歡迎學習模組 10：實現架構自動化。

模組概覽

小節目錄

1. 架構需求
2. 實現自動化的原因
3. 實現基礎設施自動化
4. 實現部署自動化
5. AWS Elastic Beanstalk

演示

- 分析 AWS CloudFormation 範本結構並創建堆疊

實驗

- 指導實驗：使用 AWS CloudFormation 實現自動化基礎設施部署
- 挑戰實驗：實現基礎設施部署自動化

 **知識測驗**

本模組包含以下章節：

1. 架構需求
2. 實現自動化的原因
3. 實現基礎設施自動化
4. 實現部署自動化
5. AWS Elastic Beanstalk

本模組還包括：

- 講師指導的演示，*首先分析 AWS CloudFormation 範本的結構，然後根據範本創建堆疊。*
- 指導實驗，*提供使用 AWS CloudFormation 在 AWS 帳戶中創建資源的實踐機會。*
- 挑戰實驗，*使用 AWS CloudFormation 創建支援咖啡館使用案例的 Amazon Web Services (AWS) 資源。*

最後，您需要完成一個*知識測驗*，以測試您對本模組中涵蓋的關鍵概念的理解程度。

模組目標

學完本模組後，您應該能夠：

- 識別何時實現自動化及原因
- 確定如何使用 AWS CloudFormation 建模、創建和管理 AWS 資源集合
- 使用 Quick Start AWS CloudFormation 範本設置架構
- 說明如何借助 AWS System Manager 和 AWS OpsWorks 實現基礎設施和部署自動化
- 說明如何使用 AWS Elastic Beanstalk 部署簡單的應用程式

學完本模組後，您應該能夠：

- 識別何時實現自動化及原因
- 確定如何使用 AWS CloudFormation 建模、創建和管理 AWS 資源集合
- 使用 Quick Start AWS CloudFormation 範本設置架構
- 說明如何借助 AWS System Manager 和 AWS OpsWorks 實現基礎設施和部署自動化
- 說明如何使用 AWS Elastic Beanstalk 部署簡單的應用程式

模組 10：實現架構自動化

第 1 節：架構需求

介紹第 1 節：架構需求。

咖啡館業務要求

咖啡館現在在多個國家/地區設有分店，必須開始自動化才能保持發展。其組織有許多不同的架構，需要一種方法對它們進行統一的部署、管理和更新。

到目前為止，咖啡館創建了 AWS 資源並手動配置了應用程式 – 主要使用 AWS 管理主控台。這種方法可以有效說明咖啡館快速開發 Web 服務，構建支援員工和客戶需求的基礎設施。但是，他們發現將部署複製到新的 AWS 區域，以支援在多個國家/地區設立的咖啡館分店挑戰重重。

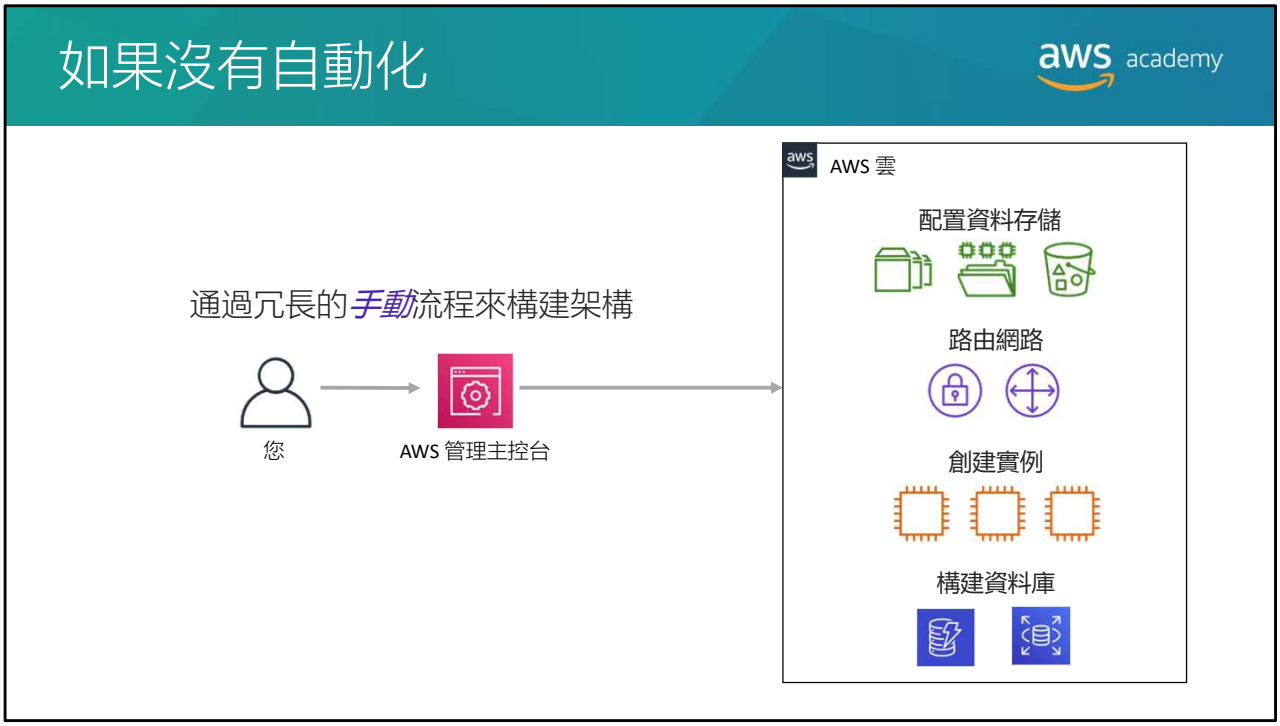
他們還希望擁有具有可靠匹配配置的獨立開發和生產環境。他們意識到，必須開始自動化才能支援持續增發展。其組織有許多不同的架構，需要一種方法快速、一致且可靠地部署、管理和更新這些架構。

在本模組中，您將瞭解提供自動化的 AWS 服務，包括 AWS CloudFormation。通過使用 AWS CloudFormation，您將能夠幫助咖啡館滿足這些新的業務需求。

模組 10：實現架構自動化

第 2 節：實現自動化的原因

介紹第 2 節：實現自動化的原因。



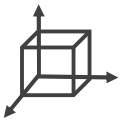
構建大規模計算環境需要耗費大量的時間和精力。

許多組織開始使用 AWS 的方法是，手動創建 Amazon Simple Storage Service (Amazon S3) 存儲桶或啟動 Amazon Elastic Compute Cloud (Amazon EC2) 實例並在其上運行 Web 伺服器。然後，隨著時間的推移，他們會手動添加更多資源，因為他們發現擴大對 AWS 的使用可以滿足其他業務需求。但是，手動管理和維護這些資源很快就會變得具有挑戰性。

要問的一些問題包括：

- 您要將主要精力用在何處：設計還是實施？手動實施有哪些風險？
- 理想情況下，如何更新生產伺服器？如何在多個地理區域推出部署？當出現問題時，又如何回滾到上個已知良好版本？
- 如何調試部署？在向客戶推出部署之前，能否修復應用程式中的錯誤？如何發現問題所在，然後修復問題，以使其*保持*修復狀態？
- 如何管理對組織中各種系統和子系統的依賴關係？
- 最後，通過手動配置完成所有這些任務是否現實？

手動流程帶來的風險



不支持大規模可重複性

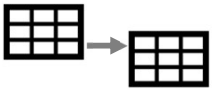
- 如何將部署複製到多個區域？

沒有版本控制

- 如何將生產環境回滾到以前的版本？

缺少審計跟蹤

- 如何確保合規性？如何在資源級別跟蹤對配置詳細資訊的更改？



資料管理不一致

- 例如，如何確保跨多個 Amazon Elastic Compute Cloud (Amazon EC2) 實例的配置匹配？

手動創建資源並向環境添加新的特性和功能*無法擴展*。如果您負責的是大型企業應用程式，則會面臨人手不足的困境。

此外，從頭開始創建架構和應用程式沒有固有的*版本控制*。在緊急情況下，將生產堆疊回滾到之前的版本很有用，但如果是手動創建環境，就不可能這樣做。

對於許多合規性和安全性相關情況，擁有*審計跟蹤*非常重要。允許組織中的任何人手動控制和編輯環境存在風險。

最後，要最大限度地降低風險，*一致性*至關重要。自動化使您能夠保持一致性。

遵循 AWS 架構完善框架的原則

- 卓越運營設計原則
  - 以代碼形式運營
  - 進行頻繁、可逆的微小更改
- 可靠性支柱設計原則
  - 管理自動化方面的變更

按照手動方法創建和維護 AWS 資源和部署並不能使您滿足這些準則。

還要考慮手動方法符合 AWS 架構完善的框架的程度。卓越運營的 六項設計原則之一是*以代碼形式運營*。在雲中，您可以將用於應用程式碼的相同的工程學科應用於整個環境。您可以將整個工作負載（應用程式、基礎設施和其他資源）定義為代碼，並使用代碼進行更新。您可以編寫運行程式腳本，並在運行時通過觸發它們來回應事件以實現自動化。通過以代碼形式運營，您可以限制人為錯誤並實現對事件的一致回應。

另一項卓越運營設計原則是*進行頻繁、可逆的微小更改*。也就是說，設計工作負載以支援對元件的定期更新，以便您增加工作負載中的有益變化。以較小的增量進行更改，如果這些更改無法幫助您識別和解決環境中引入的問題，則可以撤銷這些更改。如果可能，在進行這些更改時儘量不要影響客戶。

最後，架構完善的可靠性支柱 的設計原則之一是*管理自動化更改*。對基礎設施的更改應通過自動化來完成。因此，必須管理對自動化的更改。 對許多組織來說，更改生產系統是面臨的最大風險領域之一。在操作中，盡可能使用自動化，比如測試和部署更改、添加或刪除容量以及遷移資料。

