
23.10.2020
GITOPS
the Kubernetes way
SPARKFABRIK

The session:

- Some basic concepts
- GitOps? what is it?
- The GitOps "tools"
- The GitOps pipeline
- Conclusions


THE BASICS OF CICD - KUBERNETES - DECLARATIVE MODEL

WHAT IS ***GIT****?*

The **most widely used** modern version control system in the world today.

More one GIT: https://www.atlassian.com/git/tutorials/what-is-git

CI: *Continuous Integration*

- A software development practice where **all developers merge code changes in a central repository** (Git).
- Each change in code (commit) **triggers an automated build-and-test stage** for the given repo and **provides feedback to the developer(s) who made the change.**
- **Automates the build and unit test process of new code changes**

CD: *Continuous Delivery*

A software engineering approach in which **teams produce software in short cycles**, ensuring that the **software can be reliably released at any time** and, when releasing the software, doing so ***manually***.

More one CONTINUOUS DELIVERY: https://en.wikipedia.org/wiki/Continuous_delivery

***CD**: Continuous Deployments*

A software engineering approach in which software functionalities are delivered frequently through ***automated deployments***.

More one CONTINUOUS DEPLOYMENTS: https://en.wikipedia.org/wiki/Continuous_deployment


GITOPS
CICD
Continuous Integration
+
Continuous Delivery / Deployment

KUBERNETES

Kubernetes is a portable, extensible, open-source platform for managing containerized workloads and services, that facilitates both **declarative configuration** and **automation**.

More one KUBERNETES: https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/

Declarative model

You describe what you want to be achieved, as opposed to how to get there

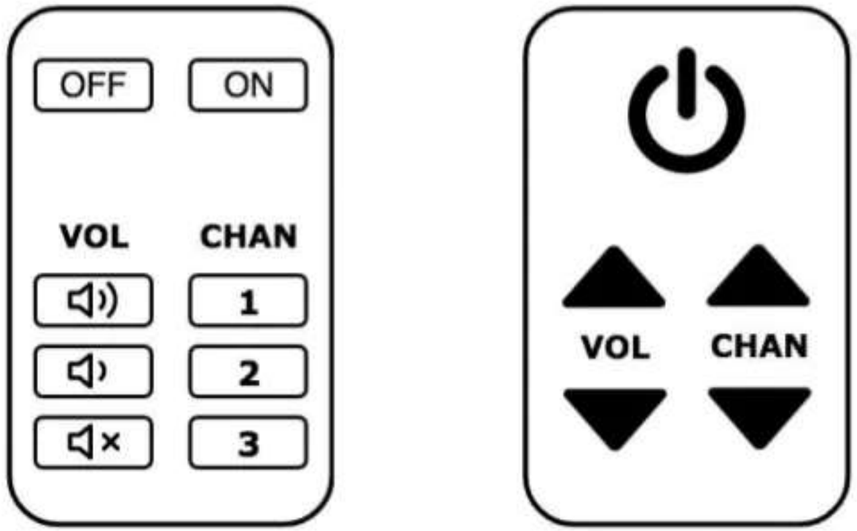
```
kubeclt apply -f nginx-deployment.yaml
```

Imperative model

You describe a sequence of instructions to manipulate the state of the system to reach your desired state

```
kubectl create deployment nginx-imperative --image=nginx:latest
kubectl scale deployment/nginx-imperative --replicas 3
kubectl annotate deployment/nginx-imperative environment=prod
kubectl annotate deployment/nginx-imperative organization=sales
```

An Imperative vs. Declarative Remote Control



https://livebook.manning.com/book/gitops-and-kubernetes/chapter-1/v-6/163

KUBERNETES *Controllers*

Controllers are control loops that watch the state of your cluster, then make or request changes where needed.

Each controller tries to **move the current cluster state** closer **to the desired state.**

The desired state is what is described declaratively in the resource's manifest.

