Amazon EKS クラスターに適した GitOps ツールの選択

AWS 規範ガイダンス

AWS 規範ガイダンス: Amazon EKS クラスターに適した GitOps ツールの選択

Table of Contents

Amazon EKS クラスターに適した GitOps ツールの選択

Pradip Kumar Pandey と Pratap Kumar Nanda、Amazon Web Services (AWS)

2025 年 4 月 (ドキュメント履歴)

クラウドネイティブテクノロジーが急速に進化する中で、GitOps はアプリケーションとインフラストラクチャを管理およびデプロイするための強力な方法論として浮上しています。Amazon Elastic Kubernetes Service (Amazon EKS) を使用している場合、GitOps の原則を実装すると、デプロイプロセスが大幅に強化され、信頼性が向上し、運用が合理化されます。さまざまな GitOps ツールが利用可能であり、EKS クラスターに適したツールを選択することは、チームの効率と DevOps プラクティスの全体的な成功に影響を与える重要な決定です。

Amazon EKS 環境に適した GitOps ツールを選択するには、特定の要件、チームの専門知識、スケーラビリティのニーズ、既存の との統合機能など、さまざまな要因を慎重に検討する必要がありますAWS のサービス。各ツールには独自の機能、長所、潜在的な制限があるため、組織の目標と運用状況に合わせて選択することが重要です。

このガイドでは、Amazon EKS 用の GitOps ツールの選択における重要な考慮事項について説明し、頻繁に使用されるオプションを比較し、情報に基づいた意思決定に役立つインサイトを提供します。9 つの一般的な GitOps ツールについて説明します。

- Argo CD
- Flux
- Weave GitOps
- Jenkins X
- GitLab CI/CD
- "
- Rancher フリート
- Codefresh
- Pulumi

ターゲットを絞ったビジネス成果

次のリストでは、開発プロセスと運用プロセスで GitOps 原則を実装するツールを選択する際の潜在的な目標と成果について説明します。

Amazon EKS とのシームレスな統合

GitOps ツールは Amazon EKS とスムーズに統合され、Amazon EKS 固有の機能や最適化との互換性を提供する必要があります。

- Amazon EKS のネイティブサポート: 簡単なクラスター接続と管理など、Amazon EKS の組み込みサポートを提供するツールを探します。
- AWS のサービス 統合: ツールが AWS Identity and Access Management (IAM)、Amazon Elastic Container Registry (Amazon ECR)、Amazon CloudWatch AWS のサービス などの他の とやり取りできることを確認します。
- Amazon EKS アドオンの互換性: ツールが Amazon EKS アドオンをサポートし、効果的に管理できることを確認します。

スケーラビリティとパフォーマンス

GitOps ツールは、小さなクラスターから大規模なマルチクラスター環境まで、Amazon EKS オペレーションのスケールを処理できる必要があります。

- リソース効率: ツールのリソース消費量とクラスターのパフォーマンスへの影響を評価します。
- 大規模なオペレーション: 多数のアプリケーションとクラスターを同時に管理するツールの能力を評価します。
- 負荷の高いパフォーマンス: 高頻度の更新や大規模なデプロイ中にツールがどのように機能するかを検討します。

セキュリティとコンプライアンス

セキュリティ機能とコンプライアンス機能は、特に規制対象業界や機密データを処理する場合に不可欠です。

- アクセスコントロール: IAM と統合された堅牢なロールベースのアクセスコントロール (RBAC) 機能を探します。
- シークレット管理: ツールが機密情報をどのように処理し、 AWS Secrets Managerまたは他の ソリューションと統合するかを評価します。
- 監査証跡: ツールがコンプライアンスとトラブルシューティングのための包括的な口グ記録と監査機能を提供していることを確認します。

- セキュリティスキャン: デプロイの脆弱性の組み込みセキュリティスキャンを提供するツールを検討してください。

使いやすさと学習曲線

このツールはユーザーフレンドリで、チームのスキルに合わせて、迅速な導入と効率的な使用を確保する必要があります。

- ユーザーインターフェイス: コマンドラインインターフェイス (CLI) とグラフィカルユーザーインターフェイス (GUI) の両方の機能の直感性を評価します。
- ドキュメントの品質: 包括的でup-to-dateドキュメントとチュートリアルを探します。
- 学習リソース: トレーニング資料、コース、コミュニティリソースの可用性を検討します。

コミュニティとネットワークのサポート

強力なコミュニティとネットワークは、貴重なリソース、プラグイン、長期的な持続可能性を提供できます。

- アクティブな開発: 更新の頻度と保守者の応答性を確認します。
- コミュニティサイズ: サポートとナレッジ共有のために、ユーザーコミュニティのサイズとアクティビティを検討します。
- サードパーティー統合: プラグインの可用性とスタック内の他のツールとの統合を評価します。

マルチクラスター管理機能

複数の EKS クラスターがある場合は、それらを効率的に管理することが重要です。

- 一元管理: 1 つのコントロールプレーンから複数のクラスターを管理できる機能を探します。
- クラスターフェデレーション: マルチクラスターアプリケーションの Kubernetes フェデレーションをサポートするツールを検討してください。
- 環境パリティ: 開発、ステージング、本番稼働など、さまざまな環境でツールがどの程度整合性を維持しているかを評価します。

オブザーバビリティとモニタリング

このツールは、デプロイの状態とクラスターの状態に関する明確なインサイトを提供する必要があります。

- デプロイの可視性: デプロイのステータスと履歴を明確に把握できる機能を探します。
- モニタリングツールとの統合: このツールが Prometheus や Grafana などの一般的なモニタリングソリューションとどの程度統合されているかを検討します。
- アラート機能: デプロイの問題やドリフトに関するアラートを設定および管理するためのツールの能力を評価します。

柔軟性とカスタマイズ

特定のワークフローや要件にツールを適応させる機能は、長期的に満足するために重要です。

- 拡張性: ツールの機能を拡張できるプラグインアーキテクチャまたは APIs を探します。
- カスタムリソースのサポート: ツールがカスタム Kubernetes リソースを効果的に処理できることを確認します。
- ワークフローのカスタマイズ: GitOps ワークフローをチームのニーズにどれだけ簡単に調整できるかを評価します。

継続的な配信とプログレッシブデプロイのサポート

高度なデプロイ戦略は、リスクを最小限に抑え、スムーズな更新を保証するためにしばしば重要です。

- Canary デプロイ: Canary リリースの組み込みサポートを探します。
- ブルー/グリーンデプロイ: ブルー/グリーンデプロイ戦略のツールの機能を評価します。
- ロールバックメカニズム: デプロイの失敗から迅速に復旧できるように、堅牢でeasy-to-useロールバック機能を確保します。

コスト効率とリソース使用率

直接コストと間接コストの両方を含め、ツールの導入と保守の全体的なコストを考慮します。

- ライセンスコスト: オープンソースオプションを商用ソリューションと比較し、サポートとエンタープライズ機能を検討します。
- 運用上のオーバーヘッド: 管理とメンテナンスの観点から追加の運用コストを評価します。
- リソース消費: 必要なコンピューティングリソースとストレージリソースの観点からツールの効率を評価します。

これらの結果とその側面を慎重に検討することで、EKS クラスターに最適な GitOps ツールについて十分な情報に基づいた意思決定を行い、ツールが組織のニーズ、機能、長期戦略に沿っていることを確認できます。

EKS クラスター用の GitOps ツール

現在市販されている Kubernetes 用の GitOps ツールがいくつかあります。以下は、最も広く使用されているオプションのリストです。

- Argo CD
- Flux
- Weave GitOps
- Jenkins X
- GitLab CI/CD
- 孤立する
- ランチャーフリート
- Codefresh
- Pulumi

これらのツールが GitOps プラクティスを実装する方法の詳細については、リンクを参照してください。各ツールには長所とユースケースがあります。選択は、特定の要件、既存のインフラストラクチャ、チームの専門知識、必要な機能などの要因によって異なります。これらのツールは、組織のニーズと Kubernetes 環境の複雑さに基づいて評価することが重要です。

Argo CD

Argo CD は、Kubernetes 用の広く使用されている GitOps 継続的デリバリー (CD) ツールで、いくつかの主要な GitOps 原則に準拠しています。

GitOps のサポート

[面積]	ツール機能
宣言型設定	Argo CD は、Git リポジトリに保存されている宣言型設定を使用します。アプリケーションとインフラストラクチャの望ましい状態は、YAML ファイルで定義されます。これらの設定

[面積]	ツール機能
	は、デプロイ方法ではなく、デプロイする内容を記述します。
信頼できる単一のソースとしてのバージョン管理システム	Git リポジトリは、システム全体の唯一の信頼できるソースとして機能します。アプリケーションとインフラストラクチャに対するすべての変更は、Git を通じて行われます。これにより、完全な監査証跡と、以前の状態にロールバックできます。
自動同期	Argo CD は Git リポジトリの変更を継続的にモニタリングします。変更が検出されると、クラスターの実際の状態と Git で定義されている目的の状態が自動的に同期されます。これにより、クラスターにリポジトリで説明されている状態が常に反映されます。
Kubernetes ネイティブ	Argo CD は、Kubernetes 環境専用に設計されています。Kubernetes の宣言的な性質とカスタムリソースを活用してアプリケーションを管理します。
自己修復とドリフト検出	Argo CD は、クラスターのライブ状態と Git の希望する状態を定期的に比較します。ドリフト (実際の状態と目的の状態の相違) が検出されると、これらの不一致を自動的に修正できます。
マルチクラスターとマルチテナンシーのサポート	Argo CD は、1 つのインスタンスから複数の Kubernetes クラスターを管理できます。マルチテナンシーをサポートしているため、さまざまなチームがアプリケーションを個別に管理できます。

[面積]	ツール機能
アプリケーション定義	Argo CD のアプリケーションは、Application CRD (カスタムリソース定義) を使用して定義されます。これにより、デプロイする対象とその方法を定義する Kubernetes ネイティブな方法が可能になります。
デプロイとリリースの分離	Argo CD は、リリースからユーザーへのコードのデプロイを分離します。これは、ブルー/グリーンデプロイや Canary デプロイなどのさまざまなデプロイ戦略を通じて実現されます。
オブザーバビリティと監査可能性	Argo CD は、アプリケーションとクラスターの状態を監視するためのウェブ UI と CLI を提供します。変更とデプロイの明確な監査証跡を提供するために、すべてのアクションがログに記録されます。
セキュリティと RBAC	Argo CD は、Kubernetes ロールベースのアクセスコントロール (RBAC) と統合されています。認証と認可のためのシングルサインオン統合をサポートしています。
プラグ可能なアーキテクチャ	Argo CD は、さまざまなソース管理システム、Helm チャート、Kustomize、およびその他の Kubernetes マニフェスト形式をサポートしています。この柔軟性により、さまざまな環境やワークフローに対応できます。
継続的デリバリー (CD)	Argo CD は継続的デリバリーに焦点を当てていますが、継続的インテグレーション (CI) ツールと統合して、完全な CI/CD パイプラインを作成できます。

これらの GitOps の原則に従うことで、Argo CD は Kubernetes デプロイを管理するための堅牢でスケーラブル、安全な方法を提供します。これにより、システムの運用状態が Git リポジトリで定義さ

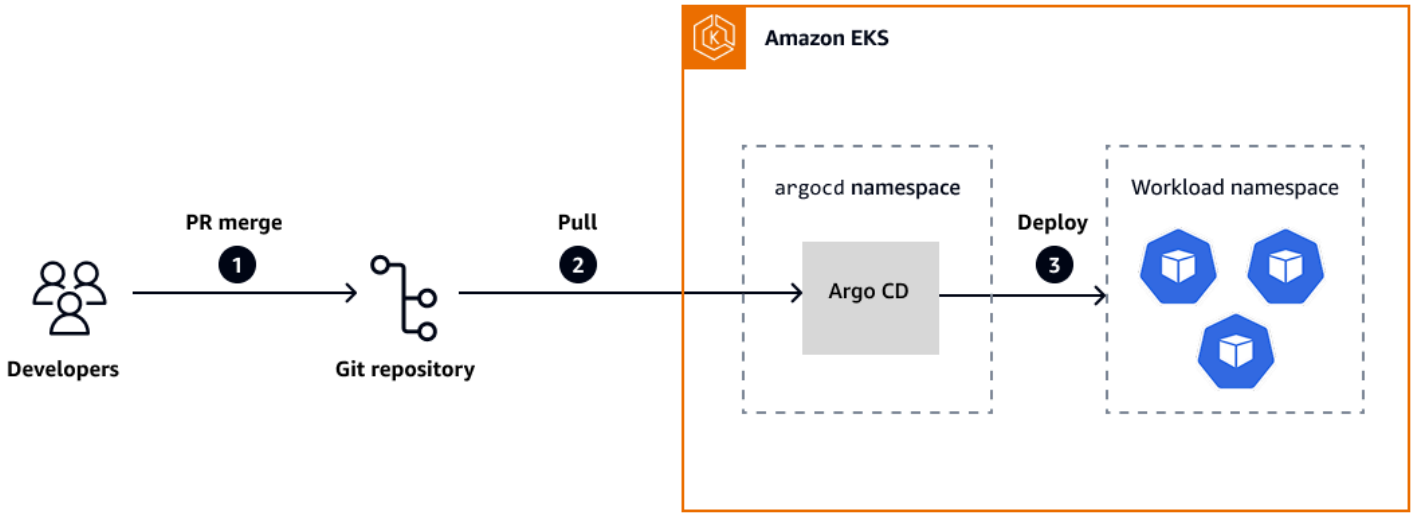
れている目的の状態と常に同期され、複雑な Kubernetes 環境での一貫性、信頼性、管理が容易になります。

Argo CD が対処できるシナリオと要件については、このガイドの後半にある「Argo CD のユースケース」を参照してください。Argo CD と Flux の比較については、このガイドの後半にある「特徴の比較」を参照してください。

詳細については、「Argo CD ドキュメント」を参照してください。

アーキテクチャ

次の図は、EKS クラスター内で Argo CD を使用する GitOps 駆動型 CD ワークフローを示しています。詳細については、「Argo CD ドキュメント」を参照してください。



各パラメータの意味は次のとおりです。

- ステップ 1: プルリクエスト (PR) マージ。開発者は、Git リポジトリに保存されている Kubernetes マニフェストまたは Helm チャートに変更をコミットします。PR がレビューされ、メインブランチにマージされると、アプリケーションの望ましい状態がソース管理で更新されます。
- ステップ 2: リポジトリ同期。Argo CD は EKS クラスター内の専用名前空間 (`argocd`) 内で実行され、設定された Git リポジトリを継続的にモニタリングします。変更を検出すると、最新の更新がプルされ、宣言された状態が調整されます。
- ステップ 3: ターゲット名前空間へのデプロイ。Argo CD は、Git から必要な状態をクラスター内のライブ状態と比較します。次に、ターゲットワークロード名前空間に必要な変更を適用して、それに応じてアプリケーションをデプロイまたは更新します。これには、デプロイ、サービス、ConfigMaps、シークレットなどの Kubernetes リソースを管理し、Git の信頼できるソースとのクラスターの一貫性を維持することが含まれます。

フラックス

Flux は、GitOps の原則を独自の方法で実装する Kubernetes 用のもう 1 つのツールです。

GitOps のサポート

[面積]	ツール機能
信頼できる唯一のソースとしての Git	Flux は Git リポジトリをシステムの目的の状態を定義するための最終的なソースとして使用します。アプリケーションとインフラストラクチャのすべての設定は Git に保存されます。
宣言型設定	Flux は、クラスターの目的の状態を宣言的に記述します。これらの説明は通常、Kubernetes マニフェスト、Helm チャート、または Kustomize オーバーレイです。
自動同期	Flux は Git リポジトリの変更を継続的にモニタリングします。変更を検出すると、自動的にクラスターに適用されます。
Kubernetes ネイティブ	Flux は、一連の Kubernetes コントローラーとカスタムリソースとして構築されています。Kubernetes の拡張機能メカニズムを使用して GitOps 機能を提供します。
プルベースのデプロイモデル	従来のプッシュベースの CI/CD システムとは異なり、Flux はプルベースのモデルを使用します。クラスターは、外部システムを使用して変更をプッシュするのではなく、Git から目的の状態を取得します。
継続的な調整	Flux は、クラスターの実際の状態を Git の望ましい状態と常に比較します。これらの状態間で検出されたドリフトは自動的に修正されます。
マルチテナンシー	Flux は、Kustomizations と HelmReleases の概念を通じてマルチテナンシーをサポートしま

[面積]	ツール機能
	す。異なるチームが設定の独自の部分を個別に管理できます。
プログレッシブ配信	Flux は、Flager コンポーネントを通じて、Canary リリースや A/B テストなどの高度なデプロイ戦略をサポートします。
Helm 統合	Flux には Helm のネイティブサポートが含まれているため、GitOps を通じて Helm リリースを簡単に管理できます。
イメージ更新の自動化	Flux は、コンテナレジストリで新しいバージョンが利用可能になると、Git のコンテナイメージを自動的に更新できます。
Kustomize サポート	Flux for Kustomize が提供するネイティブサポートを使用して、Kubernetes マニフェストをカスタマイズしてパッチを適用できます。
セキュリティと RBAC	Flux は、アクセスコントロールのために Kubernetes RBAC と統合されています。さまざまなバックエンドによるシークレット管理をサポートしています。
オブザーバビリティ	Flux は、調整とオペレーションに関するステータス情報とメトリクスを提供します。モニタリングツールと統合してオブザーバビリティを強化します。
イベント駆動型アーキテクチャ	Flux は、イベント駆動型のアプローチを使用して調整と更新を実装します。
拡張性	このツールは拡張可能なように設計されているため、カスタムコントローラーとリソースを追加できます。

[面積]	ツール機能
クラスター間の同期	Flux は、単一のリポジトリセットからの複数のクラスターの管理をサポートします。
依存関係管理	これにより、システムのさまざまな部分間の依存関係を定義し、オペレーションの正しい順序を確保できます。
Webhook レシーバー	Git プロバイダーや他のシステムからウェブフックを受信するように Flux を設定して、即時調整を開始できます。

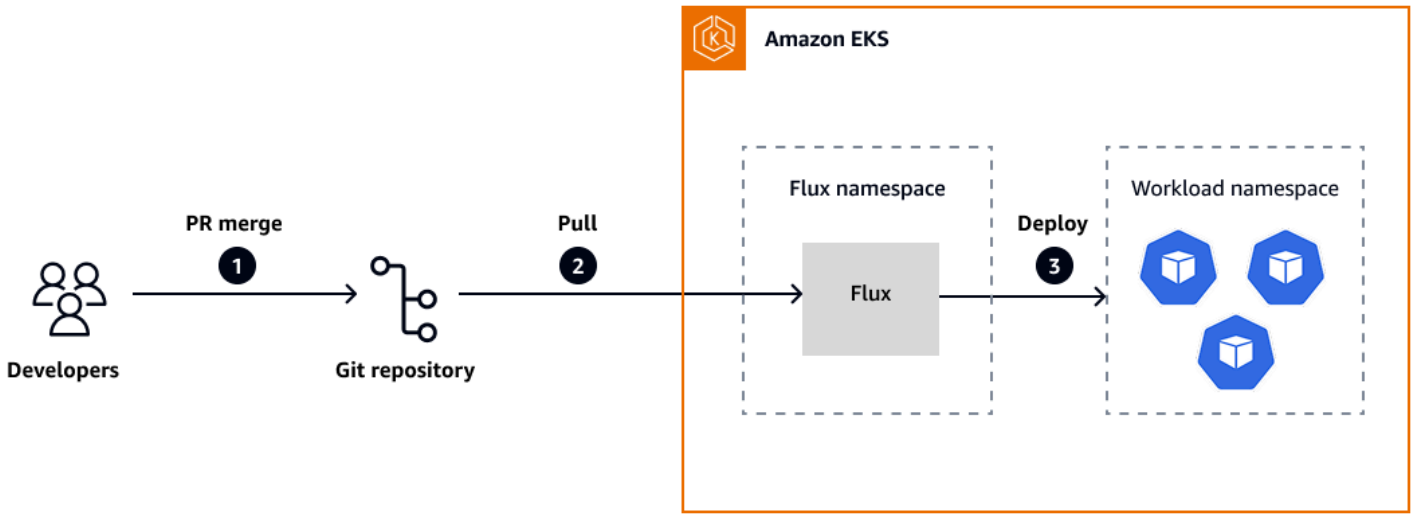
これらの GitOps 原則を実装することで、Flux は Kubernetes クラスターとアプリケーションを管理するための堅牢で柔軟なシステムを提供します。これにより、インフラストラクチャとアプリケーションが常に Git リポジトリと同期され、複雑な Kubernetes 環境で一貫性、信頼性、管理が容易になります。ツールの Kubernetes ネイティブアプローチと自動化に焦点を当てることで、特にクラウドネイティブ環境に適しています。