模組 10：實現架構自動化

第 3 節：實現基礎設施自動化

介紹第 3 節：實現基礎設施自動化。

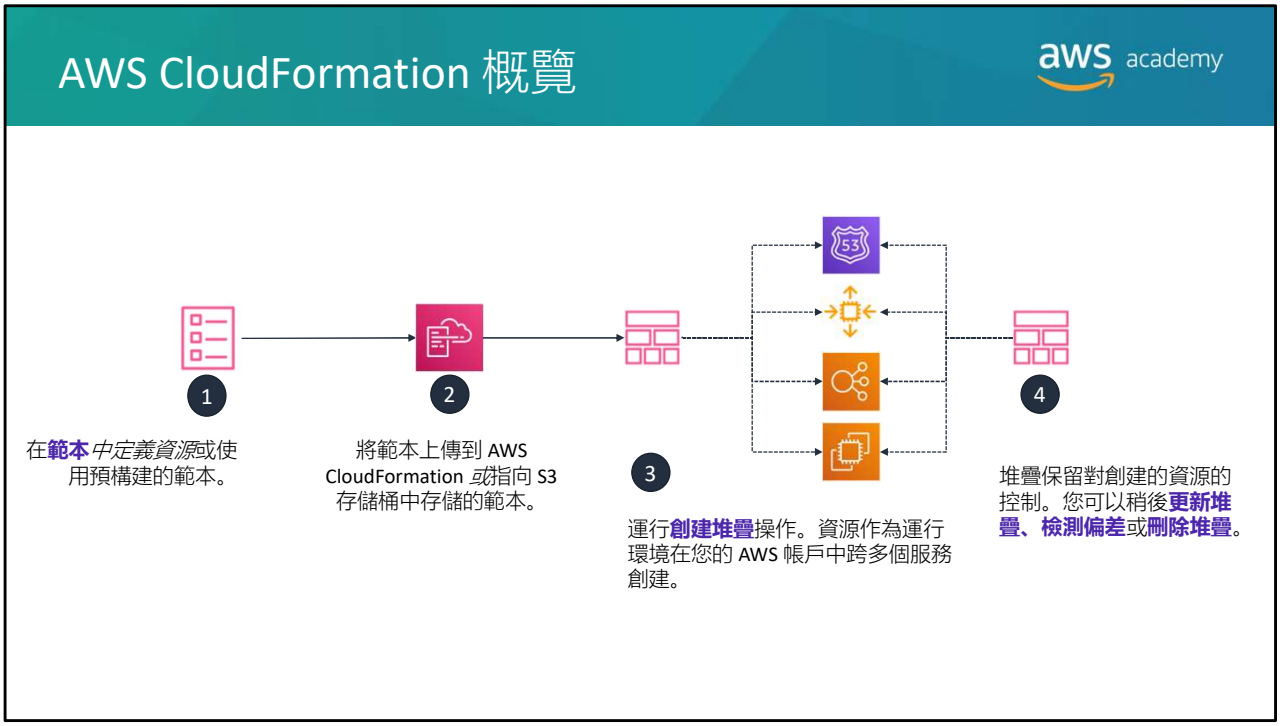
實現基礎設施自動化

- AWS CloudFormation 提供了一種簡化的方法來建模、創建和管理 AWS 資源的集合
  - 資源集合稱為 AWS CloudFormation 堆棧
  - 無需支付額外費用（僅為您創建的資源付費）
- 可以創建、更新和刪除堆疊
- 支援有序且可預測的資源預置和更新
- 啟用 AWS 資源部署的版本控制

AWS CloudFormation 以可重複的方式預置資源。它允許您構建和重建基礎設施和應用程式，無需執行手動操作或編寫自訂腳本。使用 AWS CloudFormation，您可以編寫一個文檔來描述您的基礎設施應該是什麼，包括應作為部署一部分的所有 AWS 資源。您可以將這份文檔視為*模型*。然後，使用該模型創建資源，因為 AWS CloudFormation 實際上可以在您的帳戶中創建資源。

當您使用 AWS CloudFormation 創建資源時，這稱為 AWS CloudFormation *堆棧*。您可以創建堆疊、更新堆疊或刪除堆疊。因此，您可以按有序且可預測的方式預置資源。

使用 AWS CloudFormation，您可以將基礎設施視為代碼 (IaC)。您可以使用任意代碼編輯器進行編寫，將其簽入*版本控制*系統（如 GitHub 或 AWS CodeCommit）中，並在部署到相應環境之前與團隊成員一起審核檔。如果您創建用於對部署建模的 AWS CloudFormation 文檔已簽入版本控制系統，則您始終可以刪除堆疊，簽出舊版本的文檔，然後從中創建堆疊。通過版本控制，您可以使用基本的回滾功能。



該圖演示了 AWS CloudFormation 的工作原理。首先，定義要創建的 AWS 資源。在此示例中，創建了幾個 EC2 實例、一個負載等化器、一個 Auto Scaling 組和一個 Amazon Route 53 託管區域。在 AWS CloudFormation *範本*中定義資源。您可以從頭開始創建範本，也可以使用預構建的範本。許多示例範本也可供使用。

儘管 AWS CloudFormation 為 AWS 服務提供了廣泛的支援，但並非所有資源都可以由 AWS CloudFormation 創建。有關詳細資訊，請參閱 AWS CloudFormation 支援的資源列表。

接下來，將範本上傳到 AWS CloudFormation。或者，您可以將範本存儲在 Amazon S3 上，然後將 AWS CloudFormation 指向存儲它的位置。

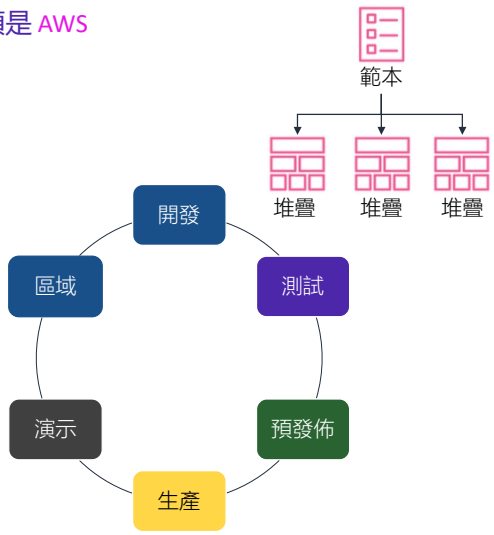
第三，運行*創建堆疊*操作。執行此操作時，AWS CloudFormation 服務將讀取範本中指定的內容，並在您的 AWS 帳戶中創建所需的資源。單個堆疊可以跨多個 AWS 服務在單個區域中創建和配置資源。

最後，您可以觀察堆疊創建過程的進度。堆疊成功完成後，它創建的 AWS 資源就存在於您的帳戶中。堆疊物件仍然存在，其作用就像創建的所有資源的控制碼一樣。當您以後要執行操作時，這很有用。例如，您可能想要*更新堆疊*（創建其他 AWS 資源或修改現有資源）或*刪除堆疊*（清理和刪除堆疊創建的資源）。

基礎設施即代碼 (IaC)

對於 AWS 雲開發，IaC 的內置選項是 AWS CloudFormation。

- IaC 是通過編寫具有以下特性的範本檔來預置和管理您的雲資源的過程 –
  - 人類可讀
  - 機器可用
- 它是可以複製、重新部署和重新定位的基礎設施
- 您可以在出現故障時回滾到上一個良好狀態

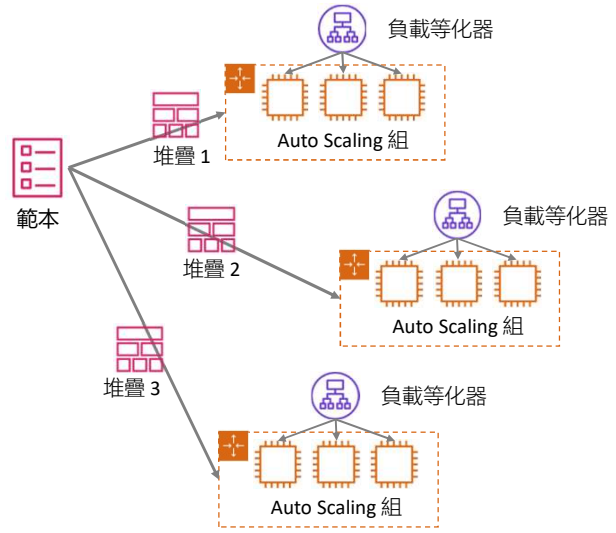


基礎設施即代碼 (IaC) 是一個行業術語，它指的是通過在人類可讀和機器可用的範本檔中定義雲資源來進行預置和管理的過程。

IaC 越來越受歡迎，因為它提供了一個可行的解決方案來應對各種挑戰，例如如何輕鬆、可靠且一致地複製和重新部署基礎設施以及重新調整其用途。

從客戶角度來看，AWS CloudFormation 的事務性是其最大的優勢之一。AWS CloudFormation 是事務性的 – 在出現故障時服務將*回滾到*上一個良好狀態。

基礎設施即代碼：優勢



減少多個匹配環境

- 快速部署複雜環境
- 提供配置一致性
- 需要時可輕鬆清理（刪除堆疊會刪除創建的資源）
- 輕松將更改傳播到所有堆疊
  - 修改範本，在所有堆疊上運行更新堆疊

優勢

- 可重用性
- 可重複性
- 可維護性

現在，請詳細考慮 IaC 的一些優勢。如果您使用代碼構建基礎設施，將獲得快速部署複雜環境的能力之類的好處。使用一個範本（或多個範本的組合），您可以重複構建相同的複雜環境。

在此示例中，一個範本可以用來創建三個不同的堆疊。每個堆疊通常可在幾分鐘內快速創建。每個堆疊一致地複製複雜的配置詳細資訊。

如果堆疊 2 是您的測試環境，堆疊 3 是您的生產環境，您可以更有把握地認為，如果測試作業在測試環境中表現良好，那麼它們也將在生產環境中表現良好。該範本將測試環境與生產環境配置不同的風險降至最低。

此外，如果必須在測試環境中進行配置更新，則可以通過更改更新範本並更新所有堆疊。該過程有助於確保對單一環境的修改可靠地傳播到應接收更新的所有環境。

另一個好處是，當不再需要時，可以更輕鬆地*清理*在帳戶中創建的所有資源以支持測試環境。這有助於降低與您不再需要的資源相關的成本，並使您的帳戶保持簡單。

AWS CloudFormation 範本語法

aws academy

AWS CloudFormation 範本

- 使用 JavaScript Object Notation (JSON) 或 YAML Ain't Markup Language (YAML) 編寫
- YAML 的優勢 –
  - 不那麼冗長（沒有 {}、"" 字元）
  - 支持嵌入式備註
- JSON 的優勢 –
  - 在其他電腦系統（例如 API）中得到了更廣泛的應用
- 推薦 – 將範本視為源代碼
  - 將它們存儲在代碼存儲庫中

```
{
 "AWSTemplateFormatVersion": "2010-09-09",
 "Resources" : {
 "awsexamplebucket1" : {
 "Type" : "AWS::S3::Bucket"
 }
 }
}
```