More one K8S CONTROLLERS: https://kubernetes.io/docs/concepts/architecture/controller/


GITOPS

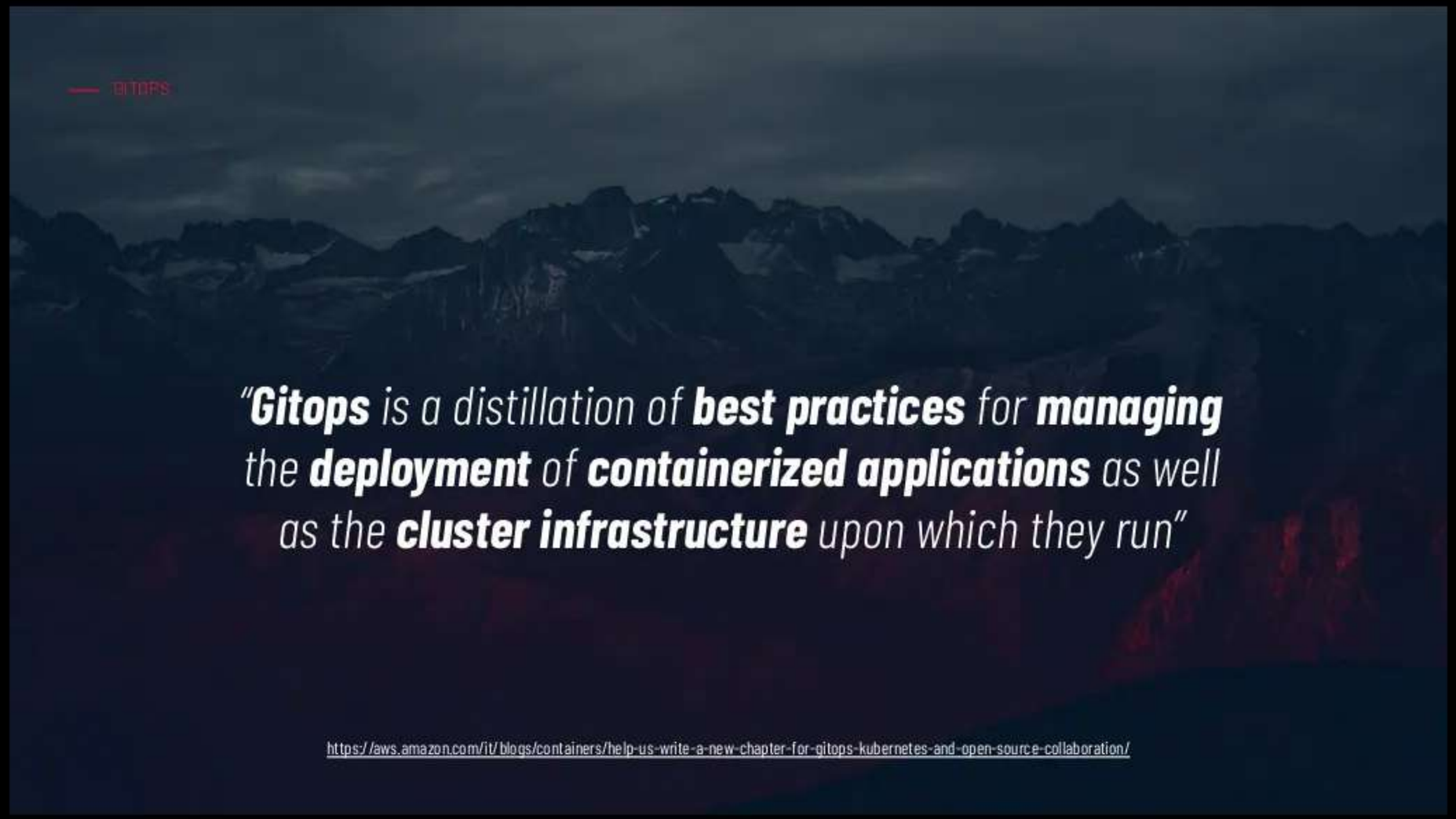
GITOPS

What is GitOps?

