AWS 白皮書

# AWS 上的 DevOps 簡介

# AWS 上的 DevOps 簡介: AWS 白皮書

# Table of Contents

# AWS 上的 DevOps 簡介

發佈日期：2023 年 4 月 7 日 (文件修訂)

如今，企業比以往更開始數位轉型之旅，與客戶建立更深入的聯繫，以實現永續且持久的商業價值。所有形狀和大小的組織都比以往更快地創新，從而打斷了其競爭對手並進入新市場。對於這些組織而言，專注於創新和軟體中斷非常重要，因此簡化其軟體交付至關重要。縮短從想法到生產時間的組織，讓速度和敏捷性成為明天的干擾因素。

雖然在成為下一個數位干擾因素時需要考慮幾個因素，但此白皮書著重於 DevOps，以及 Amazon Web Services (AWS) 平台中的服務和功能，這將有助於提高組織以高速交付應用程式和服務的能力。

## 簡介

DevOps 是文化理念、工程實務和工具的組合，可提高組織以高速和更好的品質交付應用程式和服務的能力。隨著時間的推移，在採用 DevOps 時出現了幾個基本實務：持續整合 (CI)、持續交付 (CD)、基礎設施即程式碼 (IaC)，以及監控和記錄。

本白皮書重點介紹可協助您加速 DevOps 旅程 AWS 的功能，以及 AWS 服務如何協助消除與 DevOps 適應性相關的無差別繁重工作。它還說明如何在不管理伺服器或建置節點的情況下建立持續整合和交付功能，以及如何使用 IaC 以一致且可重複的方式佈建和管理雲端資源。

- 持續整合：一種軟體開發實務，開發人員會定期將其程式碼變更合併到中央儲存庫，之後會執行自動化建置和測試。
- 持續交付：一種軟體開發實務，可自動建置、測試和準備程式碼變更，以用於生產版本。
- 基礎設施即程式碼：一種使用程式碼和軟體開發技術佈建和管理基礎設施的實務，例如版本控制和持續整合。
- 監控和記錄：讓組織了解應用程式和基礎設施效能如何影響其產品最終使用者的體驗。
- 溝通和協作：建立實務，透過建立工作流程並分配 DevOps 的責任，讓團隊更緊密。
- 安全性：應該是交叉切削的考量。您的持續整合和持續交付 (CI/CD) 管道和相關服務應受到保護，並應設定適當的存取控制許可。

對這些原則的檢查顯示與可用方案的密切關聯 AWS。

# 您是 Well-Architected 嗎？

AWS Well-Architected Framework 可協助您了解在雲端建置系統時所做決策的優缺點。架構的六大支柱可讓您學習架構最佳實務，以設計和操作可靠、安全、有效率、經濟實惠且永續的系統。使用 AWS Well-Architected Tool，在 AWS 管理主控台中免費提供，您可以透過回答每個支柱的一組問題，根據這些最佳實務來檢閱工作負載。

# 持續整合

持續整合 (CI) 是一種軟體開發實務，開發人員會定期將其程式碼變更合併到中央程式碼儲存庫，之後會執行自動化建置和測試。CI 有助於更快地尋找和解決錯誤、改善軟體品質，並減少驗證和發佈新軟體更新所需的時間。

AWS 提供下列服務以進行持續整合：

主題

- AWS CodeCommit
- AWS CodeBuild
- AWS CodeArtifact

## AWS CodeCommit

AWS CodeCommit 是一項安全、具高度可擴展性的受管來源控制服務，可託管私有 Git 儲存庫。CodeCommit 可降低您操作自有來源控制系統的需求，而且無需佈建和擴展硬體，也無需安裝、設定和操作軟體。您可以使用 CodeCommit 來存放從程式碼到二進位檔的任何內容，並支援 GitHub 的標準功能，使其能夠與您現有的 Git 型工具無縫搭配運作。您的團隊也可以使用 CodeCommit 的線上程式碼工具來瀏覽、編輯和協作專案。 AWS CodeCommit 有幾個優點：

- 協作 - AWS CodeCommit 專為協作軟體開發而設計。您可以輕鬆遞交、分支和合併程式碼，這可協助您輕鬆維持對團隊專案的控制。CodeCommit 也支援提取請求，提供請求程式碼檢閱的機制，並與協作者討論程式碼。
- 加密 — 您可以視需要 AWS CodeCommit 使用 HTTPS 或 SSH 將檔案傳輸到 和 。您的儲存庫也會使用客戶特定的金鑰，透過 AWS Key Management Service(AWS KMS) 自動進行靜態加密。
- 存取控制 - AWS CodeCommit 使用 AWS Identity and Access Management(IAM) 來控制和監控除了可存取資料的方式、時間和位置之外，還有誰可以存取您的資料。CodeCommit 也可協助您透過 AWS CloudTrail和 Amazon CloudWatch 監控儲存庫。

  高可用性和耐用性 — 將您的儲存庫 AWS CodeCommit 存放在 Amazon Simple Storage Service (Amazon S3) 和 Amazon DynamoDB 中。您的加密資料會以備援方式儲存在多個設施中。此架構可提高儲存庫資料的可用性和耐久性。
- 通知和自訂指令碼 — 您現在可以接收影響儲存庫的事件通知。通知將以 Amazon Simple Notification Service (Amazon SNS) 通知的形式提供。每個通知都會包含狀態訊息，以及其事件產生

該通知之資源的連結。此外，使用 AWS CodeCommit 儲存庫提示，您可以使用 Amazon SNS 傳送通知和建立 HTTP Webhook，或叫用 AWS Lambda函數以回應您選擇的儲存庫事件。

## AWS CodeBuild

AWS CodeBuild 是一種全受管的持續整合服務，可編譯原始程式碼、執行測試，並產生可立即部署的軟體套件。您不需要佈建、管理和擴展自己的建置伺服器。CodeBuild 可以使用 GitHub、GitHub Enterprise AWS CodeCommit、BitBucket 或 Amazon S3 作為來源提供者。

