AWS 白皮書

AWS 部署選項概觀

AWS 部署選項概觀: AWS 白皮書

Table of Contents

AWS 部署選項概觀

發佈日期：2024 年 5 月 31 日 (文件修訂)

摘要

Amazon Web Services (AWS) 提供多種選項，可讓您佈建基礎設施和部署應用程式。無論您的應用程式架構是簡單的三層 Web 應用程式或一組複雜的工作負載，AWS 都會提供部署服務，以滿足應用程式和組織的需求。

本白皮書適用於尋求 AWS 所提供不同部署服務的概觀的個人。它列出了這些部署服務中可用的常見功能，並闡明了部署和更新應用程式堆疊的基本策略。

簡介

在 AWS 上建置架構良好的應用程式時，為您的應用程式設計部署解決方案至關重要。根據您的應用程式的性質及其所需的基礎服務，您可以使用 AWS 服務來建立靈活的部署解決方案，該解決方案可根據您的應用程式和組織的需求量身打造。

不斷增長的 AWS 服務目錄不僅使決定哪些服務將構成您的應用程式架構的程序複雜化，也使決定您將如何建立、管理和更新應用程式的程序複雜化。在 AWS 上設計部署解決方案時，您應該考慮解決方案將如何處理下列功能：

- 佈建 - 建立應用程式所需的原始基礎設施或受管服務基礎設施。
- 設定 - 根據環境、執行時間、安全性、可用性、效能、網路或其他應用程式需求自訂您的基礎設施。
- 部署 - 在基礎設施資源上安裝或更新您的應用程式元件，並管理從先前應用程式版本到新應用程式版本的轉換。
- 擴展 - 根據一組使用者定義的條件，主動或被動地調整應用程式可用的資源量。
- Monitor - 提供在您的應用程式架構中啟動之資源的可見性。追蹤資源使用情況、部署成功或失敗、應用程式運作狀態、應用程式日誌、組態偏離等。

本白皮書重點介紹 AWS 提供的部署服務，並概述為任何類型的應用程式設計成功部署架構的策略。

AWS 部署服務

設計可擴展、有效率且符合成本效益的部署解決方案的任務，不應限於如何更新應用程式版本，也應考慮如何在整個應用程式生命週期中管理支援基礎設施。資源佈建、組態管理、應用程式部署、軟體更新、監控、存取控制和其他考量，都是設計部署解決方案時需要考量的重要因素。

AWS 服務可以為應用程式生命週期的一或多個層面提供管理功能。根據您所需的控制平衡（手動管理資源）與便利性 (AWS 管理資源）和應用程式類型，這些服務可以單獨使用或結合使用，以建立功能豐富的部署解決方案。本節將提供 AWS 服務的概觀，可用來讓組織更快速且可靠地建置和交付應用程式。

AWS CloudFormation

AWS CloudFormation 是一項服務，可讓客戶使用以 YAML 或 JSON 表示的自訂範本語言來佈建和管理幾乎任何 AWS 資源。範本 CloudFormation 會在稱為堆疊的群組中建立基礎設施資源，並可讓您定義和自訂操作應用程式所需的所有元件，同時保留這些資源的完整控制權。使用 範本可在您的基礎設施上實作版本控制，並能夠快速可靠地複寫您的基礎設施。

CloudFormation 提供對所有應用程式基礎設施元件的佈建和管理的精細控制，從路由表或子網路組態等低階元件，到 CloudFront 分佈等高階元件。 CloudFormation 通常與其他 AWS 部署服務或第三方工具搭配使用，結合 CloudFormation 更專業化的部署服務來管理在基礎設施元件上的應用程式程式碼部署。

除了基本功能之外，AWS 還提供 CloudFormation 服務的擴充功能：

- AWS Cloud Development Kit (AWS CDK) 是一種開放原始碼軟體開發套件 (SDK)，可透過程式設計方式使用 TypeScript、JavaScript、Python、Java 或 C#/ 建立 AWS 基礎設施的模型。NET。
- AWS Serverless Application Model (AWS SAM) 是一種開放原始碼架構，可簡化在 AWS 上建置無伺服器應用程式。它提供速記語法來表達函數、APIs、資料庫和事件來源映射。

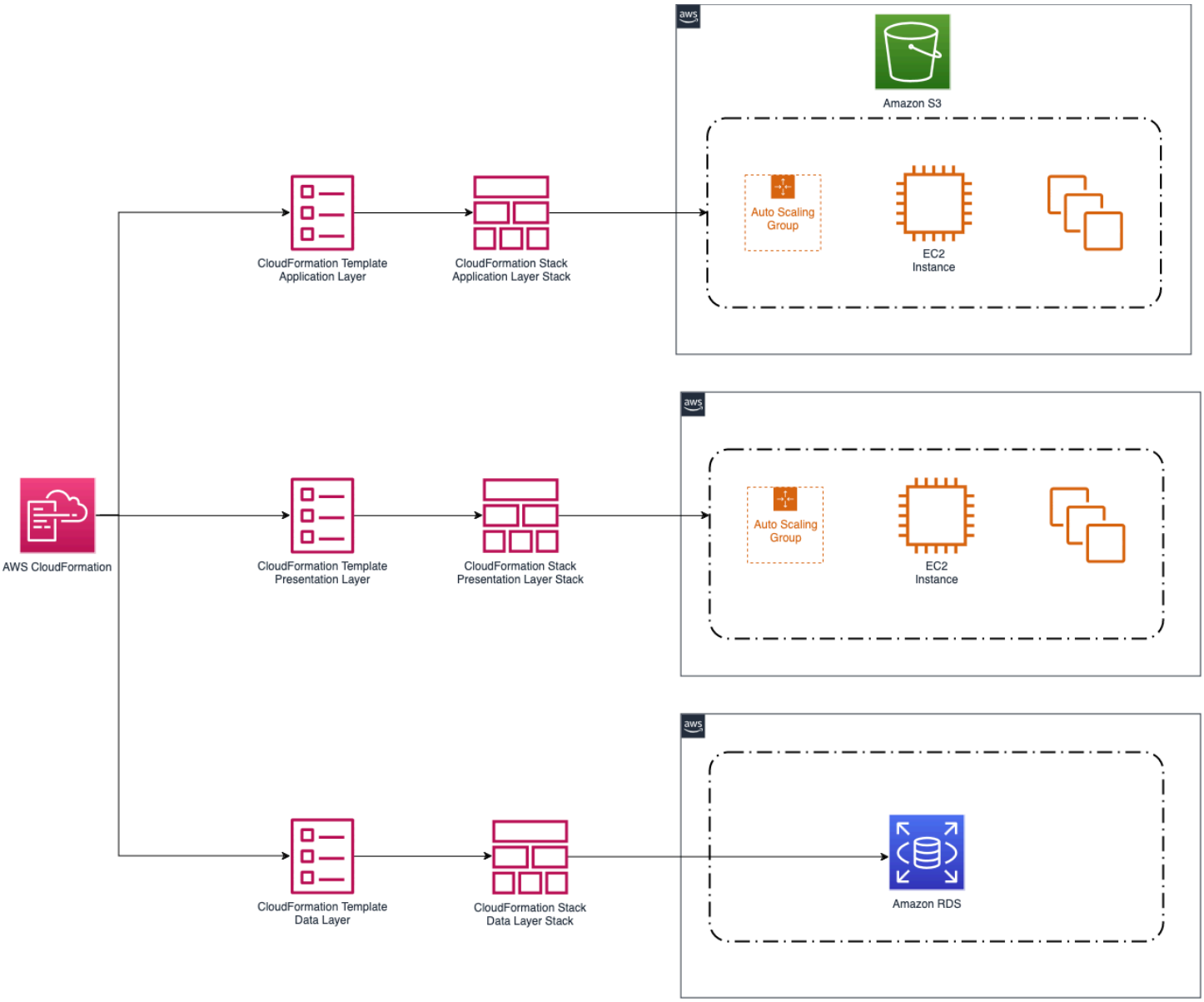
表 1： AWS CloudFormation 部署功能

功能	Description
佈建	CloudFormation 將自動建立和更新範本中定義的基礎設施元件。

功能	Description
	如需使用 CloudFormation 範本建立基礎設施的詳細資訊，請參閱AWS CloudFormation 最佳實務。
設定	CloudFormation 範本提供廣泛的彈性來自訂和更新所有基礎設施元件。 如需自訂範本的詳細資訊，請參閱CloudFormation 範本剖析。
部署	更新您的 CloudFormation 範本以變更堆疊中的資源。根據您的應用程式架構，您可能需要額外的部署服務來更新在基礎設施上執行的應用程式版本。 如需 CloudFormation 如何使用 做為部署解決方案的詳細資訊，請參閱使用 在 Amazon EC2 上部署應用程式 AWS CloudFormation。
擴展	CloudFormation 不會代表您自動處理基礎設施擴展；不過，您可以為 CloudFormation 範本中的資源設定自動擴展政策。
監控	CloudFormation 提供對範本中定義之基礎設施更新成功或失敗的原生監控，以及當範本中定義之資源不符合規格時監控的偏離偵測。需要為應用程式層級監控和指標設置其他監控解決方案。 如需如何監控基礎設施更新的詳細資訊，請參閱監控堆疊更新的進度。 CloudFormation

