Engineer

GitOps

Gitの宣言に実環境を継続的に合わせる運用手法

どのようなものか

GitOpsは、システムの望ましい状態をGitに宣言として書き、ソフトウェアエージェントが実環境を合わせ続ける運用方法です。本番を手作業やノートPCのスクリプトで変えません。変更はコミットで、環境内のエージェントが検知して反映します。OpenGitOpsは四原則に整理し、管理するGitOps Working Groupが2021年10月8日にバージョン1.0.0を公開しました。

望ましい状態には、宣言的であること、版管理され不変であること、自動取得されること、継続的に調整されることが求められます。宣言的とは、その状態へ到達させるコマンドではなく、ファイルにあるべきシステムの姿を記すことです。版管理と不変とは、保存済みの版を編集せず新しい版で置き換え、全履歴を保つことです。自動取得とは、エージェント自身が宣言をソースから取ることです。継続的な調整とは、実環境と宣言を繰り返し比べ、そろえる処理を続けることです。用語集は、継続的とは処理を続ける意味で、瞬時を意味しないとも補います。

用語集には、見落としがちな境界が二つあります。名前の由来でもあるGitは、状態を記録する標準的な保存先ですが、不変の版、アクセス制御、望ましい状態の変更の監査を提供するものなら、ほかの保存先も使えます。また、望ましい状態はシステムを再構成する設定で、通常、データベースの内容などの永続データは含みません。データベースが動くマシンを作り直せても、中の行は作り直せません。

概念を判断に生かす

「GitOpsとは？」を実際の判断に生かしましょう。その概念を、開発または購入するシステムに結び付けて考えるお手伝いをします。 変更を検証し、リリースし、元に戻せる仕組みを整え、金曜の深夜ではなく火曜の午後にお客様へ届けられるようにします。

サービスを見る お問い合わせ →

用語の背後にある前提を検証する

解説とリンク先の実績を参考に、プロジェクトについての質問を整理してください。ツールを選ぶ前に、前提を一緒に検討できます。 GitHub ActionsでGIS SaaSのデプロイを自動化 — デプロイ、データ、報告を手作業から解放

GitHub ActionsでGIS SaaSのデプロイを自動化 お問い合わせ →

なぜ重要なのか

普通のデプロイパイプラインとの違いは、何が処理を起こすかです。OpenGitOpsの用語集は、事前設定した契機で動くCI/CDと、実状態が宣言から離れるたびに調整を始めるGitOpsを対比します。新しいコミットで希望が変わる場合も、稼働中のものが宣言からずれる場合もあります。一度 `kubectl apply` しただけのパイプラインは、その後を見ません。Kubernetesの文書も、宣言的なapplyは、実オブジェクトへ直接書いた内容を保つ一方、設定ファイルへ戻して統合しないため、両者が静かに食い違い得るとします。継続的に調整するエージェントは、両方向のずれを見つけます。

二つ目の利点はアクセス権の場所です。Argo CDの文書は、自動同期なら、デプロイパイプラインはArgo CD APIサーバーの認証情報がなくても動かせて、コミットとプッシュで済むと指摘します。必要なのはクラスターでなく、リポジトリーへの書き込み権限です。変更のレビューも、もともとそこで行います。

三つ目は履歴です。望ましい状態を不変の版として保つため、何を、誰が、いつ変えたか、意図した全変更がログに残ります。戻すには前の版を選びます。Argo CDも、Gitへコミットした任意のアプリ設定へのロールバックを機能に挙げます。障害後の「何が変わったか」に、シェル履歴や記憶から組み立て直さず答えられます。