CodeBuild 會持續擴展，並可同時處理多個組建。CodeBuild 為各種版本的 Microsoft Windows 和 Linux 提供各種預先設定的環境。客戶也可以將自訂建置環境做為 Docker 容器。CodeBuild 也與開放原始碼工具整合，例如 Jenkins 和 Spinnaker。

CodeBuild 也可以建立單位、功能或整合測試的報告。這些報告提供已執行的測試案例數量，以及通過或失敗案例數量的視覺化檢視。建置程序也可以在 Amazon Virtual Private Cloud (Amazon VPC) 內執行，如果您的整合服務或資料庫部署在 VPC 內，這可能會有所幫助。

## AWS CodeArtifact

AWS CodeArtifact 是一種全受管成品儲存庫服務，組織可以使用此服務來安全地存放、發佈和共用軟體開發程序中使用的軟體套件。CodeArtifact 可設定為從公有成品儲存庫自動擷取軟體套件和相依性，以便開發人員可以存取最新版本。

軟體開發團隊越來越依賴開放原始碼套件在其應用程式套件中執行常見任務。軟體開發團隊必須持續控制特定版本的開放原始碼軟體，以確保軟體沒有漏洞。使用 CodeArtifact，您可以設定控制項來強制執行此操作。

CodeArtifact 可與常用的套件管理員和建置工具搭配使用，例如
Maven、Gradle、npm、Lunel、winbe 和 pip，讓您輕鬆整合到現有的開發工作流程。

# 持續交付

持續交付 (CD) 是一種軟體開發實務，可自動準備程式碼變更以發佈至生產環境。作為現代應用程式開發的支柱，透過在建置階段之後將所有程式碼變更部署到測試環境和/或生產環境，持續交付會在持續整合時擴展。正確實作時，開發人員一律會有部署就緒建置成品，該成品已通過標準化測試程序。

持續交付可讓開發人員自動化測試，而不只是單元測試，因此他們可以在部署給客戶之前驗證多個維度的應用程式更新。

這些測試可能包括 UI 測試、負載測試、整合測試、API 可靠性測試等。這有助於開發人員更徹底地驗證更新，並先發發現問題。使用雲端，自動化建立和複寫多個環境進行測試非常簡單且經濟實惠，而這些環境先前很難在內部部署執行。

AWS 提供下列服務以持續交付：

- AWS CodeBuild
- AWS CodeDeploy
- AWS CodePipeline

主題

- AWS CodeDeploy
- AWS CodePipeline

## AWS CodeDeploy

AWS CodeDeploy 是一種全受管部署服務，可自動化軟體部署到各種運算服務，例如 Amazon Elastic Compute Cloud (Amazon EC2)AWS Fargate AWS Lambda、 和您的內部部署伺服器。 可讓您 AWS CodeDeploy 更輕鬆地快速發行新功能、協助您避免應用程式部署期間的停機時間，並處理更新應用程式的複雜性。您可以使用 CodeDeploy 自動化軟體部署，減少對易出錯手動操作的需求。服務擴展以符合您的部署需求。

CodeDeploy 有幾個優點，符合持續部署的 DevOps 原則：

- 自動化部署 — CodeDeploy 完全自動化軟體部署，可讓您可靠快速地部署。
- 集中式控制 — CodeDeploy 可讓您透過 AWS 管理主控台 或 輕鬆啟動和追蹤應用程式部署的狀態 AWS CLI。CodeDeploy 為您提供詳細的報告，可讓您檢視每個應用程式修訂版的部署時間和目的地。您也可以建立推送通知，以接收部署的即時更新。

- 將停機時間降到最低 — CodeDeploy 有助於在軟體部署過程中最大化您的應用程式可用性。它會逐步引入變更，並根據可設定的規則追蹤應用程式運作狀態。發生錯誤時，可以輕鬆停止和復原軟體部署。
- 易於採用 — CodeDeploy 可與任何應用程式搭配使用，並在不同的平台和語言提供相同的體驗。您可以輕鬆重複使用現有的設定程式碼。CodeDeploy 也可以與您現有的軟體版本程序或持續交付工具鏈 （例如 GitHub AWS CodePipeline、Jenkins) 整合。

AWS CodeDeploy 支援多個部署選項。如需詳細資訊，請參閱本文件的部署策略一節。

# AWS CodePipeline

AWS CodePipeline 是一項持續交付服務，可用來建立模型、視覺化和自動化發行軟體所需的步驟。使用 AWS CodePipeline，您可以建立建置程式碼、部署至生產前環境、測試應用程式，以及將其發佈至生產環境的完整發行程序。 AWS CodePipeline 然後，每次變更程式碼時，都會根據定義的工作流程建置、測試和部署您的應用程式。您可以將合作夥伴工具和自己的自訂工具整合到發行程序的任何階段，以形成end-to-end持續交付解決方案。

AWS CodePipeline 有幾項優點符合持續部署的 DevOps 原則：

- 快速交付： AWS CodePipeline 自動化您的軟體版本程序，讓您快速地將新功能發佈給使用者。透過 CodePipeline，您可以快速迭代意見回饋，並更快地為使用者取得新功能。
- 改善品質 — 透過自動化建置、測試和發行程序， AWS CodePipeline 您可以透過一致的品質檢查執行所有新變更，來提高軟體更新的速度和品質。
- 易於整合 - AWS CodePipeline 可以輕鬆擴展以適應您的特定需求。您可以在發行程序的任何步驟中使用預先建置的外掛程式或您自己的自訂外掛程式。例如，您可以從 GitHub 提取原始程式碼、使用內部部署 Jenkins 建置伺服器、使用第三方服務執行負載測試，或將部署資訊傳遞至自訂操作儀表板。
- 可設定的工作流程 - AWS CodePipeline 可讓您使用主控台界面、CloudFormation、 或 AWS AWS CLI SDKs，建立軟體版本程序的不同階段模型。您可以輕鬆指定要執行的測試，並自訂部署應用程式及其相依性的步驟。