下圖顯示 的常見使用案例 CloudFormation。在這裡， CloudFormation 範本是用來定義建立簡單三層 Web 應用程式所需的所有基礎設施元件。在此範例中，我們使用 中定義的引導指令碼 CloudFormation ，將最新版的應用程式部署到 Amazon EC2 執行個體；不過，將其他部署服務與 合併 CloudFormation (CloudFormation 僅用於其基礎設施管理和佈建功能） 也是常見的做法。請注意，使用多個 CloudFormation 範本來建立基礎設施。在圖表中， CloudFormation 用於建立所有基礎

設施元件，包括 IAM 角色、VPCs、子網路、路由表、安全群組和 Amazon S3 儲存貯體政策。個別 CloudFormation 範本用於建置應用程式架構的每個網域。



AWS CloudFormation 使用案例

AWS Elastic Beanstalk

AWS Elastic Beanstalk 是一種easy-to-use的服務，可在熟悉的伺服器上部署和擴展使用 Java、.NET、.NET Core、PHP、Node.js、Python、Ruby、Go 或 Docker 開發的 Web 應用程式和服務，例如 Apache、Nginx、 Passenger 和 IIS。Elastic Beanstalk 是完整的應用程式管理解決方案，可代表您管理所有基礎設施和平台任務。

使用 Elastic Beanstalk，您可以快速部署、管理和擴展應用程式，而無需管理基礎設施的操作負擔。Elastic Beanstalk 可降低 Web 應用程式的管理複雜性，使其成為 AWS 新手或希望盡快部署 Web 應用程式的組織的理想選擇。

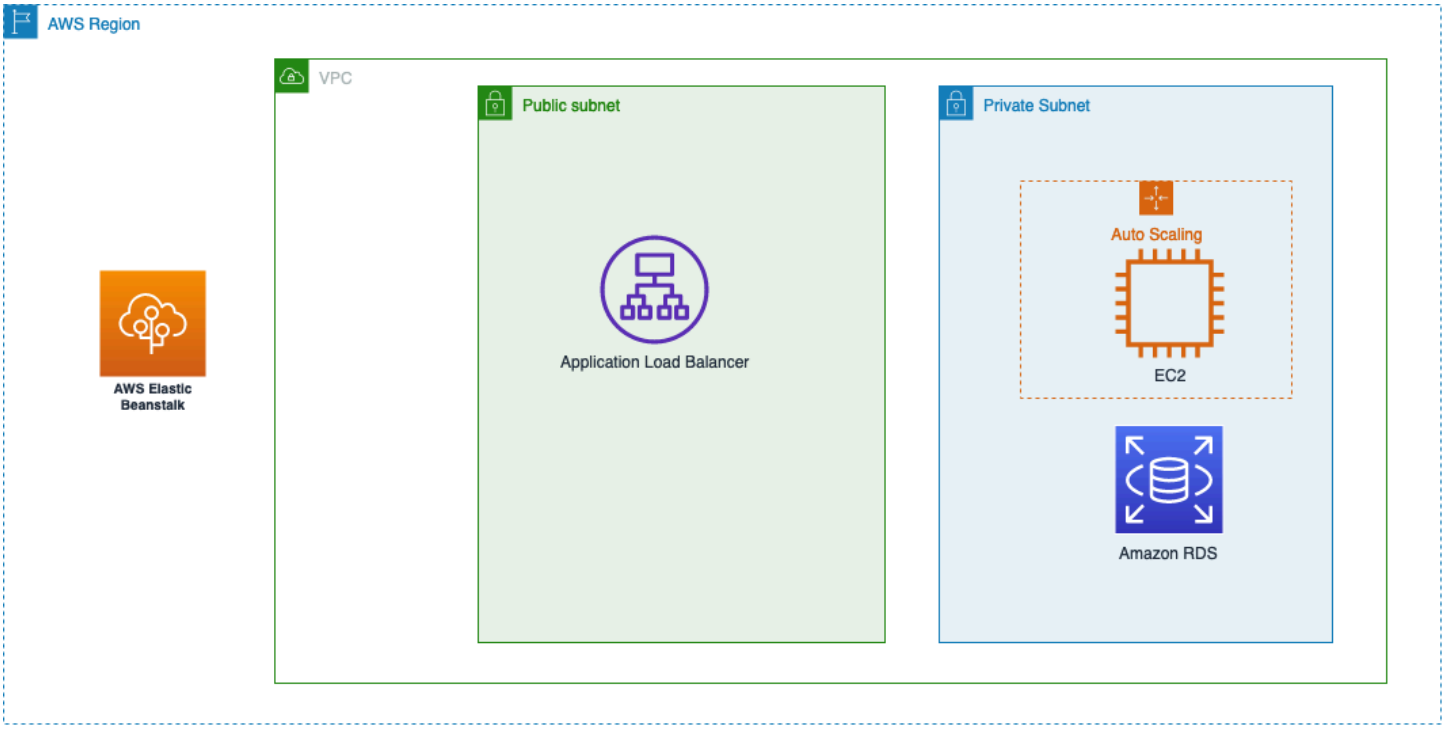
使用 Elastic Beanstalk 做為部署解決方案時，只需上傳您的原始程式碼，Elastic Beanstalk 就會佈建和操作所有必要的基礎設施，包括伺服器、資料庫、負載平衡器、網路和自動擴展群組。雖然這些資源是代表您建立的，但您可以保留這些資源的完全控制權，讓開發人員可以視需要自訂。Elastic Beanstalk 符合 ISO、PCI、SOC 1、SOC 2 和 SOC 3 合規條件，以及 HIPAA 資格條件。這表示在 Elastic Beanstalk 上執行的應用程式可以處理受管制的財務資料或受保護的健康資訊 (PHI)。

表 2： AWS Elastic Beanstalk 部署功能

功能	Description
佈建	Elastic Beanstalk 會建立操作 Web 應用程式或服務所需的所有基礎設施元件，這些應用程式或服務會在其中一個支援的平台上執行。如果您需要額外的基礎設施，則必須在 Elastic Beanstalk 外部建立。 如需 Elastic Beanstalk 支援的 Web 應用程式平台詳細資訊，請參閱 Elastic Beanstalk 平台。
設定	Elastic Beanstalk 提供各種選項，可讓您自訂環境中的資源。 如需自訂 Elastic Beanstalk 所建立資源的詳細資訊，請參閱設定 Elastic Beanstalk 環境。
部署	Elastic Beanstalk 會自動處理應用程式部署，並建立執行應用程式新版本而不影響現有使用者的環境。 如需使用 Elastic Beanstalk 部署應用程式的詳細資訊，請參閱將應用程式部署至 AWS Elastic Beanstalk 。
擴展	Elastic Beanstalk 使用 Elastic Load Balancing 和 Auto Scaling，根據應用程式的特定需求自動

功能	Description
	向內和向外擴展應用程式。多個可用區域可讓您選擇改善應用程式可靠性和可用性。 如需使用 Elastic Beanstalk 自動擴展的詳細資訊，請參閱 Elastic Beanstalk 環境的 Auto Scaling 群組。
監控	Elastic Beanstalk 為應用程式提供內建的環境監控，包括部署成功/失敗、環境運作狀態、資源效能和應用程式日誌。 如需使用 Elastic Beanstalk 進行完整堆疊監控的詳細資訊，請參閱監控環境。
Graviton 支援	AWS Graviton arm64 型處理器可為在 Amazon EC2 中執行的雲端工作負載提供最佳價格效能。使用 Elastic Beanstalk 上的 AWS Graviton，您可以選擇 Amazon EC2 執行個體類型，以滿足工作負載的最佳化需求，並受益於比類似 x86 型處理器更高的價格效能。

Elastic Beanstalk 可讓您輕鬆地在 AWS 中快速部署和管理 Web 應用程式。下列範例顯示 Elastic Beanstalk 用於部署簡單 Web 應用程式的一般使用案例。所有應用程式基礎設施（包括安全群組、IAM 角色和 CloudWatch 警示）都是由 Elastic Beanstalk 建立和管理。Amazon EC2 執行個體會自動佈建執行期環境和部署套件。Elastic Beanstalk 環境可與在 Elastic Beanstalk 外部建立的 Amazon Relational Database Service (Amazon RDS) 等資源整合。



AWS Elastic Beanstalk 使用案例

AWS CodeDeploy

AWS CodeDeploy 是一種全受管部署服務，可將應用程式部署自動化以運算服務，例如 Amazon EC2、Amazon Elastic Container Service (Amazon ECS)AWS Lambda、 或內部部署伺服器。組織可以使用 CodeDeploy 自動化應用程式的部署，並從部署程序中移除容易出錯的手動操作。CodeDeploy 可與各種應用程式內容搭配使用，包括程式碼、無伺服器函數、組態檔案等。

CodeDeploy 旨在用作建置區塊服務，專注於協助應用程式開發人員部署和更新在現有基礎設施上執行的軟體。它不是end-to-end應用程式管理解決方案，旨在與其他 AWS 部署服務搭配使用，例如 AWS CodeStar、、AWS CodePipeline其他 AWS 開發人員工具和第三方服務（請參閱AWS CodeDeploy 產品整合以取得產品整合的完整清單），作為完整 CI/CD 管道的一部分。此外，CodeDeploy 不會代表使用者管理資源的建立。

