acend

Argo CD - Basics

Raffi, Chrigu & Thämu
17. November 2021

experience knowledge

Today’s program

09:00 – 09:45	Introduction and presentation about ArgoCD Basics
09:45 – 12:00	Hands-on Labs 1 - 3
12:00 – 13:00	Lunch Break and refresh
13:10 – 16:45	Hands-on Labs 4 - 7
16:45 – 17:00	Wrap-up


App of App
Helm
Kustomize
Git
Cloud Native
GitOps
Deployment
Application Sets
Sync Windows
Argo CD
K8S
Jsonnet
Kubernetes
Continuous Deployment
Sync Phases Waves
Resource hooks

Agenda

1. Kubernetes recap
2. Introduction to ArgoCD
3. Secret Management
4. Labs

ArgoCD Basics

Kubernetes recap

Kubernetes recap

Cluster

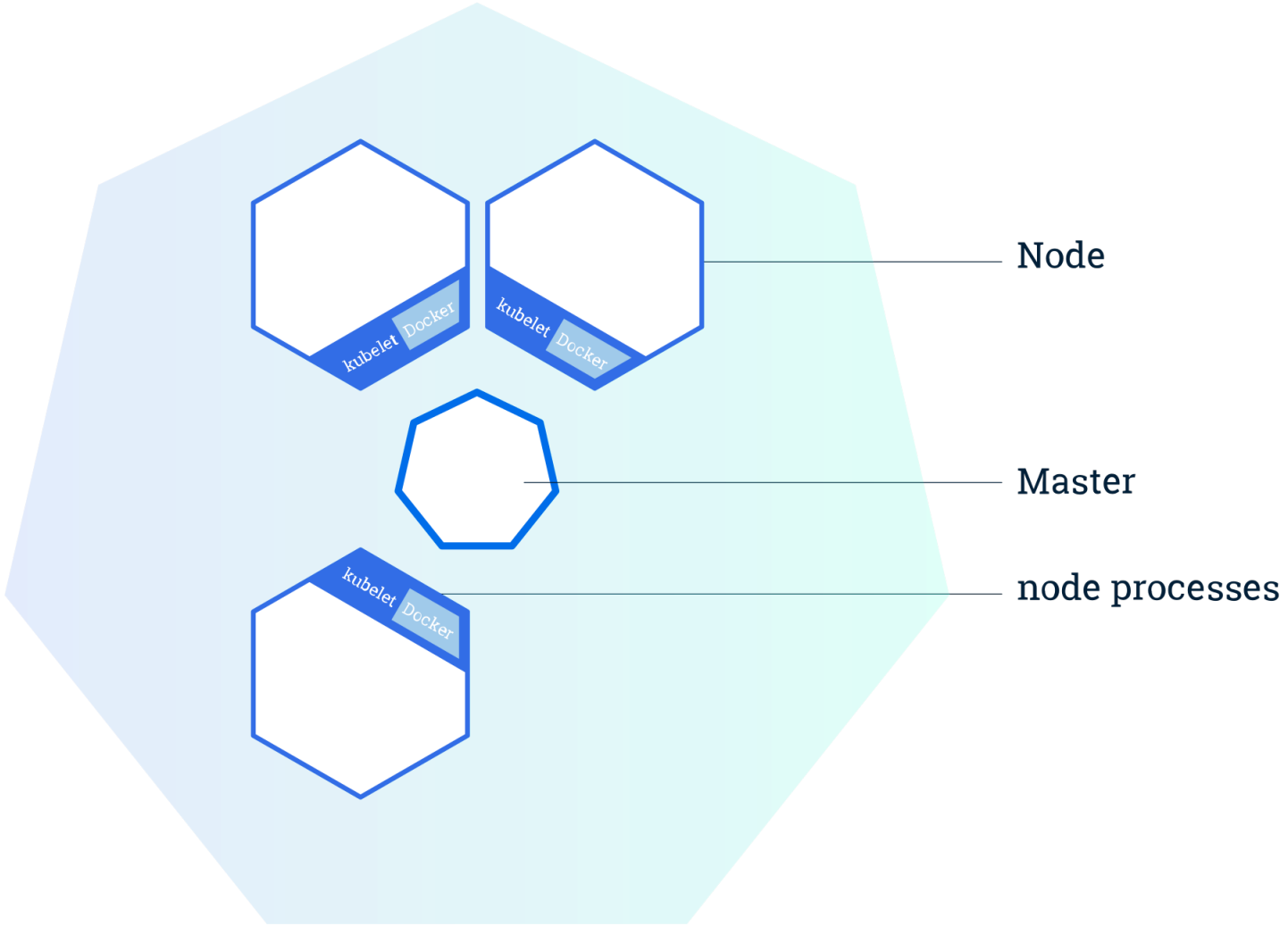


Image © 2020 The Kubernetes Authors | Documentation Distributed under CC BY 4.0

Kubernetes recap

Deploy an app

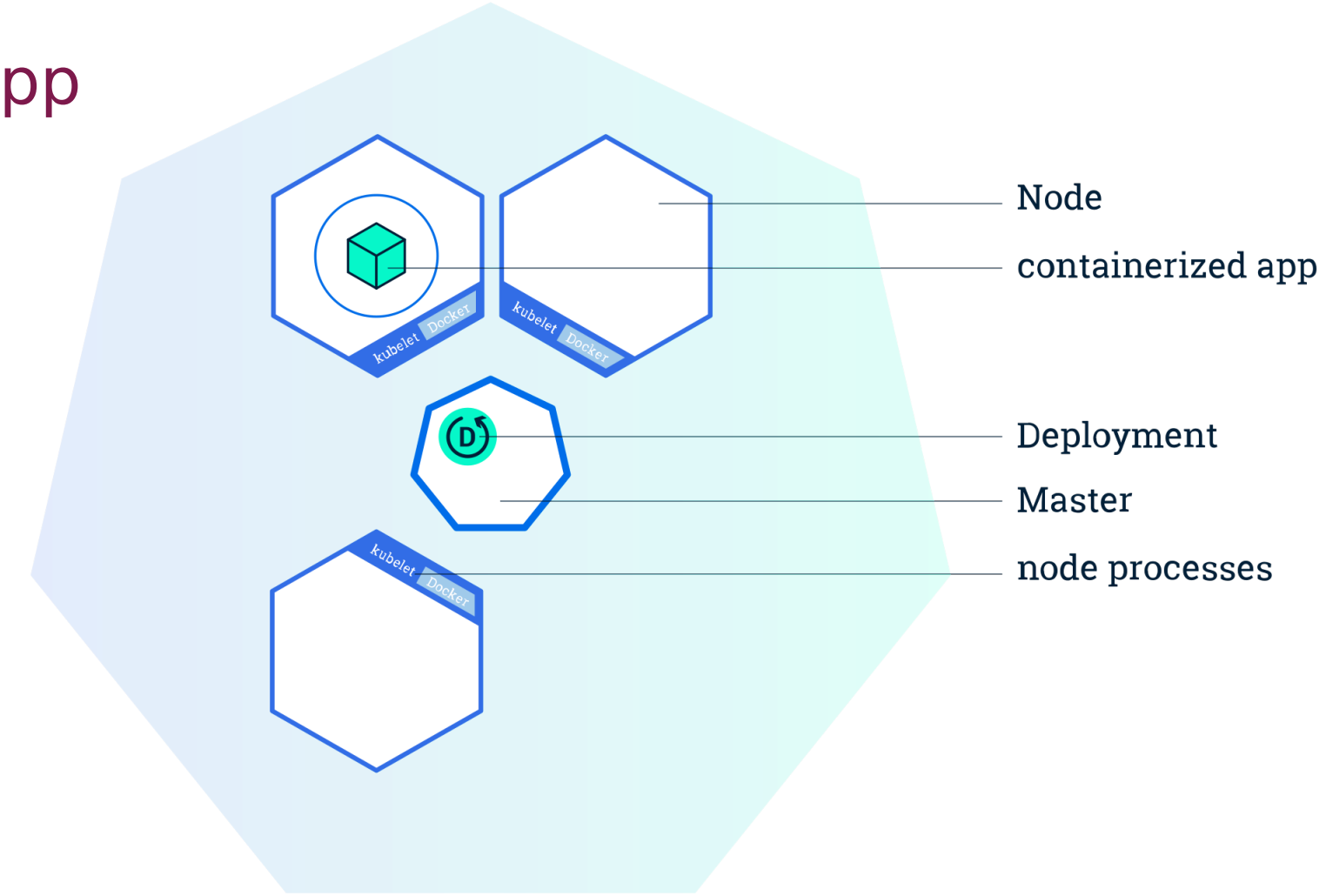


