為您的 Amazon EKS 叢集選擇正確的 GitOps 工具

AWS 方案指引

AWS 方案指引: 為您的 Amazon EKS 叢集選擇正確的 GitOps 工具

Table of Contents

為您的 Amazon EKS 叢集選擇正確的 GitOps 工具

Pradip Kumar Pandey 和 Pratap Kumar Nanda，Amazon Web Services (AWS)

2025 年 4 月 (文件歷史記錄)

在雲端原生技術的快速發展環境中，GitOps 已成為管理和部署應用程式和基礎設施的強大方法。如果您使用 Amazon Elastic Kubernetes Service (Amazon EKS)，實作 GitOps 原則可以大幅增強部署程序、改善可靠性並簡化操作。有各種 GitOps 工具可供使用，而為您的 EKS 叢集選擇正確的工具，是可能影響團隊效率和 DevOps 實務整體成功的關鍵決策。

為您的 Amazon EKS 環境選擇適當的 GitOps 工具需要仔細考慮各種因素，包括您的特定需求、團隊專業知識、可擴展性需求，以及與現有 的整合功能 AWS 服務。每個工具都有自己的一組功能、優勢和潛在限制，因此請務必將選擇與組織的目標和營運內容保持一致。

本指南探討為 Amazon EKS 選取 GitOps 工具時的重要考量，比較常用的選項，並提供洞見以協助您做出明智的決策。它涵蓋九種熱門的 GitOps 工具：

- Argo CD
- 通量
- 編織 GitOps
- Jenkins X
- GitLab CI/CD
- Spinnaker
- Rancher 機群
- Codefresh
- Pulumi

目標業務成果

當您選擇在開發和操作程序中實作 GitOps 原則的工具時，以下清單會討論潛在的目標和結果。

與 Amazon EKS 無縫整合

您的 GitOps 工具應與 Amazon EKS 順暢整合，並提供與 Amazon EKS 特定功能和最佳化的相容性。

- 原生 Amazon EKS 支援：尋找可為 Amazon EKS 提供內建支援的工具，包括簡單的叢集連線和管理。
- AWS 服務 整合：請確定工具可以與其他 互動， AWS 服務 例如 AWS Identity and Access Management (IAM)、Amazon Elastic Container Registry (Amazon ECR) 和 Amazon CloudWatch。
- Amazon EKS 附加元件相容性：確認工具支援 Amazon EKS 附加元件，並且可以有效管理這些附加元件。

可擴展性和效能

您的 GitOps 工具應該能夠處理 Amazon EKS 操作的規模，從小型叢集到大型多叢集環境。

- 資源效率：評估工具的資源耗用量及其對叢集效能的影響。
- 大規模操作：評估工具同時管理許多應用程式和叢集的能力。
- 負載下的效能：考慮工具在高頻率更新和大規模部署期間的效能。

安全與合規

安全功能和合規功能至關重要，尤其是在受管制產業或處理敏感資料時。

- 存取控制：尋找與 IAM 整合的強大角色型存取控制 (RBAC) 功能。
- 秘密管理：評估工具如何處理敏感資訊，並與 AWS Secrets Manager或其他解決方案整合。
- 稽核線索：確保工具提供完整的記錄和稽核功能，以實現合規和故障診斷。
- 安全掃描：考慮為部署中的漏洞提供內建安全掃描的工具。

易於使用和學習曲線

該工具應該易於使用，並與團隊的技能保持一致，以確保快速採用和有效使用。

- 使用者介面：評估命令列界面 (CLI) 和圖形化使用者介面 (GUI) 功能的直覺性。
- 文件品質：尋找完整up-to-date文件和教學課程。
- 學習資源：考慮培訓資料、課程和社群資源的可用性。

社群和網路支援

強大的社群和網路可以提供寶貴的資源、外掛程式和長期永續性。

- 主動開發：檢查更新的頻率和維護器的回應能力。
- 社群大小：考慮使用者社群的大小和活動，以進行支援和知識分享。
- 第三方整合：評估外掛程式的可用性，以及與堆疊中其他工具的整合。

多叢集管理功能

如果您有多個 EKS 叢集，有效管理它們的能力至關重要。

- 集中式管理：尋找允許從單一控制平面管理多個叢集的功能。
- 叢集聯合：考慮支援多叢集應用程式 Kubernetes 聯合的工具。
- 環境同位：評估工具在不同環境中維持一致性的程度，例如開發、預備和生產。

可觀測性和監控

此工具應提供部署狀態和叢集運作狀態的清晰洞見。

- 部署可見性：尋找可清楚檢視部署狀態和歷史記錄的功能。
- 與監控工具的整合：考慮工具與 Prometheus 和 Grafana 等熱門監控解決方案整合的程度。
- 提醒功能：評估工具設定和管理部署問題或偏離提醒的能力。

彈性和自訂

能夠根據特定工作流程和需求調整工具，對於長期滿意度至關重要。

- 可擴展性：尋找可讓您擴展工具功能的外掛程式架構或 APIs。
- 自訂資源支援：確認工具可以有效地處理自訂 Kubernetes 資源。
- 工作流程自訂：評估您可以輕鬆根據團隊的需求量身打造 GitOps 工作流程。

持續交付和漸進式部署支援

進階部署策略通常對於將風險降至最低並確保順利更新至關重要。

- Canary 部署：尋找 Canary 版本的內建支援。
- 藍/綠部署：評估工具的藍/綠部署策略功能。
- 回復機制：確保強大且easy-to-use回復功能，以便從失敗的部署中快速復原。

成本效益和資源使用率

考慮採用和維護工具的整體成本，包括直接和間接成本。

- 授權成本：比較開放原始碼選項與商業解決方案，並考慮支援和企業功能。
- 營運開銷：評估管理和維護方面的額外營運成本。
- 資源耗用：根據所需的運算和儲存資源，評估工具的效率。

透過仔細考慮這些結果及其層面，您可以針對最適合 EKS 叢集的 GitOps 工具做出明智決策，並確保該工具符合您組織的需求、功能和長期策略。

EKS 叢集的 GitOps 工具

目前有數種適用於 Kubernetes 的 GitOps 工具可供市場使用。以下是一些最常用選項的清單：

- Argo CD
- 磁通
- 編織 GitOps
- Jenkins X
- GitLab CI/CD
- Spinnaker
- Rancher 機群
- Codefresh
- Pulumi

請依照連結查看這些工具如何實作 GitOps 實務的詳細資訊。每個工具都有優勢和使用案例。選擇取決於您的特定需求、現有基礎設施、團隊專業知識和所需功能等因素。請務必根據組織的需求和 Kubernetes 環境的複雜性來評估這些工具。

Argo CD

Argo CD 是 Kubernetes 廣泛使用的 GitOps 持續交付 (CD) 工具，符合數個關鍵 GitOps 原則。

GitOps 支援

區域圖	工具功能
宣告式組態	Argo CD 使用儲存在 Git 儲存庫中的宣告式組態。應用程式和基礎設施的所需狀態是在 YAML 檔案中定義。這些組態描述應部署的內容，而不是如何部署。
版本控制系統做為單一事實來源	Git 儲存庫做為整個系統的單一事實來源。應用程式和基礎設施的所有變更都是透過 Git 進行。這可確保完整的稽核線索，以及復原至任何先前狀態的能力。

區域圖	工具功能
自動化同步	Argo CD 會持續監控 Git 儲存庫的變更。偵測到變更時，會自動同步叢集的實際狀態與 Git 中定義的所需狀態。這可確保叢集一律反映儲存庫中所述的狀態。
Kubernetes 原生	Argo CD 專為 Kubernetes 環境而設計。它利用 Kubernetes 中的宣告性質和自訂資源來管理應用程式。
自我修復和偏離偵測	Argo CD 會定期比較叢集的即時狀態與 Git 中所需的狀態。如果偵測到任何偏離（實際和所需狀態之間的差異），它可以自動更正這些差異。
多叢集和多租用支援	Argo CD 可以從單一執行個體管理多個 Kubernetes 叢集。它支援多租戶，因此不同的團隊可以獨立管理其應用程式。
應用程式定義	Argo CD 中的應用程式是透過使用應用程式 CRD（自訂資源定義）來定義。這可讓 Kubernetes 原生方式定義應部署的內容和方式。
部署和發行的分隔	Argo CD 會將程式碼的部署與發行版本分開給使用者。這是透過各種部署策略實現的，例如藍/綠或金絲雀部署。
可觀測性和可稽核性	Argo CD 提供 Web UI 和 CLI 來觀察應用程式和叢集的狀態。會記錄所有動作，以提供變更和部署的清晰稽核線索。
安全性和 RBAC	Argo CD 與 Kubernetes 角色型存取控制 (RBAC) 整合。它支援單一登入整合以進行身分驗證和授權。

區域圖	工具功能
可插入架構	Argo CD 支援各種來源控制系統、Helm Chart、Kustomize 和其他 Kubernetes 資訊清單格式。這種靈活性允許它適應各種環境和工作流程。
持續交付 (CD)	雖然 Argo CD 專注於持續交付，但它可以與持續整合 (CI) 工具整合，以建立完整的 CI/CD 管道。

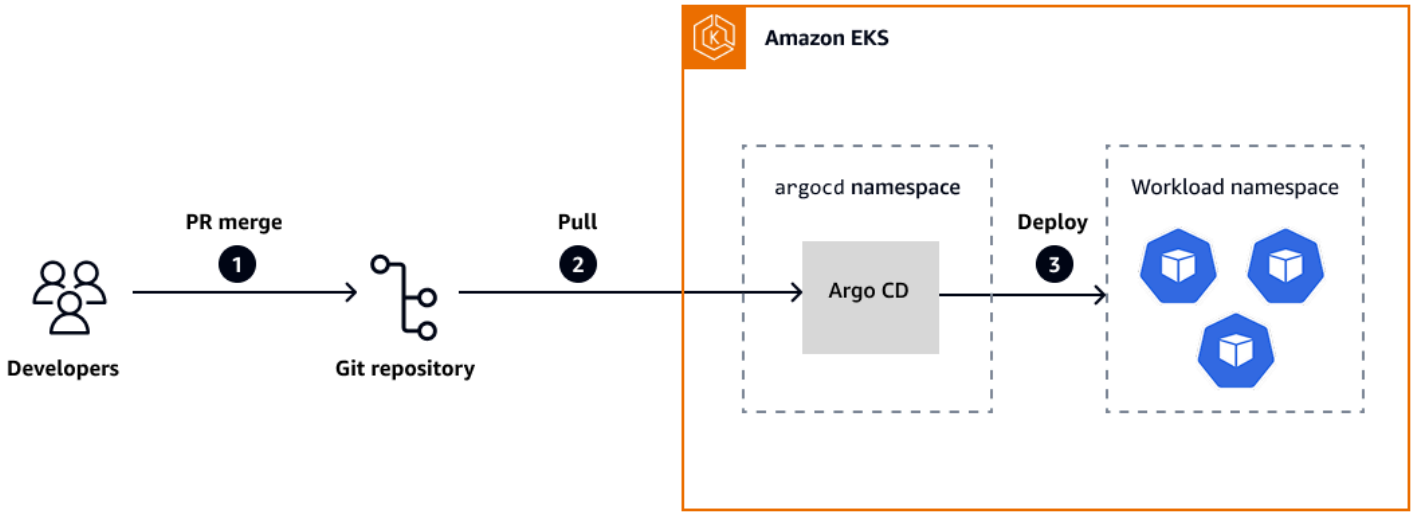
透過遵循這些 GitOps 原則，Argo CD 提供強大、可擴展且安全的方式來管理 Kubernetes 部署。它可確保系統的操作狀態始終與 Git 儲存庫中定義的所需狀態同步，並提高複雜 Kubernetes 環境中的一致性、可靠性和易於管理。

如需 Argo CD 可以解決的案例和需求，請參閱本指南稍後的 Argo CD 使用案例。如需 Argo CD 和 Flux 之間的比較，請參閱本指南稍後的功能比較。

如需詳細資訊，請參閱 Argo CD 文件。

架構

下圖說明在 EKS 叢集中使用 Argo CD 的 GitOps 驅動 CD 工作流程。如需詳細資訊，請參閱 Argo CD 文件。



其中：

- 步驟 1：提取請求 (PR) 合併。開發人員會將變更遞交至存放在 Git 儲存庫中的 Kubernetes 資訊清單或 Helm Chart。當 PR 已檢閱並合併到主分支時，應用程式所需的狀態會在來源控制中更新。
- 步驟 2：儲存庫同步。Argo CD 會在 EKS 叢集中的專用命名空間 (`argocd`) 內執行，並持續監控設定的 Git 儲存庫。當偵測到變更時，它會提取最新的更新，以協調宣告的狀態。
- 步驟 3：部署至目標命名空間。Argo CD 會將 Git 中所需的狀態與叢集中的即時狀態進行比較。然後，它會將必要的變更套用至目標工作負載命名空間，以便相應地部署或更新應用程式。這包括管理 Kubernetes 資源，例如部署、服務、ConfigMaps 和秘密，以維持叢集與 Git 事實來源的一致性。

通量

Flux 是 Kubernetes 的另一個工具，以獨特的方式實作 GitOps 原則。

GitOps 支援

區域圖	工具功能
Git 作為單一事實來源	Flux 使用 Git 儲存庫作為定義系統所需狀態的最終來源。應用程式和基礎設施的所有組態都存放在 Git 中。
宣告式組態	Flux 使用叢集所需狀態的宣告式描述。這些描述通常是 Kubernetes 資訊清單、Helm Chart 或 Kustomize 浮水印。
自動化非同步	Flux 會持續監控 Git 儲存庫的變更。當它偵測到變更時，會自動將其套用至叢集。
Kubernetes 原生	Flux 建置為一組 Kubernetes 控制器和自訂資源。它使用 Kubernetes 中的延伸機制來提供 GitOps 功能。
提取型部署模型	與傳統的推送式 CI/CD 系統不同，Flux 使用提取式模型。叢集會從 Git 提取所需的狀態，而不是使用外部系統推送變更。

區域圖	工具功能
持續對帳	Flux 會持續比較叢集的實際狀態與 Git 中所需的狀態。它會自動更正在這些狀態之間偵測到的任何偏離。
多租戶	Flux 透過 Kustomizations 和 HelmReleases 的概念支援多租戶。不同的團隊可以獨立管理自己的組態部分。
漸進式交付	Flux 透過其 Flagger 元件支援進階部署策略，例如 Canary Releases 和 A/B 測試。
Helm 整合	Flux 包含 Helm 的原生支援，因此您可以透過 GitOps 輕鬆管理 Helm 版本。
映像更新自動化	當容器登錄檔中有新版本可用時，Flux 可以自動更新 Git 中的容器映像。
Kustomize 支援	您可以使用 Flux for Kustomize 提供的原生支援來自訂和修補 Kubernetes 資訊清單。
安全性和 RBAC	Flux 與 Kubernetes RBAC 整合以進行存取控制。它支援透過各種後端進行秘密管理。
可觀測性	Flux 提供有關對帳和操作的狀態資訊和指標。它與監控工具整合，以增強可觀測性。
事件驅動型架構	Flux 使用事件驅動方法來實作對帳和更新。
可擴展性	該工具旨在可擴展，因此您可以新增自訂控制器和資源。
跨叢集同步	Flux 支援從單一組儲存庫管理多個叢集。
相依性管理	它允許在系統的不同部分之間定義相依性，並確保正確的操作順序。
Webhook 接收器	您可以設定 Flux 從 Git 提供者或其他系統接收 Webhook，以立即開始對帳。

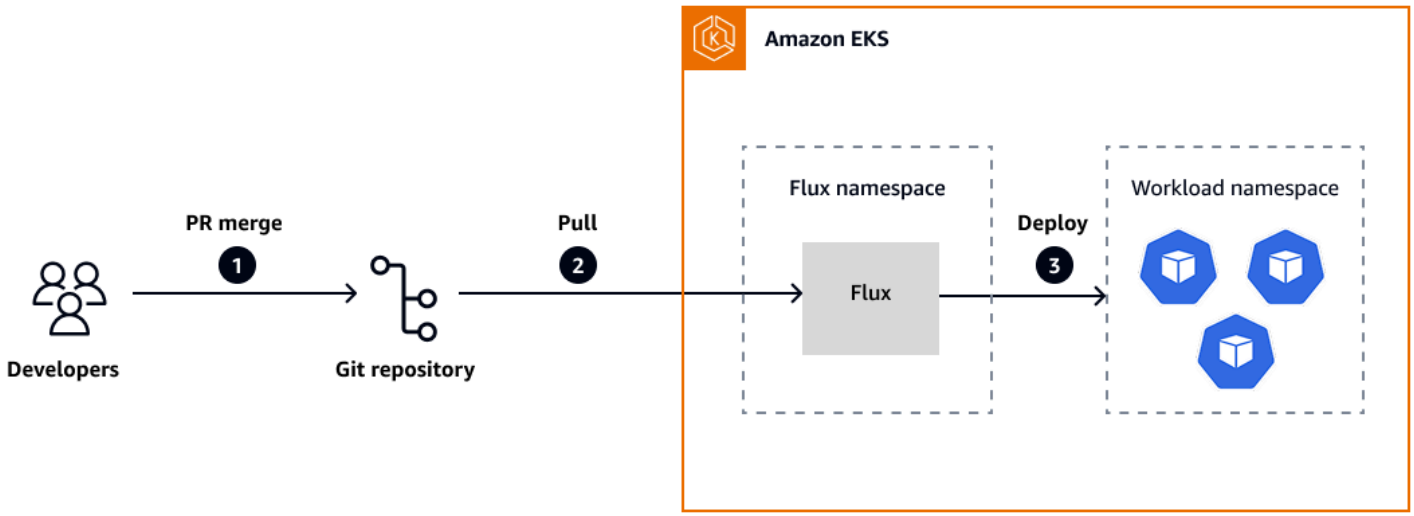
透過實作這些 GitOps 原則，Flux 提供強大且靈活的系統來管理 Kubernetes 叢集和應用程式。它可確保您的基礎設施和應用程式始終與您的 Git 儲存庫同步，並在複雜的 Kubernetes 環境中提供一致性、可靠性和易於管理。工具的 Kubernetes 原生方法和專注於自動化，使其特別適合雲端原生環境。