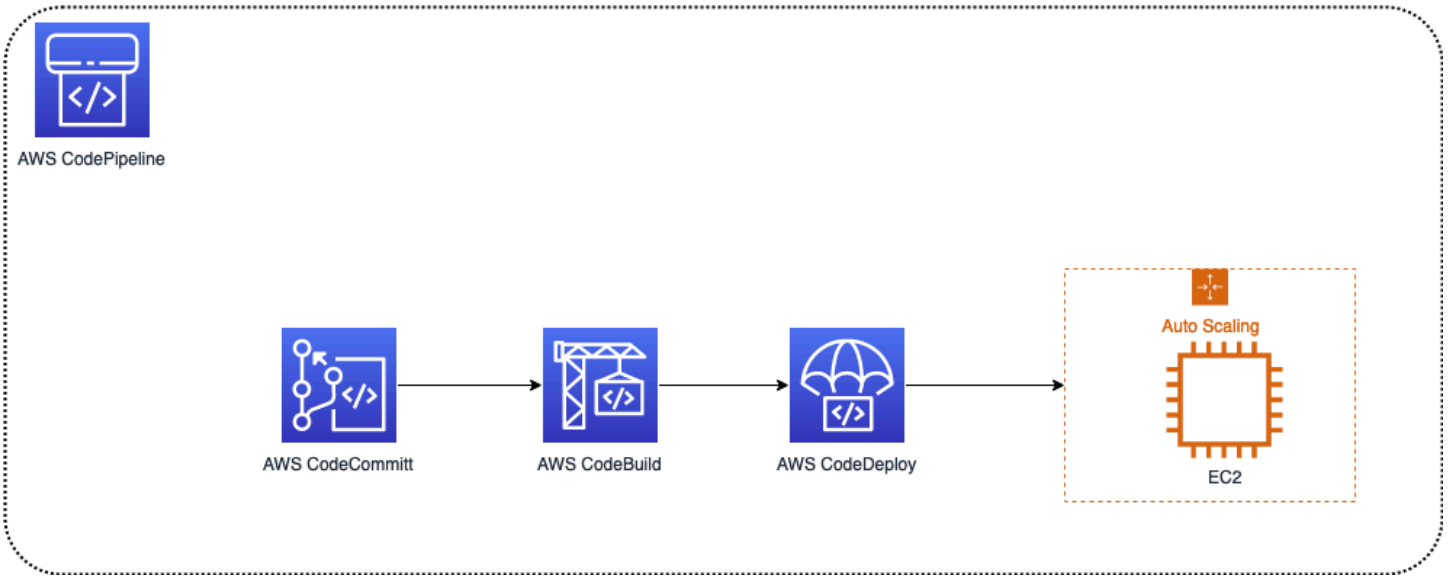
表 3： AWS CodeDeploy 部署功能

功能	Description
佈建	CodeDeploy 適用於現有運算資源，不會代表您建立資源。CodeDeploy 需要將運算資源組織成

功能	Description
	稱為部署群組的建構模組，才能部署應用程式內容。 如需將 CodeDeploy 連結至運算資源的詳細資訊，請參閱在 CodeDeploy 中使用部署群組。CodeDeploy
設定	CodeDeploy 使用應用程式規格檔案來定義運算資源的自訂。 如需使用 CodeDeploy 自訂資源的詳細資訊，請參閱 CodeDeploy AppSpec 檔案參考。CodeDeploy
部署	根據 CodeDeploy 使用的運算資源類型，CodeDeploy 提供不同的部署應用程式策略。 如需支援的部署程序類型的詳細資訊，請參閱在 CodeDeploy 中使用部署。
擴展	CodeDeploy 不支援擴展基礎應用程式基礎設施；不過，根據您的部署組態，它可能會建立其他資源以支援藍/綠部署。
監控	CodeDeploy 可以監控部署的成功或失敗，並提供所有部署的歷史記錄，但不提供效能或應用程式層級指標。 如需 CodeDeploy 提供之監控功能類型的詳細資訊，請參閱在 CodeDeploy 中監控部署 CodeDeploy

下圖說明 CodeDeploy 作為完整 CI/CD 解決方案一部分的一般使用案例。在此範例中，CodeDeploy 會與其他 AWS 開發人員工具搭配使用，也就是 AWS CodePipeline（自動化 CI/CD 管道）、AWS CodeBuild（建置和測試應用程式元件）和 AWS CodeCommit（原始程式碼儲存庫），以將應用程式部署至 Amazon EC2 執行個體群組。CodeDeploy 會與其他工具搭配使用，做為完整 CI/CD 管道的一

部分。CodeDeploy 會管理應用程式元件在屬於部署群組的運算資源上的部署。所有基礎設施元件都是在 CodeDeploy 外部建立。



AWS CodeDeploy 使用案例

AWS CodeDeploy 適用於 的 AWS Lambda

AWS CodeDeploy 的 AWS Lambda 可讓您自動化無伺服器部署，讓您更妥善地控制應用程式版本。您可以使用 CodeDeploy 將新版本的無伺服器函數部署到一小部分的 使用者或流量，並在您對新版本獲得信心時逐漸增加流量。使用 CodeDeploy，您可以定義部署群組，代表一組從相同事件來源接收流量的 Lambda 函數。例如，您可以為 API Gateway 或 Amazon EventBridge 規則啟動的一組 Lambda 函數建立部署群組。然後，您可以使用 CodeDeploy 建立部署，將新版本的 erverless 函數部署到指定的部署群組。

CodeDeploy 也可讓您定義部署組態，指定部署的設定，例如部署類型、部署策略和流量轉移規則。您可以使用 Canary 部署策略，將新版本的無伺服器函數部署到一小部分的流量，並在增加流量之前監控新版本的運作狀態和效能。

透過將 CodeDeploy 用於無伺服器，您可以自動化部署程序、減少發佈應用程式新版本所需的時間和精力，並提高無伺服器函數的穩定性和可靠性。

Amazon Elastic Container Service

Amazon Elastic Container Service (Amazon ECS) 是一種全受管容器協同運作服務，可支援 Docker 容器，並可讓您輕鬆地在受管叢集上執行應用程式。Amazon ECS 無需安裝、操作和擴展容器管理基

礎設施，並可使用安全群組、Elastic Load Balancing 和 AWS Identity and Access Management(IAM)等熟悉的 AWS 核心功能簡化環境的建立。

在 Amazon ECS 上執行應用程式時，您可以選擇使用 Amazon EC2 執行個體或容器的無伺服器運算引擎AWS Fargate，為容器提供基礎運算能力。無論哪種情況，Amazon ECS 都會根據使用者定義的組態，自動將容器放置並擴展到叢集。雖然 Amazon ECS 不會代表您建立負載平衡器或 IAM 角色等基礎設施元件，但 Amazon ECS 服務提供多種 APIs，可簡化 Amazon ECS 叢集中這些資源的建立和使用。

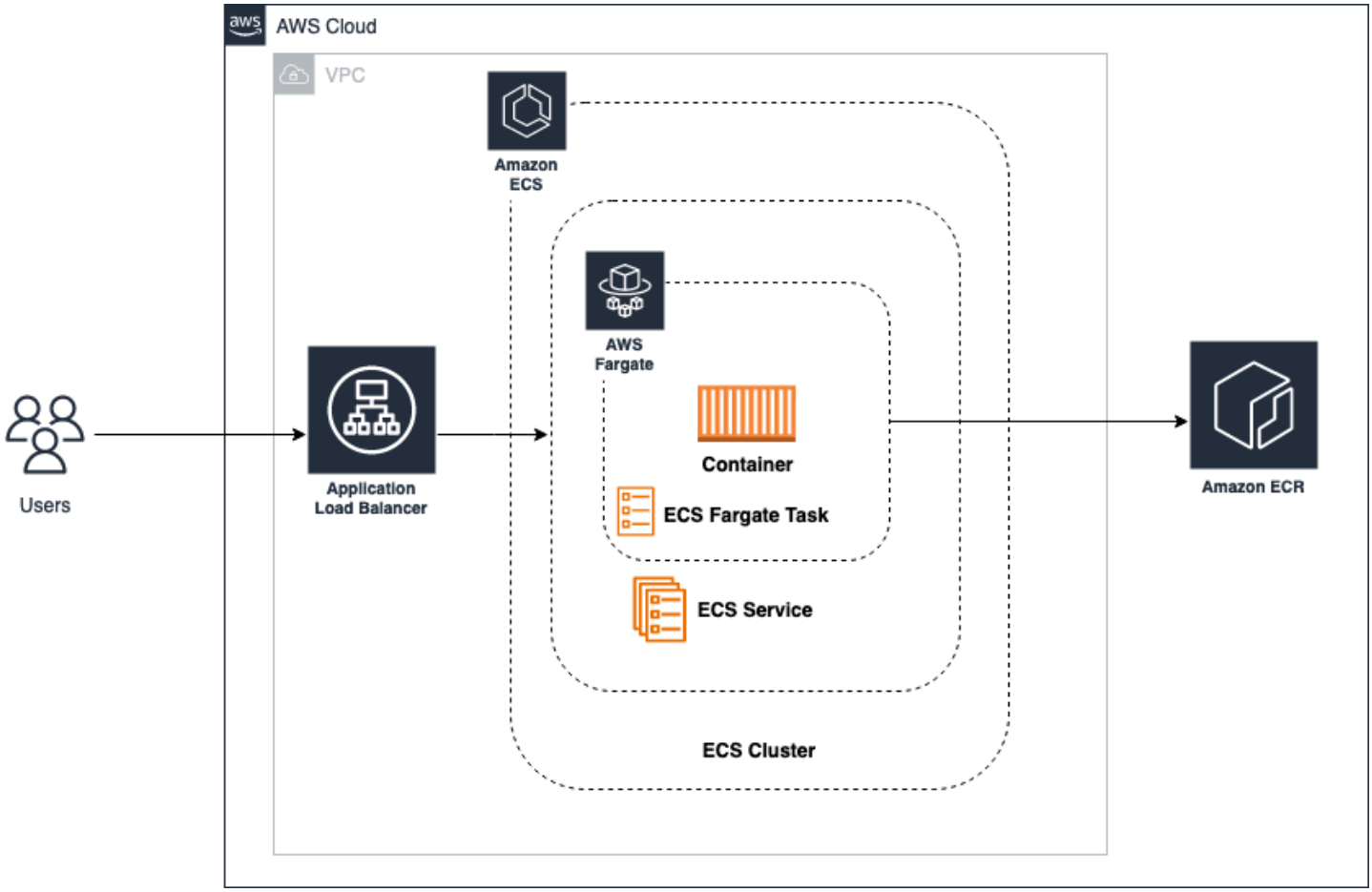
Amazon ECS 可讓開發人員直接精細控制所有基礎設施元件，進而建立自訂應用程式架構。此外，Amazon ECS 支援不同的部署策略來更新您的應用程式容器映像。