# 部署策略

部署策略會定義您要如何交付軟體。組織會根據其商業模式遵循不同的部署策略。有些選擇交付經過完整測試的軟體，有些則可能希望使用者提供意見回饋，並讓其使用者在開發功能（例如 Beta 版）下進行評估。下節討論各種部署策略。

## 就地部署：

在此策略中，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。透過就地部署，您可以部署應用程式，而無需建立新的基礎設施；不過，在這些部署期間，應用程式的可用性可能會受到影響。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。您可以使用負載平衡器，讓每個執行個體在部署期間取消註冊，然後在部署完成後還原至服務。就地部署可以是all-at-once部署、假設服務中斷，或作為滾動更新完成。 AWS CodeDeploy 而 AWS Elastic Beanstalk 提供one-at-a-time、half-at-a-time和all-at-once部署組態。

## 藍/綠部署

藍/綠部署，有時稱為紅/黑部署，是一種透過在執行應用程式不同版本之兩個相同環境之間轉移流量來釋放應用程式的技術。藍/綠部署可協助您在應用程式更新期間將停機時間降至最低，降低停機時間和回復功能的風險。

藍/綠部署可讓您啟動應用程式的新版本（綠色）與舊版本（藍色），並在重新路由流量到新版本之前監控和測試新版本，並在問題偵測時復原。

## Canary 部署

Canary 部署的目的是降低部署會影響工作負載的新版本的風險。方法會逐步部署新版本，讓新使用者以緩慢的方式看見。隨著您對部署的信心，您將部署它來取代目前的版本。

## 線性部署

線性部署表示以相等增量移動流量，每個增量之間的分鐘數相等。您可從預先指定的線性選項中指定每次增量的流量轉移百分比例，以及在每個增量之間的分鐘數。

# All-at-once部署

All-at-once部署表示所有流量都會一次從原始環境轉移到替代環境。

# 部署策略矩陣

下列矩陣列出 Amazon Elastic Container Service (Amazon ECS) AWS Lambda和 Amazon EC2/內部部署支援的部署策略。

- Amazon ECS 是全受管協同運作服務。
- AWS Lambda 可讓您執行程式碼，而無需佈建或管理伺服器。
- Amazon EC2 可讓您在雲端中執行安全、可調整大小的運算容量。

| 部署策略 | Amazon ECS | AWS Lambda | Amazon EC2/內部部署 | |
|---|---|---|---|---|
| 就地 | ✓ | ✓ | ✓ | |
| 藍/綠 | ✓ | ✓ | ✓* | |
| Canary | ✓ | ✓ | X | |
| 線性 | ✓ | ✓ | X | |
| All-at-once | ✓ | ✓ | X | |

> **Note**
>
> 藍/綠部署搭配 EC2/內部部署僅適用於 EC2 執行個體。

## AWS Elastic Beanstalk 部署策略

AWS Elastic Beanstalk 支援以下類型的部署策略：

- All-at-once 在所有執行個體上執行就地部署。
- 滾動 將執行個體分割成批次，並一次部署至一個批次。
- 使用其他批次滾動 將部署分割為批次，但第一個批次會建立新的 EC2 執行個體，而不是部署在現有的 EC2 執行個體上。

- 不可變 如果您需要使用新執行個體進行部署，而不是使用現有的執行個體。
- 流量分割 執行不可變的部署，然後在預先確定的持續時間內將流量百分比轉送至新執行個體。如果執行個體正常運作，請將所有流量轉送至新的執行個體，並關閉舊的執行個體。

# 基礎設施即程式碼

DevOps 的基本原則是以開發人員處理程式碼的方式對待基礎設施。應用程式碼具有定義的格式和語法。如果程式碼不是根據程式設計語言的規則撰寫，則無法建立應用程式。程式碼存放在版本管理或來源控制系統中，記錄程式碼開發、變更和錯誤修正的歷史記錄。當程式碼編譯或內建於應用程式中時，我們預期會建立一致的應用程式，而且建置可重複且可靠。

將基礎設施視為程式碼表示將相同的嚴格應用程式程式碼開發套用至基礎設施佈建。所有組態應以宣告方式定義，並存放在來源控制系統中，例如 AWS CodeCommit，與應用程式碼相同。基礎設施佈建、協同運作和部署也應支援使用基礎設施做為程式碼。

傳統上，基礎設施是使用指令碼和手動程序的組合進行佈建。有時候，這些指令碼會存放在版本控制系統中，或逐步記錄在文字檔案或執行手冊中。撰寫執行手冊的人員通常與執行這些指令碼或遵循執行手冊的人員不同。如果這些指令碼或 Runbook 不頻繁更新，它們可能會在部署中成為顯示停止。這會導致新環境的建立不一定是可重複、可靠或一致的。

相反地， AWS 提供以 DevOps 為重心的方法來建立和維護基礎設施。與軟體開發人員編寫應用程式程式碼的方式類似， AWS 提供以程式設計、描述性和宣告方式啟用基礎設施建立、部署和維護的服務。這些服務提供嚴謹、清晰和可靠性。本文討論 AWS 的服務是 DevOps 方法的核心，並構成許多高階 AWS DevOps 原則和實務的基礎。