如需 Flux 可以解決的案例和需求，請參閱本指南稍後的 Flux 使用案例。如需 Argo CD 和 Flux 之間的比較，請參閱本指南稍後的功能比較。

如需詳細資訊，請參閱 Flux 文件。

架構

下圖說明在 EKS 叢集中使用 Flux 的 GitOps 驅動 CD 工作流程。如需詳細資訊，請參閱 Flux 文件。



其中：

- 步驟 1：提取請求 (PR) 合併。開發人員會將變更遞交至存放在 Git 儲存庫中的 Kubernetes 資訊清單或 Helm Chart。檢閱 PR 並合併到主分支時，會在來源控制中更新應用程式的所需狀態。
- 步驟 2：儲存庫同步。Flux 會在 EKS 叢集的專用命名空間內執行，並持續監控設定的 Git 儲存庫。當偵測到變更時，它會提取最新的更新，以協調宣告的狀態。
- 步驟 3：部署至目標命名空間。Flux 會將 Git 中所需的狀態與叢集中的即時狀態進行比較。然後，它會將必要的變更套用至目標工作負載命名空間，以便相應地部署或更新應用程式。

編織 GitOps

Weave GitOps 是由 Weaveworks 開發，這是引進 GitOps 一詞的公司。此工具提供以核心 GitOps 原則為基礎的全方位 GitOps 解決方案。

GitOps 支援

區域圖	工具功能
Git 作為單一事實來源	Weave GitOps 使用 Git 儲存庫做為授權來源，以定義系統所需的狀態。所有組態，包括應用程式資訊清單、基礎設施定義和政策，都存放在 Git 中。
宣告式組態	系統依賴整個系統狀態的宣告性描述。這些描述通常是 Kubernetes 資訊清單、Helm Chart 或其他宣告格式。
自動化同步	Weave GitOps 會持續監控 Git 儲存庫的變更。當它偵測到變更時，會自動將其套用至目標環境。
Kubernetes 原生架構	Weave GitOps 建置為一組 Kubernetes 控制器和自訂資源。它使用 Kubernetes 中的延伸機制來提供 GitOps 功能。
持續對帳	此工具會持續比較叢集的實際狀態與 Git 中定義的所需狀態。它會自動更正在這些狀態之間偵測到的任何偏離。
多叢集管理	Weave GitOps 支援從單一控制平面管理多個 Kubernetes 叢集。它可在不同的環境中實現一致的應用程式部署。
政策即程式碼	Weave GitOps 將政策的概念整合為強制執行安全與合規規則的程式碼。政策是隨應用程式程式碼和基礎設施定義一起控制版本。
漸進式交付	此工具支援進階部署策略，例如 Canary Releases 和藍/綠部署。它與 Flagger 整合，以實現自動化、漸進式交付。

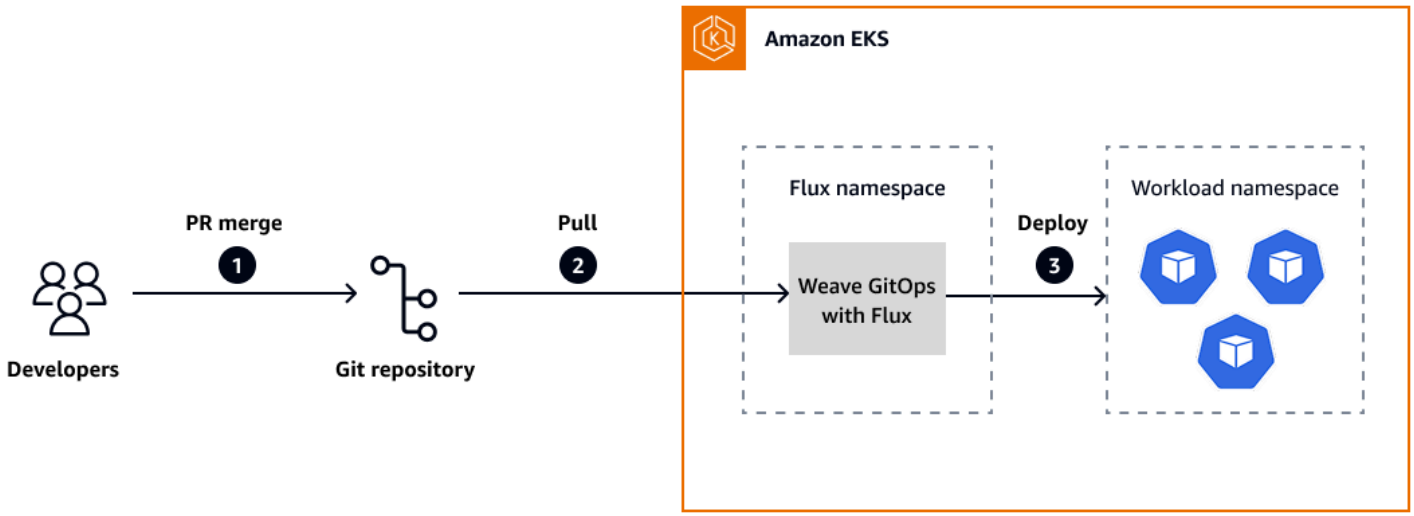
區域圖	工具功能
可觀測性和儀表板	Weave GitOps 提供內建儀表板，用於監控應用程式和叢集的狀態。它提供對調校程序和叢集運作狀態的洞察。
依設計保護	此工具實作安全最佳實務，包括 RBAC 整合和秘密管理。它支援各種身分驗證方法，並與企業身分提供者整合。
可擴展性和整合	該工具旨在使用各種雲端原生工具。它支援常用的工具，例如 Flux、Helm 和 Kustomize。
自助式開發人員平台	Weave GitOps 可讓開發人員建立自助式平台。它為應用程式部署提供範本和護欄。
GitOps 自動化	此工具可自動化 GitOps 工作流程的許多層面，包括產生更新請求。
持續交付管道	它與 CI/CD 系統整合，以建立end-to-end交付管道。
稽核與合規	編織 DevOps 提供所有變更和動作的完整稽核線索。它可協助您透過版本控制和自動化程序來滿足合規要求。
可擴展性	該工具旨在從小型專案擴展到大型企業級部署。
團隊協作	Weave GitOps 透過 Git 型工作流程促進開發和營運團隊之間的協作。
GitOps 即服務	此工具提供 GitOps 做為受管服務，可簡化採用和管理。
混合雲端和多雲端支援	Weave GitOps 可在不同的雲端提供者和內部部署環境中實現一致的管理。
持續安全性	該工具在整個部署過程中整合了安全掃描和政策強制執行。

Weave GitOps 實作這些原則，以提供超越基本部署自動化的全方位 GitOps 解決方案。它旨在為專注於安全性、可擴展性和易用性的雲端原生應用程式建立完整的操作模型。透過遵循這些 GitOps 原則，Weave GitOps 可協助組織跨多個叢集和雲端供應商，實現一致、可稽核且高效率的 Kubernetes 環境管理。

如需詳細資訊，請參閱 Weave GitOps 文件。

架構

下圖說明在 GitOps EKS 叢集中使用 Weave GitOps 的 GitOps 驅動 CD 工作流程。如需詳細資訊，請參閱 Weave GitOps 儲存庫。



其中：

- 步驟 1：提取請求 (PR) 合併。開發人員會將變更遞交至存放在 Git 儲存庫中的 Kubernetes 資訊清單或 Helm Chart。檢閱 PR 並合併到主分支時，會在來源控制中更新應用程式的所需狀態。
- 步驟 2：儲存庫同步。Weave GitOps 會在 EKS 叢集中的 Flux 命名空間內執行，並持續監控設定的 Git 儲存庫。當偵測到變更時，它會提取最新的更新，以協調宣告的狀態。
- 步驟 3：部署至目標命名空間。Weave GitOps 會將 Git 中所需的狀態與叢集中的即時狀態進行比較。然後，它會將必要的變更套用至目標工作負載命名空間，以便相應地部署或更新應用程式。

Jenkins X

Jenkins X 是雲端原生的開放原始碼 CI/CD 平台，可針對 Kubernetes 環境實作 GitOps 原則。雖然 Jenkins X 並非像 Argo CD 或 Flux 之類的 GitOps 工具，但它將 GitOps 實務納入其工作流程。

GitOps 支援

區域圖	工具功能
Git 中心工作流程	Jenkins X 使用 Git 儲存庫作為應用程式程式碼和組態的主要事實來源。應用程式和基礎設施的所有變更都是透過 Git 進行。
環境即程式碼 (EaC)	環境（例如預備和生產）在 Git 儲存庫中定義為程式碼。這允許版本控制和檢閱環境組態。
自動化 CI/CD 管道	Jenkins X 會自動設定專案的 CI/CD 管道。這些管道定義為程式碼（管道為程式碼），並存放在 Git 中。
Kubernetes 原生	Jenkins X 專為 Kubernetes 環境而建置。它使用 Kubernetes 資源和自訂資源定義 (CRDs)。
預覽環境	Jenkins X 會自動為提取請求建立暫時環境。它可在合併之前輕鬆檢閱和測試變更。
環境之間的提升	Jenkins X 使用 GitOps 方法來提升環境之間的應用程式（例如，從預備到生產）。提升是透過使用提取請求來處理，以確保適當的審核和核准程序。
Helm Chart 管理	Jenkins X 使用 Helm Chart 來封裝和部署應用程式。圖表在 Git 儲存庫中受版本控制。
自動化版本控制	Jenkins X 會自動管理應用程式和版本的版本控制。它使用語意版本控制並產生版本備註。
ChatOps 整合	Jenkins X 支援 ChatOps 的常見操作。這符合自動化和協作的 GitOps 原則。
可擴展性	此工具提供用於擴展功能的外掛程式系統。它允許與各種雲端原生工具整合。

區域圖	工具功能
基礎設施即程式碼 (IaC)	Jenkins X 支援 Terraform CloudFormation AWS Cloud Development Kit (AWS CDK)、 和其他 IaC 工具來定義和管理基礎設施。基礎設施定義與應用程式程式碼一起受版本控制。
自動化轉返	如果在部署後偵測到問題，Jenkins X 支援自動轉返。
秘密管理	該工具與外部秘密管理解決方案整合，以安全地處理敏感資訊。
可觀測性	Jenkins X 提供與監控和記錄工具的整合，以實現可觀測性。
多雲端支援	Jenkins X 旨在跨不同的雲端提供者和內部部署環境運作。
團隊協作	此工具鼓勵透過 Git 型工作流程進行協作，並提取請求。
持續意見回饋	此工具透過自動化測試和預覽環境，提供變更的快速意見回饋。
DevOps 最佳實務	Jenkins X 預設會實作 DevOps 最佳實務，包括 GitOps 原則。
宣告式組態	此工具使用宣告式組態來定義應用程式和環境。
自動升級	Jenkins X 提供工具來自動化 Jenkins X 平台本身的升級。

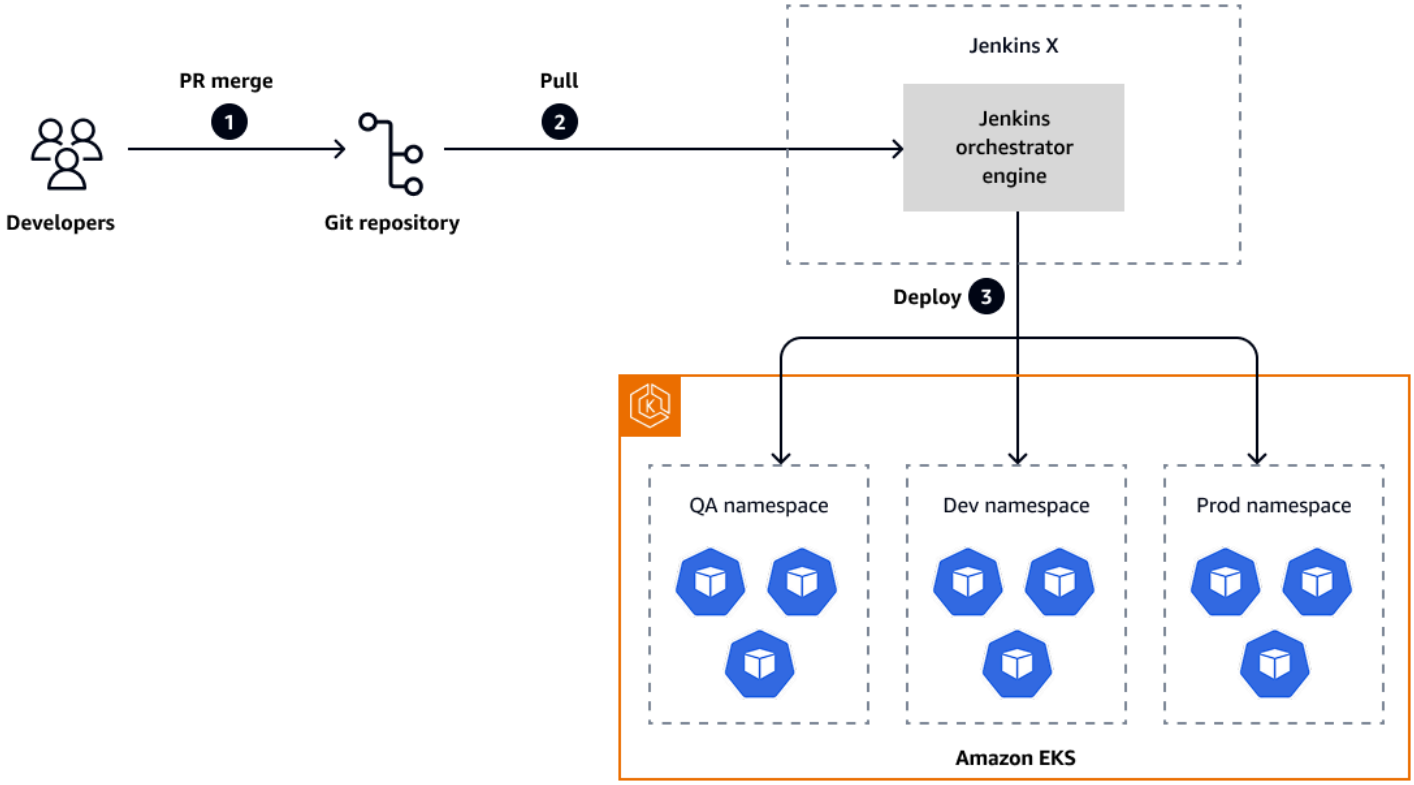
Jenkins X 實作這些 GitOps 原則，為 Kubernetes 建立全面的 CI/CD 解決方案。它旨在自動化和簡化從程式碼遞交到生產部署的整個軟體交付程序，同時遵守 GitOps 實務。透過這樣做，可協助團隊在雲端原生環境中實現更快、更可靠且更一致的部署。

Jenkins X 和 Argo CD 或 Flux 等工具之間的主要區別在於，Jenkins X 提供更全面的 CI/CD 解決方案，包括建置自動化和管道管理，同時仍然整合 GitOps 原則以進行部署和環境管理。這使得它特別適合需要涵蓋單一 GitOps 架構內 CI 和 CD 層面all-in-one解決方案的團隊。

如需詳細資訊，請參閱 Jenkins X 文件。

架構

下圖說明使用 Jenkins X 的 GitOps 驅動 CD 工作流程。如需詳細資訊，請參閱 Jenkins X 文件。



其中：

- 步驟 1：提取請求 (PR) 合併。開發人員會建立提取請求，其中包含對存放在 Git 儲存庫中的 Kubernetes 資訊清單、Helm Chart 或應用程式程式碼所做的變更。檢閱和核准之後，PR 會合併到主分支，並在來源控制中更新所需的狀態。
- 步驟 2：儲存庫同步。Jenkins X 會在偵測到變更時自動觸發 CI/CD 管道。管道會使用 GitOps 原則，透過不同的環境（例如預備和生產）建置、測試和提升應用程式。
- 步驟 3：部署至目標命名空間。Jenkins X 會使用新的應用程式版本更新環境儲存庫（預備和生產）。叢集會自動協調變更，方法是從 Git 提取最新的資訊清單，並將應用程式部署到適當的命名空間。

GitLab CI/CD

GitLab CI/CD 是 GitLab 平台的整合部分，可提供持續整合、交付和部署功能。雖然 GitLab CI/CD 不只是 GitOps 工具，但您可以設定它來實作 GitOps 原則，尤其是當您將其用於 Kubernetes 部署時。