Is a way of implementing **Continuous Deployment / Delivery** for **cloud native applications.**

It **focuses** on a **developer-centric experience** when operating infrastructure, by using tools developers are already familiar with, including Git and Continuous Deployment tools.

More one GITOPS: https://www.gitops.tech/

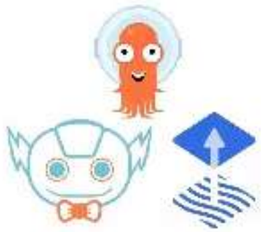

GITOPS
"Gitops is a distillation of best practices for managing
the deployment of containerized applications as well
as the cluster infrastructure upon which they run"
https://aws.amazon.com/it/blogs/containers/help-us-write-a-new-chapter-for-gitops-kubernetes-and-open-source-collaboration/


GITOPS PRINCIPLES

Principles of GitOps

The entire system is described **declaratively**

The canonical desired system state is **versioned** in git



Approved changes can be **automatically applied** to the system

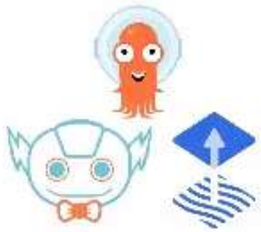
Software agents ensure correctness and alert (diffs & actions)

https://gitops-community.github.io/kit/#the-4-principles-of-gitops

Principles of GitOps

The entire system is described **declaratively**

The canonical desired system state is **versioned** in git



Approved changes can be **automatically applied** to the system

Software agents ensure correctness and alert (diffs & actions)

https://gitops-community.github.io/kit/#the-4-principles-of-gitops

System is described ***declaratively***

It allows to **describe the entire system** (services and applications) **as configuration code**.

Kubernetes, given its declarative nature and the controller pattern, is a perfect tool to do **GitOps**.

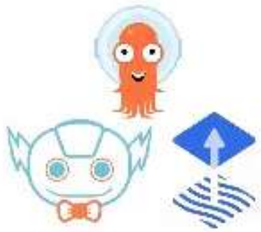
Principles of GitOps

1

The entire system is described **declaratively**

2

The canonical desired system state is **versioned** in git



Approved changes can be **automatically applied** to the system

Software agents ensure correctness and alert (diffs & actions)

https://gitops-community.github.io/kit/#the-4-principles-of-gitops

- ***Git*** *as the* ***single source of truth*** *of the system*
- ***Git*** *as the* ***single place where we operate*** *(create, change and destroy) all environments*

Principles of GitOps

1

The entire system is described **declaratively**

2

The canonical desired system state is **versioned** in git

3

Approved changes can be **automatically applied** to the system

Software agents ensure correctness and alert (diffs & actions)

Changes can be ***automatically*** *applied*

Responsible of the automation is a **GitOps Operator.**

It's a **Kubernetes operator**, *a server-side controller*, that read the desired state of a system (i.e. the manifests in a git repo) and continually tries to make the actual state of the system match those manifests.

Principles of GitOps

1

The entire system is described **declaratively**

2

The canonical desired system state is **versioned** in git

3

Approved changes can be **automatically applied** to the system

4

Software agents ensure correctness and alert (diffs & actions)