Image © 2020 The Kubernetes Authors | Documentation Distributed under CC BY 4.0

Kubernetes recap

Pods

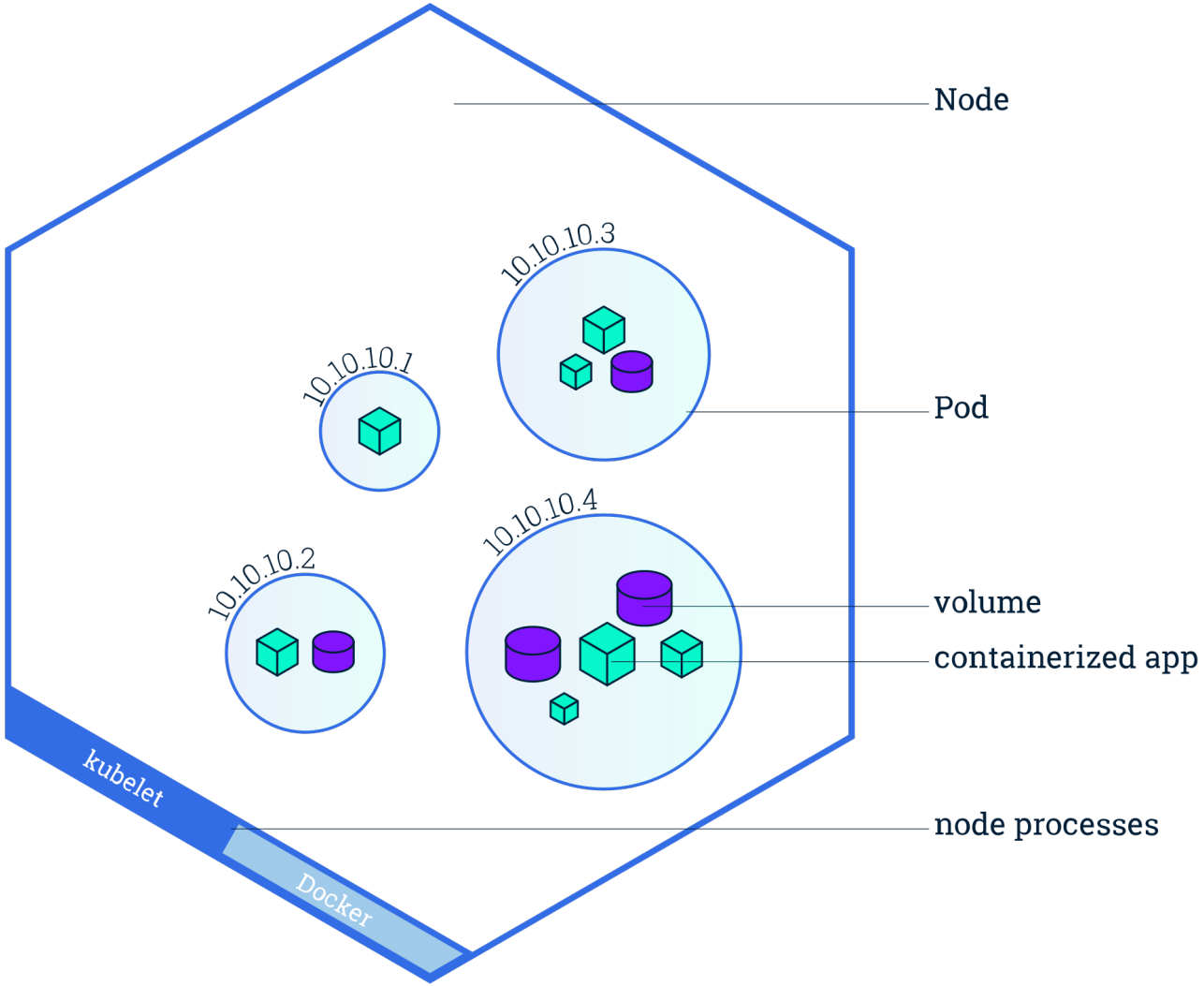


Image © 2020 The Kubernetes Authors | Documentation Distributed under CC BY 4.0

Kubernetes recap

Pods

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    name: nginx
spec:
  containers:
    - name: nginx
      image: nginx
```

Kubernetes recap

Deployments

- One of the most common workloads
- Use ReplicaSets as building block
- Add flexible life cycle management functionality

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend
  labels:
    app: guestbook
    tier: frontend
spec:
  replicas: 3
  strategy:
    type: RollingUpdate
  selector:
    matchLabels:
      tier: frontend
  template:
    metadata:
      labels :
        tier: frontend
    spec:
      containers:
      - name: nginx
        image: nginx
```

