Engineer

GitOps

керування інфраструктурою з бажаного стану, записаного в Git

Що це таке

GitOps — це спосіб експлуатації програмних систем, за якого бажаний стан системи записано як декларації в Git, а програмні агенти підтримують відповідність робочої системи цим деклараціям. Назву в українській вживають без перекладу. Ніхто не змінює продакшн вручну чи скриптом із ноутбука. Зміна — це коміт, а агент усередині середовища помічає новий коміт і приводить систему у відповідність до нього. Проєкт OpenGitOps формулює цю ідею як чотири принципи; версію 1.0.0 принципів опублікувала 8 жовтня 2021 року робоча група GitOps Working Group, яка наглядає за проєктом.

Принципи вимагають, щоб бажаний стан був декларативним, версіонованим і незмінним, автоматично витягнутим і безперервно узгоджуваним. Декларативний означає, що файли описують, якою має бути система, а не команди, які до цього ведуть. Версіонований і незмінний означає, що збережену версію ніколи не редагують, лише заміняють новою, а вся історія зберігається. Автоматично витягнутий означає, що програмні агенти самі забирають декларації з джерела. Безперервно узгоджуваний означає, що агенти постійно спостерігають за фактичним станом і намагаються застосувати бажаний; глосарій проєкту уточнює, що безперервність означає, що робота не припиняється, а не що вона відбувається миттєво.

Дві межі в цьому глосарії легко пропустити. Git, від якого походить назва, — канонічний приклад сховища стану, але підходить будь-яке сховище, яке зберігає незмінні версії й забезпечує контроль доступу та аудит змін бажаного стану. А бажаний стан — це конфігурація, потрібна, щоб відтворити систему, і зазвичай він не охоплює постійних даних на кшталт вмісту бази даних. GitOps може перебудувати машини, на яких працює база даних, але не рядки в ній.

Застосуйте поняття до рішення

Застосуйте «Що таке GitOps?» до конкретного рішення. Ми можемо пов'язати поняття із системою, яку ви створюєте або купуєте. Перевірено, випущено і з можливістю відкату, тож зміни доходять до клієнтів у вівторок удень, а не пізно в п'ятницю.

Переглянути наші послуги Зв'язатися з нами →

Перевірте припущення за терміном

Використайте пояснення та пов'язані роботи, щоб сформулювати запитання для проєкту. Ми можемо перевірити припущення разом до вибору інструмента. Автоматизувала розгортання GIS SaaS через CI/CD — розгортання, дані та звітність — усе без ручної роботи

Автоматизувала розгортання GIS SaaS через CI/CD Зв'язатися з нами →

Чому це важливо

Від звичайного конвеєра розгортання GitOps відрізняє те, що саме запускає роботу. Глосарій OpenGitOps протиставляє традиційний CI/CD, де автоматизацію здебільшого запускають наперед задані тригери, підходу GitOps, де узгодження починається щоразу, коли фактичний стан розходиться з задекларованим. Розбіжність може прийти з будь-якого боку: новий коміт змінює те, чого хочуть, або робоча система дрейфує від задекларованого. Конвеєр, який один раз виконав `kubectl apply`, нічого не знає про те, що сталося потім. Документація Kubernetes зазначає, що декларативний apply зберігає зміни, записані безпосередньо в живі об’єкти, не переносячи їх назад у конфігураційні файли, тож файли й кластер можуть непомітно розійтися. Агент, який узгоджує стан, помічає розбіжності обох видів.

Другий виграш стосується доступу. Документація Argo CD звертає увагу, що з автоматичною синхронізацією конвеєрам CI/CD більше не потрібен прямий доступ до API-сервера Argo CD, щоб розгортати: конвеєр робить коміт і надсилає його в репозиторій. Конвеєру потрібне право запису в репозиторій, а не в кластер, а саме в репозиторії зміни й так проходять рецензування.

Третій виграш — історія. Оскільки бажаний стан версіонований і незмінний, журнал репозиторію стає записом кожної задуманої зміни: що змінилося, хто це зробив і коли. Повернутися назад означає вибрати попередню версію, і Argo CD серед своїх можливостей називає відкат до будь-якої конфігурації, зафіксованої в Git. Після інциденту ця історія відповідає на перше запитання, що саме змінилося, і нікому не доводиться відновлювати це з історії оболонки та пам’яті.