Flux が対処できるシナリオと要件については、このガイドの後半にある「Flux のユースケース」を参照してください。Argo CD と Flux の比較については、このガイドの後半にある「特徴の比較」を参照してください。

詳細については、Flux のドキュメントを参照してください。

アーキテクチャ

次の図は、EKS クラスター内で Flux を使用する GitOps 駆動型 CD ワークフローを示しています。詳細については、Flux のドキュメントを参照してください。



各パラメータの意味は次のとおりです。

- ステップ 1: プルリクエスト (PR) マージ。開発者は、Git リポジトリに保存されている Kubernetes マニフェストまたは Helm チャートに変更をコミットします。PR がレビューされ、メインブランチにマージされると、アプリケーションの望ましい状態がソース管理で更新されます。
- ステップ 2: リポジトリ同期。Flux は EKS クラスター内の専用名前空間内で実行され、設定された Git リポジトリを継続的にモニタリングします。変更を検出すると、最新の更新がプルされ、宣言された状態が調整されます。
- ステップ 3: ターゲット名前空間へのデプロイ。Flux は、Git からの目的の状態をクラスター内のライブ状態と比較します。次に、ターゲットワークロード名前空間に必要な変更を適用して、それに応じてアプリケーションをデプロイまたは更新します。

Weave GitOps

Weave GitOps は、GitOps という用語を導入した会社である Weaveworks によって開発されました。このツールは、GitOps の基本原則に基づく包括的な GitOps ソリューションを提供します。

GitOps のサポート

[面積]	ツール機能
信頼できる唯一のソースとしての Git	Weave GitOps は、システムの望ましい状態を定義するための信頼できるソースとして Git リポジトリを使用します。アプリケーションマ

[面積]	ツール機能
	ニフェスト、インフラストラクチャ定義、ポリシーを含むすべての設定は Git に保存されます。
宣言型設定	システムは、システム状態全体の宣言的な説明に依存します。これらの説明は通常、Kubernetes マニフェスト、Helm チャート、またはその他の宣言形式です。
自動同期	Weave GitOps は Git リポジトリの変更を継続的にモニタリングします。変更を検出すると、自動的にターゲット環境に適用されます。
Kubernetes ネイティブアーキテクチャ	Weave GitOps は、一連の Kubernetes コントローラーとカスタムリソースとして構築されています。Kubernetes の拡張機能メカニズムを使用して GitOps 機能を提供します。
継続的な調整	このツールは、クラスターの実際の状態を Git で定義されている目的の状態と常に比較します。これらの状態間で検出されたドリフトは自動的に修正されます。
マルチクラスター管理	Weave GitOps は、単一のコントロールプレーンからの複数の Kubernetes クラスターの管理をサポートします。これにより、さまざまな環境間で一貫したアプリケーションデプロイが可能になります。
コードとしてのポリシー	Weave GitOps には、セキュリティおよびコンプライアンスルールを適用するためのコードとしてポリシーの概念が組み込まれています。ポリシーは、アプリケーションコードとインフラストラクチャ定義とともにバージョン管理されます。

[面積]	ツール機能
プログレッシブ配信	このツールは、Canary リリースや Blue/Green デプロイなどの高度なデプロイ戦略をサポートします。これは、自動化されたプログレッシブデリバリーのために Flagger と統合されます。
オブザーバビリティとダッシュボード	Weave GitOps には、アプリケーションとクラスターの状態をモニタリングするためのダッシュボードが組み込まれています。調整プロセスとクラスターの状態に関するインサイトを提供します。
設計による保護	このツールは、RBAC 統合やシークレット管理などのセキュリティのベストプラクティスを実装します。さまざまな認証方法をサポートし、エンタープライズ ID プロバイダーと統合します。
拡張性と統合	このツールは、さまざまなクラウドネイティブツールで動作するように設計されています。Flux、Helm、Kustomize などの一般的なツールをサポートしています。
セルフサービス開発者プラットフォーム	Weave GitOps を使用すると、開発者向けのセルフサービスプラットフォームを作成できます。アプリケーションデプロイ用のテンプレートとガードレールを提供します。
GitOps オートメーション	このツールは、更新のプルリクエストの生成など、GitOps ワークフローの多くの側面を自動化します。
継続的デリバリーパイプライン	CI/CD システムと統合してend-to-endの配信パイプラインを作成します。

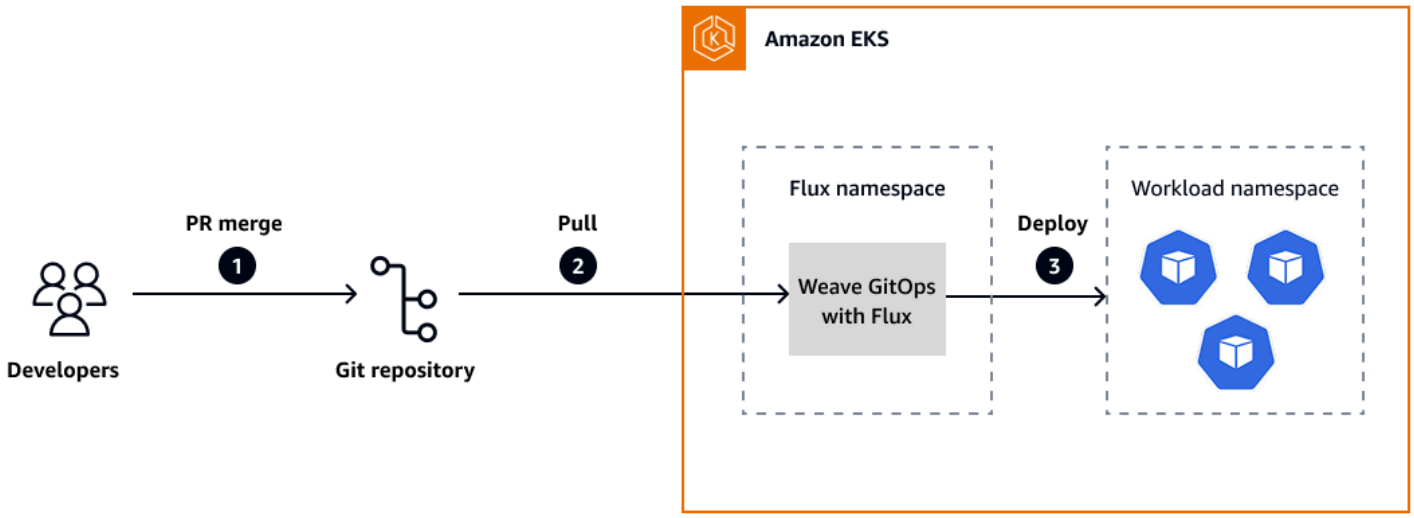
[面積]	ツール機能
監査とコンプライアンス	Weave DevOps は、すべての変更とアクションの完全な監査証跡を提供します。バージョン管理と自動プロセスを通じてコンプライアンス要件を満たすのに役立ちます。
スケーラビリティ	このツールは、小規模なプロジェクトから大規模なエンタープライズグレードのデプロイにスケールするように設計されています。
チームコラボレーション	Weave GitOps は、Git ベースのワークフローを通じて開発チームと運用チーム間のコラボレーションを容易にします。
サービスとしての GitOps	このツールはGitOps をマネージドサービスとして提供し、導入と管理を簡素化します。
ハイブリッドおよびマルチクラウドのサポート	Weave GitOps は、さまざまなクラウドプロバイダーやオンプレミス環境間で一貫した管理を可能にします。
継続的なセキュリティ	このツールは、デプロイプロセス全体でセキュリティスキャンとポリシーの適用を統合します。

Weave GitOps は、基本的なデプロイ自動化を超える包括的な GitOps ソリューションを提供するために、これらの原則を実装しています。これは、セキュリティ、スケーラビリティ、使いやすさに焦点を当てたクラウドネイティブアプリケーションの完全な運用モデルを作成することを目的としています。これらの GitOps の原則に従うことで、Weave GitOps は、組織が複数のクラスターやクラウドプロバイダーで Kubernetes 環境を一貫して、監査可能で、効率的に管理できるようにします。

詳細については、「Weave GitOps ドキュメント」を参照してください。

アーキテクチャ

次の図は、EKS クラスター内で Weave GitOps を使用する GitOps 駆動型 CD ワークフローを示しています。詳細については、「Weave GitOps リポジトリ」を参照してください。



各パラメータの意味は次のとおりです。

- ステップ 1: プルリクエスト (PR) マージ。開発者は、Git リポジトリに保存されている Kubernetes マニフェストまたは Helm チャートに変更をコミットします。PR がレビューされ、メインブランチにマージされると、アプリケーションの望ましい状態がソース管理で更新されます。
- ステップ 2: リポジトリ同期。Weave GitOps は EKS クラスターの Flux 名前空間内で実行され、設定された Git リポジトリを継続的にモニタリングします。変更が検出されると、最新の更新がプルされ、宣言された状態が調整されます。
- ステップ 3: ターゲット名前空間へのデプロイ。Weave GitOps は、Git からの目的の状態をクラスター内のライブ状態と比較します。次に、ターゲットワークロード名前空間に必要な変更を適用し、それに応じてアプリケーションをデプロイまたは更新します。

Jenkins X

Jenkins X は、Kubernetes 環境に GitOps の原則を実装するクラウドネイティブのオープンソース CI/CD プラットフォームです。Jenkins X は Argo CD や Flux などの GitOps ツールのみではありませんが、GitOps プラクティスをワークフローに組み込みます。

GitOps のサポート

[面積]	ツール機能
Git 中心のワークフロー	Jenkins X は、アプリケーションコードと設定の両方の信頼できる主要なソースとして Git リ

[面積]	ツール機能
	ポジトリを使用します。アプリケーションとインフラストラクチャに対するすべての変更は、Git を通じて行われます。
Environment as Code (EaC)	環境 (ステージングや本番稼働など) は、Git リポジトリでコードとして定義されます。これにより、環境設定のバージョン管理とレビューが可能になります。
自動 CI/CD パイプライン	Jenkins X は、プロジェクトの CI/CD パイプラインを自動的にセットアップします。これらのパイプラインはコード (パイプラインはコード) として定義され、Git に保存されます。
Kubernetes ネイティブ	Jenkins X は Kubernetes 環境専用に構築されています。Kubernetes リソースとカスタムリソース定義 (CRDsを使用します。
プレビュー環境	Jenkins X は、プルリクエストの一時環境を自動的に作成します。これにより、マージ前に変更を簡単にレビューおよびテストできます。
環境間の昇格	Jenkins X は GitOps アプローチを使用して環境間でアプリケーションを昇格させます (ステージングから本番稼働など) 。昇格は、プルリクエストを使用して適切なレビューと承認プロセスを確保することで処理されます。
Helm チャート管理	Jenkins X は Helm チャートを使用してアプリケーションをパッケージ化およびデプロイします。グラフは Git リポジトリでバージョン管理されています。
自動バージョニング	Jenkins X は、アプリケーションとリリースのバージョニングを自動的に管理します。セマンティックバージョニングを使用し、リリースノートを生成します。

[面積]	ツール機能
ChatOps 統合	Jenkins X は、一般的なオペレーションで ChatOps をサポートしています。これは、自動化とコラボレーションの GitOps 原則と一致しています。
拡張性	このツールは、機能を拡張するためのプラグインシステムを提供します。これにより、さまざまなクラウドネイティブツールとの統合が可能になります。
Infrastructure as code (IaC)	Jenkins X は、インフラストラクチャを定義および管理するための Terraform CloudFormation、 AWS Cloud Development Kit (AWS CDK)、およびその他の IaC ツールをサポートしています。インフラストラクチャ定義は、アプリケーションコードとともにバージョン管理されます。
自動ロールバック	Jenkins X は、デプロイ後に問題が検出された場合の自動ロールバックをサポートします。
シークレットの管理	このツールは外部シークレット管理ソリューションと統合され、機密情報を安全に処理します。
オブザーバビリティ	Jenkins X は、オブザーバビリティのためのモニタリングおよびログ記録ツールとの統合を提供します。
マルチクラウドのサポート	Jenkins X は、さまざまなクラウドプロバイダーやオンプレミス環境で動作するように設計されています。
チームコラボレーション	このツールは、Git ベースのワークフローとプルリクエストによるコラボレーションを促進します。

[面積]	ツール機能
継続的なフィードバック	このツールは、自動テストおよびプレビュー環境を通じて、変更に関する迅速なフィードバックを提供します。
DevOps のベストプラクティス	Jenkins X は、GitOps の原則を含む DevOps のベストプラクティスをデフォルトで実装します。 GitOps
宣言型設定	このツールは、宣言型設定を使用してアプリケーションと環境を定義します。
自動アップグレード	Jenkins X には、Jenkins X プラットフォーム自体のアップグレードを自動化するためのツールが用意されています。

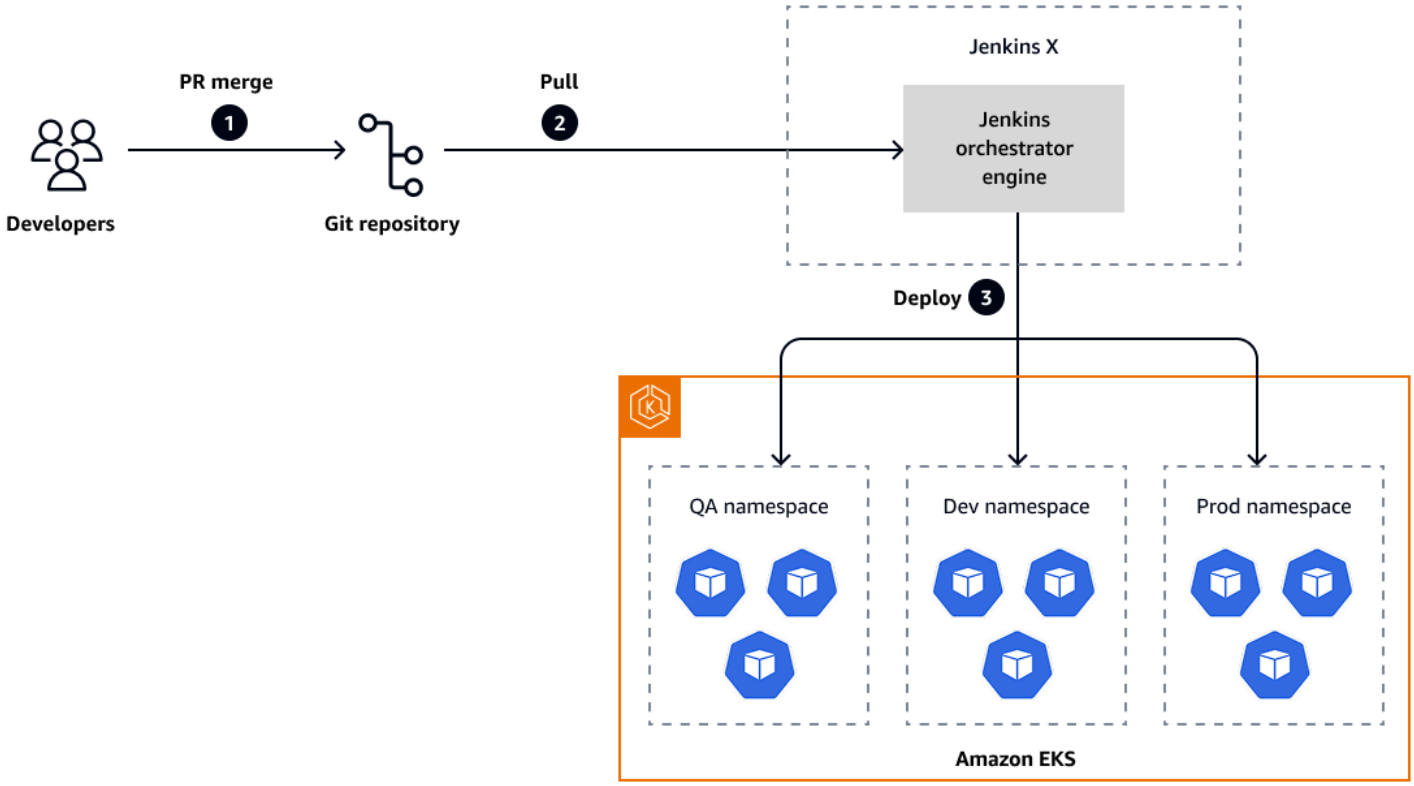
Jenkins X は、これらの GitOps 原則を実装して、Kubernetes 用の包括的な CI/CD ソリューションを作成します。GitOps プラクティスに従いながら、コードコミットから本番デプロイまで、ソフトウェア配信プロセス全体を自動化および合理化することを目指しています。これにより、チームはクラウドネイティブ環境でより速く、より信頼性が高く、より一貫性のあるデプロイを実現できます。

Jenkins X と Argo CD や Flux などのツールの主な違いは、Jenkins X がデプロイと環境管理に GitOps の原則を組み込みながら、ビルドの自動化やパイプライン管理など、より包括的な CI/CD ソリューションを提供することです。これにより、1 つの GitOps フレームワーク内で CI と CD の両方の側面をカバーするall-in-oneソリューションを必要とするチームに特に適しています。

詳細については、Jenkins X のドキュメントを参照してください。

アーキテクチャ

次の図は、Jenkins X を使用する GitOps 駆動型 CD ワークフローを示しています。詳細については、Jenkins X のドキュメントを参照してください。



各パラメータの意味は次のとおりです。

- ステップ 1: プルリクエスト (PR) マージ。開発者は、Git リポジトリに保存されている Kubernetes マニフェスト、Helm チャート、またはアプリケーションコードへの変更を含むプルリクエストを作成します。レビューと承認の後、PR はメインブランチにマージされ、ソースコントロールの目的の状態を更新します。
- ステップ 2: リポジトリ同期。Jenkins X は、変更を検出すると CI/CD パイプラインを自動的にトリガーします。パイプラインは、GitOps の原則を使用して、さまざまな環境 (ステージングや本番稼働など) を通じてアプリケーションを構築、テスト、昇格します。
- ステップ 3: ターゲット名前空間へのデプロイ。Jenkins X は、新しいアプリケーションバージョンで環境リポジトリ (ステージングと本番稼働) を更新します。クラスターは、Git から最新のマニフェストをプルし、アプリケーションを適切な名前空間にデプロイすることで、変更を自動的に調整します。

GitLab CI/CD

GitLab CI/CD は、GitLab プラットフォームの統合部分であり、継続的な統合、配信、デプロイ機能を提供します。GitLab CI/CD は GitOps ツールのみではありませんが、特に Kubernetes デプロイに使用する場合は、GitOps の原則を実装するように設定できます。

GitOps のサポート

[面積]	ツール機能
単一の情報源としての Git	GitLab CI/CD は Git リポジトリを使用して、アプリケーションコードとインフラストラクチャ設定の両方を保存します。システムへのすべての変更は Git を通じて行われ、完全な履歴と監査証跡が保証されます。
宣言型設定	GitLab CI/CD パイプラインは、Git リポジトリに保存されている宣言型設定である .gitlab-ci.yml ファイルで定義されます。Kubernetes マニフェスト、Helm チャート、またはその他の Infrastructure as Code (IaC) ファイルを同じリポジトリに保存して、インフラストラクチャの目的の状態を定義できます。
自動パイプライン	GitLab CI/CD は、変更がリポジトリにプッシュされると、パイプラインを自動的にトリガーします。これらのパイプラインには、アプリケーションの構築、テスト、デプロイのステージを含めることができます。
Kubernetes 統合	GitLab CI/CD はネイティブ Kubernetes 統合を提供し、Kubernetes クラスターへの GitOps スタイルのデプロイをサポートします。Git の設定に基づいて Kubernetes リソースを自動的に作成および管理できます。
環境管理	GitLab CI/CD は、コードとしての複数の環境 (ステージングや本番稼働など) の定義をサポー

[面積]	ツール機能
	トしています。これらの環境へのデプロイは自動化することも、GitOps のプラクティスに従って手動で承認する必要が生じることもあります。
アプリケーションの確認	GitLab は、他の GitOps ツールのプレビュー環境と同様に、マージリクエストの一時環境を自動的に作成できます。これにより、マージ前の変更を簡単に確認およびテストできます。
継続的デプロイ	GitLab CI/CD は、変更が特定のブランチにマージされたときに、Kubernetes クラスターに変更を自動的にデプロイするように設定できます。
IaC	GitLab CI/CD は、Terraform や などのツールとの統合をサポートし、インフラストラクチャ CloudFormation をコードとして管理します。インフラストラクチャ定義は、アプリケーションコードとともにバージョン管理できます。
オブザーバビリティとモニタリング	GitLab CI/CD には、Prometheus および Grafana との統合など、モニタリングおよびオブザーバビリティ機能が組み込まれています。
セキュリティスキャン	GitLab CI/CD には、GitOps ワークフローの一部としてセキュリティを適用するために CI/CD パイプラインに統合できる組み込みのセキュリティスキャンツールが含まれています。
コンテナレジストリ	GitLab CI/CD には、GitOps ワークフローでコンテナイメージ管理をシームレスに統合するための組み込みコンテナレジストリが含まれています。