JSON 示例

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
 awsexamplebucket1:
 Type: AWS::S3::Bucket
```

YAML 示例

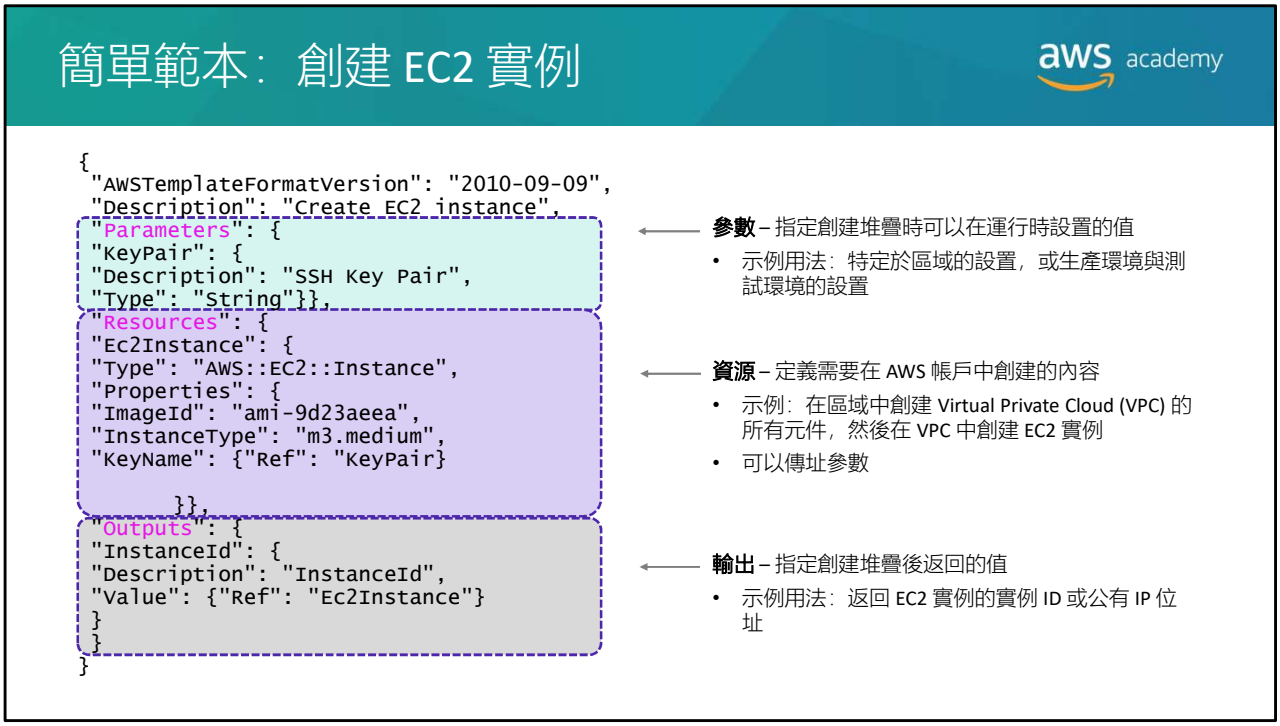
範本也可以在 AWS CloudFormation Designer（AWS 管理主控台中的圖形設計介面）中創建。

AWS CloudFormation 範本可以使用 JavaScript Object Notation (JSON) 或 YAML Ain't Markup Language (YAML) 編寫。

YAML 在可讀性方面進行了優化。同樣的資料以 YAML 格式存儲要比以 JSON 格式存儲所用的行數更少，因為 YAML 並不使用大括弧 ({})，而且使用的引號 (“”) 也更少。YAML 的另一個優點是，它原生支援嵌入式備註。與 JSON 相比，調試 YAML 文檔也更加輕鬆。使用 JSON 時，可能很難找到丟失或放錯位置的逗號或大括弧。而 YAML 不存在這個問題。

儘管 YAML 擁有很多優點，但 *JSON* 也具有一些獨特的優點。首先，它在電腦系統中得到了廣泛應用。它的普及率是一個優勢，因為以 JSON 格式存儲的資料能夠可靠地與很多系統結合使用，無需進行轉換。此外，生成和解析 JSON 通常要比生成生成和解析 YAML 更輕鬆。

AWS 管理主控台提供了一個名為 *AWS* CloudFormation Designer 的圖形介面，可用於編寫或查看 AWS CloudFormation 範本的內容。它還可以用來將有效的 JSON 範本轉換為 YAML 或將 YAML 轉換為 JSON。它提供了一個用來編寫範本的拖放介面，這些範本可以 JSON 或 YAML 格式輸出。

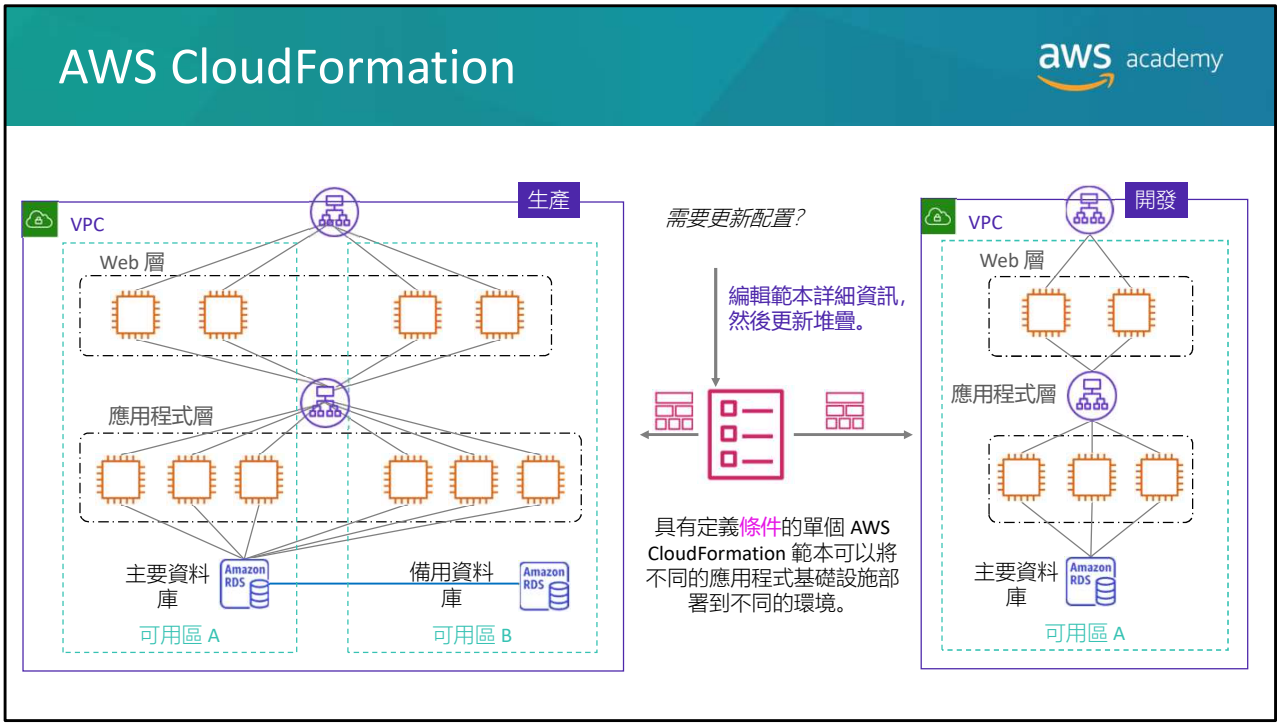


此示例 AWS CloudFormation 範本創建了一個 EC2 實例。儘管此示例並未說明範本的所有可能部分，但它確實突出了一些最常用的部分，包括參數、資源和輸出。

Parameters（參數）是範本的一個可選部分。參數是在運行時（創建或更新堆疊時）傳遞給範本的值。您可以從範本的 **Resources**（資源）和**Outputs**（輸出）部分傳址參數。當用戶在控制台中啟動 *Create Stack*（創建堆疊）嚮導時，參數的名稱和描述會出現在 *Specify Parameters*（指定參數）頁面中。

Resources（資源）是任何範本的必填部分。使用它來指定要創建的 AWS 資源及其屬性。在此示例中，指定了 *AWS::EC2::Instance* 類型的資源，用它來創建 EC2 實例。示例資源包括靜態定義的屬性（ImageId *和* InstanceType*）和引用的* KeyPair *參數。*

最後，該示例顯示了 **Outputs**（輸出）部分。**Outputs**（輸出）描述了您查看堆疊的屬性時返回的值。在示例中，已聲明輸出為 InstanceId。創建堆疊後，可在 AWS CloudFormation 控制台的堆疊詳細資訊中看到此值，方法是運行 *aws cloudformation describe-stacks* AWS 命令列介面 (AWS CLI) 命令，或使用 AWS 軟體開發套件 (SDK) 檢索此值。



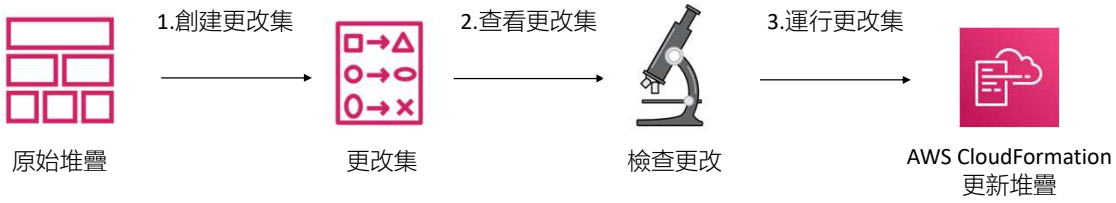
您可以使用同一 AWS CloudFormation 範本創建生產環境和開發環境。這種方法可確保（例如）在開發和生產中使用相同的應用程式二進位檔案、相同的 Java 版本和相同的資料庫版本。因此，範本可確保應用程式在生產環境中的行為方式與其在開發環境中的行為方式相同。

在示例中，您可以看到生產環境和開發環境是從同一範本創建的。但生產環境配置為跨兩個可用區運行，而開發環境在單個可用區中運行。這些特定於部署的差異可使用*條件*來實現。您可以在 AWS CloudFormation 範本中使用*條件*語句來確保開發、測試和生產環境的配置相同，即使它們的規模和範圍有所不同。

您可能需要多個測試環境，用於進行功能測試、使用者接受度測試和負載測試。手動創建這些環境存在風險。但使用 AWS CloudFormation 創建它們有助於確保一致性和可重複性。

AWS CloudFormation 更改集

更改集允許您在實施更改之前預覽更改。



使用 DeletionPolicy 屬性可以在資源堆疊刪除或更新時保留或備份資源。

更新堆疊（從而更新 AWS 資源）的一種方法是更新您用於創建堆疊的 AWS CloudFormation 範本，然後運行**更新堆疊**選項。

但是，在實際運行更新之前，您可能希望進一步瞭解 AWS CloudFormation 在您運行該命令時將實施的特定更改。如果想要進一步瞭解，可使用*更改集*。

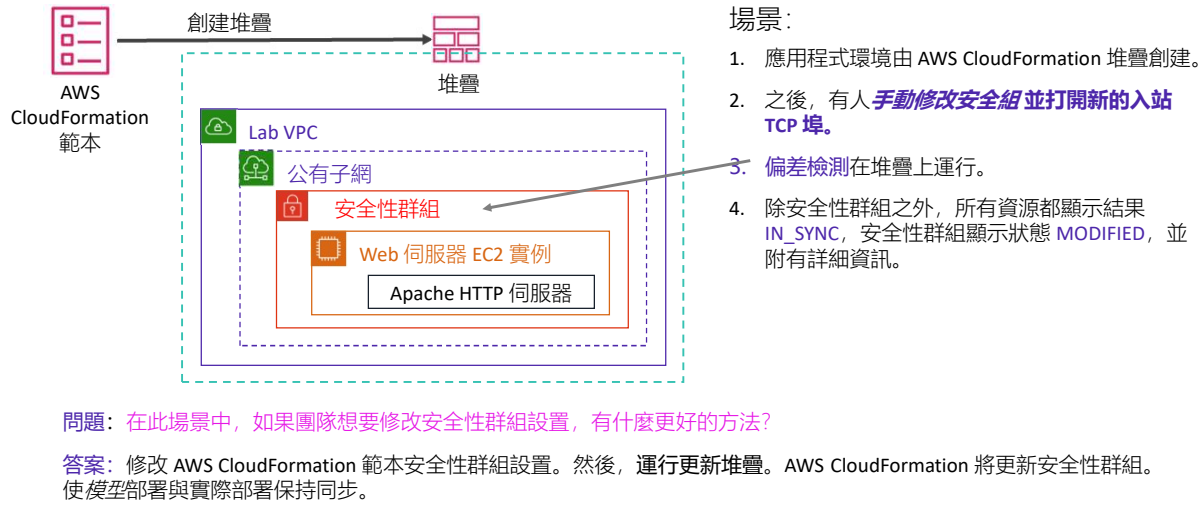
更改集使您可以預覽更改，驗證它們是否符合您的期望，批准更新之後再繼續。

遵循以下基本工作流程使用 AWS CloudFormation 更改集：

1. **創建更改集**：通過為您要更新的堆疊提交更改來實現。
2. **查看更改集**，瞭解要更改哪些堆疊設置和資源。如果您希望在決定所要進行的更改之前考慮其他更改，請創建其他更改集。
3. **運行更改集**。AWS CloudFormation 會使用這些更改來更新堆疊。

如果您使用更改集，則可能需要對某些資源設置刪除策略。DeletionPolicy *屬性可用于在刪除或更新資源堆疊時保留（或在某些情況下備份）資源。*如果某個資源不具有 DeletionPolicy 屬性，AWS CloudFormation 將刪除該資源。

偏差檢測



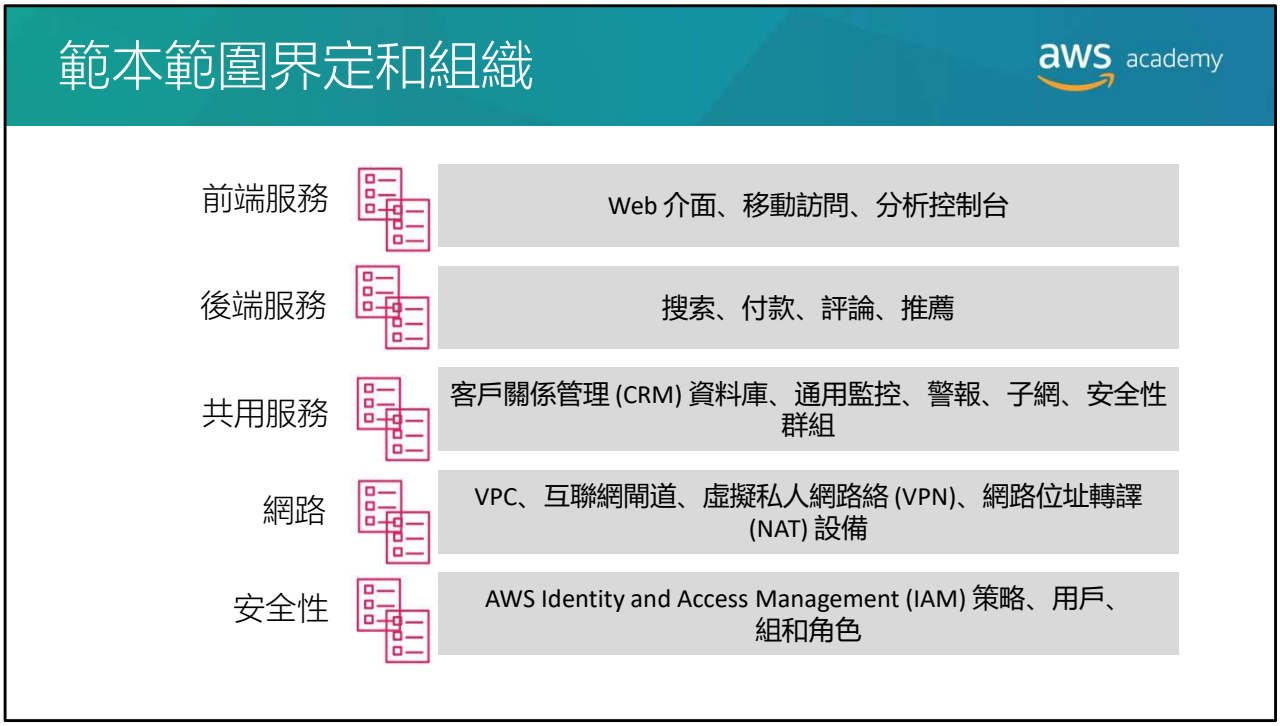
