Docker 平台。
3. 在 Application code (應用程式的程式碼)，選擇 Upload your code (上傳您的程式碼)。
4. 選擇 Local file (本機檔案)，選擇 Choose file (選擇檔案)，然後開啟原始碼套件。
5. 選擇 Review and launch (檢閱和啟動)。
6. 檢閱可用的設定，然後選擇 Create app (建立應用程式)。

Elastic Beanstalk 主控台會將您重新導向到新環境的管理儀表板。這個畫面會顯示環境的運作狀態，和 Elastic Beanstalk 服務所輸出的事件。當狀態為綠色 (Green) 時，請按一下環境名稱旁的 URL，來查看您的新網站。

連線到容器執行個體

接下來，您會連線到 Elastic Beanstalk 環境中的 Amazon EC2 執行個體，以查看正在運作的項目。

使用 EB CLI 是連線到您環境中執行個體的最簡單方法。若要使用 EB CLI，請安裝 EB CLI (如果您尚未安裝)。您也需要使用 Amazon EC2 SSH 金鑰對設定您的環境。使用主控台的安全組態頁面或 EB CLI eb init 命令進行此工作。若要連線到環境執行個體，請使用 EB CLI eb ssh 命令。

您已連線到承載您 Docker 容器的 Amazon EC2 執行個體，現在您可以查看運作的狀況。執行 `ls` 中的 `/var/app/current`：

```
[ec2-user@ip-10-0-0-117 ~]$ ls /var/app/current
Dockerrun.aws.json  php-app  proxy
```

此目錄包含原始碼套件中的檔案，您在建立環境時將此套件上傳到 Elastic Beanstalk。

```
[ec2-user@ip-10-0-0-117 ~]$ ls /var/log/containers
nginx-proxy    nginx-proxy-4ba868dbb7f3-stdouterr.log
```

```
php-app        php-app-dcc3b3c8522c-stdouterr.log       rotated
```

容器執行個體上的日誌，就是在此目錄中建立，Elastic Beanstalk 也會存取此目錄來收集日誌。Elastic Beanstalk 會在此目錄中，為每個容器建立磁碟區，您會將此磁碟區掛載到寫入日誌的容器位置。

您也可以利用 `docker ps` 來檢視 Docker，以查看執行中的容器。

```
[ec2-user@ip-10-0-0-117 ~]$ sudo docker ps
CONTAINER ID   IMAGE                              COMMAND                  CREATED
  STATUS                  PORTS                                  NAMES

4ba868dbb7f3   nginx                              "/docker-entrypoint.…"   4 minutes ago
   Up 4 minutes            0.0.0.0:80->80/tcp, :::80->80/tcp   ecs-awseb-Tutorials-env-
dc2aywfjwg-1-nginx-proxy-acca84ef87c4aca15400
dcc3b3c8522c   php:fpm                            "docker-php-entrypoi…"   4 minutes ago
   Up 4 minutes            9000/tcp                               ecs-awseb-Tutorials-env-
dc2aywfjwg-1-php-app-b8d38ae288b7b09e8101
d9367c0baad6   amazon/amazon-ecs-agent:latest   "/agent"                 5 minutes ago
  Up 5 minutes (healthy)                                          ecs-agent
```

這會顯示您所部署的兩個執行中的容器，以及統籌部署作業的 Amazon ECS 容器代理程式。

檢查 Amazon ECS 容器代理程式

Elastic Beanstalk 上 ECS 受管 Docker 環境中的 Amazon EC2 執行個體，會在 Docker 容器中執行代理程式程序。此代理程式會連線到 Amazon ECS 服務，以統籌容器的部署作業。這些部署作業會在 Amazon ECS 中以任務的形式執行，而任務是在任務定義檔案中設定的。Elastic Beanstalk 會根據您在原始碼套件中上傳的 `Dockerrun.aws.json`，來建立這些任務定義檔案。

發送 HTTP get 要求到 `http://localhost:51678/v1/metadata`，來查看容器代理程式的狀態：

```
[ec2-user@ip-10-0-0-117 ~]$ curl http://localhost:51678/v1/metadata
{
  "Cluster":"awseb-Tutorials-env-dc2aywfjwg",
  "ContainerInstanceArn":"arn:aws:ecs:us-west-2:123456789012:container-instance/awseb-
Tutorials-env-dc2aywfjwg/db7be5215cd74658aacfcb292a6b944f",
  "Version":"Amazon ECS Agent - v1.57.1 (089b7b64)"
}
```

此結構會顯示 Amazon ECS 叢集的名稱，和叢集執行個體 (您所連線的 Amazon EC2 執行個體) 的 ARN (Amazon Resource Name) 。

如需詳細資訊，請發出 HTTP get 請求到 http://localhost:51678/v1/tasks：

```
[ec2-user@ip-10-0-0-117 ~]$ curl http://localhost:51678/v1/tasks
{
   "Tasks":[
      {
         "Arn":"arn:aws:ecs:us-west-2:123456789012:task/awseb-Tutorials-env-dc2aywfjwg/
bbde7ebe1d4e4537ab1336340150a6d6",
         "DesiredStatus":"RUNNING",
         "KnownStatus":"RUNNING",
         "Family":"awseb-Tutorials-env-dc2aywfjwg",
         "Version":"1",
         "Containers":[
            {

 "DockerId":"dcc3b3c8522cb9510b7359689163814c0f1453b36b237204a3fd7a0b445d2ea6",
               "DockerName":"ecs-awseb-Tutorials-env-dc2aywfjwg-1-php-app-
b8d38ae288b7b09e8101",
               "Name":"php-app",
               "Volumes":[
                  {
                     "Source":"/var/app/current/php-app",
                     "Destination":"/var/www/html"
                  }
               ]
            },
            {

 "DockerId":"4ba868dbb7f3fb3328b8afeb2cb6cf03e3cb1cdd5b109e470f767d50b2c3e303",
               "DockerName":"ecs-awseb-Tutorials-env-dc2aywfjwg-1-nginx-proxy-
acca84ef87c4aca15400",
               "Name":"nginx-proxy",
               "Ports":[
                  {
                     "ContainerPort":80,
                     "Protocol":"tcp",
                     "HostPort":80
                  },
                  {
                     "ContainerPort":80,
                     "Protocol":"tcp",
                     "HostPort":80
                  }
               ],
```

```
                "Volumes":[
                   {
                      "Source":"/var/app/current/php-app",
                      "Destination":"/var/www/html"
                   },
                   {
                      "Source":"/var/log/containers/nginx-proxy",
                      "Destination":"/var/log/nginx"
                   },
                   {
                      "Source":"/var/app/current/proxy/conf.d",
                      "Destination":"/etc/nginx/conf.d"
                   }
                ]
             }
          ]
       }
    ]
}
```

此結構描述了一項執行的任務，就是部署這個教學課程的範例專案的兩個 docker 容器。隨即會顯示下列資訊：

- KnownStatus –RUNNING 狀態表示容器仍在運作中。
- 系列 – 任務定義的名稱，此任務定義是 Elastic Beanstalk 利用 Dockerrun.aws.json 建立的。
- 版本– 任務定義的版本。這會在任務定義檔案每次更新時遞增。
- 容器 – 關於執行個體上所執行容器的資訊。

Amazon ECS 服務本身提供了更多的資訊，您可以使用 來呼叫此服務AWS Command Line Interface 如需透過 Amazon ECS 來使用 AWS CLI 的說明，以及有關 Amazon ECS 的一般資訊，請參閱《Amazon ECS 使用者指南》。

將執行於 Amazon Linux 的多容器 Docker 遷移至執行於 Amazon Linux 2023 的 ECS

2022 年 7 月 18 日，Elastic Beanstalk 已將所有以 Amazon Linux AMI (AL1) 為基礎的平台分支狀態設為已淘汰。這包括平台分支在 64 位元 Amazon Linux 上執行的多容器 Docker。本主題可引導您將應用程式從此淘汰的平台分支遷移至執行於 64 位元 AL2023 的 ECS。這是最新且受支援的目標平台分支。

類似於舊版多容器 Docker AL1 分支，新版 ECS AL2023 平台分支使用 Amazon ECS 來協調將多個 Docker 容器部署至 Elastic Beanstalk 環境內的 Amazon ECS 叢集。新版 ECS AL2023 平台分支支援舊版多容器 Docker AL1 平台分支中的所有功能。此外支援同樣的 `Dockerrun.aws.json` v2 檔案。

章節

使用 Elastic Beanstalk 主控台遷移

若要使用 Elastic Beanstalk 主控台進行遷移，請將相同的原始程式碼部署到以執行於 AL2023 的 ECS 平台分支為基礎的新環境。無需變更原始程式碼。

遷移到執行於 Amazon Linux 2023 的 ECS 平台分支

1. 使用已部署到舊環境的應用程式原始碼套件建立應用程式原始碼。您可以使用相同的應用程式原始碼套件和相同的 `Dockerrun.aws.jsonv2` 檔案。
2. 使用執行於 Amazon Linux 2023 的 ECS 平台分支建立新環境。使用先前步驟所述應用程式的程式碼中的原始碼套件。如需詳細步驟，請參閱本章前面所述 ECS 受管 Docker 教學中的 部署到 Elastic Beanstalk。

使用 AWS CLI 遷移

您還可以選擇使用 AWS Command Line Interface (AWS CLI) 將您現有的多容器 Docker Amazon Linux Docker 環境遷移至新版 ECS AL2023 平台分支。在此情況下，您無需建立新的環境或重新部署您的原始程式碼。您只需執行 AWS CLI update-environment 命令。這會執行平台更新，以將現有的環境遷移至 ECS Amazon Linux 2023 平台分支。

使用以下語法將您的環境遷移至新的平台分支。

```
aws elasticbeanstalk update-environment \
--environment-name my-env \
--solution-stack-name "64bit Amazon Linux 2023 version running ECS" \
--region my-region
```

以下是在 us-east-1 區域將環境 beta-101 遷移至 3.0.0 版 ECS Amazon Linux 2023 平台分支的命令範例。

```
aws elasticbeanstalk update-environment \
```

```
--environment-name beta-101 \
--solution-stack-name "64bit Amazon Linux 2023 v4.0.0 running ECS" \
--region us-east-1
```

solution-stack-name 參數提供了平台分支及其版本。透過指定適當的解決方案堆疊名稱，使用最新的平台分支版本。每個平台分支的版本均包含在解決方案堆疊名稱內，如上例所示。如需 Docker 平台的最新解決方案堆疊清單，請參閱 AWS Elastic Beanstalk 平台指南中的支援的平台。

> **Note**
>
> list-available-solution-stacks 命令能提供您帳戶在 AWS 區域可以使用的平台版本清單。
>
> ```
> aws elasticbeanstalk list-available-solution-stacks --region us-east-1 --query
>  SolutionStacks
> ```

若要進一步了解 AWS CLI，請參閱 AWS Command Line Interface 使用者指南。如需有關 Elastic Beanstalk AWS CLI 命令的詳細資訊，請參閱 AWS CLIElastic Beanstalk 命令參考。

(舊式) 從執行於 Amazon Linux 的多容器 Docker 遷移至執行於 Amazon Linux 2 的 Docker 平台分支

在發佈執行於 64 位元 Amazon Linux 2 的 ECS 平台分支之前，Elastic Beanstalk 為使用以執行於 64 位元 Amazon Linux 2 的多容器 Docker 平台分支為基礎的環境的客戶提供了替代遷移路徑。本主題介紹該遷移路徑，並保留作為完成該遷移路徑的所有客戶的參考文件。

對於使用以執行於 64 位元 Amazon Linux 的多容器 Docker 平台分支為基礎的環境的客戶，我們現在建議遷移至執行於 64 位元 Amazon Linux 2 行的 ECS 平台分支。與替代遷移路徑不同，此方法繼續使用 Amazon ECS 來協調部署至 ECS 受管 Docker 環境的容器。這方面允許採取更直接的方法。無需變更原始程式碼，並且支援相同的 Dockerrun.aws.json v2。如需更多詳細資訊，請參閱 將執行於 Amazon Linux 的多容器 Docker 遷移至執行於 Amazon Linux 2023 的 ECS。

從執行於 Amazon Linux 的多容器 Docker 到 Docker Amazon Linux 2 平台分支的舊式版本遷移

您可以將在 Amazon Linux AMI 上的多容器 Docker 平台上執行的應用程式遷移至 Amazon Linux 2 Docker 平台。Amazon Linux AMI 上的多容器 Docker 平台需要您指定預先建置的應用程式映像，才能以容器身分執行。遷移之後，您將不再有此限制，因為 Amazon Linux 2 Docker 平台也允許 Elastic Beanstalk 在部署期間建置您的容器映像。您的應用程式將繼續在多容器環境中執行，並可從 Docker Compose 工具獲得額外的好處。

Docker Compose 是定義和執行多容器 Docker 應用程式的工具。若要進一步了解 Docker Compose 及如何安裝，請參閱 Docker 網站的 Docker Compose 的概觀和安裝 Docker Compose。

docker-compose.yml 檔案

Docker Compose 工具會使用 docker-compose.yml 檔案來設定您的應用程式服務。該檔案會取代您的應用程式專案目錄和應用程式原始碼套件中的 Dockerrun.aws.json v2 檔案。您可以手動建立 docker-compose.yml 檔案，並發現針對大部分參數值參考 Dockerrun.aws.json v2 檔案很有幫助。

以下是相同應用程式的 docker-compose.yml 檔案和對應 Dockerrun.aws.json v2 檔案的範例。如需有關 docker-compose.yml 檔案的詳細資訊，請參閱 Compose 檔案參考。如需有關 Dockerrun.aws.json v2 檔案的詳細資訊，請參閱Dockerrun.aws.json v2。

docker-compose.yml

```
version: '2.4'
services:
  php-app:
    image: "php:fpm"
    volumes:
      - "./php-app:/var/www/html:ro
"
      - "${EB_LOG_BASE_DIR}/php-app
:/var/log/sample-app"
    mem_limit: 128m
    environment:
      Container: PHP
  nginx-proxy:
    image: "nginx"
    ports:
      - "80:80"
    volumes:
      - "./php-app:/var/www/html:ro
"
      - "./proxy/conf.d:/etc/nginx/
conf.d:ro"
      - "${EB_LOG_BASE_DIR}/nginx-p
roxy:/var/log/nginx"
    mem_limit: 128m
```

Dockerrun.aws.json v2

```
{
  "AWSEBDockerrunVersion": 2,
  "volumes": [
    {
      "name": "php-app",
      "host": {
        "sourcePath": "/var/app/
current/php-app"
      }
    },
    {
      "name": "nginx-proxy-conf",
      "host": {
        "sourcePath": "/var/app/
current/proxy/conf.d"
      }
    }
  ],
  "containerDefinitions": [
    {
      "name": "php-app",
      "image": "php:fpm",
      "environment": [
        {
```

docker-compose.yml

```
    links:
      - php-app
```

Dockerrun.aws.json v2

```
          "name": "Container",
          "value": "PHP"
        }
      ],
      "essential": true,
      "memory": 128,
      "mountPoints": [
        {
          "sourceVolume": "php-app"
,
          "containerPath": "/var/www
/html",
          "readOnly": true
        }
      ]
    },
    {
      "name": "nginx-proxy",
      "image": "nginx",
      "essential": true,
      "memory": 128,
      "portMappings": [
        {
          "hostPort": 80,
          "containerPort": 80
        }
      ],
      "links": [
        "php-app"
      ],
      "mountPoints": [
        {
          "sourceVolume": "php-app"
,
          "containerPath": "/var/www
/html",
          "readOnly": true
        },
        {
          "sourceVolume": "nginx-pr
oxy-conf",
```

docker-compose.yml	Dockerrun.aws.json v2

```
          "containerPath": "/etc/ngi
nx/conf.d",
          "readOnly": true
        },
        {
          "sourceVolume": "awseb-lo
gs-nginx-proxy",
          "containerPath": "/var/log
/nginx"
        }
      ]
    }
  ]
}
```

其他遷移考量

Docker Amazon Linux 2 平台和多容器 Docker Amazon Linux AMI 平台實作環境屬性的方式不同。這兩個平台也有不同日誌目錄，Elastic Beanstalk 會為每個容器建立日誌目錄。從 Amazon Linux AMI 多容器 Docker 平台遷移之後，您必須注意這些新 Amazon Linux 2 Docker 平台環境的不同實作。

區域	Amazon Linux 2 上包含 Docker Compose 的 Docker 平台	Amazon Linux AMI 上的多容器 Docker 平台
環境屬性	為了讓您的容器能夠存取環境屬性，您必須在 `.env` 檔案中新增對檔案的 `docker-compose.yml` 參考。Elastic Beanstalk 會產生 `.env` 檔案，將每個屬性列為環境變數。如需詳細資訊，請參閱 在容器中參考環境變數。	Elastic Beanstalk 可以直接將環境屬性傳遞給容器。您在容器中執行的程式碼可以存取這些屬性做為環境變數，而無需任何其他組態。
日誌目錄	對於每個容器，Elastic Beanstalk 會建立名為 `/var/log/eb-docker/containers/` *`<service name>`* (或 `${EB_LOG_BASE_DIR}/<service name>`) 的日誌目錄。如需詳細資訊，	對於每個容器，Elastic Beanstalk 會建立名為 `/var/log/containers/` *`<containername>`* 的日誌目錄。如需詳細資訊，請參閱 容器定義格式 中的 `mountPoints` 欄位。

區域	Amazon Linux 2 上包含 Docker Compose 的 Docker 平台	Amazon Linux AMI 上的多容器 Docker 平台
	請參閱 Docker 容器自訂記錄 (Docker Compose)。	

遷移步驟

遷移到 Amazon Linux 2 Docker 平台

1. 根據現有的 `docker-compose.yml` 檔案，為您的應用程式建立 `Dockerrun.aws.json` v2 檔案。如需詳細資訊，請參閱上述章節 `docker-compose.yml` 檔案。
2. 在您的應用程式專案資料夾根目錄中，將 `Dockerrun.aws.json` v2 檔案取代為您剛剛建立的 `docker-compose.yml`。

   您的目錄結構應如下所示。

   ```
   ~/myApplication
   |-- docker-compose.yml
   |-- .ebextensions
   |-- php-app
   |-- proxy
   ```

3. 使用 eb init 命令來設定要部署至 Elastic Beanstalk 的本機目錄。

   ```
   ~/myApplication$ eb init -p docker application-name
   ```

4. 使用此 eb create 命令建立環境並部署 Docker 影像。

   ```
   ~/myApplication$ eb create environment-name
   ```

5. 如果您的應用程式是 Web 應用程式，環境啟動後，請使用 eb open 命令，以在 Web 瀏覽器中檢視該命令。

   ```
   ~/myApplication$ eb open environment-name
   ```

6. 您可以使用 eb status 命令來顯示新建立環境的狀態。

   ```
   ~/myApplication$ eb status environment-name
   ```

預先設定的 Docker 容器 (Amazon Linux AMI)

> **Note**
>
> 2022 年 7 月 18 日，Elastic Beanstalk 已將所有以 Amazon Linux AMI (AL1) 為基礎的平台分支狀態設為已淘汰。如需有關遷移至完全支援的目前 Amazon Linux 2023 平台分支的詳細資訊，請參閱 將您的 Elastic Beanstalk Linux 應用程式遷移到 Amazon Linux 2023 或 Amazon Linux 2。

不再支援在 Amazon Linux AMI (AL1) 上執行的預先設定的泊塢視窗 GlassFish 平台分支。若要將您的 GlassFish 應用程式遷移到受支援的 Amazon Linux 2023 平台，請部署 GlassFish 並將應用程式程式碼部署到 Amazon 2023 泊塢視窗映像。如需詳細資訊，請參閱下列主題，the section called “教程-碼 GlassFish 頭工人：通往 Amazon Linux 2023 的路徑”。

在 Amazon Linux AMI (Amazon Linux 2 之前的版本) 上開始使用預先設定的 Docker 容器

本章節說明如何於本機開發範例應用程式，然後使用預先設定的 Docker 容器將應用程式部署至 Elastic Beanstalk。

設定您的本機開發環境

對於這個演練，我們使用一個 GlassFish 示例應用程序。

設定您的環境。

1. 建立新的範例應用程式資料夾。

```
~$ mkdir eb-preconf-example
~$ cd eb-preconf-example
```

2. 下載範例應用程式程式碼到該新資料夾。

```
~$ wget https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/samples/docker-glassfish-v1.zip
~$ unzip docker-glassfish-v1.zip
~$ rm docker-glassfish-v1.zip
```

本機開發與測試

開發範例應 GlassFish 用程式

1. 將 Dockerfile 新增至應用程式的根資料夾。在檔案中，指定要用來執行本機預先設定 AWS Elastic Beanstalk Docker 容器的 Docker 基礎映像檔。您稍後會將應用程式部署到 Elastic Beanstalk 預先設定的 D GlassFish ocker 平台版本。選擇此平台版本使用的 Docker 基本映像。如需有關該平台版本最新 Docker 映像檔的詳細資訊，請參閱《AWS Elastic Beanstalk 平台指南》中支援AWS Elastic Beanstalk 的平台頁面的預先設定的 Docker 一節。

   Example 一 /E/碼頭文b-preconf-example件

   ```
   # For Glassfish 5.0 Java 8
   FROM amazon/aws-eb-glassfish:5.0-al-onbuild-2.11.1
   ```

   如需使用 Dockerfile 的詳細資訊，請參閱 Docker 組態。

2. 建置 Docker 影像。

   ```
   ~/eb-preconf-example$ docker build -t my-app-image .
   ```

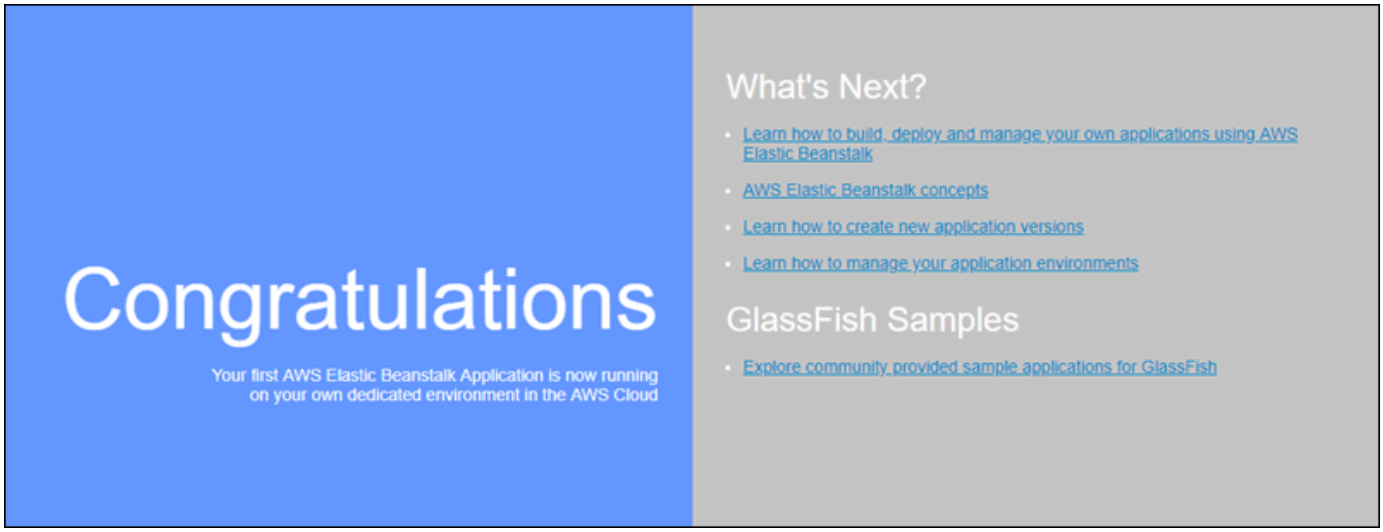
3. 自映像執行 Docker 容器。

   > **Note**
   >
   > 您必須納入 -p 旗標將容器上的連接埠 8080 對應至 localhost 連接埠 3000。Elastic Beanstalk Docker 容器永遠會公開容器上連接埠 8080 的應用程式。-it 旗標會將映像做為互動式程序來執行。容器結束時，--rm 旗標會清除容器檔案系統。您可以選擇是否加入 -d 旗標將映像做為協助程式來執行。

   ```
   $ docker run -it --rm -p 3000:8080 my-app-image
   ```

4. 欲檢視範例應用程式，請將下列 URL 輸入您的 Web 瀏覽器。

   ```
   http://localhost:3000
   ```



部署到 Elastic Beanstalk

測試應用程式後，即可準備將其部署至 Elastic Beanstalk。

將您的應用程式部署至 Elastic Beanstalk

1. 在應用程式的根資料夾中，將 `Dockerfile` 重新命名為 `Dockerfile.local`。欲讓 Elastic Beanstalk 使用 `Dockerfile`，此為必要步驟，因為該檔案內含正確說明，可供 Elastic Beanstalk 於 Elastic Beanstalk 環境的各個 Amazon EC2 執行個體建置自訂 Docker 影像。

   > **Note**
   >
   > 若您的 `Dockerfile` 中包含修改平台版本基礎 Docker 影像的指示，則不需要執行此步驟。若您的 `Dockerfile` 內僅有一行 `Dockerfile` 來指定應從中建立容器的基礎映像，則完全無須使用 `FROM`。在此情況下，`Dockerfile` 為多餘的。

2. 建立應用程式原始碼套件。

   ```
   ~/eb-preconf-example$ zip myapp.zip -r *
   ```

3. 使用這個預先配置的鏈接打開 Elastic Beanstalk 控制台：控制台。aws.amazon.com/彈性豆/家 # /新 applicationName = 教程和 environmentType = LoadBalanced
4. 針對 Platform (平台)，請在 Preconfigured Docker (預先設定的 Docker) 下，選擇 Glassfish。
5. 針對 Application code (應用程式程式碼)，選擇 Upload your code (上傳您的程式碼)，然後選擇 Upload (上傳)。

6. 選擇 Local file (本機檔案)，選擇 Browse (瀏覽)，然後開啟您剛才建立的應用程式原始碼套件。
7. 選擇上傳。
8. 選擇 Review and launch (檢閱和啟動)。
9. 檢閱可用的設定，然後選擇 Create app (建立應用程式)。
10. 在環境建立後，您可以查看部署的應用程式。選擇顯示於主控台儀表板頂端的環境 URL。

將 GlassFish 應用程式部署到碼頭平台：Amazon Linux 2023 的遷移路徑

本教學的目標是為使用預先設定的泊塢視窗 GlassFish 平台 (以 Amazon Linux AMI 為基礎) 的客戶提供 Amazon Linux 2023 的遷移路徑。您可以將 GlassFish 應用程式程式碼部署 GlassFish 到 Amazon 2023 泊塢視窗映像，將應用程式移轉到 Amazon Linux 2023。

本教學課程將逐步引導您使用 AWS Elastic Beanstalk Docker 平台，將以 Java EE 應用程式伺服器為基礎的 GlassFish 應用程式部署至 Elastic Beanstalk 環境。

我們示範了兩種建立 Docker 影像的方法：

- 簡單 — 提供您的 GlassFish 應用程式原始程式碼，讓 Elastic Beanstalk 建置並執行 Docker 映像檔，做為佈建環境的一部分。這很容易設定，但需要增加執行個體佈建時間。
- 進階 – 建置包含應用程式程式碼和相依性的自訂 Docker 影像，並提供它以在您的環境中使用 Elastic Beanstalk。此方法較複雜，並會減少環境中執行個體的佈建時間。

必要條件

此教學課程假設您具備基本的 Elastic Beanstalk 操作、Elastic Beanstalk 命令列 (EB CLI) 及 Docker 知識。若您尚不了解，請依照 開始使用 Elastic Beanstalk 中的說明來啟動您的第一個 Elastic Beanstalk 環境。此教學課程使用 EB CLI，但您也可使用 Elastic Beanstalk 主控台來建立環境並上傳應用程式。

若要按照此教學課程操作，您還需有下列 Docker 元件：

- 在本機上安裝的可用 Docker。如需詳細資訊，請參閱 Docker 文件網站上的取得 Docker。
- Docker Hub 的存取權限。您必須建立 Docker ID 才能存取 Docker Hub。如需詳細資訊，請參閱 Docker 文件網站上的共用應用程式。

若要進一步了解在 Elastic Beanstalk 平台上設定 Docker 環境，請參閱同一章中的 Docker 組態。

簡例：提供您的應用程式程式碼

這是部署 GlassFish 應用程式的簡單方法。您可以提供您的應用程式原始程式碼，以及本教學課程中所包含的 `Dockerfile`。Elastic Beanstalk 構建包含您的應用程序和軟件堆棧的 Docker 映像。GlassFish 然後 Elastic Beanstalk 會在您的環境執行個體上執行映像。

這種方法的一個問題是，每當它為您的環境建立一個執行個體時，都會在本機 Elastic Beanstalk 建立 Docker 影像。映像組建會增加執行個體佈建時間。這種影響並不局限於初始環境建立作業 — 它發生於向外擴展動作期間。

使 GlassFish 用範例應用程式啟動環境

1. 下載範例 `docker-glassfish-al2-v1.zip`，然後將 `.zip` 檔案展開至開發環境中的目錄。

```
~$ curl https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/samples/docker-
glassfish-al2-v1.zip --output docker-glassfish-al2-v1.zip
~$ mkdir glassfish-example
~$ cd glassfish-example
~/glassfish-example$ unzip ../docker-glassfish-al2-v1.zip
```

   您的目錄結構應如下所示。

```
~/glassfish-example
|-- Dockerfile
|-- Dockerrun.aws.json
|-- glassfish-start.sh
|-- index.jsp
|-- META-INF
|   |-- LICENSE.txt
|   |-- MANIFEST.MF
|   `-- NOTICE.txt
|-- robots.txt
`-- WEB-INF
    `-- web.xml
```

   下列檔案是在您的環境中建置和執行 Docker 容器的關鍵：

   - `Dockerfile` – 提供 Docker 用來建置具有應用程式和必要相依性的映像的指示。
   - `glassfish-start.sh` – 系統會執行 Docker 影像以啟動應用程式的 shell 指令碼。
   - `Dockerrun.aws.json`— 提供記錄金鑰，以將 GlassFish 應用程式伺服器記錄包含在記錄檔要求中。如果您對 GlassFish 日誌不感興趣，可以省略此文件。

2. 設定您的本機目錄以部署到 Elastic Beanstalk。

```
~/glassfish-example$ eb init -p docker glassfish-example
```

3. (選用) 使用 eb local run 於本機建置並執行您的容器。

```
~/glassfish-example$ eb local run --port 8080
```

> **Note**
>
> 欲進一步了解 eb local 命令，請參閱 the section called “eb local”。Windows 不支援該命令。或者，您可使用 docker build 和 docker run 命令來建置並執行您的容器。如需詳細資訊，請參閱 Docker 文件。

4. (選用) 在您的容器執行時，使用 eb local open 命令在 Web 瀏覽器檢視您的應用程式。或者，在 Web 瀏覽器中開啟 http://localhost:8080/。

```
~/glassfish-example$ eb local open
```

5. 使用 eb create 來建立環境並部署您的應用程式。

```
~/glassfish-example$ eb create glassfish-example-env
```

6. 環境啟動後，請使用 eb open 命令在網頁瀏覽器中檢視該命令。

```
~/glassfish-example$ eb open
```

當您完成使用該範例時，終止環境並刪除相關資源。

```
~/glassfish-example$ eb terminate --all
```

進階範例：提供預先建置的 Docker 影像

這是部署應用程式的更進階方 GlassFish 式。基於第一個示例，您創建包含應用程序代碼和 GlassFish 軟件堆棧的 Docker 映像，並將其推送到 Docker Hub。完成此一次性步驟之後，您可以根據自訂映像啟動 Elastic Beanstalk 環境。

當您啟動環境並提供 Docker 影像時，環境中的執行個體會直接下載並使用此映像，而不需要建置 Docker 影像。因此，執行個體佈建時間會減少。

> **備註**
>
> - 下列步驟會建立可公開取得的 Docker 影像。
> - 您會使用本機上所安裝 Docker 的 Docker 命令，以及您的 Docker Hub 憑證。如需詳細資訊，請參閱本主題中前述的先決條件一節。

使用預先建置的 GlassFish 應用程式 Docker 映像來啟動環境

1. 如上一個簡例所示，下載並展開簡例docker-glassfish-al2-v1.zip。如果您已經完成該範例，則可以使用您已經擁有的目錄。
2. 建立一個 Docker 影像並將其推送到 Docker Hub。在 *docker-id* 輸入您的 Docker ID，登入 Docker Hub。

   ```
   ~/glassfish-example$ docker build -t docker-id/beanstalk-glassfish-example:latest .
   ~/glassfish-example$ docker push docker-id/beanstalk-glassfish-example:latest
   ```

   > **Note**
   >
   > 推送映像前，可能需要執行 docker login。如果您執行不含參數的命令，系統會提示您輸入 Docker Hub 憑證。

3. 建立其他目錄。

   ```
   ~$ mkdir glassfish-prebuilt
   ~$ cd glassfish-prebuilt
   ```

4. 將下列範例複製到名為 Dockerrun.aws.json 的檔案中。

   Example **~/glassfish-prebuilt/Dockerrun.aws.json**

   ```
   {
     "AWSEBDockerrunVersion": "1",
     "Image": {
       "Name": "docker-username/beanstalk-glassfish-example"
     },
     "Ports": [
       {
         "ContainerPort": 8080,
   ```

```
      "HostPort": 8080
    }
  ],
  "Logging": "/usr/local/glassfish5/glassfish/domains/domain1/logs"
}
```

5. 設定您的本機目錄以部署到 Elastic Beanstalk。

```
~/glassfish-prebuilt$ eb init -p docker glassfish-prebuilt$
```

6. (選擇性) 使用 eb local run 命令在本機執行容器。

```
~/glassfish-prebuilt$ eb local run --port 8080
```

7. (選用) 在您的容器執行時，使用 eb local open 命令在 Web 瀏覽器檢視您的應用程式。或者，在 Web 瀏覽器中開啟 http://localhost:8080/。

```
~/glassfish-prebuilt$ eb local open
```

8. 使用此 eb create 命令建立環境並部署 Docker 影像。

```
~/glassfish-prebuilt$ eb create glassfish-prebuilt-env
```

9. 環境啟動後，請使用 eb open 命令在網頁瀏覽器中檢視該命令。

```
~/glassfish-prebuilt$ eb open
```

當您完成使用該範例時，終止環境並刪除相關資源。

```
~/glassfish-prebuilt$ eb terminate --all
```

在 Elastic Beanstalk 上建立和部署 Go 應用程式

AWS Elastic Beanstalk Go 可讓您使用亞馬遜網路服務輕鬆部署、管理和擴展 Go Web 應用程式。所有使用 Go 來開發或託管 Web 應用程式的人，都能使用 Elastic Beanstalk for Go。本章提供了將 Web 應用程序部署到 Elastic Beanstalk 的 step-by-step 說明。

在部署您的 Elastic Beanstalk 應用程式之後，您可以繼續使用 EB CLI 來管理您的應用程式和環境，也可以使用 Elastic Beanstalk 主控台、 AWS CLI或 API。

主題

- QuickStart：將 Go 應用程式部署到 Elastic Beanstalk
- 設定您的 Go 開發環境
- 使用 Elastic Beanstalk Go 平台

QuickStart：將 Go 應用程式部署到 Elastic Beanstalk

本 QuickStart 教學課程將引導您完成建立 Go 應用程式並將其部署至 AWS Elastic Beanstalk 環境的程序。

> **Note**
>
> 本 QuickStart 自學課程用於示範目的。請勿將本教學課程中建立的應用程式用於生產流量。

章節

- 您的 AWS 帳戶
- 必要條件
- 步驟 1：創建 Go 應用程序
- 步驟 2：使用 EB CLI 部署 Go 應用程式
- 第 3 步：在 Elastic Beanstalk 上運行應用程序
- 步驟 4：清理
- AWS 您應用程式的資源
- 後續步驟
- 使用 Elastic Beanstalk 控制台進行部署

您的 AWS 帳戶

如果您還不是 AWS 客戶，則需要創建一個 AWS 帳戶。註冊使您可以訪問 Elastic Beanstalk 和您需要的其他 AWS 服務。

如果您已經有 AWS 帳戶，則可以轉到必要條件。

創建一個 AWS 帳戶

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」。

必要條件

為了遵循本指南的程序，您需要命令列終端機或 shell 來執行命令。命令清單前會出現提示字元 ($) 及目前的目錄名稱 (如有)。

```
~/eb-project$ this is a command
this is output
```

在 Linux 和 macOS 上，您可以使用偏好的 Shell 和套件軟體管理工具。在視窗上，你可以安裝視窗子系統為 Linux 得到一個視窗集成的 Ubuntu 和 Bash 的版本。

EB CLI

本教學使用 Elastic Beanstalk 命令列界面 (EB CLI)。關於安裝和設定 EB CLI 的詳細資訊，請參閱安裝 EB CLI 和設定 EB CLI。

步驟 1：創建 Go 應用程序

建立專案目錄。

```
~$ mkdir eb-go
~$ cd eb-go
```

接著，請建立您將使用 Elastic Beanstalk 進行部署的應用程式。我們將建立一個「Hello World」RESTful Web 服務。

此範例會列印自訂問候語，該問候語依用於存取該服務的路徑而異。

於此目錄建立名為 `application.go` 的文字檔案，內含下列內容。

Example **~/eb-go/application.go**

```
package main

import (
 "fmt"
 "net/http"
)

func handler(w http.ResponseWriter, r *http.Request) {
 if r.URL.Path == "/" {
  fmt.Fprintf(w, "Hello World! Append a name to the URL to say hello. For example, use
 %s/Mary to say hello to Mary.", r.Host)
 } else {
  fmt.Fprintf(w, "Hello, %s!", r.URL.Path[1:])
 }
}

func main() {
 http.HandleFunc("/", handler)
 http.ListenAndServe(":5000", nil)
}
```

步驟 2：使用 EB CLI 部署 Go 應用程式

接著，您建立應用程式環境，並透過 Elastic Beanstalk 部署已設定的應用程式。

欲建立環境並部署您的 Go 應用程式

1. 透過 eb init 命令初始化您的 EB CLI 儲存庫。

   ```
   ~/eb-go$ eb init -p go go-tutorial --region us-east-2
   Application go-tutorial has been created.
   ```

   此命令創建一個名為的應用程序，go-tutorial並配置您的本地存儲庫以創建具有最新 Go Platform 版本的環境。

2. (選用) 再次執行 eb init 來設定預設金鑰對，藉此使用 SSH 連接至執行您應用程式的 EC2 執行個體：

   ```
   ~/eb-go$ eb init
   Do you want to set up SSH for your instances?
   (y/n): y
   Select a keypair.
   1) my-keypair
   2) [ Create new KeyPair ]
   ```

   若您已有金鑰對，請選擇一對，或依照提示建立金鑰對。若未出現提示，或稍後需要變更設定，請執行 eb init -i。

3. 使用 eb create 建立環境並於其中部署您的應用程式。Elastic Beanstalk 會自動為您的應用程序構建一個 zip 文件，並在端口 5000 上啟動它。

   ```
   ~/eb-go$ eb create go-env
   ```

   Elastic Beanstalk 大約需要五分鐘的時間來創建您的環境。

第 3 步：在 Elastic Beanstalk 上運行應用程序

當創建環境的過程完成後，打開您的網站eb open。

```
~/eb-go$ eb open
```

恭喜您！您已經使用 Elastic Beanstalk 部署了 Go 應用程序！這會開啟瀏覽器視窗，並使用為應用程式建立的網域名稱。

步驟 4：清理

您可以在完成應用程式的工作後終止環境。Elastic Beanstalk 會終止與您環境相關的所有 AWS 資源。

若要使用 EB CLI 終止 Elastic Beanstalk 環境，請執行下列命令。

```
~/eb-go$ eb terminate
```

AWS 您應用程式的資源

您剛剛建立了單一執行個體應用程式。它可作為單一 EC2 執行個體的簡單範例應用程式使用，因此不需要負載平衡或 auto 擴展。對於單個實例應用程序，Elastic Beanstalk 創建以下 AWS 資源：

- EC2 執行個體 – 設定在您所選平台上執行 Web 應用程式的 Amazon EC2 虛擬機器。

  每個平台會執行不同一套軟體、設定檔和指令碼，來支援特定的語言版本、架構、Web 容器或其組合。大多數的平台會使用 Apache 或 nginx 做為反向代理，處理您 Web 應用程式前端的網路流量、向它轉送請求、提供靜態資產，並產生存取和錯誤日誌。
- 執行個體安全群組 – 設定允許從連接埠 80 傳入流量的 Amazon EC2 安全群組。此資源可讓負載平衡器傳來的 HTTP 傳輸資料，到達執行您 Web 應用程式的 EC2 執行個體。在預設情況下，不允許傳輸資料從其他通訊埠傳送。
- Amazon S3 儲存貯體 - 儲存位置，用來儲存當您使用 Elastic Beanstalk 時所建立的原始程式碼、日誌和其他成品。
- Amazon CloudWatch 警示 — 兩個 CloudWatch 警示，用於監控環境中執行個體的負載，並在負載過高或過低時觸發。當警示觸發時，您的 Auto Scaling 群組會擴展或縮減以進行回應。
- AWS CloudFormation 堆疊 — Elastic Beanstalk 用 AWS CloudFormation 來啟動環境中的資源並傳播組態變更。資源定義於範本中，您可在 AWS CloudFormation 主控台中檢視此範本。
- 網域名稱 – 會路由到您 Web 應用程式的網域名稱，其格式為 *subdomain*.*region*.elasticbeanstalk.com。

Elastic Beanstalk 會管理所有這些資源。當您終止環境時，Elastic Beanstalk 會終止其中的所有資源。

後續步驟

在您擁有執行應用程式的環境後，可以隨時部署應用程式的新版本或不同的應用程式。部署新的應用程式版本非常快速，因無須佈建或重新啟動 EC2 執行個體。您也可以使用 Elastic Beanstalk 控制台探索您的新環境。如需詳細步驟，請參閱本指南的「入門」一章中的「探索您的環境」。

在您部署一個或以上的範例應用程式，並準備好開始在本機開發和執行 Go 應用程式時，請參閱設定您的 Go 開發環境。

使用 Elastic Beanstalk 控制台進行部署

您也可以使用 Elastic Beanstalk 控制台來啟動示例應用程序。如需詳細步驟，請參閱本指南的「入門」一章中的「建立範例應用程式」。

設定您的 Go 開發環境

設定 Go 開發環境以在本機測試您的應用程式，然後才部署到 AWS Elastic Beanstalk。本主題描述開發環境的設定步驟，並提供實用工具安裝頁面的連結。

如需了解適用所有語言的常見設定步驟和工具，請參閱設定您的開發機器搭配 Elastic Beanstalk 使用。

安裝 Go

若要在本機執行 Go，請安裝 Go。如果您不需要特定版本，請取得 Elastic Beanstalk 支援的最新版本。如需支援版本的清單，請參閱 AWS Elastic Beanstalk 平台文件中的 Go。

在 https://golang.org/doc/install 下載 Go。

安裝適用於 Go 的 AWS 開發套件

如果您需要在應用程式內管理 AWS 資源，請使用下列命令以安裝適用於 Go 的 AWS 開發套件。

```
$ go get github.com/aws/aws-sdk-go
```

如需詳細資訊，請參閱適用於 Go 的 AWS 開發套件。

使用 Elastic Beanstalk Go 平台

您可以使用 AWS Elastic Beanstalk 來執行、建置並設定以 Go 為基礎的應用程式。以簡單的 Go 應用程式而言，有兩種方式可部署應用程式：

- 請於名為 `application.go` 的根目錄提供具備原始碼檔案的原始碼套件，其中包含您的應用程式的主要套件。Elastic Beanstalk 會使用下列命令來建置二進位檔案：

  ```
  go build -o bin/application application.go
  ```

  建置應用程式之後，Elastic Beanstalk 會於連接埠 5000 將其啟動。

- 提供具備名為 `application` 的二進位檔案之原始碼套件。此二進位檔案可能位於原始碼套件的根目錄，或原始碼套件的 `bin/` 目錄。若您將 `application` 二進位檔案同時置於這兩個位置，Elastic Beanstalk 會使用 `bin/` 目錄中的檔案。

  Elastic Beanstalk 於連接埠 5000 啟動此應用程式。

在這兩種情況下，使用 Go 1.11 或更新版本，您還可以在名為 `go.mod` 的文件中提供模組要求。如需詳細資訊，請參閱 Go 部落格中的遷移至 Go 模組。

以更複雜的 Go 應用程式而言，有兩種方式可部署應用程式：

- 提供原始碼套件，其中包含您的應用程式原始碼檔案，以及 Buildfile 和 Procfile。Buildfile 內含建置應用程式的命令，而 Procfile 則包含執行應用程式的說明。
- 提供原始碼套件，其中包含您的應用程式二進位檔案，以及 Procfile。Procfile 包含執行應用程式的說明。

Go 平台包含了代理伺服器，此等伺服器可以提供靜態資產並將傳輸資料轉傳至您的應用程式。您可以針對進階的應用情境，來擴展或覆寫預設的代理組態。

如需各種擴充 Elastic Beanstalk Linux 類型平台方式的詳細資訊，請參閱 the section called “擴充 Linux 平台”。

設定您的 Go 環境

Go 平台設定可讓您微調 Amazon EC2 執行個體的行為。您可以使用 Elastic Beanstalk 主控台編輯 Elastic Beanstalk 環境的 Amazon EC2 執行個體組態。

使用 Elastic Beanstalk 主控台來啟用至 Amazon S3 的日誌輪換，和設定您的應用程式可以從環境讀取的變數。

在 Elastic Beanstalk 主控台中設定 Go 環境

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇組態。
4. 在更新、監控和日誌記錄組態類別中，選擇編輯。

日誌選項

Log Options (日誌選項) 區段有兩個設定：

- 執行個體設定檔 – 指定有權存取與您應用程式相關的 Amazon S3 儲存貯體的執行個體設定檔。
- Enable log file rotation to Amazon S3 (啟用 Amazon S3 的日誌檔案輪換) – 指定是否將應用程式 Amazon EC2 執行個體的日誌檔案複製到與應用程式關聯的 Amazon S3 儲存貯體。

靜態檔案

為改善效能，您可以使用 Static files (靜態檔案) 區段來設定代理伺服器，以為來自 Web 應用程式一組目錄中的靜態檔案 (例如 HTML 或影像) 提供服務。對於每個目錄，您可以設定目錄映射的虛擬路徑。代理伺服器收到位於指定路徑下的檔案請求時，會直接提供檔案而非將請求路由至您的應用程式。

如需使用組態檔案或 Elastic Beanstalk 主控台設定靜態檔案的詳細資訊，請參閱the section called “靜態檔案”。

環境屬性

Environment Properties (環境屬性) 的部分可讓您針對執行您應用程式的 Amazon EC2 執行個體，來指定其上的環境資訊設定。環境屬性會以金鑰值對的形式傳到應用程式。

在 Elastic Beanstalk 內所執行的 Go 環境中，可使用 `os.Getenv` 函數來存取環境變數。例如，您可使用下列程式碼，來將名為 `API_ENDPOINT` 的屬性讀取到變數：

```
endpoint := os.Getenv("API_ENDPOINT")
```

如需詳細資訊，請參閱環境屬性與其他軟體設定。

Go 組態命名空間

您可以使用組態檔案來設定組態選項，並在部署期間執行其他的執行個體設定工作。組態選項可透過 Elastic Beanstalk 服務或您使用的平台來定義，並且會組織成命名空間。

Go 平台不會定義任何平台特定的命名空間。您可以使用 `aws:elasticbeanstalk:environment:proxy:staticfiles` 命名空間設定代理提供靜態檔案。如需詳細資訊和範例，請參閱the section called “靜態檔案”。

Elastic Beanstalk 可提供許多組態選項讓您自訂環境。除了組態檔案，您也可以使用主控台、已儲存組態、EB CLI 或 AWS CLI 來設定組態選項。如需詳細資訊，請參閱組態選項。

Amazon Linux AMI (之前的 Amazon Linux 2) Go 平台

如果您的 Elastic Beanstalk Go 環境使用 Amazon Linux AMI 平台版本 (Amazon Linux 2 前身)，請閱讀本節中的其他資訊。

> **備註**
>
> - 本主題中的資訊僅適用於以 Amazon Linux AMI (AL1) 為基礎的平台分支。AL2023/AL2 平台分支與舊版 Amazon Linux AMI (AL1) 平台版本不相容，需要不同的組態設定。
> - 2022 年 7 月 18 日，Elastic Beanstalk 已將所有以 Amazon Linux AMI (AL1) 為基礎的平台分支狀態設為已淘汰。如需有關遷移至完全支援的目前 Amazon Linux 2023 平台分支的詳細資訊，請參閱 將您的 Elastic Beanstalk Linux 應用程式遷移到 Amazon Linux 2023 或 Amazon Linux 2。

Go 組態命名空間 — Amazon Linux AMI (AL1)

您可以使用組態檔案來設定組態選項，並在部署期間執行其他的執行個體設定工作。組態選項可透過 Elastic Beanstalk 服務或您使用的平台來定義，並且會組織成命名空間。

> **Note**
>
> 本主題中的資訊僅適用於以 Amazon Linux AMI (AL1) 為基礎的平台分支。AL2023/AL2 平台分支與舊版 Amazon Linux AMI (AL1) 平台版本不相容，需要不同的組態設定。

除了所有平台皆支援的命名空間外，Amazon Linux AMI Go 平台亦支援一個平台特定的組態命名空間。`aws:elasticbeanstalk:container:golang:staticfiles` 命名空間可讓您定義選項，將您 Web 應用程式的路徑對應至應用程式原始碼套件中內含靜態內容的資料夾。

例如，此組態檔案會通知代理伺服器提供路徑 `/images` 下 `staticimages` 資料夾內的檔案：

Example .ebextensions/go-settings.config

```
option_settings:
  aws:elasticbeanstalk:container:golang:staticfiles:
    /html: statichtml
```

```
    /images: staticimages
```

Elastic Beanstalk 可提供許多組態選項讓您自訂環境。除了組態檔案，您也可以使用主控台、已儲存組態、EB CLI 或 AWS CLI 來設定組態選項。如需詳細資訊，請參閱組態選項。

使用 Procfile 設定應用程式程序

欲指定啟動 Go 應用程式的自訂命令，請於原始碼套件的根目錄，納入名為 `Procfile` 的檔案。

如需有關撰寫和使用 `Procfile` 的詳細資料，請展開the section called “擴充 Linux 平台”中的 Buildfile 和 Procfile 區段。

Example Procfile

```
web: bin/server
queue_process: bin/queue_processor
foo: bin/fooapp
```

您必須呼叫主應用程式 `web`，並在 `Procfile` 中將其列為第一個命令。Elastic Beanstalk 會將主要 `web` 應用程式公開於環境的根 URL，例如，`http://my-go-env.elasticbeanstalk.com`。

Elastic Beanstalk 亦會執行名稱沒有 `web_` 前綴字的應用程式，但這些應用程式無法自您的執行個體外取用。

Elastic Beanstalk 預期 `Procfile` 執行的程序會持續執行。Elastic Beanstalk 會監控這些應用程式，並重新啟動終止的任何程序。針對短期執行的程序，請使用 Buildfile 命令。

在 Amazon Linux AMI (Amazon Linux 2 之前) 上使用 Procfile

如果您的 Elastic Beanstalk Go 環境使用 Amazon Linux AMI 平台版本 (Amazon Linux 2 前身)，請閱讀本節中的其他資訊。

> **備註**
>
> - 本主題中的資訊僅適用於以 Amazon Linux AMI (AL1) 為基礎的平台分支。AL2023/AL2 平台分支與舊版 Amazon Linux AMI (AL1) 平台版本不相容，需要不同的組態設定。
> - 2022 年 7 月 18 日，Elastic Beanstalk 已將所有以 Amazon Linux AMI (AL1) 為基礎的平台分支狀態設為已淘汰。如需有關遷移至完全支援的目前 Amazon Linux 2023 平台分支的詳細資訊，請參閱 將您的 Elastic Beanstalk Linux 應用程式遷移到 Amazon Linux 2023 或 Amazon Linux 2。

連接埠傳遞 — Amazon Linux AMI (AL1)

> **Note**
>
> 本主題中的資訊僅適用於以 Amazon Linux AMI (AL1) 為基礎的平台分支。AL2023/AL2 平台分支與舊版 Amazon Linux AMI (AL1) 平台版本不相容，需要不同的組態設定。

Elastic Beanstalk 會設定 nginx 代理程式，將對您應用程式的請求轉送至應用程式 PORT 環境屬性指定的連接埠號碼。您的應用程式應隨時接聽該連接埠。您可呼叫 `os.Getenv("PORT")` 方法，藉此於應用程式內存取此變數。

Elastic Beanstalk 會使用 PORT 環境屬性指定的連接埠號碼，做為 `Procfile` 中第一個應用程式的連接埠，然後對 `Procfile` 中的後續應用程式所用之連接埠號碼，以 100 遞增。若未設定 PORT 環境屬性，Elastic Beanstalk 會使用 5000 做為初始連接埠。

在上述範例中，`web` 應用程式的 PORT 環境屬性為 5000，`queue_process` 應用程式使用 5100，而 `foo` 應用程式使用 5200。

您可透過 aws:elasticbeanstalk:application:environment 命名空間設定 PORT 選項，藉此指定初始連接埠，如下列範例所示。

```
option_settings:
  - namespace:  aws:elasticbeanstalk:application:environment
    option_name:  PORT
    value:  <first_port_number>
```

如需應用程式設定環境屬性的詳細資訊，請參閱選項設定。

透過 Buildfile 在伺服器上建置可執行檔

欲針對 Go 應用程式指定自訂的建置和組態命令，請於原始碼套件的根目錄，納入名為 `Buildfile` 的檔案。檔案名稱區分大小寫。`Buildfile` 採用下列格式：

```
<process_name>: <command>
```

`Buildfile` 中的命令必須符合下列規則表達式：`^[A-Za-z0-9_]+:\s*.+$`。

Elastic Beanstalk 不會監控透過 `Buildfile` 執行的應用程式。針對短期執行且須在任務完成後終止的命令，請使用 `Buildfile`。針對長期執行且不應退出的應用程式程序，請使用 Procfile。

在下列 Buildfile 的範例中，build.sh 為位於原始碼套件根目錄的 shell 指令碼：

```
make: ./build.sh
```

Buildfile 內的所有路徑均相對於原始碼套件的根目錄。若您預先知道這些檔案於執行個體內的位置，您可在 Buildfile 納入絕對路徑。

設定反向代理程式

Elastic Beanstalk 使用 nginx 做為反向代理伺服器，在連接埠 80 上將您的應用程式映射到 Elastic Load Balancing 負載平衡器。Elastic Beanstalk 提供了預設的 nginx 組態，您可以加以擴展，或使用自己的組態將其完全覆寫。

Elastic Beanstalk 預設會設定 nginx 代理將請求轉送至連接埠 5000 上的應用程式。您可將 PORT 環境屬性設定為主要應用程式接聽的連接埠，藉此覆寫預設連接埠。

> **Note**
>
> 您應用程式接聽的連接埠，不會影響 nginx 伺服器為接收來自負載平衡器的請求所接聽的連接埠。

在您的平台版本上設定代理伺服器

所有 AL2023/AL2 平台皆支援統一的代理組態功能。如需有關在執行 AL2023/AL2 的平台版本上設定代理伺服器的詳細資訊，請展開the section called “擴充 Linux 平台”中的反向代理組態一節。

在 Amazon Linux AMI (之前的 Amazon Linux 2) 上設定代理

> **備註**
>
> - 本主題中的資訊僅適用於以 Amazon Linux AMI (AL1) 為基礎的平台分支。AL2023/AL2 平台分支與舊版 Amazon Linux AMI (AL1) 平台版本不相容，需要不同的組態設定。
> - 2022 年 7 月 18 日，Elastic Beanstalk 已將所有以 Amazon Linux AMI (AL1) 為基礎的平台分支狀態設為已淘汰。如需有關遷移至完全支援的目前 Amazon Linux 2023 平台分支的詳細資訊，請參閱 將您的 Elastic Beanstalk Linux 應用程式遷移到 Amazon Linux 2023 或 Amazon Linux 2。

如果您的 Elastic Beanstalk Go 環境使用 Amazon Linux AMI 平台版本 (之前的 Amazon Linux 2)，請閱讀本節中的資訊。

擴展和覆寫預設的代理組態 — Amazon Linux AMI (AL1)

Elastic Beanstalk 使用 nginx 做為反向代理程式，來將您的應用程式對應到連接埠 80 上的負載平衡器。若您想要提供自己的 nginx 組態，您可以將 `.ebextensions/nginx/nginx.conf` 檔案納入原始碼套件，藉此覆寫 Elastic Beanstalk 提供的預設組態。只要此檔案存在，Elastic Beanstalk 會用其取代預設的 nginx 組態檔案。

若您欲納入 `nginx.conf` `http` 區塊外的指令，您亦可於原始碼套件的 `.ebextensions/nginx/conf.d/` 目錄提供其他組態檔案。此目錄的所有檔案都必須採用 `.conf` 做為副檔名。

欲利用 Elastic Beanstalk 提供的功能 (例如增強型運作狀態報告與監控、自動應用程式映射，以及靜態檔案)，您必須於 nginx 組態檔案的 `server` 區塊納入下列行：

```
include conf.d/elasticbeanstalk/*.conf;
```

在 Elastic Beanstalk 建立和部署 Java 應用程式

AWS Elastic Beanstalk 可支援適用於 Java 應用程式的兩種平台。

- Tomcat – 以 Apache Tomcat 為基礎的平台，它是一種開放原始碼的 Web 容器，適用於使用 Java Servlet 和 JavaServer Pages (JSP) 來處理 HTTP 請求的應用程式。Tomcat 提供了多執行緒、宣告式的安全組態與豐富的自訂功能，來協助 Web 應用程式的開發。Elastic Beanstalk 為每個 Tomcat 目前主要版本都提供平台分支。如需詳細資訊，請參閱Tomcat 平台。
- Java SE – 這種平台適用於不使用 Web 容器的應用程式，或使用 Jetty 或 GlassFish 等 Tomcat 之外容器的應用程式。您可以在部署到 Elastic Beanstalk 的原始碼套件中，加入應用程式所使用的任何程式庫 Java Archive (JAR)。如需詳細資訊，請參閱Java SE 平台。

Tomcat 和 Java SE 平台的最新分支都是以 Amazon Linux 2 和更新版本為基礎，並使用 Corretto – AWS Java SE 分發。平台清單中的這些分支名稱包含 Corretto 此字，而不是 Java，例如 `Corretto 11 with Tomcat 8.5`。

如需目前平台版本清單，請參閱 AWS Elastic Beanstalk 平台指南中的 Tomcat 和 Java SE。

AWS 提供了幾種工具來使用 Java 與 Elastic Beanstalk。無論選擇何種平台分支，您都可以透過 適用於 Java 的 AWS 開發套件，從您的 Java 應用程式使用其他 AWS 服務。適用於 Java 的 AWS開發套

件是一組程式庫，可讓您透過應用程式的程式碼來使用 AWS API，而不需再重新編寫原始的 HTTP 呼叫。

如果您使用 Eclipse 整合開發環境 (IDE) 來開發 Java 應用程式，也可以使用 AWS Toolkit for Eclipse。AWS Toolkit for Eclipse 是開源的外掛程式，可讓您從 Eclipse IDE 中管理 AWS 資源，包括 Elastic Beanstalk 應用程式與環境。

如果您偏好使用命令列，可以安裝 Elastic Beanstalk 命令列界面 (EB CLI)，然後藉此從命令列來建立、監控和管理您的 Elastic Beanstalk 環境。如果您針對應用程式執行了多個環境。則 EB CLI 會與 Git 整合，來讓您建立每個環境與不同 Git 分支的關聯。

本章中的主題假設您對 Elastic Beanstalk 環境已有一定瞭解。如果您沒使用過 Elastic Beanstalk，請嘗試《入門教學》以了解基本概念。

主題

Elastic Beanstalk 上的 Java 入門

若要開始在 AWS Elastic Beanstalk 上使用 Java 應用程式，您只需要一個應用程式原始碼套件，來上傳做為第一個應用程式版本，並且部署到環境中。當您建立環境時，Elastic Beanstalk 會設定所有需要的 AWS 資源，以執行可擴展的 Web 應用程式。

使用範例 Java 應用程式來啟動環境

Elastic Beanstalk 為各個平台提供單頁式範例應用程式，並提供較為複雜的範例，說明如何使用諸如 Amazon RDS 的其他 AWS 資源、語言或平台特定功能以及 API。

這些單頁範例的程式碼，和您未提供自己的原始碼即建立環境時所得到的相同。GitHub 上託管了更為複雜的範例，這些範例可能需要先行編譯或建置，才能部署到 Elastic Beanstalk 環境。

範例

名稱	支援的版本	環境類型	來源	描述
Tomc (單頁)	所有 Tomcat with Corretto 平台分支	Web 伺服器 工作者	tomcat.zip	Tomcat Web 應用程式，包含設定要在網站根目錄顯示的單一頁面 (`index.jsp`)。 針對工作者環境，此範例包含了 `cron.yaml` 檔案，其中設定了排程的任務，每分鐘呼叫 `scheduled.jsp` 一次。當 `scheduled.jsp` 受到呼叫時，會寫入位於 `/tmp/sample-app.log` 的日誌檔案。最後，`.ebextensions` 中包含了組態檔案，該檔案會在您請求環境日誌時，將日誌從 `/tmp/` 複製到 Elastic Beanstalk 所讀取的位置。 如果您在執行此範例的環境中啟用 X-Ray 整合功能，則此應用程式會顯示關於 X-Ray 的其他內容，並提供選項，來產生可在 X-Ray 主控台中檢視的除錯資訊。
Corre (單頁)	Corretto 11 Corretto 8	Web 伺服器	corretto.zip	包含 `Buildfile` 與 `Procfile` 組態檔案的 Corretto 應用程式。 如果您在執行此範例的環境中啟用 X-Ray 整合功能，則此應用程式會顯示關於 X-Ray 的其他內容，並提供選項，來產生可在 X-Ray 主控台中檢視的除錯資訊。
Score	Java 8	Web 伺服器	複製 GitHub.com 的儲存庫	Scorekeep 是一種 RESTful web API，使用 Spring 架構來提供界面，此界面可用來建立和管理使用者、工作階段與遊戲。此 API 套裝隨附 Angular 1.5 Web 應用程式，這個應用程式可透過 HTTP 使用此 API。

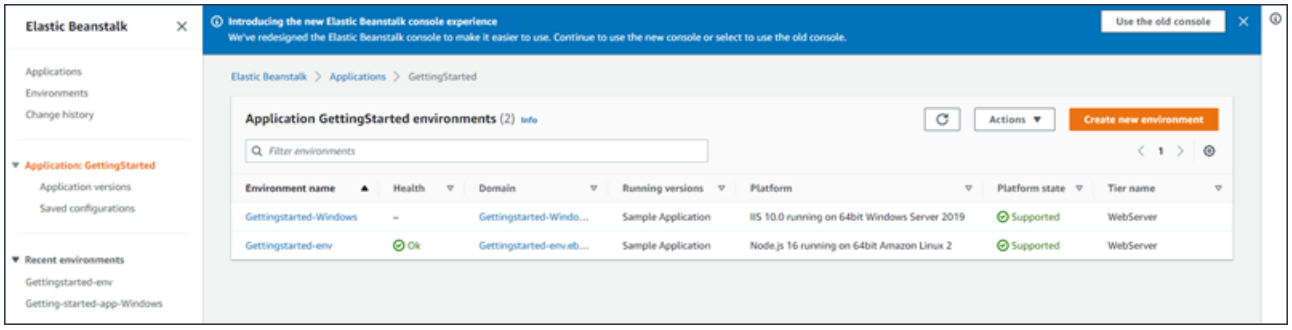
名稱	支援的版本	環境類型	來源	描述
				此應用程式使用 Java SE 平台的功能，來下載相依項目和建置啟動執行個體，將原始碼套件的檔案大小減到最小。此應用程式亦包含 nginx 組態檔案，可覆寫預設組態，藉由代理的連接埠 80 供前端 Web 應用程式靜態使用，而路由則會要求 `/api` 底下運作於 `localhost:5000` 的 API 的路徑。 Scorekeep 也包含了 `xray` 的分支，此分支顯示如何備妥 Java 應用程式以搭配 AWS X-Ray 使用。此分支顯示了使用 Servlet 篩選條件來檢測傳入的 HTTP 請求、自動和手動進行 AWS 開發套件用戶端的檢測、記錄器的組態，和檢測傳出的 HTTP 請求與 SQL 用戶端。 請參閱讀我檔案取得相關說明，或使用 AWS X-Ray 入門教學嘗試搭配應用程式和 X-Ray。
Does it Have Snak	Tomcat 8 搭配 Java 8	Web 伺服器	複製 GitHub.com 的儲存庫	Does it Have Snakes? 是一種 Tomcat Web 應用程式，會針對 Elastic Beanstalk 組態檔案、Amazon RDS、JDBC、PostgreSQL、Servlet、JSP、Simple Tag Support、標記檔案、Log4J、Bootstrap 與 Jackson，顯示使用的狀況。 此專案的原始程式碼包含了最低程度的建置指令碼，這些指令碼會將 Servlet 和模型編譯為類別檔案，並將所需的檔案封裝為 Web 封存檔案，此等封裝檔案可以部署到 Elastic Beanstalk 環境。如需完整說明，請參閱專案儲存庫中的 readme 檔案。

名稱	支援的版本	環境類型	來源	描述
Locust Load Gene	Java 8	Web 伺服器	複製 GitHub.com 的儲存庫	Web 應用程式，您可以用來針對不同 Elastic Beanstalk 環境中執行的另一個 Web 應用程式，進行負載測試。顯示 `Buildfile` 和 `Procfile` 檔案、DynamoDB 與 Locust 的使用狀況，Locust 是一種開放原始碼的負載測試工具。

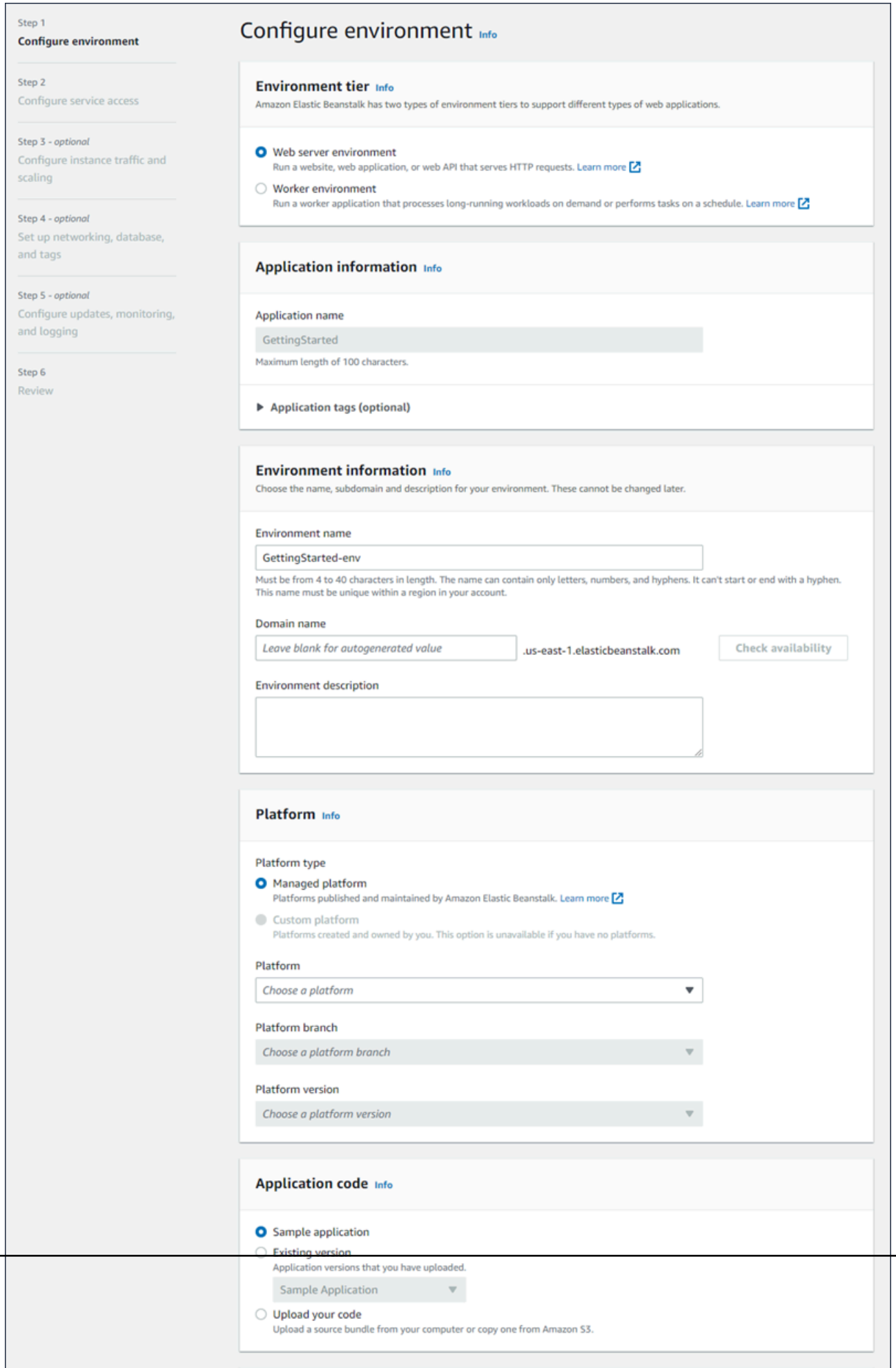
下載任一範例應用程式，並依下列步驟將其部署至 Elastic Beanstalk：

欲使用範例應用程式來啟動環境 (主控台)

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇應用程式，然後選擇清單上目前的應用程式名稱或建立一個。
3. 在應用程式概觀頁面上，選擇 Create a new environment (建立新環境)。



這會啟動 Create environment (建立環境) 精靈。精靈提供一組建立新環境的步驟。


Step 1
Configure environment
Step 2
Configure service access
Step 3 - optional
Configure instance traffic and scaling
Step 4 - optional
Set up networking, database, and tags
Step 5 - optional
Configure updates, monitoring, and logging
Step 6
Review
Configure environment Info
Environment tier Info
Amazon Elastic Beanstalk has two types of environment tiers to support different types of web applications.
Web server environment
Run a website, web application, or web API that serves HTTP requests. Learn more
Worker environment
Run a worker application that processes long-running workloads on demand or performs tasks on a schedule. Learn more
Application information Info
Application name
GettingStarted
Maximum length of 100 characters.
Application tags (optional)
Environment information Info
Choose the name, subdomain and description for your environment. These cannot be changed later.
Environment name
GettingStarted-env
Must be from 4 to 40 characters in length. The name can contain only letters, numbers, and hyphens. It can't start or end with a hyphen. This name must be unique within a region in your account.
Domain name
Leave blank for autogenerated value
.us-east-1.elasticbeanstalk.com
Check availability
Environment description
Platform Info
Platform type
Managed platform
Platforms published and maintained by Amazon Elastic Beanstalk. Learn more
Custom platform
Platforms created and owned by you. This option is unavailable if you have no platforms.
Platform
Choose a platform
Platform branch
Choose a platform branch
Platform version
Choose a platform version
Application code Info
Sample application
Existing version
Application versions that you have uploaded.
Sample Application
Upload your code
Upload a source bundle from your computer or copy one from Amazon S3.

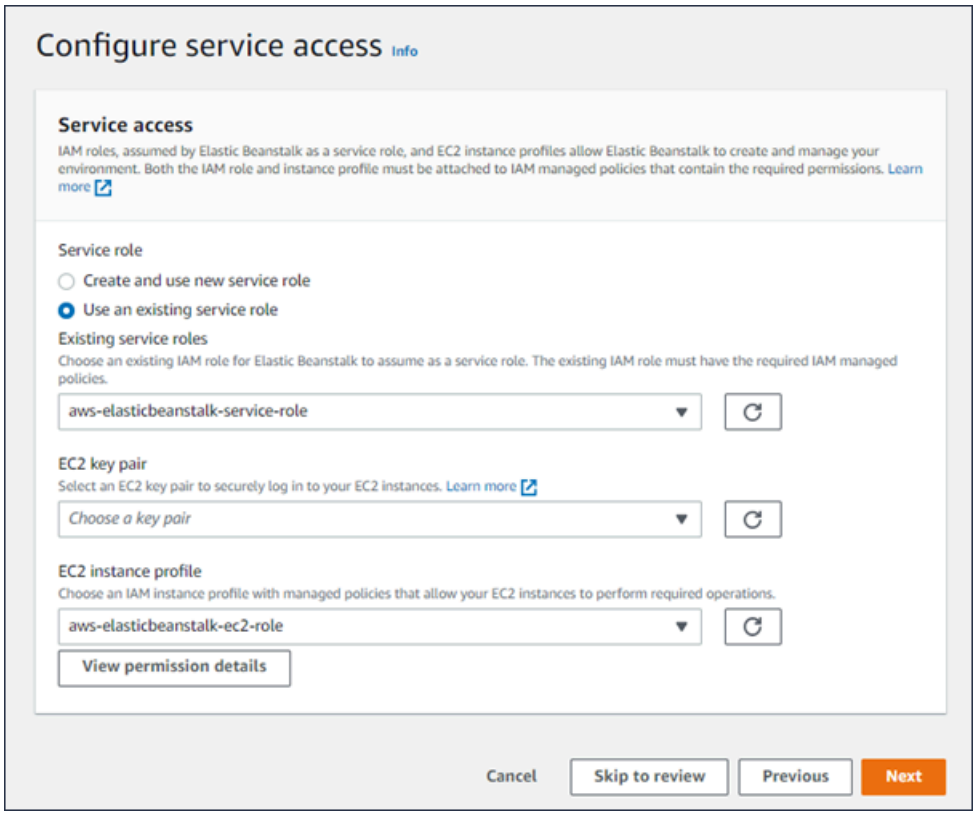
4. 關於環境階層，選擇 Web server environment (Web 伺服器環境) 或 Worker environment (工作者環境) 環境階層。建立後您即無法變更環境層。

> **Note**
>
> Windows Server 平台上的 .NET 不支援工作者環境層。

5. 在 Platform (平台)，選取符合您應用程式所使用語言的平台和平台分支。

> **Note**
>
> Elastic Beanstalk 支援所列出大多數平台的多個版本。根據預設，主控台會針對您選擇的平台和平台分支，選取建議的版本。如果您的應用程式需要不同的版本，您可以在這裡選取。如需支援的平台版本的相關詳細資訊，請參閱 the section called “支援的平台”。

6. 針對 Application code (應用程式程式碼)，選擇 Sample application (範例應用程式)。
7. 關於 Configuration presets (組態預設)，選擇 Single instance (單一執行個體)。
8. 選擇 Next (下一步)。
9. 畫面上會顯示設定服務存取頁面。



10. 針對服務角色，選擇使用現有服務角色。
11. 接下來，我們會將重點放在 EC2 執行個體設定檔下拉式清單。此下拉式清單中顯示的值可能會有所不同，視您的帳戶是否先前已建立新環境而定。

    根據清單中顯示的值，選擇下列其中一項。

    - 如果下拉式清單中顯示 `aws-elasticbeanstalk-ec2-role`，請從 EC2 執行個體設定檔下拉式清單中選取它。
    - 如果清單中顯示另一個值，而且它是適用於您環境的預設 EC2 執行個體設定檔，請從 EC2 執行個體設定檔下拉式清單中選取該值。
    - 如果 EC2 執行個體設定檔下拉式清單未顯示任何可供選擇的值，請展開隨後的程序，為 EC2 執行個體設定檔建立 IAM 角色。

      完成為 EC2 執行個體設定檔建立 IAM 角色中的步驟來建立 IAM 角色，接下來，您可以為 EC2 執行個體設定檔選取該角色。然後，請返回此步驟。

      現在您已建立 IAM 角色，重新整理清單，該角色就會在下拉式清單中顯示為選項。從 EC2 執行個體設定檔下拉式清單中選取您剛建立的 IAM 角色。
12. 在 Configure service access (設定服務存取)頁面上選擇 Skip to Review (略過以檢閱)。

    這會選取此步驟的預設值，並略過選擇性步驟。
13. Review (檢閱)頁面會顯示您所有選擇的摘要。

    若要進一步自訂您的環境，請選擇步驟旁的 Edit (編輯)，其中包含您想要設定的任何項目。下列選項僅能於環境建立期間進行設定：

    - 環境名稱
    - 網域名稱
    - 平台版本
    - 處理器
    - VPC
    - 層

    下列設定可於環境建立後變更，但需要佈建新的執行個體或其他資源，且套用時間可能較長：

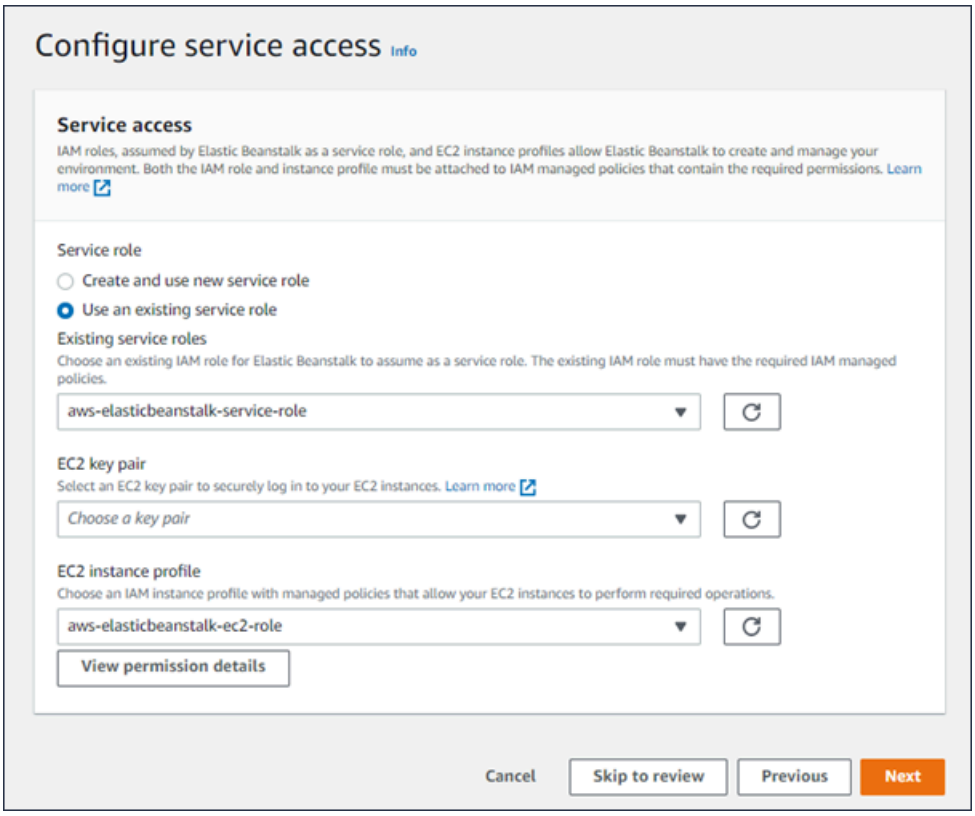
    - 執行個體類型、根磁碟區、金鑰對和 AWS Identity and Access Management (IAM) 角色
    - 內部 Amazon RDS 資料庫

- 負載平衡器

如需所有可用設定的詳細資訊，請參閱建立新的環境精靈。

14. 選擇頁面底部的 Submit (提交)，以將建立的新環境初始化。

為 EC2 執行個體設定檔建立 IAM 角色



若要建立 EC2 執行個體設定檔的 IAM 角色

1. 選擇檢視許可詳細資料。這會顯示在 EC2 執行個體設定檔下拉式清單下。

   畫面上會顯示標題為檢視執行個體設定檔許可的模態視窗。此視窗會列出您需要附加至您建立的新 EC2 執行個體設定檔的受管設定檔。它也提供啟動 IAM 主控台的連結。

2. 選擇顯示於視窗頂端的 IAM 主控台連結。
3. 在 IAM 主控台導覽窗格中，選擇 Roles (角色)。
4. 選擇 建立角色。
5. 在受信任的實體類型下，選擇 AWS 服務。
6. 在 Use case (使用案例) 下，選擇 EC2。

7. 選擇 Next (下一步)。
8. 附加合適的受管政策。在檢視執行個體設定檔許可模態視窗中捲動，查看受管政策。這些政策也會列在此處：
   - `AWSElasticBeanstalkWebTier`
   - `AWSElasticBeanstalkWorkerTier`
   - `AWSElasticBeanstalkMulticontainerDocker`
9. 選擇 Next (下一步)。
10. 輸入角色的名稱。
11. (選用) 附加標籤至角色。
12. 選擇 建立角色。
13. 返回開啟的 Elastic Beanstalk 主控台視窗。
14. 關閉檢視執行個體設定檔許可模態視窗

> **Important**
>
> 請勿關閉顯示 Elastic Beanstalk 主控台的瀏覽器頁面。

15. 選擇 EC2 執行個體設定檔下拉式清單旁的

    (重新整理)。

    這會重新整理下拉式清單，讓您剛建立的「角色」會顯示在下拉式清單中。

後續步驟

在您擁有執行應用程式的環境後，可以隨時部署應用程式的新版本或完全不同的應用程式。部署新的應用程式版本非常快速，因無須佈建或重新啟動 EC2 執行個體。

在您部署好一兩個範例應用程式，並準備好開始在本機上開發和執行 Java 應用程式之後，請參閱下一節的內容，利用您將會用到的所有工具和程式庫，來設定 Java 開發環境。

設定您的 Java 開發環境

設定 Java 開發環境於本機測試您的應用程式，之後再部署至 AWS Elastic Beanstalk。此主題概述開發環境設定步驟，並提供實用工具的安裝頁面連結。

如需了解適用所有語言的常見設定步驟和工具，請參閱設定您的開發機器。

章節

- 安裝 Java 開發套件
- 安裝 Web 容器
- 下載程式庫
- 安裝適用於 Java 的 AWS 開發套件
- 安裝 IDE 或文字編輯器
- 安裝 AWS Toolkit for Eclipse

安裝 Java 開發套件

安裝 Java 開發套件 (JDK)。若您沒有特別需求，請取得最新版本。請至 oracle.com 下載 JDK

JDK 內含 Java 編譯器，您可用其將來源檔案建立為可於 Elastic Beanstalk Web 伺服器上執行的類別檔案。

安裝 Web 容器

若您還沒有另一個 Web 容器或架構，請安裝適當版本的 Tomcat：

- 下載 Tomcat 8 (需要 Java 7 或更新版本)
- 下載 Tomcat 7 (需要 Java 6 或更新版本)

下載程式庫

Elastic Beanstalk 平台預設包含幾個程式庫。下載您應用程式將使用的程式庫，並將其儲存在您的專案資料夾，以部署至應用程式原始碼套件。

若您已於本機安裝 Tomcat，可自安裝資料夾複製 servlet API 和 JavaServer Pages (JSP) API 程式庫。若您部署至 Tomcat 平台版本，則這些檔案無須納入原始碼套件，但必須納入 `classpath` 以編譯使用這些檔案的類別。

JUnit、Google Guava 和 Apache Commons 提供多種實用的程式庫。若要進一步了解，請造訪其首頁：

- 下載 JUnit
- 下載 Google Guava

- 下載 Apache Commons

安裝適用於 Java 的 AWS 開發套件

若您需要從應用程式內管理 AWS 資源，請安裝適用於 Java 的 AWS 開發套件。例如，搭配 AWS SDK for Java，您可以使用 Amazon DynamoDB (DynamoDB)，跨多部 Web 伺服器共享 Apache Tomcat 應用程式的工作階段狀態。如需詳細資訊，請參閱適用於 Java 的 AWS 開發套件文件中的使用 Amazon DynamoDB 來管理 Tomcat 工作階段狀態。

如需詳細資訊及安裝說明，請造訪適用於 Java 的 AWS 開發套件首頁。

安裝 IDE 或文字編輯器

整合開發環境 (IDE) 提供可加速應用程式開發的各種功能。若您尚未使用 IDE 進行 Java 開發，請嘗試 Eclipse 和 IntelliJ，看哪個更適合您。

- 安裝 Eclipse IDE for Java EE Developers (Java EE 開發人員專用 Eclipse IDE
- 安裝 IntelliJ

> **Note**
>
> IDE 可能會於專案資料夾新增您不希望遞交給來源控制的檔案。欲避免將這些檔案遞交給來源控制，請使用 `.gitignore` 或等同來源控制工具的功能。

若您只想開始編碼且不需要 IDE 的所有功能，請考慮安裝 Sublime Text。

安裝 AWS Toolkit for Eclipse

AWS Toolkit for Eclipse 是 Eclipse Java IDE 的開源外掛程式，可讓開發人員使用 AWS 更輕鬆地開發、除錯與部署 Java 應用程式。如需安裝指示，請造訪 AWS Toolkit for Eclipse 首頁。

使用 Elastic Beanstalk Tomcat 平台

> **Important**
>
> AWS Elastic Beanstalk 將 Log4j 從 Amazon Linux 預設套件存儲庫安裝到適用於 Amazon Linux 1 和 Amazon Linux 2 的 Tomcat 平台中。在 Amazon Linux 1 和 Amazon Linux 2 存儲庫中提供的 Log4j 版本不受其預設組態中 CVE-2021-44228 或 CVE-2021-45046 的影響。

如果您對應用程式所使用的 log4j 進行了組態變更，或者安裝了較新版本的 log4j，則建議您採取動作更新應用程式的程式碼以緩解此問題。
出於謹慎，Elastic Beanstalk 發佈了使用最新 Amazon Linux 預設套件存儲庫的新平台版本，其中包含 Log4j hotpatched JDK，在於 2021 年 12 月 21 日發佈的 Amazon Linux 平台版本中。如果您已將 log4j 安裝自訂為應用程式相依項，建議您升級到最新的 Elastic Beanstalk 平台版本，以降低 CVE-2021-44228 或 CVE-2021-45046 的影響。作為正常更新實踐的一部分，您還可以啟用自動化受管更新。
如需有關 Amazon Linux 安全相關軟體更新的詳細資訊，請造訪 Amazon Linux 安全中心。

AWS Elastic Beanstalk Tomcat 平台是一組適用於 Java 網路應用程式的環境資訊，可在 Tomcat web 容器中執行。Tomcat 會在 nginx 代理服務器後面執行。每個平台會對應到主要的 Tomcat 版本，如使用 Tomcat 8 的 Java 8。

Elastic Beanstalk 主控台中提供了修改正在執行環境組態的組態選項。要避免在終止環境的組態時遺失組態，您可以使用已儲存組態來儲存您的設定，並在之後套用至另一個環境。

若要將設定儲存於原始程式碼，您可以包含組態檔案。每次您建立環境或部署應用程式，組態檔案裡的設定就會套用。您也可以使用組態檔案來安裝套件、執行指令碼，並在部署期間執行其他執行個體自訂操作。

Elastic Beanstalk Tomcat 平台包含了反向代理程式，可將請求轉送給您的應用程式。您可使用組態選項來設定代理伺服器，從您原始程式碼中的資料夾提供靜態資產，以減輕應用程式的負擔。在進階的使用情境中，您可以在您的原始碼套件中，加入自己的 `.conf` 檔案，來擴展或完全覆寫掉 Elastic Beanstalk 的代理組態。

> **Note**
>
> Elastic Beanstalk 支援 nginx (預設值) 和 Apache HTTP 伺服器作為 Tomcat 平台上的代理伺服器。如果您的 Elastic Beanstalk Tomcat 環境使用 Amazon Linux AMI 平台分支 (Amazon Linux 2 之前)，您也可以選擇使用 Apache HTTP 伺服器版本 2.2。在這些較舊平台分支上會預設使用 Apache 最新版。
> 2022 年 7 月 18 日，Elastic Beanstalk 已將所有以 Amazon Linux AMI (AL1) 為基礎的平台分支狀態設為已淘汰。如需有關遷移至完全支援的目前 Amazon Linux 2023 平台分支的詳細資訊，請參閱 將您的 Elastic Beanstalk Linux 應用程式遷移到 Amazon Linux 2023 或 Amazon Linux 2。

您必須使用特定的架構，來將 Java 應用程式的套件，打包為 Web 應用程式封裝 (WAR) 格式的檔案。關於所需的架構，以及該架構與您專案目錄架構之間的關係，詳細資訊請參閱建構您的專案資料夾。

若要在同一個 Web 伺服器上執行多個應用程式，您可以將多個 WAR 檔案打包為單一原始碼套件。在包含多個 WAR 原始碼的套件中，取決於 WAR 的名稱，每個應用程式會在根路徑 (ROOT.war 在 *myapp*.elasticbeanstalk.com/ 中執行) 或根路徑之下直屬的路徑 (app2.war 在 *myapp*.elasticbeanstalk.com/*app2*/ 中執行) 執行。在單一 WAR 原始碼套件中，應用程式永遠會在根路徑執行。

在 Elastic Beanstalk 主控台中套用的設定會覆寫組態檔案中相同的設定 (如存在)。這可讓您在組態檔案中擁有預設設定，並以主控台的環境專屬設定覆寫之。如需優先順序以及其他變更設定方法的詳細資訊，請參閱組態選項。

如需各種擴充 Elastic Beanstalk Linux 類型平台方式的詳細資訊，請參閱 the section called “擴充 Linux 平台”。

主題

- 設定您的 Tomcat 環境
- Tomcat 組態命名空間
- 綁定多個用於 Tomcat 環境的 WAR 檔案
- 建構您的專案資料夾
- 設定您的 Tomcat 環境的代理伺服器

設定您的 Tomcat 環境

Elastic Beanstalk Tomcat 平台除了所有平台皆有的標準選項外，還提供幾個平台特定的選項。這些選項可讓您設定在您環境 web 伺服器上執行的 Java 虛擬機器 (JVM)，和定義系統屬性，來為應用程式提供資訊組態字串。

您可以使用 Elastic Beanstalk 主控台來啟用至 Amazon S3 的日誌輪換，和設定您應用程式可從環境讀取的變數。

在 Elastic Beanstalk 主控台中設定您的 Tomcat 環境

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇組態。
4. 在更新、監控和日誌記錄組態類別中，選擇編輯。

容器選項

您可以指定這些平台特有的選項：

- 代理伺服器 – 要在您的環境執行個體上使用的代理伺服器。預設使用 nginx。

JVM 容器選項

Java 虛擬機器 (JVM) 中的堆積大小，決定了在記憶體回收發生之前，您的應用程式可以在記憶體中建立多少物件。您可以修改初始 JVM 堆積大小 (`-Xms option`) 和 JVM 堆積大小上限 (`-Xmx` 選項)。較大的初始堆積大小可在記憶體回收進行之前，讓更多的物件能夠建立。但這也表示記憶體回收器會花費更多的時間來壓縮堆疊。堆積大小的上限，指定了在繁重的作業進行期間擴充堆疊時，JVM 可以分配的最大記憶體容量。

> **Note**
>
> 可用的記憶體取決於 Amazon EC2 執行個體類型。如需可供您 Elastic Beanstalk 環境使用之 EC2 執行個體類型的詳細資訊，請參閱適用於 Linux 執行個體的 Amazon Elastic Compute Cloud 使用者指南中的執行個體類型。

永久保存區是 JVM 堆積的一個區段，用來儲存類別定義與相關的中繼資料。若要修改永久保存區的大小，請在 JVM 永久保存區大小上限 (`-XX:MaxPermSize`) 選項中輸入新的大小。此設定僅適用於 Java 7 和較舊的版本。此選項已在 JDK 8 中棄用，並由 MaxMetaspace Size (`-XX:MaxMetaspaceSize`) 選項取代。

> **Important**
>
> JDK 17 移除了對 Java `-XX:MaxPermSize` 選項的支援。將此選項與具有 Corretto 17 的 Elastic Beanstalk 平台分支上執行的環境搭配使用將導致錯誤。Elastic Beanstalk 於 2023 年 7 月 13 日發布了第一個執行 Tomcat 與 Corretto 17 的平台分支。
> 如需詳細資訊，請參閱下列資源。
>
> - Oracle Java 文件網站：已刪除 Java 選項
> - Oracle Java 文件網站：其他考量事項中的類別中繼資料區段

如需有關 Elastic Beanstalk 平台及其元件的詳細資訊，請參閱《AWS Elastic Beanstalk 平台指南》中支援的平台。

日誌選項

Log Options (日誌選項) 區段有兩個設定：

- 執行個體設定檔 – 指定有權存取與您應用程式相關的 Amazon S3 儲存貯體的執行個體設定檔。
- Enable log file rotation to Amazon S3 (啟用 Amazon S3 的日誌檔案輪換) – 指定是否將應用程式 Amazon EC2 執行個體的日誌檔案複製到與應用程式關聯的 Amazon S3 儲存貯體。

靜態檔案

為改善效能，您可以使用 Static files (靜態檔案) 區段來設定代理伺服器，以為來自 Web 應用程式一組目錄中的靜態檔案 (例如 HTML 或影像) 提供服務。對於每個目錄，您可以設定目錄映射的虛擬路徑。代理伺服器收到位於指定路徑下的檔案請求時，會直接提供檔案而非將請求路由至您的應用程式。

如需使用組態檔案或 Elastic Beanstalk 主控台設定靜態檔案的詳細資訊，請參閱the section called “靜態檔案”。

環境屬性

在 Environment Properties (環境屬性) 區段中，您可以針對執行應用程式的 Amazon EC2 執行個體來指定其上的環境資訊設定。環境屬性會以金鑰值對的形式傳到應用程式。

Tomcat 平台會針對 Tomcat 環境，定義名為 `JDBC_CONNECTION_STRING` 的佔位符屬性，用來將連線字串傳遞到外部資料庫。

> **Note**
>
> 如果您將 RDS 資料庫執行個體連接到您的環境，請從 Elastic Beanstalk 提供的 Amazon Relational Database Service (Amazon RDS) 環境屬性動態建構 JDBC 連線字串。對於非使用 Elastic Beanstalk 佈建的資料庫執行個體，請只使用 JDBC_CONNECTION_STRING。
> 關於使用 Amazon RDS 與您的 Java 應用程式的詳細資訊，請參閱將 Amazon RDS 資料庫執行個體新增到您的 Java 應用程式環境。

在 Elastic Beanstalk 內所執行的 Tomcat 環境中，可使用 `System.getProperty()` 來存取環境變數。例如，您可使用下列程式碼，來將名為 `API_ENDPOINT` 的屬性讀取到變數。

```
String endpoint = System.getProperty("API_ENDPOINT");
```

如需詳細資訊，請參閱環境屬性與其他軟體設定。

Tomcat 組態命名空間

您可以使用組態檔案來設定組態選項，並在部署期間執行其他的執行個體設定工作。組態選項可透過 Elastic Beanstalk 服務或您使用的平台來定義，並且會組織成命名空間。

除了適用於所有 Elastic Beanstalk 環境的支援選項之外，Tomcat 也支援使用下列命名空間的選項：

- `aws:elasticbeanstalk:container:tomcat:jvmoptions` - 修改 JVM 設定。使用此命名空間的選項，對應管理主控台中的選項，如下所示：
  - `Xms` - JVM command line options (JVM 命令列選項
  - `JVM Options` - JVM command line options (JVM 命令列選項
- `aws:elasticbeanstalk:environment:proxy` - 選擇環境的代理伺服器。

下列的組態檔案範例展示了 Tomcat 特定組態選項的使用。

Example .ebextensions/tomcat-settings.config

```
option_settings:
  aws:elasticbeanstalk:container:tomcat:jvmoptions:
    Xms: 512m
    JVM Options: '-Xmn128m'
  aws:elasticbeanstalk:application:environment:
```

```
    API_ENDPOINT: mywebapi.zkpexsjtmd.us-west-2.elasticbeanstalk.com
  aws:elasticbeanstalk:environment:proxy:
    ProxyServer: apache
```

Elastic Beanstalk 可提供許多組態選項讓您自訂環境。除了組態檔案，您也可以使用主控台、已儲存組態、EB CLI 或 AWS CLI 來設定組態選項。如需詳細資訊，請參閱組態選項。

Amazon Linux AMI (Amazon Linux 2 之前) Tomcat 平台

如果您的 Elastic Beanstalk Tomcat 環境使用 Amazon Linux AMI 平台版本 (Amazon Linux 2 之前)，請閱讀本節中的其他資訊。

> **備註**
>
> - 本主題中的資訊僅適用於以 Amazon Linux AMI (AL1) 為基礎的平台分支。AL2023/AL2 平台分支與舊版 Amazon Linux AMI (AL1) 平台版本不相容，需要不同的組態設定。
> - 2022 年 7 月 18 日，Elastic Beanstalk 已將所有以 Amazon Linux AMI (AL1) 為基礎的平台分支狀態設為已淘汰。如需有關遷移至完全支援的目前 Amazon Linux 2023 平台分支的詳細資訊，請參閱 將您的 Elastic Beanstalk Linux 應用程式遷移到 Amazon Linux 2023 或 Amazon Linux 2。

Tomcat 組態命名空間 — Amazon Linux AMI (AL1)

Tomcat Amazon Linux AMI 平台在下列命名空間中支援其他選項：

- `aws:elasticbeanstalk:container:tomcat:jvmoptions` - 除了此命名空間之前在此頁面上提到的選項之外，較舊的 Amazon Linux AMI 平台版本也支援：
  - `XX:MaxPermSize` - Maximum JVM permanent generation size (JVM 永久保存區大小上限
- `aws:elasticbeanstalk:environment:proxy` - 除了選擇代理伺服器之外，還會設定回應壓縮。

下列組態檔案範例展示了使用代理命名空間組態選項。

Example .ebextensions/tomcat-settings.config

```
option_settings:
```

```
  aws:elasticbeanstalk:environment:proxy:
    GzipCompression: 'true'
    ProxyServer: nginx
```

包括 Elastic Beanstalk 組態檔案 — Amazon Linux AMI (AL1)

若要部署 `.ebextensions` 設定檔，將其納入您的應用程式原始程式碼。對於單一應用程式，執行下列命令，將 `.ebextensions` 新增至壓縮的 WAR 檔案：

Example

```
zip -ur your_application.war .ebextensions
```

有關需要多個 WAR 檔案的應用程式，請參閱 綁定多個用於 Tomcat 環境的 WAR 檔案 的詳細說明。

綁定多個用於 Tomcat 環境的 WAR 檔案

若您的 Web 應用程式包含多個 Web 應用程式元件，您可在單一環境執行元件，而非針對各個元件執行不同環境，藉此簡化部署並降低作業成本。對於不需要大量資源的輕量型應用程式或開發及測試環境而言，本策略十分有效。

欲將多個 Web 應用程式部署至您的環境，請將各個元件 Web 應用程式封存檔 (WAR) 檔案結合至單一原始碼套件。

欲建立內含多個 WAR 檔案的應用程式原始碼套件，請以下列結構整理 WAR 檔案。

```
MyApplication.zip
### .ebextensions
### .platform
### foo.war
### bar.war
### ROOT.war
```

將內含多個 WAR 檔案的原始碼套件部署至 AWS Elastic Beanstalk 環境時，各個應用程式均可從根網域名稱的不同路徑存取。上述範例包含三個應用程式：`foo`、`bar` 及 `ROOT`。`ROOT.war` 為特殊檔案名稱，可指示 Elastic Beanstalk 於該根網域執行該應用程式，讓三個應用程式可於 `http://MyApplication.elasticbeanstalk.com/foo`、`http://MyApplication.elasticbeanstalk.com/bar` 及 `http://MyApplication.elasticbeanstalk.com` 使用。

原始碼套件可以包括 WAR 文件，一個選用 `.ebextensions` 資料夾和一個選用 `.platform` 資料夾。如需這些選用組態資料夾的詳細資訊，請參閱 the section called “擴充 Linux 平台”。

啟動環境 (主控台)

1. 透過此一預設連結來開啟 Elastic Beanstalk 主控台：console.aws.amazon.com/elasticbeanstalk/home#/newApplication?applicationName=tutorials&environmentType=LoadBalanced
2. 在 Platform (平台)，選取符合您應用程式所使用語言的平台和平台分支，或針對以容器為基礎的應用程式選取 Docker 平台。
3. 在 Application code (應用程式的程式碼)，選擇 Upload your code (上傳您的程式碼)。
4. 選擇 Local file (本機檔案)，選擇 Choose file (選擇檔案)，然後開啟原始碼套件。
5. 選擇 Review and launch (檢閱和啟動)。
6. 檢閱可用的設定，然後選擇 Create app (建立應用程式)。

如需建立原始碼套件的資訊，請參閱 建立應用程式原始碼套件。

建構您的專案資料夾

若要在部署到 Tomcat 伺服器時能夠運作，編譯的 Java Platform Enterprise Edition (Java EE) web 應用程式封存檔 (WAR 檔案) 必須根據特定準則來建構。您的專案目錄不需要滿足相同的標準，但最好也採用相同的結構以簡化編譯和封裝。在建置專案資料夾時，採用與 WAR 檔案內容相同的架構，也有助於您了解檔案彼此的關聯，和檔案在 Web 伺服器上的運作方式。

在下列建議的階層中，會將 web 應用程式的原始程式碼放置於 `src` 目錄中，來與建置指令碼和其所產生的 WAR 檔案隔離。

```
~/workspace/my-app/
|-- build.sh            - Build script that compiles classes and creates a WAR
|-- README.MD           - Readme file with information about your project, notes
|-- ROOT.war            - Source bundle artifact created by build.sh
`-- src                 - Source code folder
    |-- WEB-INF         - Folder for private supporting files
    |   |-- classes     - Compiled classes
    |   |-- lib         - JAR libraries
    |   |-- tags        - Tag files
    |   |-- tlds        - Tag Library Descriptor files
    |   `-- web.xml     - Deployment Descriptor
    |-- com             - Uncompiled classes
    |-- css             - Style sheets
```

```
    |-- images          - Image files
    |-- js              - JavaScript files
    `-- default.jsp     - JSP (JavaServer Pages) webpage
```

src 資料夾的內容符合您將封裝和部署到伺服器的內容，不過未包含 com 資料夾。此 com 資料夾包含您未編譯的類別 (.java 檔案)。這些需要被編譯，並放到您的應用程式程式碼可以存取的 WEB-INF/classes 目錄中。

WEB-INF 目錄中包含不會在 Web 伺服器上公開提供的程式碼與組態。位於來源目錄根的其他資料夾 (css、images 與 js)，則會於 web 伺服器上的對應路徑公開提供。

下列的範例與先前的專案目錄相同，但包含了更多的檔案和子目錄。此範例專案包含簡單的標籤、模型和支援類別，以及 record 資源的 Java Server Pages (JSP) 檔案。此範例還包含 Bootstrap 的樣式表和 JavaScript、預設的 JSP 檔案，以及 404 錯誤的錯誤訊息頁面。

WEB-INF/lib 包含了 Java Archive (JAR) 檔案，內有適用於 PostgreSQL 的 Java Database Connectivity (JDBC) 驅動程式。WEB-INF/classes 為空，因為類別檔案尚未編譯。

```
~/workspace/my-app/
|-- build.sh
|-- README.MD
|-- ROOT.war
`-- src
    |-- WEB-INF
    |   |-- classes
    |   |-- lib
    |   |   `-- postgresql-9.4-1201.jdbc4.jar
    |   |-- tags
    |   |   `-- header.tag
    |   |-- tlds
    |   |   `-- records.tld
    |   `-- web.xml
    |-- com
    |   `-- myapp
    |       |-- model
    |       |   `-- Record.java
    |       `-- web
    |           `-- ListRecords.java
    |-- css
    |   |-- bootstrap.min.css
    |   `-- myapp.css
    |-- images
```

```
    |   `-- myapp.png
    |-- js
    |   `-- bootstrap.min.js
    |-- 404.jsp
    |-- default.jsp
    `-- records.jsp
```

使用 Shell 指令碼來建置 WAR 檔案

build.sh 是極為簡單的 shell 指令碼，可編譯 Java 類別、建構 WAR 檔案，以及將檔案複製到 Tomcat 的 webapps 目錄來進行本機測試。

```
cd src
javac -d WEB-INF/classes com/myapp/model/Record.java
javac -classpath WEB-INF/lib/*:WEB-INF/classes -d WEB-INF/classes com/myapp/model/
Record.java
javac -classpath WEB-INF/lib/*:WEB-INF/classes -d WEB-INF/classes com/myapp/web/
ListRecords.java

jar -cvf ROOT.war *.jsp images css js WEB-INF
cp ROOT.war /Library/Tomcat/webapps
mv ROOT.war ../
```

在 WAR 檔案中，會具有與先前範例的 src 目錄相同的架構 (不包括 src/com 資料夾)。jar 指令會自動建立 META-INF/MANIFEST.MF 檔案。

```
~/workspace/my-app/ROOT.war
|-- META-INF
|   `-- MANIFEST.MF
|-- WEB-INF
|   |-- classes
|   |   `-- com
|   |       `-- myapp
|   |           |-- model
|   |           |   `-- Records.class
|   |           `-- web
|   |               `-- ListRecords.class
|   |-- lib
|   |   `-- postgresql-9.4-1201.jdbc4.jar
|   |-- tags
|   |   `-- header.tag
|   |-- tlds
|   |   `-- records.tld
```

```
|   `-- web.xml
|-- css
|   |-- bootstrap.min.css
|   `-- myapp.css
|-- images
|   `-- myapp.png
|-- js
|   `-- bootstrap.min.js
|-- 404.jsp
|-- default.jsp
`-- records.jsp
```

使用 `.gitignore`

為避免將編譯過的類別檔案和 WAR 檔案提交於您的 Git 儲存庫中，或是在執行 Git 指令時出現關於這些檔案的訊息，請將相關的檔案類別新增至專案資料夾中名為 `.gitignore` 的檔案內。

~/workspace/myapp/.gitignore

```
*.zip
*.class
```

設定您的 Tomcat 環境的代理伺服器

Tomcat 平台使用 nginx (預設值) 或 Apache HTTP 伺服器作為反向代理程式，來將執行個體上 80 連接埠所傳入的要求，轉傳給監聽 8080 埠的 Tomcat Web 容器。Elastic Beanstalk 提供了預設的代理組態，您可以加以擴展，或使用自己的組態將其完全覆寫。

在您的平台版本上設定代理伺服器

所有 AL2023/AL2 平台皆支援統一的代理組態功能。如需有關在執行 AL2023/AL2 的平台版本上設定代理伺服器的詳細資訊，請展開the section called “擴充 Linux 平台”中的反向代理組態一節。

在 Amazon Linux AMI (Amazon Linux 2 之前) Tomcat 平台上設定代理

如果您的 Elastic Beanstalk Tomcat 環境使用 Amazon Linux AMI 平台版本 (Amazon Linux 2 之前)，請閱讀本節中的其他資訊。

> **備註**
>
> - 本主題中的資訊僅適用於以 Amazon Linux AMI (AL1) 為基礎的平台分支。AL2023/AL2 平台分支與舊版 Amazon Linux AMI (AL1) 平台版本不相容，需要不同的組態設定。

- 2022 年 7 月 18 日，Elastic Beanstalk 已將所有以 Amazon Linux AMI (AL1) 為基礎的平台分支狀態設為已淘汰。如需有關遷移至完全支援的目前 Amazon Linux 2023 平台分支的詳細資訊，請參閱 將您的 Elastic Beanstalk Linux 應用程式遷移到 Amazon Linux 2023 或 Amazon Linux 2。

為您的 Tomcat 環境選擇代理伺服器 — Amazon Linux AMI (AL1)

以 Amazon Linux AMI (Amazon Linux 2 之前) 為基礎的 Tomcat 平台版本預設會使用 Apache 2.4 作為代理。您可以選擇使用 Apache 2.2 或 nginx，做法是將組態檔案包含在原始碼中。下列範例設定 Elastic Beanstalk 使用 nginx。

Example .ebextensions/nginx-proxy.config

```
option_settings:
  aws:elasticbeanstalk:environment:proxy:
    ProxyServer: nginx
```

從 Apache 2.2 遷移到 Apache 2.4 — Amazon Linux AMI (AL1)

如果您的應用程式是針對 Apache 2.2 開發，請讀取此部分以了解遷移到 Apache 2.4 相關資訊。

從 Tomcat 平台版本發行 3.0.0 組態入門，與 Java 和更新於 2018 年 5 月 24 日的 Tomcat 平台一起發行，Apache 2.4 是 Tomcat 平台的預設 Proxy。Apache 2.4 `.conf` 檔案大多數與 Apache 2.2 的舊版相容，但不是全部如此。Elastic Beanstalk 包含與每個 Apache 版本正常運作的預設 `.conf` 檔案。如果您的應用程式不自訂 Apache 的組態，如 擴展和覆寫預設的 Apache 組態 — Amazon Linux AMI (AL1) 所述，遷移到 Apache 2.4 應該不會有任何問題。

如果您的應用程式擴展或覆寫 Apache 的組態，您可能需要做一些變更才能遷移到 Apache 2.4。如需詳細資訊，請參閱 The Apache Software Foundation 網站上的從 2.2 升級到 2.4。關於成功地遷移到 Apache 2.4 之前的臨時測量，您可以選擇 Apache 2.2 與您的應用程式搭配使用，做法是將以下組態檔案納入原始碼。

Example .ebextensions/apache-legacy-proxy.config

```
option_settings:
  aws:elasticbeanstalk:environment:proxy:
    ProxyServer: apache/2.2
```

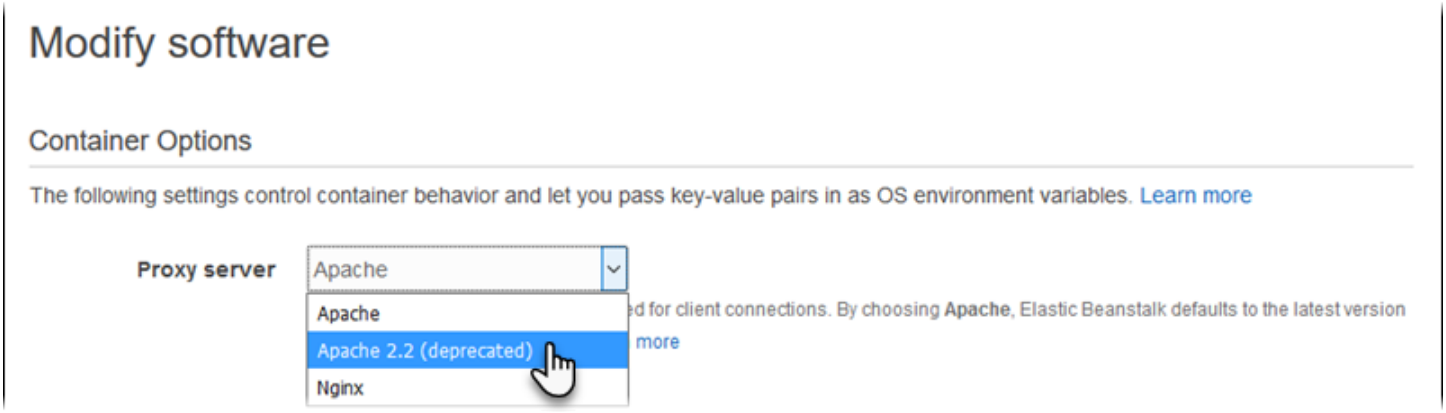
對於快速修正，您也可以選擇 Elastic Beanstalk 主控台的代理伺服器。

在 Elastic Beanstalk 主控台中選擇您 Tomcat 環境的代理

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇組態。
4. 在更新、監控和日誌記錄組態類別中，選擇編輯。
5. 對於 Proxy server (代理伺服器)，請選擇 `Apache 2.2 (deprecated)`。
6. 若要儲存變更，請選擇頁面底部的儲存變更。



擴展和覆寫預設的 Apache 組態 — Amazon Linux AMI (AL1)

您可以擴展 Elastic Beanstalk 預設 Apache 組態與您的其他組態檔案。或者，您可以完全覆寫 Elastic Beanstalk 預設 Apache 組態。

> **Note**
>
> - 所有 Amazon Linux 2 平台皆支援統一的代理組態功能。如需有關在執行 Amazon Linux 2 的 Tomcat 平台版本上設定代理伺服器的詳細資訊，請展開 the section called “擴充 Linux 平台” 中的 Reverse Proxy Configuration (反向代理組態) 區段。
> - 如果您要將 Elastic Beanstalk 應用程式遷移至 Amazon Linux 2 平台，請務必同時閱讀 the section called “遷移至 AL2023/AL2” 中的資訊。

若要擴展 Elastic Beanstalk 預設 Apache 組態，請將 `.conf` 組態檔案加入您應用程式原始碼套件中名為 `.ebextensions/httpd/conf.d` 的資料夾。Elastic Beanstalk Apache 組態會自動在此資料夾中加入 `.conf` 檔案。

```
~/workspace/my-app/
|-- .ebextensions
|   -- httpd
|      -- conf.d
|         -- myconf.conf
|         -- ssl.conf
-- index.jsp
```

例如，下列 Apache 2.4 組態會在 5000 埠上新增接聽程式。

Example .ebextensions/httpd/conf.d/port5000.conf

```
listen 5000
<VirtualHost *:5000>
  <Proxy *>
    Require all granted
  </Proxy>
  ProxyPass / http://localhost:8080/ retry=0
  ProxyPassReverse / http://localhost:8080/
  ProxyPreserveHost on

  ErrorLog /var/log/httpd/elasticbeanstalk-error_log
</VirtualHost>
```

若要完全覆寫 Elastic Beanstalk 預設 Apache 組態，請在 `.ebextensions/httpd/conf/httpd.conf` 的原始碼套件中加入組態。

```
~/workspace/my-app/
|-- .ebextensions
|   `-- httpd
|       `-- conf
|           `-- httpd.conf
`-- index.jsp
```

如果您覆寫了 Elastic Beanstalk Apache 組態，請在 `httpd.conf`中加入下列幾行程式，以提取 增強型運作狀態報告與監控 的 Elastic Beanstalk 組態、回應壓縮和靜態檔案。

```
IncludeOptional conf.d/*.conf
IncludeOptional conf.d/elasticbeanstalk/*.conf
```

如果您的環境中使用 Apache 2.2 做為其 Proxy，請將 `IncludeOptional` 指令取代為 `Include`。如需有關兩個 Apache 版本的這兩個指令行為的詳細資訊，請參閱包含在 Apache 2.4 中、IncludeOptional 在 Apache 2.4 中和包含在 Apache 2.2 中。

> **Note**
>
> 若要覆寫 80 埠的預設接聽程式，請在 `00_application.conf` 中加入名為 `.ebextensions/httpd/conf.d/elasticbeanstalk/` 的檔案，以覆寫 Elastic Beanstalk 的組態。

如需運作範例，請參見位於您環境執行個體上的 Elastic Beanstalk 預設組態檔案 (`/etc/httpd/conf/httpd.conf`)。在部署作業進行期間，您原始碼套件 `.ebextensions/httpd` 資料夾中的所有檔案，都會複製到 `/etc/httpd`。

擴展預設的 nginx 組態 — Amazon Linux AMI (AL1)

若要擴展 Elastic Beanstalk 的預設 nginx 組態，請將 `.conf` 組態檔案加進您應用程式原始碼套件中名為 `.ebextensions/nginx/conf.d/` 的資料夾。Elastic Beanstalk nginx 組態會自動在此資料夾中加入 `.conf` 檔案。

```
~/workspace/my-app/
|-- .ebextensions
|   `-- nginx
|       `-- conf.d
|           |-- elasticbeanstalk
|           |   `-- my-server-conf.conf
|           `-- my-http-conf.conf
`-- index.jsp
```

在預設組態的 `conf.d` 區塊中，會包含 `http` 資料夾中具備 .conf 副檔名的檔案。在 `conf.d/elasticbeanstalk` 區塊的 `server` 區塊中，會包含 `http` 資料夾中的檔案。

若要完全覆寫 Elastic Beanstalk 預設 nginx 組態，請在您原始碼套件的 `.ebextensions/nginx/nginx.conf` 中加入組態。

```
~/workspace/my-app/
```

```
|-- .ebextensions
|   `-- nginx
|       `-- nginx.conf
`-- index.jsp
```

備註

- 如果覆寫 Elastic Beanstalk nginx 組態，請在組態的 `server` 區塊中加入下列幾行程式，以納入 80 埠接聽程式的 Elastic Beanstalk 組態、回應內容壓縮和靜態檔案。

  ```
  include conf.d/elasticbeanstalk/*.conf;
  ```

- 若要覆寫 80 埠的預設接聽程式，請在 `00_application.conf` 中加入名為 `.ebextensions/nginx/conf.d/elasticbeanstalk/` 的檔案，以覆寫 Elastic Beanstalk 的組態。
- 另外也請在您組態的 `http` 區塊中，加入下列幾行程式，以提取 增強型運作狀態報告與監控 的 Elastic Beanstalk 組態和記錄。

  ```
      include       conf.d/*.conf;
  ```

如需運作範例，請參見位於您環境執行個體上的 Elastic Beanstalk 預設組態檔案 (`/etc/nginx/nginx.conf`)。在部署作業進行期間，您原始碼套件 `.ebextensions/nginx` 資料夾中的所有檔案，都會複製到 `/etc/nginx`。

使用 Elastic Beanstalk Java SE 平台

AWS Elastic Beanstalk Java SE 平台是一組適用於 Java Web 應用程式的平台版本，可自行從編譯的 JAR 檔案執行。您可於本機編譯應用程式，或搭配建置指令碼上傳原始碼，藉此在執行個體上進行編譯。Java SE 平台版本被分組到平台分支中，每個分支會對應於 Java 的主要版本，例如 Java 8 和 Java 7。

Note

Elastic Beanstalk 不會剖析應用程式的 JAR 檔案。請將 Elastic Beanstalk 需要的檔案保存在 JAR 檔案外面。例如，將工作者環境的 `cron.yaml` 檔案包含在應用程式原始碼套件的根目錄 (在 JAR 檔案旁邊)。

Elastic Beanstalk 主控台中提供了修改正在執行環境組態的組態選項。要避免在終止環境的組態時遺失組態，您可以使用已儲存組態來儲存您的設定，並在之後套用至另一個環境。

若要將設定儲存於原始程式碼，您可以包含組態檔案。每次您建立環境或部署應用程式，組態檔案裡的設定就會套用。您也可以使用組態檔案來安裝套件、執行指令碼，並在部署期間執行其他執行個體自訂操作。

Elastic Beanstalk Java SE 平台內含可做為反向代理程式的 nginx 伺服器，處理快取靜態內容並將請求傳送至您的應用程式。平台提供了組態選項來設定代理伺服器，從您原始程式碼中的資料夾提供靜態資產，以減輕應用程式的負擔。在進階的使用情境中，您可以在您的原始碼套件中，加入自己的 .conf 檔案，來擴展或完全覆寫掉 Elastic Beanstalk 的代理組態。

如果您只為應用程式來源提供單一 JAR 檔案 (自行提供，而非在原始碼套件中)，Elastic Beanstalk 會將 JAR 檔案重新命名為 `application.jar`，然後使用 `java -jar application.jar` 以執行之。欲設定在環境中伺服器執行個體上執行的程序，請於您的原始碼套件納入選用的 Procfile。若您的原始碼套件根目錄具有多個 JAR 檔案，或者您希望自訂 Java 命令來設定 JVM 選項，則必須使用 `Procfile`。

建議務必在原始碼套件和應用程式中，一併提供 `Procfile`。如此一來，就能精確掌控 Elastic Beanstalk 會針對應用程式執行哪些程序，以及這些程序會收到哪些引數。

欲在部署時編譯 Java 類別並於環境中 EC2 執行個體上執行其他建置命令，請將 Buildfile 納入您的應用程式原始碼套件。`Buildfile` 可讓您依原狀部署原始碼，並可建置在伺服器上，而非於本機編譯 JAR。Java SE 平台包括常見的建置工具，可讓您在伺服器上進行建置作業。

如需各種擴充 Elastic Beanstalk Linux 類型平台方式的詳細資訊，請參閱 the section called “擴充 Linux 平台”。

設定您的 Java SE 環境

Java SE 平台設定可讓您微調 Amazon EC2 執行個體的行為。您可以使用 Elastic Beanstalk 主控台編輯 Elastic Beanstalk 環境的 Amazon EC2 執行個體組態。

使用 Elastic Beanstalk 主控台來啟用至 Amazon S3 的日誌輪換，和設定您的應用程式可以從環境讀取的變數。

在 Elastic Beanstalk 主控台中設定您的 Java SE 環境

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇組態。
4. 在更新、監控和日誌記錄組態類別中，選擇編輯。

日誌選項

Log Options (日誌選項) 區段有兩個設定：

- 執行個體設定檔 – 指定有權存取與您應用程式相關的 Amazon S3 儲存貯體的執行個體設定檔。
- Enable log file rotation to Amazon S3 (啟用 Amazon S3 的日誌檔案輪換) – 指定是否將應用程式 Amazon EC2 執行個體的日誌檔案複製到與應用程式關聯的 Amazon S3 儲存貯體。

靜態檔案

為改善效能，您可以使用 Static files (靜態檔案) 區段來設定代理伺服器，以為來自 Web 應用程式一組目錄中的靜態檔案 (例如 HTML 或影像) 提供服務。對於每個目錄，您可以設定目錄映射的虛擬路徑。代理伺服器收到位於指定路徑下的檔案請求時，會直接提供檔案而非將請求路由至您的應用程式。

如需使用組態檔案或 Elastic Beanstalk 主控台設定靜態檔案的詳細資訊，請參閱the section called “靜態檔案”。

環境屬性

Environment Properties (環境屬性) 的部分可讓您針對執行您應用程式的 Amazon EC2 執行個體，來指定其上的環境資訊設定。環境屬性會以金鑰值對的形式傳到應用程式。

在 Elastic Beanstalk 內所執行的 Java SE 環境中，可使用 `System.getenv()` 來存取環境變數。例如，您可使用下列程式碼，來將名為 `API_ENDPOINT` 的屬性讀取到變數：

```
String endpoint = System.getenv("API_ENDPOINT");
```

如需詳細資訊，請參閱環境屬性與其他軟體設定。

Java SE 組態命名空間

您可以使用組態檔案來設定組態選項，並在部署期間執行其他的執行個體設定工作。組態選項可透過 Elastic Beanstalk 服務或您使用的平台來定義，並且會組織成命名空間。

Java SE 平台不會定義任何平台特定的命名空間。您可以使用 `aws:elasticbeanstalk:environment:proxy:staticfiles` 命名空間設定代理提供靜態檔案。如需詳細資訊和範例，請參閱the section called “靜態檔案”。

Elastic Beanstalk 可提供許多組態選項讓您自訂環境。除了組態檔案，您也可以使用主控台、已儲存組態、EB CLI 或 AWS CLI 來設定組態選項。如需詳細資訊，請參閱組態選項。

Amazon Linux AMI (Amazon Linux 2 之前) Java SE 平台

如果您的 Elastic Beanstalk Java SE 環境使用 Amazon Linux AMI 平台版本 (Amazon Linux 2 之前)，請閱讀本節中的其他資訊。

> **備註**
>
> - 本主題中的資訊僅適用於以 Amazon Linux AMI (AL1) 為基礎的平台分支。AL2023/AL2 平台分支與舊版 Amazon Linux AMI (AL1) 平台版本不相容，需要不同的組態設定。
> - 2022 年 7 月 18 日，Elastic Beanstalk 已將所有以 Amazon Linux AMI (AL1) 為基礎的平台分支狀態設為已淘汰。如需有關遷移至完全支援的目前 Amazon Linux 2023 平台分支的詳細資訊，請參閱 將您的 Elastic Beanstalk Linux 應用程式遷移到 Amazon Linux 2023 或 Amazon Linux 2。

Amazon SE 組態命名空間 — Amazon Linux AMI (AL1)

您可以使用組態檔案來設定組態選項，並在部署期間執行其他的執行個體設定工作。組態選項可透過 Elastic Beanstalk 服務或您使用的平台來定義，並且會組織成命名空間。

除了所有平台皆支援的命名空間，Java SE 平台亦支援一個平台特定的組態命名空間。`aws:elasticbeanstalk:container:java:staticfiles` 命名空間可讓您定義選項，將您 Web 應用程式的路徑對應至應用程式原始碼套件中內含靜態內容的資料夾。

例如，此選項 option_settings 程式碼片段會定義靜態檔案命名空間內的兩個選項。第一個會將路徑 `/public` 對應至名為 `public` 的資料夾，第二個則將路徑 `/images` 對應至名為 `img` 的資料夾：

```
option_settings:
```

```
  aws:elasticbeanstalk:container:java:staticfiles:
    /html: statichtml
    /images: staticimages
```

您使用此命名空間對應的資料夾，必須為原始碼套件根目錄內的實際資料夾。您無法將路徑對應至 JAR 檔案中的資料夾。

Elastic Beanstalk 可提供許多組態選項讓您自訂環境。除了組態檔案，您也可以使用主控台、已儲存組態、EB CLI 或 AWS CLI 來設定組態選項。如需詳細資訊，請參閱組態選項。

透過 Buildfile 在伺服器上建置 JAR

您可於原始碼套件內的 `Buildfile` 檔案呼叫建置命令，藉此在您環境的 EC2 執行個體上，建置應用程式的類別檔案和 JAR。

`Buildfile` 中的命令僅能執行一次，且完成後必須終止，而 Procfile 的命令應於應用程式的使用壽命內持續執行，且被終止後會重新啟動。若要執行應用程式內的 JAR，請使用 `Procfile`。

如需有關 `Buildfile` 配置和語法的詳細資訊，請展開 the section called “擴充 Linux 平台” 中的 Buildfile 和 Procfile 區段。

下列 `Buildfile` 範例執行 Apache Maven，藉此從原始碼建置 Web 應用程式。如需使用此功能的範例應用程式的詳細資訊，請參閱 Java Web 應用程式範例。

Example Buildfile

```
build: mvn assembly:assembly -DdescriptorId=jar-with-dependencies
```

Java SE 平台包含下列建置工具，您可從建置指令碼進行呼叫：

- `javac` – Java 編譯器
- `ant` – Apache Ant
- `mvn` – Apache Maven
- `gradle` – Gradle

使用 Procfile 設定應用程式程序

若您應用程式原始碼套件的根目錄內含多個 JAR 檔案，請務必納入 `Procfile` 檔案，以指示 Elastic Beanstalk 應執行的 JAR。您亦可為單一 JAR 應用程式納入 `Procfile` 檔案，藉此設定執行您應用程式的 Java 虛擬機器 (JVM)。

建議務必在原始碼套件和應用程式中，一併提供 `Procfile`。如此一來，就能精確掌控 Elastic Beanstalk 會針對應用程式執行哪些程序，以及這些程序會收到哪些引數。

如需有關撰寫和使用 `Procfile` 的詳細資料，請展開the section called “擴充 Linux 平台”中的 Buildfile 和 Procfile 區段。

Example Procfile

```
web: java -Xms256m -jar server.jar
cache: java -jar mycache.jar
web_foo: java -jar other.jar
```

執行應用程式中主要 JAR 的命令必須稱為 `web`，且必須為 `Procfile` 中列出的第一條命令。nginx 伺服器會將所有接收自您環境負載平衡器的 HTTP 請求，轉送至此應用程式。

Elastic Beanstalk 假設 Procfile 的所有項目應隨時執行，並自動重新啟動 Procfile 中定義的所有已終止的應用程式。欲執行將終止且不重新啟動的命令，請使用 `Buildfile`。

在 Amazon Linux AMI (Amazon Linux 2 之前) 上使用 Procfile

如果您的 Elastic Beanstalk Java SE 環境使用 Amazon Linux AMI 平台版本 (Amazon Linux 2 之前)，請閱讀本節中的其他資訊。

> **備註**
>
> - 本主題中的資訊僅適用於以 Amazon Linux AMI (AL1) 為基礎的平台分支。AL2023/AL2 平台分支與舊版 Amazon Linux AMI (AL1) 平台版本不相容，需要不同的組態設定。
> - 2022 年 7 月 18 日，Elastic Beanstalk 已將所有以 Amazon Linux AMI (AL1) 為基礎的平台分支狀態設為已淘汰。如需有關遷移至完全支援的目前 Amazon Linux 2023 平台分支的詳細資訊，請參閱 將您的 Elastic Beanstalk Linux 應用程式遷移到 Amazon Linux 2023 或 Amazon Linux 2。

連接埠傳遞 — Amazon Linux AMI (AL1)

Elastic Beanstalk 預設會設定 nginx 代理將請求轉送至連接埠 5000 上的應用程式。您可將 `PORT` 環境屬性設定為主要應用程式接聽的連接埠，藉此覆寫預設連接埠。

如果您使用 `Procfile` 來執行多個應用程式，Amazon Linux AMI 平台版本上的 Elastic Beanstalk 預期每個額外的應用程式都須接聽編號比前一個多 100 的連接埠。Elastic Beanstalk 將每個

應用程式內部可存取的 PORT 變數，設定為其預期應用程式執行所用的連接埠。您可呼叫 `System.getenv("PORT")`，藉此於應用程式程式碼內存取此變數。

在上述 `Procfile` 範例中，`web` 應用程式會接聽連接埠 5000，`cache` 會接聽連接埠 5100，而 `web_foo` 會接聽連接埠 5200。`web` 會讀取 `PORT` 變數，藉此設定其接聽的連接埠，並於該連接埠編號加 100 以判定 `cache` 接聽的連接埠，從而向其傳送請求。

設定反向代理程式

Elastic Beanstalk 使用 nginx 做為反向代理伺服器，在連接埠 80 上將您的應用程式映射到 Elastic Load Balancing 負載平衡器。Elastic Beanstalk 提供了預設的 nginx 組態，您可以加以擴展，或使用自己的組態將其完全覆寫。

Elastic Beanstalk 預設會設定 nginx 代理將請求轉送至連接埠 5000 上的應用程式。您可將 `PORT` 環境屬性設定為主要應用程式接聽的連接埠，藉此覆寫預設連接埠。

> **Note**
>
> 您應用程式接聽的連接埠，不會影響 nginx 伺服器為接收來自負載平衡器的請求所接聽的連接埠。

在您的平台版本上設定代理伺服器

所有 AL2023/AL2 平台皆支援統一的代理組態功能。如需有關在執行 AL2023/AL2 的平台版本上設定代理伺服器的詳細資訊，請展開the section called “擴充 Linux 平台”中的反向代理組態一節。

在 Amazon Linux AMI (之前的 Amazon Linux 2) 上設定代理

如果您的 Elastic Beanstalk Java SE 環境使用 Amazon Linux AMI 平台版本 (Amazon Linux 2 之前)，請閱讀本節中的其他資訊。

> **備註**
>
> - 本主題中的資訊僅適用於以 Amazon Linux AMI (AL1) 為基礎的平台分支。AL2023/AL2 平台分支與舊版 Amazon Linux AMI (AL1) 平台版本不相容，需要不同的組態設定。
> - 2022 年 7 月 18 日，Elastic Beanstalk 已將所有以 Amazon Linux AMI (AL1) 為基礎的平台分支狀態設為已淘汰。如需有關遷移至完全支援的目前 Amazon Linux 2023 平台分支的

詳細資訊，請參閱 將您的 Elastic Beanstalk Linux 應用程式遷移到 Amazon Linux 2023 或 Amazon Linux 2。

擴展和覆寫預設的代理組態 — Amazon Linux AMI (AL1)

若要擴展 Elastic Beanstalk 的預設 nginx 組態，請將 `.conf` 組態檔案加進您應用程式原始碼套件中名為 `.ebextensions/nginx/conf.d/` 的資料夾。Elastic Beanstalk nginx 組態會自動在此資料夾中加入 `.conf` 檔案。

```
~/workspace/my-app/
|-- .ebextensions
|   `-- nginx
|       `-- conf.d
|           `-- myconf.conf
`-- web.jar
```

若要完全覆寫 Elastic Beanstalk 預設 nginx 組態，請在 `.ebextensions/nginx/nginx.conf` 的原始碼套件中加入組態：

```
~/workspace/my-app/
|-- .ebextensions
|   `-- nginx
|       `-- nginx.conf
`-- web.jar
```

如果您覆寫了 Elastic Beanstalk nginx 組態，請在 `nginx.conf` 中加入下列行，以納入適用於 增強型運作狀態報告與監控、自動應用程式映射和靜態檔案的 Elastic Beanstalk 組態。

```
 include conf.d/elasticbeanstalk/*.conf;
```

以下來自 Scorekeep 範例應用程式的範例組態會覆寫 Elastic Beanstalk 的預設組態，以提供 `public` 的 `/var/app/current` 子目錄的靜態 Web 應用程式，其中 Java SE 平台複製應用程式原始程式碼。`/api` 位置轉發流量以在 `/api/` 下路由傳送到連接埠 5000 上的 Spring 應用程式接聽。根路徑的 Web 應用程式提供其他所有流量。

Example

```
user                    nginx;
error_log               /var/log/nginx/error.log warn;
```

```
pid                     /var/run/nginx.pid;
worker_processes        auto;
worker_rlimit_nofile    33282;

events {
    worker_connections  1024;
}

http {
  include       /etc/nginx/mime.types;
  default_type  application/octet-stream;

  log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                    '$status $body_bytes_sent "$http_referer" '
                    '"$http_user_agent" "$http_x_forwarded_for"';

  include       conf.d/*.conf;

  map $http_upgrade $connection_upgrade {
      default     "upgrade";
  }

  server {
      listen        80 default_server;
      root /var/app/current/public;

      location / {
      }git pull


      location /api {
          proxy_pass          http://127.0.0.1:5000;
          proxy_http_version  1.1;

          proxy_set_header    Connection          $connection_upgrade;
          proxy_set_header    Upgrade             $http_upgrade;
          proxy_set_header    Host                $host;
          proxy_set_header    X-Real-IP           $remote_addr;
          proxy_set_header    X-Forwarded-For     $proxy_add_x_forwarded_for;
      }

      access_log    /var/log/nginx/access.log main;

      client_header_timeout 60;
```

```
        client_body_timeout   60;
        keepalive_timeout     60;
        gzip                  off;
        gzip_comp_level       4;

        # Include the Elastic Beanstalk generated locations
        include conf.d/elasticbeanstalk/01_static.conf;
        include conf.d/elasticbeanstalk/healthd.conf;
    }
}
```

將 Amazon RDS 資料庫執行個體新增到您的 Java 應用程式環境

您可以使用 Amazon Relational Database Service (Amazon RDS) 資料庫執行個體存放由應用程式收集與修改的資料。資料庫可連接到您的環境並由 Elastic Beanstalk 管理，或者在外部建立與管理。

若您首次使用 Amazon RDS，請透過 Elastic Beanstalk 主控台將資料庫執行個體新增至測試環境，並確認您的應用程式是否可與其連接。

欲將資料庫執行個體新增到您的環境

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

   > **Note**
   >
   > 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇 Configuration (組態)。
4. 在 Database (資料庫) 組態類別中，選擇 Edit (編輯)。
5. 選擇資料庫引擎，並輸入使用者名稱和密碼。
6. 若要儲存變更，請選擇頁面底部的儲存變更。

新增資料庫執行個體約需要 10 分鐘。環境更新完成時，資料庫執行個體的主機名稱和其他連線資訊會透過下列環境屬性提供給您的應用程式：

屬性名稱	描述	屬性值
RDS_HOSTNAME	資料庫執行個體的主機名稱。	在 Amazon RDS 主控台：端點的連線能力和安全性索引標籤上。
RDS_PORT	資料庫執行個體接受連線的連接埠。預設值在不同資料庫引擎中有所差異。	在 Amazon RDS 主控台：連接埠的連線能力和安全性索引標籤上。
RDS_DB_NAME	資料庫名稱，**ebdb**。	在 Amazon RDS 主控台：資料庫名稱的組態索引標籤上。
RDS_USERNAME	您為資料庫設定的使用者名稱。	在 Amazon RDS 主控台：主要使用者名稱的組態索引標籤上。
RDS_PASSWORD	您為資料庫設定的密碼。	無法在 Amazon RDS 主控台中提供參考。

如需設定內部資料庫執行個體的詳細資訊，請參閱將資料庫新增至您的 Elastic Beanstalk 環境。如需有關設定外部資料庫搭配 Elastic Beanstalk 使用的說明，請參閱搭配 Amazon RDS 使用 Elastic Beanstalk。

欲連接至資料庫，請於您的應用程式新增合適的驅動程式 JAR 檔案、下載您程式碼的驅動程式類別，並使用 Elastic Beanstalk 提供的環境屬性建立連線物件。

章節

下載 JDBC 驅動程式

您將需要所選資料庫引擎的 JDBC 驅動程式 JAR 檔案。將 JAR 檔案儲存至您的原始碼，並在編譯會建立資料庫連結的類別時，將其納入 classpath。

您可於下列位置找到資料庫引擎最新的驅動程式：

- MySQL – MySQL 連接器/J
- Oracle SE-1 – Oracle JDBC 驅動程式
- Postgres – PostgreSQL JDBC 驅動程式
- SQL Server – Microsoft JDBC 驅動程式

欲使用 JDBC 驅動程式，請呼叫 `Class.forName()` 來載入，之後再於程式碼中透過 `DriverManager.getConnection()` 建立連結。

JDBC 使用的連線字串格式如下：

```
jdbc:driver://hostname:port/dbName?user=userName&password=password
```

您可從 Elastic Beanstalk 提供您應用程式的環境變數，擷取主機名稱、連接埠、資料庫名稱、使用者名稱和密碼。驅動程式名稱為您的資料庫類型和驅動程式版本專屬。下列為範例驅動程式名稱：

- `mysql`適用於 MySQL 的
- `postgresql` (適用於 PostgreSQL)
- `oracle:thin` (適用於 Oracle Thin)
- `oracle:oci` (適用於 Oracle OCI)
- `oracle:oci8` (適用於 Oracle OCI 8)
- `oracle:kprb` (適用於 Oracle KPRB)
- `sqlserver` (適用於 SQL Server)

連線至資料庫 (Java SE 平台)

在 Java SE 環境中，請使用 `System.getenv()` 從環境讀取連線變數。下列範例程式碼為會建立 PostgreSQL 資料庫連線的類別。

```
private static Connection getRemoteConnection() {
    if (System.getenv("RDS_HOSTNAME") != null) {
      try {
      Class.forName("org.postgresql.Driver");
      String dbName = System.getenv("RDS_DB_NAME");
      String userName = System.getenv("RDS_USERNAME");
      String password = System.getenv("RDS_PASSWORD");
```

```
      String hostname = System.getenv("RDS_HOSTNAME");
      String port = System.getenv("RDS_PORT");
      String jdbcUrl = "jdbc:postgresql://" + hostname + ":" + port + "/" + dbName + "?
user=" + userName + "&password=" + password;
      logger.trace("Getting remote connection with connection string from environment
 variables.");
      Connection con = DriverManager.getConnection(jdbcUrl);
      logger.info("Remote connection successful.");
      return con;
    }
    catch (ClassNotFoundException e) { logger.warn(e.toString());}
    catch (SQLException e) { logger.warn(e.toString());}
    }
    return null;
  }
```

連線至資料庫 (Tomcat 平台)

在 Tomcat 環境中，環境屬性可做為透過 `System.getProperty()` 存取的系統屬性。

下列範例程式碼為會建立 PostgreSQL 資料庫連線的類別。

```
private static Connection getRemoteConnection() {
    if (System.getProperty("RDS_HOSTNAME") != null) {
      try {
      Class.forName("org.postgresql.Driver");
      String dbName = System.getProperty("RDS_DB_NAME");
      String userName = System.getProperty("RDS_USERNAME");
      String password = System.getProperty("RDS_PASSWORD");
      String hostname = System.getProperty("RDS_HOSTNAME");
      String port = System.getProperty("RDS_PORT");
      String jdbcUrl = "jdbc:postgresql://" + hostname + ":" + port + "/" + dbName + "?
user=" + userName + "&password=" + password;
      logger.trace("Getting remote connection with connection string from environment
 variables.");
      Connection con = DriverManager.getConnection(jdbcUrl);
      logger.info("Remote connection successful.");
      return con;
    }
    catch (ClassNotFoundException e) { logger.warn(e.toString());}
    catch (SQLException e) { logger.warn(e.toString());}
    }
    return null;
```

```
  }
```

若您建立連線或執行 SQL 陳述式出現問題，請嘗試將下列程式碼放入 JSP 檔案。此程式碼會連線至資料庫執行個體、建立資料表並寫入其中。

```
<%@ page import="java.sql.*" %>
<%
  // Read RDS connection information from the environment
  String dbName = System.getProperty("RDS_DB_NAME");
  String userName = System.getProperty("RDS_USERNAME");
  String password = System.getProperty("RDS_PASSWORD");
  String hostname = System.getProperty("RDS_HOSTNAME");
  String port = System.getProperty("RDS_PORT");
  String jdbcUrl = "jdbc:mysql://" + hostname + ":" +
    port + "/" + dbName + "?user=" + userName + "&password=" + password;

  // Load the JDBC driver
  try {
    System.out.println("Loading driver...");
    Class.forName("com.mysql.jdbc.Driver");
    System.out.println("Driver loaded!");
  } catch (ClassNotFoundException e) {
    throw new RuntimeException("Cannot find the driver in the classpath!", e);
  }

  Connection conn = null;
  Statement setupStatement = null;
  Statement readStatement = null;
  ResultSet resultSet = null;
  String results = "";
  int numresults = 0;
  String statement = null;

  try {
    // Create connection to RDS DB instance
    conn = DriverManager.getConnection(jdbcUrl);

    // Create a table and write two rows
    setupStatement = conn.createStatement();
    String createTable = "CREATE TABLE Beanstalk (Resource char(50));";
    String insertRow1 = "INSERT INTO Beanstalk (Resource) VALUES ('EC2 Instance');";
    String insertRow2 = "INSERT INTO Beanstalk (Resource) VALUES ('RDS Instance');";
```

```
    setupStatement.addBatch(createTable);
    setupStatement.addBatch(insertRow1);
    setupStatement.addBatch(insertRow2);
    setupStatement.executeBatch();
    setupStatement.close();

  } catch (SQLException ex) {
    // Handle any errors
    System.out.println("SQLException: " + ex.getMessage());
    System.out.println("SQLState: " + ex.getSQLState());
    System.out.println("VendorError: " + ex.getErrorCode());
  } finally {
    System.out.println("Closing the connection.");
    if (conn != null) try { conn.close(); } catch (SQLException ignore) {}
  }

  try {
    conn = DriverManager.getConnection(jdbcUrl);

    readStatement = conn.createStatement();
    resultSet = readStatement.executeQuery("SELECT Resource FROM Beanstalk;");

    resultSet.first();
    results = resultSet.getString("Resource");
    resultSet.next();
    results += ", " + resultSet.getString("Resource");

    resultSet.close();
    readStatement.close();
    conn.close();

  } catch (SQLException ex) {
    // Handle any errors
    System.out.println("SQLException: " + ex.getMessage());
    System.out.println("SQLState: " + ex.getSQLState());
    System.out.println("VendorError: " + ex.getErrorCode());
  } finally {
       System.out.println("Closing the connection.");
      if (conn != null) try { conn.close(); } catch (SQLException ignore) {}
  }
%>
```

欲顯示結果，請將下列程式碼放入 JSP 檔案 HTML 部分的本文中。

```
<p>Established connection to RDS. Read first two rows: <%= results %></p>
```

資料庫連線故障診斷

若您從應用程式建立資料庫連線出現問題，請檢視 Web 容器日誌和資料庫。

檢視日誌

您可於 Eclipse 內檢視 Elastic Beanstalk 環境的所有日誌。若您沒看到 AWS Explorer 檢視視窗，請於工具列選擇橘色 AWS 圖示旁的箭頭，然後選擇 Show AWS Explorer View (顯示 AWS Explorer 視圖)。展開 AWS Elastic Beanstalk 以及您的環境名稱，然後開啟伺服器的內容選單 (按一下滑鼠右鍵)。選擇 Open in WTP Server Editor (於 WTP 伺服器編輯器開啟)。

選擇 Server (伺服器) 檢視的 Log (日誌) 索引標籤，以查看您環境的彙整日誌。欲開啟最新的日誌，選擇頁面右上角的 Refresh (重新整理) 按鈕。

向下捲動在 `/var/log/tomcat7/catalina.out` 中找到 Tomcat 日誌。若您在之前的數個範例已載入網頁，可能會看到下列內容。

```
-------------------------------------
/var/log/tomcat7/catalina.out
-------------------------------------
INFO: Server startup in 9285 ms
Loading driver...
Driver loaded!
SQLException: Table 'Beanstalk' already exists
SQLState: 42S01
VendorError: 1050
Closing the connection.
Closing the connection.
```

Web 應用程式傳送至標準輸出的所有資訊會顯示於 Web 容器日誌。在上述範例中，應用程式在每次載入頁面時都嘗試建立資料表。這導致第一次載入頁面後，每次載入頁面都會發現 SQL 例外狀況。

以範例而言，上述為可接受情況。但實際在應用程式中，請於結構描述物件中保存資料庫定義，在模型類別中執行交易，並與控制器 Servlet 協調請求。

連接到 RDS 資料庫執行個體

您可使用 MySQL 用戶端應用程式，直接連接至 Elastic Beanstalk 環境中的 RDS 資料庫執行個體。

首先，請開放對 RDS 資料庫執行個體的安全群組，以允許來自您電腦的流量。

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇 Configuration (組態)。
4. 在 Database (資料庫) 組態類別中，選擇 Edit (編輯)。
5. 在 Endpoint (端點) 旁，選擇 Amazon RDS 主控台連結。
6. 在 RDS Dashboard (RDS 儀表板) 執行個體詳細資訊頁面中的 Security and Network (安全與網路) 下，選擇 Security Groups (安全群組) 旁開頭為 rds- 的安全群組。

> **Note**
>
> 資料庫可能有多個項目均標示為 Security Groups (安全群組)。僅在您較舊的帳戶沒有預設 Amazon Virtual Private Cloud (Amazon VPC) 的情況下，才使用第一個項目 (開頭為 awseb)。

7. 在 Security group details (安全群組詳細資訊) 中，選擇 Inbound (傳入) 索引標籤，然後選擇 Edit (編輯)。
8. 新增 MySQL (連接埠 3306) 的規則，以允許來自您以 CIDR 格式指定的 IP 地址的流量。
9. 選擇 Save (儲存)。變更會立即生效。

返回您環境的 Elastic Beanstalk 組態詳細資訊，並記下端點。您將使用該網域名稱連接至 RDS 資料庫執行個體。

安裝 MySQL 用戶端並於連接埠 3306 啟動資料庫的連線。在 Windows 上，請自 MySQL 首頁安裝 MySQL Workbench，並依提示操作。

在 Linux 上，請使用適用您的分發的套件管理工具，來安裝 MySQL 用戶端。下列範例運作於 Ubuntu 和其他 Debian 的衍生產品。

```
// Install MySQL client
$ sudo apt-get install mysql-client-5.5
...
```

```
// Connect to database
$ mysql -h aas839jo2vwhwb.cnubrrfwfka8.us-west-2.rds.amazonaws.com -u username -
ppassword ebdb
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 117
Server version: 5.5.40-log Source distribution
...
```

連線之後，您可執行 SQL 命令來查看資料庫的狀態，了解資料表和列是否已建立和其他資訊。

```
mysql> SELECT Resource from Beanstalk;
+--------------+
| Resource     |
+--------------+
| EC2 Instance |
| RDS Instance |
+--------------+
2 rows in set (0.01 sec)
```

使用 AWS Toolkit for Eclipse

AWS Toolkit for Eclipse 將 AWS Elastic Beanstalk 管理功能整合至您的 Tomcat 開發環境，藉此促進環境建立、設定和程式碼部署。本工具組擁有多個 AWS 帳戶的支援，可管理現有環境，並於環境內直接連接至執行個體以進行故障診斷。

> **Note**
>
> AWS Toolkit for Eclipse 僅支援使用 Java 搭配 Tomcat 平台的專案，不支援搭配 Java SE 平台的專案。

如需有關事前準備和安裝 AWS Toolkit for Eclipse 的詳細資訊，請前往 https://aws.amazon.com/eclipse。您亦可查看搭配 AWS Toolkit for Eclipse 使用 AWS Elastic Beanstalk 的影片。本主題也提供實用資訊，內容涵蓋工具、如何使用主題和其他 Java 開發人員可使用的資源。

將現有環境匯入 Eclipse

您可將於 AWS 管理主控台建立的現有環境匯入 Eclipse。

若要匯入現有環境，請展開 AWS Elastic Beanstalk 節點，並於 Eclipse 按兩下 AWS Explorer 內的環境。您現在可以將 Elastic Beanstalk 應用程式部署到此環境。

管理 Elastic Beanstalk 應用程式環境

主題

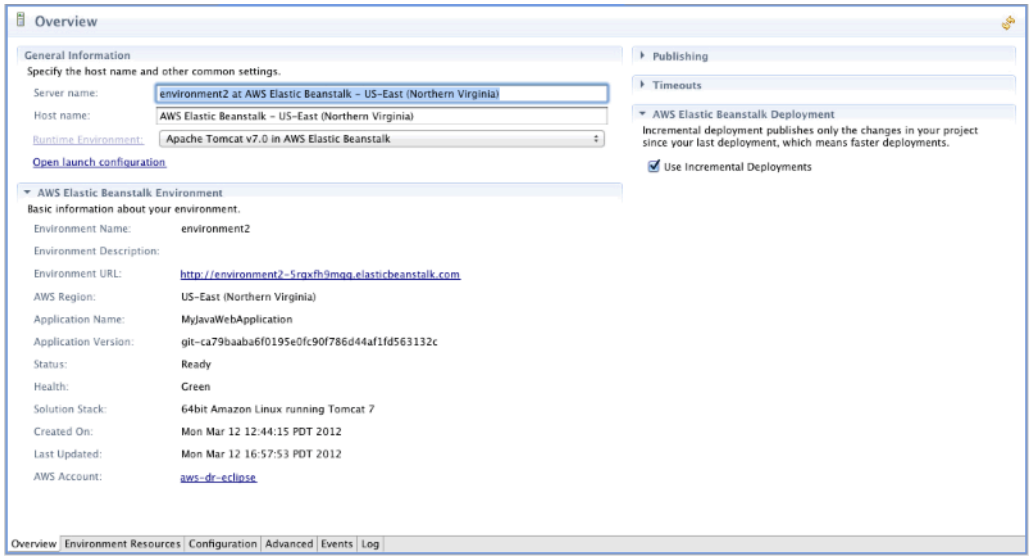
透過 AWS Toolkit for Eclipse，您可變更您應用程式環境所用之 AWS 資源的佈建與組態。如需有關如何使用 AWS 管理主控台來管理您的應用程式環境的資訊，請參閱 管理環境。本節討論您可於 AWS Toolkit for Eclipse 編輯的特定服務設定，做為應用程式環境資訊的一部分。如需有關 AWS Toolkit for Eclipse 的詳細資訊，請參閱《AWS Toolkit for Eclipse 入門指南》。

變更環境資訊設定

部署應用程式時，Elastic Beanstalk 會設定多種 AWS 雲端運算服務。您可使用 AWS Toolkit for Eclipse 來控制如何設定這些個別服務。

欲編輯應用程式的環境設定

1. 若 Eclipse 未顯示 AWS Explorer 視圖，請於選單中選擇 Window (視窗)、Show View (顯示視圖) 和 AWS Explorer。展開 Elastic Beanstalk 節點和您的應用程式節點。
2. 在 AWS Explorer 中，按兩下 Elastic Beanstalk 環境。
3. 在窗格底部按一下 Configuration (組態) 索引標籤。

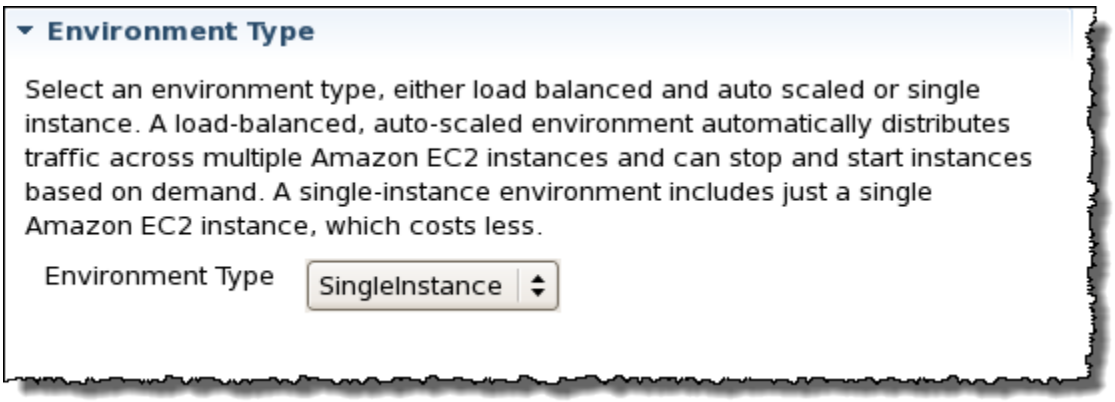


您現可進行下列設定：

- EC2 伺服器執行個體
- 負載平衡器
- 自動擴展
- 通知
- 環境類型
- 環境屬性

變更環境類型

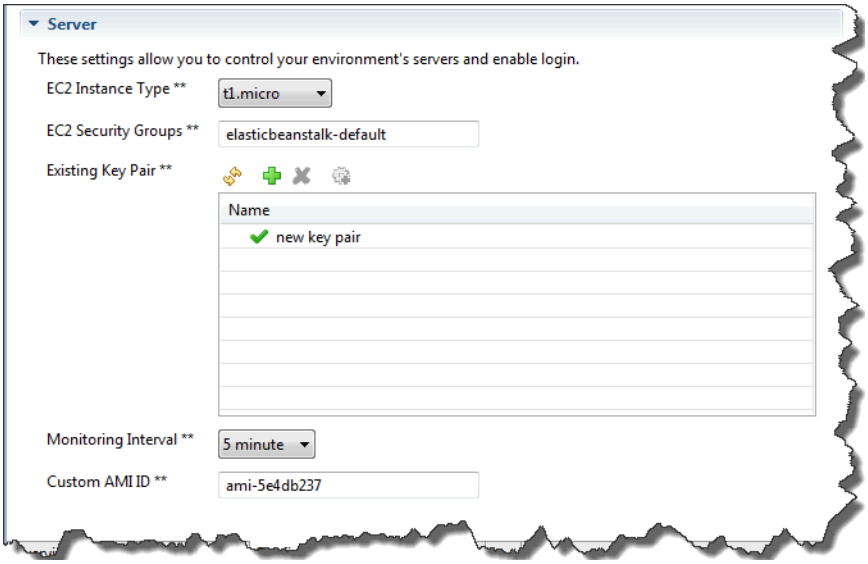
在 AWS Toolkit for Eclipse 中，Configuration (組態) 標籤的 Environment Type (環境類型) 區段可讓您根據所部署的應用程式之要求，選取 Load balanced, auto scaled (負載平衡、自動調整規模) 或 Single instance (單一執行個體) 環境。以需要擴展性的應用程式而言，選取 Load balanced, auto scaled (負載平衡、自動調整規模)。針對簡單、低流量的應用程式，則選取 Single instance (單一執行個體)。如需更多詳細資訊，請參閱 環境類型。



使用 來設定 EC2 伺服器執行個體AWS Toolkit for Eclipse

Amazon Elastic Compute Cloud (EC2) 為 Web 服務，可啟動並管理 Amazon 資料中心的伺服器執行個體。您可隨時使用 Amazon EC2 伺服器執行個體，無時間限制，且任何法律用途皆可。執行個體具備不同的大小與組態。如需詳細資訊，請前往 Amazon EC2 產品頁面。

在 Server (伺服器) 中，您可於環境 Configuration (組態) 索引標籤上的 Toolkit for Eclipse 底下，編輯 Elastic Beanstalk 環境的 Amazon EC2 執行個體組態。



Amazon EC2 執行個體類型

Instance type (執行個體類型) 顯示您 Elastic Beanstalk 應用程式可用的執行個體類型。請變更執行個體類型，以選取具有最適合您應用程式之特性 (包含記憶體大小和 CPU 能力) 的伺服器。例如，需要進行大量及長時間操作的應用程式可要求更多 CPU 或記憶體。

如需有關您 Elastic Beanstalk 應用程式可用之 Amazon EC2 執行個體類型的詳細資訊，請參閱《Amazon Elastic Compute Cloud 使用者指南》中的執行個體類型。

Amazon EC2 安全群組

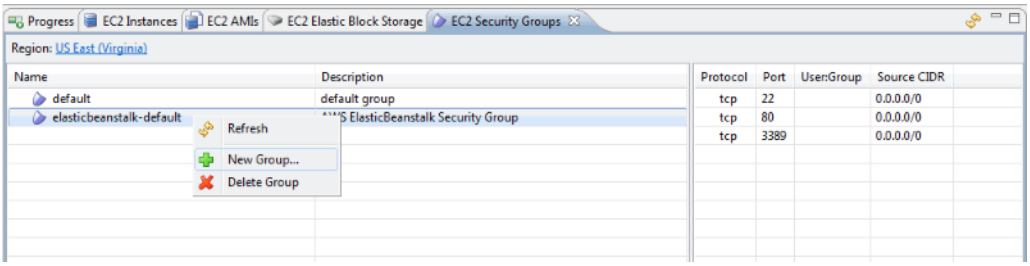
您可使用「Amazon EC2 安全群組」來控制 Elastic Beanstalk 應用程式的存取。安全群組會定義您的執行個體的防火牆規則。這些規則會指定哪些輸入 (即傳入) 網路流量應傳送至您的執行個體。所有其他傳入流量將被捨棄。您可隨時修改群組的規則。新規則會自動於所有執行中的執行個體以及未來啟動的執行個體上實施。

您可使用 AWS 管理主控台或 AWS Toolkit for Eclipse 來設定 Amazon EC2 安全群組。在 EC2 Security Groups (EC2 安全群組) 方塊中輸入一個或多個 Amazon EC2 安全群組的名稱 (以逗號分隔)，您可藉此指定哪些 Amazon EC2 安全群組可控制您 Elastic Beanstalk 應用程式的存取。

> **Note**
>
> 若您使用舊式容器類型執行應用程式，且欲啟用應用程式的運作狀態檢查，請確認可自來源 CIDR 範圍 0.0.0.0/0 存取連接埠 80 (HTTP)。如需運作狀態檢查的詳細資訊，請參閱 運作狀態檢查。欲檢查您是否正使用舊版容器類型，請參閱 the section called “為何部分平台版本標記為舊版？”

欲使用 AWS Toolkit for Eclipse 建立安全群組

1. 在 AWS Toolkit for Eclipse 中，按一下 AWS Explorer 標籤。展開 Amazon EC2 節點，然後按兩下 Security Groups (安全群組)。
2. 在左側表格任一處按一下滑鼠右鍵，然後按一下 New Group (新群組)。



3. 在 Security Group (安全群組) 對話方塊中，輸入安全群組名稱及描述，然後按一下 OK (確定)。

如需 Amazon EC2 安全群組的詳細資訊，請參閱Amazon Elastic Compute Cloud 使用者指南中的使用安全群組相關文章。

Amazon EC2 金鑰對

透過 Amazon EC2 金鑰對，您可安全登入為 Elastic Beanstalk 應用程式佈建的 Amazon EC2 執行個體。

> **Important**
>
> 您必須建立 Amazon EC2 金鑰對並設定 Elastic Beanstalk 佈建的 Amazon EC2 執行個體，以使用 Amazon EC2 金鑰對，才能存取 Elastic Beanstalk 佈建的 Amazon EC2 執行個體。將應用程式部署至 Elastic Beanstalk 時，您可使用 AWS Toolkit for Eclipse 內的 Publish to Beanstalk Wizard (發佈至 Beanstalk 精靈) 來建立金鑰對。或者，您可以使用 AWS 管理主控台，設定您的 Amazon EC2 金鑰對。如需建立 Amazon EC2 金鑰對的詳細資訊，請參閱 Amazon Elastic Compute Cloud 入門指南。

如需 Amazon EC2 金鑰對的詳細資訊，請前往Amazon Elastic Compute Cloud 使用者指南中的使用 Amazon EC2 登入資料相關文章。如需連接至 Amazon EC2 執行個體的詳細資訊，請參閱 Amazon Elastic Compute Cloud 使用者指南中的連接至執行個體及使用 PuTTY 從 Windows 連接至您的 Linux/UNIX 執行個體相關文章。

CloudWatch 指標

根據預設，只會啟用基本 Amazon CloudWatch 指標。其會每五分鐘傳回資料一次。您可以藉由為 AWS Toolkit for Eclipse 中您環境之 Configuration (組態) 標籤的 Server (伺服器) 區段中 Monitoring Interval (監控間隔)，選取 1 minute (1 分鐘)，來啟用以每一分鐘為一個間隔之更精密的 CloudWatch 指標。

> **Note**
>
> Amazon CloudWatch 服務費用可適用於一分鐘間隔指標。如需詳細資訊，請參閱 Amazon CloudWatch。

自訂 AMI ID

您可以藉由將您自訂 AMI 的識別符輸入 AWS Toolkit for Eclipse 中您環境 Configuration (組態) 標籤之 Server (伺服器) 區段中的 Custom AMI ID (自訂 AMI ID) 方塊，來使用您的自訂 AMI 覆寫您 Amazon EC2 執行個體的預設 AMI。

> **Important**
>
> 使用您自己的 AMI 為進階任務，需要謹慎操作。若您需要自訂 AMI，建議您從預設的 Elastic Beanstalk AMI 開始並加以修改。為使運作狀態良好，Elastic Beanstalk 預期 Amazon EC2 執

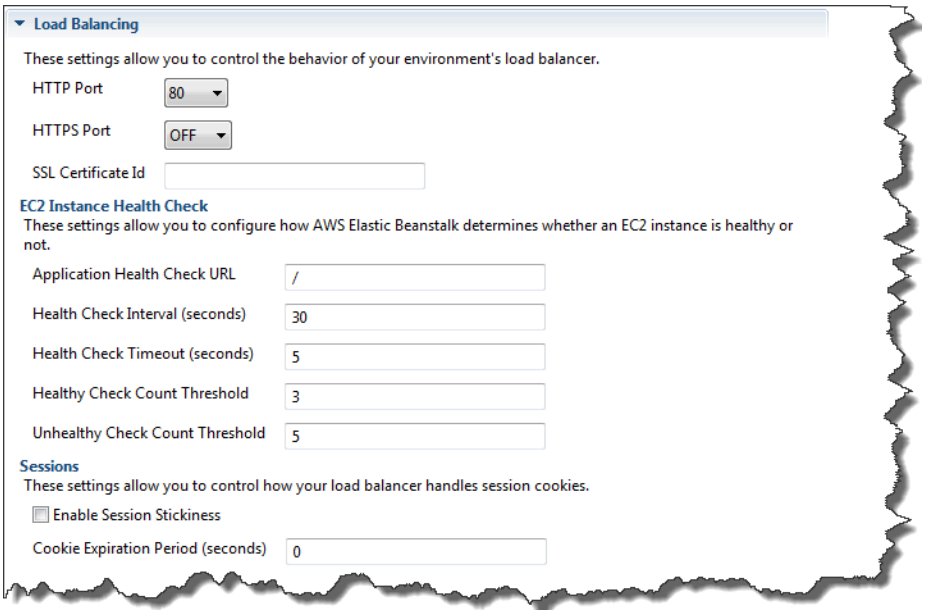
行個體應滿足一組要求，包含具有執行中的主機管理員。若未符合這些要求，您的環境可能無法正常運作。

使用 來設定 Elastic Load BalancingAWS Toolkit for Eclipse

Elastic Load Balancing 是 Amazon Web 服務，可提升應用程式的可用性和可擴展性。透過 Elastic Load Balancing，您可將應用程式負載分配至兩個或多個 Amazon EC2 執行個體。Elastic Load Balancing 可透過備援改善可用性，亦可支援應用程式的流量成長。

Elastic Load Balancing 會自動分配傳入應用程式的流量，在您執行的所有 EC2 伺服器執行個體間取得平衡。當您需要增加應用程式的容量時，本服務亦讓您輕鬆新增新的執行個體。

部署應用程式時，Elastic Beanstalk 會自動佈建 Elastic Load Balancing。在 Load Balancing (負載平衡) 下，Toolkit for Eclipse 內您環境的 Configuration (組態) 索引標籤上，您可以編輯 Elastic Beanstalk 環境的負載平衡組態。



下列章節說明您可設定的 Elastic Load Balancing 參數供應用程式使用。

連接埠

為了為您的 Elastic Beanstalk 應用程式處理請求所佈建的負載平衡器，會將請求傳送至執行您應用程式的 Amazon EC2 執行個體。所佈建的負載平衡器可於 HTTP 和 HTTPS 連接埠接聽請求，並將請求路由至 AWS Elastic Beanstalk 應用程式內的 Amazon EC2 執行個體。負載平衡器預設會處理 HTTP 連接埠上的請求。必須開啟至少一個 HTTP 或 HTTPS 連接埠。

> **Important**
>
> 請確認您指定的連接埠並未鎖定；否則，使用者將無法連接至您的 Elastic Beanstalk 應用程式。

控制 HTTP 連接埠

欲關閉 HTTP 連接埠，請於 HTTP Listener Port (HTTP 接聽程式連接埠) 選取 OFF (關閉)。欲開啟 HTTP 連接埠，請於選取 HTTP 連接埠 (如 80 (80))。

> **Note**
>
> 若要使用預設連接埠 80 以外 (例如連接埠 8080) 的連接埠存取您的環境，請將接聽程式新增至現有的負載平衡器，然後設定該新的接聽程式在該連接埠上接聽。
> 例如，使用適用於 Classic Load Balancer 的 AWS CLI，輸入如下命令，將 *LOAD_BALANCER_NAME* 取代為您 Elastic Beanstalk 負載平衡器的名稱。
>
> ```
> aws elb create-load-balancer-listeners --load-balancer-name LOAD_BALANCER_NAME
>  --listeners "Protocol=HTTP, LoadBalancerPort=8080, InstanceProtocol=HTTP,
>  InstancePort=80"
> ```
>
> 例如，使用適用於 Application Load Balancer 的 AWS CLI，輸入如下命令，將 *LOAD_BALANCER_ARN* 取代為您 Elastic Beanstalk 負載平衡器的 ARN。
>
> ```
> aws elbv2 create-listener --load-balancer-arn LOAD_BALANCER_ARN --protocol HTTP
>  --port 8080
> ```
>
> 若您想要 Elastic Beanstalk 監控您的環境，請勿移除連接埠 80 上的接聽程式。

控制 HTTPS 連接埠

Elastic Load Balancing 支援 HTTPS/TLS 通訊協定，可加密用戶端連線至負載平衡器的流量。負載平衡器至 EC2 執行個體的連線採用純文字。HTTPS 連接埠預設為關閉。

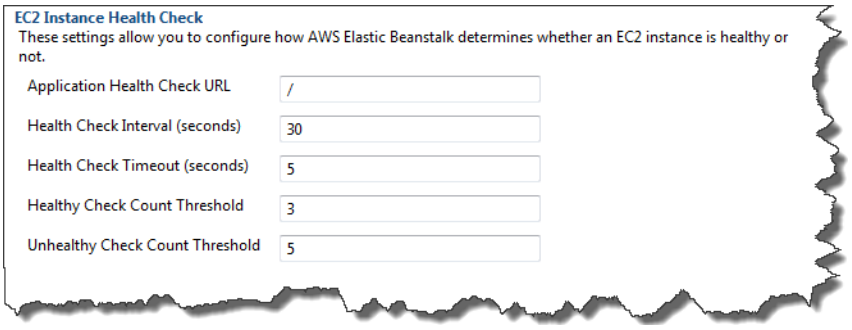
欲開啟 HTTPS 連接埠

1. 使用 AWS Certificate Manager (ACM) 建立新的憑證或將憑證和金鑰上傳至 AWS Identity and Access Management (IAM)。如需請求 ACM 憑證的詳細資訊，請參閱《AWS Certificate Manager 使用者指南》中的請求憑證。如需有關將第三方憑證匯入 ACM 的詳細資訊，請參閱《AWS Certificate Manager 使用者指南》中的匯入憑證。如果 ACM 無法在您的 AWS 區域中使用，請使用 AWS Identity and Access Management (IAM) 上傳第三方憑證。ACM 和 IAM 服務會存放憑證，並針對 SSL 憑證提供 Amazon Resource Name (ARN)。如需建立和上傳憑證至 IAM 的詳細資訊，請參閱IAM 使用者指南中的使用伺服器憑證。
2. 從 HTTPS Listener Port (HTTPS 接聽程式的連接埠) 下拉式清單中選擇連接埠，以指定 HTTPS 連接埠。
3. 在 SSL Certificate ID (SSL 憑證 ID) 文字方塊中，輸入您 SSL 憑證的 Amazon Resources Name (ARN)，例如，**`arn:aws:iam::123456789012:server-certificate/abc/certs/build`** 或 **`arn:aws:acm:us-east-2:123456789012:certificate/12345678-12ab-34cd-56ef-12345678`**。使用您於步驟 1 建立並上傳的 SSL 憑證。

欲關閉 HTTPS 連接埠，請於 HTTPS Listener Port (HTTPS 接聽程式連接埠) 選取 OFF (關閉)。

運作狀態檢查

您可於 Load Balancing (負載平衡) 面板使用 EC2 Instance Health Check (EC2 執行個體運作狀態檢查) 區段，藉此控制運作狀態檢查的設定。



下列清單說明您應用程式可設定的運作狀態檢查參數。

- 為了判定執行個體的運作狀態，Elastic Beanstalk 會在其查詢的 URL 上尋找 200 回應代碼。根據預設，Elastic Beanstalk 針對非舊式容器和舊式容器分別檢查 TCP:80 和 HTTP:80。您可於 Application Health Check URL (應用程式運作狀態檢查 URL) 方塊中輸入與應用程式現有資源相符的 URL (如 **`/myapp/index.jsp`**)，藉此覆寫預設 URL。若您覆寫預設 URL，Elastic Beanstalk 會使用 HTTP 來查詢資源。欲檢查您是否正使用舊版容器類型，請參閱 the section called “為何部分平台版本標記為舊版？”
- 在 Health Check Interval (seconds) (運作狀態檢查間隔，秒) 部分，輸入應用程式 Amazon EC2 執行個體每次運作狀態檢查間的秒數。
- 在 Health Check Timeout (運作狀態檢查逾時) 部分，指定 Elastic Load Balancing 將執行個體視為沒有回應前的等待回應的秒數。
- 使用 Healthy Check Count Threshold (運作狀態檢查計數閾值) 和 Unhealthy Check Count Threshold (不健全檢查結果計數臨界值) 方塊，來指定 Elastic Load Balancing 變更執行個體運作狀態前的連續成功或不成功 URL 探測的次數。例如，在 Unhealthy Check Count Threshold (不健全檢查結果計數臨界值) 文字方塊指定 5，表示 URL 連續五次回傳錯誤訊息或逾時後，Elastic Load Balancing 會將運作狀態檢查視為「失敗」。

工作階段

負載平衡器預設會以最小的負載，將每個請求獨立路由至伺服器執行個體。相對而言，黏性工作階段會將使用者工作階段繫結到特定的伺服器執行個體，以便工作階段期間來自使用者的所有請求都發送到相同的伺服器執行個體。

應用程式若啟用黏性工作階段，Elastic Beanstalk 會使用負載平衡器產生的 HTTP Cookie。負載平衡器會使用特殊負載平衡器產生的 Cookie，來追蹤每個請求的應用程式執行個體。當負載平衡器收到請求時，首先會檢查此 Cookie 是否存在於請求中。若是，此請求會傳送至 Cookie 中指定的應用程式執行個體。若 Cookie 不存在，則負載平衡器會根據現有負載平衡演算法選擇應用程式執行個體。回應會插入 Cookie，藉此將後續來自相同使用者的請求繫結至該應用程式執行個體。政策組態會定義 Cookie 到期日期，此為每個 Cookie 的有效使用期限。

在 Sessions (工作階段) 區段中的 Load Balancer (負載平衡器) 底下，請指定應用程式的負載平衡器是否允許讓工作階段黏著，以及每一 Cookie 的持續時間。

如需 Elastic Load Balancing 的詳細資訊，請參閱 Elastic Load Balancing 開發人員指南。

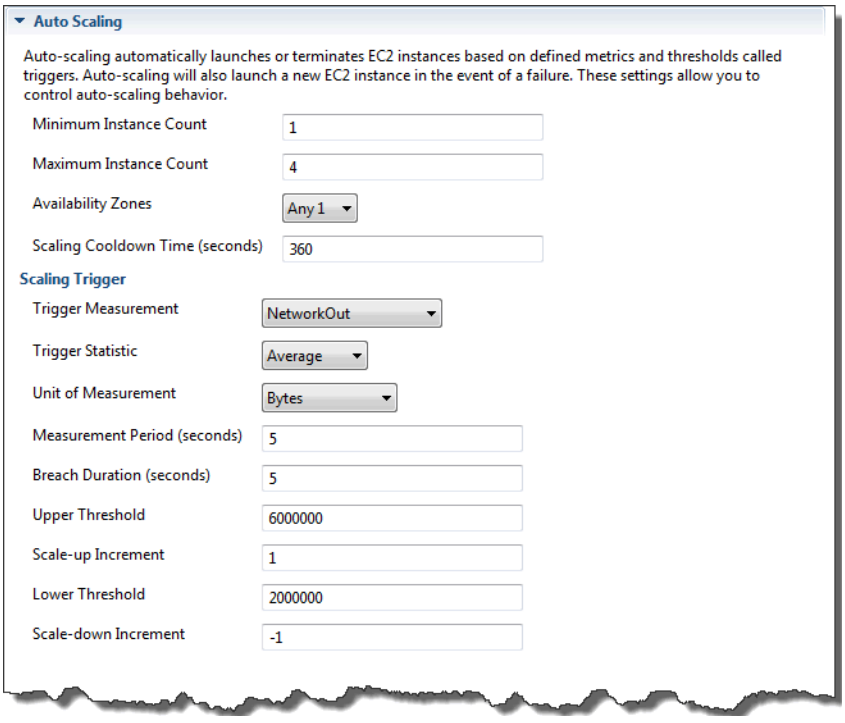
使用 來設定 Auto ScalingAWS Toolkit for Eclipse

Amazon EC2 Auto Scaling 為 Amazon Web Service，旨在根據使用者定義的觸發來自動啟動或終止 Amazon EC2 執行個體。使用者可設定 Auto Scaling 群組並將「觸發」與這些群組建立關聯，藉此根據諸如頻寬使用量或 CPU 使用率等指標，自動擴展運算資源。Amazon EC2 Auto Scaling 可與 Amazon CloudWatch 搭配使用，以擷取執行應用程式之伺服器執行個體的指標。

Amazon EC2 Auto Scaling 可讓您取得一組 Amazon EC2 執行個體，並設定各種參數，讓此群組自動增減數量。Amazon EC2 Auto Scaling 可於該群組新增或移除 Amazon EC2 執行個體，協助您無縫處理應用程式的流量變更。

Amazon EC2 Auto Scaling 亦會針對其啟動的每個 Amazon EC2 執行個體，監控其運作狀態。如果有任何執行個體未預期終止，Amazon EC2 Auto Scaling 會偵測到終止狀況，並啟動替代執行個體。此功能可讓您自動維持所需的固定 Amazon EC2 執行個體數量。

Elastic Beanstalk 會為您的應用程式佈建 Amazon EC2 Auto Scaling。在 Auto Scaling (自動縮放) 下，您可於環境 Configuration (組態) 索引標籤的 Toolkit for Eclipse 底下，編輯 Elastic Beanstalk 環境的 Auto Scaling 組態。



下列章節討論如何設定 Auto Scaling 參數供您的應用程式使用。

啟動組態

您可編輯啟動組態，以控制 Elastic Beanstalk 應用程式佈建 Amazon EC2 Auto Scaling 資源的方式。

使用 Minimum Instance Count (執行個體計數下限) 與 Maximum Instance Count (執行個體計數上限) 設定，來指定您 Elastic Beanstalk 應用程式使用的 Auto Scaling 群組大小上下限。

> **Note**
>
> 若要保持固定數量的 Amazon EC2 執行個體，請將 Minimum Instance Count (執行個體計數下限) 和 Maximum Instance Count (執行個體計數上限) 的文字方塊設為相同值。

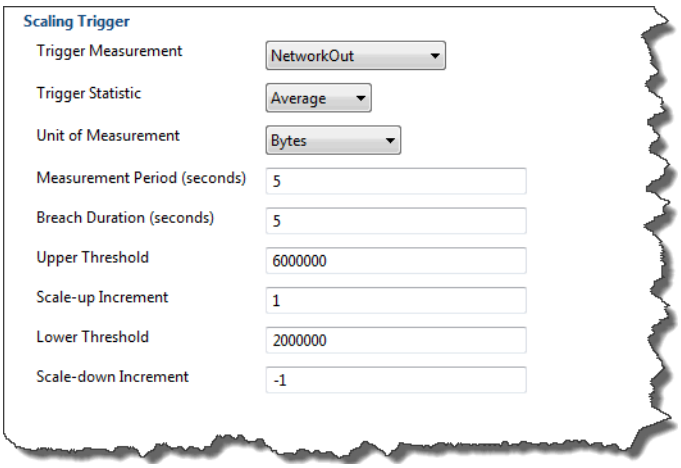
Availability Zones (可用區域) 可讓您指定 Amazon EC2 執行個體所在可用區域數量。若您希望建立容錯應用程式，請務必設定此數量：若單一可用區域故障，其他可用區域的執行個體仍會繼續運作。

> **Note**
>
> 目前，您無法指定執行個體所在的可用區域。

觸發

觸發是您可加以設定的 Amazon EC2 Auto Scaling 機制，以通知系統您想要增加 (擴展) 或減少 (縮減) 執行個體數量的時機。您可設定觸發，在指標 (如 CPU 使用率) 發佈至 Amazon CloudWatch 時觸發，並判斷是否滿足您指定的條件。在指定期間內，若達到您的指標閾值上下限，則觸發會啟動名為擴展活動的長時間執行程序。

您可使用 AWS Toolkit for Eclipse，為 Elastic Beanstalk 應用程式定義擴展觸發。



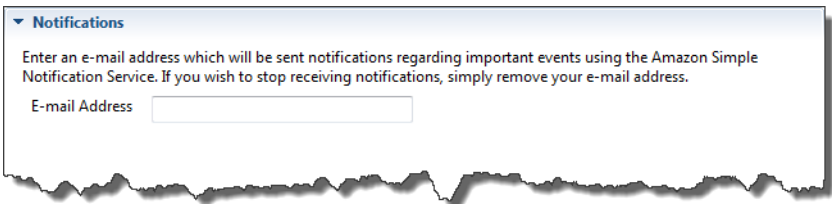
在 Toolkit for Eclipse 的您的環境中，您可於 Configuration (組態) 標籤的 Scaling Trigger (擴展觸發條件) 區段，設定下列觸發參數清單。

- Trigger Measurement (觸發條件測量指標) 可指定您的觸發指標。
- Trigger Statistic (觸發條件統計資料) 可指定觸發條件將使用的統計資料—**`Minimum`**、**`Maximum`**、**`Sum`** 或 **`Average`**。
- Unit of Measurement (測量單位) 可指定觸發條件測量的單位。
- Measurement Period (測量期間) 可指定 Amazon CloudWatch 測量觸發指標的頻率。Breach Duration (違規持續時間) 可定義指標在超過所定義的限制 (即 Upper Threshold (閾值上限) 和 Lower Threshold (閾值下限) 所指定) 後，引發觸發前的時間。
- Scale-up Increment (規模調整增幅) 和 Scale-down Increment (規模調整減幅) 可指定在進行擴展活動時，若要新增或移除的 Amazon EC2 執行個體數量。

如需 Amazon EC2 Auto Scaling 的詳細資訊，請參閱 Amazon Elastic Compute Cloud 文件中的「Amazon EC2 Auto Scaling」一節。

使用 來設定通知AWS Toolkit for Eclipse

Elastic Beanstalk 使用 Amazon Simple Notification Service (Amazon SNS) 來通知您影響應用程式的重要事件。欲啟用 Amazon SNS 通知，只要在 Toolkit for Eclipse 中您環境 Configuration (組態) 標籤上 Notifications (通知) 底下的 Email Address (電子郵件地址) 文字方塊輸入您的電子郵件地址。欲停用 Amazon SNS 通知，請將您的電子郵件地址自文字方塊移除。



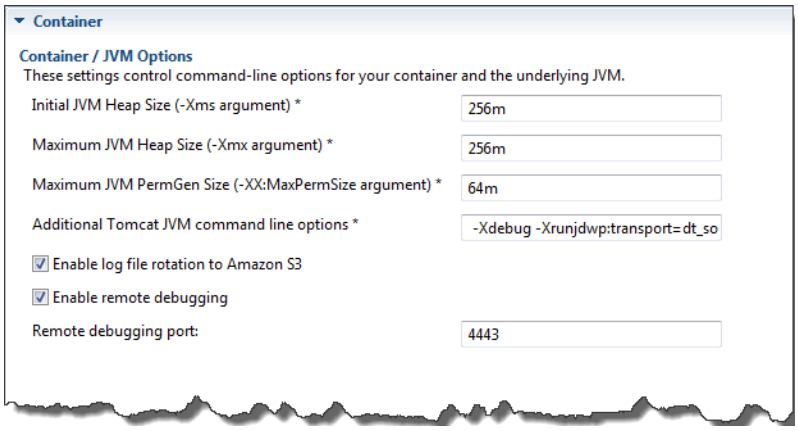
使用 來設定 Java 容器AWS Toolkit for Eclipse

Container/JVM Options (Container/JVM 選項) 面板可讓您微調 Amazon EC2 執行個體上 Java 虛擬機器的行為，也可啟用或停用 Amazon S3 日誌輪換。您可使用 AWS Toolkit for Eclipse 來設定您的容器資訊。如需 Tomcat 環境可用選項的詳細資訊，請參閱 the section called “設定您的 Tomcat 環境”。

> **Note**
>
> 您可交換環境 CNAME，藉此零停機修改您的組態設定。如需更多詳細資訊，請參閱 透過 Elastic Beanstalk 進行藍/綠部署。

存取 Elastic Beanstalk 應用程式的 Container/JVM Options (容器/JVM) 選項面板

1. 若 Eclipse 未顯示 AWS Explorer 視圖，請於選單中選擇 Window (視窗)、Show View (顯示視圖) 和 AWS Explorer。展開 Elastic Beanstalk 節點和您的應用程式節點。
2. 在 AWS Explorer 中，按兩下 Elastic Beanstalk 環境。
3. 在窗格底部按一下 Configuration (組態) 索引標籤。
4. 在 Container (容器) 底下，您可設定容器選項。



遠端除錯

欲遠端測試您的應用程式，您可以除錯模式執行應用程式。

欲啟用遠端除錯

1. 選取 Enable remote debugging (啟用遠端除錯)。
2. Remote debugging port (遠端除錯連接埠) 可指定用於遠端除錯的連接埠編號。

   將自動填入 Additional Tomcat JVM command line options (其他 Tomcat JVM 命令列選項) 設定。

欲開始遠端除錯

1. 在 AWS Toolkit for Eclipse 選單中，選擇 Window (視窗) > Show View (顯示畫面) > Other (其他)。
2. 展開 Server (伺服器) 資料夾，然後選擇 Servers (伺服器)。選擇 OK (確定)。
3. 在 Servers (伺服器) 窗格中，以滑鼠右鍵按一下您應用程式在其上執行的伺服器，然後按一下 Restart in Debug (以除錯模式重新啟動)。

透過 AWS Toolkit for Eclipse 設定系統屬性

下列範例在 AWS Toolkit for Eclipse 中設定 `JDBC_CONNECTION_STRING` 系統屬性。設定此屬性後，即可供您的 Elastic Beanstalk 應用程式做為系統屬性 `JDBC_CONNECTION_STRING`。

> **Note**
>
> AWS Toolkit for Eclipse 尚未支援修改 VPC 內環境的環境資訊 (包括系統屬性)。除非您具有使用 EC2 Classic 的舊帳戶，否則您必須使用 AWS 管理主控台 (如下節所述) 或 EB CLI。

> **Note**
>
> 環境資訊設定可包含任何可列印 ASCII 字元，重音符號 (即 ASCII 96「`」) 除外，且長度不能超過 200 個字元。

設定 Elastic Beanstalk 應用程式的系統屬性

1. 若 Eclipse 未顯示 AWS Explorer 視圖，請選擇 Window (視窗)、Show View (顯示視圖) 和 Other (其他)。展開 AWS Toolkit，然後選擇 AWS Explorer。
2. 在 AWS Explorer 窗格中，展開 Elastic Beanstalk、展開您應用程式的節點，然後按兩下您的 Elastic Beanstalk 環境。

3. 在環境的窗格底部按一下 Advanced (進階) 索引標籤。
4. 在 aws:elasticbeanstalk:application:environment 底下按一下 JDBC_CONNECTION_STRING，然後輸入連線字串。例如，下列 JDBC 連線字串會透過使用者名稱 me 及密碼 mypassword，連接至 localhost 連接埠 3306 上的 MySQL 資料庫執行個體：

   ```
   jdbc:mysql://localhost:3306/mydatabase?user=me&password=mypassword
   ```

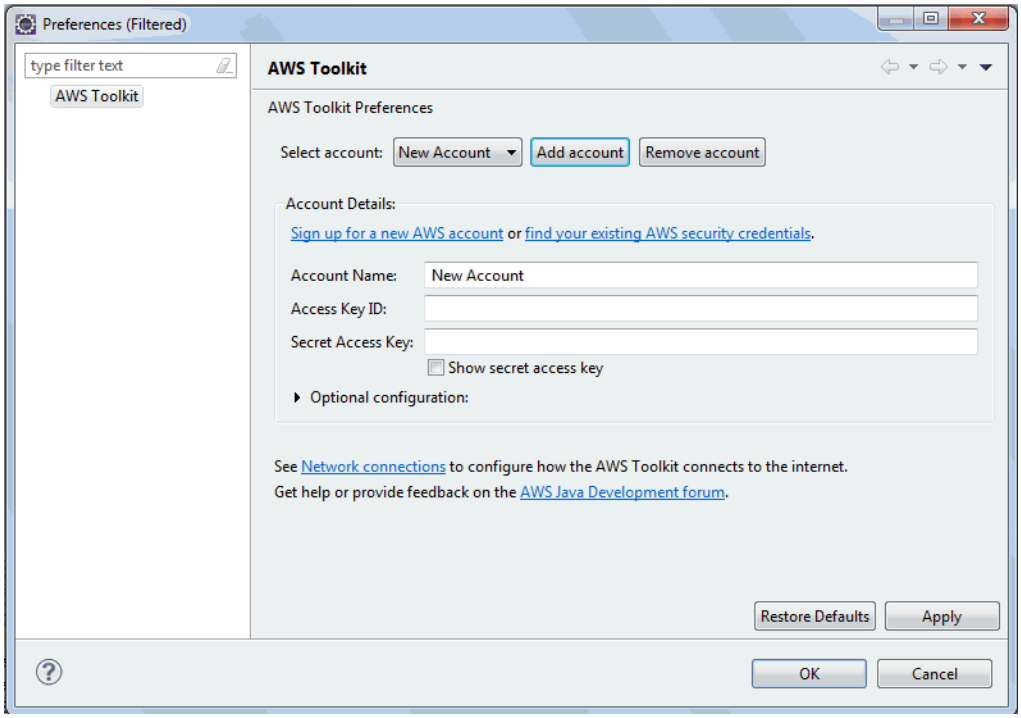
   如此一來，您的 Elastic Beanstalk 應用程式即可透過名為 JDBC_CONNECTION_STRING 的系統屬性對其進行存取。
5. 在鍵盤上按 Ctrl+C 或選擇 File (檔案)、Save (儲存)，將儲存環境資訊的變更。變更會在約一分鐘內顯現。

管理多個 AWS 帳戶

您可能想要設定不同的 AWS 帳戶來執行不同任務，例如測試、封測和生產。您可使用 AWS Toolkit for Eclipse 輕鬆新增、編輯和刪除帳戶。

使用 AWS Toolkit for Eclipse 新增 AWS 帳戶

1. 確保 Eclipse 已顯示工具列。在工具列上，按一下 AWS 圖示旁的箭頭，然後選取 Preferences (偏好設定)。
2. 按一下 Add account (新增帳戶)。



3. 在 Account Name (帳戶名稱) 文字方塊中，輸入帳戶的顯示名稱。
4. 在 Access Key ID (存取金鑰 ID) 文字方塊中，鍵入您的 AWS 存取金鑰 ID。
5. 在 Secret Access Key (私密存取金鑰) 文字方塊中，鍵入您的 AWS 私密金鑰。

   如果是 API 存取，您需要存取金鑰 ID 和私密存取金鑰。使用 IAM 使用者存取金鑰，而非 AWS 帳戶根使用者 存取金鑰。如需建立存取金鑰的詳細資訊，請參閱IAM 使用者指南中的管理 IAM 使用者的存取金鑰。
6. 按一下 OK (確定)。

使用不同帳戶將應用程式部署至 Elastic Beanstalk

1. 在 Eclipse 工具列上，按一下 AWS 圖示旁的箭頭，然後選取 Preferences (偏好設定)。
2. 針對 Default Account (預設帳戶)，請選取您欲用於將應用程式部署至 Elastic Beanstalk 的帳戶。
3. 按一下 OK (確定)。
4. 在 Project Explorer (專案瀏覽) 窗格中，請在您欲部署的應用程式上按一下滑鼠右鍵，然後選擇 Amazon Web Services > Deploy to Elastic Beanstalk (部署至 Elastic Beanstalk)。

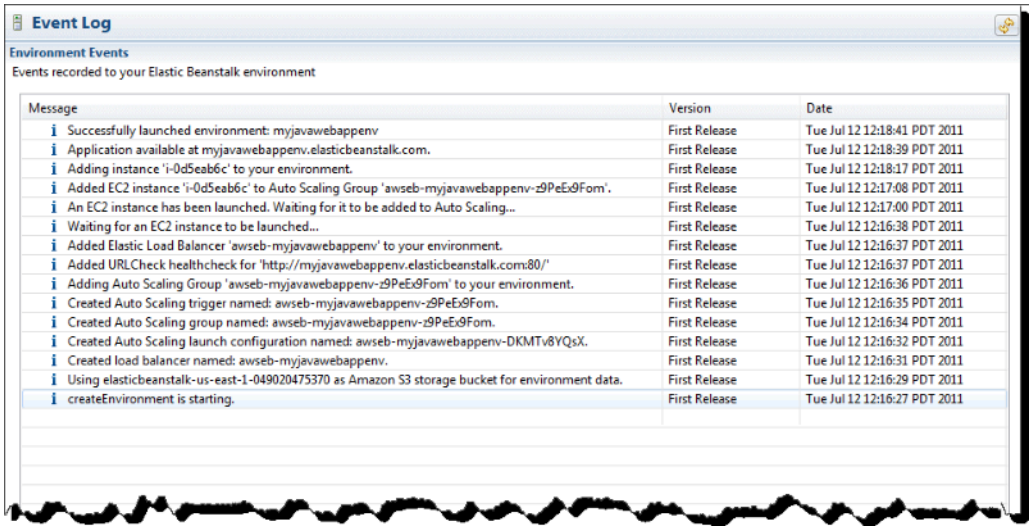
檢視 事件

您可使用 AWS Toolkit for Eclipse 來存取與您應用程式建立關聯的事件和通知。

檢視應用程式事件

1. 若 Eclipse 未顯示 AWS Explorer 視圖，請於選單中按一下 Window (視窗) > Show View (顯示視圖) > AWS Explorer。展開 Elastic Beanstalk 節點和您的應用程式節點。
2. 在 AWS Explorer 中，按兩下 Elastic Beanstalk 環境。
3. 在窗格底部按一下 Events (事件) 索引標籤。

   將顯示您應用程式所有環境的事件清單。

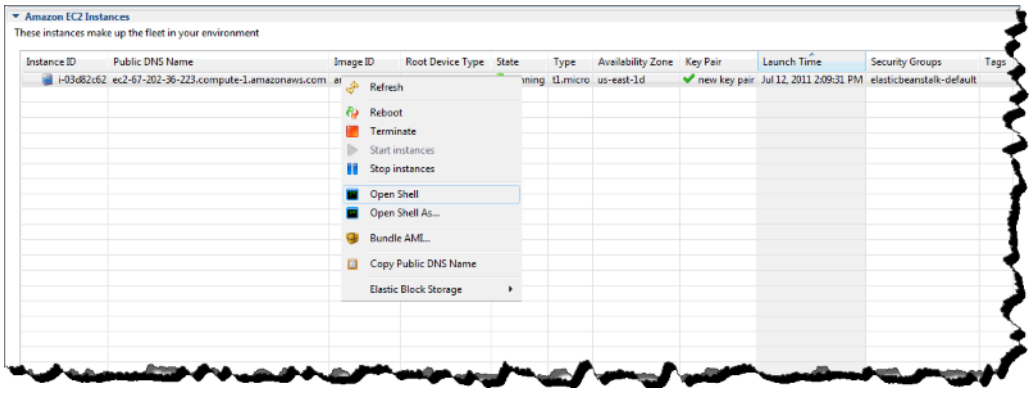


列出和連線到伺服器執行個體

您可以透過 AWS Toolkit for Eclipse，或從 AWS 管理主控台，針對執行您 Elastic Beanstalk 應用程式環境的 Amazon EC2 執行個體，來檢視其清單。您可使用安全殼層 (SSH) 連接至這些執行個體。如需透過 AWS 管理主控台列出並連接至您伺服器執行個體的詳細資訊，請參閱 列出和連線到伺服器執行個體。下列章節帶您透過 AWS Toolkit for Eclipse，逐步檢視並連接至您的伺服器執行個體。

欲檢視並連接至環境的 Amazon EC2 執行個體

1. 在 AWS Toolkit for Eclipse 中，按一下 AWS Explorer。展開 Amazon EC2 (Amazon EC2) 節點，然後按兩下 Instances (執行個體)。
2. 在 Amazon EC2 執行個體視窗的 Instance ID (執行個體 ID) 欄位，請在執行於您應用程式負載平衡器的 Amazon EC2 執行個體 Instance ID (執行個體 ID)，按一下滑鼠右鍵。然後按一下 Open Shell (開啟 Shell)。



Eclipse 會自動開啟 SSH 用戶端，並連接至 EC2 執行個體。

如需有關連接至 Amazon EC2 執行個體的詳細資訊，請參閱 Amazon Elastic Compute Cloud 入門指南。

終止環境

若要避免未使用的 AWS 資源收取費用，您可使用 AWS Toolkit for Eclipse 來終止執行環境。如需環境終止的詳細資訊，請參閱終止 Elastic Beanstalk 環境。

終止環境

1. 在 AWS Toolkit for Eclipse 中，按一下 AWS Explorer 窗格。展開 Elastic Beanstalk 節點。
2. 展開 Elastic Beanstalk 應用程式，然後在 Elastic Beanstalk 環境上按一下滑鼠右鍵。
3. 按一下 Terminate Environment (終止環境)。Elastic Beanstalk 需要幾分鐘來終止環境中執行的 AWS 資源。

資源

在開發 Java 應用程式時，您可以造訪幾個地方來取得額外的協助。

資源	描述
AWS Java 開發論壇	提出您的問題，並取得意見回饋。
Java 開發人員中心	可取得範本程式碼、文件、工具和其他資源等一應俱全的場所。

使用 Linux 上的 .NET Core

> **在 AWS 開發人員中心查看 .NET**
>
> 你有沒有停止我們的 .Net 開發人員中心？ 這是我們一站式服務所有項目 .NET 上 AWS。如需詳細資訊，請參閱開AWS 發人員中心上的 .NET。

AWS Elastic Beanstalk 適用於 Linux 上的 .NET 核心，可以使用亞馬遜網路服務輕鬆部署、管理和擴展您的 Web 應用程式。本章提供在 Amazon Linux 環境中將 .NET 核心網路應用程式部署到

Elastic Beanstalk 的指示。您可以使用彈性 Beanstalk 命令列介面 (EB CLI) 或使用彈性 BeanElastic Beanstalk 主控台，在短短幾分鐘內部署應用程式。

請遵循中的步驟，QuickStart 適用於. NET 核心以使用 EB CLI 建立和部署新的 ASP.NET 核心 Web 應用程式。

主題

- QuickStart：將 Linux 應用程序上的 .NET 核心部署到 Elastic Beanstalk
- 設定 Linux 上的 .NET Core 開發環境
- 使用 Linux 上的 .NET Core 平台
- AWS Toolkit for Visual Studio – 搭配 .Net Core 使用
- 從 Windows Server 平台上的 .NET 遷移至 Linux 上的 .NET Core 平台

QuickStart：將 Linux 應用程序上的 .NET 核心部署到 Elastic Beanstalk

本 QuickStart 教程將引導您完成在 Linux 應用程序上創建 .NET Core 並將其部署到 AWS Elastic Beanstalk 環境的過程。

> **Note**
>
> 本 QuickStart 自學課程用於示範目的。請勿將本教學課程中建立的應用程式用於生產流量。

章節

- 您的 AWS 帳戶
- 必要條件
- 第 1 步：在 Linux 應用程序上創建一個 .NET 核心
- 步驟 2：在本機執行應用程式
- 步驟 3：使用 EB CLI 在 Linux 應用程式上部署您的 .NET 核心
- 第 4 步：在 Elastic Beanstalk 上運行應用程序
- 步驟 5：清除
- AWS 您應用程式的資源
- 後續步驟

- 使用 Elastic Beanstalk 控制台進行部署

您的 AWS 帳戶

如果您還不是 AWS 客戶，則需要創建一個 AWS 帳戶。註冊使您可以訪問 Elastic Beanstalk 和您需要的其他 AWS 服務。

如果您已經有 AWS 帳戶，則可以轉到必要條件。

創建一個 AWS 帳戶

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 AWS 帳戶 AWS 帳戶根使用者、啟用和建立系統管理使用者 AWS IAM Identity Center，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」。

必要條件

為了遵循本指南的程序，您需要命令列終端機或 shell 來執行命令。命令清單前會出現提示字元 ($) 及目前的目錄名稱 (如有)。

```
~/eb-project$ this is a command
this is output
```

在 Linux 和 macOS 上，您可以使用偏好的 Shell 和套件軟體管理工具。在視窗上，你可以安裝視窗子系統為 Linux 得到一個視窗集成的 Ubuntu 和 Bash 的版本。

EB CLI

本教學使用 Elastic Beanstalk 命令列界面 (EB CLI)。關於安裝和設定 EB CLI 的詳細資訊，請參閱安裝 EB CLI 和設定 EB CLI。

Linux 上的 .NET Core

如果您的本機電腦上沒有安裝 .NET SDK，可以按照 .NET 文件網站上的下載 .NET 連結進行安裝。

執行下列命令來驗證您的 .NET SDK 安裝。

```
~$ dotnet --info
```

第 1 步：在 Linux 應用程序上創建一個 .NET 核心

建立專案目錄。

```
~$ mkdir eb-dotnetcore
~$ cd eb-dotnetcore
```

接下來，執行下列命令來建立範例 Hello World 應用程式。

```
~/eb-dotnetcore$ dotnet new web --name HelloElasticBeanstalk
~/eb-dotnetcore$ cd HelloElasticBeanstalk
```

步驟 2：在本機執行應用程式

執行下列命令以在本機執行應用程式。

```
~/eb-dotnetcore/HelloElasticBeasntalk$ dotnet run
```

輸出應該看起來像下面的文本。

```
Building...
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: https://localhost:7294
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5052
info: Microsoft.Hosting.Lifetime[0]
```

```
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
```

> **Note**
>
> 此指dotnet令會在本機執行應用程式時隨機選取連接埠。在此範例中，連接埠為 5052。當您將應用程式部署到 Elastic Beanstalk 環境時，應用程式會在連接埠 5000 上執行。

`http://localhost:`*`port`*在您的網頁瀏覽器中輸入 URL 位址。在此特定範例中，指令為`http://localhost:5052`。網絡瀏覽器應該顯示「你好世界！」。

步驟 3：使用 EB CLI 在 Linux 應用程式上部署您的 .NET 核心

執行下列命令，為此應用程式建立 Elastic Beanstalk 環境。

若要建立環境並在 Linux 應用程式上部署您的 .NET 核心

1. 將應用程式編譯並發佈至資料夾，以便部署至您即將建立的 Elastic Beanstalk 環境。

   ```
   ~$ cd eb-dotnetcore/HelloElasticBeanstalk
   ~/eb-dotnetcore/HelloElasticBeanstalk$ dotnet publish -o site
   ```

2. 導航到您剛剛發布應用程序的站點目錄。

   ```
   ~/eb-dotnetcore/HelloElasticBeanstalk$ cd site
   ```

3. 透過 eb init 命令初始化您的 EB CLI 儲存庫。

   請注意下列有關您在命令中指定的平台分支版本的詳細資訊：

   - *`x.y.z`*在以下命令中替換為 AL2023 上的平台分支 .NET 6 的最新版本。
   - 若要尋找最新的平台分支版本，請參閱《平台指南》中的 Linux 支援平AWS Elastic Beanstalk台上的 .NET Core。
   - 包含版本號碼的解決方案堆疊名稱範例為`64bit-amazon-linux-2023-v`**`3.1.1`**`-running-.net-6`。在這個例子中，分支版本是 3.1.1。

```
~eb-dotnetcore/HelloElasticBeanstalk/site$ eb init -p 64bit-amazon-linux-2023-
vx.y.z-running-.net-6 dotnetcore-tutorial --region us-east-2
Application dotnetcore-tutorial has been created.
```

此命令會建立名為的應用程式，dotnetcore-tutorial並設定您的本機存放庫，以使用指令中指定的 Linux 平台上的 .NET Core 版本來建立環境。

4. (選用) 再次執行 eb init 來設定預設金鑰對，藉此使用 SSH 連接至執行您應用程式的 EC2 執行個體：

```
~eb-dotnetcore/HelloElasticBeanstalk/site$ eb init
Do you want to set up SSH for your instances?
(y/n): y
Select a keypair.
1) my-keypair
2) [ Create new KeyPair ]
```

若您已有金鑰對，請選擇一對，或依照提示建立金鑰對。若未出現提示，或稍後需要變更設定，請執行 eb init -i。

5. 使用 eb create 建立環境並於其中部署您的應用程式。Elastic Beanstalk 會自動為您的應用程序構建一個 zip 文件，並在端口 5000 上啟動它。

至

```
~eb-dotnetcore/HelloElasticBeanstalk/site$ eb create dotnet-tutorial
```

Elastic Beanstalk 大約需要五分鐘的時間來創建您的環境。

第 4 步：在 Elastic Beanstalk 上運行應用程序

當創建環境的過程完成後，打開您的網站eb open。

```
~eb-dotnetcore/HelloElasticBeanstalk/site$ eb open
```

恭喜您！您已經使用 Elastic Beanstalk 在 Linux 應用程序上部署了一個 .NET 核心！這會開啟瀏覽器視窗，並使用為應用程式建立的網域名稱。

步驟 5：清除

您可以在完成應用程式的工作後終止環境。Elastic Beanstalk 會終止與您環境相關的所有 AWS 資源。

若要使用 EB CLI 終止 Elastic Beanstalk 環境，請執行下列命令。

```
~eb-dotnetcore/HelloElasticBeanstalk/site$ eb terminate
```

AWS 您應用程式的資源

您剛剛建立了單一執行個體應用程式。它可作為單一 EC2 執行個體的簡單範例應用程式使用，因此不需要負載平衡或 auto 擴展。對於單個實例應用程序，Elastic Beanstalk 創建以下 AWS 資源：

- EC2 執行個體 – 設定在您所選平台上執行 Web 應用程式的 Amazon EC2 虛擬機器。

  每個平台會執行不同一套軟體、設定檔和指令碼，來支援特定的語言版本、架構、Web 容器或其組合。大多數的平台會使用 Apache 或 nginx 做為反向代理，處理您 Web 應用程式前端的網路流量、向它轉送請求、提供靜態資產，並產生存取和錯誤日誌。
- 執行個體安全群組 – 設定允許從連接埠 80 傳入流量的 Amazon EC2 安全群組。此資源可讓負載平衡器傳來的 HTTP 傳輸資料，到達執行您 Web 應用程式的 EC2 執行個體。在預設情況下，不允許傳輸資料從其他通訊埠傳送。
- Amazon S3 儲存貯體 - 儲存位置，用來儲存當您使用 Elastic Beanstalk 時所建立的原始程式碼、日誌和其他成品。
- Amazon CloudWatch 警示 — 兩個 CloudWatch 警示，用於監控環境中執行個體的負載，並在負載過高或過低時觸發。當警示觸發時，您的 Auto Scaling 群組會擴展或縮減以進行回應。
- AWS CloudFormation 堆疊 — Elastic Beanstalk 用 AWS CloudFormation 來啟動環境中的資源並傳播組態變更。資源定義於範本中，您可在 AWS CloudFormation 主控台中檢視此範本。
- 網域名稱 – 會路由到您 Web 應用程式的網域名稱，其格式為 *subdomain*.*region*.elasticbeanstalk.com。

Elastic Beanstalk 會管理所有這些資源。當您終止環境時，Elastic Beanstalk 會終止其中的所有資源。

後續步驟

在您擁有執行應用程式的環境後，可以隨時部署應用程式的新版本或不同的應用程式。部署新的應用程式版本非常快速，因無須佈建或重新啟動 EC2 執行個體。您也可以使用 Elastic Beanstalk 控制台探索您的新環境。如需詳細步驟，請參閱本指南的「入門」一章中的「探索您的環境」。

在您部署一或兩個範例應用程式並準備好在本機 Linux 應用程式上開發和執行 .NET Core 之後，請參閱設定 Linux 上的 .NET Core 開發環境。

使用 Elastic Beanstalk 控制台進行部署

您也可以使用 Elastic Beanstalk 控制台來啟動示例應用程序。如需詳細步驟，請參閱本指南的「入門」一章中的「建立範例應用程式」。

設定 Linux 上的 .NET Core 開發環境

設定 .NET Core 開發環境以在本機測試您的應用程式，然後才部署到 AWS Elastic Beanstalk。此主題概述開發環境設定步驟，並提供實用工具的安裝頁面連結。

如需了解適用所有語言的常見設定步驟和工具，請參閱設定您的開發機器搭配 Elastic Beanstalk 使用。

章節

- 安裝 .NET Core SDK
- 安裝 IDE
- 安裝 AWS Toolkit for Visual Studio

安裝 .NET Core SDK

您可以使用 .NET Core SDK 來開發在 Linux 上執行的應用程式。

請參閱 .NET 下載頁面以下載並安裝 .NET Core SDK。

安裝 IDE

整合開發環境 (IDE) 提供可加速應用程式開發的各種功能。若您沒有使用 IDE 進行 .NET 開發作業的經驗，請嘗試透過 Visual Studio Community 入門。

請造訪 Visual Studio Community 頁面，下載並安裝 Visual Studio Community。

安裝 AWS Toolkit for Visual Studio

AWS Toolkit for Visual Studio 是一種 Visual Studio IDE 的開源外掛程式，可讓開發人員使用 AWS 更輕鬆開發、除錯與部署 .NET 應用程式。如需安裝指示，請參閱 Toolkit for Visual Studio 首頁。

使用 Linux 上的 .NET Core 平台

AWS Elastic Beanstalk .NET Core on Linux 平台是一組適用於 Linux 作業系統上執行的 .NET Core 應用程式的平台版本。

如需各種擴充 Elastic Beanstalk Linux 類型平台方式的詳細資訊，請參閱 the section called “擴充 Linux 平台”。以下是一些特定於平台的考量事項。

Linux 上的 .NET Core 平台簡介

代理伺服器

Elastic Beanstalk Linux 上的 .NET Core 平台包含反向代理程式，可將請求轉送給您的應用程式。預設情況下，Elastic Beanstalk 會使用 nginx 做為代理伺服器。您可以選擇不使用代理伺服器，並將 Kestrel 設定為您的 Web 伺服器。Kestrel 預設會包含在 ASP.NET Core 的專案範本中。

應用程式結構

您可以發佈「執行時間相依」的應用程式，這些應用程式使用 Elastic Beanstalk 提供的 .NET Core 執行時間。您也可以發佈自主運作的應用程式，其中包含 .NET Core 執行時間和您的應用程式在原始碼套件中的相依性。如需進一步了解，請參閱the section called “綁定應用程式”。

平台組態

欲設定在環境中伺服器執行個體上執行的程序，請於您的原始碼套件納入選用的 Procfile。如果原始碼套件中有多個應用程式，則需要有 `Procfile`。

建議務必在原始碼套件和應用程式中永遠提供 `Procfile`。透過此方式，您可以精確控制 Elastic Beanstalk 要為您的應用程式執行哪些程序。

Elastic Beanstalk 主控台中提供了修改正在執行環境組態的組態選項。要避免在終止環境的組態時遺失組態，您可以使用已儲存組態來儲存您的設定，並在之後套用至另一個環境。

若要將設定儲存於原始程式碼，您可以包含組態檔案。每次您建立環境或部署應用程式，組態檔案裡的設定就會套用。您也可以使用組態檔案來安裝套件、執行指令碼，並在部署期間執行其他執行個體自訂操作。

在 Elastic Beanstalk 主控台中套用的設定會覆寫組態檔案中相同的設定 (如存在)。這可讓您在組態檔案中擁有預設設定，並以主控台的環境專屬設定覆寫之。如需優先順序以及其他變更設定方法的詳細資訊，請參閱組態選項。

設定您的 Linux 上的 .NET Core 環境

Linux 上的 .NET Core 平台設定可讓您微調 Amazon EC2 執行個體的行為。您可以使用 Elastic Beanstalk 主控台編輯 Elastic Beanstalk 環境的 Amazon EC2 執行個體組態。

使用 Elastic Beanstalk 主控台來啟用至 Amazon S3 的日誌輪換，和設定您的應用程式可以從環境讀取的變數。

若要使用 Elastic Beanstalk 主控台設定您的 Linux 上的 .NET Core 環境

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇組態。
4. 在更新、監控和日誌記錄組態類別中，選擇編輯。

日誌選項

Log Options (日誌選項) 區段有兩個設定：

- 執行個體設定檔 – 指定有權存取與您應用程式相關的 Amazon S3 儲存貯體的執行個體設定檔。
- Enable log file rotation to Amazon S3 (啟用 Amazon S3 的日誌檔案輪換) – 指定是否將應用程式 Amazon EC2 執行個體的日誌檔案複製到與應用程式關聯的 Amazon S3 儲存貯體。

環境屬性

Environment Properties (環境屬性) 區段可讓您針對執行您應用程式的 Amazon EC2 執行個體，來指定其上的環境資訊設定。環境屬性會以金鑰值對的形式傳到應用程式。

在 Elastic Beanstalk 內所執行的 Linux 上的 .NET Core 環境中，可使用 `Environment.GetEnvironmentVariable("variable-name")` 來存取環境變數。例如，您可使用下列程式碼，來將名為 `API_ENDPOINT` 的屬性讀取到變數。

```
string endpoint = Environment.GetEnvironmentVariable("API_ENDPOINT");
```

如需詳細資訊，請參閱環境屬性與其他軟體設定。

Linux 上的 .NET Core 組態命名空間

您可以使用組態檔案來設定組態選項，並在部署期間執行其他的執行個體設定工作。組態選項可透過 Elastic Beanstalk 服務或您使用的平台來定義，並且會組織成命名空間。

除了適用於所有 Elastic Beanstalk 環境的支援選項之外，Linux 上的 .NET Core 平台也支援使用下列命名空間的選項：

- `aws:elasticbeanstalk:environment:proxy` - 選擇使用 nginx 或不使用代理伺服器。有效值為 `nginx` 或 `none`。

下列組態檔案範例示範 Linux 上的 .NET Core 特定組態選項的使用。

Example .ebextensions/proxy-settings.config

```
option_settings:
  aws:elasticbeanstalk:environment:proxy:
    ProxyServer: none
```

Elastic Beanstalk 可提供許多組態選項讓您自訂環境。除了組態檔案，您也可以使用主控台、已儲存組態、EB CLI 或 AWS CLI 來設定組態選項。如需詳細資訊，請參閱組態選項。

為 Linux 上的 .NET Core 平台綁定應用程式

您可以在 AWS Elastic Beanstalk 上執行執行時間相依和自主運作的 .NET Core 應用程式。

執行時間相依的應用程式使用 Elastic Beanstalk 提供的 .NET Core 執行時間來執行您的應用程式。Elastic Beanstalk 會使用原始碼套件中的 `runtimeconfig.json` 檔案來判斷應用程式所要使用的執行時間。Elastic Beanstalk 會為您的應用程式選擇最新的相容執行時間。

自主運作的應用程式包括 .NET Core 執行時間、您的應用程式及其相依性。若要使用的 .NET Core 執行時間版本是 Elastic Beanstalk 不包含在其平台的版本，請提供可自主運作的應用程式。

範例

您可以使用 `dotnet publish` 命令編譯自主運作和執行時間相依的應用程式。若要深入了解如何發佈 .NET Core 應用程式，請參閱 NET Core 文件中的 .NET Core 應用程式發行概觀。

下列範例檔案結構會定義使用 Elastic Beanstalk 所提供 .NET Core 執行時間的單一應用程式。

```
### appsettings.Development.json
### appsettings.json
### dotnetcoreapp.deps.json
### dotnetcoreapp.dll
### dotnetcoreapp.pdb
### dotnetcoreapp.runtimeconfig.json
### web.config
### Procfile
### .ebextensions
### .platform
```

您可以在原始碼套件中包含多個應用程式。下列範例會定義兩個要在同一個 Web 伺服器上執行的應用程式。若要執行多個應用程式，您必須在原始碼套件中包含一個 Procfile。如需完整範例應用程式，請參閱 dotnet-core-linux-multiple-apps.zip。

```
### DotnetMultipleApp1
#   ### Amazon.Extensions.Configuration.SystemsManager.dll
#   ### appsettings.Development.json
#   ### appsettings.json
#   ### AWSSDK.Core.dll
#   ### AWSSDK.Extensions.NETCore.Setup.dll
#   ### AWSSDK.SimpleSystemsManagement.dll
#   ### DotnetMultipleApp1.deps.json
#   ### DotnetMultipleApp1.dll
#   ### DotnetMultipleApp1.pdb
#   ### DotnetMultipleApp1.runtimeconfig.json
#   ### Microsoft.Extensions.PlatformAbstractions.dll
#   ### Newtonsoft.Json.dll
#   ### web.config
### DotnetMultipleApp2
#   ### Amazon.Extensions.Configuration.SystemsManager.dll
#   ### appsettings.Development.json
#   ### appsettings.json
#   ### AWSSDK.Core.dll
#   ### AWSSDK.Extensions.NETCore.Setup.dll
#   ### AWSSDK.SimpleSystemsManagement.dll
#   ### DotnetMultipleApp2.deps.json
#   ### DotnetMultipleApp2.dll
#   ### DotnetMultipleApp2.pdb
#   ### DotnetMultipleApp2.runtimeconfig.json
#   ### Microsoft.Extensions.PlatformAbstractions.dll
```

```
#   ### Newtonsoft.Json.dll
#   ### web.config
### Procfile
### .ebextensions
### .platform
```

使用 Procfile 來設定您的 Linux 上的 .NET Core 環境

若要在同一個 Web 伺服器上執行多個應用程式，您必須在原始碼套件中加入一個 `Procfile`，以告知 Elastic Beanstalk 您要執行哪些應用程式。

建議務必在原始碼套件和應用程式中永遠提供 `Procfile`。如此一來，就能精確掌控 Elastic Beanstalk 會針對應用程式執行哪些程序，以及這些程序會收到哪些引數。

下列範例使用 `Procfile` 指定 Elastic Beanstalk 要在同一個 Web 伺服器上執行的兩個應用程式。

Example Procfile

```
web: dotnet ./dotnet-core-app1/dotnetcoreapp1.dll
web2: dotnet ./dotnet-core-app2/dotnetcoreapp2.dll
```

如需有關撰寫和使用 `Procfile` 的詳細資料，請展開the section called “擴充 Linux 平台”中的 Buildfile 和 Procfile 區段。

設定 Linux 上的 .NET Core 環境的代理伺服器

AWS Elastic Beanstalk 使用 nginx 做為反向代理程式，將請求轉送到您的應用程式。Elastic Beanstalk 提供了預設的 nginx 組態，您可以加以擴展，或使用自己的組態將其完全覆寫。

Elastic Beanstalk 預設會設定 nginx 代理將請求轉送至連接埠 5000 上的應用程式。您可將 `PORT` 環境屬性設定為主要應用程式接聽的連接埠，藉此覆寫預設連接埠。

> **Note**
>
> 您應用程式接聽的連接埠，不會影響 nginx 伺服器為接收來自負載平衡器的請求所接聽的連接埠。

在您的平台版本上設定代理伺服器

所有 AL2023/AL2 平台皆支援統一的代理組態功能。如需有關在執行 AL2023/AL2 的平台版本上設定代理伺服器的詳細資訊，請展開the section called “擴充 Linux 平台”中的反向代理組態一節。

以下範例組態檔案會擴展您環境的 nginx 組態。設定會將要求導向 /api 至第二個 Web 應用程式，該應用程式會在網頁伺服器上接聽連接埠 5200。根據預設，Elastic Beanstalk 會將請求轉送至連接埠 5000 上監聽的單一應用程式。

Example **01_custom.conf**

```
location /api {
     proxy_pass          http://127.0.0.1:5200;
     proxy_http_version  1.1;

     proxy_set_header   Upgrade $http_upgrade;
     proxy_set_header   Connection $http_connection;
     proxy_set_header   Host $host;
     proxy_cache_bypass $http_upgrade;
     proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
     proxy_set_header   X-Forwarded-Proto $scheme;
}
```

AWS Toolkit for Visual Studio – 搭配 .Net Core 使用

AWS Toolkit for Visual Studio 是 Visual Studio IDE 的外掛程式。使用此工具組，您可以在 Visual Studio 環境中工作時，在 Elastic Beanstalk 部署和管理應用程式。

本主題說明如何使用 AWS Toolkit for Visual Studio 執行下列工作：

- 使用 Visual Studio 範本建立 ASP.NET Core Web 應用程式。
- 建立 Elastic Beanstalk Amazon Linux 環境。
- 將 ASP.NET Core Web 應用程式部署到新的 Amazon Linux 環境。

本主題也探索如何使用 AWS Toolkit for Visual Studio 來管理 Elastic Beanstalk 應用程式環境，以及監控應用程式的運作狀態。

章節

- 建立 Elastic Beanstalk 環境並部署您的應用程式
- 終止環境
- 管理您的 Elastic Beanstalk 應用程式環境
- 監控應用程式運作狀態

先決條件

在開始本教學課程之前，您需要安裝 AWS Toolkit for Visual Studio。如需說明，請參閱設定 AWS Toolkit for Visual Studio。

如果您之前從未使用過該工具組，則在安裝工具組後需要做的第一件事是使用工具組註冊您的 AWS 登入資料。如需此方面的詳細資訊，請參閱提供 AWS 登入資料。

建立新的應用程式專案

如果 Visual Studio 中沒有 .NET Core 應用程式專案，您也可以輕鬆使用 Visual Studio 的一個專案範本建立專案。

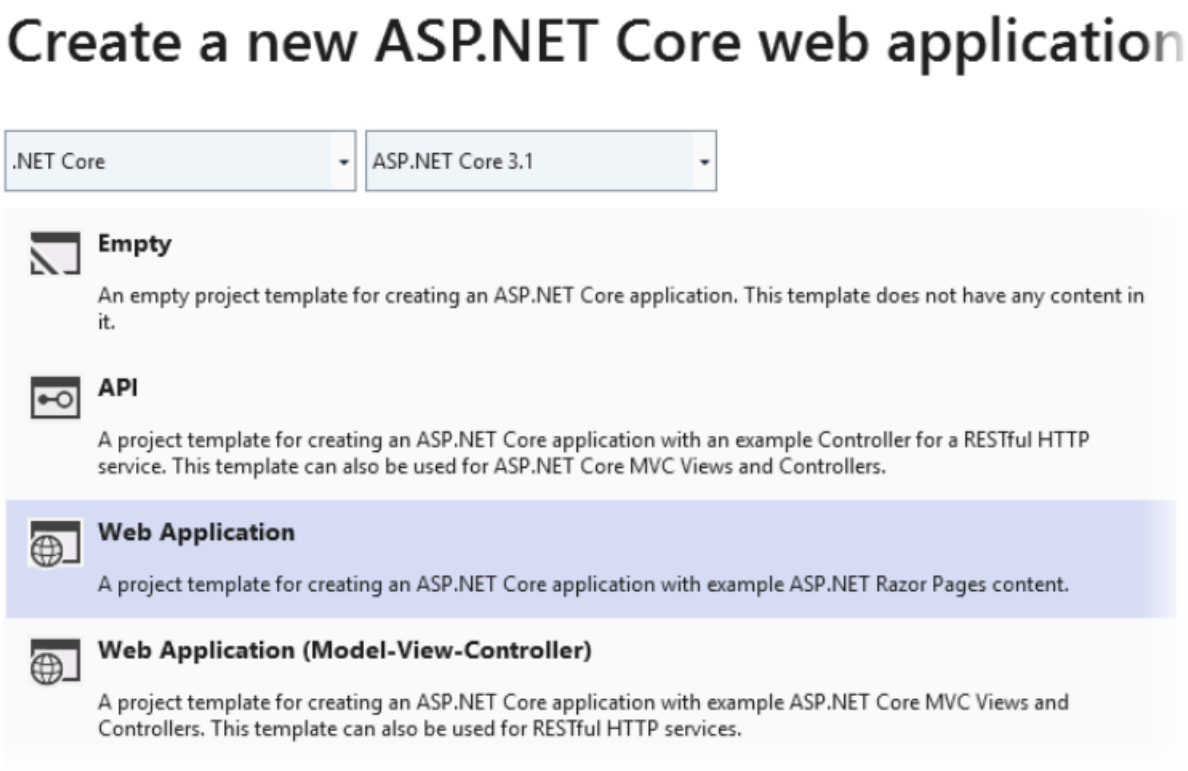
若要建立新的 ASP.NET Web 應用程式專案

1. 在 Visual Studio 中，於 File (檔案) 功能表中選擇 New (新增)，然後選擇 Project (專案)。
2. 在 Create a new project (建立新專案) 對話方塊中，選取 C#，選取 Linux，然後選取 Cloud (雲端)。
3. 從顯示的專案範本清單中選取 ASP.NET Core Web Application (ASP.NET Core Web 應用程式)，然後選取 Next (下一步)。

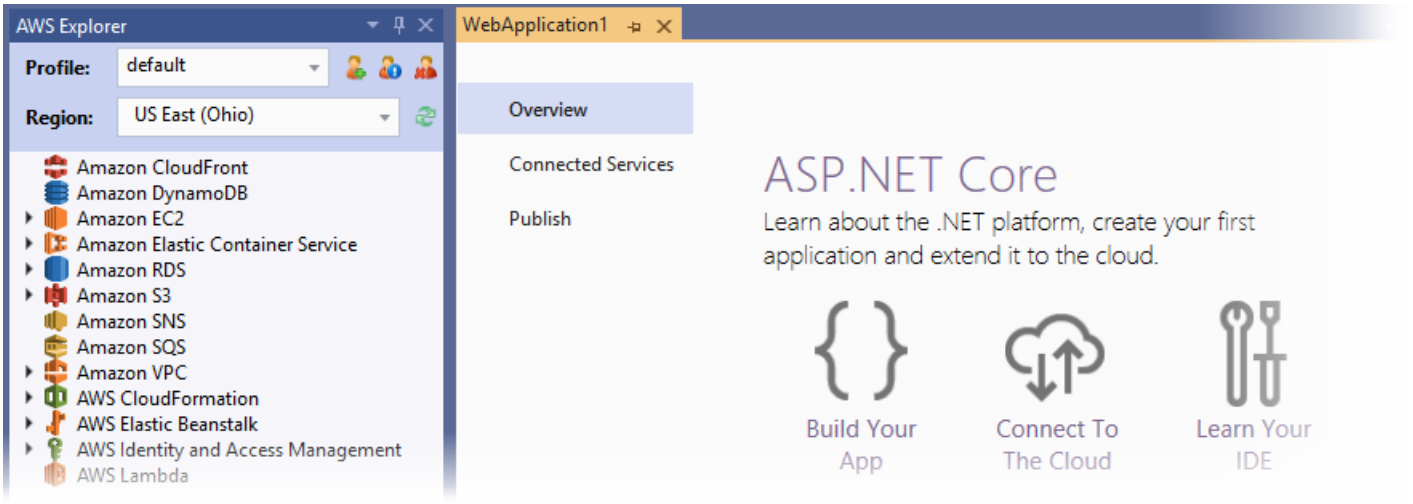
> **Note**
>
> 如果專案範本中未列出的 ASP.NET Core Web Application (ASP.NET Core Web 應用程式)，您可以在 Visual Studio 中安裝該範本。
>
> 1. 捲動至範本清單的底部，然後選取位於範本清單下的 Install more tools and features (安裝更多工具和功能) 連結。
> 2. 如果系統提示您允許 Visual Studio 應用程式變更您的裝置，請選取 Yes (是)。
> 3. 選擇 Workloads (工作負載) 標籤，然後選取 ASP.NET and web development (ASP.NET 和 Web 開發)。
> 4. 選取 Modify (修改) 按鈕。Visual Studio Installer 安裝程式會安裝專案範本。

5. 安裝程式完成後，結束面板返回您上次在 Visual Studio 離開的位置。

4. 在 Configure your new project (配置新專案) 對話方塊中，輸入專案名稱。解決方案名稱會預設使用您的專案名稱。接下來，選擇 Create (建立)。
5. 在 Create a new ASP.NET Core web application (建立新的 ASP.NET Core Web 應用程式) 對話方塊中，選取 .NET Core，然後選取 ASP.NET Core 3.1。從顯示的應用程式類型清單中選取 Web Application (Web 應用程式)，然後選取 Create (建立) 按鈕。



Visual Studio 會在建立應用程式時顯示 Creating Project (建立專案) 對話方塊。當 Visual Studio 完成產生應用程式時會顯示一個面板，其中含有您的應用程式名稱。

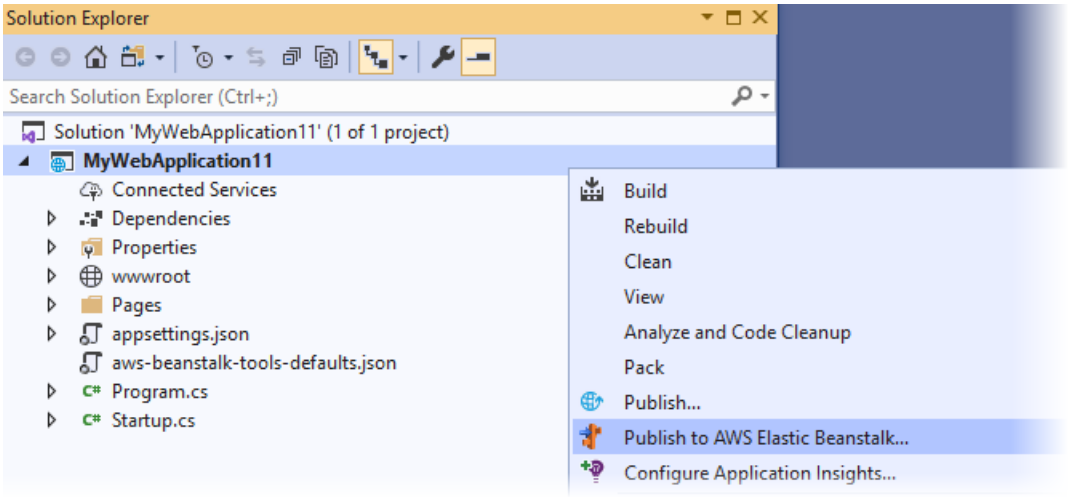


建立 Elastic Beanstalk 環境並部署您的應用程式

本節說明如何為應用程式建立 Elastic Beanstalk 環境，並將應用程式部署到該環境。

建立新環境並部署您的應用程式

1. 在 Visual Studio 中選取 View (檢視)，然後選取 Solution Explorer (方案總管)。
2. 在 Solution Explorer 中，開啟應用程式的內容 (按一下滑鼠右鍵) 選單，然後選取 Publish to AWS Elastic Beanstalk (發佈至 AWS Elastic Beanstalk)。

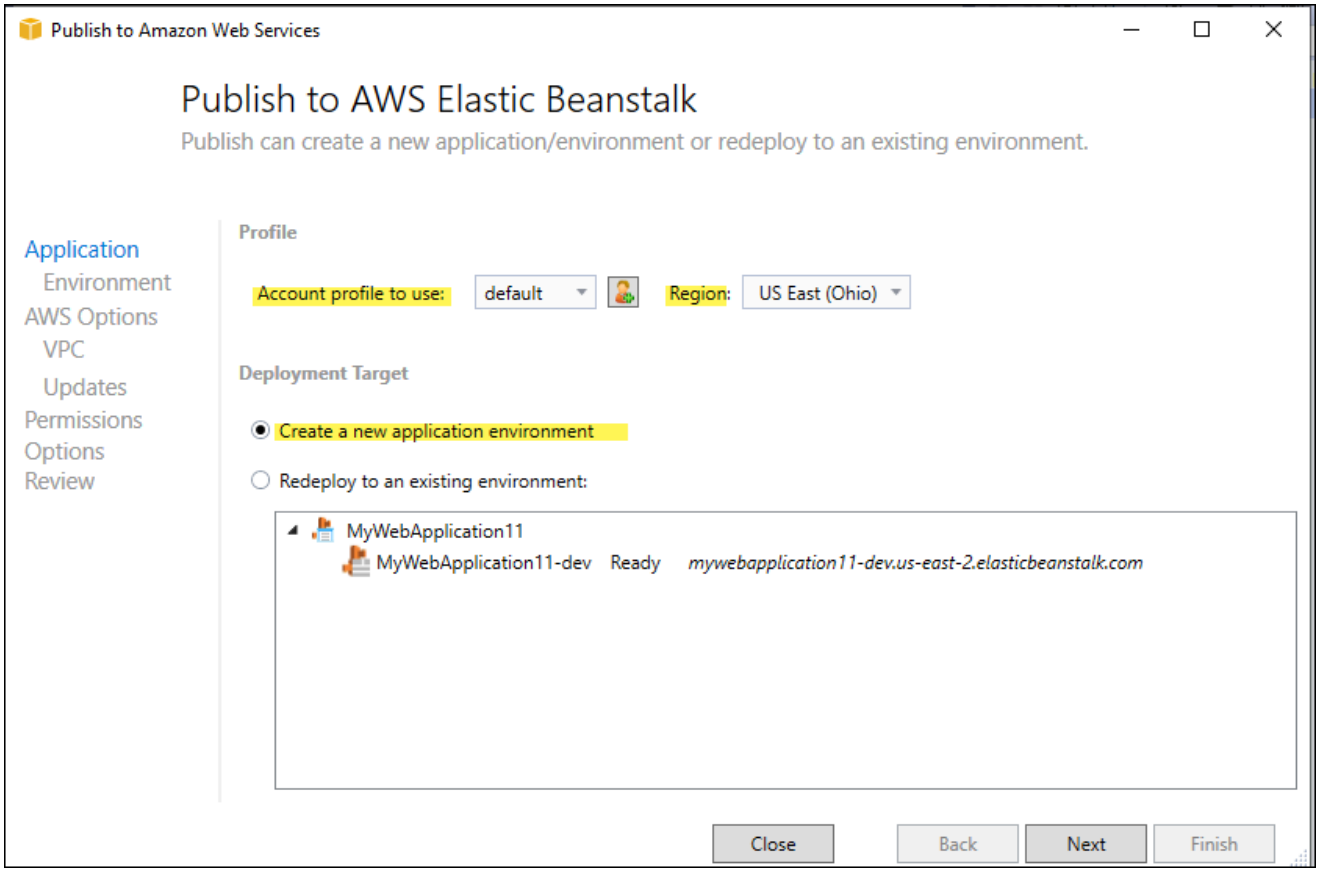


3. 在 Publish to AWS Elastic Beanstalk (發佈至 AWS Elastic Beanstalk) 精靈中，輸入您的帳戶資訊。

   a. 對於Account profile to use (要使用的帳戶設定檔)，請選取您的預設帳戶，或選擇 Add another account (新增其他帳戶) 圖示以輸入新的帳戶資訊。

b. 在 Region (區域) 部分，選取您欲部署應用程式的區域。如需可用 AWS 區域的資訊，請參閱《AWS 一般參考》中的 AWS Elastic Beanstalk 端點與配額。如果 Elastic Beanstalk 不支援您選擇的區域，即無法選擇部署至 Elastic Beanstalk 的選項。

c. 選取 Create a new application environment (建立新的應用程式環境)，然後選擇 Next (下一步)。



4. 在 Application Environment (應用程式環境) 對話方塊中，輸入新應用程式環境的詳細資料。

5. 在下一個 AWS 選項對話方塊中，為已部署的應用程式設定 Amazon EC2 選項和其他 AWS 相關選項。

a. 對於容器類型，選取執行 .NET Core 的 64bit Amazon Linux 2 v*<n.n.n>*。

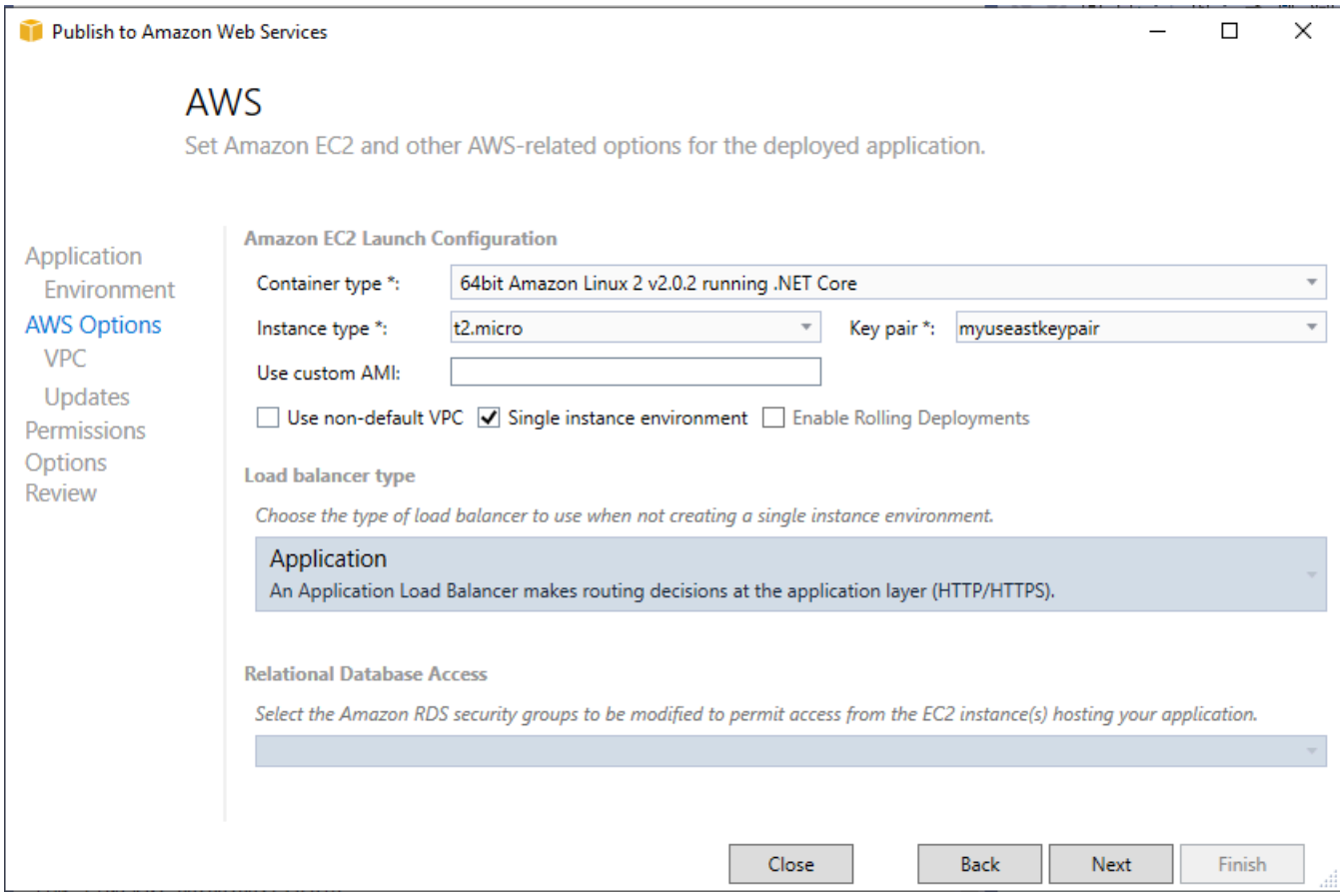
> **Note**
>
> 我們建議您選取 Linux 的目前平台版本。此版本包含了我們最新的 Amazon Machine Image (AMI) 中包含的最新安全性和錯誤修正。

b. 對於執行個體類型，請選擇 t2.micro。(選擇微型執行個體類型可將執行個體相關的成本降至最低。)

c. 在 Key pair (金鑰對) 的部分，選取 Create new key pair (建立新的金鑰對)。輸入新金鑰對的名稱，然後選擇 OK (確定)。(在此範例中，我們使用 **myuseastkeypair**。) 金鑰對可啟用您 Amazon EC2 執行個體的遠端桌面存取。如需 Amazon EC2 金鑰對的詳細資訊，請參閱《Amazon Elastic Compute Cloud 使用者指南》中的使用憑證。

d. 針對簡單、低流量的應用程式，請選取 Single instance environment (單一執行個體環境)。如需詳細資訊，請參閱環境類型。

e. 選取 下一步。



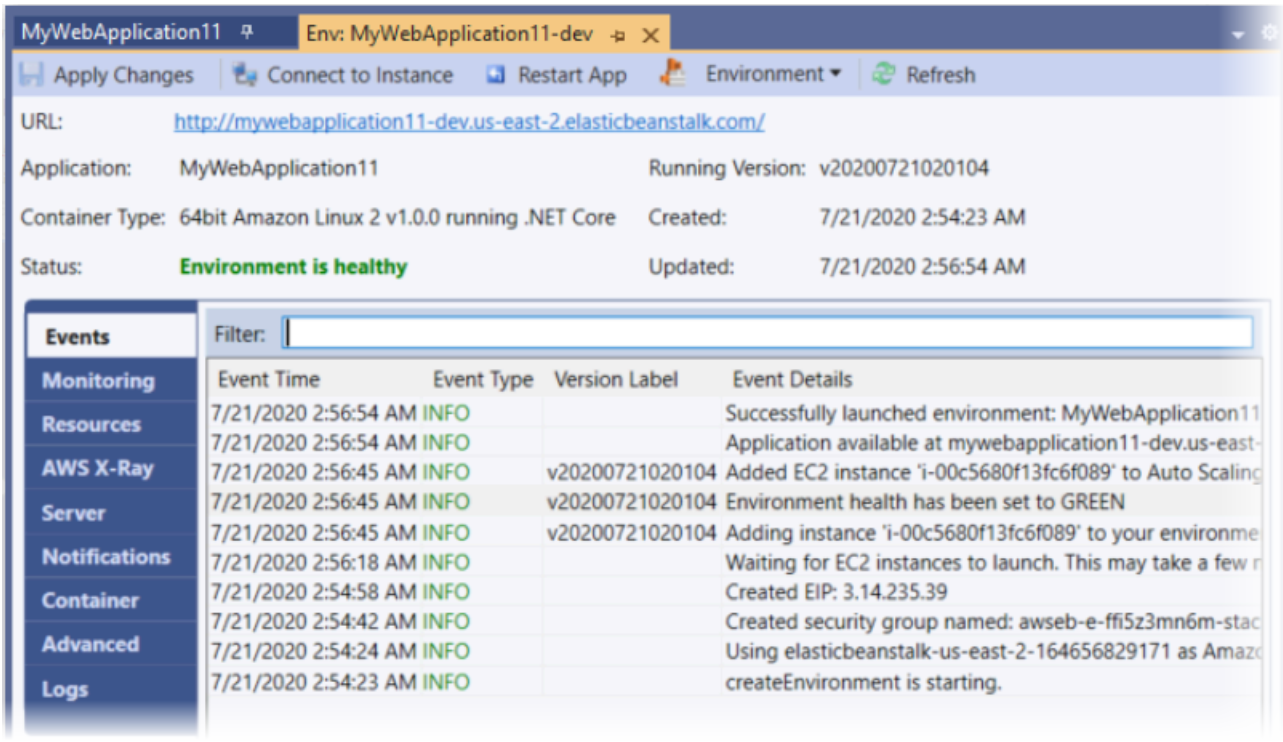
如需此範例中未使用之 AWS 選項的詳細資訊，請考慮下列頁面：

- 如需使用自訂 AMI，請參閱使用自訂的 Amazon Machine Image (AMI)。
- 如果未選取 Single instance environment (單一執行處理環境)，則需要選擇 Load balance type (負載平衡類型)。如需詳細資訊，請參閱 您的 Elastic Beanstalk 環境的負載平衡器。
- 如果您未選擇使用非預設 VPC，Elastic Beanstalk 會使用預設的 Amazon VPC (Amazon Virtual Private Cloud) 組態。如需詳細資訊，請參閱 搭配 Amazon VPC 使用 Elastic Beanstalk。

- 選擇 Enable Rolling Deployments (啟用輪流部署) 選項會將部署分為多個批次，以避免部署期間可能的停機時間。如需更多詳細資訊，請參閱 將應用程式部署至 Elastic Beanstalk 環境。
- 選擇 Relational Database Access (關聯式資料庫存取) 選項可讓您將 Elastic Beanstalk 環境透過「Amazon RDS 資料庫安全群組」連線到先前建立的 Amazon RDS 資料庫。如需詳細資訊，請參閱《Amazon RDS 使用者指南》中的使用安全群組控制存取。

6. 在許可對話方塊中選取下一步。
7. 在應用程式選項對話方塊中選取下一步。
8. 檢閱您的部署選項。確認設定正確之後，請選取 Deploy (部署)。

您的 ASP.NET Core Web 應用程式匯出為 Web 部署檔案。此檔案會上傳至 Amazon S3，且會向 Elastic Beanstalk 註冊為新的應用程式版本。Elastic Beanstalk 部署功能會監控您的環境，直到其具備新部署的程式碼且可供使用。在 Env:<environment name> 標籤中會顯示環境的狀態。狀態更新為 Environment is healthy (環境的運作狀態正常) 的狀態後，請選取 URL 位址以啟動 Web 應用程式。



終止環境

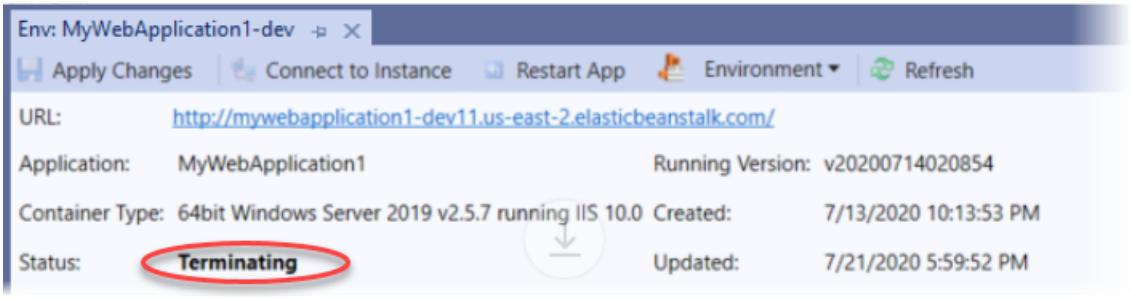
您可以使用 AWS Toolkit for Visual Studio 來終止執行中的環境，以避免未使用的 AWS 資源向您收取費用。

> **Note**
>
> 您稍後可隨時運用相同版本啟動新的環境。

終止環境

1. 展開 Elastic Beanstalk 節點和應用程式節點。在 AWS Explorer 中，開啟應用程式環境的內容 (按一下滑鼠右鍵) 選單，然後選取 Terminate Environment (終止環境)。
2. 提示出現時，按一下 Yes (是) 以確認您希望終止該環境。Elastic Beanstalk 需要幾分鐘來終止環境中執行的 AWS 資源。

在 Env:<environment name> 標籤的中您環境的狀態會變更為 Terminating (終止中) 最後會變成 Terminated (已終止)。



> **Note**
>
> 當您終止環境後，與該終止環境相關聯的 CNAME 可供任何人使用。

管理您的 Elastic Beanstalk 應用程式環境

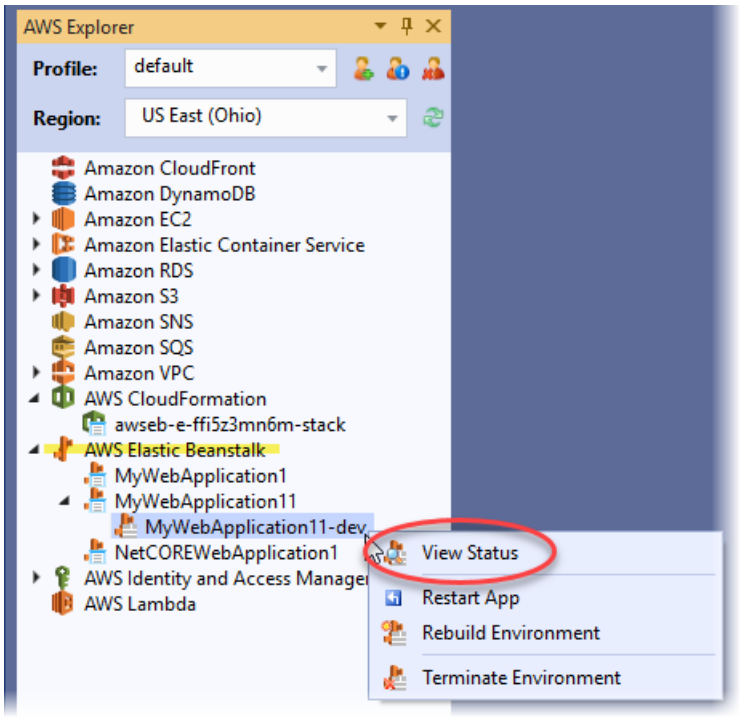
透過 AWS Toolkit for Visual Studio 及 AWS 管理主控台，您可針對應用程式環境所用的 AWS 資源，變更其佈建與組態。如需有關如何使用 AWS 管理主控台來管理您的應用程式環境的資訊，請參閱 管理環境。本節說明您可於 AWS Toolkit for Visual Studio 編輯的特定服務設定，以將其做為應用程式環境資訊的一部分。

變更環境資訊設定

部署應用程式時，Elastic Beanstalk 會設定多種 AWS 雲端運算服務。您可使用 AWS Toolkit for Visual Studio 來控制如何設定這些個別服務。

欲編輯應用程式的環境設定

1. 在 Visual Studio 中的 File (檔案) 選單上，選擇 AWS Explorer。
2. 展開 Elastic Beanstalk 節點和您的應用程式節點。開啟應用程式環境的內容 (按一下滑鼠右鍵) 功能表，然後選取 View Status (檢視狀態)。

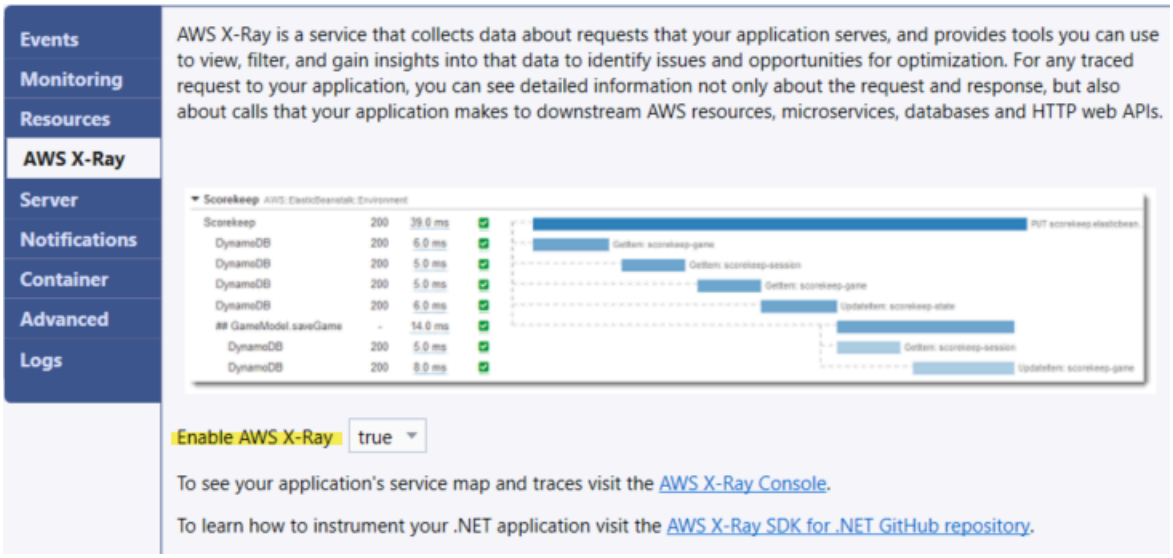


您現可進行下列設定：

- AWS X-Ray
- 伺服器
- 負載平衡器 (僅適用於多個執行個體環境)
- Auto Scaling (僅適用於多個執行個體環境)
- 通知
- 容器
- 進階組態選項

使用 AWS Toolkit for Visual Studio 設定 AWS X-Ray

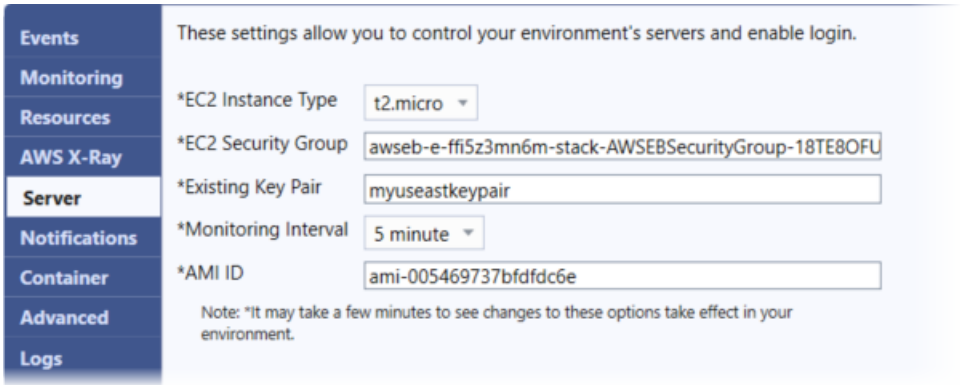
AWS X-Ray 提供請求追蹤、例外狀況收集和效能分析功能。使用 AWS X-Ray 面板，您可針對應用程式啟用或停用 X-Ray。如需 X-Ray 的詳細資訊，請參閱《AWS X-Ray 開發人員指南》。



使用 AWS Toolkit for Visual Studio 設定 EC2 執行個體

您可以使用 Amazon Elastic Compute Cloud (Amazon EC2) 在 Amazon 的資料中心啟動和管理伺服器執行個體。您可隨時使用 Amazon EC2 伺服器執行個體，無時間限制，且任何法律用途皆可。執行個體具備不同的大小與組態。如需詳細資訊，請參閱 Amazon EC2。

在 AWS Toolkit for Visual Studio 中的應用程式環境標籤，透過 Server (伺服器) 標籤即可編輯環境中的 Amazon EC2 執行個體組態。



Amazon EC2 執行個體類型

Instance type (執行個體類型) 顯示您 Elastic Beanstalk 應用程式可用的執行個體類型。請變更執行個體類型，以選取具有最適合您應用程式之特性 (包含記憶體大小和 CPU 能力) 的伺服器。例如，需要進行大量及長時間操作的應用程式可要求更多 CPU 或記憶體。

如需有關您 Elastic Beanstalk 應用程式可用之 Amazon EC2 執行個體類型的詳細資訊，請參閱《Amazon Elastic Compute Cloud 使用者指南》中的執行個體類型。

Amazon EC2 安全群組

您可使用「Amazon EC2 安全群組」來控制 Elastic Beanstalk 應用程式的存取。安全群組會定義您的執行個體的防火牆規則。這些規則會指定哪些傳入網路流量應傳送至您的執行個體。所有其他輸入流量都將捨棄。您可隨時修改群組的規則。新規則會自動於所有執行中的執行個體以及未來啟動的執行個體上實施。

您可以指定哪些 Amazon EC2 安全群組負責控制 Elastic Beanstalk 應用程式的存取。若要這樣做，請在 EC2 安全群組文字方塊中輸入特定 Amazon EC2 安全群組的名稱 (以逗號分隔多個安全群組)。您可以使用 AWS 管理主控台或 AWS Toolkit for Visual Studio 來執行這項操作。

使用 AWS Toolkit for Visual Studio 建立安全群組

1. 在 Visual Studio 的 AWS Explorer 中，展開 Amazon EC2 節點，然後選取 Security Groups (安全群組)。
2. 選取 Create Security Group (建立安全群組)，然後輸入安全群組的名稱和描述。
3. 選取 OK (確定)。

如需 Amazon EC2 安全群組的詳細資訊，請參閱Amazon Elastic Compute Cloud 使用者指南中的使用安全群組相關文章。

Amazon EC2 金鑰對

透過 Amazon EC2 金鑰對，您可安全登入為 Elastic Beanstalk 應用程式佈建的 Amazon EC2 執行個體。

> **Important**
>
> 您必須建立 Amazon EC2 金鑰對，並將 Elastic Beanstalk 佈建的 Amazon EC2 執行個體設定為能夠存取這些執行個體。將應用程式部署至 Elastic Beanstalk 時，您可使用 AWS Toolkit for Visual Studio 內的 Publish to AWS (發佈至 AWS) 精靈來建立金鑰對。若您想要使用 Toolkit 建立其他金鑰對，請依照下述步驟。或者，您可以使用 AWS 管理主控台，設定您的 Amazon EC2 金鑰對。如需建立 Amazon EC2 金鑰對的詳細資訊，請參閱 Amazon Elastic Compute Cloud 入門指南。

Existing Key Pair (現有金鑰對) 文字方塊可讓您指定欲使用的 Amazon EC2 金鑰對名稱，藉此安全登入執行 Elastic Beanstalk 應用程式的 Amazon EC2 執行個體。

欲指定 Amazon EC2 金鑰對名稱

1. 展開 Amazon EC2 節點，然後選取 Key Pairs (金鑰對)。
2. 按一下 Create Key Pair (建立金鑰對)，然後輸入金鑰對名稱。
3. 選取 OK (確定)。

如需 Amazon EC2 金鑰對的詳細資訊，請前往《Amazon Elastic Compute Cloud 使用者指南》中的使用 Amazon EC2 登入資料相關文章。如需連線至 Amazon EC2 執行個體的詳細資訊，請參閱

監控間隔

根據預設，只會啟用基本 Amazon CloudWatch 指標。其會每五分鐘傳回資料一次。您可以藉由為 AWS Toolkit for Eclipse 中您環境之 Configuration (組態) 標籤的 Server (伺服器) 區段中 Monitoring Interval (監控間隔)，選取 1 minute (1 分鐘)，來啟用以每一分鐘為一個間隔之更精密的 CloudWatch 指標。

> **Note**
>
> Amazon CloudWatch 服務費用可適用於一分鐘間隔指標。如需詳細資訊，請參閱 Amazon CloudWatch。

自訂 AMI ID

您可以藉由將您自訂 AMI 的識別符輸入 AWS Toolkit for Eclipse 中您環境 Configuration (組態) 標籤之 Server (伺服器) 區段中的 Custom AMI ID (自訂 AMI ID) 方塊，來使用您的自訂 AMI 覆寫您 Amazon EC2 執行個體的預設 AMI。

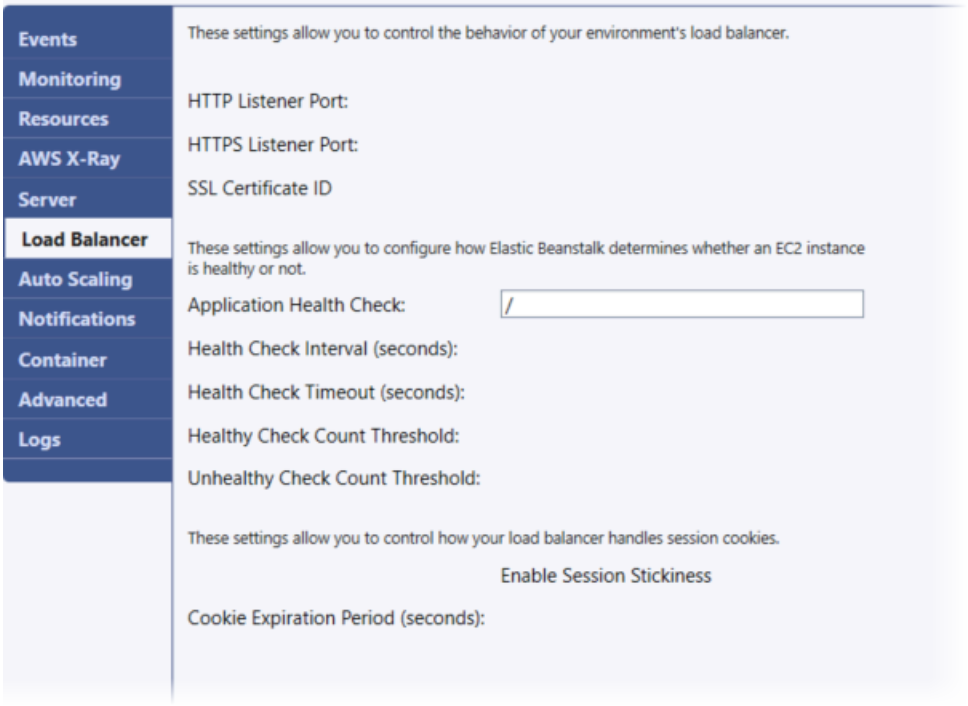
> **Important**
>
> 使用您自己的 AMI 為進階任務，需要謹慎操作。若您需要自訂 AMI，建議您從預設的 Elastic Beanstalk AMI 開始並加以修改。為使運作狀態良好，Elastic Beanstalk 預期 Amazon EC2 執行個體應滿足一組要求，包含具有執行中的主機管理員。若未符合這些要求，您的環境可能無法正常運作。

使用 AWS Toolkit for Visual Studio 設定 Elastic Load Balancing

Elastic Load Balancing 是 Amazon Web 服務，可協助您提升應用程式的可用性和可擴展性。本服務可讓您輕鬆將應用程式負載分配至兩個或多個 Amazon EC2 執行個體。Elastic Load Balancing 可透過備援改善可用性，亦可支援應用程式的流量成長。

透過 Elastic Load Balancing，您可以在所有執行中執行個體之間自動分配和平衡傳入的應用程式流量。您也可以在需要增加應用程式容量時輕鬆新增執行個體。

部署應用程式時，Elastic Beanstalk 會自動佈建 Elastic Load Balancing。在 AWS Toolkit for Visual Studio 中的應用程式環境標籤，透過 Load Balancer (負載平衡器) 標籤即可編輯 Elastic Beanstalk 環境的 Amazon EC2 執行個體組態。



下列章節說明您可設定的 Elastic Load Balancing 參數供應用程式使用。

連接埠

為了為您的 Elastic Beanstalk 應用程式處理請求所佈建的負載平衡器，會將請求傳送至執行您應用程式的 Amazon EC2 執行個體。所佈建的負載平衡器可於 HTTP 和 HTTPS 連接埠接聽請求，並將請求路由至 AWS Elastic Beanstalk 應用程式內的 Amazon EC2 執行個體。負載平衡器預設會處理 HTTP 連接埠上的請求。對於此工作，您必須開啟至少一個 HTTP 或 HTTPS 連接埠。

> **Important**
>
> 請確認您指定的連接埠並未鎖定；否則，您將無法連接至您的 Elastic Beanstalk 應用程式。

控制 HTTP 連接埠

欲關閉 HTTP 連接埠，請於 HTTP Listener Port (HTTP 接聽程式連接埠) 選取 OFF (關閉)。欲開啟 HTTP 連接埠，請於清單選取 HTTP 連接埠 (如 80 (80))。

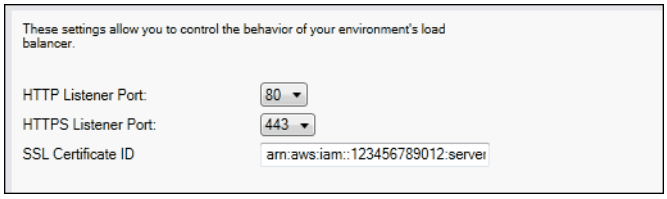
> **Note**
>
> 若要使用預設連接埠 80 以外 (例如連接埠 8080) 的連接埠存取您的環境，請將接聽程式新增至現有的負載平衡器，然後設定該新的接聽程式在該連接埠上接聽。
> 例如，使用適用於 Classic Load Balancer 的 AWS CLI，輸入如下命令，將 *LOAD_BALANCER_NAME* 取代為您 Elastic Beanstalk 負載平衡器的名稱。
>
> ```
> aws elb create-load-balancer-listeners --load-balancer-name LOAD_BALANCER_NAME
>  --listeners "Protocol=HTTP, LoadBalancerPort=8080, InstanceProtocol=HTTP,
>  InstancePort=80"
> ```
>
> 例如，使用適用於 Application Load Balancer 的 AWS CLI，輸入如下命令，將 *LOAD_BALANCER_ARN* 取代為您 Elastic Beanstalk 負載平衡器的 ARN。
>
> ```
> aws elbv2 create-listener --load-balancer-arn LOAD_BALANCER_ARN --protocol HTTP
>  --port 8080
> ```
>
> 若您想要 Elastic Beanstalk 監控您的環境，請勿移除連接埠 80 上的接聽程式。

控制 HTTPS 連接埠

Elastic Load Balancing 支援 HTTPS/TLS 通訊協定，可加密用戶端連線至負載平衡器的流量。負載平衡器至 EC2 執行個體的連線採用純文字加密。HTTPS 連接埠預設為關閉。

欲開啟 HTTPS 連接埠

1. 使用 AWS Certificate Manager (ACM) 建立新的憑證或將憑證和金鑰上傳至 AWS Identity and Access Management (IAM)。如需請求 ACM 憑證的詳細資訊，請參閱《AWS Certificate Manager 使用者指南》中的請求憑證。如需有關將第三方憑證匯入 ACM 的詳細資訊，請參閱《AWS Certificate Manager 使用者指南》中的匯入憑證。如果 ACM 無法在您的區域中使用，請使用 AWS Identity and Access Management (IAM) 上傳第三方憑證。ACM 和 IAM 服務會存放憑證，並針對 SSL 憑證提供 Amazon Resource Name (ARN)。如需建立和上傳憑證至 IAM 的詳細資訊，請參閱IAM 使用者指南中的使用伺服器憑證。
2. 選取 HTTPS Listener Port (HTTPS 接聽程式連接埠) 的連接埠來指定 HTTPS 連接埠。



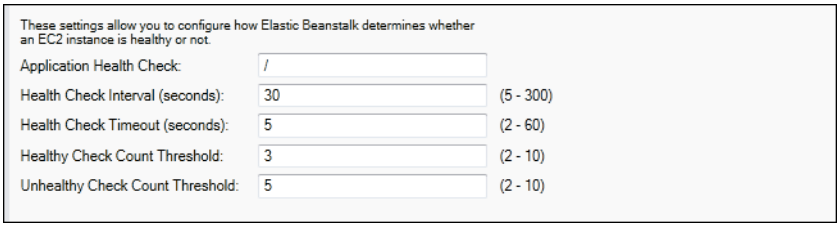
3. 針對 SSL Certificate ID (SSL 憑證 ID)，輸入您 SSL 憑證的 Amazon Resources Name (ARN)，例如 **`arn:aws:iam::123456789012:server-certificate/abc/certs/build`** 或 **`arn:aws:acm:us-east-2:123456789012:certificate/12345678-12ab-34cd-56ef-12345678`**。使用您於步驟 1 建立或上傳的 SSL 憑證。

欲關閉 HTTPS 連接埠，請於 HTTPS Listener Port (HTTPS 接聽程式連接埠) 選取 OFF (關閉)。

運作狀態檢查

運作狀態檢查的定義包括用於查詢執行個體運作狀態的 URL。根據預設，Elastic Beanstalk 針對非舊版容器和舊版容器分別使用 TCP:80 和 HTTP:80。透過 Application health check URL (應用程式運作狀態檢查 URL) 方塊中輸入與應用程式現有資源相符的 URL (例如 `/myapp/default.aspx`)，覆寫預設 URL。若您覆寫預設 URL，Elastic Beanstalk 會使用 HTTP 來查詢資源。欲檢查您是否正使用舊版容器類型，請參閱 the section called “為何部分平台版本標記為舊版？”

您可於 Load Balancing (負載平衡) 面板使用 EC2 Instance Health Check (EC2 執行個體運作狀態檢查) 區段，藉此控制運作狀態檢查的設定。



運作狀態檢查的定義包括用於查詢執行個體運作狀態的 URL。透過 Application Health Check URL (應用程式運作狀態檢查 URL) 方塊中輸入與應用程式現有資源相符的 URL (例如 `/myapp/index.jsp`)，覆寫預設 URL。

下列清單說明您應用程式可設定的運作狀態檢查參數。

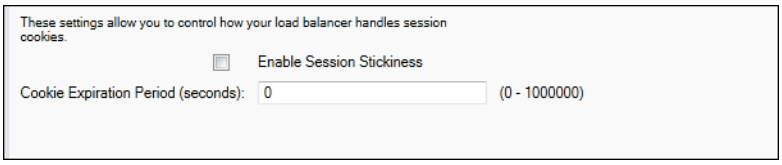
- 在 Health Check Interval (seconds) (運作狀態檢查間隔，秒) 部分，輸入 Elastic Load Balancing 對應用程式的 Amazon EC2 執行個體進行運作狀態檢查之間的等待秒數。
- 在 Health Check Timeout (seconds) (運作狀態檢查逾時，秒) 部分，指定 Elastic Load Balancing 將執行個體視為沒有回應前的等待回應的秒數。
- 在 Healthy Check Count Threshold (運作狀態檢查計數閾值) 和 Unhealthy Check Count Threshold (不健全檢查結果計數臨界值) 部分，指定 Elastic Load Balancing 變更執行個體運作狀態前的連續成功或不成功 URL 探測的次數。例如，將 Unhealthy Check Count Threshold (運作狀態不正常數量上限) 指定為 **5**，代表在 Elastic Load Balancing 認為運作狀態檢查失敗前，URL 必須連續五次傳回錯誤訊息或逾時。

工作階段

負載平衡器預設會以最小的負載，將每個請求獨立路由至伺服器執行個體。相對而言，黏性工作階段會將使用者工作階段繫結到特定的伺服器執行個體，以便工作階段期間來自使用者的所有請求都發送到相同的伺服器執行個體。

應用程式若啟用黏性工作階段，Elastic Beanstalk 會使用負載平衡器產生的 HTTP Cookie。負載平衡器會使用特殊負載平衡器產生的 Cookie，來追蹤每個請求的應用程式執行個體。當負載平衡器收到請求時，首先會檢查此 Cookie 是否存在於請求中。如果存在，請求會傳送到 cookie 中指定的應用程式執行個體。若 Cookie 不存在，則負載平衡器會根據現有負載平衡演算法選擇應用程式執行個體。回應會插入 Cookie，藉此將後續來自相同使用者的請求繫結至該應用程式執行個體。政策組態會定義 Cookie 到期日期，此為每個 Cookie 的有效使用期限。

您可使用 Load Balancer (負載平衡器) 索引標籤上的 Sessions (工作階段) 區段，以指定應用程式的負載平衡器允許讓工作階段黏著。



如需 Elastic Load Balancing 的詳細資訊，請參閱 Elastic Load Balancing 開發人員指南。

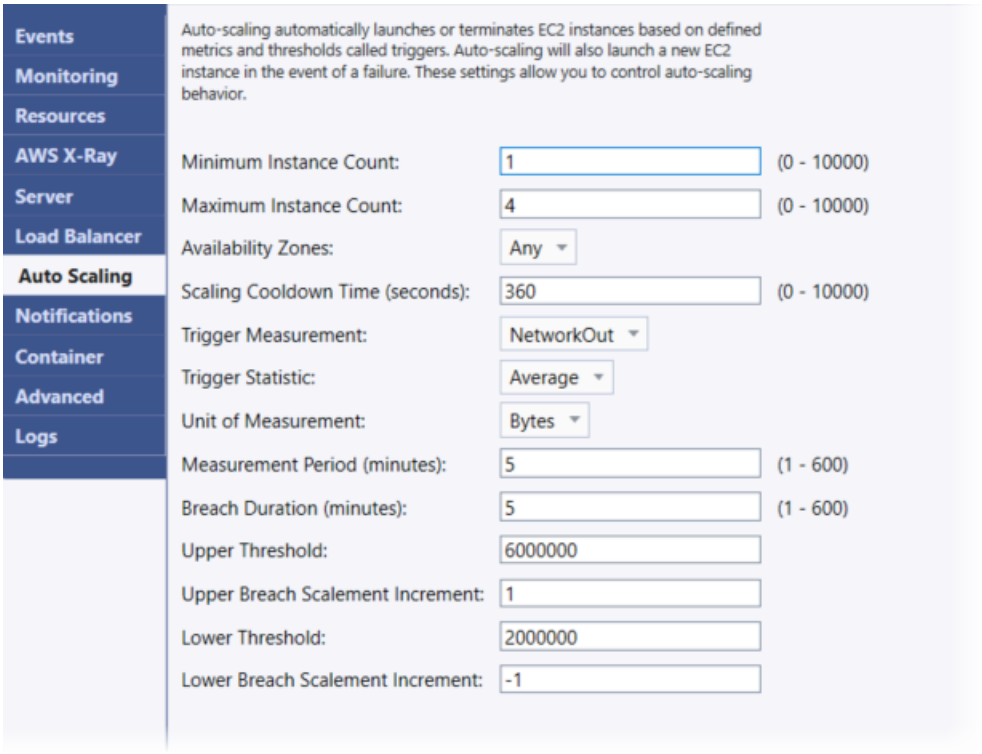
使用 AWS Toolkit for Visual Studio 設定 Auto Scaling

Amazon EC2 Auto Scaling 是一種 Amazon Web 服務，旨在根據使用者定義的觸發條件來自動啟動或終止 Amazon EC2 執行個體。您可設定「Auto Scaling 群組」並將「觸發條件」與這些群組建立關聯，藉此根據諸如頻寬使用量或 CPU 使用率等指標，自動擴展運算資源。Amazon EC2 Auto Scaling 可與 Amazon CloudWatch 搭配使用，以擷取執行應用程式之伺服器執行個體的指標。

Amazon EC2 Auto Scaling 可讓您取得一組 Amazon EC2 執行個體，並設定各種參數，讓此群組自動增減數量。Amazon EC2 Auto Scaling 可於該群組新增或移除 Amazon EC2 執行個體，協助您無縫處理應用程式的流量變更。

Amazon EC2 Auto Scaling 亦會針對其啟動的每個 Amazon EC2 執行個體，監控其運作狀態。如果有任何執行個體未預期終止，Amazon EC2 Auto Scaling 會偵測到終止狀況，並啟動替代執行個體。此功能可讓您自動維持所需的固定 Amazon EC2 執行個體數量。

Elastic Beanstalk 會為您的應用程式佈建 Amazon EC2 Auto Scaling。在 AWS Toolkit for Visual Studio 中的應用程式環境標籤，透過 Auto Scaling 標籤即可編輯 Elastic Beanstalk 環境的 Amazon EC2 執行個體組態。

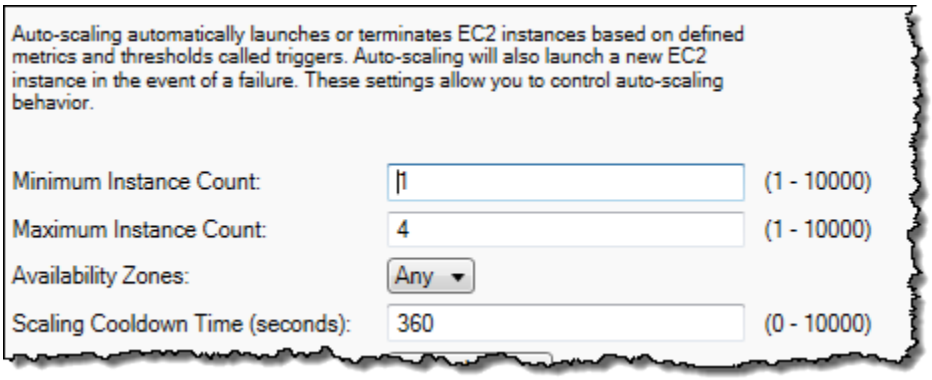


下列章節討論如何設定 Auto Scaling 參數供您的應用程式使用。

啟動組態

您可編輯啟動組態，以控制 Elastic Beanstalk 應用程式佈建 Amazon EC2 Auto Scaling 資源的方式。

Minimum Instance Count (執行個體計數下限) 與 Maximum Instance Count (執行個體計數上限) 方塊，可讓您指定您 Elastic Beanstalk 應用程式使用的 Auto Scaling 群組大小上下限。



> **Note**
>
> 欲保持固定數量的 Amazon EC2 執行個體，請將 Minimum Instance Count (執行個體計數下限) 和 Maximum Instance Count (執行個體計數上限) 設為相同值。

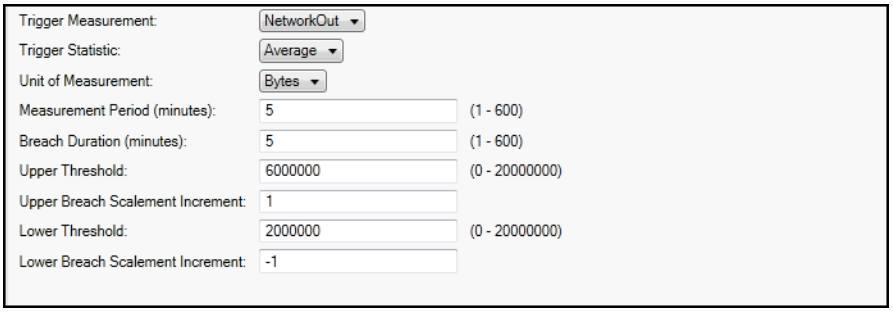
Availability Zones (可用區域) 方塊可讓您指定 Amazon EC2 執行個體所在可用區域數量。若您想要建構容錯應用程式，請務必設定此數值。若一個可用區域發生故障，您其他可用區域內的執行個體仍將繼續執行。

> **Note**
>
> 目前，您無法指定執行個體所在的可用區域。

觸發

「觸發條件」是您可加以設定的 Amazon EC2 Auto Scaling 機制，以通知系統您想要增加 (「擴展」) 或減少 (「縮減」) 執行個體數量的時機。您可設定觸發條件，在指標 (例如 CPU 使用率) 發佈至 Amazon CloudWatch 時「發動」，並判斷是否滿足您指定的條件。在特定期間內，若達到您為指標指定的條件閾值上下限，則觸發會啟動名為「擴展活動」的長時間執行程序。

您可透過 AWS Toolkit for Visual Studio，為 Elastic Beanstalk 應用程式定義擴展觸發條件。



Amazon EC2 Auto Scaling 觸發條件的工作方式是監控特定執行個體的特定 Amazon CloudWatch 指標。觸發條件包括 CPU 使用率、網路流量及磁碟活動。使用 Trigger Measurement (觸發條件測量指標) 設定以選取觸發的指標。

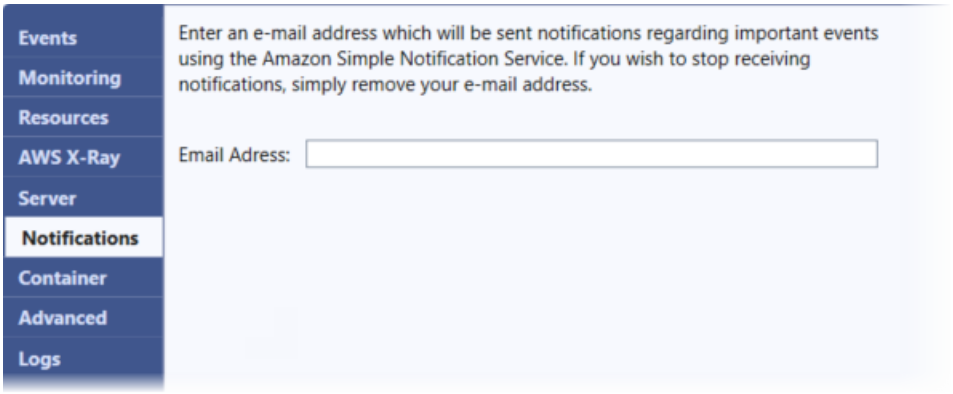
下列清單說明您可使用 AWS 管理主控台設定的觸發參數。

- 您可指定觸發應使用的統計資料。針對 Trigger Statistic (觸發條件統計資料)，您可選擇 Minimum (下限)、Maximum (上限)、Sum (總和) 或 Average (平均)。
- 在 Unit of Measurement (測量單位) 部分指定觸發條件測量的單位。
- Measurement Period (測量期間) 方塊的數值可指定 Amazon CloudWatch 測量觸發指標的頻率。在 Breach Duration (違規持續時間) 部分，您可定義指標在超過所定義的限制 (即 Upper Threshold (閾值上限) 和 Lower Threshold (閾值下限) 所指定) 後，引發觸發條件前的時間。
- 在 Upper Breach Scale Increment (上限違規規模調整增幅) 和 Lower Breach Scale Increment (下限違規規模調整增幅) 部分，可指定在進行擴展活動時，欲新增或移除的 Amazon EC2 執行個體數量。

如需 Amazon EC2 Auto Scaling 的詳細資訊，請參閱 Amazon Elastic Compute Cloud 文件中的「Amazon EC2 Auto Scaling」一節。

使用 AWS Toolkit for Visual Studio 設定通知

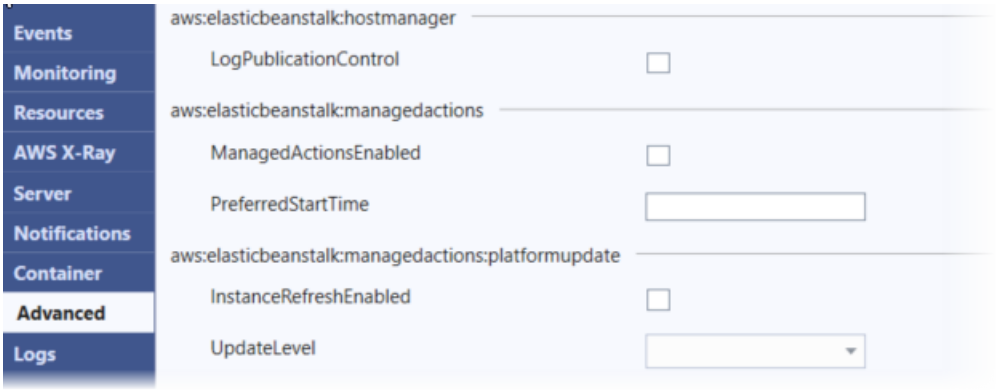
Elastic Beanstalk 使用 Amazon Simple Notification Service (Amazon SNS) 來通知您影響應用程式的重要事件。欲啟用 Amazon SNS 通知，只要在 Email Address (電子郵件地址) 方塊中輸入您的電子郵件地址即可。欲停用此通知，請將您的電子郵件地址自方塊移除。



使用 AWS Toolkit for Visual Studio 配置其他環境選項

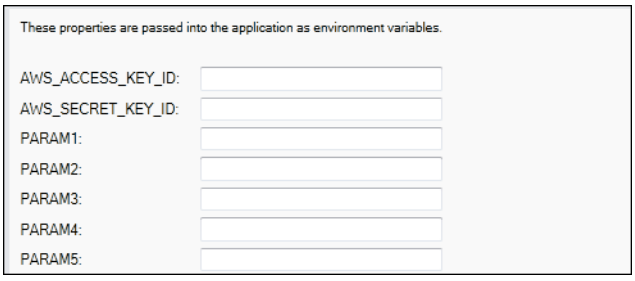
Elastic Beanstalk 定義大量組態選項，您可藉此設定環境的行為和其中的資源。組態選項會組織成像 `aws:autoscaling:asg` 的命名空間。每個命名空間都會定義環境 Auto Scaling 群組的選項。Advanced (進階) 面板會依照您可以在建立環境之後更新的字母順序，列出組態選項命名空間。

如需命名空間和選項的完整清單，以及各自的預設值和支援的值，請參閱 適用於所有環境的一般選項 和 Linux 上的 .NET Core 平台選項。



使用 AWS Toolkit for Visual Studio 配置 .NET Core 容器

Container (容器) 面板可讓您指定可從應用程式程式碼讀取的環境變數。



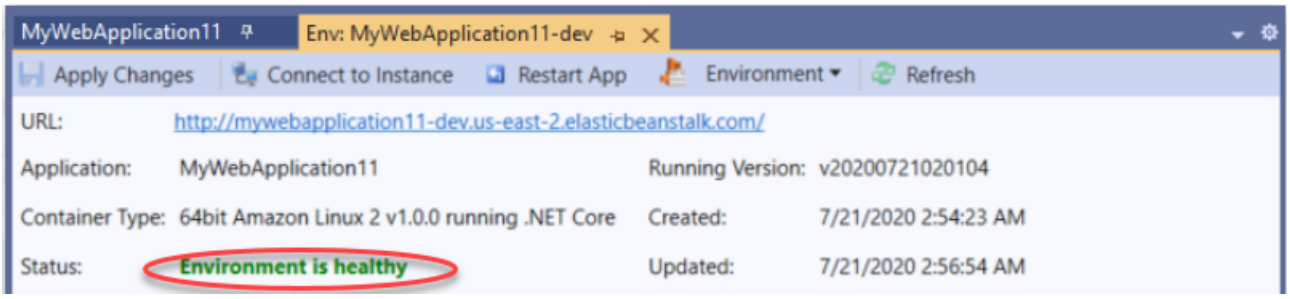
監控應用程式運作狀態

您有必要知道生產網站是否可用並會回應請求。Elastic Beanstalk 提供的功能可協助您監控應用程式的回應速度。它會監控應用程式的相關統計資料，並在超過閾值時發出警示。

如需 Elastic Beanstalk 提供的運作狀態監控詳細資訊，請參閱基礎型運作狀態報告。

您可以透過 AWS Toolkit for Visual Studio 或 AWS 管理主控台來存取您應用程式的相關運作資訊。

工具組會於 Status (狀態) 欄位顯示您環境的狀態和應用程式運作狀態。



欲監控應用程式運作狀態

1. 在 AWS Toolkit for Visual Studio 中的 AWS Explorer，展開 Elastic Beanstalk 節點，然後展開您的應用程式的節點。
2. 開啟應用程式環境的內容 (按一下滑鼠右鍵) 功能表，然後選取 View Status (檢視狀態)。
3. 在您的應用程式環境分頁中，選取 Monitoring (監控)。

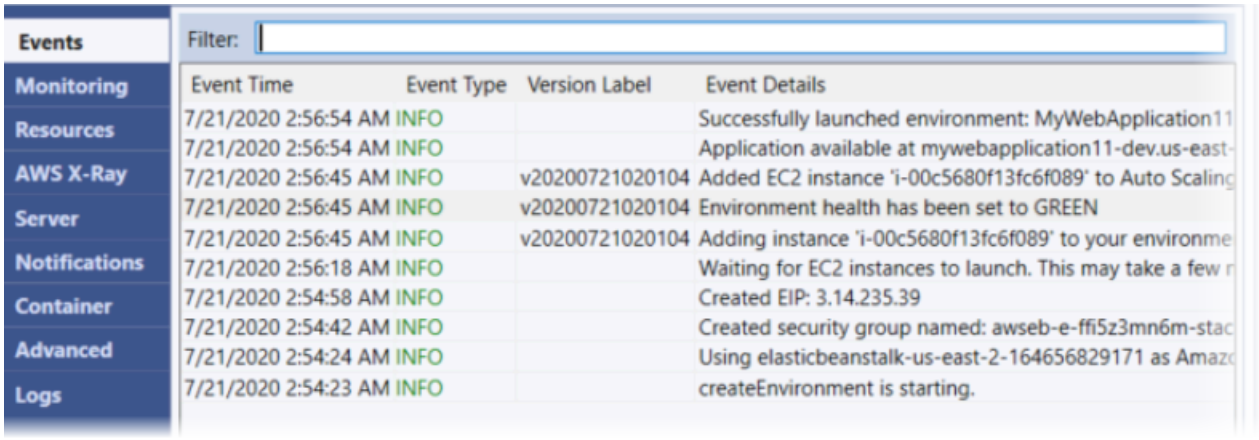
   Monitoring (監控) 面板包含一組圖表顯示您特定應用程式環境的資源使用情況。

> **Note**
>
> 時間範圍預設為最後一個小時。若要修改此設定，在 Time Range (時間範圍) 清單按一下不同的時間範圍。

您可使用 AWS Toolkit for Visual Studio 或 AWS 管理主控台來檢視與應用程式相關的事件。

檢視應用程式事件

1. 在 AWSToolkit for Visual Studio 中的 AWS Explorer，展開 Elastic Beanstalk 節點和您的應用程式節點。
2. 開啟應用程式環境的內容 (按一下滑鼠右鍵) 功能表，然後選取 View Status (檢視狀態)。
3. 在您的應用程式環境分頁中，選取 Events (事件)。



從 Windows Server 平台上的 .NET 遷移至 Linux 上的 .NET Core 平台

您可以將在 Windows Server 上的 .NET 平台上執行的應用程式遷移至 Linux 上的 .NET Core 平台。以下是從 Windows 遷移至 Linux 平台時的一些考量事項。

遷移至 Linux 上的 .NET Core 平台的考量事項

區域	變更和資訊
應用程式組態	在 Windows 平台上，您可以使用部署資訊清單來指定在環境中執行的應用程式。這些 Linux 上的 .NET Core 平台會使用 Procfile 指定在環境執行個體上執行的應用程式。如需綁定應用程式的詳細資訊，請參閱 the section called “綁定應用程式”。
代理伺服器	在 Windows 平台上，您可以使用 IIS 作為應用程式的代理伺服器。這些 Linux 上的 .NET Core 平台預設包含 nginx 做為反向代理程式。您可以選擇使用代理伺服器，並使用 Kestrel 作為您應用程式的 Web 伺服器。如需進一步了解，請參閱 the section called “代理伺服器”。
路由	在 Windows 平台上，您可以在應用程式碼中使用 IIS，並包含部署資訊清單來設定 IIS 路徑。對於 Linux 上的 .NET Core 平台，您可以在應用程式碼中使用 ASP .NET Core 路由，並更新您環境的 nginx 組態。如需進一步了解，請參閱 the section called “代理伺服器”。
日誌	Linux 和 Windows 平台會串流不同的日誌。如需詳細資訊，請參閱the section called “Elastic Beanstalk 如何設定 CloudWatch Logs”。

在 Elastic Beanstalk 上創建和部署 .NET 應用程序

> **查看 AWS 開發人員中心上的 .NET**
>
> 你有沒有停止我們的 .Net 開發人員中心？ 這是我們一站式服務所有項目 .NET 上 AWS。如需詳細資訊，請參閱開AWS 發人員中心上的 .NET。

AWS Elastic Beanstalk NET 可讓您更輕鬆地部署、管理和擴展使用亞馬遜網路服務的 ASP.NET 和 .NET 核心網路應用程式。本章提供建立、測試、部署和重新部署您的 Windows Web 應用程式至 Elastic Beanstalk 的指示。您可以使用彈性 Beanstalk 命令列介面 (EB CLI) 或使用彈性 BeanElastic Beanstalk 主控台，在短短幾分鐘內部署應用程式。

本章提供下列教學課程：

- QuickStart 對於. NET 核心

- 部署 ASP.NET 核心應用程式

如果您需要 Windows .NET 核心應用程式開發方面的協助，您可以前往幾個地方：

- .NET 開發論壇-發布您的問題並獲得反饋。
- .NET 開發人員中心 — 提供範例程式碼、文件、工具和其他資源的一站式服務。
- AWS 適用於 .NET 文件的 SDK — 閱讀有關設定 SDK 和執行程式碼範例、SDK 功能以及 SDK API 作業的詳細資訊。

> **Note**
>
> 此平台不支援以下的 Elastic Beanstalk 功能：
>
> - 工作者環境。如需詳細資訊，請參閱Elastic Beanstalk 工作者環境。
> - 套件日誌。如需詳細資訊，請參閱 檢視執行個體日誌。

主題

- 退休的 Elastic Beanstalk 視窗 2012 平台分支和 TLS 1.2 相容性
- QuickStart：將 Windows 應用程式上的 .NET 核心部署到 Elastic Beanstalk
- 教學課程：使用 Elastic Beanstalk 部署 ASP.NET 核心應用程式
- 設定您的 .NET 開發環境
- 使用 Elastic Beanstalk .NET 平台
- 將 Amazon RDS 資料庫執行個體新增到您的 .NET 應用程式環境
- AWS Toolkit for Visual Studio
- 將您的內部部署 .NET 應用程式遷移至 Elastic Beanstalk

退休的 Elastic Beanstalk 視窗 2012 平台分支和 TLS 1.2 相容性

如果您的應用程式目前正在淘汰的 Windows 伺服器 2012 R2 平台分支上執行，本主題會提供建議。它還解決了在我們的 AWS 服務 API 端點和受影響的平台分支上對 TLS 1.0 和 1.1 協議版本的已棄用支持。

Windows Server 2012 R2 平台分行已被淘汰

Elastic Beanstalk 於 2023 年 12 月 4 日淘汰視窗伺服器 2012 R2 平台分支機構，並於 2024 年 4 月 10 日將與這些平台相關聯的 AMI 設為私有。這個動作可防止在使用預設豆莖 AMI 的 Windows 伺服器 2012 年環境中啟動執行個體。

如果您有任何在淘汰的 Windows 平台分支上執行的環境，我們建議您將它們移轉至下列其中一個 Windows Server 平台，這些平台是目前且完全支援的：

- 視窗伺服器 2022 與 IIS 10.0 版本 2.x
- Windows Server 2019 搭配 IIS 10.0 版本 2.x

如需完整的遷移考量事項，請參閱從先前的 Windows Server 平台主要版本遷移。

如需有關平台版本棄用的詳細資訊，請參閱Elastic Beanstalk 平台支援政策。

> **Note**
>
> 如果您無法移轉至這些完全支援的平台，我們建議您使用以 Windows 伺服器 2012 R2 或 Windows 伺服器 2012 R2 核心 AMI 建立的自訂 AMI 做為基礎映像檔 (如果您尚未這樣做)。如需詳細說明，請參閱 保留已淘汰平台的 Amazon Machine Image (AMI) 存取權。如果您在執行以下AWS 其中一個遷移步驟時需要臨時存取 AMI，請聯絡 Support 中心。

TLS 1.2 相容性

自 2023 年 12 月 31 日 AWS 起，已開始在所有 AWS API 端點上完全強制執行 TLS 1.2。此動作移除了在所有 AWS API 上使用 TLS 版本 1.0 和 1.1 的功能。這些信息最初是在 2022 年 6 月 28 日傳達的。為了避免可用性影響的風險，請盡快將執行此處所識別之平台版本的任何環境升級至較新的版本 (如果您尚未這樣做)。

潛在影響

執行 TLS v1.1 或更早版本的 Elastic Beanstalk 平台版本會受到影響。此變更會影響包括但不限於下列各項的環境動作：組態部署、應用程式部署、auto 動擴展、新環境啟動、日誌輪替、增強型運作狀態報告，以及將應用程式日誌發佈到與應用程式相關聯的 Amazon S3 儲存貯體。

受影響的 Windows 平台版本

建議在下列平台版本上具有 Elastic Beanstalk 環境的客戶將其每個對應環境升級至 2022 年 2 月 18 日發行的 Windows 平台版本 2.8.3 或更新版本。

- Windows Server 2019 – 平台版本 2.8.2 或先前的版本

建議在下列平台版本上具有 Elastic Beanstalk 環境的客戶將其每個對應環境升級至 2022 年 12 月 28 日發行的 Windows 平台 2.10.7 版或更新版本。

- Windows Server 2016 – 平台版本 2.10.6 或先前的版本
- 視窗伺服器 2012 — 所有平台版本；此平台於 2023 年 12 月 4 日淘汰
- Windows Server 2008 — 所有平台版本；此平台於 2019 年 10 月 28 日被淘汰

如需最新和支援的 Windows Server 平台版本的清單，請參閱《AWS Elastic Beanstalk 平台指南》中的支援的平台。

如需更新環境的詳細資訊和最佳實務，請參閱 更新您 Elastic Beanstalk 環境的平台版本。

QuickStart：將 Windows 應用程式上的 .NET 核心部署到 Elastic Beanstalk

本 QuickStart 教程將引導您完成在 Windows 應用程序上創建 .NET 核心並將其部署到 AWS Elastic Beanstalk 環境的過程。

> **Note**
>
> 本 QuickStart 自學課程主要用於示範目的。請勿將本教學課程中建立的應用程式用於生產流量。

章節

- AWS 您應用程式的資源
- 後續步驟
- 使用 Elastic Beanstalk 控制台進行部署

您的 AWS 帳戶

如果您還不是 AWS 客戶，則需要創建一個 AWS 帳戶。註冊使您可以訪問 Elastic Beanstalk 和您需要的其他 AWS 服務。

如果您已經有 AWS 帳戶，則可以轉到必要條件。

創建一個 AWS 帳戶

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」。

必要條件

為了遵循本指南的程序，您需要命令列終端機或 shell 來執行命令。指令會顯示在清單中，前面有提示符號 (>) 和目前目錄的名稱 (如果適當)。

```
C:\eb-project> this is a command
this is output
```

EB CLI

本教學使用 Elastic Beanstalk 命令列界面 (EB CLI)。關於安裝和設定 EB CLI 的詳細資訊，請參閱安裝 EB CLI 和設定 EB CLI。

. NET 核心視窗

如果您的本機電腦上沒有安裝 .NET SDK，可以按照 .NET 文件網站上的下載 .NET 連結進行安裝。

執行下列命令來驗證您的 .NET SDK 安裝。

```
C:\> dotnet --info
```

第 1 步：在 Windows 應用程序上創建一個 .NET 核心

建立專案目錄。

```
C:\> mkdir eb-dotnetcore
C:\> cd eb-dotnetcore
```

接下來，通過運行以下命令創建一個示例你好世界 RESTful Web 服務應用程序。

```
C:\eb-dotnetcore> dotnet new web --name HelloElasticBeanstalk
C:\eb-dotnetcore> cd HelloElasticBeanstalk
```

步驟 2：在本機執行應用程式

執行下列命令以在本機執行應用程式。

```
C:\eb-dotnetcore\HelloElasticBeasntalk> dotnet run
```

輸出應該看起來像下面的文本。

```
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: https://localhost:7222
info: Microsoft.Hosting.Lifetime[14]
```

```
      Now listening on: http://localhost:5228
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: C:\Users\Administrator\eb-dotnetcore\HelloElasticBeanstalk
```

> **Note**
>
> 此指dotnet令會在本機執行應用程式時隨機選取連接埠。在此範例中，連接埠為 5228。當您將應用程式部署到 Elastic Beanstalk 環境時，應用程式會在連接埠 5000 上執行。

`http://localhost:`*`port`*在您的網頁瀏覽器中輸入 URL 位址。在此特定範例中，指令為`http://localhost:5228`。網絡瀏覽器應該顯示「你好世界！」。

步驟 3：使用 EB CLI 在 Windows 應用程式上部署您的 .NET 核心

執行下列命令，為此應用程式建立 Elastic Beanstalk 環境。

若要建立環境並在 Windows 應用程式上部署您的 .NET 核心

1. 在`HelloElasticBeanstalk`目錄中執行下列命令，以發佈和壓縮您的應用程式。

   ```
   C:\eb-dotnetcore\HelloElasticBeasntalk> dotnet publish -o site
   C:\eb-dotnetcore\HelloElasticBeasntalk> cd site
   C:\eb-dotnetcore\HelloElasticBeasntalk\site> Compress-Archive -Path * -
   DestinationPath ../site.zip
   C:\eb-dotnetcore\HelloElasticBeasntalk\site> cd ..
   ```

2. 使用以下內容在`HelloElasticBeanstalk`調`aws-windows-deployment-manifest.json`用中創建一個新文件：

   ```
   {
       "manifestVersion": 1,
       "deployments": {
           "aspNetCoreWeb": [
           {
               "name": "test-dotnet-core",
               "parameters": {
   ```

```
                "appBundle": "site.zip",
                "iisPath": "/",
                "iisWebSite": "Default Web Site"
            }
        }
        ]
    }
}
```

3. 透過 eb init 命令初始化您的 EB CLI 儲存庫。

```
C:\eb-dotnetcore\HelloElasticBeasntalk> eb init -p iis dotnet-windows-server-
tutorial --region us-east-2
```

此命令會建立名為的應用程式，dotnet-windows-server-tutorial並設定您的本機存放庫，以建立具有最新 Windows 伺服器平台版本的環境。

4. 使用 eb create 建立環境並於其中部署您的應用程式。Elastic Beanstalk 會自動為您的應用程序構建一個 zip 文件，並在端口 5000 上啟動它。

```
C:\eb-dotnetcore\HelloElasticBeasntalk> eb create dotnet-windows-server-env
```

Elastic Beanstalk 大約需要五分鐘的時間來創建您的環境。

第 4 步：在 Elastic Beanstalk 上運行應用程序

當創建環境的過程完成後，打開您的網站eb open。

```
C:\eb-dotnetcore\HelloElasticBeasntalk> eb open
```

恭喜您！您已經使用 Elastic Beanstalk 在 Windows 應用程序上部署了一個 .NET 核心！這會開啟瀏覽器視窗，並使用為應用程式建立的網域名稱。

步驟 5：清除

您可以在完成應用程式的工作後終止環境。Elastic Beanstalk 會終止與您環境相關的所有 AWS 資源。

若要使用 EB CLI 終止 Elastic Beanstalk 環境，請執行下列命令。

```
C:\eb-dotnetcore\HelloElasticBeasntalk> eb terminate
```

AWS 您應用程式的資源

您剛剛建立了單一執行個體應用程式。它可作為單一 EC2 執行個體的簡單範例應用程式使用，因此不需要負載平衡或 auto 擴展。對於單個實例應用程序，Elastic Beanstalk 創建以下 AWS 資源：

- EC2 執行個體 – 設定在您所選平台上執行 Web 應用程式的 Amazon EC2 虛擬機器。

  每個平台會執行不同一套軟體、設定檔和指令碼，來支援特定的語言版本、架構、Web 容器或其組合。大多數的平台會使用 Apache 或 nginx 做為反向代理，處理您 Web 應用程式前端的網路流量、向它轉送請求、提供靜態資產，並產生存取和錯誤日誌。
- 執行個體安全群組 – 設定允許從連接埠 80 傳入流量的 Amazon EC2 安全群組。此資源可讓負載平衡器傳來的 HTTP 傳輸資料，到達執行您 Web 應用程式的 EC2 執行個體。在預設情況下，不允許傳輸資料從其他通訊埠傳送。
- Amazon S3 儲存貯體 - 儲存位置，用來儲存當您使用 Elastic Beanstalk 時所建立的原始程式碼、日誌和其他成品。
- Amazon CloudWatch 警示 — 兩個 CloudWatch 警示，用於監控環境中執行個體的負載，並在負載過高或過低時觸發。當警示觸發時，您的 Auto Scaling 群組會擴展或縮減以進行回應。
- AWS CloudFormation 堆疊 — Elastic Beanstalk 用 AWS CloudFormation 來啟動環境中的資源並傳播組態變更。資源定義於範本中，您可在 AWS CloudFormation 主控台中檢視此範本。
- 網域名稱 – 會路由到您 Web 應用程式的網域名稱，其格式為 *subdomain*.*region*.elasticbeanstalk.com。

Elastic Beanstalk 會管理所有這些資源。當您終止環境時，Elastic Beanstalk 會終止其中的所有資源。

後續步驟

在您擁有執行應用程式的環境後，可以隨時部署應用程式的新版本或不同的應用程式。部署新的應用程式版本非常快速，因無須佈建或重新啟動 EC2 執行個體。您也可以使用 Elastic Beanstalk 控制台探索您的新環境。如需詳細步驟，請參閱本指南的「入門」一章中的「探索您的環境」。

在您部署一或兩個範例應用程式，並準備好開始在本機 Windows 應用程式上開發和執行 .NET Core 之後，請參閱 設定您的 .NET 開發環境

使用 Elastic Beanstalk 控制台進行部署

您也可以使用 Elastic Beanstalk 控制台來啟動示例應用程序。如需詳細步驟，請參閱本指南的「入門」一章中的「建立範例應用程式」。

教學課程：使用 Elastic Beanstalk 部署 ASP.NET 核心應用程式

在本教程中，您將逐步完成構建新的 ASP.NET 核心應用程序並將其部署到的過程 AWS Elastic Beanstalk。

首先，您將會使用 .NET Core 開發套件的 `dotnet` 命令列工具，來產生基本的 .NET Core 命令列應用程式、安裝相依的項目、編譯程式碼，然後在本機上執行應用程式。接著，您將會建立預設的 `Program.cs` 類別，並新增 ASP.NET `Startup.cs` 類別和組態檔案，以使用 ASP.NET 和 IIS 來建置處理 HTTP 請求的應用程式。

最後，Elastic Beanstalk 使用部署資訊清單針對 .NET Core 應用程式、自訂應用程式和單一伺服器上的多個 .NET Core 或 MSBuild 應用程式，進行部署設定。若要將 .NET Core 應用程式部署到 Windows Server 環境，請使用部署資訊清單，將網站封存檔加入應用程式的原始碼套件。`dotnet publish` 命令會產生編譯過的類別和相依項目，您可以將這些項目和 `web.config` 檔案一起組合，建立為網站封存檔。部署資訊清單會告知 Elastic Beanstalk 網站應執行的路徑，也可以用來設定應用程式集區，和執行位於不同路徑的多個應用程式。

源代碼可以在這裡找到：dotnet-core-windows-tutorial.zip

章節

必要條件

此教學課程使用 .NET Core 開發套件來產生基本的 .NET Core 應用程式、在本機執行該應用程式，並建置可部署的套件。

要求

- .NET Core (x64) 1.0.1、2.0.0 或更新版本

安裝 .NET Core 軟體開發套件

1. 從 microsoft.com/net/core 下載安裝程式。選擇 Windows。選擇 Download .NET SDK (下載 .NET 軟體開發套件)。
2. 執行安裝程式，並依照指示操作。

本教學課程使用命令列 ZIP 公用程式來建立您可部署至 Elastic Beanstalk 的原始碼套件。若要在 Windows 中使用 `zip` 指令，您可以安裝 `UnxUtils`，這是一些實用命令列公用程式 (例如 `zip` 和 `ls`) 的輕量套件。或者，您可以使用 Windows 檔案總管或其他任何 ZIP 公用程式，來製作原始碼套件的封存檔。

若要安裝 UnxUtils

1. 下載 `UnxUtils`.
2. 將封存項目擷取至本機目錄。例如，`C:\Program Files (x86)`。
3. 新增您 Windows PATH 使用者變數的二進位檔路徑。例如，`C:\Program Files (x86)\UnxUtils\usr\local\wbin`。
   a. 按下 Windows 鍵，然後輸入 **`environment variables`**。
   b. 選擇 Edit environment variables for your account (編輯您帳戶的環境變數)。
   c. 選擇 PATH，然後選擇 Edit (編輯)。
   d. 將路徑新增到變數值欄位中，以分號分隔。例如：*`C:\item1\path`*`;`***`C:\item2\path`***
   e. 選擇 OK (確定) 兩次以套用新的設定。
   f. 關閉任何正在執行的命令提示字元視窗，然後重新開啟命令提示字元視窗。
4. 開啟新的命令提示字元視窗，並執行 `zip` 命令以驗證其運作。

```
> zip -h
Copyright (C) 1990-1999 Info-ZIP
Type 'zip "-L"' for software license.
...
```

產生 .NET Core 專案

使用 `dotnet` 命令列工具來產生新的 C #.NET Core 專案，並在本機上執行此專案。預設的 .NET Core 應用程式是一項命令列公用程式，會在印出 `Hello World!` 之後結束。

產生新的 .NET Core 專案

1. 開啟新的命令提示視窗，然後瀏覽至您的使用者資料夾。

```
> cd %USERPROFILE%
```

2. 使用 dotnet new 指令來產生新的 .NET Core 專案。

```
C:\Users\username> dotnet new console -o dotnet-core-tutorial
Content generation time: 65.0152 ms
The template "Console Application" created successfully.
C:\Users\username> cd dotnet-core-tutorial
```

3. 使用 dotnet restore 指令來安裝相依項目。

```
C:\Users\username\dotnet-core-tutorial> dotnet restore
Restoring packages for C:\Users\username\dotnet-core-tutorial\dotnet-core-
tutorial.csproj...
Generating MSBuild file C:\Users\username\dotnet-core-tutorial\obj\dotnet-core-
tutorial.csproj.nuget.g.props.
Generating MSBuild file C:\Users\username\dotnet-core-tutorial\obj\dotnet-core-
tutorial.csproj.nuget.g.targets.
Writing lock file to disk. Path: C:\Users\username\dotnet-core-tutorial\obj
\project.assets.json
Restore completed in 1.25 sec for C:\Users\username\dotnet-core-tutorial\dotnet-
core-tutorial.csproj.

NuGet Config files used:
    C:\Users\username\AppData\Roaming\NuGet\NuGet.Config
    C:\Program Files (x86)\NuGet\Config\Microsoft.VisualStudio.Offline.config
Feeds used:
    https://api.nuget.org/v3/index.json
    C:\Program Files (x86)\Microsoft SDKs\NuGetPackages\
```

4. 使用 dotnet run 指令來建置應用程式，並在本機上執行。

```
C:\Users\username\dotnet-core-tutorial> dotnet run
Hello World!
```

啟動 Elastic Beanstalk 環境

使用 Elastic Beanstalk 主控台啟動 Elastic Beanstalk 環境。在這個範例中，您將會以 .NET 平台啟動。在啟動和設定環境之後，您可以隨時部署新的原始程式碼。

啟動環境 (主控台)

1. 使用這個預先配置的鏈接打開 Elastic Beanstalk 控制台：控制台。aws.amazon.com/彈性豆/家 # /新 applicationName = 教程和 environmentType = LoadBalanced
2. 在 Platform (平台)，選取符合您應用程式所使用語言的平台和平台分支。
3. 針對 Application code (應用程式程式碼)，選擇 Sample application (範例應用程式)。
4. 選擇 Review and launch (檢閱和啟動)。
5. 檢視可用選項。選擇您要使用的可用選項，當您準備就緒時，請選擇 Creat app (建立應用程式)。

建立環境約需 10 分鐘。在這段期間，您可以更新您的原始程式碼。

更新原始程式碼

將預設應用程式修改入使用 ASP.NET 和 IIS 的 web 應用程式。

- ASP.NET 是 .NET 的網站架構。
- IIS 是 Web 伺服器，會在您 Elastic Beanstalk 環境的 Amazon EC2 執行個體上執行應用程式。

下面的源代碼示例可以在這裡找到：dotnet-core-tutorial-source.zip

> **Note**
>
> 下列程序說明如何將專案程式碼轉換為 web 應用程式。若要簡化程序，您可以從頭開始將專案做為 web 應用程式產生。在前一節產生 .NET Core 專案中，將 `dotnet new` 步驟的命令修改成下列命令。
>
> ```
> C:\Users\username> dotnet new web -o dotnet-core-tutorial -n WindowsSampleApp
> ```

若要在您的程式碼中加入 ASP.NET 與 IIS 支援

1. 複製 `Program.cs` 到您的應用程式目錄，以當成 Web 主機產生器執行。

Example c:\users\username\dotnet-core-tutorial\ Program.cs

```
namespace Microsoft.AspNetCore.Hosting;
using WindowsSampleApp;

public static class Program
{
    public static void Main(string[] args)
    {
        CreateWebHostBuilder(args).Build().Run();
    }

    public static IWebHostBuilder CreateWebHostBuilder(string[] args) =>
        WebHost.CreateDefaultBuilder(args).UseStartup<Startup>();
}
```

2. 加入 `Startup.cs` 以執行 ASP.NET 網站。

   Example c:\users\username\dotnet-core-tutorial\ Startup.cs

   ```
   namespace WindowsSampleApp
   {
       public class Startup
       {
           public void Configure(IApplicationBuilder app)
           {
               app.UseRouting();
               app.UseEndpoints(endpoints =>
               {
                   endpoints.MapGet("/", () => "Hello World from Elastic Beanstalk");
               });
           }
       }
   }
   ```

3. 新增 `WindowsSampleApp.csproj`，其包含 IIS 中介軟體，並且包含 `web.config` 輸出的 `dotnet publish` 檔案。

> **Note**
>
> 以下範例是使用 .NET Core Runtime 2.2.1 發展出來的。您可能需要修改 `TargetFramework` 元素中的 `Version` 或 `PackageReference` 屬性數值，以符合您在自訂專案中所使用的 .NET Core Runtime 版本。

Example c:\users\username\dotnet-core-tutorial\ WindowsSampleApp .csproj

```
<Project Sdk="Microsoft.NET.Sdk.Web">

  <PropertyGroup>
    <TargetFramework>net6.0</TargetFramework>
    <RollForward>LatestMajor</RollForward>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
    <RootNamespace>WindowsSampleApp</RootNamespace>
  </PropertyGroup>

</Project>
```

接著，安裝新的相依項目，並在本機上執行 ASP.NET 網站。

若要在本機上執行網站

1. 使用 `dotnet restore` 指令來安裝相依項目。
2. 使用 `dotnet run` 指令來建置應用程式，並在本機上執行。
3. 開啟 localhost:5000 來檢視網站。

若要在 Web 伺服器上執行應用程式，您需要將編譯過的原始程式碼、`web.config` 組態檔案和執行時間的相依項目，一起封裝為套件。`dotnet` 工具提供了 `publish` 指令，此指令會根據 `dotnet-core-tutorial.csproj` 中的組態，來將這些檔案收集到目錄中。

若要建置您的網站

- 使用 `dotnet publish` 指令，來將編譯過的程式碼和相依項目輸出到名為 `site` 的資料夾。

```
C:\users\username\dotnet-core-tutorial> dotnet publish -o site
```

若要將應用程式部署到 Elastic Beanstalk，請將網站封存檔和部署資訊清單一起封存為套件。這會告訴 Elastic Beanstalk 如何執行它。

若要建立原始碼套件

1. 將網站資料夾中的檔案加入 ZIP 封存檔。

> **Note**
>
> 如果您使用不同的 ZIP 公用程式，請務必將所有檔案新增到產生 ZIP 壓縮檔的根資料夾中。這是成功將應用程式部署到您的 Elastic Beanstalk 環境的必要條件。

```
C:\users\username\dotnet-core-tutorial> cd site
C:\users\username\dotnet-core-tutorial\site> zip ../site.zip *
  adding: dotnet-core-tutorial.deps.json (164 bytes security) (deflated 84%)
  adding: dotnet-core-tutorial.dll (164 bytes security) (deflated 59%)
  adding: dotnet-core-tutorial.pdb (164 bytes security) (deflated 28%)
  adding: dotnet-core-tutorial.runtimeconfig.json (164 bytes security) (deflated
 26%)
  adding: Microsoft.AspNetCore.Authentication.Abstractions.dll (164 bytes security)
 (deflated 49%)
  adding: Microsoft.AspNetCore.Authentication.Core.dll (164 bytes security)
 (deflated 57%)
  adding: Microsoft.AspNetCore.Connections.Abstractions.dll (164 bytes security)
 (deflated 51%)
  adding: Microsoft.AspNetCore.Hosting.Abstractions.dll (164 bytes security)
 (deflated 49%)
  adding: Microsoft.AspNetCore.Hosting.dll (164 bytes security) (deflated 60%)
  adding: Microsoft.AspNetCore.Hosting.Server.Abstractions.dll (164 bytes security)
 (deflated 44%)
  adding: Microsoft.AspNetCore.Http.Abstractions.dll (164 bytes security) (deflated
 54%)
  adding: Microsoft.AspNetCore.Http.dll (164 bytes security) (deflated 55%)
  adding: Microsoft.AspNetCore.Http.Extensions.dll (164 bytes security) (deflated
 50%)
  adding: Microsoft.AspNetCore.Http.Features.dll (164 bytes security) (deflated
 50%)
```

```
 adding: Microsoft.AspNetCore.HttpOverrides.dll (164 bytes security) (deflated
49%)
 adding: Microsoft.AspNetCore.Server.IISIntegration.dll (164 bytes security)
(deflated 46%)
 adding: Microsoft.AspNetCore.Server.Kestrel.Core.dll (164 bytes security)
(deflated 63%)
 adding: Microsoft.AspNetCore.Server.Kestrel.dll (164 bytes security) (deflated
46%)
 adding: Microsoft.AspNetCore.Server.Kestrel.Https.dll (164 bytes security)
(deflated 44%)
 adding: Microsoft.AspNetCore.Server.Kestrel.Transport.Abstractions.dll (164 bytes
security) (deflated 56%)
 adding: Microsoft.AspNetCore.Server.Kestrel.Transport.Sockets.dll (164 bytes
security) (deflated 51%)
 adding: Microsoft.AspNetCore.WebUtilities.dll (164 bytes security) (deflated 55%)
 adding: Microsoft.Extensions.Configuration.Abstractions.dll (164 bytes security)
(deflated 48%)
 adding: Microsoft.Extensions.Configuration.Binder.dll (164 bytes security)
(deflated 47%)
 adding: Microsoft.Extensions.Configuration.dll (164 bytes security) (deflated
46%)
 adding: Microsoft.Extensions.Configuration.EnvironmentVariables.dll (164 bytes
security) (deflated 46%)
 adding: Microsoft.Extensions.Configuration.FileExtensions.dll (164 bytes
security) (deflated 47%)
 adding: Microsoft.Extensions.DependencyInjection.Abstractions.dll (164 bytes
security) (deflated 54%)
 adding: Microsoft.Extensions.DependencyInjection.dll (164 bytes security)
(deflated 53%)
 adding: Microsoft.Extensions.FileProviders.Abstractions.dll (164 bytes security)
(deflated 46%)
 adding: Microsoft.Extensions.FileProviders.Physical.dll (164 bytes security)
(deflated 47%)
 adding: Microsoft.Extensions.FileSystemGlobbing.dll (164 bytes security)
(deflated 49%)
 adding: Microsoft.Extensions.Hosting.Abstractions.dll (164 bytes security)
(deflated 47%)
 adding: Microsoft.Extensions.Logging.Abstractions.dll (164 bytes security)
(deflated 54%)
 adding: Microsoft.Extensions.Logging.dll (164 bytes security) (deflated 48%)
 adding: Microsoft.Extensions.ObjectPool.dll (164 bytes security) (deflated 45%)
 adding: Microsoft.Extensions.Options.dll (164 bytes security) (deflated 53%)
 adding: Microsoft.Extensions.Primitives.dll (164 bytes security) (deflated 50%)
 adding: Microsoft.Net.Http.Headers.dll (164 bytes security) (deflated 53%)
```

```
  adding: System.IO.Pipelines.dll (164 bytes security) (deflated 50%)
  adding: System.Runtime.CompilerServices.Unsafe.dll (164 bytes security) (deflated
 43%)
  adding: System.Text.Encodings.Web.dll (164 bytes security) (deflated 57%)
  adding: web.config (164 bytes security) (deflated 39%)
C:\users\username\dotnet-core-tutorial\site> cd ../
```

2. 加入部署資訊清單，此清單指向網站封存檔。

   Example c:\users\usernamedotnet-core-tutorial\aws-windows-deployment-manifest\

```
{
    "manifestVersion": 1,
    "deployments": {
        "aspNetCoreWeb": [
        {
            "name": "test-dotnet-core",
            "parameters": {
                "appBundle": "site.zip",
                "iisPath": "/",
                "iisWebSite": "Default Web Site"
            }
        }
        ]
    }
}
```

3. 使用 `zip` 指令來建立名為 `dotnet-core-tutorial.zip` 的原始碼套件。

```
C:\users\username\dotnet-core-tutorial> zip dotnet-core-tutorial.zip site.zip aws-
windows-deployment-manifest.json
  adding: site.zip (164 bytes security) (stored 0%)
  adding: aws-windows-deployment-manifest.json (164 bytes security) (deflated 50%)
```

部署您的應用程式

將原始碼套件部署到您建立的 Elastic Beanstalk 環境。

您可以在此處下載源代碼包：dotnet-core-tutorial-bundle.zip

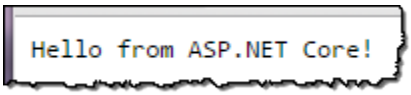
若要部署原始碼套件

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在環境概觀頁面上，選擇 Upload and deploy (上傳和部署)。
4. 使用畫面顯示對話方塊來上傳原始碼套件。
5. 選擇 Deploy (部署)。
6. 部署完成後，您可選擇網站的 URL 以在新分頁中開啟您的網站。

此應用程式只會將 `Hello from ASP.NET Core!` 寫入回應和傳回。



啟動環境來建立下列資源：

- EC2 執行個體 – Amazon Elastic Compute Cloud (Amazon EC2) 虛擬機器，已設為在您選擇的平台上執行 Web 應用程式。

  每個平台會執行特定的一套軟體、設定檔和指令碼，來支援特定的語言版本、架構、Web 容器或其組合。大多數的平台使用會 Apache 或 NGINX 做為反向代理，此反向代理會在您 Web 應用程式的前景執行、轉傳遞交給此 Web 應用程式的請求、提供靜態資產，並產生存取和錯誤日誌。
- 執行個體安全群組 - Amazon EC2 安全群組，已設為允許從連接埠 80 傳入的流量。此資源可讓負載平衡器傳來的 HTTP 傳輸資料，到達執行您 Web 應用程式的 EC2 執行個體。在預設情況下，不允許傳輸資料從其他通訊埠傳送。
- 負載平衡器 - Elastic Load Balancing 負載平衡器，可設定將請求分配到執行您應用程式的執行個體。負載平衡器也讓您的執行個體不需直接連接到網際網路。
- 負載平衡器安全群組 - Amazon EC2 安全群組，設為允許從連接埠 80 傳入的流量。此資源可讓來自網際網路的 HTTP 傳輸資料到達負載平衡器。在預設情況下，不允許傳輸資料從其他通訊埠傳送。
- Auto Scaling 群組 - Auto Scaling 群組，設為在執行個體終止或無法使用時，取代該執行個體。
- Amazon S3 儲存貯體 - 儲存位置，用來儲存當您使用 Elastic Beanstalk 時所建立的原始程式碼、日誌和其他成品。

- Amazon CloudWatch 警示 — 監控環境中執行個體負載的兩個 CloudWatch 警示，並在負載過高或過低時觸發警示。當警示觸發時，您的 Auto Scaling 群組會擴展或縮減以進行回應。
- AWS CloudFormation 堆疊 — Elastic Beanstalk 用 AWS CloudFormation 來啟動環境中的資源並傳播組態變更。資源定義於範本中，您可在 AWS CloudFormation 主控台中檢視此範本。
- 網域名稱 – 會路由到您 Web 應用程式的網域名稱，其格式為 *subdomain*.*region*.elasticbeanstalk.com。

> **Note**
>
> 為了增強 Elastic Beanstalk 應用程式的安全性，我們會在公共后缀列表 (PSL) 中註冊網域 elasticbeanstalk.com。為了加強安全性，如果您需要在 Elastic Beanstalk 應用程式的預設網域名稱中設定敏感性 Cookie，我們建議您使用具 `__Host-` 前置詞的 Cookie。此做法將有助於保護您的網域免受跨站請求偽造 (CSRF) 攻擊。如需更多資訊，請參閱 Mozilla 開發人員網路中的設定 Cookie 頁面。

這些資源全都由 Elastic Beanstalk 管理。當您終止環境時，Elastic Beanstalk 會終止其中的所有資源。

> **Note**
>
> Elastic Beanstalk 建立的 Amazon S3 儲存貯體共享於環境間，且不會於環境終止時刪除。如需更多詳細資訊，請參閱 將 Elastic Beanstalk 與 Amazon S3 搭配使用。

清除

當您完成使用 Elastic Beanstalk 時，即可終止您的環境。Elastic Beanstalk 會終止與您的環境相關聯的所有 AWS 資源，例如 Amazon EC2 執行個體、資料庫執行個體、負載平衡器、安全群組和警示。

從控制台終止 Elastic Beanstalk 環境

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 選擇 Actions (動作)，然後選擇 Terminate Environment (終止環境)。
4. 使用畫面顯示對話方塊來確認環境終止。

您可以使用 Elastic Beanstalk，隨時輕鬆地為您的應用程式建立新環境。

後續步驟

隨著您繼續開發應用程式，可能會希望無須手動建立 .zip 檔案並將其上傳至 Elastic Beanstalk 主控台，即可管理環境和部署應用程式。E lastic Beanstalk 命令列介面 (EB CLI) 提供從命令列建立、設定和部署應用程式至 Elastic Beanstalk 環境的 easy-to-use 指令。

如果您使用 Visual Studio 來開發應用程式，您也可以使用 AWS Toolkit for Visual Studio 來部署已變更、管理 Elastic Beanstalk 環境，以及管理其他 AWS 資源。如需詳細資訊，請參閱「AWS Toolkit for Visual Studio」。

進行開發和測試時，建議您使用 Elastic Beanstalk 的功能，將受管的資料庫執行個體直接加入您的環境。如需於環境中設定資料庫的說明，請參閱將資料庫新增至您的 Elastic Beanstalk 環境。

最後，如果您打算在生產環境中使用您的應用程式，請為您的環境設定自訂的網域名稱，並啟用 HTTPS 以進行安全連線。

設定您的 .NET 開發環境

設定 .NET 開發環境於本機測試您的應用程式，之後再部署至 AWS Elastic Beanstalk。此主題概述開發環境設定步驟，並提供實用工具的安裝頁面連結。

如需了解適用所有語言的常見設定步驟和工具，請參閱設定您的開發機器搭配 Elastic Beanstalk 使用。

章節

- 安裝 IDE
- 安裝 AWS Toolkit for Visual Studio

若您需要從應用程式內管理 AWS 資源，請安裝適用於 .NET 的 AWS 開發套件。例如，您可以使用 Amazon S3 來存放並擷取資料。

透過適用於 .NET 的 AWS 開發套件，您可以在幾分鐘內開始使用可下載的單一套件，其中包括 Visual Studio 專案範本、AWS .NET 程式庫、C# 程式碼範例和文件。關於使用程式庫來建置應用程式的方式，則提供 C# 的實務範例。另亦提供線上影片教學課程和參考文件，協助您學習使用程式庫和程式碼範例。

如需詳細資訊及安裝說明，請造訪適用於 .NET 的 AWS 開發套件首頁。

安裝 IDE

整合開發環境 (IDE) 提供可加速應用程式開發的各種功能。若您沒有使用 IDE 進行 .NET 開發作業的經驗，請嘗試透過 Visual Studio Community 入門。

請造訪 Visual Studio Community 頁面，下載並安裝 Visual Studio Community。

安裝 AWS Toolkit for Visual Studio

AWS Toolkit for Visual Studio 是 Visual Studio IDE 的開源外掛程式，可讓開發人員使用 AWS 更輕鬆開發、除錯與部署 .NET 應用程式。請造訪 Toolkit for Visual Studio 首頁，取得安裝說明。

使用 Elastic Beanstalk .NET 平台

AWS Elastic Beanstalk 支援多種不同版本的 .NET 程式設計架構和 Windows 伺服器的平台。如需完整清單，請參閱 AWS Elastic Beanstalk 平台文件中的具備 IIS 的 Windows Server 上的 .NET。

Elastic Beanstalk 提供組態選項，您可用其於 Elastic Beanstalk 環境中自訂 EC2 執行個體上執行的軟體。您可以設定應用程式所需的環境變數，啟用日誌輪換至 Amazon S3，並進行 .NET Framework 設定。

Elastic Beanstalk 主控台中提供了修改正在執行環境組態的組態選項。要避免在終止環境的組態時遺失組態，您可以使用已儲存組態來儲存您的設定，並在之後套用至另一個環境。

若要將設定儲存於原始程式碼，您可以包含組態檔案。每次您建立環境或部署應用程式，組態檔案裡的設定就會套用。您也可以使用組態檔案來安裝套件、執行指令碼，並在部署期間執行其他執行個體自訂操作。

在 Elastic Beanstalk 主控台中套用的設定會覆寫組態檔案中相同的設定 (如存在)。這可讓您在組態檔案中擁有預設設定，並以主控台的環境專屬設定覆寫之。如需優先順序以及其他變更設定方法的詳細資訊，請參閱組態選項。

在 Elastic Beanstalk 主控台中設定 .NET 環境

您可以使用 Elastic Beanstalk 主控台來啟用至 Amazon S3 的日誌輪換，設定您的應用程式可以從環境讀取的變數，並變更 .NET Framework 設定。

在 Elastic Beanstalk 主控台中設定 .NET 環境

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

   > **Note**
   >
   > 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇組態。
4. 在更新、監控和日誌記錄組態類別中，選擇編輯。

容器選項

- Target .NET runtime (目標 .NET 執行時間) – 設定為 `2.0` 來執行 CLR v2。
- Enable 32-bit applications (啟用 32 位元應用程式) – 設定為 `True` 來執行 32 位元應用程式。

日誌選項

Log Options (日誌選項) 區段有兩個設定：

- 執行個體設定檔 – 指定有權存取與您應用程式相關的 Amazon S3 儲存貯體的執行個體設定檔。
- Enable log file rotation to Amazon S3 (啟用 Amazon S3 的日誌檔案輪換) – 指定是否將應用程式 Amazon EC2 執行個體的日誌檔案複製到與應用程式關聯的 Amazon S3 儲存貯體。

環境屬性

Environment Properties (環境屬性) 的部分可讓您針對執行您應用程式的 Amazon EC2 執行個體，來指定其上的環境資訊設定。這些設定會以金鑰值對的形式傳到應用程式。使用 `System.GetEnvironmentVariable` 來讀取這些值。相同金鑰可以同時存在於 `web.config` 中及當作環境屬性。使用 `System.Configuration` 命名空間來讀取 `web.config` 中的數值。

```
NameValueCollection appConfig = ConfigurationManager.AppSettings;
```

```
string endpoint = appConfig["API_ENDPOINT"];
```

如需詳細資訊，請參閱「環境屬性與其他軟體設定」。

aws:elasticbeanstalk:container:dotnet:apppool 命名空間

您可以使用組態檔案來設定組態選項，並在部署期間執行其他的執行個體設定工作。組態選項可由 Elastic Beanstalk 服務或您使用的平台來定義，並且會組織成「命名空間」。

.NET 平台於 `aws:elasticbeanstalk:container:dotnet:apppool` 命名空間內定義的選項，可用來設定 .NET 執行時間。

下列範例組態檔案顯示此命名空間可用的各個選項的設定：

Example .ebextensions/dotnet-settings.config

```
option_settings:
  aws:elasticbeanstalk:container:dotnet:apppool:
    Target Runtime: 2.0
    Enable 32-bit Applications: True
```

Elastic Beanstalk 可提供許多組態選項讓您自訂環境。除了組態檔案，您也可以使用主控台、已儲存組態、EB CLI 或 AWS CLI來設定組態選項。如需詳細資訊，請參閱「組態選項」。