[面積]	ツール機能
自動 DevOps	GitLab CI/CD の自動 DevOps 機能は、Kubernetes デプロイの GitOps 原則に従う CI/CD パイプラインを自動的に設定できます。
承認ワークフロー	GitLab CI/CD は、環境間で制御されたプロモーションを提供するデプロイの承認プロセスをサポートしています。
シークレットの管理	GitLab CI/CD には、CI/CD パイプライン内のシークレットを安全に管理および使用する機能が用意されています。
バージョニングとリリース	GitLab CI/CD は、CI/CD プロセスの一環として自動バージョニングとリリース管理をサポートしています。
ロールバック	GitLab CI/CD を使用すると、デプロイ後に問題が検出された場合、以前のバージョンに簡単にロールバックできます。
監査ログ	GitLab CI/CD は、GitOps のトレーサビリティの側面をサポートするために、すべてのアクションの包括的な監査ログを提供します。
マルチプロジェクトパイプライン	GitLab CI/CD は、複数のプロジェクトまたはリポジトリにまたがる複雑な GitOps ワークフローをサポートします。
ChatOps	GitLab CI/CD は ChatOps 統合をサポートしており、チャットインターフェイスを介してコラボレーションとオペレーションを提供します。
Kubernetes クラスター管理	GitLab CI/CD は、GitLab インターフェイスから直接 Kubernetes クラスターを管理する機能を提供します。

GitLab CI/CD は GitOps 専用に設計されているわけではありませんが、特に GitOps プラクティスを効果的に実装するために使用できます。 GitLab ソース管理、CI/CD、Kubernetes 管理を組み合わせた統合アプローチにより、GitOps ワークフローを実装するための強力なツールとなります。

GitLab CI/CD と Argo CD や Flux などの専用 GitOps ツールの主な違いは、GitLab が CI/CD 機能と共にソース管理、問題追跡、その他の開発ツールを含むより包括的なプラットフォームを提供することです。これにより、より広範な開発システム内で GitOps プラクティスを実装できるall-in-oneソリューションを必要とするチームに特に適しています。

GitLab CI/CD とそのアーキテクチャの詳細については、GitLab CI/CD ドキュメントを参照してください。

Spinnaker

GitOps ツールとしてのみ設計されているわけではありませんが、特にクラウドネイティブおよび Kubernetes デプロイに使用する場合は、GitOps の原則を実装するように設定できます。

GitOps のサポート

[面積]	ツール機能
宣言型設定	" は宣言型パイプライン定義を使用します。これは通常、JSON ファイルまたは YAML ファイルとして保存されます。これらのパイプライン定義は、GitOps プラクティスに従って Git リポジトリでバージョン管理できます。
IaC	" は、インフラストラクチャとデプロイ設定の定義をコードとしてサポートしています。これらの定義は Git リポジトリに保存でき、単一の信頼できるソースとして機能します。
マルチクラウドデプロイ	" は、複数のクラウドプロバイダーと Kubernetes クラスターで動作するように設計されています。これにより、さまざまな環境で一貫した GitOps プラクティスが可能になります。

[面積]	ツール機能
コードとしてのパイプライン	パイプラインはコードとして定義し、Git リポジトリに保存できます。これにより、バージョン管理とデプロイプロセスのレビューが可能になります。
自動デプロイ	Git リポジトリの変更に基づいてデプロイを自動的に開始するように " を設定できます。このツールは、GitOps の中心となる継続的なデプロイプラクティスをサポートしています。
イミュータブルなインフラストラクチャ	" は、GitOps の主要な概念であるイミュータブルインフラストラクチャの使用を促進します。既存のインスタンスを変更するのではなく、新しいインスタンスのデプロイを推奨します。
ロールバックとバージョニング	" は、堅牢なロールバック機能と、以前の既知の正常な状態への迅速な復帰を提供します。GitOps のトレーサビリティの原則に従って、デプロイのバージョニングをサポートします。
承認ワークフロー	「」には、環境間の制御されたプロモーションをサポートするために、パイプラインに手動による判断ステージが含まれています。これにより、デプロイとリリースを分離する GitOps プラクティスがサポートされます。
Canary と Blue/Green のデプロイ	" は、安全で制御されたリリースのための GitOps プラクティスに沿った高度なデプロイ戦略をサポートしています。
バージョン管理システムとの統合	" は、さまざまな Git プロバイダーと統合して、リポジトリイベントに基づいてパイプラインを開始できます。

[面積]	ツール機能
Kubernetes 統合	" は Kubernetes のネイティブサポートを提供し、Kubernetes リソースの GitOps スタイルの管理をサポートします。
アーティファクト管理	" は、GitOps ワークフローを維持するために不可欠なアーティファクト管理とバージョニングをサポートしています。
オブザーバビリティとモニタリング	" は、GitOps のオブザーバビリティの側面をサポートするモニタリングツールとの統合を提供します。
監査証跡	" は、GitOps の監査可能性の原則をサポートする詳細なデプロイログと履歴を提供します。
ロールベースのアクセスコントロール (RBAC)	このツールは、GitOps セキュリティプラクティスに従って、どのアクションを実行できるかをきめ細かく制御するための RBAC を実装します。
テンプレート化とパラメータ化	" は、再利用可能なパラメータ化されたデプロイを可能にするパイプライン定義のテンプレートをサポートしています。
環境の昇格	" は、制御された方法で環境間のアプリケーションのプロモーション (ステージングから本番稼働など) を容易にします。
CI ツールとの統合	" は、さまざまな継続的インテグレーション (CI) ツールと統合して、GitOps の原則に準拠した完全な CI/CD パイプラインを提供できます。
カスタムステージと拡張機能	このツールはカスタムステージと拡張機能をサポートしているため、チームはニーズに合わせた GitOps ワークフローを実装できます。

[面積]	ツール機能
一元管理	" は、複数の環境とクラウドプロバイダーにまたがるデプロイを管理するための一元化されたプラットフォームを提供します。

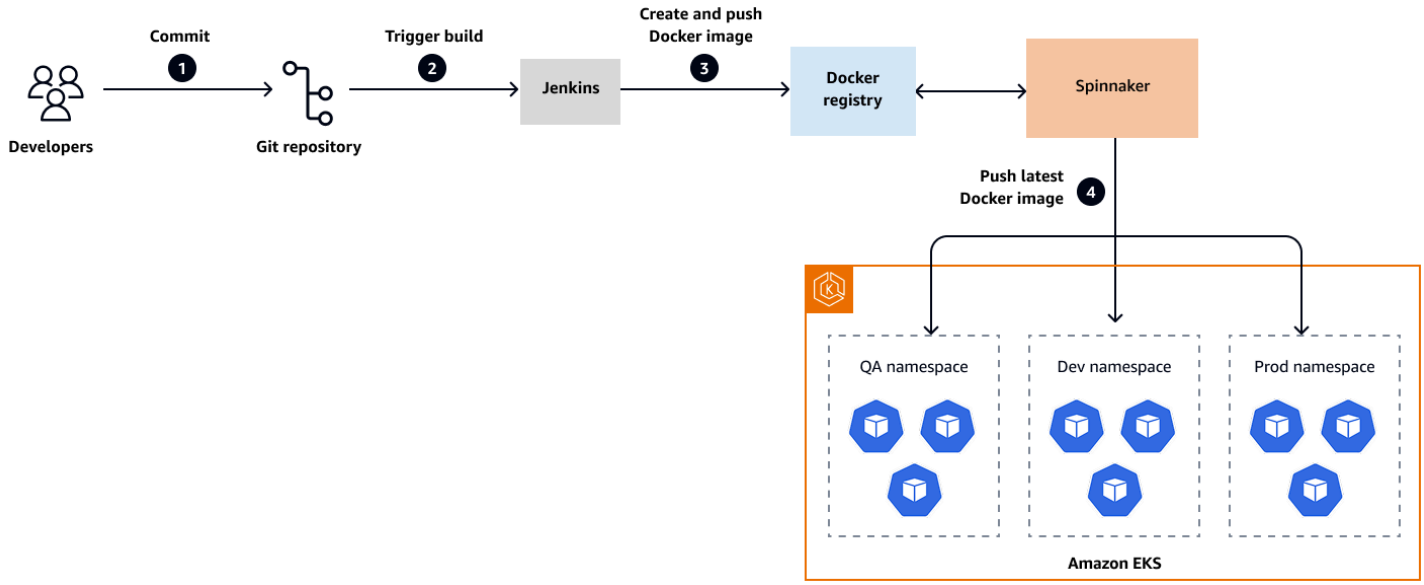
" は主に GitOps ツールとして販売されていませんが、その柔軟性と堅牢な機能セットにより、特に複雑なマルチクラウド環境で GitOps ワークフローを実装できます。Argo CD や Flux などの " と専用の GitOps ツールの主な違いは、" が高度なデプロイ戦略とマルチクラウドサポートを備えた、より包括的な継続的デリバリープラットフォームを提供することです。

「」の強みは、さまざまなクラウドプロバイダーにわたる複雑なデプロイシナリオを処理する能力と、高度なデプロイ戦略のサポートにあります。" が適切に設定されている場合、GitOps の原則を効果的に実装できます。これにより、多様で複雑な環境で GitOps プラクティスを採用したい組織にとって強力なツールになります。

詳細については、「」のドキュメントを参照してください。

アーキテクチャ

次の図は、" と Jenkins X を使用する GitOps 駆動型 CD ワークフローを示しています。詳細については、" ドキュメントを参照してください。



各パラメータの意味は次のとおりです。

- ステップ 1: コードコミット。開発者は、アプリケーションコードの変更を Git リポジトリにコミットします。これらの変更には、アプリケーション自体、Dockerfiles、または Kubernetes マニフェストの更新が含まれる場合があります。
- ステップ 2: Jenkins のビルドとイメージの作成。Jenkins は、ウェブフックまたはポーリングを介して Git リポジトリによって自動的にトリガーされます。Jenkins はアプリケーションを構築し、Docker イメージを作成し、ビルドされたイメージを設定済みの Docker レジストリ (Amazon ECR や Docker Hub など) にプッシュします。
- ステップ 3: イメージのモニタリングとパイプラインのトリガーを行います。" は、新しいイメージについて Docker レジストリを継続的にモニタリングします。新しいイメージバージョンが検出されると、" はパイプラインを自動的にトリガーしてデプロイプロセスを開始します。
- ステップ 4: ターゲット名前空間へのデプロイ。" は、新しい Docker イメージを Amazon EKS にデプロイします。パイプライン設定に基づいて、イメージはクラスター内のターゲット名前空間にデプロイされます。" は、ブルー/グリーンデプロイや Canary デプロイなどの定義されたデプロイ戦略に従って、最新のアプリケーションバージョンをデプロイします。

Rancher フリート

Rancher Fleet は、複数の Kubernetes クラスターを管理するために特別に設計された GitOps-at-scaleソリューションです。スケーラビリティとマルチクラスター管理に焦点を当てながら、GitOps の原則に厳密に従います。

GitOps のサポート

[面積]	ツール機能
単一の情報源としての Git	フリートは、複数のクラスターにわたるアプリケーションとリソースの望ましい状態を定義するための信頼できるソースとして Git リポジトリを使用します。Kubernetes マニフェスト、Helm チャート、カスタムリソースを含むすべての設定は Git に保存されます。
宣言型設定	フリートは、アプリケーションとリソースの目的の状態を宣言的に記述します。これらは、raw Kubernetes YAML、Helm チャー

[面積]	ツール機能
	ト、Kustomize ファイル、またはフリート固有のカスタムリソースです。
自動同期	フリートは Git リポジトリの変更を継続的にモニタリングします。Git 状態とクラスター状態の違いを検出すると、ターゲットクラスターに変更が自動的に適用されます。
マルチクラスター管理	フリートは、複数の Kubernetes クラスターにまたがるデプロイを管理するために特別に設計されています。1 つのコントロールプレーンから数千のクラスターを処理できます。
Kubernetes ネイティブアーキテクチャ	フリートは、一連の Kubernetes カスタムリソースとコントローラーとして構築されています。Kubernetes for GitOps オペレーションで拡張メカニズムを使用します。
継続的な調整	フリートは、クラスターの実際の状態を Git で定義されている目的の状態と常に比較します。これらの状態間で検出されたドリフトは自動的に修正されます。
クラスターのグループ化とターゲティング	フリートを使用すると、クラスターをグループ化し、特定のグループまたは個々のクラスターにデプロイをターゲットにできます。さまざまな環境やクラスタータイプにまたがる一貫したアプリケーションデプロイをサポートします。
レイヤー設定	フリートは、環境固有のオーバーレイを使用した基本設定を提供するレイヤード設定をサポートしています。これは、複数の環境を効率的に管理する GitOps のプラクティスと一致しています。

[面積]	ツール機能
Helm 統合	フリートは Helm チャートをネイティブにサポートし、複雑なアプリケーションを簡単に管理できます。GitOps ワークフローを通じて Helm リリースをバージョン管理できます。
カスタムリソース定義 (CRDs)	フリートはGitRepo や Bundle などのカスタムリソースを使用してデプロイを定義します。これらの CRDs は、GitOps ワークフローを定義する Kubernetes ネイティブな方法を提供します。
セキュリティと RBAC	フリートは、アクセスコントロールのために Kubernetes RBAC と統合されます。機密情報と認証情報の安全な管理をサポートします。
オブザーバビリティ	フリートは、クラスターとアプリケーションの同期状態に関するステータス情報を提供します。クラスターのフリート全体の GitOps プロセスに関するインサイトを提供します。
スケーラビリティ	フリートは、数千のクラスターを効率的に管理するためにスケールするように設計されています。エンタープライズ環境で大規模な GitOps オペレーションをサポートしています。
依存関係管理	さまざまなリソースとアプリケーション間の依存関係を定義できます。フリートは、複雑なデプロイで正しい順序のオペレーションに従うようにします。
カスタマイズと拡張性	フリートは、デプロイの高度なカスタマイズのためのカスタムスクリプトとライフサイクルフックをサポートしています。これにより、既存のツールやワークフローとの統合が可能になります。

[面積]	ツール機能
オフラインとエアギャップのサポート	フリートは、インターネット接続が制限されている、または接続されていない環境で動作できます。高セキュリティまたは規制された環境で GitOps ワークフローをサポートします。
プログレッシブロールアウト	フリートはクラスター間でステージングされたロールアウトをサポートしているため、制御された段階的なデプロイ戦略が可能になります。
統合管理インターフェイス	フリートは、すべてのクラスターで GitOps ワークフローを管理するための単一のインターフェイスを提供します。複雑なマルチクラスター環境でのオペレーションを簡素化します。
他の Rancher ツールとの統合	フリートは他の Rancher ツールと統合して、包括的な Kubernetes 管理ソリューションを提供します。
監査証跡とコンプライアンス	フリートは、すべての変更とデプロイの明確な監査証跡を維持します。これは、バージョン管理された Git ベースのオペレーションを通じてコンプライアンス要件を満たすのに役立ちます。

Rancher フリートは、スケーラビリティとマルチクラスター管理に重点を置いて、これらの GitOps 原則を実装しています。この設計は、さまざまな環境、データセンター、またはクラウドプロバイダーで多数の Kubernetes クラスターを管理する組織に最適です。

フリートの主な差別化要因は、GitOps を大規模に処理できることです。この機能は、多数のクラスターを管理する大企業やマネージドサービスプロバイダーにとって特に役立ちます。Argo CD や Flux などのツールは、多くの場合、個々のクラスター管理に使用されますが、フリートは大規模なクラスターフリート全体で GitOps を管理するように設計されています。

これらの GitOps の原則に従うことで、Rancher フリートは、多様で大規模な Kubernetes 環境全体でアプリケーションとリソースの一貫したスケーラブルな自動管理を実装したい組織向けのソリューションを提供します。

詳細については、フリートのドキュメントを参照してください。

アーキテクチャ

アーキテクチャとワークフローの情報については、フリートリポジトリを参照してください。

Codefresh

Codefresh は、GitOps の原則、特に Kubernetes デプロイをサポートする最新の CI/CD プラットフォームです。Codefresh は CI/CD 機能の包括的なセットを提供しており、その GitOps 機能は注目に値します。

GitOps のサポート

[面積]	ツール機能
単一の情報源としての Git	Codefresh は、Git リポジトリをアプリケーションコード、インフラストラクチャ定義、パイプライン設定の信頼できるソースとして使用します。システムへのすべての変更は Git を通じて行われ、完全な履歴と監査証跡が保証されます。
宣言型設定	Codefresh は、Git に保存されている YAML ファイルを使用して宣言型パイプライン定義をサポートしています。Kubernetes マニフェスト、Helm チャート、 CloudFormation テンプレート、およびその他の IaC ファイルは、同じリポジトリでバージョン管理できます。
GitOps ダッシュボード	Codefresh には、GitOps ワークフローを視覚化および管理するための専用の GitOps ダッシュボードが用意されています。Git とクラスターの状態間の同期ステータスを明確に表示します。
自動同期	Codefresh は Git リポジトリの変更を継続的にモニタリングします。パイプラインを自動的に

[面積]	ツール機能
	開始して、違いを検出すると、ターゲット環境に変更を適用します。
Kubernetes 統合	Codefresh は Kubernetes との深い統合を提供し、複数のクラスターにわたる GitOps スタイルのデプロイをサポートします。さまざまな Kubernetes リソースとカスタムリソース定義 (CRDsをサポートしています。
環境管理	複数の環境 (開発、ステージング、本番稼働など) をコードとして定義および管理できます。Codefresh は、GitOps プラクティスを使用して環境間の昇格をサポートしています。
Argo CD 統合	Codefresh は Argo CD と統合して GitOps 機能を強化します。CI 機能と Argo CD の CD 強度を組み合わせて、完全な GitOps ソリューションを提供します。
Helm のサポート	Codefresh は Helm チャートをサポートし、GitOps を通じて複雑なアプリケーションを簡単に管理できます。また、Helm チャートのバージョニングとプロモーションも提供します。
プログレッシブ配信	Codefresh は、Canary や Blue/Green デプロイなどの高度なデプロイ戦略をサポートしています。これらの戦略は、GitOps ワークフローを通じて実装および管理できます。
ロールバックとバージョニング	Codefresh を使用すると、デプロイ後に問題が検出された場合、以前のバージョンに簡単にロールバックできます。トレーサビリティのためにデプロイのバージョニングを維持します。

[面積]	ツール機能
承認ワークフロー	Codefresh は、デプロイの手動承認プロセスと自動承認プロセスをサポートしています。これにより、GitOps プラクティスに従って、環境間で制御されたプロモーションが可能になります。
IaC	Codefresh は、 CloudFormation や Terraform などの IaC ツールとの統合をサポートしています。これにより、アプリケーションコードとともにインフラストラクチャ定義のバージョン管理が可能になります。
オブザーバビリティとモニタリング	Codefresh には、モニタリング機能とオブザーバビリティ機能が組み込まれています。また、可視性を高めるために外部モニタリングツールとの統合も提供します。
セキュリティスキャン	Codefresh には、GitOps ワークフローに統合できるセキュリティスキャン機能が含まれています。セキュリティチェックは、自動デプロイプロセスの一部です。
監査証跡	Codefresh は、すべてのアクションと変更の包括的な監査ログを維持します。GitOps のトレーサビリティとコンプライアンスの側面をサポートします。
RBAC とアクセスコントロール	Codefresh は、きめ細かなアクセス許可管理のためのロールベースのアクセスコントロール (RBAC) を実装します。これにより、チームや環境全体で GitOps の安全な運用を確保できます。

[面積]	ツール機能
GitOps オートメーション	Codefresh には、プルリクエスト (PR) の作成やマージなど、GitOps ワークフローのさまざまな側面を自動化する機能が用意されています。
マルチクラウドおよびハイブリッドデプロイ	Codefresh は、複数のクラウドプロバイダーとオンプレミス環境にわたる GitOps ワークフローをサポートしています。
テンプレート化とパラメータ化	Codefresh は、パイプライン設定とデプロイ設定のテンプレートをサポートしています。これにより、再利用可能なパラメータ化された GitOps ワークフローが有効になります。
統合されたイメージ管理	Codefresh には、コンテナイメージ管理機能が組み込まれています。イメージビルドとデプロイを GitOps ワークフローに統合します。
GitOps for シークレット管理	Codefresh は、GitOps ワークフロー内でシークレットを管理するための安全な方法を提供します。外部シークレット管理ソリューションと統合されます。
コラボレーション機能	Codefresh は、GitOps プロセス内でのチームコラボレーションの機能を提供します。これらの機能には、コメント、通知、共有ダッシュボードが含まれます。