Kubernetes recap

Services

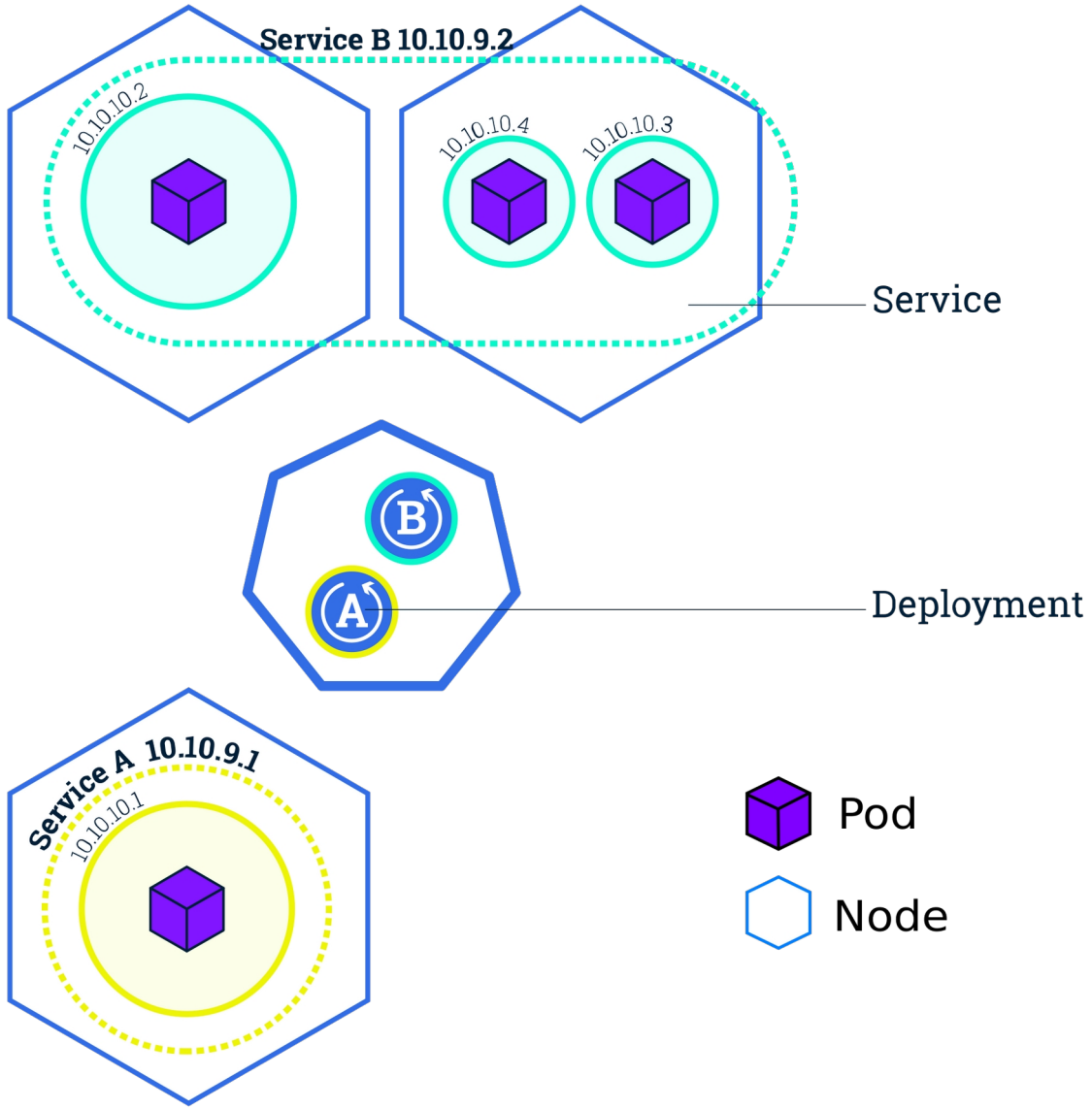


Image © 2020 The Kubernetes Authors | Documentation Distributed under CC BY 4.0

Kubernetes recap

Scaling 1

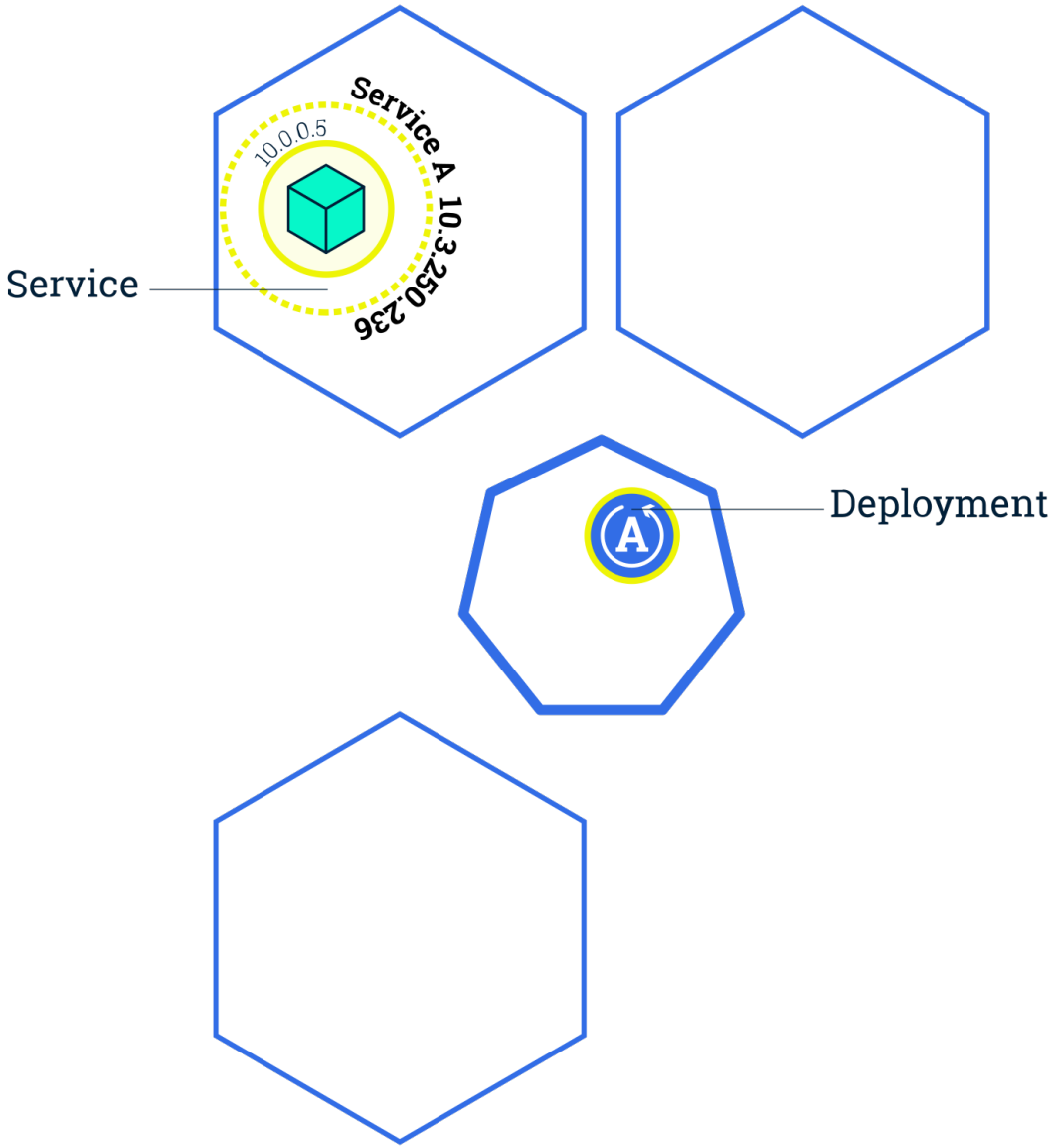


Image © 2020 The Kubernetes Authors | Documentation Distributed under CC BY 4.0

Kubernetes recap

Scaling 2

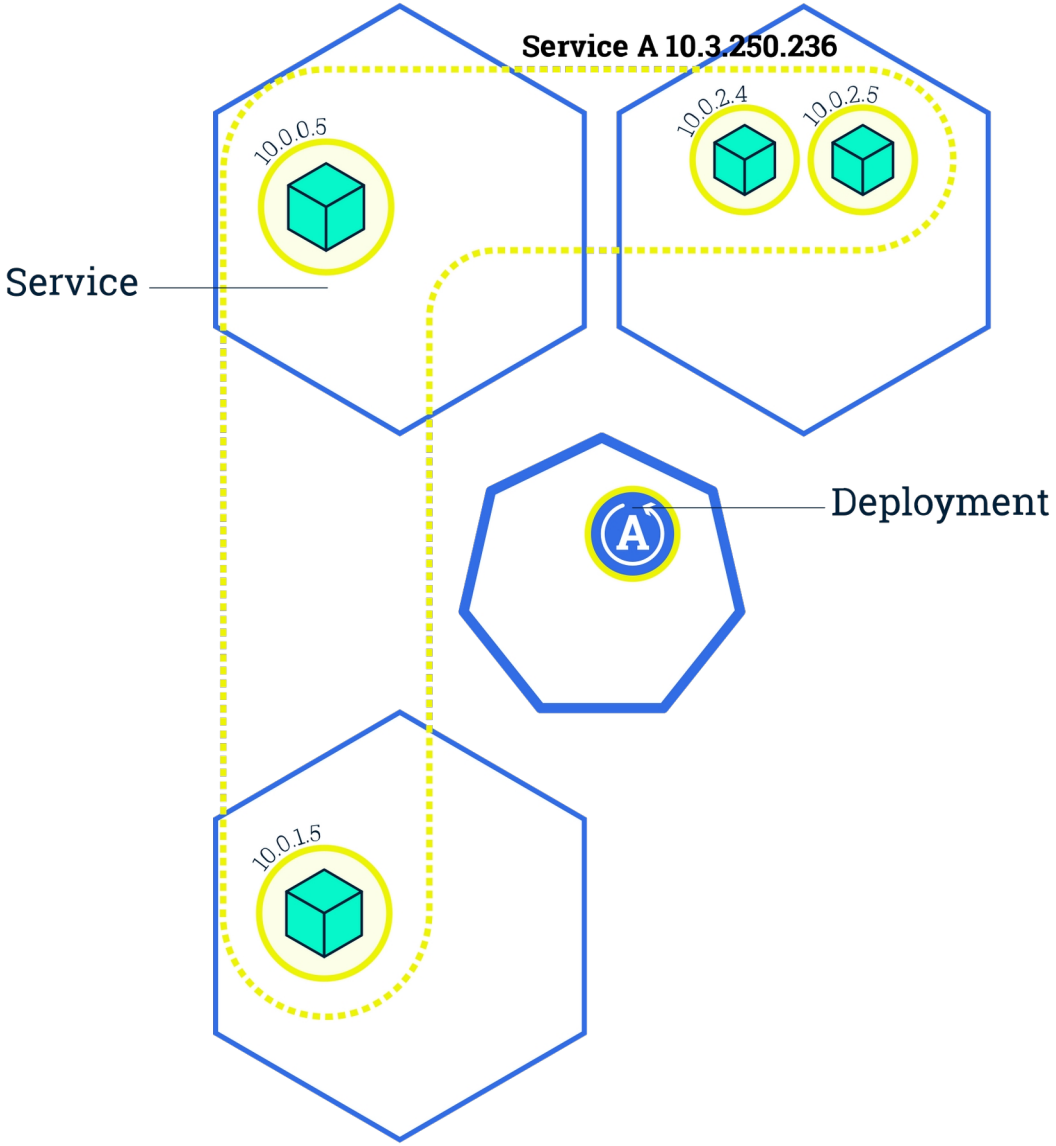