遷移 Elastic Beanstalk Windows Server 平台的主要版本

AWS Elastic Beanstalk 有它的 Windows 服務器平台的幾個主要版本。此頁面涵蓋了每個主要版本的主要改善，以及在您遷移至更新版本前應考量的事項。

Windows Server 平台目前的版本為第 2 版 (v2)。若您的應用程式使用任何 v2 之前的 Windows Server 平台版本，我們建議您遷移至 v2。

Windows Server 平台主要版本中的新功能

Windows Server 平台 V2

Elastic Beanstalk Windows Server 平台的第 2 版 (v2) 已在 2019 年 2 月發行。V2 透過數種重要方式，讓 Windows Server 平台的行為與 Elastic Beanstalk Linux 類型平台的行為更為接近。V2 可完全與舊版 v1 相容，這使得從 v1 遷移的過程更為容易。

Windows Server 平台目前支援以下功能：

- 版本控制 – 每個版本都會取得新的版本號碼，而您可以在建立和管理環境時參考以前的版本 (您可以使用的版本)。
- 增強式運作狀態 – 如需詳細資訊，請參閱增強型運作狀態報告與監控。
- 不可變部署和以額外批次進行滾動部署 – 如需部署原則的詳細資訊，請參閱將應用程式部署至 Elastic Beanstalk 環境。
- 不可變更新 – 如需更新類型的詳細資訊，請參閱組態變更。
- 受管平台更新 – 如需詳細資訊，請參閱受管平台更新。

> **Note**
>
> 新的部署和更新功能依賴增強式運作狀態。請啟用增強式運作狀態以使用他們。如需詳細資訊，請參閱啟用 Elastic Beanstalk 增強型運作狀態報告。

Windows Server 平台 V1

Elastic Beanstalk Windows Server 平台的第 1.0.0 版 (v1) 已在 2015 年 10 月發行。此版本變更了 Elastic Beanstalk 在建立環境與更新期間，處理組態檔案中命令的順序。

先前的平台版本，在解決方案堆疊名稱中並未包含版本號碼：

- 執行 IIS 8.5 的 64 位元 Windows Server 2012 R2
- 執行 IIS 8.5 的 64 位元 Windows Server Core 2012 R2
- 執行 IIS 8 的 64 位元 Windows Server 2012
- 執行 IIS 7.5 的 64 位元 Windows Server 2008 R2

在先前的版本中，組態檔案的處理順序並不一致。當環境建立時，`Container Commands` 會在應用程式原始碼部署到 IIS 後執行。部署到執行中的環境時，容器指令會在新版本部署之前執行。在進行擴展時，則完全不會處理組態檔案。

除此之外，IIS 會在容器指令執行之前啟動。因此，一些客戶在容器的命令中實作了因應措施，讓 IIS 伺服器在命令執行之前先暫停，等命令執行完成後再次啟動。

第 1 版修正了不一致性，使 Windows Server 平台的行為與 Elastic Beanstalk Linux 類型平台的行為更為接近。在 v1 平台中，Elastic Beanstalk 永遠都會先執行容器命令，再啟動 IIS 伺服器。

v1 平台解決方案堆疊在 Windows Server 版本的後方加上了 v1：

- 執行 IIS 8.5 的 64 位元 Windows Server 2012 R2 1.1.0 版
- 執行 IIS 8.5 的 64 位元 Windows Server Core 2012 R2 1.1.0 版
- 執行 IIS 8 的 64 位元 Windows Server 2012 1.1.0 版
- 執行 IIS 7.5 的 64 位元 Windows Server 2008 R2 1.1.0 版

此外，v1 平台會先將您應用程式來源套件的內容，解壓縮到 `C:\staging\`，再執行容器命令。在容器命令執行完成後，此資料夾的內容會壓縮成 .zip 檔案，然後部署到 IIS。此工作流程可讓您在進行部署之前，先用命令或指令碼來修改應用程式來源套件的內容。

從先前的 Windows Server 平台主要版本遷移

請在更新環境前，先閱讀本節的遷移考量事項。若要將您環境的平台更新至更新的版本，請參閱更新您 Elastic Beanstalk 環境的平台版本。

從 V1 到 V2

Windows Server 平台 v2 不支援 .NET Core 1.x 及 2.0。若您要將應用程式從 Windows Server v1 遷移至 v2，而您的應用程式使用了其中一個 .NET Core 版本，請將您的應用程式更新至 v2 支援的 .NET Core 版本。如需支援版本的清單，請參閱 AWS Elastic Beanstalk 平台中的具備 IIS 的 Windows Server 上的 .NET。

如果您的應用程式使用自訂的 Amazon Machine Image (AMI)，請根據 Windows Server 平台 v2 AMI 建立新自訂 AMI。如需進一步了解，請參閱使用自訂的 Amazon Machine Image (AMI)。

> **Note**
>
> Windows Server v2 的部署和更新功能依賴增強式運作狀態。當您將環境遷移至 v2 時，會停用增強式運作狀態。請啟用它以使用這些功能。如需詳細資訊，請參閱啟用 Elastic Beanstalk 增強型運作狀態報告。

從 V1 之前的版本

除了從 v1 遷移的考量事項，若您要從 v1 版本之前的 Windows Server 解決方案堆疊遷移應用程式，而您目前使用容器命令，請在遷移至更新版本時，移除任何您新增至其中，做為處理過程不一致性因應措施的命令。從 v1 開始，容器命令保證會在部署應用程式來源及啟動 IIS 前完全執行。這可讓您對 `C:\staging` 中的來源進行任何變更，並在此步驟期間修改 IIS 組態檔案，而不會發生任何問題。

例如，您可以使用 AWS CLI 將 DLL 檔案從 Amazon S3 下載到您的應用程式來源：

`.ebextensions\copy-dll.config`

```
container_commands:
  copy-dll:
    command: aws s3 cp s3://DOC-EXAMPLE-BUCKET/dlls/large-dll.dll .\lib\
```

如需有關使用組態檔案的詳細資訊，請參閱使用組態檔案 (.ebextensions) 來進行進階的環境自訂。

透過部署資訊清單執行多個應用程式和 ASP.NET 核心應用程式

您可使用部署資訊清單來指示 Elastic Beanstalk 如何部署您的應用程式。使用這個方法，您不需要用 `MSDeploy` 來產生在網站根路徑執行的單一 ASP.NET 應用程式的原始碼套件。相反地，您可以使用資訊清單檔案，在不同的路徑上執行多個應用程式。或者，您也可以告訴 Elastic Beanstalk 用 ASP.NET Core 部署和執行應用程式。您亦可使用部署資訊清單來設定應用程式集區，在其中執行您的應用程式。

部署資訊清單會將 .NET Core 應用程式適用的支援新增至 Elastic Beanstalk。您可以部署沒有部署資訊清單的 .NET Framework 應用程式。不過，.NET Core 應用程式需要部署資訊清單，才能在 Elastic Beanstalk 上執行。使用部署資訊清單時，請為每個應用程式建立網站封存檔，然後將此封存檔納入內含部署資訊清單的第二個 ZIP 封存檔。

部署資訊清單亦增加執行不同路徑的多個應用程式的能力。部署資訊清單會定義一系列部署目標，每個目標都有網站封存檔與 IIS 應執行資訊清單的路徑。例如，您可於 `/api` 路徑執行 Web API 處理非同步請求，並於使用 API 的根路徑執行 Web 應用程式。

您也可以使用部署資訊清單來使用 IIS 或 Kestrel 中的應用程式集區執行多個應用程式。您可將應用程式集區設定為定期重新啟動您的應用程式、執行 32 位元應用程式，或使用特定版本的 .NET Framework 執行時間。

如需完全自訂，您可以在 Windows 中撰寫自己的部署指令碼， PowerShell 並告知 Elastic Beanstalk 要執行哪些指令碼以安裝、解除安裝和重新啟動應用程式。

部署資訊清單和相關功能需要 Windows Server 平台 1.2.0 版或更新版本。

章節

- .NET Core 應用程式

- 執行多個應用程式
- 設定應用程式集區
- 定義自訂部署

.NET Core 應用程式

您可使用部署資訊清單，在 Elastic Beanstalk 上執行 .NET Core 應用程式。.NET Core 是跨平台版本的 .NET，隨附命令列工具 (dotnet)。您可以使用它來產生應用程式、在本機執行，並準備發佈。

> **Note**
>
> 如需使用部署資訊清單，在 Elastic Beanstalk 上執行 .NET Core 應用程式的教學課程與範例應用程式，請參閱 教學課程：使用 Elastic Beanstalk 部署 ASP.NET 核心應用程式。

欲於 Elastic Beanstalk 上執行 .NET Core 應用程式，您可以執行 `dotnet publish` 並將輸出納入 ZIP 封存檔，且不要納入任何其中的目錄。請將網站封存檔與具備類型為 `aspNetCoreWeb` 之部署目標的部署資訊清單，一同放置於原始碼套件。

下列部署資訊清單於根路徑執行的 .NET 核心應用程式，來自名為 `dotnet-core-app.zip` 的網站封存檔。

Example aws-windows-deployment-manifest. JSON-. NET 核心

```
{
  "manifestVersion": 1,
  "deployments": {
    "aspNetCoreWeb": [
      {
        "name": "my-dotnet-core-app",
        "parameters": {
          "archive": "dotnet-core-app.zip",
          "iisPath": "/"
        }
      }
    ]
  }
}
```

將資訊清單和網站封存檔打包至 ZIP 封存檔來建立原始碼套件。

Example dotnet-core-bundle. 拉鍊

```
.
|-- aws-windows-deployment-manifest.json
`-- dotnet-core-app.zip
```

網站封存檔內含編譯的應用程式程式碼、依存項目和 `web.config` 檔案。

Example dotnet-core-app. 拉鍊

```
.
|-- Microsoft.AspNetCore.Hosting.Abstractions.dll
|-- Microsoft.AspNetCore.Hosting.Server.Abstractions.dll
|-- Microsoft.AspNetCore.Hosting.dll
|-- Microsoft.AspNetCore.Http.Abstractions.dll
|-- Microsoft.AspNetCore.Http.Extensions.dll
|-- Microsoft.AspNetCore.Http.Features.dll
|-- Microsoft.AspNetCore.Http.dll
|-- Microsoft.AspNetCore.HttpOverrides.dll
|-- Microsoft.AspNetCore.Server.IISIntegration.dll
|-- Microsoft.AspNetCore.Server.Kestrel.dll
|-- Microsoft.AspNetCore.WebUtilities.dll
|-- Microsoft.Extensions.Configuration.Abstractions.dll
|-- Microsoft.Extensions.Configuration.EnvironmentVariables.dll
|-- Microsoft.Extensions.Configuration.dll
|-- Microsoft.Extensions.DependencyInjection.Abstractions.dll
|-- Microsoft.Extensions.DependencyInjection.dll
|-- Microsoft.Extensions.FileProviders.Abstractions.dll
|-- Microsoft.Extensions.FileProviders.Physical.dll
|-- Microsoft.Extensions.FileSystemGlobbing.dll
|-- Microsoft.Extensions.Logging.Abstractions.dll
|-- Microsoft.Extensions.Logging.dll
|-- Microsoft.Extensions.ObjectPool.dll
|-- Microsoft.Extensions.Options.dll
|-- Microsoft.Extensions.PlatformAbstractions.dll
|-- Microsoft.Extensions.Primitives.dll
|-- Microsoft.Net.Http.Headers.dll
|-- System.Diagnostics.Contracts.dll
|-- System.Net.WebSockets.dll
|-- System.Text.Encodings.Web.dll
|-- dotnet-core-app.deps.json
|-- dotnet-core-app.dll
|-- dotnet-core-app.pdb
```

```
|-- dotnet-core-app.runtimeconfig.json
`-- web.config
```

請參閱教學課程的完整範例。

執行多個應用程式

您可定義多個部署目標，藉此透過部署資訊清單執行多個應用程式。

以下部署資訊清單設定兩個 .NET Core 應用程式。該WebApiSampleApp應用程序實現了一個簡單的 Web API，並在/api路徑上提供異步請求。DotNetSampleApp 應用程式是一種 Web 應用程式，其在根路徑處理請求。

Example aws-windows-deployment-manifest.json-多個應用程序

```
{
  "manifestVersion": 1,
  "deployments": {
    "aspNetCoreWeb": [
      {
        "name": "WebAPISample",
        "parameters": {
          "appBundle": "WebApiSampleApp.zip",
          "iisPath": "/api"
        }
      },
      {
        "name": "DotNetSample",
        "parameters": {
          "appBundle": "DotNetSampleApp.zip",
          "iisPath": "/"
        }
      }
    ]
  }
}
```

此處提供具備多個應用程式的範例應用程式：

- 可部署的來源套裝軟體 --v2.zip dotnet-multiapp-sample-bundle
- 源代碼 dotnet-multiapp-sample-source-v2.zip

設定應用程式集區

您可以在 Windows 環境中支援多個應用程式。可採用兩種方法：

- 您可以將 out-of-process 託管模型與 Kestrel Web 服務器一起使用。使用此模型，您可以將多個應用程式設定為在一個應用程式集區中執行。
- 您可以使用進程內託管模型。憑藉此模型，您可以使用多個應用程式集區來執行多個應用程式，且每個集區中只有一個應用程式。如果您使用的是 IIS 伺服器，並且需要執行多個應用程式，則必須使用此方法。

若要設定 Kestrel 在一個應用程式集區中執行多個應用程式，請新增 `hostingModel="OutofProcess"` 檔案中的 `web.config`。請考慮下列範例。

Example 網絡配置-紅 out-of-process 隼託管模型

```
<configuration>
<location path="." inheritInChildApplications="false">
<system.webServer>
<handlers>
<add
    name="aspNetCore"
    path="*" verb="*"
    modules="AspNetCoreModuleV2"
    resourceType="Unspecified" />
</handlers>
<aspNetCore
    processPath="dotnet"
    arguments=".\CoreWebApp-5-0.dll"
    stdoutLogEnabled="false"
    stdoutLogFile=".\logs\stdout"
    hostingModel="OutofProcess" />
</system.webServer>
</location>
</configuration>
```

Example aws-windows-deployment-manifest.json-多個應用程序

```
{
"manifestVersion": 1,
  "deployments": {"msDeploy": [
      {"name": "Web-app1",
```

```
        "parameters": {"archive": "site1.zip",
          "iisPath": "/"
        }
      },
      {"name": "Web-app2",
        "parameters": {"archive": "site2.zip",
          "iisPath": "/app2"
        }
      }
    ]
  }
}
```

IIS 不支援一個應用程式集區中有多個應用程式，因為其使用進程內託管模型。因此，您需要透過將每個應用程式指派給一個應用程式集區，來設定多個應用程式。換句話說，只將一個應用程式指派給一個應用程式集區。

您可以將 IIS 設定為在 aws-windows-deployment-manifest.json 檔案中使用不同的應用程式集區。在您參考下一個範例檔案時，請進行下列更新：

- 新增包含名稱為 iisConfig 子區段的 appPools 區段。
- 在 appPools 區塊中，列出應用程式集區。
- 在 deployments 區段中，為每個應用程式定義 parameters 區段。
- 針對每個應用程式，parameters 區段會指定封存檔、執行它的路徑，以及要在其中執行的 appPool。

下列部署資訊清單會設定每 10 分鐘重新啟動應用程式的兩個應用程式集區。它們還將其應用程式附加到在指定路徑上執行的 .NET Framework Web 應用程式。

Example aws-windows-deployment-manifest.json-每個應用程序池一個應用程序

```
{
"manifestVersion": 1,
  "iisConfig": {"appPools": [
      {"name": "MyFirstPool",
       "recycling": {"regularTimeInterval": 10}
      },
      {"name": "MySecondPool",
       "recycling": {"regularTimeInterval": 10}
      }
```

```
      ]
    },
  "deployments": {"msDeploy": [
      {"name": "Web-app1",
        "parameters": {
           "archive": "site1.zip",
           "iisPath": "/",
           "appPool": "MyFirstPool"
           }
      },
      {"name": "Web-app2",
        "parameters": {
           "archive": "site2.zip",
           "iisPath": "/app2",
           "appPool": "MySecondPool"
          }
      }
     ]
    }
}
```

定義自訂部署

如需進一步控制，您可定義自訂部署，藉此完全自訂應用程式部署。

下列部署資訊清單告知 Elastic Beanstalk 執行名稱為 `install` 的 `siteInstall.ps1` 指令碼。此指令碼會在執行個體啟動和部署期間安裝網站。除此之外，部署資訊清單還會告知 Elastic Beanstalk 在部署期間安裝新版本之前執行`uninstall`指令碼，並在 AWS 管理主控台中選擇「重新啟動應用程式伺服器」時，`restart`指令碼重新啟動應用程式。

Example aws-windows-deployment-manifest.json-自定義部署

```
{
  "manifestVersion": 1,
  "deployments": {
    "custom": [
      {
        "name": "Custom site",
        "scripts": {
          "install": {
            "file": "siteInstall.ps1"
          },
          "restart": {
```

```
            "file": "siteRestart.ps1"
          },
          "uninstall": {
            "file": "siteUninstall.ps1"
          }
        }
      }
    ]
  }
}
```

將執行應用程式所需的成品納入具備資訊清單和指令碼的原始碼套件。

Example Custom-site-bundle. 拉鍊

```
.
|-- aws-windows-deployment-manifest.json
|-- siteInstall.ps1
|-- siteRestart.ps1
|-- siteUninstall.ps1
`-- site-contents.zip
```

將 Amazon RDS 資料庫執行個體新增到您的 .NET 應用程式環境

您可以使用 Amazon Relational Database Service (Amazon RDS) 資料庫執行個體存放由應用程式收集與修改的資料。資料庫可與環境耦合並由 Elastic Beanstalk 管理，或者由另一項服務在外部分開建立與管理。本主題提供了使用 Elastic Beanstalk 主控台建立 Amazon RDS 的說明。資料庫將與環境耦合並由 Elastic Beanstalk 管理。如需將 Amazon RDS 與 Elastic Beanstalk 整合的相關詳細資訊，請參閱 將資料庫新增至您的 Elastic Beanstalk 環境。

章節

將資料庫執行個體新增到您的環境

欲將資料庫執行個體新增到您的環境

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。

2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇 Configuration (組態)。
4. 在 Database (資料庫) 組態類別中，選擇 Edit (編輯)。
5. 選擇資料庫引擎，並輸入使用者名稱和密碼。
6. 若要儲存變更，請選擇頁面底部的儲存變更。

新增資料庫執行個體約需要 10 分鐘。環境更新完成時，資料庫執行個體的主機名稱和其他連線資訊會透過下列環境屬性提供給您的應用程式：

屬性名稱	描述	屬性值
RDS_HOSTNAME	資料庫執行個體的主機名稱。	在 Amazon RDS 主控台：端點的連線能力和安全性索引標籤上。
RDS_PORT	資料庫執行個體接受連線的連接埠。預設值在不同資料庫引擎中有所差異。	在 Amazon RDS 主控台：連接埠的連線能力和安全性索引標籤上。
RDS_DB_NAME	資料庫名稱，**ebdb**。	在 Amazon RDS 主控台：資料庫名稱的組態索引標籤上。
RDS_USERNAME	您為資料庫設定的使用者名稱。	在 Amazon RDS 主控台：主要使用者名稱的組態索引標籤上。
RDS_PASSWORD	您為資料庫設定的密碼。	無法在 Amazon RDS 主控台中提供參考。

如需設定與 Elastic Beanstalk 環境耦合之資料庫執行個體的相關詳細資訊，請參閱 將資料庫新增至您的 Elastic Beanstalk 環境。

下載驅動程式

利用 EntityFramework 來下載和安裝您的開發環境適用的 NuGet 套件與資料庫驅動程式。

適用於 .NET 的常見實體架構資料庫供應商

- SQL Server – Microsoft.EntityFrameworkCore.SqlServer
- MySQL – Pomelo.EntityFrameworkCore.MySql
- PostgreSQL – Npgsql.EntityFrameworkCore.PostgreSQL

連線至資料庫

Elastic Beanstalk 會在環境屬性中提供已連接的資料庫執行個體連線資訊。使用 ConfigurationManager.AppSettings 來讀取屬性和設定資料庫的連線。

Example Helpers.cs - 連線字串方法

```
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Linq;
using System.Web;

namespace MVC5App.Models
{
  public class Helpers
  {
    public static string GetRDSConnectionString()
    {
      var appConfig = ConfigurationManager.AppSettings;

      string dbname = appConfig["RDS_DB_NAME"];

      if (string.IsNullOrEmpty(dbname)) return null;

      string username = appConfig["RDS_USERNAME"];
      string password = appConfig["RDS_PASSWORD"];
      string hostname = appConfig["RDS_HOSTNAME"];
      string port = appConfig["RDS_PORT"];

      return "Data Source=" + hostname + ";Initial Catalog=" + dbname + ";User ID=" +
 username + ";Password=" + password + ";";
```

```
    }
  }
}
```

使用連線字串來將您的資料庫內容初始化。

Example DBContext.cs

```
using System.Data.Entity;
using System.Security.Claims;
using System.Threading.Tasks;
using Microsoft.AspNet.Identity;
using Microsoft.AspNet.Identity.EntityFramework;

namespace MVC5App.Models
{
  public class RDSContext : DbContext
  {
    public RDSContext()
      : base(GetRDSConnectionString())
    {
    }

    public static RDSContext Create()
    {
      return new RDSContext();
    }
  }
}
```

AWS Toolkit for Visual Studio

Visual Studio 提供範本供不同程式設計語言及應用程式類型使用，您可從這些範本入門。AWS Toolkit for Visual Studio 亦提供三種自舉應用程式開發的專案範本：AWS 主控台專案、AWS Web 專案和 AWS 空白專案。在此範例中，您將建立新的 ASP.NET Web 應用程式。

欲建立新的 ASP.NET Web 應用程式專案

1. 在 Visual Studio 中，於 File (檔案) 選單按一下 New (新增)，然後按一下 Project (專案)。
2. 在 New Project (新增專案) 對話方塊中，按一下 Installed Templates (已安裝的範本)、按一下 Visual C#，然後按一下 Web。按一下 ASP.NET Empty Web Application (ASP.NET 空白 Web 應用程式)，輸入專案名稱，然後按一下 OK (確定)。

欲執行專案

請執行下列其中一項：

1. 按 F5。
2. 在 Debug (除錯) 選單中，選擇 Start Debugging (開始除錯)。

本機測試

Visual Studio 可讓您輕鬆於本機測試應用程式。欲測試或執行 ASP.NET Web 應用程式，您需要 Web 伺服器。Visual Studio 提供多種選項，例如網際網路資訊服務 (IIS)、IIS Express 或內建的 Visual Studio 程式開發伺服器。欲了解這些選項並決定哪種最適合您，請參閱 Visual Studio 中 ASP.NET Web 專案的 Web 伺服器。

建立 Elastic Beanstalk 環境

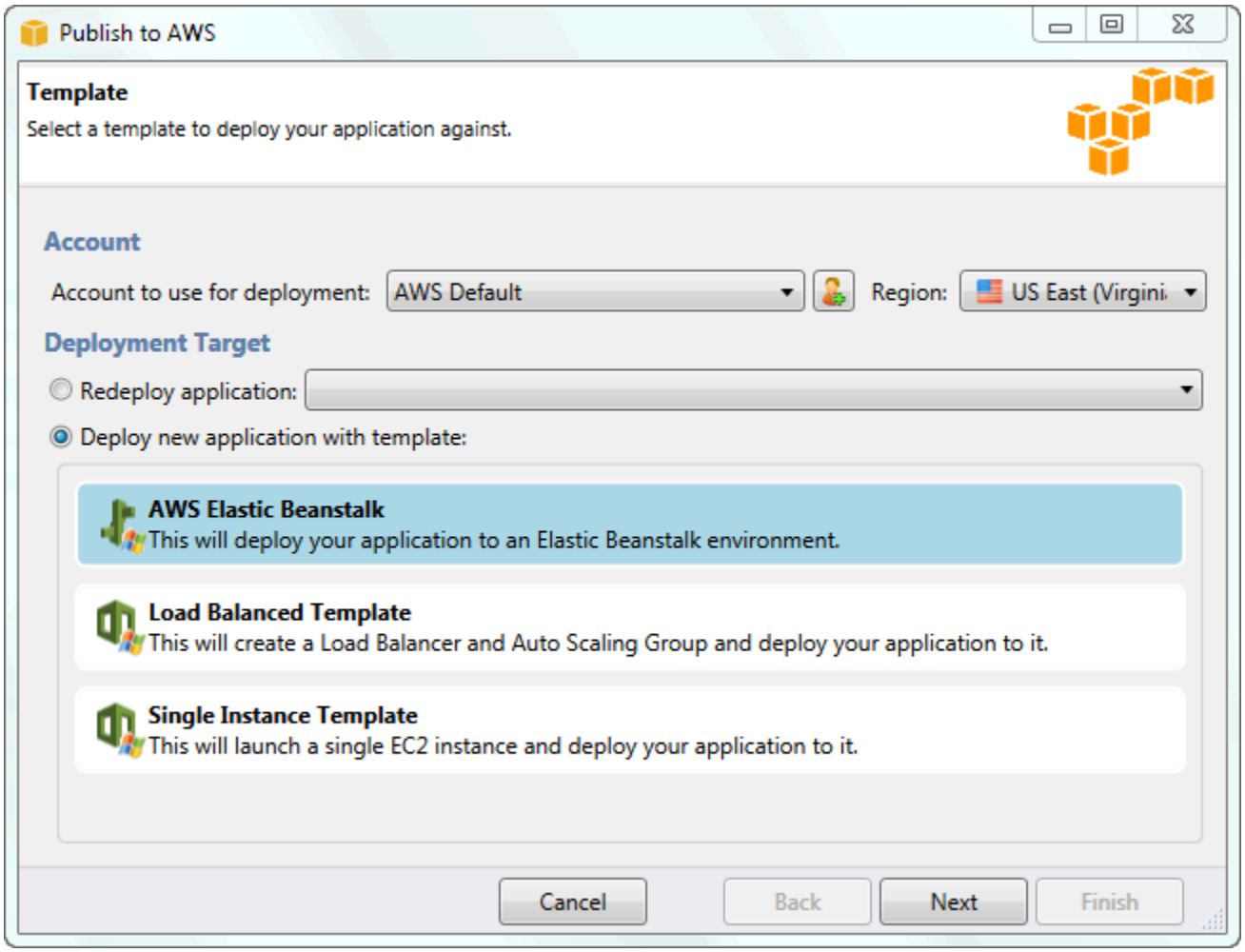
測試應用程式後，即可準備將其部署至 Elastic Beanstalk。

> **Note**
>
> 組態檔案須為封存檔內含專案的一部分。或者，若未將組態檔案納入專案，您可使用 Visual Studio 來部署專案資料夾中的所有檔案。在 Solution Explorer (方案總管) 中，在專案名稱上按一下滑鼠右鍵，然後選擇 Properties (屬性)。按一下 Package/Publish Web (封裝/發行 Web) 索引標籤。在 Items to deploy (要部署的項目) 區段，於下拉式清單選取 All Files in the Project Folder (此專案資料夾中的所有檔案)。

使用 AWS Toolkit for Visual Studio 將應用程式部署到 Elastic Beanstalk

1. 在 Solution Explorer 中，以滑鼠右鍵按一下您的應用程式，然後選取 Publish to AWS (發佈至 AWS)。
2. 在 Publish to AWS (發佈至 AWS) 精靈中，輸入您的帳戶資訊。
   a. 在 AWS account to use for deployment (用於部署的 AWS 帳戶) 部分，選取您的帳戶，或選取 Other (其他) 來輸入新的帳戶資訊。
   b. 在 Region (區域) 部分，選取您欲部署應用程式的區域。如需可用 AWS 區域的資訊，請參閱《AWS 一般參考》中的 AWS Elastic Beanstalk 端點與配額。若您選擇的區域不受 Elastic Beanstalk 支援，將無法選擇部署至 Elastic Beanstalk 的選項。

c. 按一下 Deploy new application with template (以範本部署新的應用程式)，然後選取 Elastic Beanstalk。然後按一下 Next (下一步)。



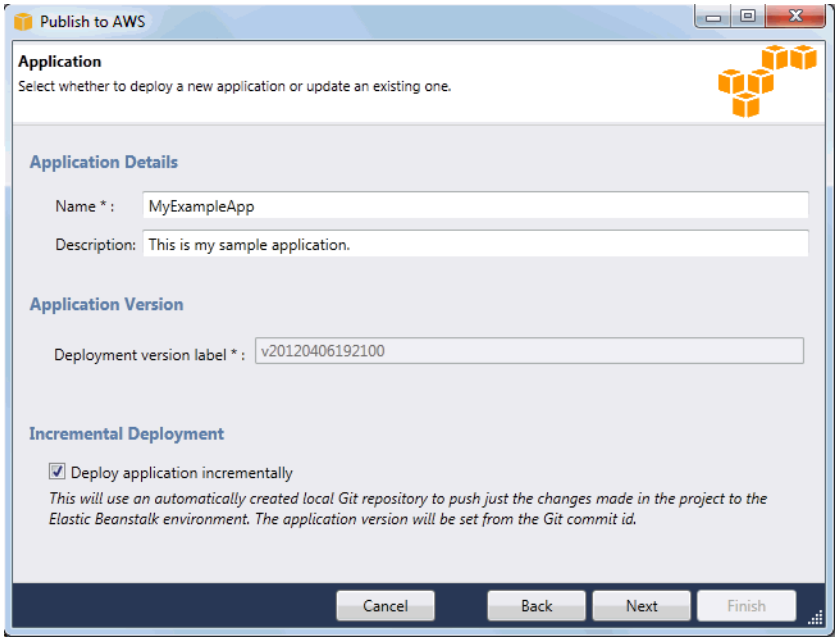
3. 在 Application (應用程式) 頁面，輸入您的應用程式詳細資訊。

   a. 在 Name (名稱) 的部分，輸入應用程式的名稱。

   b. 在 Description (描述) 的部分，輸入應用程式的描述。此步驟為選用。

   c. 應用程式的版本標籤會自動顯示在 Deployment version label (部署版本標籤) 中。

   d. 選取 Deploy application incrementally (漸進部署應用程式)，可僅部署已變更的檔案。漸進部署速度較快，因為您僅需更新已變更的檔案，而非所有檔案。若您選擇此選項，應用程式版本會自 Git commit ID 進行設定。若您未選擇漸進部署應用程式，則可在 Deployment version label (部署的版本標籤) 方塊中更新版本標籤。



e. 按一下下一步。

4. 在 Environment (環境) 頁面中，描述您的環境詳細資訊。

   a. 選取 Create a new environment for this application (為此應用程式建立新環境)。

   b. 在 Name (名稱) 的部分，輸入環境名稱。

   c. 在 Description (描述) 的部分，說明您的環境。此步驟為選用。

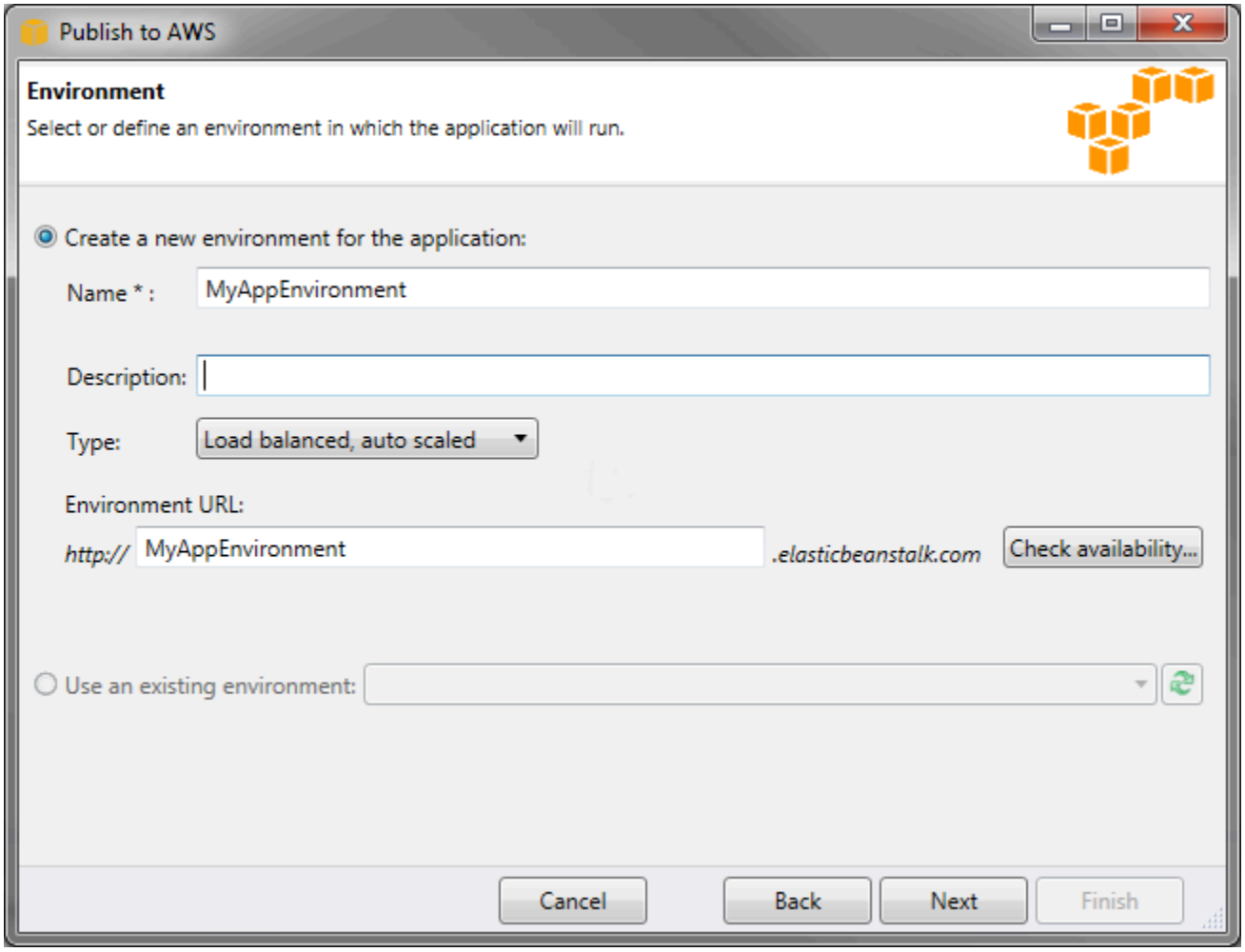
   d. 選取您想要的環境 Type (類型)。

      您可以選取 Load balanced, auto scaled (負載平衡、自動調整規模) 或 Single instance (單一執行個體) 環境。如需更多詳細資訊，請參閱 環境類型。

      > **Note**
      >
      > 以單一執行個體環境而言，負載平衡、Auto Scaling 和運作狀態檢查 URL 設定均不適用。

   e. 將游標移至 Environment URL (環境 URL) 方塊，將自動顯示環境 URL。

   f. 按一下 Check availability (檢查可用性)，以確認可用該環境 URL。



g. 按一下下一步。

5. 在 AWS Options (選項) 頁面，設定部署的其他選項和安全資訊。

   a. 在 Container Type (容器類型) 的部分，選取 64bit Windows Server 2012 running IIS 8 (執行 IIS 8 的 64 位元 Windows Server 2012) 或 64bit Windows Server 2008 running IIS 7.5 (執行 IIS 7.5 的 64 位元 Windows Server 2008)。

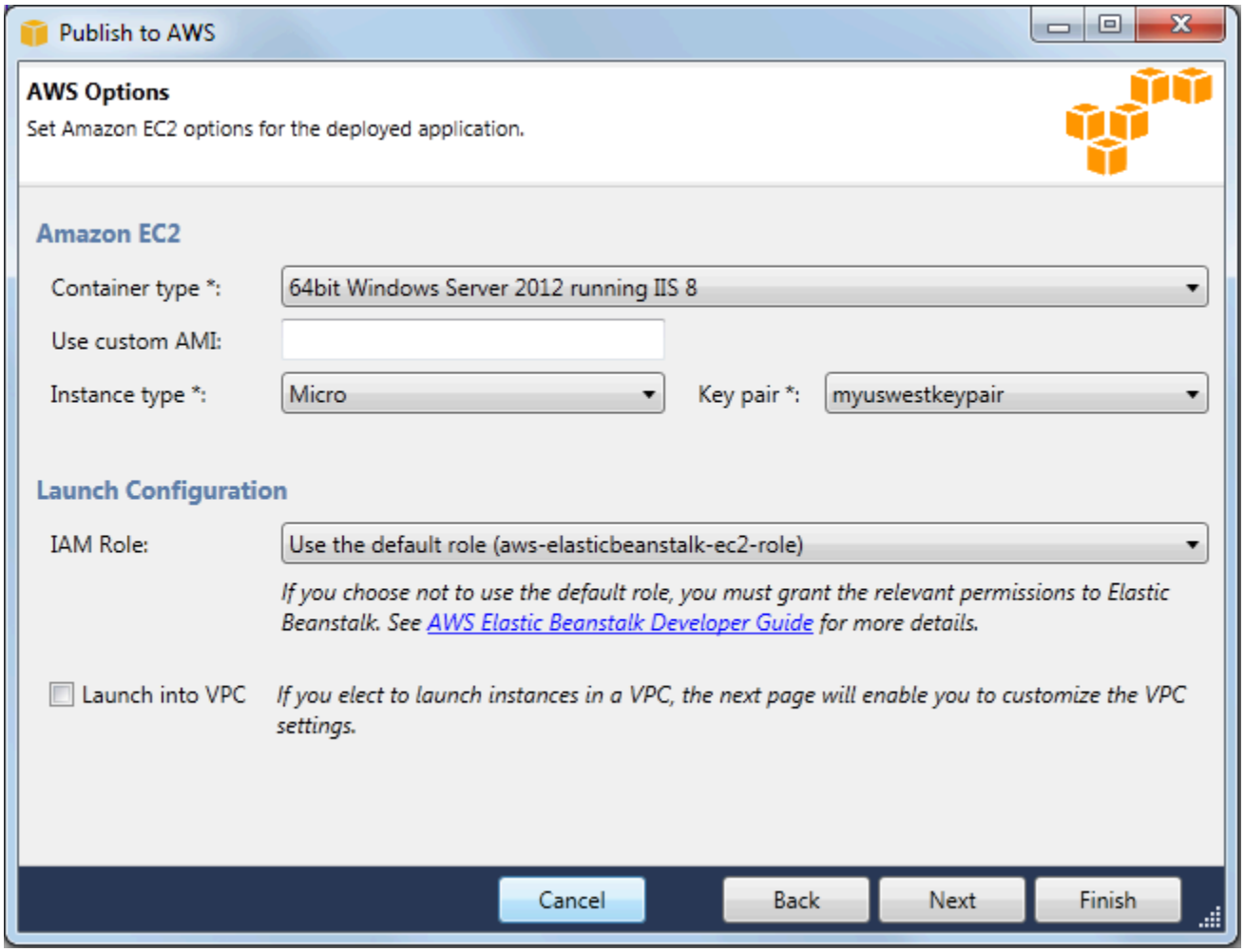
   b. 在 Instance Type (執行個體類型) 的部分，選取 Micro (微型)。

   c. 在 Key pair (金鑰對) 的部分，選取 Create new key pair (建立新的金鑰對)。輸入新的金鑰對名稱 — 以此範例而言，我們使用 **`myuswestkeypair`**，然後按一下 OK (確定)。金鑰對可啟用您 Amazon EC2 執行個體的遠端桌面存取。如需 Amazon EC2 金鑰對的詳細資訊，請參閱《Amazon Elastic Compute Cloud 使用者指南》中的使用登入資料。

   d. 選取執行個體描述檔。

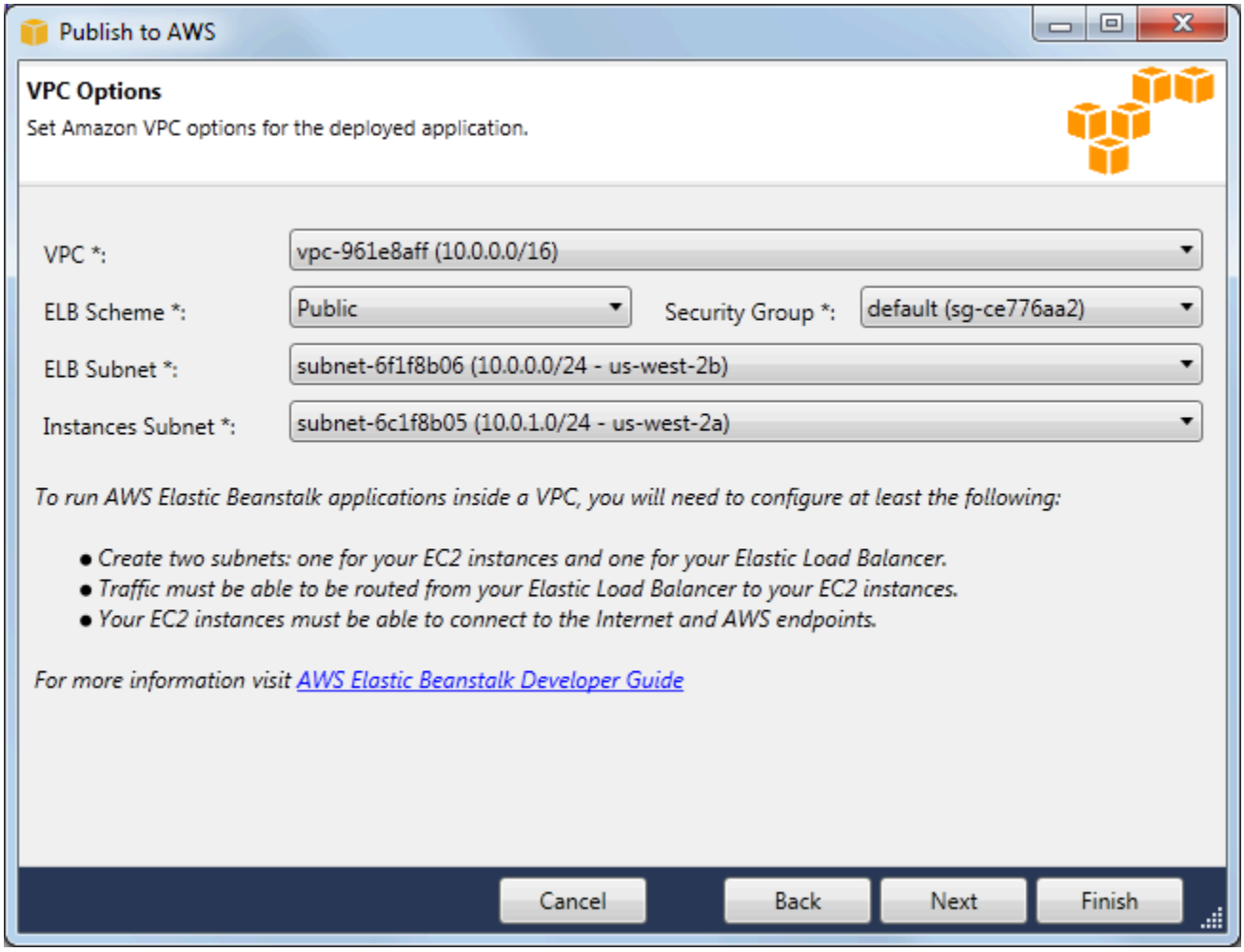
若您沒有執行個體描述檔，請選取 Create a default instance profile (建立預設執行個體描述檔)。如需有關搭配 Elastic Beanstalk 使用執行個體描述檔的詳細資訊，請參閱管理 Elastic Beanstalk 執行個體描述檔。

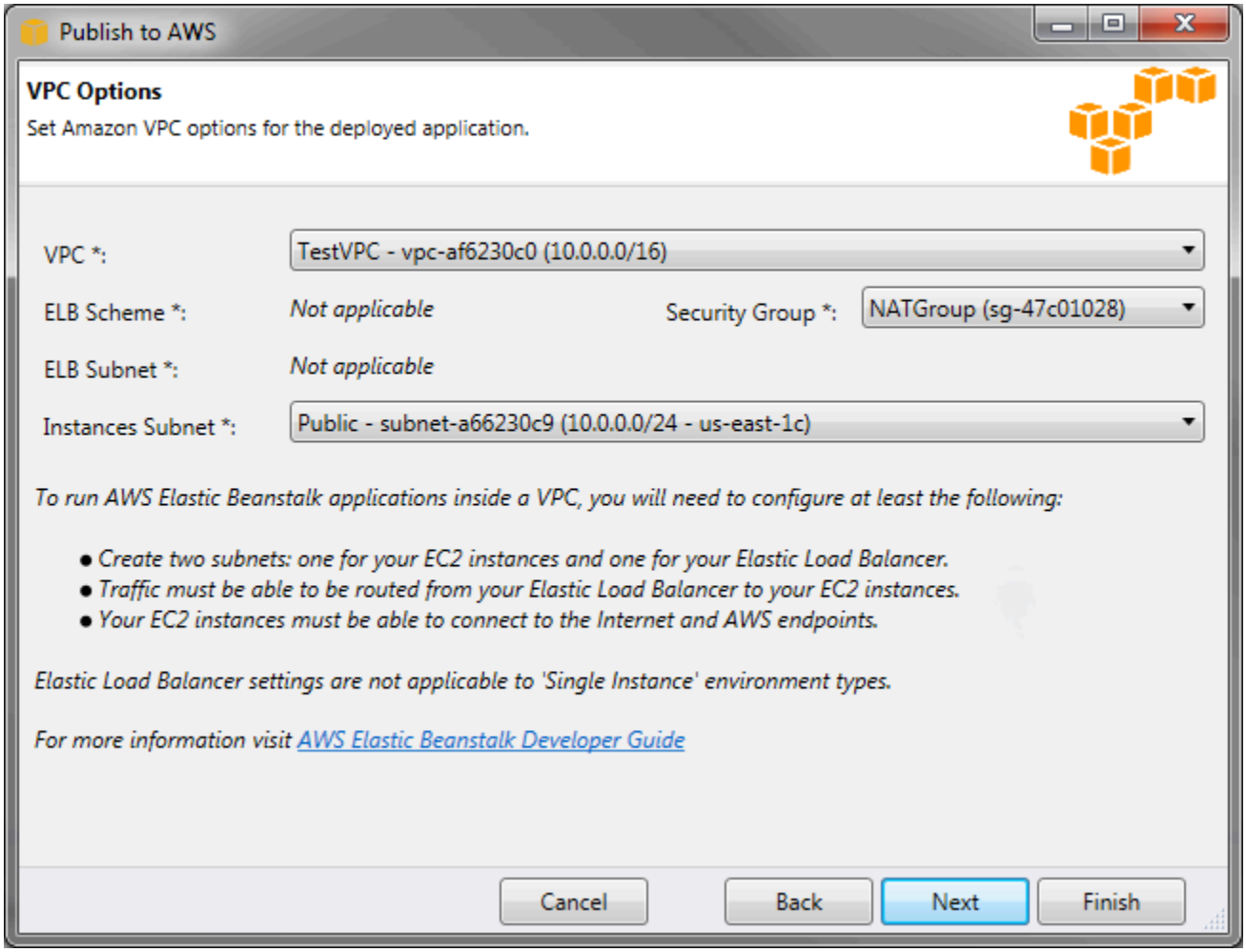
e. 若您擁有自訂 VPC，且希望環境與其搭配使用，按一下 Launch into VPC (啟動至 VPC)。您可於下個頁面設定 VPC 資訊。如需 Amazon VPC 的詳細資訊，請參閱 Amazon Virtual Private Cloud (Amazon VPC)。如需支援的非舊式容器類型的清單，請參閱 the section called "為何部分平台版本標記為舊版？"



f. 按一下下一步。

6. 若您選擇於 VPC 內啟動您的環境，將顯示 VPC Options (VPC 選項) 頁面，否則會出現 Additional Options (其他選項) 頁面。您可在此設定 VPC 選項。


Publish to AWS
VPC Options
Set Amazon VPC options for the deployed application.
VPC *:
vpc-961e8aff (10.0.0.0/16)
ELB Scheme *:
Public
Security Group *:
default (sg-ce776aa2)
ELB Subnet *:
subnet-6f1f8b06 (10.0.0.0/24 - us-west-2b)
Instances Subnet *:
subnet-6c1f8b05 (10.0.1.0/24 - us-west-2a)
To run AWS Elastic Beanstalk applications inside a VPC, you will need to configure at least the following:
Create two subnets: one for your EC2 instances and one for your Elastic Load Balancer.
Traffic must be able to be routed from your Elastic Load Balancer to your EC2 instances.
Your EC2 instances must be able to connect to the Internet and AWS endpoints.
For more information visit AWS Elastic Beanstalk Developer Guide
Cancel
Back
Next
Finish



a. 選取啟動您環境所在 VPC 的 VPC ID。

b. 對於有負載平衡且可擴展的環境而言，若您不希望 Elastic Load Balancer 可自網際網路取得，請於ELB 機制 選取私有。

   以單一執行個體環境而言，此選項不適用，因為環境沒有負載平衡器。如需更多詳細資訊，請參閱 環境類型。

c. 對於有負載平衡且可擴展的環境而言，請為 Elastic Load Balancer 和 EC2 執行個體選取子網路。若您已建立公有和私有子網路，請確認彈性負載平衡器和 EC2 執行個體與正確的子網路建立關聯。根據預設，Amazon VPC 建立的預設公有和私有子網路，分別使用 10.0.0.0/24 和 10.0.1.0/24。您可以前往 https://console.aws.amazon.com/vpc/，在 Amazon VPC 主控台中檢視現有的子網路。

   以單一執行個體環境而言，您的 VPC 只需要公有子網路供執行個體使用。選取負載平衡器的子網路並不適用，因為環境沒有負載平衡器。如需更多詳細資訊，請參閱 環境類型。

d. 對於有負載平衡且可擴展的環境而言，請選取您已為執行個體建立的安全群組 (如適用)。

以單一執行個體環境而言，您不需要 NAT 裝置。選取預設安全群組。Elastic Beanstalk 會將彈性 IP 地址指派給執行個體，讓其能夠存取網際網路。

e. 按一下下一步。

7. 在 Application Options (應用程式選項) 頁面，設定您的應用程式選項。

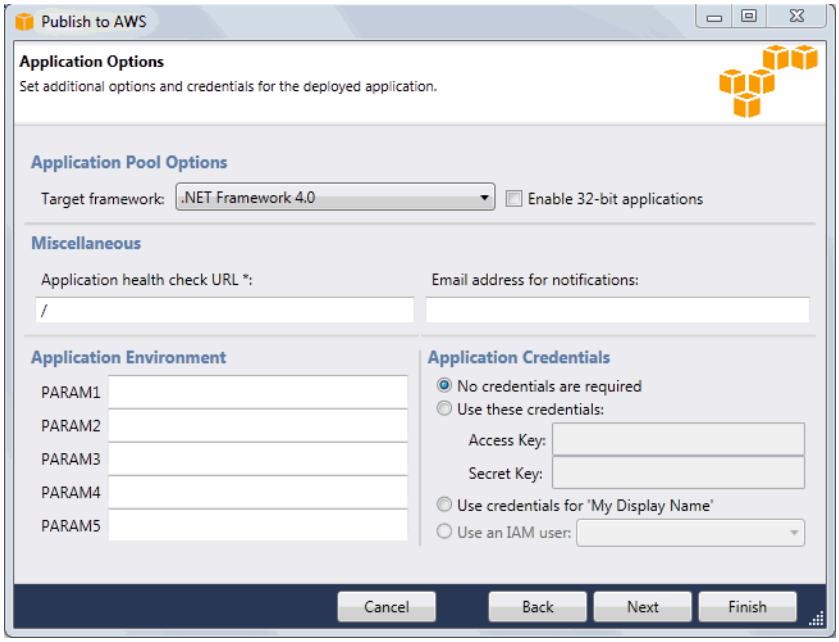
a. 在 Target framework (目標架構) 的部分，選取 .NET Framework 4.0。

b. Elastic Load Balancing 使用運作狀態檢查，來判斷執行您應用程式的 Amazon EC2 執行個體狀態是否健全。運作狀態檢查會在設定的間隔時間探測指定的 URL，藉以判斷執行個體的運作狀態。您可於 Application health check URL (應用程式運作狀態檢查 URL) 方塊中輸入與應用程式現有資源相符的 URL (如 `/myapp/index.aspx`)，藉此覆寫預設 URL。如需應用程式運作狀態檢查的詳細資訊，請參閱 運作狀態檢查。

c. 若您希望接收 Amazon Simple Notification Service (Amazon SNS) 的通知，取得影響您的應用程式的重要事件，請輸入電子郵件地址。

d. Application Environment (應用程式環境) 區段可讓您針對執行您應用程式的 Amazon EC2 執行個體，來指定其上的環境變數。由於您在環境間移動不再需要重新編譯原始碼，因此本設定可提高可攜性。

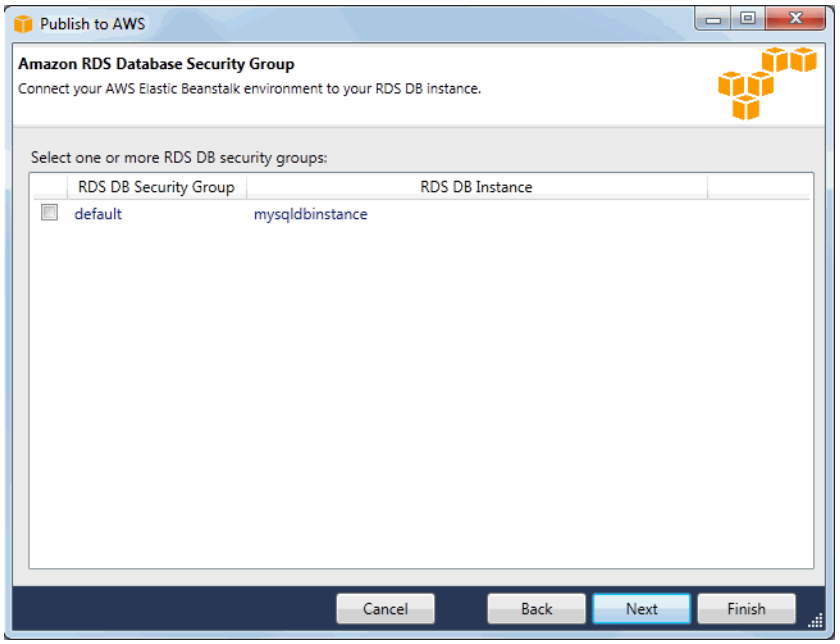
e. 選取您部署應用程式欲使用的應用程式登入資料選項。



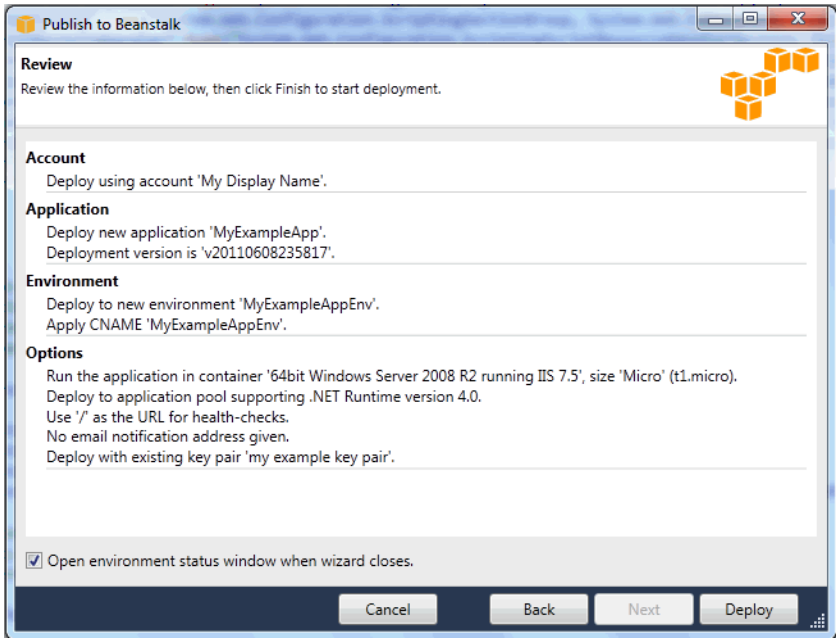
f. 按一下下一步。

8. 若您之前已設定 Amazon RDS 資料庫，將出現 Amazon RDS DB Security Group (Amazon RDS 資料庫安全群組) 頁面。若您希望將 Elastic Beanstalk 環境連接至您的 Amazon RDS 資料庫執行

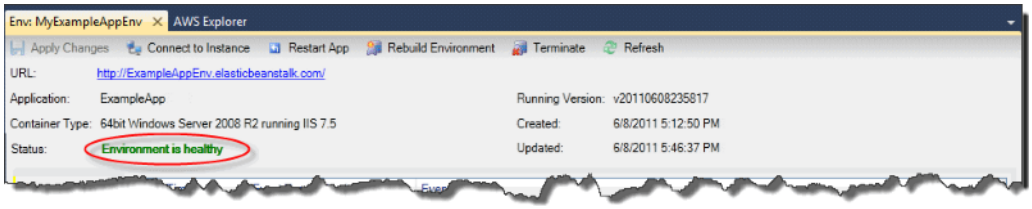
個體，請選取一個或多個安全群組。否則，請進行下一個步驟。當您就緒後，按一下 Next (下一步)。



9. 檢閱您的部署選項。若所有內容均正確，按一下 Deploy (部署)。



您的 ASP.NET 專案將匯出為 Web 部署檔案，上傳至 Amazon S3，並使用 Elastic Beanstalk 註冊為新的應用程式版本。Elastic Beanstalk 部署功能將監控您的環境，直到其具備新部署的程式碼且變為可用。在 env:<environment name> (環境：<環境名稱>) 索引標籤中，您將看到環境的狀態。



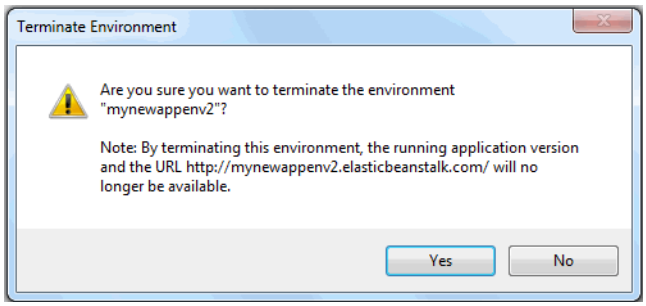
終止環境

若要避免未使用的 AWS 資源收取費用，您可使用 AWS Toolkit for Visual Studio 來終止執行環境。

> **Note**
>
> 您稍後可隨時運用相同版本啟動新的環境。

終止環境

1. 在 AWS Explorer 中，展開 Elastic Beanstalk 節點和應用程式節點。以滑鼠右鍵按一下您的應用程式環境，然後選取 Terminate Environment (終止環境)。
2. 提示出現時，按一下 Yes (是) 以確認您希望終止該環境。Elastic Beanstalk 需要幾分鐘來終止環境中執行的 AWS 資源。



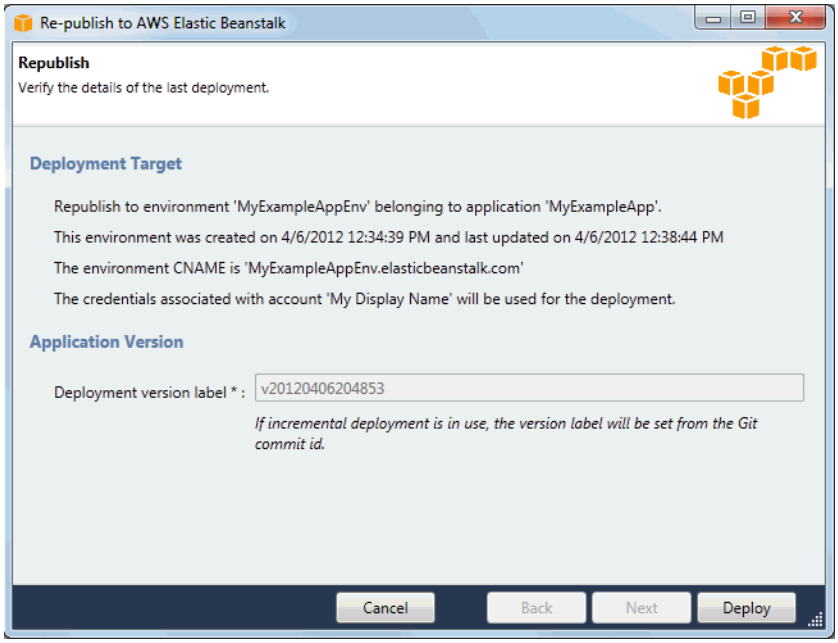
> **Note**
>
> 當您終止環境後，與該終止環境相關聯的 CNAME 可供任何人使用。

部署至您的環境

在您測試應用程式之後，即可輕鬆加以編輯並重新部署您的應用程式，並立刻查看結果。

欲編輯並重新部署您的 ASP.NET Web 應用程式

1. 在 Solution Explorer (方案總管) 中，請以滑鼠右鍵按一下您的應用程式，然後按一下 Republish to Environment <***your environment name***> (重新發佈至環境 <您的環境名稱>)。Re-publish to AWS Elastic Beanstalk (重新發佈至 AWS Elastic Beanstalk) 精靈會隨即開啟。



2. 檢視您的部署詳細資訊，然後按一下 Deploy (部署)。

> **Note**
>
> 如要變更任何設定，您可按一下 Cancel (取消) 並改為使用 Publish to AWS (發佈至 AWS) 精靈。如需說明，請參閱「建立 Elastic Beanstalk 環境」。

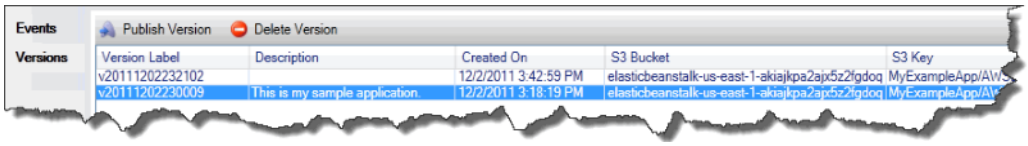
您已更新的 ASP.NET Web 專案將匯出為具有新版本標籤的 Web 部署檔案，上傳至 Amazon S3，並使用 Elastic Beanstalk 註冊為新的應用程式版本。Elastic Beanstalk 部署功能會監控您的現有環境，直到其具備新部署的程式碼且變為可用。在 env:<***environment name***> (環境：<環境名稱>) 索引標籤中，您將看到環境的狀態。

您亦可將現有應用程式部署至現有環境 (例如，若您需要還原至之前的應用程式版本)。

欲將應用程式版本部署至現有環境

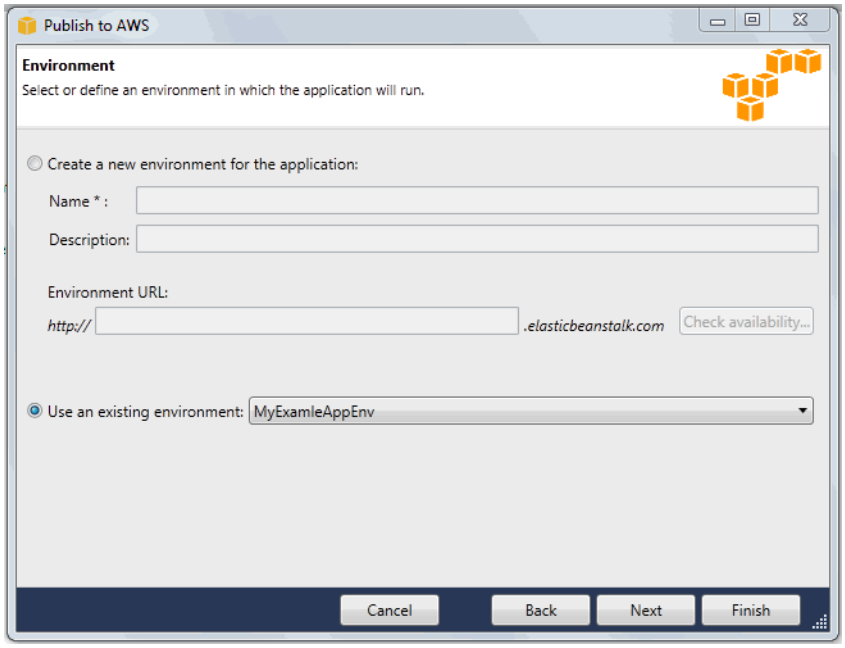
1. 在 AWS Explorer 中展開 Elastic Beanstalk 節點，以滑鼠右鍵按一下 Elastic Beanstalk 應用程式。選取 View Status (檢視狀態)。

2. 在 App: <*application name*> (應用程式：<應用程式名稱>) 索引標籤中，按一下 Versions (版本)。

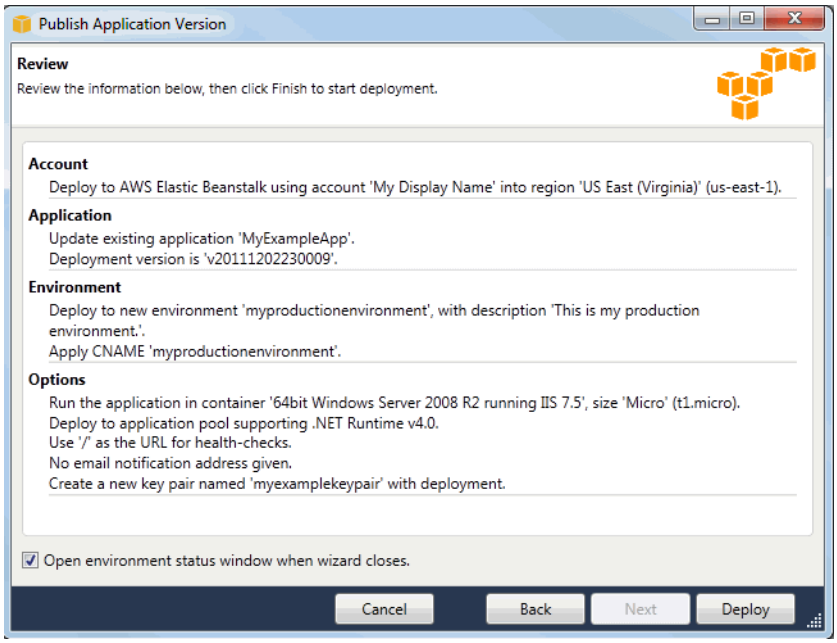


3. 按一下您欲部署的應用程式版本，然後按一下 Publish Version (發佈版本)。

4. 在 Publish Application Version (發佈應用程式版本) 精靈中，按一下 Next (下一步)。



5. 檢視您的部署選項，然後按一下 Deploy (部署)。



您的 ASP.NET 專案將匯出為 Web 部署檔案並上傳至 Amazon S3。Elastic Beanstalk 部署功能將監控您的環境，直到其具備新部署的程式碼且變為可用。在 env:<*environment name*> (環境：<環境名稱>) 索引標籤中，您將看到環境的狀態。

管理您的 Elastic Beanstalk 應用程式環境

透過 AWS Toolkit for Visual Studio 及 AWS 管理主控台，您可針對應用程式環境所用的 AWS 資源，變更其佈建與組態。如需有關如何使用 AWS 管理主控台來管理您的應用程式環境的資訊，請參閱 管理環境。本節討論您可於 AWS Toolkit for Visual Studio 編輯的特定服務設定，做為應用程式環境資訊的一部分。

變更環境資訊設定

部署應用程式時，Elastic Beanstalk 會設定多種 AWS 雲端運算服務。您可使用 AWS Toolkit for Visual Studio 來控制如何設定這些個別服務。

欲編輯應用程式的環境設定

- 展開 Elastic Beanstalk 節點和您的應用程式節點。然後以滑鼠右鍵按一下 AWS Explorer 中的 Elastic Beanstalk 環境。選取 View Status (檢視狀態)。

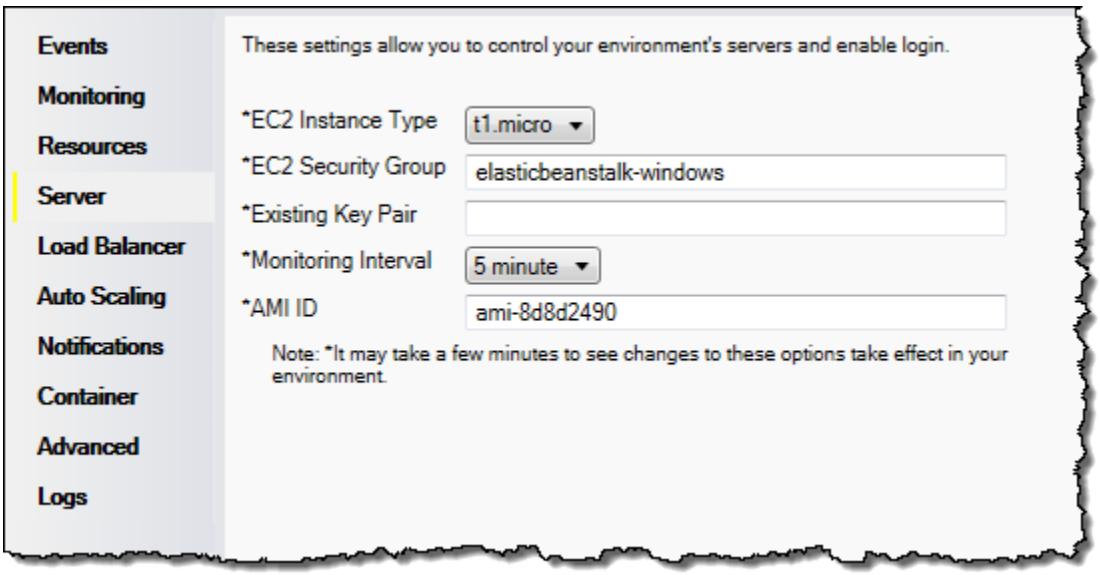
  您現可進行下列設定：

  - 伺服器
  - 負載平衡
  - 自動擴展
  - 通知
  - 環境屬性

使用 AWS Toolkit for Visual Studio 設定 EC2 伺服器執行個體

Amazon Elastic Compute Cloud (Amazon EC2) 是一種 Web 服務，可用於在 Amazon 資料中心啟動並管理伺服器執行個體。您可隨時使用 Amazon EC2 伺服器執行個體，無時間限制，且任何法律用途皆可。執行個體具備不同的大小與組態。如需詳細資訊，請前往 Amazon EC2。

在 AWS Toolkit for Visual Studio 中的應用程式環境分頁，透過 Server (伺服器) 標籤即可編輯 Elastic Beanstalk 環境的 Amazon EC2 執行個體組態。



Amazon EC2 執行個體類型

Instance type (執行個體類型) 顯示您 Elastic Beanstalk 應用程式可用的執行個體類型。請變更執行個體類型，以選取具有最適合您應用程式之特性 (包含記憶體大小和 CPU 能力) 的伺服器。例如，需要進行大量及長時間操作的應用程式可要求更多 CPU 或記憶體。

如需有關您 Elastic Beanstalk 應用程式可用之 Amazon EC2 執行個體類型的詳細資訊，請參閱《Amazon Elastic Compute Cloud 使用者指南》中的執行個體類型。

Amazon EC2 安全群組

您可使用「Amazon EC2 安全群組」來控制 Elastic Beanstalk 應用程式的存取。安全群組會定義您的執行個體的防火牆規則。這些規則會指定哪些輸入 (即傳入) 網路流量應傳送至您的執行個體。所有其他傳入流量將被捨棄。您可隨時修改群組的規則。新規則會自動於所有執行中的執行個體以及未來啟動的執行個體上實施。

您可使用 AWS 管理主控台或 AWS Toolkit for Visual Studio 來設定您的 Amazon EC2 安全群組。在 EC2 Security Groups (EC2 安全群組) 文字方塊中輸入一個或多個 Amazon EC2 安全群組的名稱 (以逗號分隔)，您可藉此指定哪些 Amazon EC2 安全群組可控制您 Elastic Beanstalk 應用程式的存取。

> **Note**
>
> 欲啟用應用程式的運作狀態檢查，請確認可自來源 CIDR 範圍 0.0.0.0/0 存取連接埠 80 (HTTP)。如需運作狀態檢查的詳細資訊，請參閱 運作狀態檢查。

使用 AWS Toolkit for Visual Studio 建立安全群組

1. 在 Visual Studio 的 AWS Explorer 中，展開 Amazon EC2 節點，然後按兩下 Security Groups (安全群組)。
2. 按一下 Create Security Group (建立安全群組)，然後輸入安全群組的名稱和描述。
3. 按一下 OK (確定)。

如需 Amazon EC2 安全群組的詳細資訊，請參閱Amazon Elastic Compute Cloud 使用者指南中的使用安全群組相關文章。

Amazon EC2 金鑰對

透過 Amazon EC2 金鑰對，您可安全登入為 Elastic Beanstalk 應用程式佈建的 Amazon EC2 執行個體。

> **Important**
>
> 您必須建立 Amazon EC2 金鑰對並設定 Elastic Beanstalk 佈建的 Amazon EC2 執行個體，以使用 Amazon EC2 金鑰對，才能存取 Elastic Beanstalk 佈建的 Amazon EC2 執行個體。將應用程式部署至 Elastic Beanstalk 時，您可使用 AWS Toolkit for Visual Studio 內的 Publish to AWS (發佈至 AWS) 精靈來建立金鑰對。若您想要使用 Toolkit 建立其他金鑰對，請依照下列步驟。或者，您可以使用 AWS 管理主控台，設定您的 Amazon EC2 金鑰對。如需建立 Amazon EC2 金鑰對的詳細資訊，請參閱 Amazon Elastic Compute Cloud 入門指南。

Existing Key Pair (現有金鑰對) 文字方塊可讓您指定欲使用的 Amazon EC2 金鑰對名稱，藉此安全登入執行 Elastic Beanstalk 應用程式的 Amazon EC2 執行個體。

欲指定 Amazon EC2 金鑰對名稱

1. 展開 Amazon EC2 節點，然後按兩下 Key Pairs (金鑰對)。
2. 按一下 Create Key Pair (建立金鑰對)，然後輸入金鑰對名稱。
3. 按一下 OK (確定)。

如需 Amazon EC2 金鑰對的詳細資訊，請前往《Amazon Elastic Compute Cloud 使用者指南》中的使用 Amazon EC2 登入資料相關文章。如需連接至 Amazon EC2 執行個體的詳細資訊，請參閱列出和連線到伺服器執行個體。

監控間隔

根據預設，只會啟用基本 Amazon CloudWatch 指標。其會每五分鐘傳回資料一次。您可以藉由為 AWS Toolkit for Eclipse 中您環境之 Configuration (組態) 標籤的 Server (伺服器) 區段中 Monitoring Interval (監控間隔)，選取 1 minute (1 分鐘)，來啟用以每一分鐘為一個間隔之更精密的 CloudWatch 指標。

> **Note**
>
> Amazon CloudWatch 服務費用可適用於一分鐘間隔指標。如需詳細資訊，請參閱 Amazon CloudWatch。

自訂 AMI ID

您可以藉由將您自訂 AMI 的識別符輸入 AWS Toolkit for Eclipse 中您環境 Configuration (組態) 標籤之 Server (伺服器) 區段中的 Custom AMI ID (自訂 AMI ID) 方塊，來使用您的自訂 AMI 覆寫您 Amazon EC2 執行個體的預設 AMI。

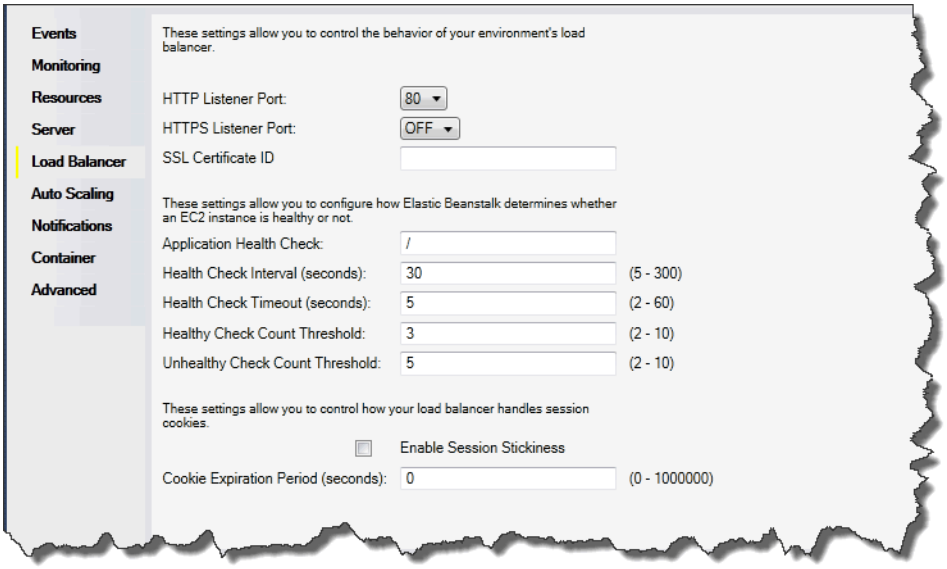
> **Important**
>
> 使用您自己的 AMI 為進階任務，需要謹慎操作。若您需要自訂 AMI，建議您從預設的 Elastic Beanstalk AMI 開始並加以修改。為使運作狀態良好，Elastic Beanstalk 預期 Amazon EC2 執行個體應滿足一組要求，包含具有執行中的主機管理員。若未符合這些要求，您的環境可能無法正常運作。

使用 AWS Toolkit for Visual Studio 設定 Elastic Load Balancing

Elastic Load Balancing 是 Amazon Web 服務，可協助您提升應用程式的可用性和可擴展性。本服務可讓您輕鬆將應用程式負載分配至兩個或多個 Amazon EC2 執行個體。Elastic Load Balancing 可透過備援實現可用性，亦可支援應用程式的流量成長。

Elastic Load Balancing 可讓您自動分配傳入應用程式的流量，在您執行的所有執行個體間取得平衡。當您需要增加應用程式的容量時，本服務亦讓您輕鬆新增新的執行個體。

部署應用程式時，Elastic Beanstalk 會自動佈建 Elastic Load Balancing。在 AWS Toolkit for Visual Studio 中的應用程式環境標籤，透過 Load Balancer (負載平衡器) 標籤即可編輯 Elastic Beanstalk 環境的 Amazon EC2 執行個體組態。



下列章節說明您可設定的 Elastic Load Balancing 參數供應用程式使用。

連接埠

為了為您的 Elastic Beanstalk 應用程式處理請求所佈建的負載平衡器，會將請求傳送至執行您應用程式的 Amazon EC2 執行個體。所佈建的負載平衡器可於 HTTP 和 HTTPS 連接埠接聽請求，並將請求路由至 AWS Elastic Beanstalk 應用程式內的 Amazon EC2 執行個體。負載平衡器預設會處理 HTTP 連接埠上的請求。必須開啟至少一個 HTTP 或 HTTPS 連接埠。

> **Important**
>
> 請確認您指定的連接埠並未鎖定；否則，使用者將無法連接至您的 Elastic Beanstalk 應用程式。

控制 HTTP 連接埠

欲關閉 HTTP 連接埠，請於 HTTP Listener Port (HTTP 接聽程式連接埠) 選取 OFF (關閉)。欲開啟 HTTP 連接埠，請於清單選取 HTTP 連接埠 (如 80 (80))。

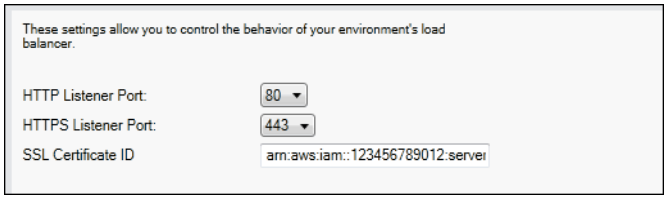
> **Note**
>
> 若要使用預設連接埠 80 以外 (例如連接埠 8080) 的連接埠存取您的環境，請將接聽程式新增至現有的負載平衡器，然後設定該新的接聽程式在該連接埠上接聽。
> 例如，使用適用於 Classic Load Balancer 的 AWS CLI，輸入如下命令，將 *LOAD_BALANCER_NAME* 取代為您 Elastic Beanstalk 負載平衡器的名稱。
>
> ```
> aws elb create-load-balancer-listeners --load-balancer-name LOAD_BALANCER_NAME
>  --listeners "Protocol=HTTP, LoadBalancerPort=8080, InstanceProtocol=HTTP,
>  InstancePort=80"
> ```
>
> 例如，使用適用於 Application Load Balancer 的 AWS CLI，輸入如下命令，將 *LOAD_BALANCER_ARN* 取代為您 Elastic Beanstalk 負載平衡器的 ARN。
>
> ```
> aws elbv2 create-listener --load-balancer-arn LOAD_BALANCER_ARN --protocol HTTP
>  --port 8080
> ```
>
> 若您想要 Elastic Beanstalk 監控您的環境，請勿移除連接埠 80 上的接聽程式。

控制 HTTPS 連接埠

Elastic Load Balancing 支援 HTTPS/TLS 通訊協定，可加密用戶端連線至負載平衡器的流量。負載平衡器至 EC2 執行個體的連線採用純文字加密。HTTPS 連接埠預設為關閉。

欲開啟 HTTPS 連接埠

1. 使用 AWS Certificate Manager (ACM) 建立新的憑證或將憑證和金鑰上傳至 AWS Identity and Access Management (IAM)。如需請求 ACM 憑證的詳細資訊，請參閱《AWS Certificate Manager 使用者指南》中的請求憑證。如需有關將第三方憑證匯入 ACM 的詳細資訊，請參閱《AWS Certificate Manager 使用者指南》中的匯入憑證。如果 ACM 無法在您的區域中使用，請使用 AWS Identity and Access Management (IAM) 上傳第三方憑證。ACM 和 IAM 服務會存放憑證，並針對 SSL 憑證提供 Amazon Resource Name (ARN)。如需建立和上傳憑證至 IAM 的詳細資訊，請參閱IAM 使用者指南中的使用伺服器憑證。
2. 選取 HTTPS Listener Port (HTTPS 接聽程式連接埠) 的連接埠來指定 HTTPS 連接埠。



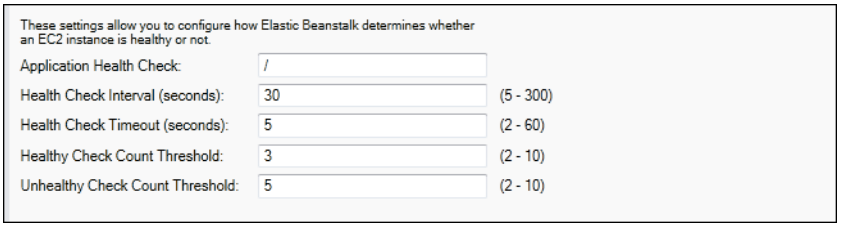
3. 針對 SSL Certificate ID (SSL 憑證 ID)，輸入您 SSL 憑證的 Amazon Resources Name (ARN)，例如 **`arn:aws:iam::123456789012:server-certificate/abc/certs/build`** 或 **`arn:aws:acm:us-east-2:123456789012:certificate/12345678-12ab-34cd-56ef-12345678`**。使用您於步驟 1 建立或上傳的 SSL 憑證。

欲關閉 HTTPS 連接埠，請於 HTTPS Listener Port (HTTPS 接聽程式連接埠) 選取 OFF (關閉)。

運作狀態檢查

運作狀態檢查的定義包括用於查詢執行個體運作狀態的 URL。根據預設，Elastic Beanstalk 針對非舊版容器和舊版容器分別使用 TCP:80 和 HTTP:80。您可於 Application Health Check URL (應用程式運作狀態檢查 URL) 方塊中輸入與應用程式現有資源相符的 URL (如 `/myapp/default.aspx`)，藉此覆寫預設 URL。若您覆寫預設 URL，Elastic Beanstalk 會使用 HTTP 來查詢資源。欲檢查您是否正使用舊版容器類型，請參閱 the section called “為何部分平台版本標記為舊版？”

您可於 Load Balancing (負載平衡) 面板使用 EC2 Instance Health Check (EC2 執行個體運作狀態檢查) 區段，藉此控制運作狀態檢查的設定。



運作狀態檢查的定義包括用於查詢執行個體運作狀態的 URL。您可於 Application Health Check URL (應用程式運作狀態檢查 URL) 方塊中輸入與應用程式現有資源相符的 URL (如 `/myapp/index.jsp`)，藉此覆寫預設 URL。

下列清單說明您應用程式可設定的運作狀態檢查參數。

- 在 Health Check Interval (seconds) (運作狀態檢查間隔，秒) 部分，輸入 Elastic Load Balancing 對應用程式的 Amazon EC2 執行個體進行運作狀態檢查之間的等待秒數。
- 在 Health Check Timeout (seconds) (運作狀態檢查逾時，秒) 部分，指定 Elastic Load Balancing 將執行個體視為沒有回應前的等待回應的秒數。

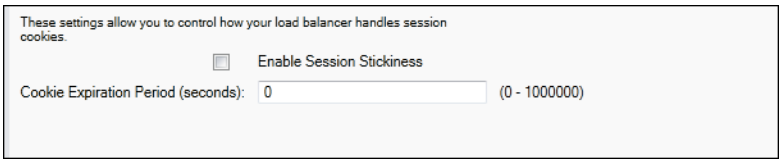
- 在 Healthy Check Count Threshold (運作狀態檢查計數閾值) 和 Unhealthy Check Count Threshold (不健全檢查結果計數臨界值) 部分，指定 Elastic Load Balancing 變更執行個體運作狀態前的連續成功或不成功 URL 探測的次數。例如，若 Unhealthy Check Count Threshold (不健全檢查結果計數臨界值) 部分指定為 **5**，表示 URL 連續五次回傳錯誤訊息或逾時後，Elastic Load Balancing 會將運作狀態檢查視為失敗。

工作階段

負載平衡器預設會以最小的負載，將每個請求獨立路由至伺服器執行個體。相對而言，黏性工作階段會將使用者工作階段繫結到特定的伺服器執行個體，以便工作階段期間來自使用者的所有請求都發送到相同的伺服器執行個體。

應用程式若啟用黏性工作階段，Elastic Beanstalk 會使用負載平衡器產生的 HTTP Cookie。負載平衡器會使用特殊負載平衡器產生的 Cookie，來追蹤每個請求的應用程式執行個體。當負載平衡器收到請求時，首先會檢查此 Cookie 是否存在於請求中。若是，此請求會傳送至 Cookie 中指定的應用程式執行個體。若 Cookie 不存在，則負載平衡器會根據現有負載平衡演算法選擇應用程式執行個體。回應會插入 Cookie，藉此將後續來自相同使用者的請求繫結至該應用程式執行個體。政策組態會定義 Cookie 到期日期，此為每個 Cookie 的有效使用期限。

您可使用 Load Balancer (負載平衡器) 索引標籤上的 Sessions (工作階段) 區段，以指定應用程式的負載平衡器是否允許讓工作階段黏著。



如需 Elastic Load Balancing 的詳細資訊，請前往 Elastic Load Balancing 開發人員指南。

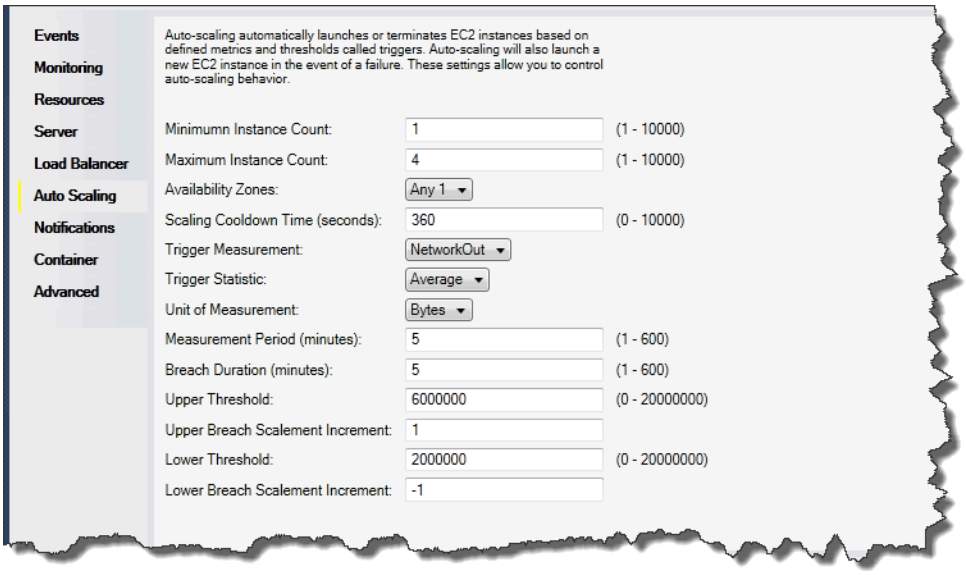
使用 AWS Toolkit for Visual Studio 設定 Auto Scaling

Amazon EC2 Auto Scaling 為 Amazon Web Service，旨在根據使用者定義的觸發來自動啟動或終止 Amazon EC2 執行個體。使用者可設定 Auto Scaling 群組並將「觸發」與這些群組建立關聯，藉此根據諸如頻寬使用量或 CPU 使用率等指標，自動擴展運算資源。Amazon EC2 Auto Scaling 可與 Amazon CloudWatch 搭配使用，以擷取執行應用程式之伺服器執行個體的指標。

Amazon EC2 Auto Scaling 可讓您取得一組 Amazon EC2 執行個體，並設定各種參數，讓此群組自動增減數量。Amazon EC2 Auto Scaling 可於該群組新增或移除 Amazon EC2 執行個體，協助您無縫處理應用程式的流量變更。

Amazon EC2 Auto Scaling 亦會針對其啟動的每個 Amazon EC2 執行個體，監控其運作狀態。如果有任何執行個體未預期終止，Amazon EC2 Auto Scaling 會偵測到終止狀況，並啟動替代執行個體。此功能可讓您自動維持所需的固定 Amazon EC2 執行個體數量。

Elastic Beanstalk 會為您的應用程式佈建 Amazon EC2 Auto Scaling。在 AWS Toolkit for Visual Studio 中的應用程式環境標籤，透過 Auto Scaling 標籤即可編輯 Elastic Beanstalk 環境的 Amazon EC2 執行個體組態。

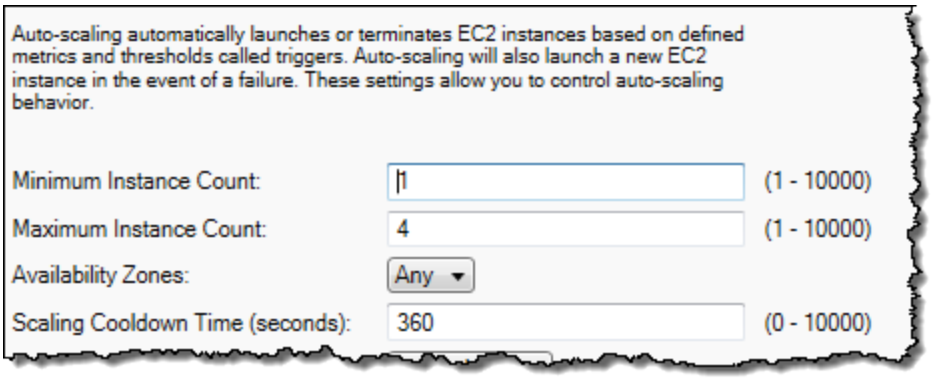


下列章節討論如何設定 Auto Scaling 參數供您的應用程式使用。

啟動組態

您可編輯啟動組態，以控制 Elastic Beanstalk 應用程式佈建 Amazon EC2 Auto Scaling 資源的方式。

Minimum Instance Count (執行個體計數下限) 與 Maximum Instance Count (執行個體計數上限) 方塊，可讓您指定您 Elastic Beanstalk 應用程式使用的 Auto Scaling 群組大小上下限。



> **Note**
>
> 欲保持固定數量的 Amazon EC2 執行個體，請將 Minimum Instance Count (執行個體計數下限) 和 Maximum Instance Count (執行個體計數上限) 設為相同值。

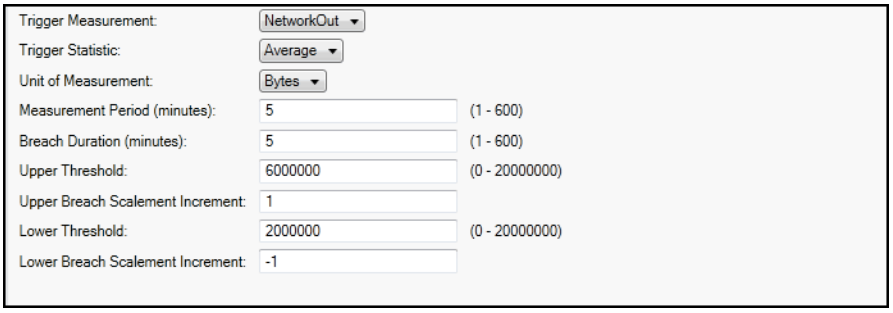
Availability Zones (可用區域) 方塊可讓您指定 Amazon EC2 執行個體所在可用區域數量。若您想要建構容錯應用程式，請務必設定此數值。若一個可用區域發生故障，您其他可用區域內的執行個體仍將繼續執行。

> **Note**
>
> 目前，您無法指定執行個體所在的可用區域。

觸發

「觸發」是您可加以設定的 Amazon EC2 Auto Scaling 機制，以通知系統您想要增加 (「擴展」) 或減少 (「縮減」) 執行個體數量的時機。您可設定觸發，在指標 (如 CPU 使用率) 發佈至 Amazon CloudWatch 時「觸發」，並判斷是否滿足您指定的條件。在特定期間內，若達到您為指標指定的條件閾值上下限，則觸發會啟動名為「擴展活動」的長時間執行程序。

您可透過 AWS Toolkit for Visual Studio，為 Elastic Beanstalk 應用程式定義擴展觸發條件。



Amazon EC2 Auto Scaling 會觀看執行個體的特定 Amazon CloudWatch 指標，藉此進行觸發作業。觸發條件包括 CPU 使用率、網路流量及磁碟活動。使用 Trigger Measurement (觸發條件測量指標) 設定以選取觸發的指標。

下列清單說明您可使用 AWS 管理主控台設定的觸發參數。

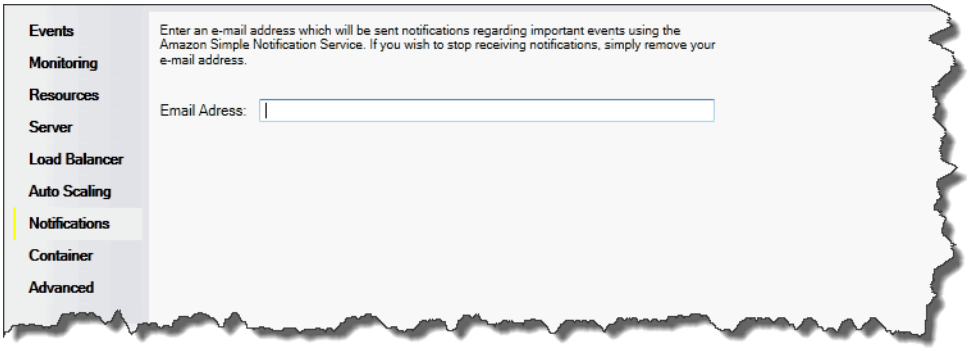
- 您可指定觸發應使用的統計資料。針對 Trigger Statistic (觸發條件統計資料)，您可選擇 Minimum (下限)、Maximum (上限)、Sum (總和) 或 Average (平均)。

- 在 Unit of Measurement (測量單位) 部分指定觸發條件測量的單位。
- Measurement Period (測量期間) 方塊的數值可指定 Amazon CloudWatch 測量觸發指標的頻率。在 Breach Duration (違規持續時間) 部分，您可定義指標在超過所定義的限制 (即 Upper Threshold (閾值上限) 和 Lower Threshold (閾值下限) 所指定) 後，引發觸發前的時間。
- 在 Upper Breach Scale Increment (上限違規規模調整增幅) 和 Lower Breach Scale Increment (下限違規規模調整增幅) 部分，可指定在進行擴展活動時，欲新增或移除的 Amazon EC2 執行個體數量。

如需 Amazon EC2 Auto Scaling 的詳細資訊，請參閱 Amazon Elastic Compute Cloud 文件中的「Amazon EC2 Auto Scaling」一節。

使用 AWS Toolkit for Visual Studio 設定通知

Elastic Beanstalk 使用 Amazon Simple Notification Service (Amazon SNS) 來通知您影響應用程式的重要事件。欲啟用 Amazon SNS 通知，只要在 Email Address (電子郵件地址) 方塊中輸入您的電子郵件地址即可。欲停用此通知，請將您的電子郵件地址自方塊移除。



使用 AWS Toolkit for Visual Studio 設定 .NET 容器

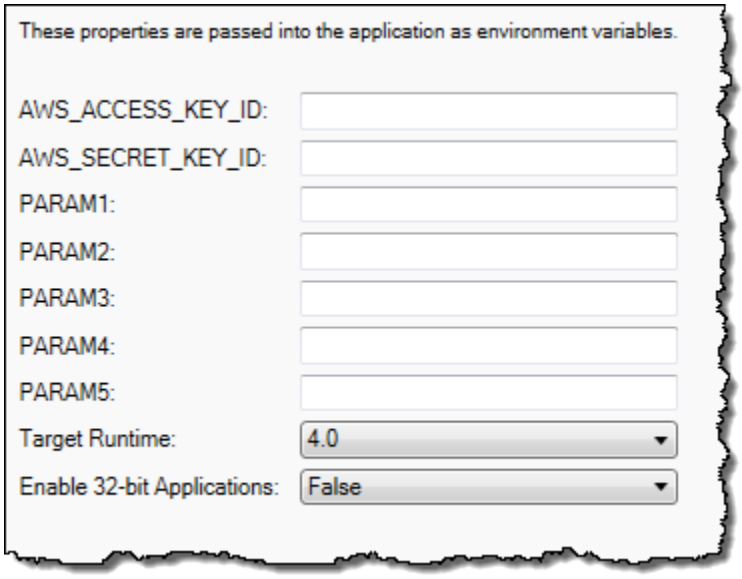
Container/.NET Options (Container/.NET 選項) 面板可讓您微調 Amazon EC2 執行個體的行為，也可啟用或停用 Amazon S3 日誌輪換。您可使用 AWS Toolkit for Visual Studio 來設定您的容器資訊。

> **Note**
>
> 您可交換環境 CNAME，藉此零停機修改您的組態設定。如需更多詳細資訊，請參閱 透過 Elastic Beanstalk 進行藍/綠部署。

若有需要，則可增加參數的數量。如需有關增加參數的詳細資訊，請參閱 選項設定。

存取 Elastic Beanstalk 應用程式的 Container/.NET 選項面板

1. 在 AWS Toolkit for Visual Studio 中，請展開 Elastic Beanstalk 節點，然後展開您的應用程式的節點。
2. 在 AWS Explorer 中，按兩下 Elastic Beanstalk 環境。
3. 在 Overview (概觀) 窗格底部，按一下 Configuration (組態) 索引標籤。
4. 在 Container (容器) 底下，您可設定容器選項。

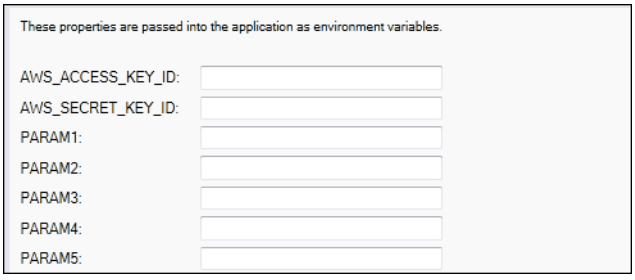


.NET 容器選項

您可選擇應用程式適用的 .NET Framework 版本。Target runtime (目標執行時間) 部分請選擇 2.0 或 4.0。若您欲啟用 32 位元應用程式，請選取 Enable 32-bit Applications (啟用 32 位元應用程式)。

應用程式設定

Application Settings (應用程式設定) 區段讓您指定可從應用程式程式碼讀取的環境變數。



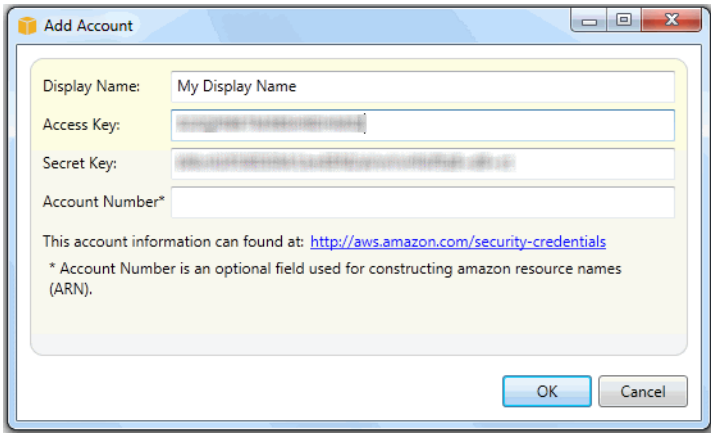
管理帳戶

若您想設定不同的 AWS 帳戶來執行不同任務 (如測試、封測和生產)，請使用 AWS Toolkit for Visual Studio 來新增、編輯和刪除帳戶。

欲管理多個帳戶

1. 在 Visual Studio 的 View (檢視) 選單中，按一下 AWS Explorer。
2. 在 Account (帳戶) 清單旁，按一下 Add Account (新增帳戶) 按鈕。

會出現 Add Account (新增帳戶) 對話方塊。



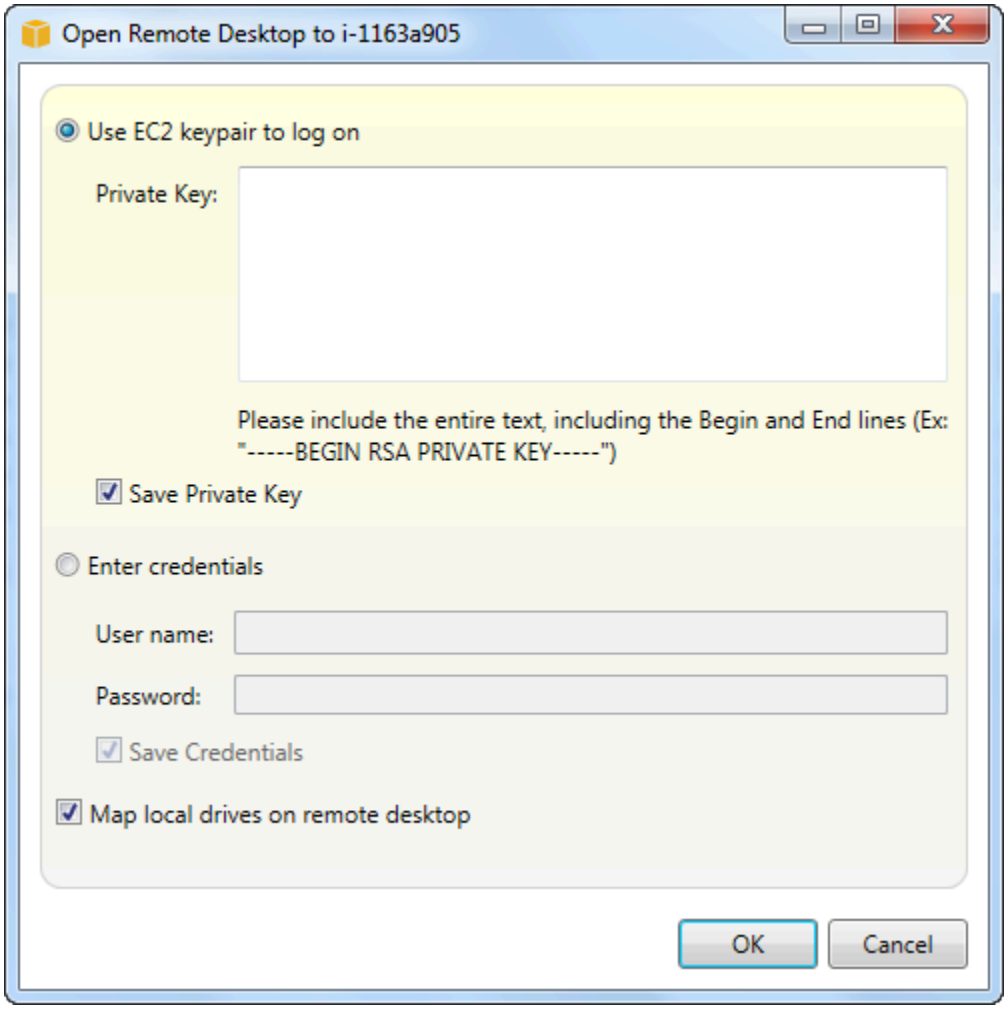
3. 填寫請求的資訊。
4. 現在，您的帳戶資訊會顯示於 AWS Explorer 標籤。在發佈至 Elastic Beanstalk 時，您可以選取欲使用的帳戶。

列出和連線到伺服器執行個體

透過 AWS Toolkit for Visual Studio，或是從 AWS 管理主控台，您可以針對執行您 Elastic Beanstalk 應用程式環境的 Amazon EC2 執行個體，來檢視其清單。您可以使用遠端桌面連線功能，來連線到這些執行個體。如需透過 AWS 管理主控台列出並連接至您伺服器執行個體的詳細資訊，請參閱 列出和連線到伺服器執行個體。下列章節帶您使用 AWS Toolkit for Visual Studio，逐步地檢視和連線到您的伺服器執行個體。

欲檢視並連接至環境的 Amazon EC2 執行個體

1. 在 Visual Studio 的 AWS Explorer 中，展開 Amazon EC2 節點，然後按兩下 Instances (執行個體)。
2. 在 Instance (執行個體) 欄中，針對您應用程式的負載平衡器中執行的 Amazon EC2 執行個體，在其執行個體 ID 上按一下滑鼠右鍵，然後從右鍵選單中選擇 Open Remote Desktop (開啟遠端桌面)。



3. 選擇 Use EC2 keypair to log on (使用 EC2 金鑰對來登入)，然後在 Private key (私密金鑰) 方塊中，貼入您用來部署應用程式的私密金鑰檔案的內容。或者，請在 User name (使用者名稱) 與 Password (密碼) 文字方塊中，輸入您的使用者名稱和密碼。

> **Note**
>
> 如果金鑰對儲存於 Toolkit 中，則不會顯示文字方塊。

4. 按一下 OK (確定)。

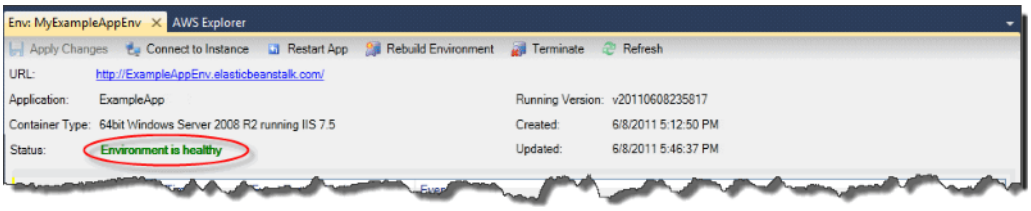
監控應用程式運作狀態

當您執行的是生產網站，了解您應用程式的可用性和回應請求的情況至關重要。為了協助監控您應用程式的回應能力，Elastic Beanstalk 提供的功能可監控應用程式的統計資訊，並建立超過閾值的觸發警示。

如需 Elastic Beanstalk 提供的運作狀態監控詳細資訊，請參閱基礎型運作狀態報告。

您可以透過 AWS Toolkit for Visual Studio 或 AWS 管理主控台來存取您應用程式的相關運作資訊。

工具組會於 Status (狀態) 欄位顯示您環境的狀態和應用程式運作狀態。



欲監控應用程式運作狀態

1. 在 AWS Toolkit for Visual Studio 中的 AWS Explorer，展開 Elastic Beanstalk 節點，然後展開您的應用程式的節點。
2. 以滑鼠右鍵按一下您的 Elastic Beanstalk 環境，然後按一下 View Status (檢視狀態)。
3. 在您的應用程式環境分頁中，按一下 Monitoring (監控)。

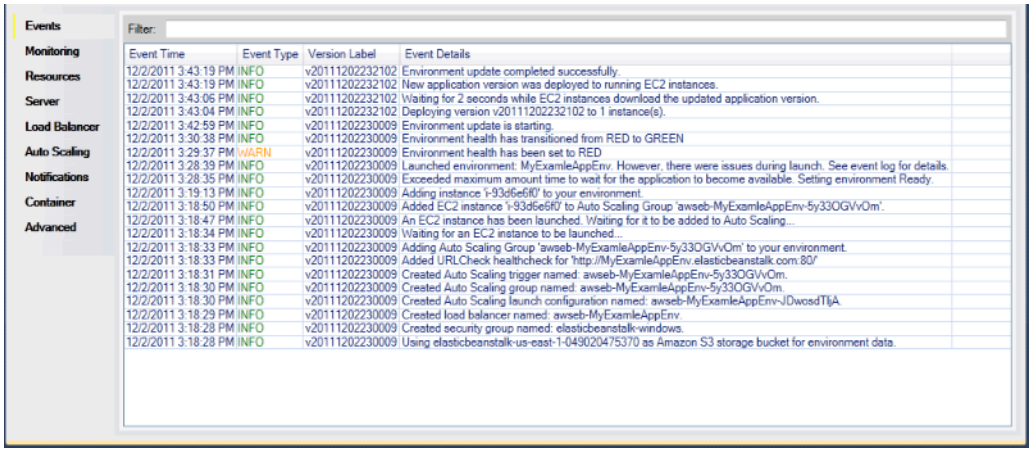
   Monitoring (監控) 面板包含一組圖表顯示您特定應用程式環境的資源使用情況。

> **Note**
>
> 時間範圍預設為最後一個小時。欲修改此設定，在 Time Range (時間範圍) 清單按一下不同的時間範圍。

您可使用 AWS Toolkit for Visual Studio 或 AWS 管理主控台來檢視與應用程式相關的事件。

檢視應用程式事件

1. 在 AWSToolkit for Visual Studio 中的 AWS Explorer，展開 Elastic Beanstalk 節點和您的應用程式節點。
2. 在 AWS Explorer 中以滑鼠右鍵按一下您的 Elastic Beanstalk 環境，然後按一下 View Status (檢視狀態)。
3. 在您的應用程式環境分頁中，按一下 Events (事件)。



使用部署工具在 .NET 中部署 Elastic Beanstalk 應用程式

AWS Toolkit for Visual Studio 內含的部署工具為命令列工具，可提供與 AWS 工具組中部署精靈相同的功能。您可將此部署工具用於您的建置管道或其他指令碼，自動化對 Elastic Beanstalk 的部署作業。

此部署工具同時支援初始部署和重新部署。若您之前曾使用部署工具來部署應用程式，可使用 Visual Studio 內的部署精靈來重新部署。同樣的，若您已使用精靈來部署，亦可使用部署工具來重新部署。

> **Note**
>
> 部署工具不像主控台或 EB CLI 會為組態選項套用建議值。使用組態檔案來確認在您啟動環境時，所需設定都已完成。

此章節會引導您使用部署工具將範例 .NET 應用程式部署至 Elastic Beanstalk，然後運用增量式部署來重新部署該應用程式。如需部署工具的深入討論 (包括參數選項)，請參閱部署工具。

先決條件

若要使用部署工具，您需要安裝 AWS Toolkit for Visual Studio。如需事前準備和安裝說明的資訊，請參閱 AWS Toolkit for Microsoft Visual Studio。

部署工具在 Windows 中通常安裝於下列其中一個目錄：

32 位元	64 位元
`C:\Program Files\AWS Tools\Deployment Tool\awsdeploy.exe`	`C:\Program Files (x86)\AWS Tools\Deployment Tool\awsdeploy.exe`

部署到 Elastic Beanstalk

欲使用部署工具將範例應用程式部署至 Elastic Beanstalk，首先需要修改 `ElasticBeanstalkDeploymentSample.txt` 目錄中提供的 `Samples` 組態檔案。此組態檔案內含部署您應用程式所需的資訊，包括應用程式名稱、應用程式版本、環境名稱和您的 AWS 存取憑證。修改組態檔案後，請使用命令列來部署範例應用程式。您的 Web 部署檔案會上傳至 Amazon S3，且會向 Elastic Beanstalk 註冊為新的應用程式版本。部署您的應用程式需要幾分鐘的時間。環境運作狀態正常後，部署工具就會為正在執行的應用程式輸出 URL。

將 .NET 應用程式部署至 Elastic Beanstalk

1. 從部署工具安裝的 `Samples` 子目錄中，開啟 `ElasticBeanstalkDeploymentSample.txt`，然後如下列範例輸入您的 AWS 存取金鑰和 AWS 私密金鑰。

```
### AWS Access Key and Secret Key used to create and deploy the application
 instance
```

```
AWSAccessKey = AKIAIOSFODNN7EXAMPLE
AWSSecretKey = wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

> **Note**
>
> 如果是 API 存取，您需要存取金鑰 ID 和私密存取金鑰。使用 IAM 使用者存取金鑰，而非 AWS 帳戶根使用者 存取金鑰。如需建立存取金鑰的詳細資訊，請參閱IAM 使用者指南中的管理 IAM 使用者的存取金鑰。

2. 在命令列提示時，輸入下列：

```
C:\Program Files (x86)\AWS Tools\Deployment Tool>awsdeploy.exe /w Samples
\ElasticBeanstalkDeploymentSample.txt
```

部署您的應用程式需要幾分鐘的時間。若部署成功，將看到訊息 `Application deployment completed; environment health is Green`。

> **Note**
>
> 若收到下列錯誤，則該 CNAME 已存在。
>
> ```
> [Error]: Deployment to AWS Elastic Beanstalk failed with exception: DNS name
>  (MyAppEnv.elasticbeanstalk.com) is not available.
> ```
>
> 由於 CNAME 必須為獨一無二的，您必須變更 `Environment.CNAME` 中的 `ElasticBeanstalkDeploymentSample.txt`。

3. 在 Web 瀏覽器中，導覽至您執行中應用程式的 URL。該 URL 的格式為 <CNAME.elasticbeanstalk.com> (如 **`MyAppEnv.elasticbeanstalk.com`**)。

將您的內部部署 .NET 應用程式遷移至 Elastic Beanstalk

如果您考慮要將 .NET 應用程式從內部部署伺服器遷移至 Amazon Web Services (AWS)，.NET Migration Assistant for AWS Elastic Beanstalk 可能適合您使用。此助理是一個互動式 PowerShell 公用程式，可將 .NET 應用程式從在內部部署環境執行 IIS 的 Windows Server 遷移至 AWS Elastic Beanstalk。此助理只要進行最少的變更 (或無需任何變更)，即可將整個網站遷移至 Elastic Beanstalk。

如需 .NET Migration Assistant for AWS Elastic Beanstalk 及其下載位置的詳細資訊，請參閱 GitHub 儲存庫 https://github.com/awslabs/windows-web-app-migration-assistant。

如果應用程式包含 Microsoft SQL Server 資料庫，GitHub 上的助理相關文件則會包含進行遷移的多個選項。

將 Node.js 應用程式部署到 Elastic Beanstalk

AWS Elastic Beanstalk Node.js 可讓您使用亞馬遜網路服務輕鬆部署、管理和擴展 Node.js 網路應用程式。所有使用 Node.js 來開發或託管 Web 應用程式的人，都能使用 Elastic Beanstalk for Node.js。本章提供將 Node.js Web 應用程式部署至 Elastic Beanstalk 的 step-by-step 指示，並提供資料庫整合和使用 Express 架構等常見工作的逐步解說。

部署 Elastic Beanstalk 應用程式之後，您可以繼續使用 EB CLI 來管理您的應用程式和環境，也可以使用 Elastic Beanstalk 主控台或 API。 AWS CLI

主題

QuickStart：將 Node.js 應用程式部署到 Elastic Beanstalk

本 QuickStart 教學課程將引導您完成建立 Node.js 應用程式並將其部署至 AWS Elastic Beanstalk 環境的程序。

> **Note**
>
> 本 QuickStart 自學課程用於示範目的。請勿將本教學課程中建立的應用程式用於生產流量。

章節

您的 AWS 帳戶

如果您還不是 AWS 客戶，則需要創建一個 AWS 帳戶。註冊使您可以訪問 Elastic Beanstalk 和您需要的其他 AWS 服務。

如果您已經有 AWS 帳戶，則可以轉到必要條件。

創建一個 AWS 帳戶

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。

   當您註冊時 AWS 帳戶，會建立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 AWS 帳戶 AWS 帳戶根使用者、啟用和建立系統管理使用者 AWS IAM Identity Center，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」。

必要條件

為了遵循本指南的程序，您需要命令列終端機或 shell 來執行命令。命令清單前會出現提示字元 ($) 及目前的目錄名稱 (如有)。

```
~/eb-project$ this is a command
this is output
```

在 Linux 和 macOS 上，您可以使用偏好的 Shell 和套件軟體管理工具。在視窗上，你可以安裝視窗子系統為 Linux 得到一個視窗集成的 Ubuntu 和 Bash 的版本。

EB CLI

本教學使用 Elastic Beanstalk 命令列界面 (EB CLI)。關於安裝和設定 EB CLI 的詳細資訊，請參閱安裝 EB CLI 和設定 EB CLI。

Node.js

按照以下操作在本地計算機上安裝 Node.js 如何在 Node.js 網站上安裝 Node.js。

執行下列命令來驗證您的 Node.js 安裝。

```
~$ node -v
```

第 1 步：創建一個 Node.js 應用程序

建立專案目錄。

```
~$ mkdir eb-nodejs
~$ cd eb-nodejs
```

接著，請建立您將使用 Elastic Beanstalk 進行部署的應用程式。我們將建立一個「Hello World」RESTful Web 服務。

Example **~/eb-nodejs/server.js**

```
const http = require('node:http');

const hostname = '127.0.0.1';
const port = 8080;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello Elastic Beanstalk!\n');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

此應用程序在端口 8080 上打開一個監聽器。默認情況下，針對 Node.js，Elastic Beanstalk 會將請求轉發到端口 8080 上的應用程序。

步驟 2：在本機執行應用程式

執行下列命令以在本機執行應用程式。

```
~/eb-nodejs$ node server.js
```

您應該會看到以下文字。

```
Server running at http://127.0.0.1:8080/
```

在網頁瀏覽器`http://127.0.0.1:8080/`中輸入 URL 位址。瀏覽器應該顯示「你好 Elastic Beanstalk！」。

步驟 3：使用 EB CLI 部署您的 Node.js 應用程式

執行下列命令，為此應用程式建立 Elastic Beanstalk 環境。

若要建立環境並部署您的 Node.js 應用程式

1. 透過 eb init 命令初始化您的 EB CLI 儲存庫。

```
~/eb-nodejs$ eb init -p node.js nodejs-tutorial --region us-east-2
```

此命令會建立名為的應用程式，nodejs-tutorial並設定您的本機存放庫，以建立具有最新 Node.js 平台版本的環境。

2. (選用) 再次執行 eb init 來設定預設金鑰對，藉此使用 SSH 連接至執行您應用程式的 EC2 執行個體：

```
~/eb-nodejs$ eb init
Do you want to set up SSH for your instances?
(y/n): y
Select a keypair.
1) my-keypair
2) [ Create new KeyPair ]
```

若您已有金鑰對，請選擇一對，或依照提示建立金鑰對。若未出現提示，或稍後需要變更設定，請執行 eb init -i。

3. 使用 eb create 建立環境並於其中部署您的應用程式。Elastic Beanstalk 會自動為您的應用程式建立一個壓縮檔案，並將其部署到環境中的 EC2 執行個體。部署應用程序後，Elastic Beanstalk 在端口 8080 上啟動它。

```
~/eb-nodejs$ eb create nodejs-env
```

Elastic Beanstalk 大約需要五分鐘的時間來創建您的環境。

第 4 步：在 Elastic Beanstalk 上運行應用程序

當創建環境的過程完成後，打開您的網站eb open。

```
~/eb-nodejs$ eb open
```

恭喜您！您已經部署了一個帶有 Elastic Beanstalk 的 Node.js 應用程序！這會開啟瀏覽器視窗，並使用為應用程式建立的網域名稱。

步驟 5：清除

您可以在完成應用程式的工作後終止環境。Elastic Beanstalk 會終止與您環境相關的所有 AWS 資源。

若要使用 EB CLI 終止 Elastic Beanstalk 環境，請執行下列命令。

```
~/eb-nodejs$ eb terminate
```

AWS 您應用程式的資源

您剛剛建立了單一執行個體應用程式。它可作為單一 EC2 執行個體的簡單範例應用程式使用，因此不需要負載平衡或 auto 擴展。對於單個實例應用程序，Elastic Beanstalk 創建以下 AWS 資源：

- EC2 執行個體 – 設定在您所選平台上執行 Web 應用程式的 Amazon EC2 虛擬機器。

  每個平台會執行不同一套軟體、設定檔和指令碼，來支援特定的語言版本、架構、Web 容器或其組合。大多數的平台會使用 Apache 或 nginx 做為反向代理，處理您 Web 應用程式前端的網路流量、向它轉送請求、提供靜態資產，並產生存取和錯誤日誌。
- 執行個體安全群組 – 設定允許從連接埠 80 傳入流量的 Amazon EC2 安全群組。此資源可讓負載平衡器傳來的 HTTP 傳輸資料，到達執行您 Web 應用程式的 EC2 執行個體。在預設情況下，不允許傳輸資料從其他通訊埠傳送。
- Amazon S3 儲存貯體 - 儲存位置，用來儲存當您使用 Elastic Beanstalk 時所建立的原始程式碼、日誌和其他成品。
- Amazon CloudWatch 警示 — 兩個 CloudWatch 警示，用於監控環境中執行個體的負載，並在負載過高或過低時觸發。當警示觸發時，您的 Auto Scaling 群組會擴展或縮減以進行回應。
- AWS CloudFormation 堆疊 — Elastic Beanstalk 用 AWS CloudFormation 來啟動環境中的資源並傳播組態變更。資源定義於範本中，您可在 AWS CloudFormation 主控台中檢視此範本。
- 網域名稱 – 會路由到您 Web 應用程式的網域名稱，其格式為 *subdomain*.*region*.elasticbeanstalk.com。

Elastic Beanstalk 會管理所有這些資源。當您終止環境時，Elastic Beanstalk 會終止其中的所有資源。

後續步驟

在您擁有執行應用程式的環境後，可以隨時部署應用程式的新版本或不同的應用程式。部署新的應用程式版本非常快速，因無須佈建或重新啟動 EC2 執行個體。您也可以使用 Elastic Beanstalk 控制台探索您的新環境。如需詳細步驟，請參閱本指南的「入門」一章中的「探索您的環境」。

> **嘗試更多教程**
>
> 如果您想要嘗試使用不同範例應用程式的其他教學課程，請參閱Node.js 的更多示例應用程序和教程。

在您部署一或兩個範例應用程式並準備好開始在本機開發和執行 Node.js 應用程式之後，請參閱設定您的 Node.js 開發環境。

使用 Elastic Beanstalk 控制台進行部署

您也可以使用 Elastic Beanstalk 控制台啟動示例應用程序。如需詳細步驟，請參閱本指南的「入門」一章中的「建立範例應用程式」。

設定您的 Node.js 開發環境

設定 Node.js 開發環境於本機測試您的應用程式，之後再部署至 AWS Elastic Beanstalk。此主題概述開發環境設定步驟，並提供實用工具的安裝頁面連結。

如需了解適用所有語言的常見設定步驟和工具，請參閱設定您的開發機器。

主題

- 安裝 Node.js。
- 確認 npm 安裝
- 安裝適用於 Node.js 的 AWS 開發套件
- 安裝 Express 產生器
- 設定 Express 架構與伺服器

安裝 Node.js。

安裝 Node.js 以於本機執行 Node.js 應用程式。若您沒有特別需求，請取得 Elastic Beanstalk 支援的最新版本。如需支援版本的清單，請參閱 AWS Elastic Beanstalk 平台文件中的 Node.js。

至 nodejs.org 下載 Node.js。

確認 npm 安裝

Node.js 使用 npm 套裝管理員，助您安裝欲在應用程式使用的工具及架構。由於 npm 與 Node.js 一起分發，您將在下載並安裝 Node.js 時自動對其進行安裝。要確認您已安裝 npm，您可以運行以下命令：

```
$ npm -v
```

有關 npm 的更多信息，請瀏覽 npmjs 網站。

安裝適用於 Node.js 的 AWS 開發套件

若您需要從應用程式內管理 AWS 資源，請安裝適用於 Node.js 中 JavaScript 的 AWS 開發套件。使用 npm 安裝 SDK：

```
$ npm install aws-sdk
```

如需詳細資訊，請造訪適用於 Node.js 中 JavaScript 的 AWS 開發套件首頁。

安裝 Express 產生器

Express 為執行 Node.js 的 Web 應用程式架構。若要使用，請先安裝 Express 產生器命令列應用程式。安裝 Express 產生器後，即可執行 express 命令，為您的 Web 應用程式產生基本專案結構。安裝基礎專案、檔案和相依性後，即可在開發機器上啟動本機 Express 伺服器。

> **Note**
>
> - 下列步驟將引導您在 Linux 作業系統上安裝 Express 產生器。
> - 對於 Linux，根據您的系統目錄許可層級，您可能需要透過 sudo 為部分命令增加字首。

在開發環境中安裝 Express 產生器

1. 為您的 Express 架構和伺服器建立工作目錄。

   ```
   ~$ mkdir node-express
   ~$ cd node-express
   ```

2. 全面安裝 Express，讓您能夠存取 `express` 命令。

```
~/node-express$ npm install -g express-generator
```

3. 依據您的作業系統而定，可能需要設定路徑以執行 `express` 命令。如果您需要設定路徑變數，上一個步驟的輸出可提供相關資訊。以下是 Linux 範例。

```
~/node-express$ export PATH=$PATH:/usr/local/share/npm/bin/express
```

若您遵循本章的教學課程，您將需要從不同的目錄執行 express 命令。每個教學課程都會在其自身目錄中設置基本 Express 專案結構。

您現在已經安裝 Express 命令列產生器。您可以用其為 Web 應用程式建立架構目錄、設定相依性，以及啟動 Web 應用程式伺服器。接下來，我們將執行步驟，在建立的 `node-express` 目錄中達成此目標。

設定 Express 架構與伺服器

請依照下列步驟建立基本 Express 架構目錄和內容。本章教學課程也包含此類步驟，以便在每個教學課程的應用程式目錄中設定基本 Express 架構。

設定 Express 架構與伺服器

1. 執行 `express` 命令。這會產生 `package.json`、`app.js` 以及幾個目錄。

```
~/node-express$ express
```

當提示您是否要繼續時，請輸入 **y**。

2. 設定本機依存項目。

```
~/node-express$ npm install
```

3. 確認 Web 應用程式伺服器已啟動。

```
~/node-express$ npm start
```

您應該會看到類似下列的輸出：

```
> nodejs@0.0.0 start /home/local/user/node-express
```

```
> node ./bin/www
```

依預設，伺服器將會在連接埠 3000 上執行。若要進行測試，請在另一部終端機上執行 `curl http://localhost:3000`，或在本機電腦開啟瀏覽器並輸入 URL 位址 `http://localhost:3000`。

按 Ctrl+C 來停止伺服器。

使用 Elastic Beanstalk Node.js 平台

AWS Elastic Beanstalk Node.js 平台是一組平台版本，用於在 NGINX 代理伺服器後方執行的 Node.js Web 應用程式。

Elastic Beanstalk 提供組態選項，您可以用來自訂在 Elastic Beanstalk 環境中 EC2 執行個體上執行的軟體。您可以設定應用程式所需的環境變數，啟用 Amazon S3 的日誌輪換，並將包含靜態檔案的應用程式來源中的資料夾映射到代理伺服器提供的路徑。

Elastic Beanstalk 主控台中提供了修改正在執行環境組態的組態選項。要避免在終止環境的組態時遺失組態，您可以使用已儲存組態來儲存您的設定，並在之後套用至另一個環境。

若要將設定儲存於原始程式碼，您可以包含組態檔案。每次您建立環境或部署應用程式，組態檔案裡的設定就會套用。您也可以使用組態檔案來安裝套件、執行指令碼，並在部署期間執行其他執行個體自訂操作。

您可以在原始碼套件中包含一個 `Package.json` 檔案，以便在部署期間安裝套件、提供啟動命令，以及指定您希望應用程式使用的 Node.js 版本。您可以包含一個 `npm-shrinkwrap.json` 檔案來鎖定相依性版本。

Node.js 平台包含一個代理伺服器，以為靜態資產提供服務、將流量轉發至您的應用程式，以及壓縮回應資料。您可以針對進階的應用情境，來擴展或覆寫預設的代理組態。

有幾個選項可以啟動您的應用程序。您可以將 Procfile 新增到原始碼套件中，以指定啟動應用程式的命令。如果您未提供 `Procfile` 但提供 `package.json` 檔案，則 Elastic Beanstalk 會執行 `npm start`。如果您也沒有提供該檔案，則 Elastic Beanstalk 會依序尋找 `app.js` 或 `server.js` 檔案，並執行程式碼。

在 Elastic Beanstalk 主控台中套用的設定會覆寫組態檔案中相同的設定 (如存在)。這可讓您在組態檔案中擁有預設設定，並以主控台的環境專屬設定覆寫之。如需優先順序以及其他變更設定方法的詳細資訊，請參閱組態選項。

如需各種擴充 Elastic Beanstalk Linux 類型平台方式的詳細資訊，請參閱 the section called “擴充 Linux 平台”。

設定您的 Node.js 環境

您可以使用 Node.js 平台設定來微調 Amazon EC2 執行個體的行為。您可以使用 Elastic Beanstalk 主控台編輯 Elastic Beanstalk 環境的 Amazon EC2 執行個體組態。

使用 Elastic Beanstalk 主控台來啟用至 Amazon S3 的日誌輪換，和設定您的應用程式可以從環境讀取的變數。

在 Elastic Beanstalk 主控台中設定 Node.js 環境

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇組態。
4. 在更新、監控和日誌記錄組態類別中，選擇編輯。

容器選項

您可以指定這些平台特有的選項：

- 代理伺服器 – 要在您的環境執行個體上使用的代理伺服器。預設為使用 NGNIX。

日誌選項

Log Options (日誌選項) 區段有兩個設定：

- Instance profile (執行個體描述檔) – 指定執行個體描述檔，此描述檔具有的許可，可存取和您應用程式相關的 Amazon S3 儲存貯體。
- Enable log file rotation to Amazon S3 (啟用 Amazon S3 的日誌檔案輪換) – 指定是否將應用程式 Amazon EC2 執行個體的日誌檔案複製到與應用程式關聯的 Amazon S3 儲存貯體。

靜態檔案

為改善效能，您可以使用 Static files (靜態檔案) 區段來設定代理伺服器，以為來自 Web 應用程式一組目錄中的靜態檔案 (例如 HTML 或影像) 提供服務。對於每個目錄，您可以設定目錄映射的虛擬路徑。代理伺服器收到位於指定路徑下的檔案請求時，會直接提供檔案而非將請求路由至您的應用程式。

如需使用組態檔案或 Elastic Beanstalk 主控台設定靜態檔案的詳細資訊，請參閱the section called “靜態檔案”。

環境屬性

使用 Environment Properties (環境屬性) 部分針對執行您應用程式的 Amazon EC2 執行個體，來指定其上的環境資訊設定。這些設定會以金鑰值對的形式傳到應用程式。

在 AWS Elastic Beanstalk 中執行的 Node.js 環境內，您可以藉由執行 `process.env.ENV_VARIABLE` 來存取環境變數。

```
var endpoint = process.env.API_ENDPOINT
```

Node.js 平台會將 PORT 環境變數設定為代理伺服器傳送流量至其中的連接埠。如需詳細資訊，請參閱設定代理伺服器。

如需詳細資訊，請參閱環境屬性與其他軟體設定。

設定 Amazon Linux AMI (先前的 Amazon Linux 2) Node.js 環境

只有使用 Amazon Linux AMI 平台版本 (先前的 Amazon Linux 2) 的 Elastic Beanstalk Node.js 環境才支援以下主控台軟體組態類別。

> **備註**
>
> - 本主題中的資訊僅適用於以 Amazon Linux AMI (AL1) 為基礎的平台分支。AL2023/AL2 平台分支與舊版 Amazon Linux AMI (AL1) 平台版本不相容，需要不同的組態設定。
> - 2022 年 7 月 18 日，Elastic Beanstalk 已將所有以 Amazon Linux AMI (AL1) 為基礎的平台分支狀態設為已淘汰。如需有關遷移至完全支援的目前 Amazon Linux 2023 平台分支的詳細資訊，請參閱 將您的 Elastic Beanstalk Linux 應用程式遷移到 Amazon Linux 2023 或 Amazon Linux 2。

容器選項 — Amazon Linux AMI (AL1)

在組態頁面上指定下列項目：

- Proxy server (代理伺服器) – 指定要使用哪一個 Web 伺服器來將代理連線連至 Node.js。預設為使用 NGINX。如果選擇 none (無)，則靜態檔案映射不會生效，而且會停用 GZIP 壓縮功能。
- Node.js 版本 – 指定 Node.js 的版本。如需支援的 Node.js 版本清單，請參閱 AWS Elastic Beanstalk 平台指南中的 Node.js。
- GZIP 壓縮 – 指定是否啟用 GZIP 壓縮。預設為啟用 GZIP 壓縮。
- Node command (節點命令) – 讓您能輸入用來啟動 Node.js 應用程式的命令。空白字串 (預設值) 表示 Elastic Beanstalk 將會依序使用 `app.js`、`server.js` 和 `npm start`。

Node.js 組態命名空間

您可以使用組態檔案來設定組態選項，並在部署期間執行其他的執行個體設定工作。組態選項可透過 Elastic Beanstalk 服務或您使用的平台來定義，並且會組織成命名空間。

您可以使用 `aws:elasticbeanstalk:environment:proxy` 命名空間來選擇要在環境的執行個體上使用的代理。以下範例會將您的環境設定為使用 Apache HTTPD 代理伺服器。

Example .ebextensions/nodejs-settings.config

```
option_settings:
  aws:elasticbeanstalk:environment:proxy:
    ProxyServer: apache
```

您可以使用 `aws:elasticbeanstalk:environment:proxy:staticfiles` 命名空間設定代理提供靜態檔案。如需詳細資訊和範例，請參閱 the section called “靜態檔案”。

Elastic Beanstalk 可提供許多組態選項讓您自訂環境。除了組態檔案，您也可以使用主控台、已儲存組態、EB CLI 或 AWS CLI 來設定組態選項。如需詳細資訊，請參閱組態選項。

Amazon Linux AMI (先前的 Amazon Linux 2) Node.js 平台

如果您的 Elastic Beanstalk Node.js 環境使用 Amazon Linux AMI 平台版本 (先前的 Amazon Linux 2)，請考慮本節中的特定組態和建議。

> **備註**
>
> - 本主題中的資訊僅適用於以 Amazon Linux AMI (AL1) 為基礎的平台分支。AL2023/AL2 平台分支與舊版 Amazon Linux AMI (AL1) 平台版本不相容，需要不同的組態設定。
> - 2022 年 7 月 18 日，Elastic Beanstalk 已將所有以 Amazon Linux AMI (AL1) 為基礎的平台分支狀態設為已淘汰。如需有關遷移至完全支援的目前 Amazon Linux 2023 平台分支的詳細資訊，請參閱 將您的 Elastic Beanstalk Linux 應用程式遷移到 Amazon Linux 2023 或 Amazon Linux 2。

Node.js 平台特定組態選項 — Amazon Linux AMI (AL1)

Elastic Beanstalk 支援適用於 Amazon Linux AMI Node.js 平台版本的一些平台特定組態。您可以選擇要在應用程式前景執行的代理伺服器、選擇要執行的特定 Node.js 版本，並選擇用來執行應用程式的命令。

對於代理伺服器，您可以使用 NGINX 或 Apache 代理伺服器。您可以將 none 值設定為 `ProxyServer` 選項。使用此設定，Elastic Beanstalk 會獨立執行您的應用程式，而不會在任何代理伺服器後方執行。如果您的環境執行獨立應用程式，請更新您的程式碼以接聽 NGINX 將流量轉發至其中的連接埠。

```
var port = process.env.PORT || 8080;

app.listen(port, function() {
  console.log('Server running at http://127.0.0.1:%s', port);
});
```

Node.js 語言版本 — Amazon Linux AMI (AL1)

在支援的語言版本方面，Node.js Amazon Linux AMI 平台與其他由 Elastic Beanstalk 管理的平台不同。這是因為每個 Node.js 平台版本僅支援幾個 Node.js 語言版本。如需支援的 Node.js 版本清單，請參閱 AWS Elastic Beanstalk 平台指南中的 Node.js。

您可以使用平台特定的組態選項來設定語言版本。如需說明，請參閱 the section called “設定您的 Node.js 環境”。或者，使用 Elastic Beanstalk 主控台來更新您的環境使用的 Node.js 版本，作為更新平台版本的一部分。

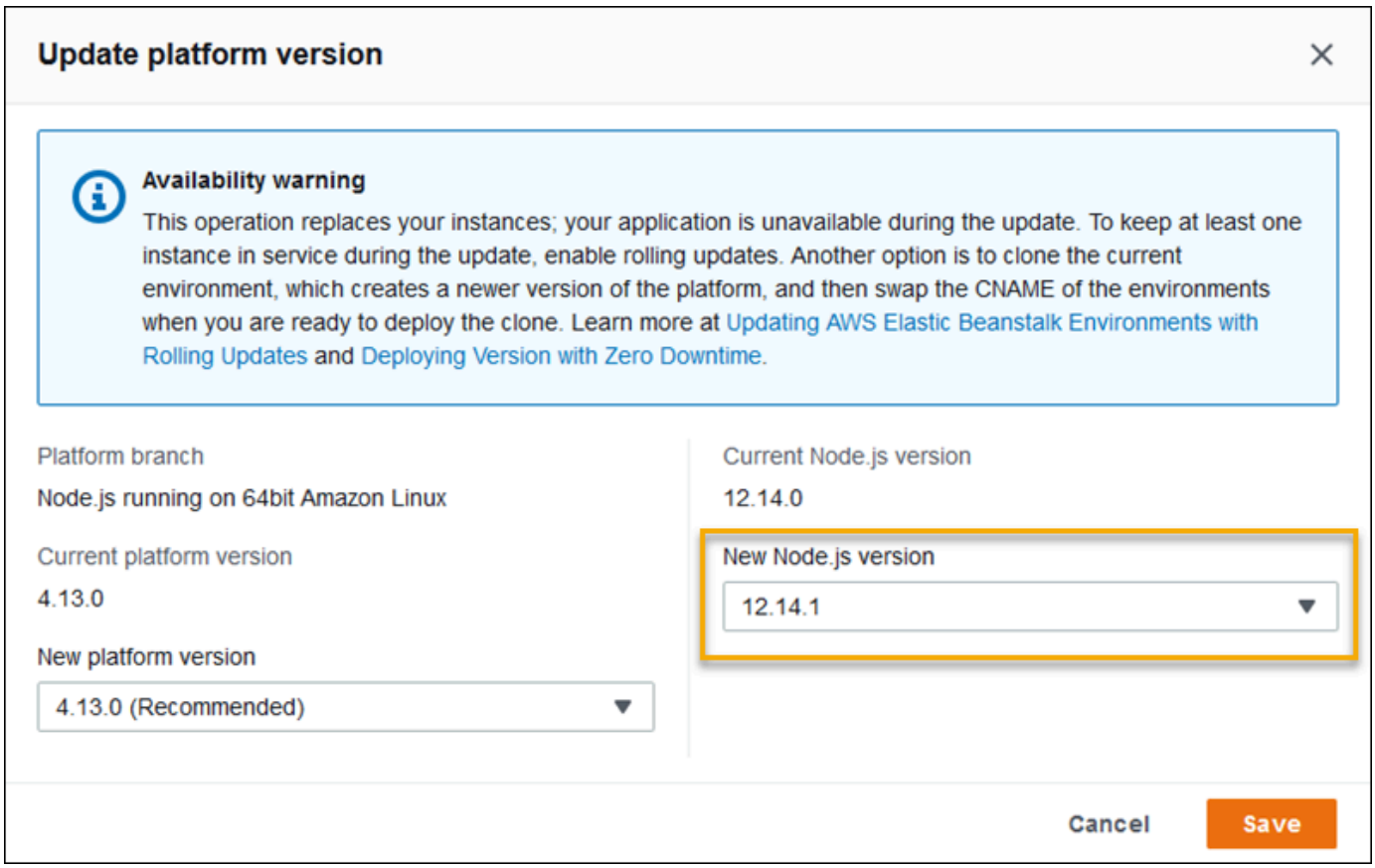
> **Note**
>
> 如果您所使用的 Node.js 版本的支援已從平台移除，您必須在平台更新之前變更或移除版本設定。當發現一個或多個版本的 Node.js 存在安全漏洞時，可能會出現此種狀況。
> 此時，如果嘗試將不支援已設定 NodeVersion 的平台更新至新版本，此動作將會失敗。為省去建立新環境的必要，請將 NodeVersion 組態選項變更為新舊平台版本皆支援的 Node.js 版本，或是移除選項設定，然後執行平台更新。

在 Elastic Beanstalk 主控台中設定環境的 Node.js 版本

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

   > **Note**
   >
   > 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在環境概觀頁面的 Platform (平台) 下，選擇 Change (變更)。
4. 在 Update platform version (更新平台版本) 對話方塊中，選取 Node.js 平台版本。



5. 選擇 Save (儲存)。

Node.js 組態命名空間 — Amazon Linux AMI (AL1)

Node.js Amazon Linux AMI 平台在 `aws:elasticbeanstalk:container:nodejs:staticfiles` 和 `aws:elasticbeanstalk:container:nodejs` 命名空間中定義其他選項。

下列組態檔案會告知 Elastic Beanstalk 使用 `npm start` 來執行應用程式。還將代理類型設定為 Apache 並啟用壓縮。最後，它設定代理從兩個來源目錄提供靜態檔案。其中一個來源是來自 `statichtml` 來源目錄，在網站根目錄下 `html` 路徑中的 HTML 檔案。另一個來源是來自 `staticimages` 來源目錄的網站根目錄下的 `images` 路徑的映像檔案。

Example .ebextensions/node-settings.config

```
option_settings:
  aws:elasticbeanstalk:container:nodejs:
    NodeCommand: "npm start"
    ProxyServer: apache
    GzipCompression: true
```

```
  aws:elasticbeanstalk:container:nodejs:staticfiles:
    /html: statichtml
    /images: staticimages
```

Elastic Beanstalk 可提供許多組態選項讓您自訂環境。除了組態檔案，您也可以使用主控台、已儲存組態、EB CLI 或 AWS CLI 來設定組態選項。如需詳細資訊，請參閱組態選項。

使用 Procfile 設定應用程式程序

您可以在原始碼套件的根目錄中包含名為 `Procfile` 的檔案，以指定啟動應用程序的命令。

Example Procfile

```
web: node index.js
```

如需有關 `Procfile` 用量的資訊，請展開 the section called “擴充 Linux 平台” 中的 Buildfile 和 Procfile 區段。

> **Note**
>
> 此功能會取代 `aws:elasticbeanstalk:container:nodejs` 命名空間中的舊版 `NodeCommand` 選項。

安裝您應用程式的相依性

您的應用程式可能與一些 Node.js 模組有相依性，例如您在 `require()` 陳述式中指定的模組。這些模組儲存在 `node_modules` 目錄中。當您的應用程式執行時，Node.js 會從此目錄載入模組。如需詳細資訊，請參閱 Node.js 文件中的從 node_modules 資料夾載入。

您可以使用 `package.json` 檔案來指定這些模組相依性。如果 Elastic Beanstalk 偵測到此檔案且 `node_modules` 目錄不存在，Elastic Beanstalk 會以 webapp 使用者的身分執行 `npm install`。`npm install` 命令將相依性安裝在 Elastic Beanstalk 事先建立的 `node_modules` 目錄中。`npm install` 命令會從公共 npm 登錄檔或其他位置存取 `package.json` 檔案中列出的套件。如需詳細資訊，請參閱 npm Docs 網站。

如果 Elastic Beanstalk 偵測到 `node_modules` 目錄，則即使有 `package.json` 檔案，Elastic Beanstalk 也不會執行 `npm install`。Elastic Beanstalk 假設 `node_modules` 目錄中有可用的相依性套件，以供 Node.js 存取和載入。

以下章節提供為應用程式建立 Node.js 模組相依性的詳細資訊。

> **Note**
>
> 如果您在 Elastic Beanstalk 執行 `npm install` 時遭遇任何部署問題，請考慮使用其他方法。將 `node_modules` 目錄以及相依性模組納入您的應用程式原始碼套件中。這樣做可以避免在排查問題時，安裝來自公共 npm 登錄檔的相依性的問題。而由於相依性模組來自本機目錄，所以這可能也有助於縮短部署時間。如需詳細資訊，請參閱 在 a node_modules 目錄中納入 Node.js 相依性

使用 package.json 檔案指定 Node.js 相依性

將 `package.json` 檔案包含在專案來源根目錄中來指定相依性套件並提供啟動命令。當有 `package.json` 檔案存在且您專案來源的根目錄中沒有 `node_modules` 目錄時，Elastic Beanstalk 會以 webapp 使用者的身分執行 `npm install`，以安裝來自公共 npm 登錄檔的相依性。Elastic Beanstalk 也會使用 `start` 命令來啟動您的應用程式。如需有關 `package.json` 檔案的詳細資訊，請參閱 npm Docs 網站中的《在 `package.json` 檔案中指定相依性》。

使用 `scripts` 關鍵字來提供啟動命令。現在，在 `aws:elasticbeanstalk:container:nodejs` 命名空間中使用 `scripts` 關鍵字來取代舊版的 `NodeCommand` 選項。

Example package.json – Express

```
{
    "name": "my-app",
    "version": "0.0.1",
    "private": true,
    "dependencies": {
      "ejs": "latest",
      "aws-sdk": "latest",
      "express": "latest",
      "body-parser": "latest"
    },
    "scripts": {
      "start": "node app.js"
    }
  }
```

生產模式和開發相依性

若要在 package.json 檔案中指定相依性，請使用 dependencies 和 devDependencies 屬性。dependencies 屬性可指定生產中應用程式所需的套件。devDependencies 屬性可指定只需要用於本機開發與測試的套件。

Elastic Beanstalk 會使用下列命令，以 webapp 使用者的身分執行 npm install。命令選項會根據應用程式執行之平台分支上所包含的 npm 版本而有所不同。

- npm v6 — 在預設情況下，Elastic Beanstalk 會在生產模式中安裝相依性。其使用命令 npm install --production。
- npm v7 或更高版本 — Elastic Beanstalk 會省略 devDependencies。其使用命令 npm install --omit=dev。

上述兩個命令都不會安裝屬於 devDependencies 的套件。

如果您需要安裝 devDependencies 套件，請將 PM_USE_PRODUCTION 環境屬性設為 false。透過此設定，我們不會在執行 npm 安裝時使用上述選項。這將導致安裝 devDependencies 套件。

SSH 及 HTTPS

從 2023 年 3 月 7 日 Amazon Linux 2 平台版本開始，您還可以使用 SSH 和 HTTPS 通訊協定從 Git 儲存庫擷取套件。平台分支 Node.js 16 同時支援 SSH 和 HTTPS 通訊協定。Node.js 14 僅支援 HTTPS 通訊協定。

Example package.json – Node.js 16 同時支援 HTTPS 和 SSH

```
...
"dependencies": {
  "aws-sdk": "https://github.com/aws/aws-sdk-js.git",
  "aws-chime": "git+ssh://git@github.com:aws/amazon-chime-sdk-js.git"
}
```

版本和版本範圍

> **Important**
>
> 指定版本範圍的功能不適用於在 AL2023 上執行的 Node.js 平台分支。在 AL2023 上特定的 Node.js 分支內，只支援一個 Node.js 版本。如果 package.json 檔案指定了版本範圍，我們將予以忽略，並預設為 Node.js 的平台分支版本。

在 `package.json` 檔案中使用 `engines` 關鍵字，來指定您希望應用程式使用的 Node.js 版本。您也可以使用 npm 標記法指定版本範圍。如需有關版本範圍語法的詳細資訊，請參閱 Node.js 網站上的 Semantic Versioning using npm (使用 npm 的語意版本控制)。Node.js `package.json` 檔案中的 `engines` 關鍵字取代 `aws:elasticbeanstalk:container:nodejs` 命名空間中的舊版 `NodeVersion` 選項。

Example **`package.json`** –單一 Node.js 版本

```
{
    ...
    "engines": { "node" : "14.16.0" }
  }
```

Example **`package.json`** – Node.js 版本範圍

```
{
    ...
    "engines": { "node" : ">=10 <11" }
  }
```

在有指出版本範圍的情況下，Elastic Beanstalk 會安裝平台在範圍內可用的最新 Node.js 版本。在此範例中，範圍指出版本必須大於或等於版本 10，但小於版本 11。因此，Elastic Beanstalk 會安裝最新的 Node.js 版本 10.x.y，可在《支援的平台》上取得此版本。

請注意，您只能指定一個與您平台分支對應的 Node.js 版本。例如，如果您使用的是 Node.js 16 平台分支，您只能指定 16.x.y Node.js 版本。您可以使用 npm 支援的版本範圍選項以允許更多的靈活性。關於每個平台分支的有效 Node.js 版本，請參閱 AWS Elastic Beanstalk 平台指南中的 Node.js。

> **Note**
>
> 如果您所使用的 Node.js 版本的支援已從平台移除，您必須在平台更新之前變更或移除 Node.js 版本設定。當發現一個或多個版本的 Node.js 存在安全漏洞時，可能會出現此種狀況。
>
> 此時，如果嘗試將不支援已設定 Node.js 版本的平台更新至新版本，此動作將會失敗。為省去建立新環境的必要，請將 `package.json` 中的 Node.js 版本設定變更為新舊平台版本皆支援的 Node.js 版本。您可以選擇指定包含支援版本的 Node.js 版本範圍，如本主題之前所述。您也可以選擇移除設定，然後部署新的來源套件。

在 a node_modules 目錄中納入 Node.js 相依性

若要將相依性套件與應用程式程式碼一起部署至環境執行個體，請將它們包含在專案來源根中名為 `node_modules` 的目錄內。如需詳細資訊，請參閱 npm Docs 網站中的《在本機下載和安裝套件》。

當您將 `node_modules` 目錄部署到 Amazon Linux 2 Node.js 平台版本時，Elastic Beanstalk 會假設您提供自己的相依性套件，並避免安裝 package.json 檔案中指定的相依性。Node.js 會在 `node_modules` 目錄中找尋相依性。如需詳細資訊，請參閱 Node.js 文件中的從 node_modules 資料夾載入。

> **Note**
>
> 如果您在 Elastic Beanstalk 執行 `npm install` 時遭遇任何部署問題，在調查問題時，請考慮使用本主題中描述的方法作為變通措施。

使用 npm shrinkwrap 鎖定相依性

Node.js 平台會在您每次部署時以 webapp 使用者身分執行 `npm install`。當您的相依檔案存在可用的新版本時，則會在您部署應用程式時安裝這些版本，故可能造成部署作業需要長時間才能完成。

您可以藉由建立 `npm-shrinkwrap.json` 檔案，來將應用程式的相依檔案鎖定為目前的版本，以避免相依檔案的升級。

```
$ npm install
$ npm shrinkwrap
wrote npm-shrinkwrap.json
```

在您的原始碼套件中加入此檔案，以確保相依檔案只會安裝一次。

設定代理伺服器

Elastic Beanstalk 可以使用 NGINX 或 Apache HTTPD 做為反向代理伺服器，在連接埠 80 上將您的應用程式映射到 Elastic Load Balancing 負載平衡器。預設值為 NGINX。Elastic Beanstalk 提供了預設的代理組態，您可以加以擴展，或使用自己的組態將其完全覆寫。

Elastic Beanstalk 預設會設定代理將請求轉送至連接埠 5000 上的應用程式。您可將 `PORT` 環境屬性設定為主要應用程式接聽的連接埠，藉此覆寫預設連接埠。

> **Note**
>
> 您應用程式接聽的連接埠，不會影響 NGINX 伺服器為接收來自負載平衡器的請求所接聽的連接埠。

在您的平台版本上設定代理伺服器

所有 AL2023/AL2 平台皆支援統一的代理組態功能。如需有關在執行 AL2023/AL2 的平台版本上設定代理伺服器的詳細資訊，請展開the section called “擴充 Linux 平台”中的反向代理組態一節。

在 Amazon Linux AMI (之前的 Amazon Linux 2) 上設定代理

如果您的 Elastic Beanstalk Node.js 環境使用 Amazon Linux AMI 平台版本 (先前的 Amazon Linux 2)，請閱讀本節中的資訊。

> **備註**
>
> - 本主題中的資訊僅適用於以 Amazon Linux AMI (AL1) 為基礎的平台分支。AL2023/AL2 平台分支與舊版 Amazon Linux AMI (AL1) 平台版本不相容，需要不同的組態設定。
> - 2022 年 7 月 18 日，Elastic Beanstalk 已將所有以 Amazon Linux AMI (AL1) 為基礎的平台分支狀態設為已淘汰。如需有關遷移至完全支援的目前 Amazon Linux 2023 平台分支的詳細資訊，請參閱 將您的 Elastic Beanstalk Linux 應用程式遷移到 Amazon Linux 2023 或 Amazon Linux 2。

擴展和覆寫預設的代理組態 — Amazon Linux AMI (AL1)

Node.js 平台使用反向代理程式，將執行個體上來自 80 通訊埠的請求，轉傳給接聽 8081 埠的應用程式。Elastic Beanstalk 提供了預設的代理組態，您可以加以擴展，或使用自己的組態將其完全覆寫。

若要擴展預設的組態，請利用組態檔案，將 `.conf` 檔案加入至 `/etc/nginx/conf.d`。如需特定範例，請參閱 在執行 Node.js 的 EC2 執行個體上終止 HTTPS。

Node.js 平台會將 PORT 環境變數設定為代理伺服器傳送流量至其中的連接埠。在您的程式碼中讀取此變數，以設定您應用程式的連接埠。

```
var port = process.env.PORT || 3000;
```

```
    var server = app.listen(port, function () {
        console.log('Server running at http://127.0.0.1:' + port + '/');
    });
```

預設的 NGINX 組態會將流量轉發至位於 127.0.0.1:8081，名為 nodejs 的上游伺服器。您可以移除預設的組態，並在[組態檔案]中提供自己的組態。

Example .ebextensions/proxy.config

下列的範例移除了預設的組態，並新增自訂組態，來將流量轉傳到 5,000 埠而非 8,081 埠。

```
files:
  /etc/nginx/conf.d/proxy.conf:
    mode: "000644"
    owner: root
    group: root
    content: |
      upstream nodejs {
        server 127.0.0.1:5000;
        keepalive 256;
      }

      server {
        listen 8080;

        if ($time_iso8601 ~ "^(\d{4})-(\d{2})-(\d{2})T(\d{2})") {
            set $year $1;
            set $month $2;
            set $day $3;
            set $hour $4;
        }
        access_log /var/log/nginx/healthd/application.log.$year-$month-$day-$hour
 healthd;
        access_log  /var/log/nginx/access.log  main;

        location / {
            proxy_pass  http://nodejs;
            proxy_set_header   Connection "";
            proxy_http_version 1.1;
            proxy_set_header        Host            $host;
            proxy_set_header        X-Real-IP       $remote_addr;
            proxy_set_header        X-Forwarded-For $proxy_add_x_forwarded_for;
        }
```

```
        gzip on;
        gzip_comp_level 4;
        gzip_types text/html text/plain text/css application/json application/x-
javascript text/xml application/xml application/xml+rss text/javascript;

        location /static {
            alias /var/app/current/static;
        }

      }

  /opt/elasticbeanstalk/hooks/configdeploy/post/99_kill_default_nginx.sh:
    mode: "000755"
    owner: root
    group: root
    content: |
      #!/bin/bash -xe
      rm -f /etc/nginx/conf.d/00_elastic_beanstalk_proxy.conf
      service nginx stop
      service nginx start

container_commands:
  removeconfig:
    command: "rm -f /tmp/deployment/config/
#etc#nginx#conf.d#00_elastic_beanstalk_proxy.conf /etc/nginx/
conf.d/00_elastic_beanstalk_proxy.conf"
```

範例組態 (`/etc/nginx/conf.d/proxy.conf`) 使用了位於 `/etc/nginx/conf.d/00_elastic_beanstalk_proxy.conf` 的預設組態做為基礎，以納入預設的伺服器區塊，此區塊具備壓縮與日誌設定，和靜態檔案映射。

`removeconfig` 命令會移除容器的預設組態，因此代理伺服器使用自訂組態。Elastic Beanstalk 會在部署每個組態時，重新建立預設組態。為了說明此點，在下列範例中，新增了 post-configuration-deployment hook (`/opt/elasticbeanstalk/hooks/configdeploy/post/99_kill_default_nginx.sh`)。這會移除預設配置，並重新啟動代理伺服器。

> **Note**
>
> 在 Node.js 平台的未來版本中，預設組態可能會變更。請使用最新版本的組態來做為您自訂項目的基礎，以確保相容性。

如果您覆寫預設組態，則必須定義所有靜態檔案映射和 GZIP 壓縮。這是因為平台無法套用標準設定。

Node.js 的更多示例應用程序和教程

若要開始使用 Node.js 應用程式 AWS Elastic Beanstalk，您只需要一個應用程式來源套件即可上傳為您的第一個應用程式版本並部署到環境。本主QuickStart 對於 Node.js題將逐步引導您完成使用 EB CLI 啟動範例 Node.js 應用程式的過程。本節提供其他應用程式和教學課程。

使用範例 Node.js 應用程式來啟動環境

Elastic Beanstalk 為每個平台提供單一頁面範例應用程式，以及更複雜的範例，顯示其他 AWS 資源的使用情況，例如 Amazon RDS 以及語言或平台特定功能和 API。

> **Note**
>
> 請依照原始碼套件 `README.md` 檔案中的步驟進行部署。

範例

環境類型	原始碼套件	描述
Web 伺服器	nodejs.zip	單頁應用程式。 若要使用 EB CLI 啟動範例應用程式，請參閱QuickStart 對於 Node.js。 您也可以使用 Elastic Beanstalk 控制台來啟動示例應用程序。如需詳細步驟，請參閱本指南的「入門」一章中的「建立範例應用程式」。
具備 Amazon RDS 的 Web 伺服器	nodejs-example-express-rds. 拉鍊	使用 Express 架構和 Amazon Relational Database Service (RDS) 的健行日誌應用程式。 教學課程

環境類型	原始碼套件	描述
網絡服務器與 Amazon ElastiCache	nodejs-example-express-elasticache.拉鍊	使用 Amazon ElastiCache 進行叢集的快速 Web 應用程式。叢集會提高您的 Web 應用程式的可用性、效能和安全性。 教學課程
具備 DynamoDB、Amazon SNS 和 Amazon SQS 的 Web 伺服器	nodejs-example-dynamo.拉鍊	針對新公司行銷活動收集使用者聯絡資訊的 Express 網站。使用 Node.js JavaScript 中的適用 AWS 開發套件將項目寫入 DynamoDB 表格，並使用 Elastic Beanstalk 組態檔案在 DynamoDB、Amazon SNS 和 Amazon SQS 中建立資源。 教學課程

後續步驟

在您擁有執行應用程式的環境後，可以隨時部署應用程式的新版本或完全不同的應用程式。部署新的應用程式版本非常快速，因無須佈建或重新啟動 EC2 執行個體。如需應用程式部署的詳細資訊，請參閱部署應用程式的新版本。

在您部署了一兩個範例應用程式並準備好開始在本機開發和執行 Node.js 應用程式之後，請參閱設定您的 Node.js 開發環境設定 Node.js 開發環境以及您將需要的所有工具。

將 Express 應用程式部署至 Elastic Beanstalk

本章節會逐步說明如何使用 Elastic Beanstalk 命令列介面 (EB CLI)，將範例應用程式部署至 Elastic Beanstalk，並介紹如何更新應用程式，以使用 Express 架構。

必要條件

本教學課程需要下列先決條件：

- Node.js 執行階段
- 預設的 Node.js 套件管理工具軟體 npm

- Express 命令列產生器
- Elastic Beanstalk 命令列界面 (EB CLI)

有關安裝所列出之前三個元件和設定本機開發環境的詳細資訊，請參閱 設定您的 Node.js 開發環境。在本教學課程中，您不需要安裝 Node.js 的 AWS SDK，這也在參考的主題中提到。

有關安裝和設定 EB CLI 的詳細資訊，請參閱 安裝 EB CLI 和 設定 EB CLI。

建立 Elastic Beanstalk 環境

應用程式目錄

對於應用程式原始碼套件，本教學課程使用的是名為 `nodejs-example-express-rds` 的目錄。為本教學課程建立 `nodejs-example-express-rds` 目錄。

```
~$ mkdir nodejs-example-express-rds
```

> **Note**
>
> 本章中的每個教學課程皆會使用其自身的應用程式原始碼套件目錄。目錄名稱與教學課程所使用的範例應用程式名稱相符。

將您目前的工作目錄變更為 `nodejs-example-express-rds`。

```
~$ cd nodejs-example-express-rds
```

現在，來設定執行 Node.js 平台和範例應用程式的 Elastic Beanstalk 環境。我們將會使用 Elastic Beanstalk 命令列介面 (EB CLI)。

設定應用程式的 EB CLI 儲存庫，並建立執行 Node.js 平台的 Elastic Beanstalk 環境

1. 使用 eb init 命令建立一個儲存庫。

   ```
   ~/nodejs-example-express-rds$ eb init --platform node.js --region <region>
   ```

   此命令會在名為 `.elasticbeanstalk` 的資料夾內建立組態檔案，其中會指定應用程式使用的環境設定，並以目前資料夾為名建立 Elastic Beanstalk 應用程式。
2. 使用 eb create 命令建立執行範例應用程式的環境。

```
~/nodejs-example-express-rds$ eb create --sample nodejs-example-express-rds
```

本命令會使用 Node.js 平台的預設設定和下列資源，建立負載平衡的環境：

- EC2 執行個體 – Amazon Elastic Compute Cloud (Amazon EC2) 虛擬機器，已設為在您選擇的平台上執行 Web 應用程式。

  每個平台會執行特定的一套軟體、設定檔和指令碼，來支援特定的語言版本、架構、Web 容器或其組合。大多數的平台使用會 Apache 或 NGINX 做為反向代理，此反向代理會在您 Web 應用程式的前景執行、轉傳遞交給此 Web 應用程式的請求、提供靜態資產，並產生存取和錯誤日誌。
- 執行個體安全群組 - Amazon EC2 安全群組，已設為允許從連接埠 80 傳入的流量。此資源可讓負載平衡器傳來的 HTTP 傳輸資料，到達執行您 Web 應用程式的 EC2 執行個體。在預設情況下，不允許傳輸資料從其他通訊埠傳送。
- 負載平衡器 - Elastic Load Balancing 負載平衡器，可設定將請求分配到執行您應用程式的執行個體。負載平衡器也讓您的執行個體不需直接連接到網際網路。
- 負載平衡器安全群組 - Amazon EC2 安全群組，設為允許從連接埠 80 傳入的流量。此資源可讓來自網際網路的 HTTP 傳輸資料到達負載平衡器。在預設情況下，不允許傳輸資料從其他通訊埠傳送。
- Auto Scaling 群組 - Auto Scaling 群組，設為在執行個體終止或無法使用時，取代該執行個體。
- Amazon S3 儲存貯體 - 儲存位置，用來儲存當您使用 Elastic Beanstalk 時所建立的原始程式碼、日誌和其他成品。
- Amazon CloudWatch 警示 — 兩個 CloudWatch 警示，用於監控環境中執行個體的負載，並在負載過高或過低時觸發警示。當警示觸發時，您的 Auto Scaling 群組會擴展或縮減以進行回應。
- AWS CloudFormation 堆疊 — Elastic Beanstalk 用 AWS CloudFormation 來啟動環境中的資源並傳播組態變更。資源定義於範本中，您可在 AWS CloudFormation 主控台中檢視此範本。
- 網域名稱 – 會路由到您 Web 應用程式的網域名稱，其格式為 *subdomain*.*region*.elasticbeanstalk.com。

> **Note**
>
> 為了增強 Elastic Beanstalk 應用程式的安全性，我們會在公共后缀列表 (PSL) 中註冊網域 elasticbeanstalk.com。為了加強安全性，如果您需要在 Elastic Beanstalk 應用程式的預設網域名稱中設定敏感性 Cookie，我們建議您使用具 __Host- 前置詞的

Cookie。此做法將有助於保護您的網域免受跨站請求偽造 (CSRF) 攻擊。如需更多資訊，請參閱 Mozilla 開發人員網路中的設定 Cookie 頁面。

3. 環境建立完成後，請使用 eb open 命令，在預設瀏覽器中開啟環境 URL。

```
~/nodejs-example-express-rds$ eb open
```

您現在已使用範例應用程式建立 Node.js Elastic Beanstalk 環境。您可以使用自己的應用程式對其進行更新。接下來，我們會更新範例應用程式，以使用 Express 架構。

更新應用程式以使用 Express

使用範例應用程式建立環境後，您可使用自己的應用程式對其進行更新。在此程序中，我們先執行 express 和 npm install 命令，以便在應用程式目錄中設定 Express 架構。然後，我們將使用 EB CLI，以便使用已更新的應用程式來更新您的 Elastic Beanstalk 環境。

欲更新您的應用程式以使用 Express

1. 執行 `express` 命令。這會產生 `package.json`、`app.js` 以及幾個目錄。

```
~/nodejs-example-express-rds$ express
```

當提示您是否要繼續時，請輸入 **y**。

> **Note**
>
> 如果 express 命令無法使用，您可能未依先前先決條件章節中的所述內容安裝 Express 命令列產生器。或者，您可能需要設定本機電腦的目錄路徑設定，才可執行 express 命令。如需有關設定開發環境的詳細步驟，請參閱先決條件章節，以繼續進行本教學課程。

2. 設定本機依存項目。

```
~/nodejs-example-express-rds$ npm install
```

3. (選用) 確認 Web 應用程式伺服器已啟動。

```
~/nodejs-example-express-rds$ npm start
```

您應該會看到類似下列的輸出：

```
> nodejs@0.0.0 start /home/local/user/node-express
> node ./bin/www
```

依預設，伺服器將會在連接埠 3000 上執行。若要進行測試，請在另一部終端機上執行 `curl http://localhost:3000`，或在本機電腦開啟瀏覽器並輸入 URL 位址 `http://localhost:3000`。

按 Ctrl+C 來停止伺服器。

4. 使用 eb deploy 命令將變更部署至您的 Elastic Beanstalk 環境。

```
~/nodejs-example-express-rds$ eb deploy
```

5. 一旦環境為綠色且就緒，請重新整理 URL，確認其正常運作。您應看到顯示 Welcome to Express 的網頁。

接著，我們會更新 Express 應用程式以提供靜態檔案，並新增新的頁面。

欲設定靜態檔案並新增新的頁面至您的 Express 應用程式

1. 使用下列內容，在 `.ebextensions` 資料夾中新增第二個組態檔案：

nodejs-example-express-rds/.ebextensions/staticfiles.config

```
option_settings:
    aws:elasticbeanstalk:environment:proxy:staticfiles:
        /stylesheets: public/stylesheets
```

此設定會將代理伺服器設定為在應用程式 `public` 路徑的 `/public` 資料夾內提供檔案。從代理伺服器靜態提供檔案，能夠減少應用程式負載。如需詳細資訊，請參閱本章先前所述的靜態檔案。

2. (選用) 若要確認靜態映射的設定是否正確，請在 `nodejs-example-express-rds/app.js` 中註解靜態映射設定。這會從節點應用程式中移除映射。

```
//  app.use(express.static(path.join(__dirname, 'public')));
```

即使在註解此行後，上一個步驟 staticfiles.config 檔案中的靜態檔案映射應仍可成功載入樣式表。若要確認靜態檔案映射是透過代理靜態檔案組態 (而非 Express 應用程式) 載入，請移除 option_settings: 後的值。將其從靜態檔案組態和節點應用程式移除之後，樣式表將無法載入。

完成測試時，請記得重設 nodejs-example-express-rds/app.js 及 staticfiles.config 的內容。

3. 新增 nodejs-example-express-rds/routes/hike.js。輸入下列內容：

```
exports.index = function(req, res) {
 res.render('hike', {title: 'My Hiking Log'});
};

exports.add_hike = function(req, res) {
};
```

4. 更新 nodejs-example-express-rds/app.js 以納入三個新的行。

首先，新增下列行來為此路由添加 require：

```
var hike = require('./routes/hike');
```

您的檔案看起來如下列程式碼片段：

```
var express = require('express');
var path = require('path');
var hike = require('./routes/hike');
```

然後，請在 nodejs-example-express-rds/app.js 後將下列兩行新增至 var app = express();：

```
app.get('/hikes', hike.index);
app.post('/add_hike', hike.add_hike);
```

您的檔案看起來如下列程式碼片段：

```
var app = express();
app.get('/hikes', hike.index);
```

```
app.post('/add_hike', hike.add_hike);
```

5. 將 nodejs-example-express-rds/views/index.jade 複製至 nodejs-example-express-rds/views/hike.jade。

```
~/nodejs-example-express-rds$ cp views/index.jade views/hike.jade
```

6. 使用 eb deploy 命令部署變更。

```
~/nodejs-example-express-rds$ eb deploy
```

7. 您的環境將在幾分鐘後更新。您的環境為綠色且就緒後，請重新整理您的瀏覽器，並將 **hikes** 附加至 URL 尾端 (即 http://node-express-env-syypntcz2q.elasticbeanstalk.com/hikes)，藉此確認其正常運作。

   您應看到標題為 My Hiking Log (My Hiking Log) 的網頁。

現在，您已建立使用 Express 架構的 Web 應用程式。在下一節中，我們將修改應用程式，以使用 Amazon Relational Database Service (RDS) 儲存健行日誌。

更新應用程式，以使用 Amazon RDS

在下一個步驟中，我們將更新應用程式，以使用 Amazon RDS for MySQL。

更新應用程式，以使用 RDS for MySQL

1. 若要建立與 Elastic Beanstalk 環境耦合的 RDS for MySQL 資料庫，請遵循本章後續新增資料庫主題中的指示。新增資料庫執行個體約需要 10 分鐘。
2. 使用下列內容更新 package.json 中的相依性區段：

```
"dependencies": {
    "async": "^3.2.4",
    "express": "4.18.2",
    "jade": "1.11.0",
    "mysql": "2.18.1",
    "node-uuid": "^1.4.8",
    "body-parser": "^1.20.1",
    "method-override": "^3.0.0",
    "morgan": "^1.10.0",
    "errorhandler": "^1.5.1"
  }
```

3. 執行 npm install。

```
~/nodejs-example-express-rds$ npm install
```

4. 更新 app.js，以便連接至資料庫、建立資料表，並插入單一預設增長日誌。每次部署此應用程式時，其會捨棄先前的增長資料表，並重新建立。

```
/**
 * Module dependencies.
 */

 const express = require('express')
 , routes = require('./routes')
 , hike = require('./routes/hike')
 , http = require('http')
 , path = require('path')
 , mysql = require('mysql')
 , async = require('async')
 , bodyParser = require('body-parser')
 , methodOverride = require('method-override')
 , morgan = require('morgan')
 , errorhandler = require('errorhandler');

const { connect } = require('http2');

const app = express()

app.set('views', __dirname + '/views')
app.set('view engine', 'jade')
app.use(methodOverride())
app.use(bodyParser.json())
app.use(bodyParser.urlencoded({ extended: true }))
app.use(express.static(path.join(__dirname, 'public')))


app.set('connection', mysql.createConnection({
host: process.env.RDS_HOSTNAME,
user: process.env.RDS_USERNAME,
password: process.env.RDS_PASSWORD,
port: process.env.RDS_PORT}));

function init() {
 app.get('/', routes.index);
```

```
 app.get('/hikes', hike.index);
 app.post('/add_hike', hike.add_hike);
}

const client = app.get('connection');
async.series([
 function connect(callback) {
   client.connect(callback);
   console.log('Connected!');
 },
 function clear(callback) {
   client.query('DROP DATABASE IF EXISTS mynode_db', callback);
 },
 function create_db(callback) {
   client.query('CREATE DATABASE mynode_db', callback);
 },
 function use_db(callback) {
   client.query('USE mynode_db', callback);
 },
 function create_table(callback) {
    client.query('CREATE TABLE HIKES (' +
                        'ID VARCHAR(40), ' +
                        'HIKE_DATE DATE, ' +
                        'NAME VARCHAR(40), ' +
                        'DISTANCE VARCHAR(40), ' +
                        'LOCATION VARCHAR(40), ' +
                        'WEATHER VARCHAR(40), ' +
                        'PRIMARY KEY(ID))', callback);
 },
 function insert_default(callback) {
   const hike = {HIKE_DATE: new Date(), NAME: 'Hazard Stevens',
         LOCATION: 'Mt Rainier', DISTANCE: '4,027m vertical', WEATHER:'Bad', ID:
 '12345'};
   client.query('INSERT INTO HIKES set ?', hike, callback);
 }
], function (err, results) {
 if (err) {
   console.log('Exception initializing database.');
   throw err;
 } else {
   console.log('Database initialization complete.');
   init();
 }
});
```

```
module.exports = app
```

5. 將下列內容新增至 routes/hike.js。這將使路線能夠將新的增長日誌插入增長資料庫。

```
const uuid = require('node-uuid');
exports.index = function(req, res) {
  res.app.get('connection').query( 'SELECT * FROM HIKES', function(err,
rows) {
    if (err) {
      res.send(err);
    } else {
      console.log(JSON.stringify(rows));
      res.render('hike', {title: 'My Hiking Log', hikes: rows});
  }});
};
exports.add_hike = function(req, res){
  const input = req.body.hike;
  const hike = { HIKE_DATE: new Date(), ID: uuid.v4(), NAME: input.NAME,
  LOCATION: input.LOCATION, DISTANCE: input.DISTANCE, WEATHER: input.WEATHER};
  console.log('Request to log hike:' + JSON.stringify(hike));
  req.app.get('connection').query('INSERT INTO HIKES set ?', hike, function(err) {
      if (err) {
        res.send(err);
      } else {
        res.redirect('/hikes');
      }
   });
};
```

6. 將 routes/index.js 的內容取代為：

```
/*
 * GET home page.
 */

exports.index = function(req, res){
  res.render('index', { title: 'Express' });
};
```

7. 將以下 Jade 範本新增至 views/hike.jade，以提供用於新增增長日誌的使用者介面。

```
extends layout
```

```
block content
  h1= title
  p Welcome to #{title}

  form(action="/add_hike", method="post")
    table(border="1")
      tr
        td Your Name
        td
          input(name="hike[NAME]", type="textbox")
      tr
        td Location
        td
          input(name="hike[LOCATION]", type="textbox")
      tr
        td Distance
        td
          input(name="hike[DISTANCE]", type="textbox")
      tr
        td Weather
        td
          input(name="hike[WEATHER]", type="radio", value="Good")
          | Good
          input(name="hike[WEATHER]", type="radio", value="Bad")
          | Bad
          input(name="hike[WEATHER]", type="radio", value="Seattle", checked)
          | Seattle
      tr
        td(colspan="2")
          input(type="submit", value="Record Hike")

  div
    h3 Hikes
    table(border="1")
      tr
        td Date
        td Name
        td Location
        td Distance
        td Weather
      each hike in hikes
        tr
          td #{hike.HIKE_DATE.toDateString()}
```

```
            td #{hike.NAME}
            td #{hike.LOCATION}
            td #{hike.DISTANCE}
            td #{hike.WEATHER}
```

8. 使用 eb deploy 命令部署變更。

```
~/nodejs-example-express-rds$ eb deploy
```

清除

如果您已完成使用 Elastic Beanstalk，您可終止環境。

使用 eb terminate 命令來終止您的環境及其中的所有資源。

```
~/nodejs-example-express-rds$ eb terminate
The environment "nodejs-example-express-rds-env" and all associated instances will be
 terminated.
To confirm, type the environment name: nodejs-example-express-rds-env
INFO: terminateEnvironment is starting.
...
```

將具備叢集功能的 Express 應用程式部署至 Elastic Beanstalk

本教學將逐步引導您使用 Elastic Beanstalk 命令列界面 (EB CLI) 將範例應用程式部署到 Elastic Beanstalk，然後更新應用程式以使用快速架構、Amazon 和叢集。 ElastiCache叢集會提高您的 Web 應用程式的可用性、效能和安全性。要了解有關 Amazon 的更多信息 ElastiCache，請訪問什麼是 ElastiCache 適用於 Memcached 的 Amazon？ 在 Amazon 的 Memcached ElastiCache 用戶指南。

> **Note**
>
> 此範例會建立可能會向您收費的 AWS 資源。如需 AWS 定價的詳細資訊，請參閱https://aws.amazon.com/pricing/。部分服務屬於 AWS 免費用量方案的一部分。若您是新客戶，可以免費試用這些服務。如需更多資訊，請參閱https://aws.amazon.com/free/。

必要條件

本教學課程需要下列先決條件：

- Node.js 執行階段
- 預設的 Node.js 套件管理工具軟體 npm
- Express 命令列產生器
- Elastic Beanstalk 命令列界面 (EB CLI)

有關安裝所列出之前三個元件和設定本機開發環境的詳細資訊，請參閱 設定您的 Node.js 開發環境。在本教學課程中，您不需要安裝 Node.js 的 AWS SDK，這也在參考的主題中提到。

有關安裝和設定 EB CLI 的詳細資訊，請參閱 安裝 EB CLI 和 設定 EB CLI。

建立 Elastic Beanstalk 環境

應用程式目錄

對於應用程式原始碼套件，本教學課程使用的是名為 `nodejs-example-express-elasticache` 的目錄。為本教學課程建立 `nodejs-example-express-elasticache` 目錄。

```
~$ mkdir nodejs-example-express-elasticache
```

> **Note**
>
> 本章中的每個教學課程皆會使用其自身的應用程式原始碼套件目錄。目錄名稱與教學課程所使用的範例應用程式名稱相符。

將您目前的工作目錄變更為 `nodejs-example-express-elasticache`。

```
~$ cd nodejs-example-express-elasticache
```

現在，來設定執行 Node.js 平台和範例應用程式的 Elastic Beanstalk 環境。我們將會使用 Elastic Beanstalk 命令列介面 (EB CLI)。

設定應用程式的 EB CLI 儲存庫，並建立執行 Node.js 平台的 Elastic Beanstalk 環境

1. 使用 eb init 命令建立一個儲存庫。

   ```
   ~/nodejs-example-express-elasticache$ eb init --platform node.js --region <region>
   ```

此命令會在名為 `.elasticbeanstalk` 的資料夾內建立組態檔案，其中會指定應用程式使用的環境設定，並以目前資料夾為名建立 Elastic Beanstalk 應用程式。

2. 使用 eb create 命令建立執行範例應用程式的環境。

```
~/nodejs-example-express-elasticache$ eb create --sample nodejs-example-express-elasticache
```

本命令會使用 Node.js 平台的預設設定和下列資源，建立負載平衡的環境：

- EC2 執行個體 – Amazon Elastic Compute Cloud (Amazon EC2) 虛擬機器，已設為在您選擇的平台上執行 Web 應用程式。

  每個平台會執行特定的一套軟體、設定檔和指令碼，來支援特定的語言版本、架構、Web 容器或其組合。大多數的平台使用會 Apache 或 NGINX 做為反向代理，此反向代理會在您 Web 應用程式的前景執行、轉傳遞交給此 Web 應用程式的請求、提供靜態資產，並產生存取和錯誤日誌。
- 執行個體安全群組 - Amazon EC2 安全群組，已設為允許從連接埠 80 傳入的流量。此資源可讓負載平衡器傳來的 HTTP 傳輸資料，到達執行您 Web 應用程式的 EC2 執行個體。在預設情況下，不允許傳輸資料從其他通訊埠傳送。
- 負載平衡器 - Elastic Load Balancing 負載平衡器，可設定將請求分配到執行您應用程式的執行個體。負載平衡器也讓您的執行個體不需直接連接到網際網路。
- 負載平衡器安全群組 - Amazon EC2 安全群組，設為允許從連接埠 80 傳入的流量。此資源可讓來自網際網路的 HTTP 傳輸資料到達負載平衡器。在預設情況下，不允許傳輸資料從其他通訊埠傳送。
- Auto Scaling 群組 - Auto Scaling 群組，設為在執行個體終止或無法使用時，取代該執行個體。
- Amazon S3 儲存貯體 - 儲存位置，用來儲存當您使用 Elastic Beanstalk 時所建立的原始程式碼、日誌和其他成品。
- Amazon CloudWatch 警示 — 監控環境中執行個體負載的兩個 CloudWatch 警示，並在負載過高或過低時觸發警示。當警示觸發時，您的 Auto Scaling 群組會擴展或縮減以進行回應。
- AWS CloudFormation 堆疊 — Elastic Beanstalk 用 AWS CloudFormation 來啟動環境中的資源並傳播組態變更。資源定義於範本中，您可在 AWS CloudFormation 主控台中檢視此範本。
- 網域名稱 – 會路由到您 Web 應用程式的網域名稱，其格式為 *subdomain*.*region*.elasticbeanstalk.com。

> **Note**
>
> 為了增強 Elastic Beanstalk 應用程式的安全性，我們會在公共后缀列表 (PSL) 中註冊網域 elasticbeanstalk.com。為了加強安全性，如果您需要在 Elastic Beanstalk 應用程式的預設網域名稱中設定敏感性 Cookie，我們建議您使用具 `__Host-` 前置詞的 Cookie。此做法將有助於保護您的網域免受跨站請求偽造 (CSRF) 攻擊。如需更多資訊，請參閱 Mozilla 開發人員網路中的設定 Cookie 頁面。

3. 環境建立完成後，請使用 eb open 命令，在預設瀏覽器中開啟環境 URL。

```
~/nodejs-example-express-elasticache$ eb open
```

您現在已使用範例應用程式建立 Node.js Elastic Beanstalk 環境。您可以使用自己的應用程式對其進行更新。接下來，我們會更新範例應用程式，以使用 Express 架構。

更新應用程式以使用 Express

更新 Elastic Beanstalk 環境中的範例應用程式，以使用 Express 框架。

您可以從 nodejs-example-express-elasticache.zip 下載最終的源代碼。

欲更新您的應用程式以使用 Express

使用範例應用程式建立環境後，您可使用自己的應用程式對其進行更新。在此程序中，我們先執行 express 和 npm install 命令，以便在應用程式目錄中設定 Express 架構。

1. 執行 `express` 命令。這會產生 `package.json`、`app.js` 以及幾個目錄。

```
~/nodejs-example-express-elasticache$ express
```

   當提示您是否要繼續時，請輸入 **y**。

> **Note**
>
> 如果 express 命令無法使用，您可能未依先前先決條件章節中的所述內容安裝 Express 命令列產生器。或者，您可能需要設定本機電腦的目錄路徑設定，才可執行 express 命令。如需有關設定開發環境的詳細步驟，請參閱先決條件章節，以繼續進行本教學課程。

2. 設定本機依存項目。

```
~/nodejs-example-express-elasticache$ npm install
```

3. (選用) 確認 Web 應用程式伺服器已啟動。

```
~/nodejs-example-express-elasticache$ npm start
```

您應該會看到類似下列的輸出：

```
> nodejs@0.0.0 start /home/local/user/node-express
> node ./bin/www
```

依預設，伺服器將會在連接埠 3000 上執行。若要進行測試，請在另一部終端機上執行 `curl http://localhost:3000`，或在本機電腦開啟瀏覽器並輸入 URL 位址 `http://localhost:3000`。

按 Ctrl+C 來停止伺服器。

4. 重新命名 `nodejs-example-express-elasticache/app.js` 為 `nodejs-example-express-elasticache/express-app.js`。

```
~/nodejs-example-express-elasticache$ mv app.js express-app.js
```

5. 將 `nodejs-example-express-elasticache/express-app.js` 中的 `var app = express();` 列更新為以下內容：

```
var app = module.exports = express();
```

6. 在您的本機電腦上，以下列程式碼建立名為 `nodejs-example-express-elasticache/app.js` 的檔案。

```
/**
 * Module dependencies.
 */

 const express = require('express'),
 session = require('express-session'),
 bodyParser = require('body-parser'),
 methodOverride = require('method-override'),
 cookieParser = require('cookie-parser'),
```

```
 fs = require('fs'),
 filename = '/var/nodelist',
 app = express();

let MemcachedStore = require('connect-memcached')(session);

function setup(cacheNodes) {
 app.use(bodyParser.raw());
 app.use(methodOverride());
 if (cacheNodes.length > 0) {
   app.use(cookieParser());

   console.log('Using memcached store nodes:');
   console.log(cacheNodes);

   app.use(session({
     secret: 'your secret here',
     resave: false,
     saveUninitialized: false,
     store: new MemcachedStore({ 'hosts': cacheNodes })
   }));
 } else {
   console.log('Not using memcached store.');
   app.use(session({
     resave: false,
     saveUninitialized: false, secret: 'your secret here'
   }));
 }

 app.get('/', function (req, resp) {
   if (req.session.views) {
     req.session.views++
     resp.setHeader('Content-Type', 'text/html')
     resp.send(`You are session: ${req.session.id}. Views: ${req.session.views}`)
   } else {
     req.session.views = 1
     resp.send(`You are session: ${req.session.id}. No views yet, refresh the page!
`)
   }
 });

 if (!module.parent) {
   console.log('Running express without cluster. Listening on port %d',
 process.env.PORT || 5000)
```

```
    app.listen(process.env.PORT || 5000)
 }
}

console.log("Reading elastic cache configuration")
// Load elasticache configuration.
fs.readFile(filename, 'UTF8', function (err, data) {
 if (err) throw err;

 let cacheNodes = []
 if (data) {
   let lines = data.split('\n');
   for (let i = 0; i < lines.length; i++) {
     if (lines[i].length > 0) {
       cacheNodes.push(lines[i])
     }
   }
 }

 setup(cacheNodes)
});

module.exports = app;
```

7. 將 nodejs-example-express-elasticache/bin/www 檔案的內容取代為下列內容：

```
#!/usr/bin/env node

/**
 * Module dependencies.
 */

const app = require('../app');
const cluster = require('cluster');
const debug = require('debug')('nodejs-example-express-elasticache:server');
const http = require('http');
const workers = {},
  count = require('os').cpus().length;

function spawn() {
  const worker = cluster.fork();
  workers[worker.pid] = worker;
  return worker;
}
```

```
/**
 * Get port from environment and store in Express.
 */

const port = normalizePort(process.env.PORT || '3000');
app.set('port', port);

if (cluster.isMaster) {
  for (let i = 0; i < count; i++) {
    spawn();
  }

  // If a worker dies, log it to the console and start another worker.
  cluster.on('exit', function (worker, code, signal) {
    console.log('Worker ' + worker.process.pid + ' died.');
    cluster.fork();
  });

  // Log when a worker starts listening
  cluster.on('listening', function (worker, address) {
    console.log('Worker started with PID ' + worker.process.pid + '.');
  });

} else {
  /**
   * Create HTTP server.
   */

  let server = http.createServer(app);

  /**
   * Event listener for HTTP server "error" event.
   */

  function onError(error) {
    if (error.syscall !== 'listen') {
      throw error;
    }

    const bind = typeof port === 'string'
      ? 'Pipe ' + port
      : 'Port ' + port;
```

```
    // handle specific listen errors with friendly messages
    switch (error.code) {
      case 'EACCES':
        console.error(bind + ' requires elevated privileges');
        process.exit(1);
        break;
      case 'EADDRINUSE':
        console.error(bind + ' is already in use');
        process.exit(1);
        break;
      default:
        throw error;
    }
  }

  /**
   * Event listener for HTTP server "listening" event.
   */

  function onListening() {
    const addr = server.address();
    const bind = typeof addr === 'string'
      ? 'pipe ' + addr
      : 'port ' + addr.port;
    debug('Listening on ' + bind);
  }

  /**
   * Listen on provided port, on all network interfaces.
   */

  server.listen(port);
  server.on('error', onError);
  server.on('listening', onListening);
}

/**
 * Normalize a port into a number, string, or false.
 */

function normalizePort(val) {
  const port = parseInt(val, 10);
```

```
  if (isNaN(port)) {
    // named pipe
    return val;
  }

  if (port >= 0) {
    // port number
    return port;
  }

  return false;
}
```

8. 使用 eb deploy 命令將變更部署至您的 Elastic Beanstalk 環境。

```
~/nodejs-example-express-elasticache$ eb deploy
```

9. 您的環境將在幾分鐘後更新。一旦環境為綠色且就緒，請重新整理 URL，確認其正常運作。您應看到顯示「Welcome to Express」的網頁。

您可以存取執行應用程式之 EC2 執行個體的日誌。如需存取日誌的說明，請參閱 在 Elastic Beanstalk 環境中檢視 Amazon EC2 執行個體的日誌。

接下來，讓我們更新快遞應用程序以使用 Amazon ElastiCache。

更新您的快遞應用程序以使用 Amazon ElastiCache

1. 在您本機電腦原始碼套件的最上層目錄建立 `.ebextensions` 目錄。在此範例中，我們使用 `nodejs-example-express-elasticache/.ebextensions`.
2. 使用以下程式碼片段建立組態檔案 `nodejs-example-express-elasticache/.ebextensions/elasticache-iam-with-script.config`。如需組態檔案的詳細資訊，請參閱 Node.js 組態命名空間。這會建立具有所需許可的 IAM 使用者，以探索 elasticache 節點並在快取變更時隨時進行寫入至檔案。您也可以從 nodejs-example-express-elasticache.zip 複製檔案。如需 ElastiCache性質的詳細資訊，請參閱〈〉範例：ElastiCache。

> **Note**
>
> YAML 憑藉一致的縮排。請在取代範例組態檔中的內容時，讓縮排層級一致，並確認您的文字編輯器使用空格而非定位字元進行縮排。

```
Resources:
  MyCacheSecurityGroup:
    Type: 'AWS::EC2::SecurityGroup'
    Properties:
      GroupDescription: "Lock cache down to webserver access only"
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort:
            Fn::GetOptionSetting:
              OptionName: CachePort
              DefaultValue: 11211
          ToPort:
            Fn::GetOptionSetting:
              OptionName: CachePort
              DefaultValue: 11211
          SourceSecurityGroupName:
            Ref: AWSEBSecurityGroup
  MyElastiCache:
    Type: 'AWS::ElastiCache::CacheCluster'
    Properties:
      CacheNodeType:
        Fn::GetOptionSetting:
          OptionName: CacheNodeType
          DefaultValue: cache.t2.micro
      NumCacheNodes:
        Fn::GetOptionSetting:
          OptionName: NumCacheNodes
          DefaultValue: 1
      Engine:
        Fn::GetOptionSetting:
          OptionName: Engine
          DefaultValue: redis
      VpcSecurityGroupIds:
        -
          Fn::GetAtt:
            - MyCacheSecurityGroup
            - GroupId
  AWSEBAutoScalingGroup :
    Metadata :
      ElastiCacheConfig :
        CacheName :
          Ref : MyElastiCache
```

```
        CacheSize :
           Fn::GetOptionSetting:
             OptionName : NumCacheNodes
             DefaultValue: 1
  WebServerUser :
    Type : AWS::IAM::User
    Properties :
      Path : "/"
      Policies:
        -
          PolicyName: root
          PolicyDocument :
            Statement :
              -
                Effect : Allow
                Action :
                  - cloudformation:DescribeStackResource
                  - cloudformation:ListStackResources
                  - elasticache:DescribeCacheClusters
                Resource : "*"
  WebServerKeys :
    Type : AWS::IAM::AccessKey
    Properties :
      UserName :
        Ref: WebServerUser

Outputs:
  WebsiteURL:
    Description: sample output only here to show inline string function parsing
    Value: |
      http://`{ "Fn::GetAtt" : [ "AWSEBLoadBalancer", "DNSName" ] }`
  MyElastiCacheName:
    Description: Name of the elasticache
    Value:
      Ref : MyElastiCache
  NumCacheNodes:
    Description: Number of cache nodes in MyElastiCache
    Value:
      Fn::GetOptionSetting:
        OptionName : NumCacheNodes
        DefaultValue: 1

files:
  "/etc/cfn/cfn-credentials" :
```

```
    content : |
      AWSAccessKeyId=`{ "Ref" : "WebServerKeys" }`
      AWSSecretKey=`{ "Fn::GetAtt" : ["WebServerKeys", "SecretAccessKey"] }`
    mode : "000400"
    owner : root
    group : root

  "/etc/cfn/get-cache-nodes" :
    content : |
      # Define environment variables for command line tools
      export AWS_ELASTICACHE_HOME="/home/ec2-user/elasticache/$(ls /home/ec2-user/
elasticache/)"
      export AWS_CLOUDFORMATION_HOME=/opt/aws/apitools/cfn
      export PATH=$AWS_CLOUDFORMATION_HOME/bin:$AWS_ELASTICACHE_HOME/bin:$PATH
      export AWS_CREDENTIAL_FILE=/etc/cfn/cfn-credentials
      export JAVA_HOME=/usr/lib/jvm/jre

      # Grab the Cache node names and configure the PHP page
      aws cloudformation list-stack-resources --stack `{ "Ref" :
 "AWS::StackName" }` --region `{ "Ref" : "AWS::Region" }` --output text | grep
 MyElastiCache | awk '{print $4}' | xargs -I {} aws elasticache describe-cache-
clusters --cache-cluster-id {} --region `{ "Ref" : "AWS::Region" }` --show-
cache-node-info --output text | grep '^ENDPOINT' | awk '{print $2 ":" $3}' >
 `{ "Fn::GetOptionSetting" : { "OptionName" : "NodeListPath", "DefaultValue" : "/
var/www/html/nodelist" } }`
    mode : "000500"
    owner : root
    group : root

  "/etc/cfn/hooks.d/cfn-cache-change.conf" :
    "content": |
      [cfn-cache-size-change]
      triggers=post.update
      path=Resources.AWSEBAutoScalingGroup.Metadata.ElastiCacheConfig
      action=/etc/cfn/get-cache-nodes
      runas=root

sources :
  "/home/ec2-user/elasticache" : "https://elasticache-downloads.s3.amazonaws.com/
AmazonElastiCacheCli-latest.zip"

commands:
  make-elasticache-executable:
    command: chmod -R ugo+x /home/ec2-user/elasticache/*/bin/*
```

```
packages :
  "yum" :
    "aws-apitools-cfn"  : []

container_commands:
  initial_cache_nodes:
    command: /etc/cfn/get-cache-nodes
```

3. 在您的本機電腦上，建立nodejs-example-express-elasticache/.ebextensions/elasticache_settings.config包含下列程式碼片段的組態檔案以進行設定 ElastiCache。

```
option_settings:
  "aws:elasticbeanstalk:customoption":
     CacheNodeType: cache.t2.micro
     NumCacheNodes: 1
     Engine: memcached
     NodeListPath: /var/nodelist
```

4. 在您的本機電腦上使用以下程式碼片段取代 nodejs-example-express-elasticache/express-app.js 。這個檔案會從磁碟 (/var/nodelist) 讀取節點清單，並設定 express 以使用 memcached 做為工作階段存放區 (若節點存在)。您的檔案看起來應該如下所示。

```
/**
 * Module dependencies.
 */

var express = require('express'),
    session = require('express-session'),
    bodyParser = require('body-parser'),
    methodOverride = require('method-override'),
    cookieParser = require('cookie-parser'),
    fs = require('fs'),
    filename = '/var/nodelist',
    app = module.exports = express();

var MemcachedStore = require('connect-memcached')(session);

function setup(cacheNodes) {
  app.use(bodyParser.raw());
  app.use(methodOverride());
  if (cacheNodes) {
      app.use(cookieParser());
```

```
      console.log('Using memcached store nodes:');
      console.log(cacheNodes);

      app.use(session({
          secret: 'your secret here',
          resave: false,
          saveUninitialized: false,
          store: new MemcachedStore({'hosts': cacheNodes})
      }));
  } else {
    console.log('Not using memcached store.');
    app.use(cookieParser('your secret here'));
    app.use(session());
  }

  app.get('/', function(req, resp){
  if (req.session.views) {
      req.session.views++
      resp.setHeader('Content-Type', 'text/html')
      resp.write('Views: ' + req.session.views)
      resp.end()
   } else {
      req.session.views = 1
      resp.end('Refresh the page!')
    }
  });

  if (!module.parent) {
      console.log('Running express without cluster.');
      app.listen(process.env.PORT || 5000);
  }
}

// Load elasticache configuration.
fs.readFile(filename, 'UTF8', function(err, data) {
    if (err) throw err;

    var cacheNodes = [];
    if (data) {
        var lines = data.split('\n');
        for (var i = 0 ; i < lines.length ; i++) {
            if (lines[i].length > 0) {
                cacheNodes.push(lines[i]);
```

```
            }
        }
    }
    setup(cacheNodes);
});
```

5. 在本機電腦上，使用以下內容更新 package.json：

```
  "dependencies": {
    "cookie-parser": "~1.4.4",
    "debug": "~2.6.9",
    "express": "~4.16.1",
    "http-errors": "~1.6.3",
    "jade": "~1.11.0",
    "morgan": "~1.9.1",
    "connect-memcached": "*",
    "express-session": "*",
    "body-parser": "*",
    "method-override": "*"
  }
```

6. 執行 npm install。

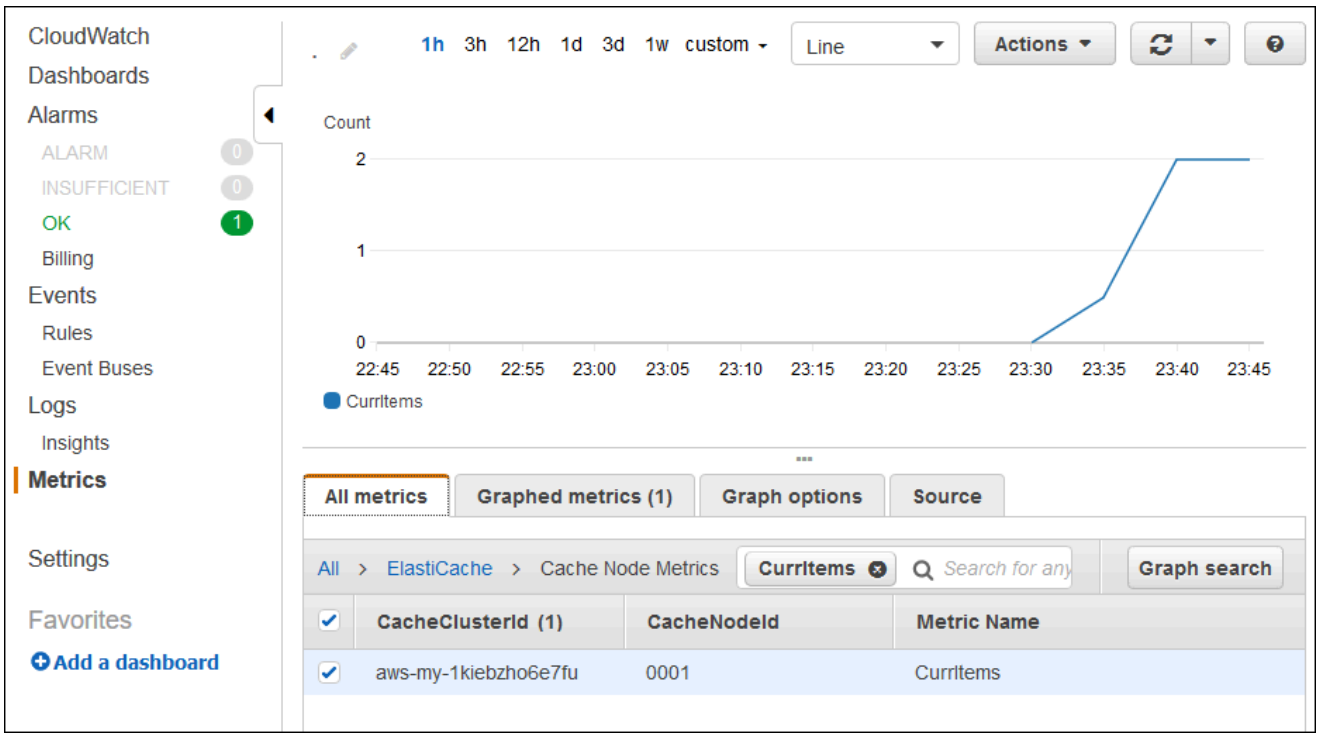
```
~/nodejs-example-express-elasticache$ npm install
```

7. 部署已更新的應用程式。

```
~/nodejs-example-express-elasticache$ eb deploy
```

8. 您的環境將在幾分鐘後更新。您的環境為綠色且就緒後，請確認該程式碼正常運作。

   a. 查看 Amazon 主 CloudWatch 控台以檢視您的 ElastiCache 指標。若要檢視 ElastiCache 指標，請選取左窗格中的「量度」，然後搜尋CurrItems。選取 ElastiCache > 快取節點測量結果，然後選取快取節點以檢視快取中的項目數。



> **Note**
>
> 請確認您查看的是應用程式所部署的相同區域。

如果您將應用程序 URL 複製並粘貼到另一個 Web 瀏覽器中並刷新頁面，則應該在 5 分鐘後看到 CurrItem 計數上升。

b. 擷取您記錄日誌的快照。如需擷取日誌的詳細資訊，請參閱在 Elastic Beanstalk 環境中檢視 Amazon EC2 執行個體的日誌。

c. 檢查日誌服務包中的檔案 `/var/log/nodejs/nodejs.log`。您應該會看到類似下列的內容：

```
Using memcached store nodes:
[ 'aws-my-1oys9co8zt1uo.1iwtrn.0001.use1.cache.amazonaws.com:11211' ]
```

清除

若您不想要再執行您的應用程式，可以終止您的環境並刪除您的應用程式來清除。

請使用 `eb terminate` 命令來終止您的環境，並使用 `eb delete` 命令來刪除您的應用程式。

終止環境

請從您建立本機存放庫的目錄中，執行 `eb terminate`。

```
$ eb terminate
```

此程序需要幾分鐘的時間。Elastic Beanstalk 會在成功終止環境後顯示訊息。

將 Node.js 應用程式與 DynamoDB 部署到 Elastic Beanstalk

本教學課程及其範例應用程式 nodejs-example-dynamo.zip 將逐步引導您完成部署 Node.js 應用程式的程序，該應用程式會 JavaScript 在 Node.js 中使用的 AWS 開發套件與 Amazon DynamoDB 服務互動。您將建立一個 DynamoDB 資料表，該資料庫位於與環境分離或外部的資料庫中。 AWS Elastic Beanstalk 您也需設定應用程式，以使用分開的資料庫。在生產環境中，最佳做法是使用與 Elastic Beanstalk 環境分開的資料庫，以使其獨立於環境的生命週期。此做法也可讓您執行藍/綠部署。

範例應用程式可說明下列項目：

- 儲存使用者提供之文字資料的 DynamoDB 資料表。
- 組態檔案，可用於建立資料表。
- Amazon Simple Notification Service 主題。
- 於部署期間使用 package.json 檔案，以安裝套件。

章節

必要條件

本教學課程需要下列先決條件：

- Node.js 執行階段
- 預設的 Node.js 套件管理工具軟體 npm
- Express 命令列產生器
- Elastic Beanstalk 命令列界面 (EB CLI)

有關安裝所列出之前三個元件和設定本機開發環境的詳細資訊，請參閱 設定您的 Node.js 開發環境。在本教學課程中，您不需要安裝 Node.js 的 AWS SDK，這也在參考的主題中提到。

有關安裝和設定 EB CLI 的詳細資訊，請參閱 安裝 EB CLI 和 設定 EB CLI。

建立 Elastic Beanstalk 環境

應用程式目錄

對於應用程式原始碼套件，本教學課程使用的是名為 `nodejs-example-dynamo` 的目錄。為本教學課程建立 `nodejs-example-dynamo` 目錄。

```
~$ mkdir nodejs-example-dynamo
```

> **Note**
>
> 本章中的每個教學課程皆會使用其自身的應用程式原始碼套件目錄。目錄名稱與教學課程所使用的範例應用程式名稱相符。

將您目前的工作目錄變更為 `nodejs-example-dynamo`。

```
~$ cd nodejs-example-dynamo
```

現在，來設定執行 Node.js 平台和範例應用程式的 Elastic Beanstalk 環境。我們將會使用 Elastic Beanstalk 命令列介面 (EB CLI)。

設定應用程式的 EB CLI 儲存庫，並建立執行 Node.js 平台的 Elastic Beanstalk 環境

1. 使用 eb init 命令建立一個儲存庫。

```
~/nodejs-example-dynamo$ eb init --platform node.js --region <region>
```

此命令會在名為 .elasticbeanstalk 的資料夾內建立組態檔案，其中會指定應用程式使用的環境設定，並以目前資料夾為名建立 Elastic Beanstalk 應用程式。

2. 使用 eb create 命令建立執行範例應用程式的環境。

```
~/nodejs-example-dynamo$ eb create --sample nodejs-example-dynamo
```

本命令會使用 Node.js 平台的預設設定和下列資源，建立負載平衡的環境：

- EC2 執行個體 – Amazon Elastic Compute Cloud (Amazon EC2) 虛擬機器，已設為在您選擇的平台上執行 Web 應用程式。

  每個平台會執行特定的一套軟體、設定檔和指令碼，來支援特定的語言版本、架構、Web 容器或其組合。大多數的平台使用會 Apache 或 NGINX 做為反向代理，此反向代理會在您 Web 應用程式的前景執行、轉傳遞交給此 Web 應用程式的請求、提供靜態資產，並產生存取和錯誤日誌。
- 執行個體安全群組 - Amazon EC2 安全群組，已設為允許從連接埠 80 傳入的流量。此資源可讓負載平衡器傳來的 HTTP 傳輸資料，到達執行您 Web 應用程式的 EC2 執行個體。在預設情況下，不允許傳輸資料從其他通訊埠傳送。
- 負載平衡器 - Elastic Load Balancing 負載平衡器，可設定將請求分配到執行您應用程式的執行個體。負載平衡器也讓您的執行個體不需直接連接到網際網路。
- 負載平衡器安全群組 - Amazon EC2 安全群組，設為允許從連接埠 80 傳入的流量。此資源可讓來自網際網路的 HTTP 傳輸資料到達負載平衡器。在預設情況下，不允許傳輸資料從其他通訊埠傳送。
- Auto Scaling 群組 - Auto Scaling 群組，設為在執行個體終止或無法使用時，取代該執行個體。
- Amazon S3 儲存貯體 - 儲存位置，用來儲存當您使用 Elastic Beanstalk 時所建立的原始程式碼、日誌和其他成品。
- Amazon CloudWatch 警示 — 監控環境中執行個體負載的兩個 CloudWatch 警示，並在負載過高或過低時觸發警示。當警示觸發時，您的 Auto Scaling 群組會擴展或縮減以進行回應。
- AWS CloudFormation 堆疊 — Elastic Beanstalk 用 AWS CloudFormation 來啟動環境中的資源並傳播組態變更。資源定義於範本中，您可在 AWS CloudFormation 主控台中檢視此範本。
- 網域名稱 – 會路由到您 Web 應用程式的網域名稱，其格式為 *subdomain*.*region*.elasticbeanstalk.com。

> **Note**
>
> 為了增強 Elastic Beanstalk 應用程式的安全性，我們會在公共后缀列表 (PSL) 中註冊網域 elasticbeanstalk.com。為了加強安全性，如果您需要在 Elastic Beanstalk 應用程式的預設網域名稱中設定敏感性 Cookie，我們建議您使用具 `__Host-` 前置詞的 Cookie。此做法將有助於保護您的網域免受跨站請求偽造 (CSRF) 攻擊。如需更多資訊，請參閱 Mozilla 開發人員網路中的設定 Cookie 頁面。

3. 環境建立完成後，請使用 eb open 命令，在預設瀏覽器中開啟環境 URL。

```
~/nodejs-example-dynamo$ eb open
```

您現在已使用範例應用程式建立 Node.js Elastic Beanstalk 環境。您可以使用自己的應用程式對其進行更新。接下來，我們會更新範例應用程式，以使用 Express 架構。

新增您環境執行個體的許可

您的應用程式會執行於負載平衡器後方的一個或多個 EC2 執行個體上，處理來自網際網路的 HTTP 請求。當應用程式收到要求使用 AWS 服務的要求時，應用程式會使用其執行個體的權限來存取這些服務。

範例應用程式會使用執行個體許可將資料寫入 DynamoDB 表格，並透過 Node.js JavaScript 中的 SDK 將通知傳送至 Amazon SNS 主題。將下列受管原則新增至預設執行個體設定檔，以授予您環境中的 EC2 執行個體存取 DynamoDB 和 Amazon SNS 的許可：

- AmazonDynamo資料庫 FullAccess
- 亞馬遜 SNS FullAccess

若要在預設的執行個體設定檔中新增原則

1. 在 IAM 主控台中開啟 角色頁面。
2. 選擇 aws-elasticbeanstalk-ec2 個角色。
3. 在 Permissions (許可) 標籤上，選擇 Attach policies (連接政策)。
4. 為您的應用程式使用的其他服務，選取受管原則，對於本教學課程，請選取 `AmazonSNSFullAccess` 和 `AmazonDynamoDBFullAccess`。
5. 選擇連接政策。

如需管理執行個體設定檔的詳細資訊，請參閱 管理 Elastic Beanstalk 執行個體描述檔。

部署範例應用程式

現在，您的環境已準備就緒，可供您部署和執行此教學課程的範例應用程式：nodejs-example-dynamo.zip。

部署和執行教學課程範例應用程式

1. 將目前的工作目錄變更為應用程式目錄 `nodejs-example-dynamo`。

```
~$ cd nodejs-example-dynamo
```

2. 下載範例應用程式來源套裝軟體 nodejs-example-dynamo.zip 的內容，並將其解壓縮至應用程式目錄`nodejs-example-dynamo`。
3. 使用 eb deploy 命令將範例應用程式部署至您的 Elastic Beanstalk 環境。

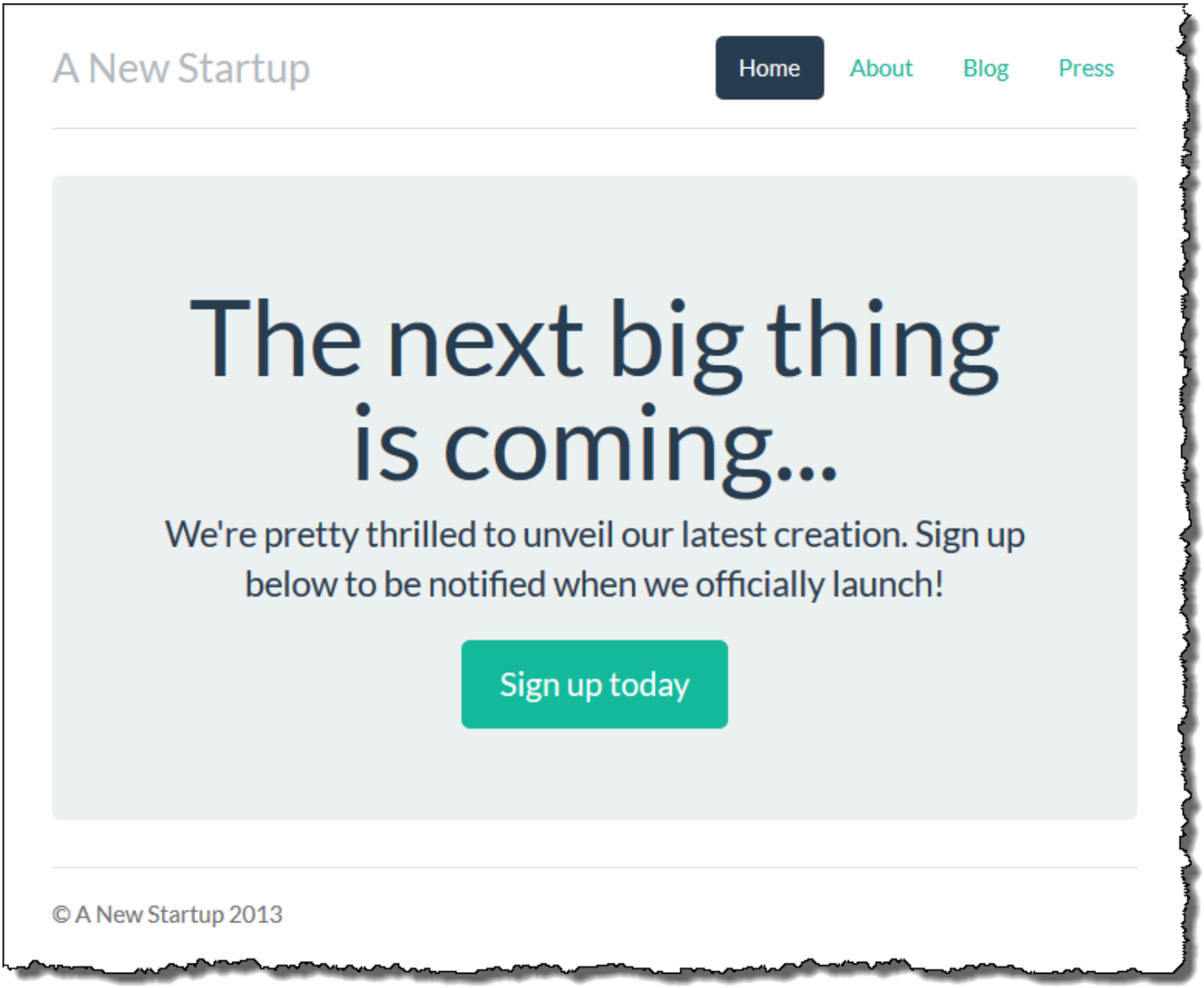
```
~/nodejs-example-dynamo$ eb deploy
```

> **Note**
>
> 依預設，此指`eb deploy`令會建立專案資料夾的 ZIP 檔案。您亦可設定 EB CLI 從建置程序部署組建成品，而非建立您專案資料夾的 ZIP 檔案。如需詳細資訊，請參閱 部署成品而非專案資料夾。

4. 環境建立完成後，請使用 eb open 命令，在預設瀏覽器中開啟環境 URL。

```
~/nodejs-example-dynamo$ eb open
```

本站點會收集使用者聯絡資訊，並使用 DynamoDB 資料表來存放資料。欲新增項目，選擇 Sign up today (立即註冊)，輸入名稱和電子郵件地址，然後選擇 Sign Up! (註冊！)。本 Web 應用程式會將表單內容寫入資料表，並觸發 Amazon SNS 電子郵件通知。



現在，Amazon SNS 主題會透過通知用的預留位置電子郵件進行設定。您很快就會更新此組態，但您可同時在 AWS Management Console內驗證 DynamoDB 資料表和 Amazon SNS 主題。

欲檢視資料表

1. 在 DynamoDB 主控台中開啟資料表頁面。
2. 尋找應用程式建立的資料表。該名稱以 awseb 開頭並包含。StartupSignupsTable
3. 選取資料表，選擇 Items (項目)，然後選擇 Start search (開始搜尋) 以檢視資料表的所有項目。

於註冊網站提交的每個電子郵件地址，在資料表內均有一個項目。本應用程式除了寫入資料表，亦會傳送訊息至具有兩個訂閱的 Amazon SNS 主題，一個用於向您發送電子郵件通知，另一個則用於

Amazon Simple Queue Service 佇列，工作者應用程式可自其中讀取以處理請求並傳送電子郵件至感興趣的客戶。

欲檢視主題

1. 在 Amazon SNS 主控台中開啟主題頁面。
2. 尋找應用程式建立的主題。該名稱以 awseb 開頭並包含。NewSignupTopic
3. 選擇主題以檢視其訂閱。

應用程式 (`app.js`) 會定義兩個路由。根路徑 (`/`) 返回從嵌入式 JavaScript （EJS）模板呈現的網頁，其中包含用戶填寫以註冊其姓名和電子郵件地址的表單。提交表單會將內含表單資料的 POST 請求傳送至 `/signup` 路由，該路由會寫入 DynamoDB 資料表，並發佈訊息至 Amazon SNS 主題以通知擁有者註冊事宜。

本範例應用程式內含組態檔案，其可建立 DynamoDB 資料表、Amazon SNS 主題和應用程式所使用的 Amazon SQS 佇列。這可讓您建立新的環境並立即測試功能，但缺點是將 DynamoDB 資料表與環境綁定。以生產環境而言，您應於環境外建立 DynamoDB 資料表，以避免您終止環境或更新其組態時遺失該資料表。

建立 DynamoDB 資料表

欲搭配於 Elastic Beanstalk 執行的應用程式使用外部 DynamoDB 資料表，請先於 DynamoDB 建立資料表。當您在 Elastic Beanstalk 外建立資瞭表時，它完全獨立於 Elastic Beanstalk 和您的 Elastic Beanstalk 環境，並且不會被 Elastic Beanstalk 終止。

透過下列設定建立資料表：

- Table name (資料表名稱) – **`nodejs-tutorial`**
- 主索引鍵 – **`email`**
- 主索引鍵類型 – String (字串)

建立 DynamoDB 資料表

1. 在 DynamoDB 管理主控台中開啟資料表頁面。
2. 選擇 建立資料表 。
3. 輸入 Table name (資料表名稱) 和 Primary key (主索引鍵)。
4. 選擇主索引鍵類型。

5. 選擇建立。

更新應用程式的組態檔案

更新應用程式原始碼內的組態檔案以使用 nodejs-tutorial (nodejs-tutorial) 資料表，無須建立新的資料表。

更新範例應用程式，以供生產使用

1. 將目前的工作目錄變更為應用程式目錄 `nodejs-example-dynamo`。

```
~$ cd nodejs-example-dynamo
```

2. 開啟 `.ebextensions/options.config` 並變更下列設定的值：

   - NewSignupEmail— 您的電子郵件地址。
   - STARTUP_SIGNUP_TABLE – nodejs-tutorial

   Example .ebextensions/options.config

```
option_settings:
  aws:elasticbeanstalk:customoption:
    NewSignupEmail: you@example.com
  aws:elasticbeanstalk:application:environment:
    THEME: "flatly"
    AWS_REGION: '`{"Ref" : "AWS::Region"}`'
    STARTUP_SIGNUP_TABLE: nodejs-tutorial
    NEW_SIGNUP_TOPIC: '`{"Ref" : "NewSignupTopic"}`'
  aws:elasticbeanstalk:container:nodejs:
    ProxyServer: nginx
  aws:elasticbeanstalk:container:nodejs:staticfiles:
    /static: /static
  aws:autoscaling:asg:
    Cooldown: "120"
  aws:autoscaling:trigger:
    Unit: "Percent"
    Period: "1"
    BreachDuration: "2"
    UpperThreshold: "75"
    LowerThreshold: "30"
```

```
    MeasureName: "CPUUtilization"
```

這將為應用程式套用以下設定：

- Amazon SNS 主題用於通知的電子郵件地址會設為您的地址，或您於 `options.config` 檔案中輸入的地址。
- 系統將會使用 nodejs-tutorial 資料表，而非 `.ebextensions/create-dynamodb-table.config` 所建立的資料表。

3. 移除 `.ebextensions/create-dynamodb-table.config`。

```
~/nodejs-tutorial$ rm .ebextensions/create-dynamodb-table.config
```

您下次部署應用程式時，將刪除此組態檔案建立的資料表。

4. 使用 eb deploy 命令將更新的應用程式部署至您的 Elastic Beanstalk 環境。

```
~/nodejs-example-dynamo$ eb deploy
```

5. 環境建立完成後，請使用 eb open 命令，在預設瀏覽器中開啟環境 URL。

```
~/nodejs-example-dynamo$ eb open
```

當您部署時，Elastic Beanstalk 會更新 Amazon SNS 主題的組態，並刪除您部署應用程式第一個版本時建立的 DynamoDB 資料表。

現在，當您終止環境時，將無法刪除 nodejs-tutorial (nodejs-tutorial) 資料表。這可讓您執行藍/綠部署、修改組態檔案或關閉您的網站而不會遺失資料。

在瀏覽器中開啟您的網站，並確認表單如預期運作。建立數個項目，然後檢查 DynamoDB 主控台來驗證資料表。

欲檢視資料表

1. 在 DynamoDB 主控台中開啟資料表頁面。
2. 尋找 nodejs-tutorial (nodejs-tutorial) 資料表。
3. 選取資料表，選擇 Items (項目)，然後選擇 Start search (開始搜尋) 以檢視資料表的所有項目。

您也可看到 Elastic Beanstalk 已刪除之前建立的資料表。

將您的環境設定為高可用性

最後，請增加執行個體計數下限，藉此設定您的環境 Auto Scaling 群組。隨時至少執行兩個執行個體，避免您環境中的 Web 伺服器出現單點故障，且無須停止網站服務即可部署變更。

若要設定您環境的 Auto Scaling 群組，以維持高可用性

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇 Configuration (組態)。
4. 在 Capacity (容量) 組態類別中，選擇 Edit (編輯)。
5. 在 Auto Scaling 群組區段，將最小執行個體設定為 **2**。
6. 若要儲存變更，請選擇頁面底部的儲存變更。

清除

當您完成使用 Elastic Beanstalk 時，即可終止您的環境。Elastic Beanstalk 會終止與您的環境相關聯的所有 AWS 資源，例如 Amazon EC2 執行個體、資料庫執行個體、負載平衡器、安全群組和警示。

從控制台終止 Elastic Beanstalk 環境

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 選擇 Actions (動作)，然後選擇 Terminate Environment (終止環境)。
4. 使用畫面顯示對話方塊來確認環境終止。

您可以使用 Elastic Beanstalk，隨時輕鬆地為您的應用程式建立新環境。

您也可以刪除您所建立的外部 DynamoDB 資料表。

刪除 DynamoDB 資料表

1. 在 DynamoDB 主控台中開啟資料表頁面。
2. 選擇資料表。
3. 選擇 Actions (動作)，然後再選擇 Delete table (刪除資料表)。
4. 選擇刪除。

後續步驟

範例應用程式使用組態檔案來設定軟體設定，並建立 AWS 資源做為您環境的一部分。如需關於組態檔案及其用途的詳細資訊，請參閱 使用組態檔案 (.ebextensions) 來進行進階的環境自訂。

本教學課程的範例應用程式使用 Node.js 適用的 Express Web 架構。如需有關 Express 的詳細資訊，請至 expressjs.com 參閱其官方文件。

最後，若您打算於生產環境中使用您的應用程式，請設定您環境的自訂網域名稱，並啟用 HTTPS 安全連線。

將 Amazon RDS 資料庫執行個體新增到您的 Node.js 應用程式環境

您可以使用 Amazon Relational Database Service (Amazon RDS) 資料庫執行個體存放由應用程式收集與修改的資料。資料庫可與環境耦合並由 Elastic Beanstalk 管理，或者由另一項服務在外部分開建立與管理。本主題提供了使用 Elastic Beanstalk 主控台建立 Amazon RDS 的說明。資料庫將與環境耦合並由 Elastic Beanstalk 管理。如需將 Amazon RDS 與 Elastic Beanstalk 整合的相關詳細資訊，請參閱 將資料庫新增至您的 Elastic Beanstalk 環境。

章節

- 將資料庫執行個體新增到您的環境
- 下載驅動程式
- 連線至資料庫

將資料庫執行個體新增到您的環境

欲將資料庫執行個體新增到您的環境

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。

2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇 Configuration (組態)。
4. 在 Database (資料庫) 組態類別中，選擇 Edit (編輯)。
5. 選擇資料庫引擎，並輸入使用者名稱和密碼。
6. 若要儲存變更，請選擇頁面底部的儲存變更。

新增資料庫執行個體約需要 10 分鐘。環境更新完成時，資料庫執行個體的主機名稱和其他連線資訊會透過下列環境屬性提供給您的應用程式：

屬性名稱	描述	屬性值
RDS_HOSTNAME	資料庫執行個體的主機名稱。	在 Amazon RDS 主控台：端點的連線能力和安全性索引標籤上。
RDS_PORT	資料庫執行個體接受連線的連接埠。預設值在不同資料庫引擎中有所差異。	在 Amazon RDS 主控台：連接埠的連線能力和安全性索引標籤上。
RDS_DB_NAME	資料庫名稱，**ebdb**。	在 Amazon RDS 主控台：資料庫名稱的組態索引標籤上。
RDS_USERNAME	您為資料庫設定的使用者名稱。	在 Amazon RDS 主控台：主要使用者名稱的組態索引標籤上。
RDS_PASSWORD	您為資料庫設定的密碼。	無法在 Amazon RDS 主控台中提供參考。

如需設定與 Elastic Beanstalk 環境耦合之資料庫執行個體的相關詳細資訊，請參閱 [將資料庫新增至您的 Elastic Beanstalk 環境]。

下載驅動程式

將資料庫驅動程式新增到您專案的 `package.json` 檔案的 `dependencies` 中。

Example **`package.json`** – Express 搭配 MySQL

```
{
  "name": "my-app",
  "version": "0.0.1",
  "private": true,
  "dependencies": {
    "ejs": "latest",
    "aws-sdk": "latest",
    "express": "latest",
    "body-parser": "latest",
    "mysql": "latest"
  },
  "scripts": {
    "start": "node app.js"
  }
}
```

適用於 Node.js 的常見驅動程式套件

- MySQL – mysql
- PostgreSQL – node-postgres
- SQL Server – node-mssql
- Oracle – node-oracledb

連線至資料庫

Elastic Beanstalk 會在環境屬性中提供已連接的資料庫執行個體連線資訊。使用 `process.env.`*`VARIABLE`* 來讀取屬性和設定資料庫的連線。

Example app.js – MySQL 資料庫連線

```
var mysql = require('mysql');

var connection = mysql.createConnection({
  host     : process.env.RDS_HOSTNAME,
  user     : process.env.RDS_USERNAME,
```

```
  password : process.env.RDS_PASSWORD,
  port     : process.env.RDS_PORT
});

connection.connect(function(err) {
  if (err) {
    console.error('Database connection failed: ' + err.stack);
    return;
  }

  console.log('Connected to database.');
});

connection.end();
```

關於使用 node-mysql 來建構連線字串，詳細資訊請參閱 npmjs.org/package/mysql。

資源

在開發 Node.js 應用程式時，您可以造訪幾個地方來取得額外的協助：

資源	描述
GitHub	針對使用 GitHub 的 Node.js 來安裝 AWS 開發套件。
Node.js 開發論壇	提出您的問題，並取得意見回饋。
適用於 Node.js 的 AWS SDK (開發人員預覽)	可取得範本程式碼、文件、工具和其他資源等一應俱全的場所。

在 Elastic Beanstalk 上建立和部署 PHP 應用程式

AWS Elastic Beanstalk PHP 使您可以使用 Amazon Web Services 輕鬆部署，管理和擴展您的 PHP Web 應用程序。本章提供了將 PHP Web 應用程序部署到 Elastic Beanstalk 的說明。您可以使用彈性 Beanstalk 命令列介面 (EB CLI) 或使用彈性 BeanElastic Beanstalk 主控台，在短短幾分鐘內部署應用程式。

本章提供下列教學課程：

- QuickStart 對於 PHP— 使用 EB CLI 部署您好世界 PHP 應用程式。

- 範例應用程式和教學— 有關 CakePHP 和 Symfony 等常見架構的深入教學課程，並將 Amazon RDS 執行個體新增至您的 PHP 應用程式環境。

如果您需要對於 PHP 應用程式開發的協助，有幾個您可以尋求幫助的地方：

- GitHub— 使用安裝 AWS SDK for PHP GitHub。
- PHP 開發人員中心-一站式服務，提供示例代碼，文檔，工具和其他資源。
- AWS SDK for PHP 見問題集 — 取得常見問題的解答。

主題

- QuickStart：將 PHP 應用程序部署到 Elastic Beanstalk
- 設定您的 PHP 開發環境
- 使用 Elastic Beanstalk PHP 平台
- PHP 的更多示例應用程序和教程

QuickStart：將 PHP 應用程序部署到 Elastic Beanstalk

本 QuickStart 教學課程將引導您完成建立 PHP 應用程式並將其部署至 AWS Elastic Beanstalk 環境的程序。

> **Note**
>
> 本 QuickStart 自學課程主要用於示範目的。請勿將本教學課程中建立的應用程式用於生產流量。

章節

- 您的 AWS 帳戶
- 必要條件
- 第 1 步：創建一個 PHP 應用程序
- 步驟 2：在本機執行應用程式
- 步驟 3：使用 EB CLI 部署您的 PHP 應用程式
- 第 4 步：在 Elastic Beanstalk 上運行應用程序
- 步驟 5：清除

- AWS 您應用程式的資源
- 後續步驟
- 使用 Elastic Beanstalk 控制台進行部署

您的 AWS 帳戶

如果您還不是 AWS 客戶，則需要創建一個 AWS 帳戶。註冊使您可以訪問 Elastic Beanstalk 和您需要的其他 AWS 服務。

如果您已經有 AWS 帳戶，則可以轉到必要條件。

建立 AWS 帳戶

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」。

必要條件

為了遵循本指南的程序，您需要命令列終端機或 shell 來執行命令。命令清單前會出現提示字元 ($) 及目前的目錄名稱 (如有)。

```
~/eb-project$ this is a command
this is output
```

在 Linux 和 macOS 上，您可以使用偏好的 Shell 和套件軟體管理工具。在視窗上，你可以安裝視窗子系統為 Linux 得到一個視窗集成的 Ubuntu 和 Bash 的版本。

EB CLI

本教學使用 Elastic Beanstalk 命令列界面 (EB CLI)。關於安裝和設定 EB CLI 的詳細資訊，請參閱安裝 EB CLI 和設定 EB CLI。

PHP

按照 PHP 網站上的「安裝和配置」在本地計算機上安裝 PHP。

第 1 步：創建一個 PHP 應用程序

在這個例子中，我們創建了一個你好世界的 PHP 應用程序。PHP 應用程序可以用最小的開銷來創建。

建立專案目錄。

```
~$ mkdir eb-php
~$ cd eb-php
```

接下來，在項目目錄中創建一個`index.php`文件。運行 PHP 時，默認情況下提供此文件。

```
~/eb-php/
|-- index.php
```

將以下內容添加到您的`index.php`文件中。

Example **`~/eb-php/index.php`**

```
echo "Hello Elastic Beanstalk! This is a PHP application.";
```

步驟 2：在本機執行應用程式

執行下列命令以在本機執行應用程式。

```
~/eb-php$ php -S localhost:5000
```

`http://localhost:5000`在您的網頁瀏覽器中輸入 URL 位址。瀏覽器應該顯示「你好 Elastic Beanstalk！這是一個 PHP 應用程序。」

步驟 3：使用 EB CLI 部署您的 PHP 應用程式

執行下列命令，為此應用程式建立 Elastic Beanstalk 環境。

若要建立環境並部署您的 PHP 應用程式

1. 透過 eb init 命令初始化您的 EB CLI 儲存庫。

   ```
   ~/eb-php$ eb init -p php php-tutorial --region us-east-2
   ```

   此命令創建一個名為的應用程序，`php-tutorial`並配置您的本地存儲庫以創建具有最新 PHP 平台版本的環境。

2. (選用) 再次執行 eb init 來設定預設金鑰對，藉此使用 SSH 連接至執行您應用程式的 EC2 執行個體：

   ```
   ~/eb-php$ eb init
   Do you want to set up SSH for your instances?
   (y/n): y
   Select a keypair.
   1) my-keypair
   2) [ Create new KeyPair ]
   ```

   若您已有金鑰對，請選擇一對，或依照提示建立金鑰對。若未出現提示，或稍後需要變更設定，請執行 eb init -i。

3. 使用 eb create 建立環境並於其中部署您的應用程式。Elastic Beanstalk 會自動為您的應用程式建立一個壓縮檔案，並將其部署到環境中的 EC2 執行個體。部署應用程序後，Elastic Beanstalk 在端口 5000 上啟動它。

   ```
   ~/eb-php$ eb create php-env
   ```

   Elastic Beanstalk 大約需要五分鐘的時間來創建您的環境。

第 4 步：在 Elastic Beanstalk 上運行應用程序

當創建環境的過程完成後，打開您的網站eb open。

```
~/eb-php$ eb open
```

恭喜您！您已經部署了一個帶有 Elastic Beanstalk 的 PHP 應用程序！這會開啟瀏覽器視窗，並使用為應用程式建立的網域名稱。

步驟 5：清除

您可以在完成應用程式的工作後終止環境。Elastic Beanstalk 會終止與您的環境相關聯的所有 AWS 資源。

若要使用 EB CLI 終止 Elastic Beanstalk 環境，請執行下列命令。

```
~/eb-php$ eb terminate
```

AWS 您應用程式的資源

您剛剛建立了單一執行個體應用程式。它可作為單一 EC2 執行個體的簡單範例應用程式使用，因此不需要負載平衡或 auto 擴展。對於單個實例應用程序，Elastic Beanstalk 創建以下 AWS 資源：

- EC2 執行個體 – 設定在您所選平台上執行 Web 應用程式的 Amazon EC2 虛擬機器。

  每個平台會執行不同一套軟體、設定檔和指令碼，來支援特定的語言版本、架構、Web 容器或其組合。大多數的平台會使用 Apache 或 nginx 做為反向代理，處理您 Web 應用程式前端的網路流量、向它轉送請求、提供靜態資產，並產生存取和錯誤日誌。
- 執行個體安全群組 – 設定允許從連接埠 80 傳入流量的 Amazon EC2 安全群組。此資源可讓負載平衡器傳來的 HTTP 傳輸資料，到達執行您 Web 應用程式的 EC2 執行個體。在預設情況下，不允許傳輸資料從其他通訊埠傳送。
- Amazon S3 儲存貯體 - 儲存位置，用來儲存當您使用 Elastic Beanstalk 時所建立的原始程式碼、日誌和其他成品。
- Amazon CloudWatch 警示 — 兩個 CloudWatch 警示，用於監控環境中執行個體的負載，並在負載過高或過低時觸發。當警示觸發時，您的 Auto Scaling 群組會擴展或縮減以進行回應。
- AWS CloudFormation 堆疊 — Elastic Beanstalk 用 AWS CloudFormation 來啟動環境中的資源並傳播組態變更。資源定義於範本中，您可在 AWS CloudFormation 主控台中檢視此範本。
- 網域名稱 – 會路由到您 Web 應用程式的網域名稱，其格式為 *subdomain*.*region*.elasticbeanstalk.com。

Elastic Beanstalk 會管理所有這些資源。當您終止環境時，Elastic Beanstalk 會終止其中的所有資源。

後續步驟

在您擁有執行應用程式的環境後，可以隨時部署應用程式的新版本或不同的應用程式。部署新的應用程式版本非常快速，因無須佈建或重新啟動 EC2 執行個體。您也可以使用 Elastic Beanstalk 控制台探索您的新環境。如需詳細步驟，請參閱本指南的「入門」一章中的「探索您的環境」。

> **嘗試更多教學**
>
> 如果您想要嘗試使用不同範例應用程式的其他教學課程，請參閱PHP 的更多示例應用程序和教程。

在您部署一或兩個範例應用程式並準備好開始在本機開發和執行 PHP 應用程式之後，請參閱設定您的 PHP 開發環境。

使用 Elastic Beanstalk 控制台進行部署

您也可以使用 Elastic Beanstalk 控制台來啟動示例應用程序。如需詳細步驟，請參閱本指南的「入門」一章中的「建立範例應用程式」。

設定您的 PHP 開發環境

設定 PHP 開發環境於本機測試您的應用程式，之後再部署至 AWS Elastic Beanstalk。此主題概述開發環境設定步驟，並提供實用工具的安裝頁面連結。

如需了解適用所有語言的常見設定步驟和工具，請參閱設定您的開發機器。

章節

- 安裝 PHP
- 安裝 Composer
- 安裝適用於 PHP 的 AWS SDK
- 安裝 IDE 或文字編輯器

安裝 PHP

安裝 PHP 和一些常見的擴展。若您沒有特別需求，請取得最新版本。視您的平台和可用套件管理工具而異，步驟會有所不同。

在 Amazon Linux 上，請使用 yum：

```
$ sudo yum install php
$ sudo yum install php-mbstring
$ sudo yum install php-intl
```

> **Note**
>
> 若要取得符合您 Elastic Beanstalk PHP 平台版本上的版本的特定 PHP 套件版本，請使用命令 `yum search php` 來尋找可用的套件版本，例如 `php72`、`php72-mbstring` 和 `php72-intl`。接著使用 `sudo yum install` *`package`* 以進行安裝。

在 Ubuntu 上，使用 apt：

```
$ sudo apt install php-all-dev
$ sudo apt install php-intl
$ sudo apt install php-mbstring
```

在 OS-X 上，請使用 brew：

```
$ brew install php
$ brew install php-intl
```

> **Note**
>
> 若要取得符合您 Elastic Beanstalk PHP 平台版本上的版本的特定 PHP 套件版本，請參閱 Homebrew Formulae 來尋找可用的 PHP 版本，例如 `php@7.2`。接著使用 `brew install` *`package`* 以進行安裝。
>
> 根據版本狀況，`php-intl` 可能會包含在主要 PHP 套件中，而不存在於個別的套件。

在 Windows 10 上，安裝適用於 Linux 的 Windows Subsystem 取得 Ubuntu 並使用 apt 安裝 PHP。針對舊版，請至 windows.php.net 前往下載頁面以取得 PHP，並閱讀此頁面了解延伸模組的資訊。

安裝 PHP 後，請重新開啟您的終端機並執行 `php --version`，確認已安裝新版本且為預設狀態。

安裝 Composer

Composer 為 PHP 適用的依存項目管理工具。您可以使用它來安裝程式庫、追蹤應用程式的相依項目，並產生熱門的 PHP 架構專案。

從 getcomposer.org 以 PHP 指令碼安裝 Composer。

```
$ curl -s https://getcomposer.org/installer | php
```

此安裝程式會在目前的目錄產生 PHAR 檔案。將此檔案移到您環境 PATH 的位置，因此您可以將它做為可執行檔。

```
$ mv composer.phar ~/.local/bin/composer
```

使用 `require` 命令安裝資料庫。

```
$ composer require twig/twig
```

Composer 新增程式庫，讓您從本機安裝到專案的 `composer.json` 檔案。當您部署專案的程式碼，Elastic Beanstalk 會使用 Composer 將此檔案中所列的程式庫安裝在您環境的應用程式執行個體。

如果您在安裝 Composer 時遇到問題，請參閱作曲家文件。

安裝適用於 PHP 的 AWS SDK

若您需要從應用程式內管理 AWS 資源，請安裝 AWS SDK for PHP。例如，透過適用於 PHP 的開發套件，您可以使用 Amazon DynamoDB (DynamoDB) 來存放使用者和工作階段資訊，無須建立關聯式資料庫。

使用 Composer 安裝適用於 PHP 的開發套件。

```
$ composer require aws/aws-sdk-php
```

如需詳細資訊及安裝說明，請前往 AWS SDK for PHP 首頁。

安裝 IDE 或文字編輯器

整合開發環境 (IDE) 提供可加速應用程式開發的各種功能。若您尚未使用 IDE 進行 PHP 開發，請嘗試 Eclipse 和 PHPStorm，看哪個更適合您。

- 安裝 Eclipse
- 安裝 PhpStorm

> **Note**
>
> IDE 可能會於專案資料夾新增您不希望遞交給來源控制的檔案。欲避免將這些檔案遞交給來源控制，請使用 `.gitignore` 或等同來源控制工具的功能。

若您只想開始編碼且不需要 IDE 的所有功能，請考慮安裝 Sublime Text。

使用 Elastic Beanstalk PHP 平台

AWS Elastic Beanstalk 支持許多不同版本的 PHP 編程語言的平台。這些平台支援可獨立執行或於 Composer 底下執行的 PHP Web 應用程式。若要進一步了解，請參閱《AWS Elastic Beanstalk 平台文件》中的 PHP。

Elastic Beanstalk 提供組態選項，您可用其於 Elastic Beanstalk 環境中自訂 EC2 執行個體上執行的軟體。您可以設定應用程式所需的環境變數，啟用 Amazon S3 的日誌輪換，將包含靜態檔案的應用程式來源中的資料夾映射到代理伺服器提供的路徑，並進行常見 PHP 初始化設定。

Elastic Beanstalk 主控台中提供了修改正在執行環境組態的組態選項。要避免在終止環境的組態時遺失組態，您可以使用已儲存組態來儲存您的設定，並在之後套用至另一個環境。

若要將設定儲存於原始程式碼，您可以包含組態檔案。每次您建立環境或部署應用程式，組態檔案裡的設定就會套用。您也可以使用組態檔案來安裝套件、執行指令碼，並在部署期間執行其他執行個體自訂操作。

若使用 Composer，您可將 `composer.json` 檔案納入您的原始碼套件，藉此於部署期間安裝套件。

以未做為組態選項提供的進階 PHP 組態和 PHP 設定而言，您可以使用組態檔案來提供 `INI` 檔案，以此擴展並覆寫 Elastic Beanstalk 套用的預設設定，或以此安裝其他延伸模組。

在 Elastic Beanstalk 主控台中套用的設定會覆寫組態檔案中相同的設定 (如存在)。這可讓您在組態檔案中擁有預設設定，並以主控台的環境專屬設定覆寫之。如需優先順序以及其他變更設定方法的詳細資訊，請參閱組態選項。

如需各種擴充 Elastic Beanstalk Linux 類型平台方式的詳細資訊，請參閱 the section called “擴充 Linux 平台”。

PHP 8.1 on Amazon Linux 2 注意事項

如果使用的是 PHP 8.1 on Amazon Linux 2 平台分支，請閱讀本節。

PHP 8.1 on Amazon Linux 2 注意事項

> **Note**
>
> 本主題中的資訊僅適用於 PHP 8.1 on Amazon Linux 2 平台分支。不適用於基於 AL2023 的 PHP 平台分支。也不適用於 PHP 8.0 Amazon Linux 2 平台分支。

Elastic Beanstalk 會將 EC2 執行個體上 PHP 8.1 on Amazon Linux 2 平台分支的 PHP 8.1 相關 RPM 套件儲存在本機目錄中，而不是 Amazon Linux 儲存庫中。您可以使用 rpm -i 來安裝套件。從 PHP 8.1 平台版本 3.5.0 開始，Elastic Beanstalk 會將與 PHP 8.1 有關的 RPM 套件儲存在以下本機 EC2 目錄中。

`/opt/elasticbeanstalk/RPMS`

以下範例會安裝 php-debuginfo 套件。

```
$rpm -i /opt/elasticbeanstalk/RPMS/php-debuginfo-8.1.8-1.amzn2.x86_64.rpm
```

套件名稱中的版本會因 EC2 本機目錄 `/opt/elasticbeanstalk/RPMS` 中所列的實際版本而有所不同。使用相同的語法來安裝其他 PHP 8.1 RPM 套件。

展開以下區塊，顯示我們提供的 RPM 套件清單。

RPM 套件

以下清單提供 Elastic Beanstalk PHP 8.1 平台在 Amazon Linux 2 上提供的 RMP 套件。這些位於本機目錄 `/opt/elasticbeanstalk/RPMS` 中。

列示套件名稱中的版本編號 8.1.8-1 和 3.7.0-1 僅為範例。

- `php-8.1.8-1.amzn2.x86_64.rpm`
- `php-bcmath-8.1.8-1.amzn2.x86_64.rpm`
- `php-cli-8.1.8-1.amzn2.x86_64.rpm`
- `php-common-8.1.8-1.amzn2.x86_64.rpm`
- `php-dba-8.1.8-1.amzn2.x86_64.rpm`
- `php-dbg-8.1.8-1.amzn2.x86_64.rpm`
- `php-debuginfo-8.1.8-1.amzn2.x86_64.rpm`
- `php-devel-8.1.8-1.amzn2.x86_64.rpm`

- php-embedded-8.1.8-1.amzn2.x86_64.rpm
- php-enchant-8.1.8-1.amzn2.x86_64.rpm
- php-fpm-8.1.8-1.amzn2.x86_64.rpm
- php-gd-8.1.8-1.amzn2.x86_64.rpm
- php-gmp-8.1.8-1.amzn2.x86_64.rpm
- php-intl-8.1.8-1.amzn2.x86_64.rpm
- php-ldap-8.1.8-1.amzn2.x86_64.rpm
- php-mbstring-8.1.8-1.amzn2.x86_64.rpm
- php-mysqlnd-8.1.8-1.amzn2.x86_64.rpm
- php-odbc-8.1.8-1.amzn2.x86_64.rpm
- php-opcache-8.1.8-1.amzn2.x86_64.rpm
- php-pdo-8.1.8-1.amzn2.x86_64.rpm
- php-pear-1.10.13-1.amzn2.noarch.rpm
- php-pgsql-8.1.8-1.amzn2.x86_64.rpm
- php-process-8.1.8-1.amzn2.x86_64.rpm
- php-pspell-8.1.8-1.amzn2.x86_64.rpm
- php-snmp-8.1.8-1.amzn2.x86_64.rpm
- php-soap-8.1.8-1.amzn2.x86_64.rpm
- php-sodium-8.1.8-1.amzn2.x86_64.rpm
- php-xml-8.1.8-1.amzn2.x86_64.rpm
- php-pecl-imagick-3.7.0-1.amzn2.x86_64.rpm
- php-pecl-imagick-debuginfo-3.7.0-1.amzn2.x86_64.rpm
- php-pecl-imagick-devel-3.7.0-1.amzn2.noarch.rpm

您可以使用 PEAR 和 PECL 套件來安裝常見的擴充功能。如需 PEAR 的詳細資訊，請參閱 PEAR PHP 擴充功能和應用程式儲存庫網站。如需 PECL 的詳細資訊，請參閱 PECL 擴充功能網站。

以下範例命令會安裝 Memcached 擴充功能。

```
$pecl install memcache
```

或者您也可以使用以下內容：

```
$pear install pecl/memcache
```

以下範例命令會安裝 Redis 擴充功能。

```
$pecl install redis
```

或者您也可以使用以下內容：

```
$pear install pecl/redis
```

設定您的 PHP 環境

您可以使用 Elastic Beanstalk 主控台來啟用至 Amazon S3 的日誌輪換，設定您的應用程式可以從環境讀取的變數，並變更 PHP 設定。

在 Elastic Beanstalk 主控台中設定 PHP 環境

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

   > **Note**
   >
   > 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇組態。
4. 在更新、監控和日誌記錄組態類別中，選擇編輯。

PHP 設定

- 代理伺服器 – 要在您的環境執行個體上使用的代理伺服器。預設使用 nginx。
- 文件根目錄 – 內含您網站預設頁面的資料夾。若您的歡迎頁面並非位於原始碼套件的根目錄，請指定相對於根路徑所在的資料夾。例如，若歡迎頁面位於名為 `/public` 的資料夾，則為 `public`。
- 記憶體限制 – 允許指令碼進行記憶體配置的容量上限。例如 `512M`。
- Zlib 輸出壓縮 – 設定為 `On` 以壓縮回應資料。
- 允許 URL fopen – 設定為 `Off` 以避免指令碼自遠端位置下載檔案。
- 顯示錯誤 – 設定為 `On` 以顯示用於除錯的內部錯誤訊息。
- 執行時間上限 – 指令碼於環境將其終止前可執行的時間上限 (秒)。

日誌選項

Log Options (日誌選項) 區段有兩個設定：

- Instance profile (執行個體描述檔) – 指定執行個體描述檔，此描述檔具備存取和您應用程式相關聯 Amazon S3 儲存貯體的許可。
- Enable log file rotation to Amazon S3 (啟用 Amazon S3 的日誌檔案輪換) – 指定是否將應用程式 Amazon EC2 執行個體的日誌檔案複製到與應用程式關聯的 Amazon S3 儲存貯體。

靜態檔案

為改善效能，您可以使用 Static files (靜態檔案) 區段來設定代理伺服器，以為來自 Web 應用程式一組目錄中的靜態檔案 (例如 HTML 或影像) 提供服務。對於每個目錄，您可以設定目錄映射的虛擬路徑。代理伺服器收到位於指定路徑下的檔案請求時，會直接提供檔案而非將請求路由至您的應用程式。

如需使用組態檔案或 Elastic Beanstalk 主控台設定靜態檔案的詳細資訊，請參閱the section called “靜態檔案”。

環境屬性

Environment Properties (環境屬性) 的部分可讓您針對執行您應用程式的 Amazon EC2 執行個體，來指定其上的環境資訊設定。這些設定會以金鑰值對的形式傳到應用程式。

您的應用程式程式碼可以使用 `$_SERVER` 或 `get_cfg_var` 函數來存取環境屬性。

```
$endpoint = $_SERVER['API_ENDPOINT'];
```

如需詳細資訊，請參閱環境屬性與其他軟體設定。

aws:elasticbeanstalk:container:php:phpini 命名空間

您可以使用組態檔案來設定組態選項，並在部署期間執行其他的執行個體設定工作。組態選項可透過 Elastic Beanstalk 服務或您使用的平台來定義，並且會組織成命名空間。

您可以使用 `aws:elasticbeanstalk:environment:proxy` 命名空間來選擇環境的代理伺服器。

您可以使用 `aws:elasticbeanstalk:environment:proxy:staticfiles` 命名空間來設定環境代理以提供靜態檔案。您可以定義虛擬路徑到應用程式目錄的對應項目。

PHP 平台會定義 `aws:elasticbeanstalk:container:php:phpini` 命名空間中的選項，Elastic Beanstalk 主控台中無法使用的命名空間也包含在內。當透過 `composer.phar install` 使用

Composer 安裝相依性時，composer_options 會設定使用自訂選項。如需包含可用選項的詳細資訊，請前往 http://getcomposer.org/doc/03-cli.md#install。

下列範例組態檔案會指定靜態檔案選項， 將名為 staticimages 的目錄對應至路徑 /images，並顯示 aws:elasticbeanstalk:container:php:phpini 命名空間中每個可用選項的設定：

Example .ebextensions/php-settings.config

```
option_settings:
  aws:elasticbeanstalk:environment:proxy:
    ProxyServer: apache
  aws:elasticbeanstalk:environment:proxy:staticfiles:
    /images: staticimages
  aws:elasticbeanstalk:container:php:phpini:
    document_root: /public
    memory_limit: 128M
    zlib.output_compression: "Off"
    allow_url_fopen: "On"
    display_errors: "Off"
    max_execution_time: 60
    composer_options: vendor/package
```

> **Note**
>
> aws:elasticbeanstalk:environment:proxy:staticfiles 命名空間未在 Amazon Linux AMI PHP 平台分支上定義 (前述的 Amazon Linux 2)。

Elastic Beanstalk 可提供許多組態選項讓您自訂環境。除了組態檔案，您也可以使用主控台、已儲存組態、EB CLI 或 AWS CLI來設定組態選項。如需詳細資訊，請參閱組態選項。

安裝您應用程式的相依性

您的應用程式可能會對其他 PHP 套件有相依性。您可以設定應用程式，以在本環境的 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體上安裝相依性。或者也可以將您應用程式的相依性加到原始碼套件內，再與應用程式一同部署。下節會介紹這兩種方式。

使用 Composer 檔案在執行個體上安裝相依性

請使用您專案來源根目錄內的 composer.json 檔案，使用 Composer 在您環境的 Amazon EC2 執行個體上安裝您應用程式所需的套件。

Example composer.json

```
{
    "require": {
        "monolog/monolog": "1.0.*"
    }
}
```

當 composer.json 檔案存在時，Elastic Beanstalk 會執行 composer.phar install 來安裝相依性。您可於 aws:elasticbeanstalk:container:php:phpini 命名空間設定 composer_options 選項，藉此新增附加命令的選項。

在原始碼套件中加入相依性

如果您的應用程式擁有相當大量的相依性，安裝過程可能會花上相當長的時間。如此會增加部署和擴展方面的操作，因為每個新的執行個體上都要安裝相依性。

若要避免影響到部署時間，請在您的開發環境中使用 Composer 來解決相依性的問題，並將相依性安裝到 vendor 資料夾。

將相依性加入應用程式原始碼套件

1. 執行以下命令：

   ```
   % composer install
   ```

2. 將產生的 vendor 資料夾加到您應用程式來源原始碼套件的根目錄。

當 Elastic Beanstalk 找到執行個體上的 vendor 資料夾時，會忽略 composer.json 檔案 (即使存在)。接著您的應用程式會使用來自 vendor 資料夾的相依性。

更新 Composer

如果您在嘗試使用 Composer 檔案安裝套件時看到錯誤訊息，或者無法使用最新的平台版本，您可能必須更新 Composer。在平台更新之間，您可以透過使用.ebextensions資料夾中的組態檔案來更新環境執行個體中的 Composer。

您可以使用以下配置自我更新作曲家。

```
commands:
  01updateComposer:
```

```
    command: /usr/bin/composer.phar self-update 2.7.0
```

下列選項設定會設定COMPOSER_HOME環境變數，以設定 Composer 快取的位置。

```
option_settings:
  - namespace: aws:elasticbeanstalk:application:environment
    option_name: COMPOSER_HOME
    value: /home/webapp/composer-home
```

您可以將這兩者合併到.ebextensions資料夾中的相同組態檔案中。

Example .ebextensions/composer.config

```
commands:
  01updateComposer:
    command: /usr/bin/composer.phar self-update 2.7.0

option_settings:
  - namespace: aws:elasticbeanstalk:application:environment
    option_name: COMPOSER_HOME
    value: /home/webapp/composer-home
```

Note

由於在 2024 年 2 月 22 日 AL2023 平台發行版本和 2024 年 2 月 28 日 AL2 平台版本中對 Composer 安裝進行了更新，如果COMPOSER_HOME在執行自我更新時設定 Composer 自我更新可能會失敗。
以下組合命令將無法執行：`export COMPOSER_HOME=/home/webapp/composer-home && /usr/bin/composer.phar self-update 2.7.0`
但是，前面的例子將起作用。在前面的範例中，的選項設定不COMPOSER_HOME會傳遞給01updateComposer執行項目，而且在執行自我更新指令時也不會設定。

Important

若您的 `composer.phar self-update` 命令省略版本編號，則您每次部署原始碼時，或當 Auto Scaling 佈建新的執行個體時，Composer 都將更新。若已發行的 Composer 版本與您的應用程式不相容，可能導致擴展操作和部署失敗。

如需 Elastic Beanstalk PHP 平台的詳細資訊，包括 Composer 版本，請參閱文件《AWS Elastic Beanstalk 平台》中的 PHP 平台版本。

擴展 php.ini

使用帶有 `files` 區塊的組態檔案，將 `.ini` 檔案新增至您環境執行個體上的 `/etc/php.d/`。主要的組態檔案 `php.ini`，會自此資料夾依字母順序提取檔案的設定。此資料夾內的檔案預設會啟用許多延伸模組。

Example .ebextensions/mongo.config

```
files:
  "/etc/php.d/99mongo.ini":
    mode: "000755"
    owner: root
    group: root
    content: |
      extension=mongo.so
```

PHP 的更多示例應用程序和教程

若要開始使用 PHP 應用程式 AWS Elastic Beanstalk，您只需要一個應用程式來源套件即可上傳為您的第一個應用程式版本，並部署至環境。本主QuickStart 對於 PHP題將逐步引導您使用 EB CLI 啟動範例 PHP 應用程式。本節提供更深入的教學課程。

PHP 教程

- 將 Laravel 應用程式部署至 Elastic Beanstalk
- 將 CakePHP 應用程式部署至 Elastic Beanstalk
- 將 Symfony 應用程式部署至 Elastic Beanstalk
- 使用外部 Amazon RDS 資料庫將高可用性 PHP 應用程式資料庫部署至 Elastic Beanstalk
- 將具有外部 Amazon RDS 資料庫的高可用性 WordPress 網站部署到 Elastic Beanstalk
- 使用外部 Amazon RDS 資料庫將高可用性 Drupal 網站資料庫部署至 Elastic Beanstalk
- 將 Amazon RDS 資料庫執行個體新增至您的 PHP 應用程式環境

將 Laravel 應用程式部署至 Elastic Beanstalk

拉維爾是一個用於 PHP 的開源 model-view-controller （MVC）框架。本教學將逐步引導您完成產生 Laravel 應用程式、將其部署到 AWS Elastic Beanstalk 環境，以及將其設定為連接至 Amazon 關聯式資料庫服務 (Amazon RDS) 資料庫執行個體的程序。

章節

必要條件

本教學假設您具備基本的 Elastic Beanstalk 操作及 Elastic Beanstalk 主控台知識。若您尚不了解，請依照 開始使用 Elastic Beanstalk 中的說明來啟動您的第一個 Elastic Beanstalk 環境。

為了遵循本指南的程序，您需要命令列終端機或 shell 來執行命令。命令清單前會出現提示字元 ($) 及目前的目錄名稱 (如有)。

```
~/eb-project$ this is a command
this is output
```

在 Linux 和 macOS 上，您可以使用偏好的 Shell 和套件軟體管理工具。在視窗上，你可以安裝視窗子系統為 Linux 得到一個視窗集成的 Ubuntu 和 Bash 的版本。

Laravel 6 需要 PHP 7.2 或更新版本。它還需要在官方 Laravel 文件中伺服器要求主題中列出的 PHP 擴充功能。按照主題 設定您的 PHP 開發環境 中的說明安裝 PHP 和 Composer。

對於 Laravel 支援和維護資訊，請參閱關於 Laravel 官方文件中的支援政策主題。

啟動 Elastic Beanstalk 環境

使用 Elastic Beanstalk 主控台建立 Elastic Beanstalk 環境。選擇 PHP (PHP) 平台，並接受預設的設定和範本程式碼。

啟動環境 (主控台)

1. 使用這個預先配置的鏈接打開 Elastic Beanstalk 控制台：控制台。aws.amazon.com/彈性豆/家 #/新 applicationName = 教程和 environmentType = LoadBalanced
2. 在 Platform (平台)，選取符合您應用程式所使用語言的平台和平台分支。
3. 針對 Application code (應用程式程式碼)，選擇 Sample application (範例應用程式)。
4. 選擇 Review and launch (檢閱和啟動)。
5. 檢視可用選項。選擇您要使用的可用選項，當您準備就緒時，請選擇 Creat app (建立應用程式)。

使用大約需要五分鐘時間建立環境，並且建立下列資源：

- EC2 執行個體 – Amazon Elastic Compute Cloud (Amazon EC2) 虛擬機器，已設為在您選擇的平台上執行 Web 應用程式。

  每個平台會執行特定的一套軟體、設定檔和指令碼，來支援特定的語言版本、架構、Web 容器或其組合。大多數的平台使用會 Apache 或 NGINX 做為反向代理，此反向代理會在您 Web 應用程式的前景執行、轉傳遞交給此 Web 應用程式的請求、提供靜態資產，並產生存取和錯誤日誌。
- 執行個體安全群組 - Amazon EC2 安全群組，已設為允許從連接埠 80 傳入的流量。此資源可讓負載平衡器傳來的 HTTP 傳輸資料，到達執行您 Web 應用程式的 EC2 執行個體。在預設情況下，不允許傳輸資料從其他通訊埠傳送。
- 負載平衡器 - Elastic Load Balancing 負載平衡器，可設定將請求分配到執行您應用程式的執行個體。負載平衡器也讓您的執行個體不需直接連接到網際網路。
- 負載平衡器安全群組 - Amazon EC2 安全群組，設為允許從連接埠 80 傳入的流量。此資源可讓來自網際網路的 HTTP 傳輸資料到達負載平衡器。在預設情況下，不允許傳輸資料從其他通訊埠傳送。
- Auto Scaling 群組 - Auto Scaling 群組，設為在執行個體終止或無法使用時，取代該執行個體。
- Amazon S3 儲存貯體 - 儲存位置，用來儲存當您使用 Elastic Beanstalk 時所建立的原始程式碼、日誌和其他成品。
- Amazon CloudWatch 警示 — 監控環境中執行個體負載的兩個 CloudWatch 警示，並在負載過高或過低時觸發警示。當警示觸發時，您的 Auto Scaling 群組會擴展或縮減以進行回應。
- AWS CloudFormation 堆疊 — Elastic Beanstalk 用 AWS CloudFormation 來啟動環境中的資源並傳播組態變更。資源定義於範本中，您可在 AWS CloudFormation 主控台中檢視此範本。

- 網域名稱 – 會路由到您 Web 應用程式的網域名稱，其格式為 *subdomain*.*region*.elasticbeanstalk.com。

> **Note**
>
> 為了增強 Elastic Beanstalk 應用程式的安全性，我們會在公共后缀列表 (PSL) 中註冊網域 elasticbeanstalk.com。為了加強安全性，如果您需要在 Elastic Beanstalk 應用程式的預設網域名稱中設定敏感性 Cookie，我們建議您使用具 `__Host-` 前置詞的 Cookie。此做法將有助於保護您的網域免受跨站請求偽造 (CSRF) 攻擊。如需更多資訊，請參閱 Mozilla 開發人員網路中的設定 Cookie 頁面。

這些資源全都由 Elastic Beanstalk 管理。當您終止環境時，Elastic Beanstalk 會終止其中的所有資源。

> **Note**
>
> Elastic Beanstalk 建立的 Amazon S3 儲存貯體會在環境間共享，且不會在環境終止時刪除。如需詳細資訊，請參閱 將 Elastic Beanstalk 與 Amazon S3 搭配使用。

安裝 Laravel 並產生網站

Composer 可透過一個命令安裝 Laravel 並建立工作專案：

```
~$ composer create-project --prefer-dist laravel/laravel eb-laravel
```

Composer 會安裝 Laravel 及其依存項目，並產生預設專案。

若您安裝 Laravel 出現任何問題，請造訪官方文件的安裝主題：https://laravel.com/docs/6.x

部署您的應用程式

建立原始碼套件，其中包含 Composer 所建立的檔案。以下命令建立一個名為 `laravel-default.zip` 的原始碼套件。它不含 `vendor` 資料夾中的檔案，其不僅佔用許多空間，在部署應用程式到 Elastic Beanstalk 時也派不上用場。

```
~/eb-laravel$ zip ../laravel-default.zip -r * .[^.]* -x "vendor/*"
```

將來源套件上傳至 Elastic Beanstalk，以將 Laravel 部署至您的環境。

若要部署原始碼套件

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在環境概觀頁面上，選擇 Upload and deploy (上傳和部署)。
4. 使用畫面顯示對話方塊來上傳原始碼套件。
5. 選擇 Deploy (部署)。
6. 部署完成後，您可選擇網站的 URL 以在新分頁中開啟您的網站。

> **Note**
>
> 若要進一步最佳化原始碼套件，可將 Git 儲存庫初始化，並使用 `git archive` 命令建立原始碼套件。預設的 Laravel 專案包含了 `.gitignore` 檔案，它告訴 Git 排除 `vendor` 資料夾和其他不需用於部署的檔案。

設定 Composer 設定值

完成部署時按一下 URL，即可在瀏覽器中開啟 Laravel 應用程式：

這是什麼？ 根據預設，Elastic Beanstalk 會將專案根目錄做為網站的根路徑。然而，在此情況中，預設頁面 (`index.php`) 位於 `public` 資料夾的下一層。您可將 `/public` 新增至 URL，藉此加以驗證。例如 `http://laravel.us-east-2.elasticbeanstalk.com/public`。

若要在根路徑提供 Laravel 應用程式，請使用 Elastic Beanstalk 主控台來設定網站的文件根。

設定網站的文件根目錄

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇組態。
4. 在更新、監控和日誌記錄組態類別中，選擇編輯。
5. 在 Document Root (文件根目錄) 部分，輸入 **`/public`**。
6. 若要儲存變更，請選擇頁面底部的儲存變更。
7. 更新完成時按一下 URL，即可在瀏覽器中重新開啟您的網站。

目前為止一切正常。接著，您要將資料庫新增至您的環境，並將 Laravel 設定為連接至該資料庫。

新增資料庫至您的環境

在您的 Elastic Beanstalk 環境啟動 RDS 資料庫執行個體。您可在 Elastic Beanstalk 上使用 MySQL、SQLServer 或 PostgreSQL 資料庫來搭配 Laravel。在此範例中，我們使用 MySQL。

將 RDS 資料庫執行個體新增至 Elastic Beanstalk 環境

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇 Configuration (組態)。
4. 在 Database (資料庫) 組態類別中，選擇 Edit (編輯)。
5. 針對 Engine (引擎) 部分，選擇 mysql (mysql)。
6. 輸入主要使用者名稱及密碼。Elastic Beanstalk 會使用環境屬性，將這些值提供給您的應用程式。
7. 若要儲存變更，請選擇頁面底部的儲存變更。

建立資料庫執行個體約需要 10 分鐘。如需有關耦合至 Elastic Beanstalk 環境之資料庫的詳細資訊，請參閱 將資料庫新增至您的 Elastic Beanstalk 環境。

在此同時，您可更新原始碼以讀取環境的連線資訊。Elastic Beanstalk 使用環境變數 (如 `RDS_HOSTNAME`) 提供連線詳細資訊，您可自應用程式存取這些變數。

Laravel 的資料庫組態存放於專案程式碼內 `database.php` 資料夾名為 `config` 的檔案中。尋找 `mysql` 項目並修改 `host`、`database`、`username`、and `password` 變數以讀取與 Elastic Beanstalk 對應的值：

Example ~/Eb-laravel/config/database.php

```
...
    'connections' => [

        'sqlite' => [
            'driver' => 'sqlite',
            'database' => env('DB_DATABASE', database_path('database.sqlite')),
            'prefix' => '',
        ],

        'mysql' => [
            'driver' => 'mysql',
            'host' => env('RDS_HOSTNAME', '127.0.0.1'),
            'port' => env('RDS_PORT', '3306'),
            'database' => env('RDS_DB_NAME', 'forge'),
            'username' => env('RDS_USERNAME', 'forge'),
            'password' => env('RDS_PASSWORD', ''),
```

```
            'unix_socket' => env('DB_SOCKET', ''),
            'charset' => 'utf8mb4',
            'collation' => 'utf8mb4_unicode_ci',
            'prefix' => '',
            'strict' => true,
            'engine' => null,
        ],
...
```

為了確認資料庫連線是否設定正確，請於 index.php 新增程式碼來連接至資料庫，並於預設回應新增一些程式碼：

Example ~/Eb-laravel/public/index.php

```
...
if(DB::connection()->getDatabaseName())
{
   echo "Connected to database ".DB::connection()->getDatabaseName();
}
$response->send();
...
```

資料庫執行個體啟動完成時，請封裝更新後的應用程式，並將其部署至您的環境。

更新您的 Elastic Beanstalk 環境

1. 建立新的原始碼套件：

   ```
   ~/eb-laravel$ zip ../laravel-v2-rds.zip -r * .[^.]* -x "vendor/*"
   ```

2. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
3. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

   > **Note**
   >
   > 如果您有許多環境，請使用搜尋列來篩選環境清單。

4. 選擇 Upload and Deploy (上傳並部署)。
5. 選擇 Browse (瀏覽)，然後上傳 laravel-v2-rds.zip。
6. 選擇部署。

部署應用程式的新版本不會超過一分鐘。部署完成時，請重新整理網頁，確認資料庫連線成功：

清除

當您完成使用 Elastic Beanstalk 時，即可終止您的環境。Elastic Beanstalk 會終止與您的環境相關聯的所有 AWS 資源，例如 Amazon EC2 執行個體、資料庫執行個體、負載平衡器、安全群組和警示。

從控制台終止 Elastic Beanstalk 環境

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 選擇 Actions (動作)，然後選擇 Terminate Environment (終止環境)。
4. 使用畫面顯示對話方塊來確認環境終止。

您可以使用 Elastic Beanstalk，隨時輕鬆地為您的應用程式建立新環境。

此外，您可終止於 Elastic Beanstalk 環境外建立的資料庫資源。終止 Amazon RDS 資料庫執行個體時，您可擷取快照，稍後再將資料還原至另一個執行個體。

終止 RDS 資料庫執行個體

1. 開啟 Amazon RDS 主控台。
2. 選擇 Databases (資料庫)。
3. 選擇資料庫執行個體。
4. 選擇動作，然後選擇刪除。

5. 選擇是否建立快照，然後選擇 Delete (刪除)。

後續步驟

有關 Laravel 的詳細資訊，請訪問 Laravel 官方網站，laravel.com。

隨著您繼續開發應用程式，您可能會希望無須手動建立 .zip 檔案並將其上傳至 Elastic Beanstalk 主控台，即可管理環境和部署應用程式。E lastic Beanstalk 命令列介面 (EB CLI) 提供從命令列建立、設定和部署應用程式至 Elastic Beanstalk 環境的 easy-to-use 指令。

在本教學中，您使用 Elastic Beanstalk 主控台來設定 Composer 選項。為了讓此組態成為應用程式原始碼的一部分，您的，您可以使用組態檔，如下所示。

Example .ebextensions/composer.config

```
option_settings:
  aws:elasticbeanstalk:container:php:phpini:
    document_root: /public
```

如需詳細資訊，請參閱 使用組態檔案 (.ebextensions) 來進行進階的環境自訂。

在 Elastic Beanstalk 環境中執行 Amazon RDS 資料庫執行個體對於開發和測試非常實用，但會將資料庫的生命週期連結至您的環境。如需連接至在環境外執行的資料庫的說明，請參閱 將 Amazon RDS 資料庫執行個體新增至您的 PHP 應用程式環境。

最後，若您打算於生產環境中使用您的應用程式，建議您設定您環境的自訂網域名稱，並啟用 HTTPS 安全連線。

將 CakePHP 應用程式部署至 Elastic Beanstalk

CakePHP 為 PHP 適用的開放原始碼 MVC 架構。本教學會逐步解說如何產生 CakePHP 專案、將其部署至 Elastic Beanstalk 環境，以及將其設定為連接至 Amazon RDS 資料庫執行個體。

章節

- 必要條件
- 啟動 Elastic Beanstalk 環境
- 安裝 CakePHP 並產生網站
- 部署您的應用程式
- 新增資料庫至您的環境

- 清除
- 後續步驟

必要條件

本教學假設您具備基本的 Elastic Beanstalk 操作及 Elastic Beanstalk 主控台知識。若您尚不了解，請依照 開始使用 Elastic Beanstalk 中的說明來啟動您的第一個 Elastic Beanstalk 環境。

為了遵循本指南的程序，您需要命令列終端機或 shell 來執行命令。命令清單前會出現提示字元 ($) 及目前的目錄名稱 (如有)。

```
~/eb-project$ this is a command
this is output
```

在 Linux 和 macOS 上，您可以使用偏好的 Shell 和套件軟體管理工具。在視窗上，你可以安裝視窗子系統為 Linux 得到一個視窗集成的 Ubuntu 和 Bash 的版本。

CakePHP 4 需要 PHP 7.2 或更新版本。它還需要在官方 CakePHP 安裝文件中列出的 PHP 擴充功能。按照主題 設定您的 PHP 開發環境 中的說明安裝 PHP 和 Composer。

啟動 Elastic Beanstalk 環境

使用 Elastic Beanstalk 主控台建立 Elastic Beanstalk 環境。選擇 PHP (PHP) 平台，並接受預設的設定和範本程式碼。

啟動環境 (主控台)

1. 使用這個預先配置的鏈接打開 Elastic Beanstalk 控制台：控制台。aws.amazon.com/彈性豆/家 # /新 applicationName = 教程和 environmentType = LoadBalanced
2. 在 Platform (平台)，選取符合您應用程式所使用語言的平台和平台分支。
3. 針對 Application code (應用程式程式碼)，選擇 Sample application (範例應用程式)。
4. 選擇 Review and launch (檢閱和啟動)。
5. 檢視可用選項。選擇您要使用的可用選項，當您準備就緒時，請選擇 Creat app (建立應用程式)。

使用大約需要五分鐘時間建立環境，並且建立下列資源：

- EC2 執行個體 – Amazon Elastic Compute Cloud (Amazon EC2) 虛擬機器，已設為在您選擇的平台上執行 Web 應用程式。

每個平台會執行特定的一套軟體、設定檔和指令碼，來支援特定的語言版本、架構、Web 容器或其組合。大多數的平台使用會 Apache 或 NGINX 做為反向代理，此反向代理會在您 Web 應用程式的前景執行、轉傳遞交給此 Web 應用程式的請求、提供靜態資產，並產生存取和錯誤日誌。

- 執行個體安全群組 - Amazon EC2 安全群組，已設為允許從連接埠 80 傳入的流量。此資源可讓負載平衡器傳來的 HTTP 傳輸資料，到達執行您 Web 應用程式的 EC2 執行個體。在預設情況下，不允許傳輸資料從其他通訊埠傳送。
- 負載平衡器 - Elastic Load Balancing 負載平衡器，可設定將請求分配到執行您應用程式的執行個體。負載平衡器也讓您的執行個體不需直接連接到網際網路。
- 負載平衡器安全群組 - Amazon EC2 安全群組，設為允許從連接埠 80 傳入的流量。此資源可讓來自網際網路的 HTTP 傳輸資料到達負載平衡器。在預設情況下，不允許傳輸資料從其他通訊埠傳送。
- Auto Scaling 群組 - Auto Scaling 群組，設為在執行個體終止或無法使用時，取代該執行個體。
- Amazon S3 儲存貯體 - 儲存位置，用來儲存當您使用 Elastic Beanstalk 時所建立的原始程式碼、日誌和其他成品。
- Amazon CloudWatch 警示 — 監控環境中執行個體負載的兩個 CloudWatch 警示，並在負載過高或過低時觸發警示。當警示觸發時，您的 Auto Scaling 群組會擴展或縮減以進行回應。
- AWS CloudFormation 堆疊 — Elastic Beanstalk 用 AWS CloudFormation 來啟動環境中的資源並傳播組態變更。資源定義於範本中，您可在 AWS CloudFormation 主控台中檢視此範本。
- 網域名稱 – 會路由到您 Web 應用程式的網域名稱，其格式為 *subdomain*.*region*.elasticbeanstalk.com。

> **Note**
>
> 為了增強 Elastic Beanstalk 應用程式的安全性，我們會在公共后缀列表 (PSL) 中註冊網域 elasticbeanstalk.com。為了加強安全性，如果您需要在 Elastic Beanstalk 應用程式的預設網域名稱中設定敏感性 Cookie，我們建議您使用具 `__Host-` 前置詞的 Cookie。此做法將有助於保護您的網域免受跨站請求偽造 (CSRF) 攻擊。如需更多資訊，請參閱 Mozilla 開發人員網路中的設定 Cookie 頁面。

這些資源全都由 Elastic Beanstalk 管理。當您終止環境時，Elastic Beanstalk 會終止其中的所有資源。

> **Note**
>
> Elastic Beanstalk 建立的 Amazon S3 儲存貯體會在環境間共享，且不會在環境終止時刪除。如需詳細資訊，請參閱 將 Elastic Beanstalk 與 Amazon S3 搭配使用。

安裝 CakePHP 並產生網站

Composer 可透過一個命令安裝 CakePHP 並建立工作專案：

```
~$ composer create-project --prefer-dist cakephp/app eb-cake
```

Composer 會安裝 CakePHP 及約 20 個依存項目，並產生預設專案。

若您安裝 CakePHP 出現任何問題，請造訪官方文件的安裝主題：http://book.cakephp.org/4.0/en/installation.html

部署您的應用程式

建立原始碼套件，其中包含 Composer 所建立的檔案。以下命令建立一個名為 `cake-default.zip` 的原始碼套件。它不含 `vendor` 資料夾中的檔案，其不僅佔用許多空間，在部署應用程式到 Elastic Beanstalk 時也派不上用場。

```
eb-cake zip ../cake-default.zip -r * .[^.]* -x "vendor/*"
```

將來源套件上傳至 Elastic Beanstalk，以將 CakePHP 部署至您的環境。

若要部署原始碼套件

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

   > **Note**
   >
   > 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在環境概觀頁面上，選擇 Upload and deploy (上傳和部署)。
4. 使用畫面顯示對話方塊來上傳原始碼套件。

5. 選擇 Deploy (部署)。
6. 部署完成後，您可選擇網站的 URL 以在新分頁中開啟您的網站。

> **Note**
>
> 若要進一步最佳化原始碼套件，可將 Git 儲存庫初始化，並使用 `git archive` 命令建立原始碼套件。預設的 Symfony 專案包含了 `.gitignore` 檔案，它告訴 Git 排除 `vendor` 資料夾和其他不需用於部署的檔案。

程序完成時按一下 URL，即可在瀏覽器中開啟 CakePHP 應用程式。

目前為止一切正常。接著，您要將資料庫新增至您的環境，並將 CakePHP 設定為連接至該資料庫。

新增資料庫至您的環境

在您的 Elastic Beanstalk 環境中啟動 Amazon RDS 資料庫執行個體。您可在 Elastic Beanstalk 上使用 MySQL、SQLServer 或 PostgreSQL 資料庫來搭配 CakePHP。在此範例中，我們使用 PostgreSQL。

將 Amazon RDS 資料庫執行個體新增至 Elastic Beanstalk 環境

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇組態。
4. 在 Database (資料庫) 下，選擇 Edit (編輯)。
5. 在 DB engine (資料庫引擎) 區段，選擇 postgres。
6. 輸入主要使用者名稱及密碼。Elastic Beanstalk 會使用環境屬性，將這些值提供給您的應用程式。
7. 若要儲存變更，請選擇頁面底部的儲存變更。

建立資料庫執行個體約需要 10 分鐘。在此同時，您可更新原始碼以讀取環境的連線資訊。Elastic Beanstalk 使用環境變數 (如 `RDS_HOSTNAME`) 提供連線詳細資訊，您可自應用程式存取這些變數。

CakePHP 的資料庫組態位於專案程式碼內 app.php 資料夾名為 config 的檔案中。開啟此檔案，並新增可自 $_SERVER 讀取環境變數並將其指派至本機變數的程式碼。插入下例第一行 (<?php) 後醒目提示的行：

Example ~/Eb-cake/config/app.php

```
<?php
if (!defined('RDS_HOSTNAME')) {
  define('RDS_HOSTNAME', $_SERVER['RDS_HOSTNAME']);
  define('RDS_USERNAME', $_SERVER['RDS_USERNAME']);
  define('RDS_PASSWORD', $_SERVER['RDS_PASSWORD']);
  define('RDS_DB_NAME', $_SERVER['RDS_DB_NAME']);
}
return [
...
```

資料庫連線可在 app.php 中進一步設定。尋找下列區段，並透過與您資料庫引擎 (Mysql、Sqlserver 或 Postgres) 相符的驅動程式名稱修改預設資料來源，然後設定 host、username、password 和 database 變數以從 Elastic Beanstalk 讀取對應的值：

Example ~/Eb-cake/config/app.php

```
...
     /**
     * Connection information used by the ORM to connect
     * to your application's datastores.
     * Drivers include Mysql Postgres Sqlite Sqlserver
     * See vendor\cakephp\cakephp\src\Database\Driver for complete list
     */
    'Datasources' => [
        'default' => [
            'className' => 'Cake\Database\Connection',
            'driver' => 'Cake\Database\Driver\Postgres',
            'persistent' => false,
            'host' => RDS_HOSTNAME,
            /*
             * CakePHP will use the default DB port based on the driver selected
             * MySQL on MAMP uses port 8889, MAMP users will want to uncomment
             * the following line and set the port accordingly
             */
            //'port' => 'non_standard_port_number',
            'username' => RDS_USERNAME,
```

```
            'password' => RDS_PASSWORD,
            'database' => RDS_DB_NAME,
            /*
             * You do not need to set this flag to use full utf-8 encoding (internal
 default since CakePHP 3.6).
             */
            //'encoding' => 'utf8mb4',
            'timezone' => 'UTC',
            'flags' => [],
            'cacheMetadata' => true,
            'log' => false,
...
```

資料庫執行個體啟動完成時，請封裝更新後的應用程式，並將其部署至您的環境：

更新您的 Elastic Beanstalk 環境

1. 建立新的原始碼套件：

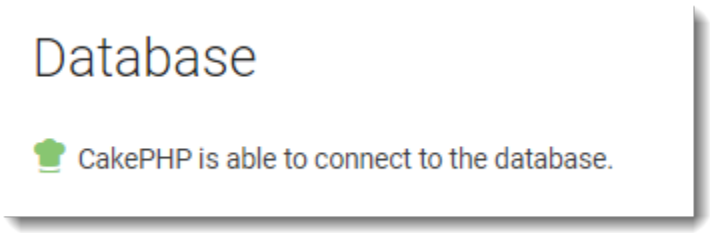
   ```
   ~/eb-cake$ zip ../cake-v2-rds.zip -r * .[^.]* -x "vendor/*"
   ```

2. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
3. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

   > **Note**
   >
   > 如果您有許多環境，請使用搜尋列來篩選環境清單。

4. 選擇 Upload and Deploy (上傳並部署)。
5. 選擇 Browse (瀏覽)，然後上傳 `cake-v2-rds.zip`。
6. 選擇部署。

部署應用程式的新版本不會超過一分鐘。部署完成時，請重新整理網頁，確認資料庫連線成功：



清除

當您完成使用 Elastic Beanstalk 時，即可終止您的環境。Elastic Beanstalk 會終止與您的環境相關聯的所有 AWS 資源，例如 Amazon EC2 執行個體、資料庫執行個體、負載平衡器、安全群組和警示。

從控制台終止 Elastic Beanstalk 環境

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 選擇 Actions (動作)，然後選擇 Terminate Environment (終止環境)。
4. 使用畫面顯示對話方塊來確認環境終止。

您可以使用 Elastic Beanstalk，隨時輕鬆地為您的應用程式建立新環境。

此外，您可終止於 Elastic Beanstalk 環境外建立的資料庫資源。終止 Amazon RDS 資料庫執行個體時，您可擷取快照，稍後再將資料還原至另一個執行個體。

終止 RDS 資料庫執行個體

1. 開啟 Amazon RDS 主控台。
2. 選擇 Databases (資料庫)。
3. 選擇資料庫執行個體。
4. 選擇動作，然後選擇刪除。
5. 選擇是否建立快照，然後選擇 Delete (刪除)。

後續步驟

如需 CakePHP 的詳細資訊，請造訪 book.cakephp.org 閱讀相關書目。

隨著您繼續開發應用程式，您可能會希望無須手動建立 .zip 檔案並將其上傳至 Elastic Beanstalk 主控台，即可管理環境和部署應用程式。E lastic Beanstalk 命令列介面 (EB CLI) 提供從命令列建立、設定和部署應用程式至 Elastic Beanstalk 環境的 easy-to-use 指令。

在 Elastic Beanstalk 環境中執行 Amazon RDS 資料庫執行個體對於開發和測試非常實用，但會將資料庫的生命週期連結至您的環境。如需連接至在環境外執行的資料庫的說明，請參閱 將 Amazon RDS 資料庫執行個體新增至您的 PHP 應用程式環境。

最後，若您打算於生產環境中使用您的應用程式，建議您設定您環境的自訂網域名稱，並啟用 HTTPS 安全連線。

將 Symfony 應用程式部署至 Elastic Beanstalk

Symfony 是一種開放原始碼架構，適用於開發動態 PHP Web 應用程式。本教學課程將逐步引導您完成產生 Symfony 應用程式並將其部署至環境的程 AWS Elastic Beanstalk 序。

章節

- 必要條件
- 啟動 Elastic Beanstalk 環境
- 安裝 Symfony 並產生網站
- 部署您的應用程式
- 設定 Composer 設定值
- 清除
- 後續步驟

必要條件

本教學假設您具備基本的 Elastic Beanstalk 操作及 Elastic Beanstalk 主控台知識。若您尚不了解，請依照 開始使用 Elastic Beanstalk 中的說明來啟動您的第一個 Elastic Beanstalk 環境。

為了遵循本指南的程序，您需要命令列終端機或 shell 來執行命令。命令清單前會出現提示字元 ($) 及目前的目錄名稱 (如有)。

```
~/eb-project$ this is a command
this is output
```

在 Linux 和 macOS 上，您可以使用偏好的 Shell 和套件軟體管理工具。在視窗上，你可以安裝視窗子系統為 Linux 得到一個視窗集成的 Ubuntu 和 Bash 的版本。

Symfony 4.4.9 需要 PHP 7.1.3 或更新版本。它還需要在官方 Symfony 安裝文件中的技術要求主題中列出的 PHP 擴充功能。在本教學中，我們使用 PHP 7.2 和對應的 Elastic Beanstalk 平台版本。按照主題 設定您的 PHP 開發環境 中的說明安裝 PHP 和 Composer。

有關 Symfony 支援和維護資訊，請參閱 Symfony 網站上的 symfony 發行主題。有關支援 Symfony 4.4.9 之 PHP 版本更新的詳細資訊，請參閱 Symfony 網站上 Symfony 4.4.9 版本說明主題。

啟動 Elastic Beanstalk 環境

使用 Elastic Beanstalk 主控台建立 Elastic Beanstalk 環境。選擇 PHP (PHP) 平台，並接受預設的設定和範本程式碼。

啟動環境 (主控台)

1. 使用這個預先配置的鏈接打開 Elastic Beanstalk 控制台：控制台。aws.amazon.com/彈性豆/家 #/新 applicationName = 教程和 environmentType LoadBalanced
2. 在 Platform (平台)，選取符合您應用程式所使用語言的平台和平台分支。
3. 針對 Application code (應用程式程式碼)，選擇 Sample application (範例應用程式)。
4. 選擇 Review and launch (檢閱和啟動)。
5. 檢視可用選項。選擇您要使用的可用選項，當您準備就緒時，請選擇 Creat app (建立應用程式)。

使用大約需要五分鐘時間建立環境，並且建立下列資源：

- EC2 執行個體 – Amazon Elastic Compute Cloud (Amazon EC2) 虛擬機器，已設為在您選擇的平台上執行 Web 應用程式。

  每個平台會執行特定的一套軟體、設定檔和指令碼，來支援特定的語言版本、架構、Web 容器或其組合。大多數的平台使用會 Apache 或 NGINX 做為反向代理，此反向代理會在您 Web 應用程式的前景執行、轉傳遞交給此 Web 應用程式的請求、提供靜態資產，並產生存取和錯誤日誌。
- 執行個體安全群組 - Amazon EC2 安全群組，已設為允許從連接埠 80 傳入的流量。此資源可讓負載平衡器傳來的 HTTP 傳輸資料，到達執行您 Web 應用程式的 EC2 執行個體。在預設情況下，不允許傳輸資料從其他通訊埠傳送。
- 負載平衡器 - Elastic Load Balancing 負載平衡器，可設定將請求分配到執行您應用程式的執行個體。負載平衡器也讓您的執行個體不需直接連接到網際網路。
- 負載平衡器安全群組 - Amazon EC2 安全群組，設為允許從連接埠 80 傳入的流量。此資源可讓來自網際網路的 HTTP 傳輸資料到達負載平衡器。在預設情況下，不允許傳輸資料從其他通訊埠傳送。
- Auto Scaling 群組 - Auto Scaling 群組，設為在執行個體終止或無法使用時，取代該執行個體。
- Amazon S3 儲存貯體 - 儲存位置，用來儲存當您使用 Elastic Beanstalk 時所建立的原始程式碼、日誌和其他成品。

- Amazon CloudWatch 警示 — 兩個 CloudWatch 警示，用於監控環境中執行個體的負載，並在負載過高或過低時觸發警示。當警示觸發時，您的 Auto Scaling 群組會擴展或縮減以進行回應。
- AWS CloudFormation 堆疊 — Elastic Beanstalk 用 AWS CloudFormation 來啟動環境中的資源並傳播組態變更。資源定義於範本中，您可在 AWS CloudFormation 主控台中檢視此範本。
- 網域名稱 – 會路由到您 Web 應用程式的網域名稱，其格式為 *subdomain*.*region*.elasticbeanstalk.com。

> **Note**
>
> 為了增強 Elastic Beanstalk 應用程式的安全性，我們會在公共后缀列表 (PSL) 中註冊網域 elasticbeanstalk.com。為了加強安全性，如果您需要在 Elastic Beanstalk 應用程式的預設網域名稱中設定敏感性 Cookie，我們建議您使用具 `__Host-` 前置詞的 Cookie。此做法將有助於保護您的網域免受跨站請求偽造 (CSRF) 攻擊。如需更多資訊，請參閱 Mozilla 開發人員網路中的設定 Cookie 頁面。

這些資源全都由 Elastic Beanstalk 管理。當您終止環境時，Elastic Beanstalk 會終止其中的所有資源。

> **Note**
>
> Elastic Beanstalk 建立的 Amazon S3 儲存貯體會在環境間共享，且不會在環境終止時刪除。如需詳細資訊，請參閱 將 Elastic Beanstalk 與 Amazon S3 搭配使用。

安裝 Symfony 並產生網站

Composer 可透過一個命令安裝 Symfony 並建立工作專案：

```
~$ composer create-project symfony/website-skeleton eb-symfony
```

Composer 會安裝 Symfony 及其依存項目，並產生預設專案。

若您在安裝 Symfony 時出現任何問題，請瀏覽官方 Symfony 文件的安裝主題。

部署您的應用程式

移至專案目錄。

```
~$ cd eb-symfony
```

建立原始碼套件，其中包含 Composer 所建立的檔案。以下命令建立一個名為 symfony-default.zip 的原始碼套件。它不含 vendor 資料夾中的檔案，其不僅佔用許多空間，在部署應用程式到 Elastic Beanstalk 時也派不上用場。

```
eb-symfony$ zip ../symfony-default.zip -r * .[^.]* -x "vendor/*"
```

將來源套件上傳至 Elastic Beanstalk，以將 Symfony 部署至您的環境。

若要部署原始碼套件

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

   > **Note**
   >
   > 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在環境概觀頁面上，選擇 Upload and deploy (上傳和部署)。
4. 使用畫面顯示對話方塊來上傳原始碼套件。
5. 選擇 Deploy (部署)。
6. 部署完成後，您可選擇網站的 URL 以在新分頁中開啟您的網站。

> **Note**
>
> 若要進一步最佳化原始碼套件，可將 Git 儲存庫初始化，並使用 `git archive` 命令建立原始碼套件。預設的 Symfony 專案包含了 `.gitignore` 檔案，它告訴 Git 排除 vendor 資料夾和其他不需用於部署的檔案。

設定 Composer 設定值

完成部署時按一下 URL，即可在瀏覽器中開啟 Symfony 應用程式。

這是什麼？ 根據預設，Elastic Beanstalk 會將專案根目錄做為網站的根路徑。然而，在此情況中，預設頁面 (app.php) 位於 web 資料夾的下一層。您可將 /public 新增至 URL，藉此加以驗證。例如 http://*symfony*.*us-east-2*.elasticbeanstalk.com/public。

若要在根路徑提供 Symfony 應用程式，請使用 Elastic Beanstalk 主控台來設定網站的「文件根」。

欲設定網站的文件根目錄

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇組態。
4. 在更新、監控和日誌記錄組態類別中，選擇編輯。
5. 在 Document Root (文件根目錄) 部分，輸入 **/public**。
6. 若要儲存變更，請選擇頁面底部的儲存變更。
7. 更新完成時按一下 URL，即可在瀏覽器中重新開啟您的網站。

清除

當您完成使用 Elastic Beanstalk 時，即可終止您的環境。Elastic Beanstalk 會終止與您的環境相關聯的所有 AWS 資源，例如 Amazon EC2 執行個體、資料庫執行個體、負載平衡器、安全群組和警示。

從控制台終止 Elastic Beanstalk 環境

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 選擇 Actions (動作)，然後選擇 Terminate Environment (終止環境)。
4. 使用畫面顯示對話方塊來確認環境終止。

您可以使用 Elastic Beanstalk，隨時輕鬆地為您的應用程式建立新環境。

後續步驟

如需 Symfony 的詳細資訊，請至 symfony.com 參閱什麼是 Symfony？一文。

隨著您繼續開發應用程式，您可能會希望無須手動建立 .zip 檔案並將其上傳至 Elastic Beanstalk 主控台，即可管理環境和部署應用程式。E lastic Beanstalk 命令列介面 (EB CLI) 提供從命令列建立、設定和部署應用程式至 Elastic Beanstalk 環境的 easy-to-use 指令。

在本教學中，您使用 Elastic Beanstalk 主控台來設定 Composer 選項。為了讓此組態成為應用程式原始碼的一部分，您的，您可以使用組態檔，如下所示。

Example .ebextensions/composer.config

```
option_settings:
  aws:elasticbeanstalk:container:php:phpini:
    document_root: /public
```

如需詳細資訊，請參閱 使用組態檔案 (.ebextensions) 來進行進階的環境自訂。

Symfony 使用自己的組態檔案設定資料庫連線。如需連接 Symfony 與資料庫的詳細資訊，請參閱 使用 Symfony 連接至資料庫。

最後，若您打算於生產環境中使用您的應用程式，建議您設定您環境的自訂網域名稱，並啟用 HTTPS 安全連線。

使用外部 Amazon RDS 資料庫將高可用性 PHP 應用程式資料庫部署至 Elastic Beanstalk

本教學將逐步引導您完成啟動外部 RDS 資料庫執行個體的程序 AWS Elastic Beanstalk，以及設定執行 PHP 應用程式以連線至該執行個體的高可用性環境。執行 Elastic Beanstalk 外的資料庫執行個體，會將該資料庫自您環境的生命週期解偶。如此一來，您即可自多個環境連接至相同的資料庫、更換資料庫，或執行藍/綠部署而不影響您的資料庫。

本教學課程透過範例 PHP 應用程式進行說明，該應用程式使用 MySQL 資料庫來存放使用者提供的文字資料。此範例應用程式使用組態檔案來進行 PHP 設定，並於資料庫建立供此應用程式使用的表格。此外，亦說明如何於部署期間使用 Composer 檔案來安裝套件。

章節

必要條件

在開始之前，請從以下位置下載範例應用程式來源套件 GitHub：eb-demo-php-simple-app- 1.3.zip

本 Amazon Relational Database Service (Amazon RDS) 任務教學中的程序假設您會在預設的 Amazon Virtual Private Cloud (Amazon VPC) 中啟動資源。所有新帳戶的各個區域都包含預設 VPC。若您沒有預設 VPC，則程序會有所不同。請參閱 搭配 Amazon RDS 使用 Elastic Beanstalk 以取得 EC2-Classic 和自訂 VPC 平台的相關說明。

在 Amazon RDS 中啟動資料庫執行個體

若要使用外部資料庫搭配執行於 Elastic Beanstalk 的應用程式，請先使用 Amazon RDS 啟動資料庫執行個體。當您使用 Amazon RDS 啟動執行個體時，它會完全獨立於 Elastic Beanstalk 和您的 Elastic Beanstalk 環境之外，而且不會受到 Elastic Beanstalk 終止或監控。

使用 Amazon RDS 主控台，啟動異地同步備份的 MySQL 資料庫執行個體。選擇異地同步備份部署可確保您的資料庫將會容錯遷移，並在主要資料庫執行個體停止服務時繼續運作。

欲在預設 VPC 中啟動 RDS 資料庫執行個體

1. 開啟 RDS 主控台。
2. 在導覽窗格中，選擇 Databases (資料庫)。
3. 選擇 Create database (建立資料庫)。
4. 選擇 Standard Create (標準建立)。

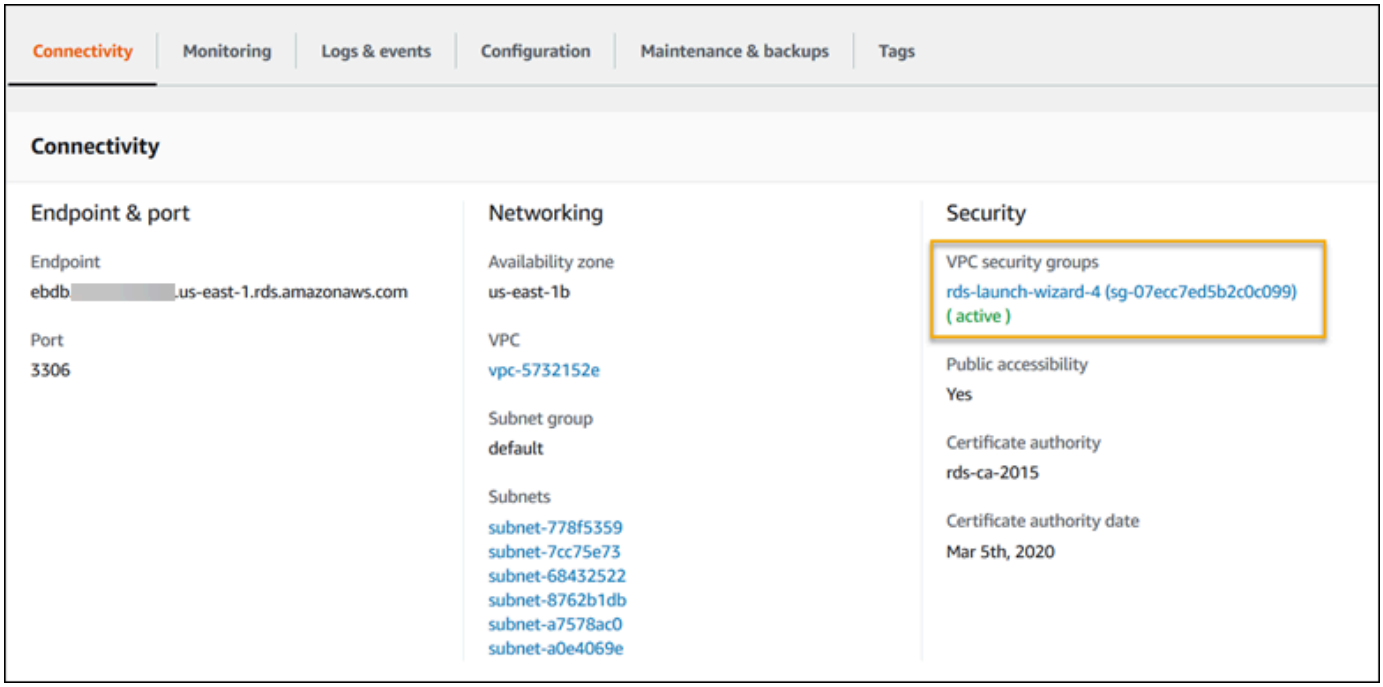
> **Important**
>
> 請勿選擇 Easy Create (輕鬆建立)。如果您選擇該選項，則您無法設定必要的設定來啟動此 RDS 資料庫。

5. 在 Additional configuration (其他設定) 下方的 Initial database name (初始資料庫名稱) 中輸入 **ebdb**。
6. 檢閱預設設定，並根據您的特定要求來調整這些設定。請注意以下選項：
   - 資料庫執行個體類別 – 選擇具有適當數量的記憶體和 CPU 功率之適合您工作負載的執行個體大小。
   - 異地同步備份部署 – 若要達到高可用性，請將此項設定為在不同的 AZ 中建立 Aurora 複本/讀取器節點。
   - 主要使用者名稱和主要密碼 – 資料庫使用者名稱和密碼。記下這些設定，以供稍後使用。
7. 檢查其餘選項的預設設定，然後選擇 Create database (建立資料庫)。

接著，修改連接至資料庫執行個體的安全群組，以允許適當連接埠的傳入流量。此安全群組與您稍後將連接至 Elastic Beanstalk 環境的相同，因此，您新增的規則將授予相同安全群組內其他資源的流量傳入許可。

修改連接至 RDS 執行個體的安全群組的傳入規則

1. 開啟 Amazon RDS 主控台。
2. 選擇 Databases (資料庫)。
3. 選擇您的資料庫執行個體名稱以檢視其詳細資料。
4. 在 Connectivity (連線) 區段中，記下顯示於此頁面的 Subnets (子網路)、Security groups (安全群組) 和 Endpoint (端點)。這樣您稍後便可使用這些資訊。
5. 在 Security (安全性) 下，可查看與資料庫執行個體相關聯的安全群組。開啟連結以檢視 Amazon EC2 主控台內的安全群組。



6. 在安全群組的詳細資訊中，選擇 Inbound (傳入)。
7. 選擇編輯。
8. 選擇 Add Rule (新增規則)。
9. 針對 Type (類型)，選擇您的應用程式所使用的資料庫引擎。
10. 對於 Source (來源)，輸入 **sg-** 檢視可用的安全群組清單。選擇與 Elastic Beanstalk 環境中使用之 Auto Scaling 群組相關聯的安全群組。以便環境中的 Amazon EC2 執行個體可以存取資料庫。

11. 選擇儲存。

建立資料庫執行個體約需要 10 分鐘。同時，建立您的 Elastic Beanstalk 環境。

建立 Elastic Beanstalk 環境

使用 Elastic Beanstalk 主控台建立 Elastic Beanstalk 環境。選擇 PHP (PHP) 平台，並接受預設的設定和範本程式碼。啟動環境之後，您可以將環境設定為連線至資料庫，然後部署從中下載的範例應用程式 GitHub。

啟動環境 (主控台)

1. 使用這個預先配置的鏈接打開 Elastic Beanstalk 控制台：控制台。aws.amazon.com/彈性豆/家 #/新 applicationName = 教程和 environmentType = LoadBalanced
2. 在 Platform (平台)，選取符合您應用程式所使用語言的平台和平台分支。
3. 針對 Application code (應用程式程式碼)，選擇 Sample application (範例應用程式)。
4. 選擇 Review and launch (檢閱和啟動)。
5. 檢視可用選項。選擇您要使用的可用選項，當您準備就緒時，請選擇 Creat app (建立應用程式)。

使用大約需要五分鐘時間建立環境，並且建立下列資源：

- EC2 執行個體 – Amazon Elastic Compute Cloud (Amazon EC2) 虛擬機器，已設為在您選擇的平台上執行 Web 應用程式。

  每個平台會執行特定的一套軟體、設定檔和指令碼，來支援特定的語言版本、架構、Web 容器或其組合。大多數的平台使用會 Apache 或 NGINX 做為反向代理，此反向代理會在您 Web 應用程式的前景執行、轉傳遞交給此 Web 應用程式的請求、提供靜態資產，並產生存取和錯誤日誌。
- 執行個體安全群組 - Amazon EC2 安全群組，已設為允許從連接埠 80 傳入的流量。此資源可讓負載平衡器傳來的 HTTP 傳輸資料，到達執行您 Web 應用程式的 EC2 執行個體。在預設情況下，不允許傳輸資料從其他通訊埠傳送。
- 負載平衡器 - Elastic Load Balancing 負載平衡器，可設定將請求分配到執行您應用程式的執行個體。負載平衡器也讓您的執行個體不需直接連接到網際網路。
- 負載平衡器安全群組 - Amazon EC2 安全群組，設為允許從連接埠 80 傳入的流量。此資源可讓來自網際網路的 HTTP 傳輸資料到達負載平衡器。在預設情況下，不允許傳輸資料從其他通訊埠傳送。
- Auto Scaling 群組 - Auto Scaling 群組，設為在執行個體終止或無法使用時，取代該執行個體。
- Amazon S3 儲存貯體 - 儲存位置，用來儲存當您使用 Elastic Beanstalk 時所建立的原始程式碼、日誌和其他成品。
- Amazon CloudWatch 警示 — 監控環境中執行個體負載的兩個 CloudWatch 警示，並在負載過高或過低時觸發警示。當警示觸發時，您的 Auto Scaling 群組會擴展或縮減以進行回應。

- AWS CloudFormation 堆疊 — Elastic Beanstalk 用 AWS CloudFormation 來啟動環境中的資源並傳播組態變更。資源定義於範本中，您可在 AWS CloudFormation 主控台中檢視此範本。
- 網域名稱 – 會路由到您 Web 應用程式的網域名稱，其格式為 *subdomain*.*region*.elasticbeanstalk.com。

> **Note**
>
> 為了增強 Elastic Beanstalk 應用程式的安全性，我們會在公共后缀列表 (PSL) 中註冊網域 elasticbeanstalk.com。為了加強安全性，如果您需要在 Elastic Beanstalk 應用程式的預設網域名稱中設定敏感性 Cookie，我們建議您使用具 `__Host-` 前置詞的 Cookie。此做法將有助於保護您的網域免受跨站請求偽造 (CSRF) 攻擊。如需更多資訊，請參閱 Mozilla 開發人員網路中的設定 Cookie 頁面。

這些資源全都由 Elastic Beanstalk 管理。當您終止環境時，Elastic Beanstalk 會終止其中的所有資源。您啟動的 RDS 資料庫執行個體在您的環境之外，因此必須負責管理其生命週期。

> **Note**
>
> Elastic Beanstalk 建立的 Amazon S3 儲存貯體會在環境間共享，且不會在環境終止時刪除。如需詳細資訊，請參閱 將 Elastic Beanstalk 與 Amazon S3 搭配使用。

設定安全群組、環境屬性和擴展

將資料庫執行個體的安全群組新增至執行環境。此程序會透過其他連接的安全群組，使 Elastic Beanstalk 重新佈建您環境中的所有執行個體。

欲將安全群組新增至您的環境

- 執行以下任意一項：
  - 使用 Elastic Beanstalk 主控台新增安全群組
    a. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
    b. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

c. 在導覽窗格中，選擇 Configuration (組態)。

d. 在 Instances (執行個體) 組態類別中，選擇 Edit (編輯)。

e. 在 EC2 安全群組下，除了 Elastic Beanstalk 建立的執行個體安全群組，請選擇要連接到執行個體的安全群組。

f. 若要儲存變更，請選擇頁面底部的儲存變更。

g. 閱讀警告的內容，然後選擇 Confirm (確認)。

- 若要使用組態檔案新增安全群組，請使用 `securitygroup-addexisting.config` 範例檔案。

接著，使用環境屬性，將連線資訊傳送至環境。此範例應用程式使用一組預設屬性，這些屬性符合您於環境佈建資料庫時 Elastic Beanstalk 所設定的屬性。

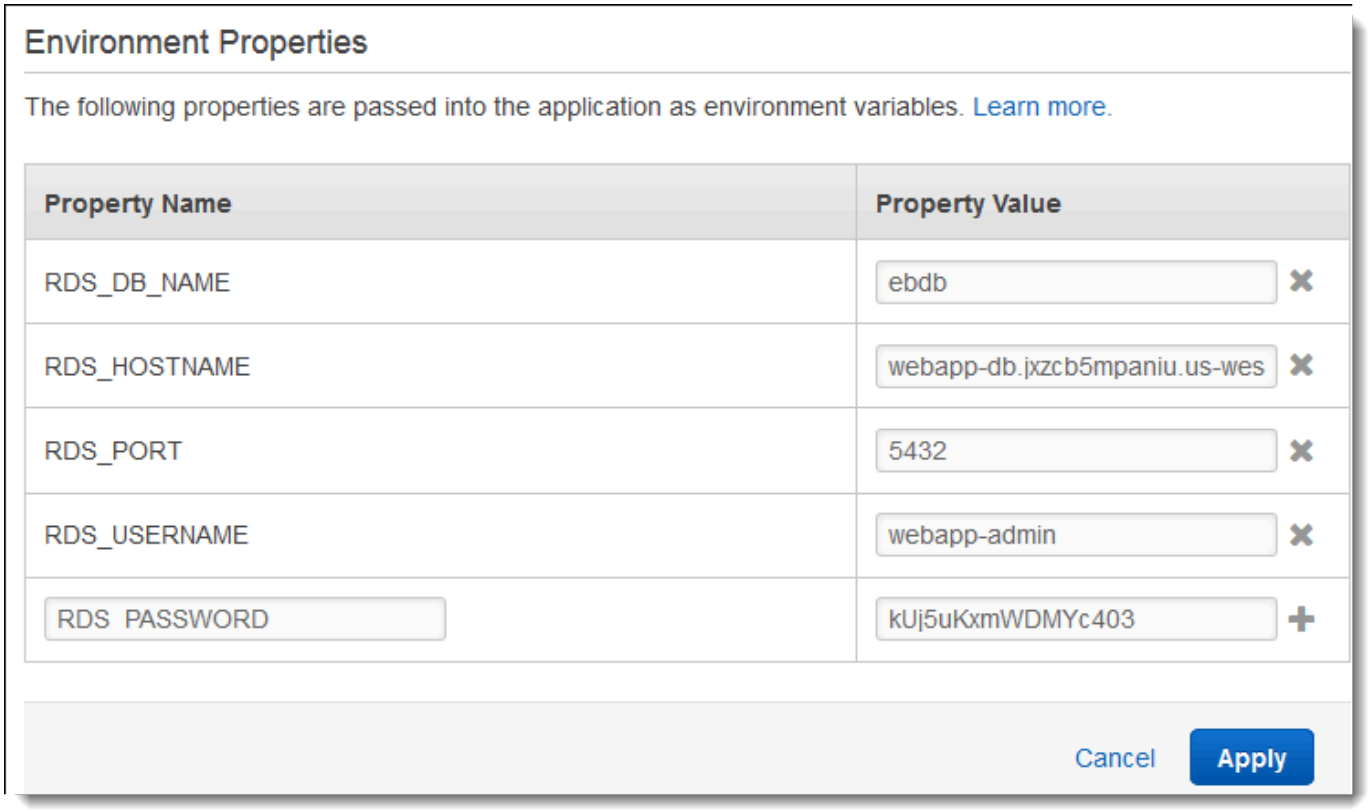
設定 Amazon RDS 資料庫執行個體的環境屬性

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇組態。
4. 在更新、監控和日誌記錄組態類別中，選擇編輯。
5. 在 Environment Properties (環境屬性) 區段，定義應用程式建立連線字串所讀取的變數。為了與具備整合式 RDS 資料庫執行個體的環境相容，請使用下列名稱與值。您可以在 RDS 主控台中找到除了密碼以外的所有值。

屬性名稱	描述	屬性值
RDS_HOSTNAME	資料庫執行個體的主機名稱。	在 Amazon RDS 主控台：端點的連線能力和安全性索引標籤上。
RDS_PORT	資料庫執行個體接受連線的連接埠。預設值在不同資料庫引擎中有所差異。	在 Amazon RDS 主控台：連接埠的連線能力和安全性索引標籤上。
RDS_DB_NAME	資料庫名稱，**ebdb**。	在 Amazon RDS 主控台：資料庫名稱的組態索引標籤上。
RDS_USERNAME	您為資料庫設定的使用者名稱。	在 Amazon RDS 主控台：主要使用者名稱的組態索引標籤上。
RDS_PASSWORD	您為資料庫設定的密碼。	無法在 Amazon RDS 主控台中提供參考。



6. 若要儲存變更，請選擇頁面底部的儲存變更。

最後，請增加執行個體計數下限，藉此設定您的環境 Auto Scaling 群組。隨時至少執行兩個執行個體，避免您環境中的 Web 伺服器出現單點故障，且無須停止網站服務即可部署變更。

若要設定您環境的 Auto Scaling 群組，以維持高可用性

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇 Configuration (組態)。
4. 在 Capacity (容量) 組態類別中，選擇 Edit (編輯)。
5. 在 Auto Scaling 群組區段，將最小執行個體設定為 **2**。

6. 若要儲存變更，請選擇頁面底部的儲存變更。

部署範例應用程式

現在，您的環境已就緒，可執行範例應用程式並連接至 Amazon RDS。將範例應用程式部署至您的環境。

> **Note**
>
> 如果您還沒有 GitHub，請從下載源代碼包：eb-demo-php-simple-app- 1.3.zip

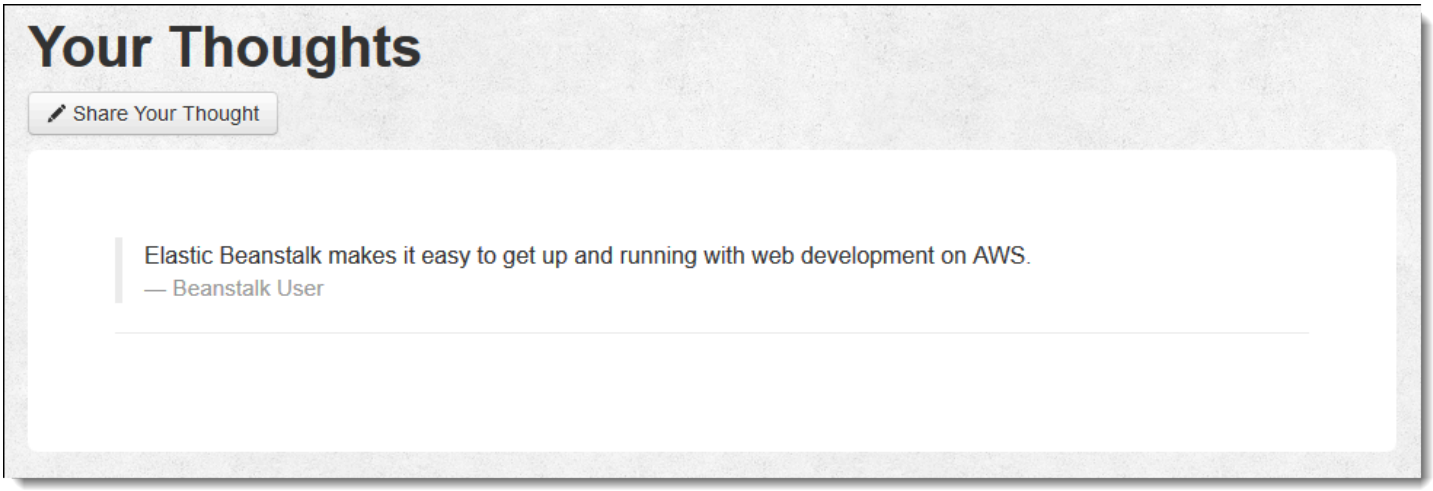
若要部署原始碼套件

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在環境概觀頁面上，選擇 Upload and deploy (上傳和部署)。
4. 使用畫面顯示對話方塊來上傳原始碼套件。
5. 選擇 Deploy (部署)。
6. 部署完成後，您可選擇網站的 URL 以在新分頁中開啟您的網站。

本站點會收集使用者評論，並使用 MySQL 資料庫來存放資料。若要新增評論，請選擇 Share Your Thought (分享您的想法)、輸入評論，然後選擇 Submit Your Thought (提交您的想法)。本 Web 應用程式會將評論寫入資料庫，環境中的任何執行個體均可讀取，而且執行個體停止服務時也不會遺失。



清除

當您完成使用 Elastic Beanstalk 時，即可終止您的環境。Elastic Beanstalk 會終止與您的環境相關聯的所有 AWS 資源，例如 Amazon EC2 執行個體、資料庫執行個體、負載平衡器、安全群組和警示。

從控制台終止 Elastic Beanstalk 環境

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 選擇 Actions (動作)，然後選擇 Terminate Environment (終止環境)。
4. 使用畫面顯示對話方塊來確認環境終止。

您可以使用 Elastic Beanstalk，隨時輕鬆地為您的應用程式建立新環境。

此外，您可終止於 Elastic Beanstalk 環境外建立的資料庫資源。終止 Amazon RDS 資料庫執行個體時，您可擷取快照，稍後再將資料還原至另一個執行個體。

終止 RDS 資料庫執行個體

1. 開啟 Amazon RDS 主控台。
2. 選擇 Databases (資料庫)。
3. 選擇資料庫執行個體。

4. 選擇動作，然後選擇刪除。
5. 選擇是否建立快照，然後選擇 Delete (刪除)。

後續步驟

隨著您繼續開發應用程式，您可能會希望無須手動建立 .zip 檔案並將其上傳至 Elastic Beanstalk 主控台，即可管理環境和部署應用程式。E lastic Beanstalk 命令列介面 (EB CLI) 提供從命令列建立、設定和部署應用程式至 Elastic Beanstalk 環境的 easy-to-use 指令。

此範例應用程式使用組態檔案來進行 PHP 設定，並於資料庫建立表格 (如尚未存在)。您亦可於環境建立期間，使用組態檔案進行執行個體的安全群組設定，以避免耗時的組態更新。如需詳細資訊，請參閱使用組態檔案 (.ebextensions) 來進行進階的環境自訂。

進行開發和測試時，建議您使用 Elastic Beanstalk 的功能，將受管的資料庫執行個體直接加入您的環境。如需於環境中設定資料庫的說明，請參閱將資料庫新增至您的 Elastic Beanstalk 環境。

若您需要高效能資料庫，請考慮使用 Amazon Aurora。Amazon Aurora 是一種與 MySQL 相容的資料庫引擎，能夠以低成本提供商用資料庫的功能。欲將您的應用程式連接至不同資料庫，請重複安全群組設定步驟，並更新 RDS 相關的環境屬性。

最後，若您打算於生產環境中使用您的應用程式，建議您設定您環境的自訂網域名稱，並啟用 HTTPS 安全連線。

將具有外部 Amazon RDS 資料庫的高可用性 WordPress 網站部署到 Elastic Beanstalk

本教學說明如何啟動外部的 Amazon RDS 資料庫執行個體 AWS Elastic Beanstalk，以及如何設定執行 WordPress 網站連線的高可用性環境。此網站使用 Amazon Elastic File System (Amazon EFS) 做為共用的儲存裝置以放置上傳的檔案。

執行 Elastic Beanstalk 外的資料庫執行個體，會將該資料庫自您環境的生命週期解偶。如此一來，您即可自多個環境連接至相同的資料庫、更換資料庫，或執行藍/綠部署而不影響您的資料庫。

> **Note**
>
> 有關 PHP WordPress 版本與版本兼容性的最新信息，請參閱 WordPress 網站上的 PHP 兼容性和 WordPress 版本。在升級到 PHP 的新版本以進行 WordPress 實現之前，您應該參考此信息。

主題

必要條件

本教學假設您具備基本的 Elastic Beanstalk 操作及 Elastic Beanstalk 主控台知識。若您尚不了解，請依照 開始使用 Elastic Beanstalk 中的說明來啟動您的第一個 Elastic Beanstalk 環境。

為了遵循本指南的程序，您需要命令列終端機或 shell 來執行命令。命令清單前會出現提示字元 ($) 及目前的目錄名稱 (如有)。

```
~/eb-project$ this is a command
this is output
```

在 Linux 和 macOS 上，您可以使用偏好的 Shell 和套件軟體管理工具。在視窗上，你可以安裝視窗子系統為 Linux 得到一個視窗集成的 Ubuntu 和 Bash 的版本。

預設 VPC

本教學中的 Amazon Relational Database Service (Amazon RDS) 程序假設您會在預設的 Amazon Virtual Private Cloud (Amazon VPC) 中啟動資源。所有新帳戶在每個 AWS 區域中都包含一個預設 VPC。若您沒有預設 VPC，則程序會有所不同。請參閱 搭配 Amazon RDS 使用 Elastic Beanstalk 以取得 EC2-Classic 和自訂 VPC 平台的相關說明。

AWS 地區

此範例應用程式使用 Amazon EFS，該應用程式僅適用於支援 Amazon EFS 的 AWS 區域。若要進一步了解支援 Amazon Elastic File System AWS 區域，請參閱 AWS 一般參考.