AWS 提供下列 服務，將基礎設施定義為程式碼。

服務

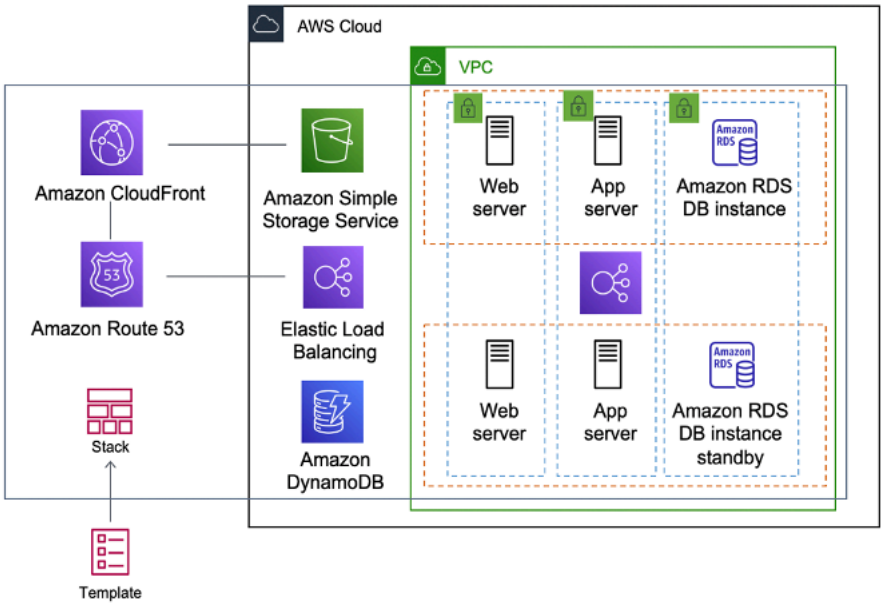
- CloudFormation
- AWS Serverless Application Model
- AWS Cloud Development Kit (AWS CDK)
- 適用於 Kubernetes 的 AWS 雲端開發套件
- 適用於 Terraform 的 AWS 雲端開發套件
- AWS 雲端控制 API

## CloudFormation

AWS CloudFormation 是一項服務，可讓開發人員以有序且可預測的方式建立 AWS 資源。資源會使用 JSON 或 YAML 格式以文字檔撰寫。範本需要特定語法和結構，而這取決於所建立和管理的資源類

型。您可以使用任何程式碼編輯器 (例如 AWS Cloud9) 以 JSON 或 YAML 撰寫資源、將其簽入版本控制系統，然後 CloudFormation 便會以安全、可重複的方式建置指定的服務。

CloudFormation 範本會以堆疊的形式部署到 AWS 環境中。您可以透過 AWS 管理主控台 AWS Command Line Interface、 或 CloudFormation APIs管理堆疊。如果您需要變更堆疊中執行的資源，請更新堆疊。在變更資源之前，您可以產生變更集以列出請求變更的摘要。變更集可讓您在實作變更之前，了解變更如何影響您的執行中資源，尤其是關鍵資源。




AWS CloudFormation 從一個範本建立整個環境 （ 堆疊 ）

您可以使用單一範本來建立和更新整個環境，或是使用個別範本來管理環境中的多層。這可讓範本模組化，也提供對許多組織很重要的控管層。

當您在 CloudFormation 主控台中建立或更新堆疊時，會顯示事件，顯示組態的狀態。如果發生錯誤，依預設，堆疊會復原至先前的狀態。Amazon SNS 會提供事件的通知。例如，您可以使用 Amazon SNS 來追蹤使用電子郵件建立和刪除堆疊的進度，並以程式設計方式與其他程序整合。

AWS CloudFormation 可讓您輕鬆地組織和部署 AWS 資源集合，並讓您在設定堆疊時描述任何相依性或傳遞特殊參數。

透過 CloudFormation 範本，您可以使用廣泛的 AWS 服務，例如 Amazon S3、Auto Scaling、Amazon CloudFront、Amazon DynamoDB、Amazon EC2、Amazon ElastiCache AWS Elastic Beanstalk、Elastic Load Balancing、IAM、AWS OpsWorks 和 Amazon VPC。如需支援資源的最新清單，請參閱AWS 資源和屬性類型參考。

# AWS Serverless Application Model

AWS Serverless Application Model (AWS SAM) 是一種開放原始碼架構，可用來建置無伺服器應用程式 AWS。

AWS SAM 與其他 AWS 服務整合，因此使用 AWS SAM 建立無伺服器應用程式可提供下列優點：

- 單一部署組態 — AWS SAM 可讓您輕鬆地組織相關的元件和資源，並在單一堆疊上操作。您可以使用 在資源之間 AWS SAM 共用組態（例如記憶體和逾時），並將所有相關資源部署為單一版本控制的實體。
- 延伸： CloudFormation因為 AWS SAM 是 的延伸 CloudFormation，所以您可以取得 的可靠部署功能 CloudFormation。您可以在 AWS SAM 範本 CloudFormation 中使用 來定義資源。
- 內建最佳實務 — 您可以使用 AWS SAM 來定義和部署 IaC。這可讓您使用和強制執行最佳實務，例如程式碼檢閱。

# AWS Cloud Development Kit (AWS CDK)

是一種開放原始碼軟體開發架構，可讓您使用熟悉AWS Cloud Development Kit (AWS CDK)的程式設計語言來建模和佈建雲端應用程式資源。 AWS CDK 可讓您使用 TypeScript、Python、Java 和 .NET 來建模應用程式基礎設施。開發人員可以使用自動完成和內嵌文件等工具，利用現有的整合式開發環境 (IDE) 來加速基礎設施的開發。

AWS CDK CloudFormation 在背景使用 ，以安全且可重複的方式佈建資源。建構是 CDK 程式碼的基本建置區塊。建構代表雲端元件，並封裝建立元件 CloudFormation 所需的一切。 AWS CDK 包含 AWS 建構程式庫，其中包含代表許多 AWS 服務的建構。透過將建構模組結合在一起，您可以快速輕鬆地建立複雜的架構以在 中部署 AWS。

# 適用於 Kubernetes 的 AWS 雲端開發套件

適用於 Kubernetes 的 AWS 雲端開發套件是一種開放原始碼軟體開發架構，可使用一般用途程式設計語言定義 Kubernetes 應用程式。

以程式設計語言定義應用程式後（截至本發佈日期，僅支援 Python 和 TypeScript)，cdk8s 會將 中的應用程式描述轉換為預先 Kubernetes YAML。然後，任何在任何地方執行的 Kubernetes 叢集都可以使用此 YAML 檔案。由於結構是以程式設計語言定義，因此您可以使用程式設計語言提供的豐富功能。您可以使用程式設計語言的抽象功能來建立自己的樣板程式碼，並在所有部署中重複使用。