GitOps 支援

區域圖	工具功能
Git 作為單一事實來源	GitLab CI/CD 使用 Git 儲存庫來存放應用程式程式碼和基礎設施組態。系統的所有變更都是透過 Git 進行，以確保完整的歷史記錄和稽核線索。
宣告式組態	GitLab CI/CD 管道在 .gitlab-ci.yml 檔案中定義，這是存放在 Git 儲存庫中的宣告式組態。Kubernetes 資訊清單、Helm Chart 或其他基礎設施即程式碼 (IaC) 檔案可以存放在相同的儲存庫中，以定義基礎設施的所需狀態。
自動化管道	將變更推送至儲存庫時，GitLab CI/CD 會自動觸發管道。這些管道可以包含建置、測試和部署應用程式的階段。
Kubernetes 整合	GitLab CI/CD 提供原生 Kubernetes 整合，並支援將 GitOps 樣式部署到 Kubernetes 叢集。它可以根據 Git 中的組態自動建立和管理 Kubernetes 資源。
環境管理	GitLab CI/CD 支援將多個環境 （例如預備和生產） 定義為程式碼。部署到這些環境可以是自動化的，或者可能需要符合 GitOps 實務的手動核准。
檢閱應用程式	GitLab 可以自動為合併請求建立臨時環境，類似於其他 GitOps 工具中的預覽環境。這支援在合併之前輕鬆檢閱和測試變更。

區域圖	工具功能
持續部署	您可以將 GitLab CI/CD 設定為在變更合併至特定分支時，自動將變更部署至 Kubernetes 叢集。
IaC	GitLab CI/CD 支援與 Terraform 等工具整合 CloudFormation ，並以程式碼形式管理基礎設施。基礎設施定義可與應用程式程式碼一起進行版本控制。
可觀測性和監控	GitLab CI/CD 提供內建的監控和可觀測性功能，包括與 Prometheus 和 Grafana 的整合。
安全性掃描	GitLab CI/CD 包含內建的安全掃描工具，可整合到 CI/CD 管道中，以在 GitOps 工作流程中強制執行安全性。
容器登錄檔	GitLab CI/CD 包含內建容器登錄檔，可在 GitOps 工作流程中無縫整合容器映像管理。
Auto DevOps	GitLab CI/CD 中的 Auto DevOps 功能可以自動設定遵循 Kubernetes 部署 GitOps 原則的 CI/CD 管道。
核准工作流程	GitLab CI/CD 支援部署的核准程序，可在環境之間提供受控制的提升。
秘密管理	GitLab CI/CD 提供在 CI/CD 管道內安全地管理和使用秘密的功能。
版本控制和版本	GitLab CI/CD 支援自動版本控制和發行管理，作為 CI/CD 程序的一部分。
轉返	如果在部署後偵測到問題，GitLab CI/CD 可讓您輕鬆轉返至先前的版本。
稽核日誌	GitLab CI/CD 為所有動作提供全面的稽核日誌，以支援 GitOps 的可追蹤性方面。

區域圖	工具功能
多專案管道	GitLab CI/CD 支援跨多個專案或儲存庫的複雜 GitOps 工作流程。
ChatOps	GitLab CI/CD 支援 ChatOps 整合，透過聊天介面提供協作和操作。
Kubernetes 叢集管理	GitLab CI/CD 提供直接從 GitLab 介面管理 Kubernetes 叢集的功能。

雖然 GitLab CI/CD 並非專門為 GitOps 設計，但它可以有效地用於實作 GitOps 實務，尤其是已使用 GitLab 作為其主要開發平台的團隊。其整合方法結合了來源控制、CI/CD 和 Kubernetes 管理，使其成為實作 GitOps 工作流程的強大工具。

GitLab CI/CD 和專用 GitOps 工具如 Argo CD 或 Flux 之間的主要區別在於，GitLab 提供更全面的平台，其中包括來源控制管理、問題追蹤和其他開發工具及其 CI/CD 功能。這使得它特別適合需要在更廣泛的開發系統中實作 GitOps 實務all-in-one解決方案的團隊。

如需 GitLab CI/CD 及其架構的詳細資訊，請參閱 GitLab CI/CD 文件。

Spinnaker

雖然 Spinnaker 並非專門設計為 GitOps 工具，但您可以將其設定為實作 GitOps 原則，尤其是在將其用於雲端原生和 Kubernetes 部署時。

GitOps 支援

區域圖	工具功能
宣告式組態	Spinnaker 使用宣告管道定義，通常會儲存為 JSON 或 YAML 檔案。這些管道定義可以在 Git 儲存庫中進行版本控制，以符合 GitOps 實務。
IaC	Spinnaker 支援將基礎設施和部署組態定義為程式碼。這些定義可以存放在 Git 儲存庫中，並可做為單一事實來源。

區域圖	工具功能
多雲端部署	Spinnaker 旨在跨多個雲端提供者和 Kubernetes 叢集運作。它可在各種環境中實現一致的 GitOps 實務。
管道做為程式碼	Spinnaker 管道可以定義為程式碼，並存放在 Git 儲存庫中。這允許版本控制和檢閱部署程序。
自動化部署	您可以設定 Spinnaker 根據 Git 儲存庫中的變更自動啟動部署。此工具支援 GitOps 核心的持續部署實務。
不可變基礎設施	Spinnaker 提升使用不可變的基礎設施，這是 GitOps 中的關鍵概念。它鼓勵部署新的執行個體，而不是修改現有的執行個體。
轉返和版本控制	Spinnaker 提供強大的復原功能，並快速還原至先前的已知良好狀態。它支援部署的版本控制，以符合 GitOps 可追蹤性原則。
核准工作流程	Spinnaker 在管道中包含手動判斷階段，以支援環境之間的受控促銷。這支援部署和版本之間的 GitOps 分離實務。
Canary 和藍/綠部署	Spinnaker 支援進階部署策略，以符合 GitOps 安全且受控版本的實務。
與版本控制系統整合	Spinnaker 可以與各種 Git 供應商整合，以根據儲存庫事件啟動管道。
Kubernetes 整合	Spinnaker 為 Kubernetes 提供原生支援，並支援 Kubernetes 資源的 GitOps 樣式管理。
成品管理	Spinnaker 支援成品管理和版本控制，這對維護 GitOps 工作流程至關重要。

區域圖	工具功能
可觀測性和監控	Spinnaker 提供與監控工具的整合，以支援 GitOps 的可觀測性方面。
稽核線索	Spinnaker 提供詳細的部署日誌和歷史記錄，支援 GitOps 的可稽核性原則。
角色型存取控制 (RBAC)	此工具實作 RBAC 以精細控制誰可以執行哪些動作，以符合 GitOps 安全實務。
範本化和參數化	Spinnaker 支援管道定義中的範本，以啟用可重複使用和參數化的部署。
環境提升	Spinnaker 以受控制的方式促進環境之間的應用程式提升 （例如，從預備到生產）。
與 CI 工具整合	Spinnaker 可以與各種持續整合 (CI) 工具整合，以提供符合 GitOps 原則的完整 CI/CD 管道。
自訂階段和擴充功能	此工具支援自訂階段和延伸，因此團隊可以實作根據其需求量身打造的 GitOps 工作流程。
集中式管理	Spinnaker 提供集中式平台，用於管理跨多個環境和雲端提供者的部署。

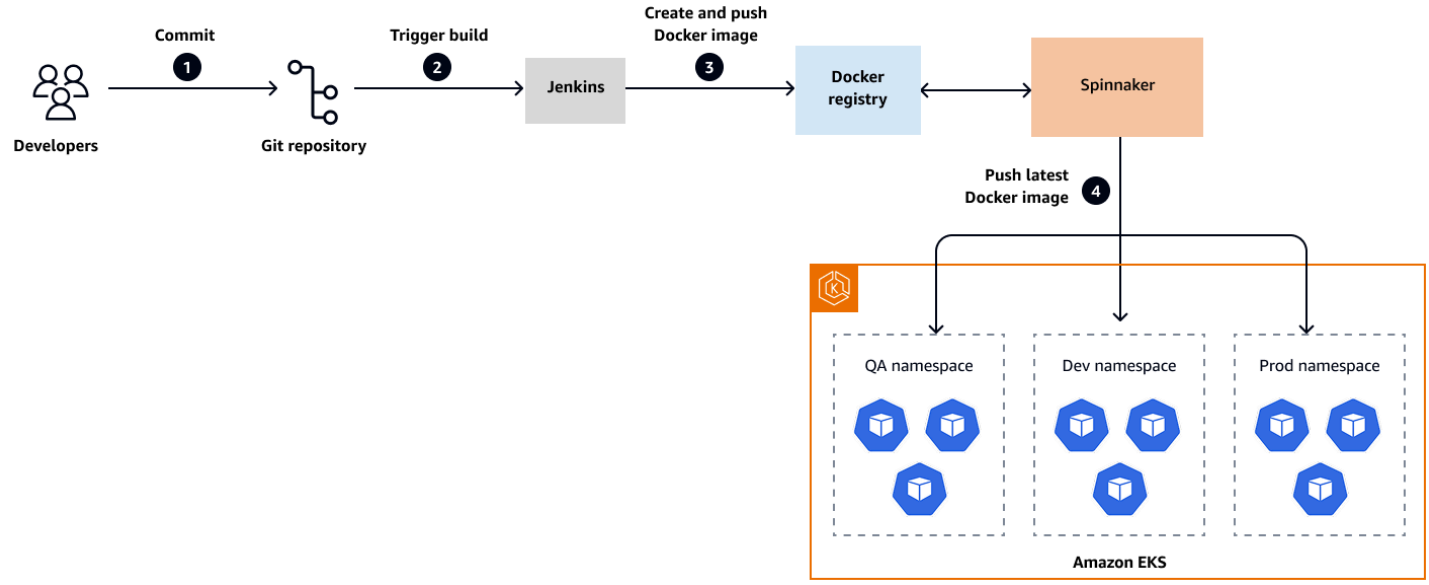
雖然 Spinnaker 主要不是以 GitOps 工具行銷，但其彈性和強大的功能集使其能夠實作 GitOps 工作流程，尤其是在複雜的多雲端環境中。Spinnaker 與 Argo CD 或 Flux 等專用 GitOps 工具之間的主要區別在於，Spinnaker 提供更全面的持續交付平台，具有進階部署策略和多雲端支援。

Spinnaker 的優勢在於能夠跨各種雲端提供者處理複雜的部署案例，以及其對進階部署策略的支援。正確設定 Spinnaker 時，可以有效實作 GitOps 原則。對於想要在各種複雜環境中採用 GitOps 實務的組織而言，這使其成為強大的工具。

如需詳細資訊，請參閱 Spinnaker 文件。

架構

下圖說明使用 Spinnaker 和 Jenkins X 的 GitOps 驅動 CD 工作流程。如需詳細資訊，請參閱 Spinnaker 文件。



其中：

- 步驟 1：程式碼遞交。開發人員將應用程式程式碼變更遞交至 Git 儲存庫。這些變更可能包括應用程式本身、Dockerfiles 或 Kubernetes 資訊清單的更新。
- 步驟 2：Jenkins 建置和建立映像。Jenkins 是由 Git 儲存庫透過 Webhook 或輪詢自動觸發。Jenkins 會建置應用程式、建立 Docker 映像，並將建置映像推送至設定的 Docker 登錄檔（例如 Amazon ECR 或 Docker Hub)。
- 步驟 3：Spinnaker 影像監控和管道觸發。Spinnaker 會持續監控 Docker 登錄檔是否有新映像。偵測到新的映像版本時，Spinnaker 會自動觸發管道以啟動部署程序。
- 步驟 4：部署至目標命名空間。Spinnaker 會將新的 Docker 映像部署到 Amazon EKS。根據管道組態，映像會部署到叢集中的目標命名空間。Spinnaker 可確保部署最新的應用程式版本，同時遵循定義的部署策略，例如藍/綠或金絲雀部署。

Rancher 機群

Rancher Fleet 是一種 GitOps-at-scale解決方案，專為管理多個 Kubernetes 叢集而設計。它嚴格遵守 GitOps 原則，同時專注於可擴展性和多叢集管理。

GitOps 支援

區域圖	工具功能
Git 作為單一事實來源	機群使用 Git 儲存庫做為授權來源，以跨多個叢集定義應用程式和資源的所需狀態。所有組態，包括 Kubernetes 資訊清單、Helm Chart 和自訂資源，都存放在 Git 中。
宣告式組態	機群使用應用程式和資源所需狀態的宣告性描述。這些可以是原始 Kubernetes YAML、Helm Chart、Kustomize 檔案或機群特定的自訂資源。
自動化同步	機群會持續監控 Git 儲存庫的變更。當偵測到 Git 狀態與叢集狀態之間的差異時，它會自動將變更套用至目標叢集。
多叢集管理	機群專為管理跨多個 Kubernetes 叢集的部署而設計。它可以處理來自單一控制平面的數千個叢集。
Kubernetes 原生架構	機群是建置為一組 Kubernetes 自訂資源和控制器。它使用 Kubernetes 中的延伸機制進行 GitOps 操作。
持續對帳	機群會持續比較叢集的實際狀態與 Git 中定義的所需狀態。它會自動更正在這些狀態之間偵測到的任何偏離。
叢集分組和目標鎖定	機群可讓您將叢集和目標部署分組到特定群組或個別叢集。它支援跨不同環境和叢集類型的一致性應用程式部署。
分層組態	機群支援分層組態，可提供具有環境特定浮水印的基礎組態。這符合 GitOps 有效管理多個環境的做法。

區域圖	工具功能
Helm 整合	機群提供 Helm Chart 的原生支援，並可輕鬆管理複雜的應用程式。它可以透過 GitOps 工作流程來版本和管理 Helm 版本。
自訂資源定義 (CRDs)	機群使用 GitRepo 和 Bundle 等自訂資源來定義部署。這些 CRDs 提供 Kubernetes 原生的方式來定義 GitOps 工作流程。
安全性和 RBAC	機群與 Kubernetes RBAC 整合以進行存取控制。它支援安全管理敏感資訊和登入資料。
可觀測性	機群提供有關叢集和應用程式同步狀態的狀態資訊。它提供整個叢集機群 GitOps 程序的洞見。
可擴展性	機群旨在進行擴展，以有效率地管理數千個叢集。它支援企業環境中的大規模 GitOps 操作。
相依性管理	您可以定義不同資源和應用程式之間的相依性。機群可確保在複雜的部署中遵循正確的操作順序。
自訂和可擴展性	機群支援自訂指令碼和生命週期掛鉤，以進階自訂部署。它允許與現有的工具和工作流程整合。
離線和氣隙隔離支援	機群可以在有限或沒有網際網路連線的環境中操作。它支援高安全性或受監管環境中的 GitOps 工作流程。
漸進式推展	機群支援跨叢集的階段推展，允許控制和逐步部署策略。
統一管理界面	機群提供單一界面，用於管理所有叢集的 GitOps 工作流程。它可簡化複雜多叢集環境中的操作。
與其他 Rancher 工具整合	機群與其他 Rancher 工具整合，以提供全方位的 Kubernetes 管理解決方案。

區域圖	工具功能
稽核追蹤和合規	機群會維護所有變更和部署的清楚稽核線索。它可協助您透過版本控制的 Git 型操作來滿足合規要求。

Rancher Fleet 實作這些 GitOps 原則，並高度專注於可擴展性和多叢集管理。其設計特別適合管理不同環境、資料中心或雲端供應商中大量 Kubernetes 叢集的組織。

機群的關鍵差異在於能夠大規模處理 GitOps。此功能對於管理許多叢集的大型企業或受管服務供應商來說特別重要。Argo CD 或 Flux 等工具通常用於個別叢集管理，而機群旨在跨大型叢集機群管理 GitOps。

Rancher Fleet 遵循這些 GitOps 原則，為希望在各種大規模 Kubernetes 環境中實作一致、可擴展且自動化應用程式和資源管理的組織提供解決方案。

如需詳細資訊，請參閱機群文件。

架構

如需架構和工作流程資訊，請參閱機群儲存庫。

Codefresh

Codefresh 是現代 CI/CD 平台，支援 GitOps 原則，特別是 Kubernetes 部署。Codefresh 提供一組完整的 CI/CD 功能，其 GitOps 功能值得注意。

GitOps 支援

區域圖	工具功能
Git 作為單一事實來源	Codefresh 使用 Git 儲存庫做為應用程式程式碼、基礎設施定義和管道組態的授權來源。系統的所有變更都是透過 Git 進行，這可確保完整的歷史記錄和稽核線索。
宣告式組態	Codefresh 使用存放在 Git 中的 YAML 檔案支援宣告管道定義。Kubernetes 資訊清單、Helm

區域圖	工具功能
	Chart、 CloudFormation 範本和其他 IaC 檔案可以在相同的儲存庫中進行版本控制。
GitOps 儀表板	Codefresh 提供專用 GitOps 儀表板，用於視覺化和管理 GitOps 工作流程。它提供 Git 和叢集狀態之間同步狀態的清晰檢視。
自動化同步	Codefresh 會持續監控 Git 儲存庫的變更。它會自動啟動管道，以在偵測到差異時將變更套用至目標環境。
Kubernetes 整合	Codefresh 提供與 Kubernetes 的深度整合，以支援跨多個叢集的 GitOps 樣式部署。它支援各種 Kubernetes 資源和自訂資源定義 (CRDs)。
環境管理	您可以定義和管理多個環境 （例如開發、預備和生產） 做為程式碼。Codefresh 使用 GitOps 實務支援環境之間的提升。
Argo CD 整合	Codefresh 與 Argo CD 整合，以增強 GitOps 功能。它結合了其 CI 功能和 Argo CD 的 CD 強度，以提供完整的 GitOps 解決方案。
Helm 支援	Codefresh 支援 Helm Chart，並透過 GitOps 輕鬆管理複雜的應用程式。它還提供 Helm Chart 版本控制和提升。
漸進式交付	Codefresh 支援進階部署策略，例如 Canary 和藍/綠部署。您可以透過 GitOps 工作流程實作和管理這些策略。
轉返和版本控制	如果在部署後偵測到問題，Codefresh 可讓您輕鬆復原至先前的版本。它會維護部署版本控制，以實現可追蹤性。