GitOps への Codefresh アプローチは、CI/CD 機能と GitOps プラクティスの統合で注目に値します。GitOps の原則に従いながら、ソフトウェア配信ライフサイクル全体をカバーする包括的なプラットフォームを提供することを目指しています。

GitOps エリアにおける Codefresh の主な差別化要因は、CI 機能と CD および GitOps 機能を組み合わせた統合プラットフォームアプローチです。これにより、GitOps プラクティスを実装しながら複雑な CI/CD シナリオを処理できるall-in-oneソリューションを必要とするチームに特に適しています。

Codefresh は、特に Kubernetes やクラウドネイティブテクノロジーを使用する場合に、より広範な CI/CD コンテキスト内で GitOps メソッドを採用したい組織向けのプラットフォームを提供します。

詳細については、Codefresh ドキュメントを参照してください。

Pulumi

Pulumi は、GitOps 専用に設計されていない IaC プラットフォームです。ただし、特にクラウドインフラストラクチャと Kubernetes デプロイでは、GitOps の原則を実装するために効果的に使用できます。

GitOps のサポート

[面積]	ツール機能
IaC	Pulumi では、Python、TypeScript、Go などの汎用プログラミング言語を使用してインフラストラクチャを定義できます。このコードベースのアプローチは、バージョン管理された宣言型設定に重点を置いた GitOps と一致しています。
信頼できる唯一のソースとしての Git	Pulumi のインフラストラクチャコードは、Git リポジトリに保存してバージョン管理できます。これにより、Git はインフラストラクチャ定義の唯一の信頼できるソースとして機能します。
宣言的な目的の状態	Pulumi はプログラミング言語を使用していますが、インフラストラクチャの望ましい状態を宣言的に記述しています。このコードは、インフラストラクチャを作成するstep-by-stepのプロセスではなく、インフラストラクチャがどのように見えるかを定義します。
自動同期	Pulumi は CI/CD パイプラインと統合して、Git でコードが更新されると変更を自動的に適用できます。これにより、GitOps の主要な原則で

[面積]	ツール機能
	あるインフラストラクチャの変更を継続的にデプロイできます。
マルチクラウドと Kubernetes のサポート	Pulumi は幅広いクラウドプロバイダーと Kubernetes をサポートしているため、さまざまな環境で GitOps のプラクティスに従うことができます。このツールを使用すると、さまざまなプラットフォーム間でリソースを一貫して管理できます。
状態の管理	Pulumi は、リモートで安全に保存できるインフラストラクチャの状態を管理します。この状態管理は GitOps プラクティスにとって重要であり、定義された状態とインフラストラクチャの実際の状態の間の一貫性を確保します。
ドリフトの検出と調整	Pulumi は、希望する状態 (コード内) とインフラストラクチャの実際の状態の違いを検出できます。これらの違いは、継続的な調整のために GitOps の原則に沿って調整されます。
コードとしてのポリシー	Pulumi CrossGuard を使用して、ポリシーをコードとして定義して適用できます。これにより、バージョン管理された GitOps スタイルのコンプライアンスおよびセキュリティポリシーの管理が可能になります。
シークレットの管理	Pulumi は、インフラストラクチャコード内で機密情報を管理するための安全な方法を提供します。GitOps セキュリティプラクティスにとって重要な外部シークレット管理システムとの統合をサポートしています。

[面積]	ツール機能
モジュール式および再利用可能なコンポーネント	Pulumi は、再利用可能なコンポーネントとモジュールの作成をサポートしています。このモジュール性は、複雑なマルチ環境デプロイを管理するための GitOps プラクティスと一致しています。
プレビューと計画	Pulumi では、変更を適用する前にプレビューできます。これにより、インフラストラクチャに対する安全で予測可能な変更という GitOps の原則がサポートされます。
ロールバックと履歴	Pulumi はデプロイの履歴を維持し、以前の状態へのロールバックをサポートします。これは、GitOps のトレーサビリティと可逆性の原則と一致しています。
インフラストラクチャの継続的な配信	Pulumi は CI/CD パイプラインに統合して、インフラストラクチャの変更を継続的に提供できます。インフラストラクチャコードの自動テストと検証をサポートしています。
RBAC とアクセスコントロール	Pulumi は、インフラストラクチャを変更できるユーザーを管理するためのロールベースのアクセスコントロールを提供します。これにより、GitOps セキュリティとガバナンスのプラクティスがサポートされます。
オブザーバビリティとログ記録	Pulumi は、インフラストラクチャの変更のログ記録とモニタリング機能を提供します。これらの機能は、GitOps プラクティスのオブザーバビリティの側面をサポートします。
他のツールとの統合	Pulumi はクラウド内のさまざまなツールと統合できます。この柔軟性により、包括的な GitOps ワークフローが可能になります。

[面積]	ツール機能
環境管理	Pulumi は、異なる設定で同じコードベースを使用することで、複数の環境 (開発、ステージング、本番稼働) の管理をサポートします。これは、一貫したマルチ環境管理のための GitOps プラクティスと一致しています。
依存関係管理	Pulumi はリソース間の依存関係を処理し、オペレーションの正しい順序を確保します。これは、相互依存コンポーネントを含む複雑な GitOps デプロイに不可欠です。
カスタムリソースプロバイダー	Pulumi では、API 駆動型サービスを管理するためのカスタムプロバイダーを作成できます。これによりGitOps のプラクティスは、標準のクラウドサービスを超える幅広いリソースに拡張されます。
コラボレーション機能	Pulumi は、共有状態とアクセスコントロールによるチームコラボレーションをサポートしています。これにより、チーム環境での GitOps ワークフローが容易になります。

これらの Pulumi 機能を使用することで、組織はインフラストラクチャに GitOps プラクティスを実装できます。特に、きめ細かな制御や複雑なロジックが必要な場合や、単一の一貫性のあるフレームワーク内で多様なクラウドリソースとオンプレミスリソースのセットを管理したい場合などです。

GitOps に対する Pulumi のアプローチは、GitOps の原則に準拠しながら、汎用プログラミング言語のパワーと柔軟性をインフラストラクチャ管理にもたらすため、ユニークです。これは、使い慣れたプログラミング言語の使用を好み、ソフトウェアエンジニアリングのベストプラクティスをインフラストラクチャ管理に適用したいチームにとって特に有利です。

GitOps での Pulumi の主な差別化要因は、標準プログラミング言語を使用してインフラストラクチャを定義することです。従来の GitOps ツールでは YAML またはドメイン固有の言語がよく使用されますが、Pulumi ではより複雑なロジック、コードの再利用の向上、既存の開発ワークフローとの統合が容易になります。

詳細については、Pulumi のドキュメントを参照してください。

GitOps ツールの比較

前のセクションで説明した 9 つの GitOps ツールの比較を次に示します。ツールを選択するときは、特定の要件、既存のインフラストラクチャ、チームの専門知識、必要なレベルの制御とカスタマイズを考慮してください。

使いやすさ:

- Argo CD、Flux、Rancher フリートは一般的にセットアップが容易です。
- " と Jenkins X には、より急な学習曲線があります。
- Weave GitOps では、高度な機能のセットアップがさらに必要になる場合があります。
- GitLab CI/CD と Codefresh は、統合されたエクスペリエンスを提供します。

Kubernetes 統合

- Argo CD、Flux、および Rancher フリートは、非常に Kubernetes 中心です。
- Jenkins X と Weave GitOps は、より広範な DevOps 機能を提供します。
- 他のツールは、排他的に焦点を絞ることなく Kubernetes をサポートします。

CI/CD 機能

- Jenkins X、GitLab CI/CD、および Codefresh は、完全な CI/CD ソリューションを提供します。
- Argo CD、Flux、および Weave GitOps はワークフローの CD の側面に重点を置いており、多くの場合、個別の CI ツールとの統合が必要です。

GitOps の改良

- Argo CD と Flux は、特に GitOps に焦点を当てたツールです。
- 他のツールには、さまざまな程度の GitOps の原則が組み込まれています。

マルチクラウドのサポート

- マルチクラウドシナリオでは、" と Pulumi が優れています。
- 他のツールはクラウド間で動作できますが、追加のセットアップが必要になる場合があります。

マルチクラスターのサポート

- すべてのツールは、マルチクラスターデプロイをサポートしています。
- Argo CD と Weave GitOps には、より高度なマルチクラスター管理機能があります。

Integration

- Flux には強力な Cloud Native Computing Foundation (CNCF) バッキングがあります。
- Argo CD には大規模でアクティブなコミュニティがあります。
- Argo CD と Flux には強力な Kubernetes 統合があります。
- Jenkins X はより広範な Jenkins システムを使用します。
- Weave GitOps は新しいですが、強力な商用バッキングで成長しています。
- GitLab CI/CD は GitLab と緊密に統合されています。
- Rancher フリートは Rancher システム内でうまく機能します。

コミュニティとサポート

- Flux には強力な CNCF バッキングがあります。
- Argo CD、GitLab、および " には大規模なコミュニティがあります。
- ほとんどのツールで商用サポートを利用できます。

エンタープライズ機能

- Weave GitOps と Jenkins X は、デフォルトでよりエンタープライズに焦点を当てた機能を提供します。
- Argo CD と Flux にはエンタープライズサービスがあり、エンタープライズ用に拡張できます。

柔軟性と拡張性

- Flux は高度にモジュール化され、拡張可能です。
- Argo CD は、優れたカスタマイズオプションを提供します。
- Jenkins X は非常に拡張可能ですが、より多くの労力が必要になる場合があります。
- Weave GitOps は、拡張性を必要とせずに完全なソリューションを提供することを目指しています。

スケーラビリティ

- " と GitLab CI/CD は、エンタープライズスケーラビリティで知られています。
- Argo CD と Flux は、大規模な Kubernetes デプロイを適切に処理します。

インフラストラクチャ管理

- Pulumi はインフラストラクチャ管理に重点を置いています。
- Weave GitOps と Flux は、優れた IaC 機能を提供します。

プログラミングモデルと言語のサポート

- Pulumi では、Python、Go、TypeScript、C#、Java などの汎用プログラミング言語を使用してインフラストラクチャを定義できます。Pulumi の標準言語を使用すると、インフラストラクチャコードを使い慣れた開発ワークフロー、テストプラクティス、複雑なロジックと統合できます。
- Terraform は HashiCorp 設定言語 (HCL) を使用します。
- CloudFormation は JSON テンプレートと YAML テンプレートを使用します。
- Argo CD、Flux、Rancher フリート、Weave GitOps、"、GitLab CI/CD は主に YAML または宣言型設定ファイルを管理します。
- Jenkins X は YAML およびスクリプトベースのパイプラインを管理しますが、IaC 用の汎用プログラミングをネイティブに提供していません。

Argo CD と Flux のユースケース

このセクションでは、純粋な GitOps 機能を提供する Argo CD と Flux の 2 つのツールに焦点を当てます。このコンテキストでは、純粋な GitOps は、Git リポジトリがアプリケーションとインフラストラクチャの望ましい状態の唯一の信頼できるソースとして機能するモデルを指します。すべての変更は Git コミットを通じて行われ、システムはリポジトリで定義されている状態と一致するようにライブ環境を自動的に同期します。Git オペレーション以外では、手動による介入は必要ありません。

一般的な考慮事項

- ビジュアル管理とアプリケーション中心のワークフローが重要な環境で Argo CD を使用することをお勧めします。
- 軽量ソリューション、強力なマルチテナンシー、またはより広範な Cloud Native Computing Foundations (CNCF) ネットワークとの深い統合が必要な場合は、Flux を選択できます。
- Argo CD は、直感的な UI により、従来の CI/CD から GitOps に移行しているチームにとって魅力的なことがよくあります。
- Flux は、CLI ベースのワークフローと IaC プラクティスが既に確立されているクラウドネイティブ環境で好まれることがよくあります。

最終的に、Argo CD と Flux の選択は、多くの場合、特定の組織のニーズ、既存のツール、チームの好みによって異なります。どちらのツールもほとんどの GitOps シナリオを処理できるため、特定のユースケースと要件に基づいて評価することをお勧めします。

Argo CD のユースケース

ビジュアル管理：

- デプロイを管理し、アプリケーションの状態を視覚化するために使いやすい UI が必要な場合。
- モニタリングとトラブルシューティングのためのグラフィカルインターフェイスを好むチーム向け。

アプリケーション中心のアプローチ：

- 個々のリソースを管理するのではなく、アプリケーションレベルでデプロイを管理する場合。
- アプリケーションの概念を中心にデプロイを構築する組織向け。

マルチクラスター管理：

- 複数のクラスターにまたがるデプロイを管理する場合、これが主な要件です。
- クラスターが多い複雑な分散環境の場合。

ウェーブのロールバックと同期：

- 同期ウェーブや手動介入など、デプロイプロセスをきめ細かく制御する必要がある場合。
- 複雑なロールバック戦略を必要とするシナリオの場合。

既存のツールとの統合：

- Argo ワークフローや Argo イベントなど、Argo プロジェクトで既に他のツールを使用している場合。

エンタープライズ環境：

- デフォルトでは、堅牢な RBAC とシングルサインオン統合を必要とする大企業向け。

Flux のユースケース

軽量デプロイ：

- より軽量でリソースを大量に消費しない GitOps ソリューションが必要な場合。
- リソースが制約される可能性のあるエッジコンピューティングまたは IoT シナリオの場合。

イメージの自動更新：

- 新しいコンテナイメージの自動検出とデプロイが重要な要件である場合。
- 頻繁なイメージ更新による継続的なデプロイに重点を置くチーム向け。

マルチテナンシー：

- 特に共有クラスター環境で強力なマルチテナンシーサポートが必要な場合。
- チームまたはプロジェクトを厳密に分離しているサービスプロバイダーまたは大規模な組織向け。

IaC:

- 同じ GitOps ワークフローを使用してアプリケーションとインフラストラクチャの両方を管理する場合は、重要です。
- IaC パラダイムに大きく投資されているチーム向け。

Helm 統合：

- Helm チャートの広範な使用がデプロイ戦略の一部である場合。
- Helm ベースのデプロイが複雑な環境の場合。

CNCF プロジェクト統合：

- 他の CNCF プロジェクトとの緊密な統合が重要な場合。
- CNCF のテクノロジーと原則に沿った組織向け。

モジュラーアーキテクチャ：

- GitOps ツールキットの特定のコンポーネントのみを柔軟に使用する必要がある場合。
- モジュラーコンポーネントを使用してカスタム GitOps ワークフローを構築するチーム向け。

プログレッシブ配信：

- Canary リリースや A/B テストなどの高度なデプロイ戦略がコア要件である場合。

機能の比較

[面積]	Argo CD	Flux
GitOps の基本原則のサポート	 はい	 はい

[面積]	Argo CD	Flux
アーキテクチャ	Kubernetes GitOps ワークフローを実装するためのEnd-to-endアプリケーション	GitOps 用の Kubernetes CRDs とコントローラーを提供します GitOps
セットアップ	シンプル	複雑
Helm のサポート	 はい	 はい
Kustomize のサポート	 はい	 はい
統合 GUI	CLI とフル機能のウェブ UI	CLI とオプションの軽量ウェブインターフェイス
RBAC サポート	きめ細かなコントロール	Kubernetes ネイティブ RBAC
マルチテナンシーとマルチクラスターのサポート	マルチクラスターの曖昧なサポート	マルチテナンシーの優れたサポート
シングルサインオン認証	 はい	 はい
同期オートメーション	ウィンドウを同期する機能	調整間隔を設定する機能
部分同期	 はい	 いいえ

[面積]	Argo CD	Flux
照合プロセス	手動同期と自動同期をサポートします。いくつかの異なる戦略を使用できます。	手動同期と自動同期をサポートします。
拡張性	カスタムラグインをサポートします。カスタマイズオプションが制限されています。	カスタムコントローラーをサポートします。優れた拡張性とサードパーティー統合。
同時実行数のサポート	大規模でアクティブなコミュニティ。	小規模ではあるが成長しているコミュニティ。
スケーラビリティ	スケーラビリティは良好ですが、ウェブ UI のデータ取得レートによって制限されます。コミュニティ分析では、数万のアプリケーションのサポートが提案されています。	水平方向および垂直方向のスケーラビリティ、最大数万のアプリケーションに関する明確なガイド。

GitOps ツールを選択するためのベストプラクティス

このセクションでは、EKS クラスターの GitOps ツールを選択するための考慮事項、ヒント、ベストプラクティスについて説明します。適切な選択は、特定のコンテキスト、要件、長期戦略によって異なります。多くの場合、最終決定を下す前に、上位の選択で概念実証を実施することが有益です。

組織のニーズと能力を評価します。

- チームの現在のスキルセットと、新しいツールを学習する意欲を考慮してください。
- Amazon EKS 環境の複雑さを評価します。(たとえば、1 つのクラスターまたは複数のクラスターを使用していますか?)
- コンプライアンス、セキュリティ、スケーラビリティに関する特定の要件を決定します。

> **ベストプラクティス**
>
> 必要な機能と、有用だが必須ではない機能の概要を示す詳細な要件ドキュメントを作成します。

ツールの成熟度と導入を評価します。

- 潜在的な GitOps ツールの成熟度とその業界での採用率を調査します。
- Amazon EKS 環境で実績のあるツールを探します。

> **ベストプラクティス**
>
> Cloud Native Computing Foundation (CNCF) ネットワークで広く採用され、強力なプレゼンスを持つツールを優先します。

既存のツールチェーンとの統合を検討してください。

- GitOps ツールが現在の CI/CD パイプライン、モニタリングソリューション、その他の運用ツールとどの程度統合されているかを評価します。
- IAM、Amazon ECR、CloudWatch AWS のサービス などの とのネイティブ統合を探します。

> **ベストプラクティス**
>
> 最終的な決定を行う前に、統合機能をテストするための概念実証を作成します。

セキュリティ機能を評価します。

- 堅牢なロールベースのアクセスコントロール (RBAC) 機能を持つツールを優先し、IAM と適切に統合します。
- 安全なシークレット管理とポリシーの適用をサポートする機能を探します。

> **ベストプラクティス**
>
> コードとしてのポリシーや自動コンプライアンスチェックなど、GitOps ベースのセキュリティプラクティスをサポートするツールを選択します。

スケーラビリティとパフォーマンスを評価します。

- 多数のアプリケーションやクラスターでツールがどのように機能するかを検討してください。
- クラスターのパフォーマンスとリソースの消費への影響を評価します。

> **ベストプラクティス**
>
> 本番環境に似たワークロードでパフォーマンステストを実行し、ツールがスケールを処理できることを確認します。

マルチクラスターとマルチ環境のサポートを検討してください。

- 複数の EKS クラスターがある場合、または持つ予定がある場合は、強力なマルチクラスター管理機能を持つツールを優先します。
- さまざまな環境 (開発、ステージング、本番稼働など) での一貫したデプロイをサポートする機能を探します。

> **ベストプラクティス**
>
> 環境固有の設定を維持しながら、複数のクラスターを一元管理できるツールを選択します。

オブザーバビリティとモニタリング機能を評価します。

- デプロイの状態とクラスターの状態を明確に可視化するツールを探します。
- ツールが既存のモニタリングおよびログ記録ソリューションとどの程度統合されているかを検討してください。

> **ベストプラクティス**
>
> プロアクティブな問題検出のために、カスタマイズ可能なダッシュボードとアラートメカニズムを提供するツールに優先順位を付けます。

学習曲線とドキュメントを評価します。

- ツールのドキュメントの品質と包括性を評価します。
- トレーニングリソースの可用性とコミュニティサポートを検討してください。

> **ベストプラクティス**
>
> 適切に管理されたドキュメント、アクティブなコミュニティフォーラム、公式のトレーニングプログラムまたは認定があるツールを選択します。

コストとリソースの使用率を考慮します。

- ツールの導入に伴う直接コスト (ライセンスやサポートなど) と間接コスト (運用オーバーヘッドやトレーニングコストなど) の両方を評価します。
- コンピューティングリソースとストレージリソースの消費量の観点からツールの効率を評価します。

> **ベストプラクティス**
>
> 短期コストと長期コストの両方を含む総所有コスト (TCO) 分析を実行します。

柔軟性とカスタマイズオプションを評価します。

- 特定のニーズに合わせてワークフローをカスタマイズできるツールを探します。
- プラグインまたは APIs。

> **ベストプラクティス**
>
> デフォルトの機能と、独自の要件に合わせて をカスタマイズする機能のバランスを取るツールを選択します。

継続的デリバリーとプログレッシブデプロイ機能を評価します。

- Canary リリースやブルー/グリーンデプロイなどの高度なデプロイ戦略をサポートするツールを探します。
- これらの戦略の実装と管理の容易さを評価します。

> **ベストプラクティス**
>
> プログレッシブ配信パターンの組み込みサポートを提供するツールを優先して、デプロイのリスクを最小限に抑えます。

ベンダーのロックインと移植性を検討してください。

- 特定のクラウドプロバイダーまたはテクノロジーに対するツールの依存関係を評価します。
- 必要に応じて、将来的に別のツールへの移行が容易になることを検討してください。