# 適用於 Terraform 的 AWS 雲端開發套件

CDK for Terraform (CDKTF) 建立在開放原始碼 JSII 程式庫之上，可讓您在您選擇的 C#、Python、TypeScript、Java 或 Go 中撰寫 Terraform 組態，但仍受益於 Terraform 提供者和模組的完整生態系統。您可以將任何現有的提供者或模組從 Terraform 登錄檔匯入您的應用程式，而 CDKTF 將產生資源類別，供您在目標程式設計語言中與 互動。

使用 CDKTF，開發人員可以設定 IaC，而無需從熟悉的程式設計語言切換內容，使用相同的工具和語法來佈建類似應用程式商業邏輯的基礎設施資源。團隊可以熟悉的語法進行協作，同時仍然使用 Terraform 生態系統的強大功能，並透過已建立的 Terraform 部署管道部署其基礎設施組態。

# AWS 雲端控制 API

AWS 雲端控制 API 是一項新 AWS 功能，引進一組常見的建立、讀取、更新、刪除和清單 (CRUDL) APIs，協助開發人員以簡單且一致的方式管理其雲端基礎設施。Cloud Control API 通用 APIs 可讓開發人員統一管理 AWS 和第三方服務的生命週期。

身為開發人員，您可能偏好簡化管理所有資源生命週期的方式。您可以使用 Cloud Control API 的統一資源組態模型搭配預先定義的格式來標準化雲端資源組態。此外，您將受益於統一 API 行為 （回應元素和錯誤），同時管理您的 資源。

例如，您會發現在 CRUDL 操作期間，透過與操作資源無關的 Cloud Control API 所呈現的統一錯誤代碼，可以輕鬆地偵錯錯誤。使用 Cloud Control API，您也可以輕鬆設定跨資源相依性。您也不再需要跨多個廠商工具和 APIs 撰寫和維護自訂程式碼，以一起使用 AWS 和第三方資源。

# 自動化和工具

DevOps 的另一個核心理念和實務是自動化。自動化著重於基礎設施及其上執行的應用程式的設定、組態、部署和支援。透過使用自動化，您可以以標準化且可重複的方式更快速地設定環境。移除手動程序是成功 DevOps 策略的關鍵。從歷史上看，伺服器組態和應用程式部署主要是手動程序。環境變得非標準，在發生問題時重新產生環境並不容易。

使用自動化對於實現雲端的完整優勢至關重要。在內部，AWS 非常依賴自動化，以提供彈性和可擴展性的核心功能。

手動程序容易出錯、不可靠且不足以支援敏捷業務。通常，組織可能會綁定高技能的資源，以提供手動組態，因為在業務中花費的時間可能更長支援其他、更關鍵和更高價值的活動。

現代操作環境通常依賴完全自動化，以消除手動介入或對生產環境的存取。這包括所有軟體發行、機器組態、作業系統修補、疑難排解或錯誤修正。許多層級的自動化實務可以一起使用，以提供更高層級 end-to-end自動化程序。

自動化有下列主要優點：

- 快速變更
- 提高生產力
- 可重複的組態
- 可重現的環境
- 彈性
- 自動調整規模
- 自動化測試

自動化是 AWS 服務的基石，而且在所有 服務、功能和產品中都受到內部支援。

主題

- AWS OpsWorks
- AWS Elastic Beanstalk
- EC2 Image Builder
- AWS Proton
- AWS Service Catalog
- AWS Cloud9

- AWS CloudShell
- Amazon CodeGuru

# AWS OpsWorks

AWS OpsWorks 更進一步採用 DevOps 的原則 AWS Elastic Beanstalk。它可以被視為應用程式管理服務，而不只是應用程式容器。 OpsWorks 提供更多層級的自動化功能，例如與組態管理軟體 (Chef) 和應用程式生命週期管理整合。您可以使用應用程式生命週期管理來定義何時設定、設定、部署、取消部署或結束資源。

為了提高靈活性 AWS OpsWorks ，您可以在可設定的堆疊中定義應用程式。您也可以選取預先定義的應用程式堆疊。應用程式堆疊包含應用程式所需的 AWS 資源的所有佈建，包括應用程式伺服器、Web 伺服器、資料庫和負載平衡器。

應用程式堆疊會組織成架構層，以便可以獨立維護堆疊。範例層可能包括 Web 層、應用程式層和資料庫層。AWS OpsWorks 現成也可簡化 AWS Auto Scaling 群組和 Elastic Load Balancing (ELB) 負載平衡器的設定，進一步說明自動化的 DevOps 原則。如同 AWS Elastic Beanstalk，AWS OpsWorks 支援應用程式版本控制、持續部署和基礎設施組態管理


OpsWorks 顯示 DevOps 功能和架構

AWS OpsWorks 也支援監控和記錄的 DevOps 實務 （請參閱下一節）。監控支援由 Amazon CloudWatch 提供。系統會記錄所有生命週期事件，而個別的 Chef 日誌會記錄執行的任何 Chef 配方，以及任何例外狀況。

# AWS Elastic Beanstalk

AWS Elastic Beanstalk 是一項服務，可在熟悉的伺服器 (例如 Apache、Nginx、Passenger 和 IIS) 上部署和擴展以 Java、.NET、PHP、Node.js、Python、Ruby、Go 和 Docker 開發的 Web 應用程式。

Elastic Beanstalk 是 Amazon EC2、Auto Scaling 上的抽象概念，並透過提供複製、藍/綠部署、Elastic Beanstalk 命令列界面 (EB CLI) 等其他功能，以及與 AWS Toolkit for Visual Studio、Visual Studio Code、Eclipse 和 IntelliJ 整合來簡化部署，以提高開發人員生產力。

# EC2 Image Builder

EC2 Image Builder 是一項全受管 AWS 服務，可協助您自動建立、維護、驗證、共用和部署自訂、安全且up-to-date Linux 或 Windows 自訂 AMI。EC2 Image Builder 也可用於建立容器映像。您可以使用 AWS 管理主控台、 AWS CLI或 APIs 在 AWS 帳戶中建立自訂映像。

