

The logo consists of a solid blue square. Inside the square, the word "GitOps" is written in a white, sans-serif font. The "G" is slightly larger and more prominent than the other letters.

GitOps

Deploy a Node.js Microservice to Kubernetes

with GitOps Pattern using Helm and Flux

00 - Who am I ?

Dridi Walid, hi@pocteo.io

- Kubernetes Consultant / Trainer
- Lead ~Ops @FnacDarty
- Ceo @Pocteo, k8s Lab





02 - Agenda

- What is GitOps ?
- Why GitOps ?
- GitOps vs CiOps,
- Key benefits of GitOps,
- Branching Strategy,
- GitOps Flow,
- GitOps Agent (Kubernetes Operators),
- Environment/Namespace Management (dev, qa, rec, ...),
- Configurations management,
- Secrets management,
- Observability.

03 - What is GitOps ?



03 - What is GitOps ?

GitOps was introduced by Alexis Richardson the co-founder and CEO of Weaveworks.



weaveworks

It's about the use of Git version control system to track and approve changes to:

- Application (Source Code),
- Infrastructure (K8s Kubernetes).

“Git as source of truth for whole
infrastructure”

Alexis Richardson
WeaveWorks

03 - What is GitOps ?

Git as source of truth for whole infrastructure

- Code (Application)
- Config (Desired State)
- Monitoring (Observed State)
- Policy (Manifests Control)

04 - Why GitOps ?

Business desire:

- velocity
- agility

Best products require:

- How fast and well defined the business can communicate user requirements for tech teams,
- How fast tech teams can act on that

Push vs Pull

06 - Key Benefits of GitOps

- For developers: Allow them playing with infrastructure configuration and code deployment using the same manner to how they manage their development process using a familiar tool: Git.
- Cost-saving: don't require wasting valuable engineer time for manual configuration,
- Open Git Pull-Request instead of Ticket,
- Self-Service: Changes made by developer are automatically applied by the GitOps operator and immediately available for the developer.
- Code Reviews: Team' members (developer/ops) leave feedbacks on changes before approving.

07 - Branching Strategy

- Using a separate Git repository to hold kubernetes manifests (config)
- Two different types of git repository:
 - Application Repo: Application source code,
 - Config Repo: Declarative manifests for configuration

Projects

Your projects 2

Starred projects 0

Explore projects

All Personal

D

pocteo-factory / env / **demo** 🔒

Owner

D

pocteo-factory / app / **demo** 🔒


Owner

08 - Single branch with multiple directories

→ the `master` branch will contain the config used in each environment

master

demo / +

**Init env**
Dridi Walid authored just now

Add README

Add LICENSE

Add CHANGELOG

Name	Last commit
dev	init tag
pre	Init env
qa	Init env
rec	Init env

09 - Multiple Repositories

→ Each team has its own ENV repository

10 - GitOps Flow

- App Repository
 - Create new feature branch,
 - Make Changes,
 - Create Pull-Request,
 - Build a Docker Image and Push to Registry
 - Update Manifest Repository with the new Docker image tag (CI_COMMIT_SHORT_SHA)
 - Create Manifest Repository Pull Request via CI Jobs
- Manifest Repository
 - Review PR
 - Merge Pull-Request on Master
 - Kubernetes Operator detects changes and apply manifests

...

- Engineers just need to merge the Pull-Request
- When you need to rollback, revert the merged Pull-Request
- NO MANUAL CHANGES (DOCKER IMAGE TAG) TO THE MANIFEST

11 - GitOps Agent (Operator)

- Watch Docker Registry for new Image Tag
- Pull Manifest Repository from Git,
- Update Docker Image in Deployment Manifest
- Commit changes to Manifest Repository (Sync)
- Apply Manifests from Inside Cluster