在 Amazon RDS 中啟動資料庫執行個體

當您使用 Amazon RDS 啟動執行個體時，它會完全獨立於 Elastic Beanstalk 和您的 Elastic Beanstalk 環境之外，而且不會受到 Elastic Beanstalk 終止或監控。

在下列步驟中，您將使用 Amazon RDS 主控台：

- 啟動使用 MySQL 引擎的資料庫。
- 啟用 Multi-AZ deployment (異地同步備份部署)。這會在不同的可用區域 (AZ) 中建立待命狀態，以提供資料備援、消除 I/O 凍結，並將系統備份期間的延遲峰值降至最低。

欲在預設 VPC 中啟動 RDS 資料庫執行個體

1. 開啟 RDS 主控台。
2. 在導覽窗格中，選擇 Databases (資料庫)。
3. 選擇 Create database (建立資料庫)。
4. 選擇 Standard Create (標準建立)。

> **Important**
>
> 請勿選擇 Easy Create (輕鬆建立)。如果您選擇該選項，則您無法設定必要的設定來啟動此 RDS 資料庫。

5. 在 Additional configuration (其他設定) 下方的 Initial database name (初始資料庫名稱) 中輸入 **ebdb**。
6. 檢閱預設設定，並根據您的特定要求來調整這些設定。請注意以下選項：
   - 資料庫執行個體類別 – 選擇具有適當數量的記憶體和 CPU 功率之適合您工作負載的執行個體大小。
   - 異地同步備份部署 – 若要達到高可用性，請將此項設定為在不同的 AZ 中建立 Aurora 複本/讀取器節點。
   - 主要使用者名稱和主要密碼 – 資料庫使用者名稱和密碼。記下這些設定，以供稍後使用。

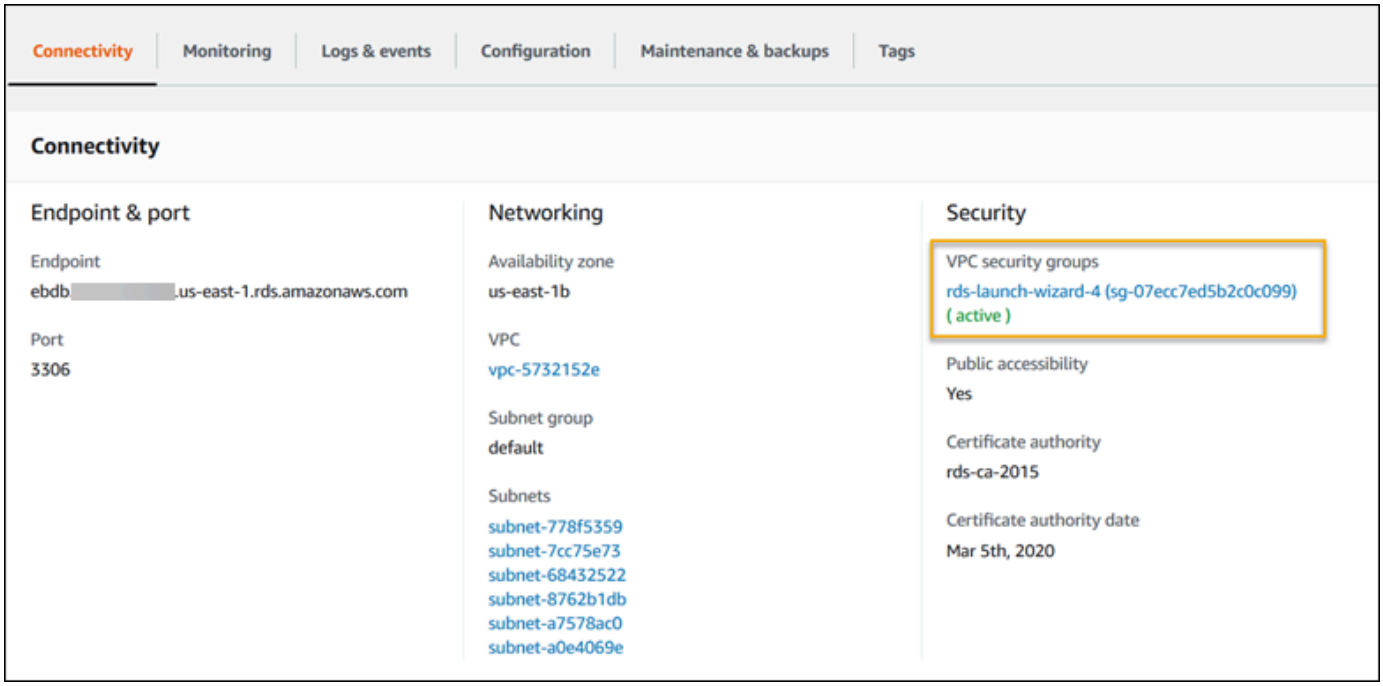
7. 檢查其餘選項的預設設定，然後選擇 Create database (建立資料庫)。

建立您的資料庫執行個體之後，請修改連接至該執行個體的安全群組，以允許適當連接埠的傳入流量。

> **Note**
>
> 此安全群組與您稍後將連接至 Elastic Beanstalk 環境的相同，因此，您新增的規則現在將授予相同安全群組內其他資源的流量傳入許可。

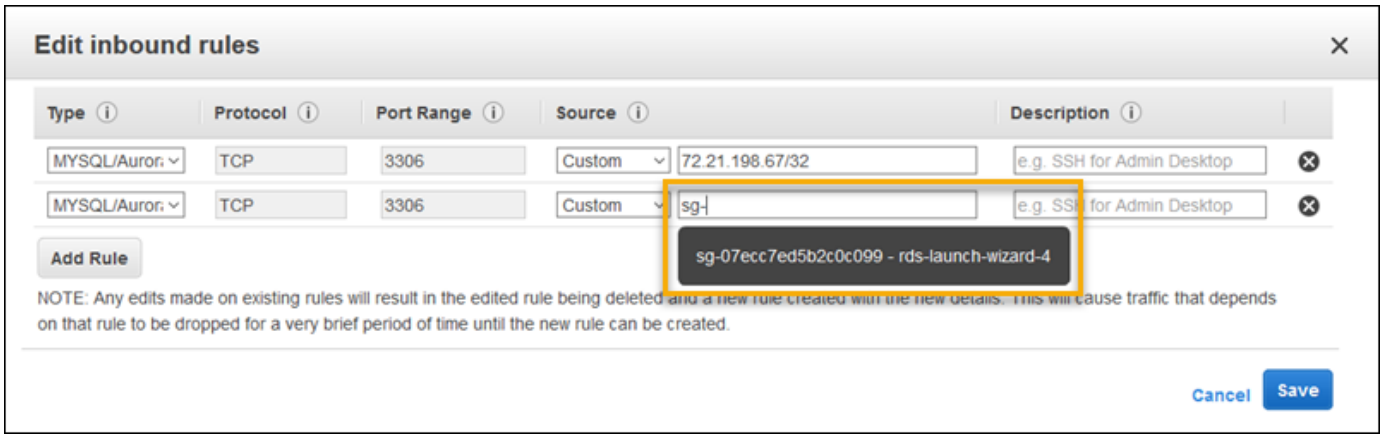
修改連接至 RDS 執行個體的安全群組的傳入規則

1. 開啟 Amazon RDS 主控台。
2. 選擇 Databases (資料庫)。
3. 選擇您的資料庫執行個體名稱以檢視其詳細資料。
4. 在 Connectivity (連線) 區段中，記下顯示於此頁面的 Subnets (子網路)、Security groups (安全群組) 和 Endpoint (端點)。這樣您稍後便可使用這些資訊。
5. 在 Security (安全性) 下，可查看與資料庫執行個體相關聯的安全群組。開啟連結以檢視 Amazon EC2 主控台內的安全群組。



6. 在安全群組的詳細資訊中，選擇 Inbound (傳入)。
7. 選擇編輯。

8. 選擇 Add Rule (新增規則)。
9. 針對 Type (類型)，選擇您的應用程式所使用的資料庫引擎。
10. 對於 Source (來源)，輸入 **sg-** 檢視可用的安全群組清單。選擇與 Elastic Beanstalk 環境中使用之 Auto Scaling 群組相關聯的安全群組。以便環境中的 Amazon EC2 執行個體可以存取資料庫。



11. 選擇儲存。

建立資料庫執行個體約需要 10 分鐘。同時，下載 WordPress 並創建您的 Elastic Beanstalk 環境。

下載 WordPress

若要準備 WordPress 使用進行部署 AWS Elastic Beanstalk，您必須將 WordPress 檔案複製到您的電腦，並提供正確的組態資訊。

建立 WordPress 專案的步驟

1. WordPress 從下載.

```
~$curl https://wordpress.org/wordpress-6.2.tar.gz -o wordpress.tar.gz
```

2. 從範本儲存庫下載組態檔案。

```
~$ wget https://github.com/aws-samples/eb-php-wordpress/releases/download/v1.1/eb-
php-wordpress-v1.zip
```

3. 解壓縮 WordPress 並變更資料夾的名稱。

```
~$ tar -xvf wordpress.tar.gz
~$ mv wordpress wordpress-beanstalk
~$ cd wordpress-beanstalk
```

4. 在 WordPress 安裝過程中解壓縮配置檔案。

```
~/wordpress-beanstalk$ unzip ../eb-php-wordpress-v1.zip
 creating: .ebextensions/
inflating: .ebextensions/dev.config
inflating: .ebextensions/efs-create.config
inflating: .ebextensions/efs-mount.config
inflating: .ebextensions/loadbalancer-sg.config
inflating: .ebextensions/wordpress.config
inflating: LICENSE
inflating: README.md
inflating: wp-config.php
```

啟動 Elastic Beanstalk 環境

使用 Elastic Beanstalk 主控台建立 Elastic Beanstalk 環境。啟動環境之後，您可以將其設定為連線至資料庫，然後將 WordPress 程式碼部署至環境。

在下列步驟中，您將使用 Elastic Beanstalk 主控台：

- 使用受管 PHP 平台建立 Elastic Beanstalk 應用程式。
- 接受預設設定和範本程式碼。

啟動環境 (主控台)

1. 使用這個預先配置的鏈接打開 Elastic Beanstalk 控制台：控制台。aws.amazon.com/彈性豆/家 # /新 applicationName = 教程和 environmentType = LoadBalanced
2. 在 Platform (平台)，選取符合您應用程式所使用語言的平台和平台分支。
3. 針對 Application code (應用程式程式碼)，選擇 Sample application (範例應用程式)。
4. 選擇 Review and launch (檢閱和啟動)。
5. 檢視可用選項。選擇您要使用的可用選項，當您準備就緒時，請選擇 Creat app (建立應用程式)。

大約需要五分鐘時間建立環境，並建立下列資源。

Elastic Beanstalk 建立的資源

- EC2 執行個體 – Amazon Elastic Compute Cloud (Amazon EC2) 虛擬機器，已設為在您選擇的平台上執行 Web 應用程式。

每個平台會執行特定的一套軟體、設定檔和指令碼，來支援特定的語言版本、架構、Web 容器或其組合。大多數的平台使用會 Apache 或 NGINX 做為反向代理，此反向代理會在您 Web 應用程式的前景執行、轉傳遞交給此 Web 應用程式的請求、提供靜態資產，並產生存取和錯誤日誌。

- 執行個體安全群組 - Amazon EC2 安全群組，已設為允許從連接埠 80 傳入的流量。此資源可讓負載平衡器傳來的 HTTP 傳輸資料，到達執行您 Web 應用程式的 EC2 執行個體。在預設情況下，不允許傳輸資料從其他通訊埠傳送。
- 負載平衡器 - Elastic Load Balancing 負載平衡器，可設定將請求分配到執行您應用程式的執行個體。負載平衡器也讓您的執行個體不需直接連接到網際網路。
- 負載平衡器安全群組 - Amazon EC2 安全群組，設為允許從連接埠 80 傳入的流量。此資源可讓來自網際網路的 HTTP 傳輸資料到達負載平衡器。在預設情況下，不允許傳輸資料從其他通訊埠傳送。
- Auto Scaling 群組 - Auto Scaling 群組，設為在執行個體終止或無法使用時，取代該執行個體。
- Amazon S3 儲存貯體 - 儲存位置，用來儲存當您使用 Elastic Beanstalk 時所建立的原始程式碼、日誌和其他成品。
- Amazon CloudWatch 警示 — 監控環境中執行個體負載的兩個 CloudWatch 警示，並在負載過高或過低時觸發警示。當警示觸發時，您的 Auto Scaling 群組會擴展或縮減以進行回應。
- AWS CloudFormation 堆疊 — Elastic Beanstalk 用 AWS CloudFormation 來啟動環境中的資源並傳播組態變更。資源定義於範本中，您可在 AWS CloudFormation 主控台中檢視此範本。
- 網域名稱 – 會路由到您 Web 應用程式的網域名稱，其格式為 *subdomain*.*region*.elasticbeanstalk.com。

> **Note**
>
> 為了增強 Elastic Beanstalk 應用程式的安全性，我們會在公共后缀列表 (PSL) 中註冊網域 elasticbeanstalk.com。為了加強安全性，如果您需要在 Elastic Beanstalk 應用程式的預設網域名稱中設定敏感性 Cookie，我們建議您使用具 `__Host-` 前置詞的 Cookie。此做法將有助於保護您的網域免受跨站請求偽造 (CSRF) 攻擊。如需更多資訊，請參閱 Mozilla 開發人員網路中的設定 Cookie 頁面。

這些資源全都由 Elastic Beanstalk 管理。當您終止環境時，Elastic Beanstalk 會終止其中的所有資源。

您啟動的 Amazon RDS 執行個體在您的環境之外，因此您必須負責管理其生命週期。

> **Note**
>
> Elastic Beanstalk 建立的 Amazon S3 儲存貯體會在環境間共享，且不會在環境終止時刪除。如需詳細資訊，請參閱 將 Elastic Beanstalk 與 Amazon S3 搭配使用。

設定安全群組和環境屬性

將資料庫執行個體的安全群組新增至執行環境。此程序會透過其他連接的安全群組，使 Elastic Beanstalk 重新佈建您環境中的所有執行個體。

欲將安全群組新增至您的環境

- 執行以下任意一項：
  - 使用 Elastic Beanstalk 主控台新增安全群組
    a. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
    b. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

    > **Note**
    >
    > 如果您有許多環境，請使用搜尋列來篩選環境清單。

    c. 在導覽窗格中，選擇 Configuration (組態)。
    d. 在 Instances (執行個體) 組態類別中，選擇 Edit (編輯)。
    e. 在 EC2 安全群組下，除了 Elastic Beanstalk 建立的執行個體安全群組，請選擇要連接到執行個體的安全群組。
    f. 若要儲存變更，請選擇頁面底部的儲存變更。
    g. 閱讀警告的內容，然後選擇 Confirm (確認)。
  - 若要使用組態檔案新增安全群組，請使用 `securitygroup-addexisting.config` 範例檔案。

接著，使用環境屬性，將連線資訊傳送至環境。

WordPress 應用程式會使用一組預設屬性，這些屬性與您在環境中佈建資料庫時 Elastic Beanstalk 所設定的屬性相符。

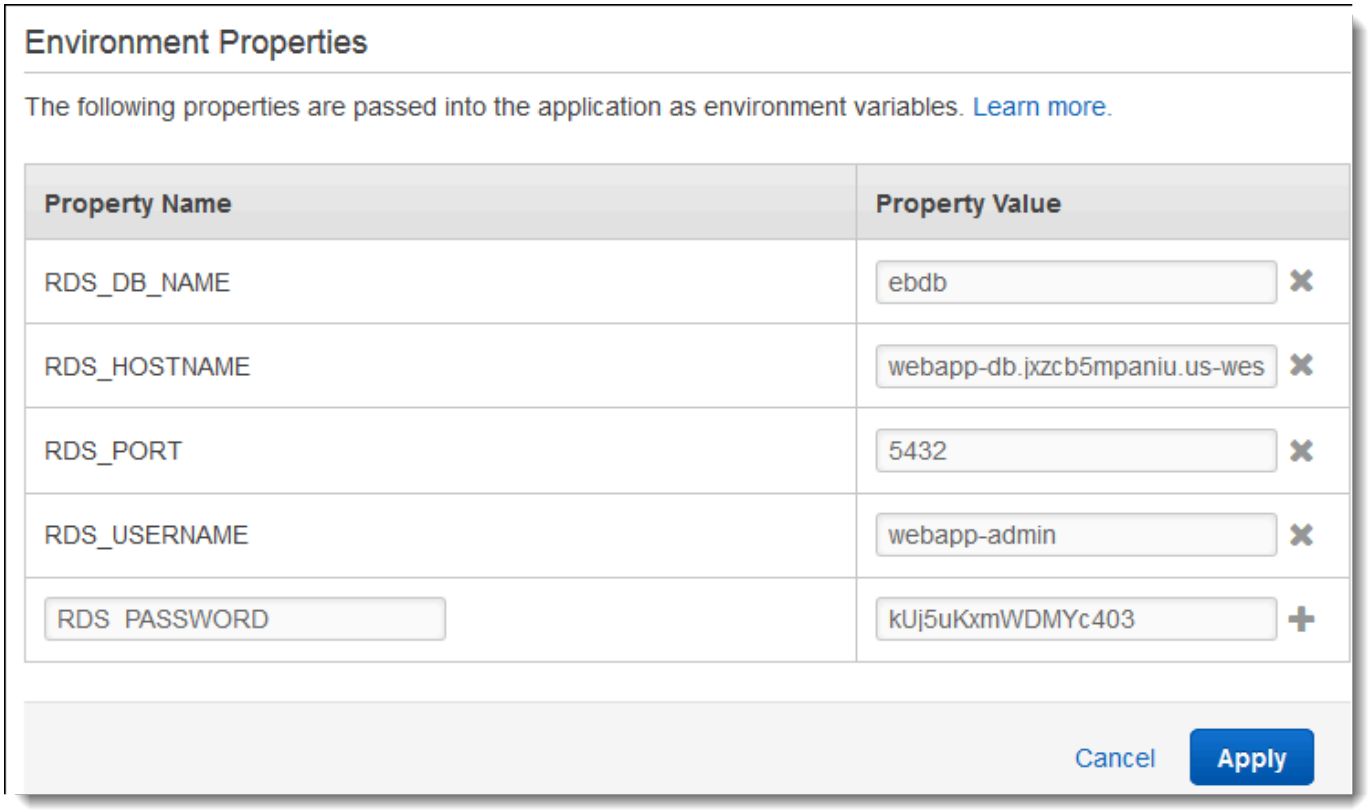
設定 Amazon RDS 資料庫執行個體的環境屬性

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇組態。
4. 在更新、監控和日誌記錄組態類別中，選擇編輯。
5. 在 Environment Properties (環境屬性) 區段，定義應用程式建立連線字串所讀取的變數。為了與具備整合式 RDS 資料庫執行個體的環境相容，請使用下列名稱與值。您可以在 RDS 主控台中找到除了密碼以外的所有值。

屬性名稱	描述	屬性值
RDS_HOSTNAME	資料庫執行個體的主機名稱。	在 Amazon RDS 主控台：端點的連線能力和安全性索引標籤上。
RDS_PORT	資料庫執行個體接受連線的連接埠。預設值在不同資料庫引擎中有所差異。	在 Amazon RDS 主控台：連接埠的連線能力和安全性索引標籤上。
RDS_DB_NAME	資料庫名稱，**ebdb**。	在 Amazon RDS 主控台：資料庫名稱的組態索引標籤上。
RDS_USERNAME	您為資料庫設定的使用者名稱。	在 Amazon RDS 主控台：主要使用者名稱的組態索引標籤上。
RDS_PASSWORD	您為資料庫設定的密碼。	無法在 Amazon RDS 主控台中提供參考。



6. 若要儲存變更，請選擇頁面底部的儲存變更。

設定並部署您的應用程式

驗證您的 `wordpress-beanstalk` 資料夾結構是否正確，如下所示。

```
wordpress-beanstalk$ tree -aL 1
.
### .ebextensions
### index.php
### LICENSE
### license.txt
### readme.html
### README.md
### wp-activate.php
### wp-admin
### wp-blog-header.php
### wp-comments-post.php
### wp-config.php
### wp-config-sample.php
```

```
### wp-content
### wp-cron.php
### wp-includes
### wp-links-opml.php
### wp-load.php
### wp-login.php
### wp-mail.php
### wp-settings.php
### wp-signup.php
### wp-trackback.php
### xmlrpc.php
```

來自專案儲存庫的自訂 `wp-config.php` 檔案使用您在前一步驟中定義的環境變數來設定資料庫連線。`.ebextensions` 資料夾包含在 Elastic Beanstalk 環境中建立其他資源的組態檔案。

需修改組態檔案才可適用於您的帳戶。以適當的 ID 來更換檔案中的預留位置值，並建立原始碼套件。

更新組態檔案並建立原始碼套件

1. 如下修改組態檔案。

   - `.ebextensions/dev.config`— 在 WordPress 安裝過程中限制對環境的存取以保護環境。將靠近檔案頂端的預留位置 IP 位址取代為您將用來存取環境網站以完成 WordPress 安裝的電腦的公用 IP 位址。

     > **Note**
     >
     > 根據您的網路而定，您可能需要使用 IP 地址區塊。

   - `.ebextensions/efs-create.config` – 會在您 VPC 的各個可用區域/子網路中建立 EFS 檔案系統和掛載點。在 Amazon VPC 主控台中辨識您的預設 VPC 主控台和子網路 ID。

2. 建立原始碼套件，其中包含專案資料夾中的檔案。以下命令建立一個名為 `wordpress-beanstalk.zip` 的原始碼套件。

   ```
   ~/eb-wordpress$ zip ../wordpress-beanstalk.zip -r * .[^.]*
   ```

將來源服務包上傳到 Elastic Beanstalk，以部署 WordPress 到您的環境。

若要部署原始碼套件

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在環境概觀頁面上，選擇 Upload and deploy (上傳和部署)。
4. 使用畫面顯示對話方塊來上傳原始碼套件。
5. 選擇 Deploy (部署)。
6. 部署完成後，您可選擇網站的 URL 以在新分頁中開啟您的網站。

安裝 WordPress

若要完成您的 WordPress 安裝

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在瀏覽器中選擇環境 URL 來開啟您的網站。系統會將您重新導向至 WordPress 安裝精靈，因為您尚未設定網站。
4. 執行標準安裝。`wp-config.php` 檔案已存在於原始碼中，且會設定為自環境讀取資料庫連線資訊。應該不會提示您設定連線。

安裝約需要 1 分鐘來完成。

更新金鑰和鹽

WordPress 組態檔案`wp-config.php`也會從環境屬性中讀取關鍵字和鹽的值。這些屬性目前均由 `test` 資料夾的 `wordpress.config` 檔案設定為 `.ebextensions`。

雜湊鹽可以是任何符合環境屬性要求的值，但是不應存放於原始控制碼中。使用 Elastic Beanstalk 主控台直接於環境設定這些屬性。

若要更新環境屬性

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇 Configuration (組態)。
4. 在 Software (軟體) 下，選擇 Edit (編輯)。
5. 對於 `Environment properties`，修改以下屬性：
   - `AUTH_KEY` – 針對 `AUTH_KEY` 所選的值。
   - `SECURE_AUTH_KEY` – 針對 `SECURE_AUTH_KEY` 所選的值。
   - `LOGGED_IN_KEY` – 針對 `LOGGED_IN_KEY` 所選的值。
   - `NONCE_KEY` – 針對 `NONCE_KEY` 所選的值。
   - `AUTH_SALT` – 針對 `AUTH_SALT` 所選的值。
   - `SECURE_AUTH_SALT` – 針對 `SECURE_AUTH_SALT` 所選的值。
   - `LOGGED_IN_SALT` – 針對 `LOGGED_IN_SALT` 所選的值。
   - `NONCE_SALT` – 針對 `NONCE_SALT` 所選的值。
6. 若要儲存變更，請選擇頁面底部的儲存變更。

> **Note**
>
> 於環境直接設定的屬性會覆寫 `wordpress.config` 中的值。

移除存取限制

範例專案包含組態檔案 `loadbalancer-sg.config`。它會使用您在 `dev.config` 中設定的 IP 地址，建立安全群組並將其指派至環境的負載平衡器。它會將連接埠 80 上的 HTTP 存取限制為來自您網路的連線。否則，在您安裝 WordPress 和設定管理員帳戶之前，外部方可能會連線到您的網站。

現在您已經安裝了 WordPress，請刪除配置文件以向全世界打開網站。

欲移除限制並更新您的環境

1. 從專案目錄中刪除 `.ebextensions/loadbalancer-sg.config` 檔案。

```
~/wordpress-beanstalk$ rm .ebextensions/loadbalancer-sg.config
```

2. 建立原始碼套件。

```
~/eb-wordpress$ zip ../wordpress-beanstalk-v2.zip -r * .[^.]*
```

將來源服務包上傳到 Elastic Beanstalk，以部署 WordPress 到您的環境。

若要部署原始碼套件

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在環境概觀頁面上，選擇 Upload and deploy (上傳和部署)。
4. 使用畫面顯示對話方塊來上傳原始碼套件。
5. 選擇 Deploy (部署)。
6. 部署完成後，您可選擇網站的 URL 以在新分頁中開啟您的網站。

設定 Auto Scaling 群組

最後，請增加執行個體計數下限，藉此設定您的環境 Auto Scaling 群組。隨時執行至少兩個執行個體，以防止您環境中的 web 伺服器出現單一故障點。這也可讓您在不停用網站的情況下部署變更。

若要設定您環境的 Auto Scaling 群組，以維持高可用性

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇 Configuration (組態)。
4. 在 Capacity (容量) 組態類別中，選擇 Edit (編輯)。
5. 在 Auto Scaling 群組區段，將最小執行個體設定為 **2**。
6. 若要儲存變更，請選擇頁面底部的儲存變更。

若要支援跨多個執行個體上傳內容，範例專案會使用 Amazon EFS 來建立共用檔案系統。在網站上建立貼文並上傳內容以將內容存放在共用檔案系統上。查看貼文並重新整理頁面數次來點擊兩個執行個體並確認共用檔案系統正在執行中。

升級 WordPress

若要升級到新版本的 WordPress，請備份您的網站並將其部署到新環境。

> **Important**
>
> 請勿在中使用更新功能， WordPress 或更新來源檔案以使用新版本。這兩個動作都會導致您的貼文 URL 傳回 404 錯誤，即使他們仍在資料庫與檔案系統中。

若要升級 WordPress

1. 在 WordPress 管理控制台中，使用匯出工具將貼文匯出至 XML 檔案。
2. 使用與安裝先前版本相同 WordPress 的步驟將新版本部署並安裝到 Elastic Beanstalk。為了避免停機時間，您可以使用新版本建立環境。
3. 在新版本上，在管理控制台中安裝導入 WordPress 器工具，並使用它來導入包含您帖子的 XML 文件。如果貼文是由管理員使用者在舊版本上建立的，請在新網站上指定他們給管理員使用者，而非嘗試匯入管理員使用者。
4. 若您部署了新版本到個別的環境，請進行 CNAME 調換來從舊網站重新引導使用者到新網站。

清除

當您完成使用 Elastic Beanstalk 時，即可終止您的環境。Elastic Beanstalk 會終止與您的環境相關聯的所有 AWS 資源，例如 Amazon EC2 執行個體、資料庫執行個體、負載平衡器、安全群組和警示。

從控制台終止 Elastic Beanstalk 環境

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 選擇 Actions (動作)，然後選擇 Terminate Environment (終止環境)。
4. 使用畫面顯示對話方塊來確認環境終止。

您可以使用 Elastic Beanstalk，隨時輕鬆地為您的應用程式建立新環境。

此外，您可終止於 Elastic Beanstalk 環境外建立的資料庫資源。終止 Amazon RDS 資料庫執行個體時，您可擷取快照，稍後再將資料還原至另一個執行個體。

終止 RDS 資料庫執行個體

1. 開啟 Amazon RDS 主控台。
2. 選擇 Databases (資料庫)。
3. 選擇資料庫執行個體。
4. 選擇動作，然後選擇刪除。
5. 選擇是否建立快照，然後選擇 Delete (刪除)。

後續步驟

隨著您繼續開發應用程式，您可能會希望無須手動建立 .zip 檔案並將其上傳至 Elastic Beanstalk 主控台，即可管理環境和部署應用程式。E lastic Beanstalk 命令列介面 (EB CLI) 提供從命令列建立、設定和部署應用程式至 Elastic Beanstalk 環境的 easy-to-use 指令。

此範例應用程式使用組態檔案來進行 PHP 設定，並於資料庫建立表格 (如尚未存在)。您亦可於環境建立期間，使用組態檔案進行執行個體的安全群組設定，以避免耗時的組態更新。如需詳細資訊，請參閱使用組態檔案 (.ebextensions) 來進行進階的環境自訂。

進行開發和測試時，建議您使用 Elastic Beanstalk 的功能，將受管的資料庫執行個體直接加入您的環境。如需於環境中設定資料庫的說明，請參閱將資料庫新增至您的 Elastic Beanstalk 環境。

若您需要高效能資料庫，請考慮使用 Amazon Aurora。Amazon Aurora 是一種與 MySQL 相容的資料庫引擎，能夠以低成本提供商用資料庫的功能。欲將您的應用程式連接至不同資料庫，請重複安全群組設定步驟，並更新 RDS 相關的環境屬性。

最後，若您打算於生產環境中使用您的應用程式，建議您設定您環境的自訂網域名稱，並啟用 HTTPS 安全連線。

使用外部 Amazon RDS 資料庫將高可用性 Drupal 網站資料庫部署至 Elastic Beanstalk

本教學課程將逐步引導您啟動外部 RDS 資料庫執行個體的程序 AWS Elastic Beanstalk。然後說明任何設定執行 Drupal 網站的高可用性環境來與執行個體連接。此網站使用 Amazon Elastic File System (Amazon EFS) 做為共用的儲存裝置以放置上傳的檔案。執行 Elastic Beanstalk 外的資料庫執行個體，會將該資料庫自您環境的生命週期分離，並讓您可自多個環境連接至相同的資料庫、更換資料庫，或執行藍/綠部署而不影響您的資料庫。

章節

- 必要條件
- 在 Amazon RDS 中啟動資料庫執行個體
- 啟動 Elastic Beanstalk 環境
- 設定安全設定和環境屬性
- 設定並部署您的應用程式
- 安裝 Drupal
- 更新 Drupal 組態和移除存取限制
- 設定 Auto Scaling 群組
- 清除
- 後續步驟

必要條件

本教學假設您具備基本的 Elastic Beanstalk 操作及 Elastic Beanstalk 主控台知識。若您尚不了解，請依照 開始使用 Elastic Beanstalk 中的說明來啟動您的第一個 Elastic Beanstalk 環境。

為了遵循本指南的程序，您需要命令列終端機或 shell 來執行命令。命令清單前會出現提示字元 ($) 及目前的目錄名稱 (如有)。

```
~/eb-project$ this is a command
this is output
```

在 Linux 和 macOS 上，您可以使用偏好的 Shell 和套件軟體管理工具。在視窗上，你可以安裝視窗子系統為 Linux 得到一個視窗集成的 Ubuntu 和 Bash 的版本。

本 Amazon Relational Database Service (Amazon RDS) 任務教學中的程序假設您會在預設的 Amazon Virtual Private Cloud (Amazon VPC) 中啟動資源。所有新帳戶的各個區域都包含預設 VPC。若您沒有預設 VPC，則程序會有所不同。請參閱 搭配 Amazon RDS 使用 Elastic Beanstalk 以取得 EC2-Classic 和自訂 VPC 平台的相關說明。

此範例應用程式使用 Amazon EFS。它僅適用於支援 Amazon EFS 的 AWS 區域。若要進一步了解支援 AWS 區域的 Amazon Elastic File System 訊，請參閱 AWS 一般參考.

如果您的 Elastic Beanstalk 環境的平台使用 PHP 7.4 或更早版本，我們建議您使用 Drupal 8.9.13 版的本教學課程。對於安裝 PHP 8.0 或更新版本的平台，我們建議您使用 Drupal 9.1.5。

有關 Drupal 版本和其支援之 PHP 版本的詳細資訊，請參閱 Drupal 網站上的 PHP 要求。Drupal 推薦的核心版本在網站 https://www.drupal.org/project/drupal 上列出。

在 Amazon RDS 中啟動資料庫執行個體

若要使用外部資料庫搭配執行於 Elastic Beanstalk 的應用程式，請先使用 Amazon RDS 啟動資料庫執行個體。當您使用 Amazon RDS 啟動執行個體時，它會完全獨立於 Elastic Beanstalk 和您的 Elastic Beanstalk 環境之外，而且不會受到 Elastic Beanstalk 終止或監控。

使用 Amazon RDS 主控台，啟動異地同步備份的 MySQL 資料庫執行個體。選擇異地同步備份部署可確保您的資料庫將會容錯遷移，並在來源資料庫執行個體停止服務時繼續運作。

欲在預設 VPC 中啟動 RDS 資料庫執行個體

1. 開啟 RDS 主控台。

2. 在導覽窗格中，選擇 Databases (資料庫)。
3. 選擇 Create database (建立資料庫)。
4. 選擇 Standard Create (標準建立)。

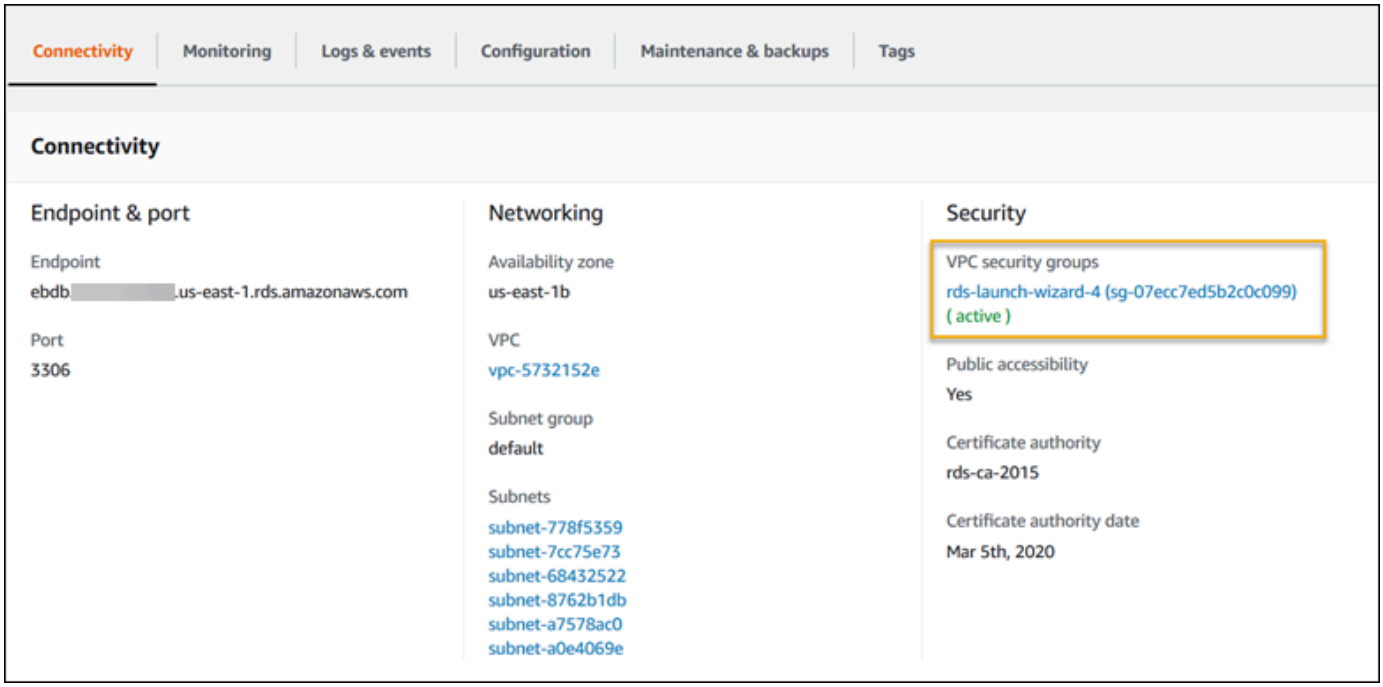
> **Important**
>
> 請勿選擇 Easy Create (輕鬆建立)。如果您選擇該選項，則您無法設定必要的設定來啟動此 RDS 資料庫。

5. 在 Additional configuration (其他設定) 下方的 Initial database name (初始資料庫名稱) 中輸入 **ebdb**。
6. 檢閱預設設定，並根據您的特定要求來調整這些設定。請注意以下選項：
   - 資料庫執行個體類別 – 選擇具有適當數量的記憶體和 CPU 功率之適合您工作負載的執行個體大小。
   - 異地同步備份部署 – 若要達到高可用性，請將此項設定為在不同的 AZ 中建立 Aurora 複本/讀取器節點。
   - 主要使用者名稱和主要密碼 – 資料庫使用者名稱和密碼。記下這些設定，以供稍後使用。
7. 檢查其餘選項的預設設定，然後選擇 Create database (建立資料庫)。

接著，修改連接至資料庫執行個體的安全群組，以允許適當連接埠的傳入流量。此安全群組與您稍後將連接至 Elastic Beanstalk 環境的相同，因此，您新增的規則將授予相同安全群組內其他資源的流量傳入許可。

修改連接至 RDS 執行個體的安全群組的傳入規則

1. 開啟 Amazon RDS 主控台。
2. 選擇 Databases (資料庫)。
3. 選擇您的資料庫執行個體名稱以檢視其詳細資料。
4. 在 Connectivity (連線) 區段中，記下顯示於此頁面的 Subnets (子網路)、Security groups (安全群組) 和 Endpoint (端點)。這樣您稍後便可使用這些資訊。
5. 在 Security (安全性) 下，可查看與資料庫執行個體相關聯的安全群組。開啟連結以檢視 Amazon EC2 主控台內的安全群組。



6. 在安全群組的詳細資訊中，選擇 Inbound (傳入)。
7. 選擇編輯。
8. 選擇 Add Rule (新增規則)。
9. 針對 Type (類型)，選擇您的應用程式所使用的資料庫引擎。
10. 對於 Source (來源)，輸入 **sg-** 檢視可用的安全群組清單。選擇與 Elastic Beanstalk 環境中使用之 Auto Scaling 群組相關聯的安全群組。以便環境中的 Amazon EC2 執行個體可以存取資料庫。

11. 選擇儲存。

建立資料庫執行個體約需要 10 分鐘。同時，啟動您的 Elastic Beanstalk 環境。

啟動 Elastic Beanstalk 環境

使用 Elastic Beanstalk 主控台建立 Elastic Beanstalk 環境。選擇 PHP (PHP) 平台，並接受預設的設定和範本程式碼。在您啟動環境之後，可設定開環境連接至資料庫，然後將 Drupal 程式碼部署至該環境。

啟動環境 (主控台)

1. 使用這個預先配置的鏈接打開 Elastic Beanstalk 控制台：控制台。aws.amazon.com/彈性豆/家 # /新 applicationName = 教程和 environmentType = LoadBalanced
2. 在 Platform (平台)，選取符合您應用程式所使用語言的平台和平台分支。
3. 針對 Application code (應用程式程式碼)，選擇 Sample application (範例應用程式)。
4. 選擇 Review and launch (檢閱和啟動)。
5. 檢視可用選項。選擇您要使用的可用選項，當您準備就緒時，請選擇 Creat app (建立應用程式)。

使用大約需要五分鐘時間建立環境，並且建立下列資源：

- EC2 執行個體 – Amazon Elastic Compute Cloud (Amazon EC2) 虛擬機器，已設為在您選擇的平台上執行 Web 應用程式。

  每個平台會執行特定的一套軟體、設定檔和指令碼，來支援特定的語言版本、架構、Web 容器或其組合。大多數的平台使用會 Apache 或 NGINX 做為反向代理，此反向代理會在您 Web 應用程式的前景執行、轉傳遞交給此 Web 應用程式的請求、提供靜態資產，並產生存取和錯誤日誌。
- 執行個體安全群組 - Amazon EC2 安全群組，已設為允許從連接埠 80 傳入的流量。此資源可讓負載平衡器傳來的 HTTP 傳輸資料，到達執行您 Web 應用程式的 EC2 執行個體。在預設情況下，不允許傳輸資料從其他通訊埠傳送。
- 負載平衡器 - Elastic Load Balancing 負載平衡器，可設定將請求分配到執行您應用程式的執行個體。負載平衡器也讓您的執行個體不需直接連接到網際網路。
- 負載平衡器安全群組 - Amazon EC2 安全群組，設為允許從連接埠 80 傳入的流量。此資源可讓來自網際網路的 HTTP 傳輸資料到達負載平衡器。在預設情況下，不允許傳輸資料從其他通訊埠傳送。
- Auto Scaling 群組 - Auto Scaling 群組，設為在執行個體終止或無法使用時，取代該執行個體。
- Amazon S3 儲存貯體 - 儲存位置，用來儲存當您使用 Elastic Beanstalk 時所建立的原始程式碼、日誌和其他成品。
- Amazon CloudWatch 警示 — 監控環境中執行個體負載的兩個 CloudWatch 警示，並在負載過高或過低時觸發警示。當警示觸發時，您的 Auto Scaling 群組會擴展或縮減以進行回應。

- AWS CloudFormation 堆疊 — Elastic Beanstalk 用 AWS CloudFormation 來啟動環境中的資源並傳播組態變更。資源定義於範本中，您可在 AWS CloudFormation 主控台中檢視此範本。
- 網域名稱 – 會路由到您 Web 應用程式的網域名稱，其格式為 *subdomain*.*region*.elasticbeanstalk.com。

> **Note**
>
> 為了增強 Elastic Beanstalk 應用程式的安全性，我們會在公共后缀列表 (PSL) 中註冊網域 elasticbeanstalk.com。為了加強安全性，如果您需要在 Elastic Beanstalk 應用程式的預設網域名稱中設定敏感性 Cookie，我們建議您使用具 `__Host-` 前置詞的 Cookie。此做法將有助於保護您的網域免受跨站請求偽造 (CSRF) 攻擊。如需更多資訊，請參閱 Mozilla 開發人員網路中的設定 Cookie 頁面。

這些資源全都由 Elastic Beanstalk 管理。當您終止環境時，Elastic Beanstalk 會終止其中的所有資源。您啟動的 RDS 資料庫執行個體在您的環境之外，因此必須負責管理其生命週期。

> **Note**
>
> Elastic Beanstalk 建立的 Amazon S3 儲存貯體會在環境間共享，且不會在環境終止時刪除。如需詳細資訊，請參閱 將 Elastic Beanstalk 與 Amazon S3 搭配使用。

設定安全設定和環境屬性

將資料庫執行個體的安全群組新增至執行環境。此程序會透過其他連接的安全群組，使 Elastic Beanstalk 重新佈建您環境中的所有執行個體。

欲將安全群組新增至您的環境

- 執行以下任意一項：
  - 使用 Elastic Beanstalk 主控台新增安全群組
    a. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
    b. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

c. 在導覽窗格中，選擇 Configuration (組態)。

d. 在 Instances (執行個體) 組態類別中，選擇 Edit (編輯)。

e. 在 EC2 安全群組下，除了 Elastic Beanstalk 建立的執行個體安全群組，請選擇要連接到執行個體的安全群組。

f. 若要儲存變更，請選擇頁面底部的儲存變更。

g. 閱讀警告的內容，然後選擇 Confirm (確認)。

- 若要使用組態檔案新增安全群組，請使用 `securitygroup-addexisting.config` 範例檔案。

接著，使用環境屬性，將連線資訊傳送至環境。此範例應用程式使用一組預設屬性，這些屬性符合您於環境佈建資料庫時 Elastic Beanstalk 所設定的屬性。

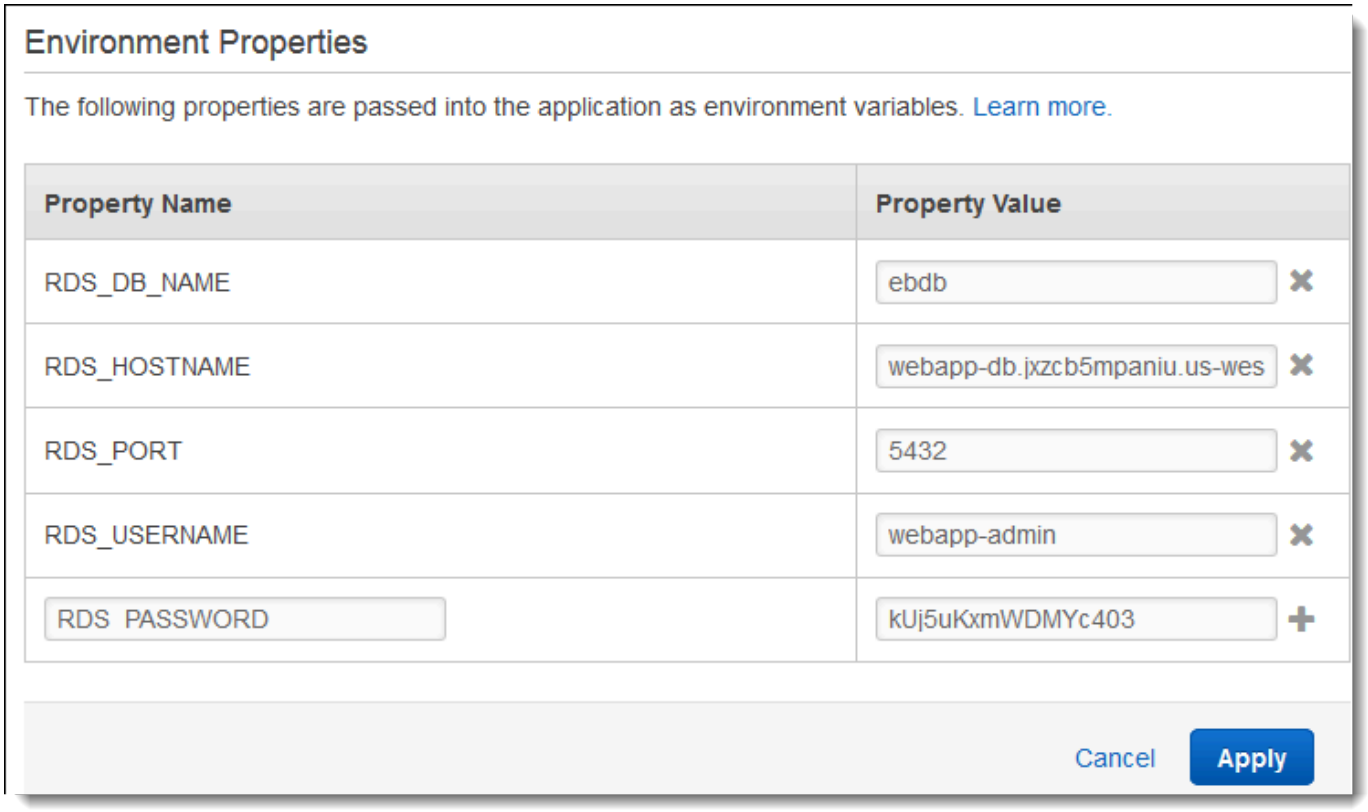
設定 Amazon RDS 資料庫執行個體的環境屬性

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇組態。
4. 在更新、監控和日誌記錄組態類別中，選擇編輯。
5. 在 Environment Properties (環境屬性) 區段，定義應用程式建立連線字串所讀取的變數。為了與具備整合式 RDS 資料庫執行個體的環境相容，請使用下列名稱與值。您可以在 RDS 主控台中找到除了密碼以外的所有值。

屬性名稱	描述	屬性值
RDS_HOSTNAME	資料庫執行個體的主機名稱。	在 Amazon RDS 主控台：端點的連線能力和安全性索引標籤上。
RDS_PORT	資料庫執行個體接受連線的連接埠。預設值在不同資料庫引擎中有所差異。	在 Amazon RDS 主控台：連接埠的連線能力和安全性索引標籤上。
RDS_DB_NAME	資料庫名稱，**ebdb**。	在 Amazon RDS 主控台：資料庫名稱的組態索引標籤上。
RDS_USERNAME	您為資料庫設定的使用者名稱。	在 Amazon RDS 主控台：主要使用者名稱的組態索引標籤上。
RDS_PASSWORD	您為資料庫設定的密碼。	無法在 Amazon RDS 主控台中提供參考。



6. 若要儲存變更，請選擇頁面底部的儲存變更。

安裝 Drupal 後，您需要使用 SSH 連接到執行個體，以擷取一些組態詳細資訊。指派 SSH 金鑰給您環境的執行個體。

設定 SSH

1. 如果您先前尚未建立金鑰對，請開啟 Amazon EC2 主控台中的金鑰對頁面，並依照指示建立帳戶。
2. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
3. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

4. 在導覽窗格中，選擇組態。
5. 在 Security (安全性) 下，選擇 Edit (編輯)。

6. 於 EC2 key pair (EC2 金鑰對) 中，選擇您的金鑰對。
7. 若要儲存變更，請選擇頁面底部的儲存變更。

設定並部署您的應用程式

要為 Elastic Beanstalk 創建一個 Drupal 項目，請下載 Drupal 源代碼，並將其與 aws-eb-php-drupal Samples/ 存儲庫中的文件組合起來。 GitHub

建立 Drupal 專案

1. 執行下列命令從 www.drupal.org/download 下載 Drupal。要了解關於下載的詳細資訊，請參閱 Drupal 網站。

   如果您的 Elastic Beanstalk 環境的平台使用 PHP 7.4 或更早版本，我們建議您下載 Drupal 8.9.13 版的本教學課程。您可以執行下列命令進行下載。

   ```
   ~$ curl https://ftp.drupal.org/files/projects/drupal-8.9.13.tar.gz -o drupal.tar.gz
   ```

   如果您的平台使用 PHP 8.0 或更新版本，我們建議您下載 Drupal 9.1.5。您可以使用此命令來進行下載。

   ```
   ~$ curl https://ftp.drupal.org/files/projects/drupal-9.1.5.tar.gz -o drupal.tar.gz
   ```

   有關 Drupal 版本和其支援之 PHP 版本的詳細資訊，請參閱 Drupal 官方文件中的 PHP 要求。Drupal 網站上列出了 Drupal 推薦的核心版本。
2. 使用下列命令從範例儲存庫下載組態檔案：

   ```
   ~$ wget https://github.com/aws-samples/eb-php-drupal/releases/download/v1.1/eb-php-drupal-v1.zip
   ```
3. 解壓縮 Drupal 並變更資料夾名稱。

   若您已下載 Drupal 8.9.13：

   ```
   ~$ tar -xvf drupal.tar.gz
   ~$ mv drupal-8.9.13 drupal-beanstalk
   ~$ cd drupal-beanstalk
   ```

   若您已下載 Drupal 9.1.5：

```
~$ tar -xvf drupal.tar.gz
~$ mv drupal-9.1.5 drupal-beanstalk
~$ cd drupal-beanstalk
```

4. 在 Drupal 安裝時解壓縮組態檔案。

```
~/drupal-beanstalk$ unzip ../eb-php-drupal-v1.zip
 creating: .ebextensions/
 inflating: .ebextensions/dev.config
 inflating: .ebextensions/drupal.config
 inflating: .ebextensions/efs-create.config
 inflating: .ebextensions/efs-filesystem.template
 inflating: .ebextensions/efs-mount.config
 inflating: .ebextensions/loadbalancer-sg.config
 inflating: LICENSE
 inflating: README.md
 inflating: beanstalk-settings.php
```

驗證您的 drupal-beanstalk 資料夾結構是否正確，如下所示。

```
drupal-beanstalk$ tree -aL 1
.
### autoload.php
### beanstalk-settings.php
### composer.json
### composer.lock
### core
### .csslintrc
### .ebextensions
### .ebextensions
### .editorconfig
### .eslintignore
### .eslintrc.json
### example.gitignore
### .gitattributes
### .htaccess
### .ht.router.php
### index.php
### LICENSE
### LICENSE.txt
### modules
```

```
### profiles
### README.md
### README.txt
### robots.txt
### sites
### themes
### update.php
### vendor
### web.config
```

來自專案儲存庫的 `beanstalk-settings.php` 檔案使用您在前一步驟中定義的環境變數，設定資料庫連線。`.ebextensions` 資料夾包含在 Elastic Beanstalk 環境中建立其他資源的組態檔案。

需修改組態檔案才可適用於您的帳戶。以適當的 ID 來更換檔案中的預留位置值，並建立原始碼套件。

若要更新組態檔案並建立原始碼套件。

1. 如下修改組態檔案。

   - `.ebextensions/dev.config` – 會將您環境的存取限制為您的 IP 地址，在 Drupal 安裝程序期間予以保護。將檔案上方附近的預留位置 IP 地址取代為您的公有 IP 地址。
   - `.ebextensions/efs-create.config` – 會在您 VPC 的各個可用區域/子網路中建立 EFS 檔案系統和掛載點。在 Amazon VPC 主控台中辨識您的預設 VPC 主控台和子網路 ID。

2. 建立原始碼套件，其中包含專案資料夾中的檔案。以下命令建立一個名為 `drupal-beanstalk.zip` 的原始碼套件。它不含 `vendor` 資料夾中的檔案，其不僅佔用許多空間，在部署應用程式到 Elastic Beanstalk 時也派不上用場。

   ```
   ~/eb-drupal$ zip ../drupal-beanstalk.zip -r * .[^.]* -x "vendor/*"
   ```

將來源套件上傳至 Elastic Beanstalk，以將 Drupal 部署至您的環境。

若要部署原始碼套件

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在環境概觀頁面上，選擇 Upload and deploy (上傳和部署)。
4. 使用畫面顯示對話方塊來上傳原始碼套件。
5. 選擇 Deploy (部署)。
6. 部署完成後，您可選擇網站的 URL 以在新分頁中開啟您的網站。

安裝 Drupal

欲完成 Drupal 安裝

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在瀏覽器中選擇環境 URL 來開啟您的網站。由於網站尚未設定，您將重新引導至 Drupal 安裝精靈。
4. 使用下列資料庫的設定來執行標準安裝：
   - 資料庫名稱 – Amazon RDS 主控台顯示的資料庫名稱。
   - 資料庫使用者名稱和密碼 – 您建立資料庫時輸入的主要使用者名稱和主要密碼的值。
   - 進階選項 > 主機 – Amazon RDS 主控台顯示的資料庫執行個體端點。

安裝約需要 1 分鐘來完成。

更新 Drupal 組態和移除存取限制

Drupal 安裝程序建立名為 `settings.php` 的檔案於執行個體上的 `sites/default`資料夾。您需要原始程式碼中的這個檔案，以避免在後續部署時重設您的網站，但該檔案目前包含您不想認可來源的秘密。連接到應用程式執行個體以從設定檔擷取資訊。

若要使用 SSH 連結至您的應用程式執行個體

1. 開啟 Amazon EC2 主控台的執行個體頁面。
2. 選擇應用程式執行個體。此執行個體根據您的 Elastic Beanstalk 環境來命名。
3. 選擇連線。
4. 依照說明將執行個體與 SSH 連接。命令結果類似如下。

```
$ ssh -i ~/.ssh/mykey ec2-user@ec2-00-55-33-222.us-west-2.compute.amazonaws.com
```

從設定檔最後一行取得同步目錄 ID。

```
[ec2-user ~]$ tail -n 1 /var/app/current/sites/default/settings.php
$config_directories['sync'] = 'sites/default/files/
config_4ccfX2sPQm79p1mk5IbUq9S_FokcENO4mxyC-L18-4g_xKj_7j9ydn31kDOYOgnzMu071Tvc4Q/
sync';
```

此檔案還包含網站目前的雜湊鍵，但您可以略過目前的值，並使用自己的值。

將同步目錄路徑和雜湊鍵指派到環境屬性。來自專案儲存庫的自訂設定檔案會讀取這些屬性，以在部署期間設定網站，除了您稍早設定的資料庫連線屬性外。

Drupal 組態屬性

- SYNC_DIR – 同步目錄的路徑。
- HASH_SALT – 所有符合環境屬性要求的字串值。

在 Elastic Beanstalk 主控台中設定環境屬性

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇組態。
4. 在更新、監控和日誌記錄組態類別中，選擇編輯。

5. 向下捲動至環境屬性。
6. 選取新增環境屬性。
7. 輸入屬性名稱和值對。
8. 如果需要新增更多變數，請重複步驟 6 和步驟 7。
9. 若要儲存變更，請選擇頁面底部的儲存變更。

最終，此範例專案內含的組態檔案 (loadbalancer-sg.config)，會使用您於 dev.config 設定的 IP 地址，將連接埠 80 的 HTTP 存取限制為來自您網路的連線，以建立安全群組並將其指派至環境的負載平衡器。否則，可在您安裝 Drupal 並設定管理者帳戶前，避免可能的外部人士連線至您的網站。

更新 Drupal 的組態和移除存取限制

1. 從專案目錄中刪除 .ebextensions/loadbalancer-sg.config 檔案。

```
~/drupal-beanstalk$ rm .ebextensions/loadbalancer-sg.config
```

2. 將自訂的 settings.php 檔案複製到網站資料夾。

```
~/drupal-beanstalk$ cp beanstalk-settings.php sites/default/settings.php
```

3. 建立原始碼套件。

```
~/eb-drupal$ zip ../drupal-beanstalk-v2.zip -r * .[^.]* -x "vendor/*"
```

將來源套件上傳至 Elastic Beanstalk，以將 Drupal 部署至您的環境。

若要部署原始碼套件

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在環境概觀頁面上，選擇 Upload and deploy (上傳和部署)。
4. 使用畫面顯示對話方塊來上傳原始碼套件。

5. 選擇 Deploy (部署)。
6. 部署完成後，您可選擇網站的 URL 以在新分頁中開啟您的網站。

設定 Auto Scaling 群組

最後，請增加執行個體計數下限，藉此設定您的環境 Auto Scaling 群組。隨時至少執行兩個執行個體，避免您環境中的 Web 伺服器出現單點故障，且無須停止網站服務即可部署變更。

若要設定您環境的 Auto Scaling 群組，以維持高可用性

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇 Configuration (組態)。
4. 在 Capacity (容量) 組態類別中，選擇 Edit (編輯)。
5. 在 Auto Scaling 群組區段，將最小執行個體設定為 **2**。
6. 若要儲存變更，請選擇頁面底部的儲存變更。

若要支援跨多個執行個體上傳內容，範例專案會使用 Amazon Elastic File System 來建立共用檔案系統。在網站上建立貼文並上傳內容以將內容存放在共用檔案系統上。查看貼文並重新整理頁面數次來點擊兩個執行個體並確認共用檔案系統正在執行中。

清除

當您完成使用 Elastic Beanstalk 時，即可終止您的環境。Elastic Beanstalk 會終止與您的環境相關聯的所有 AWS 資源，例如 Amazon EC2 執行個體、資料庫執行個體、負載平衡器、安全群組和警示。

從控制台終止 Elastic Beanstalk 環境

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 選擇 Actions (動作)，然後選擇 Terminate Environment (終止環境)。
4. 使用畫面顯示對話方塊來確認環境終止。

您可以使用 Elastic Beanstalk，隨時輕鬆地為您的應用程式建立新環境。

此外，您可終止於 Elastic Beanstalk 環境外建立的資料庫資源。終止 Amazon RDS 資料庫執行個體時，您可擷取快照，稍後再將資料還原至另一個執行個體。

終止 RDS 資料庫執行個體

1. 開啟 Amazon RDS 主控台。
2. 選擇 Databases (資料庫)。
3. 選擇資料庫執行個體。
4. 選擇動作，然後選擇刪除。
5. 選擇是否建立快照，然後選擇 Delete (刪除)。

後續步驟

隨著您繼續開發應用程式，您可能會希望無須手動建立 .zip 檔案並將其上傳至 Elastic Beanstalk 主控台，即可管理環境和部署應用程式。E lastic Beanstalk 命令列介面 (EB CLI) 提供從命令列建立、設定和部署應用程式至 Elastic Beanstalk 環境的 easy-to-use 指令。

此範例應用程式使用組態檔案來進行 PHP 設定，並於資料庫建立表格 (如尚未存在)。您亦可於環境建立期間，使用組態檔案進行執行個體的安全群組設定，以避免耗時的組態更新。如需詳細資訊，請參閱使用組態檔案 (.ebextensions) 來進行進階的環境自訂。

進行開發和測試時，建議您使用 Elastic Beanstalk 的功能，將受管的資料庫執行個體直接加入您的環境。如需於環境中設定資料庫的說明，請參閱將資料庫新增至您的 Elastic Beanstalk 環境。

若您需要高效能資料庫，請考慮使用 Amazon Aurora。Amazon Aurora 是一種與 MySQL 相容的資料庫引擎，能夠以低成本提供商用資料庫的功能。欲將您的應用程式連接至不同資料庫，請重複安全群組設定步驟，並更新 RDS 相關的環境屬性。

最後，若您打算於生產環境中使用您的應用程式，建議您設定您環境的自訂網域名稱，並啟用 HTTPS 安全連線。

將 Amazon RDS 資料庫執行個體新增至您的 PHP 應用程式環境

您可以使用 Amazon Relational Database Service (Amazon RDS) 資料庫執行個體存放由應用程式收集與修改的資料。資料庫可與環境耦合並由 Elastic Beanstalk 管理，或者由另一項服務在外部分開建立與管理。本主題提供了使用 Elastic Beanstalk 主控台建立 Amazon RDS 的說明。資料庫將與環境耦合並由 Elastic Beanstalk 管理。如需將 Amazon RDS 與 Elastic Beanstalk 整合的相關詳細資訊，請參閱 將資料庫新增至您的 Elastic Beanstalk 環境。

章節

將資料庫執行個體新增到您的環境

欲將資料庫執行個體新增到您的環境

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

   > **Note**
   >
   > 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇 Configuration (組態)。
4. 在 Database (資料庫) 組態類別中，選擇 Edit (編輯)。
5. 選擇資料庫引擎，並輸入使用者名稱和密碼。
6. 若要儲存變更，請選擇頁面底部的儲存變更。

新增資料庫執行個體約需要 10 分鐘。環境更新完成時，資料庫執行個體的主機名稱和其他連線資訊會透過下列環境屬性提供給您的應用程式：

屬性名稱	描述	屬性值
RDS_HOSTNAME	資料庫執行個體的主機名稱。	在 Amazon RDS 主控台：端點的連線能力和安全性索引標籤上。
RDS_PORT	資料庫執行個體接受連線的連接埠。預設值在不同資料庫引擎中有所差異。	在 Amazon RDS 主控台：連接埠的連線能力和安全性索引標籤上。
RDS_DB_NAME	資料庫名稱，**ebdb**。	在 Amazon RDS 主控台：資料庫名稱的組態索引標籤上。
RDS_USERNAME	您為資料庫設定的使用者名稱。	在 Amazon RDS 主控台：主要使用者名稱的組態索引標籤上。
RDS_PASSWORD	您為資料庫設定的密碼。	無法在 Amazon RDS 主控台中提供參考。

如需設定與 Elastic Beanstalk 環境耦合之資料庫執行個體的相關詳細資訊，請參閱 將資料庫新增至您的 Elastic Beanstalk 環境。

下載驅動程式

欲使用 PHP 資料物件 (PDO) 來連接至資料庫，請安裝與您所選的資料庫引擎相符的驅動程式。

- MySQL – PDO_MYSQL
- PostgreSQL – PDO_PGSQL
- Oracle – PDO_OCI
- SQL Server – PDO_SQLSRV

如需更多詳細資訊，請參閱 http://php.net/manual/en/pdo.installation.php。

使用 PDO 或 MySQLi 連接至資料庫

您可以使用 $_SERVER[`*VARIABLE*`]，自環境讀取連線資訊。

以 PDO 而言，請依據主機、連接埠和名稱，建立資料來源名稱 (DSN)。使用資料庫使用者名稱和密碼，將該 DSN 傳送至 PDO 建構函式。

Example 使用 PDO 連接至 RDS 資料庫 - MySQL

```
<?php
$dbhost = $_SERVER['RDS_HOSTNAME'];
$dbport = $_SERVER['RDS_PORT'];
$dbname = $_SERVER['RDS_DB_NAME'];
$charset = 'utf8' ;

$dsn = "mysql:host={$dbhost};port={$dbport};dbname={$dbname};charset={$charset}";
$username = $_SERVER['RDS_USERNAME'];
$password = $_SERVER['RDS_PASSWORD'];

$pdo = new PDO($dsn, $username, $password);
?>
```

以其他驅動程式而言，請將 `mysql` 取代為您驅動程式的名稱 `pgsql`、`oci` 或 `sqlsrv`。

以 MySQLi 而言，請將主機名稱、使用者名稱、密碼、資料庫名稱和連接埠傳送至 `mysqli` 建構函數。

Example 使用 mysqli_connect() 連接至 RDS 資料庫

```
$link = new mysqli($_SERVER['RDS_HOSTNAME'], $_SERVER['RDS_USERNAME'],
 $_SERVER['RDS_PASSWORD'], $_SERVER['RDS_DB_NAME'], $_SERVER['RDS_PORT']);
```

使用 Symfony 連接至資料庫

對於 Symfony 3.2 版和更新版本，您可以根據 Elastic Beanstalk 設定的環境屬性，使用 `%env(`*PROPERTY_NAME*`)%` 在組態檔案中設定資料庫參數。

Example app/config/parameters.yml

```
parameters:
    database_driver:   pdo_mysql
    database_host:     '%env(RDS_HOSTNAME)%'
    database_port:     '%env(RDS_PORT)%'
    database_name:     '%env(RDS_DB_NAME)%'
    database_user:     '%env(RDS_USERNAME)%'
    database_password: '%env(RDS_PASSWORD)%'
```

如需詳細資訊，請參閱外部參數 (Symfony 3.4)。

對於舊版 Symfony，只能存取開頭為 SYMFONY__ 的環境變數。這表示無法存取 Elastic Beanstalk 定義的環境屬性，您必須定義自己的環境屬性以傳遞 Symfony 的連線資訊。

若要連接資料庫與 Symfony 2，建立環境屬性給每個參數。然後，使用 %*property.name*% 於存取組態檔中 Symfony 轉換的變數。例如，名為 SYMFONY__DATABASE__USER 的環境屬性可當成 database.user 存取。

```
    database_user:     "%database.user%"
```

如需詳細資訊，請參閱外部參數 (Symfony 2.8)。

使用 Python

此章節提供教學課程，以及使用 AWS Elastic Beanstalk 部署 Python 應用程式的資訊。

本章中的主題假設您對 Elastic Beanstalk 環境已有一定瞭解。如果您沒使用過 Elastic Beanstalk，請嘗試《入門教學》以了解基本概念。

主題

- 設定您的 Python 開發環境
- 使用 Elastic Beanstalk Python 平台
- 將 Flask 應用程式部署至 Elastic Beanstalk
- 將 Django 應用程式部署至 Elastic Beanstalk
- 將 Amazon RDS 資料庫執行個體新增到您的 Python 應用程式環境
- Python 工具與資源

設定您的 Python 開發環境

將 Python 開發環境設定為先在本機測試您的應用程式，再將其部署至 AWS Elastic Beanstalk。此主題概述開發環境設定步驟，並提供實用工具的安裝頁面連結。

為了遵循本指南的程序，您需要命令列終端機或 shell 來執行命令。命令清單前會出現提示字元 ($) 及目前的目錄名稱 (如有)。

```
~/eb-project$ this is a command
this is output
```

在 Linux 和 macOS 上，您可以使用偏好的 Shell 和套件軟體管理工具。在視窗上，你可以安裝視窗子系統為 Linux 得到一個視窗集成的 Ubuntu 和 Bash 的版本。

如需了解適用所有語言的常見設定步驟和工具，請參閱設定您的開發機器。

章節

必要條件

針對您將透過 Elastic Beanstalk 部署的所有 Python 應用程式，常見的先決條件為：

1. 與您的應用程式將使用的 Elastic Beanstalk Python 平台版本相符的 Python 版本。
2. 符合您 Python 版本的 `pip` 公用程式。這可用於安裝並列出專案的依存項目，讓 Elastic Beanstalk 知道如何設定您應用程式的環境。
3. 命 AWS Elastic Beanstalk 令列介面 (EB CLI)。這可透過 Elastic Beanstalk 部署所需的檔案，初始化您的應用程式。
4. 正常運作的 `ssh` 安裝。在您需要檢查或除錯部署時，這可用於連接執行中的執行個體。
5. `virtualenv` 套件。這可建立將用於開發並測試您應用程式的環境，Elastic Beanstalk 無須安裝您應用程式不需要的額外套件，即可複寫該環境。使用以下命令安裝此套件：

   ```
   $ pip install virtualenv
   ```

如需安裝 Python、pip 和 EB CLI 的說明，請參閱安裝 EB CLI。

使用虛擬環境

先決條件安裝完成後，請透過 `virtualenv` 設定虛擬環境，以安裝應用程式的依存項目。透過虛擬環境，您可清楚分辨應用程式所需的套件，將該套件安裝於執行您應用程式的 EC2 執行個體。

欲設定虛擬環境

1. 開啟命令列視窗並輸入：

   ```
   $ virtualenv /tmp/eb_python_app
   ```

將 *eb_python_app* 取代為對您應用程式有意義的名稱 (最好使用您應用程式的名稱)。該 `virtualenv` 命令在指定的目錄中為您建立一個虛擬環境，並列印其操作的結果：

```
Running virtualenv with interpreter /usr/bin/python
New python executable in /tmp/eb_python_app/bin/python3.7
Also creating executable in /tmp/eb_python_app/bin/python
Installing setuptools, pip...done.
```

2. 您的虛擬環境就緒後，請執行位於環境 `activate` 目錄的 `bin` 指令碼來加以啟動。例如，欲啟動前一步建立的 eb_python_app 環境，您需要輸入：

```
$ source /tmp/eb_python_app/bin/activate
```

虛擬環境會於命令提示字元的開頭列印其名稱 (如 (`eb_python_app`))，藉此提醒您正處於虛擬 Python 環境。

3. 要停止使用您的虛擬環境並返回到系統的預設 Python 解譯器及其所有已安裝的程式庫，請執行 `deactivate` 命令。

```
(eb_python_app) $ deactivate
```

> **Note**
>
> 建立後，即可隨時再次執行虛擬環境的 `activate` 指令碼，將其重新啟動。

設定適用 Elastic Beanstalk 的 Python 專案

您可使用 Elastic Beanstalk CLI，準備將您的 Python 應用程式透過 Elastic Beanstalk 進行部署。

透過 Elastic Beanstalk 設定 Python 應用程式以進行部署

1. 在您的虛擬環境中，回到您專案樹狀目錄 (`python_eb_app`) 的最上層，並輸入：

```
pip freeze >requirements.txt
```

此命令會將您安裝於虛擬環境的套件名稱和版本，複製至 `requirements.txt`，例如，若 PyYAML 套件版本 3.11 安裝在您的虛擬環境中，那麼此檔案將包含下列這一行：

```
PyYAML==3.11
```

這可讓 Elastic Beanstalk 使用與您用於開發並測試應用程式的相同套件和版本，複寫您應用程式的 Python 環境。

2. 透過 eb init 命令設定 EB CLI 儲存庫。依提示選擇區域、平台和其他選項。如需詳細說明，請參閱使用 EB CLI 管理 Elastic Beanstalk 環境。

根據預設，Elastic Beanstalk 會尋找名為 `application.py` 的檔案，以啟動您的應用程式。若您建立的 Python 專案並沒有該項目，則您應用程式的環境必須進行調整。您也將需要設定環境變數，藉此載入您應用程式的模組。如需詳細資訊，請參閱「使用 Elastic Beanstalk Python 平台」。

使用 Elastic Beanstalk Python 平台

AWS Elastic Beanstalk Python 平台是一組 Python web 應用程式的平台版本，可以使用 WSGI 在代理伺服器後面執行。每個平台分支對應到一個 Python 版本，如 Python 3.8。

從 Amazon Linux 2 平台分支開始，Elastic Beanstalk 提供 Gunicorn 作為預設 WSGI 伺服器。

您可以新增 `Procfile` 到原始碼套件中，為您的應用程式指定和設定 WSGI 伺服器。如需詳細資訊，請參閱the section called “Procfile”。

您可以使用由 Pipenv 建立的 `Pipfile` 和 `Pipfile.lock` 檔案來指定 Python 套件相依性和其他要求。如需有關指定相依性的詳細資訊，請參閱the section called “指定相依性”。

Elastic Beanstalk 提供組態選項，您可用其於 Elastic Beanstalk 環境中自訂 EC2 執行個體上執行的軟體。您可以設定應用程式所需的環境變數，啟用至 Amazon S3 的日誌輪換，並在應用程式來源中，將包含靜態檔案的資料夾映射到代理伺服器提供的路徑。

Elastic Beanstalk 主控台中提供了修改正在執行環境組態的組態選項。要避免在終止環境的組態時遺失組態，您可以使用已儲存組態來儲存您的設定，並在之後套用至另一個環境。

若要將設定儲存於原始程式碼，您可以包含組態檔案。每次您建立環境或部署應用程式，組態檔案裡的設定就會套用。您也可以使用組態檔案來安裝套件、執行指令碼，並在部署期間執行其他執行個體自訂操作。

在 Elastic Beanstalk 主控台中套用的設定會覆寫組態檔案中相同的設定 (如存在)。這可讓您在組態檔案中擁有預設設定，並以主控台的環境專屬設定覆寫之。如需優先順序以及其他變更設定方法的詳細資訊，請參閱組態選項。

若要透過 pip 提供 Python 套件，您可以在應用程式原始碼的根目錄納入要求檔案。Elastic Beanstalk 會於部署期間安裝要求檔案指定的任何套件。如需詳細資訊，請參閱the section called “指定相依性”。

如需各種擴充 Elastic Beanstalk Linux 類型平台方式的詳細資訊，請參閱 the section called “擴充 Linux 平台”。

設定您的 Python 環境

Python 平台設定可讓您微調 Amazon EC2 執行個體的行為。您可以使用 Elastic Beanstalk 主控台編輯 Elastic Beanstalk 環境的 Amazon EC2 執行個體組態。

使用 Elastic Beanstalk 主控台來設定 Python 程序設定、啟用 AWS X-Ray、啟用至 Amazon S3 的日誌輪換，以及設定您的應用程式可以從環境讀取的變數。

在 Elastic Beanstalk 主控台中設定 Python 環境

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

   > **Note**
   >
   > 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇 Configuration (組態)。
4. 在更新、監控和日誌記錄組態類別中，選擇編輯。

Python 設定

- 代理伺服器 – 要在您的環境執行個體上使用的代理伺服器。預設使用 nginx。
- WSGI Path (WSGI 路徑) – 主要應用程式檔案的名稱或路徑，例如 `application.py` 或 `django/wsgi.py`。
- NumProcesses (NumProcesses) – 每個應用程式執行個體上執行的程序數目。
- NumThreads (NumThreads) – 每個程序中執行的執行緒數目。

AWS X-Ray 設定

- X-Ray daemon – 執行 AWS X-Ray 常駐程式以處理來自 適用於 Python 的 AWS X-Ray SDK 的追蹤資料。

日誌選項

Log Options (日誌選項) 區段有兩個設定：

- Instance profile (執行個體描述檔) – 指定執行個體描述檔，此描述檔具備存取和您應用程式相關聯 Amazon S3 儲存貯體的許可。
- Enable log file rotation to Amazon S3 (啟用 Amazon S3 的日誌檔案輪換) – 指定是否將應用程式 Amazon EC2 執行個體的日誌檔案複製到與應用程式關聯的 Amazon S3 儲存貯體。

靜態檔案

為改善效能，您可以使用 Static files (靜態檔案) 區段來設定代理伺服器，以為來自 Web 應用程式一組目錄中的靜態檔案 (例如 HTML 或影像) 提供服務。對於每個目錄，您可以設定目錄映射的虛擬路徑。代理伺服器收到位於指定路徑下的檔案請求時，會直接提供檔案而非將請求路由至您的應用程式。

如需使用組態檔案或 Elastic Beanstalk 主控台設定靜態檔案的詳細資訊，請參閱the section called “靜態檔案”。

根據預設，Python 環境中的代理伺服器會從 `static` 路徑上名為 `/static` 的資料夾中提供任何檔案。例如，若您的應用程式原始碼在名為 `logo.png` 的資料夾中有一個名為 `static` 的檔案，則代理伺服器會以 *`subdomain`*`.elasticbeanstalk.com/static/logo.png` 將此檔案提供給使用者。您可以如本節所述來設定其他對應。

環境屬性

您可以使用環境屬性，為您的應用程式提供資訊，並設定環境變數。例如，您可以建立名為 `CONNECTION_STRING` 的環境屬性，藉此指定連線字串，供您的應用程式使用以連接至資料庫。

在執行於 Elastic Beanstalk 的 Python 環境內，可透過 Python 的 `os.environ` 字典取得這些數值。如需詳細資訊，請參閱 http://docs.python.org/library/os.html。

您可以使用類似下列的程式碼，存取金鑰和參數：

```
import os
endpoint = os.environ['API_ENDPOINT']
```

環境屬性亦可為架構提供資訊。例如，您可以建立一個名為 `DJANGO_SETTINGS_MODULE` 的屬性，將 Django 設定為使用特定設定模組。依據環境而異，此值可能是 `development.settings`、`production.settings` 等。

如需詳細資訊，請參閱 環境屬性與其他軟體設定。

Python 組態命名空間

您可以使用組態檔案來設定組態選項，並在部署期間執行其他的執行個體設定工作。組態選項可由 Elastic Beanstalk 服務或您使用的平台來定義，並且會組織成「命名空間」。

Python 平台於 `aws:elasticbeanstalk:environment:proxy`、`aws:elasticbeanstalk:environment:proxy:st` 和 `aws:elasticbeanstalk:container:python` 命名空間定義的選項。

下列範例組態檔案指定組態選項設定，以建立名為 `DJANGO_SETTINGS_MODULE` 的環境屬性、選擇 Apache 代理伺服器、將名為 `statichtml` 的目錄對應至路徑 `/html` 的兩個靜態檔案選項，以及將名為 `staticimages` 的目錄對應至路徑 `/images`，以及指定 `aws:elasticbeanstalk:container:python` 命名空間的其他設定。此命名空間內含的選項，可讓您指定原始碼內 WSGI 指令碼的位置，以及以 WSGI 執行所需的執行緒數量和程序。

```
option_settings:
  aws:elasticbeanstalk:application:environment:
    DJANGO_SETTINGS_MODULE: production.settings
  aws:elasticbeanstalk:environment:proxy:
    ProxyServer: apache
  aws:elasticbeanstalk:environment:proxy:staticfiles:
    /html: statichtml
    /images: staticimages
  aws:elasticbeanstalk:container:python:
    WSGIPath: ebdjango.wsgi:application
    NumProcesses: 3
    NumThreads: 20
```

備註

- 如果您使用的是 Amazon Linux AMI Python 平台版本 (Amazon Linux 2 之前的版本)，請將 `WSGIPath` 的數值替換為 `ebdjango/wsgi.py`。此範例中的數值適用於 Gunicorn WSGI 伺服器，Amazon Linux AMI 平台版本不支援此伺服器。
- 此外，這些較舊的平台版本會使用不同的命名空間來設定靜態檔案 — `aws:elasticbeanstalk:container:python:staticfiles`。它具有與標準靜態文件命名空間相同的選項名稱和語義。

組態檔案亦支援多個金鑰，可進一步修改您環境執行個體上的軟體。本範例使用套件金鑰來安裝具備 yum 的 Memcached，同時透過容器指令於部署期間執行可設定伺服器的命令：

```
packages:
  yum:
    libmemcached-devel: '0.31'

container_commands:
  collectstatic:
    command: "django-admin.py collectstatic --noinput"
  01syncdb:
    command: "django-admin.py syncdb --noinput"
    leader_only: true
  02migrate:
    command: "django-admin.py migrate"
    leader_only: true
  03wsgipass:
    command: 'echo "WSGIPassAuthorization On" >> ../wsgi.conf'
  99customize:
    command: "scripts/customize.sh"
```

Elastic Beanstalk 可提供許多組態選項讓您自訂環境。除了組態檔案，您也可以使用主控台、已儲存組態、EB CLI 或 AWS CLI 來設定組態選項。如需詳細資訊，請參閱「組態選項」。

使用 Procfile 設定 WSGI 伺服器

您可以新增 `Procfile` 到原始碼套件中，為您的應用程式指定和設定 WSGI 伺服器。下列範例使用 `Procfile` 將 uWSGI 指定為伺服器並進行設定。

Example Procfile

```
web: uwsgi --http :8000 --wsgi-file application.py --master --processes 4 --threads 2
```

下列範例使用 `Procfile` 來設定 Gunicorn (預設 WSGI 伺服器)。

Example Procfile

```
web: gunicorn --bind :8000 --workers 3 --threads 2 project.wsgi:application
```

> **備註**
>
> - 如果您設定 Gunicorn 以外的任何 WSGI 伺服器，請務必也將其指定為應用程式的相依性，以便將它安裝在您的環境執行個體上。如需有關相依性規格的詳細資訊，請參閱 the section called “指定相依性”。
> - WSGI 伺服器的預設連接埠是 8000。如果您在 `Procfile` 命令中指定不同的連接埠號碼，請將 `PORT` 環境屬性設定為此連接埠號碼。

當您使用 `Procfile` 時，它會覆寫您使用組態檔設定的 `aws:elasticbeanstalk:container:python` 命名空間選項。

如需有關 `Procfile` 用量的詳細資訊，請展開the section called “擴充 Linux 平台”中的 Buildfile 和 Procfile 區段 。

使用要求檔案指定相依性

Python 應用程式通常與其他第三方 Python 套件具有相依性。使用 Elastic Beanstalk Python 平台時，您有數種方法可以指定您的應用程式相依的 Python 套件。

使用 **`pip`** 和 **`requirements.txt`**

安裝 Python 套件的標準工具是 `pip`。它有一項功能可讓您於單一要求檔案內，指定所需的所有套件 (及其版本)。如需有關要求檔案的詳細資訊，請參閱 pip 文件網站上的要求檔案格式。

建立名為 `requirements.txt` 的檔案並將其置於原始碼套件的最上層目錄。以下是 Django 的範例 `requirements.txt` 檔案。

```
Django==2.2
mysqlclient==2.0.3
```

在您的開發環境中，您可以使用 `pip freeze` 命令來產生您的要求檔案。

```
~/my-app$ pip freeze > requirements.txt
```

為了確保您的要求檔案僅包含您應用程式實際使用的套件，請使用虛擬環境來安裝這些套件。在虛擬環境外，`pip freeze` 輸出將包含安裝於開發機器的所有 `pip` 套件，包括您的作業系統隨附的套件。

> **Note**
>
> 在 Amazon Linux AMI Python 平台版本上，Elastic Beanstalk 本身不支援 Pipenv 或 Pipenv。如果您使用 Pipenv 管理應用程式相依性，請執行下列命令來產生 `requirements.txt` 檔案。
>
> ```
> ~/my-app$ pipenv lock -r > requirements.txt
> ```
>
> 若要進一步了解，請參閱 Pipenv 文件中的產生 requirements.txt。

使用 Pipenv 和 **`Pipfile`**

Pipenv 是一個現代化的 Python 套件工具。它將套件安裝與建立和管理相依性檔案以及應用程式的 virtualenv 結合在一起。如需詳細資訊，請參閱 Pipenv：適用於人類的 Python 開發工作流程。

Pipenv 會維護兩個檔案：

- `Pipfile` — 此檔案包含各種類型的相依性和要求。
- `Pipfile.lock` — 此檔案包含可啟用決定性組建的版本快照。

您可在開發環境中建立此類檔案，並將它們包含在您部署到 Elastic Beanstalk 之原始碼套件的最上層目錄中。如需有關這兩個檔案的詳細資訊，請參閱範例 Pipfile 和 Pipfile.lock。

下列範例使用 Pipenv 安裝 Django 和 Django REST 框架。這些命令會建立檔案 `Pipfile` 和 `Pipfile.lock`。

```
~/my-app$ pipenv install django
~/my-app$ pipenv install djangorestframework
```

優先順序

如果您包含本主題中描述的多個要求檔案，Elastic Beanstalk 只會使用其中一個。下列清單以遞減順序顯示優先順序。

1. `requirements.txt`
2. `Pipfile.lock`
3. `Pipfile`

> **Note**
>
> 從 2023 年 3 月 7 日 Amazon Linux 2 平台版本開始，如果您提供一個以上的此類檔案，Elastic Beanstalk 將發出主控台訊息，說明部署期間所使用的相依性檔案。

下列步驟說明了 Elastic Beanstalk 在部署執行個體時安裝相依性所遵循的邏輯。

- 如果有 `requirements.txt` 檔案，我們將使用命令 `pip install -r requirements.txt`。
- 從 2023 年 3 月 7 日 Amazon Linux 2 平台版本開始，如果沒有 `requirements.txt` 檔案，但有 `Pipfile.lock` 檔案，我們將使用命令 `pipenv sync`。在該版本之前，我們使用的是 `pipenv install --ignore-pipfile`。
- 如果既沒有 `requirements.txt` 檔案，也沒有 `Pipfile.lock` 檔案，但有 `Pipfile` 檔案，我們將使用命令 `pipenv install --skip-lock`。
- 如果找不到這三個要求檔案，我們不會安裝任何應用程式相依性。

將 Flask 應用程式部署至 Elastic Beanstalk

Flask 是適用於 Python 的開放原始碼 Web 應用程式架構。本教程將引導您完成生成 Flask 應用程序並將其部署到 AWS Elastic Beanstalk 環境的過程。

在本教學中，您將執行下列作業：

- 透過 Flask 設定 Python 虛擬環境
- 建立 Flask 應用程式
- 使用 EB CLI 部署您的網站
- 清除

必要條件

本教學假設您具備基本的 Elastic Beanstalk 操作及 Elastic Beanstalk 主控台知識。若您尚不了解，請依照 開始使用 Elastic Beanstalk 中的說明來啟動您的第一個 Elastic Beanstalk 環境。

為了遵循本指南的程序，您需要命令列終端機或 shell 來執行命令。命令清單前會出現提示字元 ($) 及目前的目錄名稱 (如有)。

```
~/eb-project$ this is a command
```

```
this is output
```

在 Linux 和 macOS 上，您可以使用偏好的 Shell 和套件軟體管理工具。在視窗上，你可以安裝視窗子系統為 Linux 得到一個視窗集成的 Ubuntu 和 Bash 的版本。

Flask 需要 Python 3.7 或更高版本。本教學課程中，我們會使用 Python 3.7 和對應的 Elastic Beanstalk 平台版本。遵循設定您的 Python 開發環境中的指示安裝 Python。

教學課程將會安裝 Flask 架構。

本教學也使用 Elastic Beanstalk 命令列界面 (EB CLI)。關於安裝和設定 EB CLI 的詳細資訊，請參閱安裝 EB CLI 和設定 EB CLI。

透過 Flask 設定 Python 虛擬環境

建立應用程式的專案目錄和虛擬環境，並安裝 Flask。

設定專案環境

1. 建立專案目錄。

   ```
   ~$ mkdir eb-flask
   ~$ cd eb-flask
   ```

2. 建立並啟用名為 `virt` 的虛擬環境：

   ```
   ~/eb-flask$ virtualenv virt
   ~$ source virt/bin/activate
   (virt) ~/eb-flask$
   ```

   您的命令提示字元前方將加上 (`virt`)，表示您正位於虛擬環境中。本教學課程的其餘部分都使用此虛擬環境。

3. 使用 `pip install` 安裝 Flask：

   ```
   (virt)~/eb-flask$ pip install flask==2.0.3
   ```

4. 使用 `pip freeze` 檢視已安裝的程式庫：

   ```
   (virt)~/eb-flask$ pip freeze
   click==8.1.1
   ```

```
Flask==2.0.3
itsdangerous==2.1.2
Jinja2==3.1.1
MarkupSafe==2.1.1
Werkzeug==2.1.0
```

本命令列出您虛擬環境安裝的所有套件。由於您在虛擬環境中，全域性安裝的套件 (例如 EB CLI) 不會出現。

5. 將 `pip freeze` 的輸出儲存至名為 `requirements.txt` 的檔案。

```
(virt)~/eb-flask$ pip freeze > requirements.txt
```

這個檔案指示 Elastic Beanstalk 在部署期間安裝程式庫。如需詳細資訊，請參閱 使用要求檔案指定相依性。

建立 Flask 應用程式

接著，請建立您將使用 Elastic Beanstalk 進行部署的應用程式。我們將建立一個「Hello World」RESTful Web 服務。

於此目錄建立名為 `application.py` 的新文字檔案，內含下列內容：

Example **`~/eb-flask/application.py`**

```
from flask import Flask

# print a nice greeting.
def say_hello(username = "World"):
    return '<p>Hello %s!</p>\n' % username

# some bits of text for the page.
header_text = '''
    <html>\n<head> <title>EB Flask Test</title> </head>\n<body>'''
instructions = '''
    <p><em>Hint</em>: This is a RESTful web service! Append a username
    to the URL (for example: <code>/Thelonious</code>) to say hello to
    someone specific.</p>\n'''
home_link = '<p><a href="/">Back</a></p>\n'
footer_text = '</body>\n</html>'
```

```
# EB looks for an 'application' callable by default.
application = Flask(__name__)

# add a rule for the index page.
application.add_url_rule('/', 'index', (lambda: header_text +
    say_hello() + instructions + footer_text))

# add a rule when the page is accessed with a name appended to the site
# URL.
application.add_url_rule('/<username>', 'hello', (lambda username:
    header_text + say_hello(username) + home_link + footer_text))

# run the app.
if __name__ == "__main__":
    # Setting debug to True enables debug output. This line should be
    # removed before deploying a production app.
    application.debug = True
    application.run()
```

此範例會列印自訂問候語，該問候語依用於存取該服務的路徑而異。

> **Note**
>
> 在執行應用程式前新增 `application.debug = True`，可啟用除錯輸出，以避免錯誤發生。以開發而言，這是好的作法，但您應於生產程式碼移除除錯陳述式，因為除錯輸出可能會透露應用程式的內在層面。

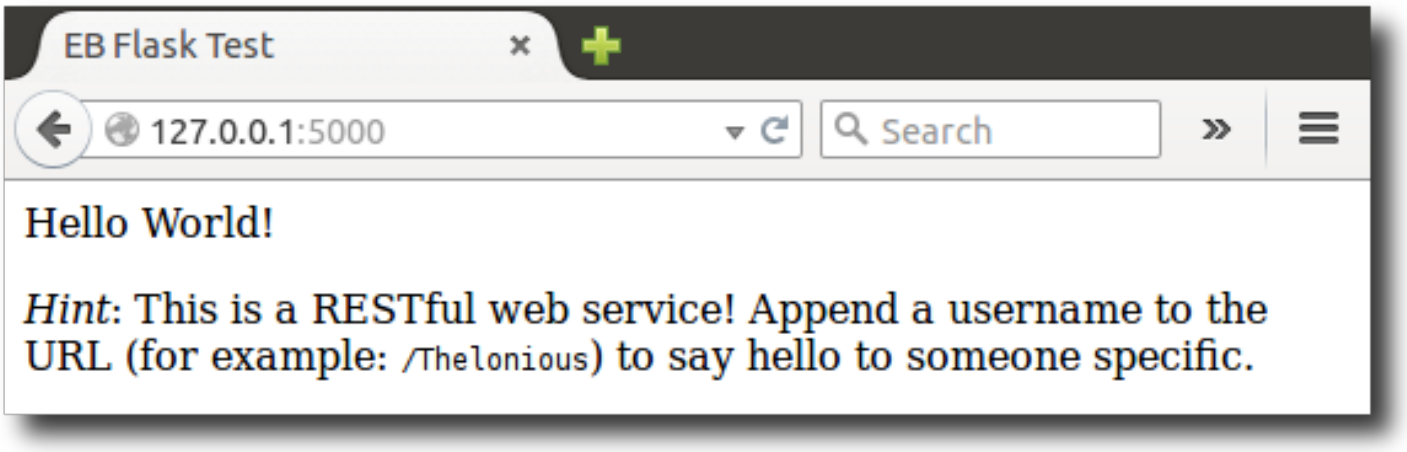
使用 `application.py` 做為檔案名稱並提供可呼叫的 `application` 物件 (此例為 Flask 物件)，Elastic Beanstalk 將能夠輕鬆找到應用程式的程式碼。

與 Python 一起執行 `application.py`：

```
(virt) ~/eb-flask$ python application.py
 * Serving Flask app "application" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
```

```
 * Debugger PIN: 313-155-123
```

在 Web 瀏覽器中開啟 `http://127.0.0.1:5000/`。您應看到應用程式正在執行，並顯示索引頁面：



檢查伺服器日誌，以查看您請求的輸出。您可以輸入 Ctrl+C，停止 web 伺服器並返回您的虛擬環境。

若您看到的是除錯輸出，請修正錯誤，並確認應用程式正於本機執行，之後再針對 Elastic Beanstalk 進行設定。

使用 EB CLI 部署您的網站

您已新增於 Elastic Beanstalk 部署應用程式所需的所有事物。您的專案目錄現在應如下所示：

```
~/eb-flask/
|-- virt
|-- application.py
`-- requirements.txt
```

不過，在 Elastic Beanstalk 上執行應用程式並不需要 `virt` 資料夾。部署時，Elastic Beanstalk 會在伺服器執行個體上建立新的虛擬環境，並安裝 `requirements.txt` 中所列的程式庫。為了將您在部署期間上傳的原始碼套件大小降到最低，請新增 .ebignore 檔案，其會告知 EB CLI 略過 `virt` 資料夾。

Example ~/eb-flask/.ebignore

```
virt
```

接著，您將建立應用程式環境，並透過 Elastic Beanstalk 部署已設定的應用程式。

欲建立環境並部署您的 Flask 應用程式

1. 透過 eb init 命令初始化您的 EB CLI 儲存庫：

   ```
   ~/eb-flask$ eb init -p python-3.7 flask-tutorial --region us-east-2
   Application flask-tutorial has been created.
   ```

   本命令會建立名為 `flask-tutorial` 的新應用程式，並設定您的本機儲存庫，使用最新的 Python 3.7 平台版本建立環境。
2. (選用) 再次執行 eb init 來設定預設金鑰對，藉此透過 SSH 連接至執行您應用程式的 EC2 執行個體：

   ```
   ~/eb-flask$ eb init
   Do you want to set up SSH for your instances?
   (y/n): y
   Select a keypair.
   1) my-keypair
   2) [ Create new KeyPair ]
   ```

   若您已有金鑰對，請選擇一對，或依照提示建立新的金鑰對。若未出現提示，或稍後需要變更設定，請執行 eb init -i。
3. 透過 eb create 建立環境並於其中部署您的應用程式：

   ```
   ~/eb-flask$ eb create flask-env
   ```

使用大約需要五分鐘時間建立環境，並且建立下列資源：

- EC2 執行個體 – Amazon Elastic Compute Cloud (Amazon EC2) 虛擬機器，已設為在您選擇的平台上執行 Web 應用程式。

  每個平台會執行特定的一套軟體、設定檔和指令碼，來支援特定的語言版本、架構、Web 容器或其組合。大多數的平台使用會 Apache 或 NGINX 做為反向代理，此反向代理會在您 Web 應用程式的前景執行、轉傳遞交給此 Web 應用程式的請求、提供靜態資產，並產生存取和錯誤日誌。
- 執行個體安全群組 - Amazon EC2 安全群組，已設為允許從連接埠 80 傳入的流量。此資源可讓負載平衡器傳來的 HTTP 傳輸資料，到達執行您 Web 應用程式的 EC2 執行個體。在預設情況下，不允許傳輸資料從其他通訊埠傳送。

- 負載平衡器 - Elastic Load Balancing 負載平衡器，可設定將請求分配到執行您應用程式的執行個體。負載平衡器也讓您的執行個體不需直接連接到網際網路。
- 負載平衡器安全群組 - Amazon EC2 安全群組，設為允許從連接埠 80 傳入的流量。此資源可讓來自網際網路的 HTTP 傳輸資料到達負載平衡器。在預設情況下，不允許傳輸資料從其他通訊埠傳送。
- Auto Scaling 群組 - Auto Scaling 群組，設為在執行個體終止或無法使用時，取代該執行個體。
- Amazon S3 儲存貯體 - 儲存位置，用來儲存當您使用 Elastic Beanstalk 時所建立的原始程式碼、日誌和其他成品。
- Amazon CloudWatch 警示 — 監控環境中執行個體負載的兩個 CloudWatch 警示，並在負載過高或過低時觸發警示。當警示觸發時，您的 Auto Scaling 群組會擴展或縮減以進行回應。
- AWS CloudFormation 堆疊 — Elastic Beanstalk 用 AWS CloudFormation 來啟動環境中的資源並傳播組態變更。資源定義於範本中，您可在 AWS CloudFormation 主控台中檢視此範本。
- 網域名稱 – 會路由到您 Web 應用程式的網域名稱，其格式為 *subdomain*.*region*.elasticbeanstalk.com。

> **Note**
>
> 為了增強 Elastic Beanstalk 應用程式的安全性，我們會在公共后缀列表 (PSL) 中註冊網域 elasticbeanstalk.com。為了加強安全性，如果您需要在 Elastic Beanstalk 應用程式的預設網域名稱中設定敏感性 Cookie，我們建議您使用具 `__Host-` 前置詞的 Cookie。此做法將有助於保護您的網域免受跨站請求偽造 (CSRF) 攻擊。如需更多資訊，請參閱 Mozilla 開發人員網路中的設定 Cookie 頁面。

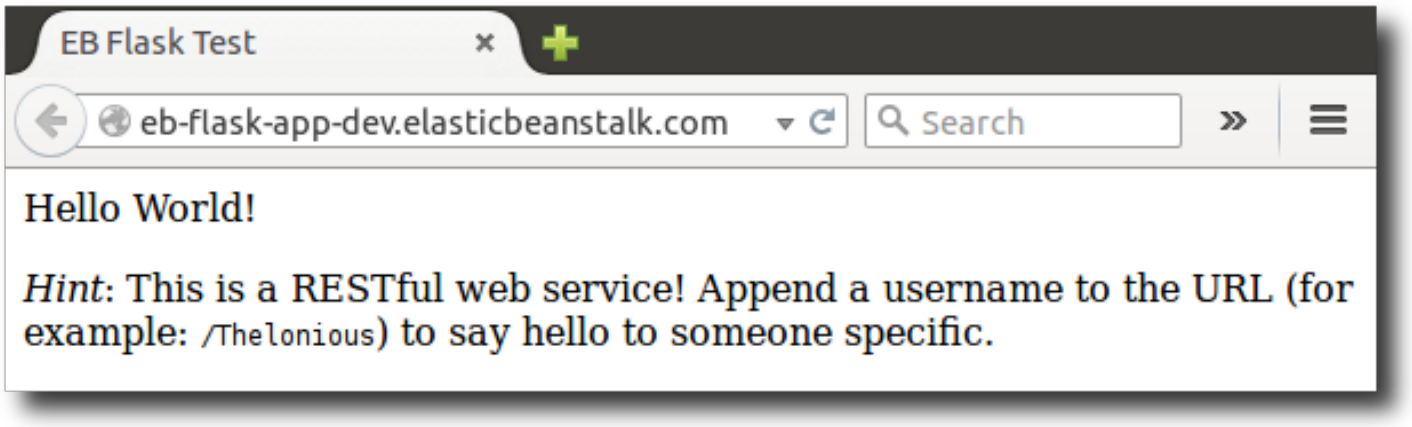
這些資源全都由 Elastic Beanstalk 管理。當您終止環境時，Elastic Beanstalk 會終止其中的所有資源。

> **Note**
>
> Elastic Beanstalk 建立的 Amazon S3 儲存貯體會在環境間共享，且不會在環境終止時刪除。如需詳細資訊，請參閱 將 Elastic Beanstalk 與 Amazon S3 搭配使用。

當環境建立程序完成，請透過 eb open 開啟您的網站：

```
~/eb-flask$ eb open
```

將開啟瀏覽器視窗，並使用為應用程式建立的網域名稱。您應該會看到與本機建立和測試相同的 Flask 網站。



若未顯示您的應用程式正在執行，或收到錯誤訊息，請參閱疑難排解部署尋求協助，以判斷錯誤發生的原因。

若您「確實」看見您的應用程式正在執行，恭喜您，您透過 Elastic Beanstalk 部署的第一個 Flask 應用程式已完成！

清除

當您完成使用 Elastic Beanstalk 時，即可終止您的環境。Elastic Beanstalk 會終止與您的環境相關聯的所有 AWS 資源，例如 Amazon EC2 執行個體、資料庫執行個體、負載平衡器、安全群組和警示。

從控制台終止 Elastic Beanstalk 環境

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 選擇 Actions (動作)，然後選擇 Terminate Environment (終止環境)。
4. 使用畫面顯示對話方塊來確認環境終止。

您可以使用 Elastic Beanstalk，隨時輕鬆地為您的應用程式建立新環境。

或者，使用 EB CLI：

```
~/eb-flask$ eb terminate flask-env
```

後續步驟

如需 Flask 的詳細資訊，請造訪 flask.pocoo.org。

若您欲嘗試其他 Python Web 框架，請參閱將 Django 應用程式部署至 Elastic Beanstalk。

將 Django 應用程式部署至 Elastic Beanstalk

本教學會演練如何將自動產生的預設 Django 網站部署至執行 Python 的 AWS Elastic Beanstalk 環境。本教學示範說明如何使用 Elastic Beanstalk 環境在雲端託管 Python Web 應用程式。

在本教學中，您將執行下列作業：

- 設定 Python 虛擬環境並安裝 Django
- 建立 Django 專案
- 針對 Elastic Beanstalk 設定您的 Django 應用程式
- 使用 EB CLI 部署您的網站
- 更新您的應用程式
- 清除

先決條件

若要使用包括 Elastic Beanstalk 在內的任何 AWS 服務，您必須擁有 AWS 帳戶及憑證。若要進一步了解並註冊，請造訪 https://aws.amazon.com/。

按照這個教學課程，您應安裝適用 Python 的所有常見先決條件，包括下列套件：

- Python 3.7 或更高版本
- `pip`
- `virtualenv`
- `awsebcli`

教學中會安裝 Django 架構。

> **Note**
>
> 要以 EB CLI 建立環境，就必須有服務角色。您可以在 Elastic Beanstalk 主控台建立環境，以建立服務角色。如果您沒有服務角色，EB CLI 會嘗試在您執行 `eb create` 時為您建立。

設定 Python 虛擬環境並安裝 Django

透過 `virtualenv` 建立虛擬環境，並用其安裝 Django 及依存項目。透過虛擬環境，您可清楚分辨應用程式需要哪些套裝服務，以便在執行您應用程式的 Amazon EC2 執行個體上安裝所需套裝服務。

下列步驟示範 Unix 系統和 Windows 必須輸入的命令，以不同的標籤顯示。

欲設定您的虛擬環境

1. 建立名為 `eb-virt` 的虛擬環境。

   Unix-based systems

   ```
   ~$ virtualenv ~/eb-virt
   ```

   Windows

   ```
   C:\> virtualenv %HOMEPATH%\eb-virt
   ```

2. 啟用虛擬環境。

   Unix-based systems

   ```
   ~$ source ~/eb-virt/bin/activate
   (eb-virt) ~$
   ```

   Windows

   ```
   C:\>%HOMEPATH%\eb-virt\Scripts\activate
   (eb-virt) C:\>
   ```

   您的命令提示字元前方會加上 `(eb-virt)`，表示您正位於虛擬環境中。

> **Note**
>
> 其餘說明會顯示您主目錄的 Linux 命令提示字元 ~$。在 Windows 上，此為 C:\Users*USERNAME*>，其中 *USERNAME* 是您的 Windows 登入名稱。

3. 使用 pip 安裝 Django。

```
(eb-virt)~$ pip install django==2.2
```

> **Note**
>
> 您安裝的 Django 版本必須與您在 Elastic Beanstalk Python 組態選擇用於部署應用程式的 Python 版本相容。如需部署的資訊，請參閱本主題中的 ???。
> 如需目前 Python 平台版本的詳細資訊，請參閱《AWS Elastic Beanstalk 平台文件》中的 Python。
> 如需 Django 版本與 Python 的相容性詳細資訊，請參閱 What Python version can I use with Django?

4. 若要確認是否已安裝 Django，請輸入以下內容。

```
(eb-virt)~$ pip freeze
Django==2.2
...
```

本命令列出您虛擬環境安裝的所有套件。稍後，使用此命令的輸出設定您的專案，以搭配 Elastic Beanstalk 使用。

建立 Django 專案

現在您已準備就緒，能夠建立 Django 專案並使用虛擬環境執行於您的機器。

> **Note**
>
> 本教學使用 Python 隨附的資料庫引擎 SQLite。這個資料庫會和您的專案檔案一同部署。至於生產環境，則建議使用 Amazon Relational Database Service (Amazon RDS)，並將之從您

的環境中獨立出來。如需更多詳細資訊，請參閱 將 Amazon RDS 資料庫執行個體新增到您的 Python 應用程式環境。

欲產生 Django 應用程式

1. 啟用您的虛擬環境。

   Unix-based systems

   ```
   ~$ source ~/eb-virt/bin/activate
   (eb-virt) ~$
   ```

   Windows

   ```
   C:\>%HOMEPATH%\eb-virt\Scripts\activate
   (eb-virt) C:\>
   ```

   您的命令提示字元前方會加上 (eb-virt) 前綴，表示您正位於虛擬環境中。

   > **Note**
   >
   > 其餘說明會顯示您主目錄的 Linux 命令提示字元 ~$ 以及 Linux 主目錄 ~/。在 Windows 上，此為 C:\Users*USERNAME*>，其中 *USERNAME* 是您的 Windows 登入名稱。

2. 使用 django-admin startproject 命令建立名為 ebdjango 的 Django 專案。

   ```
   (eb-virt)~$ django-admin startproject ebdjango
   ```

   此命令會建立名為 ebdjango 的標準 Django 網站，採用下列目錄結構。

   ```
   ~/ebdjango
     |-- ebdjango
     |   |-- __init__.py
     |   |-- settings.py
     |   |-- urls.py
     |   `-- wsgi.py
     `-- manage.py
   ```

3. 使用 `manage.py runserver` 在本機執行您的 Django 網站。

```
(eb-virt) ~$ cd ebdjango
```

```
(eb-virt) ~/ebdjango$ python manage.py runserver
```

4. 在 Web 瀏覽器中開啟 `http://127.0.0.1:8000/` 以檢視此網站。
5. 檢查伺服器日誌，以查看您請求的輸出。若要停止 Web 伺服器並返回您的虛擬環境，請按 Ctrl +C。

```
Django version 2.2, using settings 'ebdjango.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
[07/Sep/2018 20:14:09] "GET / HTTP/1.1" 200 16348
Ctrl+C
```

針對 Elastic Beanstalk 設定您的 Django 應用程式

現在您的本機已具備採用 Django 的網站，因此您可將其設定為透過 Elastic Beanstalk 進行部署。

根據預設，Elastic Beanstalk 會尋找名為 `application.py` 的檔案，以啟動您的應用程式。因為您建立的 Django 專案中不存在此檔案，所以您需要對您的應用程式環境稍加調整。您也必須設定環境變數，以便載入您的應用程式模組。

欲針對 Elastic Beanstalk 設定您的網站

1. 啟用您的虛擬環境。

   Unix-based systems

   ```
   ~/ebdjango$ source ~/eb-virt/bin/activate
   ```

   Windows

   ```
   C:\Users\USERNAME\ebdjango>%HOMEPATH%\eb-virt\Scripts\activate
   ```

2. 執行 `pip freeze`，然後將輸出儲存至名為 `requirements.txt` 的檔案。

```
(eb-virt) ~/ebdjango$ pip freeze > requirements.txt
```

Elastic Beanstalk 使用 `requirements.txt` 來判斷執行您應用程式的 EC2 執行個體應安裝哪些套件。

3. 建立名為 `.ebextensions` 的目錄。

```
(eb-virt) ~/ebdjango$ mkdir .ebextensions
```

4. 在 `.ebextensions` 目錄中，新增具有下列文字的組態檔案 `django.config`。

Example ~/ebdjango/.ebextensions/django.config

```
option_settings:
  aws:elasticbeanstalk:container:python:
    WSGIPath: ebdjango.wsgi:application
```

`WSGIPath` 設定會指定 Elastic Beanstalk 用於啟動應用程式的 WSGI 指令碼之位置。

> **Note**
>
> 如果您使用的是 Amazon Linux AMI Python 平台版本 (Amazon Linux 2 之前的版本)，請將 `WSGIPath` 的數值替換為 `ebdjango/wsgi.py`。此範例中的數值適用於 Gunicorn WSGI 伺服器，Amazon Linux AMI 平台版本不支援此伺服器。

5. 使用 `deactivate` 命令停用您的虛擬環境。

```
(eb-virt) ~/ebdjango$ deactivate
```

每當需要將套裝服務新增至應用程式，或於本機執行應用程式時，即可重新啟用您的虛擬環境。

使用 EB CLI 部署您的網站

您已新增於 Elastic Beanstalk 部署應用程式所需的所有事物。您的專案目錄現應如下所示。

```
~/ebdjango/
|-- .ebextensions
|   `-- django.config
|-- ebdjango
|   |-- __init__.py
|   |-- settings.py
```

```
|   |-- urls.py
|   `-- wsgi.py
|-- db.sqlite3
|-- manage.py
`-- requirements.txt
```

接著，您將建立應用程式環境，並透過 Elastic Beanstalk 部署已設定的應用程式。

部署後，您要立即編輯 Django 的組態，將 Elastic Beanstalk 指派給您應用程式的網域名稱新增至 Django 的 ALLOWED_HOSTS。然後，您要重新部署應用程式。這是一個 Django 安全要求，旨在阻擋 HTTP Host 標頭攻擊。如需詳細資訊，請參閱主機標頭驗證。

欲建立環境並部署您的 Django 應用程式

> **Note**
>
> 本教學使用 EB CLI 做為部署機制，但您亦可使用 Elastic Beanstalk 主控台來部署內含您專案內容的 .zip 檔案。

1. 透過 eb init 命令初始化您的 EB CLI 儲存庫。

   ```
   ~/ebdjango$ eb init -p python-3.7 django-tutorial
   Application django-tutorial has been created.
   ```

   此命令會建立名為 django-tutorial 的應用程式。這也會設定您的本機儲存庫，使用最新的 Python 3.7 平台版本建立環境。

2. (選用) 再次執行 eb init 來設定預設金鑰對，藉此使用 SSH 連接至執行您應用程式的 EC2 執行個體：

   ```
   ~/ebdjango$ eb init
   Do you want to set up SSH for your instances?
   (y/n): y
   Select a keypair.
   1) my-keypair
   2) [ Create new KeyPair ]
   ```

   若您已有金鑰對，請選擇一對，或依照提示建立金鑰對。若未出現提示，或稍後需要變更設定，請執行 eb init -i。

3. 使用 eb create 建立環境並於其中部署您的應用程式。

```
~/ebdjango$ eb create django-env
```

> **Note**
>
> 若您看到了 "service role required" (需要服務角色) 錯誤訊息，請以互動方式執行 eb create (無須指定環境名稱)，EB CLI 便會為您建立角色。

本命令會建立名為 django-env 的負載平衡 Elastic Beanstalk 環境。建立環境約需要 5 分鐘。當 Elastic Beanstalk 建立執行應用程式所需要的資源時，它會輸出 EB CLI 轉送至您終端機的資訊訊息。

4. 完成環境建立程序後，請執行 eb status 尋找新環境的網域名稱。

```
~/ebdjango$ eb status
Environment details for: django-env
  Application name: django-tutorial
  ...
  CNAME: eb-django-app-dev.elasticbeanstalk.com
  ...
```

您環境的網域名稱為 CNAME 屬性的數值。

5. 在 settings.py 目錄中，開啟檔案 ebdjango。找出 ALLOWED_HOSTS 設定，然後將您在上個步驟中找到的應用程式網域名稱新增至設定值中。如果您無法找到檔案中的這個設定，請在新行中新增。

```
...
ALLOWED_HOSTS = ['eb-django-app-dev.elasticbeanstalk.com']
```

6. 儲存檔案，然後執行 eb deploy 來部署應用程式。執行 eb deploy 時，EB CLI 會封裝您專案目錄的內容，並將其部署至您的環境。

```
~/ebdjango$ eb deploy
```

> **Note**
>
> 如果您使用 Git 來處理專案，請參閱搭配 Git 使用 EB CLI。

7. 完成環境更新程序後，請使用 eb open 開啟您的網站。

```
~/ebdjango$ eb open
```

這會開啟瀏覽器視窗，並使用為應用程式建立的網域名稱。您應該會看到與本機建立和測試相同的 Django 網站。

若未顯示您的應用程式正在執行，或收到錯誤訊息，請參閱疑難排解部署尋求協助，以判斷錯誤發生的原因。

若您「確實」看見您的應用程式正在執行，恭喜您，您透過 Elastic Beanstalk 部署的第一個 Django 應用程式已完成！

更新您的應用程式

現在，您已於 Elastic Beanstalk 上執行應用程式，可以更新並重新部署應用程式或其組態，Elastic Beanstalk 會負責更新您的執行個體，並啟動新的應用程式版本。

在此範例中，我們將啟用 Django 管理主控台，並進行數項設定。

修改您的網站設定

根據預設，您的 Django 網站會使用 UTC 時區顯示時間。您可在 `settings.py` 中指定時區來進行變更。

欲變更您網站的時區

1. 在 `TIME_ZONE` 中修改 `settings.py` 設定。

   Example ~/ebdjango/ebdjango/settings.py

```
...
# Internationalization
LANGUAGE_CODE = 'en-us'
TIME_ZONE = 'US/Pacific'
USE_I18N = True
```

```
USE_L10N = True
USE_TZ = True
```

如需時區清單，請造訪此頁面。

2. 將應用程式部署至您的 Elastic Beanstalk 環境。

```
~/ebdjango/$ eb deploy
```

建立網站管理員

您可以為 Django 應用程式建立網站管理員，從網站直接存取管理主控台。管理員登入詳細資訊安全存放於本機資料庫映像，位於 Django 產生的預設專案中。

欲建立網站管理員

1. 初始化您 Django 應用程式的本機資料庫。

```
(eb-virt) ~/ebdjango$ python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying sessions.0001_initial... OK
```

2. 執行 `manage.py createsuperuser` 以建立管理員。

```
(eb-virt) ~/ebdjango$ python manage.py createsuperuser
Username: admin
```

```
Email address: me@mydomain.com
Password: ********
Password (again): ********
Superuser created successfully.
```

3. 若要指示 Django 存放靜態檔案的位置，請在 STATIC_ROOT 中定義 settings.py。

   Example ~/ebdjango/ebdjango/settings.py

```
# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/2.2/howto/static-files/
STATIC_URL = '/static/'
STATIC_ROOT = 'static'
```

4. 執行 manage.py collectstatic，將管理員網站的靜態資產 (JavaScript、CSS 和映像) 填入 static 目錄。

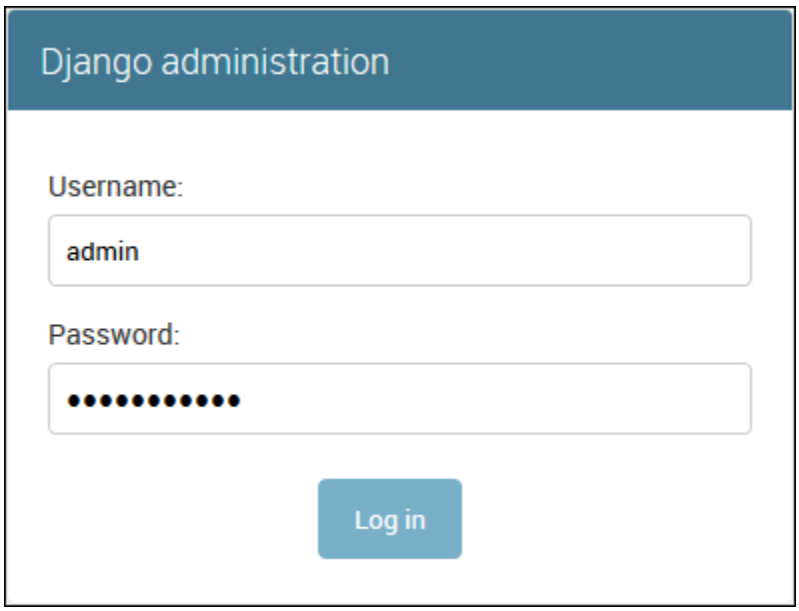
```
(eb-virt) ~/ebdjango$ python manage.py collectstatic
119 static files copied to ~/ebdjango/static
```

5. 部署您的應用程式。

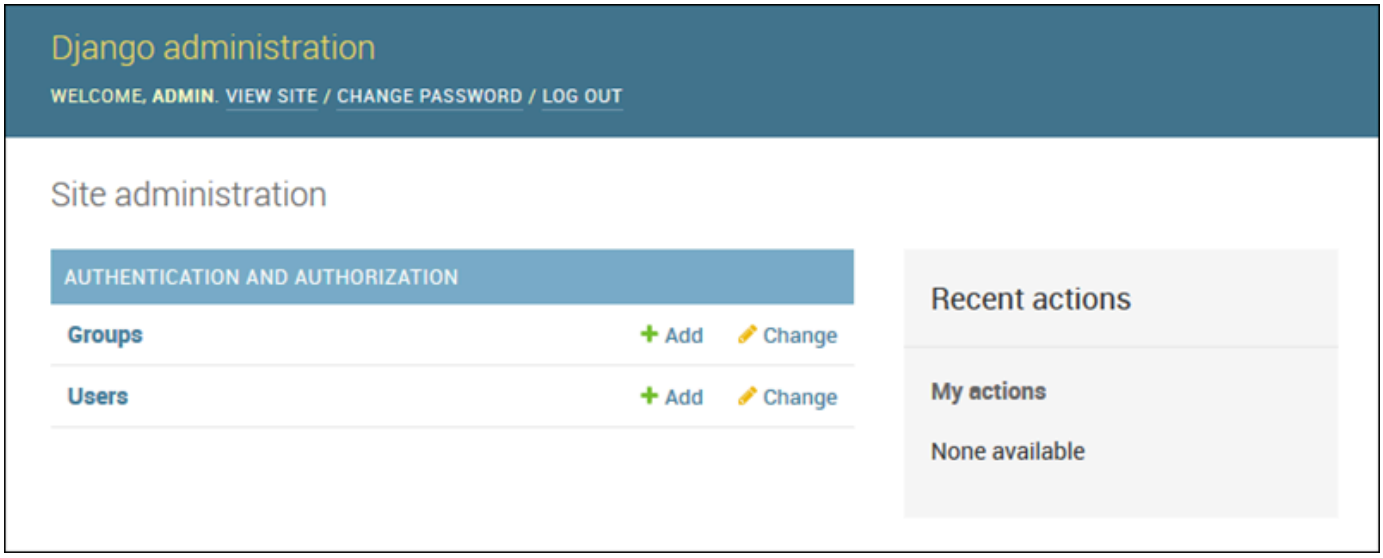
```
~/ebdjango$ eb deploy
```

6. 在您的瀏覽器中開啟網站，並將 /admin/ 附加至網站 URL，藉此檢視管理主控台，如下所示。

```
http://djang-env.p33kq46sfh.us-west-2.elasticbeanstalk.com/admin/
```



7. 使用您於步驟 2 中設定的使用者名稱和密碼登入。



您可以使用類似於本機更新/測試的程序，之後進行 eb deploy。Elastic Beanstalk 會負責更新線上伺服器，因此您能夠專心開發應用程式，不受伺服器管理作業影響！

新增資料庫遷移組態檔案

您可以將命令新增至 `.ebextensions` 指令碼，這是更新網站時所執行的指令碼。這可讓您自動產生資料庫遷移。

欲在部署應用程式時新增遷移步驟

1. 使用下列內容，建立名為 db-migrate.config 的組態檔案。

   Example ~/ebdjango/.ebextensions/db-migrate.config

   ```
   container_commands:
     01_migrate:
       command: "source /var/app/venv/*/bin/activate && python3 manage.py migrate"
       leader_only: true
   option_settings:
     aws:elasticbeanstalk:application:environment:
       DJANGO_SETTINGS_MODULE: ebdjango.settings
   ```

   此組態檔案會先在部署過程中啟用伺服器的虛擬環境並執行 manage.py migrate 命令，再啟動您的應用程式。由於此命令在啟動應用程式前執行，因此您必須明確設定 DJANGO_SETTINGS_MODULE 環境變數 (wsgi.py 通常會於啟動時負責此作業)。於命令中指定 leader_only: true，可確保當您部署多個執行個體時，該命令僅執行一次。

2. 部署您的應用程式。

   ```
   ~/ebdjango$ eb deploy
   ```

清除

為了節省開發階段間的執行個體時數和其他 AWS 資源，請透過 eb terminate 終止您的 Elastic Beanstalk 環境。

```
~/ebdjango$ eb terminate django-env
```

此命令會終止環境及於其中執行的所有 AWS 資源。但它不會刪除應用程式，因此您隨時能夠再次執行 eb create，以同一組態建立更多的環境。如需 EB CLI 命令的詳細資訊，請參閱使用 EB CLI 管理 Elastic Beanstalk 環境。

如不再需要範例應用程式，您也可以移除專案資料夾和虛擬環境。

```
~$ rm -rf ~/eb-virt
~$ rm -rf ~/ebdjango
```

下一步驟

如需 Django 的詳細資訊 (包括更深入的教學)，請參閱官方文件。

若欲嘗試其他 Python Web 框架，請參閱將 Flask 應用程式部署至 Elastic Beanstalk。

將 Amazon RDS 資料庫執行個體新增到您的 Python 應用程式環境

您可以使用 Amazon Relational Database Service (Amazon RDS) 資料庫執行個體存放由應用程式收集與修改的資料。資料庫可與環境耦合並由 Elastic Beanstalk 管理，或者由另一項服務在外部分開建立與管理。本主題提供了使用 Elastic Beanstalk 主控台建立 Amazon RDS 的說明。資料庫將與環境耦合並由 Elastic Beanstalk 管理。如需將 Amazon RDS 與 Elastic Beanstalk 整合的相關詳細資訊，請參閱 將資料庫新增至您的 Elastic Beanstalk 環境。

章節

將資料庫執行個體新增到您的環境

欲將資料庫執行個體新增到您的環境

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

   > **Note**
   >
   > 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇 Configuration (組態)。
4. 在 Database (資料庫) 組態類別中，選擇 Edit (編輯)。
5. 選擇資料庫引擎，並輸入使用者名稱和密碼。
6. 若要儲存變更，請選擇頁面底部的儲存變更。

新增資料庫執行個體約需要 10 分鐘。環境更新完成時，資料庫執行個體的主機名稱和其他連線資訊會透過下列環境屬性提供給您的應用程式：

屬性名稱	描述	屬性值
RDS_HOSTNAME	資料庫執行個體的主機名稱。	在 Amazon RDS 主控台：端點的連線能力和安全性索引標籤上。
RDS_PORT	資料庫執行個體接受連線的連接埠。預設值在不同資料庫引擎中有所差異。	在 Amazon RDS 主控台：連接埠的連線能力和安全性索引標籤上。
RDS_DB_NAME	資料庫名稱，**ebdb**。	在 Amazon RDS 主控台：資料庫名稱的組態索引標籤上。
RDS_USERNAME	您為資料庫設定的使用者名稱。	在 Amazon RDS 主控台：主要使用者名稱的組態索引標籤上。
RDS_PASSWORD	您為資料庫設定的密碼。	無法在 Amazon RDS 主控台中提供參考。

如需設定與 Elastic Beanstalk 環境耦合之資料庫執行個體的相關詳細資訊，請參閱 將資料庫新增至您的 Elastic Beanstalk 環境。

下載驅動程式

將資料庫驅動程式新增到您專案的要求檔案中。

Example requirements.txt – 使用 MySQL 的 Django

```
Django==2.2
mysqlclient==2.0.3
```

適用於 Python 的常見驅動程式套件

- MySQL – `mysqlclient`
- PostgreSQL – `psycopg2`
- Oracle – `cx_Oracle`
- SQL Server – `adodbapi`

有關更多信息，請參閱 Python 資料庫界面和 Django 2.2 - 支援的資料庫。

連線至資料庫

Elastic Beanstalk 會在環境屬性中提供已連接的資料庫執行個體連線資訊。使用 `os.environ['VARIABLE']` 來讀取屬性和設定資料庫的連線。

Example Django 設定檔案 – 資料庫字典

```
import os

if 'RDS_HOSTNAME' in os.environ:
    DATABASES = {
        'default': {
            'ENGINE': 'django.db.backends.mysql',
            'NAME': os.environ['RDS_DB_NAME'],
            'USER': os.environ['RDS_USERNAME'],
            'PASSWORD': os.environ['RDS_PASSWORD'],
            'HOST': os.environ['RDS_HOSTNAME'],
            'PORT': os.environ['RDS_PORT'],
        }
    }
```

Python 工具與資源

在開發 Python 應用程式時，您可以造訪幾個地方來取得額外的協助：

資源	描述
Boto (適用於 Python 的 AWS SDK)	使用 GitHub 安裝 Boto。
Python 開發論壇	提出您的問題，並取得意見回饋。
Python 開發人員中心	可取得範本程式碼、文件、工具和其他資源等一應俱全的場所。

在 Elastic Beanstalk 上建立及部署 Ruby 應用程式

適用於 Ruby 的 AWS Elastic Beanstalk 可讓您使用 Amazon Web Services，輕鬆地部署、管理和擴展您的 Ruby Web 應用程式。所有使用 Ruby 來開發或託管 Web 應用程式的人，都能使用 Elastic

Beanstalk for Ruby。本章節會逐步解說如何使用 Elastic Beanstalk 命令列界面 (EB CLI)，將範例應用程式部署至 Elastic Beanstalk，並解釋如何更新應用程式，以使用 Rails 和 Sinatra Web 應用程式框架。

本章中的主題假設您對 Elastic Beanstalk 環境已有一定瞭解。如果您沒使用過 Elastic Beanstalk，請嘗試《入門教學》以了解基本概念。

主題

- 設定您的 Ruby 開發環境
- 使用 Elastic Beanstalk Ruby 平台
- 將 Rails 應用程式部署到 Elastic Beanstalk
- 將 Sinatra 應用程式部署到 Elastic Beanstalk
- 將 Amazon RDS 資料庫執行個體新增到您的 Ruby 應用程式環境

設定您的 Ruby 開發環境

設定 Ruby 開發環境於本機測試您的應用程式，之後再部署至 AWS Elastic Beanstalk。此主題概述開發環境設定步驟，並提供實用工具的安裝頁面連結。

為了遵循本指南的程序，您需要命令列終端機或 shell 來執行命令。命令清單前會出現提示字元 ($) 及目前的目錄名稱 (如有)。

```
~/eb-project$ this is a command
this is output
```

在 Linux 和 macOS 上，您可以使用偏好的 Shell 和套件軟體管理工具。在視窗上，你可以安裝視窗子系統為 Linux 得到一個視窗集成的 Ubuntu 和 Bash 的版本。

如需了解適用所有語言的常見設定步驟和工具，請參閱設定您的開發機器搭配 Elastic Beanstalk 使用。

章節

- 安裝 Ruby
- 安裝適用於 AWS SDK for Ruby
- 安裝 IDE 或文字編輯器

安裝 Ruby

如果您沒有 C 編譯器，請安裝 GCC。在 Ubuntu 上，使用 `apt`。

```
~$ sudo apt install gcc
```

在 Amazon Linux 上，使用 `yum`。

```
~$ sudo yum install gcc
```

安裝 RVM 來管理您的機器上的 Ruby 語言安裝。使用於 rvm.io 的命令來取得專案金鑰並執行安裝指令碼。

```
~$ gpg2 --recv-keys key1 key2
~$ curl -sSL https://get.rvm.io | bash -s stable
```

此指令碼在使用者目錄中名為 `.rvm` 的資料夾中安裝 RVM，並在您開啟新的終端機時修改您的 Shell 設定檔來載入設定指令碼。手動載入指令碼以開始使用。

```
~$ source ~/.rvm/scripts/rvm
```

請使用 `rvm get head` 來取得最新版本。

```
~$ rvm get head
```

查看 Ruby 的可用版本。

```
~$ rvm list known
# MRI Rubies
...
[ruby-]2.6[.8]
[ruby-]2.7[.4]
[ruby-]3[.0.2]
...
```

查看《AWS Elastic Beanstalk 平台文件》中的 Ruby，以尋找 Elastic Beanstalk 平台上可用的最新 Ruby 版本。安裝該版本。

```
~$ rvm install 3.0.2
```

```
Searching for binary rubies, this might take some time.
Found remote file https://rubies.travis-ci.org/ubuntu/20.04/x86_64/ruby-3.0.2.tar.bz2
Checking requirements for ubuntu.
Updating system..
...
Requirements installation successful.
ruby-3.0.2 - #configure
ruby-3.0.2 - #download
...
```

測試您的 Ruby 安裝。

```
~$ ruby --version
ruby 3.0.2p107 (2021-07-07 revision 0db68f0233) [x86_64-linux]
```

安裝適用於 AWS SDK for Ruby

如果您需要從應用程式內管理 AWS 資源，請安裝 AWS SDK for Ruby. 例如，透過適用於 Ruby 的開發套件，您可以使用 Amazon DynamoDB (DynamoDB) 來存放使用者和工作階段資訊，無須建立關聯式資料庫。

使用 gem 命令來安裝適用於 Ruby 的開發套件及其相依性。

```
$ gem install aws-sdk
```

如需詳細資訊及安裝說明，請前往 AWS SDK for Ruby 首頁。

安裝 IDE 或文字編輯器

整合開發環境 (IDE) 提供可加速應用程式開發的各種功能。如果您還沒有使用 IDE 進行 Ruby 開發，請嘗試使用 Aptana RubyMine 並查看哪種最適合您。

- 安裝 Aptana
- RubyMine

> **Note**
>
> IDE 可能會於專案資料夾新增您不希望遞交給來源控制的檔案。欲避免將這些檔案遞交給來源控制，請使用 `.gitignore` 或等同來源控制工具的功能。

若您只想開始編碼且不需要 IDE 的所有功能，請考慮安裝 Sublime Text。

使用 Elastic Beanstalk Ruby 平台

AWS Elastic Beanstalk Ruby 平台是一組適用於 Ruby Web 應用程式的環境資訊，可在 Puma 應用程式伺服器下的 NGNIX 代理伺服器後面執行。每個平台分支都對應一個 Ruby 版本。若使用 RubyGems，您可將 `Gemfile` 檔案納入您的原始碼套件，藉此於部署期間安裝套件。

應用程式伺服器組態

Elastic Beanstalk 會根據您在建立環境時選擇的 Ruby 平台分支安裝 Puma 應用程式伺服器。如需 Ruby 平台版本提供之元件的詳細資訊，請參閱AWS Elastic Beanstalk 平台指南中的支援的平台。

您可以使用自己提供的 Puma 伺服器設定您的應用程序。這提供了一個選項，以使用 Ruby 平台分支預先安裝的 Puma 其他版本。您也可以將應用程式設定為使用不同的應用程式伺服器，例如 Passenger。若要這麼做，您必須在您的部署中包含並自訂 `Gemfile`。您還需要設定 `Procfile` 以啟動應用程式伺服器。如需更多詳細資訊，請參閱使用 Procfile 設定應用程式程序。

其他組態選項

Elastic Beanstalk 提供組態選項，可讓您用來自訂在 Elastic Beanstalk 環境中 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體上執行的軟體。您可以設定應用程式所需的環境變數，啟用至 Amazon S3 的日誌輪換，並在應用程式來源中，將包含靜態檔案的資料夾映射到代理伺服器提供的路徑。平台也會預先定義與 Rails 和 Rack 相關的部分常見環境變數，方便探索及使用。

Elastic Beanstalk 主控台中提供了修改正在執行環境組態的組態選項。要避免在終止環境的組態時遺失組態，您可以使用已儲存組態來儲存您的設定，並在之後套用至另一個環境。

若要將設定儲存於原始程式碼，您可以包含組態檔案。每次您建立環境或部署應用程式，組態檔案裡的設定就會套用。您也可以使用組態檔案來安裝套件、執行指令碼，並在部署期間執行其他執行個體自訂操作。

在 Elastic Beanstalk 主控台中套用的設定會覆寫組態檔案中相同的設定 (如存在)。這可讓您在組態檔案中擁有預設設定，並以主控台的環境專屬設定覆寫之。如需優先順序以及其他變更設定方法的詳細資訊，請參閱組態選項。

如需各種擴充 Elastic Beanstalk Linux 類型平台方式的詳細資訊，請參閱 the section called “擴充 Linux 平台”。

設定您的 Ruby 環境

您可以使用 Elastic Beanstalk 主控台來啟用至 Amazon S3 的日誌輪換，和設定您應用程式可從環境讀取的變數。

欲存取環境的軟體組態設定

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

   > **Note**
   >
   > 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇 Configuration (組態)。
4. 在更新、監控和日誌記錄組態類別中，選擇編輯。

日誌選項

Log options (日誌選項) 區段有兩個設定：

- Instance profile (執行個體描述檔) – 指定執行個體描述檔，此描述檔具備存取和您應用程式相關聯 Amazon S3 儲存貯體的許可。
- Enable log file rotation to Amazon S3 (啟用 Amazon S3 的日誌檔案輪換) – 指定是否將應用程式 Amazon EC2 執行個體的日誌檔案複製到與應用程式關聯的 Amazon S3 儲存貯體。

靜態檔案

為改善效能，您可以使用 Static files (靜態檔案) 區段來設定代理伺服器，以為來自 Web 應用程式一組目錄中的靜態檔案 (例如 HTML 或影像) 提供服務。對於每個目錄，您可以設定目錄映射的虛擬路徑。代理伺服器收到位於指定路徑下的檔案請求時，會直接提供檔案而非將請求路由至您的應用程式。

如需使用組態檔案或 Elastic Beanstalk 主控台設定靜態檔案的詳細資訊，請參閱the section called “靜態檔案”。

預設情況下，Ruby 環境中的代理伺服器會設定為提供靜態檔案如下：

- `public` 資料夾中的檔案透過 `/public` 路徑和根網域 (`/` 路徑) 提供。
- `public/assets` 子資料夾中的檔案透過 `/assets` 路徑提供。

下列範例說明預設組態的運作方式：

- 若您的應用程式原始碼在名為 public 的資料夾中有一個名為 logo.png 的檔案，則代理伺服器會透過 *subdomain*.elasticbeanstalk.com/public/logo.png 和 *subdomain*.elasticbeanstalk.com/logo.png 將此檔案提供給使用者。
- 若您的應用程式原始碼在 public 資料夾中有一個名為 assets 的資料夾，裡面含有一個名為 logo.png 的檔案，則代理伺服器會透過 *subdomain*.elasticbeanstalk.com/assets/logo.png 提供此檔案。

您可以為靜態檔案設定其他映射項目。如需詳細資訊，請參閱本主題後面部分的 Ruby 組態命名空間。

> **Note**
>
> 如果是 Ruby 2.7 AL2 3.3.7 版之前的平台版本，預設的 Elastic Beanstalk nginx 代理伺服器組態不支援透過根網域 (*subdomain*.elasticbeanstalk.com/) 提供靜態檔案。此平台版本於 2021 年 10 月 21 日推出。如需詳細資訊，請參閱AWS Elastic Beanstalk版本備註的新平台版本 - Ruby。

環境屬性

Environment Properties (環境屬性) 的部分可讓您針對執行您應用程式的 Amazon EC2 執行個體，來指定其上的環境資訊設定。環境屬性會以金鑰值對的形式傳到應用程式。

Ruby 平台定義下列環境資訊的屬性：

- BUNDLE_WITHOUT – 自 Gemfile 安裝相依項目時應忽略的群組清單，以冒號分隔。
- BUNDLER_DEPLOYMENT_MODE – 設為 true (預設值) 以使用 Bundler，在部署模式中安裝相依性。設定為 false，在開發模式中執行 bundle install。

  > **Note**
  >
  > 這個環境屬性並未在 Amazon Linux AMI Ruby 平台分支 (Amazon Linux 2 之前的分支) 上定義。

- RAILS_SKIP_ASSET_COMPILATION – 設為 true 以在部署期間跳過執行 rake assets:precompile。
- RAILS_SKIP_MIGRATIONS – 設為 true 以在部署期間跳過執行 rake db:migrate。

- RACK_ENV – 指定 Rack 的環境階段。例如，development、production 或 test。

在 Elastic Beanstalk 內所執行的 Ruby 環境中，可使用 ENV 物件來存取環境變數。例如，您可使用下列程式碼，來將名為 API_ENDPOINT 的屬性讀取到變數：

```
endpoint = ENV['API_ENDPOINT']
```

如需詳細資訊，請參閱 環境屬性與其他軟體設定。

Ruby 組態命名空間

您可以使用組態檔案來設定組態選項，並在部署期間執行其他的執行個體設定工作。組態選項可由 Elastic Beanstalk 服務或您使用的平台來定義，並且會組織成「命名空間」。

您可以使用 aws:elasticbeanstalk:environment:proxy:staticfiles 命名空間來設定環境代理以提供靜態檔案。您可以定義虛擬路徑到應用程式目錄的對應項目。

Ruby 平台不會定義任何特定於平台的命名空間。反而會定義通用 Rails 和 Rack 選項的環境屬性。

下列組態檔案會指定靜態檔案選項，將對應名為 staticimages 的目錄對應至路徑 /images，設定每個平台定義的環境屬性，以及設定名為 LOGGING 的其他環境屬性。

Example .ebextensions/ruby-settings.config

```
option_settings:
  aws:elasticbeanstalk:environment:proxy:staticfiles:
    /images: staticimages
  aws:elasticbeanstalk:application:environment:
    BUNDLE_WITHOUT: test
    BUNDLER_DEPLOYMENT_MODE: true
    RACK_ENV: development
    RAILS_SKIP_ASSET_COMPILATION: true
    RAILS_SKIP_MIGRATIONS: true
    LOGGING: debug
```

> **Note**
>
> BUNDLER_DEPLOYMENT_MODE 環境屬性和 aws:elasticbeanstalk:environment:proxy:staticfiles 命名空間並未在 Amazon Linux AMI Ruby 平台分支 (Amazon Linux 2 之前的分支) 上定義。

Elastic Beanstalk 可提供許多組態選項讓您自訂環境。除了組態檔案，您也可以使用主控台、已儲存組態、EB CLI 或 AWS CLI 來設定組態選項。如需詳細資訊，請參閱 組態選項。

使用 Gemfile 安裝套件

使用您專案來源根目錄中的 `Gemfile` 檔案，利用 RubyGems 來安裝您的應用程式所需的套件。

Example Gemfile

```
source "https://rubygems.org"
gem 'sinatra'
gem 'json'
gem 'rack-parser'
```

如 `Gemfile` 檔案存在，Elastic Beanstalk 會執行 `bundle install` 來安裝相依性。如需詳細資訊，請參閱 Bundler.io 網站上的 Gemfiles 和 Bundle 頁面。

> **Note**
>
> 除了預先安裝在 Ruby 平台上的預設版本外，您還可以使用 Puma 的其他版本。若要這麼做，請在 `Gemfile` 的項目中指定版本。您也可以使用自訂 `Gemfile` 來指定不同的應用程式伺服器，例如 Passenger。
> 對於這兩種情況，您還需要設定 `Procfile` 以啟動應用程式伺服器。
> 如需更多詳細資訊，請參閱使用 Procfile 設定應用程式程序。

使用 Procfile 設定應用程式程序

欲指定啟動 Ruby 應用程式的命令，請於原始碼套件的根目錄，納入名為 `Procfile` 的檔案。

> **Note**
>
> Elastic Beanstalk 在 Amazon Linux AMI Ruby 平台分支 (Amazon Linux 2 之前的分支) 上不支援此功能。含有 with Puma 或 with Passenger 名稱的平台分支，不論其 Ruby 版本為何，都會優先於 Amazon Linux 2 且不支援 `Procfile` 功能。

如需有關撰寫和使用 `Procfile` 的詳細資料，請展開the section called “擴充 Linux 平台”中的 Buildfile 和 Procfile 區段。

當您沒有提供 `Procfile` 時，Elastic Beanstalk 會產生以下預設檔案，此檔案會假設您使用的是預先安裝的 Puma 應用程式伺服器。

```
web: puma -C /opt/elasticbeanstalk/config/private/pumaconf.rb
```

如果你想使用自己提供的 Puma 伺服器，你可以使用 Gemfile 來進行安裝。下列範例 `Procfile` 將說明如何開始。

Example Procfile

```
web: bundle exec puma -C /opt/elasticbeanstalk/config/private/pumaconf.rb
```

如果你想使用 Passenger 應用程式伺服器，請使用下面的範例檔案，設定 Ruby 環境來安裝和使用 Passenger。

1. 使用此範例檔案安裝 Passenger。

   Example Gemfile

   ```
   source 'https://rubygems.org'
   gem 'passenger'
   ```

2. 使用此範例檔案，指示 Elastic Beanstalk 啟動 Passenger。

   Example Procfile

   ```
   web: bundle exec passenger start /var/app/current --socket /var/run/puma/my_app.sock
   ```

> **Note**
>
> 您不必更改 nginx 代理伺服器組態中的任何內容，即可使用 Passenger。要使用其他應用程序伺服器，您可能需要自訂 nginx 組態，以適當地將請求轉送到應用程式。

將 Rails 應用程式部署到 Elastic Beanstalk

Rails 是一個開源的， model-view-controller （ MVC ）框架的紅寶石。本教程將引導您完成生成 Rails 應用程序並將其部署到 AWS Elastic Beanstalk 環境的過程。

章節

必要條件

基本 Elastic Beanstalk 知識

本教學假設您具備基本的 Elastic Beanstalk 操作及 Elastic Beanstalk 主控台知識。若您尚不了解，請依照 開始使用 Elastic Beanstalk 中的說明來啟動您的第一個 Elastic Beanstalk 環境。

命令列

為了遵循本指南的程序，您需要命令列終端機或 shell 來執行命令。命令清單前會出現提示字元 ($) 及目前的目錄名稱 (如有)。

```
~/eb-project$ this is a command
this is output
```

在 Linux 和 macOS 上，您可以使用偏好的 Shell 和套件軟體管理工具。在視窗上，你可以安裝視窗子系統為 Linux 得到一個視窗集成的 Ubuntu 和 Bash 的版本。

Rails 相依性

Rails 架構 6.1.4.1 具有下列相依性。請確定您已安裝所有項目。

- Ruby 2.5.0 或更新版本 – 如需安裝說明，請參閱 設定您的 Ruby 開發環境。

  本教學課程中，我們使用 Ruby 3.0.2 和對應的 Elastic Beanstalk 平台版本。
- Node.js – 如需安裝說明，請參閱透過套件管理員安裝 Node.js。
- Yarn - 如需安裝說明，請參閱 Yarn 網站上的 Installation。

啟動 Elastic Beanstalk 環境

使用 Elastic Beanstalk 主控台建立 Elastic Beanstalk 環境。選擇 Ruby (Ruby) 平台，並接受預設的設定和範本程式碼。

啟動環境 (主控台)

1. 使用這個預先配置的鏈接打開 Elastic Beanstalk 控制台：控制台。aws.amazon.com/彈性豆/家 #/新 applicationName = 教程和 environmentType = LoadBalanced
2. 在 Platform (平台)，選取符合您應用程式所使用語言的平台和平台分支。
3. 針對 Application code (應用程式程式碼)，選擇 Sample application (範例應用程式)。
4. 選擇 Review and launch (檢閱和啟動)。
5. 檢視可用選項。選擇您要使用的可用選項，當您準備就緒時，請選擇 Creat app (建立應用程式)。

使用大約需要五分鐘時間建立環境，並且建立下列資源：

- EC2 執行個體 – Amazon Elastic Compute Cloud (Amazon EC2) 虛擬機器，已設為在您選擇的平台上執行 Web 應用程式。

  每個平台會執行特定的一套軟體、設定檔和指令碼，來支援特定的語言版本、架構、Web 容器或其組合。大多數的平台使用會 Apache 或 NGINX 做為反向代理，此反向代理會在您 Web 應用程式的前景執行、轉傳遞交給此 Web 應用程式的請求、提供靜態資產，並產生存取和錯誤日誌。
- 執行個體安全群組 - Amazon EC2 安全群組，已設為允許從連接埠 80 傳入的流量。此資源可讓負載平衡器傳來的 HTTP 傳輸資料，到達執行您 Web 應用程式的 EC2 執行個體。在預設情況下，不允許傳輸資料從其他通訊埠傳送。
- 負載平衡器 - Elastic Load Balancing 負載平衡器，可設定將請求分配到執行您應用程式的執行個體。負載平衡器也讓您的執行個體不需直接連接到網際網路。
- 負載平衡器安全群組 - Amazon EC2 安全群組，設為允許從連接埠 80 傳入的流量。此資源可讓來自網際網路的 HTTP 傳輸資料到達負載平衡器。在預設情況下，不允許傳輸資料從其他通訊埠傳送。
- Auto Scaling 群組 - Auto Scaling 群組，設為在執行個體終止或無法使用時，取代該執行個體。
- Amazon S3 儲存貯體 - 儲存位置，用來儲存當您使用 Elastic Beanstalk 時所建立的原始程式碼、日誌和其他成品。
- Amazon CloudWatch 警示 — 監控環境中執行個體負載的兩個 CloudWatch 警示，並在負載過高或過低時觸發警示。當警示觸發時，您的 Auto Scaling 群組會擴展或縮減以進行回應。
- AWS CloudFormation 堆疊 — Elastic Beanstalk 用 AWS CloudFormation 來啟動環境中的資源並傳播組態變更。資源定義於範本中，您可在 AWS CloudFormation 主控台中檢視此範本。

- 網域名稱 – 會路由到您 Web 應用程式的網域名稱，其格式為 *subdomain*.*region*.elasticbeanstalk.com。

> **Note**
>
> 為了增強 Elastic Beanstalk 應用程式的安全性，我們會在公共后缀列表 (PSL) 中註冊網域 elasticbeanstalk.com。為了加強安全性，如果您需要在 Elastic Beanstalk 應用程式的預設網域名稱中設定敏感性 Cookie，我們建議您使用具 `__Host-` 前置詞的 Cookie。此做法將有助於保護您的網域免受跨站請求偽造 (CSRF) 攻擊。如需更多資訊，請參閱 Mozilla 開發人員網路中的設定 Cookie 頁面。

這些資源全都由 Elastic Beanstalk 管理。當您終止環境時，Elastic Beanstalk 會終止其中的所有資源。

> **Note**
>
> Elastic Beanstalk 建立的 Amazon S3 儲存貯體會在環境間共享，且不會在環境終止時刪除。如需詳細資訊，請參閱 將 Elastic Beanstalk 與 Amazon S3 搭配使用。

安裝 Rails 並產生網站

使用 `gem` 命令來安裝 Rails 及其相依性。

```
~$ gem install rails
Fetching: concurrent-ruby-1.1.9.gem
Successfully installed concurrent-ruby-1.1.9
Fetching: rack-2.2.3.gem
Successfully installed rack-2.2.3
...
```

測試您的 Rails 安裝。

```
~$ rails --version
Rails 6.1.4.1
```

使用 `rails new` 與應用程式的名稱，來建立新的 Rails 專案。

```
~$ rails new ~/eb-rails
```

Rails 會使用指定的名稱來建立目錄、產生在本機執行範例專案需要的所有檔案，然後執行 Bundler 來安裝專案的 Gemfile 中定義的所有相依項目 (Gems)。

> **Note**
>
> 此過程將為該專案安裝最新的 Puma 版本。此版本可能與 Elastic Beanstalk 在您的環境的 Ruby 平台版本上提供的版本不同。如需查看 Elastic Beanstalk 提供的 Puma 版本，請參閱《AWS Elastic Beanstalk 平台指南》中的 Ruby 平台歷史記錄。如需最新 Puma 版本的詳細資訊，請參閱 Puma.io 網站。如果兩個 Puma 版本不同，請使用以下選項之一：
>
> - 使用之前 `rails new` 命令安裝的 Puma 版本。在這種情況下，您必須新增 `Procfile` 供平台使用，這樣平台才能使用自己提供的 Puma 伺服器版本。如需詳細資訊，請參閱 使用 Procfile 設定應用程式程序。
> - 更新 Puma 版本，使其與您環境的 Ruby 平台版本上預先安裝的版本相同。若要這麼做，請在專案來源根目錄修改 Gemfile 中的 Puma 版本。然後執行 `bundle update`。如需詳細資訊，請參閱 Bundler.io 網站上的套件更新頁面。

在本機上執行預設的專案，以測試您的 Rails 安裝。

```
~$ cd eb-rails
~/eb-rails$ rails server
=> Booting Puma
=> Rails 6.1.4.1 application starting in development
=> Run `bin/rails server --help` for more startup options
Puma starting in single mode...
* Puma version: 5.5.2 (ruby 3.0.2-p107) ("Zawgyi")
*  Min threads: 5
*  Max threads: 5
*  Environment: development
*          PID: 77857
* Listening on http://127.0.0.1:3000
* Listening on http://[::1]:3000
Use Ctrl-C to stop
...
```

在 web 瀏覽器中開啟 `http://localhost:3000`，查看活動中的預設專案。

此頁面僅可於開發模式中顯示。新增一些內容到應用程式的首頁來支援對 Elastic Beanstalk 的生產部署。使用 `rails generate` 來建立控制器、路徑和您歡迎頁面的檢視。

```
~/eb-rails$ rails generate controller WelcomePage welcome
      create  app/controllers/welcome_page_controller.rb
       route  get 'welcome_page/welcome'
      invoke  erb
      create    app/views/welcome_page
      create    app/views/welcome_page/welcome.html.erb
      invoke  test_unit
      create    test/controllers/welcome_page_controller_test.rb
      invoke  helper
      create    app/helpers/welcome_page_helper.rb
      invoke    test_unit
      invoke  assets
      invoke    coffee
      create      app/assets/javascripts/welcome_page.coffee
      invoke    scss
```

```
      create      app/assets/stylesheets/welcome_page.scss.
```

可將給予您存取 /welcome_page/welcome 頁面所需的所有權限。不過，在您發佈變更之前，請在檢視中變更內容，並新增路徑，來讓此頁面出現在網站最頂端的層級中。

使用文字編輯器來編輯 app/views/welcome_page/welcome.html.erb 中的內容。在此範例中，您將使用 cat 來直接覆寫現有檔案的內容。

Example app/views/welcome_page/welcome.html.erb

```
<h1>Welcome!</h1>
<p>This is the front page of my first Rails application on Elastic Beanstalk.</p>
```

最後，將下列的路徑加入 config/routes.rb：

Example config/routes.rb

```
Rails.application.routes.draw do
  get 'welcome_page/welcome'
  root 'welcome_page#welcome'
```

這會讓 Rails 將請求轉傳到網站的根目錄，傳給歡迎頁面控制器的歡迎方法，此方法會以歡迎視圖 (welcome.html.erb) 呈現內容。

我們需要更新 Gemfile.lock，讓 Elastic Beanstalk 能在 Ruby 平台上成功部署應用程式。Gemfile.lock 的某些相依性可能僅限於平台。因此，我們需要將 **platform ruby** 新增至 Gemfile.lock，以便在部署過程中一併安裝所有必要的相依性。

Example

```
~/eb-rails$ bundle lock --add-platform ruby
Fetching gem metadata from https://rubygems.org/.....
Resolving dependencies...
Writing lockfile to /Users/janedoe/EBDPT/RubyApps/eb-rails-doc-app/Gemfile.lock
```

配置 Rails 設定

使用 Elastic Beanstalk 主控台來設定 Rails 的環境屬性。將 SECRET_KEY_BASE 環境屬性設定成最多 256 個英數字元的字串。

Rails 使用此屬性來建立金鑰。因此，您應該將它保持私密，並且不要在來源控制中以純文字格式存放它。而是會在您的環境上利用環境屬性將它提供給 Rails 程式碼。

在 Elastic Beanstalk 主控台中設定環境屬性

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇組態。
4. 在更新、監控和日誌記錄組態類別中，選擇編輯。
5. 向下捲動至環境屬性。
6. 選取新增環境屬性。
7. 輸入屬性名稱和值對。
8. 如果需要新增更多變數，請重複步驟 6 和步驟 7。
9. 若要儲存變更，請選擇頁面底部的儲存變更。

現在您準備好部署網站到環境。

部署您的應用程式

建立原始碼套件，其中包含 Rails 所建立的檔案。以下命令建立一個名為 `rails-default.zip` 的原始碼套件。

```
~/eb-rails$ zip ../rails-default.zip -r * .[^.]*
```

將來源套件上傳至 Elastic Beanstalk，以將 Rails 部署到您的環境。

若要部署原始碼套件

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在環境概觀頁面上，選擇 Upload and deploy (上傳和部署)。
4. 使用畫面顯示對話方塊來上傳原始碼套件。
5. 選擇 Deploy (部署)。
6. 部署完成後，您可選擇網站的 URL 以在新分頁中開啟您的網站。

清除

當您完成使用 Elastic Beanstalk 時，即可終止您的環境。Elastic Beanstalk 會終止與您的環境相關聯的所有 AWS 資源，例如 Amazon EC2 執行個體、資料庫執行個體、負載平衡器、安全群組和警示。

從控制台終止 Elastic Beanstalk 環境

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 選擇 Actions (動作)，然後選擇 Terminate Environment (終止環境)。
4. 使用畫面顯示對話方塊來確認環境終止。

您可以使用 Elastic Beanstalk，隨時輕鬆地為您的應用程式建立新環境。

後續步驟

如需 Rails 的詳細資訊，請瀏覽 rubyonrails.org。

隨著您繼續開發應用程式，您可能會希望無須手動建立 .zip 檔案並將其上傳至 Elastic Beanstalk 主控台，即可管理環境和部署應用程式。E lastic Beanstalk 命令列介面 (EB CLI) 提供從命令列建立、設定和部署應用程式至 Elastic Beanstalk 環境的 easy-to-use 指令。

最後，若您打算於生產環境中使用您的應用程式，建議您設定您環境的自訂網域名稱，並啟用 HTTPS 安全連線。

將 Sinatra 應用程式部署到 Elastic Beanstalk

此說明會逐步引導您將簡易的 Sinatra web 應用程式部署至 AWS Elastic Beanstalk。

必要條件

本教學假設您具備基本的 Elastic Beanstalk 操作及 Elastic Beanstalk 主控台知識。若您尚不了解，請依照 開始使用 Elastic Beanstalk 中的說明來啟動您的第一個 Elastic Beanstalk 環境。

為了遵循本指南的程序，您需要命令列終端機或 shell 來執行命令。命令清單前會出現提示字元 ($) 及目前的目錄名稱 (如有)。

```
~/eb-project$ this is a command
this is output
```

在 Linux 和 macOS 上，您可以使用偏好的 Shell 和套件軟體管理工具。在視窗上，你可以安裝視窗子系統為 Linux 得到一個視窗集成的 Ubuntu 和 Bash 的版本。

Sinatra 2.1.0 需要 Ruby 2.3.0 或更新版本。本教學課程中，我們使用 Ruby 3.0.2 和對應的 Elastic Beanstalk 平台版本。遵循 設定您的 Ruby 開發環境 的指示安裝 Ruby。

啟動 Elastic Beanstalk 環境

使用 Elastic Beanstalk 主控台建立 Elastic Beanstalk 環境。選擇 Ruby (Ruby) 平台，並接受預設的設定和範本程式碼。

啟動環境 (主控台)

1. 使用這個預先配置的鏈接打開 Elastic Beanstalk 控制台：控制台。aws.amazon.com/彈性豆/家 # /新 applicationName = 教程和 environmentType = LoadBalanced
2. 在 Platform (平台)，選取符合您應用程式所使用語言的平台和平台分支。
3. 針對 Application code (應用程式程式碼)，選擇 Sample application (範例應用程式)。
4. 選擇 Review and launch (檢閱和啟動)。
5. 檢視可用選項。選擇您要使用的可用選項，當您準備就緒時，請選擇 Creat app (建立應用程式)。

使用大約需要五分鐘時間建立環境，並且建立下列資源：

- EC2 執行個體 – Amazon Elastic Compute Cloud (Amazon EC2) 虛擬機器，已設為在您選擇的平台上執行 Web 應用程式。

  每個平台會執行特定的一套軟體、設定檔和指令碼，來支援特定的語言版本、架構、Web 容器或其組合。大多數的平台使用會 Apache 或 NGINX 做為反向代理，此反向代理會在您 Web 應用程式的前景執行、轉傳遞交給此 Web 應用程式的請求、提供靜態資產，並產生存取和錯誤日誌。
- 執行個體安全群組 - Amazon EC2 安全群組，已設為允許從連接埠 80 傳入的流量。此資源可讓負載平衡器傳來的 HTTP 傳輸資料，到達執行您 Web 應用程式的 EC2 執行個體。在預設情況下，不允許傳輸資料從其他通訊埠傳送。
- 負載平衡器 - Elastic Load Balancing 負載平衡器，可設定將請求分配到執行您應用程式的執行個體。負載平衡器也讓您的執行個體不需直接連接到網際網路。
- 負載平衡器安全群組 - Amazon EC2 安全群組，設為允許從連接埠 80 傳入的流量。此資源可讓來自網際網路的 HTTP 傳輸資料到達負載平衡器。在預設情況下，不允許傳輸資料從其他通訊埠傳送。
- Auto Scaling 群組 - Auto Scaling 群組，設為在執行個體終止或無法使用時，取代該執行個體。
- Amazon S3 儲存貯體 - 儲存位置，用來儲存當您使用 Elastic Beanstalk 時所建立的原始程式碼、日誌和其他成品。
- Amazon CloudWatch 警示 — 兩個 CloudWatch 警示，用於監控環境中執行個體的負載，並在負載過高或過低時觸發警示。當警示觸發時，您的 Auto Scaling 群組會擴展或縮減以進行回應。
- AWS CloudFormation 堆疊 — Elastic Beanstalk 用 AWS CloudFormation 來啟動環境中的資源並傳播組態變更。資源定義於範本中，您可在 AWS CloudFormation 主控台中檢視此範本。
- 網域名稱 – 會路由到您 Web 應用程式的網域名稱，其格式為 *subdomain*.*region*.elasticbeanstalk.com。

> **Note**
>
> 為了增強 Elastic Beanstalk 應用程式的安全性，我們會在公共后缀列表 (PSL) 中註冊網域 elasticbeanstalk.com。為了加強安全性，如果您需要在 Elastic Beanstalk 應用程式的預設網域名稱中設定敏感性 Cookie，我們建議您使用具 `__Host-` 前置詞的 Cookie。此做法將有助於保護您的網域免受跨站請求偽造 (CSRF) 攻擊。如需更多資訊，請參閱 Mozilla 開發人員網路中的設定 Cookie 頁面。

這些資源全都由 Elastic Beanstalk 管理。當您終止環境時，Elastic Beanstalk 會終止其中的所有資源。

> **Note**
>
> Elastic Beanstalk 建立的 Amazon S3 儲存貯體會在環境間共享，且不會在環境終止時刪除。如需詳細資訊，請參閱 將 Elastic Beanstalk 與 Amazon S3 搭配使用。

寫入基本 Sinatra 網站

欲建立並部署 Sinatra 應用程式

1. 使用下列內容，建立名為 config.ru 的組態檔案。

   Example config.ru

   ```
   require './helloworld'
   run Sinatra::Application
   ```

2. 使用下列內容，建立名為 helloworld.rb 的 Ruby 程式碼檔案。

   Example helloworld.rb

   ```
   require 'sinatra'
   get '/' do
     "Hello World!"
   end
   ```

3. 使用下列內容建立 Gemfile。

   Example Gemfile

   ```
   source 'https://rubygems.org'
   gem 'sinatra'
   gem 'puma'
   ```

4. 執行套件安裝作業以產生 `Gemfile.lock`

   Example

   ```
   ~/eb-sinatra$ bundle install
   Fetching gem metadata from https://rubygems.org/....
   Resolving dependencies...
   Using bundler 2.2.22
   ```

```
Using rack 2.2.3
...
```

5. 我們需要更新 Gemfile.lock，讓 Elastic Beanstalk 能在 Ruby 平台上成功部署應用程式。Gemfile.lock 的某些相依性可能僅限於平台。因此，我們需要將 **platform ruby** 新增至 Gemfile.lock，以便在部署過程中一併安裝所有必要的相依性。

   Example

   ```
   ~/eb-sinatra$ bundle lock --add-platform ruby
   Fetching gem metadata from https://rubygems.org/....
   Resolving dependencies...
   Writing lockfile to /Users/janedoe/EBDPT/RubyApps/eb-sinatra/Gemfile.lock
   ```

6. 使用下列內容建立 Procfile。

   Example Procfile

   ```
   web: bundle exec puma -C /opt/elasticbeanstalk/config/private/pumaconf.rb
   ```

部署您的應用程式

建立原始碼套件，其中包含您的原始碼。以下命令建立一個名為 sinatra-default.zip 的原始碼套件。

```
~/eb-sinatra$ zip ../sinatra-default.zip -r * .[^.]*
```

將來源套件上傳至 Elastic Beanstalk，以將 Sinatra 部署到您的環境。

若要部署原始碼套件

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

   > **Note**
   >
   > 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在環境概觀頁面上，選擇 Upload and deploy (上傳和部署)。

4. 使用畫面顯示對話方塊來上傳原始碼套件。
5. 選擇 Deploy (部署)。
6. 部署完成後，您可選擇網站的 URL 以在新分頁中開啟您的網站。

清除

當您完成使用 Elastic Beanstalk 時，即可終止您的環境。Elastic Beanstalk 會終止與您的環境相關聯的所有 AWS 資源，例如 Amazon EC2 執行個體、資料庫執行個體、負載平衡器、安全群組和警示。

從控制台終止 Elastic Beanstalk 環境

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 選擇 Actions (動作)，然後選擇 Terminate Environment (終止環境)。
4. 使用畫面顯示對話方塊來確認環境終止。

您可以使用 Elastic Beanstalk，隨時輕鬆地為您的應用程式建立新環境。

後續步驟

如需 Sinatra 的詳細資訊，請瀏覽 sinatrarb.com。

隨著您繼續開發應用程式，您可能會希望無須手動建立 .zip 檔案並將其上傳至 Elastic Beanstalk 主控台，即可管理環境和部署應用程式。E lastic Beanstalk 命令列介面 (EB CLI) 提供從命令列建立、設定和部署應用程式至 Elastic Beanstalk 環境的 easy-to-use 指令。

最後，若您打算於生產環境中使用您的應用程式，建議您設定您環境的自訂網域名稱，並啟用 HTTPS 安全連線。

將 Amazon RDS 資料庫執行個體新增到您的 Ruby 應用程式環境

您可以使用 Amazon Relational Database Service (Amazon RDS) 資料庫執行個體存放由應用程式收集與修改的資料。資料庫可與環境耦合並由 Elastic Beanstalk 管理，或者由另一項服務在外部分開建立與管理。本主題提供了使用 Elastic Beanstalk 主控台建立 Amazon RDS 的說明。資料庫將與環境耦

合並由 Elastic Beanstalk 管理。如需將 Amazon RDS 與 Elastic Beanstalk 整合的相關詳細資訊，請參閱 將資料庫新增至您的 Elastic Beanstalk 環境。

章節

- 將資料庫執行個體新增到您的環境
- 下載轉接器
- 連線至資料庫

將資料庫執行個體新增到您的環境

欲將資料庫執行個體新增到您的環境

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

   > **Note**
   >
   > 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇 Configuration (組態)。
4. 在 Database (資料庫) 組態類別中，選擇 Edit (編輯)。
5. 選擇資料庫引擎，並輸入使用者名稱和密碼。
6. 若要儲存變更，請選擇頁面底部的儲存變更。

新增資料庫執行個體約需要 10 分鐘。環境更新完成時，資料庫執行個體的主機名稱和其他連線資訊會透過下列環境屬性提供給您的應用程式：

屬性名稱	描述	屬性值
RDS_HOSTNAME	資料庫執行個體的主機名稱。	在 Amazon RDS 主控台：端點的連線能力和安全性索引標籤上。
RDS_PORT	資料庫執行個體接受連線的連接埠。預設值在不同資料庫引擎中有所差異。	在 Amazon RDS 主控台：連接埠的連線能力和安全性索引標籤上。

屬性名稱	描述	屬性值
RDS_DB_NAME	資料庫名稱，**ebdb**。	在 Amazon RDS 主控台：資料庫名稱的組態索引標籤上。
RDS_USERNAME	您為資料庫設定的使用者名稱。	在 Amazon RDS 主控台：主要使用者名稱的組態索引標籤上。
RDS_PASSWORD	您為資料庫設定的密碼。	無法在 Amazon RDS 主控台中提供參考。

如需設定與 Elastic Beanstalk 環境耦合之資料庫執行個體的相關詳細資訊，請參閱 將資料庫新增至您的 Elastic Beanstalk 環境。

下載轉接器

將資料庫轉接器新增到您專案的 gem 檔案中。

Example Gemfile – 具備 MySQL 的 Rails

```
source 'https://rubygems.org'
gem 'puma'
gem 'rails', '~> 6.1.4', '>= 6.1.4.1'
gem 'mysql2'
```

Ruby 的常見轉接器 Gems

- MySQL – mysql2
- PostgreSQL – pg
- Oracle – activerecord-oracle_enhanced-adapter
- SQL Server – activerecord-sqlserver-adapter

連線至資料庫

Elastic Beanstalk 會在環境屬性中提供已連接的資料庫執行個體連線資訊。使用 `ENV['VARIABLE']` 來讀取屬性和設定資料庫的連線。

Example config/database.yml – Ruby on Rails 資料庫組態 (MySQL)

```
production:
  adapter: mysql2
  encoding: utf8
  database: <%= ENV['RDS_DB_NAME'] %>
  username: <%= ENV['RDS_USERNAME'] %>
  password: <%= ENV['RDS_PASSWORD'] %>
  host: <%= ENV['RDS_HOSTNAME'] %>
  port: <%= ENV['RDS_PORT'] %>
```

教學和範例

語言和框架特定的教程分佈在 AWS Elastic Beanstalk 開發人員指南中。全新及更新後的教學課程，會在發佈後新增至此清單。最近更新的列於最上方。

這些教學課程針對中級使用者而設計，可能不會納入基本步驟的說明，例如註冊 AWS的說明。如果這是您第一次使用 AWS 或 Elastic Beanstalk，請查看入門逐步解說，讓您的第一個 Elastic Beanstalk 環境啟動並執行。

- Ruby on Rails - 將 Rails 應用程式部署到 Elastic Beanstalk
- Ruby 和 Sinatra - 將 Sinatra 應用程式部署到 Elastic Beanstalk
- PHP 和 MySQL HA 組態 - 使用外部 Amazon RDS 資料庫將高可用性 PHP 應用程式資料庫部署至 Elastic Beanstalk
- PHP 和 Laravel - 將 Laravel 應用程式部署至 Elastic Beanstalk
- PHP 和 CakePHP - 將 CakePHP 應用程式部署至 Elastic Beanstalk
- PHP 和 Drupal HA 組態 - 使用外部 Amazon RDS 資料庫將高可用性 Drupal 網站資料庫部署至 Elastic Beanstalk
- PHP 和 WordPress HA 配置-將具有外部 Amazon RDS 資料庫的高可用性 WordPress 網站部署到 Elastic Beanstalk
- Node.js 和 DynamoDB HA 組態 - 將 Node.js 應用程式與 DynamoDB 部署到 Elastic Beanstalk
- ASP.NET Core - 教學課程：使用 Elastic Beanstalk 部署 ASP.NET 核心應用程式
- Python 和 Flask - 將 Flask 應用程式部署至 Elastic Beanstalk
- Python 和 Django - 將 Django 應用程式部署至 Elastic Beanstalk
- Node.js 和 Express - 將 Express 應用程式部署至 Elastic Beanstalk
- Docker、PHP 和 nginx - 使用 Elastic Beanstalk 主控台來建置 ECS 受管 Docker 環境

當您透過下列連結建立環境時，您可以下載使用 Elastic Beanstalk 的範例應用程式，無須提供原始碼套件：

- Docker – docker.zip
- 多容器泊塢視窗 — 2.zip docker-multicontainer-v
- 預配置碼頭工人（玻璃魚）-1.zip docker-glassfish-v
- Go – go.zip

- Corretto – corretto.zip
- Tomcat – tomcat.zip
- dotnet-core-linux. NET 核心
- . NET 核心-dotnet-asp-windows. 郵編
- Node.js – nodejs.zip
- PHP – php.zip
- Python – python.zip
- Ruby – ruby.zip

更多參與的示例應用程序顯示使用額外的 Web 框架，庫和工具可作為開源項目提供 GitHub：

- 負載平衡 WordPress (教學課程) — 用於在負載平衡的 Elastic Beanstalk 環境中 WordPress 安全安裝和執行的組態檔案。
- 負載平衡的 Drupal (教學) – 用於安全安裝 Drupal 並在負載平衡 Elastic Beanstalk 環境內執行的組態檔案。
- 記分保持-RESTful Web API，它使用 Spring 框架，並提供用於創建和管理用戶，會話和遊戲的接口。 AWS SDK for Java 此 API 隨附 Angular 1.5 Web 應用程式，可透過 HTTP 使用該 API。包括顯示與 Amazon Cognito 和 Amazon Relational Database Service 服務整合的分支機構。 AWS X-Ray

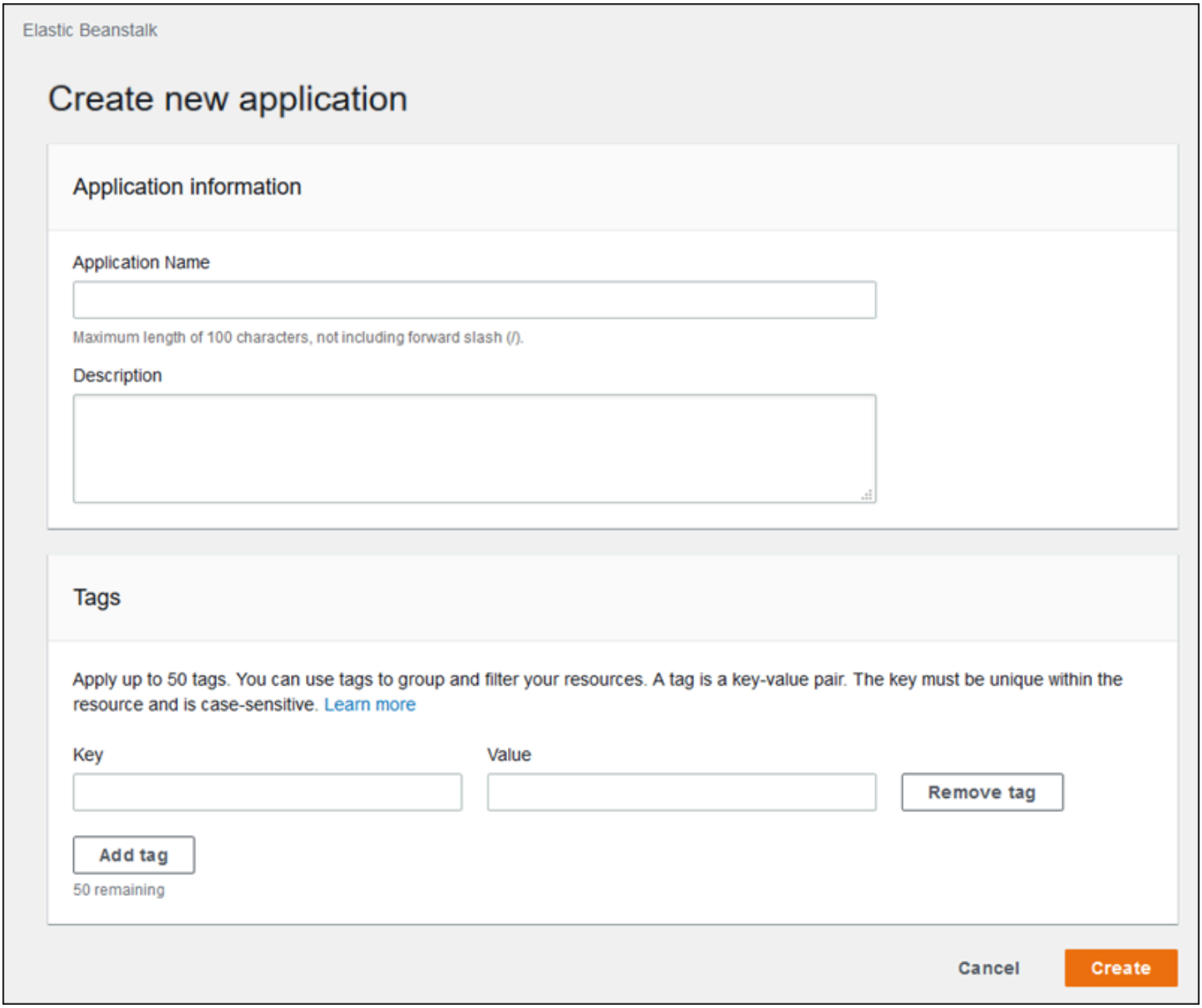
  此應用程式使用 Java SE 平台的功能，來下載相依項目和建置啟動執行個體，將原始碼套件的檔案大小減到最小。此應用程式亦包含 nginx 組態檔案，可覆寫預設組態，藉由代理的連接埠 80 供前端 Web 應用程式靜態使用，而路由則會要求 `/api` 底下運作於 `localhost:5000` 的 API 的路徑。
- 它有蛇嗎？ -顯示在 Elastic Beanstalk 的 Java EE Web 應用程式中使用 RDS 的 Tomcat 應用程式。此專案說明如何使用 Servlet、JSP、Simple Tag Support、Tag File、JDBC、SQL、Log4J、Bootstrap、Jackson 和 Elastic Beanstalk 組態檔案。
- Locust Load Generator - 此專案說明如何使用 Java SE 平台功能來安裝並執行以 Python 撰寫而成的負載產生工具 Locust。專案包含安裝及設定 Locust 的組態檔案、可設定 DynamoDB 資料表的建置指令碼及執行 Locust 的 Procfile。
- 分享您的想法 (教學) - 說明如何於 Amazon RDS 上的 MySQL、Composer 和組態檔案使用 MySQL 的 PHP 應用程式。
- 新的啟動 (教學課程)-Node.js 範例應用程式，顯示 DynamoDB 的使用方式、Node.js JavaScript 中的 AWS SDK、npm 套件管理和設定檔。

管理和設定 Elastic Beanstalk 應用程式

使用 AWS Elastic Beanstalk 的第一步是建立應用程式，這就是您在 AWS 中的 Web 應用程式。在 Elastic Beanstalk 中，應用程式可做為容器，用於容納執行 Web 應用程式的環境、Web 應用程式原始碼版本、已儲存組態、日誌和其他您使用 Elastic Beanstalk 時建立的成品。

建立應用程式

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Applications (應用程式)，然後選擇 Create application (建立應用程式)。
3. 使用螢幕上的表單提供應用程式名稱。
4. 您可選擇性地提供描述，並新增標籤索引鍵和值。
5. 選擇 Create (建立)。



建立應用程式後，主控台會提示您為其建立環境。如需所有可用選項的詳細資訊，請參閱[建立 Elastic Beanstalk 環境]。

您不再需要某個應用程式，您可以刪除它。

> **Warning**
>
> 刪除應用程式會終止所有相關聯的環境，而且會刪除屬於該應用程式的所有應用程式版本和已儲存組態。

欲刪除應用程式

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Applications (應用程式)，然後在清單中選取應用程式。
3. 選擇 Actions (動作)，然後選擇 Delete application (刪除應用程式)。

主題

Elastic Beanstalk 應用程式管理主控台

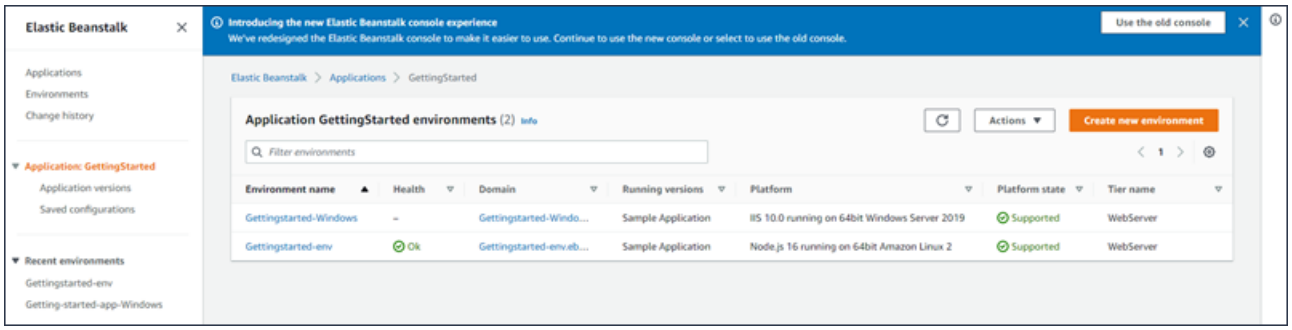
您可以使用 AWS Elastic Beanstalk 主控台來管理應用程式、應用程式版本和儲存的組態。

若要使用應用程式管理主控台

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Applications (應用程式)，然後在清單上選取應用程式名稱。

> **Note**
>
> 如果您有許多應用程式，請使用搜尋列來篩選應用程式清單。

應用程式概觀頁面會顯示一份清單，其中包含與應用程式相關之所有環境的概觀。



3. 您有幾種方法可以繼續：

a. 選擇 Actions (動作) 下拉式功能表，然後選擇其中一個應用程式管理動作。若要在此應用程式中啟動環境，您可以直接選擇 Create a new environment (建立新環境)。如需詳細資訊，請參閱the section called “建立環境”。

b. 選擇環境名稱，前往該環境的環境管理主控台，您可在其中設定、監控或管理此環境。

c. 選擇導覽窗格中應用程式名稱後的 Application versions (應用程式版本)，以檢視和管理應用程式的應用程式版本。

   應用程式版本是您應用程式程式碼的上傳版本。您可上傳新版本、將現有的版本部署到應用程式的任何環境，或刪除舊版本。如需更多詳細資訊，請參閱 管理應用程式版本。

d. 選擇導覽窗格中應用程式名稱後 Saved configurations (已儲存的組態)，以檢視和管理從執行環境儲存的組態。

   儲存的組態是一組設定，可用來將環境的設定恢復到之前的狀態，或建立具有相同設定的環境。如需詳細資訊，請參閱使用 Elastic Beanstalk 已儲存組態。

管理應用程式版本

只要您上傳原始程式碼，Elastic Beanstalk 就會建立一個應用程式版本。當您使用環境管理主控台或 EB CLI 建立環境或上傳並部署程式碼時，通常就會建立應用程式版本。Elastic Beanstalk 會根據應用程式的生命週期政策並在您刪除應用程式時刪除這些應用程式版本。如需關於應用程式生命週期政策的詳細資訊，請參閱 進行應用程式版本生命週期的設定。

您亦可上傳原始碼套件，且無須於應用程式管理主控台或使用 EB CLI 命令 eb appversion 進行部署。Elastic Beanstalk 將原始碼套件存放於 Amazon Simple Storage Service (Amazon S3)，且不會自動刪除這些套件。

您可以在建立時，將標籤套用至應用程式版本，並編輯現有應用程式版本的標籤。如需詳細資訊，請參閱標記應用程式版本。

欲建立新的應用程式版本

您也可以使用 EB CLI 建立新的應用程式版本。如需詳細資訊，請參閱 EB CLI 命令一章中的 eb appversion。

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Applications (應用程式)，然後在清單上選取應用程式名稱。

> **Note**
> 如果您有許多應用程式，請使用搜尋列來篩選應用程式清單。

3. 在導覽窗格中，找到應用程式名稱並選擇 Application versions (應用程式版本)。
4. 選擇 Upload (上傳)。使用畫面顯示表單來上傳應用程式的原始碼套件。

> **Note**
> 原始碼套件的檔案大小上限為 500 MB。

5. 您可選擇性地提供簡短描述，並新增標籤索引鍵和值。
6. 選擇 Upload (上傳)。

您指定的檔案會與您的應用程式建立關聯。您可將應用程式版本部署至新的或現有環境。

您的應用程式將隨時間累積許多應用程式版本。為了節省儲存空間並避免達到應用程式版本配額，建議您刪除不再需要的應用程式版本。

> **Note**
> 刪除應用程式版本不會影響目前執行該版本的環境。

欲刪除應用程式版本

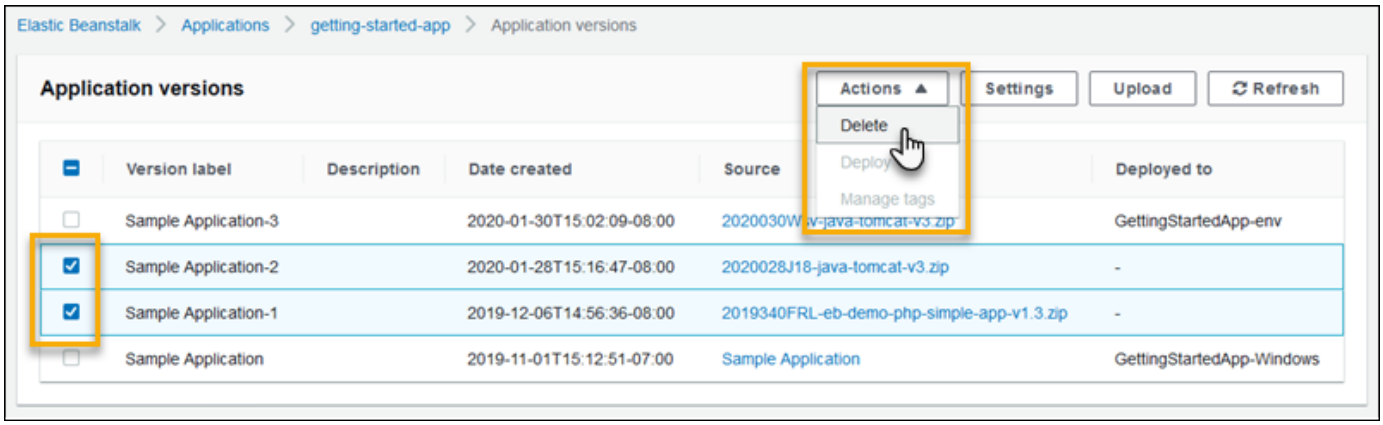
您也可以使用 EB CLI 刪除應用程式版本。如需詳細資訊，請參閱 EB CLI 命令一章中的 eb appversion。

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Applications (應用程式)，然後在清單上選取應用程式名稱。

> **Note**
> 如果您有許多應用程式，請使用搜尋列來篩選應用程式清單。

3. 在導覽窗格中，找到應用程式名稱並選擇 Application versions (應用程式版本)。

4. 選取您要刪除的一或多個應用程式版本。



5. 選擇 Actions (動作)，然後選擇 Delete (刪除)。

6. (選用) 若要在您的 Amazon Simple Storage Service (Amazon S3) 儲存貯體內保留這些應用程式版本的應用程式原始碼套件，請清除 Delete versions from Amazon S3 (從 Amazon S3 刪除版本) 方塊。

7. 選擇 Delete (刪除)。

您也可以設定應用程式版本生命週期設定，將 Elastic Beanstalk 設定為會自動刪除舊版本。如果您設定這些生命週期設定，將在您建立新的應用程式版本時套用。例如，若您設定應用程式版本的數量上限為 25 個，在您上傳第 26 個版本時，Elastic Beanstalk 會刪除最舊的版本。若您將最大期限設定為 90 天，當您上傳新版本時，即會刪除任何早於 90 天的版本。如需詳細資訊，請參閱the section called “版本生命週期”。

如果您不選擇從 Amazon S3 刪除原始碼套件，Elastic Beanstalk 仍會從其記錄中刪除該版本。不過，原始碼套件會留存在您的 Elastic Beanstalk 儲存貯體中。應用程式版本配額僅適用於 Elastic Beanstalk 追蹤的版本。因此，您可以刪除版本以保持在配額內，但將所有原始碼套件保留於 Amazon S3 中。

> **Note**
>
> 應用程式版本配額不適用於原始碼套件，但您可能仍須支付 Amazon S3 的費用，並在使用完成後仍繼續保留個人資訊。Elastic Beanstalk 永遠不會自動刪除原始碼套件。您應該在不需要時刪除原始碼套件。

進行應用程式版本生命週期的設定

每當您使用 Elastic Beanstalk 主控台或 EB CLI 上傳應用程式的新版本時，Elastic Beanstalk 都會建立一個應用程式版本。如果不刪除不再使用的版本，最終將會達到應用程式版本配額，而無法建立該應用程式的新版本。

您可在應用程式套用應用程式版本生命週期政策，藉此避免達到配額。生命週期政策會告知 Elastic Beanstalk，讓 Elastic Beanstalk 刪除舊的應用程式版本，或是在應用程式版本總數超過指定的數字時，刪除應用程式的版本。

每當您建立新的應用程式版本時，Elastic Beanstalk 都會套用應用程式的生命週期政策，並在每次套用生命週期政策時，刪除最多 100 個版本。Elastic Beanstalk 會先刪除舊版本再建立新版本，而且不會將新版本計入政策所定義的版本數上限。

Elastic Beanstalk 不會刪除環境目前正在使用的應用程式版本，也不會刪除部署到此政策觸發前 10 週內終止的環境的應用程式版本。

應用程式版本數的配額，適用於同一區域中的所有應用程式。如果您有多個應用程式、請針對每個應用程式設定適合的生命週期政策，以避免達到配額。例如，若您在一個區域有 10 個應用程式，而應用程式版本配額為 1,000，請考慮將所有應用程式的生命週期政策設定為 99 個應用程式版本配額，或者個別設定每個應用程式的限制值，只要應用程式版本總和不超過 1,000。只有在應用程式版本建立成功時，Elastic Beanstalk 才會套用政策，因此若已達到配額，務必先手動刪除一些版本，再建立新版本。

Elastic Beanstalk 預設會在 Amazon S3 中保留應用程式版本的原始碼套件，以防止資料遺失。您可以刪除原始碼套件來節省儲存空間。

您可以透過 Elastic Beanstalk CLI 和 API 來進行生命週期設定。如需詳細資訊，請參閱 eb appversion、CreateApplication (使用 `ResourceLifecycleConfig`參數) 和 UpdateApplicationResourceLifecycle。

在主控台中進行應用程式生命週期設定

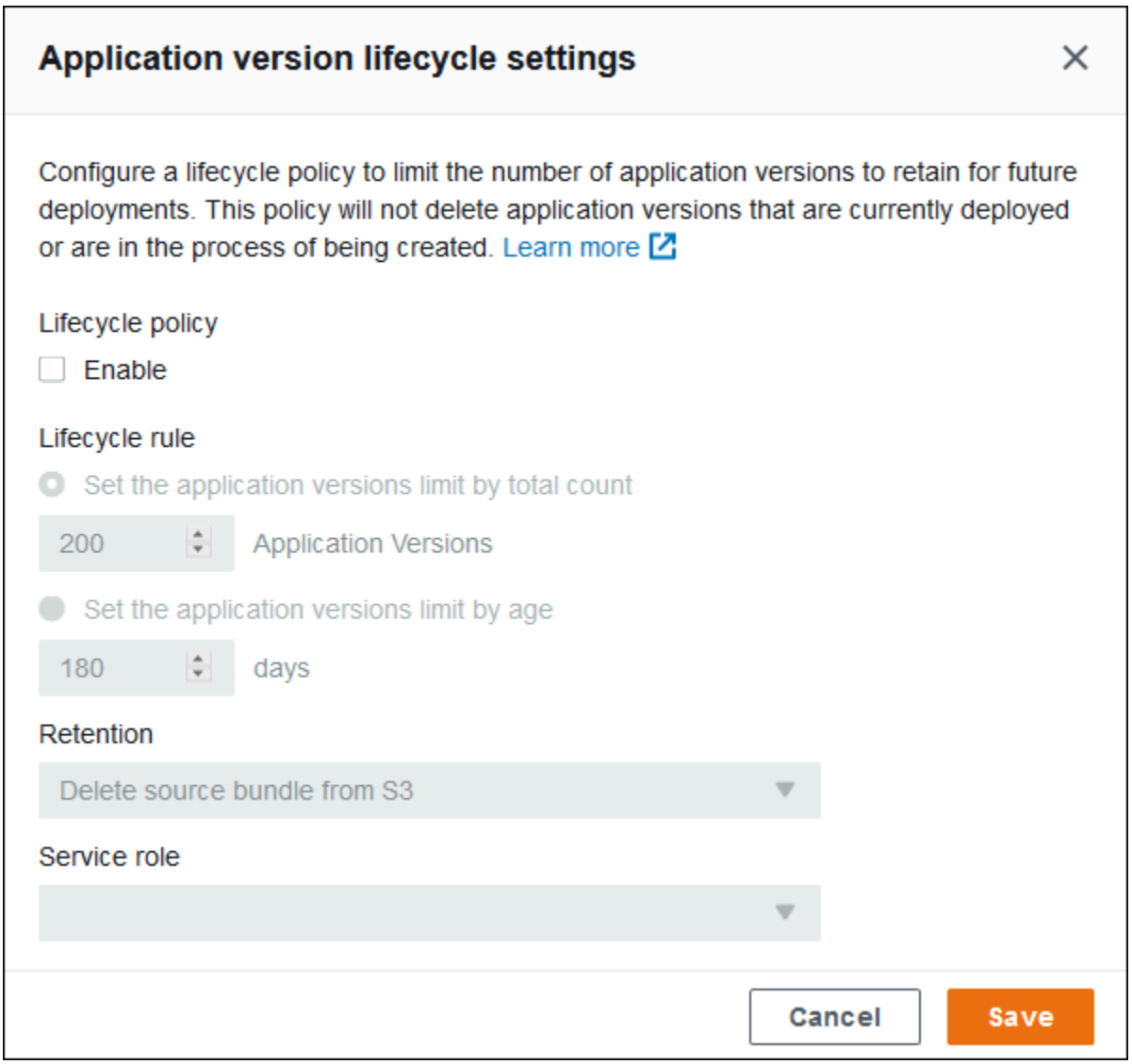
您可以在 Elastic Beanstalk 主控台指定生命週期的設定。

指定您的應用程式生命週期設定

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Applications (應用程式)，然後在清單上選取應用程式名稱。

> **Note**
>
> 如果您有許多應用程式，請使用搜尋列來篩選應用程式清單。

3. 在導覽窗格中，找到應用程式名稱並選擇 Application versions (應用程式版本)。
4. 選擇 Settings (設定)。
5. 使用畫面顯示表單來設定應用程式生命週期設定。
6. 選擇 Save (儲存)。



在此設定頁面上，您可以執行下列動作。

- 根據應用程式版本總數或應用程式版本的期限來設定生命週期設定。
- 指定是否要在刪除應用程式版本時同時刪除 S3 內的原始碼套件。
- 指定要刪除所屬應用程式版本的服務角色。若要將所刪除版本需要的所有許可一併刪除，請選擇名為 `aws-elasticbeanstalk-service-role` 的預設 Elastic Beanstalk 服務角色，或者其他使用 Elastic Beanstalk 受管服務政策的服務角色。如需更多詳細資訊，請參閱 管理 Elastic Beanstalk 服務角色。

標記應用程式版本

您可以將標籤套用至 AWS Elastic Beanstalk 應用程式版本。標籤是與 AWS 資源關聯的金鑰值對。如需 Elastic Beanstalk 資源標記、使用案例、標籤索引鍵和值限制條件的相關資訊，以及支援的資源類型，請參閱標記 Elastic Beanstalk 應用程式資源。

您可以在建立應用程式版本時指定標籤。您可以在現有的應用程式版本中新增或移除標籤，以及更新現有標籤的值。您最多可以為每個應用程式版本新增 50 個標籤。

在應用程式版本建立期間新增標籤

當您使用 Elastic Beanstalk 主控台建立環境，並選擇上傳您的應用程式碼版本時，您可以指定要與新的應用程式版本建立關聯的標籤金鑰和值。

您也可以使用 Elastic Beanstalk 主控台上傳應用程式版本，無須立即在環境中使用。在上傳應用程式版本時，您可以指定標籤索引鍵和值。

若為 AWS CLI 或其他 API 型用戶端，請在 create-application-version 命令中使用 `--tags` 參數來新增標籤。

```
$ aws elasticbeanstalk create-application-version \
      --tags Key=mytag1,Value=value1 Key=mytag2,Value=value2 \
      --application-name my-app --version-label v1
```

當您使用 EB CLI 來建立或更新環境時，系統會從您部署的程式碼建立應用程式版本。您無法在 EB CLI 建立應用程式版本期間直接標記應用程式版本。請參閱以下區段，了解如何將標籤新增至現有的應用程式版本。

管理現有應用程式版本的標籤

您可以新增、更新和刪除現有 Elastic Beanstalk 應用程式版本中的標籤。

若要使用 Elastic Beanstalk 主控台管理應用程式版本的標籤

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Applications (應用程式)，然後在清單上選取應用程式名稱。

> **Note**
>
> 如果您有許多應用程式，請使用搜尋列來篩選應用程式清單。

3. 在導覽窗格中，找到應用程式名稱並選擇 Application versions (應用程式版本)。
4. 選取您要管理的應用程式版本。
5. 選擇 Actions (動作)，然後選擇 Manage tags (管理標籤)。
6. 使用畫面顯示表單來新增、更新或刪除標籤。
7. 若要儲存變更，請選擇頁面底部的儲存變更。

如果您使用 EB CLI 更新應用程式版本，請使用 eb tags 新增、更新、刪除或列出標籤。

例如，以下命令會列出應用程式版本中的標籤。

```
~/workspace/my-app$ eb tags --list --resource "arn:aws:elasticbeanstalk:us-east-2:my-account-id:applicationversion/my-app/my-version"
```

下列命令會更新標籤 mytag1 並刪除標籤 mytag2。

```
~/workspace/my-app$ eb tags --update mytag1=newvalue --delete mytag2 \
      --resource "arn:aws:elasticbeanstalk:us-east-2:my-account-id:applicationversion/my-app/my-version"
```

如需完整選項清單和更多範例，請參閱 eb tags。

若為 AWS CLI 或其他 API 型用戶端，請使用 list-tags-for-resource 命令來列出應用程式版本的標籤。

```
$ aws elasticbeanstalk list-tags-for-resource --resource-arn
 "arn:aws:elasticbeanstalk:us-east-2:my-account-id:applicationversion/my-app/my-version"
```

使用 update-tags-for-resource 命令新增、更新或刪除應用程式版本中的標籤。

```
$ aws elasticbeanstalk update-tags-for-resource \
      --tags-to-add Key=mytag1,Value=newvalue --tags-to-remove mytag2 \
      --resource-arn "arn:aws:elasticbeanstalk:us-east-2:my-account-id:applicationversion/my-app/my-version"
```

在 --tags-to-add 的 update-tags-for-resource 參數中，同時指定欲新增和欲更新的標籤。如此將新增不存在的標籤，並更新現有標籤的值。

> **Note**
>
> 若要將某些 EB CLI 和 AWS CLI 命令與 Elastic Beanstalk 應用程式版本搭配使用，則需要應用程式版本的 ARN。您可使用下列命令來擷取 ARN。
>
> ```
> $ aws elasticbeanstalk describe-application-versions --application-name my-app
>  --version-label my-version
> ```

建立應用程式原始碼套件

當您使用 AWS Elastic Beanstalk 主控台部署新的應用程式或應用程式版本，將需要上傳原始碼套件。您的原始碼套件必須符合下列要求：

- 內含單一 ZIP 檔案或 WAR 檔案 (您可於 WAR 檔案納入多個 ZIP 檔案)
- 不超過 500 MB
- 不含父資料夾或最上層目錄 (可包含子目錄)

若您想要部署會定期處理背景任務的工作者應用程式，您的應用程式原始碼套件務必納入 `cron.yaml` 檔案。如需更多詳細資訊，請參閱 周期性任務。

若您透過 Elastic Beanstalk 命令列界面 (EB CLI)、AWS Toolkit for Eclipse 或 AWS Toolkit for Visual Studio 來部署應用程式，將自動正確建構 ZIP 或 WAR 檔案。如需詳細資訊，請參閱 使用 Elastic Beanstalk 命令列界面 (EB CLI)、在 Elastic Beanstalk 建立和部署 Java 應用程式 及 AWS Toolkit for Visual Studio。

章節

- 自命令列建立原始碼套件
- 透過 Git 建立原始碼套件
- 於 Mac OS X Finder 或 Windows 檔案總管壓縮檔案
- 建立 .NET 應用程式的原始碼套件
- 測試您的原始碼套件

自命令列建立原始碼套件

使用 `zip` 命令來建立原始碼套件。欲包含隱藏檔案和資料夾，需使用的模式如下。

```
~/myapp$ zip ../myapp.zip -r * .[^.]*
  adding: app.js (deflated 63%)
  adding: index.js (deflated 44%)
  adding: manual.js (deflated 64%)
  adding: package.json (deflated 40%)
  adding: restify.js (deflated 85%)
  adding: .ebextensions/ (stored 0%)
  adding: .ebextensions/xray.config (stored 0%)
```

如此可確保 Elastic Beanstalk 組態檔案與其他以句點 (.) 為首的檔案和資料夾，均包含在封存中。

以 Tomcat Web 應用程式而言，請使用 `jar` 來建立 Web 封存檔。

```
~/myapp$ jar -cvf myapp.war .
```

上述命令納入的隱藏檔案，可能會增加原始碼套件不必要的大小。如需其他控制，請使用更詳細的檔案模式，或透過 Git 建立您的原始碼套件。

透過 Git 建立原始碼套件

若您透過 Git 管理您的應用程式原始碼，請使用 `git archive` 命令來建立您的原始碼套件。

```
$ git archive -v -o myapp.zip --format=zip HEAD
```

`git archive` 僅納入存放於 Git 的檔案，且會排除忽略檔案和 Git 檔案，如此將盡可能減少原始碼套件的大小。如需詳細資訊，請前往 git-archive 手冊頁面。

於 Mac OS X Finder 或 Windows 檔案總管壓縮檔案

當您於 Mac OS X Finder 或 Windows 檔案總管建立 `ZIP` 檔案，請確保您壓縮的是檔案和子資料夾本身，而非壓縮父資料夾。

> **Note**
>
> Mac OS X 和 Linux 作業系統的圖形使用者介面 (GUI)，不會顯示檔名以句點 (.) 為首的檔案和資料夾。若 `ZIP` 檔案必須納入諸如 `.ebextensions` 的隱藏資料夾，請使用命令列來壓縮您的應用程式，不要使用 GUI。如需於 Mac OS X 或 Linux 作業系統上透過命令列程序建立 `ZIP` 檔案的詳細資訊，請參閱 自命令列建立原始碼套件。

Example

假設您有一個標記為 myapp 的 Python 專案資料夾，其中包含下列檔案和子資料夾：

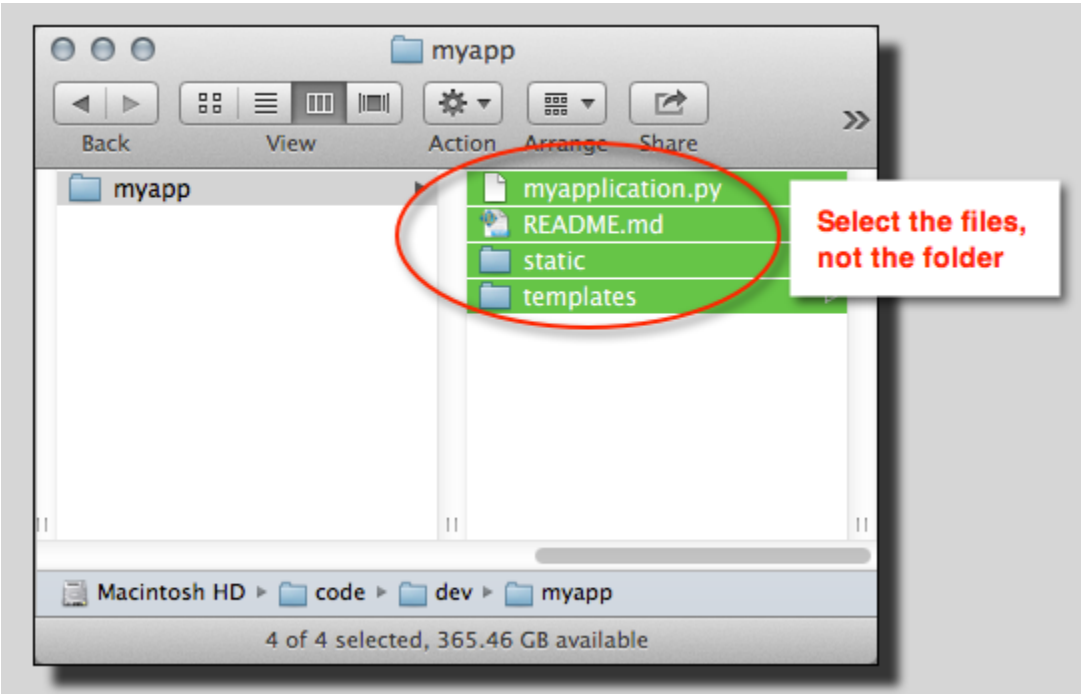
```
myapplication.py
README.md
static/
static/css
static/css/styles.css
static/img
static/img/favicon.ico
static/img/logo.png
templates/
templates/base.html
templates/index.html
```

如上列要求，您壓縮的原始碼套件不得包含父資料夾，因此其解壓縮後的結構，不會出現額外的最上層目錄。在此範例中，檔案解壓縮時不應建立 myapp 資料夾 (，或者，不應於命令列將 myapp 區段新增至檔案路徑)。

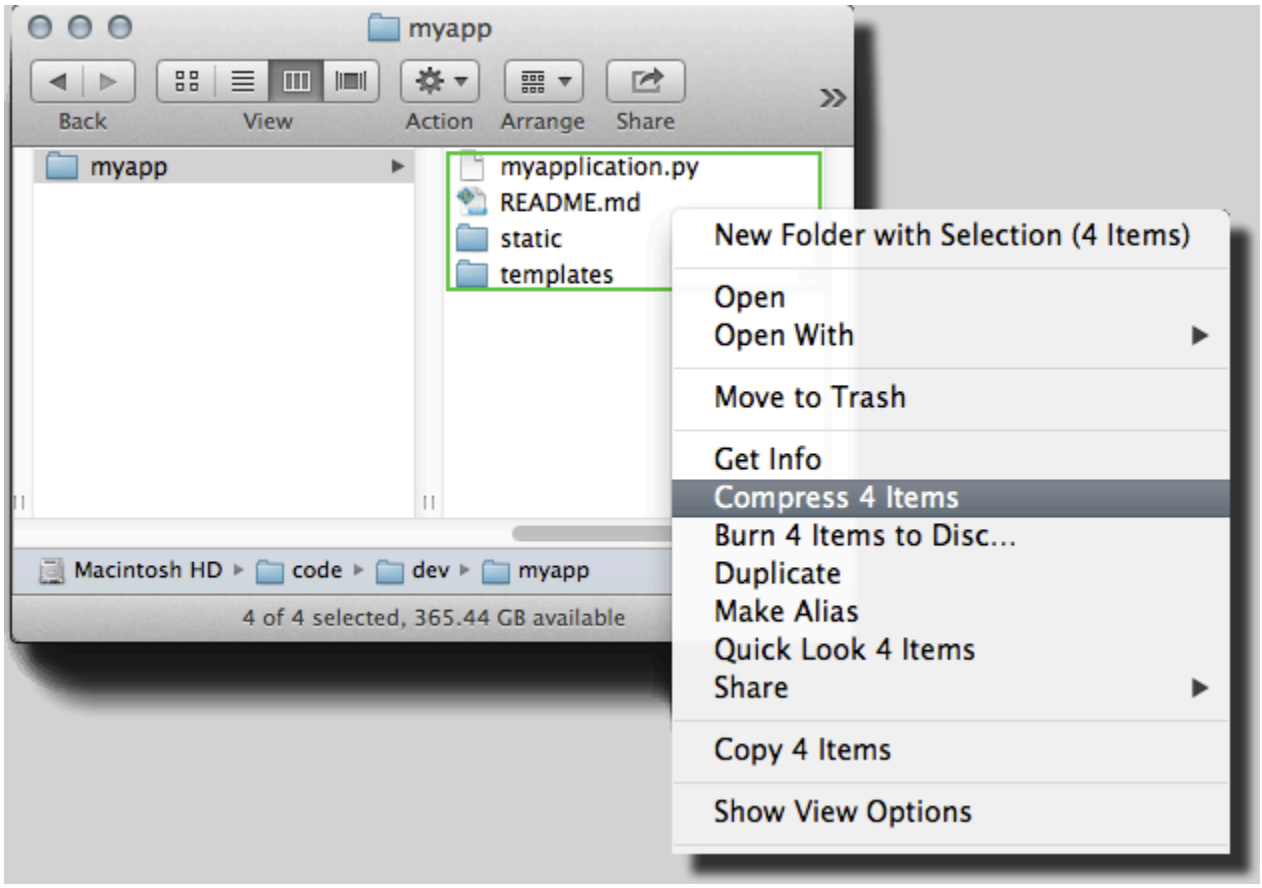
本主題全程採用此範例檔案結構，藉此說明如何壓縮檔案。

欲在 Mac OS X Finder 壓縮檔案

1. 開啟最上層的專案資料夾，並選取其中的所有檔案和子資料夾。請不要選取最上層資料夾本身。

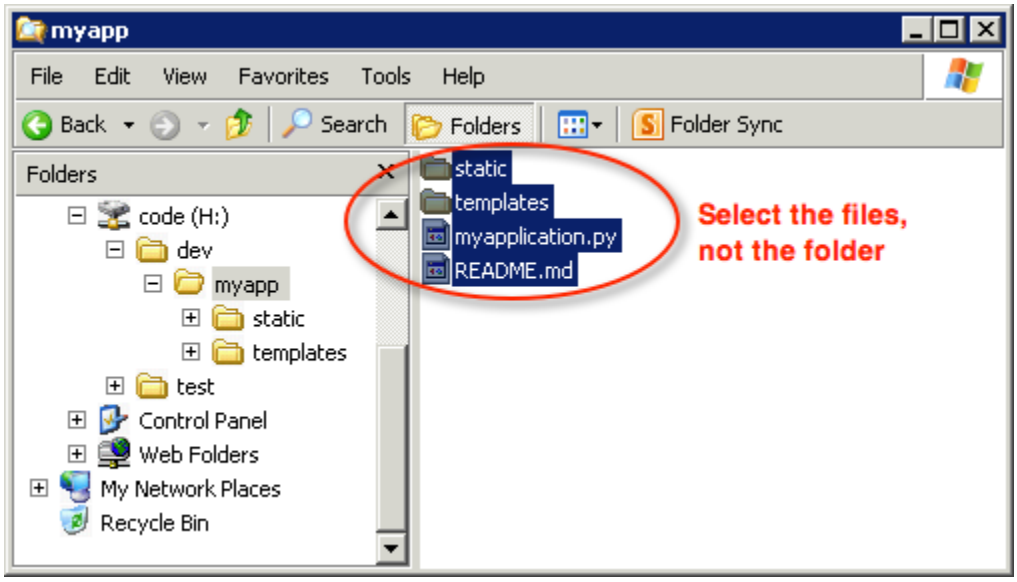


2. 在所選檔案上按一下滑鼠右鍵，然後選擇 Compress X items (壓縮 X 個項目)，其中 X 是您所選擇的檔案和子資料夾的數量。

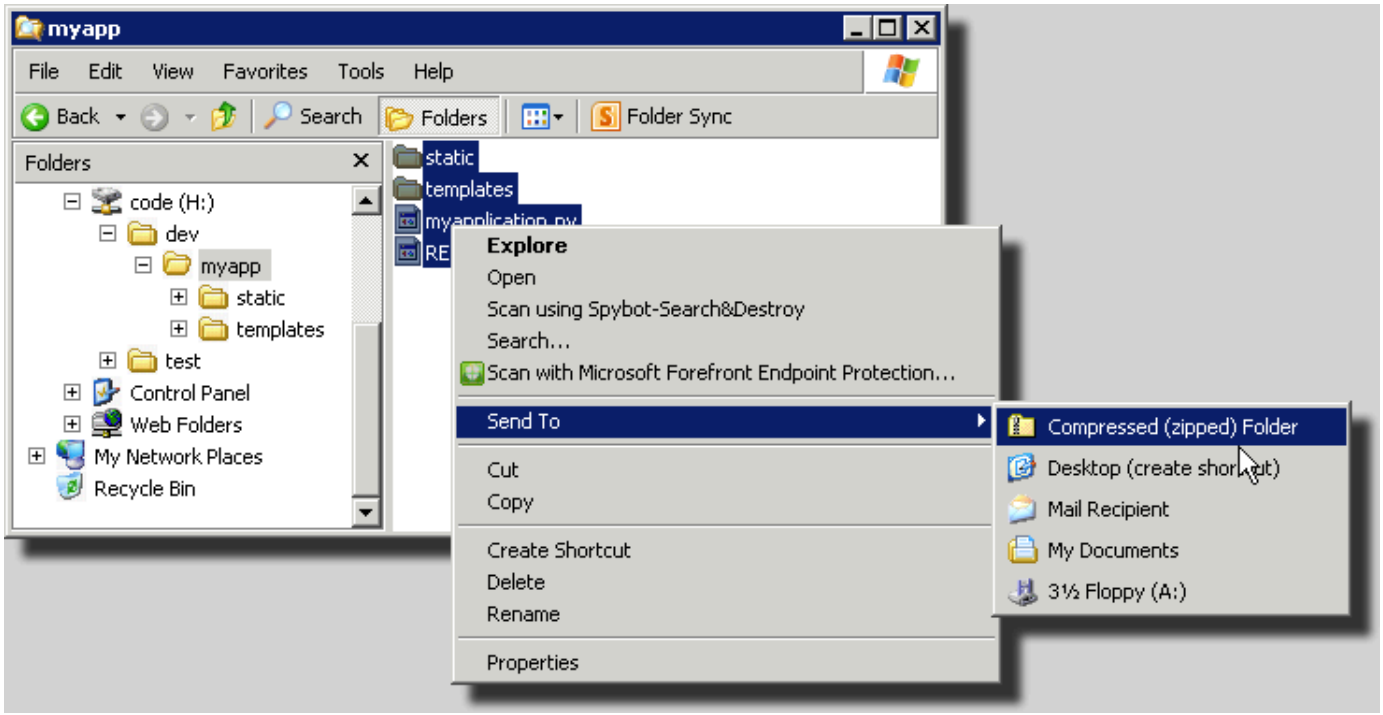


欲在 Windows 檔案總管壓縮檔案

1. 開啟最上層的專案資料夾，並選取其中的所有檔案和子資料夾。請不要選取最上層資料夾本身。



2. 在所選檔案上按一下滑鼠右鍵，選擇 Send to (傳送到)，然後選擇 Compressed (zipped) folder (壓縮的 (zipped) 資料夾)。



建立 .NET 應用程式的原始碼套件

如果您使用 Visual Studio，您可以使用包含在 AWS Toolkit for Visual Studio 中的部署工具，將您的 .NET 應用程式部署到 Elastic Beanstalk。如需更多詳細資訊，請參閱 使用部署工具在 .NET 中部署 Elastic Beanstalk 應用程式。

若您需要手動建立 .NET 應用程式的原始碼套件，您無法直接建立內含專案目錄的 ZIP 檔案。您必須為您的專案建立能夠部署至 Elastic Beanstalk 的 Web 部署套件。建立部署套件可採用多種方法：

- 使用 Visual Studio 內的 Publish Web (發佈 Web) 精靈，藉此建立部署套件。如需詳細資訊，請前往 How to: Create a Web Deployment Package in Visual Studio。

  > **Important**
  >
  > 建立 Web 部署套件時，Site name (網站名稱) 開頭必須為 `Default Web Site`。

- 若您有 .NET 專案，則可以使用 msbuild 命令來建立部署套件，如下列範例所示。

  > **Important**
  >
  > `DeployIisAppPath` 參數開頭必須為 `Default Web Site`。

  ```
  C:/> msbuild <web_app>.csproj /t:Package /p:DeployIisAppPath="Default Web Site"
  ```

- 若您有網站專案，則可以使用 IIS Web 部署工具來建立部署套件。如需詳細資訊，請前往 Packaging and Restoring a Web site。

  > **Important**
  >
  > `apphostconfig` 參數開頭必須為 `Default Web Site`。

若您部署多個應用程式或 ASP.NET Core 應用程式，請將 `.ebextensions` 資料夾放置於原始碼套件的根目錄，與應用程式套件和資訊清單檔案並列：

```
~/workspace/source-bundle/
|-- .ebextensions
|   |-- environmentvariables.config
|   `-- healthcheckurl.config
|-- AspNetCore101HelloWorld.zip
|-- AspNetCoreHelloWorld.zip
|-- aws-windows-deployment-manifest.json
`-- VS2015AspNetWebApiApp.zip
```

測試您的原始碼套件

建議您先於本機測試來源套件，之後再上傳至 Amazon Elastic Beanstalk。由於 Elastic Beanstalk 基本上使用命令列來擷取檔案，因此建議從命令列進行測試，而不要使用 GUI 工具。

欲在 Mac OS X 或 Linux 測試檔案擷取

1. 開啟終端視窗 (Mac OS X) 或連接至 Linux 伺服器。前往內含原始碼套件的目錄。
2. 使用 `unzip` 或 `tar xf` 命令，解壓縮封存檔。
3. 請確認解壓縮的檔案與封存檔出現在相同的資料夾，而非在新的最上層資料夾或目錄。

> **Note**
>
> 若您使用 Mac OS X Finder 來解壓縮封存檔時，無論封存檔本身的結構為何，都會建立新的最上層資料夾。為了獲得最佳結果，請使用命令列。

欲在 Windows 測試檔案擷取

1. 下載或安裝可讓您透過命令列擷取壓縮檔案的程式。例如，您可以自 http://stahlforce.com/dev/index.php?tool=zipunzip 免費下載 unzip.exe 程式。
2. 可視需要將執行檔複製至內含原始碼套件的目錄。若您已安裝全系統的工具，則可略過此步驟。
3. 使用合適的命令，解壓縮封存檔。若您已使用步驟 1 的連結下載 unzip.exe，則命令為 `unzip <archive-name>`。
4. 請確認解壓縮的檔案與封存檔出現在相同的資料夾，而非在新的最上層資料夾或目錄。

標記 Elastic Beanstalk 應用程式資源

您可以將標籤套用至 AWS Elastic Beanstalk 應用程式的資源。標籤是與 AWS 資源關聯的金鑰值對。標籤可協助您分類資源。如果您隨著多個 AWS 應用程式管理許多資源，則標籤特別有用。

以下是使用標籤搭配 Elastic Beanstalk 資源的一些方法：

- 部署階段 - 識別與應用程式的不同階段 (例如開發、試用版和生產) 相關聯的資源。
- 成本分配 – 使用成本分配報告來追蹤與各種支出帳戶關聯的 AWS 資源的用量。此報告同時包含已標記和未標記的資源，並且會根據標籤彙總成本。如需成本分配報告使用標籤方式的資訊，請參閱《AWS 帳單與成本管理使用者指南》中的針對自訂帳單報告使用成本分配標籤。

- 存取控制 - 使用標籤來管理對於請求與資源的許可。例如，只能建立和管理 beta 環境的使用者，應該只能存取試用版階段資源。如需詳細資訊，請參閱使用標籤來控制對 Elastic Beanstalk 資源的存取。

每個 資源最多可新增 50 個標籤。環境稍有不同：Elastic Beanstalk 會新增三個預設系統標籤至環境，而且您無法編輯或刪除這些標籤。除了預設標籤，您至多可於每個環境新增其他 47 個標籤。

以下限制適用於標籤索引鍵和值：

- 金鑰和值可包含字母、數字、空格和下列符號：`_ . : / = + - @`
- 金鑰最多可包含 127 個字元。值最多可包含 255 個字元。

> **Note**
>
> 這些長度限制用於 UTF-8 格式的 Unicode 字元。對於其他多位元組編碼，限制可能較低。

- 金鑰會區分大小寫。
- 金鑰的開頭不可為 `aws:` 或 `elasticbeanstalk:`。

對啟動範本的標籤傳輸

Elastic Beanstalk 提供了一個選項，可啟用對啟動範本的標籤傳輸。此選項延續了對啟動範本的標籤型存取控制 (TBAC) 的支援。

> **Note**
>
> 啟動組態已逐步淘汰，並由啟動範本取代。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的啟動組態。

為了防止執行 EC2 執行個體時發生停機，AWS CloudFormation 不會將標籤傳播到現有的啟動範本。如果有使用案例需要環境資源具備標籤，您可以啟用 Elastic Beanstalk，建立包含這些資源標籤的啟動範本。若要這麼做，請將 aws:autoscaling:launchconfiguration 命名空間中的 `LaunchTemplateTagPropagationEnabled` 選項設定為 `true`。預設值為 `false`。

下列組態檔案範例可啟用對啟動範本的標籤傳輸。

```
option_settings:
```

```
  aws:autoscaling:launchconfiguration:
    LaunchTemplateTagPropagationEnabled: true
```

Elastic Beanstalk 只能將下列資源標籤傳播到啟動範本：

- EBS 磁碟區
- EC2 執行個體
- EC2 網路界面
- 定義資源的 AWS CloudFormation 啟動範本

存在此限制的原因是 CloudFormation 僅允許在建立範本時為資源設定標籤。如需詳細資訊，請參閱《AWS CloudFormation 使用者指南》中的 TagSpecification。

> **Important**
>
> - 將現有環境的此選項值從 `false` 變更為 `true` 可能導致現有標籤發生重大變更。
> - 啟用此功能後，標籤的傳播會需要 EC2 替換，這可能會導致停機。您可以啟用滾動式更新，批次套用組態變更，從而避免在更新程序期間停機。如需詳細資訊，請參閱組態變更。

如需有關啟動範本的詳細資訊，請參閱以下內容：

- Amazon EC2 Auto Scaling User Guide 中的 Launch templates。
- 《AWS CloudFormation 使用者指南》中的使用範本
- 《AWS CloudFormation 使用者指南》中的 Elastic Beanstalk 範本程式碼片段

您可以標記的資源

以下是您可以標記的 Elastic Beanstalk 資源的類型，以及有關為各個資源管理標籤的特定主題連結：

- 應用程式
- 環境
- 應用程式版本
- 已儲存的組態
- 自訂平台版本

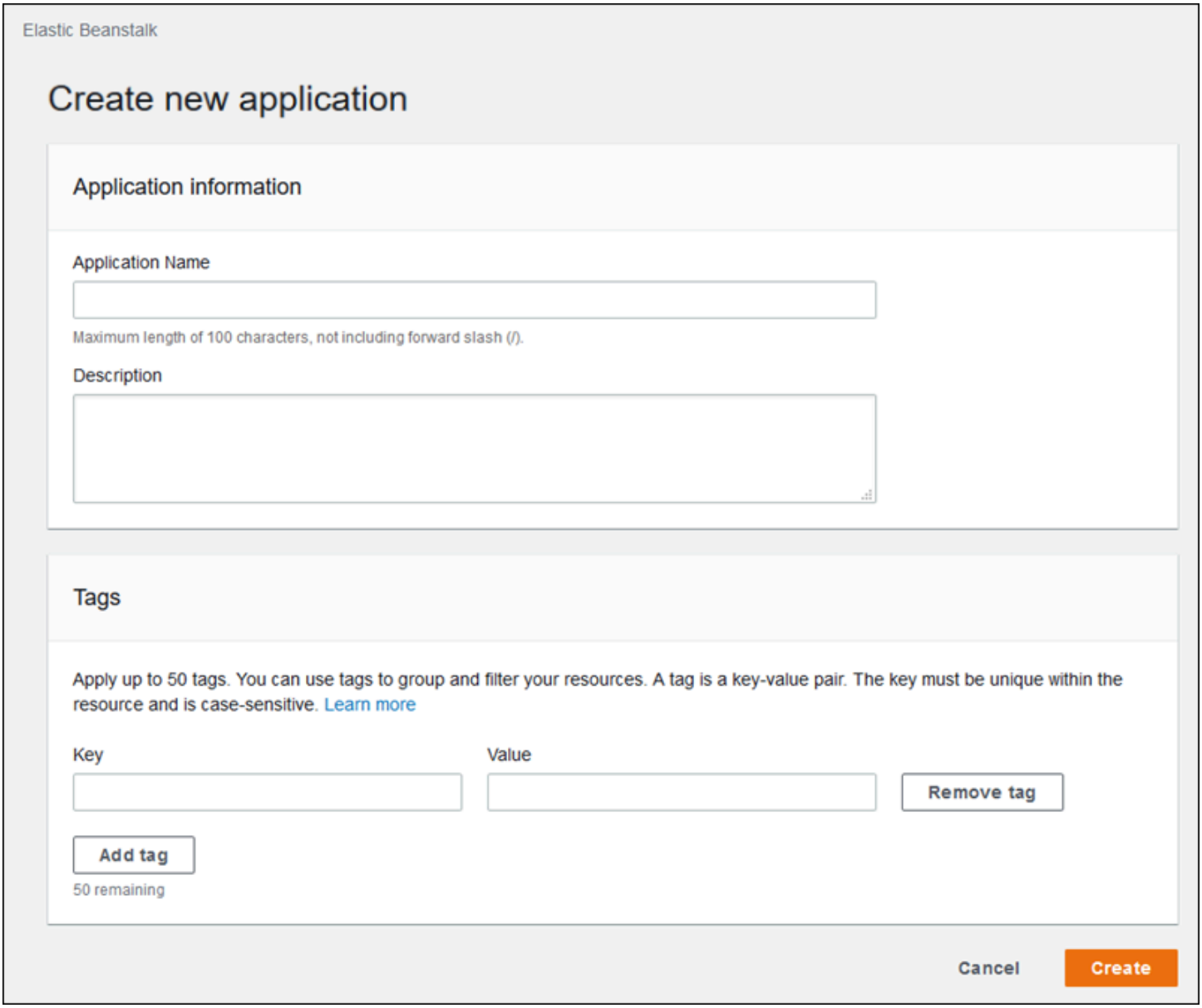
標記應用程式

您可以將標籤套用至 AWS Elastic Beanstalk 應用程式。標籤是與 AWS 資源關聯的金鑰值對。如需 Elastic Beanstalk 資源標記、使用案例、標籤索引鍵和值限制條件的相關資訊，以及支援的資源類型，請參閱標記 Elastic Beanstalk 應用程式資源。

您可以在建立應用程式時指定標籤。您可以在現有的應用程式中新增或移除標籤，以及更新現有標籤的值。您最多可以為每個應用程式新增 50 個標籤。

在應用程式建立期間新增標籤

使用 Elastic Beanstalk 主控台來建立應用程式時，您可以在建立新的應用程式對話方塊中指定標籤索引鍵和值。



若您使用 EB CLI 來建立應用程式，請使用帶有 `--tags` 的 eb init 選項來新增標籤。

```
~/workspace/my-app$ eb init --tags mytag1=value1,mytag2=value2
```

若為 AWS CLI 或其他 API 型用戶端，請在 create-application 命令中使用 `--tags` 參數來新增標籤。

```
$ aws elasticbeanstalk create-application \
      --tags Key=mytag1,Value=value1 Key=mytag2,Value=value2 \
      --application-name my-app --version-label v1
```

管理現有應用程式的標籤

您可以在現有的 Elastic Beanstalk 應用程式中新增、更新和刪除標籤。

在 Elastic Beanstalk 主控台中管理應用程式的標籤

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Applications (應用程式)，然後在清單上選取應用程式名稱。

   > **Note**
   >
   > 如果您有許多應用程式，請使用搜尋列來篩選應用程式清單。

3. 選擇 Actions (動作)，然後選擇 Manage tags (管理標籤)。
4. 使用畫面顯示表單來新增、更新或刪除標籤。
5. 若要儲存變更，請選擇頁面底部的儲存變更。

如果您使用 EB CLI 更新應用程式，請使用 eb tags 新增、更新、刪除或列出標籤。

例如，以下命令會列出應用程式中的標籤。

```
~/workspace/my-app$ eb tags --list --resource "arn:aws:elasticbeanstalk:us-east-2:my-account-id:application/my-app"
```

下列命令會更新標籤 `mytag1` 並刪除標籤 `mytag2`。

```
~/workspace/my-app$ eb tags --update mytag1=newvalue --delete mytag2 \
      --resource "arn:aws:elasticbeanstalk:us-east-2:my-account-id:application/my-app"
```

如需完整選項清單和更多範例，請參閱 `eb tags`。

若為 AWS CLI 或其他 API 型用戶端，請使用 list-tags-for-resource 命令來列出應用程式的標籤。

```
$ aws elasticbeanstalk list-tags-for-resource --resource-arn
 "arn:aws:elasticbeanstalk:us-east-2:my-account-id:application/my-app"
```

使用 update-tags-for-resource 命令新增、更新或刪除應用程式中的標籤。

```
$ aws elasticbeanstalk update-tags-for-resource \
```

```
      --tags-to-add Key=mytag1,Value=newvalue --tags-to-remove mytag2 \
      --resource-arn "arn:aws:elasticbeanstalk:us-east-2:my-account-id:application/my-
app"
```

在 `--tags-to-add` 的 update-tags-for-resource 參數中，同時指定欲新增和欲更新的標籤。如此將新增不存在的標籤，並更新現有標籤的值。

> **Note**
>
> 若要將某些 EB CLI 和 AWS CLI 命令與 Elastic Beanstalk 應用程式搭配使用，則需要應用程式的 ARN。您可使用下列命令來擷取 ARN。
>
> ```
> $ aws elasticbeanstalk describe-applications --application-names my-app
> ```

管理環境

AWS Elastic Beanstalk 可讓您為應用程式輕鬆建立新的環境。您可針對開發、測試和生產使用等目的，分別建立並管理不同環境，也可於環境部署任何版本的應用程式。環境可為長期執行或暫時的環境。終止環境時，您可儲存其組態供未來重新建立使用。

當您開發應用程式時，您可能常常需要將其部署至不同用途的多個環境。Elastic Beanstalk 可讓您設定部署的執行方式。您可同時部署至環境中的所有執行個體，或透過滾動部署將部署分為不同批次。

組態變更會與部署分開進行，且有自己的範圍。例如，若您變更執行您應用程式的 EC2 執行個體類型，必須替換所有執行個體。另一方面，若您修改環境負載平衡器的組態，此變更無須中斷服務或降低容量即可完成。您亦可透過滾動組態更新，以批次套用組態變更來修改環境中的執行個體。

> **Note**
>
> 僅使用 Elastic Beanstalk 來修改您環境中的資源。若您使用其他服務的主控台、CLI 命令或 SDK 來修改資源，Elastic Beanstalk 將無法準確監控這些資源的狀態，而且您也無法儲存其組態或可靠重建環境。外部變更也會在更新或終止環境時造成問題。

啟動環境時，您會選擇平台版本。我們會使用新的平台版本定期更新平台，以改善效能並提供新功能。您可以隨時將您的環境更新至最新的平台版本。

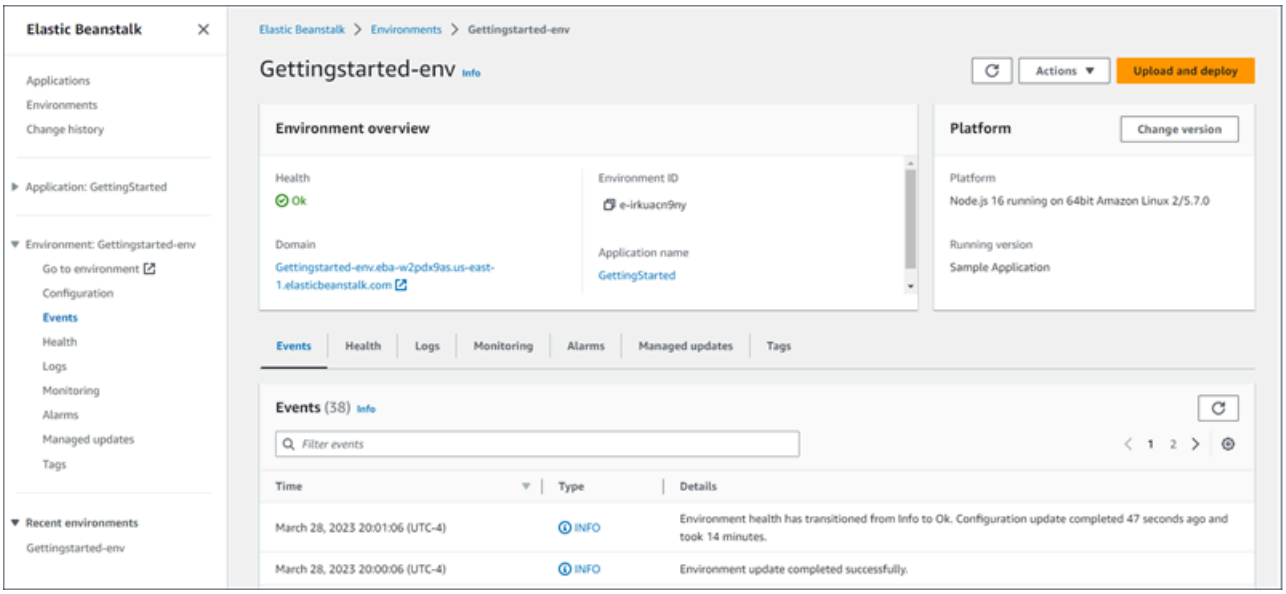
隨著您應用程式的複雜性增加，您可將其分割為多個元件，分別在不同環境中執行。對於長時間執行的任務負載，您可以啟動處理 Amazon Simple Queue Service (Amazon SQS) 佇列任務的工作者環境。

主題

- 在 Elastic Beanstalk 環境之間建立連結

使用 Elastic Beanstalk 環境管理主控台

Elastic Beanstalk 主控台為您提供一個 Environment overview (環境概觀) 頁面，以便您管理每個 AWS Elastic Beanstalk 環境。在 Environment overview (環境概觀) 面中，您可以管理環境的組態並執行常見動作。這些動作包括重新啟動在環境中執行的 Web 伺服器、複製您的環境，或從頭開始重新建置環境。



使用環境管理主控台

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

您會看到 Environment overview (環境概觀) 頁面。主控台的導覽窗格會顯示環境所屬的應用程式名稱和相關的應用程式管理頁面，以及環境名稱和環境管理頁面。

主題

- 環境概觀

- 環境動作
- 事件
- 醫療保健
- 日誌
- 監控
- 警示
- 受管更新
- Tags (標籤)
- 組態

環境概觀

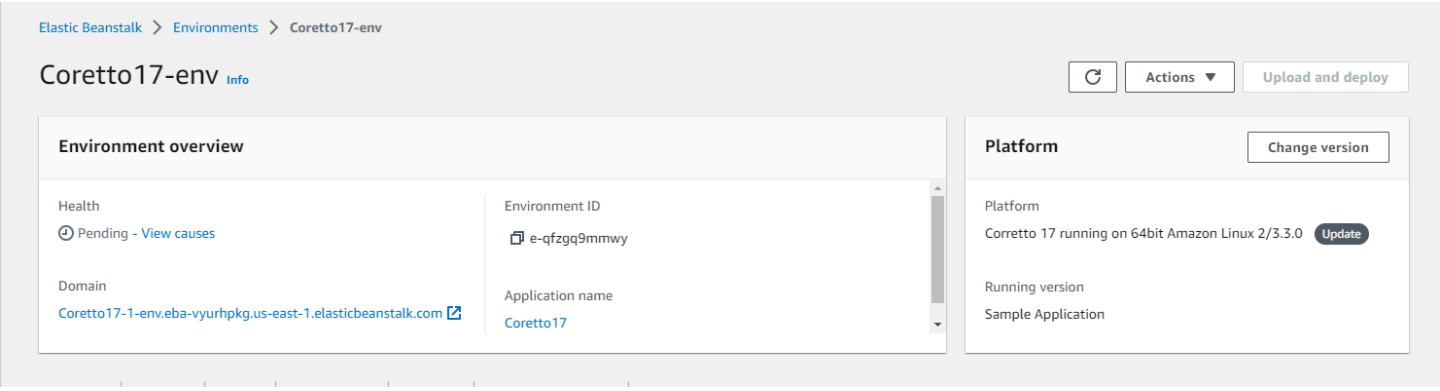
若要檢視 Environment overview (環境概觀) 頁面，請在導覽窗格中選擇環境名稱 (如果是目前的環境)。或者，請從 Applications (應用程式) 頁面或從 Environments (環境) 頁面上的主要環境清單導覽至環境。

環境概觀頁面的頂端窗格會顯示您環境的最上層資訊。這包括它的名稱、URL、目前的運作狀態、目前部署的應用程式版本名稱，以及應用程式執行所在的平台版本。在概觀窗格下方，您可以看到最近的環境事件。

選擇 Refresh (重新整理) 來更新顯示的資訊。概觀頁面包含下列資訊和選項。

醫療保健

環境的整體運作狀態。如果環境的運作狀態下降，檢視原因連結會在環境運作狀態旁顯示。選取此連結，可檢視具有更多詳細資訊的運作狀態標籤。



網域

環境的 Domain (網域) 或 URL 位於 Environment overview (環境概觀) 頁面的上部、環境 Health (運作狀態) 下方。這是環境所執行之 Web 應用程式的 URL。

環境 ID

環境 ID。這是建立環境時產生的內部 ID。

應用程式名稱

部署在您環境中並執行的應用程式名稱。

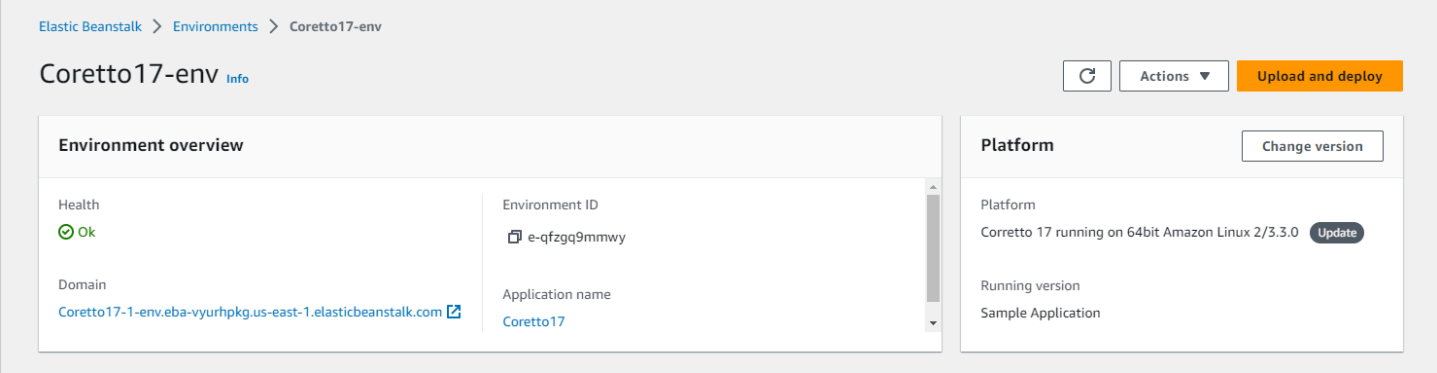
執行版本

部署在您環境中並執行的應用程式版本名稱。選擇 Upload and deploy (上傳並部署) 來上傳 source bundle (原始碼套件)，並將其部署至您的環境。此選項會建立新的應用程式版本。

平台

在您環境中執行的平台版本名稱。通常，這包括架構、作業系統 (OS)、語言和應用程式伺服器 (統稱為平台分支)，並具有特定的平台版本編號。

如果您的平台版本不是最新版本，平台區段中的平台版本旁會顯示狀態標籤。更新標籤會指示雖然支援該平台版本，但仍有更新版本可用。平台版本也可能會被標示為已過時或已淘汰。選取變更版本，可將您的平台分支更新為較新的版本。如需有關平台版本狀態的詳細資訊，請參閱 Elastic Beanstalk 平台詞彙表 中的平台分支章節。



環境概觀索引標籤

頁面下半部分顯示的索引標籤含有更多關於環境的詳細資訊，並可讓您存取其他功能：

- Events (事件) – 顯示此環境所用資源之 Elastic Beanstalk 服務和其他服務的資訊或錯誤訊息。
- Health (運作狀態) – 顯示執行您應用程式之 Amazon EC2 執行個體的狀態和相關的詳細運作狀態資訊。

- Logs (日誌) – 從您環境中的 Amazon EC2 擷取和下載日誌。您可以擷取完整的日誌或最近的活動。擷取到的日誌的可用時間為 15 分鐘。
- Monitoring (監控) – 顯示環境的統計資料，例如平均延遲和 CPU 使用率。
- Alarms (警示) – 顯示您設定的環境指標警示。您可以在此頁面上新增、修改或刪除警示。
- Managed updates (受管更新) – 顯示即將到來和已完成之受管平台更新與執行個體替換的相關資訊。
- Tags (標籤) – 顯示環境標籤並允許您管理它們。標籤是套用至您環境的鍵/值對。

> **Note**
>
> 主控台左側的導覽窗格列出連結以及與索引標籤相同的名稱。選取其中任何一個連結將會顯示相應索引標籤的內容。

環境動作

環境概觀頁面內包含可用於執行環境的常見操作之 Actions (動作) 功能表。此功能表顯示在 Create a new environment (建立新的環境) 選項旁邊的環境標頭右側。

> **Note**
>
> 某些動作只有在特定條件下才能使用，直至符合正確的條件才會保持停用狀態。

載入組態

載入之前的已儲存組態。組態會儲存至您的應用程式，並可載入任何相關聯的環境。若已變更環境資訊，您可載入已儲存組態來復原這些變更。您亦可載入從執行相同應用程式的不同環境儲存之組態，在環境間傳播組態變更。

儲存組態

將您環境的目前組態儲存至應用程式。在進行環境組態變更前，請儲存目前的組態，以利稍後需要可還原。啟動新環境時，亦可套用已儲存組態。

交換環境網域 (URL)

交換目前環境與新環境的 CNAME。在 CNAME 交換後，所有使用環境 URL 應用程式的流量會前往新的環境。當您準備好部署新的應用程式版本，可在新版本下啟動不同環境。當新環境準備好接受請求，

請執行 CNAME 交換，開始將流量路由至新的環境。這樣做不會中斷您的服務。如需更多詳細資訊，請參閱 透過 Elastic Beanstalk 進行藍/綠部署。

複製環境

透過目前執行環境的相同組態來啟動新的環境。

透過最新平台複製

透過使用中的 Elastic Beanstalk 平台最新版本，複製您目前的環境。此選項僅在目前環境平台的新版本可用時適用。

中止目前操作

停止進行中的環境更新。停用操作可能會造成環境中部分執行個體出現不同的狀態，此情況取決於操作的進度。此選項僅在當環境正被更新時可用。

重新啟動應用程式伺服器

重新啟動在環境執行個體上執行的 Web 伺服器。此選項不會終止或重新啟動任何 AWS 資源。若您的環境因為一些錯誤要求，出現奇怪的反應，重新啟動應用程式伺服器可在您排解根本原因時暫時還原功能。

重建環境

終止所有執行環境中的資源，並以相同設定建立新的環境。此操作需要幾分鐘，類似於從頭部署新環境所需的時間。任何於環境資料層中執行的 Amazon RDS 執行個體，在重建期間會被刪除。若您需要資料，請建立快照。您可於 RDS 主控台手動建立快照，或設定您資料層的刪除政策，讓其於執行個體刪除前自動建立快照。這是您建立資料層時的預設設定。

終止環境

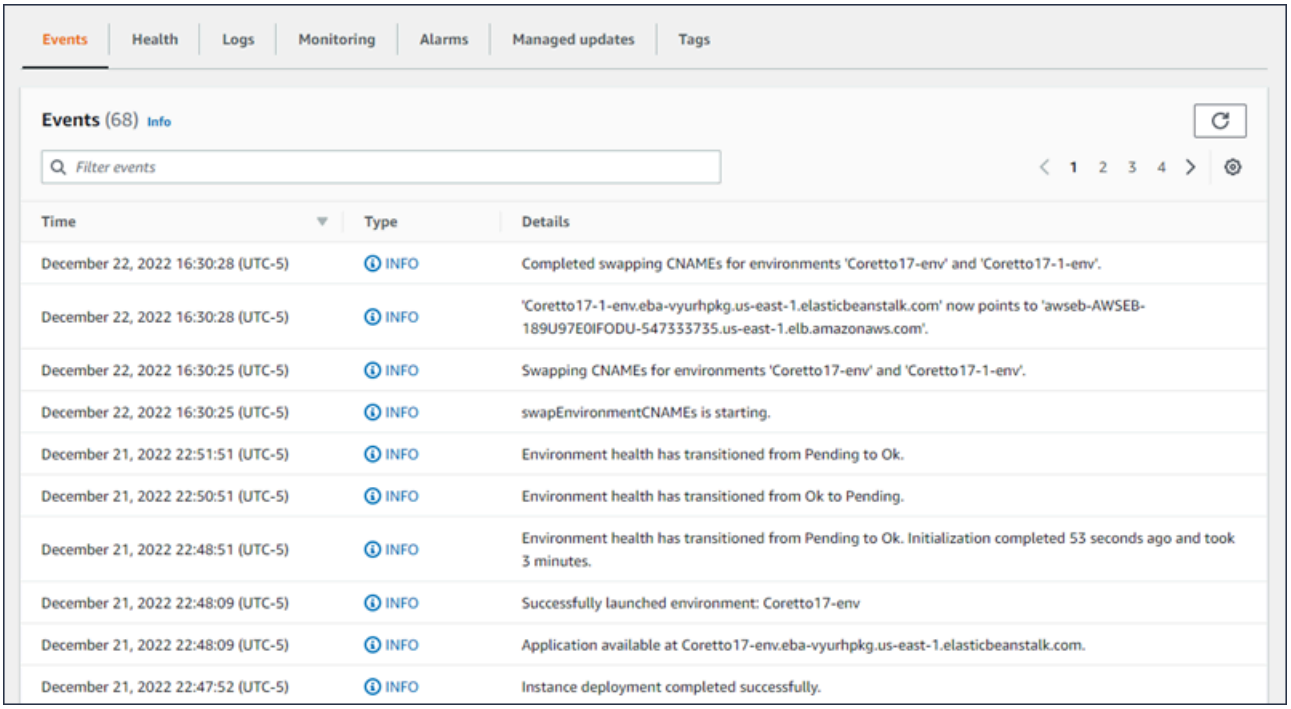
終止執行環境中的所有資源，並將環境自應用程式移除。若您有在資料層中執行的 RDS 執行個體，且需要保留其資料，請確保已將 database deletion policy (資料庫刪除政策) 設定為 `Snapshot` 或 `Retain`。如需詳細資訊，請參閱本指南的「設定環境」一章中的 資料庫生命週期。

還原環境

環境終止一小時內，可於此頁面還原環境。超過一小時後，您可自應用程式概觀頁面還原環境。

事件

Events (事件) 索引標籤會顯示環境的事件串流。只要您與環境互動，或因此建立或修改環境的資源，Elastic Beanstalk 都會輸出事件訊息。

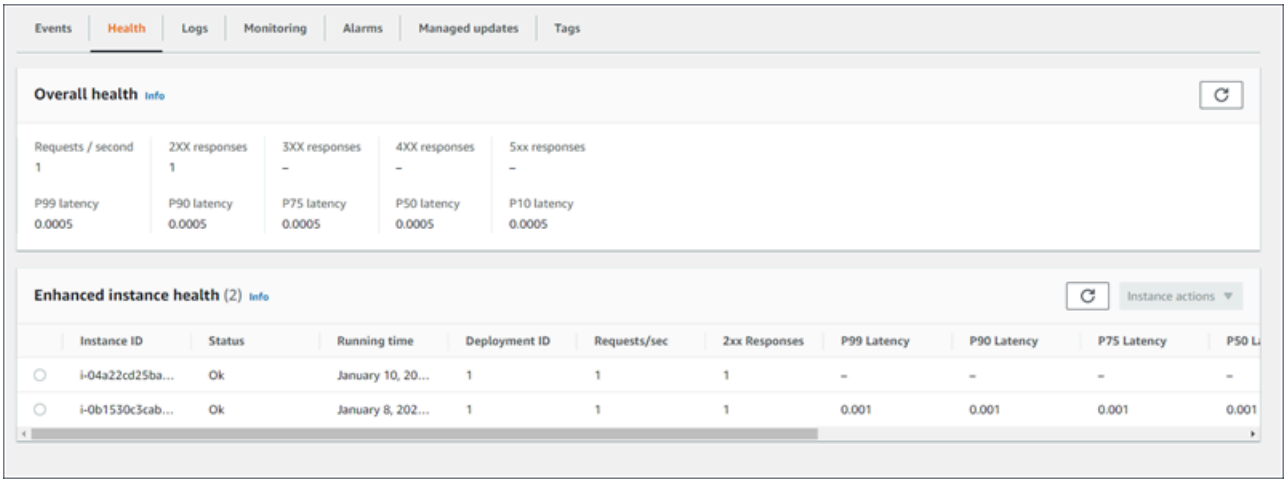


如需更多詳細資訊，請參閱 檢視 Elastic Beanstalk 環境的事件資料流。

醫療保健

若啟用增強型運作狀態監控，此頁面會顯示執行個體的即時運作狀態資訊。Overall health (整體運作狀態) 窗格會顯示運作狀態資料，其為結合所有環境執行個體的平均值。Enhanced instance health (增強型執行個體運作狀態) 窗格會顯示您環境中個別執行個體的即時運作狀態資訊。增強型運作狀態監控讓 Elastic Beanstalk 能仔細監控您環境中的資源，以便更精確評估應用程式的運作狀態。

當啟用增強型運作狀態監控時，此頁面會顯示環境中執行個體提供的請求資訊，以及作業系統的指標，包括延遲、負載和 CPU 使用率。

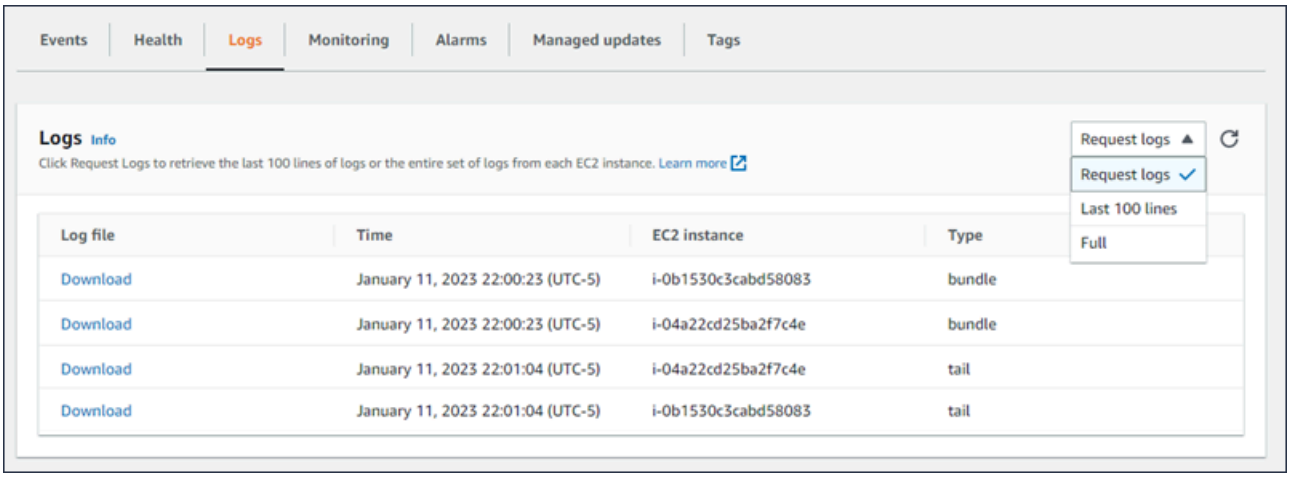


如需更多詳細資訊，請參閱 增強型運作狀態報告與監控。

日誌

Logs (日誌) 頁面可讓您擷取環境中 EC2 執行個體的日誌。請求日誌時，Elastic Beanstalk 會傳送命令至執行個體，並將日誌上傳至 Amazon S3 內的 Elastic Beanstalk 儲存貯體。於此頁面請求日誌時，Elastic Beanstalk 在 15 分鐘後會自動將日誌從 Amazon S3 刪除。

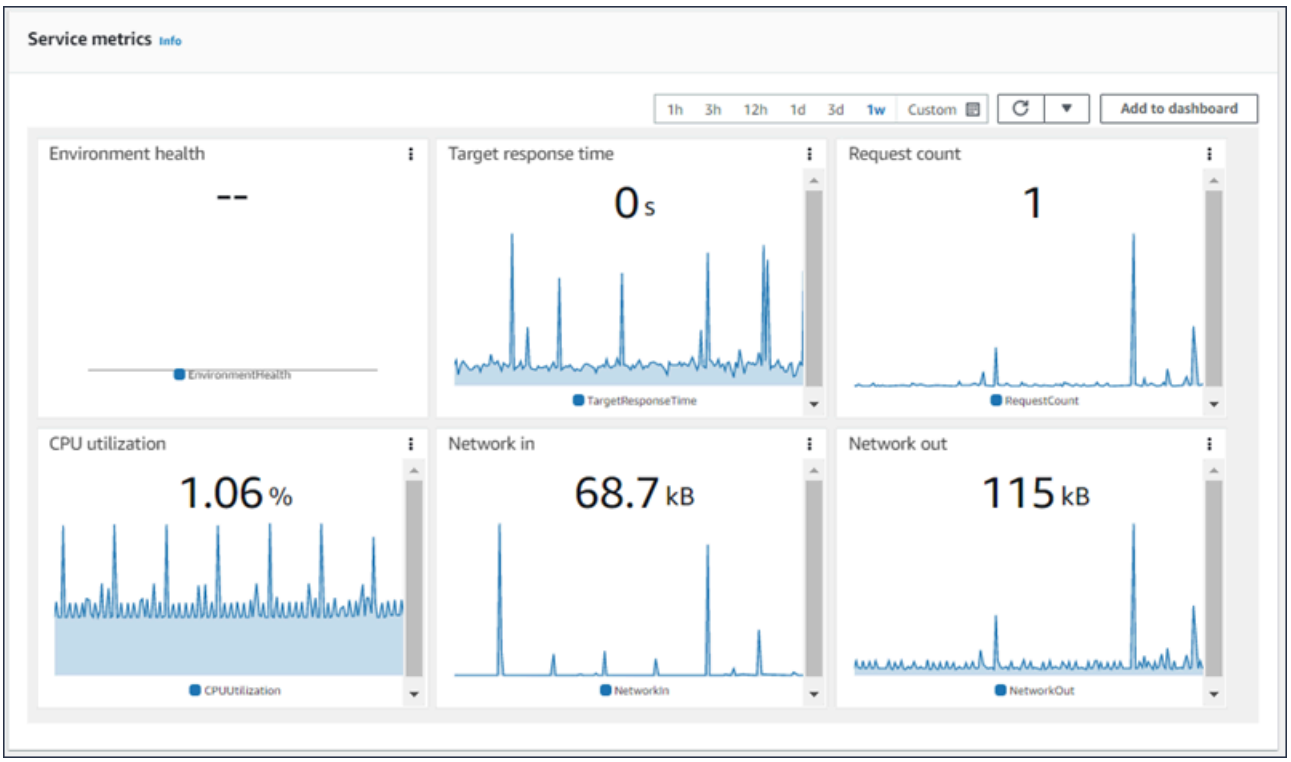
您亦可設定環境的執行個體，將本機輪換後的日誌上傳至 Amazon S3 永久儲存。



如需更多詳細資訊，請參閱 在 Elastic Beanstalk 環境中檢視 Amazon EC2 執行個體的日誌。

監控

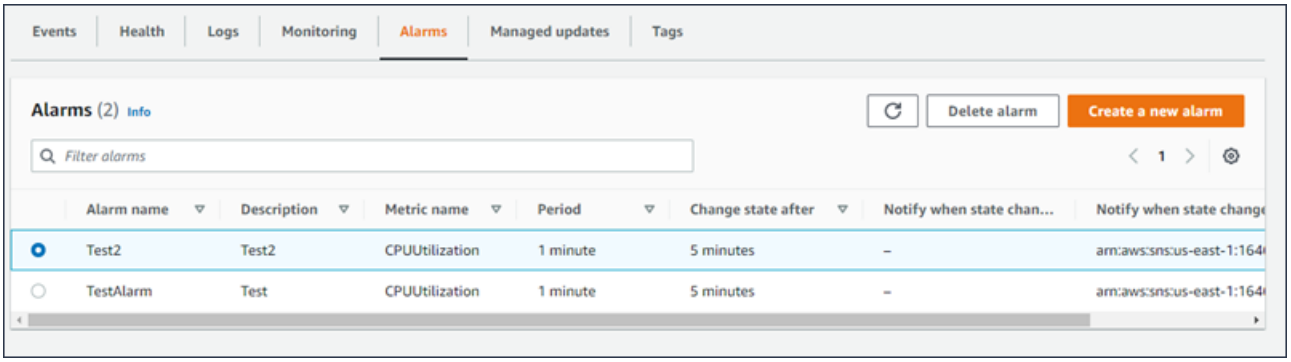
Monitoring (監控) 頁面會顯示您環境的運作狀態資訊概觀。這包括由 Elastic Load Balancing 與 Amazon EC2 提供的預設指標，以及顯示環境運作狀態隨時間變更的圖表。



如需更多詳細資訊，請參閱 在 AWS 管理主控台中監控環境的運作狀態。

警示

Existing alarms (現有警示) 頁面會顯示您為環境設定的任何警示資訊。您可使用此頁面的選項來建立或刪除警示。



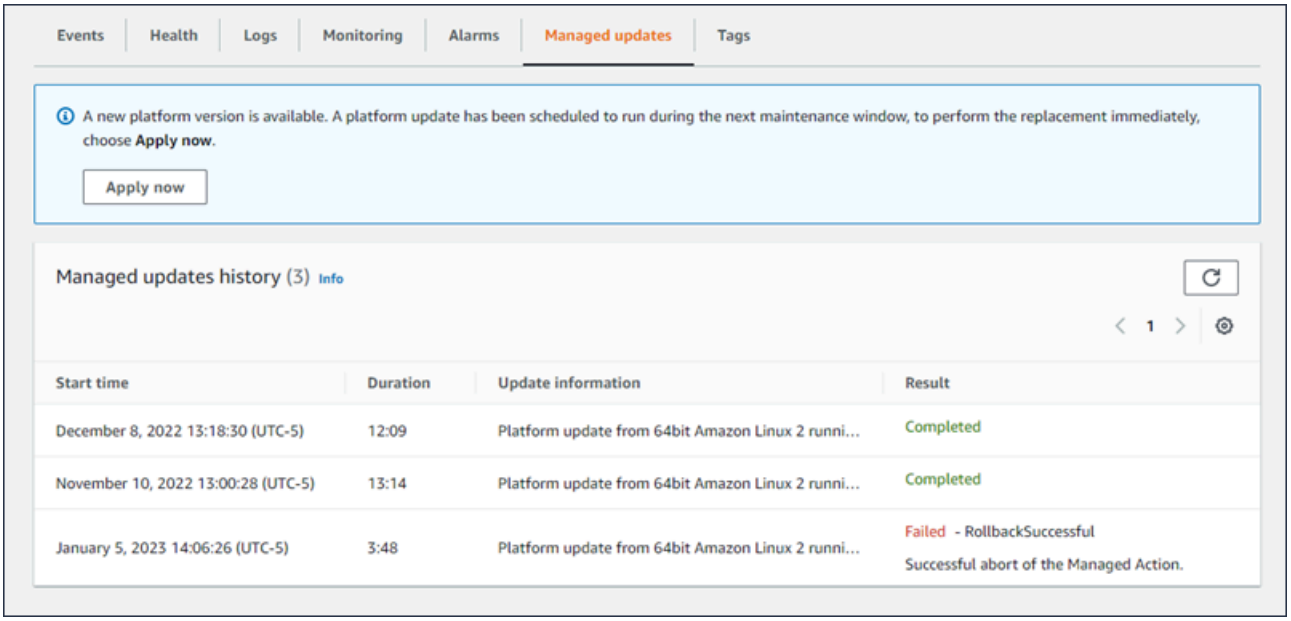
如需更多詳細資訊，請參閱 管理警示。

受管更新

Managed updates overview (受管更新概觀) 頁面會顯示即將到來和已完成之受管平台更新與執行個體替換的資訊。

受管的更新功能可讓您將環境設定為在所選的每週維護時段期間，自動更新至最新的平台版本。在平台版本釋出間，您可選擇讓環境在維護時段期間替換所有 Amazon EC2 執行個體。這可減少應用程式長期間執行所發生的問題。

如需更多詳細資訊，請參閱 受管平台更新。

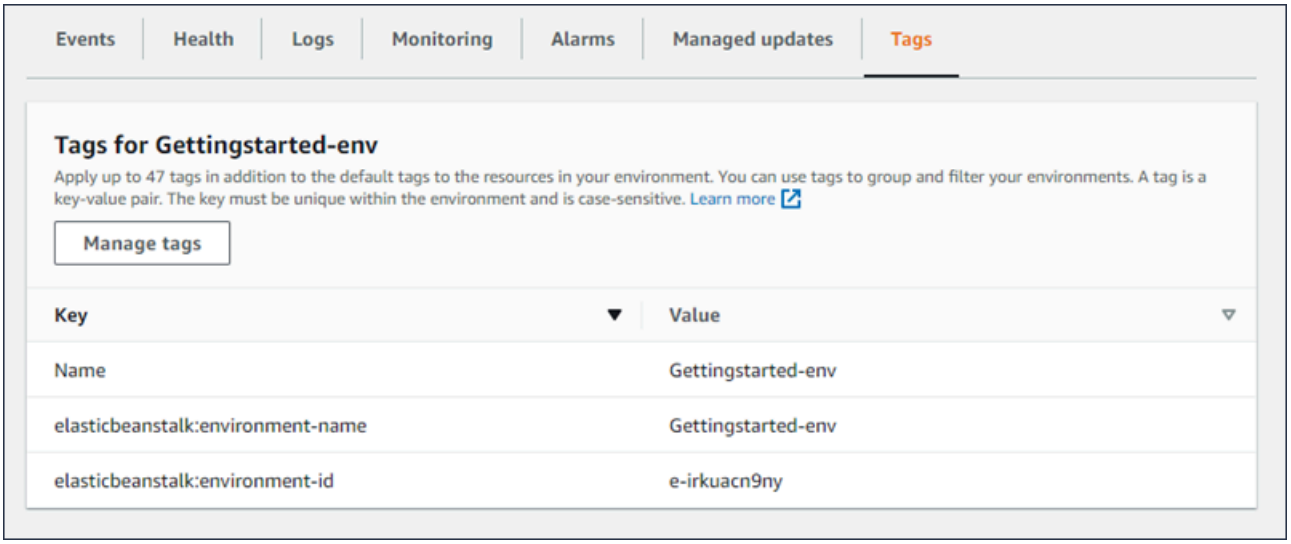


如需更多詳細資訊，請參閱 受管平台更新。

Tags (標籤)

Tags (標籤) 頁面會顯示您建立環境時 Elastic Beanstalk 套用的標籤，以及您新增的任何標籤。您可以新增、編輯和刪除自訂標籤。您無法編輯或刪除 Elastic Beanstalk 套用的標籤。

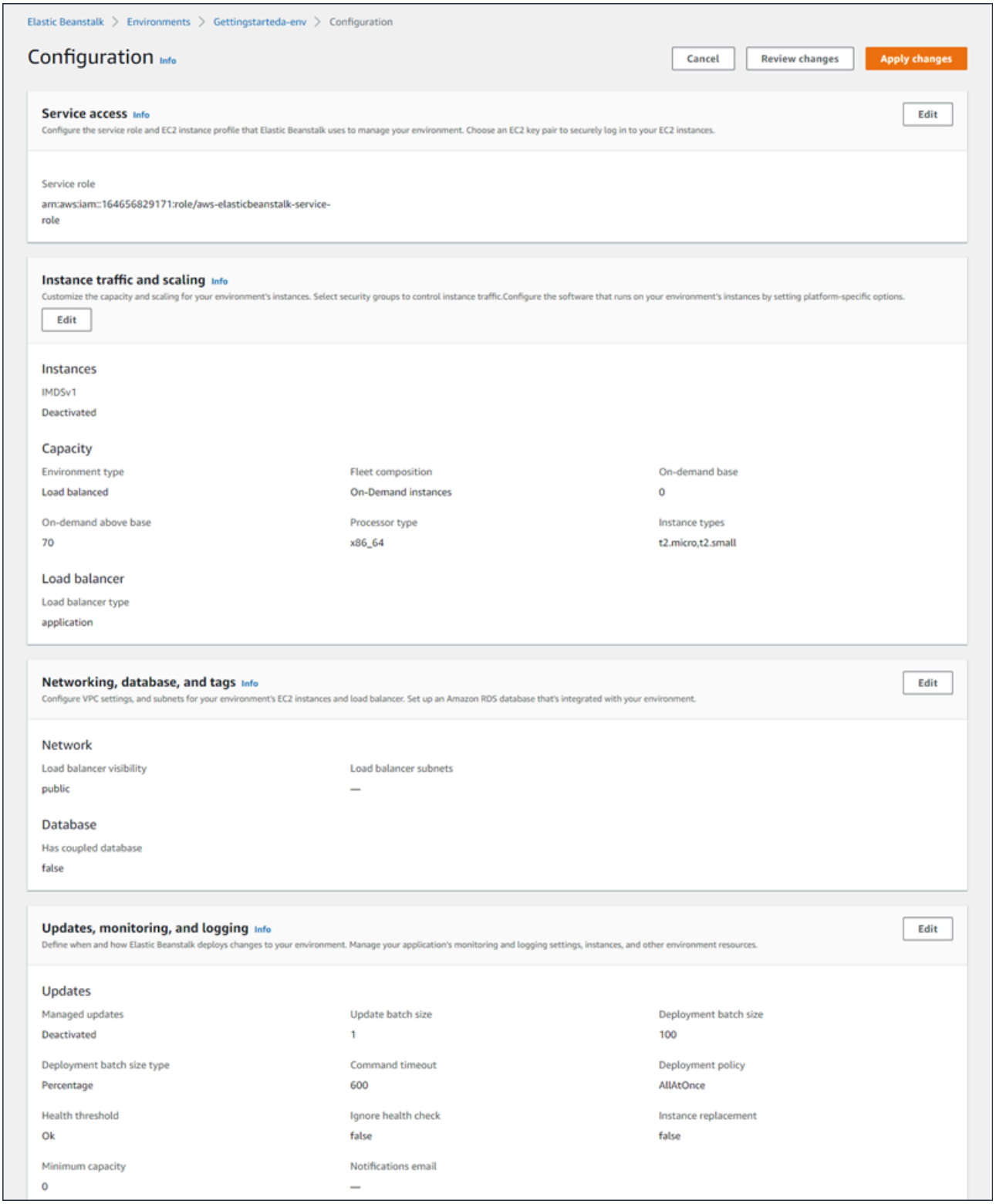
環境標籤會套用至 Elastic Beanstalk 建立的各個資源，以支援您的應用程式。



如需更多詳細資訊，請參閱 在您的 Elastic Beanstalk 環境中標記資源。

組態

Configuration (組態) 頁面會顯示您環境的目前組態及其資源，包括 Amazon EC2 執行個體、負載平衡器、通知和運作狀態監控設定。使用此頁面的設定，在部署期間自訂環境行為、啟用其他功能，並在環境建立期間修改執行個體類型和所選的其他設定。



如需更多詳細資訊，請參閱 [設定 Elastic Beanstalk 環境](#)。

建立 Elastic Beanstalk 環境

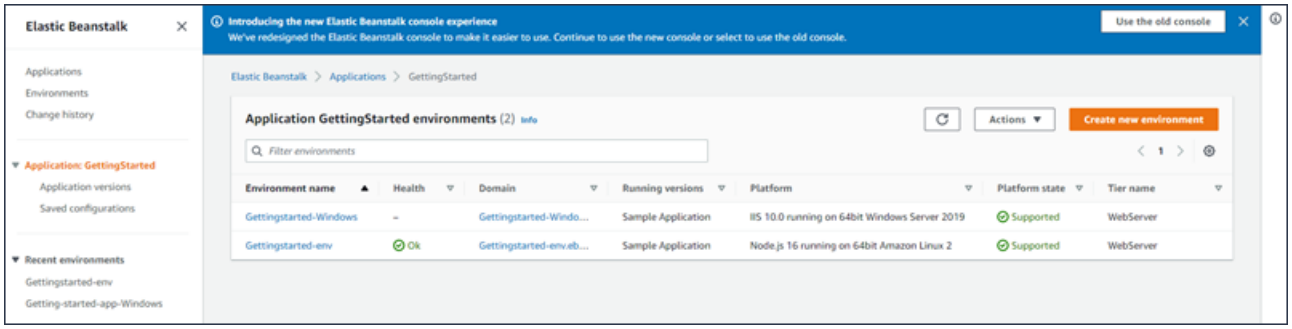
AWS Elastic Beanstalk 環境為執行某個應用程式版本的 AWS 資源的集合。當您需要執行多個版本的應用程式時，您可以部署多個環境。例如，您可能有開發、整合和生產的環境。

下列程序會啟動執行預設應用程式的新環境。這些步驟經過簡化，可使用預設選項值讓您的環境快速啟動並執行。有關包含可用於設定 Elastic Beanstalk 以代表您部署資源的許多選項詳細說明，請參閱建立新的環境精靈。

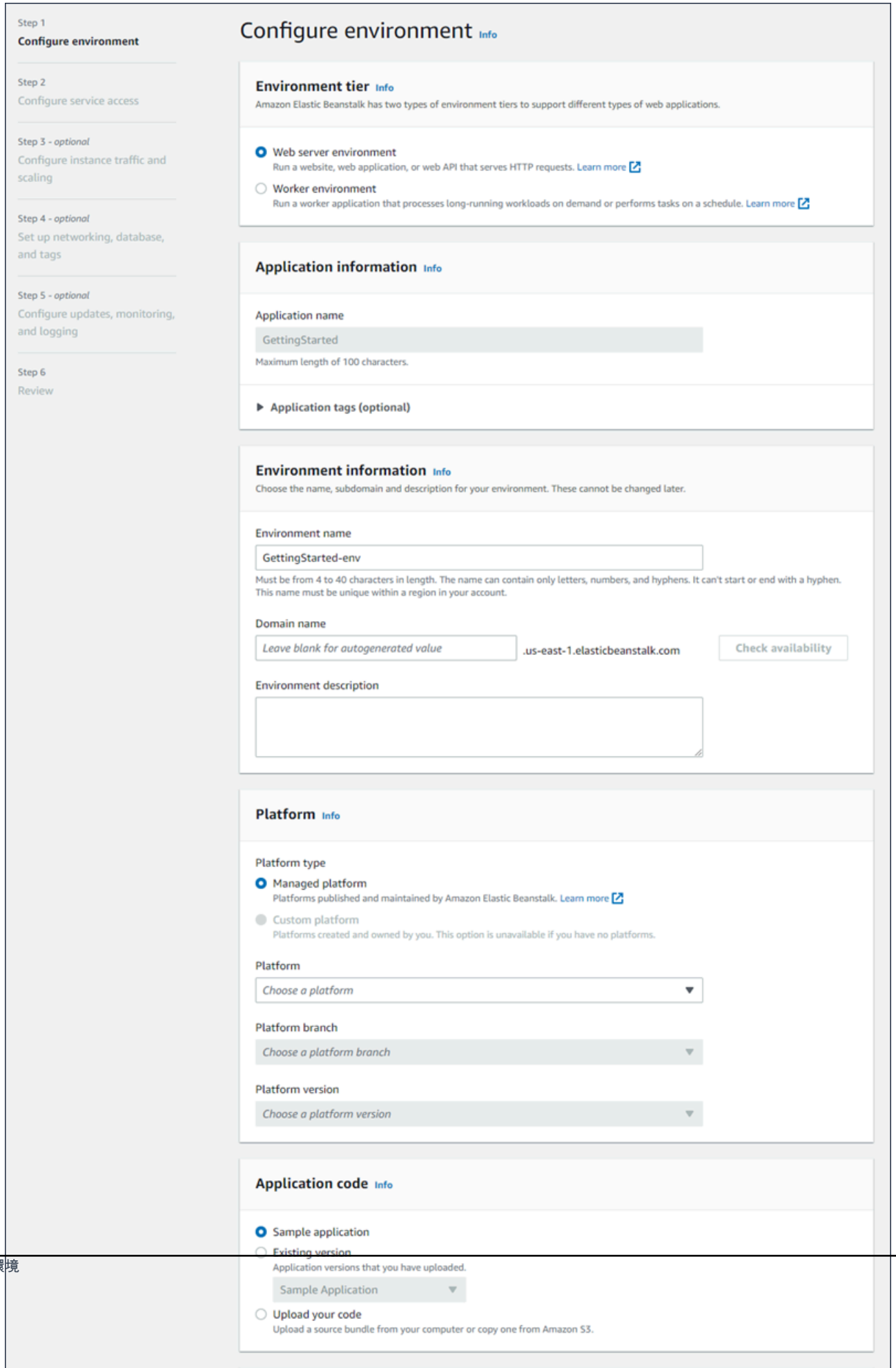
> **備註**
>
> - 如需使用 EB CLI 建立和管理環境的詳細資訊，請參閱 使用 EB CLI 管理 Elastic Beanstalk 環境。
> - 建立環境需要在 Elastic Beanstalk 中完整存取受管政策的許可。如需詳細資訊，請參閱「Elastic Beanstalk 使用者政策」。

欲使用範例應用程式來啟動環境 (主控台)

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇應用程式，然後選擇清單上目前的應用程式名稱或建立一個。
3. 在應用程式概觀頁面上，選擇 Create a new environment (建立新環境)。



這會啟動 Create environment (建立環境) 精靈。精靈提供一組建立新環境的步驟。


Step 1
Configure environment
Step 2
Configure service access
Step 3 - optional
Configure instance traffic and scaling
Step 4 - optional
Set up networking, database, and tags
Step 5 - optional
Configure updates, monitoring, and logging
Step 6
Review
Configure environment Info
Environment tier Info
Amazon Elastic Beanstalk has two types of environment tiers to support different types of web applications.
Web server environment
Run a website, web application, or web API that serves HTTP requests. Learn more
Worker environment
Run a worker application that processes long-running workloads on demand or performs tasks on a schedule. Learn more
Application information Info
Application name
GettingStarted
Maximum length of 100 characters.
Application tags (optional)
Environment information Info
Choose the name, subdomain and description for your environment. These cannot be changed later.
Environment name
GettingStarted-env
Must be from 4 to 40 characters in length. The name can contain only letters, numbers, and hyphens. It can't start or end with a hyphen. This name must be unique within a region in your account.
Domain name
Leave blank for autogenerated value
.us-east-1.elasticbeanstalk.com
Check availability
Environment description
Platform Info
Platform type
Managed platform
Platforms published and maintained by Amazon Elastic Beanstalk. Learn more
Custom platform
Platforms created and owned by you. This option is unavailable if you have no platforms.
Platform
Choose a platform
Platform branch
Choose a platform branch
Platform version
Choose a platform version
Application code Info
Sample application
Existing version
Application versions that you have uploaded.
Sample Application
Upload your code
Upload a source bundle from your computer or copy one from Amazon S3.

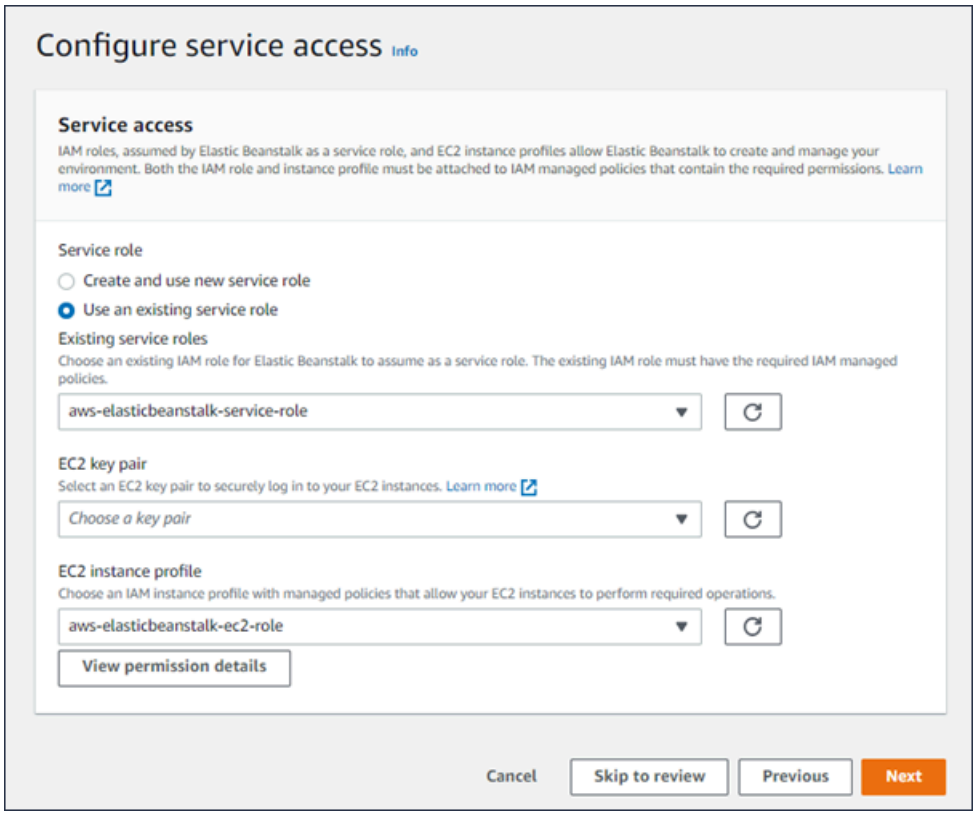
4. 關於環境階層，選擇 Web server environment (Web 伺服器環境) 或 Worker environment (工作者環境) 環境階層。建立後您即無法變更環境層。

> **Note**
>
> Windows Server 平台上的 .NET 不支援工作者環境層。

5. 在 Platform (平台)，選取符合您應用程式所使用語言的平台和平台分支。

> **Note**
>
> Elastic Beanstalk 支援所列出大多數平台的多個版本。根據預設，主控台會針對您選擇的平台和平台分支，選取建議的版本。如果您的應用程式需要不同的版本，您可以在這裡選取。如需支援的平台版本的相關詳細資訊，請參閱 the section called “支援的平台”。

6. 針對 Application code (應用程式程式碼)，選擇 Sample application (範例應用程式)。
7. 關於 Configuration presets (組態預設)，選擇 Single instance (單一執行個體)。
8. 選擇 Next (下一步)。
9. 畫面上會顯示設定服務存取頁面。



10. 針對服務角色，選擇使用現有服務角色。
11. 接下來，我們會將重點放在 EC2 執行個體設定檔下拉式清單。此下拉式清單中顯示的值可能會有所不同，視您的帳戶是否先前已建立新環境而定。

    根據清單中顯示的值，選擇下列其中一項。

    - 如果下拉式清單中顯示 `aws-elasticbeanstalk-ec2-role`，請從 EC2 執行個體設定檔下拉式清單中選取它。
    - 如果清單中顯示另一個值，而且它是適用於您環境的預設 EC2 執行個體設定檔，請從 EC2 執行個體設定檔下拉式清單中選取該值。
    - 如果 EC2 執行個體設定檔下拉式清單未顯示任何可供選擇的值，請展開隨後的程序，為 EC2 執行個體設定檔建立 IAM 角色。

      完成為 EC2 執行個體設定檔建立 IAM 角色中的步驟來建立 IAM 角色，接下來，您可以為 EC2 執行個體設定檔選取該角色。然後，請返回此步驟。

      現在您已建立 IAM 角色，重新整理清單，該角色就會在下拉式清單中顯示為選項。從 EC2 執行個體設定檔下拉式清單中選取您剛建立的 IAM 角色。
12. 在 Configure service access (設定服務存取)頁面上選擇 Skip to Review (略過以檢閱)。

    這會選取此步驟的預設值，並略過選擇性步驟。
13. Review (檢閱)頁面會顯示您所有選擇的摘要。

    若要進一步自訂您的環境，請選擇步驟旁的 Edit (編輯)，其中包含您想要設定的任何項目。下列選項僅能於環境建立期間進行設定：

    - 環境名稱
    - 網域名稱
    - 平台版本
    - 處理器
    - VPC
    - 層

    下列設定可於環境建立後變更，但需要佈建新的執行個體或其他資源，且套用時間可能較長：

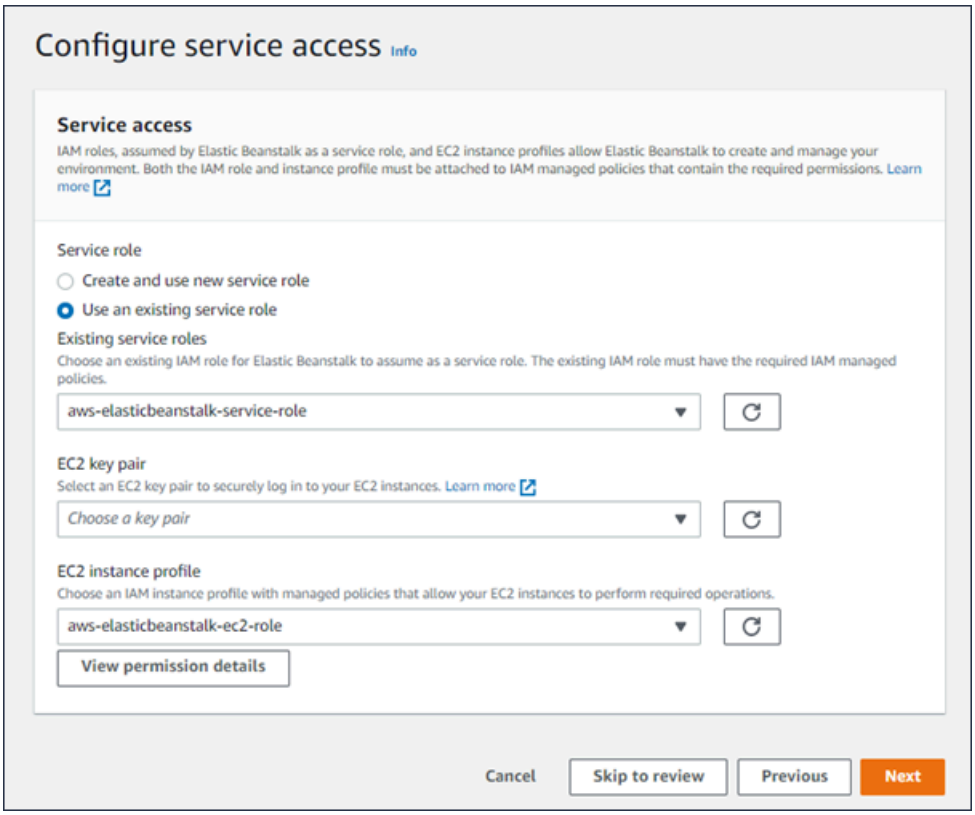
    - 執行個體類型、根磁碟區、金鑰對和 AWS Identity and Access Management (IAM) 角色
    - 內部 Amazon RDS 資料庫

- 負載平衡器

如需所有可用設定的詳細資訊，請參閱建立新的環境精靈。

14. 選擇頁面底部的 Submit (提交)，以將建立的新環境初始化。

為 EC2 執行個體設定檔建立 IAM 角色



若要建立 EC2 執行個體設定檔的 IAM 角色

1. 選擇檢視許可詳細資料。這會顯示在 EC2 執行個體設定檔下拉式清單下。

   畫面上會顯示標題為檢視執行個體設定檔許可的模態視窗。此視窗會列出您需要附加至您建立的新 EC2 執行個體設定檔的受管設定檔。它也提供啟動 IAM 主控台的連結。

2. 選擇顯示於視窗頂端的 IAM 主控台連結。
3. 在 IAM 主控台導覽窗格中，選擇 Roles (角色)。
4. 選擇 建立角色。
5. 在受信任的實體類型下，選擇 AWS 服務。
6. 在 Use case (使用案例) 下，選擇 EC2。

7. 選擇 Next (下一步)。
8. 附加合適的受管政策。在檢視執行個體設定檔許可模態視窗中捲動，查看受管政策。這些政策也會列在此處：
   - `AWSElasticBeanstalkWebTier`
   - `AWSElasticBeanstalkWorkerTier`
   - `AWSElasticBeanstalkMulticontainerDocker`
9. 選擇 Next (下一步)。
10. 輸入角色的名稱。
11. (選用) 附加標籤至角色。
12. 選擇 建立角色。
13. 返回開啟的 Elastic Beanstalk 主控台視窗。
14. 關閉檢視執行個體設定檔許可模態視窗

> **Important**
>
> 請勿關閉顯示 Elastic Beanstalk 主控台的瀏覽器頁面。

15. 選擇 EC2 執行個體設定檔下拉式清單旁的

    (重新整理)。

    這會重新整理下拉式清單，讓您剛建立的「角色」會顯示在下拉式清單中。

雖然 Elastic Beanstalk 會建立您的環境，您會被重新引導至 Elastic Beanstalk 主控台。當環境運作狀態顯示綠色時，請選擇環境名稱旁的 URL 以檢視執行中的應用程式。此 URL 通常可從網際網路存取，除非您將環境設定為使用內部負載平衡器的自訂 VPC。

主題

- 建立新的環境精靈
- 複製 Elastic Beanstalk 環境
- 終止 Elastic Beanstalk 環境
- 使用 AWS CLI 建立 Elastic Beanstalk 環境
- 使用 API 建立 Elastic Beanstalk 環境

- 建構立即啟動 URL
- 建立和更新 Elastic Beanstalk 環境的群組

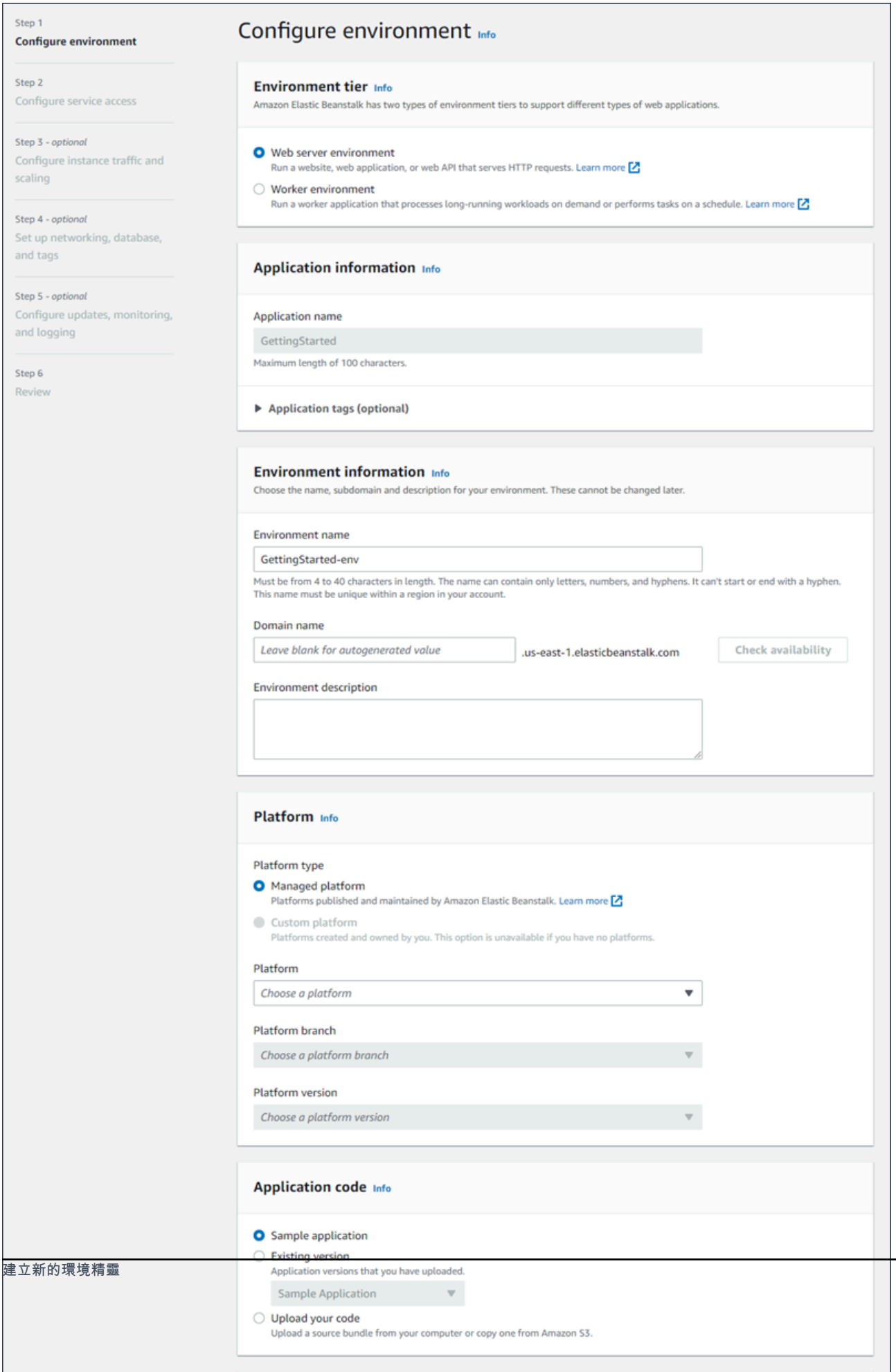
建立新的環境精靈

在 建立 Elastic Beanstalk 環境 中，我們示範了如何開啟 Create new environment (建立新環境) 精靈並快速建立環境。選擇 Create environment (建立環境) 可啟動具備預設環境名稱、自動產生的網域、範例應用程式碼及建議設定的環境。

此主題說明 Create environment (建立環境) 精靈，以及您能用來設定您要建立的環境的所有方式。

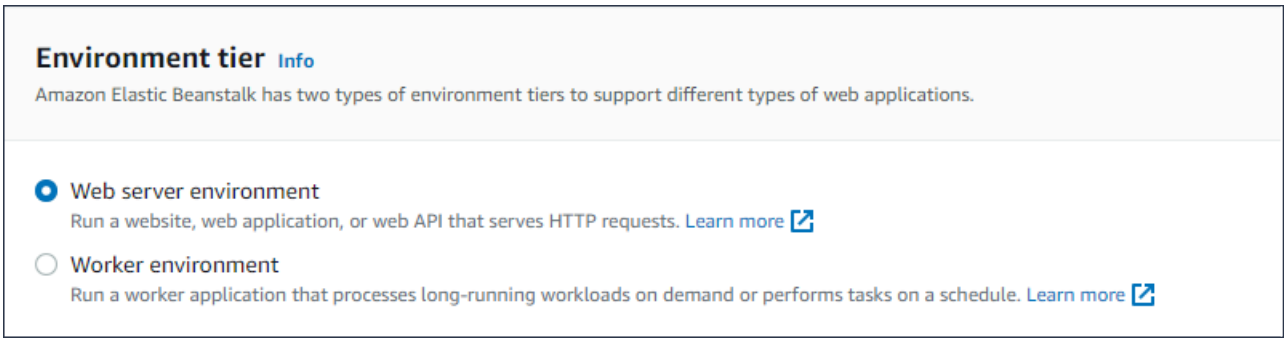
精靈頁面

Create environment (建立環境) 精靈提供一組建立新環境的步驟。


Step 1
Configure environment
Step 2
Configure service access
Step 3 - optional
Configure instance traffic and scaling
Step 4 - optional
Set up networking, database, and tags
Step 5 - optional
Configure updates, monitoring, and logging
Step 6
Review
Configure environment Info
Environment tier Info
Amazon Elastic Beanstalk has two types of environment tiers to support different types of web applications.
Web server environment
Run a website, web application, or web API that serves HTTP requests. Learn more
Worker environment
Run a worker application that processes long-running workloads on demand or performs tasks on a schedule. Learn more
Application information Info
Application name
GettingStarted
Maximum length of 100 characters.
Application tags (optional)
Environment information Info
Choose the name, subdomain and description for your environment. These cannot be changed later.
Environment name
GettingStarted-env
Must be from 4 to 40 characters in length. The name can contain only letters, numbers, and hyphens. It can't start or end with a hyphen. This name must be unique within a region in your account.
Domain name
Leave blank for autogenerated value
.us-east-1.elasticbeanstalk.com
Check availability
Environment description
Platform Info
Platform type
Managed platform
Platforms published and maintained by Amazon Elastic Beanstalk. Learn more
Custom platform
Platforms created and owned by you. This option is unavailable if you have no platforms.
Platform
Choose a platform
Platform branch
Choose a platform branch
Platform version
Choose a platform version
Application code Info
Sample application
Existing version
Application versions that you have uploaded.
Sample Application
Upload your code
Upload a source bundle from your computer or copy one from Amazon S3.