EC2 Image Builder 透過提供簡單的圖形界面、內建自動化和 AWS提供的安全設定，大幅減少將映像保持在up-to-date狀態和安全的工作量。使用 EC2 Image Builder，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。

# AWS Proton

AWS Proton 可讓平台團隊連線和協調開發團隊在基礎設施佈建、程式碼部署、監控和更新所需的所有不同工具。 AWS Proton 啟用自動化基礎設施，做為無伺服器和容器型應用程式的程式碼佈建和部署。

AWS Proton 可讓平台團隊定義其基礎設施和部署工具，同時為開發人員提供自助式體驗，以取得基礎設施和部署程式碼。透過 AWS Proton平台團隊佈建共用資源並定義應用程式堆疊，包括 CI/CD 管道和可觀測性工具。然後，您可以管理可供開發人員使用的基礎設施和部署功能。

# AWS Service Catalog

AWS Service Catalog 可讓組織建立和管理已核准的 IT 服務目錄 AWS。這些 IT 服務可以包含虛擬機器映像、伺服器、軟體、資料庫等所有項目，以完成多層應用程式架構。可讓您集中管理部署 AWS Service Catalog 的 IT 服務、應用程式、資源和中繼資料，以實現 IaC 範本的一致控管。

透過 AWS Service Catalog，您可以滿足您的合規要求，同時確保您的客戶可以快速部署他們所需的已核准 IT 服務。最終使用者可在機構所設的限制範圍內，迅速地只部署自己需要且經核准的 IT 服務。

## AWS Cloud9

AWS Cloud9 是以雲端為基礎的 IDE，可讓您使用瀏覽器撰寫、執行和偵錯程式碼。它包含程式碼編輯器、除錯器和 terminal. AWS Cloud9 comes，預先封裝了適用於熱門程式設計語言的基本工具，包括 JavaScript、Python、PHP 等，因此您不需要安裝檔案或設定開發機器來啟動新專案。由於 IDE AWS Cloud9 是以雲端為基礎，因此您可以使用網際網路連線的機器，從辦公室、家中或任何位置處理專案。

## AWS CloudShell

AWS CloudShell 是以瀏覽器為基礎的 Shell，可讓您更輕鬆地安全地管理、探索 資源，以及與您的 AWS 資源互動。 AWS CloudShell 已使用您的主控台登入資料進行預先驗證。常見的開發和操作工具已預先安裝，因此不需要在本機電腦上安裝或設定軟體。

## Amazon CodeGuru

Amazon CodeGuru 是一種開發人員工具，可提供智慧型建議，以改善程式碼品質並識別應用程式最昂貴的程式碼行。將 CodeGuru 整合到您現有的軟體開發工作流程中，以在應用程式開發期間自動化程式碼檢閱，並持續監控應用程式在生產中的效能，並提供有關如何改善程式碼品質、應用程式效能和降低整體成本的建議和視覺化線索。CodeGuru 有兩個元件：

- Amazon CodeGuru Reviewer — Amazon CodeGuru Reviewer 是一種自動程式碼檢閱服務，可識別 Java 和 Python 程式碼的關鍵瑕疵和偏離編碼最佳實務。它會掃描提取請求中的程式碼行，並根據從主要開放原始碼專案以及 Amazon 程式碼庫學到的標準提供智慧型建議。
- Amazon CodeGuru Profiler — Amazon CodeGuru Profiler 會分析應用程式執行期設定檔，並提供智慧型建議和視覺化，引導開發人員如何改善程式碼最相關部分的效能。

# 監控和可觀測性

溝通和協作是 DevOps 理念的基礎。為了促進這一點，意見回饋至關重要。此意見回饋由我們的監控和可觀測性服務套件提供。

AWS 提供下列 服務來監控和記錄：

主題

- Amazon CloudWatch 指標
- Amazon CloudWatch 警示
- Amazon CloudWatch Logs
- Amazon CloudWatch Logs Insights
- Amazon CloudWatch Events
- Amazon EventBridge
- AWS CloudTrail
- Amazon DevOps Guru
- AWS X-Ray
- Amazon Managed Service for Prometheus
- Amazon Managed Grafana

## Amazon CloudWatch 指標

Amazon CloudWatch 指標會自動從 Amazon EC2 執行個體、Amazon EBS 磁碟區和 Amazon RDS 資料庫 (DB) 執行個體等 AWS 服務收集資料。然後，這些指標可以組織為儀表板，也可以建立警示或事件來觸發事件或執行 Auto Scaling 動作。

## Amazon CloudWatch 警示

您可以根據 Amazon CloudWatch 指標所收集的指標，使用 Amazon CloudWatch 警示來設定警示。Amazon CloudWatch 警示接著可以將通知傳送至 Amazon SNS 主題，或啟動 Auto Scaling 動作。警示需要期間（評估指標的時間長度）、評估期間（最近資料點的數量）和要警示的資料點（評估期間內的資料點數量）。

# Amazon CloudWatch Logs

Amazon CloudWatch Logs 是一種日誌彙總和監控服務。 AWS CodeBuild CodeCommit、CodeDeploy 和 CodePipeline 提供與 CloudWatch 日誌的整合，以便集中監控所有日誌。此外，先前提到的服務各種其他 AWS 服務提供與 CloudWatch 的直接整合。

透過 CloudWatch Logs，您可以：

- 查詢您的日誌資料
- 從 Amazon EC2 執行個體監控日誌
- 監控 AWS CloudTrail 記錄的事件
- 定義日誌保留政策

# Amazon CloudWatch Logs Insights

Amazon CloudWatch Logs Insights 會掃描您的日誌，並可讓您執行互動式查詢和視覺化。它了解各種日誌格式，並從 JSON 日誌自動探索欄位。

# Amazon CloudWatch Events

Amazon CloudWatch Events 提供近乎即時的系統事件串流，描述 AWS 資源的變更。使用您可以快速設定的簡單規則，您可以比對事件並將它們路由到一或多個目標函數或串流。

CloudWatch Events 在操作變更時會查覺到。CloudWatch Events 會回應這些操作變更並視需要進行修正動作，透過傳送訊息來回應環境、啟用功能、執行變更和擷取狀態資訊。