> **ベストプラクティス**
>
> オープン標準を使用し、GitOps 設定のエクスポート機能を提供するツールを優先します。

コミュニティのサポートと拡張機能を評価します。

- ユーザーコミュニティのサイズとアクティビティを確認します。
- サードパーティーの統合とプラグインの可用性を評価します。

> ベストプラクティス
>
> コミュニティフォーラムまたはユーザーグループに参加して、決定する前に他のユーザーから直接体験してください。

コンプライアンスと監査の要件を検討します。

- 監査証跡やレポートなど、ツールがコンプライアンスニーズをどの程度サポートしているかを評価します。
- コンプライアンスの維持と実証に役立つ機能を探します。

> ベストプラクティス
>
> 包括的な監査ログを提供し、コンプライアンスレポートの生成をサポートするツールを選択します。

ロールバック機能とディザスタリカバリ機能を評価します。

- ロールバックメカニズムの容易性と信頼性を評価します。
- ツールがディザスタリカバリシナリオをどのようにサポートしているかを検討してください。

> ベストプラクティス
>
> 評価の一環として、ロールバックと復旧プロセスを徹底的にテストします。

よくある質問

Q: Amazon EKS で最も人気のある GitOps ツールは何ですか？

A: Amazon EKS の最も一般的な GitOps ツールには、Argo CD、Flux、Jenkins X、GitLab CI/CD などがあります。各ツールには長所がありますが、Argo CD と Flux は Kubernetes ネイティブアプローチと強力なコミュニティサポートで特に高く評価されています。

Q: GitOps は EKS クラスター管理をどのように改善しますか？

A: GitOps は、インフラストラクチャのバージョン管理、自動デプロイ、宣言型設定によるセキュリティの向上、ロールバックの簡素化、監査可能性の向上を提供することで、EKS クラスター管理を改善します。また、コラボレーションを強化し、デプロイの人為的ミスを減らします。

Q: Amazon EKS の GitOps ツールでは、どのような主要な機能を探す必要がありますか？

A: 探すべき主な機能には、シームレスな Amazon EKS 統合、堅牢な RBAC、マルチクラスターサポート、オブザーバビリティ機能、プログレッシブデリバリー戦略のサポート、スケーラビリティ、IAM や Amazon ECR AWS のサービス などの との統合などがあります。

Q: Amazon EKS で GitOps を実装するときにセキュリティを確保するにはどうすればよいですか？

A: セキュリティを確保するには、IAM との強力な RBAC 統合、安全なシークレット管理、暗号化された Git リポジトリのサポート、およびセキュリティポリシーをコードとして実装する機能を持つツールを選択します。また、ツールが包括的な監査ログを提供していることを確認します。

Q: GitOps ツールはマルチクラスター Amazon EKS 環境を処理できますか？

A: はい。Argo CD や Flux などの GitOps ツールには、堅牢なマルチクラスター管理機能があります。これにより、1 つのコントロールプレーンから複数の EKS クラスターを管理できるため、環境間の一貫性が確保されます。

Q: GitOps ツールは既存の CI/CD パイプラインとどのように統合されますか？

A: GitOps ツールは通常、パイプラインのデプロイステージとして機能することで、既存の CI/CD パイプラインと統合されます。変更は Git リポジトリにプッシュされたときに CI ツールによってトリガーされ、EKS クラスターへのデプロイプロセスを自動化します。

Q: Amazon EKS で GitOps を実装する際の課題は何ですか？

A: 一般的な課題には、シークレットの安全な管理、適切なアクセスコントロールの確保、ステートフルアプリケーションの処理、Git とクラスター状態間のドリフトの管理、GitOps モデルへのチームワークフローの適応などがあります。

Q: GitOps ツールは Amazon EKS のロールバックをどのように処理しますか？

A: GitOps ツールは通常、Git リポジトリの以前のコミットに戻すことでロールバックを処理します。これにより、以前の既知の正常な状態のデプロイが自動的にトリガーされ、高速で信頼性の高いロールバックが発生します。

Q: GitOps ツールは Amazon EKS アドオンやその他の AWS リソースを管理できますか？

A: 多くの GitOps ツールでは、特に Terraform や などの IaC ツールと組み合わせると、Amazon EKS アドオンや一部の AWS リソースを管理できます CloudFormation。ただし、この機能の範囲は異なる場合があります。各ツールに関する具体的な情報については、GitOps ツールセクションを参照してください。

Q: GitOps ツールは Amazon EKS のコンプライアンス要件をどのようにサポートしますか？

A: GitOps ツールは、すべての変更の明確な監査証跡を提供し、承認プロセスを実施し、コンプライアンスチェックを自動化するためのコードとしてポリシーを実装し、詳細なログ記録とレポート機能を提供することで、コンプライアンスをサポートします。

Q: Amazon EKS で GitOps を実装するための学習曲線は何ですか？

A: 学習曲線は、ツールとチームの既存の知識によって異なります。一般的に、Git、Kubernetes、Amazon EKS に精通しているチームは、他のチームよりも迅速に適応します。ほとんどの一般的なツールは、導入を容易にするために広範なドキュメントとトレーニングリソースを提供します。

Q: GitOps ツールは Amazon EKS でシークレット管理をどのように処理しますか？

A: GitOps ツールは通常、 AWS Secrets Manager や HashiCorp Vault などの外部シークレット管理ソリューションと統合されます。一部のツールでは、Git リポジトリに保存されているシークレットの暗号化が組み込まれています。

Q: GitOps ツールは Amazon EKS のステートレスアプリケーションとステートフルアプリケーションの両方で動作しますか？

A: はい、GitOps ツールはステートレスアプリケーションとステートフルアプリケーションの両方で使用できます。ただし、ステートフルアプリケーションの管理には、永続的なボリュームの処理や更新中のデータ整合性の確保など、追加の考慮事項が必要になることがよくあります。

Q: GitOps ツールは Amazon EKS での Canary または Blue/Green デプロイをどのようにサポートしますか？

A: 多くの GitOps ツールでは、高度なデプロイ戦略のサポートが組み込まれています。新しいバージョンの段階的なロールアウトを管理し、問題をモニタリングし、問題が検出された場合は自動的にロールバックできます。これらのオペレーションはすべて、Git リポジトリでコードとして定義されます。

Q: GitOps ツールを使用する場合と CI/CD パイプライン**`kubectl apply`**で を使用する場合の違いは何ですか？

A: GitOps ツールは、ドリフトの自動検出と調整、プルベースのデプロイによるセキュリティの向上、監査性の向上、より高度なデプロイ戦略など、シンプルな`kubectl apply`コマンドよりも利点があります。また、クラスターの状態全体を管理するためのより包括的なアプローチも提供します。

リソース

以下のリソースは、EKS クラスターの GitOps ツールを選択する際に、情報に基づいた意思決定に役立つ公式のドキュメント、実践的なガイド、ケーススタディ、詳細な分析を提供します。実装戦略、ベストプラクティス、さまざまなツール間の比較、実際のエクスペリエンスなど、GitOps のさまざまな側面について説明します。

AWS リソース：

- Amazon EKS ドキュメント
- GitOps による Amazon EKS の自動化 (AWS ブログ記事)
- Weaveworks を使用した EKS での GitOps の概要 (AWS ワークショップ)
- Flux ラボ (Amazon EKS ワークショップ)
- Argo CD ラボ (Amazon EKS ワークショップ)

GitOps とツールのドキュメント：

- GitOps 継続的デプロイとプログレッシブセキュリティのベストプラクティス (オンデマンド DevOps.com ウェビナー)
- Kubernetes documentation
- Argo CD ドキュメント
- Flux ドキュメント
- Weave GitOps ドキュメント
- Jenkins X ドキュメント
- GitLab CI/CD ドキュメント
- " ドキュメント
- Rancher フリートのドキュメント
- Codefresh ドキュメント

ドキュメント履歴

以下の表は、本ガイドの重要な変更点について説明したものです。今後の更新に関する通知を受け取る場合は、RSS フィード をサブスクライブできます。

変更	説明	日付
初版発行	—	2025 年 4 月 30 日

AWS 規範ガイダンスの用語集

以下は、 AWS 規範ガイダンスが提供する戦略、ガイド、パターンで一般的に使用される用語です。エントリを提案するには、用語集の最後のフィードバックの提供リンクを使用します。

数字

7 Rs

アプリケーションをクラウドに移行するための 7 つの一般的な移行戦略。これらの戦略は、ガートナーが 2011 年に特定した 5 Rs に基づいて構築され、以下で構成されています。

- リファクタリング/アーキテクチャの再設計 — クラウドネイティブ特徴を最大限に活用して、俊敏性、パフォーマンス、スケーラビリティを向上させ、アプリケーションを移動させ、アーキテクチャを変更します。これには、通常、オペレーティングシステムとデータベースの移植が含まれます。例: オンプレミスの Oracle データベースを Amazon Aurora PostgreSQL 互換エディションに移行する。
- リプラットフォーム (リフトアンドリシェイプ) — アプリケーションをクラウドに移行し、クラウド機能を活用するための最適化レベルを導入します。例: お客様のオンプレミスの Oracle データベースを AWS クラウドの Oracle 用の Amazon Relational Database Service (Amazon RDS) に移行する。
- 再購入 (ドロップアンドショップ) — 通常、従来のライセンスから SaaS モデルに移行して、別の製品に切り替えます。例: 顧客関係管理 (CRM) システムを Salesforce.com に移行する。
- リホスト (リフトアンドシフト) — クラウド機能を活用するための変更を加えずに、アプリケーションをクラウドに移行します。例: お客様のオンプレミスの Oracle データベースを AWS クラウドの EC2 インスタンス上の Oracle に移行する。
- 再配置 (ハイパーバイザーレベルのリフトアンドシフト) – 新しいハードウェアを購入したり、アプリケーションを書き換えたり、既存の運用を変更したりすることなく、インフラストラクチャをクラウドに移行できます。オンプレミスプラットフォームから同じプラットフォームのクラウドサービスにサーバーを移行します。例： Microsoft Hyper-Vアプリケーションを に移行します AWS。
- 保持 (再アクセス) — アプリケーションをお客様のソース環境で保持します。これには、主要なリファクタリングを必要とするアプリケーションや、お客様がその作業を後日まで延期したいアプリケーション、およびそれらを移行するためのビジネス上の正当性がないため、お客様が保持するレガシーアプリケーションなどがあります。
- 廃止 — お客様のソース環境で不要になったアプリケーションを停止または削除します。

A

ABAC

「属性ベースのアクセス制御」をご覧ください。

抽象化されたサービス

「マネージドユーザー」をご覧ください。

ACID

「原子性、一貫性、分離性、耐久性 (ACID)」をご覧ください。

アクティブ/アクティブ移行

(双方向レプリケーションツールまたは二重書き込み操作を使用して) ソースデータベースとターゲットデータベースを同期させ、移行中に両方のデータベースが接続アプリケーションからのトランザクションを処理するデータベース移行方法。この方法では、1 回限りのカットオーバーの必要がなく、管理された小規模なバッチで移行できます。アクティブ/パッシブ移行よりも柔軟な方法ですが、さらに多くの作業が必要となります。

アクティブ/パッシブ移行

ソースデータベースとターゲットデータベースを同期させながら、データがターゲットデータベースにレプリケートされている間、接続しているアプリケーションからのトランザクションをソースデータベースのみで処理するデータベース移行方法。移行中、ターゲットデータベースはトランザクションを受け付けません。

集計関数

複数行に処理を行い、グループ全体を対象に単一の戻り値を計算する SQL 関数。集計関数の例としては、SUM や MAX などがあります。

AI

「人工知能」をご覧ください。

AIOps

「AI オペレーション」をご覧ください。

匿名化

データセット内の個人情報を完全に削除するプロセス。匿名化は個人のプライバシー保護に役立ちます。匿名化されたデータは、もはや個人データとは見なされません。

アンチパターン

繰り返し起こる問題に対して頻繁に用いられる解決策で、その解決策が逆効果であったり、効果がなかったり、代替案よりも効果が低かったりするもの。

アプリケーション制御

マルウェアからシステムを保護するために、承認されたアプリケーションのみを使用できるようにするセキュリティアプローチ。

アプリケーションポートフォリオ

アプリケーションの構築と維持にかかるコスト、およびそのビジネス価値を含む、組織が使用する各アプリケーションに関する詳細情報の集まり。この情報は、ポートフォリオの検出と分析プロセス の重要な要素であり、移行、モダナイズ、最適化するアプリケーションを特定し、優先順位を付けるのに役立ちます。

人工知能 (AI)

コンピューティングテクノロジーを使用し、学習、問題の解決、パターンの認識など、通常は人間に関連づけられる認知機能の実行に特化したコンピュータサイエンスの分野。詳細については、「人工知能 (AI) とは何ですか?」をご覧ください。

AI オペレーション (AIOps)

機械学習技術を使用して運用上の問題を解決し、運用上のインシデントと人の介入を減らし、サービス品質を向上させるプロセス。 AWS 移行戦略での AIOps の使用方法については、オペレーション統合ガイド を参照してください。

非対称暗号化

暗号化用のパブリックキーと復号用のプライベートキーから成る 1 組のキーを使用した、暗号化のアルゴリズム。パブリックキーは復号には使用されないため共有しても問題ありませんが、プライベートキーの利用は厳しく制限する必要があります。

原子性、一貫性、分離性、耐久性 (ACID)

エラー、停電、その他の問題が発生した場合でも、データベースのデータ有効性と運用上の信頼性を保証する一連のソフトウェアプロパティ。

属性ベースのアクセス制御 (ABAC)

部署、役職、チーム名など、ユーザーの属性に基づいてアクセス許可をきめ細かく設定する方法。詳細については、 AWS Identity and Access Management (IAM) ドキュメントの「 の ABAC AWS」を参照してください。

信頼できるデータソース

最も信頼性のある情報源とされるデータのプライマリーバージョンを保存する場所。匿名化、編集、仮名化など、データを処理または変更する目的で、信頼できるデータソースから他の場所にデータをコピーすることができます。

アベイラビリティーゾーン (AZ)

他のアベイラビリティーゾーンの障害から AWS リージョン 隔離され、同じリージョン内の他のアベイラビリティーゾーンへの低コストで低レイテンシーのネットワーク接続を提供する 内の別の場所。

AWS クラウド導入フレームワーク (AWS CAF)

組織がクラウドへの移行を成功させるための効率的で効果的な計画を立て AWS るための、 のガイドラインとベストプラクティスのフレームワークです。 AWS CAF は、ビジネス、人材、ガバナンス、プラットフォーム、セキュリティ、運用という 6 つの重点分野にガイダンスを整理しています。ビジネス、人材、ガバナンスの観点では、ビジネススキルとプロセスに重点を置き、プラットフォーム、セキュリティ、オペレーションの視点は技術的なスキルとプロセスに焦点を当てています。例えば、人材の観点では、人事 (HR)、人材派遣機能、および人材管理を扱うステークホルダーを対象としています。この観点から、 AWS CAF は、クラウド導入を成功させるための組織の準備に役立つ人材開発、トレーニング、コミュニケーションのガイダンスを提供します。詳細については、AWS CAF ウェブサイトと AWS CAF のホワイトペーパー を参照してください。

AWS ワークロード認定フレームワーク (AWS WQF)

データベース移行ワークロードを評価し、移行戦略を推奨し、作業見積もりを提供するツール。AWS WQF は AWS Schema Conversion Tool (AWS SCT) に含まれています。データベーススキーマとコードオブジェクト、アプリケーションコード、依存関係、およびパフォーマンス特性を分析し、評価レポートを提供します。

B

不正なボット

個人や組織に混乱や損害を与えることを目的としたボット。

BCP

「ビジネス継続性計画 (BCP)」をご覧ください。

動作グラフ

リソースの動作とインタラクションを経時的に示した、一元的なインタラクティブビュー。Amazon Detective の動作グラフを使用すると、失敗したログオンの試行、不審な API 呼び出し、その他同様のアクションを調べることができます。詳細については、Detective ドキュメントの「動作グラフのデータ」を参照してください。

ビッグエンディアンシステム

最上位バイトを最初に格納するシステム。「エンディアン性」もご覧ください。

二項分類

バイナリ結果 (2 つの可能なクラスのうちの 1 つ) を予測するプロセス。例えば、お客様の機械学習モデルで「この E メールはスパムですか、それともスパムではありませんか」などの問題を予測する必要があるかもしれません。または「この製品は書籍ですか、車ですか」などの問題を予測する必要があるかもしれません。

ブルームフィルター

要素がセットのメンバーであるかどうかをテストするために使用される、確率的でメモリ効率の高いデータ構造。

ブルー/グリーンデプロイ

それぞれが独立しているが、同一の環境を 2 つ作成するデプロイ戦略。現在のアプリケーションバージョンを 1 つの環境 (ブルー) で実行し、新しいアプリケーションバージョンを別の環境 (グリーン) で実行します。この戦略は、最小限の影響で迅速にロールバックするのに役立ちます。

ボット

インターネット経由で自動タスクを実行し、人間のアクティビティややり取りをシミュレートするソフトウェアアプリケーション。インターネット上の情報のインデックスを作成するウェブクローラーなど、一部のボットは有用または有益です。悪質なボットと呼ばれる他のボットの中には、個人や組織を混乱させたり、損害を与えたりすることを意図したものもあります。

ボットネット

マルウェアに感染しており、ボットハーダーまたはボットオペレーターと呼ばれる単一の当事者によって制御されているボットのネットワーク。ボットネットは、ボットとその影響力を拡大する仕組みとして、非常によく知られています。

ブランチ

コードリポジトリに含まれる領域。リポジトリに最初に作成するブランチは、メインブランチといいます。既存のブランチから新しいブランチを作成し、その新しいブランチで機能を開発した

り、バグを修正したりできます。機能を構築するために作成するブランチは、通常、機能ブランチと呼ばれます。機能をリリースする準備ができたら、機能ブランチをメインブランチに統合します。詳細については、「ブランチの概要」(GitHub ドキュメント) を参照してください。

ブレークグラスアクセス

例外的な状況では、承認されたプロセスを通じて、ユーザーが AWS アカウント 通常アクセス許可を持たない にすばやくアクセスできるようにします。詳細については、 AWS Well-Architected ガイダンスの「ブレークグラス手順の実装」インジケータを参照してください。

ブラウンフィールド戦略

環境の既存インフラストラクチャ。システムアーキテクチャにブラウンフィールド戦略を導入する場合、現在のシステムとインフラストラクチャの制約に基づいてアーキテクチャを設計します。既存のインフラストラクチャを拡張している場合は、ブラウンフィールド戦略とグリーンフィールド戦略を融合させることもできます。

バッファキャッシュ

アクセス頻度が最も高いデータが保存されるメモリ領域。

ビジネス能力

価値を生み出すためにビジネスが行うこと (営業、カスタマーサービス、マーケティングなど)。マイクロサービスのアーキテクチャと開発の決定は、ビジネス能力によって推進できます。詳細については、AWSでのコンテナ化されたマイクロサービスの実行ホワイトペーパーの「ビジネス機能を中心に組織化」セクションを参照してください。

ビジネス継続性計画 (BCP)

大規模移行など、中断を伴うイベントが運用に与える潜在的な影響に対処し、ビジネスを迅速に再開できるようにする計画。

C

CAF

「AWS クラウド導入フレームワーク」を参照してください

カナリアデプロイ

エンドユーザーへのバージョンリリースを、時間をかけて段階的に行うこと。確信が持てたら新規バージョンをデプロイして、現在のバージョン全体を置き換えます。

CCoE

「Cloud Center of Excellence」を参照してください。

CDC

「変更データキャプチャ」を参照してください。

変更データキャプチャ (CDC)

データソース (データベーステーブルなど) の変更を追跡し、その変更に関するメタデータを記録するプロセス。CDC は、ターゲットシステムでの変更を監査またはレプリケートして同期を維持するなど、さまざまな目的に使用できます。

カオスエンジニアリング

障害や破壊的なイベントを意図的に導入して、システムの耐障害性をテストすること。AWS Fault Injection Service (AWS FIS) を使用して、 AWS ワークロードにストレスを与え、その応答を評価する実験を実行できます。

CI/CD

「継続的インテグレーションと継続的デリバリー」を参照してください。

分類

予測を生成するのに役立つ分類プロセス。分類問題の機械学習モデルは、離散値を予測します。離散値は、常に互いに区別されます。例えば、モデルがイメージ内に車があるかどうかを評価する必要がある場合があります。

クライアント側の暗号化

ターゲットが AWS のサービス 受信する前に、ローカルでデータを暗号化します。

Cloud Center of Excellence (CCoE)