環境層

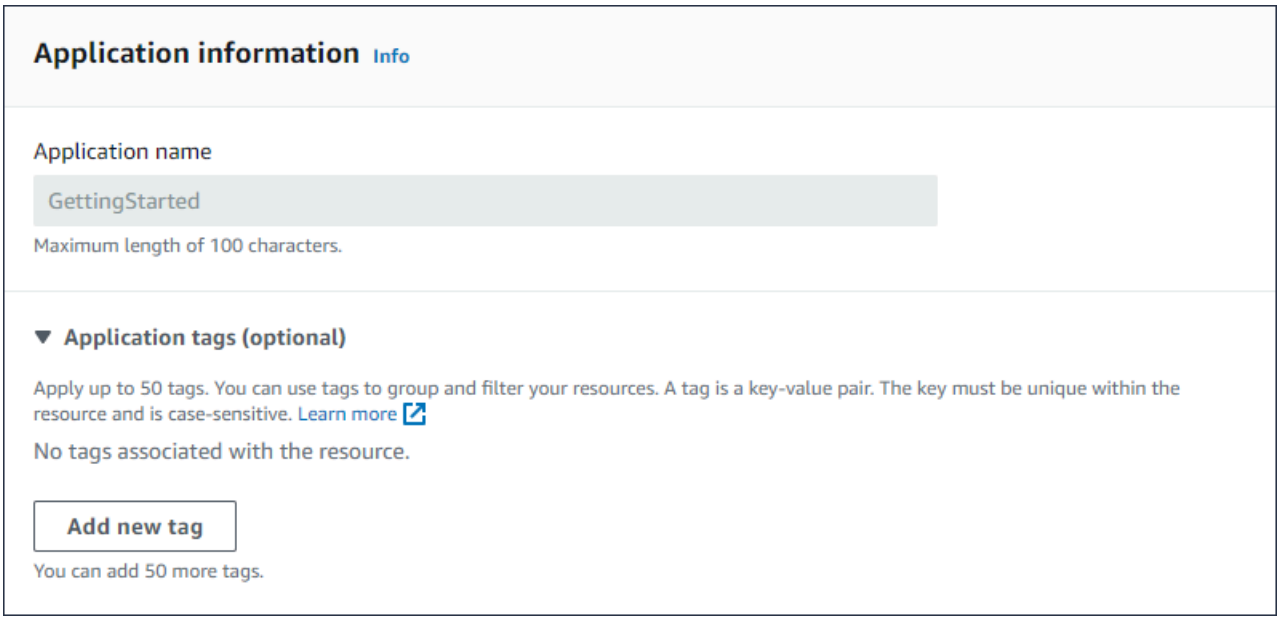
關於 environment tier (環境階層)，選擇 Web server environment (Web 伺服器環境) 或 Worker environment (工作者環境) 環境階層。建立後您即無法變更環境層。



> **Note**
>
> Windows Server 平台上的 .NET 不支援工作者環境層。

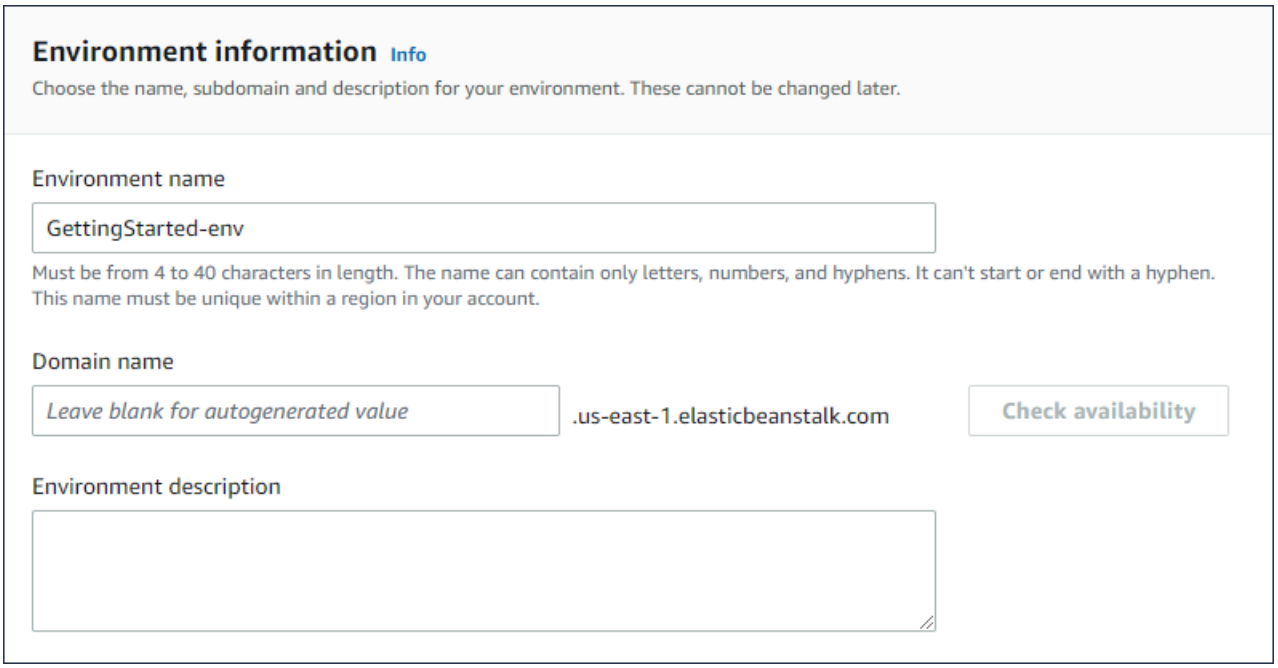
應用程式資訊

如果您是從 Application overview (應用程式概觀) 頁面選取 Create new environment (建立新環境) 來啟動精靈，則會預先填入 Application name (應用程式名稱)。否則，請輸入應用程式名稱。或者可以選擇新增 application tags (應用程式索引標籤)。



環境資訊

設定環境的名稱和網域，並建立您環境的描述。請注意，這些環境設定無法變更建立後的環境。



- Name (名稱) - 輸入環境的名稱。該表單會提供產生的名稱。
- Domain (網域) - (Web 伺服器環境) 輸入您環境的唯一網域名稱。預設名稱是環境的名稱。您可以輸入不同的網域名稱。Elastic Beanstalk 會使用此名稱來建立環境的唯一 CNAME。若要檢查所需網域名稱是否可用，請選擇 Check Availability (檢查可用性)。
- Description (描述) - 輸入此環境的描述。

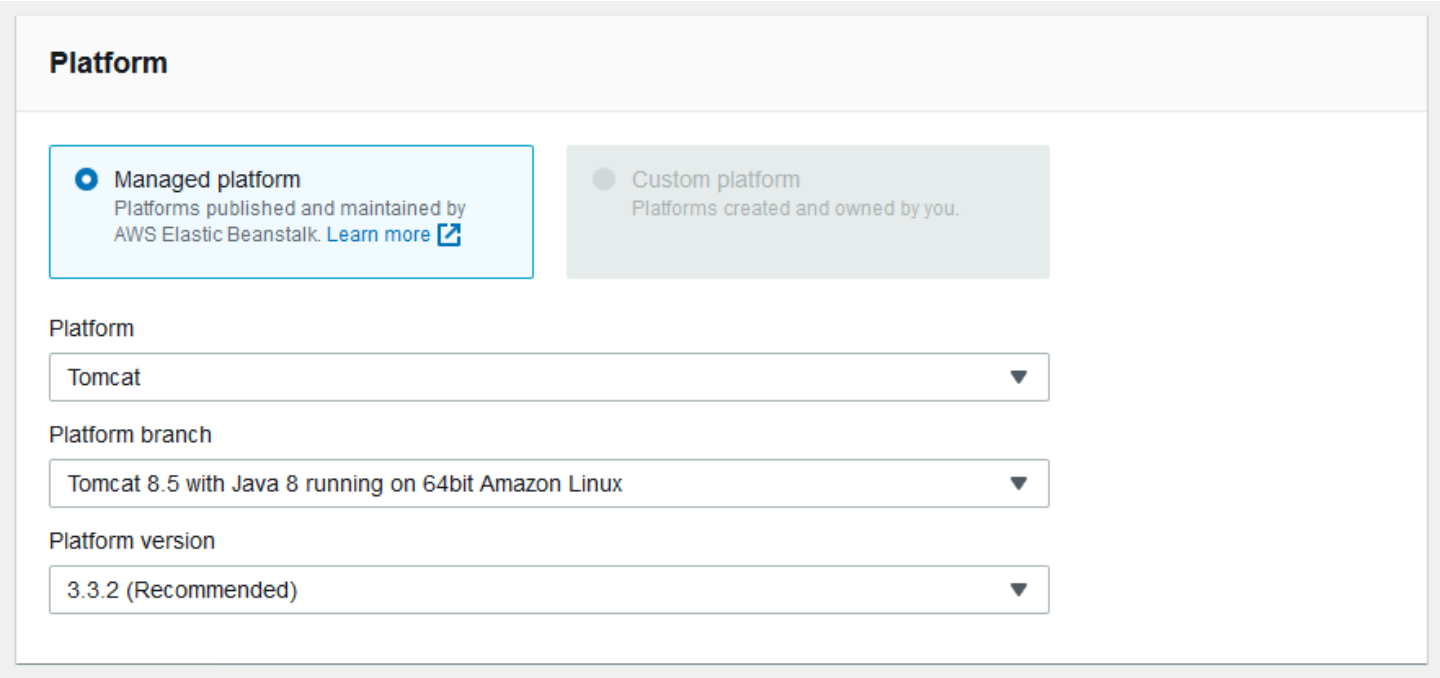
選擇新環境的平台

您可自兩種平台類型建立新的環境：

- 受管平台
- 自訂平台

受管平台

在大多數情況下，您會針對新環境使用 Elastic Beanstalk 受管平台。當新的環境精靈啟動後，預設會選取 Managed platform (受管平台) 選項。



選取平台、該平台內的平台分支，以及分支中的特定平台版本。當您選取平台分支時，依預設會選取分支內的建議版本。此外，您可以選擇以前使用過的任何平台版本。

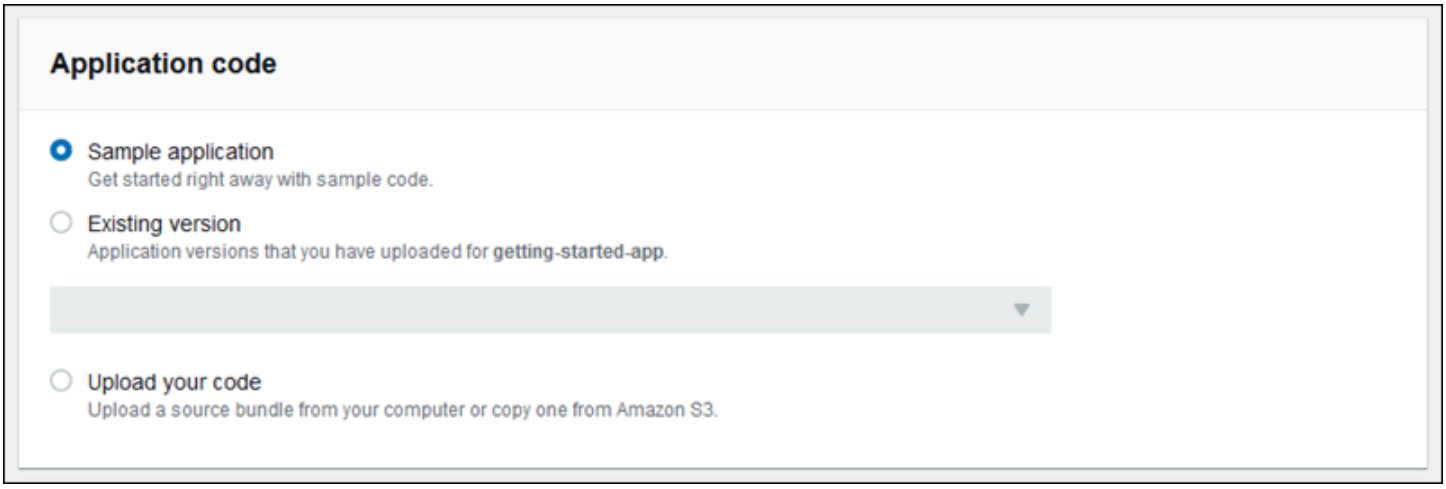
> **Note**
>
> 對於生產環境，建議您在支援的平台分支中選擇平台版本。如需有關平台分支狀態的詳細資訊，請參閱 the section called “平台詞彙表” 中的平台分支定義。

自訂平台

若立即可用的平台不符您的需求，您可以透過自訂平台建立新的環境。若要指定自訂平台，請選擇 Custom platform (自訂平台) 選項，然後選取其中一個可用的自訂平台。若沒有可用自訂平台，此選項會顯示為灰色。

提供應用程式的程式碼

您現已選擇欲使用的平台，下一步是提供應用程式的程式碼。

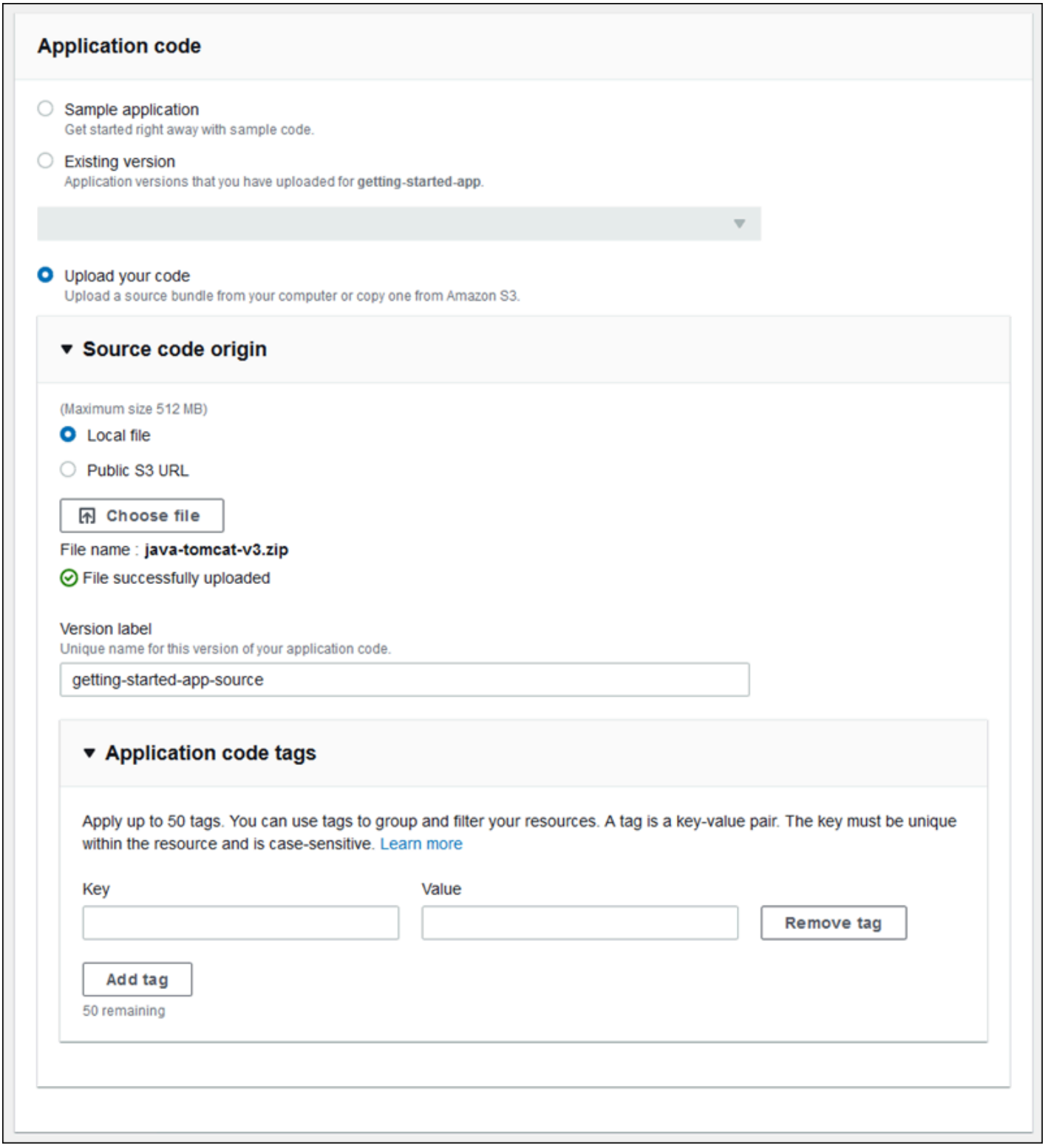


您有多種選擇：

- 您可以使用 Elastic Beanstalk 為每個平台提供的範例應用程式。
- 您可以使用已部署到 Elastic Beanstalk 的程式碼。選擇 Existing version (現有的版本) 和您在 Application code (應用程式的程式) 區段中的應用程式。
- 您可以上傳新的程式碼。選擇 Upload your code (上傳您的程式碼)，然後選擇 Upload (上傳)。您可以從本機檔案上傳新的應用程式的程式碼，或您可以指定內含應用程式的程式碼的 Amazon S3 儲存貯體的 URL。

> **Note**
>
> 您可依據選取的平台版本，以 ZIP 來源套件、WAR 檔案或純文字 Docker 組態等格式上傳應用程式。檔案大小上限為 500 MB。

選擇上傳新程式碼時，您也可以提供要與您的新程式碼關聯的標籤。若需有關標記應用程式版本的詳細資訊，請參閱 the section called “標記應用程式版本”。



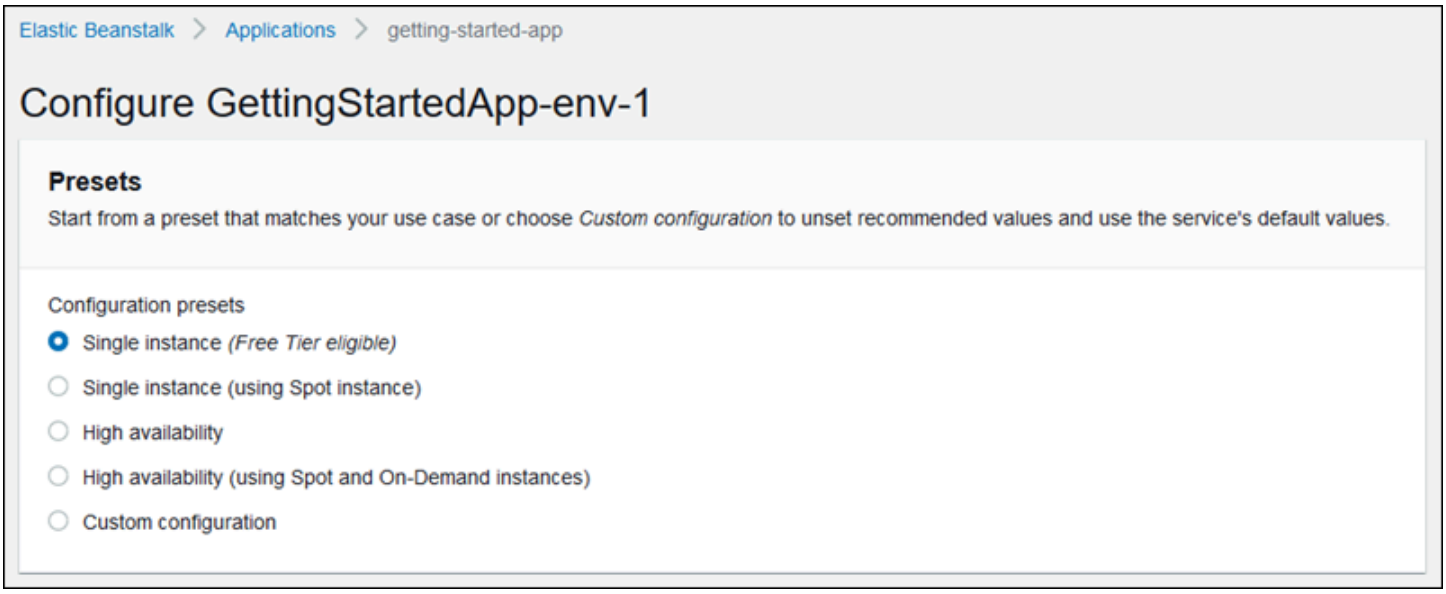
若要使用預設組態選項快速建立環境，您現在可以選擇 Create environment (建立環境)。選擇 Configure more options (設定多個選項)，做額外的組態變更，如以下各節所示。

精靈組態頁面

當您選擇 Configure more options (設定更多選項) 時，精靈會顯示 Configure (設定) 頁面。您可以在此頁面選取組態預設集、變更您要環境使用的平台版本，或是針對新環境進行特定的組態選擇。

選擇預設集組態

在頁面的 Presets (預設集) 區段中，Elastic Beanstalk 會針對不同使用案例提供數個組態預設集。每種預設均包含數個組態選項的建議值。



高可用性預設包含一個負載平衡器，建議用於生產環境。如果您想要可執行多個執行個體以達到高可用性並進行擴展以回應載入的環境，請使用這些預設。Single instance (單一執行個體) 預設主要建議用於開發。這些預設的其中兩個會啟用 Spot 執行個體請求。如需 Elastic Beanstalk 容量組態的詳細資訊，請參閱Auto Scaling 群組。

最後一個預設為 Custom configuration (自訂組態)，會移除角色設定外的所有建議值，並使用 API 預設值。若您部署的原始碼套件具備設定組態選項的組態檔案，請選擇此選項。Custom configuration (自訂組態) 也會自動選定若您修改 Low cost (低成本) 或 High availability (高可用性) 組態預設。

自訂您的組態

除了 (或改為) 選擇組態預設，您可以微調環境中的組態選項。Configure (組態) 精靈會顯示數個組態類別。每個組態類別顯示一群組態設定值的摘要。選擇 Edit (編輯) 以編輯此組設定。

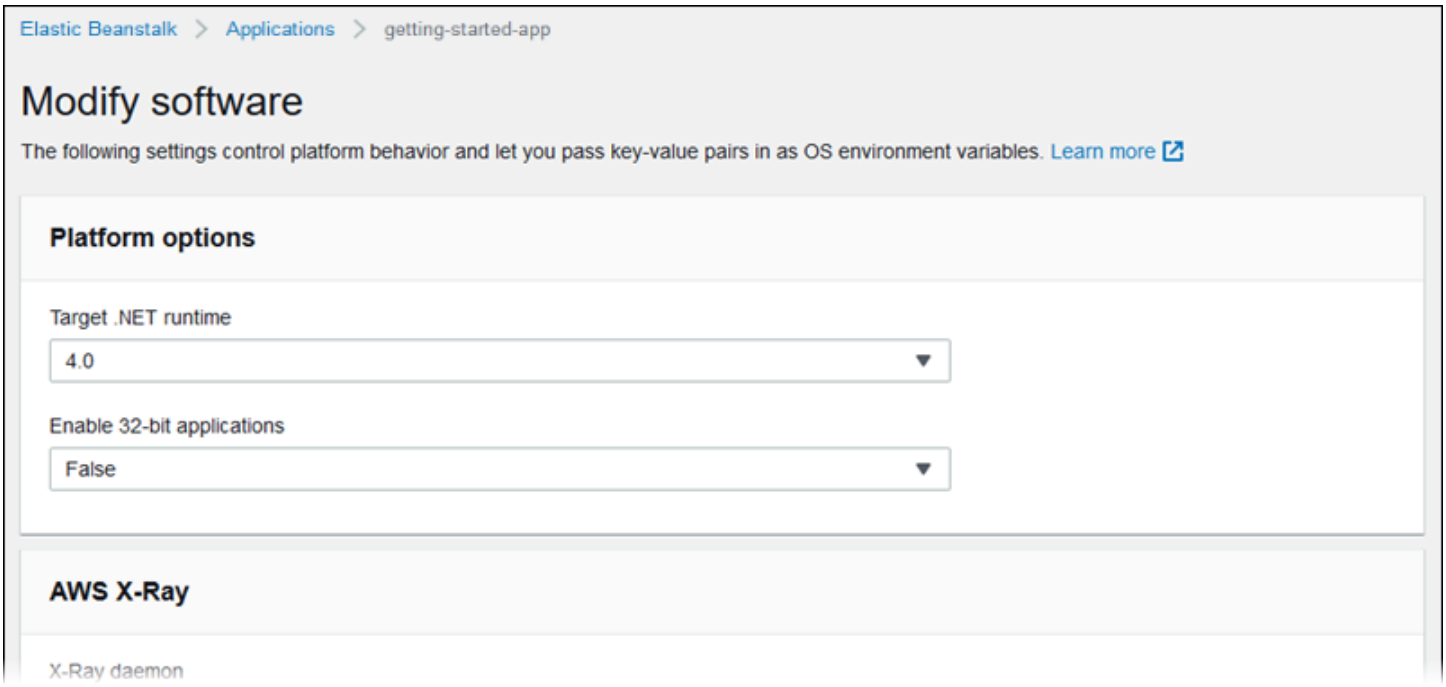
組態類別

- 軟體設定

- 執行個體
- 容量
- 負載平衡器
- 滾動更新和部署
- 安全性
- 監控
- 受管更新
- 通知
- 網路
- 資料庫
- Tags (標籤)
- 工作者環境

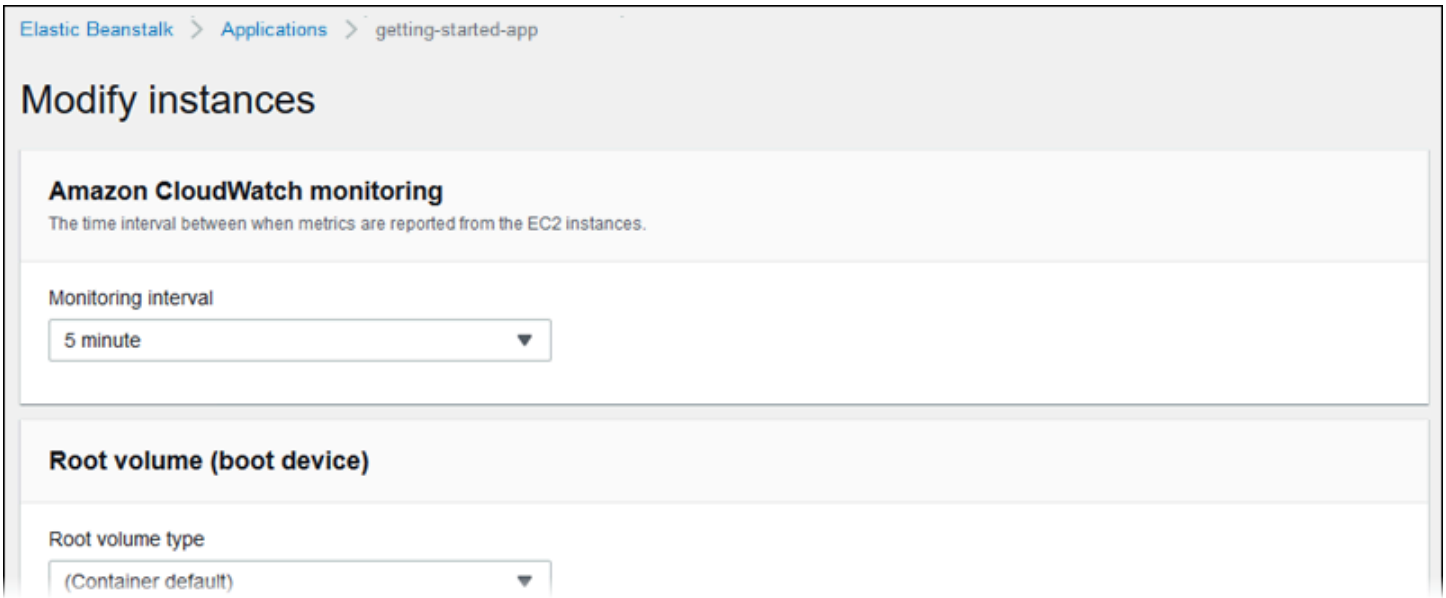
軟體設定

使用 Modify software (修改軟體) 組態頁面可在執行應用程式的 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體上設定軟體。您可以設定環境屬性、AWS X-Ray 除錯、執行個體日誌儲存和串流，以及平台特定的設定。如需詳細資訊，請參閱the section called “環境屬性與軟體設定”。



執行個體

您可以在 Modify instances (修改執行個體) 頁面設定執行您應用程式的 Amazon EC2 執行個體。如需詳細資訊，請參閱the section called “Amazon EC2 執行個體”。



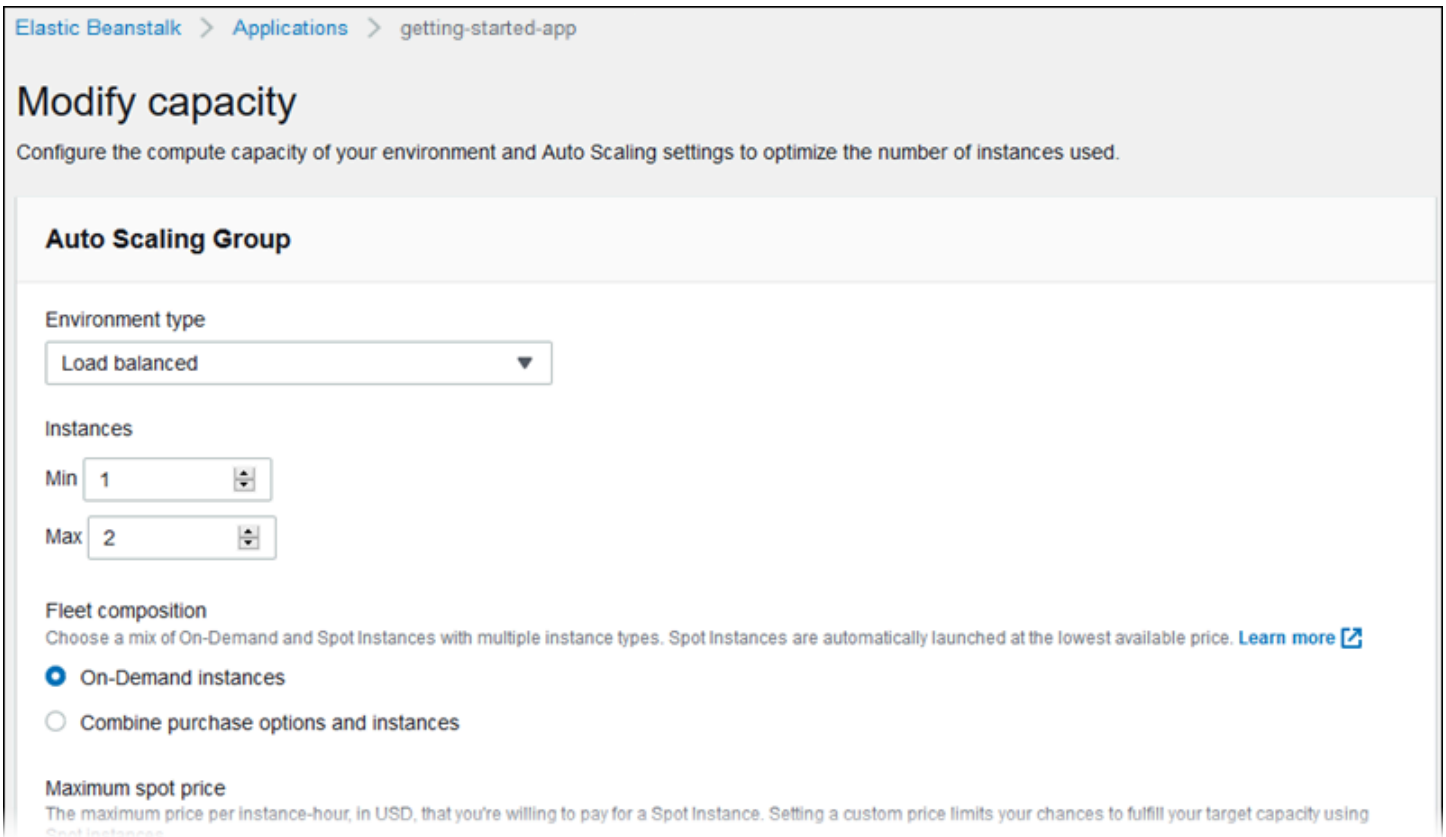
容量

您可以使用 Modify capacity (修改容量) 組態頁面來設定環境的運算容量和 Auto Scaling group (Auto Scaling 群組) 設定，以最佳化您使用的執行個體數量和類型。您也可以根據觸發程序或排程變更環境容量。

負載平衡環境可執行多個執行個體，實現高可用性，並避免組態更新及部署期間停機。在負載平衡環境中，網域名稱會對應至負載平衡器。在單一執行個體環境中，網域名稱則會對應至執行個體上的彈性 IP 地址。

> **Warning**
>
> 單一執行個體環境並非可立即生產。如果執行個體在部署期間變得不穩定，或者 Elastic Beanstalk 在組態更新期間終止並重新啟動執行個體，您的應用程式可能會有一段時間無法使用。使用單一執行個體環境來開發、測試或封測。使用負載平衡環境來生產。

如需環境容量設定的詳細資訊，請參閱the section called “Auto Scaling 群組”和the section called “Amazon EC2 執行個體”。

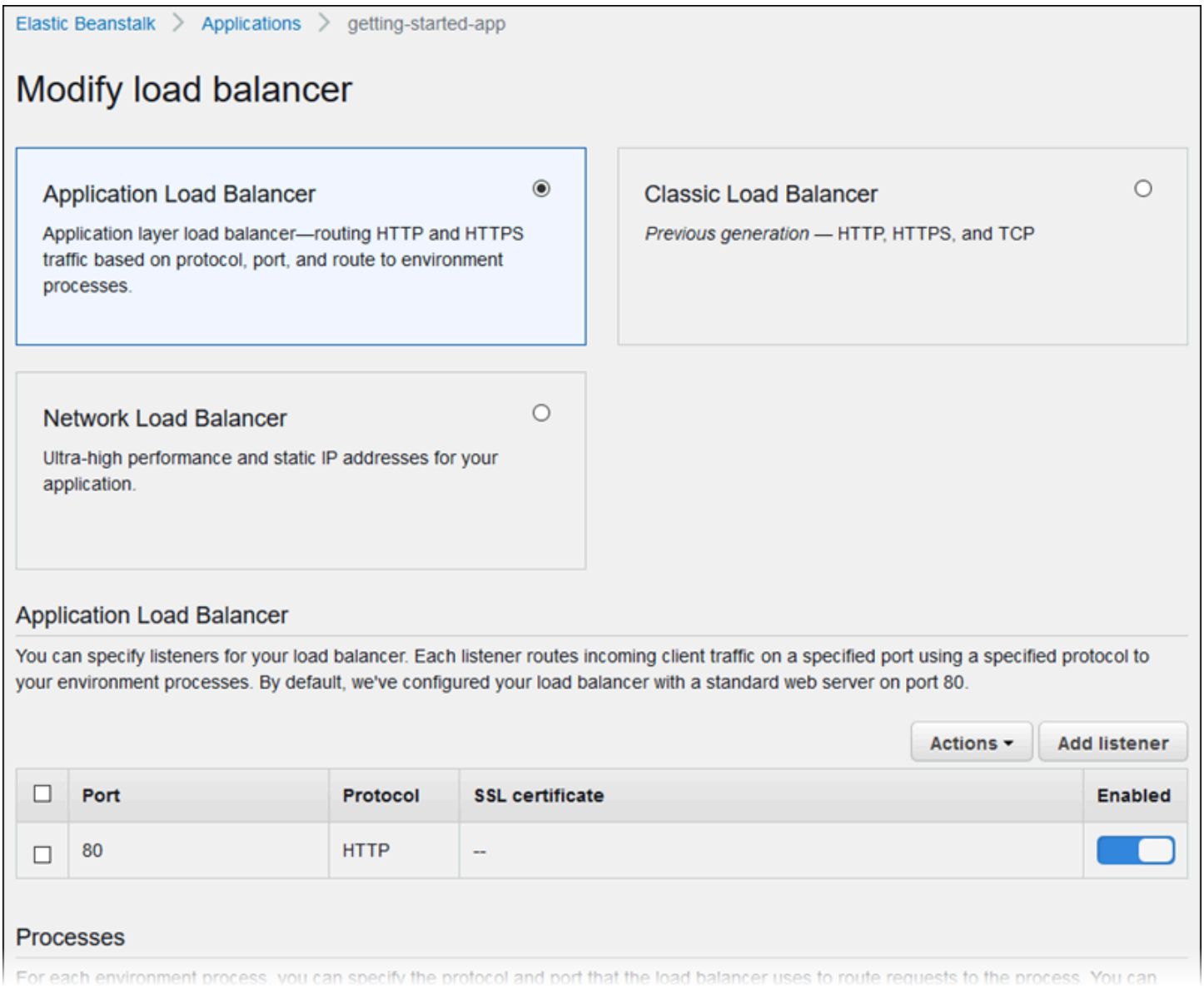


負載平衡器

使用 Modify load balancer (修改負載平衡器) 組態頁面選取負載平衡器類型，並其配置設定。在負載平衡的環境中，您環境的負載平衡器是前往您應用程式的流量的進入點。Elastic Beanstalk 支援多種類型的負載平衡器。依預設，Elastic Beanstalk 主控台會建立 Application Load Balancer，並將其設定為在連接埠 80 上提供 HTTP 流量服務。

> **Note**
>
> 您只能在環境建立期間選取環境的負載平衡器類型。

如需負載平衡器類型和設定的詳細資訊，請參閱 the section called “負載平衡器” 和 the section called “HTTPS”。

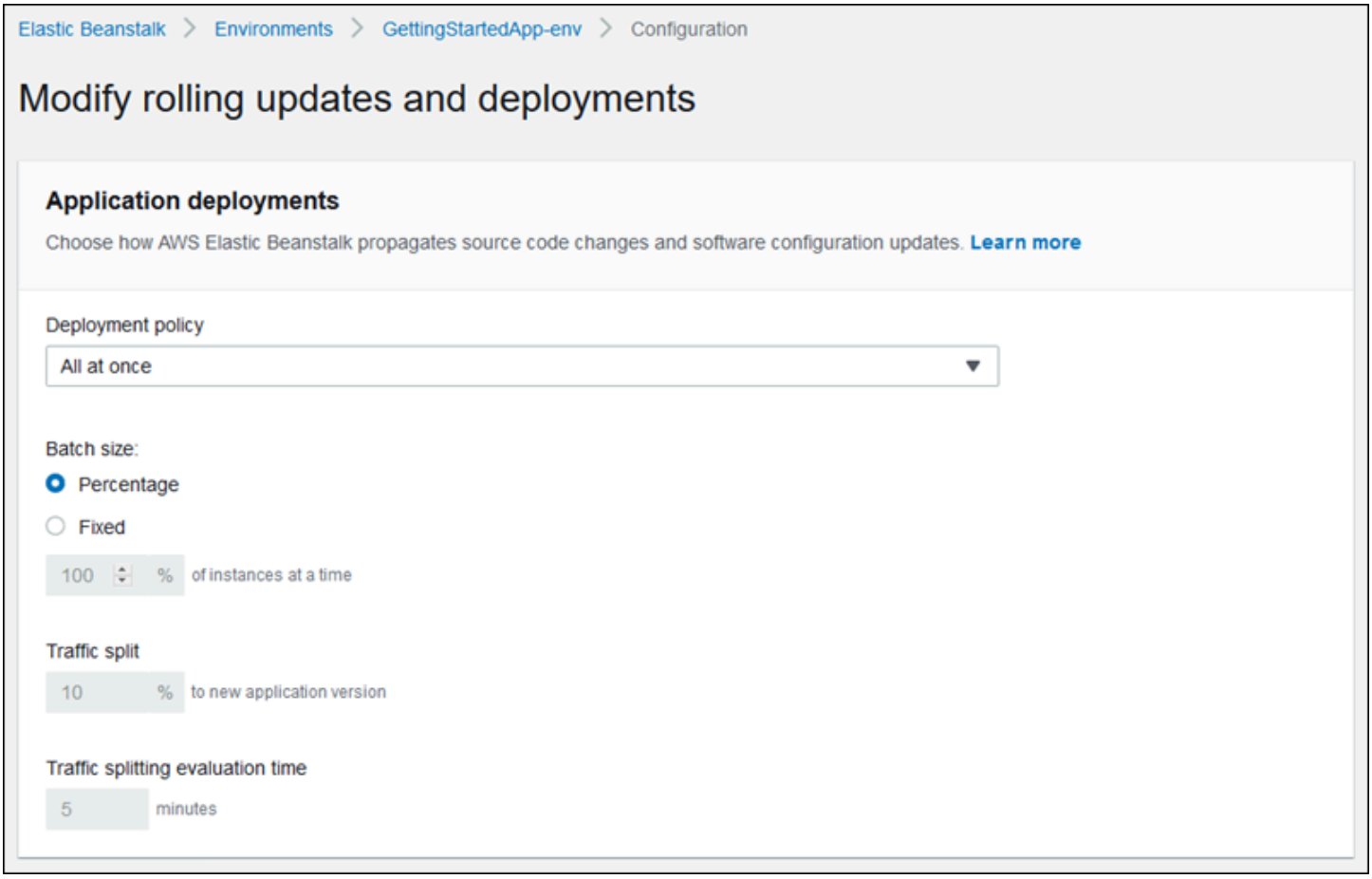


> **Note**
>
> 建立環境主控台精靈上的 Classic Load Balancer (CLB) 選項已停用。如果您已使用 Classic Load Balancer 設定現有環境，則可以使用 Elastic Beanstalk 主控台或 EB CLI 來 複製現有環境以建立新的環境。您也可以選擇使用 EB CLI 或 AWS CLI 來建立使用 Classic Load Balancer 設定的新環境。這些命令列工具會建立具有 CLB 的新環境，即使您的帳戶中沒有環境存在亦然。

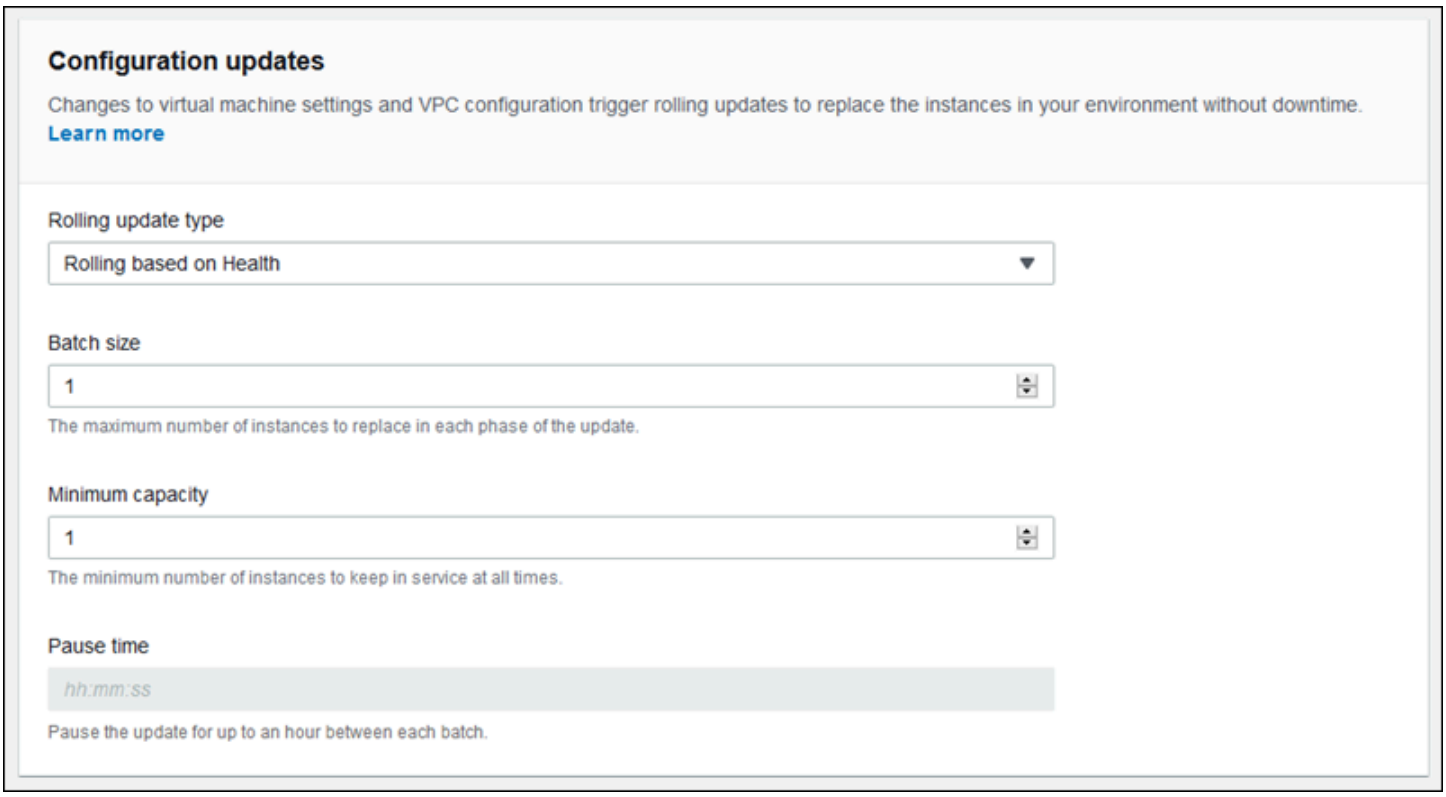
滾動更新和部署

您可以使用 Modify rolling updates and deployments (修改滾動更新和部署) 組態頁面來設定 Elastic Beanstalk 如何處理環境的應用程式部署和組態更新。

當您上傳更新的應用程式原始碼套件，並將其部署到您的環境時，就會發生應用程式部署。如需有關設定部署的詳細資訊，請參閱 the section called “部署選項”。



修改啟動組態或 VPC 設定的組態變更，需要終止環境中所有執行個體，並加以替換。如需設定更新類型和其他選項的詳細資訊，請參閱 the section called “組態變更”。



安全性

使用 Configure service access (設定服務存取) 頁面來設定服務和執行個體安全性設定。

如需 Elastic Beanstalk 安全性概念的說明，請參閱許可。

當您首次於 Elastic Beanstalk 主控台建立環境時，您必須建立具有一組預設許可的 EC2 執行個體設定檔。如果 EC2 執行個體設定檔下拉式清單未顯示任何可供選擇的值，請展開隨後的程序。它提供建立角色的步驟，接下來，您就可以為 EC2 執行個體設定檔選取角色。

為 EC2 執行個體設定檔建立 IAM 角色

若要建立 EC2 執行個體設定檔的 IAM 角色

1. 選擇檢視許可詳細資料。這會顯示在 EC2 執行個體設定檔下拉式清單下。

   畫面上會顯示標題為檢視執行個體設定檔許可的模態視窗。此視窗會列出您需要附加至您建立的新 EC2 執行個體設定檔的受管設定檔。它也提供啟動 IAM 主控台的連結。
2. 選擇顯示於視窗頂端的 IAM 主控台連結。
3. 在 IAM 主控台導覽窗格中，選擇 Roles (角色)。
4. 選擇 建立角色。

5. 在受信任的實體類型下，選擇 AWS 服務。
6. 在 Use case (使用案例) 下，選擇 EC2。
7. 選擇 Next (下一步)。
8. 附加合適的受管政策。在檢視執行個體設定檔許可模態視窗中捲動，查看受管政策。這些政策也會列在此處：

   - `AWSElasticBeanstalkWebTier`
   - `AWSElasticBeanstalkWorkerTier`
   - `AWSElasticBeanstalkMulticontainerDocker`
9. 選擇 Next (下一步)。
10. 輸入角色的名稱。
11. (選用) 附加標籤至角色。
12. 選擇 建立角色。
13. 返回開啟的 Elastic Beanstalk 主控台視窗。
14. 關閉檢視執行個體設定檔許可模態視窗

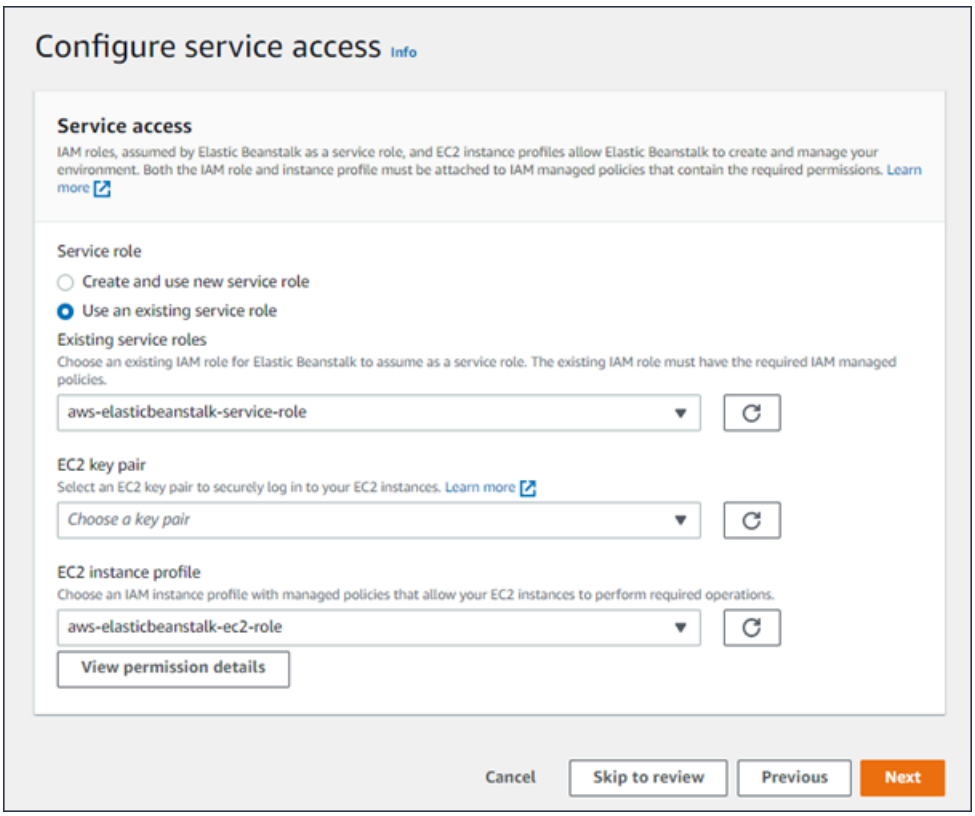
    > **Important**
    >
    > 請勿關閉顯示 Elastic Beanstalk 主控台的瀏覽器頁面。

15. 選擇 EC2 執行個體設定檔下拉式清單旁的

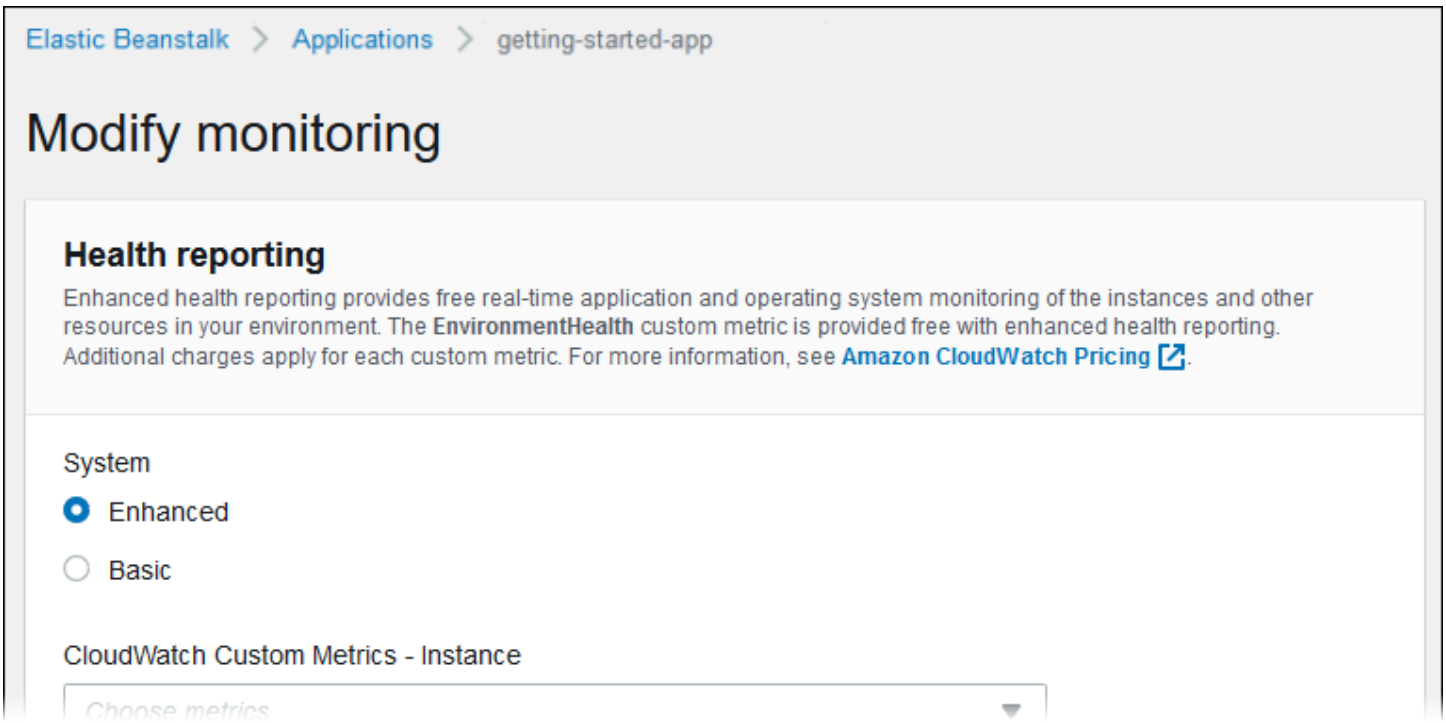
    (重新整理)。

    這會重新整理下拉式清單，讓您剛建立的「角色」會顯示在下拉式清單中。



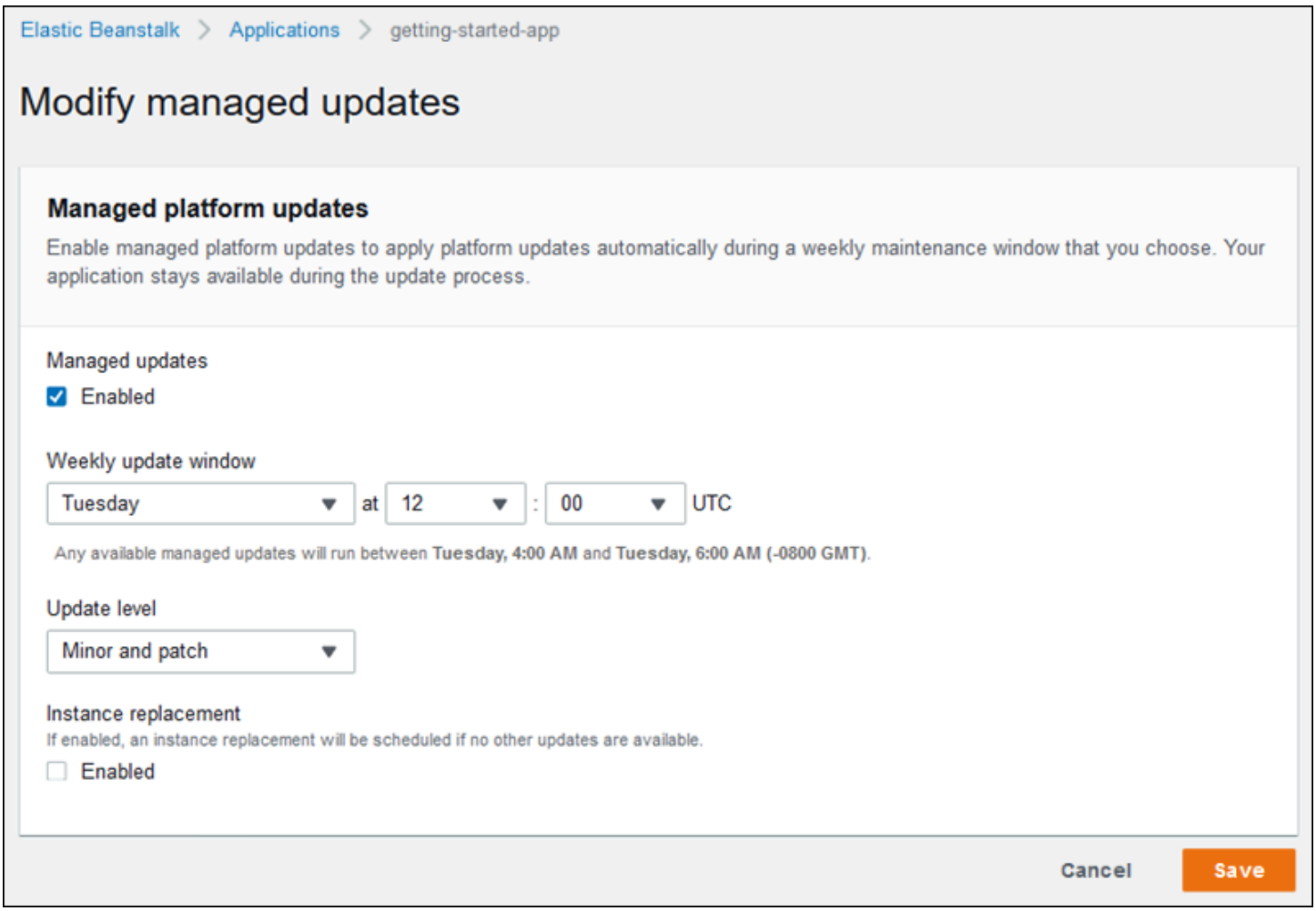
監控

您可以使用 Modify monitoring (修改監控) 組態頁面來設定運作狀態報告、監控規則和運作狀態事件串流。如需詳細資訊，請參閱 the section called “啟用增強型報告”、the section called “增強型運作狀態規則” 和 the section called “串流環境運作狀態”。



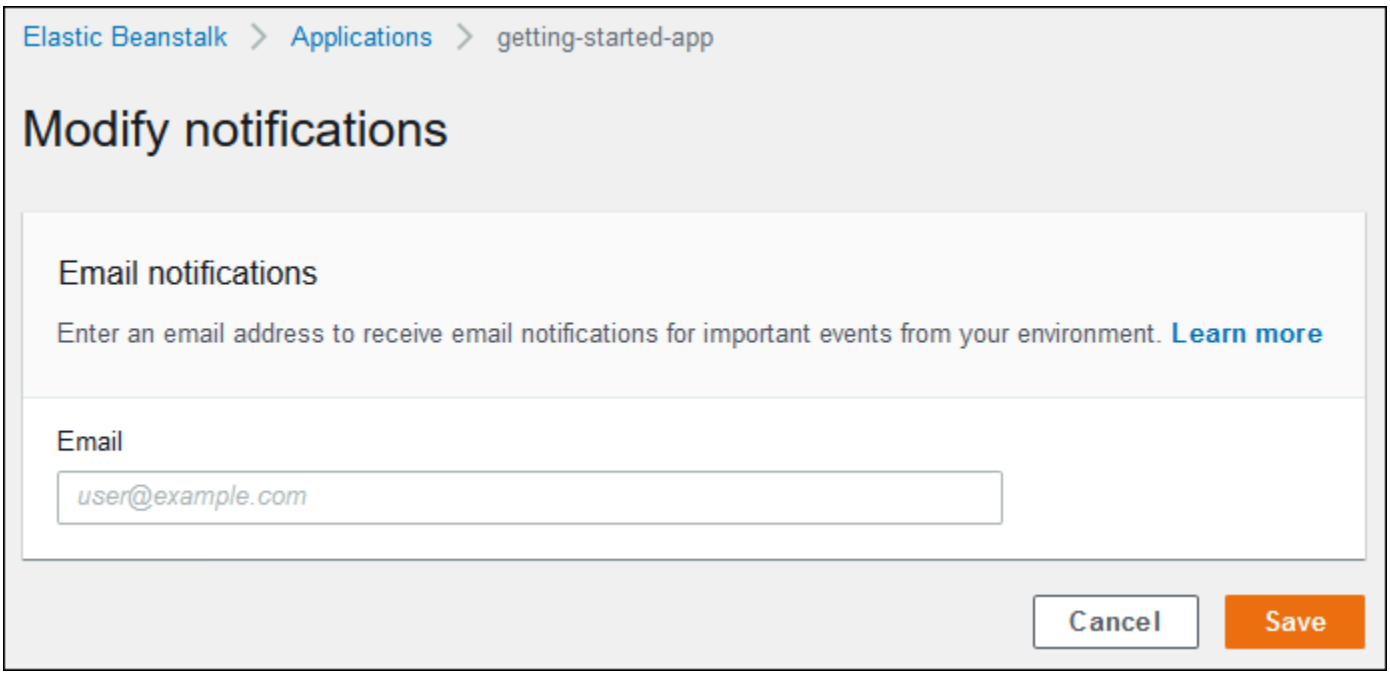
受管更新

使用 Modify managed updates (修改受管的更新) 組態頁面來設定受管的平台更新。您可以決定是否要啟用它們、設定排程以及設定其他屬性。如需詳細資訊，請參閱the section called “受管更新”。



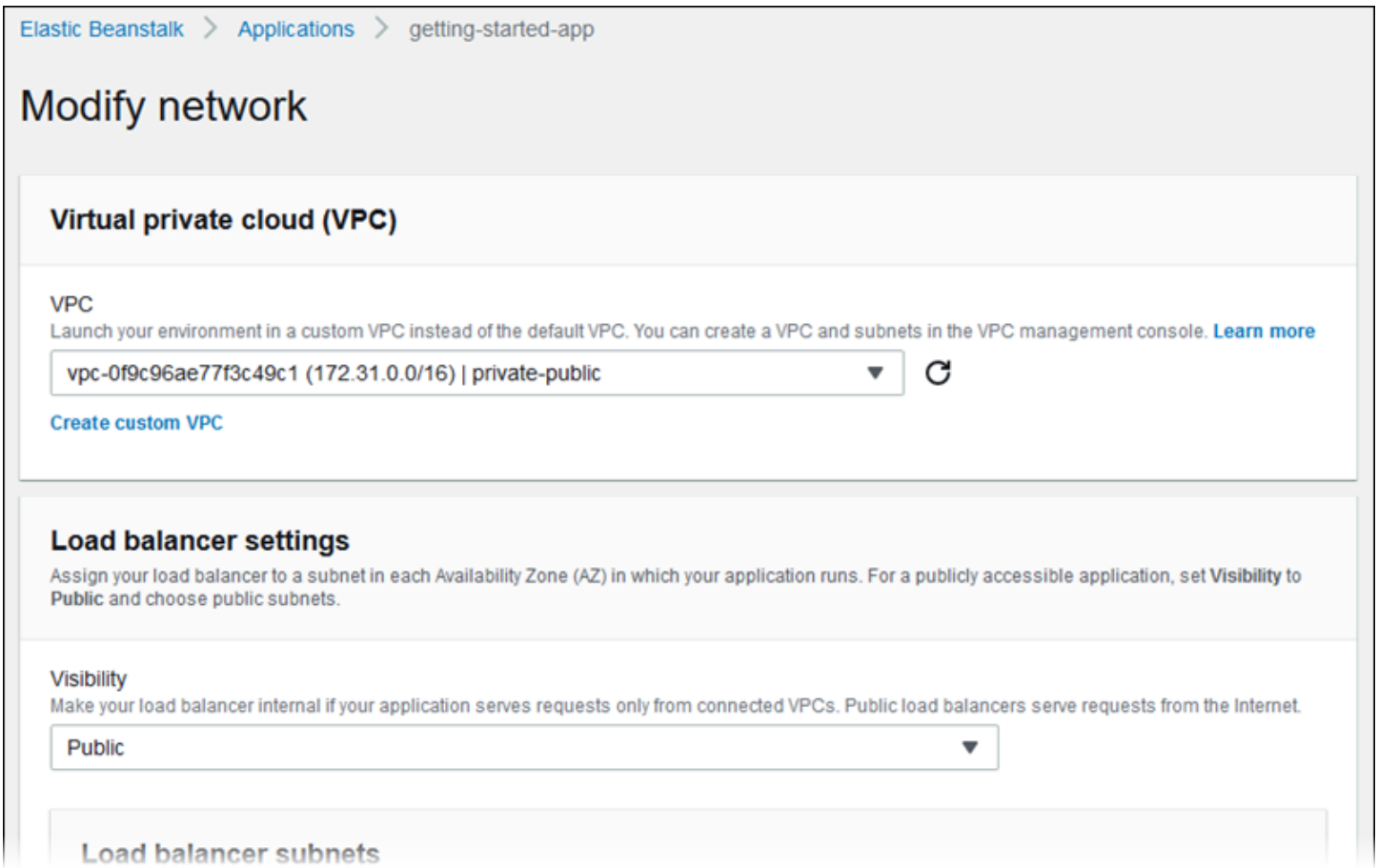
通知

使用 Modify notifications (修改通知) 組態頁面指定電子郵件地址，以接收環境中重要事件的電子郵件通知。



網路

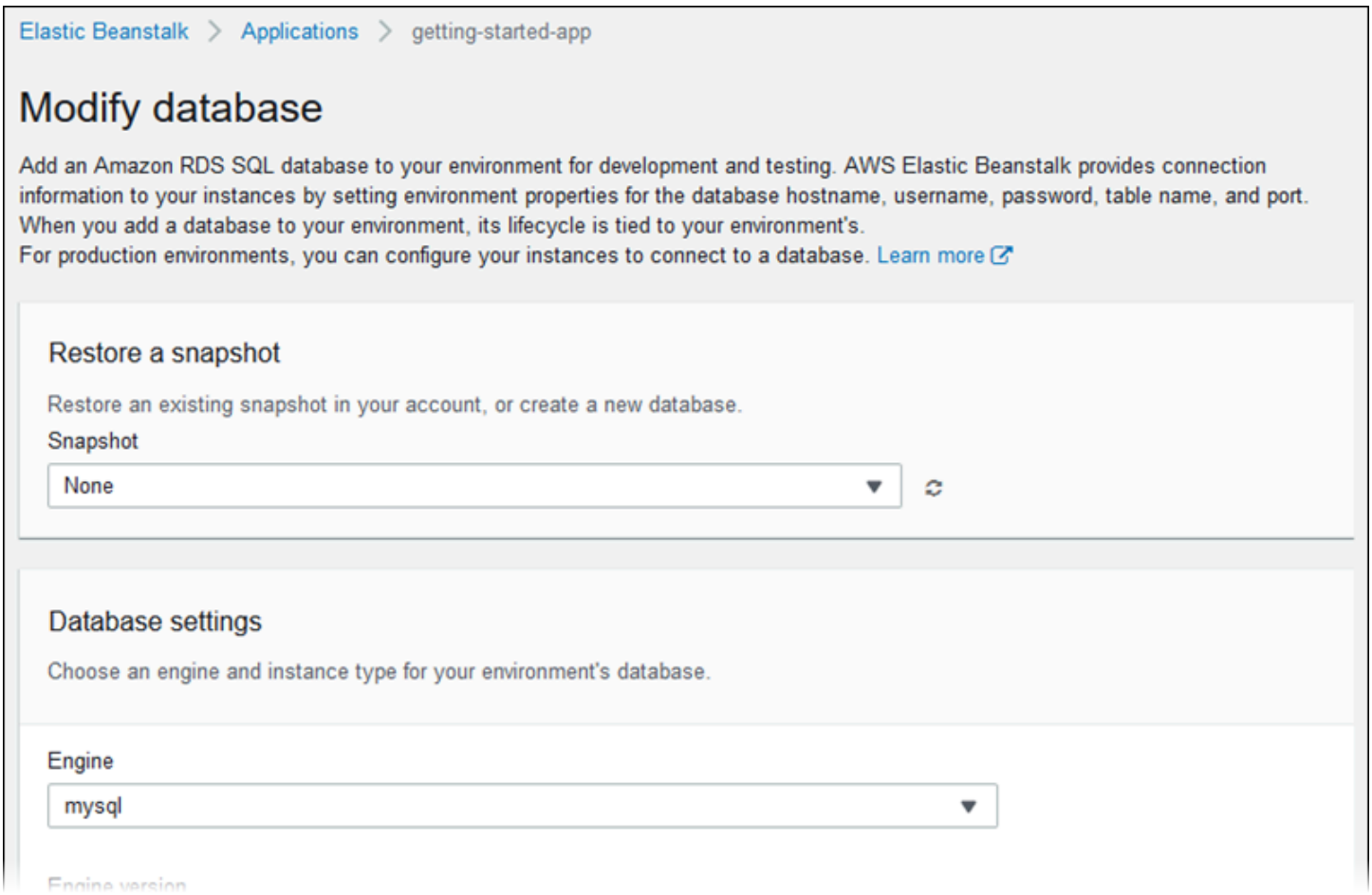
如果您已建立自訂 VPC，則 Modify network (修改網路) 組態頁面會設定您的環境以使用它。若您未選擇 VPC，Elastic Beanstalk 會使用預設的 VPC 和子網路。



資料庫

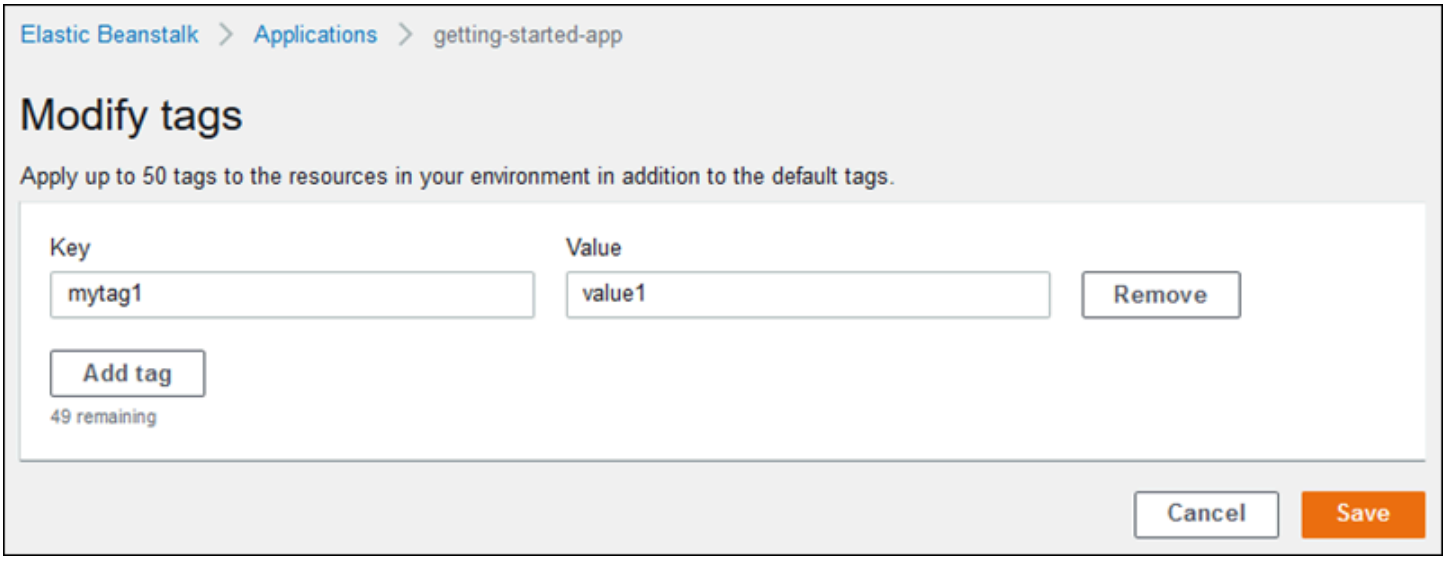
使用 Modify database (修改資料庫) 組態頁面，可將 Amazon Relational Database Service (Amazon RDS) 資料庫新增至您的環境，以進行開發和測試。Elastic Beanstalk 會設定資料庫主機名稱、使用者名稱、密碼、資料表名稱和連接埠的環境屬性，藉此將連線資訊提供給您的執行個體。

如需詳細資訊，請參閱the section called “資料庫”。



Tags (標籤)

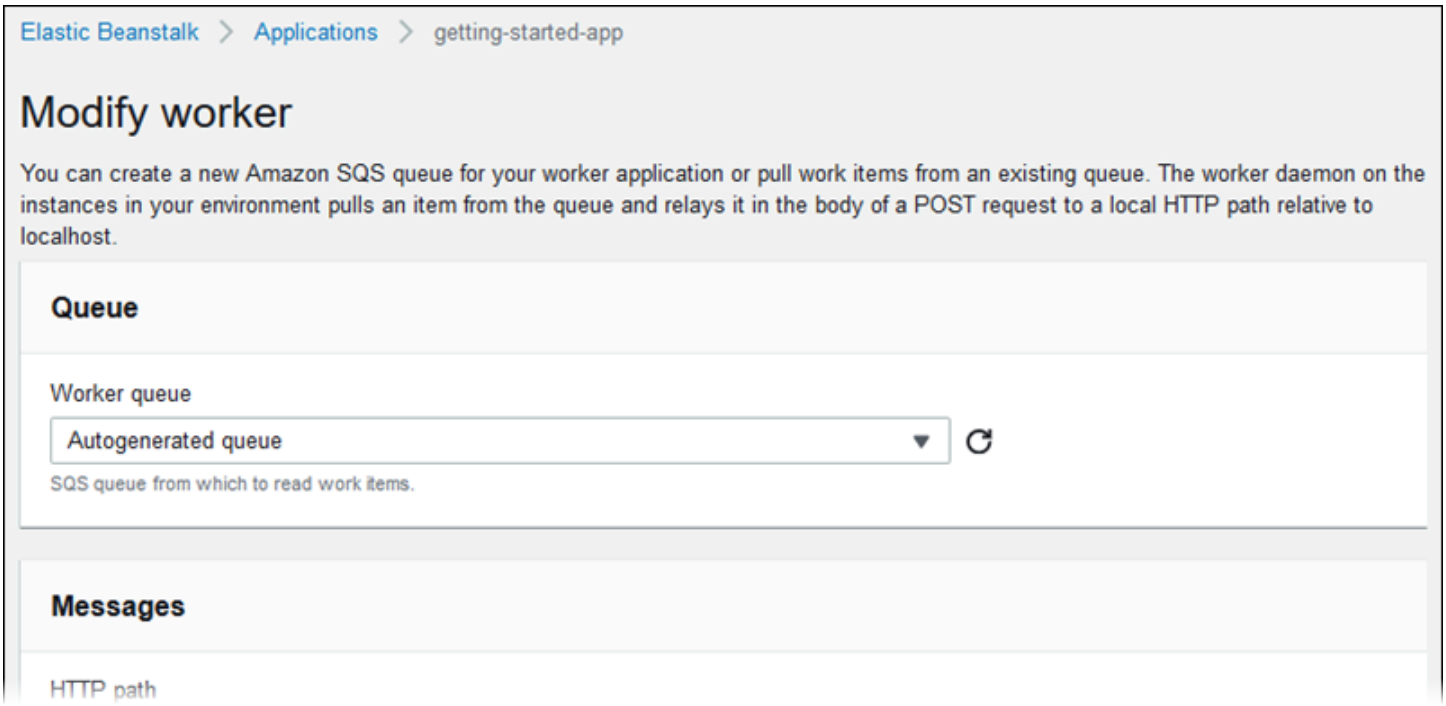
使用 Modify tags (修改標籤) 組態頁面，以新增標籤至您環境中的資源。如需環境標記的詳細資訊，請參閱在您的 Elastic Beanstalk 環境中標記資源。



工作者環境

如果您要建立工作者層環境，請使用 Modify worker (修改工作者) 組態頁面來設定工作者環境。您環境中執行個體上的工作者協助程式會從 Amazon Simple Queue Service (Amazon SQS) 佇列中提取項目，並將它們作為 POST 訊息轉傳到您的工作者應用程式。您可以選擇工作者協助程式讀取的 Amazon SQS 佇列 (自動產生或現有)。您也可以設定工作者協助程式傳送至應用程式的訊息。

如需更多詳細資訊，請參閱 the section called “工作者環境”。



複製 Elastic Beanstalk 環境

您可以透過複製現有環境，使用現有 Elastic Beanstalk 環境做為新環境的基礎。例如，您可能會想要建立複製，以便使用原始環境平台所用的較新版本平台分支。Elastic Beanstalk 會使用原始環境使用的環境設定來設定複製品。藉由複製現有環境而非建立新環境，您不需要手動設定選項設定、環境變數和您使用 Elastic Beanstalk 服務所做的其他設定。Elastic Beanstalk 還創建與原始環境關聯的任何 AWS 資源的副本。

重要的是要注意以下情況：

- 在複製過程中，Elastic Beanstalk 不會將資料從 Amazon RDS 複製到複製。
- Elastic Beanstalk 不會將未受管的資源變更納入複製環境。您使用 Elastic Beanstalk 主控台、命令列工具或 API 以外的工具變更 AWS 資源，均視為未受管的變更。

- 輸入的安全性群組會被視為未受管理的變更。複製的 Elastic Beanstalk 環境不會攜帶安全群組進入，因此環境對所有網際網路流量都開放。您必須為複製的環境重新建立輸入安全性群組。

您只能將環境複製到同一個平台分支的不同平台版本。不同的平台分支不保證相容。若要使用不同的平台分支，您必須手動建立新環境、部署應用程式程式碼，以及在程式碼和選項中進行任何必要的變更，以確保應用程式能夠在新平台分支上正確運作。

AWS 管理主控台

> **Important**
>
> 複製的 Elastic Beanstalk 環境不會攜帶安全群組進入，因此環境對所有網際網路流量都開放。您必須為複製的環境重新建立輸入安全性群組。
> 您可以檢查環境組態的漂移狀態，查看可能無法複製的資源。有關更多信息，請參閱AWS CloudFormation 用戶指南中的檢測整個 CloudFormation 堆棧上的漂移。

欲複製環境

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

   > **Note**
   >
   > 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在「環境概觀」頁面上，選擇動作。
4. 選擇複製環境。
5. 在 Clone environment (複製環境) 頁面，檢閱 Original Environment (原始環境) 區段的資訊，確認您已選擇欲建立複製的環境。
6. 在 New Environment (新環境) 區段，您可自由選擇是否變更 Elastic Beanstalk 根據原始環境自動設定的 Environment name (環境名稱)、Environment URL (環境 URL)、Description (描述)、Platform version (平台版本) 和 Service role (服務角色) 等值。

> **Note**
>
> 如果原始環境中使用的平台版本不是建議用於平台分支的版本，系統會警告您建議使用不同的平台版本。選擇 Platform version (平台版本)，您就可以在清單上看到建議的平台版本，例如 3.3.2 (Recommended) (3.3.2 (建議使用))。

7. 當您就緒後，選擇 Clone (複製)。

Elastic Beanstalk 命令列界面 (EB CLI)

> **Important**
>
> 複製的 Elastic Beanstalk 環境不會攜帶安全群組進入，因此環境對所有網際網路流量都開放。您必須為複製的環境重新建立輸入安全性群組。
> 您可以檢查環境組態的漂移狀態，查看可能無法複製的資源。有關更多信息，請參閱AWS CloudFormation 用戶指南中的檢測整個 CloudFormation 堆棧上的漂移。

使用 eb clone 命令來複製執行環境，如下所示。

```
~/workspace/my-app$ eb clone my-env1
Enter name for Environment Clone
(default is my-env1-clone): my-env2
Enter DNS CNAME prefix
(default is my-env1-clone): my-env2
```

您可於複製命令指定來源環境的名稱，或不指定以複製目前專案資料夾的預設環境。EB CLI 會提示您輸入新環境的名稱和 DNS 前綴字。

根據預設，eb clone 建立的新環境會使用來源環境平台的最新可用版本。即使有更新可用版本仍欲強制 EB CLI 使用相同版本，請使用 `--exact` 選項。

```
~/workspace/my-app$ eb clone --exact
```

如需此命令的詳細資訊，請參閱 eb clone。

終止 Elastic Beanstalk 環境

您可以使用 Elastic Beanstalk 主控台來終止執行中的 AWS Elastic Beanstalk 環境。透過執行這項操作，則可以避免未使用的 AWS 資源產生費用。

> **Note**
>
> 您稍後可隨時運用相同版本啟動新的環境。

如果您希望保留環境中的資料，請在終止環境之前，將資料庫刪除政策設定為 `Retain`。這可使資料庫在 Elastic Beanstalk 之外運作。在這項操作之後，任何 Elastic Beanstalk 環境都必須以外部資料庫的形式連接它。如果您想要在不保持資料庫運作的情況下備份資料，請將刪除政策設定為在終止環境之前建立資料庫的快照。如需詳細資訊，請參閱本指南的「設定環境」一章中的 資料庫生命週期。

Elastic Beanstalk 可能無法終止您的環境。一個常見的原因是，另一個環境的安全群組對您想要終止的環境安全群組具有相依性。如需如何避免此問題的說明，請參閱本指南的「EC2 執行個體」頁面的 安全群組。

> **Important**
>
> 如果您終止環境，您也必須刪除您建立的任何 CNAME 映射，因為其他客戶可能重複使用可用的主機名稱。請務必刪除指向終止環境的 DNS 記錄，以防止懸置 DNS 項目。懸置的 DNS 項目可能會降低您網域的網際網路流量的安全性，使其暴露在易於攻擊的弱點中。另外，它還可能存在其他風險。
>
> 如需詳細資訊，請參閱 Amazon Route 53 開發人員指南中的在 Route 53 上防止懸置委派記錄。您也可以在AWS安全部落格中針對適用於 Amazon CloudFront 請求的強化網域保護來進一步了解懸置 DNS 項目的資訊。

Elastic Beanstalk 主控台

終止環境

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 選擇 Actions (動作)，然後選擇 Terminate Environment (終止環境)。
4. 使用畫面顯示對話方塊來確認環境終止。

> **Note**
>
> 當您終止環境後，將釋出與該終止環境相關聯的 CNAME 供任何人使用。

Elastic Beanstalk 需要幾分鐘來終止在環境中執行的 AWS 資源。

AWS CLI

終止環境

- 執行下列命令。

```
$ aws elasticbeanstalk terminate-environment --environment-name my-env
```

API

終止環境

- 使用下列參數呼叫 `TerminateEnvironment`：

  `EnvironmentName = SampleAppEnv`

```
https://elasticbeanstalk.us-west-2.amazon.com/?EnvironmentName=SampleAppEnv
&Operation=TerminateEnvironment
&AuthParams
```

使用 AWS CLI 建立 Elastic Beanstalk 環境

如需有關 Elastic Beanstalk AWS CLI 指令的詳細資訊，請參閱AWS CLI 命令參考。

1. 檢查環境的 CNAME 是否可用。

```
$ aws elasticbeanstalk check-dns-availability --cname-prefix my-cname
{
    "Available": true,
    "FullyQualifiedCNAME": "my-cname.elasticbeanstalk.com"
}
```

2. 確定您的應用程式版本存在。

```
$ aws elasticbeanstalk describe-application-versions --application-name my-app --
version-label v1
```

   如果您還沒有適用於您來源的應用程式版本，請建立一個。例如，以下命令會從 Amazon Simple Storage Service (Amazon S3) 中的來源套件建立應用程式版本。

```
$ aws elasticbeanstalk create-application-version --application-name my-app --
version-label v1 --source-bundle S3Bucket=DOC-EXAMPLE-BUCKET,S3Key=my-source-
bundle.zip
```

3. 為應用程式建立組態範本。

```
$ aws elasticbeanstalk create-configuration-template --application-name my-app --
template-name v1 --solution-stack-name "64bit Amazon Linux 2015.03 v2.0.0 running
 Ruby 2.2 (Passenger Standalone)"
```

4. 建立環境。

```
$ aws elasticbeanstalk create-environment --cname-prefix my-cname --application-
name my-app --template-name v1 --version-label v1 --environment-name v1clone --
option-settings file://options.txt
```

   在 options.txt 檔案中定義了選項設定：

```
[
    {
        "Namespace": "aws:autoscaling:launchconfiguration",
```

```
        "OptionName": "IamInstanceProfile",
        "Value": "aws-elasticbeanstalk-ec2-role"
    }
]
```

上述的選項設定定義了 IAM 執行個體描述檔。您可以指定 ARN 或設定檔名稱。

5. 判斷新的環境是否是已就緒可用。

```
$ aws elasticbeanstalk describe-environments --environment-names my-env
```

如果新環境啟動時並未就緒可用，則您應決定是否要嘗試操作，或是讓環境維持目前的狀態以進行調查。務必在完成後終止環境，並清除任何未使用的資源。

> **Note**
>
> 如果環境未在合理的時間啟動，您可以調整逾時的時間長度。

使用 API 建立 Elastic Beanstalk 環境

1. 使用下列參數呼叫 `CheckDNSAvailability`：

   - `CNAMEPrefix` = `SampleApp`

   Example

   ```
   https://elasticbeanstalk.us-east-2.amazonaws.com/?CNAMEPrefix=sampleapplication
   &Operation=CheckDNSAvailability
   &AuthParams
   ```

2. 使用下列參數來呼叫 `DescribeApplicationVersions`：

   - `ApplicationName` = `SampleApp`
   - `VersionLabel` = `Version2`

   Example

   ```
   https://elasticbeanstalk.us-east-2.amazonaws.com/?ApplicationName=SampleApp
   ```

```
&VersionLabel=Version2
&Operation=DescribeApplicationVersions
&AuthParams
```

3. 使用下列參數來呼叫 CreateConfigurationTemplate：

   - ApplicationName = SampleApp
   - TemplateName = MyConfigTemplate
   - SolutionStackName = 64bit%20Amazon%20Linux%202015.03%20v2.0.0%20running%20Ruby%202.2%20(Passenger%20Standalone)

   Example

   ```
   https://elasticbeanstalk.us-east-2.amazonaws.com/?ApplicationName=SampleApp
   &TemplateName=MyConfigTemplate
   &Operation=CreateConfigurationTemplate
   &SolutionStackName=64bit%20Amazon%20Linux%202015.03%20v2.0.0%20running%20Ruby
   %202.2%20(Passenger%20Standalone)
   &AuthParams
   ```

4. 使用下列其中一組參數來呼叫 CreateEnvironment。

   a. Web 伺服器環境層請使用下列參數：

      - EnvironmentName = SampleAppEnv2
      - VersionLabel = Version2
      - Description = description
      - TemplateName = MyConfigTemplate
      - ApplicationName = SampleApp
      - CNAMEPrefix = sampleapplication
      - OptionSettings.member.1.Namespace = aws:autoscaling:launchconfiguration
      - OptionSettings.member.1.OptionName = IamInstanceProfile
      - OptionSettings.member.1.Value = aws-elasticbeanstalk-ec2-role

Example

```
https://elasticbeanstalk.us-east-2.amazonaws.com/?ApplicationName=SampleApp
&VersionLabel=Version2
&EnvironmentName=SampleAppEnv2
&TemplateName=MyConfigTemplate
&CNAMEPrefix=sampleapplication
&Description=description
&Operation=CreateEnvironment
&OptionSettings.member.1.Namespace=aws%3Aautoscaling%3Alaunchconfiguration
&OptionSettings.member.1.OptionName=IamInstanceProfile
&OptionSettings.member.1.Value=aws-elasticbeanstalk-ec2-role
&AuthParams
```

b. 工作者環境層請使用下列參數：

- EnvironmentName = SampleAppEnv2
- VersionLabel = Version2
- Description = description
- TemplateName = MyConfigTemplate
- ApplicationName = SampleApp
- Tier = Worker
- OptionSettings.member.1.Namespace = aws:autoscaling:launchconfiguration
- OptionSettings.member.1.OptionName = IamInstanceProfile
- OptionSettings.member.1.Value = aws-elasticbeanstalk-ec2-role
- OptionSettings.member.2.Namespace = aws:elasticbeanstalk:sqsd
- OptionSettings.member.2.OptionName = WorkerQueueURL
- OptionSettings.member.2.Value = sqsd.elasticbeanstalk.us-east-2.amazonaws.com
- OptionSettings.member.3.Namespace = aws:elasticbeanstalk:sqsd
- OptionSettings.member.3.OptionName = HttpPath
- OptionSettings.member.3.Value = /
- OptionSettings.member.4.Namespace = aws:elasticbeanstalk:sqsd

- OptionSettings.member.4.OptionName = MimeType
- OptionSettings.member.4.Value = application/json
- OptionSettings.member.5.Namespace = aws:elasticbeanstalk:sqsd
- OptionSettings.member.5.OptionName = HttpConnections
- OptionSettings.member.5.Value = 75
- OptionSettings.member.6.Namespace = aws:elasticbeanstalk:sqsd
- OptionSettings.member.6.OptionName = ConnectTimeout
- OptionSettings.member.6.Value = 10
- OptionSettings.member.7.Namespace = aws:elasticbeanstalk:sqsd
- OptionSettings.member.7.OptionName = InactivityTimeout
- OptionSettings.member.7.Value = 10
- OptionSettings.member.8.Namespace = aws:elasticbeanstalk:sqsd
- OptionSettings.member.8.OptionName = VisibilityTimeout
- OptionSettings.member.8.Value = 60
- OptionSettings.member.9.Namespace = aws:elasticbeanstalk:sqsd
- OptionSettings.member.9.OptionName = RetentionPeriod
- OptionSettings.member.9.Value = 345600

Example

```
https://elasticbeanstalk.us-east-2.amazonaws.com/?ApplicationName=SampleApp
&VersionLabel=Version2
&EnvironmentName=SampleAppEnv2
&TemplateName=MyConfigTemplate
&Description=description
&Tier=Worker
&Operation=CreateEnvironment
&OptionSettings.member.1.Namespace=aws%3Aautoscaling%3Alaunchconfiguration
&OptionSettings.member.1.OptionName=IamInstanceProfile
&OptionSettings.member.1.Value=aws-elasticbeanstalk-ec2-role
&OptionSettings.member.2.Namespace=aws%3Aelasticbeanstalk%3Asqsd
&OptionSettings.member.2.OptionName=WorkerQueueURL
&OptionSettings.member.2.Value=sqsd.elasticbeanstalk.us-east-2.amazonaws.com
&OptionSettings.member.3.Namespace=aws%3elasticbeanstalk%3sqsd
&OptionSettings.member.3.OptionName=HttpPath
```

```
&OptionSettings.member.3.Value=%2F
&OptionSettings.member.4.Namespace=aws%3Aelasticbeanstalk%3Asqsd
&OptionSettings.member.4.OptionName=MimeType
&OptionSettings.member.4.Value=application%2Fjson
&OptionSettings.member.5.Namespace=aws%3Aelasticbeanstalk%3Asqsd
&OptionSettings.member.5.OptionName=HttpConnections
&OptionSettings.member.5.Value=75
&OptionSettings.member.6.Namespace=aws%3Aelasticbeanstalk%3Asqsd
&OptionSettings.member.6.OptionName=ConnectTimeout
&OptionSettings.member.6.Value=10
&OptionSettings.member.7.Namespace=aws%3Aelasticbeanstalk%3Asqsd
&OptionSettings.member.7.OptionName=InactivityTimeout
&OptionSettings.member.7.Value=10
&OptionSettings.member.8.Namespace=aws%3Aelasticbeanstalk%3Asqsd
&OptionSettings.member.8.OptionName=VisibilityTimeout
&OptionSettings.member.8.Value=60
&OptionSettings.member.9.Namespace=aws%3Aelasticbeanstalk%3Asqsd
&OptionSettings.member.9.OptionName=RetentionPeriod
&OptionSettings.member.9.Value=345600
&AuthParams
```

建構立即啟動 URL

您可建構自訂 URL，讓任何人都能夠在 AWS Elastic Beanstalk快速部署並執行預先決定的 Web 應用程式。此 URL 稱為立即啟動 URL。例如，為了示範專門於 Elastic Beanstalk 執行的 Web 應用程式，您可能就需要立即啟動 URL。透過立即啟動 URL，您可以使用參數，將所需資訊事先新增至建立應用程式精靈。在您將此資訊新增至精靈之後，任何人都只需要幾個步驟即可使用該 URL 連結，透過您的 Web 應用程式原始碼啟動 Elastic Beanstalk 環境。這表示使用者無須手動上傳或指定應用程式原始碼套件的位置。他們也不需要向精靈提供任何其他資訊。

立即啟動 URL 向 Elastic Beanstalk 提供建立應用程式所需要的最少資訊：應用程式名稱、解決方案堆疊、執行個體類型以及環境類型。對於在您的自訂立即啟動 URL 中未明確指定的組態詳細資訊，Elastic Beanstalk 會使用預設值。

立即啟動 URL 使用標準 URL 語法。如需詳細資訊，請參閱 RFC 3986 - Uniform Resource Identifier (URI): Generic Syntax。

URL 參數

URL 必須包含下列區分大小寫的參數：

- 區域 — 指定 AWS 區域。如需 Elastic Beanstalk 支援的區域清單，請參閱《AWS 一般參考》中的 AWS Elastic Beanstalk 端點與配額。
- applicationName – 指定您應用程式的名稱。Elastic Beanstalk 會顯示應用程式在 Elastic Beanstalk 主控台內的名稱，以區別其他應用程式。根據預設，應用程式名稱亦是環境名稱及環境 URL 的組成基礎。
- platform (平台) – 指定要針對環境使用的平台版本。使用以下其中一個方法，然後以 URL 編碼您的選擇：
  - 指定不具版本的平台 ARN。Elastic Beanstalk 會選取對應平台主要版本的最新平台版本。例如，若要選取最新的 Python 3.6 平台版本，請指定 `Python 3.6 running on 64bit Amazon Linux`。
  - 指定平台名稱。Elastic Beanstalk 會選取平台最新語言執行時間 (例如，Python) 的最新版本。

  如需所有可用平台及其版本的說明，請參閱支援 Elastic Beanstalk 的平台。

  您可以使用 AWS Command Line Interface (AWS CLI) 取得所有可用平台版本清單及其個別的 ARN。`list-platform-versions` 命令會列出所有可用平台版本的詳細資訊。使用 `--filters` 引數可縮小清單的範圍。例如，您可以將清單限制為僅顯示特定語言的平台版本。

  以下範例會查詢所有 Python 平台版本，並透過一系列的命令傳送輸出。其結果是一份平台版本 ARN 的清單 (不含 */version* 結尾)，以可供人員閱讀的格式呈現，且不會進行 URL 編碼。

```
$ aws elasticbeanstalk list-platform-versions --filters
 'Type="PlatformName",Operator="contains",Values="Python"' | grep PlatformArn | awk -
F '"' '{print $4}' | awk -F '/' '{print $2}'
Preconfigured Docker - Python 3.4 running on 64bit Debian
Preconfigured Docker - Python 3.4 running on 64bit Debian
Python 2.6 running on 32bit Amazon Linux
Python 2.6 running on 32bit Amazon Linux 2014.03
...
Python 3.6 running on 64bit Amazon Linux
```

  以下範例將 Perl 命令加入到最後一個範例，以便對輸出進行 URL 編碼。

```
$ aws elasticbeanstalk list-platform-versions --filters
 'Type="PlatformName",Operator="contains",Values="Python"' | grep PlatformArn | awk
 -F '"' '{print $4}' | awk -F '/' '{print $2}' | perl -MURI::Escape -ne 'chomp;print
 uri_escape($_),"\n"'
Preconfigured%20Docker%20-%20Python%203.4%20running%20on%2064bit%20Debian
Preconfigured%20Docker%20-%20Python%203.4%20running%20on%2064bit%20Debian
```

```
Python%202.6%20running%20on%2032bit%20Amazon%20Linux
Python%202.6%20running%20on%2032bit%20Amazon%20Linux%202014.03
...
Python%203.6%20running%20on%2064bit%20Amazon%20Linux
```

立即啟動 URL 可選擇性納入下列參數。若您的立即啟動 URL 不含選用參數，Elastic Beanstalk 會使用預設值來建立並執行您的應用程式。如果您不包含sourceBundleUrl參數，Elastic Beanstalk 會使用指定平台的預設範例應用程式。

- sourceBundleUrl— 以 URL 格式指定 Web 應用程式來源套裝軟體的位置。例如，如果您將來源服務包上傳到 Amazon S3 儲存貯體，則可以將sourceBundleUrl參數的值指定為`https://mybucket.s3.amazonaws.com/myobject`。

  > **Note**
  >
  > 您可以將sourceBundleUrl參數值指定為 HTTP URL，但是使用者的網頁瀏覽器會套用 HTML URL 編碼，視需要轉換字元。

- environmentType – 指定環境是否有負載平衡且可擴展，或為單一執行個體環境。如需更多詳細資訊，請參閱 環境類型。參數值可指定為 `LoadBalancing` 或 `SingleInstance`。
- tierName – 指定環境支援的 Web 應用程式，是否會處理 Web 請求或執行背景任務。如需更多詳細資訊，請參閱 Elastic Beanstalk 工作者環境。您可指定為 `WebServer` 或 `Worker`。
- instanceType – 指定具有最適合您應用程式之特性 (包含記憶體大小和 CPU 能力) 的伺服器。如需 Amazon EC2 執行個體系列和類型的詳細資訊，請參閱 Amazon EC2 使用者指南中的執行個體類型或 Amazon EC2 使用者指南中的執行個體類型。如需有關跨區域可用執行個體類型的詳細資訊，請參閱 Amazon EC2 使用者指南中的可用執行個體類型或 Amazon EC2 使用者指南中的可用執行個體類型。
- withVpc – 指定是否於 Amazon VPC 中建立環境。您可指定為 `true` 或 `false`。如需搭配 Amazon VPC 使用 Elastic Beanstalk 的詳細資訊，請參閱搭配 Amazon VPC 使用 Elastic Beanstalk。
- withRds – 指定是否透過此環境建立 Amazon RDS 資料庫執行個體。如需更多詳細資訊，請參閱 搭配 Amazon RDS 使用 Elastic Beanstalk。您可指定為 `true` 或 `false`。
- rdsDBEngine – 指定環境中 Amazon EC2 執行個體欲使用的資料庫引擎。您可指定為 `mysql`、`oracle-sel`、`sqlserver-ex`、`sqlserver-web` 或 `sqlserver-se`。預設值為 `mysql`。
- RDSDB AllocatedStorage — 指定配置的資料庫儲存體大小 (GB)。您可以指定下列值：
  - MySQL - 5 至 1024。預設值為 5。

  - Oracle - 10 至 1024。預設值為 10。
  - Microsoft SQL Server Express 版本 - 30。
  - Microsoft SQL Server Web 版本 - 30。
  - Microsoft SQL Server Standard 版本 - 200。
- RDSDB InstanceClass — 指定資料庫執行處理類型。預設值為 db.t2.micro (非執行於 Amazon VPC 的環境則為 db.m1.large)。如需 Amazon RDS 支援的資料庫執行個體類別清單，請參閱《Amazon Relational Database Service 使用者指南》中的資料庫執行個體類別。
- rdsMultiAZDatabase - 指定 Elastic Beanstalk 是否需要跨多個可用區域建立資料庫執行個體。您可指定為 true 或 false。如需使用 Amazon RDS 進行多個可用區域部署的詳細資訊，請參閱《Amazon Relational Database Service 使用者指南》中的區域與可用區域。
- RDSDB DeletionPolicy — 指定是否要在環境終止時刪除或建立資料庫執行處理的快照。您可指定為 Delete 或 Snapshot。

範例

下列是立即啟動 URL 的範例。建構自己的 URL 後，即可提供給您的使用者。例如，您可以在網頁或培訓教材中內嵌 URL。當使用者透過立即啟動 URL 建立應用程式時，Elastic Beanstalk 的建立應用程式精靈不需要其他輸入。

```
https://console.aws.amazon.com/elasticbeanstalk/home?region=us-west-2#/
newApplication?applicationName=YourCompanySampleApp&platform=PHP%207.3%20running
%20on%2064bit%20Amazon%20Linux&sourceBundleUrl=http://s3.amazonaws.com/mybucket/
myobject&environmentType=SingleInstance&tierName=WebServer&instanceType=m1.small&withVpc=true&w
```

當使用者選擇立即啟動 URL，Elastic Beanstalk 會顯示如下頁面。

Create a web app

Create a new application and environment with a sample application or your own code. By creating an environment, you allow AWS Elastic Beanstalk to manage AWS resources and permissions on your behalf. Learn more

Application information

Application name YourCompanySampleApp

Up to 100 Unicode characters, not including forward slash (/).

Environment information

Choose the name, subdomain, and description for your environment. These cannot be changed later.

Environment name Yourcompanysampleapp-env

Domain Leave blank for autogenerated value .us-east-1.elasticbeanstalk.com

Check availability

Description

Base configuration

Tier Web Server (Choose tier)

Platform Preconfigured platform

Platforms published and maintained by AWS Elastic Beanstalk.

PHP

Custom platform NEW

Platforms created and owned by you. Learn more

-- Choose a custom platform --

Application code Sample application

Get started right away with sample code.

Upload your code

Upload a source bundle from your computer or copy one from Amazon S3.

Upload ZIP or WAR

Cancel Review and launch

欲使用立即啟動 URL

1. 選擇立即啟動 URL。
2. 在 Elastic Beanstalk 主控台開啟後，請於 Create a web app (建立 Web 應用程式) 頁面選擇 Review and launch (檢閱並發佈)，即可檢視 Elastic Beanstalk 用於建立應用程式並啟動應用程式執行之環境所使用的設定。
3. 在 Configure (設定) 頁面中選擇 Create app (建立應用程式) 以建立應用程式。

建立和更新 Elastic Beanstalk 環境的群組

透過 AWS Elastic Beanstalk `Compose Environments` API，您可以在單一應用程式中建立和更新 Elastic Beanstalk 環境群組。群組中的每個環境，皆可執行服務導向架構應用程式的不同元件。`Compose Environments` API 需要應用程式版本的清單和選用的群組名稱。Elastic Beanstalk 會針對每個應用程式版本建立環境，如果環境已存在，則會將應用程式版本部署到這些環境。

建立 Elastic Beanstalk 環境之間的連結，將一個環境指派為另一個環境的相依環境。當您使用 `Compose Environments` API 來建立一組環境時，Elastic Beanstalk 只會在其相依的環境開始正常運作之後，才會建立依賴的環境。如需關於環境連結的詳細資訊，請參閱在 Elastic Beanstalk 環境之間建立連結。

`Compose Environments` API 會使用環境資訊清單來儲存環境群組所共用的設定詳細資訊。每個元件應用程式的應用程式原始碼套件中，都必須具有 `env.yaml` 組態檔案，其中指定用來建立其環境的參數。

`Compose Environments` 需要在每個元件應用程式的環境資訊清單中，指定 `EnvironmentName` 和 `SolutionStack`。

您可以將 `Compose Environments` API 與 Elastic Beanstalk 命令列介面 (EB CLI) AWS CLI、或 SDK 搭配使用。如需 EB CLI 的說明，請參閱 利用 EB CLI，以群組方式管理多個 Elastic Beanstalk 環境。

使用 **`Compose Environments`** API

例如，您可以建立名為 `Media Library` 的應用程式，來讓使用者上傳和管理儲存於 Amazon Simple Storage Service (Amazon S3) 中的影像和影片。這個應用程式具備前端環境 `front`，此環境會執行 Web 應用程式，來讓使用者上傳和下載個別檔案、檢視其程式庫和起始批次處理工作。

前端應用程式會將任務加入 Amazon SQS 佇列，而非直接處理任務。第二個環境中，worker 從佇列提取並處理任務。worker 使用具有高效能 GPU 的 G2 執行個體類型，同時 front 可執行於更具有成本效益的一般執行個體類型。

您可以將專案資料夾 (Media Library) 針對每個元件劃分為不同的目錄，而每個目錄包含環境定義檔案 (env.yaml) 和每個元件的原始程式碼：

```
~/workspace/media-library
|-- front
|   `-- env.yaml
`-- worker
    `-- env.yaml
```

下列的清單列出了每個元件應用程式的 env.yaml。

~/workspace/media-library/front/env.yaml

```
EnvironmentName: front+
EnvironmentLinks:
  "WORKERQUEUE" : "worker+"
AWSConfigurationTemplateVersion: 1.1.0.0
EnvironmentTier:
  Name: WebServer
  Type: Standard
SolutionStack: 64bit Amazon Linux 2015.09 v2.0.4 running Java 8
OptionSettings:
  aws:autoscaling:launchconfiguration:
    InstanceType: m4.large
```

~/workspace/media-library/worker/env.yaml

```
EnvironmentName: worker+
AWSConfigurationTemplateVersion: 1.1.0.0
EnvironmentTier:
  Name: Worker
  Type: SQS/HTTP
SolutionStack: 64bit Amazon Linux 2015.09 v2.0.4 running Java 8
OptionSettings:
  aws:autoscaling:launchconfiguration:
    InstanceType: g2.2xlarge
```

在為前端 (front-v1) 與工作者 (worker-v1) 應用程式組件建立應用程式版本之後，您可以用該版本名稱來呼叫 Compose Environments API。在此範例中，我們 AWS CLI 使用呼叫 API。

```
# Create application versions for each component:
~$ aws elasticbeanstalk create-application-version --application-name media-
library --version-label front-v1 --process --source-bundle S3Bucket="DOC-EXAMPLE-
BUCKET",S3Key="front-v1.zip"
  {
    "ApplicationVersion": {
        "ApplicationName": "media-library",
        "VersionLabel": "front-v1",
        "Description": "",
        "DateCreated": "2015-11-03T23:01:25.412Z",
        "DateUpdated": "2015-11-03T23:01:25.412Z",
        "SourceBundle": {
            "S3Bucket": "DOC-EXAMPLE-BUCKET",
            "S3Key": "front-v1.zip"
        }
    }
  }
~$ aws elasticbeanstalk create-application-version --application-name media-
library --version-label worker-v1 --process --source-bundle S3Bucket="DOC-EXAMPLE-
BUCKET",S3Key="worker-v1.zip"
  {
    "ApplicationVersion": {
        "ApplicationName": "media-library",
        "VersionLabel": "worker-v1",
        "Description": "",
        "DateCreated": "2015-11-03T23:01:48.151Z",
        "DateUpdated": "2015-11-03T23:01:48.151Z",
        "SourceBundle": {
            "S3Bucket": "DOC-EXAMPLE-BUCKET",
            "S3Key": "worker-v1.zip"
        }
    }
  }
# Create environments:
~$ aws elasticbeanstalk compose-environments --application-name media-library --group-
name dev --version-labels front-v1 worker-v1
```

第三次呼叫會建立兩個環境：front-dev 與 worker-dev。API 會把 EnvironmentName 檔案中所指定的 env.yaml 和 group name 呼叫中所指定的 Compose Environments 選項串連在一起，中

間以連字符號分隔，來做為環境的名稱。這兩個選項加上連字號的總長度，不得超過環境名稱長度的 23 個字元上限。

在 `front-dev` 環境中執行的應用程式，可以藉由讀取 `worker-dev` 變數，來取得連結至 `WORKERQUEUE` 環境的 Amazon SQS 佇列的名稱。如需關於環境連結的詳細資訊，請參閱在 Elastic Beanstalk 環境之間建立連結。

將應用程式部署至 Elastic Beanstalk 環境

您可以使用 AWS Elastic Beanstalk 主控台來上傳已更新的原始碼套件，並將其部署至您的 Elastic Beanstalk 環境，或重新部署之前上傳的版本。

各個部署都能以部署 ID 識別。部署 ID 起始為 1，每次進行部署和變更執行個體組態時，累加 1。若啟用增強型運作狀態報告，Elastic Beanstalk 會在報告執行個體運作狀態時，在運作狀態主控台和 EB CLI 顯示部署 ID。部署 ID 可協助您於滾動更新失敗時，判斷環境的狀態。

Elastic Beanstalk 提供數個部署政策和設定。如需設定政策和其他設定的詳細資料，請參閱 the section called “部署選項”。下表列出支援這些項目的政策和環境類型。

支援的部署政策

部署政策	負載平衡環境	單一執行個體環境	舊式 Windows Server 環境†
一次全部	✓ 是	✓ 是	✓ 是
滾動	✓ 是	X 否	✓ 是
以額外批次進行滾動	✓ 是	X 否	X 否
固定	✓ 是	✓ 是	X 否
流量分割	✓ 是 (Application Load Balancer)	X 否	X 否

† 在此表格中，「舊式 Windows Server 環境」是一種使用 IIS 8.5 以前版本的 Windows Server 平台組態為基礎的環境。

> **Warning**
>
> 部署或更新期間，部分原則會取代所有執行個體。這會造成所有的累計 Amazon EC2 爆量餘額遺失。這發生的情況如下：
>
> - 執行個體更換啟用的受管平台更新
> - 不可變更新
> - 不可變更新或流量分割啟用的部署

選擇部署政策

為您的應用程式選擇正確的部署政策是數個考量事項的取捨過程，並取決於您的特定需求。the section called “部署選項” 頁面包含每個政策的詳細資訊，以及其中一些政策運作方式的詳細說明。

以下清單提供不同部署政策的摘要資訊，並新增相關的考量事項。

- All at once (一次全部) - 最快速的部署方法。如果您可以接受短暫的服務中斷，且快速部署對您來說相當重要，便適合使用此部署政策。使用此方法，Elastic Beanstalk 會將新的應用程式版本部署到每個執行個體。然後，Web 代理或應用程式伺服器可能需要重新啟動。因此，您的應用程式可能會在短時間內無法供使用者使用 (或是可用性降低)。
- Rolling (滾動) - 避免停機時間，並藉由拉長部署時間，將降低可用性的影響降至最低。如果您無法接受任何期間的完全服務中斷，便適合使用此部署政策。使用此方法，您的應用程式會一次部署一個批次的執行個體到您的環境。在整個部署期間您可以保留大部分的頻寬。
- Rolling with additional batch (以額外批次進行滾動) - 相較於「Rolling」(滾動) 方法，將部署時間拉得更長，避免降低任何可用性。如果您必須在整個部署期間維持相同的頻寬，便適合使用此部署政策。使用此方法，Elastic Beanstalk 會啟動額外的執行個體批次，然後執行滾動部署。啟動額外的批次需要時間，並會確保在整個部署過程中保留相同的頻寬。
- Immutable (不可變) - 較慢的部署方法，可確保您的新應用程式版本一律會部署到新的執行個體，而非更新現有的執行個體。在部署失敗時，其還具有快速且可安全轉返的額外優點。使用此方法，Elastic Beanstalk 會執行不可變的更新來部署您的應用程式。在不可變的更新作業中，您的環境會啟動第二個 Auto Scaling 群組，新舊版本此時同時為流量提供服務，直到新的執行個體通過運作狀態檢查。
- Traffic splitting (流量分割) - Canary 測試部署方法。若您希望使用一部分的傳入流量測試您新應用程式版本的運作狀態，同時讓舊的應用程式版本繼續處理其餘流量，便適合使用此部署政策。

下表比較部署方法屬性。

部署方法

方法	部署失敗的影響	部署時間	零停機	DNS 未變更	轉返程序	程式碼部署至
一次全部	停機	🕒	✗ 否	✓ 是	手動重新部署	現有執行個體
滾動	單一批次停止服務；失敗前即成功的任何批次，會執行新的應用程式版本	🕒🕒	✓ 是	✓ 是	手動重新部署	現有執行個體
以額外批次進行滾動	若第一個批次即失敗，則影響極小；否則與 Rolling (滾動) 的程度相近	🕒🕒	✓ 是	✓ 是	手動重新部署	新執行個體和現有執行個體
固定	極小	🕒🕒	✓ 是	✓ 是	終止新的執行個體	新執行個體
流量分割	暫時受到影響的並路由至新版本的用戶端流量百分比	🕒🕒	✓ 是	✓ 是	重新路由流量並終止新執行個體	新執行個體
藍/綠	極小	🕒🕒	✓ 是	✗ 否	交換 URL	新執行個體

† 依批次大小而異。

†† 會根據評估時間選項設定而異。

部署新的應用程式版本

您可以從環境的儀表板執行部署。

若要將新的應用程式版本部署至 Elastic Beanstalk 環境

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

> **Note**
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 選擇 Upload and deploy (上傳並部署)。
4. 使用畫面顯示表單來上傳應用程式的原始碼套件。
5. 選擇 Deploy (部署)。

重新部署舊版本

您亦可自應用程式版本頁面，將之前已上傳的應用程式舊版本部署至任何環境。

欲將現有應用程式版本部署至現有環境

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Applications (應用程式)，然後在清單上選取應用程式名稱。

> **Note**
> 如果您有許多應用程式，請使用搜尋列來篩選應用程式清單。

3. 在導覽窗格中，找到應用程式名稱並選擇 Application versions (應用程式版本)。
4. 選取要部署的 application version (應用程式版本)。
5. 選擇 Actions (動作)，然後選擇 Deploy (部署)。
6. 選取環境，然後選擇 Deploy (部署)。

其他部署應用程式的方式

若您經常部署，請考慮使用 Elastic Beanstalk 命令列界面 (EB CLI) 來管理您的環境。EB CLI 會在您的來源碼旁建立儲存庫。其也可以建立原始碼套件，將其上傳至 Elastic Beanstalk，並使用單一命令進行部署。

仰賴資源組態變更的部署，或仰賴無法與舊版本一同執行的新版本之部署，對此您可以啟動搭配新版本的新環境，並執行藍/綠部署的 CNAME 交換。

部署政策和設定

AWS Elastic Beanstalk 提供多種處理部署的選項，包括部署政策 (All at once [一次全部]、Rolling [滾動]、Rolling with additional batch [以額外批次進行滾動]、Immutable [不可變] 以及 Traffic splitting [流量分割]) 以及可讓您於部署期間設定批次大小和運作狀態檢查行為的部署選項。您的環境預設設定為一次使用全部部署。如果您使用 EB CLI 建立環境，且該環境為自動擴展環境 (您未指定 `--single` 選項)，則該環境會使用滾動部署。

透過「滾動部署」，Elastic Beanstalk 會將環境的 Amazon EC2 執行個體分割成批次，然後將應用程式的新版本一次部署到一個批次。其會將環境中的其餘執行個體維持在執行舊版本應用程式的狀態。在滾動部署期間，部分執行個體會以舊版應用程式處理要求，而批次已完成的執行個體將以新版本處理其他要求。如需詳細資訊，請參閱the section called “滾動部署運作方式”。

若要在部署期間維持完整容量，您可以先將環境設為啟動新的執行個體批次，再從服務中移除執行個體。此選項稱為「以額外批次進行滾動部署」。部署完成之後，Elastic Beanstalk 會終止額外的執行個體批次。

「不可變部署」會執行不可變的更新來啟動一組在不同 Auto Scaling 群組中執行新版本應用程式的完整新執行個體，並與執行舊版本應用程式的執行個體一同執行。不可變部署可避免滾動部署部分完成時所造成的問題。若新的執行個體未通過運作狀態檢查，Elastic Beanstalk 會將其終止，而原始執行個體保持不變。

「流量分割部署」可讓您將 Canary 測試做為應用程式部署的一部分執行。在流量分割部署中，Elastic Beanstalk 會啟動一組完整的新執行個體，就像在不可變的部署期間一樣。之後，其會在指定的評估期間，將指定百分比的傳入用戶端流量轉送至新的應用程式版本。如果新的執行個體運作狀態保持良好，Elastic Beanstalk 會將所有流量轉送給這些新執行個體，並終止舊的執行個體。如果新的執行個體並未通過運作狀態檢查，或是您選擇中止部署，Elastic Beanstalk 會將流量移回舊的執行個體並終止新的執行個體。絕對不會發生任何服務中斷現象。如需詳細資訊，請參閱the section called “流量分割部署的運作方式”。

> **Warning**
>
> 部署或更新期間，部分原則會取代所有執行個體。這會造成所有的累計 Amazon EC2 爆量餘額遺失。這發生的情況如下：
>
> - 執行個體更換啟用的受管平台更新
> - 不可變更新
> - 不可變更新或流量分割啟用的部署

若您的應用程式未通過所有運作狀態檢查，但仍能以較低等級的運作狀態正常運作，您可以修改 Healthy threshold (運作狀態閾值) 選項，藉此允許執行個體以較低等級的狀態 (如 `Warning`) 通過運作狀態檢查。若您的部署因運作狀態檢查未通過而失敗，而且無論運作狀態為何，您都需要強制進行更新，請指定 Ignore health check (忽略運作狀態檢查) 選項。

當您指定批次大小來進行滾動更新，Elastic Beanstalk 亦使用該值以滾動方式重新啟動應用程式。當您需要重新啟動環境中執行個體上執行的代理程式和應用程式伺服器，且不造成停機時間，請以滾動方式重新啟動。

設定應用程式部署

請於環境管理主控台中，透過編輯環境 Configuration (組態) 頁面上的 Updates and Deployments (更新和部署)，藉此啟用並設定以批次進行應用程式版本部署。

欲設定部署 (主控台)

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

   > **Note**
   >
   > 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇 Configuration (組態)。
4. 在 Rolling updates and deployments (滾動更新和部署) 組態類別中，選擇 Edit (編輯)。
5. 在 Application Deployments (應用程式部署) 區段，請選擇 Deployment policy (部署政策)、批次設定和運作狀態檢查選項。
6. 若要儲存變更，請選擇頁面底部的儲存變更。

Rolling updates and deployments (滾動更新和部署) 頁面的 Application deployments (應用程式部署) 區段，含有以下應用程式部署選項：

- Deployment policy (部署政策) - 自下列部署選項選擇：
  - All at once (一次全部) - 將新版本同時部署至所有執行個體。當部署開始後，您環境中的所有執行個體都將短暫停止服務。
  - Rolling (滾動) - 批次部署新的版本。各個批次將於部署階段自服務中移除，您環境的容量會減少一個批次的執行個體數量。
  - Rolling with additional batch (以額外批次進行滾動) - 批次部署新的版本，但會先啟動新的執行個體批次，藉此確保部署程序期間的完整容量。
  - Immutable (不可變) - 藉由執行不可變的更新作業，將新版本部署至全新的執行個體群組。
  - Traffic splitting (流量分割) - 將新版本部署到全新的執行個體群組，並暫時分割現有應用程式版本和新版本之間的傳入用戶端流量。

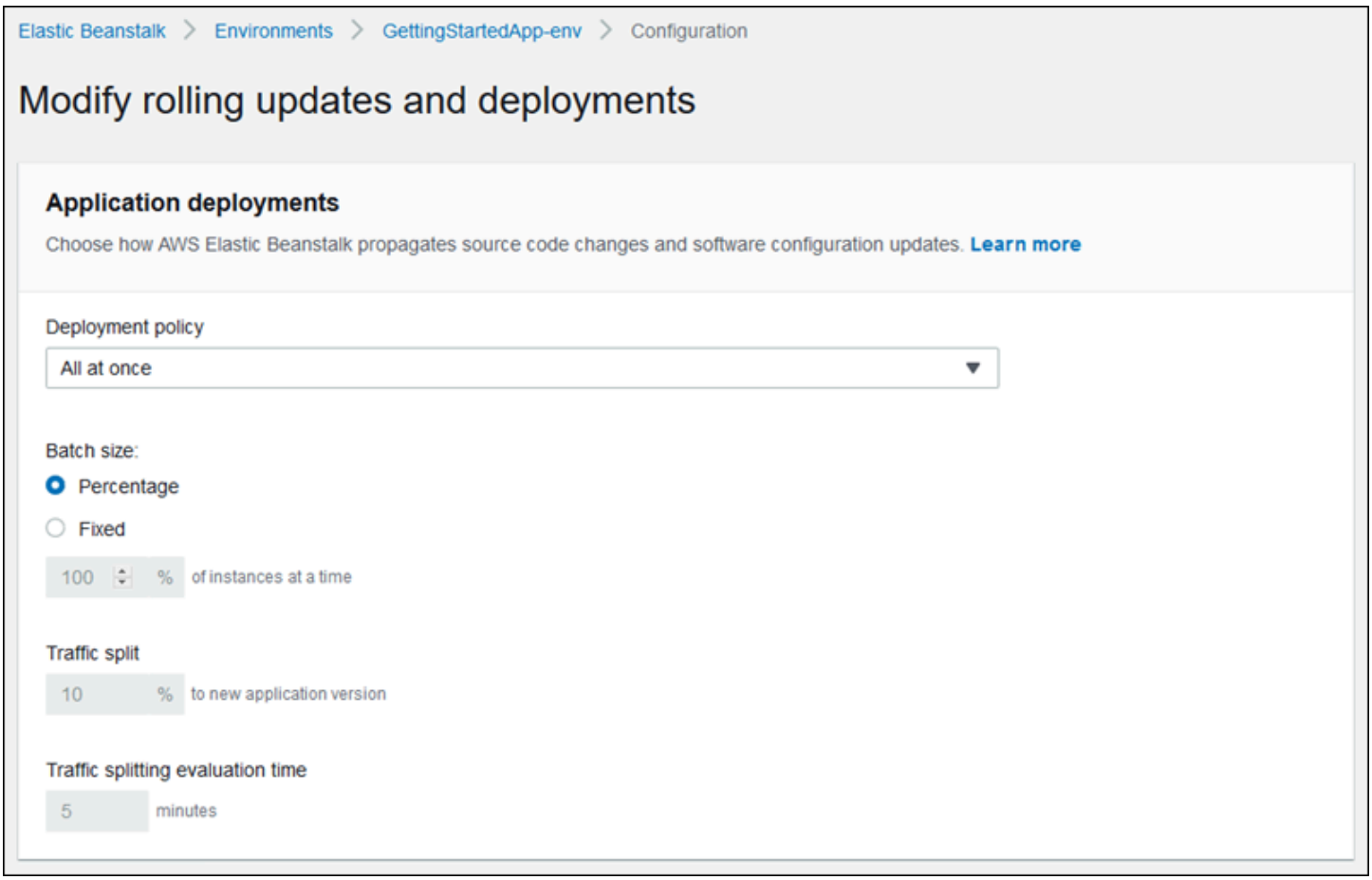
針對 Rolling (滾動) 和 Rolling with additional batch (以額外批次進行滾動) 部署政策，您可以設定：

- Batch size (批次大小) - 在每個批次中部署的一組執行個體的大小。

  選擇 Percentage (百分比) 設定 Auto Scaling 群組中 EC2 執行個體總數的百分比 (最多 100%)，或選擇 Fixed (固定) 設定固定數量的執行個體 (最多可達您環境的 Auto Scaling 組態最大執行個體計數)。

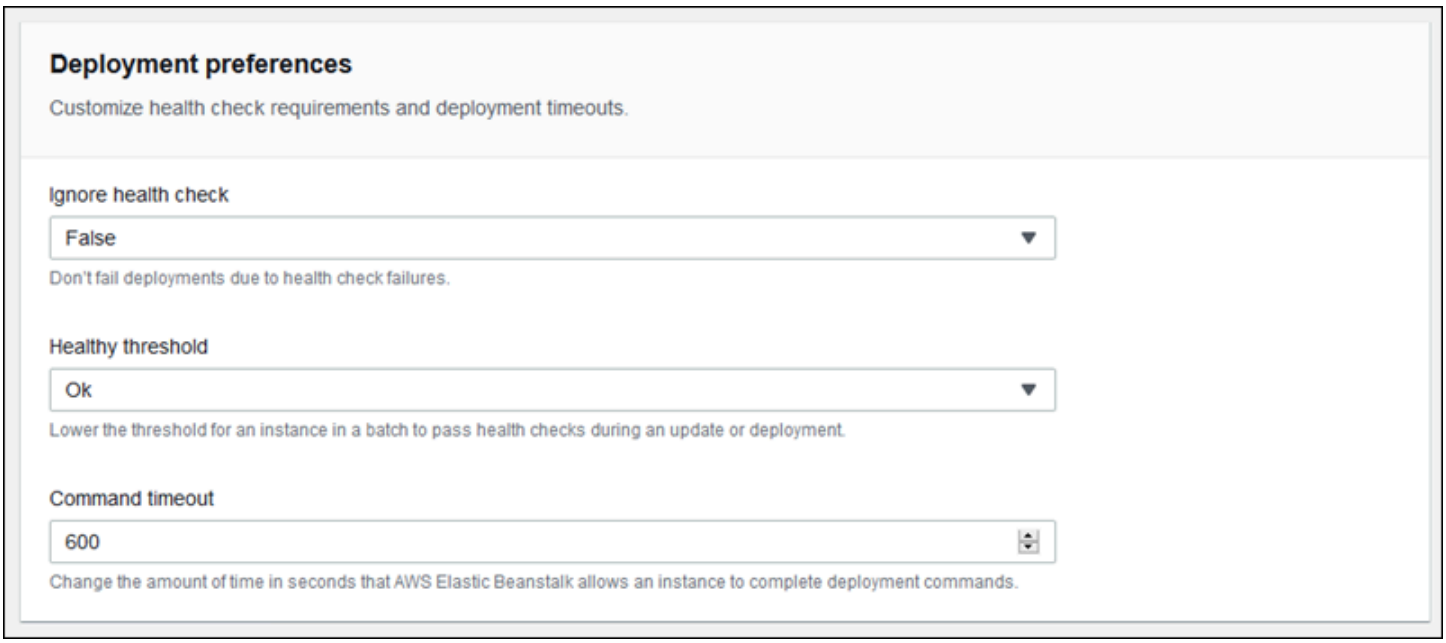
針對 Traffic splitting (流量分割) 部署政策，您可以設定以下內容：

- Traffic split (流量分割) - Elastic Beanstalk 移動到執行您正在部署新應用程式版本環境執行個體的傳入用戶端流量初始百分比。
- Traffic splitting evaluation time (流量分割評估時間) - 時間間隔 (以分鐘為單位)，Elastic Beanstalk 會在初始運作狀態良好的部署之後等待這段時間，再繼續將所有傳入用戶端流量移動到您正在部署的新應用程式版本。



Deployment preferences (部署偏好設定) 區段則包含關於運作狀態檢查的選項。

- Ignore health check (忽略運作狀態檢查) - 當一個批次無法於 Command timeout (命令逾時) 內恢復良好狀態時，可避免部署還原。
- Healthy threshold (運作狀態閾值) - 進行滾動部署、滾動更新及不可變更新時，若執行個體運作狀態良好，則可降低閾值。
- Command timeout (命令逾時) - 在取消部署之前，執行個體恢復良好狀態的等待秒數；若已設定 Ignore health check (忽略運作狀態檢查)，則是繼續下一批次前的等待秒數。



滾動部署運作方式

當一個批次正在處理時，Elastic Beanstalk 會將批次中的所有執行個體自負載平衡器分開，並部署新的應用程式版本，然後重新連接執行個體。若啟用連接耗盡，Elastic Beanstalk 會將批次內 Amazon EC2 執行個體的現有連線耗盡，之後才進行部署。

Elastic Load Balancing 將批次內的執行個體重新連接至負載平衡器後，會等待它們通過最小數量的 Elastic Load Balancing 運作狀態檢查 (Healthy check count threshold (運作狀態檢查計數閾值)) 數值，然後開始將流量路由傳送至這些執行個體。若未設定運作狀態檢查 URL，此流程可能會十分快速，因為只要執行個體接受 TCP 連線，就會立刻通過運作狀態檢查。若已設定運作狀態檢查 URL，負載平衡器則不會將流量路由至已更新的執行個體，除非執行個體回傳 `200 OK` 狀態碼以回應運作狀態檢查 URL 的 `HTTP GET` 請求。

Elastic Beanstalk 會等到批次內所有執行個體皆正常運作，才會繼續下一批次。若採用基礎型運作狀態報告，執行個體運作狀態取決於 Elastic Load Balancing 運作狀態檢查狀態。當批次內所有執行個體皆通過足以讓 Elastic Load Balancing 視為良好運作的運作狀態檢查時，此批次就會完成。若啟用增強型運作狀態報告，Elastic Beanstalk 會考量其他數個因素，包括傳入請求的結果。使用增強型運作狀態報告時，Web 伺服器環境的所有執行個體都必須在兩分鐘內通過 12 項連續運作狀態檢查並取得 OK 狀態，而工作者環境則必須在 3 分鐘內通過 18 項運作狀態檢查。

若執行個體批次未於命令逾時期間內恢復正常運作，則部署會失敗。部署失敗後，會檢查環境中執行個體的運作狀態，以了解失敗的原因。然後執行其他部署具備固定或已知良好版本的應用程式與復原。

在一個或多個批次順利完成後仍部署失敗，已完成的批次會執行應用程式的新版本，而待定批次會持續執行舊的版本。您可於主控台的運作狀態頁面，辨識您環境執行個體執行的版本。此頁面會顯示您環境執行個體最近執行的部署 ID。若您因部署失敗而終止執行個體，Elastic Beanstalk 會將其取代為執行最近成功部署的應用程式版本之執行個體。

流量分割部署的運作方式

流量分割部署可讓您執行 Canary 測試。您可以先將一部分傳入用戶端流量導向您的新應用程式版本，驗證應用程式的運作狀態，再遞交到新版本並將所有流量導向新版本。

在流量分割部署期間，Elastic Beanstalk 會在個別的暫時 Auto Scaling 群組中建立新的一組執行個體。Elastic Beanstalk 接著會指示負載平衡器將您環境傳入流量中特定百分比的流量導向新的執行個體。然後，Elastic Beanstalk 會依設定的時間長度，持續追蹤這組新執行個體的運作狀態。如果一切順利，Elastic Beanstalk 會將其餘流量移動到新的執行個體，並將這些執行個體連接到環境的原始 Auto Scaling 群組，取代舊的執行個體。然後 Elastic Beanstalk 會進行清理，即終止舊執行個體並移除暫時 Auto Scaling 群組。

> **Note**
>
> 在流量分割部署期間，環境的容量不會變更。Elastic Beanstalk 會在暫時的 Auto Scaling 群組中啟動執行個體，其數量與部署開始時原始 Auto Scaling 群組中的執行個體數相同。接著在部署的持續時間內，兩個 Auto Scaling 群組都維持固定的執行個體數。請在設定環境的流量分割評估時間時將此納入考量。

將部署轉返到先前的應用程式版本相當快速，且不會對服務用戶端流量造成影響。如果新的執行個體並未通過運作狀態檢查，或是您選擇中止部署，Elastic Beanstalk 會將流量移回舊的執行個體並終止新的執行個體。您可以使用 Elastic Beanstalk 主控台的環境概觀頁面，選擇 Environment actions (環境動作) 中的 Abort current operation (中止目前操作) 來中止任何部署。您也可以呼叫 AbortEnvironmentUpdate API 或相等的 AWS CLI 命令。

流量分割部署需要 Application Load Balancer。Elastic Beanstalk 根據預設會在您使用 Elastic Beanstalk 主控台或 EB CLI 建立環境時使用此負載平衡器類型。

部署選項命名空間

您可以使用 `aws:elasticbeanstalk:command` 命名空間中的組態選項來設定您的部署。若您選擇流量分割政策，此政策的其他選項可在 `aws:elasticbeanstalk:trafficsplitting` 命名空間中取得。

透過 DeploymentPolicy 選項設定部署類型，支援下列值：

- AllAtOnce - 停用滾動部署，並一律同時部署至所有執行個體。
- Rolling - 啟用標準滾動部署。
- RollingWithAdditionalBatch - 啟動執行個體的額外批次，再啟動部署，以維持完整容量。
- Immutable - 針對每個部署執行不可變更新。
- TrafficSplitting - 執行流量分割部署來針對您的應用程式部署進行 Canary 測試。

當您啟用滾動部署，請設定 BatchSize 和 BatchSizeType 選項，進行各個批次大小的設定。例如，若要在每個批次中部署 25% 的執行個體，請指定下列選項和值。

Example .ebextensions/rolling-updates.config

```
option_settings:
  aws:elasticbeanstalk:command:
    DeploymentPolicy: Rolling
    BatchSizeType: Percentage
    BatchSize: 25
```

無論執行中的執行個體數量為何，若要在每個批次中部署五個執行個體，並在從服務移除執行個體前安裝五個執行新版本的執行個體額外批次，請指定下列選項和值。

Example .ebextensions/rolling-additionalbatch.config

```
option_settings:
  aws:elasticbeanstalk:command:
    DeploymentPolicy: RollingWithAdditionalBatch
    BatchSizeType: Fixed
    BatchSize: 5
```

欲針對運作狀態檢查閾值為 Warning (警告) 的部署執行不可變動的更新作業，而且即使批次內執行個體未於 15 分鐘的逾時期間內通過運作狀態檢查，仍要繼續部署，請指定下列選項和值。

Example .ebextensions/immutable-ignorehealth.config

```
option_settings:
  aws:elasticbeanstalk:command:
    DeploymentPolicy: Immutable
    HealthCheckSuccessThreshold: Warning
```

```
    IgnoreHealthCheck: true
    Timeout: "900"
```

如要執行流量分割部署，請將用戶端流量的 15% 轉送至新的應用程式版本並評估運作狀態 10 分鐘，請遵循下列選項和值。

Example .ebextensions/traffic-splitting.config

```
option_settings:
  aws:elasticbeanstalk:command:
    DeploymentPolicy: TrafficSplitting
  aws:elasticbeanstalk:trafficsplitting:
    NewVersionPercent: "15"
    EvaluationTime: "10"
```

EB CLI 和 Elastic Beanstalk 主控台會為前述選項套用建議的數值。若您想要使用組態檔進行相同的設定，您必須移除這些設定。如需詳細資訊，請參閱「建議值」。

透過 Elastic Beanstalk 進行藍/綠部署

由於 AWS Elastic Beanstalk 會於您更新應用程式版本時就地執行更新，使用者可能會在短時間內無法使用您的應用程式。若是要避免發生這類情狀，請執行藍/綠部署。如果要進行這項操作，可以將新版本部署至獨立環境，然後交換兩個環境的 CNAME 以立刻將流量重新引導至新版本。

若是您想要將環境更新到不相容的平台版本，也需要使用藍/綠部署。如需更多詳細資訊，請參閱 the section called “平台更新”。

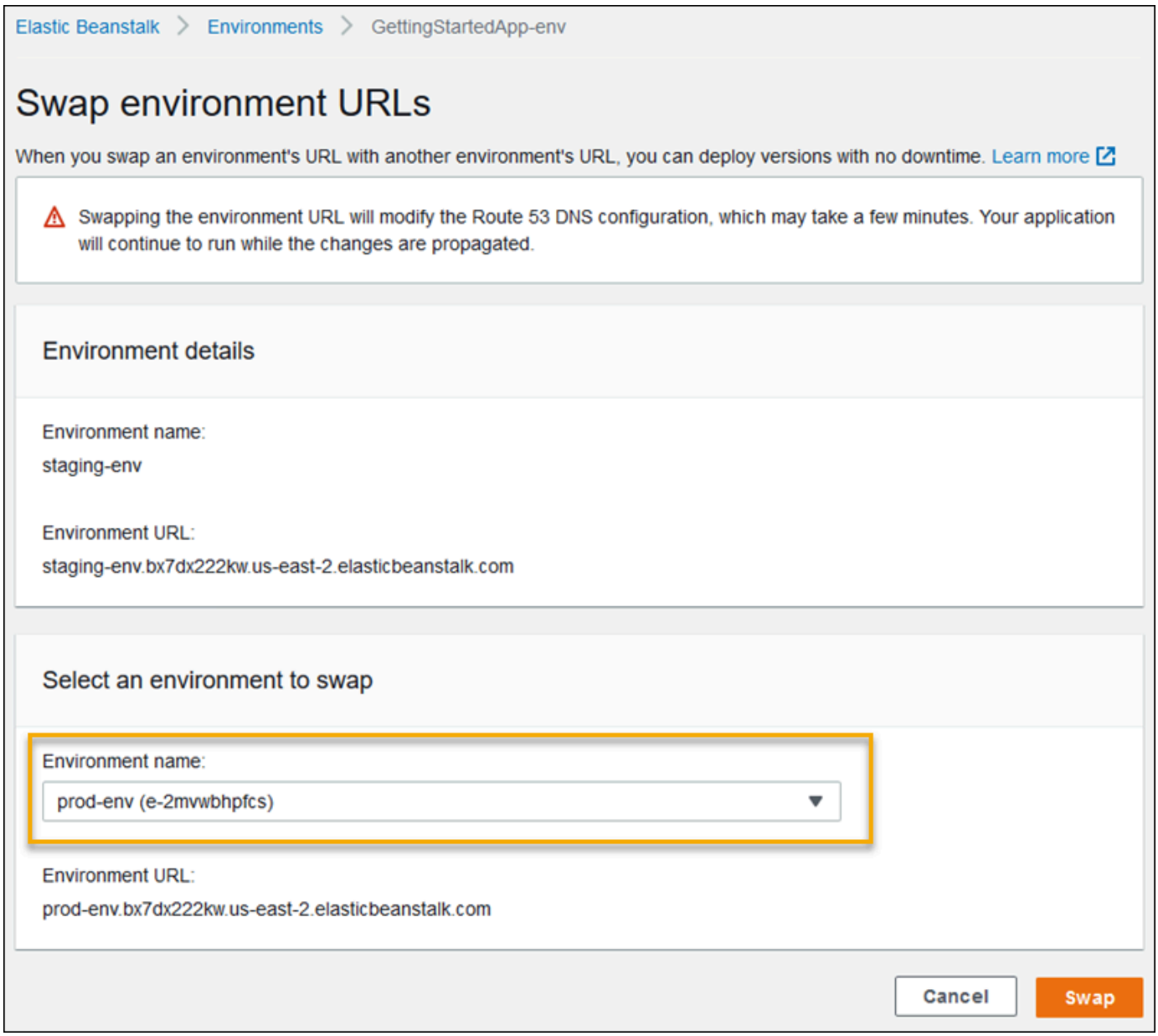
若您的應用程式使用生產資料庫，藍/綠部署會要求您的環境獨立於此資料庫執行。如果您的環境包含 Elastic Beanstalk 代表您建立的資料庫，則除非您採取特定的動作，否則不會保留環境的資料庫和連線。如果您有想要保留的資料庫，請使用其中一個 Elastic Beanstalk 資料庫生命週期選項。您可以選擇「Retain」(保留) 選項，以便在資料庫解耦後保持資料庫和環境運作。如需詳細資訊，請參閱在本指南的《設定環境》章節中的 資料庫生命週期。

如需如何設定應用程式以連接至非由 Elastic Beanstalk 管理的 Amazon RDS 執行個體的相關說明，請參閱 搭配 Amazon RDS 使用 Elastic Beanstalk。

欲執行藍/綠部署

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 複製目前環境，或啟動新環境，以執行您想要的平台版本。
3. 部署新的應用程式版本至新環境。

4. 於新環境測試新的版本。
5. 在環境概觀頁面上，選擇 Actions (動作)，然後選擇 Swap environment URLs (交換環境 URL)。
6. 在 Environment name (環境名稱)，選取目前環境。



7. 選擇 Swap (交換)。

Elastic Beanstalk 會交換新舊環境的 CNAME 記錄，將流量自舊版本重新引導至新版本。

Elastic Beanstalk 完成交換操作後，請驗證在您嘗試連接至舊環境 URL 時，新環境是否回應。然而，在 DNS 變更散佈完成且舊 DNS 記錄過期前，請勿終止您的舊環境。DNS 伺服器不一定會根據您在 DNS 記錄上設定的存留時間 (TTL)，從快取中清除舊的記錄。

組態變更

在環境管理主控台內的 Configuration (組態) 區段修改組態選項設定時，AWS Elastic Beanstalk 會將變更傳播至所有受影響的資源。這些資源包括將流量分配至執行您應用程式之 Amazon EC2 執行個體的負載平衡器、管理這些執行個體的 Auto Scaling 群組及 EC2 執行個體本身。

許多組態變更可套用至執行環境，無須替換現有的執行個體。例如，設定運作狀態檢查 URL 會觸發環境更新來修改負載平衡器設定，但不會產生停機時間，因為執行您應用程式的執行個體會在更新傳播時持續處理請求。

修改啟動組態或 VPC 設定的組態變更，需要終止環境中所有執行個體，並加以替換。例如，變更環境的執行個體類型或 SSH 金鑰設定時，必須終止及替換您的 EC2 執行個體。Elastic Beanstalk 提供了幾個決定如何進行替換的原則。

- 滾動更新 – Elastic Beanstalk 會以批次套用這些組態變更，讓最低數量的執行個體維持運作並隨時處理流量。此方法可避免更新程序期間的停機時間。如需詳細資訊，請參閱滾動更新。
- 不可變更新 – Elastic Beanstalk 會透過以新組態執行之另一組執行個體，在您的環境外啟動臨時的 Auto Scaling 群組。接著，Elastic Beanstalk 會將這些執行個體放置於您環境的負載平衡器後方。新舊執行個體可同時處理流量，直到新的執行個體通過運作狀態檢查為止。此時，Elastic Beanstalk 會將新執行個體移至您環境的 Auto Scaling 群組中，並終止臨時群組和舊執行個體。如需詳細資訊，請參閱不可變更新。
- 停用 – Elastic Beanstalk 不會嘗試避免停機時間。它會終止您環境的現有執行個體並將其取代為以新組態執行的新執行個體。

> **Warning**
>
> 部署或更新期間，部分原則會取代所有執行個體。這會造成所有的累計 Amazon EC2 爆量餘額遺失。這發生的情況如下：
>
> - 執行個體更換啟用的受管平台更新
> - 不可變更新
> - 不可變更新或流量分割啟用的部署

支援的更新類型

滾動更新設定	負載平衡環境	單一執行個體環境	舊版 Windows Server 環境†
已停用	✓是	✓是	✓是
根據運作狀態進行滾動	✓是	×否	✓是
根據時間進行滾動	✓是	×否	✓是
固定	✓是	✓是	×否

† 針對此表格的用途，「傳統 Windows Server 環境」是以使用 IIS 8.5 以前版本的 Windows Server 平台組態為基礎的環境。

主題

- Elastic Beanstalk 滾動環境資訊更新
- 不可變的環境更新

Elastic Beanstalk 滾動環境資訊更新

當組態變更需要替換執行個體時，Elastic Beanstalk 能夠以批次方式執行更新，避免在傳播變更期間發生停機。在滾動更新期間，您可將容量設定為僅減少單一批次的大小。Elastic Beanstalk 會停止單一批次執行個體的服務，將其終止，然後啟動具備新組態的批次。新的批次開始處理請求後，Elastic Beanstalk 會繼續進行下一批次。

滾動組態更新的批次可根據時間定期處理 (每個批次間會出現延遲)，或是根據運作狀態處理。對於根據時間進行的滾動更新，您可設定 Elastic Beanstalk 完成啟動一批次執行個體後的等待時間，之後才繼續進行下一批次。此暫停時間可讓您的應用程式進行啟動載入作業，並開始處理請求。

對於根據運作狀態進行的滾動更新，Elastic Beanstalk 繼續進行下一批次前，會等待前一批次的執行個體通過運作狀態檢查。執行個體的運作狀態由運作狀態報告系統 (基礎型或增強型) 判定。若是基本運作狀態，單一批次內的所有執行個體一通過 Elastic Load Balancing (ELB) 運作狀態檢查，即可視為運作狀態良好。

若是增強型運作狀態報告，單一批次內的所有執行個體都必須連續通過多次運作狀態檢查，之後 Elastic Beanstalk 才會繼續進行下一批次。增強型運作狀態除了具備只檢查執行個體的 ELB 運作狀態

檢查，亦會監控應用程式日誌以及環境其他資源的狀態。若 Web 伺服器環境搭配增強型運作狀態，所有執行個體必須在兩分鐘內通過 12 個運作狀態檢查 (工作者環境則是在三分鐘內通過 18 個檢查)。若一個執行個體無法通過一項運作狀態檢查，則計數會重設。

若單一批次未在滾動更新逾時內 (預設為 30 分鐘) 恢復良好運作，則會取消更新。滾動更新逾時是可在 `aws:autoscaling:updatepolicy:rollingupdate` 命名空間使用的組態選項。若您的應用程式未通過運作狀態檢查 (即未處於 Ok 狀態)，但就另一層級而言仍屬穩定，您可於 `aws:elasticbeanstalk:healthreporting:system` 命名空間中設定 `HealthCheckSuccessThreshold` 選項，變更在 Elastic Beanstalk 中視執行個體運作良好的層級。

若滾動更新程序失敗，Elastic Beanstalk 會開始另一次滾動更新，以復原為之前的組態。滾動更新失敗的原因可能是未通過運作狀態檢查，或是由於啟動新執行個體導致超過您帳戶的配額。例如，若您達到 Amazon EC2 執行個體的數量配額，滾動更新可能會在嘗試佈建新一批次的執行個體時失敗。在此情況下，轉返亦會失敗。

轉返失敗會結束更新程序，讓您的環境停留在運作狀態不佳的狀態。尚未處理的批次仍使用舊的組態運行執行個體，而成功完成的批次則會具備新的組態。欲修正轉返失敗後的環境，請先解決造成更新失敗的根本原因，然後啟動另一次環境更新。

另一個方法是將新版本的應用程式部署至不同環境，然後執行 CNAME 交換重新導向流量，達到零停機時間。如需詳細資訊，請參閱 透過 Elastic Beanstalk 進行藍/綠部署。

滾動更新與滾動部署比較

進行滾動更新的情況是，當您變更的設定需要為環境佈建新的 Amazon EC2 執行個體，這包括變更 Auto Scaling 群組組態 (如執行個體類型和金鑰對設定) 以及變更 VPC 設定。滾動更新會先終止各個批次的執行個體，再佈建新的批次予以替換。

進行滾動部署的情況是當您部署應用程式的時候，且滾動部署通常無須替換環境中的執行個體。Elastic Beanstalk 會停止各個批次的服務，部署新的應用程式版本，然後恢復其服務。

例外情況是當您變更需要替換執行個體的設定時，同時部署新的應用程式版本。例如，如果您變更原始碼套件中組態檔案的金鑰名稱設定並將其部署至您的環境，即會觸發滾動更新。此時不會將新的應用程式版本部署至現有執行個體的各個批次，而是會佈建具備新組態的新批次執行個體。在此情況下，不會另行部署，因為新的應用程式版本會採用新的執行個體。

只要環境更新包含佈建新的執行個體，就會出現部署階段，此時應用程式原始碼會部署至新的執行個體，且會套用修改執行個體上作業系統或軟體的組態設定。部署運作狀態檢查設定 (Ignore health check (忽略運作狀態檢查)、Healthy threshold (運作狀態閾值) 和 Command timeout (命令逾時)) 也會於部署期間，套用至根據運作狀態進行的滾動更新和不可變更新。

設定滾動更新

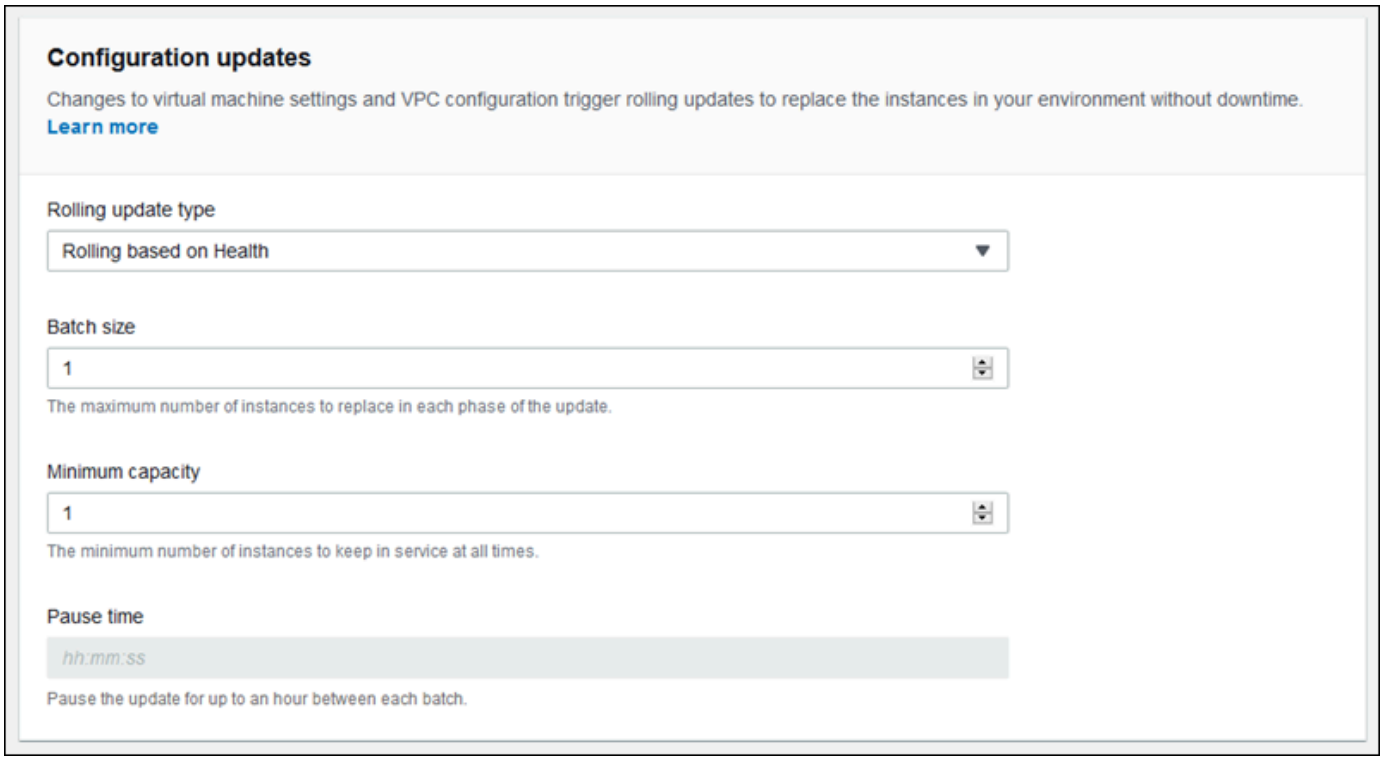
您可於 Elastic Beanstalk 主控台啟用並設定滾動更新。

欲啟用滾動更新

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇 Configuration (組態)。
4. 在 Rolling updates and deployments (滾動更新和部署) 組態類別中，選擇 Edit (編輯)。
5. 在 Configuration updates (組態更新) 區段中，針對 Rolling update type (滾動更新類型)，選取其中一個 Rolling (滾動) 選項。



6. 選擇 Batch size (批次大小)、Minimum capacity (容量下限) 和 Pause time (暫停時間) 設定。
7. 若要儲存變更，請選擇頁面底部的儲存變更。

Rolling updates and deployments (滾動更新和部署) 頁面的 Configuration updates (組態更新) 區段，具有以下滾動更新的選項：

- Rolling update type (滾動更新類型) – Elastic Beanstalk 完成更新一批次執行個體後的等待時間，之後才繼續進行下一批次，以允許這些執行個體完成啟動載入作業並開始處理流量。您可以從以下選項中選擇：
  - Rolling based on Health (根據運作狀態進行滾動) – 等待目前批次內執行個體恢復良好運作並恢復服務，之後才繼續下一批次。
  - Rolling based on Time (根據時間進行滾動) – 指定新執行個體啟動與其恢復服務並進行下一批次間的等待時間。
  - Immutable (不可變) – 透過不可變的更新作業，將組態變更套用至全新的執行個體群組。
- Batch size (批次大小) – 各個批次欲替換的執行個體數量 (介於 **1** 和 **10000** 之間)。根據預設，此數值為 Auto Scaling 群組大小下限的三分之一，無條件進位至整數。
- Minimum capacity (最低容量) – 更新其他執行個體同時保持執行的執行個體數量下限 (介於 **0** 和 **9999** 之間)。預設值為 Auto Scaling 群組的規模下限或 Auto Scaling 群組的規模上限減一，以較低者為準。
- Pause time (暫停時間) (僅適用根據時間進行的更新) – 批次更新完成直到繼續進行下一批前的等待時間，讓您的應用程式能夠開始接收流量。介於 0 秒與 1 小時之間。

aws:autoscaling:updatepolicy:rollingupdate 命名空間

您亦可使用 `aws:autoscaling:updatepolicy:rollingupdate` 命名空間中的組態選項來設定滾動更新。

使用 `RollingUpdateEnabled` 選項來啟用滾動更新，`RollingUpdateType` 則可選擇更新類型。`RollingUpdateType` 支援下列值：

- `Health` (根據運作狀態進行滾動) – 等待目前批次內執行個體恢復良好運作並恢復服務，之後才繼續下一批次。
- `Time` – 指定新執行個體啟動與其恢復服務並進行下一批次間的等待時間。
- `Immutable` – 透過不可變的更新作業，將組態變更套用至全新的執行個體群組。

當您啟用滾動更新，請設定 `MaxBatchSize` 和 `MinInstancesInService` 選項，進行各個批次大小的設定。以根據時間和根據運作狀態的滾動更新而言，您亦可分別設定 `PauseTime` 和 `Timeout`。

例如，欲一次啟動最多五個執行個體同時至少兩個執行個體保持服務，並在批次之間等待 5 分 30 秒，請指定下列選項與值。

Example .ebextensions/timebased.config

```
option_settings:
  aws:autoscaling:updatepolicy:rollingupdate:
    RollingUpdateEnabled: true
    MaxBatchSize: 5
    MinInstancesInService: 2
    RollingUpdateType: Time
    PauseTime: PT5M30S
```

欲啟用根據運作狀態的滾動更新，且各個批次的逾時為 45 分鐘，請指定下列選項與值：

Example .ebextensions/healthbased.config

```
option_settings:
  aws:autoscaling:updatepolicy:rollingupdate:
    RollingUpdateEnabled: true
    MaxBatchSize: 5
    MinInstancesInService: 2
    RollingUpdateType: Health
    Timeout: PT45M
```

`Timeout` 和 `PauseTime` 值都必須以 ISO8601 持續時間格式指定：PT#H#M#S，其中每個 # 分別為時數、分鐘數或秒數。

EB CLI 和 Elastic Beanstalk 主控台會為前述選項套用建議的數值。若您想要使用組態檔進行相同的設定，您必須移除這些設定。如需詳細資訊，請參閱「建議值」。

不可變的環境更新

不可變的環境更新是滾動更新的替代選項。不可變環境更新可確保需要取代執行個體的組態變更有效安全執行。若不可變環境更新失敗，轉返程序只要終止 Auto Scaling 群組即可。另一方面，若滾動更新失敗，則需要另外執行滾動更新來還原變更。

欲執行不可變環境更新，Elastic Beanstalk 會於環境負載平衡器後方建立第二個臨時 Auto Scaling 群組，以容納新的執行個體。首先，Elastic Beanstalk 會於新群組內，使用新組態來啟動單一執行個體。此執行個體會處理流量，搭配執行之前組態的原始 Auto Scaling 群組中的所有執行個體運作。

第一個執行個體通過運作狀態檢查後，Elastic Beanstalk 會使用新組態來啟動額外的執行個體，總執行個體數量與原始 Auto Scaling 群組內執行的數量相符。所有新執行個體通過運作狀態檢查後，Elastic Beanstalk 會將其傳輸至原始 Auto Scaling 群組內，並終止臨時 Auto Scaling 群組和舊的執行個體。

> **Note**
>
> 不可變環境更新期間，當新的 Auto Scaling 群組的執行個體開始處理請求後，在原始 Auto Scaling 群組執行個體終止前，您環境的容量會短暫加倍。若您的環境擁有多個執行個體，或您的隨需執行個體配額較低，請確認您的容量足以執行不可變環境更新。若您已接近配額，請考慮改使用滾動更新。

不可變更新需要增強型運作狀態報告，以評估環境於更新期間的運作狀態。增強型運作狀態報告結合標準負載平衡器的運作狀態檢查與執行個體監控，以確保執行新組態的執行個體成功處理請求。

您亦可將不可變更新做為滾動部署的替代方法，藉此部署應用程式的新版本。若您將 Elastic Beanstalk 的應用程式部署設定為使用不可變更新，則在您每次部署新的應用程式版本時，將會取代環境中的所有執行個體。若不可變應用程式部署失敗，Elastic Beanstalk 會終止新的 Auto Scaling 群組以立即回復變更。這可避免部分批次完成後，在滾動更新失敗時出現部分陣列部署。

> **Warning**
>
> 部署或更新期間，部分原則會取代所有執行個體。這會造成所有的累計 Amazon EC2 爆量餘額遺失。這發生的情況如下：
>
> - 執行個體更換啟用的受管平台更新
> - 不可變更新
> - 不可變更新或流量分割啟用的部署

若不可變更新失敗，新的執行個體在 Elastic Beanstalk 將其終止前，會上傳套件日誌至 Amazon S3。Elastic Beanstalk 在刪除不可變更新失敗的日誌前，會於 Amazon S3 保留一小時，而非套件或結尾日誌的標準 15 分鐘。

> **Note**
>
> 若您將不可變更新用於應用程式版本部署，但未用於組態，則當您嘗試部署的應用程式版本內含通常會觸發滾動更新的組態變更時 (例如變更執行個體類型的組態)，可能會出現錯誤。欲避免此情況，請於不同更新作業內進行組態變更，或將不可變更新設定為用於部署及組態變更。

您的不可變更新執行無法搭配資源組態變更。例如，您無法在變更需要替換執行個體的設定，同時更新其他設定，或使用會變更組態設定或原始碼內其他資源的組態檔案，執行不可變部署。若您嘗試變更資源設定 (如負載平衡器設定)，並同時執行不可變更新，Elastic Beanstalk 會傳回錯誤。

若您的資源組態變更不需要變更原始碼或執行個體組態，請於兩次更新作業中執行變更。若須仰賴它們，請改為執行藍/綠部署。

設定不可變更新

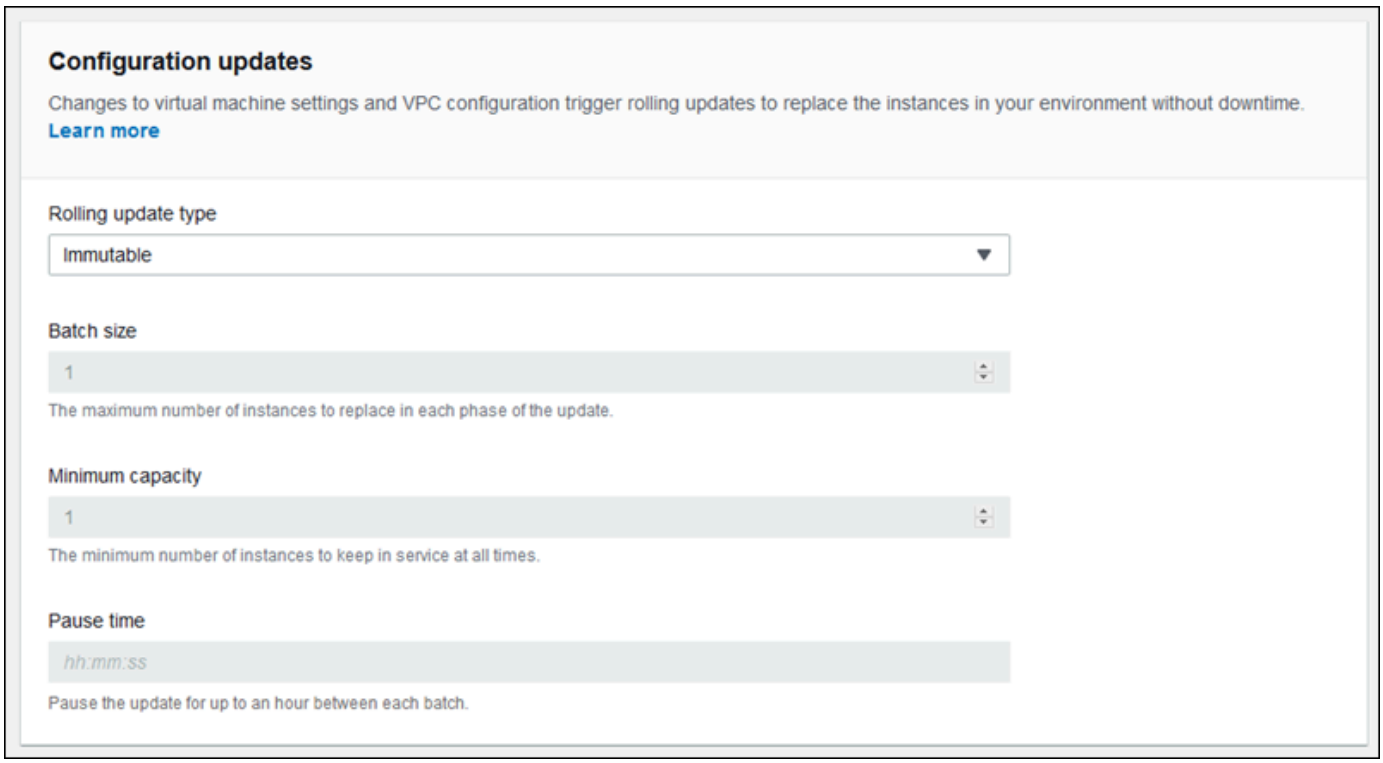
您可於 Elastic Beanstalk 主控台啟用並設定不可變更新。

欲啟用不可變更新 (主控台)

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

   > **Note**
   >
   > 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇 Configuration (組態)。
4. 在 Rolling updates and deployments (滾動更新和部署) 組態類別中，選擇 Edit (編輯)。
5. 在 Configuration Updates (組態更新) 區段，將 Rolling update type (滾動更新類型) 設定為 Immutable (不可變)。



6. 若要儲存變更，請選擇頁面底部的儲存變更。

aws:autoscaling:updatepolicy:rollingupdate 命名空間

您亦可使用 `aws:autoscaling:updatepolicy:rollingupdate` 命名空間中的選項來設定不可變更新。下列範例組態檔案會針對組態變更啟用不可變更新。

Example .ebextensions/immutable-updates.config

```
option_settings:
  aws:autoscaling:updatepolicy:rollingupdate:
    RollingUpdateType: Immutable
```

下列範例會針對組態變更及部署兩者啟用不可變更新。

Example .ebextensions/immutable-all.config

```
option_settings:
  aws:autoscaling:updatepolicy:rollingupdate:
    RollingUpdateType: Immutable
  aws:elasticbeanstalk:command:
    DeploymentPolicy: Immutable
```

EB CLI 和 Elastic Beanstalk 主控台會為前述選項套用建議的數值。若您想要使用組態檔進行相同的設定，您必須移除這些設定。如需詳細資訊，請參閱「建議值」。

更新您 Elastic Beanstalk 環境的平台版本

Elastic Beanstalk 會定期推出新的平台版本，更新所有以 Linux 和 Windows Server 為基礎的平台。新平台版本提供現有軟體元件的更新，並支援新的功能和組態選項。若要了解平台和平台版本，請參閱 Elastic Beanstalk 平台詞彙表。

您可以使用 Elastic Beanstalk 主控台或 EB CLI 來更新環境的平台版本。Elastic Beanstalk 會根據您要更新的目標平台版本，建議您使用下列兩種方法的其中一種來執行平台更新。

- 方法 1 – 更新您環境的平台版本. 若您要在平台分支內更新到最新的平台版本，但不變更執行時間、Web 伺服器、應用程式伺服器和作業系統，也不變更主要平台版本，則我們建議您使用這個方法。這是最常見的例行平台更新。
- 方法 2 – 執行藍/綠部署. 若您要在另一個平台分支內更新到最新的平台版本，而且要變更執行時間、Web 伺服器、應用程式伺服器或作業系統，還要變更主要平台版本，則我們建議您使用這個方法。當您想利用新的執行時間功能或最新的 Elastic Beanstalk 功能，或者當您想要移出已取代或已淘汰的平台分支時，這是一個很好的方法。

  從舊版平台進行遷移需要使用藍/綠部署，因為這些平台版本與目前支援的版本不相容。

  將 Linux 應用程式遷移至 Amazon Linux 2 需要使用藍/綠部署，因為 Amazon Linux 2 平台版本與舊版 Amazon Linux AMI 平台不相容。

如需選擇最佳平台更新方法的詳細說明，請展開您環境平台的區段。

Docker

使用方法 1 執行平台更新。

多容器 Docker

使用方法 1 執行平台更新。

預先設定的 Docker

請考慮下列情況：

- 如果您要將應用程式遷移到其他平台 (例如從 Go 1.4 (Docker) 遷移到 Go 1.11，或從 Python 3.4 (Docker) 遷移到 Python 3.6)，請使用方法 2。
- 如果您要將應用程式遷移到其他 Docker 容器版本 (例如從 Glassfish 4.1 (Docker) 遷移到 Glassfish 5.0 (Docker))，請使用方法 2。
- 如果您要更新到最新的平台版本，不變更容器版本或主要版本，請使用方法 1。

Go

使用方法 1 執行平台更新。

Java SE

請考慮下列情況：

- 如果您要將應用程式遷移到其他 Java 執行時間版本 (例如從 Java 7 遷移到 Java 8)，請使用方法 2。
- 如果您要更新到最新平台版本，不變更執行時間版本，請使用方法 1。

Java 搭配 Tomcat

請考慮下列情況：

- 如果您要將應用程式遷移到其他 Java 執行時間版本或 Tomcat 應用程式伺服器版本 (例如從搭配 Tomcat 7 的 Java 7 遷移到搭配 Tomcat 8.5 的 Java 8)，請使用方法 2。
- 如果您要將應用程式遷移到其他搭配 Tomcat 的主要 Java 平台版本 (1.x.x 版、2.x.x 版和 3.x.x 版)，請使用方法 2。
- 如果您要更新到最新的平台版本，不變更執行時間版本、應用程式伺服器版本或主要版本，請使用方法 1。

Windows Server 上的 .NET 搭配 IIS

請考慮下列情況：

- 如果您要將應用程式遷移到其他 Windows 作業系統版本 (例如從 Windows Server 2008 R2 更新到 Windows Server 2016)，請使用方法 2。
- 如果您要將應用程式遷移到其他主要 Windows Server 平台版本，請參閱 從先前的 Windows Server 平台主要版本遷移，並使用方法 2。

- 如果您的應用程式目前正在 Windows Server 平台 2.x.x 版上執行，且您要更新到最新的平台版本，請使用方法 1。

> **Note**
>
> 並未在語意上建立第 2 版之前的 Windows Server 平台版本。您只能啟動每個 Windows Server 主要平台版本中的最新版本，且無法在升級之後轉返。

Node.js

使用方法 2 執行平台更新。

PHP

請考慮下列情況：

- 如果您要將應用程式遷移到其他 PHP 執行時間版本 (例如從 PHP 5.6 遷移到 PHP 7.2)，請使用方法 2。
- 如果您要將應用程式遷移到其他主要 PHP 平台版本 (1.x.x 版和 2.x.x 版)，請使用方法 2。
- 如果您要更新到最新平台版本，不變更執行時間版本或主要版本，請使用方法 1。

Python

請考慮下列情況：

- 如果您要將應用程式遷移到其他 Python 執行時間版本 (例如從 Python 2.7 遷移到 Python 3.6)，請使用方法 2。
- 如果您要將應用程式遷移到其他主要 Python 平台版本 (1.x.x 版和 2.x.x 版)，請使用方法 2。
- 如果您要更新到最新平台版本，不變更執行時間版本或主要版本，請使用方法 1。

Ruby

請考慮下列情況：

- 如果您要將應用程式遷移到其他 Ruby 執行時間版本或應用程式伺服器版本 (例如從搭配 Puma 的 Ruby 2.3 遷移到搭配 Puma 的 Ruby 2.6)，請使用方法 2。

- 如果您要將應用程式遷移到其他主要 Ruby 平台版本 (1.x.x 版和 2.x.x 版)，請使用方法 2。
- 如果您要更新到最新的平台版本，不變更執行時間版本、應用程式伺服器版本或主要版本，請使用方法 1。

方法 1 – 更新您環境的平台版本

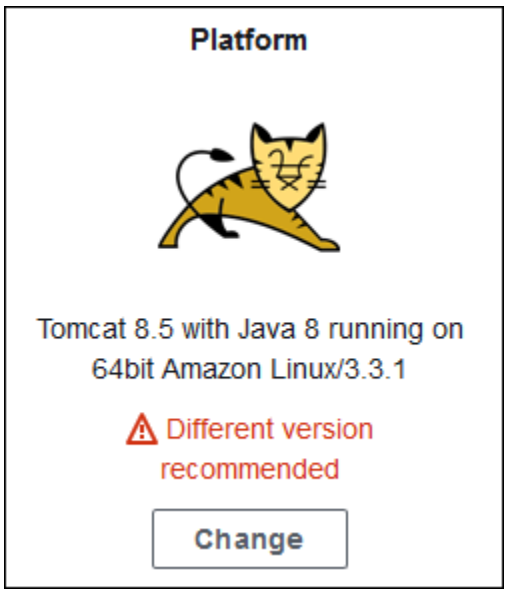
使用此方式將您環境的平台分支更新到最新版本。如果您之前已使用較舊的平台版本建立環境或升級舊版環境，您也可以使用此方法來還原至先前的平台版本，前提是要在相同平台分支中。

若要更新您環境的平台版本

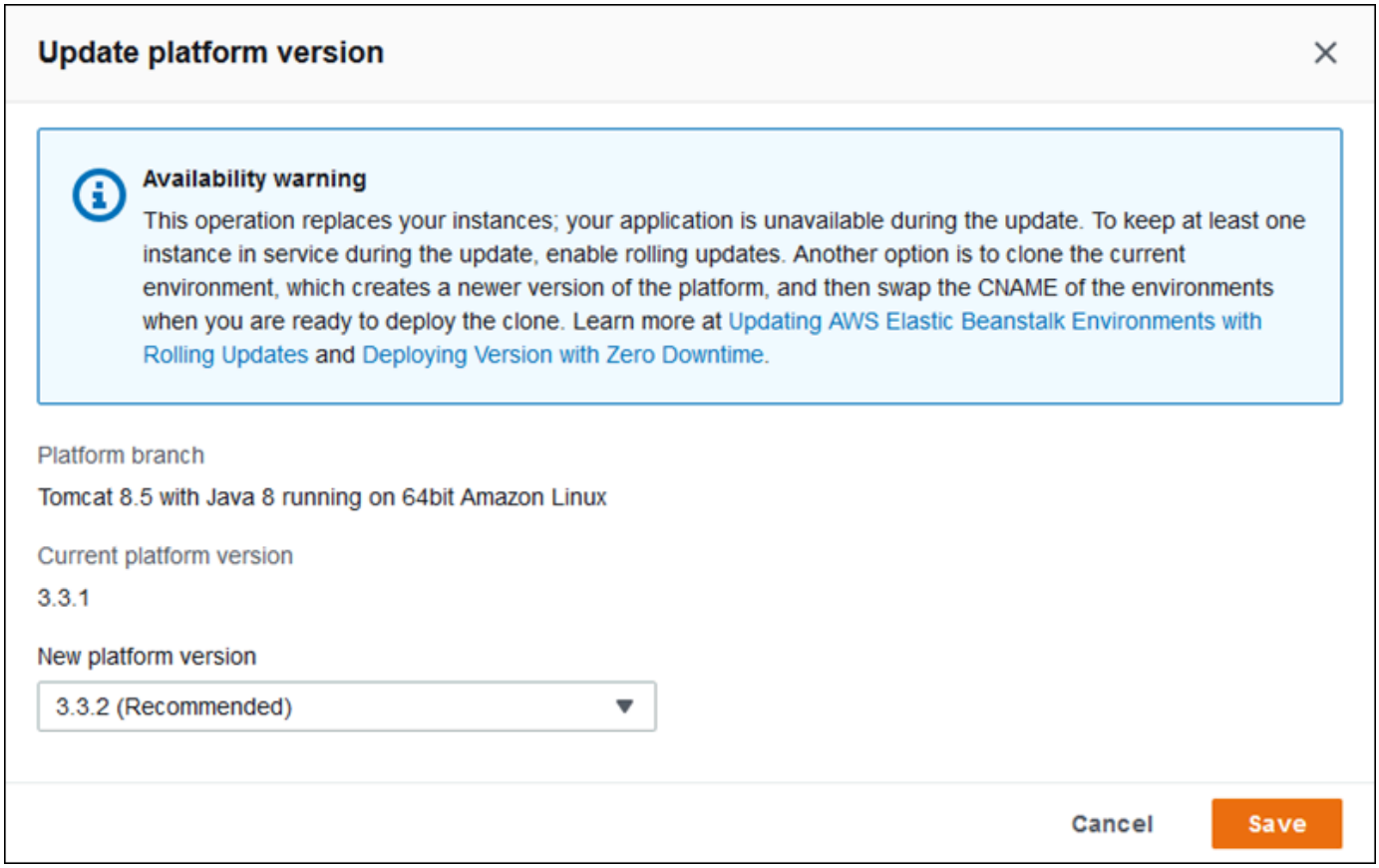
1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在環境概觀頁面的 Platform (平台) 下，選擇 Change (變更)。



4. 在 Update platform version (更新平台版本) 對話方塊中，選取平台版本。系統會自動選取分支中的最新 (建議) 平台版本。您可以更新至過去使用過的任何版本。



5. 選擇 Save (儲存)。

Elastic Beanstalk 可協助您進一步簡化平台更新。您可將環境設定為在每週的可設定維護時段期間，自動套用次要及修補程式版本更新。Elastic Beanstalk 套用受管更新不需要停機時間，也不會降低容量，若執行應用程式的執行個體在新版本未通過運作狀態檢查，將會立即取消更新。如需詳細資訊，請參閱受管平台更新。

方法 2 – 執行藍/綠部署

使用此方法以使用不同的執行時間、Web 伺服器、應用程式伺服器或作業系統，更新至不同的平台分支，或不同的主要平台版本。若您要運用新的執行時間功能或最新的 Elastic Beanstalk 功能，一般必須採用此方法。當您移出已棄用或已淘汰的平台分支時，這也是必需的。

若您遷移到其他主要平台版本或具有重大元件更新的平台版本，整個應用程式或其中一部分的功能，很可能不會如預期地在新的平台版本上正常運作，因此可能需要執行變更。

執行遷移之前，請將本機開發機器更新到新的執行時間版本，並更新您想要遷移的其他平台元件。確認應用程式仍可以如預期地正常運作，並視需要修改程式碼及執行其他變更。然後使用以下最佳實務步驟，將您的環境安全地遷移到新的平台版本。

將您的環境遷移到具有重大更新的平台版本

1. 建立新環境，使用新的目標平台版本，並將您的應用程式代碼部署到其中。新的環境應該在 Elastic Beanstalk 應用程式中，其中包含您正在遷移的環境。請勿終止現有的環境。
2. 使用新環境遷移應用程式。尤其是：

   - 尋找您在開發階段中無法偵測到的應用程式相容性問題，並加以修復。
   - 確保所有透過組態檔案執行的應用程式自訂項目可以在新環境中正常運作。自訂項目可能包括選項設定、其他外安裝套件、自訂安全原則，以及安裝在環境執行個體上的指令碼或組態檔案。
   - 如果您的應用程式使用自訂的 Amazon Machine Image (AMI)，請根據新平台版本的 AMI 建立新的自訂 AMI。如需進一步了解，請參閱使用自訂的 Amazon Machine Image (AMI)。如果您的應用程式使用具有自訂 AMI 的 Windows Server 平台，且您要遷移到 Windows Server V2 平台版本，您就必須執行這項操作。在這種情況中，請一併參閱從先前的 Windows Server 平台主要版本遷移。

   重複測試和部署您的修補程式，直到符合應用程式在新環境的需求
3. 透過將其 CNAME 與現有的生產環境的 CNAME 交換，將新環境轉換到您的生產環境。如需詳細資訊，請參閱透過 Elastic Beanstalk 進行藍/綠部署。
4. 當您滿意新環境的生產狀態，請終止舊環境。如需詳細資訊，請參閱 終止 Elastic Beanstalk 環境。

受管平台更新

AWS Elastic Beanstalk 會定期發佈平台更新，來提供修正程式，軟體更新和新功能。利用受管平台更新，您可以將環境設定為在排程的維護時段期間，自動升級至平台的最新版本。在更新程序進行期間，您的應用程式仍會繼續提供服務，功能不會減少。在單一執行個體環境和負載平衡環境中，皆可進行受管更新作業。

> **Note**
>
> 這個功能不適用於第 2 版 (v2) 以前的 Windows Server 平台版本。

您可以將環境設定為自動套用修補程式版本更新，或同時套用修補程式版本與次要版本的更新。受管平台更新不支援跨平台分支的更新 (不同主要平台元件版本的更新，例如作業系統、執行時間或 Elastic Beanstalk 元件)，因為這些更新可能會引進無法回溯相容的變更。

您也可以設定 Elastic Beanstalk 在維護時段期間更換您環境中的所有執行個體，即使無可用的平台更新亦然。如果您的應用程式在長時間執行後出現錯誤或記憶體的問題，則更換您環境中的所有執行個體會有所幫助。

在 2019 年 11 月 25 日或之後使用 Elastic Beanstalk 主控台建立的環境上，會盡可能根據預設啟用受管更新。受管更新需要啟用增強型運作狀態。當您選取其中一個組態預設時，系統會根據預設啟用增強型運作狀態，而當您選取 Custom configuration (自訂組態) 時，則會將其停用。若為不支援增強型運作狀態的舊版平台，或在增強型運作狀態停用時，主控台則無法啟用受管更新。當主控台為新環境啟用受管更新時，Weekly update window (每週更新時段) 會設為一週當中的某個隨機的天的某個隨機時段。Update level (更新層級) 已設為 Minor and patch (次要和修補程式)，且 Instance replacement (執行個體更換) 已停用。您可以在最終環境建立步驟前停用或重新設定受管更新。

針對現有環境，請隨時使用 Elastic Beanstalk 主控台來設定受管平台更新。

> **Important**
>
> 如果一個 AWS 帳戶中有大量 Beanstalk 環境，在受管更新期間可能會遭遇限流問題。大量是一個相對數量，具體取決於為環境排程的受管更新緊密程度。如果帳戶緊密排程了超過 200 個環境，就可能會導致限流問題，數量低於這個數值也可能會有問題。
> 為了平衡受管更新的資源負載，建議在帳戶中將環境的排程維護時段鋪開。
> 此外，可考慮使用多帳戶策略。如需詳細資訊，請參閱 AWS 白皮書與指南網站上的《使用多帳戶組織您的 AWS 環境》。

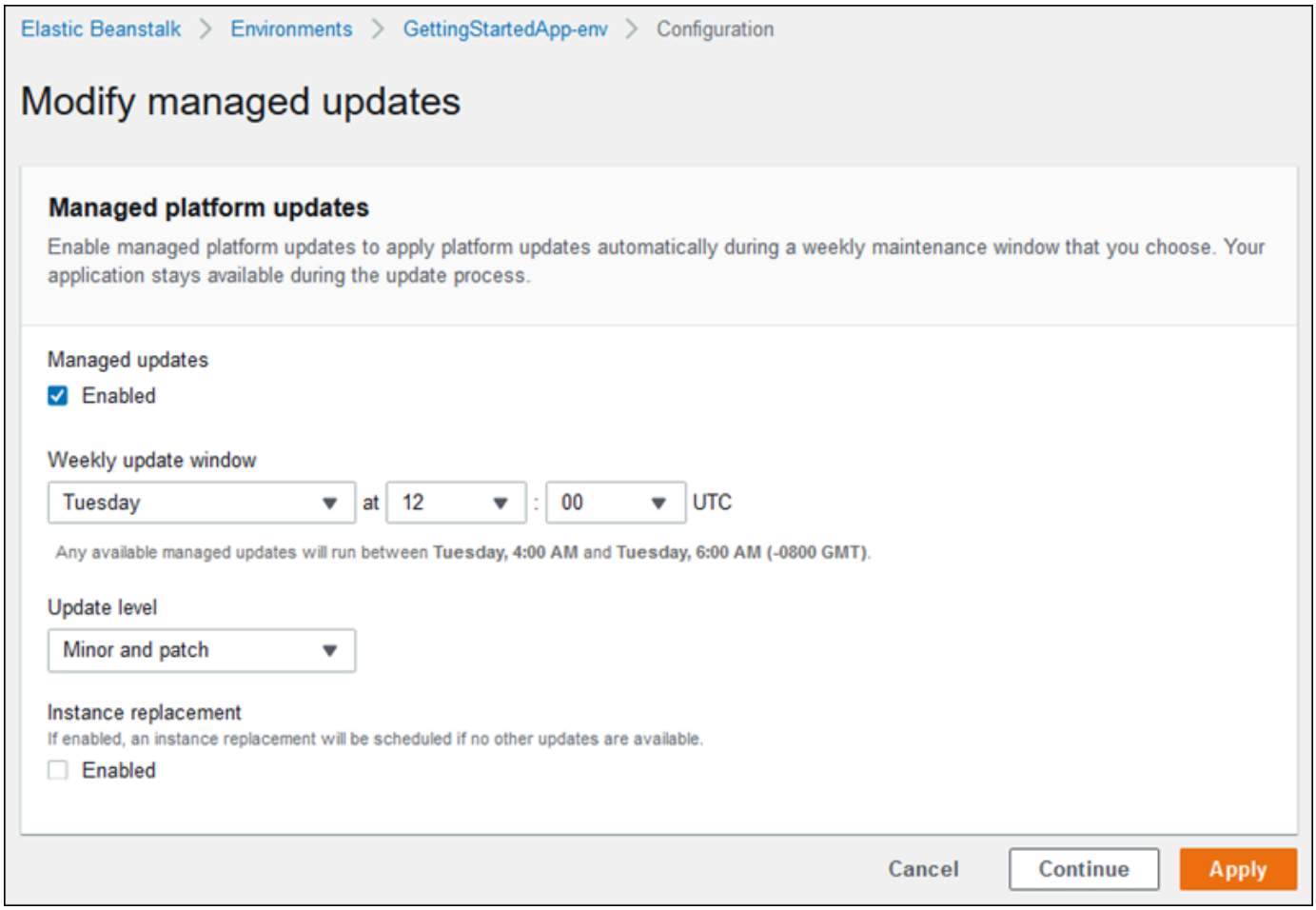
設定受管平台更新

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

   > **Note**
   >
   > 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇 Configuration (組態)。
4. 在 Managed updates (受管理的更新) 類別中，選擇 Edit (編輯)。
5. 停用或啟用 Managed updates (受管更新)。
6. 若已啟用受管更新，請選取維護時段，然後選取 Update level (更新層級)。

7. (選用) 選取 Instance replacement (執行個體更換) 來啟用每週的執行個體更換作業。



8. 若要儲存變更，請選擇頁面底部的儲存變更。

受管平台更新需依據增強的健全狀況報告，來判定您應用程式的運作狀態是否夠健全，以評斷平台更新作業是否成功。如需說明，請參閱 啟用 Elastic Beanstalk 增強型運作狀態報告。

章節

- 進行受管平台更新所需的許可
- 受管更新的維護時段
- 次要和修補程式版本更新
- 不可變的環境更新
- 管理受管更新作業
- 受管動作選項的命名空間

進行受管平台更新所需的許可

Elastic Beanstalk 需要許可來代表您起始平台更新作業。為取得這些許可，Elastic Beanstalk 會擔任受管更新服務角色。當您將預設服務角色用於環境時，Elastic Beanstalk 主控台也會使用其做為受管更新服務角色。主控台會將 AWSElasticBeanstalkManagedUpdatesCustomerRolePolicy 受管原則指派給您的服務角色。此原則具有 Elastic Beanstalk 執行受管平台更新所需的所有許可。

如需有關設定受管更新服務角色之其他方法的詳細資訊，請參閱the section called “服務角色”。

> **Note**
>
> 如果您使用組態檔案來擴展您的環境，以加入其他的資源，則您可能需要針對在環境的受管更新服務角色中新增許可。當您以名稱參考其他區段或檔案中的這些資源時，通常需要新增許可。

如果更新失敗，在受管更新頁面上會列出失敗的原因。

受管更新的維護時段

當 AWS 針對您環境的平台發佈新版本時，Elastic Beanstalk 會將受管平台更新作業排程在下一次的每週維護時段。維護時段的時間長度是兩個小時。Elastic Beanstalk 會在維護時段啟動排程更新。該更新可能會在時段結束後才完成。

> **Note**
>
> 在大多數情況下，Elastic Beanstalk 會將您的受管更新安排在未來的每週維護時段。系統在安排受管更新時，會將各種更新安全性和服務可用性納入考量。在極少數情況下，更新可能不會安排在第一個即將到來的維護時段中。如果發生這種情況，系統會再次嘗試安排在下一個維護時段中。若要手動套用受管更新，請選擇 Apply now (馬上套用)，如本頁之管理受管更新作業所述。

次要和修補程式版本更新

您可以讓受管平台更新只套用修補程式版本的更新，或同時套用次要版本與修補程式版本的更新。修補程式版本的更新提供錯誤修正與效能改善功能，也可以包含對執行個體上軟體、指令碼和組態選項的次要組態變更。次要版本更新提供對新 Elastic Beanstalk 功能的支援。您不能套用重大版本更新，此種更新可能會進行與受管平台更新回溯不相容的變更。

在平台版本編號中，第二個數字是次要更新的版本，第三個數字是修補程式的版本。例如，2.0.7 平台版本包含 0 的次要版本和 7 的修補程式版本。

不可變的環境更新

受管平台更新作業會進行不可變的環境更新，來將您的環境升級為新的平台版本。不可變的更新作業會在不移除任何服務執行個體或修改環境的狀況下，更新您的環境，再確認執行新版本的執行個體能否通過運作狀態檢查。

在不可變的更新作業中，Elastic Beanstalk 會使用新平台版本，來部署與目前執行中執行個體數量一樣多的執行個體。新的執行個體會和這些執行舊版本的執行個體，一起接收要求。如果新一組的執行個體通過所有的運作狀態檢查，則 Elastic Beanstalk 會終止舊的那組執行個體，只留下使用新版的執行個體。

即使是在維護時段以外的時間執行，受管平台更新作業也一律會進行不可變的更新。如果您是在 Dashboard (儀表板) 變更平台版本，則 Elastic Beanstalk 會套用您針對組態更新所選擇的更新原則。

> **Warning**
>
> 部署或更新期間，部分原則會取代所有執行個體。這會造成所有的累計 Amazon EC2 爆量餘額遺失。這發生的情況如下：
>
> - 執行個體更換啟用的受管平台更新
> - 不可變更新
> - 不可變更新或流量分割啟用的部署

管理受管更新作業

Elastic Beanstalk 主控台會在 Managed updates overview (受管更新概觀) 頁面上顯示有關受管更新的詳細資訊。

若要檢視關於受管更新的資訊 (主控台)

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 選擇 Managed updates (受管更新)。

Managed Updates Overview (受管更新概觀) 區段會提供已排程和待處理受管更新作業的相關資訊。History (歷程記錄) 區段會列出已成功的更新作業和失敗的嘗試。

您可以選擇立即執行排程更新作業，而非等到維護時段再進行。

若要立即執行受管平台更新作業 (主控台)

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 選擇 Managed updates (受管更新)。
4. 選擇 Apply now (立即套用)。
5. 確認更新詳細資訊，然後選擇 Apply (套用)。

當您在維護時段以外的時間執行受管平台更新作業時，Elastic Beanstalk 會進行不可變更新。如果您是在儀表板或使用不同的用戶端來更新環境的平台，則 Elastic Beanstalk 會使用您針對組態變更所選取的更新類型。

如果您尚未排程受管更新作業，則您的環境可能已在執行最新的版本。沒有更新作業排程的其他原因還包括：

- 有次要版本的可用更新，但您的環境已設定只自動套用修補程式版本的更新。
- 您的環境自從更新發佈之後尚未掃描過。Elastic Beanstalk 通常會每個小時檢查一次更新。
- 有更新作業等待處理中或已正在進行。

當您的維護時段開始或當您選擇 Apply now (立即套用) 時，排程的更新作業會先進入等待處理中狀態再執行。

受管動作選項的命名空間

您可以在 aws:elasticbeanstalk:managedactions 和 aws:elasticbeanstalk:managedactions:platformupdate 命名空間中使用組態選項，來啟用和設定受管平台更新作業。

ManagedActionsEnabled 選項會啟用受管平台更新作業。將此選項設定為 true 來啟用受管平台更新，並使用其他選項來設定更新動作。

利用 PreferredStartTime 來設定每週維護時段開始的時間 (格式為 *day*:*hour*:*minute*)。

將 UpdateLevel 設定為 minor 或 patch 來同時套用次要版本與修補程式版本的更新，或只套用修補程式版本的更新。

當受管平台更新作業啟用時，您可以藉由將 InstanceRefreshEnabled 選項設定為 true，來啟用執行個體的更換。當此設定啟用時，Elastic Beanstalk 會每週針對您的環境進行不可變的更新，無論是否有可用的新平台版本。

下列的範例組態檔案會在每週二上午 9:00 UTC 開始的維護時段中，啟用修補程式更新版的受管平台更新作業。

Example .ebextensions/managed-platform-update.config

```
option_settings:
  aws:elasticbeanstalk:managedactions:
    ManagedActionsEnabled: true
    PreferredStartTime: "Tue:09:00"
  aws:elasticbeanstalk:managedactions:platformupdate:
    UpdateLevel: patch
    InstanceRefreshEnabled: true
```

自舊版平台遷移您的應用程式

若您已部署使用舊版平台的 Elastic Beanstalk 應用程式，應透過非舊版平台，將應用程式遷移至新環境，藉此取得新的功能。若您不確定是否正使用舊版平台來執行應用程式，您可於 Elastic Beanstalk 主控台檢查。如需說明，請參閱「檢視您使用的是否為舊版平台」。

舊版平台缺少哪些新功能？

舊版平台不支援下列功能：

- 組態檔案 (如 使用組態檔案 (.ebextensions) 來進行進階的環境自訂 主題所述)
- ELB 運作狀態檢查 (如 基礎型運作狀態報告 主題所述)
- 執行個體描述檔 (如 管理 Elastic Beanstalk 執行個體描述檔 主題所述)
- VPC (如 搭配 Amazon VPC 使用 Elastic Beanstalk 主題所述)
- 資料層 (如 將資料庫新增至您的 Elastic Beanstalk 環境 主題所述)
- 工作者層 (如 工作者環境 主題所述)
- 單一執行個體環境 (如 環境類型 主題所述)
- 標籤 (如 在您的 Elastic Beanstalk 環境中標記資源 主題所述)
- 滾動更新 (如 Elastic Beanstalk 滾動環境資訊更新 主題所述)

為何部分平台版本標記為舊版？

部分較舊的平台版本不支援最新的 Elastic Beanstalk 功能，這些版本在 Elastic Beanstalk 主控台中的環境概觀頁面上會標記為 (legacy) ((傳統))。

檢視您使用的是否為舊版平台

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

   > **Note**
   >
   > 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在環境概觀頁面上，檢視 Platform (平台) 名稱。

   若平台名稱旁出現了 (legacy) ((舊版))，則表示您的應用程式使用的是舊版平台。

欲遷移您的應用程式

1. 將您的應用程式部署至新環境。如需取得說明，請前往 建立 Elastic Beanstalk 環境。
2. 若您具備 Amazon RDS 資料庫執行個體，請更新您的資料庫安全群組，以允許您新環境的 EC2 安全群組存取。關於使用 AWS 管理主控台來找出 EC2 安全群組名稱的方法，詳細指示請參閱 安

全群組。如需設定 EC2 安全群組的詳細資訊，請參閱《Amazon Relational Database Service 使用者指南》中使用資料庫安全群組的「授權網路存取 Amazon EC2 安全群組」章節。

3. 交換環境的 URL。如需取得說明，請前往 透過 Elastic Beanstalk 進行藍/綠部署。
4. 終止您的舊環境。如需取得說明，請前往 終止 Elastic Beanstalk 環境。

> **Note**
>
> 若您使用 AWS Identity and Access Management (IAM)，將需要更新您的原則，在其中納入 AWS CloudFormation 和 Amazon RDS (如適用)。如需更多詳細資訊，請參閱 使用 Elastic Beanstalk 搭配 AWS Identity and Access Management。

將您的 Elastic Beanstalk Linux 應用程式遷移到 Amazon Linux 2023 或 Amazon Linux 2

本節說明如何使用下列其中一個遷移路徑來遷移應用程式。

- 從 Amazon Linux 2 平台分支遷移到 Amazon Linux 2023 平台分支。
- 從 Amazon Linux AMI(AL1) 平台分支遷移到 Amazon Linux 2023 (建議) 或 Amazon Linux 2 平台分支。

主題

- 從 Amazon Linux 2 遷移到 Amazon Linux 2023
- 從 Amazon Linux AMI (AL1) 遷移到 AL2 或 AL2023

從 Amazon Linux 2 遷移到 Amazon Linux 2023

本主題的內容用於指導您將應用程式從 Amazon Linux 2 平台分支遷移到 Amazon Linux 2023 平台分支。

差異和相容性

在 Elastic Beanstalk AL2 與 AL2023 平台之間

Elastic Beanstalk Amazon Linux 2 和 Amazon Linux 2023 平台之間具有高度相容性。雖然有部分差異需要注意：

- 執行個體中繼資料服務版本 1 (IMDSv1) – AL2023 平台上的 DisableIMDSv1 選項設定預設為 `true`。AL2 平台上的預設值為 `false`。
- pkg-repo 執行個體工具 — pkg-repo 工具不適用於在 AL2023 平台上執行的環境。但您可以手動將套件和作業系統更新套用至 AL2023 執行個體。如需詳細資訊，請參閱《Amazon Linux 2023 使用者指南》中的管理套件和作業系統更新。
- Apache HTTPd 組態 — AL2023 平台的 Apache `httpd.conf` 檔案具有部分與 AL2 不同的組態設定：
  - 在預設情況下，拒絕存取伺服器的整個檔案系統。Apache 網站安全性提示頁面上的依預設保護伺服器檔案，提供了對此類設定的說明。
  - 避免使用者覆寫您已設定的安全性功能。除特別啟用的目錄外，組態會拒絕存取所有目錄中的 `.htaccess` 設定。Apache 網站安全性提示頁面上的保護系統設定，提供了對此設定的說明。Apache HTTP 伺服器教學課程：.htaccess 檔案頁面指出，這項設定可能有助於改善效能。
  - 拒絕存取具有名稱模式 `.ht*` 的檔案。此設定可防止 Web 用戶端檢視 `.htaccess` 和 `.htpasswd` 檔案。

您可以針對您的環境變更上述任何組態設定。如需詳細資訊，請參閱擴充 Elastic Beanstalk Linux 平台。展開反向代理主題，以查看設定 Apache HTTPD 章節。

在 Amazon Linux 作業系統之間

如需有關 Amazon Linux 2 與 Amazon Linux 2023 平台之間差異的詳細資訊，請參閱 Amazon Linux 2023 User Guide 中的 Comparing Amazon Linux 2 and Amazon Linux 2023。

如需有關 Amazon Linux 2023 的詳細資訊，請參閱 Amazon Linux 2023 User Guide 中的 What is Amazon Linux 2023?。

一般遷移程序

如果準備用於生產環境，Elastic Beanstalk 需要進行藍/綠部署才能執行升級。使用藍/綠部署程序進行遷移時，我們建議遵循以下一般最佳實務步驟。

準備進行遷移測試

在部署應用程式並開始測試之前，請先檢閱上一節 差異和相容性 中的資訊。另請參閱 Amazon Linux 2023 User Guide 中的 Comparing Amazon Linux 2 and Amazon Linux 2023。記下此內容中適用或可能適用您的應用程式和組態設定的特定資訊。

高階遷移步驟

1. 建立以 AL2023 平台分支為基礎的新環境。
2. 將應用程式部署至目標 AL 2023 環境。

   您現有的生產環境將保持作用中狀態且不受影響，同時您可以測試和調整新環境，進行反覆。
3. 在新環境中全面測試應用程式。
4. 目標 AL2023 環境準備好用於生產後，交換兩個環境的 CNAME 以將流量重新導向至新 AL2023 環境。

更詳細的遷移步驟和最佳實務

如需更詳細的藍/綠部署程序，請參閱 透過 Elastic Beanstalk 進行藍/綠部署。

如需更具體的指南和詳細的最佳實務步驟，請參閱藍/綠方法。

更多有助於規劃遷移的參考資料

下列參考資料可提供有關遷移規劃的其他資訊。

- AWS Elastic Beanstalk 平台中 Elastic Beanstalk 支援的平台
- 淘汰的平台分支歷史記錄
- the section called “Linux 平台”
- 平台淘汰常見問答集

從 Amazon Linux AMI (AL1) 遷移到 AL2 或 AL2023

如果您的 Elastic Beanstalk 應用程式是以 Amazon Linux AMI 平台分支為基礎，請參閱本節了解如何將應用程式的環境遷移至 Amazon Linux 2 或 Amazon Linux 2023。以 Amazon Linux AMI 為基礎的上一代平台分支現在已淘汰。

我們強烈建議您遷移到 Amazon Linux 2023，因為它比 Amazon Linux 2 更新。Amazon Linux 2 作業系統將在 Amazon Linux 2023 之前終止支援，因此，如果您遷移到 Amazon Linux 2023，可以獲得更長的支援時間。

值得注意的是，Elastic Beanstalk Amazon Linux 2 和 Amazon Linux 2023 平台之間具有高度相容性。雖然有些方面確實有所不同：執行個體中繼資料服務第 1 版 (IMDSv1) 選項預設值、對 pkg-repo 執行個體工具的支援，以及部分 Apache HTTPD 組態。如需詳細資訊，請參閱 Amazon Linux 2023

差異和相容性

以 AL2023/AL2 為基礎的平台分支並不保證能與您現有的應用程式回溯相容。同樣需要注意的是，即使應用程式的程式碼成功部署到新平台版本，仍可能會因作業系統和執行時間差異而有不同的運作或執行方式。

雖然 Amazon Linux AMI 和 AL2023/AL2 具有相同的 Linux 核心，它們在以下方面仍有所不同：初始化系統、`libc` 版本、編譯器工具鏈及各種套件。如需詳細資訊，請參閱 Amazon Linux 2 常見問答集。

Elastic Beanstalk 服務也已更新執行時間的平台專用版本、建置工具及其他依存項目。

因此，建議您慢慢地在開發環境中徹底測試應用程式，然後進行任何必要的調整。

一般遷移程序

如果準備用於生產環境，Elastic Beanstalk 需要進行藍/綠部署才能執行升級。使用藍/綠部署程序進行遷移時，我們建議遵循以下一般最佳實務步驟。

準備進行遷移測試

在您部署應用程式並開始測試之前，請檢閱查看本主題後文 所有 Linux 平台的考量事項 中的資訊。另外，請參閱後文 平台特定考量事項 章節中適用於您平台的資訊。記下此內容中適用或可能適用您的應用程式和組態設定的特定資訊。

高階遷移步驟

1. 建立以 AL2 或 AL2023 平台分支為基礎的新環境。我們建議您遷移至 AL2023 平台分支。
2. 將應用程式部署至目標 AL2023/AL2 環境。

   您現有的生產環境將保持作用中狀態且不受影響，同時您可以測試和調整新環境，進行反覆。
3. 在新環境中全面測試應用程式。
4. 目標 AL2023/AL2 環境準備好用於生產後，交換兩個環境的 CNAME 以將流量重新導向至新環境。

更詳細的遷移步驟和最佳實務

如需更詳細的藍/綠部署程序，請參閱 透過 Elastic Beanstalk 進行藍/綠部署。

如需更具體的指南和詳細的最佳實務步驟，請參閱藍/綠方法。

更多有助於規劃遷移的參考資料

下列參考資料可提供有關遷移規劃的其他資訊。

- Amazon Linux 2023 User Guide 中的 Comparing Amazon Linux 2 and Amazon Linux 2023。
- Amazon Linux 2023 User Guide 中的 What is Amazon Linux 2023?
- AWS Elastic Beanstalk 平台中 Elastic Beanstalk 支援的平台
- 淘汰的平台分支歷史記錄
- the section called “Linux 平台”
- 平台淘汰常見問答集

所有 Linux 平台的考量事項

下表說明了在規劃將應用程式遷移至 AL2023/AL2 時應注意的考量事項。無論使用何種特定程式設計語言或應用程式伺服器，這些考量事項皆適用於任何 Elastic Beanstalk Linux 平台。

方面	變更和資訊
組態檔案	在 AL2023/AL2 平台上，您可以如往常使用組態檔案，且所有區段的運作方式都一樣。不過，特定設定的運作方式可能與舊版 Amazon Linux AMI 平台上的運作方式不同。例如： • 您使用組態檔案安裝的某些軟體套件可能無法在 AL2023/AL2 上使用，或其名稱可能已經變更。 • 某些平台專用組態選項已從其平台專用命名空間移至不同的平台無關命名空間。 • `.ebextensions/nginx` 目錄中提供的代理組態檔案應移至 `.platform/nginx` 平台勾點目錄。如需詳細資訊，請展開the section called “擴充 Linux 平台”中的反向代理組態一節。 建議使用平台勾點，以在環境執行個體上執行自訂程式碼。您仍可在 `.ebextensions` 組態檔案中使用命令和容器命令，但它們並不容易使用。例如，在 YAML 檔案內寫入命令指令碼可能很麻煩，且難以測試。 您仍須將 `.ebextensions` 組態檔案用於需要參考 AWS CloudFormation 資源的任何指令碼。

方面	變更和資訊
平台勾點	AL2 平台將可執行的檔案新增到環境執行個體上的勾點目錄，提供了擴展環境平台的新方式。在舊版 Linux 平台中，您可能已使用自訂平台勾點。這些勾點不是專為受管平台設計且不受支援，但在某些情況下能以實用的方式運作。在 AL2023/AL2 平台版本中，自訂平台勾點無法運作。您應將所有勾點遷移至新的平台勾點。如需詳細資訊，請展開the section called “擴充 Linux 平台”中的平台勾點一節。
支援的代理伺服器	AL2023/AL2 平台版本支援與 Amazon Linux AMI 平台版本支援的每個平台相同的反向代理伺服器。全部 AL2023/AL2 平台版本均使用 nginx 作為預設反向代理伺服器，ECS 和 Docker 平台除外。Tomcat、Node.js、PHP 和 Python 平台也支援 Apache HTTPD 作為替代方案。所有平台都會以統一的方式啟用代理伺服器設定，如本節所述。但是，設定代理伺服器與 Amazon Linux AMI 上的伺服器略有不同。以下是所有平台的差異： • 預設為 nginx – 所有 AL2023/AL2 平台版本上的預設代理伺服器都是 nginx。在 Tomcat、PHP 和 Python 的 Amazon Linux AMI 平台版本，預設的代理伺服器是 Apache HTTPD。 • 一致的命名空間 – 所有 AL2023/AL2 平台版本都使用 `aws:elasticbeanstalk:environment:proxy` 命名空間來設定代理伺服器。在 Amazon Linux AMI 平台版本中，這是每個平台的決定，Node.js 使用了不同的命名空間。 • 組態檔案位置 – 所有 AL2023/AL2 平台版本的代理組態檔案都應放置在 `.platform/nginx` 和 `.platform/httpd` 目錄中。在 Amazon Linux AMI 平台版本上，這些位置分別是 `.ebextensions/nginx` 和 `.ebextensions/httpd` 。 若要平台特定的代理組態變更的詳細資訊，請參閱the section called “平台特定考量事項”。如需有關 AL2023/AL2 平台上代理組態的詳細資訊，請展開 the section called “擴充 Linux 平台” 中的反向代理組態一節。
代理組態變更	除了特定於每個平台的代理組態變更外，還有一致地套用至所有平台的代理組態變更。參考兩者以準確設定您的環境，這一點非常重要。 • 所有平台 – 展開 the section called “擴充 Linux 平台” 中的反向代理組態一節。 • 平台特定 – 請參閱 the section called “平台特定考量事項”。

方面	變更和資訊
執行個體描述檔	AL2023/AL2 平台需要執行個體設定檔才能進行設定。若無此描述檔，環境建立可能會暫時成功，但在建立完成後，當需要執行個體描述檔的動作開始失敗時，環境可能很快會出現錯誤。如需詳細資訊，請參閱the section called “執行個體設定檔”。
增強型運作狀態	AL2023/AL2 平台版本會預設啟用增強型運作狀態。如果您不使用 Elastic Beanstalk 主控台來建立環境，這就會是一項變更。無論使用的平台版本為何，主控台都會根據預設盡可能啟用增強型運作狀態。如需詳細資訊，請參閱the section called “增強型運作狀態報告與監控”。
自訂 AMI	如果環境使用自訂 AMI，請透過 Elastic Beanstalk AL2023/AL2 平台為新環境建立以 AL2023/AL2 為基礎的新 AMI。
自訂平台	AL2023/AL2 平台版本的受管 AMI 不支援自訂平台。

平台特定考量事項

本節討論特定 Elastic Beanstalk Linux 平台的特定遷移考量事項。

Docker

以 Amazon Linux AMI (AL1) 為基礎的 Docker 平台分支系列包括三個平台分支。我們會為每個平台分支提供不同的遷移路徑建議。

AL1 平台分支	遷移至 AL2023/AL2 的路徑
由在 Amazon Linux AMI (AL1) 上執行的 Amazon ECS 管理的多容器 Docker	以 ECS 為基礎的 Docker AL2023/AL2 平台分支 以 ECS 為基礎的 Docker AL2023/AL2 平台分支為執行於多容器 Docker AL1 平台分支的環境提供了直接的遷移路徑。 • 類似於舊版多容器 Docker AL1 分支，AL2023/AL2 平台分支使用 Amazon ECS 來協調將多個 Docker 容器部署至 Elastic Beanstalk 環境內的 Amazon ECS 叢集。 • AL2023/AL2 平台分支支援舊版多容器 Docker AL1 平台分支中的所有功能。 • AL2023/AL2 平台分支也支援相同的 `Dockerrun.aws.json` v2 檔案。

AL1 平台分支	遷移至 AL2023/AL2 的路徑
	如需詳細信息，了解如何將執行於多容器 Docker Amazon Linux 平台分支的應用程式遷移至執行於 AL2023/AL2 的 Amazon ECS 平台分支，請參閱 ???。

<table>
<tr><th>AL1 平台分支</th><th>遷移至 AL2023/AL2 的路徑</th></tr>
<tr><td>在 Amazon Linux AMI (AL1) 上執行的 Docker

執行 Amazon Linux AMI (AL1) 的預先設定的 Docker (Glassfish 5.0)</td><td>
執行於 AL2023/AL2 平台分支的 Docker

我們建議您將在基於預先設定的 Docker (Glassfish 5.0) 或執行於 Amazon Linux AMI (AL1) 的 Docker 的環境中執行的應用程式遷移至基於執行於 Amazon Linux 2 的 Docker 或執行於 AL2023 的 Docker 平台分支的環境。

如果您的環境基於預先設定的 Docker (Glassfish 5.0)平台分支，請參閱 the section called “教程-碼 GlassFish 頭工人：通往 Amazon Linux 2023 的路徑”。

下表列出了執行於 AL2023/AL2 的 Docker 平台分支的遷移資訊。
<table>
<tr><th>方面</th><th>變更和資訊</th></tr>
<tr><td>儲存空間</td><td>Elastic Beanstalk 將 Docker 設定為使用儲存體驅動程式來儲存 Docker 影像和容器資料。在 Amazon Linux AMI，Elastic Beanstalk 使用 裝置映射器儲存驅動程式。為了提高效能，Elastic Beanstalk 佈建了一個額外的 Amazon EBS 磁碟區。在 AL2023/AL2 Docker 平台版本上，Elastic Beanstalk 使用 OverlayFS 儲存驅動程式，以達到更佳的效能，並且不再需要單獨的磁碟區。

使用 Amazon Linux AMI 時，如果您使用 <code>BlockDeviceMappings</code> 命名空間的 <code>aws:autoscaling:launchconfiguration</code> 選項將自訂儲存磁碟區新增至 Docker 環境，我們建議您同時新增 Elastic Beanstalk 佈建的 <code>/dev/xvdcz</code> Amazon EBS 磁碟區。Elastic Beanstalk 不再佈建此磁碟區，因此您應該將其從組態檔案中刪除。如需詳細資訊，請參閱the section called “Amazon Linux AMI (Amazon Linux 2 之前的版本) 上的 Docker 組態”。</td></tr>
<tr><td>私有儲存庫身分驗證</td><td>當您提供 Docker 產生的身分驗證檔案以連線至私有儲存庫時，您不再需要將其轉換為 Amazon Linux AMI Docker 平台版本所需的舊格式。AL2023/AL2 Docker 平台版本支援新格式。如需詳細資訊，請參閱the section called “使用私有儲存庫中的映像”。</td></tr>
<tr><td>代理伺服器</td><td>AL2023/AL2 Docker 平台版本不支援不在代理伺服器後面執行的獨立容器。在 Amazon Linux AMI Docker 平台版本中，這曾經可以透過</td></tr>
</table>
</td></tr>
</table>

AL1 平台分支	遷移至 AL2023/AL2 的路徑
	方面 變更和資訊
	`aws:elasticbeanstalk:environment:proxy` 命名空間中的 `ProxyServer` 選項的 none 值來實現。

Go

下表列出了 Go 平台中 AL2023/AL2 平台版本的遷移資訊。

方面	變更和資訊
連接埠傳遞	在 AL2023/AL2 平台上，Elastic Beanstalk 不會透過 PORT 環境變數將連接埠值傳遞到應用程式程序。您可以自行設定 PORT 環境屬性，藉此模擬此程序的運作方式。不過，如果您有多個程序，並將 Elastic Beanstalk 傳遞增量連接埠值計算至程序 (5000、5100、5200 等)，則應修改您的實作。如需詳細資訊，請展開the section called “擴充 Linux 平台”中的反向代理組態一節。

Amazon Corretto

下表列出 Java SE 平台中 Corretto 平台分支的遷移資訊。

方面	變更和資訊
Corretto 相較於 OpenJDK	為實作 Java 平台標準版本 (Java SE)，AL2023/AL2 平台分支會使用 Amazon Corretto，也就是 Open Java Development Kit (OpenJDK) 的 AWS 發行版本。先前的 Elastic Beanstalk Java SE 平台分支會使用 Amazon Linux AMI 隨附的 OpenJDK 套件。
建置工具	AL2023/AL2 平台有新版建置工具：`gradle`、`maven` 和 `ant`。
JAR 檔案處理	在 AL2023/AL2 平台上，如果來源套件 (ZIP 檔案) 包含單一 JAR 檔案且不包含任何其他檔案，Elastic Beanstalk 不會再將 JAR 檔案重新命名為 `application.jar` 。唯有在單獨提交 JAR 檔案本身 (而不是在 ZIP 檔案內) 時，才會進行重新命名。

方面	變更和資訊
連接埠傳遞	在 AL2023/AL2 平台上，Elastic Beanstalk 不會透過 PORT 環境變數將連接埠值傳遞到應用程式程序。您可以自行設定 PORT 環境屬性，藉此模擬此程序的運作方式。不過，如果您有多個程序，並將 Elastic Beanstalk 傳遞增量連接埠值計算至程序 (5000、5100、5200 等)，則應修改您的實作。如需詳細資訊，請展開the section called “擴充 Linux 平台”中的反向代理組態一節。
Java 7	Elastic Beanstalk 不支援 AL2023/AL2 Java 7 平台分支。如果您有 Java 7 應用程式，請將其遷移至 Corretto 8 或 Corretto 11。

Tomcat

下表列出了 Tomcat 平台中 AL2023/AL2 平台版本的遷移資訊。

方面	變更和資訊
組態選項	在 AL2023/AL2 平台版本上，Elastic Beanstalk 僅支援 `aws:elasticbeanstalk:environment:proxy` 命名空間中組態選項和選項值的子集。以下是每個選項的遷移資訊。

選項	遷移資訊
`GzipCompression`	在 AL2023/AL2 平台版本上不受支援。
`ProxyServer`	AL2023/AL2 Tomcat 平台版本同時支援 nginx 和 Apache HTTPD 2.4 版代理伺服器。不過，不支援 Apache 版本 2.2。 在 Amazon Linux AMI 平台版本中，預設代理是 Apache 2.4。如果使用預設代理伺服器設定並新增自訂代理伺服器組態檔案，代理伺服器組態應該仍然可以使用 AL2023/AL2。但是，如果您使用了 `apache/2.2` 選項值，您現在必須將代理組態移轉到 Apache 2.4 版。

方面	變更和資訊
	`aws:elasticbeanstalk:container:tomcat:jvmoptions` 命名空間中的 `XX:MaxPermSize` 選項不支援 AL2023/AL2 平台版本。修改永久生成大小的 JVM 設定僅適用於 Java 7 和更早版本，因此不適用於 AL2023/AL2 平台版本。
應用程式路徑	在 AL2023/AL2 平台上，環境中 Amazon EC2 執行個體上應用程式目錄的路徑為 `/var/app/current` 。在 Amazon Linux AMI 平台，路徑是 `/var/lib/tomcat8/webapps` 。

Node.js

下表列出了 Node.js 平台中 AL2023/AL2 平台版本的遷移資訊。

方面	變更和資訊
已安裝的 Node.js 版本	在 AL2023/AL2 平台上，Elastic Beanstalk 維護了多個 Node.js 平台分支，並且只在每個平台版本上，安裝與平台分支對應的最新版本 Node.js 主要版本。例如，Node.js 12 平台分支中的每個平台版本，預設只會安裝 Node.js 12.x.y。在 Amazon Linux AMI 平台版本上，我們在每個平台版本上安裝了多個 Node.js 版本的多個版本，並且只維護一個平台分支。 選擇與您應用程式所需的 Node.js 主要版本對應的 Node.js 平台分支。
Apache HTTPD 日誌檔案名稱	在 AL2023/AL2 平台上，如果您使用 Apache HTTPD 代理伺服器，HTTPD 日誌檔案名稱會為 `access_log` 與 `error_log` ，這與支援 Apache HTTPD 的所有其他平台一致。在 Amazon Linux AMI 平台版本中，這些日誌檔案分別命名為 `access.log` 和 `error.log` 。 如需所有平台之日誌檔案名稱和位置的詳細資訊，請參閱 the section called “Elastic Beanstalk 如何設定 CloudWatch Logs”。
組態選項	在 AL2023/AL2 平台上，Elastic Beanstalk 不支援 `aws:elasticbeanstalk:container:nodejs` 命名空間中的組態選項。部分選項擁有替代方案。以下是每個選項的遷移資訊。

方面	變更和資訊

選項	遷移資訊
NodeCommand	在 package.json 檔案中使用 Procfile 或 scripts 關鍵字來指定啟動指令碼。
NodeVersion	在 package.json 檔案中使用 engines 關鍵字來指定 Node.js 版本。請注意，您只能指定與您平台分支對應的 Node.js 版本。例如，如果您使用的是 Node.js 12 平台分支，則只能指定 12.x.y Node.js 版本。如需詳細資訊，請參閱the section called “使用 package.json 檔案指定 Node.js 相依性”。
GzipCompression	在 AL2023/AL2 平台版本上不受支援。

ProxyServer	在 AL2023/AL2 Node.js 平台版本上，此選項會移至 aws:elasticbeanstalk:environment:proxy 命名空間。您可以在 nginx (預設值) 和 apache 之間進行選擇。
	
	AL2023/AL2 Node.js 平台版本不支援不在代理伺服器後執行的獨立應用程式。在 Amazon Linux AMI Node.js 平台版本中，這曾經可以透過 aws:elasticbeanstalk:container:nodejs 命名空間中的 ProxyServer 選項的 none 值來實現。如果您的環境執行獨立應用程式，請更新您的程式碼以接聽代理伺服器 (nginx 或 Apache) 轉送流量的連接埠。
	
	`var port = process.env.PORT || 5000;`
	
	`app.listen(port, function() {`
	`  console.log('Server running at http://127.0.0.1:%s',`
	` port);`
	`});`

PHP

下表列出了 PHP 平台中 AL2023/AL2 平台版本的遷移資訊。

方面	變更和資訊
PHP 檔案處理	在 AL2023/AL2 平台上，PHP 檔案係使用 PHP-FPM (CGI 程序管理器) 進行處理。在 Amazon Linux AMI 平台上，我們使用 mod_php (一種 Apache 模組)。
代理伺服器	AL2023/AL2 PHP 平台版本同時支援 nginx 和 Apache HTTPD 代理伺服器。預設為 nginx。 Amazon Linux AMI PHP 平台版本只支援 Apache HTTPD。如果新增自訂 Apache 組態檔案，可以將 `aws:elasticbeanstalk:environment:proxy` 命名空間中的 `ProxyServer` 選項設為 `apache`。

Python

下表列出了 Python 平台中 AL2023/AL2 平台版本的遷移資訊。

方面	變更和資訊
WSGI 伺服器	在 AL2023/AL2 平台上，Gunicorn 是預設 WSGI 伺服器。根據預設，Gunicorn 會在連接埠 8000 上接聽。此連接埠可能與應用程式在 Amazon Linux AMI 平台上使用的不同。如果您正在設定 `aws:elasticbeanstalk:container:python` 命名空間的 `WSGIPath` 選項，請用 Gunicorn 的語法替換該數值。如需詳細資訊，請參閱the section called “Python 組態命名空間”。 或者，您也可以使用 `Procfile` 來指定和設定 WSGI 伺服器。如需詳細資訊，請參閱the section called “Procfile”。
應用程式路徑	在 AL2023/AL2 平台上，環境中 Amazon EC2 執行個體上應用程式目錄的路徑為 `/var/app/current` 。在 Amazon Linux AMI 平台，路徑是 `/opt/python/current/app` 。
代理伺服器	AL2023/AL2 Python 平台版本同時支援 nginx 和 Apache HTTPD 代理伺服器。預設為 nginx。 Amazon Linux AMI Python 平台版本只支援 Apache HTTPD。如果新增自訂 Apache 組態檔案，可以將 `aws:elasticbeanstalk:environment:proxy` 命名空間中的 `ProxyServer` 選項設為 `apache`。

Ruby

下表列出了 Ruby 平台中 AL2023/AL2 平台版本的遷移資訊。

方面	變更和資訊
已安裝的 Ruby 版本	在 AL2023/AL2 平台上，Elastic Beanstalk 只會在每個平台版本上安裝與平台分支對應的最新單一 Ruby 版本。例如，Ruby 2.6 平台分支中的每個平台版本只安裝 Ruby 2.6.x。在 Amazon Linux AMI 平台版本上，我們安裝了多個 Ruby 版本的最新版本，例如 2.4.x、2.5.x 和 2.6.x。 如果您應用程式使用的 Ruby 版本無法對應您所使用的平台分支，我們建議切換至適用於您應用程式的正確 Ruby 版本平台分支。
應用程式伺服器	在 AL2023/AL2 平台上，Elastic Beanstalk 只會在所有 Ruby 平台版本上安裝 Puma 應用程式伺服器。您可以使用 `Procfile` 來啟動不同的應用程式伺服器，並使用 `Gemfile` 來進行安裝。 在 Amazon Linux AMI 平台上，我們支援每個 Ruby 版本兩種平台分支 — 一種支援 Puma 應用程式伺服器，另一種支援 Passenger 應用程式伺服器。如果您的應用程式使用 Passenger，您可以設定您的 Ruby 環境來安裝並使用 Passenger。 如需詳細資訊和範例，請參閱 the section called “Ruby 平台”。

平台淘汰常見問答集

> **Note**
>
> 2022 年 7 月 18 日，Elastic Beanstalk 淘汰所有以 Amazon Linux AMI (AL1) 為基礎的平台分支。

此常見問答集中的答案參考了以下主題：

- Elastic Beanstalk 平台支援政策
- 淘汰的平台分支歷史記錄
- AWS Elastic Beanstalk 平台中由 Elastic Beanstalk 支援的平台
- 將您的 Elastic Beanstalk Linux 應用程式遷移到 Amazon Linux 2023 或 Amazon Linux 2

- Amazon Linux 2 常見問答集。

1. 淘汰平台分支意味著什麼？

在宣佈的平台分支淘汰日期之後，您將無法再根據淘汰的平台分支建立新環境，除非您已具有基於該平台分支的作用中環境。如需詳細資訊，請參閱常見問答集 #11。Elastic Beanstalk 不會為這些平台分支提供新的維護更新。不建議在生產環境中使用淘汰的平台分支。如需詳細資訊，請參閱常見問答集 #5。

2. 為什麼要 AWS 淘汰基於 AL1 的平台分支機構？

當平台元件被廠商棄用或淘汰時，Elastic Beanstalk 會淘汰平台分支。在這種情況下，截至 2020 年 12 月 31 日，Amazon Linux AMI (AL1) 已結束標準支援。在 2022 年結束之前 Elastic Beanstalk 仍然繼續提供以 AL1 為基礎的平台，但我們已發佈具有最新功能的以 AL2 和 AL2023 為基礎的平台。如果客戶要繼續取得未來發佈的最新安全性和功能，遷移至以 AL2 或 AL2023 為基礎的平台至關重要。

3. 淘汰哪些平台分支？

如需已淘汰的平台元件和平台分支清單，請參閱 淘汰的平台分支歷史記錄。

4. 目前支援哪些平台？

請參閱 AWS Elastic Beanstalk 平台中由 Elastic Beanstalk 支援的平台。

5. 淘汰後，Elastic Beanstalk 是否會移除或終止我環境中的任何元件？

我們的淘汰平台分支原則不會移除對環境的存取，也不會刪除資源。但是，基於已淘汰平台分支的環境最終可能會出現不可預測的情況，因為 Elastic Beanstalk 由於供應商將已淘汰平台分支的元件標記為生命週期結束 (EOL) 而無法提供已淘汰平台分支的安全更新、技術支援或修補程序。例如，在淘汰的平台分支上執行的環境中可能會出現有害且嚴重的安全漏洞。或者，如果 EB API 動作隨著時間的推移變得與 Elastic Beanstalk 服務不相容，則可能會停止在此環境中工作。以淘汰的平台分支為基礎的環境保持作用中狀態的時間越長，這些類型的風險的機會就越大。

如果您的應用程式在淘汰的平台分支上執行時應該遇到問題，而您無法將其移轉到支援的平台，則需要考慮其他替代方案。解決方法包括將應用程式封裝到 Docker 映像檔以將其作為 Docker 容器執行。這將允許客戶使用我們的任何碼頭解決方案，例如我們的 Elastic Beanstalk 2023/AL2 碼頭平台或其他基於碼頭的服務，例如 Amazon ECS 或 Amazon EKS。非 Docker 替代方案包括我們的 AWS CodeDeploy 服務，它允許您完全自定義所需的運行時間。

6. 我是否可以提交延長淘汰日期的請求？

否。在淘汰日期之後，現有環境將繼續運行。但是，Elastic Beanstalk 將不再提供平台維護和安全更新。因此，如果您仍在以 AL1 為基礎的平台上執行應用程式，就必須遷移至 AL2 或 AL2023。如需風險和解決方法的詳細資訊，請參閱常見問答集 #5。

7. 如果我無法及時完成 AL2 或 AL2023 遷移，有什麼解決方法？

客戶可以繼續執行此環境，但我們強烈建議您計劃將所有 Elastic Beanstalk 環境遷移至支援的平台版本。這樣做將最大限度地減少風險，並繼續享有更新版本提供的重要安全、效能和功能增強的好處。如需風險和解決方法的詳細資訊，請參閱常見問答集 #5。

8. 遷移至 AL2 或 AL2023 平台的建議流程是什麼？

如需全面的 AL1 至 AL2023/AL2 遷移說明，請參閱 將您的 Elastic Beanstalk Linux 應用程式遷移到 Amazon Linux 2023 或 Amazon Linux 2。本主題說明 Elastic Beanstalk 需要藍/綠部署才能執行升級。

9. 如果我有一個在淘汰的平台上執行的環境，會產生什麼影響？

基於已淘汰平台分支的環境最終可能會出現不可預測的情況，因為 Elastic Beanstalk 由於供應商將已淘汰平台分支的元件標記為生命週期結束 (EOL) 而無法提供已淘汰平台分支的安全更新、技術支援或修補程序。例如，在淘汰的平台分支上執行的環境中可能會出現有害且嚴重的安全漏洞。或者，如果 EB API 動作隨著時間的推移變得與 Elastic Beanstalk 服務不相容，則可能會停止在此環境中工作。已淘汰平台分支上的環境保持作用中狀態的時間越長，這些類型的風險的機會就越大。如需詳細資訊，請參閱常見問答集 #5。

10. 淘汰日期後 90 天會發生什麼？

我們的淘汰平台分支原則不會移除對環境的存取，也不會刪除資源。但是，請注意，基於已淘汰平台分支的環境最終可能會出現不可預測的情況，因為 Elastic Beanstalk 由於供應商將已淘汰平台分支的元件標記為生命週期結束 (EOL) 而無法提供已淘汰平台分支的安全更新、技術支援或修補程序。例如，在淘汰的平台分支上執行的環境中可能會出現有害且嚴重的安全漏洞。或者，如果 EB API 動作隨著時間的推移變得與 Elastic Beanstalk 服務不相容，則可能會停止在此環境中工作。已淘汰平台分支上的環境保持作用中狀態的時間越長，這些類型的風險的機會就越大。如需詳細資訊，請參閱常見問答集 #5。

11. 是否能夠以淘汰的平台為基礎建立新環境？

如果您已使用該平台分支透過相同帳戶和在相同區域中建立現有環境，則能夠以淘汰的平台分支為基礎建立新環境。退休的平台分支將無法在 Elastic Beanstalk 控制台中使用。但是，對於具有以已淘汰平

台分支為基礎的現有環境的客戶，它將透過 EB CLI、EB API 和 AWS CLI提供。此外，現有客戶可以使用複製環境和重建環境主控台。但是，請注意，以已淘汰平台分支為基礎的環境最終可能會出現不可預測的情況。如需詳細資訊，請參閱常見問答集 #5。

12. 如果我在已淘汰的平台分支上執行現有的環境，直到何時可以根據淘汰的平台分支建立新環境？ 是否可以使用主控台、CLI 或 API 執行此操作？

您可以在處分日期之後建立環境。但是，請注意，淘汰的平台分支最終可能會出現不可預測的情況。此類環境的建立或作用中時間越長，環境遇到未預期問題的風險就越高。如需有關建立新環境的詳細資訊，請參閱常見問答集 #11。

13. 是否可以複製或重建以已淘汰平台為基礎的環境？

是。您可以使用複製環境和重建環境主控台來執行此操作。您也可以使用 EB CLI、EB API 和. AWS CLI如需有關建立新環境的詳細資訊，請參閱常見問答集 #11。

但是，我們強烈建議您計劃將所有 Elastic Beanstalk 環境遷移至支援的平台版本。這樣做將最大限度地減少風險，並繼續享有更新版本提供的重要安全、效能和功能增強的好處。如需風險和解決方法的詳細資訊，請參閱常見問答集 #5。

14. 在退休日期之後，我以退休平台分支為基礎的 Elastic Beanstalk 環境的 AWS 資源會發生什麼情況？ 例如，如果正在執行的 EC2 執行個體已終止，Elastic Beanstalk 是否可以啟動新的以 AL1 為基礎的 EC2 執行個體來維護容量？

環境資源將保持作用中狀態，並繼續運作。是，Elastic Beanstalk 可能會自動擴展環境中的 AL1 EC2 執行個體。但是，Elastic Beanstalk 將停止向環境提供新的平台維護更新，這可能導致環境隨著時間的推移最終出現不可預測的情況。如需詳細資訊，請參閱常見問答集 #5。

15. AL2023/AL2 與 Amazon Linux AMI (AL1) 作業系統之間的主要區別是什麼？ Elastic Beanstalk AL2023/AL2 平台分支會受到什麼影響？

雖然 Amazon Linux AMI 和 AL2023/AL2 具有相同的 Linux 核心，但它們的初始化系統、`libc` 版本、編譯器工具鏈及各種套件仍有所不同。如需詳細資訊，請參閱 Amazon Linux 2 常見問答集。

Elastic Beanstalk 服務也已更新執行時間的平台專用版本、建置工具及其他依存項目。以 AL2023/AL2 為基礎的平台分支並不保證能與您現有的應用程式回溯相容。此外，即使應用程式的程式碼成功部署到新平台版本，可能仍會因作業系統和執行時間差異而有不同的運作或執行方式。如需您需要檢閱和測試的組態和自訂的清單和描述，請參閱 將您的 Elastic Beanstalk Linux 應用程式遷移到 Amazon Linux 2023 或 Amazon Linux 2。

取消環境資訊更新及應用程式部署

您可取消環境資訊變更觸發的進行中更新。您亦可取消進行中的新應用程式版本的部署。例如，若您決定使用現有環境資訊進行設定，而非套用新的環境資訊設定，您可能想要取消更新。或者，您可能發現正部署的新應用程式版本出現問題，造成無法正常開始或執行。藉由取消環境更新或應用程式版本更新，您就不用等待至更新或部署程序完成，即可嘗試新的環境或應用程式版本更新。

> **Note**
>
> 清除階段會移除不再需要的舊資源，若此期間最後一批次的執行個體已更新完成，之後即無法取消更新。

Elastic Beanstalk 會以最後一次成功更新的相同方式來進行轉返。例如，若您的環境已啟用根據時間進行的滾動更新，則 Elastic Beanstalk 會等待兩批次執行個體轉返變更間，所指定的暫停時間。或者，若您近期啟用滾動更新，但環境資訊設定最後一次的成功更新並未使用滾動更新，則 Elastic Beanstalk 會同時轉返所有執行個體。

一旦 Elastic Beanstalk 開始取消更新，您便無法停止轉返至之前的環境資訊。轉返程序會持續下去，直到環境中所有執行個體均具備之前的環境資訊，或直到轉返程序失敗。以應用程式版本的部署而言，取消部署只會停止部署，部分執行個體將具備新的應用程式版本，而其他執行個體仍將持續執行現有應用程式版本。您可稍後再部署相同或另一個應用程式版本。

如需滾動更新的詳細資訊，請參閱 Elastic Beanstalk 滾動環境資訊更新。如需有關以批次進行應用程式版本部署的詳細資訊，請參閱 部署政策和設定。

欲取消更新

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

   > **Note**
   >
   > 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在環境概觀頁面上，選擇 Actions (動作)，然後選擇 Abort current operation (中止目前的操作)。

重建 Elastic Beanstalk 環境

若您未使用 Elastic Beanstalk 功能來修改或終止環境的基礎 AWS 資源，可能會導致 AWS Elastic Beanstalk 環境無法使用。如發生此情況，您可以重建環境，嘗試將其還原至運作狀態。重建環境會終止其中的所有資源，並替換為具備相同組態的新資源。

終止環境後，您也可於六週 (42 天) 內將其重建。重建時，Elastic Beanstalk 會嘗試以相同名稱、ID 和組態建立新的環境。

重建執行環境

您可透過 Elastic Beanstalk 主控台或 `RebuildEnvironment` API 來重建環境。

欲重建執行環境 (主控台)

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 選擇 Actions (動作)，然後選擇 Rebuild environment (重建環境)。
4. 選擇 Rebuild (重建)。

重建執行環境所建立的新資源，具備與舊資源相同的組態，然而，資源 ID 會不同，也不會還原舊資源上的資料。例如，重建具備 Amazon RDS 資料庫執行個體的環境，會以相同組態建立新的資料庫，但快照不會套用至新的資料庫。

若要透過 Elastic Beanstalk API 重建執行環境，請使用 `RebuildEnvironment` 動作並搭配 AWS CLI 或 AWS 軟體開發套件。

```
$ aws elasticbeanstalk rebuild-environment --environment-id e-vdnftxubwq
```

重建已終止環境

您可使用 Elastic Beanstalk 主控台、EB CLI 或 `RebuildEnvironment` API，藉此重建並還原已終止環境。

> **Note**
>
> 除非您的終止環境使用自訂網域名稱，否則環境會使用 elasticbeanstalk.com 的子網域。這些子網域共享於一個 Elastic Beanstalk 區域內。因此，相同區域的其他客戶可使用這些子網域來建立環境。在您環境終止後，另一個環境可使用其子網域。在這種情況下，重建將會失敗。您可使用自訂網域，藉此避免此問題。如需詳細資訊，請參閱 您 Elastic Beanstalk 環境的網域名稱。

最近終止的環境會於應用程式概觀中顯示最長一小時。在此期間內，您可於環境的儀表板檢視環境事件，並使用 Restore environment (還原環境) 動作來重建。

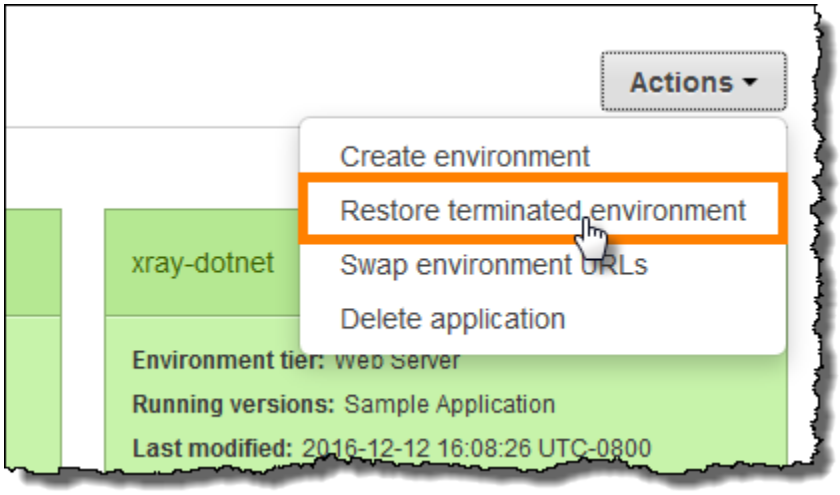
若要重建已不再顯示的環境，請於應用程式頁面使用 Restore terminated environment (還原已終止環境) 選項。

欲重建已終止環境 (主控台)

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Applications (應用程式)，然後在清單上選取應用程式名稱。

   > **Note**
   >
   > 如果您有許多應用程式，請使用搜尋列來篩選應用程式清單。

3. 選擇 Actions (動作)，然後選擇 Restore terminated environment (還原已終止環境)。



4. 選擇已終止環境
5. 選擇 Restore (還原)。

Elastic Beanstalk 會嘗試以相同名稱、ID 和組態建立新的環境。在您嘗試重建時，若具備相同名稱或 URL 的環境存在，則重建會失敗。刪除已部署至環境的應用程式版本，也會導致重建失敗。

若您使用 EB CLI 來管理您的環境，請使用 eb restore 命令來重建已終止環境。

```
$ eb restore e-vdnftxubwq
```

如需詳細資訊，請參閱 eb restore。

若要透過 Elastic Beanstalk API 重建已終止環境，請使用 RebuildEnvironment 動作並搭配 AWS CLI 或 AWS 軟體開發套件。

```
$ aws elasticbeanstalk rebuild-environment --environment-id e-vdnftxubwq
```

環境類型

在 AWS Elastic Beanstalk 中，您可建立有負載平衡且可擴展的環境或單一執行個體環境。您需要的環境類型，取決於您部署的應用程式。例如，您可在單一執行個體環境開發並測試應用程式，藉此節省成本，然後在應用程式準備投入生產時，將環境升級為有負載平衡且可擴展環境。

> **Note**
>
> 能夠處理背景任務的 Web 應用程式，其工作者環境層不包含負載平衡器。然而，工作者環境確實能夠新增執行個體，有效擴展為 Auto Scaling 群組，當負載需要時即可處理來自 Amazon SQS 佇列的資料。

有負載平衡且可擴展的環境

負載平衡和具可擴展性的環境會使用 Elastic Load Balancing 和 Amazon EC2 Auto Scaling 服務，藉此佈建您部署的應用程式所需 Amazon EC2 執行個體。Amazon EC2 Auto Scaling 會自動啟動額外的執行個體，以容納您的應用程式增加的負載。若您的應用程式負載減少，Amazon EC2 Auto Scaling 會停止執行個體，但永遠會保留執行您指定的執行個體最低數量。若您的應用程式需要擴展以取得在多個可用區域執行的選項，請使用有負載平衡且可擴展環境。若您不確定該選擇哪類型的環境，可直接挑選一個，之後再視需要切換環境類型。

單一執行個體環境

單一執行個體環境包含一個 Amazon EC2 執行個體和彈性 IP 地址。單一執行個體環境不含負載平衡器，相對於有負載平衡且可擴展的環境，可助您降低成本。儘管單一執行個體環境會使用 Amazon EC2 Auto Scaling 服務，但是執行個體數量上下限和所需容量均設定為 1。因此，不會啟動新的執行個體來容納您的應用程式增加的負載。

若您的生產應用程式預料將擁有低流量，或者您希望遠端開發，請使用單一執行個體環境。若您不確定該選擇哪類型的環境，可直接挑選一個，之後再視需要切換環境類型。如需更多詳細資訊，請參閱 變更環境類型。

變更環境類型

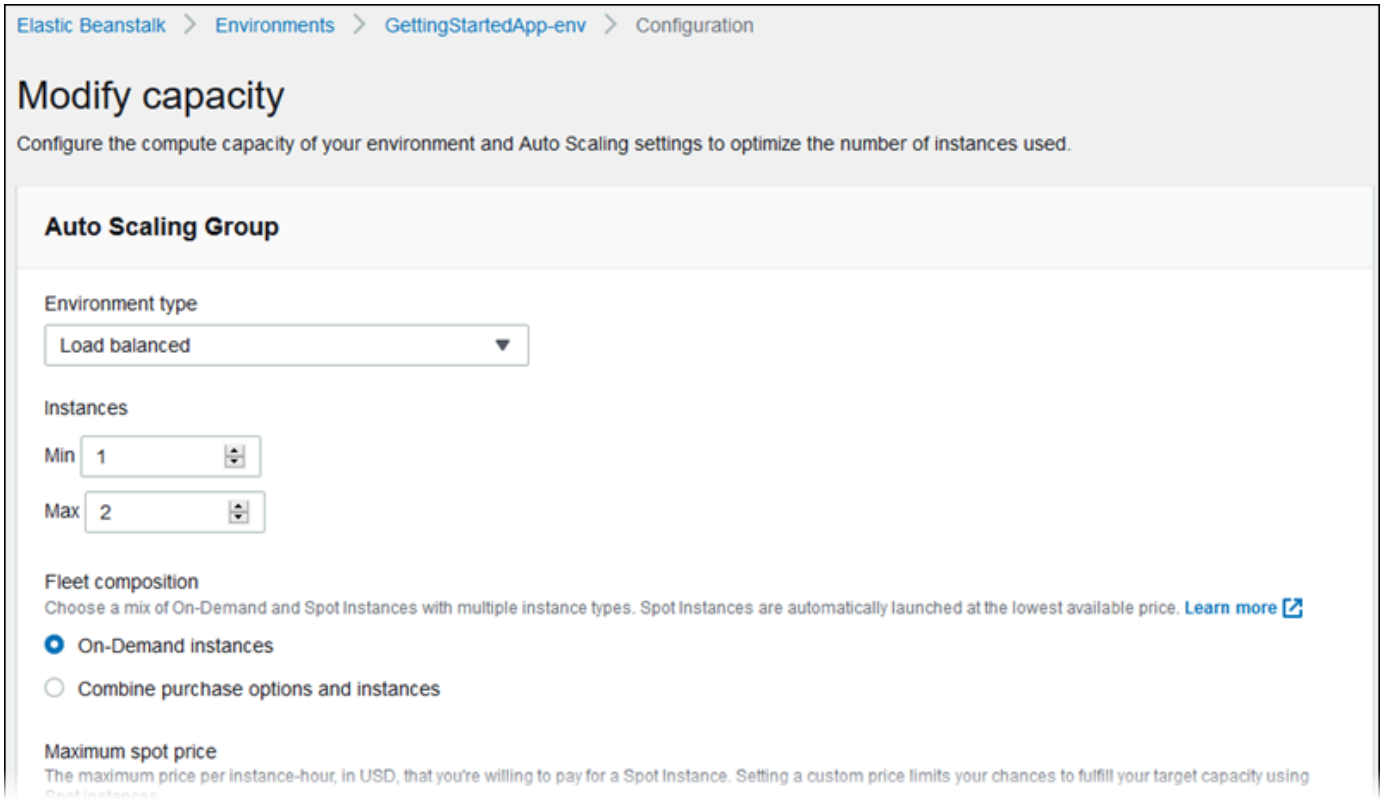
您可以編輯環境的組態，藉此將環境類型變更為單一執行個體環境或有負載平衡且可擴展的環境。在某些情況下，您可能想要變更環境類型。例如，假設您為了節省成本，已在單一執行個體環境開發並測試應用程式。在應用程式準備投入生產時，您可以將環境變更為有負載平衡且可擴展的環境，讓其能夠擴展以滿足客戶的需求。

欲變更環境的類型

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

   > **Note**
   >
   > 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇 Configuration (組態)。
4. 在 Capacity (容量) 類別中，選擇 Edit (編輯)。
5. 在 Environment Type (環境類型) 清單中，選取您想要的環境類型。



6. 選擇 Save (儲存)。

   在 Elastic Beanstalk 佈建 AWS 資源的同時，環境更新可能需要幾分鐘。

若您的環境位於 VPC，請選取放置 Elastic Load Balancing 和 Amazon EC2 執行個體的子網路。執行您應用程式的每個可用區域，都必須具備兩者。如需詳細資訊，請參閱 搭配 Amazon VPC 使用 Elastic Beanstalk。

Elastic Beanstalk 工作者環境

若您 AWS Elastic Beanstalk 應用程式執行的操作或工作流程需要長時間完成，您可將這些任務交由專屬工作者環境處理。將 Web 應用程式前端自阻擋式操作的執行程序去耦，是確保應用程式在負載下保持回應能力的常見方式。

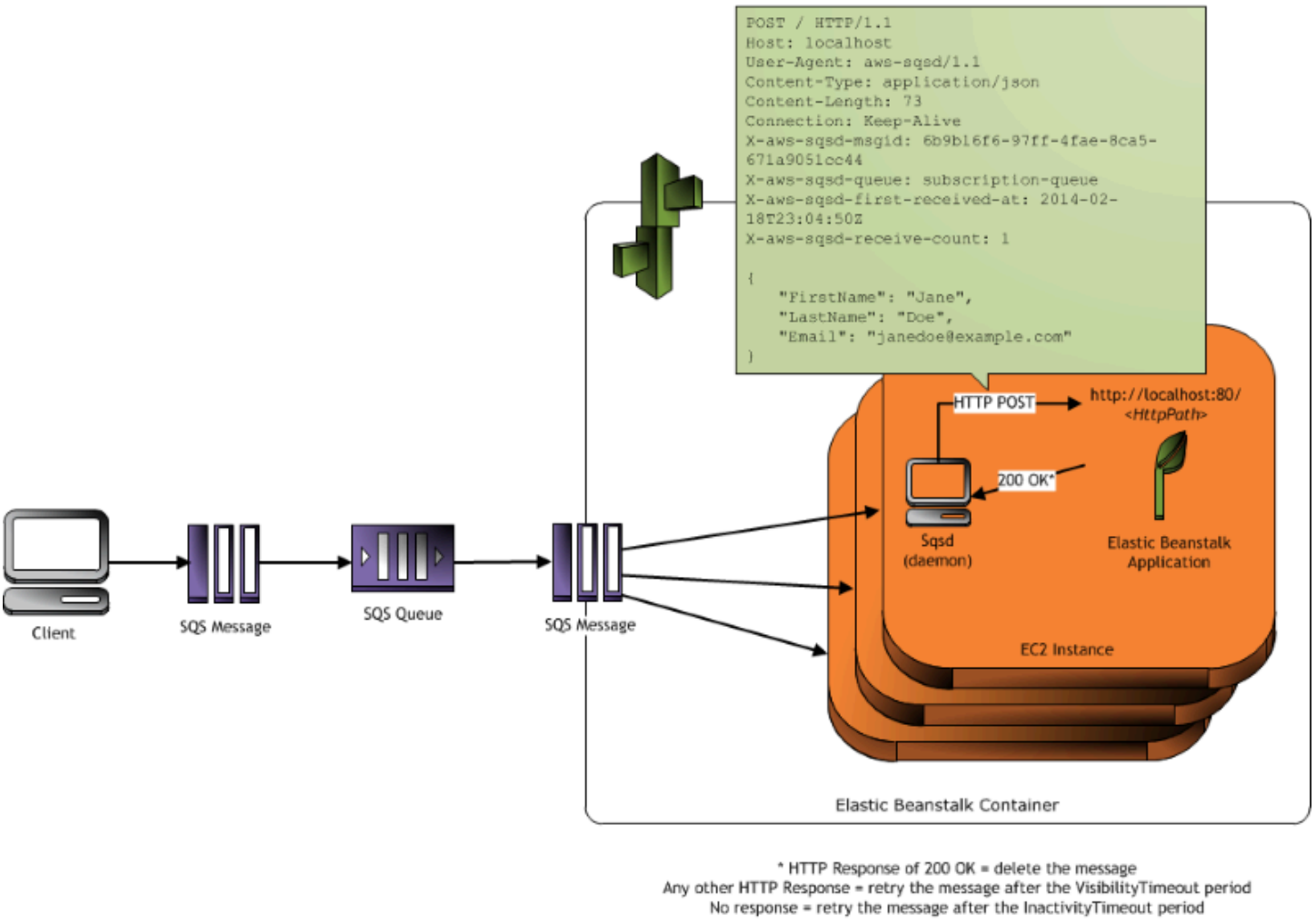
長時間執行的任務是指會大幅增加完成請求時間的操作，例如處理圖片或影片、傳送電子郵件或產生 ZIP 封存檔。這些操作只需一或兩秒就能完成，但對於要在 500 毫秒內完成的 Web 請求而言，延遲幾秒鐘是很長的時間。

一個選項是於本機使用工作者程序，回傳成功值，以非同步方式處理任務。如果您的執行個體處理任務的速度跟得上接收到的任務，這個運作方式即有效。然而，在高負載下，執行個體可能因背景任務而導

致負擔過重，無法回應高優先順序的請求。若個別使用者可產生多個任務，則負載的增加量可能與使用者增加量不相符，導致 Web 伺服器層無法有效擴展。

欲避免在本機執行的任務時間過長，您可使用您程式設計語言適用的 AWS 開發套件，將任務傳送至 Amazon Simple Queue Service (Amazon SQS) 佇列，並於另一組執行個體上執行這些任務。然後您可以設計這些工作者執行個體，只會從佇列提取有能力執行的項目，避免負擔過重。

Elastic Beanstalk 工作者環境會管理 Amazon SQS 佇列，並在為您讀取佇列的執行個體上執行協助程式程序，藉此簡化整個程序。當協助程式從佇列提取項目時，會將 HTTP POST 請求透過本機傳送至連接埠 80 上的 `http://localhost/`，內文則包含佇列訊息的內容。您的應用程式只需要回應 POST，執行長時間的任務。您可設定協助程式以發佈到其他路徑、使用 application/JSON 以外的 MIME 類型、連接至現有佇列或自訂連線 (最大並行工作數)、逾時和重試。



透過周期性任務，您亦可將工作者協助程式設定為根據 Cron 排程排序訊息。每項周期性任務均可向不同路徑發送 POST 請求。將 YAML 檔案納入定義任務排程和路徑的原始碼，藉此啟用周期性任務。

> **Note**
>
> Windows Server 平台上的 .NET 不支援工作者環境。

章節

工作者環境 SQS 協助程式

工作者環境會執行 Elastic Beanstalk 提供的協助程式程序。本協助程式會定期更新，以新增功能和修正錯誤。欲取得協助程式的最新版本，請更新至最新的平台版本。

當工作者環境的應用程式回傳 `200 OK` 回應以確認已接收並成功處理請求，協助程式會向 Amazon SQS 佇列傳送 `DeleteMessage` 呼叫，藉此從佇列刪除訊息。若應用程式回傳 `200 OK` 之外的其他回應，則 Elastic Beanstalk 會等待已設定的 `ErrorVisibilityTimeout` 期間，之後將訊息放回佇列。若沒有回應，Elastic Beanstalk 會等待 `InactivityTimeout` 期間，之後才將訊息放回佇列，以便再次嘗試處理該訊息。

> **Note**
>
> Amazon SQS 佇列的屬性 (訊息順序、至少傳遞一次和訊息採樣) 可能會影響您於工作者環境設計 Web 應用程式的方式。如需詳細資訊，請參閱《Amazon Simple Queue Service 開發人員指南》中的分散式佇列屬性。

Amazon SQS 會自動刪除處於佇列中的時間超過已設定 `RetentionPeriod` 之訊息。

協助程式會設定下列 HTTP 標頭。

HTTP 標頭

名稱	Value (值)

HTTP 標頭	

`User-Agent`	`aws-sqsd`
	
	`aws-sqsd/1.1 1`

`X-Aws-Sqsd-Msgid`	SQS 訊息 ID，用於偵測訊息風暴 (新訊息數量異常高的情況).
`X-Aws-Sqsd-Queue`	SQS 佇列名稱.
`X-Aws-Sqsd-First-Received-At`	訊息首次接收到的 UTC 時間 (採用 ISO 8601 格式)。
`X-Aws-Sqsd-Receive-Count`	SQS 訊息接收計數.
`X-Aws-Sqsd-Attr-` *`message-attribute-name`*	指派至正在處理訊息的自訂訊息屬性。`message-attribute-name` 為實際訊息屬性名稱。所有字串和數字訊息屬性都會新增到標頭。二進位屬性將會予以捨棄，不會包含在標頭。
`Content-Type`	Mime 類型組態，預設為 `application/json`。

無效信件佇列

Elastic Beanstalk 工作者環境支援 Amazon Simple Queue Service (Amazon SQS) 無效字母佇列。當訊息因某些原因而無法成功處理時，其他 (來源) 佇列可將該訊息傳送至無效字母佇列。無效字母佇列的主要優點，就是能夠放棄並隔離未成功處理的訊息。您可之後再分析傳送至無效字母佇列的訊息，設法判斷未成功處理的原因。

若您建立工作者環境層時指定自動產生的 Amazon SQS 佇列，則工作者環境預設會啟用無效字母佇列。若您針對工作者環境選擇現有 SQS 佇列，則必須使用 SQS 來單獨設定無效字母佇列。如需有關如何使用 SQS 來設定無效字母佇列的資訊，請參閱使用 Amazon SQS 無效字母佇列相關文章。

您無法停用無效字母佇列。無法傳送的訊息最終會傳送至無效字母佇列。然而，您可將 `MaxRetries` 選項設定為最大有效值 100，即可成功停用此功能。

如果未針對您工作者環境的 Amazon SQS 佇列設定無效信件佇列，Amazon SQS 會將訊息保留在佇列上，直到保留期間到期為止。如需設定保留期間的詳細資訊，請參閱the section called “設定工作者環境”。

> **Note**
>
> Elastic Beanstalk `MaxRetries` 選項等同於 SQS `MaxReceiveCount` 選項。若您的工作者環境未使用自動產生的 SQS 佇列，請於 SQS 使用 `MaxReceiveCount` 選項來成功停用您的無效字母佇列。如需詳細資訊，請參閱使用 Amazon SQS 無效字母佇列相關文章。

如需 SQS 訊息生命週期的詳細資訊，請參閱訊息生命週期相關文章。

周期性任務

您可於原始碼套件內名為 `cron.yaml` 的檔案定義周期性任務，以將工作自動定期新增至您的工作者環境佇列。

例如，下列 `cron.yaml` 檔案會建立兩個週期性工作。第一個每 12 小時執行一次，第二個每天在 UTC 時間下午 11:00 執行一次。

Example cron.yaml

```
version: 1
cron:
 - name: "backup-job"
   url: "/backup"
   schedule: "0 */12 * * *"
 - name: "audit"
   url: "/audit"
   schedule: "0 23 * * *"
```

每個任務的 **name** (名稱) 都必須獨一無二。此 URL 則是 POST 請求會傳送的路徑，以觸發工作。排程是決定任務執行時間的 CRON 表達式。

執行任務時，協助程式會向環境的 SQS 佇列發佈訊息，其標頭會指出需要執行的工作。環境中的執行個體都可提取訊息並處理工作。

> **Note**
>
> 如果您以現有的 SQS 佇列來設定您的工作者環境，並選擇了一個 Amazon SQS FIFO 佇列，則不支援週期性任務。

Elastic Beanstalk 會使用領導者選擇，以決定哪個工作者環境佇列的執行個體應排序周期性任務。每個執行個體可寫入 Amazon DynamoDB 資料表，藉此嘗試成為領導者。第一個成功的執行個體就是領導者，而且必須持續寫入資料表以維持其領導者狀態。若領導者停止服務，另一個執行個體會快速替補。

以周期性任務而言，工作者協助程式會設定下列其他標頭。

HTTP 標頭

名稱	Value (值)
`X-Aws-Sqsd-Taskname`	欲執行的周期性任務名稱。
`X-Aws-Sqsd-Scheduled-At`	周期性任務排程的時間。
`X-Aws-Sqsd-Sender-Id`	訊息寄件者的 AWS 帳戶編號。

使用 Amazon CloudWatch 在工作者環境層中自動調整規模

Amazon EC2 Auto Scaling 和 CloudWatch 會共同監控工作者環境中正在執行的執行個體之 CPU 使用率。您如何設定 CPU 容量的自動調整規模限制，會決定 Auto Scaling 群組為了適當管理 Amazon SQS 佇列中的訊息傳輸量，須執行的執行個體數量。每個 EC2 執行個體會將其 CPU 使用率指標發佈至 CloudWatch 。Amazon EC2 Auto Scaling 會從 CloudWatch 擷取在工作者環境所有執行個體的平均 CPU 使用量。您必須設定閾值上下限，並根據 CPU 容量，設定欲新增或終止的執行個體數量。當 Amazon EC2 Auto Scaling 偵測到您已達到指定的 CPU 容量閾值上限，Elastic Beanstalk 會於工作者環境建立新的執行個體。當 CPU 負載降回閾值內時，將刪除這些執行個體。

> **Note**
>
> 執行個體被終止時未處理的訊息，會回傳至佇列供仍在執行的執行個體上之協助程式處理。

您亦可使用 Elastic Beanstalk 主控台、CLI 或選項檔案，視需要設定其他 CloudWatch 警示。如需詳細資訊，請參閱搭配 Amazon CloudWatch 使用 Elastic Beanstalk和建立 Auto Scaling 群組與步驟擴展政策。

設定工作者環境

您可於環境管理主控台中的 Configuration (組態) 頁面編輯 Worker (工作者) 類別，藉此管理工作者環境的組態。

Elastic Beanstalk > Environments > GettingStartedApp-env > Configuration

Modify worker

You can create a new Amazon SQS queue for your worker application or pull work items from an existing queue. The worker daemon on the instances in your environment pulls an item from the queue and relays it in the body of a POST request to a local HTTP path relative to localhost.

Queue

Worker queue

Autogenerated queue

SQS queue from which to read work items.

Messages

HTTP path

/

The daemon pulls items from the Amazon SQS queue and posts them locally to this path.

MIME type

application/json

Change the MIME type of the POST requests that the worker daemon sends to your application.

HTTP connections

50

Maximum number of concurrent connections to the application.

Visibility timeout

300 seconds

The amount of time to lock an incoming message for processing before returning it to the queue.

Error visibility timeout

seconds

The amount of time to wait before resending a message after an error response from the application.

Advanced options

The following settings control advanced behavior of the worker tier daemon. Learn more

Max retries

10

Maximum number of retries after which the message is discarded.

Connection timeout

5

Inactivity timeout

> **Note**
>
> 您可以設定 URL 路徑來張貼工作者佇列中的訊息，但您無法設定 IP 連接埠。Elastic Beanstalk 一律會在連接埠 80 上張貼工作者佇列訊息。工作者環境應用程式或其 Proxy 必須聆聽連接埠 80。

欲設定工作者協助程式

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

   > **Note**
   >
   > 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇 Configuration (組態)。
4. 在 Worker (工作者) 組態類別中，選擇 Edit (編輯)。

Modify worker (修改工作者) 組態頁面具有下列選項。

在 Queue (佇列) 區段中：

- Worker queue (工作者佇列) – 指定協助程式讀取的 Amazon SQS 佇列。如有的話，您可選擇現有佇列。若您選擇 Autogenerated queue (自動產生佇列)，則 Elastic Beanstalk 會建立新的 Amazon SQS 佇列以及對應的 Worker queue URL (工作者佇列 URL)。

  > **Note**
  >
  > 當您選擇 Autogenerated queue (自動產生佇列) 時，Elastic Beanstalk 建立的佇列將為標準 Amazon SQS 佇列。當您選擇現有佇列時，您可以提供標準或 FIFO Amazon SQS 佇列。請注意，如果您提供了 FIFO 佇列，則不支援週期性任務。

- Worker queue URL (工作者佇列 URL) – 若您選擇現有 Worker queue (工作者佇列)，則此設定會顯示與 Amazon SQS 佇列相關聯的 URL。

在 Messages (訊息) 區段中：

- HTTP path (HTTP 路徑) – 指定將自 Amazon SQS 佇列接收資料的應用程式相對路徑。此資料會插入 HTTP POST 訊息的訊息本文。預設值為 `/`。
- MIME type (MIME 類型) – 指出 HTTP POST 訊息使用的 MIME 類型。預設值為 `application/json`。然而，任何值均有效，因為您可以建立並指定自己的 MIME 類型。
- HTTP connections (HTTP 連線) – 指定協助程式可對 Amazon EC2 執行個體內的應用程式進行並行連線的數量上限。預設值為 **50**。您可以指定 **1** 至 **100**。
- Visibility timeout (可見性逾時) – 指出從 Amazon SQS 佇列內送的訊息被鎖定以進行處理的所需時間 (秒)。在設定的時間數過後，該則訊息才能在佇列中被其他協助程式再次讀取。請選擇多於您預期應用程式處理訊息的時間的值，上限為 **43200** 秒。
- Error visibility timeout (錯誤可見性逾時) – 指出因明確錯誤而失敗的處理嘗試之後、Elastic Beanstalk 傳回訊息給 Amazon SQS 佇列前經過的時間 (秒)。您可以指定 **0** 至 **43200** 秒。

在 Advanced options (進階選項) 區段中：

- Max retries (重試次數上限) – 指定 Elastic Beanstalk 在將訊息移往無效字母佇列前嘗試將訊息傳送至 Amazon SQS 佇列的次數上限。預設值為 **10**。您可以指定 **1** 至 **100**。

  > **Note**
  >
  > 此 Max retries (重試次數上限) 選項僅適用於設定了無效字母佇列的 Amazon SQS 佇列。對於任何未設定無效字母佇列的 Amazon SQS 佇列，Amazon SQS 會將訊息保留在該佇列中，直到 Retention period (保留期間) 選項指定的期限為止。

- Connection timeout (連線逾時) – 指出等待成功連線到應用程式所需的時間 (以秒為單位)。預設值為 **5**。您可以指定 **1** 至 **60** 秒。
- Inactivity timeout (閒置逾時) – 指出等待現有應用程式連線出現回應的所需時間 (以秒為單位)。預設值為 **180**。您可以指定 **1** 至 **36000** 秒。
- Retention period (保留期間) – 指出訊息有效並主動處理所需的時間 (秒)。預設值為 **345600**。您可以指定 **60** 至 **1209600** 秒。

若您使用現有 Amazon SQS 佇列，則在您建立工作者環境時所進行的設定，可能會與直接在 Amazon SQS 內進行的設定衝突。例如，若您設定工作者環境所使用的 `RetentionPeriod` 值高於 Amazon SQS 內所設定的 `MessageRetentionPeriod` 值，則 Amazon SQS 會在訊息存在時間超過 `MessageRetentionPeriod` 值時將其刪除。

反之，若您於工作者環境設定的 RetentionPeriod 值低於 Amazon SQS 內所設定的 MessageRetentionPeriod 值，則協助程式會早 Amazon SQS 一步刪除訊息。以 VisibilityTimeout 而言，您於工作者環境設定的協助程式所設定的值，會覆寫 Amazon SQS 的 VisibilityTimeout 設定。請比較您的 Elastic Beanstalk 設定和 Amazon SQS 設定，確保訊息會適當刪除。

在 Elastic Beanstalk 環境之間建立連結

由於您應用程式的大小和複雜性均提升，建議您將其分割為具備不同開發與營運生命週期的元件。透過執行藉由定義良好的界面與彼此互動的較小服務，團隊能夠獨立工作，且部署可降低風險。 AWS Elastic Beanstalk 可讓您連結您的環境，以共用彼此相關之元件間的資訊。

> **Note**
>
> 除了多容器 Docker 外，Elastic Beanstalk 目前支援所有平台的環境連結。

透過環境連結，您可將應用程式元件間的環境連線，指定為具名參考。當您建立定義連結的環境，Elastic Beanstalk 會以連結為名，設定一個環境變數。此變數的值就是您可用於連結其他元件的端點，可能是 Web 伺服器或工作者環境。

例如，若您應用程式的前端可收集電子郵件地址，而其工作者則將歡迎電子郵件傳送至前端收集的電子郵件地址，則您可於前端建立該工作者的連結，前端即可自動尋找工作者的端點 (佇列 URL)。

於環境資訊清單 (一個名為 env.yaml 的 YAML 格式檔案，位於應用程式原始碼的根目錄) 中，定義其他環境的連結。下列資訊清單定義名為 worker 的環境的連結：

~/workspace/my-app/frontend/env.yaml

```
AWSConfigurationTemplateVersion: 1.1.0.0
EnvironmentLinks:
  "WORKERQUEUE": "worker"
```

當您透過內含上述環境資訊清單的應用程式版本來建立環境，Elastic Beanstalk 會尋找屬於相同應用程式且名為 worker 的環境。如果該環境存在，Elastic Beanstalk 會建立一個名為 WORKERQUEUE 的環境屬性。WORKERQUEUE 的值是 Amazon SQS 佇列 URL。前端應用程式可讀取此屬性，如同讀取環境變數一般。如需詳細資訊，請參閱 環境資訊清單 (env.yaml)。

若要使用環境連結，請將環境資訊清單新增至應用程式來源，然後使用 EB CLI AWS CLI 或 SDK 上傳。如果您使用 AWS CLI 或 SDK，請在呼叫時設定process旗標CreateApplicationVersion：

```
$ aws elasticbeanstalk create-application-version --process --application-name
 my-app --version-label frontend-v1 --source-bundle S3Bucket="DOC-EXAMPLE-
BUCKET",S3Key="front-v1.zip"
```

此選項會指示 Elastic Beanstalk 在您建立應用程式版本時，於原始碼套件內驗證環境資訊清單及組態檔案。若您的專案目錄內含環境資訊清單，EB CLI 會自動設定此旗標。

使用任一用戶端正常建立您的環境。當您需要終止環境時，請先終止帶有連結的環境。若某環境受到另一環境連結，Elastic Beanstalk 將保護受連結環境不被終止。欲覆寫此項保護，請使用 `ForceTerminate` 旗標。此參數在 AWS CLI 則為 `--force-terminate`：

```
$ aws elasticbeanstalk terminate-environment --force-terminate --environment-name
 worker
```

設定 Elastic Beanstalk 環境

AWS Elastic Beanstalk 提供了廣泛的選項，可用於自訂環境中的資源，以及 Elastic Beanstalk 行為和平台設定。當您建立 Web 伺服器環境時，Elastic Beanstalk 會建立數個資源來支援您應用程式的操作。

- EC2 執行個體 – Amazon Elastic Compute Cloud (Amazon EC2) 虛擬機器，已設為在您選擇的平台上執行 Web 應用程式。

  每個平台會執行特定的一套軟體、設定檔和指令碼，來支援特定的語言版本、架構、Web 容器或其組合。大多數的平台使用會 Apache 或 NGINX 做為反向代理，此反向代理會在您 Web 應用程式的前景執行、轉傳遞交給此 Web 應用程式的請求、提供靜態資產，並產生存取和錯誤日誌。
- 執行個體安全群組 - Amazon EC2 安全群組，已設為允許從連接埠 80 傳入的流量。此資源可讓負載平衡器傳來的 HTTP 傳輸資料，到達執行您 Web 應用程式的 EC2 執行個體。在預設情況下，不允許傳輸資料從其他通訊埠傳送。
- 負載平衡器 - Elastic Load Balancing 負載平衡器，可設定將請求分配到執行您應用程式的執行個體。負載平衡器也讓您的執行個體不需直接連接到網際網路。
- 負載平衡器安全群組 - Amazon EC2 安全群組，設為允許從連接埠 80 傳入的流量。此資源可讓來自網際網路的 HTTP 傳輸資料到達負載平衡器。在預設情況下，不允許傳輸資料從其他通訊埠傳送。
- Auto Scaling 群組 - Auto Scaling 群組，設為在執行個體終止或無法使用時，取代該執行個體。
- Amazon S3 儲存貯體 - 儲存位置，用來儲存當您使用 Elastic Beanstalk 時所建立的原始程式碼、日誌和其他成品。
- Amazon CloudWatch 警示 — 監控環境中執行個體負載的兩個 CloudWatch 警示，並在負載過高或過低時觸發警示。當警示觸發時，您的 Auto Scaling 群組會擴展或縮減以進行回應。
- AWS CloudFormation 堆疊 — Elastic Beanstalk 用 AWS CloudFormation 來啟動環境中的資源並傳播組態變更。資源定義於範本中，您可在 AWS CloudFormation 主控台中檢視此範本。
- 網域名稱 – 會路由到您 Web 應用程式的網域名稱，其格式為 *subdomain*.*region*.elasticbeanstalk.com。

> **Note**
>
> 為了增強 Elastic Beanstalk 應用程式的安全性，我們會在公共后缀列表 (PSL) 中註冊網域 elasticbeanstalk.com。為了加強安全性，如果您需要在 Elastic Beanstalk 應用程式的預設網域名稱中設定敏感性 Cookie，我們建議您使用具 `__Host-` 前置詞的 Cookie。此做法將有

助於保護您的網域免受跨站請求偽造 (CSRF) 攻擊。如需更多資訊，請參閱 Mozilla 開發人員網路中的設定 Cookie 頁面。

本主題著重於 Elastic Beanstalk 主控台中可用的資源組態選項。下列主題顯示如何在主控台設定您的環境。這些主題也同時描述基礎命名空間，其對應到主控台選項來和組態檔案或 API 組態選項搭配使用。若要了解進階組態方法，請參閱 設定環境 (進階)。

主題

使用 Elastic Beanstalk 主控台設定環境資訊

您可以使用 Elastic Beanstalk 主控台檢視和修改您的環境及其資源的許多組態選項。您可以在部署期間自訂環境行為、啟用其他功能，並在環境建立期間修改執行個體類型和所選的其他設定。

檢視環境資訊摘要

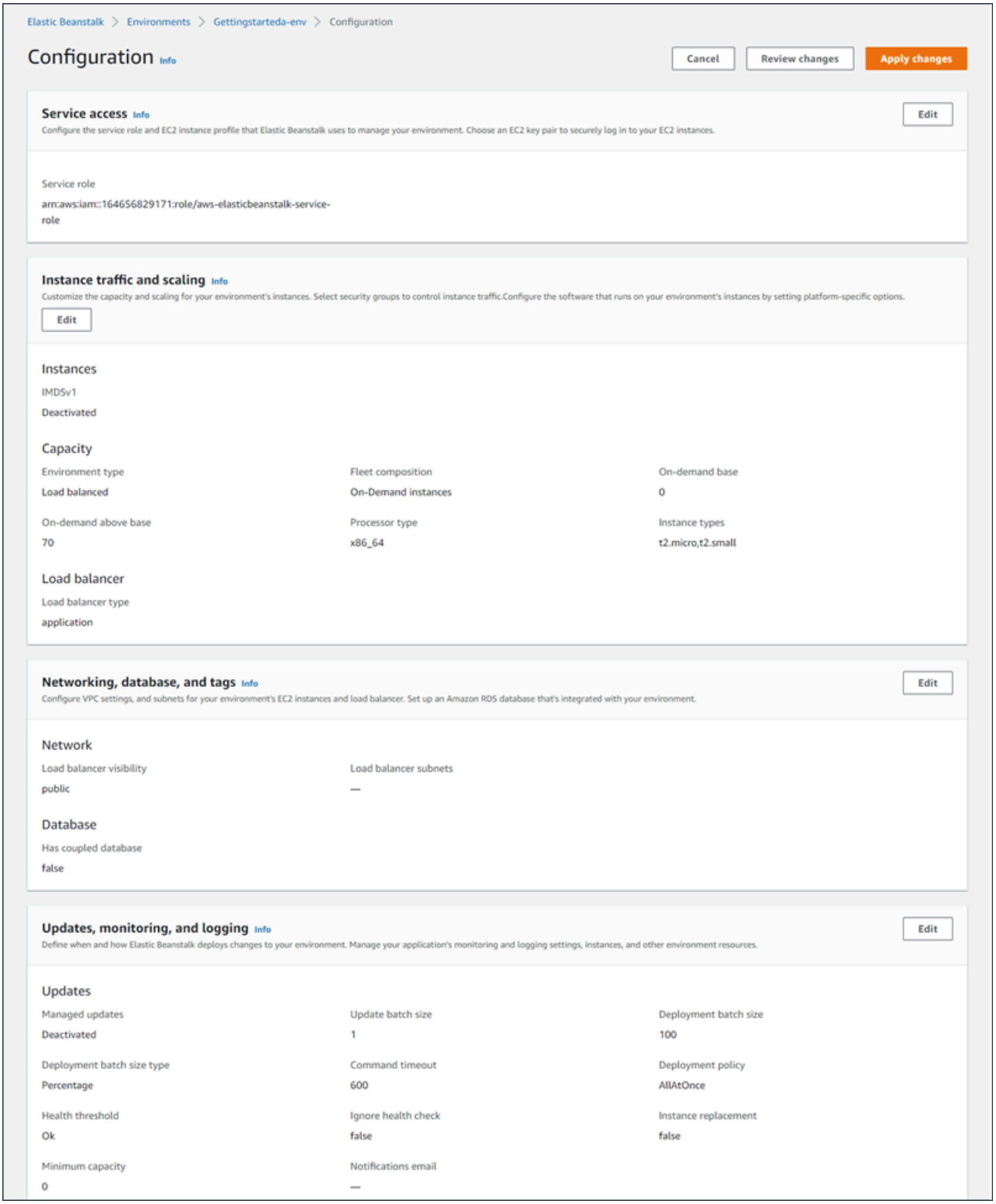
1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇 Configuration (組態)。

組態頁面

Configuration overview (組態概觀) 頁面會顯示一組組態類別。每個組態類別彙總的目前狀態一組相關的選項。



選擇組態類別中的 Edit (編輯) 以取得相關組態頁面，您可在此查看完整的選項值並進行變更。完成檢視及修改選項時，可選擇以下其中一個動作：

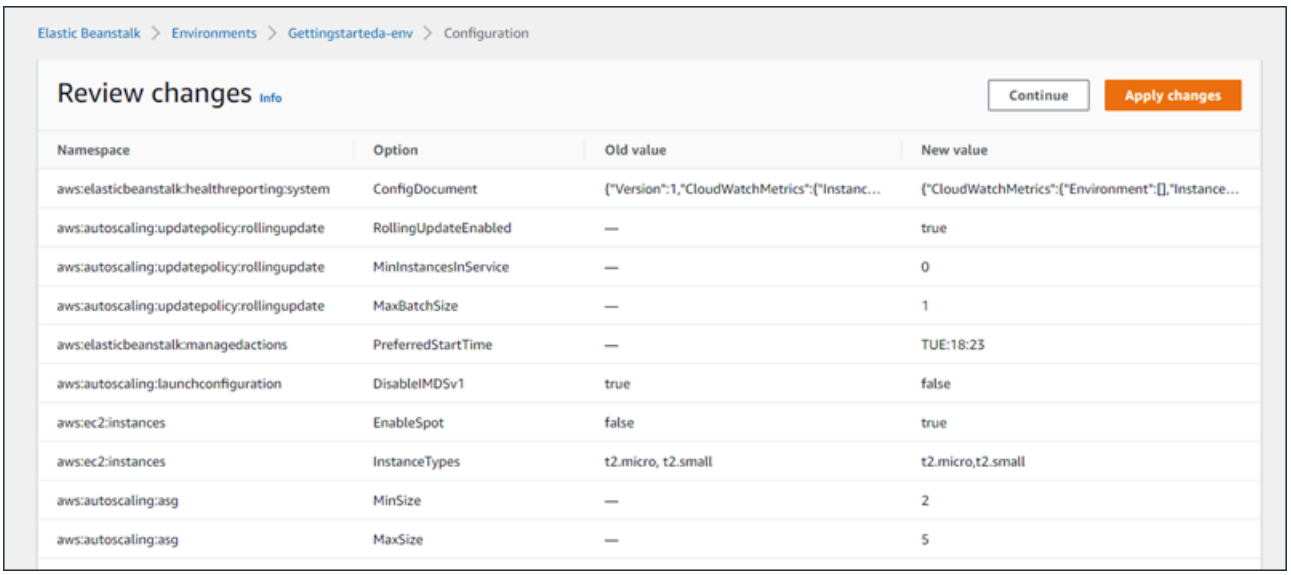
- Cancel (取消) – 返回環境的儀表板，而不套用您的組態變更。當您選擇 Cancel (取消) 時，主控台會遺失您在任何組態類別上所做的所有待處理的變更。

  您也可以藉由選擇另一個主控台頁面來取消組態變更，例如 Events (事件) 或 Logs (日誌)。在此情況下，如果有任何待處理的組態變更，主控台會提示您確認同意忽視這些變更。
- Review changes (檢閱變更) – 取得您在任何組態類別上所做的所有待處理變更的摘要。如需詳細資訊，請參閱檢閱變更頁面。
- Apply changes (套用變更) – 將您在任何組態類別中所做的變更套用至您的環境。在某些情況下，將會提示您確認其中一個組態決策的後果。

檢閱變更頁面

Review Changes (檢閱變更) 頁面會顯示一個表格，其中包含您在任何組態類別中所做的所有待處理且尚未套用至環境的選項變更。

這些表格會以 Namespace (命名空間) 加 Option (選項) 的方式列出每個選項，Elastic Beanstalk 可透過此組合識別選項。如需詳細資訊，請參閱組態選項。



完成檢閱變更時，可選擇以下其中一個動作：

- Continue (繼續) – 返回 Configuration overview (組態概觀) 頁面。然後，您可以繼續進行變更或套用待處理的變更。
- Apply changes (套用變更) – 將您在任何組態類別中所做的變更套用至您的環境。在某些情況下，將會提示您確認其中一個組態決策的後果。

您 Elastic Beanstalk 環境的 Amazon EC2 執行個體

建立網頁伺服器環境時，請 AWS Elastic Beanstalk 建立一或多個 Amazon Elastic Compute Cloud (Amazon EC2) 虛擬機器 (稱為執行個體)。

您環境中的執行個體會設定為可在您選擇的平台上執行 Web 應用程式。您可以在建立環境時，或在已運作後，變更您環境執行個體的各種屬性和行為。或者，您可以透過修改部署至環境的原始碼來進行這些變更。如需詳細資訊，請參閱 the section called “組態選項”。

> **Note**
>
> 您環境內的 Auto Scaling 群組可管理執行您應用程式的 Amazon EC2 執行個體。當您執行此頁面所述的組態變更時，啟動組態也會隨著變動。啟動組態是 Amazon EC2 啟動範本或 Auto Scaling 群組啟動組態的資源。此變更需要更換所有執行個體，此外也會觸發滾動更新或不可變更新，實際情形取決於您所設定的項目。

Elastic Beanstalk 支援多種 Amazon EC2 執行個體購買選項：「隨需執行個體」、「預留執行個體」和「Spot 執行個體」。隨需執行個體是一項 pay-as-you-go資源，使用時不需要長期承諾。預留執行個體是預先購買的計費折扣，會自動套用到您環境中相符的隨需執行個體。Spot 執行個體是未使用的 Amazon EC2 執行個體，其使用價格低於隨需定價。您可以設定單一選項，藉此在環境中啟用 Spot 執行個體。您可以使用其他選項，設定 Spot 執行個體使用方式，包括混合使用隨需與 Spot 執行個體。如需詳細資訊，請參閱 Auto Scaling 群組。

章節

Amazon EC2 執行個體類型

當您建立新環境時，Elastic Beanstalk 會根據您選擇的 Amazon EC2 執行個體類型來佈建 Amazon EC2 執行個體。您選擇的執行個體類型會決定執行您執行個體的主機硬體。EC2 執行個體類型可依

據其採取的處理器架構來分類。Elastic Beanstalk 支援以下列處理器架構為基礎的執行個體類型：AWS 引力同 64 位元臂架構 (arm64)、64 位元架構 (x86) 和 32 位元架構 (i386)。當您建立新環境時，Elastic Beanstalk 預設會選取 x86 處理器架構。

> **Note**
>
> 大多數 Elastic Beanstalk 平台都不再支援 i386 32 位元架構。建議您改為選擇 x86 或 arm64 架構類型。Elastic Beanstalk 會在 `aws:ec2:instances` 命名空間中提供 i386 處理器執行個體類型的組態選項。

指定 Elastic Beanstalk 環境組態中的所有執行個體類型都必須採取相同類型的處理器架構。假設您在已擁有 t2.medium 執行個體類型 (以 x86 架構為基礎) 的現有環境中新增執行個體類型，您只能新增相同架構的其他執行個體類型，例如 t2.small。如果您想要以不同架構的執行個體類型取代現有的執行個體類型，可以這麼做。但請確定命令中的所有執行個體類型都是以相同類型的架構為基礎。

Amazon EC2 推出相容的執行個體類型後，Elastic Beanstalk 會不斷新增對這些執行個體類型的相關支援。如需有關可用執行個體類型的資訊，請參閱 Amazon EC2 使用者指南中的執行個體類型或 Amazon EC2 使用者指南中的執行個體類型。

> **Note**
>
> Elastic Beanstalk 現在在所有支援重力彈性的區域中，在所有最新的 Amazon Linux 2 平台上提供重力子的支援。 AWS 如需進一步了解如何使用以 arm64 為基礎的執行個體類型建立 Elastic Beanstalk 環境，請參閱設定您環境的 Amazon EC2 執行個體。
> 建立在 arm64 架構上執行 Amazon EC2 執行個體的新環境，並使用 Elastic Beanstalk 中的部署選項，將現有的應用程式遷移到該環境。
> 若要進一步了解以重力通 arm64 為基礎的處理器，請參閱下列資源： AWS
>
> - 優點 — AWS 重力子處理器
> - 開始使用和其他主題，例如特定於語言的考量 — 開始使用 AWS Graviton 文章 GitHub

設定您環境的 Amazon EC2 執行個體

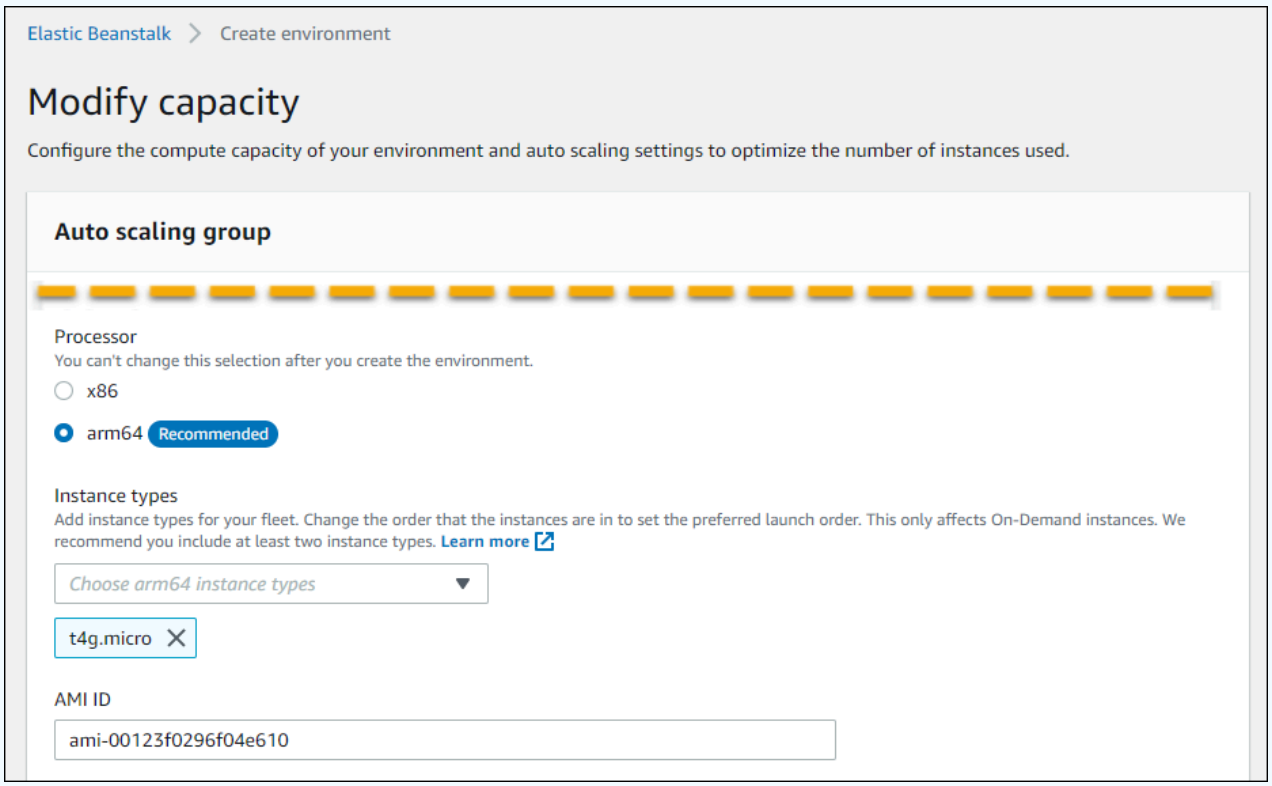
您可以在 Elastic Beanstalk 主控台中建立或修改 Elastic Beanstalk 環境的 Amazon EC2 執行個體組態。

> **Note**
>
> 雖然 Elastic Beanstalk 主控台不提供變更現有環境之處理器架構的選項，但您可以使用. AWS CLI如需指令範例，請參閱使用設定環境的 AWS EC2 執行個體 AWS CLI。

建立環境時透過 Elastic Beanstalk 主控台設定 Amazon EC2 執行個體

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇 Environments (環境)。
3. 選擇 Create a new environment (建立新環境) 以開始建立您的環境。
4. 在精靈的主頁上，在選擇 Create environment (建立環境) 之前，選擇 Configure more options (設定更多選項)。
5. 在 Instances (執行個體) 組態類別中，選擇 Edit (編輯)。變更此類別中的設定，然後選擇 Apply (套用)。如需設定說明，請參閱本頁的the section called “執行個體類別設定”一節。
6. 在 Capacity (容量) 組態類別中，選擇 Edit (編輯)。變更此類別中的設定，然後選擇 Continue (繼續)。如需設定說明，請參閱本頁的the section called “容量類別設定”一節。

> **選取處理器架構**
>
> 向下捲動至 Processor (處理器)，選取 EC2 執行個體的處理器架構。主控台會根據您先前在 Create environment (建立環境) 面板中選擇的平台，列出該平台所支援的處理器架構。如果您沒有看到所需的處理器架構，請返回組態類別清單，選取支援該架構的平台。在 Modify Capacity (修改容量) 面板中選擇 Cancel (取消)。接著，選擇 Change platform version (變更平台版本)，選擇新的平台設定。接下來，在 Capacity (容量) 組態類別中選擇 Edit (編輯)，再次查看處理器架構的各種選擇。



7. 選擇 Save (儲存)，然後針對您環境所需對其他組態做變更。
8. 選擇 Create environment (建立環境)。

在 Elastic Beanstalk 主控台中設定執行環境的 Amazon EC2 執行個體

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

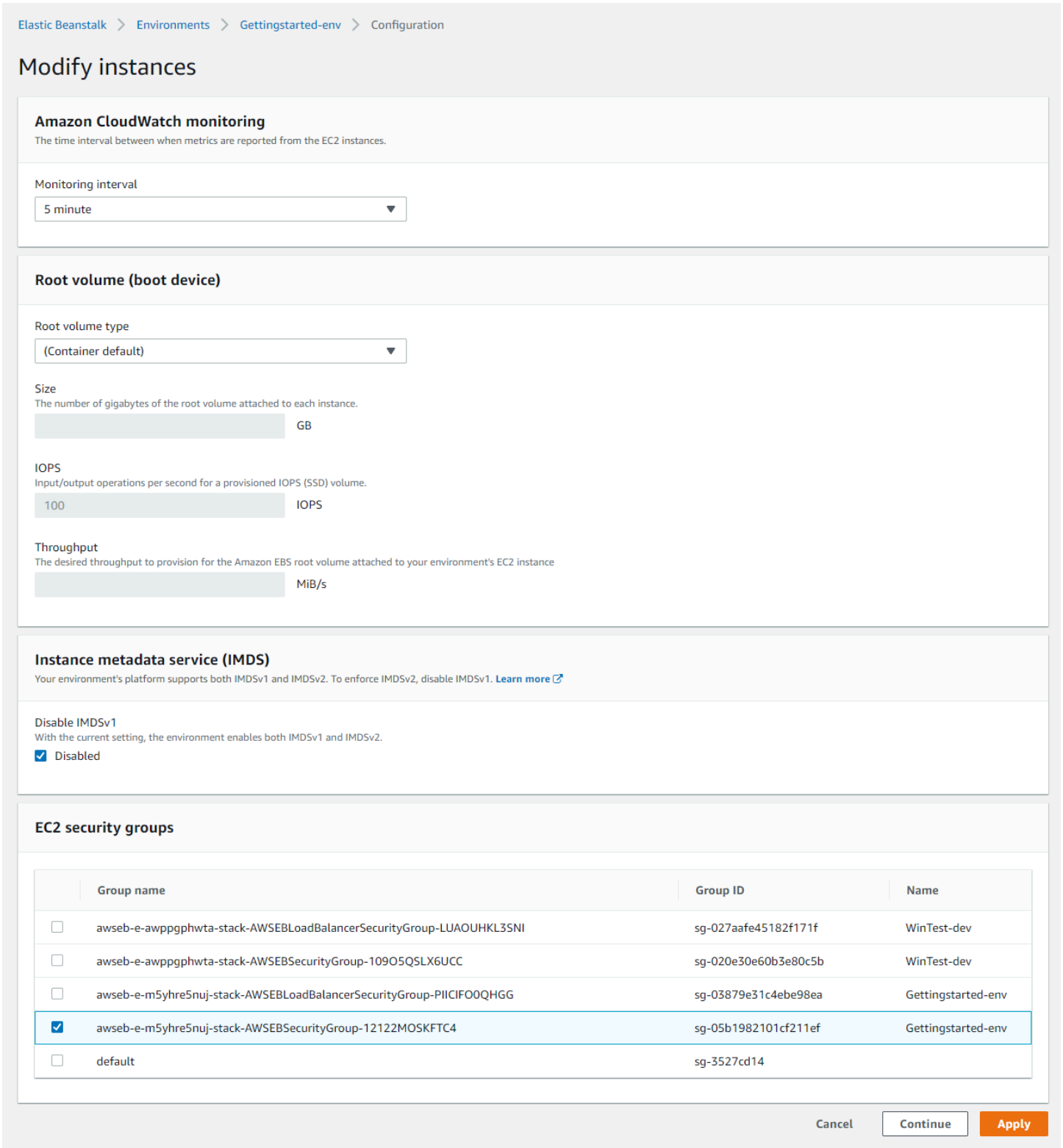
3. 在導覽窗格中，選擇 Configuration (組態)。
4. 在 Instances (執行個體) 組態類別中，選擇 Edit (編輯)。變更此類別中的設定，然後選擇 Apply (套用)。如需設定說明，請參閱本頁的the section called “執行個體類別設定”一節。
5. 在 Capacity (容量) 組態類別中，選擇 Edit (編輯)。變更此類別中的設定，然後選擇 Continue (繼續)。如需設定說明，請參閱本頁的the section called “容量類別設定”一節。

執行個體類別設定

以下與 Amazon EC2 執行個體相關的設定適用於 Instances (執行個體) 組態類別。

選項

- 監控間隔
- 根磁碟區 (開機裝置)
- 執行個體中繼資料服務
- 安全群組


Elastic Beanstalk > Environments > Gettingstarted-env > Configuration
Modify instances
Amazon CloudWatch monitoring
The time interval between when metrics are reported from the EC2 instances.
Monitoring interval
5 minute
Root volume (boot device)
Root volume type
(Container default)
Size
The number of gigabytes of the root volume attached to each instance.
GB
IOPS
Input/output operations per second for a provisioned IOPS (SSD) volume.
100
IOPS
Throughput
The desired throughput to provision for the Amazon EBS root volume attached to your environment's EC2 instance
MiB/s
Instance metadata service (IMDS)
Your environment's platform supports both IMDSv1 and IMDSv2. To enforce IMDSv2, disable IMDSv1. Learn more
Disable IMDSv1
With the current setting, the environment enables both IMDSv1 and IMDSv2.
Disabled
EC2 security groups
Group name
Group ID
Name
awseb-e-awppgphwta-stack-AWSEBLoadBalancerSecurityGroup-LUAOUHKL3SNI
sg-027aafe45182f171f
WinTest-dev
awseb-e-awppgphwta-stack-AWSEBSecurityGroup-109O5QSLX6UCC
sg-020e30e60b3e80c5b
WinTest-dev
awseb-e-m5yhre5nuj-stack-AWSEBLoadBalancerSecurityGroup-PIICIFO0QHGG
sg-03879e31c4ebe98ea
Gettingstarted-env
awseb-e-m5yhre5nuj-stack-AWSEBSecurityGroup-12122MOSKFTC4
sg-05b1982101cf211ef
Gettingstarted-env
default
sg-3527cd14
Cancel
Continue
Apply

監控間隔

根據預設，環境中的執行個體會以五分鐘的間隔將基本運作狀態指標發佈到 Amazon CloudWatch ，無需額外費用。

如需更詳細的報告，您可以將監視間隔設定為 1 分鐘，以增加環境中資源將基本健全狀況指標發佈到 CloudWatch at 的頻率。 CloudWatch 服務費用適用於一分鐘間隔指標。有關更多信息，請參閱 Amazon CloudWatch。

根磁碟區 (開機裝置)

您環境中的每個執行個體均透過根磁碟區進行設定。根磁碟區為連接至執行個體的 Amazon EBS 區塊型儲存設備，以存放作業系統、程式庫、指令碼和您應用程式的原始碼。所有平台預設會使用一般用途的 SSD 區塊型儲存設備。

您可修改根磁碟區類型以使用磁帶儲存或佈建 IOPS SSD 磁碟區類型，並視需要增加磁碟區大小。以佈建 IOPS 磁碟區而言，您亦必須選取欲佈建的 IOPS 數量。輸送量只適用於 gp3 SSD 磁碟區類型。您可以輸入要佈建的所需輸送量，範圍可介於每秒 125 到 1000 MiB (MiB/s) 之間。選取滿足您的效能和價格要求的磁碟區類型。

如需詳細資訊，請參閱 Amazon EC2 使用者指南中的 Amazon EBS 磁碟區類型和 Amazon EBS 產品詳細資訊。

執行個體中繼資料服務

執行個體中繼資料服務 (IMDS) 是一種執行個體上的元件，可供執行個體上的程式碼用來安全地存取執行個體中繼資料。程式碼可以從以下兩種方法擇一使用，從執行中的執行個體存取執行個體中繼資料。這兩種方法分別為執行個體中繼資料服務第 1 版 (IMDSv1) 或執行個體中繼資料服務第 2 版 (IMDSv2)。IMDSv2 更安全。停用 IMDSv1 以強制執行 IMDSv2。如需詳細資訊，請參閱 the section called “IMDS”。

> **Note**
>
> 此組態頁面上的 IMDS 區段只會在支援 IMDSv2 的平台版本中顯示。

安全群組

您的執行個體所連接的安全群組能決定哪些流量可以傳入執行個體，以及哪些流量可以傳出執行個體。Elastic Beanstalk 建立的安全群組，可允許 HTTP (80) 和 HTTPS (443) 標準連接埠上來自負載平衡器的流量。

您可指定其他已建立的安全群組，以允許其他連接埠上或其他來源的流量。例如，您可建立 SSH 存取的安全群組，允許連接埠 22 接收受限 IP 地址範圍所傳入的流量；或者也能建立安全群組，允許僅限您可以存取的堡壘主機所傳送的流量，以提升安全性。

> **Note**
>
> 欲允許環境 A 和 B 執行個體間的流量，您可將規則新增至 Elastic Beanstalk 連接到環境 B 的安全群組。然後您便可指定 Elastic Beanstalk 連接到環境 A 的安全群組，如此即可允許環境 A 執行個體的流量出入。然而，此作法會在兩個安全群組間建立相依性。若您之後嘗試終止環境 A，Elastic Beanstalk 無法刪除環境的安全群組，因為環境 B 的安全群組與其相依。
> 因此，建議您先建立獨立的安全群組，再將安全群組連接至環境 A，並於環境 B 安全群組的規則內指定該安全群組。

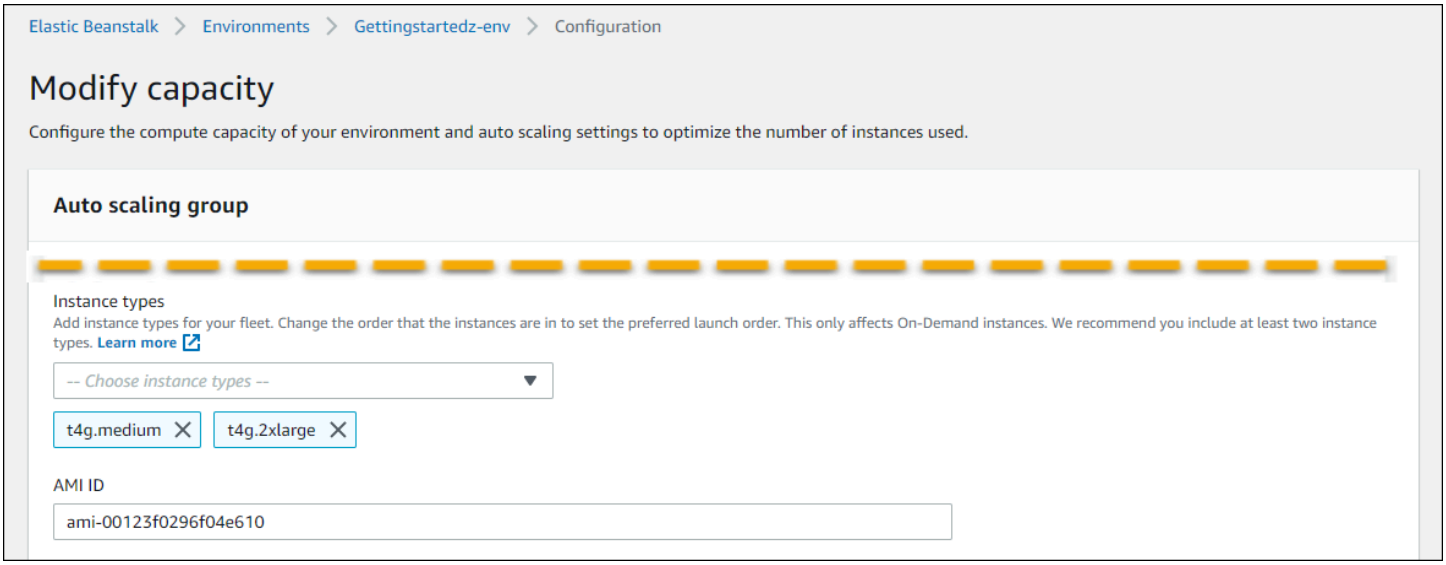
如需有關 Amazon EC2 安全群組的詳細資訊，請參閱 Amazon EC2 使用者指南中的 Amazon EC2 安全群組。

容量類別設定

以下與 Amazon EC2 執行個體相關的設定適用於 Capacity (容量) 組態類別。

選項

- 執行個體類型
- AMI ID



執行個體類型

Instance types (執行個體類型) 設定能決定執行您應用程式所需啟動的 Amazon EC2 執行個體。此組態頁面會顯示 Instance types (執行個體類型) 清單。您可以選取一或多個執行個體類型。Elastic Beanstalk 主控台會根據您為環境所設定的處理器架構，僅顯示相關的執行個體類型，因此您只能新增相同處理器架構的執行個體類型，

> **Note**
>
> 雖然 Elastic Beanstalk 主控台不提供變更現有環境之處理器架構的選項，但您可以使用. AWS CLI如需指令範例，請參閱使用設定環境的 AWS EC2 執行個體 AWS CLI。

所選的執行個體，必須充分配置以執行負載量大的應用程式，但功能無須過大以至於多數時間都處於閒置狀態。若以開發為目的，t2 系列的執行個體可提供中等程度的能力，可於短時間內大幅提升效能。若是大規模、高可用性的應用程式，請使用執行個體集區，以確保單一執行個體故障不會對容量產生太大的影響。首先，請使用可讓您在一般時間中等負載下執行五個執行個體的執行個體類型。若任何執行個體故障，其他執行個體可吸收剩餘的流量。若流量於尖峰時間開始上升，容量緩衝區亦可讓環境有時間擴展。

如需 Amazon EC2 執行個體系列和類型的詳細資訊，請參閱 Amazon EC2 使用者指南中的執行個體類型或 Amazon EC2 使用者指南中的執行個體類型。若要判斷哪些執行個體類型符合您的需求及其支援的區域，請參閱 Amazon EC2 使用者指南中的可用執行個體類型或 Amazon EC2 使用者指南中的可用執行個體類型。

AMI ID

Amazon Machine Image (AMI) 是 Amazon Linux 或 Windows Server 的機器映像，Elastic Beanstalk 會用其啟動您環境中的 Amazon EC2 執行個體。Elastic Beanstalk 提供的機器映像內含執行您應用程式所需的工具和資源。

Elastic Beanstalk 會根據您所選的區域、平台版本和處理器架構，為您的環境選取預設 AMI。若您已建立自訂 AMI，請將預設的 AMI ID 更換成您預設的自訂 AMI ID。

使用設定環境的 AWS EC2 執行個體 AWS CLI

使用命 AWS 令列介面 (AWS CLI)，透過命令列殼層中的指令建立和設定 Elastic Beanstalk 環境。本節會提供 create-environment 和 update-environment 命令的使用範例。

前兩個範例都是建立新環境。該命令會指定以 arm64 處理器架構為基礎的 Amazon EC2 執行個體類型 t4g.small。Elastic Beanstalk 會根據區域、平台版本和執行個體類型，為 EC2 執行個體提供預設的映像 ID (AMI)。執行個體類型會與處理器架構相互對應。`solution-stack-name` 參數會套用至平台版本。

Example 1 — 建立以 arm64 為基礎的新環境 (命名空間選項內嵌)

```
aws elasticbeanstalk create-environment \
--region us-east-1 \
--application-name my-app \
--environment-name my-env \
--solution-stack-name "64bit Amazon Linux 2 v3.4.7 running Docker" \
--option-settings \
Namespace=aws:autoscaling:launchconfiguration,OptionName=IamInstanceProfile,Value=aws-elasticbeanstalk-ec2-role \
Namespace=aws:ec2:instances,OptionName=InstanceTypes,Value=t4g.small
```

或者，您也可以使用 `options.json` 檔案來指定命名空間選項，而非透過內嵌方式一併包含於其中。

Example 2 — 建立以 arm64 為基礎的新環境 (命名空間選項位於 **options.json** 檔案中)

```
aws elasticbeanstalk create-environment \
--region us-east-1 \
--application-name my-app \
--environment-name my-env \
--solution-stack-name "64bit Amazon Linux 2 v3.4.7 running Docker" \
--option-settings file://options.json
```

Example

```
### example options.json ###
[
  {
    "Namespace": "aws:autoscaling:launchconfiguration",
    "OptionName": "IamInstanceProfile",
    "Value": "aws-elasticbeanstalk-ec2-role"
  },
  {
    "Namespace": "aws:ec2:instances",
    "OptionName": "InstanceTypes",
    "Value": "t4g.small"
  }
```

```
]
```

接下來的兩個範例會使用 update-environment 命令，更新現有環境的組態。在這個範例中，我們要新增另一個同樣是以 arm64 處理器架構為基礎的執行個體類型。為現有環境新增的所有執行個體類型都必須具備相同的處理器架構。如果您想要以不同架構的執行個體類型取代現有的執行個體類型，可以這麼做。但請確定命令中的所有執行個體類型都具有相同類型的架構。

Example 3 — 更新以 arm64 為基礎的現有環境 (命名空間選項內嵌)

```
aws elasticbeanstalk update-environment \
--region us-east-1 \
--application-name my-app \
--environment-name my-env \
--solution-stack-name "64bit Amazon Linux 2 v3.4.7 running Docker" \
--option-settings \
Namespace=aws:autoscaling:launchconfiguration,OptionName=IamInstanceProfile,Value=aws-elasticbeanstalk-ec2-role \
Namespace=aws:ec2:instances,OptionName=InstanceTypes,Value=t4g.small,t4g.micro
```

或者，您也可以使用 options.json 檔案來指定命名空間選項，而非透過內嵌方式一併包含於其中。

Example 4 — 更新以 arm64 為基礎的現有環境 (命名空間選項位於 **options.json** 檔案)

```
aws elasticbeanstalk update-environment \
--region us-east-1 \
--application-name my-app \
--environment-name my-env \
--solution-stack-name "64bit Amazon Linux 2 v3.4.7 running Docker" \
--option-settings file://options.json
```

Example

```
### example options.json ###
[
  {
    "Namespace": "aws:autoscaling:launchconfiguration",
    "OptionName": "IamInstanceProfile",
    "Value": "aws-elasticbeanstalk-ec2-role"
  },
  {
```

```
    "Namespace": "aws:ec2:instances",
    "OptionName": "InstanceTypes",
    "Value": "t4g.small, t4g.micro"
  }
]
```

接下來的兩個範例會顯示更多 create-environment 命令，但不會提供 InstanceTypes 的值。若未指定 InstanceTypes 值，Elastic Beanstalk 預設為採用 x86 處理器架構。環境 EC2 執行個體的映像 ID (AMI) 會根據區域、平台版本和預設的執行個體類型而採取不同預設。執行個體類型會與處理器架構相互對應。

Example 5 — 建立以 x86 為基礎的新環境 (命名空間選項內嵌)

```
aws elasticbeanstalk create-environment \
--region us-east-1 \
--application-name my-app \
--environment-name my-env \
--solution-stack-name "64bit Amazon Linux 2 v3.4.7 running Docker" \
--option-settings \
Namespace=aws:autoscaling:launchconfiguration,OptionName=IamInstanceProfile,Value=aws-elasticbeanstalk-ec2-role
```

或者，您也可以使用 options.json 檔案來指定命名空間選項，而非透過內嵌方式一併包含於其中。

Example 6 — 建立以 x86 為基礎的新環境 (命名空間選項位於 **options.json** 檔案中)

```
aws elasticbeanstalk create-environment \
--region us-east-1 \
--application-name my-app \
--environment-name my-env \
--solution-stack-name "64bit Amazon Linux 2 v3.4.7 running Docker" \
--option-settings file://options.json
```

Example

```
### example options.json ###
[
  {
    "Namespace": "aws:autoscaling:launchconfiguration",
    "OptionName": "IamInstanceProfile",
```

```
    "Value": "aws-elasticbeanstalk-ec2-role"
  }
]
```

首波 Graviton arm64 環境的適用建議

> **Note**
>
> 本節資訊僅適用於部分客戶。如果您在 2021 年 11 月 24 日前建立新環境並採用以 Graviton arm64 為基礎的執行個體類型，本節的資訊可能就適合您參考。

首波 Graviton arm64 環境的建議動作

從 2021 年 10 月和 11 月開始，Elastic Beanstalk 開始針對某些區域和平台版本陸續增加 Graviton arm64 處理器的相關支援。本次首波支援已於 2021 年 10 月 13 日、10 月 21 日和 11 月 19 日的AWS Elastic Beanstalk 版本備註中宣布。如果您是建立以 arm64 為基礎的環境，指示會告訴您使用版本備註中提供的自訂 AMI 來設定執行個體。既然現在已針對 Graviton arm64 提供更多支援，Elastic Beanstalk 會在最新的平台版本中，針對 arm64 執行個體類型使用預設的 AMI。

如果您使用首波支援所提供的自訂 AMI 建立環境，建議您執行下列動作，以維持環境健全且正常運作。

1. 從您的環境中移除自訂 AMI。
2. 使用最新的平台版本更新環境。
3. 設定受管平台更新，在排定的維護時段自動升級至平台的最新版本。

> **Note**
>
> Elastic Beanstalk 不會自動更換自訂 AMI。您必須在步驟 1 中刪除自訂 AMI，這樣您在步驟 2 中執行下一項平台更新作業時，平台才會予以更新。

接下來的程序會引導您完成這些步驟。這 AWS CLI 些範例適用於使用下列資訊建立的環境。

```
aws elasticbeanstalk create-environment \
--region us-east-1 \
--application-name my-app \
--environment-name my-env \
```

```
--solution-stack-name "64bit Amazon Linux 2 v3.4.7 running Docker" \
--option-settings \
Namespace=aws:autoscaling:launchconfiguration,OptionName=IamInstanceProfile,Value=aws-elasticbeanstalk-ec2-role \
Namespace=aws:ec2:instances,OptionName=InstanceTypes,Value=t4g.small \
Namespace=aws:autoscaling:launchconfiguration,OptionName=ImageId,Value=ami-0fbdb88ce139244bf
```

更新在首波 Graviton arm64 支援下建立的 arm64 環境

1. 執行 update-environment以移除自訂 AMI 設定。

   ```
   aws elasticbeanstalk update-environment \
   --region us-east-1 \
   --environment-name my-env \
   --options-to-remove \
   Namespace=aws:autoscaling:launchconfiguration,OptionName=ImageId
   ```

2. 使用最新的平台版本更新環境。選擇下列任一選項。

   - 主控台選項 — 使用 Elastic Beanstalk 主控台更新平台版本。如需詳細資訊，請參閱更新您環境的平台版本。
   - AWS CLI 選項：執行 AWS 更新環境命令，指定最近可用的平台版本。

     ```
     aws elasticbeanstalk update-environment \
     --region us-east-1 \
     --environment-name my-env \
     --solution-stack-name "64bit Amazon Linux 2 v3.4.9 running Docker"
     ```

     > **Note**
     >
     > 該list-available-solution-stacks命令提供您在某個 AWS 地區中帳戶可用的平台版本列表。
     >
     > ```
     > aws elasticbeanstalk list-available-solution-stacks --region us-east-1 --query SolutionStacks
     > ```

3. 使用 Elastic Beanstalk 主控台為您的環境設定受管平台更新。受管平台更新可以在排定的維護時段，將您的環境自動升級至最新的平台版本。更新期間，您的應用程式仍能正常提供服務。如需詳細資訊，請參閱受管平台更新。

aws:autoscaling:launchconfiguration 命名空間

您可以在 aws:autoscaling:launchconfiguration 命名空間中使用 configuration options (組態選項) 設定您環境的執行個體，包括主控台尚未提供的其他選項，都能在此使用。

以下組態檔案範例會使用本主題中提及的基本組態選項。例如，其會使用在 IMDS 中討論的 DisableIMDSv1 選項。也會使用在 安全 中討論的 EC2KeyName 和 IamInstanceProfile 選項，以及主控台不提供的 BlockDeviceMappings 選項。

```
option_settings:
  aws:autoscaling:launchconfiguration:
    SecurityGroups: my-securitygroup
    MonitoringInterval: "1 minute"
    DisableIMDSv1: false
    EC2KeyName: my-keypair
    IamInstanceProfile: "aws-elasticbeanstalk-ec2-role"
    BlockDeviceMappings: "/dev/sdj=:100,/dev/sdh=snap-51eef269,/dev/sdb=ephemeral0"
```

您可以使用 BlockDeviceMappings 來為執行個體設定其他區塊型儲存設備。如需詳細資訊，請參閱 Amazon EC2 使用者指南中的區塊裝置對應。

EB CLI 和 Elastic Beanstalk 主控台會為前述選項套用建議的數值。若您想要使用組態檔進行相同的設定，您必須移除這些設定。如需詳細資訊，請參閱「建議值」。

在您的環境執行個體上設定執行個體中繼資料服務

執行個體中繼資料是與 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體相關的資料，可供應用程式用來設定或管理執行中的執行個體。執行個體中繼資料服務 (IMDS) 是一種執行個體上的元件，可供執行個體上的程式碼用來安全地存取執行個體中繼資料。此代碼可以是環境實例上的 Elastic Beanstalk 平台代碼，應用程序可能正在使用的 AWS SDK，甚至是應用程序自己的代碼。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的執行個體中繼資料與使用者資料。

程式碼可使用下列兩種方法之一，從執行中執行個體存取執行個體中繼資料：執行個體中繼資料服務第 1 版 (IMDSv1) 或執行個體中繼資料服務第 2 版 (IMDSv2)。IMDSv2 會使用工作階段導向的請求，並減緩可能用來嘗試存取 IMDS 的幾種漏洞類型。如需這兩種方法的相關資訊，請參閱 Amazon EC2 使用者指南中的設定執行個體中繼資料服務。

章節

- IMDS 的平台支援

- 選擇 IMDS 方法
- 使用 Elastic Beanstalk 主控台來設定 IMDS。
- aws:autoscaling:launchconfiguration 命名空間

IMDS 的平台支援

舊版 Elastic Beanstalk 平台可支援 IMDSv1。新版 Elastic Beanstalk 平台 (所有 Amazon Linux 2 平台版本) 可同時支援 IMDSv1 和 IMDSv2。您可將環境設定為同時支援這兩種方法 (預設值) 或停用 IMDSv1。

> **Note**
>
> 停用 IMDSv1 需要使用 Amazon EC2 啟動範本。當您在環境建立或更新期間設定此類功能時，Elastic Beanstalk 便會嘗試設定您的環境來使用 Amazon EC2 啟動範本 (如果環境尚未使用這些範本)。在此情況下，如果您的使用者政策缺乏必要的許可，環境建立或更新可能會失敗。因此我們建議您使用我們的受管使用者政策，或將所需的許可新增至您的自訂政策。如需所需許可的詳細資訊，請參閱the section called “建立自訂使用者政策”。

選擇 IMDS 方法

在您決定要讓環境支援哪種 IMDS 方法時，請考慮下列使用案例：

- AWS SDK — 如果您的應用程式使用 AWS SDK，請確定您使用的是最新版本的 SDK。開 AWS 發套件會進行 IMDS 呼叫，而較新的 SDK 版本會盡可能使用 IMDSv2。如果您曾停用 IMDSv1，或若您的應用程式使用舊版 SDK，IMDS 呼叫可能會失敗。
- 您的應用程式程式碼 — 如果您的應用程式進行 IMDS 呼叫，請考慮使用 AWS SDK，以便您可以進行呼叫，而不是直接發出 HTTP 要求。如此一來，您就不需要變更程式碼以在 IMDS 方法之間切換。 AWS SDK 會盡可能使用 IMDSv2。
- Elastic Beanstalk 平台代碼 — 我們的代碼通過 AWS SDK 進行 IMDS 調用，因此在所有支持的平台版本上使用 IMDSv2。如果您的程式碼使用 up-to-date AWS SDK 並透過 SDK 進行所有 IMDS 呼叫，您可以安全地停用 IMDSv1。

使用 Elastic Beanstalk 主控台來設定 IMDS。

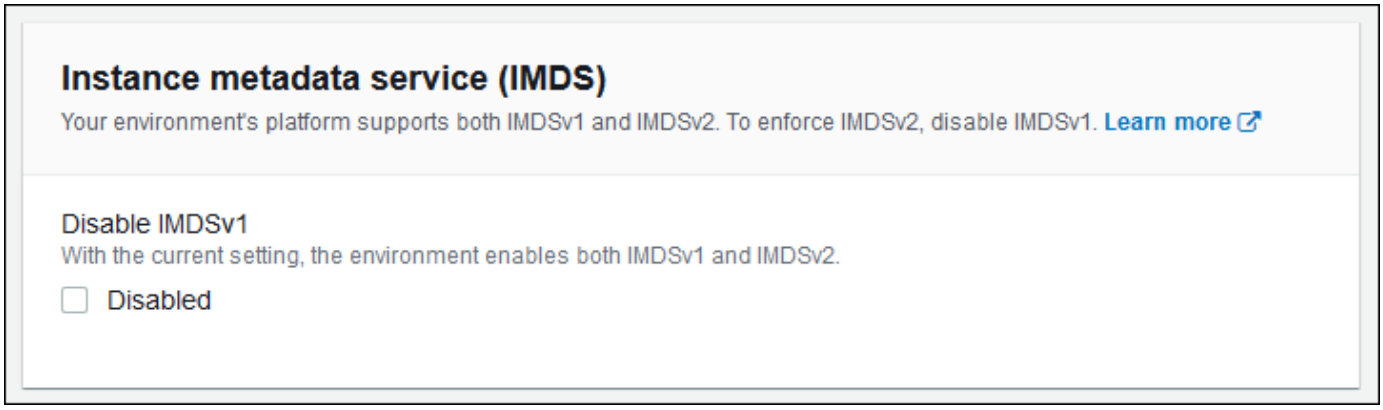
您可以在 Elastic Beanstalk 主控台中修改 Elastic Beanstalk 環境的 Amazon EC2 執行個體組態。

在 Elastic Beanstalk 主控台中，於 Amazon EC2 執行個體上設定 IMDS

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇 Configuration (組態)。
4. 在 Instances (執行個體) 組態類別中，選擇 Edit (編輯)。



5. 設定 Disable IMDSv1 (停用 IMDSv1) 以強制執行 IMDSv2。清除 Disable IMDSv1 (停用 IMDSv1) 以同時啟用 IMDSv1 和 IMDSv2。
6. 若要儲存變更，請選擇頁面底部的儲存變更。

aws:autoscaling:launchconfiguration 命名空間

您可以使用 `aws:autoscaling:launchconfiguration` 命名空間中的組態選項，以在您環境的執行個體上設定 IMDS。

下列組態檔案範例會使用 `DisableIMDSv1` 選項來停用 IMDSv1。

```
option_settings:
  aws:autoscaling:launchconfiguration:
    DisableIMDSv1: true
```

適用於您 Elastic Beanstalk 環境的 Auto Scaling 群組

您的 AWS Elastic Beanstalk 環境包括一個 Auto Scaling 群組，用於管理環境中的 Amazon EC2 執行個體。在單一執行個體環境中，Auto Scaling 群組可確保隨時都有一個執行個體正在執行。在負載平衡的環境中，您可以設定群組欲執行的各種執行個體，而 Auto Scaling 會依據負載視需要新增或移除執行個體。

Auto Scaling 群組也會為您環境中的執行個體套用啟動組態。您可以修改啟動組態來變更執行個體類型、金鑰對、Amazon Elastic Block Store (Amazon EBS) 儲存，以及其他僅能於執行個體啟動時進行的設定。

Auto Scaling 組使用兩個 Amazon CloudWatch 警報來觸發擴展操作。當每個執行個體的平均傳出網路流量，在五分鐘期間高於 6 MiB 或低於 2 MiB 時，預設的觸發條件就會擴展。如要有效地使用 Auto Scaling，請根據您的應用程式、執行個體類型和服務需求，設定適用的觸發。您可以根據多項統計資料來進行擴展，包括延遲、磁碟 I/O、CPU 使用率和請求計數。

如要在可預測的尖峰流量期間最佳化您環境使用 Amazon EC2 執行個體的情況，請設定您的 Auto Scaling 群組來透過排程變更其執行個體計數。您可以排程您群組組態每日或每週的變更，或排程一次性變更以因應將大幅提升至您網站流量的行銷活動。

做為一個選項，Elastic Beanstalk 可以為您的環境結合隨需和 Spot 執行個體。您可以啟用容量重新平衡，以設定 Amazon EC2 Auto Scaling 來監控並自動回應影響 Spot 執行個體可用性的變更。

Auto Scaling 亦會針對其啟動的每個 Amazon EC2 執行個體，監控其運作狀態。如果有任何執行個體未預期終止，Auto Scaling 會偵測到終止狀況，並啟動替代執行個體。如要設定群組使用負載平衡器的運作狀態檢查機制，請參閱Auto Scaling 運作狀態檢查設定。

您可以使用 Elastic Beanstalk 主控台、EB CLI 或組態選項，設定您環境的 Auto Scaling。

主題

Spot 執行個體支援

如要利用 Amazon EC2 Spot 執行個體，您可以為您的環境啟用 Spot 選項。然後您環境的 Auto Scaling 群組會結合 Amazon EC2 購買選項，並維持隨需和 Spot 執行個體的混合。

本主題說明為環境啟用 Spot 執行個體請求的下列方法：

- Elastic Beanstalk 主控台 – 如需詳細資訊，請參閱 the section called “使用 Elastic Beanstalk 主控台設定 Auto Scaling 群組” 中的機群組成。
- EB CLI - 如需詳細資訊，請參閱 the section called “使用 EB CLI 設定 Auto Scaling 群組”。
- `aws:ec2:instances` 命名空間組態選項 – 如需詳細資訊，請參閱 the section called “組態選項”。

> **Important**
>
> Spot 執行個體的需求可能會隨時產生極大的變化，而取決於有多少可用的未使用 Amazon EC2 執行個體，Spot 執行個體的可用性也可能會有顯著的變化。Spot 執行個體隨時都有可能中斷。
>
> 為了協助將這些中斷對應用程式的影響降至最低，您可以啟用 Amazon EC2 Auto Scaling 包含的容量重新平衡選項。啟用此功能後，EC2 會在中斷前自動嘗試取代 Auto Scaling 群組中的 Spot 執行個體。使用 Elastic Beanstalk 主控台設定 Auto Scaling 群組來啟用此功能。您也可以在 aws:autoscaling:asg 命名空間中將 Elastic Beanstalk `EnableCapacityRebalancing` 組態選項設定為 `true`。
>
> 如需詳細資訊，請參閱 Amazon EC2 Auto Scaling 使用者指南中的容量重新平衡和 Amazon EC2 使用者指南中的競價型執行個體中斷。

Elastic Beanstalk 提供支援 Spot 功能的多個組態選項。下列各節將探討設定 Auto Scaling 群組的組態選項。

在 aws:ec2:instances 命名空間，其中兩個選項應特別注意：

- `SpotFleetOnDemandBase`
- `SpotFleetOnDemandAboveBasePercentage`

這兩個選項與 aws:autoscaling:asg 命名空間中的 `MinSize` 選項相互關聯：

- 只有 `MinSize` 會決定您環境的初始容量，亦即您要執行的最少執行個體數量。

- SpotFleetOnDemandBase 不會影響初始容量。啟用 Spot 時，此選項僅會決定在考慮使用 Spot 執行個體前，要佈建的隨需執行個體數量。
- 當 SpotFleetOnDemandBase 小於 MinSize 時考慮。你仍會獲得 MinSize 個執行個體做為初始容量。其中至少 SpotFleetOnDemandBase 個必須是隨需執行個體。
- 當 SpotFleetOnDemandBase 大於 MinSize 時考慮。隨著環境擴展，這可保證您至少能取得相當於兩個值之差的額外執行個體數量。亦即保證在達到 SpotFleetOnDemandBase 個的數量要求前，您至少能取得 (SpotFleetOnDemandBase - MinSize) 個額外的隨需執行個體。

在生產環境中，做為可擴展且有負載平衡環境中一部分的 Spot 執行個體格外實用。不建議您將 Spot 用於單一執行個體環境。如果沒有可用的 Spot 執行個體，您可能會失去環境的整個容量 (單一執行個體)。您可能仍會想要將 Spot 執行個體用於單一執行個體環境，以進行開發或測試。當您這麼做時，請務必將 SpotFleetOnDemandBase 與 SpotFleetOnDemandAboveBasePercentage 同時設為零。任何其他設定都會產生隨選執行個體。

> **備註**
>
> - 某些較舊的 AWS 帳戶可能會提供 Elastic Beanstalk 以及不支援競價型執行個體 (例如 t1.micro) 的預設執行個體類型。如果您啟用了 Spot 執行個體請求，並且看到 None of the instance types you specified supports Spot (您指定的執行個體類型皆不支援 Spot) 錯誤，請務必設定可支援 Spot 的執行個體類型。如要選擇 Spot 執行個體類型，請使用 Spot Instance Advisor。
> - 啟用 Spot 執行個體請求需要使用 Amazon EC2 啟動範本。當您在環境建立或更新期間設定此類功能，Elastic Beanstalk 便會嘗試設定您的環境來使用 Amazon EC2 啟動範本 (如果環境尚未使用這些範本)。在此情況下，如果您的使用者政策缺乏必要的許可，環境建立或更新可能會失敗。因此我們建議您使用我們的受管使用者政策，或將所需的許可新增至您的自訂政策。如需所需許可的詳細資訊，請參閱the section called “建立自訂使用者政策”。

以下範例示範設定各種擴展選項的不同情況。所有範例均假設使用已啟用 Spot 執行個體請求的負載平衡環境。

Example 1：做為初始容量一部分的隨需和 Spot

選項設定

選項	命名空間	Value
MinSize	aws:autoscaling:asg	10
MaxSize	aws:autoscaling:asg	24
SpotFleetOnDemandBase	aws:ec2:instances	4
SpotFleetOnDemandAboveBasePercentage	aws:ec2:instances	50

在此範例中，環境是從十個執行個體開始，其中七個是隨需 (四個是基本數量，50% 的六個高於基本數量)，其中三個是 Spot。環境最多可擴展到 24 個執行個體。隨著環境擴展，高於四個基本數量隨需執行個體機群部分中的隨需部分會保持在 50%，整體最多 24 個執行個體，其中 14 個是隨需 (四個基本數量，50% 的 20 個高於基本數量)，其中十個是 Spot。

Example 2：所有隨需初始容量

選項設定

選項	命名空間	Value
MinSize	aws:autoscaling:asg	4
MaxSize	aws:autoscaling:asg	24
SpotFleetOnDemandBase	aws:ec2:instances	4
SpotFleetOnDemandAboveBasePercentage	aws:ec2:instances	50

在此範例中，環境是從四個執行個體開始，全部都是隨需。環境最多可擴展到 24 個執行個體。隨著環境擴展，高於四個基本數量隨需執行個體機群部分中的隨需部分會保持在 50%，整體最多 24 個執行個體，其中 14 個是隨需 (四個基本數量，50% 的 20 個高於基本數量)，其中十個是 Spot。

Example 3：超過初始容量的額外隨需基本數量

選項設定

選項	命名空間	Value
MinSize	aws:autoscaling:asg	3
MaxSize	aws:autoscaling:asg	24
SpotFleetOnDemandBase	aws:ec2:instances	4
SpotFleetOnDemandAboveBasePercentage	aws:ec2:instances	50

在此範例中，環境是從三個執行個體開始，全部都是隨需。環境最多可擴展到 24 個執行個體。超過初始三個的第一個額外執行個體是隨需，可讓四個基本數量隨需執行個體變得完整。隨著環境進一步擴展，高於四個基本數量隨需執行個體機群部分中的隨需部分會保持在 50%，整體最多 24 個執行個體，其中 14 個是隨需 (四個基本數量，50% 的 20 個高於基本數量)，其中十個是 Spot。

使用 Elastic Beanstalk 主控台設定 Auto Scaling 群組

您可以在 Elastic Beanstalk 主控台中，於環境的 Configuration (組態) 頁面上編輯 Capacity (容量)，以設定 Auto Scaling 的運作方式。

在 Elastic Beanstalk 主控台中設定 Auto Scaling 群組

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇 Configuration (組態)。
4. 在 Capacity (容量) 組態類別中，選擇 Edit (編輯)。
5. 在 Auto Scaling group (Auto Scaling 群組) 區段，設定下列設定。

- Environment type (環境類型) – 選取 Load balanced (負載平衡)。
- Min instances (最少執行個體) – 該群組應隨時包含的 EC2 執行個體數量下限。群組以計數下限開始，當滿足擴展觸發條件時，將新增執行個體。
- Max instances (最大執行個體) – 該群組應隨時包含的 EC2 執行個體數量上限。

> **Note**
>
> 若您採用滾動更新，請確保執行個體計數上限高於滾動更新的 Minimum instances in service (服務中執行個體數量下限) 設定。

- 機群組成 – 預設為隨需執行個體。選取 Combined purchase options and instances (組合的購買選項和執行個體) 以啟用 Spot Instance (Spot 執行個體) 請求。

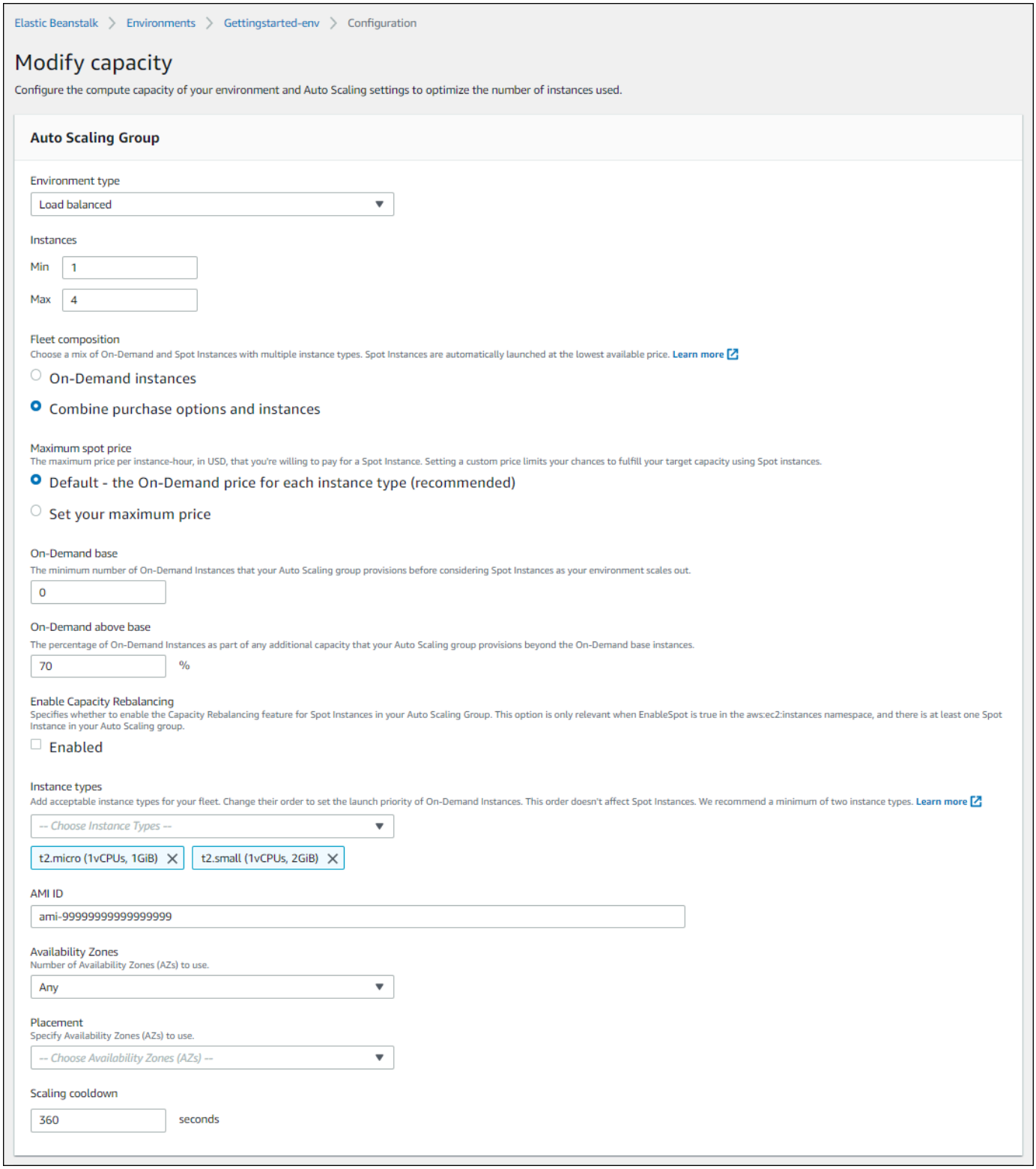
  如果您選取啟用 Spot 執行個體 (Spot 執行個體) 請求，便會啟用下列選項：

  - 最高現貨價格 — 有關競價型執行個體最高價格選項的建議，請參閱 Amazon EC2 使用者指南中的競價型執行個體定價歷史記錄。
  - On-Demand base (隨需執行個體基本數量) – 隨著環境擴展，在考慮使用 Spot 執行個體前，Auto Scaling 群組要佈建的最小隨需執行個體數量。
  - On-Demand above base (超過基本數量的隨需執行個體) – 隨需執行個體百分比 (Auto Scaling 群組超出隨需執行個體基本數量所佈建的一部分額外容量)。

    > **Note**
    >
    > On-Demand base (隨需執行個體基本數量) 和 On-Demand above base (超過基本數量的隨需執行個體) 選項與前面列出的執行個體 Min (下限) 和 Max (上限) 選項關聯。如需有關這些選項和範例的詳細資訊，請參閱 the section called “Spot 執行個體支援”。

  - Enable Capacity Rebalancing (啟用容量重新平衡) - 此選項只有 Auto Scaling 群組中至少有一個 Spot 執行個體時有用。啟用此功能後，EC2 會在 Auto Scaling 群組中的 Spot 執行個體中斷前，自動嘗試取代，將 Spot 執行個體中斷對應用程式的影響降至最低。如需詳細資訊，請參閱 Amazon EC2 Auto Scaling 使用者指南中的容量重新平衡。
- Instance type (執行個體類型) – 啟動的 Amazon EC2 執行個體類型，用以執行您應用程式。如需詳細資訊，請參閱 the section called “執行個體類型”。
- AMI ID – Elastic Beanstalk 用來在您環境中啟動 Amazon EC2 執行個體的機器映像。如需詳細資訊，請參閱 the section called “AMI ID”。

- Availability Zones (可用區域) – 選擇您環境執行個體可散佈到的可用區域數。根據預設，Auto Scaling 群組會在所有可用區域中平均啟動執行個體。欲將您的執行個體集中於較少區域，請選擇欲使用的區域數量。以生產環境而言，請使用至少兩個區域，以確保您的應用程式在其中一個可用區域服務中斷時仍能使用。
- Placement (置放) (選用) – 選擇欲使用的可用區域。如果若您的執行個體需要連線至特定區域的資源，或者您已購買區域特定的預留執行個體，請使用此設定。若您於自訂 VPC 啟動環境，您無法設定此選項。於自訂 VPC 中，您可以針對指派給您環境的子網路，選擇其可用區域。
- Scaling cooldown (擴展冷卻) – 擴展之後，在繼續評估觸發前等待執行個體啟動或終止的時間 (以秒計)。如需詳細資訊，請參閱擴展冷卻。



6. 若要儲存變更，請選擇頁面底部的儲存變更。

使用 EB CLI 設定 Auto Scaling 群組

使用 eb create 命令建立環境時，您可以指定與您環境 Auto Scaling 群組相關的幾個選項。這些是可協助您控制環境的容量的其中一些選項。

`--single`

建立有一個 Amazon EC2 執行個體且沒有負載平衡器的環境。如果您未使用此選項，系統會將負載平衡器新增至建立的環境環境。

`--enable-spot`

為您的環境啟用 Spot 執行個體請求。

eb create 命令的下列選項僅可搭配 `--enable-spot` 使用。

`--instance-types`

列出您要讓環境使用的 Amazon EC2 執行個體類型。

`--spot-max-price`

您願意為 Spot 執行個體支付的每單位小時最高價格 (以美元為單位)。如需競價型執行個體最高價格選項的建議，請參閱 Amazon EC2 使用者指南中的競價型執行個體定價歷史記錄。

`--on-demand-base-capacity`

隨著環境擴展，Auto Scaling 群組在考量 Spot 執行個體前佈建的最小隨需執行個體數量。

`--on-demand-above-base-capacity`

隨需執行個體百分比 (Auto Scaling 群組超出 `--on-demand-base-capacity` 選項所指定之執行個體數量所佈建的一部分額外容量)。

以下範例會建立環境並將 Auto Scaling 群組設為針對新環境啟用 Spot 執行個體請求。以這個例子來說，有三個執行個體類型可使用。

```
$ eb create --enable-spot --instance-types "t2.micro,t3.micro,t3.small"
```

> **Important**
>
> 還有另一個名稱類似的選項 `--instance-type` (沒有“s”)，EB CLI 僅在處理隨需執行個體時才會區分這類選項名稱。請勿將 `--instance-type` (沒有 "s") 與 `--enable-spot` 選項搭配

使用。如果搭配使用，EB CLI 會加以忽略。反之，請將 `--instance-types` (有 "s") 與 `--enable-spot` 選項搭配使用。

組態選項

Elastic Beanstalk 在以下兩個命名空間中提供 Auto Scaling 設定的組態選項：`aws:autoscaling:asg` 和 `aws:ec2:instances`。

aws:autoscaling:asg 命名空間

`aws:autoscaling:asg` 命名空間提供了可用於整體擴展和可用性的選項。

下列組態檔案範例會設定 Auto Scaling 群組，以使用二到四個執行個體、特定可用區域，以及 12 分鐘 (720 秒) 的冷卻時間。已啟用 Spot 執行個體的容量重新平衡。如此範例後面的組態檔案範例所示，最後一個選項只會在 `EnableSpot` 於 `aws:ec2:instances` 命名空間中設定為 `true` 時生效。

```
option_settings:
  aws:autoscaling:asg:
    Availability Zones: Any
    Cooldown: '720'
    Custom Availability Zones: 'us-west-2a,us-west-2b'
    MaxSize: '4'
    MinSize: '2'
    EnableCapacityRebalancing: true
```

aws:ec2:instances 命名空間

`aws:ec2:instances` 命名空間提供了與環境的執行個體相關的選項，包括 Spot 執行個體管理。這與 `aws:autoscaling:launchconfiguration` 及 `aws:autoscaling:asg` 相輔相成。

當您更新環境資訊並從 `InstanceTypes` 選項中移除一或多個執行個體類型時，Elastic Beanstalk 會終止在已移除的執行個體類型上執行的任何 Amazon EC2 執行個體。然後，您環境的 Auto Scaling 群組會視需要啟動新執行個體，以使用目前指定的執行個體類型來完成所需的容量。

以下組態檔案範例會設定 Auto Scaling 群組，為您的環境啟用 Spot 執行個體請求。有三個可能的執行個體類型可使用。至少將一個隨需執行個體用於基準容量，以及將認可的 33% 隨需執行個體用於額外容量。

```
option_settings:
```

```
  aws:ec2:instances:
    EnableSpot: true
    InstanceTypes: 't2.micro,t3.micro,t3.small'
    SpotFleetOnDemandBase: '1'
    SpotFleetOnDemandAboveBasePercentage: '33'
```

如要選擇 Spot 執行個體類型，請使用 Spot Instance Advisor。

Auto Scaling 觸發程式

您 Elastic Beanstalk 環境中的 Auto Scaling 群組會使用兩個 Amazon CloudWatch 警示觸發擴展操作。當每個執行個體的平均傳出網路流量，在五分鐘期間高於 6 MB 或低於 2 MB 時，預設的觸發條件就會擴展。如要有效地使用 Amazon EC2 Auto Scaling，請根據您的應用程式、執行個體類型和服務需求，設定適用的觸發。您可以根據多項統計資料來進行擴展，包括延遲、磁碟 I/O、CPU 使用率和請求計數。

如需 CloudWatch 指標和警示的詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的 Amazon CloudWatch 概念。

設定 Auto Scaling 觸發

您可以在 Elastic Beanstalk 主控台中設定觸發，來調整您環境 Auto Scaling 群組中的執行個體數量。

在 Elastic Beanstalk 主控台中設定觸發

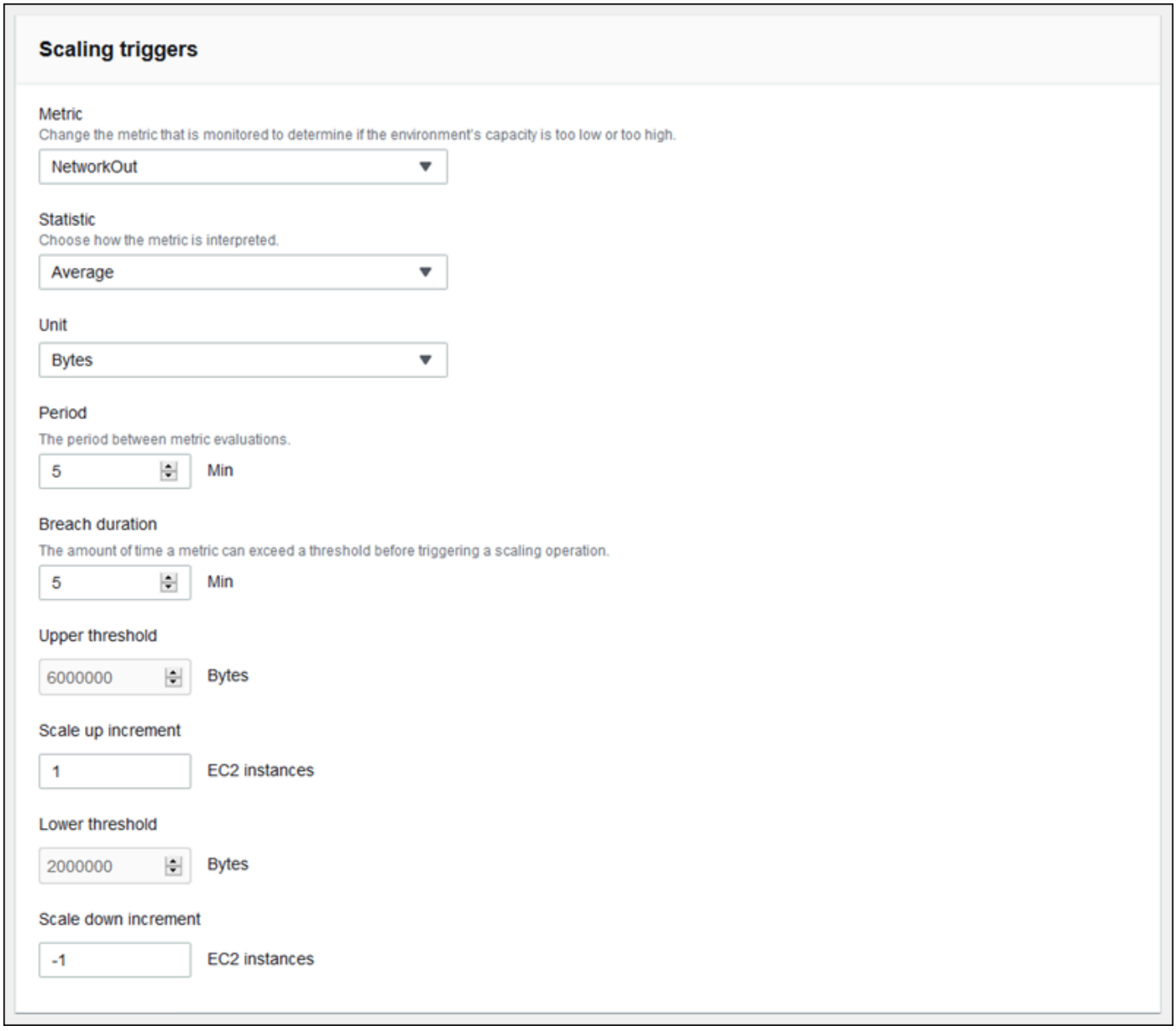
1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

   > **Note**
   >
   > 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇 Configuration (組態)。
4. 在 Capacity (容量) 組態類別中，選擇 Edit (編輯)。
5. 在 Scaling triggers (擴展觸發) 區段，設定以下設定：

   - Metric (指標) – 用於 Auto Scaling 觸發的指標。
   - Statistic (統計) – 觸發條件應使用的統計資料，例如 `Average`。
   - Unit (單位) – 觸發指標的單位，例如 Bytes (位元組)。

- Period (期間) – 為您的觸發指定 Amazon CloudWatch 衡量指標的頻率。
- Breach duration (違規持續時間) – 在觸發擴展操作之前，指標可超出閥值上限和下限的時間長度 (以分鐘為單位)。
- Upper threshold (閥值上限) – 如果指標超過此數字達違規持續時間，則會觸發擴展操作。
- Scale up increment (規模調增) – 在進行擴展活動時要新增的 Amazon EC2 執行個體數。
- Lower threshold (閥值下限) – 如果指標低於此數字達違規持續時間，則會觸發擴展操作。
- Scale down increment (規模調減) – 在進行擴展活動時要移除的 Amazon EC2 執行個體數。



6. 若要儲存變更，請選擇頁面底部的儲存變更。

aws:autoscaling:trigger 命名空間

Elastic Beanstalk 會在 aws:autoscaling:trigger 命名空間中提供 Auto Scaling 設定的組態選項。此命名空間中的設定，會根據其所套用的資源來編排。

```
option_settings:
  AWSEBAutoScalingScaleDownPolicy.aws:autoscaling:trigger:
    LowerBreachScaleIncrement: '-1'
  AWSEBAutoScalingScaleUpPolicy.aws:autoscaling:trigger:
    UpperBreachScaleIncrement: '1'
  AWSEBCloudwatchAlarmHigh.aws:autoscaling:trigger:
    UpperThreshold: '6000000'
  AWSEBCloudwatchAlarmLow.aws:autoscaling:trigger:
    BreachDuration: '5'
    EvaluationPeriods: '1'
    LowerThreshold: '2000000'
    MeasureName: NetworkOut
    Period: '5'
    Statistic: Average
    Unit: Bytes
```

排程的 Auto Scaling 動作

如要在可預測的尖峰流量期間最佳化您環境使用 Amazon EC2 執行個體的情況，請設定您的 Amazon EC2 Auto Scaling 群組來透過排程變更其執行個體計數。針對具備重複動作的環境，您可將其設定為每天上午進行擴展，並在夜間低流量時進行縮減。舉例來說，若您正進行的行銷活動，將在特定期間為您的網站提升流量，您可將一次性的擴展與縮減事件分別安排在活動開始與結束的時候。

每個環境至多可定義 120 個作用中的排定動作。Elastic Beanstalk 也會保留至多 150 個過期的動作，讓您可以重複使用以更新其設定。

設定排定的動作

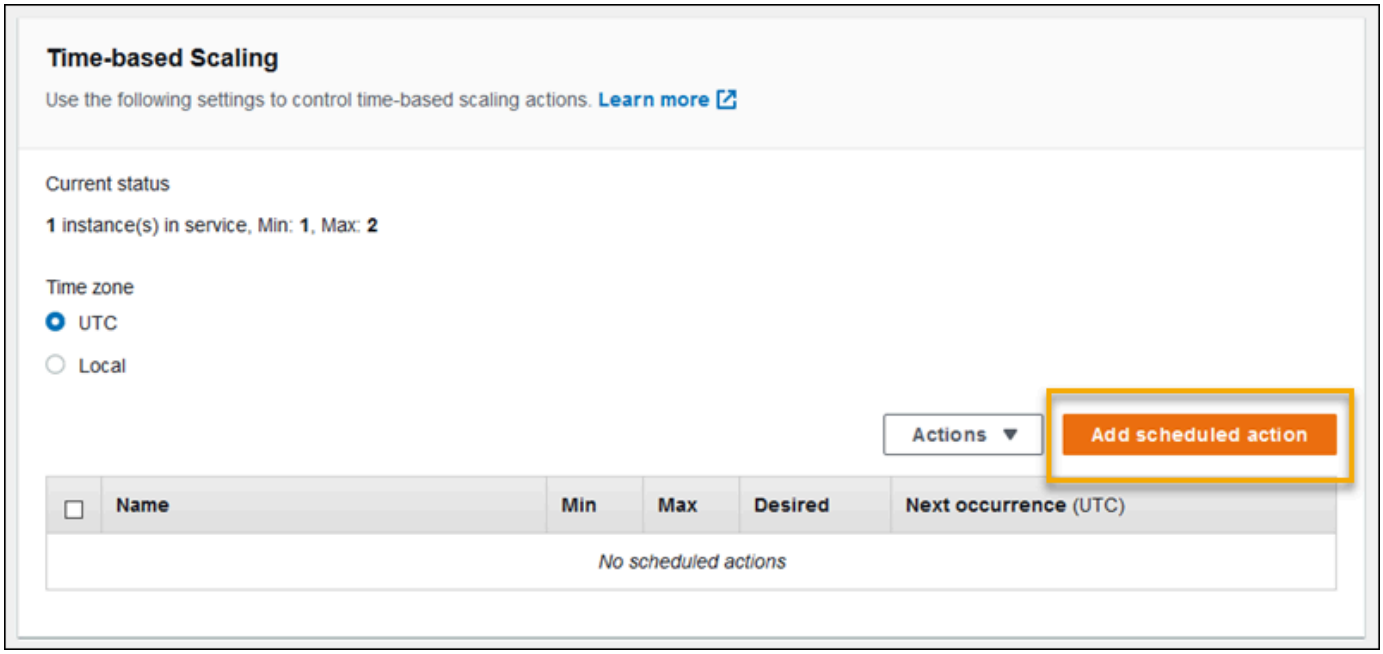
您可以在 Elastic Beanstalk 主控台中為您環境的 Auto Scaling 群組建立排程動作。

在 Elastic Beanstalk 主控台中設定排程動作

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇 Configuration (組態)。
4. 在 Capacity (容量) 組態類別中，選擇 Edit (編輯)。
5. 在以 Time-based scaling (時間型擴展) 區段中，請選擇 Add scheduled action (新增排定的動作)。



6. 請填寫以下排定動作設定：

   - Name (名稱) – 指定唯一的名稱，至多 255 個英數字元且不含空格。
   - Instances (執行個體) – 選擇欲套用至 Auto Scaling 群組的執行個體計數上下限。
   - Desired capacity (所需的容量) (選用) – 設定 Auto Scaling 群組所需的初始容量。在套用排程動作後，觸發會依它們的設定調整所需的容量。
   - Occurrence (出現) – 選擇 Recurring (經常性) 來重複排程上的擴展動作。
   - Start time (開始時間) – 針對一次性動作，選擇執行動作的日期和時間。

     對於重複動作，開始時間是選擇性的。指定此選項以選擇最早執行動作的時間。在此時間之後，動作會根據 Recurrence (重複) 表達式重複執行。
   - Recurrence (週期) – 使用 Cron 表達式指定您希望排程動作發生的頻率。例如，`30 6 * * 2` 會在 UTC 每週二上午 6:30 執行動作。

- End time (結束時間) (選擇性) – 重複動作的選項。如果指定此選項，動作會根據 Recurrence (重複) 表達式重複執行，並在此時間之後不再執行。

  當排程的動作結束時，Auto Scaling 不會自動回到之前的設定。設定第二個排程動作，並視需要使 Auto Scaling 返回原始設定。

7. 選擇 Add (新增)。
8. 若要儲存變更，請選擇頁面底部的儲存變更。

> **Note**
>
> 要在套用後才會儲存排程的動作。

aws:autoscaling:scheduledaction 命名空間

若您需要設定大量的排程動作，可使用組態檔案或 Elastic Beanstalk API，套用來自 YAML 或 JSON 檔案的組態選項變更。這些方法亦可讓您存取 Suspend 選項，暫時停用排定的重複動作。

> **Note**
>
> 在主控台外使用排定的動作組態選項時，請使用 ISO 8601 時間格式，以 UTC 時間指定開始時間和結束時間。例如：2015-04-28T04:07:02Z。如需 ISO 8601 時間格式的詳細資訊，請參閱 Date and Time Formats。日期在所有排定的動作都必須是獨一無二的。

Elastic Beanstalk 在 `aws:autoscaling:scheduledaction` 命名空間中，提供了排程動作設定適用的組態選項。使用 `resource_name` 欄位來指定排定的動作名稱。

Example Scheduled-scale-up-specific-time-long.config

此組態檔案會指示 Elastic Beanstalk 在 2015-12-12T00:00:00Z，從五個執行個體擴展為 10 個執行個體。

```
option_settings:
  - namespace: aws:autoscaling:scheduledaction
    resource_name: ScheduledScaleUpSpecificTime
    option_name: MinSize
    value: '5'
  - namespace: aws:autoscaling:scheduledaction
```

```
    resource_name: ScheduledScaleUpSpecificTime
    option_name: MaxSize
    value: '10'
  - namespace: aws:autoscaling:scheduledaction
    resource_name: ScheduledScaleUpSpecificTime
    option_name: DesiredCapacity
    value: '5'
  - namespace: aws:autoscaling:scheduledaction
    resource_name: ScheduledScaleUpSpecificTime
    option_name: StartTime
    value: '2015-12-12T00:00:00Z'
```

Example Scheduled-scale-up-specific-time.config

欲搭配 EB CLI 或組態檔案使用速記語法，請在命名空間前方加上資源名稱。

```
option_settings:
  ScheduledScaleUpSpecificTime.aws:autoscaling:scheduledaction:
    MinSize: '5'
    MaxSize: '10'
    DesiredCapacity: '5'
    StartTime: '2015-12-12T00:00:00Z'
```

Example Scheduled-scale-down-specific-time.config

此組態檔案會指示 Elastic Beanstalk 在 2015-12-12T07:00:00Z 進行擴展。

```
option_settings:
  ScheduledScaleDownSpecificTime.aws:autoscaling:scheduledaction:
    MinSize: '1'
    MaxSize: '1'
    DesiredCapacity: '1'
    StartTime: '2015-12-12T07:00:00Z'
```

Example Scheduled-periodic-scale-up.config

此組態檔案會指示 Elastic Beanstalk 每天上午 9 點進行擴展。此排定的動作將於 2015 年 5 月 14 日開始，並在 2016 年 1 月 12 日結束。

```
option_settings:
  ScheduledPeriodicScaleUp.aws:autoscaling:scheduledaction:
    MinSize: '5'
```

```
    MaxSize: '10'
    DesiredCapacity: '5'
    StartTime: '2015-05-14T07:00:00Z'
    EndTime: '2016-01-12T07:00:00Z'
    Recurrence: 0 9 * * *
```

Example Scheduled-periodic-scale-down.config

此組態檔案會指示 Elastic Beanstalk 每天下午 6 點進行擴展至沒有任何運作中的執行個體。如果您知道您的應用程式在上班時間外都在閒置，您可以建立一個類似的排程動作。如果您的應用程式在上班時間外必須關閉，請將 `MaxSize` 變更為 `0`。

```
option_settings:
  ScheduledPeriodicScaleDown.aws:autoscaling:scheduledaction:
    MinSize: '0'
    MaxSize: '1'
    DesiredCapacity: '0'
    StartTime: '2015-05-14T07:00:00Z'
    EndTime: '2016-01-12T07:00:00Z'
    Recurrence: 0 18 * * *
```

Example Scheduled-weekend-scale-down.config

此組態檔案會指示 Elastic Beanstalk 每個星期五下午 6 點進行擴展。若您知道應用程式週末接收的流量較少，您可建立類似的排定動作。

```
option_settings:
  ScheduledWeekendScaleDown.aws:autoscaling:scheduledaction:
    MinSize: '1'
    MaxSize: '4'
    DesiredCapacity: '1'
    StartTime: '2015-12-12T07:00:00Z'
    EndTime: '2016-01-12T07:00:00Z'
    Recurrence: 0 18 * * 5
```

Auto Scaling 運作狀態檢查設定

Amazon EC2 Auto Scaling 會針對其啟動的每個 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體監控其運作狀態。如果有任何執行個體未預期終止，Auto Scaling 會偵測到終止狀況，並啟動替代執行個體。根據預設，針對您的環境所建立的 Auto Scaling 群組，會使用 Amazon EC2 狀態檢查功能。如果您環境中的執行個體未通過 Amazon EC2 狀態檢查，Auto Scaling 會將其撤下並進行替換。

Amazon EC2 狀態檢查僅涵蓋執行個體的運作狀態，而未針對您的應用程式、伺服器或執行個體上所執行的任何 Docker 容器，來檢查其運作狀態。如果您的應用程式當機，但其上所執行的執行個體仍然運作狀態良好，則此執行個體可能會被移出負載平衡器，但 Auto Scaling 不會自動替換該項目。預設的處理動作有利於故障排除作業的進行。即使應用程式在啟動後很快地就當機，但如果應用程式一當機，Auto Scaling 就替換了執行個體，則您可能不會知道發生了錯誤。

如果您希望 Auto Scaling 在執行個體的應用程式停止回應時，就取代這些執行個體，可以利用組態檔案來設定 Auto Scaling 群組使用 Elastic Load Balancing 運作狀態檢查功能。下列範例會設定群組，使用負載平衡器的運作狀態檢查，以及 Amazon EC2 狀態檢查功能，來判斷執行個體的運作狀態。

Example .ebextensions/autoscaling.config

```
Resources:
  AWSEBAutoScalingGroup:
    Type: "AWS::AutoScaling::AutoScalingGroup"
    Properties:
      HealthCheckType: ELB
      HealthCheckGracePeriod: 300
```

如需 `HealthCheckType` 和 `HealthCheckGracePeriod` 屬性的詳細資訊，請參閱《AWS CloudFormation 使用者指南》中的 AWS::AutoScaling::AutoScalingGroup，以及《Amazon EC2 Auto Scaling 使用者指南》中的 Auto Scaling 執行個體運作狀態檢查。

根據預設，Elastic Load Balancing 運作狀態檢查會設為透過連接埠 80，使用 TCP 來嘗試連線到您的執行個體。這樣可確認在執行個體上執行的 web 伺服器正在接受連線。不過，您可能想讓 自訂負載平衡器的運作狀態檢查，以確保您的應用程式與 web 伺服器都處於良好的狀態。這項寬限期間的設定，指定了執行個體在受到運作狀態檢查時可以失敗的秒數，未超過此秒數就不會遭到終止和替換。執行個體在被移出負載平衡器後可以回復，所以請預留適合您應用程式的轉換時間給執行個體。

您的 Elastic Beanstalk 環境的負載平衡器

負載平衡器會在您的環境執行個體之間分配流量。啟用負載平衡時，AWS Elastic Beanstalk 會建立專用於您環境的 Elastic Load Balancing 負載平衡器。Elastic Beanstalk 會完全管理此負載平衡器，處理安全設定，並在終止環境時終止負載平衡器。

或者，您可以選擇在多個 Elastic Beanstalk 環境中共享負載平衡器。透過共享負載平衡器，您可以避免每個環境的專用負載平衡器來節省營運成本。您也會承擔更多環境所使用之共享負載平衡器的管理責任。

Elastic Load Balancing 有下列負載平衡器類型：

- Classic Load Balancer - 上一代負載平衡器。路由傳送 HTTP、HTTPS 或 TCP 請求流量到不同的連接埠環境執行個體。
- Application Load Balancer - 應用程式層負載平衡器。路由傳送 HTTP 或 HTTPS 根據請求路徑來請求流量到不同的連接埠環境執行個體。
- Network Load Balancer - 網路層負載平衡器。路由傳送 TCP 請求流量到不同的連接埠環境執行個體。支援使用中和被動式的運作狀態檢查。

Elastic Beanstalk 支援所有三種負載平衡器類型。下表顯示您可以搭配兩種使用模式使用的類型：

負載平衡器類型	專用型	共同
Classic Load Balancer	✓ 是	X 否
Application Load Balancer	✓ 是	✓ 是
Network Load Balancer	✓ 是	X 否

> **Note**
>
> 建立環境主控台精靈上的 Classic Load Balancer (CLB) 選項已停用。如果您已使用 Classic Load Balancer 設定現有環境，則可以使用 Elastic Beanstalk 主控台或 EB CLI 來 複製現有環境以建立新的環境。您也可以選擇使用 EB CLI 或 AWS CLI 來建立使用 Classic Load Balancer 設定的新環境。這些命令列工具會建立具有 CLB 的新環境，即使您的帳戶中沒有環境存在亦然。

在預設情況下，當您使用 Elastic Beanstalk 主控台或 EB CLI 啟用負載平衡時，Elastic Beanstalk 會為您的環境建立 Application Load Balancer。您的負載平衡器會設定為監聽 80 埠上的 HTTP 傳輸資料，並將此流量轉傳給位於同一個通訊埠上的執行個體。您可以選擇您的環境只會在環境建立時使用的負載平衡器類型。之後您可以變更設定，以管理執行之環境的負載平衡器行為。

> **Note**
>
> 您環境所在的 VPC 必須至少在兩個可用區域擁有子網路，如此才能建立 Application Load Balancer。所有新的 AWS 帳戶皆包含符合此需求的預設 VPC。

請參閱下列主題以了解 Elastic Beanstalk 支援的每個負載平衡器類型及其功能，如何在 Elastic Beanstalk 環境中進行設定和管理，以及如何設定負載平衡器以將存取日誌上傳到 Amazon S3。

主題

設定 Classic Load Balancer

若啟用負載平衡，您的 AWS Elastic Beanstalk 環境會配備 Elastic Load Balancing 負載平衡器，將流量分配到您環境中的執行個體。Elastic Load Balancing 支援多種類型的負載平衡器。若要了解這些資訊，請參閱 Elastic Load Balancing 使用者指南。Elastic Beanstalk 可以為您建立負載平衡器，或讓您指定已建立的共享負載平衡器。

本主題說明 Elastic Beanstalk 建立並專用於您環境的Classic Load Balancer 的組態。如需設定 Elastic Beanstalk 支援之所有負載平衡器類型的詳細資訊，請參閱您的 Elastic Beanstalk 環境的負載平衡器。

> **Note**
>
> 您可以選擇您的環境只會在環境建立時使用的負載平衡器類型。之後您可以變更設定，以管理執行之環境的負載平衡器行為。

簡介

Classic Load Balancer 是 Elastic Load Balancing 上一代的負載平衡器。其支援路由傳送 HTTP、HTTPS 或 TCP 請求流量到不同的連接埠環境執行個體。

當環境使用 Classic Load Balancer 時，Elastic Beanstalk 會預設將它設定為監聽連接埠 80 上的 HTTP 流量，轉送給同一連接埠上的執行個體。雖然您無法刪除連接埠 80 預設接聽程式，但您可以停用它，藉由封鎖流量來達到相同的功能。請注意，您可以新增或刪除其他偵聽程式。若要支援安全連線，您可以設定負載平衡器使用 443 埠上的接聽程式和 TLS 憑證。

負載平衡器使用運作狀態檢查功能，判斷執行您應用程式的 Amazon EC2 執行個體狀態是否良好。運作狀態檢查以設定的間隔發出請求至指定的 URL。如果 URL 傳回錯誤訊息，或是無法在指定的逾時期間內回傳，運作狀態檢查就會失敗。

如果您的應用程式在處理從單一伺服器上同一個用戶端傳來的多個請求時，表現較為理想，則您可以設定負載平衡器使用黏性工作階段。使用黏性工作階段時，負載平衡器會在 HTTP 回應中加入 cookie，此 cookie 可用來識別處理請求的 Amazon EC2 執行個體。從同一個用戶端收到後續的請求時，負載平衡器會使用 cookie，來將請求傳送給同一個執行個體。

若使用跨區域負載平衡，Classic Load Balancer 的每個負載平衡器節點會將請求平均分配到所有已啟用可用區域中已註冊的執行個體。若顯示跨區域負載平衡，每個負載平衡器節點會平均地將請求僅分配到其可用區域中已註冊的執行個體。

如果執行個體因為變得運作狀態不佳或環境規模縮減，而從負載平衡器移除，則在結束該執行個體與負載平衡器間的連線之前，連接耗盡功能會給執行個體時間來完成請求。您可以變更留給執行個體來傳送回應的時間長短，或完全停用連接耗盡功能。

> **Note**
>
> 當您使用 Elastic Beanstalk 主控台或 EB CLI 建立環境時，連接耗盡功能會預設為啟用。如果使用其他用戶端，您可以透過組態選項來啟用此功能。

您可以使用進階的負載平衡器設定，在任意通訊埠上設定接聽程式、修改其他的黏性工作階段設定，和設定負載平衡器安全地連線到 EC2 執行個體。透過組態選項即可使用這些設定，這些選項可以在原始程式碼中使用組態檔案來設定，或是使用 Elastic Beanstalk API 直接在環境上設定。這些設定大多也能在 Elastic Beanstalk 主控台使用。此外，您可以設定負載平衡器將存取日誌上傳到 Amazon S3。

使用 Elastic Beanstalk 主控台設定 Classic Load Balancer

在環境建立期間或之後在環境執行時，您可以使用 Elastic Beanstalk 主控台設定 Classic Load Balancer 的連接埠、HTTPS 憑證和其他設定。

> **Note**
>
> 建立環境主控台精靈上的 Classic Load Balancer (CLB) 選項已停用。如果您已使用 Classic Load Balancer 設定現有環境，則可以使用 Elastic Beanstalk 主控台或 EB CLI 來 複製現有環境以建立新的環境。您也可以選擇使用 EB CLI 或 AWS CLI 來建立使用 Classic Load

Balancer 設定的新環境。這些命令列工具會建立具有 CLB 的新環境，即使您的帳戶中沒有環境存在亦然。

若要在 Elastic Beanstalk 主控台設定執行中環境的 Classic Load Balancer

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇組態。
4. 在 Load balancer (負載平衡器) 組態類別中，選擇 Edit (編輯)。

> **Note**
> 如果 Load balancer (負載平衡器) 組態類別沒有 Edit (編輯) 按鈕，您的環境便沒有負載平衡器。若要了解如何設定一個負載平衡器，請參閱 變更環境類型。

5. 根據環境所需，變更 Classic Load Balancer 組態。
6. 若要儲存變更，請選擇頁面底部的儲存變更。

Classic Load Balancer 設定

- 接聽程式
- 工作階段
- 跨區域負載平衡
- 連接耗盡
- 運作狀態檢查

接聽程式

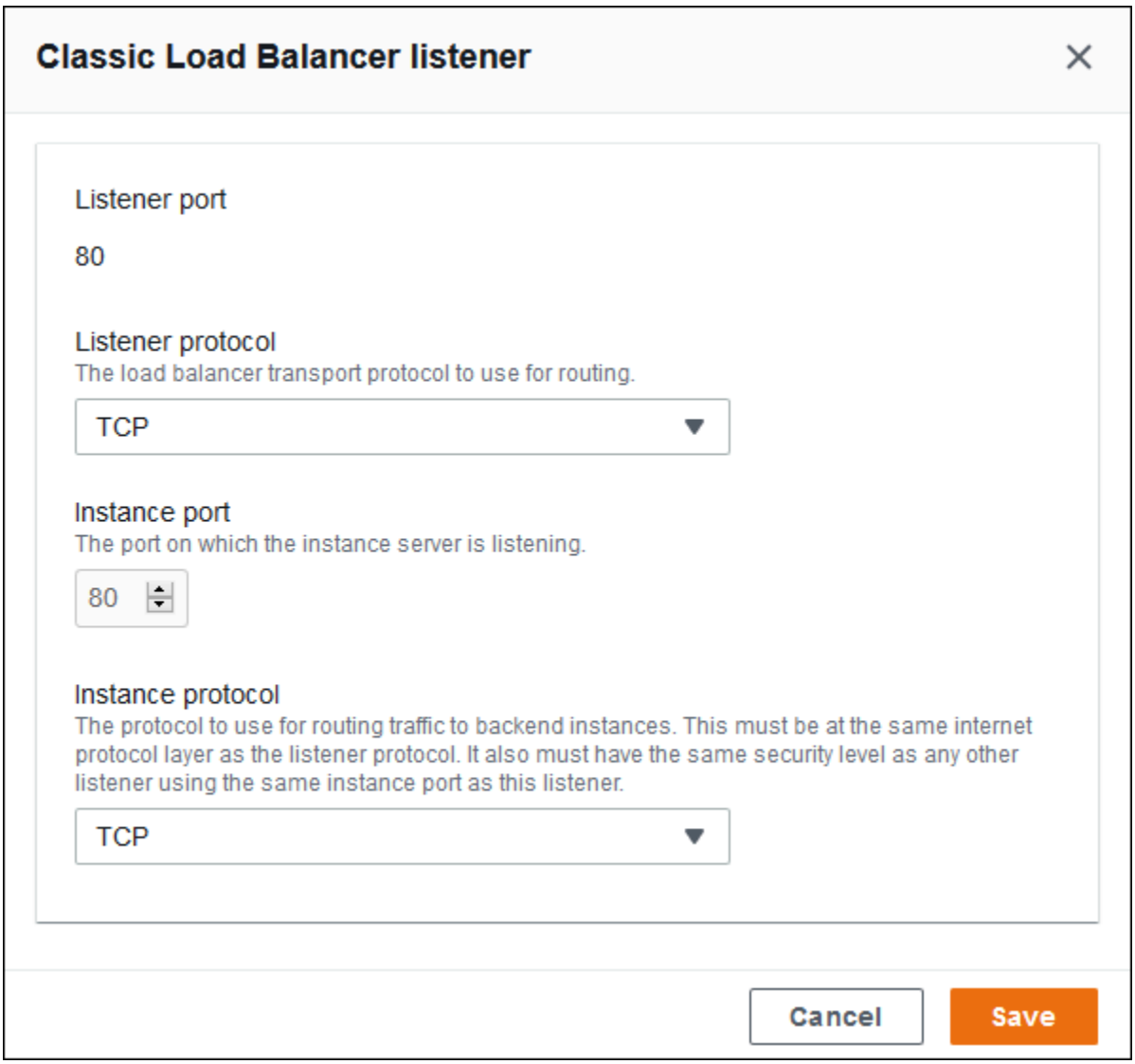
使用此名單為您的負載平衡器指定接聽程式。每個接聽程式會使用指定的通訊協定，在指定的連接埠將傳入用戶端流量路由傳送到您的執行個體。最初清單會顯示預設接聽程式，其會將連接埠 80 的 HTTP 傳入流量轉送到您接聽到連接埠 80 上 HTTP 流量的環境執行個體伺服器。

> **Note**
>
> 雖然您無法刪除連接埠 80 預設接聽程式，但您可以停用它，藉由封鎖流量來達到相同的功能。

要設定現有的接聽程式

1. 選取表格項目旁的核取方塊，選擇 Actions (動作)，然後選擇您要的動作。
2. 如果您選擇 Edit (編輯)，使用 Classic Load Balancer listener (Classic Load Balancer 接聽程式) 對話方塊編輯設定，然後選擇 Save (儲存)。

例如，如果您希望負載平衡器依原狀轉傳請求，您可以編輯預設的接聽程式，並將 Protocol (通訊協定) 從 HTTP 變更為 TCP。這可防止負載平衡器重新撰寫標頭 (包括 `X-Forwarded-For`)。此技術不適用黏性工作階段。



加入接聽程式

1. 選擇 Add listener (新增接聽程式)。
2. 在 Classic Load Balancer listener (Classic Load Balancer 接聽程式) 對話方塊中，設定所需的設定，然後選擇 Add (新增)。

新增安全的接聽程式是常用案例。下圖中的範例在連接埠 443 新增 HTTPS 流量的接聽程式。此接聽程式將傳入流量路由傳送到環境執行個體伺服器，其接聽連接埠 443 上的 HTTPS 流量。

在設定接聽程式之前，請確定您擁有有效的 SSL 憑證。執行以下任意一項：

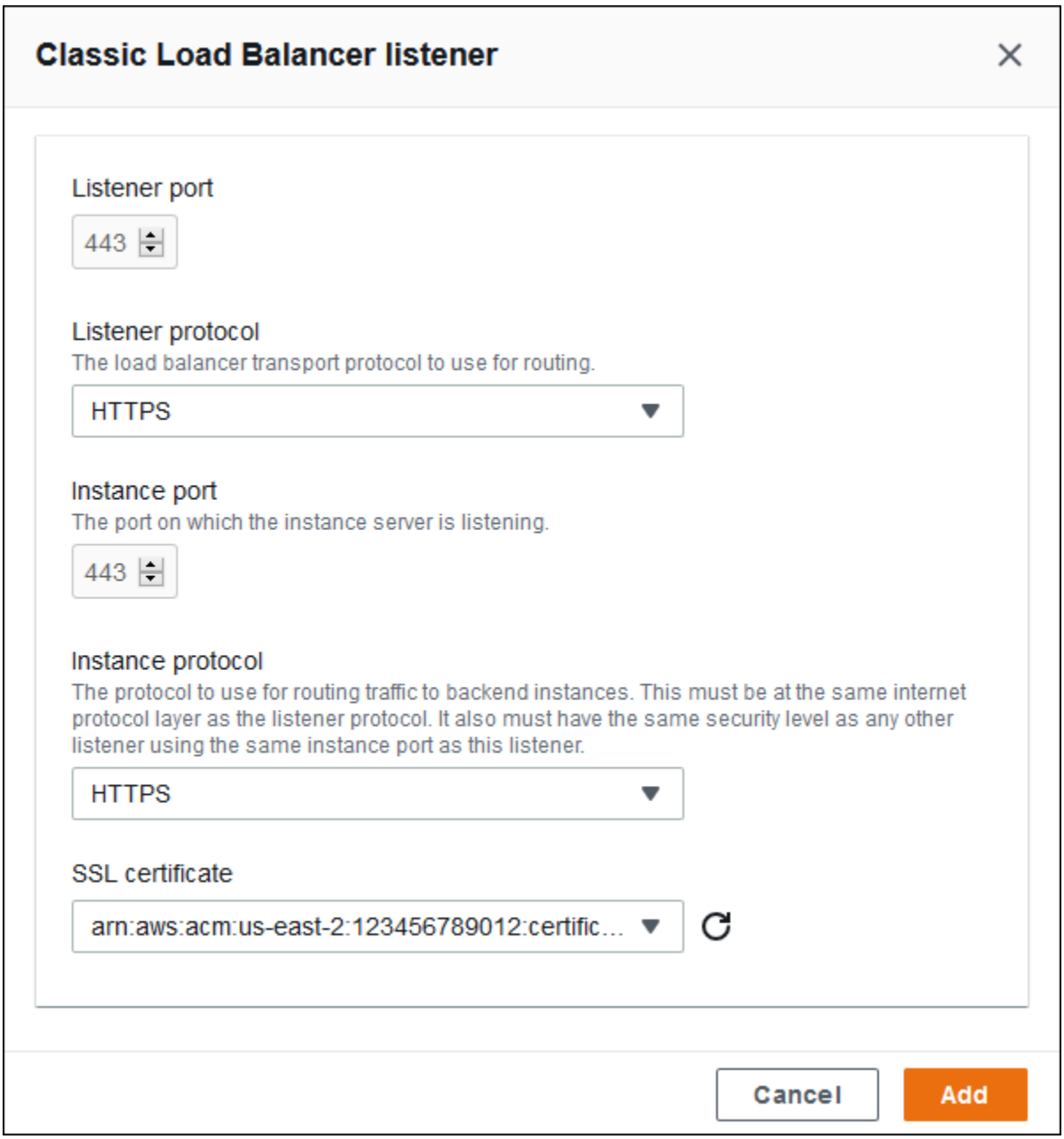
- 如果 AWS Certificate Manager (ACM) 可在您的 AWS 區域中使用，請使用 ACM 建立或匯入憑證。如需請求 ACM 憑證的詳細資訊，請參閱《AWS Certificate Manager 使用者指南》中的請求憑證。

如需有關將第三方憑證匯入 ACM 的詳細資訊，請參閱《AWS Certificate Manager 使用者指南》中的匯入憑證。

- 如果 ACM 無法在您的 AWS 區域中使用，請上傳您現有的憑證和金鑰至 IAM。如需建立和上傳憑證至 IAM 的詳細資訊，請參閱《IAM 使用者指南》中的使用伺服器憑證。

如需設定 HTTPS 和在 Elastic Beanstalk 中使用憑證的詳細資訊，請參閱為您的 Elastic Beanstalk 環境設定 HTTPS。

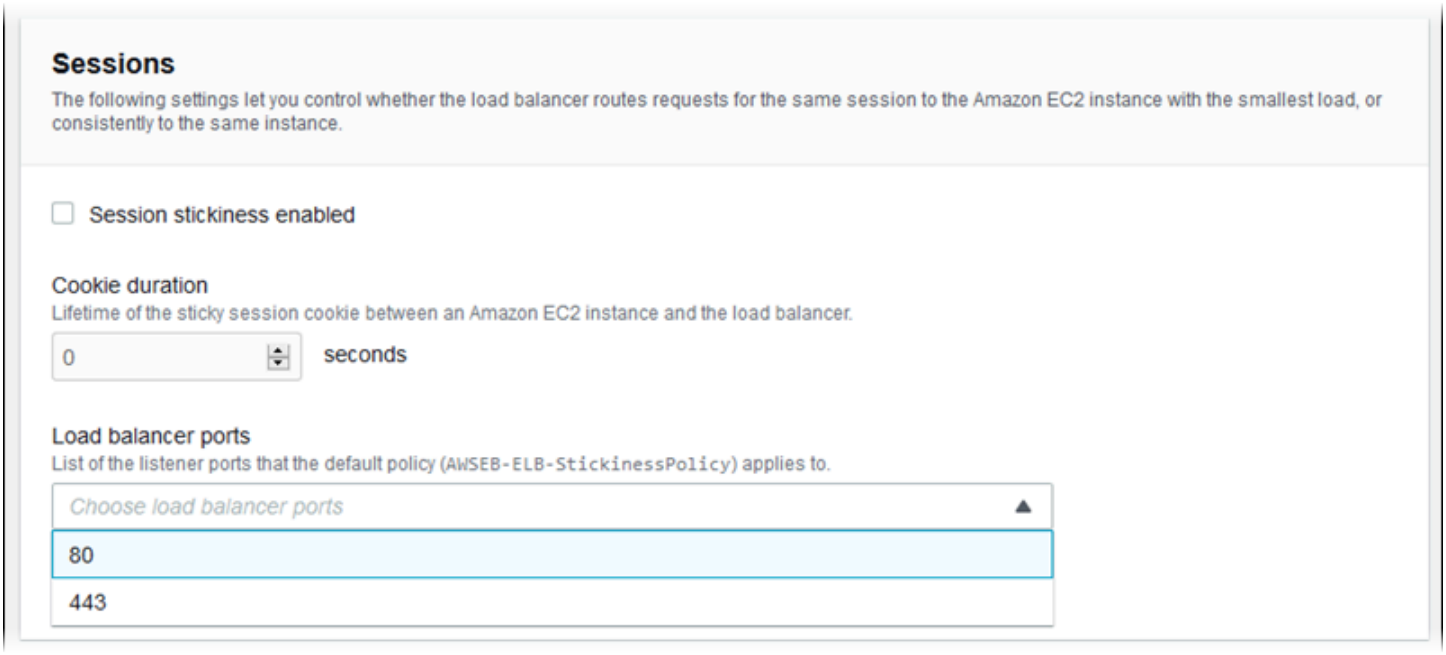
針對 SSL certificate (SSL 憑證)，選擇您 SSL 憑證的 ARN。例如 `arn:aws:iam::123456789012:server-certificate/abc/certs/build` 或 `arn:aws:acm:us-east-2:123456789012:certificate/12345678-12ab-34cd-56ef-12345678`。



如需關於設定 HTTPS 和在 Elastic Beanstalk 中使用憑證的詳細資訊，請參閱[為您的 Elastic Beanstalk 環境設定 HTTPS]。

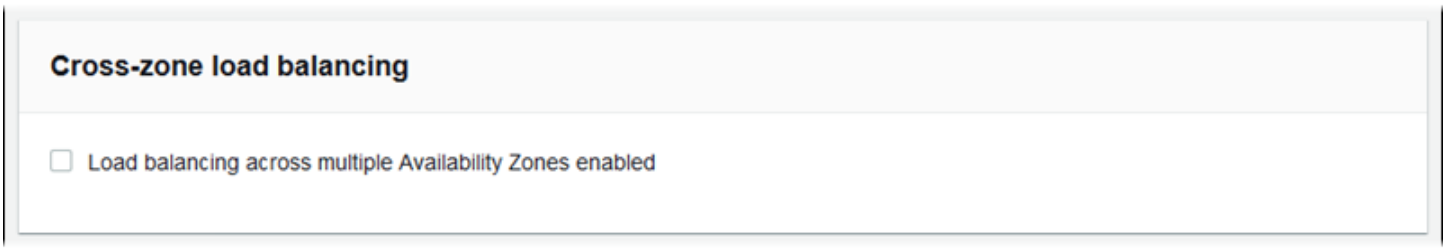
工作階段

選取或清除 Session stickiness enabled (工作階段黏著已啟用) 方塊，以啟用或停用黏性工作階段。使用 Cookie duration (Cookie 持續時間) 設定黏性工作階段的持續時間，最長 **1000000** 秒。在 Load balancer ports (負載平衡器連接埠) 清單上，選取預設政策 (`AWSEB-ELB-StickinessPolicy`) 適用的接聽程式連接埠。