実際の活用例

Kubernetesで実現するエージェントにFluxとArgo CDがあります。Argo CDは自身を、Kubernetes向けの宣言的なGitOps継続的デリバリーツールと説明し、FluxはコントローラーとAPIの集合をGitOps Toolkitと呼びます。本番のマニフェストをfleetというリポジトリーへ置くチームを考えます。Fluxの二つのオブジェクトで、クラスターをこのリポジトリーへ接続します。最初が、場所と取得間隔をソースコントローラーへ、二つ目が適用するディレクトリーとクラスターとの照合間隔をkustomizeコントローラーへ伝えます。

```
apiVersion: source.toolkit.fluxcd.io/v1
kind: GitRepository
metadata:
  name: fleet
```

```
  namespace: flux-system
spec:
  interval: 5m
  url: https://git.example.com/platform/fleet.git
  ref:
    branch: main
---
apiVersion: kustomize.toolkit.fluxcd.io/v1
kind: Kustomization
metadata:
  name: production
  namespace: flux-system
spec:
  interval: 10m
  sourceRef:
    kind: GitRepository
    name: fleet
  path: "./production"
  prune: true
  timeout: 1m
```

一つの変更を追います。技術者が新しいイメージタグへ変える提案をマージすると、五分ごとにブランチを取るソースコントローラーが、次回、新しい版を見つけます。新しいソースの版は次の間隔を待たず、すぐ扱います。kustomizeコントローラーが `./production` 内のマニフェストをビルドして適用します。Fluxの文書は、この組み合わせの残りの動作も説明します。十分ごとという設定に従い、`Kustomization` はサーバー側applyのドライランでずれを検知し、調整します。同僚が `kubectl edit` した変更は元へ戻ります。 `prune` が有効なため、ディレクトリーから宣言を消したオブジェクトも、次の版でクラスターから消します。戻す操作は `git revert` とマージで、ほかと同じ経路を通ります。

Argo CDは `Application` リソースで同じ仕事をしますが、既定値が違います。一つの文書は三分ごとにリポジトリーを確認するとし、自動同期の文書は120秒に最大60秒のランダムなずらしを加え、最大三分とします。実環境がGitの目標と違うとOutOfSyncになり、自動同期が有効なら合わせます。ただし、既定ではGitから消えたものを削除せず、クラスターを直接変えても反応しません。削除と自己修復には、それぞれ `prune` と `selfHeal` という別のスイッチがあり、どちらも初めは無効です。

同じ型はKubernetesの外でも使えます。2026年8月の、20レコードのDNSゾーンを事業者APIと照合して合わせる仕事は、管理画面にあったレコードをリポジトリーの宣言へ移しました。一つの実行入口は宣言を事業者側の状態と照合して合わせ、明示的な選択が必要な別の入口は、何も変えずに差だけを報告する監査を行います。事例は、合わせるより違いを知りたいことが多いため、監査を最も使

うとします。四原則に照らすと、宣言と版管理という最初の二つです。エージェントが自動取得して適用するとは書かれていないため、残り二つは主張しません。自動修正より先にずれを見せる、Argo CDの既定と同じ始め方の理由にもなります。

問題が起きるところ

機密情報を平文でGitへ置きます。 望ましい全状態を入れると、パスワードも入れたくなりますが、削除後も履歴に残ります。Fluxは、Kubernetesのシークレットを平文やbase64で置くのは、アクセス制限した非公開リポジトリーでも危険とします。コミット前に暗号化してください。Fluxは `.spec.decryption` で設定し、age、OpenPGP、クラウドの鍵管理サービスを使うSOPSで復号します。`metadata` 、 `kind` 、 `apiVersion` は読める必要があり、暗号化するのは `data` と `stringData` だけです。Argo CDは別の方法を強く勧めます。生成時の注入でなく、Sealed Secrets、External Secrets Operator、Secrets Store CSI Driverなどの道具を使い、配置先のクラスターでシークレットの値を解決します。生成時の注入を避ける理由の一つは、結果のマニフェストをRedisキャッシュが平文で持つことです。

緊急修正が残らない、またはずれが消えません。 既定が異なります。Fluxは `kubectl edit` 、 `patch` 、 `delete` の変更を速やかに戻します。普段は正しくても、障害中の手動修正まで消えます。Argo CDは自己修復が無効なら直接の変更で同期を起動せず、OutOfSyncのまま、いつまでも残り得ます。対処はどちらも、Gitで変えるか、調整を明示的に止めることです。Fluxの `.spec.suspend` は、新版の適用とずれの修正を、falseへ戻すまで止めます。自動拡縮が管理する `Deployment` の複製数など、別のコントローラーの欄には、JSON Pointerの `/spec/replicas` のようなパスを `.spec.ignore` へ渡し、上書きし合うのを止めます。戻すときもGitを通してください。Argo CDは自動同期が有効なアプリケーションのロールバックを拒否します。