クラウドのベストプラクティスの作成、リソースの移動、移行のタイムラインの確立、大規模変革を通じて組織をリードするなど、組織全体のクラウド導入の取り組みを推進する学際的なチーム。詳細については、 AWS クラウド エンタープライズ戦略ブログの CCoE 投稿を参照してください。

クラウドコンピューティング

リモートデータストレージと IoT デバイス管理に通常使用されるクラウドテクノロジー。クラウドコンピューティングは、一般的に、エッジコンピューティングに接続されています。

クラウド運用モデル

IT 組織において、1 つ以上のクラウド環境を構築、成熟、最適化するために使用される運用モデル。詳細については、「クラウド運用モデルの構築」を参照してください。

導入のクラウドステージ

組織が、 AWS クラウドへの移行時に通常実行する 4 つの段階。

- プロジェクト — 概念実証と学習を目的として、クラウド関連のプロジェクトをいくつか実行する
- 基礎固め — お客様のクラウドの導入を拡大するための基礎的な投資 (ランディングゾーンの作成、CCoE の定義、運用モデルの確立など)
- 移行 — 個々のアプリケーションの移行
- 再発明 — 製品とサービスの最適化、クラウドでのイノベーション

これらのステージは、 AWS クラウド エンタープライズ戦略ブログのブログ記事「クラウドファーストへのジャーニー」と「導入のステージ」で Stephen Orban によって定義されました。移行戦略との関連性については、 AWS 「移行準備ガイド」を参照してください。

CMDB

「構成管理データベース (CMDB)」を参照してください。

コードリポジトリ

ソースコードやその他の資産 (ドキュメント、サンプル、スクリプトなど) が保存され、バージョン管理プロセスを通じて更新される場所。一般的なクラウドリポジトリには、GitHub や Bitbucket Cloud があります。コードの各バージョンはブランチと呼ばれます。マイクロサービスの構造では、各リポジトリは 1 つの機能専用です。1 つの CI/CD パイプラインで複数のリポジトリを使用できます。

コールドキャッシュ

空である、または、かなり空きがある、もしくは、古いデータや無関係なデータが含まれているバッファキャッシュ。データベースインスタンスはメインメモリまたはディスクから読み取る必要があり、バッファキャッシュから読み取るよりも時間がかかるため、パフォーマンスに影響します。

コールドデータ

めったにアクセスされず、通常は過去のデータです。この種類のデータをクエリする場合、通常は低速なクエリでも問題ありません。このデータを低パフォーマンスで安価なストレージ階層またはクラスに移動すると、コストを削減することができます。

コンピュータビジョン (CV)

機械学習を使用してデジタルイメージやビデオといった、ビジュアル形式の情報を分析および抽出する AI の分野。例えば、Amazon SageMaker AI では、CV 用の画像処理アルゴリズムを利用できます。

設定ドリフト

ワークロードにおいて、設定が想定した状態から変化すること。これによって、ワークロードが非準拠になる可能性がありますが、この状態は、徐々に生じ、意図的なものではありません。

構成管理データベース (CMDB)

データベースとその IT 環境 (ハードウェアとソフトウェアの両方のコンポーネントとその設定を含む) に関する情報を保存、管理するリポジトリ。通常、CMDB のデータは、移行のポートフォリオの検出と分析の段階で使用します。

コンフォーマンスパック

コンプライアンスチェックとセキュリティチェックをカスタマイズするためにアセンブルできる AWS Config ルールと修復アクションのコレクション。YAML テンプレートを使用して、コンフォーマンスパックを AWS アカウント および リージョンの単一のエンティティとしてデプロイすることも、組織全体にデプロイすることもできます。詳細については、 AWS Config ドキュメントの「コンフォーマンスパック」を参照してください。

継続的インテグレーションと継続的デリバリー (CI/CD)

ソフトウェアリリースプロセスのソース、ビルド、テスト、ステージング、本番の各ステージを自動化するプロセス。CI/CD は一般的にパイプラインと呼ばれます。プロセスの自動化、生産性の向上、コード品質の向上、配信の加速化を可能にします。詳細については、「継続的デリバリーの利点」を参照してください。CD は継続的デプロイ (Continuous Deployment) の略語でもあります。詳細については「継続的デリバリーと継続的なデプロイ」を参照してください。

CV

「コンピュータビジョン」を参照してください。

D

保管中のデータ

ストレージ内にあるデータなど、常に自社のネットワーク内にあるデータ。

データ分類

ネットワーク内のデータを重要度と機密性に基づいて識別、分類するプロセス。データに適した保護および保持のコントロールを判断する際に役立つため、あらゆるサイバーセキュリティのリスク管理戦略において重要な要素です。データ分類は、 AWS Well-Architected フレームワークのセキュリティの柱のコンポーネントです。詳細については、「データ分類」を参照してください。

データドリフト

実稼働データと ML モデルのトレーニングに使用されたデータとの間に有意な差異が生じたり、入力データが時間の経過と共に有意に変化したりすることです。データドリフトは、ML モデル予測の全体的な品質、精度、公平性を低下させる可能性があります。

転送中のデータ

ネットワーク内 (ネットワークリソース間など) を活発に移動するデータ。

データメッシュ

非一元的で分散型のデータ所有権を持つとともに、一元的な管理およびガバナンスを行えるアーキテクチャフレームワーク。

データ最小化

厳密に必要なデータのみを収集し、処理するという原則。でデータ最小化を実践 AWS クラウドすることで、プライバシーリスク、コスト、分析のカーボンフットプリントを削減できます。

データ境界

AWS 環境内の一連の予防ガードレール。信頼できる ID のみが、期待されるネットワークから信頼できるリソースにアクセスできるようにします。詳細については、「 でのデータ境界の構築 AWS」を参照してください。

データの前処理

raw データをお客様の機械学習モデルで簡単に解析できる形式に変換すること。データの前処理とは、特定の列または行を削除して、欠落している、矛盾している、または重複する値に対処することを意味します。

データ出所

データの生成、送信、保存の方法など、データのライフサイクル全体を通じてデータの出所と履歴を追跡するプロセス。

データ件名

データを収集、処理している個人。

データウェアハウス

分析などのビジネスインテリジェンスをサポートするデータ管理システム。データウェアハウスには、一般的に、大量の履歴データが含まれており、多くの場合、それらはクエリや分析に使用されます。

データベース定義言語 (DDL)

データベース内のテーブルやオブジェクトの構造を作成または変更するためのステートメントまたはコマンド。

データベース操作言語 (DML)

データベース内の情報を変更 (挿入、更新、削除) するためのステートメントまたはコマンド。

DDL

「データベース定義言語」を参照してください。

ディープアンサンブル

予測のために複数の深層学習モデルを組み合わせます。ディープアンサンブルを使用して、より正確な予測を取得したり、予測の不確実性を推定したりできます。

深層学習

人工ニューラルネットワークの複数層を使用して、入力データと対象のターゲット変数の間のマッピングを識別する機械学習サブフィールド。

多層防御

一連のセキュリティメカニズムとコントロールをコンピュータネットワーク全体に層状に重ねて、ネットワークとその内部にあるデータの機密性、整合性、可用性を保護する情報セキュリティの手法。この戦略を採用するときは AWS、 AWS Organizations 構造の異なるレイヤーに複数のコントロールを追加して、リソースの安全性を確保します。たとえば、多層防御アプローチでは、多要素認証、ネットワークセグメンテーション、暗号化を組み合わせることができます。

委任管理者

では AWS Organizations、互換性のあるサービスが AWS メンバーアカウントを登録して組織のアカウントを管理し、そのサービスのアクセス許可を管理できます。このアカウントを、そのサービスの委任管理者と呼びます。詳細、および互換性のあるサービスの一覧は、 AWS

Organizations ドキュメントの「AWS Organizationsで使用できるサービス」を参照してください。

トラブルシューティング

アプリケーション、新機能、コードの修正をターゲットの環境で利用できるようにするプロセス。デプロイでは、コードベースに変更を施した後、アプリケーションの環境でそのコードベースを構築して実行します。

開発環境

「環境」を参照してください。

検出管理

イベントが発生したときに、検出、ログ記録、警告を行うように設計されたセキュリティコントロール。これらのコントロールは副次的な防衛手段であり、実行中の予防的コントロールをすり抜けたセキュリティイベントをユーザーに警告します。詳細については、「 AWSでのセキュリティコントロールの実装」の「検出的コントロール」を参照してください。

開発バリューストリームマッピング (DVSM)

ソフトウェア開発ライフサイクルのスピードと品質に悪影響を及ぼす制約を特定し、優先順位を付けるために使用されるプロセス。DVSM は、もともとリーンマニュファクチャリング・プラクティスのために設計されたバリューストリームマッピング・プロセスを拡張したものです。ソフトウェア開発プロセスを通じて価値を創造し、動かすために必要なステップとチームに焦点を当てています。

デジタルツイン

建物、工場、産業機器、生産ラインなど、現実世界のシステムを仮想的に表現したものです。デジタルツインは、予知保全、リモートモニタリング、生産最適化をサポートします。

ディメンションテーブル

スタースキーマにおいて、ファクトテーブルの定量データに関するデータ属性が含まれる小さいテーブル。ディメンションテーブルの属性は、通常、テキストフィールド、またはテキストのように扱える個別の数値で示されます。これらの属性は、一般的に、クエリの制約、フィルタリング、結果セットのラベル付けに使用されます。

ディザスタ

ワークロードまたはシステムが、導入されている主要な場所でのビジネス目標の達成を妨げるイベント。これらのイベントは、自然災害、技術的障害、または意図しない設定ミスやマルウェア攻撃などの人間の行動の結果である場合があります。

ディザスタリカバリ (DR)

ディザスタによるダウンタイムとデータ損失を最小限に抑えるための戦略とプロセス。詳細については、 AWS Well-Architected フレームワークの「 Disaster Recovery of Workloads on AWS: Recovery in the Cloud」を参照してください。

DML

「データベース操作言語」を参照してください。

ドメイン駆動型設計

各コンポーネントが提供している変化を続けるドメイン、またはコアビジネス目標にコンポーネントを接続して、複雑なソフトウェアシステムを開発するアプローチ。この概念は、エリック・エヴァンスの著書、Domain-Driven Design: Tackling Complexity in the Heart of Software (ドメイン駆動設計:ソフトウェアの中心における複雑さへの取り組み) で紹介されています (ボストン: Addison-Wesley Professional、2003)。strangler fig パターンでドメイン駆動型設計を使用する方法の詳細については、「コンテナと Amazon API Gateway を使用して、従来の Microsoft ASP.NET (ASMX) ウェブサービスを段階的にモダナイズ」を参照してください。

DR

「ディザスタリカバリ」を参照してください。

ドリフト検出

ベースライン設定からの偏差を追跡します。たとえば、 AWS CloudFormation を使用してシステムリソースのドリフトを検出したり、 を使用して AWS Control Tower 、ガバナンス要件への準拠に影響するランディングゾーンの変更を検出したりできます。

DVSM

「開発バリューストリームマッピング」を参照してください。

E

EDA

「探索的データ分析」を参照してください。

EDI

「電子データ交換」を参照してください。

エッジコンピューティング

IoT ネットワークのエッジにあるスマートデバイスの計算能力を高めるテクノロジー。クラウドコンピューティングと比較すると、エッジコンピューティングは通信レイテンシーを短縮し、応答時間を改善できます。

電子データ交換 (EDI)

組織間で行う、ビジネスドキュメントの自動交換。詳細については、「電子データ交換とは」を参照してください。

暗号化

人間が読み取り可能なプレーンテキストデータを暗号文に変換するコンピューティング処理。

暗号化キー

暗号化アルゴリズムが生成した、ランダム化されたビットからなる暗号文字列。キーの長さは決まっておらず、各キーは予測できないように、一意になるように設計されています。

エンディアン

コンピュータメモリにバイトが格納される順序。ビッグエンディアンシステムでは、最上位バイトが最初に格納されます。リトルエンディアンシステムでは、最下位バイトが最初に格納されます。

エンドポイント

「サービスエンドポイント」を参照してください。

エンドポイントサービス

仮想プライベートクラウド (VPC) 内でホストして、他のユーザーと共有できるサービス。を使用してエンドポイントサービスを作成し AWS PrivateLink 、他の AWS アカウント または AWS Identity and Access Management (IAM) プリンシパルにアクセス許可を付与できます。これらのアカウントまたはプリンシパルは、インターフェイス VPC エンドポイントを作成することで、エンドポイントサービスにプライベートに接続できます。詳細については、Amazon Virtual Private Cloud (Amazon VPC) ドキュメントの「エンドポイントサービスを作成する」を参照してください。

エンタープライズリソースプランニング (ERP)

エンタープライズの主要なビジネスプロセス (会計、MES、プロジェクト管理など) を自動化および管理するシステム。

エンベロープ暗号化

暗号化キーを、別の暗号化キーを使用して暗号化するプロセス。詳細については、 AWS Key Management Service (AWS KMS) ドキュメントの「エンベロープ暗号化」を参照してください。

環境

実行中のアプリケーションのインスタンス。クラウドコンピューティングにおける一般的な環境の種類は以下のとおりです。

- 開発環境 — アプリケーションのメンテナンスを担当するコアチームのみが使用できる、実行中のアプリケーションのインスタンス。開発環境は、上位の環境に昇格させる変更をテストするときに使用します。このタイプの環境は、テスト環境と呼ばれることもあります。
- 下位環境 — 初期ビルドやテストに使用される環境など、アプリケーションのすべての開発環境。
- 本番環境 — エンドユーザーがアクセスできる、実行中のアプリケーションのインスタンス。CI/CD パイプラインでは、本番環境が最後のデプロイ環境になります。
- 上位環境 — コア開発チーム以外のユーザーがアクセスできるすべての環境。これには、本番環境、本番前環境、ユーザー承認テスト環境などが含まれます。

エピック

アジャイル方法論で、お客様の作業の整理と優先順位付けに役立つ機能カテゴリ。エピックでは、要件と実装タスクの概要についてハイレベルな説明を提供します。例えば、 AWS CAF セキュリティエピックには、ID とアクセスの管理、検出コントロール、インフラストラクチャセキュリティ、データ保護、インシデント対応が含まれます。 AWS 移行戦略のエピックの詳細については、プログラム実装ガイドを参照してください。

ERP

「エンタープライズリソース計画」を参照してください。

探索的データ分析 (EDA)

データセットを分析してその主な特性を理解するプロセス。お客様は、データを収集または集計してから、パターンの検出、異常の検出、および前提条件のチェックのための初期調査を実行します。EDA は、統計の概要を計算し、データの可視化を作成することによって実行されます。

F

ファクトテーブル

スタースキーマの中央にあるテーブル。ビジネスオペレーションに関する定量的データが保存されます。一般的に、ファクトテーブルは、2 種類の列で構成されます。1 つは測定値が含まれる列、もう 1 つはディメンションテーブルへの外部キーが含まれる列です。

フェイルファスト

開発ライフサイクルを短縮するために、頻繁かつ段階的にテストを行う哲学であり、アジャイルアプローチでは、この考え方がきわめて重要です。

障害分離境界

では AWS クラウド、アベイラビリティーゾーン AWS リージョン、コントロールプレーン、データプレーンなどの境界で、障害の影響を制限し、ワークロードの耐障害性を向上させるのに役立ちます。詳細については、「AWS 障害分離境界」を参照してください。

機能ブランチ

「ブランチ」を参照してください。

特徴量

お客様が予測に使用する入力データ。例えば、製造コンテキストでは、特徴量は製造ラインから定期的にキャプチャされるイメージの可能性もあります。

特徴量重要度

モデルの予測に対する特徴量の重要性。これは通常、Shapley Additive Deskonations (SHAP) や積分勾配など、さまざまな手法で計算できる数値スコアで表されます。詳細については、「 を使用した機械学習モデルの解釈可能性 AWS」を参照してください。

機能変換

追加のソースによるデータのエンリッチ化、値のスケーリング、単一のデータフィールドからの複数の情報セットの抽出など、機械学習プロセスのデータを最適化すること。これにより、機械学習モデルはデータの恩恵を受けることができます。例えば、「2021-05-27 00:15:37」の日付を「2021 年」、「5 月」、「木」、「15」に分解すると、学習アルゴリズムがさまざまなデータコンポーネントに関連する微妙に異なるパターンを学習するのに役立ちます。

数ショットプロンプト

LLM に、タスクと望ましい出力を示す例を少数提示した後に、類似のタスクを実行させること。この手法は、プロンプトに記述された例 (ショット) からモデルが学習する「インコンテキスト学

習」の一種です。数ショットプロンプトは、特定のフォーマット、推論、専門知識が必要なタスクに効果的です。「ゼロショットプロンプト」も参照してください。

FGAC

「きめ細かなアクセス制御」を参照してください。

きめ細かなアクセス制御 (FGAC)

複数の条件を使用してアクセス要求を許可または拒否すること。

フラッシュカット移行

変更データのキャプチャによる継続的なデータ複製を利用して、段階的なアプローチではなく、可能な限り短時間でデータを移行するデータベース移行方法。目的はダウンタイムを最小限に抑えることです。

FM

「基盤モデル」を参照してください。

基盤モデル (FM)

大規模な深層学習ニューラルネットワークであり、一般化およびラベル付けされていないデータからなる大規模データセットでトレーニングされています。FM により、言語理解、テキストおよび画像生成、自然言語での会話といった、一般的な各種タスクを実行できます。詳細については、「基盤モデルとは何ですか?」を参照してください。

G

生成 AI

AI モデルのサブセット。大量のデータでトレーニングされており、シンプルなテキストプロンプトを使用して、画像、動画、テキスト、オーディオなどの新しいコンテンツやアーティファクトを作成できます。詳細については、「生成 AI とは何ですか?」を参照してください。

ジオブロッキング

「地理的制限」を参照してください。

地理的制限 (ジオブロッキング)

特定の国のユーザーがコンテンツ配信にアクセスできないようにするための、Amazon CloudFront のオプション。アクセスを許可する国と禁止する国は、許可リストまたは禁止リスト

を使って指定します。詳細については、CloudFront ドキュメントの「コンテンツの地理的ディストリビューションの制限」を参照してください。

Gitflow ワークフロー

下位環境と上位環境が、ソースコードリポジトリでそれぞれ異なるブランチを使用する方法。Gitflow ワークフローは古いと見なされている方法であり、トランクベースのワークフローは推奨されている新しい方法です。

ゴールデンイメージ

システムまたはソフトウェアのスナップショットであり、システムまたはソフトウェアの新規インスタンスをデプロイするテンプレートとして使用されます。製造の例で言えば、ゴールデンイメージを使用すると、複数のデバイスにソフトウェアをプロビジョニングして、デバイス製造オペレーションの速度、スケーラビリティ、生産性を向上させることができます。

グリーンフィールド戦略

新しい環境に既存のインフラストラクチャが存在しないこと。システムアーキテクチャにグリーンフィールド戦略を導入する場合、既存のインフラストラクチャ (別名ブラウンフィールド) との互換性の制約を受けることなく、あらゆる新しいテクノロジーを選択できます。既存のインフラストラクチャを拡張している場合は、ブラウンフィールド戦略とグリーンフィールド戦略を融合させることもできます。

ガードレール

組織単位 (OU) 全般のリソース、ポリシー、コンプライアンスを管理するのに役立つ概略的なルール。予防ガードレールは、コンプライアンス基準に一致するようにポリシーを実施します。これらは、サービスコントロールポリシーと IAM アクセス許可の境界を使用して実装されます。検出ガードレールは、ポリシー違反やコンプライアンス上の問題を検出し、修復のためのアラートを発信します。これらは AWS Config、、 AWS Security Hub CSPM、Amazon GuardDuty、、 AWS Trusted Advisor Amazon Inspector、およびカスタム AWS Lambda チェックを使用して実装されます。

H

HA

「高可用性」を参照してください。

異種混在データベースの移行

別のデータベースエンジンを使用するターゲットデータベースへお客様の出典データベースの移行 (例えば、Oracle から Amazon Aurora)。異種間移行は通常、アーキテクチャの再設計作業の一部であり、スキーマの変換は複雑なタスクになる可能性があります。AWS は、スキーマの変換に役立つ AWS SCTを提供します。

高可用性 (HA)

課題や災害が発生した場合に、介入なしにワークロードを継続的に運用できること。HA システムは、自動的にフェイルオーバーし、一貫して高品質のパフォーマンスを提供し、パフォーマンスへの影響を最小限に抑えながらさまざまな負荷や障害を処理するように設計されています。

ヒストリアンのモダナイゼーション

製造業のニーズによりよく応えるために、オペレーションテクノロジー (OT) システムをモダナイズし、アップグレードするためのアプローチ。ヒストリアンは、工場内のさまざまなソースからデータを収集して保存するために使用されるデータベースの一種です。