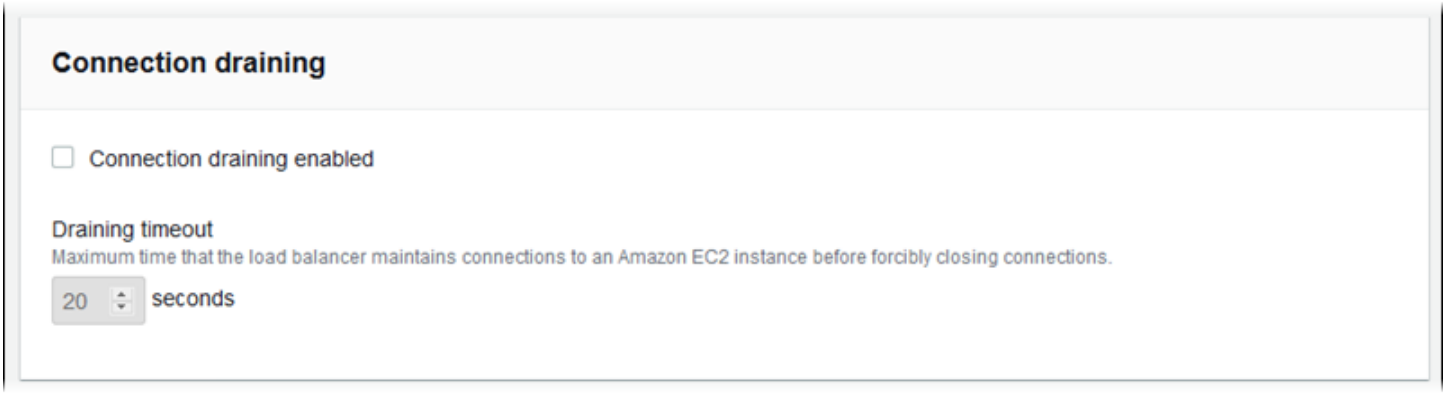
跨區域負載平衡

選取或清除 Load balancing across multiple Availability Zones enabled (橫跨多個可用區域啟用負載平衡) 方塊，以啟用或停用跨區域負載平衡。



連接耗盡

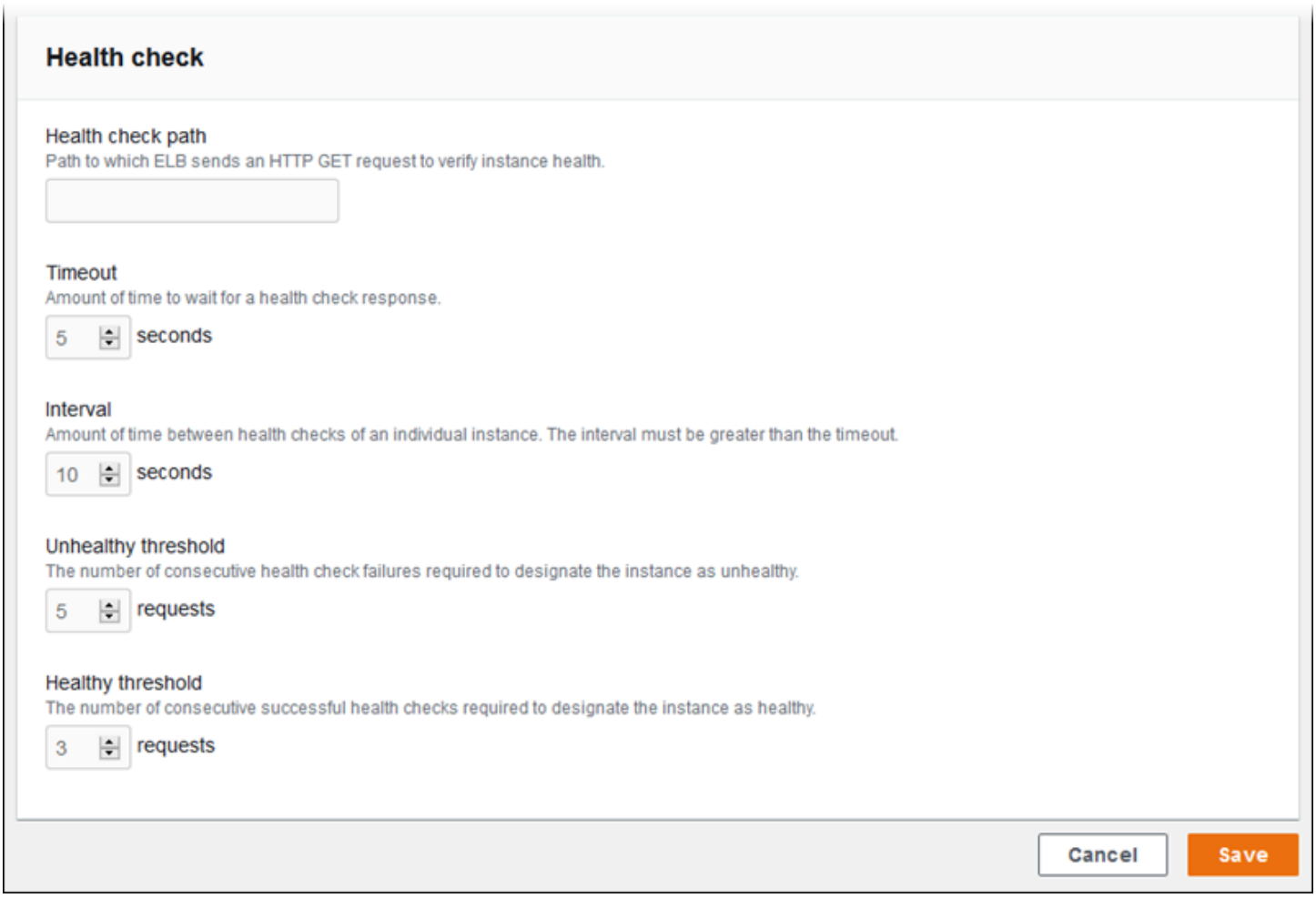
選取或清除 Connection draining enabled (啟用連接耗盡) 方塊，以啟用或停用連接耗盡。設定 Draining timeout (耗盡逾時)，最長 **3600** 秒。



運作狀態檢查

使用以下設定負載平衡器運作狀態檢查：

- Health check path (運作狀態檢查路徑) - 負載平衡器傳送運作狀態檢查的路徑。如果未設定路徑，負載平衡器就會嘗試在連接埠 80 上進行 TCP 連線，以驗證執行狀況。
- Timeout (逾時) - 等待運作狀態檢查回應的時間，以秒為單位。
- Interval (間隔) - 個別執行個體的每個運作狀態檢查之間的時間，以秒為單位。間隔必須大於逾時期間。
- Unhealthy threshold (狀態不良閾值)、Healthy threshold (運作良好閾值) - 在 Elastic Load Balancing 變更執行個體狀態之前，分別必須失敗或通過的運作狀態檢查數。



> **Note**
>
> Elastic Load Balancing 運作狀態檢查不會影響環境 Auto Scaling 群組的運作狀態檢查行為。Amazon EC2 Auto Scaling 不會自動取代未通過 Elastic Load Balancing 運作狀態檢查的執行個體，除非您手動設定 Amazon EC2 Auto Scaling 執行此項動作。如需詳細資訊，請參閱 Auto Scaling 運作狀態檢查設定。

如需關於運作狀態檢查，以及這些檢查如何影響您環境整體健全狀態的詳細資訊，請參閱基礎型運作狀態報告。

使用 EB CLI 設定 Classic Load Balancer

在您執行 eb create 時，EB CLI 會提示您選擇負載平衡器類型。

```
$ eb create
Enter Environment Name
(default is my-app): test-env
Enter DNS CNAME prefix
(default is my-app): test-env-DLW24ED23SF

Select a load balancer type
1) classic
2) application
3) network
(default is 1):
```

按 Enter 選擇 `classic`。

您亦可透過 `--elb-type` 選項，指定負載平衡器類型。

```
$ eb create test-env --elb-type classic
```

Classic Load Balancer 組態命名空間

您可以在下列命名空間中找到與 Classic Load Balancer 相關的設定：

- `aws:elb:healthcheck` - 設定負載平衡器運作狀態檢查的閾值、檢查間隔和逾時時間。
- `aws:elasticbeanstalk:application` - 設定運作狀態檢查 URL。

- aws:elb:loadbalancer - 啟用跨區域負載平衡。指派安全群組給負載平衡器，並覆寫掉 Elastic Beanstalk 所建立的預設安全群組。此命名空間也包含已作廢的選項，這些選項是用來設定標準和安全接聽程式，已經被 aws:elb:listener 命名空間中的選項所取代。
- aws:elb:listener - 設定連接埠 80 上的預設接聽程式、連接埠 443 上的安全接聽程式，或是任意埠上針對任何通訊協定的其他接聽程式。如果您指定 aws:elb:listener 做為命名空間，則設定會套用到 80 埠上的預設接聽程式。如果您指定了通訊埠 (例如，aws:elb:listener:443)，也會設定該通訊埠上的接聽程式。
- aws:elb:policies - 針對您的負載平衡器進行額外的設定。使用此命名空間中的選項，在任意連接埠上設定接聽程式、修改其他黏性工作階段設定，和設定負載平衡器以安全地連線到 Amazon EC2 執行個體。

EB CLI 和 Elastic Beanstalk 主控台會為前述選項套用建議的數值。若您想要使用組態檔進行相同的設定，您必須移除這些設定。如需詳細資訊，請參閱「建議值」。

Example .ebextensions/loadbalancer-terminatehttps.config

下列的範例組態檔案在 443 埠上建立了 HTTPS 接聽程式、指派負載平衡器用來終止安全連線的憑證，並停用 80 埠上的預設接聽程式。負載平衡器會將解密的請求，轉傳到您環境中位於 HTTP:80 的 EC2 執行個體。

```
option_settings:
  aws:elb:listener:443:
    ListenerProtocol: HTTPS
    SSLCertificateId: arn:aws:acm:us-
east-2:123456789012:certificate/12345678-12ab-34cd-56ef-12345678
    InstancePort: 80
    InstanceProtocol: HTTP
  aws:elb:listener:
    ListenerEnabled: false
```

設定 Application Load Balancer

當您啟用負載平衡時，您的 AWS Elastic Beanstalk 環境會配備 Elastic Load Balancing 器，以便在環境中的執行個體之間分配流量。Elastic Load Balancing 支援多種類型的負載平衡器。若要了解這些資訊，請參閱 Elastic Load Balancing 使用者指南。Elastic Beanstalk 可以為您建立負載平衡器，或讓您指定已建立的共享負載平衡器。

本主題說明 Elastic Beanstalk 建立並專用於您環境的 Application Load Balancer 的組態。另請參閱the section called “共享 Application Load Balancer”。如需設定 Elastic Beanstalk 支援之所有負載平衡器類型的詳細資訊，請參閱the section called “負載平衡器”。

> **Note**
>
> 您可以選擇您的環境只會在環境建立時使用的負載平衡器類型。您可以變更設定，以管理執行之環境的負載平衡器行為。您也無法從專用的負載平衡器切換到共享負載平衡器，反之亦然。

簡介

Application Load Balancer 會檢查應用程式網路通訊協定層的流量來識別請求的路徑，以便將送往不同路徑的請求導向不同目的地。

您的環境若使用 Application Load Balancer，Elastic Beanstalk 會預設將其設定為執行與 Classic Load Balancer 相同的功能。預設的接聽程式會接受 80 埠上的 HTTP 請求，並將這些請求分送到您環境中的執行個體。您可以在連接埠 443 上新增安全接聽程式 (此程式具有解密 HTTPS 傳輸資料的憑證)、設定運作狀態檢查的動作，以及將負載平衡器傳來的存取日誌推送到 Amazon Simple Storage Service (Amazon S3) 儲存貯體。

> **Note**
>
> 有別於 Classic Load Balancer 或 Network Load Balancer，Application Load Balancer 無法擁有傳輸層 (第 4 層) TCP 或 SSL/TLS 接聽程式。它僅支援 HTTP 和 HTTPS 接聽程式。此外，它無法使用後端身分驗證以驗證於負載平衡器之間與後端執行個體之間的 HTTPS 連線。

在 Elastic Beanstalk 環境中，您可以使用 Application Load Balancer，將送往某些路徑的流量導向 Web 伺服器執行個體上的不同程序。使用 Classic Load Balancer 時，傳送到接聽程式的所有流量，都會路由傳送至後端執行個體上的單一程序。使用 Application Load Balancer 時，您可以在接聽程式上設定多項「規則」，來將目的地為某些路徑的請求，路由傳送到不同的後端程序。您可以使用程序接聽的連接埠來設定每個程序。

例如，您可以執行和主要應用程式分開的登入程序。當於您環境執行個體上主要的應用程式接受大部分的請求和接聽連接埠 80 時，您的登入程序則會接聽連接埠 5000 並接受傳至 `/login` 路徑的請求。所有自用戶端內送的要求皆於連接埠 80 進來。使用 Application Load Balancer 時，您可以在連接埠 80 上設定用於傳入流量的單一接聽程式，透過兩個規則將流量根據請求中的路徑分別路由傳送到兩個個別

程序。您可以新增自訂規則，將流量路由傳送至 /login 之後再送到登入處理序接聽連接埠 5000。預設的規則為路由所有其他流量至在連接埠 80 上的主要應用程式接聽。

Application Load Balancer 規則會將請求對應至「目標群組」。在 Elastic Beanstalk 中，目標群組由「程序」表示。您可以使用協定、連接埠以及運作狀態檢查設定來設定程序。此程序代表您環境中的執行個體上執行的程序。預設程序是反向代理 (nginx 或 Apache) 80 埠上的接聽程式，此反向代理會在您應用程式的前景執行。

> **Note**
>
> 在 Elastic Beanstalk 外，目標群組對應至一組執行個體。接聽程式可使用規則和目標群組，根據路徑將流量路由到不同的執行個體。在 Elastic Beanstalk 內，您環境中的所有執行個體都是相同的，因此差別在於監聽不同連接埠的程序。

Classic Load Balancer 針對整個環境使用單一運作狀態檢查路徑。使用 Application Load Balancer 時，每個程序都有個別的運作狀態檢查路徑，該路徑由負載平衡器和 Elastic Beanstalk 增強型運作狀態監控進行監控。

若要使用 Application Load Balancer，您的環境必須在預設或自訂 VPC 中，而且必須擁有具備一組標準許可的服務角色。如果您有較舊的服務角色，可能會需要更新其權限，來納入 `elasticloadbalancing:DescribeTargetHealth` 和 `elasticloadbalancing:DescribeLoadBalancers`。如需 Application Load Balancer 的詳細資訊，請參閱什麼是 Application Load Balancer？。

> **Note**
>
> Application Load Balancer 運作狀態檢查不使用 Elastic Beanstalk 運作狀態檢查路徑。反之，它為每個程序各別使用特定的路徑設定。

使用 Elastic Beanstalk 主控台設定 Application Load Balancer

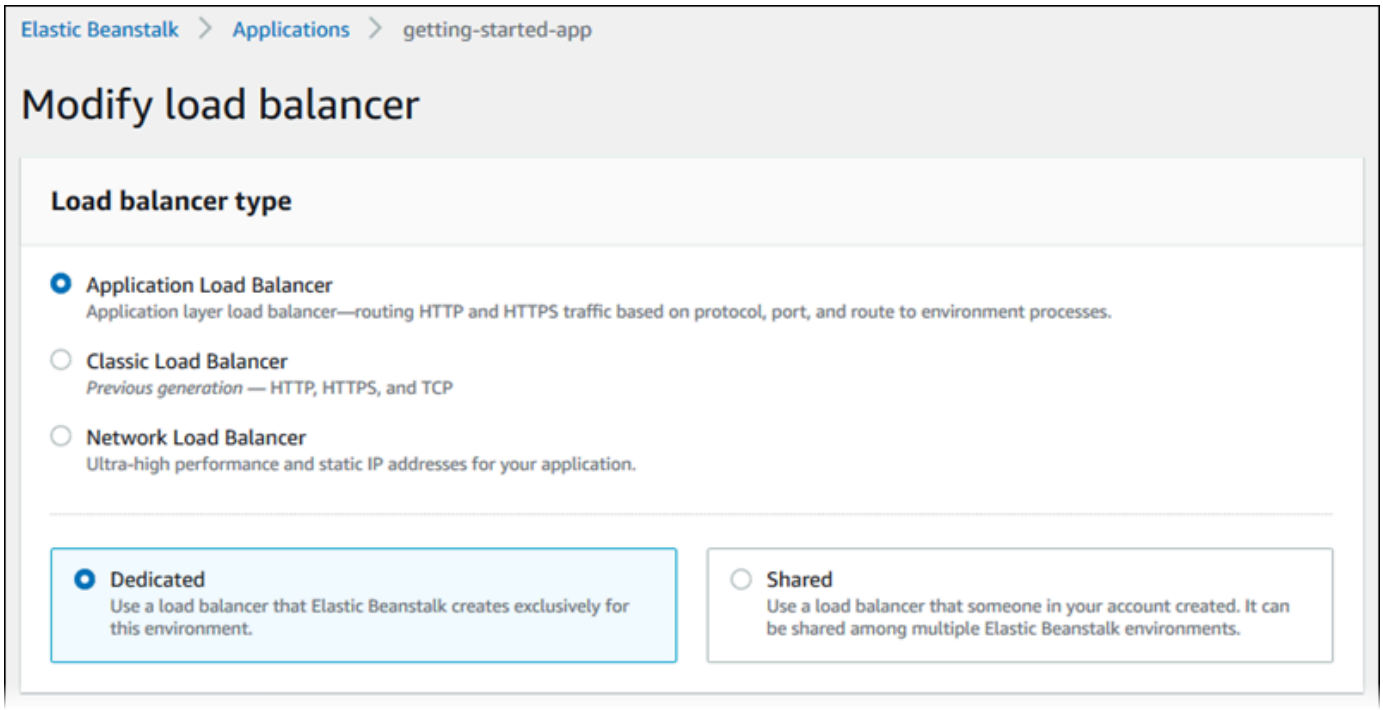
在環境建立期間或之後在環境執行時，您可以使用 Elastic Beanstalk 主控台設定 Application Load Balancer 的接聽程式、程序和規則。

若要在環境建立期間於 Elastic Beanstalk 主控台設定 Application Load Balancer

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域

2. 在導覽窗格中，選擇 Environments (環境)。
3. 選擇 Create a new environment (建立新環境) 以開始建立您的環境。
4. 在精靈的主頁上，在選擇 Create environment (建立環境) 之前，選擇 Configure more options (設定更多選項)。
5. 選擇 High availability (高可用性) 組態預設。

   或者，在 Capacity (容量) 組態類別中，設定Load balanced (負載平衡器) 環境類型。如需詳細資訊，請參閱 容量。
6. 在 Load balancer (負載平衡器) 組態類別中，選擇 Edit (編輯)。
7. 選取 Application Load Balancer 和 Dedicated (專用) 選項 (如果尚未選取)。



8. 根據環境所需，變更任何 Application Load Balancer 組態。
9. 選擇 Save (儲存)，然後針對您環境所需對其他組態做變更。
10. 選擇 Create environment (建立環境)。

若要在 Elastic Beanstalk 主控台設定執行中環境的 Application Load Balancer

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇組態。
4. 在 Load balancer (負載平衡器) 組態類別中，選擇 Edit (編輯)。

> **Note**
>
> 如果 Load balancer (負載平衡器) 組態類別沒有 Edit (編輯) 按鈕，您的環境便沒有負載平衡器。若要了解如何設定一個負載平衡器，請參閱 變更環境類型。

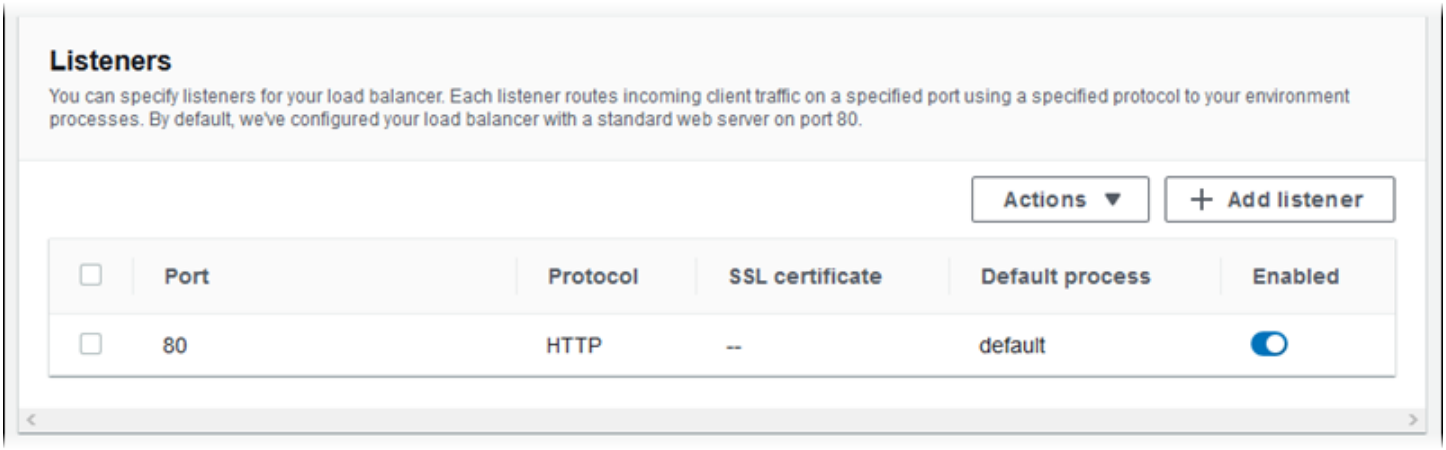
5. 根據環境所需，變更 Application Load Balancer 組態。
6. 若要儲存變更，請選擇頁面底部的儲存變更。

Application Load Balancer 設定

- 接聽程式
- Processes
- 規則
- 存取日誌擷取

接聽程式

使用此名單為您的負載平衡器指定接聽程式。每個接聽程式會藉由特定的通訊協定至您執行個體上的一個或多個程序，將傳入用戶端流量路由至指定的連接埠上。一開始，清單會顯示將連接埠 80 上傳入的 HTTP 流量路由傳送至名為預設的接聽程式。



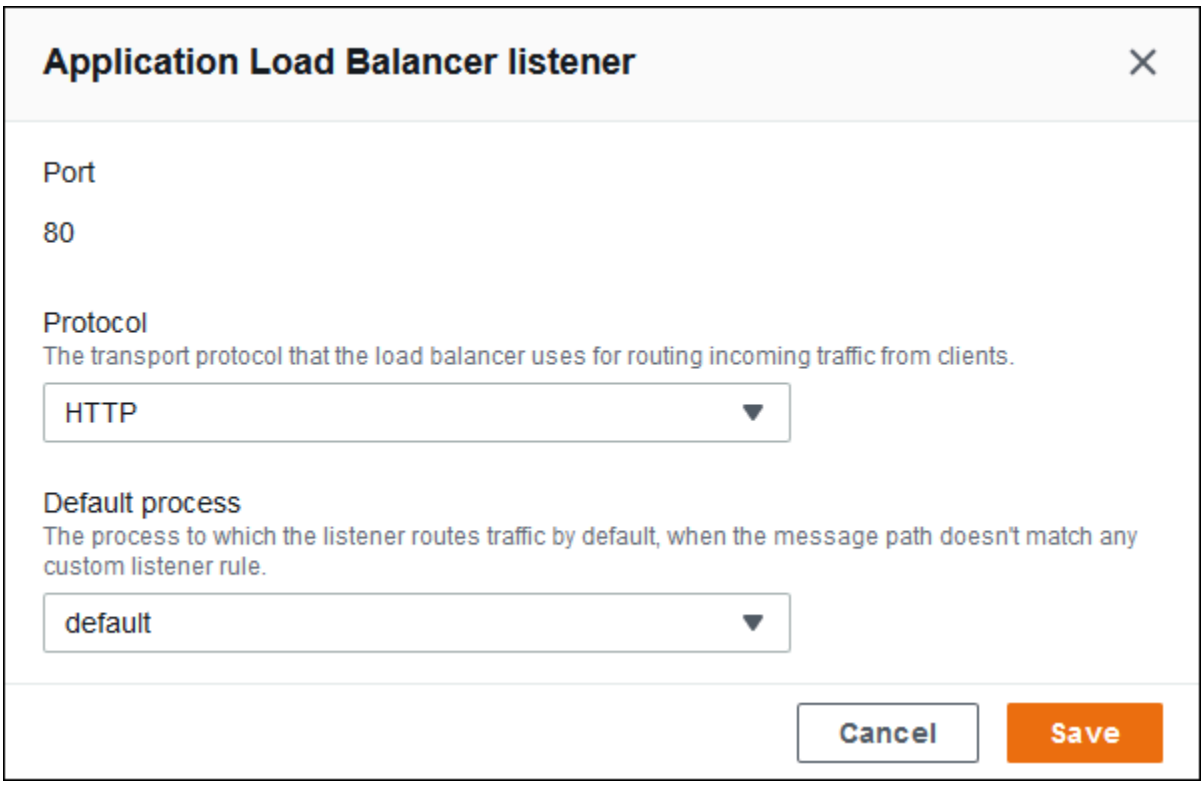
要設定現有的接聽程式

1. 選取其表項目旁邊的核取方塊，然後選擇 Actions (動作)，Edit (編輯)。
2. 使用 Application Load Balancer listener (應用程式負載平衡器接聽程式) 對話方塊編輯設定，然後選擇 Save (儲存)。

加入接聽程式

1. 選擇 Add listener (新增接聽程式)。
2. 在 Application Load Balancer 接聽程式對話方塊中，設定所需的設定，然後選擇新增。

使用 Application Load Balancer 接聽程式對話方塊設定來選擇接聽程式接聽流量的連接埠和協定。如果您選擇 HTTPS 通訊協定，請設定 SSL 設定。



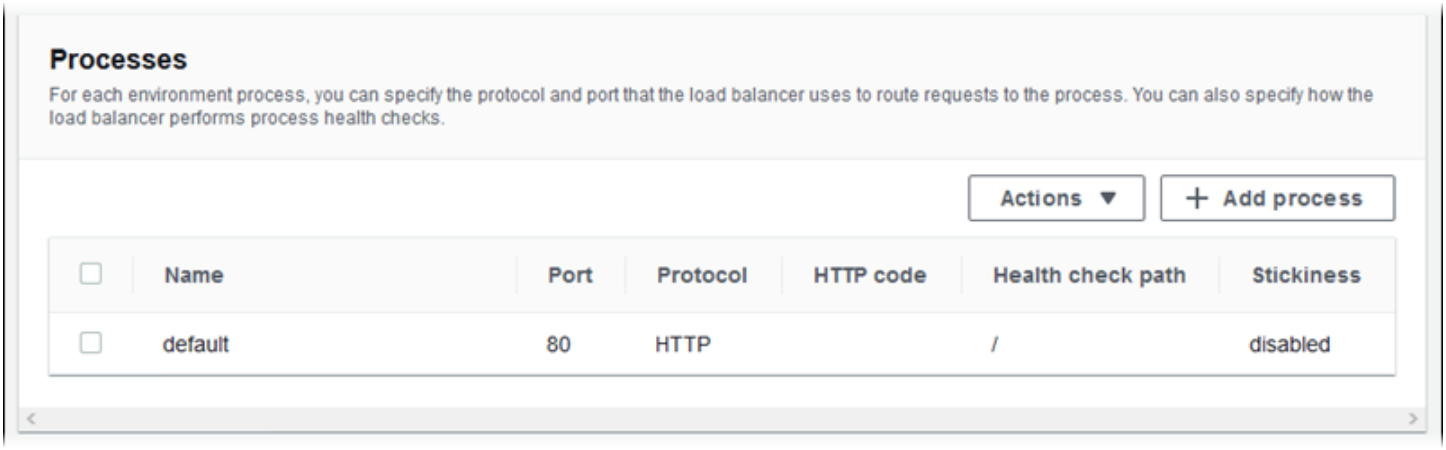
在設定接聽程式之前，請確定您擁有有效的 SSL 憑證。執行以下任意一項：

- 如果您的 AWS 區域中有 AWS Certificate Manager (ACM) 可用，請使用 ACM 建立或匯入憑證。如需請求 ACM 憑證的詳細資訊，請參閱《AWS Certificate Manager 使用者指南》中的請求憑證。如需有關將第三方憑證匯入 ACM 的詳細資訊，請參閱《AWS Certificate Manager 使用者指南》中的匯入憑證。
- 如果您的 AWS 區域無法使用 ACM，請將現有憑證和金鑰上傳至 IAM。如需建立和上傳憑證至 IAM 的詳細資訊，請參閱《IAM 使用者指南》中的使用伺服器憑證。

如需設定 HTTPS 和在 Elastic Beanstalk 中使用憑證的詳細資訊，請參閱為您的 Elastic Beanstalk 環境設定 HTTPS。

Processes

使用此名單為您的負載平衡器指定程序。程序為接聽程式路由流量的目標。每個接聽程式會藉由特定的通訊協定至您執行個體上的一個或多個程序，將傳入用戶端流量路由至指定的連接埠上。一開始，清單顯示接聽於連接埠 80 上傳入的 HTTP 流量的預設程序。



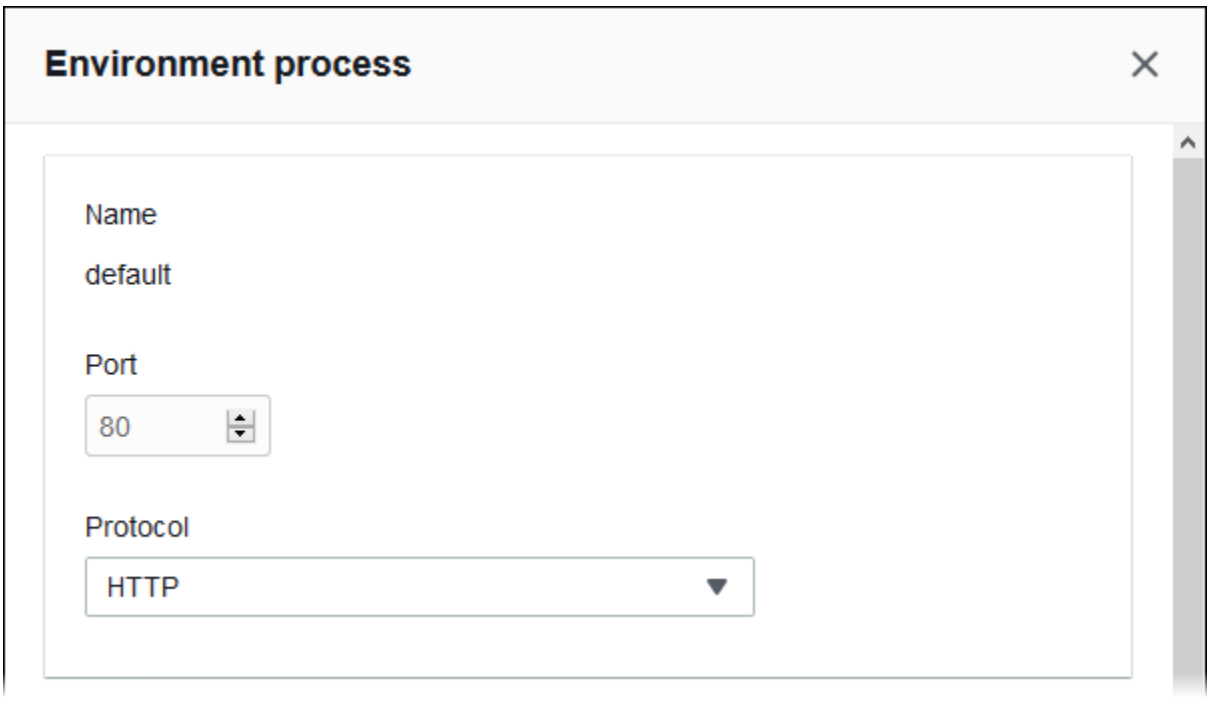
您可以編輯現有程序的設定，或增加新的程序。若要開始編輯清單中的一個程序或新增一個程序，使用列於接聽程式清單的相同步驟。Environment process (環境程序) 對話方塊開啟。

Application Load Balancer 的環境程序對話方塊設定

- 定義
- 運作狀態檢查
- 工作階段

定義

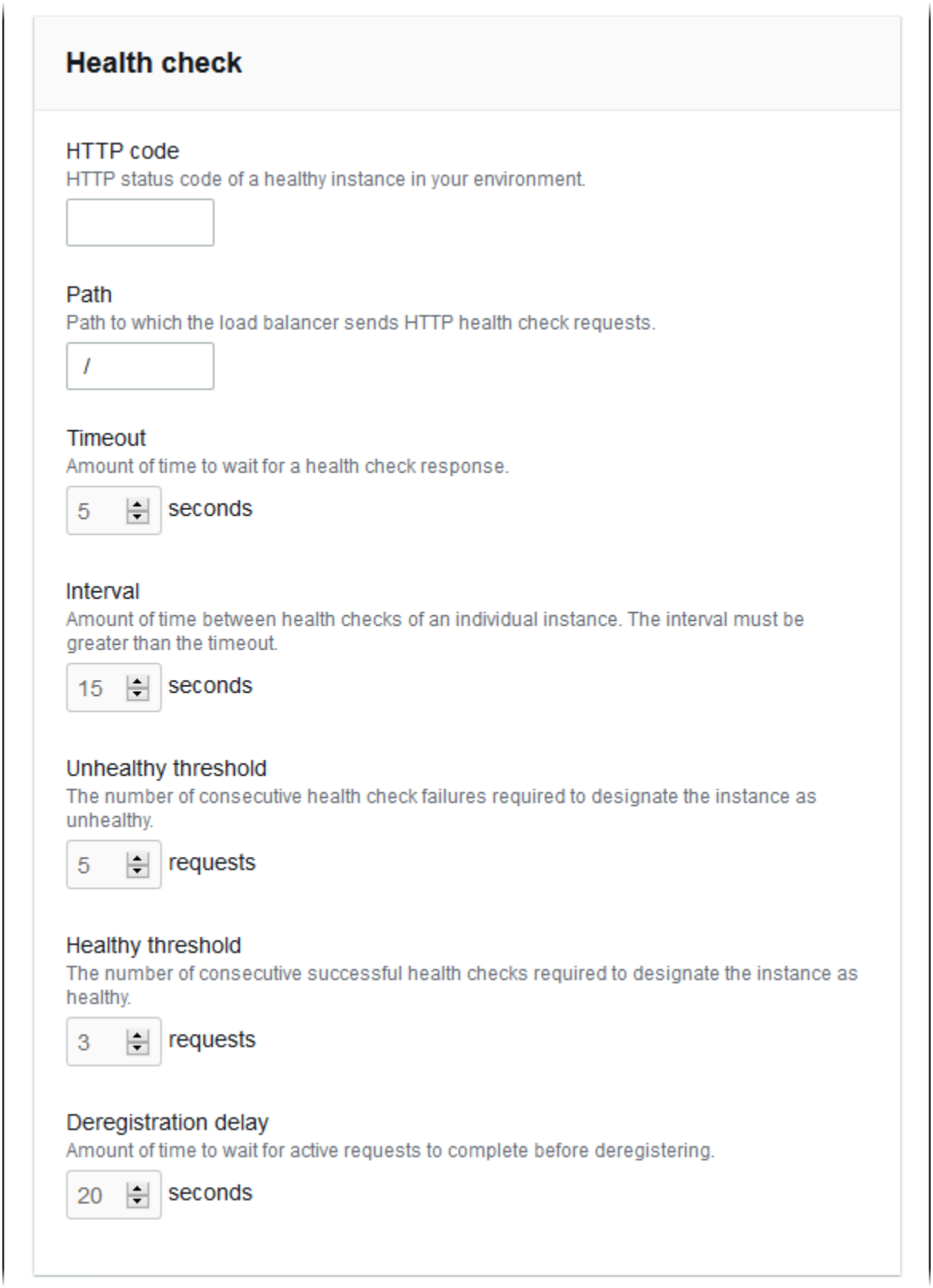
使用這些設定來定義程序：它的 Name (名稱) 和其接聽請求的 Port (連接埠) 和 Protocol (協定)。



運作狀態檢查

使用以下設定以設定程序運作狀態檢查：

- HTTP code (HTTP 代碼) - 指定運作狀態良好程序的 HTTP 狀態碼。
- Path (路徑) - 程序的運作狀態檢查請求路徑。
- Timeout (逾時) - 等待運作狀態檢查回應的時間，以秒為單位。
- Interval (間隔) - 個別執行個體的每個運作狀態檢查之間的時間，以秒為單位。間隔必須大於逾時期間。
- Unhealthy threshold (狀態不良閾值)、Healthy threshold (運作良好閾值) - 在 Elastic Load Balancing 變更執行個體狀態之前，分別必須失敗或通過的運作狀態檢查數。
- Deregistration delay (取消註冊延遲) - 在取消註冊執行個體前，等待作用中請求完成的時間，以秒為單位。

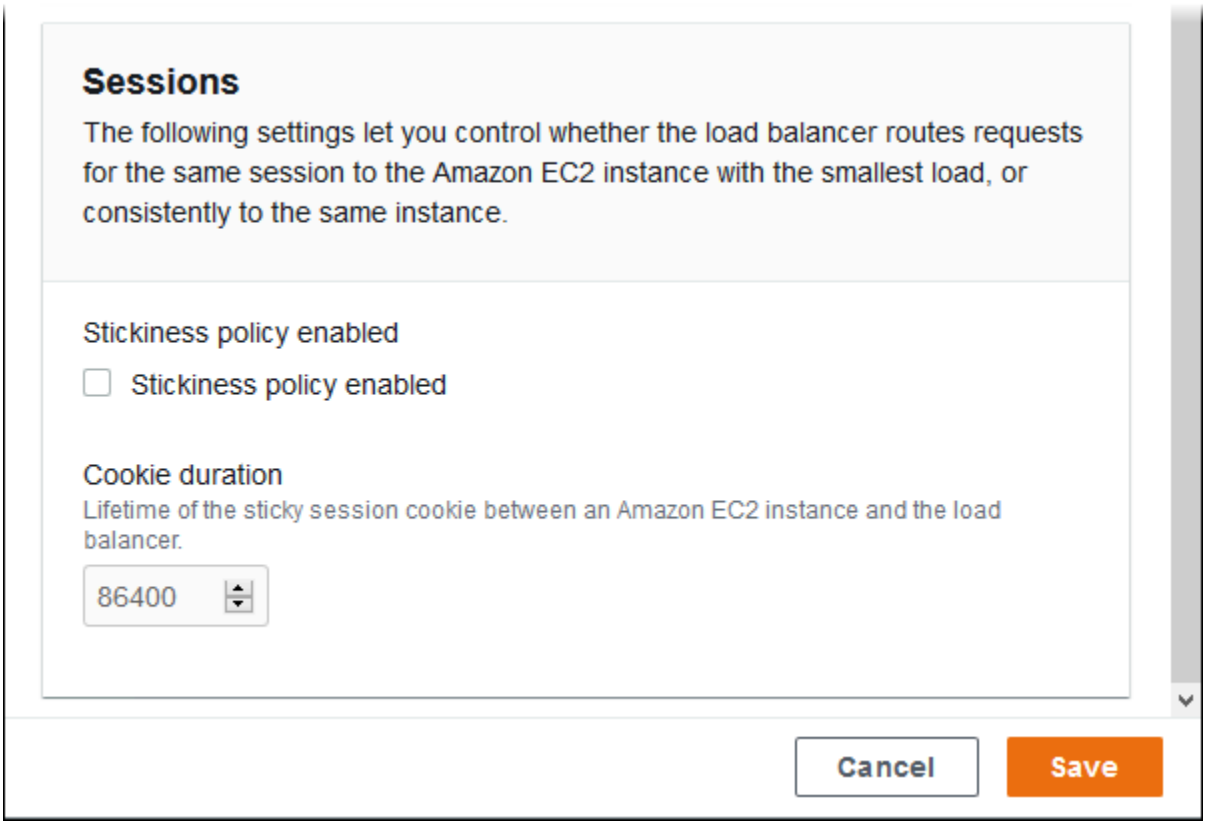

Health check
HTTP code
HTTP status code of a healthy instance in your environment.
Path
Path to which the load balancer sends HTTP health check requests.
/
Timeout
Amount of time to wait for a health check response.
5 seconds
Interval
Amount of time between health checks of an individual instance. The interval must be greater than the timeout.
15 seconds
Unhealthy threshold
The number of consecutive health check failures required to designate the instance as unhealthy.
5 requests
Healthy threshold
The number of consecutive successful health checks required to designate the instance as healthy.
3 requests
Deregistration delay
Amount of time to wait for active requests to complete before deregistering.
20 seconds

> **Note**
>
> Elastic Load Balancing 運作狀態檢查不會影響環境 Auto Scaling 群組的運作狀態檢查行為。Amazon EC2 Auto Scaling 不會自動取代未通過 Elastic Load Balancing 運作狀態檢查的執行個體，除非您手動設定 Amazon EC2 Auto Scaling 執行此項動作。如需詳細資訊，請參閱 Auto Scaling 運作狀態檢查設定。

如需關於運作狀態檢查，以及這些檢查如何影響您環境整體健全狀態的詳細資訊，請參閱基礎型運作狀態報告。

工作階段

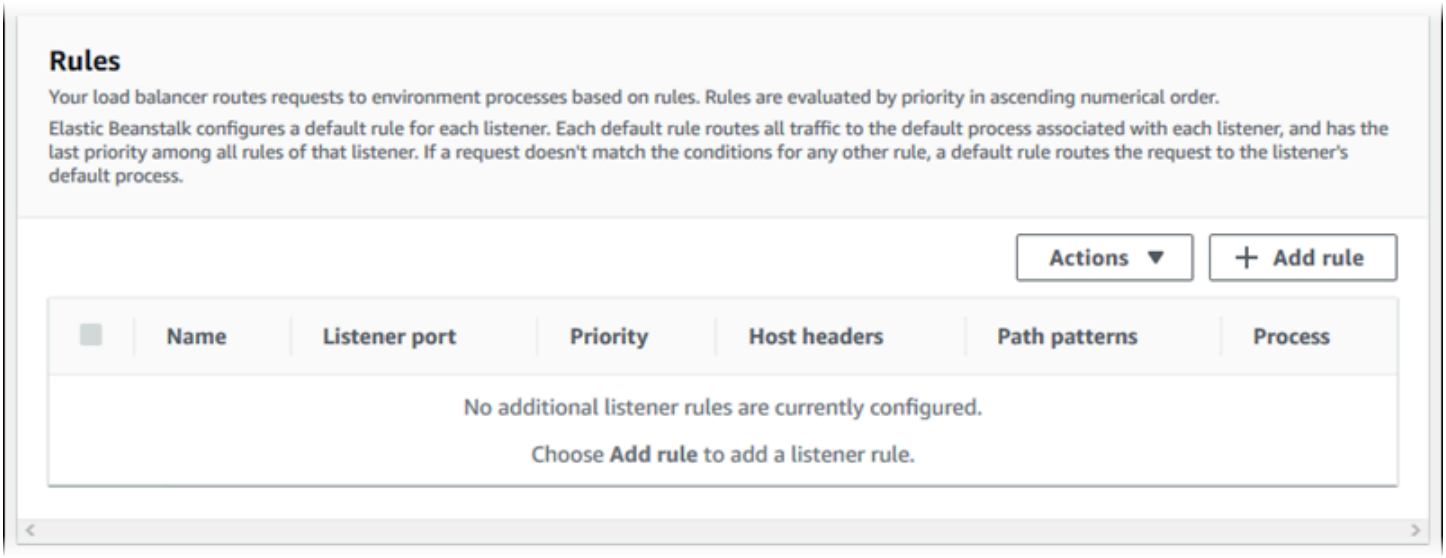
選取或清除 Stickiness policy enabled (啟用黏性政策) 方塊，以啟用或停用黏性工作階段。使用 Cookie duration (Cookie 持續時間) 設定黏性工作階段的持續時間，最長 **604800** 秒。



規則

使用此名單為您的負載平衡器指定自訂接聽程式規則。規則映射在特定路徑模式接聽程式接收到的要求於目標程序中。每個接聽程式可以有多個規則，在您的執行個體上於不同的路徑路由請求到不同的程序。

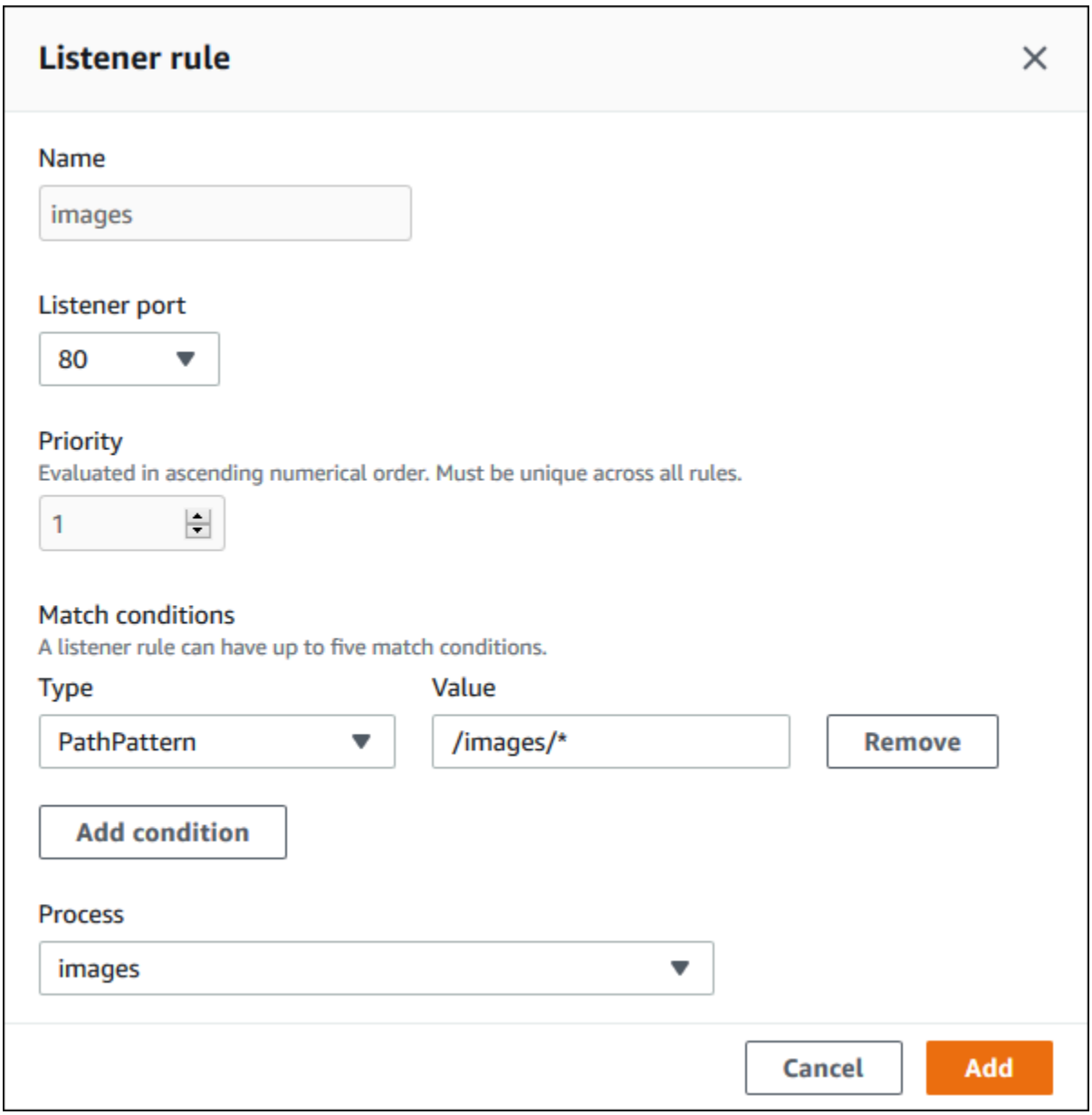
規則有數字優先順序，判斷套用到傳入要求的優先順序。對於您新增的每個新接聽程式，Elastic Beanstalk 會新增一個將所有接聽程式的流量都路由傳送到預設程序的預設規則。預設規則的優先順序是最低的，如果相同的接聽程式沒有其它規則符合傳入的請求，則套用。一開始，如果您尚未新增自訂規則，清單會是空的。不會顯示所有接聽程式的預設規則。



您可以編輯現有規則的設定，或增加新的規則。若要開始編輯清單中的一個規則或新增一個規則，使用列於接聽程式清單的相同步驟。Listener rule (接聽程式規則) 對話方塊開啟，依以下設定：

- Name (名稱) - 規則的名稱。
- Listener port (接聽程式連接埠) - 套用規則的接聽程式連接埠。
- Priority (優先順序) - 規則的優先順序。較低的號碼有較高的優先順序。接聽程式規則的優先順序必須是唯一的。
- Match conditions (比對條件) - 要套用規則的請求 URL 條件清單。條件有兩種類型：HostHeader(URL 的網域部分) 和 PathPattern(URL 的路徑部分)。您最多可以新增五個條件。每個條件值最多可包含 128 個字元，且可包含萬用字元。
- Process (程序) - 負載平衡器路由傳送符合規則之請求的目標程序。

當編輯任何現有的規則，您無法更改其 Name (名稱) 和 Listener port (接聽程式連接埠)。

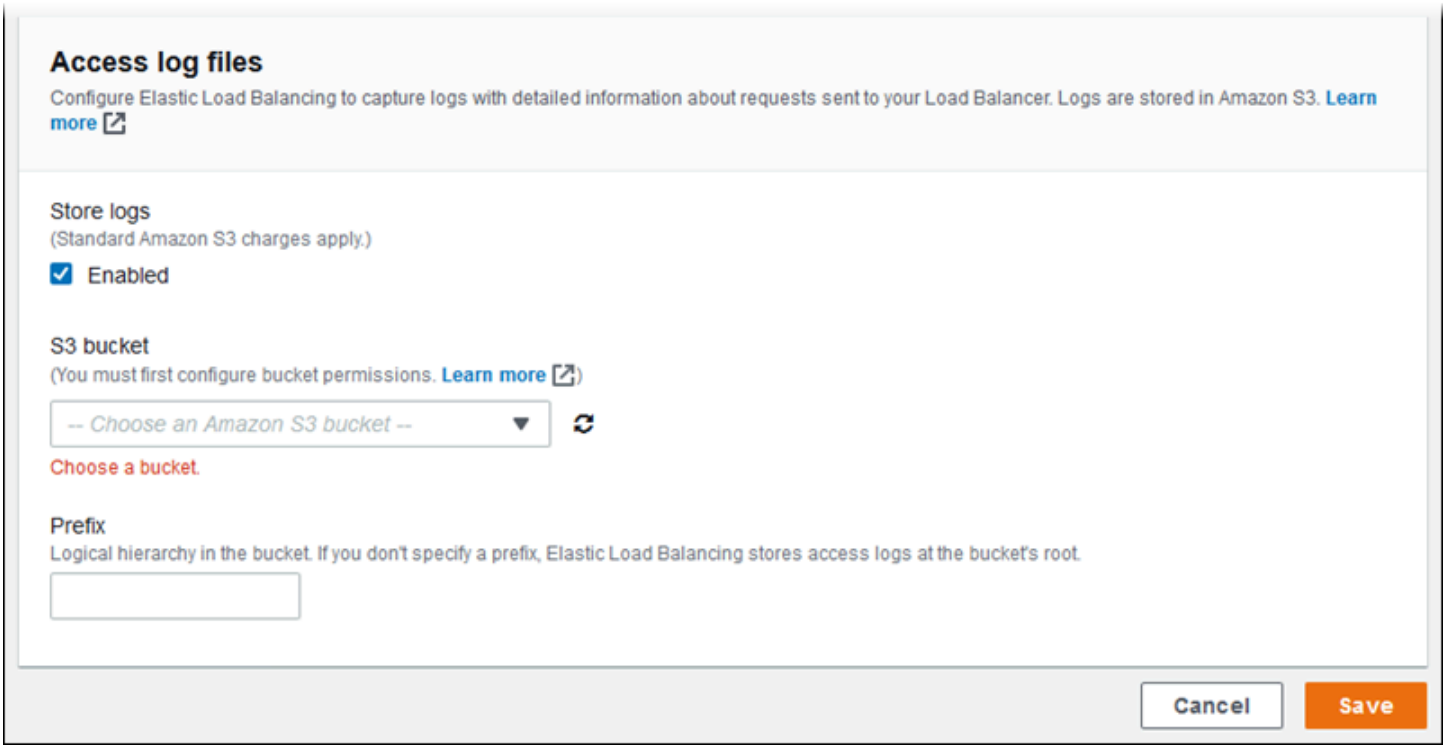


存取日誌擷取

使用這些設定可設定 Elastic Load Balancing 以擷取日誌，其中包含傳送至 Application Load Balancer 之請求的詳細資訊。存取日誌擷取預設為停用。啟用 Store logs (存放日誌) 時，Elastic Load Balancing 會將日誌存放在您設定的 S3 bucket (S3 儲存貯體)。Prefix (字首) 設定會指定儲存貯體中存放日誌的最上層資料夾。Elastic Load Balancing 會將日誌放在此字首之下名為 AWSLogs 的資料夾。如果不指定字首，Elastic Load Balancing 會將該資料夾放在儲存貯體的根層級。

> **Note**
>
> 如果您為存取日誌擷取設定的 Amazon S3 儲存貯體不是 Elastic Beanstalk 為您的帳戶建立的儲存貯體，請務必將具有適當許可的使用者政策新增至您的 AWS Identity and Access Management (IAM) 使用者。Elastic Beanstalk 僅提供 Elastic Beanstalk 受管資源的許可的受管使用者政策。

如需有關存取日誌的詳細資訊，包括許可和其他要求，請參閱 Application Load Balancer 的存取日誌。



範例：具有安全接聽程式和兩個程序的 Application Load Balancer

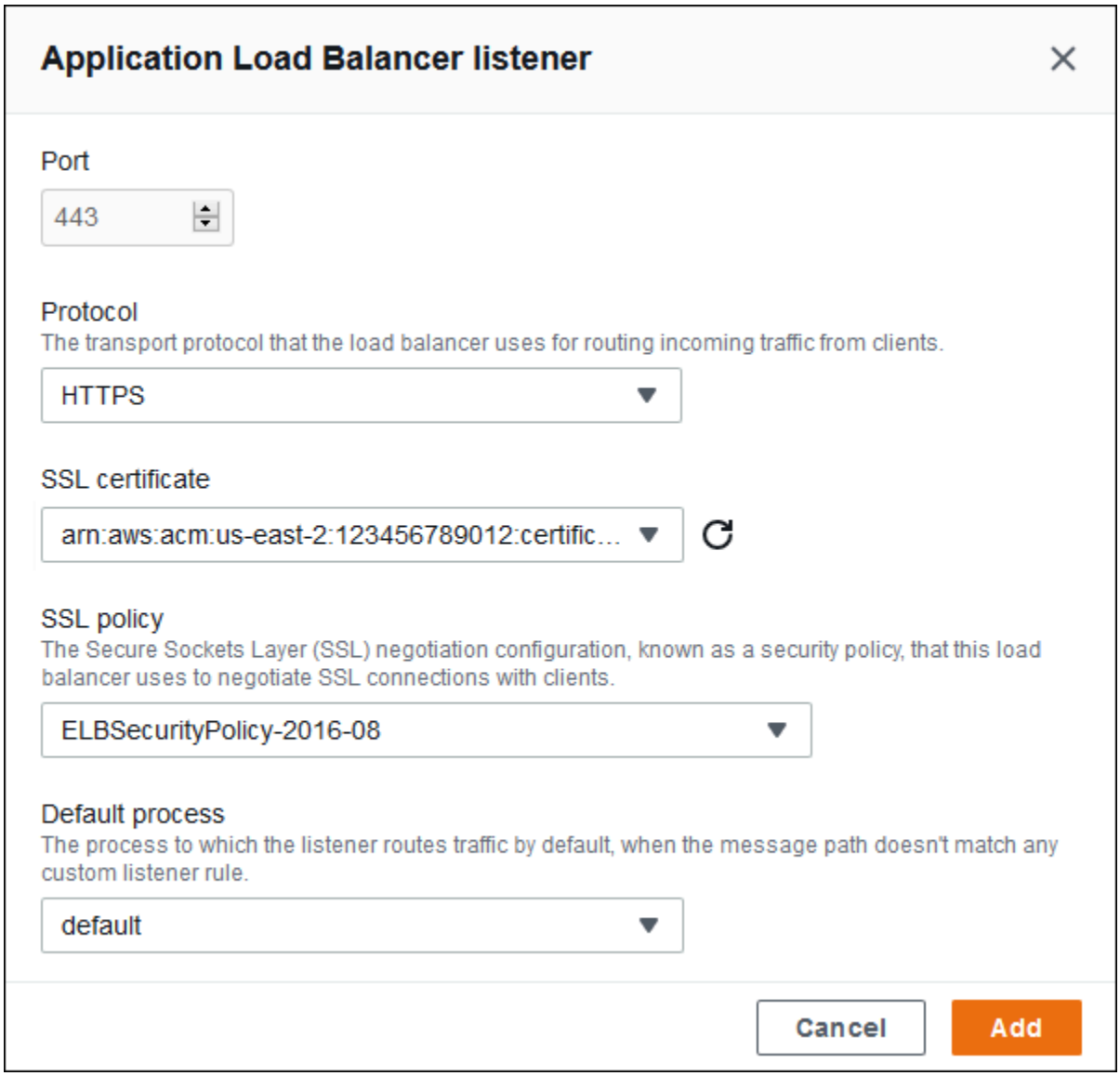
在此範例中，您的應用程式需要 end-to-end 流量加密和單獨的處理程序來處理系統管理要求。

若要設定您環境的 Application Load Balancer 以滿足這些請求，請移除預設接聽程式，新增 HTTPS 接聽程式，指出預設程序會監聽 HTTPS 上的連接埠 443，然後新增程序和接聽程式規則以用於不同路徑的管理流量。

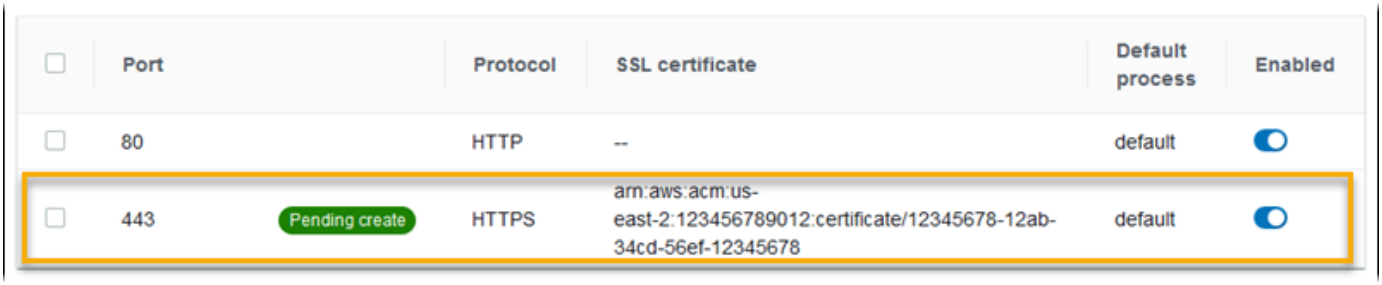
要設定負載平衡器在這個範例

1. 新增一個安全接聽程式。對於連接埠，請輸入 **443**。對於通訊協定，請選取 **HTTPS**。對於 SSL 憑證，請選擇您 SSL 憑證的 ARN。例如 **arn:aws:iam::123456789012:server-certificate/abc/certs/build** 或 **arn:aws:acm:us-east-2:123456789012:certificate/12345678-12ab-34cd-56ef-12345678**。

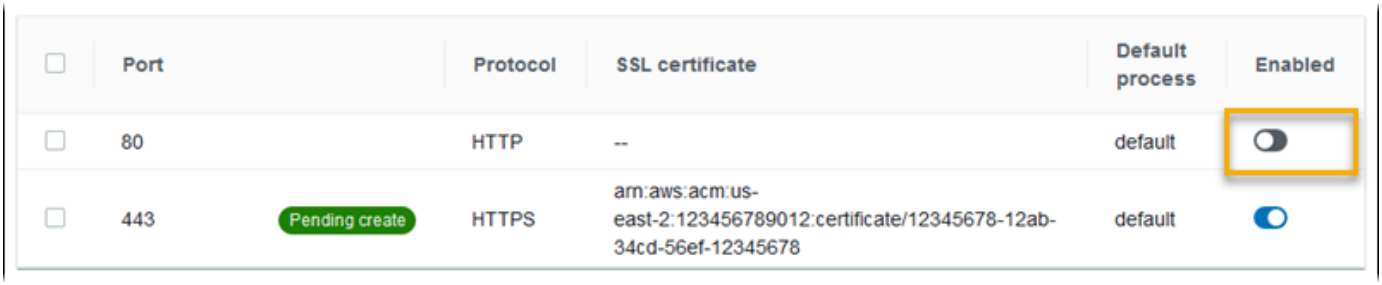
   對於預設程序，請保持選取 **default**。



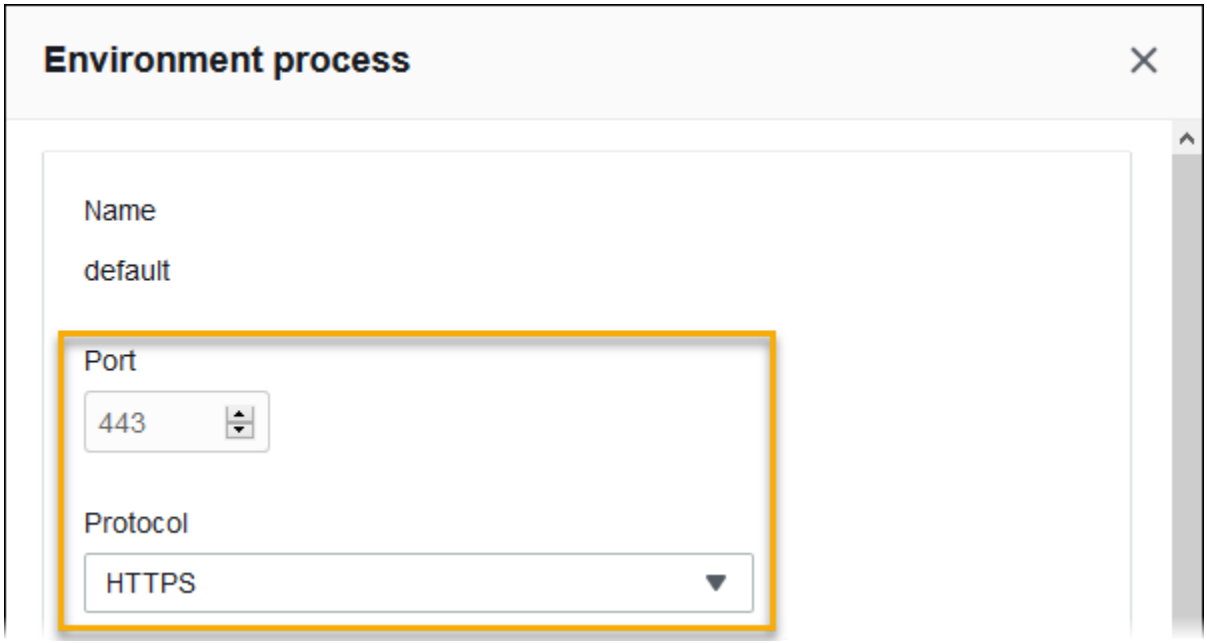
   您現在可以查看您清單上的額外接聽程式。



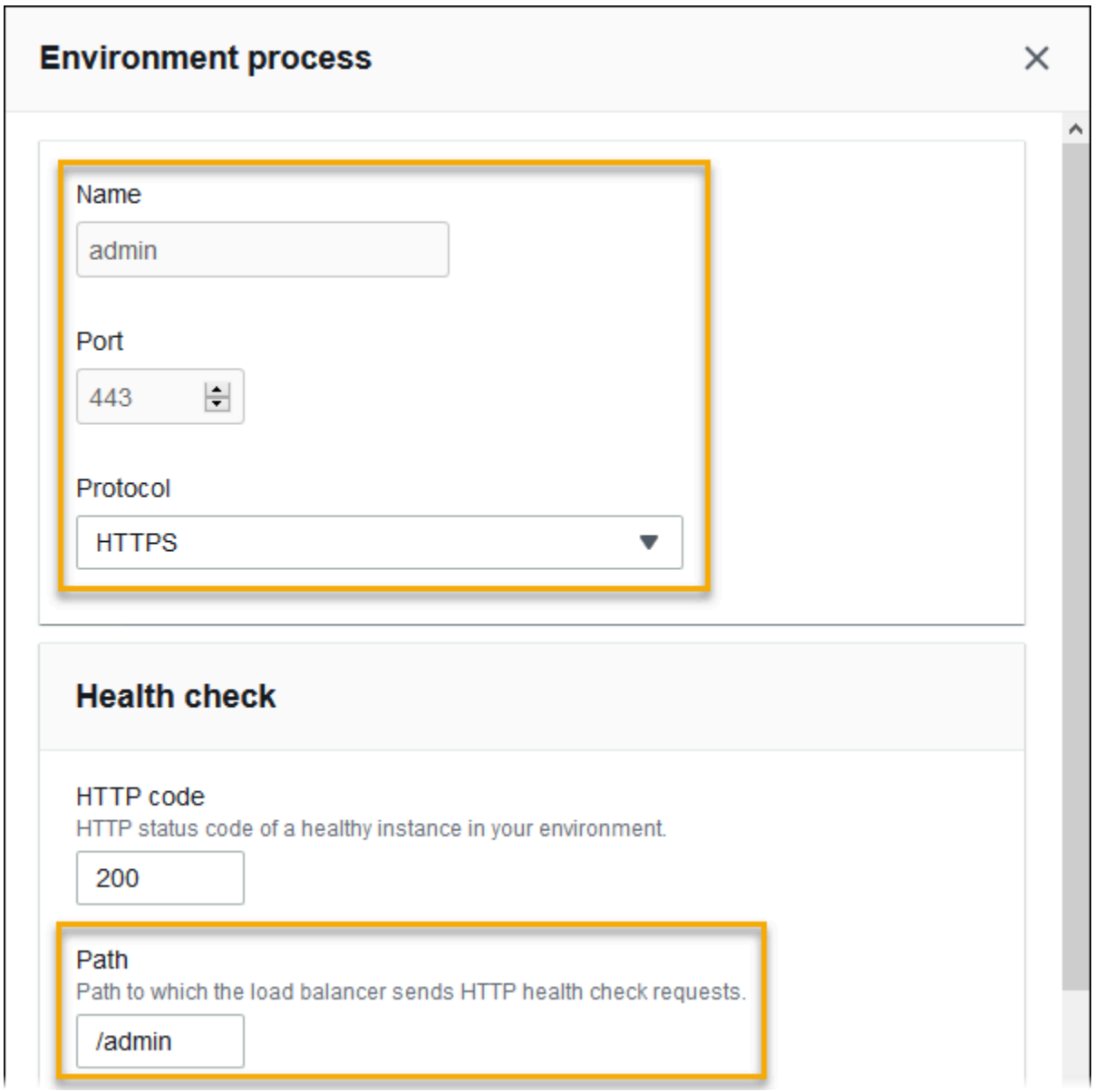
2. 停用預設的連接埠 80 HTTP 接聽程式。針對預設的接聽程式，關閉 Enabled (已啟用) 選項。



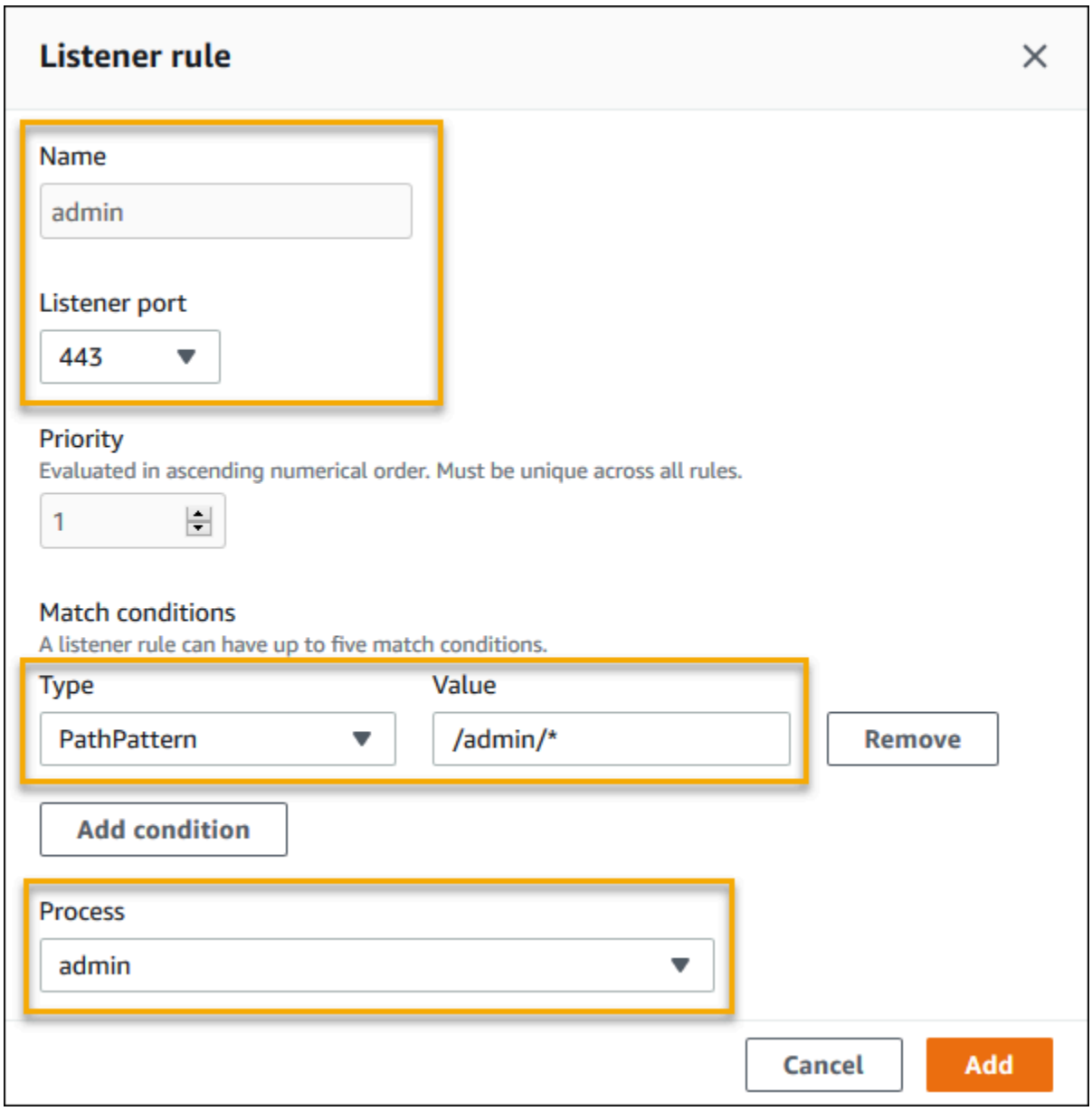
3. 設定預設程序至 HTTPS。選取預設程序，然後在 Actions (動作) 中選擇 Edit (編輯)。對於連接埠，請輸入 **443**。對於通訊協定，請選取 **HTTPS**。



4. 新增管理程序。在 Name (名稱) 輸入 **admin**。對於連接埠，請輸入 **443**。對於通訊協定，請選取 **HTTPS**。對於運作狀態檢查下方的路徑，請輸入 **/admin**。



5. 為管理流量新增一個規則。在 Name (名稱) 輸入 **admin**。對於接聽程式連接埠，請輸入 **443**。對於「符合條件」，請使PathPattern用值加入一個**/admin/***。對於程序，請選取 **admin**。



使用 EB CLI 設定 Application Load Balancer

在您執行 eb create 時，EB CLI 會提示您選擇負載平衡器類型。

```
$ eb create
Enter Environment Name
(default is my-app): test-env
Enter DNS CNAME prefix
(default is my-app): test-env-DLW24ED23SF

Select a load balancer type
```

```
1) classic
2) application
3) network
(default is 2):
```

您亦可透過 `--elb-type` 選項，指定負載平衡器類型。

```
$ eb create test-env --elb-type application
```

Application Load Balancer 命名空間

您可以在下列命名空間找到與 Application Load Balancer 相關的設定：

- `aws:elasticbeanstalk:environment` - 選擇環境的負載平衡器類型。Application Load Balancer 的值為 `application`。

  您無法在組態檔案 (.Ebextensions) 中設定此選項。
- `aws:elbv2:loadbalancer` - 將套用至 Application Load Balancer 的存取日誌和其他設定進行整體設定。
- `aws:elbv2:listener` - 設定 Application Load Balancer 上的接聽程式。這些設定對應至 Classic Load Balancer 之 `aws:elb:listener` 中的設定。
- `aws:elbv2:listenerrule` - 設定規則，根據請求的路徑，將流量路由傳送至不同程序。規則對 Application Load Balancer 是唯一的。
- `aws:elasticbeanstalk:environment:process` - 設定運作狀態檢查，並針對執行於環境執行個體的程序，指定連接埠和通訊協定。此連接埠和通訊協定設定，對應至 Classic Load Balancer 接聽程式之 `aws:elb:listener` 中的執行個體連接埠和執行個體通訊協定設定。運作狀態檢查設定則對應至 `aws:elb:healthcheck` 和 `aws:elasticbeanstalk:application` 命名空間內的設定。

Example .ebextension/. alb-access-logs 配置

下列組態檔案可讓您使用 Application Load Balancer 上傳環境的存取日誌。

```
option_settings:
  aws:elbv2:loadbalancer:
    AccessLogsS3Bucket: DOC-EXAMPLE-BUCKET
    AccessLogsS3Enabled: 'true'
    AccessLogsS3Prefix: beanstalk-alb
```

Example .ebextension/. alb-default-process 配置

下列的組態檔案會修改預設程序的運作狀態檢查與黏著度設定。

```
option_settings:
  aws:elasticbeanstalk:environment:process:default:
    DeregistrationDelay: '20'
    HealthCheckInterval: '15'
    HealthCheckPath: /
    HealthCheckTimeout: '5'
    HealthyThresholdCount: '3'
    UnhealthyThresholdCount: '5'
    Port: '80'
    Protocol: HTTP
    StickinessEnabled: 'true'
    StickinessLBCookieDuration: '43200'
```

Example .ebextension/. alb-secure-listener 配置

下列的組態檔案在連接埠 443 上新增了安全接聽程式和對應的程序。

```
option_settings:
  aws:elbv2:listener:443:
    DefaultProcess: https
    ListenerEnabled: 'true'
    Protocol: HTTPS
    SSLCertificateArns: arn:aws:acm:us-
east-2:123456789012:certificate/21324896-0fa4-412b-bf6f-f362d6eb6dd7
  aws:elasticbeanstalk:environment:process:https:
    Port: '443'
    Protocol: HTTPS
```

Example .ebextension/. alb-admin-rule 配置

下列的組態檔案新增了具備使用 /admin 請求路徑路由流量至在連接埠 4443 上接聽名為 admin 程序的規則。

```
option_settings:
  aws:elbv2:listener:443:
    DefaultProcess: https
    ListenerEnabled: 'true'
    Protocol: HTTPS
```

```
    Rules: admin
    SSLCertificateArns: arn:aws:acm:us-
east-2:123456789012:certificate/21324896-0fa4-412b-bf6f-f362d6eb6dd7
  aws:elasticbeanstalk:environment:process:https:
    Port: '443'
    Protocol: HTTPS
  aws:elasticbeanstalk:environment:process:admin:
    HealthCheckPath: /admin
    Port: '4443'
    Protocol: HTTPS
  aws:elbv2:listenerrule:admin:
    PathPatterns: /admin/*
    Priority: 1
    Process: admin
```

設定共享 Application Load Balancer

若啟用負載平衡，您的 AWS Elastic Beanstalk 環境會配備 Elastic Load Balancing 負載平衡器，將流量分配到您環境中的執行個體。Elastic Load Balancing 支援多種類型的負載平衡器。若要了解這些資訊，請參閱 Elastic Load Balancing 使用者指南。Elastic Beanstalk 可以為您建立負載平衡器，或讓您能夠指定已建立的共享負載平衡器。

本主題說明您建立並與環境產生關聯的共享 Application Load Balancer 組態。另請參閱 the section called “Application Load Balancer”。如需設定 Elastic Beanstalk 支援之所有負載平衡器類型的詳細資訊，請參閱您的 Elastic Beanstalk 環境的負載平衡器。

> **Note**
>
> 您可以選擇您的環境只會在環境建立時使用的負載平衡器類型。您可以變更設定，以管理執行之環境的負載平衡器行為。您也無法從專用的負載平衡器切換到共享負載平衡器，反之亦然。

簡介

「共享負載平衡器」是您用 Amazon Elastic Compute Cloud (Amazon EC2) 服務建立和自我管理的一種負載平衡器，之後可在多個 Elastic Beanstalk 環境中使用。

當您建立負載平衡、具擴展性的環境並選擇使用 Application Load Balancer 時，Elastic Beanstalk 預設會建立專用於您環境的負載平衡器。若要了解 Application Load Balancer 是什麼，以及它在 Elastic Beanstalk 環境中的運作方式，請參閱簡介來為 Elastic Beanstalk 設定 Application Load Balancer。

在某些情況下，您可能想要節省擁有多個專用負載平衡器的成本。當您有多個環境時，例如，如果您的應用程式是微型服務套件，而不是整合式服務，這將會十分實用。在這種情況下，您可以選擇使用共享負載平衡器。

若要使用共享負載平衡器，請先在 Amazon EC2 中建立它，然後新增一個或多個接聽程式。接著在建立 Elastic Beanstalk 環境期間，您提供負載平衡器並選擇接聽程式連接埠。Elastic Beanstalk 會將接聽程式與您環境中的預設程序產生關聯。您可以新增自訂接聽程式規則，將流量從特定主機標頭和路徑路由傳送至其他環境處理程序。

Elastic Beanstalk 會將標籤新增至共享負載平衡器。標籤名稱為 `elasticbeanstalk:shared-elb-environment-count`，其值為共享此負載平衡器的環境數目。

使用共享負載平衡器與使用專用負載平衡器的方式不同。

Regarding	專用 Application Load Balancer	共享 Application Load Balancer
管理	Elastic Beanstalk 會建立和管理負載平衡器、接聽程式、接聽程式規則和程序 (目標群組)。Elastic Beanstalk 也會在您終止環境時移除它們。Elastic Beanstalk 可以設定負載平衡器存取日誌擷取 (如果您選擇該選項)。	您可以在 Elastic Beanstalk 之外建立和管理負載平衡器和接聽程式。Elastic Beanstalk 會建立和管理預設規則和預設程序，而您可以新增規則和程序。Elastic Beanstalk 會移除環境建立期間新增的接聽程式規則和程序。
接聽程式規則	Elastic Beanstalk 為每個接聽程式建立一個預設規則，以將所有流量路由傳送到接聽程式的預設程序。	Elastic Beanstalk 只會將預設規則與連接埠 80 接聽程式 (如果存在) 產生關聯。如果您選擇不同的預設接聽程式連接埠，則必須將預設規則與其產生關聯 (Elastic Beanstalk 主控台和 EB CLI 會為您執行此操作)。 為了解決共享負載平衡器環境間的接聽程式規則條件衝突，Elastic Beanstalk 會將環境的 CNAME 做為主機標頭條件新增至接聽程式規則。 Elastic Beanstalk 將規則優先順序設定視為共享負載平衡器環境間的相對優先順序，並在建立期間將其對應至絕對優先順序。

Regarding	專用 Application Load Balancer	共享 Application Load Balancer
安全群組	Elastic Beanstalk 會建立預設安全群組，並將其連接至負載平衡器。	您可以設定一或多個安全群組以用於負載平衡器。如果不這樣做，Elastic Beanstalk 會檢查 Elastic Beanstalk 所管理的現有安全群組是否已連接至負載平衡器。如果沒有，Elastic Beanstalk 會建立安全群組，並將其連接至負載平衡器。當共享負載平衡器的最後一個環境終止時，Elastic Beanstalk 會刪除此安全群組。
更新	您可以在建立環境後更新您的 Application Load Balancer。您可以編輯接聽程式、接聽程式規則和處理程序。您可以設定負載平衡器存取日誌擷取。	您無法使用 Elastic Beanstalk 在您的 Application Load Balancer 中設定存取日誌擷取，並且在建立環境後無法更新接聽程式和接聽程式規則。您只能更新處理程序 (目標群組)。若要設定存取日誌擷取，以及更新接聽程式和接聽程式規則，請使用 Amazon EC2。

使用 Elastic Beanstalk 主控台設定共享 Application Load Balancer

在建立環境期間，您可以使用 Elastic Beanstalk 主控台設定共享 Application Load Balancer。您可以選取其中一個帳戶的可共享負載平衡器以用於環境、選取預設接聽程式連接埠，以及設定其他處理程序和接聽程式規則。

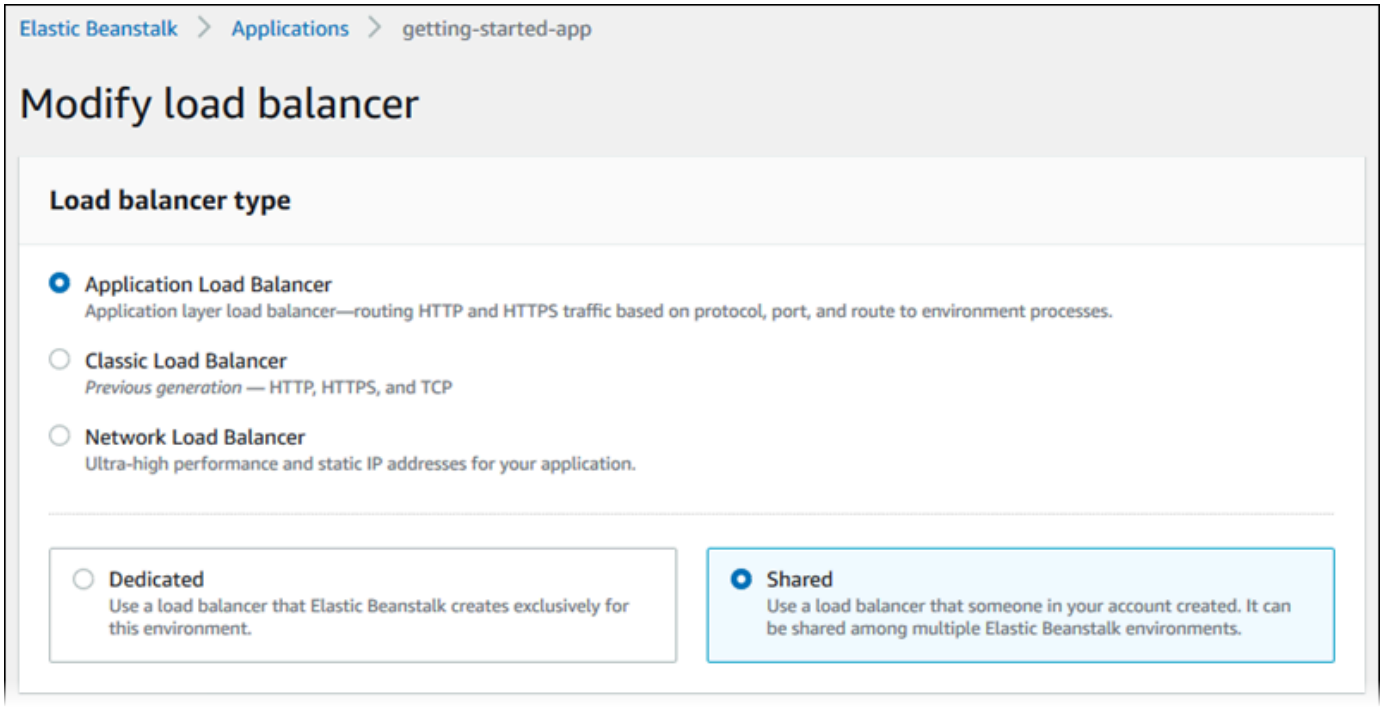
建立環境之後，您無法在 Application Load Balancer 主控台中編輯共享 Application Load Balancer 組態。若要設定接聽程式、接聽程式規則、程序 (目標群組) 和存取日誌擷取，請使用 Amazon EC2。

若要在環境建立期間於 Elastic Beanstalk 主控台設定 Application Load Balancer

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Environments (環境)。
3. 選擇 Create a new environment (建立新環境) 以開始建立您的環境。
4. 在精靈的主頁上，在選擇 Create environment (建立環境) 之前，選擇 Configure more options (設定更多選項)。
5. 選擇 High availability (高可用性) 組態預設。

   或者，在 Capacity (容量) 組態類別中，設定Load balanced (負載平衡器) 環境類型。如需詳細資訊，請參閱容量。

6. 在 Load balancer (負載平衡器) 組態類別中，選擇 Edit (編輯)。
7. 選取 Application Load Balancer 選項 (如果尚未選取)，然後選取 Shared (共享) 選項。



8. 根據環境所需，變更任何共享 Application Load Balancer 組態。
9. 選擇 Save (儲存)，然後針對您環境所需對其他組態做變更。
10. 選擇 Create environment (建立環境)。

共享 Application Load Balancer 設定

- 共享 Application Load Balancer
- Processes
- 規則

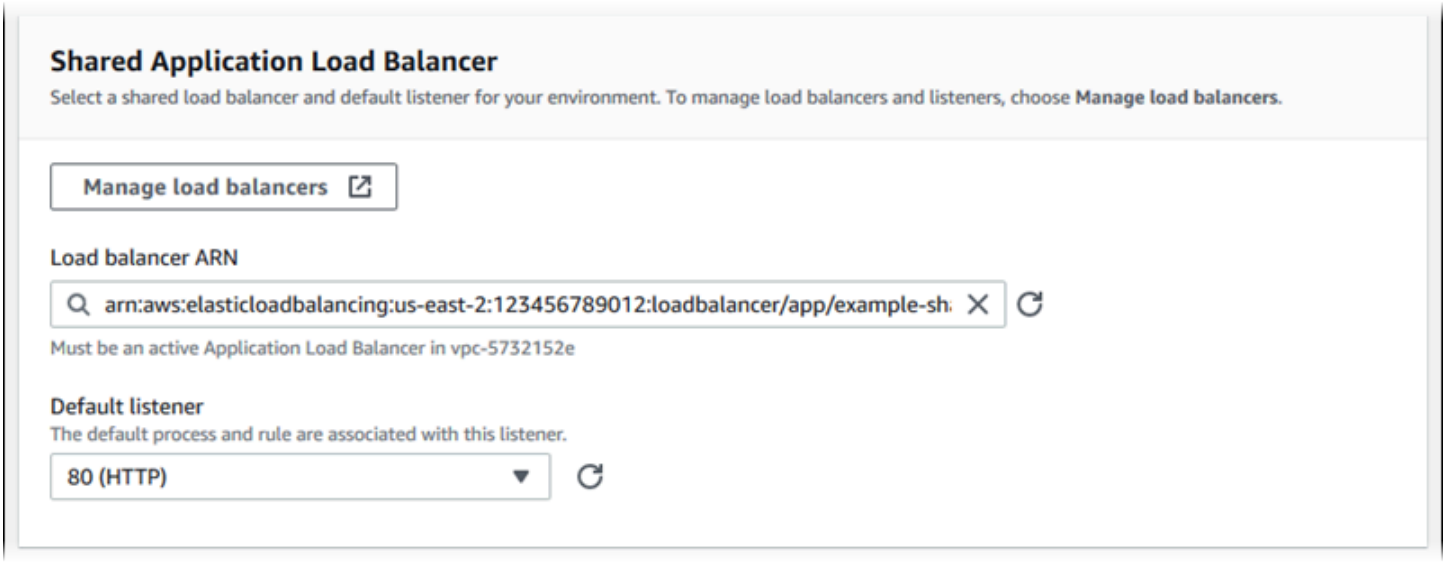
共享 Application Load Balancer

使用此區段可為您的環境選擇共享 Application Load Balancer，並設定預設流量路由。

在您可以在此設定共享 Application Load Balancer 之前，請先使用 Amazon EC2 在您的帳戶中至少定義一個用於共享的 Application Load Balancer，內含至少一個接聽程式。如果您尚未這麼做，可以選擇 Manage load balancers (管理負載平衡器)。Elastic Beanstalk 會在新的瀏覽器標籤中開啟 Amazon EC2 主控台。

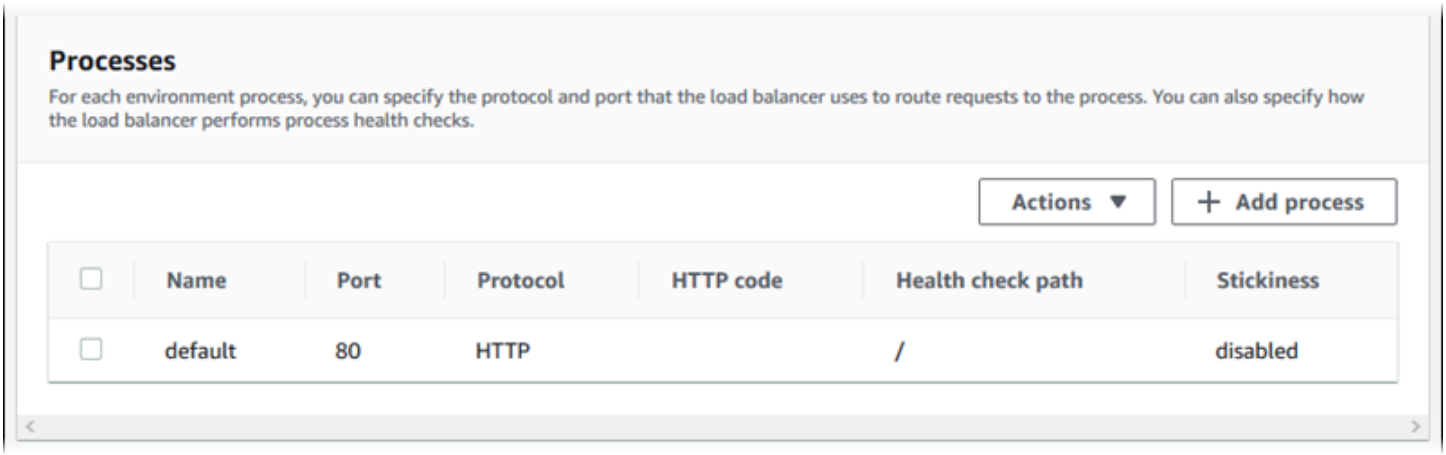
當您在 Elastic Beanstalk 外完成設定共享負載平衡器時，請在此主控台區段中設定下列設定：

- Load balancer ARN (負載平衡器 ARN) - 在此環境中使用的共享負載平衡器。從負載平衡器清單中選取，或輸入負載平衡器 Amazon Resource Name (ARN)。
- Default listener port (預設接聽程式連接埠) - 共享負載平衡器監聽的接聽程式連接埠。從現有的接聽程式連接埠清單中選取。來自此接聽程式的流量 (含有主機標頭中環境的 CNAME)，會路由傳送至此環境中的預設程序。



Processes

使用此名單為您的負載平衡器指定處理程序。程序為接聽程式路由流量的目標。一開始，此清單會顯示預設程序，該程序會從預設接聽程式接收流量。



設定現有處理程序

1. 選取其表項目旁邊的核取方塊，然後選擇 Actions (動作)，Edit (編輯)。

2. 使用環境處理程序對話方塊編輯設定，然後選擇儲存。

若要新增處理程序

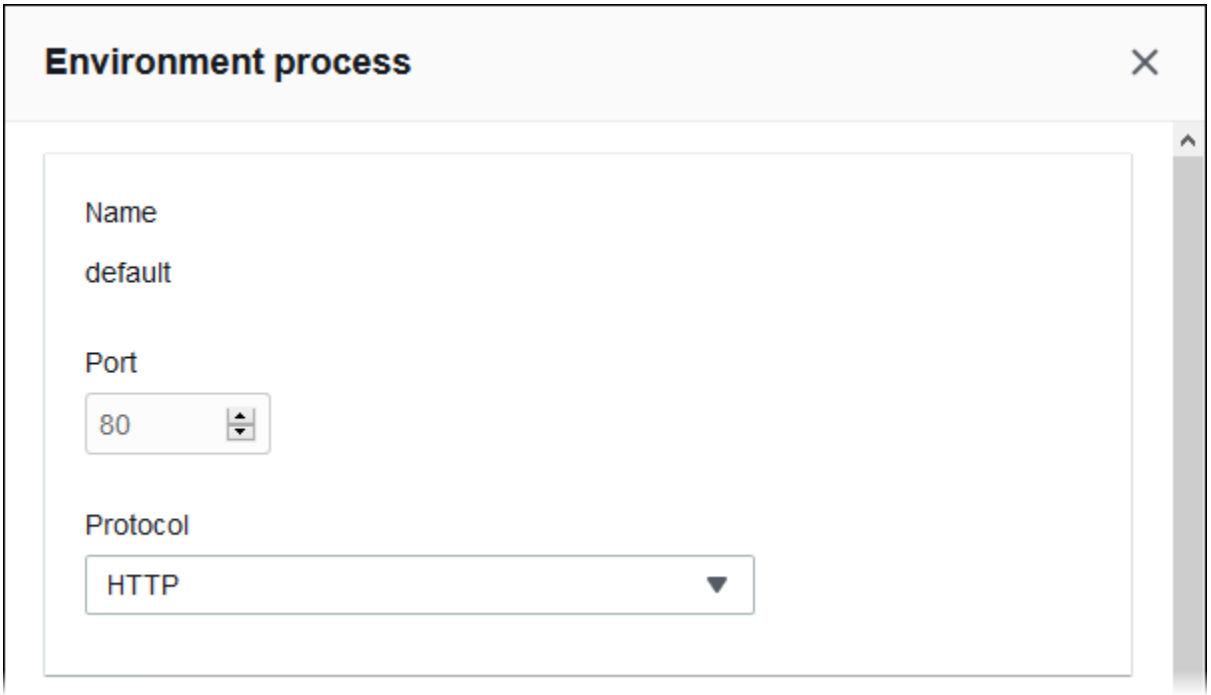
1. 選擇新增處理程序。
2. 在環境處理程序對話方塊中，設定您想要的設定，然後選擇新增。

Application Load Balancer 的環境程序對話方塊設定

- 定義
- 運作狀態檢查
- 工作階段

定義

使用這些設定來定義程序：它的 Name (名稱) 和其接聽請求的 Port (連接埠) 和 Protocol (協定)。

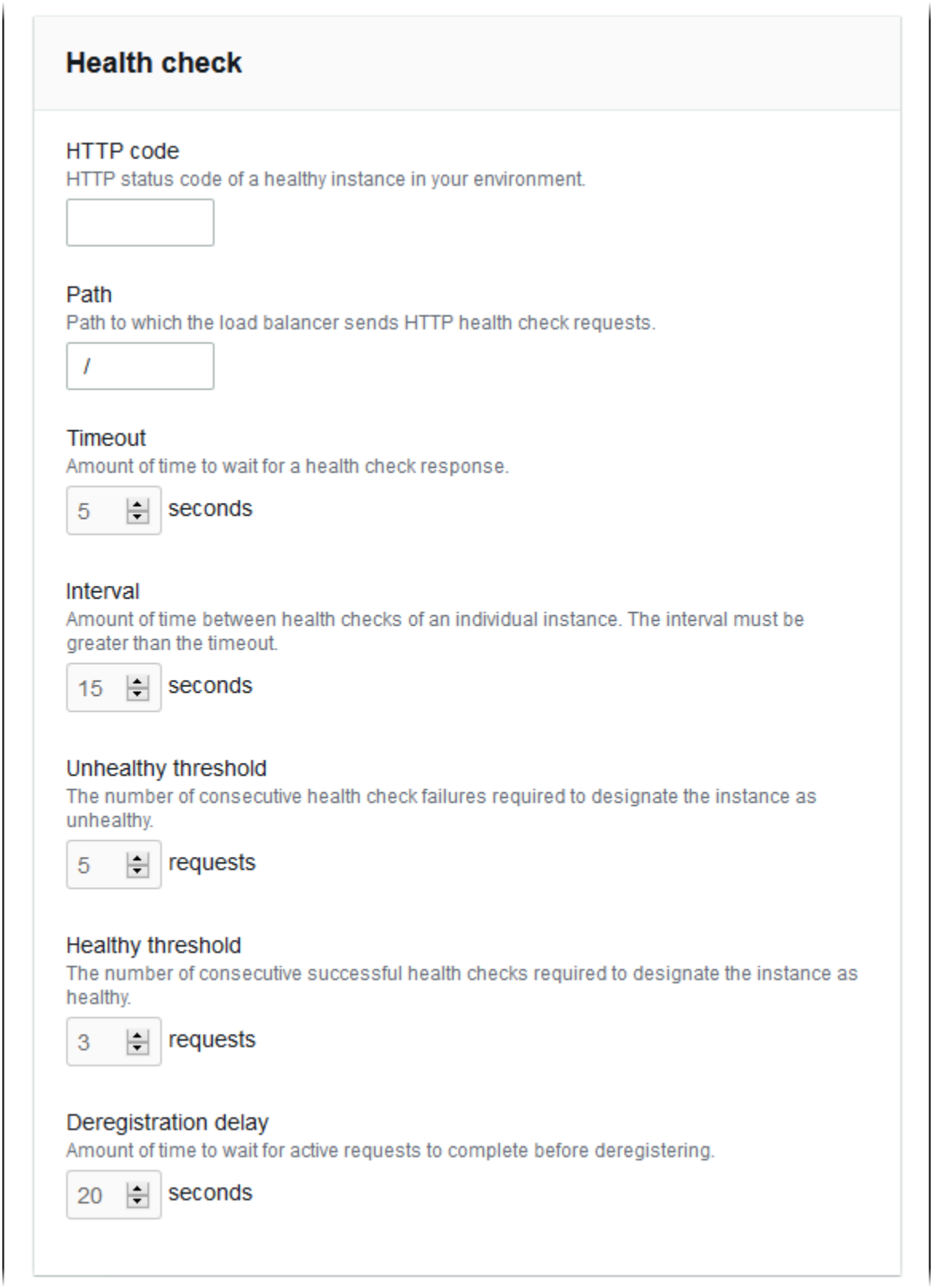


運作狀態檢查

使用以下設定以設定程序運作狀態檢查：

- HTTP code (HTTP 代碼) - 指定運作狀態良好程序的 HTTP 狀態碼。

- Path (路徑) - 程序的運作狀態檢查請求路徑。
- Timeout (逾時) - 等待運作狀態檢查回應的時間，以秒為單位。
- Interval (間隔) - 個別執行個體的每個運作狀態檢查之間的時間，以秒為單位。間隔必須大於逾時期間。
- Unhealthy threshold (狀態不良閾值)、Healthy threshold (運作良好閾值) - 在 Elastic Load Balancing 變更執行個體狀態之前，分別必須失敗或通過的運作狀態檢查數。
- Deregistration delay (取消註冊延遲) - 在取消註冊執行個體前，等待作用中請求完成的時間，以秒為單位。

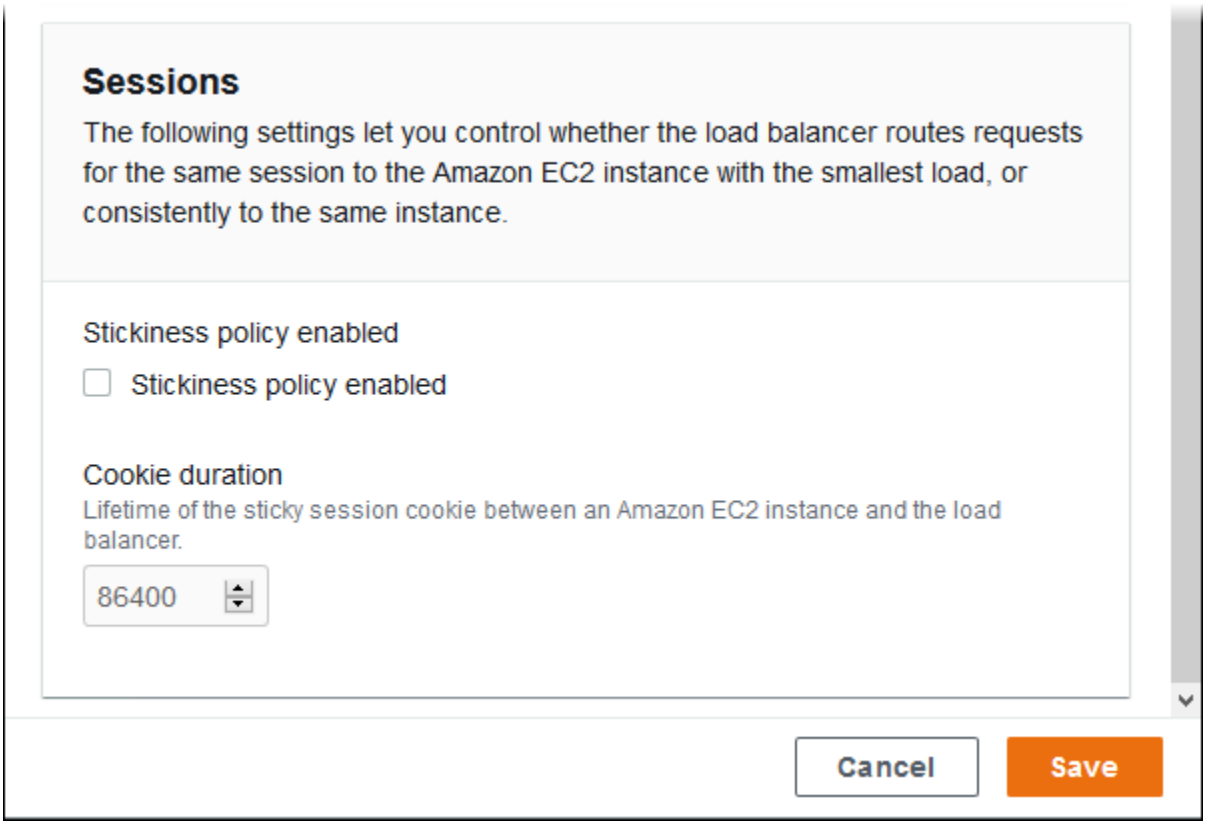

Health check
HTTP code
HTTP status code of a healthy instance in your environment.
Path
Path to which the load balancer sends HTTP health check requests.
/
Timeout
Amount of time to wait for a health check response.
5 seconds
Interval
Amount of time between health checks of an individual instance. The interval must be greater than the timeout.
15 seconds
Unhealthy threshold
The number of consecutive health check failures required to designate the instance as unhealthy.
5 requests
Healthy threshold
The number of consecutive successful health checks required to designate the instance as healthy.
3 requests
Deregistration delay
Amount of time to wait for active requests to complete before deregistering.
20 seconds

> **Note**
>
> Elastic Load Balancing 運作狀態檢查不會影響環境 Auto Scaling 群組的運作狀態檢查行為。Amazon EC2 Auto Scaling 不會自動取代未通過 Elastic Load Balancing 運作狀態檢查的執行個體，除非您手動設定 Amazon EC2 Auto Scaling 執行此項動作。如需詳細資訊，請參閱 Auto Scaling 運作狀態檢查設定。

如需關於運作狀態檢查，以及這些檢查如何影響您環境整體健全狀態的詳細資訊，請參閱基礎型運作狀態報告。

工作階段

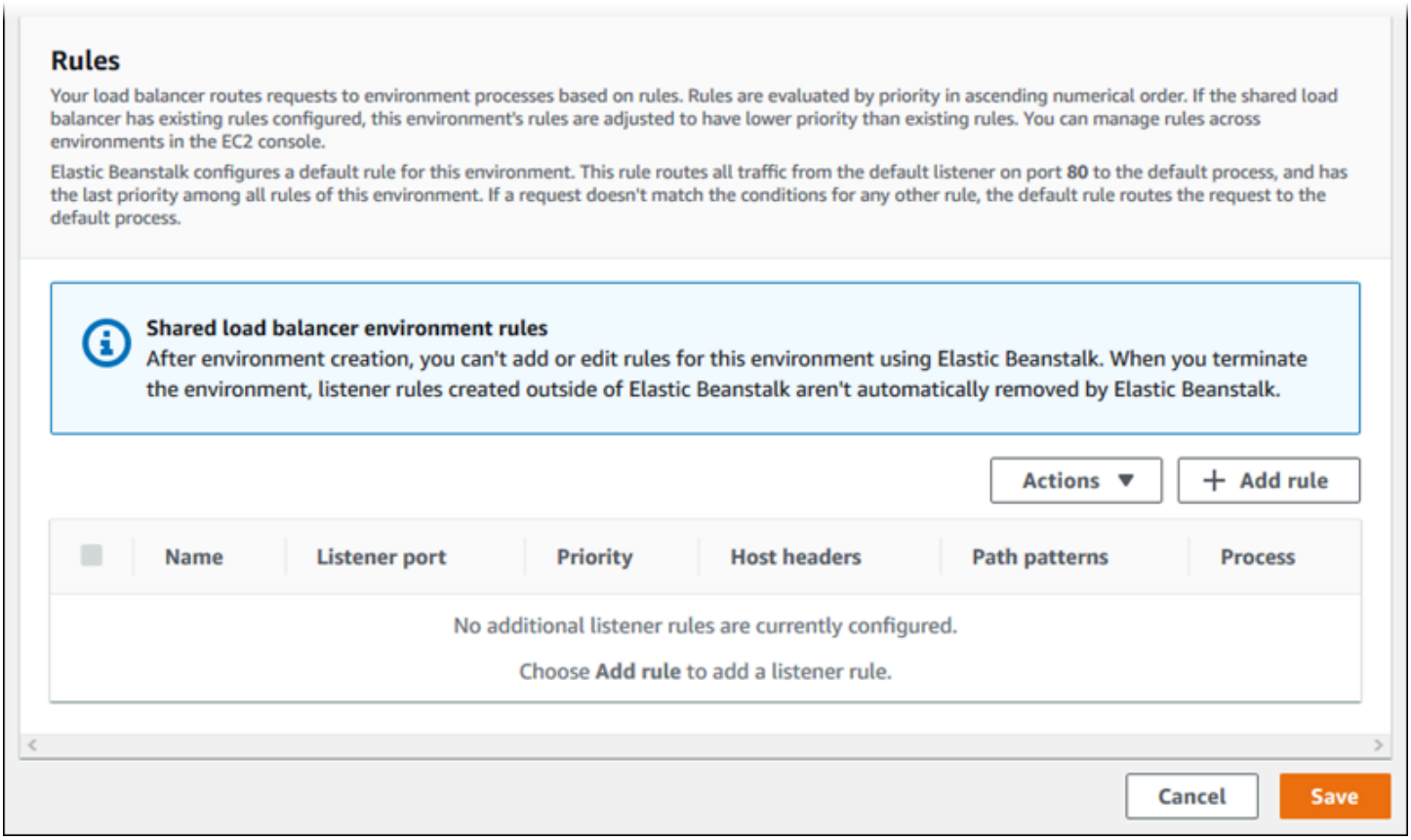
選取或清除 Stickiness policy enabled (啟用黏性政策) 方塊，以啟用或停用黏性工作階段。使用 Cookie duration (Cookie 持續時間) 設定黏性工作階段的持續時間，最長 **604800** 秒。



規則

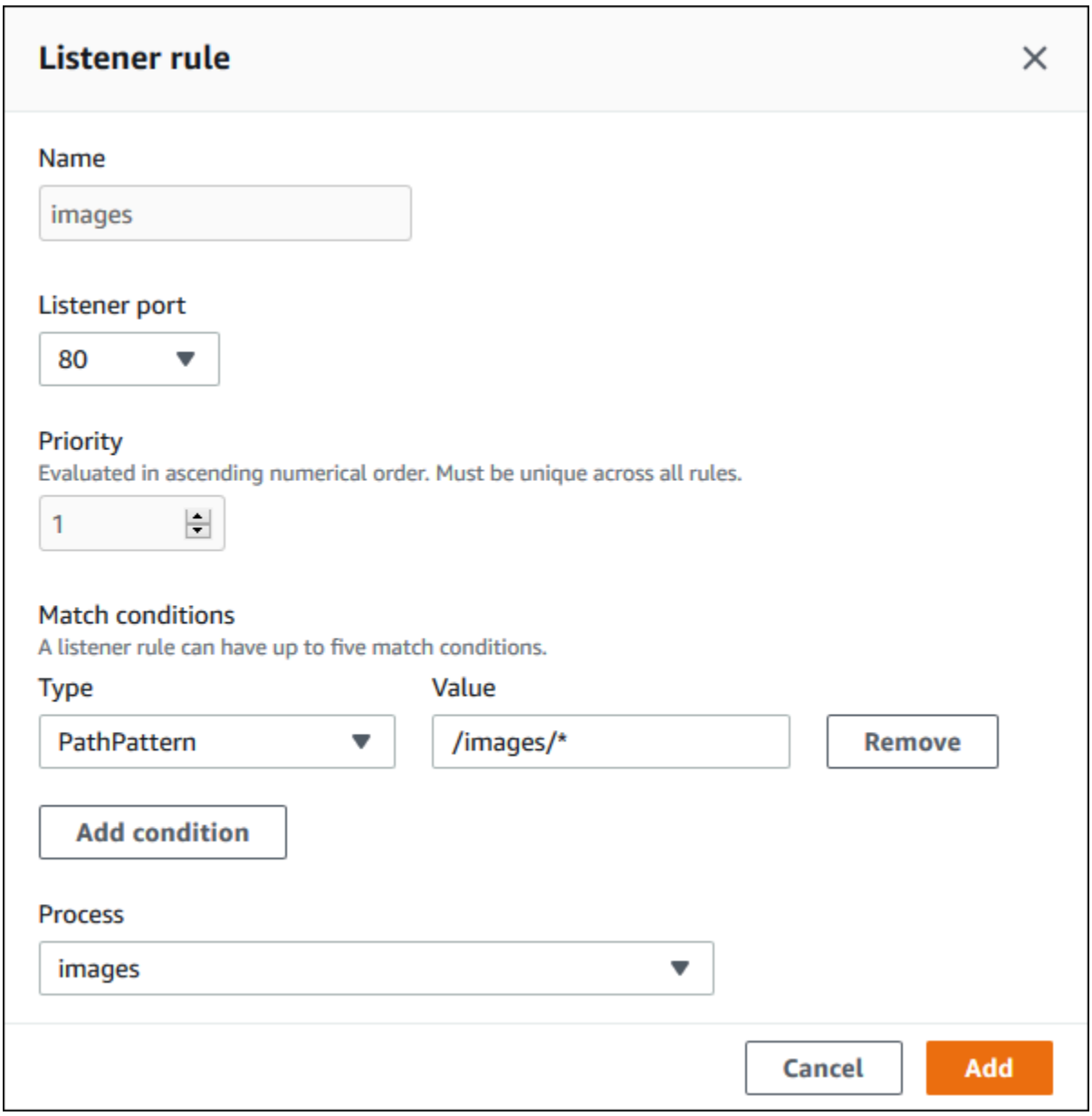
使用此清單為您的共享負載平衡器指定自訂接聽程式規則。規則映射在特定路徑模式接聽程式接收到的要求於目標程序中。每個接聽程式都可以有多個規則，將不同路徑上的請求路由傳送到共享接聽程式的不同環境執行個體上的不同處理程序。

規則有數字優先順序，判斷套用到傳入要求的優先順序。Elastic Beanstalk 會新增一個將所有預設接聽程式的流量都路由傳送至新環境預設程序的預設規則。預設規則的優先順序是最低的，如果相同的接聽程式沒有其它規則符合傳入的請求，則套用。一開始，如果您尚未新增自訂規則，清單會是空的。預設規則不會顯示。



您可以編輯現有規則的設定，或增加新的規則。若要開始編輯清單中的一個規則或新增一個規則，使用列於處理程序清單的相同步驟。Listener rule (接聽程式規則) 對話方塊開啟，依以下設定：

- Name (名稱) - 規則的名稱。
- Listener port (接聽程式連接埠) - 套用規則的接聽程式連接埠。
- Priority (優先順序) - 規則的優先順序。較低的號碼有較高的優先順序。接聽程式規則的優先順序必須是唯一的。Elastic Beanstalk 將規則優先順序視為共享環境間的相對優先順序，並在建立期間將其對應至絕對優先順序。
- Match conditions (比對條件) - 要套用規則的請求 URL 條件清單。條件有兩種類型： HostHeader (URL 的網域部分) 以及 PathPattern (URL 的路徑部分)。一個條件會保留給環境子網域，而且您最多可以加入四個條件。每個條件值的長度最多為 128 個字元，且可以包含萬用字元。
- Process (程序) - 負載平衡器路由傳送符合規則之請求的目標程序。



範例：針對安全的微型服務應用程式使用共享 Application Load Balancer

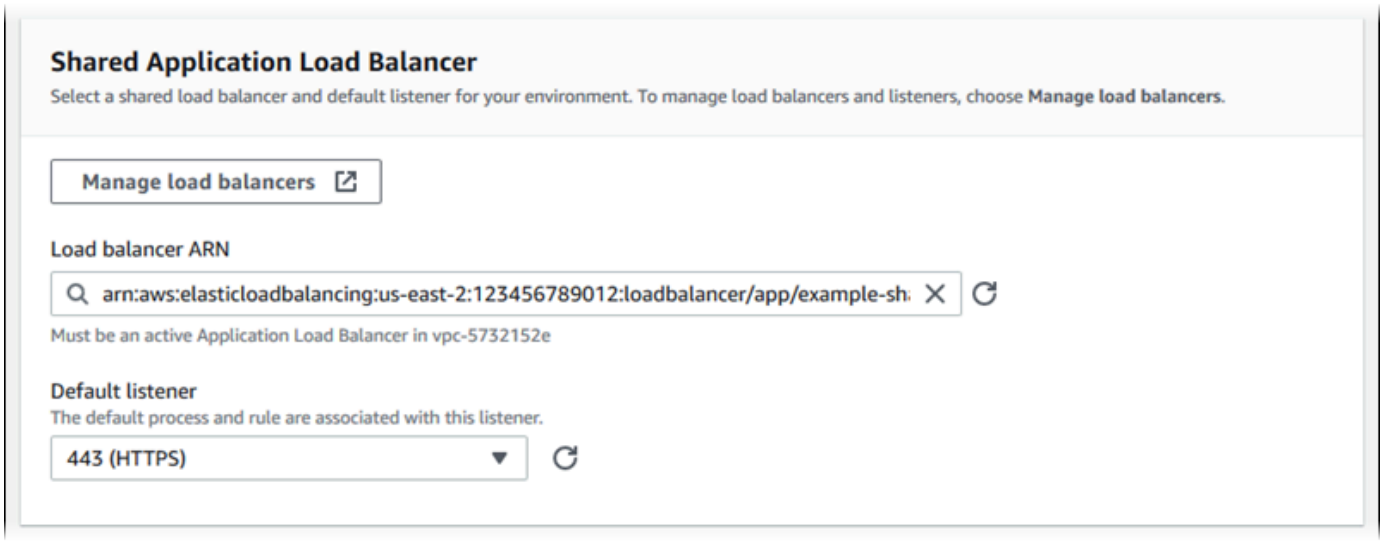
在此範例中，您的應用程式包含數個微型服務，每個服務都會以 Elastic Beanstalk 環境進行實作。此外，您還需要端對端流量加密。我們將示範其中一個微型服務環境，其中包含使用者請求的主要程序，以及處理系統管理請求的個別程序。

若要符合這些需求，請使用 Amazon EC2 建立您將在微型服務間共享的 Application Load Balancer。在連接埠 443 和 HTTPS 通訊協定上新增安全接聽程式。然後將多個 SSL 憑證新增到接聽程式中，每個微型服務網域各新增一個。如需建立 Application Load Balancer 和安全接聽程式的詳細資訊，請參閱《Application Load Balancer 使用者指南》中的建立 Application Load Balancer和建立適用於您 Application Load Balancer 的 HTTPS 接聽程式。

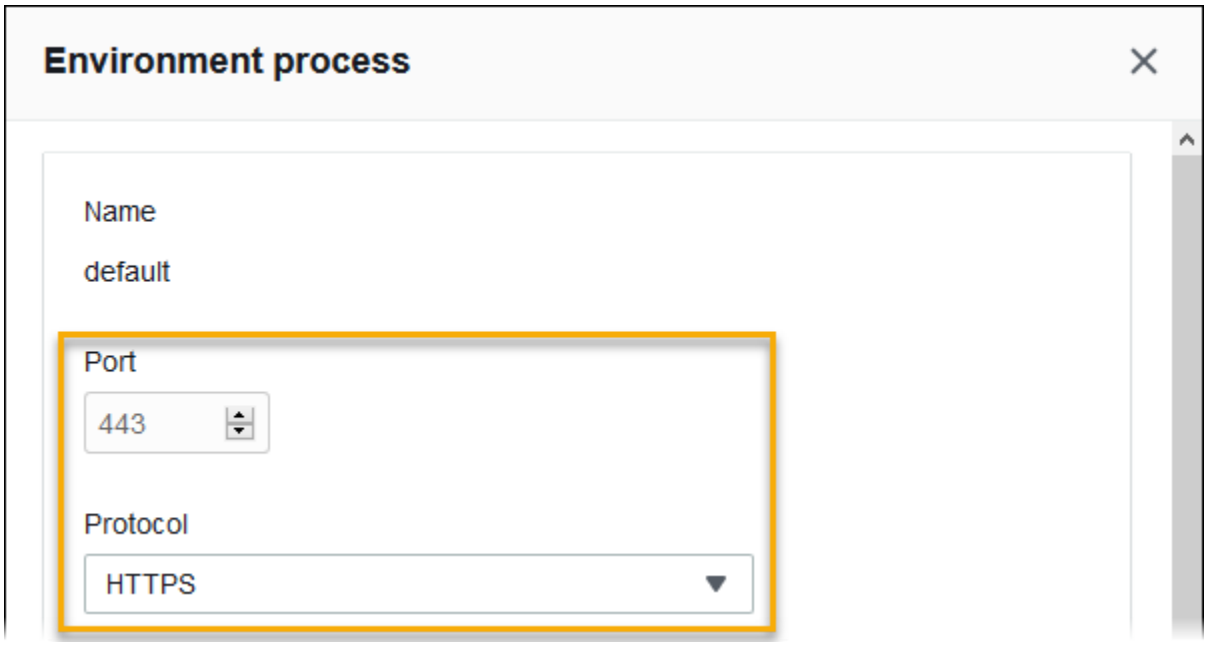
在 Elastic Beanstalk 中，將每個微型服務環境設定為使用共享 Application Load Balancer，並將預設接聽程式連接埠設定為 443。在此示範的特定環境中，指出預設程序在 HTTPS 上會接聽連接埠 443，並為不同路徑上的管理流量新增處理程序和接聽程式規則。

設定此範例的負載平衡器

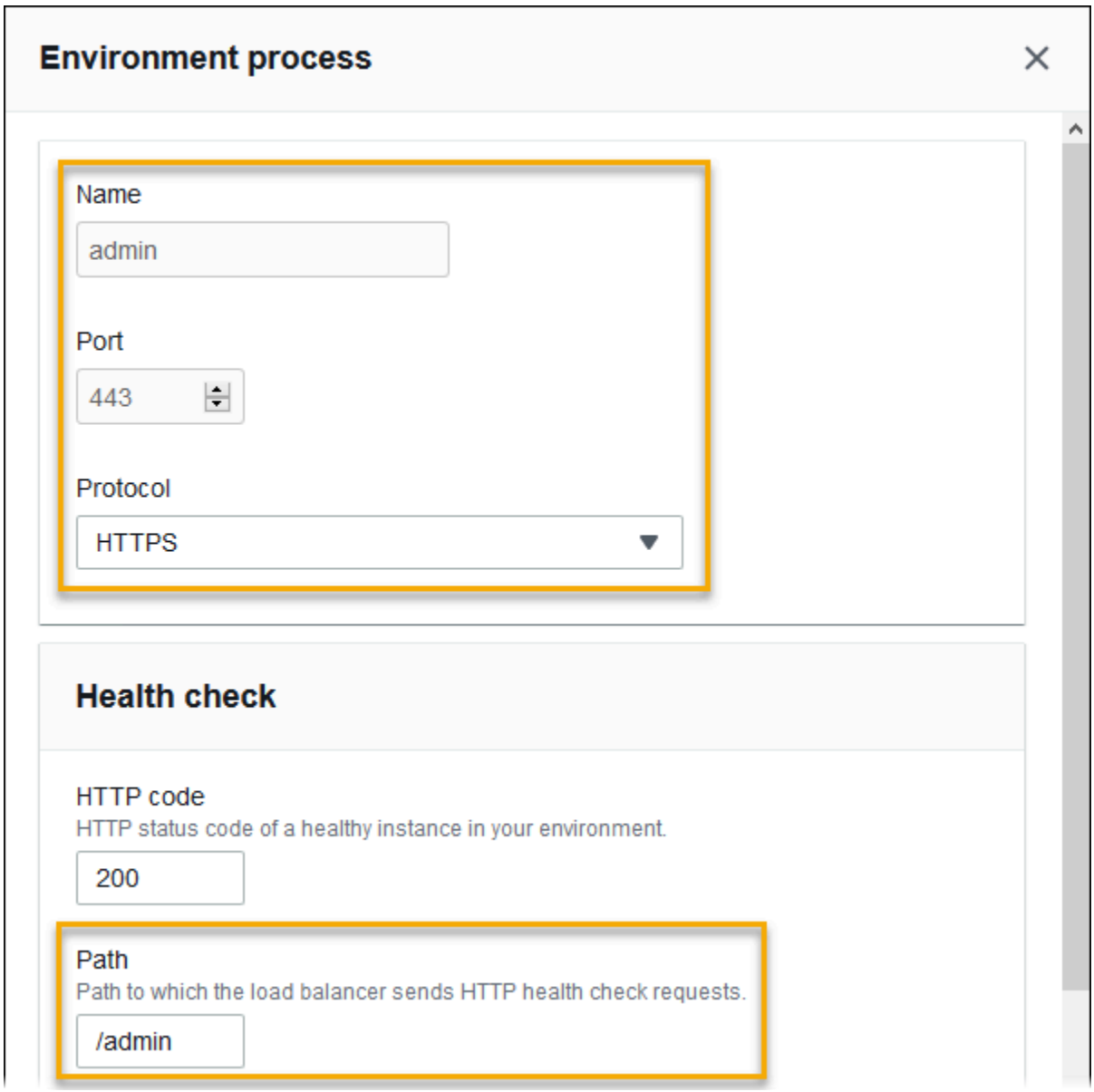
1. 在 Shared Application Load Balancer (共享 Application Load Balancer) 區段中，選取您的負載平衡器，然後在 Default listener port (預設接聽程式連接埠) 中選取 **443**。如果接聽程式連接埠是負載平衡器唯一擁有的接聽程式，則系統應該已經選取該接聽連接埠。



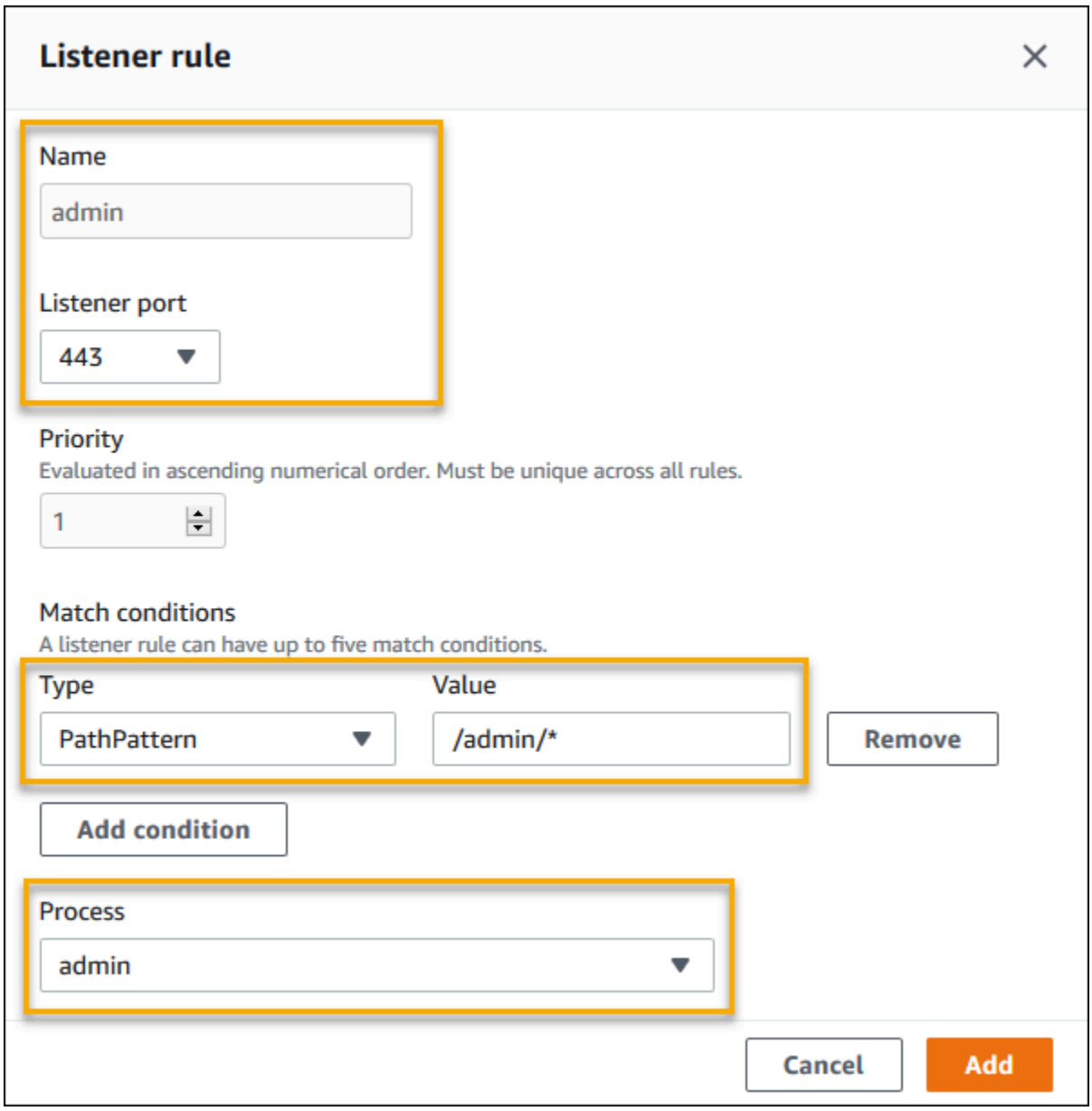
2. 設定預設程序至 HTTPS。選取預設程序，然後在 Actions (動作) 中選擇 Edit (編輯)。對於連接埠，輸入 **443**。對於通訊協定，請選取 **HTTPS**。



3. 新增管理程序。對於 Name (名稱)，輸入 **admin**。對於連接埠，輸入 **443**。對於通訊協定，請選取 **HTTPS**。對於運作狀態檢查下方的路徑，輸入 **/admin**。



4. 為管理流量新增一個規則。對於 Name (名稱)，輸入 **admin**。對於接聽程式連接埠，輸入 **443**。對於比對條件，新增一個帶有**/admin/***值的 PathPattern。對於程序，請選取 **admin**。



使用 EB CLI 設定共享 Application Load Balancer

在您執行 eb create 時，EB CLI 會提示您選擇負載平衡器類型。如果您選擇 `application` (預設值)，且您的帳戶至少有一個可共享的 Application Load Balancer，EB CLI 也會詢問您是否要使用共享 Application Load Balancer。如果您回答 **y**，系統也會提示您選取負載平衡器和預設連接埠。

```
$ eb create
Enter Environment Name
(default is my-app): test-env
Enter DNS CNAME prefix
(default is my-app): test-env-DLW24ED23SF
```

```
Select a load balancer type
1) classic
2) application
3) network
(default is 2):

Your account has one or more sharable load balancers. Would you like your new
 environment to use a shared load balancer?(y/N) y

Select a shared load balancer
1)MySharedALB1 - arn:aws:elasticloadbalancing:us-east-2:123456789012:loadbalancer/app/
MySharedALB1/6d69caa75b15d46e
2)MySharedALB2 - arn:aws:elasticloadbalancing:us-east-2:123456789012:loadbalancer/app/
MySharedALB2/e574ea4c37ad2ec8
(default is 1): 2

Select a listener port for your shared load balancer
1) 80
2) 100
3) 443
(default is 1): 3
```

您也可以使用命令選項指定共享負載平衡器。

```
$ eb create test-env --elb-type application --shared-lb MySharedALB2 --shared-lb-
port 443
```

共享 Application Load Balancer 命名空間

您可以在下列命名空間中找到與共享 Application Load Balancer 相關的設定：

- aws:elasticbeanstalk:environment - 選擇環境的負載平衡器類型，並告知 Elastic Beanstalk 您將使用共享負載平衡器。

  您無法在組態檔案 (.Ebextensions) 中設定這兩個選項。
- aws:elbv2:loadbalancer - 設定共享 Application Load Balancer ARN 和安全群組。
- aws:elbv2:listener - 藉由列出接聽程式規則，將共享 Application Load Balancer 的接聽程式與環境程序產生關聯。
- aws:elbv2:listenerrule - 設定接聽程式規則，根據請求的路徑，將流量路由傳送至不同程序。專用和共享的規則對 Application Load Balancer 都是唯一的規則。

- aws:elasticbeanstalk:environment:process - 設定運作狀態檢查，並針對執行於環境執行個體的程序，指定連接埠和通訊協定。

Example .ebextensions/application-load-balancer-shared.config

若要開始使用共享 Application Load Balancer，請使用 Elastic Beanstalk 主控台、EB CLI 或 API，將負載平衡器類型設定為 application 並選擇使用共享負載平衡器。使用組態檔案來設定共享負載平衡器。

```
option_settings:
  aws:elbv2:loadbalancer:
    SharedLoadBalancer: arn:aws:elasticloadbalancing:us-
east-2:123456789012:loadbalancer/app/MySharedALB2/e574ea4c37ad2ec8
```

> **Note**
>
> 您只能在建立環境時設定此選項。

Example .ebextensions/alb-shared-secure-listener.config

下列組態檔案會為共享負載平衡器選取連接埠 443 上的預設安全接聽程式，並將預設程序設定為接聽連接埠 443。

```
option_settings:
  aws:elbv2:loadbalancer:
    SharedLoadBalancer: arn:aws:elasticloadbalancing:us-
east-2:123456789012:loadbalancer/app/MySharedALB2/e574ea4c37ad2ec8
  aws:elbv2:listener:443:
    rules: default
  aws:elasticbeanstalk:environment:process:default:
    Port: '443'
    Protocol: HTTPS
```

Example .ebextensions/alb-shared-admin-rule.config

下列組態檔案會以上一個範例為基礎並新增規則，該規則會將流量以 /admin 的請求路徑路由傳送至在連接埠 4443 上接聽名為 admin 的處理程序。

```
option_settings:
```

```
  aws:elbv2:loadbalancer:
    SharedLoadBalancer: arn:aws:elasticloadbalancing:us-
east-2:123456789012:loadbalancer/app/MySharedALB2/e574ea4c37ad2ec8
  aws:elbv2:listener:443:
    rules: default,admin
  aws:elasticbeanstalk:environment:process:default:
    Port: '443'
    Protocol: HTTPS
  aws:elasticbeanstalk:environment:process:admin:
    HealthCheckPath: /admin
    Port: '4443'
    Protocol: HTTPS
  aws:elbv2:listenerrule:admin:
    PathPatterns: /admin/*
    Priority: 1
    Process: admin
```

設定 Network Load Balancer

若啟用負載平衡，您的 AWS Elastic Beanstalk 環境會配備 Elastic Load Balancing 負載平衡器，將流量分配到您環境中的執行個體。Elastic Load Balancing 支援多種類型的負載平衡器。若要了解這些資訊，請參閱 Elastic Load Balancing 使用者指南。Elastic Beanstalk 可以為您建立負載平衡器，或讓您指定已建立的共享負載平衡器。

本主題說明 Elastic Beanstalk 建立並專用於您環境的 Network Load Balancer 的組態。如需設定 Elastic Beanstalk 支援之所有負載平衡器類型的詳細資訊，請參閱您的 Elastic Beanstalk 環境的負載平衡器。

> **Note**
>
> 您可以選擇您的環境只會在環境建立時使用的負載平衡器類型。您可以變更設定，以管理執行之環境的負載平衡器行為。

簡介

透過 Network Load Balancer，預設接聽程式會接受連接埠 80 上的 TCP 請求，並將這些請求分配到您環境中的執行個體。您可以設定運作狀態檢查行為、設定接聽程式連接埠，或在另一個連接埠上新增接聽程式。

> **Note**
>
> 有別於 Classic Load Balancer 或 Application Load Balancer，Network Load Balancer 無法擁有應用程式層 (第 7 層) HTTP 或 HTTPS 接聽程式。它僅支援傳輸層 (第 4 層) TCP 接聽程式。HTTP 和 HTTPS 流量可透過 TCP 路由到您的環境。若要在 Web 用戶端與您的環境之間建立安全的 HTTPS 連線，請在環境中的執行個體上安裝自我簽署的憑證，並設定執行個體監聽適當的連接埠 (通常是 443) 及終止的 HTTPS 連線。每個組態視平台而有所不同。如需說明，請參閱 於執行個體設定您的應用程式以終止 HTTPS 連線。然後，設定 Network Load Balancer 以新增接聽程式，對應到監聽適當連接埠的程序。

Network Load Balancer 支援主動運作狀態檢查。這些檢查是根據至根 (/) 路徑的訊息。此外，Network Load Balancer 還支援被動運作狀態檢查。能夠自動偵測故障的後端執行個體，並僅將流量路由至運作狀態良好的執行個體。

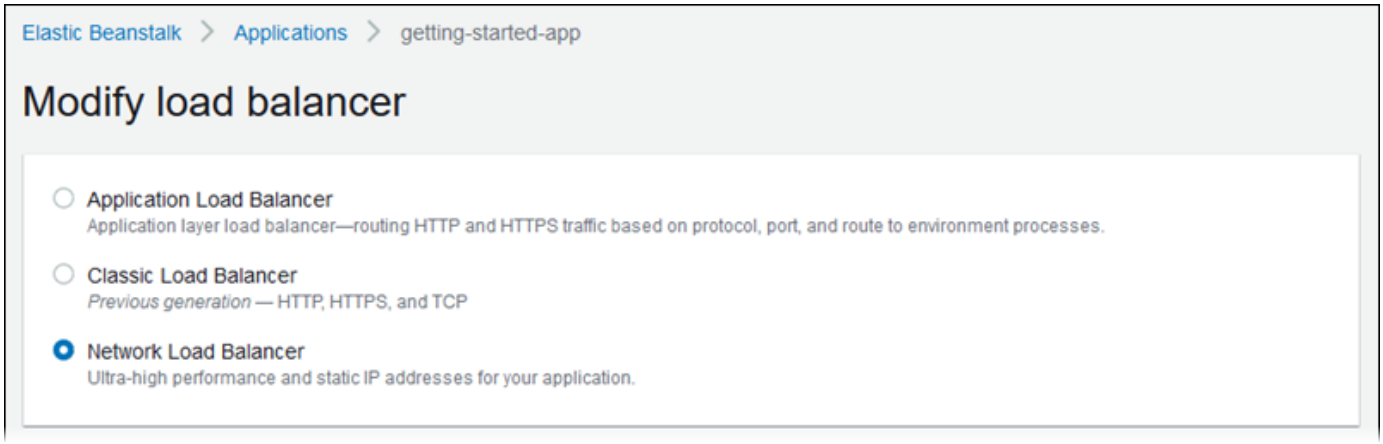
使用 Elastic Beanstalk 主控台設定 Network Load Balancer

在環境建立期間或之後在環境執行時，您可以使用 Elastic Beanstalk 主控台設定 Network Load Balancer 的接聽程式和程序。

若要在環境建立期間於 Elastic Beanstalk 主控台設定 Network Load Balancer

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Environments (環境)。
3. 選擇 Create a new environment (建立新環境) 以開始建立您的環境。
4. 在精靈的主頁上，在選擇 Create environment (建立環境) 之前，選擇 Configure more options (設定更多選項)。
5. 選擇 High availability (高可用性) 組態預設。

   或者，在 Capacity (容量) 組態類別中，設定Load balanced (負載平衡器) 環境類型。如需詳細資訊，請參閱容量。
6. 在 Load balancer (負載平衡器) 組態類別中，選擇 Edit (編輯)。
7. 選取 Network Load Balancer 選項 (如果尚未選取)。



8. 根據環境所需，變更任何 Network Load Balancer 組態。
9. 選擇 Save (儲存)，然後針對您環境所需對其他組態做變更。
10. 選擇 Create environment (建立環境)。

若要在 Elastic Beanstalk 主控台設定執行中環境的 Network Load Balancer

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇 Configuration (組態)。
4. 在 Load balancer (負載平衡器) 組態類別中，選擇 Edit (編輯)。

> **Note**
>
> 如果 Load balancer (負載平衡器) 組態類別沒有 Edit (編輯) 按鈕，您的環境便沒有負載平衡器。若要了解如何設定一個負載平衡器，請參閱 變更環境類型。

5. 根據環境所需，變更 Network Load Balancer 組態。
6. 若要儲存變更，請選擇頁面底部的儲存變更。

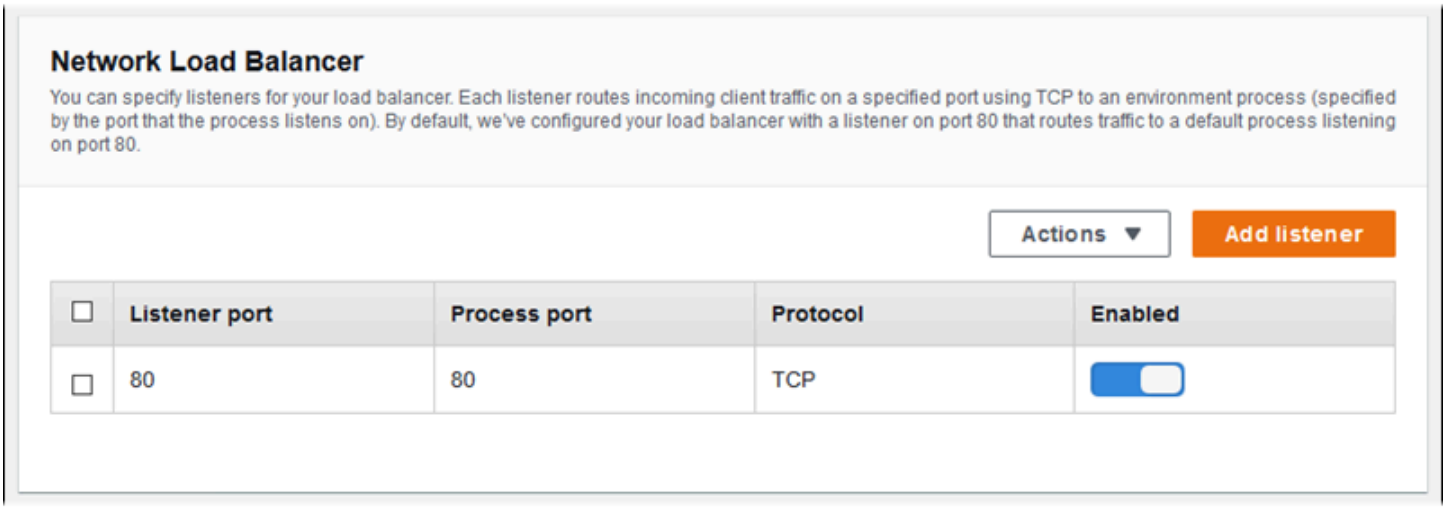
Network Load Balancer 設定

- 接聽程式

- Processes

接聽程式

使用此名單為您的負載平衡器指定接聽程式。每個接聽程式會將傳入用戶端流量路由傳送到您執行個體上的程序。一開始，清單顯示預設的接聽程式，它會將連接埠 80 上傳入的流量路由至接聽自接埠 80 且名為 default (預設) 的程序。



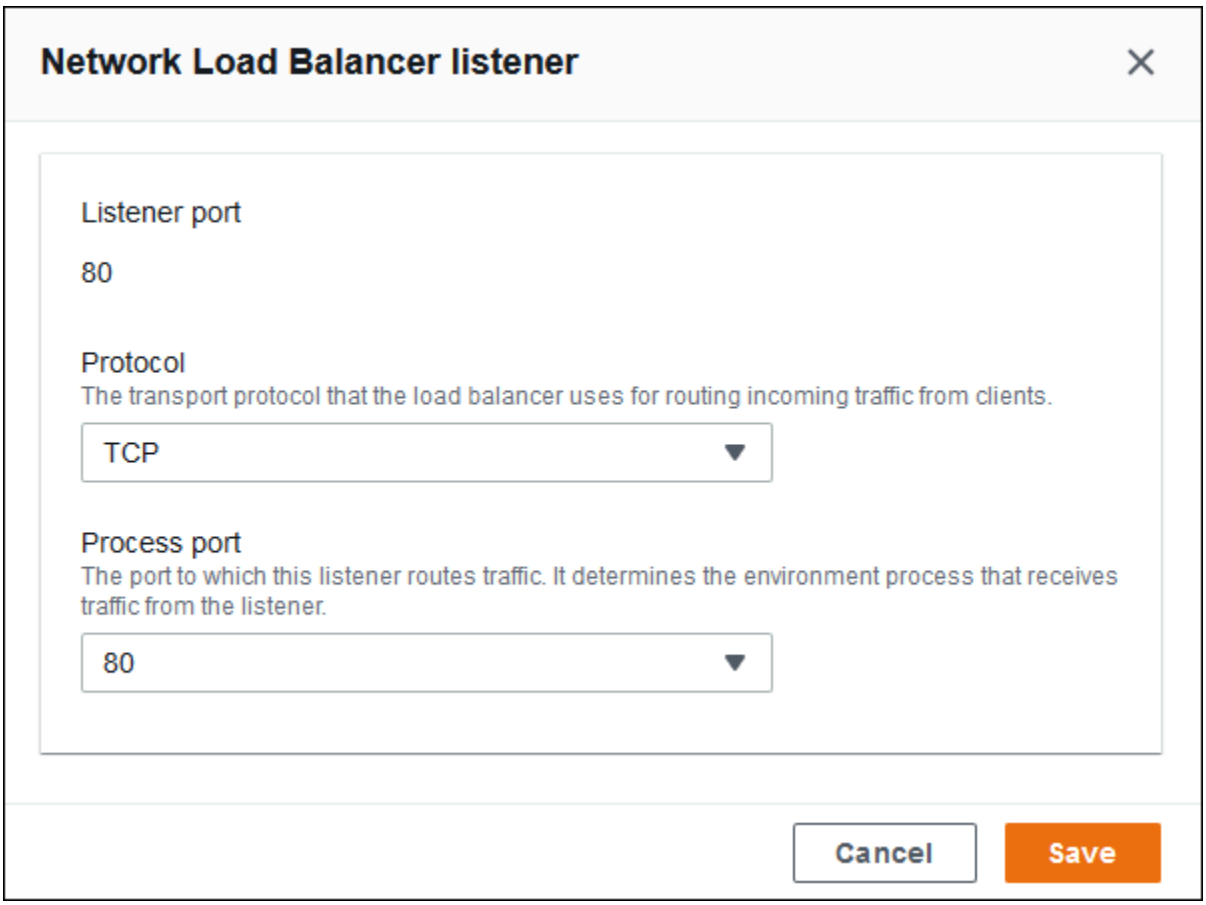
要設定現有的接聽程式

1. 選取其表項目旁邊的核取方塊，然後選擇 Actions (動作)，Edit (編輯)。
2. 使用 Network Load Balancer listener (Network Load Balancer 接聽程式) 對話方塊編輯設定，然後選擇 Save (儲存)。

加入接聽程式

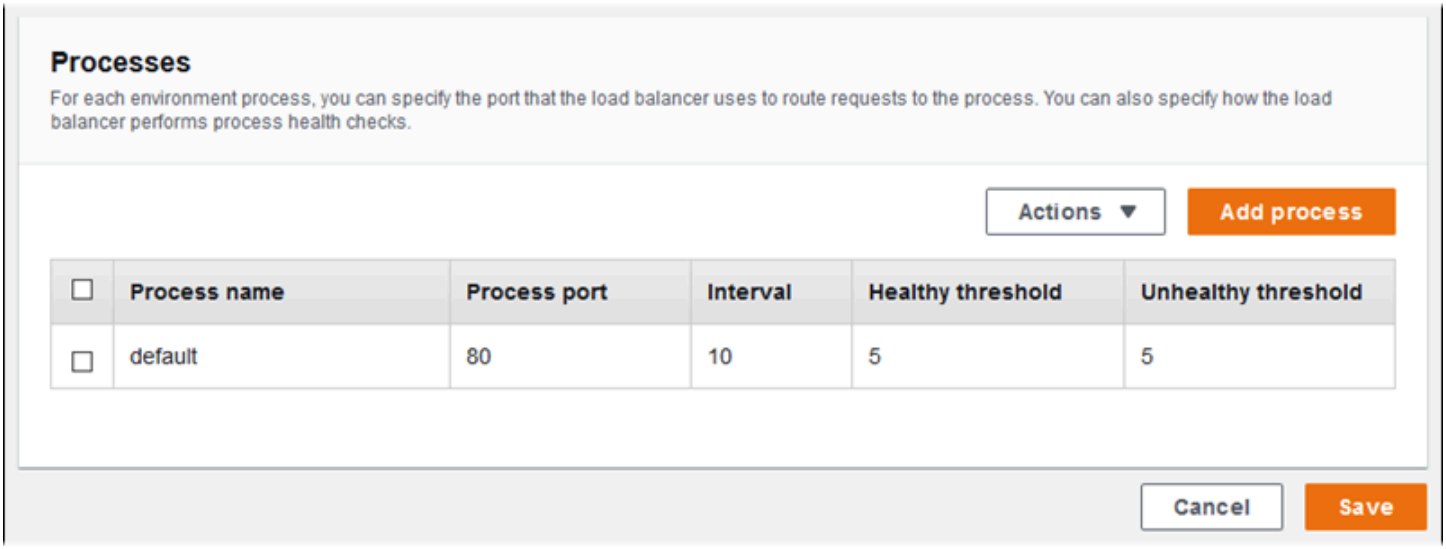
1. 選擇 Add listener (新增接聽程式)。
2. 在 Network Load Balancer listener (Network Load Balancer 接聽程式) 對話方塊中，設定所需的設定，然後選擇 Add (新增)。

使用 Network Load Balancer listener (Network Load Balancer 接聽程式) 對話方塊，設定接聽程式監聽流量的連接埠，然後選擇您要將流量路由至哪個程序 (以接聽程式監聽的連接埠指定)。



Processes

使用此名單為您的負載平衡器指定程序。程序為接聽程式路由流量的目標。每個接聽程式會將傳入用戶端流量路由傳送到您執行個體上的程序。一開始，清單顯示接聽於連接埠 80 上傳入的流量的預設程序。



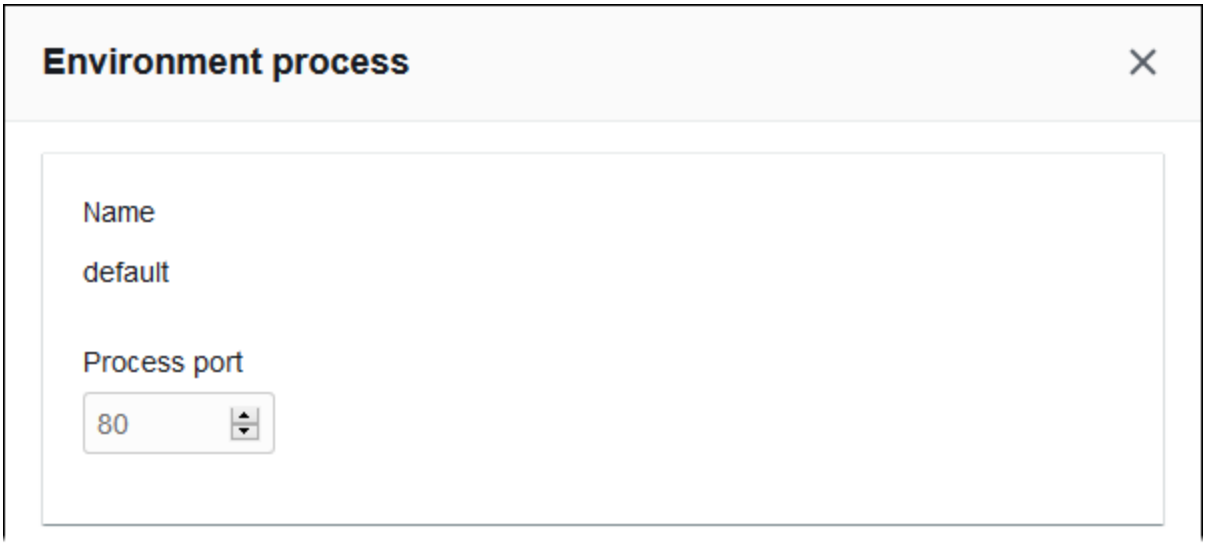
您可以編輯現有程序的設定，或增加新的程序。若要開始編輯清單中的一個程序或新增一個程序，使用列於接聽程式清單的相同步驟。Environment process (環境程序) 對話方塊開啟。

Network Load Balancer 的環境程序對話方塊設定

- 定義
- 運作狀態檢查

定義

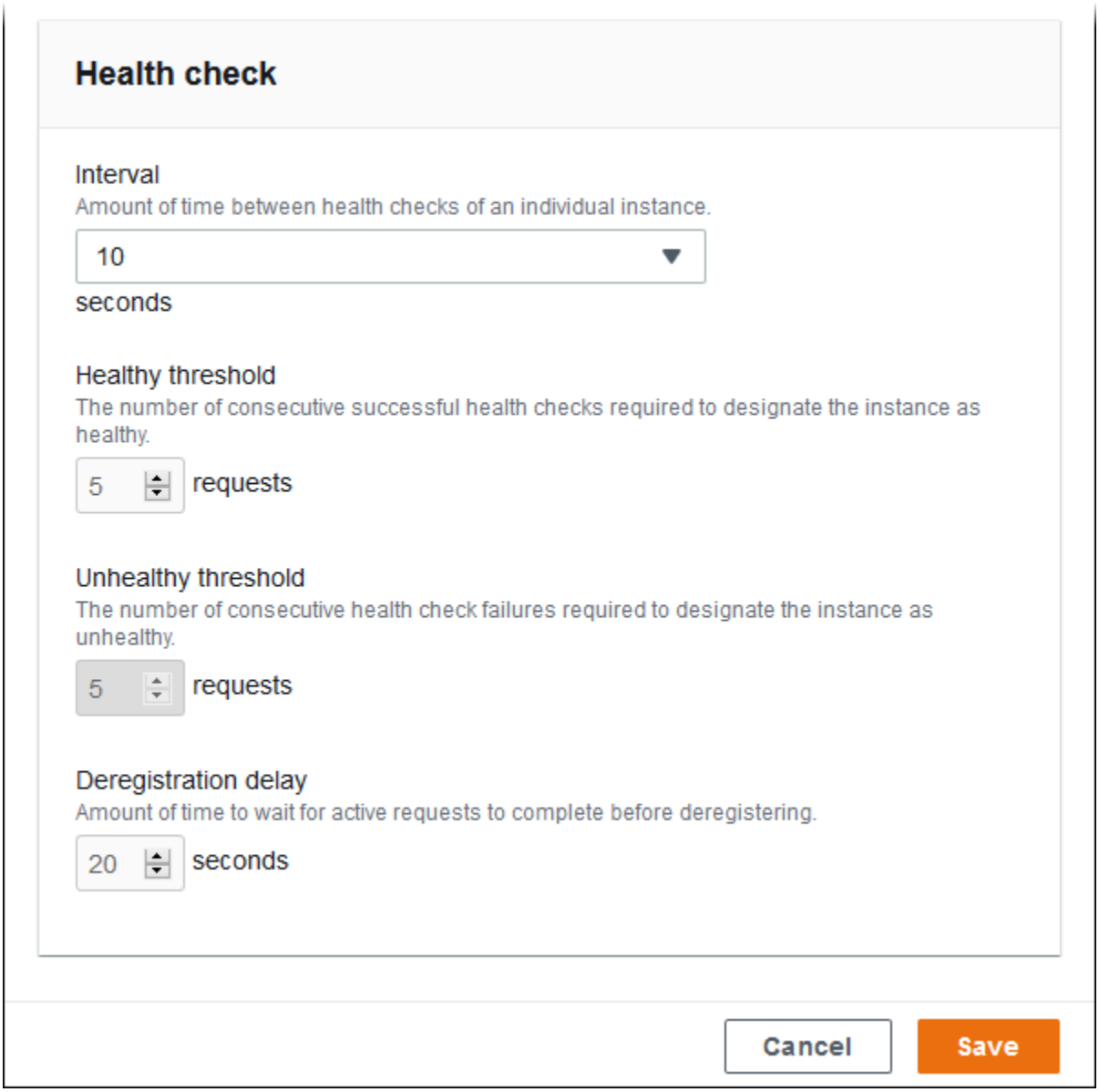
使用這些設定來定義程序：它的 Name (名稱) 和其接聽請求的 Process Port (程序連接埠)。



運作狀態檢查

使用以下設定以設定程序運作狀態檢查：

- Interval (間隔) - 個別執行個體的每個運作狀態檢查之間的時間，以秒為單位。
- Healthy threshold (運作良好閾值) - 在 Elastic Load Balancing 變更執行個體狀態之前，必須通過的運作狀態檢查數。(就 Network Load Balancer 而言，Unhealthy threshold (狀態不良閾值) 是唯讀設定，一律等於運作良好閾值)。
- Deregistration delay (取消註冊延遲) - 在取消註冊執行個體前，等待作用中請求完成的時間，以秒為單位。



> **Note**
>
> Elastic Load Balancing 運作狀態檢查不會影響環境 Auto Scaling 群組的運作狀態檢查行為。Amazon EC2 Auto Scaling 將不會自動取代未通過 Elastic Load Balancing 運作狀態檢查的執行個體，除非您手動設定 Amazon EC2 Auto Scaling 執行此項動作。如需詳細資訊，請參閱 Auto Scaling 運作狀態檢查設定。

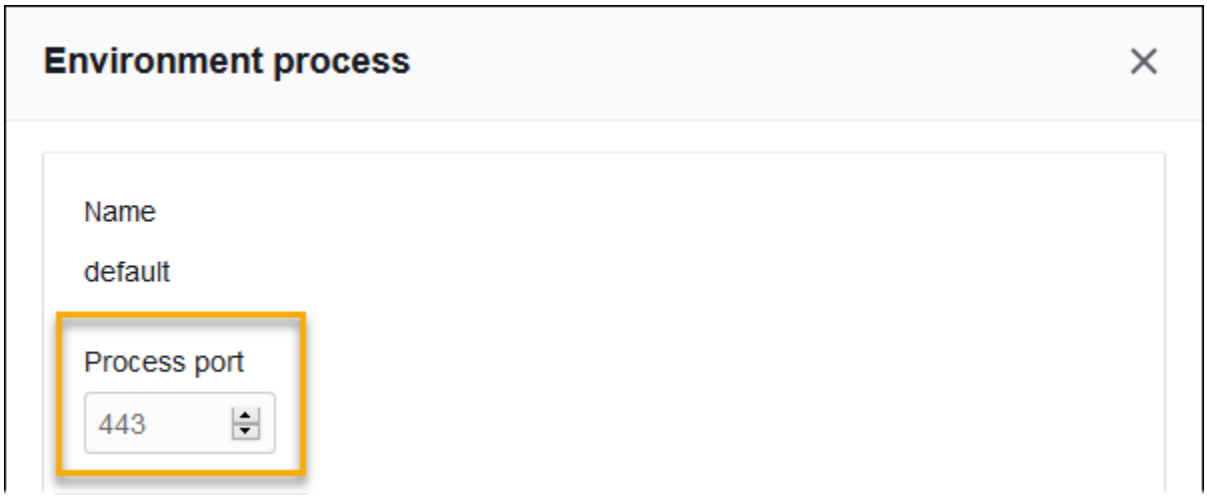
如需關於運作狀態檢查，以及這些檢查如何影響您環境整體健全狀態的詳細資訊，請參閱基礎型運作狀態報告。

範例：Network Load Balancer 用於具有端對端加密的環境

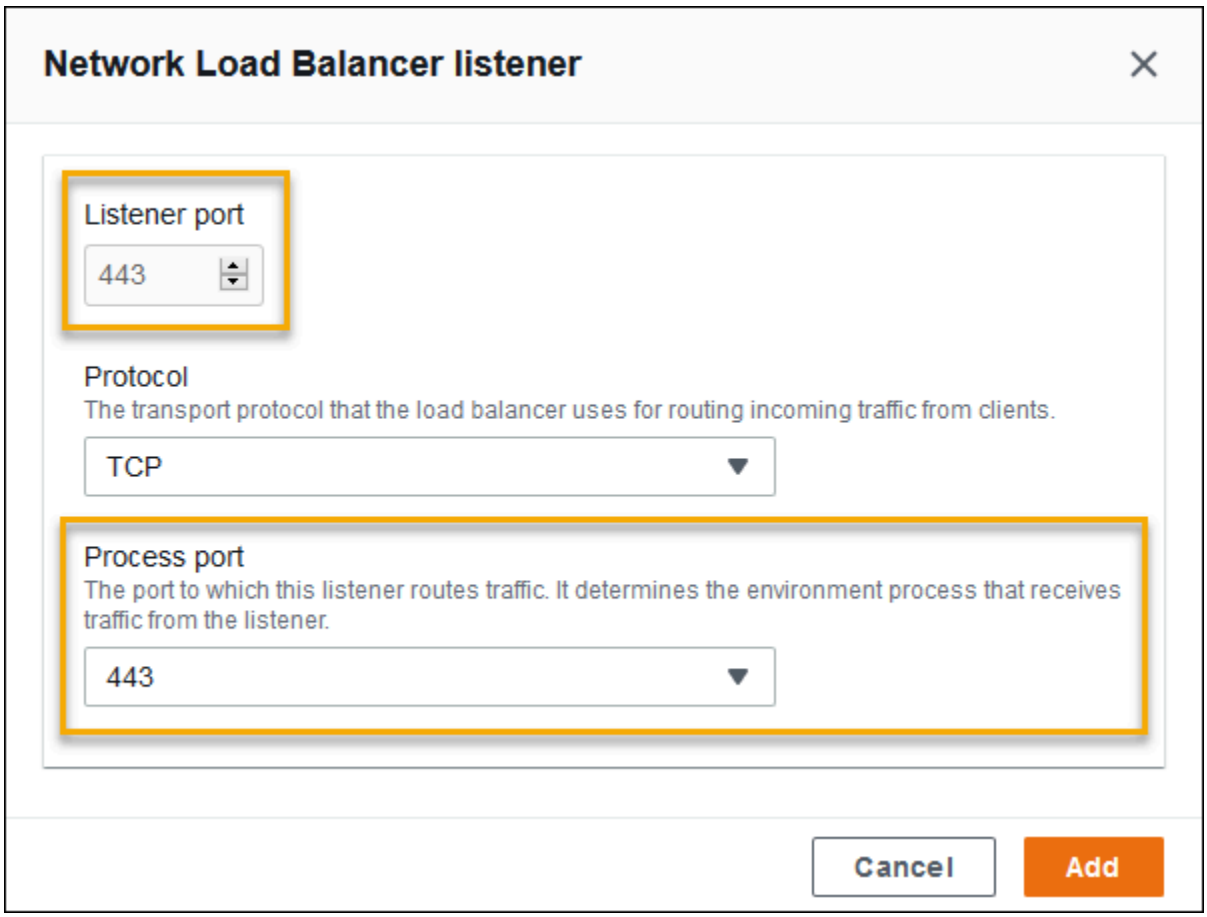
在此範例中，您的應用程式需要端對端流量加密。若要設定您環境的 Network Load Balancer 以符合這些要求，您必須將預設程序設定為監聽連接埠 443，新增接聽程式至連接埠 443 來將流量路由傳送到預設程序，然後停用預設接聽程式。

要設定負載平衡器在這個範例

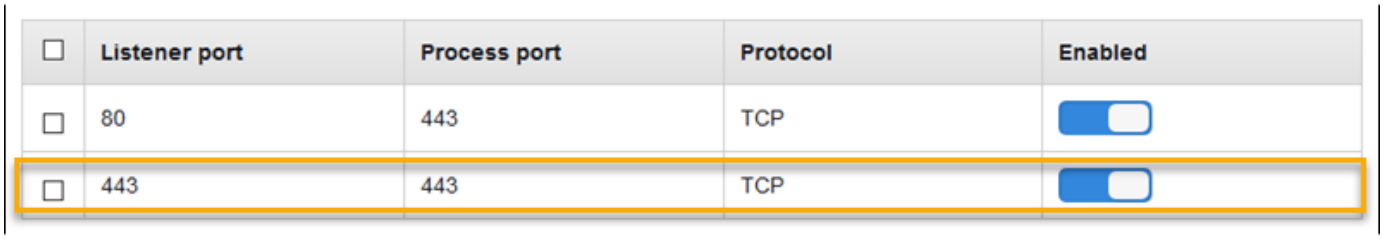
1. 設定預設程序。選取預設程序，然後在 Actions (動作) 中選擇 Edit (編輯)。在 Process Port (程序連接埠) 欄位中輸入 443。



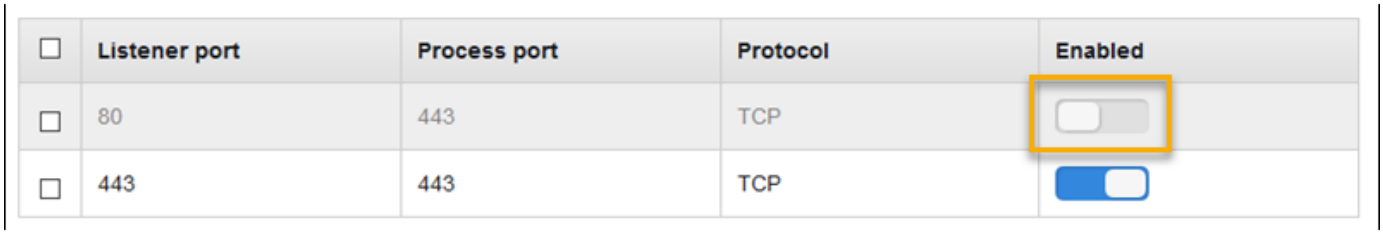
2. 新增連接埠 443 接聽程式。新增一個新的接聽程式。在 Listener port (接聽程式連接埠) 中輸入 443。確定已在 Process port (程序連接埠) 中選取 443。



您現在可以查看您清單上的額外接聽程式。



3. 停用預設的連接埠 80 接聽程式。針對預設的接聽程式，關閉 Enabled (已啟用) 選項。



使用 EB CLI 設定 Network Load Balancer

在您執行 eb create 時，EB CLI 會提示您選擇負載平衡器類型。

```
$ eb create
Enter Environment Name
(default is my-app): test-env
Enter DNS CNAME prefix
(default is my-app): test-env-DLW24ED23SF

Select a load balancer type
1) classic
2) application
3) network
(default is 1): 3
```

您亦可透過 `--elb-type` 選項，指定負載平衡器類型。

```
$ eb create test-env --elb-type network
```

Network Load Balancer 命名空間

您可以在下列命名空間中找到與 Network Load Balancer 相關的設定：

- `aws:elasticbeanstalk:environment` - 選擇環境的負載平衡器類型。Network Load Balancer 的值為 `network`。
- `aws:elbv2:listener` - 在 Network Load Balancer 上設定接聽程式。這些設定對應至 Classic Load Balancer 之 `aws:elb:listener` 中的設定。
- `aws:elasticbeanstalk:environment:process` - 設定運作狀態檢查，並針對執行於環境執行個體的程序，指定連接埠和通訊協定。此連接埠和通訊協定設定，對應至 Classic Load Balancer 接聽程式之 `aws:elb:listener` 中的執行個體連接埠和執行個體通訊協定設定。運作狀態檢查設定則對應至 `aws:elb:healthcheck` 和 `aws:elasticbeanstalk:application` 命名空間內的設定。

Example .ebextensions/network-load-balancer.config

若要開始使用 Network Load Balancer，請使用組態檔案，將負載平衡器類型設定為 `network`。

```
option_settings:
  aws:elasticbeanstalk:environment:
    LoadBalancerType: network
```

> **Note**
>
> 您只能在建立環境時設定負載平衡器的類型。

Example .ebextensions/nlb-default-process.config

下列組態檔案會修改預設程序的運作狀態檢查設定。

```
option_settings:
  aws:elasticbeanstalk:environment:process:default:
    DeregistrationDelay: '20'
    HealthCheckInterval: '10'
    HealthyThresholdCount: '5'
    UnhealthyThresholdCount: '5'
    Port: '80'
    Protocol: TCP
```

Example .ebextensions/nlb-secure-listener.config

下列組態檔案會針對連接埠 443 上的安全流量新增接聽程式，以及新增接聽連接埠 443 的相符目標程序。

```
option_settings:
  aws:elbv2:listener:443:
    DefaultProcess: https
    ListenerEnabled: 'true'
  aws:elasticbeanstalk:environment:process:https:
    Port: '443'
```

`DefaultProcess` 選項以此方式命名，原因是 Application Load Balancer 能夠在相同連接埠上針對送往特定路徑的流量擁有非預設接聽程式 (詳細資訊請參閱Application Load Balancer)。以 Network Load Balancer 而言，此選項會指定此接聽程式的唯一目標程序。

在此範例中，我們將程序命名為 `https`，因為它會接聽安全 (HTTPS) 流量。由於 Network Load Balancer 僅能與 TCP 搭配運作，因此接聽程式會使用 TCP 通訊協定，將流量傳送至指定連接埠上的程序。此作法沒問題，因為 HTTP 和 HTTPS 的網路流量實作於 TCP 之上。

設定存取日誌

您可以使用組態檔案設定您環境的負載平衡器，將存取日誌上傳到 Amazon S3 儲存貯體。如需相關說明，請參閱 GitHub 上的下列範例組態檔案：

- loadbalancer-accesslogs-existingbucket.config - 設定負載平衡器將存取日誌上傳到現有的 Amazon S3 儲存貯體。
- loadbalancer-accesslogs-newbucket.config - 設定負載平衡器將存取日誌上傳到新的儲存貯體。

將資料庫新增至您的 Elastic Beanstalk 環境

Elastic Beanstalk 提供與 Amazon Relational Database Service (Amazon RDS) 的整合。您可以使用 Elastic Beanstalk 在現有環境中新增 MySQL、PostgreSQL、Oracle 或 SQL Server 資料庫，或在您建立時新增新的資料庫。在您新增資料庫執行個體時，Elastic Beanstalk 會為您的應用程式提供連線資訊。系統會藉由設定資料庫主機名稱、連接埠、使用者名稱、密碼和資料庫名稱的環境屬性來執行此動作。

若您未曾搭配應用程式使用資料庫執行個體，建議您先使用本主題所述的程序，以使用 Elastic Beanstalk 服務將資料庫新增至測試環境。如此一來，您便可確認應用程式即使不需要 Elastic Beanstalk 外部資料庫的其他組態工作，也能讀取環境屬性、建構連線字串，以及連線至資料庫執行個體。

在您確認應用程式可與資料庫搭配正常運作後，您可以考慮移向生產環境。此時，您可以選擇將資料庫從 Elastic Beanstalk 環境解耦，以便移向可提供更多靈活性的組態。已解耦的資料庫仍能作為外部 Amazon RDS 資料庫執行個體運作。解耦資料庫不會影響環境的正常運作狀態。如果您需要終止環境便可如此操作，也可選擇讓資料庫在 Elastic Beanstalk 之外維持可用且持續運作。

使用外部資料庫有幾個優點。您可以從多個環境連線至外部資料庫環境、使用整合資料庫不支援的資料庫類型，以及執行藍/綠部署。您也可以在 Elastic Beanstalk 環境之外建立資料庫執行個體，以作為使用 Elastic Beanstalk 所建立解耦資料庫的替代方案。兩個選項都會造成位於您 Elastic Beanstalk 環境外部的資料庫執行個體，且需要額外的安全群組和連線字串組態。如需更多詳細資訊，請參閱 搭配 Amazon RDS 使用 Elastic Beanstalk。

章節

- 資料庫生命週期
- 使用主控台將 Amazon RDS 資料庫執行個體新增至您的環境

- 連線到資料庫
- 利用主控台設定整合的 RDS 資料庫執行個體
- 利用組態檔案設定整合的 RDS 資料庫執行個體
- 使用主控台解耦 RDS 資料庫執行個體
- 使用主控台檔案解耦 RDS 資料庫執行個體

資料庫生命週期

將資料庫從 Elastic Beanstalk 環境中解耦後，您可以選擇要對資料庫執行的動作。您可以從中進行選擇的選項統稱為刪除政策。將資料庫從 Elastic Beanstalk 環境中解耦或終止 Elastic Beanstalk 環境後，以下刪除政策便會套用至資料庫。

- Snapshot (快照) - 在 Elastic Beanstalk 終止資料庫之前，其會儲存資料庫的快照。在您將資料庫執行個體新增至 Elastic Beanstalk 環境或建立獨立資料庫時，可以從快照還原資料庫。如需有關透過快照建立新獨立資料庫執行個體的詳細資訊，請參閱《Amazon RDS 使用者指南》中的從資料庫快照還原。存放資料庫快照可能需要支付費用。如需詳細資訊，請參閱 Amazon RDS 定價中的備份儲存一節。
- Delete (刪除) - Elastic Beanstalk 會終止資料庫。在其終止後，資料庫執行個體就不可再用於任何操作。
- Retain (保留) - 資料庫執行個體未終止。雖然與 Elastic Beanstalk 解耦，但其仍可以使用和運作。然後，您可以設定一個或多個環境，以作為外部 Amazon RDS 資料庫執行個體，藉此連線至資料庫。如需更多詳細資訊，請參閱 搭配 Amazon RDS 使用 Elastic Beanstalk。

使用主控台將 Amazon RDS 資料庫執行個體新增至您的環境

您可以藉由使用 Elastic Beanstalk 主控台，將資料庫執行個體新增到您的環境。

欲將資料庫執行個體新增到您的環境

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇 Configuration (組態)。
4. 在 Database (資料庫) 組態類別中，選擇 Edit (編輯)。
5. 選擇資料庫引擎，並輸入使用者名稱和密碼。
6. 若要儲存變更，請選擇頁面底部的儲存變更。

您可以設定下列選項：

- Snapshot (快照) – 選擇現有的資料庫快照。Elastic Beanstalk 還原快照，並將其新增到您的環境。預設值為 None (無)。當值是 None (無) 時，您可使用此頁面上的其他設定來設定新的資料庫。
- Engine (引擎) – 選擇資料庫引擎。
- Engine version (引擎版本) – 選擇特定的資料庫引擎版本。
- Instance class (執行個體類別) – 選擇資料庫執行個體類別。如需有關資料庫執行個體類別的詳細資訊，請參閱 https://aws.amazon.com/rds/。
- Storage (儲存) – 選擇要為您的資料庫所佈建的儲存容量。您可以在日後增加配置的儲存容量，但不能縮減。如需關於儲存容量分配的資訊，請參閱功能相關文章。
- Username (使用者名稱) - 僅使用包含數字和字母的組合來輸入所選擇的使用者名稱。
- Password (密碼) – 輸入包含 8–16 個可列印的 ASCII 字元 (不含 /、\ 和 @) 選用密碼。
- Availability (可用性) – 選擇 High (Multi-AZ) (高 (異地同步備份))，在第二個可用區域中執行暖備份，以維持高可用性。
- Database deletion policy (資料庫刪除政策) - 刪除政策會決定資料庫在從您的環境解耦後的情況。系統可以將其設定為下列其中一個值：`Create Snapshot`、`Retain` 或 `Delete`。這些值會在此相同主題的 資料庫生命週期 中加以說明。

> **Note**
>
> Elastic Beanstalk 使用您提供的使用者名稱和密碼建立資料庫的主要使用者。若要進一步了解主要使用者及其權限，請參閱主要使用者帳戶權限。

新增資料庫執行個體約需要 10 分鐘。在更新完成時，系統會將新資料庫耦合至您的環境。資料庫執行個體的主機名稱和其他連線資訊，會透過下列環境屬性提供給您的應用程式：

屬性名稱	描述	屬性值
RDS_HOSTNAME	資料庫執行個體的主機名稱。	在 Amazon RDS 主控台：端點的連線能力和安全性索引標籤上。
RDS_PORT	資料庫執行個體接受連線的連接埠。預設值在不同資料庫引擎中有所差異。	在 Amazon RDS 主控台：連接埠的連線能力和安全性索引標籤上。
RDS_DB_NAME	資料庫名稱，**ebdb**。	在 Amazon RDS 主控台：資料庫名稱的組態索引標籤上。
RDS_USERNAME	您為資料庫設定的使用者名稱。	在 Amazon RDS 主控台：主要使用者名稱的組態索引標籤上。
RDS_PASSWORD	您為資料庫設定的密碼。	無法在 Amazon RDS 主控台中提供參考。

連線到資料庫

使用連線資訊，從您的應用程式中利用環境變數來連線到您的資料庫。關於搭配您的應用程式使用 Amazon RDS，詳細資訊請參閱下列主題。

- Java SE – 連線至資料庫 (Java SE 平台)
- Java 搭配 Tomcat – 連線至資料庫 (Tomcat 平台)
- Node.js – 連線至資料庫
- .NET – 連線至資料庫
- PHP – 使用 PDO 或 MySQLi 連接至資料庫
- Python – 連線至資料庫
- Ruby – 連線至資料庫

利用主控台設定整合的 RDS 資料庫執行個體

您可以在 Elastic Beanstalk 主控台中，從環境 Configuration (組態) 頁面的 Database (資料庫) 區段，來檢視和修改您資料庫執行個體的組態設定。

在 Elastic Beanstalk 主控台中設定環境的資料庫執行個體

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇 Configuration (組態)。
4. 在 Database (資料庫) 組態類別中，選擇 Edit (編輯)。

在建立資料庫之後，您可以修改 Instance class(執行個體類別)、Storage (儲存)、Password (密碼)、Availability (可用性) 和 Database deletion policy (資料庫刪除政策) 設定。如果您變更執行個體類別，Elastic Beanstalk 會重新佈建資料庫執行個體。

如果您不再需要 Elastic Beanstalk 為資料庫與環境建立關聯，可以選擇將資料庫解耦，方法是選取 Decouple database (解耦資料庫)。請務必了解此操作所涉及的選項和考量。如需更多詳細資訊，請參閱 the section called “使用主控台解耦 RDS 資料庫執行個體”。

> **警告**
>
> 請勿修改耦合的資料庫執行個體上 Elastic Beanstalk 所提供功能以外的設定 (例如，在 Amazon RDS 主控台中)。如果這麼做，您的 Amazon RDS 資料庫組態可能與您環境的定義不同步。當您更新或重新啟動您的環境，環境中指定的設定會覆寫在 Elastic Beanstalk 外的任何設定。
>
> 如果您需要修改 Elastic Beanstalk 未直接支援的設定，請使用 Elastic Beanstalk 組態檔案。

利用組態檔案設定整合的 RDS 資料庫執行個體

您可以使用組態檔案設定您環境的資料庫執行個體。使用`aws:rds:dbinstance`命名空間中的選項。以下範例修改配置的資料庫儲存大小為 100 GB。

Example .ebextensions/db-instance-options.config

```
option_settings:
  aws:rds:dbinstance:
    DBAllocatedStorage: 100
```

如果您要設定 Elastic Beanstalk 不支援的資料庫執行個體屬性，您仍可使用組態檔案，並使用 `resources` 金鑰指定您的設定。以下範例設定值至 `StorageType` 和 `Iops` Amazon RDS 屬性。

Example .ebextensions/db-instance-properties.config

```
Resources:
  AWSEBRDSDatabase:
    Type: AWS::RDS::DBInstance
    Properties:
      StorageType:io1
      Iops: 1000
```

使用主控台解耦 RDS 資料庫執行個體

您可以將資料庫從 Elastic Beanstalk 環境解耦，而不會影響環境的正常運作狀態。在解耦資料庫之前，請考慮下列要求：

- 將資料庫解耦後可能會發生哪些情況？

  您可以選擇建立資料庫的快照然後終止資料庫、將資料庫獨立於 Elastic Beanstalk 外持續運作，或永久刪除資料庫。Database deletion policy (資料庫刪除政策) 設定會決定此結果。如需有關刪除政策的詳細說明，請參閱此相同主題中的 資料庫生命週期。
- 在解耦之前是否需要變更任何資料庫組態設定？

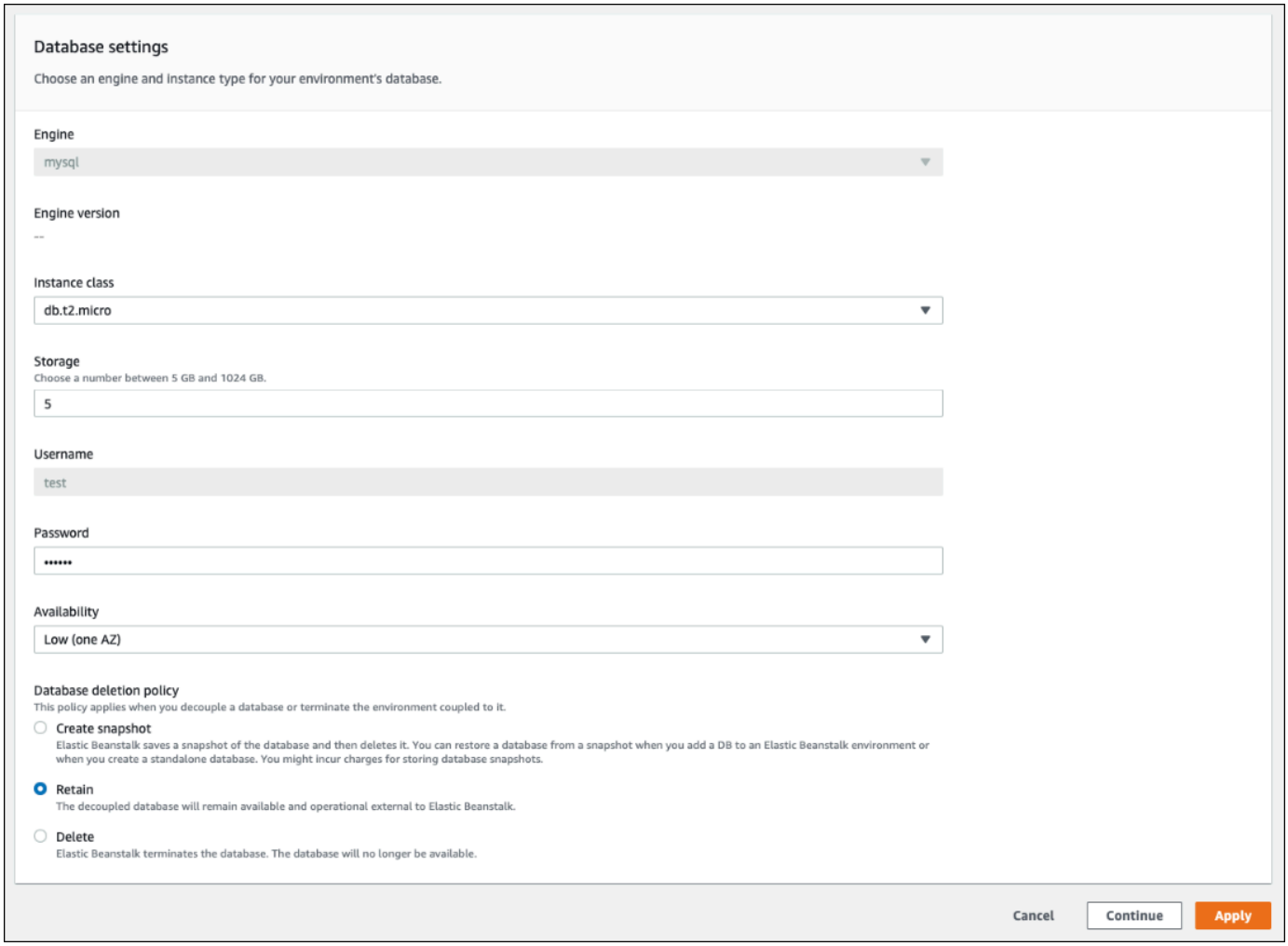
  如果您需要變更資料庫的任何組態，應在解耦資料庫前，將其套用至資料庫。這包括變更 Database deletion policy (資料庫刪除政策)。系統將忽略任何與 Decouple database (解耦資料庫) 設定同時提交的待處理變更項目，同時僅會套用解耦設定。

將資料庫執行個體從環境解耦

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇 Configuration (組態)。
4. 在 Database (資料庫) 組態類別中，選擇 Edit (編輯)。
5. 檢閱 Database settings (資料庫設定) 區段中的所有組態值，尤其是 Database deletion policy (資料庫刪除政策)，其會決定資料庫解耦後的情況。



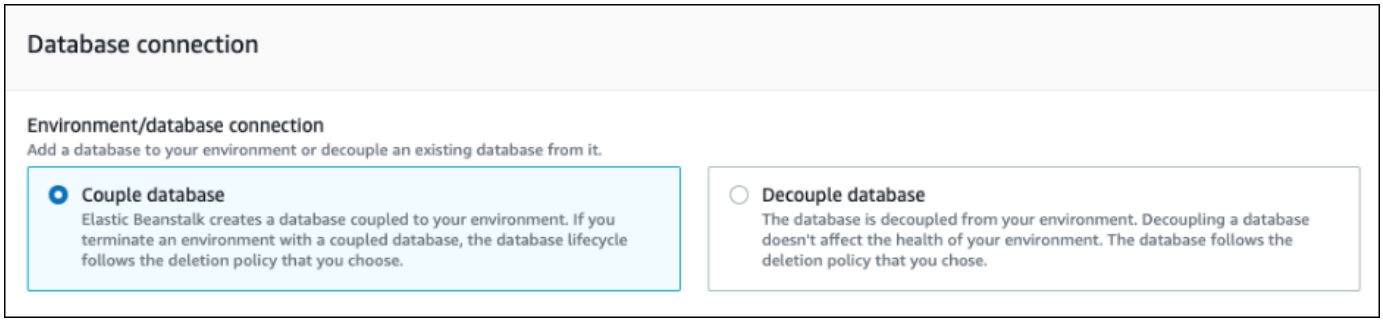
如果所有其他組態設定皆正確，請跳至步驟 6 來解耦資料庫。

> **Warning**
>
> 請務必將 Database deletion policy (資料庫刪除政策) 設定與 Decouple database (解耦資料庫) 分開套用。如果您選取 Apply (套用) 且欲同時保存 Decouple database (解耦資料庫) 和新選取的 Database deletion policy (資料庫刪除政策)，則系統會忽略您選擇的新刪除政策。Elastic Beanstalk 會依照先前設定的刪除政策來解耦資料庫。如果先前設定的刪除政策為 `Delete` 或 `Create Snapshot`，您可能會有遺失資料庫的風險，而非遵循預期的待處理政策。

如果有任何組態設定需要更新，請執行下列動作：

1. 在 Database settings (資料庫設定) 面板中進行必要的修改。
2. 選擇 Apply (套用)。需要幾分鐘的時間來儲存資料庫的組態變更。
3. 回到步驟 3，然後透過導覽窗格選擇 Configuration (組態)。

6. 前往窗格上的 Database connection (資料庫連線) 區段。



7. 選擇 Decouple database (解耦資料庫)。
8. 選擇 Apply (套用) 來啟動資料庫解耦操作。

刪除政策設定會決定資料庫的結果，以及解耦資料庫所需的時間長度。

- 如果刪除政策設定為 `Delete`，則系統會刪除資料庫。作業時間可能需要大約 10-20 分鐘，視資料庫的大小而定。
- 如果刪除政策設定為 `Snapshot`，系統會建立資料庫的快照。之後，系統會刪除資料庫。此處理程序所需的時間長度依資料庫的大小而定。

- 如果刪除政策設定為 Retain，則資料庫會在 Elastic Beanstalk 環境外部保持運作。解耦資料庫通常需要不到五分鐘的時間。

如果您決定將資料庫保留在 Elastic Beanstalk 環境之外，則需要採取其他步驟來加以設定。如需更多詳細資訊，請參閱 搭配 Amazon RDS 使用 Elastic Beanstalk。如果您計劃為生產環境使用解耦的資料庫，請確認該資料庫使用的儲存類型是否適合您的工作負載。如需詳細資訊，請參閱《Amazon RDS 使用者指南》中的資料庫執行個體儲存體和修改資料庫執行個體。

使用主控台檔案解耦 RDS 資料庫執行個體

您可以將資料庫執行個體從 Elastic Beanstalk 環境解耦，而不會影響環境的正常運作狀態。資料庫執行個體會遵循在資料庫受到解耦時套用的資料庫刪除政策。

解耦資料庫所需的兩個選項都位於 the section called “aws:rds:dbinstance” 命名空間中。如下所示：

- DBDeletionPolicy 選項會設定刪除政策。系統可以將其設定為下列其中一個值：Snapshot、Delete 或 Retain。這些值會在此相同主題的 資料庫生命週期 中加以說明。
- HasCoupledDatabase 選項會決定您的環境是否具有耦合的資料庫。
  - 如果切換為 true，Elastic Beanstalk 會建立與您環境耦合的新資料庫執行個體。
  - 如果切換為 false，Elastic Beanstalk 會開始從您的環境中解耦資料庫執行個體。

如果您想要在解耦資料庫之前變更資料庫組態，請先在個別操作中套用任何組態變更。這包括變更 DBDeletionPolicy 組態。套用變更後，請執行個別命令以設定解耦選項。如果您同時提交其他組態設定和解耦設定，則系統會在套用解耦設定時會忽略其他組態選項設定。

> **Warning**
>
> 請務必以兩個獨立操作執行命令，來套用 DBDeletionPolicy 和 HasCoupledDatabase 設定。如果啟用中的刪除政策已設定為 Delete 或 Snapshot，您可能會有遺失資料庫的風險。資料庫會遵循目前啟用中的刪除政策，而非您指示的待處理刪除政策。

將資料庫執行個體從環境解耦

依照下列步驟將資料庫從您的 Elastic Beanstalk 環境解耦。您可以使用 EB CLI 或 AWS CLI 來完成這些步驟。如需詳細資訊，請參閱使用組態檔案來進行進階的環境自訂。

1. 如果您要變更刪除政策，請以下列格式來設定組態檔案。在此範例中，刪除政策會設為保留。

Example

```
option_settings:
  aws:rds:dbinstance:
    DBDeletionPolicy: Retain
```

2. 使用您偏好的工具來執行命令，以完成組態更新。
3. 設定組態檔案，以將 HasCoupledDatabase 設定為 false。

Example

```
option_settings:
  aws:rds:dbinstance:
    HasCoupledDatabase: false
```

4. 使用您偏好的工具來執行命令，以完成組態更新。

刪除政策設定會決定資料庫的結果，以及解耦資料庫所需的時間長度。

- 如果刪除政策設定為 Delete，則系統會刪除資料庫。作業時間可能需要大約 10-20 分鐘，視資料庫的大小而定。
- 如果刪除政策設定為 Snapshot，系統會建立資料庫的快照。之後，系統會刪除資料庫。此處理程序所需的時間長度依資料庫的大小而定。
- 如果刪除政策設定為 Retain，則資料庫會在 Elastic Beanstalk 環境外部保持運作。解耦資料庫通常需要不到五分鐘的時間。

如果您決定將資料庫保留在 Elastic Beanstalk 環境之外，則需要採取其他步驟來加以設定。如需更多詳細資訊，請參閱 搭配 Amazon RDS 使用 Elastic Beanstalk。如果您計劃為生產環境使用解耦的資料庫，請確認該資料庫使用的儲存類型是否適合您的工作負載。如需詳細資訊，請參閱《Amazon RDS 使用者指南》中的資料庫執行個體儲存體和修改資料庫執行個體。

您的 AWS Elastic Beanstalk 環境安全

Elastic Beanstalk 提供多種選項，以控制您環境的服務存取 (安全性) 和其中的 Amazon EC2 執行個體。此主題討論設定這些選項的組態。

章節

- 設定您的環境安全性
- 環境安全組態命名空間

設定您的環境安全性

您可以在 Elastic Beanstalk 主控台中修改 Elastic Beanstalk 環境安全組態。

在 Elastic Beanstalk 主控台中設定環境服務存取 (安全性)

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

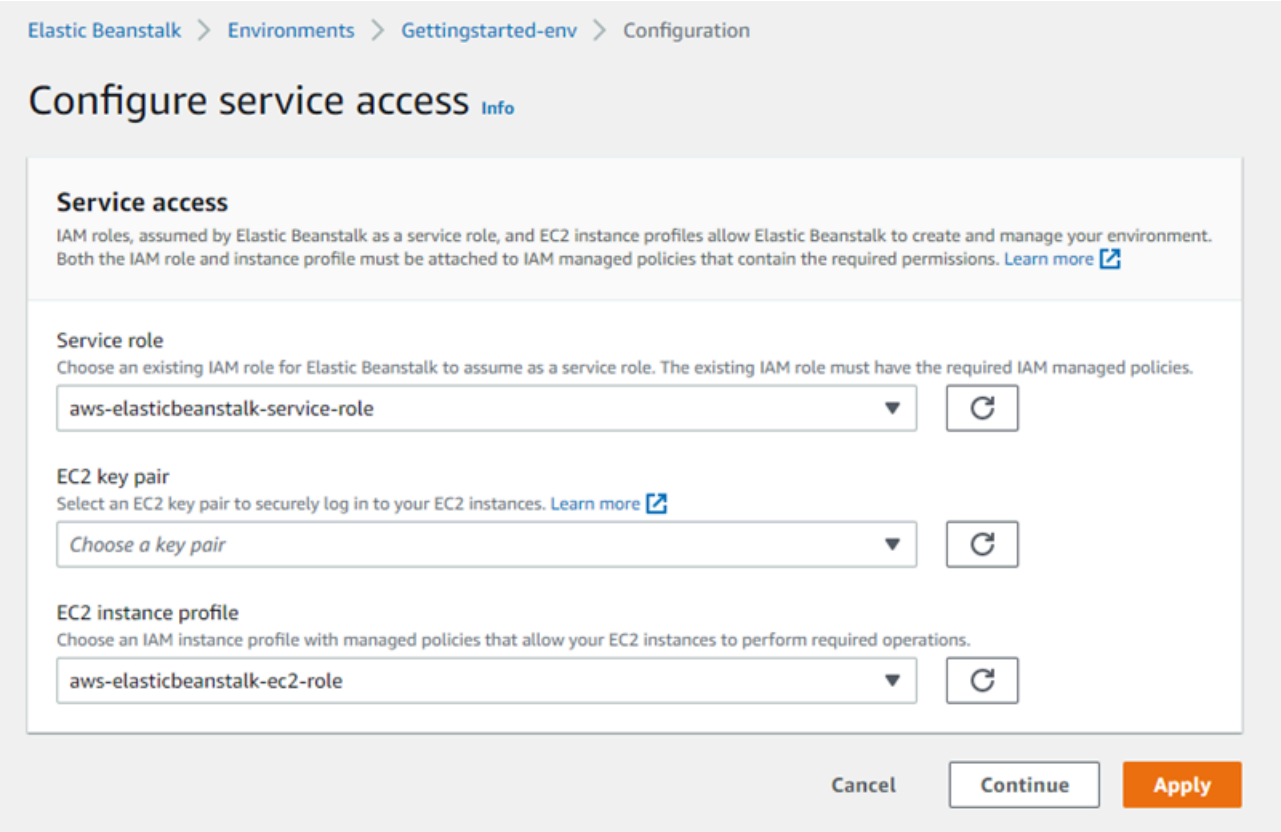
> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇組態。
4. 在 Service access (服務存取) 組態類別中，選擇 Edit (編輯)。

可用的設定如下所示。

設定

- 服務角色
- EC2 key pair (EC2 金鑰對)
- IAM 執行個體描述檔



服務角色

選取服務角色，與您 Elastic Beanstalk 環境建立關聯性。當 Elastic Beanstalk 代表您存取其他服務時，會擔任該 AWS 服務角色。如需詳細資訊，請參閱 管理 Elastic Beanstalk 服務角色。

EC2 key pair (EC2 金鑰對)

透過 Amazon EC2 金鑰對，您可安全登入為 Elastic Beanstalk 應用程式佈建的 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體。如需建立 key pair 的指示，請參閱 Amazon EC2 使用者指南中的使用 Amazon EC2 建立金鑰配對。

> **Note**
>
> 當您建立金鑰對，Amazon EC2 會存放公開金鑰副本。如果您不再需要使用它連接任何環境執行個體，您可以從 Amazon EC2 將其刪除。如需詳細資訊，請參閱 Amazon EC2 使用者指南中的刪除金鑰配對。

從下拉式選單選擇 EC2 key pair (EC2 金鑰對)，將其指派給您環境的執行個體。指派金鑰對時，公有金鑰會存放於執行個體，以驗證存放於本機的私有金鑰。私密金鑰永遠不會儲存在 AWS。

如需有關連線到 Amazon EC2 執行個體的詳細資訊，請參閱 Amazon EC2 使用者指南中的使用 PuTTY 從 Windows Connect 到執行個體以及從 Windows 連線到 Linux/Unix 執行個體。

IAM 執行個體描述檔

EC2 執行個體設定檔是您 Elastic Beanstalk 環境啟動的執行個體所套用的 IAM 角色。Amazon EC2 執行個體會假設執行個體設定檔角色，以簽署 API 的請求 AWS 並存取，例如將日誌上傳到 Amazon S3。

當您首次於 Elastic Beanstalk 主控台建立環境時，Elastic Beanstalk 會提示您建立具有預設之一組許可的執行個體描述檔。您可以新增此設定檔的權限，讓您的執行個體存取其他 AWS 服務。如需詳細資訊，請參閱 管理 Elastic Beanstalk 執行個體描述檔。

> **Note**
>
> 之前，Elastic Beanstalk 建立了一個預設的 EC2 執行個體設定檔，名為`aws-elasticbeanstalk-ec2-role` AWS 帳戶第一次建立環境時。此執行個體設定檔包含預設受管政策。如果您的帳戶已經擁有此執行個體設定檔，您仍然可以將其指派給您的環境。
> 不過，最近的 AWS 安全性準則不允許 AWS 服務自動建立具有信任政策的角色給其他 AWS 服務 (在這種情況下為 EC2)。基於這些安全性準則，Elastic Beanstalk 不再建立預設 `aws-elasticbeanstalk-ec2-role` 執行個體設定檔。

環境安全組態命名空間

以下命名空間中 Elastic Beanstalk 提供組態選項，讓您能夠自訂環境的安全性：

- `aws:elasticbeanstalk:environment` – 使用 `ServiceRole` 選項設定環境的服務角色。
- `aws:autoscaling:launchconfiguration` – 使用 `EC2KeyName` 和 `IamInstanceProfile` 選項設定環境的 Amazon EC2 執行個體許可。

EB CLI 和 Elastic Beanstalk 主控台會為前述選項套用建議的數值。若您想要使用組態檔進行相同的設定，您必須移除這些設定。如需詳細資訊，請參閱「建議值」。

在您的 Elastic Beanstalk 環境中標記資源

您可以將標籤套用至您的 AWS Elastic Beanstalk 環境。標籤是與 AWS 資源相關聯的索引鍵值配對。如需 Elastic Beanstalk 資源標記、使用案例、標籤索引鍵和值限制條件的相關資訊，以及支援的資源類型，請參閱標記 Elastic Beanstalk 應用程式資源。

Elastic Beanstalk 將環境標籤應用於環境資源本身，以及 Elastic Beanstalk 為環境創建的其他 AWS 資源。您也可以使用標記來管理環境中特定資源層級的許可。如需詳細資訊，請參閱 Amazon EC2 使用者指南中的標記您的 Amazon EC2 資源。

依預設，Elastic Beanstalk 會將幾個標籤套用至您的環境：

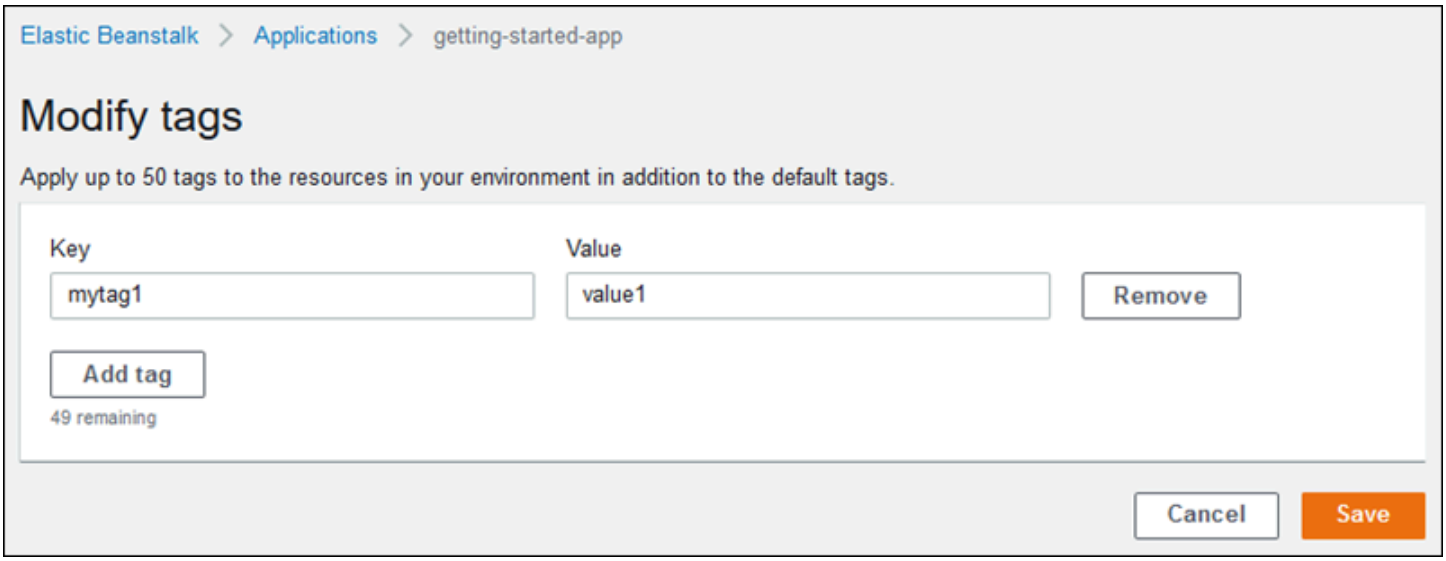
- `elasticbeanstalk:environment-name` - 環境名稱。
- `elasticbeanstalk:environment-id` - 環境 ID。
- `Name` – 亦為環境名稱。在 Amazon EC2 儀表板上，`Name` 用於辨識並排序資源。

您無法編輯這些預設標籤。

您可以在建立 Elastic Beanstalk 環境時指定標籤。在現有環境中，您可以新增或移除標籤，並更新現有標籤的值。一個環境最多可以有 50 個標籤，包括預設標籤。

在環境建立期間新增標籤

使用 Elastic Beanstalk 主控台來建立環境時，您可於 Create New Environment wizard (建立新環境精靈) 的 Modify tags (修改標籤) 頁面，以指定標籤索引鍵和值。



若您使用 EB CLI 來建立環境，請使用帶有 eb create 的 --tags 選項來新增標籤。

```
~/workspace/my-app$ eb create --tags mytag1=value1,mytag2=value2
```

對於 AWS CLI 或其他以 API 為基礎的用戶端，請使用命令上的--tags參數。create-environment

```
$ aws elasticbeanstalk create-environment \
      --tags Key=mytag1,Value=value1 Key=mytag2,Value=value2 \
      --application-name my-app --environment-name my-env --cname-prefix my-app --
version-label v1 --template-name my-saved-config
```

已儲存組態包含使用者定義的標記。當您於環境建立期間套用內含標籤的已儲存組態，只要您尚未指定任何新標籤，這些標籤將套用至新環境。若您使用上述方法將標籤新增至環境，將會捨棄已儲存組態內定義的任何標籤。

管理現有環境的標籤

您可以在現有的 Elastic Beanstalk 環境中新增、更新和刪除標籤。Elastic Beanstalk 會將變更套用至您環境的資源。

但是，您無法編輯 Elastic Beanstalk 套用至您環境的預設標籤。

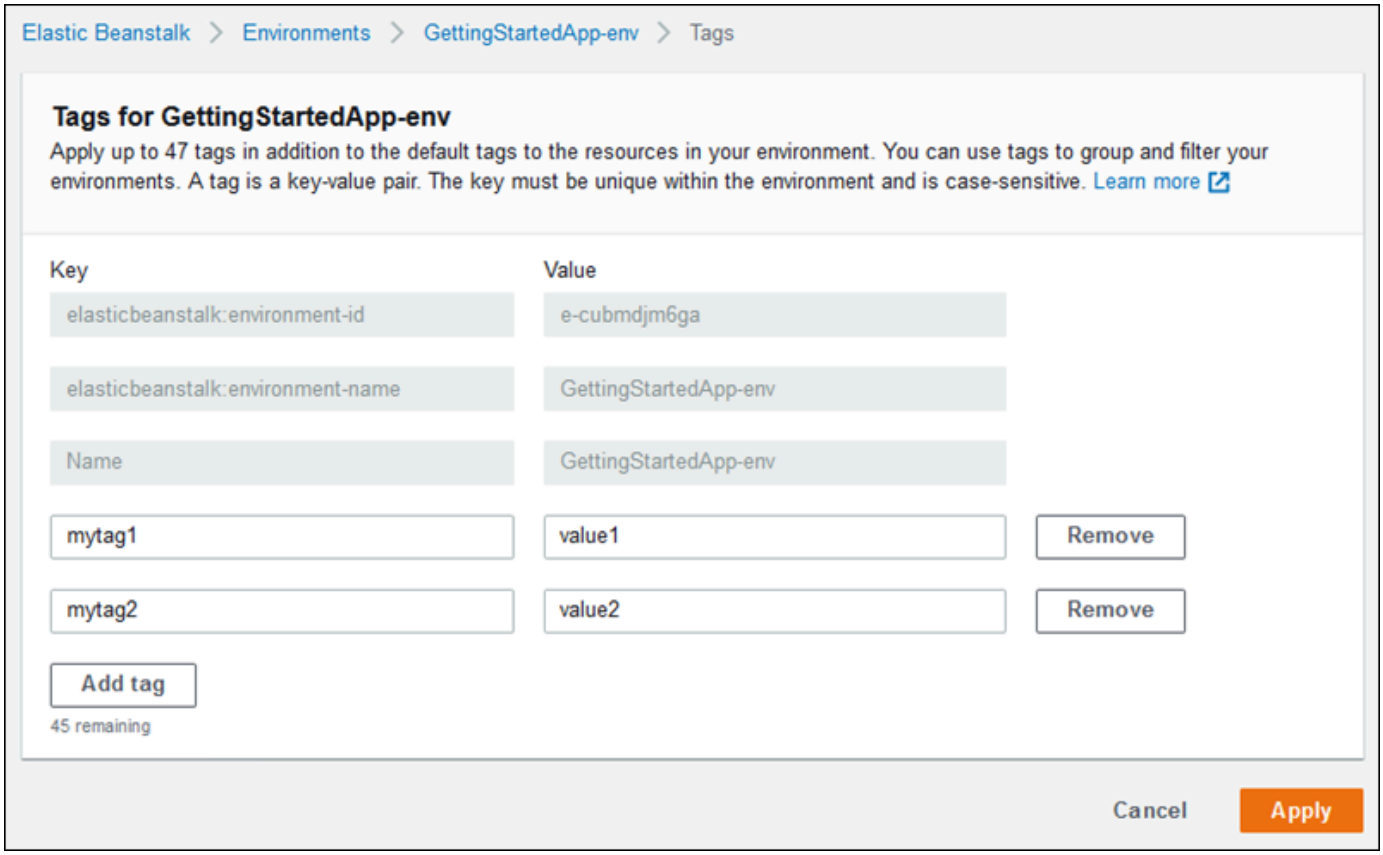
在 Elastic Beanstalk 主控台中管理環境的標籤

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

   > **Note**
   >
   > 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇 Tags (標籤)。

   標籤管理頁面會顯示環境中現有的標籤清單。



4. 新增、更新或刪除標籤：

   - 若要新增標籤，請於清單底部的空白方塊中輸入標籤。若要新增其他標籤，請選擇 Add tag (新增標籤)，然後 Elastic Beanstalk 會新增另一對空白方塊。
   - 欲更新標籤的金鑰或值，請於標籤列各自的方塊進行編輯。
   - 若要刪除標籤，請選擇標籤值方塊旁邊的 Remove (移除)。

5. 若要儲存變更，請選擇頁面底部的儲存變更。

若您透過 EB CLI 更新環境，請使用 eb tags 來新增、更新、刪除或列出標籤。

例如，下列命令會列出您預設環境的標籤。

```
~/workspace/my-app$ eb tags --list
```

下列命令會更新標籤 `mytag1` 並刪除標籤 `mytag2`。

```
~/workspace/my-app$ eb tags --update mytag1=newvalue --delete mytag2
```

如需完整選項清單和更多範例，請參閱 eb tags。

對於 AWS CLI 或其他以 API 為基礎的用戶端，請使用list-tags-for-resource指令列出環境的標籤。

```
$ aws elasticbeanstalk list-tags-for-resource --resource-arn
 "arn:aws:elasticbeanstalk:us-east-2:my-account-id:environment/my-app/my-env"
```

使用 update-tags-for-resource 命令來新增、更新或刪除環境內的標籤。

```
$ aws elasticbeanstalk update-tags-for-resource \
      --tags-to-add Key=mytag1,Value=newvalue --tags-to-remove mytag2 \
      --resource-arn "arn:aws:elasticbeanstalk:us-east-2:my-account-id:environment/my-
app/my-env"
```

在 `--tags-to-add` 的 update-tags-for-resource 參數中，同時指定欲新增和欲更新的標籤。如此將新增不存在的標籤，並更新現有標籤的值。

> **Note**
>
> 要在 Elastic Beanstalk 環境中使用這兩個 AWS CLI 命令，您需要環境的 ARN。您可使用下列命令來擷取 ARN。
>
> ```
> $ aws elasticbeanstalk describe-environments
> ```

環境屬性與其他軟體設定

設定更新、監控和日誌組態頁面可讓您在執行應用程式的 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體上設定軟體。您可以設定環境屬性、AWS X-Ray 除錯、執行個體日誌儲存和串流，以及平台特定的設定。

主題

- 設定平台特定設定
- 設定環境屬性 (環境變數)
- 軟體設定命名空間
- 存取環境屬性
- 設定 AWS X-Ray 除錯
- 檢視您的 Elastic Beanstalk 環境日誌

設定平台特定設定

除了適用於所有環境的一組標準選項外，大部分的 Elastic Beanstalk 平台也可讓您指定特定語言或特定架構的設定。它們會顯示在設定更新、監控和日誌頁面的平台軟體區段中，而且可採取下列形式。

- 預設的環境屬性 – Ruby 平台會針對架構設定使用環境屬性，例如 `RACK_ENV` 和 `BUNDLE_WITHOUT`。
- 預留位置環境屬性 – Tomcat 平台會定義名為 `JDBC_CONNECTION_STRING` 的環境屬性，此屬性未設定為任何值。這類設定較常見於舊的平台版本。
- 組態選項 – 大多數的平台會以特定平台或共用的命名空間 (例如 `aws:elasticbeanstalk:xray` 或 `aws:elasticbeanstalk:container:python`) 定義組態選項。

在 Elastic Beanstalk 主控台中設定平台特定設定

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

   > **Note**
   >
   > 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇 Configuration (組態)。
4. 在更新、監控和日誌記錄組態類別中，選擇編輯。
5. 在平台軟體下，進行必要的選項設定變更。
6. 若要儲存變更，請選擇頁面底部的儲存變更。

如需有關平台特定的選項，以及在您的程式碼中取得環境屬性值的詳細資訊，請參閱您的語言或架構的平台主題：

- Docker – the section called “環境資訊”
- Go – 使用 Elastic Beanstalk Go 平台
- Java SE – 使用 Elastic Beanstalk Java SE 平台
- Tomcat – 使用 Elastic Beanstalk Tomcat 平台
- Linux 上的 .NET Core – 使用 Linux 上的 .NET Core 平台
- .NET – 使用 Elastic Beanstalk .NET 平台

- Node.js – 使用 Elastic Beanstalk Node.js 平台
- PHP – 使用 Elastic Beanstalk PHP 平台
- Python – 使用 Elastic Beanstalk Python 平台
- Ruby – 使用 Elastic Beanstalk Ruby 平台

設定環境屬性 (環境變數)

可使用環境屬性 (亦稱為環境變數)，將秘密、端點、除錯設定和其他資訊傳遞到您的應用程式。環境屬性可協助您針對不同用途，在多個環境中執行應用程式，例如開發、測試、整備與生產。

另外，當您新增資料庫到您的環境時，Elastic Beanstalk 會設定像是 RDS_HOSTNAME 的環境屬性，您可在應用程式的程式碼中讀取這些屬性，來建構與物件或字串的連線。

> **環境變數**
>
> 在大多數的情況中，環境屬性會以環境變數的形式傳遞到您的應用程式，但行為則取決於平台。例如，Java SE 平台會設定您用 `System.getenv` 所擷取的環境變數，Tomcat 平台則會設定您用 `System.getProperty` 擷取的 Java 系統屬性。一般而言，若您連接至執行個體並執行 `env`，將不會顯示屬性。

在 Elastic Beanstalk 主控台中設定環境屬性

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

   > **Note**
   >
   > 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇 Configuration (組態)。
4. 在更新、監控和日誌記錄組態類別中，選擇編輯。
5. 向下捲動至環境屬性。
6. 選取新增環境屬性。
7. 輸入屬性名稱和值對。
8. 如果需要新增更多變數，請重複步驟 6 和步驟 7。

9. 若要儲存變更，請選擇頁面底部的儲存變更。

環境屬性限制

- 金鑰可以包含任何英數字元和下列符號：`_ . : / + \ - @`

  列出的這些符號對於環境屬性金鑰是有效的，但是對於您環境平台上的環境變數名稱，可能會是無效的。為了能夠相容於所有的平台，請將環境屬性限定為下列的模式：`[A-Z_][A-Z0-9_]*`
- 值可以包含任何英數字元、空格和下列符號：`_ . : / = + \ - @ ' "`

> **Note**
>
> 環境屬性值中的某些字元必須逸出。使用反斜線字元 (\) 來代表一些特殊字元和控制字元。下列清單包含表示某些需要逸出的字元的範例：
>
> - 反斜線 (\) — 代表使用 \\
> - 單引號 (') — 代表使用 \'
> - 雙引號 (") — 代表使用 \"

- 金鑰和值會區分大小寫。
- 在做為字串儲存時 (格式為 *key*=*value*)，所有環境屬性加起來的大小不能超過 4,096 個位元組。

軟體設定命名空間

您可以使用組態檔案來設定組態選項，並在部署期間執行其他的執行個體設定工作。組態選項可由 Elastic Beanstalk 服務或您使用的平台來定義，並且會組織成「命名空間」。

您可以使用 Elastic Beanstalk 組態檔案，在您的來源碼中設定環境屬性和組態選項。使用 `aws:elasticbeanstalk:application:environment` 命名空間來定義環境屬性。

Example .ebextensions/options.config

```
option_settings:
  aws:elasticbeanstalk:application:environment:
    API_ENDPOINT: www.example.com/api
```

如果您是使用組態檔案或 AWS CloudFormation 範本來建立自訂資源，則可以使用 AWS CloudFormation 函數來取得關於資源的資訊，並且在部署期間，動態地將資源指派給環境屬性。下列

是 elastic-beanstalk-samples GitHub 儲存庫中的範例，此範例使用了 Ref 函數，取得其建立 Amazon SNS 主題的 ARN，並將此 ARN 指派給名為 NOTIFICATION_TOPIC 的環境屬性。

> **備註**
>
> - 如果您使用 AWS CloudFormation 函數來定義環境屬性，Elastic Beanstalk 主控台會先顯示屬性的值，再評估函數。您可以使用 get-config 平台指令碼來確認可供您應用程式使用的環境屬性值。
> - 多容器 Docker 平台不使用 AWS CloudFormation 建立容器資源。因此，此平台不支援使用 AWS CloudFormation 函數定義環境屬性。

Example .Ebextensions/sns-topic.config

```
Resources:
  NotificationTopic:
    Type: AWS::SNS::Topic

option_settings:
  aws:elasticbeanstalk:application:environment:
    NOTIFICATION_TOPIC: ``{"Ref" : "NotificationTopic"}``
```

您也可以使用此功能來散佈 AWS CloudFormation 虛擬參數提供的資訊。此範例會取得目前區域的資訊，並將其指派給名為 AWS_REGION 的屬性。

Example .Ebextensions/env-regionname.config

```
option_settings:
  aws:elasticbeanstalk:application:environment:
    AWS_REGION: ``{"Ref" : "AWS::Region"}``
```

大多數的 Elastic Beanstalk 平台會透過用來設定執行個體上所執行軟體的選項 (例如將請求轉送給應用程式的反向代理)，來定義額外的命名空間。如需關於您的平台上可用命名空間的詳細資訊，請參閱以下項目：

- Go – Go 組態命名空間
- Java SE – Java SE 組態命名空間
- Tomcat – Tomcat 組態命名空間

- Linux 上的 .NET Core – Linux 上的 .NET Core 組態命名空間
- .NET – aws:elasticbeanstalk:container:dotnet:apppool 命名空間
- Node.js – Node.js 組態命名空間
- PHP – aws:elasticbeanstalk:container:php:phpini 命名空間
- Python – Python 組態命名空間
- Ruby – Ruby 組態命名空間

Elastic Beanstalk 可提供許多組態選項讓您自訂環境。除了組態檔案，您也可以使用主控台、已儲存組態、EB CLI 或 AWS CLI 來設定組態選項。如需詳細資訊，請參閱「組態選項」。

存取環境屬性

在大部分的情況中，您會在應用程式的程式碼中存取環境屬性 (如同存取環境變數)。不過，一般而言，環境屬性只會傳遞到應用程式，即使連線到您環境中的執行個體並執行 env，也無法檢視。

- Go – `os.Getenv`

```
endpoint := os.Getenv("API_ENDPOINT")
```

- Java SE – `System.getenv`

```
String endpoint = System.getenv("API_ENDPOINT");
```

- Tomcat – `System.getProperty`

```
String endpoint = System.getProperty("API_ENDPOINT");
```

- Linux 上的 .NET Core – `Environment.GetEnvironmentVariable`

```
string endpoint = Environment.GetEnvironmentVariable("API_ENDPOINT");
```

- .NET – `appConfig`

```
NameValueCollection appConfig = ConfigurationManager.AppSettings;
string endpoint = appConfig["API_ENDPOINT"];
```

- Node.js – `process.env`

```
var endpoint = process.env.API_ENDPOINT
```

- PHP – `$_SERVER`

```
$endpoint = $_SERVER['API_ENDPOINT'];
```

- Python – `os.environ`

```
import os
endpoint = os.environ['API_ENDPOINT']
```

- Ruby – ENV

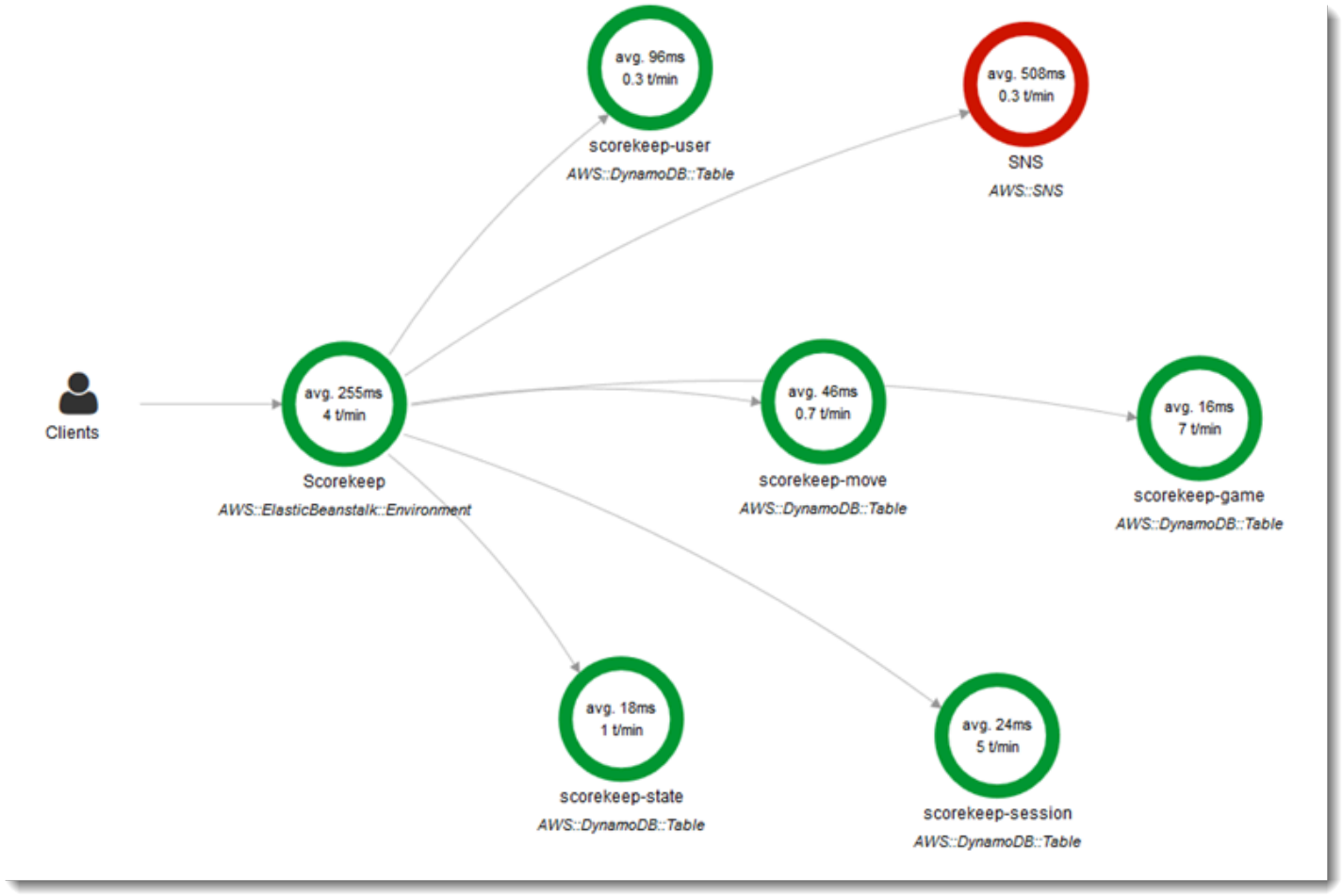
```
endpoint = ENV['API_ENDPOINT']
```

在應用程式的程式碼以外 (例如在部署期間執行的指令碼)，您可以使用 `get-config` 平台指令碼來存取環境屬性。如需使用 `get-config` 的範例組態，請參閱 elastic-beanstalk-samples GitHub 儲存庫。

設定 AWS X-Ray 除錯

您可以使用 AWS Elastic Beanstalk 主控台或組態檔案，對您環境中的執行個體執行 AWS X-Ray 常駐程式。X-Ray 是一項 AWS 服務，可蒐集您應用程式提供請求的資料，並使用此資料建構服務地圖，藉以辨識應用程式的問題與最佳化的機會。

> **Note**
>
> 部分區域未提供 X-Ray。若您於這些區域的其中之一建立環境，將無法在環境中的執行個體上執行 X-Ray 協助程式。
> 如需各區域提供的 AWS 服務的資訊，請參閱區域表。



X-Ray 提供了可用於檢測應用程式程式碼的軟體開發套件，以及一個將偵錯資訊從軟體開發套件轉送至 X-Ray API 的協助程式應用程式。

支援的平台

您可以搭配下列 Elastic Beanstalk 平台使用 X-Ray 開發套件：

- Go - 2.9.1 版及更新版本
- Java 8 - 2.3.0 版和更新版本
- Java 8 搭配 Tomcat 8 - 2.4.0 版和更新版本
- Node.js - 3.2.0 版和更新版本
- Windows Server - 所有在 2016 年 12 月 18 日或其之後發行的平台版本
- Python - 2.5.0 版和更新版本

您可以在支援的平台上透過組態選項，於環境中的執行個體上執行 X-Ray 協助程式。您可於 Elastic Beanstalk 主控台或透過使用組態檔案來啟用協助程式。

為了將資料上傳至 X-Ray，X-Ray 協助程式需要在 AWSXrayWriteOnlyAccess 受管政策中具備 IAM 許可。這些許可包含在 Elastic Beanstalk 執行個體描述檔中。如果您不使用預設執行個體描述檔，請參閱《AWS X-Ray 開發人員指南》中的授予常駐程式將資料傳送至 X-Ray 的許可。

使用 X-Ray 進行偵錯需要使用 X-Ray 開發套件。如需說明和範例應用程式，請參閱《AWS X-Ray 開發人員指南》中的 AWS X-Ray 入門。

若您使用不含精靈的平台版本，您仍可在組態檔案中透過指令碼執行它。如需詳細資訊，請參閱《AWS X-Ray 開發人員指南》中的手動下載及執行 X-Ray 常駐程式 (進階)。

章節

- 設定除錯
- aws:elasticbeanstalk:xray 命名空間

設定除錯

您可以在 Elastic Beanstalk 主控台中於正在執行的環境上啟用 X-Ray 協助程式。

在 Elastic Beanstalk 主控台中啟用偵錯

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

   > **Note**
   >
   > 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇 Configuration (組態)。
4. 在更新、監控和日誌記錄組態類別中，選擇編輯。
5. 在 Amazon X-Ray 區段中，選取已啟動。
6. 若要儲存變更，請選擇頁面底部的儲存變更。

您亦可於環境建立期間啟用此選項。如需更多詳細資訊，請參閱 建立新的環境精靈。

aws:elasticbeanstalk:xray 命名空間

您可於 `XRayEnabled` 命名空間使用 `aws:elasticbeanstalk:xray` 選項，以啟用除錯。

欲在您部署應用程式時自動啟用除錯，請於原始碼的組態檔案中設定選項，如下所示。

Example .ebextensions/debugging.config

```
option_settings:
  aws:elasticbeanstalk:xray:
    XRayEnabled: true
```

檢視您的 Elastic Beanstalk 環境日誌

AWS Elastic Beanstalk 提供兩種定期從 Amazon EC2 執行個體 (其執行您的應用程式) 檢視日誌的方式：

- 設定您的 Elastic Beanstalk 環境，將輪換的執行個體日誌上傳至環境的 Amazon S3 儲存貯體。
- 設定環境，將執行個體日誌串流至 Amazon CloudWatch Logs。

當您設定將執行個體日誌串流至 CloudWatch Logs 時，Elastic Beanstalk 會在 Amazon EC2 執行個體上建立代理和部署的 CloudWatch Logs 日誌群組，並將這些日誌檔案即時傳輸至 CloudWatch Logs。如需執行個體日誌的詳細資訊，請參閱 在 Elastic Beanstalk 環境中檢視 Amazon EC2 執行個體的日誌。

除了執行個體日誌，如果您為環境啟用增強行運作狀態，您可以將環境設定為將運作狀態資訊串流到 CloudWatch Logs。當環境的運作狀態變更，Elastic Beanstalk 會新增記錄到運作狀態日誌群組，並提供新的狀態和描述變更的原因。如需環境運作狀態串流的詳細資訊，請參閱 將 Elastic Beanstalk 環境運作狀態資訊串流至 Amazon CloudWatch Logs。

設定執行個體日誌檢視

如要檢視執行個體日誌，您可以在 Elastic Beanstalk 主控台啟用執行個體日誌輪換和日誌串流。

在 Elastic Beanstalk 主控台設定執行個體日誌輪換和日誌串流

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇 Configuration (組態)。
4. 在更新、監控和日誌記錄組態類別中，選擇編輯。
5. 在 S3 日誌儲存區段中，選擇輪換日誌下的已啟動以將輪換日誌上傳到 Amazon S3。
6. 在 Instance log streaming to CloudWatch Logs (執行個體日誌串流至 CloudWatch Logs) 區段中進行以下設定：
   - 日誌串流 – 選取已啟動以啟用日誌串流。
   - Retention (保留) – 指定在 CloudWatch Logs 中保留日誌的天數。
   - Lifecycle (生命週期) – 設為 Delete logs upon termination (終止即刪除日誌)，在環境終止立即刪除 CloudWatch Logs 的日誌，無須等待其過期。
7. 若要儲存變更，請選擇頁面底部的儲存變更。

您在啟用日誌串流後，返回 Software (軟體) 組態類別或頁面，並且尋找 Log Groups (日誌群組) 連結。按一下此連結，在 CloudWatch 主控台中查看您的執行個體日誌。

設定環境運作狀態日誌檢視

如要檢視環境運作狀態日誌，您可以在 Elastic Beanstalk 主控台啟用環境運作狀態日誌串流。

在 Elastic Beanstalk 主控台中設定環境運作狀態日誌串流

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇 Configuration (組態)。
4. 在更新、監控和日誌記錄組態類別中，選擇編輯。
5. 轉至監控區段。

6. 在 Health event streaming to CloudWatch Logs (運作狀態事件串流至 CloudWatch Logs) 之下設定以下設定：
   - 日誌串流 – 選擇已啟動以啟用日誌串流。
   - Retention (保留) – 指定在 CloudWatch Logs 中保留日誌的天數。
   - Lifecycle (生命週期) – 設為 Delete logs upon termination (終止即刪除日誌)，在環境終止立即刪除 CloudWatch Logs 的日誌，無須等待其過期。
7. 若要儲存變更，請選擇頁面底部的儲存變更。

日誌檢視命名空間

以下命名空間包含日誌檢視的設定：

- `aws:elasticbeanstalk:hostmanager` – 設定將輪換日誌上傳到 Amazon S3。
- `aws:elasticbeanstalk:cloudwatch:logs` – 設定將執行個體日誌串流至 CloudWatch。
- `aws:elasticbeanstalk:cloudwatch:logs:health` – 設定將環境運作狀態串流至 CloudWatch。

使用 Amazon SNS 的 Elastic Beanstalk 環境通知

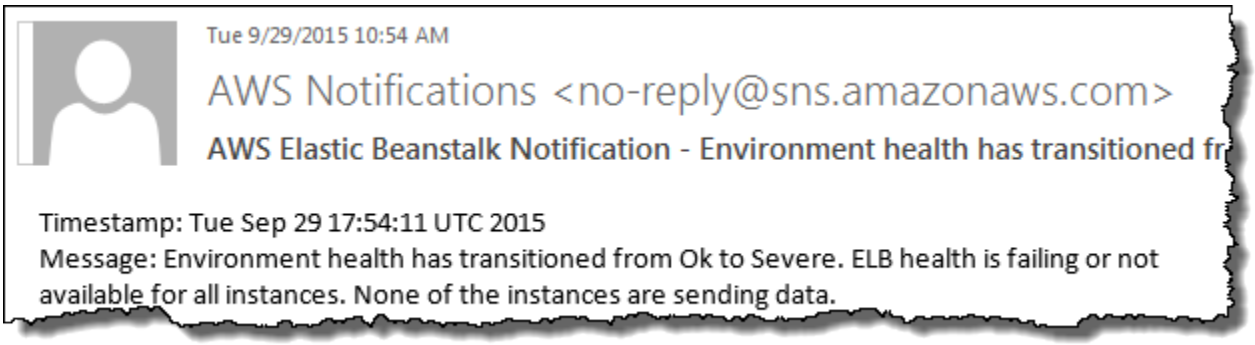
您可以將 AWS Elastic Beanstalk 環境設定為使用 Amazon Simple Notification Service (Amazon SNS) 來通知您影響應用程式的重要事件。若要在發生錯誤或環境運作狀態發生變更時接收 AWS 的電子郵件，請在建立環境或稍後建立環境時指定電子郵件地址。

> **Note**
>
> Elastic Beanstalk 使用 Amazon SNS 進行通知。如需 Amazon SNS 定價的相關資訊，請參閱 https://aws.amazon.com/sns/pricing/。

當您為環境設定通知時，Elastic Beanstalk 會代表您為您的環境建立 Amazon SNS 主題。若要傳送訊息給 Amazon SNS 主題，Elastic Beanstalk 必須具有所需的許可。如需更多詳細資訊，請參閱 設定傳送通知的許可。

當值得注意的事件發生時，Elastic Beanstalk 會傳送訊息到主題。然後，Amazon SNS 會將收到的訊息轉傳給該主題的訂閱者。值得注意的事件包括環境建立的錯誤，以及環境中和執行個體運作狀態的所

有變更。Amazon EC2 Auto Scaling 操作的事件 (例如為環境新增和移除執行個體) 和其他資訊參考事件，則不會觸發通知。



您可以在建立環境或之後某個時候，在 Elastic Beanstalk 主控台中輸入電子郵件地址。這將會建立一個 Amazon SNS 主題並訂閱。Elastic Beanstalk 會管理主題的生命週期，並且在您的環境終止時，或是當您在環境管理主控台中移除電子郵件地址時，將主題刪除。

`aws:elasticbeanstalk:sns:topics` 命名空間提供了選項，可藉由使用組態檔案、CLI 或 SDK 來設定 Amazon SNS 主題。透過使用這些方法之一，您可以設定訂閱者和端點的類型。對於訂閱者類型，您可以選擇 Amazon SQS 佇列或 HTTP URL。

您只能開啟或關閉 Amazon SNS 通知。根據您環境的規模和組成，傳送至主題的通知頻率可能很高。若要設定在特定情況下傳送通知，您可以選擇其他選項。您可以使用 Amazon EventBridge 設定事件驅動的規則，以便在 Elastic Beanstalk 發出符合特定條件的事件時通知您。或者，您也可以設定環境發佈自訂指標，並設定 Amazon CloudWatch 警示，以便在這些指標達到狀況不良的臨界值時通知您。

使用 Elastic Beanstalk 主控台設定通知

您可以在 Elastic Beanstalk 主控台輸入電子郵件地址，以為您的環境建立 Amazon SNS 主題。

在 Elastic Beanstalk 主控台中設定通知

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

   > **Note**
   >
   > 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇 Configuration (組態)。

4. 在更新、監控和日誌記錄組態類別中，選擇編輯。
5. 向下捲動至電子郵件通知區段。
6. 輸入電子郵件地址。
7. 若要儲存變更，請選擇頁面底部的儲存變更。

當您輸入通知用的電子郵件地址時，Elastic Beanstalk 會為您的環境建立 Amazon SNS 主題，並新增訂閱。Amazon SNS 會發送一封電子郵件到訂閱地址，來確認訂閱。您必須按一下確認電子郵件中的連結，來啟用訂閱和接收通知。

使用組態選項來設定通知

使用 `aws:elasticbeanstalk:sns:topics` 命名空間中的選項，來設定您環境的 Amazon SNS 通知。您可以使用組態檔案、CLI 或軟體開發套件來設定這些選項。

- 通知端點 – 要作為通知傳送目的地的電子郵件地址、Amazon SQS 佇列或 URL。如果您設定此選項，則會建立指定端點的 SQS 佇列和訂閱。如果端點並非電子郵件地址，您也必須設定 `Notification Protocol` 選項。SNS 會根據 `Notification Endpoint` 的值來驗證 `Notification Protocol` 的值。設定此選項多次，會產生額外的主題訂閱。如果您移除此選項，則會刪除主題。
- 通知通訊協定 – 用來傳送通知給 `Notification Endpoint` 的通訊協定。此選項的預設值為 `email`。將此選項設定為 `email-json`，會傳送 JSON 格式的電子郵件；設定為 `http` 或 `https`，會將 JSON 格式的通知發佈到 HTTP 端點；設定為 `sqs`，則會傳送通知到 SQS 佇列。

  > **Note**
  >
  > 不支援 AWS Lambda 通知。

- 通知主題 ARN - 在為您的環境設定通知端點之後，請讀取此設定，來取得 SNS 主題的 ARN。您也可以設定此選項，來使用現有的通知用 SNS 主題。當您變更此選項或終止環境時，您透過此選項連接到環境的主題不會遭到刪除。

  若要設定 Amazon SNS 通知，您需要擁有所需的許可。如果您的 IAM 使用者使用 Elastic Beanstalk AdministratorAccess-AWSElasticBeanstalk 受管使用者政策，則您應該已經擁有必要的許可，可設定 Elastic Beanstalk 為您的環境建立的預設 Amazon SNS 主題。但是，如果您設定了 Elastic Beanstalk 無法管理的 Amazon SNS 主題，則需要將下列政策新增至您的使用者角色。

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sns:SetTopicAttributes",
        "sns:GetTopicAttributes",
        "sns:Subscribe",
        "sns:Unsubscribe",
        "sns:Publish"
      ],
      "Resource": [
        "arn:aws:sns:us-east-2:123456789012:sns_topic_name"
      ]
    }
  ]
}
```

- 通知主題名稱 - 設定此選項，以自訂環境通知用的 Amazon SNS 主題的名稱。如果已存在具有相同名稱的主題，Elastic Beanstalk 會將該主題連接到環境。

  > **Warning**
  >
  > 如果使用 `Notification Topic Name` 來將現有的 SNS 主題連接到環境，則當您在未來終止環境或變更此設定時，Elastic Beanstalk 將會刪除該主題。

  如果您變更此選項，則 `Notification Topic ARN` 也會變更。如果主題已連接到環境，則 Elastic Beanstalk 會刪除舊主題，然後建立新主題和訂閱。

  透過使用自訂主題名稱，您也必須提供在外部建立之自訂主題的 ARN。受管的使用者政策不會自動偵測具有自訂名稱的主題，因此您必須向 IAM 使用者提供自訂 Amazon SNS 許可。使用類似於用於自訂主題 ARN 的政策，但包括下列新增項目：

  - 在 `Actions` 清單中包含另外兩個動作，特別是：`sns:CreateTopic`、`sns:DeleteTopic`
  - 如果要將 `Notification Topic Name` 從一個自訂主題名稱變更為另一個自訂主題名稱，您也必須在 `Resource` 清單中包含這兩個主題的 ARN。或者，包括涵蓋兩個主題的常規表達式。通過這種方式，Elastic Beanstalk 有許可刪除舊主題並建立新主題。

EB CLI 和 Elastic Beanstalk 主控台會為前述選項套用建議的數值。若您想要使用組態檔進行相同的設定，您必須移除這些設定。如需詳細資訊，請參閱「建議值」。

設定傳送通知的許可

本節討論與使用 Amazon SNS 進行通知相關的安全性考量。有兩種不同的情況：

- 使用 Elastic Beanstalk 為您的環境建立的預設 Amazon SNS 主題。
- 透過組態選項提供外部 Amazon SNS 主題。

Amazon SNS 主題的預設存取政策只允許主題擁有者發佈或訂閱該主題。但是，透過適當的政策組態，可以授予 Elastic Beanstalk 發佈至 Amazon SNS 主題的許可，以本節所述兩種情況之一的方式進行。如需詳細資訊，請參閱以下子節。

預設主題的許可

當您為環境設定通知時，Elastic Beanstalk 會為您的環境建立 Amazon SNS 主題。若要傳送訊息給 Amazon SNS 主題，Elastic Beanstalk 必須具有所需的許可。如果您的環境使用 Elastic Beanstalk 主控台或 EB CLI 為其產生的服務角色，或使用帳戶的監控服務連結角色，那麼您就無需執行其他操作。這些受管角色包括必要的許可，允許 Elastic Beanstalk 將訊息傳送至 Amazon SNS 主題。

不過，如果您在建立環境時提供自訂服務角色，請確定這個自訂服務角色包含下列政策。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sns:Publish"
      ],
      "Resource": [
        "arn:aws:sns:us-east-2:123456789012:ElasticBeanstalkNotifications*"
      ]
    }
  ]
}
```

外部主題的許可

使用組態選項來設定通知 會說明如何使用另一個 Amazon SNS 主題來取代 Elastic Beanstalk 提供的 Amazon SNS 主題。如果您取代了該主題，Elastic Beanstalk 必須確認您具有發佈至此 SNS 主題的

許可，才能將 SNS 主題與環境相關聯。您應該具備 `sns:Publish`。服務角色會使用相同的許可。為了確認這一情形，Elastic Beanstalk 會傳送測試通知給 SNS，做為您建立或更新環境動作程序的一部分。如果此測試失敗，則您嘗試建立或更新環境也會失敗。Elastic Beanstalk 會顯示說明此次失敗原因的訊息。

如果您為環境提供自訂服務角色，請確定您的自訂服務角色包含下列政策，以允許 Elastic Beanstalk 將訊息傳送至 Amazon SNS 主題。在下列程式碼中，以您在組態選項中提供的 Amazon SNS 主題名稱取代 *sns_topic_name*。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sns:Publish"
      ],
      "Resource": [
        "arn:aws:sns:us-east-2:123456789012:sns_topic_name"
      ]
    }
  ]
}
```

有關 Amazon SNS 存取控制的詳細資訊，請參閱《Amazon Simple Notification Service 開發人員指南》中的 Amazon SNS 存取控制範例。

透過 Elastic Beanstalk 設定 Amazon Virtual Private Cloud (Amazon VPC)

Amazon Virtual Private Cloud (Amazon VPC) 是是能安全地將流量路由傳送到在 Elastic Beanstalk 中執行您應用程式的 EC2 執行個體的聯網服務。當您啟動環境時，如果未設定 VPC，Elastic Beanstalk 會使用預設 VPC。

您可以啟動您的環境於自訂的 VPC，以自訂聯網和安全設定。Elastic Beanstalk 可讓您選擇要將哪些子網路用於您的資源，以及如何為您環境中的執行個體和負載平衡器設定 IP 地址。當您建立環境時，環境是鎖定到 VPC 的，但您可在執行中的環境變更子網路和 IP 地址。

> **Note**
>
> 如果您在 2013 年 12 月 4 日前建立 AWS 帳戶，您可能會在某些 AWS 區域中使用 Amazon EC2-Classic 網路組態的環境而非使用 Amazon VPC。如需將環境從 EC2-Classic 遷移至 VPC 網路組態的詳細資訊，請參閱將 Elastic Beanstalk 環境從 EC2-Classic 遷移至 VPC。

在 Elastic Beanstalk 主控台設定 VPC 設定

如果您在建立環境時選擇自訂 VPC，您可以在 Elastic Beanstalk 主控台修改該 VPC 設定。

若要設定您的環境 VPC 設定

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

   > **Note**
   >
   > 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇 Configuration (組態)。
4. 在 Network (網路) 組態類別中，選擇 Edit (編輯)。

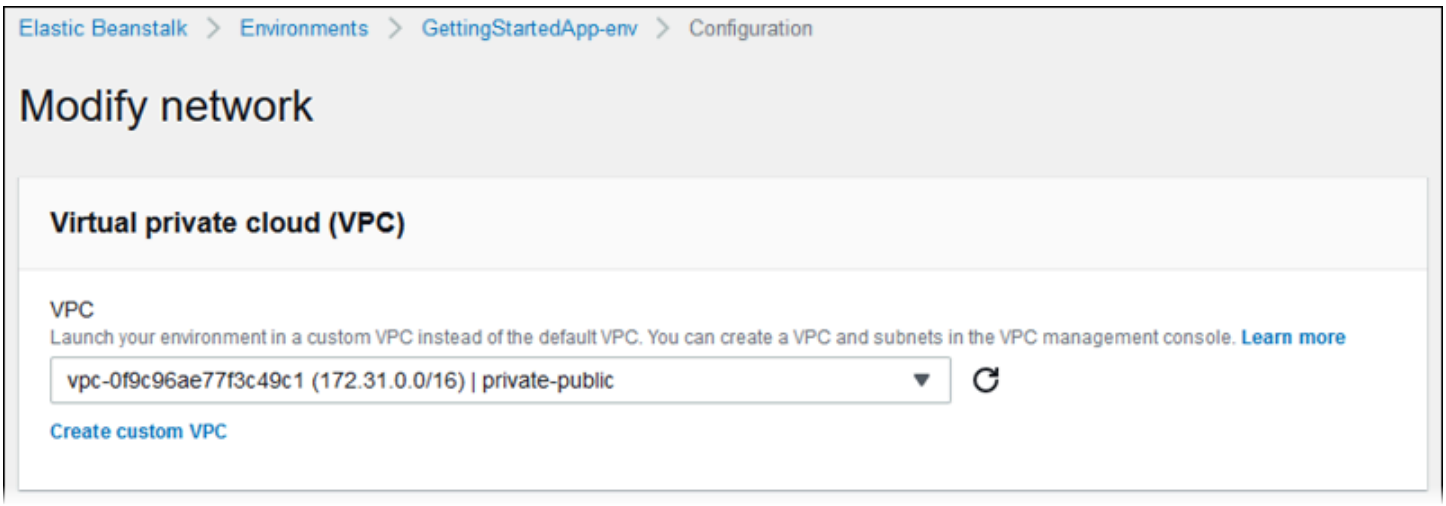
可用的設定如下所示。

選項

- VPC
- 負載平衡器可見性
- 負載平衡器子網路
- 執行個體的公有 IP 地址。
- 執行個體子網路
- 資料庫子網路

VPC

選擇您環境的 VPC。您只能在環境建立時變更此設定。

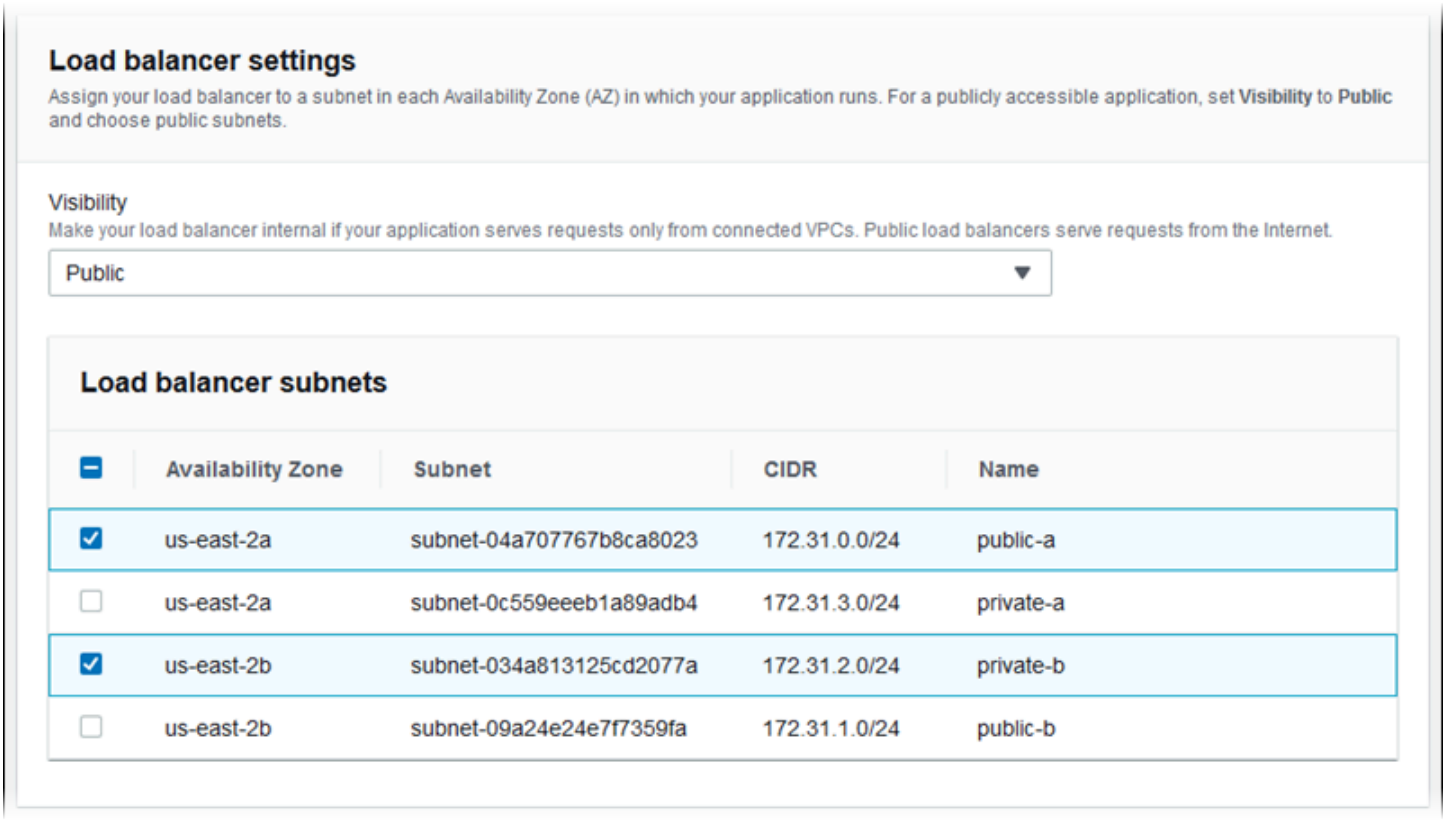


負載平衡器可見性

於負載平衡的環境中，選擇負載平衡器機制。在預設情況下，負載平衡器為公有狀態，包含公有的 IP 地址和網域名稱。如果您的應用程式只會從您的 VPC 或連線的 VPN 內部提供流量，取消選取此選項並選擇適用於您的負載平衡器的私有子網路，讓負載平衡器為內部使用且從網際網路停用存取。

負載平衡器子網路

於負載平衡的環境中，選擇您的負載平衡器用於提供流量的子網路。於公有應用程式中，選擇公有子網路。於多個可用區域使用子網路以有高可用性。於內部應用程式，請選擇私有子網路且停用負載平衡器的可見性。

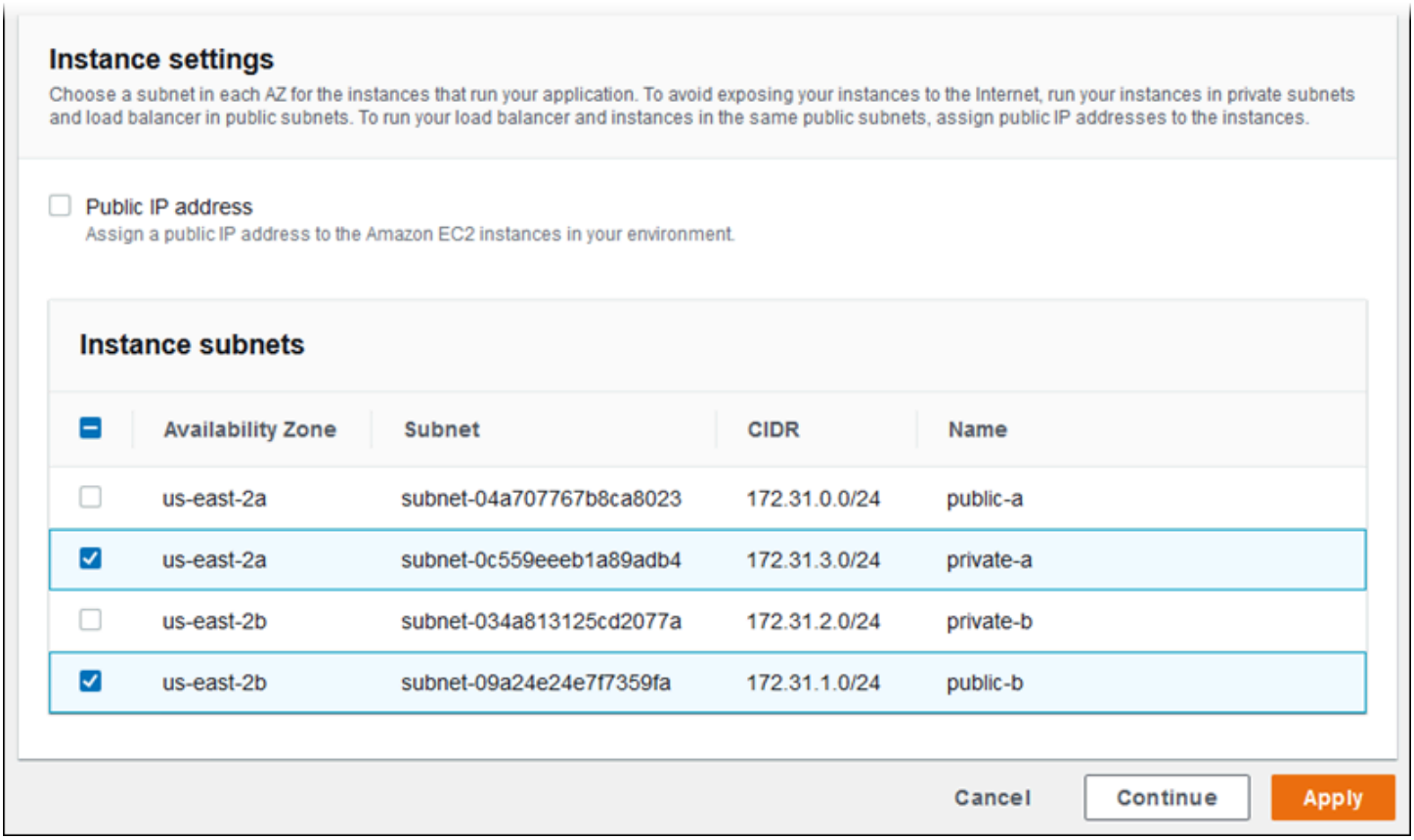


執行個體的公有 IP 地址。

如果您為您的應用程式執行個體選擇公有子網路，讓公有 IP 地址可從網際網路使其路由。

執行個體子網路

選擇適用於您的應用程式執行個體的子網路。為您的負載平衡器使用的每個可用區域，選擇至少一個子網路。如果您為您的執行個體選擇私有子網路，您的 VPC 必須擁有 NAT 閘道於可以用於存取網際網路的公有子網路執行個體。



資料庫子網路

執行連接至 Elastic Beanstalk 環境的 Amazon RDS 資料庫時，請為資料庫執行個體選擇子網路。為了高可用性，讓資料庫可 multi-AZ 且為每個可用區域選擇子網路。為了確保您的應用程式可以連接到您的資料庫，於相同子網路執行上述兩項。

aws:ec2:vpc 命名空間

您可以在 `aws:ec2:vpc` 命名空間中使用組態選項來設定您的環境網路設定。

以下組態檔案使用於此命名空間的選項來設定公有私有組態環境的 VPC 和子網路。為了設定 VPC ID 於組態檔案中，檔案必須包含在環境建立時的應用程式原始碼套件。如需環境建立時設定這些設定的其他方法，請參閱於環境建立期間設定組態選項。

Example .ebextensions/vpc.config - 公有 - 私有

```
option_settings:
   aws:ec2:vpc:
      VPCId: vpc-087a68c03b9c50c84
      AssociatePublicIpAddress: 'false'
```

```
      ELBScheme: public
      ELBSubnets: subnet-0fe6b36bcb0ffc462,subnet-032fe3068297ac5b2
      Subnets: subnet-026c6117b178a9c45,subnet-0839e902f656e8bd1
```

此範例顯示執行於相同公有子網路中的負載平衡器和 EC2 執行個體的公有 - 公有組態。

Example .ebextensions/vpc.config - 公有 - 公有

```
option_settings:
   aws:ec2:vpc:
      VPCId: vpc-087a68c03b9c50c84
      AssociatePublicIpAddress: 'true'
      ELBScheme: public
      ELBSubnets: subnet-0fe6b36bcb0ffc462,subnet-032fe3068297ac5b2
      Subnets: subnet-0fe6b36bcb0ffc462,subnet-032fe3068297ac5b2
```

將 Elastic Beanstalk 環境從 EC2-Classic 遷移至 VPC

本主題說明如何將 Elastic Beanstalk 環境從 EC2-Classic 網路平台遷移至 Amazon Virtual Private Cloud (Amazon VPC) 網路的不同選項。

如果您是在 2013 年 12 月 4 日之前建立 AWS 帳戶，則在某些 AWS 區域 區域中可能有使用 EC2-Classic 網路組態的環境。在 2013 年 12 月 4 日當日或之後建立的所有 AWS 帳戶在每個 AWS 區域都已成為僅適用於 VPC。唯一的豁免情況為，因為支援請求而啟用 Amazon EC2-Classic。

> **Note**
>
> 您可以在 Elastic Beanstalk 主控台的組態概觀頁面上的 Network configuration (網路組態) 類別中，檢視環境的網路組態設定。

您應該移轉的理由

Amazon EC2-Classic 將於 2022 年 8 月 15 日終止其標準支援。為避免工作負載中斷，建議您在 2022 年 8 月 15 日之前從 Amazon EC2-Classic 遷移到 VPC。我們也要求您在未來不要在 Amazon EC2-Classic 啟動任何 AWS 資源，而是使用 Amazon VPC。

當您將 Elastic Beanstalk 環境從 Amazon EC2-Classic 遷移到 Amazon VPC 時，您必須建立新的 AWS 帳戶。您還必須在您的新 AWS 帳戶中重新建立 AWS EC2-Classic 環境。不必為環境執行任何額

外的組態工作，即可使用預設 VPC。如果預設 VPC 不符合您的要求，您可以手動建立自訂 VPC 並將該 VPC 與您的環境建立關聯。

或者，如果您現有的 AWS 帳戶具有無法遷移至新 AWS 帳戶的資源，則可以將 VPC 新增至目前帳戶。然後，將您的環境設定為使用 VPC。

如需詳細資訊，請參閱 EC2-Classic 網路正在淘汰 - 本文介紹如何準備的部落格文章。

將環境從 EC2-Classic 遷移至新 AWS 帳戶 (建議)

如果您尚未在 2013 年 12 月 4 日或之後建立的 AWS 帳戶，請先建立新帳戶。您將會將您的環境遷移到這個新帳戶。

1. 您的新 AWS 帳戶會為其環境提供預設 VPC。如果您不需要建立自訂 VPC，請跳至步驟 2。

   您可以使用下列其中一種方式建立自訂 VPC：

   - 使用 Amazon VPC 主控台精靈，透過其中一個可用的組態選項，快速建立 VPC。如需詳細資訊，請參閱 Amazon VPC 主控台精靈組態。
   - 如果您對 VPC 有更具體的要求，請在 Amazon VPC 主控台上建立自訂 VPC。例如，如果您的使用案例需要特定數量的子網路，我們建議您執行這項操作。如需詳細資訊，請參閱 VPC 和子網路。
   - 如果您偏好在 Elastic Beanstalk 環境中使用 AWS CloudFormation 範本，請使用 GitHub 網站上的 elastic-beanstalk-samples 儲存庫建立 VPC。此儲存庫包含 AWS CloudFormation 範本。如需更多詳細資訊，請參閱 搭配 Amazon VPC 使用 Elastic Beanstalk。

   > **Note**
   >
   > 您也可以在使用建立新環境精靈在新 AWS 帳戶中重新建立環境時，建立自訂 VPC。如果您使用精靈並選擇建立自訂 VPC，則精靈將您重新引導至 Amazon VPC 主控台。

2. 在您的新 AWS 帳戶建立新環境。我們建議該環境包含與您要遷移的 AWS 帳戶中現有環境相同的組態。您可以使用下列其中一種方法來執行這項作業。

   > **Note**
   >
   > 如果您的新環境必須在遷移之後使用相同的 CNAME，請在 EC2-Classic 平台上終止原始環境。這會釋出 CNAME 以供使用。然而，執行這樣的方式將會導致該環境停機，並且可

能會面臨的風險是另一個客戶可能在您終止 EC2-Classic 環境和建立新環境之間的這段時間內，選擇您的 CNAME。如需更多詳細資訊，請參閱 終止 Elastic Beanstalk 環境。對於擁有自己專屬網域名稱的環境，CNAME 沒有這個問題。您只需更新您的網域名稱系統 (DNS)，即可將請求轉寄到您的新 CNAME。

- 使用 Elastic Beanstalk 主控台的建立新環境精靈。該精靈提供建立自訂 VPC 的選項。如果您選擇不建立自訂 VPC，則會指派預設 VPC。
- 使用 Elastic Beanstalk 命令列界面 (EB CLI) 在新的 AWS 帳戶中重新建立環境。eb create 命令描述中的其中一個範例會示範在自訂 VPC 中建立環境的方法。如果您不提供 VPC ID，則環境會使用預設 VPC。

  使用這種方法，您就可以在兩個 AWS 帳戶中使用已儲存的組態檔案。因此，您不需要手動輸入所有的組態資訊。然而，您必須需要使用 eb config save 命令來儲存您正在遷移的 EC2-Classic 環境的組態設定。將儲存的組態檔案複製到新帳號環境的新目錄。

  > **Note**
  >
  > 您必須編輯在已儲存的組態檔案中的某些資料，然後才能在新帳戶中使用。您還必須使用新帳戶的正確資料來更新先前帳戶的資訊。例如，您必須將 AWS Identity and Access Management (IAM) 角色的 Amazon 資源名稱 (ARN) 替換為新帳戶的 IAM 角色 ARN。

  如果您使用具有 `cfg` 的 eb create 命令，將會使用指定的已儲存組態檔案建立新環境。如需更多詳細資訊，請參閱 使用 Elastic Beanstalk 已儲存組態。

在同一個 AWS 帳戶中從 EC2-Classic 遷移環境

您現有的 AWS 帳戶可能有無法遷移至新 AWS 帳戶的資源。在這樣的情況下，您必須重新建立環境，並為您建立的每個環境手動設定 VPC。

將您的環境移轉至自訂 VPC

先決條件

在開始之前，您必須先擁有 VPC。您可以使用下列其中一種方式建立非預設 (自訂) VPC：

- 使用 Amazon VPC 主控台精靈，透過其中一個可用的組態選項，快速建立 VPC。如需詳細資訊，請參閱 Amazon VPC 主控台精靈組態。
- 如果您對 VPC 有更具體的要求，請在 Amazon VPC 主控台上建立自訂 VPC。例如，如果您的使用案例需要特定數量的子網路，我們建議您執行這項操作。如需詳細資訊，請參閱 VPC 和子網路。
- 如果您偏好在 Elastic Beanstalk 環境中使用 AWS CloudFormation 範本，請使用 GitHub 網站上的 elastic-beanstalk-samples 儲存庫建立 VPC。此儲存庫包含 AWS CloudFormation 範本。如需更多詳細資訊，請參閱 搭配 Amazon VPC 使用 Elastic Beanstalk。

在下列步驟中，您可以在新環境中設定 VPC 時使用產生的 VPC ID 和子網路 ID。

1. 建立包含與現有環境相同組態的新環境。您可以使用下列其中一種方法來執行這項作業。

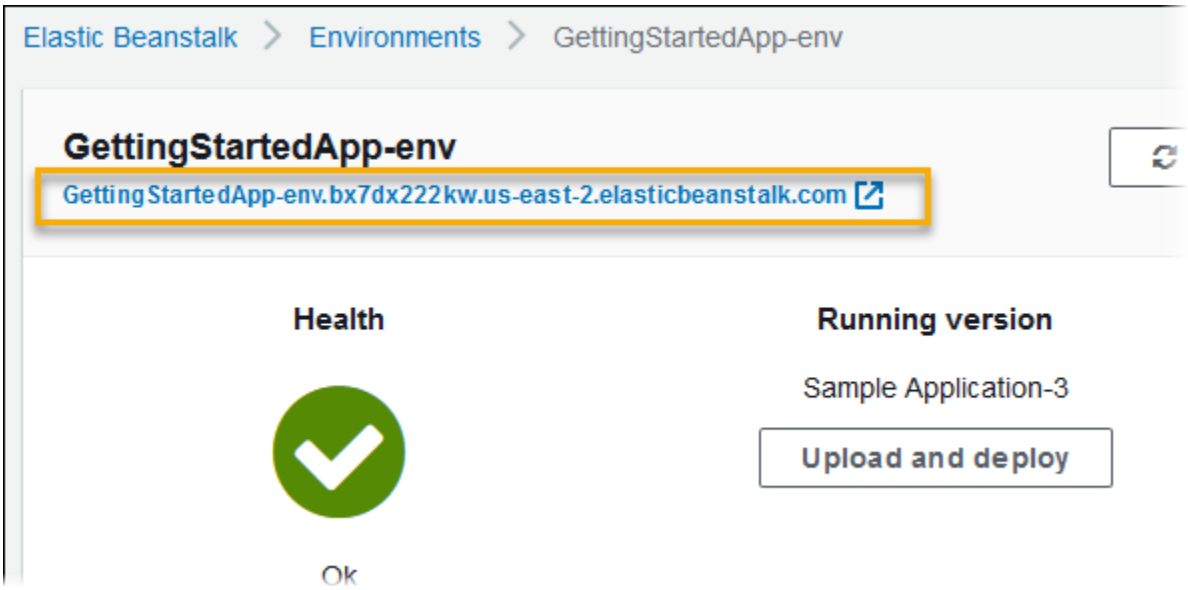
   > **Note**
   >
   > 儲存的組態功能可協助您在新帳戶中重新建立環境。此功能可儲存環境資訊，因此您可以在建立或更新其他環境時套用它。如需更多詳細資訊，請參閱 使用 Elastic Beanstalk 已儲存組態。

   - 使用 Elastic Beanstalk 主控台，在要您設定新環境時，從 EC2-Classic 環境套用已儲存的組態。此組態將會使用 VPC。如需更多詳細資訊，請參閱 使用 Elastic Beanstalk 已儲存組態。
   - 使用 Elastic Beanstalk 命令行界面 (EB CLI)，執行 eb create 命令以重新建立您的環境。提供原始環境的參數和 VPC 識別碼。eb create 命令描述中的其中一個範例會說明如何在自訂 VPC 中建立環境。
   - 使用 AWS Command Line Interface (AWS CLI)，並使用 elasticbeanstalk create-environment 命令重新建立您的環境。使用 VPC 識別碼提供原始環境的參數。如需指示，請參閱 使用 AWS CLI 建立 Elastic Beanstalk 環境。
2. 交換現有環境與新環境的 CNAME。如此一來，則可以使用熟悉的位址來參考您建立的新環境。您可以使用 EB CLI 或 AWS CLI。
   - 使用 EB CLI，執行 eb swap 命令交換環境的 CNAME。如需更多詳細資訊，請參閱 使用 Elastic Beanstalk 命令列界面 (EB CLI)。
   - 使用 AWS CLI，將環境的 CNAME 與 elasticbeanstalk swap-environment-cnames 命令交換。如需詳細資訊，請參閱《AWS CLI 命令參考》。

您 Elastic Beanstalk 環境的網域名稱

在預設情況下，您的環境可供 `elasticbeanstalk.com` 子網域的使用者使用。建立環境時，您可以為您的應用程式選擇主機名稱。子網域與網域會自動填入 *`region`*`.elasticbeanstalk.com`。

為了將使用者轉送至您的環境，Elastic Beanstalk 會登錄 CNAME 記錄，指向您環境的負載平衡器。您可以在 Elastic Beanstalk 主控台的環境概觀頁面中看到環境應用程式的 URL 與 CNAME 目前值。



選擇概觀頁面上的 URL，或在導覽窗格中選擇 Go to environment (移至環境)，導覽至應用程式的網頁。

您可將您環境的 CNAME 與其他環境的互換，來加以變更。如需說明，請參閱 透過 Elastic Beanstalk 進行藍/綠部署。

如果您擁有網域名稱，可以使用 Amazon Route 53 來將該名稱解析對應到您的環境。您可以使用 Amazon Route 53 來購買網域名稱，或使用從其他供應商所購買的名稱。

若要使用 Route 53 來購買網域名稱，請參閱《Amazon Route 53 開發人員指南》中的註冊新網域。

若要進一步了解使用自訂網域的詳細資訊，請參閱《Amazon Route 53 開發人員指南》中的將流量路由到 AWS Elastic Beanstalk 環境。

> **Important**
>
> 如果您終止環境，您也必須刪除您建立的任何 CNAME 映射，因為其他客戶可能重複使用可用的主機名稱。請務必刪除指向終止環境的 DNS 記錄，以防止懸置 DNS 項目。懸置的 DNS 項

目可能會降低您網域的網際網路流量的安全性，使其暴露在易於攻擊的弱點中。另外，它還可能存在其他風險。

如需詳細資訊，請參閱 Amazon Route 53 開發人員指南中的在 Route 53 上防止懸置委派記錄。您也可以在AWS安全部落格中針對適用於 Amazon CloudFront 請求的強化網域保護來進一步了解懸置 DNS 項目的資訊。

設定 Elastic Beanstalk 環境 (進階)

當您建立 AWS Elastic Beanstalk 環境時，Elastic Beanstalk 會佈建和設定所有需要的 AWS 資源，來執行和支援您的應用程式。除了設定環境的中繼資料和更新行為之外，您也可以藉由提供組態選項的值，來自訂這些資源。例如，您可以新增 Amazon SQS 佇列和監控佇列深度的警示，或是新增 Amazon ElastiCache 叢集。

大多數的組態選項具有預設值，會由 Elastic Beanstalk 自動套用。您可以透過組態檔案、儲存的組態、命令列選項，直接呼叫 Elastic Beanstalk API，來覆寫這些預設值。EB CLI 和 Elastic Beanstalk 主控台也會為某些選項套用建議的數值。

在部署應用程式版本的同時，您可以在原始碼套件中加入組態檔案，來輕鬆自訂您的環境。在自訂您執行個體上的軟體時，相較於建立自訂 AMI，使用組態檔案的做法具有更多優點，因為您不需要維持一組 AMI。

在部署您的應用程式時，您可以自訂和設定應用程式所需的軟體。這些檔案可以是應用程式所需的相依檔案 (例如由 yum 儲存庫所提供的其他套件) 或組態檔案 (例如 httpd.conf 的替代檔案，可用來覆寫掉 AWS Elastic Beanstalk 預設的特定設定)。

主題

組態選項

Elastic Beanstalk 定義大量組態選項，您可藉此設定環境的行為和其中的資源。組態選項以命名空間分類，例如，`aws:autoscaling:asg`，定義環境 Auto Scaling 群組的選項。

在您建立環境時，Elastic Beanstalk 主控台和 EB CLI 會設定組態選項，其中包括您明確設定的選項以及用戶端定義的建議值。您亦可於已儲存組態和組態檔案中設定組態選項。若在多個位置設定相同選項，使用的數值將取決於優先順序。

在環境建立前，組態選項設定能夠由文字格式組成並儲存，再於環境建立期間透過支援的用戶端套用，環境建立後則能夠新增、修改或移除。如需這三階段使用組態選項的可用方法詳細分析，請閱讀下列主題：

- 於環境建立前設定組態選項
- 於環境建立期間設定組態選項
- 於環境建立後設定組態選項

如需命名空間和選項的完整清單，以及各自的預設值和支援的值，請參閱 適用於所有環境的一般選項 和 平台特定選項。

優先順序

在環境建立期間，多個來源的組態選項依下列優先順序 (自最高到最低) 套用：

- 直接套用至環境的設定 – 由任何用戶端 (包含 Elastic Beanstalk 主控台、EB CLI、AWS CLI 和軟體開發套件) 利用 Elastic Beanstalk API 在建立環境或更新環境操作期間指定的設定。Elastic Beanstalk 主控台和 EB CLI 亦針對此層級的部分選項套用建議值 (除非受到覆寫)。
- 已儲存組態 – 未直接套用至環境的選項設定會從已儲存組態載入 (若已指定)。
- 組態檔案 (.ebextensions) – 未直接套用至環境且未於已儲存組態指定的選項設定，會自應用程式原始碼套件根目錄的 `.ebextensions` 資料夾內的組態檔案載入。

  組態檔案會依字母順序執行。例如，`.ebextensions/01run.config` 在 `.ebextensions/02do.config` 之前執行。
- 預設值 – 若組態選項具備預設值，此值僅於上述層級未設定此選項時套用。

若在多個位置定義相同組態選項，將套用具備最高優先順序的設定。若採用已儲存組態套用的設定或採用直接套用至環境的設定，該設定將做為環境資訊的一部分存放。您可以透過 AWS CLI 或透過 EB CLI 來移除這些設定。

組態檔案內的設定不會直接套用至環境，且若未修改組態檔案並部署新的應用程式版本，將無法移除。若以其他方法套用的設定被移除，相同設定將自原始碼套件的組態檔案載入。

例如，假設您於環境建立期間，透過 Elastic Beanstalk 主控台、命令列選項或已儲存組態，將環境內的執行個體數量下限設定為 5。而您的應用程式原始碼套件納入的組態檔案，將執行個體數量下限設定為 2。

在您建立環境時，Elastic Beanstalk 會將 MinSize 命名空間中的 aws:autoscaling:asg 選項設定為 5。若之後環境資訊移除該選項，此值會從組態檔案載入，並將執行個體數量下限設定為 2。若之後原始碼套件移除該組態檔案並重新部署，Elastic Beanstalk 會使用預設設定 1。

建議值

Elastic Beanstalk 命令列界面 (EB CLI) 和 Elastic Beanstalk 主控台均為部分組態選項提供建議值。這些值可與預設值不同，且是在您建立環境時由 API 層級設定。建議值可讓 Elastic Beanstalk 改善預設環境資訊，不會使 API 回溯成為不相容的變更。

例如，EB CLI 和 Elastic Beanstalk 主控台均可針對 EC2 執行個體類型設定組態選項 (InstanceType 命名空間內的 aws:autoscaling:launchconfiguration)。各個用戶端可採用不同方式來覆寫預設設定。在主控台內，您可於 Create New Environment (建立新的環境) 精靈的 Configuration Details (組態詳細資訊) 頁面，自其中的下拉式選單選擇不同的執行個體類型。若採用 EB CLI，則可針對 --instance_type 使用 eb create 參數。

由於建議值是在 API 層級設定，這些值會覆寫您於組態檔案或已儲存組態內設定的相同選項的數值。下列選項會經過設定：

Elastic Beanstalk 主控台

- 命名空間：aws:autoscaling:launchconfiguration

  選項名稱：IamInstanceProfile、EC2KeyName、InstanceType
- 命名空間：aws:autoscaling:updatepolicy:rollingupdate

  選項名稱：RollingUpdateType、RollingUpdateEnabled
- 命名空間：aws:elasticbeanstalk:application

  選項名稱：Application Healthcheck URL
- 命名空間：aws:elasticbeanstalk:command

  選項名稱：DeploymentPolicy、BatchSize、BatchSizeType
- 命名空間：aws:elasticbeanstalk:environment

  選項名稱：ServiceRole
- 命名空間：aws:elasticbeanstalk:healthreporting:system

  選項名稱：SystemType 和 HealthCheckSuccessThreshold

- 命名空間：aws:elasticbeanstalk:sns:topics

  選項名稱：Notification Endpoint
- 命名空間：aws:elasticbeanstalk:sqsd

  選項名稱：HttpConnections
- 命名空間：aws:elb:loadbalancer

  選項名稱：CrossZone
- 命名空間：aws:elb:policies

  選項名稱：ConnectionDrainingTimeout、ConnectionDrainingEnabled

EB CLI

- 命名空間：aws:autoscaling:launchconfiguration

  選項名稱：IamInstanceProfile、InstanceType
- 命名空間：aws:autoscaling:updatepolicy:rollingupdate

  選項名稱：RollingUpdateType、RollingUpdateEnabled
- 命名空間：aws:elasticbeanstalk:command

  選項名稱：BatchSize 和 BatchSizeType
- 命名空間：aws:elasticbeanstalk:environment

  選項名稱：ServiceRole
- 命名空間：aws:elasticbeanstalk:healthreporting:system

  選項名稱：SystemType
- 命名空間：aws:elb:loadbalancer

  選項名稱：CrossZone
- 命名空間：aws:elb:policies

  選項名稱：ConnectionDrainingEnabled

於環境建立前設定組態選項

AWS Elastic Beanstalk 支援大量組態選項，可讓您修改套用至環境之資源的設定。您可覆寫數個選項的預設值，以自訂您的環境。您亦可設定其他選項來啟用其他功能。

Elastic Beanstalk 支援已儲存組態選項設定的兩種方法。YAML 或 JSON 格式的組態檔案可納入名為 `.ebextensions` 之目錄的應用程式原始碼，並成為應用程式原始碼套件的一部分進行部署。您會於本機建立並管理組態檔案。

已儲存組態是您自執行環境或 JSON 選項檔案建立並存放於 Elastic Beanstalk 的範本。現有已儲存組態亦可進行擴展，以建立新的組態。

> **Note**
>
> 組態檔案和已儲存組態內定義的設定 (包括 Elastic Beanstalk 主控台和 EB CLI 套用的建議值)，優先順序低於環境建立期間或之後所進行的設定。如需詳細資訊，請參閱「優先順序」。

您亦可以 JSON 文件來指定選項，並在透過 EB CLI 或 建立或更新環境時直接提供給 Elastic BeanstalkAWS CLI 以這種方式直接提供給 Elastic Beanstalk 的選項，會覆寫其他方法提供的選項。

如需可用選項的完整清單，請參閱組態選項。

方法

- 組態檔案 (.ebextensions)
- 已儲存的組態
- JSON 文件
- EB CLI 組態

組態檔案 (`.ebextensions`)

使用 `.ebextensions` 來設定讓應用程式正常運作的選項，並提供可由更高層級優先順序覆寫之其他選項的預設值。`.ebextensions` 指定的選項優先順序最低，會被任何層級的設定覆寫。

欲使用組態檔案，請於專案原始碼的最上層建立名為 `.ebextensions` 的資料夾。新增副檔名為 `.config` 的檔案，並以下列方式指定選項：

```
option_settings:
```

```
  - namespace:  namespace
    option_name:  option name
    value:  option value
  - namespace:  namespace
    option_name:  option name
    value:  option value
```

例如，下列組態檔案會將應用程式的運作狀態檢查 url 設定為 `/health`：

`healthcheckurl.config`

```
option_settings:
  - namespace:  aws:elasticbeanstalk:application
    option_name:  Application Healthcheck URL
    value:  /health
```

JSON 為：

```
{
 "option_settings" :
    [
      {
        "namespace" : "aws:elasticbeanstalk:application",
        "option_name" : "Application Healthcheck URL",
        "value" : "/health"
      }
    ]
}
```

這會將 Elastic Beanstalk 環境內的 Elastic Load Balancing 負載平衡器，設定為將 HTTP 請求發送至路徑 `/health` 給各個 EC2 執行個體，以判斷其是否正常運作。

> **Note**
>
> YAML 憑藉一致的縮排。請在取代範例組態檔中的內容時，讓縮排層級一致，並確認您的文字編輯器使用空格而非定位字元進行縮排。

將 `.ebextensions` 目錄納入應用程式原始碼套件，並將其部署至新的或現有 Elastic Beanstalk 環境。

除了 `option_settings` 之外，組態檔案亦支援數個區段，可自訂執行於您環境伺服器上的軟體和檔案。如需更多詳細資訊，請參閱 .Ebextensions。

已儲存的組態

可使用 Elastic Beanstalk 主控台、EB CLI 或 來建立已儲存組態，藉此儲存在環境建立期間或之後套用至現有環境的設定AWS CLI 已儲存組態屬於應用程式，可套用至應用程式新的或現有的環境。

用戶端

- Elastic Beanstalk 主控台
- EB CLI
- AWS CLI

Elastic Beanstalk 主控台

建立已儲存組態 (Elastic Beanstalk 主控台)

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

   > **Note**
   > 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 選擇 Actions (動作)，然後選擇 Save Configuration (儲存組態)。
4. 使用畫面顯示對話方塊來完成動作。

已儲存組態會存放在 Elastic Beanstalk S3 儲存貯體內以您的應用程式命名之資料夾。例如，us-west-2 區域中對於 123456789012 帳戶一個名為 `my-app` 應用程式的組態位於 `s3://elasticbeanstalk-us-west-2-123456789012/resources/templates/my-app`。

EB CLI

EB CLI 亦於 eb config 下提供子命令，可與已儲存組態互動：

欲建立已儲存組態 (EB CLI)

1. 儲存連接環境的目前組態：

```
~/project$ eb config save --cfg my-app-v1
```

EB CLI 會將組態儲存至 ~/project/.elasticbeanstalk/saved_configs/*my-app-v1*.cfg.yml。

2. 視需要於本機修改已儲存組態。
3. 將已儲存組態上傳至 S3：

```
~/project$ eb config put my-app-v1
```

AWS CLI

從執行環境透過 aws elasticbeanstalk create-configuration-template 建立已儲存組態。

建立已儲存組態 (AWS CLI)

1. 透過 Elastic Beanstalk 辨識您 describe-environments 環境的環境 ID：

```
$ aws elasticbeanstalk describe-environments --environment-name my-env
{
    "Environments": [
        {
            "ApplicationName": "my-env",
            "EnvironmentName": "my-env",
            "VersionLabel": "89df",
            "Status": "Ready",
            "Description": "Environment created from the EB CLI using \"eb create
\"",
            "EnvironmentId": "e-vcghmm2zwk",
            "EndpointURL": "awseb-e-v-AWSEBLoa-1JUM8159RA11M-43V6ZI1194.us-
west-2.elb.amazonaws.com",
            "SolutionStackName": "64bit Amazon Linux 2015.03 v2.0.2 running Multi-
container Docker 1.7.1 (Generic)",
            "CNAME": "my-env-nfptuqaper.elasticbeanstalk.com",
            "Health": "Green",
            "AbortableOperationInProgress": false,
            "Tier": {
                "Version": " ",
                "Type": "Standard",
```

```
                "Name": "WebServer"
            },
            "HealthStatus": "Ok",
            "DateUpdated": "2015-10-01T00:24:04.045Z",
            "DateCreated": "2015-09-30T23:27:55.768Z"
        }
    ]
}
```

2. 透過 `create-configuration-template` 儲存環境的目前組態：

```
$ aws elasticbeanstalk create-configuration-template --environment-id e-vcghmm2zwk
 --application-name my-app --template-name v1
```

Elastic Beanstalk 會將組態儲存到 Amazon S3 中的 Elastic Beanstalk 儲存貯體中。

JSON 文件

若您使用 AWS CLI 來建立並更新環境，亦可提供 JSON 格式的組態選項。若您使用 AWS CLI 來建立並管理環境，JSON 格式的組態檔案程式庫非常實用。

例如，下列 JSON 文件會將應用程式的運作狀態檢查 url 設定為 `/health`：

~/ebconfigs/healthcheckurl.json

```
[
  {
    "Namespace": "aws:elasticbeanstalk:application",
    "OptionName": "Application Healthcheck URL",
    "Value": "/health"
  }
]
```

EB CLI 組態

EB CLI 除了透過 eb config 命令支援已儲存組態和直接的環境資訊，還有另一個組態檔案，其中名為 `default_ec2_keyname` 的選項可用來指定 Amazon EC2 金鑰對，以便對環境中執行個體進行 SSH 存取。EB CLI 在 `EC2KeyName` 命名空間使用此選項設定 `aws:autoscaling:launchconfiguration` 組態選項。

~/workspace/my-app/.elasticbeanstalk/config.yml

```
branch-defaults:
  master:
    environment: my-env
  develop:
    environment: my-env-dev
deploy:
  artifact: ROOT.war
global:
  application_name: my-app
  default_ec2_keyname: my-keypair
  default_platform: Tomcat 8 Java 8
  default_region: us-west-2
  profile: null
  sc: git
```

於環境建立期間設定組態選項

使用 Elastic Beanstalk 主控台、EB CLI、AWS CLI、開發套件或 Elastic Beanstalk API 建立 AWS Elastic Beanstalk 環境時，您可提供組態選項的值來自訂您的環境和其中啟動的 AWS 資源。

以非一次性的組態變更而言，您可於本機、原始碼套件或 Amazon S3 存放組態檔案。

這個主題包括於環境建立期間設定組態選項之所有方法的程序。

用戶端

- 在 Elastic Beanstalk 主控台中
- 使用 EB CLI
- 使用 AWS CLI

在 Elastic Beanstalk 主控台中

當您於 Elastic Beanstalk 主控台建立 Elastic Beanstalk 環境時，可透過組態檔案、已儲存組態和 Create New Environment (建立新的環境) 精靈內的表單，提供組態選項。

方法

- 使用組態檔案 (.ebextensions)
- 使用已儲存組態
- 使用新的環境精靈

使用組態檔案 (**.ebextensions**)

將 .config 檔案放入應用程式原始碼套件內名為 .ebextensions 的資料夾。

如需組態檔案的詳細資訊，請參閱 .Ebextensions。

```
~/workspace/my-app-v1.zip
|-- .ebextensions
|   |-- environmentvariables.config
|   `-- healthcheckurl.config
|-- index.php
`-- styles.css
```

通常會在環境建立期間將原始碼套件上傳至 Elastic Beanstalk。

Elastic Beanstalk 主控台會針對部分組態選項套用建議值，其他選項則提供表單欄位。Elastic Beanstalk 主控台設定的選項，會直接套用至環境，並覆寫組態檔案中的設定。

使用已儲存組態

使用 Elastic Beanstalk 主控台建立新的環境時，第一個步驟就是選擇組態。組態可為預先定義的組態，通常是平台 (如 PHP 或 Tomcat) 的最新版本，也可以是已儲存組態。

欲在環境建立期間套用已儲存組態 (Elastic Beanstalk 主控台)

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Applications (應用程式)，然後在清單上選取應用程式名稱。

   > **Note**
   >
   > 如果您有許多應用程式，請使用搜尋列來篩選應用程式清單。

3. 在導覽窗格中，找到應用程式名稱並選擇 Saved configurations (已儲存的配置)。
4. 選取您要套用的已儲存組態，然後選擇 Launch environment (啟動環境)。
5. 繼續透過精靈來建立您的環境。

已儲存組態僅適用特定應用程式。如需建立已儲存組態的詳細資訊，請參閱 已儲存的組態。

使用新的環境精靈

多數標準組態選項會顯示在建立新環境精靈的 Configure more options (設定更多選項) 頁面。若您為環境建立 Amazon RDS 資料庫或設定 VPC，其他組態選項也可用於這些資源。

在環境建立期間設定已儲存組態 (Elastic Beanstalk 主控台)

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Applications (應用程式)。
3. 選擇或建立應用程式。
4. 選擇 Actions (動作)，然後選擇 Create environment (建立環境)。
5. 繼續執行精靈，選擇 Configure more options (設定多個選項)。
6. 選擇任何 configuration presets (組態預設)，然後在一或多個組態類別選擇 Edit (編輯) 以變更一組相關的組態選項。
7. 當您選定選項，請選擇 Create environment (建立環境)。

新的環境精靈中所設定的任何選項，都會直接設定至環境，並覆寫您套用的已儲存組態或組態檔案 (`.ebextensions`) 內的選項設定。使用 EB CLI 或 AWS CLI 建立環境後，您可以移除設定，允許已儲存組態或組態檔案內的設定浮現。

如需新環境精靈的詳細資訊，請參閱 建立新的環境精靈。

使用 EB CLI

方法

- 使用組態檔案 (.ebextensions)
- 使用已儲存組態
- 使用命令列選項

使用組態檔案 (`.ebextensions`)

於您專案資料夾的 `.config` 底下納入 `.ebextensions`，與應用程式程式碼一同部署。

如需組態檔案的詳細資訊，請參閱 .Ebextensions。

```
~/workspace/my-app/
|-- .ebextensions
|   |-- environmentvariables.config
```

```
|   `-- healthcheckurl.config
|-- .elasticbeanstalk
|   `-- config.yml
|-- index.php
`-- styles.css
```

透過 eb create 建立您的環境，並部署原始碼。

```
~/workspace/my-app$ eb create my-env
```

使用已儲存組態

欲在使用 eb create 建立環境時套用已儲存組態，請使用 --cfg 選項。

```
~/workspace/my-app$ eb create --cfg savedconfig
```

已儲存組態可存放於專案資料夾內，或者 Amazon S3 上的 Elastic Beanstalk 儲存位置。在前面範例中，EB CLI 首先會尋找資料夾 savedconfig.cfg.yml 內名為 .elasticbeanstalk/saved_configs/ 的已儲存組態檔案。使用 .cfg.yml 套用已儲存組態時，請不要納入檔案副檔名 (--cfg)。

```
~/workspace/my-app/
|-- .ebextensions
|   `-- healthcheckurl.config
|-- .elasticbeanstalk
|   |-- saved_configs
|   |   `-- savedconfig.cfg.yml
|   `-- config.yml
|-- index.php
`-- styles.css
```

若 EB CLI 於本機找不到該組態，則會尋找 Amazon S3 的 Elastic Beanstalk 儲存位置。如需建立、編輯和上傳已儲存組態的詳細資訊，請參閱已儲存的組態。

使用命令列選項

EB CLI eb create 命令有數個選項，您可用其在環境建立期間設定組態選項。您可以使用這些選項來將 RDS 資料庫新增至您的環境、設定 VPC，或覆寫建議值。

例如，EB CLI 預設使用 t2.micro 執行個體類型。欲選擇不同的執行個體類型，請使用 --instance_type 選項。

```
$ eb create my-env --instance_type t2.medium
```

欲建立 Amazon RDS 資料庫執行個體並將其連接至您的環境，請使用 `--database` 選項。

```
$ eb create --database.engine postgres --database.username dbuser
```

若您省略建立環境所需的環境名稱、資料庫密碼或其他參數，EB CLI 會提示您將其輸入。

如需可用選項完整清單及使用範例，請參閱 eb create。

使用 AWS CLI

當您透過 AWS CLI 使用 `create-environment` 命令來建立 Elastic Beanstalk 環境時，AWS CLI 不會套用任何建議值。您指定原始碼套件，其中的組態檔案會定義所有組態選項。

方法

- 使用組態檔案 (.ebextensions)
- 使用已儲存組態
- 使用命令列選項

使用組態檔案 (**`.ebextensions`**)

若要將組態檔案套用至您透過 AWS CLI 建立的環境，請將其納入您上傳至 Amazon S3 的應用程式原始碼套件。

如需組態檔案的詳細資訊，請參閱 .Ebextensions。

```
~/workspace/my-app-v1.zip
|-- .ebextensions
|   |-- environmentvariables.config
|   `-- healthcheckurl.config
|-- index.php
`-- styles.css
```

使用 上傳應用程式原始碼套件並建立環境AWS CLI

1. 若您的 Amazon S3 內還沒有 Elastic Beanstalk 儲存貯體，請使用 `create-storage-location` 建立。

```
$ aws elasticbeanstalk create-storage-location
{
    "S3Bucket": "elasticbeanstalk-us-west-2-123456789012"
}
```

2. 將您的應用程式原始碼套件上傳至 Amazon S3。

```
$ aws s3 cp sourcebundle.zip s3://elasticbeanstalk-us-west-2-123456789012/my-app/
sourcebundle.zip
```

3. 建立應用程式版本。

```
$ aws elasticbeanstalk create-application-version --application-name my-app --
version-label v1 --description MyAppv1 --source-bundle S3Bucket="elasticbeanstalk-
us-west-2-123456789012",S3Key="my-app/sourcebundle.zip" --auto-create-application
```

4. 建立環境。

```
$ aws elasticbeanstalk create-environment --application-name my-app --environment-
name my-env --version-label v1 --solution-stack-name "64bit Amazon Linux 2015.03
 v2.0.0 running Tomcat 8 Java 8"
```

使用已儲存組態

若要在建立期間將已儲存組態套用至環境，請使用 `--template-name` 參數。

```
$ aws elasticbeanstalk create-environment --application-name my-app --environment-name
 my-env --template-name savedconfig --version-label v1
```

指定已儲存組態時，請勿同時指定解決方案堆疊名稱。已儲存組態已指定解決方案堆疊，若您嘗試同時使用這些選項，Elastic Beanstalk 將回傳錯誤。

使用命令列選項

使用 `--option-settings` 參數以 JSON 格式指定組態選項。

```
$ aws elasticbeanstalk create-environment --application-name my-app --environment-name
 my-env --version-label v1 --template-name savedconfig --option-settings '[
  {
```

```
    "Namespace": "aws:elasticbeanstalk:application",
    "OptionName": "Application Healthcheck URL",
    "Value": "/health"
  }
]
```

若要自檔案載入 JSON，請使用 `file://` 字首。

```
$ aws elasticbeanstalk create-environment --application-name my-app --environment-
name my-env --version-label v1 --template-name savedconfig --option-settings file://
healthcheckurl.json
```

Elastic Beanstalk 會將您使用 `--option-settings` 選項指定的選項設定，直接套用至環境。若已儲存組態或組態檔案已指定相同選項，`--option-settings` 會覆寫這些值。

於環境建立後設定組態選項

您可以套用已儲存組態、上傳具備組態檔案 (`.ebextensions`) 或使用 JSON 文件的新原始碼套件，藉此於執行環境修改選項設定。EB CLI 和 Elastic Beanstalk 主控台亦具備用戶端特定的功能，可設定和更新組態選項。

設定或變更組態選項時，可能會依據變更幅度而觸發全面的環境更新。例如，於 `aws:autoscaling:launchconfiguration` (如 `InstanceType`) 內變更選項時，您環境內的 Amazon EC2 執行個體會重新佈建。這會觸發滾動更新。套用其他組態變更則無須中斷或重新佈建。

您可以使用 EB CLI 或 AWS CLI 命令，從環境中移除選項設定。若移除環境 API 層級直接設定的選項，組態檔案的設定將可浮現並生效，否則會受到直接套用於環境的設定遮蓋。

已儲存的組態和組態檔案中的設定是可覆寫的，方法是透過直接在環境中使用其他組態之其中一個方法來設定相同的選項。不過，這些只能透過套用已更新的儲存組態或組態檔案完全移除。若選項未設定於已儲存組態、組態檔案，或直接設定於環境，將套用預設值 (若有)。如需詳細資訊，請參閱「優先順序」。

用戶端

- Elastic Beanstalk 主控台
- EB CLI
- AWS CLI

Elastic Beanstalk 主控台

您可以部署內含組態檔案的應用程式原始碼套件、套用已儲存組態，或於環境管理主控台的 Configuration (組態) 頁面直接修改環境，藉此更新 Elastic Beanstalk 主控台的組態選項設定。

方法

- 使用組態檔案 (.ebextensions)
- 使用已儲存組態
- 使用 Elastic Beanstalk 主控台

使用組態檔案 (`.ebextensions`)

於您的來源目錄更新組態檔案、建立新的原始碼套件並將新的版本部署至您的 Elastic Beanstalk 環境，藉此套用變更。

如需組態檔案的詳細資訊，請參閱 .Ebextensions。

若要部署原始碼套件

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

   > **Note**
   >
   > 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在環境概觀頁面上，選擇 Upload and deploy (上傳和部署)。
4. 使用畫面顯示對話方塊來上傳原始碼套件。
5. 選擇 Deploy (部署)。
6. 部署完成後，您可選擇網站的 URL 以在新分頁中開啟您的網站。

變更組態檔案不會覆寫已儲存組態的選項設定，也不會覆寫直接套用至環境 API 層級的設定。詳細資訊請參閱優先順序。

使用已儲存組態

將已儲存組態套用至執行環境，藉此套用其定義的選項設定。

將已儲存組態套用到執行的環境 (Elastic Beanstalk 主控台)

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Applications (應用程式)，然後在清單上選取應用程式名稱。

> **Note**
>
> 如果您有許多應用程式，請使用搜尋列來篩選應用程式清單。

3. 在導覽窗格中，找到應用程式名稱並選擇 Saved configurations (已儲存的配置)。
4. 選取您要套用的儲存組態，然後選擇 Load (載入)。
5. 選取環境，然後選擇 Load (載入)。

已儲存組態定義的設定會覆寫組態檔案內的設定，且可使用環境管理主控台進行設定加以覆寫。

如需建立已儲存組態的詳細資訊，請參閱 已儲存的組態。

使用 Elastic Beanstalk 主控台

Elastic Beanstalk 主控台會在 Configuration (組態) 頁面為每個環境提供許多組態選項。

變更執行環境的組態選項 (Elastic Beanstalk 主控台)

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇 Configuration (組態)。
4. 尋找您想要編輯的組態頁面：
   - 如果您看到自己感興趣的選項，或知道選項所在的組態類別，請選擇其組態類別中的 Edit (編輯)。
   - 若要尋找選項，請開啟 Table View (資料表檢視)，然後在搜尋方塊中搜尋詞彙。輸入時，清單會越變越短，且只會顯示符合搜尋詞彙的選項。

     當您看到正在尋找的選項時，請選擇包含其組態類別中的 Edit (編輯)。

5. 變更設定，然後選擇 Save (儲存)。
6. 視需要在其他組態類別重複之前兩個步驟。
7. 選擇 Apply (套用)。

對環境管理主控台中的組態選項所做的變更，會直接套用至環境。這些變更會複寫組態檔案或已儲存組態中相同選項的設定。詳細資訊請參閱優先順序。

如需有關使用 Elastic Beanstalk 主控台在執行中環境變更組態選項的詳細資訊，請參閱設定 Elastic Beanstalk 環境下的主題。

EB CLI

您可以部署內含組態檔案的原始碼、套用已儲存組態的設定，或直接使用 eb config 命令修改環境資訊，藉此透過 EB CLI 更新組態選項設定。

方法

- 使用組態檔案 (.ebextensions)
- 使用已儲存組態
- 使用 eb config
- 使用 eb setenv

使用組態檔案 (`.ebextensions`)

於您專案資料夾的 `.config` 底下納入 `.ebextensions`，與應用程式程式碼一同部署。

如需組態檔案的詳細資訊，請參閱 .Ebextensions。

```
~/workspace/my-app/
|-- .ebextensions
|   |-- environmentvariables.config
|   `-- healthcheckurl.config
|-- .elasticbeanstalk
|   `-- config.yml
|-- index.php
`-- styles.css
```

透過 eb deploy 部署您的原始碼。

```
~/workspace/my-app$ eb deploy
```

使用已儲存組態

您可以使用 eb config 命令，將已儲存組態套用至執行環境。使用 `--cfg` 選項搭配已儲存組態的名稱，將其設定套用至您的環境。

```
$ eb config --cfg v1
```

在此範例中，v1 為之前建立和儲存的組態檔案名稱。

使用此命令套用至環境的設定，會覆寫環境建立期間套用的設定，以及應用程式原始碼套件內組態檔案所定義的設定。

使用 eb config

EB CLI 的 eb config 命令可讓您使用文字編輯器，直接設定並移除環境的選項設定。

當您執行 eb config 時，EB CLI 會顯示套用至您環境的設定，其所有來源包括組態檔案、已儲存組態、建議值、直接設定於環境的選項和 API 預設值。

> **Note**
>
> eb config 不會顯示環境的屬性。若要設定可以從您的應用程式中讀取的環境屬性，請使用。eb setenv

以下範例顯示 `aws:autoscaling:launchconfiguration` 命名空間中套用的設定。這些設定包括：

- 針對 IamInstanceProfile 及 InstanceType 在環境建立期間由 EB CLI 套用的兩個建議值。
- 根據儲存庫組態建立的期間，直接在環境設定的選項 EC2KeyName。
- 其他選項適用的 API 預設值。

```
ApplicationName: tomcat
DateUpdated: 2015-09-30 22:51:07+00:00
EnvironmentName: tomcat
SolutionStackName: 64bit Amazon Linux 2015.03 v2.0.1 running Tomcat 8 Java 8
settings:
...
aws:autoscaling:launchconfiguration:
    BlockDeviceMappings: null
    EC2KeyName: my-key
    IamInstanceProfile: aws-elasticbeanstalk-ec2-role
    ImageId: ami-1f316660
    InstanceType: t2.micro
...
```

透過 eb config 設定或變更組態選項

1. 執行 eb config 以檢視您的環境資訊。

   ```
   ~/workspace/my-app/$ eb config
   ```

2. 使用預設文字編輯器變更任何設定值。

   ```
   aws:autoscaling:launchconfiguration:
       BlockDeviceMappings: null
       EC2KeyName: my-key
       IamInstanceProfile: aws-elasticbeanstalk-ec2-role
       ImageId: ami-1f316660
       InstanceType: t2.medium
   ```

3. 儲存暫時組態檔案並離開。
4. EB CLI 將更新您的環境資訊。

透過 eb config 設定的組態選項將覆寫所有其他來源的設定。

您亦可透過 eb config，將選項自您的環境移除。

移除組態選項 (EB CLI)

1. 執行 eb config 以檢視您的環境資訊。

```
~/workspace/my-app/$ eb config
```

2. 取代顯示字串 null 的任何值。您亦可將包含欲移除選項的整行程式碼刪除。

```
aws:autoscaling:launchconfiguration:
    BlockDeviceMappings: null
    EC2KeyName: my-key
    IamInstanceProfile: aws-elasticbeanstalk-ec2-role
    ImageId: ami-1f316660
    InstanceType: null
```

3. 儲存暫時組態檔案並離開。
4. EB CLI 將更新您的環境資訊。

透過 eb config 將選項自您的環境移除，將使得相同選項的設定自您應用程式原始碼套件的組態檔案浮現。詳細資訊請參閱優先順序。

使用 eb setenv

若要透過 EB CLI 設定環境屬性，請使用 eb setenv。

```
~/workspace/my-app/$ eb setenv ENVVAR=TEST
INFO: Environment update is starting.
INFO: Updating environment my-env's configuration settings.
INFO: Environment health has transitioned from Ok to Info. Command is executing on all
 instances.
INFO: Successfully deployed new configuration to environment.
```

此命令將環境屬性設定於 aws:elasticbeanstalk:application:environment 命名空間。經過短暫的更新程序後，您的應用程式即可取得透過 eb setenv 設定的環境屬性。

檢視透過 eb printenv 設定於您環境的環境屬性。

```
~/workspace/my-app/$ eb printenv
 Environment Variables:
     ENVVAR = TEST
```

AWS CLI

您可以部署內含組態檔案的原始碼套件、套用遠端存放的已儲存組態，或直接使用 `aws elasticbeanstalk update-environment` 命令修改環境，藉此透過 AWS CLI 更新組態選項設定。

方法

- 使用組態檔案 (.ebextensions)
- 使用已儲存組態
- 使用命令列選項

使用組態檔案 (**.ebextensions**)

若要透過 AWS CLI 將組態檔案套用於執行環境，請將其納入您上傳至 Amazon S3 的應用程式原始碼套件。

如需組態檔案的詳細資訊，請參閱 .Ebextensions。

```
~/workspace/my-app-v1.zip
|-- .ebextensions
|   |-- environmentvariables.config
|   `-- healthcheckurl.config
|-- index.php
`-- styles.css
```

上傳應用程式原始碼套件並套用至執行環境 (AWS CLI)

1. 若您的 Amazon S3 內還沒有 Elastic Beanstalk 儲存貯體，請使用 `create-storage-location` 建立：

   ```
   $ aws elasticbeanstalk create-storage-location
   {
       "S3Bucket": "elasticbeanstalk-us-west-2-123456789012"
   }
   ```

2. 將您的應用程式原始碼套件上傳至 Amazon S3。

   ```
   $ aws s3 cp sourcebundlev2.zip s3://elasticbeanstalk-us-west-2-123456789012/my-app/
   sourcebundlev2.zip
   ```

3. 建立應用程式版本。

```
$ aws elasticbeanstalk create-application-version --application-
name my-app --version-label v2 --description MyAppv2 --source-bundle
 S3Bucket="elasticbeanstalk-us-west-2-123456789012",S3Key="my-app/
sourcebundlev2.zip"
```

4. 更新環境。

```
$ aws elasticbeanstalk update-environment --environment-name my-env --version-label
 v2
```

使用已儲存組態

您可以透過 `--template-name` 命令的 `aws elasticbeanstalk update-environment` 選項，將已儲存組態套用至執行環境。

已儲存組態務必位於您的 Elastic Beanstalk 儲存貯體內 `resources/templates` 底下以您應用程式命名的路徑。例如，帳戶 123456789012 美國西部 (奧勒岡) 區域 (us-west-2) 中的 v1 應用程式的 `my-app` 範本位於 `s3://elasticbeanstalk-us-west-2-123456789012/resources/templates/my-app/v1`

將已儲存組態套用至執行中環境 (AWS CLI)

- 透過 `update-environment` 選項於 `--template-name` 呼叫中指定已儲存組態。

```
$ aws elasticbeanstalk update-environment --environment-name my-env --template-
name v1
```

當您使用 `aws elasticbeanstalk create-configuration-template` 建立已儲存組態，Elastic Beanstalk 將這些組態存放至此位置。您亦可將已儲存組態的存放位置修改為本機，並將這些組態自行存放於此。

使用命令列選項

透過 JSON 文件變更組態選項 (AWS CLI)

1. 於本機檔案以 JSON 格式定義您的選項設定。
2. 使用 `update-environment` 選項執行 `--option-settings`。

```
$ aws elasticbeanstalk update-environment --environment-name my-env --option-
settings file://~/ebconfigs/as-zero.json
```

在此範例中，as-zero.json 定義的選項將環境執行個體數量的上下限均設定為零。如此將停止環境中執行個體的運作，無須終止環境。

~/ebconfigs/as-zero.json

```
[
    {
        "Namespace": "aws:autoscaling:asg",
        "OptionName": "MinSize",
        "Value": "0"
    },
    {
        "Namespace": "aws:autoscaling:asg",
        "OptionName": "MaxSize",
        "Value": "0"
    },
    {
        "Namespace": "aws:autoscaling:updatepolicy:rollingupdate",
        "OptionName": "RollingUpdateEnabled",
        "Value": "false"
    }
]
```

> **Note**
>
> 透過 update-environment 設定的組態選項將覆寫所有其他來源的設定。

您亦可透過 update-environment，將選項自您的環境移除。

移除組態選項 (AWS CLI)

- 使用 update-environment 選項執行 --options-to-remove 命令。