Як це виглядає на практиці

Flux і Argo CD — два агенти, які втілюють цей підхід на Kubernetes. Argo CD описує себе як декларативний інструмент безперервної доставки для Kubernetes, побудований на GitOps, а Flux називає свій набір контролерів і API GitOps Toolkit. Візьмімо команду, яка тримає маніфести продакшну в репозиторії з назвою fleet. Два об'єкти Flux пов'язують із ним кластер: перший повідомляє source controller у Flux, де лежить репозиторій і як часто його забирати, а другий повідомляє kustomize controller, який каталог застосовувати і як часто звіряти з ним кластер.

```
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
```

Простежмо одну зміну. Інженерка зливає pull request, який переводить сервіс на новий тег образу. Source controller забирає гілку кожні п'ять хвилин і під час наступного забирання бачить нову ревізію. Нову ревізію джерела обробляють одразу, а не на наступному інтервалі, тож kustomize controller збирає маніфести з `./production` і застосовує їх. Решту документація Flux описує саме для цієї пари об'єктів. Кожні десять хвилин, як задає інтервал, `Kustomization` виконує пробний server-side apply, щоб виявити й виправити дрейф, тож колега, який змінить той самий об'єкт через `kubectl edit`, побачить, що зміну скасовано. Оскільки `prune` увімкнено, об'єкт, чий

маніфест прибрали з каталогу, буде видалено з кластера з наступною ревізією. Відкат — це `git revert` і злиття, які проходять тим самим шляхом, що й будь-яка інша зміна.

Argo CD робить ту саму роботу через ресурс `Application`, але з іншими типовими налаштуваннями. Одна сторінка його документації каже, що він опитує репозиторії кожні три хвилини; інша, про автоматичну синхронізацію, називає інтервал 120 секунд плюс випадковий зсув до 60 секунд, тобто не більше трьох хвилин. Коли живий стан відрізняється від цільового стану в Git, застосунок позначається як OutOfSync, і з увімкненою автоматичною синхронізацією Argo CD його синхронізує. Проте за замовчуванням автоматична синхронізація не видаляє того, що зникло з Git, і не реагує на зміни, зроблені безпосередньо в кластері. Прибирання і самовідновлення — окремі перемикачі, `prune` і `selfHeal`, і обидва спочатку вимкнені.

Той самий підхід працює й поза Kubernetes. Робота над зоною DNS із 20 записів, узгодженою з API провайдера, виконана у серпні 2026 року, перетворила записи, що жили у вебконсолі провайдера, на декларацію в репозиторії. Одна точка входу узгоджує декларацію з провайдером; інша, яка запускається лише на явний запит, тільки проводить аудит і показує різницю, нічого не змінюючи. Сторінка називає аудит частиною, якою користуються найчастіше, бо знати різницю потрібно частіше, ніж її усувати. Якщо міряти чотирма принципами, це перші два: зону задекларовано й версіоновано. Сторінка не описує агента, який сам забирає декларацію і застосовує її, тож на два останні принципи робота не претендує. Це також добрий аргумент на користь того, щоб починати там, де за замовчуванням починає Argo CD: спершу показати розбіжність, а вже потім дозволяти щось її виправляти.

Де виникають проблеми

Секрети відкритим текстом у Git. Репозиторій, у якому лежить увесь бажаний стан, спокушає покласти туди й паролі, а історія зберігає їх і після видалення. Документація Flux вважає небезпечним тримати секрети Kubernetes у Git відкритим текстом або в base64, хоч би якою була видимість репозиторію чи обмеження доступу до нього. Секрети треба шифрувати до коміту. Flux розшифровує їх за допомогою SOPS, з age, OpenPGP або хмарною службою керування ключами, що налаштовується через `.spec.decryption`; шифруйте лише поля `data` і `stringData`, бо `metadata`, `kind` і `apiVersion` мають лишатися читабельними. Документація Argo CD наполегливо радить інший шлях: заповнювати секрети в цільовому кластері

інструментом на кшталт Sealed Secrets, External Secrets Operator чи Secrets Store CSI Driver, а не вставляти їх, поки Argo CD генерує маніфести. Від останнього вона застерігає, зокрема тому, що Argo CD зберігає згенеровані маніфести незашифрованими у своєму кеші Redis.

Термінове виправлення, яке не тримається, або дрейф, який нікуди не зникає. Тут два інструменти за замовчуванням поводяться по-різному. У Flux зміну, зроблену через `kubectl edit`, `patch` чи `delete`, швидко скасовують, і це правильно, доки хтось не внесе термінове виправлення вручну під час простою і не побачить, як воно зникає. В Argo CD зміна, зроблена безпосередньо в кластері, не запускає автоматичної синхронізації, якщо не ввімкнено самовідновлення, тож дрейф може лишатися скільки завгодно з позначкою OutOfSync. Рішення однакове для обох: робіть зміну в Git або відкрито призупиняйте узгодження. Поле `.spec.suspend` у Flux зупиняє застосування нових ревізій і призупиняє виправлення дрейфу, доки його знову не встановлять у false. Для поля, яким володіє інший контролер, як-от кількість реплік у `Deployment`, якою керує автомасштабувальник, поле `.spec.ignore` у Flux приймає шляхи JSON Pointer на кшталт `/spec/replicas`, і два контролери перестають перезаписувати один одного. Відкочуйтеся теж через Git: Argo CD відмовляє у відкаті застосунку, для якого ввімкнено автоматичну синхронізацію.