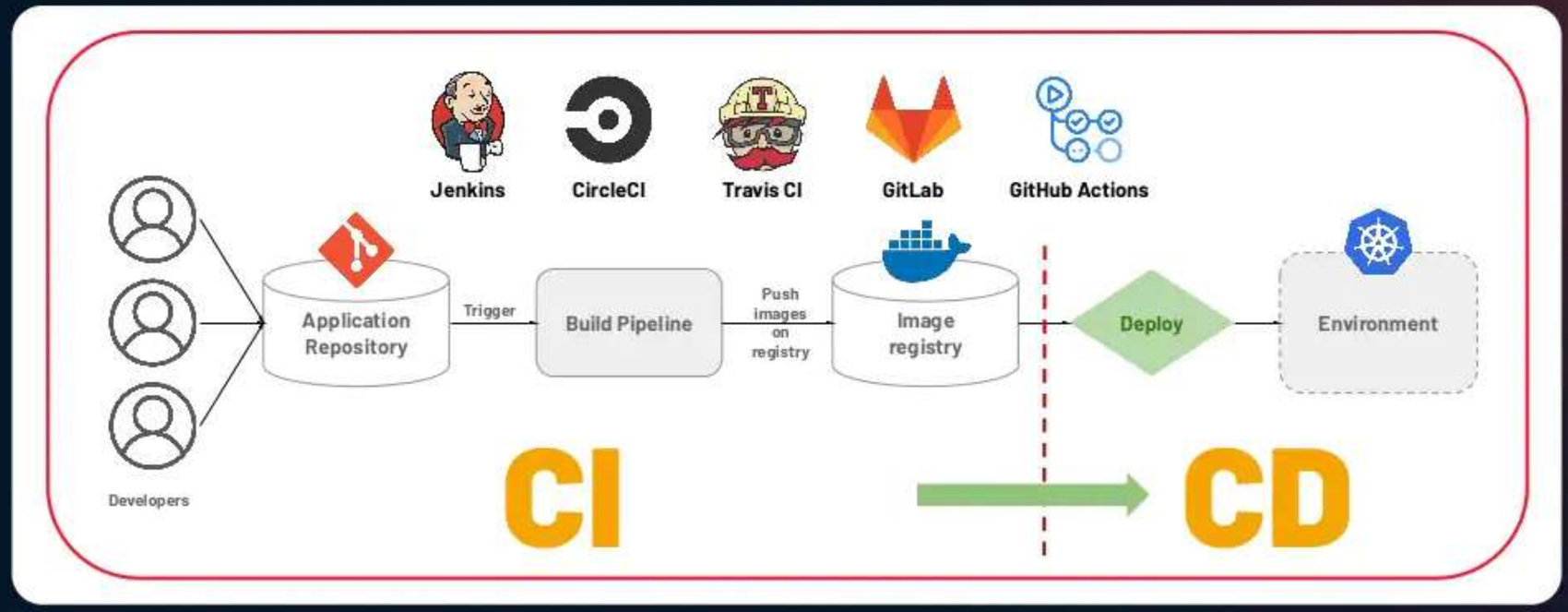

GITOPS
What do I need to do GitOps in K8s?

GITOPS

What do I need to do GitOps in K8s?