區域圖	工具功能
核准工作流程	Codefresh 支援部署的手動和自動核准程序。它可以根據 GitOps 實務，在環境之間進行受控的促銷。
IaC	Codefresh 支援與 IaC 工具整合，例如 CloudFormation 和 Terraform。它啟用基礎設施定義與應用程式程式碼的版本控制。
可觀測性和監控	Codefresh 提供內建的監控和可觀測性功能。它還提供與外部監控工具的整合，以提高可見性。
安全性掃描	Codefresh 包含可整合到 GitOps 工作流程的安全性掃描功能。安全檢查是自動化部署程序的一部分。
稽核線索	Codefresh 會維護所有動作和變更的完整稽核日誌。它支援 GitOps 的可追蹤性和合規方面。
RBAC 和存取控制	Codefresh 實作角色型存取控制 (RBAC)，以進行精細的許可管理。這有助於確保跨團隊和環境的安全 GitOps 操作。
GitOps 自動化	Codefresh 提供自動化 GitOps 工作流程各方面的功能，包括提取請求 (PR) 建立和合併。
多雲端和混合部署	Codefresh 支援跨多個雲端提供者和內部部署環境的 GitOps 工作流程。
範本化和參數化	Codefresh 支援管道和部署組態中的範本。這可啟用可重複使用和參數化的 GitOps 工作流程。
整合式映像管理	Codefresh 提供內建的容器映像管理功能。它將映像建置和部署整合到 GitOps 工作流程中。
秘密管理的 GitOps	Codefresh 提供在 GitOps 工作流程中管理秘密的安全方法。它與外部秘密管理解決方案整合。

區域圖	工具功能
協同合作功能	Codefresh 提供在 GitOps 程序內進行團隊協作的功能。這些功能包括註解、通知和共用儀表板。

GitOps 的 Codefresh 方法因其將 CI/CD 功能與 GitOps 實務整合而值得注意。它旨在提供涵蓋整個軟體交付生命週期的全方位平台，同時遵守 GitOps 原則。

GitOps 區域中 Codefresh 的關鍵差異在於其統一的平台方法，結合了 CI 功能與 CD 和 GitOps 功能。這使得它特別適合想要all-in-one處理複雜 CI/CD 案例，同時實作 GitOps 實務的全方位解決方案的團隊。

Codefresh 為希望在更廣泛的 CI/CD 環境中採用 GitOps 方法的組織提供平台，特別是在使用 Kubernetes 和雲端原生技術時。

如需詳細資訊，請參閱 Codefresh 文件。

Pulumi

Pulumi 是一種 IaC 平台，並非專為 GitOps 而設計。不過，它可以有效地用於實作 GitOps 原則，尤其是雲端基礎設施和 Kubernetes 部署。

GitOps 支援

區域圖	工具功能
IaC	Pulumi 可讓您使用 Python、TypeScript 和 Go 等一般用途程式設計語言來定義基礎設施。這種以程式碼為基礎的方法符合 GitOps 對版本控制的宣告式組態的強調。
Git 作為單一事實來源	Pulumi 中的基礎設施程式碼可以在 Git 儲存庫中儲存和版本控制。這可確保 Git 做為基礎設施定義的單一事實來源。
宣告所需狀態	雖然 Pulumi 使用程式設計語言，但仍會宣告地描述所需的基礎設施狀態。此程式碼會定義基礎

區域圖	工具功能
	設施的外觀，而不是建立它的step-by-step程序。
自動化同步	Pulumi 可與 CI/CD 管道整合，以在 Git 中更新程式碼時自動套用變更。這可啟用基礎設施變更的持續部署，這是關鍵 GitOps 原則。
多雲端和 Kubernetes 支援	Pulumi 支援各種雲端供應商和 Kubernetes，因此您可以在各種環境中遵循 GitOps 實務。此工具可跨不同平台一致管理資源。
狀態管理	Pulumi 會管理基礎設施的狀態，這些基礎設施可以遠端安全地存放。此狀態管理對於 GitOps 實務至關重要，可確保定義狀態與基礎設施實際狀態之間的一致性。
偏離偵測和調校	Pulumi 可以偵測所需狀態（在程式碼中）與基礎設施實際狀態之間的差異。它會根據 GitOps 原則來協調這些差異，以持續進行協調。
政策即程式碼	您可以使用 Pulumi CrossGuard 將政策定義為程式碼，並強制執行政策。這可讓版本控制的 GitOps 樣式管理合規和安全性政策。
秘密管理	Pulumi 提供安全的方法來管理基礎設施程式碼中的敏感資訊。它支援與外部秘密管理系統整合，這對 GitOps 安全實務至關重要。
模組化和可重複使用的元件	Pulumi 支援建立可重複使用的元件和模組。此模組化符合管理複雜多環境部署的 GitOps 實務。
預覽和規劃	Pulumi 提供在套用變更之前預覽變更的功能。這支援對基礎設施進行安全、可預測的變更的 GitOps 原則。

區域圖	工具功能
轉返和歷史記錄	Pulumi 會維護部署歷史記錄，並支援復原至先前的狀態。這符合 GitOps 的可追蹤性和可逆性原則。
基礎設施的持續交付	Pulumi 可以整合到 CI/CD 管道中，以持續交付基礎設施變更。它支援基礎設施程式碼的自動測試和驗證。
RBAC 和存取控制	Pulumi 提供以角色為基礎的存取控制，用於管理誰可以變更基礎設施。這支援 GitOps 安全與控管實務。
可觀測性和記錄	Pulumi 提供基礎設施變更的記錄和監控功能。這些功能支援 GitOps 實務的可觀測性方面。
與其他工具整合	Pulumi 可與雲端中的各種工具整合。這種靈活性允許全面的 GitOps 工作流程。
環境管理	Pulumi 使用具有不同組態的相同程式碼庫來支援管理多個環境（開發、預備、生產）。這符合 GitOps 的一致多環境管理實務。
相依性管理	Pulumi 會處理資源之間的相依性，並確保正確的操作順序。這對涉及相互依賴元件的複雜 GitOps 部署至關重要。
自訂資源提供者	Pulumi 可讓您建立自訂提供者來管理任何 API 驅動的服務。這將 GitOps 實務擴展到標準雲端產品以外的各種資源。
協同合作功能	Pulumi 透過共用狀態和存取控制支援團隊合作。這有助於團隊環境中的 GitOps 工作流程。

透過使用這些 Pulumi 功能，組織可以為其基礎設施實作 GitOps 實務，尤其是在需要精細控制或複雜邏輯，或想要在單一、一致的架構內管理各種雲端和內部部署資源的情況下。

Pulumi 的 GitOps 方法是唯一的，因為它將一般用途程式設計語言的力量和靈活性帶入基礎設施管理，同時遵守 GitOps 原則。對於偏好使用熟悉程式設計語言，並希望將軟體工程最佳實務套用至基礎設施管理的團隊來說，這可能特別有利。

GitOps 中 Pulumi 的主要差異在於使用標準程式設計語言來定義基礎設施。傳統 GitOps 工具通常使用 YAML 或特定網域的語言，而 Pulumi 允許更複雜的邏輯、更好的程式碼重複使用，以及更輕鬆地與現有的開發工作流程整合。

如需詳細資訊，請參閱 Pulumi 文件。

GitOps 工具比較

以下是先前章節討論的九種 GitOps 工具的比較。當您選擇工具時，請考慮您的特定需求、現有的基礎設施、團隊專業知識，以及所需的控制和自訂層級。

易於使用

- Argo CD、Flux 和 Rancher Fleet 通常更容易設定。
- Spinnaker 和 Jenkins X 有更陡峭的學習曲線。
- Weave GitOps 可能需要為進階功能進行更多設定。
- GitLab CI/CD 和 Codefresh 提供整合的體驗。

Kubernetes 整合

- Argo CD、Flux 和 Rancher Fleet 以 Kubernetes 為中心。
- Jenkins X 和 Weave GitOps 提供更廣泛的 DevOps 功能。
- 其他工具支援 Kubernetes，無需獨佔焦點。

CI/CD 功能

- Jenkins X、GitLab CI/CD 和 Codefresh 提供完整的 CI/CD 解決方案。
- Argo CD、Flux 和 Weave GitOps 更專注於工作流程的 CD 方面，通常需要與單獨的 CI 工具整合。

GitOps 純量

- Argo CD 和 Flux 是特別著重於 GitOps 的工具。
- 其他工具將 GitOps 原則納入不同程度。

多雲端支援

- 多雲端案例中的 Spinnaker 和 Pulumi Excel。
- 其他工具可以跨雲端運作，但可能需要額外的設定。

多叢集支援

- 所有工具都支援多叢集部署。
- Argo CD 和 Weave GitOps 具有更進階的多叢集管理功能。

整合

- Flux 具有強大的雲端原生運算基金會 (CNCF) 後端。
- Argo CD 具有大型且活躍的社群。
- Argo CD 和 Flux 具有強大的 Kubernetes 整合。
- Jenkins X 使用更廣泛的 Jenkins 系統。
- Weave GitOps 較新，但隨著強大的商業支援而成長。
- GitLab CI/CD 與 GitLab 緊密整合。
- Rancher Fleet 在 Rancher 系統中運作良好。

社群和支援

- Flux 具有強大的 CNCF 後端。
- Argo CD、GitLab 和 Spinnaker 都有大型社群。
- 大多數工具都提供商業支援。

企業功能

- 根據預設，Weave GitOps 和 Jenkins X 提供更多以企業為重心的功能。
- Argo CD 和 Flux 有企業產品，也可以擴充以供企業使用。

彈性和可擴展性

- Flux 具有高度模組化和可擴展性。
- Argo CD 提供良好的自訂選項。
- Jenkins X 非常可擴展，但可能需要更多精力。
- Weave GitOps 旨在提供完整的解決方案，減少擴充性的需求。

可擴展性

- Spinnaker 和 GitLab CI/CD 以企業可擴展性著稱。
- Argo CD 和 Flux 可妥善處理大規模的 Kubernetes 部署。

基礎設施管理

- Pulumi 專注於基礎設施管理。
- Weave GitOps 和 Flux 提供良好的 IaC 功能。

程式設計模型和語言支援

- 在 Pulumi 中，您可以使用 Python、Go、TypeScript、C# 和 Java 等一般用途程式設計語言來定義基礎設施。Pulumi 使用標準語言，可將基礎設施程式碼與熟悉的開發工作流程、測試實務和複雜邏輯整合。
- Terraform 使用 HashiCorp 組態語言 (HCL)。
- CloudFormation 使用 JSON 和 YAML 範本。
- Argo CD、Flux、Rancher Fleet、Weave GitOps、Spinnaker 和 GitLab CI/CD 主要管理 YAML 或宣告式組態檔案。
- Jenkins X 管理 YAML 和以指令碼為基礎的管道，但本質上不提供 IaC 的一般用途程式設計。

Argo CD 和 Flux 使用案例

本節著重於提供純 GitOps 功能的兩種工具：Argo CD 和 Flux。在這種情況下，純 GitOps 是指 Git 儲存庫做為應用程式和基礎設施所需狀態之單一事實來源的模型。所有變更都是透過 Git 遞交進行，系統會自動同步即時環境，以符合儲存庫中定義的狀態。Git 操作之外不需要手動介入。

一般考量

- 您可能偏好在視覺化管理和以應用程式為中心的工作流程很重要的環境中使用 Argo CD。
- 如果您需要更輕量的解決方案、強大的多租用戶或與更廣泛的雲端原生運算基礎 (CNCF) 網路深度整合，您可以選擇 Flux。
- Argo CD 通常吸引從傳統 CI/CD 轉換到 GitOps 的團隊，因為其直覺式 UI。
- 在已建立 CLI 型工作流程和 IaC 實務的雲端原生環境中，通常偏好 Flux。

最後，Argo CD 和 Flux 之間的選擇通常取決於您的特定組織需求、現有工具和團隊偏好設定。這兩種工具都能夠處理大多數 GitOps 案例，因此我們建議您根據您的特定使用案例和需求進行評估。

Argo CD 使用案例

視覺化管理：

- 當您需要易於使用的 UI 來管理部署和視覺化應用程式狀態時。
- 對於偏好圖形界面進行監控和故障診斷的團隊。

以應用程式為中心的方法：

- 當您想要在應用程式層級管理部署，而不是管理個別資源時。
- 適用於根據應用程式概念建構其部署的組織。

多叢集管理：

- 當跨多個叢集管理部署時，這是主要需求。
- 適用於具有許多叢集的複雜分散式環境。

回復和同步波：

- 當您需要精細控制部署程序時，包括同步波紋和手動介入。
- 對於需要複雜復原策略的案例。

與現有工具整合：

- 當您已在 Argo 專案中使用其他工具時，例如 Argo Workflows 和 Argo Events。

企業環境：

- 對於預設需要強大 RBAC 和單一登入整合的大型企業。

磁通使用案例

輕量型部署：

- 當您需要更輕量、資源密集度較低的 GitOps 解決方案時。
- 對於可能限制資源的邊緣運算或 IoT 案例。

自動化映像更新：

- 當自動偵測和部署新的容器映像是關鍵需求時。
- 對於專注於持續部署並經常更新映像的團隊。

多租戶：

- 當需要強大的多租用戶支援時，特別是在共用叢集環境中。
- 對於在團隊或專案之間具有嚴格區隔的服務提供者或大型組織。

IaC：

- 透過相同的 GitOps 工作流程管理應用程式和基礎設施時很重要。
- 對於大量投資於 IaC 範例的團隊。

Helm 整合：

- 當大量使用 Helm Chart 是部署策略的一部分時。
- 適用於具有複雜 Helm 型部署的環境。

CNCF 專案整合：

- 當與其他 CNCF 專案緊密整合很重要時。
- 適用於符合 CNCF 技術和原則的組織。

模組化架構：

- 當您需要彈性來僅使用 GitOps 工具組的特定元件時。
- 對於想要使用模組化元件建置自訂 GitOps 工作流程的團隊。

漸進式交付：

- 當 Canary Releases 或 A/B 測試等進階部署策略是核心需求時。

功能比較

區域圖	Argo CD	磁通
支援核心 GitOps 原則	✓ 是	✓ 是
架構	實作 Kubernetes GitOps 工作流程的End-to-end應用程式	為 GitOps 提供 Kubernetes CRDs和控制器
設定	簡便	複雜
Helm 支援	✓ 是	✓ 是
Kustomize 支援	✓ 是	✓ 是
整合式 GUI	CLI 和功能完整的 Web UI	CLI 和選用的輕量型 Web 介面
RBAC 支援	精細控制	Kubernetes 原生 RBAC

區域圖	Argo CD	磁通
多租戶和多叢集支援	對多叢集的卓越支援	對多租戶的卓越支援
單一登入身分驗證	✅ 是	✅ 是
同步自動化	同步視窗的能力	能夠設定調校間隔
部分同步	✅ 是	❌否
調校程序	支援手動和自動同步。有幾種不同的策略可用。	支援手動和自動同步。
可擴展性	支援自訂 lugins。有限的自訂選項。	支援自訂控制器。良好的可擴展性和第三方整合。
Cummunity 支援	大型且活躍的社群。	較小但不斷成長的社群。
可擴展性	良好的可擴展性，但受限於 Web UI 的資料擷取速率。社群分析建議支援數萬個應用程式。	清楚的水平和垂直可擴展性指南，最多可達數萬個應用程式。

選擇 GitOps 工具的最佳實務

本節提供為您的 EKS 叢集選擇 GitOps 工具的考量事項、秘訣和最佳實務。正確的選擇取決於您的特定內容、需求和長期策略。在做出最終決策之前，使用您最熱門的選擇進行概念驗證通常很有幫助。

評估組織的需求和功能：

- 考慮您的團隊目前的技能集和學習新工具的意願。
- 評估 Amazon EKS 環境的複雜性。（例如，您是使用單一叢集還是多個叢集？）
- 確定您對合規、安全和可擴展性的特定要求。

> **最佳實務**
>
> 建立詳細需求文件，概述必要的功能，以及實用但非必要的功能。

評估工具成熟度和採用：

- 研究潛在 GitOps 工具的成熟度及其在業界的採用率。
- 尋找在 Amazon EKS 環境中具有經驗證追蹤記錄的工具。

> **最佳實務**
>
> 優先考慮廣泛採用並在雲端原生運算基金會 (CNCF) 網路中具有強大影響力的工具。

考慮與您現有的工具鏈整合：

- 評估 GitOps 工具與您目前的 CI/CD 管道整合的程度、監控解決方案和其他操作工具。
- 尋找與 的原生整合， AWS 服務 例如 IAM、Amazon ECR 和 CloudWatch。