請考慮以下情況：通過運行 AWS CloudFormation 堆疊來創建應用程式環境。然後，某人決定手動修改部署的環境設置。他們在堆疊創建的安全性群組中創建新的入站規則。但他們是在 AWS CloudFormation 環境外部進行的更改，例如，使用 Amazon EC2 控制台。作為此應用程式的架構師，您想知道您*部署的*環境是否不再匹配 AWS CloudFormation 範本中定義的*模型*環境。

如何知道哪些資源已經過修改，不再完全符合堆疊的規範？

可從控制台中的 **Stack actions**（堆疊操作）功能表選擇**Detect Drift**（檢測偏差）以對堆疊運行偏差檢測。對堆疊執行偏差檢測來確定堆疊是否*偏離*預期的範本配置。偏差檢測將返回有關在堆疊中支援偏差檢測的每個資源的偏差狀態的詳細資訊。

當您刪除存在偏差的堆疊時，AWS CloudFormation 的資源清理過程不會處理該偏差。如果堆疊具有未解決的資源依賴項，則可能會導致堆疊刪除操作失敗。在這種情況下，可能需要手動解決問題。有關詳細資訊，請參閱支持偏差檢測的資源列表。

範本範圍界定和組織



隨著您的組織使用的 AWS CloudFormation 範本越來越多，制定範本策略對您來說非常重要。該策略將定義單個範本應創建的範圍，以及使您想要在多個範本中定義 AWS 基礎設施的一般特徵。

該圖提供了一些關於如何整理範本以便維護，並以合理的方式相互組合使用的想法。一個好的策略是將資源定義分組到範本中，類似于將大型企業應用程式的功能整理到不同部分的方式。

考慮基礎設施中連接更為緊密的元件，將它們放在相同的範本中。在此示例中，AWS CloudFormation 範本用於在以下五個領域之一創建和維護 AWS 資源：前端服務、後端服務、共用服務、網路和安全性。在每個領域，您都可以維護一個範本，其作用于單個應用程式或單個部門的需求。

無論您如何整理和限定每個 AWS CloudFormation 範本，都要將範本視為需要版本控制的代碼。將範本存儲在原始程式碼控制系統中。

AWS Quick Start

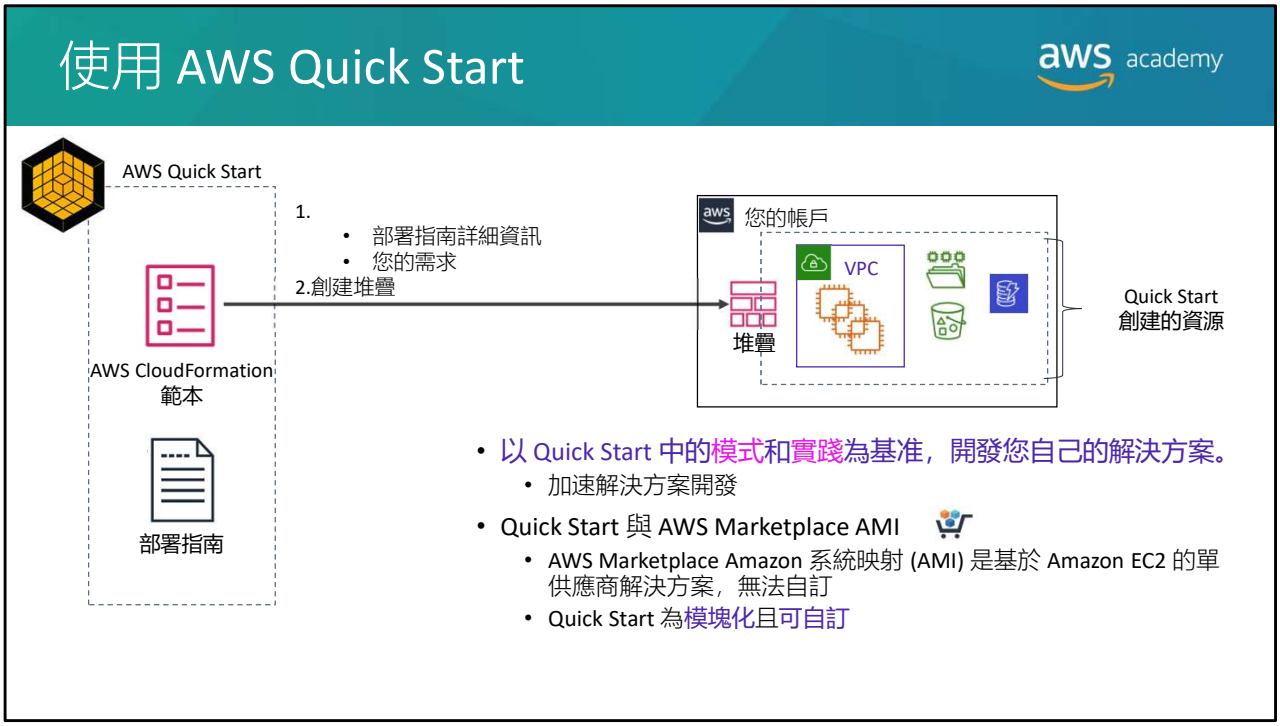
AWS Quick Start

由 AWS 解決方案架構師構建的 AWS CloudFormation 範本

- 黃金標準部署方案
- 遵循安全性和高可用性方面的 AWS 最佳實踐
- 一鍵即可在一小時內創建整個架構
- 可用于實驗，也可以作為您自己的架構的基礎

AWS Quick Start 提供了 AWS CloudFormation 範本。Quick Start 由 AWS 解決方案架構師和合作夥伴構建，旨在根據 AWS 的安全性和高可用性最佳實踐，在 AWS 上部署熱門解決方案。這些參考部署可在不到一個小時內在您的 AWS 帳戶中預置，通常只需幾分鐘即可完成。它們有助於您通過幾個步驟構建架構完善的測試或生產環境。您可以用它們創建整個架構，也可以用它們來試驗新的部署方法。

有關詳細資訊，請轉至 AWS Quick Start 頁面。



每個 Quick Start 均包含一個 AWS CloudFormation 範本和一個部署指南。該指南提供了有關部署選項以及如何配置部署以滿足您的需求的詳細資訊。

自訂部署以滿足您的需求並創建堆疊。根據必須創建的 AWS 資源，Quick Start 將在幾分鐘或幾小時內完成部署。

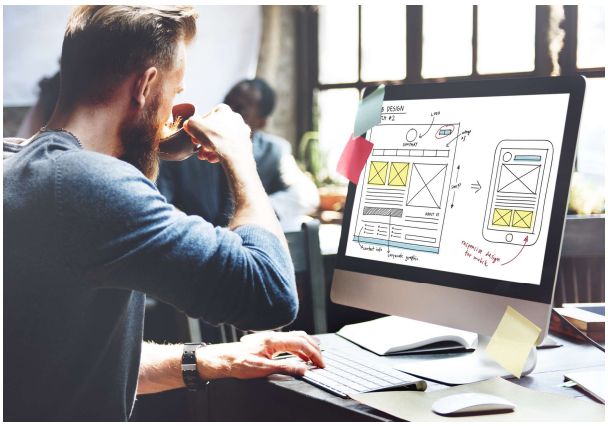
即使不使用 Quick Start，瞭解 Quick Start 遵循的*模式*和*實踐*類型也很有幫助。如果您借鑒 Quick Start 的某個部分並將其嵌入自己的範本，可能有助於加速範本開發。

AWS Marketplace Amazon 系統映射 (AMI) 是 人們偶爾會使用的另一種解決方案。可從 Amazon EC2 控制台啟動這些資源。AWS Marketplace AMI 提供在 EC2 實例上運行的單一供應商解決方案。相比之下，AWS Quick Start 是模組化的、更可定制的解決方案，可能使用（也可能不使用）Amazon EC2。

演示：分析 AWS CloudFormation 範本結構並創建堆疊

講師可能選擇演示 AWS CloudFormation 範本的結構，然後使用範本創建 AWS CloudFormation 堆棧。

模組 10 – 指導實驗：使用 AWS CloudFormation 實現自動化基礎設施部署



您現在將完成“模組 10 – 指導實驗：使用 AWS CloudFormation 實現自動化基礎設施部署”。

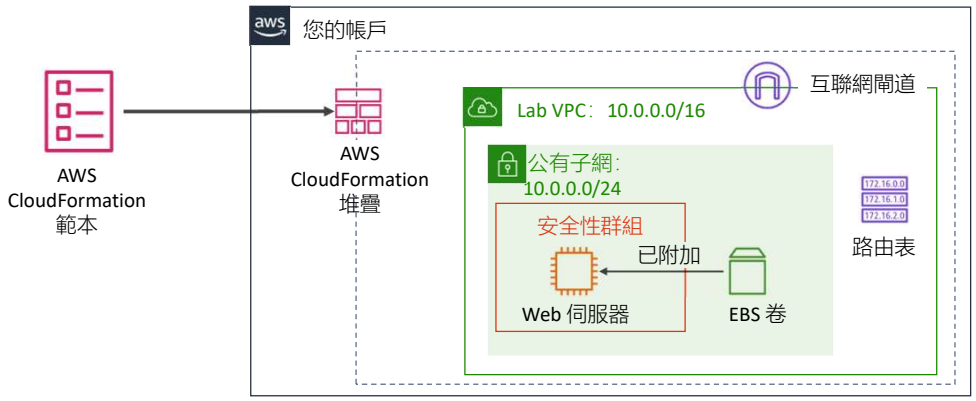
指導實驗：任務

1. 部署網路層
2. 部署應用程式層
3. 更新堆疊
4. 使用 AWS CloudFormation Designer 流覽範本
5. 刪除堆疊

在本指導實驗中，您將完成以下任務：

1. 部署網路層
2. 部署應用程式層
3. 更新堆疊
4. 使用 AWS CloudFormation Designer 流覽範本
5. 刪除堆疊

指導實驗：最終產品



在本指導實驗結束時，您將使用 AWS CloudFormation 在圖中創建資源。網路層中的資源是您在創建第一個堆疊時創建的。EC2 實例、安全性群組和 Amazon Elastic Block Store (Amazon EBS) 卷是您在創建第二個堆疊時創建的。然後，在用於創建第二個堆疊的範本中更新安全性群組設置。當您運行更新堆疊操作時，此修改將應用于安全性群組資源。

現在可以開始指導實驗了。

指導實驗總結：要點

完成這個指導實驗之後，您的講師可能會帶您討論此指導實驗的要點。

第 3 節要點

aws academy

- AWS CloudFormation 是一項基礎設施即代碼 (IaC) 服務，使用此服務，您可以建模、創建和管理 AWS 資源的集合
- AWS CloudFormation IaC 在使用 JSON 或 YAML 編寫的範本中定義
- 堆疊是您在使用範本創建 AWS 資源時創建的
- 可對現有堆疊執行的操作包括更新堆疊、檢測偏差和刪除堆疊
- AWS Quick Start 提供了解決方案架構師構建的反映 AWS 最佳實踐的 AWS CloudFormation 範本

本模組中這節內容的要點包括：

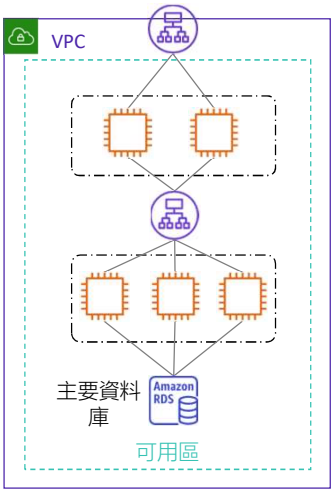