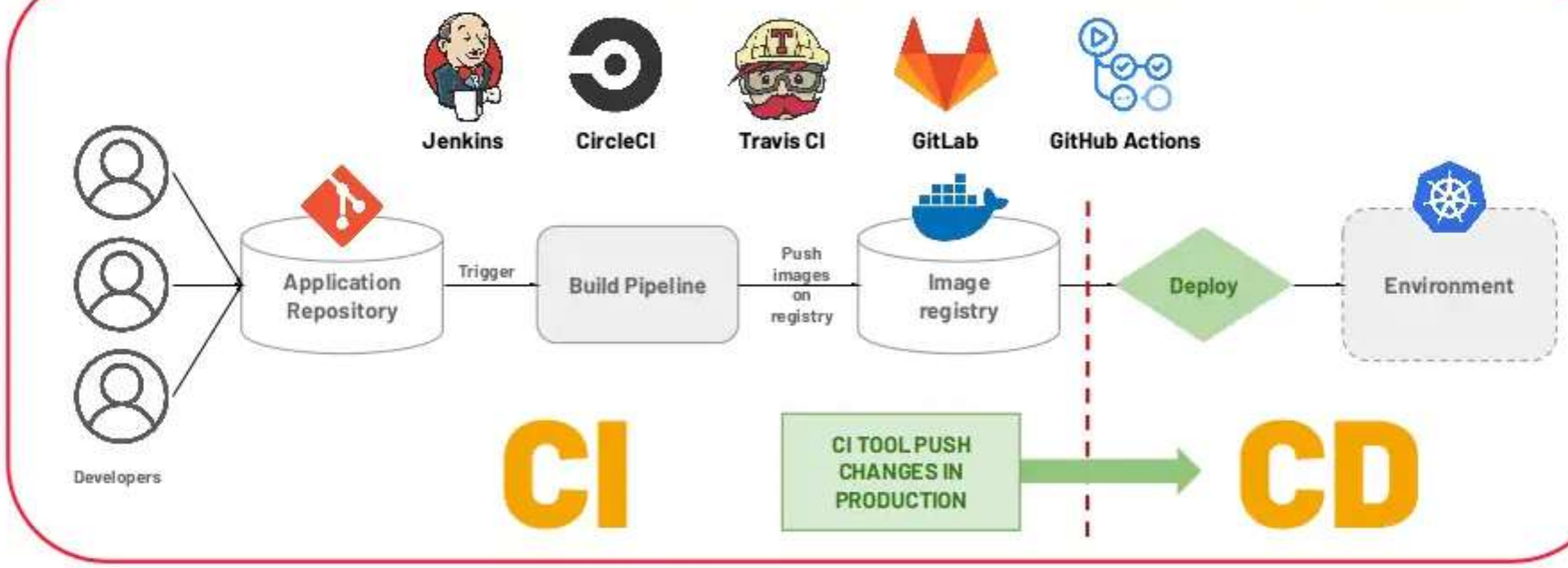

GITOPS OPERATORS

A tipical Kubernetes CI/CD pipeline (the push model)