表 4：Amazon ECS 部署功能

功能	Description
佈建	Amazon ECS 將根據擴展政策和 Amazon ECS 組態佈建新的應用程式容器執行個體和運算資源。諸如 Load Balancer 等基礎設施資源將需要在 Amazon ECS 外部建立。 如需可使用 Amazon ECS 建立之資源類型的詳細資訊，請參閱 Amazon ECS 入門。
設定	Amazon ECS 支援自訂為執行容器化應用程式而建立的運算資源，以及應用程式容器的執行時間條件（例如，環境變數、公開連接埠、預留記憶體/CPU)。只有在使用 Amazon EC2 執行個體時，才能自訂基礎運算資源。 如需如何自訂 Amazon ECS 叢集以執行容器化應用程式的詳細資訊，請參閱建立叢集。
部署	Amazon ECS 為您的容器化應用程式支援多種部署策略。 如需支援的部署程序類型的詳細資訊，請參閱 Amazon ECS 部署類型。

功能	Description
擴展	Amazon ECS 可與自動擴展政策搭配使用，以自動調整 Amazon ECS 叢集中執行的容器數量。 如需在 Amazon ECS 上為容器化應用程式設定自動擴展的詳細資訊，請參閱 Service Auto Scaling。
監控	Amazon ECS 支援使用 CloudWatch 監控運算資源和應用程式容器。 如需 Amazon ECS 提供之監控功能類型的詳細資訊，請參閱監控 Amazon ECS。

下圖說明 Amazon ECS 用於管理簡單的容器化應用程式。在此範例中，基礎設施元件是在 Amazon ECS 外部建立，而 Amazon ECS 用於管理叢集上應用程式容器的部署和操作



Amazon ECS 使用案例

> **Note**
>
> - 應用程式基礎設施 （ 包括 Amazon Elastic Container Registry (Amazon ECR) 儲存庫、Amazon ECS 組態和 Load Balancer) 是在 Amazon ECS 部署之外佈建和管理。
> - Amazon ECS 會將在 Amazon ECS 服務內執行的應用程式容器部署管理為從 Amazon ECR 等容器登錄檔取得的任務。

Amazon ECS 支援多個容器執行個體類型，例如 Linux 和 Windows，以及外部執行個體類型，例如使用 Amazon ECS Anywhere 的現場部署虛擬機器 (VM)。

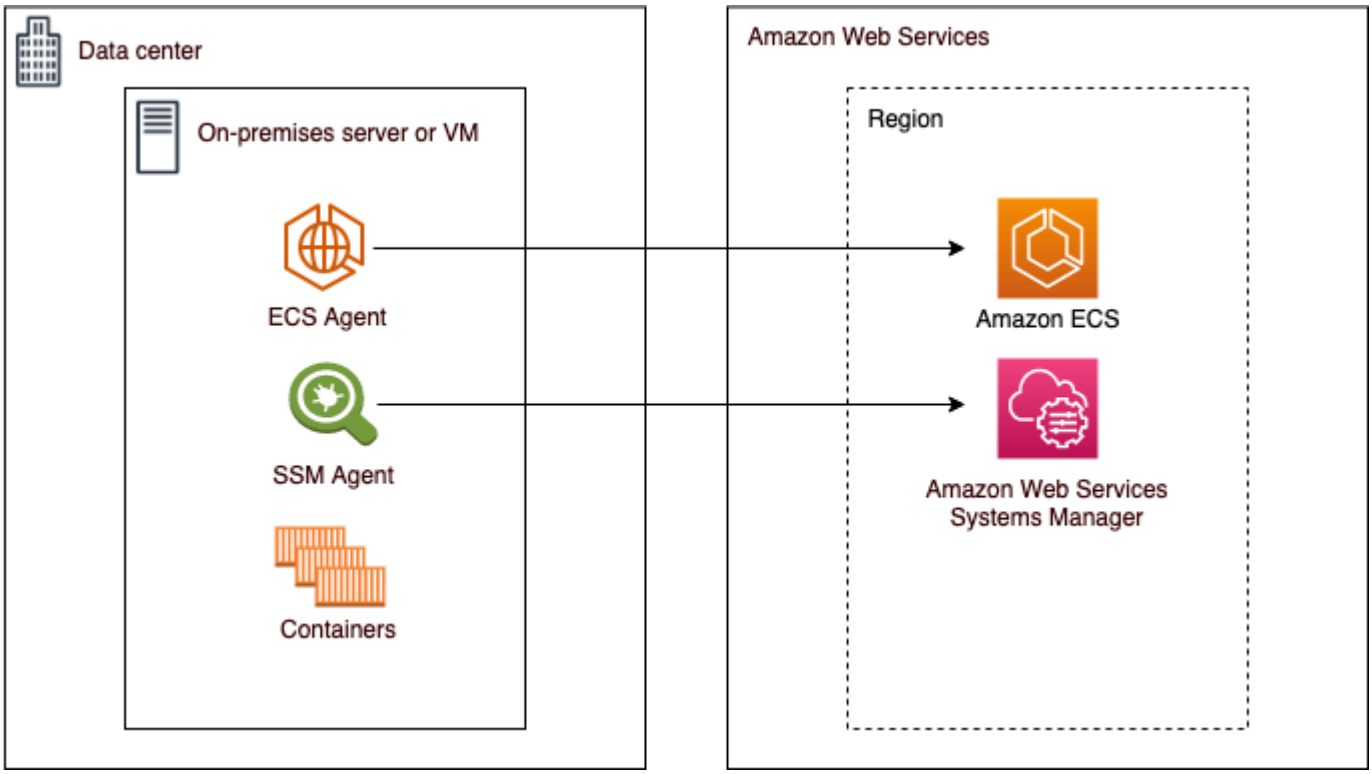
Amazon ECS Anywhere

Amazon ECS Anywhere 可讓您在任何地方執行 Amazon ECS 任務，無論是內部部署還是在其他雲端環境中。透過 Amazon ECS Anywhere，您可以在混合基礎設施中輕鬆部署和管理容器化應用程式，同

時保持一致的操作體驗。此服務的運作方式是將 Amazon ECS 平台延伸至任何環境，包括內部部署資料中心、遠端辦公室和其他雲端環境。它可讓您使用相同的熟悉 Amazon ECS APIs 和工具，在所有環境中部署和管理容器，而不必擔心基礎基礎設施。

Amazon ECS Anywhere 使用 Amazon ECS 代理程式來管理容器的部署和生命週期，讓您能夠使用您在 中使用的相同 Amazon ECS 任務定義和組態檔案 AWS 雲端。這有助於簡化跨混合基礎設施部署和管理容器的程序，並減少手動組態和管理所需的時間和精力。

使用 Amazon ECS Anywhere，您也可以利用其他 AWS 服務，例如 IAM CloudFormation和 Amazon ECR，來管理您的容器化應用程式。這有助於確保您的應用程式安全、合規，並與其他 AWS 服務整合。



Amazon ECS Anywhere architecture

上的 Amazon Elastic Container Service AWS Outposts

上的 Amazon ECS AWS Outposts 是一項全受管 AWS 服務，可讓您使用您在 中使用的相同 APIs 和工具，在內部部署執行 Amazon ECS 任務 AWS 雲端。開啟 Amazon ECS 後 AWS Outposts，無論您是在內部部署還是雲端中執行容器化應用程式，您都可以以一致且熟悉的方式部署和管理這些應用程式。 AWS Outposts 是一種全受管服務，可將 AWS 基礎設施、服務、APIs 和工具延伸至您的內部部署環境。開啟 Amazon ECS 後 AWS Outposts，您可以在組織專用的硬體上執行 Amazon ECS 任務，而不必擔心基礎基礎設施。這有助於確保您的應用程式以安全且合規的方式部署，同時可讓您利用雲端的彈性和可擴展性。

上的 Amazon ECS AWS Outposts 的運作方式是將一組 AWS 服務和 APIs 部署到您的內部部署環境，讓您能夠在專用硬體上執行 Amazon ECS 任務。這包括 Amazon ECS 代理程式，其可管理容器的部署和生命週期，以及 AWS Outposts 基礎設施，可為執行容器化應用程式提供安全且合規的環境。開啟 Amazon ECS 後 AWS Outposts，您可以使用您在 中使用的相同 Amazon ECS APIs 和工具 AWS 雲端，以一致且熟悉的方式輕鬆部署和管理容器化應用程式。這有助於減少手動組態和管理所需的時間和精力，並改善混合基礎設施的一致性和可靠性。上的 Amazon ECS AWS Outposts 也會與其他 AWS 服務整合，例如 IAM CloudFormation和 Amazon ECR，以管理您的容器化應用程式。這有助於確保您的應用程式安全、合規，並與其他 AWS 服務整合。