Видалено в Git, але досі працює. Прибрати маніфест означає прибрати декларацію, а не об'єкт. Автоматична синхронізація Argo CD не видаляє ресурси, яких більше немає в Git, якщо не ввімкнено прибирання, а Flux видаляє їх, лише коли `.spec.prune`, поле, яке має задати кожна `Kustomization`, дорівнює true. Якщо лишити його вимкненим, кластер поступово заповнюється об'єктами, яких ніхто не декларує. Увімкніть його і позначте ті кілька об'єктів, що мають пережити свій маніфест, міткою або анотацією `kustomize.toolkit.fluxcd.io/prune: disabled`. Argo CD має запобіжник, який варто зберегти: автоматична синхронізація з прибиранням не залишить застосунок зовсім без ресурсів, якщо не задано `allowEmpty`, і це не дає хибному коміту, який спорожнив каталог, спорожнити й кластер.

Злиття, яке чекає на наступне опитування. Агент, що забирає зміни за таймером, обмінює швидкість на незалежність: Argo CD називає своїм періодом опитування три хвилини, а Flux бачить коміт під час наступного забирання джерела. Обидва приймають вебхук від Git-сервера, щоб скоротити очікування. Argo CD приймає його на ендпоінті `/api/webhook`, а Flux використовує ресурс `Receiver`, який перетворює подію push на узгодження перелічених у ньому джерел. Спільний секрет для вебхука в

Argo CD необов'язковий, бо корисне навантаження лише запускає оновлення, але документація радить задавати його, коли Argo CD доступний публічно, щоб запобігти розподіленим атакам на відмову в обслуговуванні. Flux генерує унікальний шлях вебхука з назви ресурсу, простору імен і секретного токена, а для GitHub перевіряє HMAC-підпис корисного навантаження.

Git стає залежністю продакшну. Якщо агент забирає зміни з Git-сервера, збій Git зупиняє будь-яку зміну, зокрема й виправлення. Варіант Flux на основі OCI, який проєкт називає Gitless GitOps, лишає Git інтерфейсом, у якому працюють люди, але змушує кластер забирати версіоновані артефакти з реєстру контейнерів, тож, за словами документації Flux, Git-сервер перестає бути залежністю продакшну. Хоч би з якого джерела читав агент, забезпечте йому доступність, якої ви чекаєте від продакшну, бо під час збою це єдиний шлях, яким може пройти зміна.

Що почитати далі

- OpenGitOps — чотири принципи GitOps у версії 1.0.0 від GitOps Working Group.
- GitOps Glossary v1.0.0 — визначення, на яких стоять принципи: бажаний стан, дрейф, узгодження і сховище стану.
- Automated Sync Policy — автоматична синхронізація в Argo CD з перемикачами прибирання, самовідновлення і порожніх застосунків, а також те, коли запускається кожна синхронізація.
- Kustomization — API Flux для застосування каталогу маніфестів: інтервал, prune, suspend, ignore і розшифрування через SOPS.
- Secret Management — чому Argo CD радить заповнювати секрети в цільовому кластері, а не під час генерування маніфестів.
- Declarative Management of Kubernetes Objects Using Configuration Files — як `kubectl apply` і `kubectl diff` працюють із каталогом конфігураційних файлів і що apply робить зі змінами в живих об'єктах.

Словник

Обирайте за фактичними умовами

Опишіть шлях розгортання, експлуатаційні ризики та чергову команду. Ми можемо визначити зміну з перевірками, шляхом відновлення та чіткою відповідальністю.

Дивитися, скільки це коштує Зв'язатися з нами →

Зробіть термінологію практичною

Запитайте, як «Що таке GitOps?» стосується вашої організації. Конкретного прикладу з поточної системи достатньо для початку розмови.

Переглянути наші послуги

welcome@engineer.company +45 50 10 61 84 Копенгаген, Данія

Візьміть цю сторінку із собою

Візитівка

Дбайливо зроблено з любов'ю — D.

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

Авторське право © 2025 – дотепер · Engineer ApS · Усі права захищено.

Надруковано з https://engineer.company/uk/glossary/gitops/