Image © 2020 The Kubernetes Authors | Documentation Distributed under CC BY 4.0

Argo CD - Training

Introduction to ArgoCD

What is Argo CD?

Argo CD is a declarative, GitOps continuous delivery tool for Kubernetes.

What is Argo CD?

¬Argo CD helps managing K8s resources with familiar tools and processes – Git

¬Argo CD is **not** a CI pipeline tool

¬Argo CD is the interface to the K8s Cluster which manages K8s resources

What is Argo CD?

- K8s Manifests
- Environments
- Secrets
- ...

K8s Cluster

K8s Cluster

K8s Cluster

What is Continuous Deployment?

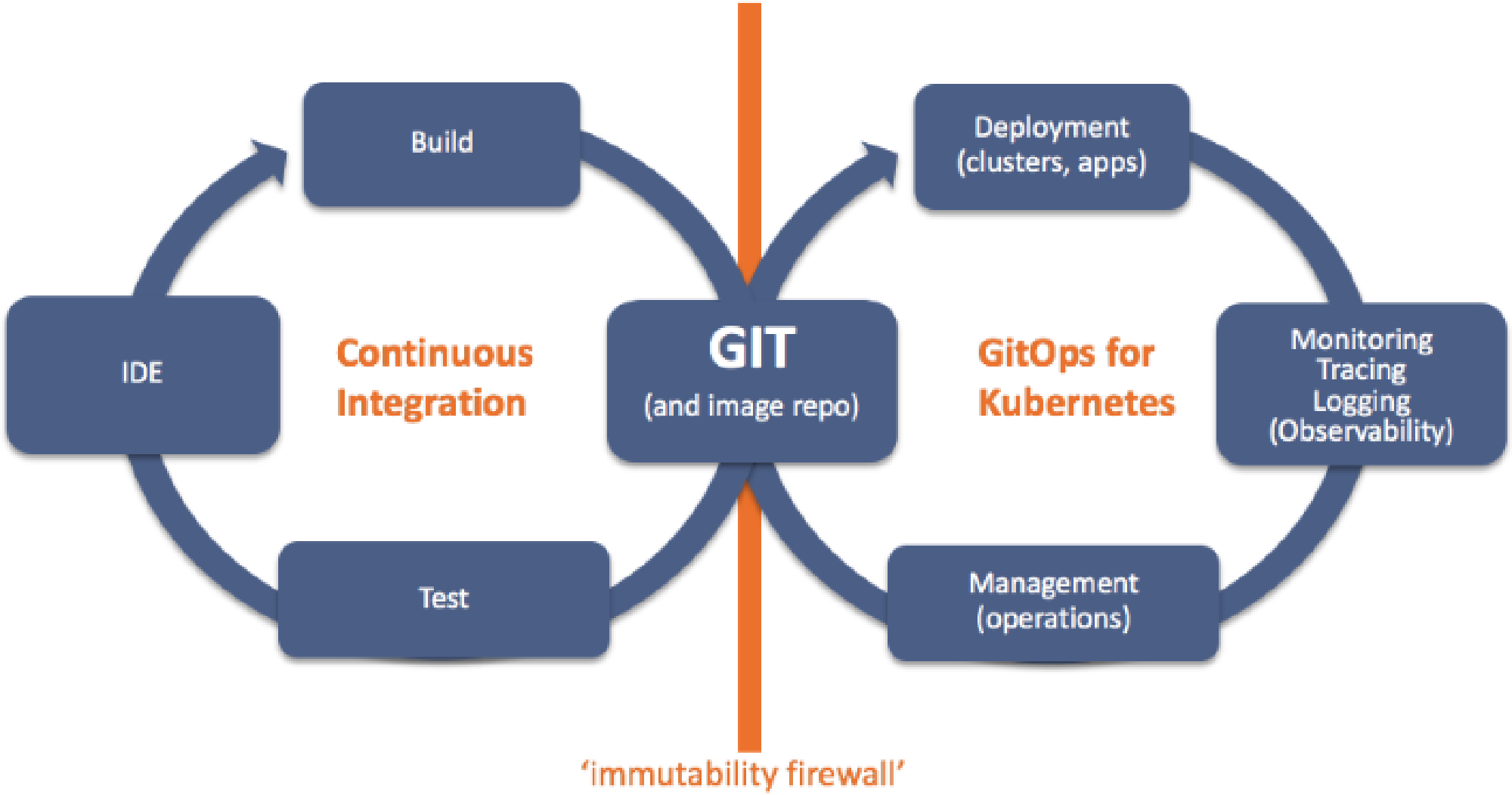
- Continuous Delivery without the push of a button
- Every commit will be deployed to production
- Rolling Deployments
- Canary Releasing
- Blue – Green Deployment