您可以在 Amazon CloudWatch Events 中設定規則，以提醒您 AWS 服務中的變更，並使用 Amazon EventBridge 將這些事件與其他第三方系統整合。以下是與 CloudWatch Events 整合的 AWS DevOps 相關服務。

- Application Auto Scaling 事件
- CodeBuild 事件
- CodeCommit 事件
- CodeDeploy 事件
- CodePipeline 事件

## Amazon EventBridge

> **Note**
>
> Amazon CloudWatch Events 和 EventBridge 是相同的基礎服務和 API，但 EventBridge 提供更多功能。

Amazon EventBridge 是無伺服器事件匯流排，可整合 AWS 服務、軟體即服務 (SaaS) 和您的應用程式。除了建置事件驅動的應用程式之外，EventBridge 還可以用來通知來自 CodeBuild、CodeDeploy、CodePipeline 和 CodeCommit 等服務的事件。

## AWS CloudTrail

為了接受 DevOps 的協作、溝通和透明度原則，請務必了解誰正在修改您的基礎設施。在 中 AWS，此透明度由 提供AWS CloudTrail。所有 AWS 互動都會透過由 監控和記錄的 AWS API 呼叫來處理 AWS CloudTrail。所有產生的日誌檔案都會存放在您定義的 Amazon S3 儲存貯體中。日誌檔案使用 Amazon S3 伺服器端加密 (SSE) 進行加密。無論 API 呼叫是直接來自使用者，還是由 AWS 服務代表使用者，都會記錄所有 API 呼叫。許多群組可以從 CloudTrail 日誌中受益，包括支援的操作團隊、控管的安全團隊，以及計費的財務團隊。

## Amazon DevOps Guru

Amazon DevOps Guru是一種採用機器學習 (ML) 的服務，旨在輕鬆改善應用程式的操作效能和可用性。DevOps Guru 可協助偵測偏離正常操作模式的行為，因此您可以在營運問題影響您的客戶之前，早就找出這些問題。

DevOps Guru 使用以多年 Amazon.com 和卓越 AWS 營運為基礎的 ML 模型，以協助識別異常應用程式行為（例如，增加延遲、錯誤率、資源限制等），以及可能導致潛在中斷或服務中斷的表面關鍵問題。

當 DevOps Guru 識別關鍵問題時，它會從大量資料來源擷取相關和特定資訊，並自動傳送提醒，並提供相關異常的摘要，以及問題發生的時間和位置內容，以節省偵錯時間。

## AWS X-Ray

AWS X-Ray 協助開發人員分析和偵錯生產、分散式應用程式，例如使用微服務架構建置的應用程式。透過 X-Ray，您可以了解應用程式及其基礎服務的效能，以識別效能問題和錯誤的根本原因並進行故

障診斷。X-Ray 會在請求通過您的應用程式時提供請求的end-to-end檢視，並顯示應用程式基礎元件的映射。X-Ray 可讓您輕鬆地：

- 建立服務映射 – 透過追蹤對應用程式的請求，X-Ray 可以建立應用程式所使用的服務映射。這可讓您檢視應用程式中服務之間的連線，並可讓您建立相依性樹狀目錄、在可用區域或區域間 AWS 作業時偵測延遲或錯誤、在未如預期運作的服務上為零，以此類推。
- 識別錯誤 – X-Ray 可以透過分析對應用程式提出的每個請求的回應碼，自動反白應用程式程式碼中的錯誤。這可讓您輕鬆偵錯應用程式程式碼，而無需您重現錯誤。
- 建置您自己的分析和視覺化應用程式 – X-Ray 提供一組查詢 APIs，可用來建置您自己的分析和視覺化應用程式，這些應用程式使用 X-Ray 記錄的資料。

## Amazon Managed Service for Prometheus

Amazon Managed Service for Prometheus 是一種無伺服器監控服務，適用於與開放原始碼 Prometheus 相容的指標，可讓您更輕鬆地安全地監控和提醒容器環境。Amazon Managed Service for Prometheus 可減少在 Amazon Elastic Kubernetes Service、Amazon Elastic Container Service AWS Fargate以及自我管理 Kubernetes 叢集中開始使用監控應用程式所需的繁重工作。

## Amazon Managed Grafana

Amazon Managed Grafana 是一項全受管服務，具有豐富的互動式資料視覺化，可協助客戶分析、監控和警示多個資料來源之間的指標、日誌和追蹤。您可以建立互動式儀表板，並使用自動擴展、高可用性和企業安全服務與組織中的任何人共用。

# 通訊和協作

無論您是在組織中採用 DevOps 文化，還是經歷 DevOps 文化轉型，溝通和協作都是您方法的重要環節。在 Amazon，我們意識到需要改變團隊的思維，因此採用雙izza 團隊的概念。

「我們嘗試建立不大於 的隊伍，由兩個比薩提供」，Bezos 說。「我們稱之為雙比薩隊伍規則。」

團隊越小，協作越好。協同合作非常重要，因為軟體版本的移動速度比以往更快。而團隊交付軟體的能力，對於您的組織而言，可能是與競爭不同的因素。假設需要發佈新產品功能或需要修正錯誤的情況。您希望盡快發生這種情況，因此您可以縮短go-to-market時間。您不希望轉型成為緩慢移動的過程；您希望敏捷的方法，其中變革的波紋開始產生影響。

團隊之間的溝通也很重要，因為您要朝向共同的責任模型邁進，並開始脫離孤立的開發方法。這將擁有的概念帶到團隊，並轉移他們的觀點，將流程視為end-to-end企業。您的團隊不應將生產環境視為沒有可見性的黑色方塊。

文化轉型也很重要，因為您可能正在建立共同的 DevOps 團隊，或團隊中有 DevOps 聚焦的成員。這兩種方法都會將 中的共同責任引入 團隊。

# 安全

無論您是第一次經歷 DevOps 轉型還是實作 DevOps 原則，您都應該將安全性視為整合在 DevOps 程序中。這應該是跨建置、測試部署階段的交叉切削考量。