- AWS CloudFormation 是一項基礎設施即代碼 (IaC) 服務，使用此服務，您可以建模、創建和管理 AWS 資源的集合
- AWS CloudFormation IaC 在使用 JSON 或 YAML 編寫的範本中定義
- 堆棧是您在使用範本創建 AWS 資源時創建的
- 可對現有堆疊執行的操作包括更新堆疊、檢測偏差和刪除堆疊
- AWS Quick Start 提供了解決方案架構師構建的反映 AWS 最佳實踐的 AWS CloudFormation 範本

模組 10：實現架構自動化

第 4 節：實現部署自動化

介紹第 4 節：實現部署自動化。

如何保持佇列更新?



您可能要管理數百個實例。
如何應用訪客作業系統 (OS) 補丁並更新安裝在其上的軟體?

在上一節中，您學習了如何使用 AWS CloudFormation 自動創建整個基礎設施。其功能非常強大，但仍有些重要問題需要解決。

如何更新 EC2 實例佇列中的軟體？您是否要自行遠端登入每個實例並運行更新命令？如果出現錯誤，如何還原更改？如果您有成百上千台運行許多不同應用程式的伺服器怎麼辦？

傳統工具可以說明應對這些情況，但現成的解決方案會更加方便。

AWS Systems Manager

瞭解運行狀況並對 AWS 資源執行操作。

- 自動執行操作任務
  - 示例：在 EC2 實例佇列中應用作業系統補丁和軟體升級
- 簡化資源和應用程式管理
  - 管理軟體清單
  - 查看整個佇列的詳細系統組態
- 在本地和雲中管理伺服器

即使您使用像 AWS CloudFormation 這樣的 IaC 工具來創建和維護 AWS 資源部署，擁有其他可用的工具也很有幫助。例如，這些工具可以滿足環境對配置管理的持續需求。這些需求可能發生在預置基礎設施資源之後和基礎設施啟動並運行之後。AWS Systems Manager 服務可解決這一難題。

AWS Systems Manager 是一項管理服務，高度專注於自動化。它允許配置和管理在本地或 AWS 中運行的系統。AWS Systems Manager 使您能夠識別要管理的實例，然後定義要對這些實例執行的管理任務。您可以免費使用 AWS Systems Manager 來管理您的 Amazon EC2 和本地資源。

您可以使用 AWS Systems Manager 完成的一些任務包括：

- 收集軟體清單
- 應用作業系統 (OS) 補丁
- 創建系統映射
- 配置 Microsoft Windows 和 Linux 作業系統

這些功能可幫助您定義和跟蹤系統組態，防止偏差，並確保 Amazon EC2 和本地配置的軟體合規性。

本示例說明了如何使用 Systems Manager 來更新、管理和配置 EC2 實例佇列。

您可以在 EC2 實例上，甚至在本機伺服器或虛擬機器 (VM) 上安裝 *AWS Systems Manager 代理（SSM 代理）*。在安裝 SSM 代理後，Systems Manager 就可以更新、管理和配置它所安裝到的伺服器。代理處理來自 Systems Manager 的請求，然後根據請求中提供的規範運行它們。然後，代理將狀態和相關資訊發回 Systems Manager。

預設情況下，大多數 Microsoft Windows Server AMI、所有 Amazon Linux 和 Amazon Linux 2 AMI 以及某些 Ubuntu AMI 上都會預先安裝 SSM 代理。但您必須在從其他 Linux AMI 創建的 EC2 實例上手動安裝代理。有關完整詳細資訊，請參閱使用 SSM 代理 AWS 文檔。

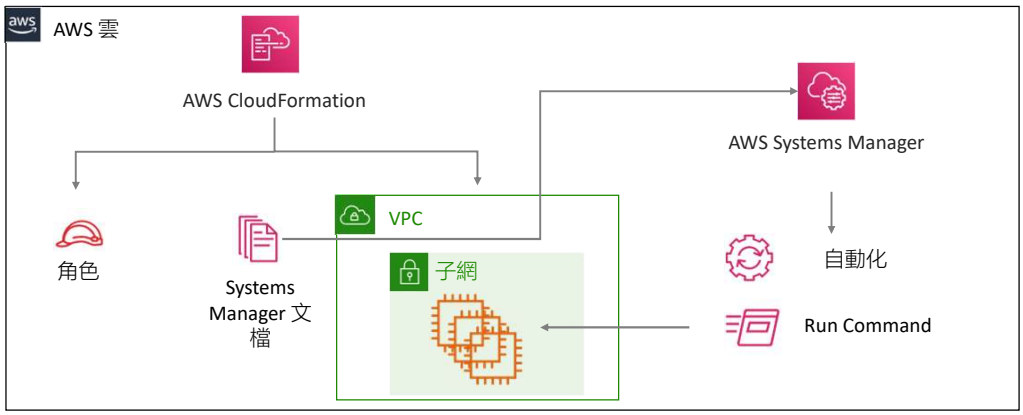
AWS Systems Manager 提供了各種工具：

- **Run Command** 使您能夠以遠端方式安全地管理託管實例的配置。命令可以在沒有 Secure Shell (SSH) 或遠端桌面協議 (RDP) 訪問的情況下運行，因此您可以使用它們來減少對堡壘主機的需求。運行 Bash、PowerShell、Salt 或 Ansible 腳本。
- **維護時段**使您能夠制定計劃，對您的實例執行潛在的破壞性操作。示例包括修補作業系統、更新驅動程式或安裝軟體或補丁。
- **參數倉庫**可提供安全存儲，用於配置資料和金鑰管理。例如，您可以將密碼、資料庫字串和許可證代碼存儲為參數值。
- **補丁管理器**可自動執行通過安全相關的更新以及其他類型的更新修補託管實例的流程。
- **狀態管理器**可自動執行將您的 Amazon EC2 和混合基礎設施保持在您定義的狀態的過程。
- **Automation** 使您能夠構建自動化工作流程以配置和管理實例和 AWS 資源。

- **Session Manager** 使您能夠通過基於互動式流覽器的 shell 管理 EC2 實例。
- **Inventory** 使您可以瞭解 Amazon EC2 和本地計算環境。您可以使用 Inventory 功能從託管實例收集中繼資料。
- **文檔**定義了 Systems Manager 對您的託管實例執行的操作。您可以通過在運行時指定參數來使用十多個預先配置的文檔。您還可以定義 JSON 或 YAML 格式的文檔，並指定步驟和參數。

AWS CloudFormation 和 Systems Manager 相互補充

AWS CloudFormation 適用於定義 AWS 雲資源。
Systems Manager 適用於在訪客作業系統內實現自動化。



現在，您已瞭解 AWS CloudFormation 和 AWS Systems Manager 的功能，請考慮這兩種服務如何互補。

Systems Manager 適用於在訪客作業系統*內*實現自動化。相比之下，AWS CloudFormation 適用于定義 AWS 雲資源。

您可以在 AWS 雲層使用 AWS CloudFormation 來定義 AWS 資源。然後，如圖所示，您可以使用 AWS Systems Manager 配置由 AWS CloudFormation 堆疊創建的實例的作業系統。