> **最佳實務**
>
> 在做出最終決策之前，建立概念驗證來測試整合功能。

評估安全功能：

- 優先考慮具有強大角色型存取控制 (RBAC) 功能，並與 IAM 完美整合的工具。
- 尋找支援安全秘密管理和政策強制執行的功能。

> **最佳實務**
>
> 選擇支援 GitOps 型安全實務的工具，包括做為程式碼的政策和自動化合規檢查。

評估可擴展性和效能：

- 考慮該工具如何對大量應用程式和叢集執行。
- 評估其對叢集效能和資源耗用量的影響。

> **最佳實務**
>
> 使用類似於生產環境的工作負載執行效能測試，以確保工具可以處理您的擴展。

考慮多叢集和多環境支援：

- 如果您有或計劃擁有多個 EKS 叢集，請優先考慮具有強大多叢集管理功能的工具。
- 尋找支援跨不同環境（例如開發、預備和生產）一致部署的功能。

> **最佳實務**
>
> 選擇允許集中管理多個叢集的工具，同時維護環境特定的組態。

評估可觀測性和監控功能：

- 尋找可讓您清楚了解部署狀態和叢集運作狀態的工具。
- 考慮工具與現有監控和記錄解決方案的整合程度。

> **最佳實務**
>
> 優先考慮提供可自訂儀表板和警示機制的工具，以主動偵測問題。

評估學習曲線和文件：

- 評估工具文件的品質和完整性。
- 考慮培訓資源和社群支援的可用性。

> **最佳實務**
>
> 選擇具有妥善維護的文件、作用中的社群論壇，以及官方訓練計畫或認證的工具。

考慮成本和資源使用率：

- 評估採用工具的直接成本（例如授權和支援）和間接成本（例如營運開銷和訓練成本）。
- 評估工具在運算和儲存資源消耗方面的效率。

> **最佳實務**
>
> 執行包含短期和長期成本的總擁有成本 (TCO) 分析。

評估彈性和自訂選項：

- 尋找可讓您自訂工作流程以符合特定需求的工具。
- 透過外掛程式或 APIs 考慮工具的可擴展性。

> **最佳實務**
>
> 選擇可平衡預設功能和自訂您唯一需求能力的工具。

評估持續交付和漸進式部署功能：

- 尋找支援進階部署策略的工具，例如 Canary Releases 和藍/綠部署。
- 評估實作和管理這些策略的難易度。

> **最佳實務**
>
> 排定工具的優先順序，為漸進式交付模式提供內建支援，將部署的風險降至最低。

考慮廠商鎖定和可攜性：

- 評估工具對特定雲端提供者或技術的相依性。
- 如有需要，請考慮未來遷移到不同工具的難易度。

> **最佳實務**
>
> 偏好使用開放標準的工具，並為 GitOps 組態提供匯出功能。

評估社群支援和延伸：

- 查看使用者社群的大小和活動。
- 評估第三方整合和外掛程式的可用性。

> **最佳實務**
>
> 加入社群論壇或使用者群組，在做出決策之前，先從其他使用者獲得第一手的體驗。

考慮合規和稽核要求：

- 評估工具支援您合規需求的程度，包括稽核追蹤和報告。
- 尋找有助於維護和示範合規的功能。

> **最佳實務**
>
> 選擇可提供完整稽核日誌並支援產生合規報告的工具。

評估復原和災難復原功能：

- 評估復原機制的簡易性和可靠性。
- 考慮工具如何支援災難復原案例。

> **最佳實務**
>
> 在評估過程中，徹底測試轉返和復原程序。

常見問答集

問：哪些是 Amazon EKS 最熱門的 GitOps 工具？

答：Amazon EKS 最熱門的 GitOps 工具包括 Argo CD、Flux、Jenkins X 和 GitLab CI/CD。每個工具都有優勢，但 Argo CD 和 Flux 對於其 Kubernetes 原生方法和強大的社群支援特別受到重視。

問：GitOps 如何改善 EKS 叢集管理？

答：GitOps 透過提供基礎設施的版本控制、自動化部署、透過宣告式組態改善安全性、更輕鬆的轉返和更好的可稽核性，來改善 EKS 叢集管理。它還增強了協同合作，並減少了部署中的人為錯誤。

問：我應該在 Amazon EKS 的 GitOps 工具中尋找哪些主要功能？

答：要尋找的主要功能包括：無縫 Amazon EKS 整合、強大的 RBAC、多叢集支援、可觀測性功能、對漸進式交付策略的支援、可擴展性和與 IAM 和 Amazon ECR AWS 服務 的整合。

問：在 Amazon EKS 中實作 GitOps 時，如何確保安全性？

答：為確保安全性，請選擇具有與 IAM 強式 RBAC 整合的工具、安全秘密管理、對加密 Git 儲存庫的支援，以及實作安全政策做為程式碼的能力。此外，請確認工具提供完整的稽核日誌。

問：GitOps 工具是否可以處理多叢集 Amazon EKS 環境？

答：是，Argo CD 和 Flux 等 GitOps 工具具有強大的多叢集管理功能。它們可讓您從單一控制平面管理多個 EKS 叢集，以確保環境之間的一致性。

問：GitOps 工具如何與現有的 CI/CD 管道整合？

答：GitOps 工具通常會做為管道的部署階段，與現有的 CI/CD 管道整合。當變更推送到 Git 儲存庫時，CI 工具可能會觸發它們，而且它們會將部署程序自動化到 EKS 叢集。

問：在 Amazon EKS 中實作 GitOps 的挑戰是什麼？

答：常見的挑戰包括安全地管理秘密、確保適當的存取控制、處理具狀態的應用程式、管理 Git 和叢集狀態之間的偏離，以及根據 GitOps 模型調整團隊工作流程。

問：GitOps 工具如何處理 Amazon EKS 中的轉返？

答：GitOps 工具通常會透過還原至 Git 儲存庫中的先前遞交來處理轉返。這會自動觸發先前已知良好狀態的部署，這會導致快速且可靠的轉返。

問：GitOps 工具是否可以管理 Amazon EKS 附加元件和其他 AWS 資源？

答：許多 GitOps 工具可以管理 Amazon EKS 附加元件和一些 AWS 資源，特別是當它們與 Terraform 或 等 IaC 工具結合時 CloudFormation。不過，此功能的範圍可能有所不同；如需每個工具的特定資訊，請參閱 GitOps 工具一節。

問：GitOps 工具如何支援 Amazon EKS 中的合規要求？

答：GitOps 工具透過提供所有變更的清晰稽核線索、強制執行核准程序、實作政策做為自動合規檢查的程式碼，以及提供詳細的記錄和報告功能，來支援合規。

問：在 Amazon EKS 中實作 GitOps 的學習曲線是什麼？

答：學習曲線可能會因工具和您團隊的現有知識而有所不同。一般而言，熟悉 Git、Kubernetes 和 Amazon EKS 的團隊將比其他團隊更快地適應。最熱門的工具提供廣泛的文件和訓練資源，以簡化採用。

問：GitOps 工具如何處理 Amazon EKS 中的秘密管理？

答：GitOps 工具通常與外部秘密管理解決方案整合，例如 AWS Secrets Manager 或 HashiCorp Vault。有些工具也為存放在 Git 儲存庫中的秘密提供內建加密。

問：GitOps 工具是否可以在 Amazon EKS 中使用無狀態和有狀態應用程式？

答：是，GitOps 工具可以同時適用於無狀態和有狀態的應用程式。不過，管理具狀態應用程式通常需要額外考量，例如處理持久性磁碟區，並確保更新期間的資料一致性。

問：GitOps 工具如何支援 Amazon EKS 中的 Canary 或藍/綠部署？

答：許多 GitOps 工具提供進階部署策略的內建支援。他們可以管理新版本的逐步推出、監控問題，並在偵測到問題時自動復原。所有這些操作都會在 Git 儲存庫中定義為程式碼。

問：使用 GitOps 工具與**`kubectl apply`**搭配 CI/CD 管道使用 有何不同？

答：GitOps 工具提供優於簡單`kubectl apply`命令的優勢，包括自動偏離偵測和調校、透過提取式部署提高安全性、提高可稽核性，以及更複雜的部署策略。它們也提供更全面的方法來管理整個叢集狀態。

資源

下列資源提供官方文件、實用指南、案例研究和深入分析，可協助您在為 EKS 叢集選擇 GitOps 工具時做出明智的決定。它們涵蓋 GitOps 的各種層面，包括實作策略、最佳實務、不同工具之間的比較，以及實際體驗。

AWS 資源：

- Amazon EKS 文件
- 使用 GitOps 自動化 Amazon EKS (AWS 部落格文章）
- GitOps on EKS with Weaveworks 簡介 (AWS workshop)
- Flux 實驗室 (Amazon EKS 研討會）
- Argo CD 實驗室 (Amazon EKS 研討會）

GitOps 和工具文件：

- 持續部署和漸進式安全性的 GitOps 最佳實務 （隨需 DevOps.com 網路研討會）
- Kubernetes 文件
- Argo CD 文件
- 磁通文件
- 編織 GitOps 文件
- Jenkins X 文件
- GitLab CI/CD 文件
- Spinnaker 文件
- Rancher Fleet 文件
- Codefresh 文件

文件歷史紀錄

下表描述了本指南的重大變更。如果您想收到有關未來更新的通知，可以訂閱 RSS 摘要。

變更	描述	日期
初次出版	—	2025 年 4 月 30 日

AWS 規範性指引詞彙表

以下是 AWS Prescriptive Guidance 提供的策略、指南和模式中常用的術語。若要建議項目，請使用詞彙表末尾的提供意見回饋連結。

數字

7 R

將應用程式移至雲端的七種常見遷移策略。這些策略以 Gartner 在 2011 年確定的 5 R 為基礎，包括以下內容：

- 重構/重新架構 – 充分利用雲端原生功能來移動應用程式並修改其架構，以提高敏捷性、效能和可擴展性。這通常涉及移植作業系統和資料庫。範例：將您的現場部署 Oracle 資料庫 遷移至 Amazon Aurora PostgreSQL 相容版本。
- 平台轉換 (隨即重塑) – 將應用程式移至雲端，並引入一定程度的優化以利用雲端功能。範例：將您的現場部署 Oracle 資料庫 遷移至 中的 Amazon Relational Database Service (Amazon RDS) for Oracle AWS 雲端。
- 重新購買 (捨棄再購買) – 切換至不同的產品，通常從傳統授權移至 SaaS 模型。範例：將您的客戶關係管理 (CRM) 系統 遷移至 Salesforce.com。
- 主機轉換 (隨即轉移) – 將應用程式移至雲端，而不進行任何變更以利用雲端功能。範例：將您的現場部署 Oracle 資料庫 遷移至 中 EC2 執行個體上的 Oracle AWS 雲端。
- 重新放置 (虛擬機器監視器等級隨即轉移) – 將基礎設施移至雲端，無需購買新硬體、重寫應用程式或修改現有操作。您可以將伺服器從內部部署平台遷移到相同平台的雲端服務。範例：將 Microsoft Hyper-V應用程式 遷移至 AWS。
- 保留 (重新檢視) – 將應用程式保留在來源環境中。其中可能包括需要重要重構的應用程式，且您希望將該工作延遲到以後，以及您想要保留的舊版應用程式，因為沒有業務理由來進行遷移。
- 淘汰 – 解除委任或移除來源環境中不再需要的應用程式。

A

A2A Agent-to-Agent)

支援任務委派和狀態轉移的agent-to-agent協同合作的狀態通訊協定。

ABAC

請參閱屬性型存取控制。

抽象服務

請參閱 受管服務。

ACID

請參閱原子性、一致性、隔離性、耐久性。

主動-主動式遷移

一種資料庫遷移方法，其中來源和目標資料庫保持同步 (透過使用雙向複寫工具或雙重寫入操作)，且兩個資料庫都在遷移期間處理來自連接應用程式的交易。此方法支援小型、受控制批次的遷移，而不需要一次性切換。它更靈活，但比主動-被動遷移需要更多的工作。

主動-被動式遷移

一種資料庫遷移方法，其中來源和目標資料庫保持同步，但只有來源資料庫會在資料複寫至目標資料庫時處理來自連線應用程式的交易。目標資料庫在遷移期間不接受任何交易。

客服人員

一種 AI 系統，可使用工具自動推理、規劃和採取行動來實現目標。

客服人員操作

在生產環境中大規模建置、測試、部署和執行 AI 代理器的操作實務。

彙總函數

在一組資料列上操作並計算群組單一傳回值的 SQL 函數。彙總函數的範例包括 SUM和 MAX。

AI

請參閱人工智慧。

AIOps

請參閱人工智慧操作。

匿名化

永久刪除資料集中個人資訊的程序。匿名化有助於保護個人隱私權。匿名資料不再被視為個人資料。

反模式

經常用於經常性問題的解決方案，其中解決方案具有反生產力、無效或比替代解決方案更有效。

應用程式控制

一種安全方法，僅允許使用核准的應用程式，以協助保護系統免受惡意軟體攻擊。

應用程式組合

有關組織使用的每個應用程式的詳細資訊的集合，包括建置和維護應用程式的成本及其商業價值。此資訊是產品組合探索和分析程序的關鍵，有助於識別要遷移、現代化和優化的應用程式並排定其優先順序。

人工智慧 (AI)

電腦科學領域，致力於使用運算技術來執行通常與人類相關的認知功能，例如學習、解決問題和識別模式。如需詳細資訊，請參閱什麼是人工智慧？

人工智慧操作 (AIOps)

使用機器學習技術解決操作問題、減少操作事件和人工干預以及提高服務品質的程序。如需有關如何在 AWS 遷移策略中使用 AIOps 的詳細資訊，請參閱操作整合指南。

非對稱加密

一種加密演算法，它使用一對金鑰：一個用於加密的公有金鑰和一個用於解密的私有金鑰。您可以共用公有金鑰，因為它不用於解密，但對私有金鑰存取應受到高度限制。

原子性、一致性、隔離性、耐久性 (ACID)

一組軟體屬性，即使在出現錯誤、電源故障或其他問題的情況下，也能確保資料庫的資料有效性和操作可靠性。

屬性型存取控制 (ABAC)

根據使用者屬性 (例如部門、工作職責和團隊名稱) 建立精細許可的實務。如需詳細資訊，請參閱《 AWS Identity and Access Management (IAM) 文件》中的 ABAC for AWS。

授權資料來源

存放主要版本資料的位置，被視為最可靠的資訊來源。您可以將授權資料來源中的資料複製到其他位置，以處理或修改資料，例如匿名、修訂或假名化資料。

可用區域

中的不同位置 AWS 區域 ，可隔離其他可用區域中的故障，並提供相同區域中其他可用區域的低成本、低延遲網路連線。

AWS 雲端採用架構 (AWS CAF)

的指導方針和最佳實務架構 AWS ，可協助組織制定高效且有效的計劃，以成功地移至雲端。 AWS CAF 將指導方針組織到六個重點領域：業務、人員、治理、平台、安全和營運。業務、人員和控管層面著重於業務技能和程序；平台、安全和操作層面著重於技術技能和程序。例如，人員層面針對處理人力資源 (HR)、人員配備功能和人員管理的利害關係人。因此， AWS CAF 為人員開發、訓練和通訊提供指引，協助組織做好成功採用雲端的準備。如需詳細資訊，請參閱 AWS CAF 網站和 AWS CAF 白皮書。

AWS 工作負載資格架構 (AWS WQF)

一種工具，可評估資料庫遷移工作負載、建議遷移策略，並提供工作預估值。 AWS WQF 隨附於 AWS Schema Conversion Tool (AWS SCT)。它會分析資料庫結構描述和程式碼物件、應用程式程式碼、相依性和效能特性，並提供評估報告。

B

錯誤的機器人

旨在中斷或傷害個人或組織的機器人。

BCP

請參閱業務持續性規劃。

行為圖

資源行為的統一互動式檢視，以及一段時間後的互動。您可以將行為圖與 Amazon Detective 搭配使用來檢查失敗的登入嘗試、可疑的 API 呼叫和類似動作。如需詳細資訊，請參閱偵測文件中的行為圖中的資料。

大端序系統

首先儲存最高有效位元組的系統。另請參閱 Endianness。

二進制分類

預測二進制結果的過程 (兩個可能的類別之一)。例如，ML 模型可能需要預測諸如「此電子郵件是否是垃圾郵件？」等問題 或「產品是書還是汽車？」

Bloom 篩選條件

一種機率性、記憶體高效的資料結構，用於測試元素是否為集的成員。

藍/綠部署

一種部署策略，您可以在其中建立兩個不同但相同的環境。您可以在一個環境（藍色）中執行目前的應用程式版本，並在另一個環境（綠色）中執行新的應用程式版本。此策略可協助您快速復原，並將影響降至最低。

機器人

透過網際網路執行自動化任務並模擬人類活動或互動的軟體應用程式。有些機器人有用或有益，例如在網際網路上編製資訊索引的 Web 爬蟲程式。某些其他機器人稱為惡意機器人，旨在中斷或傷害個人或組織。

殭屍網路

受到惡意軟體感染且受單一方控制之機器人的網路，稱為機器人繼承器或機器人運算子。殭屍網路是擴展機器人及其影響的最佳已知機制。

分支

程式碼儲存庫包含的區域。儲存庫中建立的第一個分支是主要分支。您可以從現有分支建立新分支，然後在新分支中開發功能或修正錯誤。您建立用來建立功能的分支通常稱為功能分支。當準備好發佈功能時，可以將功能分支合併回主要分支。如需詳細資訊，請參閱關於分支 (GitHub 文件)。