ホールドアウトデータ

機械学習モデルのトレーニング用データセットから保留される、ラベル付き履歴データの一部。ホールドアウトデータを使用すると、モデル予測をホールドアウトデータと比較して、モデルのパフォーマンスを評価できます。

同種データベースの移行

お客様の出典データベースを、同じデータベースエンジンを共有するターゲットデータベース (Microsoft SQL Server から Amazon RDS for SQL Server など) に移行する。同種間移行は、通常、リホストまたはリプラットフォーム化の作業の一部です。ネイティブデータベースユーティリティを使用して、スキーマを移行できます。

ホットデータ

リアルタイムデータや最近の翻訳データなど、頻繁にアクセスされるデータ。通常、このデータには高速なクエリ応答を提供する高性能なストレージ階層またはクラスが必要です。

ホットフィックス

本番環境の重大な問題を修正するために緊急で配布されるプログラム。緊急性が高いため、通常の DevOps のリリースワークフローからは外れた形で実施されます。

ハイパーケア期間

カットオーバー直後、移行したアプリケーションを移行チームがクラウドで管理、監視して問題に対処する期間。通常、この期間は 1～4 日です。ハイパーケア期間が終了すると、アプリケーションに対する責任は一般的に移行チームからクラウドオペレーションチームに移ります。

I

IaC

「Infrastructure as Code」を参照してください。

ID ベースのポリシー

AWS クラウド 環境内のアクセス許可を定義する 1 つ以上の IAM プリンシパルにアタッチされたポリシー。

アイドル状態のアプリケーション

90 日間の平均的な CPU およびメモリ使用率が 5～20% のアプリケーション。移行プロジェクトでは、これらのアプリケーションを廃止するか、オンプレミスに保持するのが一般的です。

IIoT

「インダストリアル IoT」を参照してください。

イミュータブルインフラストラクチャ

既存インフラストラクチャの更新、パッチ適用、変更などを行わずに、本番環境ワークロードに使用する新規インフラストラクチャをデプロイするモデル。本質的に、イミュータブルインフラストラクチャは、ミュータブルインフラストラクチャよりも一貫性、信頼性、予測性に優れています。詳細については、 AWS Well-Architected フレームワークにある「イミュータブルインフラストラクチャを使用してデプロイする」のベストプラクティスを参照してください。

インバウンド (受信) VPC

AWS マルチアカウントアーキテクチャでは、アプリケーションの外部からネットワーク接続を受け入れ、検査し、ルーティングする VPC。AWS Security Reference Architecture では、アプリケーションとより広範なインターネット間の双方向のインターフェイスを保護するために、インバウンド、アウトバウンド、インスペクションの各 VPC を使用してネットワークアカウントを設定することを推奨しています。

増分移行

アプリケーションを 1 回ですべてカットオーバーするのではなく、小さい要素に分けて移行するカットオーバー戦略。例えば、最初は少数のマイクロサービスまたはユーザーのみを新しいシステムに移行する場合があります。すべてが正常に機能することを確認できたら、残りのマイクロサービスやユーザーを段階的に移行し、レガシーシステムを廃止できるようにします。この戦略により、大規模な移行に伴うリスクが軽減されます。

インダストリー 4.0

2016 年に Klaus Schwab 氏が提唱した用語で、接続、リアルタイムデータ、オートメーション、分析、AI/ML の進歩による、ビジネスプロセスのモダナイズを意味します。

インフラストラクチャ

アプリケーションの環境に含まれるすべてのリソースとアセット。

Infrastructure as Code (IaC)

アプリケーションのインフラストラクチャを一連の設定ファイルを使用してプロビジョニングし、管理するプロセス。IaC は、新しい環境を再現可能で信頼性が高く、一貫性のあるものにするため、インフラストラクチャを一元的に管理し、リソースを標準化し、スケールを迅速に行えるように設計されています。

インダストリアル IoT (IIoT)

製造、エネルギー、自動車、ヘルスケア、ライフサイエンス、農業などの産業部門におけるインターネットに接続されたセンサーやデバイスの使用。詳細については、「インダストリアル IoT (IIoT) デジタルトランスフォーメーション戦略の構築」を参照してください。

インスペクション VPC

AWS マルチアカウントアーキテクチャでは、VPC (同一または異なる 内 AWS リージョン)、インターネット、オンプレミスネットワーク間のネットワークトラフィックの検査を管理する一元化された VPCs。AWS Security Reference Architecture では、アプリケーションとより広範なインターネット間の双方向のインターフェイスを保護するために、インバウンド、アウトバウンド、インスペクションの各 VPC を使用してネットワークアカウントを設定することを推奨しています。

IoT

インターネットまたはローカル通信ネットワークを介して他のデバイスやシステムと通信する、センサーまたはプロセッサが組み込まれた接続済み物理オブジェクトのネットワーク。詳細については、「IoT とは」を参照してください。

解釈可能性

機械学習モデルの特性で、モデルの予測がその入力にどのように依存するかを人間が理解できる度合いを表します。詳細については、「 を使用した機械学習モデルの解釈可能性 AWS」を参照してください。

IoT

「IoT」を参照してください。

IT 情報ライブラリ (ITIL)

IT サービスを提供し、これらのサービスをビジネス要件に合わせるための一連のベストプラクティス。ITIL は ITSM の基盤を提供します。

IT サービス管理 (ITSM)

組織の IT サービスの設計、実装、管理、およびサポートに関連する活動。クラウドオペレーションと ITSM ツールの統合については、オペレーション統合ガイドを参照してください。

ITIL

「IT 情報ライブラリ」を参照してください。

ITSM

「IT サービス管理」を参照してください。

L

ラベルベースアクセス制御 (LBAC)

強制アクセス制御 (MAC) の実装で、ユーザーとデータ自体にそれぞれセキュリティラベル値が明示的に割り当てられます。ユーザーセキュリティラベルとデータセキュリティラベルが交差する部分によって、ユーザーに表示される行と列が決まります。

ランディングゾーン

ランディングゾーンは、スケーラブルで安全な、適切に設計されたマルチアカウント AWS 環境です。これは、組織がセキュリティおよびインフラストラクチャ環境に自信を持ってワークロードとアプリケーションを迅速に起動してデプロイできる出発点です。ランディングゾーンの詳細については、「安全でスケーラブルなマルチアカウント AWS 環境のセットアップ」を参照してください。

大規模言語モデル (LLM)

大量のデータで事前トレーニングされた深層学習 AI モデル。LLM では、質問への回答、ドキュメントの要約、他言語へのテキスト翻訳、文を完成させるなど、さまざまなタスクを実行できます。詳細については、「大規模言語モデル (LLM) とは何ですか?」を参照してください。

大規模な移行

300 台以上のサーバの移行。

LBAC

「ラベルベースアクセス制御」を参照してください。

最小特権

タスクの実行には必要最低限の権限を付与するという、セキュリティのベストプラクティス。詳細については、IAM ドキュメントの「最小特権アクセス許可を適用する」を参照してください。

リフトアンドシフト

「7 Rs」を参照してください。

リトルエンディアンシステム

最下位バイトを最初に格納するシステム。「エンディアン性」もご覧ください。

LLM

「大規模言語モデル」を参照してください。

下位環境

「環境」を参照してください。

M

機械学習 (ML)

パターン認識と学習にアルゴリズムと手法を使用する人工知能の一種。ML は、モノのインターネット (IoT) データなどの記録されたデータを分析して学習し、パターンに基づく統計モデルを生成します。詳細については、「機械学習」を参照してください。

メインブランチ

「ブランチ」を参照してください。

マルウェア

コンピュータのセキュリティやプライバシーを侵害するように設計されたソフトウェア。マルウェアは、コンピュータシステムの中断、機密情報の漏洩、不正アクセスを招く可能性があります。マルウェアの例には、ウイルス、ワーム、ランサムウェア、トロイの木馬、スパイウェア、キーロガーなどがあります。

マネージドサービス

AWS のサービス はインフラストラクチャレイヤー、オペレーティングシステム、プラットフォーム AWS を運用し、エンドポイントにアクセスしてデータを保存および取得します。マネージドサービスの例として、Amazon Simple Storage Service (Amazon S3) と Amazon DynamoDB が挙げられます。このサービスは、抽象化されたサービスとも呼ばれます。

製造実行システム (MES)

生産プロセスを追跡、モニタリング、文書化、制御するソフトウェアシステムであり、工場では、これによって、原材料から製品を完成させます。

MAP

「Migration Acceleration Program」を参照してください。

メカニズム

ツールを作成してその導入を推進し、導入結果を調べて調整を行うための包括的なプロセス。メカニズムとは、運用中にそれ自体を強化し改善するサイクルを意味します。詳細については、AWS 「 Well-Architected フレームワーク」の「メカニズムの構築」を参照してください。

メンバーアカウント

組織の一部である管理アカウント AWS アカウント 以外のすべて AWS Organizations。 アカウントが 組織のメンバーになることができるのは、一度に 1 つのみです。

MES

「製造実行システム」を参照してください。

Message Queuing Telemetry Transport (MQTT)

発行/サブスクライブのパターンに基づく、軽量のマシンツーマシン (M2M) 通信プロトコルであり、リソースに限りのある IoT デバイスに使用されます。

マイクロサービス

明確に定義された API を介して通信し、通常は小規模な自己完結型のチームが所有する、小規模で独立したサービスです。例えば、保険システムには、販売やマーケティングなどのビジネス

機能、または購買、請求、分析などのサブドメインにマッピングするマイクロサービスが含まれる場合があります。マイクロサービスの利点には、俊敏性、柔軟なスケーリング、容易なデプロイ、再利用可能なコード、回復力などがあります。詳細については、AWS「サーバーレスサービスを使用したマイクロサービスの統合」を参照してください。

マイクロサービスアーキテクチャ

各アプリケーションプロセスをマイクロサービスとして実行する独立したコンポーネントを使用してアプリケーションを構築するアプローチ。これらのマイクロサービスは、軽量 API を使用して、明確に定義されたインターフェイスを介して通信します。このアーキテクチャの各マイクロサービスは、アプリケーションの特定の機能に対する需要を満たすように更新、デプロイ、およびスケーリングできます。詳細については、「 でのマイクロサービスの実装 AWS」を参照してください。

Migration Acceleration Program (MAP)

組織がクラウドに移行するための強力な運用基盤を構築し、移行の初期コストを相殺するのに役立つコンサルティングサポート、トレーニング、サービスを提供する AWS プログラム。MAP には、組織的な方法でレガシー移行を実行するための移行方法論と、一般的な移行シナリオを自動化および高速化する一連のツールが含まれています。

大規模な移行

アプリケーションポートフォリオの大部分を次々にクラウドに移行し、各ウェーブでより多くのアプリケーションを高速に移動させるプロセス。この段階では、以前の段階から学んだベストプラクティスと教訓を使用して、移行ファクトリー チーム、ツール、プロセスのうち、オートメーションとアジャイルデリバリーによってワークロードの移行を合理化します。これは、AWS 移行戦略 の第 3 段階です。

移行ファクトリー

自動化された俊敏性のあるアプローチにより、ワークロードの移行を合理化する部門横断的なチーム。移行ファクトリーチームには、通常、運用、ビジネスアナリストおよび所有者、移行エンジニア、デベロッパー、およびスプリントで作業する DevOps プロフェッショナルが含まれます。エンタープライズアプリケーションポートフォリオの 20～50% は、ファクトリーのアプローチによって最適化できる反復パターンで構成されています。詳細については、このコンテンツセットの移行ファクトリーに関する解説とCloud Migration Factory ガイドを参照してください。

移行メタデータ

移行を完了するために必要なアプリケーションおよびサーバーに関する情報。移行パターンごとに、異なる一連の移行メタデータが必要です。移行メタデータの例としては、ターゲットサブネット、セキュリティグループ、 AWS アカウントなどがあります。

移行パターン

移行戦略、移行先、および使用する移行アプリケーションまたはサービスを詳述する、反復可能な移行タスク。例: AWS Application Migration Service を使用して Amazon EC2 への移行をリホストします。

Migration Portfolio Assessment (MPA)

オンラインツール。これによって、 AWS クラウドに移行するビジネスケースの検証に必要な情報を得られます。MPA は、詳細なポートフォリオ評価 (サーバーの適切なサイジング、価格設定、TCO 比較、移行コスト分析) および移行プラン (アプリケーションデータの分析とデータ収集、アプリケーションのグループ化、移行の優先順位付け、およびウェーブプランニング) を提供します。MPA ツール (ログインが必要) は、すべての AWS コンサルタントと APN パートナーコンサルタントが無料で利用できます。

移行準備状況評価 (MRA)

AWS CAF を使用して、組織のクラウド準備状況に関するインサイトを取得し、長所と短所を特定し、特定されたギャップを埋めるためのアクションプランを構築するプロセス。詳細については、移行準備状況ガイドを参照してください。MRA は、AWS 移行戦略の第一段階です。

移行戦略

ワークロードを AWS クラウドに移行するために使用するアプローチ。詳細については、この用語集の 7 Rs エントリと、「組織を動員して大規模な移行を加速する」を参照してください。

ML

「機械学習」を参照してください。

モダナイゼーション

古い (レガシーまたはモノリシック) アプリケーションとそのインフラストラクチャをクラウド内の俊敏で弾力性のある高可用性システムに変換して、コストを削減し、効率を高め、イノベーションを活用します。詳細については、「AWS クラウドでのアプリケーションのモダナイズ戦略」を参照してください。

モダナイゼーション準備状況評価

組織のアプリケーションのモダナイゼーションの準備状況を判断し、利点、リスク、依存関係を特定し、組織がこれらのアプリケーションの将来の状態をどの程度適切にサポートできるかを決定するのに役立つ評価。評価の結果として、ターゲットアーキテクチャのブループリント、モダナイゼーションプロセスの開発段階とマイルストーンを詳述したロードマップ、特定されたギャップに対処するためのアクションプランが得られます。詳細については、「AWS クラウドでのアプリケーションのモダナイゼーションの準備状況を評価する」を参照してください。

モノリシックアプリケーション (モノリス)

緊密に結合されたプロセスを持つ単一のサービスとして実行されるアプリケーション。モノリシックアプリケーションにはいくつかの欠点があります。1 つのアプリケーション機能エクスペリエンスの需要が急増する場合は、アーキテクチャ全体をスケーリングする必要があります。モノリシックアプリケーションの特徴を追加または改善することは、コードベースが大きくなると複雑になります。これらの問題に対処するには、マイクロサービスアーキテクチャを使用できます。詳細については、「モノリスをマイクロサービスに分解する」を参照してください。

MPA

「Migration Portfolio Assessment」を参照してください。

MQTT

「Message Queuing Telemetry Transport」を参照してください。

多クラス分類

複数のクラスの予測を生成するプロセス (2 つ以上の結果の 1 つを予測します)。例えば、機械学習モデルが、「この製品は書籍、自動車、電話のいずれですか?」 または、「このお客様にとって最も関心のある商品のカテゴリはどれですか?」と聞くかもしれません。

ミュータブルなインフラストラクチャ

本番ワークロードに使用する既存のインフラストラクチャを更新および変更するためのモデル。Well-Architected AWS フレームワークでは、一貫性、信頼性、予測可能性を向上させるために、イミュータブルインフラストラクチャの使用をベストプラクティスとして推奨しています。

O

OAC

「オリジンアクセス制御」を参照してください。

OAI

「オリジンアクセスアイデンティティ」を参照してください。

OCM

「組織変更管理」を参照してください。

オフライン移行

移行プロセス中にソースワークロードを停止させる移行方法。この方法はダウンタイムが長くなるため、通常は重要ではない小規模なワークロードに使用されます。

OI

「オペレーション統合」を参照してください。

Ola

「オペレーショナルレベルアグリーメント」を参照してください。

オンライン移行

ソースワークロードをオフラインにせずにターゲットシステムにコピーする移行方法。ワークロードに接続されているアプリケーションは、移行中も動作し続けることができます。この方法はダウンタイムがゼロから最小限で済むため、通常は重要な本番稼働環境のワークロードに使用されます。

OPC-UA

「Open Process Communications - Unified Architecture」を参照してください。

Open Process Communications - Unified Architecture (OPC-UA)

産業オートメーション用のマシンツーマシン (M2M) 通信プロトコル。OPC-UA により、相互運用の際に、データ暗号化、認証、認可の各スキームを標準化できます。

オペレーショナルレベルアグリーメント (OLA)

サービスレベルアグリーメント (SLA) をサポートするために、どの機能的 IT グループが互いに提供することを約束するかを明確にする契約。

運用準備状況レビュー (ORR)

質問と関連するベストプラクティスのチェックリスト。インシデントや起こり得る障害を理解、評価、防止したり、その範囲を縮小したりする際に役立ちます。詳細については、 AWS Well-Architected フレームワークの「Operational Readiness Reviews (ORR)」を参照してください。

運用テクノロジー (OT)

産業オペレーション、機器、インフラストラクチャを制御するために物理環境と連携させるハードウェアおよびソフトウェアシステム。製造分野では、Industry 4.0 への変革を進める上で、OT と情報技術 (IT) システムの統合に焦点が当てられています。

オペレーション統合 (OI)

クラウドでオペレーションをモダナイズするプロセスには、準備計画、オートメーション、統合が含まれます。詳細については、オペレーション統合ガイドを参照してください。

組織の証跡

組織 AWS アカウント 内のすべての のすべてのイベント AWS CloudTrail を ログに記録する によって作成された証跡 AWS Organizations。証跡は、組織に含まれている各 AWS アカウント に作成され、各アカウントのアクティビティを追跡します。詳細については、CloudTrail ドキュメントの「組織の証跡の作成」を参照してください。

組織変更管理 (OCM)

人材、文化、リーダーシップの観点から、主要な破壊的なビジネス変革を管理するためのフレームワーク。OCM は、変化の導入を加速し、移行問題に対処し、文化や組織の変化を推進することで、組織が新しいシステムと戦略の準備と移行するのを支援します。 AWS 移行戦略では、クラウド導入プロジェクトに必要な変化のスピードにより、このフレームワークは人材アクセラレーションと呼ばれます。詳細については、OCM ガイドを参照してください。

オリジンアクセス制御 (OAC)

Amazon Simple Storage Service (Amazon S3) コンテンツを保護するための、CloudFront のアクセス制限の強化オプション。OAC は AWS リージョン、すべての S3 バケット、 AWS KMS (SSE-KMS) によるサーバー側の暗号化、S3 バケットへの動的 PUT および DELETEリクエストをサポートします。

オリジンアクセスアイデンティティ (OAI)

CloudFront の、Amazon S3 コンテンツを保護するためのアクセス制限オプション。OAI を使用すると、CloudFront が、Amazon S3 に認証可能なプリンシパルを作成します。認証されたプリンシパルは、S3 バケット内のコンテンツに、特定の CloudFront ディストリビューションを介してのみアクセスできます。OAC も併せて参照してください。OAC では、より詳細な、強化されたアクセス制御が可能です。

ORR

「運用準備状況レビュー」を参照してください。

OT

「運用テクノロジー」を参照してください。

アウトバウンド (送信) VPC

AWS マルチアカウントアーキテクチャでは、アプリケーション内から開始されたネットワーク接続を処理する VPC。AWS Security Reference Architecture では、アプリケーションとより広範なインターネット間の双方向のインターフェイスを保護するために、インバウンド、アウトバウンド、インスペクションの各 VPC を使用してネットワークアカウントを設定することを推奨しています。

P

アクセス許可の境界

ユーザーまたはロールが使用できるアクセス許可の上限を設定する、IAM プリンシパルにアタッチされる IAM 管理ポリシー。詳細については、IAM ドキュメントのアクセス許可の境界を参照してください。

個人を特定できる情報 (PII)

直接閲覧した場合、または他の関連データと組み合わせた場合に、個人の身元を合理的に推測するために使用できる情報。PII の例には、氏名、住所、連絡先情報などがあります。

PII

「個人を特定できる情報」を参照してください。

プレイブック

クラウドでのコアオペレーション機能の提供など、移行に関連する作業を取り込む、事前定義された一連のステップ。プレイブックは、スクリプト、自動ランブック、またはお客様のモダナイズされた環境を運用するために必要なプロセスや手順の要約などの形式をとることができます。

PLC

「プログラマブルロジックコントローラー」を参照してください。

PLM

「製品ライフサイクル管理」を参照してください。

ポリシー

次の操作を可能にするオブジェクト: アクセス許可を定義する (ID ベースのポリシーを参照)。アクセス条件を指定する (リソースベースのポリシーを参照)。 AWS Organizations の組織における全アカウントにアクセス許可の上限を定義する (サービスコントロールポリシーを参照)。

多言語の永続性

データアクセスパターンやその他の要件に基づいて、マイクロサービスのデータストレージテクノロジーを個別に選択します。マイクロサービスが同じデータストレージテクノロジーを使用している場合、実装上の問題が発生したり、パフォーマンスが低下する可能性があります。マイクロサービスは、要件に最も適合したデータストアを使用すると、より簡単に実装でき、パフォーマンスとスケーラビリティが向上します。