Why Argo CD?

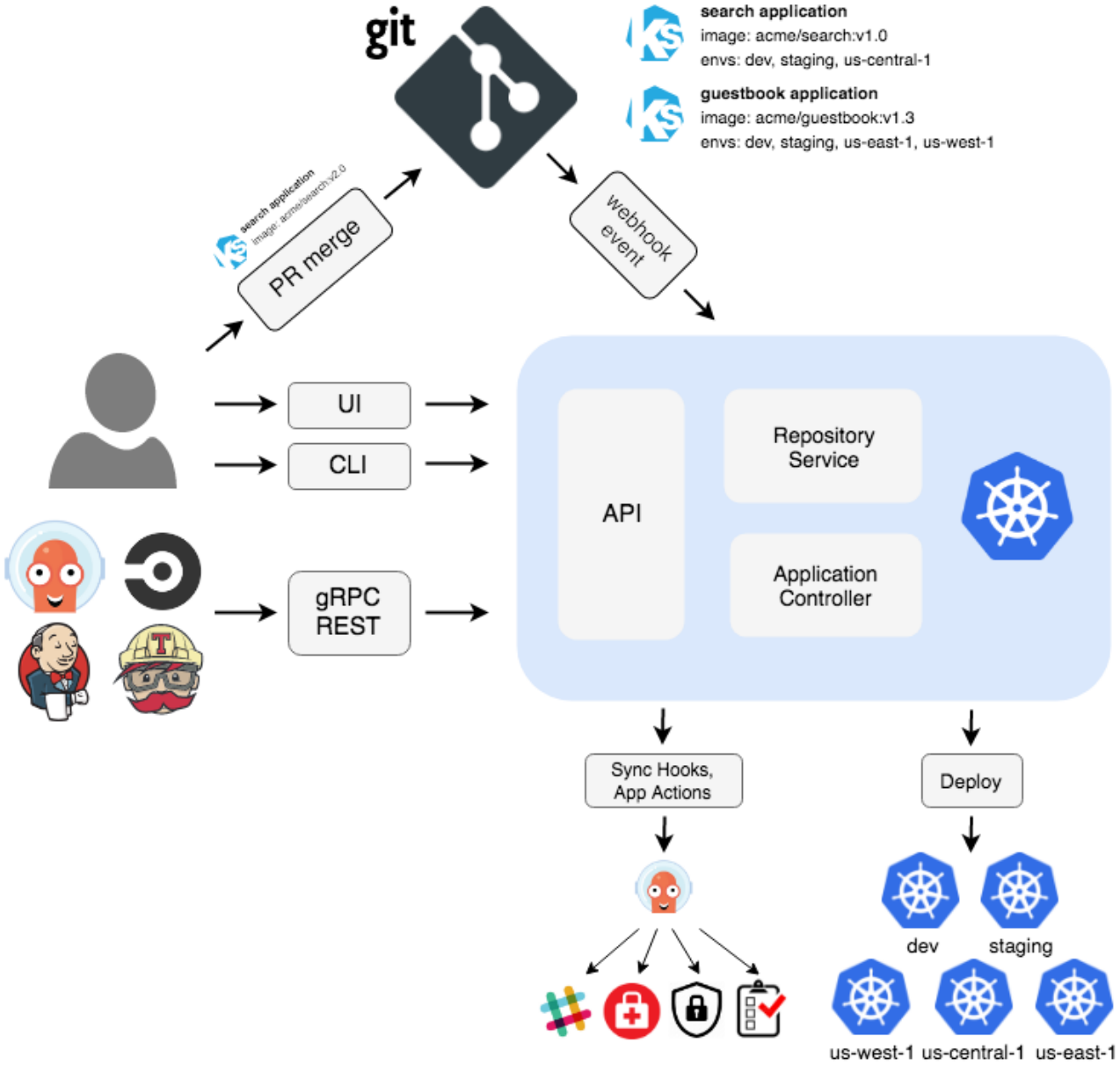
Application definitions, configurations, and environments should be declarative and version controlled. Application deployment and lifecycle management should be automated, auditable, and easy to understand.