```
$ aws elasticbeanstalk update-environment --environment-name my-env --options-to-
remove Namespace=aws:autoscaling:launchconfiguration,OptionName=InstanceType
```

透過 `update-environment` 將選項自您的環境移除，將使得相同選項的設定自您應用程式原始碼套件的組態檔案浮現。若未將選項設定為使用這些方法，將套用 API 預設值 (若有)。詳細資訊請參閱優先順序。

適用於所有環境的一般選項

命名空間

- aws:autoscaling:asg
- aws:autoscaling:launchconfiguration
- aws:autoscaling:scheduledaction
- aws:autoscaling:trigger
- aws:autoscaling:updatepolicy:rollingupdate
- aws:ec2:instances
- aws:ec2:vpc
- aws:elasticbeanstalk：應用程式
- aws:elasticbeanstalk:application:environment
- aws:elasticbeanstalk:cloudwatch:logs
- aws:elasticbeanstalk:cloudwatch:logs:health
- aws:elasticbeanstalk:command
- aws:elasticbeanstalk:environment
- aws:elasticbeanstalk:environment:process:default
- aws:elasticbeanstalk:environment:process:process_name
- aws:elasticbeanstalk:environment:proxy:staticfiles
- aws:elasticbeanstalk:healthreporting:system
- aws:elasticbeanstalk:hostmanager
- aws:elasticbeanstalk:managedactions
- aws:elasticbeanstalk:managedactions:platformupdate
- aws:elasticbeanstalk:monitoring
- aws:elasticbeanstalk:sns:topics
- aws:elasticbeanstalk:sqsd
- aws:elasticbeanstalk:trafficsplitting
- aws:elasticbeanstalk:xray

- aws:elb:healthcheck
- aws:elb:loadbalancer
- aws:elb:listener
- aws:elb:listener:listener_port
- aws:elb:policies
- aws:elb:policies:policy_name
- aws:elbv2:listener:default
- aws:elbv2:listener:listener_port
- aws:elbv2:listenerrule:rule_name
- aws:elbv2:loadbalancer
- aws:rds:dbinstance

aws:autoscaling:asg

設定您環境的 Auto Scaling 群組。如需更多詳細資訊，請參閱 the section called “Auto Scaling 群組”。

命名空間：**`aws:autoscaling:asg`**

名稱	描述	預設	有效值
Availability Zones	可用區域 (AZ) 是 AWS 區域內的不同位置，其設計目的是與其他 AZ 中的故障隔離。它們提供同一區域中其他可用區域的價廉、低延遲網路連線能力。為您的執行個體選擇可用區域數目。	Any	Any Any 1 Any 2 Any 3
Cooldown	冷卻時間有助於在之前的活動產生影響之前，避免 Amazon EC2 Auto Scaling 啟動額外擴展活動。冷卻時間是指一個擴展活動完成後、另一個擴展活動可開始前的時間量 (以秒為單位)。	360	0 設定為 10000
Custom Availability Zones	為您的執行個體定義可用區域。	無	us-east-1a us-east-1b

名稱	描述	預設	有效值

			`us-east-1c`
			
			`us-east-1d`
			
			`us-east-1e`
			
			`eu-central-1`

EnableCapacityRebalancing	指定是否為 Auto Scaling 群組中的 Spot 執行個體啟用容量重新平衡功能。如需詳細資訊，請參閱 Amazon EC2 Auto Scaling 使用者指南中的容量重新平衡。
	
	此選項只有在 `EnableSpot` 於 `aws:ec2:instances` 命名空間中設定為 `true`，且 Auto Scaling 群組中至少有一個 Spot 執行個體時有用。
		`false`	`true`
			
			`false`

MinSize	在 Auto Scaling 群組中您想要的執行個體的最小數量。	`1`	`1` 設定為 `10000`
MaxSize	在 Auto Scaling 群組中您想要的執行個體的最大數量。	`4`	`1` 設定為 `10000`

aws:autoscaling:launchconfiguration

為您的環境設定 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體。

用於您環境的執行個體是使用 Amazon EC2 啟動範本或 Auto Scaling 群組啟動組態資源建立。以下選項可與這兩種資源類型搭配使用。

如需詳細資訊，請參閱 the section called “Amazon EC2 執行個體”。您也可以在 Amazon Amazon EC2 使用者指南中的亞馬遜 EBS 章節中參考有關亞馬遜彈性區塊存放區 (EBS) 的詳細資訊。

命名空間：**aws:autoscaling:launchconfiguration**

名稱	描述	預設	有效值

DisableIMDSv1	設為 `true` 以停用執行個體中繼資料服務第 1 版 (IMDSv1)。
	
	根據平台作業系統，您環境的執行個體預設如下：
	
	• Windows 伺服器、AL2 和更早版本 – 啟用 IMDSv1 和 IMDSv2
	• AL2023 — 僅啟用 IMDSv2
	
	如需詳細資訊，請參閱設定執行個體中繼資料服務 (Amazon Linux) 或設定執行個體中繼資料服務 (Windows 伺服器)。
		`false` – 以 Windows 伺服器、Amazon Linux 2 和更早版本為基礎的平台
		
		`true` – 以 Amazon Linux 2023 為基礎的平台
			`true`
			
			`false`

EC2KeyName	您可以使用金鑰對安全地登入 EC2 執行個體。 **Note** 如果使用 Elastic Beanstalk 主控台建立環境，您無法在組態檔案中設定此選項。主控台會以建議值覆寫此選項。	無	
IamInstanceProfile	執行個體設定檔可讓 AWS Identity and Access Management (IAM) 使用者和 AWS 服務存取臨時安全登入資料以進行 AWS API 呼叫。指定執行個體設定檔的名稱或其 ARN。 範例：	無	執行個體描述檔名稱或 ARN。

名稱	描述	預設	有效值
	• `aws-elasticbeanstalk-ec2-role` • `arn:aws:iam::123456789012:instance-profile/aws-elasticbeanstalk-ec2-role` **Note** 如果使用 Elastic Beanstalk 主控台或 EB CLI 建立環境，您無法在組態檔案中設定此選項。主控台和 EB CLI 會以建議值覆寫此選項。		
ImageId	您可以透過指定您自己的自訂 AMI ID 來覆寫預設的 Amazon Machine Image (AMI)。 範例：`ami-1f316660`	無	

名稱	描述	預設	有效值
InstanceType	在 Elastic Beanstalk 環境中用來執行您應用程式的執行個體類型。 **Important** `InstanceType` 選項已淘汰。這個選項會由 `aws:ec2:instances` 命名空間中更新、功能更強的 `InstanceTypes` 選項取代。您可以使用此新選項為環境指定一個或多個執行個體類型的清單。該清單上的第一個值等於這裡所述 `aws:autoscaling:launchconfiguration` 命名空間中包含的 `InstanceType` 選項。我們建議您使用新選項來指定執行個體類型。如果已指定，則新選項會優先於舊選項。如需詳細資訊，請參閱 the section called “aws:ec2:instances 命名空間”。 可用的執行個體類型會依使用的可用區域和區域而定。如果您選擇一個子網路，包含該子網路的可用區域會決定可用的執行個體類型。 • Elastic Beanstalk 不支援 Amazon EC2 Mac 執行個體類型。 • 如需 Amazon EC2 執行個體系列和類型的詳細資訊，請參閱 Amazon EC2 使用者指南中的執行個體類型或	因帳戶和區域而異。	一種 EC2 執行個體類型。 因帳戶、區域和可用區域而異。您可以取得依據這些值篩選的 Amazon EC2 執行個體類型清單。如需詳細資訊，請參閱 Amazon EC2 使用者指南中的可用執行個體類型或 Amazon EC2 使用者指南中的可用執行個體類型。

名稱	描述	預設	有效值
	Amazon EC2 使用者指南中的執行個體類型。 • 如需跨區域可用執行個體類型的詳細資訊，請參閱 Amazon EC2 使用者指南中的可用執行個體類型或 Amazon EC2 使用者指南中的可用執行個體類型。 **Note** 如果使用 Elastic Beanstalk 主控台或 EB CLI 建立環境，您無法在組態檔案中設定此選項。主控台和 EB CLI 會以建議值覆寫此選項。		

名稱	描述	預設	有效值

LaunchTemplateTagPropagationEnabled	設定為 `true`，即可啟用對啟動範本傳播佈建至環境的特定資源的標籤。
	
	Elastic Beanstalk 只能將下列資源標籤傳播到啟動範本：
	
	• EBS 磁碟區
	• EC2 執行個體
	• EC2 網路界面
	• AWS CloudFormation 啟動定義資源的範本
	
	此條件約束的存在是因為 CloudFormation 只允許針對特定資源建立範本時進行標記 若要取得更多資訊，請參閱使AWS CloudFormation 用指南TagSpecification中的〈
	
	⚠ Important
	
	• 將現有環境的此選項值從 `false` 變更為 `true` 可能導致現有標籤發生重大變更。
	• 啟用此功能後，標籤的傳播會需要 EC2 替換，這可能會導致停機。您可以啟用滾動式更新，批次套用組態變更，從而避免在更新程序期間停機。如需詳細資訊，請參閱 組態變更。
	
	如需有關啟動範本的詳細資訊，請參閱以下內容：
		`false`	`true`
			
			`false`

名稱	描述	預設	有效值
	• Amazon EC2 Auto Scaling User Guide 中的 Launch templates。 • 《AWS CloudFormation 使用者指南》中的使用範本 • 《AWS CloudFormation 使用者指南》中的 Elastic Beanstalk 範本程式碼片段 如需有關此選項的詳細資訊，請參閱 對啟動範本的標籤傳輸。		

MonitoringInterval	您希望傳回 Amazon CloudWatch 指標的間隔 (以分鐘為單位)。	`5 minute`	`1 minute`
			
			`5 minute`

SecurityGroups	列出 Amazon EC2 安全群組並在 Auto Scaling 群組中指派到 EC2 執行個體，為執行個體定義防火牆規則。 您可以提供一串逗號分隔值，其中包含現有 Amazon EC2 安全群組的名稱或範本中建立的 AWS::EC2::SecurityGroup 資源參考。安全群組名稱區分大小寫。 如果您搭配 Elastic Beanstalk 使用 Amazon Virtual Private Cloud (Amazon VPC)，您的執行個體會在 Virtual Private Cloud (VPC) 中啟動，並指定安全群組 ID 而不是安全群組名稱。	`elasticbeanstalk-default`	

名稱	描述	預設	有效值
SSHSourceRestriction	用於鎖定 SSH 對環境的存取。例如，您可以鎖定 SSH 對 EC2 執行個體的存取，因此只有 Bastion 主機可以存取私有子網路中的執行個體。 此字串的格式如下： *protocol*, *fromPort*, *toPort*, *source_restriction* *protocol* 傳入規則的通訊協定。 *fromPort* 起始連接埠號碼。 *toPort* 結束連接埠號碼。 *source_restriction* 流量必須路由經過的 CIDR 範圍或安全群組名稱。若要從另一個帳戶 (僅限 EC2-Classic，必須位於相同區域) 指定安全群組，請在安全群組名稱之前包含帳戶 ID。使用下列格式：*other_account_id*/*security_group_name*。如果您搭配 Elastic Beanstalk 使用 Amazon Virtual Private Cloud (Amazon VPC)，您的執行個體會在 Virtual Private Cloud (VPC) 中啟動，並指定安全群組 ID 而不是安全群組名稱。	無	

名稱	描述	預設	有效值
	範例：`tcp, 22, 22, 54.240.196.185/32` 範例：`tcp, 22, 22, my-security-group` 範例 (EC2-Classic)：`tcp, 22, 22, 123456789012/their-security-group` 範例 (VPC)：`tcp, 22, 22, sg-903004f8`		

名稱	描述	預設	有效值
BlockDeviceMappings	將其他 Amazon EBS 磁碟區或執行個體存放區磁碟區連接到 Auto Scaling 群組上的所有執行個體。 映射執行個體存放區時，您只需將裝置名稱映射至磁碟區名稱。不過，我們建議在映射 Amazon EBS 磁碟區時，您應另外指定以下部分或全部欄位 (每個欄位必須以冒號分開)： • 快照 ID • 大小，以 GB 為單位 • 在終止時刪除 (`true` 或 `false`) • 儲存類型 (僅適用於 `gp3`、`gp2`、`standard`、`st1`、`sc1` 或 `io1`) • IOPS (僅適用於 `gp3` 或 `io1`) • 輸送量 (僅適用於 `gp3`) 以下範例連接三個 Amazon EBS 磁碟區、一個空白 100GB gp2 磁碟區和一個快照，一個佈建了 2000 個 IOPS 的空白 20GB io1 磁碟區，以及執行個體存放區磁碟區 `ephemeral0` 。如果該執行個體類型支援的話，則可以連接多個執行個體存放區磁碟區。 `/dev/sdj=:100:true:gp2,/dev/sdh=snap-51eef269,/dev/sdi=:20:true:io1:2000,/dev/sdb=ephemeral0`	無	• 大小 — 必須介於 500 到 16384 GiB 之間 • 輸送量 — 必須介於每秒 125 到 1000 MiB (MiB/s) 之間

名稱	描述	預設	有效值
RootVolumeType	磁碟區類型 (磁帶、一般用途 SSD 或佈建 IOPS SSD)，以用於連接到您環境的 EC2 執行個體的根 Amazon EBS 磁碟區。	依平台而有所不同。	`standard` 用於磁帶儲存。 `gp2` 或 `gp3` 用於一般用途 SSD。 `io1` 用於佈建 IOPS SSD。
RootVolumeSize	整個 GB 中根 Amazon EBS 磁碟區的儲存容量。 如果您設定 `RootVolumeType` 為佈建 IOPS SSD，則此為必要。 例如 "64"。	對磁帶儲存和一般用途 SSD 而言，每個平台都不同。 非用於佈建 IOPS SSD。	`10` 以 `16384` GB 為一般用途和佈建 IOPS SSD。 `8` 以 `1024` GB 為磁帶用途。
RootVolumeIOPS	所需的佈建 IOPS SSD 根磁碟區或一般用途 gp3 SSD 根磁碟區每秒輸入/輸出操作次數 (IOPS)。 IOPS 與磁碟區大小的最大比例為 500 比 1。例如，有 3000 IOPS 的磁碟區必須至少為 6 GiB。	無	io1 佈建 IOPS SSD 根磁碟區為 `100` 至 `20000`。 一般用途 gp3 SSD 根磁碟區為 `3000` 至 `16000`。
RootVolumeThroughput	要為連接至您環境 EC2 執行個體的 Amazon EBS 根磁碟區佈建的每秒 MiB (MiB/s) 輸送量。 **Note** 此選項僅適用於 gp3 儲存類型。	無	`125` 設定為 `1000`

aws:autoscaling:scheduledaction

為您環境的 Auto Scaling 群組設定排程動作。為每個動作的各個設定指定一個除了選項名稱、命名空間與數值的`resource_name`。如需範例，請參閱 aws:autoscaling:scheduledaction 命名空間。

命名空間：**`aws:autoscaling:scheduledaction`**

名稱	描述	預設	有效值
StartTime	如需一次性動作，選擇執行動作的日期和時間。對於重複的動作，選擇何時要啟動該動作。	無	ISO-8601 時間戳記 在所有已排程擴展動作中都是獨一無二的。
EndTime	當您希望重複排定的擴展動作在未來的日期和時間 (在 UTC/GMT 時區) 停止重複。如果未指定 EndTime，動作會根據`Recurrence` 運算式重複執行。 範例：`2015-04-28T04:07:2Z` 當排定的動作結束時，Amazon EC2 Auto Scaling 不會自動還原到之前的設定。設定第二個排定動作並視需要返回原始設定。	無	ISO-8601 時間戳記 在所有已排程擴展動作中都是獨一無二的。
MaxSize	動作執行時套用最大執行個體計數。	無	`0` 設定為 `10000`
MinSize	動作執行時套用最小執行個體計數。	無	`0` 設定為 `10000`
DesiredCapacity	為 Auto Scaling 群組設定初始所需的容量。在套用排程動作後，觸發會依它們的設定調整所需的容量。	無	`0` 設定為 `10000`
Recurrence	您想要排定之動作發生的頻率。如果您不指定週期，則只會發生一次縮放動作，如 `StartTime` 。	無	Cron 表達式。
Suspend	設定以 `true` 暫時停用重複排定的動作。	`false`	`true`

名稱	描述	預設	有效值
			`false`

aws:autoscaling:trigger

為您環境的 Auto Scaling 群組設定擴展觸發程式。

> **Note**
>
> 此命名空間中的三個選項會決定在超過其定義的限制時，觸發的指標可以維持的時間，在此時間後才啟動觸發條件。這些選項的相關如下所示：
> `BreachDuration = Period * EvaluationPeriods`
> 這些選項的預設值 (分別為 5、5 和 1) 可滿足此方程式。如果您指定不一致的數值，Elastic Beanstalk 可能會修改其中一個值，以繼續滿足方程式。

命名空間：**`aws:autoscaling:trigger`**

名稱	描述	預設	有效值
BreachDuration	所需時間 (以分鐘為單位)，指標可在叫用觸發程序前超出其定義的限制 (如 `UpperThreshold` 和 `LowerThreshold`)。	`5`	`1` 設定為 `600`
LowerBreachScaleIncrement	執行擴展活動時需移除多少個 Amazon EC2 執行個體。	`-1`	
LowerThreshold	如果違反持續時間的測量低於此數值，則叫用觸發程序。	`2000000`	`0` 設定為 `20000000`

MeasureName	用於 Auto Scaling 觸發條件的指標。
	
	Note
	`HealthyHostCount`、`UnhealthyHostCount` 和 `TargetRes`
		`NetworkOut`	`CPUUtilization`
			
			`NetworkIn`
			
			`NetworkOut`

名稱	描述	預設	有效值
	ponseTime 僅適用於具有專用負載平衡器的環境。對於使用共用負載平衡器設定的環境，這些不是有效的指標值。如需負載平衡器類型的詳細資訊，請參閱您的 Elastic Beanstalk 環境的負載平衡器。		DiskWriteOps DiskReadBytes DiskReadOps DiskWriteBytes Latency RequestCount HealthyHostCount UnhealthyHostCount TargetResponseTime
Period	指定 Amazon 測 CloudWatch 量觸發器指標的頻率。此值為兩個連續期間之間的分鐘數。	5	1 設定為 600
EvaluationPeriods	用來判斷是否發生違規的連續評估期間數。	1	1 設定為 600
Statistic	觸發條件應使用的統計數字，例如 Average。	Average	Minimum Maximum Sum Average

名稱	描述	預設	有效值
Unit	觸發程序測量單位，例如 Bytes。	Bytes	Seconds Percent Bytes Bits Count Bytes/Second Bits/Second Count/Second None
UpperBreachScaleIncrement	指定執行縮放活動時需新增多少個 Amazon EC2 執行個體。	1	
UpperThreshold	如果違反持續時間的測量高於此數值，則叫用觸發程序。	6000000	0 設定為 20000000

aws:autoscaling:updatepolicy:rollingupdate

設定輪流更新您環境的 Auto Scaling 群組。

命名空間：**aws:autoscaling:updatepolicy:rollingupdate**

名稱	描述	預設	有效值
MaxBatchSize	包含在每個輪流更新批次的執行個體數量。	Auto Scaling 群組最低大小的三分之一，四捨五入到下一個最高整數。	1 設定為 10000
MinInstancesInService	在其他執行個體終止時，必須有最少執行	Auto Scaling 群組的規模下限或小於	0 設定為 9999

名稱	描述	預設	有效值
	個體在 Auto Scaling 群組內服務中。	Auto Scaling 群組的規模上限，以較低者為準。	

名稱	描述	預設	有效值

RollingUpdateEnabled	如果 `true`，其會為環境啟用滾動更新。輪流更新有助您為 Elastic Beanstalk 軟體應用程式進行小型、頻繁的更新，也能避免應用程式的停機時間。
	
	將此值設為 true 會自動啟用`MaxBatchSize` 、`MinInstancesInService` 、和 `PauseTime` 選項。設定這些選項還會自動將 `RollingUpdateEnabled` 選項值設為 `true`。設定此選項 `false` 以停用輪流更新。
	
	Note
	如果使用 Elastic Beanstalk 主控台或 EB CLI 建立環境，您無法在組態檔案中設定此選項。主控台和 EB CLI 會以建議
		`false`	`true`
			
			`false`

名稱	描述	預設	有效值
	值覆寫此選項。		

名稱	描述	預設	有效值

RollingUpdateType	這包括三種類型：以時間為基礎的滾動更新、以運作狀態為基礎的滾動更新，以及不可變的更新。
	
	以時間為基礎的滾動更新會 PauseTime 在批次間套用。以運作狀態為基礎的輪流更新，會在移動至下一個批次前等待新的執行個體通過運作狀態檢查。不可變動的更新，即可在新的 Auto Scaling 群組中啟動完整的執行個體。
	
	Note
	如果使用 Elastic Beanstalk 主控台或 EB CLI 建立環境，您無法在組態檔案中設定此選項。主控台和 EB CLI 會以建議值覆寫此選項。
		`Time`	`Time`
			
			`Health`
			
			`Immutable`

名稱	描述	預設	有效值
PauseTime	Elastic Beanstalk 服務在完成一個批次執行個體的更新之後，以及繼續進行下一個批次之前等待的時間量 (數秒鐘、數分鐘或數小時)。	根據執行個體類型和容器的自動計算。	PT0S* (0 秒) 至PT1H (1 小時)
Timeout	取消更新之前，等待一個批次中所有執行個體通過運作狀態檢查的時間上限 (數分鐘或數小時)。	PT30M (30 分鐘)	PT5M* (5 分鐘) 至PT1H (1 小時) * ISO8601 持續時間格式：PT#H#M#S，其中每個 # 分別代表小時、分鐘，和/或秒。

aws:ec2:instances

設定您環境的執行個體，包括 Spot 選項。此命名空間與 `aws:autoscaling:launchconfiguration` 及 `aws:autoscaling:asg` 相輔相成。

如需詳細資訊，請參閱 the section called “Auto Scaling 群組”。

命名空間：**`aws:ec2:instances`**

名稱	描述	預設	有效值

EnableSpot	為您的環境啟用 Spot 執行個體請求。若為 `false`，此命名空間中的某些選項則無法生效。	`false`	`true`
			
			`false`

InstanceTypes	您要讓環境使用的執行個體類型清單，以逗號分隔 (例如 `t2.micro, t3.micro`)。	包含兩個執行個體	一至十種 EC2 執行個體類型。建議至少使用兩種。

名稱	描述	預設	有效值
	Spot 執行個體未啟用 (EnableSpot 為 false) 時，系統只會使用清單上的第一個執行個體類型。 此選項清單上的第一個執行個體類型等於 InstanceType 命名空間中 aws:autoscaling:launchconfiguration 選項的數值。我們不建議使用後一個選項，因為其已淘汰。如果您同時指定這兩者，則會使用 InstanceTypes 選項清單上的第一個執行個體類型，並忽略 InstanceType 。 可用的執行個體類型會依使用的可用區域和區域而定。如果您選擇一個子網路，包含該子網路的可用區域會決定可用的執行個體類型。 • Elastic Beanstalk 不支援 Amazon EC2 Mac 執行個體類型。 • 如需 Amazon EC2 執行個體系列和類型的詳細資訊，請參閱 Amazon EC2 使用者指南中的執行個體類型或 Amazon EC2 使用者指南中的執行個體類型。 • 如需跨區域可用執行個體類型的詳細資訊，請參閱 Amazon EC2 使用者指南中的可用執行個體類型或 Amazon EC2 使用者指南中的可用執行個體類型。	類型的清單。 因帳戶和區域而異。	因帳戶、區域和可用區域而異。您可以取得依據這些值篩選的 Amazon EC2 執行個體類型清單。如需詳細資訊，請參閱 Amazon EC2 使用者指南中的可用執行個體類型或 Amazon EC2 使用者指南中的可用執行個體類型。 執行個體類型必須全部屬於相同架構 (arm64、x86_64、i386)。 Supported Architectures 也是這個命名空間的一部分。如果您針對 Supported Architectures 提供任何值，您為 InstanceTypes 輸入的值必須屬於您為 Supported Architectures 提供的任何一個架構，而且只能屬於這個架構。

名稱	描述	預設	有效值
	Note 某些較舊的 AWS 帳戶可能會提供 Elastic Beanstalk 以及不支援競價型執行個體 (例如 t1.micro) 的預設執行個體類型。如果您啟用 Spot 執行個體請求，並收到執行個體類型不支援 Spot 的錯誤訊息，請務必設定可支援 Spot 的執行個體類型。如要選擇 Spot 執行個體類型，請使用 Spot Instance Advisor。 當您更新環境資訊並從 `InstanceTypes` 選項中移除一或多個執行個體類型時，Elastic Beanstalk 會終止在已移除的執行個體類型上執行的任何 Amazon EC2 執行個體。然後，您環境的 Auto Scaling 群組會視需要啟動新執行個體，以使用目前指定的執行個體類型來完成所需的容量。		
SpotFleet OnDemandBase	隨著環境擴展，Auto Scaling 群組在考量 Spot 執行個體前佈建的最小隨需執行個體數量。 此選項只有在 `EnableSpot` 為 `true` 時有用。	`0`	`0` 命名空間中的 `MaxSize` 至 `aws:autoscaling:asg` 選項

名稱	描述	預設	有效值
SpotFleetOnDemandAboveBasePercentage	隨需執行個體百分比 (Auto Scaling 群組超出 SpotOnDemandBase 個執行個體數量所佈建的一部分額外容量)。 此選項只有在 EnableSpot 為 true 時有用。	0：適用於單一執行個體環境 70：適用於負載平衡的環境	0 設定為 100
SpotMaxPrice	您願意為 Spot 執行個體支付的每單位小時最高價格 (USD)。如需競價型執行個體最高價格選項的建議，請參閱 Amazon EC2 使用者指南中的競價型執行個體定價歷史記錄。 此選項只有在 EnableSpot 為 true 時有用。	每個執行個體類型的隨需價格。在此情況下，選項的數值為 null。	0.001 設定為 20.0 null

名稱	描述	預設	有效值

SupportedArchitectures	您要讓環境使用的 EC2 執行個體架構類型清單 (各項間以逗號分隔)。
	
	Elastic Beanstalk 支援以下列處理器架構為基礎的執行個體類型：
	
	• AWS 重力子 64 位元臂架構
	• 64 位元架構 (x86_64)
	• 32 位元架構 (i386)
	
	如需處理器架構和 Amazon EC2 執行個體類型的詳細資訊，請參閱the section called “Amazon EC2 執行個體類型”。
		無	`arm64`
			
			`x86_64`
			
			`i386`
			
			Note
			大多數 Elastic Beanstalk 平台皆不支援 32 位元架構 `i386`。建議您改為選擇 `x86_64` 或 `arm64` 架構類型。

aws:ec2:vpc

設定您的環境以在自訂的 Amazon Virtual Private Cloud (Amazon VPC) 中啟動資源。如果您不在此命名空間中設定，Elastic Beanstalk 會在預設 VPC 中啟動資源。

命名空間：**`aws:ec2:vpc`**

名稱	描述	預設	有效值
VPCId	您的 Amazon VPC 的 ID。	無	
Subnets	Auto Scaling 群組子網路或子網路的 ID。如果您有多個子網路，指定一個值作為子網路 ID 的一個單一逗號分隔字串 (例如`"subnet-11111111,subnet-22222222"`)。	無	
ELBSubnets	Elastic Load Balancer 的子網路或子網路 ID。如果您有多個子網路，指定一個值作為子網路 ID 的一個	無	

名稱	描述	預設	有效值
	單一逗號分隔字串 (例如"subnet-11111111,subnet-22222222")。		
ELBScheme	如果您想要在 Amazon VPC 中建立一個內部負載平衡器，以讓您的 Elastic Beanstalk 應用程式無法從 Amazon VPC 外部存取，則請指定 internal。如果您指定 public 或 internal 以外的數值，Elastic Beanstalk 會忽略該數值。	public	public internal
DBSubnets	包含資料庫子網路的 ID。這僅適用於當您希望新增 Amazon RDS 資料庫執行個體做為您應用程式的一部分時。如果您有多個子網路，指定一個值作為子網路 ID 的一個單一逗號分隔字串 (例如"subnet-11111111,subnet-22222222")。	無	
AssociatePublicIpAddress	指定是否要使用公有 IP 地址來啟動 Amazon VPC 中的執行個體。公有 IP 地址執行個體不需藉由 NAT 裝置與網際網路通訊。如果要在單一公有子網路中納入您的負載平衡器與執行個體，您必須將值設為 true。 此選項並不會影響單一執行個體環境，該環境一律都有具備彈性 IP 地址的單一 Amazon EC2 執行個體。此選項與有負載平衡且可擴展的環境相關。	無	true false

aws:elasticbeanstalk：應用程式

為您的應用程式設定運作狀態檢查路徑。如需詳細資訊，請參閱 基礎型運作狀態報告。

命名空間：**aws:elasticbeanstalk:application**

名稱	描述	預設	有效值
應用程式狀況檢查 URL	傳送運作狀態檢查請求的路徑。如果未設定此路徑，負載平衡器會嘗試在連接埠 80 上建立 TCP 連線，以確認應用程式的運作狀態。設定開始為 / 的路徑，以向該路徑傳送一個 HTTP GET	無	有效值包含： / (HTTP GET 到根路徑) */health*

名稱	描述	預設	有效值

	請求。您也可以在路徑前先包含通訊協定 (HTTP、HTTPS、TCP 或 SSL) 和連接埠，以檢查 HTTPS 連線或使用非預設的連接埠。
	
	Note
	如果使用 Elastic Beanstalk 主控台建立環境，您無法在組態檔案中設定此選項。主控台會以建議值覆寫此選項。
			`HTTPS:443/`
			
			`HTTPS:443/ health`

EB CLI 和 Elastic Beanstalk 主控台會為前述選項套用建議的數值。若您想要使用組態檔進行相同的設定，您必須移除這些設定。如需詳細資訊，請參閱「建議值」。

aws:elasticbeanstalk:application:environment

為您的應用程式設定環境屬性。

命名空間：**`aws:elasticbeanstalk:application:environment`**

名稱	描述	預設	有效值
任何環境變數名稱。	以金鑰數值組通過。	無	任何環境變數值。

如需詳細資訊，請參閱環境屬性與其他軟體設定。

aws:elasticbeanstalk:cloudwatch:logs

為您的應用程式設定執行個體日誌串流。

命名空間：**aws:elasticbeanstalk:cloudwatch:logs**

名稱	描述	預設	有效值
StreamLogs	指定是否要在 Proxy 和部署 CloudWatch 記錄檔的記錄檔中建立群組，以及是否要從環境中的每個執行個體建立串流記錄。	false	true false
DeleteOnTerminate	指定是否在環境終止時刪除日誌群組。若為 false，日誌將保留 RetentionInDays 日。	false	true false
RetentionInDays	在保留日誌事件過期之前的天數。	7	1、3、5、7、14、30、60、90、120、150、180、365、400、545、731、1827、3653

aws:elasticbeanstalk:cloudwatch:logs:health

為您的應用程式設定環境運作狀態日誌串流。

命名空間：**aws:elasticbeanstalk:cloudwatch:logs:health**

名稱	描述	預設	有效值
HealthStreamingEnabled	對於已啟用增強健全狀況報告的環境，請指定是否要在環境健全狀況的 CloudWatch 記錄中建立群組，並封存 Elastic Beanstalk 環境健全狀況資料。如需啟動增強運作狀態的相關資訊，請參閱 aws:elasticbeanstalk:healthreporting:system 。	false	true false
DeleteOnTerminate	指定是否在環境終止時刪除日誌群組。若為 false，運作狀態資料會保留 RetentionInDays 天。	false	true false
RetentionInDays	在過期前保留已封存運作資料的天數。	7	1、3、5、7、14、30、60、9

名稱	描述	預設	有效值
			0、120、150、180、365、400、545、731、1827、3653

aws:elasticbeanstalk:command

設定應用程式程式碼的部署原則。如需詳細資訊，請參閱 the section called “部署選項”。

命名空間：**aws:elasticbeanstalk:command**

名稱	描述	預設	有效值

DeploymentPolicy	為應用程式版本部署選擇 部署原則。
	
	Note
	如果使用 Elastic Beanstalk 主控台建立環境，您無法在組態檔案中設定此選項。主控台會以建議值覆寫此選項。
		`AllAtOnce`	`AllAtOnce`
			
			`Rolling`
			
			`RollingWithAdditionalBatch`
			
			`Immutable`
			
			`TrafficSplitting`

Timeout	等待執行個體完成執行命令的時間 (以秒為單位)。 Elastic Beanstalk 會在內部對 `Timeout` 值增加 240 秒 (四分鐘)。例如，預設的有效逾時是 840 秒 (600 + 240) 或 14 分鐘。	`600`	1 設定為 `3600`
BatchSizeType	在中指定的數字類型BatchSize。	`Percentage`	`Percentage`

名稱	描述	預設	有效值
	Note 如果使用 Elastic Beanstalk 主控台或 EB CLI 建立環境，您無法在組態檔案中設定此選項。主控台和 EB CLI 會以建議值覆寫此選項。		`Fixed`
BatchSize	Auto Scaling 群組中可同時執行部署的 Amazon EC2 執行個體百分比或固定數目。根據所使用的BatchSizeType設定，有效值會有所不同。 **Note** 如果使用 Elastic Beanstalk 主控台或 EB CLI 建立環境，您無法在組態檔案中設定此選項。主控台和 EB CLI 會以建議值覆寫此選項。	`100`	1 至 `100` (`Percentage`)。 1到 aws:自動縮放:ASG:: () MaxSize `Fixed`
IgnoreHealthCheck	不要因為失敗的運作狀態檢查而取消部署。	`false`	`true` `false`

aws:elasticbeanstalk:environment

設定您的環境架構和服務角色。

命名空間：**`aws:elasticbeanstalk:environment`**

名稱	描述	預設	有效值
EnvironmentType	設定 `SingleInstance` 以在不使用負載平衡器的情況下啟動一個 EC2 執行個體。	`LoadBalanced`	`SingleInstance`

名稱	描述	預設	有效值
			`LoadBalanced`
ServiceRole	Elastic Beanstalk 用來為環境管理資源的 IAM 角色名稱。指定角色名稱 (選擇性地在字首加上自訂路徑) 或其 ARN。 範例： • `aws-elasticbeanstalk-service-role` • *`custom-path /custom-role`* • `arn:aws:iam::123456789012:role/aws-elasticbeanstalk-service-role` **Note** 如果使用 Elastic Beanstalk 主控台或 EB CLI 建立環境，您無法在組態檔案中設定此選項。主控台和 EB CLI 會以建議值覆寫此選項。	無	IAM 角色名稱、路徑/名稱或 ARN

LoadBalancerType	您環境的負載平衡器類型。如需詳細資訊，請參閱 the section called “負載平衡器”。	`classic`	`classic`
			
			`application`
			
			`network`

名稱	描述	預設	有效值

LoadBalancerIsShared	指定環境的負載平衡器是專用還是共用。此選項僅能針對 Application Load Balancer 設定。其無法在建立環境後更改。
	
	`false` 時，環境有自己的專用負載平衡器，由 Elastic Beanstalk 建立和管理。使用 `true` 時，環境會使用由您建立的共享負載平衡器，並在 aws:elbv2:loadbalancer 命名空間的 `SharedLoadBalancer` 選項中指定。
		`false`	`true`
			
			`false`

aws:elasticbeanstalk:environment:process:default

設定您環境的預設程序。

命名空間：**`aws:elasticbeanstalk:environment:process:default`**

名稱	描述	預設	有效值
DeregistrationDelay	取消註冊之前等待作用中請求完成的時間 (以秒為單位)。	`20`	`0` 設定為 `3600`
HealthCheckInterval	Elastic Load Balancing 會檢查應用程式 Amazon EC2 執行個體之運作狀態的時間間隔 (以秒為單位)。	使用傳統或 Application Load Balancer：`15` 使用 Network Load Balancer：`30`	使用傳統或 Application Load Balancer：`5`到`300` 使用 Network Load Balancer：`10`、`30`
HealthCheckPath	運作狀態檢查 HTTP 請求傳送的路徑。	`/`	可路由路徑。
HealthCheckTimeout	運作狀態檢查期間等待回應的時間，以秒為單位。	`5`	`1` 設定為 `60`

名稱	描述	預設	有效值
	此選項只適用於 Application Load Balancer 的環境。		
HealthyThresholdCount	在 Elastic Load Balancing 變更執行個體運作狀態之前的連續成功請求數目。	使用傳統或 Application Load Balancer：3 使用 Network Load Balancer：5	2 設定為 10
MatcherHTTPCode	以逗號分隔的 HTTP 程式碼清單，表示執行個體正常運作。 此選項只適用於具有 Network Load Balancer 或 Application Load Balancer 的環境。	200	使用 Application Load Balancer：200 至 499 使用 Network Load Balancer：200 至 399
Port	接聽處理程序的連接埠。	80	1 設定為 65535
Protocol	處理程序使用的通訊協定。 透過 Application Load Balancer，您僅能將此選項設為HTTP或HTTPS。 透過 Network Load Balancer，您僅能將此選項設為TCP。	使用傳統或 Application Load Balancer：HTTP 使用 Network Load Balancer：TCP	TCP HTTP HTTPS

名稱	描述	預設	有效值

StickinessEnabled	設定為 true，以啟用黏性工作階段。
	
	此選項只適用於 Application Load Balancer 的環境。
		`'false'`	`'false'`
			
			`'true'`

StickinessLBCookieDuration	黏性工作階段 Cookie 的生命週期 (以秒為單位)。 此選項只適用於 Application Load Balancer 的環境。	`86400` (一天)	`1` 設定為 `604800`
StickinessType	設定為 `lb_cookie` 以為黏性工作階段使用 cookies。 此選項只適用於 Application Load Balancer 的環境。	`lb_cookie`	`lb_cookie`
UnhealthyThresholdCount	在 Elastic Load Balancing 變更執行個體運作狀態之前的連續不成功請求數目。	`5`	`2` 設定為 `10`

aws:elasticbeanstalk:environment:process:process_name

為您的環境設定額外的程序。

命名空間：**aws:elasticbeanstalk:environment:process:*process_name***

名稱	描述	預設	有效值
DeregistrationDelay	取消註冊之前等待作用中請求完成的時間 (以秒為單位)。	20	0 設定為 3600
HealthCheckInterval	Elastic Load Balancing 會檢查應用程式的 Amazon EC2 執行個體運作狀態的間隔 (以秒為單位)。	使用傳統或 Application Load Balancer：15 使用 Network Load Balancer：30	使用傳統或 Application Load Balancer：5到300 使用 Network Load Balancer：10、30
HealthCheckPath	運作狀態檢查 HTTP 請求傳送的路徑。	/	可路由路徑。
HealthCheckTimeout	運作狀態檢查期間等待回應的時間，以秒為單位。 此選項只適用於 Application Load Balancer 的環境。	5	1 設定為 60
HealthyThresholdCount	在 Elastic Load Balancing 變更執行個體運作狀態之前的連續成功請求數目。	使用傳統或 Application Load Balancer：3 使用 Network Load Balancer：5	2 設定為 10
MatcherHTTPCode	以逗號分隔的 HTTP 程式碼清單，表示執行個體正常運作。 此選項只適用於具有 Network Load Balancer 或 Applicati	200	使用 Application Load Balancer：200 至 499 使用 Network Load Balancer：200 至 399

名稱	描述	預設	有效值
	on Load Balancer 的環境。		
Port	接聽處理程序的連接埠。	80	1 設定為 65535
Protocol	處理程序使用的通訊協定。 透過 Application Load Balancer，您僅能將此選項設為HTTP或HTTPS。 透過 Network Load Balancer，您僅能將此選項設為TCP。	使用傳統或 Application Load Balancer：HTTP 使用 Network Load Balancer：TCP	TCP HTTP HTTPS
StickinessEnabled	設定為 true，以啟用黏性工作階段。 此選項只適用於 Application Load Balancer 的環境。	'false'	'false' 'true'
StickinessLBCookieDuration	黏性工作階段 Cookie 的生命週期 (以秒為單位)。 此選項只適用於 Application Load Balancer 的環境。	86400 (一天)	1 設定為 604800

名稱	描述	預設	有效值
StickinessType	設定為 `lb_cookie` 以為黏性工作階段使用 cookies。 此選項只適用於 Application Load Balancer 的環境。	`lb_cookie`	`lb_cookie`
UnhealthyThreshold Count	在 Elastic Load Balancing 變更執行個體運作狀態之前的連續不成功請求數目。	5	2 設定為 `10`

aws:elasticbeanstalk:environment:proxy:staticfiles

您可以使用以下命名空間，設定代理伺服器以提供靜態檔案。代理伺服器收到位於指定路徑下的檔案請求時，會直接提供檔案而非將請求路由至您的應用程式。這可減少應用程式必須處理的請求數量。

將代理伺服器提供的路徑對應至包含靜態資產之原始程式碼中的資料夾。您在此命名空間中定義的每個選項都會對應至不同路徑。

> **Note**
>
> 此命名空間適用於以 Amazon Linux 2 和更新版本為基礎的平台分支。如果您的環境會使用以 Amazon Linux AMI 為基礎的平台版本 (Amazon Linux 2 之前的版本)，請參閱 the section called “平台特定選項” 以取得平台特定的靜態檔案命名空間。

命名空間：**`aws:elasticbeanstalk:environment:proxy:staticfiles`**

名稱	Value
代理伺服器提供檔案的路徑。以 `/` 啟動值。	內含檔案的資料夾名稱。 例如，指定 `staticimages` 以從原始碼套件最上層名為 `staticimages` 的資料夾提供檔案。

名稱	Value
例如，指定 `/images` 以在 *`subdomain`* `.elasticbeanstalk.com/images` 提供檔案。	

aws:elasticbeanstalk:healthreporting:system

為您的環境設定增強的執行狀況報告。

命名空間：**`aws:elasticbeanstalk:healthreporting:system`**

名稱	描述	預設	有效值

SystemType	運作狀態報告系統 (基本或增強)。增強式運作狀態報告需要一個服務角色和第 2 版或更新版本的平台版本。
	
	Note
	如果使用 Elastic Beanstalk 主控台或 EB CLI 建立環境，您無法在組態檔案中設定此選項。主控台和 EB CLI 會以建議值覆寫此選項。
		`basic`	`basic`
			
			`enhanced`

ConfigDocument	說明要發佈到的環境和執行個體指標的 JSON 文件 CloudWatch。	無	

EnhancedHealthAuthEnabled	啟用內部 API 的授權，Elastic Beanstalk 會將其用於將增強型運作狀態資訊從您的環境執行個體傳送至 Elastic Beanstalk 服務。
	
	如需詳細資訊，請參閱 the section called “增強型運作狀態角色”。
		`true`	`true`
			
			`false`

名稱	描述	預設	有效值
	Note 此選項僅適用於增強型運作狀態報告 (例如 SystemType 設為 enhanced 時)。		
HealthCheckSuccessThreshold	為執行個體降低閾值以通過運作狀態檢查。 Note 如果使用 Elastic Beanstalk 主控台建立環境，您無法在組態檔案中設定此選項。主控台會以建議值覆寫此選項。	Ok	Ok Warning Degraded Severe

aws:elasticbeanstalk:hostmanager

在您的環境設定 EC2 執行個體以上傳 Amazon S3 輪換日誌。

命名空間：**aws:elasticbeanstalk:hostmanager**

名稱	描述	預設	有效值
LogPublicationControl	將應用程式的 Amazon EC2 執行個體日誌檔案複製到與您的應用程式相關聯的 Amazon S3 儲存貯體。	false	true false

aws:elasticbeanstalk:managedactions

為您的環境設定受管平台更新。

命名空間：**aws:elasticbeanstalk:managedactions**

名稱	描述	預設	有效值
ManagedActionsEnabled	啟用受管平台更新。	false	true false

名稱	描述	預設	有效值
	當您將此設定為 `true`，您還必須指定 `PreferredStartTime` 和 `UpdateLevel`。		
PreferredStartTime	在 UTC 中設定受管動作的維護時段。 例如 `"Tue:09:00"`。	無	日期和時間格式： *#*：*##*：*##* 格式。
ServiceRoleForManagedUpdates	Elastic Beanstalk 用來執行您環境的受管平台更新的 IAM 角色名稱。 您可以使用與針對 `ServiceRole` 命名空間之 `aws:elasticbeanstalk:environment` 選項所指定相同的角色，或使用您帳戶的受管更新服務連結角色。在後者的情況中，如果帳戶還沒有受管更新服務連結角色，則 Elastic Beanstalk 會建立。	無	與 `ServiceRole` 相同 或 `AWSServiceRoleForElasticBeanstalkManagedUpdates`

aws:elasticbeanstalk:managedactions:platformupdate

為您的環境設定受管平台更新。

命名空間：**`aws:elasticbeanstalk:managedactions:platformupdate`**

名稱	描述	預設	有效值
UpdateLevel	使用受管平台更新套用最高層級的更新。平台的版本命名方式為 *major*.*minor*.*patch*。例	無	僅 `patch`適用於修補程式版本更新。

名稱	描述	預設	有效值
	如，2.0.8 有 2 主要版本、無次要版本，和 8 修補程式版本。		`minor` 次要和修補程式版本更新。

InstanceRefreshEnabled	啟用每週執行個體替換。
	
	這需要將 `ManagedActionsEnabled` 設定為 `true`。
		`false`	`true`
			
			`false`

aws:elasticbeanstalk:monitoring

設定您的環境以終止運作狀態檢查失敗的 EC2 執行個體。

命名空間：**`aws:elasticbeanstalk:monitoring`**

名稱	描述	預設	有效值

Automatically Terminate Unhealthy Instances	如果執行個體的運作狀態檢查失敗即終止該執行個體。
	
	Note
	只有舊式環境支援此選項。它根據能否達到以及以其他執行個體為基礎的指標，來判斷執行個體的運作狀態。Elastic Beanstalk 並未提供自動根據應用程式運作狀態來終止執行個體的功能。
		`true`	`true`
			
			`false`

aws:elasticbeanstalk:sns:topics

為您的環境設定通知。

命名空間：**`aws:elasticbeanstalk:sns:topics`**

名稱	描述	預設	有效值
Notification Endpoint	通知影響您應用程式的重要事件端點。 **Note** 如果使用 Elastic Beanstalk 主控台建立環境，您無法在組態檔案中設定此選項。主控台會以建議值覆寫此選項。	無	

Notification Protocol	用於傳送通知給端點的通訊協定。	`email`	`http`
			
			`https`
			
			`email`
			
			`email-json`
			
			`sqs`

Notification Topic ARN	您訂閱主題的 Amazon Resource Name (ARN)。	無	
Notification Topic Name	您訂閱的主題名稱。	無	

aws:elasticbeanstalk:sqsd

為工作者環境設定 Amazon SQS 佇列。

命名空間：**aws:elasticbeanstalk:sqsd**

名稱	描述	預設	有效值
WorkerQueueURL	在工作者環境方案讀取訊息的協助程式佇列 URL。 **Note** 當您未指定值的時候，Elastic Beanstalk 自動建立的佇列將為標準 Amazon SQS 佇列。當您提供值的時候，您可以提供標準或 FIFO Amazon SQS 佇列的 URL。請注意，如果您提供了 FIFO 佇列，則不支援週期性任務。	自動產生	如果您不指定數值，則 Elastic Beanstalk 會自動建立佇列。
HttpPath	HTTP POST 訊息傳送到的應用程式相對路徑。	/	

MimeType	在 HTTP POST 請求中所傳送訊息的 MIME 類型。	`application/json`	`application/json`
			
			`application/x-www-form-urlencoded`
			
			`application/xml`
			
			`text/plain`
			
			自訂 MIME 類型。

HttpConnections	並行連線到任何 Amazon EC2 執行個體內應用程式的最大上限數。	`50`	1 設定為 100

名稱	描述	預設	有效值
	Note 如果使用 Elastic Beanstalk 主控台建立環境，您無法在組態檔案中設定此選項。主控台會以建議值覆寫此選項。		
ConnectTimeout	等待成功連線到應用程式所需的時間 (以秒為單位)。	5	1 設定為 60
InactivityTimeout	等待現有連線回應到應用程式所需的時間 (以秒為單位)。訊息會重新處理，直到協助程式從工作者環境方案中的應用程式接收到 200 (OK) 回應或 `Retention Period` 過期。	299	1 設定為 36000
VisibilityTimeout	從 Amazon SQS 佇列內送的訊息所需的時間 (秒) 是被鎖定以進行處理的。在設定的時間數過後，該則訊息才能在佇列中被任何其他協助程式再次讀取。	300	0 設定為 43200
ErrorVisibilityTimeout	在處理嘗試因明確錯誤而失敗之後，Elastic Beanstalk 傳回訊息給 Amazon SQS 佇列之前經過的時間 (秒)。	2 秒	0到43200秒
Retention Period	訊息有效並主動處理所需的時間 (以秒為單位)。	345600	60 設定為 1209600
MaxRetries	Elastic Beanstalk 嘗試傳送訊息到 Web 應用程式的次數上限，將在該訊息移動到無效字母佇列之前處裡。	10	1 設定為 100

aws:elasticbeanstalk:trafficsplitting

設定您環境的流量分割部署。

當您將 aws:elasticbeanstalk:command 命名空間的 DeploymentPolicy 選項設為 TrafficSplitting 時，便會套用此命名空間。如需部署原則的詳細資訊，請參閱 the section called "部署選項"。

命名空間：**aws:elasticbeanstalk:trafficsplitting**

名稱	描述	預設	有效值
NewVersionPercent	Elastic Beanstalk 移動到執行您正在部署新應用程式版本環境用戶端的傳入用戶端流量初始百分比。	10	1 設定為 100
EvaluationTime	時間間隔 (分鐘)，Elastic Beanstalk 會在初始運作狀態良好的部署之後等待這段時間，再繼續將所有傳入用戶端流量移動到您正在部署的新應用程式版本。	5	3 設定為 600

aws:elasticbeanstalk:xray

執行 AWS X-Ray 協助程式，以轉送 X-Ray 整合應用程式的追蹤資訊。

命名空間：**aws:elasticbeanstalk:xray**

名稱	描述	預設	有效值
XRayEnabled	設定為 true 以在您環境的執行個體執行 X-Ray 協助程式。	false	true false

aws:elb:healthcheck

為 Classic Load Balancer 設定運作狀態檢查。

命名空間：**aws:elb:healthcheck**

名稱	描述	預設	有效值
HealthyThreshold	在 Elastic Load Balancing 變更執行個體運作狀態之前的連續成功請求數目。	3	2 設定為 10
Interval	Elastic Load Balancing 檢查應用程式 Amazon EC2 執行個體運作狀態的間隔。	10	5 設定為 300
Timeout	Elastic Load Balancing 在將執行個體視為沒有回應之前等待回應的時間 (以秒為單位)。	5	2 設定為 60
UnhealthyThreshold	在 Elastic Load Balancing 變更執行個體運作狀態之前的連續不成功請求數目。	5	2 設定為 10
(已廢除) Target	運作狀態檢查傳送至後端執行個體的目的地。使用 Application Healthcheck URL 在 aws:elasticbeanstalk:application 命名空間。	TCP:80	格式內目標 *PROTOCOL*：*PORT* */PATH*

aws:elb:loadbalancer

設定您環境的 Classic Load Balancer。

在這個命名空間的數個選項已不再受到支援，以便使用 aws:elb:listener 命名空間中的接聽程式特定選項。若使用這些不再支援的選項，您只能在標準連接埠上設定兩個接聽程式 (一個安全，一個不安全)。

命名空間：**aws:elb:loadbalancer**

名稱	描述	預設	有效值
CrossZone	在所有可用區域的所有執行個體平均設定負載平衡器至路由流量，而不是只在每個區域內設定。	false	true false

名稱	描述	預設	有效值
	Note 如果使用 Elastic Beanstalk 主控台或 EB CLI 建立環境，您無法在組態檔案中設定此選項。主控台和 EB CLI 會以建議值覆寫此選項。		
SecurityGroups	指派一或多個安全群組，讓您建立負載平衡器。	無	一或多個安全群組 ID。
ManagedSecurityGroup	為您環境的負載平衡器指派現有的安全群組，而不用建立新的。若要使用此設定，請更新此命名空間中的 `SecurityGroups` 設定以包含安全群組的 ID，並移除自動建立之安全群組的 ID (如果已建立)。 若要允許流量從負載平衡器到您環境的 EC2 執行個體，Elastic Beanstalk 會將規則新增至執行個體的安全群組，允許從受管安全群組的傳入流量。	無	一個安全群組 ID。

(已廢除) LoadBalancerHTTPPort	不安全接聽程式的接聽連接埠。	`80`	`OFF`
			
			`80`

(已廢除) LoadBalancerPortProtocol	用於不安全接聽程式的通訊協定。	`HTTP`	`HTTP`
			
			`TCP`

(已廢除) LoadBalancerHTTPSPort	安全接聽程式的接聽連接埠。	`OFF`	`OFF`
			
			`443`
			
			`8443`

名稱	描述	預設	有效值
(已廢除) LoadBalancerSSLPortProtocol	用於安全接聽程式的通訊協定。	HTTPS	HTTPS SSL
(已廢除) SSLCertificateId	要繫結至安全接聽程式的 SSL 憑證的 Amazon Resource Name (ARN)。	無	

aws:elb:listener

在 Classic Load Balancer 上設定預設接聽程式 (連接埠 80)。

命名空間：**aws:elb:listener**

名稱	描述	預設	有效值
ListenerProtocol	接聽程式使用的通訊協定。	HTTP	HTTP TCP
InstancePort	此接聽程式與 EC2 執行個體溝通所使用的連接埠。	80	1 設定為 65535
InstanceProtocol	此接聽程式與 EC2 執行個體溝通所使用的通訊協定。 需與 ListenerProtocol 有相同的網際網路通訊協定。也必須與任何其他接聽程式有相同的安全性等級，使用 InstancePort 作為此接聽程式。 例如，如果 ListenerProtocol 是 HTTPS (應用程式層級，使用安全連線)，您可以將 InstanceProtocol 設為 HTTP (也在應用程式層級，使用不安全的連線)。此外，若您將 InstancePort 設為 80，必須在所有 InstanceProtocol 設為 HTTP 的其他接聽程式中將 InstancePort 設為 80。	HTTP，當 ListenerProtocol 是 HTTP 時 TCP，當 ListenerProtocol 是 TCP 時	HTTP 或 HTTPS，當 ListenerProtocol 是 HTTP 或 HTTPS 時 TCP 或 SSL，當 ListenerProtocol 是 TCP 或 SSL 時

名稱	描述	預設	有效值
PolicyNames	套用至此接聽程式連接埠，原則名稱以逗號分隔的清單。我們建議您改用aws:elb:policies命名空間的 LoadBalancerPorts 選項。	無	

ListenerEnabled	指定這個接聽程式是否已啟用。若您指定 `false`，則接聽程式不會包含在負載平衡器中。	`true`	`true`
			
			`false`

aws:elb:listener:listener_port

Classic Load Balancer 上設定額外的接聽程式。

命名空間：**`aws:elb:listener:`*`listener_port`***

名稱	描述	預設	有效值

ListenerProtocol	接聽程式使用的通訊協定。	`HTTP`	`HTTP`
			`HTTPS` `TCP`
			`SSL`

InstancePort	此接聽程式與 EC2 執行個體溝通所使用的連接埠。	與*`listener_port`* 相同。	`1` 設定為 `65535`
InstanceProtocol	此接聽程式與 EC2 執行個體溝通所使用的通訊協定。 需與 `ListenerProtocol` 有相同的網際網路通訊協定。也必須與任何其他接聽程式有相同的安全性等級，使用 `InstancePort` 作為此接聽程式。 例如，如果 `ListenerProtocol` 是 HTTPS (應用程式層級，使用安全連線)，您可以將 `InstanceProtocol` 設為 HTTP (也在應用程式層級，使用不安全的連線)。此	`HTTP`，當 `ListenerProtocol` 是 HTTP 或 HTTPS 時 `TCP`，當 `ListenerProtocol`	`HTTP` 或 `HTTPS`，當 `ListenerProtocol` 是 HTTP 或 HTTPS 時 `TCP` 或 `SSL`，當 `ListenerProtocol`

名稱	描述	預設	有效值
	外，若您將 InstancePort 設為 80，必須在所有 InstanceProtocol 設為 HTTP 的其他接聽程式中將 InstancePort 設為 80。	是 TCP 或 SSL 時	是 TCP 或 SSL 時
PolicyNames	套用至此接聽程式連接埠，原則名稱以逗號分隔的清單。我們建議您改用aws:elb:policies命名空間的 LoadBalancerPorts 選項。	無	
SSLCertificateId	要繫結至接聽程式的 SSL 憑證的 Amazon Resource Name (ARN)。	無	
ListenerEnabled	指定這個接聽程式是否已啟用。若您指定 false，則接聽程式不會包含在負載平衡器中。	若有設定任何其他選項即為 true，否則為 false。	true false

aws:elb:policies

為 Classic Load Balancer 修改預設黏著性和全球負載平衡器原則。

命名空間：**aws:elb:policies**

名稱	描述	預設	有效值
ConnectionDraining Enabled	指定負載平衡器維持現有連結至已變為狀態不佳或取消註冊的執行個體，以完成正在進行的請求。 **Note** 如果使用 Elastic Beanstalk 主控台或 EB CLI 建立環境，您無法在組	false	true false

名稱	描述	預設	有效值
	態檔案中設定此選項。主控台和 EB CLI 會以建議值覆寫此選項。		
ConnectionDrainingTimeout	在強制關閉連接之前，負載平衡器在連接耗盡期間維持與執行個體的現有連接最大秒數。 **Note** 如果使用 Elastic Beanstalk 主控台建立環境，您無法在組態檔案中設定此選項。主控台會以建議值覆寫此選項。	20	1 設定為 3600
ConnectionSettingIdleTimeout	負載平衡器等待透過連線傳送或接收任何資料的時間 (以秒為單位)。如果在此段時間過後沒有資料已傳送或接收，則負載平衡器關閉連線。	60	1 設定為 3600
LoadBalancerPorts	預設原則 (`AWSEB-ELB-StickinessPolicy`) 適用的接聽程式連接埠的逗號分隔清單。	無	您可以使用 `:all` 以表示所有接聽程式連接埠
Stickiness Cookie Expiration	每個 Cookie 有效所需的時間 (秒)。使用預設原則 (`AWSEB-ELB-StickinessPolicy`)。	`0`	`0` 設定為 `1000000`
Stickiness Policy	將使用者會話繫結到特定的伺服器執行個體，以便會話期間來自使用者的所有請求都發送到相同的伺服器執行個體。使用預設原則 (`AWSEB-ELB-StickinessPolicy`)。	`false`	`true false`

aws:elb:policies:policy_name

為 Classic Load Balancer 建立額外的負載平衡器原則。

命名空間：**aws:elb:policies:*policy_name***

名稱	描述	預設	有效值
CookieName	應用程式產生的 cookie 名稱，用於控制 AppCookieStickinessPolicyType 原則的工作階段生命週期。此原則僅可與 HTTP/HTTPS 接聽程式關聯。	無	
InstancePorts	此原則適用的執行個體連接埠的逗號分隔清單。	無	連接埠清單，或 :all
LoadBalancerPorts	此原則適用的接聽程式連接埠的逗號分隔清單。	無	連接埠清單，或 :all
ProxyProtocol	對於 ProxyProtocolPolicyType 原則，指定是否包含 IP 地址和連接埠的原始請求 TCP 訊息。此原則僅可與 TCP/SSL 接聽程式關聯。	無	true false
PublicKey	在身分驗證後端伺服器或伺服器時，PublicKeyPolicyType 政策所使用的公有金鑰內容。此政策無法直接套用至後端伺服器或接聽程式。它必須是 BackendServerAuthenticationPolicyType 政策的一部分。	無	
PublicKeyPolicyNames	以逗號分隔清單政策名稱 (從 PublicKeyPolicyType 政策) 的 BackendServerAuthenticationPolicyType 政策，控制身分驗證到後端伺服器或伺服器。此政策僅可與使用 HTTPS/SSL 的後端伺服器相關聯。	無	
SSLProtocols	為 SSLNegotiationPolicyType 政策啟用以逗號分隔的 SSL 通訊協定，以定	無	

名稱	描述	預設	有效值
	義負載平衡器可接受的加密方式和通訊協定。此原則僅可與 HTTPS/SSL 接聽程式關聯。		
SSLReferencePolicy	符合安全性最佳作法的預先定義 AWS 安全性原則名稱，且您想要針對定義負載平衡器所接受之加密和通訊協定的SSLNegotiationPolicyType 原則啟動。此原則僅可與 HTTPS/SSL 接聽程式關聯。	無	
Stickiness Cookie Expiration	每個 Cookie 有效所需的時間 (秒)。	`0`	`0` 設定為 `1000000`
Stickiness Policy	將使用者會話繫結到特定的伺服器執行個體，以便會話期間來自使用者的所有請求都發送到相同的伺服器執行個體。	`false`	`true false`

aws:elbv2:listener:default

在 Application Load Balancer 或 Network Load Balancer 設定預設接聽程式 (連接埠 80)。

此命名空間不適用於使用共享負載平衡器的環境。共享負載平衡器沒有預設接聽程式。

命名空間：**`aws:elbv2:listener:default`**

名稱	描述	預設	有效值
DefaultProcess	當沒有相符的規則時，轉送流量至該處理程序的名稱。	`default`	程序名稱。

ListenerEnabled	設定為`false`以停用接聽程式。您可以使用此選項，停用連接埠 80 上的預設接聽程式。	`true`	`true`
			
			`false`

名稱	描述	預設	有效值
Protocol	要處理的流量通訊協定。	使用 Application Load Balancer：HTTP 使用 Network Load Balancer：TCP	使用 Application Load Balancer：HTTP、HTTPS 使用 Network Load Balancer：TCP
Rules	套用至接聽程式的規則清單 此選項只適用於 Application Load Balancer 的環境。	無	以逗號分隔的規則名稱清單。
SSLCertificateArns	要繫結至接聽程式的 SSL 憑證的 Amazon Resource Name (ARN)。 此選項只適用於 Application Load Balancer 的環境。	無	存放在 IAM 或 ACM 憑證的 ARN。
SSLPolicy	指定安全原則以套用到接聽程式。 此選項只適用於 Application Load Balancer 的環境。	無 (ELB 預設)	負載平衡器安全原則名稱。

aws:elbv2:listener:listener_port

在 Application Load Balancer 或 Network Load Balancer 設定額外接聽程式。

> **Note**
>
> 如果是共享 Application Load Balancer，您只能指定 Rule 選項。其他選項不適用於共享負載平衡器。

命名空間：**aws:elbv2:listener:*listener_port***

名稱	描述	預設	有效值
DefaultProcess	當沒有相符的規則時流量被轉發之處理程序的名稱。	default	程序名稱。
ListenerEnabled	設定為false以停用接聽程式。您可以使用此選項，停用連接埠 80 上的預設接聽程式。	true	true false
Protocol	要處理的流量通訊協定。	使用 Application Load Balancer：HTTP 使用 Network Load Balancer：TCP	使用 Application Load Balancer：HTTP、HTTPS 使用 Network Load Balancer：TCP
Rules	套用至接聽程式的規則清單 此選項只適用於 Application Load Balancer 的環境。 如果您的環境使用共享 Application Load Balancer，而且您沒有為任何接聽程式指定此選項，則 Elastic	無	以逗號分隔的規則名稱清單。

名稱	描述	預設	有效值
	Beanstalk 會自動將 `default` 規則與連線埠 80 接聽程式產生關聯。		
SSLCertificateArns	要繫結至接聽程式的 SSL 憑證的 Amazon Resource Name (ARN)。 此選項只適用於 Application Load Balancer 的環境。	無	存放在 IAM 或 ACM 憑證的 ARN。
SSLPolicy	指定安全原則以套用到接聽程式。 此選項只適用於 Application Load Balancer 的環境。	無 (ELB 預設)	負載平衡器安全原則名稱。

aws:elbv2:listenerrule:rule_name

定義 Application Load Balancer 的接聽程式規則。如果請求符合規則中的主機名稱或路徑，負載平衡器會將請求轉發到指定的程序中。若要使用規則，將其新增到 `Rules` 命名空間中的 `aws:elbv2:listener:`*`listener_port`* 選項。

> **Note**
>
> 這個命名空間不適用於使用 Network Load Balancer 的環境。

命名空間：**aws:elbv2:listenerrule:*rule_name***

名稱	描述	預設	有效值
HostHeaders	要比對的主機名稱清單。例如，`my.example.com` 。	專用負載平衡器：無 共享負載平衡器：環境的 CNAME	每個名稱最多可包含 128 個字元。模式可包含大寫和小寫字母、數字、連字號 (–)，以及最多三個萬用字元 (* 等同於零個或多個字元，？等同於一個字元)。您可列出多個名稱，且以逗號分隔各個名稱。Application Load Balancer 最多支援五個合併的 `HostHeader` 和 `PathPattern` 規則。 如需詳細資訊，請參閱 Application Load Balancer 使用者指南中的主機條件。
PathPatterns	要匹配的路徑模式 (例如 `/img/*`)。 此選項只適用於 Application Load Balancer 的環境。	無	每個模式最多可包含 128 個字元。模式可包含大寫和小寫字母、數字、連字號 (–)，以及最多三個萬用字元 (* 等同於零個或多個字元，？等同於一個字元)。您可以新增多個以逗號分隔的路徑模式。Application Load Balancer 最多支援五個合併的 `HostHeade`

名稱	描述	預設	有效值
			r 和 PathPattern 規則。 如需詳細資訊，請參閱 Application Load Balancer 使用者指南中的路徑條件。
Priority	多個規則相符時，此規則的優先順序。數量較低的優先。不會同時有兩個規則有相同的優先順序。 使用共享負載平衡器時，Elastic Beanstalk 將規則優先順序視為共享環境中的相對優先順序，並在建立期間將其對應至絕對優先順序。	1	1 設定為 1000
Process	當此規則符合要求時，要轉送流量至該處理程序的名稱。	default	程序名稱。

aws:elbv2:loadbalancer

設定一個 Application Load Balancer。

對於共享負載平衡器，只有 SharedLoadBalancer 和 SecurityGroups 選項有效。

> **Note**
>
> 這個命名空間不適用於使用 Network Load Balancer 的環境。

命名空間：**aws:elbv2:loadbalancer**

名稱	描述	預設	有效值
AccessLogsS3Bucket	存放存取日誌的 Amazon S3 儲存貯體。儲存貯體必須與環境位於同一	無	儲存貯體名稱。

名稱	描述	預設	有效值
	個區域中，並允許負載平衡器寫入存取。		
AccessLogsS3Enabled	啟用存取日誌儲存。	`false`	`true` `false`
AccessLogsS3Prefix	用於存取日誌名稱的前綴。根據預設，負載平衡器會將記錄檔上傳到您指定值區 AWSLogs 中名稱的目錄。指定首碼以將 AWSLogs 目錄放置在另一個目錄中。	無	
IdleTimeout	在關閉與用戶端和執行個體的連線之前等待請求完成的時間 (以秒為單位)。	無	1 設定為 3600
ManagedSecurityGroup	為您環境的負載平衡器指派現有的安全群組，而不用建立新的。若要使用此設定，請在這個命名空間包含您的安全群組 ID 以更新`SecurityGroups` 設定，並移除自動建立的安全群組 ID (若有的話)。 若要允許流量從負載平衡器到您環境的 EC2 執行個體，Elastic Beanstalk 會將規則新增至執行個體的安全群組，允許從受管安全群組的傳入流量。	Elastic Beanstalks 為您的負載平衡器建立的安全群組。	一個安全群組 ID。

名稱	描述	預設	有效值
SecurityGroups	連接到負載平衡器的安全群組清單。 對於共享負載平衡器，如果您未指定此值，Elastic Beanstalk 會檢查其管理的現有安全群組是否已連接至負載平衡器。如果沒有連接到負載平衡器，Elastic Beanstalk 會建立安全群組並將其連接到負載平衡器。當共享負載平衡器的最後一個環境終止時，Elastic Beanstalk 會刪除此安全群組。 負載平衡器安全群組用於設定 Amazon EC2 執行個體安全群組輸入規則。	Elastic Beanstal 為您的負載平衡器建立的安全群組。	以逗號分隔的安全群組 ID 清單。

名稱	描述	預設	有效值
SharedLoadBalancer	共用負載平衡器的 Amazon Resource Name (ARN)。此選項僅與 Application Load Balancer 相關。當 aws:elasticbeanstalk:environment 命名空間的 `LoadBalancerIsShared` 選項設為 `true` 時，這為必要項目。建立環境後，您無法變更共用負載平衡器 ARN。 有效值的條件： • 它必須是環境所在 AWS 區域中有效的作用中負載平衡器。 • 它必須位於與環境相同的 Amazon Virtual Private Cloud (Amazon VPC)。 • 它不能是 Elastic Beanstalk 作為另一個環境的專用負載平衡器所建立的負載平衡器。您可以透過前綴 `awseb-` 來識別這些專用負載平衡器。 範例： `arn:aws:elasticloadbalancing:us-east-2:123456789012:loadbalancer/app/FrontEndLB/0dbf78d8ad96abbc`	無	符合此處所述全部準則的有效負載平衡器 ARN。

aws:rds:dbinstance

設定連接的 Amazon RDS 資料庫執行個體。

命名空間：**aws:rds:dbinstance**

名稱	描述	預設	有效值
DBAllocatedStorage	分配的資料庫儲存大小，以 GB 為單位。	MySQL: 5 Oracle: 10 sqlserver-se：200 sqlserver-ex：30 sqlserver-web：30	MySQL: 5-1024 Oracle: 10-1024 sqlserver：不能修改
DBDeletionPolicy	指定在環境終止時，是否要保留、刪除或建立資料庫執行個體的快照。 此選項適合搭配 HasCoupledDatabase 使用，也是此命名空間的一個選項。 ⚠ Warning 刪除資料庫執行個體會產生永久性資料遺失。	Delete	Delete Retain Snapshot
DBEngine	要為此執行個體使用的資料庫引擎名稱。	mysql	mysql oracle-se1 sqlserver-ex sqlserver-web sqlserver-se

名稱	描述	預設	有效值
			`postgres`
DBEngineVersion	資料庫引擎版本編號。	`5.5`	
DBInstanceClass	資料庫執行個體類型。	`db.t2.micro` (`db.m1.large` 適用於不在 Amazon VPC 中執行的環境)	如需詳細資訊，請參閱 Amazon Relational Database Service 使用者指南中的資料庫執行個體類別。
DBPassword	要為此資料庫執行個體使用的主要使用者密碼名稱。	無	
DBSnapshotIdentifier	恢復資料庫快照的識別碼。	無	
DBUser	此資料庫執行個體的主要使用者名稱。	ebroot	

名稱	描述	預設	有效值

HasCoupledDatabase	指定資料庫執行個體是否要與環境耦合。如果切換為 `true`，Elastic Beanstalk 會建立與您環境耦合的新資料庫執行個體。如果切換為 `false`，Elastic Beanstalk 會從您的環境啟動資料庫執行個體的解耦。
	
	此選項適合搭配 `DBDeletionPolicy` 使用，也是此命名空間的一個選項。
	
	Note
	注意：如果您在解耦舊有資料庫後，將此值切換回 `true`，則 Elastic Beanstalk 會使用舊有資料庫選項設定來建立新資料庫。不過，為了維護環境的安全性，其不會保留現有的 `DBUser` 和 `DBPassword` 設定。您需要再次指定 `DBUser` 和 `DBPassword` 。
		`false`	`true`
			
			`false`

MultiAZDatabase	指定是否需要建立資料庫執行個體異地同步備份部署。如需使用 Amazon Relational Database Service (RDS) 進行異地同步備份部署的詳細資訊，請參閱 Amazon Relational Database Service 使用者指南中的區域與可用區域。	`false`	`true`
			
			`false`

平台特定選項

某些 Elastic Beanstalk 平台會定義特定於平台的選項命名空間。這些命名空間及其選項會列在每個平台的下方。

> **Note**
>
> 先前在以 Amazon Linux AMI (Amazon Linux 2 之前的版本) 為基礎的平台版本中，下列兩個功能及其各自的命名空間會視為平台特定的功能，並在此列出每個平台：
>
> - 靜態檔案的代理組態 –`aws:elasticbeanstalk:environment:proxy:staticfiles`
> - AWS X-Ray 支援t – `aws:elasticbeanstalk:xray`
>
> 在 Amazon Linux 2 平台版本中，Elastic Beanstalk 會以一致的方式在所有支援的平台上實作這些功能。相關的命名空間現在會列在 the section called “一般選項” 頁面中。對於具有不同名稱的命名空間平台，我們只會在此頁面上提到它們。

平台

- Docker 平台選項
- Go 平台選項
- Java SE 平台選項
- 搭配 Tomcat 平台的 Java 選項
- Linux 上的 .NET Core 平台選項
- .NET 平台選項
- Node.js 平台選項
- PHP 平台選項
- Python 平台選項
- Ruby 平台選項

Docker 平台選項

下列 Docker 特定的組態選項會套用至 Docker 和預先設定的 Docker 平台。

> **Note**
>
> 這些組態選項不適用於
>
> - 包含 Docker Compose 的 Docker 平台 (Amazon Linux 2)

- 多容器 Docker 平台 (Amazon Linux AMI)

命名空間：**aws:elasticbeanstalk:environment:proxy**

名稱	描述	預設	有效值
ProxyServer	指定 Web 伺服器當成 Proxy 使用。	nginx	nginx none – Amazon Linux AM 和 Docker (僅包含 DC)

Go 平台選項

Amazon Linux AMI (Amazon Linux 2 之前的版本) 平台選項

命名空間：**aws:elasticbeanstalk:container:golang:staticfiles**

您可以使用以下命名空間，設定代理伺服器以提供靜態檔案。代理伺服器收到位於指定路徑下的檔案請求時，會直接提供檔案而非將請求路由至您的應用程式。這可減少應用程式必須處理的請求數量。

將代理伺服器提供的路徑對應至包含靜態資產之原始程式碼中的資料夾。您在此命名空間中定義的每個選項都會對應至不同路徑。

名稱	Value (值)
代理伺服器提供檔案的路徑。 範例：/images 將於 *subdomain* .elasticbeanstalk.com/images 提供檔案。	內含檔案的資料夾名稱。 範例：staticimages 將自原始碼套件最上層名為 staticimages 的資料夾提供檔案。

Java SE 平台選項

Amazon Linux AMI (Amazon Linux 2 之前的版本) 平台選項

命名空間：**aws:elasticbeanstalk:container:java:staticfiles**

您可以使用以下命名空間，設定代理伺服器以提供靜態檔案。代理伺服器收到位於指定路徑下的檔案請求時，會直接提供檔案而非將請求路由至您的應用程式。這可減少應用程式必須處理的請求數量。

將代理伺服器提供的路徑對應至包含靜態資產之原始程式碼中的資料夾。您在此命名空間中定義的每個選項都會對應至不同路徑。

名稱	Value (值)
代理伺服器提供檔案的路徑。 範例：`/images` 將於 *`subdomain`* `.elasticbeanstalk.com/images` 提供檔案。	內含檔案的資料夾名稱。 範例：`staticimages` 將自原始碼套件最上層名為 `staticimages` 的資料夾提供檔案。

搭配 Tomcat 平台的 Java 選項

命名空間：**aws:elasticbeanstalk:application:environment**

名稱	描述	預設	有效值
JDBC_CONNECTION_STRING	外部資料庫的連線字串。	N/A	無

如需詳細資訊，請參閱 環境屬性與其他軟體設定。

命名空間：**aws:elasticbeanstalk:container:tomcat:jvmoptions**

名稱	描述	預設	有效值
JVM Options	在啟動時將命令列選項傳送至 JVM。	無	無
Xmx	JVM 堆積大小上限。	256m	無
XX:MaxPermSize	JVM 堆積的一個區段，用來儲存類別定義與相關的中繼資料。	64m	N/A

名稱	描述	預設	有效值
	Note 此選項僅適用於 Java 8 之前的 Java 版本，在以 Amazon Linux 2 和更新版本為依據的 Elastic Beanstalk Tomcat 平台上不受支援。		
Xms	初始 JVM 堆積大小。	`256m`	無
`optionName`	除了 Tomcat 平台定義的選項，亦指定任意 JVM 選項。	無	無

命名空間：**`aws:elasticbeanstalk:environment:proxy`**

名稱	描述	預設	有效值
GzipCompression	設定為 `false` 以停用回應內容壓縮。 僅適用於 Amazon Linux AMI (Amazon Linux 2 之前的版本) 平台版本。	`true`	`true` `false`

ProxyServer	設定 Proxy 在您的環境的執行個體上使用。如果您將此選項設定為 `apache`，則 Elastic Beanstalk 會使用 Apache 2.4。
	如果您的應用程式因為不相容的代理組態設定而未準備好從 Apache 2.2 遷移，請設定 `apache/2.2` 。此數值僅適用於 Amazon Linux AMI (Amazon Linux 2 之前的版本) 平台版本。
	設定為 `nginx` 以使用 nginx。這是從 Amazon Linux 2 平台版本開始的預設值。
	如需更多詳細資訊，請參閱 設定您的 Tomcat 環境的代理伺服器。
		`nginx` (Amazon Linux 2)
		`apache` (Amazon Linux AMI)
			`apache`
			`apache/2.2` – 僅限 Amazon Linux AMI
			`nginx`

Linux 上的 .NET Core 平台選項

命名空間：**aws:elasticbeanstalk:environment:proxy**

名稱	描述	預設	有效值
ProxyServer	指定 Web 伺服器當成 Proxy 使用。	nginx	nginx none

.NET 平台選項

命名空間：**aws:elasticbeanstalk:container:dotnet:apppool**

名稱	描述	預設	有效值
Target Runtime	選擇您應用程式適用的 .NET Framework 版本。	4.0	2.0 4.0
Enable 32-bit Applications	設定為 True 以執行 32 位元應用程式。	False	True False

Node.js 平台選項

命名空間：**aws:elasticbeanstalk:environment:proxy**

名稱	描述	預設	有效值
ProxyServer	設定 Proxy 在您的環境的執行個體上使用。	nginx	apache nginx

Amazon Linux AMI (Amazon Linux 2 之前的版本) 平台選項

命名空間：**`aws:elasticbeanstalk:container:nodejs`**

名稱	描述	預設	有效值
NodeCommand	用於啟動 Node.js 應用程式的命令。若指定空白字串，可依序使用 `app.js`、`server.js` 和 `npm start`。	""	無
NodeVersion	Node.js 版本。例如：`4.4.6` 支援的 Node.js 版本會依 Node.js 平台版本而有所不同。如需目前支援版本的清單，請參閱 AWS Elastic Beanstalk 平台文件中的 Node.js。 **Note** 如果您所使用的 Node.js 版本的支援已從平台移除，您必須在平台更新之前變更或移除版本設定。當發現一個或多個版本的 Node.js 存在安全漏洞時，可能會出現此種狀況。此時，如果嘗試將不支援已設定 NodeVersion 的平台更新至新版本，此動作將會失敗。為省去建立新環境的必要，請將 NodeVersion 組態選項變更為新舊平台版本皆支援的 Node.js 版本，或是移除選項設定，然後執行平台更新。	各有不同	各有不同

GzipCompression	指定是否啟用 gzip 壓縮。若 ProxyServer 設定為 none，則會停用 gzip 壓縮。	`false`	`true`
			
			`false`

名稱	描述	預設	有效值
ProxyServer	指定哪些 Web 伺服器應用於 Node.js 的代理連線。若 ProxyServer 設為 none，則靜態檔案映射不會生效且 gzip 壓縮會停用。	nginx	apache nginx none

命名空間：**aws:elasticbeanstalk:container:nodejs:staticfiles**

您可以使用以下命名空間，設定代理伺服器以提供靜態檔案。代理伺服器收到位於指定路徑下的檔案請求時，會直接提供檔案而非將請求路由至您的應用程式。這可減少應用程式必須處理的請求數量。

將代理伺服器提供的路徑對應至包含靜態資產之原始程式碼中的資料夾。您在此命名空間中定義的每個選項都會對應至不同路徑。

> **Note**
>
> 若 aws:elasticbeanstalk:container:nodejs::ProxyFiles 設定為 none，則靜態檔案設定不會套用。

名稱	Value (值)
代理伺服器提供檔案的路徑。 範例：/images 將於 *subdomain* .elasticbeanstalk.com/images 提供檔案。	內含檔案的資料夾名稱。 範例：staticimages 將自原始碼套件最上層名為 staticimages 的資料夾提供檔案。

PHP 平台選項

命名空間：**aws:elasticbeanstalk:container:php:phpini**

名稱	描述	預設	有效值
document_root	指定您專案視為公開 Web 根的子目錄。	/	空白字串將視為 /，或以 / 開頭指定字串

名稱	描述	預設	有效值
memory_limit	已配置給 PHP 環境的記憶體數量。	256M	無
zlib.output_compression	指定 PHP 是否應使用壓縮來輸出。	Off	On Off true false
allow_url_fopen	指定 PHP 檔案函數是否允許自遠端位置 (如網站或 FTP 伺服器) 擷取資料。	On	On Off true false
display_errors	指定錯誤訊息是否為輸出的一部分。	Off	On Off
max_execution_time	設定指令碼於環境將其終止前可執行的時間上限 (秒)。	60	0 至 9223372036854775807 (PHP_INT_MAX)
composer_options	使用 Composer 透過 composer.phar install 安裝依存項目時，設定欲使用的自訂選項。如需包含可用選項的詳細資訊，請前往 http://getcomposer.org/doc/03-cli.md#install。	無	無

命名空間：**aws:elasticbeanstalk:environment:proxy**

名稱	描述	預設	有效值
ProxyServer	設定 Proxy 在您的環境的執行個體上使用。	nginx	apache

名稱	描述	預設	有效值
			`nginx`

> **Note**
>
> 如需 PHP 平台的詳細資訊，請參閱使用 Elastic Beanstalk PHP 平台.

Python 平台選項

命名空間：**`aws:elasticbeanstalk:application:environment`**

名稱	描述	預設	有效值
DJANGO_SETTINGS_MODULE	指定欲使用的設定檔案。	無	無

如需詳細資訊，請參閱 環境屬性與其他軟體設定。

命名空間：**`aws:elasticbeanstalk:container:python`**

名稱	描述	預設	有效值
WSGIPath	內含 WSGI 應用程式的檔案。此檔案必須擁有可呼叫的 `application` 。	在 Amazon Linux 2 Python 平台版本上：`application` 在 Amazon Linux AMI Python 平台版本上：`application.py`	無

名稱	描述	預設	有效值
NumProcesses	執行 WSGI 應用程式時，應為程序群組啟動的協助程式程序數量。	1	無
NumThreads	執行 WSGI 應用程式時，在程序群組各個協助程式程序中處理請求需建立的執行緒數量。	15	無

命名空間：**aws:elasticbeanstalk:environment:proxy**

名稱	描述	預設	有效值
ProxyServer	設定 Proxy 在您的環境的執行個體上使用。	nginx	apache nginx

Amazon Linux AMI (Amazon Linux 2 之前的版本) 平台選項

命名空間：**aws:elasticbeanstalk:container:python:staticfiles**

您可以使用以下命名空間，設定代理伺服器以提供靜態檔案。代理伺服器收到位於指定路徑下的檔案請求時，會直接提供檔案而非將請求路由至您的應用程式。這可減少應用程式必須處理的請求數量。

將代理伺服器提供的路徑對應至包含靜態資產之原始程式碼中的資料夾。您在此命名空間中定義的每個選項都會對應至不同路徑。

根據預設，Python 環境中的代理伺服器會從 `static` 路徑上名為 `/static` 的資料夾中提供任何檔案。

命名空間：**aws:elasticbeanstalk:container:python:staticfiles**

名稱	Value (值)
代理伺服器提供檔案的路徑。 範例：`/images` 將於 *subdomain* `.elasticbeanstalk.com/images` 提供檔案。	內含檔案的資料夾名稱。 範例：`staticimages` 將自原始碼套件最上層名為 `staticimages` 的資料夾提供檔案。

Ruby 平台選項

命名空間：**aws:elasticbeanstalk:application:environment**

名稱	描述	預設	有效值
RAILS_SKIP_MIGRATIONS	指定是否代表使用者應用程式執行 `rake db:migrate` ，或者應略過。僅適用 Rails 3 應用程式。	false	true false
RAILS_SKIP_ASSET_COMPILATION	指定容器是否應代表使用者應用程式執行 `rake assets:precompile` ，或者應略過。亦僅適用 Rails 3 應用程式。	false	true false
BUNDLE_WITHOUT	自 Gemfile 安裝依存項目時應忽略的群組清單，以冒號 (:) 分隔。	test:development	無
RACK_ENV	指定可以執行應用程式的環境階段。常見環境的範例包括開發、生產、測試。	production	無

如需詳細資訊，請參閱「環境屬性與其他軟體設定」。

自訂選項

使用 aws:elasticbeanstalk:customoption 命名空間來定義可於其他組態檔案 Resources 區塊讀取的選項與值。使用自訂選項來收集使用者於單一組態檔案中指定的設定。

例如，您可能有一個複雜的組態檔案，該檔案定義使用者能夠於啟動環境時設定的資源。若您使用 Fn::GetOptionSetting 來擷取自訂選項的數值，您可以將該選項的定義，放置於另一個使用者能夠更輕鬆探索並修改的組態檔案。

此外，由於自訂選項為組態選項，因此可於 API 層級設定，覆寫組態檔案設定的數值。如需詳細資訊，請參閱優先順序。

自訂選項的定義方式與其他選項類似：

```
option_settings:
  aws:elasticbeanstalk:customoption:
    option name: option value
```

例如，下列組態檔案會建立名為 ELBAlarmEmail 的選項，並將值設定為 someone@example.com：

```
option_settings:
  aws:elasticbeanstalk:customoption:
    ELBAlarmEmail: someone@example.com
```

在其他方面，組態檔案的定義的 SNS 主題，可透過 Fn::GetOptionSetting 讀取選項來填入 Endpoint 屬性的數值：

```
Resources:
  MySNSTopic:
    Type: AWS::SNS::Topic
    Properties:
      Subscription:
        - Endpoint:
            Fn::GetOptionSetting:
              OptionName: ELBAlarmEmail
              DefaultValue: nobody@example.com
          Protocol: email
```

您可在 Fn::GetOptionSetting 找到更多使用 新增和自訂 Elastic Beanstalk 環境資源 的範例程式碼片段。

使用組態檔案 (.ebextensions) 來進行進階的環境自訂

您可以將 AWS Elastic Beanstalk 組態檔案 (.ebextensions) 新增至 Web 應用程式的原始程式碼，以設定環境並自訂其包含的 AWS 資源。組態檔案是具有副.config檔名的 YAML 或 JSON 格式文件，您可以將其放置在應用程式來源服務包中名為.ebextensions並部署的資料夾中。

Example .ebextension/. network-load-balancer 配置

此範例會進行簡單的組態變更。它會修改組態選項，以將環境負載平衡器的類型設定為 Network Load Balancer。

```
option_settings:
  aws:elasticbeanstalk:environment:
    LoadBalancerType: network
```

我們建議對組態檔案使用 YAML，因為與 JSON 相較，此格式可讀性較高。YAML 支援評論、多列命令、使用引號的數個替代選項等等。不過，您可以在 Elastic Beanstalk 組態檔案中以完全相同的方法使用 YAML 或 JSON 來進行任何組態變更。

> **秘訣**
>
> 在您開發或測試新的組態檔案時，請啟動執行預設應用程式的空白環境，然後將組態檔案部署到此環境。不當格式的組態檔案，將會導致新環境啟動失敗而無法回復。

組態檔案的 `option_settings` 區段定義了組態選項的值。組態選項可讓您設定 Elastic Beanstalk 環境、其中的 AWS 資源，以及執行應用程式的軟體。組態檔案只是設定組態選項的幾種方式其中一種。

此`Resources`區段可讓您進一步自訂應用程式環境中的資源，並定義組態選項所提供功能以外的其他 AWS 資源。您可以新增和設定任何由 AWS CloudFormation Elastic Beanstalk 用來建立環境的資源。

組態檔案中的其他區段
(`packages`、`sources`、`files`、`users`、`groups`、`commands`、`container_commands` 和 `services`)，可讓您設定您的環境中啟動的 EC2 執行個體。每當伺服器在您的環境中啟動時，Elastic Beanstalk 會執行這些區段中所定義的操作，來準備適用於您應用程式的作業系統和儲存系統。

如需常用的 .ebextensions 範例，請參閱 Elastic Beanstalk 組態檔案儲存庫。

要求

- 位置-Elastic Beanstalk 將處理部署中存在的所有`.ebextensions`文件夾。但是，我們建議您將所有組態檔放在來源套件的根目錄中`.ebextensions`，名為的單一資料夾中。檔案瀏覽器可以隱藏用英文句點開頭的資料夾，因此在您建立原始碼套件時，請確認已加入此資料夾。如需詳細資訊，請參閱 建立應用程式原始碼套件。
- 命名 – 組態檔案的副檔名必須為 `.config`。
- 格式化 – 組態檔案必須符合 YAML 或 JSON 規格的格式要求。

  使用 YAML 時，一律使用空格來對位於不同巢狀層級的金鑰進行縮排。如需關於 YAML 的詳細資訊，請參閱 YAML 不是標記語言 (YAML™) 1.1 版。
- 唯一性 – 在每個組態檔案中僅使用每個金鑰一次。

> **警告**
>
> 如果您在同一個組態檔案中使用某個金鑰 (例如option_settings) 兩次，則其中一個區段將會被刪除。請將重複的區段合併為單一區段，或是將這些區段放置於不同的組態檔案中。

進行部署的程序，會視您用來管理環境的用戶端而有些微的不同。如需詳細資訊，請參閱下列的章節：

- Elastic Beanstalk 主控台
- EB CLI
- AWS CLI

主題

- 選項設定
- 在 Linux 伺服器上自訂軟體
- 在 Windows Server 上自訂軟體
- 新增和自訂 Elastic Beanstalk 環境資源

選項設定

您可使用 option_settings 金鑰來修改 Elastic Beanstalk 組態，並定義您的應用程式使用環境變數所擷取的變數。部分命名空間可讓您增加參數數量，並指定參數名稱。如需命名空間和組態選項的清單，請參閱 組態選項。

選項設定可於環境建立或環境更新期間，直接套用至環境。直接套用至環境的設定，會覆寫組態檔案中相同選項的設定。若您移除環境資訊的設定，組態檔案內的設定將生效。如需詳細資訊，請參閱 優先順序。

語法

選項設定的標準語法為物件陣列，各個陣列均具有 namespace、option_name 和 value 金鑰。

```
option_settings:
  - namespace:  namespace
    option_name:  option name
    value:  option value
```

```
  - namespace:  namespace
    option_name:  option name
    value:  option value
```

namespace 金鑰為選用。若您未指定命名空間，使用的預設為 aws:elasticbeanstalk:application:environment：

```
option_settings:
  - option_name:  option name
    value:  option value
  - option_name:  option name
    value:  option value
```

Elastic Beanstalk 亦支援選項設定的速記語法，您能夠於命名空間下方以金鑰值對形式指定選項：

```
option_settings:
  namespace:
    option name: option value
    option name: option value
```

範例

下列範例於 aws:elasticbeanstalk:container:tomcat:jvmoptions 命名空間及名為 MYPARAMETER 的環境屬性中，設定 Tomcat 平台特定的選項。

標準 YAML 格式為：

Example .ebextensions/options.config

```
option_settings:
  - namespace:  aws:elasticbeanstalk:container:tomcat:jvmoptions
    option_name:  Xmx
    value:  256m
  - option_name: MYPARAMETER
    value: parametervalue
```

速記格式為：

Example .ebextensions/options.config

```
option_settings:
  aws:elasticbeanstalk:container:tomcat:jvmoptions:
```

```
    Xmx: 256m
  aws:elasticbeanstalk:application:environment:
    MYPARAMETER: parametervalue
```

JSON 為：

Example .ebextensions/options.config

```
{
  "option_settings": [
    {
      "namespace": "aws:elasticbeanstalk:container:tomcat:jvmoptions",
      "option_name": "Xmx",
      "value": "256m"
    },
    {
      "option_name": "MYPARAMETER",
      "value": "parametervalue"
    }
  ]
}
```

在 Linux 伺服器上自訂軟體

您可能想要針對您的應用程式相依使用的軟體，來進行自訂和設定。您可以新增要在執行個體佈建期間執行的命令；定義 Linux 使用者和群組；以及在您的環境執行個體上下載或直接建立檔案。這些檔案可能是應用程式所需的相依檔案 (例如由 yum 儲存庫所提供的其他套件)，或組態檔案 (例如代理組態檔案的替代檔案，可用來覆寫 Elastic Beanstalk 預設的特定設定)。

本節說明可以在組態檔案中納入的資訊類型，此等組態檔案是用來針對執行 Linux 的 EC2 執行個體，自訂其上的軟體。如需自訂和設定 Elastic Beanstalk 環境的一般資訊，請參閱設定 Elastic Beanstalk 環境。關於針對執行 Windows 的 EC2 執行個體，來自訂其上的軟體，詳細資訊請參閱 在 Windows Server 上自訂軟體。

> **備註**
>
> - 在 Amazon Linux 2 平台上，我們強烈建議您使用 Buildfile，而不要在 .ebextensions 組態檔案中提供檔案和命令。Procfile 和平台掛鉤會盡可能在執行個體佈建期間，在您的環境執行個體上設定和執行自訂程式碼。如需有關這些機制的詳細資訊，請參閱 the section called “擴充 Linux 平台”。

- YAML 憑藉一致的縮排。請在取代範例組態檔中的內容時，讓縮排層級一致，並確認您的文字編輯器使用空格而非定位字元進行縮排。

組態檔案支援下列金鑰，這些金鑰會對執行您應用程式的 Linux 伺服器造成影響。

金鑰

- 套件
- 群組
- 使用者
- 來源
- 檔案
- 命令
- 服務
- 容器命令
- 範例：使用自訂 Amazon CloudWatch 指標

這些金鑰會以此列的順序處理。

觀看您環境的事件的同時，開發與測試組態檔案。Elastic Beanstalk 忽略組態檔案，其中包含驗證錯誤，例如無效的金鑰，並不處理同一個檔案其他任何金鑰。當發生這種情況，Elastic Beanstalk 會將警告事件新增到事件日誌。

套件

您可以使用 packages 金鑰，來下載和安裝預先整裝好的應用程式與元件。

語法

```
packages:
  name of package manager:
    package name: version
    ...
  name of package manager:
    package name: version
    ...
  ...
```

您可以使用每個套件管理工具的金鑰來指定多個套件。

支援的套件格式

Elastic Beanstalk 目前支援下列的套件軟體管理工具：yum、rubygems、python 和 rpm。套件會以下列順序處理：rpm、yum，然後是 rubygems 和 python。rubygems 和 python 沒有一定的先後順序。在每個套件管理工具內，不保證套件安裝順序。請使用您的作業系統支援的套件軟體管理工具。

> **Note**
>
> Elastic Beanstalk 支援 Python 適用的兩種基本套件軟體管理工具：pip 和 easy_install。不過，在組態檔案的語法中，您必須將套件軟體管理工具的名稱指定為 `python`。當您使用組態檔案指定 Python 套件管理工具時，Elastic Beanstalk 會使用 Python 2.7。如果您的應用程式倚賴不同版本的 Python，您可以在 `requirements.txt` 檔案中指定要安裝的套件。如需更多詳細資訊，請參閱 使用要求檔案指定相依性。

指定版本

在每個套件管理工具中，系統會以套件名稱和版本清單來指定各個套件。版本可以是字串、版本清單，亦能是空白字串或清單；空白字串或清單代表要安裝最新版本。如果管理工具為 rpm 格式，則系統會以磁碟上的檔案路徑或 URL 來指定版本。不支援相對路徑。

如果您指定套件的版本，則即使在執行個體上已經安裝該套件的較新版本，Elastic Beanstalk 也會試著安裝您指定的版本。如果已經安裝較新的版本，部署作業會失敗。部分套件管理工具支援多個版本，有些則不支援。如需詳細資訊，請參閱套件管理工具的文件。如果您並未指定版本，而且套件已經安裝某版本，則 Elastic Beanstalk 不會安裝新版本，因為 Elastic Beanstalk 假設您想要保留和使用現有的版本。

範例程式碼片段

下列的程式碼片段針對 rpm 指定了版本 URL、要求 yum 的最新版本，以及 rubygems 的 chef 0.10.2 版。

```
packages:
  yum:
    libmemcached: []
    ruby-devel: []
    gcc: []
  rpm:
    epel: http://download.fedoraproject.org/pub/epel/5/i386/epel-release-5-4.noarch.rpm
```

```
  rubygems:
    chef: '0.10.2'
```

群組

您可以使用 groups 金鑰來建立 Linux/UNIX 群組和指派群組 ID。若要建立群組，請新增一組新的索引鍵/值組，此對組會將新的群組名稱對應至選用的群組 ID。groups 金鑰可以包含一個或多個群組名稱。下表會列出可用的金鑰。

語法

```
groups:
  name of group: {}
  name of group:
    gid: "group id"
```

選項

gid

群組 ID 編號。

如果指定了群組 ID，而該群組的名稱已存在，則群組建立作業會失敗；如果另一個群組已具有指定的 ID，作業系統可能會拒絕群組的建立。

範例程式碼片段

下列程式碼片段指定了名為 groupOne 的群組 (未指派群組 ID)，以及名為 groupTwo 的群組 (指定群組 ID 值 45)。

```
groups:
  groupOne: {}
  groupTwo:
    gid: "45"
```

使用者

您可以使用 users 金鑰，在 EC2 執行個體上建立 Linux/UNIX 使用者。

語法

```
users:
```

```
  name of user:
    groups:
      - name of group
    uid: "id of the user"
    homeDir: "user's home directory"
```

選項

uid

使用者 ID。如果使用者名稱具有不同的使用者 ID，則建立程序會失敗。如果該使用者 ID 已指派給現有的使用者，作業系統可能會拒絕建立要求。

groups

群組名稱清單。使用者會新增到清單中的各群組。

homeDir

使用者的主目錄。

會使用 /sbin/nologin 的 shell，來將使用者建立為非互動式系統的使用者。這是設計本身所致，無法修改。

範例程式碼片段

```
users:
  myuser:
    groups:
      - group1
      - group2
    uid: "50"
    homeDir: "/tmp"
```

來源

您可以使用 sources 金鑰，從公開的 URL 下載封存檔案，然後將檔案解壓縮至 EC2 執行個體上的目標目錄。

語法

```
sources:
```

```
  target directory: location of archive file
```

支援的格式

支援的格式包括 tar、tar+gzip、tar+bz2 和 zip。只要 URL 是可公開存取，您就可以參照 Amazon Simple Storage Service (Amazon S3) 等外部位置 (例如，`https://mybucket.s3.amazonaws.com/myobject`)。

範例程式碼片段

下列範例從 Amazon S3 儲存貯體下載了公有的 .zip 檔案，並將該檔案解壓縮至 `/etc/myapp`：

```
sources:
  /etc/myapp: https://mybucket.s3.amazonaws.com/myobject
```

> **Note**
>
> 多項擷取不應重複使用相同的目標路徑。擷取另一個相同目標路徑的來源，將會取代而不是附加資料到內容。

檔案

透過 `files` 金鑰，即可在 EC2 執行個體上建立檔案。此內容可以內嵌於組態檔案，或是從 URL 取得。檔案會依詞典編纂順序寫入磁碟。

您可以提供授權用的執行個體描述檔，以使用 `files` 金鑰來從 Amazon S3 下載私有檔案。

如果您指定的檔案路徑已經存在於例證上，則會保留現有檔案，並將副檔名 `.bak` 附加至其名稱。

語法

```
files:
  "target file location on disk":
     mode: "six-digit octal value"
     owner: name of owning user for file
     group: name of owning group for file
     source: URL
     authentication: authentication name:

  "target file location on disk":
     mode: "six-digit octal value"
```

```
    owner: name of owning user for file
    group: name of owning group for file
    content: |
     # this is my
     # file content
    encoding: encoding format
    authentication: authentication name:
```

選項

`content`

要新增到檔案的字串內容。請指定 `content` 或 `source` 其中之一。

`source`

要下載檔案的 URL。請指定 `content` 或 `source` 其中之一。

`encoding`

使用 `content` 選項所指定字串的編碼格式。

有效值：`plain` | `base64`

`group`

擁有檔案的 Linux 群組。

`owner`

擁有檔案的 Linux 使用者。

`mode`

六位數的 8 進制值，代表此檔案的模式；Windows 系統不支援此金鑰。請使用前三位數來建立符號連結，並採用後三位數來設定許可。若要建立符號連結，請指定 `120`*`xxx`*，其中 `xxx` 定義目標檔案的許可。若要指定檔案的許可，請使用三個數字，例如 `000644`。

`authentication`

所要使用 AWS CloudFormation 身分驗證方法的名稱。您可以利用 Resources (資源) 金鑰，在 Auto Scaling 群組的中繼資料中新增身分驗證方法。如需範例，請參閱下列內容。

範例程式碼片段

```
files:
```

```
  "/home/ec2-user/myfile" :
    mode: "000755"
    owner: root
    group: root
    source: http://foo.bar/myfile

  "/home/ec2-user/myfile2" :
    mode: "000755"
    owner: root
    group: root
    content: |
      this is my
      file content
```

使用符號連結的範例。這會建立一個 /tmp/myfile2.txt 連結，指向現有檔案 /tmp/myfile1.txt。

```
files:
  "/tmp/myfile2.txt" :
    mode: "120400"
    content: "/tmp/myfile1.txt"
```

下列範例使用了 Resources (資源) 金鑰來新增名為 S3Auth 的身分驗證方法，並使用此方法，從 Amazon S3 儲存貯體下載私有檔案。

```
Resources:
  AWSEBAutoScalingGroup:
    Metadata:
      AWS::CloudFormation::Authentication:
        S3Auth:
          type: "s3"
          buckets: ["elasticbeanstalk-us-west-2-123456789012"]
          roleName:
            "Fn::GetOptionSetting":
              Namespace: "aws:autoscaling:launchconfiguration"
              OptionName: "IamInstanceProfile"
              DefaultValue: "aws-elasticbeanstalk-ec2-role"

files:
  "/tmp/data.json" :
    mode: "000755"
    owner: root
```

```
    group: root
    authentication: "S3Auth"
    source: https://elasticbeanstalk-us-west-2-123456789012.s3-us-west-2.amazonaws.com/
data.json
```

命令

您可以使用 commands 金鑰來執行 EC2 執行個體上的指令。在設定應用程式和 Web 伺服器和解壓縮應用程式版本的檔案之前執行命令。

以根使用者身分執行指定的命令，並依照名稱的字母順序處理。根據預設，指令會在根目錄中執行。若要從其他目錄來執行指令，請使用 cwd 選項。

若要對命令的問題進行疑難排解，您可以在執行個體日誌中尋找命令的輸出。

語法

```
commands:
  command name:
    command: command to run
    cwd: working directory
    env:
      variable name: variable value
    test: conditions for command
    ignoreErrors: true
```

選項

command

指定要執行命令的陣列 (YAML 語法的區塊序列集合) 或字串。部分重要說明：

- 如果使用字串，您不需要將整個字串括在引號中。如果使用引號，請逸出出現的相同類型引號常值。
- 如果使用陣列，則不需跳脫空格字元或用引號括起指令參數。每個陣列元素是單一命令引數。不使用陣列指定多個命令。

以下範例都是同等的：

```
commands:
  command1:
    command: git commit -m "This is a comment."
```

```
  command2:
    command: "git commit -m \"This is a comment.\""
  command3:
    command: 'git commit -m "This is a comment."'
  command4:
    command:
      - git
      - commit
      - -m
      - This is a comment.
```

若要指定多個命令，請使用常值區塊純量，如下所示。

```
commands:
  command block:
    command: |
      git commit -m "This is a comment."
      git push
```

`env`

(選用) 設定命令用的環境變數。此屬性會進行覆寫，而非附加至現有的環境。

`cwd`

(選用) 工作目錄。若未指定，指令會從根目錄 (/) 執行。

`test`

(選用) 此指令必須傳回 `true` 值 (結束代碼 0)，Elastic Beanstalk 才能處理 `command` 金鑰中包含的指令 (例如 shell 指令碼)。

`ignoreErrors`

(選用) 一個布林值，在 `command` 金鑰中所包含的指令失敗時 (傳回非零的值)，可用來判定其他的指令是否應該執行。如果即使命令失敗，也想繼續執行命令，請將此值設定為 `true`。如果要在命令失敗時停止執行命令，請將此值設定為 `false`。預設值為 `false`。

範例程式碼片段

下列的範例片段執行了 Python 指令碼。

```
commands:
  python_install:
```

```
    command: myscript.py
    cwd: /home/ec2-user
    env:
      myvarname: myvarvalue
    test: "[ -x /usr/bin/python ]"
```

服務

您可以使用 services 金鑰，來定義執行個體啟動時應啟動或停止的服務。services 金鑰也可讓您指定來源、套件與檔案的相依關係，因此如果安裝中的檔案需要重新啟動，則 Elastic Beanstalk 會負責重新啟動服務。

語法

```
services:
  sysvinit:
    name of service:
      enabled: "true"
      ensureRunning: "true"
      files:
        - "file name"
      sources:
        - "directory"
      packages:
        name of package manager:
          "package name[: version]"
      commands:
        - "name of command"
```

選項

ensureRunning

設定為 true，以確保服務會在 Elastic Beanstalk 完成後執行。

設定為 false，以確保服務不會在 Elastic Beanstalk 完成後執行。

如果略過此金鑰，將不會改變服務狀態。

enabled

設定為 true，來確保服務會在開機時自動啟動。

設定為 false，來確保服務不會在開機時自動啟動。

如果略過此金鑰，將不會改變此屬性。

`files`

檔案清單。如果 Elastic Beanstalk 透過檔案區塊直接變更檔案，服務會重新啟動。

`sources`

目錄清單。如果 Elastic Beanstalk 將封存解壓縮至這些目錄的其中之一，服務會重新啟動。

`packages`

套件軟體管理工具與套件名稱清單的對應圖。如果 Elastic Beanstalk 安裝或更新這些套件的其中之一，服務會重新啟動。

`commands`

命令名稱清單。如果 Elastic Beanstalk 執行指定的命令，服務會重新啟動。

範例程式碼片段

下列是範例程式碼片段：

```
services:
  sysvinit:
    myservice:
      enabled: true
      ensureRunning: true
```

容器命令

您可以使用 `container_commands` 金鑰，來執行會影響您應用程式原始碼的指令。容器命令的執行會在應用程式和 Web 伺服器設定完成及應用程式版本封存檔解壓縮之後，但是在應用程式版本安裝之前。非容器的命令及其他自訂操作，則會在應用程式的原始碼解壓縮之前執行。

以根使用者身分執行指定的命令，並依照名稱的字母順序處理。容器命令會從暫存目錄執行，您的原始碼會在部署到應用程式伺服器之前，先解壓縮到該目錄。當原始碼部署到最終的位置時，您使用容器命令對暫存目錄中原始碼所進行的變更，也會納入部署。

> **Note**
>
> 容器命令的輸出會記錄在 `cfn-init-cmd.log` 執行個體日誌中。如需擷取和檢視執行個體日誌的詳細資訊，請參閱從 Amazon EC2 執行個體檢視日誌。

如果只要在單一執行個體上執行指令，您可以使用 leader_only 選項；或者，請將 test 設定為只在測試指令得出的評估值為 true 時，才執行指令。僅限領導者的容器命令，只會在環境建立與部署時執行；其他的命令與伺服器自訂操作，則會在每次佈建或更新執行個體時執行。僅限領導者的容器命令會因為啟動組態變更 (例如變更 AMI ID 或執行個體類型) 而不執行。

語法

```
container_commands:
  name of container_command:
    command: "command to run"
    leader_only: true
  name of container_command:
    command: "command to run"
```

選項

command

要執行的字串或字串陣列。

env

(選用) 在執行命令前先設定環境變數，覆寫掉任何現有的值。

cwd

(選用) 工作目錄。根據預設，這是已經解壓縮應用程式的暫存目錄。

leader_only

(選用) 只在 Elastic Beanstalk 所選擇的單一執行個體上執行指令。僅限領導者的容器命令會先於其他容器命令執行。命令可以是只限領導者或具有 test，但只能是其中一種 (leader_only 會具有優先性)。

test

(選用) 執行測試命令，此命令必須傳回 true 才能執行容器命令。命令可以是只限領導者或具有 test，但只能是其中一種 (leader_only 會具有優先性)。

ignoreErrors

(選用) 如果容器命令傳回 0 以外的值 (成功)，則請勿讓部署失敗。設定為 true 以啟用。

範例程式碼片段

下列是範例程式碼片段。

```
container_commands:
  collectstatic:
    command: "django-admin.py collectstatic --noinput"
  01syncdb:
    command: "django-admin.py syncdb --noinput"
    leader_only: true
  02migrate:
    command: "django-admin.py migrate"
    leader_only: true
  99customize:
    command: "scripts/customize.sh"
```

範例：使用自訂 Amazon CloudWatch 指標

Amazon CloudWatch 是一種 Web 服務，可讓您監控、管理和發佈各種指標，以及根據指標中的資料設定警示動作。您可以為自己的使用定義自定義指標，Elastic Beanstalk 會將這些指標推送到 Amazon。 CloudWatch一旦 Amazon CloudWatch 包含您的自定義指標，您可以在 Amazon CloudWatch 控制台中查看這些指標。

> **Important**
>
> Amazon CloudWatch 監控腳本已被棄用。 CloudWatch 代理程式現在已取代 CloudWatch監控指令碼，以收集指標和記錄檔。
> 如果您仍在從已取代的監控指令碼遷移至代理程式，並需要監控指令碼的相關資訊，請參閱已取代：使用 Amazon EC2 使用者指南中的 CloudWatch 監控指令碼收集指標。

Amazon CloudWatch 代理

Amazon CloudWatch 代理程式可跨作業系統從 Amazon EC2 執行個體和現場部署伺服器收集 CloudWatch 指標和日誌。此代理程式支援在系統層級收集的指標，也支援從您的應用程式或服務收集自訂日誌和指標。如需 Amazon CloudWatch 代理程式的詳細資訊，請參閱 Amazon CloudWatch 使用者指南中的使用 CloudWatch 代理程式收集指標和日誌。

> **Note**
>
> Elastic Beanstalk 增強 Health 報告具有將各種執行個體和環境指標發佈到的原生支援。CloudWatch如需詳細資訊，請參閱 為環境發佈 Amazon CloudWatch 自訂指標。

主題

- .Ebextensions 組態檔案
- 許可
- 在 CloudWatch 主控台中檢視指標

.Ebextensions 組態檔案

此範例使用 .ebextensions 組態檔案中的檔案和命令，在 Amazon Linux 2 平台上設定和執行 Amazon CloudWatch 代理程式。此代理程式會與 Amazon Linux 2 預先封裝。如果您使用的是其他作業系統，則可能需要執行額外步驟來安裝代理程式。如需詳細資訊，請參閱 Amazon CloudWatch 使用者指南中的安裝 CloudWatch 代理程式。

若要使用此範例，請將該範例儲存至名為 `cloudwatch.config` 的檔案 (位於您專案目錄最上層名為 `.ebextensions` 的目錄中)，然後使用 Elastic Beanstalk 主控台 (在您的原始碼套件中加入 .ebextensions 目錄) 或 EB CLI 來部署您的應用程式。

如需關於組態檔案的詳細資訊，請參閱 使用組態檔案 (.ebextensions) 來進行進階的環境自訂。

.ebextensions/cloudwatch.config

```
files:
  "/opt/aws/amazon-cloudwatch-agent/bin/config.json":
    mode: "000600"
    owner: root
    group: root
    content: |
      {
        "agent": {
          "metrics_collection_interval": 60,
          "run_as_user": "root"
        },
        "metrics": {
          "namespace": "System/Linux",
```

```
          "append_dimensions": {
            "AutoScalingGroupName": "${aws:AutoScalingGroupName}"
          },
          "metrics_collected": {
            "mem": {
              "measurement": [
                "mem_used_percent"
              ]
            }
          }
        }
      }
container_commands:
  start_cloudwatch_agent:
    command: /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-
config -m ec2 -s -c file:/opt/aws/amazon-cloudwatch-agent/bin/config.json
```

此檔案包含兩個區段：

- `files` — 此區段會新增代理程式組態檔案。它指出了代理應該發送到 Amazon 的指標和日誌 CloudWatch。在此範例中，我們僅傳送 mem_used_percent 指標。如需 Amazon CloudWatch 代理程式支援的系統層級指標的完整清單，請參閱 Amazon CloudWatch 使用者指南中的 CloudWatch 代理程式收集的指標。
- `container_commands` — 此區段包含以參數形式傳遞組態檔案來啟動代理程式的命令。如需 `container_commands` 的詳細資訊，請參閱 容器命令。

許可

您環境中的執行個體需要適當的 IAM 許可，才能使用 Amazon CloudWatch 代理程式發佈自訂 Amazon CloudWatch 指標。您可以將許可加入環境的執行個體描述檔中，來授予您環境的執行個體。您可以在部署應用程式之前或之後，將權限加入執行個體描述檔。

授與發佈 CloudWatch 量度的權限

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. 選擇您的環境的執行個體描述檔角色。在預設情況下，當您使用 Elastic Beanstalk 主控台或 EB CLI 來建立環境時，這會是 `aws-elasticbeanstalk-ec2-role`。
4. 選擇許可索引標籤標籤。

5. 在 Permissions Policies (許可政策) 下的 Permissions (許可) 區段中，選擇 Attach policies (連接政策)。
6. 在 [附加權限] 下，選擇受 AWS 管理的策略CloudWatchAgentServerPolicy。然後，按一下 Attach Policy (連接政策)。

如需管理政策的詳細資訊，請參閱《IAM 使用者指南》中的使用政策。

在 CloudWatch 主控台中檢視指標

將 CloudWatch 組態檔案部署到您的環境後，請檢查 Amazon CloudWatch 主控台以檢視您的指標。自訂指標將位於 CWAgent 命名空間。

如需詳細資訊，請參閱 Amazon CloudWatch 使用者指南中的檢視可用指標。

在 Windows Server 上自訂軟體

您可能想要針對您的應用程式相依使用的軟體，來進行自訂和設定。這些檔案可能是應用程式所必須用到的，例如，需執行的其他套件或服務。如需自訂和設定 Elastic Beanstalk 環境的一般資訊，請參閱設定 Elastic Beanstalk 環境。

> **Note**
>
> YAML 憑藉一致的縮排。請在取代範例組態檔中的內容時，讓縮排層級一致，並確認您的文字編輯器使用空格而非定位字元進行縮排。

組態檔案支援下列金鑰，這些金鑰會對執行您應用程式的 Windows 伺服器造成影響。

金鑰

- 套件
- Sources
- 檔案
- 命令
- 服務
- 容器命令

這些金鑰會以此列的順序處理。

> **Note**
>
> 較舊 (未進行版本控制) 的 .NET 平台版本不會以正確的順序處理組態檔案。如需進一步了解，請參閱遷移 Elastic Beanstalk Windows Server 平台的主要版本。

觀看您環境的事件的同時，開發與測試組態檔案。Elastic Beanstalk 忽略組態檔案，其中包含驗證錯誤，例如無效的金鑰，並不處理同一個檔案其他任何金鑰。當發生這種情況，Elastic Beanstalk 會將警告事件新增到事件日誌。

套件

使用 `packages` 金鑰來下載和安裝預先整裝好的應用程式與元件。

在 Windows 環境中，Elastic Beanstalk 支援下載和安裝 MSI 套件。(Linux 環境支援其他套件管理工具。如需詳細資訊，請參閱在 Linux 伺服器上自訂軟體頁面的套件)。

只要 URL 是可公開存取，您就可以參考任何外部位置 (例如，Amazon Simple Storage Service (Amazon S3) 物件)。

如果您指定多個 `msi:` 套件，則不保證其安裝順序。

語法

指定您的選擇名稱做為套件名稱，並以 MSI 檔案位置的 URL 做為值。您可以在 `msi:` 金鑰下指定多個套件。

```
packages:
  msi:
    package name: package url
    ...
```

範例

下列範例指定了從 `https://dev.mysql.com/` 下載 mysql (mysql) 的 URL。

```
packages:
  msi:
    mysql: https://dev.mysql.com/get/Downloads/Connector-Net/mysql-connector-
net-8.0.11.msi
```

以下範例指定 Amazon S3 物件做為 MSI 檔案位置。

```
packages:
  msi:
    mymsi: https://mybucket.s3.amazonaws.com/myobject.msi
```

Sources

使用 `sources` 金鑰，從公開 URL 下載封存檔案，並在 EC2 執行個體上的目標目錄中，將該檔案解壓縮。

語法

```
sources:
  target directory: location of archive file
```

支援的格式

在 Windows 環境中，Elastic Beanstalk 支援 .zip 格式。(Linux 環境支援額外的格式。 如需詳細資訊，請參閱來源在在 Linux 伺服器上自訂軟體((() 頁面。)

只要 URL 是可公開存取，您就可以參考任何外部位置 (例如，Amazon Simple Storage Service (Amazon S3) 物件)。

範例

下列範例從 Amazon S3 儲存貯體下載了公有的 .zip 檔案，並將該檔案解壓縮至 `c:/myproject/myapp`。

```
sources:
  "c:/myproject/myapp": https://mybucket.s3.amazonaws.com/myobject.zip
```

檔案

使用 `files` 金鑰，在 EC2 執行個體上建立檔案。此內容可以內嵌於組態檔案，或從 URL 取得。檔案會依詞典編纂順序寫入磁碟。若要從 Amazon S3 下載私有檔案，請提供授權用的執行個體描述檔。

語法

```
files:
```

```
  "target file location on disk":
    source: URL
    authentication: authentication name:

  "target file location on disk":
    content: |
      this is my content
    encoding: encoding format
```

選項

`content`

(選用) 字串。

`source`

(選用) 載入該檔案的 URL。您無法使用 content 金鑰來指定此選項。

`encoding`

(選用) 編碼格式。此選項只用於提供內容金鑰值。預設值為 `plain`。

有效值：`plain` | `base64`

`authentication`

(選用) 所要使用的 AWS CloudFormation 身份驗證方法名稱。您可以利用 Resources (資源) 金鑰，在 Auto Scaling 群組的中繼資料中新增身份驗證方法。

範例

以下範例顯示兩種方式提供檔案內容：從 URL，或是內嵌在組態檔案。

```
files:
  "c:\\targetdirectory\\targetfile.txt":
    source: http://foo.bar/myfile

  "c:/targetdirectory/targetfile.txt":
    content: |
      # this is my file
      # with content
```

> **Note**
>
> 如果在檔案路徑中使用反斜線 (\)，則必須在 \ 前面冠上另一個反斜線 (跳脫字元)，如先前的範例所示。

下列範例使用了 Resources (資源) 金鑰來新增名為 S3Auth 的身份驗證方法，並使用此方法，從 Amazon S3 儲存貯體下載私有檔案。

```
files:
  "c:\\targetdirectory\\targetfile.zip":
    source: https://elasticbeanstalk-us-east-2-123456789012.s3.amazonaws.com/prefix/
myfile.zip
    authentication: S3Auth

Resources:
  AWSEBAutoScalingGroup:
    Metadata:
      AWS::CloudFormation::Authentication:
        S3Auth:
          type: "s3"
          buckets: ["elasticbeanstalk-us-east-2-123456789012"]
          roleName:
            "Fn::GetOptionSetting":
              Namespace: "aws:autoscaling:launchconfiguration"
              OptionName: "IamInstanceProfile"
              DefaultValue: "aws-elasticbeanstalk-ec2-role"
```

命令

使用 commands 金鑰，在 EC2 執行個體上執行命令。命令會依其名稱的字母順序處理，而且會在應用程式和 Web 伺服器設定完成及應用程式版本檔案解壓縮前，就先行執行。

指定的命令會以管理員使用者的身分執行。

若要對命令的問題進行疑難排解，您可以在執行個體日誌中尋找命令的輸出。

語法

```
commands:
  command name:
```

```
    command: command to run
```

選項

`command`

以陣列或字串來指定要執行的命令。如果使用陣列，則不需跳脫空格字元或用引號括起命令參數。

`cwd`

(選用) 工作目錄。依預設，Elastic Beanstalk 會嘗試尋找專案的目錄位置。如果找不到，則會使用 `c:\Windows\System32` 做為預設位置。

`env`

(選用) 設定命令用的環境變數。此屬性會進行覆寫，而非附加至現有的環境。

`ignoreErrors`

(選用) 一個布林值，在 `command` 金鑰中所包含的命令失敗時 (傳回非零的值)，可用來判定其他的命令是否應該執行。如果即使指令失敗，也想繼續執行指令，請將此值設定為 `true`。如果要在命令失敗時停止執行命令，請將此值設定為 `false`。預設值為 `false`。

`test`

(選用) 此命令必須傳回值 `true` (結束代碼 0)，Elastic Beanstalk 才能處理 `command` 金鑰中包含的命令。

`waitAfterCompletion`

(選用) 等命令執行完成後再執行下一個命令前的等待秒數。如果系統需要在命令完成後重新啟動，則系統會在指定的秒數時間經過之後才重新啟動。如果系統因為命令而重新啟動，則 Elastic Beanstalk 將會回復到組態檔案中該項命令之後的位置點。預設值為 **60** 秒。您也可以指定 **forever**，但系統必須先重新啟動，您才可以執行另一個命令。

範例

下列範例會將 `set` 命令的輸出結果儲存到指定的檔案。如果有後續的命令，則 Elastic Beanstalk 會在此命令完成後立即執行該後續命令。如果此命令需要重新啟動，則 Elastic Beanstalk 會在命令完成後立即重新啟動執行個體。

```
commands:
  test:
    command: set > c:\\myapp\\set.txt
```

```
    waitAfterCompletion: 0
```

服務

使用 services 金鑰，來定義執行個體啟動時應啟動或停止的服務。services 金鑰也可讓您指定來源、套件與檔案的相依關係，因此如果安裝中的檔案需要重新啟動，則 Elastic Beanstalk 會負責重新啟動服務。

語法

```
services:
  windows:
    name of service:
      files:
        - "file name"
      sources:
        - "directory"
      packages:
        name of package manager:
            "package name[: version]"
      commands:
        - "name of command"
```

選項

`ensureRunning`

(選用) 設定為 `true`，以確保服務會在 Elastic Beanstalk 完成後執行。

設定為 `false`，以確保服務不會在 Elastic Beanstalk 完成後執行。

如果略過此金鑰，將不會改變服務狀態。

`enabled`

(選用) 設定為 `true`，來確保服務會在開機時自動啟動。

設定為 `false`，來確保服務不會在開機時自動啟動。

如果略過此金鑰，將不會改變此屬性。

`files`

檔案清單。如果 Elastic Beanstalk 透過檔案區塊直接變更檔案，服務會重新啟動。

`sources`

目錄清單。如果 Elastic Beanstalk 將封存解壓縮至這些目錄的其中之一，服務會重新啟動。

`packages`

套件軟體管理工具與套件名稱清單的對應圖。如果 Elastic Beanstalk 安裝或更新這些套件的其中之一，服務會重新啟動。

`commands`

命令名稱清單。如果 Elastic Beanstalk 執行指定的命令，服務會重新啟動。

範例

```
services:
  windows:
    myservice:
      enabled: true
      ensureRunning: true
```

容器命令

使用 `container_commands` 金鑰，來執行會影響您應用程式原始碼的命令。容器指令的執行會在應用程式和 Web 伺服器設定完成及應用程式版本封存檔解壓縮之後，但是在應用程式版本安裝之前。非容器的指令及其他自訂操作，則會在應用程式的原始碼解壓縮之前執行。

容器指令會從暫存目錄執行，您的原始碼會在部署到應用程式伺服器之前，先解壓縮到該目錄。當原始碼部署到最終的位置時，您使用容器命令對暫存目錄中原始碼所進行的變更，也會納入部署。

若要對容器命令的問題進行疑難排解，您可以在執行個體日誌中尋找命令的輸出。

如果只要在單一執行個體上執行命令，請使用 `leader_only` 選項；或者，請將 `test` 設定為只在測試命令得出的評估值為 `true` 時，才執行命令。僅限領導者的容器指令，只會在環境建立與部署時執行；其他的指令與伺服器自訂操作，則會在每次佈建或更新執行個體時執行。僅限領導者的容器命令會因為啟動組態變更 (例如變更 AMI ID 或執行個體類型) 而不執行。

語法

```
container_commands:
```

```
  name of container_command:
    command: command to run
```

選項

command

要執行的字串或字串陣列。

env

(選用) 在執行命令前先設定環境變數，覆寫掉任何現有的值。

cwd

(選用) 工作目錄。根據預設，這是已經解壓縮應用程式的暫存目錄。

leader_only

(選用) 只在 Elastic Beanstalk 所選擇的單一執行個體上執行指令。僅限領導者的容器命令會先於其他容器命令執行。指令可以是只限領導者或具有 test，但只能是其中一種 (leader_only 會具有優先性)。

test

(選用) 執行測試命令，此命令必須傳回 true 才能執行容器命令。指令可以是只限領導者或具有 test，但只能是其中一種 (leader_only 會具有優先性)。

ignoreErrors

(選用) 如果容器命令傳回 0 以外的值 (成功)，則請勿讓部署失敗。設定為 true 以啟用。

waitAfterCompletion

(選用) 等命令執行完成後再執行下一個命令前的等待秒數。如果系統需要在命令完成後重新啟動，則系統會在指定的秒數時間經過之後才重新啟動。如果系統因為命令而重新啟動，則 Elastic Beanstalk 將會回復到組態檔案中該項命令之後的位置點。預設值為 **60** 秒。您也可以指定 **forever**，但系統必須先重新啟動，您才可以執行另一個命令。

範例

下列範例會將 set 命令的輸出結果儲存到指定的檔案。Elastic Beanstalk 會在單一執行個體上執行指令，然後在該指令完成時立即重新啟動執行個體。

```
container_commands:
  foo:
    command: set > c:\\myapp\\set.txt
    leader_only: true
    waitAfterCompletion: 0
```

新增和自訂 Elastic Beanstalk 環境資源

您可以自訂包含在您 Elastic Beanstalk 環境中的環境資源。例如，您可以新增 Amazon SQS 佇列和監控佇列深度的警示，或是新增 Amazon ElastiCache 叢集。在部署應用程式版本的同時，您可以在原始碼套件中加入組態檔案，來輕鬆自訂您的環境。

您可以在組態檔案中使用 `Resources` 金鑰，來建立和自訂您環境中的 AWS 資源。在組態檔案中所定義的資源，會加進用來啟動您環境的 AWS CloudFormation 範本中。所有的 AWS CloudFormation 資源類型皆可使用。

> **Note**
>
> 每當您新增不是由 Elastic Beanstalk 管理的資源時，請務必將具有適當許可的使用者政策新增給 AWS Identity and Access Management (IAM) 使用者。Elastic Beanstalk 僅提供 Elastic Beanstalk 受管資源的許可的受管使用者政策。

例如，下列的組態檔案在 Elastic Beanstalk 所建立的預設 Auto Scaling 群組中，加入了 Auto Scaling 生命週期關聯：

~/my-app/.ebextensions/as-hook.config

```
Resources:
  hookrole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument: {
               "Version" : "2012-10-17",
               "Statement": [ {
                  "Effect": "Allow",
                  "Principal": {
                     "Service": [ "autoscaling.amazonaws.com" ]
                  },
                  "Action": [ "sts:AssumeRole" ]
```

```
              } ]
           }
      Policies: [ {
              "PolicyName": "SNS",
              "PolicyDocument": {
                    "Version": "2012-10-17",
                    "Statement": [{
                        "Effect": "Allow",
                        "Resource": "*",
                        "Action": [
                            "sqs:SendMessage",
                            "sqs:GetQueueUrl",
                            "sns:Publish"
                        ]
                      }
                    ]
                }
              } ]
  hooktopic:
    Type: AWS::SNS::Topic
    Properties:
      Subscription:
        - Endpoint: "my-email@example.com"
          Protocol: email
  lifecyclehook:
    Type: AWS::AutoScaling::LifecycleHook
    Properties:
      AutoScalingGroupName: { "Ref" : "AWSEBAutoScalingGroup" }
      LifecycleTransition: autoscaling:EC2_INSTANCE_TERMINATING
      NotificationTargetARN: { "Ref" : "hooktopic" }
      RoleARN: { "Fn::GetAtt" : [ "hookrole", "Arn"] }
```

此範例定義了三項資源：hookrole、hooktopic 與 lifecyclehook。前兩項資源是 IAM 角色 (此角色會授予許可給 Amazon EC2 Auto Scaling，以發佈訊息至 Amazon SNS) 和 SNS 主題 (會將來自 Auto Scaling 群組的訊息轉傳到電子郵件地址)。Elastic Beanstalk 會使用指定的屬性和類型來建立這些資源。

最後一項資源 lifecyclehook 是生命週期關聯本身：

```
  lifecyclehook:
    Type: AWS::AutoScaling::LifecycleHook
    Properties:
      AutoScalingGroupName: { "Ref" : "AWSEBAutoScalingGroup" }
```

```
      LifecycleTransition: autoscaling:EC2_INSTANCE_TERMINATING
      NotificationTargetARN: { "Ref" : "hooktopic" }
      RoleARN: { "Fn::GetAtt" : [ "hookrole", "Arn"] }
```

生命週期關聯定義會使用兩個函數來填入關聯屬性的值。`{ "Ref" : "AWSEBAutoScalingGroup" }` 會擷取 Elastic Beanstalk 為環境所建立 Auto Scaling 群組的名稱。`AWSEBAutoScalingGroup` 是 Elastic Beanstalk 所提供的標準資源名稱之一。

針對 `AWS::IAM::Role`，`Ref` 只會傳回角色的名稱，而不會傳回 ARN。若要取得 `RoleARN` 參數的 ARN，您可以改而使用另一項內部函數 (`Fn::GetAtt`)，這項功能可取得資源的任何屬性。`RoleARN: { "Fn::GetAtt" : [ "hookrole", "Arn"] }` 從 `Arn` 資源取得 `hookrole` 屬性。

`{ "Ref" : "hooktopic" }` 會針對先前在組態檔案中所建立的 Amazon SNS 主題，取得其 ARN。`Ref` 所傳回的值，會隨資源類型而有不同，在 AWS CloudFormation 使用者指南的 AWS::SNS::Topic 資源類型主題中，提供了這些傳回值的相關說明。

修改 Elastic Beanstalk 為您環境建立的資源

Elastic Beanstalk 為您環境建立的資源具有名稱。您可以透過函數使用這些名稱以取得資源的資訊，或者修改資源屬性來自訂其行為。本主題說明 Elastic Beanstalk 在不同類型的環境中使用的 AWS 資源。

> **Note**
>
> 先前的主題自訂資源提供自訂環境資源的一些使用案例和範例。您也可以在稍後的主題 自訂資源範例 中找到組態檔案的更多範例。

Web 伺服器環境具有下列資源。

Web 伺服器環境

- `AWSEBAutoScalingGroup` (AWS::AutoScaling::AutoScalingGroup) – 連接至您環境的 Auto Scaling 群組。
- 下列兩項資源的其中一項。
  - `AWSEBAutoScalingLaunchConfiguration` (AWS::AutoScaling::LaunchConfiguration) – 連接至您環境 Auto Scaling 群組的啟動組態。
  - `AWSEBEC2LaunchTemplate` (AWS::EC2::LaunchTemplate) – 您環境的 Auto Scaling 群組所使用的 Amazon EC2 啟動範本。

> **Note**
>
> 如果您的環境使用需要 Amazon EC2 啟動範本的功能，而使用者政策缺少所需的許可，環境建立或更新作業可能會失敗。使用 AdministratorAccess-AWSElasticBeanstalk 受管使用者政策，或將所需的許可新增到自訂政策。

- `AWSEBEnvironmentName` (AWS::ElasticBeanstalk::Environment) – 您的環境。
- `AWSEBSecurityGroup` (AWS::EC2::SecurityGroup) – 連接至您 Auto Scaling 群組的安全群組。
- `AWSEBRDSDatabase` (AWS::RDS::DBInstance) – 連接至您環境的 Amazon RDS 資料庫執行個體 (如適用)。

在負載平衡的環境中，您可以存取與負載平衡器相關的其他資源。Classic Load Balancer 有一個資源用於負載平衡器，及一個資源用於其連接的安全群組。應用程式和網路負載平衡器還有其他資源，可用於負載平衡器的預設接聽程式、接聽程式規則和目標群組。

負載平衡環境

- `AWSEBLoadBalancer` (AWS::ElasticLoadBalancing::LoadBalancer) – 您環境的 classic load balancer。
- `AWSEBV2LoadBalancer` (AWS::ElasticLoadBalancingV2::LoadBalancer) – 您環境的應用程式或 Network Load Balancer。
- `AWSEBLoadBalancerSecurityGroup` (AWS::EC2::SecurityGroup) – Elastic Beanstalk 為負載平衡器建立的安全群組名稱，僅適用 Amazon Virtual Private Cloud (Amazon VPC) 中。在預設 VPC 或 EC2 classic 中，Elastic Load Balancing 會將預設安全群組指派至負載平衡器。
- `AWSEBV2LoadBalancerListener` (AWS::ElasticLoadBalancingV2::Listener) – 允許負載平衡器檢查連線請求並將其轉送至一或多個目標群組的接聽程式。
- `AWSEBV2LoadBalancerListenerRule` (AWS::ElasticLoadBalancingV2::ListenerRule) – Elastic Load Balancing 接聽程式須採取動作的請求及其可採取的動作。
- `AWSEBV2LoadBalancerTargetGroup` (AWS::ElasticLoadBalancingV2::TargetGroup) – 將請求路由至一或多個登錄目標 (例如 Amazon EC2 執行個體) 的 Elastic Load Balancing 目標群組。

工作者環境具備用於 SQS 佇列的資源，可藉此緩衝傳入請求，也有一個執行個體可用於領導者選擇的 Amazon DynamoDB 資料表。

工作者環境

- `AWSEBWorkerQueue` (AWS::SQS::Queue) – 精靈可從中提取需處理請求的 Amazon SQS 佇列。
- `AWSEBWorkerDeadLetterQueue` (AWS::SQS::Queue) – 用於存放無法交付或精靈未成功處理訊息的 Amazon SQS 佇列。
- `AWSEBWorkerCronLeaderRegistry` (AWS::DynamoDB::Table) – Amazon DynamoDB 資料表，為精靈針對定期任務所使用的內部登錄。

其他 AWS CloudFormation 模板鍵

我們已經引入了配置文件密鑰， AWS CloudFormation 例如 `Resources`files，，和packages。Elastic Beanstalk 會將組態檔案的內容新增至支援您環境的 AWS CloudFormation 範本，因此您可以使用其他 AWS CloudFormation 區段在組態檔案中執行進階工作。

鍵

- 參數
- 輸出
- 映射項目

參數

參數可替代 Elastic Beanstalk 自己的自訂選項，可用於定義您在組態檔案他處使用的值。如同自訂選項，您可使用參數在單一位置收集使用者可設定的值。與自定義選項不同，您不能使用 Elastic Beanstalk 的 API 來設置參數值，並且您可以在模板中定義的參數數量受到限制。 AWS CloudFormation

您可能想要使用參數的一個原因是將配置文件作為 AWS CloudFormation 模板加倍。如果您使用參數而不是自訂選項，則可以使用組態檔案在其自己的堆疊中 AWS CloudFormation 建立相同的資源。例如，您的組態檔案可將 Amazon EFS 檔案系統新增至您的環境進行測試，然後使用相同檔案來建立未繫結至您環境生命週期的獨立檔案系統，供生產使用。

下列範例說明如何使用參數在組態檔案上方收集使用者可設定的值。

Example Loadbalancer-accesslogs-existingbucket. 配置-參數

```
Parameters:
  bucket:
```

```
    Type: String
    Description: "Name of the Amazon S3 bucket in which to store load balancer logs"
    Default: "DOC-EXAMPLE-BUCKET"
  bucketprefix:
    Type: String
    Description: "Optional prefix. Can't start or end with a /, or contain the word
 AWSLogs"
    Default: ""
```

輸出

您可使用 Outputs 區塊將所建立資源的相關資訊匯出至 AWS CloudFormation。然後，您可以使用該Fn::ImportValue函數將值拉入 Elastic Beanstalk 之外的 AWS CloudFormation 模板中。

下列範例會建立 Amazon SNS 主題，並 AWS CloudFormation 使用名稱NotificationTopicArn將其 ARN 匯出至。

Example sns-topic.config

```
Resources:
  NotificationTopic:
    Type: AWS::SNS::Topic

Outputs:
  NotificationTopicArn:
    Description: Notification topic ARN
    Value: { "Ref" : "NotificationTopic" }
    Export:
      Name: NotificationTopicArn
```

在不同環境的配置文件或 Elastic Beanstalk 之外的 AWS CloudFormation 模板中，您可以使用該Fn::ImportValue函數來獲取導出的 ARN。本範例將匯出的值指派至名為 TOPIC_ARN 的環境屬性。

Example env.config

```
option_settings:
  aws:elasticbeanstalk:application:environment:
    TOPIC_ARN: '`{ "Fn::ImportValue" : "NotificationTopicArn" }`'
```

映射項目

您可使用對應依據命名空間來存放金鑰值對。對應可助您整理您於組態中使用的值，或根據其他值來變更參數值。例如，下列組態會根據目前區域設定帳戶 ID 參數的值。

Example Loadbalancer-accesslogs-newbucket. 配置-映射

```
Mappings:
  Region2ELBAccountId:
    us-east-1:
      AccountId: "111122223333"
    us-west-2:
      AccountId: "444455556666"
    us-west-1:
      AccountId: "123456789012"
    eu-west-1:
      AccountId: "777788889999"
...
            Principal:
              AWS:
                ? "Fn::FindInMap"
                :
                  - Region2ELBAccountId
                  -
                    Ref: "AWS::Region"
                  - AccountId
```

函數

針對來自其他資源或來自 Elastic Beanstalk 組態選項設定之資訊的資源屬性，您可使用組態檔案中的函數產生其值。Elastic Beanstalk 支援 AWS CloudFormation 函數 (`Ref`、`Fn::GetAtt`、`Fn::Join`)，以及 Elastic Beanstalk 專用函數 `Fn::GetOptionSetting`。

函數

- Ref
- Fn::GetAtt
- Fn::Join
- Fn::GetOptionSetting

Ref

使用 Ref 來擷取 AWS 資源的預設字串顯示方式。Ref 回傳的值取決於資源類型，有時則視其他因素而異。例如，安全群組 (AWS::EC2::SecurityGroup) 會傳回安全群組的名稱或 ID，取決於安全群組是在預設的 Amazon Virtual Private Cloud (Amazon VPC)、EC2 classic 或在自訂 VPC 中。

```
{ "Ref" : "resource name" }
```

> **Note**
>
> 如需各個資源類型的詳細資訊 (包括 Ref 的傳回值)，請參閱《AWS CloudFormation 使用者指南》中的 AWS 資源類型參考。

在範例 Auto Scaling 生命週期關聯中：

```
Resources:
  lifecyclehook:
    Type: AWS::AutoScaling::LifecycleHook
    Properties:
      AutoScalingGroupName: { "Ref" : "AWSEBAutoScalingGroup" }
```

您亦可使用 Ref 來擷取相同檔案他處或不同組態檔案定義的 AWS CloudFormation 參數值。

Fn::GetAtt

使用 Fn::GetAtt 來擷取 AWS 資源的屬性值。

```
{ "Fn::GetAtt" : [ "resource name", "attribute name"] }
```

在範例 Auto Scaling 生命週期關聯中：

```
Resources:
  lifecyclehook:
    Type: AWS::AutoScaling::LifecycleHook
    Properties:
      RoleARN: { "Fn::GetAtt" : [ "hookrole", "Arn"] }
```

如需詳細資訊，請參閱 Fn::GetAtt。

Fn::Join

使用 Fn::Join 來結合帶有分隔符號的字串。字串可為硬式編碼，或使用 Fn::GetAtt 或 Ref 的輸出。

```
{ "Fn::Join" : [ "delimiter", [ "string1", "string2" ] ] }
```

如需詳細資訊，請參閱 Fn::Join。

Fn::GetOptionSetting

使用 Fn::GetOptionSetting 來擷取環境套用之組態選項設定的值。

```
"Fn::GetOptionSetting":
  Namespace: "namespace"
  OptionName: "option name"
  DefaultValue: "default value"
```

在存放私密金鑰範例中：

```
Resources:
  AWSEBAutoScalingGroup:
    Metadata:
      AWS::CloudFormation::Authentication:
        S3Auth:
          type: "s3"
          buckets: ["elasticbeanstalk-us-west-2-123456789012"]
          roleName:
            "Fn::GetOptionSetting":
              Namespace: "aws:autoscaling:launchconfiguration"
              OptionName: "IamInstanceProfile"
              DefaultValue: "aws-elasticbeanstalk-ec2-role"
```

自訂資源範例

下列是範例組態檔案的清單，您可以利用這些檔案來自訂您的 Elastic Beanstalk 環境：

- DynamoDB、CloudWatch 和 SNS
- Elastic Load Balancing 與 CloudWatch
- ElastiCache

- RDS 與 CloudWatch
- SQS、SNS 與 CloudWatch

此頁面的子主題會提供一些延伸範例，說明如何在 Elastic Beanstalk 環境中新增和設定自訂資源。

範例

- 範例：ElastiCache
- 範例：SQS、CloudWatch 和 SNS
- 範例：DynamoDB、CloudWatch 和 SNS

範例：ElastiCache

下列範例將 Amazon ElastiCache 叢集新增至 EC2-Classic 和 EC2-VPC (預設和自訂 Amazon Virtual Private Cloud (Amazon VPC)) 平台。如需這些平台的詳細資訊，並進一步了解如何判斷哪些 EC2 支援您的區域與您的 AWS 帳戶，請參閱https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-supported-platforms.html。之後，請參考本主題下適用您的平台的章節。

- EC2-Classic 平台
- EC2-VPC (預設)
- EC2-VPC (自訂)

EC2-Classic 平台

本範例將 Amazon ElastiCache 叢集新增至具備 EC2-Classic 平台所啟動的執行個體的環境。本範例列出的所有屬性都是最低要求的屬性，各個資源類型均必須設定。您可以在 ElastiCache 範例下載範例。

> **Note**
>
> 本範例會建立 AWS 資源，您可能需要為其支付費用。如需 AWS 定價的詳細資訊，請參閱 https://aws.amazon.com/pricing/。某些服務屬於 AWS 免費用量方案。若您是新客戶，可以免費試用這些服務。如需詳細資訊，請參閱 https://aws.amazon.com/free/。

要使用此範例，請依照下列項目：

1. 在原始碼套件的最上層目錄建立 `.ebextensions` 目錄。

2. 以 .config 延伸建立兩個組態檔案，並且置於 .ebextensions 目錄。一個組態檔案定義資源，另一個組態檔案定義選項。
3. 將您的應用程式部署至 Elastic Beanstalk。

   YAML 憑藉一致的縮排。請在取代範例組態檔中的內容時，讓縮排層級一致，並確認您的文字編輯器使用空格而非定位字元進行縮排。

建立定義資源的組態檔案 (如 elasticache.config)。在此範例中，我們指定 ElastiCache 叢集資源的名稱 (MyElastiCache)、宣告其類型，並設定叢集屬性，藉此建立 ElastiCache 叢集。本範例參考的 ElastiCache 安全群組資源名稱，均由此組態檔案建立並定義。接著，我們建立 ElastiCache 安全群組。我們定義資源名稱、宣告其類型，並新增安全群組的說明。最後，我們設定 ElastiCache 安全群組的輸入規則，僅允許 ElastiCache 安全群組 (MyCacheSecurityGroup) 和 Elastic Beanstalk 安全群組 (AWSEBSecurityGroup) 內的執行個體進行存取。參數名稱 AWSEBSecurityGroup 則是固定的資源名稱，由 Elastic Beanstalk 所提供。您必須將 AWSEBSecurityGroup 新增至您的 ElastiCache 安全群組輸入規則，如此一來，您的 Elastic Beanstalk 應用程式即可連接至您 ElastiCache 叢集內的執行個體。)

```
#This sample requires you to create a separate configuration file that defines the
 custom option settings for CacheCluster properties.

Resources:
  MyElastiCache:
    Type: AWS::ElastiCache::CacheCluster
    Properties:
      CacheNodeType:
         Fn::GetOptionSetting:
             OptionName : CacheNodeType
             DefaultValue: cache.m1.small
      NumCacheNodes:
           Fn::GetOptionSetting:
             OptionName : NumCacheNodes
             DefaultValue: 1
      Engine:
           Fn::GetOptionSetting:
             OptionName : Engine
             DefaultValue: memcached
      CacheSecurityGroupNames:
        - Ref: MyCacheSecurityGroup
  MyCacheSecurityGroup:
    Type: AWS::ElastiCache::SecurityGroup
```

```
    Properties:
      Description: "Lock cache down to webserver access only"
  MyCacheSecurityGroupIngress:
    Type: AWS::ElastiCache::SecurityGroupIngress
    Properties:
      CacheSecurityGroupName:
        Ref: MyCacheSecurityGroup
      EC2SecurityGroupName:
        Ref: AWSEBSecurityGroup
```

如需本範例組態檔案所使用的資源的詳細資訊，請參閱下列參考：

- AWS::ElastiCache::CacheCluster
- AWS::ElastiCache::SecurityGroup
- AWS::ElastiCache:SecurityGroupIngress

建立另一個名為 options.config 的組態檔案，並定義自訂選項設定。

```
option_settings:
  "aws:elasticbeanstalk:customoption":
     CacheNodeType : cache.m1.small
     NumCacheNodes : 1
     Engine : memcached
```

這些行會指示 Elastic Beanstalk 自組態檔案 (本範例為 options.config) 的CacheNodeType、NumCacheNodes 和 Engine (引擎) 值，取得 CacheNodeType、NumCacheNodes 和 Engine (引擎) 屬性的值，其中的 option_settings 區段具備 aws:elasticbeanstalk:customoption 區段，內含的名稱-值對帶有該使用的實際值。上述範例表示這些值為 cache.m1.small、1 和 memcached。如需有關 Fn::GetOptionSetting 的詳細資訊，請參閱 函數。

EC2-VPC (預設)

本範例將 Amazon ElastiCache 叢集新增至具備 EC2-VPC 平台所啟動的執行個體的環境。具體而言，本章節的資訊適用 EC2 於預設 VPC 啟動執行個體的情境。本範例的所有屬性都是最低要求的屬性，各個資源類型均必須設定。如需預設 VPC 的詳細資訊，請參閱您的預設 VPC 與子網路相關文章。

> **Note**
>
> 本範例會建立 AWS 資源，您可能需要為其支付費用。如需 AWS 定價的詳細資訊，請參閱 https://aws.amazon.com/pricing/。某些服務屬於 AWS 免費用量方案。若您是新客戶，可以免費試用這些服務。如需詳細資訊，請參閱 https://aws.amazon.com/free/。

要使用此範例，請依照下列項目：

1. 在原始碼套件的最上層目錄建立 `.ebextensions` 目錄。
2. 以 `.config` 延伸建立兩個組態檔案，並且置於 `.ebextensions` 目錄。一個組態檔案定義資源，另一個組態檔案定義選項。
3. 將您的應用程式部署至 Elastic Beanstalk。

   YAML 憑藉一致的縮排。請在取代範例組態檔中的內容時，讓縮排層級一致，並確認您的文字編輯器使用空格而非定位字元進行縮排。

現請為資源組態檔案 `elasticache.config` 命名。本範例指定 ElastiCache 叢集資源的名稱 (`MyElastiCache`)、宣告其類型，並設定叢集屬性，藉此建立 ElastiCache 叢集。本範例參考的安全群組資源 ID，均由我們於此組態檔案中建立並定義。

接著，我們建立 EC2 安全群組。我們定義此資源的名稱、宣告其類型、新增說明，並設定安全群組的輸入規則，僅允許 Elastic Beanstalk 安全群組 (`AWSEBSecurityGroup`) 內的執行個體進行存取。(參數名稱 `AWSEBSecurityGroup` 是 Elastic Beanstalk 提供的固定資源名稱。您必須將 `AWSEBSecurityGroup` 新增至您的 ElastiCache 安全群組輸入規則，如此一來，您的 Elastic Beanstalk 應用程式即可連接至您 ElastiCache 叢集內的執行個體)。

EC2 安全群組的輸入規則亦定義快取節點可接受連線的 IP 通訊協定和連接埠號碼。以 Redis 而言，預設連接埠號碼為 6379。

```
#This sample requires you to create a separate configuration file that defines the
 custom option settings for CacheCluster properties.

Resources:
  MyCacheSecurityGroup:
    Type: "AWS::EC2::SecurityGroup"
    Properties:
      GroupDescription: "Lock cache down to webserver access only"
      SecurityGroupIngress :
```

```
        - IpProtocol : "tcp"
          FromPort :
            Fn::GetOptionSetting:
              OptionName : "CachePort"
              DefaultValue: "6379"
          ToPort :
            Fn::GetOptionSetting:
              OptionName : "CachePort"
              DefaultValue: "6379"
          SourceSecurityGroupName:
            Ref: "AWSEBSecurityGroup"
  MyElastiCache:
    Type: "AWS::ElastiCache::CacheCluster"
    Properties:
      CacheNodeType:
        Fn::GetOptionSetting:
          OptionName : "CacheNodeType"
          DefaultValue : "cache.t2.micro"
      NumCacheNodes:
        Fn::GetOptionSetting:
          OptionName : "NumCacheNodes"
          DefaultValue : "1"
      Engine:
        Fn::GetOptionSetting:
          OptionName : "Engine"
          DefaultValue : "redis"
      VpcSecurityGroupIds:
        -
          Fn::GetAtt:
            - MyCacheSecurityGroup
            - GroupId

Outputs:
  ElastiCache:
    Description : "ID of ElastiCache Cache Cluster with Redis Engine"
    Value :
      Ref : "MyElastiCache"
```

如需本範例組態檔案所使用的資源的詳細資訊，請參閱下列參考：

- AWS::ElastiCache::CacheCluster
- AWS::EC2::SecurityGroup

接著，請為選項組態檔案 options.config 命名，並定義自訂選項設定。

```
option_settings:
  "aws:elasticbeanstalk:customoption":
    CacheNodeType : cache.t2.micro
    NumCacheNodes : 1
    Engine : redis
    CachePort : 6379
```

這些行會指示 Elastic Beanstalk 自組態檔案 (本範例為 CacheNodeType) 中的 NumCacheNodes、Engine、CachePort 和 CacheNodeType 值，取得 NumCacheNodes、Engine、CachePort 和 options.config 屬性的值。該檔案具備 aws:elasticbeanstalk:customoption 區段 (option_settings 之下)，內含的名稱/值對帶有該使用的實際值。上述範例的這些值為 cache.t2.micro、1、redis 和 6379。如需有關 Fn::GetOptionSetting 的詳細資訊，請參閱 函數。

EC2-VPC (自訂)

若您於 EC2-VPC 平台建立自訂 VPC，並將其指定為 EC2 啟動執行個體的 VPC，則將 Amazon ElastiCache 叢集新增至您環境的程序，與預設 VPC 的程序不同。主要差異在於，您必須建立 ElastiCache 叢集的子網路群組。本範例的所有屬性都是最低要求的屬性，各個資源類型均必須設定。

> **Note**
>
> 本範例會建立 AWS 資源，您可能需要為其支付費用。如需 AWS 定價的詳細資訊，請參閱 https://aws.amazon.com/pricing/。某些服務屬於 AWS 免費用量方案。若您是新客戶，可以免費試用這些服務。如需詳細資訊，請參閱 https://aws.amazon.com/free/。

要使用此範例，請依照下列項目：

1. 在原始碼套件的最上層目錄建立 .ebextensions 目錄。
2. 以 .config 延伸建立兩個組態檔案，並且置於 .ebextensions 目錄。一個組態檔案定義資源，另一個組態檔案定義選項。
3. 將您的應用程式部署至 Elastic Beanstalk。

   YAML 憑藉一致的縮排。請在取代範例組態檔中的內容時，讓縮排層級一致，並確認您的文字編輯器使用空格而非定位字元進行縮排。

現請為資源組態檔案 `elasticache.config` 命名。本範例指定 ElastiCache 叢集資源的名稱 (`MyElastiCache`)、宣告其類型，並設定叢集屬性，藉此建立 ElastiCache 叢集。本範例屬性參考 ElastiCache 叢集的子網路群組名稱，以及由我們於此組態檔案建立並定義的安全群組資源 ID。

接著，我們建立 EC2 安全群組。我們定義此資源的名稱、宣告其類型、新增說明和 VPC ID，並設定安全群組的輸入規則，僅允許 Elastic Beanstalk 安全群組 (`AWSEBSecurityGroup`) 內的執行個體進行存取。(參數名稱 `AWSEBSecurityGroup` 是 Elastic Beanstalk 提供的固定資源名稱。您必須將 `AWSEBSecurityGroup` 新增至您的 ElastiCache 安全群組輸入規則，如此一來，您的 Elastic Beanstalk 應用程式即可連接至您 ElastiCache 叢集內的執行個體)。

EC2 安全群組的輸入規則亦定義快取節點可接受連線的 IP 通訊協定和連接埠號碼。以 Redis 而言，預設連接埠號碼為 6379。最後，本範例建立 ElastiCache 叢集的子網路群組。我們定義資源名稱、宣告其類型，並新增子網路群組的子網路說明和 ID。

> **Note**
>
> 我們建議您使用適用於 ElastiCache 叢集的私有子網路。如需 VPC 和私有子網路的詳細資訊，請參閱 https://docs.aws.amazon.com/vpc/latest/userguide/VPC_Scenario2.html。

```
#This sample requires you to create a separate configuration file that defines the
 custom option settings for CacheCluster properties.

Resources:
  MyElastiCache:
    Type: "AWS::ElastiCache::CacheCluster"
    Properties:
      CacheNodeType:
        Fn::GetOptionSetting:
          OptionName : "CacheNodeType"
          DefaultValue : "cache.t2.micro"
      NumCacheNodes:
        Fn::GetOptionSetting:
          OptionName : "NumCacheNodes"
          DefaultValue : "1"
      Engine:
        Fn::GetOptionSetting:
          OptionName : "Engine"
          DefaultValue : "redis"
      CacheSubnetGroupName:
        Ref: "MyCacheSubnets"
```

```
      VpcSecurityGroupIds:
        - Ref: "MyCacheSecurityGroup"
  MyCacheSecurityGroup:
    Type: "AWS::EC2::SecurityGroup"
    Properties:
      GroupDescription: "Lock cache down to webserver access only"
      VpcId:
        Fn::GetOptionSetting:
          OptionName : "VpcId"
      SecurityGroupIngress :
        - IpProtocol : "tcp"
          FromPort :
            Fn::GetOptionSetting:
              OptionName : "CachePort"
              DefaultValue: "6379"
          ToPort :
            Fn::GetOptionSetting:
              OptionName : "CachePort"
              DefaultValue: "6379"
          SourceSecurityGroupId:
            Ref: "AWSEBSecurityGroup"
  MyCacheSubnets:
    Type: "AWS::ElastiCache::SubnetGroup"
    Properties:
      Description: "Subnets for ElastiCache"
      SubnetIds:
        Fn::GetOptionSetting:
          OptionName : "CacheSubnets"
Outputs:
  ElastiCache:
    Description : "ID of ElastiCache Cache Cluster with Redis Engine"
    Value :
      Ref : "MyElastiCache"
```

如需本範例組態檔案所使用的資源的詳細資訊，請參閱下列參考：

- AWS::ElastiCache::CacheCluster
- AWS::EC2::SecurityGroup
- AWS::ElastiCache::SubnetGroup

接著，請為選項組態檔案 `options.config` 命名，並定義自訂選項設定。

> **Note**
>
> 在下列範例中，請使用您自己的子網路和 VPC，取代範例 CacheSubnets 和 VpcId 的值。

```
option_settings:
  "aws:elasticbeanstalk:customoption":
    CacheNodeType : cache.t2.micro
    NumCacheNodes : 1
    Engine : redis
    CachePort : 6379
    CacheSubnets:
      - subnet-1a1a1a1a
      - subnet-2b2b2b2b
      - subnet-3c3c3c3c
    VpcId: vpc-4d4d4d4d
```

這些行會指示 Elastic Beanstalk 自組態檔案 (本範例為 CacheNodeType) 中的 NumCacheNodes、Engine、CachePort、CacheSubnets、VpcId 和 CacheNodeType 值，取得 NumCacheNodes、Engine、CachePort、CacheSubnets、VpcId 和 options.config 屬性的值。該檔案具備 aws:elasticbeanstalk:customoption 區段 (option_settings 之下)，內含的名稱/值對帶有範例值。上述範例的這些值為 cache.t2.micro、1、redis、6379、subnet-1a1a1a1a、subnet-2b2b2b2b、subnet-3c3c3c3c 和 vpc-4d4d4d4d。如需有關 Fn::GetOptionSetting 的詳細資訊，請參閱 函數。

範例：SQS、CloudWatch 和 SNS

此範例將 Amazon SQS 佇列和佇列深度的警示新增到環境。在此範例中所顯示的屬性，是您必須針對這些資源的各個設定的最低要求屬性。您可從 SQS、SNS 和 CloudWatch 下載此範例。

> **Note**
>
> 本範例會建立 AWS 資源，您可能需要為其支付費用。如需 AWS 定價的詳細資訊，請參閱 https://aws.amazon.com/pricing/。某些服務屬於 AWS 免費用量方案。若您是新客戶，可以免費試用這些服務。如需詳細資訊，請參閱 https://aws.amazon.com/free/。

要使用此範例，請依照下列項目：

1. 在原始碼套件的最上層目錄建立 .ebextensions 目錄。

2. 以 .config 延伸建立兩個組態檔案，並且置於 .ebextensions 目錄。一個組態檔案定義資源，另一個組態檔案定義選項。
3. 將您的應用程式部署至 Elastic Beanstalk。

   YAML 憑藉一致的縮排。請在取代範例組態檔中的內容時，讓縮排層級一致，並確認您的文字編輯器使用空格而非定位字元進行縮排。

建立組態檔案 (例如 sqs.config)，此檔案是用來定義資源。在此範例中，我們會建立 SQS 佇列，並定義 VisbilityTimeout 資源中的 MySQSQueue 屬性。然後，我們會建立 SNS Topic，並指定在警示觸發時，傳送電子郵件到 someone@example.com。最後，我們會建立 CloudWatch 警示，此警示會在佇列增加到超過 10 筆訊息時觸發。在 Dimensions 屬性中，我們會指定維度的名稱，以及代表維度測量值的值。我們會使用 Fn::GetAtt，來從 QueueName 傳回 MySQSQueue 的值。

```
#This sample requires you to create a separate configuration file to define the custom
 options for the SNS topic and SQS queue.
Resources:
  MySQSQueue:
    Type: AWS::SQS::Queue
    Properties:
      VisibilityTimeout:
        Fn::GetOptionSetting:
          OptionName: VisibilityTimeout
          DefaultValue: 30
  AlarmTopic:
    Type: AWS::SNS::Topic
    Properties:
      Subscription:
        - Endpoint:
            Fn::GetOptionSetting:
              OptionName: AlarmEmail
              DefaultValue: "nobody@amazon.com"
          Protocol: email
  QueueDepthAlarm:
    Type: AWS::CloudWatch::Alarm
    Properties:
      AlarmDescription: "Alarm if queue depth grows beyond 10 messages"
      Namespace: "AWS/SQS"
      MetricName: ApproximateNumberOfMessagesVisible
      Dimensions:
        - Name: QueueName
          Value : { "Fn::GetAtt" : [ "MySQSQueue", "QueueName"] }
```

```
      Statistic: Sum
      Period: 300
      EvaluationPeriods: 1
      Threshold: 10
      ComparisonOperator: GreaterThanThreshold
      AlarmActions:
        - Ref: AlarmTopic
      InsufficientDataActions:
        - Ref: AlarmTopic

Outputs :
  QueueURL:
    Description : "URL of newly created SQS Queue"
    Value : { Ref : "MySQSQueue" }
  QueueARN :
    Description : "ARN of newly created SQS Queue"
    Value : { "Fn::GetAtt" : [ "MySQSQueue", "Arn"]}
  QueueName :
    Description : "Name newly created SQS Queue"
    Value : { "Fn::GetAtt" : [ "MySQSQueue", "QueueName"]}
```

如需本範例組態檔案所使用的資源的詳細資訊，請參閱下列參考：

- AWS::SQS::Queue
- AWS::SNS::Topic
- AWS::CloudWatch::Alarm

建立另一個名為 `options.config` 的組態檔案，並定義自訂選項設定。

```
option_settings:
  "aws:elasticbeanstalk:customoption":
     VisibilityTimeout : 30
     AlarmEmail : "nobody@example.com"
```

這幾行程式碼指示 Elastic Beanstalk 從組態檔案 (在我們的範例中為 options.config) 中的 VisibilityTimeout and Subscription Endpoint (可見性逾時與訂閱端點) 值，來取得 VisibilityTimeout and Subscription Endpoint (可見性逾時與訂閱端點) 屬性的值；該組態檔案包含了 option_settings 的區段，其中包括 aws:elasticbeanstalk:customoption 區段，此部分內含名稱-值的對組，而此對組中包含了實際要使用的值。在上列的範例中，這表示將會使用 30 和「nobody@amazon.com」來做為值。如需有關 `Fn::GetOptionSetting` 的詳細資訊，請參閱 the section called “函數”。

範例：DynamoDB、CloudWatch 和 SNS

此組態檔案使用適用於 PHP 的 AWS 開發套件 2，將 DynamoDB 資料表設定為以 PHP 為基礎的應用程式之工作階段處理常式。欲使用此範例，您必須具備 IAM 執行個體描述檔，本描述檔會新增至您環境的執行個體，並可用於存取 DynamoDB 資料表。

您可從 DynamoDB 工作階段支援範例下載本步驟要使用的範本。本範本內含下列檔案：

- 範例應用程式 `index.php`
- 組態檔案 `dynamodb.config` 用於建立並設定 DynamoDB 資料表和其他 AWS 資源，並於 EC2 執行個體上安裝軟體，以在 Elastic Beanstalk 環境中託管應用程式
- 組態檔案 `options.config`，會使用此特定安裝的設定來覆寫 `dynamodb.config` 的預設值

`index.php`

```
<?php

// Include the SDK using the Composer autoloader
require '../vendor/autoload.php';

use Aws\DynamoDb\DynamoDbClient;

// Grab the session table name and region from the configuration file
list($tableName, $region) = file(__DIR__ . '/../sessiontable');
$tableName = rtrim($tableName);
$region = rtrim($region);

// Create a DynamoDB client and register the table as the session handler
$dynamodb = DynamoDbClient::factory(array('region' => $region));
$handler = $dynamodb->registerSessionHandler(array('table_name' => $tableName,
 'hash_key' => 'username'));

// Grab the instance ID so we can display the EC2 instance that services the request
$instanceId = file_get_contents("http://169.254.169.254/latest/meta-data/instance-id");
?>
<h1>Elastic Beanstalk PHP Sessions Sample</h1>
<p>This sample application shows the integration of the Elastic Beanstalk PHP
container and the session support for DynamoDB from the AWS SDK for PHP 2.
Using DynamoDB session support, the application can be scaled out across
multiple web servers. For more details, see the
<a href="https://aws.amazon.com/php/">PHP Developer Center</a>.</p>
```

```
<form id="SimpleForm" name="SimpleForm" method="post" action="index.php">
<?php
echo 'Request serviced from instance ' . $instanceId . '<br/>';
echo '<br/>';

if (isset($_POST['continue'])) {
  session_start();
  $_SESSION['visits'] = $_SESSION['visits'] + 1;
  echo 'Welcome back ' . $_SESSION['username'] . '<br/>';
  echo 'This is visit number ' . $_SESSION['visits'] . '<br/>';
  session_write_close();
  echo '<br/>';
  echo '<input type="Submit" value="Refresh" name="continue" id="continue"/>';
  echo '<input type="Submit" value="Delete Session" name="killsession"
 id="killsession"/>';
} elseif (isset($_POST['killsession'])) {
  session_start();
  echo 'Goodbye ' . $_SESSION['username'] . '<br/>';
  session_destroy();
  echo 'Username: <input type="text" name="username" id="username" size="30"/><br/>';
  echo '<br/>';
  echo '<input type="Submit" value="New Session" name="newsession" id="newsession"/>';
} elseif (isset($_POST['newsession'])) {
  session_start();
  $_SESSION['username'] = $_POST['username'];
  $_SESSION['visits'] = 1;
  echo 'Welcome to a new session ' . $_SESSION['username'] . '<br/>';
  session_write_close();
  echo '<br/>';
  echo '<input type="Submit" value="Refresh" name="continue" id="continue"/>';
  echo '<input type="Submit" value="Delete Session" name="killsession"
 id="killsession"/>';
} else {
  echo 'To get started, enter a username.<br/>';
  echo '<br/>';
  echo 'Username: <input type="text" name="username" id="username" size="30"/><br/>';
  echo '<input type="Submit" value="New Session" name="newsession" id="newsession"/>';
}
?>
</form>
```

.ebextensions/dynamodb.config

```
Resources:
  SessionTable:
    Type: AWS::DynamoDB::Table
    Properties:
      KeySchema:
        HashKeyElement:
          AttributeName:
            Fn::GetOptionSetting:
              OptionName : SessionHashKeyName
              DefaultValue: "username"
          AttributeType:
            Fn::GetOptionSetting:
              OptionName : SessionHashKeyType
              DefaultValue: "S"
      ProvisionedThroughput:
        ReadCapacityUnits:
          Fn::GetOptionSetting:
            OptionName : SessionReadCapacityUnits
            DefaultValue: 1
        WriteCapacityUnits:
          Fn::GetOptionSetting:
            OptionName : SessionWriteCapacityUnits
            DefaultValue: 1

  SessionWriteCapacityUnitsLimit:
    Type: AWS::CloudWatch::Alarm
    Properties:
      AlarmDescription: { "Fn::Join" : ["", [{ "Ref" : "AWSEBEnvironmentName" }, "
 write capacity limit on the session table." ]]}
      Namespace: "AWS/DynamoDB"
      MetricName: ConsumedWriteCapacityUnits
      Dimensions:
        - Name: TableName
          Value: { "Ref" : "SessionTable" }
      Statistic: Sum
      Period: 300
      EvaluationPeriods: 12
      Threshold:
          Fn::GetOptionSetting:
            OptionName : SessionWriteCapacityUnitsAlarmThreshold
            DefaultValue: 240
      ComparisonOperator: GreaterThanThreshold
      AlarmActions:
```

```
        - Ref: SessionAlarmTopic
      InsufficientDataActions:
        - Ref: SessionAlarmTopic

  SessionReadCapacityUnitsLimit:
    Type: AWS::CloudWatch::Alarm
    Properties:
      AlarmDescription: { "Fn::Join" : ["", [{ "Ref" : "AWSEBEnvironmentName" }, " read
capacity limit on the session table." ]]}
      Namespace: "AWS/DynamoDB"
      MetricName: ConsumedReadCapacityUnits
      Dimensions:
        - Name: TableName
          Value: { "Ref" : "SessionTable" }
      Statistic: Sum
      Period: 300
      EvaluationPeriods: 12
      Threshold:
          Fn::GetOptionSetting:
            OptionName : SessionReadCapacityUnitsAlarmThreshold
            DefaultValue: 240
      ComparisonOperator: GreaterThanThreshold
      AlarmActions:
        - Ref: SessionAlarmTopic
      InsufficientDataActions:
        - Ref: SessionAlarmTopic

  SessionThrottledRequestsAlarm:
    Type: AWS::CloudWatch::Alarm
    Properties:
      AlarmDescription: { "Fn::Join" : ["", [{ "Ref" : "AWSEBEnvironmentName" }, ":
requests are being throttled." ]]}
      Namespace: AWS/DynamoDB
      MetricName: ThrottledRequests
      Dimensions:
        - Name: TableName
          Value: { "Ref" : "SessionTable" }
      Statistic: Sum
      Period: 300
      EvaluationPeriods: 1
      Threshold:
        Fn::GetOptionSetting:
          OptionName: SessionThrottledRequestsThreshold
          DefaultValue: 1
```

```
      ComparisonOperator: GreaterThanThreshold
      AlarmActions:
        - Ref: SessionAlarmTopic
      InsufficientDataActions:
        - Ref: SessionAlarmTopic

  SessionAlarmTopic:
    Type: AWS::SNS::Topic
    Properties:
      Subscription:
        - Endpoint:
            Fn::GetOptionSetting:
              OptionName: SessionAlarmEmail
              DefaultValue: "nobody@amazon.com"
          Protocol: email

files:
  "/var/app/sessiontable":
    mode: "000444"
    content: |
      `{"Ref" : "SessionTable"}`
      `{"Ref" : "AWS::Region"}`

  "/var/app/composer.json":
    mode: "000744"
    content:
      {
        "require": {
           "aws/aws-sdk-php": "*"
        }
      }

container_commands:
 "1-install-composer":
   command: "cd /var/app; curl -s http://getcomposer.org/installer | php"
 "2-install-dependencies":
   command: "cd /var/app; php composer.phar install"
 "3-cleanup-composer":
   command: "rm -Rf /var/app/composer.*"
```

在範本組態檔案中，我們首先建立 DynamoDB 資料表，並設定其主金鑰結構與容量單位，以配置足夠的資源來提供請求的傳輸量。接著，我們建立 `WriteCapacity` 和 `ReadCapacity`

的 CloudWatch 警示。我們會建立 SNS 主題，若超過警示閾值，此主題會傳送電子郵件至「nobody@amazon.com」。

建立並設定環境的 AWS 資源後，我們必須自訂 EC2 執行個體。我們使用 `files` 金鑰將 DynamoDB 資料表的詳細資訊傳送至環境中的 EC2 執行個體，並針對適用於 PHP 的 AWS 開發套件 2，於 `composer.json` 檔案新增一個 "require"。最後，我們執行容器命令來安裝 Composer 和所需依存項目，然後移除安裝程式。

`.ebextensions/options.config`

```
option_settings:
  "aws:elasticbeanstalk:customoption":
     SessionHashKeyName                      : username
     SessionHashKeyType                      : S
     SessionReadCapacityUnits                : 1
     SessionReadCapacityUnitsAlarmThreshold  : 240
     SessionWriteCapacityUnits               : 1
     SessionWriteCapacityUnitsAlarmThreshold : 240
     SessionThrottledRequestsThreshold       : 1
     SessionAlarmEmail                       : me@example.com
```

使用您希望接收警示通知的電子郵件，取代 SessionAlarmEmail 的值。`options.config` 檔案內含部分定義於 `dynamodb.config` 的變數的值。例如，`dynamodb.config` 內含下列行：

```
Subscription:
  - Endpoint:
      Fn::GetOptionSetting:
        OptionName: SessionAlarmEmail
        DefaultValue: "nobody@amazon.com"
```

這些行會指示 Elastic Beanstalk 自組態檔案 (在我們的範例應用程式為 `options.config`) 中的 SessionAlarmEmail 值，取得 Endpoint (端點) 屬性的值，該組態檔案中的 option_settings 區段帶有 aws:elasticbeanstalk:customoption 區段，其中包含要使用之實際值的名稱-值對。在上述範例中，這表示 SessionAlarmEmail 將指派 `nobody@amazon.com` 值。

如需本範例所使用的 CloudFormation 資源的詳細資訊，請參閱下列參考：

- AWS::DynamoDB::Table
- AWS::CloudWatch::Alarm
- AWS::SNS::Topic

使用 Elastic Beanstalk 已儲存組態

您可將環境的組態儲存為 Amazon Simple Storage Service (Amazon S3) 的物件，並於環境建立期間套用至其他環境，或套用至執行環境。「已儲存組態」為 YAML 格式的範本，定義了環境的平台版本、環境層、組態選項設定和標籤。

您可以在建立時將標籤套用至已儲存的組態，並編輯現有已儲存的組態的標籤。套用到已儲存的組態的標籤，與使用 `Tags:` 索引鍵的已儲存的組態中指定的標籤無關。後者會在您將已儲存的組態套用至環境時，套用到環境。如需詳細資訊，請參閱標記已儲存組態。

> **Note**
>
> 您可以使用多種方法建立和套用已儲存組態至 Elastic Beanstalk 環境。其中包括 Elastic Beanstalk 主控台、EB CLI，以及 AWS CLI。
> 有關建立和套用已儲存組態的替代方法，請參閱下列主題：
>
> - 於環境建立前設定組態選項
> - 於環境建立期間設定組態選項
> - 於環境建立後設定組態選項

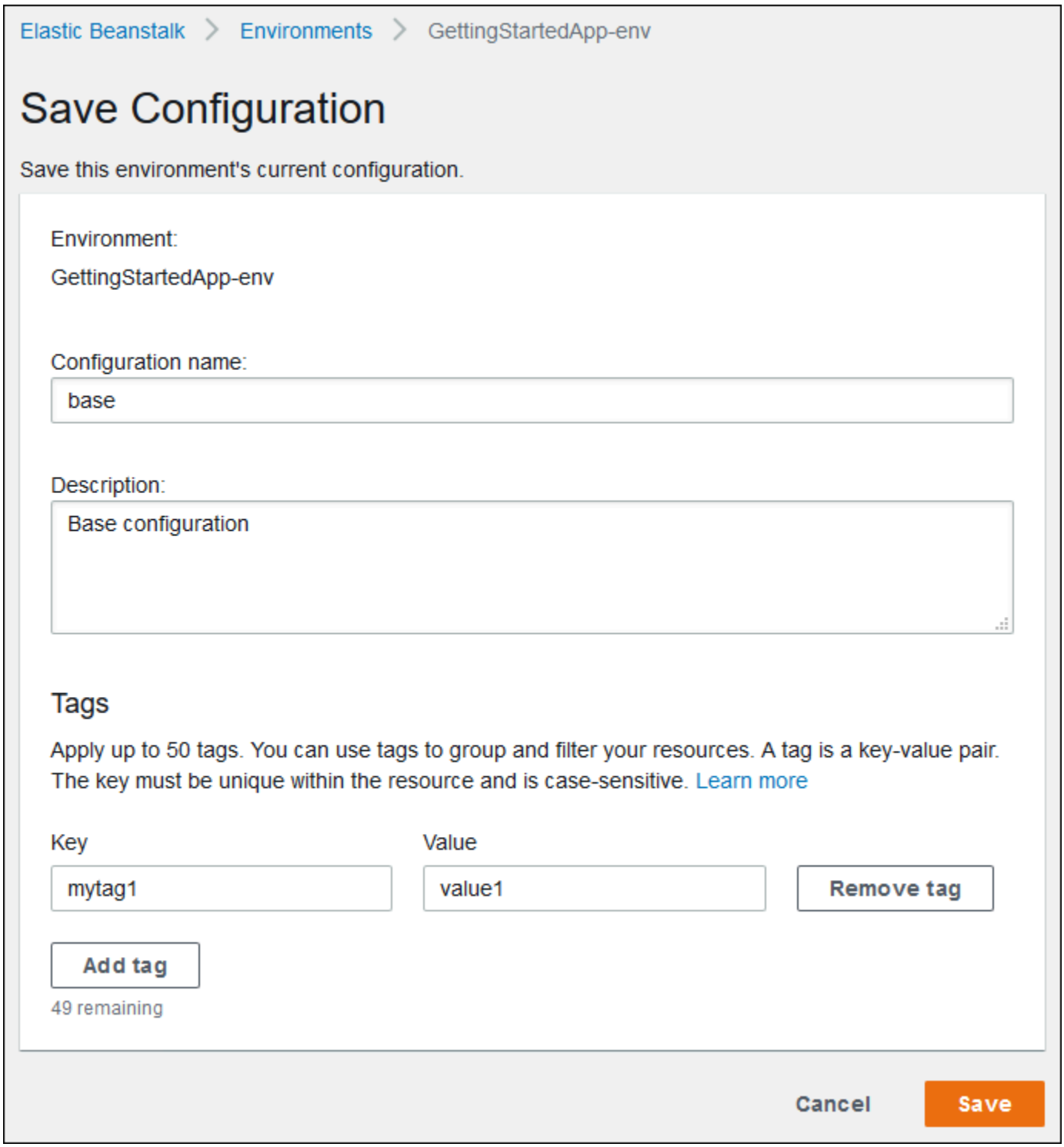
於 Elastic Beanstalk 管理主控台中，自您環境的目前狀態建立已儲存組態。

欲儲存環境的組態

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

   > **Note**
   >
   > 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 選擇 Actions (動作)，然後選擇 Save Configuration (儲存組態)。
4. 使用畫面顯示表單來命名儲存的組態。您可選擇性地提供簡短描述，並新增標籤索引鍵和值。
5. 選擇 Save (儲存)。



您透過主控台或其他運用 Elastic Beanstalk API 的用戶端套用至環境的任何設定，均會納入已儲存組態。接著，您可在日後將已儲存組態套用至您的環境以還原至先前狀態，或於環境建立期間將其套用至新環境。

您可透過 EB CLI the section called “eb config” 命令下載組態，如下列範例所示。*NAME* 是已儲存的組態名稱。

```
eb config get NAME
```

欲在環境建立期間套用已儲存組態 (Elastic Beanstalk 主控台)

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Applications (應用程式)，然後在清單上選取應用程式名稱。

> **Note**
>
> 如果您有許多應用程式，請使用搜尋列來篩選應用程式清單。

3. 在導覽窗格中，找到應用程式名稱並選擇 Saved configurations (已儲存的配置)。
4. 選取您要套用的已儲存組態，然後選擇 Launch environment (啟動環境)。
5. 繼續透過精靈來建立您的環境。

已儲存組態不包含您應用程式原始碼的組態檔案所套用的設定。若組態檔案和已儲存組態均套用相同設定，會優先採用已儲存組態中的設定。同樣的，Elastic Beanstalk 主控台中指定的選項會覆寫已儲存組態中的選項。如需更多詳細資訊，請參閱 優先順序。

已儲存組態會存放在 Elastic Beanstalk S3 儲存貯體內以您的應用程式命名之資料夾。例如，us-west-2 區域中對於 123456789012 帳戶一個名為 `my-app` 應用程式的組態位於 `s3://elasticbeanstalk-us-west-2-123456789012/resources/templates/my-app/`。

於文字編輯器開啟已儲存組態，以檢視其內容。下列範例組態呈現 Elastic Beanstalk 管理主控台啟動之 Web 伺服器環境的組態。

```
EnvironmentConfigurationMetadata:
  Description: Saved configuration from a multicontainer Docker environment created
 with the Elastic Beanstalk Management Console
  DateCreated: '1520633151000'
  DateModified: '1520633151000'
Platform:
  PlatformArn: arn:aws:elasticbeanstalk:us-east-2::platform/Java 8 running on 64bit
 Amazon Linux/2.5.0
OptionSettings:
  aws:elasticbeanstalk:command:
```

```
    BatchSize: '30'
    BatchSizeType: Percentage
  aws:elasticbeanstalk:sns:topics:
    Notification Endpoint: me@example.com
  aws:elb:policies:
    ConnectionDrainingEnabled: true
    ConnectionDrainingTimeout: '20'
  aws:elb:loadbalancer:
    CrossZone: true
  aws:elasticbeanstalk:environment:
    ServiceRole: aws-elasticbeanstalk-service-role
  aws:elasticbeanstalk:application:
    Application Healthcheck URL: /
  aws:elasticbeanstalk:healthreporting:system:
    SystemType: enhanced
  aws:autoscaling:launchconfiguration:
    IamInstanceProfile: aws-elasticbeanstalk-ec2-role
    InstanceType: t2.micro
    EC2KeyName: workstation-uswest2
  aws:autoscaling:updatepolicy:rollingupdate:
    RollingUpdateType: Health
    RollingUpdateEnabled: true
EnvironmentTier:
  Type: Standard
  Name: WebServer
AWSConfigurationTemplateVersion: 1.1.0.0
Tags:
  Cost Center: WebApp Dev
```

您可以修改已儲存組態的內容，並將其儲存於 Amazon S3 內的相同位置。任何已儲存組態只要格式正確且存放於適當位置，均可透過 Elastic Beanstalk 管理主控台套用至環境。

支援下列按鍵。

- AWSConfigurationTemplateVersion (必要) – 組態範本版本 (1.1.0.0)。

```
AWSConfigurationTemplateVersion: 1.1.0.0
```

- 平台 – 環境平台版本的 Amazon Resource Name (ARN)。您可依照 ARN 或解決方案堆疊名稱指定平台。

```
Platform:
```

```
  PlatformArn: arn:aws:elasticbeanstalk:us-east-2::platform/Java 8 running on 64bit
Amazon Linux/2.5.0
```

- SolutionStack – 用於建立環境的解決方案堆疊全名。

```
SolutionStack: 64bit Amazon Linux 2017.03 v2.5.0 running Java 8
```

- OptionSettings – 欲套用至環境的組態選項設定。例如，下列項目將執行個體類型設定為 t2.micro。

```
OptionSettings:
  aws:autoscaling:launchconfiguration:
    InstanceType: t2.micro
```

- 標籤 – 最多 47 個標籤可套用至環境內建立的資源。

```
Tags:
  Cost Center: WebApp Dev
```

- EnvironmentTier – 欲建立的環境類型。以 Web 伺服器環境而言，此區段可以排除 (Web 伺服器為預設值)。以工作者環境而言，請使用下列。

```
EnvironmentTier:
  Name: Worker
  Type: SQS/HTTP
```

> **Note**
>
> 您可以使用多種方法建立和套用已儲存組態至 Elastic Beanstalk 環境。其中包括 Elastic Beanstalk 主控台、EB CLI，以及 AWS CLI。
> 有關建立和套用已儲存組態的替代方法，請參閱下列主題：
>
> - 於環境建立前設定組態選項
> - 於環境建立期間設定組態選項
> - 於環境建立後設定組態選項

標記已儲存組態

您可以將標籤套用到 AWS Elastic Beanstalk 已儲存組態。標籤是與 AWS 資源關聯的金鑰值對。如需 Elastic Beanstalk 資源標記、使用案例、標籤索引鍵和值限制條件的相關資訊，以及支援的資源類型，請參閱標記 Elastic Beanstalk 應用程式資源。

您可以在建立已儲存組態時指定標籤。您可以在現有的已儲存組態中新增或移除標籤，以及更新現有標籤的值。您最多可以為每個已儲存組態新增 50 個標籤。

在建立已儲存組態時新增標籤

當您使用 Elastic Beanstalk 主控台儲存組態時，您可以在 Save Configuration (儲存組態) 頁面上指定標籤索引鍵和值。

如果您使用 EB CLI 來儲存組態，請使用 eb config 的 `--tags` 選項來新增標籤。

```
~/workspace/my-app$ eb config --tags mytag1=value1,mytag2=value2
```

若為 AWS CLI 或其他 API 型用戶端，請在 create-configuration-template 命令中使用 `--tags` 參數來新增標籤。

```
$ aws elasticbeanstalk create-configuration-template \
      --tags Key=mytag1,Value=value1 Key=mytag2,Value=value2 \
      --application-name my-app --template-name my-template --solution-stack-
name solution-stack
```

管理現有已儲存組態的標籤

您可以新增、更新和刪除現有 Elastic Beanstalk 已儲存組態中的標籤。

使用 Elastic Beanstalk 主控台管理已儲存組態的標籤

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Applications (應用程式)，然後在清單上選取應用程式名稱。

> **Note**
>
> 如果您有許多應用程式，請使用搜尋列來篩選應用程式清單。

3. 在導覽窗格中，找到應用程式名稱並選擇 Saved configurations (已儲存的配置)。
4. 選取您要管理的已儲存組態。
5. 選擇 Actions (動作)，然後選擇 Manage tags (管理標籤)。
6. 使用畫面顯示表單來新增、更新或刪除標籤。
7. 若要儲存變更，請選擇頁面底部的儲存變更。

如果您使用 EB CLI 更新已儲存組態，請使用 eb tags 新增、更新、刪除或列出標籤。

例如，以下命令會列出已儲存組態中的標籤。

```
~/workspace/my-app$ eb tags --list --resource "arn:aws:elasticbeanstalk:us-east-2:my-account-id:configurationtemplate/my-app/my-template"
```

下列命令會更新標籤 mytag1 並刪除標籤 mytag2。

```
~/workspace/my-app$ eb tags --update mytag1=newvalue --delete mytag2 \
      --resource "arn:aws:elasticbeanstalk:us-east-2:my-account-id:configurationtemplate/my-app/my-template"
```

如需完整選項清單和更多範例，請參閱 eb tags。

若為 AWS CLI 或其他 API 型用戶端，請使用 list-tags-for-resource 命令來列出已儲存組態的標籤。

```
$ aws elasticbeanstalk list-tags-for-resource --resource-arn
 "arn:aws:elasticbeanstalk:us-east-2:my-account-id:configurationtemplate/my-app/my-template"
```

使用 update-tags-for-resource 命令新增、更新或刪除已儲存組態中的標籤。

```
$ aws elasticbeanstalk update-tags-for-resource \
      --tags-to-add Key=mytag1,Value=newvalue --tags-to-remove mytag2 \
      --resource-arn "arn:aws:elasticbeanstalk:us-east-2:my-account-id:configurationtemplate/my-app/my-template"
```

在 --tags-to-add 的 update-tags-for-resource 參數中，同時指定欲新增和欲更新的標籤。如此將新增不存在的標籤，並更新現有標籤的值。

> **Note**
>
> 若要使用 Elastic Beanstalk 已儲存組態的部分 EB CLI 和 AWS CLI 命令，您需要已儲存組態的 ARN。若要建構 ARN，請先使用下列命令擷取已儲存組態的名稱。
>
> ```
> $ aws elasticbeanstalk describe-applications --application-names my-app
> ```
>
> 在命令輸入中尋找 `ConfigurationTemplates` 索引鍵。這個元素會顯示已儲存組態的名稱。如果此頁面提及的命令已指定 *`my-template`*，請使用此名稱。

環境資訊清單 (`env.yaml`)

您可於應用程式原始碼套件的根目錄內，納入 YAML 格式的環境資訊清單，藉此設定建立環境時可用的環境名稱、解決方案堆疊和環境連結。

此檔案格式可支援環境群組。欲使用群組，請於資訊清單內，將環境名稱的尾端指定為 + 符號。建立或更新環境時，請透過 `--group-name` (AWS CLI) 或 `--env-group-suffix` (EB CLI) 來指定群組名稱。如需群組的詳細資訊，請參閱建立和更新 Elastic Beanstalk 環境的群組。

下列範例資訊清單定義 Web 伺服器環境，其中包含其相依的工作者環境元件的連結。資訊清單會使用群組，以允許使用相同原始碼套件來建立多個環境：

`~/myapp/frontend/env.yaml`

```
AWSConfigurationTemplateVersion: 1.1.0.0
SolutionStack: 64bit Amazon Linux 2015.09 v2.0.6 running Multi-container Docker 1.7.1
 (Generic)
OptionSettings:
  aws:elasticbeanstalk:command:
    BatchSize: '30'
    BatchSizeType: Percentage
  aws:elasticbeanstalk:sns:topics:
    Notification Endpoint: me@example.com
  aws:elb:policies:
    ConnectionDrainingEnabled: true
    ConnectionDrainingTimeout: '20'
  aws:elb:loadbalancer:
    CrossZone: true
  aws:elasticbeanstalk:environment:
    ServiceRole: aws-elasticbeanstalk-service-role
```

```
  aws:elasticbeanstalk:application:
    Application Healthcheck URL: /
  aws:elasticbeanstalk:healthreporting:system:
    SystemType: enhanced
  aws:autoscaling:launchconfiguration:
    IamInstanceProfile: aws-elasticbeanstalk-ec2-role
    InstanceType: t2.micro
    EC2KeyName: workstation-uswest2
  aws:autoscaling:updatepolicy:rollingupdate:
    RollingUpdateType: Health
    RollingUpdateEnabled: true
Tags:
  Cost Center: WebApp Dev
CName: front-A08G28LG+
EnvironmentName: front+
EnvironmentLinks:
  "WORKERQUEUE" : "worker+"
```

支援下列按鍵。

- AWSConfigurationTemplateVersion (必要) – 組態範本版本 (1.1.0.0)。

```
AWSConfigurationTemplateVersion: 1.1.0.0
```

- 平台 – 環境平台版本的 Amazon Resource Name (ARN)。您可依照 ARN 或解決方案堆疊名稱指定平台。

```
Platform:
  PlatformArn: arn:aws:elasticbeanstalk:us-east-2::platform/Java 8 running on 64bit
 Amazon Linux/2.5.0
```

- SolutionStack – 用於建立環境的解決方案堆疊全名。

```
SolutionStack: 64bit Amazon Linux 2017.03 v2.5.0 running Java 8
```

- OptionSettings – 欲套用至環境的組態選項設定。例如，下列項目將執行個體類型設定為 t2.micro。

```
OptionSettings:
  aws:autoscaling:launchconfiguration:
    InstanceType: t2.micro
```

- 標籤 – 最多 47 個標籤可套用至環境內建立的資源。

```
Tags:
  Cost Center: WebApp Dev
```

- EnvironmentTier – 欲建立的環境類型。以 Web 伺服器環境而言，此區段可以排除 (Web 伺服器為預設值)。以工作者環境而言，請使用下列。

```
EnvironmentTier:
  Name: Worker
  Type: SQS/HTTP
```

- CName – 環境的 CNAME，名稱結尾包含 + 字元以啟用群組。

```
CName: front-A08G28LG+
```

- EnvironmentName – 欲建立的環境名稱。名稱結尾包含 + 字元以啟用群組。

```
EnvironmentName: front+
```

  如需啟用群組，務必於建立環境時指定群組名稱。Elastic Beanstalk 會將群組名稱以連字號附加於環境名稱。以環境名稱 `front+` 和群組名稱 `dev` 為例，Elastic Beanstalk 將建立的環境名稱為 `front-dev`。

- EnvironmentLinks – 相依的變數名稱和環境名稱的對應。下列範例使得 `worker+` 環境成為依存項目，並指示 Elastic Beanstalk 將連結資訊儲存至名為 `WORKERQUEUE` 的變數。

```
EnvironmentLinks:
  "WORKERQUEUE" : "worker+"
```

  連結變數的值取決於連結環境的類型。以 Web 伺服器環境而言，該連結是環境的 CNAME。以工作者環境而言，該連結是環境 Amazon Simple Queue Service (Amazon SQS) 佇列的名稱。

CName、EnvironmentName 和 EnvironmentLinks 鍵都可用於建立環境群組及其他環境的連結。EB CLI、AWS CLI 或軟體開發套件現均支援這些功能。

使用自訂的 Amazon Machine Image (AMI)

建立 AWS Elastic Beanstalk 環境時，您可以指定要使用的 Amazon 機器映像 (AMI)，而不是平台版本中包含的標準 Elastic Beanstalk AMI。如果您需要安裝標準 AMI 中未包含的許多軟體，則自訂 AMI 可在您環境中啟動執行個體時，縮短佈建的時間。

使用組態檔案最能快速一致地設定和自訂您的環境。不過，在建立和更新環境時，套用組態可能會需要很長的時間。如果您在組態檔中執行許多伺服器組態，可以透過自訂 AMI (其中包含您所需的軟體和設定)，來縮短執行作業的時間。

自訂 AMI 也可讓您變更低層級元件 (例如，Linux 核心)，這類元件難以實作或需要花費很長的時間來套用至組態檔案。若要建立自訂的 AMI，請在 Amazon EC2 中啟動 Elastic Beanstalk 平台的 AMI、根據您的需要來自訂軟體和設定，然後停止執行個體和儲存產生的 AMI。

建立自訂 AMI

識別基礎 Elastic Beanstalk AMI

1. 在命令視窗中，執行如下所示的命令。若要取得更多資訊，請參閱《指AWS CLI 令參考》describe-platform-version中的。

   指定您要使用自定義 AMI 的 AWS 區域，並將平台 ARN 和版本號替換為您的應用程序所依據的 Elastic Beanstalk 平台。

   Example – Mac 作業系統/Linux 作業系統

```
$ aws elasticbeanstalk describe-platform-version --region us-east-2 \
      --platform-arn "arn:aws:elasticbeanstalk:us-east-2::platform/Tomcat 8.5 with
 Java 8 running on 64bit Amazon Linux/3.1.6" \
      --query PlatformDescription.CustomAmiList
[
    {
        "VirtualizationType": "pv",
        "ImageId": ""
    },
    {
        "VirtualizationType": "hvm",
        "ImageId": "ami-020ae06fdda6a0f66"
    }
]
```

Example – Windows 作業系統

```
C:\> aws elasticbeanstalk describe-platform-version --region us-east-2 --platform-
arn"arn:aws:elasticbeanstalk:us-east-2::platform/IIS 10.0 running on 64bit Windows
 Server 2019/2.6.4" --query PlatformDescription.CustomAmiList
[
    {
        "VirtualizationType": "pv",
        "ImageId": ""
    },
    {
        "VirtualizationType": "hvm",
        "ImageId": "ami-020ae06fdda6a0f66"
    }
]
```

2. 請記錄結果中看起來像是 `ami-020ae06fdda6a0f66` 的 `ImageId` 值。

值為與您的應用程式相關的平台版本、EC2 執行個體架構和 AWS 區域的存貨 Elastic Beanstalk AMI。如果您需要為多個平台、架構或 AWS 區域建立 AMI，請重複此程序以識別每個組合的正確基礎 AMI。

> **備註**
>
> - 請勿利用 Elastic Beanstalk 環境中已啟動的執行個體，來建立 AMI。Elastic Beanstalk 會在佈建時對執行個體進行變更，而這可能會在儲存的 AMI 中造成問題。如果從 Elastic Beanstalk 環境中的執行個體儲存映像，也會製作已經部署到固定映像部分之執行個體的應用程式版本。
> - 我們建議您一律使用最新的平台版本。當您更新至新平台版本時，我們也建議您將自訂 AMI 重設為新平台版本的 AMI。如此可大幅減少因不相容的套件或程式庫版本所導致的部署失敗。

對於 Linux，您也可以利用 Elastic Beanstalk 未發佈的社群 AMI，來建立自訂的 AMI。您可以使用最新的 Amazon Linux AMI 做為起點。當您以非 Elastic Beanstalk 管理之 Linux AMI 啟動環境時，Elastic Beanstalk 會嘗試安裝平台軟體 (語言、框架、代理伺服器等) 和其他元件以支援功能，例如增強型運作狀態報告。

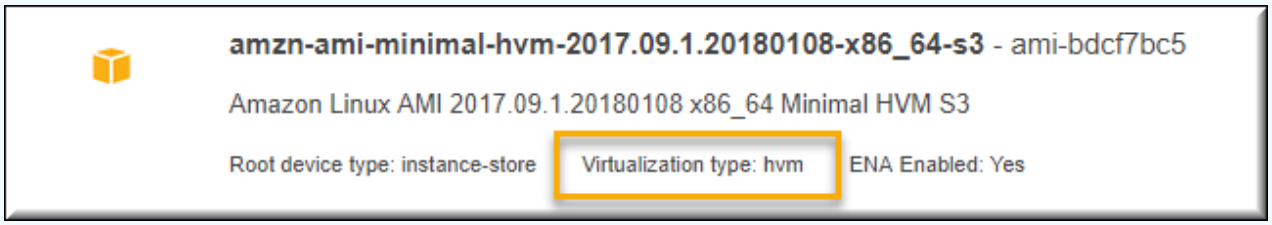
> **Note**
>
> 以 Windows 伺服器為基礎的自訂 AMI 需要從 `describe-platform-version` 傳回的庫存 Elastic Beanstalk AMI，如先前步驟 1 所示。

雖然 Elastic Beanstalk 可使用非 Elastic Beanstalk 管理的 AMI，但因為 Elastic Beanstalk 安裝缺少元件所造成的佈建時間增加，會降低或消除一開始就建立自訂 AMI 時所預期帶來的效益。其他的 Linux 版本也許可以在進行一些故障排除後使用，但並未正式支援。如果您的應用程式需要特定的 Linux 發行版本，一個替代方法是建立 Docker 影像，然後在 Elastic Beanstalk 上的 Docker 平台或多容器 Docker 平台上執行此影像。

若要建立自訂的 AMI

1. 在 https://console.aws.amazon.com/ec2/ 開啟 Amazon EC2 主控台。
2. 選擇 Launch Instance (啟動執行個體)。
3. 選擇 Community AMI (社群 AMI)。
4. 如果您識別了Elastic Beanstalk AMI (使用 `describe-platform-version`) 或 Amazon Linux AMI，請搜尋方塊中輸入其 AMI ID。然後按 Enter。

   您也可以搜尋清單，來尋找符合您需求的其他社群 AMI。

   > **Note**
   >
   > 我們建議您選擇使用 HVM 虛擬化的 AMI。這些 AMI 會在描述中顯示 Virtualization type: hvm (虛擬化類型：hvm)。
   >
   > 
   >
   >
   > 如需執行個體虛擬化類型的詳細資訊，請參閱 Amazon EC2 使用者指南中的 Linux AMI 虛擬化類型或 Amazon EC2 使用者指南中的 Windows AMI 虛擬化類型。

5. 選擇 Select (選取) 來選擇 AMI。
6. 選取執行個體類型，然後選擇 Next: Configure Instance Details (下一步：設定執行個體的詳細資訊)。

7. (若為 Linux 平台) 展開 Advanced Details (進階詳細資訊) 的區段，並將下列的文字內容貼入 User Data (使用者資料)欄位中。

```
#cloud-config
  repo_releasever: repository version number
  repo_upgrade: none
```

儲存庫版本編號代表 AMI 名稱中的版本年份和月份。例如，採用 Amazon Linux 2015 年 3 月版的 AMI，會具有 2015.03 的儲存庫版本編號。對於 Elastic Beanstalk 映像，這符合您平台版本解決方案堆疊名稱中所顯示的日期 (根據 Amazon Linux AMI，先前為 Amazon Linux 2)。

> **Note**
>
> 此repo_releasever設定會設定 Amazon Linux lock-on-launch AMI 的功能。這會導致 AMI 在啟動時使用固定的特定儲存庫版本。Amazon Linux 2 不支援此功能 — 如果您的環境使用目前的 Amazon Linux 2 平台分支，則請勿指定。如果您只在 Amazon Linux AMI 平台分支上將自訂 AMI 搭配 Elastic Beanstalk 使用 (先前為 Amazon Linux 2)，則需要此設定。
>
> 此 repo_upgrade 設定會停用自動安裝安全性更新的功能。如果要將自訂 AMI 搭配 Elastic Beanstalk 使用，需執行這兩項動作。

8. 繼續透過精靈來啟動 EC2 執行個體。出現提示時，請選擇您有權存取的金鑰對，以連接到執行個體來進行後續步驟。
9. 使用 SSH 或 RDP 來連接到執行個體。
10. 進行任何所需的自訂。
11. (Windows 平台) 執行 EC2Config 服務 Sysprep。有關 EC2Config 的詳細資訊，請參閱使用 EC2Config 服務來設定 Windows 執行個體。請確定將 Sysprep 設定為產生隨機密碼，此密碼可以從 AWS Management Console擷取。
12. 在 Amazon EC2 主控台，停止 EC2 執行個體。然後在 Instance Actions (執行個體動作) 功能表中，選擇 Create Image (EBS AMI) (建立映像 (EBS AMI))。
13. 為避免產生額外 AWS 費用，請終止 EC2 執行個體。

若要在 Elastic Beanstalk 環境使用您的自訂 AMI

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇 Configuration (組態)。
4. 在 Capacity (容量) 組態類別中，選擇 Edit (編輯)。
5. 如需 AMI ID，請輸入您的自訂 AMI ID。
6. 若要儲存變更，請選擇頁面底部的儲存變更。

當您使用自訂的 AMI 來建立新的環境時，您應該使用做為基礎來建立 AMI 時所使用的相同平台版本。如果您稍後將平台更新套用到使用自訂 AMI 的環境，Elastic Beanstalk 會嘗試在開機載入作業期間套用程式庫與組態更新。

清除自訂 AMI

當您完成自訂 AMI 且不需要再啟動 Elastic Beanstalk 環境，請考慮將其清除，以將儲存成本降到最低。清除自訂 AMI 涉及從 Amazon EC2 取消註冊，並刪除其他關聯的資源。如需詳細資訊，請參閱取消註冊您的 Linux AMI 或取消註冊您的 Windows AMI。

保留已淘汰平台的 Amazon Machine Image (AMI) 存取權

當分支使用的作業系統或主要元件達到生命週期結束時，Elastic Beanstalk 會將平台分支狀態設定為已淘汰。平台分支的基礎 Elastic Beanstalk AMI 可能也會變成私有，以防止使用此過時的 AMI。使用已變成私有的 AMI 的環境將無法再啟動執行個體。

如果您無法在其淘汰前將應用程式遷移到支援的環境，則您的環境可能會遇到這種情況。可能會需要更新基礎 Elastic Beanstalk AMI 已被設為私有的 Beanstalk 平台分支的環境。還有另外一種方式。您可以根據您的環境所使用的基礎 Elastic Beanstalk AMI 的副本來更新現有環境。

這個主題提供一些步驟和一個獨立指令碼，以便根據您的環境所使用的基礎 Elastic Beanstalk AMI 的副本來更新現有環境。一旦您能夠將應用程式遷移到支援的平台，就可以繼續使用標準程序來維護您的應用程式和支援的環境。

手動步驟

根據基礎 Elastic Beanstalk AMI 的 AMI 副本來更新環境

1. 判斷您的環境正在使用的 AMI。此命令會傳回您在參數中提供的 Elastic Beanstalk 環境所使用的 AMI。在下一個步驟中，會將傳回的值作為 source-ami-id。

   在命令視窗中，執行如下所示的命令。如需詳細資訊，請參閱《AWS CLI 命令參考》中的 describe-configuration-settings。

   指定儲存您要複製的來源 AMI 的 AWS 區域。將應用程式名稱和環境名稱換成基於來源 AMI 的應用程式名稱和環境名稱。輸入如所示之查詢參數的文字。

   Example

   ```
   >aws elasticbeanstalk describe-configuration-settings \
     --application-name my-application \
     --environment-name my-environment \
     --region us-east-2 \
     --query "ConfigurationSettings[0].OptionSettings[?OptionName=='ImageId'] |
    [0].Value"
   ```

2. 將 AMI 複製到您的帳戶。這個命令會傳回因複製先前步驟中傳回的 source-ami-id 而得到的新 AMI。

   > **Note**
   >
   > 務必記下此命令輸出的新 AMI ID。您需要在下一步中輸入它，用來替換範例命令中的 copied-ami-id 。

   在命令視窗中，執行如下所示的命令。如需詳細資訊，請參閱《AWS CLI 命令參考》中的 copy-image。

   指定您要複製的來源 AMI 的 AWS 區域 (--source-region) 以及您要使用新的自訂 AMI 的區域 (--region)。將 source-ami-id 換成複製的映像的 AMI。先前步驟中的命令傳回 source-ami-id。將 new-ami-name 換成描述目的地區域中新 AMI 的名稱。依循此程序的指令碼會將字串 "Copy of" 附加到 source-ami-id 名稱的開頭，產生新的 AMI 名稱。

```
>aws ec2 copy-image \
    --region us-east-2 \
    --source-image-id source-ami-id \
    --source-region us-east-2 \
    --name new-ami-name
```

3. 更新環境以使用複製的 AMI。命令執行後，會回傳環境的狀態。

   在命令視窗中，執行如下所示的命令。如需詳細資訊，請參閱《AWS CLI 命令參考》中的 update-environment 一節。

   指定您需要更新的環境和應用程式的 AWS 區域。將應用程式名稱和環境名稱換成您需要與先前步驟中 copied-ami-id 相關聯的應用程式名稱和環境名稱。對於 --option-setttings 參數，將*copied-ami-id* 換成您從先前命令的輸出中記錄下來的 AMI ID。

```
>aws elasticbeanstalk update-environment \
  --application-name my-application \
  --environment-name my-environment \
  --region us-east-2 \
  --option-settings
 "Namespace=aws:autoscaling:launchconfiguration,OptionName=ImageId,Value=copied-ami-id"
```

> **Note**
>
> 為將儲存成本降至最低，當不再需要使用自訂 AMI 來啟動 Elastic Beanstalk 環境時，請考慮將其清除。如需更多詳細資訊，請參閱 清除自訂 AMI。

獨立指令碼

以下指令碼提供與先前手動步驟相同的結果。選取此連結以下載指令碼：copy_ami_and_update_env.zip。

指令碼來源：copy_ami_and_update_env.sh

```
#!/bin/bash

set -ue
```

```
USAGE="This script is used to copy an AMI used by your Elastic Beanstalk environment
 into your account to use in your environment.\n\n"
USAGE+="Usage:\n\n"
USAGE+="./$(basename $0) [OPTIONS]\n"
USAGE+="OPTIONS:\n"
USAGE+="\t--application-name <application-name>\tThe name of your Elastic Beanstalk
 application.\n"
USAGE+="\t--environment-name <environment-name>\tThe name of your Elastic Beanstalk
 environment.\n"
USAGE+="\t--region <region> \t\t\tThe AWS region your Elastic Beanstalk environment is
 deployed to.\n"
USAGE+="\n\n"
USAGE+="Script Usage Example(s):\n"
USAGE+="./$(basename $0) --application-name my-application --environment-name my-
environment --region us-east-1\n"

if [ $# -eq 0 ]; then
  echo -e $USAGE
  exit
fi

while [[ $# -gt 0 ]]; do
  case $1 in
    --application-name)       APPLICATION_NAME="$2"; shift ;;
    --environment-name)       ENVIRONMENT_NAME="$2"; shift ;;
    --region)                 REGION="$2"; shift ;;
    *)                        echo "Unknown option $1" ; echo -e $USAGE ; exit ;;
  esac
  shift
done

aws_cli_version="$(aws --version)"
if [ $? -ne 0 ]; then
  echo "aws CLI not found. Please install it: https://docs.aws.amazon.com/cli/latest/
userguide/getting-started-install.html. Exiting."
  exit 1
fi
echo "Using aws CLI version: ${aws_cli_version}"

account=$(aws sts get-caller-identity --query "Account" --output text)
echo "Using account ${account}"

environment_ami_id=$(aws elasticbeanstalk describe-configuration-settings \
```

```
  --application-name "$APPLICATION_NAME" \
  --environment-name "$ENVIRONMENT_NAME" \
  --region "$REGION" \
  --query "ConfigurationSettings[0].OptionSettings[?OptionName=='ImageId'] | [0].Value"
 \
  --output text)
echo "Image associated with environment ${ENVIRONMENT_NAME} is ${environment_ami_id}"

owned_image=$(aws ec2 describe-images \
  --owners self \
  --image-ids "$environment_ami_id" \
  --region "$REGION" \
  --query "Images[0]" \
  --output text)
if [ "$owned_image" != "None" ]; then
  echo "${environment_ami_id} is already owned by account ${account}. Exiting."
  exit
fi

source_image_name=$(aws ec2 describe-images \
  --image-ids "$environment_ami_id" \
  --region "$REGION" \
  --query "Images[0].Name" \
  --output text)
if [ "$source_image_name" = "None" ]; then
  echo "Cannot find ${environment_ami_id}. Please contact AWS support if you need
 additional help: https://aws.amazon.com/support."
  exit 1
fi

copied_image_name="Copy of ${source_image_name}"
copied_ami_id=$(aws ec2 describe-images \
  --owners self \
  --filters Name=name,Values="${copied_image_name}" \
  --region "$REGION" \
  --query "Images[0].ImageId" \
  --output text)
if [ "$copied_ami_id" != "None" ]; then
  echo "Detected that ${environment_ami_id} has already been copied by account
 ${account}. Skipping image copy."
else
  echo "Copying ${environment_ami_id} to account ${account} with name
 ${copied_image_name}"
  copied_ami_id=$(aws ec2 copy-image \
```

```
    --source-image-id "$environment_ami_id" \
    --source-region "$REGION" \
    --name "$copied_image_name" \
    --region "$REGION" \
    --query "ImageId" \
    --output text)
  echo "New AMI ID is ${copied_ami_id}"

  echo "Waiting for ${copied_ami_id} to become available"
  aws ec2 wait image-available \
    --image-ids "$copied_ami_id" \
    --region "$REGION"
  echo "${copied_ami_id} is now available"
fi

echo "Updating environment ${ENVIRONMENT_NAME} to use ${copied_ami_id}"
environment_status=$(aws elasticbeanstalk update-environment \
  --application-name "$APPLICATION_NAME" \
  --environment-name "$ENVIRONMENT_NAME" \
  --option-settings
 "Namespace=aws:autoscaling:launchconfiguration,OptionName=ImageId,Value=
${copied_ami_id}" \
  --region "$REGION" \
  --query "Status" \
  --output text)
echo "Environment ${ENVIRONMENT_NAME} is now ${environment_status}"

echo "Waiting for environment ${ENVIRONMENT_NAME} update to complete"
aws elasticbeanstalk wait environment-updated \
  --application-name "$APPLICATION_NAME" \
  --environment-names "$ENVIRONMENT_NAME" \
  --region "$REGION"
echo "Environment ${ENVIRONMENT_NAME} update complete"
```

Note

您必須安裝有 AWS CLI 才能執行指令碼。如需安裝說明，請參閱《AWS Command Line Interface 使用者指南》中的安裝或更新 AWS CLI 的最新版本。

安裝 AWS CLI 之後，您還必須進行設定以使用擁有該環境的 AWS 帳戶。如需詳細資訊，請參閱《AWS Command Line Interface 使用者指南》中的設定 AWS CLI。該帳戶還必須具有建立 AMI 和更新 Elastic Beanstalk 環境的許可。

這些步驟說明指令碼依循的程序。

1. 列印使用中的帳戶。
2. 判斷環境所使用的 AMI (來源 AMI)。
3. 檢查帳戶是否已擁有來源 AMI。如果是，請退出。
4. 判斷來源 AMI 的名稱，以便能在新的 AMI 名稱中使用。這也用於確認對來源 AMI 的存取。
5. 檢查來源 AMI 是否已複製到帳戶。使用該帳戶擁有的已複製 AMI 的名稱來搜尋 AMI，完成上述操作。如果在指令碼執行之間 AMI 名稱發生了變化，則會再次複製映像。
6. 如果尚未複製來源 AMI，請將來源 AMI 複製到帳戶，然後等待新 AMI 可用。
7. 更新環境組態以使用新 AMI。
8. 等待環境更新完成。

將 copy_ami_and_update_env.zip 檔案解壓縮取得指令碼後，執行以下範例來執行指令碼。將範例中的應用程式名稱和環境名稱換成您自己的值。

```
>sh copy_ami_and_update_env.sh \
  --application-name my-application \
  --environment-name my-environment \
  --region us-east-1
```

> **Note**
>
> 為將儲存成本降至最低，當不再需要使用自訂 AMI 來啟動 Elastic Beanstalk 環境時，請考慮將其清除。如需更多詳細資訊，請參閱 清除自訂 AMI。

提供靜態檔案

為了改善效能，您可以設定 Proxy 伺服器以提供靜態檔案 (例如 HTML 或影像)，這些靜態檔案都來自 Web 應用程式中的一組目錄。代理伺服器收到位於指定路徑下的檔案請求時，會直接提供檔案而非將請求路由至您的應用程式。

Elastic Beanstalk 支援設定代理以根據 Amazon Linux 2 在大多數平台分支上提供靜態檔案。唯一的例外是 Docker。

> **Note**
>
> 在 Python 和 Ruby 平台上，預設情況下 Elastic Beanstalk 設定一些靜態資料夾。如需詳細資訊，請參閱 Python 和 Ruby 靜態檔案的組態部分。您可以按照本頁的說明來設定其他資料夾。

使用主控台設定靜態檔案

若要設定代理伺服器來提供靜態檔案

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

   > **Note**
   >
   > 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇 Configuration (組態)。
4. 在更新、監控和日誌記錄組態類別中，選擇編輯。
5. 捲動至平台軟體區段，找到靜態檔案群組。

   a. 若要新增靜態檔案對應，請選取新增靜態檔案。在出現的額外列中，您將輸入提供靜態檔案的路徑以及包含所提供靜態檔案的目錄。

   - 在路徑欄位中，以斜線 (/) 作為路徑名稱的開頭 (例如，「/images」)。
   - 在目錄欄位中，指定位於應用程式原始程式碼根目錄的目錄名稱。請勿以斜線作為目錄的開頭 (例如，「static/image-files」)。

   > **Note**
   >
   > 如果沒有看見 Static files (靜態檔案) 區段，您必須使用組態檔案。如需詳細資訊，請參閱本頁上的the section called “使用組態選項設定靜態檔案”。

   b. 若要移除應對，請按一下移除。
6. 若要儲存變更，請選擇頁面底部的儲存變更。

使用組態選項設定靜態檔案

您可以使用組態檔案，使用組態選項來設定靜態檔案路徑和目錄位置。您可以將組態檔案新增至應用程式的來源套件，並在建立環境或稍後部署期間部署。

如果您的環境使用以 Amazon Linux 2 為基礎的平台分支，請使用 `aws:elasticbeanstalk:environment:proxy:staticfiles` 命名空間。

下列範例組態檔案會通知代理伺服器在路徑上提供 `statichtml` 資料夾中的檔案，以及在路徑 `/html` 上提供路徑 `/images` 的 `staticimages` 資料夾中的檔案。

Example .ebextensions/static-files.config

```
option_settings:
  aws:elasticbeanstalk:environment:proxy:staticfiles:
    /html: statichtml
    /images: staticimages
```

如果您的 Elastic Beanstalk 環境使用 Amazon Linux AMI 平台版本 (Amazon Linux 2 之前的版本)，請閱讀以下額外資訊：

Amazon Linux AMI 平台特定的命名空間

在 Amazon Linux AMI 平台分支上，靜態檔案組態命名空間因平台而異。如需詳細資訊，請參閱下列其中一個頁面：

- Go 組態命名空間
- Java SE 組態命名空間
- Tomcat 組態命名空間
- Node.js 組態命名空間
- Python 組態命名空間

為您的 Elastic Beanstalk 環境設定 HTTPS

如果您已經為您的 Elastic Beanstalk 環境購買和設定自訂網域名稱，可以使用 HTTPS 來讓使用者安全地連線到您的網站。如果您未擁有網域名稱，仍可使用 HTTPS 搭配自我簽署的憑證，以進行開發和測試。對於傳送使用者資料或登入資訊的任何應用程式而言，HTTPS 是必須使用的機制。

搭配 Elastic Beanstalk 環境使用 HTTPS 最簡單的方法，就是指派伺服器憑證給您環境的負載平衡器。當您設定負載平衡器來終止 HTTPS 時，用戶端和負載平衡器之間的連線是安全的。負載平衡器和 EC2 執行個體之間的後端連線使用 HTTP，因此不需再額外設定執行個體。

> **Note**
>
> 使用 AWS Certificate Manager (ACM)，您可以為您的網域名稱免費建立受信任的憑證。ACM 憑證只能與 AWS 負載平衡器和 Amazon CloudFront 分佈搭配使用，且 ACM 只能在特定 AWS 區域使用。
> 若要將 ACM 憑證與 Elastic Beanstalk 搭配使用，請參閱設定您 Elastic Beanstalk 環境的負載平衡器來終止 HTTPS。

如果您是在單一執行個體的環境中執行應用程式，或需要保護整個連線 (一直到負載平衡器後方的 EC2 執行個體)，您可以設定執行個體上執行的代理伺服器，來終止 HTTPS。若要設定您的執行個體終止 HTTPS 連線，需使用組態檔案來修改執行個體上所執行的軟體，並修改安全群組以實現安全的連線。

如果是負載平衡環境中的端對端 HTTPS，您可以結合執行個體與負載平衡器的終止動作，來加密這兩種連線。在預設情況下，如果您設定負載平衡器使用 HTTPS 來將傳輸資料轉傳，負載平衡器將會信任後端執行個體所提供的任何憑證。為實現最高的安全性，您可以將政策套用到負載平衡器，來防止負載平衡器連線到未提供受信任公有憑證的執行個體。

> **Note**
>
> 您也可以設定負載平衡器，在不解密的狀態下轉傳 HTTPS 的傳輸資料。這項方法的缺點是負載平衡器無法查看請求，因此無法將轉傳作業最佳化和報告回應指標。

如果您所在的區域未提供 ACM 服務，您可以向第三方購買受信任的憑證。第三方憑證可用來針對您負載平衡器或後端執行個體或這兩者上的 HTTPS 傳輸資料，進行解密的動作。

若要進行開發與測試，您可以使用開放原始碼工具，來自行建立和簽署憑證。自我簽署的憑證易於建立而且免費，但無法用於公有網站上的前端解密。如果您試著針對與用戶端的 HTTPS 連線來使用自我簽署的憑證，則使用者的瀏覽器將會顯示錯誤訊息，告知您的網站並不安全。不過，您可以使用自我簽署的憑證來保護後端連線，而不會出現問題。

ACM 是佈建、管理和部署您伺服器憑證的慣用工具，其以程式設計方式或使用 AWS CLI。如果 ACM 無法在您的 AWS 區域中使用，您可以使用 AWS CLI 上傳第三方或自簽憑證和私密金鑰至 AWS

Identity and Access Management (IAM)。儲存於 IAM 中的憑證可搭配負載平衡器和 CloudFront 分佈使用。

> **Note**
>
> GitHub 上的 Does it have Snakes? 範例應用程式，針對使用 Tomcat Web 應用程式來設定 HTTPS 的每項方法，包含了相關的組態檔案和說明。如需詳細資訊，請參閱 readme 檔案與 HTTPS 說明。

主題

- 建立和簽署 X509 憑證
- 上傳憑證到 IAM
- 設定您 Elastic Beanstalk 環境的負載平衡器來終止 HTTPS
- 於執行個體設定您的應用程式以終止 HTTPS 連線
- 在負載平衡的 Elastic Beanstalk 環境中設定端對端加密
- 針對 TCP 傳遞設定您環境的負載平衡器
- 將私有金鑰安全地儲存於 Amazon S3 中
- 將 HTTP 設定為 HTTPS 重新導向

建立和簽署 X509 憑證

您可以使用 OpenSSL 來建立您應用程式適用的 X509 憑證。OpenSSL 是一個標準的開放原始碼程式庫，支援多種加密功能，包括建立和簽署 x509 憑證。如需關於 OpenSSL 的詳細資訊，請造訪 www.openssl.org。

> **Note**
>
> 如果您想在單一執行個體的環境中使用 HTTPS，或是使用自我簽署的憑證來在後端重新加密，只需要在本機建立憑證即可。如果您有自己的網域名稱，可以在 AWS 中建立憑證，然後使用 AWS Certificate Manager (ACM)，在負載平衡的環境中免費使用該憑證。如需相關說明，請參閱 AWS Certificate Manager 使用者指南中的請求憑證。

在命令列中執行 `openssl version`，來檢視是否已安裝 OpenSSL。如果尚未安裝，您可以根據公有 GitHub 儲存庫的說明，或使用您偏好的套件軟體管理工具，來建置和安裝原始程式碼。OpenSSL 也會

安裝在 Elastic Beanstalk 的 Linux 映像上，所以一項快速的替代方法是使用 EB CLI 的 eb ssh 命令，來連線到正在運作環境中的 EC2 執行個體：

```
~/eb$ eb ssh
[ec2-user@ip-255-55-55-255 ~]$ openssl version
OpenSSL 1.0.1k-fips 8 Jan 2015
```

您需要建立 RSA 私有金鑰，來產生您的憑證簽署請求 (CSR)。若要建立您的私密金鑰，請使用 openssl genrsa 命令：

```
[ec2-user@ip-255-55-55-255 ~]$ openssl genrsa 2048 > privatekey.pem
Generating RSA private key, 2048 bit long modulus
.....................................................................................
+++
..............+++
e is 65537 (0x10001)
```

privatekey.pem

> 您想要在其中儲存私有金鑰的檔案的名稱。一般而言，openssl genrsa 命令會將私密金鑰的內容印出到畫面上，但此命令會將輸出內容導向檔案。選擇任意的檔案名稱，並將檔案儲存到安全的位置，以供日後擷取。如果您遺失私有金鑰，就無法使用您的憑證。

CSR 是一種檔案，您會將此等檔案傳送到憑證認證機構 (CA)，以申請數位伺服器憑證。若要建立 CSR，請使用 openssl req 命令：

```
$ openssl req -new -key privatekey.pem -out csr.pem
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
```

輸入要求填寫的資訊，然後按下 Enter (Enter) 鍵。下表說明和顯示了每個欄位的範例：

名稱	描述	範例
Country Name (國家/地區名稱)	兩個字母的 ISO 縮寫，用來代表您的國家/地區。	US = 美國
State or Province (州或省)	您組織位在的州名或省名。此名稱不得縮寫。	華盛頓州
Locality Name (地區名稱)	您組織所在城市的名稱。	西雅圖
Organization Name (組織名稱)	您組織的完整法定名稱。請不要使用您組織名稱的縮寫。	範例公司
Organizational Unit (組織單位)	選填，這是額外的組織資訊。	行銷部門
Common Name (通用名稱)	您網站的完整網域名稱。此名稱必須符合使用者造訪您網站時所看到的網域名稱，否則將會顯示憑證錯誤。	www.example.com
電子郵件地址	網站管理員的電子郵件地址。	someone@example.com

您可以向第三方提交簽署請求來進行簽署，或是針對開發和測試用途自行簽署。自我簽署的憑證也可用於負載平衡器與 EC2 執行個體之間的後端 HTTPS。

若要簽署憑證，請使用 openssl x509 命令。下例使用前一步驟的私密金鑰 (*privatekey.pem*) 和簽署請求 (*csr.pem*)，建立名為 *public.crt* 的公有憑證，其有效期限為 *365* 天。

```
$ openssl x509 -req -days 365 -in csr.pem -signkey privatekey.pem -out public.crt
Signature ok
subject=/C=us/ST=washington/L=seattle/O=example corporation/OU=marketing/
CN=www.example.com/emailAddress=someone@example.com
Getting Private key
```

保留私有金鑰和公有憑證以供日後使用。您可以放棄簽署請求。一律將私密金鑰儲存於安全的位置，並且避免將該金鑰加入您的原始程式碼中。

若要在 Windows Server 平台上使用憑證，您必須將憑證轉換為 PFX 格式。利用下列指令，從私有金鑰和公有憑證檔案來產生 PFX 憑證：

```
$ openssl pkcs12 -export -out example.com.pfx -inkey privatekey.pem -in public.crt
Enter Export Password: password
Verifying - Enter Export Password: password
```

當您擁有憑證後，可以將憑證上傳至 IAM，以搭配負載平衡器使用，或是在您的環境中設定執行個體以終止 HTTPS。

上傳憑證到 IAM

若要搭配 Elastic Beanstalk 環境的負載平衡器使用您的憑證，請將憑證及私有金鑰上傳至 AWS Identity and Access Management (IAM)。您可以使用儲存在 IAM 中的憑證，搭配 Elastic Load Balancing 負載平衡器和 Amazon CloudFront 分佈使用。

> **Note**
>
> AWS Certificate Manager (ACM) 為佈建、管理和部署您伺服器憑證的首選工具。如需請求 ACM 憑證的詳細資訊，請參閱《AWS Certificate Manager 使用者指南》中的請求憑證。如需有關將第三方憑證匯入 ACM 的詳細資訊，請參閱《AWS Certificate Manager 使用者指南》中的匯入憑證。唯有當 ACM 無法在您的 AWS 區域中使用時，才使用 IAM 上傳憑證。

您可以使用 AWS Command Line Interface (AWS CLI) 上傳憑證。下列命令會上傳名為 *https-cert.crt* 的自我簽署憑證，以及名為 *private-key.pem* 的私密金鑰：

```
$ aws iam upload-server-certificate --server-certificate-name elastic-beanstalk-x509 --
certificate-body file://https-cert.crt --private-key file://private-key.pem
{
    "ServerCertificateMetadata": {
        "ServerCertificateId": "AS5YBEIONO2Q7CAIHKNGC",
        "ServerCertificateName": "elastic-beanstalk-x509",
        "Expiration": "2017-01-31T23:06:22Z",
        "Path": "/",
        "Arn": "arn:aws:iam::123456789012:server-certificate/elastic-beanstalk-x509",
        "UploadDate": "2016-02-01T23:10:34.167Z"
    }
}
```

`file://` 字首會通知 AWS CLI 載入目前目錄中的檔案內容。*`elastic-beanstalk-x509`* 則指定 IAM 內憑證的名稱。

若您自憑證授權機構購買憑證並收到憑證鏈檔案，請納入 `--certificate-chain` 選項將其一同上傳：

```
$ aws iam upload-server-certificate --server-certificate-name elastic-beanstalk-x509 --
certificate-chain file://certificate-chain.pem --certificate-body file://https-cert.crt
 --private-key file://private-key.pem
```

請記下您的憑證的 Amazon Resource Name (ARN)。更新負載平衡器組態設定以使用 HTTPS 時，將使用該 ARN。

> **Note**
>
> 上傳到 IAM 的憑證將持續存放，即使未來不會在任何環境的負載平衡器中使用。其中包含敏感資料。如果任何環境都不再需要此憑證，請務必刪除它。如需關於從 IAM 刪除憑證的詳細資訊，請參閱https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_server-certs.html#delete-server-certificate。

如需關於在 IAM 中的伺服器憑證之詳細資訊，請參閱IAM 使用者指南中的使用伺服器憑證。

設定您 Elastic Beanstalk 環境的負載平衡器來終止 HTTPS

若要將 AWS Elastic Beanstalk 環境更新為使用 HTTPS，您需要為環境中的負載平衡器設定 HTTPS 接聽程式。兩種負載平衡器類型支援 HTTPS 接聽程式：Classic Load Balancer 和 Application Load Balancer。

您可以使用 Elastic Beanstalk 主控台或組態檔來設定安全接聽程式，並指派憑證。

> **Note**
>
> 單一執行個體環境沒有負載平衡器，不在負載平衡器支援 HTTPS 終止。

使用 Elastic Beanstalk 主控台設定安全接聽程式

若要指派憑證給您環境中的負載平衡器

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇組態。
4. 在 Load balancer (負載平衡器) 組態類別中，選擇 Edit (編輯)。

> **Note**
>
> 如果 Load balancer (負載平衡器) 組態類別沒有 Edit (編輯) 按鈕，您的環境便沒有負載平衡器。

5. 在修改負載平衡器頁面，程序會因與您的環境關聯的負載平衡器類型而不同。
   - Classic Load Balancer
     a. 選擇 Add listener (新增接聽程式)。
     b. 在 Classic Load Balancer listener (Classic Load Balancer 接聽程式) 對話方塊中，設定下列設定：
        - 對於 Listener port (接聽程式連接埠)，輸入傳入流量連接埠，通常為 443。
        - 對於 Listener protocol (接聽程式協定)，選擇 HTTPS 。
        - 對於 Instance type (執行個體連接埠)，輸入 80。
        - 對於 Instance protocol (執行個體協定)，選擇 HTTP。
        - 對於 SSL certificate (SSL 憑證)，選擇您的憑證。
     c. 選擇新增。
   - Application Load Balancer
     a. 選擇 Add listener (新增接聽程式)。
     b. 在 Application Load Balancer listener (Application Load Balancer 接聽程式) 對話方塊中，設定下列設定：

- 在 Port (連接埠) 中輸入傳入流量連接埠，通常為 443。
- 請在 Protocol (通訊協定) 中選擇 HTTPS。
- 對於 SSL certificate (SSL 憑證)，選擇您的憑證。

c. 選擇新增。

> **Note**
>
> 若是使用 Classic Load Balancer 和 Application Load Balancer，如果下拉式選單未顯示任何憑證，您應該在 AWS Certificate Manager (ACM) (偏好) 中為自訂網域名稱建立或上傳憑證。或是使用 AWS CLI將憑證上傳到 IAM。

- Network Load Balancer

a. 選擇 Add listener (新增接聽程式)。

b. 在 Network Load Balancer listener (Network Load Balancer 接聽程式) 對話方塊中，對於 Port (連接埠)，輸入傳入流量連接埠，通常是 443。

c. 選擇新增。

6. 若要儲存變更，請選擇頁面底部的儲存變更。

使用組態檔案設定安全接聽程式

您可以利用以下其中一個組態檔設定負載平衡器的安全接聽程式。

Example .ebextensions/securelistener-clb.config

當您的環境具有 Classic Load Balancer 時，請使用此範例。範例使用了 `aws:elb:listener` 命名空間中的選項，以指定的憑證設定 443 埠的 HTTPS 接聽程式，並透過 80 通訊埠，將加密的傳輸資料轉傳給您環境中的執行個體。

```
option_settings:
  aws:elb:listener:443:
    SSLCertificateId: arn:aws:acm:us-east-2:1234567890123:certificate/
####################################
    ListenerProtocol: HTTPS
    InstancePort: 80
```

用您憑證的 ARN 來取代反白顯示的文字。憑證可以是您在 AWS Certificate Manager (ACM) 中建立或上傳的憑證 (偏好)，也可以是您使用 AWS CLI.

如需 Classic Load Balancer 設定選項的詳細資訊，請參閱Classic Load Balancer 組態命名空間。

Example .ebextensions/securelistener-alb.config

當您的環境具有 Application Load Balancer 時，請使用此範例。範例使用在 `aws:elbv2:listener` 命名空間中的選項，使用指定的憑證在連接埠 443 上設定 HTTPS 接聽程式。此接聽程式將流量路由傳送到預設的程序。

```
option_settings:
  aws:elbv2:listener:443:
    ListenerEnabled: 'true'
    Protocol: HTTPS
    SSLCertificateArns: arn:aws:acm:us-east-2:1234567890123:certificate/
###################################
```

Example .ebextensions/securelistener-nlb.config

當您的環境具有 Network Load Balancer 時，請使用此範例。此範例使用 `aws:elbv2:listener` 命名空間中的選項來設定連接埠 443 上的接聽程式。此接聽程式將流量路由傳送到預設的程序。

```
option_settings:
  aws:elbv2:listener:443:
    ListenerEnabled: 'true'
```

設定安全群組

如果您設定負載平衡器，將傳輸資料轉傳到第 80 埠以外的執行個體通訊埠，則您必須新增安全群組的規則，允許從您的負載平衡器通過執行個體通訊埠傳入資料。如果您在自訂 VPC 中建立環境，Elastic Beanstalk 會為您新增此規則。

您可以在您應用程式的 `.ebextensions` 目錄中，將 `Resources` 金鑰加入組態檔，來新增這項規則。

以下範例組態檔案將傳入規則新增至 `AWSEBSecurityGroup` 安全群組。如此將允許來自負載平衡器安全群組的連接埠 1000 流量。

Example .ebextensions/sg-ingressfromlb.config

```
Resources:
```

```
  sslSecurityGroupIngress:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: {"Fn::GetAtt" : ["AWSEBSecurityGroup", "GroupId"]}
      IpProtocol: tcp
      ToPort: 1000
      FromPort: 1000
      SourceSecurityGroupId: {"Fn::GetAtt" : ["AWSEBLoadBalancerSecurityGroup",
 "GroupId"]}
```

於執行個體設定您的應用程式以終止 HTTPS 連線

您可以使用組態檔案，針對將流量傳入您應用程式的代理伺服器進行設定，以終止 HTTPS 連線。若您希望於單一執行個體環境使用 HTTPS，或者將流量設定為透過負載平衡器傳輸而無須解密，此功能十分實用。

若要啟用 HTTPS，您必須允許連接埠 443 的流量傳入運作您 Elastic Beanstalk 應用程式的 EC2 執行個體。您可於組態檔案中使用 `Resources` 金鑰，於 AWSEBSecurityGroup 安全群組傳入規則新增連接埠 443 的規則，達成此目標。

下列片段將傳入規則新增至 `AWSEBSecurityGroup` 安全群組，於單一執行個體環境開放連接埠 443 的所有流量：

`.ebextensions/https-instance-securitygroup.config`

```
Resources:
  sslSecurityGroupIngress:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: {"Fn::GetAtt" : ["AWSEBSecurityGroup", "GroupId"]}
      IpProtocol: tcp
      ToPort: 443
      FromPort: 443
      CidrIp: 0.0.0.0/0
```

在預設 Amazon Virtual Private Cloud (Amazon VPC) 的負載平衡環境中，您可以將此政策修改為僅接受來自負載平衡器的流量。如需範例，請參閱 在負載平衡的 Elastic Beanstalk 環境中設定端對端加密。

平台

- 在執行 Docker 的 EC2 執行個體上終止 HTTPS

- 在執行 Go 的 EC2 執行個體上終止 HTTPS
- 在執行 Java SE 的 EC2 執行個體上終止 HTTPS
- 在執行 Node.js 的 EC2 執行個體上終止 HTTPS
- 在執行 PHP 的 EC2 執行個體上終止 HTTPS
- 在執行 Python 的 EC2 執行個體上終止 HTTPS
- 在執行 Ruby 的 EC2 執行個體上終止 HTTPS
- 在執行 Tomcat 的 EC2 執行個體上終止 HTTPS
- 在執行 .NET Core on Linux 的 Amazon EC2 執行個體上終止 HTTPS
- 在執行 .NET 的 Amazon EC2 執行個體上終止 HTTPS

在執行 Docker 的 EC2 執行個體上終止 HTTPS

針對 Docker 容器，您可使用組態檔案來啟用 HTTPS。

在您的組態檔案中加入下列的程式碼片段、依照指示來更換憑證與私有金鑰資料，並儲存於原始碼套件的 `.ebextensions` 目錄中。此組態檔案會執行下列任務：

- `files` 金鑰會於執行個體上建立下列檔案：

  `/etc/nginx/conf.d/https.conf`

  設定 nginx 伺服器。nginx 服務開始後，即會載入此檔案。

  `/etc/pki/tls/certs/server.crt`

  在執行個體上建立憑證檔案。將`######`取代為您的憑證內容。

  > **Note**
  >
  > YAML 憑藉一致的縮排。請在取代範例組態檔中的內容時，讓縮排層級一致，並確認您的文字編輯器使用空格而非定位字元進行縮排。

  如果您有中繼憑證，請在網站憑證後將其納入 `server.crt` 中。

```
    -----BEGIN CERTIFICATE-----
certificate file contents
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
```

```
first intermediate certificate
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
second intermediate certificate
-----END CERTIFICATE-----
```

`/etc/pki/tls/certs/server.key`

在執行個體上建立私有金鑰。將######取代為用於建立憑證請求或自我簽署憑證的私密金鑰內容。

Example .ebextensions/https-instance.config

```
files:
  /etc/nginx/conf.d/https.conf:
    mode: "000644"
    owner: root
    group: root
    content: |
      # HTTPS Server

      server {
        listen 443;
        server_name localhost;

        ssl on;
        ssl_certificate /etc/pki/tls/certs/server.crt;
        ssl_certificate_key /etc/pki/tls/certs/server.key;

        ssl_session_timeout 5m;

        ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
        ssl_prefer_server_ciphers on;

        location / {
          proxy_pass http://docker;
          proxy_http_version 1.1;

          proxy_set_header Connection "";
          proxy_set_header Host $host;
          proxy_set_header X-Real-IP $remote_addr;
          proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
```

```
          proxy_set_header X-Forwarded-Proto https;
        }
      }

  /etc/pki/tls/certs/server.crt:
    mode: "000400"
    owner: root
    group: root
    content: |
      -----BEGIN CERTIFICATE-----
      certificate file contents
      -----END CERTIFICATE-----

  /etc/pki/tls/certs/server.key:
    mode: "000400"
    owner: root
    group: root
    content: |
      -----BEGIN RSA PRIVATE KEY-----
      private key contents # See note below.
      -----END RSA PRIVATE KEY-----
```

> **Note**
>
> 請避免將包含您私有金鑰的組態檔遞交到原始檔控制。當您測試好組態並確認其正常運作後，請將您的私有金鑰儲存在 Amazon S3，然後修改組態，以在部署期間予以下載。如需指示，請參閱 將私有金鑰安全地儲存於 Amazon S3 中。

在單一執行個體環境中，您也必須修改執行個體的安全群組，以允許連接埠 443 上的流量。下列組態檔案會使用 AWS CloudFormation 函數擷取安全群組 ID，並對其新增規則。

Example .ebextensions/https-instance-single.config

```
Resources:
  sslSecurityGroupIngress:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: {"Fn::GetAtt" : ["AWSEBSecurityGroup", "GroupId"]}
      IpProtocol: tcp
      ToPort: 443
      FromPort: 443
```

```
      CidrIp: 0.0.0.0/0
```

針對使用負載平衡的環境，您應設定負載平衡器，使其以未處理之方式通過安全流量，或針對端點對端點加密進行解密與重新加密。

在執行 Go 的 EC2 執行個體上終止 HTTPS

如果是 Go 容器類型，您可使用組態檔案來啟用 HTTPS，以及使用 nginx 組態檔案來設定 nginx 伺服器使用 HTTPS。

在您的組態檔案中加入下列的程式碼片段、依照指示來更換憑證與私有金鑰佔位符，並儲存於原始碼套件的 `.ebextensions` 目錄中。此組態檔案會執行下列任務：

- `Resources` 金鑰可於連接埠 443 上啟用安全群組，供您環境的執行個體使用。
- `files` 金鑰會於執行個體上建立下列檔案：

  `/etc/pki/tls/certs/server.crt`

  在執行個體上建立憑證檔案。將`######`取代為您的憑證內容。

  > **Note**
  >
  > YAML 憑藉一致的縮排。請在取代範例組態檔中的內容時，讓縮排層級一致，並確認您的文字編輯器使用空格而非定位字元進行縮排。

  如果您有中繼憑證，請在網站憑證後將其納入 `server.crt` 中。

```
    -----BEGIN CERTIFICATE-----
certificate file contents
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
first intermediate certificate
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
second intermediate certificate
-----END CERTIFICATE-----
```

`/etc/pki/tls/certs/server.key`

在執行個體上建立私有金鑰。將######取代為用於建立憑證請求或自我簽署憑證的私密金鑰內容。

- `container_commands` 金鑰會在所有項目設定完成後重新啟動 nginx 伺服器，以讓伺服器載入 nginx 組態檔案。

Example .ebextensions/https-instance.config

```
files:
  /etc/pki/tls/certs/server.crt:
    content: |
      -----BEGIN CERTIFICATE-----
      certificate file contents
      -----END CERTIFICATE-----

  /etc/pki/tls/certs/server.key:
    content: |
      -----BEGIN RSA PRIVATE KEY-----
      private key contents # See note below.
      -----END RSA PRIVATE KEY-----

container_commands:
  01restart_nginx:
    command: "service nginx restart"
```

> **Note**
>
> 請避免將包含您私有金鑰的組態檔遞交到原始檔控制。當您測試好組態並確認其正常運作後，請將您的私有金鑰儲存在 Amazon S3，然後修改組態，以在部署期間予以下載。如需指示，請參閱 將私有金鑰安全地儲存於 Amazon S3 中。

將下列內容加入您原始碼套件 `.conf` 目錄中的檔案，該檔案的副檔名為 `.ebextensions/nginx/conf.d/` (例如 `.ebextensions/nginx/conf.d/https.conf`)。將 *`app_port`* 換成您應用程式接聽的埠號。本範例設定 nginx 伺服器使用 SSL 來接聽 443 埠。關於 Go 平台上的這些組態檔案，詳細資訊請參閱 設定反向代理程式。

Example .ebextensions/nginx/conf.d/https.conf

```
# HTTPS server

server {
    listen       443;
    server_name  localhost;

    ssl                  on;
    ssl_certificate      /etc/pki/tls/certs/server.crt;
    ssl_certificate_key  /etc/pki/tls/certs/server.key;

    ssl_session_timeout  5m;

    ssl_protocols  TLSv1 TLSv1.1 TLSv1.2;
    ssl_prefer_server_ciphers   on;

    location / {
        proxy_pass  http://localhost:app_port;
        proxy_set_header   Connection "";
        proxy_http_version 1.1;
        proxy_set_header        Host            $host;
        proxy_set_header        X-Real-IP       $remote_addr;
        proxy_set_header        X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header        X-Forwarded-Proto https;
    }
}
```

在單一執行個體環境中，您也必須修改執行個體的安全群組，以允許連接埠 443 上的流量。下列組態檔案會使用 AWS CloudFormation 函數擷取安全群組 ID，並對其新增規則。

Example .ebextensions/https-instance-single.config

```
Resources:
  sslSecurityGroupIngress:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: {"Fn::GetAtt" : ["AWSEBSecurityGroup", "GroupId"]}
      IpProtocol: tcp
      ToPort: 443
      FromPort: 443
      CidrIp: 0.0.0.0/0
```

針對使用負載平衡的環境，您應設定負載平衡器，使其以未處理之方式通過安全流量，或針對端點對端點加密進行解密與重新加密。

在執行 Java SE 的 EC2 執行個體上終止 HTTPS

如果是 Java SE 容器類型，您可使用 .ebextensions 組態檔案來啟用 HTTPS，以及使用 nginx 組態檔案來設定 nginx 伺服器使用 HTTPS。

所有 AL2023/AL2 平台皆支援統一的代理組態功能。如需有關在執行 AL2023/AL2 的平台版本上設定代理伺服器的詳細資訊，請展開the section called “擴充 Linux 平台”中的反向代理組態一節。

在您的組態檔案中加入下列的程式碼片段、依照指示來更換憑證與私有金鑰佔位符，並儲存於 `.ebextensions` 目錄中。此組態檔案會執行下列任務：

- `files` 金鑰會於執行個體上建立下列檔案：

  `/etc/pki/tls/certs/server.crt`

  在執行個體上建立憑證檔案。將`######`取代為您的憑證內容。

  > **Note**
  >
  > YAML 憑藉一致的縮排。請在取代範例組態檔中的內容時，讓縮排層級一致，並確認您的文字編輯器使用空格而非定位字元進行縮排。

  如果您有中繼憑證，請在網站憑證後將其納入 `server.crt` 中。

  ```
      -----BEGIN CERTIFICATE-----
  certificate file contents
  -----END CERTIFICATE-----
  -----BEGIN CERTIFICATE-----
  first intermediate certificate
  -----END CERTIFICATE-----
  -----BEGIN CERTIFICATE-----
  second intermediate certificate
  -----END CERTIFICATE-----
  ```

  `/etc/pki/tls/certs/server.key`

  在執行個體上建立私有金鑰。將`######`取代為用於建立憑證請求或自我簽署憑證的私密金鑰內容。

- container_commands 金鑰會在所有項目設定完成後重新啟動 nginx 伺服器，以讓伺服器載入 nginx 組態檔案。

Example .ebextensions/https-instance.config

```
files:
  /etc/pki/tls/certs/server.crt:
    content: |
      -----BEGIN CERTIFICATE-----
      certificate file contents
      -----END CERTIFICATE-----

  /etc/pki/tls/certs/server.key:
    content: |
      -----BEGIN RSA PRIVATE KEY-----
      private key contents # See note below.
      -----END RSA PRIVATE KEY-----

container_commands:
  01restart_nginx:
    command: "service nginx restart"
```

> **Note**
>
> 請避免將包含您私有金鑰的組態檔遞交到原始檔控制。當您測試好組態並確認其正常運作後，請將您的私有金鑰儲存在 Amazon S3，然後修改組態，以在部署期間予以下載。如需說明，請參閱 將私有金鑰安全地儲存於 Amazon S3 中。

將下列內容加入您原始碼套件 .conf 目錄中的檔案，該檔案的副檔名為 .ebextensions/nginx/conf.d/ (例如 .ebextensions/nginx/conf.d/https.conf)。將 *app_port* 換成您應用程式接聽的埠號。本範例設定 nginx 伺服器使用 SSL 來接聽 443 埠。關於 Java SE 平台上的這些組態檔案，詳細資訊請參閱 設定反向代理程式。

Example .ebextensions/nginx/conf.d/https.conf

```
# HTTPS server

server {
    listen       443;
```

```
    server_name  localhost;

    ssl                  on;
    ssl_certificate      /etc/pki/tls/certs/server.crt;
    ssl_certificate_key  /etc/pki/tls/certs/server.key;

    ssl_session_timeout  5m;

    ssl_protocols  TLSv1 TLSv1.1 TLSv1.2;
    ssl_prefer_server_ciphers   on;

    location / {
        proxy_pass  http://localhost:app_port;
        proxy_set_header   Connection "";
        proxy_http_version 1.1;
        proxy_set_header        Host            $host;
        proxy_set_header        X-Real-IP       $remote_addr;
        proxy_set_header        X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header        X-Forwarded-Proto https;
    }
}
```

在單一執行個體環境中，您也必須修改執行個體的安全群組，以允許連接埠 443 上的流量。下列組態檔案會使用 AWS CloudFormation 函數擷取安全群組 ID，並對其新增規則。

Example .ebextensions/https-instance-single.config

```
Resources:
  sslSecurityGroupIngress:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: {"Fn::GetAtt" : ["AWSEBSecurityGroup", "GroupId"]}
      IpProtocol: tcp
      ToPort: 443
      FromPort: 443
      CidrIp: 0.0.0.0/0
```

針對使用負載平衡的環境，您應設定負載平衡器，使其以未處理之方式通過安全流量，或針對端點對端點加密進行解密與重新加密。

在執行 Node.js 的 EC2 執行個體上終止 HTTPS

下列範例組態檔案會延伸預設 nginx 組態，以透過公有憑證與私密金鑰來接聽連接埠 443 並終止 SSL/TLS 連線。

如果您已設定環境來啟用增強型運作狀態報告，則需要設定 nginx 來產生存取日誌。若要這麼做，請移除 `# For enhanced health...` 這行註解下方各行開頭的 # 字元，以取消註解整個行區塊。

Example .ebextensions/https-instance.config

```
files:
  /etc/nginx/conf.d/https.conf:
    mode: "000644"
    owner: root
    group: root
    content: |
      # HTTPS server

      server {
          listen       443;
          server_name  localhost;

          ssl                  on;
          ssl_certificate      /etc/pki/tls/certs/server.crt;
          ssl_certificate_key  /etc/pki/tls/certs/server.key;

          ssl_session_timeout  5m;

          ssl_protocols  TLSv1 TLSv1.1 TLSv1.2;
          ssl_prefer_server_ciphers   on;

          # For enhanced health reporting support, uncomment this block:

          #if ($time_iso8601 ~ "^(\d{4})-(\d{2})-(\d{2})T(\d{2})") {
          #    set $year $1;
          #    set $month $2;
          #    set $day $3;
          #    set $hour $4;
          #}
          #access_log /var/log/nginx/healthd/application.log.$year-$month-$day-$hour
 healthd;
          #access_log  /var/log/nginx/access.log  main;
```

```
        location / {
            proxy_pass  http://nodejs;
            proxy_set_header   Connection "";
            proxy_http_version 1.1;
            proxy_set_header        Host            $host;
            proxy_set_header        X-Real-IP       $remote_addr;
            proxy_set_header        X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header        X-Forwarded-Proto https;
        }
    }

/etc/pki/tls/certs/server.crt:
  mode: "000400"
  owner: root
  group: root
  content: |
    -----BEGIN CERTIFICATE-----
    certificate file contents
    -----END CERTIFICATE-----

/etc/pki/tls/certs/server.key:
  mode: "000400"
  owner: root
  group: root
  content: |
    -----BEGIN RSA PRIVATE KEY-----
    private key contents # See note below.
    -----END RSA PRIVATE KEY-----
```

`files` 金鑰會於執行個體上建立下列檔案：

`/etc/nginx/conf.d/https.conf`

設定 nginx 伺服器。nginx 服務開始後，即會載入此檔案。

`/etc/pki/tls/certs/server.crt`

在執行個體上建立憑證檔案。將######取代為您的憑證內容。

> **Note**
>
> YAML 憑藉一致的縮排。請在取代範例組態檔中的內容時，讓縮排層級一致，並確認您的文字編輯器使用空格而非定位字元進行縮排。

如果您有中繼憑證，請在網站憑證後將其納入 server.crt 中。

```
    -----BEGIN CERTIFICATE-----
certificate file contents
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
first intermediate certificate
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
second intermediate certificate
-----END CERTIFICATE-----
```

/etc/pki/tls/certs/server.key

在執行個體上建立私有金鑰。將######取代為用於建立憑證請求或自我簽署憑證的私密金鑰內容。

> **Note**
>
> 請避免將包含您私有金鑰的組態檔遞交到原始檔控制。當您測試好組態並確認其正常運作後，請將您的私有金鑰儲存在 Amazon S3，然後修改組態，以在部署期間予以下載。如需指示，請參閱 將私有金鑰安全地儲存於 Amazon S3 中。

在單一執行個體環境中，您也必須修改執行個體的安全群組，以允許連接埠 443 上的流量。下列組態檔案會使用 AWS CloudFormation 函數擷取安全群組 ID，並對其新增規則。

Example .ebextensions/https-instance-single.config

```
Resources:
  sslSecurityGroupIngress:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: {"Fn::GetAtt" : ["AWSEBSecurityGroup", "GroupId"]}
      IpProtocol: tcp
      ToPort: 443
      FromPort: 443
      CidrIp: 0.0.0.0/0
```

針對使用負載平衡的環境，您應設定負載平衡器，使其以未處理之方式通過安全流量，或針對端點對端點加密進行解密與重新加密。

在執行 PHP 的 EC2 執行個體上終止 HTTPS

以 PHP 容器類型而言，您可使用組態檔案，讓 Apache HTTP Server 使用 HTTPS。

在您的組態檔案中加入下列的程式碼片段、依照指示來更換憑證與私有金鑰資料，並儲存於原始碼套件的 `.ebextensions` 目錄中。

此組態檔案會執行下列任務：

- `packages` 金鑰使用 yum 來安裝 `mod24_ssl`。
- `files` 金鑰會於執行個體上建立下列檔案：

  `/etc/httpd/conf.d/ssl.conf`

  設定 Apache 伺服器。Apache 服務開始後，即會載入此檔案。

  `/etc/pki/tls/certs/server.crt`

  在執行個體上建立憑證檔案。將`######`取代為您的憑證內容。

  > **Note**
  >
  > YAML 憑藉一致的縮排。請在取代範例組態檔中的內容時，讓縮排層級一致，並確認您的文字編輯器使用空格而非定位字元進行縮排。

  如果您有中繼憑證，請在網站憑證後將其納入 `server.crt` 中。

```
    -----BEGIN CERTIFICATE-----
certificate file contents
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
first intermediate certificate
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
second intermediate certificate
-----END CERTIFICATE-----
```

`/etc/pki/tls/certs/server.key`

在執行個體上建立私有金鑰。將######取代為用於建立憑證請求或自我簽署憑證的私密金鑰內容。

Example .ebextensions/https-instance.config

```
packages:
  yum:
    mod24_ssl : []

files:
  /etc/httpd/conf.d/ssl.conf:
    mode: "000644"
    owner: root
    group: root
    content: |
      LoadModule ssl_module modules/mod_ssl.so
      Listen 443
      <VirtualHost *:443>
        <Proxy *>
          Order deny,allow
          Allow from all
        </Proxy>

        SSLEngine             on
        SSLCertificateFile    "/etc/pki/tls/certs/server.crt"
        SSLCertificateKeyFile "/etc/pki/tls/certs/server.key"
        SSLCipherSuite        EECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH
        SSLProtocol           All -SSLv2 -SSLv3
        SSLHonorCipherOrder   On
        SSLSessionTickets     Off

        Header always set Strict-Transport-Security "max-age=63072000;
 includeSubdomains; preload"
        Header always set X-Frame-Options DENY
        Header always set X-Content-Type-Options nosniff

        ProxyPass / http://localhost:80/ retry=0
        ProxyPassReverse / http://localhost:80/
        ProxyPreserveHost on
        RequestHeader set X-Forwarded-Proto "https" early
```

```
    </VirtualHost>

/etc/pki/tls/certs/server.crt:
  mode: "000400"
  owner: root
  group: root
  content: |
    -----BEGIN CERTIFICATE-----
    certificate file contents
    -----END CERTIFICATE-----

/etc/pki/tls/certs/server.key:
  mode: "000400"
  owner: root
  group: root
  content: |
    -----BEGIN RSA PRIVATE KEY-----
    private key contents # See note below.
    -----END RSA PRIVATE KEY-----
```

> **Note**
>
> 請避免將包含您私有金鑰的組態檔遞交到原始檔控制。當您測試好組態並確認其正常運作後，請將您的私有金鑰儲存在 Amazon S3，然後修改組態，以在部署期間予以下載。如需指示，請參閱 將私有金鑰安全地儲存於 Amazon S3 中。

在單一執行個體環境中，您也必須修改執行個體的安全群組，以允許連接埠 443 上的流量。下列組態檔案會使用 AWS CloudFormation 函數擷取安全群組 ID，並對其新增規則。

Example .ebextensions/https-instance-single.config

```
Resources:
  sslSecurityGroupIngress:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: {"Fn::GetAtt" : ["AWSEBSecurityGroup", "GroupId"]}
      IpProtocol: tcp
      ToPort: 443
      FromPort: 443
      CidrIp: 0.0.0.0/0
```

針對使用負載平衡的環境，您應設定負載平衡器，使其以未處理之方式通過安全流量，或針對端點對端點加密進行解密與重新加密。

在執行 Python 的 EC2 執行個體上終止 HTTPS

透過 Web 伺服器閘道界面 (WSGI) 使用 Apache HTTP 伺服器的 Python 容器類型，您可使用組態檔案，讓 Apache HTTP 伺服器能夠使用 HTTPS。

在您的組態檔案中加入下列的程式碼片段、依照指示來更換憑證與私密金鑰資料，並儲存於原始碼套件的 `.ebextensions` 目錄中。此組態檔案會執行下列任務：

- `packages` 金鑰使用 yum 來安裝 `mod24_ssl`。
- `files` 金鑰會於執行個體上建立下列檔案：

  `/etc/httpd/conf.d/ssl.conf`

  設定 Apache 伺服器。若您的應用程式名稱不是 `application.py`，請將 `WSGIScriptAlias` 值中反白顯示的文字取代為應用程式的本機路徑。例如，django 應用程式的路徑可能為 `django/wsgi.py`。此位置應與您為環境設定的 `WSGIPath` 選項的值相符。

  依據您的應用程式需求，可能需要將其他目錄新增至 python-path 參數。

  `/etc/pki/tls/certs/server.crt`

  在執行個體上建立憑證檔案。將`######`取代為您的憑證內容。

  > **Note**
  >
  > YAML 憑藉一致的縮排。請在取代範例組態檔中的內容時，讓縮排層級一致，並確認您的文字編輯器使用空格而非定位字元進行縮排。

  如果您有中繼憑證，請在網站憑證後將其納入 `server.crt` 中。

```
    -----BEGIN CERTIFICATE-----
certificate file contents
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
first intermediate certificate
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
second intermediate certificate
```

```
-----END CERTIFICATE-----
```

`/etc/pki/tls/certs/server.key`

在執行個體上建立私有金鑰。將######取代為用於建立憑證請求或自我簽署憑證的私密金鑰內容。

- `container_commands` 金鑰會在所有項目設定完成後停止 httpd 服務，讓服務能夠使用新的 `https.conf` 檔案和憑證。

> **Note**
>
> 這個範例只能使用 Python 平台在環境中運作。

Example .ebextensions/https-instance.config

```
packages:
  yum:
    mod24_ssl : []

files:
  /etc/httpd/conf.d/ssl.conf:
    mode: "000644"
    owner: root
    group: root
    content: |
      LoadModule wsgi_module modules/mod_wsgi.so
      WSGIPythonHome /opt/python/run/baselinenv
      WSGISocketPrefix run/wsgi
      WSGIRestrictEmbedded On
      Listen 443
      <VirtualHost *:443>
        SSLEngine on
        SSLCertificateFile "/etc/pki/tls/certs/server.crt"
        SSLCertificateKeyFile "/etc/pki/tls/certs/server.key"

        Alias /static/ /opt/python/current/app/static/
        <Directory /opt/python/current/app/static>
        Order allow,deny
        Allow from all
        </Directory>
```

```
        WSGIScriptAlias / /opt/python/current/app/application.py

        <Directory /opt/python/current/app>
        Require all granted
        </Directory>

        WSGIDaemonProcess wsgi-ssl processes=1 threads=15 display-name=%{GROUP} \
          python-path=/opt/python/current/app \
          python-home=/opt/python/run/venv \
          home=/opt/python/current/app \
          user=wsgi \
          group=wsgi
        WSGIProcessGroup wsgi-ssl

      </VirtualHost>

  /etc/pki/tls/certs/server.crt:
    mode: "000400"
    owner: root
    group: root
    content: |
      -----BEGIN CERTIFICATE-----
      certificate file contents
      -----END CERTIFICATE-----

  /etc/pki/tls/certs/server.key:
    mode: "000400"
    owner: root
    group: root
    content: |
      -----BEGIN RSA PRIVATE KEY-----
      private key contents # See note below.
      -----END RSA PRIVATE KEY-----

container_commands:
  01killhttpd:
    command: "killall httpd"
  02waitforhttpddeath:
    command: "sleep 3"
```

> **Note**
>
> 請避免將包含您私有金鑰的組態檔遞交到原始檔控制。當您測試好組態並確認其正常運作後，請將您的私有金鑰儲存在 Amazon S3，然後修改組態，以在部署期間予以下載。如需指示，請參閱 將私有金鑰安全地儲存於 Amazon S3 中。

在單一執行個體環境中，您也必須修改執行個體的安全群組，以允許連接埠 443 上的流量。下列組態檔案會使用 AWS CloudFormation 函數擷取安全群組 ID，並對其新增規則。

Example .ebextensions/https-instance-single.config

```
Resources:
  sslSecurityGroupIngress:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: {"Fn::GetAtt" : ["AWSEBSecurityGroup", "GroupId"]}
      IpProtocol: tcp
      ToPort: 443
      FromPort: 443
      CidrIp: 0.0.0.0/0
```

針對使用負載平衡的環境，您應設定負載平衡器，使其以未處理之方式通過安全流量，或針對端點對端點加密進行解密與重新加密。

在執行 Ruby 的 EC2 執行個體上終止 HTTPS

以 Ruby 容器類型而言，您啟用 HTTPS 的方式取決於使用的應用程式伺服器類型。

主題

- 針對搭配 Puma 的 Ruby 設定 HTTPS
- 針對搭配 Passenger 的 Ruby 設定 HTTPS

針對搭配 Puma 的 Ruby 設定 HTTPS

以使用 Puma 做為應用程式伺服器的 Ruby 容器類型而言，您可使用組態檔案來啟用 HTTPS。

在您的組態檔案中加入下列的程式碼片段、依照指示來更換憑證與私有金鑰資料，並儲存於原始碼套件的 `.ebextensions` 目錄中。此組態檔案會執行下列任務：

- `files` 金鑰會於執行個體上建立下列檔案：

  `/etc/nginx/conf.d/https.conf`

  設定 nginx 伺服器。nginx 服務開始後，即會載入此檔案。

  `/etc/pki/tls/certs/server.crt`

  在執行個體上建立憑證檔案。將######取代為您的憑證內容。

  > **Note**
  >
  > YAML 憑藉一致的縮排。請在取代範例組態檔中的內容時，讓縮排層級一致，並確認您的文字編輯器使用空格而非定位字元進行縮排。

  如果您有中繼憑證，請在網站憑證後將其納入 `server.crt` 中。

  ```
      -----BEGIN CERTIFICATE-----
  certificate file contents
  -----END CERTIFICATE-----
  -----BEGIN CERTIFICATE-----
  first intermediate certificate
  -----END CERTIFICATE-----
  -----BEGIN CERTIFICATE-----
  second intermediate certificate
  -----END CERTIFICATE-----
  ```

  `/etc/pki/tls/certs/server.key`

  在執行個體上建立私有金鑰。將######取代為用於建立憑證請求或自我簽署憑證的私密金鑰內容。

- `container_commands` 金鑰會在所有項目設定完成後重新啟動 nginx 伺服器，以讓伺服器使用新的 `https.conf` 檔案。

Example .ebextensions/https-instance.config

```
files:
  /etc/nginx/conf.d/https.conf:
    content: |
      # HTTPS server
```

```
    server {
        listen       443;
        server_name  localhost;

        ssl                  on;
        ssl_certificate      /etc/pki/tls/certs/server.crt;
        ssl_certificate_key  /etc/pki/tls/certs/server.key;

        ssl_session_timeout  5m;

        ssl_protocols  TLSv1 TLSv1.1 TLSv1.2;
        ssl_prefer_server_ciphers   on;

        location / {
            proxy_pass  http://my_app;
            proxy_set_header        Host            $host;
            proxy_set_header        X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header        X-Forwarded-Proto https;
        }

        location /assets {
          alias /var/app/current/public/assets;
          gzip_static on;
          gzip on;
          expires max;
          add_header Cache-Control public;
        }

        location /public {
          alias /var/app/current/public;
          gzip_static on;
          gzip on;
          expires max;
          add_header Cache-Control public;
        }
    }

/etc/pki/tls/certs/server.crt:
  content: |
    -----BEGIN CERTIFICATE-----
    certificate file contents
    -----END CERTIFICATE-----
```

```
  /etc/pki/tls/certs/server.key:
    content: |
      -----BEGIN RSA PRIVATE KEY-----
      private key contents # See note below.
      -----END RSA PRIVATE KEY-----

container_commands:
  01restart_nginx:
    command: "service nginx restart"
```

> **Note**
>
> 請避免將包含您私有金鑰的組態檔遞交到原始檔控制。當您測試好組態並確認其正常運作後，請將您的私有金鑰儲存在 Amazon S3，然後修改組態，以在部署期間予以下載。如需指示，請參閱 將私有金鑰安全地儲存於 Amazon S3 中。

在單一執行個體環境中，您也必須修改執行個體的安全群組，以允許連接埠 443 上的流量。下列組態檔案會使用 AWS CloudFormation 函數擷取安全群組 ID，並對其新增規則。

Example .ebextensions/https-instance-single.config

```
Resources:
  sslSecurityGroupIngress:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: {"Fn::GetAtt" : ["AWSEBSecurityGroup", "GroupId"]}
      IpProtocol: tcp
      ToPort: 443
      FromPort: 443
      CidrIp: 0.0.0.0/0
```

針對使用負載平衡的環境，您應設定負載平衡器，使其以未處理之方式通過安全流量，或針對端點對端點加密進行解密與重新加密。

針對搭配 Passenger 的 Ruby 設定 HTTPS

以使用 Passenger 做為應用程式伺服器的 Ruby 容器類型而言，您可同時使用組態檔案和 JSON 檔案來啟用 HTTPS。

欲針對搭配 Passenger 的 Ruby 設定 HTTPS

1. 在您的組態檔案中加入下列的程式碼片段、依照指示來更換憑證與私有金鑰資料，並儲存於原始碼套件的 .ebextensions 目錄中。此組態檔案會執行下列任務：

   - files 金鑰會於執行個體上建立下列檔案：

     /etc/pki/tls/certs/server.crt

     在執行個體上建立憑證檔案。將######取代為您的憑證內容。

     > **Note**
     >
     > YAML 憑藉一致的縮排。請在取代範例組態檔中的內容時，讓縮排層級一致，並確認您的文字編輯器使用空格而非定位字元進行縮排。

     如果您有中繼憑證，請在網站憑證後將其納入 server.crt 中。

     ```
     -----BEGIN CERTIFICATE-----
     certificate file contents
     -----END CERTIFICATE-----
     -----BEGIN CERTIFICATE-----
     first intermediate certificate
     -----END CERTIFICATE-----
     -----BEGIN CERTIFICATE-----
     second intermediate certificate
     -----END CERTIFICATE-----
     ```

     /etc/pki/tls/certs/server.key

     在執行個體上建立私有金鑰。將######取代為用於建立憑證請求或自我簽署憑證的私密金鑰內容。

   Example 針對搭配 Passenger 之 Ruby 設定 HTTPS 的 .Ebextensions 程式碼片段

   ```
   files:
     /etc/pki/tls/certs/server.crt:
       content: |
         -----BEGIN CERTIFICATE-----
         certificate file contents
   ```

```
      -----END CERTIFICATE-----

  /etc/pki/tls/certs/server.key:
    content: |
      -----BEGIN RSA PRIVATE KEY-----
      private key contents # See note below.
      -----END RSA PRIVATE KEY-----
```

> **Note**
>
> 請避免將包含您私有金鑰的組態檔遞交到原始檔控制。當您測試好組態並確認其正常運作後，請將您的私有金鑰儲存在 Amazon S3，然後修改組態，以在部署期間予以下載。如需指示，請參閱 將私有金鑰安全地儲存於 Amazon S3 中。

2. 建立一個文字檔案，並將下列 JSON 新增至檔案。將其儲存於您原始碼套件的根目錄，並取名為 `passenger-standalone.json`。此 JSON 檔案會將 Passenger 設定為使用 HTTPS。

> **Important**
>
> 此 JSON 檔案不得包含位元組順序記號 (BOM)。否則，Passenger JSON 程式庫將無法正確讀取檔案，且 Passenger 服務不會啟動。

Example passenger-standalone.json

```
{
  "ssl" : true,
  "ssl_port" : 443,
  "ssl_certificate" : "/etc/pki/tls/certs/server.crt",
  "ssl_certificate_key" : "/etc/pki/tls/certs/server.key"
}
```

在單一執行個體環境中，您也必須修改執行個體的安全群組，以允許連接埠 443 上的流量。下列組態檔案會使用 AWS CloudFormation 函數擷取安全群組 ID，並對其新增規則。

Example .ebextensions/https-instance-single.config

```
Resources:
  sslSecurityGroupIngress:
```

```
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: {"Fn::GetAtt" : ["AWSEBSecurityGroup", "GroupId"]}
      IpProtocol: tcp
      ToPort: 443
      FromPort: 443
      CidrIp: 0.0.0.0/0
```

針對使用負載平衡的環境，您應設定負載平衡器，使其以未處理之方式通過安全流量，或針對端點對端點加密進行解密與重新加密。

在執行 Tomcat 的 EC2 執行個體上終止 HTTPS

以 Tomcat 容器類型而言，您須使用組態檔案，讓 Apache HTTP Server 做為 Tomcat 的反向代理程式時能夠使用 HTTPS。

在您的組態檔案中加入下列的程式碼片段、依照指示來更換憑證與私有金鑰資料，並儲存於原始碼套件的 `.ebextensions` 目錄中。此組態檔案會執行下列任務：

- `files` 金鑰會於執行個體上建立下列檔案：

  `/etc/pki/tls/certs/server.crt`

  在執行個體上建立憑證檔案。將`######`取代為您的憑證內容。

  > **Note**
  >
  > YAML 憑藉一致的縮排。請在取代範例組態檔中的內容時，讓縮排層級一致，並確認您的文字編輯器使用空格而非定位字元進行縮排。

  `/etc/pki/tls/certs/server.key`

  在執行個體上建立私有金鑰。將`######`取代為用於建立憑證請求或自我簽署憑證的私密金鑰內容。

  `/opt/elasticbeanstalk/hooks/appdeploy/post/99_start_httpd.sh`

  建立後部署勾點指令碼來重新啟動 httpd 服務。

Example .ebextensions/https-instance.config

```
files:
  /etc/pki/tls/certs/server.crt:
    mode: "000400"
    owner: root
    group: root
    content: |
      -----BEGIN CERTIFICATE-----
      certificate file contents
      -----END CERTIFICATE-----

  /etc/pki/tls/certs/server.key:
    mode: "000400"
    owner: root
    group: root
    content: |
      -----BEGIN RSA PRIVATE KEY-----
      private key contents # See note below.
      -----END RSA PRIVATE KEY-----

  /opt/elasticbeanstalk/hooks/appdeploy/post/99_start_httpd.sh:
    mode: "000755"
    owner: root
    group: root
    content: |
      #!/usr/bin/env bash
      sudo service httpd restart
```

您還必須設定您環境的代理伺服器以監聽連接埠 443。下列 Apache 2.4 組態會在 443 埠上新增接聽程式。如需進一步了解，請參閱設定您的 Tomcat 環境的代理伺服器。

Example .ebextensions/httpd/conf.d/ssl.conf

```
Listen 443
<VirtualHost *:443>
  ServerName server-name
  SSLEngine on
  SSLCertificateFile "/etc/pki/tls/certs/server.crt"
  SSLCertificateKeyFile "/etc/pki/tls/certs/server.key"

  <Proxy *>
    Require all granted
```

```
  </Proxy>
  ProxyPass / http://localhost:8080/ retry=0
  ProxyPassReverse / http://localhost:8080/
  ProxyPreserveHost on

  ErrorLog /var/log/httpd/elasticbeanstalk-ssl-error_log

</VirtualHost>
```

您的憑證廠商可能包括您可安裝的中繼憑證，提升行動用戶端的相容性。請將下列新增至您的 SSL 組態檔案 (參閱 擴展和覆寫預設的 Apache 組態 — Amazon Linux AMI (AL1) 了解位置)，藉此透過中繼憑證授權機構 (CA) 套件組合設定 Apache：

- 在 `ssl.conf` 檔案內容指定鏈檔案：

```
SSLCertificateKeyFile "/etc/pki/tls/certs/server.key"
SSLCertificateChainFile "/etc/pki/tls/certs/gd_bundle.crt"
SSLCipherSuite        EECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH
```

- 使用中繼憑證的內容在 `files` 金鑰新增項目：

```
files:
  /etc/pki/tls/certs/gd_bundle.crt:
    mode: "000400"
    owner: root
    group: root
    content: |
      -----BEGIN CERTIFICATE-----
      First intermediate certificate
      -----END CERTIFICATE-----
      -----BEGIN CERTIFICATE-----
      Second intermediate certificate
      -----END CERTIFICATE-----
```

> **Note**
>
> 請避免將包含您私有金鑰的組態檔遞交到原始檔控制。當您測試好組態並確認其正常運作後，請將您的私有金鑰儲存在 Amazon S3，然後修改組態，以在部署期間予以下載。如需指示，請參閱 將私有金鑰安全地儲存於 Amazon S3 中。

在單一執行個體環境中，您也必須修改執行個體的安全群組，以允許連接埠 443 上的流量。下列組態檔案會使用 AWS CloudFormation 函數擷取安全群組 ID，並對其新增規則。

Example .ebextensions/https-instance-single.config

```
Resources:
  sslSecurityGroupIngress:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: {"Fn::GetAtt" : ["AWSEBSecurityGroup", "GroupId"]}
      IpProtocol: tcp
      ToPort: 443
      FromPort: 443
      CidrIp: 0.0.0.0/0
```

針對使用負載平衡的環境，您應設定負載平衡器，使其以未處理之方式通過安全流量，或針對端點對端點加密進行解密與重新加密。

在執行 .NET Core on Linux 的 Amazon EC2 執行個體上終止 HTTPS

如果是 .NET Core on Linux 容器類型，您可使用 `.ebextensions` 組態檔案來啟用 HTTPS，以及使用 nginx 組態檔案來設定 nginx 伺服器使用 HTTPS。

在您的組態檔案中加入下列的程式碼片段、依照指示來更換憑證與私有金鑰佔位符，並儲存於 `.ebextensions` 目錄中。此組態檔案會執行下列任務：

- `files` 金鑰會於執行個體上建立下列檔案：

  `/etc/pki/tls/certs/server.crt`

  在執行個體上建立憑證檔案。將######取代為您的憑證內容。

  > **Note**
  >
  > YAML 憑藉一致的縮排。請在取代範例組態檔中的內容時，讓縮排層級一致，並確認您的文字編輯器使用空格而非定位字元進行縮排。

  如果您有中繼憑證，請在網站憑證後將其納入 `server.crt` 中。

  ```
      -----BEGIN CERTIFICATE-----
  ```

```
    certificate file contents
    -----END CERTIFICATE-----
    -----BEGIN CERTIFICATE-----
    first intermediate certificate
    -----END CERTIFICATE-----
    -----BEGIN CERTIFICATE-----
    second intermediate certificate
    -----END CERTIFICATE-----
```

`/etc/pki/tls/certs/server.key`

在執行個體上建立私有金鑰。將######取代為用於建立憑證請求或自我簽署憑證的私密金鑰內容。

- `container_commands` 金鑰會在所有項目設定完成後重新啟動 nginx 伺服器，以讓伺服器載入 nginx 組態檔案。

Example .ebextensions/https-instance.config

```
files:
  /etc/pki/tls/certs/server.crt:
    content: |
      -----BEGIN CERTIFICATE-----
      certificate file contents
      -----END CERTIFICATE-----

  /etc/pki/tls/certs/server.key:
    content: |
      -----BEGIN RSA PRIVATE KEY-----
      private key contents # See note below.
      -----END RSA PRIVATE KEY-----

container_commands:
  01restart_nginx:
    command: "systemctl restart nginx"
```

> **Note**
>
> 請避免將包含您私有金鑰的組態檔遞交到原始檔控制。當您測試好組態並確認其正常運作後，請將您的私有金鑰儲存在 Amazon S3，然後修改組態，以在部署期間予以下載。如需指示，請參閱 將私有金鑰安全地儲存於 Amazon S3 中。

將下列內容加入您原始碼套件 `.conf` 目錄中的檔案，該檔案的副檔名為 `.platform/nginx/conf.d/` (例如 `.platform/nginx/conf.d/https.conf`)。將 *app_port* 換成您應用程式接聽的埠號。本範例設定 nginx 伺服器使用 SSL 來接聽 443 埠。如需 .NET Core on Linux 平台上這些組態檔案的詳細資訊，請參閱the section called “代理伺服器”。

Example .platform/nginx/conf.d/https.conf

```
# HTTPS server

server {
    listen       443 ssl;
    server_name  localhost;

    ssl_certificate      /etc/pki/tls/certs/server.crt;
    ssl_certificate_key  /etc/pki/tls/certs/server.key;

    ssl_session_timeout  5m;

    ssl_protocols  TLSv1 TLSv1.1 TLSv1.2;
    ssl_prefer_server_ciphers   on;

    location / {
        proxy_pass  http://localhost:app_port;
        proxy_set_header   Connection "";
        proxy_http_version 1.1;
        proxy_set_header        Host            $host;
        proxy_set_header        X-Real-IP       $remote_addr;
        proxy_set_header        X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header        X-Forwarded-Proto https;
    }
}
```

在單一執行個體環境中，您也必須修改執行個體的安全群組，以允許連接埠 443 上的流量。下列組態檔案會使用 AWS CloudFormation 函數擷取安全群組 ID，並對其新增規則。

Example .ebextensions/https-instance-single.config

```
Resources:
  sslSecurityGroupIngress:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: {"Fn::GetAtt" : ["AWSEBSecurityGroup", "GroupId"]}
      IpProtocol: tcp
```

```
      ToPort: 443
      FromPort: 443
      CidrIp: 0.0.0.0/0
```

針對使用負載平衡的環境，您應設定負載平衡器，使其以未處理之方式通過安全流量，或針對端點對端點加密進行解密與重新加密。

在執行 .NET 的 Amazon EC2 執行個體上終止 HTTPS

下列組態檔案會建立並執行執行下列工作的 Windows PowerShell 指令碼：

- 檢查連接埠 443 的現有 HTTPS 憑證繫結。
- 從 Amazon S3 存儲桶獲取 PFX 證書。

> **Note**
>
> 將`AmazonS3ReadOnlyAccess`政策新增`aws-elasticbeanstalk-ec2-role`至以存取 Amazon S3 儲存貯體中的 SSL 憑證。

- 從中獲取密碼 AWS Secrets Manager。

> **Note**
>
> 在中新增陳述式`aws-elasticbeanstalk-ec2-role`，允許`secretsmanager:GetSecretValue`針對包含憑證密碼的密碼執行動作

- 安裝憑證。
- 將憑證繫結至連接埠 443。

> **Note**
>
> 若要移除 HTTP 端點 (連接埠 80)，請將 `Remove-WebBinding` 命令納入此範例 Remove the HTTP binding (移除 HTTP 綁定) 部分之下。

Example .ebextension/. https-instance-dotnet 配置

```
files:
  "C:\\certs\\install-cert.ps1":
    content: |
```

```
      import-module webadministration
      ## Settings - replace the following values with your own
      $bucket = "DOC-EXAMPLE-BUCKET"  ## S3 bucket name
      $certkey = "example.com.pfx"    ## S3 object key for your PFX certificate
      $secretname = "example_secret"  ## AWS Secrets Manager name for a secret that
contains the certificate's password
      ##

      # Set variables
      $certfile = "C:\cert.pfx"
      $pwd = Get-SECSecretValue -SecretId $secretname | select -expand SecretString

      # Clean up existing binding
      if ( Get-WebBinding "Default Web Site" -Port 443 ) {
        Echo "Removing WebBinding"
        Remove-WebBinding -Name "Default Web Site" -BindingInformation *:443:
      }
      if ( Get-Item -path IIS:\SslBindings\0.0.0.0!443 ) {
        Echo "Deregistering WebBinding from IIS"
        Remove-Item -path IIS:\SslBindings\0.0.0.0!443
      }

      # Download certificate from S3
      Read-S3Object -BucketName $bucket -Key $certkey -File $certfile

      # Install certificate
      Echo "Installing cert..."
      $securepwd = ConvertTo-SecureString -String $pwd -Force -AsPlainText
      $cert = Import-PfxCertificate -FilePath $certfile cert:\localMachine\my -Password
$securepwd

      # Create site binding
      Echo "Creating and registering WebBinding"
      New-WebBinding -Name "Default Web Site" -IP "*" -Port 443 -Protocol https
      New-Item -path IIS:\SslBindings\0.0.0.0!443 -value $cert -Force

      ## Remove the HTTP binding
      ## (optional) Uncomment the following line to unbind port 80
      # Remove-WebBinding -Name "Default Web Site" -BindingInformation *:80:
      ##

      # Update firewall
      netsh advfirewall firewall add rule name="Open port 443" protocol=TCP
localport=443 action=allow dir=OUT
```

```
commands:
  00_install_ssl:
    command: powershell -NoProfile -ExecutionPolicy Bypass -file C:\\certs\\install-
cert.ps1
```

在單一執行個體環境中，您也必須修改執行個體的安全群組，以允許連接埠 443 上的流量。下列組態檔案會使用 AWS CloudFormation 函數擷取安全性群組的 ID，並將規則新增至其中。

Example .ebextension/. https-instance-single 配置

```
Resources:
  sslSecurityGroupIngress:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: {"Fn::GetAtt" : ["AWSEBSecurityGroup", "GroupId"]}
      IpProtocol: tcp
      ToPort: 443
      FromPort: 443
      CidrIp: 0.0.0.0/0
```

對於負載平衡環境，您可以將負載平衡器設定為透過未觸及傳遞安全流量，或解密並重新加密以進行加密。 end-to-end

在負載平衡的 Elastic Beanstalk 環境中設定端對端加密

終止負載平衡器的安全連線並於後端使用 HTTP，對您的應用程式而言可能已足夠。即使在相同帳戶中執行，AWS 資源間的網路流量仍無法由不屬於該連線的執行個體接聽。

不過，若正開發的應用程式需要遵循嚴格的外部法規，您可能需要保護所有網路連線。您可使用 Elastic Beanstalk 主控台或組態檔案，將您 Elastic Beanstalk 環境的負載平衡器安全地連接至後端執行個體，以滿足這些要求。下列程序著重於組態檔案。

首先，將安全接聽程式新增至您的負載平衡器 (若尚未執行)。

您亦必須設定環境中的執行個體，以接聽安全的連接埠並終止 HTTPS 連線。每個組態視平台而有所不同。如需說明，請參閱 於執行個體設定您的應用程式以終止 HTTPS 連線。您的 EC2 執行個體可使用自我簽署的簽憑，而不會出現問題。

接著，將接聽程式設定為在您應用程式所用之安全連接埠上使用 HTTPS 來轉送流量。根據您環境中使用的負載平衡器類型，使用以下其中一個組態檔案。

.ebextensions/https-reencrypt-clb.config

搭配 Classic Load Balancer 使用此組態檔案。除了設定負載平衡器，組態檔案也會變更預設的運作狀態檢查來使用連接埠 443 和 HTTPS，以確保負載平衡器能夠安全地連線。

```
option_settings:
  aws:elb:listener:443:
    InstancePort: 443
    InstanceProtocol: HTTPS
  aws:elasticbeanstalk:application:
    Application Healthcheck URL: HTTPS:443/
```

.ebextensions/https-reencrypt-alb.config

搭配 Application Load Balancer 使用此組態檔案。

```
option_settings:
  aws:elbv2:listener:443:
    DefaultProcess: https
    ListenerEnabled: 'true'
    Protocol: HTTPS
  aws:elasticbeanstalk:environment:process:https:
    Port: '443'
    Protocol: HTTPS
```

.ebextensions/https-reencrypt-nlb.config

搭配 Network Load Balancer 使用此組態檔案。

```
option_settings:
  aws:elbv2:listener:443:
    DefaultProcess: https
    ListenerEnabled: 'true'
  aws:elasticbeanstalk:environment:process:https:
    Port: '443'
```

`DefaultProcess` 選項會以此方式命名，是因為 Application Load Balancer 可能在相同連接埠上，針對傳送至特定路徑的流量使用非預設的接聽程式 (詳細資訊請參閱Application Load Balancer)。以 Network Load Balancer 而言，此選項會指定此接聽程式的唯一目標程序。

在此範例中，我們將程序命名為 `https`，因為它會接聽安全 (HTTPS) 流量。由於 Network Load Balancer 僅能與 TCP 搭配運作，因此接聽程式會使用 TCP 通訊協定，將流量傳送至指定連接埠上的程序。此作法沒問題，因為 HTTP 和 HTTPS 的網路流量實作於 TCP 之上。

> **Note**
>
> EB CLI 和 Elastic Beanstalk 主控台會為前述選項套用建議的數值。若您想要使用組態檔進行相同的設定，您必須移除這些設定。如需詳細資訊，請參閱「建議值」。

在接下來的任務，您需要修改負載平衡器的安全群組，以允許流量。根據您啟動您環境的 Amazon Virtual Private Cloud (Amazon VPC)—預設 VPC 或自訂 VPC—負載平衡器的安全群組將會不同。在預設 VPC 中，Elastic Load Balancing 會提供預設安全群組，所有負載平衡器均可加以使用。在您建立的 Amazon VPC 中，Elastic Beanstalk 會針對要使用的負載平衡器建立安全群組。

為了同時滿足這兩種情境，您可建立安全群組，並指示 Elastic Beanstalk 使用該安全群組。下列組態檔案會建立安全群組，並將其連接至負載平衡器。

`.ebextensions/https-lbsecuritygroup.config`

```
option_settings:
  # Use the custom security group for the load balancer
  aws:elb:loadbalancer:
    SecurityGroups: '`{ "Ref" : "loadbalancersg" }`'
    ManagedSecurityGroup: '`{ "Ref" : "loadbalancersg" }`'

Resources:
  loadbalancersg:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: load balancer security group
      VpcId: vpc-########
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 443
          ToPort: 443
          CidrIp: 0.0.0.0/0
        - IpProtocol: tcp
          FromPort: 80
          ToPort: 80
          CidrIp: 0.0.0.0/0
      SecurityGroupEgress:
```

```
        - IpProtocol: tcp
          FromPort: 80
          ToPort: 80
          CidrIp: 0.0.0.0/0
```

將醒目提示的文字取代為您的預設或自訂 VPC ID。先前範例包含連接埠 80 上的進出流量，可允許 HTTP 連線。若您僅允許安全連線，可移除這些屬性。

最後，新增可允許負載平衡器安全群組和執行個體安全群組透過連接埠 443 進行通訊的輸入和輸出規則。

.ebextensions/https-backendsecurity.config

```
Resources:
  # Add 443-inbound to instance security group (AWSEBSecurityGroup)
  httpsFromLoadBalancerSG:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: {"Fn::GetAtt" : ["AWSEBSecurityGroup", "GroupId"]}
      IpProtocol: tcp
      ToPort: 443
      FromPort: 443
      SourceSecurityGroupId: {"Fn::GetAtt" : ["loadbalancersg", "GroupId"]}
  # Add 443-outbound to load balancer security group (loadbalancersg)
  httpsToBackendInstances:
    Type: AWS::EC2::SecurityGroupEgress
    Properties:
      GroupId: {"Fn::GetAtt" : ["loadbalancersg", "GroupId"]}
      IpProtocol: tcp
      ToPort: 443
      FromPort: 443
      DestinationSecurityGroupId: {"Fn::GetAtt" : ["AWSEBSecurityGroup", "GroupId"]}
```

此作法不在建立安全群組時進行，可讓您限制來源和目的安全群組，無須建立循環相依性。

在您完成所有前述步驟後，負載平衡器即可使用 HTTPS 安全連接至您的後端執行個體。不論您執行個體的憑證為自我簽署或由信任憑證授權單位核發，負載平衡器都會接受任何向其遞交的憑證。

您可新增政策至負載平衡器，指示其僅信任特定憑證，藉此變更此行為。下列組態檔案會建立兩個政策。一個政策指定公有憑證，另一個則指示負載平衡器僅信任該憑證來連接至執行個體連接埠 443。

.ebextensions/https-backendauth.config

```
option_settings:
  # Backend Encryption Policy
  aws:elb:policies:backendencryption:
    PublicKeyPolicyNames: backendkey
    InstancePorts:  443
  # Public Key Policy
  aws:elb:policies:backendkey:
    PublicKey: |
      -----BEGIN CERTIFICATE-----
      ################################################################
      ################################################################
      ################################################################
      ################################################################
      ################################################
      -----END CERTIFICATE-----
```

將醒目提示的文字取代為您的 EC2 執行個體公有憑證的內容。

針對 TCP 傳遞設定您環境的負載平衡器

若您不想在 AWS Elastic Beanstalk 環境的負載平衡器解密 HTTPS 流量，您可以設定安全接聽程式，將請求依原狀轉送後端執行個體。

首先，將您環境的 EC2 執行個體設定為終止 HTTPS。在單一執行個體環境內測試組態，以確保一切正常運作，之後再將負載平衡器新增至組合。

將組態檔案新增至您的專案，藉此在連接埠 443 設定接聽程式，使其能夠依原狀將 TCP 封包傳送至後端執行個體的連接埠 443：

`.ebextensions/https-lb-passthrough.config`

```
option_settings:
  aws:elb:listener:443:
    ListenerProtocol: TCP
    InstancePort: 443
    InstanceProtocol: TCP
```

在預設 Amazon Virtual Private Cloud (Amazon VPC) 中，您亦需要新增規則至執行個體的安全群組，以允許 443 上來自負載平衡器的傳入流量：

`.ebextensions/https-instance-securitygroup.config`

```
Resources:
  443inboundfromloadbalancer:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: {"Fn::GetAtt" : ["AWSEBSecurityGroup", "GroupId"]}
      IpProtocol: tcp
      ToPort: 443
      FromPort: 443
      SourceSecurityGroupName: { "Fn::GetAtt": ["AWSEBLoadBalancer",
 "SourceSecurityGroup.GroupName"] }
```

在自訂 VPC 中，Elastic Beanstalk 會為您更新安全群組組態。

將私有金鑰安全地儲存於 Amazon S3 中

您用來簽署公有憑證的私密金鑰是私有的，不應直接寫在原始程式碼中。您可以將私有金鑰上傳到 Amazon S3，然後設定 Elastic Beanstalk 在部署應用程式時從 Amazon S3 下載檔案，來避免將私有金鑰儲存於組態檔案中。

下列範例顯示組態檔案的資源和檔案區段，此組態檔案從 Amazon S3 儲存貯體下載了私有金鑰檔案。

Example .ebextensions/privatekey.config

```
Resources:
  AWSEBAutoScalingGroup:
    Metadata:
      AWS::CloudFormation::Authentication:
        S3Auth:
          type: "s3"
          buckets: ["elasticbeanstalk-us-west-2-123456789012"]
          roleName:
            "Fn::GetOptionSetting":
              Namespace: "aws:autoscaling:launchconfiguration"
              OptionName: "IamInstanceProfile"
              DefaultValue: "aws-elasticbeanstalk-ec2-role"
files:
  # Private key
  "/etc/pki/tls/certs/server.key":
    mode: "000400"
    owner: root
    group: root
    authentication: "S3Auth"
```

```
    source: https://elasticbeanstalk-us-west-2-123456789012.s3.us-west-2.amazonaws.com/
server.key
```

用您自己的資料取代範例中的儲存貯體名稱和 URL。這個檔案中的第一個項目，將名為 S3Auth 的身份驗證方法，加進了環境的 Auto Scaling 群組的中繼資料。如果您已經為環境設定了自訂的執行個體描述檔，將會使用該設定檔，否則就會套用 aws-elasticbeanstalk-ec2-role 的預設值。預設的執行個體描述檔具有權限，可從 Elastic Beanstalk 儲存貯體讀取。如果您使用不同的儲存貯體，請新增權限到執行個體描述檔。

第二個項目會使用 S3Auth 身份驗證方法，來從指定的 URL 下載私有金鑰，然後將金鑰儲存至 /etc/pki/tls/certs/server.key。之後，代理伺服器即可從此位置讀取私密金鑰，以終止執行個體的 HTTPS 連線。

指派給您環境的 EC2 執行個體的執行個體描述檔，必須擁有權限，以從指定的儲存貯體讀取金鑰物件。請確定執行個體描述檔具有許可，能夠讀取 IAM 中的物件，而且儲存貯體和物件上的許可不會禁止執行個體描述檔。

若要查看儲存貯體的權限

1. 開啟 Amazon S3 管理主控台。
2. 選擇儲存貯體。
3. 選擇 Properties (屬性)，然後選擇 Permissions (權限)。
4. 確認您的帳戶是儲存貯體上的承授者，具備讀取權限。
5. 如果已連接儲存貯體政策，請選擇 Bucket policy (儲存貯體政策)，以檢視已指派給儲存貯體的許可。

將 HTTP 設定為 HTTPS 重新導向

在為您的 Elastic Beanstalk 環境設定 HTTPS及其主題中，我們將討論將 Elastic Beanstalk 環境設定為使用 HTTPS，來確保對您的應用程式進行流量加密。此主題說明在最終使用者仍啟動對您應用程式的 HTTP 流量時，如何精心處理。您需要設定 HTTP 到 HTTPS 重新導向 (有時稱為強制 HTTPS) 來這麼做。

要設定重新導向，您先設定您的環境，來處理 HTTPS 流量。然後您便可以將 HTTP 流量重新導向到 HTTPS。這兩個步驟會在下列小節討論。

設定您的環境以處理 HTTPS 流量

取決於您環境的負載平衡組態，請執行以下其中一項作業：

- 負載平衡環境 – 設定您的負載平衡器以終止 HTTPS。
- 單一執行個體環境 – 設定您的應用程式以終止執行個體上的 HTTPS 連線。此組態取決於您環境的平台。

將 HTTP 流量重新導向至 HTTPS

您可以設定您環境執行個體上的 web 伺服器或環境的 Application Load Balancer 來將 HTTP 流量重新導向至 HTTPS。執行下列任一步驟：

- 設定執行個體 web 伺服器 - 此方法適用於任何 web 伺服器環境。設定您 Amazon Elastic Compute Cloud (Amazon EC2) 上的 Web 伺服器，以使用 HTTP 重新導向回應狀態回應 HTTP 流量。此組態取決於您環境的平台。尋找在 GitHub 上在 `https-redirect` 集合適用於您平台的資料夾，並使用該資料夾中的範例組態檔案。

  如果您的環境使用 Elastic Load Balancing 運作狀態檢查，負載平衡器預期運作狀態良好的執行個體會使用 HTTP 200 (OK) 回應來回應該 HTTP 運作狀態檢查訊息。因此，Web 伺服器不應將這些訊息重新導向到 HTTPS。在 `https-redirect` 中的範例設定檔案會正確處理這項需求。
- 設定負載平衡器 — 如果您的負載平衡環境使用 Application Load Balancer，則此方法可運作。Application Load Balancer 可以在 HTTP 流量進入時傳送重新導向回應。在這種情況下，您不需要在環境的執行個體上設定重新導向。我們在 GitHub 上有兩個範例組態檔案，示範如何設定 Application Load Balancer 以進行重新導向。`alb-http-to-https-redirection-full.config` 組態檔案會在連接埠 443 上建立 HTTPS 接聽程式，並修改預設的連接埠 80 接聽程式，使其將傳入的 HTTP 流量重新導向至 HTTPS。`alb-http-to-https-redirection.config` 組態檔案預期定義 443 接聽程式 (您可以使用標準 Elastic Beanstalk 組態命名空間，或是 Elastic Beanstalk 主控台)。然後其會負責修改連接埠 80 接聽程式以進行重新導向。

監控環境

當您執行的是生產網站，了解您應用程式的可用性和回應請求的情況至關重要。為了協助監控您應用程式的回應能力，Elastic Beanstalk 提供的功能可監控應用程式的統計資訊，並建立超過閾值的觸發提醒。

主題

在 AWS 管理主控台中監控環境的運作狀態

您可以從 Elastic Beanstalk 主控台存取有關您應用程式的運作資訊。該主控台會顯示您環境的狀態和應用程式健全狀況概觀。在主控台的 Environments (環境) 頁面和每個應用程式的頁面中，清單上的環境會以顏色編碼，以指示狀態。

在 Elastic Beanstalk 主控台中監控環境

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

   > **Note**
   >
   > 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇 Monitoring (監控)。

監控頁面會顯示關於您環境的整體統計資料，例如 CPU 使用率和平均延遲。除了整體統計資料之外，您還可以查看監控圖表，這些圖表會隨時間顯示資源的使用量。您可以按一下任何圖表來檢視更多詳細資訊。

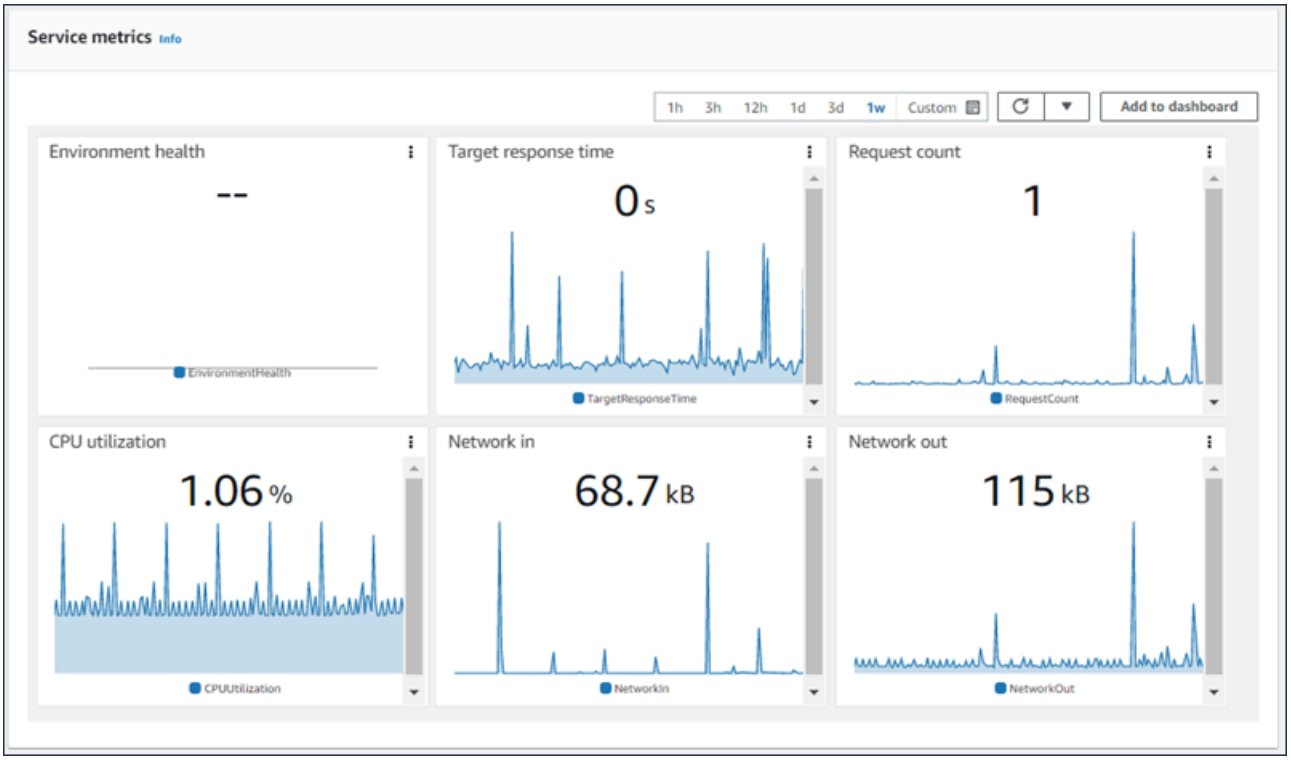
> **Note**
>
> 預設只會啟用基本的 CloudWatch 指標，傳回 5 分鐘期間的資料。透過編輯環境的組態設定，您可以啟用更精細的 1 分鐘 CloudWatch 指標。

監控圖表

監控頁面會顯示您環境的運作狀態相關指標概觀。這包括由 Elastic Load Balancing 與 Amazon EC2 提供的預設指標，以及顯示環境運作狀態隨時間變更的圖表。

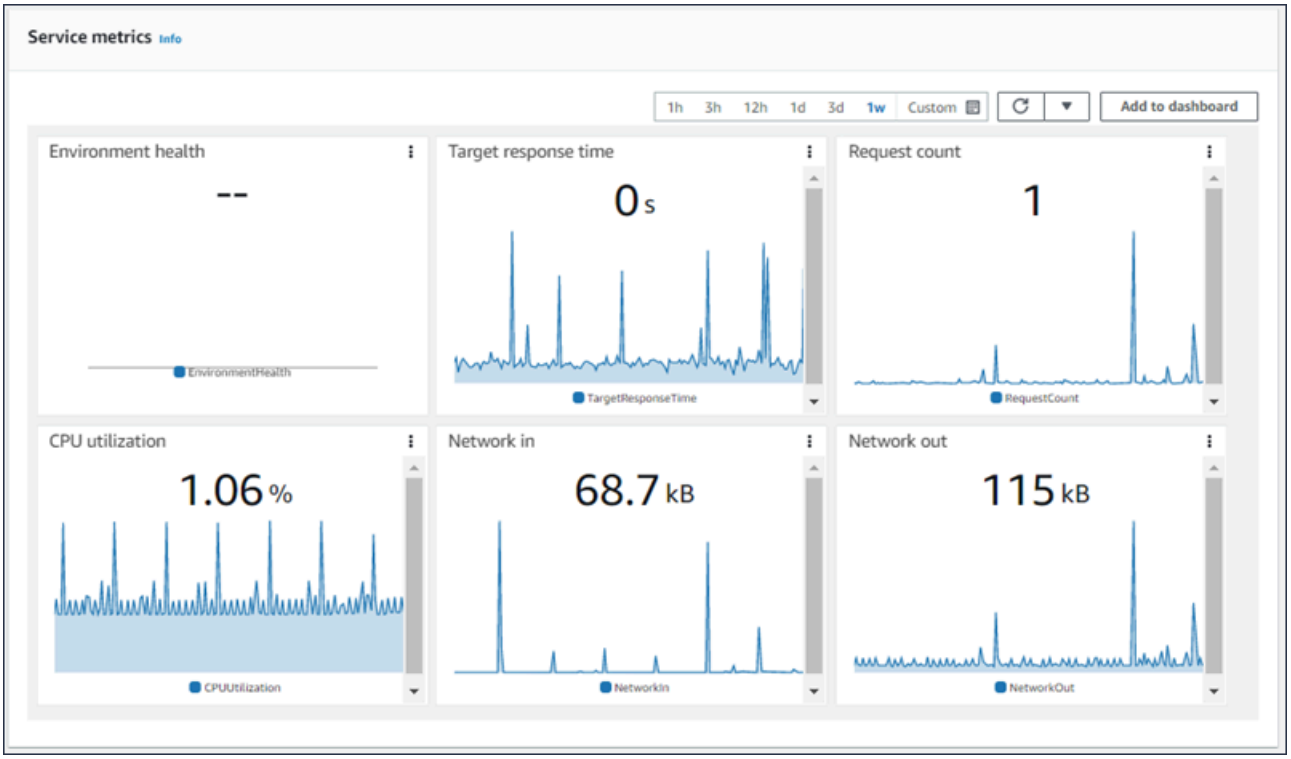
圖表上方的長條提供不同的時間間隔，以供您選取。例如，選取 1w 可顯示過去一週的資訊。或者，選取 3h 可顯示過去三小時的資訊。

如需更多種時間間隔選項，請選擇自訂。此處共有兩個範圍選項：絕對或相對。絕對選項可讓您指定特定的日期範圍，例如 2023 年 1 月 1 日至 2023 年 6 月 30 日。相對選項可選取具有特定時間單位的整數：分鐘、小時、日、週或月。範例包括 10 小時、10 日和 10 個月。



自訂監控主控台

若要建立和檢視自訂指標，就必須使用 Amazon CloudWatch。透過 CloudWatch，您可建立自訂儀表板，於單一檢視中監控資源。選取新增至儀表板，即可從監控頁面瀏覽至 Amazon CloudWatch 主控台。Amazon CloudWatch 會提供您建立新儀表板或選取現有儀表板的選項。如需詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的使用 Amazon CloudWatch 儀表板。



Elastic Load Balancing 和 Amazon EC2 指標可以在所有環境中啟用。

有了增強型運作狀態功能，EnvironmentHealth 指標會予以啟用，並自動將圖表加入監控主控台。增強型運作狀態功能也會將運作狀態頁面新增到管理主控台。如需可用的增強型運作狀態指標清單，請參閱 為環境發佈 Amazon CloudWatch 自訂指標。

基礎型運作狀態報告

AWS Elastic Beanstalk 使用來自多個來源的資訊來判斷您的環境是否可用，並處理來自網際網路的要求。環境的運作狀態是以四種顏色的其中一種表示，並顯示在 Elastic Beanstalk 主控台的環境概觀頁面上。它也可以從 DescribeEnvironmentsAPI 和使用 EB CLI 呼叫eb status來取得。

在第 2 版 Linux 平台版本之前，唯一的運作狀態報告系統為基本運作狀態。基礎型運作狀態報告系統會根據 Elastic Load Balancing (針對負載平衡環境) 或 Amazon Elastic Compute Cloud (針對單一執行個體環境) 執行的運作狀態檢查，提供 Elastic Beanstalk 環境中執行個體的運作狀態資訊。

除了檢查您的 EC2 執行個體的運作狀態，Elastic Beanstalk 亦會監控環境中的其他資源，並回報缺少或設定不正確的資源，避免造成使用者無法使用您的環境。

您環境中資源收集的指標會 CloudWatch 在五分鐘內發佈到 Amazon。其中包含 EC2 的作業系統指標或 Elastic Load Balancing 的請求指標。您可以在環境主控台的「監督」頁面檢視以這些測 CloudWatch 量結果為基礎的圖形。以基礎型運作狀態而言，這些指標不會用來判定環境的運作狀態。

主題

運作狀態顏色

Elastic Beanstalk 會根據於 Web 伺服器環境中執行的應用程式回應運作狀態檢查的情形，回報該環境的運作狀態。Elastic Beanstalk 使用四種顏色來描述狀態，如下表所示：

顏色	描述
灰色	您的環境正在更新。
Green	您的環境已通過最近的運作狀態檢查。您的環境至少有一個執行個體可用，且其正在接收請求。
Yellow	您的環境未通過一項或多項運作狀態檢查。向您環境發送的部分請求會失敗。
紅色	您的環境未通過三項或更多運作狀態檢查，或是一個資源變成不可使用。請求會持續失敗。

這些描述僅適用使用基礎型運作狀態報告的環境。如需增強型運作狀態的詳細資訊，請參閱運作狀態顏色和狀態。

Elastic Load Balancing 運作狀態檢查

在負載平衡環境中，Elastic Load Balancing 每 10 秒會傳送請求至環境中的各個執行個體，確認其運作狀態良好。負載平衡器預設設定為開啟連接埠 80 上的 TCP 連線。若執行個體確認連線，將視為運作狀態良好。

您可在應用程式指定現有資源，藉此覆寫這項設定。若您指定路徑 (如 `/health`)，則運作狀態檢查 URL 會設定為 `HTTP:80/health`。運作狀態檢查 URL 應一律設在您應用程式提供服務的路徑。若此 URL 設定在應用程式前方之 Web 伺服器處理或快取的靜態頁面，運作狀態檢查將不會顯示應用程式伺服器或 Web 容器的問題。如需修改運作狀態檢查 URL 的說明，請參閱 運作狀態檢查。

若已設定運作狀態檢查 URL，Elastic Load Balancing 預期其傳送的 GET 請求回傳 `200 OK` 回應。若應用程式在 5 秒內未回應，或回應其他 HTTP 狀態碼，則無法通過運作狀態檢查。連續 5 次運作狀態檢查失敗，Elastic Load Balancing 會停止該執行個體的服務。

如需有關 Elastic Load Balancing 運作狀態檢查的詳細資訊，請參閱《Elastic Load Balancing 使用者指南》中的運作狀態檢查。

> **Note**
>
> 設定運作狀態檢查 URL 不會變更環境的 Auto Scaling 群組的運作狀態檢查行為。運作狀態不良的執行個體會從負載平衡器移除，但 Amazon EC2 Auto Scaling 不會自動替換，除非您將 Amazon EC2 Auto Scaling 設定為使用 Elastic Load Balancing 運作狀態檢查，做為替換執行個體的根據。若要設定 Amazon EC2 Auto Scaling 以取代 Elastic Load Balancing 運作狀態檢查失敗的執行個體，請參閱 Auto Scaling 運作狀態檢查設定。

單一執行個體和工作者層環境運作狀態檢查

在單一執行個體或工作者層環境中，Elastic Beanstalk 會監控其 Amazon EC2 執行個體的狀態，藉此判定執行個體的運作狀態。Elastic Load Balancing 運作狀態設定 (包括 HTTP 運作狀態檢查 URL) 無法用於這些環境類型。

如需 Amazon EC2 執行個體狀態檢查的詳細資訊，請參閱 Amazon EC2 使用者指南中的使用狀態檢查監控執行個體。

其他檢查

除了 Elastic Load Balancing 運作狀態檢查，Elastic Beanstalk 會監控您環境中的資源，若其部署失敗、設定不正確或變為不可用，其運作狀態將變更為紅色。這些檢查會確認：

- 環境的 Auto Scaling 群組為可用，且具備至少一個執行個體。
- 環境的安全群組為可用，且設定為允許連接埠 80 上的傳入流量。
- 存在環境 CNAME，且其指向正確的負載平衡器。
- 在工作者環境中，Amazon Simple Queue Service (Amazon SQS) 佇列至少每三分鐘輪詢一次。

Amazon CloudWatch 指標

使用基本健康狀態報告，Elastic Beanstalk 服務不會向 Amazon 發布任何指標。 CloudWatch環境主控台的 [監視] 頁面上用來產生圖形的 CloudWatch 度量會由您環境中的資源發佈。

例如，EC2 會針對您環境 Auto Scaling 群組中的執行個體，發佈下列指標：

`CPUUtilization`

目前使用中的運算單位百分比。

`DiskReadBytes`, `DiskReadOps`, `DiskWriteBytes`, `DiskWriteOps`

讀取及寫入的位元組數，以及讀取及寫入操作的數量。

`NetworkIn`, `NetworkOut`

傳送及接收的位元組數。

Elastic Load Balancing 會針對您環境的負載平衡器發佈下列指標：

`BackendConnectionErrors`

負載平衡器和環境執行個體間連線失敗的數量。

`HTTPCode_Backend_2XX`, `HTTPCode_Backend_4XX`

您環境執行個體產生成功 (2XX) 及用戶端錯誤 (4XX) 回應代碼的數量。

`Latency`

負載平衡器將請求轉送至執行個體到接收回應的秒數。

`RequestCount`

完成的請求數。

此並非完整清單。如需可針對這些資源報告的指標完整清單，請參閱 Amazon CloudWatch 開發人員指南中的以下主題：

指標

命名空間	主題
AWS::ElasticLoadBalancing::LoadBalancer	Elastic Load Balancing 指標與資源
AWS::AutoScaling::AutoScaling集團	Amazon Elastic Compute Cloud 指標與資源
AWS::SQS::Queue	Amazon SQS 指標與資源
AWS::RDS::DBInstance	Amazon RDS 維度與指標

工作者環境運作狀態指標

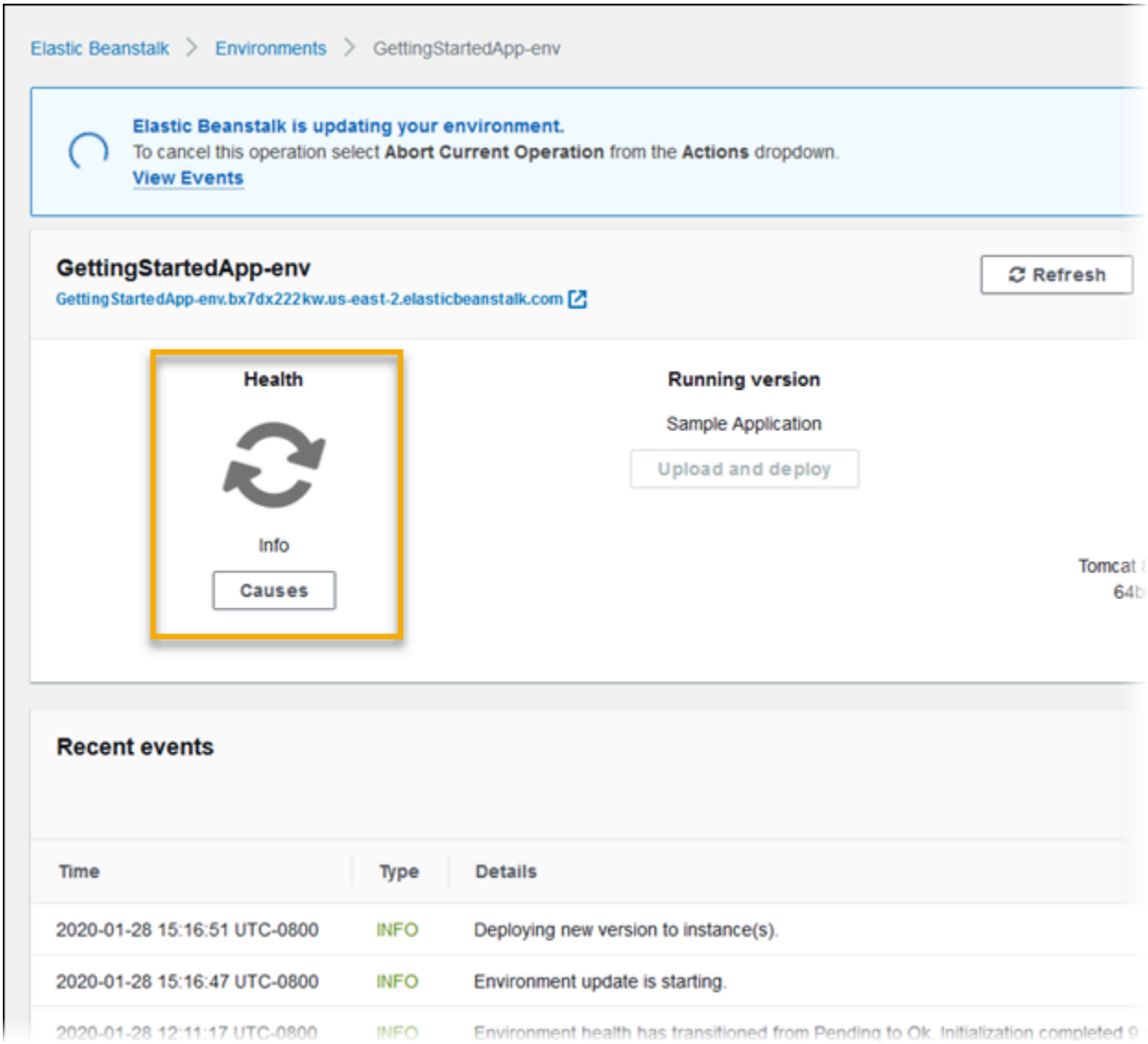
僅針對 Worker 環境，SQS 精靈會將環境健全狀況的自訂指標發佈到 CloudWatch，其中值 1 為綠色。您可以使用`ElasticBeanstalk/SQSD`命名空間檢閱帳戶中的 CloudWatch 健全狀況指標資料。指標維度為 `EnvironmentName`，而指標名稱為 `Health`。所有執行個體都將其指標發佈至相同的命名空間。

欲啟用協助程式來發佈指標，環境的執行個體描述檔必須具備呼叫 `cloudwatch:PutMetricData` 的許可。此許可會納入預設的執行個體描述檔。如需更多詳細資訊，請參閱 管理 Elastic Beanstalk 執行個體描述檔。

增強型運作狀態報告與監控

您可於環境啟用增強型運作狀態報告，讓 AWS Elastic Beanstalk 收集您環境資源的其他資訊。Elastic Beanstalk 會分析所收集到的資訊，以提供更全面的整體環境運作狀態，並協助辨識可能會導致應用程式無法運作的問題。

除了可變更運作狀態顏色的運作方式，增強型運作狀態更增加狀態描述項，當環境為黃色或紅色時，可提供所觀察之問題嚴重性的指標。若系統提供目前狀態的更多資訊，您可選擇 Causes (成因) 按鈕，於運作狀態頁面檢視詳細的運作狀態資訊。



為了提供在您環境中所執行 Amazon EC2 執行個體的詳細運作狀態資訊，Elastic Beanstalk 在 Amazon Machine Image (AMI) 中包含了每個支援增強型運作狀態平台版本的運作狀態代理程式。運作狀態代理程式會監控 Web 伺服器日誌和系統指標，並將其轉送至 Elastic Beanstalk 服務。Elastic Beanstalk 會分析這些指標以及來自 Elastic Load Balancing 和 Amazon EC2 Auto Scaling 的資料，以提供環境運作狀態的整體概況。

除了能夠收集並呈現您環境資源的相關資訊，Elastic Beanstalk 亦可針對環境中的資源監控其多種錯誤條件，給予通知協助您避免故障並解決組態問題。影響環境運作狀態的因素包括應用程式每次處理請求的結果、執行個體作業系統的指標，以及最近的部署狀態。

您可使用 Elastic Beanstalk 主控台中的環境概觀頁面或 Elastic Beanstalk 命令列界面 (EB CLI) 中的 eb health 命令，即時檢視運作狀態。若要隨時間記錄並追蹤環境和執行個體運作狀態，環境可設定為

將 Elastic Beanstalk 收集的增強型運作狀態報告相關資訊，發佈至 Amazon CloudWatch 做為自訂指標。CloudWatch 的自訂指標費用適用所有指標，免費的 `EnvironmentHealth` 除外。

增強式運作狀態報告需要第 2 版或更新版本的平台版本。若要監控資源並發佈指標，您的環境必須同時具有執行個體描述檔和服務角色。根據預設，多容器 Docker 平台不含 web 伺服器，但若您將 web 伺服器設為以適當格式提供日誌，則可搭配增強式運作狀態報告使用。

> **Windows 平台備註**
>
> - 這個功能不適用於第 2 版 (v2) 以前的 Windows Server 平台版本。
> - 當您在 Windows Server 環境上啟用增強式運作狀態報告時，請不要變更 IIS 記錄日誌組態。若要讓增強式運作狀態監控正常運作，IIS 記錄日誌必須使用 W3C 格式和 ETW event only (僅 ETW 事件) 或 Both log file and ETW event (日誌檔案與 ETW 事件) 日誌事件目標進行設定。
>
>   此外，請不要停用或停止任何您環境執行個體上的 Elastic Beanstalk 運作狀態代理程式 Windows 服務。若要收集和報告執行個體的增強式運作狀態資訊，此服務必須處於啟用及執行中的狀態。

增強型運作狀態的環境需要執行個體描述檔。執行個體描述檔應有為環境執行個體提供權限的角色，以收集和回報增強型運作狀態資訊。當您第一次在 Elastic Beanstalk 主控台中透過 v2 平台版本建立環境時，Elastic Beanstalk 會提示您建立所需角色，並且根據預設會啟用增強型運作狀態報告。請繼續閱讀以了解增強型運作狀態報告的運作方式詳細資訊，或參閱 啟用 Elastic Beanstalk 增強型運作狀態報告以立即開始使用。

Amazon Linux 2 平台需要執行個體描述檔，這樣才能無條件支援增強型運作狀態。當您使用 Amazon Linux 2 平台建立環境時，Elastic Beanstalk 一律會啟用增強型運作狀態。無論您如何建立環境，包括使用 Elastic Beanstalk 主控台、EB CLI、AWS CLI 或 API，皆是如此。

主題

- 更新、部署和擴展期間的增強式運作狀態報告行為
- 啟用 Elastic Beanstalk 增強型運作狀態報告
- 使用環境管理主控台來進行增強型健全狀態監控
- 運作狀態顏色和狀態
- 執行個體指標
- 設定環境的增強型健康狀況規則
- 為環境發佈 Amazon CloudWatch 自訂指標
- 將增強型運作狀態報告與 Elastic Beanstalk API 搭配使用
- 增強型運作狀態日誌格式
- 通知與故障診斷

Elastic Beanstalk 運作狀態代理程式

Elastic Beanstalk 運作狀態代理程式是一種精靈程序 (或在 Windows 環境上的服務)，該程序會在您環境中的每個 Amazon EC2 執行個體上執行，監控作業系統和應用程式層級的運作狀態指標，並向 Elastic Beanstalk 報告問題。自每個平台的版本 2.0 起，所有平台版本均隨附運作狀態代理程式。

運作狀態代理程式會報告類似於 Amazon EC2 Auto Scaling 和 Elastic Load Balancing 發佈至 CloudWatch 的指標，做為基礎型運作狀態報告的一部分，其中包括 CPU 負載、HTTP 代碼和延遲。然而，運作狀態代理程式直接向 Elastic Beanstalk 報告的精細度和頻率，高於基礎型運作狀態報告。

基礎型運作狀態每 5 分鐘會發佈一次這些指標，並可於環境管理主控台以圖表監控。透過增強型運作狀態，Elastic Beanstalk 運作狀態代理程式每 10 秒會向 Elastic Beanstalk 報告指標。Elastic Beanstalk 會使用運作狀態代理程式提供的指標，判斷環境中各個執行個體的運作狀態，同時結合其他因素，判定環境的整體運作狀態。

環境的整體運作狀態可在 Elastic Beanstalk 主控台的環境概觀頁面中即時檢視，Elastic Beanstalk 也會每 60 秒將其發佈至 CloudWatch。您可以在 EB CLI 中使用 eb health 命令，即時檢視運作狀態代理程式報告的詳細指標。

若支付額外費用，您也可以選擇每 60 秒將個別執行個體和環境層級指標發佈至 CloudWatch。發佈至 CloudWatch 的指標之後可用來在環境管理主控台中建立監控圖表。

增強型運作狀態報告只在您選擇將增強型運作狀態指標發佈至 CloudWatch 時，才會產生費用。當您使用增強型運作狀態時，即使您選擇不要發佈增強型運作狀態指標，仍可免費發佈基礎型運作狀態指標。

如需運作狀態代理程式報告指標的詳細資訊，請參閱 執行個體指標。如需將增強型運作狀態指標發佈至 CloudWatch 的詳細資訊，請參閱為環境發佈 Amazon CloudWatch 自訂指標。

判斷執行個體和環境運作狀態的因素

除了基本運作狀態報告系統檢查之外 (包含 Elastic Load Balancing 運作狀態檢查 和資源監控)，Elastic Beanstalk 增強型運作狀態報告還會收集您環境中執行個體狀態的其他資料。這包含作業系統指標、伺服器日誌及正在進行的環境操作狀態 (例如部署和更新)。Elastic Beanstalk 運作狀態報告服務會結合所有可用來源的資訊，並加以分析以判斷環境的整體運作狀態。

操作和命令

當您在環境上執行操作時 (例如部署新的應用程式版本)，Elastic Beanstalk 會進行幾項影響您環境運作狀態的變更。

例如，當您將新的應用程式版本部署至執行多個執行個體的環境時，您可能會在使用 EB CLI 監控環境的運作狀態時，看到與以下內容相似的訊息。

```
  id              status     cause
    Overall       Info       Command is executing on 3 out of 5 instances
  i-bb65c145      Pending    91 % of CPU is in use. 24 % in I/O wait
                             Performing application deployment (running for 31 seconds)
  i-ba65c144      Pending    Performing initialization (running for 12 seconds)
  i-f6a2d525      Ok         Application deployment completed 23 seconds ago and took 26
seconds
  i-e8a2d53b      Pending    94 % of CPU is in use. 52 % in I/O wait
                             Performing application deployment (running for 33 seconds)
  i-e81cca40      Ok
```

在此範例中，環境整體狀態為 Ok，而成因是 Command is executing on 3 out of 5 instances (5 個執行個體中，命令正於其中 3 個上執行)。環境的三個執行個體狀態為 Pending (待定)，表示操作正在進行。

操作完成後，Elastic Beanstalk 會報告操作的其他資訊。例如，Elastic Beanstalk 會顯示下列有關執行個體已更新應用程式版本的資訊：

```
i-f6a2d525      Ok         Application deployment completed 23 seconds ago and took 26
 seconds
```

執行個體運作狀態資訊亦包含環境中每個執行個體最近部署的詳細資訊。各個執行個體都會報告部署 ID 和狀態。部署 ID 為整數，每次部署新的應用程式版本或變更執行個體上組態選項的設定 (如環境變數) 時，此整數會加一。滾動部署失敗後，您可使用部署資訊來辨識執行錯誤應用程式版本的執行個體。

在原因欄中，Elastic Beanstalk 包含的參考訊息包括成功操作及多次運作狀態檢查的其他正常運作狀態資訊，但這些項目無法永久保存。環境運作狀態不良的成因會加以保存，直到環境回傳良好運作狀態。

命令逾時

Elastic Beanstalk 在操作開始時會套用命令逾時，讓執行個體轉換為良好運作狀態。您可於環境更新及部署組態 (aws:elasticbeanstalk:command 命名空間中) 設定此命令逾時，預設為 10 分鐘。

在滾動更新期間，Elastic Beanstalk 會在操作的各個批次套用不同的逾時。您可於環境的滾動更新組態 (aws:autoscaling:updatepolicy:rollingupdate 命名空間中) 設定此逾時。若批次中的所有執行個體在滾動更新逾時內都維持良好運作狀態，操作會繼續進行下一批次。否則，操作會失敗。

> **Note**
>
> 若您的應用程式並未通過運作狀態檢查 (即未處於良好 (OK) 狀態)，但以另一層級而言仍屬穩定，您可以設定 `HealthCheckSuccessThreshold` 中的 `aws:elasticbeanstalk:command namespace` 選項，來變更 Elastic Beanstalk 將執行個體視為運作狀態良好的層級。

欲讓 Web 伺服器環境視為良好運作狀態，環境或批次中每個執行個體都必須在 2 分鐘內連續通過 12 個運作狀態檢查。針對工作者層環境，每個執行個體都必須通過 18 項運作狀態檢查。在命令逾時前，Elastic Beanstalk 不會在運作狀態檢查失敗時降低環境的運作狀態。若環境中的執行個體在命令逾時內恢復良好運作狀態，操作即為成功。

HTTP 請求

環境未進行操作時，執行個體和環境運作狀態的主要資訊來源為各個執行個體的 Web 伺服器日誌。為了判斷執行個體和環境整體的運作狀態，Elastic Beanstalk 會考量請求數目、每次請求結果，以及每次請求的解決速度。

在 Linux 類型的平台上，Elastic Beanstalk 會讀取並剖析 Web 伺服器日誌來取得 HTTP 請求的資訊。在 Windows Server 平台上，Elastic Beanstalk 會直接從 IIS Web 伺服器接收此資訊。

您的環境可能沒有作用中的 web 伺服器。例如，多容器 Docker 平台便不包含 web 伺服器。其他平台則會包含 web 伺服器，但您的應用程式可能會停用它。在這些情況下，您的環境便需要額外的組態，才能以其轉送資訊至 Elastic Beanstalk 服務時所需要的格式，提供 Elastic Beanstalk 運作狀態代理程式日誌。如需詳細資訊，請參閱「增強型運作狀態日誌格式」。

作業系統指標

Elastic Beanstalk 會監控運作狀態代理程式報告的作業系統指標，以辨識系統資源始終不足的執行個體。

如需運作狀態代理程式報告指標的詳細資訊，請參閱 執行個體指標。

運作狀態檢查規則自訂

Elastic Beanstalk 增強型運作狀態報告倚賴一組規則，以確定您環境的運作狀態。這些的部分規則可能不適用於您的特定應用程式。常用案例是一種應用程式，其會傳回設計上經常性 HTTP 4xx 錯誤。Elastic Beanstalk 使用其中一個預設規則，獲得出現差錯的結論，然後將您環境的運作狀態從正常變更為警告、降級或嚴重，視錯誤率而定。若要正確處理這種情況下，Elastic Beanstalk 可讓您設定此規則，並忽略應用程式 HTTP 4xx 錯誤。如需詳細資訊，請參閱設定環境的增強型健康狀況規則。

增強型運作狀態角色

增強型運作狀態報告需要兩個角色：Elastic Beanstalk 的服務角色和環境的執行個體描述檔。服務角色可讓 Elastic Beanstalk 代表您與其他 AWS 服務互動，以收集環境資源的相關資訊。執行個體描述檔可讓您環境中的執行個體將日誌檔案寫入 Amazon S3，並將增強的運作狀態資訊傳送至 Elastic Beanstalk 服務。

當您使用 Elastic Beanstalk 主控台或 EB CLI 建立 Elastic Beanstalk 環境時，Elastic Beanstalk 會建立預設服務角色，並將必要的受管理政策附加至您環境的預設執行個體描述檔。

若您使用 API、軟體開發套件或 AWS CLI 來建立環境，您必須事先建立這些角色，並在建立環境期間指定他們，以使用增強式運作狀態。如需為您環境建立適當角色的說明，請參閱服務角色、執行個體描述檔和使用者政策。

建議您針對執行個體描述檔和服務角色使用受管理政策。受管政策是 Elastic Beanstalk 維護的 AWS Identity and Access Management (IAM) 政策。使用受管理政策可確保您的環境擁有正常運作所需的所有許可。

對於執行個體描述檔，您可以分別針對 Web 伺服器層或工作者階層環境使用 `AWSElasticBeanstalkWebTier` 或 `AWSElasticBeanstalkWorkerTier` 受管理的策略。如需這兩個受管理執行個體描述檔政策的詳細資訊，請參閱 the section called “執行個體設定檔”。

增強的運作狀態授權

Elastic Beanstalk 執行個體描述檔受管的政策包含 `elasticbeanstalk:PutInstanceStatistics` 動作的許可。此動作不屬於 Elastic Beanstalk API 的一部分。環境執行個體在內部使用的是不同 API 的一部分，會將增強的運作狀態資訊傳達給 Elastic Beanstalk 服務。您不會直接呼叫這個 API。

建立新環境時，授權 `elasticbeanstalk:PutInstanceStatistics` 動作預設為啟用。若要增加環境的安全性，並協助防止運作狀態資料假冒您的身分進行詐騙，建議您讓此動作的授權保持啟用狀態。如果您為執行個體描述檔使用受管政策，這項功能可供您新環境使用，無需任何進一步設定。如果您使用自訂執行個體描述檔而非受管政策，您的環境可能顯示無資料運作狀態。發生這種情況是因為執行個體未經授權，無法將增強型運作狀態資料傳送至服務。

若要授權此動作，請在您的執行個體描述檔中包含下列陳述式。

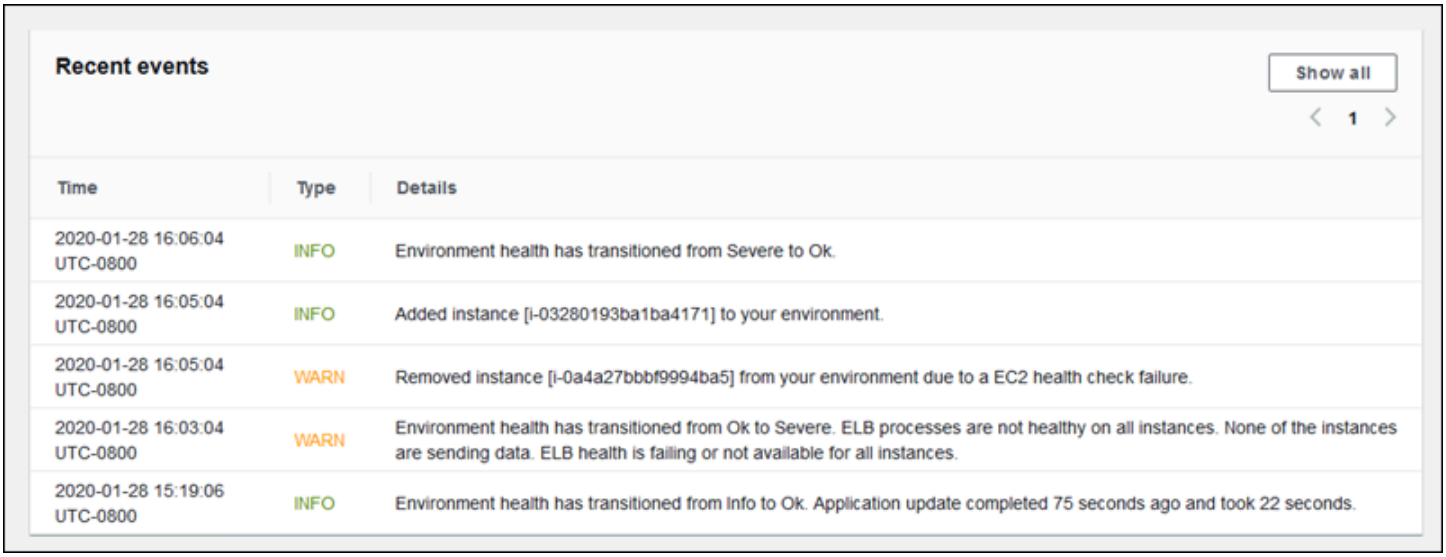
```
{
  "Sid": "ElasticBeanstalkHealthAccess",
  "Action": [
    "elasticbeanstalk:PutInstanceStatistics"
  ],
  "Effect": "Allow",
  "Resource": [
    "arn:aws:elasticbeanstalk:*:*:application/*",
    "arn:aws:elasticbeanstalk:*:*:environment/*"
  ]
}
```

如果您目前不想使用增強型運作狀態授權，請將 the section called “aws:elasticbeanstalk:healthreporting:system” 命名空間的 `EnhancedHealthAuthEnabled` 選項設定為 `false`，加以停用。如果停用此選項，就不需要先前說明的許可。您可以為您應用程式和環境的最低權限存取，從執行個體描述檔中移除這些許可。

> **Note**
>
> 之前 `EnhancedHealthAuthEnabled` 預設設定為 `false`，導致 `elasticbeanstalk:PutInstanceStatistics` 動作的授權也預設為停用。若要為現有環境啟用此動作，請將 the section called “aws:elasticbeanstalk:healthreporting:system” 命名空間的 `EnhancedHealthAuthEnabled` 選項設定為 `true`。您可以使用組態檔案中的選項設定來設定此選項。

增強型運作狀態事件

環境轉換狀態時，增強型運作狀態系統會產生事件。以下範例顯示了在資訊 (Info)、良好 (OK) 及嚴重 (Severe) 狀態間轉換的環境事件輸出。



當轉換成較嚴重的狀態時，增強式運作狀態事件會包含指出轉換原因的訊息。

並非所有執行個體層級的變更都會使 Elastic Beanstalk 發出事件。為了避免錯誤的警示，Elastic Beanstalk 只會在問題持續出現在多次檢查中時產生運作狀態相關的事件。

環境層級的即時運作狀態資訊 (包括狀態、顏色和原因) 都可在 Elastic Beanstalk 主控台的環境概觀頁面和 EB CLI 取得。透過將 EB CLI 連接至您的環境並執行 eb health 命令，您也可以檢視環境中各個執行個體的即時狀態。

更新、部署和擴展期間的增強式運作狀態報告行為

啟用增強型運作狀態報告會影響您的環境在組態更新和部署期間的行為。Elastic Beanstalk 在直到所有執行個體皆一致通過運作狀態檢查之前，都不會完成更新批次。此外，因為增強式運作狀態報告會套用運作狀態的更高標準並監控更多因素，通過基本運作狀態報告 ELB 運作狀態檢查的執行個體，不一定會同時通過增強式運作狀態報告。如需運作狀態檢查如何影響更新程序的詳細資訊，請參閱滾動組態更新和滾動部署的主題。

增強型運作狀態報告亦可強調 Elastic Load Balancing 須設定適當的運作狀態檢查 URL 之需求。當您的環境為滿足需求而擴展，新的執行個體一通過 ELB 運作狀態檢查就會開始處理請求。若未設定運作狀態檢查 URL，執行個體接受 TCP 連線後 20 秒即可開始。

在負載平衡器宣告環境運作狀態良好前，若您的應用程式尚未完成啟動以接收流量，將出現大量失敗的請求，而您環境的運作狀態檢查會開始失敗。命中由您應用程式所提供路徑的運作狀態檢查 URL 可避

免此問題。在直到向運作狀態檢查 URL 發出的 GET 請求傳回 200 狀態代碼前，ELB 運作狀態檢查都不會通過。

啟用 Elastic Beanstalk 增強型運作狀態報告

以最新平台版本建立的新環境包含 AWS Elastic Beanstalk 運作狀態代理程式，可支援增強型運作狀態報告。若您在 Elastic Beanstalk 主控台中，或透過 EB CLI 建立您的環境，就會依預設啟用增強型運作狀態。您亦可使用組態檔案，於應用程式原始碼中設定您的運作狀態報告偏好。

增強型運作狀態報告需要執行個體描述檔和具備一組標準許可的服務角色。當您在 Elastic Beanstalk 主控台中建立環境時，Elastic Beanstalk 會自動建立所需的角色。請參閱開始使用 Elastic Beanstalk 以了解如何建立您的第一個環境。

主題

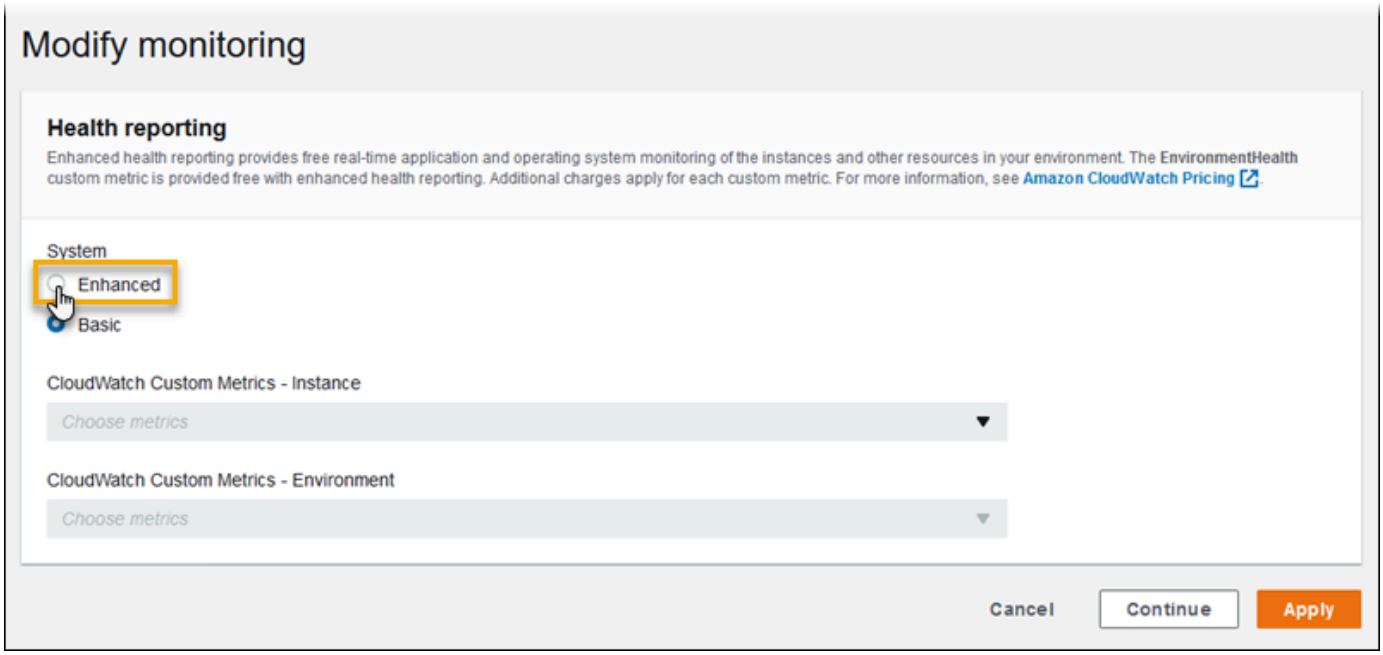
使用 Elastic Beanstalk 主控台啟用增強型運作狀態報告

使用 Elastic Beanstalk 主控台，在執行中的環境內啟用增強型運作狀態報告

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

   > **Note**
   >
   > 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇 Configuration (組態)。
4. 在 Monitoring (監控) 組態類別中，選擇 Edit (編輯)。
5. 於 Health Reporting (運作狀態報告) 下的 System (系統) 中，選擇 Enhanced (增強型)。



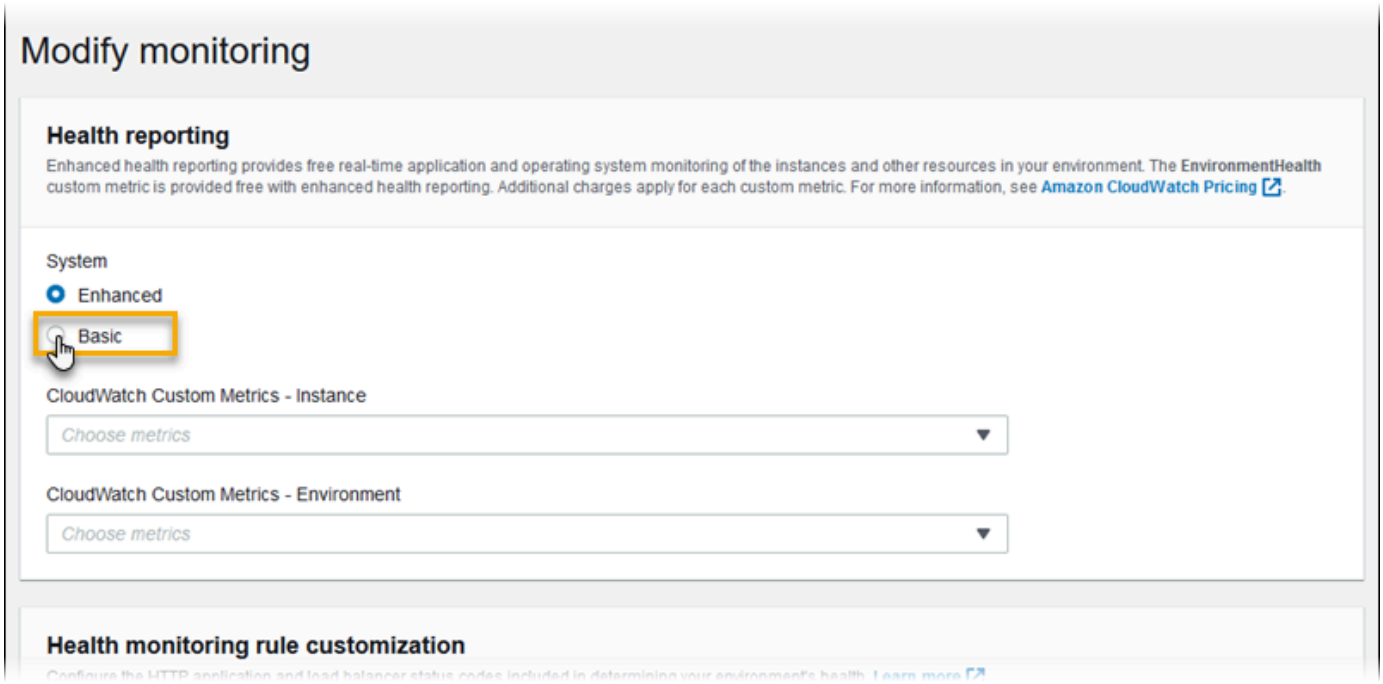
> **Note**
>
> 若您使用不支援的平台或版本，將不會出現增強型運作狀態報告的選項。

6. 若要儲存變更，請選擇頁面底部的儲存變更。

當您透過第 2 版 (v2) 平台版本建立新環境時，根據預設，Elastic Beanstalk 主控台會啟用增強型運作狀態報告。您可於環境建立期間變更運作狀態報告選項，藉此停用增強型運作狀態報告。

使用 Elastic Beanstalk 主控台，在建立環境時停用增強型運作狀態報告

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 建立應用程式或選取現有程式。
3. 建立環境。在 Create a new environment (建立新環境) 頁面，在選擇 Create environment (建立環境) 之前，選擇 Configure more options (設定更多選項)。
4. 在 Monitoring (監控) 組態類別中，選擇 Edit (編輯)。
5. 於 Health Reporting (運作狀態報告) 下的 System (系統) 中選擇 Basic (基本)。



6. 選擇 Save (儲存)。

使用 EB CLI 啟用增強式運作狀態報告

當您透過 eb create 命令建立新環境時，EB CLI 預設會啟用增強型運作狀態報告，並套用預設執行個體描述檔和服務角色。

您可以使用 `--service-role` 選向，透過名稱指定不同的服務角色。

若您有在 v2 平台版本上搭配基本運作狀態報告執行的環境，而您希望切換至增強式運作狀態，請遵循這些步驟。

使用 EB CLI 於執行環境中啟用增強型運作狀態

1. 使用 eb config 命令於預設文字編輯器開啟組態檔案。

```
~/project$ eb config
```

2. 於設定區段找尋 `aws:elasticbeanstalk:environment` 命名空間。請確認 `ServiceRole` 的值不是 null，且符合您的服務角色的名稱。

```
  aws:elasticbeanstalk:environment:
    EnvironmentType: LoadBalanced
    ServiceRole: aws-elasticbeanstalk-service-role
```

3. 於 aws:elasticbeanstalk:healthreporting:system: 命名空間之下，將 SystemType 的值變更為 **enhanced**。

```
  aws:elasticbeanstalk:healthreporting:system:
    SystemType: enhanced
```

4. 儲存組態檔案並關閉文字編輯器。
5. EB CLI 會開始進行環境更新，以套用您的組態變更。等待操作完成，或按 Ctrl+C 以安全離開。

```
~/project$ eb config
Printing Status:
INFO: Environment update is starting.
INFO: Health reporting type changed to ENHANCED.
INFO: Updating environment no-role-test's configuration settings.
```

使用組態檔案啟用增強式運作狀態報告

您可於原始碼套件納入組態檔案，藉此啟用增強型運作狀態報告。下列範例呈現的組態檔案，可啟用增強型運作狀態報告，並將預設服務和執行個體描述檔指派至環境：

Example .ebextensions/enhanced-health.config