碎片存取

在特殊情況下，並透過核准的程序，讓使用者快速取得他們通常無權存取 AWS 帳戶 之 的存取權。如需詳細資訊，請參閱 Well-Architected 指南中的 AWS 實作碎片程序指標。

棕地策略

環境中的現有基礎設施。對系統架構採用棕地策略時，可以根據目前系統和基礎設施的限制來設計架構。如果正在擴展現有基礎設施，則可能會混合棕地和綠地策略。

緩衝快取

儲存最常存取資料的記憶體區域。

業務能力

業務如何創造價值 (例如，銷售、客戶服務或營銷)。業務能力可驅動微服務架構和開發決策。如需詳細資訊，請參閱在 AWS上執行容器化微服務白皮書的圍繞業務能力進行組織部分。

業務連續性規劃 (BCP)

一種解決破壞性事件 (如大規模遷移) 對營運的潛在影響並使業務能夠快速恢復營運的計畫。

C

CAF

請參閱AWS 雲端採用架構。

Canary 部署

版本對最終使用者的緩慢和增量版本。當您有信心時，您可以部署新版本並完全取代目前的版本。

CCoE

請參閱 Cloud Center of Excellence。

CDC

請參閱變更資料擷取。

變更資料擷取 (CDC)

追蹤對資料來源 (例如資料庫表格) 的變更並記錄有關變更的中繼資料的程序。您可以將 CDC 用於各種用途，例如稽核或複寫目標系統中的變更以保持同步。

混沌工程

故意引入故障或破壞性事件，以測試系統的彈性。您可以使用 AWS Fault Injection Service (AWS FIS) 執行實驗，為您的 AWS 工作負載帶來壓力，並評估其回應。

CI/CD

請參閱持續整合和持續交付。

分類

有助於產生預測的分類程序。用於分類問題的 ML 模型可預測離散值。離散值永遠彼此不同。例如，模型可能需要評估影像中是否有汽車。

公民開發人員

在沒有專業技術技能的情況下，使用無程式碼/低程式碼平台建立 AI 應用程式的商業使用者。

用戶端加密

在目標 AWS 服務 接收資料之前，在本機加密資料。

雲端卓越中心 (CCoE)

一個多學科團隊，可推動整個組織的雲端採用工作，包括開發雲端最佳實務、調動資源、制定遷移時間表以及領導組織進行大規模轉型。如需詳細資訊，請參閱 AWS 雲端 企業策略部落格上的 CCoE 文章。

雲端運算

通常用於遠端資料儲存和 IoT 裝置管理的雲端技術。雲端運算通常連接到邊緣運算技術。

雲端操作模型

在 IT 組織中，用於建置、成熟和最佳化一或多個雲端環境的操作模型。如需詳細資訊，請參閱建置您的雲端操作模型。

採用雲端階段

組織在遷移至 時通常會經歷的四個階段 AWS 雲端：

- 專案 – 執行一些與雲端相關的專案以進行概念驗證和學習用途
- 基礎 – 進行基礎投資以擴展雲端採用 (例如，建立登陸區域、定義 CCoE、建立營運模型)
- 遷移 – 遷移個別應用程式
- 重塑 – 優化產品和服務，並在雲端中創新

部落格文章中的 Stephen Orban 定義了這些階段： AWS 雲端 企業策略部落格上的邁向雲端優先之旅和採用階段。如需有關它們如何與 AWS 遷移策略關聯的資訊，請參閱遷移整備指南。

CMDB

請參閱組態管理資料庫。

程式碼儲存庫

透過版本控制程序來儲存及更新原始程式碼和其他資產 (例如文件、範例和指令碼) 的位置。常見的雲端儲存庫包括 GitHub或 Bitbucket Cloud。程式碼的每個版本都稱為分支。在微服務結構中，每個儲存庫都專用於單個功能。單一 CI/CD 管道可以使用多個儲存庫。

冷快取

一種緩衝快取，它是空的、未填充的，或者包含過時或不相關的資料。這會影響效能，因為資料庫執行個體必須從主記憶體或磁碟讀取，這比從緩衝快取讀取更慢。

冷資料

很少存取且通常是歷史資料的資料。查詢這類資料時，通常可接受慢查詢。將此資料移至效能較低且成本較低的儲存層或類別，可以降低成本。

電腦視覺 (CV)

使用機器學習從數位影像和影片等視覺化格式分析和擷取資訊的 AI 欄位。例如，Amazon SageMaker AI 提供 CV 的影像處理演算法。

組態偏離

對於工作負載，組態會從預期狀態變更。這可能會導致工作負載不合規，而且通常是漸進和無意的。

組態管理資料庫 (CMDB)

儲存和管理有關資料庫及其 IT 環境的資訊的儲存庫，同時包括硬體和軟體元件及其組態。您通常在遷移的產品組合探索和分析階段使用 CMDB 中的資料。

一致性套件

您可以組合的 AWS Config 規則和修補動作集合，以自訂您的合規和安全檢查。您可以使用 YAML 範本，將一致性套件部署為 AWS 帳戶 和 區域中或整個組織的單一實體。如需詳細資訊，請參閱 AWS Config 文件中的一致性套件。

持續整合和持續交付 (CI/CD)

自動化軟體發行程序的來源、建置、測試、暫存和生產階段的程序。CI/CD 通常被描述為管道。CI/CD 可協助您將程序自動化、提升生產力、改善程式碼品質以及加快交付速度。如需詳細資訊，請參閱持續交付的優點。CD 也可表示持續部署。如需詳細資訊，請參閱持續交付與持續部署。

CV

請參閱電腦視覺。

D

靜態資料

網路中靜止的資料，例如儲存中的資料。

資料分類

根據重要性和敏感性來識別和分類網路資料的程序。它是所有網路安全風險管理策略的關鍵組成部分，因為它可以協助您確定適當的資料保護和保留控制。資料分類是 AWS Well-Architected Framework 中安全支柱的元件。如需詳細資訊，請參閱資料分類。

資料偏離

生產資料與用於訓練 ML 模型的資料之間有意義的變化，或輸入資料隨時間有意義的變更。資料偏離可以降低 ML 模型預測的整體品質、準確性和公平性。

傳輸中的資料

在您的網路中主動移動的資料，例如在網路資源之間移動。

資料網格

架構架構，提供分散式、分散式資料擁有權與集中式管理。

資料最小化

僅收集和處理嚴格必要資料的原則。在 中實作資料最小化 AWS 雲端 可以降低隱私權風險、成本和分析碳足跡。

資料周邊

AWS 環境中的一組預防性防護機制，可協助確保只有信任的身分才能從預期的網路存取信任的資源。如需詳細資訊，請參閱在 上建置資料周邊 AWS。

資料預先處理

將原始資料轉換成 ML 模型可輕鬆剖析的格式。預處理資料可能意味著移除某些欄或列，並解決遺失、不一致或重複的值。

資料來源

在整個資料生命週期中追蹤資料的來源和歷史記錄的程序，例如資料的產生、傳輸和儲存方式。

資料主體

正在收集和處理資料的個人。

資料倉儲

支援商業智慧的資料管理系統，例如 分析。資料倉儲通常包含大量歷史資料，通常用於查詢和分析。

資料庫定義語言 (DDL)

用於建立或修改資料庫中資料表和物件之結構的陳述式或命令。

資料庫處理語言 (DML)

用於修改 (插入、更新和刪除) 資料庫中資訊的陳述式或命令。

DDL

請參閱資料庫定義語言。

深度整體

結合多個深度學習模型進行預測。可以使用深度整體來獲得更準確的預測或估計預測中的不確定性。

深度學習

一個機器學習子領域，它使用多層人工神經網路來識別感興趣的輸入資料與目標變數之間的對應關係。

深度防禦

這是一種資訊安全方法，其中一系列的安全機制和控制項會在整個電腦網路中精心分層，以保護網路和其中資料的機密性、完整性和可用性。當您在 上採用此策略時 AWS，您可以在 AWS Organizations 結構的不同層新增多個控制項，以協助保護資源。例如，defense-in-depth方法可能會結合多重要素驗證、網路分割和加密。

委派的管理員

在 中 AWS Organizations，相容的服務可以註冊 AWS 成員帳戶，以管理組織的帳戶和管理該服務的許可。此帳戶稱為該服務的委派管理員。如需詳細資訊和相容服務清單，請參閱 AWS Organizations 文件中的可搭配 AWS Organizations運作的服務。

deployment

在目標環境中提供應用程式、新功能或程式碼修正的程序。部署涉及在程式碼庫中實作變更，然後在應用程式環境中建置和執行該程式碼庫。

開發環境

請參閱 環境。

偵測性控制

一種安全控制，用於在事件發生後偵測、記錄和提醒。這些控制是第二道防線，提醒您注意繞過現有預防性控制的安全事件。如需詳細資訊，請參閱在 AWS上實作安全控制中的偵測性控制。

開發值串流映射 (DVSM)

一種程序，用於識別對軟體開發生命週期中的速度和品質造成負面影響的限制並排定優先順序。DVSM 擴展了最初專為精簡製造實務設計的價值串流映射程序。它著重於透過軟體開發程序建立和移動價值所需的步驟和團隊。

數位分身

真實世界系統的虛擬呈現，例如建築物、工廠、工業設備或生產線。數位分身支援預測性維護、遠端監控和生產最佳化。

維度資料表

在星星結構描述中，較小的資料表包含有關事實資料表中量化資料的資料屬性。維度資料表屬性通常是文字欄位或離散數字，其行為類似於文字。這些屬性通常用於查詢限制、篩選和結果集標記。

災難

防止工作負載或系統在其主要部署位置中實現其業務目標的事件。這些事件可能是自然災難、技術故障或人為動作的結果，例如意外設定錯誤或惡意軟體攻擊。

災難復原 (DR)

您用來將災難造成的停機時間和資料遺失降至最低的策略和程序。如需詳細資訊，請參閱 AWS Well-Architected Framework 中的 上工作負載的災難復原 AWS：雲端中的復原。

DML

請參閱資料庫處理語言。

領域驅動的設計

一種開發複雜軟體系統的方法，它會將其元件與每個元件所服務的不斷發展的領域或核心業務目標相關聯。Eric Evans 在其著作 Domain-Driven Design: Tackling Complexity in the Heart of Software (Boston: Addison-Wesley Professional, 2003) 中介紹了這一概念。如需有關如何將領域驅動的設計與 strangler fig 模式搭配使用的資訊，請參閱使用容器和 Amazon API Gateway 逐步現代化舊版 Microsoft ASP.NET (ASMX) Web 服務。

DR

請參閱災難復原。

偏離偵測

追蹤與基準組態的偏差。例如，您可以使用 AWS CloudFormation 來偵測系統資源中的偏離，也可以使用 AWS Control Tower 來偵測登陸區域中可能影響控管要求合規性的變更。 https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/using-cfn-stack-drift.html

DVSM

請參閱開發值串流映射。

E

EDA

請參閱探索性資料分析。

EDI

請參閱電子資料交換。

邊緣運算

提升 IoT 網路邊緣智慧型裝置運算能力的技術。與雲端運算相比，邊緣運算可以減少通訊延遲並改善回應時間。

電子資料交換 (EDI)

在組織之間自動交換商業文件。如需詳細資訊，請參閱什麼是電子資料交換。

加密

一種運算程序，可將人類可讀取的純文字資料轉換為加密文字。

加密金鑰

由加密演算法產生的隨機位元的加密字串。金鑰長度可能有所不同，每個金鑰的設計都是不可預測且唯一的。

端序

位元組在電腦記憶體中的儲存順序。大端序系統首先儲存最高有效位元組。小端序系統首先儲存最低有效位元組。

端點

請參閱 服務端點。

端點服務

您可以在虛擬私有雲端 (VPC) 中託管以與其他使用者共用的服務。您可以使用 建立端點服務，AWS PrivateLink 並將許可授予其他 AWS 帳戶 或 AWS Identity and Access Management (IAM) 委託人。這些帳戶或主體可以透過建立介面 VPC 端點私下連接至您的端點服務。如需詳細資訊，請參閱 Amazon Virtual Private Cloud (Amazon VPC) 文件中的建立端點服務。

企業資源規劃 (ERP)

一種系統，可自動化和管理企業的關鍵業務流程 （例如會計、MES 和專案管理）。

信封加密

使用另一個加密金鑰對某個加密金鑰進行加密的程序。如需詳細資訊，請參閱 AWS Key Management Service (AWS KMS) 文件中的信封加密。

環境

執行中應用程式的執行個體。以下是雲端運算中常見的環境類型：

- 開發環境 – 執行中應用程式的執行個體，只有負責維護應用程式的核心團隊才能使用。開發環境用來測試變更，然後再將開發環境提升到較高的環境。此類型的環境有時稱為測試環境。
- 較低的環境 – 應用程式的所有開發環境，例如用於初始建置和測試的開發環境。
- 生產環境 – 最終使用者可以存取的執行中應用程式的執行個體。在 CI/CD 管道中，生產環境是最後一個部署環境。
- 較高的環境 – 核心開發團隊以外的使用者可存取的所有環境。這可能包括生產環境、生產前環境以及用於使用者接受度測試的環境。

epic

在敏捷方法中，有助於組織工作並排定工作優先順序的功能類別。epic 提供要求和實作任務的高層級描述。例如， AWS CAF 安全概念包括身分和存取管理、偵測控制、基礎設施安全、資料保護和事件回應。如需有關 AWS 遷移策略中的 Epic 的詳細資訊，請參閱計畫實作指南。

ERP

請參閱企業資源規劃。

探索性資料分析 (EDA)

分析資料集以了解其主要特性的過程。您收集或彙總資料，然後執行初步調查以尋找模式、偵測異常並檢查假設。透過計算摘要統計並建立資料可視化來執行 EDA。

F

事實資料表

星狀結構描述中的中央資料表。它存放有關業務操作的量化資料。一般而言，事實資料表包含兩種類型的資料欄：包含度量的資料，以及包含維度資料表外部索引鍵的資料欄。

快速失敗

一種使用頻繁和增量測試來縮短開發生命週期的理念。這是敏捷方法的關鍵部分。

故障隔離界限

在 中 AWS 雲端，像是可用區域 AWS 區域、控制平面或資料平面等界限會限制故障的影響，並有助於改善工作負載的彈性。如需詳細資訊，請參閱AWS 故障隔離界限。

功能分支

請參閱分支。

特徵

用來進行預測的輸入資料。例如，在製造環境中，特徵可能是定期從製造生產線擷取的影像。

功能重要性

特徵對於模型的預測有多重要。這通常表示為可以透過各種技術來計算的數值得分，例如 Shapley Additive Explanations (SHAP) 和積分梯度。如需詳細資訊，請參閱 的機器學習模型可解譯性 AWS。

特徵轉換

優化 ML 程序的資料，包括使用其他來源豐富資料、調整值、或從單一資料欄位擷取多組資訊。這可讓 ML 模型從資料中受益。例如，如果將「2021-05-27 00:15:37」日期劃分為「2021」、「五月」、「週四」和「15」，則可以協助學習演算法學習與不同資料元件相關聯的細微模式。

少量擷取提示

在要求 LLM 執行類似的任務之前，提供少量示範任務和所需輸出的範例。此技術是內容內學習的應用程式，其中模型會從內嵌在提示中的範例 (快照) 中學習。少量的提示對於需要特定格式、推理或網域知識的任務來說非常有效。另請參閱零鏡頭提示。

FGAC

請參閱精細存取控制。

精細存取控制 (FGAC)

使用多個條件來允許或拒絕存取請求。

閃切遷移

一種資料庫遷移方法，透過變更資料擷取使用連續資料複寫，以盡可能在最短的時間內遷移資料，而不是使用分階段方法。目標是將停機時間降至最低。

FM

請參閱基礎模型。

基礎模型 (FM)

大型深度學習神經網路，已針對廣義和未標記資料的大量資料集進行訓練。FMs 能夠執行各種一般任務，例如了解語言、產生文字和影像，以及以自然語言交談。如需詳細資訊，請參閱什麼是基礎模型。

FM 閘道

集中式中介，可控制和標準化對基礎模型的存取。也稱為 LLM 閘道。

G

生成式 AI

已針對大量資料進行訓練的 AI 模型子集，可使用簡單的文字提示建立新的內容和成品，例如影像、影片、文字和音訊。如需詳細資訊，請參閱什麼是生成式 AI。

地理封鎖

請參閱地理限制。

地理限制 (地理封鎖)

Amazon CloudFront 中的選項，可防止特定國家/地區的使用者存取內容分發。您可以使用允許清單或封鎖清單來指定核准和禁止的國家/地區。如需詳細資訊，請參閱 CloudFront 文件中的限制內容的地理分佈。

Gitflow 工作流程

這是一種方法，其中較低和較高環境在原始碼儲存庫中使用不同分支。Gitflow 工作流程被視為舊版，而以幹線為基礎的工作流程是現代、偏好的方法。

黃金影像

系統或軟體的快照，做為部署該系統或軟體新執行個體的範本。例如，在製造中，黃金映像可用於在多個裝置上佈建軟體，並有助於提高裝置製造操作的速度、可擴展性和生產力。

綠地策略

新環境中缺乏現有基礎設施。對系統架構採用綠地策略時，可以選擇所有新技術，而不會限制與現有基礎設施的相容性，也稱為棕地。如果正在擴展現有基礎設施，則可能會混合棕地和綠地策略。

防護機制

有助於跨組織單位 (OU) 來管控資源、政策和合規的高層級規則。預防性防護機制會強制執行政策，以確保符合合規標準。透過使用服務控制政策和 IAM 許可界限來將其實作。偵測性防護機制可