GitOps

- Declarative deployments
- Git as the central source of truth
- GitOps diffs declared state (in Git) with observed state (live system)
- Changes are observable, verifiable, and audited
- “Operations by pull requests”: all intended operations are commited by PRs
- Rollback & disaster recovery



Source: https://www.weave.works/blog/what-is-gitops-really



Source: https://argoproj.github.io/argo-cd/

Argo CD tools

Kubernetes manifests can be spcified in several formats:

- kustomize applications
- helm charts
- jsonnet files
- Plain directory of YAML/json manifests
- Any custom config management tool configured as a config management plugin
  - e.g. kustomize with OpenShift Manifest support

Argo CD core concepts

- Clusters (Servers): Kubernetes clusters to deploy on
- Repositories: pre-configured git repos, incl. credentials
- Applications: A group of Kubernetes resources, represented in a git repository
- Projects: a logical grouping of Argo CD applications
  - Project-based restrictions
  - Useful when multiple Teams work with the same Argo CD instance

Separating Config Vs. Source Code Repositories

¬Highly recommended: Clean Separation of Code and Configruration

¬Cleaner Audit Log – Cleaner Git History

¬Application is in more than one Git Repo – Config and Deployment just in one

¬Separation of access

¬CI Jobs get a lot more complicated – Separation of concern

Immutable Manifests for helm and kustomize

¬Templating tools allow to use upstream manifests

¬Make sure to reference fix versions

```
bases:
- github.com/argoproj/argo-cd//manifests/cluster-install
```

Upstream might change

```
bases:
- github.com/argoproj/argo-cd//manifests/cluster-install?ref=v0.11.1
```

Version is fixed

CI integration

Can Argo CD be integrated with your CI pipeline?

¬Yes,

  ¬via Git

  ¬and webhooks

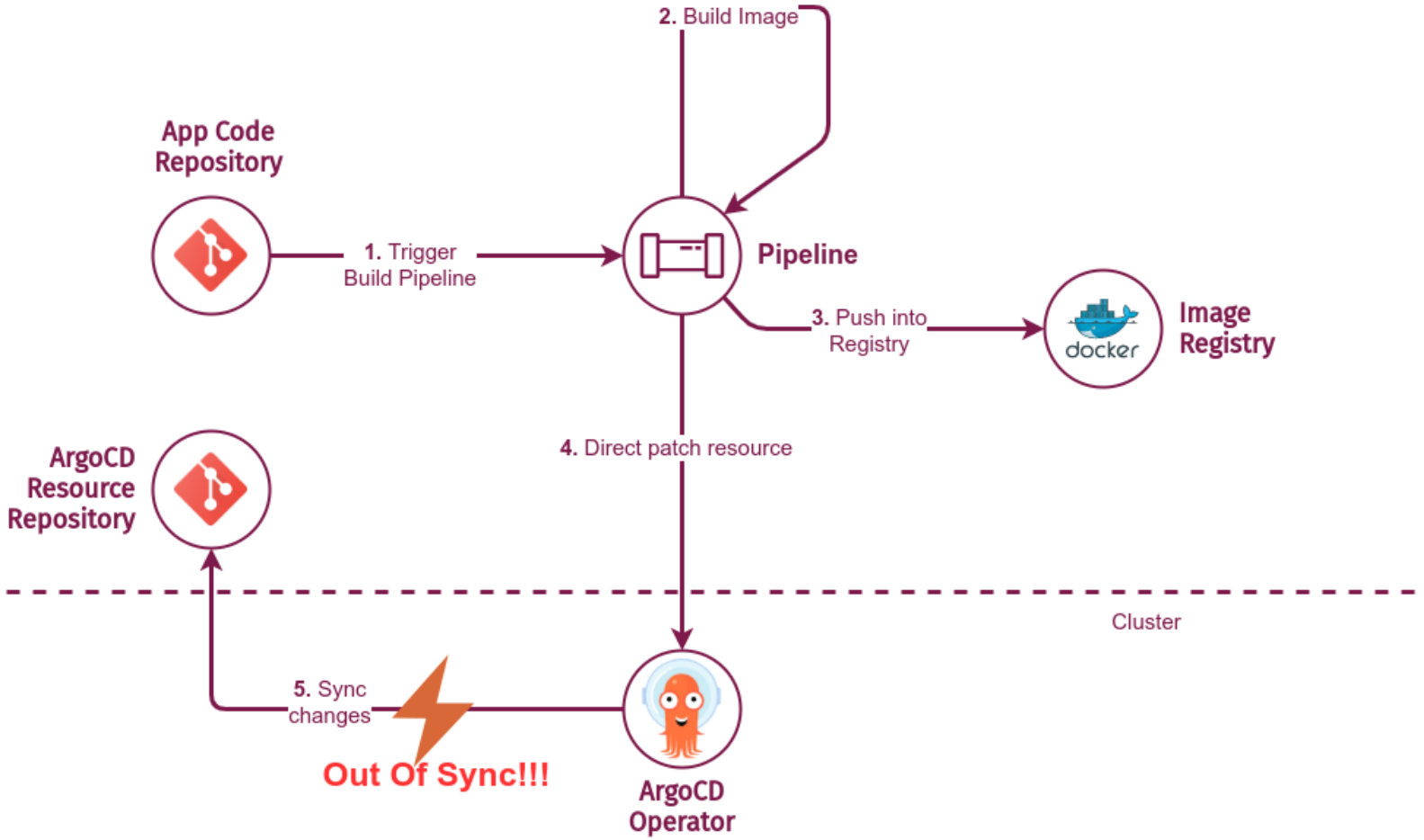
¬... but ...