```
option_settings:
  aws:elasticbeanstalk:healthreporting:system:
    SystemType: enhanced
  aws:autoscaling:launchconfiguration:
    IamInstanceProfile: aws-elasticbeanstalk-ec2-role
  aws:elasticbeanstalk:environment:
    ServiceRole: aws-elasticbeanstalk-service-role
```

若您已建立自己的執行個體描述檔或服務角色，請將反白顯示的文字取代為這些角色的名稱。

使用環境管理主控台來進行增強型健全狀態監控

如果您在 AWS Elastic Beanstalk 中啟用增強型運作狀態報告功能，就可以透過環境管理主控台來監控環境的運作狀態。

主題

- 環境概觀

- 環境健全狀態頁面
- 監控頁面

環境概觀

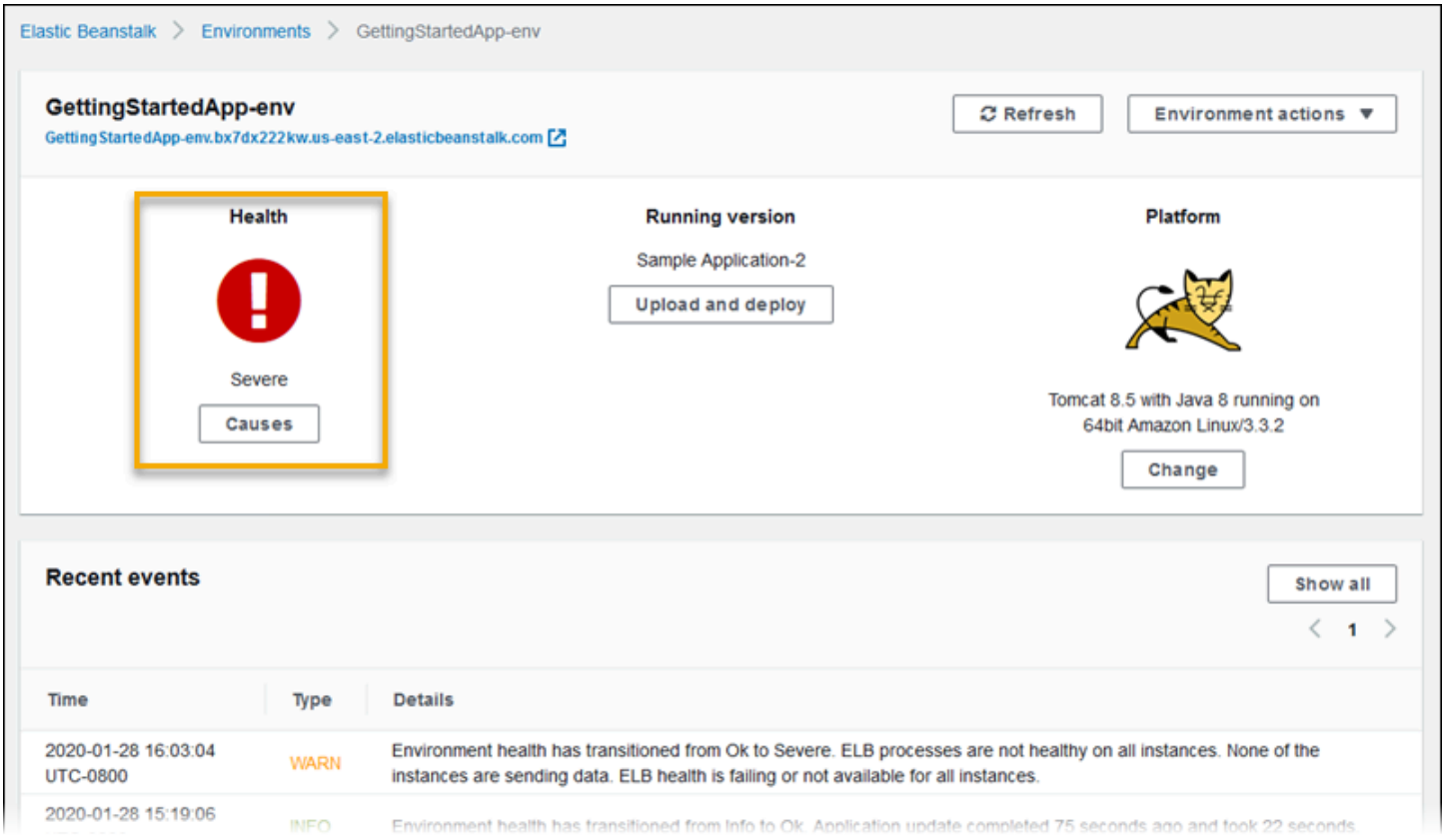
環境概觀會顯示環境的運作狀態和列出事件，這些事件提供了關於運作狀態最近變化的資訊。

檢視環境概觀

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

如需目前環境運作狀態的詳細資訊，請選擇 Causes(原因) 來開啟 Health (運作狀態) 頁面。或者，在導覽窗格中，選擇 Health (運作狀態)。



環境健全狀態頁面

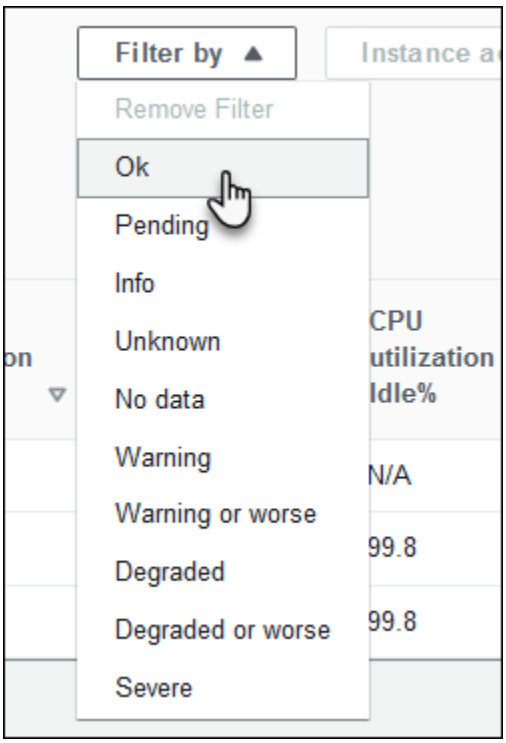
運作狀態頁面會針對環境和環境中的每個 Amazon EC2 執行個體，顯示運作狀態、指標與原因。

> **Note**
>
> 只有在您已針對環境啟用增強型運作狀態監控時，Elastic Beanstalk 才會顯示 Health (運作狀態)頁面。

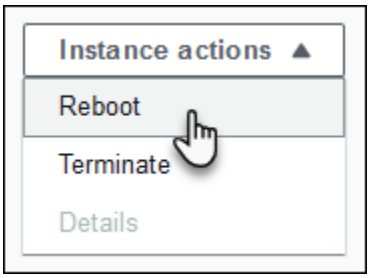
下圖顯示了 Linux 環境的 Health (運作狀態) 頁面。

下圖顯示了 Windows 環境的 Health (運作狀態) 頁面。請注意，CPU 指標與 Linux 環境上的指標不同。

您可以在頁面頂端查看環境執行個體的總數，以及每個狀態的執行個體數量。若要只顯示具有特定狀態的執行個體，請選擇 Filter By (篩選條件)，然後選取 status (狀態)。



若要重新啟動或終止健全狀態不佳的執行個體，請選擇 Instance Actions (執行個體動作)，然後選擇 Reboot (重新啟動) 或 Terminate (終止)。

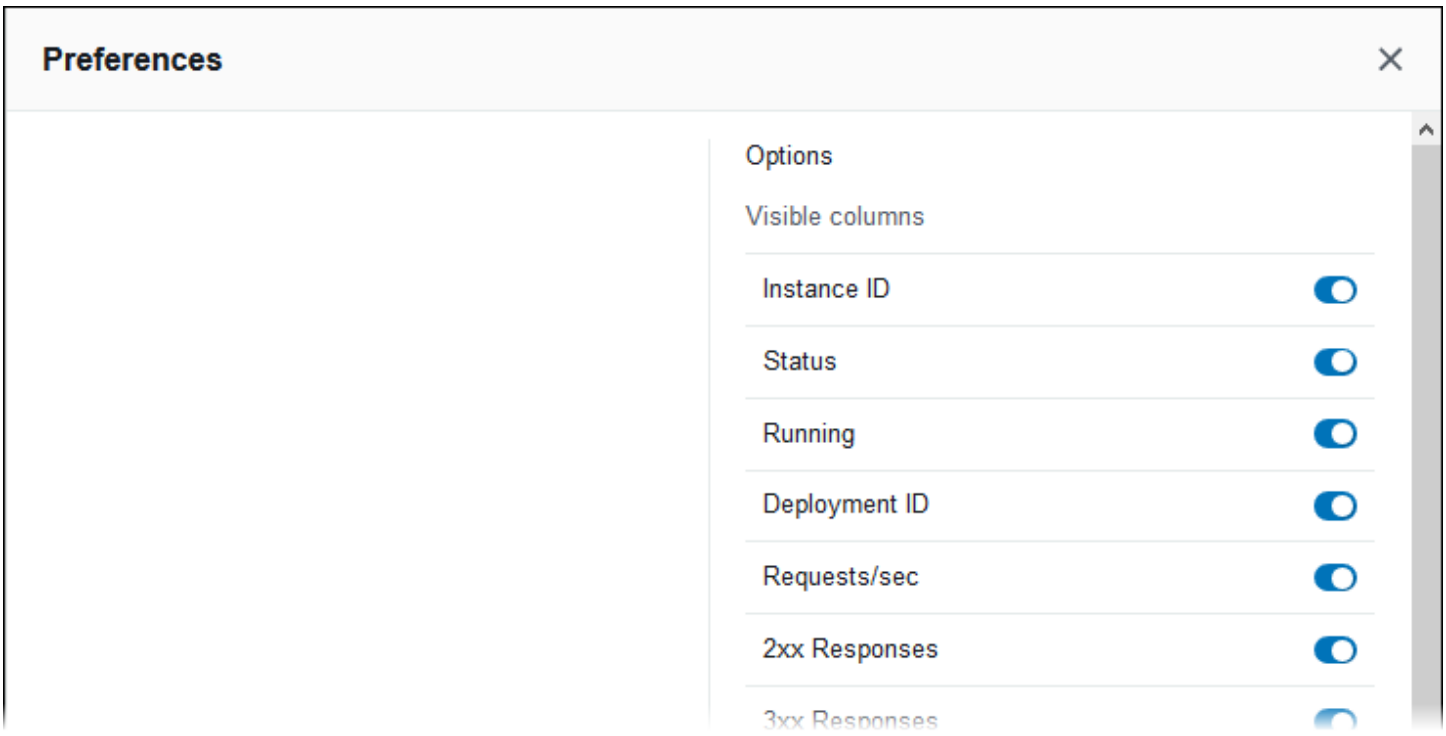


Elastic Beanstalk 會每 10 秒更新一次運作狀態頁面。它會報告環境和執行個體運作狀態的相關資訊。

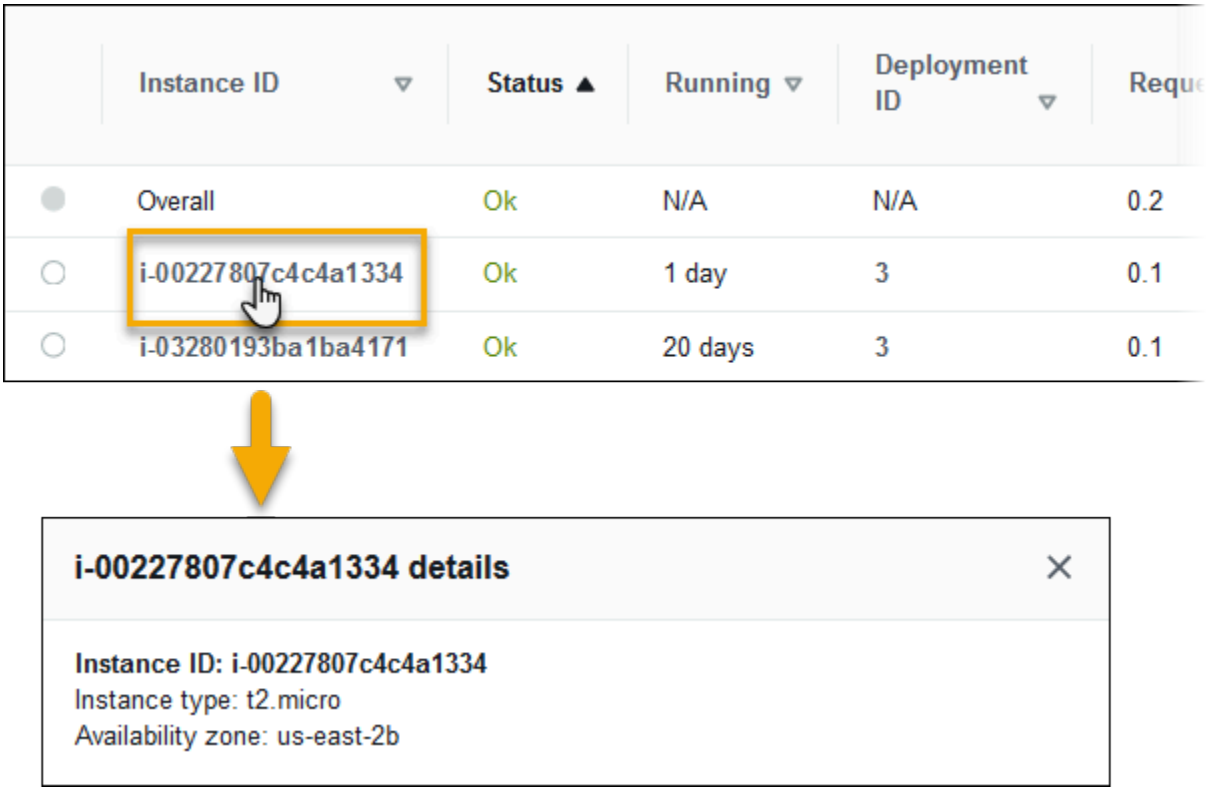
對於環境中的每個 Amazon EC2 執行個體，頁面會顯示執行個體的 ID 和狀態、執行處理個體後的時間量、最近在執行個體上執行的部署 ID、服務的執行個體要求的回應和延遲，以及負載和 CPU 使用率資訊。Overall (整體) 資料列會顯示整體環境的平均回應和延遲資訊。

此頁面會以非常寬的資料表顯示許多詳細資訊。若要隱藏部分欄位，請選擇

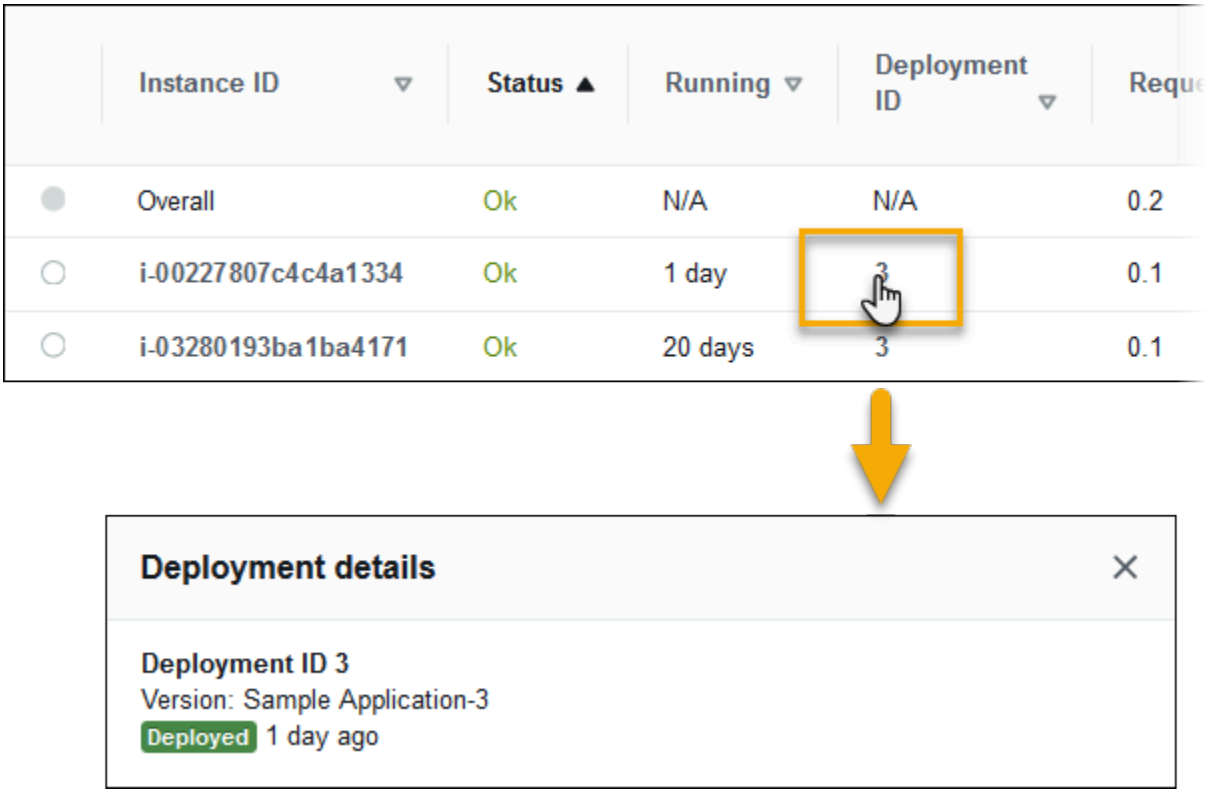
(偏好設定)。選取或清除欄位名稱，然後選擇 Confirm (確認)。



選擇任何執行個體的 Instance ID (執行個體 ID)，以檢視有關執行個體的詳細資訊，包括其可用區域和執行個體類型。



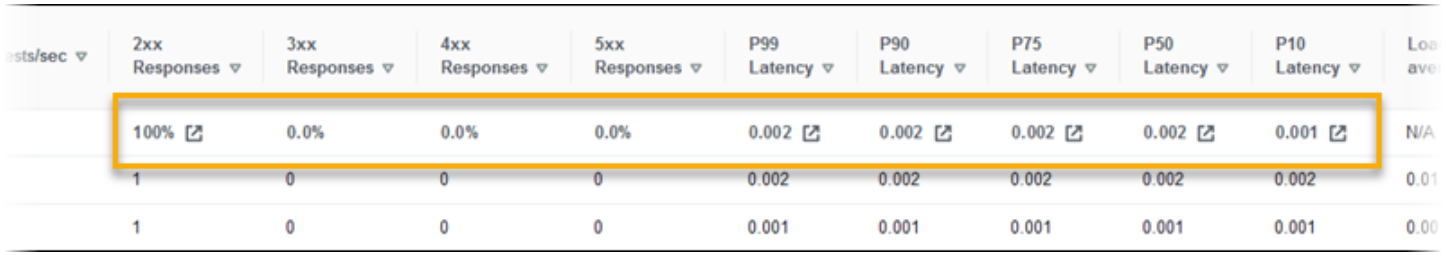
選擇任何執行個體的 Deployment ID (部署 ID)，以檢視執行個體最新 deployment (部署) 的詳細資訊。



部署資訊包含下列項目：

- 部署 ID — 部署的唯一識別符。部署 ID 從 1 開始，每次當您部署新的應用程式版本或變更組態設定時 (這些設定會影響您環境執行個體上所執行的軟體或作業系統)，此 ID 就會增加 1。
- 版本 — 部署中所使用應用程式原始程式碼的版本標籤。
- 狀態 — 部署的狀態，可以是 `In Progress`、`Deployed` 或 `Failed`。
- 時間 — 對於進行中的部署，這是其部署開始的時間。對於已完成的部署，這是其部署結束的時間。

如果您在環境中啟用 X-Ray 整合，並使用 AWS X-Ray 開發套件來檢測您的應用程式，則 Health (運作狀態) 頁面會在概觀列中，新增指向 AWS X-Ray 主控台的連結。



如果選取連結，可在 AWS X-Ray 主控台中，檢視與反白顯示的統計資料相關的追蹤資料。

監控頁面

監控頁面會針對增強型運作狀態報告系統所產生的自訂 Amazon CloudWatch 指標，顯示摘要統計資料與圖表。如需在此頁面中新增圖表和統計資料的相關指示，請參閱在 AWS 管理主控台中監控環境的運作狀態。

運作狀態顏色和狀態

增強式運作狀態報告會使用四種顏色，代表執行個體和整體環境的運作狀態，與基本運作狀態報告相似。增強式運作狀態報告亦提供七種運作狀態，採用單一詞彙描述，更完整地表示您環境的狀態。

執行個體狀態與環境狀態

每次 Elastic Beanstalk 針對您的環境執行運作狀態檢查，增強型運作狀態報告會分析所有可用資料，來檢查環境中的各個執行個體運作狀態。若任何較低層級的檢查失敗，Elastic Beanstalk 會調降執行個體的運作狀態等級。

Elastic Beanstalk 會在環境管理主控台顯示整體環境的運作狀態資訊 (顏色、狀態與成因)。此資訊亦可自 EB CLI 取得。個別執行個體的運作狀態與原因訊息每 10 秒會更新一次，而您可以在使用 eb health 檢視運作狀態時，從 EB CLI 取得此資訊。

Elastic Beanstalk 會使用執行個體運作狀態的變化，藉此評估環境的運作狀態，但不會立即變更環境運作狀態。若執行個體在任一分鐘內未通過運作狀態檢查達三次，Elastic Beanstalk 可能會調降環境的運作狀態等級。視環境中執行個體數量及所辨識的問題而定，一個運作狀態不良的執行個體可能會使 Elastic Beanstalk 顯示資訊訊息，或變更環境的運作狀態，將顏色從綠色 (良好) 調整為黃色 (警告) 或紅色 (降級或嚴重)。

正常 (綠色)

此狀態會在以下時機顯示：

- 執行個體通過運作狀態檢查，且運作狀態代理程式未報告任何問題。
- 環境中多數執行個體均通過運作狀態檢查，且運作狀態代理程式未報告重大問題。
- 執行個體通過運作狀態檢查且正常完成請求。

範例：您最近部署的環境正常處理請求。5% 的請求回傳 400 系列的錯誤。各個執行個體上的部署均正常完成。

訊息 (執行個體)：應用程式部署於 23 秒前完成，總共費時 26 秒。

警告 (黃色)

此狀態會在以下時機顯示：

- 運作狀態代理程式報告某一執行個體或環境出現中等數量的請求失敗或其他問題。
- 執行個體上正在進行某個操作，且花費很長的時間。

範例：環境中一個執行個體的狀態為嚴重。

訊息 (環境)：5 個執行個體中其中 1 個服務受損。

降級 (紅色)

此狀態會在運作狀態代理程式報告某一執行個體或環境出現大量請求失敗或其他問題時顯示。

範例：環境正在向上擴展至 5 個執行個體。

訊息 (環境)：4 個作用中的執行個體低於 Auto Scaling 群組的大小最低限制 5。

嚴重 (紅色)

此狀態會在運作狀態代理程式報告某一執行個體或環境出現極大量請求失敗或其他問題時顯示。

範例：Elastic Beanstalk 無法聯絡負載平衡器以取得執行個體運作狀態。

訊息 (環境)：ELB 運作狀態不良，或無法取得所有執行個體的運作狀態。無執行個體正在傳送資料。無法擔任角色「arn:aws:iam::123456789012:role/aws-elasticbeanstalk-service-role」。驗證角色是否存在，以及設定是否正確。

訊息 (執行個體)：執行個體 ELB 運作狀態已處於無法使用狀態達 37 分鐘。無資料。最後的資料為 37 分鐘之前。

資訊 (綠色)

此狀態會在以下時機顯示：

- 執行個體上正在進行某個操作。
- 環境中數個執行個體正在進行某個操作。

範例：正將新的應用程式版本部署至執行中的執行個體。

訊息 (環境)：5 個執行個體中的其中 3 個正在執行命令。

訊息 (執行個體)：正在執行應用程式部署 (已執行 3 秒)。

待定 (灰色)

此狀態會在執行個體上於命令逾時範圍內正在進行某個操作時顯示。

範例：您最近已建立環境，且正在引導執行個體。

訊息：正在執行初始化 (已執行 12 秒)。

未知 (灰色)

此狀態會在 Elastic Beanstalk 及運作狀態代理程式報告執行個體上的資料量不足時顯示。

範例：未接收到任何資料。

暫停 (灰色)

此狀態會在 Elastic Beanstalk 停止監控環境的運作狀態時顯示。環境可能無法正常運作。某些嚴重的運作狀態如果持續太久，會導致 Elastic Beanstalk 將環境轉換成暫停狀態。

範例：Elastic Beanstalk 無法存取環境的服務角色。

範例：已刪除 Elastic Beanstalk 為環境建立的 Auto Scaling 群組。

訊息：環境運作狀態已從良好轉換成嚴重。沒有任何執行個體。已將 Auto Scaling 群組所需的容量設為 1。

執行個體指標

執行個體指標可提供您環境中執行個體運作狀態的資訊。Elastic Beanstalk 運作狀態代理程式會在每個執行個體上執行。它會收集並將執行個體的指標轉送給 Elastic Beanstalk，讓其分析指標來判斷您環境中執行個體的運作狀態。

執行個體上的 Elastic Beanstalk 運作狀態代理程式會從 Web 伺服器和作業系統收集執行個體的指標。為了取得 Linux 類型平台上的 Web 伺服器資訊，Elastic Beanstalk 會讀取和剖析 Web 伺服器日誌。在 Windows Server 平台上，Elastic Beanstalk 會直接從 IIS Web 伺服器接收此資訊。Web 伺服器會提供傳入 HTTP 請求的資訊：傳入的請求數量、導致錯誤的請求數量，以及解決錯誤的時間長度。作業系統會提供執行個體資源狀態的快照資訊：CPU 負載和每個處理類型的用時分配。

運作狀態代理程式會收集 Web 伺服器和作業系統指標，並每 10 秒轉送至 Elastic Beanstalk。Elastic Beanstalk 會分析資料，並使用結果來更新每個執行個體和環境的運作狀態。

主題

Web 伺服器指標

在以 Linux 為基礎的平台上，Elastic Beanstalk 運作狀態代理程式會從日誌讀取 Web 伺服器指標，這些日誌來自環境中執行個體處理請求的 Web 容器或伺服器。Elastic Beanstalk 平台可設定為產生兩個日誌：一個是人類可讀的格式，另一個則是機器可讀的格式。運作狀態代理程式每 10 秒會將供機器閱讀的日誌轉送至 Elastic Beanstalk。

如需 Elastic Beanstalk 使用的日誌格式的詳細資訊，請參閱增強型運作狀態日誌格式。

在 Windows Server 平台上，Elastic Beanstalk 會將模組新增至 IIS Web 伺服器的請求管道，並擷取 HTTP 請求時間和回應代碼的指標。模組會使用高效能處理序間通訊 (IPC) 通道，將這些指標傳送到執行個體上的運作狀態代理程式。如需實作的詳細資訊，請參閱 Windows Server 上 IIS 內擷取到的 Web 伺服器指標。

報告的 Web 伺服器指標

`RequestCount`

過去 10 秒內 Web 伺服器每秒處理請求的數目。會在 EB CLI 及 環境健全狀態頁面 中顯示為平均 `r/sec` (每秒請求數)。

`Status2xx`, `Status3xx`, `Status4xx`, `Status5xx`

過去 10 秒導致每個類型的狀態碼的請求數量。例如，成功的請求會傳回 200 OK；重新導向則傳回 301；若輸入的 URL 不符合應用程式內的任何資源，則傳回 404。

EB CLI 和 環境健全狀態頁面 顯示這些指標的格式包括：執行個體的請求原始數據，以及佔環境整體請求的比例。

`p99.9`, `p99`, `p95`, `p90`, `p85`, `p75`, `p50`, `p10`

過去 10 秒最慢 x% 的請求的平均延遲，其中 x 為數字與 100 的差。例如，`p99 1.403` 表示過去 10 秒最慢的 1% 請求的平均延遲為 1.403 秒。

作業系統指標

Elastic Beanstalk 運作狀態代理程式會報告下列作業系統指標。Elastic Beanstalk 使用這些指標來辨識持續承受高負載的執行個體。指標會因作業系統而不同。

報告的作業系統指標 — Linux

`Running`

執行個體啟動後已經過的時間。

`Load 1`, `Load 5`

過去 1 分鐘和 5 分鐘期間的平均負載。以十進位值顯示，表示那段期間程序執行的平均數量。若該數值高於可用 vCPU (執行緒) 的數量，則剩餘部分則表示等待中的程序平均數量。

例如，若您的執行個體類型有四個 vCPU，且負載為 4.5，則平均有 .5 個程序在那段期間處於等待狀態，相當於一個程序的等待時間佔了 50%。

`User %`, `Nice %`, `System %`, `Idle %`, `I/O Wait %`

過去 10 秒內 CPU 花在每個狀態的時間百分比。

報告的作業系統指標 — Windows

`Running`

執行個體啟動後已經過的時間。

`% User Time`, `% Privileged Time`, `% Idle Time`

過去 10 秒內 CPU 花在每個狀態的時間百分比。

Windows Server 上 IIS 內擷取到的 Web 伺服器指標

在 Windows Server 平台上，Elastic Beanstalk 會將模組新增至 IIS Web 伺服器的請求管道，並擷取 HTTP 請求時間和回應代碼的指標。模組會使用高效能處理序間通訊 (IPC) 通道，將這些指標傳送到執行個體上的運作狀態代理程式。運作狀態代理程式會彙整這些指標，將它們與作業系統指標合併，並傳送至 Elastic Beanstalk 服務。

實作詳細資訊

為了從 IIS 擷取指標，Elastic Beanstalk 會實作受管 `IHttpModule`，並訂閱 `BeginRequest` 和 `EndRequest` 事件。這可讓模組報告所有由 IIS 處理的 web 請求 HTTP 請求延遲及回應代碼。為了

將模組新增至 IIS 請求管道，Elastic Beanstalk 會在 IIS 組態檔案 `%windir%\System32\inetsrv\config\applicationHost.config` 的 `<modules>` 區段中註冊模組。

IIS 中的 Elastic Beanstalk 模組會將擷取到的 web 請求指標傳送到執行個體上的運作狀態代理程式，該程式是名為 `HealthD` 的 Windows 服務。為了傳送此資料，模組會使用 `NetNamedPipeBinding`，提供針對機器上通訊最佳化的安全、可靠繫結。

設定環境的增強型健康狀況規則

AWS Elastic Beanstalk 增強型運作狀態報告倚賴一組規則，以確定您環境的運作狀態。這些的部分規則可能不適用於您的特定應用程式。以下是一些常見範例：

- 您使用用戶端測試工具。在此案例中，預期經常發生 HTTP 用戶端 (4xx) 錯誤。
- 您可以將 AWS WAF 與環境的 Application Load Balancer 搭配使用，以封鎖不必要的傳入流量。在此案例中，Application Load Balancer 會為每個拒絕的傳入訊息傳回 HTTP 403。

根據預設，在判斷環境的運作狀態時，Elastic Beanstalk 包含所有應用程式 HTTP 4xx 錯誤。它會根據錯誤率，將您的環境運作狀態從 OK (正常) 變更為 Warning (警告)、Degraded (降級) 或 Severe (嚴重)。若要正確處理如範例中所提的情況，Elastic Beanstalk 可讓您設定一些增強型運作狀態規則。您可以選擇忽略環境執行個體上的應用程式 HTTP 4xx 錯誤，或忽略環境負載平衡器傳回的 HTTP 4xx 錯誤。本主題說明如何進行這些組態變更。

> **Note**
>
> 目前，這是唯一可用的增強型運作狀態規則自訂。您無法將增強型運作狀態設為忽略 4xx 以外的其他 HTTP 錯誤。

使用 Elastic Beanstalk 主控台設定進階型運作狀態規則

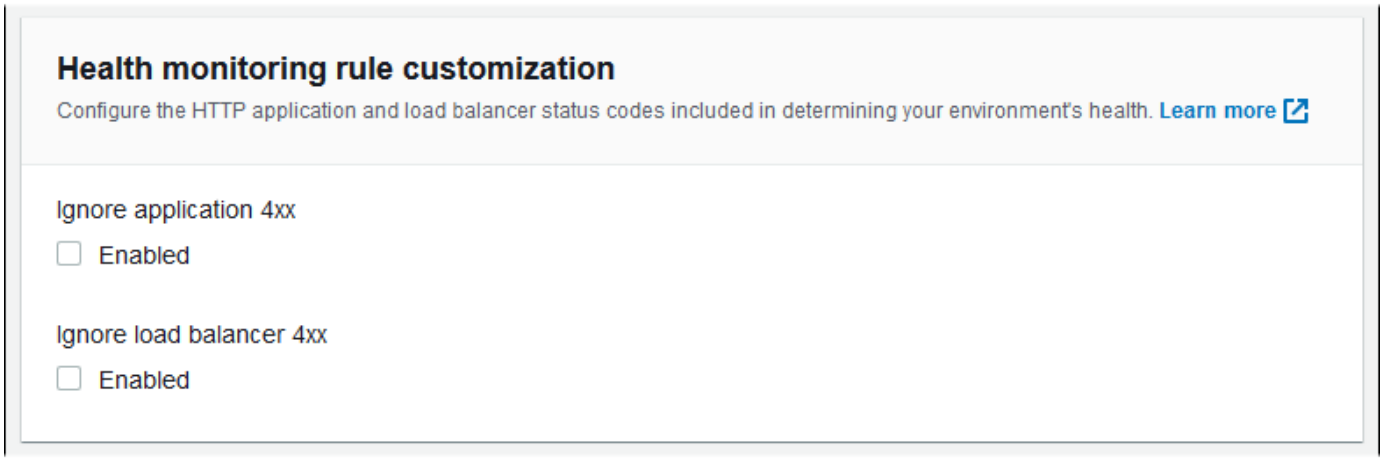
您可以使用 Elastic Beanstalk 主控台在您環境中設定增強型運作狀態規則。

使用 Elastic Beanstalk 主控台設定 HTTP 4xx 狀態碼檢查

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇 Configuration (組態)。
4. 在 Monitoring (監控) 組態類別中，選擇 Edit (編輯)。
5. 在 Health monitoring rule customization (運作狀態監控規則自訂) 下，啟用或停用所需的 Ignore (忽略) 選項。



6. 若要儲存變更，請選擇頁面底部的儲存變更。

使用 EB CLI 設定增強型運作狀態規則

您可以透過在本機儲存環境組態、新增設定增強型運作狀態規則的項目，然後將組態上傳到 Elastic Beanstalk。您可在建立期間或之後將儲存的設定套用到環境中。

使用 EB CLI 和儲存的組態設定 HTTP 4xx 狀態碼檢查

1. 初始化您的專案資料夾 eb init。
2. 透過執行 eb create 命令來建立環境。
3. 透過執行 eb config save 命令以在本機儲存組態範本。以下範例使用 `--cfg` 選項以指定組態的名稱。

```
$ eb config save --cfg 01-base-state
Configuration saved at: ~/project/.elasticbeanstalk/saved_configs/01-base-
state.cfg.yml
```

4. 在文字編輯器中開啟已儲存的組態檔案。
5. 在 OptionSettings > aws:elasticbeanstalk:healthreporting:system: 下，新增 ConfigDocument 金鑰以列出每個增強型運作狀態規則來做設定。以下 ConfigDocument 停用應用程式 HTTP 4xx 狀態碼的檢查，同時保持負載平衡器 HTTP 4xx 代碼的檢查。

```
OptionSettings:
  ...
  aws:elasticbeanstalk:healthreporting:system:
    ConfigDocument:
      Rules:
        Environment:
          Application:
            ApplicationRequests4xx:
              Enabled: false
          ELB:
            ELBRequests4xx:
              Enabled: true
      Version: 1
    SystemType: enhanced
...
```

> **Note**
>
> 您可將 Rules 和 CloudWatchMetrics 合併在相同的 ConfigDocument 選項設定中。CloudWatchMetrics 會在 為環境發佈 Amazon CloudWatch 自訂指標 中詳加說明。
>
> 如果您之前已啟用 CloudWatchMetrics，使用 eb config save 命令擷取的組態檔案已經擁有 ConfigDocument 金鑰及 CloudWatchMetrics 部分。請勿刪除 — 將 Rules 區段加入到相同的 ConfigDocument 選項值。

6. 儲存組態檔案並關閉文字編輯器。在這個範例中，已更新的組態檔案是以不同於下載的組態檔案名稱儲存 (02-cloudwatch-enabled.cfg.yml)。這會在檔案上傳時建立單獨的已儲存組態。您可以使用與下載的檔案相同的名稱，以覆寫現有的組態，不用建立新的名稱。
7. 使用 eb config put 命令上傳更新的組態檔案到 Elastic Beanstalk。

```
$ eb config put 02-cloudwatch-enabled
```

當您在已儲存的組態使用 eb config get 和 put 命令時，請不要包含副檔名。

8. 套用儲存的組態至您的執行環境。

```
$ eb config --cfg 02-cloudwatch-enabled
```

該 `--cfg` 選項會指定一個已套用至環境並已命名的組態檔案。您可以在本機儲存組態檔，或可儲存在 Elastic Beanstalk 中。如果已指定名稱的組態檔案同時存在兩個位置中，則 EB CLI 會使用本機檔案。

使用 Config 文件設定增強型運作狀態規則

適用於增強型運作狀態規則的組態 (config) 文件是 JSON 文件，其中列出要設定的規則。

以下範例顯示的 Config 文件停用應用程式 HTTP 4xx 狀態碼的檢查，並啟用負載平衡器 HTTP 4xx 狀態碼的檢查。

```
{
  "Rules": {
    "Environment": {
      "Application": {
        "ApplicationRequests4xx": {
          "Enabled": false
        }
      },
      "ELB": {
        "ELBRequests4xx": {
          "Enabled": true
        }
      }
    }
  },
  "Version": 1
}
```

對於 AWS CLI，您將文件當做 `Value` 金鑰值在選項設定引數中傳遞，而其本身是一種 JSON 物件。在這種情況下，您必須將嵌入文件中的引號逸出。以下命令會檢查組態設定是否有效。

```
$ aws elasticbeanstalk validate-configuration-settings --application-name my-app --
environment-name my-env --option-settings '[
    {
        "Namespace": "aws:elasticbeanstalk:healthreporting:system",
```

```
        "OptionName": "ConfigDocument",
        "Value": "{\"Rules\": { \"Environment\": { \"Application\":
 { \"ApplicationRequests4xx\": { \"Enabled\": false } }, \"ELB\": { \"ELBRequests4xx\":
 {\"Enabled\": true } } } }, \"Version\": 1 }"
    }
]'
```

如需 YAML.ebextensions 中的組態檔案，您可以依原狀提供 JSON 文件。

```
  option_settings:
    - namespace: aws:elasticbeanstalk:healthreporting:system
      option_name: ConfigDocument
      value: {
  "Rules": {
    "Environment": {
      "Application": {
        "ApplicationRequests4xx": {
          "Enabled": false
        }
      },
      "ELB": {
        "ELBRequests4xx": {
          "Enabled": true
        }
      }
    }
  },
  "Version": 1
}
```

為環境發佈 Amazon CloudWatch 自訂指標

您可以將 AWS Elastic Beanstalk 增強型運作狀態報告所收集到的資料，發佈到 Amazon CloudWatch 以做為自訂指標。發佈指標到 CloudWatch 以讓您隨時監控應用程式效能變化，並透過追蹤資源使用情況並要求延遲擴展負載來找出潛在的問題。

透過發佈指標至 CloudWatch，您也可藉由監控圖表和警示讓它們可供使用。一個免費指標，當您使用增強型運作狀態狀況報告，會自動啟用 EnvironmentHealth。EnvironmentHealth 以外的自訂指標會產生標準 CloudWatch 費用。

若要為環境發佈 CloudWatch 自訂指標，您必須在環境中先啟用增強型運作狀態報告。如需說明，請參閱 啟用 Elastic Beanstalk 增強型運作狀態報告。

主題

增強型運作狀態報告指標

當您在環境中啟用增強型運作狀態報告時，增強型運作狀態報告系統會自動發佈一個 CloudWatch 自訂指標 (EnvironmentHealth)。若要發佈其他指標至 CloudWatch，可透過使用 Elastic Beanstalk 主控台、EB CLI 或 .ebextensions，利用那些指標來設定您的環境。

您可以從您的環境發佈以下增強型運作狀態指標至 CloudWatch。

可用指標 — 所有平台

`EnvironmentHealth`

僅用於環境。除非設定其他指標，否則這是唯一增強型運作狀態報告系統發佈的 CloudWatch 指標。環境運作狀態係由七種狀態其中之一來表示。在 CloudWatch 主控台中，這些狀態對應到以下值：

- 0 - 良好
- 1 - 資訊
- 5 - 不明
- 10 - 無資料
- 15 - 警告
- 20 - 降級
- 25 - 嚴重

`InstancesSevere`, `InstancesDegraded`, `InstancesWarning`, `InstancesInfo`, `InstancesOk`, `InstancesPending`, `InstancesUnknown`, `InstancesNoData`

僅用於環境。這些指標表示在環境中的執行個體數目與個別運作狀態。`InstancesNoData` 表示沒有收到資料的執行個體數量。

`ApplicationRequestsTotal`, `ApplicationRequests5xx`, `ApplicationRequests4xx`, `ApplicationRequests3xx`, `ApplicationRequests2xx`

執行個體和環境。表示由執行個體或環境所完成請求的總數量，以及每個狀態碼類別完成的請求數量。

`ApplicationLatencyP10`, `ApplicationLatencyP50`, `ApplicationLatencyP75`, `ApplicationLatencyP85`, `ApplicationLatencyP90`, `ApplicationLatencyP95`, `ApplicationLatencyP99`, `ApplicationLatencyP99.9`

執行個體和環境。表示它完成 x % 請求的最快平均時間 (秒)。

`InstanceHealth`

僅執行個體。表示目前執行個體的運作狀態。執行個體運作狀態係由七種狀態其中之一來表示。在 CloudWatch 主控台中，這些狀態對應到以下值：

- 0 - 良好
- 1 - 資訊
- 5 - 不明
- 10 - 無資料
- 15 - 警告
- 20 - 降級
- 25 - 嚴重

可用指標 — Linux

`CPUIrq`, `CPUIdle`, `CPUUser`, `CPUSystem`, `CPUSoftirq`, `CPUIowait`, `CPUNice`

僅執行個體。顯示過去一分鐘內 CPU 花在每個狀態的時間百分比。

`LoadAverage1min`

僅執行個體。在過去一分鐘執行個體的平均 CPU 負載。

`RootFilesystemUtil`

僅執行個體。表示使用中的磁碟空間的百分比。

可用指標 — Windows

`CPUIdle`, `CPUUser`, `CPUPriveleged`

僅執行個體。顯示過去一分鐘內 CPU 花在每個狀態的時間百分比。

使用 Elastic Beanstalk 主控台設定 CloudWatch 指標

您可以使用 Elastic Beanstalk 主控台來設定您的環境以發佈增強型運作狀態報告指標至 CloudWatch，並使它們可用於監控圖表和警示。

在 Elastic Beanstalk 主控台中設定 CloudWatch 自訂指標

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇 Configuration (組態)。
4. 在 Monitoring (監控) 組態類別中，選擇 Edit (編輯)。
5. 在運作狀態報告下，選取要發佈到 CloudWatch 的執行個體和環境指標。若要選擇多個指標，請在選擇時同時按下 Ctrl (Ctrl) 鍵。
6. 若要儲存變更，請選擇頁面底部的儲存變更。

啟用 CloudWatch 自訂指標可將它們新增至 監控 頁面上可用的指標清單。

使用 EB CLI 設定 CloudWatch 自訂指標

您可以透過在本機保存您的環境設定、新增定義要發佈指標的項目來使用 EB CLI 設定自訂指標，然後將設定上傳到 Elastic Beanstalk。您可在建立期間或之後將儲存的設定套用到環境中。

使用 EB CLI 和儲存的組態設定 CloudWatch 自訂指標

1. 初始化您的專案資料夾 eb init。
2. 透過執行 eb create 命令來建立環境。

3. 透過執行 eb config save 命令以在本機儲存組態範本。以下範例使用 --cfg 選項以指定組態的名稱。

```
$ eb config save --cfg 01-base-state
Configuration saved at: ~/project/.elasticbeanstalk/saved_configs/01-base-
state.cfg.yml
```

4. 在文字編輯器中開啟已儲存的組態檔案。
5. 在 OptionSettings > aws:elasticbeanstalk:healthreporting:system: 下方，新增 ConfigDocument 金鑰以啟用每個您想要的 CloudWatch 指標。例如，以下的 ConfigDocument 發佈 ApplicationRequests5xx 和 ApplicationRequests4xx 環境層級的指標，以及 ApplicationRequestsTotal 在執行個體層級的指標。

```
OptionSettings:
  ...
  aws:elasticbeanstalk:healthreporting:system:
    ConfigDocument:
      CloudWatchMetrics:
        Environment:
          ApplicationRequests5xx: 60
          ApplicationRequests4xx: 60
        Instance:
          ApplicationRequestsTotal: 60
      Version: 1
    SystemType: enhanced
...
```

在此範例中，60 表示測量之間的秒數。目前，這是唯一支援的值。

> **Note**
>
> 您可將 CloudWatchMetrics 和 Rules 合併在相同的 ConfigDocument 選項設定中。Rules 會在 設定環境的增強型健康狀況規則 中詳加說明。
> 如果您之前已使用 Rules 設定增強型運作狀態規則，那麼使用 eb config save 命令擷取的組態檔案，已經擁有 ConfigDocument 金鑰及 Rules 部分。請勿刪除 — 將 CloudWatchMetrics 區段加入到相同的 ConfigDocument 選項值。

6. 儲存組態檔案並關閉文字編輯器。在這個範例中，已更新的組態檔案是以不同於下載的組態檔案名稱儲存 (02-cloudwatch-enabled.cfg.yml)。這會在檔案上傳時建立單獨的已儲存組態。您可以使用與下載的檔案相同的名稱，以覆寫現有的組態，不用建立新的名稱。

7. 使用 eb config put 命令上傳更新的組態檔案到 Elastic Beanstalk。

```
$ eb config put 02-cloudwatch-enabled
```

當您搭配已儲存的組態使用 eb config `get` 和 `put` 命令時，請不要包含副檔名。

8. 套用儲存的組態至您的執行環境。

```
$ eb config --cfg 02-cloudwatch-enabled
```

該 `--cfg` 選項會指定一個已套用至環境並已命名的組態檔案。您可以在本機儲存組態檔，或可儲存在 Elastic Beanstalk 中。如果已指定名稱的組態檔案同時存在兩個位置中，則 EB CLI 會使用本機檔案。

提供自訂指標設定文件

Amazon CloudWatch 自訂指標組態 (config) 文件是一種 JSON 文件，能以環境與執行個體的層級列出要發佈的指標。以下範例顯示能在 Linux 啟用所有可用自訂指標的設定文件。

```
{
  "CloudWatchMetrics": {
    "Environment": {
      "ApplicationLatencyP99.9": 60,
      "InstancesSevere": 60,
      "ApplicationLatencyP90": 60,
      "ApplicationLatencyP99": 60,
      "ApplicationLatencyP95": 60,
      "InstancesUnknown": 60,
      "ApplicationLatencyP85": 60,
      "InstancesInfo": 60,
      "ApplicationRequests2xx": 60,
      "InstancesDegraded": 60,
      "InstancesWarning": 60,
      "ApplicationLatencyP50": 60,
      "ApplicationRequestsTotal": 60,
      "InstancesNoData": 60,
      "InstancesPending": 60,
      "ApplicationLatencyP10": 60,
      "ApplicationRequests5xx": 60,
      "ApplicationLatencyP75": 60,
```

```
      "InstancesOk": 60,
      "ApplicationRequests3xx": 60,
      "ApplicationRequests4xx": 60
    },
    "Instance": {
      "ApplicationLatencyP99.9": 60,
      "ApplicationLatencyP90": 60,
      "ApplicationLatencyP99": 60,
      "ApplicationLatencyP95": 60,
      "ApplicationLatencyP85": 60,
      "CPUUser": 60,
      "ApplicationRequests2xx": 60,
      "CPUIdle": 60,
      "ApplicationLatencyP50": 60,
      "ApplicationRequestsTotal": 60,
      "RootFilesystemUtil": 60,
      "LoadAverage1min": 60,
      "CPUIrq": 60,
      "CPUNice": 60,
      "CPUIowait": 60,
      "ApplicationLatencyP10": 60,
      "LoadAverage5min": 60,
      "ApplicationRequests5xx": 60,
      "ApplicationLatencyP75": 60,
      "CPUSystem": 60,
      "ApplicationRequests3xx": 60,
      "ApplicationRequests4xx": 60,
      "InstanceHealth": 60,
      "CPUSoftirq": 60
    }
  },
  "Version": 1
}
```

對於 AWS CLI，您將文件當做 `Value` 金鑰值在選項設定引數中傳遞，而其本身是一種 JSON 物件。在這種情況下，您必須將嵌入文件中的引號逸出。

```
$ aws elasticbeanstalk validate-configuration-settings --application-name my-app --
environment-name my-env --option-settings '[
    {
        "Namespace": "aws:elasticbeanstalk:healthreporting:system",
        "OptionName": "ConfigDocument",
```

```
        "Value": "{\"CloudWatchMetrics\": {\"Environment\":
 {\"ApplicationLatencyP99.9\": 60,\"InstancesSevere\": 60,\"ApplicationLatencyP90\":
 60,\"ApplicationLatencyP99\": 60,\"ApplicationLatencyP95\": 60,\"InstancesUnknown
\": 60,\"ApplicationLatencyP85\": 60,\"InstancesInfo\": 60,\"ApplicationRequests2xx
\": 60,\"InstancesDegraded\": 60,\"InstancesWarning\": 60,\"ApplicationLatencyP50\":
 60,\"ApplicationRequestsTotal\": 60,\"InstancesNoData\": 60,\"InstancesPending
\": 60,\"ApplicationLatencyP10\": 60,\"ApplicationRequests5xx\": 60,
\"ApplicationLatencyP75\": 60,\"InstancesOk\": 60,\"ApplicationRequests3xx\": 60,
\"ApplicationRequests4xx\": 60},\"Instance\": {\"ApplicationLatencyP99.9\": 60,
\"ApplicationLatencyP90\": 60,\"ApplicationLatencyP99\": 60,\"ApplicationLatencyP95\":
 60,\"ApplicationLatencyP85\": 60,\"CPUUser\": 60,\"ApplicationRequests2xx\":
 60,\"CPUIdle\": 60,\"ApplicationLatencyP50\": 60,\"ApplicationRequestsTotal\":
 60,\"RootFilesystemUtil\": 60,\"LoadAverage1min\": 60,\"CPUIrq\": 60,\"CPUNice
\": 60,\"CPUIowait\": 60,\"ApplicationLatencyP10\": 60,\"LoadAverage5min\": 60,
\"ApplicationRequests5xx\": 60,\"ApplicationLatencyP75\": 60,\"CPUSystem\": 60,
\"ApplicationRequests3xx\": 60,\"ApplicationRequests4xx\": 60,\"InstanceHealth\": 60,
\"CPUSoftirq\": 60}},\"Version\": 1}"
    }
]'
```

如需 YAML.ebextensions 中的組態檔案，您可以依原狀提供 JSON 文件。

```
  option_settings:
    - namespace: aws:elasticbeanstalk:healthreporting:system
      option_name: ConfigDocument
      value: {
  "CloudWatchMetrics": {
    "Environment": {
      "ApplicationLatencyP99.9": 60,
      "InstancesSevere": 60,
      "ApplicationLatencyP90": 60,
      "ApplicationLatencyP99": 60,
      "ApplicationLatencyP95": 60,
      "InstancesUnknown": 60,
      "ApplicationLatencyP85": 60,
      "InstancesInfo": 60,
      "ApplicationRequests2xx": 60,
      "InstancesDegraded": 60,
      "InstancesWarning": 60,
      "ApplicationLatencyP50": 60,
      "ApplicationRequestsTotal": 60,
      "InstancesNoData": 60,
      "InstancesPending": 60,
```

```
      "ApplicationLatencyP10": 60,
      "ApplicationRequests5xx": 60,
      "ApplicationLatencyP75": 60,
      "InstancesOk": 60,
      "ApplicationRequests3xx": 60,
      "ApplicationRequests4xx": 60
    },
    "Instance": {
      "ApplicationLatencyP99.9": 60,
      "ApplicationLatencyP90": 60,
      "ApplicationLatencyP99": 60,
      "ApplicationLatencyP95": 60,
      "ApplicationLatencyP85": 60,
      "CPUUser": 60,
      "ApplicationRequests2xx": 60,
      "CPUIdle": 60,
      "ApplicationLatencyP50": 60,
      "ApplicationRequestsTotal": 60,
      "RootFilesystemUtil": 60,
      "LoadAverage1min": 60,
      "CPUIrq": 60,
      "CPUNice": 60,
      "CPUIowait": 60,
      "ApplicationLatencyP10": 60,
      "LoadAverage5min": 60,
      "ApplicationRequests5xx": 60,
      "ApplicationLatencyP75": 60,
      "CPUSystem": 60,
      "ApplicationRequests3xx": 60,
      "ApplicationRequests4xx": 60,
      "InstanceHealth": 60,
      "CPUSoftirq": 60
    }
  },
  "Version": 1
}
```

將增強型運作狀態報告與 Elastic Beanstalk API 搭配使用

由於 AWS Elastic Beanstalk 增強型運作狀態報告需要使用角色和解決方案堆疊，因此您必須先更新在增強型運作狀態報告發佈之前所使用的指令碼和程式碼，才能使用此報告。為了保持回溯相容性，在您使用 Elastic Beanstalk API 來建立環境時，增強型運作狀態報告預設不會啟用。

請為您的環境設定服務角色、執行個體描述檔和 Amazon CloudWatch 組態選項，以設定增強型運作狀態報告。您可以透過三種方式來進行設定：設定 `.ebextensions` 資料夾中的組態選項、使用儲存的組態，或是在 `create-environment` 呼叫的 `option-settings` 參數中直接設定這些項目。

若要使用 API、軟體開發套件或 AWS 命令列界面 (CLI)，來建立支援增強型運作狀態的環境，您必須：

- 使用適當的權限來建立服務角色和執行個體描述檔
- 使用新的平台版本建立新的環境
- 設定運作狀態系統類型、執行個體描述檔和服務角色組態選項

請使用 `aws:elasticbeanstalk:healthreporting:system`、`aws:autoscaling:launchconfiguration` 和 `aws:elasticbeanstalk:environment` 命名空間中的下列組態選項，來設定您環境的增強型運作狀態報告。

增強型運作狀態組態選項

SystemType

命名空間：`aws:elasticbeanstalk:healthreporting:system`

若要啟用增強型運作狀態報告，請設定為 **enhanced**。

IamInstanceProfile

命名空間：`aws:autoscaling:launchconfiguration`

設定為執行個體描述檔的名稱，此設定檔已設定為搭配 Elastic Beanstalk 使用。

ServiceRole

命名空間：`aws:elasticbeanstalk:environment`

設定為服務角色的名稱，此服務角色已設定為搭配 Elastic Beanstalk 使用。

ConfigDocument (選用)

命名空間：`aws:elasticbeanstalk:healthreporting:system`

JSON 文件，其中定義了要發佈到 CloudWatch 的執行個體和環境指標。例如：

```
{
```

```
  "CloudWatchMetrics":
    {
    "Environment":
      {
      "ApplicationLatencyP99.9":60,
      "InstancesSevere":60
      }
    "Instance":
      {
      "ApplicationLatencyP85":60,
      "CPUUser": 60
      }
    }
  "Version":1
}
```

> **Note**
>
> 取決於您提供給 Elastic Beanstalk 的格式，組態文件可能需要使用特別的格式，例如逸出引號。如需範例，請參閱「提供自訂指標設定文件」。

增強型運作狀態日誌格式

AWS Elastic Beanstalk 平台使用自訂 Web 伺服器日誌格式，來將 HTTP 請求的相關資訊有效率地轉送至增強式運作狀態報告系統。系統會分析日誌、識別問題，並相應地設定執行個體和環境運作狀態。若您停用環境的 Web 伺服器代理並直接從 Web 容器處理請求，您仍可將伺服器設定為透過 Elastic Beanstalk 運作狀態代理程式使用的位置和格式來輸出日誌，藉此善加運用增強型運作狀態報告。

> **Note**
>
> 此頁面上的資訊只和 Linux 類型平台相關。在 Windows Server 平台上，Elastic Beanstalk 會直接從 IIS Web 伺服器接收 HTTP 請求的資訊。如需詳細資訊，請參閱Windows Server 上 IIS 內擷取到的 Web 伺服器指標。

Web 伺服器日誌組態

Elastic Beanstalk 平台設定為輸出兩個與 HTTP 請求有關的資訊之日誌。第一個為詳細記錄的格式，並提供有關請求的詳細資訊，包括申請者的使用者代理程式資訊和人類可讀的時間戳記。

/var/log/nginx/access.log

下列範例來自執行於 Ruby Web 伺服器環境上的 nginx 代理，但格式與 Apache 的類似。

```
172.31.24.3 - - [23/Jul/2015:00:21:20 +0000] "GET / HTTP/1.1" 200 11 "-" "curl/7.22.0
 (x86_64-pc-linux-gnu) libcurl/7.22.0 OpenSSL/1.0.1 zlib/1.2.3.4 libidn/1.23
 librtmp/2.3" "177.72.242.17"
172.31.24.3 - - [23/Jul/2015:00:21:21 +0000] "GET / HTTP/1.1" 200 11 "-" "curl/7.22.0
 (x86_64-pc-linux-gnu) libcurl/7.22.0 OpenSSL/1.0.1 zlib/1.2.3.4 libidn/1.23
 librtmp/2.3" "177.72.242.17"
172.31.24.3 - - [23/Jul/2015:00:21:22 +0000] "GET / HTTP/1.1" 200 11 "-" "curl/7.22.0
 (x86_64-pc-linux-gnu) libcurl/7.22.0 OpenSSL/1.0.1 zlib/1.2.3.4 libidn/1.23
 librtmp/2.3" "177.72.242.17"
172.31.24.3 - - [23/Jul/2015:00:21:22 +0000] "GET / HTTP/1.1" 200 11 "-" "curl/7.22.0
 (x86_64-pc-linux-gnu) libcurl/7.22.0 OpenSSL/1.0.1 zlib/1.2.3.4 libidn/1.23
 librtmp/2.3" "177.72.242.17"
172.31.24.3 - - [23/Jul/2015:00:21:22 +0000] "GET / HTTP/1.1" 200 11 "-" "curl/7.22.0
 (x86_64-pc-linux-gnu) libcurl/7.22.0 OpenSSL/1.0.1 zlib/1.2.3.4 libidn/1.23
 librtmp/2.3" "177.72.242.17"
```

第二個日誌則採用簡潔格式，其中包含僅與增強型運作狀態報告相關的資訊。本日誌會輸出至名為 `healthd` 的子資料夾，且每小時輪換一次。輪換出的舊日誌會立即刪除。

/var/log/nginx/healthd/application.log.2015-07-23-00

下列範例的日誌採用機器可讀取的格式。

```
1437609879.311"/"200"0.083"0.083"177.72.242.17
1437609879.874"/"200"0.347"0.347"177.72.242.17
1437609880.006"/bad/path"404"0.001"0.001"177.72.242.17
1437609880.058"/"200"0.530"0.530"177.72.242.17
1437609880.928"/bad/path"404"0.001"0.001"177.72.242.17
```

增強型運作狀態日誌格式包含下列資訊：

- 請求的時間，格式為 Unix 時間
- 請求的路徑
- 結果的 HTTP 狀態代碼
- 請求時間
- 上游時間

- X-Forwarded-For HTTP 標頭

以 nginx 代理而言，時間為具有三個小數位的浮點秒。Apache 則用整微秒表示。

> **Note**
>
> 若您在日誌檔案中看到類似下列的警告 (其中 DATE-TIME 為日期和時間)，並且您使用自訂代理 (例如在多容器 Docker 環境中的情況)，您必須使用 .ebextension 來設定您的環境，讓 healthd 可讀取您的日誌檔案：
>
> ```
> W, [DATE-TIME #1922] WARN -- : log file "/var/log/nginx/healthd/
> application.log.DATE-TIME" does not exist
> ```
>
> 您可以從多容器 Docker 範例中的 .ebextension 開始。

/etc/nginx/conf.d/webapp_healthd.conf

下列範例為 nginx 的日誌組態，並反白顯示 healthd 日誌格式。

```
upstream my_app {
  server unix:///var/run/puma/my_app.sock;
}

log_format healthd '$msec"$uri"'
                '$status"$request_time"$upstream_response_time"'
                '$http_x_forwarded_for';

server {
  listen 80;
  server_name _ localhost; # need to listen to localhost for worker tier

  if ($time_iso8601 ~ "^(\d{4})-(\d{2})-(\d{2})T(\d{2})") {
    set $year $1;
    set $month $2;
    set $day $3;
    set $hour $4;
  }

  access_log  /var/log/nginx/access.log  main;
  access_log /var/log/nginx/healthd/application.log.$year-$month-$day-$hour healthd;
```

```
  location / {
    proxy_pass http://my_app; # match the name of upstream directive which is defined
 above
    proxy_set_header Host $host;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
  }

  location /assets {
    alias /var/app/current/public/assets;
    gzip_static on;
    gzip on;
    expires max;
    add_header Cache-Control public;
  }

  location /public {
    alias /var/app/current/public;
    gzip_static on;
    gzip on;
    expires max;
    add_header Cache-Control public;
  }
}
```

/etc/httpd/conf.d/healthd.conf

以下範例顯示 Apache 的日誌組態。

```
LogFormat "%{%s}t\"%U\"%s\"%D\"%D\"%{X-Forwarded-For}i" healthd
CustomLog "|/usr/sbin/rotatelogs /var/log/httpd/healthd/application.log.%Y-%m-%d-%H
 3600" healthd
```

產生增強型運作狀態報告的日誌

欲提供日誌給運作狀態代理程式，您必須執行下列操作：

- 如上述以正確格式輸出日誌
- 將日誌輸出至 /var/log/nginx/healthd/
- 日誌命名格式如下：application.log.$year-$month-$day-$hour
- 每小時輪換日誌一次

- 請勿截斷日誌

通知與故障診斷

本頁列出了常見問題的原因訊息範例，以及檢閱詳細資訊的連結。原因訊息會顯示於 Elastic Beanstalk 主控台的環境概觀頁面，當運作狀態問題經過幾次檢查後仍持續存在時，在事件中就會記錄原因訊息。

部署

在部署之後，Elastic Beanstalk 會監控您的環境以確保一致性。如果滾動部署失敗，則在您環境中的執行個體上所執行的應用程式，其版本可能會有不同。如果一個或多個批次的部署成功，但是在所有批次的部署完成之前失敗了，就可能會發生此種狀況。

在 5 個執行個體上有 2 個的應用程式版本不正確。預期的版本「v1」(部署 1)。

環境執行個體上的應用程式版本不正確。預期的版本「v1」(部署 1)。

預期的應用程式版本，並未在環境中的部分或全部執行個體上執行。

不正確的應用程式版本「v2」(部署 2)。預期的版本「v1」(部署 1)。

已部署到執行個體上的應用程式和預期的版本不同。如果部署失敗，則預期的版本會重設為最近成功部署的版本。在上述的範例中，第一個部署 (「v1」) 成功，但第二個部署 (「v2」) 失敗了。任何執行個體中的「v2」都會被視為運作狀態不佳。

若要解決此問題，請開始進行另一項部署作業。您可以重新部署已知運作正常的先前版本，或是設定您的環境在部署進行時略過運作狀態檢查，然後重新部署新的版本，以強制完成部署作業。

您也可以找出和終止執行應用程式錯誤版本的執行個體。Elastic Beanstalk 將會啟動執行正確版本的執行個體，來取代您所終止的任何執行個體。利用 EB CLI 運作狀態指令，來找出執行應用程式錯誤版本的執行個體。

應用程式伺服器

有 15% 的請求發生 HTTP 4xx 錯誤

傳送給 ELB 的請求，有 20% 發生 HTTP 4xx 錯誤。

傳送到執行個體或環境的 HTTP 請求，有極高的百分比因為發生了 4xx 錯誤而失敗。

400 系列的狀態碼表示使用者發出了不良的請求，例如請求不存在的頁面 (404 找不到檔案)，或使用者不具存取權限 (403 禁止存取)。少量的 404 錯誤是常見的，但是大量的 404 錯誤，則可能代表有內部

或外部的連結連到了無法使用的頁面。修正無效的內部連結，並針對無效的外部連結新增重新導向功能，即可解決這些問題。

有 5% 的請求因為發生 HTTP 5xx 錯誤而失敗

傳送給 ELB 的請求，有 3% 因為發生 HTTP 5xx 錯誤而失敗。

傳送到執行個體或環境的 HTTP 請求，有極高的百分比因為發生了 500 系列狀態代碼的錯誤而失敗。

500 系列狀態碼表示應用程式伺服器發生了內部錯誤。這些問題代表您應用程式的程式碼中存在錯誤，應該盡快找出和修正這些錯誤。

CPU 的使用率為

在執行個體上，運作狀態代理程式報告了極高百分比的 CPU 使用率，並且將執行個體的運作狀態設為警告或降級。

請擴展您的環境，來減少執行個體的負載。

工作者執行個體

在佇列中有 20 筆訊息待處理 (25 秒前)

請求新增到您工作者環境佇列中的速度，比處理請求的速度還快。請擴展您的環境，以提高處理能力。

在無效字母佇列中有 5 筆訊息 (15 秒前)

工作者的請求正重複失敗，並且新增到 the section called “無效信件佇列” 中。查看無效字母佇列中的請求，來了解其失敗原因。

其他資源

4 個作用中的執行個體低於 Auto Scaling 群組的大小最低限制 5

您環境中所執行的執行個體數量，少於針對 Auto Scaling 群組所設定的最低數量限制。

Auto Scaling 群組 (groupname) 通知已刪除或修改

針對您的 Auto Scaling 群組所設定的通知，已經在 Elastic Beanstalk 外部修改。

管理警示

您可以針對您使用 Elastic Beanstalk 主控台監控的指標建立警示。警示會協助您監控 AWS Elastic Beanstalk 環境的變更，讓您在發生問題前即可輕鬆辨識並化解。例如，在環境 CPU 使用率超過特定

閾值時，您可以設定警示通知您，確保潛在問題發生前就收到通知。如需更多詳細資訊，請參閱 搭配 Amazon CloudWatch 使用 Elastic Beanstalk。

> **Note**
>
> Elastic Beanstalk 會使用 CloudWatch 進行監控和警示，這表示您使用的任何警示都會對您的 AWS 帳戶產生 CloudWatch 費用。

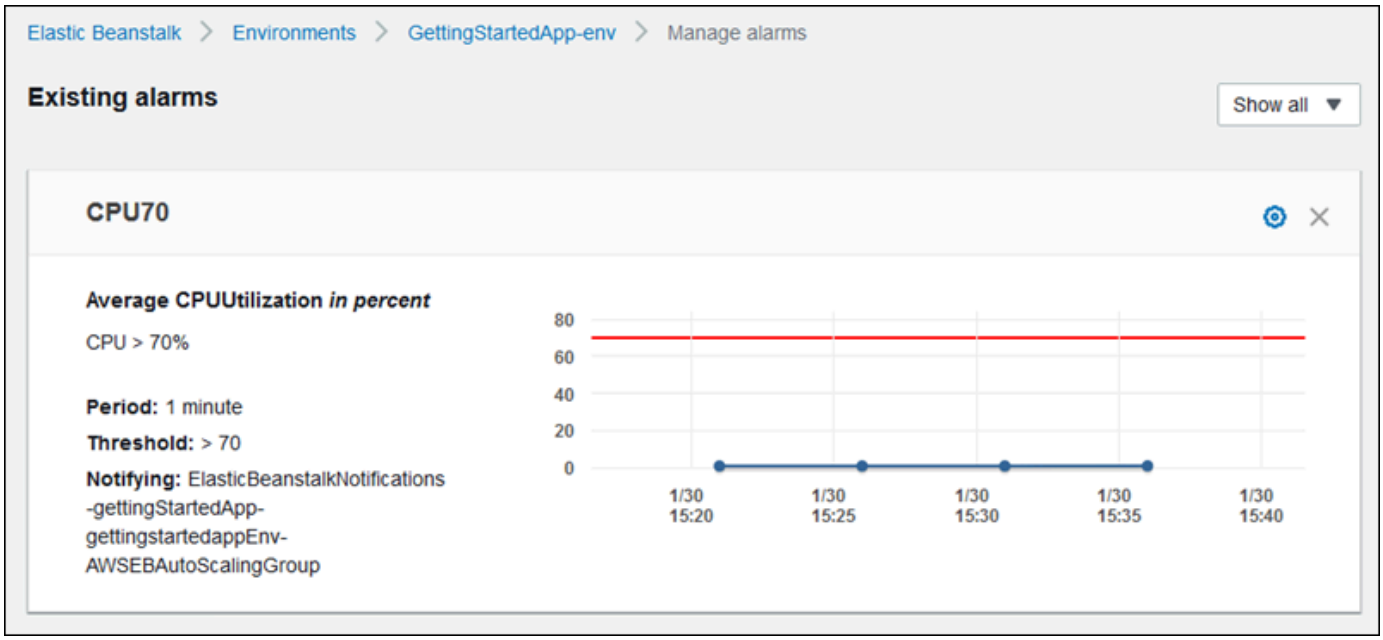
如需關於監控特定指標的詳細資訊，請參閱 基礎型運作狀態報告。

欲檢查警示狀態

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇警示。



此頁面會顯示現有警示清單。若警示處於警示狀態，則會出現此標記

(警告)。

4. 若要篩選警示，請選擇下拉式選單，然後選取篩選器。
5. 若要編輯或刪除警示，請分別選擇

(編輯)

或 (刪除)。

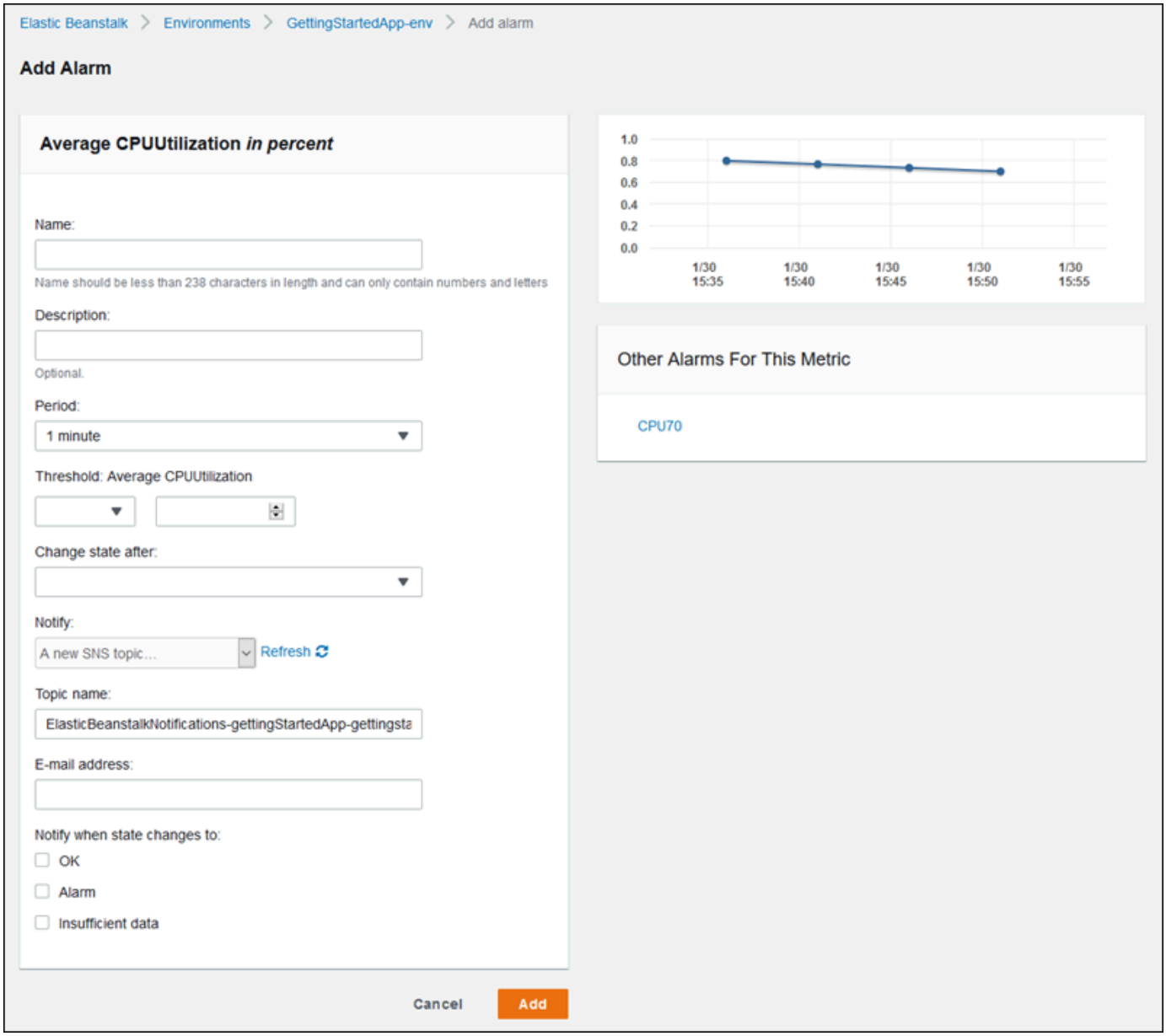
欲建立警示

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇 Monitoring (監控)。
4. 找出您要建立警示的指標，然後選擇

(警示)。此時會顯示 Add alarm (新增警示) 頁面。



5. 輸入警示的詳細資訊：

   - Name (名稱)：此警示的名稱。
   - Description (描述) (選用)：警示的簡短描述。
   - Period (期間)：兩次讀數間的時間間隔。
   - Threshold (閾值)：描述行為和指標必須超過的值以觸發警示。
   - Change state after (變更狀態時間)：超過閾值後觸發警示狀態變更的所需時間。
   - Notify (通知)：警示變更狀態時所通知的 Amazon SNS 主題。
   - 變更為下列狀態時進行通知：

- OK (正常)：指標位於定義閾值內。
- Alarm (警示)：指標超過定義閾值。
- Insufficient data (資料不足)：警示剛開始、無法使用指標，或資料不足讓指標判斷警示狀態。

6. 選擇 Add (新增)。環境更新時，環境狀態會變更為灰色。您可以在導覽窗格中選擇 Alarms (警報) 來檢視您所建立的警示。

檢視 Elastic Beanstalk 環境的變更歷史記錄

您可以使用 AWS 管理主控台檢視對 Elastic Beanstalk 環境所做的組態變更歷史記錄。Elastic Beanstalk 會從 AWS CloudTrail 中記錄的事件擷取您的變更歷史記錄，並將它們顯示在您可以輕鬆瀏覽和篩選的清單中。

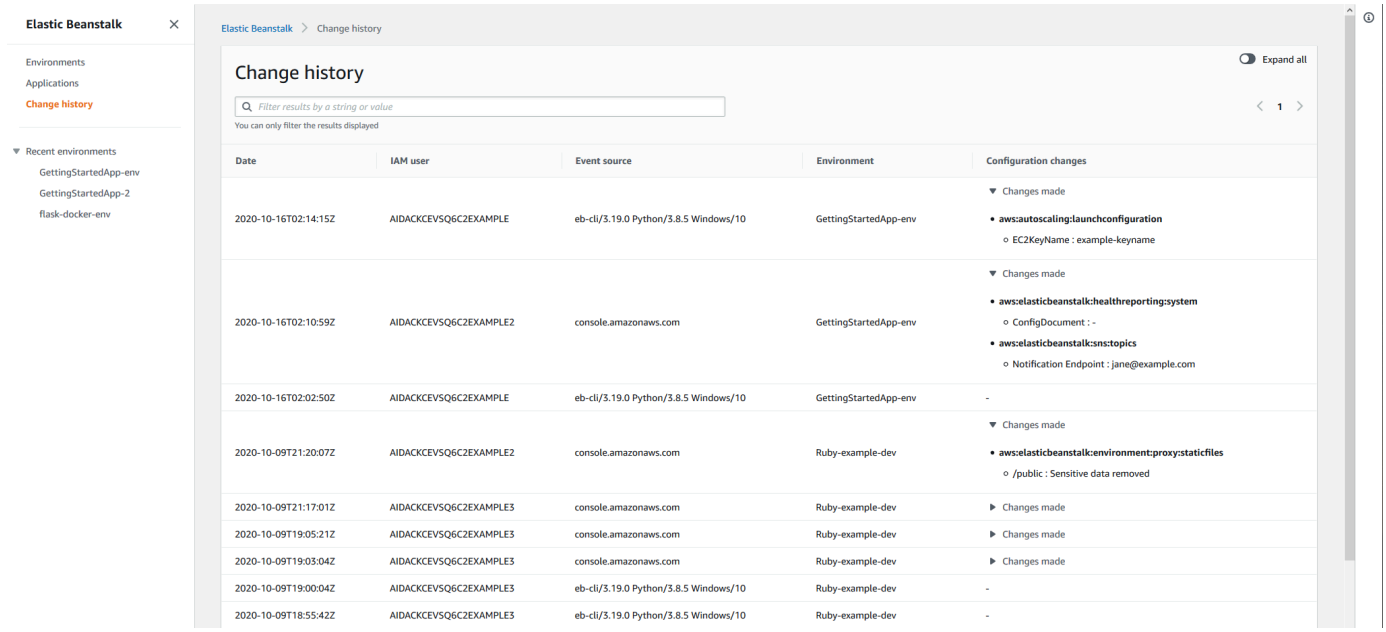
Change History (變更歷史記錄) 面板會針對您的環境所做的變更顯示下列資訊：

- 進行變更的日期和時間
- 負責所做變更的 IAM 使用者
- 用來進行變更的來源工具 (可能是 Elastic Beanstalk 命令列介面 (EB CLI) 或主控台)
- 設定的組態參數和新值

任何屬於變更一部分的敏感資料，例如受變更影響的資料庫使用者名稱，都不會顯示在面板中。

檢視變更歷史記錄

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，請選擇 Change history (變更歷史記錄)。



Change history (變更歷史記錄) 頁面會顯示對 Elastic Beanstalk 環境所做的組態變更清單。您可以選擇 < (上一頁) 或 > (下一頁)，或選擇特定頁碼，來瀏覽清單。在 Configuration changes (組態變更) 欄下，請選取箭頭圖示，在 Changes made (所做變更) 標題下的展開和收合變更清單之間切換。使用搜尋列來篩選變更歷史記錄清單中的結果。您可以輸入任何字串，以縮小顯示的變更清單。

請注意下列有關篩選顯示結果的事項：

- 搜尋篩選不區分大小寫。
- 您可以根據 Configuration changes (組態變更) 欄下的資訊篩選顯示的變更，即使由於在 Changes made (所做變更) 內被收合而無法顯示。
- 您只能篩選顯示的結果。不過，即使您選擇移至另一個頁面以顯示更多結果，篩選器仍會保持不變。篩選的結果還會附加至下一頁的結果集中。

下列範例示範如何篩選之前畫面上顯示的資料：

- 在搜尋方塊中輸入 **`GettingStartedApp-env`** 縮小結果範圍，僅包含對名稱為 GettingStartedApp-env 的環境所做的變更。
- 在搜尋方塊中輸入 **`example3`** 縮小結果範圍，僅包含由其使用者名稱包含字串 example3 的 IAM 使用者所做的變更。

- 在搜尋方塊中輸入 **`2020-10`** 縮小結果範圍，僅包含 2020 年 10 月所做的變更。將搜尋值變更為 **`2020-10-16`**，進一步篩選顯示的結果，僅包含 2020 年 10 月 16 日當天所做的變更。
- 在搜尋方塊中輸入 **`proxy:staticfiles`** 縮小結果範圍，僅包含對名稱為 aws:elasticbeanstalk:environment:proxy:staticfiles 的命名空間所做的變更。顯示的列即篩選的結果。即使在 Changes made (所做變更) 下摺疊的結果亦是如此。

檢視 Elastic Beanstalk 環境的事件資料流

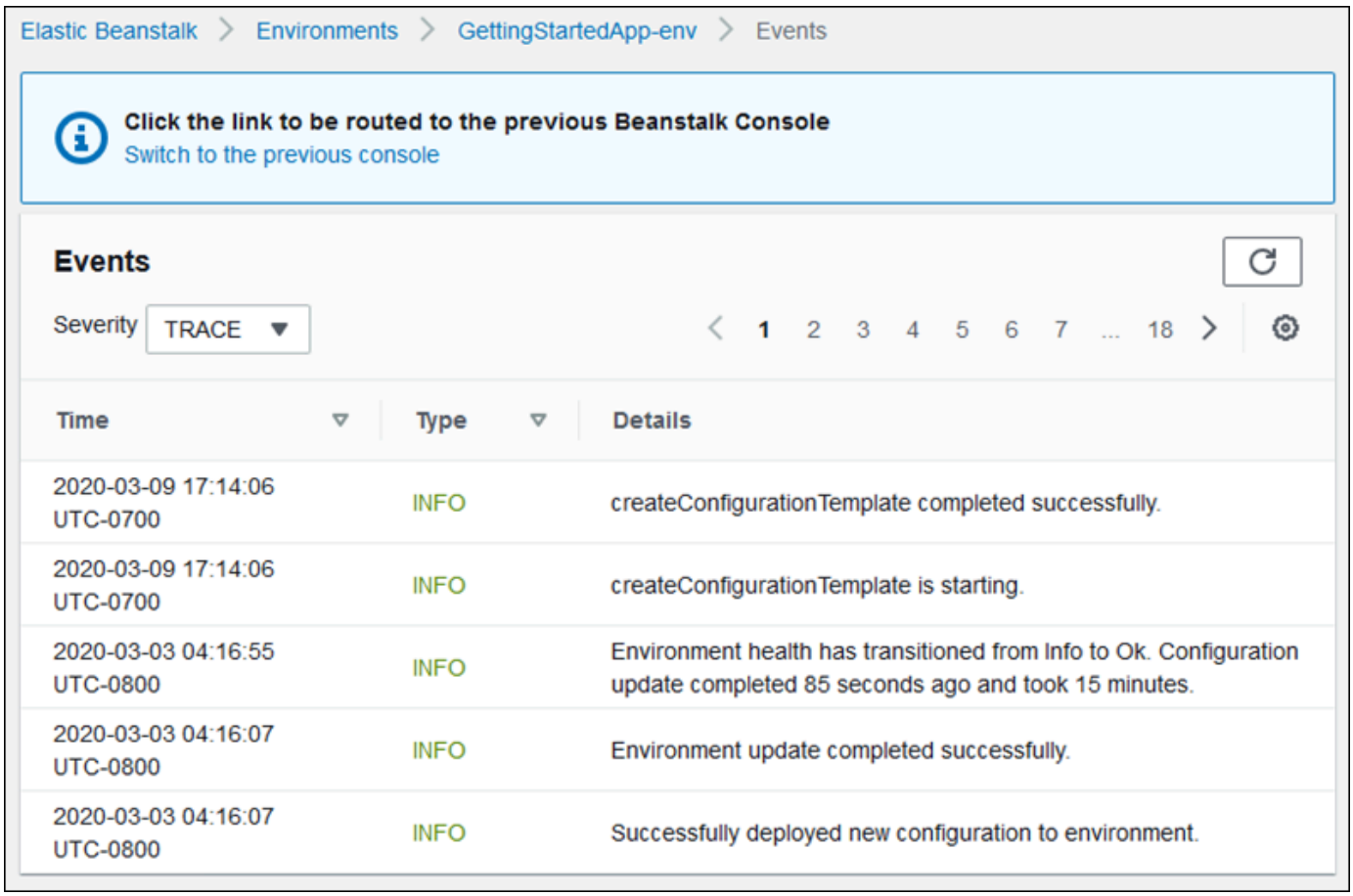
您可使用 AWS 管理主控台來存取與您的應用程式建立關聯的事件和通知。

檢視事件

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

   > **Note**
   >
   > 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇 Events (事件)。



事件頁面會顯示已為環境記錄的所有事件清單。您可以選擇 < (上一頁)、> (下一頁) 或頁碼來翻閱清單。您可使用 Severity (嚴重性) 下拉式清單來篩選要顯示的事件類型。

EB CLI 及 AWS CLI 均提供擷取事件的命令。若您正使用 EB CLI 管理環境，請使用 eb events 來列印事件清單。本命令也具備 `--follow` 選項，可持續顯示新事件，直到您按 Ctrl+C 來停止輸出。

若要使用 AWS CLI 叫出事件，請使用 `describe-events` 命令，並依名稱或 ID 指定環境：

```
$ aws elasticbeanstalk describe-events --environment-id e-gbjzqccra3
{
    "Events": [
        {
            "ApplicationName": "elastic-beanstalk-example",
            "EnvironmentName": "elasticBeanstalkExa-env",
            "Severity": "INFO",
            "RequestId": "a4c7bfd6-2043-11e5-91e2-9114455c358a",
            "Message": "Environment update completed successfully.",
            "EventDate": "2015-07-01T22:52:12.639Z"
```

```
        },
...
```

如需命令列工具的詳細資訊，請參閱工具相關文章。

列出和連線到伺服器執行個體

您可以透過 Elastic Beanstalk 主控台檢視執行 AWS Elastic Beanstalk 應用程式環境的 Amazon EC2 執行個體清單。您可使用 SSH 用戶端連接至執行個體。您可以使用遠端桌面，連接至執行 Windows 的執行個體。

有些和特定開發環境相關的備註：

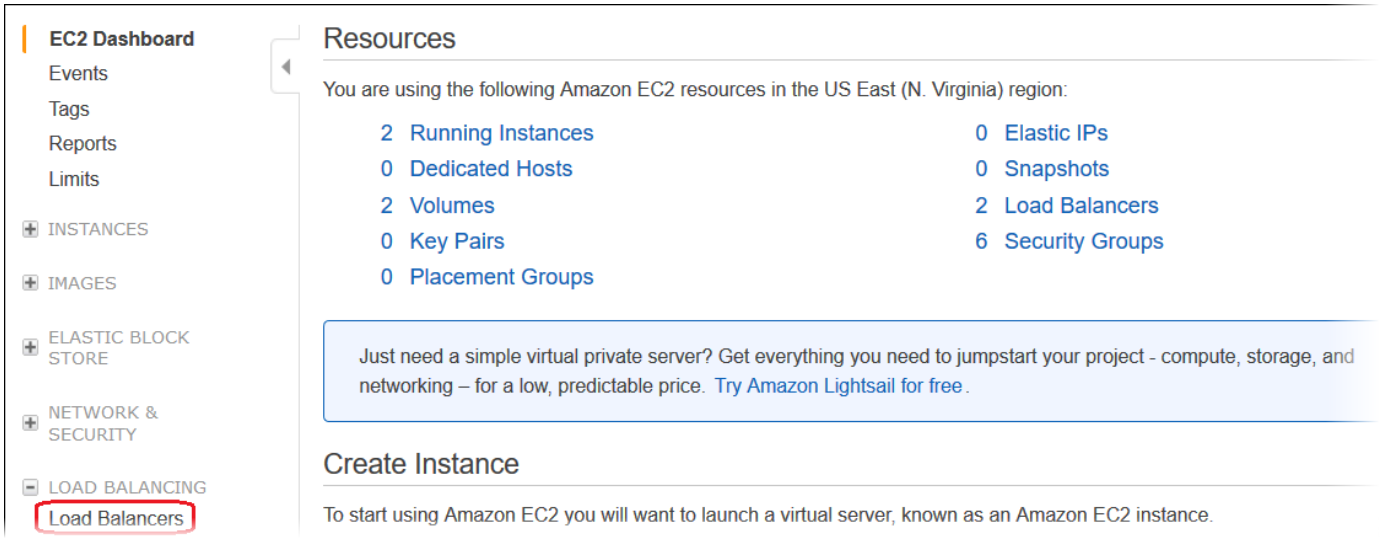
- 如需有關使用列出和連線至伺服器執行個體的詳細資訊 AWS Toolkit for Eclipse，請參閱列出和連線到伺服器執行個體。
- 如需有關使用列出和連線至伺服器執行個體的詳細資訊 AWS Toolkit for Visual Studio，請參閱列出和連線到伺服器執行個體。

> **Important**
>
> 存取 Elastic Beanstalk 佈建的 Amazon EC2 執行個體之前，您必須建立 Amazon EC2 金鑰對並設定 Elastic Beanstalk 佈建的 Amazon EC2 執行個體，以使用 Amazon EC2 金鑰對。您可以使用 AWS 管理主控台，設定您的 Amazon EC2 金鑰對。如需建立 Amazon EC2 金鑰對的詳細資訊，請參閱《Amazon EC2 入門指南》。如需有關如何將您的 Amazon EC2 執行個體設定為使用 Amazon EC2 金鑰對，請參閱 EC2 key pair (EC2 金鑰對)。
> 依預設，Elastic Beanstalk 不會啟用遠端連線至 Windows 容器內的 EC2 執行個體，除非是舊型 Windows 容器。(Elastic Beanstalk 將舊型 Windows 容器內的 EC2 執行個體設定為使用連接埠 3389 以供進行 RDP 連線。) 您可以新增規則至安全群組，授權執行個體的傳入流量，藉此啟用遠端連線至您執行 Windows 的 EC2 執行個體。我們強烈建議您結束遠端連線時，將規則移除。您可以在下次需要從遠端登入時，再次新增規則。如需詳細資訊，請參閱適用於 Microsoft Windows 的 Amazon Elastic Compute Cloud 使用者指南中的將 RDP 流量傳入規則新增至 Windows 執行個體以及連接至您的 Windows 執行個體。

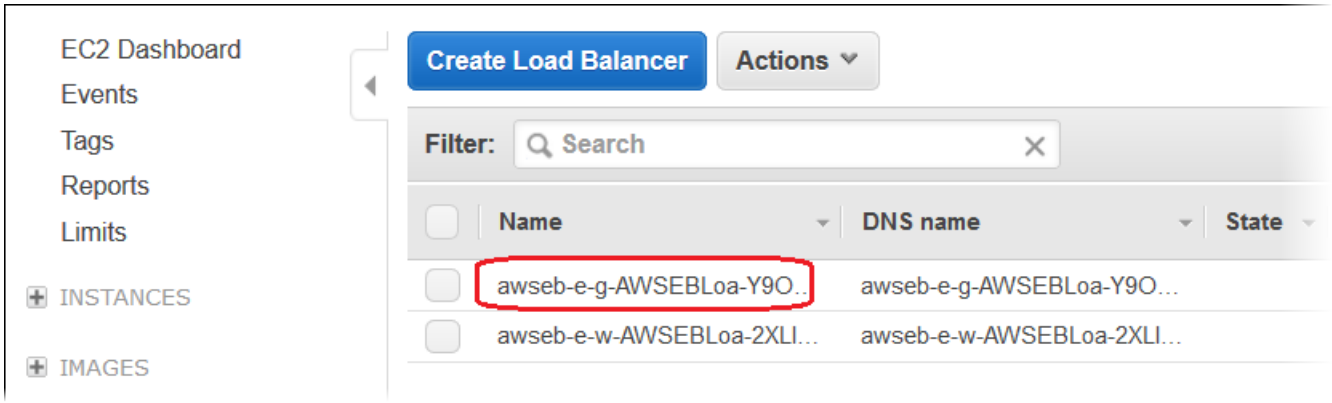
欲檢視並連接至環境的 Amazon EC2 執行個體

1. 前往 https://console.aws.amazon.com/ec2/ 開啟 Amazon EC2 主控台。

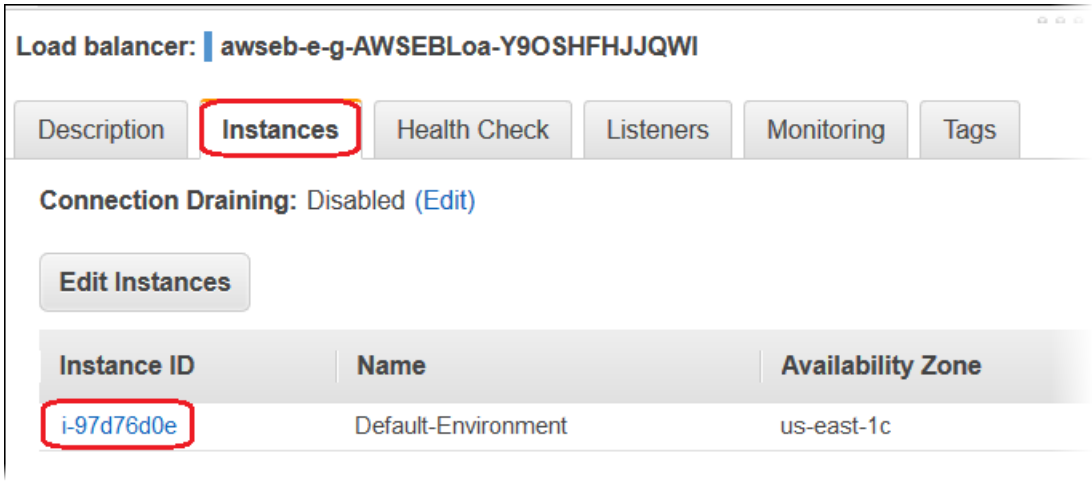
2. 在主控台的導覽窗格，選擇 Load Balancers (負載平衡器)。



3. Elastic Beanstalk 建立的負載平衡器名稱會出現 awseb。找到您環境的負載平衡器，然後按一下。

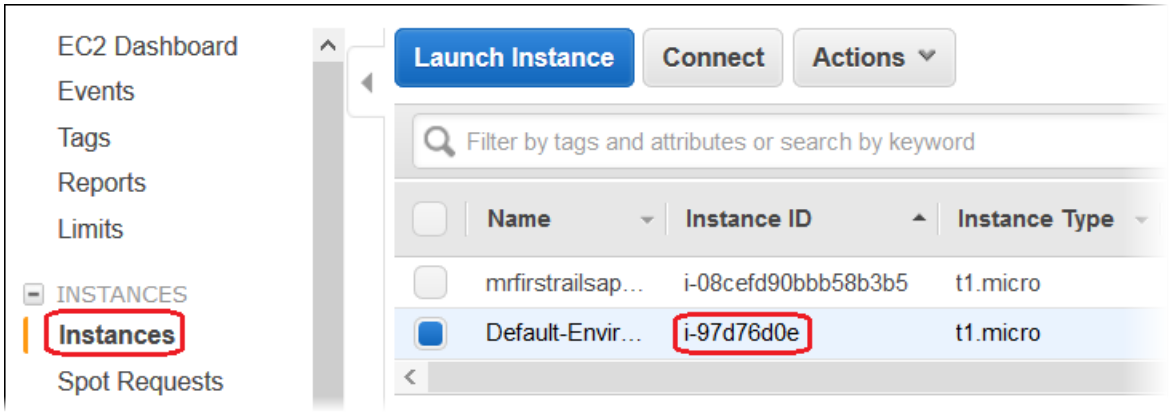


4. 選擇主控台底部窗格的 Instances (執行個體) 索引標籤。



將顯示您 Elastic Beanstalk 環境的負載平衡器所使用的執行個體清單。請記下您欲連接的執行個體 ID。

5. 在 Amazon EC2 主控台導覽窗格中，選擇 Instances (執行個體)，在清單中尋找您的執行個體 ID。



6. 在執行於您環境負載平衡器的 Amazon EC2 執行個體 ID 按一下滑鼠右鍵，然後選擇內容選單的 Connect (連接)。
7. 請於 Description (描述) 索引標籤記下該執行個體的公有 DNS 地址。
8. 使用您選擇的安全殼層用戶端 Connect 至執行 Linux 的執行個體，然後輸入 ssh-i。of-the-instance

如需連接至亞馬遜 EC2 Linux 執行個體的詳細資訊，請參閱 Amazon EC2 使用者指南中的開始使用 Amazon EC2 Linux 執行個體。

如果您的 Elastic Beanstalk 環境使用 Windows 伺服器平台上的 .NET，請參閱 Amazon EC2 使用者指南中的亞 Amazon EC2 視窗執行個體入門。

在 Elastic Beanstalk 環境中檢視 Amazon EC2 執行個體的日誌

Elastic Beanstalk 環境的 Amazon EC2 執行個體會產生日誌，您可加以檢視針對應用程式或組態檔案進行問題疑難排解。由 Web 伺服器、應用程式伺服器、Elastic Beanstalk 平台指令碼和 AWS CloudFormation 建立的日誌會儲存在個別本機執行個體上。您可以使用環境管理主控台或 EB CLI 輕鬆擷取他們。您也可以將環境設定為即時將日誌串流至 Amazon CloudWatch Logs。

結尾日誌是最常使用日誌檔案 (Elastic Beanstalk 操作日誌及來自 web 伺服器或應用程式伺服器的日誌) 的最後 100 行。當您透過環境主控台或 eb logs 請求結尾日誌時，環境中的一個執行個體會將最近的日誌項目串連為單一文字檔案，並上傳至 Amazon S3。

套件日誌為更多日誌檔案的完整日誌，包括來自 yum 和 cron 的日誌以及數個來自 AWS CloudFormation 的日誌。當您請求套件日誌時，環境中的執行個體會將完整日誌檔案封裝為 ZIP 封存檔，並上傳至 Amazon S3。

> **Note**
>
> Elastic Beanstalk Windows Server 平台不支援套件日誌。

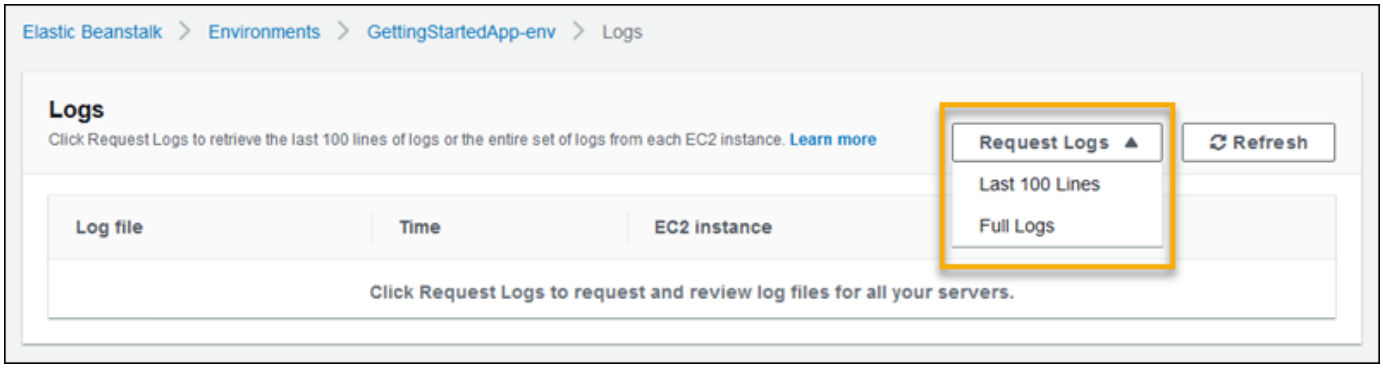
若要將輪換日誌上傳至 Amazon S3，環境中的執行個體必須具備執行個體描述檔，其中包含寫入至 Elastic Beanstalk Amazon S3 儲存貯體的許可。當您在 Elastic Beanstalk 主控台首次啟動環境時，Elastic Beanstalk 提示您建立的執行個體描述檔預設即包含這些許可。

擷取執行個體日誌

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

   > **Note**
   >
   > 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇 Logs (日誌)。
4. 選擇 Request Logs (請求日誌)，然後選擇要擷取的日誌類型。欲取得結尾日誌，選擇 Last 100 Lines (最後 100 行)。欲取得套件日誌，選擇 Full Logs (完整日誌)。



5. 當 Elastic Beanstalk 完成擷取記錄檔時，請選擇 Download (下載)。

Elastic Beanstalk 會將結尾和套件日誌儲存在 Amazon S3 儲存貯體中，並產生一個預先簽章的 Amazon S3 URL，讓您用來存取日誌。Elastic Beanstalk 會在 15 分鐘後從 Amazon S3 刪除檔案。

> **Warning**
>
> 擁有預先簽章之 Amazon S3 URL 的任何人皆可在檔案被刪除之前存取檔案。請務必僅將此 URL 提供給可信任者。

> **Note**
>
> 您的使用者政策必須具有 `s3:DeleteObject` 許可。Elastic Beanstalk 會使用您的使用者許可從 Amazon S3 刪除日誌。

若要保留日誌，您可以將環境設定為在日誌輪換後自動發佈至 Amazon S3。若要啟用 Amazon S3 的日誌輪換，請遵循 設定執行個體日誌檢視 中的程序。環境中的執行個體會每小時嘗試上傳已輪換的日誌。

若您的應用程式並未在做為您環境平台預設組態一部分的位置內產生日誌，您可以使用組態檔案 (`.ebextensions`) 來延伸預設組態。您可將應用程式的日誌檔案，新增至結尾日誌、套件日誌或日誌輪換。

若要進行即時日誌串流和長期儲存，請設定您的環境以串流日誌到 Amazon CloudWatch Logs。

章節

- Amazon EC2 執行個體上的日誌位置
- Amazon S3 中的日誌位置
- Linux 上的日誌輪換設定
- 擴展預設日誌任務組態
- 將日誌檔案串流至 Amazon CloudWatch Logs

Amazon EC2 執行個體上的日誌位置

日誌會存放於環境中 Amazon EC2 執行個體上的標準位置。Elastic Beanstalk 會產生下列日誌。

Amazon Linux 2

- `/var/log/eb-engine.log`

Amazon Linux AMI (AL1)

> **Note**
>
> 2022 年 7 月 18 日，Elastic Beanstalk 已將所有以 Amazon Linux AMI (AL1) 為基礎的平台分支狀態設為已淘汰。如需有關遷移至完全支援的目前 Amazon Linux 2023 平台分支的詳細資訊，請參閱 將您的 Elastic Beanstalk Linux 應用程式遷移到 Amazon Linux 2023 或 Amazon Linux 2。

- `/var/log/eb-activity.log`
- `/var/log/eb-commandprocessor.log`

Windows Server

- `C:\Program Files\Amazon\ElasticBeanstalk\logs\`
- `C:\cfn\log\cfn-init.log`

這些日誌會納入部署活動的訊息，包括與組態檔案 (`.ebextensions`) 相關的訊息。

各個應用程式和 Web 伺服器將日誌存放於自己的資料夾：

- Apache – `/var/log/httpd/`
- IIS – `C:\inetpub\wwwroot\`
- Node.js – `/var/log/nodejs/`
- nginx – `/var/log/nginx/`
- Passenger – `/var/app/support/logs/`
- Puma – `/var/log/puma/`
- Python – `/opt/python/log/`
- Tomcat – `/var/log/tomcat/`

Amazon S3 中的日誌位置

當您自環境請求結尾或套件日誌，或當執行個體上傳輪換日誌時，這些日誌會存放於 Amazon S3 內的 Elastic Beanstalk 儲存貯體。Elastic Beanstalk 會在您建立環境的各個 AWS 區域中建立名為

elasticbeanstalk-*region*-*account-id* 的儲存貯體。在此儲存貯體內，日誌會存放於此路徑下：resources/environments/logs/*logtype*/*environment-id*/*instance-id*。

例如，在帳戶 123456789012 中 AWS 區域 us-west-2 內 Elastic Beanstalk 環境 e-mpcwnwheky 中執行個體 i-0a1fd158 的日誌，會存放在下列位置：

- 結尾日誌 –

  s3://elasticbeanstalk-us-west-2-123456789012/resources/environments/logs/tail/e-mpcwnwheky/i-0a1fd158
- 套件日誌 –

  s3://elasticbeanstalk-us-west-2-123456789012/resources/environments/logs/bundle/e-mpcwnwheky/i-0a1fd158
- 輪換日誌 –

  s3://elasticbeanstalk-us-west-2-123456789012/resources/environments/logs/publish/e-mpcwnwheky/i-0a1fd158

> **Note**
>
> 您可以在環境管理主控台中找到您的環境 ID。

在結尾和套件日誌產生後的 15 分鐘，Elastic Beanstalk 會自動將其自 Amazon S3 刪除。輪換日誌會保留，直到您將其刪除或移至 S3 Glacier。

Linux 上的日誌輪換設定

在 Linux 平台上，Elastic Beanstalk 會使用 logrotate 來定期輪換日誌。若經過設定，日誌於本機輪換後，將由日誌輪換任務選取並上傳至 Amazon S3。於本機輪換的日誌，預設不會顯示於結尾或套件日誌。

您可於 /etc/logrotate.elasticbeanstalk.hourly/ 找到 logrotate 的 Elastic Beanstalk 組態檔案。這些輪換設定為平台特定，未來的平台版本可能會加以變更。如需有關可用設定和範例組態的詳細資訊，請執行 man logrotate。

定時守護作業 (cron job) 會於 /etc/cron.hourly/ 中叫用組態檔案。如需關於 cron的詳細資訊，請執行 man cron。

擴展預設日誌任務組態

Elastic Beanstalk 會使用 Amazon EC2 執行個體上 `/opt/elasticbeanstalk/tasks` (Linux) 或 `C:\Program Files\Amazon\ElasticBeanstalk\config` (Windows Server) 的子資料夾內的檔案，設定結尾日誌、套件日誌和日誌輪換的任務。

在 Amazon Linux 上：

- 結尾日誌 –

  ```
  /opt/elasticbeanstalk/tasks/taillogs.d/
  ```
- 套件日誌 –

  ```
  /opt/elasticbeanstalk/tasks/bundlelogs.d/
  ```
- 輪換日誌 –

  ```
  /opt/elasticbeanstalk/tasks/publishlogs.d/
  ```

在 Windows Server 上：

- 結尾日誌 –

  ```
  c:\Program Files\Amazon\ElasticBeanstalk\config\taillogs.d\
  ```
- 輪換日誌 –

  ```
  c:\Program Files\Amazon\ElasticBeanstalk\config\publogs.d\
  ```

例如，Linux 上的檔案 `eb-activity.conf` 會將兩個日誌檔新增至結尾日誌任務。

`/opt/elasticbeanstalk/tasks/taillogs.d/eb-activity.conf`

```
/var/log/eb-commandprocessor.log
/var/log/eb-activity.log
```

您可使用環境資訊檔案 (`.ebextensions`) 將自己的 `.conf` 檔案新增至這些資料夾。`.conf` 檔案會列出您應用程式特定的日誌檔案，Elastic Beanstalk 會將其新增至日誌檔案任務。

使用 files 區段將組態檔案新增至您欲修改的任務。例如，下列組態文字會將日誌組態檔案加入至您環境中的每個執行個體。此日誌組態檔案 cloud-init.conf 新增 /var/log/cloud-init.log 到結尾日誌。

```
files:
  "/opt/elasticbeanstalk/tasks/taillogs.d/cloud-init.conf" :
    mode: "000755"
    owner: root
    group: root
    content: |
      /var/log/cloud-init.log
```

將此文字新增至副檔名為 .config 的檔案，再將此檔案新增至名為 .ebextensions 的資料夾下原始碼套件中。

```
~/workspace/my-app
|-- .ebextensions
|   `-- tail-logs.config
|-- index.php
`-- styles.css
```

在 Linux 平台上，您也可以在日誌任務組態中使用萬用字元。此組態檔案會自應用程式根目錄的 .log 資料夾，將副檔名為 log 的所有檔案新增至套件日誌。

```
files:
  "/opt/elasticbeanstalk/tasks/bundlelogs.d/applogs.conf" :
    mode: "000755"
    owner: root
    group: root
    content: |
      /var/app/current/log/*.log
```

日誌任務組態在 Windows 平台上不支援萬用字元。

> **Note**
>
> 若要協助熟悉記錄自訂程序，您可以使用 EB CLI 部署範例應用程式。為此，EB CLI 會建立包含具有範例組態之 .ebextentions 子目錄的本機應用程式目錄。您也可以使用範例應用程式的日誌檔案來探索本主題所述的日誌擷取功能。如需使用 EB CLI 建立範例應用程式的詳細資訊，請參閱 EB CLI 基本知識。

如需使用組態檔案的詳細資訊，請參閱使用組態檔案 (.ebextensions) 來進行進階的環境自訂。

您可使用組態檔案來擴展日誌輪換，如同擴展結尾日誌和套件日誌。每當 Elastic Beanstalk 輪換自己的日誌並將其上傳至 Amazon S3 時，也會輪換並上傳您的其他日誌。日誌輪換擴展的行為依平台作業系統而異。下列小節描述兩種情況。

於 Linux 上擴展日誌輪換

如Linux 上的日誌輪換設定中所說明，在 Linux 平台上，Elastic Beanstalk 會使用 `logrotate` 來輪換日誌。當您將應用程式的日誌檔案設定為日誌輪換，該應用程式不需要建立日誌檔案的副本。Elastic Beanstalk 會設定 `logrotate`，於每次輪換複製您的應用程式日誌檔案。因此，應用程式在主動寫入日誌期間之外，必須讓日誌檔案保持解鎖狀態。

於 Windows Server 上擴展日誌輪換

在 Windows Server 上，當您將應用程式的日誌檔案設定為日誌輪換，該應用程式必須定期輪換日誌檔案。Elastic Beanstalk 會尋找檔名開頭與您所設定的模式相符的檔案，將其選取以上傳至 Amazon S3。此外，Elastic Beanstalk 會忽略檔名的句點，將句點前的名稱視為基本日誌檔案名稱。

Elastic Beanstalk 會上傳基本日誌檔案的所有版本 (除了最新版本)，因其認為最新版本為作用中的應用程式日誌檔案，可能會被鎖定。因此，您的應用程式於輪換之間可鎖定作用中的日誌檔案。

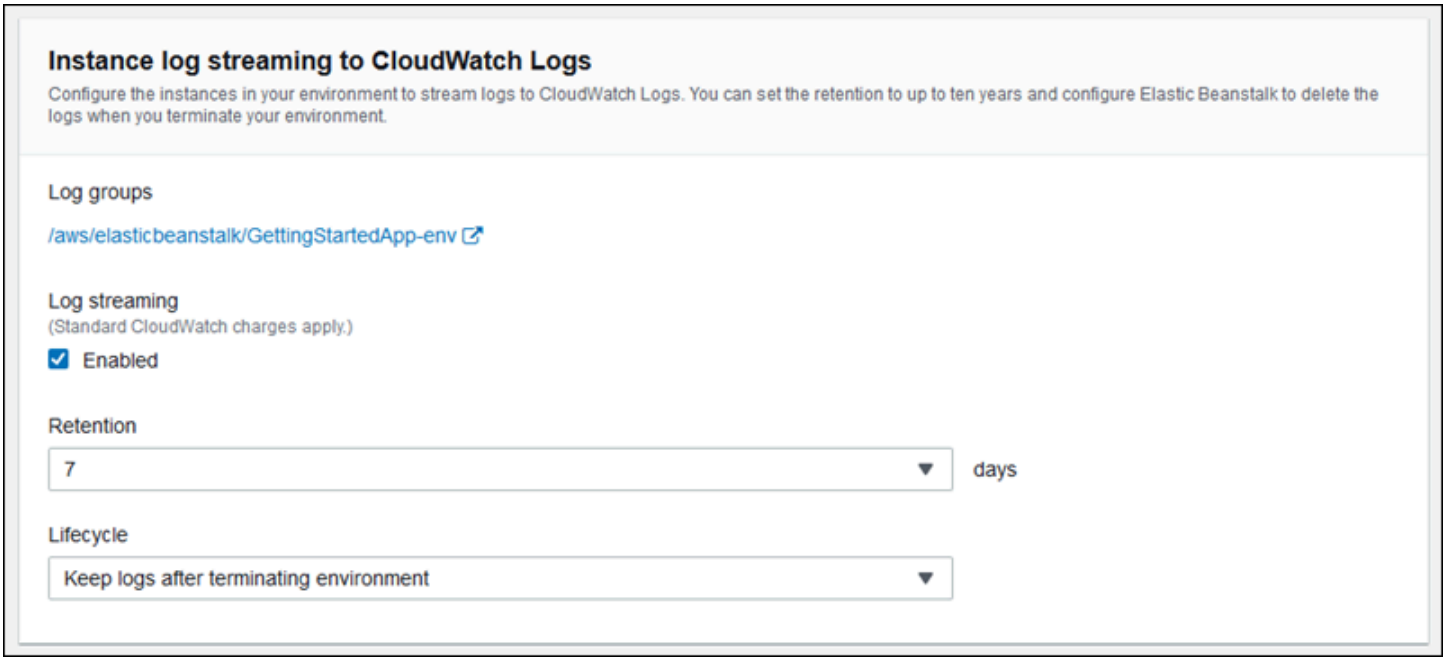
例如，您的應用程式寫入名為 `my_log.log` 的日誌檔案，而且您在 `.conf` 檔案中指定了此名稱。應用程式會定期輪換檔案。在 Elastic Beanstalk 輪換循環時，會於日誌檔案資料夾找出下列檔案：`my_log.log`、`my_log.0800.log`、`my_log.0830.log`。Elastic Beanstalk 會將它們視為基本名稱 `my_log` 的版本。檔案 `my_log.log` 的修改時間最晚，因此 Elastic Beanstalk 僅會上傳其他兩個檔案：`my_log.0800.log` 和 `my_log.0830.log`。

將日誌檔案串流至 Amazon CloudWatch Logs

您可將環境設定為在 Elastic Beanstalk 主控台中串流日誌至 Amazon CloudWatch Logs，或使用組態選項來進行此作業。透過 CloudWatch Logs，您環境的各個執行個體會將日誌串流至日誌群組，您可將日誌群組設定為保留數週或數年，甚至保留到環境終止之後。

所串流的日誌組會依環境而異，不過一律會包含 `eb-engine.log`，以及於應用程式前方執行的 nginx 或 Apache 代理伺服器存取日誌。

您可以在建立環境期間或為現有環境，在 Elastic Beanstalk 主控台設定日誌串流。在以下範例中，日誌儲存會長達 7 天，即使環境終止也一樣。



下列組態檔案的日誌串流保留期間為 180 天，即使環境終止也是如此。

Example .ebextensions/log-streaming.config

```
option_settings:
  aws:elasticbeanstalk:cloudwatch:logs:
    StreamLogs: true
    DeleteOnTerminate: false
    RetentionInDays: 180
```

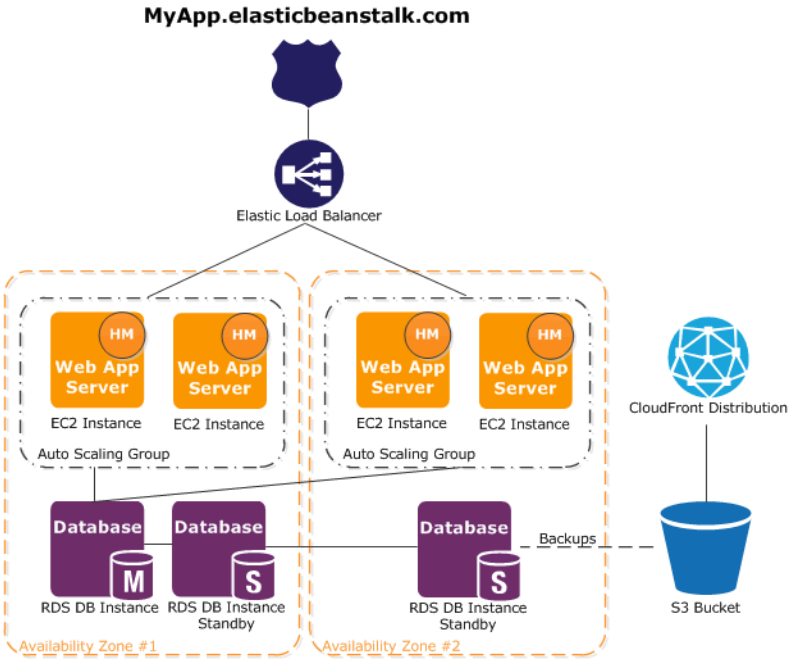
將 Elastic Beanstalk 與其他 AWS 服務一起使用

為了實作您應用程式的環境，Elastic Beanstalk 管理其他 AWS 服務的資源或使用它們的功能。此外，Elastic Beanstalk 與不會直接使用於您的環境中一部分的 AWS 服務整合。此章節的主題描述您可以使用這些額外的服務於您的 Elastic Beanstalk 應用程式的許多方法。

主題

架構概觀

以下圖表說明使用其他 AWS 產品，例如 Amazon CloudFront、Amazon Simple Storage Service (Amazon S3) 和 Amazon Relational Database Service (Amazon RDS) 的跨多個可用區域 Elastic Beanstalk 範例架構。



為了規劃容錯能力，建議您預備 N+1 個 Amazon EC2 執行個體，並將執行個體散佈至多個可用區域。假如一個可用區域故障，執行於其他可用區域的 Amazon EC2 執行個體仍將正常運作。您可以調整 Amazon EC2 Auto Scaling 以允許執行個體的最低數量以及多個可用區域。如需如何執行此動作的詳細資訊，請參閱適用於您 Elastic Beanstalk 環境的 Auto Scaling 群組。如需如何建置容錯應用程式的詳細資訊，請移至在 AWS 上建置容錯應用程式。

下面的章節會詳細討論與 Amazon CloudFront、Amazon CloudWatch、Amazon DynamoDB Amazon ElastiCache、Amazon RDS、Amazon Route 53、Amazon Simple Storage Service、Amazon VPC 和 IAM 的整合。

搭配 Amazon CloudFront 使用 Elastic Beanstalk

Amazon CloudFront 是一種 Web 服務，可更快速分發靜態與動態內容供最終使用者閱覽，例如 .html、.css、.php、圖片及媒體檔案。CloudFront 透過全球節點網路提供您的內容。當最終使用者請求您透過 CloudFront 提供的內容時，會自動將該使用者路由到可提供最低延遲的節點，因此能以最佳的效能發佈內容。如果該內容已存在於該節點，CloudFront 會立即提供該內容。如果內容不存在於該節點，CloudFront 會從您指定為最終內容版本來源的 Amazon S3 儲存貯體或 HTTP 伺服器 (例如 Web 伺服器) 取回該內容。

您建立並部署 Elastic Beanstalk 應用程式之後，可以向 CloudFront 註冊並開始用 CloudFront 將您的內容發佈出去。若要進一步了解 CloudFront，請參閱 Amazon CloudFront 開發人員指南。

使用 AWS CloudTrail 記錄 Elastic Beanstalk API 呼叫

Elastic Beanstalk 整合了 AWS CloudTrail，後者是一項服務，可提供由使用者、角色或 AWS 服務在 Elastic Beanstalk 中所採取之動作的記錄。CloudTrail 會將所有針對 Elastic Beanstalk 的 API 呼叫擷取為事件，包括來自 Elastic Beanstalk 主控台、來自 EB CLI，以及從程式碼到 Elastic Beanstalk API 的呼叫。如果您建立線索，就可以將 CloudTrail 事件持續交付至 Amazon S3 儲存貯體，包括 Elastic Beanstalk 的事件。即使您未設定線索，依然可以透過 CloudTrail 主控台中的 Event history (事件歷史記錄) 檢視最新事件。您可以利用 CloudTrail 所收集的資訊來判斷向 Elastic Beanstalk 發出的請求，以及發出請求的 IP 地址、人員、時間和其他詳細資訊。

若要進一步了解 CloudTrail，請參閱 AWS CloudTrail 使用者指南。

CloudTrail 中的 Elastic Beanstalk 資訊

當您建立帳戶時，系統即會在 AWS 帳戶中啟用 CloudTrail。此外，Elastic Beanstalk 發生活動時，系統便會將該活動記錄至 CloudTrail 事件，並將其他 AWS 服務事件記錄到 Event history (事件歷史記錄) 中。您可以檢視、搜尋和下載 AWS 帳戶的最新事件。如需詳細資訊，請參閱使用 CloudTrail 事件歷史記錄檢視事件。

如需 AWS 帳戶中正在進行事件的記錄 (包含 Elastic Beanstalk 的事件)，請建立線索。線索能讓 CloudTrail 將日誌檔案交付至 Amazon S3 儲存貯體。根據預設，當您在主控台建立追蹤記錄時，追蹤記錄會套用到所有區域。線索會記錄來自 AWS 分割區中所有區域的事件，然後將所有日誌檔案交付到您指定的 Amazon S3 儲存貯體。除此之外，您還能設定其他 AWS 服務，以進一步分析和處理 CloudTrail 日誌收集到的事件資料。如需詳細資訊，請參閱：

- 建立追蹤的概觀
- CloudTrail 支援的服務和整合
- 設定 CloudTrail 的 Amazon SNS 通知
- 從多個區域接收 CloudTrail 日誌檔案，以及從多個帳戶接收 CloudTrail 日誌檔案

CloudTrail 會記錄所有 Elastic Beanstalk 動作，並記錄在 AWS Elastic Beanstalk API 參考中。例如，對 `DescribeApplications`、`UpdateEnvironment` 以及 `ListTagsForResource` 動作發出的呼叫會在 CloudTrail 日誌檔案中產生項目。

每一筆事件或記錄項目都會包含產生請求者的資訊。身分資訊可協助您判斷下列事項：

- 該請求是否使用根或 IAM 使用者登入資料提出。

- 提出該要求時，是否使用了特定角色或聯合身分使用者的暫時安全登入資料.
- 該請求是否由另一項 AWS 服務提出

如需詳細資訊，請參閱 CloudTrail userIdentity 元素。

了解 Elastic Beanstalk 日誌檔項目

追蹤是一種組態，能讓事件以日誌檔案的形式交付至您指定的 Amazon S3 儲存貯體。CloudTrail 日誌檔案包含一個或多個日誌項目。一個事件為任何來源提出的單一請求，並包含請求動作、請求的日期和時間、請求參數等資訊。CloudTrail 日誌檔案並非依公有 API 呼叫追蹤記錄的堆疊排序，因此不會以任何特定順序出現。

下列範例為 CloudTrail 日誌項目，此項目所示的是 `sample-app` 應用程式中的 `sample-env` 環境下，名為 `intern` 的 IAM 使用者所呼叫的 `UpdateEnvironment` 動作。

```
{
  "Records": [{
    "eventVersion": "1.05",
    "userIdentity": {
      "type": "IAMUser",
      "principalId": "AIXDAYQEXAMPLEUMLYNGL",
      "arn": "arn:aws:iam::123456789012:user/intern",
      "accountId": "123456789012",
      "accessKeyId": "ASXIAGXEXAMPLEQULKNXV",
      "userName": "intern",
      "sessionContext": {
        "attributes": {
          "mfaAuthenticated": "false",
          "creationDate": "2016-04-22T00:23:24Z"
        }
      },
      "invokedBy": "signin.amazonaws.com"
    },
    "eventTime": "2016-04-22T00:24:14Z",
    "eventSource": "elasticbeanstalk.amazonaws.com",
    "eventName": "UpdateEnvironment",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "255.255.255.54",
    "userAgent": "signin.amazonaws.com",
    "requestParameters": {
      "applicationName": "sample-app",
```

```
      "environmentName": "sample-env",
      "optionSettings": []
    },
    "responseElements": null,
    "requestID": "84ae9ecf-0280-17ce-8612-705c7b132321",
    "eventID": "e48b6a08-c6be-4a22-99e1-c53139cbfb18",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
  }]
}
```

搭配 Amazon CloudWatch 使用 Elastic Beanstalk

Amazon CloudWatch 可讓您監控、管理並發佈各種指標，並根據這些指標的資料來設定警示動作。Amazon CloudWatch 監控可讓您收集、分析並檢視系統和應用程式指標，讓您可以更快速、更放心制定營運和商業決策。

您可以使用 Amazon CloudWatch 來收集有關 Amazon Web Services (AWS) 資源的指標，例如 Amazon EC2 執行個體的效能。您亦可直接向 Amazon CloudWatch 發佈自己的指標。Amazon CloudWatch 警示可讓您根據您定義的規則，傳送通知或自動變更您正監控的資源，協助您更輕鬆實作決策。例如，您可以建立警示，代您啟動 Amazon EC2 Auto Scaling 和 Amazon Simple Notification Service (Amazon SNS) 動作。

Elastic Beanstalk 會自動使用 Amazon CloudWatch 協助您監控應用程式和環境狀態。您可以導覽至 Amazon CloudWatch 主控台，查看您的儀表板，取得所有資源及警示的概觀。您亦可選擇檢視更多指標或新增自訂指標。

如需 Amazon CloudWatch 的詳細資訊，請前往《Amazon CloudWatch 開發人員指南》。如需如何搭配 Elastic Beanstalk 使用 Amazon CloudWatch 的範例，請參閱the section called “範例：使用自訂 Amazon CloudWatch 指標”。

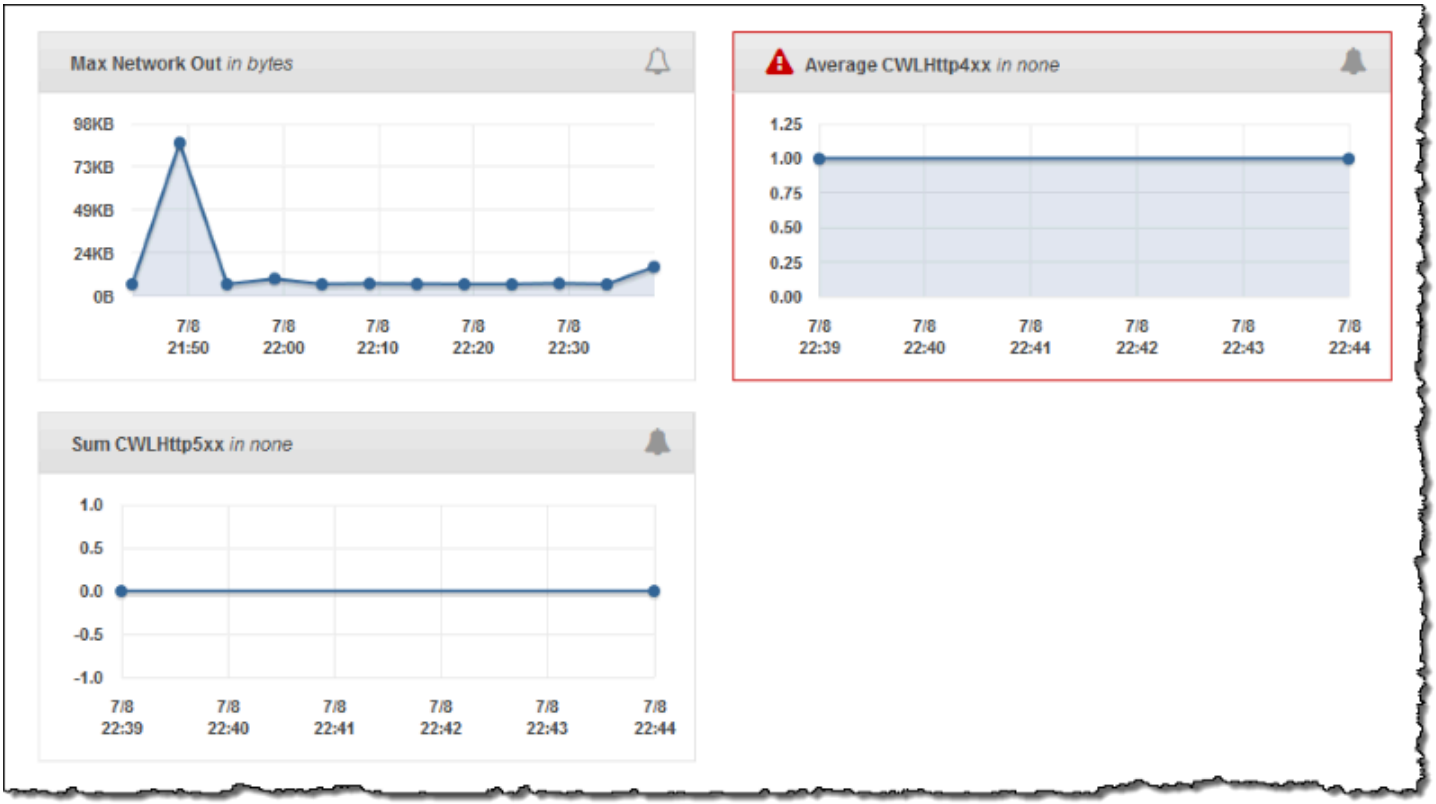
搭配 Amazon CloudWatch Logs 使用 Elastic Beanstalk

您可以使用 CloudWatch Logs 監控和封存 Elastic Beanstalk 應用程式、系統，以及來自您環境的 Amazon EC2 執行個體的自訂日誌檔。您也可以設定警示，讓您更輕鬆地回應指標篩選條件所擷取的特定日誌串流事件。安裝在環境中每個 Amazon EC2 執行個體上的 CloudWatch Logs 代理程式，會針對您所設定的每個日誌群組，將指標資料點發佈到 CloudWatch 服務。每個日誌群組會套用自己的篩選模式，來判斷哪些日誌串流事件做為資料點傳送到 CloudWatch。屬於同一個日誌群組的日誌串

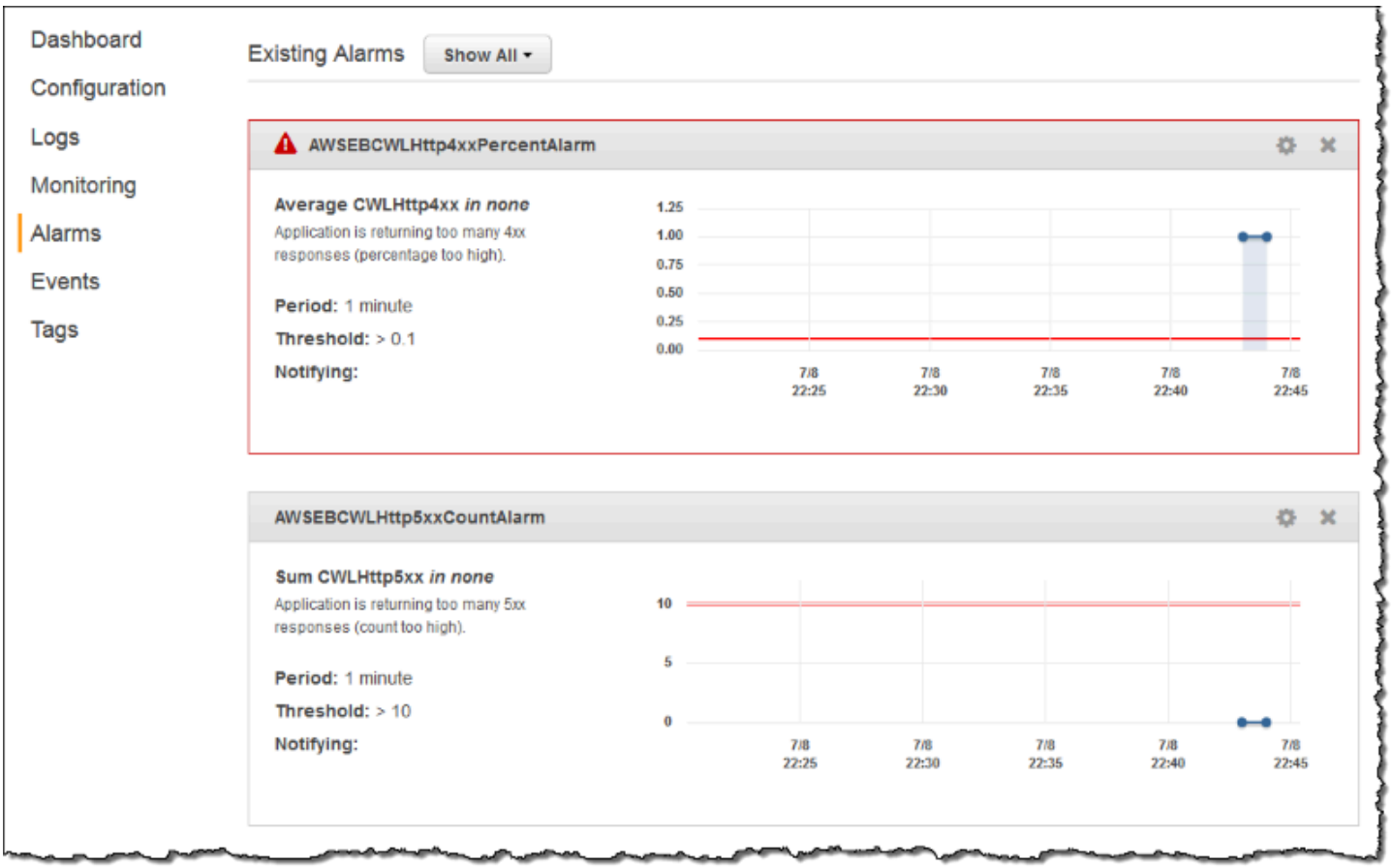
流，會共用相同的保留、監控和存取控制設定。您可以將 Elastic Beanstalk 設定為自動將日誌串流到 CloudWatch 服務，如 將執行個體日誌串流至 CloudWatch Logs 中所述。如需 CloudWatch Logs 的詳細資訊，包括術語和概念，請參閱 Amazon CloudWatch Logs 使用者指南。

除了執行個體日誌，如果您為環境啟用增強行運作狀態，您可以將環境設定為將運作狀態資訊串流到 CloudWatch Logs。請參閱 將 Elastic Beanstalk 環境運作狀態資訊串流至 Amazon CloudWatch Logs。

下圖顯示使用 CloudWatch Logs 整合所設定之環境的 Monitoring (監控) 頁面和圖表。此環境中的範例指標命名為 CWLHttp4xx 和 CWLHttp5xx。其中一個圖顯示 CWLHttp4xx 指標已根據組態檔案中所指定的條件觸發警示。



下圖顯示了 Alarms (警示) 頁面和圖表，其針對名為 AWSEBCWLHttp4xxPercentAlarm 和 AWSEBCWLHttp5xxCountAlarm 的範例警示，這些警示分別對應於 CWLHttp4xx 與 CWLHttp5xx 指標。



主題

- 執行個體日誌串流到 CloudWatch Logs 的必要條件
- Elastic Beanstalk 如何設定 CloudWatch Logs
- 將執行個體日誌串流至 CloudWatch Logs
- 疑難排解 CloudWatch Logs 整合
- 將 Elastic Beanstalk 環境運作狀態資訊串流至 Amazon CloudWatch Logs

執行個體日誌串流到 CloudWatch Logs 的必要條件

若要啟用從環境的 Amazon EC2 執行個體到 CloudWatch Logs 的日誌串流，您必須符合以下條件。

- 平台 – 由於此功能僅適用於此版本或之後發行的平台版本，如果您使用較早的平台版本，請將您的環境更新至目前的平台版本。
- 如果您的 Elastic Beanstalk 執行個體設定檔中未包含 AWSElasticBeanstalkWebTier 或 AWSElasticBeanstalkWorkerTier 受管政策，則您必須將下列程式碼加入您的設定檔中，以啟用此功能。

```
{
  "Version": "2012-10-17",
  "Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "logs:PutLogEvents",
      "logs:CreateLogStream"
    ],
    "Resource": [
    "*"
    ]
  }
  ]
}
```

Elastic Beanstalk 如何設定 CloudWatch Logs

Elastic Beanstalk 會在其建立的每個執行個體上，使用預設的組態設定安裝 CloudWatch 日誌代理程式。如需進一步了解，請參閱 CloudWatch Logs 代理程式參考。

當您啟用執行個體日誌串流到 CloudWatch Logs，Elastic Beanstalk 會將從您環境執行個體的日誌檔案傳送到 CloudWatch Logs。不同的平台會串流不同的日誌。下表根據平台列出了日誌。

平台/平台分支	日誌
Docker/ 平台分支：執行於 64 位元 Amazon Linux 2 的 Docker	• /var/log/eb-engine.log • /var/log/eb-hooks.log • /var/log/docker • /var/log/docker-events.log • /var/log/eb-docker/containers/eb-current-app/stdouterr.log • /var/log/nginx/access.log • /var/log/nginx/error.log
Docker/	• /var/log/docker-events.log • /var/log/eb-ecs-mgr.log • /var/log/eb-engine.log

平台/平台分支	日誌
平台分支：執行於 64 位元 Amazon Linux 2 的 ECS	• /var/log/eb-hooks.log • /var/log/ecs/ecs-agent.log • /var/log/ecs/ecs-init.log
Go Linux 上的 .NET Core Java/平台分支：執行於 64 位元 Amazon Linux 2 的 Corretto	• /var/log/eb-engine.log • /var/log/eb-hooks.log • /var/log/web.stdout.log • /var/log/nginx/access.log • /var/log/nginx/error.log
Node.js Python	• /var/log/eb-engine.log • /var/log/eb-hooks.log • /var/log/web.stdout.log • /var/log/httpd/access_log • /var/log/httpd/error_log • /var/log/nginx/access.log • /var/log/nginx/error.log
Tomcat PHP	• /var/log/eb-engine.log • /var/log/eb-hooks.log • /var/log/httpd/access_log • /var/log/httpd/error_log • /var/log/nginx/access.log • /var/log/nginx/error.log
Windows Server 上的 .NET	• C:\inetpub\logs\LogFiles\W3SVC1\u_ex*.log • C:\Program Files\Amazon\ElasticBeanstalk\logs\AWSDeployment.log • C:\Program Files\Amazon\ElasticBeanstalk\logs\Hooks.log

平台/平台分支	日誌
Ruby	• /var/log/eb-engine.log • /var/log/eb-hooks.log • /var/log/puma/puma.log • /var/log/web.stdout.log • /var/log/nginx/access.log • /var/log/nginx/error.log

Amazon Linux AMI 平台上的日誌檔案

> **Note**
>
> 2022 年 7 月 18 日，Elastic Beanstalk 已將所有以 Amazon Linux AMI (AL1) 為基礎的平台分支狀態設為已淘汰。如需有關遷移至完全支援的目前 Amazon Linux 2023 平台分支的詳細資訊，請參閱 將您的 Elastic Beanstalk Linux 應用程式遷移到 Amazon Linux 2023 或 Amazon Linux 2。

下表依據 Amazon Linux AMI (前身為 Amazon Linux 2)，列出從平台上的執行個體串流的日誌檔案。

平台/平台分支	日誌
Docker/ 平台分支：執行於 64 位元 Amazon Linux 的 Docker	• /var/log/eb-activity.log • /var/log/nginx/error.log • /var/log/docker-events.log • /var/log/docker • /var/log/nginx/access.log • /var/log/eb-docker/containers/eb-current-app/stdouterr.log
Docker/ 平台分支：執行於 64 位元 Amazon Linux 的多容器 Docker	• /var/log/eb-activity.log • /var/log/ecs/ecs-init.log • /var/log/eb-ecs-mgr.log • /var/log/ecs/ecs-agent.log

平台/平台分支	日誌
	• /var/log/docker-events.log
Glassfish (預先設定的 Docker)	• /var/log/eb-activity.log • /var/log/nginx/error.log • /var/log/docker-events.log • /var/log/docker • /var/log/nginx/access.log
Go	• /var/log/eb-activity.log • /var/log/nginx/error.log • /var/log/nginx/access.log
Java/ 平台分支：執行於 64 位元 Amazon Linux 的 Java 8 平台分支：執行於 64 位元 Amazon Linux 的 Java 7	• /var/log/eb-activity.log • /var/log/nginx/access.log • /var/log/nginx/error.log • /var/log/web-1.error.log • /var/log/web-1.log
Tomcat	• /var/log/eb-activity.log • /var/log/httpd/error_log • /var/log/httpd/access_log • /var/log/nginx/error_log • /var/log/nginx/access_log
Node.js	• /var/log/eb-activity.log • /var/log/nodejs/nodejs.log • /var/log/nginx/error.log • /var/log/nginx/access.log • /var/log/httpd/error.log • /var/log/httpd/access.log

平台/平台分支	日誌
PHP	• /var/log/eb-activity.log • /var/log/httpd/error_log • /var/log/httpd/access_log
Python	• /var/log/eb-activity.log • /var/log/httpd/error_log • /var/log/httpd/access_log • /opt/python/log/supervisord.log
Ruby/ 平台分支：執行於 64 位元 Amazon Linux 的 Puma with Ruby	• /var/log/eb-activity.log • /var/log/nginx/error.log • /var/log/puma/puma.log • /var/log/nginx/access.log
Ruby/ 平台分支：執行於 64 位元 Amazon Linux 的 Passenger with Ruby	• /var/log/eb-activity.log • /var/app/support/logs/passenger.log • /var/app/support/logs/access.log • /var/app/support/logs/error.log

Elastic Beanstalk 在 CloudWatch Logs 中針對其串流的各個日誌檔案，設定日誌群組。若要從 CloudWatch Logs 擷取特定的日誌檔案，您必須了解對應的日誌群組的名稱。日誌群組命名結構描述取決於平台的作業系統。

對於 Linux 平台，使用 `/aws/elasticbeanstalk/`*`environment_name`* 做為執行個體日誌檔案位置的前置詞，以取得日誌群組名稱。例如，如果要擷取檔案 `/var/log/nginx/error.log`，請指定日誌群組名稱 `/aws/elasticbeanstalk/`*`environment_name`*`/var/log/nginx/error.log`。

對於 Windows 平台，請參閱下表中對應到每個日誌檔的日誌群組。

執行個體上的日誌檔	日誌群組
C:\Program Files\Amazon\ElasticBeanstalk\logs\AWSDeployment.log	/aws/elasticbeanstalk/<environment-name>/EBDeploy-Log
C:\Program Files\Amazon\ElasticBeanstalk\logs\Hooks.log	/aws/elasticbeanstalk/<environment-name>/EBHooks-Log
C:\inetpub\logs\LogFiles (整個目錄)	/aws/elasticbeanstalk/<environment-name>/IIS-Log

將執行個體日誌串流至 CloudWatch Logs

您可以使用 Elastic Beanstalk 主控台、EB CLI 或組態選項，啟用串流至 CloudWatch Logs 的執行個體日誌。

啟用前，請設定搭配 CloudWatch Logs 代理程式使用的 IAM 許可。您可以將下列的自訂政策，連接到您指派給環境的執行個體設定檔。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogGroups",
        "logs:DescribeLogStreams"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

使用 Elastic Beanstalk 主控台的執行個體日誌串流

將執行個體日誌串流至 CloudWatch Logs

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇組態。
4. 在更新、監控和日誌記錄組態類別中，選擇編輯。
5. 在 Instance log streaming to CloudWatch Logs (執行個體日誌串流到 CloudWatch Logs) 下：
   - 啟用 Log streaming (日誌串流)。
   - 將 Retention (保留) 設為儲存日誌的天數。
   - 選取 Lifecycle (生命週期) 設定，其決定是否在環境終止後儲存日誌。
6. 若要儲存變更，請選擇頁面底部的儲存變更。

您在啟用日誌串流後，返回 Software (軟體) 組態類別或頁面，並且尋找 Log Groups (日誌群組) 連結。按一下此連結，在 CloudWatch 主控台中查看您的日誌。

使用 EB CLI 執行個體日誌串流

若要使用 EB CLI 將執行個體日誌串流到 CloudWatch Logs，請使用 eb logs 命令。

```
$ eb logs --cloudwatch-logs enable
```

您也可以使用 eb logs 擷取 CloudWatch Logs 的日誌，您可以擷取所有環境的執行個體日誌，或使用命令的許多選項，以指定要擷取的日誌子集。例如，以下命令為您的環境中擷取整組的執行個體日誌，並將它們儲存到 `.elasticbeanstalk/logs` 底下目錄。

```
$ eb logs --all
```

尤其是，`--log-group` 選項可讓您擷取特定日誌群組的執行個體日誌，其對應特定現場執行個體日誌檔。若要這樣做，您需要知道對應到您要擷取的日誌檔案的日誌群組名稱。您可以在 Elastic Beanstalk 如何設定 CloudWatch Logs 中尋找此資訊。

使用組態檔案的執行個體日誌串流

當您建立或更新環境時，您可以使用組態檔案來設定與配置執行個體日誌串流到 CloudWatch Logs。以下範例組態檔啟用預設執行個體日誌串流。Elastic Beanstalk 為您環境的平台串流預設的日誌檔組合。若要使用此範例，可將文字複製到您的應用程式來源套件最上層，`.ebextensions` 目錄中具有 `.config` 副檔名的檔案。

```
option_settings:
  - namespace: aws:elasticbeanstalk:cloudwatch:logs
    option_name: StreamLogs
    value: true
```

自訂日誌檔串流

與 CloudWatch Logs 的 Elastic Beanstalk 整合不會直接支援您的應用程式所產生的自訂日誌檔串流。若要串流日誌檔，請使用組態檔直接安裝 CloudWatch Logs 代理程式，並設定要推送的檔案。如需範例組態檔，請參閱 `logs-streamtocloudwatch-linux.config`。

> **Note**
>
> 此範例不適用於 Windows 平台。

如需有關設定 CloudWatch Logs 的詳細資訊，請參閱《Amazon CloudWatch Logs 使用者指南》中的 CloudWatch Logs 代理程式參考。

疑難排解 CloudWatch Logs 整合

如果在 CloudWatch Logs 中找不到您預期的一些環境執行個體日誌，您可以調查以下常見問題：

- 您的 IAM 角色缺少所需的 IAM 權限。
- 您在不支援 CloudWatch Logs 的 AWS 區域啟動您的環境。
- 其中一個自訂日誌檔不存在於您指定的路徑。

將 Elastic Beanstalk 環境運作狀態資訊串流至 Amazon CloudWatch Logs

如果您啟用增強型運作狀態報告，您可以將環境設定為將運作狀態資訊串流到 CloudWatch Logs。此串流與 Amazon EC2 執行個體日誌串流無關。此主題說明環境運作狀態資訊串流。如需執行個體日誌串流的詳細資訊，請參閱搭配 Amazon CloudWatch Logs 使用 Elastic Beanstalk。

當您設定環境運作狀態串流時，Elastic Beanstalk 會針對環境運作狀態建立 CloudWatch Logs 日誌群組。日誌群組的名稱是 `/aws/elasticbeanstalk/`*`environment-name`*`/environment-health.log`。在這個日誌群組內，Elastic Beanstalk 會建立名為 *`YYYY-MM-DD`*`#`*`<hash-suffix>`* 的日誌串流 (每個日期可能有一個以上的日誌串流)。

當環境的運作狀態變更，Elastic Beanstalk 會將記錄新增到運作狀態日誌串流。該記錄代表運作狀態轉換 - 新的狀態和描述變更的原因。例如，環境的狀態會因負載平衡器失敗而變更為「嚴重」。如需增強型運作狀況的詳細說明，請參閱運作狀態顏色和狀態。

環境運作狀態串流至 CloudWatch Logs 的必要條件

若要啟用 CloudWatch Logs 的環境運作狀態串流，您必須符合以下條件：

- 平台 – 您必須使用可支援增強型運作狀態報告的平台版本。
- 許可 – 您必須將特定記錄相關許可授與 Elastic Beanstalk，以便代表您為環境串流運作狀態資訊。如果您的環境未使用 Elastic Beanstalk 為它建立的服務角色 `aws-elasticbeanstalk-service-role`，或您的帳戶服務連結角色 `AWSServiceRoleForElasticBeanstalk`，請務必將以下許可新增至您的自訂服務角色。

```
{
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogStreams",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": "arn:aws:logs:*:*:log-group:/aws/elasticbeanstalk/*:log-stream:*"
}
```

將環境運作狀態日誌串流至 CloudWatch Logs

您可以使用 Elastic Beanstalk 主控台、EB CLI 或組態選項，啟用串流至 CloudWatch Logs 的環境運作狀態。

使用 Elastic Beanstalk 主控台的環境運作狀態日誌串流

將環境運作狀態日誌串流至 CloudWatch Logs

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。

2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇 Configuration (組態)。
4. 在 Monitoring (監控) 組態類別中，選擇 Edit (編輯)。
5. 在 Health reporting (運作狀態報告) 下，確保報告 System (系統) 設為 Enhanced (增強型)。
6. 在 Health event streaming to CloudWatch Logs (運作狀態事件串流到 CloudWatch Logs) 下

   - 啟用 Log streaming (日誌串流)。
   - 將 Retention (保留) 設為儲存日誌的天數。
   - 選取 Lifecycle (生命週期) 設定，其決定是否在環境終止後儲存日誌。
7. 若要儲存變更，請選擇頁面底部的儲存變更。

您在啟用日誌串流後，返回 Monitoring (監控) 組態類別或頁面，並且尋找 Log Group (日誌群組) 連結。按一下此連結，在 CloudWatch 主控台查看您的環境運作狀態日誌。

使用 EB CLI 的環境運作狀態日誌串流

若要使用 EB CLI 將環境運作狀態日誌串流到 CloudWatch Logs，請使用 eb logs 命令。

```
$ eb logs --cloudwatch-logs enable --cloudwatch-log-source environment-health
```

您也可以使用 eb logs 擷取 CloudWatch Logs 的日誌，例如，以下命令為您的環境中擷取所有的運作狀態日誌，並將它們儲存到 `.elasticbeanstalk/logs` 底下的目錄。

```
$ eb logs --all --cloudwatch-log-source environment-health
```

使用組態檔案的環境運作狀態日誌串流

當您建立或更新環境時，您可以使用組態檔案來設定與配置串流到 CloudWatch Logs 的環境運作狀態日誌。若要使用以下範例，可將文字複製到您的應用程式來源套件最上層，`.config` 目錄中具有 `.ebextensions` 副檔名的檔案。在範例中，設定 Elastic Beanstalk 以啟用環境運作狀態日誌串流，終止環境後保留日誌，並儲存 30 天。

Example 運作狀態串流組態檔案

```
##############################################################################
##  Sets up Elastic Beanstalk to stream environment health information
##  to Amazon CloudWatch Logs.
##  Works only for environments that have enhanced health reporting enabled.
##############################################################################

option_settings:
  aws:elasticbeanstalk:cloudwatch:logs:health:
    HealthStreamingEnabled: true
    ### Settings below this line are optional.
    # DeleteOnTerminate: Delete the log group when the environment is
    # terminated. Default is false. If false, the health data is kept
    # RetentionInDays days.
    DeleteOnTerminate: false
    # RetentionInDays: The number of days to keep the archived health data
    # before it expires, if DeleteOnTerminate isn't set. Default is 7 days.
    RetentionInDays: 30
```

如需選項預設值和有效值的詳細資訊，請參閱 aws:elasticbeanstalk:cloudwatch:logs:health。

搭配 Amazon EventBridge 使用 Elastic Beanstalk

使用 Amazon EventBridge，您可以設定事件驅動規則，監控您的 Elastic Beanstalk 資源，並啟動使用其他 AWS 服務的目標動作。例如，您可以透過在生產環境的運作狀態變更為 Warning (警告) 狀態時傳送 Amazon SNS 主題，來設定傳出電子郵件通知的規則。或者，您可以在環境的運作狀態變更為 Degraded (已降級) 或 Severe (嚴重) 狀態時傳遞通知至 Slack，來設定 Lambda 函數。

您可以在 Amazon EventBridge 中建立規則，以對下列任何一個 Elastic Beanstalk 事件採取行動：

- 環境操作的狀態變更 (包括建立、更新和終止操作)。事件將指明狀態變更是否已開始、成功或失敗。
- 其他資源的狀態變更。除了環境之外，受監控的其他資源包括負載平衡器、Auto Scaling 群組和執行個體。
- 環境的運作狀態轉換。事件指出環境運作狀態已從運作狀態轉換到另一個運作狀態。
- 受管更新的狀態變更。事件將指明狀態變更是否已開始、成功或失敗。

若要擷取您感興趣的特定 Elastic Beanstalk 事件，請定義 EventBridge 可用來偵測事件的特定事件模式。事件模式擁有與其相符事件相同的結構。該模式引用您欲比對的欄位，並提供您正在尋找的數值。盡可能發出事件。在正常操作情況下，其可近乎即時地從 Elastic Beanstalk 傳送至 EventBridge。但是，可能會發生延遲或阻止事件傳遞的情況。

如需 Elastic Beanstalk 事件中包含的欄位清單及其可能的字串值，請參閱 Elastic Beanstalk 事件欄位映射。如需 EventBridge 規則如何使用事件模式的詳細資訊，請參閱 EventBridge 中的事件和事件模式。

使用 EventBridge 監控 Elastic Beanstalk 資源

使用 EventBridge，您可以建立規則，其定義當 Elastic Beanstalk 針對其資源發出事件時要採取的動作。例如，您可以建立一個規則，在環境狀態變更時傳送電子郵件訊息給您。

EventBridge 主控台具有預先定義的模式選項，可建置 Elastic Beanstalk 事件模式。如果建立規則時在 EventBridge 主控台中選取此選項，您即可快速建置 Elastic Beanstalk 事件模式。您只需選取事件欄位和值。在您選取後，主控台即會建置並顯示事件模式。或者，您可以手動編輯您建置的事件模式，並將其儲存為自訂模式。此外，主控台還會提供選項來顯示詳細的範例事件，您可以複製並貼至您正在建置的事件模式。

如果您想要輸入或複製事件模式並貼至 EventBridge 主控台，可以選擇使用主控台中的 Custom pattern (自訂模式) 選項。這樣做，您不需要執行前述選取欄位和值的步驟。本主題提供您可以使用的事件比對模式和 Elastic Beanstalk 事件的範例。

建立資源事件的規則

1. 使用具有使用 EventBridge 和 Elastic Beanstalk 許可的帳戶登入 AWS。
2. 在 https://console.aws.amazon.com/events/ 開啟 Amazon EventBridge 主控台。
3. 在導覽窗格中，選擇 Rules (規則)。
4. 選擇 Create rule (建立規則)。
5. 輸入規則的Name (名稱)，或者輸入描述。
6. 針對 Event bus (事件匯流排) 選擇 default (預設值)。當您帳戶中的 AWS 服務發出事件時，一律會前往您帳戶的預設事件匯流排。
7. 針對 Rule type (規則類型) 選擇 Rule with an event pattern (具有事件模式的規則)。
8. 選擇 Next (下一步)。
9. 在 Event source (事件來源) 欄位中，選擇 AWS events or EventBridge partner events (事件或 EventBridge 合作夥伴事件)。

10. (選用) 對於 Sample event (範例事件)，請選擇 AWS 事件。在搜尋欄位中輸入 Elastic Beanstalk。這將提供您可以選擇顯示的 Elastic Beanstalk 事件範例清單。此步驟僅顯示您可以參考的範例事件。它不會影響規則建立的結果。本主題後面的 Elastic Beanstalk 事件範例 區段提供相同類型事件的範例。
11. 在 Event pattern (事件模式) 區段中，選擇 Event pattern form (事件模式表單)。

    > **Note**
    >
    > 如果已經有事件模式的文字，而且不需要使用 EventBridge 主控台進行建置，您可以選取 Custom pattern (JSON editor) (自訂模式 (JSON 編輯器))。然後，可以手動輸入或複製文字並貼至 Event Pattern (事件模式) 方塊。選擇 Next (下一步)，然後轉到有關輸入目標的步驟。

12. 在 Event source (事件來源) 欄位中，選擇 AWS services (服務)。
13. 針對 AWS Service (AWS 服務)，選取 Elastic Beanstalk。
14. 針對 Event type (事件類型)，選取 Status Change (狀態變更)。
15. 此步驟涵蓋如何使用 Elastic Beanstalk 的 detail type (詳細資訊類型)、status (狀態) 和 severity (嚴重性) 事件欄位。選擇這些欄位和想要比對的值後，主控台會建置並顯示事件模式。

    - 如果您為 Specific detail type(s) (特定詳細資訊類型)選擇 僅一個值，您可以針對階層中的下一個欄位選擇一個或多個值。
    - 如果您為 Specific detail type(s) (特定詳細資訊類型) 選擇多個值，則請勿針對階層中下一個欄位選擇特定值。這樣可以防止事件模式中欄位的模糊比對邏輯。

    environment (環境) 事件欄位不受此階層的影響，因此它會依照下一個步驟中所述顯示。
16. 針對 Environment (環境)，選取 Any environment (任何環境) 或 Specific environment(s) (特定環境)。

    - 如果您選取 Specific environment(s) (特定環境)，可以從下拉式清單中選擇一或多個環境。EventBridge 會新增您在事件模式 detail (詳細資訊) 區段的 EnvironmentName[ ] 清單中選取的所有環境。然後，您的規則會篩選所有事件，以僅包含您選擇的特定環境。
    - 如果您選取 Any environment (任何環境)，則不會將任何環境新增至您的事件模式。因此，您的規則不會根據環境來篩選任何 Elastic Beanstalk 事件。
17. 選擇 Next (下一步)。
18. 在 Target types (目標類型) 欄位中，選擇 AWS service (服務)。

19. 在 Select a target (選取目標) 下，選擇從 Elastic Beanstalk 收到資料狀態變更事件時，要採取的目標動作。

    例如，您可以使用 Amazon Simple Notification Service (SNS) 主題，在事件發生時，傳送電子郵件或文字訊息。若要執行此操作，您需要使用 Amazon SNS 主控台建立 Amazon SNS 主題。若要進一步了解，請參閱使用 Amazon SNS 傳送使用者通知。

    > **Important**
    >
    > 某些目標動作可能需要使用其他服務，並產生額外費用，例如 Amazon SNS 或 Lambda 服務。如需 AWS 定價的詳細資訊，請參閱 https://aws.amazon.com/pricing/。某些服務屬於 AWS 免費用量方案。若您是新客戶，可以免費試用這些服務。如需詳細資訊，請參閱 https://aws.amazon.com/free/。

20. (選用) 選擇 Add another target (新增其他目標)，以指定事件規則的其他目標動作。
21. 選擇 Next (下一步)。
22. (選用) 為規則輸入一或多個標籤。如需詳細資訊，請參閱《Amazon EventBridge 使用者指南》中的 Amazon EventBridge 標籤。
23. 選擇 Next (下一步)。
24. 檢閱規則的詳細資訊，然後選擇 Create rule (建立規則)。

Elastic Beanstalk 事件模式範例

事件模式擁有與其相符事件相同的結構。該模式引用您欲比對的欄位，並提供您正在尋找的數值。

- 所有環境的運作狀態變更

```
{
   "source": [
    "aws.elasticbeanstalk"
  ],
  "detail-type": [
    "Health status change"
    ]
}
```

- 下列環境的運作狀態變更：myEnvironment1 和 myEnvironment2。此事件模式會針對這兩個特定環境進行篩選，而之前的 Health status change (運作狀態變更) 範例不會針對所有環境來傳送事件。

```
{"source": [
    "aws.elasticbeanstalk"
    ],
    "detail-type": [
        "Health status change"
    ],
    "detail": {
        "EnvironmentName": [
            "myEnvironment1",
            "myEnvironment2"
        ]
    }
}
```

- 所有環境的 Elastic Beanstalk 資源狀態變更

```
{
  "source": [
    "aws.elasticbeanstalk"
  ],
  "detail-type": [
    "Elastic Beanstalk resource status change"
    ]
}
```

- 下列環境的 Elastic Beanstalk 資源狀態變更且 Status 環境更新失敗並顯示 Severity 錯誤：myEnvironment1 和 myEnvironment2

```
{"source": [
    "aws.elasticbeanstalk"
    ],
    "detail-type": [
        "Elastic Beanstalk resource status change"
    ],
    "detail": {
        "Status": [
            "Environment update failed"
            ],
```

```
        "Severity": [
            "ERROR"
            ],
        "EnvironmentName": [
            "myEnvironment1",
            "myEnvironment2"
        ]
    }
}
```

- 負載平衡器、Auto Scaling 群組和執行個體的其他資源狀態變更

```
{
   "source": [
    "aws.elasticbeanstalk"
  ],
  "detail-type": [
    "Other resource status change"
    ]
}
```

- 所有環境的受管更新狀態變更

```
{
   "source": [
    "aws.elasticbeanstalk"
  ],
  "detail-type": [
    "Managed update status change"
    ]
}
```

- 從 Elastic Beanstalk 擷取所有事件 (不包括 `detail-type` 區段)

```
{
  "source": [
    "aws.elasticbeanstalk"
  ]
}
```

Elastic Beanstalk 事件範例

以下是資源狀態變更的 Elastic Beanstalk 事件範例：

```
{
   "version":"0",
   "id":"1234a678-1b23-c123-12fd3f456e78",
   "detail-type":"Elastic Beanstalk resource status change",
   "source":"aws.elasticbeanstalk",
   "account":"111122223333",
   "time":"2020-11-03T00:31:54Z",
   "region":"us-east-1",
   "resources":[
      "arn:was:elasticbeanstalk:us-east-1:111122223333:environment/myApplication/
myEnvironment"
   ],
   "detail":{
      "Status":"Environment creation started",
      "EventDate":1604363513951,
      "ApplicationName":"myApplication",
      "Message":"createEnvironment is starting.",
      "EnvironmentName":"myEnvironment",
      "Severity":"INFO"
   }
}
```

以下是運作狀態變更的 Elastic Beanstalk 事件範例：

```
{
   "version":"0",
   "id":"1234a678-1b23-c123-12fd3f456e78",
   "detail-type":"Health status change",
   "source":"aws.elasticbeanstalk",
   "account":"111122223333",
   "time":"2020-11-03T00:34:48Z",
   "region":"us-east-1",
   "resources":[
      "arn:was:elasticbeanstalk:us-east-1:111122223333:environment/myApplication/
myEnvironment"
   ],
   "detail":{
      "Status":"Environment health changed",
      "EventDate":1604363687870,
```

```
      "ApplicationName":"myApplication",
      "Message":"Environment health has transitioned from Pending to Ok. Initialization
 completed 1 second ago and took 2 minutes.",
      "EnvironmentName":"myEnvironment",
      "Severity":"INFO"
   }
}
```

Elastic Beanstalk 事件欄位映射

下表會將 Elastic Beanstalk 事件欄位及其可能的字串值映射至 EventBridge `detail-type` 欄位。如需有關 EventBridge 如何處理服務事件模式的詳細資訊，請參閱 EventBridge 中的事件和事件模式。

EventBridge 欄位詳細資訊類型	Elastic Beanstalk 欄位狀態	Elastic Beanstalk 欄位嚴重性	Elastic Beanstalk 欄位訊息
Elastic Beanstalk 資源狀態變更	已啟動環境建立	INFO	createEnvironment 正在啟動。
	環境建立成功	INFO	createEnvironment 已順利完成。
	環境建立成功	INFO	已啟動的環境：<Environment Name>。但是，啟動期間發生問題。如需詳細資訊，請參閱事件日誌。
	環境建立失敗	ERROR	無法啟動環境。
	已啟動環境更新	INFO	環境更新正在啟動。
	環境更新成功	INFO	環境更新已順利完成。
	環境更新失敗	ERROR	無法部署組態。
	已啟動環境終止	INFO	terminateEnvironment 正在啟動。

EventBridge 欄位詳細資訊類型	Elastic Beanstalk 欄位狀態	Elastic Beanstalk 欄位嚴重性	Elastic Beanstalk 欄位訊息
	環境終止成功	INFO	terminateEnvironment 已順利完成。
	環境終止失敗	INFO	環境終止步驟失敗，因為至少有一個環境終止工作流程失敗。
其他資源狀態變更	已建立 Auto Scaling 群組	INFO	createEnvironment 正在啟動。
	已刪除 Auto Scaling 群組	INFO	createEnvironment 正在啟動。
	已新增執行個體	INFO	已將執行個體 [i-123456789a12b1234] 新增至您的環境。
	已移除執行個體	INFO	已將執行個體 [i-123456789a12b1234] 從您的環境中移除。
	已建立負載平衡器	INFO	建立的負載平衡器名稱為：<LB Name>
	已刪除負載平衡器	INFO	已刪除的負載平衡器名稱為：<LB Name>
運作狀態變更	環境運作狀態變更	資訊/警告	環境運作狀態已轉換為 <healthStatus>.
	環境運作狀態變更	資訊/警告	環境運作狀態已從 <healthStatus> 轉換至 <healthStatus>。

EventBridge 欄位詳細資訊類型	Elastic Beanstalk 欄位狀態	Elastic Beanstalk 欄位嚴重性	Elastic Beanstalk 欄位訊息
受管更新狀態變更	已啟動受管更新	INFO	受管平台更新正在進行中。
	受管更新失敗	INFO	受管更新失敗，請在 %s 分鐘內重試。

使用 AWS Config 尋找和追蹤 Elastic Beanstalk 資源

AWS Config 提供您 AWS 帳戶中 AWS 資源組態的詳細檢視。您可以了解資源如相關聯，可以取得組態變更歷程記錄，並且了解關係和組態隨時間產生的變化。您可以使用 AWS Config 來定義評估資料合規性的資源組態的規則。

數種 Elastic Beanstalk 資源類型已與 整合AWS Config

- 應用程式
- 應用程式版本
- 環境

下一節說明如何設定 AWS Config 以記錄這些類型的資源。

如需 AWS Config 的詳細資訊，請參閱《AWS Config 開發人員指南》。如需定價資訊，請參閱AWS Config 定價資訊頁面。

設定 AWS Config

若要初始設定 AWS Config，請參閱下列在 AWS Config 開發人員指南中的主題。

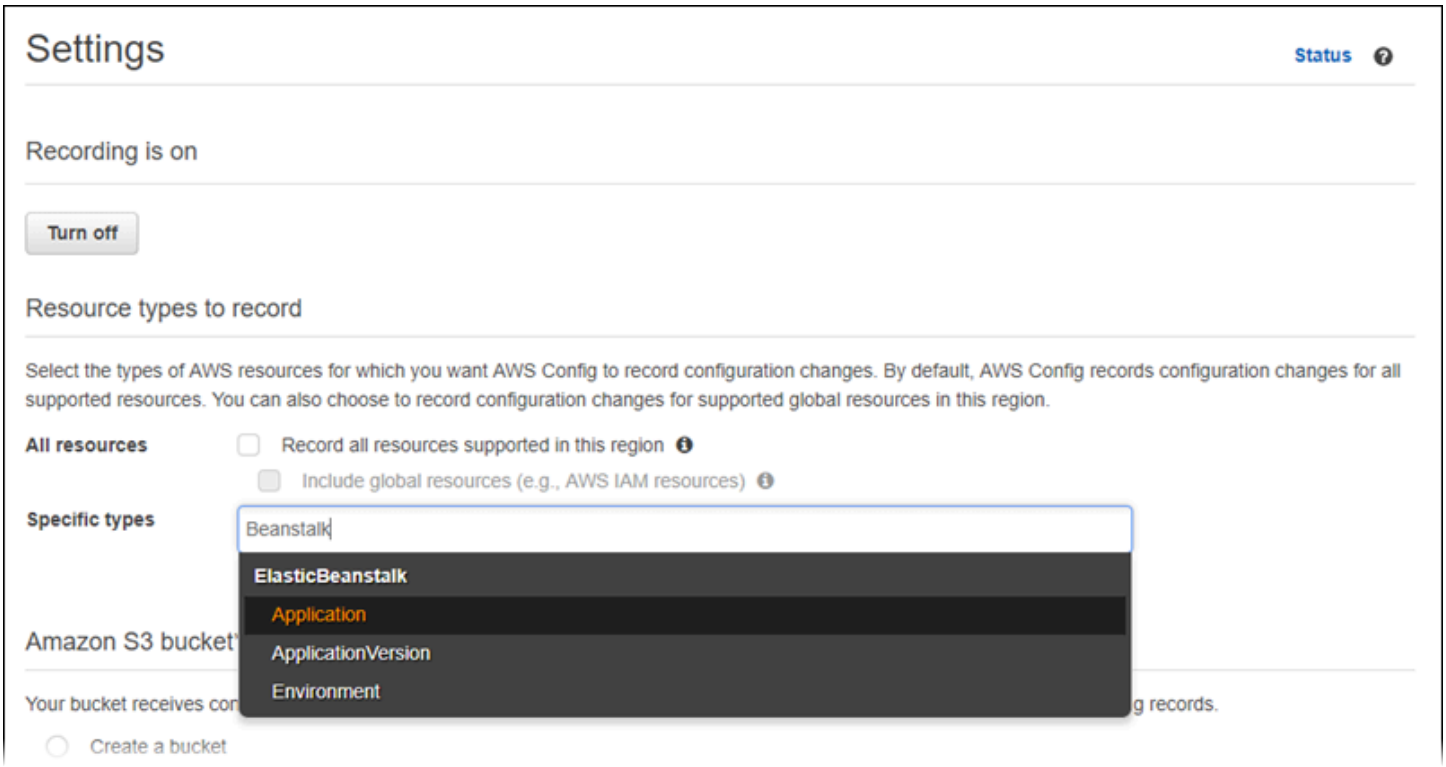
- 設定 AWS Config 與主控台
- 設定 AWS Config 與 AWS CLI

設定 AWS Config 以記錄 Elastic Beanstalk 資源

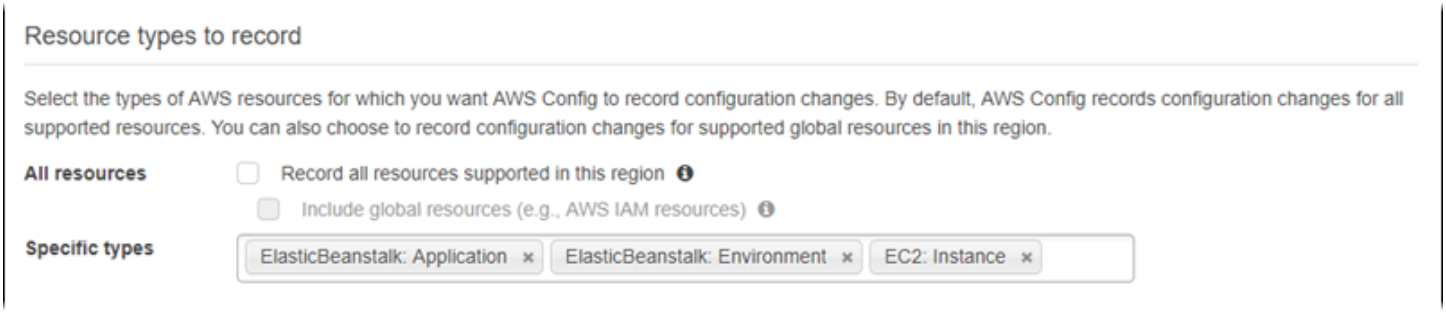
在預設情況下，AWS Config 會記錄所有區域資源支援類型的組態變更，它會探索環境正在執行的區域。您可以自訂 AWS Config 以記錄變更，其僅適用於特定的資源類型，或全域資源的變化。

例如，您可以設定 AWS Config 來記錄 Elastic Beanstalk 資源的變更，以及 Elastic Beanstalk 為您啟動的其他 AWS 資源子集變更。使用 AWS Config 主控台，您可以從 Specific Types (特定類型) 欄位中選取 Elastic Beanstalk 作為 AWS Config Settings (設定) 頁面中的資源。從該處，您可以選擇記錄任何 Elastic Beanstalk 資源類型：Application (應用程式)、ApplicationVersion 和 Environment (環境)。

下圖顯示 AWS Config 的 Settings (設定) 頁面，以及您可以選擇記錄的 Elastic Beanstalk 資源類型：Application (應用程式)、ApplicationVersion (應用程式版本) 和 Environment (環境)。



在您選擇幾個資源類型後，這是顯示 Specific types (特定類型) 清單的方式。



如欲了解「區域」與「全域」資源，以及完整的自訂程序，請參閱選取 AWS Config 記錄的資源。

在 AWS Config 主控台中檢視 Elastic Beanstalk 組態詳細資訊

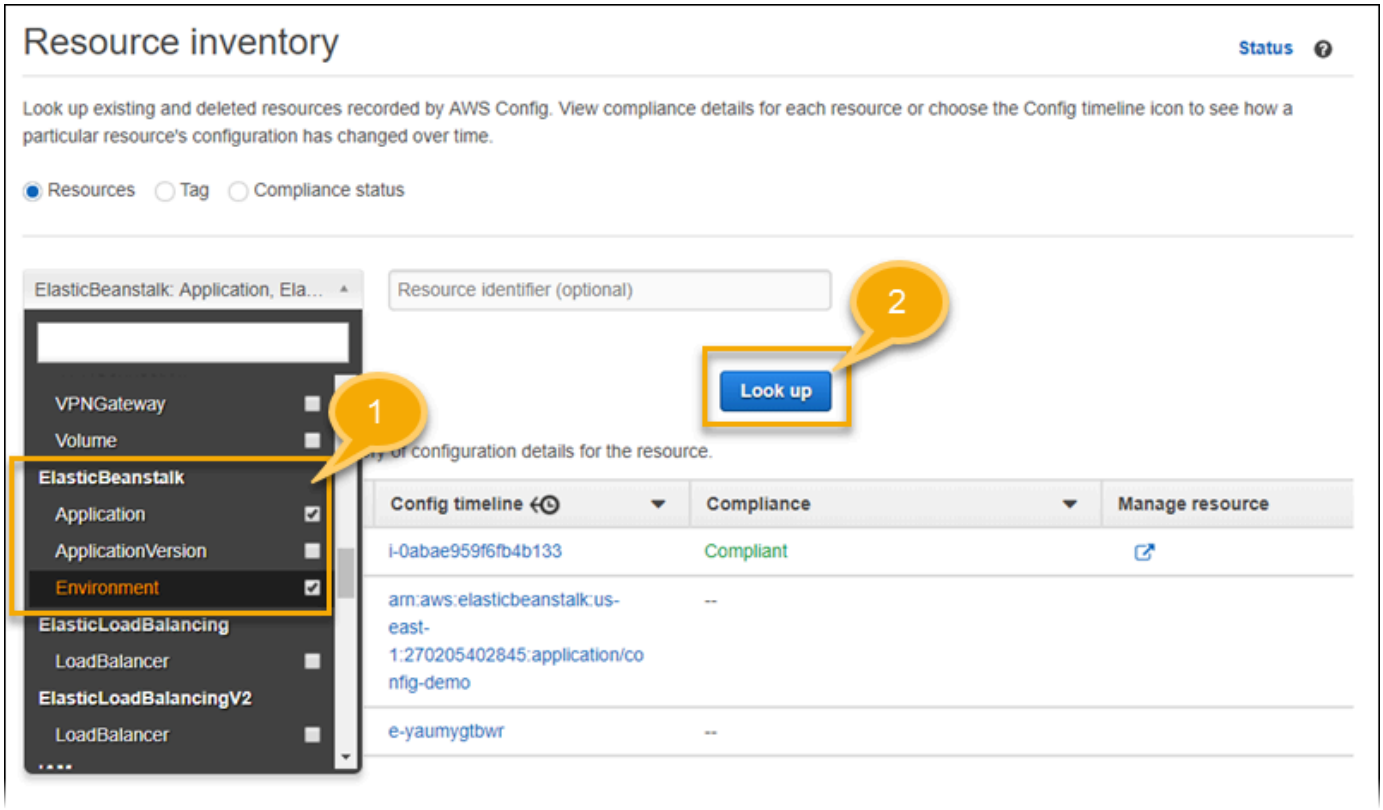
您可以使用 AWS Config 主控台來尋找 Elastic Beanstalk 資源，並取得其組態的目前和歷史詳細資訊。以下範例示範如何尋找 Elastic Beanstalk 環境的相關資訊。

在 AWS Config 主控台中尋找 Elastic Beanstalk 環境

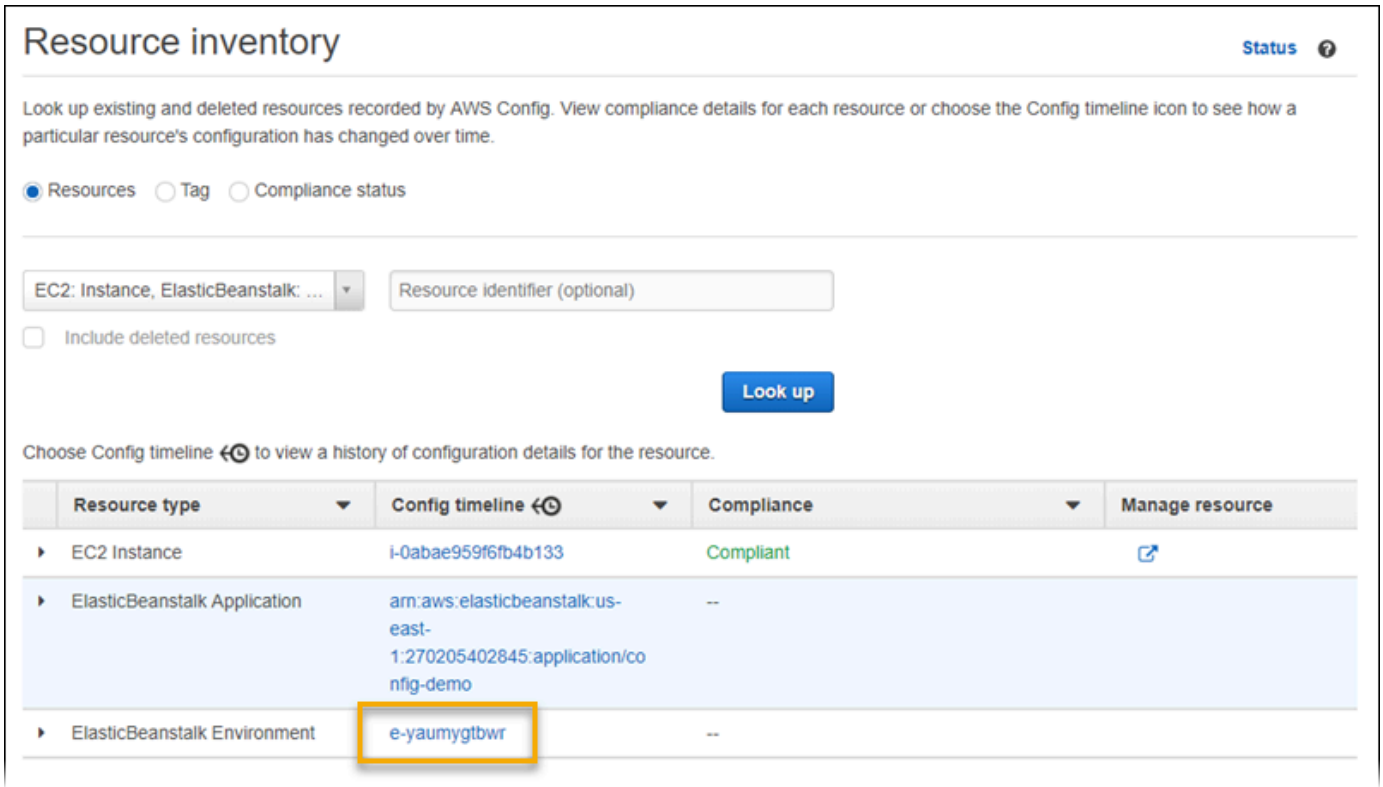
1. 開啟 AWS Config 主控台。
2. 選擇 Resources (資源)。
3. 在 Resource (資源) 清單頁面上，選擇 Resources (資源)。
4. 開啟 Resource type (資源類型) 選單，捲動到 ElasticBeanstalk，然後選擇一或多個 Elastic Beanstalk 資源類型。

> **Note**
>
> 若要檢視 Elastic Beanstalk 為您應用程式所建立的其他資源組態詳細資訊，請選擇其他資源類型。例如，您可以選擇 EC2 (EC2) 下的 Instance (執行個體)。

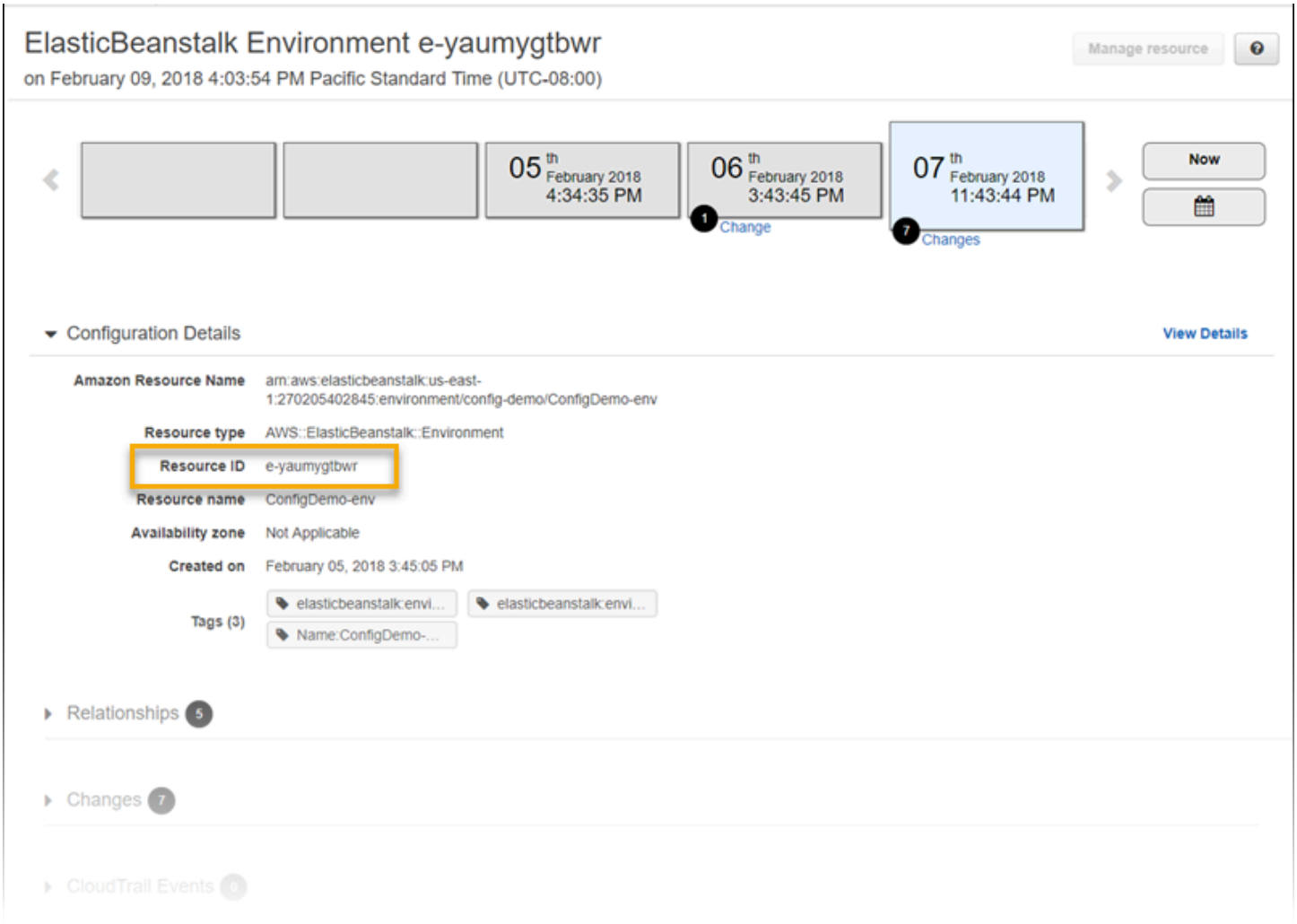
5. 選擇 Look up (查閱)。參見下圖中的 2 (2)。



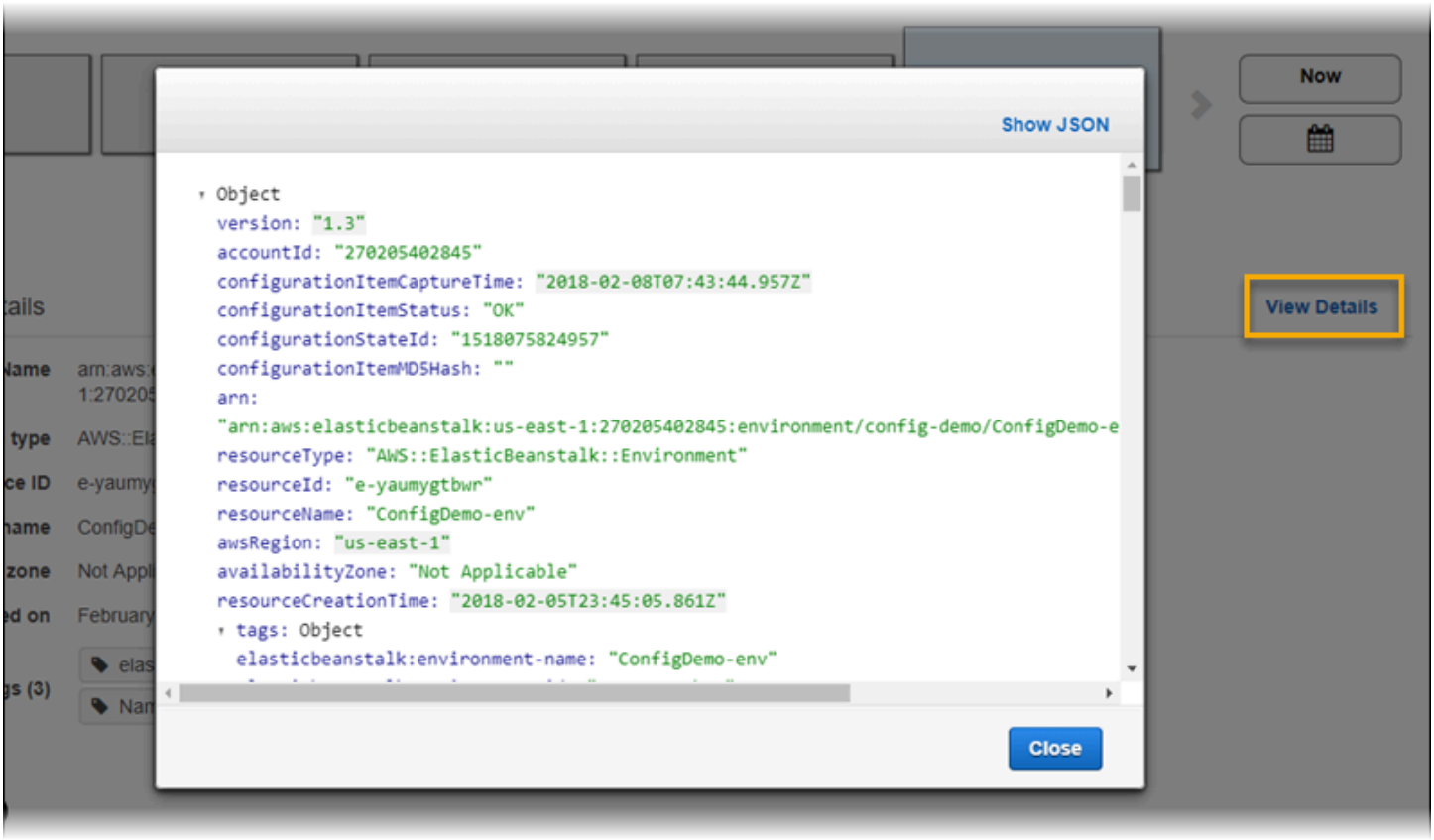
6. 在 AWS Config 顯示的資源清單中選擇資源 ID。



AWS Config 顯示您選取的資源的組態詳細資訊和其他資訊。



若要查看記錄組態的完整詳細資訊，請選擇 View Details (檢視詳細資訊)。



若要了解尋找資源和檢視此頁面資訊的更多方式，請參閱 AWS 開發人員指南中的檢視 AWS Config 資源組態和歷史紀錄。

使用 AWS Config 規則評估 Elastic Beanstalk 資源

您可以建立 AWS Config 規則，這代表 Elastic Beanstalk 資源的理想組態設定。可以使用預先定義的 AWS Managed Config Rules (AWS 受管 Config 規則)，或是定義自訂規則。AWS Config 會持續追蹤您資源組態的變更，以判斷變更是否會違反您的規則條件。AWS Config 主控台會顯示您規則和資源的合規狀態。

如果資源違反規則並標示為不合規，AWS Config 可以使用 Amazon Simple Notification Service (Amazon SNS) 主題來提醒您。如要透過程式設計方式使用這些 AWS Config 提醒中的資料，請使用 Amazon Simple Queue Service (Amazon SQS) 佇列做為 Amazon SNS 主題的通知端點。例如，當有人修改您環境的 Auto Scaling 群組組態時，您可能想要編寫程式碼以啟動一段工作流程。

若要進一步了解設定及使用規則，請參閱 AWS Config 開發人員指南中的使用 AWS Config 規則評估資源。

搭配 Amazon DynamoDB 使用 Elastic Beanstalk

Amazon DynamoDB 是一項完全受管的 NoSQL 資料庫服務，可提供快速且可預期的效能及無縫的可擴展性。如果您是開發人員，可以使用 DynamoDB 來建立資料庫資料表，來存放和擷取任意數量的資料，並針對任何量級的請求傳輸量提供服務。DynamoDB 會自動將資料表的資料與傳輸流分散到足夠數量的伺服器上，以因應客戶所指定的請求處理容量和儲存的資料量，同時保持快速、一致的效能。所有資料項目皆儲存於固態硬碟 (SSD) 上，並自動複製到 AWS 區域中的多個可用區域，以提供內建的高可用性和資料耐久性。

如果您在工作者環境中使用週期性任務，Elastic Beanstalk 會建立 DynamoDB 資料表，並利用此資料表來選擇領導者和儲存關於任務的資訊。環境中的每個執行個體，都會試著每隔幾秒寫入資料表，以成為領導者，並執行排定的任務。

您可以使用組態檔案來建立適合您應用程式的 DynamoDB 資料表。請參閱 GitHub 上的 eb-node-express-sample，此一範例 Node.js 應用程式會使用組態檔案來建立資料表，並透過適用於 Node.js 中 JavaScript 的 AWS 開發套件，來連線到此資料表。如需逐步說明搭配 DynamoDB 使用 PHP 的範例，請參閱範例：DynamoDB、CloudWatch 和 SNS。如需使用 AWS SDK for Java 的範例，請參閱 AWS SDK for Java 文件中的使用 DynamoDB 管理 Tomcat 工作階段狀態。

當您使用組態檔案建立 DynamoDB 表格，表格不受限於您環境的生命週期，不會在您終止環境時刪除。為了確保未不必要保留個人資訊，當不再需要任何記錄或表格時，請刪除。

如需有關 DynamoDB 的詳細資訊，請參閱 DynamoDB 開發人員指南。

搭配 Amazon ElastiCache 使用 Elastic Beanstalk

Amazon ElastiCache 是一種 Web 服務，可以針對雲端中的分散式記憶體內快取環境進行設定、管理與擴展。此項服務提供高效能、可擴展和符合成本效益的記憶體內快取，同時消除了部署與管理分散式快取環境的相關複雜性。ElastiCache 在通訊協定方面與 Redis 和 Memcached 相容，因此您現有 Redis 和 Memcached 環境目前所使用的程式碼、應用程式和最熱門的工具，仍可順暢地搭配該服務使用。如需有關 ElastiCache 的詳細資訊，請前往 Amazon ElastiCache 產品頁面。

搭配 Amazon ElastiCache 使用 Elastic Beanstalk

1. 建立一個 ElastiCache 叢集。

   - 如需有關如何使用 Redis 建立 ElastiCache 叢集的說明，請參閱《ElastiCache for Redis 使用者指南》中的 Amazon ElastiCache for Redis 入門。

- 如需有關如何使用 Memcached 建立 ElastiCache 叢集的說明，請參閱《ElastiCache for Memcached 使用者指南》中的 Amazon ElastiCache for Memcached 入門。

2. 設定您的 ElastiCache 安全群組，以允許從您的 Elastic Beanstalk 應用程式所使用的 Amazon EC2 安全群組來進行存取。如需有關如何使用 AWS 管理主控台尋找 EC2 安全群組名稱的方法，請參閱 EC2 執行個體文件頁面上的 安全群組。

   - 如需有關 Redis 的詳細資訊，請參閱 ElastiCache for Redis 使用者指南中的授權存取。
   - 如需有關 Memcached 的詳細資訊，請參閱《ElastiCache for Memcached 使用者指南》中的授權存取。

您可以使用組態檔案來自訂您的 Elastic Beanstalk 環境以使用 ElastiCache。如需整合 ElastiCache 與 Elastic Beanstalk 的組態檔案範例，請參閱範例：ElastiCache。

搭配 Amazon Elastic File System 使用 Elastic Beanstalk

您可以使用 Amazon Elastic File System (Amazon EFS) 來建立網路檔案系統，多個可用區域中的執行個體可掛載這些檔案系統。Amazon EFS 檔案系統是一種 AWS 資源，會針對您預設或自訂 VPC 中的網路，使用安全群組來控管對此等網路的存取。

在 Elastic Beanstalk 環境中，您可以使用 Amazon EFS 來建立共用的目錄，供您的應用程式儲存使用者上傳和修改的檔案。您的應用程式可以處理掛載的 Amazon EFS 磁碟區，例如本機儲存體。這樣一來，您就不需為了擴展成多個執行個體而變更應用程式的程式碼。

如需 Amazon EFS 的詳細資訊，請參閱 Amazon Elastic File System 使用者指南。

> **Note**
>
> Elastic Beanstalk 會建立一個 webapp 使用者，您可以將其設定為 Amazon EC2 執行個體上應用程式目錄的擁有者。如需詳細資訊，請參閱本指南的設計考量主題中的持久性儲存。

章節

組態檔案

Elastic Beanstalk 提供組態檔案，可用來建立和掛載 Amazon EFS 檔案系統。您可以建立 Amazon EFS 磁碟區來做為您環境的一部分，或是掛載您在 Elastic Beanstalk 以外另行建立的 Amazon EFS 磁碟區。

- storage-efs-createfilesystem.config - 使用 `Resources` 金鑰，來在 Amazon EFS 中建立新的檔案系統和掛載點。您環境中的所有執行個體，都可以連接到相同的檔案系統，來使用共用、可擴展的儲存空間。請使用 `storage-efs-mountfilesystem.config`，來在每個執行個體上掛載檔案系統。

  > **內部資源**
  >
  > 您使用組態檔案建立的任何資源都會與您環境的生命週期連結。如果您終止環境或移除組態檔案，就會失去這些資源。

- storage-efs-mountfilesystem.config - 將 Amazon EFS 檔案系統，掛載到您環境中執行個體上的本機路徑。您可以在環境中使用 `storage-efs-createfilesystem.config` 建立磁碟區。或者，您可以使用 Amazon EFS 主控台、AWS CLI 或 AWS SDK 將磁碟區掛載到您的環境。

若要使用組態檔案，請先利用 `storage-efs-createfilesystem.config` 來建立 Amazon EFS 檔案系統。遵循組態檔案中的指示，並將組態檔案加入您原始程式碼中的 .ebextensions 目錄，以在您的 VPC 中建立檔案系統。

將更新的原始碼部署到您的 Elastic Beanstalk 環境，以便確認檔案系統是否建立成功。然後，加入 `storage-efs-mountfilesystem.config`，以將檔案系統掛載到您環境中的執行個體。在兩個不同的部署中執行此操作，可確保掛載操作失敗時，檔案系統不會受到影響。如果在同一個部署中執行這兩項操作，則任何一個步驟所發生的問題，都會造成檔案系統在部署失敗時終止。

加密的檔案系統

Amazon EFS 支援加密的檔案系統。本主題中所討論的 `storage-efs-createfilesystem.config` 組態檔案定義了兩個自訂選項。您可以使用這些選項來建立 Amazon EFS 加密檔案系統。如需詳細資訊，請參閱組態檔案中的指示。

範例應用程式

Elastic Beanstalk 也提供範例應用程式，這些應用程式使用 Amazon EFS 來共用儲存。兩個專案包括可以搭配標準 WordPress 使用的組態檔案，或是 Drupal 安裝程式，此程式可在負載平衡的環境中，執

行部落格或其他內容管理系統。當使用者上傳相片或其他媒體時，檔案會儲存在 Amazon EFS 檔案系統中。這樣可以避免使用替代方案，也就是使用外掛程式將上傳的檔案存儲在 Amazon S3 中。

- Load-balanced WordPress - 包括組態檔案，用來安全地安裝 WordPress，並且在負載平衡的 Elastic Beanstalk 環境中執行。
- Load-balanced Drupal – 包括組態檔案與指示，用來安全地安裝 Drupal，並且在負載平衡的 Elastic Beanstalk 環境中執行。

清除檔案系統

如果您在 Elastic Beanstalk 環境中使用組態檔案建立 Amazon EFS 檔案系統，則當您終止環境時，Elastic Beanstalk 會移除檔案系統。為了盡可能降低執行中應用程式的儲存成本，請定期刪除您的應用程式不需要的檔案。或者，請確保應用程式的程式碼正確維護檔案生命週期。

> **Important**
>
> 如果您在 Elastic Beanstalk 環境外建立 Amazon EFS 檔案系統，並掛載到環境的執行個體，那麼 Elastic Beanstalk 並不會在您終止環境時移除檔案系統。為了確保不保留個人資訊和降低儲存成本，如果不再需要應用程式存放的檔案，請予以刪除。或者，您可以移除整個檔案系統。

使用 Elastic Beanstalk 搭配 AWS Identity and Access Management

AWS Identity and Access Management (IAM) 可協助您安全地控制 AWS 資源的存取。本節包含參考資料，說明使用 IAM 政策、執行個體描述檔和服務角色的相關資訊。

如需許可的概觀，請參閱服務角色、執行個體描述檔和使用者政策。對於大多數環境，在您啟動第一個環境時，由 Elastic Beanstalk 主控台提示您建立的服務角色和執行個體描述檔，會具有您需要的所有許可。同樣地，Elastic Beanstalk 所提供的完整存取和唯讀存取受管政策，也包含了日常使用需要的所有使用者許可。

IAM 使用者指南提供 AWS 許可的深入內容。

主題

- 管理 Elastic Beanstalk 執行個體描述檔
- 管理 Elastic Beanstalk 服務角色
- 使用 Elastic Beanstalk 的服務連結角色

- 管理 Elastic Beanstalk 使用者政策
- Elastic Beanstalk 的 Amazon Resource Name 格式
- Elastic Beanstalk 動作的資源與條件
- 使用標籤來控制對 Elastic Beanstalk 資源的存取
- 以受管政策為基礎的範例政策
- 根據資源許可的範例政策
- 防止跨環境 Amazon S3 儲存貯體存取

管理 Elastic Beanstalk 執行個體描述檔

執行個體設定檔是 AWS Identity and Access Management (IAM) 角色的容器，您可以在執行個體啟動時將角色資訊傳遞給 Amazon EC2 執行個體。

如果您的 AWS 帳戶沒有 EC2 執行個體設定檔，您必須使用 IAM 服務建立一個。然後，您可以將 EC2 執行個體設定檔指派給您建立的新環境。建立環境精靈提供可指導您完成 IAM 服務的資訊，以便您建立具有所需許可的 EC2 執行個體設定檔。建立執行個體設定檔後，您就可以返回主控台，將其選為 EC2 執行個體設定檔，然後繼續建立環境的步驟。

> **Note**
>
> 之前，Elastic Beanstalk 建立了一個預設的 EC2 執行個體設定檔，名為`aws-elasticbeanstalk-ec2-role` AWS 帳戶第一次建立環境時。此執行個體設定檔包含預設受管政策。如果您的帳戶已經擁有此執行個體設定檔，您仍然可以將其指派給您的環境。
> 不過，最近的 AWS 安全性準則不允許 AWS 服務自動建立具有信任政策的角色給其他 AWS 服務 (在這種情況下為 EC2)。基於這些安全性準則，Elastic Beanstalk 不再建立預設 `aws-elasticbeanstalk-ec2-role` 執行個體設定檔。

受管政策

Elastic Beanstalk 提供多種受管政策，讓您的環境能夠滿足不同的使用案例。若要符合環境的預設使用案例，這些政策必須附加至 EC2 執行個體設定檔的角色。

- AWSElasticBeanstalkWebTier— 授予應用程式將日誌上傳到 Amazon S3 並將資訊偵錯到的許可 AWS X-Ray。若要檢視受管理的策略內容，請參閱《AWS 受管策略參考指南》AWSElasticBeanstalkWebTier中的。

- AWSElasticBeanstalkWorkerTier— 授與記錄檔上傳、偵錯、指標發佈和背景工作執行個體工作的權限，包括佇列管理、領導者選舉和定期工作。若要檢視受管理的策略內容，請參閱《AWS 受管策略參考指南》AWSElasticBeanstalkWorkerTier中的。
- AWSElasticBeanstalkMulticontainerDocker— 授予 Amazon 彈性容器服務的許可，以協調 Docker 環境的叢集任務。若要檢視受管理的策略內容，請參閱《AWS 受管策略參考指南》AWSElasticBeanstalkMulticontainerDocker中的。

> **Important**
>
> Elastic Beanstalk 受管政策不提供精細許可，其會授予使用 Elastic Beanstalk 應用程式可能需要的所有許可。在某些情況下，您可能希望進一步限制我們託管策略的權限。如需一個使用案例的範例，請參閱防止跨環境 Amazon S3 儲存貯體存取。
> 此外，我們的受管政策也不涵蓋您可能新增至解決方案，並且不由 Elastic Beanstalk 管理的自訂資源許可。若要實作更精細的許可、最低必要許可或自訂資源許可，請使用自訂政策。

EC2 的信任關係政策

為了讓環境中的 EC2 執行個體能夠擔任必要的角色，執行個體設定檔必須在信任關係政策中，將 Amazon EC2 指定為受信任實體，如下所示。

```
{
  "Version": "2008-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

若要自訂許可，您可將政策新增至連接預設執行個體描述檔的角色，或自行建立具備一組受限許可的執行個體描述檔。

章節

建立執行個體描述檔

執行個體描述檔為標準 IAM 角色的包裝函式，可允許 EC2 執行個體擔任該角色。您可以建立其他執行個體設定檔來自訂不同應用程式的許可。或者，如果您不使用這些功能，則可以建立不授予工作者層或 ECS 受管 Docker 環境許可的執行個體設定檔。

建立執行個體設定檔

1. 在 IAM 主控台中開啟 Roles (角色) 頁面。
2. 選擇 建立角色。
3. 在受信任的實體類型下，選擇 AWS 服務。
4. 在 Use case (使用案例) 下，選擇 EC2。
5. 選擇下一步。
6. 連接 Elastic Beanstalk 提供的合適受管政策，以及為您的應用程式提供所需許可的其他政策。
7. 選擇下一步。
8. 輸入角色的名稱。
9. (選用) 附加標籤至角色。
10. 選擇建立角色。

驗證指派給執行個體設定檔的許可

指派給您預設執行個體描述檔的許可，可能會根據其建立的時間、上次啟動環境的時間和使用的用戶端，而有所不同。您可以在 IAM 主控台中，驗證預設執行個體描述檔上的許可。

若要驗證預設執行個體描述檔的許可

1. 在 IAM 主控台中開啟 Roles (角色) 頁面。
2. 選擇指派為 EC2 執行個體設定檔的角色。
3. 在 Permissions (許可) 索引標籤中，請檢閱角色連接的政策清單。
4. 若要查看政策所授予的許可，請選擇政策。

更新 out-of-date 預設執行個體設定檔

如果預設執行個體設定檔缺少必要的許可，您可手動將受管政策新增至指派為 EC2 執行個體設定檔的角色。

若要在連接至預設執行個體描述檔的角色中新增受管政策

1. 在 IAM 主控台中開啟 Roles (角色) 頁面。
2. 選擇指派為 EC2 執行個體設定檔的角色。
3. 在 Permissions (許可) 標籤上，選擇 Attach policies (連接政策)。
4. 輸入 **AWSElasticBeanstalk** 來篩選政策。
5. 選擇下列的政策，然後選擇 Attach policy (連接政策)：

   - `AWSElasticBeanstalkWebTier`
   - `AWSElasticBeanstalkWorkerTier`
   - `AWSElasticBeanstalkMulticontainerDocker`

於預設執行個體描述檔新增許可

如果您的應用程式存取預設執行個體設定檔中未授與許可的 AWS API 或資源，請在 IAM 主控台中新增授予權限的政策。

若要在連接至預設執行個體描述檔的角色中新增政策

1. 在 IAM 主控台中開啟 Roles (角色) 頁面。
2. 選擇指派為 EC2 執行個體設定檔的角色。
3. 在 Permissions (許可) 標籤上，選擇 Attach policies (連接政策)。
4. 為您的應用程式使用的其他服務，選取受管原則，例如 `AmazonS3FullAccess` 或 `AmazonDynamoDBFullAccess`。
5. 選擇連接政策。

管理 Elastic Beanstalk 服務角色

若要管理和監視您的環境，請代表您對環境資源 AWS Elastic Beanstalk 執行動作。Elastic Beanstalk 需要特定許可才能執行這些動作，並假定 AWS Identity and Access Management (IAM) 服務角色來取得這些許可。

Elastic Beanstalk 每次擔任服務角色時，都必須使用暫時性的安全登入資料。為取得這些憑證，Elastic Beanstalk 會傳送請求至區域專用端點上的 AWS Security Token Service (AWS STS)。如需詳細資訊，請參閱《IAM 使用者指南》中的臨時安全登入資料。

> **Note**
>
> 如果您的環境所在區域的 AWS STS 端點已停用，Elastic Beanstalk 會在無法停用的替代端點上傳送請求。此端點與其他區域相關聯。因此，該請求是跨區域請求。如需詳細資訊，請參閱 IAM 使用者指南AWS STS 中的在 AWS 區域中啟用和停用。

使用 Elastic Beanstalk 主控台和 EB CLI 管理服務角色

透過一組足夠的許可，您可以使用 Elastic Beanstalk 主控台和 EB CLI，為您的環境設定服務角色。它們會建立預設服務角色並在其中使用受管政策。

受管服務角色政策

Elastic Beanstalk 提供了一個用於增強型運作狀態監控的受管政策，以及另一個具備受管平台更新所需額外許可的受管政策。主控台和 EB CLI 會將這兩種政策指派給其為您建立的預設服務角色。這些政策應該僅用於此預設服務角色。它們不應該與您帳戶中的其他使用者或角色搭配使用。

`AWSElasticBeanstalkEnhancedHealth`

此政策會授予許可，讓 Elastic Beanstalk 能夠監控執行個體和環境運作狀態。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:DescribeInstanceHealth",
                "elasticloadbalancing:DescribeLoadBalancers",
                "elasticloadbalancing:DescribeTargetHealth",
                "ec2:DescribeInstances",
                "ec2:DescribeInstanceStatus",
                "ec2:GetConsoleOutput",
                "ec2:AssociateAddress",
                "ec2:DescribeAddresses",
                "ec2:DescribeSecurityGroups",
                "sqs:GetQueueAttributes",
```

```
                "sqs:GetQueueUrl",
                "autoscaling:DescribeAutoScalingGroups",
                "autoscaling:DescribeAutoScalingInstances",
                "autoscaling:DescribeScalingActivities",
                "autoscaling:DescribeNotificationConfigurations",
                "sns:Publish"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

AWSElasticBeanstalkManagedUpdatesCustomerRolePolicy

此政策會授予許可，讓 Elastic Beanstalk 能夠代表您更新環境以執行受管平台更新。

服務層級許可分組

此政策會根據提供的許可集分組到陳述式中。

- *ElasticBeanstalkPermissions* – 此許可群組用於呼叫 Elastic Beanstalk 服務動作 (Elastic Beanstalk API)。
- *AllowPassRoleToElasticBeanstalkAndDownstreamServices* – 此許可群組允許將任何角色傳遞給 Elastic Beanstalk 和其他下游服務，如 AWS CloudFormation。
- *ReadOnlyPermissions* – 此許可群組用於收集執行中環境的相關資訊。
- **OperationPermissions* – 具有此命名模式的群組用於呼叫必要的操作來執行平台更新。
- **BroadOperationPermissions* – 具有此命名模式的群組用於呼叫必要的操作來執行平台更新。它們也包含支援舊式環境的廣泛許可。
- **TagResource*— 具有此命名模式的群組適用於使用 tag-on-create API 在 Elastic Beanstalk 環境中建立的資源上附加標籤的呼叫。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ElasticBeanstalkPermissions",
            "Effect": "Allow",
            "Action": [
```

```
                "elasticbeanstalk:*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowPassRoleToElasticBeanstalkAndDownstreamServices",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::*:role/*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "elasticbeanstalk.amazonaws.com",
                        "ec2.amazonaws.com",
                        "ec2.amazonaws.com.cn",
                        "autoscaling.amazonaws.com",
                        "elasticloadbalancing.amazonaws.com",
                        "ecs.amazonaws.com",
                        "cloudformation.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Sid": "ReadOnlyPermissions",
            "Effect": "Allow",
            "Action": [
                "autoscaling:DescribeAccountLimits",
                "autoscaling:DescribeAutoScalingGroups",
                "autoscaling:DescribeAutoScalingInstances",
                "autoscaling:DescribeLaunchConfigurations",
                "autoscaling:DescribeLoadBalancers",
                "autoscaling:DescribeNotificationConfigurations",
                "autoscaling:DescribeScalingActivities",
                "autoscaling:DescribeScheduledActions",
                "ec2:DescribeAccountAttributes",
                "ec2:DescribeAddresses",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeImages",
                "ec2:DescribeInstanceAttribute",
                "ec2:DescribeInstances",
                "ec2:DescribeKeyPairs",
                "ec2:DescribeLaunchTemplates",
                "ec2:DescribeLaunchTemplateVersions",
```

```
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSnapshots",
                "ec2:DescribeSpotInstanceRequests",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcClassicLink",
                "ec2:DescribeVpcs",
                "elasticloadbalancing:DescribeInstanceHealth",
                "elasticloadbalancing:DescribeLoadBalancers",
                "elasticloadbalancing:DescribeTargetGroups",
                "elasticloadbalancing:DescribeTargetHealth",
                "logs:DescribeLogGroups",
                "rds:DescribeDBEngineVersions",
                "rds:DescribeDBInstances",
                "rds:DescribeOrderableDBInstanceOptions",
                "sns:ListSubscriptionsByTopic"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "EC2BroadOperationPermissions",
            "Effect": "Allow",
            "Action": [
                "ec2:AllocateAddress",
                "ec2:AssociateAddress",
                "ec2:AuthorizeSecurityGroupEgress",
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:CreateLaunchTemplate",
                "ec2:CreateLaunchTemplateVersion",
                "ec2:CreateSecurityGroup",
                "ec2:DeleteLaunchTemplate",
                "ec2:DeleteLaunchTemplateVersions",
                "ec2:DeleteSecurityGroup",
                "ec2:DisassociateAddress",
                "ec2:ReleaseAddress",
                "ec2:RevokeSecurityGroupEgress",
                "ec2:RevokeSecurityGroupIngress"
            ],
            "Resource": "*"
        },
        {
            "Sid": "EC2RunInstancesOperationPermissions",
            "Effect": "Allow",
```

```
            "Action": "ec2:RunInstances",
            "Resource": "*",
            "Condition": {
                "ArnLike": {
                    "ec2:LaunchTemplate": "arn:aws:ec2:*:*:launch-template/*"
                }
            }
        },
        {
            "Sid": "EC2TerminateInstancesOperationPermissions",
            "Effect": "Allow",
            "Action": [
                "ec2:TerminateInstances"
            ],
            "Resource": "arn:aws:ec2:*:*:instance/*",
            "Condition": {
                "StringLike": {
                    "ec2:ResourceTag/aws:cloudformation:stack-id": [
                        "arn:aws:cloudformation:*:*:stack/awseb-e-*",
                        "arn:aws:cloudformation:*:*:stack/eb-*"
                    ]
                }
            }
        },
        {
            "Sid": "ECSBroadOperationPermissions",
            "Effect": "Allow",
            "Action": [
                "ecs:CreateCluster",
                "ecs:DescribeClusters",
                "ecs:RegisterTaskDefinition"
            ],
            "Resource": "*"
        },
        {
            "Sid": "ECSDeleteClusterOperationPermissions",
            "Effect": "Allow",
            "Action": "ecs:DeleteCluster",
            "Resource": "arn:aws:ecs:*:*:cluster/awseb-*"
        },
        {
            "Sid": "ASGOperationPermissions",
            "Effect": "Allow",
            "Action": [
```

```
                "autoscaling:AttachInstances",
                "autoscaling:CreateAutoScalingGroup",
                "autoscaling:CreateLaunchConfiguration",
                "autoscaling:CreateOrUpdateTags",
                "autoscaling:DeleteLaunchConfiguration",
                "autoscaling:DeleteAutoScalingGroup",
                "autoscaling:DeleteScheduledAction",
                "autoscaling:DetachInstances",
                "autoscaling:DeletePolicy",
                "autoscaling:PutScalingPolicy",
                "autoscaling:PutScheduledUpdateGroupAction",
                "autoscaling:PutNotificationConfiguration",
                "autoscaling:ResumeProcesses",
                "autoscaling:SetDesiredCapacity",
                "autoscaling:SuspendProcesses",
                "autoscaling:TerminateInstanceInAutoScalingGroup",
                "autoscaling:UpdateAutoScalingGroup"
            ],
            "Resource": [
                "arn:aws:autoscaling:*:*:launchConfiguration:*:launchConfigurationName/
awseb-e-*",
                "arn:aws:autoscaling:*:*:launchConfiguration:*:launchConfigurationName/
eb-*",
                "arn:aws:autoscaling:*:*:autoScalingGroup:*:autoScalingGroupName/awseb-
e-*",
                "arn:aws:autoscaling:*:*:autoScalingGroup:*:autoScalingGroupName/eb-*"
            ]
        },
        {
            "Sid": "CFNOperationPermissions",
            "Effect": "Allow",
            "Action": [
                "cloudformation:*"
            ],
            "Resource": [
                "arn:aws:cloudformation:*:*:stack/awseb-*",
                "arn:aws:cloudformation:*:*:stack/eb-*"
            ]
        },
        {
            "Sid": "ELBOperationPermissions",
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:AddTags",
```

```
                "elasticloadbalancing:ApplySecurityGroupsToLoadBalancer",
                "elasticloadbalancing:ConfigureHealthCheck",
                "elasticloadbalancing:CreateLoadBalancer",
                "elasticloadbalancing:DeleteLoadBalancer",
                "elasticloadbalancing:DeregisterInstancesFromLoadBalancer",
                "elasticloadbalancing:DeregisterTargets",
                "elasticloadbalancing:RegisterInstancesWithLoadBalancer",
                "elasticloadbalancing:RegisterTargets"
            ],
            "Resource": [
                "arn:aws:elasticloadbalancing:*:*:targetgroup/awseb-*",
                "arn:aws:elasticloadbalancing:*:*:targetgroup/eb-*",
                "arn:aws:elasticloadbalancing:*:*:loadbalancer/awseb-*",
                "arn:aws:elasticloadbalancing:*:*:loadbalancer/eb-*",
                "arn:aws:elasticloadbalancing:*:*:loadbalancer/*/awseb-*/*",
                "arn:aws:elasticloadbalancing:*:*:loadbalancer/*/eb-*/*"
            ]
        },
        {
            "Sid": "CWLogsOperationPermissions",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:DeleteLogGroup",
                "logs:PutRetentionPolicy"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:/aws/elasticbeanstalk/*"
        },
        {
            "Sid": "S3ObjectOperationPermissions",
            "Effect": "Allow",
            "Action": [
                "s3:DeleteObject",
                "s3:GetObject",
                "s3:GetObjectAcl",
                "s3:GetObjectVersion",
                "s3:GetObjectVersionAcl",
                "s3:PutObject",
                "s3:PutObjectAcl",
                "s3:PutObjectVersionAcl"
            ],
            "Resource": "arn:aws:s3:::elasticbeanstalk-*/*"
        },
        {
```

```
            "Sid": "S3BucketOperationPermissions",
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:GetBucketPolicy",
                "s3:ListBucket",
                "s3:PutBucketPolicy"
            ],
            "Resource": "arn:aws:s3:::elasticbeanstalk-*"
        },
        {
            "Sid": "SNSOperationPermissions",
            "Effect": "Allow",
            "Action": [
                "sns:CreateTopic",
                "sns:GetTopicAttributes",
                "sns:SetTopicAttributes",
                "sns:Subscribe"
            ],
            "Resource": "arn:aws:sns:*:*:ElasticBeanstalkNotifications-*"
        },
        {
            "Sid": "SQSOperationPermissions",
            "Effect": "Allow",
            "Action": [
                "sqs:GetQueueAttributes",
                "sqs:GetQueueUrl"
            ],
            "Resource": [
                "arn:aws:sqs:*:*:awseb-e-*",
                "arn:aws:sqs:*:*:eb-*"
            ]
        },
        {
            "Sid": "CWPutMetricAlarmOperationPermissions",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricAlarm"
            ],
            "Resource": [
                "arn:aws:cloudwatch:*:*:alarm:awseb-*",
                "arn:aws:cloudwatch:*:*:alarm:eb-*"
            ]
        },
```

```
        {
            "Sid": "AllowECSTagResource",
            "Effect": "Allow",
            "Action": [
                "ecs:TagResource"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "ecs:CreateAction": [
                        "CreateCluster",
                        "RegisterTaskDefinition"
                    ]
                }
            }
        }
    ]
}
```

若要檢視受管政策的內容，您也可以使用 IAM 主控台中的 Policies (政策) 頁面。

> **Important**
>
> Elastic Beanstalk 受管政策不提供精細許可，其會授予使用 Elastic Beanstalk 應用程式可能需要的所有許可。在某些情況下，您可能希望進一步限制我們託管策略的權限。如需一個使用案例的範例，請參閱防止跨環境 Amazon S3 儲存貯體存取。
> 此外，我們的受管政策也不涵蓋您可能新增至解決方案，並且不由 Elastic Beanstalk 管理的自訂資源許可。若要實作更精細的許可、最低必要許可或自訂資源許可，請使用自訂政策。

已廢除的 受管政策

在過去，Elastic Beanstalk 支援AWSElasticBeanstalkService受管理服務角色原則。此政策已被取代AWSElasticBeanstalkManagedUpdatesCustomerRolePolicy。您可能仍然可以在 IAM 主控台中查看和使用先前的政策。

若要檢視受管理的策略內容，請參閱《AWS 受管策略參考指南》AWSElasticBeanstalkService中的。

但是，我們建議您轉換為使用新的受管理策略 (AWSElasticBeanstalkManagedUpdatesCustomerRolePolicy)。新增自訂政策以授予自訂資源的許可 (如果您有)。

使用 Elastic Beanstalk 主控台

當您在 Elastic Beanstalk 主控台啟動環境時，主控台會建立名為 aws-elasticbeanstalk-service-role 的預設服務角色，然後將具有預設許可的受管政策連接至該服務角色。

為了讓 Elastic Beanstalk 能夠擔任 aws-elasticbeanstalk-service-role 角色，服務角色會在信任關係政策中，將 Elastic Beanstalk 指定為信任實體。

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

當您為環境啟用受管平台更新時，Elastic Beanstalk 會擔任另一個受管更新服務角色以執行受管更新。根據預設，Elastic Beanstalk 主控台使用同一個產生的服務角色 aws-elasticbeanstalk-service-role 做為受管更新服務角色。如果您變更預設服務角色，主控台會將受管更新服務角色設定為使用受管更新服務連結角色：AWSServiceRoleForElasticBeanstalkManagedUpdates。如需服務連結角色的詳細資訊，請參閱the section called “使用服務連結角色”。

> **Note**
>
> 由於許可問題，Elastic Beanstalk 服務不一定能夠成功為您建立此服務連結角色。因此，主控台會嘗試明確地建立。為確保您的帳戶具有此服務連結角色，請至少在使用主控台後建立環境，然後將受管更新設定為在建立環境前啟用。

使用 EB CLI

如果您使用 Elastic Beanstalk 命令列界面 (EB CLI) 的 the section called “eb create” 命令啟動環境，而且不透過 `--service-role` 選項指定服務角色，Elastic Beanstalk 會建立預設的服務角色 `aws-elasticbeanstalk-service-role`。若預設服務角色已存在，Elastic Beanstalk 會將其運用於新環境。在這些情況下，Elastic Beanstalk 主控台也會執行類似的動作。

與主控台不同的是，您無法使用 EB CLI 命令選項指定受管更新服務角色。如果您為環境啟用受管更新，您必須透過組態選項設定受管更新服務角色。下列範例會啟用受管更新，並使用預設服務角色以做為受管更新服務角色。

Example .ebextension/. managed-platform-update 配置

```
option_settings:
  aws:elasticbeanstalk:managedactions:
    ManagedActionsEnabled: true
    PreferredStartTime: "Tue:09:00"
    ServiceRoleForManagedUpdates: "aws-elasticbeanstalk-service-role"
  aws:elasticbeanstalk:managedactions:platformupdate:
    UpdateLevel: patch
    InstanceRefreshEnabled: true
```

使用 Elastic Beanstalk API 管理服務角色

當您使用 Elastic Beanstalk API 的 `CreateEnvironment` 動作建立環境時，請使用 `aws:elasticbeanstalk:environment` 命名空間中的 `ServiceRole` 組態選項來指定服務角色。如需有關搭配 Elastic Beanstalk API 使用增強型運作狀態監視的詳細資訊，請參閱將增強型運作狀態報告與 Elastic Beanstalk API 搭配使用 。

此外，如果您為環境啟用受管平台更新，您可以使用 `aws:elasticbeanstalk:managedactions` 命名空間的 `ServiceRoleForManagedUpdates` 選項指定受管更新服務角色。

使用服務連結角色

服務連結角色是一種唯一的服務角色類型，由 Elastic Beanstalk 預先定義，用於包含服務代表您呼叫其他 AWS 服務所需的所有權限。服務連結角色會您的帳戶建立關聯。Elastic Beanstalk 只會建立一次此角色，然後在建立其他環境時重複使用。如需透過 Elastic Beanstalk 環境使用服務連結角色的詳細資訊，請參閱使用 Elastic Beanstalk 的服務連結角色。

如果您使用 Elastic Beanstalk API 建立環境，且未指定服務角色，Elastic Beanstalk 會為您的帳戶建立監控服務連結角色 (如果該角色不存在)。Elastic Beanstalk 會將此角色用於新的環境。您也可以使

用 IAM，預先建立您帳戶的監控服務連結角色。在您的帳戶擁有此角色之後，您可以使用它來建立使用 Elastic Beanstalk API、Elastic Beanstalk 主控台或 EB CLI 的環境。

如果您啟用環境的受管平台更新，並指定 `AWSServiceRoleForElasticBeanstalkManagedUpdates` 為 `aws:elasticbeanstalk:managedactions` 命名空間 `ServiceRoleForManagedUpdates` 選項的值，Elastic Beanstalk 會為您的帳戶建立受管更新服務連結角色 (如果該角色不存在)。Elastic Beanstalk 使用此角色為新環境執行受管更新。

> **Note**
>
> 當您建立環境時，若 Elastic Beanstalk 嘗試為您的帳戶建立監控和受管更新服務連結角色，您必須擁有 `iam:CreateServiceLinkedRole` 許可。如果您沒有此許可，環境建立會失敗，並顯示一則說明問題的訊息。
> 或者，具有建立服務連結角色許可的另一個使用者可用 IAM 預先建立服務連結角色。使用此方法，您不需要 `iam:CreateServiceLinkedRole` 許可即可建立環境。

驗證預設服務角色許可

由您的預設服務角色所授予的許可，可能會基於角色建立的時間、上次啟動環境的時間和使用的用戶端，而有所不同。在 IAM 主控台中，您可以驗證預設服務角色所授予的許可。

若要驗證預設服務角色的許可

1. 在 IAM 主控台中，開啟 Roles (角色) 頁面。
2. 選擇aws-elasticbeanstalk-service-role。
3. 在 Permissions (許可) 索引標籤中，請檢閱角色連接的政策清單。
4. 若要檢視政策所授予的許可，請選擇政策。

更新 out-of-date 預設服務角色

如果預設的服務角色缺少所需的許可，您可以藉由在 Elastic Beanstalk 環境的管理主控台中建立新環境，來更新該角色的許可。

或者，您可以透過手動方式，將受管政策新增至預設的服務角色。

若要受管政策新增到預設的服務角色

1. 在 IAM 主控台中，開啟 Roles (角色) 頁面。
2. 選擇aws-elasticbeanstalk-service-role。
3. 在 Permissions (許可) 標籤上，選擇 Attach policies (連接政策)。
4. 輸入 **AWSElasticBeanstalk** 來篩選政策。
5. 選擇下列的政策，然後選擇 Attach policy (連接政策)：

   - `AWSElasticBeanstalkEnhancedHealth`
   - `AWSElasticBeanstalkManagedUpdatesCustomerRolePolicy`

新增預設服務角色的許可

如果您的應用程式包含參照權限未包含在預設服務角色中的 AWS 資源的設定檔，Elastic Beanstalk 可能需要其他權限。當它在受管更新期間處理組態檔案時，需要這些額外的許可來解析這些引用。如果缺少許可，則更新會失敗，而且 Elastic Beanstalk 會傳回訊息，指示所需的許可。請依照下列步驟將其他服務的許可新增至 IAM 主控台中的預設服務角色。

若要針對預設服務角色新增額外的政策

1. 在 IAM 主控台中，開啟 Roles (角色) 頁面。
2. 選擇aws-elasticbeanstalk-service-role。
3. 在 Permissions (許可) 標籤上，選擇 Attach policies (連接政策)。
4. 為您的應用程式使用的其他服務，選取受管原則，例如 `AmazonAPIGatewayAdministrator` 或 `AmazonElasticFileSystemFullAccess`。
5. 選擇連接政策。

建立服務角色

如果您無法使用預設服務角色，請建立服務角色。

若要建立服務角色

1. 在 IAM 主控台中，開啟 Roles (角色) 頁面。
2. 選擇建立角色。
3. 在 AWS service (AWS 服務) 之下選擇 AWS Elastic Beanstalk，然後選取您的使用案例。

4. 選擇下一步：許可。
5. 連接 AWSElasticBeanstalkManagedUpdatesCustomerRolePolicy 與 AWSElasticBeanstalkEnhancedHealth 受管政策，以及提供您應用程式所需許可的其他任何政策。
6. 選擇下一步：標籤。
7. (選用) 附加標籤至角色。
8. 選擇下一步：檢閱。
9. 輸入角色的名稱。
10. 選擇建立角色。

當您利用環境建立精靈或 eb create 指令中的 --service-role 選項來建立環境時，請套用自訂的服務角色。

使用 Elastic Beanstalk 的服務連結角色

AWS Elastic Beanstalk 使用 AWS Identity and Access Management (IAM) 服務連結角色。服務連結角色是直接連結至 Elastic Beanstalk 的一種獨特 IAM 角色類型。服務連結角色由 Elastic Beanstalk 預先定義，並包含服務代表您呼叫其他 AWS 服務所需的所有權限。

Elastic Beanstalk 定義了幾種類型的服務連結角色：

- 「監控服務連結角色」- 可讓 Elastic Beanstalk 監控執行中環境的運作狀態，並發佈運作狀態事件通知。
- 「維護服務連結角色」- 可讓 Elastic Beanstalk 為執行中環境執行定期維護活動。
- 受管理的更新服務連結角色 - 讓 Elastic Beanstalk 能執行您執行環境中排定的平台更新。

主題

- 監控服務連結角色
- 維護服務連結角色
- 受管更新服務連結角色

監控服務連結角色

AWS Elastic Beanstalk 使用 AWS Identity and Access Management (IAM) 服務連結角色。服務連結角色是直接連結至 Elastic Beanstalk 的一種獨特 IAM 角色類型。服務連結角色由 Elastic Beanstalk 預先定義，並包含服務代表您呼叫其他 AWS 服務所需的所有權限。

服務連結角色讓設定 Elastic Beanstalk 變得更簡單，因為您不必手動新增必要的許可。Elastic Beanstalk 定義其服務連結角色的許可，除非另有定義，否則僅有 Elastic Beanstalk 可以擔任其角色。定義的許可包括信任政策和許可政策，且該許可政策無法附加至其他 IAM 實體。

您必須先刪除服務連結角色的相關資源，才能將其刪除。這可保護您的 Elastic Beanstalk 資源，避免您不小心移除資源的存取許可。

如需關於支援服務連結角色的其他服務的資訊，請參閱可搭配 IAM 運作的AWS 服務，並尋找 Service-Linked Role (服務連結角色) 欄顯示為 Yes (是) 的服務。選擇具有連結的是，以檢視該服務的服務連結角色文件。

Elastic Beanstalk 的服務連結角色許可

Elastic Beanstalk 使用名為的服務連結角色 AWSServiceRoleForElasticBeanstalk— 允許 Elastic Beanstalk 監視執行環境的健全狀況，並發佈健康事件通知。

服 AWSServiceRoleForElasticBeanstalk 務連結角色會信任下列服務擔任該角色：

- `elasticbeanstalk.amazonaws.com`

AWSServiceRoleForElasticBeanstalk 服務連結角色的權限原則包含 Elastic Beanstalk 代表您完成動作所需的所有權限：

AllowCloudformationReadOperationsOnElasticBeanstalkStacks

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCloudformationReadOperationsOnElasticBeanstalkStacks",
            "Effect": "Allow",
            "Action": [
                "cloudformation:DescribeStackResource",
                "cloudformation:DescribeStackResources",
                "cloudformation:DescribeStacks"
            ],
```

```
            "Resource": [
                "arn:aws:cloudformation:*:*:stack/awseb-*",
                "arn:aws:cloudformation:*:*:stack/eb-*"
            ]
        },
        {
            "Sid": "AllowOperations",
            "Effect": "Allow",
            "Action": [
                "autoscaling:DescribeAutoScalingGroups",
                "autoscaling:DescribeAutoScalingInstances",
                "autoscaling:DescribeNotificationConfigurations",
                "autoscaling:DescribeScalingActivities",
                "autoscaling:PutNotificationConfiguration",
                "ec2:DescribeInstanceStatus",
                "ec2:AssociateAddress",
                "ec2:DescribeAddresses",
                "ec2:DescribeInstances",
                "ec2:DescribeSecurityGroups",
                "elasticloadbalancing:DescribeInstanceHealth",
                "elasticloadbalancing:DescribeLoadBalancers",
                "elasticloadbalancing:DescribeTargetHealth",
                "elasticloadbalancing:DescribeTargetGroups",
                "sqs:GetQueueAttributes",
                "sqs:GetQueueUrl",
                "sns:Publish"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

您必須設定許可，IAM 實體 (如使用者、群組或角色) 才可建立、編輯或刪除服務連結角色。如需詳細資訊，請參閱《IAM 使用者指南》中的服務連結角色許可。

或者，您可以使用 AWS 受管政策來提供對 Elastic Beanstalk 的完整存取權。

為 Elastic Beanstalk 建立服務連結角色

您不需要手動建立一個服務連結角色。當您使用 Elastic Beanstalk API 建立 Elastic Beanstalk 環境，但未指定服務角色時，Elastic Beanstalk 會為您建立服務連結角色。

> **Important**
>
> 如果您在 2017 年 9 月 27 日之前使用 Elastic Beanstalk 服務，當它開始支援服務 AWSServiceRoleForElasticBeanstalk 連結角色時，且您的帳戶需要該服務，則 Elastic Beanstalk 會在您的帳戶中建立該 AWSServiceRoleForElasticBeanstalk 角色。若要進一步了解，請參閱我的 IAM 帳戶中出現的新角色。

當您建立環境時，Elastic Beanstalk 嘗試為您的帳戶建立 AWSServiceRoleForElasticBeanstalk 服務連結角色時，您必須擁有該權限。`iam:CreateServiceLinkedRole`如果您沒有此許可，環境建立會失敗，且您會看到一則說明問題的訊息。

或者，具有建立服務連結角色許可的另一個使用者可用 IAM 來預先建立服務連結角色。然後，即使沒有 `iam:CreateServiceLinkedRole` 權限，您也可以建立您的環境。

您 (或其他使用者) 可以使用 IAM 主控台透過 Elastic Beanstalk 使用案例建立服務連結角色。在 IAM CLI 或 IAM API 中，建立一個使用 `elasticbeanstalk.amazonaws.com` 服務名稱的服務連結角色。如需詳細資訊，請參閱 IAM 使用者指南中的建立服務連結角色。如果您刪除此服務連結角色，您可以使用此相同的程序以再次建立該角色。

若您刪除此服務連結角色，之後需要再次建立，您可以在帳戶中使用相同程序重新建立角色。當您使用 Elastic Beanstalk API 建立 Elastic Beanstalk 環境，但未指定服務角色時，Elastic Beanstalk 會再次為您建立服務連結角色。

編輯 Elastic Beanstalk 的服務連結角色

Elastic Beanstalk 不允許您編輯 AWSServiceRoleForElasticBeanstalk 服務連結角色。因為有各種實體可能會參考服務連結角色，所以您無法在建立角色之後變更角色名稱。然而，您可使用 IAM 來編輯角色描述。如需更多資訊，請參閱 IAM 使用者指南中的編輯服務連結角色。

刪除 Elastic Beanstalk 的服務連結角色

若您不再使用需要服務連結角色的功能或服務，我們建議您刪除該角色。如此一來，您就沒有未主動監控或維護的未使用實體。然而，在手動刪除服務連結角色之前，您必須先清除資源。

清除服務連結角色

您必須先確保所有 Elastic Beanstalk 環境皆使用不同的服務角色或已終止，才能使用 IAM 刪除服務連結角色。

> **Note**
>
> 當您嘗試終止環境時，如果 Elastic Beanstalk 服務仍在使用該服務連結角色，則終止可能會失敗。若此情況發生，請等待數分鐘後並再次嘗試操作。

終止使用 AWSServiceRoleForElasticBeanstalk （控制台）的 Elastic Beanstalk 環境

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 選擇 Actions (動作)，然後選擇 Terminate Environment (終止環境)。
4. 使用畫面顯示對話方塊來確認環境終止。

如需使用 EB CLI 終止 Elastic Beanstalk 環境的詳細資訊，請參閱eb terminate。

TerminateEnvironment如需有關使用 API 終止 Elastic Beanstalk 環境的詳細資訊，請參閱。

手動刪除服務連結角色

使用 IAM 主控台、IAM CLI 或 IAM API 刪除 AWSServiceRoleForElasticBeanstalk 服務連結角色。如需詳細資訊，請參閱《IAM 使用者指南》中的刪除服務連結角色。

Elastic Beanstalk 服務連結角色的支援區域

Elastic Beanstalk 在所有提供服務的區域中都支援使用服務連結角色。如需詳細資訊，請參閱 AWS Elastic Beanstalk 端點和配額。

維護服務連結角色

AWS Elastic Beanstalk 使用 AWS Identity and Access Management (IAM) 服務連結角色。服務連結角色是直接連結至 Elastic Beanstalk 的一種獨特 IAM 角色類型。服務連結角色由 Elastic Beanstalk 預先定義，並包含服務代表您呼叫其他 AWS 服務所需的所有權限。

服務連結角色讓設定 Elastic Beanstalk 變得更簡單，因為您不必手動新增必要的許可。Elastic Beanstalk 定義其服務連結角色的許可，除非另有定義，否則僅有 Elastic Beanstalk 可以擔任其角色。定義的許可包括信任政策和許可政策，且該許可政策無法附加至其他 IAM 實體。

您必須先刪除服務連結角色的相關資源，才能將其刪除。這可保護您的 Elastic Beanstalk 資源，避免您不小心移除資源的存取許可。

如需關於支援服務連結角色的其他服務的資訊，請參閱可搭配 IAM 運作的AWS 服務，並尋找 Service-Linked Role (服務連結角色) 欄顯示為 Yes (是) 的服務。選擇具有連結的是，以檢視該服務的服務連結角色文件。

Elastic Beanstalk 的服務連結角色許可

Elastic Beanstalk 使用名為的服務連結角色 AWSServiceRoleForElasticBeanstalkMaintenance— 允許 Elastic Beanstalk 針對您的執行環境執行定期維護活動。

服 AWSServiceRoleForElasticBeanstalkMaintenance 務連結角色會信任下列服務擔任該角色：

- `maintenance.elasticbeanstalk.amazonaws.com`

AWSServiceRoleForElasticBeanstalkMaintenance 服務連結角色的權限原則包含 Elastic Beanstalk 代表您完成動作所需的所有權限：

```
{
    "Version": "2012-10-17",
    "Statement":
        {
            "Sid": "AllowCloudformationChangeSetOperationsOnElasticBeanstalkStacks",
            "Effect": "Allow",
            "Action": [
                "cloudformation:CreateChangeSet",
                "cloudformation:DescribeChangeSet",
                "cloudformation:ExecuteChangeSet",
                "cloudformation:DeleteChangeSet",
                "cloudformation:ListChangeSets",
                "cloudformation:DescribeStacks"
            ],
            "Resource": [
                "arn:aws:cloudformation:*:*:stack/awseb-*",
                "arn:aws:cloudformation:*:*:stack/eb-*"
            ]
        }
```

```
}
```

您必須設定許可，IAM 實體 (如使用者、群組或角色) 才可建立、編輯或刪除服務連結角色。如需詳細資訊，請參閱《IAM 使用者指南》中的服務連結角色許可。

或者，您可以使用 AWS 受管政策來提供對 Elastic Beanstalk 的完整存取權。

為 Elastic Beanstalk 建立服務連結角色

您不需要手動建立一個服務連結角色。當您使用 Elastic Beanstalk API 建立 Elastic Beanstalk 環境，但未指定執行個體描述檔時，Elastic Beanstalk 會為您建立服務連結角色。

> **Important**
>
> 此服務連結角色可以顯示在您的帳戶，如果您於其他服務中完成一項動作時，可以使用支援此角色的功能。如果您在 2019 年 4 月 18 日之前使用 Elastic Beanstalk 服務，那麼當它開始支援 AWSServiceRoleForElasticBeanstalkMaintenance 服務連結角色時，且您的帳戶需要它，則 Elastic Beanstalk 會在您的帳戶中建立該 AWSServiceRoleForElasticBeanstalkMaintenance 角色。若要進一步了解，請參閱我的 IAM 帳戶中出現的新角色。

若您刪除此服務連結角色，之後需要再次建立，您可以在帳戶中使用相同程序重新建立角色。當您使用 Elastic Beanstalk API 建立 Elastic Beanstalk 環境，但未指定執行個體描述檔時，Elastic Beanstalk 會再次為您建立服務連結角色。

編輯 Elastic Beanstalk 的服務連結角色

Elastic Beanstalk 不允許您編輯 AWSServiceRoleForElasticBeanstalkMaintenance 服務連結角色。因為有各種實體可能會參考服務連結角色，所以您無法在建立角色之後變更角色名稱。然而，您可使用 IAM 來編輯角色描述。如需更多資訊，請參閱 IAM 使用者指南中的編輯服務連結角色。

刪除 Elastic Beanstalk 的服務連結角色

若您不再使用需要服務連結角色的功能或服務，我們建議您刪除該角色。如此一來，您就沒有未主動監控或維護的未使用實體。然而，在手動刪除服務連結角色之前，您必須先清除資源。

清除服務連結角色

您必須先終止使用服務連結角色的任何 Elastic Beanstalk 環境，才能使用 IAM 刪除該服務連結角色。

> **Note**
>
> 當您嘗試終止環境時，如果 Elastic Beanstalk 服務仍在使用該服務連結角色，則終止可能會失敗。若此情況發生，請等待數分鐘後並再次嘗試操作。

終止使用 AWSServiceRoleForElasticBeanstalkMaintenance （控制台）的 Elastic Beanstalk 環境

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域
2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 選擇 Actions (動作)，然後選擇 Terminate Environment (終止環境)。
4. 使用畫面顯示對話方塊來確認環境終止。

如需使用 EB CLI 終止 Elastic Beanstalk 環境的詳細資訊，請參閱eb terminate。

TerminateEnvironment如需有關使用 API 終止 Elastic Beanstalk 環境的詳細資訊，請參閱。

手動刪除服務連結角色

使用 IAM 主控台、IAM CLI 或 IAM API 刪除 AWSServiceRoleForElasticBeanstalkMaintenance 服務連結角色。如需詳細資訊，請參閱《IAM 使用者指南》中的刪除服務連結角色。

Elastic Beanstalk 服務連結角色的支援區域

Elastic Beanstalk 在所有提供服務的區域中都支援使用服務連結角色。如需詳細資訊，請參閱 AWS Elastic Beanstalk 端點和配額。

受管更新服務連結角色

AWS Elastic Beanstalk 使用 AWS Identity and Access Management (IAM) 服務連結角色。服務連結角色是直接連結至 Elastic Beanstalk 的一種獨特 IAM 角色類型。服務連結角色由 Elastic Beanstalk 預先定義，並包含服務代表您呼叫其他 AWS 服務所需的所有權限。

服務連結角色讓設定 Elastic Beanstalk 變得更簡單，因為您不必手動新增必要的許可。Elastic Beanstalk 定義其服務連結角色的許可，除非另有定義，否則僅有 Elastic Beanstalk 可以擔任其角色。定義的許可包括信任政策和許可政策，且該許可政策無法附加至其他 IAM 實體。

您必須先刪除服務連結角色的相關資源，才能將其刪除。這可保護您的 Elastic Beanstalk 資源，避免您不小心移除資源的存取許可。

如需關於支援服務連結角色的其他服務的資訊，請參閱可搭配 IAM 運作的AWS 服務，並尋找 Service-Linked Role (服務連結角色) 欄顯示為 Yes (是) 的服務。選擇具有連結的是，以檢視該服務的服務連結角色文件。

Elastic Beanstalk 的服務連結角色許可

Elastic Beanstalk 使用名為的服務連結角色 AWSServiceRoleForElasticBeanstalkManagedUpdates—允許 Elastic Beanstalk 執行正在執行的環境的已排程平台更新。

服 AWSServiceRoleForElasticBeanstalkManagedUpdates 務連結角色會信任下列服務擔任該角色：

- `managedupdates.elasticbeanstalk.amazonaws.com`

受管理策略AWSElasticBeanstalkManagedUpdatesServiceRolePolicy允許 AWSServiceRoleForElasticBeanstalkManagedUpdates 服務連結角色 Elastic Beanstalk 代表您完成受管理更新動作所需的所有權限。若要檢視受管理的策略內容，請參閱《AWS 受管理策略參考指南》中的AWSElasticBeanstalkManagedUpdatesServiceRolePolicy頁面。

您必須設定許可，IAM 實體 (如使用者、群組或角色) 才可建立、編輯或刪除服務連結角色。如需詳細資訊，請參閱《IAM 使用者指南》中的服務連結角色許可。

或者，您可以使用 AWS 受管政策來提供對 Elastic Beanstalk 的完整存取權。

為 Elastic Beanstalk 建立服務連結角色

您不需要手動建立一個服務連結角色。當您使用 Elastic Beanstalk API 建立 Elastic Beanstalk 環境，啟用受管更新，並指定 `AWSServiceRoleForElasticBeanstalkManagedUpdates` 為 `aws:elasticbeanstalk:managedactions` 命名空間 `ServiceRoleForManagedUpdates` 選項的值時，Elastic Beanstalk 會為您建立服務連結角色。

當您建立環境時，Elastic Beanstalk 嘗試為您的帳戶建立 AWSServiceRoleForElasticBeanstalkManagedUpdates 服務連結角色時，您必須擁有該權限。`iam:CreateServiceLinkedRole`如果您沒有此許可，環境建立會失敗，且您會看到一則說明問題的訊息。

或者，具有建立服務連結角色許可的另一個使用者可用 IAM 來預先建立服務連結角色。然後，即使沒有 `iam:CreateServiceLinkedRole` 權限，您也可以建立您的環境。

您 (或其他使用者) 可以使用 IAM 主控台透過 Elastic Beanstalk Managed Updates (Elastic Beanstalk 受管更新) 使用案例建立服務連結角色。在 IAM CLI 或 IAM API 中，建立一個使用 `managedupdates.elasticbeanstalk.amazonaws.com` 服務名稱的服務連結角色。如需詳細資訊，請參閱 IAM 使用者指南中的建立服務連結角色。如果您刪除此服務連結角色，您可以使用此相同的程序以再次建立該角色。

若您刪除此服務連結角色，之後需要再次建立，您可以在帳戶中使用相同程序重新建立角色。當您使用 Elastic Beanstalk API 建立 Elastic Beanstalk 環境，啟用受管更新，並指定 `AWSServiceRoleForElasticBeanstalkManagedUpdates` 為 `aws:elasticbeanstalk:managedactions` 命名空間 `ServiceRoleForManagedUpdates` 選項的值時，Elastic Beanstalk 會再次為您建立服務連結角色。

編輯 Elastic Beanstalk 的服務連結角色

Elastic Beanstalk 不允許您編輯 AWSServiceRoleForElasticBeanstalkManagedUpdates 服務連結角色。因為有各種實體可能會參考服務連結角色，所以您無法在建立角色之後變更角色名稱。然而，您可使用 IAM 來編輯角色描述。如需更多資訊，請參閱 IAM 使用者指南中的編輯服務連結角色。

刪除 Elastic Beanstalk 的服務連結角色

若您不再使用需要服務連結角色的功能或服務，我們建議您刪除該角色。如此一來，您就沒有未主動監控或維護的未使用實體。然而，在手動刪除服務連結角色之前，您必須先清除資源。

清除服務連結角色

您必須先確保已啟用受管更新的 Elastic Beanstalk 環境皆使用不同的服務角色或已終止，才能使用 IAM 刪除服務連結角色。

> **Note**
>
> 當您嘗試終止環境時，如果 Elastic Beanstalk 服務仍在使用該服務連結角色，則終止可能會失敗。若此情況發生，請等待數分鐘後並再次嘗試操作。

終止使用 AWSServiceRoleForElasticBeanstalkManagedUpdates （控制台）的 Elastic Beanstalk 環境

1. 開啟彈性魔豆控制台，然後在區域清單中選取您的. AWS 區域

2. 在導覽窗格中，選擇環境，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 選擇 Actions (動作)，然後選擇 Terminate Environment (終止環境)。
4. 使用畫面顯示對話方塊來確認環境終止。

如需使用 EB CLI 終止 Elastic Beanstalk 環境的詳細資訊，請參閱eb terminate。

TerminateEnvironment如需有關使用 API 終止 Elastic Beanstalk 環境的詳細資訊，請參閱。

手動刪除服務連結角色

使用 IAM 主控台、IAM CLI 或 IAM API 刪除 AWSServiceRoleForElasticBeanstalkManagedUpdates 服務連結角色。如需詳細資訊，請參閱《IAM 使用者指南》中的刪除服務連結角色。

Elastic Beanstalk 服務連結角色的支援區域

Elastic Beanstalk 在所有提供服務的區域中都支援使用服務連結角色。如需詳細資訊，請參閱 AWS Elastic Beanstalk 端點和配額。

管理 Elastic Beanstalk 使用者政策

AWS Elastic Beanstalk 提供兩個受管政策，可讓您為 Elastic Beanstalk 管理的所有資源指派完整存取權或唯讀存取權。您可以將政策附加到 AWS Identity and Access Management (IAM) 使用者或群組，或附加至使用者所擔任的角色。

受管使用者政策

- AdministratorAccessAWSElasticBeanstalk- 提供使用者完整的管理權限，以建立、修改和刪除 Elastic Beanstalk 應用程式、應用程式版本、組態設定、環境及其基礎資源。若要檢視受管理的策略內容，請參閱AWS 受管理策略參考指南中的 AdministratorAccess-AWSElasticBeanstalk 頁面。
- AWSElasticBeanstalkReadOnly— 允許使用者檢視應用程式和環境，但不能執行修改應用程式和環境的作業。它提供對所有 Elastic Beanstalk 資源以及 Elastic Beanstalk 控制台擷取的其他 AWS 資源的唯讀存取權。請注意，唯讀存取不會啟用諸如下載 Elastic Beanstalk 日誌供您讀取等動作。這是因為日誌暫存在 Amazon S3 儲存貯體，其中 Elastic Beanstalk 需要寫入許可。請參閱本主題結尾的範例，了解如何啟用 Elastic Beanstalk 日誌的存取。若要檢視受管理的策略內容，請參閱《AWS 受管理策略參考指南》中的AWSElasticBeanstalkReadOnly頁面。

> **Important**
>
> Elastic Beanstalk 受管政策不提供精細許可，其會授予使用 Elastic Beanstalk 應用程式可能需要的所有許可。在某些情況下，您可能希望進一步限制我們託管策略的權限。如需一個使用案例的範例，請參閱防止跨環境 Amazon S3 儲存貯體存取。
> 此外，我們的受管政策也不涵蓋您可能新增至解決方案，並且不由 Elastic Beanstalk 管理的自訂資源許可。若要實作更精細的許可、最低必要許可或自訂資源許可，請使用自訂政策。

已廢除的 受管政策

之前，Elastic Beanstalk 支援另外兩個受管理的使用者政策，AWSElasticBeanstalkFullAccess以及. AWSElasticBeanstalkReadOnlyAccess 我們計劃淘汰這些之前的政策。您可能仍然可以在 IAM 主控台中查看和使用它們。不過，我們建議您轉換為使用新的受管使用者政策，並新增自訂政策來授予自訂資源許可 (若有)。

與其他服務整合的政策

如果您偏好使用其他服務，我們也提供更精細的政策，讓您能夠將環境與此類其他服務整合。

- AWSElasticBeanstalkRoleCWL— 允許環境管理 Amazon CloudWatch 日誌記錄群組。
- AWSElasticBeanstalkRoleRDS— 允許環境整合 Amazon RDS 執行個體。
- AWSElasticBeanstalkRoleWorkerTier— 允許工作者環境層建立 Amazon DynamoDB 表格和 Amazon SQS 佇列。
- AWSElasticBeanstalkRoleECS— 允許多容器泊塢視窗環境管理 Amazon ECS 叢集。
- AWSElasticBeanstalkRoleCore— 允許 Web 服務環境的核心操作。
- AWSElasticBeanstalkRoleSNS— 允許環境啟用 Amazon SNS 主題整合。

若要查看特定受管原則的 JSON 來源，請參閱受AWS 管政策參考指南。

透過受管政策控制存取

您可以使用受管政策，來授予對 Elastic Beanstalk 的完整存取或唯讀存取。當需要其他許可才能存取新功能時，Elastic Beanstalk 會自動更新這些政策。

若要將受管政策套用至 IAM 使用者或群組

1. 在 IAM 主控台中開啟 Policies (政策) 頁面。

2. 在搜尋方塊中，輸入 **AWSElasticBeanstalk** 以篩選政策。
3. 在策略清單中，選取AWSElasticBeanstalkReadOnly或 AdministratorAccess- 旁邊的核取方塊 AWSElasticBeanstalk。
4. 選擇政策動作，再選擇附加。
5. 選取一或多個使用者和群組以將政策連接到。您可用篩選功能表和搜尋方塊來篩選主體實體清單。
6. 選擇連接政策。

建立自訂使用者政策

您可以建立自己的 IAM 政策，以允許或拒絕對特定 Elastic Beanstalk 資源上執行特定 Elastic Beanstalk API 動作，並控制對不由 Elastic Beanstalk 管理的自訂資源的存取。如需將政策連接至使用者或群組的相關詳細資訊，請參閱《IAM 使用者指南》中的使用政策。如需有關建立自訂政策的詳細資訊，請參閱《IAM 使用者指南》中的建立 IAM 政策。

> **Note**
>
> 請注意，雖然您可限制使用者與 Elastic Beanstalk API 的互動方式，但目前仍沒有有效的方法，可針對具備建立必要基礎資源許可的使用者，避免其在 Amazon EC2 和其他服務內建立其他資源。
> 您可以將這些政策視為分配 Elastic Beanstalk 責任的有效方法，而不是保護所有基礎資源的方法。

2019 年 11 月，Elastic Beanstalk 發佈支援 Amazon EC2 啟動範本。這是您環境的 Auto Scaling 群組可用來啟動 Amazon EC2 執行個體的新資源類型，而且需要新的許可。大多數客戶應該不會受到影響，因為如果使用者政策缺少所需權限，環境仍可使用舊式資源、啟動組態。不過，如果您嘗試使用需要 Amazon EC2 啟動範本的新功能，而且有自訂政策，則環境建立或更新作業可能會失敗。在此情況下，請確保自訂政策具有下列權限。

Amazon EC2 啟動範本的必要許可

- `EC2:CreateLaunchTemplate`
- `EC2:CreateLaunchTemplateVersions`
- `EC2:DeleteLaunchTemplate`
- `EC2:DeleteLaunchTemplateVersions`
- `EC2:DescribeLaunchTemplate`

- `EC2:DescribeLaunchTemplateVersions`

IAM 政策內含描述您欲授予之許可的政策陳述式。為 Elastic Beanstalk 建立政策陳述式時，您必須了解如何使用政策陳述式的下列四個部分：

- Effect (效果) 指定欲允許或拒絕陳述式內的動作。
- Action (動作) 指定您欲控制的 API 操作。例如，使用 `elasticbeanstalk:CreateEnvironment` 來指定 `CreateEnvironment` 操作。特定操作 (如建立環境) 需要其他許可來執行這些動作。如需詳細資訊，請參閱 Elastic Beanstalk 動作的資源與條件。

  > **Note**
  >
  > 若要使用 `UpdateTagsForResource` API 操作，請指定下列兩個虛擬動作之一 (或同時)，而非 API 操作名稱：
  >
  > `elasticbeanstalk:AddTags`
  >
  > 控制呼叫 `UpdateTagsForResource` 並傳送欲在 `TagsToAdd` 參數新增之標籤清單的許可。
  >
  > `elasticbeanstalk:RemoveTags`
  >
  > 控制呼叫 `UpdateTagsForResource` 並傳送欲在 `TagsToRemove` 參數移除之標籤金鑰清單的許可。

- Resource (資源) 指定您欲控制存取的資源。若要指定 Elastic Beanstalk 資源，請列出各個資源的 Amazon Resource Name (ARN)。
- (選用) Condition (條件) 指定陳述式授予之許可的限制。如需詳細資訊，請參閱 Elastic Beanstalk 動作的資源與條件。

以下章節示範幾個案例，其中您可能會考慮自訂使用者政策。

啟用有限 Elastic Beanstalk 環境建立

下列範例中的政策可讓使用者呼叫 `CreateEnvironment` 動作來建立名稱開頭為 **`Test`** 且具備特定應用程式及應用程式版本的環境。

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

```
    {
      "Sid":"CreateEnvironmentPerm",
      "Action": [
        "elasticbeanstalk:CreateEnvironment"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My First Elastic
 Beanstalk Application/Test*"
      ],
      "Condition": {
        "StringEquals": {
          "elasticbeanstalk:InApplication": ["arn:aws:elasticbeanstalk:us-
east-2:123456789012:application/My First Elastic Beanstalk Application"],
          "elasticbeanstalk:FromApplicationVersion": ["arn:aws:elasticbeanstalk:us-
east-2:123456789012:applicationversion/My First Elastic Beanstalk Application/First
 Release"]
        }
      }
    },
    {
      "Sid":"AllNonResourceCalls",
      "Action":[
        "elasticbeanstalk:CheckDNSAvailability",
        "elasticbeanstalk:CreateStorageLocation"
      ],
      "Effect":"Allow",
      "Resource":[
        "*"
      ]
    }
  ]
}
```

上述政策說明如何授予對 Elastic Beanstalk 操作有限制的存取。若要實際啟動環境，使用者必須擁有權限，才能建立為環境提供動力的 AWS 資源。例如，下列政策針對用於 Web 伺服器環境的預設資源組合，授予存取：

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

```
      "Action": [
        "ec2:*",
        "ecs:*",
        "elasticloadbalancing:*",
        "autoscaling:*",
        "cloudwatch:*",
        "s3:*",
        "sns:*",
        "cloudformation:*",
        "sqs:*"
        ],
      "Resource": "*"
    }
  ]
}
```

啟用存放在 Amazon S3 的 Elastic Beanstalk 日誌的存取

下列範例中的政策可讓使用者提取 Elastic Beanstalk 日誌、將日誌暫存在 Amazon S3 中，以及擷取日誌。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "s3:DeleteObject",
        "s3:GetObjectAcl",
        "s3:PutObjectAcl"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::elasticbeanstalk-*"
    }
  ]
}
```

> **Note**
>
> 若要僅限制這些日誌路徑的權限，請使用下列資源格式。

```
"arn:aws:s3:::elasticbeanstalk-us-east-2-123456789012/resources/environments/
logs/*"
```

啟用特定 Elastic Beanstalk 應用程式的管理

以下範例的政策可讓使用者在一個特定 Elastic Beanstalk 應用程式內管理環境和其他資源。此政策拒絕對其他應用程式的資源執行 Elastic Beanstalk 動作，也拒絕建立和刪除 Elastic Beanstalk 應用程式。

> **Note**
>
> 此政策不會拒絕存取透過其他服務的任何資源。它示範由不同使用者分配 Elastic Beanstalk 應用程式管理責任的有效方法，而不是做為保護基礎資源的方法。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "elasticbeanstalk:CreateApplication",
        "elasticbeanstalk:DeleteApplication"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Deny",
      "Action": [
        "elasticbeanstalk:CreateApplicationVersion",
        "elasticbeanstalk:CreateConfigurationTemplate",
        "elasticbeanstalk:CreateEnvironment",
        "elasticbeanstalk:DeleteApplicationVersion",
        "elasticbeanstalk:DeleteConfigurationTemplate",
        "elasticbeanstalk:DeleteEnvironmentConfiguration",
        "elasticbeanstalk:DescribeApplicationVersions",
        "elasticbeanstalk:DescribeConfigurationOptions",
```

```
        "elasticbeanstalk:DescribeConfigurationSettings",
        "elasticbeanstalk:DescribeEnvironmentResources",
        "elasticbeanstalk:DescribeEnvironments",
        "elasticbeanstalk:DescribeEvents",
        "elasticbeanstalk:DeleteEnvironmentConfiguration",
        "elasticbeanstalk:RebuildEnvironment",
        "elasticbeanstalk:RequestEnvironmentInfo",
        "elasticbeanstalk:RestartAppServer",
        "elasticbeanstalk:RetrieveEnvironmentInfo",
        "elasticbeanstalk:SwapEnvironmentCNAMEs",
        "elasticbeanstalk:TerminateEnvironment",
        "elasticbeanstalk:UpdateApplicationVersion",
        "elasticbeanstalk:UpdateConfigurationTemplate",
        "elasticbeanstalk:UpdateEnvironment",
        "elasticbeanstalk:RetrieveEnvironmentInfo",
        "elasticbeanstalk:ValidateConfigurationSettings"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringNotEquals": {
          "elasticbeanstalk:InApplication": [
            "arn:aws:elasticbeanstalk:us-east-2:123456789012:application/myapplication"
          ]
        }
      }
    }
  ]
}
```

Elastic Beanstalk 的 Amazon Resource Name 格式

您可以使用資源的 Amazon Resource Name (ARN)，藉此指定 IAM 政策的資源。在 Elastic Beanstalk 中，ARN 採用下列格式。

```
arn:aws:elasticbeanstalk:region:account-id:resource-type/resource-path
```

其中：

- *region* 為資源所在的區域 (如)**us-west-2**。
- *account-id* 是 AWS 帳戶 ID，無連字號 (例如，**123456789012**)

- *resource-type* 識別 Elastic Beanstalk 資源的類型 (如)environment。請參閱下表，取得所有 Elastic Beanstalk 資源類型的清單。
- *resource-path* 為能夠辨識特定資源的部分。Elastic Beanstalk 資源具備識別該資源的唯一路徑。請參閱下表，取得每個資源類型的資源路徑格式。例如，環境始終會與應用程式建立關聯。應用程式 **myEnvironment** 內環境 **myApp** 的資源路徑如下：

```
myApp/myEnvironment
```

Elastic Beanstalk 擁有您可於政策內指定的數種資源類型。下表說明每個資源類型的 ARN 格式和範例。

資源類型	ARN 格式
application	arn:aws:elasticbeanstalk: *region*:*account-id* :application/ *application-name* 範例：**arn:aws:elasticbeanstalk:us-east-2:123456789012:application/My App**
applicationversion	arn:aws:elasticbeanstalk: *region*:*account-id* :applicationversion/ *application-name* /*version-label* 範例：**arn:aws:elasticbeanstalk:us-east-2:123456789012:applicationversion/My App/My Version**
configurationtemplate	arn:aws:elasticbeanstalk: *region*:*account-id* :configurationtemplate/ *application-name* /*template-name* 範例：**arn:aws:elasticbeanstalk:us-east-2:123456789012:configurationtemplate/My App/My Template**
environment	arn:aws:elasticbeanstalk: *region*:*account-id* :environment/ *application-name* /*environment-name* 範例：**arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App/MyEnvironment**

資源類型	ARN 格式
platform	arn:aws:elasticbeanstalk: *region*:*account-id* :platform / *platform-name* /*platform-version* 範例：**arn:aws:elasticbeanstalk:us-east-2:123456789012:platform/MyPlatform/1.0**
solutionstack	arn:aws:elasticbeanstalk: *region*::solutionstack/ *solutionstack-name* 範例：**arn:aws:elasticbeanstalk:us-east-2::solutionstack/32bit Amazon Linux running Tomcat 7**

特定應用程式內永遠包含環境、應用程式版本和組態範本。您將發現這些資源的資源路徑均包含應用程式名稱，因此這些資源可由唯一的資源名稱和其中的應用程式辨識。儘管組態範本和環境會使用解決方案堆疊，但解決方案堆疊並非專屬於應用程式或 AWS 帳戶，而且其 ARN 中並不包含應用程式或 AWS 帳戶。

Elastic Beanstalk 動作的資源與條件

本章節說明您可於政策陳述式內使用的資源與條件，以授予能夠對特定 Elastic Beanstalk 資源執行特定 Elastic Beanstalk 動作的許可。

條件可讓您指定完成動作所需資源的許可。例如，當您可以呼叫 `CreateEnvironment` 動作，必須也指定欲部署的應用程式版本及內含應用程式名稱的應用程式。當您設定 `CreateEnvironment` 動作的許可，應使用 `InApplication` 和 `FromApplicationVersion` 條件，指定欲執行動作的應用程式及應用程式版本。

此外，您可使用解決方案堆疊 (`FromSolutionStack`) 或組態範本 (`FromConfigurationTemplate`) 來指定環境資訊。下列政策陳述式允許 `CreateEnvironment` 動作，透過搭配 **myenv** 組態 (`Resource`) 的應用程式版本 **My App** (`InApplication`) 的方式，在應用程式 **My Version** (由 `FromApplicationVersion` 條件指定) 內建立名為 **32bit Amazon Linux running Tomcat 7** (由 `FromSolutionStack` 指定) 的環境：

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

```
      "Action": [
        "elasticbeanstalk:CreateEnvironment"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App/myenv"
      ],
      "Condition": {
        "StringEquals": {
          "elasticbeanstalk:InApplication": ["arn:aws:elasticbeanstalk:us-
east-2:123456789012:application/My App"],
          "elasticbeanstalk:FromApplicationVersion": ["arn:aws:elasticbeanstalk:us-
east-2:123456789012:applicationversion/My App/My Version"],
          "elasticbeanstalk:FromSolutionStack": ["arn:aws:elasticbeanstalk:us-
east-2::solutionstack/32bit Amazon Linux running Tomcat 7"]
        }
      }
    }
  ]
}
```

> **Note**
>
> 本主題內提及的大多數條件索引鍵都是專屬於 Elastic Beanstalk，其名稱包含 `elasticbeanstalk:` 字首。為了簡潔起見，以下幾節提到這類條件鍵時，會省略名稱中的字首。舉例來說，會使用 `InApplication` 而非其全名 `elasticbeanstalk:InApplication`。
>
> 相對在提及少數幾個跨 AWS 服務使用的條件索引鍵時，會加上 `aws:` 字首，以突顯該例外情況。
>
> 政策範例中一律會顯示完整的條件鍵名稱，包括字首。

章節

- Elastic Beanstalk 動作的政策資訊
- Elastic Beanstalk 動作的條件金鑰

Elastic Beanstalk 動作的政策資訊

下表列出所有 Elastic Beanstalk 動作、每個動作針對的資源，以及可透過條件提供的其他情境資訊。

Elastic Beanstalk 動作的政策資訊，包括資源、條件、範例和依存項目

資源	條件	範例陳述式
動作：AbortEnvironmentUpdate		
application environment	aws:ResourceTag/ *key-name* (選用) aws:TagKeys (選用)	下列政策允許使用者在名為 My App 的應用程式中的環境內中止環境更新操作。 `{ "Version": "2012-10-17", "Statement": [ { "Action": [ "elasticbeanstalk:AbortEnvironmentUpdate" ], "Effect": "Allow", "Resource": [ "arn:aws:elasticbeanstalk:us-east-2:123456789012:application/My App" ] } ] }`
動作：CheckDNSAvailability		
"*"	N/A	`{ "Version": "2012-10-17", "Statement": [ { "Action": [ "elasticbeanstalk:CheckDNSAvailability" ], "Effect": "Allow", "Resource": "*" } ] }`

資源	條件	範例陳述式
動作：ComposeEnvironments		
application	aws:ResourceTag/ *key-name* (選用) aws:TagKeys (選用)	下列政策允許使用者撰寫屬於名為 My App 應用程式的環境。 `{ "Version": "2012-10-17", "Statement": [ { "Action": [ "elasticbeanstalk:ComposeEnvironments" ], "Effect": "Allow", "Resource": [ "arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App" ] } ] }`
動作：CreateApplication		

資源	條件	範例陳述式
application	aws:RequestTag/ *key-name* (選用) aws:TagKeys (選用)	此範例允許 CreateApplication 動作建立名稱開頭為 **DivA** 的應用程式： `{ "Version": "2012-10-17", "Statement": [ { "Action": [ "elasticbeanstalk:CreateApplication" ], "Effect": "Allow", "Resource": [ "arn:aws:elasticbeanstalk:us-east-2:123456789012:application/DivA*" ] } ] }`
動作：CreateApplicationVersion		

資源	條件	範例陳述式

applicationversion	InApplication
	
	aws:RequestTag/ *key-name* (選用)
	
	aws:TagKeys (選用)
		此範例允許 CreateApplicationVersion 動作在應用程式 * 中建立任意名稱 (**My App**) 的應用程式版本：
		
		{
		"Version": "2012-10-17",
		"Statement": [
		{
		"Action": [
		"elasticbeanstalk:CreateApplicationVersion"
		],
		"Effect": "Allow",
		"Resource": [
		"arn:aws:elasticbeanstalk:us-east-2:123456789012:applicationversion/My App/*"
		],
		"Condition": {
		"StringEquals": {
		"elasticbeanstalk:InApplication": ["arn:aws:elasticbeanstalk:us-east-2:123456789012:application/My App"]
		}
		}
		}
		]
		}

動作：CreateConfigurationTemplate		

資源	條件	範例陳述式

configurationtemplate	InApplication
	
	FromApplication
	
	FromApplicationVersion
	
	FromConfigurationTemplate
	
	FromEnvironment
	
	FromSolutionStack
	
	aws:RequestTag/ *key-name* (選用)
	
	aws:TagKeys (選用)
		下列政策允許 CreateConfigurationTemplate 動作在應用程式 **My Template** 中建立名稱開頭為 My Template* (**My App**) 的組態範本：
		
		{
		"Version": "2012-10-17",
		"Statement": [
		{
		"Action": [
		"elasticbeanstalk:CreateConfigurationTemplate"
		],
		"Effect": "Allow",
		"Resource": [
		"arn:aws:elasticbeanstalk:us-east-2:123456789012:configurationtemplate/My App/My Template*"
		],
		"Condition": {
		"StringEquals": {
		"elasticbeanstalk:InApplication": ["arn:aws:elasticbeanstalk:us-east-2:123456789012:application/My App"],
		"elasticbeanstalk:FromSolutionStack": ["arn:aws:elasticbeanstalk:us-east-2::solutionstack/32bit Amazon Linux running Tomcat 7"]
		}
		}
		}
		]
		}

動作：CreateEnvironment		

資源	條件	範例陳述式
environment	InApplication FromApplicationVersion FromConfigurationTemplate FromSolutionStack aws:RequestTag/ *key-name* (選用) aws:TagKeys (選用)	下列政策允許 CreateEnvironment 動作在應用程式 **myenv** 中，使用解決方案堆疊 **My App** 來建立名為 **32bit Amazon Linux running Tomcat 7** 的環境： `{ "Version": "2012-10-17", "Statement": [{ "Action": ["elasticbeanstalk:CreateEnvironment"], "Effect": "Allow", "Resource": ["arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App/myenv"], "Condition": { "StringEquals": { "elasticbeanstalk:InApplication": ["arn:aws:elasticbeanstalk:us-east-2:123456789012:application/My App"], "elasticbeanstalk:FromApplicationVersion": ["arn:aws:elasticbeanstalk:us-east-2:123456789012:applicationversion/My App/My Version"], "elasticbeanstalk:FromSolutionStack": ["arn:aws:elasticbeanstalk:us-east-2::solutionstack/32bit Amazon Linux running Tomcat 7"] } } }] }`

動作：CreatePlatformVersion

資源	條件	範例陳述式

platform	aws:RequestTag/ *key-name* (選用)
	
	aws:TagKeys (選用)
		此範例允許 CreatePlatformVersion 動作建立以 us-east-2 區域為目標的平台版本，其中的名稱開頭為 **us-east-2_** ：
		
		{
		"Version": "2012-10-17",
		"Statement": [
		{
		"Action": [
		"elasticbeanstalk:CreatePlatformVersion"
		],
		"Effect": "Allow",
		"Resource": [
		"arn:aws:elasticbeanstalk:us-east-2:123456789012:platform/us-east-2_*"
		]
		}
		]
		}

動作：CreateStorageLocation		

"*"	N/A	{
		"Version": "2012-10-17",
		"Statement": [
		{
		"Action": [
		"elasticbeanstalk:CreateStorageLocation"
		],
		"Effect": "Allow",
		"Resource": "*"
		}
		]
		}

動作：DeleteApplication		

資源	條件	範例陳述式
application	aws:ResourceTag/ *key-name* (選用) aws:TagKeys (選用)	下列政策允許 DeleteApplication 動作刪除應用程式 **My App**： `{ "Version": "2012-10-17", "Statement": [ { "Action": [ "elasticbeanstalk:DeleteApplication" ], "Effect": "Allow", "Resource": [ "arn:aws:elasticbeanstalk:us-east-2:123456789012:application/My App" ] } ] }`
動作：DeleteApplicationVersion		

資源	條件	範例陳述式
applicationversion	InApplication aws:ResourceTag/ *key-name* (選用) aws:TagKeys (選用)	下列政策允許 DeleteApplicationVersion 動作在應用程式 **My Version** 中刪除名為 **My App** 的應用程式版本： `{ "Version": "2012-10-17", "Statement": [{ "Action": ["elasticbeanstalk:DeleteApplicationVersion"], "Effect": "Allow", "Resource": ["arn:aws:elasticbeanstalk:us-east-2:123456789012:applicationversion/My App/My Version"], "Condition": { "StringEquals": { "elasticbeanstalk:InApplication": ["arn:aws:elasticbeanstalk:us-east-2:123456789012:application/My App"] } } }] }`

動作：DeleteConfigurationTemplate

資源	條件	範例陳述式
configurationtemplate	InApplication (選用) aws:ResourceTag/ *key-name* (選用) aws:TagKeys (選用)	下列政策允許 DeleteConfigurationTemplate 動作在應用程式 **My Template** 中刪除名為 **My App** 的組態範本。將應用程式名稱指定為條件為選用。 `{ "Version": "2012-10-17", "Statement": [ { "Action": [ "elasticbeanstalk:DeleteConfigurationTemplate" ], "Effect": "Allow", "Resource": [ "arn:aws:elasticbeanstalk:us-east-2:123456789012:configurationtemplate/My App/My Template" ] } ] }`
動作：DeleteEnvironmentConfiguration		

資源	條件	範例陳述式
environment	InApplication (選用)	下列政策允許 DeleteEnvironmentConfiguration 動作在應用程式 **myenv** 中刪除環境 **My App** 的草稿組態。將應用程式名稱指定為條件為選用。 `{ "Version": "2012-10-17", "Statement": [ { "Action": [ "elasticbeanstalk:DeleteEnvironmentConfiguration" ], "Effect": "Allow", "Resource": [ "arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App/myenv" ] } ] }`
動作：DeletePlatformVersion		

資源	條件	範例陳述式

platform	aws:ResourceTag/ *key-name* (選用)
	
	aws:TagKeys (選用)
		以下政策允許 DeletePlatformVersion 動作刪除以 us-east-2 區域為目標的平台版本，其中的名稱開頭為 **us-east-2_** ：
		
		{
		"Version": "2012-10-17",
		"Statement": [
		{
		"Action": [
		"elasticbeanstalk:DeletePlatformVersion"
		],
		"Effect": "Allow",
		"Resource": [
		"arn:aws:elasticbeanstalk:us-east-2:123456789012:platform/us-east-2_*"
		]
		}
		]
		}

動作：DescribeApplications		

資源	條件	範例陳述式

application	aws:ResourceTag/ *key-name* (選用)
	
	aws:TagKeys (選用)
		下列政策允許 DescribeApplications 動作描述應用程式 My App。
		
		{
		"Version": "2012-10-17",
		"Statement": [
		{
		"Action": [
		"elasticbeanstalk:DescribeApplications"
		],
		"Effect": "Allow",
		"Resource": [
		"arn:aws:elasticbeanstalk:us-east-2:123456789012:application/My App"
		]
		}
		]
		}

動作：DescribeApplicationVersions		

資源	條件	範例陳述式
applicationversion	InApplication (選用) aws:ResourceTag/ *key-name* (選用) aws:TagKeys (選用)	下列政策允許 DescribeApplicationVersions 動作在應用程式 **My Version** 中描述應用程式版本 **My App**。將應用程式名稱指定為條件為選用。 `{ "Version": "2012-10-17", "Statement": [ { "Action": [ "elasticbeanstalk:DescribeApplicationVersions" ], "Effect": "Allow", "Resource": [ "arn:aws:elasticbeanstalk:us-east-2:123456789012:applicationversion/My App/My Version" ] } ] }`
動作：DescribeConfigurationOptions		

資源	條件	範例陳述式
environment configurationtemplate solutionstack	InApplication (選用) aws:ResourceTag/ *key-name* (選用) aws:TagKeys (選用)	下列政策允許 DescribeConfigurationOptions 動作在應用程式 **myenv** 中描述環境 **My App** 的組態選項。將應用程式名稱指定為條件為選用。 `{ "Version": "2012-10-17", "Statement": [ { "Action": "elasticbeanstalk:DescribeConfigurationOptions", "Effect": "Allow", "Resource": [ "arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App/myenv" ] } ] }`
動作：DescribeConfigurationSettings		

資源	條件	範例陳述式

environment
 	
configurationtemplate
	InApplication (選用)
	
	aws:ResourceTag/ *key-name* (選用)
	
	aws:TagKeys (選用)
		下列政策允許 DescribeConfigurationSettings 動作在應用程式 **myenv** 中描述環境 **My App** 的組態設定。將應用程式名稱指定為條件為選用。
		
		{
		"Version": "2012-10-17",
		"Statement": [
		{
		"Action": "elasticbeanstalk:DescribeConfigurationSettings",
		"Effect": "Allow",
		"Resource": [
		"arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App/myenv"
		]
		}
		]
		}

動作：DescribeEnvironmentHealth		

資源	條件	範例陳述式
environment	aws:ResourceTag/ *key-name* (選用) aws:TagKeys (選用)	下列政策允許使用 DescribeEnvironmentHealth 來擷取名為 **myenv** 之環境的運作狀態資訊。 `{ "Version": "2012-10-17", "Statement": [ { "Action": "elasticbeanstalk:DescribeEnvironmentHealth", "Effect": "Allow", "Resource": [ "arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App/myenv" ] } ] }`
動作：DescribeEnvironmentResources		

資源	條件	範例陳述式
environment	InApplication (選用) aws:ResourceTag/ *key-name* (選用) aws:TagKeys (選用)	下列政策允許 DescribeEnvironmentResources 動作在應用程式 **My App** 中回傳環境 **myenv** 的 AWS 資源清單。將應用程式名稱指定為條件為選用。 `{ "Version": "2012-10-17", "Statement": [ { "Action": "elasticbeanstalk:DescribeEnvironmentResources", "Effect": "Allow", "Resource": [ "arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App/myenv" ] } ] }`
動作：DescribeEnvironments		

資源	條件	範例陳述式
environment	InApplication (選用) aws:ResourceTag/ *key-name* (選用) aws:TagKeys (選用)	下列政策允許 DescribeEnvironments 動作在應用程式 **myenv** 中描述環境 **myotherenv** 和 **My App**。將應用程式名稱指定為條件為選用。 `{ "Version": "2012-10-17", "Statement": [ { "Action": "elasticbeanstalk:DescribeEnvironments", "Effect": "Allow", "Resource": [ "arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App/myenv", "arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App2/myotherenv" ] } ] }`
動作：DescribeEvents		

資源	條件	範例陳述式
application applicationversion configurationtemplate environment	InApplication aws:ResourceTag/ *key-name* (選用) aws:TagKeys (選用)	下列政策允許 DescribeEvents 動作列出應用程式 **myenv** 中環境 **My Version** 及應用程式版本 **My App** 的事件描述。 `{ "Version": "2012-10-17", "Statement": [{ "Action": "elasticbeanstalk:DescribeEvents", "Effect": "Allow", "Resource": ["arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App/myenv", "arn:aws:elasticbeanstalk:us-east-2:123456789012:applicationversion/My App/My Version"], "Condition": { "StringEquals": { "elasticbeanstalk:InApplication": ["arn:aws:elasticbeanstalk:us-east-2:123456789012:application/My App"] } } }] }`
動作：DescribeInstancesHealth		

資源	條件	範例陳述式
environment	N/A	下列政策允許使用 DescribeInstancesHealth 來擷取名為 **myenv** 之環境內的執行個體運作狀態資訊。 `{ "Version": "2012-10-17", "Statement": [ { "Action": "elasticbeanstalk:DescribeInstancesHealth", "Effect": "Allow", "Resource": [ "arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App/myenv" ] } ] }`
動作：DescribePlatformVersion		

資源	條件	範例陳述式
platform	aws:ResourceTag/ *key-name* (選用) aws:TagKeys (選用)	以下政策允許 DescribePlatformVersion 動作描述以 us-east-2 區域為目標的平台版本，其中的名稱開頭為 **us-east-2_** ： `{ "Version": "2012-10-17", "Statement": [ { "Action": [ "elasticbeanstalk:DescribePlatformVersion" ], "Effect": "Allow", "Resource": [ "arn:aws:elasticbeanstalk:us-east-2:123456789012:platform/us-east-2_*" ] } ] }`
動作：ListAvailableSolutionStacks		

資源	條件	範例陳述式
solutionstack	N/A	以下政策僅允許 ListAvailableSolutionStacks 動作傳回解決方案堆疊 **32bit Amazon Linux running Tomcat 7**。 `{ "Version": "2012-10-17", "Statement": [ { "Action": [ "elasticbeanstalk:ListAvailableSolutionStacks" ], "Effect": "Allow", "Resource": "arn:aws:elasticbeanstalk:us-east-2::solutionstack/32bit Amazon Linux running Tomcat 7" } ] }`

動作：ListPlatformVersions

資源	條件	範例陳述式
platform	aws:RequestTag/ *key-name* (選用) aws:TagKeys (選用)	此範例允許 CreatePlatformVersion 動作建立以 us-east-2 區域為目標的平台版本，其中的名稱開頭為 **us-east-2_** ： `{ "Version": "2012-10-17", "Statement": [ { "Action": [ "elasticbeanstalk:ListPlatformVersions" ], "Effect": "Allow", "Resource": [ "arn:aws:elasticbeanstalk:us-east-2:123456789012:platform/us-east-2_*" ] } ] }`
動作：ListTagsForResource		

資源	條件	範例陳述式

application
 	
applicationversion
 	
configurationtemplate
 	
environment
 	
platform
	aws:ResourceTag/ *key-name* (選用)
	
	aws:TagKeys (選用)
		以下政策僅允許 ListTagsForResource 動作列出現有資源的標籤，且僅限具有名為 stage 且含有 test 此值之標籤的資源：
		
		{
		"Version": "2012-10-17",
		"Statement": [
		{
		"Action": [
		"elasticbeanstalk:ListTagsForResource"
		],
		"Effect": "Allow",
		"Resource": "*",
		"Condition": {
		"StringEquals": {
		"aws:ResourceTag/stage": ["test"]
		}
		}
		}
		]
		}

動作：RebuildEnvironment		

資源	條件	範例陳述式
environment	InApplication aws:ResourceTag/ *key-name* (選用) aws:TagKeys (選用)	下列政策允許 RebuildEnvironment 動作在應用程式 **myenv** 中重建環境 **My App**。 `{ "Version": "2012-10-17", "Statement": [{ "Action": ["elasticbeanstalk:RebuildEnvironment"], "Effect": "Allow", "Resource": ["arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App/myenv"], "Condition": { "StringEquals": { "elasticbeanstalk:InApplication": ["arn:aws:elasticbeanstalk:us-east-2:123456789012:application/My App"] } } }] }`
動作：RequestEnvironmentInfo		

資源	條件	範例陳述式
environment	InApplication aws:ResourceTag/ *key-name* (選用) aws:TagKeys (選用)	下列政策允許 RequestEnvironmentInfo 動作在應用程式 **myenv** 中編譯環境 **My App** 的資訊。 `{ "Version": "2012-10-17", "Statement": [{ "Action": ["elasticbeanstalk:RequestEnvironmentInfo"], "Effect": "Allow", "Resource": ["arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App/myenv"], "Condition": { "StringEquals": { "elasticbeanstalk:InApplication": ["arn:aws:elasticbeanstalk:us-east-2:123456789012:application/My App"] } } }] }`
動作：RestartAppServer		

資源	條件	範例陳述式
environment	InApplication	下列政策允許 RestartAppServer 動作在應用程式 **myenv** 中重新啟動環境 **My App** 的應用程式容器伺服器。 `{ "Version": "2012-10-17", "Statement": [{ "Action": ["elasticbeanstalk:RestartAppServer"], "Effect": "Allow", "Resource": ["arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App/myenv"], "Condition": { "StringEquals": { "elasticbeanstalk:InApplication": ["arn:aws:elasticbeanstalk:us-east-2:123456789012:application/My App"] } } }] }`
動作：RetrieveEnvironmentInfo		

資源	條件	範例陳述式
environment	InApplication aws:ResourceTag/ *key-name* (選用) aws:TagKeys (選用)	下列政策允許 RetrieveEnvironmentInfo 動作在應用程式 **myenv** 中擷取環境 **My App** 的編譯資訊。 `{ "Version": "2012-10-17", "Statement": [{ "Action": ["elasticbeanstalk:RetrieveEnvironmentInfo"], "Effect": "Allow", "Resource": ["arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App/myenv"], "Condition": { "StringEquals": { "elasticbeanstalk:InApplication": ["arn:aws:elasticbeanstalk:us-east-2:123456789012:application/My App"] } } }] }`
動作：SwapEnvironmentCNAMEs		

資源	條件	範例陳述式

environment	InApplication (選用)
	
	FromEnvironment (選用)
		下列政策允許 SwapEnvironmentCNAMEs 動作交換環境 **mysrcenv** 及 **mydestenv** 的 CNAME。
		
		{
		"Version": "2012-10-17",
		"Statement": [
		{
		"Action": [
		"elasticbeanstalk:SwapEnvironmentCNAMEs"
		],
		"Effect": "Allow",
		"Resource": [
		"arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App/mysrcenv",
		"arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App/mydestenv"
		]
		}
		]
		}

動作：TerminateEnvironment		

資源	條件	範例陳述式
environment	InApplication aws:ResourceTag/ *key-name* (選用) aws:TagKeys (選用)	下列政策允許 TerminateEnvironment 動作在應用程式 **myenv** 中終止環境 **My App**。 `{ "Version": "2012-10-17", "Statement": [{ "Action": ["elasticbeanstalk:TerminateEnvironment"], "Effect": "Allow", "Resource": ["arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App/myenv"], "Condition": { "StringEquals": { "elasticbeanstalk:InApplication": ["arn:aws:elasticbeanstalk:us-east-2:123456789012:application/My App"] } } }] }`
動作：UpdateApplication		

資源	條件	範例陳述式

application	aws:ResourceTag/ *key-name* (選用)
	
	aws:TagKeys (選用)
		下列政策允許 UpdateApplication 動作更新應用程式 **My App** 的屬性。
		
		{
		"Version": "2012-10-17",
		"Statement": [
		{
		"Action": [
		"elasticbeanstalk:UpdateApplication"
		],
		"Effect": "Allow",
		"Resource": [
		"arn:aws:elasticbeanstalk:us-east-2:123456789012:application/My App"
		]
		}
		]
		}

動作：UpdateApplicationResourceLifecycle		

資源	條件	範例陳述式
application	aws:ResourceTag/ *key-name* (選用) aws:TagKeys (選用)	以下政策會允許 UpdateApplicationResourceLifecycle 動作來更新應用程式 **My App** 的生命週期設定。 `{ "Version": "2012-10-17", "Statement": [ { "Action": [ "elasticbeanstalk:UpdateApplicationResourceLifecycle" ], "Effect": "Allow", "Resource": [ "arn:aws:elasticbeanstalk:us-east-2:123456789012:application/My App" ] } ] }`
動作：UpdateApplicationVersion		

資源	條件	範例陳述式

applicationversion	InApplication
	
	aws:ResourceTag/ *key-name* (選用)
	
	aws:TagKeys (選用)
		下列政策允許 UpdateApplicationVersion 動作在應用程式 **My Version** 中更新應用程式版本 **My App** 的屬性。
		
		{
		"Version": "2012-10-17",
		"Statement": [
		{
		"Action": [
		"elasticbeanstalk:UpdateApplicationVersion"
		],
		"Effect": "Allow",
		"Resource": [
		"arn:aws:elasticbeanstalk:us-east-2:123456789012:applicationversion/My App/My Version"
		],
		"Condition": {
		"StringEquals": {
		"elasticbeanstalk:InApplication": ["arn:aws:elasticbeanstalk:us-east-2:123456789012:application/My App"]
		}
		}
		}
		]
		}

動作：UpdateConfigurationTemplate

資源	條件	範例陳述式

configurationtemplate	InApplication
	
	aws:ResourceTag/ *key-name* (選用)
	
	aws:TagKeys (選用)
		下列政策允許 UpdateConfigurationTemplate 動作在應用程式 **My Template** 中更新組態範本 **My App** 的屬性或選項。
		
		{
		"Version": "2012-10-17",
		"Statement": [
		{
		"Action": [
		"elasticbeanstalk:UpdateConfigurationTemplate"
		],
		"Effect": "Allow",
		"Resource": [
		"arn:aws:elasticbeanstalk:us-east-2:123456789012:configurationtemplate/My App/My Template"
		],
		"Condition": {
		"StringEquals": {
		"elasticbeanstalk:InApplication": ["arn:aws:elasticbeanstalk:us-east-2:123456789012:application/My App"]
		}
		}
		}
		]
		}

動作：UpdateEnvironment

資源	條件	範例陳述式
environment	InApplication FromApplicationVersion FromConfigurationTemplate aws:ResourceTag/ *key-name* (選用) aws:TagKeys (選用)	下列政策允許 UpdateEnvironment 動作部署應用程式版本 **myenv**，藉此在應用程式 **My App** 中更新環境 **My Version**。 `{ "Version": "2012-10-17", "Statement": [{ "Action": ["elasticbeanstalk:UpdateEnvironment"], "Effect": "Allow", "Resource": ["arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App/myenv"], "Condition": { "StringEquals": { "elasticbeanstalk:InApplication": ["arn:aws:elasticbeanstalk:us-east-2:123456789012:application/My App"], "elasticbeanstalk:FromApplicationVersion": ["arn:aws:elasticbeanstalk:us-east-2:123456789012:applicationversion/My App/My Version"] } } }] }`
動作：UpdateTagsForResource - AddTags		

資源	條件	範例陳述式
application applicationversion configurationtemplate environment platform	aws:ResourceTag/ *key-name* (選用) aws:RequestTag/ *key-name* (選用) aws:TagKeys (選用)	AddTags 動作是與 UpdateTagsForResource API 相關聯的兩個虛擬動作之一。 以下政策僅允許 AddTags 動作修改現有資源的標籤，且僅限具有名為 stage 且含有 test 此值之標籤的資源： `{ "Version": "2012-10-17", "Statement": [ { "Action": [ "elasticbeanstalk:AddTags" ], "Effect": "Allow", "Resource": "*", "Condition": { "StringEquals": { "aws:ResourceTag/stage": ["test"] } } } ] }`
動作：UpdateTagsForResource - RemoveTags		

資源	條件	範例陳述式

application
 	
applicationversion
 	
configurationtemplate
 	
environment
 	
platform
	aws:ResourceTag/ *key-name* (選用)
	
	aws:TagKeys (選用)
		RemoveTags 動作是與 UpdateTagsForResource API 相關聯的兩個虛擬動作之一。
		
		以下政策拒絕 RemoveTags 動作請求移除現有環境中名為 stage 的標籤：
		
		{
		"Version": "2012-10-17",
		"Statement": [
		{
		"Action": [
		"elasticbeanstalk:RemoveTags"
		],
		"Effect": "Deny",
		"Resource": "*",
		"Condition": {
		"ForAnyValue:StringEquals": {
		"aws:TagKeys": ["stage"]
		}
		}
		}
		]
		}

動作：ValidateConfigurationSettings		

資源	條件	範例陳述式

template
 	
environment
	InApplication
	
	aws:ResourceTag/ *key-name* (選用)
	
	aws:TagKeys (選用)
		下列政策允許 ValidateConfigurationSettings 動作在應用程式 **myenv** 中根據環境 **My App** 驗證組態設定。
		
		{
		"Version": "2012-10-17",
		"Statement": [
		{
		"Action": [
		"elasticbeanstalk:ValidateConfigurationSettings"
		],
		"Effect": "Allow",
		"Resource": [
		"arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App/myenv"
		],
		"Condition": {
		"StringEquals": {
		"elasticbeanstalk:InApplication": ["arn:aws:elasticbeanstalk:us-east-2:123456789012:application/My App"]
		}
		}
		}
		]
		}

Elastic Beanstalk 動作的條件金鑰

金鑰可讓您指定條件以表達依存項目、限制許可或指定動作輸入參數的限制。Elastic Beanstalk 支援下列金鑰。

`InApplication`

指定內含動作執行針對之資源的應用程式。

下列範例允許 UpdateApplicationVersion 動作更新應用程式版本 **My Version** 的屬性。InApplication 條件將 **My App** 指定為 **My Version** 的容器。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "elasticbeanstalk:UpdateApplicationVersion"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:elasticbeanstalk:us-east-2:123456789012:applicationversion/My App/
My Version"
      ],
      "Condition": {
        "StringEquals": {
          "elasticbeanstalk:InApplication": ["arn:aws:elasticbeanstalk:us-
east-2:123456789012:application/My App"]
        }
      }
    }
  ]
}
```

FromApplicationVersion

將應用程式版本指定為依存項目或輸入參數的限制。

下列範例允許 UpdateEnvironment 動作在應用程式 **myenv** 中更新環境 **My App**。FromApplicationVersion 條件會限制 VersionLabel 參數，僅允許應用程式版本 **My Version** 更新環境。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "elasticbeanstalk:UpdateEnvironment"
      ],
      "Effect": "Allow",
      "Resource": [
```

```
        "arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App/myenv"
      ],
      "Condition": {
        "StringEquals": {
          "elasticbeanstalk:InApplication": ["arn:aws:elasticbeanstalk:us-
east-2:123456789012:application/My App"],
          "elasticbeanstalk:FromApplicationVersion": ["arn:aws:elasticbeanstalk:us-
east-2:123456789012:applicationversion/My App/My Version"]
        }
      }
    }
  ]
}
```

FromConfigurationTemplate

將組態範本指定為依存項目或輸入參數的限制。

下列範例允許 UpdateEnvironment 動作在應用程式 **myenv** 中更新環境 **My App**。FromConfigurationTemplate 條件會限制 TemplateName 參數，僅允許組態範本 **My Template** 更新環境。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "elasticbeanstalk:UpdateEnvironment"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App/myenv"
      ],
      "Condition": {
        "StringEquals": {
          "elasticbeanstalk:InApplication": ["arn:aws:elasticbeanstalk:us-
east-2:123456789012:application/My App"],
          "elasticbeanstalk:FromConfigurationTemplate":
 ["arn:aws:elasticbeanstalk:us-east-2:123456789012:configurationtemplate/My App/My
 Template"]
        }
      }
    }
```

```
  ]
}
```

FromEnvironment

將環境指定為依存項目或輸入參數的限制。

下列範例允許 SwapEnvironmentCNAMEs 動作交換所有環境內 **My App** 內名稱開頭為 **mysrcenv** 和 **mydestenv** 之環境的 CNAME，但不適用名稱開頭為 **mysrcenvPROD*** 和 **mydestenvPROD*** 的環境。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "elasticbeanstalk:SwapEnvironmentCNAMEs"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App/
mysrcenv*",
        "arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App/
mydestenv*"
      ],
      "Condition": {
        "StringNotLike": {
          "elasticbeanstalk:FromEnvironment": [
            "arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App/
mysrcenvPROD*",
            "arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App/
mydestenvPROD*"
          ]
        }
      }
    }
  ]
}
```

FromSolutionStack

將解決方案堆疊指定為依存項目或輸入參數的限制。

下列政策允許 CreateConfigurationTemplate 動作在應用程式 **My Template** 中建立名稱開頭為 My Template* (**My App**) 的組態範本。FromSolutionStack 條件會限制 solutionstack 參數，該參數僅能以解決方案堆疊 **32bit Amazon Linux running Tomcat 7** 做為輸入值。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "elasticbeanstalk:CreateConfigurationTemplate"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:elasticbeanstalk:us-east-2:123456789012:configurationtemplate/My
 App/My Template*"
      ],
      "Condition": {
        "StringEquals": {
          "elasticbeanstalk:InApplication": ["arn:aws:elasticbeanstalk:us-
east-2:123456789012:application/My App"],
          "elasticbeanstalk:FromSolutionStack": ["arn:aws:elasticbeanstalk:us-
east-2::solutionstack/32bit Amazon Linux running Tomcat 7"]
        }
      }
    }
  ]
}
```

aws:ResourceTag/*key-name*, aws:RequestTag/*key-name*, aws:TagKeys

指定標記型條件。如需詳細資訊，請參閱使用標籤來控制對 Elastic Beanstalk 資源的存取。

使用標籤來控制對 Elastic Beanstalk 資源的存取

AWS Identity and Access Management (IAM) 使用者政策陳述式中的條件，屬於您用來指定 Elastic Beanstalk 動作完成所需的資源存取許可的語法。如需指定政策陳述式條件的詳細資訊，請參閱Elastic Beanstalk 動作的資源與條件。在條件中使用標記是控制資源和請求的存取權限的方式之一。如需標記 Elastic Beanstalk 資源的相關資訊，請參閱標記 Elastic Beanstalk 應用程式資源。本主題討論的是標記型的存取控制。

設計 IAM 政策時，您可能會透過授予對特定資源的存取來設定精細許可。隨著您管理的資源數量增加，此任務變得越來越困難。標記資源並在政策陳述式條件中使用標籤，可讓此任務更輕鬆。您可以對具有特定標籤的任何資源大量授予存取。然後，您會在建立期間或之後，對相關資源重複套用此標籤。

可以將標記連接到資源或在請求中將標記傳遞至支援標記的服務。在 Elastic Beanstalk 中，資源可以擁有標籤，也有一些動作會包含標籤。在建立 IAM 政策時，可使用標記條件鍵來控制以下項目：

- 可在環境上執行動作的使用者 (以環境擁有的標記為準)。
- 可在動作請求中傳遞的標記。
- 請求中是否可使用特定的標籤鍵。

關於標籤條件索引鍵的完整語法和語義，請參閱 IAM 使用者指南中的使用標籤控制存取。

以下範例示範如何指定 Elastic Beanstalk 使用者政策中的標籤條件。

Example 1：根據請求中的標籤限制動作

Elastic Beanstalk AdministratorAccess-AWSElasticBeanstalk 受管使用者政策可提供使用者不受限制的許可，以在任何 Elastic Beanstalk 受管資源上執行任何 Elastic Beanstalk 動作。

以下政策會限制此能力，並拒絕未授權的使用者許可，禁止其建立 Elastic Beanstalk 生產環境。為了達到此種效果，如果該請求指定了名為 `CreateEnvironment` 的標記，含有 `stage` 或 `gamma` 的其中一值，其會拒絕 `prod` 動作。此外，該政策不允許標籤修改動作，因此無法加入相同的標籤值，也無法完全移除 `stage` 標籤，以此方式來防止未授權的使用者竄改生產環境的階段。除了受管使用者政策之外，客戶的管理員必須將此 IAM 政策連接到未授權的 IAM 使用者。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "elasticbeanstalk:CreateEnvironment",
        "elasticbeanstalk:AddTags"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/stage": ["gamma", "prod"]
        }
```

```
      }
    },
    {
      "Effect": "Deny",
      "Action": [
        "elasticbeanstalk:RemoveTags"
      ],
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:TagKeys": ["stage"]
        }
      }
    }
  ]
}
```

Example 2：根據資源標籤限制動作

Elastic Beanstalk AdministratorAccess-AWSElasticBeanstalk 受管使用者政策可提供使用者不受限制的許可，以在任何 Elastic Beanstalk 受管資源上執行任何 Elastic Beanstalk 動作。

以下政策會限制此能力，並拒絕未授權的使用者許可，禁止其在 Elastic Beanstalk 生產環境上執行動作。為了達到此種效果，如果該環境擁有名為 `stage` 的標記，含有 `gamma` 或 `prod` 的其中一值，其會拒絕特定動作。除了受管使用者政策之外，客戶的管理員必須將此 IAM 政策連接到未授權的 IAM 使用者。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "elasticbeanstalk:AddTags",
        "elasticbeanstalk:RemoveTags",
        "elasticbeanstalk:DescribeEnvironments",
        "elasticbeanstalk:TerminateEnvironment",
        "elasticbeanstalk:UpdateEnvironment",
        "elasticbeanstalk:ListTagsForResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
```

```
          "aws:ResourceTag/stage": ["gamma", "prod"]
        }
      }
    }
  ]
}
```

Example 3：根據請求中的標籤允許動作

下列政策會授予使用者許可來建立 Elastic Beanstalk 開發應用程式。

若要這樣做，它會在請求指定名為 `CreateApplication` 且值為 `AddTags` 的標籤時允許 `stage` 和 `development` 動作。`aws:TagKeys` 條件可確保使用者無法新增其他標籤鍵。尤其是，其可確保 `stage` 標籤鍵區分大小寫。請注意，此政策對於未連接 Elastic Beanstalk AdministratorAccess-AWSElasticBeanstalk 受管使用者政策的 IAM 使用者來說很實用。此受管政策可提供使用者不受限制的許可，以在任何 Elastic Beanstalk 受管資源上執行任何 Elastic Beanstalk 動作。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticbeanstalk:CreateApplication",
        "elasticbeanstalk:AddTags"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/stage": "development"
        },
        "ForAllValues:StringEquals": {
          "aws:TagKeys": ["stage"]
        }
      }
    }
  ]
}
```

Example 4：根據資源標籤允許動作

下列政策會授予使用者對 Elastic Beanstalk 開發應用程式執行動作和取得相關資訊的許可。

若要這樣做，它會在應用程式有名為 stage 且值為 development 的標籤時允許特定動作。aws:TagKeys 條件可確保使用者無法新增其他標籤鍵。尤其是，其可確保 stage 標籤鍵區分大小寫。請注意，此政策對於未連接 Elastic Beanstalk AdministratorAccess-AWSElasticBeanstalk 受管使用者政策的 IAM 使用者來說很實用。此受管政策可提供使用者不受限制的許可，以在任何 Elastic Beanstalk 受管資源上執行任何 Elastic Beanstalk 動作。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticbeanstalk:UpdateApplication",
        "elasticbeanstalk:DeleteApplication",
        "elasticbeanstalk:DescribeApplications"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/stage": "development"
        },
        "ForAllValues:StringEquals": {
          "aws:TagKeys": ["stage"]
        }
      }
    }
  ]
}
```

以受管政策為基礎的範例政策

此章節說明如何控制使用者存取 AWS Elastic Beanstalk，其中的範例政策提供常用案例的必要存取。這些政策衍生自 Elastic Beanstalk 受管政策。如需有關將受管政策連結至使用者和群組的資訊，請參閱 管理 Elastic Beanstalk 使用者政策。

在此案例中，軟體公司 Example Corp. 擁有三個負責公司網站的團隊：管理基礎設施的管理員、建置網站軟體的開發人員，以及測試網站的 QA 團隊。為了協助管理 Elastic Beanstalk 資源的許可，Example Corp. 建立三個群組，其中各團隊的成員分屬以下各群組：管理員、開發人員和測試人員。Example Corp. 希望管理員群組擁有所有應用程式、環境及其基礎資源的完整存取能力，讓他們能夠建立、故障診斷或刪除所有 Elastic Beanstalk 的資產。開發人員需要的許可，則包含檢視 Elastic Beanstalk 資產的許可，以及建立並部署應用程式版本的許可。開發人員不應具有建立新應用程式或

環境、或終止執行環境的許可。測試人員必須檢視所有 Elastic Beanstalk 資源，以監控並測試應用程式。測試人員不應具有變更任何 Elastic Beanstalk 資源的許可。

下列範例政策提供各個群組所需的許可。

範例 1：管理員群組 - 所有 Elastic Beanstalk 及相關服務 API

下列政策提供使用者使用 Elastic Beanstalk 所需的所有動作之許可。這項政策也允許 Elastic Beanstalk 代表您在以下服務中佈建和管理資源。Elastic Beanstalk 就是靠著這些額外服務在建立環境時佈建基礎資源。

- Amazon Elastic Compute Cloud
- Elastic Load Balancing
- Auto Scaling
- Amazon CloudWatch
- Amazon Simple Storage Service
- Amazon Simple Notification Service
- Amazon Relational Database Service
- AWS CloudFormation

請注意，本政策為範例。它對於 Elastic Beanstalk 用於管理應用程式和環境的 AWS 服務，提供廣泛的許可。例如，`ec2:*` 可讓 AWS Identity and Access Management (IAM) 使用者對 AWS 帳戶中的任何 Amazon EC2 資源執行任何動作。這些許可不限於您搭配 Elastic Beanstalk 使用的資源。以最佳實務而言，您應僅授予個人執行其職責所需的許可。

```
{
  "Version" : "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "elasticbeanstalk:*",
        "ec2:*",
        "elasticloadbalancing:*",
        "autoscaling:*",
        "cloudwatch:*",
        "s3:*",
```

```
        "sns:*",
        "rds:*",
        "cloudformation:*"
      ],
      "Resource" : "*"
    }
  ]
}
```

範例 2：開發人員群組 - 所有操作，但具有高度特殊權限者除外

下列政策拒絕建立應用程式和環境的許可，但允許所有其他 Elastic Beanstalk 動作。

請注意，本政策為範例。其針對 Elastic Beanstalk 管理應用程式和環境所用的 AWS 產品，提供廣泛的許可。例如，`ec2:*` 可讓 IAM 使用者對 AWS 帳戶中的任何 Amazon EC2 資源執行任何動作。這些許可不限於您搭配 Elastic Beanstalk 使用的資源。以最佳實務而言，您應僅授予個人執行其職責所需的許可。

```
{
  "Version" : "2012-10-17",
  "Statement" : [
    {
      "Action" : [
        "elasticbeanstalk:CreateApplication",
        "elasticbeanstalk:CreateEnvironment",
        "elasticbeanstalk:DeleteApplication",
        "elasticbeanstalk:RebuildEnvironment",
        "elasticbeanstalk:SwapEnvironmentCNAMEs",
        "elasticbeanstalk:TerminateEnvironment"],
      "Effect" : "Deny",
      "Resource" : "*"
    },
    {
      "Action" : [
        "elasticbeanstalk:*",
        "ec2:*",
        "elasticloadbalancing:*",
        "autoscaling:*",
        "cloudwatch:*",
        "s3:*",
        "sns:*",
        "rds:*",
        "cloudformation:*"],
```

```
      "Effect" : "Allow",
      "Resource" : "*"
    }
  ]
}
```

範例 3：測試人員 - 僅限檢視

下列政策允許所有應用程式、應用程式版本、事件和環境的唯讀存取。其不允許執行任何動作。

```
{
  "Version" : "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "elasticbeanstalk:Check*",
        "elasticbeanstalk:Describe*",
        "elasticbeanstalk:List*",
        "elasticbeanstalk:RequestEnvironmentInfo",
        "elasticbeanstalk:RetrieveEnvironmentInfo",
        "ec2:Describe*",
        "elasticloadbalancing:Describe*",
        "autoscaling:Describe*",
        "cloudwatch:Describe*",
        "cloudwatch:List*",
        "cloudwatch:Get*",
        "s3:Get*",
        "s3:List*",
        "sns:Get*",
        "sns:List*",
        "rds:Describe*",
        "cloudformation:Describe*",
        "cloudformation:Get*",
        "cloudformation:List*",
        "cloudformation:Validate*",
        "cloudformation:Estimate*"
      ],
      "Resource" : "*"
    }
  ]
}
```

根據資源許可的範例政策

本節將逐步介紹一個使用案例，此案例針對存取特定 Elastic Beanstalk 資源的 Elastic Beanstalk 動作，控制使用者的許可。我們將逐步介紹支援該使用案例的範例政策。如需 Elastic Beanstalk 資源政策的詳細資訊，請參閱建立自訂使用者政策。如需將政策連接到使用者和群組的相關資訊，請參閱《使用 AWS Identity and Access Management》中的管理 IAM 政策。

在我們的使用案例中，Example Corp. (範例公司) 是一家小型的顧問公司，針對兩種不同的客戶開發應用程式。John 是開發經理，負責監督兩種 Elastic Beanstalk 應用程式 (應用程式 1 和應用程式 2) 的開發。John 負責開發和針對兩種應用程式進行一些測試，而且只有他可以更新兩個應用程式的正式生產環境。下列是他所需的應用程式 1 和應用程式 2 的許可：

- 檢視應用程式、應用程式版本、環境和組態範本
- 建立應用程式版本，並將這些版本部署到預備環境
- 更新正式生產環境
- 建立和終止環境

Jill 是一位測試人員，需要存取許可來檢視下列資源，以監控和測試兩種應用程式：應用程式、應用程式版本、環境和組態範本。但是，她不應具有變更任何 Elastic Beanstalk 資源的許可。

Jack 是應用程式 1 的開發人員，需要存取許可來檢視應用程式 1 的所有資源，也需要建立應用程式 1 的應用程式版本，並將這些版本部署到預備環境。

Judy 是 Example Corp. 的 AWS 帳戶管理員，她為 John、Jill 和 Jack 建立了 IAM 使用者，並將下列政策連結到這些使用者，以針對應用程式 1 和應用程式 2 授予適當的許可。

範例 1：John - 應用程式 1、應用程式 2 的開發經理

我們已將 John 的政策細分為三項不同的政策，讓這些政策更易於閱讀和管理。這些政策共同授予了 John 所需的許可，使其能對這兩個應用程式執行開發、測試和部署動作。

第一個政策指定 Auto Scaling、Amazon S3、Amazon EC2、CloudWatch、Amazon SNS、Elastic Load Balancing、Amazon RDS 和 AWS CloudFormation 的動作。Elastic Beanstalk 就是靠著這些額外服務在建立環境時佈建基礎資源。

請注意，本政策為範例。其針對 Elastic Beanstalk 管理應用程式和環境所用的 AWS 產品，提供廣泛的許可。例如，`ec2:*` 可讓 IAM 使用者對 AWS 帳戶中的任何 Amazon EC2 資源執行任何動作。這些許可不限於您搭配 Elastic Beanstalk 使用的資源。以最佳實務而言，您應僅授予個人執行其職責所需的許可。

```
{
   "Version": "2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "ec2:*",
            "ecs:*",
            "ecr:*",
            "elasticloadbalancing:*",
            "autoscaling:*",
            "cloudwatch:*",
            "s3:*",
            "sns:*",
            "cloudformation:*",
            "dynamodb:*",
            "rds:*",
            "sqs:*",
            "logs:*",
            "iam:GetPolicyVersion",
            "iam:GetRole",
            "iam:PassRole",
            "iam:ListRolePolicies",
            "iam:ListAttachedRolePolicies",
            "iam:ListInstanceProfiles",
            "iam:ListRoles",
            "iam:ListServerCertificates",
            "acm:DescribeCertificate",
            "acm:ListCertificates",
            "codebuild:CreateProject",
            "codebuild:DeleteProject",
            "codebuild:BatchGetBuilds",
            "codebuild:StartBuild"
         ],
         "Resource":"*"
      }
   ]
}
```

第二個政策指定針對應用程式 1 和應用程式 2 的資源，John 能夠執行的 Elastic Beanstalk 動作。`AllCallsInApplications` 陳述式針對應用程式 1 和應用程式 2 內的所有資源 (例如 `"elasticbeanstalk:*"`)，允許所有能夠執行的 Elastic Beanstalk 動作

(elasticbeanstalk:CreateEnvironment)。AllCallsOnApplications 陳述式針對應用程式 1 和應用程式 2 的資源 (例如 "elasticbeanstalk:*"、elasticbeanstalk:DescribeApplications 等)，允許所有能夠執行的 Elastic Beanstalk 動作 (elasticbeanstalk:UpdateApplication)。AllCallsOnSolutionStacks 陳述式針對解決方案堆疊的資源 (例如 "elasticbeanstalk:*")，允許所有能夠執行的 Elastic Beanstalk 動作 (elasticbeanstalk:ListAvailableSolutionStacks)。

```
{
   "Version": "2012-10-17",
   "Statement":[
      {
         "Sid":"AllCallsInApplications",
         "Action":[
            "elasticbeanstalk:*"
         ],
         "Effect":"Allow",
         "Resource":[
            "*"
         ],
         "Condition":{
            "StringEquals":{
               "elasticbeanstalk:InApplication":[
                  "arn:aws:elasticbeanstalk:us-east-2:123456789012:application/app1",
                  "arn:aws:elasticbeanstalk:us-east-2:123456789012:application/app2"
               ]
            }
         }
      },
      {
         "Sid":"AllCallsOnApplications",
         "Action":[
            "elasticbeanstalk:*"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:elasticbeanstalk:us-east-2:123456789012:application/app1",
            "arn:aws:elasticbeanstalk:us-east-2:123456789012:application/app2"
         ]
      },
      {
         "Sid":"AllCallsOnSolutionStacks",
```

```
         "Action":[
            "elasticbeanstalk:*"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:elasticbeanstalk:us-east-2::solutionstack/*"
         ]
      }
   ]
}
```

第三個政策指定第二個政策需要許可才能完成的 Elastic Beanstalk 動作。AllNonResourceCalls 陳述式允許了 elasticbeanstalk:CheckDNSAvailability 動作的執行，這項動作是呼叫 elasticbeanstalk:CreateEnvironment 和其他動作所必需。此陳述式也允許了 elasticbeanstalk:CreateStorageLocation 動作的執行，這項動作是 elasticbeanstalk:CreateApplication、elasticbeanstalk:CreateEnvironment 和其他動作所必需。

```
{
   "Version": "2012-10-17",
   "Statement":[
      {
         "Sid":"AllNonResourceCalls",
         "Action":[
            "elasticbeanstalk:CheckDNSAvailability",
            "elasticbeanstalk:CreateStorageLocation"
         ],
         "Effect":"Allow",
         "Resource":[
            "*"
         ]
      }
   ]
}
```

範例 2：Jill - 應用程式 1、應用程式 2 的測試人員

我們已將 Jill 的政策細分為三項不同的政策，讓這些政策更易於閱讀和管理。這些政策共同授予了 Jill 所需的許可，使其能對兩種應用程式執行測試和監控動作。

第一個政策指定對 Auto Scaling、Amazon S3、Amazon EC2、CloudWatch、Amazon SNS、Elastic Load Balancing、Amazon RDS 和 AWS CloudFormation (適用於非舊式容器類型) 執行的

Describe*、List* 和 Get* 動作，讓 Elastic Beanstalk 動作能夠擷取應用程式 1 和應用程式 2 基礎資源的相關資訊。

```
{
   "Version": "2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "ec2:Describe*",
            "elasticloadbalancing:Describe*",
            "autoscaling:Describe*",
            "cloudwatch:Describe*",
            "cloudwatch:List*",
            "cloudwatch:Get*",
            "s3:Get*",
            "s3:List*",
            "sns:Get*",
            "sns:List*",
            "rds:Describe*",
            "cloudformation:Describe*",
         "cloudformation:Get*",
         "cloudformation:List*",
         "cloudformation:Validate*",
         "cloudformation:Estimate*"
         ],
         "Resource":"*"
      }
   ]
}
```

第二個政策指定針對應用程式 1 與應用程式 2 的資源，Jill 能夠執行的 Elastic Beanstalk 動作。AllReadCallsInApplications 陳述式允許 Jill 呼叫 Describe* 動作和環境資訊動作。AllReadCallsOnApplications 陳述式允許 Jill 針對應用程式 1 與應用程式 2 的應用程式資源，呼叫 DescribeApplications 與 DescribeEvents 動作。AllReadCallsOnSolutionStacks 允許針對解決方案堆疊資源 (ListAvailableSolutionStacks、DescribeConfigurationOptions 和 ValidateConfigurationSettings)，執行相關的檢視動作。

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

```
      {
         "Sid":"AllReadCallsInApplications",
         "Action":[
            "elasticbeanstalk:Describe*",
            "elasticbeanstalk:RequestEnvironmentInfo",
            "elasticbeanstalk:RetrieveEnvironmentInfo"
         ],
         "Effect":"Allow",
         "Resource":[
            "*"
         ],
         "Condition":{
            "StringEquals":{
               "elasticbeanstalk:InApplication":[
                  "arn:aws:elasticbeanstalk:us-east-2:123456789012:application/app1",
                  "arn:aws:elasticbeanstalk:us-east-2:123456789012:application/app2"
               ]
            }
         }
      },
      {
         "Sid":"AllReadCallsOnApplications",
         "Action":[
            "elasticbeanstalk:DescribeApplications",
            "elasticbeanstalk:DescribeEvents"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:elasticbeanstalk:us-east-2:123456789012:application/app1",
            "arn:aws:elasticbeanstalk:us-east-2:123456789012:application/app2"
         ]
      },
      {
         "Sid":"AllReadCallsOnSolutionStacks",
         "Action":[
            "elasticbeanstalk:ListAvailableSolutionStacks",
            "elasticbeanstalk:DescribeConfigurationOptions",
            "elasticbeanstalk:ValidateConfigurationSettings"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:elasticbeanstalk:us-east-2::solutionstack/*"
         ]
      }
```

```
   ]
}
```

第三個政策指定第二個政策需要許可才能完成的 Elastic Beanstalk 動作。`AllNonResourceCalls` 陳述式允許了 `elasticbeanstalk:CheckDNSAvailability` 動作的執行，這項動作是呼叫某些檢視動作所必需。

```
{
   "Version": "2012-10-17",
   "Statement":[
      {
         "Sid":"AllNonResourceCalls",
         "Action":[
            "elasticbeanstalk:CheckDNSAvailability"
         ],
         "Effect":"Allow",
         "Resource":[
            "*"
         ]
      }
   ]
}
```

範例 3：Jack - 應用程式 1 的開發人員

我們已將 Jack 的政策細分為三項不同的政策，讓這些政策更易於閱讀和管理。這些政策共同授予了 Jack 所需的許可，使其能對 應用程式 1 資源執行開發、測試和部署動作。

第一個政策指定對 Auto Scaling、Amazon S3、Amazon EC2、CloudWatch、Amazon SNS、Elastic Load Balancing、Amazon RDS 和 AWS CloudFormation (適用於非舊式容器類型) 執行的動作，讓 Elastic Beanstalk 動作能夠檢視和使用應用程式 1 的基礎資源。如需支援的非舊式容器類型的清單，請參閱 the section called “為何部分平台版本標記為舊版？”

請注意，本政策為範例。其針對 Elastic Beanstalk 管理應用程式和環境所用的 AWS 產品，提供廣泛的許可。例如，`ec2:*` 可讓 IAM 使用者對 AWS 帳戶中的任何 Amazon EC2 資源執行任何動作。這些許可不限於您搭配 Elastic Beanstalk 使用的資源。以最佳實務而言，您應僅授予個人執行其職責所需的許可。

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

```
      {
         "Effect":"Allow",
         "Action":[
            "ec2:*",
            "elasticloadbalancing:*",
            "autoscaling:*",
            "cloudwatch:*",
            "s3:*",
            "sns:*",
            "rds:*",
            "cloudformation:*"
         ],
         "Resource":"*"
      }
   ]
}
```

第二個政策指定針對應用程式 1 的資源，Jack 能夠執行的 Elastic Beanstalk 動作。

```
{
   "Version": "2012-10-17",
   "Statement":[
      {
         "Sid":"AllReadCallsAndAllVersionCallsInApplications",
         "Action":[
            "elasticbeanstalk:Describe*",
            "elasticbeanstalk:RequestEnvironmentInfo",
            "elasticbeanstalk:RetrieveEnvironmentInfo",
            "elasticbeanstalk:CreateApplicationVersion",
            "elasticbeanstalk:DeleteApplicationVersion",
            "elasticbeanstalk:UpdateApplicationVersion"
         ],
         "Effect":"Allow",
         "Resource":[
            "*"
         ],
         "Condition":{
            "StringEquals":{
               "elasticbeanstalk:InApplication":[
                  "arn:aws:elasticbeanstalk:us-east-2:123456789012:application/app1"
               ]
            }
         }
```

```
      },
      {
         "Sid":"AllReadCallsOnApplications",
         "Action":[
            "elasticbeanstalk:DescribeApplications",
            "elasticbeanstalk:DescribeEvents"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:elasticbeanstalk:us-east-2:123456789012:application/app1"
         ]
      },
      {
         "Sid":"UpdateEnvironmentInApplications",
         "Action":[
            "elasticbeanstalk:UpdateEnvironment"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/app1/app1-
staging*"
         ],
         "Condition":{
            "StringEquals":{
               "elasticbeanstalk:InApplication":[
                  "arn:aws:elasticbeanstalk:us-east-2:123456789012:application/app1"
               ]
            },
            "StringLike":{
               "elasticbeanstalk:FromApplicationVersion":[
                  "arn:aws:elasticbeanstalk:us-east-2:123456789012:applicationversion/
app1/*"
               ]
            }
         }
      },
      {
         "Sid":"AllReadCallsOnSolutionStacks",
         "Action":[
            "elasticbeanstalk:ListAvailableSolutionStacks",
            "elasticbeanstalk:DescribeConfigurationOptions",
            "elasticbeanstalk:ValidateConfigurationSettings"
         ],
         "Effect":"Allow",
```

```
         "Resource":[
            "arn:aws:elasticbeanstalk:us-east-2::solutionstack/*"
         ]
      }
   ]
}
```

第三個政策指定第二個政策需要許可才能完成的 Elastic Beanstalk 動作。`AllNonResourceCalls` 陳述式允許了 `elasticbeanstalk:CheckDNSAvailability` 動作的執行，這項動作是呼叫 `elasticbeanstalk:CreateEnvironment` 和其他動作所必需。此陳述式也允許了 `elasticbeanstalk:CreateStorageLocation` 動作的執行，這項動作是 `elasticbeanstalk:CreateEnvironment` 和其他動作所必需。

```
{
   "Version": "2012-10-17",
   "Statement":[
      {
         "Sid":"AllNonResourceCalls",
         "Action":[
            "elasticbeanstalk:CheckDNSAvailability",
            "elasticbeanstalk:CreateStorageLocation"
         ],
         "Effect":"Allow",
         "Resource":[
            "*"
         ]
      }
   ]
}
```

防止跨環境 Amazon S3 儲存貯體存取

Elastic Beanstalk 提供託管政策，以處理您帳戶中 Elastic Beanstalk 環境所需的 AWS 資源。 AWS 預設情況下提供給您 AWS 帳戶中某個應用程式的許可，可存取屬於同一 AWS 帳戶中其他應用程式的 S3 資源。

如果您的 AWS 帳戶執行多個 Beanstalk 應用程式，您可以建立自己的自訂政策以附加至每個環境的服務角色或執行個體設定檔，藉此降低原則的安全性範圍。然後，您可以將自訂政策中的 S3 許可限制在特定環境中。

> **Note**
>
> 請注意，您必須負責維護自訂原則。如果您的自訂原則所依據的 Elastic Beanstalk 受管政策發生變更，您將需要根據基本原則的個別變更來修改自訂原則。如需 Elastic Beanstalk 管理原則的變更記錄，請參閱。受管理政策的 Elastic Beanstalk 更新 AWS

關閉範圍權限的範例

下列範例是以AWSElasticBeanstalkWebTier受管理的策略為基礎。

預設政策包括 S3 儲存貯體許可的下列幾行。此預設政策不會將 S3 儲存貯體動作限制在特定環境或應用程式。

```
{
   "Sid" : "BucketAccess",
   "Action" : [
      "s3:Get*",
      "s3:List*",
      "s3:PutObject"
     ],
   "Effect" : "Allow",
   "Resource" : [
      "arn:aws:s3:::elasticbeanstalk-*",
      "arn:aws:s3:::elasticbeanstalk-*/*"
     ]
}
```

您可以將特定資源限定為指定為的服務角色來縮小存取範圍`Principal`。下列範例提供具有 id 之環境中 S3 儲存貯體的自訂服務角色`aws-elasticbeanstalk-ec2-role-my-example-env`許可`my-example-env-ID`。

Example 僅將權限授予特定環境的 S3 儲存貯體

```
{
   "Sid": "BucketAccess",
   "Action": [
      "s3:Get*",
      "s3:List*",
      "s3:PutObject"
```

```
    ],
   "Effect": "Allow",
   "Principal": {
      "AWS": "arn:aws:iam::...:role/aws-elasticbeanstalk-ec2-role-my-example-env"
     },
   "Resource": [
      "arn:aws:s3:::elasticbeanstalk-my-region-account-id-12345",
      "arn:aws:s3:::elasticbeanstalk-my-region-account-id-12345/resources/environments/
my-example-env-ID/*"
    ]
}
```

> **Note**
>
> 資源 ARN 必須包含 Elastic Beanstalk 環境 ID（不是環境名稱）。您可以從「環境概觀」頁面上的 Elastic Beanstalk 主控台取得環境識別碼。您也可以使用 AWS CLI 描述環境指令來取得此資訊。

如需協助您更新 Elastic Beanstalk 環境的 S3 儲存貯體許可的詳細資訊，請參閱下列資源：

- 本指南中的 將 Elastic Beanstalk 與 Amazon S3 搭配使用
- Amazon S3 在服務授權參考指南中定義的資源類型
- IAM 使用者指南中的 ARN 格式

搭配 Amazon RDS 使用 Elastic Beanstalk

您可以搭配 Amazon Relational Database Service (Amazon RDS) 使用 Elastic Beanstalk，以設定、操作和擴展關聯式資料庫。有以下兩個選項來開始使用。

- 在 Amazon RDS 中建立新的資料庫。
- 從先前由 Elastic Beanstalk 建立，並在隨後從 Beanstalk 環境解耦的資料庫開始。如需更多詳細資訊，請參閱 the section called “資料庫”。

您可以使用兩種方法之一在 Amazon RDS 中執行資料庫執行個體，並將應用程式設定為在啟動時連接至該執行個體。您可將多個環境連接至資料庫，並可透過藍/綠部署執行無縫更新。

> **Note**
>
> 若您未曾搭配應用程式使用資料庫執行個體，建議您先使用 Elastic Beanstalk 主控台將資料庫新增至測試環境。如此一來，您便可確認應用程式是否能讀取環境屬性、建構連線字串，以及連線至資料庫執行個體，且不需要獨立資料庫需要的其他組態工作。如需更多詳細資訊，請參閱 將資料庫新增至您的 Elastic Beanstalk 環境。

若要允許環境中的 Amazon EC2 執行個體連線至外部資料庫，則可以為與環境相關聯的 Auto Scaling 群組設定額外的安全群組。您可以連接已與資料庫執行個體連接的相同安全群組。或者，您可以使用獨立的安全群組。如果您連接不同的安全群組，則必須設定連接至資料庫的安全性群組，以允許來自此安全群組的傳入存取。

> **Note**
>
> 您可將規則新增至已連接資料庫的安全群組，藉此將您的環境連接至資料庫。此規則必須允許自動產生的安全群組的傳入存取，該安全群組將由 Elastic Beanstalk 連接至您環境的 Auto Scaling 群組。然而，請注意，藉由建立這項規則，您也在兩個安全群組間建立了相依性。之後當您嘗試終止環境時，Elastic Beanstalk 將無法刪除環境的安全群組，因為資料庫的安全群組與其相依。

在您啟動資料庫執行個體並設定安全群組後，您可使用環境屬性將連線資訊 (諸如端點和密碼) 傳遞至您的應用程式。當您在環境中執行資料庫執行個體時，這是 Elastic Beanstalk 在背景使用的相同機制。

如需多一層的安全性，您可將連線資訊存放於 Amazon S3，並將 Elastic Beanstalk 設定為在部署期間擷取該資訊。透過組態檔案 (`.ebextensions`)，您在部署應用程式時，可以在環境中設定執行個體，安全地從 Amazon S3 擷取檔案。

主題

- 於預設 VPC 內啟動並連線至外部 Amazon RDS 執行個體
- 於 EC2 Classic 內啟動並連接至外部 Amazon RDS 執行個體
- 將 Amazon RDS 憑證存放在 AWS Secrets Manager 中
- 清除外部 Amazon RDS 執行個體

於預設 VPC 內啟動並連線至外部 Amazon RDS 執行個體

若是要搭配在 Elastic Beanstalk 中執行的應用程式使用外部資料庫，您有兩個選項。或者，您可以使用 Amazon RDS 啟動資料庫執行個體。您使用 Amazon RDS 啟動的任何執行個體均完全獨立於 Elastic Beanstalk 和 Elastic Beanstalk 環境。這表示您可使用任何 Amazon RDS 支援的資料庫引擎和執行個體類型，即使是 Elastic Beanstalk 未使用的亦可。

或者，您可以從先前由 Elastic Beanstalk 建立，並在隨後從 Beanstalk 環境解耦的資料庫開始，將此作法作為啟動新資料庫執行個體的替代方式。如需更多詳細資訊，請參閱 the section called “資料庫”。使用此選項，您就不需要完成啟動新資料庫的程序。不過，您確實需要完成在本主題中所描述的後續程序。

下列為針對預設 VPC 的說明程序。若您使用自訂 VPC，程序亦相同。唯一的額外要求是，您的環境和資料庫執行個體必須位於相同的子網路，或位於能夠互相通訊的不同子網路。如需設定自訂 VPC 以搭配使用 Elastic Beanstalk 的詳細資訊，請參閱 搭配 Amazon VPC 使用 Elastic Beanstalk。

> **Note**
>
> - 如果您是從 Elastic Beanstalk 建立，並隨後從 Beanstalk 環境解耦的資料庫開始，則可以略過第一組步驟，並以修改 RDS 執行個體安全群組的傳入規則下分組的步驟繼續進行。
> - 如果您計劃為生產環境使用解耦的資料庫，請確認該資料庫使用的儲存類型是否適合您的工作負載。如需詳細資訊，請參閱《Amazon RDS 使用者指南》中的資料庫執行個體儲存體和修改資料庫執行個體。

欲在預設 VPC 中啟動 RDS 資料庫執行個體

1. 開啟 RDS 主控台。
2. 在導覽窗格中，選擇 Databases (資料庫)。
3. 選擇 Create database (建立資料庫)。
4. 選擇 Standard Create (標準建立)。

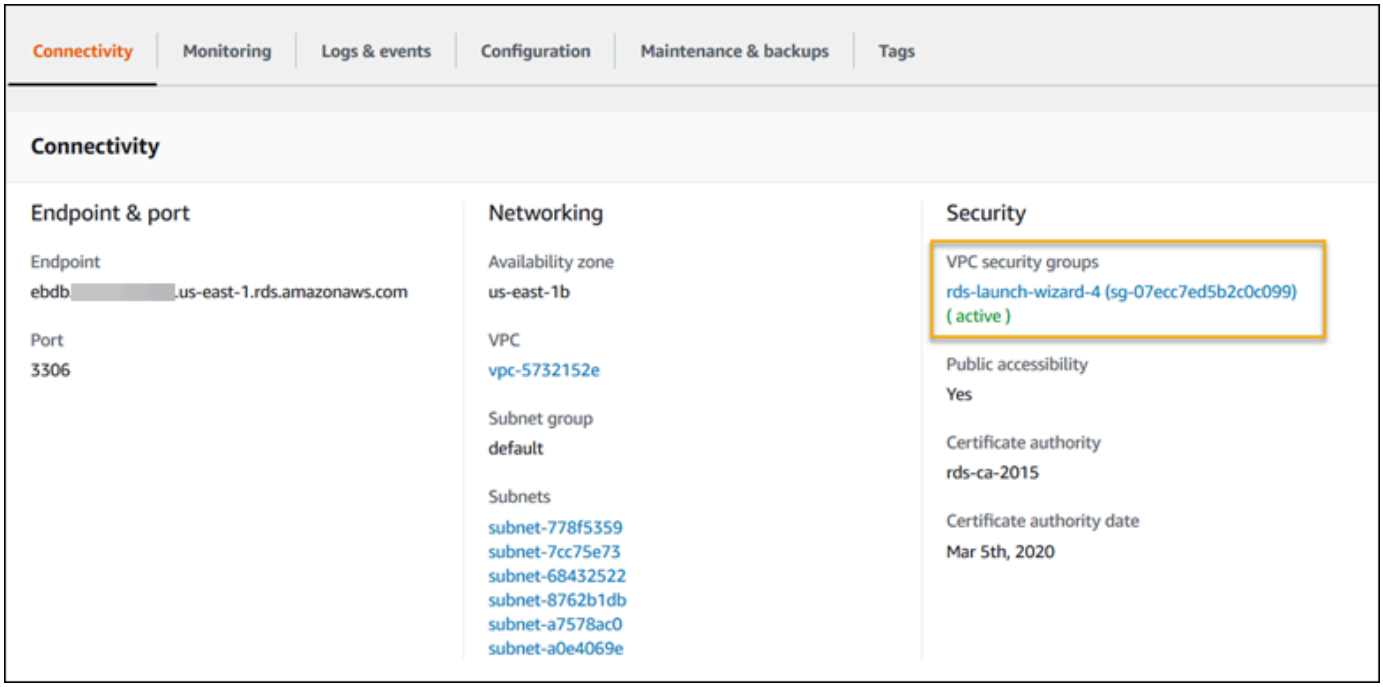
> **Important**
>
> 請勿選擇 Easy Create (輕鬆建立)。如果您選擇該選項，則您無法設定必要的設定來啟動此 RDS 資料庫。

5. 在 Additional configuration (其他設定) 下方的 Initial database name (初始資料庫名稱) 中輸入 **ebdb**。
6. 檢閱預設設定，並根據您的特定要求來調整這些設定。請注意以下選項：
   - 資料庫執行個體類別 – 選擇具有適當數量的記憶體和 CPU 功率之適合您工作負載的執行個體大小。
   - 異地同步備份部署 – 若要達到高可用性，請將此項設定為在不同的 AZ 中建立 Aurora 複本/讀取器節點。
   - 主要使用者名稱和主要密碼 – 資料庫使用者名稱和密碼。記下這些設定，以供稍後使用。
7. 檢查其餘選項的預設設定，然後選擇 Create database (建立資料庫)。

接著，修改連接至資料庫執行個體的安全群組，以允許適當連接埠的傳入流量。此安全群組與您稍後將連接至 Elastic Beanstalk 環境的安全群組相同。因此，您新增的規則會將傳入存取許可授予在相同安全群組內的其他資源。

修改連接至 RDS 執行個體的安全群組的傳入規則

1. 開啟 Amazon RDS 主控台。
2. 選擇 Databases (資料庫)。
3. 選擇您的資料庫執行個體名稱以檢視其詳細資料。
4. 在 Connectivity (連線) 區段中，記下顯示於此頁面的 Subnets (子網路)、Security groups (安全群組) 和 Endpoint (端點)。這樣您稍後便可使用這些資訊。
5. 在 Security (安全性) 下，可查看與資料庫執行個體相關聯的安全群組。開啟連結以檢視 Amazon EC2 主控台內的安全群組。



6. 在安全群組的詳細資訊中，選擇 Inbound (傳入)。
7. 選擇 **編輯** 。
8. 選擇 Add Rule (新增規則)。
9. 針對 Type (類型)，選擇您的應用程式所使用的資料庫引擎。
10. 對於 Source (來源)，輸入 **sg-** 檢視可用的安全群組清單。選擇與 Elastic Beanstalk 環境中使用之 Auto Scaling 群組相關聯的安全群組。以便環境中的 Amazon EC2 執行個體可以存取資料庫。

11. 選擇 Save (儲存)。

接著，請將資料庫執行個體的安全群組新增至執行環境。在此程序中，Elastic Beanstalk 會透過其他連接的安全群組，重新佈建您環境中的所有執行個體。

欲將安全群組新增至您的環境

- 執行下列任意一項：
  - 使用 Elastic Beanstalk 主控台新增安全群組

    a. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。

    b. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

    > **Note**
    >
    > 如果您有許多環境，請使用搜尋列來篩選環境清單。

    c. 在導覽窗格中，選擇 Configuration (組態)。

    d. 在 Instances (執行個體) 組態類別中，選擇 Edit (編輯)。

    e. 在 EC2 安全群組下，除了 Elastic Beanstalk 建立的執行個體安全群組，請選擇要連接到執行個體的安全群組。

    f. 若要儲存變更，請選擇頁面底部的儲存變更。

    g. 閱讀警告的內容，然後選擇 Confirm (確認)。
  - 若要使用組態檔案新增安全群組，請使用 `securitygroup-addexisting.config` 範例檔案。

接著，使用環境屬性，將連線資訊傳送至環境。在您透過 Elastic Beanstalk 主控台將資料庫執行個體新增到環境時，Elastic Beanstalk 會使用諸如 RDS_HOSTNAME 等環境屬性，將連線資訊傳遞至您的應用程式。您可以使用相同的屬性。藉此，您可以為整合資料庫執行個體和外部資料庫執行個體使用相同的應用程式程式碼。或者，您也可以選擇自己的屬性名稱。

設定 Amazon RDS 資料庫執行個體的環境屬性

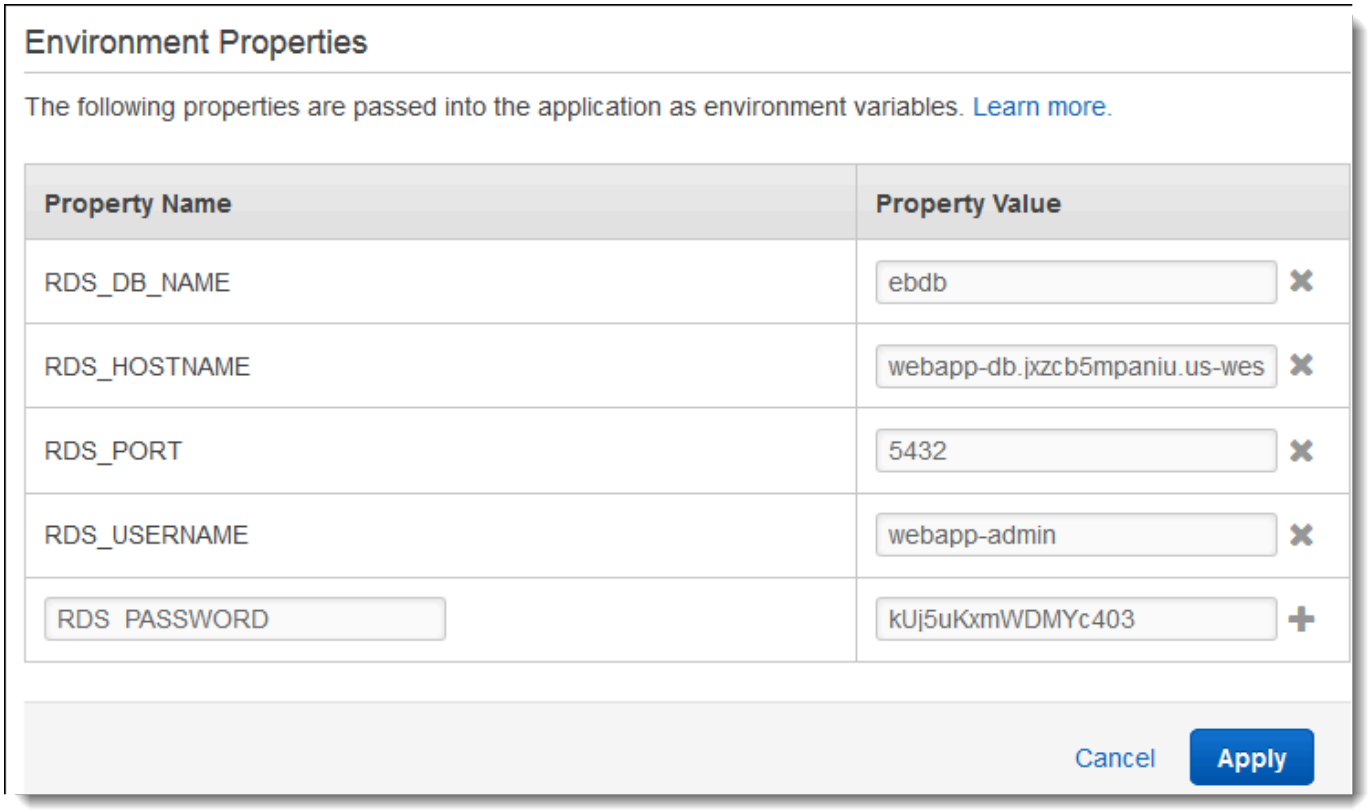
1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

   > **Note**
   >
   > 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇 Configuration (組態)。
4. 在更新、監控和日誌記錄組態類別中，選擇編輯。

5. 在 Environment Properties (環境屬性) 區段，定義應用程式建立連線字串所讀取的變數。為了與具備整合式 RDS 資料庫執行個體的環境相容，請使用下列名稱與值。您可以在 RDS 主控台中找到除了密碼以外的所有值。

屬性名稱	描述	屬性值
RDS_HOSTNAME	資料庫執行個體的主機名稱。	在 Amazon RDS 主控台：端點的連線能力和安全性索引標籤上。
RDS_PORT	資料庫執行個體接受連線的連接埠。預設值在不同資料庫引擎中有所差異。	在 Amazon RDS 主控台：連接埠的連線能力和安全性索引標籤上。
RDS_DB_NAME	資料庫名稱，**ebdb**。	在 Amazon RDS 主控台：資料庫名稱的組態索引標籤上。
RDS_USERNAME	您為資料庫設定的使用者名稱。	在 Amazon RDS 主控台：主要使用者名稱的組態索引標籤上。
RDS_PASSWORD	您為資料庫設定的密碼。	無法在 Amazon RDS 主控台中提供參考。



6. 若要儲存變更，請選擇頁面底部的儲存變更。

若您尚未將應用程式設計為讀取環境屬性並建構連線字串，請參閱下列語言特定的主題以取得說明：

- Java SE – 連線至資料庫 (Java SE 平台)
- Java 搭配 Tomcat – 連線至資料庫 (Tomcat 平台)
- Node.js – 連線至資料庫
- .NET – 連線至資料庫
- PHP – 使用 PDO 或 MySQLi 連接至資料庫
- Python – 連線至資料庫
- Ruby – 連線至資料庫

最後，根據您應用程式讀取環境變數的時間，您可能需要重新啟動環境中執行個體上的應用程式伺服器。

欲重新啟動您環境的應用程式伺服器

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 選擇 Actions (動作)，然後選擇 Restart App Server(s) (重新啟動應用程式伺服器)。