通過使用 AWS CloudFormation 維護雲資源，您可以保持部署堆疊，然後從單個範本中刪除堆疊的能力。Systems Manager 則為您提供了一種執行持續任務的方法，例如，通過補丁更新來更新 EC2 實例訪客作業系統，將日誌集中聚合到 Amazon CloudWatch。

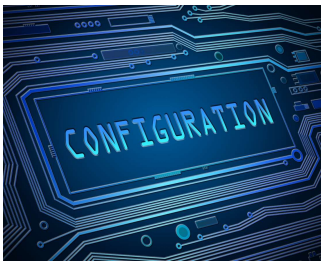
有關將這兩項服務結合使用的示例解決方案的更多資訊，請參閱 Using AWS Systems Manager Automation and AWS CloudFormation together 博客文章。

AWS OpsWorks

AWS OpsWorks

AWS OpsWorks 是一項配置管理服務。

- 實現伺服器配置、部署和管理方式的自動化
- 提供 Chef 和 Puppet 的託管實例
  - Chef 和 Puppet 是常用的自動化平臺
- 三個版本可供選擇 –
  - AWS OpsWorks for Chef Automate
  - AWS OpsWorks for Puppet Enterprise
  - AWS OpsWorks Stacks



AWS OpsWorks 是一項配置管理服務。您可以使用 OpsWorks 來自動配置、部署和管理 EC2 實例。

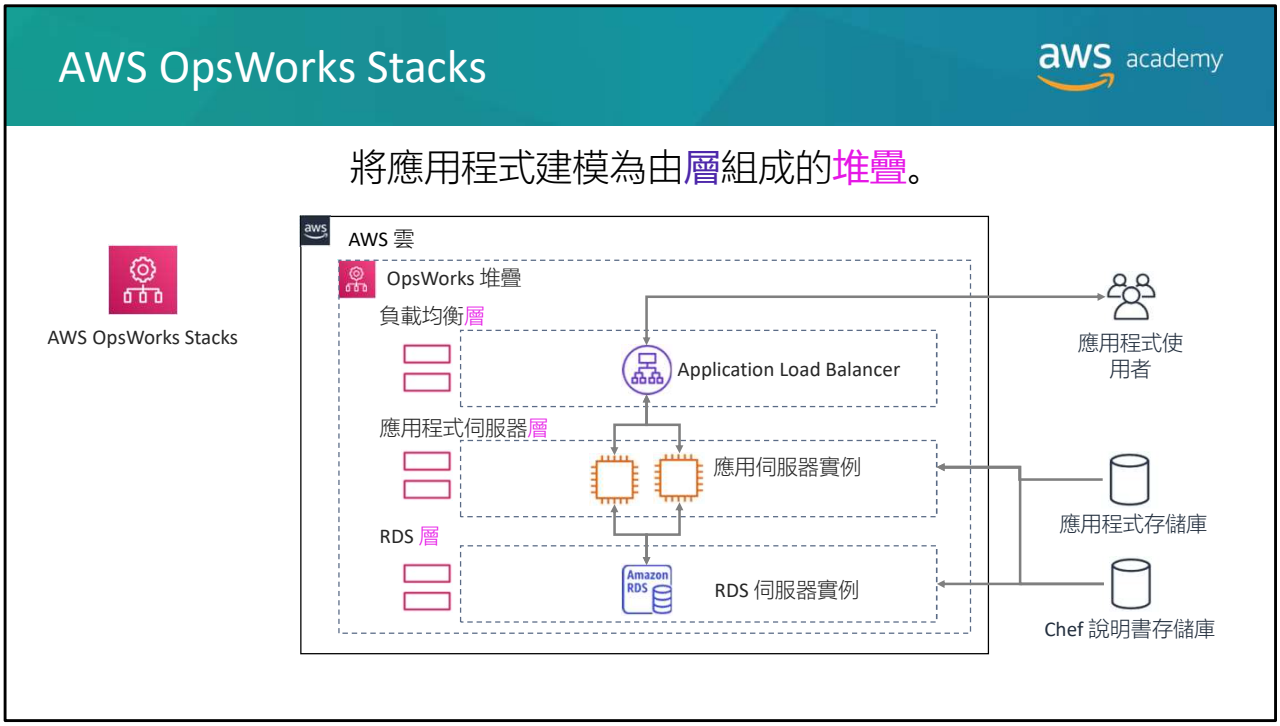
AWS OpsWorks 有三種不同的版本：

- **AWS OpsWorks for Chef Automate** 提供了一個完全託管的 Chef Automate 伺服器，該伺服器針對持續部署提供了工作流程自動化，並針對合規性和安全性提供了自動化測試。Chef Automate 平臺可以處理軟體與作業系統配置、持續合規性、套裝軟體安裝和資料庫設置等多種操作任務。您可以使用 Chef Automate 來創建和管理基於 AWS 雲運行的動態基礎設施。Chef Automate 伺服器通過告知 Chef 用戶端在節點上運行哪些 Chef 配方來管理環境中節點的配置。它還會存儲有關節點的資訊，並作為 Chef 說明書的中央存儲庫。

- **AWS OpsWorks for Puppet Enterprise** 提供了一個託管的 Puppet Enterprise 伺服器和一套自動化工具，允許針對編排、自動預置和可追蹤性視覺化實現工作流程自動化。借助 Puppet Enterprise，您能夠以支持維護和版本控制（類似應用程式原始程式碼）的方式定義伺服器的配置。主 Puppet 伺服器旨在一致地配置和維護您的其他 Puppet 伺服器（或節點）。您也可以根據其他節點的狀態動態配置您的節點。

- **AWS OpsWorks Stacks** 是一項配置管理服務，可說明您利用 Chef 配置和操作各種類型和規模的應用程式。您可以定義應用程式的架構和每個元件的規格，包括套裝軟體安裝、軟體配置和存儲等資源。

AWS OpsWorks Stacks



該示例演示了如何使用 AWS OpsWorks Stacks 管理基本應用程式。OpsWorks Stacks 應用程式中的基本創建單位是*堆棧*。

創建堆疊後，您可以向該堆疊添加多個*層*。因此，您可以將應用程式構建為一組相關功能的交互層。

在這種情況下，一組應用程式伺服器在*應用程式伺服器層*中運行。這些應用程式在*負載均衡層*中定義的 Elastic Load Balancing 負載等化器之後運行。該示例還包括在 *RDS 層*中定義的後端 Amazon Relational Database Service (Amazon RDS) 資料庫伺服器。

層依靠 Chef 配方來處理諸如在實例上安裝套裝軟體、部署應用程式、運行腳本等任務。OpsWorks Stacks 使用 Chef 說明書處理安裝和配置套裝軟體和部署應用程式等任務。 您的自訂說明書必須存儲在線上存儲庫中，它或者是一個存檔（如 .zip 檔），或者是一個原始程式碼控制管理器（如 Git）。

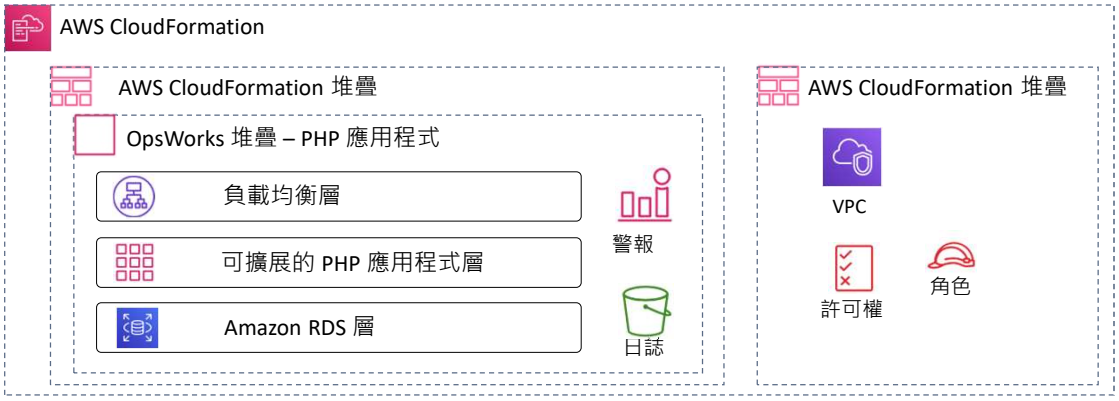
OpsWorks Stacks 的一項重要功能是一組*生命週期事件*（包括設置、配置、部署、取消部署和關閉），這些事件會在適當的時間自動在每個實例上運行一組指定的配方。

每個層都可以有一組分配給每個生命週期事件的配方。這些配方可處理該事件和層的各種任務。

OpsWorks Stacks 是 AWS CloudFormation 的補充

1.使用 AWS CloudFormation 創建基礎設施（VPC、IAM 角色等）。

2.使用 OpsWorks Stacks 部署應用程式層。



由於可通過 AWS CloudFormation 創建 OpsWorks Stacks，因此這兩種技術可以互補使用。

例如，您可以使用一個 AWS CloudFormation 範本為您的環境創建 AWS 資源基礎設施，包括 VPC。然後，您可以使用另一個 AWS CloudFormation 範本創建將在該 VPC 中部署的 OpsWorks 堆疊。在您的帳戶中創建兩個 AWS CloudFormation 堆疊之後，您可以使用 OpsWorks 堆疊來管理應用程式。

模組 10：實現架構自動化

第 5 節：AWS Elastic Beanstalk

介紹第 5 節：AWS Elastic Beanstalk。

一般性挑戰

圍繞應用程式部署管理基礎基礎設施可能*非常困難*

管理和配置伺服器可能*非常耗時*

多個專案或應用程式之間可能*缺乏一致性*

現在，考慮在著手管理雲基礎設施時可能會遇到的一些一般性挑戰。

首先，部署應用程式可能*非常困難*。如何確保它高度可用，即使在高峰使用期間也能支持用戶請求？如何確保應用程式具有彈性，並出於災難恢復 (DR) 目的定期備份？管理和配置伺服器可能*非常耗時*。同時，您希望在專案和應用程式之間保持*一致性*，但這種一致性可能很難實現。

AWS Elastic Beanstalk

AWS Elastic Beanstalk

- 啟動並運行 Web 應用程式的簡單方式
- 可自動處理以下任務的託管服務 –
  - 基礎設施預置和配置
  - 部署
  - 負載均衡
  - 自動擴展
  - 運行狀況監控
  - 分析和調試