Gitで消しても、動き続けます。 マニフェストを削除すると、宣言は消えますが、実オブジェクトは消えません。Argo CDの自動同期は、削除機能が有効でなければ、Gitで定義されなくなったリソースを削除せず、Fluxも各 `Kustomization` に必須の `.spec.prune` がtrueの場合だけ消します。無効なら、クラスター内に、誰も宣言していないオブジェクトが徐々に増えます。有効にしつつ、宣言より長く残す必要のある少数のものには、 `kustomize.toolkit.fluxcd.io/prune: disabled` のラベルか注釈を付けてください。Argo CDは `allowEmpty` がなければ、削除を有効にした自動同期で、アプリケーションの全リソースをなくすことを許しません。壊れたコミットでディレクトリーが空になっても、クラスターまで空にしないための保護です。

マージが次の確認を待ちます。 タイマーで取得すると、独立性を得る代わりに、反映の速さを犠牲にします。Argo CDは三分、Fluxは次回取得で見つけます。両方ともGitサーバーのWebhookで待ち時間を減らせます。Argo CDは `/api/webhook` で受け、Fluxでは `Receiver` リソースが、プッシュイベントを、そこに列挙されたソースの調整へ変えます。Argo CDの共有シークレットは、通知が再確認だけを起動するため任意ですが、Argo CDが一般公開されている場合には、分散サービス拒否攻撃を防ぐために文書が推奨しています。Fluxはレシーバーの名前、名前空間、秘密のトークンから固有のパスを作り、GitHubの場合は内容のHMAC署名も検証します。

Gitが本番の依存先になります。 エージェントがGitサーバーから取得する構成では、そのサーバーの停止で、修正を含む全変更が止まります。FluxのOCI方式、Gitless GitOpsは、人の操作先をGitに残し、クラスターはコンテナーレジストリーから版付き成果物を取り、Gitサーバーへ接続せず本番の成果物を得られるようにします。何を読むにせよ、期待する本番と同じ可用性を与えてください。障害時には、そこが変更を運ぶ唯一の経路になるためです。

参考資料

- OpenGitOps — GitOps Working Groupによる、版1.0.0の四原則です。
- GitOps用語集 v1.0.0 — 望ましい状態、ずれ、調整、状態保存先の定義です。
- 自動同期の方針 — 削除、自己修復、空状態の許可、同期の契機を含むArgo CDの説明です。
- Kustomization — マニフェストのディレクトリーを適用するFlux APIの、間隔、削除、中断、無視、SOPS復号です。
- 機密情報の管理 — マニフェストの生成時ではなく配置先のクラスターでシークレットを解決する、Argo CDの推奨です。
- 設定ファイルによるKubernetesオブジェクトの宣言的管理 — 設定ファイルのディレクトリーに対する `kubectl apply` 、 `kubectl diff` と、直接の変更をapplyが扱う方法です。

用語集

実際の制約に合わせて選ぶ

デプロイの流れ、運用上のリスク、緊急対応の担当者をお知らせください。検査、復旧手段、明確な責任の所在を伴う変更の範囲を定められます。

料金を見る お問い合わせ →

用語を実務に結び付ける

「GitOpsとは？」がお客様の組織にどう当てはまるかをご相談ください。現在のシステムの具体例が一つあれば、話を始められます。

サービスを見る

welcome@engineer.company +45 50 10 61 84 デンマーク、コペンハーゲン

このページを持ち歩く

連絡先カード

心を込めて丁寧に制作： D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

著作権 © 2025 – 現在 · Engineer ApS · すべての権利を留保します。

印刷元： https://engineer.company/ja/glossary/gitops/