於 EC2 Classic 內啟動並連接至外部 Amazon RDS 執行個體

> **Important**
>
> Amazon EC2-Classic 將於 2022 年 8 月 15 日終止其標準支援。為避免工作負載中斷，建議您在此上述時間之前從 Amazon EC2 Classic 遷移至 VPC。我們也要求您日後不要在 Amazon EC2-Classic 啟動任何 AWS 資源，而是使用 Amazon VPC。如需詳細資訊，請參閱 從 EC2-Classic 移轉至 VPC 和 EC2-Classic 網路正在淘汰 - 本文介紹如何準備的部落格文章。

若您使用 EC2 Classic (無 VPC) 搭配 AWS Elastic Beanstalk，程序會稍微變更，因為安全群組的運作方式有所不同。在 EC2 Classic 中，資料庫執行個體無法使用 EC2 安全群組，因此會取得僅可搭配 Amazon RDS 運作的資料庫安全群組。

您可將規則新增至資料庫安全群組，此安全群組允許來自 EC2 安全群組的傳入存取。不過，您無法將資料庫安全群組連接至與您的環境相關聯的 Auto Scaling 群組。為了避免在資料庫安全群組與您的環境間建立相依性，您必須在 Amazon EC2 建立第三個安全群組。然後，您需要在資料庫安全群組中新增規則，以將傳入存取權限授予新安全群組。最後，您應將其指派至 Elastic Beanstalk 環境中的 Auto Scaling 群組。

> **Note**
>
> - 如果您是從 Elastic Beanstalk 建立，並隨後從 Beanstalk 環境解耦的資料庫開始，則可以略過第一組步驟，並以建立橋接安全群組下分組的步驟繼續進行。

- 如果您計劃為生產環境使用解耦的資料庫，請確認該資料庫使用的儲存類型是否適合您的工作負載。如需詳細資訊，請參閱《Amazon RDS 使用者指南》中的資料庫執行個體儲存體和修改資料庫執行個體。

欲於 EC2 Classic (無 VPC) 中啟動 RDS 執行個體

1. 開啟 RDS 管理主控台。
2. 選擇 Create database (建立資料庫)。
3. 繼續透過精靈進行。請記下您輸入下列選項的值：
   - Master username (主要使用者名稱)
   - Master Password (主密碼)
4. 當您進入 Network and Security (網路與安全) 設定的 Configure advanced settings (設定進階設定) 時，請選擇下列項目：
   - VPC – **`Not in VPC`**. 若此選項不可用，您的帳戶可能不支援 EC2-Classic，或者您可能已選擇僅可用於 VPC 的執行個體類型。
   - 可用區域 – **`No Preference`**
   - 資料庫安全群組 – **`Create new Security Group`**
5. 設定剩餘選項，然後選擇 Create database (建立資料庫)。請記下您輸入下列選項的值：
   - Database Name (資料庫名稱)
   - Database Port (資料庫連接埠)

在 EC2-Classic 中，您的資料庫執行個體具備資料庫安全群組，而非 VPC 安全群組。您無法將資料庫安全群組連接至您的 Elastic Beanstalk 環境。反之，您需要建立新的安全群組，您可以授權該安全群組存取資料庫執行個體並連接至您的環境。我們將其稱為橋接安全群組，並命名為 **`webapp-bridge`**。

欲建立橋接安全群組

1. 開啟 Amazon EC2 主控台。
2. 在導覽側列的 Network & Security (網路與安全) 下，選擇 Security Groups (安全群組)。
3. 選擇 Create Security Group (建立安全群組)。
4. 針對 Security group name (安全群組名稱)，輸入 **`webapp-bridge`**。

5. 針對 Description (描述)，輸入 **`Provide access to DB instance from Elastic Beanstalk environment instances.`**。
6. 對於 VPC (VPC)，請保留預設選擇。
7. 選擇 Create (建立)

接著，修改連接至資料庫執行個體的安全群組，以允許來自橋接安全群組的傳入流量。

修改 RDS 執行個體安全群組的傳入規則

1. 開啟 Amazon RDS 主控台。
2. 選擇 Databases (資料庫)。
3. 選擇您的資料庫執行個體名稱以檢視其詳細資料。
4. 在 Connectivity (連線) 區段中，位於 Security (安全性) 下方，會顯示與資料庫執行個體相關聯的安全群組。開啟連結以檢視 Amazon EC2 主控台內的安全群組。
5. 在安全群組詳細資訊中，將 Connection Type (連線類型) 設定為 EC2 Security Group (EC2 安全群組)。
6. 將 EC2 Security Group Name (EC2 安全群組名稱) 設定為您建立的橋接安全群組名稱。
7. 選擇 Authorize (授權)。

接著，請將橋接安全群組新增至執行環境。此程序必須透過其他連接的安全群組，重新佈建您環境中的所有執行個體。

欲將安全群組新增至您的環境

- 執行下列任意一項：
  - 使用 Elastic Beanstalk 主控台新增安全群組

    a. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。

    b. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

    > **Note**
    >
    > 如果您有許多環境，請使用搜尋列來篩選環境清單。

    c. 在導覽窗格中，選擇 Configuration (組態)。

    d. 在 Instances (執行個體) 組態類別中，選擇 Edit (編輯)。

e. 在 EC2 安全群組下，除了 Elastic Beanstalk 建立的執行個體安全群組，請選擇要連接到執行個體的安全群組。

f. 若要儲存變更，請選擇頁面底部的儲存變更。

g. 閱讀警告的內容，然後選擇 Confirm (確認)。

- 若要使用組態檔案新增安全群組，請使用 `securitygroup-addexisting.config` 範例檔案。

接著，使用環境屬性，將連線資訊傳送至環境。在您透過 Elastic Beanstalk 主控台將資料庫執行個體新增到環境時，Elastic Beanstalk 會使用諸如 RDS_HOSTNAME 等環境屬性，將連線資訊傳遞至您的應用程式。您可使用相同屬性，讓您為整合式資料庫執行個體和外部資料庫執行個體使用相同的應用程式程式碼。或者，您也可以選擇自己的屬性名稱。

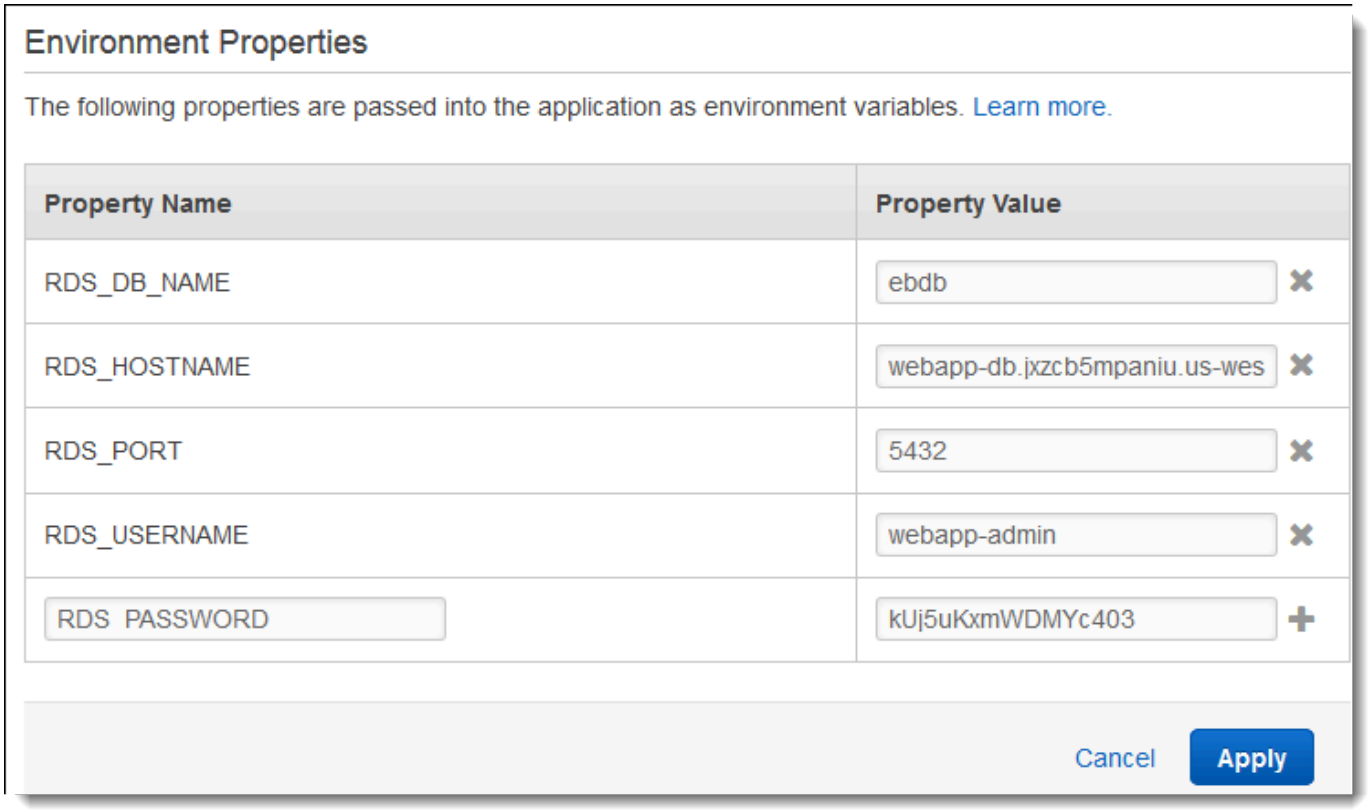
欲設定環境屬性

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 在導覽窗格中，選擇 Configuration (組態)。
4. 在更新、監控和日誌記錄組態類別中，選擇編輯。
5. 在 Environment Properties (環境屬性) 區段，定義應用程式建立連線字串所讀取的變數。為了與具備整合式 RDS 執行個體的環境相容，請使用下列項目：

   - RDS_DB_NAME – 在 Amazon RDS 主控台中的 DB Name (資料庫名稱)。
   - RDS_USERNAME – 您將資料庫新增至環境時輸入的 Master Username (主要使用者名稱)。
   - RDS_PASSWORD – 您將資料庫新增至環境時輸入的 Master Password (主要密碼)。
   - RDS_HOSTNAME – 在 Amazon RDS 主控台中的資料庫執行個體的 Endpoint (端點)。
   - RDS_PORT – 在 Amazon RDS 主控台中的 Port (連接埠)。



6. 選擇 Apply (套用)

若您尚未將應用程式設計為讀取環境屬性並建構連線字串，請參閱下列語言特定的主題以取得說明：

- Java SE – 連線至資料庫 (Java SE 平台)
- Java 搭配 Tomcat – 連線至資料庫 (Tomcat 平台)
- Node.js – 連線至資料庫
- .NET – 連線至資料庫
- PHP – 使用 PDO 或 MySQLi 連接至資料庫
- Python – 連線至資料庫
- Ruby – 連線至資料庫

最後，根據您應用程式讀取環境變數的時間，您可能需要重新啟動環境中執行個體上的應用程式伺服器。

重新啟動環境的應用程式伺服器

1. 開啟 Elastic Beanstalk 主控台，然後在 Regions (區域) 清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇 Environments (環境)，然後在清單中選擇您環境的名稱。

> **Note**
>
> 如果您有許多環境，請使用搜尋列來篩選環境清單。

3. 選擇 Actions (動作)，然後選擇 Restart App Server(s) (重新啟動應用程式伺服器)。

將 Amazon RDS 憑證存放在 AWS Secrets Manager 中

AWS Secrets Manager 透過提供儲存和擷取加密憑證的功能，協助您改善安全狀態。將憑證儲存在 Secrets Manager 中有助於避免任何可以檢查您的應用程式或相關元件的人可能造成的危害。您的程式碼可以對 Secrets Manager 服務進行執行期呼叫，以動態擷取憑證。Secrets Manager 還提供了諸如執行期語言的用戶端秘密快取元件等功能，其中包括 Python、Go 和 Java 等語言。

如需詳細資訊，請參閱 AWS Secrets Manager 使用者指南中的以下主題：

- Amazon RDS 如何使用 AWS Secrets Manager
- 建立 AWS Secrets Manager 資料庫秘密
- 從 AWS Secrets Manager 中擷取秘密

清除外部 Amazon RDS 執行個體

當您將外部 Amazon RDS 執行個體連接至 Elastic Beanstalk 環境時，資料庫執行個體並非依存於您的環境生命週期，因此不會在您終止環境時刪除該資料庫執行個體。為了確保可能已存放在資料庫執行個體的個人資訊不被不必要的保留，請刪除任何不再需要的記錄。或者，請刪除資料庫執行個體。

將 Elastic Beanstalk 與 Amazon S3 搭配使用

Amazon Simple Storage Service (Amazon S3) 提供具備高耐用性、容錯能力的資料儲存體。

Elastic Beanstalk 為您建立環境的各個區域中建立名為 `elasticbeanstalk-region-account-id` 的 Amazon S3 儲存貯體。Elastic Beanstalk 會使用此儲存貯體來儲存您應用程式正確操作時所需的物件 (如暫存組態檔)。

Elastic Beanstalk 不會開啟其建立的 Amazon S3 儲存貯體的預設加密功能。即是在預設情況下，物件會以未加密的狀態存放在儲存貯體中 (任何獲得授權的使用者均可存取)。有些應用程式儲存物件時，需要將所有物件加密 - 例如存放在硬碟機、資料庫等處時 (也稱為靜態加密)。如果您有此要求，可將您帳戶的儲存貯體設定為預設加密。如需詳細資訊，請參閱《Amazon Simple Storage Service 使用者指南》中的 S3 儲存貯體的 Amazon S3 預設加密。

Elastic Beanstalk Amazon S3 儲存貯體的內容

下表列出一些 Elastic Beanstalk 存放在您的 `elasticbeanstalk-*` Amazon S3 儲存貯體的物件。此表也顯示哪些物件必須手動刪除。為了避免不必要的儲存成本並確保個人資訊不被保留，當您不再需要時，請確認手動刪除這些物件。

物件	當儲存時？	當刪除時？
應用程式版本	當您建立環境或部署您的應用程式程式碼於現有環境中，Elastic Beanstalk 將應用程式版本存放於 Amazon S3，並與環境建立關聯。	在應用程式刪除時，並根據 版本生命週期。
原始碼套件	當您使用 Elastic Beanstalk 主控台或 EB CLI 上傳新的應用程式版本時，Elastic Beanstalk 存放其副本於 Amazon S3 中，並將它設定為您的環境原始碼套件。	手動。當您刪除應用程式版本，您可以選擇 Delete versions from Amazon S3 (從 Amazon S3 刪除版本) 以同時刪除相關的原始碼套件。如需詳細資訊，請參閱管理應用程式版本。
自訂平台	當您建立自訂平台時，Elastic Beanstalk 暫時存放相關資料於 Amazon S3 中。	成功完成的自訂平台建立。
日誌檔	您可以請求 Elastic Beanstalk 擷取執行個體的日誌檔案 (結尾或套件日誌)，並將這些檔案存放在 Amazon S3。您也可以啟用日誌輪換，且在日誌輪換後，設定您的環境自動發佈日誌至 Amazon S3。	結尾和套件日誌；在建立後的 15 分鐘。 旋轉日誌： 手動。
已儲存的組態	手動。	手動。

刪除 Elastic Beanstalk Amazon S3 儲存貯體中的物件

當您終止環境或刪除應用程式時，Elastic Beanstalk 會從 Amazon S3 刪除最相關的物件。為了盡可能降低執行中應用程式的儲存成本，定期刪除您的應用程式不需要的物件。此外，請留意需要手動刪除的物件，如列於 Elastic Beanstalk Amazon S3 儲存貯體的內容 中的。為了確保私有資訊被不必要的保留，當您不再需要時，刪除這些物件。

- 刪除您不再希望用於您的應用程式的應用程式版本。當您刪除應用程式版本時，您可以選取Delete versions from Amazon S3 (從 Amazon S3 刪除版本)，以同時刪除相關的原始碼套件 - 當您部署應用程式或上傳應用程式版本時，Elastic Beanstalk 上傳到 Amazon S3 您的應用程式的原始程式碼和組態檔案的副本。若要了解如何刪除應用程式版本的詳細資訊，請參閱 管理應用程式版本。
- 刪除您不需要的輪換日誌。或者，下載或將他們移至 Amazon S3 Glacier 以進行進一步的分析。
- 刪除您不會再於任何環境中使用的已儲存組態。

刪除 Elastic Beanstalk Amazon S3 儲存貯體

當 Elastic Beanstalk 建立儲存貯體時，它也會建立套用至新儲存貯體的儲存貯體政策。此政策有兩個用途：

- 允許環境寫入儲存貯體。
- 防止意外刪除儲存貯體。

由於 Elastic Beanstalk 適用於它為您的環境建立的儲存貯體的政策，因此不允許您刪除這些儲存貯體，除非您先刻意刪除儲存貯體政策。可從 Amazon S3 主控台的儲存貯體屬性的許可區段，刪除儲存貯體政策。

> **警告**
>
> 如果您刪除 Elastic Beanstalk 在您的帳戶中建立的儲存貯體，而且在對應的區域中，仍具備現有的應用程式和執行的環境，您的應用程式可能會停止正常運作。例如：
>
> - 當環境可擴展時，Elastic Beanstalk 應該能夠找到於 Amazon S3 儲存貯體中的環境應用程式版本，並使用它來啟動新的 Amazon EC2 執行個體。
> - 當您建立自訂平台時，Elastic Beanstalk 會在建立過程中使用臨時的 Amazon S3 儲存。

我們建議從您的 Elastic Beanstalk Amazon S3 儲存貯體刪除特定不必要的物件，而不是刪除整個儲存貯體。

刪除 Elastic Beanstalk 儲存貯體 (主控台)

Amazon S3 使用者指南中的刪除 S3 儲存貯體也描述了刪除 S3 儲存貯體的一般程序。由於我們要在下列程序中刪除 Elastic Beanstalk 建立的儲存貯體，因此我們包含了其他步驟來首先刪除儲存貯體政策。

1. 開啟 Amazon S3 主控台。
2. 選擇儲存貯體名稱，以開啟 Elastic Beanstalk 儲存貯體的頁面。
3. 選擇 許可 標籤。
4. 選擇 Bucket Policy (儲存貯體政策)。
5. 選擇 Delete (刪除)。
6. 返回 Amazon S3 主控台的主要頁面，然後刪除 Elastic Beanstalk 儲存貯體。
7. 選擇 Delete Bucket (刪除儲存貯體)。
8. 在文字欄位中輸入儲存貯體名稱以確認您要刪除該儲存貯體，然後選擇刪除儲存貯體。

搭配 Amazon VPC 使用 Elastic Beanstalk

您可以使用 Amazon Virtual Private Cloud (Amazon VPC) 為您的 Elastic Beanstalk 應用程式及相關 AWS 資源建立安全的網路。建立您的環境時，您可以選擇哪些 VPC、子網路和安全群組是用於您的應用程式執行個體和負載平衡器。您可以使用任何 VPC 組態，只要符合以下需求即可。

VPC 要求

- 網際網路存取 – 執行個體可透過以下其中一種方法存取網際網路：
  - 公有子網路 – 執行個體具備公有 IP 地址，並使用網際網路閘道存取網際網路。
  - 私有子網路 – 執行個體使用 NAT 裝置存取網際網路。

> **Note**
>
> 如果您將 VPC 中的 VPC 端點設定為連線到 `elasticbeanstalk` 和 `elasticbeanstalk-health` 服務，則網際網路存取是選擇性的，而且只有在應用程式特別需要時才需要。如果沒有 VPC 端點，您的 VPC 必須能夠存取網際網路。
> Elastic Beanstalk 為您設定的預設 VPC 提供網際網路存取。

Elastic Beanstalk 不支援使用 `HTTPS_PROXY` 等代理設定來設定 Web 代理。

- NTP – 您 Elastic Beanstalk 環境中的執行個體會使用網路時間協定 (NTP) 同步系統時鐘。若執行個體上無法於 UDP 連接埠 123 通訊，時鐘可能會無法同步，並於 Elastic Beanstalk 運作狀態報告出現問題。請確認您的 VPC 安全群組和網路 ACL，允許連接埠 123 上的 UDP 流量出入，以避免這些問題。

elastic-beanstalk-samples 儲存庫提供 AWS CloudFormation 範本，可讓您用來建立 VPC 以搭配 Elastic Beanstalk 環境使用。

若要使用 AWS CloudFormation 範本來建立資源

1. 複製範本儲存庫，或是使用 README 中的連結下載範本。
2. 開啟 AWS CloudFormation 主控台。
3. 選擇 Create Stack (建立堆疊)。
4. 選擇 Upload a template to Amazon S3 (上傳範本到 Amazon S3)。
5. 選擇 Upload file (上傳檔案) 並從您的本機機器上傳範本檔案。
6. 選擇 Next (下一步) 並依照說明操作來使用範本中的資源建立堆疊。

堆疊建立完成後，檢查 Outputs (輸出) 索引標籤，尋找 VPC ID 和子網路 ID。在新的環境精靈網路組態類別中使用這些設定 VPC。

主題

- 公有 VPC
- 公有/私有 VPC
- 私有 VPC
- 範例：透過堡壘主機啟動 VPC 內的 Elastic Beanstalk 應用程式

- 範例：透過 Amazon RDS 在 VPC 中啟動 Elastic Beanstalk
- 搭配 VPC 端點使用 Elastic Beanstalk

公有 VPC

AWS CloudFormation 範本 – vpc-public.yaml

設定 (負載平衡)

- 負載平衡器可見度 – 公有
- 負載平衡器子網路 – 兩個公有子網路
- 執行個體公有 IP – 已啟用
- 執行個體子網路 – 兩個公有子網路
- 執行個體安全群組 – 新增預設安全群組

設定 (單一執行個體)

- 執行個體子網路 – 其中一個公有子網路
- 執行個體安全群組 – 新增預設安全群組

基本僅公有 VPC 版面配置包含一或多個公有子網路、網際網路閘道和預設的安全群組，讓 VPC 中各資源之間存在流量。當您在 VPC 中建立環境時，Elastic Beanstalk 會根據環境類型建立額外的資源。

VPC 資源

- 單一執行個體 &endash; Elastic Beanstalk 會針對應用程式執行個體建立安全群組，允許來自連接埠 80 的網際網路流量，並為此執行個體指派彈性 IP，為其指定一個公有 IP 地址。環境的網域名稱解析成執行個體的公有 IP 地址。
- 負載平衡 &endash; Elastic Beanstalk 為負載平衡器建立安全群組，允許來自連接埠 80 的網際網路流量，並會為應用程式執行個體建立安全群組，允許來自負載平衡器安全群組的流量。環境的網域名稱解析成負載平衡器的網域名稱。

這與 Elastic Beanstalk 在您使用預設 VPC 時管理聯網的方式相似。公有子網路中的安全取決於負載平衡器以及 Elastic Beanstalk 建立的執行個體安全群組。這是最便宜的配置，因為它不需要 NAT 閘道。

公有/私有 VPC

AWS CloudFormation 範本 – vpc-privatepublic.yaml

設定 (負載平衡)

- 負載平衡器可見度 – 公有
- 負載平衡器子網路 – 兩個公有子網路
- 執行個體公有 IP – 已停用
- 執行個體子網路 – 兩個私有子網路
- 執行個體安全群組 – 新增預設安全群組

如需額外的安全性，可新增私有子網路到您的 VPC 以建立公有-私有配置。此配置需要公有子網路中的負載平衡器和 NAT 閘道，讓您執行應用程式執行個體、資料庫和私有子網路中的任何其他資源。私有子網路中的執行個體只能透過負載平衡器和 NAT 閘道與網際網路通訊。

私有 VPC

AWS CloudFormation 範本 – vpc-private.yaml

設定 (負載平衡)

- 負載平衡器可見度 – 私有
- 負載平衡器子網路 – 兩個私有子網路
- 執行個體公有 IP – 已停用
- 執行個體子網路 – 兩個私有子網路
- 執行個體安全群組 – 新增預設安全群組

對於不應被網際網路存取的內部應用程式，您可以在私有子網路執行所有項目，並設定負載平衡器為朝向內部 (變更 Load balancer visibility (負載平衡器可見度) 至 Internal (內部))。此範本會建立不含公有子網路和網際網路閘道的 VPC。使用應用程式的這個配置，其只能從相同的 VPC 或連接的 VPN 存取。

在私有 VPC 中執行 Elastic Beanstalk 環境

當您在私有 VPC 中建立 Elastic Beanstalk 環境時，環境無法存取網際網路。您的應用程式可能需要存取 Elastic Beanstalk 服務或其他服務。您的環境可能會使用增強型運作狀態報告，在此情況下，環境

執行個體會將運作狀態資訊傳送至增強型運作狀態服務。環境執行個體上的 Elastic Beanstalk 程式碼將流量發送到其他 AWS 服務，並將其他流量發送到非 AWS 端點 (例如，為您的應用程式下載相依性套件)。以下是在此情況下，您可能需要採取的一些步驟，以確保您的環境正常運作。

- 「為 Elastic Beanstalk 設定 VPC 端點」– Elastic Beanstalk 及其增強型運作狀態服務支援 VPC 端點，以確保這些服務的流量保持在 Amazon 網路中，且不需要網際網路存取。如需更多詳細資訊，請參閱 the section called “VPC 端點”。
- 為其他服務設定 VPC 端點 – Elastic Beanstalk 執行個體會代您將流量傳送到其他數個 AWS 服務：Amazon Simple Storage Service (Amazon S3)、Amazon Simple Queue Service (Amazon SQS)、AWS CloudFormation 和 Amazon CloudWatch Logs。您也必須為這些服務設定 VPC 端點。如需 VPC 端點的詳細資訊 (包括每個服務連結)，請參閱《Amazon VPC 使用者指南》中的 VPC 端點。

  > **Note**
  >
  > 有些 AWS 服務 (包括 Elastic Beanstalk) 僅在有限數量的 AWS 區域中支援 VPC 端點。當您設計私有 VPC 解決方案時，請確認此處提及的 Elastic Beanstalk 與其他相依服務支援您選擇 AWS 區域中的 VPC 端點。

- 「提供私有 Docker 影像」– 在 Docker 環境中，環境執行個體上的程式碼可能會嘗試在建立環境期間，從網際網路提取已設定的 Docker 影像並失敗。如要避免此失敗，請在您的環境上建置自訂 Docker 影像，或是使用存放在 Amazon Elastic Container Registry (Amazon ECR) 中的 Docker 影像，然後為 Amazon ECS 服務設定 VPC 端點。
- 啟用 DNS 名稱 – 環境執行個體上的 Elastic Beanstalk 程式碼會使用其公有端點，將流量傳送到所有 AWS 服務。若要確保此流量經過，請在設定所有界面 VPC 端點時選擇 Enable DNS name (啟用 DNS 名稱) 選項。這會在 VPC 中新增 DNS 項目，此項目會將公有服務端點映射至界面 VPC 端點。

  > **Important**
  >
  > 如果您的 VPC 並非私有且具有公有網際網路存取權，而且任何 VPC 端點的 Enable DNS name (啟用 DNS 名稱) 已停用，則個別服務的流量會透過公用網際網路傳輸。這可能不是您想要的。使用私有 VPC 很容易偵測到此問題，因為它可以防止此流量發生，並且您會收到錯誤。但是，對於公有面向的 VPC，則不會有任何指示。

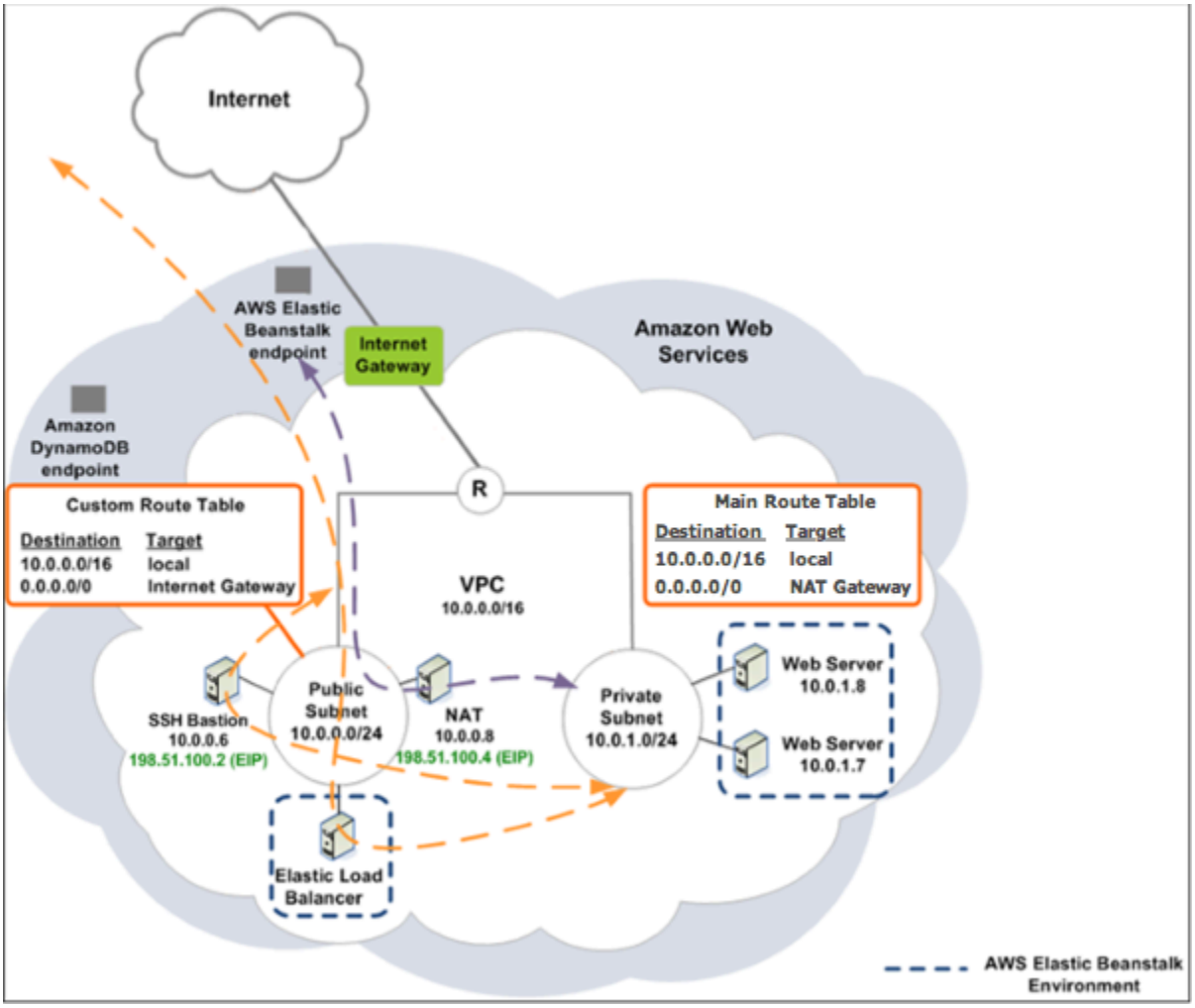
- 「包含應用程式相依性」– 如果您的應用程式具有相依性 (例如語言執行時間套件)，則其可能會在環境建立期間嘗試從網際網路下載進行安裝並失敗。為了避免這種失敗，請在應用程式的原始碼套件中包含所有相依性套件。

- 「使用目前的平台版本」– 請確定您的環境使用 2020 年 2 月 24 日以後發行的平台版本。具體來說，使用在這兩個更新之一或之後發行的平台版本：Linux Update 2020-02-28、Windows Update 2020-02-24。

> **Note**
>
> 需要更新平台版本的原因是，舊版本有一個可能阻止由 Enable DNS name (啟用 DNS 名稱) 選項所建立的 DNS 項目為 Amazon SQS 正常運作的問題。

範例：透過堡壘主機啟動 VPC 內的 Elastic Beanstalk 應用程式

如果您的 Amazon EC2 執行個體位於私有子網路內，將無法從遠端進行連線。欲連接至您的執行個體，您可在公有子網路內設定堡壘伺服器做為代理。例如，您可於公有子網路內設定 SSH 連接埠轉寄站或 RDP 閘道，代理從您自己的網路流向資料庫伺服器的流量。本章節的範例說明如何透過私有和公有子網路建立 VPC。執行個體位於私有子網路內，而堡壘主機、NAT 閘道和負載平衡器則位於公有子網路內。您的基礎設施將與下圖類似。



如要使用堡壘主機在 VPC 內部部署 Elastic Beanstalk 應用程式，請完成下列小節所述的步驟。

步驟

- 建立包含公有和私有子網路的 VPC
- 建立並設定堡壘主機安全群組
- 更新執行個體安全群組
- 建立堡壘主機

建立包含公有和私有子網路的 VPC

在 公有/私有 VPC 完成所有程序。部署應用程式時，您必須為執行個體指定 Amazon EC2 金鑰對，以便從遠端連線。如需指定執行個體金鑰對的詳細資訊，請參閱 您 Elastic Beanstalk 環境的 Amazon EC2 執行個體。

建立並設定堡壘主機安全群組

建立堡壘主機的安全群組，並新增規則允許來自網際網路的傳入 SSH 流量，以及傳出至內含 Amazon EC2 執行個體私有子網路的 SSH 流量。

建立堡壘主機安全群組

1. 在 https://console.aws.amazon.com/vpc/ 開啟 Amazon VPC 主控台。
2. 在導覽窗格中，選擇 Security Groups (安全群組)。
3. 選擇 Create Security Group (建立安全群組)。
4. 在 Create Security Group (建立安全群組) 對話方塊中，輸入如下內容並選擇 Yes, Create (是，建立)。

   Name tag (名稱標籤) (選用)

   輸入安全群組的名稱標籤。

   Group name (群組名稱):

   輸入安全群組的名稱。

   描述

   輸入安全群組的描述。

   VPC

   選取您的 VPC。

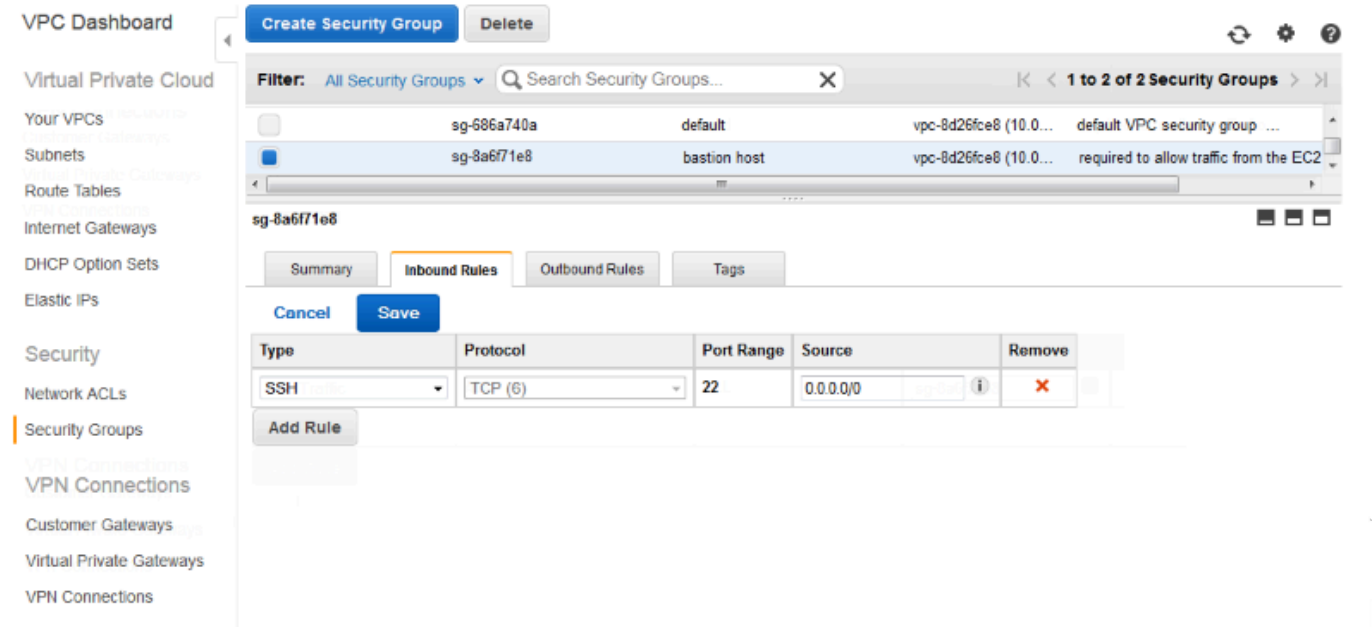
   安全群組便會建立，並出現在 Security Groups (安全群組) 頁面上。請注意其會有 ID (例如：`sg-xxxxxxxx`)。您可能必須按一下頁面右上角的 Show/Hide (顯示/隱藏) 來開啟 Group ID (群組 ID) 資料行。

欲設定堡壘主機安全群組

1. 在安全群組清單中，選取您剛為堡壘主機建立的安全群組核取方塊。
2. 在 Inbound Rules (傳入規則) 標籤上，選擇 Edit (編輯)。
3. 視需要選擇 Add another rule (新增其他規則)。
4. 若您的防禦主機為 Linux 執行個體，在 Type (類型) 底下選取 SSH (SSH)。

若您的防禦主機為 Windows 執行個體，在 Type (類型) 底下選取 RDP (RDP)。

5. 在 Source (來源) 欄位輸入所需來源 CIDR 範圍，然後選擇 Save (儲存)。

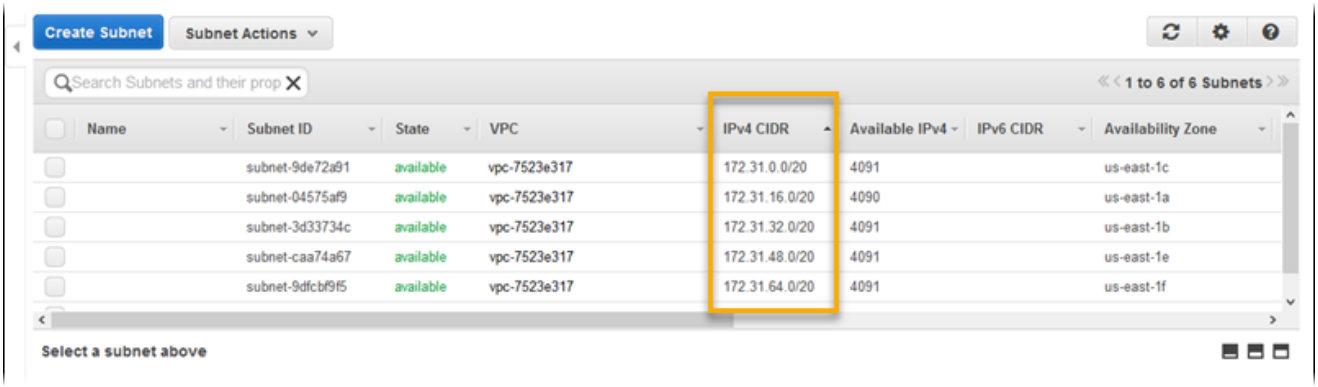


6. 在 Outbound Rules (傳出規則) 索引標籤上，選擇 Edit (編輯)。
7. 視需要選擇 Add another rule (新增其他規則)。
8. 在 Type (類型) 下，選取您指定類型的傳入規則。
9. 在 Source (來源) 欄位中，輸入 VPC 私有子網路中主機子網路的 CIDR 範圍。

   尋找：

   a. 在 https://console.aws.amazon.com/vpc/ 開啟 Amazon VPC 主控台。
   b. 在導覽窗格中，選擇 Subnets (子網路)。
   c. 請注意，每個 Availability Zone (可用區域) 在 IPv4 CIDR (IPv4 CIDR) 下的值，其中有您希望防禦主機橋接的主機。

> **Note**
>
> 如果您在多個可用區域擁有主機，請針對這些可用區域的每一個建立傳出規則。



10. 選擇 Save (儲存)。

更新執行個體安全群組

您的執行個體安全群組預設不允許傳入流量。Elastic Beanstalk 會修改執行個體的預設群組以允許 SSH 流量，但如果您的執行個體為 Windows 執行個體，則必須自行修改自訂執行個體的安全群組，以允許 RDP 流量。

欲更新 RDP 執行個體安全群組

1. 在安全群組清單中，選取執行個體安全群組的核取方塊。
2. 在 Inbound (傳入) 標籤上，選擇 Edit (編輯)。
3. 視需要選擇 Add another rule (新增其他規則)。
4. 輸入下列值，然後選擇 Save (儲存)。

   類型。

   ```
   RDP
   ```

   Protocol (協定) – 。

   ```
   TCP
   ```

   連接埠範圍

   ```
   3389
   ```

   來源

   輸入防禦主機安全群組的 ID (如 `sg-8a6f71e8`)，並選擇 Save (儲存)。

建立堡壘主機

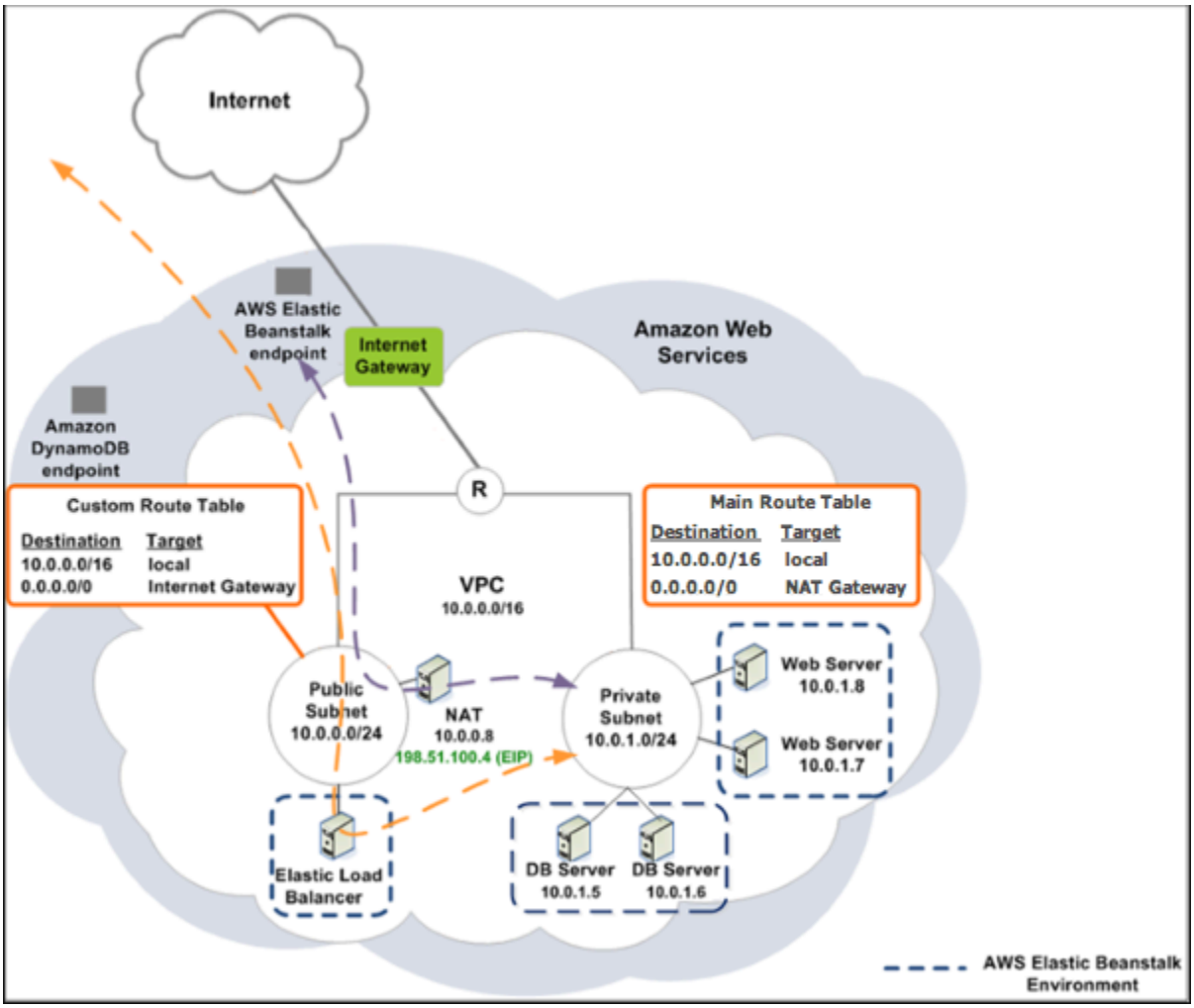
如要建立堡壘主機，您可以在您的公有子網路中啟動將做為堡壘主機的 Amazon EC2 執行個體。

如需在私有子網路中為 Windows 執行個體設定堡壘主機的詳細資訊，請參閱使用堡壘伺服器控制 EC2 執行個體的網路存取。

如需在私有子網路中為 Linux 執行個體設定堡壘主機的詳細資訊，請參閱安全連線至在私有 Amazon VPC 內執行的 Linux 執行個體。

範例：透過 Amazon RDS 在 VPC 中啟動 Elastic Beanstalk

本節會逐步引導您完成任務，使用 NAT 閘道於 VPC 中透過 Amazon RDS 部署 Elastic Beanstalk 應用程式。您的基礎設施將與下圖類似。



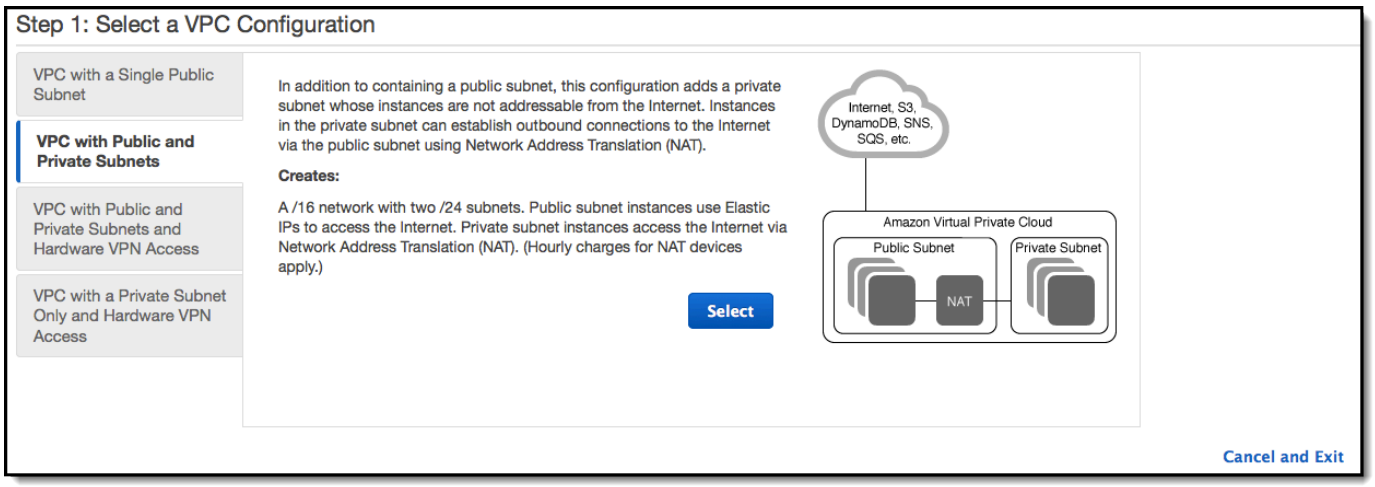
> **Note**
>
> 若您尚未搭配應用程式使用資料庫執行個體，請嘗試將資料庫執行個體新增至測試環境，然後連線至外部資料庫執行個體，之後再將 VPC 組態新增至組合。

建立包含公有和私有子網路的 VPC

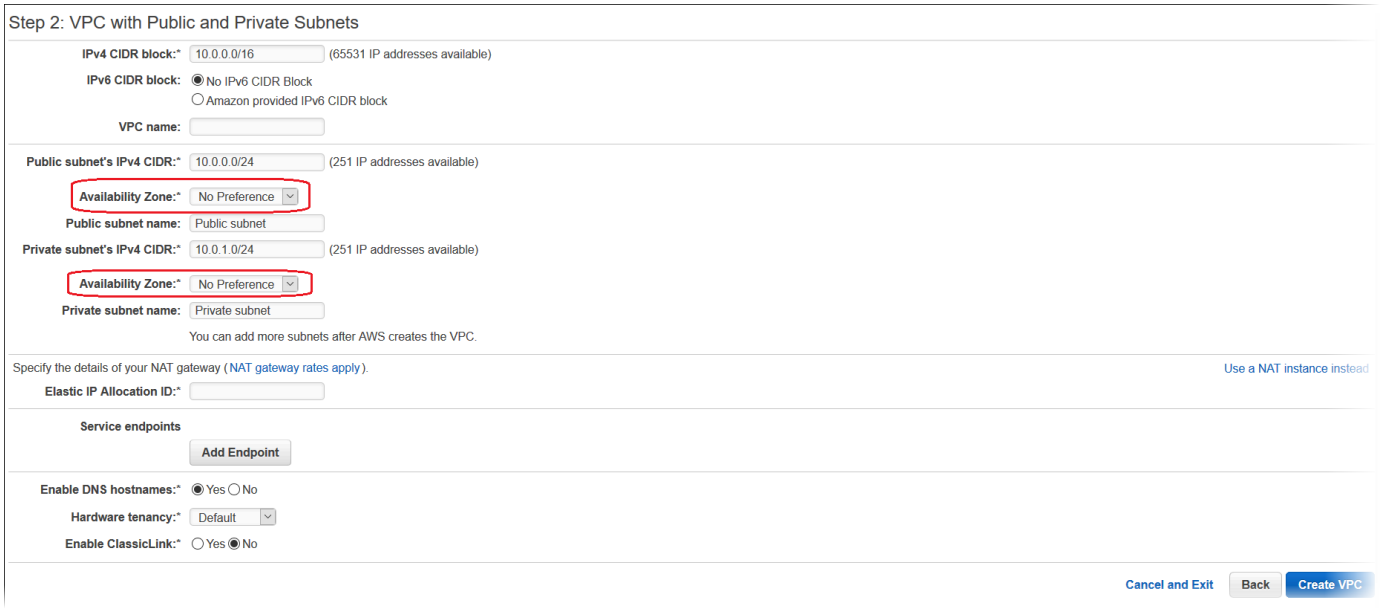
您可以使用 Amazon VPC 主控台來建立 VPC。

建立 VPC

1. 請登入 Amazon VPC 主控台。
2. 在導覽窗格中，選擇 VPC dashboard (VPC 儀表板)。然後選擇 Create VPC (建立 VPC)。
3. 選擇 VPC with Public and Private Subnets (具公有及私有子網路的 VPC)、然後選擇 Select (選取)。



4. 您的 Elastic Load Balancing 負載平衡器及您的 Amazon EC2 執行個體必須在同一個可用區域中，才能互相通訊。從各 Availability Zone (可用區域) 清單中選擇同一個可用區域。

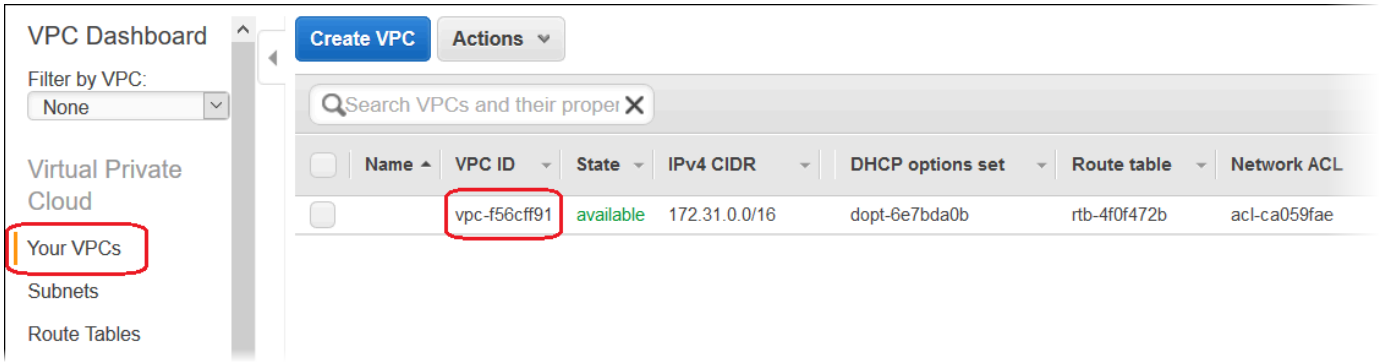


5. 為您的 NAT 閘道選擇彈性 IP 地址。
6. 選擇 Create VPC (建立 VPC)。

   精靈便會開始建立您的 VPC、子網路及網際網路閘道。同時它也會更新主路由表及建立自訂路由表。最後，精靈會在公有子網路中建立 NAT 閘道。

   > **Note**
   >
   > 您可以選擇在公有子網路中啟動 NAT 執行個體，而非 NAT 閘道。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的案例 2：具有公有及私有子網路 (NAT) 的 VPC。

7. 在 VPC 成功建立後，您便會取得 VPC ID。您在下一個步驟中需要使用到此數值。如要檢視您的 VPC ID，請選擇 Amazon VPC 主控台左側窗格中的 Your VPCs (您的 VPC)。

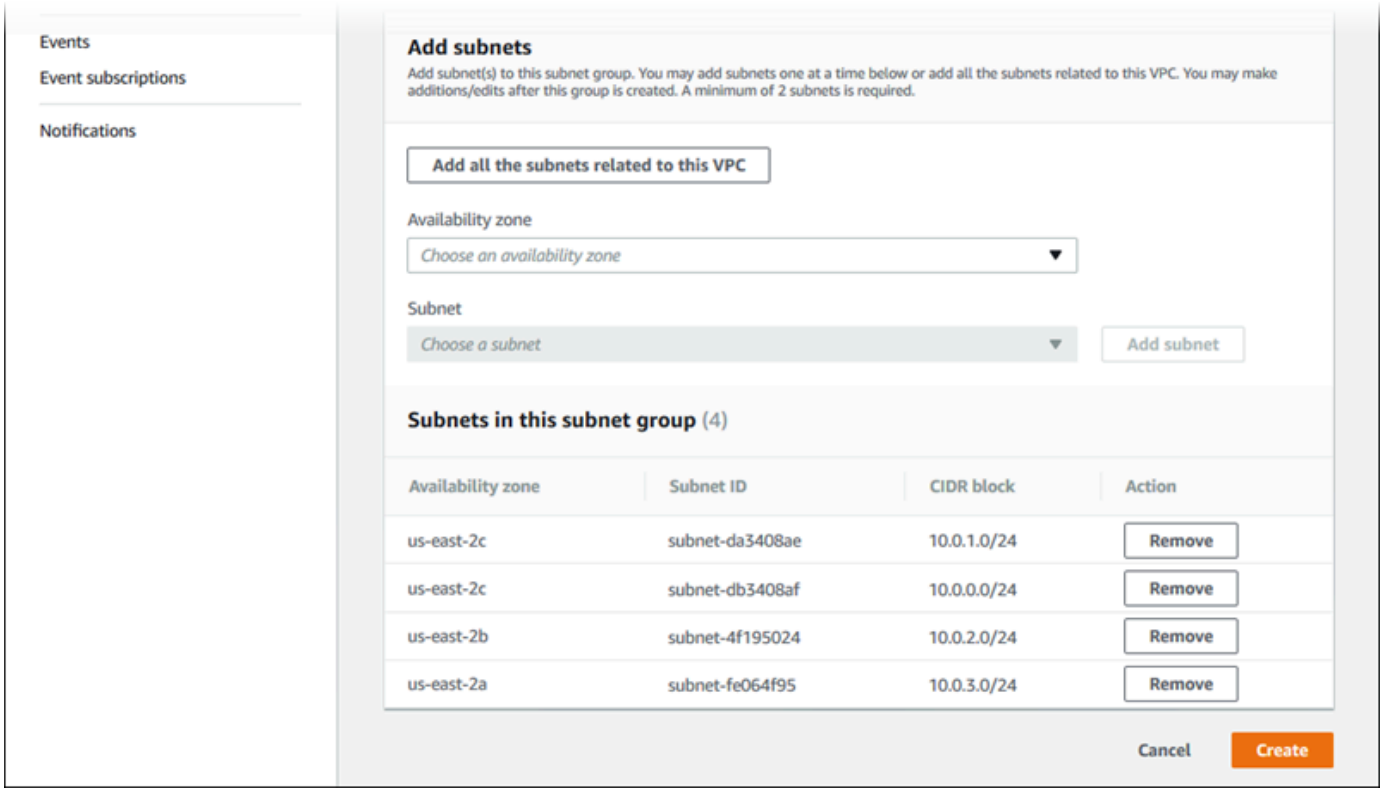


建立資料庫子網路群組

VPC 的資料庫子網路群組是子網路的集合 (通常為私有)，可為您的後端 RDS 資料庫執行個體指定。各個資料庫子網路群組在特定 AWS 區域中，都應在每個可用區域至少有一個子網路。若要進一步了解，請參閱在您的 VPC 中建立子網路。

建立資料庫子網路群組

1. 開啟 Amazon RDS 主控台。
2. 在導覽窗格中選擇 Subnet groups (子網路群組)。
3. 選擇 Create DB Subnet Group (建立資料庫子網路群組)。
4. 選擇 Name (名稱)，然後輸入您資料庫子網路群組的名稱。
5. 選擇 Description (描述)，然後說明您的資料庫子網路群組。
6. 對於 VPC (VPC)，選擇先前建立的 VPC 的 ID。
7. 在 Add subnets (新增子網路) 中，選擇 Add all the subnets related to this VPC (新增與此 VPC 相關的所有子網路)。



8. 完成時，選擇 Create (建立映像)。

您新的資料庫子網路群組會顯示在 Amazon RDS 主控台的子網路群組清單中。您可加以選擇，以在頁面底部的詳細資訊窗格顯示詳細資訊，例如與此群組相關聯的所有子網路。

部署到 Elastic Beanstalk

在您設定完 VPC 之後，您便可以在其中建立您的環境，並將您的應用程式部署到 Elastic Beanstalk。您可以使用 Elastic Beanstalk 主控台，或是使用 AWS 工具組、AWS CLI、EB CLI 或 Elatic Beanstalk API 來執行此作業。若您使用 Elastic Beanstalk 主控台，只需要上傳您的 `.war` 或 `.zip` 檔案，並在精靈中選取 VPC 設定即可。Elastic Beanstalk 接著會在您的 VPC 中建立環境，並部署您的應用程式。或者，您也可以使用 AWS 工具組、AWS CLI、EB CLI 或 Elastic Beanstalk API 來部署您的應用程式。若要執行此作業，您需要在組態檔中定義您的 VPC 選項設定，並與您的來源套件組合一同部署此檔案。本主題提供同時兩種方法的說明。

使用 Elastic Beanstalk 主控台部署

Elastic Beanstalk 主控台會帶您逐步在 VPC 內建立您的新環境。您需要提供 `.war` 檔案 (適用於 Java 應用程式) 或 `.zip` 檔案 (適用於所有其他應用程式)。在 Elastic Beanstalk 環境精靈的 VPC Configuration (VPC 組態) 頁面上，您必須選取下列選項：

VPC

選取您的 VPC。

VPC 安全群組

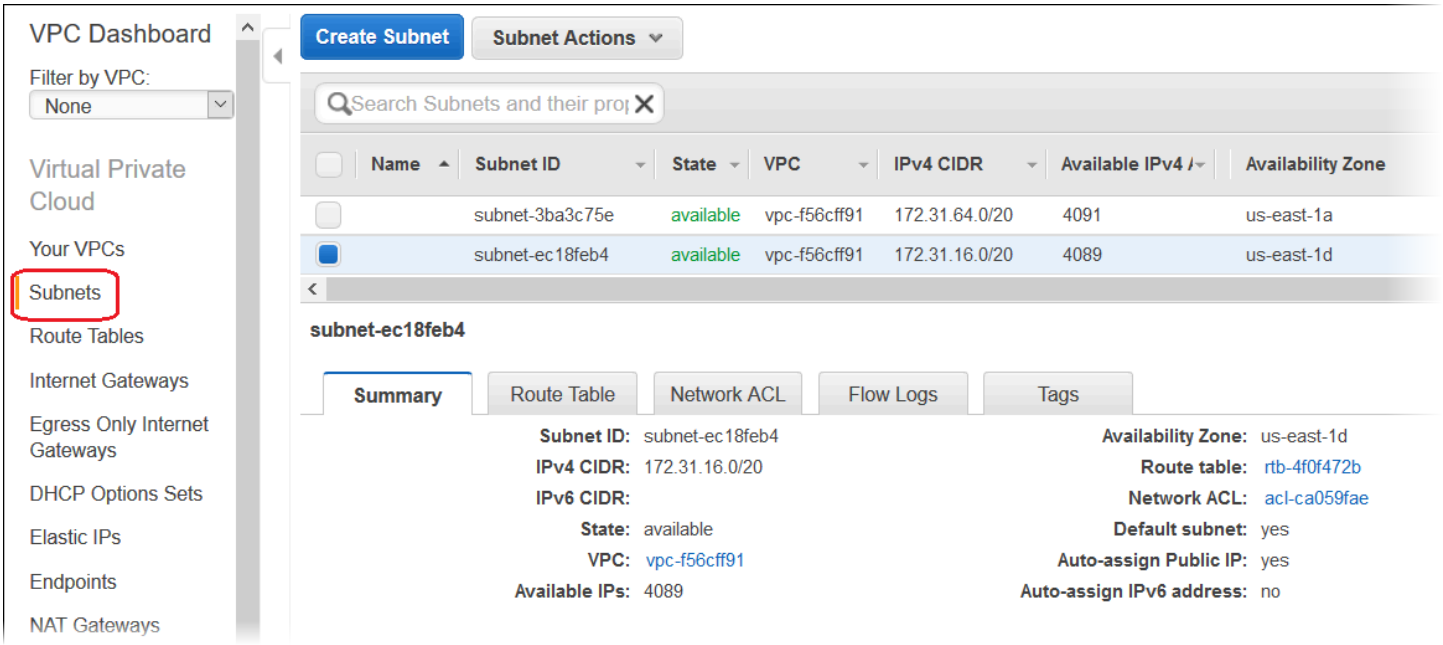
選取您在上方說明中建立的執行個體安全群組。

ELB visibility (ELB 可見性

若您的負載平衡器需可公開使用，請選取 `External`；若您的負載平衡器僅需可在您的 VPC 中使用，請選取 `Internal`。

選取您負載平衡器及 EC2 執行個體的子網路。請確認您已為您的負載平衡器選取公有子網路，並為您的 Amazon EC2 執行個體選取了私有子網路。根據預設，VPC 建立精靈會在 `10.0.0.0/24` 中建立公有子網路，並在 `10.0.1.0/24` 中建立私有子網路。

您可以藉由在 Amazon VPC 主控台中選擇 Subnets (子網路) 來檢視您的子網路 ID。



使用 AWS 工具組、EB CLI、AWS CLI 或 API 來進行部署

當您使用 AWS 工具組、EB CLI、AWS CLI 或 API 將您的應用程式部署到 Elastic Beanstalk 時，可以在檔案中指定您的 VPC 選項設定，並和您的來源套件一同部署。如需詳細資訊，請參閱 [使用組態檔案 (.ebextensions) 來進行進階的環境自訂]。

更新選項設定時，您至少需要指定下列項目：

- VPCId – 內含 VPC 的 ID。
- Subnets (子網路) – 內含 Auto Scaling 群組子網路的 ID。此範例為私有子網路的 ID。
- ELBSubnets – 內含負載平衡器的子網路 ID。此範例為公有子網路的 ID。
- SecurityGroups – 內含安全群組的 ID。
- DBSubnets – 內含資料庫子網路的 ID。

> **Note**
>
> 使用 DB 子網路時，您必須在 VPC 中建立其他子網路，以涵蓋 AWS 區域中的所有可用區域。

您亦可選擇指定下列資訊：

- ELBScheme – 如要在您的 VPC 內部建立內部負載平衡器，讓您的 Elastic Beanstalk 應用程式無法從 VPC 外部存取，請指定 `internal`。

下列範例是在 VPC 中部署 Elastic Beanstalk 應用程式時，您可以使用的選項設定。如需 VPC 選項設定 (包括指定範例、預設值和有效值) 的詳細資訊，請參閱組態選項中的 aws:ec2:vpc 命名空間表格。

```
option_settings:
  - namespace: aws:autoscaling:launchconfiguration
    option_name: EC2KeyName
    value: ec2keypair

  - namespace: aws:ec2:vpc
    option_name: VPCId
    value: vpc-170647c

  - namespace: aws:ec2:vpc
    option_name: Subnets
    value: subnet-4f195024

  - namespace: aws:ec2:vpc
    option_name: ELBSubnets
    value: subnet-fe064f95

  - namespace: aws:ec2:vpc
    option_name: DBSubnets
    value: subnet-fg148g78

  - namespace: aws:autoscaling:launchconfiguration
    option_name: InstanceType
    value: m1.small

  - namespace: aws:autoscaling:launchconfiguration
    option_name: SecurityGroups
    value: sg-7f1ef110
```

> **Note**
>
> 使用 DB 子網路時，您的 VPC 務必具有子網路，以涵蓋 AWS 區域中的所有可用區域。

搭配 VPC 端點使用 Elastic Beanstalk

VPC 端點可讓您將 VPC 私密地連線至支援的 AWS 服務以及具有 AWS PrivateLink 功能的 VPC 端點服務，而不需要網際網路閘道、NAT 裝置、VPN 連接或 AWS Direct Connect 連線。

VPC 中的執行個體不需要公有 IP 地址，即可與服務中的資源通訊。VPC 與另一個服務之間的流量都會保持在 Amazon 網路的範圍內。如需 VPC 端點的完整資訊，請參閱《Amazon VPC 使用者指南》中的 VPC 端點。

AWS Elastic Beanstalk 支援 AWS PrivateLink，此提供 Elastic Beanstalk 服務的私有連線，並免除公有網際網路的流量暴露。若要讓您的應用程式使用 AWS PrivateLink 傳送請求至 Elastic Beanstalk，您可以設定稱為界面 VPC 端點 (界面端點) 的 VPC 端點類型。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的界面 VPC 端點 (AWS PrivateLink)。

> **Note**
>
> Elastic Beanstalk 支援在有限數量的 AWS 區域中 AWS PrivateLink 和界面 VPC 端點。我們正在努力在不久的未來將支援延伸至更多 AWS 區域。

設定 Elastic Beanstalk 的 VPC 端點

如要在您的 VPC 中建立 Elastic Beanstalk 服務的 VPC 端點，請遵循建立界面端點程序。對於 Service Name (服務名稱)，請選擇 com.amazonaws.***region***.elasticbeanstalk。

如果您已經為您的 VPC 設定公有網際網路存取，您的應用程式仍然可以使用 `elasticbeanstalk.`***region***`.amazonaws.com` 公有端點，透過網際網路存取 Elastic Beanstalk。您可以確保在端點建立期間 Enable DNS name (啟用 DNS 名稱) (預設為 true)，以避免這種情況。這會在 VPC 中新增 DNS 項目，此項目會將公有服務端點映射至界面 VPC 端點。

設定 VPC 端點以增強運作狀態

若您已啟用環境的增強型運作狀態報告，您也可以將增強運作狀態資訊設定為透過 AWS PrivateLink 來傳送。增強型運作狀態資訊會透過 `healthd` 協助程式 (一種位於您環境執行個體上的 Elastic Beanstalk 元件) 傳送至個別的 Elastic Beanstalk 增強型運作狀態服務。如要在您的 VPC 中建立此服務的界面 VPC 端點，請遵循建立執行個體端點程序。對於 Service Name (服務名稱)，請選擇 com.amazonaws.***region***.elasticbeanstalk-health。

> **Important**
>
> healthd 協助程式會將增強的運作狀態資訊傳送至公有端點 elasticbeanstalk-health.*region*.amazonaws.com。如果已使用公有網際網路存取設定 VPC，且 VPC 端點的 Enable DNS name (啟用 DNS 名稱) 已停用，則增強的運作狀態資訊會透過公有網際網路傳送。當您設定增強運作狀態 VPC 端點時，這可能不是您的目的。確定 Enable DNS name (啟用 DNS 名稱) (預設為 true)。

在私有 VPC 中使用 VPC 端點

私有 VPC 或 VPC 中的私有子網路，都沒有公有網際網路存取。建議您在私有 VPC 中執行 Elastic Beanstalk 環境，並設定界面 VPC 端點以增強安全性。在這種情況下，請注意您的環境可能會因為其他原因 (除了與 Elastic Beanstalk 服務聯絡) 而嘗試連線到網際網路。若要進一步了解如何在私有 VPC 中執行環境，請參閱 the section called “在私有 VPC 中執行 Elastic Beanstalk 環境”。

使用端點政策搭配 VPC 端點來控制存取

根據預設，VPC 端點允許完整存取與其相關聯的服務。當您建立或修改端點時，可以將端點政策連接至其中。

端點政策是 AWS Identity and Access Management (IAM) 資源政策，此政策可控制從端點到指定服務的存取。端點政策是端點特有的。其與您環境可能擁有任何使用者或執行個體 IAM 政策是分開的，不會覆寫任何這類政策。如需撰寫和使用 VPC 端點政策的詳細資訊，請參閱《Amazon VPC 使用者指南》中的使用 VPC 端點控制對服務的存取。

下列範例會拒絕所有使用者透過 VPC 端點終止環境的許可，並允許對所有其他動作的完整存取權。

```
{
    "Statement": [
        {
            "Action": "*",
            "Effect": "Allow",
            "Resource": "*",
            "Principal": "*"
        },
        {
            "Action": "elasticbeanstalk:TerminateEnvironment",
            "Effect": "Deny",
            "Resource": "*",
```

```
            "Principal": "*"
        }
    ]
}
```

> **Note**
>
> 目前，只有主要 Elastic Beanstalk 服務支援將端點政策連接至其 VPC 端點。增強運作狀態服務不支援端點政策。

設定您的開發機器搭配 Elastic Beanstalk 使用

此頁面會示範如何設定本機電腦來開發 AWS Elastic Beanstalk 應用程式。內容涵蓋資料夾架構、原始程式碼控制和 CLI 工具。

主題

建立專案資料夾

建立您的專案資料夾。您可以將此資料夾存放於本機磁碟上的任何位置，只要您擁有其讀取和寫入許可即可。您可於使用者資料夾建立資料夾。若您計劃在多個應用程式展開工作，請將您的專案資料夾建立在名為類似 `workspace` 或 `projects` 的其他資料夾內，以維持所有事情有條不紊：

```
workspace/
|-- my-first-app
`-- my-second-app
```

專案資料夾的內容依您應用程式使用的 Web 容器或架構而異。

> **Note**
>
> 於資料夾名稱或任何路徑元素中，避免使用單引號 (') 或雙引號 (") 字元做為資料夾或路徑。當資料夾名稱具備這兩種字元之一時，部分在其中執行的 Elastic Beanstalk 命令會失敗。

設定來源控制

設定來源控制以避免意外刪除自己專案資料夾內的檔案或程式碼，亦可還原造成專案中斷的變更。

若您沒有來源控制系統，請考慮使用 Git，這不但易於使用而且免費，並與 Elastic Beanstalk 命令列界面 (CLI) 完美整合。請造訪 Git 首頁來安裝 Git。

依 Git 網站的說明來安裝 Git 並進行設定，然後於您的專案資料夾內執行 `git init` 來設定本機儲存庫：

```
~/workspace/my-first-app$ git init
Initialized empty Git repository in /home/local/username/workspace/my-first-app/.git/
```

您於專案資料夾新增和更新內容時，請將變更遞交至您的 Git 儲存庫：

```
~/workspace/my-first-app$ git add default.jsp
~/workspace/my-first-app$ git commit -m "add default JSP"
```

每次遞交都會建立專案快照，若出現任何錯誤，稍後即能以此還原。如需 Git 命令和工作流程的詳細資訊，請參閱 Git documentation。

設定遠端儲存庫

若您的硬碟當機，或您想要在不同電腦處理專案，應該怎麼辦？ 欲於線上備份您的原始碼並自任何電腦存取，請設定您可推播遞交的遠端儲存庫。

AWS CodeCommit 可讓您在 AWS 雲端中建立私有儲存庫。CodeCommit 在 AWS 免費方案中是免費的，您帳戶中最多可有五名 AWS Identity and Access Management (IAM) 使用者。 如需定價詳情，請參閱 AWS CodeCommit 定價。

請造訪《AWS CodeCommit 使用者指南》取得設定說明。

GitHub 是另一個熱門的選項，可於線上存放您的專案程式碼，讓您建立公有的線上儲存庫，此外亦支援按月計費的私有儲存庫。請至 github.com 註冊 GitHub。

您的專案建立遠端儲存庫後，請透過 `git remote add` 將其連接至您的本機儲存庫：

```
~/workspace/my-first-app$ git remote add origin ssh://git-codecommit.us-
east-2.amazonaws.com/v1/repos/my-repo
```

安裝 EB CLI

使用 EB CLI 來管理您的 Elastic Beanstalk 環境，並自命令列監控運作狀態。請參閱安裝 EB CLI 相關文章，取得安裝說明。

根據預設，EB CLI 會將您專案資料夾的所有內容包裝成套件，並上傳至 Elastic Beanstalk 做為原始碼套件。當您同時使用 Git 和 EB CLI，即可透過 `.gitignore` 避免將已建立的類別檔案遞交至來源，並透過 `.ebignore` 防止部署來源檔案。

您亦可將 EB CLI 設定為部署組建成品 (WAR 或 ZIP 檔案)，而非部署您專案資料夾的內容。

安裝 AWS CLI

AWS Command Line Interface (AWS CLI) 是 AWS 服務的統一用戶端，可提供所有公有 API 操作的命令。這些命令的層級較 EB CLI 所提供的命令為低，因此透過 AWS CLI 執行操作通常需要更多指令。另一方面，AWS Command Line Interface 讓您能夠作業於帳戶內的任何應用程式或環境，無須於本機設定儲存庫。使用 AWS CLI 建立指令碼，可將操作任務簡化或自動化。

如需有關支援服務及下載 AWS Command Line Interface 的詳細資訊，請參閱 AWS Command Line Interface。

使用 Elastic Beanstalk 命令列界面 (EB CLI)

EB CLI 是一個命令列介面，提供互動式指令，可簡化從本機存放庫建立、更新和監控環境的過程。AWS Elastic Beanstalk 日常開發和測試週期可使用 EB CLI 來替代 Elastic Beanstalk 主控台。

> **Note**
>
> EB CLI 目前版本的基本命令，與版本 3.0 之前的版本不同。若您使用較舊版本，請參閱遷移至 EB CLI 3 和 CodeCommit 取得遷移資訊。

安裝 EB CLI 並設定專案目錄後，即可使用單一命令建立環境：

```
~/my-app$ eb create my-env
```

EB CLI 的原始碼是開放原始碼專案。它位於aws/aws-elastic-beanstalk-cli GitHub 存放庫中。您可以透過報告問題、提出建議，以及提交提取請求來參與。我們重視您的貢獻！針對您只想依現狀使用 EB CLI 的環境，我們建議您使用 the section called “使用設定指令碼安裝 EB CLI” 中詳述的其中一個 EB CLI 安裝指令碼來進行安裝。

Elastic Beanstalk 過去支援可直接存取 API 操作的 CLI，稱為 Elastic Beanstalk API CLI。這已被替換為 AWS CLI，它提供了相同的功能，但適用於所有 AWS 服務的 API。

使用 AWS CLI 您可以直接訪問 Elastic Beanstalk API。這 AWS CLI 是偉大的腳本，但不是那麼容易從命令行使用，因為你需要運行的命令的數量和每個命令的參數的數量。例如，建立環境需要一系列命令：

```
~$ aws elasticbeanstalk check-dns-availability --cname-prefix my-cname
~$ aws elasticbeanstalk create-application-version --application-name my-application --
version-label v1 --source-bundle S3Bucket=DOC-EXAMPLE-BUCKET,S3Key=php-proxy-sample.zip
~$ aws elasticbeanstalk create-environment --cname-prefix my-cname --application-name
 my-app --version-label v1 --environment-name my-env --solution-stack-name "64bit
 Amazon Linux 2015.03 v2.0.0 running Ruby 2.2 (Passenger Standalone)"
```

如需安裝 EB CLI、設定儲存庫和搭配環境的資訊，請參閱下列主題。

主題

- 安裝 EB CLI
- 設定 EB CLI

- 使用 EB CLI 管理 Elastic Beanstalk 環境
- 搭配 AWS CodeBuild 使用 EB CLI
- 搭配 Git 使用 EB CLI
- 搭配 AWS CodeCommit 使用 EB CLI
- 使用 EB CLI 來監控環境運作狀態
- 利用 EB CLI，以群組方式管理多個 Elastic Beanstalk 環境
- 使用 EB CLI 來進行問題的故障診斷
- EB CLI 命令參考
- EB CLI 2.6 (已淘汰)
- Elastic Beanstalk API 命令列界面 (已淘汰)

安裝 EB CLI

AWS Elastic Beanstalk 命令列界面 (EB CLI) 為一種命令列用戶端，可用於建立、設定及管理 Elastic Beanstalk 環境。如需 EB CLI 的詳細資訊，請參閱 EB CLI。

主題

- 使用設定指令碼安裝 EB CLI
- 手動安裝 EB CLI

使用設定指令碼安裝 EB CLI

最簡單且建議使用的 EB CLI 安裝方法，是使用 GitHub 上的 EB CLI 安裝指令碼。使用指令碼在 Linux、macOS 或 Windows 上安裝 EB CLI。此指令碼會安裝 EB CLI 及其相依性，包括 Python 和 `pip`。指令碼還會為 EB CLI 建立虛擬環境。如需安裝說明，請參閱 GitHub 上的 `aws/aws-elastic-beanstalk-cli-setup` 儲存庫。

手動安裝 EB CLI

若要安裝 EB CLI，建議您使用 EB CLI 設定指令碼。如果設定指令碼與您的開發環境不相容，請手動安裝 EB CLI。

EB CLI 在 Linux、macOS 和 Windows 的主要分發方式是 `pip`。這是 Python 適用的套件管理工具，其提供一個簡單的方法來安裝、升級和移除 Python 套件及其相依性。以 macOS 而言，您亦可使用 `Homebrew` 取得最新版本的 EB CLI。

相容性備註

EB CLI 以 Python 開發，需要 Python 版本 3.11 或更新版本。

建議您使用 EB CLI 設定指令碼 來安裝 EB CLI 及其相依性。如果您手動安裝 EB CLI，可能難以在您的開發環境中管理相依性衝突。

EB CLI 和 AWS Command Line Interface (AWS CLI) 在 botocore Python 套件上共享相依性。由於 `botocore` 不同版本的重大變更，這兩個 CLI 工具取決於不同版本的 `botocore`。

最新版本的兩個 CLI 可相容。如果您需要使用舊版，請參閱下表以取得要使用的相容版本。

EB CLI 版本	相容 AWS CLI 版本
3.14.5 或舊版本	1.16.9 或舊版本
3.14.6 或更新版本	1.16.11 或更新版本

安裝 EB CLI

如果您已經有 `pip` 和支援的 Python 版本，則可使用下列程序來安裝 EB CLI。

如果您沒有 Python 和 `pip`，請使用您使用的作業系統的程序。

- 在 Linux 上安裝 Python、pip 和 EB CLI
- 在 macOS 上安裝 EB CLI
- 在 Windows 上安裝 Python、pip 和 EB CLI

欲安裝 EB CLI

1. 執行下列命令。

   ```
   $ pip install awsebcli --upgrade --user
   ```

   此 `--upgrade` 選項通知 `pip` 升級已安裝的任何要求。此 `--user` 選項通知 `pip` 將程式安裝到使用者目錄的子目錄中，以避免修改作業系統使用的程式庫。

> **Note**
>
> 如果透過 pip 嘗試安裝 EB CLI 時遇到問題，可以在虛擬環境中安裝 EB CLI，藉此隔離該工具及其依存項目，或者使用平常不使用的 Python 版本。

2. 新增可執行檔路徑到您的 PATH 變數：

   - 在 Linux 和 macOS：

     Linux – `~/.local/bin`

     macOS – `~/Library/Python/3.7/bin`

     若要修改您的 PATH 變數 (Linux、Unix 或 macOS)：

     a. 在您的使用者資料夾中尋找 Shell 的描述檔指令碼。如果您不確定您擁有哪個 shell，請執行 `echo $SHELL`。

     ```
     $ ls -a ~
     .  ..  .bash_logout  .bash_profile  .bashrc  Desktop  Documents  Downloads
     ```

     - Bash – `.bash_profile`、`.profile` 或 `.bash_login`。
     - Zsh – `.zshrc`
     - Tcsh -– `.tcshrc`、`.cshrc` 或 `.login`。

     b. 將匯出命令新增至您的描述檔指令碼。下列範例會將由 *LOCAL_PATH* 代表的路徑新增至目前的 PATH 變數。

     ```
     export PATH=LOCAL_PATH:$PATH
     ```

     c. 將第一個步驟所述的描述檔指令碼載入您目前的工作階段。下列範例會載入由 *PROFILE_SCRIPT* 代表的描述檔指令碼。

     ```
     $ source ~/PROFILE_SCRIPT
     ```

   - 在 Windows 上：

     Python 3.7 – `%USERPROFILE%\AppData\Roaming\Python\Python37\Scripts`

     較早版本的 Python – `%USERPROFILE%\AppData\Roaming\Python\Scripts`

修改 PATH 變數 (Windows)：

a. 按下 Windows 鍵，然後輸入 **environment variables**。
b. 選擇 Edit environment variables for your account (編輯您帳戶的環境變數)。
c. 選擇 PATH，然後選擇 Edit (編輯)。
d. 將路徑新增到變數值欄位中，以分號分隔。例如：*C:\item1\path*;***C:\item2\path***
e. 選擇 OK (確定) 兩次以套用新的設定。
f. 關閉任何正在執行的命令提示字元視窗，然後重新開啟命令提示字元視窗。

3. 執行 eb --version 來驗證 EB CLI 是否正確安裝。

```
$ eb --version
EB CLI 3.14.8 (Python 3.7)
```

EB CLI 會定期更新，新增支援最新 Elastic Beanstalk 功能的功能。要更新到最新版本的 EB CLI，再次執行安裝命令。

```
$ pip install awsebcli --upgrade --user
```

如果您需要解除安裝 EB CLI，請使用 `pip uninstall`。

```
$ pip uninstall awsebcli
```

在 Linux 上安裝 Python、pip 和 EB CLI

EB CLI 需要 Python 2.7、3.4 或更新版本。如果您的發行版本沒有隨附 Python，或者隨附了舊版本，請在安裝 `pip` 和 EB CLI 之前先安裝 Python。

在 Linux 上安裝 Python 3.7

1. 判斷是否已安裝 Python。

```
$ python --version
```

> **Note**
>
> 如果您的 Linux 分發隨附 Python，您可能需要安裝 Python 開發人員套件，以便獲取編譯擴展所需的標頭和程式庫並安裝 EB CLI。使用您的套件管理員來安裝開發人員套件 (通常名為 `python-dev` 或 `python-devel`)。

2. 如果未安裝 Python 2.7 或更新版本，請使用您的分發套件管理工具安裝 Python 3.7。命令和套件名稱有所不同：

   - 在 Debian 的衍生產品上，例如 Ubuntu，使用 APT。

     ```
     $ sudo apt-get install python3.7
     ```

   - 在 Red Hat 和衍生產品，請使用 `yum`。

     ```
     $ sudo yum install python37
     ```

   - 在 SUSE 和衍生產品，請使用 `zypper`。

     ```
     $ sudo zypper install python3-3.7
     ```

3. 若要驗證 Python 是否正確安裝，請開啟終端機或 Shell，並執行以下命令。

   ```
   $ python3 --version
   Python 3.7.3
   ```

使用 Python Packaging Authority 提供的指令碼來安裝 `pip`，然後安裝 EB CLI。

安裝 **`pip`** 和 EB CLI

1. 從 pypa.io 下載安裝指令碼。

   ```
   $ curl -O https://bootstrap.pypa.io/get-pip.py
   ```

   指令碼會下載並安裝最新版本的 `pip` 和另一個名為 `setuptools` 的必要套件。

2. 使用 Python 執行指令碼。

   ```
   $ python3 get-pip.py --user
   ```

```
Collecting pip
  Downloading pip-8.1.2-py2.py3-none-any.whl (1.2MB)
Collecting setuptools
  Downloading setuptools-26.1.1-py2.py3-none-any.whl (464kB)
Collecting wheel
  Downloading wheel-0.29.0-py2.py3-none-any.whl (66kB)
Installing collected packages: pip, setuptools, wheel
Successfully installed pip setuptools wheel
```

使用 python3 命令 (而非 python) 來直接呼叫 Python 版本 3，可確保即使系統存在 Python 的較早系統版本，pip 仍會安裝於適當位置。

3. 新增可執行檔路徑 (~/.local/bin) 到您的 PATH 變數。

   若要修改您的 PATH 變數 (Linux、Unix 或 macOS)：

   a. 在您的使用者資料夾中尋找 Shell 的描述檔指令碼。如果您不確定您擁有哪個 shell，請執行 echo $SHELL。

      ```
      $ ls -a ~
      .  ..  .bash_logout  .bash_profile  .bashrc  Desktop  Documents  Downloads
      ```

      - Bash – .bash_profile、.profile 或 .bash_login。
      - Zsh – .zshrc
      - Tcsh -– .tcshrc、.cshrc 或 .login。

   b. 將匯出命令新增至您的描述檔指令碼。下列範例會將由 *LOCAL_PATH* 代表的路徑新增至目前的 PATH 變數。

      ```
      export PATH=LOCAL_PATH:$PATH
      ```

   c. 將第一個步驟所述的描述檔指令碼載入您目前的工作階段。下列範例會載入由 *PROFILE_SCRIPT* 代表的描述檔指令碼。

      ```
      $ source ~/PROFILE_SCRIPT
      ```

4. 確認已正確安裝 pip。

   ```
   $ pip --version
   pip 8.1.2 from ~/.local/lib/python3.7/site-packages (python 3.7)
   ```

5. 使用 pip 安裝 EB CLI。

```
$ pip install awsebcli --upgrade --user
```

6. 確認已正確安裝 EB CLI。

```
$ eb --version
EB CLI 3.14.8 (Python 3.7)
```

更新到最新版本的 & CLI，再次執行安裝命令。

```
$ pip install awsebcli --upgrade --user
```

在 macOS 上安裝 EB CLI

若您使用 Homebrew 套件管理工具，則可使用 brew 命令來安裝 EB CLI。您亦可安裝 Python 和 pip，然後使用 pip 來安裝 EB CLI。

使用 Homebrew 安裝 EB CLI

當 Homebrew 出現新版本的 EB CLI，通常數日之後即可在 pip 使用。

使用 **Homebrew** 安裝 EB CLI

1. 確認您具備最新版本的 Homebrew。

```
$ brew update
```

2. 執行 brew install awsebcli。

```
$ brew install awsebcli
```

3. 確認已正確安裝 EB CLI。

```
$ eb --version
EB CLI 3.14.8 (Python 3.7)
```

在 macOS 上安裝 Python、pip 和 EB CLI

您可以先安裝最新版本的 Python 和 pip，然後使用它們來安裝 EB CLI。

欲在 macOS 上安裝 EB CLI

1. 從 Python.org 的下載頁面下載 Python 並加以安裝。我們使用 3.7 版來示範。

   > **Note**
   >
   > EB CLI 需要 Python 2 的 2.7 版，或 Python 3 的 3.4 到 3.7 版。

2. 使用 Python Packaging Authority 提供的指令碼來安裝 `pip`。

   ```
   $ curl -O https://bootstrap.pypa.io/get-pip.py
   $ python3 get-pip.py --user
   ```

3. 使用 `pip` 安裝 EB CLI。

   ```
   $ pip3 install awsebcli --upgrade --user
   ```

4. 新增可執行檔路徑 (`~/Library/Python/3.7/bin`) 到您的 `PATH` 變數。

   若要修改您的 `PATH` 變數 (Linux、Unix 或 macOS)：

   a. 在您的使用者資料夾中尋找 Shell 的描述檔指令碼。如果您不確定您擁有哪個 shell，請執行 `echo $SHELL`。

      ```
      $ ls -a ~
      .  ..  .bash_logout  .bash_profile  .bashrc  Desktop  Documents  Downloads
      ```

      - Bash – `.bash_profile`、`.profile` 或 `.bash_login`。
      - Zsh – `.zshrc`
      - Tcsh – `.tcshrc`、`.cshrc` 或 `.login`。

   b. 將匯出命令新增至您的描述檔指令碼。下列範例會將由 *LOCAL_PATH* 代表的路徑新增至目前的 `PATH` 變數。

      ```
      export PATH=LOCAL_PATH:$PATH
      ```

   c. 將第一個步驟所述的描述檔指令碼載入您目前的工作階段。下列範例會載入由 *PROFILE_SCRIPT* 代表的描述檔指令碼。

      ```
      $ source ~/PROFILE_SCRIPT
      ```

5. 確認已正確安裝 EB CLI。

```
$ eb --version
EB CLI 3.14.8 (Python 3.7)
```

更新到最新版本的 & CLI，再次執行安裝命令。

```
$ pip3 install awsebcli --upgrade --user
```

在 Windows 上安裝 Python、pip 和 EB CLI

Python 軟體基金會為 Windows 提供包含 `pip` 的安裝程式。

安裝 Python 和 **`pip`** (Windows)

1. 從 Python.org 的下載頁面下載最新的 Python Windows x86-64 可執行檔安裝程式。
2. 執行您在先前步驟中下載的 Python 安裝程式可執行檔。

   從 Python 安裝程式視窗中選取下列選項，以針對接下來的 EB CLI 安裝步驟進行設定。

   a. 選擇將 Python 可執行檔新增至您的路徑中。
   b. 選擇 Install Now (立即安裝)。

> **Note**
>
> 如需有關安裝選項的詳細資訊，請參閱 Python 網站上的在 Windows 上使用 Python 頁面。
> 文件網站在頁面頂端提供了一個下拉式清單，您可以在其中選取文件的 Python 版本。

安裝程式將 Python 安裝到使用者資料夾中，並將其可執行目錄新增到使用者路徑中。

使用 **`pip`** 安裝 AWS CLI (Windows)

1. 從開始功能表開啟命令提示字元視窗。
2. 使用下列命令確認 Python 和 `pip` 是否都安裝正確。

```
C:\Users\myname> python --version
```

```
Python 3.11.4
C:\Users\myname> pip --version
pip 23.1.2 from C:\Users\myname\AppData\Local\Programs\Python\Python311\Lib\site-
packages\pip (python 3.11)
```

3. 使用 pip 安裝 EB CLI。

```
C:\Users\myname> pip install awsebcli --upgrade --user
```

4. 將下列可執行檔路徑新增至 Windows 使用者帳戶中的 Path 環境變數。安裝位置可能不同，取決於您為一位使用者或是所有使用者安裝 Python。

   %USERPROFILE%\AppData\Roaming\Python\Python311\Scripts

5. 重新啟動新的命令 shell，使新的 Path 變數生效。
6. 確認已正確安裝 EB CLI。

```
C:\Users\myname>  eb --version
EB CLI 3.14.8 (Python 3.11)
```

更新到最新版本的 & CLI，再次執行安裝命令。

```
C:\Users\myname> pip install awsebcli --upgrade --user
```

在虛擬環境中安裝 EB CLI

您可以透過在虛擬環境中安裝 EB CLI，避免所需版本與其他 pip 套件發生衝突。

欲在虛擬環境中安裝 EB CLI

1. 使用 pip 安裝 virtualenv。

```
$ pip install --user virtualenv
```

2. 建立虛擬環境。

```
$ virtualenv ~/eb-ve
```

   若要使用預設以外的 Python 可執行檔，請使用 -p 選項。

```
$ virtualenv -p /usr/bin/python3.7 ~/eb-ve
```

3. 啟用虛擬環境。

   Linux、Unix 或 macOS

   ```
   $ source ~/eb-ve/bin/activate
   ```

   Windows

   ```
   $ %USERPROFILE%\eb-ve\Scripts\activate
   ```

4. 安裝 EB CLI。

   ```
   (eb-ve)~$ pip install awsebcli --upgrade
   ```

5. 確認已正確安裝 EB CLI。

   ```
   $ eb --version
   EB CLI 3.14.8 (Python 3.7)
   ```

您可以使用 `deactivate` 命令來離開虛擬環境。無論您何時開始新的工作階段，請重新執行啟用命令。

更新到最新版本的 & CLI，再次執行安裝命令。

```
(eb-ve)~$ pip install awsebcli --upgrade
```

設定 EB CLI

安裝 EB CLI 之後，您就可以執行 eb init 以設定您的專案目錄和 EB CLI。

下列範例說明首次在名為 eb init 的專案資料夾執行 eb 的設定步驟。

欲初始化 EB CLI 專案

1. 首先，EB CLI 會提示您選擇一個區域。輸入您要使用之區域相對應的編號，然後按 Enter (確認)。

```
~/eb $ eb init
Select a default region
1) us-east-1 : US East (N. Virginia)
2) us-west-1 : US West (N. California)
3) us-west-2 : US West (Oregon)
4) eu-west-1 : Europe (Ireland)
5) eu-central-1 : Europe (Frankfurt)
6) ap-south-1 : Asia Pacific (Mumbai)
7) ap-southeast-1 : Asia Pacific (Singapore)
...
(default is 3): 3
```

2. 接著，提供您的存取金鑰和私密金鑰，讓 EB CLI 能夠為您管理資源。存取金鑰於 AWS Identity and Access Management 主控台內建立。如果您沒有金鑰，請參閱《Amazon Web Services 一般參考》中的如何取得安全憑證？。

```
You have not yet set up your credentials or your credentials are incorrect.
You must provide your credentials.
(aws-access-id): AKIAJOUAASEXAMPLE
(aws-secret-key): 5ZRIrtTM4ciIAvd4EXAMPLEDtm+PiPSzpoK
```

3. Elastic Beanstalk 中的應用程式為資源，其中包含一組與單一 Web 應用程式建立關聯的應用程式版本 (來源)、環境及已儲存組態。您每次使用 EB CLI 將來源碼部署至 Elastic Beanstalk 時，都會建立新的應用程式版本，並新增至清單。

```
Select an application to use
1) [ Create new Application ]
(default is 1): 1
```

4. 預設應用程式名稱為您執行 eb init 的資料夾名稱。輸入可描述專案的名稱。

```
Enter Application Name
(default is "eb"): eb
Application eb has been created.
```

5. 選擇符合您 Web 應用程式開發所用之語言或架構的平台。若您尚未開發應用程式，請選擇您喜歡的平台。您很快就會了解如何啟動範例應用程式，而且稍後可隨時變更此設定。

```
Select a platform.
1) Node.js
2) PHP
```

```
3) Python
4) Ruby
5) Tomcat
6) IIS
7) Docker
8) Multi-container Docker
9) GlassFish
10) Go
11) Java
(default is 1): 1
```

6. 選擇 yes (是) 以指派 SSH 金鑰對到您 Elastic Beanstalk 環境中的執行個體。這可讓您直接連接以進行故障排除。

```
Do you want to set up SSH for your instances?
(y/n): y
```

7. 選擇現有金鑰對或建立新的金鑰對。欲使用 eb init 來建立新的金鑰對，ssh-keygen 必須安裝於您的本機機器且可從命令列取用。EB CLI 會為您向 Amazon EC2 註冊新的金鑰對，並將私有金鑰存放於本機使用者目錄中名為 `.ssh` 的資料夾。

```
Select a keypair.
1) [ Create new KeyPair ]
(default is 1): 1
```

您的 EB CLI 安裝設定現已完成且就緒。如需建立並使用 Elastic Beanstalk 環境的說明，請參閱 使用 EB CLI 管理 Elastic Beanstalk 環境。

進階組態

- 使用 .ebignore 忽略檔案
- 使用命名設定檔
- 部署成品而非專案資料夾
- 組態設定和優先順序
- 執行個體中繼資料

使用 .ebignore 忽略檔案

您可將檔案 .ebignore 新增至專案目錄，藉此指示 EB CLI 忽略該目錄中的特定檔案。此檔案的運作方式如同 .gitignore 檔案。將專案目錄部署至 Elastic Beanstalk 並建立新的應用程式版本時，EB CLI 不會將 .ebignore 指定的檔案納入其所建立的來源套件中。

若 .ebignore 不存在但 .gitignore 存在，則 EB CLI 會忽略 .gitignore 指定的檔案。若 .ebignore 存在，則 EB CLI 不會讀取 .gitignore。

若 .ebignore 存在，EB CLI 不會使用 git 命令來建立您的原始碼套件。這表示 EB CLI 會忽略 .ebignore 指定的檔案，並納入所有其他檔案，尤其是未遞交的來源檔案。

> **Note**
>
> 在 Windows 中，新增 .ebignore 會使 EB CLI 在建立原始碼套件時，存取符號連結並納入連結檔案。此已知問題將於未來的更新中修正。

使用命名設定檔

如您將登入資料做為具名描述檔存放在 credentials 或 config 檔案，您可使用 --profile 選項來明確指定描述檔。例如，下列命令會使用 user2 設定檔來建立新的應用程式。

```
$ eb init --profile user2
```

您亦可設定 AWS_EB_PROFILE 環境變數來變更預設的設定檔。設定此變數後，EB CLI 會自指定設定檔讀取登入資料，而不從 default 或 eb-cli 讀取。

Linux、macOS 或 Unix

```
$ export AWS_EB_PROFILE=user2
```

Windows

```
> set AWS_EB_PROFILE=user2
```

部署成品而非專案資料夾

您可將下列行新增至專案資料夾內的 .elasticbeanstalk/config.yml，藉此指示 EB CLI 部署您所產生的 ZIP 檔或 WAR 檔案，做為不同的建置程序的一部分。

```
deploy:
  artifact: path/to/buildartifact.zip
```

若您於 Git 儲存庫設定 EB CLI，而且不要將成品遞交至來源，請使用 `--staged` 選項來部署最新的建置。

```
~/eb$ eb deploy --staged
```

組態設定和優先順序

EB CLI 使用提供者鏈結在許多不同的地方尋找 AWS 登入資料，包括系統或使用者環境變數以及本機 AWS 組態檔案。

EB CLI 根據以下順序尋找登入資料和組態設定：

1. 命令列選項 – 使用 `--profile` 來指定具名描述檔以覆寫預設設定。
2. 環境變數 – `AWS_ACCESS_KEY_ID` 和 `AWS_SECRET_ACCESS_KEY`。
3. AWS 登入資料檔案 – 位於 Linux 和 OS X 系統上的 `~/.aws/credentials`，或是 Windows 系統上的 `C:\Users\`*`USERNAME`*`\.aws\credentials`。除了預設的設定檔外，此檔案還可以包含多個命名設定檔。
4. AWS CLI 組態檔案 – 位於 Linux 和 OS X 系統上的 `~/.aws/config`，或是 Windows 系統上的 `C:\Users\`*`USERNAME`*`\.aws\config`。此檔案可以包含每個預設設定檔、命名設定檔和 AWS CLI 特定組態參數。
5. 傳統 EB CLI 組態檔案 – 位於 Linux 和 OS X 系統上的 `~/.elasticbeanstalk/config`，或是 Windows 系統上的 `C:\Users\`*`USERNAME`*`\.elasticbeanstalk\config`。
6. 執行個體描述檔登入資料 – 這些登入資料可用於具備指派執行個體角色的 Amazon EC2 執行個體，並透過 Amazon EC2 中繼資料服務交付。執行個體描述檔必須具備使用 Elastic Beanstalk 的許可。

若登入資料檔案包含名為「eb-cli」的命名設定檔，則 EB CLI 會偏好該設定檔，而非預設設定檔。若找不到描述檔，或所找到的描述檔未具備使用 Elastic Beanstalk 的許可，則 EB CLI 會提示您輸入金鑰。

執行個體中繼資料

若要從 Amazon EC2 執行個體使用 EB CLI，請建立可以存取所需資源的角色，並在執行個體啟動時，指派該角色到執行個體。啟動執行個體並使用 `pip` 安裝 EB CLI。

```
~$ sudo pip install awsebcli
```

pip 預先安裝在 Amazon Linux 上。

EB CLI 會讀取來自執行個體中繼資料的登入資料。如需詳細資訊，請參閱《IAM 使用者指南》中的授予在 Amazon EC2 執行個體上執行的應用程式存取 AWS 資源的許可。

使用 EB CLI 管理 Elastic Beanstalk 環境

安裝 EB CLI 並設定您的專案目錄後，即可使用 EB CLI 建立 Elastic Beanstalk 環境、部署原始碼和組態更新，以及叫出日誌和事件。

> **Note**
>
> 要以 EB CLI 建立環境，就必須有服務角色。您可以在 Elastic Beanstalk 主控台建立環境，以建立服務角色。如果您沒有服務角色，EB CLI 會嘗試在您執行 `eb create` 時為您建立。

EB CLI 會從所有成功的命令傳回零 (0) 結束代碼，而在遇到任何錯誤時，則傳回非零結束代碼。

下列範例使用名為 eb 的空白專案資料夾，該資料夾經 EB CLI 初始化以搭配範例 Docker 應用程式使用。

基本命令

- Eb create
- Eb status
- Eb health
- Eb events
- Eb logs
- Eb open
- Eb deploy
- Eb config
- Eb terminate

Eb create

如欲建立您的第一個環境，請執行 eb create 並依提示操作。若您的專案目錄具備原始碼，EB CLI 會將其封裝並部署至您的環境。否則，將使用範例應用程式。

```
~/eb$ eb create
Enter Environment Name
(default is eb-dev): eb-dev
Enter DNS CNAME prefix
(default is eb-dev): eb-dev
WARNING: The current directory does not contain any source code. Elastic Beanstalk is
 launching the sample application instead.
Environment details for: elasticBeanstalkExa-env
  Application name: elastic-beanstalk-example
  Region: us-west-2
  Deployed Version: Sample Application
  Environment ID: e-j3pmc8tscn
  Platform: 64bit Amazon Linux 2015.03 v1.4.3 running Docker 1.6.2
  Tier: WebServer-Standard
  CNAME: eb-dev.elasticbeanstalk.com
  Updated: 2015-06-27 01:02:24.813000+00:00
Printing Status:
INFO: createEnvironment is starting.
 -- Events -- (safe to Ctrl+C) Use "eb abort" to cancel the command.
```

您的環境可能需要幾分鐘才能就緒。環境建立時，請按 Ctrl+C 返回命令列。

Eb status

執行 eb status，檢視環境的目前狀態。若狀態為 `ready`，則可於 elasticbeanstalk.com 取得範例應用程式，且環境已準備好更新。

```
~/eb$ eb status
Environment details for: elasticBeanstalkExa-env
  Application name: elastic-beanstalk-example
  Region: us-west-2
  Deployed Version: Sample Application
  Environment ID: e-gbzqc3jcra
  Platform: 64bit Amazon Linux 2015.03 v1.4.3 running Docker 1.6.2
  Tier: WebServer-Standard
  CNAME: elasticbeanstalkexa-env.elasticbeanstalk.com
  Updated: 2015-06-30 01:47:45.589000+00:00
```

```
  Status: Ready
  Health: Green
```

Eb health

使用 eb health 命令，檢視您環境中執行個體的運作狀態資訊和整體環境的狀態。使用 --refresh 選項以互動方式檢視每 10 秒更新的運作狀態。

```
~/eb$ eb health
 api                                 Ok                2016-09-15 18:39:04
WebServer                                                          Java 8
  total      ok    warning  degraded  severe    info   pending  unknown
    3        3        0        0        0        0        0        0

  instance-id           status     cause                              health
    Overall             Ok
  i-0ef05ec54918bf567   Ok
  i-001880c1187493460   Ok
  i-04703409d90d7c353   Ok

  instance-id           r/sec    %2xx   %3xx   %4xx   %5xx      p99      p90      p75
   p50     p10
    Overall             8.6     100.0    0.0    0.0    0.0    0.083*   0.065    0.053
 0.040   0.019
  i-0ef05ec54918bf567   2.9        29      0      0      0    0.069*   0.066    0.057
 0.050   0.023
  i-001880c1187493460   2.9        29      0      0      0    0.087*   0.069    0.056
 0.050   0.034
  i-04703409d90d7c353   2.8        28      0      0      0    0.051*   0.027    0.024
 0.021   0.015

  instance-id           type       az   running     load 1  load 5      user%  nice%
 system%  idle%   iowait%
  i-0ef05ec54918bf567   t2.micro   1c   23 mins       0.19    0.05        3.0    0.0
   0.3   96.7       0.0
  i-001880c1187493460   t2.micro   1a   23 mins        0.0     0.0        3.2    0.0
   0.3   96.5       0.0
  i-04703409d90d7c353   t2.micro   1b   1 day          0.0     0.0        3.6    0.0
   0.2   96.2       0.0

  instance-id           status     id   version                  ago
            deployments
  i-0ef05ec54918bf567   Deployed   28   app-bc1b-160915_181041   20 mins
```

```
  i-001880c1187493460   Deployed   28   app-bc1b-160915_181041   20 mins
  i-04703409d90d7c353   Deployed   28   app-bc1b-160915_181041   27 mins
```

Eb events

使用 eb events 來檢視 Elastic Beanstalk 輸出的事件清單。

```
~/eb$ eb events
2015-06-29 23:21:09    INFO    createEnvironment is starting.
2015-06-29 23:21:10    INFO    Using elasticbeanstalk-us-east-2-EXAMPLE as Amazon S3
 storage bucket for environment data.
2015-06-29 23:21:23    INFO    Created load balancer named: awseb-e-g-AWSEBLoa-EXAMPLE
2015-06-29 23:21:42    INFO    Created security group named: awseb-e-gbzqc3jcra-stack-
AWSEBSecurityGroup-EXAMPLE
...
```

Eb logs

使用 eb logs 自您環境中的執行個體叫出日誌。根據預設，eb logs 會從第一個啟動的執行個體叫出日誌，並以標準輸出顯示。您可透過 --instance 選項指定執行個體 ID，取得特定執行個體的日誌。

--all 選項會叫出所有執行個體的日誌，並將其儲存至 `.elasticbeanstalk/logs` 底下的子目錄。

```
~/eb$ eb logs --all
Retrieving logs...
Logs were saved to /home/local/ANT/mwunderl/ebcli/environments/test/.elasticbeanstalk/
logs/150630_201410
Updated symlink at /home/local/ANT/mwunderl/ebcli/environments/test/.elasticbeanstalk/
logs/latest
```

Eb open

如欲在瀏覽器開啟您環境的網站，請使用 eb open：

```
~/eb$ eb open
```

在視窗化環境中，預設將以新視窗開啟瀏覽器。在終端機環境中，將使用諸如 w3m 的命令列瀏覽器 (如有)。

Eb deploy

環境建立並就緒後，即可使用 eb deploy 進行更新。

此命令更適用於部分原始碼的封裝與部署作業，因此在此範例中，我們在專案目錄使用下列內容來建立 `Dockerfile`：

~/eb/Dockerfile

```
FROM ubuntu:12.04

RUN apt-get update
RUN apt-get install -y nginx zip curl

RUN echo "daemon off;" >> /etc/nginx/nginx.conf
RUN curl -o /usr/share/nginx/www/master.zip -L https://codeload.github.com/
gabrielecirulli/2048/zip/master
RUN cd /usr/share/nginx/www/ && unzip master.zip && mv 2048-master/* . && rm -rf 2048-
master master.zip

EXPOSE 80

CMD ["/usr/sbin/nginx", "-c", "/etc/nginx/nginx.conf"]
```

此 `Dockerfile` 會部署 Ubuntu 12.04 的映像並安裝遊戲 2048。執行 eb deploy 將應用程式上傳至您的環境：

```
~/eb$ eb deploy
Creating application version archive "app-150630_014338".
Uploading elastic-beanstalk-example/app-150630_014338.zip to S3. This may take a while.
Upload Complete.
INFO: Environment update is starting.
 -- Events -- (safe to Ctrl+C) Use "eb abort" to cancel the command.
```

執行 eb deploy 時，EB CLI 會封裝您專案目錄的內容，並將其部署至您的環境。

> **Note**
>
> 若您已初始化專案資料夾內的 Git 儲存庫，EB CLI 永遠會部署最新的遞交內容，即使您有待定變更。遞交您的變更，然後執行 eb deploy 以將其部署至您的環境。

Eb config

透過 eb config 命令查看您執行環境可用的組態選項：

```
~/eb$ eb config
ApplicationName: elastic-beanstalk-example
DateUpdated: 2015-06-30 02:12:03+00:00
EnvironmentName: elasticBeanstalkExa-env
SolutionStackName: 64bit Amazon Linux 2015.03 v1.4.3 running Docker 1.6.2
settings:
  AWSEBAutoScalingScaleDownPolicy.aws:autoscaling:trigger:
    LowerBreachScaleIncrement: '-1'
  AWSEBAutoScalingScaleUpPolicy.aws:autoscaling:trigger:
    UpperBreachScaleIncrement: '1'
  AWSEBCloudwatchAlarmHigh.aws:autoscaling:trigger:
    UpperThreshold: '6000000'
...
```

此命令會在文字編輯器顯示可用的組態選項清單。所示的許多選項都具備 null 值，這些並非預設值，且您能夠加以修改來更新環境中的資源。如需這些選項的詳細資訊，請參閱組態選項。

Eb terminate

若您目前已完成使用環境，請使用 eb terminate 終止。

```
~/eb$ eb terminate
The environment "eb-dev" and all associated instances will be terminated.
To confirm, type the environment name: eb-dev
INFO: terminateEnvironment is starting.
INFO: Deleted CloudWatch alarm named: awseb-e-jc8t3pmscn-stack-
AWSEBCloudwatchAlarmHigh-1XLMU7DNCBV6Y
INFO: Deleted CloudWatch alarm named: awseb-e-jc8t3pmscn-stack-
AWSEBCloudwatchAlarmLow-8IVI04W2SCXS
INFO: Deleted Auto Scaling group policy named: arn:aws:autoscaling:us-
east-2:123456789012:scalingPolicy:1753d43e-ae87-4df6-
a405-11d31f4c8f97:autoScalingGroupName/awseb-e-jc8t3pmscn-stack-
AWSEBAutoScalingGroup-90TTS2ZL4MXV:policyName/awseb-e-jc8t3pmscn-stack-
AWSEBAutoScalingScaleUpPolicy-A070H1BMUQAJ
INFO: Deleted Auto Scaling group policy named: arn:aws:autoscaling:us-
east-2:123456789012:scalingPolicy:1fd24ea4-3d6f-4373-
affc-4912012092ba:autoScalingGroupName/awseb-e-jc8t3pmscn-stack-
AWSEBAutoScalingGroup-90TTS2ZL4MXV:policyName/awseb-e-jc8t3pmscn-stack-
AWSEBAutoScalingScaleDownPolicy-LSWFUMZ46H1V
```

```
INFO: Waiting for EC2 instances to terminate. This may take a few minutes.
 -- Events -- (safe to Ctrl+C)
```

如需可用的 EB CLI 命令完整清單，請參閱EB CLI 命令參考。

> **Important**
>
> 如果您終止環境，您也必須刪除您建立的任何 CNAME 映射，因為其他客戶可能重複使用可用的主機名稱。請務必刪除指向終止環境的 DNS 記錄，以防止懸置 DNS 項目。懸置的 DNS 項目可能會降低您網域的網際網路流量的安全性，使其暴露在易於攻擊的弱點中。另外，它還可能存在其他風險。
>
> 如需詳細資訊，請參閱 Amazon Route 53 開發人員指南中的在 Route 53 上防止懸置委派記錄。您也可以在AWS安全部落格中針對適用於 Amazon CloudFront 請求的強化網域保護來進一步了解懸置 DNS 項目的資訊。

搭配 AWS CodeBuild 使用 EB CLI

AWS CodeBuild 可編譯來源碼、執行單位測試，並產生可立即部署的成品。您可以跟 EB CLI 一起使用 CodeBuild，以自動化從原始程式碼建置應用程式的作業。環境建立與每次部署之後便從建置步驟開始，然後部署產生的應用程式。

> **Note**
>
> 某些區域不提供 CodeBuild。Elastic Beanstalk 和 CodeBuild 之間的整合在這些區域中不會發生作用。
>
> 如需各區域提供的 AWS 服務的資訊，請參閱區域表。

建立應用程式

建立使用 CodeBuild 的 Elastic Beanstalk 應用程式

1. 將 CodeBuild 建置規格檔案 `buildspec.yml` 放入您的應用程式資料夾。
2. 將具有 Elastic Beanstalk 特定選項的 `eb_codebuild_settings` 項目加入至檔案。
3. 在資料夾中執行 eb init。

> **Note**
>
> 當您將 EB CLI 與 CodeBuild 搭配使用時，請勿在應用程式名稱中使用句號 (.) 或空格 ( ) 字元。

Elastic Beanstalk 會延伸 CodeBuild 建置規格檔案格式，以包含下列其他設定：

```
eb_codebuild_settings:
  CodeBuildServiceRole: role-name
  ComputeType: size
  Image: image
  Timeout: minutes
```

`CodeBuildServiceRole`

AWS Identity and Access Management (IAM) 服務角色的 ARN 或名稱；CodeBuild 可以藉此代表您與相依 AWS 服務互動。此值為必填。如果您省略此值，任何後續 eb create 或 eb deploy 命令都會失敗。

若要進一步了解如何為 CodeBuild 建立服務角色，請參閱《AWS CodeBuild 使用者指南》中的建立 CodeBuild 服務角色。

> **Note**
>
> 您也需要在 CodeBuild 中自行執行動作的許可。Elastic Beanstalk AdministratorAccess-AWSElasticBeanstalk 受管使用者政策包含所有必要的 CodeBuild 動作許可。如果您未使用受管政策，請務必在使用者政策中允許下列許可。
>
> ```
> "codebuild:CreateProject",
> "codebuild:DeleteProject",
> "codebuild:BatchGetBuilds",
> "codebuild:StartBuild"
> ```
>
> 如需詳細資訊，請參閱管理 Elastic Beanstalk 使用者政策。

`ComputeType`

CodeBuild 建置環境中 Docker 容器使用的資源量。有效值為 BUILD_GENERAL1_SMALL、BUILD_GENERAL1_MEDIUM 及 BUILD_GENERAL1_LARGE。

`Image`

CodeBuild 用於建置環境的 Docker Hub 或 Amazon ECR 映像名稱。此 Docker 影像應包含建置程式碼所需的各種工具和執行時間程式庫，且應符合應用程式的目標平台。CodeBuild 管理和維護一組專門用於 Elastic Beanstalk 的影像。建議您使用其中一個。如需詳細資訊，請參閱《AWS CodeBuild 使用者指南》中的 CodeBuild 提供的 Docker 映像。

此 `Image` 值是選用的。如果您省略此值，eb init 命令會嘗試選取最符合您目標平台的映像。此外，如果您在互動式模式中執行 eb init，但卻沒有為您選擇映像，系統則會提示您選擇一個映像。在成功初始化結束時，eb init 會將選擇的映像寫入 `buildspec.yml` 檔案。

`Timeout`

CodeBuild 組建版本逾時前可執行的持續時間 (分鐘)。此值是選用的。如需有效值和預設值的詳細資訊，請參閱在 CodeBuild 中建立建置專案。

> **Note**
>
> 此逾時控制 CodeBuild 執行一次的最大持續時間，且 EB CLI 也會採用此逾時做為建立應用程式版本的第一個步驟。這不同於您使用 eb create 的 `--timeout` 選項或 eb deploy 命令指定的數值。後面的數值會控制供 EB CLI 等候環境建立或更新的最大持續時間。

建置和部署您的應用程式碼

每次需要部署應用程式碼時，EB CLI 會使用 CodeBuild 執行建置，然後將產生的建置成品部署至您的環境。當您使用 eb create 命令建置應用程式的 Elastic Beanstalk 環境，以及之後每次使用 eb deploy 命令將程式碼變更部署至環境時，就會發生這種情況。

如果 CodeBuild 步驟失敗，則不會開始進行環境建立或部署。

搭配 Git 使用 EB CLI

EB CLI 可與 Git 整合。本章節將概要說明如何搭配 EB CLI 使用 Git。

欲安裝 Git 並初始化您的 Git 儲存庫

1. 造訪 http://git-scm.com 下載 Git 的最新版本。
2. 輸入下列資訊藉此初始化您的 Git 儲存庫：

   ```
   ~/eb$ git init
   ```

   EB CLI 現將得知您的應用程式已設定 Git。
3. 若您尚未執行 eb init，請現在執行：

   ```
   ~/eb$ eb init
   ```

將 Elastic Beanstalk 環境與 Git 分支建立關聯

您可將不同的環境與程式碼的每個分支建立關聯。切換分支時，變更會部署至關聯的環境。例如，您可輸入下列內容，將您的生產環境與主線分支建立關聯，並將不同的開發環境與您的開發分支建立關聯：

```
~/eb$ git checkout mainline
~/eb$ eb use prod
~/eb$ git checkout develop
~/eb$ eb use dev
```

部署變更

EB CLI 預設會使用遞交 ID 和訊息分別當做應用程式版本標籤及描述，部署目前分支的最新遞交內容。若您希望在不遞交的情況下即可向環境部署，可使用 `--staged` 選向來部署已新增至暫存區的變更。

欲在不遞交的情況下部署變更

1. 將新檔案和已變更檔案新增至暫存區：

   ```
   ~/eb$ git add .
   ```
2. 透過 eb deploy 部署暫存變更：

   ```
   ~/eb$ eb deploy --staged
   ```

若您已將 EB CLI 設定為部署成品，而且您不要將成品遞交至 Git 儲存庫，請使用 `--staged` 選項來部署最新的建置。

使用 Git 子模組

部分程式碼專案受益於 Git 子模組，此為位於最上層儲存庫的儲存庫。當您使用 eb create 或 eb deploy 來部署程式碼，EB CLI 可將子模組納入應用程式版本 zip 檔，並將其與剩餘的程式碼一同上傳。

您可於專案資料夾內的 EB CLI 組態檔案 `include_git_submodules`，透過 `global` 區段的 `.elasticbeanstalk/config.yml` 選項，藉此控制子模組的納入情形。

欲納入子模組，請將此選項設定為 `true`：

```
global:
  include_git_submodules: true
```

缺少 `include_git_submodules` 選項或此選項設定為 `false` 時，EB CLI 不會將子模組納入上傳的 zip 檔。

如需 Git 子模組的詳細資訊，請參閱 Git Tools - 子模組。

> **預設行為**
>
> 執行 eb init 來設定您的專案時，EB CLI 會新增 `include_git_submodules` 選項並將其設定為 `true`。此能確保您專案的子模組都納入部署作業中。
> EB CLI 不一定會支援納入子模組。在新增子模組支援前，為了避免已有專案出現意外或不想要的變更，若缺少 `include_git_submodules` 選項則 EB CLI 不會納入子模組。若您希望在其中一個現有專案的部署作業納入子模組，請如本章節所述，新增此選項並將其設定為 `true`。

> **CodeCommit 行為**
>
> Elastic Beanstalk 與 CodeCommit 的整合目前不支援子模組。若您啟用環境與 CodeCommit 的整合，子模組不會納入您的部署。

將 Git 標籤指派至您的應用程式版本

您可使用 Git 標籤做為版本標籤，藉此辨識您環境正執行的應用程式版本。例如，請輸入下列內容：

```
~/eb$ git tag -a v1.0 -m "My version 1.0"
```

搭配 AWS CodeCommit 使用 EB CLI

您可使用 EB CLI，直接自 AWS CodeCommit 儲存庫部署您的應用程式。透過 CodeCommit，您部署時可僅將變更上傳至儲存庫，無須上傳整個專案。若您的專案較大或網際網路連線能力受限，如此可節省您的時間和頻寬。使用 eb appversion、eb create 或 eb deploy 時，EB CLI 會推播本機遞交內容，並用其建立應用程式版本。

若要部署變更，CodeCommit 整合會要求您先遞交變更。然而，當您開發或除錯時，建議您不要推播正處理待確認的變更。您可以暫存變更並使用會執行標準部署的 eb deploy --staged，藉此避免遞交變更。或者，先將您的變更遞交至開發或測試分支，並在您程式碼就緒後，再合併至主線分支。透過 eb use，您可將 EB CLI 設定為自您的開發分支部署至環境，並自主線分支部署至不同環境。

> **Note**
>
> 某些區域未提供 CodeCommit。Elastic Beanstalk 和 CodeCommit 之間的整合在這些區域中不會發生作用。
> 如需各區域提供的 AWS 服務的資訊，請參閱區域表。

章節

必要條件

若要將 CodeCommit 與 AWS Elastic Beanstalk 搭配使用，您需要帶有至少一個遞交的本機 Git 儲存庫 (現有或新建立的皆可)，使用 CodeCommit 的許可，以及 CodeCommit 支援區域內的 Elastic Beanstalk 環境。您的環境和儲存庫必須位於相同區域。

欲初始化 Git 儲存庫

1. 於專案資料夾執行 `git init`。

```
~/my-app$ git init
```

2. 使用 `git add` 暫存您的專案檔案。

```
~/my-app$ git add .
```

3. 使用 `git commit` 遞交變更。

```
~/my-app$ git commit -m "Elastic Beanstalk application"
```

使用 EB CLI 建立 CodeCommit 儲存庫

若要開始使用 CodeCommit，請執行 eb init。在儲存庫設定期間，EB CLI 會提示您使用 CodeCommit 來存放您的程式碼，並加速部署。即使您之前已使用 eb init 設定專案，您可以再執行一次來設定 CodeCommit。

使用 EB CLI 建立 CodeCommit 儲存庫

1. 於專案資料夾執行 eb init。在設定期間，EB CLI 會詢問您是否希望使用 CodeCommit 來存放您的程式碼，並加速部署。如果您之前已使用 eb init 設定專案，您仍可再執行一次來設定 CodeCommit。提示出現後。輸入 **y** 以設定 CodeCommit。

```
~/my-app$ eb init
Note: Elastic Beanstalk now supports AWS CodeCommit; a fully-managed source control
 service. To learn more, see Docs: https://aws.amazon.com/codecommit/
Do you wish to continue with CodeCommit? (y/n)(default is n): y
```

2. 選擇 Create new Repository (建立新的儲存庫)。

```
Select a repository
1) my-repo
2) [ Create new Repository ]
(default is 2): 2
```

3. 輸入儲存庫名稱或按 Enter (確認) 接受預設名稱。

```
Enter Repository Name
(default is "codecommit-origin"): my-app
Successfully created repository: my-app
```

4. 選擇遞交內容的現有分支，或使用 EB CLI 來建立新的分支。

```
Enter Branch Name
***** Must have at least one commit to create a new branch with CodeCommit *****
(default is "mainline"): ENTER
Successfully created branch: mainline
```

從您的 CodeCommit 儲存庫進行部署。

當您透過 EB CLI 儲存庫設定 CodeCommit 時，EB CLI 會使用儲存庫的內容來建立原始碼套件。執行 eb deploy 或 eb create 時，EB CLI 會推播新的遞交內容，並使用分支的 HEAD 修訂版來建立封存，即可將其部署至環境內的 EC2 執行個體。

搭配 EB CLI 使用 CodeCommit 整合

1. 使用 eb create 建立新環境。

```
~/my-app$ eb create my-app-env
Starting environment deployment via CodeCommit
--- Waiting for application versions to be pre-processed ---
Finished processing application version app-ac1ea-161010_201918
Setting up default branch
Environment details for: my-app-env
  Application name: my-app
  Region: us-east-2
  Deployed Version: app-ac1ea-161010_201918
  Environment ID: e-pm5mvvkfnd
  Platform: 64bit Amazon Linux 2016.03 v2.1.6 running Java 8
  Tier: WebServer-Standard
  CNAME: UNKNOWN
  Updated: 2016-10-10 20:20:29.725000+00:00
Printing Status:
INFO: createEnvironment is starting.
...
```

EB CLI 會使用追蹤分支內最新的遞交內容，以建立將部署至環境的應用程式版本。

2. 當您有新的本機遞交內容，請使用 eb deploy 來推播遞交並部署至您的環境。

```
~/my-app$ eb deploy
Starting environment deployment via CodeCommit
INFO: Environment update is starting.
INFO: Deploying new version to instance(s).
INFO: New application version was deployed to running EC2 instances.
INFO: Environment update completed successfully.
```

3. 欲在遞交前測試變更，請使用 `--staged` 選項來部署您於暫存區域透過 `git add` 新增的變更。

```
~/my-app$ git add new-file
~/my-app$ eb deploy --staged
```

使用 `--staged` 選項進行部署，會繞過 CodeCommit 執行標準部署。

設定其他分支與環境

CodeCommit 組態適用於單一分支。您可使用 eb use 與 eb codesource，以設定其他分支或修改目前分支的組態。

使用 EB CLI 設定 CodeCommit 整合

1. 若要變更遠端分支，請使用 eb use 命令的 `--source` 選項。

```
~/my-app$ eb use test-env --source my-app/test
```

2. 若要建立新的分支和環境，請查看新的分支，將其推播至 CodeCommit，建立環境，然後使用 eb use 來連接本機分支、遠端分支和環境。

```
~/my-app$ git checkout -b production
~/my-app$ git push --set-upstream production
~/my-app$ eb create production-env
~/my-app$ eb use --source my-app/production production-env
```

3. 若要以互動方式設定 CodeCommit，請使用 eb codesource codecommit。

```
~/my-app$ eb codesource codecommit
Current CodeCommit setup:
  Repository: my-app
  Branch: test
```

```
Do you wish to continue (y/n): y

Select a repository
1) my-repo
2) my-app
3) [ Create new Repository ]
(default is 2): 2

Select a branch
1) mainline
2) test
3) [ Create new Branch with local HEAD ]
(default is 1): 1
```

4. 若要停用 CodeCommit 整合，請使用 eb codesource local。

```
~/my-app$ eb codesource local
Current CodeCommit setup:
  Repository: my-app
  Branch: mainline
Default set to use local sources
```

使用現有的 CodeCommit 儲存庫

如果您已有 CodeCommit 儲存庫，希望將其與 Elastic Beanstalk 搭配使用，請在本機 Git 儲存庫的根目錄執行 eb init。

將現有的 CodeCommit 儲存庫與 EB CLI 搭配使用

1. 複製您的 CodeCommit 儲存庫。

```
~$ git clone ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/my-app
```

2. 查看並推播分支以用於您的 Elastic Beanstalk 環境。

```
~/my-app$ git checkout -b dev-env
~/my-app$ git push --set-upstream origin dev-env
```

3. 執行 eb init.選擇您目前正在使用的相同區域、儲存庫和分支名稱。

```
~/my-app$ eb init
```

```
Select a default region
1) us-east-1 : US East (N. Virginia)
2) us-west-1 : US West (N. California)
3) us-west-2 : US West (Oregon)
4) eu-west-1 : Europe (Ireland)
5) eu-central-1 : Europe (Frankfurt)
6) ap-south-1 : Asia Pacific (Mumbai)
7) ap-southeast-1 : Asia Pacific (Singapore)
...
(default is 3): 1
...
Note: Elastic Beanstalk now supports AWS CodeCommit; a fully-managed source control
 service. To learn more, see Docs: https://aws.amazon.com/codecommit/
Do you wish to continue with CodeCommit? (y/n)(default is n): y

Select a repository
1) my-app
2) [ Create new Repository ]
(default is 1): 1

Select a branch
1) mainline
2) dev-env
3) [ Create new Branch with local HEAD ]
(default is 2): 2
```

如需使用 eb init 的詳細資訊，請參閱 設定 EB CLI。

使用 EB CLI 來監控環境運作狀態

Elastic Beanstalk 命令列界面 (EB CLI) 為管理 AWS Elastic Beanstalk 環境的命令列工具。您亦可使用 EB CLI 來即時監控您環境的運作狀態，且精細程度較 Elastic Beanstalk 主控台內現有的監控高。

安裝及設定 EB CLI 後，您可以啟動新的環境，並透過 eb create 命令將程式碼部署其中。如果您已在 Elastic Beanstalk 主控台中建立環境，您可以在專案資料夾中執行 eb init，並依提示操作 (專案資料夾可以是空白的)，將 EB CLI 連接至環境。

> **Important**
>
> 請執行帶有 `pip install` 選項的 `--upgrade`，確認您正使用 EB CLI 的最新版本：

```
$ sudo pip install --upgrade awsebcli
```

如需 EB CLI 的完整安裝說明，請參閱 安裝 EB CLI。

欲使用 EB CLI 來監控您環境的運作狀態，首先必須執行 eb init 並依提示操作，藉此設定本機專案資料夾。如需完整說明，請參閱 設定 EB CLI。

若您已有在 Elastic Beanstalk 中執行的環境，且希望使用 EB CLI 來監控其運作狀態，請遵循這些步驟，將其連接至現有環境。

欲將 EB CLI 連接至現有環境

1. 開啟命令列終端機並瀏覽至您的使用者資料夾。
2. 為您的環境建立並開啟新的資料夾。
3. 執行 eb init 命令，然後選擇欲監控其運作狀態的應用程式和環境。若您只有一個環境執行所選擇的應用程式，您不需要選擇該環境，因為 EB CLI 會自動選取，如以下範例所示。

```
~/project$ eb init
Select an application to use
1) elastic-beanstalk-example
2) [ Create new Application ]
(default is 2): 1
Select the default environment.
You can change this later by typing "eb use [environment_name]".
1) elasticBeanstalkEx2-env
2) elasticBeanstalkExa-env
(default is 1): 1
```

欲使用 EB CLI 來監控運作狀態

1. 開啟命令列並瀏覽至您的專案資料夾。
2. 執行 eb health 命令來顯示環境中執行個體的運作狀態。在此範例中，有五個執行個體在 Linux 環境中執行。

```
~/project $ eb health
 elasticBeanstalkExa-env                                  Ok
 2015-07-08 23:13:20
```

```
WebServer
    Ruby 2.1 (Puma)
  total      ok    warning  degraded  severe    info   pending  unknown
    5        5        0        0        0        0        0        0

  instance-id   status     cause
                                          health
    Overall     Ok
  i-d581497d    Ok
  i-d481497c    Ok
  i-136e00c0    Ok
  i-126e00c1    Ok
  i-8b2cf575    Ok

  instance-id   r/sec    %2xx   %3xx   %4xx   %5xx      p99      p90      p75
 p50     p10                                  requests
    Overall     671.8   100.0    0.0    0.0    0.0    0.003    0.002    0.001
 0.001   0.000
  i-d581497d    143.0    1430      0      0      0    0.003    0.002    0.001
 0.001   0.000
  i-d481497c    128.8    1288      0      0      0    0.003    0.002    0.001
 0.001   0.000
  i-136e00c0    125.4    1254      0      0      0    0.004    0.002    0.001
 0.001   0.000
  i-126e00c1    133.4    1334      0      0      0    0.003    0.002    0.001
 0.001   0.000
  i-8b2cf575    141.2    1412      0      0      0    0.003    0.002    0.001
 0.001   0.000

  instance-id   type       az   running     load 1  load 5      user%  nice%
 system%  idle%   iowait%                             cpu
  i-d581497d    t2.micro   1a   12 mins        0.0    0.04        6.2    0.0
 1.0   92.5       0.1
  i-d481497c    t2.micro   1a   12 mins       0.01    0.09        5.9    0.0
 1.6   92.4       0.1
  i-136e00c0    t2.micro   1b   12 mins       0.15    0.07        5.5    0.0
 0.9   93.2       0.0
  i-126e00c1    t2.micro   1b   12 mins       0.17    0.14        5.7    0.0
 1.4   92.7       0.1
  i-8b2cf575    t2.micro   1c   1 hour        0.19    0.08        6.5    0.0
 1.2   92.1       0.1

  instance-id   status     id   version             ago
                                        deployments
```

```
  i-d581497d    Deployed   1    Sample Application   12 mins
  i-d481497c    Deployed   1    Sample Application   12 mins
  i-136e00c0    Deployed   1    Sample Application   12 mins
  i-126e00c1    Deployed   1    Sample Application   12 mins
  i-8b2cf575    Deployed   1    Sample Application   1 hour
```

在此範例中，有一個單一執行個體在 Windows 環境中執行。

```
~/project $ eb health
 WindowsSampleApp-env                                  Ok
      2018-05-22 17:33:19
WebServer                                              IIS 10.0 running on 64bit
 Windows Server 2016/2.2.0
  total      ok    warning  degraded  severe    info   pending  unknown
    1        1        0        0        0        0        0        0

  instance-id           status     cause
                                          health
    Overall             Ok
  i-065716fba0e08a351   Ok

  instance-id           r/sec    %2xx   %3xx   %4xx   %5xx      p99      p90
 p75     p50     p10                          requests
    Overall              13.7   100.0    0.0    0.0    0.0    1.403    0.970
 0.710   0.413   0.079
  i-065716fba0e08a351     2.4   100.0    0.0    0.0    0.0    1.102*   0.865
 0.601   0.413   0.091

  instance-id           type       az   running     % user time    % privileged
 time  % idle time                                  cpu
  i-065716fba0e08a351   t2.large   1b   4 hours             0.2
 0.1         99.7

  instance-id           status     id   version              ago
                                      deployments
  i-065716fba0e08a351   Deployed   2    Sample Application   4 hours
```

讀取輸出

輸出會在畫面頂端顯示環境名稱、環境整體運作狀態及目前的日期。

```
elasticBeanstalkExa-env                                    Ok
 2015-07-08 23:13:20
```

之後的三行會顯示環境類型 (此案例中是 "WebServer")、組態 (搭配 Puma 的 Ruby 2.1)，以及七種狀態各自的執行個體數量分析。

```
WebServer
 Ruby 2.1 (Puma)
  total      ok    warning  degraded  severe    info   pending  unknown
    5         5        0         0         0         0         0         0
```

輸出的剩餘部分區分為四個區段。第一部分會顯示環境整體的狀態和狀態成因，之後則是每個執行個體的狀態和成因。以下範例顯示環境中兩個狀態為 Info 的執行個體，以及指出已啟動部署的原因。

```
  instance-id    status     cause
                                    health
    Overall      Ok
  i-d581497d     Info       Performing application deployment (running for 3 seconds)
  i-d481497c     Info       Performing application deployment (running for 3 seconds)
  i-136e00c0     Ok
  i-126e00c1     Ok
  i-8b2cf575     Ok
```

如需運作狀態和顏色的資訊，請參閱運作狀態顏色和狀態。

requests (請求) 區段會顯示每一執行個體的 Web 伺服器日誌資訊。在此範例中，各個執行個體皆正常處理請求，未出現錯誤。

```
  instance-id    r/sec     %2xx    %3xx    %4xx    %5xx      p99      p90      p75      p50
    p10                                 requests
    Overall      13.7    100.0     0.0     0.0     0.0    1.403    0.970    0.710    0.413
  0.079
  i-d581497d     2.4     100.0     0.0     0.0     0.0    1.102*   0.865    0.601    0.413
  0.091
  i-d481497c     2.7     100.0     0.0     0.0     0.0    0.842*   0.788    0.480    0.305
  0.062
  i-136e00c0     4.1     100.0     0.0     0.0     0.0    1.520*   1.088    0.883    0.524
  0.104
  i-126e00c1     2.2     100.0     0.0     0.0     0.0    1.334*   0.791    0.760    0.344
  0.197
```

```
  i-8b2cf575     2.3     100.0    0.0    0.0    0.0    1.162*   0.867    0.698   0.477
  0.076
```

cpu 區段會顯示每個執行個體的作業系統指標。輸出會因作業系統而不同。此為 Linux 環境的輸出。

```
  instance-id   type       az   running     load 1  load 5      user%  nice%  system%
idle%   iowait%                              cpu
  i-d581497d    t2.micro   1a   12 mins       0.0    0.03        0.2    0.0      0.0
99.7       0.1
  i-d481497c    t2.micro   1a   12 mins       0.0    0.03        0.3    0.0      0.0
99.7       0.0
  i-136e00c0    t2.micro   1b   12 mins       0.0    0.04        0.1    0.0      0.0
99.9       0.0
  i-126e00c1    t2.micro   1b   12 mins      0.01    0.04        0.2    0.0      0.0
99.7       0.1
  i-8b2cf575    t2.micro   1c   1 hour        0.0    0.01        0.2    0.0      0.1
99.6       0.1
```

此為 Windows 環境的輸出。

```
  instance-id           type       az   running     % user time    % privileged time  %
idle time
  i-065716fba0e08a351   t2.large   1b   4 hours            0.2                  0.0
      99.8
```

如需所顯示的伺服器和作業系統指標資訊，請參閱執行個體指標。

最後的 deployments (部署) 區段，則顯示每一執行個體的部署狀態。若滾動部署失敗，您可以使用所顯示的部署 ID、狀態和版本標籤，辨識環境中執行錯誤版本的執行個體。

```
  instance-id   status     id   version              ago
                                  deployments
  i-d581497d    Deployed   1    Sample Application   12 mins
  i-d481497c    Deployed   1    Sample Application   12 mins
  i-136e00c0    Deployed   1    Sample Application   12 mins
  i-126e00c1    Deployed   1    Sample Application   12 mins
  i-8b2cf575    Deployed   1    Sample Application   1 hour
```

互動式運作狀態檢視畫面

eb health 命令會顯示環境運作狀態的快照。若要每 10 秒重新整理所顯示的資訊，請使用 `--refresh` 選項。

```
$ eb health --refresh
 elasticBeanstalkExa-env                          Ok
 2015-07-09 22:10:04 (1 secs)
WebServer
          Ruby 2.1 (Puma)
  total      ok    warning  degraded  severe    info   pending  unknown
    5        5        0        0        0        0        0        0

  instance-id   status     cause
                                      health
    Overall     Ok
  i-bb65c145    Ok         Application deployment completed 35 seconds ago and took 26
 seconds
  i-ba65c144    Ok         Application deployment completed 17 seconds ago and took 25
 seconds
  i-f6a2d525    Ok         Application deployment completed 53 seconds ago and took 26
 seconds
  i-e8a2d53b    Ok         Application deployment completed 32 seconds ago and took 31
 seconds
  i-e81cca40    Ok

  instance-id   r/sec    %2xx   %3xx   %4xx   %5xx      p99      p90      p75     p50
   p10                                requests
    Overall     671.8   100.0    0.0    0.0    0.0    0.003    0.002    0.001   0.001
 0.000
  i-bb65c145    143.0    1430      0      0      0    0.003    0.002    0.001   0.001
 0.000
  i-ba65c144    128.8    1288      0      0      0    0.003    0.002    0.001   0.001
 0.000
  i-f6a2d525    125.4    1254      0      0      0    0.004    0.002    0.001   0.001
 0.000
  i-e8a2d53b    133.4    1334      0      0      0    0.003    0.002    0.001   0.001
 0.000
  i-e81cca40    141.2    1412      0      0      0    0.003    0.002    0.001   0.001
 0.000

  instance-id   type       az   running     load 1  load 5      user%  nice%  system%
 idle%   iowait%                              cpu
  i-bb65c145    t2.micro   1a   12 mins        0.0    0.03        0.2    0.0      0.0
 99.7       0.1
  i-ba65c144    t2.micro   1a   12 mins        0.0    0.03        0.3    0.0      0.0
 99.7       0.0
```

```
 i-f6a2d525    t2.micro   1b   12 mins        0.0    0.04        0.1    0.0      0.0
99.9       0.0
 i-e8a2d53b    t2.micro   1b   12 mins       0.01    0.04        0.2    0.0      0.0
99.7       0.1
 i-e81cca40    t2.micro   1c   1 hour         0.0    0.01        0.2    0.0      0.1
99.6       0.1

 instance-id   status     id   version              ago
                                 deployments
 i-bb65c145    Deployed   1    Sample Application   12 mins
 i-ba65c144    Deployed   1    Sample Application   12 mins
 i-f6a2d525    Deployed   1    Sample Application   12 mins
 i-e8a2d53b    Deployed   1    Sample Application   12 mins
 i-e81cca40    Deployed   1    Sample Application   1 hour

(Commands: Help,Quit, # # # #)
```

此範例顯示最近自一個執行個體擴展至五個的環境。擴展操作成功，且所有執行個體現正通過運作狀態檢查，並準備好接受請求。在互動式模式中，運作狀態每 10 秒更新。在右上角，計時器會倒數下次更新的時間。

在左下角，報告會顯示選項清單。欲退出互動式模式，請按 Q。欲捲動，請按箭頭鍵。欲查看其他命令清單，請按 H。

互動式運作狀態檢視畫面選項

以互動方式檢視環境運作狀態時，您可以使用鍵盤按鍵來調整檢視，並告知 Elastic Beanstalk 取代或重新啟動個別執行個體。若要在以互動模式檢視運作狀態報告時查看可用命令的清單，請按 H。

```
 up,down,home,end   Scroll vertically
 left,right         Scroll horizontally
 F                  Freeze/unfreeze data
 X                  Replace instance
 B                  Reboot instance
 <,>                Move sort column left/right
 -,+                Sort order descending/ascending
 P                  Save health snapshot data file
 Z                  Toggle color/mono mode
 Q                  Quit this program

 Views
 1                  All tables/split view
```

```
  2                  Status Table
  3                  Request Summary Table
  4                  CPU%/Load Table
  H                  This help menu


(press Q or ESC to return)
```

利用 EB CLI，以群組方式管理多個 Elastic Beanstalk 環境

您可以使用 EB CLI 建立 AWS Elastic Beanstalk 環境群組，每個群組執行不同的服務導向架構應用程式元件。EB CLI 使用 ComposeEnvironments API 來管理這類群組。

> **Note**
>
> 環境群組不同於多容器 Docker 環境中的多個容器。使用環境群組，您的應用程式中的每個元件在個別的 Elastic Beanstalk 環境中執行，有自己專用的一組 Amazon EC2 執行個體。每個元件可分開擴展。有了多容器 Docker，您將一個應用程式中的多個元件結合為單一環境。所有元件共用相同的一組 Amazon EC2 執行個體，每個執行個體執行多個 Docker 容器。根據您的應用程式需求，選擇其中一個架構。
>
> 如需關於多容器 Docker 的詳細資訊，請參閱使用 Amazon ECS 平台分支。

將您的應用程式元件依下列的資料夾結構來整理劃分：

```
~/project-name
|-- component-a
|   `-- env.yaml
`-- component-b
    `-- env.yaml
```

每個子資料夾皆包含應用程式獨立元件的原始碼 (將會在自己的環境中執行)，以及名為 env.yaml 的環境定義檔案。關於 env.yaml 格式的詳細資訊，請參閱環境資訊清單 (env.yaml)。

若要使用 Compose Environments API，請先從專案資料夾中執行 eb init，然後利用 --modules 選項，根據其所屬資料夾的名稱來指定各元件。

```
~/workspace/project-name$ eb init --modules component-a component-b
```

EB CLI 會提示您設定各個元件，然後在每個元件資料夾中建立 .elasticbeanstalk 目錄。EB CLI 不會在父目錄中建立組態檔案。

```
~/project-name
|-- component-a
|   |-- .elasticbeanstalk
|   `-- env.yaml
`-- component-b
    |-- .elasticbeanstalk
    `-- env.yaml
```

接著，執行 eb create 指令，根據環境的清單，針對每個元件各建立一個環境。

```
~/workspace/project-name$ eb create --modules component-a component-b --env-group-suffix group-name
```

此指令會針對每個元件建立環境。將 EnvironmentName 檔案中所指定的 env.yaml，與群組名稱串連在一起，中間以連字號分隔，來做為環境的名稱。這兩個選項加上連字號的總長度，不得超過環境名稱長度的 23 個字元上限。

若要更新環境，請使用 eb deploy 指令：

```
~/workspace/project-name$ eb deploy --modules component-a component-b
```

您可以個別更新每個元件，也可用群組方式來更新這些元件。使用 --modules 選項來指定想要更新的元件。

EB CLI 會將您搭配 eb create 使用的群組名稱，存放於 EB CLI 組態檔案 (branch-defaults) 的 /.elasticbeanstalk/config.yml 區段中。若要將應用程式部署到不同的群組，當您執行 --env-group-suffix 時請使用 eb deploy 選項。如果該群組不存在，EB CLI 會建立多個環境的新群組：

```
~/workspace/project-name$ eb deploy --modules component-a component-b --env-group-suffix group-2-name
```

若要終止環境，請針對每個模組執行資料夾中的 eb terminate。根據預設，如果您嘗試終止另一個運作中環境所相依的環境，EB CLI 將會顯示錯誤。請先終止從屬的環境，或使用 --ignore-links 選項來取代預設的動作：

```
~/workspace/project-name/component-b$ eb terminate --ignore-links
```

使用 EB CLI 來進行問題的故障診斷

此主題列出了在使用 EB CLI 時常見的錯誤訊息，以及可能的解決方法。如果您遇到此處未顯示的錯誤訊息，請利用意見回饋連結告知我們。

錯誤：處理 git 指令時所發生的錯誤。錯誤碼：128 錯誤：嚴重：非有效的物件名稱 HEAD

原因：當您將 Git 儲存庫初始化但尚未執行遞交時，會顯示此錯誤訊息。當您的專案資料夾包含 Git 儲存庫時，EB CLI 會尋找 HEAD 修訂版。

解決方法：在您的專案資料夾中，將檔案新增至整備區，然後執行遞交：

```
~/my-app$ git add .
~/my-app$ git commit -m "First commit"
```

錯誤：此分支不具備預設環境。您必須輸入「eb status my-env-name」來指定環境，或輸入「eb use my-env-name」來設定預設的環境。

原因：當您在 git 中建立新的分支時，該分支預設不會連接到 Elastic Beanstalk 環境。

解決方法：執行 eb list 來查看可用環境的清單。然後執行 eb use ***env-name***，以使用其中一個可用的環境。

錯誤：2.0+ 平台需要服務角色。您可以利用 --service-role 選項來提供

原因：如果您使用 eb create (例如 eb create my-env) 來指定環境名稱，則 EB CLI 不會試著為您建立服務角色。如果不具有預設服務角色，將會顯示上述的錯誤。

解決方法：執行 eb create 而不指定環境名稱，然後再依照提示來建立預設的服務角色。

故障診斷部署

如果您的 Elastic Beanstalk 部署作業進行得不如計畫的順暢，可能會出現 404 (如果應用程式啟動失敗) 或 500 (如果應用程式在執行時間失敗) 的回應訊息，而非顯示網站。若要排除許多常見問題，您可以使用 EB CLI 來查看部署的狀態、檢視其日誌、使用 SSH 來存取您的 EC2 執行個體，或是開啟您應用程式環境的 AWS 管理主控台頁面。

若要使用 EB CLI 來協助排除部署的問題

1. 執行 eb status 以查看目前部署作業的狀態和 EC2 主機的健全狀況。例如：

```
$ eb status --verbose

Environment details for: python_eb_app
  Application name: python_eb_app
  Region: us-west-2
  Deployed Version: app-150206_035343
  Environment ID: e-wa8u6rrmqy
  Platform: 64bit Amazon Linux 2014.09 v1.1.0 running Python 2.7
  Tier: WebServer-Standard-
  CNAME: python_eb_app.elasticbeanstalk.com
  Updated: 2015-02-06 12:00:08.557000+00:00
  Status: Ready
  Health: Green
  Running instances: 1
      i-8000528c: InService
```

> **Note**
>
> 使用 --verbose 條件判斷式，來提供關於執行中執行個體狀態的資訊。如果未使用此條件判斷式，eb status 只會列出關於您環境的一般資訊。

2. 執行 eb health 來檢視關於您環境的健全狀況資訊：

```
$ eb health --refresh
 elasticBeanstalkExa-env                                  Degraded
 2016-03-28 23:13:20
WebServer
    Ruby 2.1 (Puma)
  total      ok    warning  degraded  severe    info   pending  unknown
    5        2        0        2        1        0        0        0

  instance-id   status     cause
    Overall     Degraded  Incorrect application version found on 3 out of 5
 instances. Expected version "Sample Application" (deployment 1).
  i-d581497d    Degraded  Incorrect application version "v2" (deployment 2).
 Expected version "Sample Application" (deployment 1).
  i-d481497c    Degraded  Incorrect application version "v2" (deployment 2).
 Expected version "Sample Application" (deployment 1).
```

```
 i-136e00c0    Severe    Instance ELB health has not been available for 5 minutes.
 i-126e00c1    Ok
 i-8b2cf575    Ok

 instance-id   r/sec    %2xx   %3xx   %4xx   %5xx      p99      p90      p75
p50     p10
   Overall     646.7   100.0    0.0    0.0    0.0    0.003    0.002    0.001
0.001   0.000
 i-dac3f859    167.5    1675      0      0      0    0.003    0.002    0.001
0.001   0.000
 i-05013a81    161.2    1612      0      0      0    0.003    0.002    0.001
0.001   0.000
 i-04013a80    0.0         -      -      -      -        -        -       -
-       -
 i-3ab524a1    155.9    1559      0      0      0    0.003    0.002    0.001
0.001   0.000
 i-bf300d3c    162.1    1621      0      0      0    0.003    0.002    0.001
0.001   0.000

 instance-id   type       az   running     load 1  load 5      user%  nice%
system%  idle%   iowait%
 i-d581497d    t2.micro   1a   25 mins       0.16     0.1        7.0    0.0
1.7   91.0       0.1
 i-d481497c    t2.micro   1a   25 mins       0.14     0.1        7.2    0.0
1.6   91.1       0.0
 i-136e00c0    t2.micro   1b   25 mins        0.0    0.01        0.0    0.0
0.0   99.9       0.1
 i-126e00c1    t2.micro   1b   25 mins       0.03    0.08        6.9    0.0
2.1   90.7       0.1
 i-8b2cf575    t2.micro   1c   1 hour        0.05    0.41        6.9    0.0
2.0   90.9       0.0

 instance-id   status     id   version             ago
     deployments
 i-d581497d    Deployed   2    v2                  9 mins
 i-d481497c    Deployed   2    v2                  7 mins
 i-136e00c0    Failed     2    v2                  5 mins
 i-126e00c1    Deployed   1    Sample Application  25 mins
 i-8b2cf575    Deployed   1    Sample Application  1 hour
```

上述範例顯示了具有 5 個執行個體的環境，其中在第 3 個執行個體上的「v2」版本部署作業已失敗。在部署失敗之後，預期的版本會重設為已經成功的最新版本，在此例中為第一個部署作業的「範例應用程式」。如需詳細資訊，請參閱使用 EB CLI 來監控環境運作狀態。

3. 執行 eb logs，以下載和檢視與您應用程式部署相關的日誌。

```
$ eb logs
```

4. 執行 eb ssh，以連線到執行您應用程式的 EC2 執行個體，並直接對此執行個體進行檢查。在執行個體上，您所部署的應用程式會出現在 `/opt/python/current/app` 目錄中，而您的 Python 環境將會出現在 `/opt/python/run/venv/` 中。
5. 執行 eb console，以在 AWS 管理主控台上檢視您的應用程式環境。您可以使用 Web 介面，來輕鬆地檢查部署作業的各種面向，包括應用程式的組態、狀態、事件和日誌。您也可以下載已部署到伺服器的目前或過去的應用程式版本。

EB CLI 命令參考

您可使用 Elastic Beanstalk 命令列界面 (EB CLI) 來執行各種操作，以部署並管理您的 Elastic Beanstalk 應用程式及環境。欲部署在 Git 來源控制下的應用程式原始碼，EB CLI 可與 Git 整合。如需詳細資訊，請參閱 使用 Elastic Beanstalk 命令列界面 (EB CLI) 及 搭配 Git 使用 EB CLI。

命令

- eb abort
- eb appversion
- eb clone
- eb codesource
- eb config
- eb console
- eb create
- eb deploy
- eb events
- eb health
- eb init
- eb labs
- eb list
- eb local
- eb logs
- eb open

eb abort

描述

在環境資訊變更執行個體過程中，取消升級作業。

> **Note**
>
> 若您正在更新兩個以上的環境，將提示您選取欲還原變更的環境名稱。

語法

eb abort

eb abort ***environment-name***

選項

名稱	描述
常用選項	

輸出

本命令會顯示目前正在更新的環境清單，並提示您選擇欲中止更新的環境。若目前只有一個環境正在更新，您不需要指定環境名稱。若成功，本命令會還原環境資訊變更。轉返程序會持續下去，直到環境中所有執行個體均具備之前的環境資訊，或直到轉返程序失敗。

範例

下列範例取消平台升級。

```
$ eb abort
Aborting update to environment "tmp-dev".
<list of events>
```

eb appversion

描述

EB CLI `appversion` 命令可管理 Elastic Beanstalk 應用程式版本。您可以在不部署的情況下建立新版本的應用程式、刪除應用程式版本或建立應用程式版本生命週期原則。若您未使用任何選項叫用此命令，將進入互動模式。

使用 `--create` 選項來建立新版本的應用程式。

使用 `--delete` 選項來刪除應用程式的版本。

使用 `lifecycle` 選項來顯示或建立應用程式版本生命週期政策。如需更多詳細資訊，請參閱 the section called “版本生命週期”。

語法

eb appversion

eb appversion [-c | --create]

eb appversion [-d | --delete] ***version-label***

eb appversion lifecycle [-p | --print]

選項

名稱	描述 類型：字串
-a *application-name* 或 --application_name *application-name*	應用程式名稱。如果找不到具有指定名稱的應用程式，EB CLI 會為新應用程式建立應用程式版本。 僅適用與 `--create` 選項一起使用。 類型：字串
-c 或 --create	建立新版本的應用程式。
-d *version-label* 或 --delete *version-label*	刪除標記為 *version-label* 的應用程式版本。
`-l` *version_label* 或 `--label` *version_label*	指定用於 EB CLI 建立的版本的標籤。如果您不使用此選項，EB CLI 會產生新的唯一標籤。如果您提供版本標籤，請確保它是唯一的。 僅適用與 `--create` 選項一起使用。 類型：字串
lifecycle	叫用預設編輯器來建立新的應用程式版本生命週期政策。使用此政策可避免達到應用程式版本配額。
lifecycle -p 或 lifecycle --print	顯示目前的應用程式生命週期政策。

名稱	描述 類型：字串
-m "*version_description* " 或 --message "*version_description* "	應用程式版本的描述。它用雙引號括住。 僅適用與 --create 選項一起使用。 類型：字串
-p 或 --process	預處理並驗證原始碼套件中的環境資訊清單和組態檔案。驗證組態檔案可能會識別問題。我們建議您在將應用程式版本部署至環境之前執行此操作。 僅適用與 --create 選項一起使用。
--source codecommit/ *repository-name*/*branch-name*	CodeCommit 儲存庫和分支。如需更多詳細資訊，請參閱 搭配 AWS CodeCommit 使用 EB CLI。 僅適用與 --create 選項一起使用。
--staged	使用 git 索引中暫存的文件 (而不是 HEAD 認可) 來建立應用程式版本。 僅適用與 --create 選項一起使用。
--timeout *##*	命令逾時前的分鐘數。 僅適用與 --create 選項一起使用。
常用選項	

以互動方式使用命令

如果您使用不帶有任何引數的命令，輸出將顯示應用程式版本。它們以相反的時間順序列出，最新版本最先列出。有關畫面外觀的範例，請參閱 Examples (範例) 區段。請注意，狀態行顯示在底端。狀態行會顯示上下文相關資訊。

按 d 來刪除應用程式版本，按 l 來管理您的應用程式生命週期政策，或按 q 來退出且不進行任何變更。

> **Note**
>
> 若該版本已部署至任一環境，將無法刪除該版本。

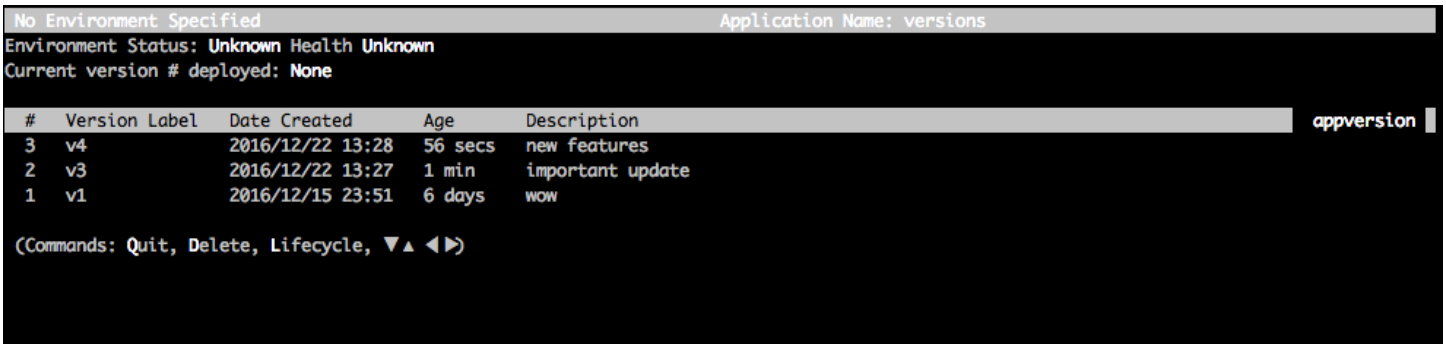
輸出

帶有 `--create` 選項的命令，會顯示應用程式版本已建立的確認訊息。

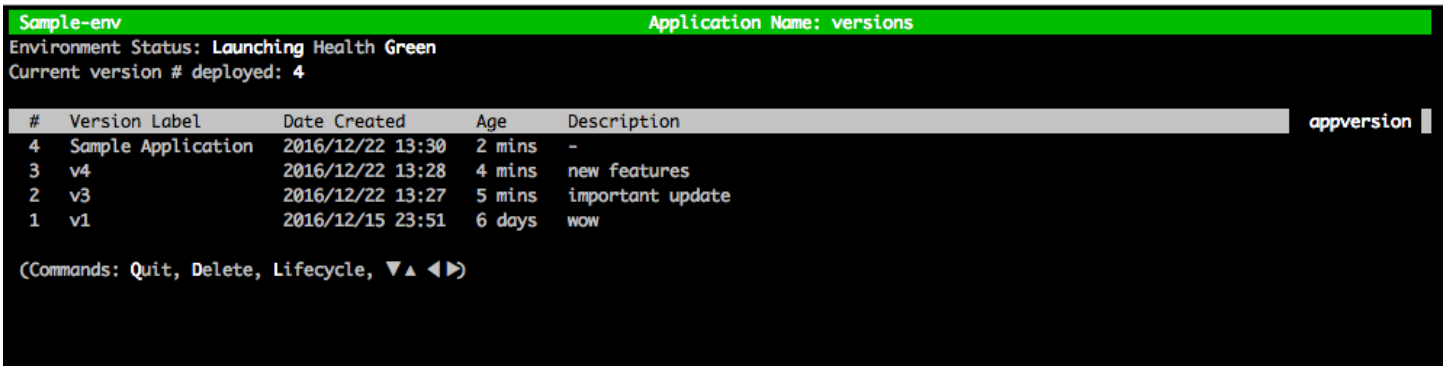
帶有 `--delete` *`version-label`* 選項的命令，會顯示應用程式版本已刪除的確認訊息。

範例

下列範例顯示無任何部署的應用程式之互動視窗。



下列範例為已部署第四版且版本標籤為 Sample Application (範例應用程式) 的應用程式互動視窗。



下列範例為 eb appversion lifecycle -p 命令的輸出，其中 *`ACCOUNT-ID`* 為使用者的帳戶 ID：

```
Application details for: lifecycle
  Region: sa-east-1
  Description: Application created from the EB CLI using "eb init"
```

```
  Date Created: 2016/12/20 02:48 UTC
  Date Updated: 2016/12/20 02:48 UTC
  Application Versions: ['Sample Application']
  Resource Lifecycle Config(s):
    VersionLifecycleConfig:
      MaxCountRule:
        DeleteSourceFromS3: False
        Enabled: False
        MaxCount: 200
      MaxAgeRule:
        DeleteSourceFromS3: False
        Enabled: False
        MaxAgeInDays: 180
    ServiceRole: arn:aws:iam::ACCOUNT-ID:role/aws-elasticbeanstalk-service-role
```

eb clone

描述

將環境複製到新的環境，讓兩個環境擁有相同的環境設定。

> **Note**
>
> 在預設情況下，您用來建立複製環境的環境，無論其解決方案堆疊的版本為何，eb clone 指令都會使用最新的解決方案堆疊來複製環境。您可以在執行該指令時包含 `--exact`，來停用此設定。

> **Important**
>
> 複製的 Elastic Beanstalk 環境不會攜帶安全群組進入，因此環境對所有網際網路流量都開放。您必須為複製的環境重新建立輸入安全性群組。
> 您可以檢查環境組態的漂移狀態，查看可能無法複製的資源。有關詳情，請參閱《AWS CloudFormation 使用者指南》中的偵測整個 CloudFormation 堆疊上的漂移。

語法

eb clone

eb clone *environment-name*

選項

名稱	描述
-n *##* 或 --clone_name *##*	複製的環境所要使用的名稱。
-c *##* 或 --cname *##*	複製的環境所要使用的 CNAME 字首。
--envvars	以逗號分隔清單列出的環境屬性，格式為 *name*=*value*。 類型：字串 約束: • 金鑰-值的對組必須以英文逗號分隔。 • 索引鍵和值可包含任何語言的字母字元、數字字元、空格、隱藏分隔符號和下列符號：_ . : / + \ - @ • 金鑰最多可包含 128 個字元。值最多可包含 256 個字元。 • 金鑰和值會區分大小寫。 • 值不能與環境名稱相同。 • 值不可包含 aws: 或 elasticbeanstalk: 。 • 所有環境屬性加總起來的大小不能超過 4096 位元組。
--exact	防止 Elastic Beanstalk 將新的複製環境的解決方案堆疊版本，更新為可用的最新版本 (適用於原始環境的平台)。
--scale *##*	當複製的環境啟動時，在環境中執行的執行個體的數量。

名稱	描述
--tags *##=#*	標記，用來標記您環境中的資源 (以英文逗號分隔的清單)，格式為 *name*=*value*。
--timeout	命令逾時前的分鐘數。
常用選項	

輸出

如果成功的話，此指令會建立環境，此環境會具有和原始環境相同的設定，也可透過任何 eb clone 選項指定，以修改環境。

範例

下列的範例會複製指定的環境。

```
$ eb clone
Enter name for Environment Clone
(default is tmp-dev-clone):
Enter DNS CNAME prefix
(default is tmp-dev-clone):
Environment details for: tmp-dev-clone
  Application name: tmp
  Region: us-west-2
  Deployed Version: app-141029_144740
  Environment ID: e-vjvrqnn5pv
  Platform: 64bit Amazon Linux 2014.09 v1.0.9 running PHP 5.5
  Tier: WebServer-Standard-1.0
  CNAME: tmp-dev-clone.elasticbeanstalk.com
  Updated: 2014-10-29 22:00:23.008000+00:00
Printing Status:
2018-07-11 21:04:20    INFO: createEnvironment is starting.
2018-07-11 21:04:21    INFO: Using elasticbeanstalk-us-west-2-888888888888 as Amazon S3
 storage bucket for environment data.
...
2018-07-11 21:07:10    INFO: Successfully launched environment: tmp-dev-clone
```

eb codesource

描述

將 EB CLI 設為自 CodeCommit 儲存庫進行部署，或者停用 CodeCommit 整合並從您的本機電腦上傳來源套件。

> **Note**
>
> 某些 AWS 區域未提供 CodeCommit。Elastic Beanstalk 和 CodeCommit 之間的整合在這些區域中不會發生作用。
> 如需各區域提供的 AWS 服務的資訊，請參閱區域表。

語法

eb codesource

eb codesource codecommit

eb codesource local

選項

名稱	描述
常用選項	

輸出

eb codesource 提示您選擇 CodeCommit 整合或標準部署。

eb codesource codecommit 會啟動 CodeCommit 整合的互動式儲存庫組態。

eb codesource local 會顯示原始組態並停用 CodeCommit 整合。

範例

為目前分支使用 eb codesource codecommit 來設定 CodeCommit 整合。

```
~/my-app$ eb codesource codecommit
Select a repository
1) my-repo
2) my-app
3) [ Create new Repository ]
(default is 1): 1

Select a branch
1) mainline
2) test
3) [ Create new Branch with local HEAD ]
(default is 1): 1
```

為目前分支使用 eb codesource local 來停用 CodeCommit 整合。

```
~/my-app$ eb codesource local
Current CodeCommit setup:
  Repository: my-app
  Branch: mainline
Default set to use local sources
```

eb config

描述

管理您環境的作用中組態設定和已儲存組態。您可以使用此命令來上傳、下載或列出您環境的已儲存組態。您也可以使用它來下載、顯示或更新其作用中的組態設定。

如果根目錄包含 `platform.yaml` 檔案，其中指定自訂的平台，則此命令也會變更建置器的組態設定。這是根據 `platform.yaml` 中設定的值完成的。

> **Note**
>
> eb config 不會顯示環境屬性。若要設定可以從您的應用程式中讀取的環境屬性，請改用 eb setenv。

語法

以下是用於 eb config 命令與您環境的作用中組態設定搭配使用的語法部分。如需特定範例，請參閱本主題後半內容中的 範例 部分。

- eb config – 在您設定為 EDITOR 環境變數的文字編輯器中顯示環境的作用中組態設定。當您儲存對檔案進行的變更並關閉編輯器時，會使用檔案中儲存的選項設定來更新環境。

  > **Note**
  >
  > 如果您沒有設定 EDITOR 環境變數，EB CLI 會在 YAML 檔案的預設編輯器中顯示您的選項設定。

- eb config ***environment-name*** – 顯示並更新已命名環境的組態。組態會顯示在您設定的文字編輯器或預設編輯器 YAML 檔案中。
- eb config save – 將目前環境的作用中組態設定儲存至 `.elasticbeanstalk/saved_configs/` (使用 `[configuration-name].cfg.yml` 作為檔案名稱)。根據預設，EB CLI 會根據環境的名稱，使用*`configuration-name`*來儲存組態設定。在執行指令時，您可以加入 `--cfg` 選項搭配所要的組態名稱，來指定不同的組態名稱。

  您可以使用 `--tags` 選項來標記已儲存的組態。
- eb config **`--display`** – 將環境的作用中組態設定寫入 stdout，而不是檔案。依預設，這將顯示終端機的組態設定。
- eb config **`--update`** ***`configuration_string`*** **`|`** ***`file_path`*** – 使用在 *`configuration_string`* 中指定的資訊，或在 *`file_path`* 識別的檔案內部，更新目前環境的作用中組態設定。

> **Note**
>
> `--display` 和 `--update` 選項提供了以程式設計方式讀取和修訂環境組態設定的靈活性。

下文說明使用 eb config 命令來使用已儲存組態的語法。如需範例，請參閱本主題後半內容中的 範例 部分。

- eb config get ***`config-name`*** – 從 Amazon S3 下載已命名儲存的組態。

- eb config delete ***config-name*** – 刪除 Amazon S3 中已命名儲存的組態。如果您已下載它，也會在本機刪除它。
- eb config list - 列出在 Amazon S3 中的已儲存組態。
- eb config put ***filename*** - 將已命名儲存的組態上傳到 Amazon S3 儲存貯體。####必須有副檔名 .cfg.yml。若要指定不包含路徑的檔案名稱，您可以在執行指令之前，將檔案儲存到 .elasticbeanstalk 或 .elasticbeanstalk/saved_configs/ 資料夾。或者，您可以提供完整的路徑來指定####。

選項

名稱	描述
--cfg *config-name*	用於已儲存組態的名稱。 此選項僅適用 eb config save。
-d 或 --display	顯示目前環境的組態設定 (寫入 stdout)。 與 --format 選項一起使用，以指定要在 JSON 或 YAML 中的輸出。如果您未指定，輸出採用 YAML 格式。 只有在您使用不含任何其他子命令的 eb config 命令時，此選項才起作用。
-f *format_type* 或 --format *format_type*	指定顯示格式。有效值是 JSON 或 YAML。 預設為 YAML。 此選項僅與 --display 選項一起使用。
--tags *key1*=*value1*[,*ke*	要新增到您的已儲存的組態的標籤。在清單中指定標籤時，將它們指定為 key=value 對，並以逗號分隔每個標籤。 如需更多詳細資訊，請參閱 標記已儲存組態。 此選項僅適用 eb config save。
--timeout *timeout*	命令逾時前的分鐘數。

名稱	描述

`-u` *configuration_string* | *file_path*
 	
或
 	
`--update` *configuration_string* | *file_path*
	更新目前環境的作用中組態設定。
	
	只有在您使用不含任何其他子命令的 eb config 命令時，此選項才起作用。
	
	configuration_string | *file_path* 參數屬於字串類型。字串提供命名空間清單和對應選項，以新增、更新或從環境的組態設定中移除。或者，輸入字串可以代表包含相同資訊的檔案。
	
	若要指定檔案名稱，輸入字串必須遵循格式 "file://<*path*><*filename*>"。若要指定不包含 *path* 的檔案名稱，請將檔案儲存至您執行命令的資料夾。或者，透過提供完整的路徑來指定檔案名稱。
	
	組態資訊必須符合下列條件。至少需要其中一個區段 OptionSettings 或 OptionsToRemove。使用 OptionSettings 來新增或變更選項。使用 OptionsToRemove 從命名空間中移除選項。如需特定範例，請參閱本主題後半內容中的 範例 部分。
	
	Example
	
	YAML 格式
	
	`OptionSettings:`
	`  namespace1:`
	`    option-name-1:` *option-value-1*
	`    option-name-2:` *option-value-2*
	`    ...`
	`OptionsToRemove:`
	`  namespace1:`
	`    option-name-1`
	`    option-name-2`
	`    ...`
	
	Example
	
	JSON 格式

名稱	描述

	<pre>{
	  "OptionSettings": {
	  "namespace1": {
	  "option-name-1": " *option-value-1* ",
	  "option-name-2": " *option-value-2* ",
	  ...
	  }
	  },
	  "OptionsToRemove": {
	  "namespace1": {
	  "option-name-1",
	  "option-name-2",
	  ...
	  }
	  }
	}</pre>

常用選項	

輸出

如果在未新增子命令或選項的情況下成功執行 eb config 或 eb config ***environment-name*** 命令，命令會在您設定為 EDITOR 環境變數的文字編輯器中，顯示目前的選項設定。如果您沒有設定 EDITOR 環境變數，EB CLI 會在 YAML 檔案的預設編輯器中顯示您的選項設定。

當您儲存對檔案進行的變更並關閉編輯器時，會使用檔案中儲存的選項設定來更新環境。會顯示下列輸出以確認組態更新。

```
$ eb config myApp-dev
    Printing Status:
    2021-05-19 18:09:45    INFO    Environment update is starting.
    2021-05-19 18:09:55    INFO    Updating environment myApp-dev's configuration
 settings.
    2021-05-19 18:11:20    INFO    Successfully deployed new configuration to
 environment.
```

如果命令使用 `--display` 選項成功執行，會顯示目前環境的組態設定 (寫入 stdout)。

如果指令使用 `get` 參數並執行成功，此指令會顯示您所下載本機副本的位置。

如果指令使用 save 參數並執行成功，此指令會顯示已儲存檔案的位置。

範例

本節說明如何變更用來檢視與編輯選項設定檔案的文字編輯器。

如果使用 Linux 和 UNIX，下列範例會將編輯器變更為 vim：

```
$ export EDITOR=vim
```

如果使用 Linux 和 UNIX，下列範例會將編輯器變更為 /usr/bin/kate 所安裝的任何編輯器。

```
$ export EDITOR=/usr/bin/kate
```

如果使用 Windows，下列範例會將編輯器變更為 Notepad++。

```
> set EDITOR="C:\Program Files\Notepad++\Notepad++.exe
```

本節提供的範例，適用於搭配子命令執行時的 eb config 命令。

下列範例刪除了名為 app-tmp 的已儲存組態。

```
$ eb config delete app-tmp
```

下列範例從您的 Amazon S3 儲存貯體下載了名為 app-tmp 的已儲存組態。

```
$ eb config get app-tmp
```

下列範例列出了您的 Amazon S3 儲存貯體中所儲存組態的名稱。

```
$ eb config list
```

下列範例將名為 app-tmp 的已儲存組態的本機副本，上傳到您的 Amazon S3 儲存貯體。

```
$ eb config put app-tmp
```

下列範例從目前執行的環境儲存了組態設定。如果您未提供用於已儲存組態的名稱，則 Elastic Beanstalk 會根據環境的名稱來將組態檔案命名。例如，名為 tmp-dev 的環境將稱為 tmp-

dev.cfg.yml。Elastic Beanstalk 會將檔案儲存至 /.elasticbeanstalk/saved_configs/ 資料夾。

```
$ eb config save
```

下列範例說明如何使用 --cfg 選項，來將 tmp-dev 環境的組態設定儲存到名為 v1-app-tmp.cfg.yml 的檔案。Elastic Beanstalk 會將檔案儲存至 /.elasticbeanstalk/saved_configs/ 資料夾。如果您未指定環境名稱，則 Elastic Beanstalk 會從目前執行中的環境儲存組態設定。

```
$ eb config save tmp-dev --cfg v1-app-tmp
```

本節提供的範例，適用於在沒有子命令的情況下執行時的 eb config 命令。

下列命令會在文字編輯器中顯示目前環境的選項設定。

```
$ eb config
```

下列命令會在文字編輯器中顯示 my-env 環境的選項設定。

```
$ eb config my-env
```

下列範例會顯示目前環境的選項設定。它以 YAML 格式輸出，因為沒有使用 --format 選項指定特定的格式。

```
$ eb config --display
```

下列範例使用名為 example.txt 的檔案中的規格更新目前環境的選項設定。檔案採用 YAML 或 JSON 格式。EB CLI 會自動偵測檔案格式。

- 命名空間 aws:autoscaling:asg 的 Minsize 選項設定為 1。
- 命名空間 aws:elasticbeanstalk:command 的批次大小設定為 30%。
- 它會從命名空間 AWSEBV2LoadBalancer.aws:elbv2:loadbalancer 移除 IdleTimeout: None (IdleTimeout：無) 的選項設定。

```
$ eb config --update "file://example.txt"
```

Example - filename：**example.txt** – YAML 格式

```
OptionSettings:
  'aws:elasticbeanstalk:command':
    BatchSize: '30'
    BatchSizeType: Percentage
  'aws:autoscaling:asg':
    MinSize: '1'
OptionsToRemove:
  'AWSEBV2LoadBalancer.aws:elbv2:loadbalancer':
    IdleTimeout
```

Example - filename：**example.txt** – JSON 格式

```
{
    "OptionSettings": {
        "aws:elasticbeanstalk:command": {
            "BatchSize": "30",
            "BatchSizeType": "Percentage"
        },
        "aws:autoscaling:asg": {
            "MinSize": "1"
        }
    },
    "OptionsToRemove": {
        "AWSEBV2LoadBalancer.aws:elbv2:loadbalancer": {
            "IdleTimeout"
        }
    }
}
```

下列範例會更新目前環境的選項設定。此命令會將 `aws:autoscaling:asg` 命名空間的 Minsize 選項設定為 1。

> **Note**
>
> 這些範例特定於 Windows PowerShell。它們透過在雙引號 (") 字元前面加一個斜線 (\) 字元來逸出其文字出現。不同的作業系統和命令列環境可能具有不同的逸出序列。因此，我們建議您使用先前範例所示的檔案選項。在檔案中指定組態選項不需要逸出字元，而且在不同的作業系統中是一致的。

下列範例採用 JSON 格式。EB CLI 會偵測格式採用 JSON 還是 YAML。

```
PS C:\Users\myUser\EB_apps\myApp-env>eb config --update '{\"OptionSettings\":
{\"aws:autoscaling:asg\":{\"MaxSize\":\"1\"}}}'
```

下列範例採用 YAML 格式。若要以正確格式輸入 YAML 字串，該命令包括 YAML 檔案中所需的間距和行尾返回。

- 使用 "enter" 或 "return" 鍵結束每一行。
- 以兩個空格開始第二行，以四個空格開始第三行。

```
PS C:\Users\myUser\EB_apps\myApp-env>eb config --update 'OptionSettings:
>>  aws:autoscaling:asg:
>>    MinSize: \"1\"'
```

eb console

描述

開啟瀏覽器以在 Elastic Beanstalk 管理主控台中顯示環境組態儀表板。

若根目錄內含指定自訂平台的 `platform.yaml` 檔案，此命令同時會在 Elastic Beanstalk 管理主控台中顯示 `platform.yaml` 指定的建置器環境組態。

語法

eb console

eb console ***environment-name***

選項

名稱	描述
常用選項	

eb create

描述

建立新的環境，然後在其中部署應用程式版本。

> **Note**
>
> - 若要在 .NET 應用程式上使用 eb create，您必須依照 建立 .NET 應用程式的原始碼套件 所述建立部署套件，接著設定 CLI 組態，將套件部署為成品，如 部署成品而非專案資料夾 所述。
> - 要以 EB CLI 建立環境，就必須有服務角色。您可以在 Elastic Beanstalk 主控台建立環境，以建立服務角色。如果您沒有服務角色，EB CLI 會嘗試在您執行 `eb create` 時為您建立。

您可自數個來源來部署應用程式版本：

- 預設情況：自本機專案目錄下的應用程式原始碼。
- 使用 `--version` 選項：自應用程式內已存在的應用程式版本。
- 您的專案目錄沒有應用程式程式碼，或使用 `--sample` 選項時：自您的環境平台專屬的範例應用程式部署。

語法

eb create

eb create ***environment-name***

環境名稱長度必須介於 4 到 40 個字元。僅可包含字母、數字與連字號 (-)。環境名稱的開頭和結尾不可為連字號。

若您的命令納入環境名稱，EB CLI 不會提示您進行選擇或建立服務角色。

若您執行不帶環境名稱引數的命令，將會以互動式流程執行，並提示您輸入或選取部分設定的值。在此互動式流程中，若您正部署範例應用程式，EB CLI 亦會問您是否要將該範例應用程式下載至本機專案目錄中。下載後，您稍後即可搭配新環境使用 EB CLI，執行需要應用程式程式碼的操作 (如 eb deploy)。

某些互動式流程提示只會在特定情況下顯示。例如，如果您選擇使用 Application Load Balancer (而您的帳戶至少有一個可共享的 Application Load Balancer，則 Elastic Beanstalk 會顯示提示，詢問您是否要使用共享負載平衡器。如果您的帳戶中沒有可共享的 Application Load Balancer，則不會顯示此提示。

選項

這些選項均非必要。若您不帶選項執行 eb create，EB CLI 將提示您輸入或選取每個設定的值。

名稱	描述

`-d`
或
`--branch_default`
	將環境設定為目前儲存庫的預設環境。

`--cfg` *`config-name`*	使用已儲存組態的平台設定，位置位於 `.elasticbeanstalk/saved_configs/` 或您的 Amazon S3 儲存貯體。僅指定該檔案的名稱，不要納入 `.cfg.yml` 副檔名。
`-c` *`subdomain-name`* 或 `--cname` *`subdomain-name`*	CNAME DNS 項目前綴的子網域名稱，可路由至您的網站。 類型：字串 預設：環境名稱

`-db`
或
`--database`
	將資料庫連接至環境。若您將 eb create 搭配 `--database` 選項執行，但沒有 `--database.username` 和 `--database.password` 選項，則 EB CLI 會提示您輸入資料庫的主要使用者名稱及密碼。

`-db.engine` *`##`* 或 `--database.engine` *`##`*	資料庫引擎類型。若您搭配此選項執行 eb create，EB CLI 會啟動連接至資料庫的環境。即使您沒有透過 `--database` 選項執行此命令也是。 類型：字串

名稱	描述
	有效值：mysql、oracle-se1 、postgres、sqlserver-ex 、sqlserver-web、sqlserver-se
-db.i *instance_type* 或 --database.instance *instance_type*	要用於資料庫的 Amazon EC2 執行個體類型。若您搭配此選項執行 eb create，EB CLI 會啟動連接至資料庫的環境。即使您沒有透過 --database 選項執行此命令也是。 類型：字串 有效值： Amazon RDS 支援一組標準的資料庫執行個體。若要為資料庫引擎選取適當的資料庫執行個體，您必須將幾項特定條件一併納入考量。如需詳細資訊，請參閱 Amazon RDS 使用者指南中的資料庫執行個體類別。
-db.pass *##* 或 --database.password *##*	資料庫的密碼。若您搭配此選項執行 eb create，EB CLI 會啟動連接至資料庫的環境。即使您沒有透過 --database 選項執行此命令也是。

名稱	描述
-db.size *number_of_gigabytes* 或 --database.size *number_of_gigabytes*	為資料庫儲存配置的 GB 數。若您搭配此選項執行 eb create，EB CLI 會啟動連接至資料庫的環境。即使您沒有透過 --database 選項執行此命令也是。 類型：數字 有效值： • MySQL - 5 至 1024。預設值為 5。 • Postgres - 5 至 1024。預設值為 5。 • Oracle - 10 至 1024。預設值為 10。 • Microsoft SQL Server Express 版本 - 30。 • Microsoft SQL Server Web 版本 - 30。 • Microsoft SQL Server Standard 版本 - 200。
-db.user *#####* 或 --database.username *#####*	資料庫的使用者名稱。若您搭配此選項執行 eb create，EB CLI 會啟動連接至資料庫的環境，即使此命令未納入 --database 選項。若您執行的 eb create 具備 --database 選項，但沒有 --database.username 和 --database.password 選項，則 EB CLI 會提示您輸入主要資料庫的使用者名稱及密碼。
-db.version *##* 或 --database.version *version*	用於指定資料庫引擎版本。若此旗標已存在，環境將搭配指定版本編號的資料庫啟動，即使 --database 旗標不存在。
--elb-type *##*	負載平衡器類型。 類型：字串 有效值：classic、application 、network 預設：application

名稱	描述

`-es`
 	
或
 	
`--enable-spot`
	為您的環境啟用 Spot 執行個體請求。如需更多詳細資訊，請參閱 Auto Scaling 群組。
	
	相關選項：
	
	• `--instance-types`
	• `--on-demand-base-capacity`
	• `--on-demand-above-base-capacity`
	• `--spot-max-price`

`--env-group-suffix` *`groupname`*	欲附加至環境名稱的群組名稱。僅能搭配 編寫環境 使用。
`--envvars`	以逗號分隔清單列出的環境屬性，格式為 *`name`*=*`value`*。請參閱 設定環境屬性 (環境變數) 了解限制。
`-ip` *`profile_name`* 或 `--instance_profile` *`profile_name`*	具有 IAM 角色的執行個體設定檔，以及您的應用程式存取資源所需的臨時安全登入 AWS 資料。

名稱	描述

`-it`
 	
或
 	
`--instance-types` *`type1`*`[,`*`type2`* `...]`
	您要讓環境使用的逗號分隔 Amazon EC2 執行個體類型清單。如果您未指定此選項，則 Elastic Beanstalk 會提供預設的執行個體類型。
	
	如需詳細資訊，請參閱 Amazon EC2 執行個體 及 Auto Scaling 群組。
	
	Important
	EB CLI 僅將此選項套用至 Spot 執行個體。除非搭配 `--enable-spot` 選項使用此選項，否則 EB CLI 會加以忽略。若要指定隨需執行個體的執行個體類型，請改用 `--intance-type` (沒有 "s") 選項。

`-i`
 	
或
 	
`--instance_type`
	您要讓環境使用的 Amazon EC2 執行個體類型。如果您未指定此選項，則 Elastic Beanstalk 會提供預設的執行個體類型。
	
	如需更多詳細資訊，請參閱 Amazon EC2 執行個體。
	
	Important
	EB CLI 只會將此選項套用於隨需執行個體。請勿搭配 `--enable-spot` 選項使用此選項，因為 EB CLI 會在您如此使用時忽略此選項。若要指定 Spot 執行個體的執行個體類型，請改用 `--intance-types` (有 "s") 選項。

`-k` *`key_name`* 或 `--keyname` *`key_name`*	要搭配安全殼層 (SSH) 用戶端使用的 Amazon EC2 金鑰對名稱，以安全登入執行 Elastic Beanstalk 應用程式的 Amazon EC2 執行個體。若您的 eb create 命令納入此選項，則您提供的該值會覆寫 eb init 指定的金鑰名稱。 有效值：向 Amazon EC2 註冊的現有金鑰名稱

名稱	描述
`-im` *number-of-instances* 或 `--min-instances` *number-of-instances*	要求環境具備的 Amazon EC2 執行個體數目下限。 類型：數值 (整數) 預設：1 有效值：1 至 10000
`-ix` *number-of-instances* 或 `--max-instances` *number-of-instances*	允許環境擁有的 Amazon EC2 執行個體數目上限。 類型：數值 (整數) 預設：4 有效值：1 至 10000
`--modules` *component-a component-b*	欲建立的元件環境清單。僅能搭配編寫環境使用。

`-sb`
 	
或
 	
`--on-demand-base-capacity`
	隨著環境擴展，Auto Scaling 群組在考量 Spot 執行個體前佈建的最小隨需執行個體數量。
	
	此選項只能搭配 `--enable-spot` 選項一起指定。如需更多詳細資訊，請參閱 Auto Scaling 群組。
	
	類型：數值 (整數)
	
	預設：0
	
	有效值：0 到 `--max-instances` (缺少時：`MaxSize` 命名空間中的 `aws:autoscaling:asg` 選項)

名稱	描述

`-sp`
 	
或
 	
`--on-demand-above-base-capacity`
	隨需執行個體百分比 (Auto Scaling 群組超出 `--on-demand-base-capacity` 選項所指定之執行個體數量所怖建的一部分額外容量)。
	
	此選項只能搭配 `--enable-spot` 選項一起指定。如需詳細資訊，請參閱 Auto Scaling 群組。
	
	類型：數值 (整數)
	
	預設：`0` 適用於單一執行個體環境；`70` 適用於負載平衡環境
	
	有效值：`0` 至 `100`

名稱	描述
-p *platform-version* 或 --platform *platform-version*	要使用的平台版本。您可以指定平台名稱、平台名稱及版本、平台分支、解決方案堆疊名稱或解決方案堆疊 ARN。例如： • php、PHP、node.js – 指定平台的最新平台版本 • php-7.2、"PHP 7.2" - 建議使用 (通常是最新版本) 的 PHP 7.2 平台版本 • "PHP 7.2 running on 64bit Amazon Linux" - 此平台分支中建議使用 (通常是最新版本) 的 PHP 平台版本 • "64bit Amazon Linux 2017.09 v2.6.3 running PHP 7.1" - 此解決方案堆疊名稱指定的 PHP 平台版本 • "arn:aws:elasticbeanstalk:us-east-2::platform/PHP 7.1 running on 64bit Amazon Linux/2.6.3" - 此解決方案堆疊 ARN 指定的 PHP 平台版本 使用 eb platform list 取得可用組態的清單。 如果您指定 --platform 選項，即會覆寫 eb init 期間所提供的數值。
-pr 或 --process	預處理並驗證原始碼套件中的環境資訊清單和組態檔案。驗證組態檔案可在將應用程式版本部署至環境前辨識問題。
-r *##* 或 --region *##*	您要 AWS 部署應用程式的區域。 如需可以為此選項指定的值之清單，請參閱《AWS 一般參考》中的AWS Elastic Beanstalk 端點與配額。

名稱	描述
`--sample`	將範例應用程式部署至新的環境，而非儲存庫中的程式碼。
`--scale` *number-of-instances*	以特定數量的執行個體來啟動
`--service-role` *servicerole*	將非預設服務角色指派至環境。 **Note** 請勿輸入 ARN。只輸入角色名稱。Elastic Beanstalk 會在角色名稱前綴正確的值，在內部建立所產生的 ARN。
`-ls` *load-balancer* 或 `--shared-lb` *load-balancer*	將環境設定為使用共享負載平衡器。提供您帳戶中可共享負載平衡器的名稱或 ARN (由您明確建立而不是其他 Elastic Beanstalk 環境所建立的 Application Load Balancer)。如需更多詳細資訊，請參閱 共享 Application Load Balancer。 參數範例： • `FrontEndLB` - 負載平衡器名稱。 • `arn:aws:elasticloadbalancing:us-east-2:123456789012:loadbalancer/app/FrontEndLB/0dbf78d8ad96abbc` - Application Load Balancer ARN。 您只能使用 `--elb-type application` 指定此選項。如果您指定該選項但未指定 `--shared-lb`，Elastic Beanstalk 會為該環境建立專用負載平衡器。

名稱	描述

`-lp ###`
 	
或
 	
`--shared-lb-port ###`
	此環境之共享負載平衡器的預設接聽程式連接埠。Elastic Beanstalk 會新增接聽程式規則，以將此接聽程式的所有流量路由傳送至預設環境程序。如需更多詳細資訊，請參閱共享 Application Load Balancer。
	
	類型：數值 (整數)
	
	預設：80
	
	有效值：代表共享負載平衡器之接聽程式連線埠的任何整數。

`--single`	透過單一 Amazon EC2 執行個體建立環境，無須負載平衡器。 **Warning** 單一執行個體環境並非可立即生產。如果執行個體在部署期間變得不穩定，或者 Elastic Beanstalk 在組態更新期間終止並重新啟動執行個體，您的應用程式可能會有一段時間無法使用。使用單一執行個體環境來開發、測試或封測。使用負載平衡環境來生產。

名稱	描述

`-sm`
 	
或
 	
`--spot-max-price`
	您願意為 Spot 執行個體支付的每單位小時最高價格 (以美元為單位)。
	
	此選項只能搭配 `--enable-spot` 選項一起指定。如需詳細資訊，請參閱 Auto Scaling 群組。
	
	類型：數字 (浮點數)
	
	預設：每個執行個體類型的隨需價格。在此情況下，選項的數值為 `null`。
	
	有效值：`0.001` 至 `20.0`
	
	如需競價型執行個體最高價格選項的建議，請參閱 Amazon EC2 使用者指南中的競價型執行個體定價歷史記錄。

`--tags` *key1*`=`*value1*`[,`*key2*`=va`	標記您環境的資源。指定標籤的格式為以逗號分隔的 `key=value` 對清單。 如需詳細資訊，請參閱 標記環境。

`-t worker`
 	
或
 	
`--tier worker`
	建立工作者環境。忽略此選項來建立 Web 伺服器環境。

`--timeout` *##*	設定命令逾時前的分鐘數。
`--version` *version_label*	指定欲部署至環境的應用程式版本，而非本機專案目錄中的應用程式原始碼。 類型：字串 有效值：現有應用程式版本標籤
`--vpc`	設定您環境的 VPC。納入此選項時，EB CLI 會提示您輸入所有必要的設定，然後才會啟動環境。

名稱	描述
`--vpc.dbsubnets` *subnet1,subnet2*	指定 VPC 內資料庫執行個體的子網路。指定 `--vpc.id` 時為必要。
`--vpc.ec2subnets` *subnet1,subnet2*	指定 VPC 內 Amazon EC2 執行個體的子網路。指定 `--vpc.id` 時為必要。
`--vpc.elbpublic`	在您 VPC 的公有子網路啟動 Elastic Load Balancing 負載平衡器。 您可以使用 `--tier worker` 或 `--single` 選項指定此選項。
`--vpc.elbsubnets` *subnet1,subnet2*	指定 VPC 內 Elastic Load Balancing 負載平衡器的子網路。 您可以使用 `--tier worker` 或 `--single` 選項指定此選項。
`--vpc.id` *ID*	在指定 VPC 內啟動您的環境。
`--vpc.publicip`	在您 VPC 的公有子網路啟動 Amazon EC2 執行個體。 您可以使用 `--tier worker` 選項指定此選項。
`--vpc.securitygroups` *securitygroup1,securitygroup2*	指定安全群組 ID。指定 `--vpc.id` 時為必要。
常用選項	

輸出

若成功，本命令會以問題來提示您，然後回傳建立操作的狀態。若啟動期間出現問題，可使用 eb events 操作取得詳細資訊。

如果您在應用程式中啟用 CodeBuild 支援，eb create會在建置程 CodeBuild 式碼時顯示來自的資訊。如需有關 Elastic Beanstalk CodeBuild 支援的資訊，請參閱。搭配 AWS CodeBuild 使用 EB CLI

範例

下列範例以互動式模式建立環境。

```
$ eb create
Enter Environment Name
(default is tmp-dev): ENTER
Enter DNS CNAME prefix
(default is tmp-dev): ENTER
Select a load balancer type
1) classic
2) application
3) network
(default is 2): ENTER
Environment details for: tmp-dev
  Application name: tmp
  Region: us-east-2
  Deployed Version: app-141029_145448
  Environment ID: e-um3yfrzq22
  Platform: 64bit Amazon Linux 2014.09 v1.0.9 running PHP 5.5
  Tier: WebServer-Standard-1.0
  CNAME: tmp-dev.elasticbeanstalk.com
  Updated: 2014-10-29 21:54:51.063000+00:00
Printing Status:
...
```

下列範例亦以互動式模式建立環境。在此範例中，您的專案目錄沒有應用程式程式碼。此命令會部署範例應用程式，並將其下載至您的本機專案目錄。

```
$ eb create
Enter Environment Name
(default is tmp-dev): ENTER
Enter DNS CNAME prefix
(default is tmp-dev): ENTER
Select a load balancer type
1) classic
2) application
3) network
(default is 2): ENTER
NOTE: The current directory does not contain any source code. Elastic Beanstalk is
 launching the sample application instead.
Do you want to download the sample application into the current directory?
(Y/n): ENTER
```

```
INFO: Downloading sample application to the current directory.
INFO: Download complete.
Environment details for: tmp-dev
  Application name: tmp
  Region: us-east-2
  Deployed Version: Sample Application
  Environment ID: e-um3yfrzq22
  Platform: 64bit Amazon Linux 2014.09 v1.0.9 running PHP 5.5
  Tier: WebServer-Standard-1.0
  CNAME: tmp-dev.elasticbeanstalk.com
  Updated: 2017-11-08 21:54:51.063000+00:00
Printing Status:
...
```

下列命令會建立環境，且不會顯示任何提示。

```
$ eb create dev-env
Creating application version archive "app-160312_014028".
Uploading test/app-160312_014028.zip to S3. This may take a while.
Upload Complete.
Application test has been created.
Environment details for: dev-env
  Application name: test
  Region: us-east-2
  Deployed Version: app-160312_014028
  Environment ID: e-6fgpkjxyyi
  Platform: 64bit Amazon Linux 2015.09 v2.0.8 running PHP 5.6
  Tier: WebServer-Standard
  CNAME: UNKNOWN
  Updated: 2016-03-12 01:40:33.614000+00:00
Printing Status:
...
```

下列命令會在自訂 VPC 內建立環境。

```
$ eb create dev-vpc --vpc.id vpc-0ce8dd99 --vpc.elbsubnets subnet-
b356d7c6,subnet-02f74b0c --vpc.ec2subnets subnet-0bb7f0cd,subnet-3b6697c1 --
vpc.securitygroup sg-70cff265
Creating application version archive "app-160312_014309".
Uploading test/app-160312_014309.zip to S3. This may take a while.
Upload Complete.
Environment details for: dev-vpc
  Application name: test
```

```
  Region: us-east-2
  Deployed Version: app-160312_014309
  Environment ID: e-pqkcip3mns
  Platform: 64bit Amazon Linux 2015.09 v2.0.8 running Java 8
  Tier: WebServer-Standard
  CNAME: UNKNOWN
  Updated: 2016-03-12 01:43:14.057000+00:00
Printing Status:
...
```

eb deploy

描述

將應用程式原始碼套件從初始化專案目錄部署至執行中的應用程式。

若已安裝 git，則 EB CLI 會根據最近的 `git archive` 命令的內容，使用 `.zip` 命令來建立 `git commit` 檔案。

然而，當 `.ebignore` 存在於您的專案目錄時，EB CLI 不會使用 git 命令和語義來建立您的原始碼套件。這表示 EB CLI 會忽略 `.ebignore` 指定的檔案，並納入所有其他檔案，尤其是未遞交的來源檔案。

> **Note**
>
> 您亦可設定 EB CLI 從建置程序部署組建成品，而非建立您專案資料夾的 ZIP 檔案。如需詳細資訊，請參閱 部署成品而非專案資料夾。

語法

eb deploy

eb deploy ***environment-name***

選項

名稱	描述
`-l` *`version_label`*	指定用於 EB CLI 建立的版本的標籤。若已使用該標籤，EB CLI 會重新部署使用該標籤的之前版本。

名稱	描述
或 `--label` *version_label*	類型：字串
`--env-group-suffix` *groupname*	欲附加至環境名稱的群組名稱。僅能搭配 編寫環境 使用。

`-m "`*version_description*`"`
 	
或
 	
`--message "`*version_description*`"`
	應用程式版本的描述，以雙引號括住。
	
	類型：字串

`--modules` *component-a component-b*	欲更新元件的清單。僅能搭配 編寫環境 使用。

`-p`
 	
或
 	
`--process`
	預處理並驗證原始碼套件中的環境資訊清單和組態檔案。驗證組態檔案可在將應用程式版本部署至環境前辨識問題。

`--source codecommit/` *repository-name*`/`*branch-name*	CodeCommit 儲存庫和分支。請參閱搭配 AWS CodeCommit 使用 EB CLI。
`--staged`	部署暫存於 git 索引的檔案，而非 HEAD 遞交的檔案。
`--timeout` *##*	命令逾時前的分鐘數。
`--version` *version_label*	欲部署的現有應用程式版本。 類型：字串
常用選項	

輸出

若成功，本命令會回傳 deploy 操作的狀態。

若您已在應用程式啟用 CodeBuild 支援，eb deploy 會在您建置程式碼時顯示 CodeBuild 的資訊。如需 Elastic Beanstalk 中 CodeBuild 支援的詳細資訊，請參閱搭配 AWS CodeBuild 使用 EB CLI。

範例

下列範例會部署現行應用程式。

```
$ eb deploy
2018-07-11 21:05:22    INFO: Environment update is starting.
2018-07-11 21:05:27    INFO: Deploying new version to instance(s).
2018-07-11 21:05:53    INFO: New application version was deployed to running EC2
 instances.
2018-07-11 21:05:53    INFO: Environment update completed successfully.
```

eb events

描述

回傳環境最近的事件。

如果根目錄包含 platform.yaml 檔案，其中指定自訂的平台，則此指令也會傳回建置器環境的最新事件。

Syntax (語法)

eb events

eb events ***environment-name***

選項

名稱	描述
-f 或	串流事件。若要取消，請按下 CTRL+C。

名稱	描述
--follow	

輸出

如果成功，此指令會傳回最近的事件。

範例

下列範例會傳回最新的事件。

```
$ eb events
2014-10-29 21:55:39     INFO    createEnvironment is starting.
2014-10-29 21:55:40     INFO    Using elasticbeanstalk-us-west-2-111122223333 as Amazon
 S3 storage bucket for environment data.
2014-10-29 21:55:57     INFO    Created load balancer named: awseb-e-r-AWSEBLoa-
NSKUOK5X6Z9J
2014-10-29 21:56:16     INFO    Created security group named: awseb-e-rxgrhjr9bx-stack-
AWSEBSecurityGroup-1UUHU5LZ20ZY7
2014-10-29 21:57:18     INFO    Waiting for EC2 instances to launch. This may take a
 few minutes.
2014-10-29 21:57:18     INFO    Created Auto Scaling group named: awseb-e-rxgrhjr9bx-
stack-AWSEBAutoScalingGroup-1TE320ZCJ9RPD
2014-10-29 21:57:22     INFO    Created Auto Scaling group policy named:
 arn:aws:autoscaling:us-east-2:11122223333:scalingPolicy:2cced9e6-859b-421a-
be63-8ab34771155a:autoScalingGroupName/awseb-e-rxgrhjr9bx-stack-
AWSEBAutoScalingGroup-1TE320ZCJ9RPD:policyName/awseb-e-rxgrhjr9bx-stack-
AWSEBAutoScalingScaleUpPolicy-1I2ZSNVU4APRY
2014-10-29 21:57:22     INFO    Created Auto Scaling group policy named:
 arn:aws:autoscaling:us-east-2:11122223333:scalingPolicy:1f08b863-
bf65-415a-b584-b7fa3a69a0d5:autoScalingGroupName/awseb-e-rxgrhjr9bx-stack-
AWSEBAutoScalingGroup-1TE320ZCJ9RPD:policyName/awseb-e-rxgrhjr9bx-stack-
AWSEBAutoScalingScaleDownPolicy-1E3G7PZKZPSOG
2014-10-29 21:57:25     INFO    Created CloudWatch alarm named: awseb-e-rxgrhjr9bx-
stack-AWSEBCloudwatchAlarmLow-VF5EJ549FZBL
2014-10-29 21:57:25     INFO    Created CloudWatch alarm named: awseb-e-rxgrhjr9bx-
stack-AWSEBCloudwatchAlarmHigh-LA9YEW3O6WJO
2014-10-29 21:58:50     INFO    Added EC2 instance 'i-c7ee492d' to Auto ScalingGroup
 'awseb-e-rxgrhjr9bx-stack-AWSEBAutoScalingGroup-1TE320ZCJ9RPD'.
2014-10-29 21:58:53     INFO    Successfully launched environment: tmp-dev
2014-10-29 21:59:14     INFO    Environment health has been set to GREEN
```

```
2014-10-29 21:59:43     INFO    Adding instance 'i-c7ee492d' to your environment.
```

eb health

描述

回傳環境最近的運作狀態。

如果根目錄包含 `platform.yaml` 檔案，其中指定自訂的平台，則此命令也會傳回建置器環境的最新運作狀態。

語法

eb health

eb health ***environment-name***

選項

名稱	描述
`-r`	以互動方式顯示運作狀態資訊，並在報告新資訊時每 10 秒更新。
或	
`--refresh`	
`--mono`	輸出不要顯示顏色。

輸出

如果成功，此命令會傳回最近的運作狀態。

範例

以下範例會傳回 Linux 環境最近的運作狀態資訊。

```
~/project $ eb health
 elasticBeanstalkExa-env                                 Ok
 2015-07-08 23:13:20
```

```
WebServer
 Ruby 2.1 (Puma)
  total      ok    warning  degraded  severe    info   pending  unknown
    5        5        0        0        0        0        0        0

  instance-id   status     cause
                                        health
    Overall     Ok
  i-d581497d    Ok
  i-d481497c    Ok
  i-136e00c0    Ok
  i-126e00c1    Ok
  i-8b2cf575    Ok

  instance-id   r/sec    %2xx   %3xx   %4xx   %5xx      p99      p90      p75     p50
   p10                                  requests
    Overall     671.8   100.0    0.0    0.0    0.0    0.003    0.002    0.001   0.001
0.000
  i-d581497d    143.0    1430      0      0      0    0.003    0.002    0.001   0.001
0.000
  i-d481497c    128.8    1288      0      0      0    0.003    0.002    0.001   0.001
0.000
  i-136e00c0    125.4    1254      0      0      0    0.004    0.002    0.001   0.001
0.000
  i-126e00c1    133.4    1334      0      0      0    0.003    0.002    0.001   0.001
0.000
  i-8b2cf575    141.2    1412      0      0      0    0.003    0.002    0.001   0.001
0.000

  instance-id   type       az   running     load 1  load 5      user%  nice%  system%
idle%   iowait%                              cpu
  i-d581497d    t2.micro   1a   12 mins        0.0    0.04        6.2    0.0      1.0
92.5       0.1
  i-d481497c    t2.micro   1a   12 mins       0.01    0.09        5.9    0.0      1.6
92.4       0.1
  i-136e00c0    t2.micro   1b   12 mins       0.15    0.07        5.5    0.0      0.9
93.2       0.0
  i-126e00c1    t2.micro   1b   12 mins       0.17    0.14        5.7    0.0      1.4
92.7       0.1
  i-8b2cf575    t2.micro   1c   1 hour        0.19    0.08        6.5    0.0      1.2
92.1       0.1

  instance-id   status     id   version             ago
                                    deployments
```

```
  i-d581497d    Deployed   1    Sample Application   12 mins
  i-d481497c    Deployed   1    Sample Application   12 mins
  i-136e00c0    Deployed   1    Sample Application   12 mins
  i-126e00c1    Deployed   1    Sample Application   12 mins
  i-8b2cf575    Deployed   1    Sample Application   1 hour
```

eb init

描述

透過一系列問題提示，針對使用 EB CLI 為 Elastic Beanstalk 應用程式建立的預設值進行設定。

> **Note**
>
> 您透過 eb init 設定的值，僅適用於目前電腦上的目前目錄與儲存庫。
> 此命令不會在您的 Elastic Beanstalk 帳戶中建立任何內容。若要建立 Elastic Beanstalk 環境，請在執行 eb create 後執行 eb init。

語法

eb init

eb init *application-name*

選項

若您未指示 eb init 選項即執行 `--platform`，EB CLI 將提示您輸入每個設定的值。

> **Note**
>
> 欲使用 eb init 來建立新的金鑰對，`ssh-keygen` 必須安裝於您的本機機器且可從命令列取用。

名稱	描述	

`-i`
 	
`--interactive`
	強制 EB CLI 提示您提供每個 eb init 命令選項的值。	

名稱	描述	
	Note `init` 命令會提示您為沒有 (預設) 值的 eb init 命令選項提供值。首次於目錄中執行 eb init 命令後，EB CLI 可能不會提示您輸入命令選項。因此，欲變更您之前的設定，請使用 `--interactive` 選項。	
`-k` *keyname* `--keyname` *keyname*	要搭配安全殼層 (SSH) 用戶端使用的 Amazon EC2 金鑰對名稱，以安全登入執行 Elastic Beanstalk 應用程式的 Amazon EC2 執行個體。	
`--modules` *folder-1 folder-2*	欲初始化的子目錄清單。僅能搭配 編寫環境 使用。	

名稱	描述	
-p *platform-version* --platform *platform-version*	要使用的平台版本。您可以指定平台名稱、平台名稱及版本、平台分支、解決方案堆疊名稱或解決方案堆疊 ARN。例如： • php、PHP、node.js – 指定平台的最新平台版本 • php-7.2、"PHP 7.2" - 建議使用 (通常是最新版本) 的 PHP 7.2 平台版本 • "PHP 7.2 running on 64bit Amazon Linux" - 此平台分支中建議使用 (通常是最新版本) 的 PHP 平台版本 • "64bit Amazon Linux 2017.09 v2.6.3 running PHP 7.1" - 此解決方案堆疊名稱指定的 PHP 平台版本 • "arn:aws:elasticbeanstalk:us-east-2::platform/PHP 7.1 running on 64bit Amazon Linux/2.6.3" - 此解決方案堆疊 ARN 指定的 PHP 平台版本 使用 eb platform list 取得可用組態的清單。 指定 --platform 選項以略過互動式組態。 **Note** 指定此選項時，EB CLI 不會提示您輸入其他選項的值，而會取得每個選項的預設值。您可指定不想使用預設值的選項。	
--source codecommit/ *repository-name*/*branch-name*	CodeCommit 儲存庫和分支。請參閱搭配 AWS CodeCommit 使用 EB CLI。	

名稱	描述	
`--tags key1=value1`	標記您的應用程式。指定標籤的格式為以逗號分隔的 `key=value` 對清單。 如需詳細資訊，請參閱 標記應用程式。	
常用選項		

CodeBuild 支援

若您在包含 buildspec.yml 檔案的資料夾中執行 eb init，Elastic Beanstalk 會針對選項為 Elastic Beanstalk 專屬選項的 eb_codebuild_settings 項目剖析檔案。如需 Elastic Beanstalk 中 CodeBuild 支援的詳細資訊，請參閱搭配 AWS CodeBuild 使用 EB CLI。

輸出

若成功，本命令將透過一連串提示引導您設定新的 Elastic Beanstalk 應用程式。

範例

下列範例請求會初始化 EB CLI，並提示您輸入應用程式的資訊。以自訂值取代####的文字。

```
$ eb init -i
Select a default region
1) us-east-1 : US East (N. Virginia)
2) us-west-1 : US West (N. California)
3) us-west-2 : US West (Oregon)
4) eu-west-1 : Europe (Ireland)
5) eu-central-1 : Europe (Frankfurt)
6) ap-south-1 : Asia Pacific (Mumbai)
7) ap-southeast-1 : Asia Pacific (Singapore)
...
(default is 3): 3

Select an application to use
1) HelloWorldApp
2) NewApp
3) [ Create new Application ]
(default is 3): 3
```

```
Enter Application Name
(default is "tmp"):
Application tmp has been created.

It appears you are using PHP. Is this correct?
(y/n): y

Select a platform branch.
1) PHP 7.2 running on 64bit Amazon Linux
2) PHP 7.1 running on 64bit Amazon Linux (Deprecated)
3) PHP 7.0 running on 64bit Amazon Linux (Deprecated)
4) PHP 5.6 running on 64bit Amazon Linux (Deprecated)
5) PHP 5.5 running on 64bit Amazon Linux (Deprecated)
6) PHP 5.4 running on 64bit Amazon Linux (Deprecated)
(default is 1): 1
Do you want to set up SSH for your instances?
(y/n): y

Select a keypair.
1) aws-eb
2) [ Create new KeyPair ]
(default is 2): 1
```

eb labs

描述

eb labs 的子命令支援進行中或實驗性功能。這些命令可能會於 EB CLI 的未來版本移除或修改，且不保證正向相容。

如需可用的子命令清單和描述，請執行 eb labs --help。

eb list

描述

列出目前應用程式中的所有環境，或所有應用程式中的所有環境，如 `--all` 選項所指定。

如果根目錄包含 `platform.yaml` 檔案，其中指定自訂的平台，則此指令也會列出建置器的環境。

語法

eb list

選項

名稱	描述
-a 或 --all	列出所有應用程式的所有環境。
-v 或 --verbose	提供關於所有環境的詳細資訊，包括執行個體。
常用選項	

輸出

如果成功，指令會傳回環境名稱的清單，其中會以星號 (*) 標記您目前的環境。

範例 1

以下範例列出了您的環境，並指出 tmp-dev 是您預設的環境。

```
$ eb list
* tmp-dev
```

範例 2

以下範例列出了您的環境和其他的詳細資訊。

```
$ eb list --verbose
Region: us-west-2
Application: tmp
    Environments: 1
        * tmp-dev : ['i-c7ee492d']
```

eb local

描述

透過 eb local run 於 Docker 本機內執行應用程式的容器。透過 eb local status 檢查應用程式的容器狀態。使用 eb local open 在 Web 瀏覽器開啟應用程式；透過 eb local logs 擷取應用程式的日誌位置。

eb local setenv 和 eb local printenv 可讓您設定及檢視提供給 Docker 容器的環境變數，並透過 eb local run 於本機執行。

您必須使用 eb local 將 Docker 應用程式初始化為 EB CLI 儲存庫，之後才能在其專案目錄執行所有 eb init 命令。

> **Note**
>
> 於執行 Linux 或 macOS 的本機電腦上使用 eb local。此命令不支援 Windows。
> 在 macOS 上使用命令之前，請安裝 Mac 版 Docker，並確保未安裝 boot2docker (或不在執行路徑中)。eb local 命令會嘗試使用 boot2docker (如果存在)，但在 macOS 上此動作無法正確運作。

語法

eb local run

eb local status

eb local open

eb local logs

eb local setenv

eb local printenv

選項

eb local run

名稱	描述
`--envvars` *key1=value1,key2=value2*	設定 EB CLI 將傳送至本機 Docker 容器的環境變數。在多容器環境中，所有變數都會傳送至所有容器。
`--port` *hostport*	將主機上的連接埠對應至容器的外部連接埠。若您未指定此選項，EB CLI 會於主機和容器上使用相同的連接埠。 此選項僅適用於 Docker 平台的應用程式，不適用於多容器 Docker 平台。
常用選項	

eb local status

eb local open

eb local logs

eb local setenv

eb local printenv

名稱	描述
常用選項	

輸出

eb local run

來自 Docker 的狀態訊息。只要應用程式正在執行，將維持啟用。按 Ctrl+C 來停止應用程式。

eb local status

應用程式使用的每個容器的狀態，表示是否執行中。

eb local open

在 Web 瀏覽器開啟應用程式並退出。

eb local logs

應用程式透過 eb local run 於本機執行在您的專案目錄所產生的日誌位置。

eb local setenv

無

eb local printenv

透過 eb local setenv 設定的環境變數名稱和值。

範例

eb local run

```
~/project$ eb local run
Creating elasticbeanstalk_phpapp_1...
Creating elasticbeanstalk_nginxproxy_1...
Attaching to elasticbeanstalk_phpapp_1, elasticbeanstalk_nginxproxy_1
phpapp_1     | [23-Apr-2015 23:24:25] NOTICE: fpm is running, pid 1
phpapp_1     | [23-Apr-2015 23:24:25] NOTICE: ready to handle connections
```

eb local status

檢視您的本機容器狀態：

```
~/project$ eb local status
Platform: 64bit Amazon Linux 2014.09 v1.2.1 running Multi-container Docker 1.3.3
 (Generic)
Container name: elasticbeanstalk_nginxproxy_1
Container ip: 127.0.0.1
Container running: True
Exposed host port(s): 80
Full local URL(s): 127.0.0.1:80

Container name: elasticbeanstalk_phpapp_1
Container ip: 127.0.0.1
Container running: True
Exposed host port(s): None
```

```
Full local URL(s): None
```

eb local logs

檢視目前專案的日誌路徑：

```
~/project$ eb local logs
Elastic Beanstalk will write logs locally to /home/user/project/.elasticbeanstalk/logs/
local.
Logs were most recently created 3 minutes ago and written to /home/user/
project/.elasticbeanstalk/logs/local/150420_234011665784.
```

eb local setenv

設定搭配 eb local run 使用的環境變數。

```
~/project$ eb local setenv PARAM1=value
```

列印透過 eb local setenv 設定的環境變數。

```
~/project$ eb local printenv
Environment Variables:
PARAM1=value
```

eb logs

描述

此 eb logs 命令有兩個不同的用途：啟用或停用將日誌串流到 CloudWatch Logs，以及擷取執行個體日誌或 CloudWatch Logs 日誌。使用 `--cloudwatch-logs` (`-cw`) 選項，該命令會啟用或停用日誌串流。如果沒有此選項，它會擷取日誌。

當擷取日誌、指定 `--all`、`--zip` 或 `--stream` 選項以擷取完整的日誌。如果您不指定這些的任何選項，Elastic Beanstalk 會擷取結尾日誌。

命令會處理指定或預設環境的日誌。相關日誌依容器類型而異。如果根目錄包含 `platform.yaml` 檔案，其中指定自訂的平台，則此命令也會處理建置器環境的日誌。

如需詳細資訊，請參閱the section called “CloudWatch Logs”。

語法

啟用或停用將日誌串流到 CloudWatch Logs：

```
eb logs --cloudwatch-logs [enable | disable] [--cloudwatch-log-source instance |
 environment-health | all] [environment-name]
```

若要擷取執行個體日誌：

```
eb logs [-all | --zip | --stream] [--cloudwatch-log-source instance] [--
instance instance-id] [--log-group log-group] [environment-name]
```

若要擷取環境運作狀態日誌：

```
eb logs [-all | --zip | --stream] --cloudwatch-log-source environment-health
 [environment-name]
```

選項

名稱	描述

`-cw [enable | disable]`
 	
或
 	
`--cloudwatch-logs [enable | disable]`
	啟用或停用將日誌串流到 CloudWatch Logs。若未提供引數，則會啟用日誌串流。如果未另外指定 `--cloudwatch-log-source` (`-cls`) 選項，便會啟用或停用執行個體日誌串流。

`-cls instance | environment-health | all`
 	
或
 	
`--cloudwatch-log-source instance | environment-health | all`
	指定使用 CloudWatch Logs 時的日誌來源。使用啟用或停用形式的命令，這些是可為其啟用或停用 CloudWatch Logs 串流的日誌。使用擷取形式的命令，這些是從 CloudWatch Logs 擷取的日誌。
	
	有效值：
	
	• 含 `--cloudwatch-logs` (啟用或停用) – `instance | environment-health | all`
	• 不含 `--cloudwatch-logs` (擷取) – `instance | environment-health`

名稱	描述
	值含意： • instance (預設) – 執行個體日誌 • environment-health - 環境運作狀態日誌 (只在環境中啟用增強的運作狀態時提供支援) • all- 兩個日誌來源
-a 或 --all	擷取完整日誌，並將其儲存至 .elasticbeanstalk/logs 目錄。
-z 或 --zip	擷取完整日誌，將其壓縮為 .zip 檔案並儲存至 .elasticbeanstalk/logs 目錄。
--stream	串流 (持續輸出) 完整的日誌。使用此選項時，命令會持續執行直到中斷 (按下 **Ctrl+C**)。
-i *instance-id* 或 --instance *instance-id*	僅擷取指定執行個體的日誌。

名稱	描述
`-g` *log-group* 或 `--log-group` *log-group*	指定擷取日誌之來源的 CloudWatch Logs 日誌群組。此選項只在啟用執行個體日誌串流到 CloudWatch Logs 時有效。 如果啟用執行個體日誌串流，而且您未指定 `--log-group` 選項，預設的日誌群組是以下其中一項： • Amazon Linux 2 – `/aws/elasticbeanstalk/` *environment-name* `/var/log/eb-engine.log` • Windows 平台 - `/aws/elasticbeanstalk/` *environment-name* `/EBDeploy-Log` • Amazon Linux AMI (AL1) – `/aws/elasticbeanstalk/` *environment-name* `/var/log/eb-activity.log` **Note** 2022 年 7 月 18 日，Elastic Beanstalk 已將所有以 Amazon Linux AMI (AL1) 為基礎的平台分支狀態設為已淘汰。如需有關遷移至完全支援的目前 Amazon Linux 2023 平台分支的詳細資訊，請參閱 將您的 Elastic Beanstalk Linux 應用程式遷移到 Amazon Linux 2023 或 Amazon Linux 2。 如需有關對應到每個日誌檔的日誌群組的詳細資訊，請參閱 Elastic Beanstalk 如何設定 CloudWatch Logs。
常用選項	

輸出

在預設情況下，會直接在終端機顯示日誌。使用分頁程式以顯示輸出。按下 **Q** 或 **q** 以退出。

使用 `--stream`，在終端機顯示現有的日誌，並持續執行。按下 **Ctrl+C** 離開。

使用 `--all` 和 `--zip`，將日誌儲存到本機檔案，並顯示檔案的位置。

範例

以下範例啟用將執行個體日誌串流到 CloudWatch Logs。

```
$ eb logs -cw enable
Enabling instance log streaming to CloudWatch for your environment
After the environment is updated you can view your logs by following the link:
https://console.aws.amazon.com/cloudwatch/home?region=us-east-1#logs:prefix=/aws/
elasticbeanstalk/environment-name/
Printing Status:
2018-07-11 21:05:20    INFO: Environment update is starting.
2018-07-11 21:05:27    INFO: Updating environment environment-name's configuration
 settings.
2018-07-11 21:06:45    INFO: Successfully deployed new configuration to environment.
```

以下範例擷取執行個體日誌到 `.zip` 檔案。

```
$ eb logs --zip
Retrieving logs...
Logs were saved to /home/workspace/environment/.elasticbeanstalk/logs/150622_173444.zip
```

eb open

描述

在預設瀏覽器開啟您網站的公開 URL。

語法

eb open

eb open ***environment-name***

選項

名稱	描述
常用選項	

輸出

命令 eb open 不會產生輸出，而會在瀏覽器視窗開啟應用程式。

eb platform

描述

本命令支援兩種不同的工作空間：

平台

使用此工作空間來管理自訂平台。

環境

使用此工作空間來選取預設平台，或顯示目前平台的資訊。

Elastic Beanstalk 為 ebp 提供捷徑 eb platform。

> **Note**
>
> Windows PowerShell 使用 ebp 做為命令別名。若您於 Windows PowerShell 執行 EB CLI，請使用此命令的長格式：eb platform。

在自訂平台使用 eb 平台

列出目前平台的版本，並可讓您管理自訂平台。

語法

eb platform create [***version***] [***options***]

eb platform delete [***version***] [***options***]

eb platform events [***version***] [***options***]

eb platform init [***platform***] [***options***]

eb platform list [***options***]

eb platform logs [***version***] [***options***]

eb platform status [***version***] [***options***]

eb platform use [***platform***] [***options***]

選項

名稱	描述
`create [version] [options]`	建立平台的新版本。進一步了解。
`delete version [options]`	刪除平台版本。進一步了解。
`events [version] [options]`	顯示一個平台版本的事件。進一步了解。
`init [platform] [options]`	初始化平台儲存庫。進一步了解。
`list [options]`	列出目前平台的版本。進一步了解。
`logs [version] [options]`	顯示平台版本建置器環境的日誌。進一步了解。
`status [version] [options]`	顯示平台版本的狀態。進一步了解。
`use [platform] [options]`	選取已建立新版本的不同平台。進一步了解。
常用選項	

常用選項

所有 eb platform 命令均包含下列常用選項。

名稱	描述
`-h`	顯示協助訊息並退出。

名稱	描述
或 `--help`	
`--debug`	顯示其他除錯輸出。
`--quiet`	抑制所有輸出。

`-v`
或
`--verbose`
	顯示其他輸出。

`--profile` *PROFILE*	使用您登入資料指定的 *PROFILE*。
`-r` *REGION* 或 `--region` *REGION*	使用區域 *REGION*。
`--no-verify-ssl`	不要驗證 AWS SSL 憑證。

Eb 平台建立

建立平台的新版本並回傳其 ARN。若沒有建置器環境在目前區域中執行，本命令會啟動一個建置器環境。*##* 和遞增選項 (`-M`、`-m` 和 `-p`) 互斥。

選項

名稱	描述
version	若未指定 *##*，會依據最近的平台建立新版本，且修補程式版本 (n.n.N 中的 N) 會遞增。
`-M` 或	增加主要版本編號 (N.n.n 中的 N)。

名稱	描述
`--major-increment`	

`-m`
或
`--minor-increment`
	增加次要版本編號 (n.N.n 中的 N)。

`-p`
或
`--patch-increment`
	增加修補程式版本編號 (n.n.N 中的 N)。

`-i` *INSTANCE_TYPE* 或 --instance-type *INSTANCE_TYPE*	將 *INSTANCE_TYPE* 做為執行個體類型 (如 **t1.micro**)。
`-ip` *INSTANCE_PROFILE* 或 `--instance-profile` *INSTANCE_PROFILE*	在建立自訂平台的 AMI 時將 *INSTANCE_PROFILE* 做為執行個體描述檔。 若不指定 `-ip` 選項，會建立執行個體描述檔 `aws-elasticbeanstalk-custom-platforme-ec2-role` 並將其用於自訂平台。
`--tags` *key1*=*value1*[,*ke*	標記您的自訂平台版本。指定標籤的格式為以逗號分隔的 `key=value` 對清單。 如需詳細資訊，請參閱 標記自訂平台版本。
`--timeout` *##*	設定命令逾時前的分鐘數。
`--vpc.id` *VPC_ID*	在其中建立 Packer 的 VPC ID。
`--vpc.subnets` *VPC_SUBNETS*	在其中建立 Packer 的 VPC 子網路。

名稱	描述
--vpc.publicip	將公有 IP 與啟動的 EC2 執行個體建立關聯。

Eb 平台刪除

刪除平台版本。若環境正在使用該版本，將不會刪除。

選項

名稱	描述
version	欲刪除的版本。此值為必填。
--cleanup	移除所有處於 Failed 狀態的平台版本。
--all-platforms	若指定為 --cleanup ，會移除所有平台處於 Failed 狀態的平台版本。
--force	刪除版本時不需要進行確認。

Eb 平台事件

顯示一個平台版本的事件。若已指定*##*，會顯示該版本的事件，否則會顯示目前版本的事件。

選項

名稱	描述
version	欲顯示事件的版本。此值為必填。
-f 或 --follow	在事件發生時持續顯示。

Eb 平台 init

初始化平台儲存庫。

選項

名稱	描述
platform	欲初始化的平台名稱。此值為必填，除非啟用 -i (互動式模式)。
-i 或 --interactive	使用互動式模式。
-k *KEYNAME* 或 --keyname *KEYNAME*	預設 EC2 金鑰名稱。

您可於之前已初始化的目錄執行本命令，但是將無法變更工作空間類型。

欲使用不同選項重新初始化，請使用 -i 選項。

Eb 平台清單

列出與工作空間 (目錄) 或區域關聯的平台版本。

根據您所執行的工作空間類型，命令會傳回不同結果，如下所示：

- 在平台工作空間 (由 eb platform init 進行目錄初始化)，命令傳回工作空間所定義之自訂平台的所有平台版本清單。新增 --all-platforms 或 --verbose 選項，取得與工作空間關聯的區域中，您帳戶所擁有的所有自訂平台的平台版本清單。
- 在應用程式工作空間 (由 eb init 進行目錄初始化)，命令傳回所有平台版本，包括 Elastic Beanstalk 受管的平台以及您帳戶的自訂平台。此清單使用較短的平台版本名稱，且部分平台版本變化可能會合併。新增 --verbose 選項以取得詳細清單，其中包含完整名稱並分別列出所有變化。
- 在未初始化目錄，命令只可搭配 --region 選項。這會傳回此區域所支援的所有 Elastic Beanstalk 受管平台版本的清單。此清單使用較短的平台版本名稱，且部分平台版本變化可能會合併。新增 --verbose 選項以取得詳細清單，其中包含完整名稱並分別列出所有變化。

選項

名稱	描述

`-a`
或
`--all-platforms`
	僅在初始化工作空間為有效 (由 `eb platform init` 或 `eb init` 進行目錄初始化)。列出與您帳戶關聯的所有自訂平台的版本清單。

`-s` *STATUS* 或 `--status` *STATUS*	僅列出 *STATUS* 相符的平台： • 備妥 • 失敗 • 正在刪除 • 正在建立

Eb 平台日誌

顯示平台版本建置器環境的日誌。

選項

名稱	描述
version	欲顯示記錄的平台版本。若省略，則會顯示目前版本的日誌。
`--stream`	串流透過 CloudWatch 設定的部署日誌。

Eb 平台狀態

顯示平台版本的狀態。

選項

名稱	描述
version	欲擷取狀態的平台版本。若省略，則會顯示目前版本的狀態。

Eb 平台使用

選取已建立新版本的不同平台。

選項

名稱	描述
platform	將 ## 指定為本工作空間的使用中版本。此值為必填。

將 eb platform 用於環境

列出支援平台，並讓您設定啟動環境時所用的預設平台和平台版本。使用 eb platform list 來檢視所有支援平台的清單。使用 eb platform select 來變更您專案的平台。使用 eb platform show 來檢視您專案所選的平台。

語法

eb platform list

eb platform select

eb platform show

選項

名稱	描述
list	列出目前平台的版本。
select	選取預設平台。
show	顯示目前平台的資訊。

範例 1

下列範例列出 Elastic Beanstalk 支援的平台所有組態名稱。

```
$ eb platform list
docker-1.5.0
```

```
glassfish-4.0-java-7-(preconfigured-docker)
glassfish-4.1-java-8-(preconfigured-docker)
go-1.3-(preconfigured-docker)
go-1.4-(preconfigured-docker)
iis-7.5
iis-8
iis-8.5
multi-container-docker-1.3.3-(generic)
node.js
php-5.3
php-5.4
php-5.5
python
python-2.7
python-3.4
python-3.4-(preconfigured-docker)
ruby-1.9.3
ruby-2.0-(passenger-standalone)
ruby-2.0-(puma)
ruby-2.1-(passenger-standalone)
ruby-2.1-(puma)
ruby-2.2-(passenger-standalone)
ruby-2.2-(puma)
tomcat-6
tomcat-7
tomcat-7-java-6
tomcat-7-java-7
tomcat-8-java-8
```

範例 2

下列範例會提示您為指定平台選擇欲部署的平台和版本清單。

```
$ eb platform select
Select a platform.
1) PHP
2) Node.js
3) IIS
4) Tomcat
5) Python
6) Ruby
7) Docker
8) Multi-container Docker
9) GlassFish
```

```
10) Go
(default is 1): 5

Select a platform version.
1) Python 2.7
2) Python
3) Python 3.4 (Preconfigured - Docker)
```

範例 3

下列範例會顯示目前預設平台的資訊。

```
$ eb platform show
Current default platform: Python 2.7
New environments will be running:  64bit Amazon Linux 2014.09 v1.2.0 running Python 2.7

Platform info for environment "tmp-dev":
Current: 64bit Amazon Linux 2014.09 v1.2.0 running Python
Latest:  64bit Amazon Linux 2014.09 v1.2.0 running Python
```

eb printenv

描述

於命令視窗列印所有環境屬性。

語法

eb printenv

eb printenv ***environment-name***

選項

名稱	描述
常用選項	

輸出

若成功，本命令會回傳 printenv 操作的狀態。

範例

下列範例列印指定環境的環境屬性。

```
$ eb printenv
Environment Variables:
     PARAM1 = Value1
```

eb restore

描述

重建已終止環境，建立具備相同名稱、ID 和組態的新環境。環境名稱、網域名稱和應用程式版本必須能夠使用才能重建成功。

語法

eb restore

eb restore ***environment_id***

選項

名稱	描述
常用選項	

輸出

EB CLI 顯示可還原的已終止環境清單。

範例

```
$ eb restore
Select a terminated environment to restore

  #   Name          ID             Application Version      Date Terminated        Ago
```

```
  3   gamma          e-s7mimej8e9   app-77e3-161213_211138   2016/12/14 20:32 PST   13
mins
  2   beta           e-sj28uu2wia   app-77e3-161213_211125   2016/12/14 20:32 PST   13
mins
  1   alpha          e-gia8mphu6q   app-77e3-161213_211109   2016/12/14 16:21 PST   4
hours

(Commands: Quit, Restore, # #)

Selected environment alpha
Application:    scorekeep
Description:    Environment created from the EB CLI using "eb create"
CNAME:          alpha.h23tbtbm92.us-east-2.elasticbeanstalk.com
Version:        app-77e3-161213_211109
Platform:       64bit Amazon Linux 2016.03 v2.1.6 running Java 8
Terminated:     2016/12/14 16:21 PST
Restore this environment? [y/n]: y

2018-07-11 21:04:20    INFO: restoreEnvironment is starting.
2018-07-11 21:04:39    INFO: Created security group named: sg-e2443f72
...
```

eb scale

描述

設定執行個體數量的上下限，將環境擴展以永遠於特定數量的執行個體上執行。

語法

eb scale ***number-of-instances***

eb scale ***number-of-instances environment-name***

選項

名稱	描述
--timeout	命令逾時前的分鐘數。
常用選項	

輸出

若成功，本命令會將執行個體數量的上下限更新為特定數量。

範例

下列範例將執行個體數量設定為 2。

```
$ eb scale 2
2018-07-11 21:05:22    INFO: Environment update is starting.
2018-07-11 21:05:27    INFO: Updating environment tmp-dev's configuration settings.
2018-07-11 21:08:53    INFO: Added EC2 instance 'i-5fce3d53' to Auto Scaling Group
 'awseb-e-2cpfjbra9a-stack-AWSEBAutoScalingGroup-7AXY7U13ZQ6E'.
2018-07-11 21:08:58    INFO: Successfully deployed new configuration to environment.
2018-07-11 21:08:59    INFO: Environment update completed successfully.
```

eb setenv

描述

設定預設環境的環境屬性。

語法

eb setenv ***key*=*value***

您可依需求納入任意數量的屬性，但所有屬性的總大小不能超過 4096 個位元組。您可將值保留空白，藉此刪除變數。請參閱 設定環境屬性 (環境變數) 了解限制。

> **Note**
>
> 若 `value` 包含特殊字元，務必在前方加上 \ 字元來加以逸出。

選項

名稱	描述
--timeout	命令逾時前的分鐘數。

名稱	描述
常用選項	

輸出

若成功，本命令會顯示環境更新成功。

範例

下列範例設定環境變數 ExampleVar。

```
$ eb setenv ExampleVar=ExampleValue
2018-07-11 21:05:25    INFO: Environment update is starting.
2018-07-11 21:05:29    INFO: Updating environment tmp-dev's configuration settings.
2018-07-11 21:06:50    INFO: Successfully deployed new configuration to environment.
2018-07-11 21:06:51    INFO: Environment update completed successfully.
```

下列命令設定多個環境屬性。其會新增名為 `foo` 的環境屬性並將該值設定為 `bar`，變更 `JDBC_CONNECTION_STRING` 屬性的值，並刪除 `PARAM4` 和 `PARAM5` 屬性。

```
$ eb setenv foo=bar JDBC_CONNECTION_STRING=hello PARAM4= PARAM5=
```

eb ssh

描述

> **Note**
>
> 本命令不適用執行 Windows Server 執行個體的環境。

使用安全殼層 (SSH) 連接至環境中的 Linux Amazon EC2 執行個體。若環境具有多個運作中的執行個體，EB CLI 會提示您指定欲連接的執行個體。若要使用此命令，SSH 必須安裝於本機機器，並可從命令列使用。私有金鑰檔案必須位於使用者目錄底下名為 `.ssh` 的資料夾，而環境中的 EC2 執行個體必須具備公有 IP 地址。

若根目錄包含指定自訂平台的 `platform.yaml` 檔案，則此命令也會連接至自訂環境內的執行個體。

> **SSH 金鑰**
>
> 若您尚未設定 SSH，可使用 EB CLI 在執行 eb init 時建立金鑰。若您已執行 eb init，請搭配 `--interactive` 選項再執行一次，然後在提示設定 SSH 時，選取 Yes (是) 及 Create New Keypair (建立新的金鑰對)。在此程序建立的金鑰，將由 EB CLI 存放在適當的資料夾。

本命令暫時開啟您環境安全群組的連接埠 22 供來自 0.0.0.0/0 (所有 IP 地址) 傳入的流量使用 (若尚未針對連接埠 22 設定規則)。若您已將環境安全群組的連接埠 22，設定為向受限 CIDR 範圍開啟以提升安全性，EB CLI 會採用該設定並放棄對該安全群組的任何變更。欲覆寫此行為並要求 EB CLI 開啟連接埠 22 接收所有傳入流量，請使用 `--force` 選項。

如需設定您環境安全群組的資訊，請參閱 安全群組。

語法

eb ssh

eb ssh ***environment-name***

選項

名稱	描述

`-i`
或
`--instance`
	指定欲連接之執行個體的執行個體 ID。建議您使用此選項。

`-n`
或
`--number`
	按編號指定欲連接的執行個體。

`-o` 或	在 SSH 工作階段結束後，將安全群組的連接埠 22 保持開啟。

名稱	描述
--keep_open	
--command	針對特定執行個體執行 shell 命令，而非啟動 SSH 工作階段。
--custom	指定欲使用的 SSH 命令，而非「ssh -i keyfile」。請勿納入遠端使用者和主機名稱。
--setup	變更指派至環境執行個體的金鑰對 (需要替換執行個體)。
--force	針對環境安全群組的連接埠 22，開啟來自 0.0.0.0/0 的傳入流量，不論安全群組是否已設定 SSH。 若您已將環境安全群組的連接埠 22 設定為向受限 CIDR 範圍開啟，其中未包含您正嘗試發送連接的 IP 地址，請使用此選項。
--timeout ##	設定命令逾時前的分鐘數。 僅可搭配 --setup 引數來使用。
常用選項	

輸出

若成功，本命令會開啟 SSH 連線至執行個體。

範例

下列範例會將您連接至指定的環境。

```
$ eb ssh
Select an instance to ssh into
1) i-96133799
2) i-5931e053
(default is 1): 1
INFO: Attempting to open port 22.
INFO: SSH port 22 open.
The authenticity of host '54.191.45.125 (54.191.45.125)' can't be established.
RSA key fingerprint is ee:69:62:df:90:f7:63:af:52:7c:80:60:1b:3b:51:a9.
Are you sure you want to continue connecting (yes/no)? yes
```

```
Warning: Permanently added '54.191.45.125' (RSA) to the list of known hosts.

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

https://aws.amazon.com/amazon-linux-ami/2014.09-release-notes/
No packages needed for security; 1 packages available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-8-185 ~]$ ls
[ec2-user@ip-172-31-8-185 ~]$ exit
logout
Connection to 54.191.45.125 closed.
INFO: Closed port 22 on ec2 instance security group
```

eb status

描述

提供關於環境狀態的資訊。

如果根目錄包含 `platform.yaml` 檔案，其中指定自訂的平台，則此指令也會提供關於建置器環境的資訊。

語法

eb status

eb status ***environment-name***

選項

名稱	描述

`-v`
或
`--verbose`
	提供更多關於個別執行個體的資訊，例如這些執行個體與 Elastic Load Balancing 負載平衡器的狀態。

常用選項	

輸出

如果成功的話，指令會傳回下列關於環境的資訊：

- 環境名稱
- 應用程式名稱
- 部署應用程式版本
- 環境 ID
- 平台
- 環境層
- CNAME
- 環境上次更新的時間
- 狀態
- 運作狀態

如果您使用 verbose 模式，則 EB CLI 也會提供執行中 Amazon EC2 執行個體的數目。

範例

下列範例會顯示環境 tmp-dev 的狀態。

```
$ eb status
Environment details for: tmp-dev
  Application name: tmp
  Region: us-west-2
  Deployed Version: None
  Environment ID: e-2cpfjbra9a
  Platform: 64bit Amazon Linux 2014.09 v1.0.9 running PHP 5.5
  Tier: WebServer-Standard-1.0
  CNAME: tmp-dev.elasticbeanstalk.com
  Updated: 2014-10-29 21:37:19.050000+00:00
  Status: Launching
  Health: Grey
```

eb swap

描述

將某環境的 CNAME 與另一個環境的 CNAME 交換 (例如，為了在更新應用程式版本時避免停機時間)。

> **Note**
>
> 如果您有兩個以上的環境，系統會提示您，請您從環境清單中，針對目前正在使用您所要 CNAME 的環境，選取該環境的名稱。若不要讓此提示出現，您可以藉由在執行指令時加入 `-n` 選項，來指定所要使用的環境的名稱。

語法

eb swap

eb swap ***environment-name***

> **Note**
>
> *environment-name* 是您希望具有不同 CNAME 的環境。如果您不在執行 eb swap 時，指定 *environment-name* 做為命令列的參數，則 EB CLI 會更新預設環境的 CNAME。

選項

名稱	描述

`-n`
或
`--destination_name`
	指定您想要交換其 CNAME 的環境的名稱。如果您未在執行 eb swap 時包含此選項，則 EB CLI 會提示您從環境清單中選擇。

常用選項	

輸出

若成功，本命令會回傳 swap 操作的狀態。

範例

下列範例會將環境 tmp-dev 與 live-env 交換。

```
$ eb swap
Select an environment to swap with.
1) staging-dev
2) live-env
(default is 1): 2
2018-07-11 21:05:25    INFO: swapEnvironmentCNAMEs is starting.
2018-07-11 21:05:26    INFO: Swapping CNAMEs for environments 'tmp-dev' and 'live-env'.
2018-07-11 21:05:30    INFO: 'tmp-dev.elasticbeanstalk.com' now points to 'awseb-e-j-
AWSEBLoa-M7U21VXNLWHN-487871449.us-west-2.elb.amazonaws.com'.
2018-07-11 21:05:30    INFO: Completed swapping CNAMEs for environments 'tmp-dev' and
 'live-env'.
```

下列的範例會將環境 tmp-dev 與環境 live-env 交換，但並未提示您針對任何設定來輸入或選擇值。

```
$ eb swap tmp-dev --destination_name live-env
2018-07-11 21:18:12    INFO: swapEnvironmentCNAMEs is starting.
2018-07-11 21:18:13    INFO: Swapping CNAMEs for environments 'tmp-dev' and 'live-env'.
2018-07-11 21:18:17    INFO: 'tmp-dev.elasticbeanstalk.com' now points to 'awseb-e-j-
AWSEBLoa-M7U21VXNLWHN-487871449.us-west-2.elb.amazonaws.com'.
2018-07-11 21:18:17    INFO: Completed swapping CNAMEs for environments 'tmp-dev' and
 'live-env'.
```

eb tags

描述

新增、刪除、更新和列出 Elastic Beanstalk 資源的標籤。

如需 Elastic Beanstalk 中資源標記的詳細資訊，請參閱標記 Elastic Beanstalk 應用程式資源。

語法

eb tags [***environment-name***] [--resource ***ARN***] -l | --list

eb tags [***environment-name***] [--resource ***ARN***] -a | --add ***key1***=***value1***[,***key2***=***value2*** ...]

eb tags [*environment-name*] [--resource *ARN*] -u | --update *key1*=*value1*[,*key2*=*value2* ...]

eb tags [*environment-name*] [--resource *ARN*] -d | --delete *key1*[,*key2* ...]

您可將 --add、--update 和 --delete 子命令選項結合為單一命令，至少需要其中一個選項。這三個子命令選項的任何一個均無法與 --list 結合使用。

如果沒有任何額外的引數，所有這些命令會列出或修改目前目錄的應用程式中預設環境的標籤。使用 *environment-name* 引數，此命令會列出或修改該環境的標籤。使用 --resource 選項，此命令會列出或修改任何 Elastic Beanstalk 資源 - 應用程式、環境、應用程式版本、已儲存的組態，或自訂平台版本的標籤。透過其 Amazon Resource Name (ARN) 指定資源。

選項

這些選項均非必要。若您不帶任何選項執行 eb create，將提示您輸入或選取每個設定的值。

名稱	描述
-l 或 --list	列出目前套用至資源的所有標籤。
-a *key1*=*value1*[,*key2*=*value2* 或 --add *key1*=*value1*[,*key2*=*val*	將新標籤套用至資源。將標籤指定為以逗號分隔的 key=value 對清單。您無法指定現有標籤的金鑰。 有效值：請參閱 標記 資源。
-u *key1*=*value1*[,*key2*=*value2* 或 --updat e *key1*=*value1*[,*key2*=*value2* .	更新現有資源標籤的值。將標籤指定為以逗號分隔的 key=value 對清單。您必須指定現有標籤的金鑰。 有效值：請參閱 標記 資源。
-d *key1*[,*key2* ...] 或 --delete *key1*[,*key2* ...]	刪除現有的資源標籤。將標籤指定為以逗號分隔的金鑰清單。您必須指定現有標籤的金鑰。 有效值：請參閱 標記 資源。

名稱	描述
`-r` *##* 或 `--region` *##*	您的資源所在的 AWS 區域。 預設值：設定的預設區域。 如需可以為此選項指定的值之清單，請參閱《AWS 一般參考》中的AWS Elastic Beanstalk 端點與配額。
`--resource` *ARN*	命令修改或列出標籤之資源的 ARN。如果未指定，此命令是指目錄中的應用程式的 (預設或指定) 環境。 有效值：請參閱 標記 資源 中您感興趣的特定資源的其中一個子主題。這些主題會說明資源的 ARN 的建構方式，並說明如何取得存在可供您的應用程式或帳戶使用的此資源 ARN 的清單。

輸出

`--list` 子命令選項會顯示資源標籤清單。輸出會顯示 Elastic Beanstalk 預設套用的標籤和自訂標籤。

```
$ eb tags --list
Showing tags for environment 'MyApp-env':

Key                                 Value

Name                                MyApp-env
elasticbeanstalk:environment-id     e-63cmxwjaut
elasticbeanstalk:environment-name   MyApp-env
mytag                               tagvalue
tag2                                2nd value
```

`--add`、`--update` 和 `--delete` 子命令選項成功執行時，不會有任何輸出。您可以新增 `--verbose` 選項，以查看命令活動的詳細輸出。

```
$ eb tags --verbose --update "mytag=tag value"
Updated Tags:

Key                                 Value
```

```
mytag                                tag value
```

範例

以下命令會成功將具有索引鍵 tag1 和值 value1 的標籤新增到應用程式的預設環境，並同時刪除標籤 tag2。

```
$ eb tags --add tag1=value1 --delete tag2
```

以下命令會成功新增標籤到應用程式內已儲存的組態。

```
$ eb tags --add tag1=value1 \
      --resource "arn:aws:elasticbeanstalk:us-east-2:my-account-
id:configurationtemplate/my-app/my-template"
```

下列命令失敗，因其嘗試更新不存在的標籤。

```
$ eb tags --update tag3=newval
ERROR: Tags with the following keys can't be updated because they don't exist:

  tag3
```

下列命令失敗，因其嘗試更新並刪除相同金鑰。

```
$ eb tags --update mytag=newval --delete mytag
ERROR: A tag with the key 'mytag' is specified for both '--delete' and '--update'. Each
 tag can be either deleted or updated in a single operation.
```

eb terminate

描述

終止執行環境，讓您無須支付未使用的 AWS 資源費用。

使用 --all 選項來透過 eb init 刪除目前目錄初始化的應用程式。此命令會終止在應用程式中的所有環境。此命令也會終止應用程式的應用程式版本以及已儲存的組態，然後刪除應用程式。

若根目錄包含指定自訂平台的 platform.yaml 檔案，則此命令會終止執行中的自訂環境。

> **Note**
>
> 您稍後可隨時運用相同版本啟動新的環境。

如果您希望保留環境中的資料，請在終止環境之前，將資料庫刪除政策設定為 `Retain`。這可使資料庫在 Elastic Beanstalk 之外運作。在這項操作之後，任何 Elastic Beanstalk 環境都必須以外部資料庫的形式連接它。如果您想要在不保持資料庫運作的情況下備份資料，請將刪除政策設定為在終止環境之前建立資料庫的快照。如需詳細資訊，請參閱本指南的「設定環境」一章中的 資料庫生命週期。

> **Important**
>
> 如果您終止環境，您也必須刪除您建立的任何 CNAME 映射，因為其他客戶可能重複使用可用的主機名稱。請務必刪除指向終止環境的 DNS 記錄，以防止懸置 DNS 項目。懸置的 DNS 項目可能會降低您網域的網際網路流量的安全性，使其暴露在易於攻擊的弱點中。另外，它還可能存在其他風險。
>
> 如需詳細資訊，請參閱 Amazon Route 53 開發人員指南中的在 Route 53 上防止懸置委派記錄。您也可以在AWS安全部落格中針對適用於 Amazon CloudFront 請求的強化網域保護來進一步了解懸置 DNS 項目的資訊。

語法

eb terminate

eb terminate ***environment-name***

選項

名稱	描述
`--all`	終止應用程式中所有環境、應用程式的應用程式版本以及已儲存的組態，然後刪除應用程式。
`--force`	不經提示確認即終止環境。
`--ignore-links`	即使某環境具備與其連結的相依環境仍將其終止。請參閱編寫環境。

名稱	描述
--timeout	命令逾時前的分鐘數。

輸出

若成功，本命令會回傳 terminate 操作的狀態。

範例

下列範例請求會終止環境 tmp-dev。

```
$ eb terminate
The environment "tmp-dev" and all associated instances will be terminated.
To confirm, type the environment name: tmp-dev
2018-07-11 21:05:25    INFO: terminateEnvironment is starting.
2018-07-11 21:05:40    INFO: Deleted CloudWatch alarm named: awseb-e-2cpfjbra9a-stack-
AWSEBCloudwatchAlarmHigh-16V08YOF2KQ7U
2018-07-11 21:05:41    INFO: Deleted CloudWatch alarm named: awseb-e-2cpfjbra9a-stack-
AWSEBCloudwatchAlarmLow-6ZAWH9F20P7C
2018-07-11 21:06:42    INFO: Deleted Auto Scaling group policy named:
 arn:aws:autoscaling:us-east-2:11122223333:scalingPolicy:5d7d3e6b-
d59b-47c5-b102-3e11fe3047be:autoScalingGroupName/awseb-e-2cpfjbra9a-stack-
AWSEBAutoScalingGroup-7AXY7U13ZQ6E:policyName/awseb-e-2cpfjbra9a-stack-AWSEBAutoSca
lingScaleUpPolicy-1876U27JEC34J
2018-07-11 21:06:43    INFO: Deleted Auto Scaling group policy named:
 arn:aws:autoscaling:us-east-2:11122223333:scalingPolicy:29c6e7c7-7ac8-46fc-91f5-
cfabb65b985b:autoScalingGroupName/awseb-e-2cpfjbra9a-stack-
AWSEBAutoScalingGroup-7AXY7U13ZQ6E:policyName/awseb-e-2cpfjbra9a-stack-AWSEBAutoSca
lingScaleDownPolicy-SL4LHODMOMU
2018-07-11 21:06:48    INFO: Waiting for EC2 instances to terminate. This may take a
 few minutes.
2018-07-11 21:08:55    INFO: Deleted Auto Scaling group named: awseb-e-2cpfjbra9a-
stack-AWSEBAutoScalingGroup-7AXY7U13ZQ6E
2018-07-11 21:09:10    INFO: Deleted security group named: awseb-e-2cpfjbra9a-stack-
AWSEBSecurityGroup-XT4YYGFL7I99
2018-07-11 21:09:40    INFO: Deleted load balancer named: awseb-e-2-AWSEBLoa-
AK6RRYFQVV3S
2018-07-11 21:09:42    INFO: Deleting SNS topic for environment tmp-dev.
2018-07-11 21:09:52    INFO: terminateEnvironment completed successfully.
```

eb upgrade

描述

將您環境的平台升級至現正執行環境的平台之最新版本。

若根目錄包含指定自訂平台的 `platform.yaml` 檔案，此命令會將環境升級至現正執行環境的自訂平台之最新版本。

語法

eb upgrade

eb upgrade ***environment-name***

選項

名稱	描述
`--force`	升級程序前您無須確認環境名稱，即可進行升級。
`--noroll`	更新所有執行個體，無須透過滾動更新即可於升級期間保留部分執行個體的服務。
常用選項	

輸出

本命令會顯示變更的概觀，並提示您輸入環境名稱以確認升級。若成功，您的環境會進行更新，並以最新的平台版本啟動。

範例

下列範例將指定環境目前的平台版本升級至最新的平台版本。

```
$ eb upgrade
Current platform: 64bit Amazon Linux 2014.09 v1.0.9 running Python 2.7
Latest platform:  64bit Amazon Linux 2014.09 v1.2.0 running Python 2.7

WARNING: This operation replaces your instances with minimal or zero downtime. You may
 cancel the upgrade after it has started by typing "eb abort".
```

```
You can also change your platform version by typing "eb clone" and then "eb swap".

To continue, type the environment name:
```

eb use

描述

將指定環境設定為預設環境。

使用 Git 時，eb use 會設定目前分支的預設環境。在您欲將其部署至 Elastic Beanstalk 的各個分支中，執行此命令一次。

語法

eb use ***environment-name***

選項

名稱	描述
`--source codecommit/` *repository-name*`/`*branch-name*	CodeCommit 儲存庫和分支。請參閱搭配 AWS CodeCommit 使用 EB CLI。
`-r` *region* `--region` *region*	變更您建立環境的區域。
常用選項	

常用選項

下列選項可用於所有 EB CLI 命令。

名稱	描述
`--debug`	列印除錯資訊。

名稱	描述
-h, --help	顯示協助訊息。 類型：字串 預設：無
--no-verify-ssl	略過 SSL 憑證驗證。若您透過代理使用 CLI 出現問題，請使用此選項。
--profile	使用您 AWS 憑證檔案的特定描述檔。
--quiet	抑制該命令的所有輸出。
--region	使用指定的區域。
-v, --verbose	顯示詳細資訊。

EB CLI 2.6 (已淘汰)

此 EB CLI 版本及其文件已取代為版本 3 (在本節中，EB CLI 3 代表 EB CLI 版本 3 及更新版本)。如需新版本的資訊，請參閱 使用 Elastic Beanstalk 命令列界面 (EB CLI)。

您應該遷移至最新版本的 EB CLI 3。其可以管理您使用 EB CLI 2.6 或舊版 EB CLI 來啟動的環境。

與 EB CLI 版本 3 的差異

EB 為適用 Elastic Beanstalk 的命令列界面 (CLI) 工具，您能夠藉此快速並更輕鬆地部署應用程式。EB 的最新版本由 Elastic Beanstalk 於 EB CLI 3 推出。EB CLI 會從使用 EB 建立的環境自動擷取設定 (若該環境正在執行)。請注意，EB CLI 3 不會如同先前版本將選項設定存放於本機。

EB CLI 具備命令 eb create、eb deploy、eb open、eb console、eb scale、eb setenv、eb config、eb terminate、eb clone、eb list、eb use、eb printenv 和 eb ssh。在 EB CLI 3.1 或更新版本中，您亦可使用 eb swap 命令。僅 EB CLI 3.2 可以使用 eb abort、eb platform 和 eb upgrade 命令。除了這些新的命令，EB CLI 3 的下列命令與 EB CLI 2.6 命令不同：

- eb init - 使用 eb init 於現有專案目錄建立 `.elasticbeanstalk` 目錄並為專案建立新的 Elastic Beanstalk 應用程式。EB CLI 3 和更新版本不同於之前版本，不會提示您建立環境。
- eb start – EB CLI 3 不包含命令 eb start。使用 eb create 來建立環境。

- eb stop – EB CLI 3 不包含命令 eb stop。使用 eb terminate 來完全終止並清除環境。
- eb push 和 **`git aws.push`** – EB CLI 3 不包含命令 eb push 或 `git aws.push`。使用 eb deploy 來更新您的應用程式程式碼。
- eb update – EB CLI 3 不包含命令 eb update。使用 eb config 來更新環境。
- eb branch – EB CLI 3 不包含命令 eb branch。

如需有關使用 EB CLI 3 命令來建立並管理應用程式的詳細資訊，請參閱 EB CLI 命令參考。如需使用 EB CLI 3 來部署範例應用程式的演練，請參閱 使用 EB CLI 管理 Elastic Beanstalk 環境。

遷移至 EB CLI 3 和 CodeCommit

Elastic Beanstalk 不僅淘汰了 EB CLI 2.6，也移除了一些 2.6 功能。2.6 的最大變更是 EB CLI 不再原生支援程式碼遞增式更新 (eb push、`git aws.push`) 或分支 (eb branch)。本節說明如何自 EB CLI 2.6 遷移至 EB CLI 最新版本以及如何使用 CodeCommit 做為您的程式碼儲存庫。

若您尚未於 CodeCommit 建立程式碼儲存庫，請依遷移至 CodeCommit 一文所述建立。

您安裝並設定 EB CLI 後，有兩個方式將應用程式與 CodeCommit 儲存庫 (包含特定分支) 建立關聯。

- 執行 eb init，如下列範例所示，其中 *myRepo* 為您 CodeCommit 儲存庫的名稱，而 *myBranch* 則是 CodeCommit 的分支。

```
eb init --source codecommit/myRepo/myBranch
```

- 執行 eb deploy，如下列範例所示，其中 *myRepo* 為您 CodeCommit 儲存庫的名稱，而 *myBranch* 則是 CodeCommit 的分支。

```
eb deploy --source codecommit/myRepo/myBranch
```

如需進一步了解如何於 Elastic Beanstalk 環境部署程式碼遞增式更新而不用重新上傳整個專案，請參閱搭配 AWS CodeCommit 使用 EB CLI。

Elastic Beanstalk API 命令列界面 (已淘汰)

本工具 Elastic Beanstalk API 命令列界面 (API CLI) 的功能已由 AWS CLI 取代，藉此為所有 AWS 服務提供等同 API 的命令。請參閱「AWS Command Line Interface 使用者指南」來開始使用 AWS CLI。您也可以嘗試使用 EB CLI，體驗簡化的高層級命令列。

轉換 Elastic Beanstalk API CLI 指令碼

將舊的 EB API CLI 指令碼轉換為使用 AWS CLI 或Tools for Windows PowerShell，以存取最新的 Elastic Beanstalk API。下表列出 Elastic Beanstalk 以 API 為基礎的 CLI 命令及其在 AWS CLI 和 Tools for Windows PowerShell中的同等命令。

Elastic Beanstalk API CLI	AWS CLI	AWS Tools for Windows PowerShell
`elastic-beanstalk-check-dns-availability`	`check-dns-availability`	`Get-EBDNSAvailability`
`elastic-beanstalk-create-application`	`create-application`	`New-EBApplication`
`elastic-beanstalk-create-application-version`	`create-application-version`	`New-EBApplicationVersion`
`elastic-beanstalk-create-configuration-template`	`create-configuration-template`	`New-EBConfigurationTemplate`
`elastic-beanstalk-create-environment`	`create-environment`	`New-EBEnvironment`
`elastic-beanstalk-create-storage-location`	`create-storage-location`	`New-EBStorageLocation`
`elastic-beanstalk-delete-application`	`delete-application`	`Remove-EBApplication`

Elastic Beanstalk API CLI	AWS CLI	AWS Tools for Windows PowerShell
elastic-beanstalk-delete-application-version	delete-application-version	Remove-EBApplicationVersion
elastic-beanstalk-delete-configuration-template	delete-configuration-template	Remove-EBConfigurationTemplate
elastic-beanstalk-delete-environment-configuration	delete-environment-configuration	Remove-EBEnvironmentConfiguration
elastic-beanstalk-describe-application-versions	describe-application-versions	Get-EBApplicationVersion
elastic-beanstalk-describe-applications	describe-applications	Get-EBApplication
elastic-beanstalk-describe-configuration-options	describe-configuration-options	Get-EBConfigurationOption
elastic-beanstalk-describe-configuration-settings	describe-configuration-settings	Get-EBConfigurationSetting

Elastic Beanstalk API CLI	AWS CLI	AWS Tools for Windows PowerShell
elastic-beanstalk-describe-environment-resources	describe-environment-resources	Get-EBEnvironmentResource
elastic-beanstalk-describe-environments	describe-environments	Get-EBEnvironment
elastic-beanstalk-describe-events	describe-events	Get-EBEvent
elastic-beanstalk-list-available-solution-stacks	list-available-solution-stacks	Get-EBAvailableSolutionStack
elastic-beanstalk-rebuild-environment	rebuild-environment	Start-EBEnvironmentRebuild
elastic-beanstalk-request-environment-info	request-environment-info	Request-EBEnvironmentInfo
elastic-beanstalk-restart-app-server	restart-app-server	Restart-EBAppServer
elastic-beanstalk-retrieve-environment-info	retrieve-environment-info	Get-EBEnvironmentInfo
elastic-beanstalk-swap-environment-cnames	swap-environment-cnames	Set-EBEnvironmentCNAME

Elastic Beanstalk API CLI	AWS CLI	AWS Tools for Windows PowerShell
elastic-beanstalk-terminate-environment	terminate-environment	Stop-EBEnvironment
elastic-beanstalk-update-application	update-application	Update-EBApplication
elastic-beanstalk-update-application-version	update-application-version	Update-EBApplicationVersion
elastic-beanstalk-update-configuration-template	update-configuration-template	Update-EBConfigurationTemplate
elastic-beanstalk-update-environment	update-environment	Update-EBEnvironment
elastic-beanstalk-validate-configuration-settings	validate-configuration-settings	Test-EBConfigurationSetting

AWS Elastic Beanstalk 安全

雲端安全是 AWS 最重視的一環。身為 AWS 客戶的您，將能從資料中心和網路架構的建置中獲益，以滿足組織最為敏感的安全要求。

安全是 AWS 與您共同肩負的責任。共同責任模型將此描述為「雲端本身的安全」和「雲端內部的安全」。

雲端安全性 – AWS 負責保護執行 AWS 雲端提供之所有服務的基礎設施，並提供您可安全使用的服務。安全責任在 AWS 為最優先，而且 AWS 合規計劃會由第三方稽核員定期測試和驗證安全有效性。因與 Elastic Beanstalk 有關，請檢閱 AWS 保證計劃範圍內的 AWS 服務以取得相關資訊。

雲端內部的安全 – 您的責任取決於您使用的 AWS 服務，其他因素則包括資料的敏感性、組織的要求以及適用的法律規章。本文件旨在協助您了解如何在使用 Elastic Beanstalk 時套用共同責任模型。

使用下列安全主題以進一步了解 Elastic Beanstalk 負責的安全任務，以及使用 Elastic Beanstalk 時應考慮的安全組態，以符合您的安全與合規目標。

主題

Elastic Beanstalk 的資料保護

AWS 共同的責任模型適用於 AWS Elastic Beanstalk 中的資料保護。如此模型所述，AWS 負責保護執行所有 AWS 雲端 的全球基礎設施。您必須負責維護在此基礎設施上託管之內容的控制權。您也必須負責您所使用的 AWS 服務 安全性設定和管理工作。如需有關資料隱私權的詳細資訊，請參閱資料隱私權常見問答集。如需有關歐洲資料保護的相關資訊，請參閱 AWS 安全性部落格上的 AWS 共同的責任模型和 GDPR 部落格文章。

基於資料保護目的，建議您使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 保護 AWS 帳戶 憑證，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 資源通訊。我們需要 TLS 1.2 並建議使用 TLS 1.3。
- 使用 AWS CloudTrail 設定 API 和使用者活動記錄。
- 使用 AWS 加密解決方案，以及 AWS 服務 內的所有預設安全控制項。
- 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 AWS 時，需要 FIPS 140-2 驗證的加密模組，請使用 FIPS 端點。如需有關 FIPS 和 FIPS 端點的詳細資訊，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 Name (名稱) 欄位。這包括當您使用 Elastic Beanstalk 或使用主控台、API、AWS CLI 或 AWS 軟體開發套件的其他 AWS 服務 時。您在標籤或自由格式文字欄位中輸入的任何資料都可能用於計費或診斷日誌。如果您提供外部伺服器的 URL，我們強烈建議請勿在驗證您對該伺服器請求的 URL 中包含憑證資訊。

如需其他 Elastic Beanstalk 安全主題，請參閱 AWS Elastic Beanstalk 安全。

主題

- 使用加密保護資料
- 網際網路流量隱私權

使用加密保護資料

Elastic Beanstalk 會將各種物件存放在 Amazon Simple Storage Service (Amazon S3) 儲存貯體中，這是為建立環境所在的每個 AWS 區域建立。如需詳細資訊，請參閱the section called “Amazon S3”。

您提供一些已存放的物件，並將它們傳送至 Elastic Beanstalk，例如應用程式版本和原始碼套件。Elastic Beanstalk 會產生其他物件，例如日誌檔案。除了 Elastic Beanstalk 存放的資料，您的應用程式可以在操作期間傳輸及/或存放資料。

資料保護是指保護傳輸中的資料 (當其往返於 Elastic Beanstalk 時)，以及靜態資料 (當其存放在 AWS 資料中心時)。

傳輸中加密

您有兩種方式可保護正在傳輸的資料：使用 Secure Sockets Layer (SSL) 加密連線，或使用用戶端加密 (物件在傳送前即已加密)。這兩種方法都能保護您的應用程式資料。若要保護連線，請在您的應用程式、其開發人員和管理員及其最終使用者傳送或接收任何物件時，使用 SSL 加密連線。如需加密傳入傳出應用程式的 Web 流量詳細資訊，請參閱the section called “HTTPS”。

用戶端加密不是在您上傳的應用程式版本和原始碼套件中保護原始碼的有效方法。Elastic Beanstalk 需要存取這些物件，因此無法加密。因此，請務必保護開發或部署環境與 Elastic Beanstalk 間的連線安全。

靜態加密

若要保護應用程式的靜態資料，請了解應用程式所用儲存服務中的資料保護。例如，請參閱《Amazon RDS 使用者指南》中的 Amazon RDS 的資料保護、《Amazon Simple Storage Service 使用者指南》中的 Amazon S3 的資料保護，或《Amazon Elastic File System 使用者指南》中的在 EFS 中加密資料和中繼資料。

Elastic Beanstalk 不會開啟其建立的 Amazon S3 儲存貯體的預設加密功能。這表示，根據預設，物件會以未加密的狀態存放在儲存貯體中 (僅供獲得儲存貯體讀取權限的使用者存取)。如果您的應用程式需要靜態加密，您可以將帳戶的儲存貯體設定為預設加密。如需詳細資訊，請參閱《Amazon Simple Storage Service 使用者指南》中的 S3 儲存貯體的Amazon S3 預設加密。

如需關於資料保護的詳細資訊，請參閱 AWS 安全部落格上的 AWS 共同責任模型和歐盟《一般資料保護規範》(GDPR) 部落格文章。

如需其他 Elastic Beanstalk 安全主題，請參閱 AWS Elastic Beanstalk 安全。

網際網路流量隱私權

您可以使用 Amazon Virtual Private Cloud (Amazon VPC) 在 Elastic Beanstalk 應用程式的資源之間建立邊界，並控制這些資源、內部部署網路和網際網路之間的流量。如需詳細資訊，請參閱the section called “Amazon VPC”。

如需 Amazon VPC 安全性的詳細資訊，請參閱《Amazon VPC 使用者指南》中的安全性。

如需關於資料保護的詳細資訊，請參閱 AWS 安全部落格上的 AWS 共同責任模型和歐盟《一般資料保護規範》(GDPR) 部落格文章。

如需其他 Elastic Beanstalk 安全主題，請參閱 AWS Elastic Beanstalk 安全。

Elastic Beanstalk 的 Identity and Access Management

AWS Identity and Access Management (IAM) 是一種 AWS 服務，讓管理員能夠安全地控制對 AWS 資源的存取。IAM 管理員會控制誰可經身份驗證 (已登入) 和授權 (具有許可) 來使用 AWS Elastic Beanstalk 資源。IAM 是一種您可以免費使用的 AWS 服務。

如需有關使用 IAM 的詳細資訊，請參閱搭配 AWS Identity and Access Management 使用 Elastic Beanstalk。

如需其他 Elastic Beanstalk 安全主題，請參閱 AWS Elastic Beanstalk 安全。

AWS 受管理的政策 AWS Elastic Beanstalk

受 AWS 管理的策略是由建立和管理的獨立策略 AWS。 AWS 受管理的策略旨在為許多常見使用案例提供權限，以便您可以開始將權限指派給使用者、群組和角色。

請記住， AWS 受管理的政策可能不會為您的特定使用案例授與最低權限權限，因為這些權限可供所有 AWS 客戶使用。我們建議您定義使用案例專屬的客戶管理政策，以便進一步減少許可。

您無法變更受 AWS 管理策略中定義的權限。如果 AWS 更新 AWS 受管理原則中定義的權限，則此更新會影響附加原則的所有主體識別 (使用者、群組和角色)。 AWS 當新的啟動或新 AWS 服務 的 API 操作可用於現有服務時，最有可能更新 AWS 受管理策略。

如需詳細資訊，請參閱《IAM 使用者指南》中的 AWS 受管政策。

受管理政策的 Elastic Beanstalk 更新 AWS

檢視有關自 2021 年 3 月 1 日起 Elastic Beanstalk AWS 受管政策的更新詳細資料。

若要查看特定受管原則的 JSON 來源，請參閱受AWS 管政策參考指南。

變更	描述	日期
下列政策已更新：	這些政策已更新，以允許 Elastic Beanstalk 在建立或更新 AWS CloudFormation 堆疊或變更集時新增或移除標籤。	2024 年 4 月 30 日

變更	描述	日期
• AWSElasticBeanstalkInternalMaintenanceRolePolicy • AWSElasticBeanstalkMaintenance • AWSElasticBeanstalkManagedUpdatesInternalServiceRolePolicy • AWSElasticBeanstalkManagedUpdatesServiceRolePolicy • AWSElasticBeanstalkRoleCore	如需有關 AWSElasticBeanstalkManagedUpdatesServiceRolePolicy 的詳細資訊，請參閱 Elastic Beanstalk 的服務連結角色許可。 如需有關 AWSElasticBeanstalkRoleCore 的詳細資訊，請參閱 與其他服務整合的政策。	

變更	描述	日期
AWSElasticBeanstalkService-更新了現有策略	此政策已更新，讓 Elastic Beanstalk 能在為 Elastic Load Balancing、Auto Scaling 群組 (ASG) 和 Amazon ECS 建立資源時，標記資源。 **Note** 此政策先前已被 `AWSElasticBeanstalkManagedUpdatesCustomerRolePolicy` 取代。雖然此政策不再供連接至新的 IAM 使用者、群組或角色，但仍可連接至先前的現有項目。 如需詳細資訊，請參閱受管服務角色政策。	2023 年 5 月 10 日
AWSElasticBeanstalkMulticontainerDocker-更新了現有策略	已更新此政策，以便讓 Elastic Beanstalk 能在 Amazon ECS 建立時標記資源。 如需詳細資訊，請參閱 管理 Elastic Beanstalk 執行個體描述檔。	2023 年 3 月 23 日

變更	描述	日期
AWSElasticBeanstalkRoleECS-更新了現有策略	已更新此政策，以便讓 Elastic Beanstalk 能在 Amazon ECS 建立時標記資源。 如需詳細資訊，請參閱 與其他服務整合的政策。	2023 年 3 月 23 日
AdministratorAccess-AWSElasticBeanstalk- 更新了現有策略	已更新此政策，以便讓 Elastic Beanstalk 能在 Amazon ECS 建立時標記資源。 如需詳細資訊，請參閱 管理 Elastic Beanstalk 使用者政策。	2023 年 3 月 23 日
AWSElasticBeanstalkManagedUpdatesServiceRolePolicy -更新了現有策略	已更新此政策，以便讓 Elastic Beanstalk 能夠在建立 Amazon ECS 資源時新增標籤。 如需詳細資訊，請參閱 Elastic Beanstalk 的服務連結角色許可。	2023 年 3 月 23 日
AWSElasticBeanstalkManagedUpdatesCustomerRolePolicy-更新了現有策略	已更新此政策，以便讓 Elastic Beanstalk 能夠在建立 Amazon ECS 資源時新增標籤。 如需詳細資訊，請參閱受管服務角色政策。	2023 年 3 月 23 日
AWSElasticBeanstalkManagedUpdatesServiceRolePolicy-更新了現有策略	已更新此政策，以便讓 Elastic Beanstalk 能夠在建立 Auto Scaling 群組時新增標籤。 如需詳細資訊，請參閱 受管更新服務連結角色。	2023 年 1 月 27 日

變更	描述	日期
AWSElasticBeanstalkManagedUpdatesCustomerRolePolicy-更新了現有策略	已更新此政策，以允許 Elastic Beanstalk 在建立 Auto Scaling 群組 (ASG) 時新增標籤。 如需詳細資訊，請參閱受管服務角色政策。	2023 年 1 月 23 日
AWSElasticBeanstalkManagedUpdatesCustomerRolePolicy-更新了現有策略	已更新此政策，以允許 Elastic Beanstalk 在建立 Elastic Load Balancing (ELB) 時新增標籤。 如需詳細資訊，請參閱受管服務角色政策。	2022 年 12 月 21 日
AWSElasticBeanstalkManagedUpdatesServiceRolePolicy-更新了現有策略	已將許可新增至此政策，以便讓 Elastic Beanstalk 能夠在受管更新期間執行以下操作： • 建立和刪除啟動範本和範本版本。 • 使用啟動範本啟動 Amazon EC2 執行個體。 • 如果存在 Amazon RDS，則會擷取可用資料庫引擎的清單及已佈建 RDS 執行個體的資訊。 如需詳細資訊，請參閱 受管更新服務連結角色。	2022 年 8 月 23 日

變更	描述	日期
AWSElasticBeanstalkReadOnlyAccess— 已棄用 GovCloud (美國) AWS 區域	此政策已被 `AWSElasticBeanstalkReadOnly` 取代。 此政策將在 GovCloud （美國） AWS 區域逐步淘汰。 此政策逐步淘汰時，在 2021 年 6 月 17 日後將無法再連接至新的 IAM 使用者、群組或角色。 如需詳細資訊，請參閱使用者政策。	2021 年 6 月 17 日
AWSElasticBeanstalkManagedUpdatesCustomerRolePolicy-更新了現有策略	此原則已更新為允許 Elastic Beanstalk 讀取 EC2 可用區域的屬性。可讓 Elastic Beanstalk 更有效地驗證您在可用區域中的執行個體類型選擇。 如需詳細資訊，請參閱受管服務角色政策。	2021 年 6 月 16 日

變更	描述	日期
AWSElasticBeanstalkFullAccess— 已棄用 GovCloud (美國) AWS 區域	此政策已被 `AdministratorAccess-AWSElasticBeanstalk` 取代。 此政策將在 GovCloud （美國） AWS 區域逐步淘汰。 此政策逐步淘汰時，在 2021 年 6 月 10 日後將無法再連接至新的 IAM 使用者、群組或角色。 如需詳細資訊，請參閱使用者政策。	2021 年 6 月 10 日
下列受管理政策已在中國所有國家中棄 AWS 區域用： • AWSElasticBeanstalkFullAccess • AWSElasticBeanstalkReadOnlyAccess	`AWSElasticBeanstalkFullAccess` 政策已被 `AdministratorAccess-AWSElasticBeanstalk` 取代。 `AWSElasticBeanstalkReadOnlyAccess` 政策已被 `AWSElasticBeanstalkReadOnly` 取代。 這些政策在所有中國 AWS 區域都被淘汰。 在 2021 年 6 月 3 日後，這些政策將無法再連接至新的 IAM 使用者、群組或角色。 如需詳細資訊，請參閱使用者政策。	2021 年 6 月 3 日

變更	描述	日期
AWSElasticBeanstalkService— 已棄用	此政策已被 `AWSElasticBeanstalkManagedUpdatesCustomerRolePolicy` 取代。 此政策正在逐步淘汰，且將無法再用於連接至新的 IAM 使用者、群組或角色。 如需詳細資訊，請參閱受管服務角色政策。	2021 年 6 月 - 2022 年 1 月
除了中國和 GovCloud (美國) 以外，下列受管理政策已在所有 AWS 區域政策中被棄用： • AWSElasticBeanstalkFullAccess • AWSElasticBeanstalkReadOnlyAccess	`AWSElasticBeanstalkFullAccess` 政策已被 `AdministratorAccess-AWSElasticBeanstalk` 取代。 `AWSElasticBeanstalkReadOnlyAccess` 政策已被 `AWSElasticBeanstalkReadOnly` 取代。 這些政策在所有國家都被淘汰，除了中國和 GovCloud（美國）。 AWS 區域 在 2021 年 4 月 16 日後，這些政策將無法再連接至新的 IAM 使用者、群組或角色。 如需詳細資訊，請參閱使用者政策。	2021 年 4 月 16 日

變更	描述	日期
下列受管政策已更新： • AdministratorAccess-AWSElasticBeanstalk • AWSElasticBeanstalkManagedUpdatesCustomerRolePolicy	這兩項政策現在都支援中國的 PassRole 許可 AWS 區域。 如需 `AdministratorAccess-AWSElasticBeanstalk` 的詳細資訊，請參閱使用者政策。 如需有關 `AWSElasticBeanstalkManagedUpdatesCustomerRolePolicy` 的詳細資訊，請參閱受管服務角色政策。	2021 年 3 月 9 日
AWSElasticBeanstalkManagedUpdatesCustomerRolePolicy – 新政策	Elastic Beanstalk 新增了新的政策來取代 `AWSElasticBeanstalkService` 受管政策。 這個新的受管政策會套用更嚴格的許可集來改善資源的安全性。 如需詳細資訊，請參閱受管服務角色政策。	2021 年 3 月 3 日
Elastic Beanstalk 開始追蹤變更	Elastic Beanstalk 開始追蹤 AWS 受管理政策的變更。	2021 年 3 月 1 日

Elastic Beanstalk 中的記錄和監控

監控對於維護 AWS Elastic Beanstalk 和您 AWS 解決方案的可靠性、可用性和效能十分重要。您應該從 AWS 解決方案各個部分收集監控資料，以便在發生多點失敗時，可更輕鬆地偵錯。AWS 提供多種工具，讓您監控 Elastic Beanstalk 資源及回應潛在的事件。

如需監控的詳細資訊，請參閱監控環境。

如需其他 Elastic Beanstalk 安全性主題，請參閱AWS Elastic Beanstalk 安全。

增強型運作狀態報告

您可於環境啟用增強型運作狀態報告，讓 Elastic Beanstalk 收集您環境資源的其他資訊。Elastic Beanstalk 會分析資訊，以提供更全面的整體環境運作狀態，並協助辨識可能會導致應用程式無法運作的問題。如需更多詳細資訊，請參閱 增強型運作狀態報告與監控。

Amazon EC2 執行個體日誌

Elastic Beanstalk 環境的 Amazon EC2 執行個體會產生日誌，您可加以檢視針對應用程式或組態檔案進行問題疑難排解。由 Web 伺服器、應用程式伺服器、Elastic Beanstalk 平台指令碼和 AWS CloudFormation 建立的日誌會儲存在個別本機執行個體上。您可以使用環境管理主控台或 EB CLI 輕鬆擷取他們。您也可以將環境設定為即時將日誌串流至 Amazon CloudWatch Logs。如需詳細資訊，請參閱 在 Elastic Beanstalk 環境中檢視 Amazon EC2 執行個體的日誌。

環境通知

您可以將 Elastic Beanstalk 環境設定為使用 Amazon Simple Notification Service (Amazon SNS) 來通知您影響應用程式的重要事件。在建立環境期間或之後指定電子郵件地址，以在發生錯誤或在您環境的運作狀態改變時，接收由 AWS 傳送的電子郵件。如需詳細資訊，請參閱 使用 Amazon SNS 的 Elastic Beanstalk 環境通知。

Amazon CloudWatch 警示

您可以使用 CloudWatch 警示觀察單一指標一段指定的時間。如果指標超過指定的閾值，一個通知會傳送至 Amazon SNS 主題或 AWS Auto Scaling 政策。CloudWatch 警示不會因為它們處於特定狀態而叫用動作。當狀態變更且維持一段指定期間時，警示會改呼叫動作。如需詳細資訊，請參閱 搭配 Amazon CloudWatch 使用 Elastic Beanstalk。

AWS CloudTrail 日誌

CloudTrail 會提供由使用者、角色或 AWS 服務在 Elastic Beanstalk 中採取動作的記錄。您可以利用 CloudTrail 所收集的資訊來判斷向 Elastic Beanstalk 發出的請求，以及發出請求的 IP 地址、人員、時間和其他詳細資訊。如需詳細資訊，請參閱 使用 AWS CloudTrail 記錄 Elastic Beanstalk API 呼叫。

AWS X-Ray 除錯

X-Ray 是一項 AWS 服務，可蒐集您應用程式提供請求的資料，並使用此資料建構服務地圖，藉以辨識應用程式的問題與最佳化的機會。您可以使用 AWS Elastic Beanstalk 主控台或組態檔案，對您環境中的執行個體執行 X-Ray 常駐程式。如需詳細資訊，請參閱 設定 AWS X-Ray 除錯。

Elastic Beanstalk 的合規驗證

在多個 AWS 合規計劃中，由第三方稽核人員評估 AWS Elastic Beanstalk 的安全和合規。這些包括 SOC、PCI、FedRAMP、HIPAA 和其他。AWS 提供經常更新的服務清單 AWS，在特定合規計劃範圍內的 AWS 服務範圍合規計劃。

您可以使用 AWS Artifact 下載第三方稽核報告。如需詳細資訊，請參閱在 AWS Artifact 中下載報告。

如需 AWS 合規計劃的詳細資訊，請參閱 AWS 合規計劃。

使用 Elastic Beanstalk 時的合規責任取決於資料的敏感度、組織的合規目標，以及適用的法律和法規。如果您使用的 Elastic Beanstalk 服務必須符合 HIPAA、PCI 或 FedRAMP 等標準，AWS 會提供資源予以協助：

- 安全與合規快速入門指南 – 部署指南討論在 AWS 上部署以安全及合規為重心之基準環境的架構考量和步驟。
- HIPAA 安全與合規架構白皮書 – 本白皮書說明公司可如何運用 AWS 來建立 HIPAA 合規的應用程式。
- AWS 合規資源 – 可能適合您產業和位置的合規性手冊和指南集合。
- AWS Config – 此服務可評定資源組態與內部實務、業界準則和法規的合規狀態。
- AWS Security Hub – 全面檢視您 AWS 中的安全狀態，可助您檢查是否符合安全產業標準和最佳實務。

如需其他 Elastic Beanstalk 安全主題，請參閱 AWS Elastic Beanstalk 安全。

Elastic Beanstalk 的彈性

AWS 全球基礎設施是以 AWS 區域與可用區域為中心建置的。

AWS 區域提供多個分開且隔離的實際可用區域，並以低延遲、高輸送量和高度備援聯網功能相互連結。

透過可用區域，您所設計與操作的應用程式和資料庫，就能夠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、容錯能力和擴充能力，均較單一或多個資料中心的傳統基礎設施還高。

如需 AWS 區域與可用區域的詳細資訊，請參閱 AWS 全球基礎設施。

AWS Elastic Beanstalk 會代您管理及自動化 AWS 全球基礎設施的運用。使用 Elastic Beanstalk 時，您會得益於 AWS 提供的可用性和容錯機制。

如需其他 Elastic Beanstalk 安全主題，請參閱 AWS Elastic Beanstalk 安全。

Elastic Beanstalk 中的基礎設施安全

AWS Elastic Beanstalk 為受管服務，受到 Amazon Web Services：安全程序概觀白皮書所述的 AWS 全球網路安全程序所保護。

您可使用 AWS 發佈的 API 呼叫，透過網路存取 Elastic Beanstalk。用戶端必須支援 Transport Layer Security (TLS) 1.0 或更新版本。建議使用 TLS 1.2 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)。現代平台 (如 Java 7 和更新版本) 大多支援這些模式。

此外，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) 產生暫時性安全憑證來簽署請求。

如需其他 Elastic Beanstalk 安全主題，請參閱 AWS Elastic Beanstalk 安全。

Elastic Beanstalk 中的組態與漏洞分析

AWS 和我們的客戶共同承擔實現高階軟體元件安全性和合規性的責任。透過提供受管更新功能，AWS Elastic Beanstalk 協助您執行您這邊的共同責任模型。此功能會自動套用 Elastic Beanstalk 支援平台版本的修補程式和次要更新。

如需更多詳細資訊，請參閱Elastic Beanstalk 平台維護的共同責任模型。

如需其他 Elastic Beanstalk 安全主題，請參閱 AWS Elastic Beanstalk 安全。

Elastic Beanstalk 的安全最佳實務

在您開發與實作自己的安全政策時，可考慮使用 AWS Elastic Beanstalk 提供的多種安全功能。以下最佳實務為一般準則，並不代表完整的安全解決方案。因為這些最佳實務可能不適合或無法滿足您的環境，所以請將它們視為實用建議，不要當成指示。

如需其他 Elastic Beanstalk 安全性主題，請參閱AWS Elastic Beanstalk 安全。

預防性安全最佳實務

預防性安全控制會嘗試在事件發生前防止事件發生。

實作最低權限存取

Elastic Beanstalk 為執行個體描述檔、服務角色和 IAM 使用者提供 AWS Identity and Access Management (IAM) 受管政策。這些受管政策會指定讓環境和應用程式正確操作可能需要的所有許可。

您的應用程式可能不需要受管政策中的所有許可。您可以自訂它們，並只授予您環境執行個體、Elastic Beanstalk 服務和使用者執行任務所需的許可。這特別關係到不同使用者角色可能有不同許可需求的使用者政策。對降低錯誤或惡意意圖所引起的安全風險和影響而言，實作最低權限存取是相當重要的一環。

定期更新您的平台

Elastic Beanstalk 會定期發佈新的平台版本，以更新其所有平台。新的平台版本會提供作業系統、執行時間、應用程式伺服器和 Web 伺服器更新，以及 Elastic Beanstalk 元件的更新。其中許多平台更新包含重要的安全性修正。請確認您的 Elastic Beanstalk 環境在支援的平台版本上執行 (通常是您平台的最新版本)。如需詳細資訊，請參閱「更新您 Elastic Beanstalk 環境的平台版本」。

將環境平台保持在最新狀態的最簡單方法，是將環境設定為使用受管平台更新。

在環境執行個體上強制執行 IMDSv2

Elastic Beanstalk 環境中的 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體使用執行個體中繼資料服務 (IMDS)，這是一種執行個體上的元件，可安全地存取執行個體中繼資料。IMDS 支援兩種存取資料的方法：IMDSv1 和 IMDSv2。IMDSv2 會使用工作階段導向的請求，並減緩可能用來嘗試存取 IMDS 的幾種漏洞類型。如需 IMDSv2 優勢的詳細資訊，請參閱提高 EC2 執行個體中繼資料服務防禦深度的增強功能。

IMDSv2 更安全，因此，最好在您的執行個體上強制使用 IMDSv2。若要強制執行 IMDSv2，請確定應用程式的所有元件皆支援 IMDSv2，然後停用 IMDSv1。如需更多詳細資訊，請參閱 the section called “IMDS”。

偵測性安全最佳實務

偵測性安全控制會在安全違規發生後識別出它們。它們可協助您偵測潛在的安全威脅或事件。

實作監控

監控是維持 Elastic Beanstalk 解決方案之可靠性、安全性、可用性和效能的重要環節。AWS 會提供數種工具和服務，幫助您監控 AWS 服務。

以下是一些要監控的項目範例：

- 適用於 Elastic Beanstalk 的 Amazon CloudWatch 指標 – 針對關鍵的 Elastic Beanstalk 指標和應用程式的自訂指標設定警示。如需詳細資訊，請參閱「搭配 Amazon CloudWatch 使用 Elastic Beanstalk」。
- AWS CloudTrail 項目 – 追蹤可能影響可用性的動作，例如 `UpdateEnvironment` 或 `TerminateEnvironment`。如需詳細資訊，請參閱「使用 AWS CloudTrail 記錄 Elastic Beanstalk API 呼叫」。

啟用 AWS Config

AWS Config 提供在您的帳戶中 AWS 資源組態的詳細檢視。您可以了解資源如相關聯，可以取得組態變更歷程記錄，並且了解關係和組態隨時間產生的變化。

您可以使用 AWS Config 來定義評估資料合規性的資源組態規則。AWS Config 規則代表您 Elastic Beanstalk 資源的理想組態設定。如果資源違反規則並標示為不合規，AWS Config 可以使用 Amazon Simple Notification Service (Amazon SNS) 主題來提醒您。如需詳細資訊，請參閱「使用 AWS Config 尋找和追蹤 Elastic Beanstalk 資源」。

疑難排解

本章提供對 Elastic Beanstalk 環境進行故障診斷的指引。其會提供下列資訊：

- AWS Systems Manager 工具簡介，外加執行預先定義的 Elastic Beanstalk 執行手冊 (輸出故障診斷步驟和建議) 的程序。
- 如果您的環境狀態下降，您可以採取的動作以及您可以檢視的資源的一般指引。
- 依主題類別提供更具體的故障診斷提示。

如果環境的運作狀態變更為紅色，建議您先使用包含預先定義執行手冊的 AWS Systems Manager 工具，對 Elastic Beanstalk 進行故障診斷。如需詳細資訊，請參閱本章下一節中的 使用 Systems Manager 工具。

主題

使用 AWS Systems Manager Elastic Beanstalk 執行手冊

可以使用 Systems Manager 對 Elastic Beanstalk 環境進行故障診斷。為了協助您快速開始使用，Systems Manager 為 Elastic Beanstalk 提供預先定義的自動化執行手冊。自動化執行手冊是一種 Systems Manager 文件類型，它定義要對環境的執行個體和其他 AWS 資源執行的操作。

文件 `AWSSupport-TroubleshootElasticBeanstalk` 是一個自動化執行手冊，旨在幫助識別可能讓 Elastic Beanstalk 環境降級的許多常見問題。為此，它會檢查環境的元件，包括下列項目：EC2 執行個體、VPC、AWS CloudFormation 堆疊、負載平衡器、Auto Scaling 群組，以及與安全群組規則、路由表和 ACL 相關聯的網路組態。

它也提供從您的環境中將綁定的日誌檔案上傳至 AWS Support 的選項。

如需詳細資訊，請參閱 AWS Systems Manager Automation Runbook 參考 中的 `AWSSupport-TroubleshootElasticBeanstalk`。

使用 Systems Manager 執行 **AWSSupport-TroubleshootElasticBeanstalk** 執行手冊

> **Note**
>
> 在 Elastic Beanstalk 環境所在的相同 AWS 區域 中執行此程序。

1. 開啟 AWS Systems Manager 主控台。
2. 在導覽窗格的變更管理區段中，選擇自動化。
3. 選擇 Execute automation (執行自動化)。
4. 在由 Amazon 擁有索引標籤的自動化文件搜尋方塊中，輸入 `AWSSupport-TroubleshootElasticBeanstalk`。
5. 選取 AWSSupport-TroubleshootElasticBeanstalk 卡，然後選擇下一步。
6. 選取執行。
7. 在輸入參數區段中：
   a. 從自動化擔任角色下拉式清單中，選取允許 Systems Manager 代表您執行動作的角色的 ARN。
   b. 在應用程式名稱中，輸入 Elastic Beanstalk 應用程式的名稱。
   c. 在環境名稱中，輸入 Elastic Beanstalk 環境。
   d. (選用) 對於 S3UploaderLink，如果 AWS 支援工程師已提供 S3 連結以收集日誌，請輸入連結。
8. 選擇 Execute (執行)。

   如果有任何步驟失敗，請在步驟 ID 欄下選取失敗步驟的連結。這會顯示步驟的執行詳細資訊頁面。驗證錯誤訊息區段會顯示需要注意的步驟摘要。例如，`IAMPermissionCheck` 可能會顯示警告訊息。在這種情況下，您可以檢查在自動化擔任角色下拉式清單中選取的角色是否有必需的許可。

順利完成所有步驟之後，輸出會提供故障診斷步驟和建議，以將您的環境還原至健康狀態。

一般指導方針

錯誤訊息可以顯示在主控台的「事件」頁面中、日誌中或「運作狀態」頁面中。您也可以採取行動，從最近變更所導致的降級環境中復原。如果您環境的健全狀態變更為紅色，請嘗試進行下列動作：

- 檢閱最近的環境事件。由 Elastic Beanstalk 所提供的關於部署、載入和設定問題的訊息，經常會出現在此處。
- 檢閱最近的環境變更歷史記錄。變更歷史記錄會列出對您環境所做的所有組態變更，並包括其他資訊，例如哪些 IAM 使用者進行變更，以及設定了哪些組態參數。
- 叫出日誌來檢視最近的日誌檔案項目。Web 伺服器的日誌包含了關於傳入要求和錯誤的資訊。
- 連線到執行個體，並檢查系統資源。
- 轉返到應用程式先前運作中的版本。
- 還原最近的組態變更，或恢復儲存的組態。
- 部署新的環境。如果環境顯示為健全，請執行 CNAME 交換，將流量路由至新的環境，並繼續針對之前的環境來進行除錯。

類別

本主題依類別提供更具體的故障診斷提示。

主題

- 連線能力
- 建立環境並啟動執行個體
- 部署
- 醫療保健
- 組態
- Docker 容器故障診斷
- 常見問答集

連線能力

問題：在 Elastic Beanstalk 主控台中所建立的伺服器未出現於 Toolkit for Eclipse 中

您可以遵循將現有環境匯入 Eclipse 的指示，以手動方式匯入伺服器。

問題：無法從 Elastic Beanstalk 連線到 Amazon RDS。

若要將分開的 Amazon RDS 連線到您的 Elastic Beanstalk 應用程式，請執行下列動作：

- 確定 RDS 和您的 Elastic Beanstalk 應用程式位於同一個區域。
- 請確定您執行個體的 RDS 安全群組，對於您的 Elastic Beanstalk 環境使用的 Amazon EC2 安全群組，已獲得授權。關於使用 AWS 管理主控台來找出 EC2 安全群組名稱的方法，詳細指示請參閱 安全群組。如需設定 EC2 安全群組的詳細資訊，請參閱《Amazon Relational Database Service 使用者指南》中使用資料庫安全群組的「授權網路存取 Amazon EC2 安全群組」章節。
- 針對 Java，請確定 MySQL JAR 檔案位於您的 WEB-INF/lib 中。如需詳細資訊，請參閱將 Amazon RDS 資料庫執行個體新增到您的 Java 應用程式環境。

建立環境並啟動執行個體

事件：啟動環境失敗

當 Elastic Beanstalk 嘗試啟動環境卻遭遇失敗，就會發生此事件。Events (事件) 頁面上的過去事件將提醒您此問題的根源。

事件：建立環境操作已完成，但出現命令逾時。嘗試增加逾時期間。

若您使用的組態檔案會在執行個體上執行命令、下載大型檔案或安裝套件，部署您的應用程式可能會花費較長時間。增加命令逾時，讓應用程式在部署期間有更多時間來啟動。

事件：下列資源建立失敗：[AWSEBInstanceLaunchWaitCondition]

此訊息表示您環境的 Amazon EC2 執行個體未與成功啟動的 Elastic Beanstalk 進行通訊。若執行個體沒有網際網路連線，本情況就可能會發生。若您已將環境設定為在私有 VPC 子網路中啟動執行個體，請確認子網路具備 NAT，以允許執行個體連接至 Elastic Beanstalk。

事件：本區域需要服務角色。請將「服務角色」選項新增至環境。

Elastic Beanstalk 會使用服務角色來監控環境中的資源，並支援受管平台更新。如需詳細資訊，請參閱管理 Elastic Beanstalk 服務角色。

部署

問題：部署期間無法使用應用程式

由於 Elastic Beanstalk 使用便利的升級程序，因此可能會出現幾秒鐘的停機時間。使用滾動部署將部署對您生產環境的影響降到最低。

事件：無法建立 AWS Elastic Beanstalk 應用程式版本

您的應用程式原始碼套件可能過於龐大，或者您可能已達到應用程式版本配額。

事件：更新環境操作已完成，但出現命令逾時。嘗試增加逾時期間。

若您使用的組態檔案會在執行個體上執行命令、下載大型檔案或安裝套件，部署您的應用程式可能會花費較長時間。增加命令逾時，讓應用程式在部署期間有更多時間來啟動。

醫療保健

事件：CPU 使用率超過 95.00%

嘗試執行更多執行個體，或選擇不同的執行個體類型。

事件：Elastic Load Balancer awseb-*myapp* 沒有運作狀態良好的執行個體

若您的應用程式似乎正在運作，請確認應用程式的運作狀態檢查 URL 正確設定。如非此情況，請檢查運作狀態畫面及環境日誌以取得更多資訊。

事件：無法找到 Elastic Load Balancer awseb-*myapp*

您環境的負載平衡器可能已從外部移除。使用組態選項及 Elastic Beanstalk 提供的擴充功能，僅變更您環境的資源。重建您的環境或啟動新的環境。

事件：EC2 執行個體啟動失敗。等待新的 EC2 執行個體啟動...

您環境執行個體類型的可用性可能較低，或者您已達到帳戶的執行個體配額。檢查服務運作狀態儀表板，確認 Elastic Compute Cloud (Amazon EC2) 服務為綠色，或請求提高配額。

組態

事件：您無法為 Elastic Beanstalk 環境設定 Elastic Load Balancing Target 選項與應用程式運作狀態檢查 URL 選項的數值

`Target` 命名空間的 `aws:elb:healthcheck` 選項已作廢。請自環境移除 `Target` 選項命名空間，然後再嘗試更新。

事件：ELB 無法連接至位於相同 AZ 的多個子網路

若您嘗試在相同可用區域的子網路間移動負載平衡器，將出現此訊息。變更負載平衡器上的子網路，需要將其移出原始可用區域，然後將所需子網路移回原始可用區域。在此過程中，所有執行個體將於 AZ 內遷移，會產生重大停機時間。否則，請考慮建立新的環境，並執行 CNAME 交換。

Docker 容器故障診斷

事件：無法擷取 Docker 映像檔：最新的：無效的儲存庫名稱 ()，僅允許 [a-z0-9-_.]。針對日誌執行 tail 指令以取得詳細資訊。

使用 JSON 驗證程式來檢查 `dockerrun.aws.json` 檔案的語法。另外也請根據 Docker 組態 中所說明的要求，來驗證 dockerfile 的內容

事件：Dockerfile 中找不到 EXPOSE 指令，請中止部署

`Dockerfile` 或 `dockerrun.aws.json` 檔案並未宣告容器的通訊埠。使用 `EXPOSE` 指令 (`Dockerfile`) 或 `Ports` 區塊 (`dockerrun.aws.json` 檔案) 來公開傳入資料用的通訊埠。

事件： 無法從######下載身分驗證登入資料###

`dockerrun.aws.json` 為 `.dockercfg` 檔案提供了無效的 EC2 金鑰對和/或 S3 儲存貯體。或者，執行個體描述檔不具有 S3 儲存貯體適用的 GetObject 授權。請確認 `.dockercfg` 檔案包含有效的 S3 儲存貯體和 EC2 金鑰對。針對執行個體描述檔中的 IAM 角色，授予動作 `s3:GetObject` 的許可。如需詳細資訊，請參閱 管理 Elastic Beanstalk 執行個體描述檔

事件：活動執行失敗，原因：警告：無效的授權組態檔案

您的身分驗證檔案 (`config.json`) 格式不正確。請參閱 使用私有儲存庫中的映像

常見問答集

問：如何將我的應用程式 URL 自 myapp.us-west-2.elasticbeanstalk.com 變更為 www.myapp.com？

在 DNS 伺服器，註冊 CNAME 記錄，例如 **`www.mydomain.com CNAME mydomain.elasticbeanstalk.com`**。

問：如何為我的 Elastic Beanstalk 應用程式指定特定的可用區域？

您可使用 API、CLI、Eclipse 外掛程式或 Visual Studio 外掛程式，挑選特定的可用區域。如需有關使用 Elastic Beanstalk 主控台來指定可用區域的詳細資訊，請參閱適用於您 Elastic Beanstalk 環境的 Auto Scaling 群組。

問：如何變更我的環境內的執行個體類型？

若要變更環境的執行個體類型，請移至環境組態頁面，並在執行個體組態類別中選擇編輯。然後，選擇新的執行個體類型，然後選擇 Apply (套用) 以更新您的環境。在此之後，Elastic Beanstalk 會終止所有執行中的執行個體，並取代為新的執行個體。

問：如何判斷是否有人對環境做出組態變更？

若要查看此資訊，請在 Elastic Beanstalk 主控台的導覽窗格中，選擇 Change history (變更歷史記錄) 以顯示所有環境的組態變更清單。此清單包括變更的日期和時間、變更的目標組態參數和值，以及做出變更的 IAM 使用者。如需詳細資訊，請參閱變更歷史記錄。

問：能否在執行個體終止時，防止 Amazon EBS 磁碟區遭到刪除？

您環境的執行個體使用 Amazon EBS 來儲存，然而，當 Auto Scaling 終止執行個體時，根磁碟區將遭到刪除。不建議將狀態或其他資料存放於您的執行個體。若有需要，您可透過 AWS CLI 防止磁碟區遭到刪除：`$ aws ec2 modify-instance-attribute -b '/dev/sdc=<vol-id>:false`，如 AWS CLI 參考中所述。

問：如何從我的 Elastic Beanstalk 應用程式刪除個人資訊？

您的 Elastic Beanstalk 應用程式使用的 AWS 資源可能存放個人資訊。當您終止環境時，Elastic Beanstalk 會終止其建立的所有資源。使用組態檔案所新增的資源也會終止。不過，如果您在 Elastic Beanstalk 環境外建立 AWS 資源，並將其與應用程式建立關聯，您可能需要以手動方式檢查應用程式儲存的個人資訊未有保留情況。在這整個開發人員指南中，我們無論何時討論建立其他資源，都會提及何時應該考慮刪除。

Elastic Beanstalk 資源

以下相關資源可協助您使用此服務。

- Elastic Beanstalk API 參考 – 所有 SOAP 和查詢 API 的完整說明。此外，還包含所有 SOAP 資料類型的清單。
- elastic-beanstalk-samples on GitHub — 具有 Elastic Beanstalk 樣本配置文件（.ebextensions）的 GitHub 存儲庫。存放庫的`README.md`檔案包含其他 GitHub 儲存庫與範例應用程式的連結。
- Elastic Beanstalk 技術常見問答集 – 開發人員針對此產品最常詢問的問題。
- AWS Elastic Beanstalk 版本說明 — Elastic Beanstalk 服務、平台、主控台和 EB CLI 版本中的新功能、更新和修正程式的詳細資訊。

- 課程和工作坊 — 除了可以幫助提高 AWS 技能並獲得實踐經驗的自定進度實驗室之外，還可以鏈接到基於角色和專業課程的鏈接。
- AWS 開發人員中心 — 探索教學課程、下載工具，以及瞭解 AWS 開發人員活動。
- AWS 開發人員工具 — 開發人員工具、SDK、IDE 工具組，以及用於開發和管理 AWS 應用程式的命令列工具的連結。
- 入門資源中心 — 瞭解如何設定 AWS 帳戶、加入 AWS 社群，以及啟動您的第一個應用程式。
- 實作教學課程 — 按照 step-by-step 教學課程啟動您的第一個應用程式 AWS。
- AWS 白皮書 — 完整的技術 AWS 白皮書清單連結，涵蓋架構、安全性和經濟等主題，並由 AWS 解決方案架構師或其他技術專家撰寫。
- AWS Support 中心 — 建立和管理 AWS Support 案例的中心。同時也包含其他實用資源的連結，例如論壇、技術常見問答集、服務健康狀態和 AWS Trusted Advisor.
- AWS Support— 有關資訊的主要網頁 AWS Supportone-on-one, 快速回應的支援管道，可協助您在雲端中建置和執行應用程式.
- 聯絡我們 – 查詢有關 AWS 帳單、帳戶、事件、濫用與其他問題的聯絡中心。
- 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。

範例應用程式

下列是範例應用程式的下載連結，這些範例程式會部署為 開始使用 Elastic Beanstalk 的一部分。

> **Note**
>
> 自從您正在使用的平台推出後，可能又發佈了一些範例使用功能。如果範例執行失敗，請試著將您的平台更新為最新版本，請參閱 the section called “支援的平台” 中的說明。

- Docker – docker.zip
- 多容器泊塢視窗 — 2.zip docker-multicontainer-v
- 預配置碼頭工人（玻璃魚）-1.zip docker-glassfish-v
- Go – go.zip
- Corretto – corretto.zip
- Tomcat – tomcat.zip
- dotnet-core-linux. NET 核心
- . NET 核心-dotnet-asp-windows. 郵編
- Node.js – nodejs.zip
- PHP – php.zip
- Python – python.zip
- Ruby – ruby.zip

平台歷史記錄

AWS Elastic Beanstalk 平台歷程記錄的位置已變更。請參閱 AWS Elastic Beanstalk 平台文件中的平台歷史記錄。

主題

- Elastic Beanstalk 自訂平台

Elastic Beanstalk 自訂平台

> **Note**
>
> 2022 年 7 月 18 日，Elastic Beanstalk 已將所有以 Amazon Linux AMI (AL1) 為基礎的平台分支狀態設為已淘汰。這包括自訂平台。Elastic Beanstalk 不支援自訂平台。如需 Amazon Linux AMI 的 Elastic Beanstalk 淘汰的詳細資訊，請參閱 平台淘汰常見問答集。

本主題仍保留在本文件中，供任何在 Elastic Beanstalk 自訂平台功能淘汰前使用該功能的客戶做參考。過去，Elastic Beanstalk 自訂平台支援從 Amazon Linux AMI、RHEL 7、RHEL 6 或 Ubuntu 16.04 基本 AMI 建置 AMI。Elastic Beanstalk 不再支援這些作業系統。若要閱讀更多關於不再支援的自訂平台功能的資訊，請展開以下主題。

自訂平台

相較於自訂映像，自訂平台在多方面都是較為進階的自訂方式。自訂平台可讓您從頭開發完整的全新平台，您可自訂 Elastic Beanstalk 於平台執行個體上執行的作業系統、其他軟體和指令碼。若 Elastic Beanstalk 未針對使用的語言或基礎設施軟體的應用程式提供受管平台，此等靈活性可讓您自行建置一個。自訂映像可讓您修改 Amazon Machine Image (AMI) 供現有 Elastic Beanstalk 平台使用，相較於此，Elastic Beanstalk 仍會提供平台指令碼並控制平台的軟體堆疊。此外，透過自訂平台，您可以自動化、指令碼的方式建立您的自訂過程並加以維護，若是透過自訂映像，運作中執行個體的變更必須手動進行。

若要建立自訂平台，您要從支援的作業系統之一：Ubuntu、RHEL 或 Amazon Linux (確切版本編號請參閱 Platform.yaml 檔案格式 的 `flavor` 項目) 建置 AMI，並進一步新增自訂項目。您可以使用 Packer 來建立自己的 Elastic Beanstalk 平台，Packer 是一種開放原始碼工具，可針對許多平台建立機

器映像，包括與 Amazon Elastic Compute Cloud (Amazon EC2) 搭配使用的 AMI。Elastic Beanstalk 平台包括設定為執行一組支援應用程式之軟體的 AMI，以及可納入自訂組態選項和預設組態選項設定的中繼資料。

Elastic Beanstalk 會將 Packer 做為個別的內建平台進行管理，您無須擔心 Packer 組態及版本。

您須向 Elastic Beanstalk 提供 Packer 範本，藉此建立平台，同時建立範本為建置 AMI 所呼叫的指令碼和檔案。這些元件會與指定範本與中繼資料的平台定義檔案，一同封裝為名為平台定義封存的 ZIP 封存。

建立自訂平台時，不需要執行 Packer 的彈性 IP，即可啟動單一執行個體環境。Packer 隨後會啟動另一個執行個體來建置映像。您可重複使用此環境供多個平台或各個平台的多種版本使用。

> **Note**
>
> 自定義平台是特定於 AWS 地區的。若您在多個區域使用 Elastic Beanstalk，務必在每個區域各自建立您的平台。
> 在某些情況下，不會清除 Packer 啟動的執行個體，必須手動將其終止。若要了解如何手動清除這些執行個體，請參閱Packer 執行個體清除。

您帳戶內的使用者可於環境建立期間指定平台 ARN，藉此使用您的自訂平台。您用於建立自訂平台的 eb platform create 命令，會回傳這些 ARN。

每次建置自訂平台時，Elastic Beanstalk 會建立新的平台版本。使用者可指定平台名稱，取得平台的最新版本，或納入版本編號以取得該版本。

例如，欲透過 ARN **MyCustomPlatformARN** (可能為版本 3.0) 部署自訂平台的最新版本，您的 EB CLI 命令列將如下所示：

```
eb create -p MyCustomPlatformARN
```

欲部署版本 2.1，EB CLI 命令列將如下所示：

```
eb create -p MyCustomPlatformARN --version 2.1
```

您可以在建立和編輯現有自訂平台版本的標籤時，將標籤套用至自訂平台版本。如需詳細資訊，請參閱標記自訂平台版本。

建立自訂平台

欲建立自訂平台，您的應用程式根目錄必須納入平台定義檔案 `platform.yaml`，此會定義用於建立自訂平台的建置器類型。此檔案的格式如 Platform.yaml 檔案格式 中所述。您可從頭建立您的自訂平台，或使用其中一個範例自訂平台著手。

使用範例自訂平台

著手建立自訂平台的另一個方法是，使用其中一個平台定義封存範例來啟動載入您的自訂平台。在使用此範例之前，您必須設定的項目為來源 AMI 與區域。

> **Note**
>
> 請不要於生產中使用未修改的範例自訂平台。範例是為了說明自訂平台可用的部分功能，但尚未經過強化供生產使用。

NodePlatform_Ubuntu.zip

此自訂平台以 Ubuntu 16.04 為基礎，並支援 Node.js 4.4.4。我們於本章節中使用此自訂平台做為範例。

NodePlatform_RHEL.zip

此自訂平台以 RHEL 7.2 為基礎，並支援 Node.js 4.4.4。

NodePlatform_AmazonLinux. 拉鍊

此自訂平台以 Amazon Linux 2016.09.1 為基礎，並支援 Node.js 4.4.4。

TomcatPlatform_Ubuntu.zip

此自訂平台以 Ubuntu 16.04 為基礎，並支援 Tomcat 7/Java 8。

CustomPlatform_NodeSampleApp. 拉鍊

使用 express 和 ejs 來顯示靜態網頁的 Node.js 範例。

CustomPlatform_TomcatSampleApp. 拉鍊

部署時會顯示靜態網頁的 Tomcat 範例。

下載範例平台定義封存：`NodePlatform_Ubuntu.zip`。此檔案內含平台定義檔案、Packer 範本、Packer 於映像建立期間執行的指令碼，以及 Packer 於平台建立期間將複製至建置器執行個體的指令碼和組態檔案。

Example NodePlatform_Ubuntu.zip

```
|-- builder                Contains files used by Packer to create the custom platform
|-- custom_platform.json   Packer template
|-- platform.yaml          Platform definition file
|-- ReadMe.txt             Briefly describes the sample
```

平台定義檔案 `platform.yaml` 會讓 Elastic Beanstalk 知道 Packer 範本 `custom_platform.json` 的名稱。

```
version: "1.0"

provisioner:
  type: packer
  template: custom_platform.json
  flavor: ubuntu1604
```

Packer 範本會使用 Ubuntu AMI 做為 HVM 執行個體類型的平台映像基礎，讓 Packer 知道如何建置平台的 AMI。`provisioners` 區段則通知 Packer 將封存檔內 `builder` 資料夾的所有檔案，複製至執行個體，並於執行個體上執行 `builder.sh` 指令碼。指令碼完成時，Packer 會從已修改的執行個體建立映像。

Elastic Beanstalk 會建立三個可用於在 Packer 標籤 AMI 的環境變數：

AWS_EB_PLATFORM_ARN

自訂平台的 ARN。

AWS_EB_PLATFORM_NAME

自訂平台的名稱。

AWS_EB_PLATFORM_VERSION

自訂平台的版本。

範例 `custom_platform.json` 檔案使用這些變數來定義下列用於指令碼的值：

- `platform_name`，由 `platform.yaml` 設定
- `platform_version`，由 `platform.yaml` 設定
- `platform_arn`，由範例 `builder.sh` 檔案結尾顯示的主要建置指令碼 `custom_platform.json` 所設定

custom_platform.json 檔案包含兩個您必須提供屬性的值：source_ami 和 region。如需有關選擇正確 AMI 和 Region 值的詳細資訊，請參閱更新eb-custom-platforms-samples GitHub 儲存庫中的封裝程式範本。

Example custom_platform.json

```
{
  "variables": {
    "platform_name": "{{env `AWS_EB_PLATFORM_NAME`}}",
    "platform_version": "{{env `AWS_EB_PLATFORM_VERSION`}}",
    "platform_arn": "{{env `AWS_EB_PLATFORM_ARN`}}"
  },
  "builders": [
    {
      ...
      "region": "",
      "source_ami": "",
      ...
    }
  ],
  "provisioners": [
    {...},
    {
      "type": "shell",
      "execute_command": "chmod +x {{ .Path }}; {{ .Vars }} sudo {{ .Path }}",
      "scripts": [
        "builder/builder.sh"
      ]
    }
  ]
}
```

您納入平台定義封存的指令碼和其他檔案，視您欲在執行個體上進行的修改有極大差異。範例平台包含下列指令碼：

- 00-sync-apt.sh – 已標示為註解：apt -y update。我們將此命令變更為註解，因為其會要求使用者輸入訊息，中斷自動化套件更新。這可能是 Ubuntu 的問題。然而，最佳作法仍建議執行 apt -y update。因此，我們的範例指令碼留下此命令做為參考。
- 01-install-nginx.sh – 安裝 nginx。
- 02-setup-platform.sh – 安裝 wget、tree 和 git。將掛勾和記錄組態複製至執行個體，並建立下列目錄：

- `/etc/SampleNodePlatform` – 部署期間容器組態檔案的上傳位置。
- `/opt/elasticbeanstalk/deploy/appsource/` – 部署期間 `00-unzip.sh` 指令碼上傳應用程式原始碼的位置 (如需此指令碼的資訊，請參閱平台指令碼工具章節)。
- `/var/app/staging/` – 部署期間處理應用程式原始碼的位置。
- `/var/app/current/` – 應用程式原始碼處理後執行的位置。
- `/var/log/nginx/healthd/` – 增強型運作狀態代理程式寫入日誌的位置。
- `/var/nodejs` – 部署期間 Node.js 檔案上傳的位置。

使用 EB CLI，透過範例平台定義封存來建立您的第一個自訂平台。

欲建立自訂平台

1. 安裝 EB CLI。
2. 建立您將擷取範例自訂平台的目錄。

```
~$ mkdir ~/custom-platform
```

3. 擷取 `NodePlatform_Ubuntu.zip` 到目錄，然後變更到解壓縮目錄。

```
~$ cd ~/custom-platform
~/custom-platform$ unzip ~/NodePlatform_Ubuntu.zip
~/custom-platform$ cd NodePlatform_Ubuntu
```

4. 編輯 `custom_platform.json` 檔案，並且為 `source_ami` 和 `region` 屬性提供值。如需詳細資訊，請參閱更新 Packer 範本。
5. 執行 eb platform init 並依提示初始化平台儲存庫。

   您可將 eb platform 縮短為 ebp。

> **Note**
>
> 視窗 PowerShell 使用ebp做為指令別名。如果您在 Windows 中執行 EB CLI PowerShell，請使用以下命令的長形式：eb platform。

```
~/custom-platform$ eb platform init
```

此命令亦於目前目錄內建立目錄 `.elasticbeanstalk`，並將組態檔案 `config.yml` 新增至該目錄。請勿變更或刪除此檔案，因為 Elastic Beanstalk 建立自訂平台時需要此檔案。

eb platform init 預設會使用目前資料夾的名稱做為自訂平台的名稱，在此範例中為 `custom-platform`。

6. 執行 eb platform create 來啟動 Packer 環境，並取得自訂平台的 ARN。您稍後在此自訂平台建立環境時，將需要此值。

```
~/custom-platform$ eb platform create
...
```

依預設，Elastic Beanstalk 會建立自訂平台的執行個體描述檔 `aws-elasticbeanstalk-custom-platform-ec2-role`。若您想要使用現有的執行個體描述檔，請將選項 `-ip` *`INSTANCE_PROFILE`* 新增至 eb platform create 命令。

> **Note**
>
> 如果您使用 Elastic Beanstalk 預設執行個體描述檔 `aws-elasticbeanstalk-ec2-role`，Packer 將無法建立自訂平台。

EB CLI 會顯示 Packer 環境的事件輸出，直至建立完成。您可按 Ctrl+C 退出事件檢視。

7. 您可使用 eb platform logs 命令來檢查日誌是否有錯誤。

```
~/custom-platform$ eb platform logs
...
```

8. 您稍後可透過 eb platform events 來檢查程序。

```
~/custom-platform$ eb platform events
...
```

9. 透過 eb platform status 來檢查您平台的狀態。

```
~/custom-platform$ eb platform status
...
```

操作完成時，您將具備可用於啟動 Elastic Beanstalk 環境的平台。

透過主控台建立環境時，可使用自訂平台。請參閱建立新的環境精靈。

欲在您的自訂平台上啟動環境

1. 為您的應用程式建立目錄。

```
~$ mkdir custom-platform-app
~$ cd ~/custom-platform-app
```

2. 初始化應用程式儲存庫。

```
~/custom-platform-app$ eb init
...
```

3. 下載範例應用程式 NodeSampleApp.zip。
4. 擷取範例應用程式。

```
~/custom-platform-app$ unzip ~/NodeSampleApp.zip
```

5. 執行 eb create -p ***CUSTOM-PLATFORM-ARN*** 來啟動執行您自訂平台的環境，其中 *CUSTOM-PLATFORM-ARN* 為 eb platform create 命令回傳的 ARN。

```
~/custom-platform-app$ eb create -p CUSTOM-PLATFORM-ARN
...
```

平台定義存檔內容

平台定義封存為相當於應用程式原始碼套件的平台。平台定義封存為 ZIP 檔案，其中包含平台定義檔案、Packer 範本及 Packer 範本用來建立您平台的指令碼和檔案。

> **Note**
>
> 當您使用 EB CLI 來建立自訂平台，EB CLI 會自平台儲存庫的檔案和資料夾建立平台定義封存，因此您無須手動建立封存。

平台定義檔案是 YAML 格式的檔案，必須命名為 `platform.yaml`，並位於平台定義封存的根目錄中。如需平台定義檔案支援的必要和選用金鑰清單，請參閱建立自訂平台。

您無須以特定方式來命名 Packer 範本，但該檔案名稱必須與平台定義檔案中指定的佈建範本相符。如需建立 Packer 範本的說明，請參閱官方 Packer 文件。

平台定義封存中的其他檔案為範本在建立 AMI 前用來自訂執行個體的指令碼和檔案。

自訂平台掛勾

Elastic Beanstalk 使用標準化目錄結構在自訂平台上進行掛接。這些是生命週期事件期間及回應管理操作所執行的指令碼，如您環境中的執行個體啟動時，或使用者啟動部署或使用重新啟動應用程式伺服器功能時。

您需要將指令碼掛接在 `/opt/elasticbeanstalk/hooks/` 資料夾的其中一個子資料夾。

> **Warning**
>
> 不支援在受管平台上使用自訂平台掛勾。自訂平台掛勾是專為自訂平台所設計。在 Elastic Beanstalk 受管平台上，它們的運作方式可能不同或有些問題，跨平台的行為可能有所不同。在 Amazon Linux AMI 平台 (前身是 Amazon Linux 2) 上，在某些情況下，它們可能仍然以有用的方式運作，請謹慎使用。
>
> 自訂平台掛鉤是存在於 Amazon Linux AMI 平台上的舊版功能。在 Amazon Linux 2 平台，`/opt/elasticbeanstalk/hooks/` 資料夾中的自訂平台掛勾會完全停止使用。Elastic Beanstalk 不會讀取或執行這些掛勾。Amazon Linux 2 平台支援一種新的平台掛勾，專門設計用於擴展 Elastic Beanstalk 受管平台。您可以將自訂指令碼和程式直接新增到應用程式原始碼套件中的掛勾目錄。Elastic Beanstalk 會在各個執行個體佈建階段期間執行這些項目。如需詳細資訊，請展開the section called “擴充 Linux 平台”中的平台掛勾一節。

掛勾會整理成下列資料夾：

- `appdeploy` – 應用程式部署期間執行的指令碼。當新的執行個體啟動時，或用戶端啟動新版本的部署時，Elastic Beanstalk 會執行應用程式部署。
- `configdeploy` – 用戶端在執行會影響執行個體上軟體組態的組態更新時，所執行的指令碼，例如，設定環境屬性或啟用日誌輪換至 Amazon S3。
- `restartappserver` – 用戶端執行重新啟動應用程式伺服器操作所執行的指令碼。
- `preinit` – 執行個體啟動載入作業期間所執行的指令碼。
- `postinit` – 執行個體啟動載入作業之後所執行的指令碼。

appdeploy、configdeploy 與 restartappserver 資料夾包含 pre、enact 和 post 子資料夾。在操作的各個階段，會以字母順序執行 pre 資料夾內的所有指令碼，之後則依序為 enact 資料夾和 post 資料夾中的指令碼。

執行個體啟動時，Elastic Beanstalk 會依序執行 preinit、appdeploy 和 postinit。後續在運作中執行個體上進行部署時，Elastic Beanstalk 會執行 appdeploy 掛勾。當使用者更新執行個體軟體組態設定時，會執行 configdeploy 掛勾。使用者啟動應用程式伺服器重新啟動，僅在此時才會執行 restartappserver 掛勾。

當您的指令碼出現錯誤，能夠以非零狀態退出並寫入 stderr，來讓操作失敗。寫入 stderr 的訊息會出現在操作失敗所輸出的事件。Elastic Beanstalk 也會在日誌檔案 /var/log/eb-activity.log 中擷取此資訊。若您不希望操作失敗，請回傳 0 (零)。寫入 stderr 或 stdout 的訊息會顯示於部署日誌，但不會顯示於事件串流，除非操作失敗。

Packer 執行個體清除

部分情況下 (例如在 Packer 建置器程序完成前將其刪除)，Packer 啟動的執行個體不會被清除。這些執行個體不屬於 Elastic Beanstalk 環境，僅能使用 Amazon EC2 服務來檢視並終止。

欲手動清除這些執行個體

1. 開啟 Amazon EC2 主控台。
2. 請確定您位於使用 Packer 建立執行個體的相同 AWS 區域中。
3. 在 Resources (資源) 下選擇 *N* Running Instances (N 個運作中的執行個體)，其中 *N* 表示運作中執行個體的數目。
4. 按一下查詢文字方塊。
5. 選取 Name (名稱) 標記。
6. 輸入 packer。

   查詢應類似：tag:Name: packer
7. 選取符合查詢的執行個體。
8. 若 Instance State (執行個體狀態) 為 running (運作中)，請依序選擇 Actions (動作)、Instance State (執行個體狀態)、Stop (停止)，然後再選取 Actions (動作)、Instance State (執行個體狀態)、Terminate (終止)。

Platform.yaml 檔案格式

此 platform.yaml 檔案的格式如下。

```
version: "version-number"

provisioner:
   type: provisioner-type
   template: provisioner-template
   flavor: provisioner-flavor

metadata:
   maintainer: metadata-maintainer
   description: metadata-description
   operating_system_name: metadata-operating_system_name
   operating_system_version: metadata-operating_system_version
   programming_language_name: metadata-programming_language_name
   programming_language_version: metadata-programming_language_version
   framework_name: metadata-framework_name
   framework_version: metadata-framework_version

option_definitions:
   - namespace: option-def-namespace
     option_name: option-def-option_name
     description: option-def-description
     default_value: option-def-default_value

option_settings:
   - namespace: "option-setting-namespace"
     option_name: "option-setting-option_name"
     value: "option-setting-value"
```

以這些值取代預留位置：

version-number

必要。YAML 定義的版本。必須為 **1.0**。

provisioner-type

必要。用於建立自訂平台的建置器類型。必須為 **packer**。

provisioner-template

必要。內含 *provisioner-type* 設定的 JSON 檔案。

provisioner-flavor

選用。供 AMI 使用的基礎作業系統。下列其中一項：

amazon (預設)

Amazon Linux。若未指定，則是平台建立時的 Amazon Linux 最新版本。

Amazon Linux 2 不是支援的作業系統類別。

ubuntu1604

Ubuntu 16.04 LTS

rhel7

RHEL 7

rhel6

RHEL 6

metadata-maintainer

選用。平台擁有人的聯絡資訊 (100 個字元)。

metadata-description

選用。平台的描述 (2000 個字元)。

metadata-operating_system_name

選用。平台作業系統的名稱 (50 個字元)。篩選 ListPlatformVersionsAPI 的輸出時，可使用此值。

metadata-operating_system_version

選用。平台作業系統的版本 (20 個字元)。

metadata-programming_language_name

選用。平台支援的程式設計語言 (50 個字元)。

metadata-programming_language_version

選用。平台語言的版本 (20 個字元)。

metadata-framework_name

選用。平台使用之 Web 架構的名稱 (50 個字元)。

metadata-framework_version

選用。平台 Web 架構的版本 (20 個字元)。

option-def-namespace

選用。`aws:elasticbeanstalk:container:custom` 底下的命名空間 (100 個字元)。

option-def-option_ ##

選用。選項的名稱 (100 個字元)。您至多可定義 50 個由平台向使用者提供的自訂組態選項。

option-def-description

選用。選項的描述 (1024 個字元)。

option-def-default_ #

選用。使用者未指定時應使用的預設值。

下列範例會建立選項 **NPM_START**。

```
options_definitions:
 -  namespace: "aws:elasticbeanstalk:container:custom:application"
    option_name: "NPM_START"
    description: "Default application startup command"
    default_value: "node application.js"
```

option-setting-namespace

選用。選項的命名空間。

option-setting-option_ ##

選用。選項的名稱。您可以指定最多 50 個 Elastic Beanstalk 提供的選項。

option-setting-value

選用。使用者未指定時應使用的值。

下列範例會建立選項 **TEST**。

```
option_settings:
 - namespace: "aws:elasticbeanstalk:application:environment"
   option_name: "TEST"
   value: "This is a test"
```

標記自訂平台版本

您可以將標籤套用至自 AWS Elastic Beanstalk 訂平台版本。標籤是與 AWS 資源相關聯的索引鍵值配對。如需 Elastic Beanstalk 資源標記、使用案例、標籤索引鍵和值限制條件的相關資訊，以及支援的資源類型，請參閱標記 Elastic Beanstalk 應用程式資源。

您可以在建立自訂平台版本時指定標籤。您可以在現有的自訂平台版本中新增或移除標籤，以及更新現有標籤的值。您最多可以為每個自訂平台版本新增 50 個標籤。

在自訂平台版本建立期間新增標籤

若您使用 EB CLI 來建立自訂平台版本，請使用帶有 eb platform create 的 `--tags` 選項來新增標籤。

```
~/workspace/my-app$ eb platform create --tags mytag1=value1,mytag2=value2
```

對於 AWS CLI 或其他 API 型用戶端，請使用命令上的`--tags`參數來新增標籤。create-platform-version

```
$ aws elasticbeanstalk create-platform-version \
      --tags Key=mytag1,Value=value1 Key=mytag2,Value=value2 \
      --platform-name my-platform --platform-version 1.0.0 --platform-definition-bundle
 S3Bucket=DOC-EXAMPLE-BUCKET,S3Key=sample.zip
```

管理現有自訂平台版本的標籤

您可以新增、更新和刪除現有 Elastic Beanstalk 自訂平台版本中的標籤。

如果您使用 EB CLI 來更新自訂平台版本，請使用 eb tags 新增、更新、刪除或列出標籤。

例如，以下命令會列出自訂平台版本中的標籤。

```
~/workspace/my-app$ eb tags --list --resource "arn:aws:elasticbeanstalk:us-east-2:my-
account-id:platform/my-platform/1.0.0"
```

下列命令會更新標籤 `mytag1` 並刪除標籤 `mytag2`。

```
~/workspace/my-app$ eb tags --update mytag1=newvalue --delete mytag2 \
      --resource "arn:aws:elasticbeanstalk:us-east-2:my-account-id:platform/my-
platform/1.0.0"
```

如需完整選項清單和更多範例，請參閱 `eb tags`。

對於 AWS CLI 或其他 API 型用戶端，請使用指list-tags-for-resource令列出自訂平台版本的標籤。

```
$ aws elasticbeanstalk list-tags-for-resource --resource-arn
 "arn:aws:elasticbeanstalk:us-east-2:my-account-id:platform/my-platform/1.0.0"
```

使用 update-tags-for-resource 命令新增、更新或刪除自訂平台版本中的標籤。

```
$ aws elasticbeanstalk update-tags-for-resource \
      --tags-to-add Key=mytag1,Value=newvalue --tags-to-remove mytag2 \
      --resource-arn "arn:aws:elasticbeanstalk:us-east-2:my-account-id:platform/my-
platform/1.0.0"
```

在 `--tags-to-add` 的 update-tags-for-resource 參數中，同時指定欲新增和欲更新的標籤。如此將新增不存在的標籤，並更新現有標籤的值。

> **Note**
>
> 若要將某些 EB CLI 和 AWS CLI 命令搭配 Elastic Beanstalk 自訂平台版本使用，您需要自訂平台版本的 ARN。您可使用下列命令來擷取 ARN。
>
> ```
> $ aws elasticbeanstalk list-platform-versions
> ```
>
> 使用 `--filters` 選項來篩選輸出以縮減到您的自訂平台名稱。

文件歷史紀錄

下表說明自 2024 年 4 月以來，《AWS Elastic Beanstalk 開發人員指南》的重要變更。

變更	描述	日期
QuickStart 對於. NET 核心視窗	新 QuickStart 的. NET 核心視窗。	2024年6月28日
QuickStart 對於碼頭工人	碼頭工 QuickStart 人的新功能。	2024年6月19 日
防止跨環境 Amazon S3 儲存貯體存取	新功能防止跨環境 Amazon S3 儲存貯體存取。	2024年6月12日
QuickStart 適用於. NET 核心	新 QuickStart 的. NET 核心視窗。	2024年5月28日
QuickStart 對於 PHP	PHP QuickStart 的新功能。	2024年5月10日
QuickStart 對於 Node.js	QuickStart 適用於 Node.js 的新功能。	2024年5月5日
QuickStart 為了去	Go QuickStart 的新功能。	2024年5月5日
Elastic Beanstalk 平台發布時間表	已新增包含排程的新主題即將發布的平台分支。移動正在淘汰的平台分支排程並淘汰的平台分支歷史記錄進入此主題。	2024年5月1日
AWSElasticBeanstalkRoleCore AWS 受管理政策	已更新 AWS 受管理策略中的權限。	2024 年 4 月 30 日
AWSElasticBeanstalkManagedUpdatesServiceRolePolicy AWS 受管理政策	已更新 AWS 受管理策略中的權限。	2024 年 4 月 30 日

AWSElasticBeanstalkManagedUpdatesInternalServiceRolePolicy AWS 受管理政策	已更新 AWS 受管理策略中的權限。	2024 年 4 月 30 日
AWSElasticBeanstalkMaintenance AWS 受管理政策	已更新 AWS 受管理策略中的權限。	2024 年 4 月 30 日
AWSElasticBeanstalkInternalMaintenanceRolePolicy AWS 受管理政策	已更新 AWS 受管理策略中的權限。	2024 年 4 月 30 日

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。