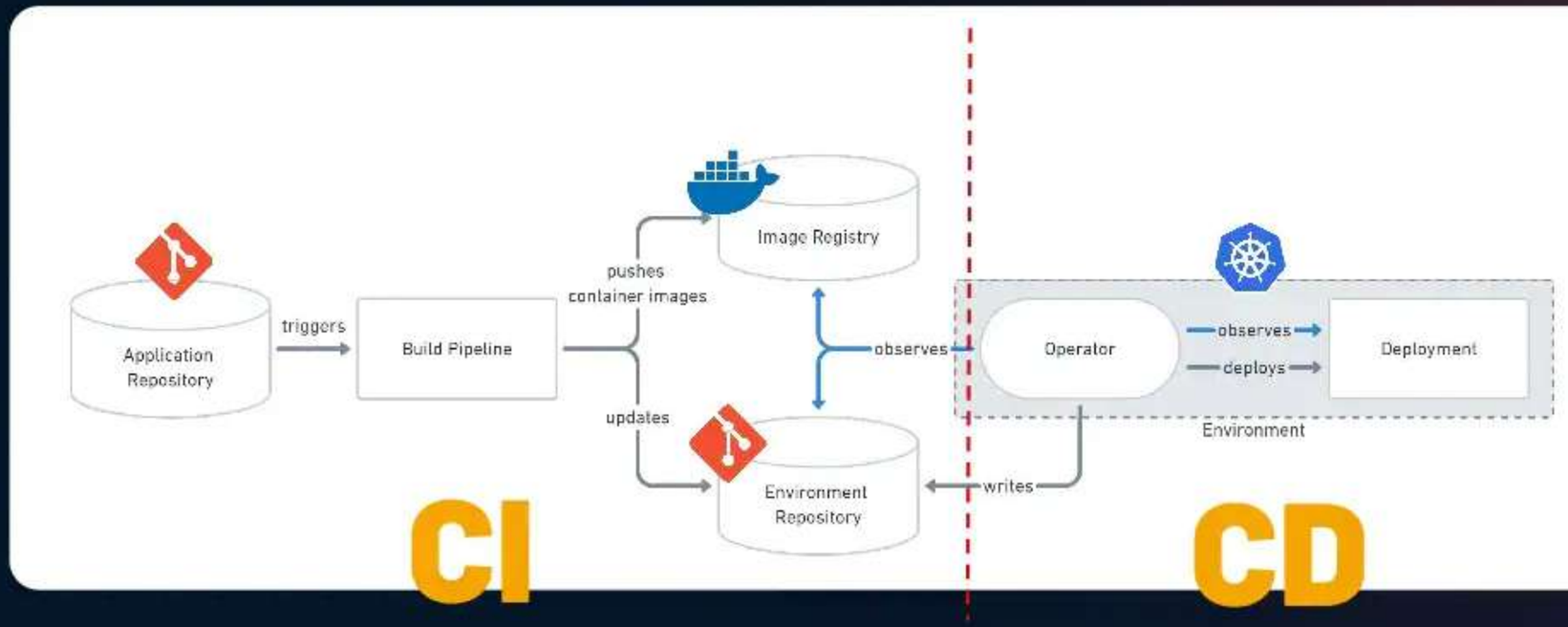


CICD PIPELINE AND GITOPS

A tipical Kubernetes CI/CD pipeline (the push model)