偵測政策違規和合規問題，並產生提醒以便修正。它們是透過使用 AWS Config AWS Security Hub CSPM、Amazon GuardDuty、、Amazon Inspector AWS Trusted Advisor和自訂 AWS Lambda 檢查來實作。

護欄 (AI)

安全機制可篩選、驗證和限制代理程式輸入和輸出，以協助確保負責任且安全的 AI 行為。

H

HA

請參閱高可用性。

異質資料庫遷移

將來源資料庫遷移至使用不同資料庫引擎的目標資料庫 (例如，Oracle 至 Amazon Aurora)。異質遷移通常是重新架構工作的一部分，而轉換結構描述可能是一項複雜任務。AWS 提供有助於結構描述轉換的 AWS SCT。

高可用性 (HA)

在遇到挑戰或災難時，工作負載能夠在不介入的情況下持續運作。HA 系統的設計目的是自動容錯移轉、持續提供高品質的效能，並處理不同的負載和故障，並將效能影響降至最低。

歷史現代化

一種方法，用於現代化和升級操作技術 (OT) 系統，以更好地滿足製造業的需求。歷史資料是一種資料庫，用於從工廠中的各種來源收集和存放資料。

保留資料

從用於訓練機器學習模型的資料集中保留的部分歷史標記資料。您可以使用保留資料，透過比較模型預測與保留資料來評估模型效能。

human-in-the-loop (HitL)

一種工作流程模式，其中代理程式執行會在關鍵決策點暫停進行人工審核和核准。

異質資料庫遷移

將您的來源資料庫遷移至共用相同資料庫引擎的目標資料庫 (例如，Microsoft SQL Server 至 Amazon RDS for SQL Server)。同質遷移通常是主機轉換或平台轉換工作的一部分。您可以使用原生資料庫公用程式來遷移結構描述。

熱資料

經常存取的資料，例如即時資料或最近的轉譯資料。此資料通常需要高效能儲存層或類別，才能提供快速的查詢回應。

修補程序

緊急修正生產環境中的關鍵問題。由於其緊迫性，通常會在典型 DevOps 發行工作流程之外執行修補程式。

超級護理期間

在切換後，遷移團隊在雲端管理和監控遷移的應用程式以解決任何問題的時段。通常，此期間的長度為 1-4 天。在超級護理期間結束時，遷移團隊通常會將應用程式的責任轉移給雲端營運團隊。

I

IaC

將基礎設施視為程式碼。

身分型政策

連接至一或多個 IAM 主體的政策，可定義其在 AWS 雲端 環境中的許可。

閒置應用程式

90 天期間 CPU 和記憶體平均使用率在 5% 至 20% 之間的應用程式。在遷移專案中，通常會淘汰這些應用程式或將其保留在內部部署。

IIoT

請參閱工業物聯網。

不可變的基礎設施

為生產工作負載部署新基礎設施的模型，而不是更新、修補或修改現有的基礎設施。不可變基礎設施本質上比可變基礎設施更一致、可靠且可預測。如需詳細資訊，請參閱 AWS Well-Architected Framework 中的使用不可變基礎設施的部署最佳實務。

傳入 (輸入) VPC

在 AWS 多帳戶架構中，接受、檢查和路由來自應用程式外部之網路連線的 VPC。AWS 安全參考架構建議您使用傳入、傳出和檢查 VPC 來設定網路帳戶，以保護應用程式與更廣泛的網際網路之間的雙向介面。

增量遷移

一種切換策略，您可以在其中將應用程式分成小部分遷移，而不是執行單一、完整的切換。例如，您最初可能只將一些微服務或使用者移至新系統。確認所有項目都正常運作之後，您可以逐步移動其他微服務或使用者，直到可以解除委任舊式系統。此策略可降低與大型遷移關聯的風險。

工業 4.0

2016 年 Klaus Schwab 推出的術語，透過連線能力、即時資料、自動化、分析和 AI/ML 的進展，指製造程序的現代化。

基礎設施

應用程式環境中包含的所有資源和資產。

基礎設施即程式碼 (IaC)

透過一組組態檔案來佈建和管理應用程式基礎設施的程序。IaC 旨在協助您集中管理基礎設施，標準化資源並快速擴展，以便新環境可重複、可靠且一致。

工業物聯網 (IIoT)

在製造業、能源、汽車、醫療保健、生命科學和農業等產業領域使用網際網路連線的感測器和裝置。如需詳細資訊，請參閱建立工業物聯網 (IIoT) 數位轉型策略。

檢查 VPC

在 AWS 多帳戶架構中，集中式 VPC，可管理 VPCs 之間 （在相同或不同的 中 AWS 區域)、網際網路和內部部署網路之間的網路流量檢查。AWS 安全參考架構建議您使用傳入、傳出和檢查 VPC 來設定網路帳戶，以保護應用程式與更廣泛的網際網路之間的雙向介面。

物聯網 (IoT)

具有內嵌式感測器或處理器的相連實體物體網路，其透過網際網路或本地通訊網路與其他裝置和系統進行通訊。如需詳細資訊，請參閱什麼是 IoT？

可解釋性

機器學習模型的一個特徵，描述了人類能夠理解模型的預測如何依賴於其輸入的程度。如需詳細資訊，請參閱 的機器學習模型可解譯性 AWS。

IoT

請參閱物聯網。

IT 資訊庫 (ITIL)

一組用於交付 IT 服務並使這些服務與業務需求保持一致的最佳實務。ITIL 為 ITSM 提供了基礎。

IT 服務管理 (ITSM)

與組織的設計、實作、管理和支援 IT 服務關聯的活動。如需有關將雲端操作與 ITSM 工具整合的資訊，請參閱操作整合指南。

ITIL

請參閱 IT 資訊庫。

ITSM

請參閱 IT 服務管理。

L

標籤型存取控制 (LBAC)

強制存取控制 (MAC) 的實作，其中使用者和資料本身都會獲得明確指派的安全標籤值。使用者安全標籤和資料安全標籤之間的交集會決定使用者可以看到哪些資料列和資料欄。

登陸區域

登陸區域是架構良好的多帳戶 AWS 環境，可擴展且安全。這是一個起點，您的組織可以從此起點快速啟動和部署工作負載與應用程式，並對其安全和基礎設施環境充滿信心。如需有關登陸區域的詳細資訊，請參閱設定安全且可擴展的多帳戶 AWS 環境。

大型語言模型 (LLM)

預先訓練大量資料的深度學習 AI 模型。LLM 可以執行多個任務，例如回答問題、摘要文件、將文字翻譯成其他語言，以及完成句子。如需詳細資訊，請參閱什麼是 LLMs。

大型遷移

遷移 300 部或更多伺服器。

LBAC

請參閱標籤型存取控制。

最低權限

授予執行任務所需之最低許可的安全最佳實務。如需詳細資訊，請參閱 IAM 文件中的套用最低權限許可。

隨即轉移

請參閱 7 個 R。

小端序系統

首先儲存最低有效位元組的系統。另請參閱 Endianness。

LLM

請參閱大型語言模型。

較低的環境

請參閱 環境。

M

機器學習 (ML)

一種使用演算法和技術進行模式識別和學習的人工智慧。機器學習會進行分析並從記錄的資料 (例如物聯網 (IoT) 資料) 中學習，以根據模式產生統計模型。如需詳細資訊，請參閱機器學習。

主要分支

請參閱分支。

惡意軟體

旨在危及電腦安全或隱私權的軟體。惡意軟體可能會中斷電腦系統、洩露敏感資訊，或取得未經授權的存取。惡意軟體的範例包括病毒、蠕蟲、勒索軟體、特洛伊木馬程式、間諜軟體和鍵盤記錄器。

受管服務

AWS 服務 會 AWS 操作基礎設施層、作業系統和平台，而您會存取端點來存放和擷取資料。Amazon Simple Storage Service (Amazon S3) 和 Amazon DynamoDB 是受管服務的範例。這些也稱為抽象服務。

製造執行系統 (MES)

一種軟體系統，用於追蹤、監控、記錄和控制生產程序，將原物料轉換為現場成品。

MAP

請參閱遷移加速計劃。

MCP

請參閱模型內容通訊協定。

模型內容通訊協定 (MCP)

適用於代理程式對工具通訊的無狀態通訊協定。

MCP 伺服器

透過模型內容通訊協定公開一或多個工具的服務。

機制

建立工具、推動工具採用，然後檢查結果以進行調整的完整程序。機制是在操作時強化和改善自身的循環。如需詳細資訊，請參閱 AWS Well-Architected Framework 中的建置機制。

成員帳戶

除了屬於組織一部分的管理帳戶 AWS 帳戶 之外的所有 AWS Organizations。帳戶一次只能是一個組織的成員。

製造執行系統

請參閱製造執行系統。

訊息佇列遙測傳輸 (MQTT)

根據發佈/訂閱模式的輕量型machine-to-machine(M2M) 通訊協定，適用於資源受限的 IoT 裝置。

微服務

一種小型的獨立服務，它可透過定義明確的 API 進行通訊，通常由小型獨立團隊擁有。例如，保險系統可能包含對應至業務能力 (例如銷售或行銷) 或子領域 (例如購買、索賠或分析) 的微服務。微服務的優點包括靈活性、彈性擴展、輕鬆部署、可重複使用的程式碼和適應力。如需詳細資訊，請參閱使用無 AWS 伺服器服務整合微服務。

微服務架構

一種使用獨立元件來建置應用程式的方法，這些元件會以微服務形式執行每個應用程式程序。這些微服務會使用輕量型 API，透過明確定義的介面進行通訊。此架構中的每個微服務都可以進行更新、部署和擴展，以滿足應用程式特定功能的需求。如需詳細資訊，請參閱在 上實作微服務 AWS。

Migration Acceleration Program (MAP)

此 AWS 計畫提供諮詢支援、訓練和服務，以協助組織建立強大的營運基礎，以移至雲端，並協助抵銷遷移的初始成本。MAP 包括用於有條不紊地執行舊式遷移的遷移方法以及一組用於自動化和加速常見遷移案例的工具。

大規模遷移

將大部分應用程式組合依波次移至雲端的程序，在每個波次中，都會以更快的速度移動更多應用程式。此階段使用從早期階段學到的最佳實務和經驗教訓來實作團隊、工具和流程的遷移工廠，以透過自動化和敏捷交付簡化工作負載的遷移。這是 AWS 遷移策略的第三階段。

遷移工廠

可透過自動化、敏捷的方法簡化工作負載遷移的跨職能團隊。遷移工廠團隊通常包括營運、業務分析師和擁有者、遷移工程師、開發人員以及從事 Sprint 工作的 DevOps 專業人員。20% 至 50% 之間的企業應用程式組合包含可透過工廠方法優化的重複模式。如需詳細資訊，請參閱此內容集中的遷移工廠的討論和雲端遷移工廠指南。

遷移中繼資料

有關完成遷移所需的應用程式和伺服器的資訊。每種遷移模式都需要一組不同的遷移中繼資料。遷移中繼資料的範例包括目標子網路、安全群組和 AWS 帳戶。

遷移模式

可重複的遷移任務，詳細描述遷移策略、遷移目的地以及所使用的遷移應用程式或服務。範例：使用 AWS Application Migration Service 重新託管遷移至 Amazon EC2。

遷移組合評定 (MPA)

線上工具，提供驗證商業案例以遷移至 的資訊 AWS 雲端。MPA 提供詳細的組合評定 (伺服器適當規模、定價、總體擁有成本比較、遷移成本分析) 以及遷移規劃 (應用程式資料分析和資料收集、應用程式分組、遷移優先順序，以及波次規劃)。MPA 工具 （需要登入） 可供所有 AWS 顧問和 APN 合作夥伴顧問免費使用。

遷移準備程度評定 (MRA)

使用 AWS CAF 取得組織雲端整備狀態的洞見、識別優缺點，以及建立行動計劃以消除已識別差距的程序。如需詳細資訊，請參閱遷移準備程度指南。MRA 是 AWS 遷移策略的第一階段。

遷移策略

用來將工作負載遷移至 的方法 AWS 雲端。如需詳細資訊，請參閱本詞彙表中的 7 個 Rs 項目，並請參閱動員您的組織以加速大規模遷移。

機器學習 (ML)

請參閱機器學習。

現代化

將過時的 (舊版或單一) 應用程式及其基礎架構轉換為雲端中靈活、富有彈性且高度可用的系統，以降低成本、提高效率並充分利用創新。如需詳細資訊，請參閱《》中的現代化應用程式的策略 AWS 雲端。

現代化準備程度評定

這項評估可協助判斷組織應用程式的現代化準備程度；識別優點、風險和相依性；並確定組織能夠在多大程度上支援這些應用程式的未來狀態。評定的結果就是目標架構的藍圖、詳細說明現代化程序的開發階段和里程碑的路線圖、以及解決已發現的差距之行動計畫。如需詳細資訊，請參閱《》中的評估應用程式的現代化準備 AWS 雲端程度。

單一應用程式 (單一)

透過緊密結合的程序作為單一服務執行的應用程式。單一應用程式有幾個缺點。如果一個應用程式功能遇到需求激增，則必須擴展整個架構。當程式碼庫增長時，新增或改進單一應用程式的功能也會變得更加複雜。若要解決這些問題，可以使用微服務架構。如需詳細資訊，請參閱將單一體系分解為微服務。

MPA

請參閱遷移產品組合評估。

MQTT

請參閱訊息佇列遙測傳輸。

多類別分類

一個有助於產生多類別預測的過程 (預測兩個以上的結果之一)。例如，機器學習模型可能會詢問「此產品是書籍、汽車還是電話？」 或者「這個客戶對哪種產品類別最感興趣？」

可變基礎設施

更新和修改生產工作負載現有基礎設施的模型。為了提高一致性、可靠性和可預測性， AWS Well-Architected Framework 建議使用不可變基礎設施作為最佳實務。

O

OAC

請參閱原始存取控制。

OAI

請參閱原始存取身分。

OCM

請參閱組織變更管理。

離線遷移

一種遷移方法，可在遷移過程中刪除來源工作負載。此方法涉及延長停機時間，通常用於小型非關鍵工作負載。

OI

請參閱 操作整合。

OLA

請參閱操作層級協議。

線上遷移

一種遷移方法，無需離線即可將來源工作負載複製到目標系統。連接至工作負載的應用程式可在遷移期間繼續運作。此方法涉及零至最短停機時間，通常用於關鍵的生產工作負載。

OPC-UA

請參閱開放程序通訊 - 統一架構。

開放程序通訊 - 統一架構 (OPC-UA)

用於工業自動化的machine-to-machine(M2M) 通訊協定。OPC-UA 提供資料加密、身分驗證和授權機制的互通性標準。

操作水準協議 (OLA)

一份協議，闡明 IT 職能群組承諾向彼此提供的內容，以支援服務水準協議 (SLA)。

操作整備審查 (ORR)

問題和相關最佳實務的檢查清單，可協助您了解、評估、預防或減少事件和可能失敗的範圍。如需詳細資訊，請參閱 AWS Well-Architected Framework 中的操作準備審查 (ORR)。

操作技術 (OT)

使用實體環境控制工業操作、設備和基礎設施的硬體和軟體系統。在製造中，整合 OT 和資訊技術 (IT) 系統是工業 4.0 轉型的關鍵重點。

操作整合 (OI)

在雲端中將操作現代化的程序，其中包括準備程度規劃、自動化和整合。如需詳細資訊，請參閱操作整合指南。

組織追蹤

由 建立的線索 AWS CloudTrail 會記錄 AWS 帳戶 組織中所有 的所有事件 AWS Organizations。在屬於組織的每個 AWS 帳戶 中建立此追蹤，它會跟蹤每個帳戶中的活動。如需詳細資訊，請參閱 CloudTrail 文件中的建立組織追蹤。

組織變更管理 (OCM)

用於從人員、文化和領導力層面管理重大、顛覆性業務轉型的架構。OCM 透過加速變更採用、解決過渡問題，以及推動文化和組織變更，協助組織為新系統和策略做好準備，並轉移至新系統和策略。在 AWS 遷移策略中，此架構稱為人員加速，因為雲端採用專案所需的變更速度。如需詳細資訊，請參閱 OCM 指南。

原始存取控制 (OAC)

CloudFront 中的增強型選項，用於限制存取以保護 Amazon Simple Storage Service (Amazon S3) 內容。OAC 支援所有 S3 儲存貯體中的所有伺服器端加密 AWS KMS (SSE-KMS) AWS 區域，以及對 S3 儲存貯體的動態PUT和DELETE請求。

原始存取身分 (OAI)

CloudFront 中的一個選項，用於限制存取以保護 Amazon S3 內容。當您使用 OAI 時，CloudFront 會建立一個可供 Amazon S3 進行驗證的主體。經驗證的主體只能透過特定 CloudFront 分發來存取 S3 儲存貯體中的內容。另請參閱 OAC，它可提供更精細且增強的存取控制。

ORR

請參閱操作整備審核。

OT

請參閱操作技術。

傳出 (輸出) VPC

在 AWS 多帳戶架構中，處理從應用程式內啟動之網路連線的 VPC。AWS 安全參考架構建議您使用傳入、傳出和檢查 VPC 來設定網路帳戶，以保護應用程式與更廣泛的網際網路之間的雙向介面。

P

許可界限

附接至 IAM 主體的 IAM 管理政策，可設定使用者或角色擁有的最大許可。如需詳細資訊，請參閱 IAM 文件中的許可界限。

個人身分識別資訊 (PII)