Amazon Elastic Kubernetes Service

Amazon Elastic Kubernetes Service (Amazon EKS) 是一項全受管、經認證的 Kubernetes 合規服務，可簡化在 AWS 上建置、保護、操作和維護 Kubernetes 叢集的程序。Amazon EKS 與 CloudWatch、Auto Scaling Groups 和 IAM 等核心 AWS 服務整合，提供監控、擴展和負載平衡您容器化應用程式的無縫體驗。

Amazon EKS 為 Kubernetes 工作負載提供可擴展、高可用性的控制平面。當您在 Amazon EKS 上執行應用程式時，如同使用 Amazon ECS 時，您可以選擇為具有 Amazon EC2 執行個體或使用 的容器提供基礎運算能力 AWS Fargate。

Amazon VPC Lattice 是直接內建於 AWS 網路基礎設施的全受管應用程式聯網服務，可用於跨多個帳戶和虛擬私有雲端 (VPCs) 來連接、保護和監控您的服務。透過 Amazon EKS，您可以透過使用 Kubernetes Gateway API 的實作 AWS Gateway API 控制器來利用 VPC Lattice。您可以使用 VPC Lattice，以簡單且一致的方式使用標準 Kubernetes 語意設定跨叢集連線。

您可以使用 Amazon EKS 搭配下列任何部署選項：

- Amazon EKS Distro：Amazon EKS Distro 是由 Amazon EKS 部署在雲端的相同開放原始碼 Kubernetes 軟體和相依項目的發行版本。Amazon EKS 發行版遵循與 Amazon EKS 相同的 Kubernetes 版本發行週期，並以開放原始碼專案提供。如需進一步了解，請參閱 Amazon EKS Distro。
- Amazon EKS on AWS Outposts – 在您的內部部署設施中 AWS Outposts 啟用原生 AWS 服務、基礎設施和操作模型。Amazon EKS on AWS Outposts，您可以選擇執行擴充或本機叢集。使用擴充叢集時，Kubernetes 控制平面會在 中執行， AWS 區域 節點也會在 上執行 AWS Outposts。使用本機叢集時，整個 Kubernetes 叢集會在本機上執行 AWS Outposts，包括 Kubernetes 控制平面和節點。
- Amazon EKS Anywhere：Amazon EKS Anywhere 是 Amazon EKS 的部署選項，可讓您輕鬆地在內部部署建立和操作 Kubernetes 叢集。Amazon EKS 和 Amazon EKS Anywhere 均建置

在 Amazon EKS Distro 上。若要進一步了解 Amazon EKS Anywhere，請參閱使用 Amazon EKS Anywhere 執行混合容器工作負載、Amazon EKS Anywhere 概觀，以及比較 Amazon EKS Anywhere 與 Amazon EKS。

在選擇用於 Kubernetes 叢集的部署選項時，請考慮以下事項：

表 5：Kubernetes 部署功能

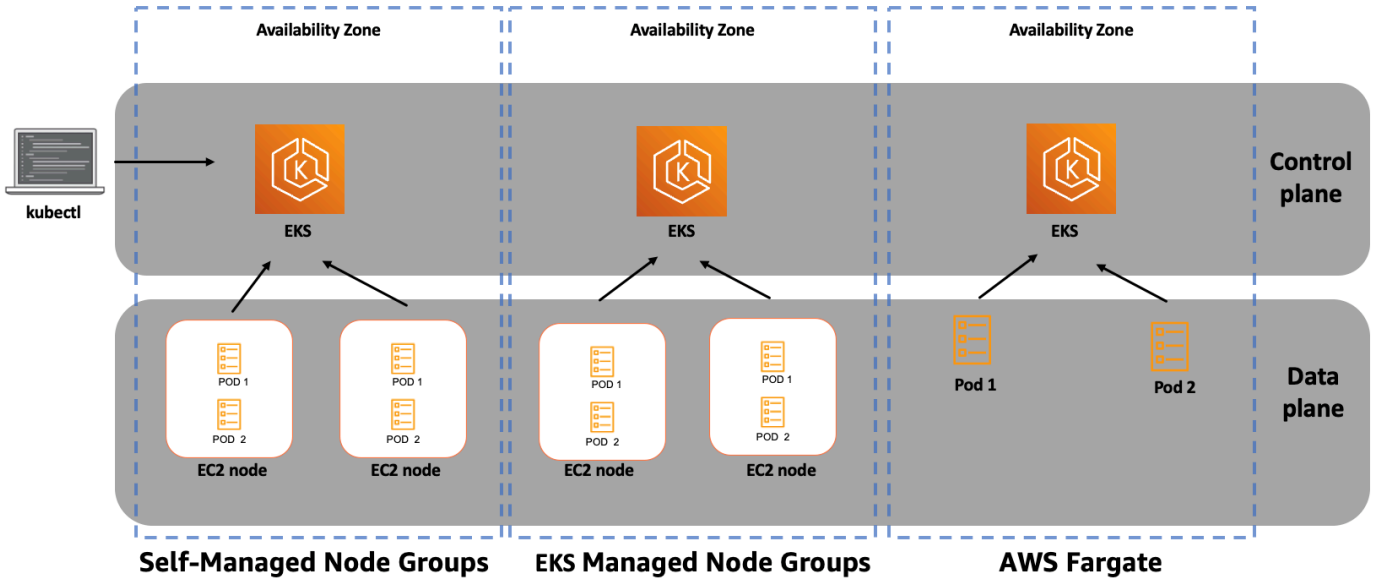
功能	Amazon EKS	上的 Amazon EKS AWS Outposts	Amazon EKS Anywhere	Amazon EKS Distro
硬體	AWS 提供的	AWS 提供的	由您提供	由您提供
部署位置	AWS 雲端	您的資料中心	您的資料中心	您的資料中心
Kubernetes 控制平面位置	AWS 雲端	AWS 雲端 或您的資料中心	您的資料中心	您的資料中心
Kubernetes 資料平面位置	AWS 雲端	您的資料中心	您的資料中心	您的資料中心
支援	AWS 支援	AWS 支援	AWS 支援	OSS 社群支援

表 6：Amazon EKS 部署功能

功能	Description
佈建	Amazon EKS 會佈建特定資源以支援容器化應用程式： • 負載平衡器，如有需要 • 運算資源或工作者 (Amazon EKS 支援 Windows 和 Linux) • 應用程式容器執行個體或 Pod

功能	Description
	如需 Amazon EKS 叢集佈建的詳細資訊，請參閱 Amazon EKS 入門。
設定	如果您使用 Amazon EC2 執行個體提供運算能力，Amazon EKS 支援自訂運算資源（工作者）。Amazon EKS 也支援應用程式容器 (Pod) 執行時間條件的自訂。 如需詳細資訊，請參閱工作者節點和 Fargate Pod 組態文件。
部署	Amazon EKS 支援與 Kubernetes 相同的部署策略。如需詳細資訊，請參閱撰寫 Kubernetes 部署規格 -> 策略。
擴展	Amazon EKS 使用 Kubernetes Cluster Autoscaler 擴展工作者，並使用 Kubernetes Horizontal Pod Autoscaler 和 Kubernetes Vertical Pod Autoscaler 擴展 Pod。Amazon EKS 也支援 Karpenter，這是一種開放原始碼、靈活且高效能的 Kubernetes 叢集自動擴展器，可透過快速啟動適當大小的運算資源來回應不斷變化的應用程式負載，以協助改善您的應用程式可用性和叢集效率。
監控	Amazon EKS 控制平面日誌會將稽核和診斷資訊直接提供給 CloudWatch Logs。Amazon EKS 控制平面也與 整合 AWS CloudTrail ，以記錄在 Amazon EKS 中採取的動作。 如需詳細資訊，請參閱記錄和監控 Amazon EKS。

Amazon EKS 可讓組織利用開放原始碼 Kubernetes 工具和外掛程式，對於使用現有 Kubernetes 環境遷移至 AWS 的組織來說，是理想的選擇。下圖說明 Amazon EKS 用於管理一般容器化應用程式。



Amazon EKS use case

Amazon EKS Anywhere

Amazon EKS Anywhere 可讓您在自己的基礎設施上建立和操作 Kubernetes 叢集。Amazon EKS Anywhere 以 Amazon EKS Distro 的優勢為基礎，提供最新且修補的開放原始碼軟體，因此您可以擁有比自我管理 Kubernetes 產品更可靠的內部部署 Kubernetes 環境。

Amazon EKS Anywhere 會在內部部署建立 Kubernetes 叢集給所選的供應商。支援的提供者包括裸機（透過 Tinkerbell)、CloudStack 和 vSphere。若要管理該叢集，您可以從 Ubuntu 或 Mac 管理機器執行叢集建立和刪除命令。