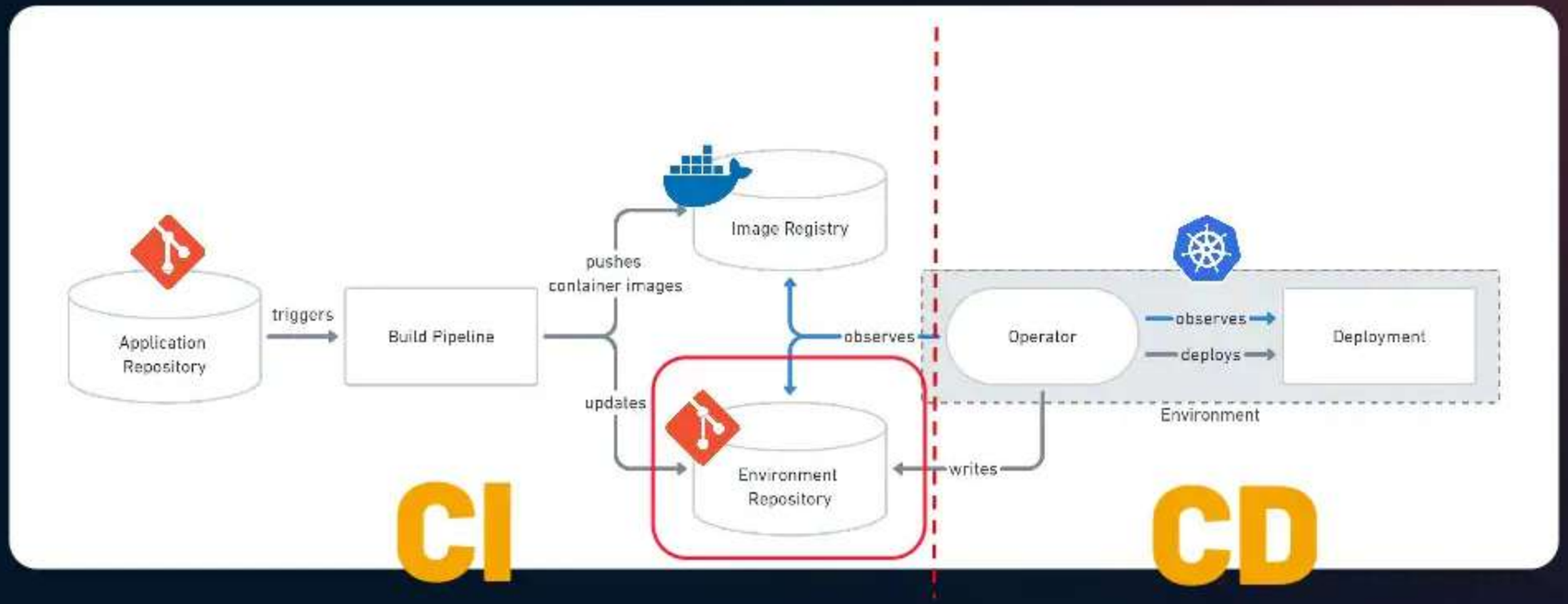


The GitOps **pull-based** model for CD



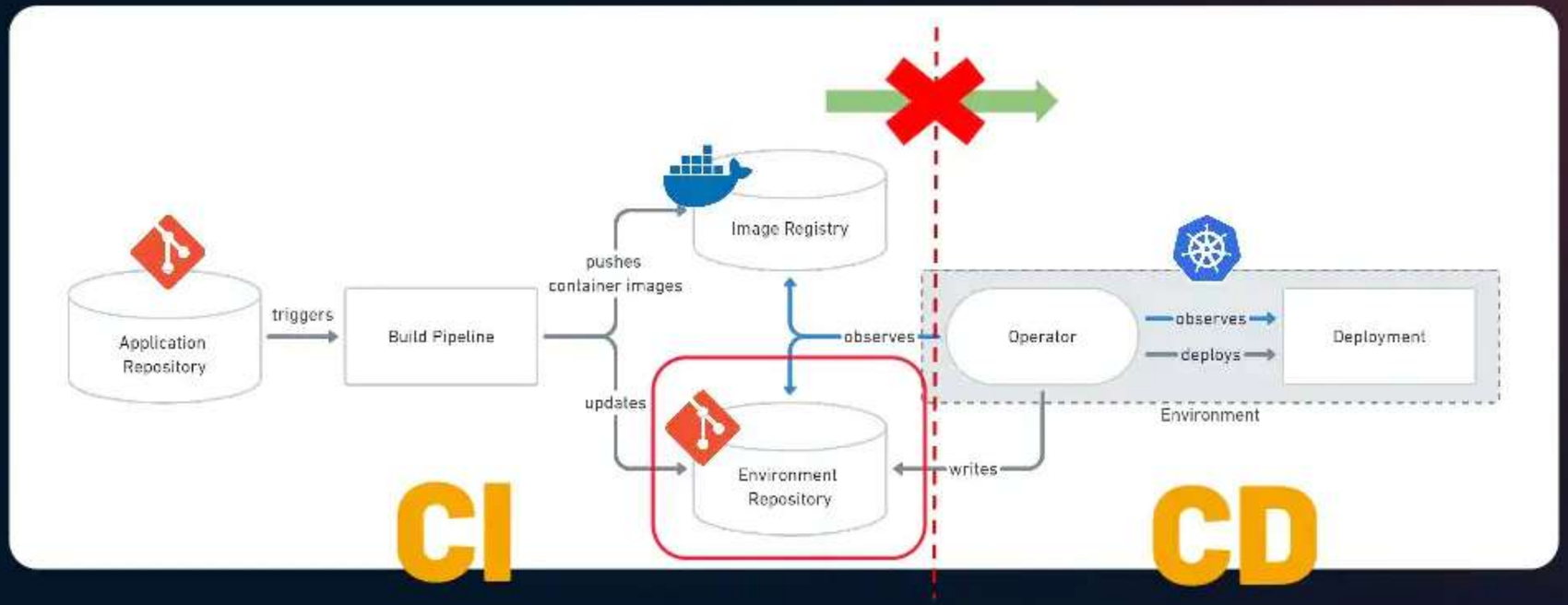
https://www.gitops.tech/#pull-based-deployments