ポートフォリオ評価

移行を計画するために、アプリケーションポートフォリオの検出、分析、優先順位付けを行うプロセス。詳細については、「移行の準備状況の評価」を参照してください。

述語

`true` または `false` を返すためのクエリ条件。一般的に、`WHERE` 句に記述されます。

述語プッシュダウン

データベースクエリを最適化する手法。これによって、転送前にクエリ内のデータをフィルタリングします。この手法を取ると、リレーショナルデータベースから取得し処理する必要のあるデータの量が減少するため、クエリのパフォーマンスが向上します。

予防的コントロール

イベントの発生を防ぐように設計されたセキュリティコントロール。このコントロールは、ネットワークへの不正アクセスや好ましくない変更を防ぐ最前線の防御です。詳細については、「AWSでのセキュリティコントロールの実装」の「予防的コントロール」を参照してください。

プリンシパル

アクションを実行し AWS 、リソースにアクセスできる のエンティティ。このエンティティは通常、、IAM AWS アカウントロール、または ユーザーのルートユーザーです。詳細については、IAM ドキュメントの「ロールに関する用語と概念」にあるプリンシパルを参照してください。

プライバシーバイデザイン

開発プロセス全体を通してプライバシーが考慮されているシステムエンジニアリングのアプローチ。

プライベートホストゾーン

1 つ以上の VPC 内のドメインとそのサブドメインへの DNS クエリに対し、Amazon Route 53 がどのように応答するかに関する情報を保持するコンテナ。詳細については、Route 53 ドキュメントの「プライベートホストゾーンの使用」を参照してください。

プロアクティブコントロール

非準拠リソースのデプロイ防止を目的としたセキュリティコントロール。このコントロールにより、プロビジョニング前にリソースをスキャンします。コントロールに準拠していないリソースは、プロビジョニングされません。詳細については、 AWS Control Tower ドキュメントの「コントロールリファレンスガイド」および「 セキュリティコントロールの実装」の「プロアクティブコントロール」を参照してください。 AWS

製品ライフサイクル管理 (PLM)

製品の設計、開発、発売から、成長、成熟、衰退、廃棄に至る、製品のライフサイクル全体を通してデータとプロセスを管理すること。

本番環境

「環境」を参照してください。

プログラマブルロジックコントローラー (PLC)

製造分野で使用される、信頼性と適応性に優れたコンピュータであり、これによって、マシンをモニタリングするとともに、製造プロセスを自動化します。

プロンプトチェイニング

1 つの LLM プロンプトによる出力を次のプロンプトの入力に使用して、より良いレスポンスを生成します。この手法を使用すると、複雑なタスクをサブタスクに分割したり、事前レスポンスを繰り返し改良または拡張したりできます。これによって、モデルのレスポンスの精度と関連性が向上し、粒度の高いパーソナライズされた結果を得られます。

仮名化

データセット内の個人識別子をプレースホルダー値に置き換えるプロセス。仮名化は個人のプライバシー保護に役立ちます。仮名化されたデータは、依然として個人データとみなされます。

発行/サブスクライブ (pub/sub)

マイクロサービス間の非同期通信を可能にするパターン。これにより、スケーラビリティと応答性を向上させます。例えば、マイクロサービスベースの MES の場合、マイクロサービスは、他のマイクロサービスがサブスクライブ可能なチャネルにイベントメッセージを発行できます。このシステムでは、発行サービスの変更なしに、新規マイクロサービスを追加できます。

Q

クエリプラン

手順などの一連のステップであり、SQL リレーショナルデータベースシステムのデータにアクセスするために使用されます。

クエリプランのリグレッション

データベースサービスのオプティマイザーが、データベース環境に特定の変更が加えられる前に選択されたプランよりも最適性の低いプランを選択すること。これは、統計、制限事項、環境設定、クエリパラメータのバインディングの変更、およびデータベースエンジンの更新などが原因である可能性があります。

R

RACI マトリックス

「実行責任者、説明責任者、協業先、報告先 (RACI)」を参照してください。

RAG

「検索拡張生成」を参照してください。

ランサムウェア

決済が完了するまでコンピュータシステムまたはデータへのアクセスをブロックするように設計された、悪意のあるソフトウェア。

RASCI マトリックス

「実行責任者、説明責任者、協業先、報告先 (RACI)」を参照してください。

RCAC

「行と列のアクセス制御」を参照してください。

リードレプリカ

読み取り専用に使用されるデータベースのコピー。クエリをリードレプリカにルーティングして、プライマリデータベースへの負荷を軽減できます。

リアーキテクト

「7 Rs」を参照してください。

目標復旧時点 (RPO)

最後のデータリカバリポイントからの最大許容時間です。これにより、最後の回復時点からサービスが中断されるまでの間に許容できるデータ損失の程度が決まります。

目標復旧時間 (RTO)

サービスの中断から復旧までの最大許容遅延時間。

リファクタリング

「7 Rs」を参照してください。

リージョン

地理的エリア内の AWS リソースのコレクション。各 AWS リージョン は、耐障害性、安定性、耐障害性を提供するために、他の から分離され、独立しています。詳細については、「アカウントが使用できる AWS リージョン を指定する」を参照してください。

リグレッション

数値を予測する機械学習手法。例えば、「この家はどれくらいの値段で売れるでしょうか?」という問題を解決するために、機械学習モデルは、線形回帰モデルを使用して、この家に関する既知の事実 (平方フィートなど) に基づいて家の販売価格を予測できます。

リホスト

「7 Rs」を参照してください。

リリース

デプロイプロセスで、変更を本番環境に昇格させること。

再配置

「7 Rs」を参照してください。

リプラットフォーム

「7 Rs」を参照してください。

再購入

「7 Rs」を参照してください。

回復性

中断に抵抗または中断から回復するアプリケーションの機能。 AWS クラウドでの回復力を計画する際には、一般的に、高可用性とディザスタリカバリが考慮されます。詳細については、「AWS クラウド の耐障害性」を参照してください。

リソースベースのポリシー

Amazon S3 バケット、エンドポイント、暗号化キーなどのリソースにアタッチされたポリシー。このタイプのポリシーは、アクセスが許可されているプリンシパル、サポートされているアクション、その他の満たすべき条件を指定します。

実行責任者、説明責任者、協業先、報告先 (RACI) に基づくマトリックス

移行活動とクラウド運用に関わるすべての関係者の役割と責任を定義したマトリックス。マトリックスの名前は、マトリックスで定義されている責任の種類、すなわち責任 (R) 、説明責任 (A) 、協議 (C) 、情報提供 (I) に由来します。サポート (S) タイプはオプションです。サポートが含まれる場合は RASCI マトリックスと呼ばれ、含まれない場合は RACI マトリックスと呼ばれます。

レスポンシブコントロール

有害事象やセキュリティベースラインからの逸脱について、修復を促すように設計されたセキュリティコントロール。詳細については、「 AWSでのセキュリティコントロールの実装」の「レスポンシブコントロール」を参照してください。

保持

「7 Rs」を参照してください。

廃止

「7 Rs」を参照してください。

検索拡張生成 (RAG)

生成 AI の技術。これにより、LLM では、レスポンスの生成前に、トレーニングデータソースの外部にある信頼できるデータソースが参照されます。例えば、RAG モデルによって、組織のナレッジベースまたはカスタムデータのセマンティック検索を実行できる場合があります。細については、「RAG (検索拡張生成) とは何ですか?」を参照してください。

ローテーション

定期的にシークレット情報を更新して、攻撃者が認証情報にアクセスするのをより困難にするプロセス。

行と列のアクセス制御 (RCAC)

アクセスルールが定義された、基本的で柔軟な SQL 表現の使用。RCAC は行権限と列マスクで構成されています。

RPO

「目標復旧時点」を参照してください。

RTO

「目標復旧時間」を参照してください。

ランブック

特定のタスクを実行するために必要な手動または自動化された一連の手順。これらは通常、エラー率の高い反復操作や手順を合理化するために構築されています。

S

SAML 2.0

多くの ID プロバイダー (IdP) が使用しているオープンスタンダード。この機能を使用すると、フェデレーティッドシングルサインオン (SSO) が有効になるため、ユーザーは組織内のすべてのユーザーを IAM で作成しなくても、 AWS マネジメントコンソール にログインしたり AWS 、API オペレーションを呼び出すことができます。SAML 2.0 ベースのフェデレーションの詳細については、IAM ドキュメントの「SAML 2.0 ベースのフェデレーションについて」を参照してください。

SCADA

「監視制御とデータ取得」を参照してください。

SCP

「サービスコントロールポリシー」を参照してください。

シークレット

暗号化された形式で保存するパスワードやユーザー認証情報などの AWS Secrets Manager機密情報または制限付き情報。シークレット値とそのメタデータで構成されます。シークレット値には、バイナリ、1 つの文字列、複数の文字列を指定できます。詳細については、Secrets Manager ドキュメントの「Secrets Manager シークレットの概要」を参照してください。

セキュリティバイデザイン

開発プロセス全体を通してセキュリティが考慮されているシステムエンジニアリングのアプローチ。

セキュリティコントロール

脅威アクターによるセキュリティ脆弱性の悪用を防止、検出、軽減するための、技術上または管理上のガードレール。セキュリティコントロールには、主に 4 つの種類があります。4 つとは、予防、検出、レスポンシブ、プロアクティブです。

セキュリティ強化

アタックサーフェスを狭めて攻撃への耐性を高めるプロセス。このプロセスには、不要になったリソースの削除、最小特権を付与するセキュリティのベストプラクティスの実装、設定ファイル内の不要な機能の無効化、といったアクションが含まれています。

Security Information and Event Management (SIEM) システム

セキュリティ情報管理 (SIM) とセキュリティイベント管理 (SEM) のシステムを組み合わせたツールとサービス。SIEM システムは、サーバー、ネットワーク、デバイス、その他ソースからデータを収集、モニタリング、分析して、脅威やセキュリティ違反を検出し、アラートを発信します。

セキュリティレスポンスの自動化

セキュリティイベントへの自動レスポンスまたは自動修復を目的として、事前定義およびプログラムされたアクション。これらの自動化は、セキュリティのベストプラクティスを実装するのに役立つ検出的または応答的な AWS セキュリティコントロールとして機能します。自動レスポンスアクションの例には、VPC セキュリティグループの変更、Amazon EC2 インスタンスへのパッチ適用、認証情報の更新などがあります。

サーバー側の暗号化

送信先にあるデータの、それ AWS のサービス を受け取る による暗号化。

サービスコントロールポリシー (SCP)

AWS Organizationsの組織内の、すべてのアカウントのアクセス許可を一元的に管理するポリシー。SCP は、管理者がユーザーまたはロールに委任するアクションに、ガードレールを定義したり、アクションの制限を設定したりします。SCP は、許可リストまたは拒否リストとして、許可または禁止するサービスやアクションを指定する際に使用できます。詳細については、 AWS Organizations ドキュメントの「サービスコントロールポリシー」を参照してください。

サービスエンドポイント

のエントリポイントの URL AWS のサービス。ターゲットサービスにプログラムで接続するには、エンドポイントを使用します。詳細については、「AWS 全般のリファレンス」の「AWS のサービス エンドポイント」を参照してください。

サービスレベルアグリーメント (SLA)

サービスのアップタイムやパフォーマンスなど、IT チームがお客様に提供すると約束したものを明示した合意書。

サービスレベルインジケータ (SLI)

エラー率、可用性、スループットといった、サービスパフォーマンス面の指標。

サービスレベル目標 (SLO)

サービスレベルインジケータによって測定され、サービスの状態を表すターゲットメトリクス。

責任共有モデル

クラウドのセキュリティとコンプライアンス AWS について と共有する責任を説明するモデル。AWS はクラウドのセキュリティを担当しますが、お客様はクラウドのセキュリティを担当します。詳細については、「責任共有モデル」を参照してください。

SIEM

「Security Information and Event Management システム」を参照してください。

単一障害点 (SPOF)

特定のアプリケーションを構成する単一の重要なコンポーネントで発生し、システム稼働に支障をきたす可能性のある障害。

SLA

「サービスレベルアグリーメント」を参照してください。

SLI

「サービスレベルインジケータ」を参照してください。

SLO

「サービスレベルの目標」を参照してください。

スプリットアンドシードモデル

モダナイゼーションプロジェクトのスケーリングと加速のためのパターン。新機能と製品リリースが定義されると、コアチームは解放されて新しい製品チームを作成します。これにより、お客様の組織の能力とサービスの拡張、デベロッパーの生産性の向上、迅速なイノベーションのサポートに役立ちます。詳細については、「AWS クラウドでのアプリケーションをモダナイズするための段階的アプローチ」を参照してください。

SPOF

「単一障害点」を参照してください。

スタースキーマ

データベースの編成構造を意味し、1 つの大きいファクトテーブルにトランザクションデータまたは測定データが保存され、1 つ以上の小さいディメンションテーブルにデータ属性が保存されます。この構造は、データウェアハウスやビジネスインテリジェンスを用途とするように設計されています。

strangler fig パターン

レガシーシステムが廃止されるまで、システム機能を段階的に書き換えて置き換えることにより、モノリシックシステムをモダナイズするアプローチ。このパターンは、宿主の樹木から根を成長させ、最終的にその宿主を包み込み、宿主に取って代わるイチジクのつるを例えています。そのパターンは、モノリシックシステムを書き換えるときのリスクを管理する方法として Martin Fowler により提唱されました。このパターンの適用方法の例については、「コンテナと Amazon API Gateway を使用して、従来の Microsoft ASP.NET (ASMX) ウェブサービスを段階的にモダナイズ」を参照してください。

サブネット

VPC 内の IP アドレスの範囲。サブネットは、1 つのアベイラビリティーゾーンに存在する必要があります。

監視制御とデータ取得 (SCADA)

製造分野において、ハードウェアとソフトウェアを使用して物理アセットと本番運用をモニタリングするシステム。

対称暗号化

データの暗号化と復号に同じキーを使用する暗号化のアルゴリズム。

合成テスト

ユーザーとのやり取りをシミュレートして、起こり得る問題を検出したり、パフォーマンスをモニタリングしたりすることで、システムをテストします。Amazon CloudWatch Synthetics を使用すると、こうしたテストを作成できます。

システムプロンプト

コンテキスト、指示、ガイドラインなどを提示して、LLM に動作を指示する手法。システムプロンプトは、コンテキストを設定して、ユーザーとやり取りするルールを確立するのに有用です。

T

タグ

AWS リソースを整理するためのメタデータとして機能するキーと値のペア。タグは、 リソースの管理、識別、整理、検索、フィルタリングに役立ちます。詳細については、「AWS リソースのタグ付け」を参照してください。

ターゲット変数

監督された機械学習でお客様が予測しようとしている値。これは、結果変数のことも指します。例えば、製造設定では、ターゲット変数が製品の欠陥である可能性があります。

タスクリスト

ランブックの進行状況を追跡するために使用されるツール。タスクリストには、ランブックの概要と完了する必要のある一般的なタスクのリストが含まれています。各一般的なタスクには、推定所要時間、所有者、進捗状況が含まれています。

テスト環境

「環境」を参照してください。

トレーニング

お客様の機械学習モデルに学習するデータを提供すること。トレーニングデータには正しい答えが含まれている必要があります。学習アルゴリズムは入力データ属性をターゲット (お客様が予測したい答え) にマッピングするトレーニングデータのパターンを検出します。これらのパターンをキャプチャする機械学習モデルを出力します。そして、お客様が機械学習モデルを使用して、ターゲットがわからない新しいデータでターゲットを予測できます。

トランジットゲートウェイ

VPC とオンプレミスネットワークを相互接続するために使用できる、ネットワークの中継ハブ。詳細については、 AWS Transit Gateway ドキュメントの「トランジットゲートウェイとは」を参照してください。

トランクベースのワークフロー

デベロッパーが機能ブランチで機能をローカルにビルドしてテストし、その変更をメインブランチにマージするアプローチ。メインブランチはその後、開発環境、本番前環境、本番環境に合わせて順次構築されます。

信頼されたアクセス

ユーザーに代わって AWS Organizations およびそのアカウントで組織内でタスクを実行するために指定したサービスにアクセス許可を付与します。信頼されたサービスは、サービスにリンクされたロールを必要なときに各アカウントに作成し、ユーザーに代わって管理タスクを実行します。詳細については、 ドキュメントの「 を他の AWS のサービス AWS Organizations で使用する AWS Organizations 」を参照してください。

チューニング

機械学習モデルの精度を向上させるために、お客様のトレーニングプロセスの側面を変更する。例えば、お客様が機械学習モデルをトレーニングするには、ラベル付けセットを生成し、ラベルを追加します。これらのステップを、異なる設定で複数回繰り返して、モデルを最適化します。

ツーピザチーム

2 枚のピザを分け合えることができるくらい小さな DevOps チーム。ツーピザチームの規模では、ソフトウェア開発におけるコラボレーションに最適な機会が確保されます。

U

不確実性

予測機械学習モデルの信頼性を損なう可能性がある、不正確、不完全、または未知の情報を指す概念。不確実性には、次の 2 つのタイプがあります。認識論的不確実性は、限られた、不完全なデータによって引き起こされ、弁論的不確実性 は、データに固有のノイズとランダム性によって引き起こされます。詳細については、深層学習システムにおける不確実性の定量化ガイドを参照してください。

未分化なタスク

ヘビーリフティングとも呼ばれ、アプリケーションの作成と運用には必要だが、エンドユーザーに直接的な価値をもたらさなかったり、競争上の優位性をもたらしたりしない作業です。未分化なタスクの例としては、調達、メンテナンス、キャパシティプランニングなどがあります。

上位環境

「環境」を参照してください。

V

バキューミング

ストレージを再利用してパフォーマンスを向上させるために、増分更新後にクリーンアップを行うデータベースのメンテナンス操作。

バージョンコントロール

リポジトリ内のソースコードへの変更など、変更を追跡するプロセスとツール。

VPC ピアリング

プライベート IP アドレスを使用してトラフィックをルーティングできる、2 つの VPC 間の接続。詳細については、Amazon VPC ドキュメントの「VPC ピア機能とは」を参照してください。

脆弱性

システムのセキュリティを脅かすソフトウェアまたはハードウェアの欠陥。

W

ウォームキャッシュ

頻繁にアクセスされる最新の関連データを含むバッファキャッシュ。データベースインスタンスはバッファキャッシュから、メインメモリまたはディスクからよりも短い時間で読み取りを行うことができます。

ウォームデータ

アクセス頻度の低いデータ。この種類のデータをクエリする場合、通常は適度に遅いクエリでも問題ありません。

ウィンドウ関数

現在のレコードに何らかの形で関連している行のグループに計算を実行する SQL 関数。ウィンドウ関数は、移動平均を計算したり、現在の行の相対位置に基づいて他の行の値にアクセスするといったタスクの処理に役立ちます。

ワークロード

ビジネス価値をもたらすリソースとコード (顧客向けアプリケーションやバックエンドプロセスなど) の総称。

ワークストリーム

特定のタスクセットを担当する移行プロジェクト内の機能グループ。各ワークストリームは独立していますが、プロジェクト内の他のワークストリームをサポートしています。たとえば、ポートフォリオワークストリームは、アプリケーションの優先順位付け、ウェーブ計画、および移行メタデータの収集を担当します。ポートフォリオワークストリームは、これらの設備を移行ワークストリームで実現し、サーバーとアプリケーションを移行します。

WORM

「Write-Once-Read-Many」を参照してください。

WQF

「AWS ワークロード資格フレームワーク」を参照してください

Write-Once-Read-Many (WORM)

データを 1 回のみ書き込むことで、データの削除や変更を防ぐストレージモデル。承認済みユーザーは、必要な回数だけデータを読み取ることができますが、変更することはできません。このデータストレージインフラストラクチャは、イミュータブルと見なされます。

Z

ゼロデイエクスプロイト

ゼロデイ脆弱性を悪用した攻撃 (一般的にマルウェアによる)。

ゼロデイ脆弱性

実稼働システムにおける未解決の欠陥または脆弱性。脅威アクターは、このような脆弱性を利用してシステムを攻撃する可能性があります。開発者は、よく攻撃の結果で脆弱性に気付きます。

ゼロショットプロンプト

LLM にタスク実行の手順は提示するが、実行のガイドとして役立つ例 (ショット) は提示しない方法。LLM は、事前トレーニング済みの知識を使用してタスクを処理する必要があります。ゼロショットプロンプトの有効性は、タスクの複雑さとプロンプトの品質によって異なります。「数ショットプロンプト」も参照してください。

ゾンビアプリケーション

平均 CPU およびメモリ使用率が 5% 未満のアプリケーション。移行プロジェクトでは、これらのアプリケーションを廃止するのが一般的です。

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。