AWS App Runner

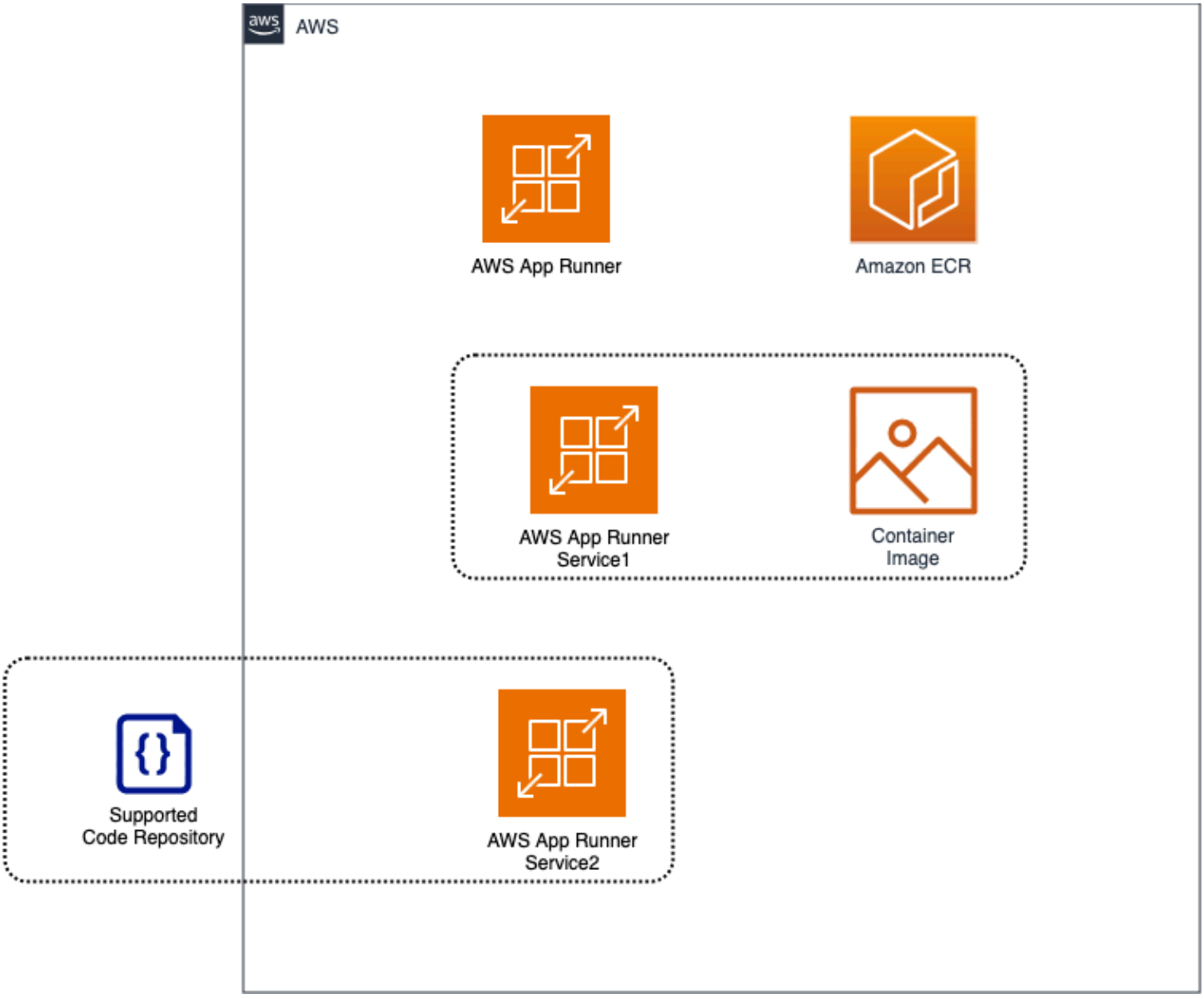
AWS App Runner 是一種全受管容器應用程式服務，可讓您建置、部署和執行容器化 Web 應用程式和 API 服務，而不需要先前的基礎設施或容器體驗。App Runner 會直接連線至您的程式碼或映像儲存庫。它提供具有全受管操作、高效能、可擴展性和安全性的自動整合和交付管道。

App Runner 會從儲存庫取得您的原始碼或原始映像，然後在 中為您建立和維護執行中的 Web 服務 AWS 雲端。一般而言，您只需要呼叫一個 App Runner 動作 `CreateService`，即可建立您的服務。透過來源映像儲存庫，您可以提供 App Runner 可以部署以執行 Web 服務的ready-to-use容器映像。使用原始程式碼儲存庫，您可以提供建置和執行 Web 服務的程式碼和指示，並以特定執行時間環境為目標。App Runner 支援多個程式設計平台，每個平台都有一或多個平台主要版本的受管執行時

間。App Runner 支援容器映像，以及執行期和 Web 架構，包括 Node.js 和 Python。App Runner 會監控傳送至應用程式的並行請求數量，並根據請求磁碟區自動新增其他執行個體。如果您的應用程式未收到傳入請求，App Runner 會將容器縮減為佈建執行個體，CPU 限流執行個體已準備好在幾毫秒內處理傳入請求。

此時，App Runner 可以從 GitHub 儲存庫擷取來源碼，或從 中的 Amazon ECR 擷取來源映像 AWS 帳戶。

下圖顯示 App Runner 服務架構的概觀。在圖表中，有兩個範例服務：一個從 GitHub 部署來源碼，另一個從 Amazon ECR 部署來源映像。



App Runner use case

App Runner 支援完整堆疊開發，包括使用 HTTP 和 HTTPS 通訊協定的前端和後端 Web 應用程式。這些應用程式包括 API 服務、後端 Web 服務和網站。App Runner 支援容器映像，以及執行期和 Web 架構，包括 Node.js 和 Python。

Amazon Lightsail

Amazon Lightsail 是一種簡單且經濟實惠的雲端服務，可讓小型企業、新創公司和個人在雲端輕鬆部署和管理其應用程式。它提供易於使用的界面，可抽象化大部分的基礎基礎設施管理，並讓您輕鬆在雲端中啟動和執行應用程式。透過 Lightsail，您可以快速部署和管理虛擬私有伺服器 (VPS)、資料庫和儲存執行個體。此服務提供預先設定的執行個體，針對 WordPress、Drupal 和 Joomla 等各種工作負載進行最佳化。這有助於減少設定和設定環境所需的時間和精力。 Lightsail也提供整合式負載平衡器和自動擴展，讓您無需手動介入即可處理流量需求的變更。此服務也提供監控和提醒，讓您隨時掌握應用程式的運作狀態和效能。

的主要優點之一Lightsail是其簡單易用。此服務專為具有最少雲端運算體驗的使用者而設計，使其成為想要在雲端快速入門的小型企業或個人的理想選擇。此外， Lightsail 具有成本效益，具有可預測的定價，包括運算、儲存和資料傳輸。

Amazon Lightsail 容器

Amazon Lightsail Containers 是一種全受管容器服務 AWS，可讓您在雲端輕鬆部署和管理容器化應用程式。它提供簡單且符合成本效益的方式，使用常見的容器管理工具啟動和執行容器，例如 Docker 和 Kubernetes。

Lightsail 容器提供整合式環境，用於建置、測試和部署容器化應用程式。它透過提供易於使用的界面來抽象化大部分的基礎基礎設施管理，從而簡化部署和管理容器的程序。

使用Lightsail容器，您只需按幾下滑鼠，即可將容器化應用程式部署至 VPC。此服務提供適用於熱門程式設計語言的預先設定容器映像，例如 Node.js、Python、Ruby 和 Java。這有助於減少設定和設定容器環境所需的時間和精力。

Lightsail 容器也提供整合式負載平衡器，可自動將流量分配到容器執行個體，進而改善應用程式的可用性和可擴展性。此外， 服務還提供容器執行個體的自動擴展，可讓您處理流量需求的變更，而無需手動介入。

使用Lightsail容器，您可以使用內建指標和日誌來監控容器化應用程式的效能。您也可以與其他 AWS 服務整合，例如 Amazon S3、Amazon RDS 和 AWS CodePipeline，為您的容器化應用程式建立全自動化且整合的 CI/CD 管道。

Red Hat OpenShift Service on AWS

Red Hat OpenShift Service on AWS (ROSA) 是一種可透過 AWS 管理主控台取得的受管服務。使用 ROSA，身為 Red Hat OpenShift 使用者，您可以在 AWS 上建置、擴展和管理容器化應用程式。您可以使用 ROSA 來使用 Red Hat OpenShift APIs 和工具建立 Kubernetes 叢集，並可存取 AWS 服務的完整廣度和深度。ROSA 可簡化將內部部署 Red Hat OpenShift 工作負載移至 AWS，並提供與其他 AWS 服務的緊密整合。您也可以直接透過 AWS 存取 Red Hat OpenShift 授權、計費和支援。

每個 ROSA 叢集都隨附全受管控制平面和運算節點。安裝、管理、維護和升級是由 Red Hat SRE 搭配聯合 Red Hat 和 Amazon 支援執行。也可使用叢集服務（例如記錄、指標和監控）。ROSA 僅支援 Red Hat Enterprise Linux CoreOS (RHCOS) 工作者。

ROSA 將與一系列 AWS 運算、儲存、資料庫、分析、機器學習、聯網、行動和各種應用程式服務整合，這將使客戶能夠受益於在全球範圍內擴展的強大 AWS 服務產品組合。這些 AWS 原生服務可直接存取，透過相同的管理界面快速部署和擴展服務。

AWS 本地區域

AWS Local Zone 是 AWS 區域 的延伸，靠近您的使用者。Local Zones 有自己的網際網路連線，並支援 AWS Direct Connect。在 Local Zone 中建立的資源可以為本機使用者提供低延遲的通訊服務。Local Zone 以區域代碼表示，後面接著識別符指出位置（例如 us-west-2-lax-1a)。