The GitOps **pull-based** model for CD



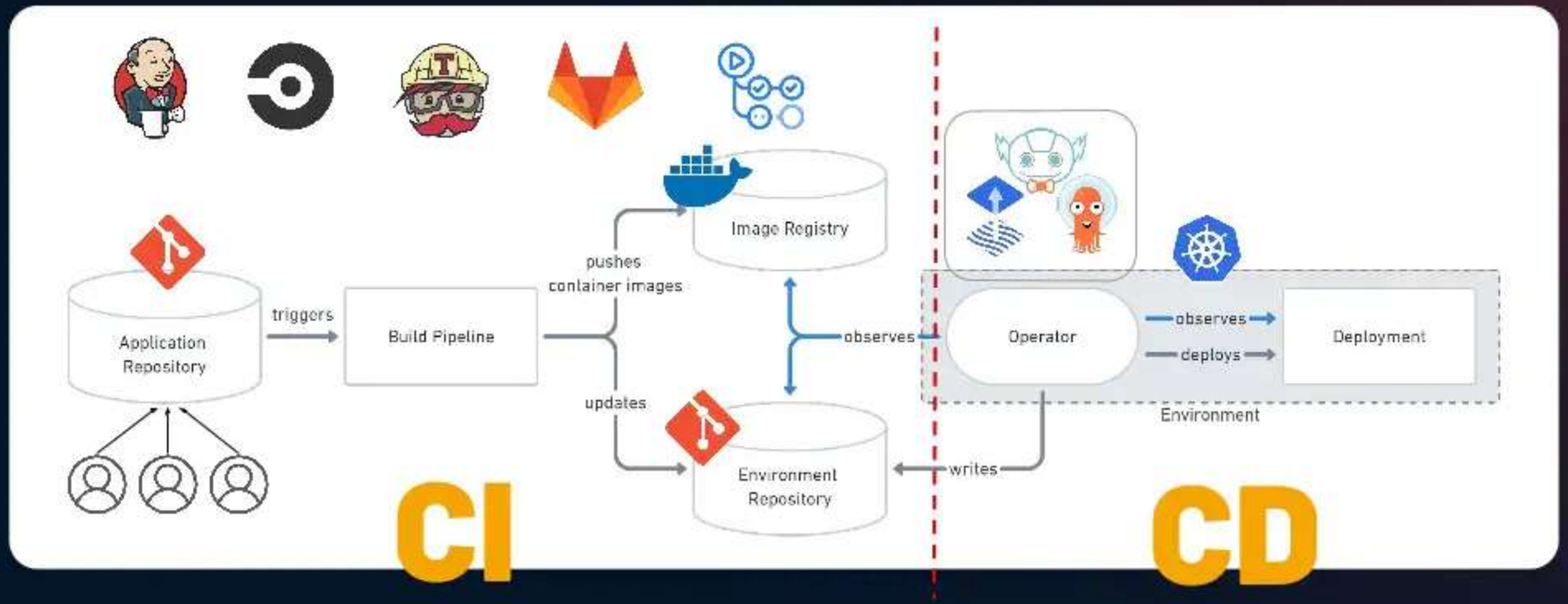
https://www.gitops.tech/#pull-based-deployments

The GitOps **pull-based** model for CD



https://www.gitops.tech/#pull-based-deployments

The GitOps **pull-based** model for CD



https://www.gitops.tech/#pull-based-deployments

GITOPS BENEFITS

GitOps benefits

- **Improve system observability**: allow running state and desired state to be observable
- **Improve security**: no need to expone to internet K8s API server or to give access to developers to the application cluster.
- **Simpler disaster recovery** and rollback procedure which is consistent with the normal deployment experience (**git revert** commit)
- **Increased Productivity**: what can be described can be automated - continuous deployment automation with an integrated feedback control loop via the **operator**

GitOps: the bad and the ugly

- **You need to structure the environment repo and your pipelines to prevent concurrent push on the same repo (since remote may be out of sync)**: multiple CI processes can end up writing to the same GitOps repo, causing conflicts or errors. This is may happen for example if you have configured a single repository environment to describe describe all applications deployed in a given cluster (Flux CD V1 for example supported only one env repo)

- **Doesn't give you an opinionated way to deal with secret management** (Git repositories are not great places to store secrets, as you have to encrypt and decrypt them)

https://blog.container-solutions.com/gitops-limitations
https://www.reddit.com/r/kubernetes/comments/imgqoj/gitops_the_bad_and_the_ugly/

GitOps - Suggestions?

- **Use two repos**: one for app source code, another for manifests.
- **Never store secrets or keys in manifests repo** as plain text and yes, base64 is plain text! (plan how to manage secrets)


THANK YOU!