直接檢視或與其他相關資料配對時，可用來合理推斷個人身分的資訊。PII 的範例包括名稱、地址和聯絡資訊。

PII

請參閱個人身分識別資訊。

手冊

一組預先定義的步驟，可擷取與遷移關聯的工作，例如在雲端中提供核心操作功能。手冊可以採用指令碼、自動化執行手冊或操作現代化環境所需的程序或步驟摘要的形式。

PLC

請參閱可程式設計邏輯控制器。

PLM

請參閱產品生命週期管理。

政策

可定義許可的物件（請參閱身分型政策）、指定存取條件（請參閱資源型政策），或定義組織中所有帳戶的最大許可 AWS Organizations（請參閱服務控制政策）。

混合持久性

根據資料存取模式和其他需求，獨立選擇微服務的資料儲存技術。如果您的微服務具有相同的資料儲存技術，則其可能會遇到實作挑戰或效能不佳。如果微服務使用最適合其需求的資料儲存，則可以更輕鬆地實作並達到更好的效能和可擴展性。

組合評定

探索、分析應用程式組合並排定其優先順序以規劃遷移的程序。如需詳細資訊，請參閱評估遷移準備程度。

述詞

傳回 `true`或 的查詢條件`false`，通常位於 `WHERE` 子句中。

述詞下推

一種資料庫查詢最佳化技術，可在傳輸前篩選查詢中的資料。這可減少必須從關聯式資料庫擷取和處理的資料量，並改善查詢效能。

預防性控制

旨在防止事件發生的安全控制。這些控制是第一道防線，可協助防止對網路的未經授權存取或不必要變更。如需詳細資訊，請參閱在 AWS上實作安全控制中的預防性控制。

委託人

中可執行動作和存取資源 AWS 的實體。此實體通常是 AWS 帳戶、IAM 角色或使用者的根使用者。如需詳細資訊，請參閱 IAM 文件中角色術語和概念中的主體。

設計隱私權

透過整個開發程序將隱私權納入考量的系統工程方法。

私有託管區域

一種容器，它包含有關您希望 Amazon Route 53 如何回應一個或多個 VPC 內的域及其子域之 DNS 查詢的資訊。如需詳細資訊，請參閱 Route 53 文件中的使用私有託管區域。

主動控制

旨在防止部署不合規資源的安全控制。這些控制項會在佈建資源之前對其進行掃描。如果資源不符合控制項，則不會佈建。如需詳細資訊，請參閱 AWS Control Tower 文件中的控制項參考指南，並參閱實作安全控制項中的主動控制項。 AWS

產品生命週期管理 (PLM)

管理產品整個生命週期的資料和程序，從設計、開發和啟動，到成長和成熟，再到拒絕和移除。

生產環境

請參閱 環境。

可程式設計邏輯控制器 (PLC)

在製造中，高度可靠、可調整的電腦，可監控機器並自動化製造程序。

提示鏈結

使用一個 LLM 提示的輸出做為下一個提示的輸入，以產生更好的回應。此技術用於將複雜任務分解為子任務，或反覆精簡或展開初步回應。它有助於提高模型回應的準確性和相關性，並允許更精細、個人化的結果。

擬匿名化

以預留位置值取代資料集中個人識別符的程序。假名化有助於保護個人隱私權。假名化資料仍被視為個人資料。

發佈/訂閱 (pub/sub)

一種模式，可啟用微服務之間的非同步通訊，以提高可擴展性和回應能力。例如，在微服務型 MES 中，微服務可以將事件訊息發佈到其他微服務可訂閱的頻道。系統可以新增新的微服務，而無需變更發佈服務。

Q

查詢計劃

一系列步驟，如指示，用於存取 SQL 關聯式資料庫系統中的資料。

查詢計劃迴歸

在資料庫服務優化工具選擇的計畫比對資料庫環境進行指定的變更之前的計畫不太理想時。這可能因為對統計資料、限制條件、環境設定、查詢參數繫結的變更以及資料庫引擎的更新所導致。

R

RACI 矩陣

請參閱負責、負責、諮詢、告知 (RACI)。

RAG

請參閱擷取增強生成。

勒索軟體

一種惡意軟體，旨在阻止對計算機系統或資料的存取，直到付款為止。

RASCI 矩陣

請參閱負責、負責、諮詢、告知 (RACI)。

RCAC

請參閱資料列和資料欄存取控制。

僅供讀取複本

用於唯讀用途的資料庫複本。您可以將查詢路由至僅供讀取複本以減少主資料庫的負載。

重新架構師

請參閱 7 個 R。

復原點目標 (RPO)

自上次資料復原點以來可接受的時間上限。這會決定最後一個復原點與服務中斷之間可接受的資料遺失。

復原時間目標 (RTO)

服務中斷與服務還原之間的可接受延遲上限。

重構

請參閱 7 個 R。

區域

地理區域中的 AWS 資源集合。每個 AWS 區域 都獨立於其他 ，以提供容錯能力、穩定性和彈性。如需詳細資訊，請參閱指定 AWS 區域 您的帳戶可以使用哪些 。

迴歸

預測數值的 ML 技術。例如，為了解決「這房子會賣什麼價格？」的問題 ML 模型可以使用線性迴歸模型，根據已知的房屋事實 (例如，平方英尺) 來預測房屋的銷售價格。

重新託管

請參閱 7 個 R。

版本

在部署程序中，它是將變更提升至生產環境的動作。

重新放置

請參閱 7 個 R。

Replatform

請參閱 7 個 R。

回購

請參閱 7 個 R。

彈性

應用程式抵禦中斷或從中斷中復原的能力。在 中規劃彈性時，高可用性和災難復原是常見的考量 AWS 雲端。如需詳細資訊，請參閱AWS 雲端 彈性。

資源型政策

附接至資源的政策，例如 Amazon S3 儲存貯體、端點或加密金鑰。這種類型的政策會指定允許存取哪些主體、支援的動作以及必須滿足的任何其他條件。

負責者、當責者、事先諮詢者和事後告知者 (RACI) 矩陣

矩陣，定義所有參與遷移活動和雲端操作之各方的角色和責任。矩陣名稱衍生自矩陣中定義的責任類型：負責人 (R)、責任 (A)、諮詢 (C) 和知情 (I)。支援 (S) 類型為選用。如果您包含支援，則矩陣稱為 RASCI 矩陣，如果您排除它，則稱為 RACI 矩陣。

回應性控制

一種安全控制，旨在驅動不良事件或偏離安全基準的補救措施。如需詳細資訊，請參閱在 AWS上實作安全控制中的回應性控制。

保留

請參閱 7 個 R。

淘汰

請參閱 7 個 R。

檢索增強生成 (RAG)

一種生成式 AI 技術，其中 LLM 會在產生回應之前參考訓練資料來源以外的授權資料來源。例如，RAG 模型可能會對組織的知識庫或自訂資料執行語意搜尋。如需詳細資訊，請參閱什麼是 RAG。

輪換

定期更新秘密的程序，讓攻擊者更難存取登入資料。

資料列和資料欄存取控制 (RCAC)

使用已定義存取規則的基本、彈性 SQL 表達式。RCAC 包含資料列許可和資料欄遮罩。

RPO

請參閱復原點目標。

RTO

請參閱復原時間目標。

執行手冊

執行特定任務所需的一組手動或自動程序。這些通常是為了簡化重複性操作或錯誤率較高的程序而建置。

S

SAML 2.0

許多身分提供者 (IdP) 使用的開放標準。此功能會啟用聯合單一登入 (SSO)，讓使用者可以登入 AWS 管理主控台 或呼叫 AWS API 操作，而不必為您組織中的每個人在 IAM 中建立使用者。如需有關以 SAML 2.0 為基礎的聯合詳細資訊，請參閱 IAM 文件中的關於以 SAML 2.0 為基礎的聯合。

斯卡達

請參閱監督控制和資料擷取。

SCP

請參閱服務控制政策。

秘密

您以加密形式存放的 AWS Secrets Manager機密或限制資訊，例如密碼或使用者登入資料。它包含秘密值及其中繼資料。秘密值可以是二進位、單一字串或多個字串。如需詳細資訊，請參閱 Secrets Manager 秘密中的內容？ Secrets Manager 文件中的 。

設計安全性

透過整個開發程序將安全性納入考量的系統工程方法。

安全控制

一種技術或管理防護機制，它可預防、偵測或降低威脅行為者利用安全漏洞的能力。安全控制有四種主要類型：預防性、偵測性、回應性和主動性。

安全強化

減少受攻擊面以使其更能抵抗攻擊的過程。這可能包括一些動作，例如移除不再需要的資源、實作授予最低權限的安全最佳實務、或停用組態檔案中不必要的功能。

安全資訊與事件管理 (SIEM) 系統

結合安全資訊管理 (SIM) 和安全事件管理 (SEM) 系統的工具與服務。SIEM 系統會收集、監控和分析來自伺服器、網路、裝置和其他來源的資料，以偵測威脅和安全漏洞，並產生提醒。

安全回應自動化

預先定義和程式設計的動作，旨在自動回應或修復安全事件。這些自動化可做為偵測或回應式安全控制，協助您實作 AWS 安全最佳實務。自動化回應動作的範例包括修改 VPC 安全群組、修補 Amazon EC2 執行個體或輪換登入資料。

伺服器端加密

由接收資料的 AWS 服務 在其目的地加密資料。

服務控制政策 (SCP)

為 AWS Organizations中的組織的所有帳戶提供集中控制許可的政策。SCP 會定義防護機制或設定管理員可委派給使用者或角色的動作限制。您可以使用 SCP 作為允許清單或拒絕清單，以指定允許或禁止哪些服務或動作。如需詳細資訊，請參閱 AWS Organizations 文件中的服務控制政策。

服務端點

的進入點 URL AWS 服務。您可以使用端點，透過程式設計方式連接至目標服務。如需詳細資訊，請參閱 AWS 一般參考 中的 AWS 服務 端點。

服務水準協議 (SLA)

一份協議，闡明 IT 團隊承諾向客戶提供的服務，例如服務正常執行時間和效能。

服務層級指標 (SLI)

服務效能方面的測量，例如其錯誤率、可用性或輸送量。

服務層級目標 (SLO)

代表服務運作狀態的目標指標，由服務層級指標測量。

共同責任模式

描述您與 共同 AWS 承擔雲端安全與合規責任的模型。 AWS 負責雲端的安全，而 負責雲端的安全。如需詳細資訊，請參閱共同責任模式。

陰影 AI

在組織內受管管道之外建置或使用的未授權 AI 應用程式。

SIEM

請參閱安全資訊和事件管理系統。

單一故障點 (SPOF)

應用程式的單一關鍵元件故障，可能會中斷系統。

SLA

請參閱服務層級協議。

SLI

請參閱服務層級指標。

SLO

請參閱服務層級目標。

先拆分後播種模型

擴展和加速現代化專案的模式。定義新功能和產品版本時，核心團隊會進行拆分以建立新的產品團隊。這有助於擴展組織的能力和服務，提高開發人員生產力，並支援快速創新。如需詳細資訊，請參閱 中的階段式應用程式現代化方法 AWS 雲端。

SPOF

請參閱單一故障點。

星狀結構描述

使用一個大型事實資料表來存放交易或測量資料的資料庫組織結構，並使用一或多個較小的維度資料表來存放資料屬性。此結構旨在用於資料倉儲或商業智慧用途。

Strangler Fig 模式

一種現代化單一系統的方法，它會逐步重寫和取代系統功能，直到舊式系統停止使用為止。此模式源自無花果藤，它長成一棵馴化樹並最終戰勝且取代了其宿主。該模式由 Martin Fowler 引入，作為重寫單一系統時管理風險的方式。如需有關如何套用此模式的範例，請參閱使用容器和 Amazon API Gateway 逐步現代化舊版 Microsoft ASP.NET (ASMX) Web 服務。

子網

您 VPC 中的 IP 地址範圍。子網必須位於單一可用區域。

監控控制和資料擷取 (SCADA)

在製造中，使用硬體和軟體來監控實體資產和生產操作的系統。

對稱加密

使用相同金鑰來加密及解密資料的加密演算法。

合成測試

以模擬使用者互動的方式測試系統，以偵測潛在問題或監控效能。您可以使用 Amazon CloudWatch Synthetics 來建立這些測試。

系統提示

一種向 LLM 提供內容、指示或指導方針以指示其行為的技術。系統提示有助於設定內容，並建立與使用者互動的規則。

T

標籤

做為中繼資料以組織 AWS 資源的鍵值對。標籤可協助您管理、識別、組織、搜尋及篩選資源。如需詳細資訊，請參閱標記您的 AWS 資源。

目標變數

您嘗試在受監督的 ML 中預測的值。這也被稱為結果變數。例如，在製造設定中，目標變數可能是產品瑕疵。

任務清單

用於透過執行手冊追蹤進度的工具。任務清單包含執行手冊的概觀以及要完成的一般任務清單。對於每個一般任務，它包括所需的預估時間量、擁有者和進度。

測試環境

請參閱 環境。

訓練

為 ML 模型提供資料以供學習。訓練資料必須包含正確答案。學習演算法會在訓練資料中尋找將輸入資料屬性映射至目標的模式 (您想要預測的答案)。它會輸出擷取這些模式的 ML 模型。可以使用 ML 模型，來預測您不知道的目標新資料。

tool

代理程式可以叫用以在外部系統中執行操作的函數或 API。

傳輸閘道

可以用於互連 VPC 和內部部署網路的網路傳輸中樞。如需詳細資訊，請參閱 AWS Transit Gateway 文件中的什麼是傳輸閘道。

主幹型工作流程

這是一種方法，開發人員可在功能分支中本地建置和測試功能，然後將這些變更合併到主要分支中。然後，主要分支會依序建置到開發環境、生產前環境和生產環境中。

受信任的存取權

將許可授予您指定的服務，以代表您在組織中 AWS Organizations 及其帳戶中執行任務。受信任的服務會在需要該角色時，在每個帳戶中建立服務連結角色，以便為您執行管理工作。如需詳細資訊，請參閱 文件中的 AWS Organizations 搭配使用 AWS Organizations 與其他 AWS 服務。

調校

變更訓練程序的各個層面，以提高 ML 模型的準確性。例如，可以透過產生標籤集、新增標籤、然後在不同的設定下多次重複這些步驟來訓練 ML 模型，以優化模型。

雙比薩團隊

兩個比薩就能吃飽的小型 DevOps 團隊。雙披薩團隊規模可確保軟體開發中的最佳協作。

U

不確定性

這是一個概念，指的是不精確、不完整或未知的資訊，其可能會破壞預測性 ML 模型的可靠性。有兩種類型的不確定性：認知不確定性是由有限的、不完整的資料引起的，而隨機不確定性是由資料中固有的噪聲和隨機性引起的。

未區分的任務

也稱為繁重工作，這是建立和操作應用程式的必要工作，但不為最終使用者提供直接價值或提供競爭優勢。未區分任務的範例包括採購、維護和容量規劃。

較高的環境

請參閱 環境。

V

清空

一種資料庫維護操作，涉及增量更新後的清理工作，以回收儲存並提升效能。

版本控制

追蹤變更的程序和工具，例如儲存庫中原始程式碼的變更。

VPC 對等互連

兩個 VPC 之間的連線，可讓您使用私有 IP 地址路由流量。如需詳細資訊，請參閱 Amazon VPC 文件中的什麼是 VPC 對等互連。

漏洞

危害系統安全性的軟體或硬體瑕疵。

W

暖快取

包含經常存取的目前相關資料的緩衝快取。資料庫執行個體可以從緩衝快取讀取，這比從主記憶體或磁碟讀取更快。

暖資料

不常存取的資料。查詢這類資料時，通常可接受中等緩慢的查詢。

視窗函數

SQL 函數，對與目前記錄在某種程度上相關的資料列群組執行計算。視窗函數適用於處理任務，例如根據目前資料列的相對位置計算移動平均值或存取資料列的值。

工作負載

提供商業價值的資源和程式碼集合，例如面向客戶的應用程式或後端流程。

工作串流

遷移專案中負責一組特定任務的功能群組。每個工作串流都是獨立的，但支援專案中的其他工作串流。例如，組合工作串流負責排定應用程式、波次規劃和收集遷移中繼資料的優先順序。組合工作串流將這些資產交付至遷移工作串流，然後再遷移伺服器和應用程式。

WORM

請參閱寫入一次，讀取許多。

WQF

請參閱AWS 工作負載資格架構。

寫入一次，讀取許多 (WORM)

儲存模型，可一次性寫入資料，並防止刪除或修改資料。授權使用者可以視需要多次讀取資料，但無法變更資料。此資料儲存基礎設施被視為不可變。

Z

零時差入侵

利用零時差漏洞的攻擊，通常是惡意軟體。

零時差漏洞

生產系統中未緩解的缺陷或漏洞。威脅行為者可以使用這種類型的漏洞來攻擊系統。開發人員經常因為攻擊而意識到漏洞。

零鏡頭提示

提供 LLM 執行任務的指示，但沒有可協助引導任務的範例 (快照)。LLM 必須使用其預先訓練的知識來處理任務。零鏡頭提示的有效性取決於任務的複雜性和提示的品質。另請參閱少量擷取提示。

殭屍應用程式

CPU 和記憶體平均使用率低於 5% 的應用程式。在遷移專案中，通常會淘汰這些應用程式。

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。