當需要低延遲或本機資料處理時，Amazon ECS 支援使用 Local Zones 的工作負載。Amazon ECS 控制平面一律會在 中執行 AWS 區域。

Amazon EKS 支援本機區域中的特定資源。這包括自我管理的 Amazon EC2 節點、Amazon EBS 磁碟區和 Application Load Balancer。Amazon EKS 受管 Kubernetes 控制平面一律在 AWS 區域中執行。Amazon EKS 受管 Kubernetes 控制平面無法在本機區域中執行。由於 Local Zones 在 VPC 內顯示為子網路，因此 Kubernetes 會將您的本機區域資源作為該子網路的一部分。

AWS Wavelength

AWS Wavelength 是一種 AWS 基礎設施，可讓您部署更接近 5G-connected使用者和裝置的工作負載。您可以使用 Wavelength 部署 Amazon EC2 執行個體、Amazon EKS 叢集，以及 AWS Marketplace 上提供的一組支援合作夥伴解決方案。Wavelength 區域是電信供應商網路內邏輯上隔離的資料中心，透過備援、低延遲和高輸送量連線來連線至 AWS 區域。

Wavelength 的一些重要功能包括能夠在 Wavelength 區域中建立 Amazon EC2 執行個體、Amazon EBS 磁碟區，以及 Amazon VPC 子網路和電信業者閘道。您也可以使用協調或使用 Amazon

EC2、Amazon EBS 和 Amazon VPC 的服務，例如 Amazon EC2 Auto Scaling、Amazon EKS 叢集、Amazon ECS 叢集、Amazon EC2 Systems Manager、Amazon CloudWatch AWS CloudTrail AWS CloudFormation和 Application Load Balancer。Wavelength 服務是 VPC 的一部分，透過可靠的高頻寬連線連接到 AWS 區域，以便輕鬆存取 服務，包括 Amazon DynamoDB 和 Amazon Relational Database Service (Amazon RDS)。

其他部署服務

Amazon Simple Storage Service (Amazon S3) 可作為靜態內容和單頁應用程式 (SPA) 的 Web 伺服器。結合 Amazon CloudFront 以提高靜態內容交付的效能，使用 Amazon S3 可以是部署和更新靜態內容的簡單且強大方式。如需此方法的詳細資訊，請參閱白皮書上的託管靜態網站 AWS。

AWS Proton

AWS Proton 是一項全受管服務，可簡化並自動化部署和管理微服務和容器型應用程式的程序。它提供統一且一致的部署體驗，與熱門的 DevOps 工具和服務整合，讓您更輕鬆地管理和簡化應用程式開發。Proton 可讓開發人員定義和建立應用程式元件，例如基礎設施、程式碼和管道，做為可重複使用的範本。這些範本可用來建立多個環境，例如開發、測試和生產，並可跨團隊或組織共用。此方法有助於降低部署和管理微服務和容器型應用程式的複雜性，這可能耗時且容易出錯。

AWS Proton 為常見類型的微服務提供預先建置的範本，例如 Web 應用程式、APIs 和資料庫，可自訂以滿足特定需求。它還與常見的 DevOps 工具整合，例如 AWS CodePipeline、AWS CodeCommit 和 AWS CodeBuild，以啟用持續整合和部署 (CI/CD) 工作流程。

透過使用 AWS Proton，開發人員可以減少部署和管理微服務和容器型應用程式所需的時間和精力。這種方法可讓團隊專注於開發和改善其應用程式，而不是花時間進行部署和管理程序。

AWS App2Container

AWS App2Container 是一種命令列工具，可將 Java 和 .NET Web 應用程式遷移和現代化為容器格式。App2Container 會分析和建置在裸機、虛擬機器、Amazon EC2 執行個體或雲端中執行的應用程式庫存。您只需選取要容器化的應用程式，而 App2Container 會將應用程式成品和已識別的相依性封裝到容器映像中，設定網路連接埠，並產生 ECS 任務和 Kubernetes Pod 定義。App2Container 會識別虛擬機器中執行的支援 ASP.NET 和 Java 應用程式，以建置您環境中所有應用程式的完整清查。App2Container 可以在 Linux 上執行的 Windows 或 Java 應用程式上，或在 JBoss、Apache Tomcat、Springboot、IBM Websphere 和 Oracle Weblogic 等應用程式伺服器上，將 IIS 中執行的 ASP.NET Web 應用程式容器化。

AWS Copilot

AWS Copilot 是一種命令列界面 (CLI)，可用來在 AWS 上快速啟動和管理容器化應用程式。它簡化了在 Amazon ECS、Fargate 和 App Runner 上執行的應用程式。AWS Copilot 目前支援 Linux、macOS 和 Windows 系統。Copilot 可讓您使用負載平衡 Web 服務之類的服務模式來佈建基礎設施、部署到測試或生產等多個環境，甚至使用 AWS CodePipeline 發行管道進行自動化部署。

AWS Serverless Application Model

AWS Serverless Application Model (AWS SAM) 是用於建置無伺服器應用程式的開放原始碼架構。它提供速記語法來表達函數、APIs、資料庫和事件來源映射。每個資源只要幾行，您就可以定義所需的應用程式，並使用 YAML 建立模型。在部署期間，SAM 會將 SAM 語法轉換並擴展為 AWS CloudFormation 語法，讓您更快速地建置無伺服器應用程式。

AWS SAM CLI 是一種開放原始碼命令列工具，可讓您在 AWS 上輕鬆開發、測試和部署無伺服器應用程式。它是使用 AWS SAM 規格建置無伺服器應用程式的命令列界面，該規格是 AWS CloudFormation 的延伸。

CLI AWS SAM 可讓開發人員在本機定義和測試其無伺服器應用程式，然後再將其部署至 AWS。它提供模擬 AWS Lambda 和 API Gateway 的本機測試環境，可讓開發人員在將程式碼和組態部署到雲端之前進行測試。

CLI AWS SAM 也包含各種有用的功能，例如自動程式碼部署、記錄和偵錯功能。它可讓開發人員使用單一命令建置、封裝和部署其應用程式，減少部署和管理無伺服器應用程式所需的時間和精力。

此外，AWS SAM CLI 支援各種程式設計語言，包括 Node.js、Python、Java 和 .NET Core 等。這可讓開發人員使用他們偏好的程式設計語言和工具來建置和部署其無伺服器應用程式。

AWS SAM CLI 與其他 AWS 服務整合，例如 AWS CodePipeline 和 AWS CodeBuild，為無伺服器應用程式提供全自動化且整合的 CI/CD 管道。它還允許開發人員使用其他 AWS 服務，例如 Amazon S3、Amazon DynamoDB 和 Amazon SNS，作為其無伺服器應用程式的一部分。

AWS Cloud Development Kit (AWS CDK)

AWS Cloud Development Kit (AWS CDK) (AWS CDK) 是一種開放原始碼軟體開發架構，可將雲端基礎設施定義為具有現代程式設計語言的程式碼，並透過 AWS CloudFormation 部署。AWS Cloud Development Kit (AWS CDK) 使用常見的程式設計語言來為您的應用程式建立模型，以加速雲端開發。AWS CDK 可讓您在雲端中建置可靠、可擴展且符合成本效益的應用程式，並具有程式設計語言的可觀表達能力。

將 AWS CDK 視為以開發人員為中心的工具組，利用現代程式設計語言的完整功能，將 AWS 基礎設施定義為程式碼。執行 AWS CDK 應用程式時，會向下編譯至完整格式的 CloudFormation JSON/YAML 範本，然後提交至 CloudFormation 服務進行佈建。由於 AWS CDK 利用 CloudFormation，您仍然享有 CloudFormation 提供的所有好處，例如安全部署、自動復原和偏離偵測。

這種方法產生許多好處，包括：

- 使用高階建構來建置 ，自動為您的 AWS 資源提供明智且安全的預設值，以較少的程式碼定義更多基礎設施。
- 使用程式設計範例，例如參數、條件、迴圈、合成和繼承，從 AWS 和其他 提供的建置區塊建立您的系統設計的模型。
- 將您的基礎設施、應用程式程式碼和組態全部放在一個位置，確保您在每個里程碑都有完整的雲端可部署系統。
- 採用程式碼檢閱、單元測試和來源控制等軟體工程實務，讓您的基礎設施更強大。
- AWS Solutions Constructs 是 AWS CDK 的開放原始碼程式庫延伸。AWS Solutions Constructs 為您提供一系列使用 AWS Well-Architected Framework 建立的最佳實務所建置的經過審核的多服務架構模式。

AWS Serverless Application Model 和 AWS CDK 都會以程式碼的形式抽象化 AWS 基礎設施，讓您更輕鬆地定義雲端基礎設施。AWS SAM 特別專注於無伺服器使用案例和架構，可讓您在精簡的宣告式 JSON/YAML 範本中定義基礎設施。AWS CDK 為所有 AWS 服務提供廣泛的涵蓋範圍，並可讓您以現代程式設計語言定義雲端基礎設施

Amazon EC2 Image Builder

EC2 Image Builder 可簡化建置、測試和部署 VM 和容器映像，以用於 AWS 或內部部署。將 VM 和容器映像保持在up-to-date可能會耗時、資源密集且容易出錯。目前，客戶可以手動更新和快照 VMs，也可以讓團隊建置自動化指令碼來維護映像。Image Builder 提供簡單的圖形界面、內建自動化和 AWS 提供的安全設定，可大幅減少讓映像保持up-to-date狀態和安全的工作量。使用映像建置器，您不需要手動更新映像的步驟，也不需要建置自己的自動化管道。除了用於建立、存放和共用映像的基礎 AWS 資源成本之外，映像建置器可免費提供。