在 DevOps on 中探索安全性之前 AWS，本文會介紹 AWS 共同責任模型。

主題

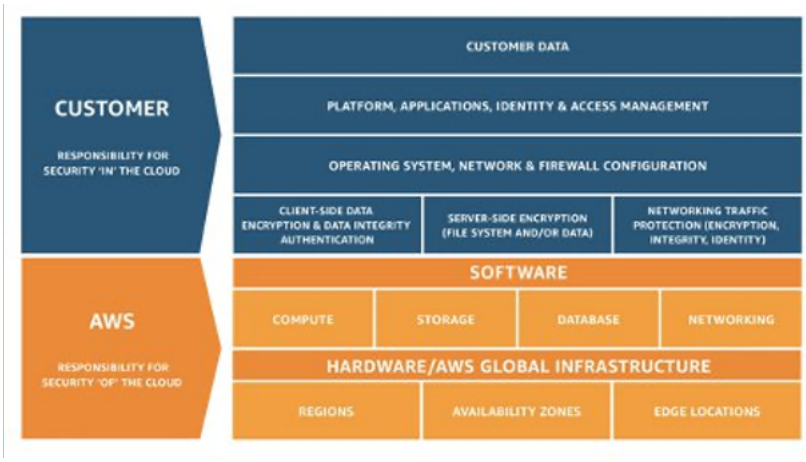
- AWS 共同責任模型
- 身分和存取權管理

## AWS 共同責任模型

安全性是 AWS 和 客戶之間的共同責任。共同責任模型的不同部分如下：

- AWS 負責「雲端安全性」 - AWS 負責保護執行 中提供之所有服務的基礎設施 AWS 雲端。此基礎設施由執行 AWS 雲端 服務的硬體、軟體、聯網和設施組成。
- 客戶責任「雲端中的安全」 – 客戶責任取決於客戶選取的 AWS 雲端 服務。這也確定了客戶在履行其安全性責任的過程中，必須執行的設定工作量。

此共用模型有助於減輕客戶的營運負擔，因為 會 AWS 操作、管理和控制從主機作業系統和虛擬化層到服務營運所在設施實體安全的元件。這在客戶想要了解其建置環境的安全性時至關重要。




AWS 共同責任模型

對於 DevOps，請根據最低權限許可模型指派許可。此模型指出「使用者 （或服務） 應該擁有完成其角色責任所需的確切存取權限，不會再多也不少。 ”。

許可保留在 IAM 中。您可以使用 IAM 來控制已驗證（已登入）和已授權（具有許可）使用 資源的人員。

## 身分和存取權管理

AWS Identity and Access Management (IAM) 定義用於管理 AWS 資源存取的控制項和政策。您可以使用 IAM 建立使用者和群組，並定義各種 DevOps 服務的許可。

除了使用者之外，各種 服務也可能需要存取 AWS 資源。例如，您的 CodeBuild 專案可能需要存取，才能將 Docker 映像存放在 Amazon Elastic Container Registry (Amazon ECR) 中，而且需要寫入 Amazon ECR 的許可。這些類型的許可是由稱為服務角色的特殊類型角色定義。

IAM 是 AWS 安全基礎設施的一個元件。透過 IAM，您可以集中管理群組、使用者、服務角色和安全性登入資料，例如密碼、存取金鑰和許可政策，以控制使用者可以存取的 AWS 服務和資源。IAM 政策可讓您定義一組許可。然後，此政策可以連接到角色、使用者或服務，https://docs.aws.amazon.com/IAM/latest/UserGuide/using-service-linked-roles.html以定義其許可。

您也可以使用 IAM 來建立在所需 DevOps 策略中廣泛使用的角色。在某些情況下，以程式設計方式使用 AssumeRole 而非直接取得許可是很合理的。當服務或使用者擔任角色時，他們會收到臨時登入資料，以存取他們通常無法存取的服務。

# 結論

為了讓雲端之旅順利、有效率且有效，技術公司應該接受 DevOps 原則和實務。這些原則內嵌在 中 AWS，並構成許多 AWS 服務的基石，尤其是部署和監控產品中的基石。

首先使用 服務或 將基礎設施定義為程式碼 AWS CloudFormation AWS CDK。接著，定義您的應用程式在 AWS CodeBuild諸如 AWS CodeDeploy AWS CodePipeline和 等服務的協助下，使用持續部署的方式 AWS CodeCommit。在應用程式層級，使用容器 AWS Elastic Beanstalk，例如 Amazon ECS 或 Amazon Elastic Kubernetes Service (Amazon EKS)。使用 OpsWorks 簡化常見架構的組態。使用這些服務也可輕鬆包含其他重要的服務，例如 Auto Scaling 和 Elastic Load Balancing。

最後，使用 DevOps 監控策略，例如 Amazon CloudWatch，以及 IAM 等堅實安全實務。

使用 AWS 作為您的合作夥伴，您的 DevOps 原則為您的業務和 IT 組織帶來敏捷性，並加速您的雲端之旅。

# 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 |
| --- | --- | --- |
| Updated | Updated | 2023 年 4 月 7 日 |
| 更新章節以包含新服務 | 更新章節以包含新服務 | 2020 年 10 月 16 日 |
| 初次出版 | 白皮書首次發佈 | 2014 年 12 月 1 日 |

# 貢獻者

本文件的貢獻者包括：

- Abhra Sinha，解決方案架構師
- Anil Nadiminti，解決方案架構師
- Muhammad Mansoor，解決方案架構師
- Ajit Zadgaonkar，全球技術領導者，現代化
- Juan Lamadrid，解決方案架構師
- Darren Ball，解決方案架構師
- Rajeswari Malladi，解決方案架構師
- Pallavi Nargund，解決方案架構師
- Bert Zahniser，解決方案架構師
- Abdullahi Olaoye，雲端解決方案架構師
- Mohamed Kiswani，軟體開發經理
- Tara McCann，解決方案架構師經理

# 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，這些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「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。