  ¬its core principle is declarative,

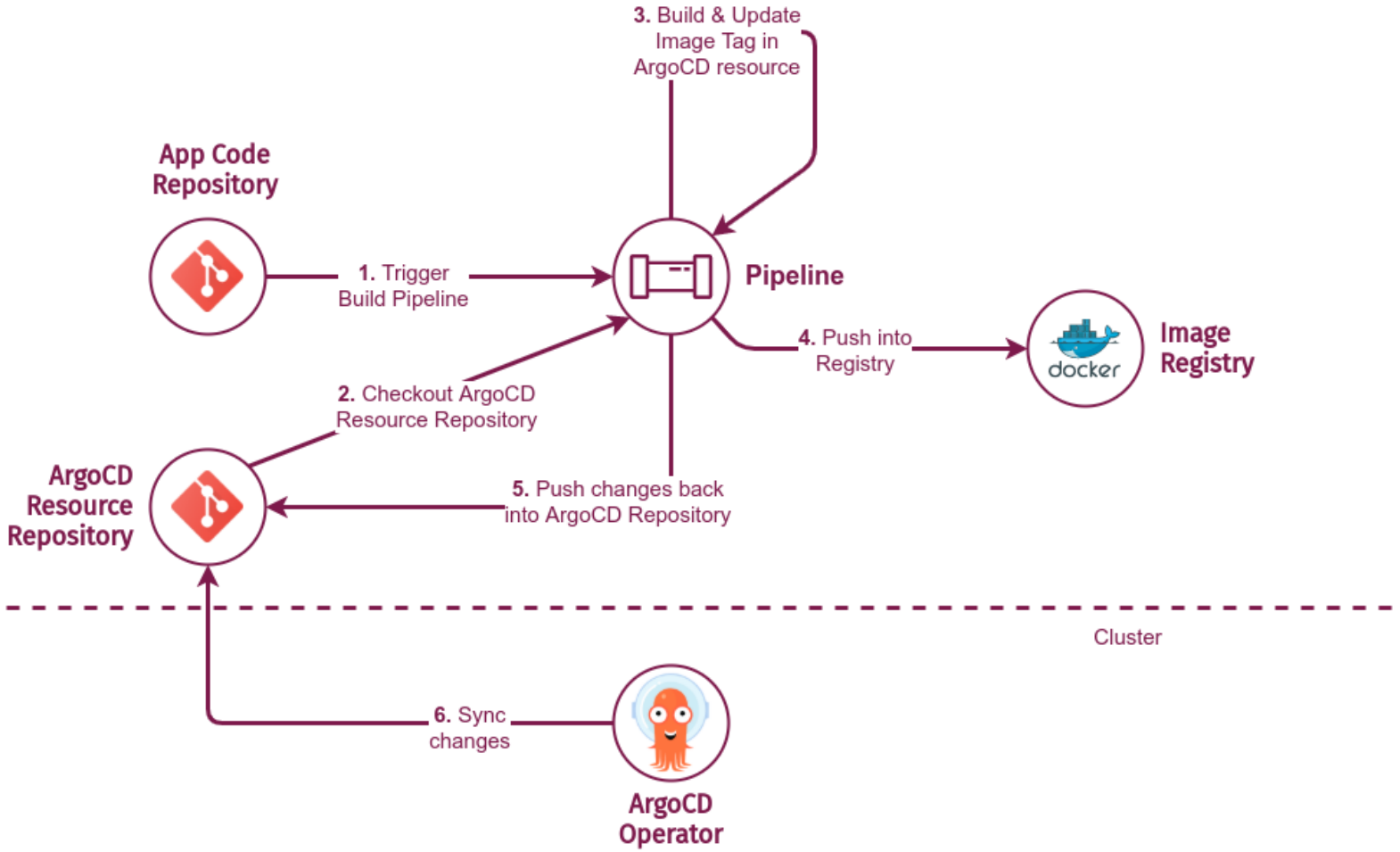
  ¬not imperative

  ¬you have to do on your own

CI integration – Bad practice!



CI integration – Best practice!



Argo CD

Secret Management

Argo CD & Secret Management

- Argo CD has no awareness of k8s **`Secrets`**, and treats them like all other resources
- Argo CD itself does not provide any support for Secret management and delegates this Task to other tools

Secret Management approaches with Argo CD

Two different approaches for managing secrets when using Argo CD with GitOps principles:

- Secrets are pushed to **Git**, but are **encrypted**. A third party tool is used to decrypt the secrets.
- Secrets are **stored in a third party tool** and are **referenced** in the template/manifest. The references are typically resolved by an additional tool before/during the sync process.

Tools for Secret Management

Mature tools to be used for managing secrets

¬Bitnami Sealed Secrets

¬Hashicorp Vault

¬External Secrets

¬Helm Secrets

Secret Management by example with SealedSecrets

How SealedSecret works

- Secrets are sealed by CLI tool **`kubeseal`**. The output is a manifest of kind **`SealedSecrets`** which contains the encrypted secret.
- Secrets are write-only for developers. Once they are sealed only the controller can unseal them.
- Git repository contains only manifests of **`SealedSecrets`**
- Argo CD syncs **`SealedSecrets`** like the other manifests to the cluster
- The controller decrypts the **`SealedSecret`** and converts them to a k8s **`Secret`**

Unsealing SealedSecrets by controller

Before unsealing

```
apiVersion: bitnami.com/v1alpha1
kind: SealedSecret
metadata:
  name: mysecret
  namespace: mynamespace
spec:
  encryptedData:
    foo: AgBy3i4OJSWK+PiTySYZZA9…
```

After unsealing

```
apiVersion: v1
kind: Secret
metadata:
  name: mysecret
  namespace: mynamespace

data:
  foo: bar
```

Argo CD

Labs

Labs

Getting started

Training goals and info

- Do the labs in your own pace
- Use your own workstation
- We are here if you're stuck

Communication

- Google Meets

URLs

- Labs: argocd-training.training.acend.ch
- Rancher cluster: rancher.puzzle.ch
- WebShell: webide-<namespace>.labapp.acend.ch
- App Domain: *.labapp.acend.ch


THX
experience knowledge