EC2 Image Builder 可以簡化建立和管理自訂映像的程序，以便與 Amazon EC2、容器和內部部署伺服器搭配使用，從而在 AWS 上簡化部署。此服務提供簡化且靈活的方法來建立和管理自訂映像，搭配自動化建置管道，可讓您簡化映像建立和管理程序。

EC2 Image Builder 提供易於使用的界面，可抽象化大部分的基礎基礎設施管理，讓開發人員更輕鬆地建立和管理自訂映像。使用 EC2 Image Builder，開發人員可以指定要包含在映像中的作業系統、應用

程式和套件，且該服務可自動化建置和測試映像的程序，包括更新、修補程式和安全性修正。自動化建置管道可讓開發人員簡化映像建立和管理程序，減少手動映像建立和測試所需的時間和精力。這有助於提高一致性、減少錯誤，並確保映像up-to-date、安全且合規的。

以下是 EC2 Image Builder 的一些優點：

- 簡化的映像建立：EC2 Image Builder 提供簡化且靈活的方法來建立自訂映像，以便與 Amazon EC2、容器和內部部署伺服器搭配使用。這有助於減少建立和維護自訂映像所需的時間和精力，並讓您能夠專注於部署的其他層面，例如應用程式開發和測試。
- 自動化映像建置管道：EC2 Image Builder 提供自動化管道來建置、測試和部署自訂映像，這有助於簡化映像建立和管理程序。這有助於確保您的映像是up-to-date、安全且合規的，並減少手動建立和測試映像所需的時間和精力。
- 與 AWS 服務整合：EC2 Image Builder 與其他 AWS 服務整合，例如 Amazon Elastic Container Registry (ECR) 和 Amazon Elastic Kubernetes Service (EKS)，可讓您建置自訂映像以搭配容器使用。這有助於簡化容器建置和部署程序，讓您能夠建置包含應用程式、程式庫和組態的自訂映像。
- 彈性映像建立：EC2 Image Builder 提供建立自訂映像的彈性方式，可讓您指定要包含在映像中的作業系統、應用程式和套件。這有助於確保您的映像是根據您的特定使用案例和需求量身打造，並降低部署期間發生錯誤或不相容的風險。
- 改善映像安全性和合規性：EC2 Image Builder 可讓您自動化映像測試，包括漏洞和合規掃描，以確保您的映像安全且合規。這有助於降低安全漏洞的風險並改善合規性，並讓您能夠放心地部署應用程式。

部署策略

除了選擇正確的工具來更新您的應用程式程式碼和支援基礎設施之外，實作正確的部署程序是完整、功能良好的部署解決方案的關鍵部分。您選擇更新應用程式的部署程序，取決於您想要的控制、速度、成本、風險承受能力和其他因素平衡。

每個 AWS 部署服務都支援多種部署策略。本節將概述可與部署解決方案搭配使用的一般用途部署策略。

預製與引導 AMIs

如果您的應用程式高度依賴於將應用程式自訂或部署到 Amazon EC2 執行個體，則您可以透過引導和預先封裝實務來最佳化部署。

每當啟動 Amazon EC2 執行個體時，安裝您的應用程式、相依性或自訂稱為引導執行個體。如果您有複雜的應用程式或需要大量下載，這可能會降低部署和擴展事件的速度。

Amazon Machine Image (AMI) 提供啟動執行個體（作業系統、儲存磁碟區、許可、軟體套件等）所需的資訊。您可以從單一 AMI 啟動多個相同的執行個體。每當啟動 EC2 執行個體時，您都會選取要用作範本的 AMI。Prebaking 是在 AMI 中嵌入大部分應用程式成品的程序。

將應用程式元件預先封裝到 AMI 可以加快啟動和操作 Amazon EC2 執行個體的時間。在部署過程中可以結合預製和引導實務，以快速建立針對目前環境自訂的新執行個體。

藍/綠部署

藍/綠部署是一種部署策略，您可以在其中建立兩個不同但相同的環境。一個環境（藍色）正在執行目前的應用程式版本，一個環境（綠色）正在執行新的應用程式版本。使用藍/綠部署策略可提高應用程式可用性，並在部署失敗時簡化復原程序，進而降低部署風險。在綠色環境中完成測試後，即時應用程式流量會導向綠色環境，並取代藍色環境。

許多 AWS 部署服務支援藍/綠部署策略，包括 Elastic Beanstalk、OpsWorks、CloudFormation、CodeDeploy 和 Amazon ECS。如需為應用程式實作藍/綠部署程序的詳細資訊和策略，請參閱 AWS 上的藍/綠部署。

滾動部署

滾動部署是一種部署策略，透過完全取代應用程式執行所在的基礎設施，將舊版應用程式慢慢取代為新版應用程式。例如，在 Amazon ECS 的滾動部署中，執行舊版應用程式的容器將one-by-one取代為執行新版應用程式的容器。

滾動部署通常比藍/綠部署更快；但是，與藍/綠部署不同，在滾動部署中，舊應用程式和新應用程式版本之間沒有環境隔離。這可讓滾動部署更快完成，但也會增加風險，並在部署失敗時使復原程序複雜化。

滾動部署策略可與大多數部署解決方案搭配使用。如需使用 CloudFormation 滾動部署的詳細資訊，請參閱 CloudFormation 更新政策；使用 Amazon ECS 滾動更新，以取得使用 Amazon ECS 滾動部署的詳細資訊；Elastic Beanstalk 滾動環境組態更新，以取得使用 Elastic Beanstalk 滾動部署的詳細資訊；以及在 中使用滾動部署 AWS OpsWorks，以取得使用 OpsWorks 滾動部署的詳細資訊。CloudFormation

Canary 部署

Canary 部署是一種藍/綠部署策略，風險更大。此策略涉及分階段方法，其中流量會以兩個增量轉移到應用程式的新版本。第一個增量是一小部分的流量，稱為 Canary 群組。此群組用於測試新版本，如果成功，流量會以第二個增量轉移到新版本。

Canary 部署可以透過兩個步驟或線性方式實作。在兩步驟方法中，會部署並公開新的應用程式程式碼以供試用。接受時，它會以線性方式推展到環境的其餘部分。線性方法涉及逐漸增加應用程式新版本的流量，直到所有流量流向新版本為止。

就地部署：

就地部署是一種部署策略，可在不取代任何基礎設施元件的情況下更新應用程式版本。在就地部署中，會停止每個運算資源上先前的應用程式版本、安裝最新的應用程式，並啟動和驗證應用程式的新版本。這可讓應用程式部署以對基礎基礎設施的最小干擾繼續。

就地部署可讓您在不建立新基礎設施的情況下部署應用程式；不過，在這些部署期間可能會影響應用程式的可用性。此方法也會將與建立新資源相關的基礎設施成本和管理開銷降至最低。

如需搭配 CodeDeploy 使用就地部署策略的詳細資訊，請參閱就地部署的概觀。

結合部署服務

AWS 上沒有適合所有部署解決方案的「單一大小」。在設計部署解決方案的情況下，請務必考慮應用程式的類型，因為這可以決定最適合的 AWS 服務。若要提供完整功能來佈建、設定、部署、擴展和監控您的應用程式，通常需要合併多個部署服務

AWS 應用程式常見的模式是使用 CloudFormation （及其擴充功能） 來管理一般用途的基礎設施，並使用更專業的部署解決方案來管理應用程式更新。在容器化應用程式的情況下，CloudFormation 可用於建立應用程式基礎設施，而 Amazon ECS 和 Amazon EKS 可用於佈建、部署和監控容器。

AWS 部署服務也可以與第三方部署服務結合。這可讓組織輕鬆將 AWS 部署服務整合至其現有的 CI/CD 管道或基礎設施管理解決方案。例如，OpsWorks 可用來同步內部部署和 AWS 節點之間的組態，而 CodeDeploy 可與許多第三方 CI/CD 服務搭配使用，做為完整管道的一部分。

結論

AWS 提供許多工具來簡化和自動化應用程式的基礎設施佈建和部署；每個部署服務都提供管理應用程式的不同功能。若要建置成功的部署架構，請根據您的應用程式和組織需求，評估每個服務的可用功能。

貢獻者

本文件的貢獻者包括：

- Manikandan Chandrasekaran，首席技術人員
- Anil Nadiminti，資深解決方案架構師
- Bryant Bost，AWS ProServe 顧問

深入閱讀

如需其他資訊，請參閱：

- AWS 白皮書頁面
- DevOps on AWS 簡介 - 部署策略

文件修訂

若要收到此白皮書更新的通知，請訂閱 RSS 摘要。

變更	描述	日期
白皮書已更新	針對最新的部署服務和策略更新整個	2024 年 5 月 31 日
次要更新	為了清楚起見，已修訂藍/綠部署章節。	2021 年 4 月 8 日
白皮書已更新	更新了最新的服務和功能。	2020 年 6 月 3 日
初次出版	白皮書首次發佈	2015 年 3 月 1 日

注意

客戶有責任對本文件中的資訊進行自己的獨立評定。本文件：(a) 僅供參考，(b) 代表目前的 AWS 產品產品和實務，如有變更，恕不另行通知，且 (c) 不會從 AWS 及其附屬公司、供應商或授權方建立任何承諾或保證。AWS 產品或服務以「原樣」提供，不做任何明示或暗示的保證、表示或條件。AWS 對其客戶的責任與義務應由 AWS 協議管轄，本文並非 AWS 與其客戶之間的任何協議的一部分，也並非上述協議的修改。

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。