  - 日志記錄
- 無需額外付費即可使用
  - 只需為您使用的底層資源付費

AWS Elastic Beanstalk 是另一種 AWS 計算服務選項。它是一種平臺即服務 (PaaS) 產品，有助於快速部署、擴展和管理您的 Web 應用程式和服務。它能夠解決前面提到的許多難題。

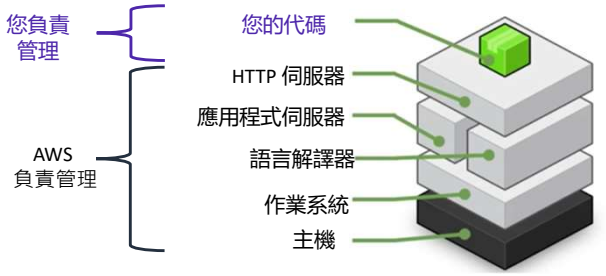
借助 Elastic Beanstalk，您可以保持對代碼的控制，由 AWS 維護底層基礎設施。使用 AWS 管理主控台中的簡單嚮導創建和部署所需的 AWS 資源。該嚮導要求您選擇實例類型和大小、資料庫類型和大小以及要使用的自動擴展設置。它允許您訪問伺服器日誌檔，並在負載等化器上啟用安全 HTTP (HTTPS)。

在您上傳代碼後，Elastic Beanstalk 將自動處理包括容量預置、負載均衡、自動擴展和應用程式運行狀況監控在內的部署工作。同時，您保留了對支援您的應用程式的 AWS 資源的完全控制，並且可以隨時訪問底層資源。

AWS Elastic Beanstalk 不收取額外費用。您只需為您創建的用於存儲和運行應用程式的 AWS 資源付費，例如 EC2 實例或 S3 存儲桶。您只需在使用時為您的實際用量支付費用。

AWS Elastic Beanstalk 部署

- 它支援為通用平臺編寫的 Web 應用程式
  - Java、.NET、PHP、Node.js、Python、Ruby、Go 和 Docker
- 上傳您的代碼
  - Elastic Beanstalk 將自動處理部署
  - 部署在 Apache、NGINX、Passenger、Puma 和 Microsoft Internet 資訊服務 (IIS) 等伺服器上

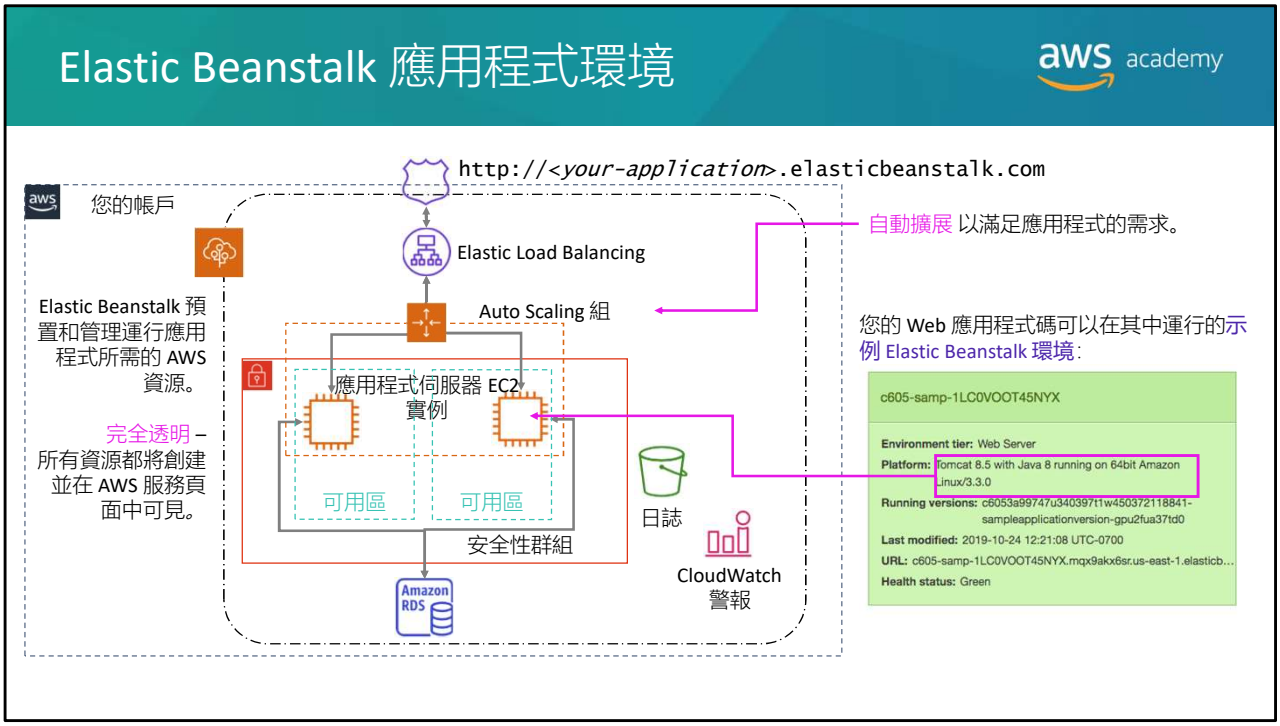


Elastic Beanstalk 會使用在選定平臺上運行應用程式所需的必要元件來配置您環境中的每個 EC2 實例。您無需負責登錄實例來安裝和配置應用程式堆疊。

您只需創建代碼。Elastic Beanstalk 旨在使您的應用程式部署變得快速而簡單。它支持一系列平臺，包括 **Docker**、**Go**、**Java**、**.NET**、**Node.js**、**PHP**、**Python** 和 **Ruby**。

AWS Elastic Beanstalk 將您的代碼部署在：

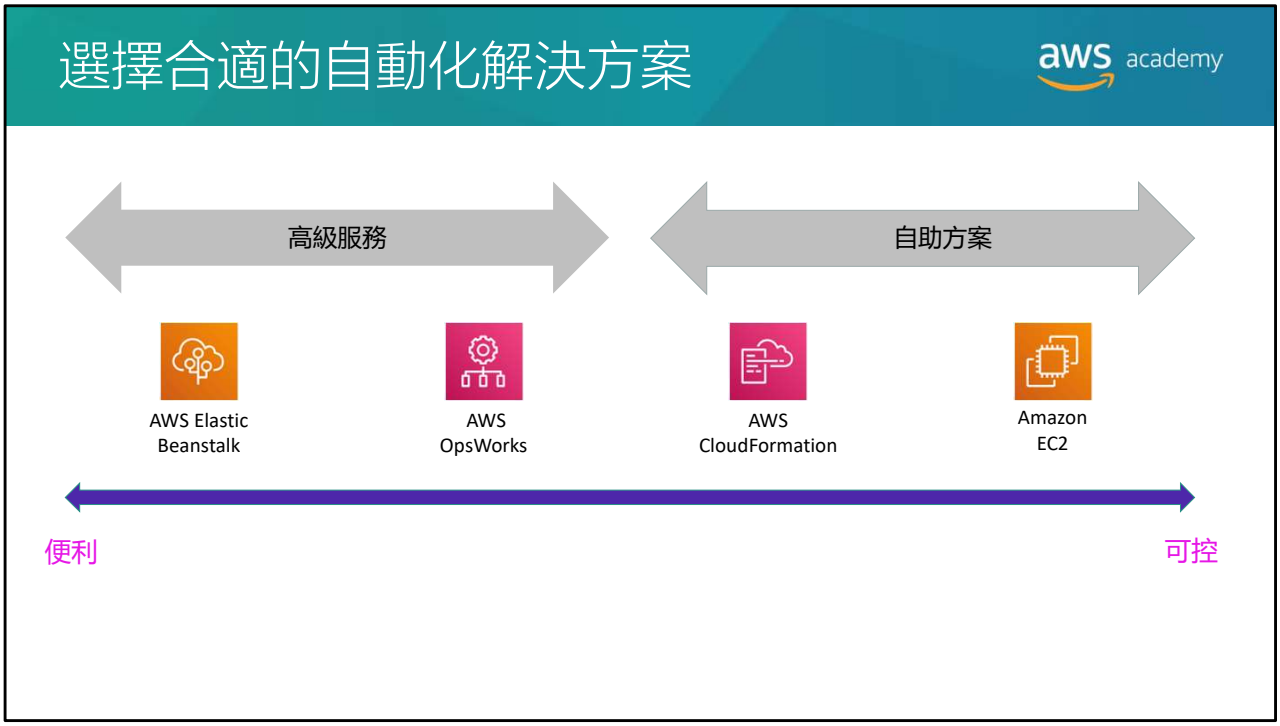
- **Apache Tomcat**，適用于 Java 應用程式
- **Apache HTTP Server**，適用于 PHP 和 Python 應用程式
- **NGINX** 或 **Apache HTTP Server**，適用於 Node.js 應用程式
- **Passenger** 或 **Puma**，適用于 Ruby 應用程式
- **Microsoft Internet Information Services (IIS)**，適用於 .NET、Java SE、Docker 和 Go 應用程式。



使用 Elastic Beanstalk 時，您可以從兩種類型的環境中選擇。單實例環境*允許您啟動單個 EC2 實例，且不包含負載均衡或自動擴縮。*另一種類型的環境（本例所示）可以啟動*多個 EC2 實例*，且包含負載均衡和自動擴縮配置。託管資料庫層是可選的。

由 Elastic Beanstalk 創建的 AWS 資源會顯示在您的 AWS 帳戶中。例如，創建 Elastic Beanstalk 應用程式後，打開 AWS 管理主控台，然後打開 Amazon EC2 控制台。您將看到 Elastic Beanstalk 代表您管理的實例。在本例中，我們創建了兩個 EC2 實例。每個 EC2 實例都運行 Amazon Linux 訪客作業系統，並安裝了 Java 8 和 Apache Tomcat 8.5 Web 伺服器。您的應用程式碼將在這些伺服器上運行。配置了自動擴縮功能，因此，如果負載開始使這兩個實例上的資源緊張（例如，過高的 CPU 利用率持續 5 分鐘以上），則系統會自動啟動更多的應用程式伺服器實例。本示例還顯示了 RDS 資料庫實例可用，並可從 EC2 實例訪問。您可以將應用程式資料存儲在資料庫中，並在應用程式碼中使用結構化查詢語言 (SQL) 來訪問和更新這些資料。Elastic Beanstalk 管理資料庫實例，並說明維護 EC2 實例和資料庫之間的連接。

Elastic Beanstalk 創建和管理可擴展的環境。您可以將 Auto Scaling 組配置為自動擴展應用程式，以處理大量流量負載。它還為您的應用程式環境提供唯一的功能變數名稱。URL 語法是 *<your-application>*.elasticbeanstalk.com。您也可以使用 Amazon Route 53 將自己的功能變數名稱解析為提供的功能變數名稱。



本模組至少向您介紹了四種 AWS 服務。一個常見問題是，關於提供應用程式管理功能的多種服務，它們之間的界線是什麼，或者哪種服務應在什麼情況下使用？您的決定應取決於所需的相對便利和控制水準。

Elastic Beanstalk 是一項易於使用的應用程式服務，用於構建在 Java、PHP、Node.js、Python、Ruby 或 Docker 上運行的 Web 應用程式。如果您希望無需自定義環境即可上傳代碼，那麼 Elastic Beanstalk 可能是一個不錯的選擇。

OpsWorks 讓您可以啟動應用程式並定義其架構和每個元件的規格，包括套裝軟體安裝、軟體配置和資源（比如存儲）。您可以使用用於常見技術的範本（應用程式伺服器、資料庫等），也可以構建自己的範本。

相較編寫和維護 AWS CloudFormation 範本以創建堆疊或直接管理 EC2 實例，Elastic Beanstalk 和 OpsWorks 提供了更高級別的服務。但正確選擇要使用的服務（或服務組合）取決於您的需求。這些工具都可供您使用。作為架構師，您必須確定哪些服務最適合您的使用案例。

第 5 節要點

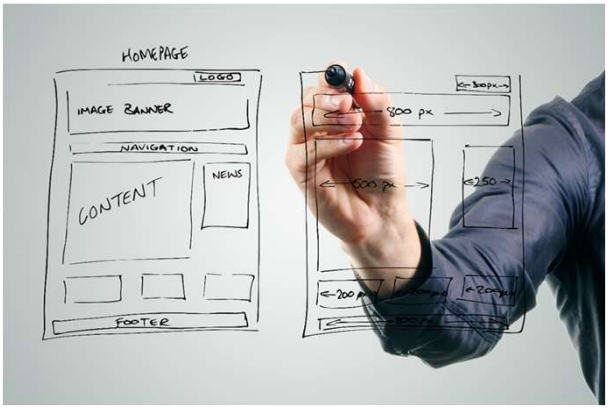
aws academy