Open Source Operators

- <https://docs.fluxcd.io/en/1.17.0/>

12 - Environment/Namespace Management

1. Kubernetes Namespace = Environment
2. Kubernetes Namespace = Team
3. Kubernetes Namespace = Service



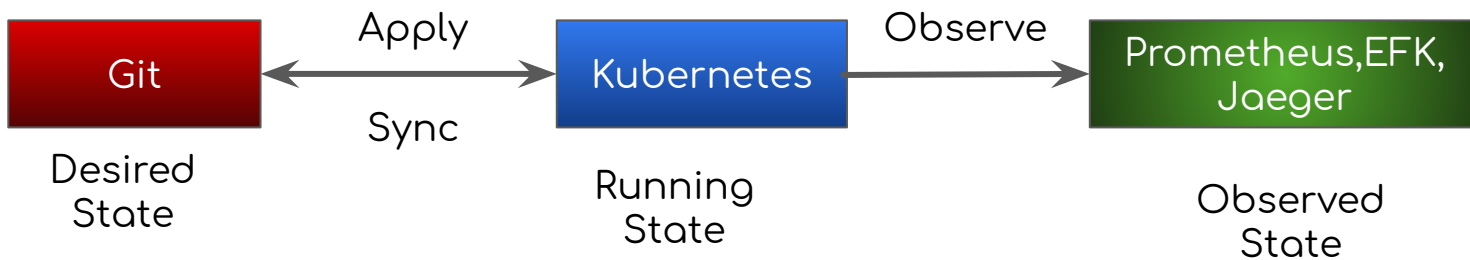
14 - Secret Management

1. [HashiCorp's Vault](#)
2. [Sealed Secrets from Bitnami](#)

“Observability is like TDD for production. Never accept a pull request if you don’t know how to identify if it is working”

[Adriano Bastos](#)

15 - Observability



Two sources of truth to well adopt GitOps

- Git provides a source of truth for the desired state of the system,
- Observability provides a source of truth for the actual production state of the running system,

→ GitOps use both to manage our applications

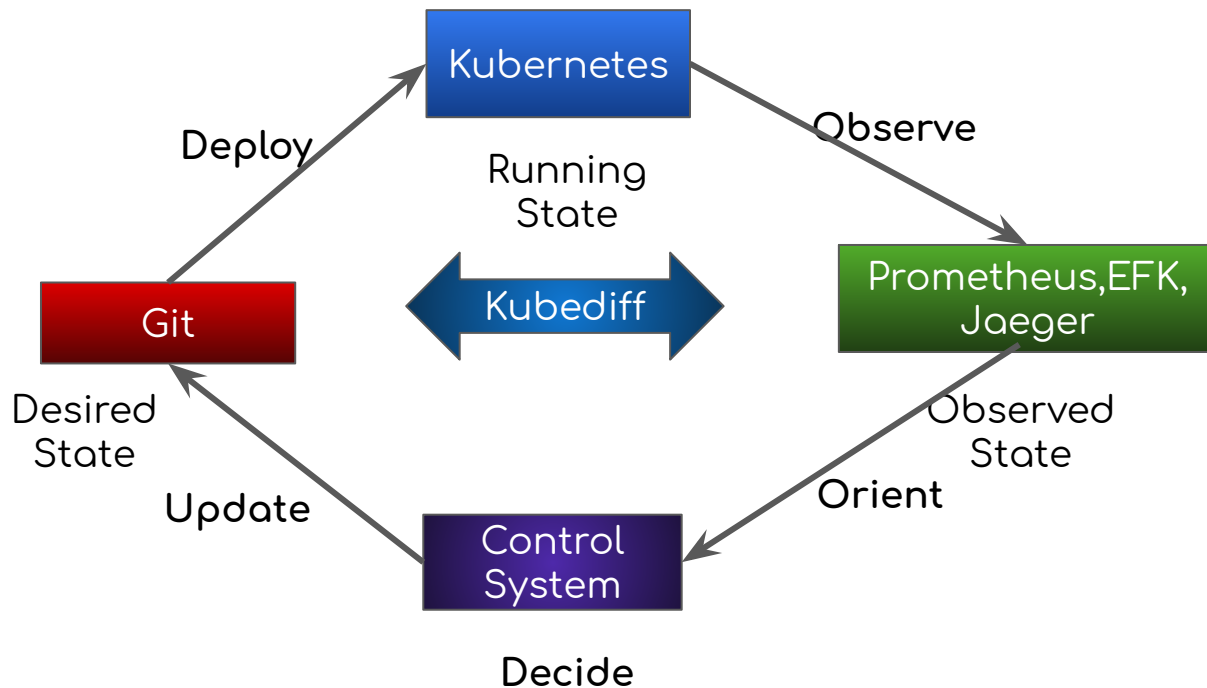
Diffs check Observed vs Desired State

- Validating that our currently observed production state corresponds to the system desired state in Git
- Instantly alerting us if it is not - via Prometheus & Slack and informing us of the nature of the divergence

Tools

- Diffs do not require custom creation or coding.
- Kubediff: checks the cluster periodically and alerts if the number changes from 4, In general terms, kubediff turns yaml files into queries on running state.

15 - Observability



Thank you for listening!

Questions?