- Elastic Beanstalk 可創建和管理可擴縮且高度可用的 Web 應用程式環境，使您能夠專注于應用程式碼
- 您可以使用 Java、.NET、PHP、Node.js、Python、Ruby、Go 或 Docker 編寫 Elastic Beanstalk 應用程式碼
- 由 Elastic Beanstalk 創建的 AWS 資源是完全透明的 – 在 AWS 管理主控台服務資料頁檢視中可見
- Elastic Beanstalk 不收取額外費用 – 您只需為使用的底層資源付費

本模組中這節內容的要點包括：

- AWS Elastic Beanstalk 用於創建和管理可擴展且高度可用的 Web 應用程式環境，使您能夠專注于應用程式碼
- 您可以使用 Java、.NET、PHP、Node.js、Python、Ruby、Go 或 Docker 編寫 Elastic Beanstalk 應用程式碼
- 由 Elastic Beanstalk 創建的 AWS 資源是完全透明的 – 在 AWS 管理主控台服務資料頁檢視中可見
- Elastic Beanstalk 不收取額外費用 – 您只需為使用的底層資源付費

模組 10 – 挑戰實驗：實現基礎設施部署自動化



您現在將完成“模組 10 – 挑戰實驗：實現基礎設施部署自動化”。

業務需求：實施 IaC

- 咖啡館現在在多個國家/地區設有分店，必須開始自動化才能保持發展。
- 他們需要一種方法，能夠跨多個 AWS 服務一致地部署、管理和更新咖啡館資源。
- 他們希望能夠跨 AWS 區域可靠地創建可重複的環境，以滿足開發和生產需求。

長期以來，咖啡館一直在創建 AWS 資源並手動配置應用程式。這種方法可以有效說明咖啡館快速開發 Web 服務，構建支援員工和客戶需求的基礎設施。但是，他們發現將部署複製到新的 AWS 區域，以支援在多個國家/地區設立的咖啡館分店挑戰重重。

咖啡館還希望擁有具有可靠匹配配置的獨立開發和生產環境。他們意識到，必須開始自動化才能支援持續增發展。

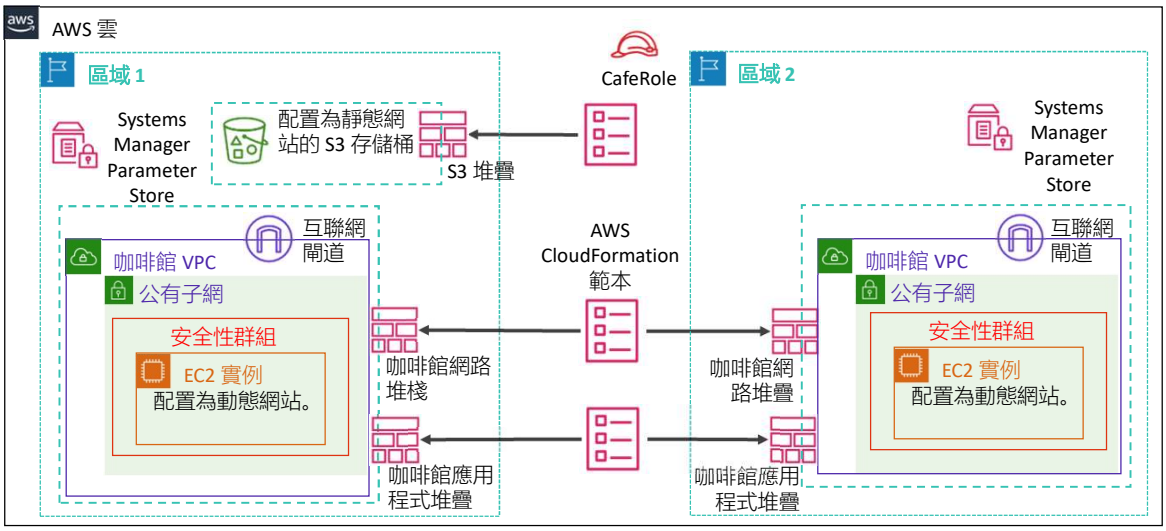
挑戰實驗：任務

1. 從頭創建 AWS CloudFormation 範本
2. 將存儲桶配置為網站並更新堆疊
3. 克隆包含 AWS CloudFormation 範本的 CodeCommit 存儲庫
4. 使用 AWS CloudFormation、CodeCommit 和 CodePipeline 創建新的網路層
5. 更新網路堆疊
6. 定義 EC2 實例資源並創建應用程式堆疊
7. 將咖啡館網路和網站複製到另一個 AWS 區域

在本挑戰實驗中，您將完成以下任務：

1. 從頭創建 AWS CloudFormation 範本
2. 將存儲桶配置為網站並更新堆疊
3. 克隆包含 AWS CloudFormation 範本的 CodeCommit 存儲庫
4. 使用 AWS CloudFormation、CodeCommit 和 CodePipeline 創建新的網路層
5. 更新網路堆疊
6. 定義 EC2 實例資源並創建應用程式堆疊
7. 將咖啡館網路和網站複製到另一個 AWS 區域

挑戰實驗：最終產品



該圖顯示了您將在挑戰實驗中構建的完整架構。

現在可以開始挑戰實驗了。

挑戰實驗總結：要點

完成這個指導實驗之後，您的講師可能會帶您討論此指導實驗的要點。

模組 10：實現架構自動化

模組總結

現在來回顧下本模組，並對知識測驗和對實踐認證考試問題的討論進行總結。

模組總結

aws academy

總體來說，您在本模組中學習了如何：

- 識別何時實現自動化及原因
- 確定如何使用 AWS CloudFormation 建模、創建和管理 AWS 資源集合
- 使用 Quick Start AWS CloudFormation 範本設置架構
- 說明如何借助 AWS System Manager 和 AWS OpsWorks 實現基礎設施和部署自動化
- 說明如何使用 AWS Elastic Beanstalk 部署簡單的應用程式

總體來說，您在本模組中學習了如何：

- 識別何時實現自動化及原因
- 確定如何使用 AWS CloudFormation 建模、創建和管理 AWS 資源集合
- 使用 Quick Start AWS CloudFormation 範本設置架構
- 說明如何借助 AWS System Manager 和 AWS OpsWorks 實現基礎設施和部署自動化
- 說明如何使用 AWS Elastic Beanstalk 部署簡單的應用程式

完成知識測驗

現在可以完成本模組的知識測驗。

樣題

aws academy

考慮這樣一種情況，您希望創建單一的 AWS CloudFormation 範本，該範本既能創建跨兩個可用區的生產環境，又能創建存在于單個可用區中的開發環境。

您希望利用 AWS CloudFormation 範本的哪個可選部分來配置支援此功能的邏輯？

A. 資源
B. 輸出
C. 條件
D. 描述

請查看答案選項，並根據之前突出顯示的關鍵字排除錯誤選項。

正確答案是 C – 條件。可選的**條件**部分包含一些聲明，定義了在哪些情況下創建或配置實體。**資源**不是 AWS CloudFormation 範本的可選部分。輸出**部分不會影響堆疊運行時範本將部署的 AWS 資源數量。**描述**部分不影響配置。**

其他資源

- AWS 上的部署選項概覽
- 使用 AWS CloudFormation 範本
- AWS CloudFormation 示例範本
- AWS OpsWorks Stacks 常見問題
- AWS Systems Manager 功能
- AWS Elastic Beanstalk 常見問題

如果您想瞭解有關本模組所涵蓋主題的更多資訊，下面這些其他資源可能會有所幫助：

- AWS 上的部署選項概覽
- 使用 AWS CloudFormation 範本
- AWS CloudFormation 示例範本
- AWS OpsWorks Stacks 常見問題
- AWS Systems Manager 功能
- AWS Elastic Beanstalk 常見問題

感謝您完成本模組的學習。