acend

GitOps

Ch Schlatter
20. September 2022

experience knowledge

Christian Schlatter

CI/CD Engineer at Puzzle ITC, Bern

Trainer at Acend, Bern

schlatter@puzzle.ch

www.puzzle.ch // www.acend.ch

GitOps

Agenda

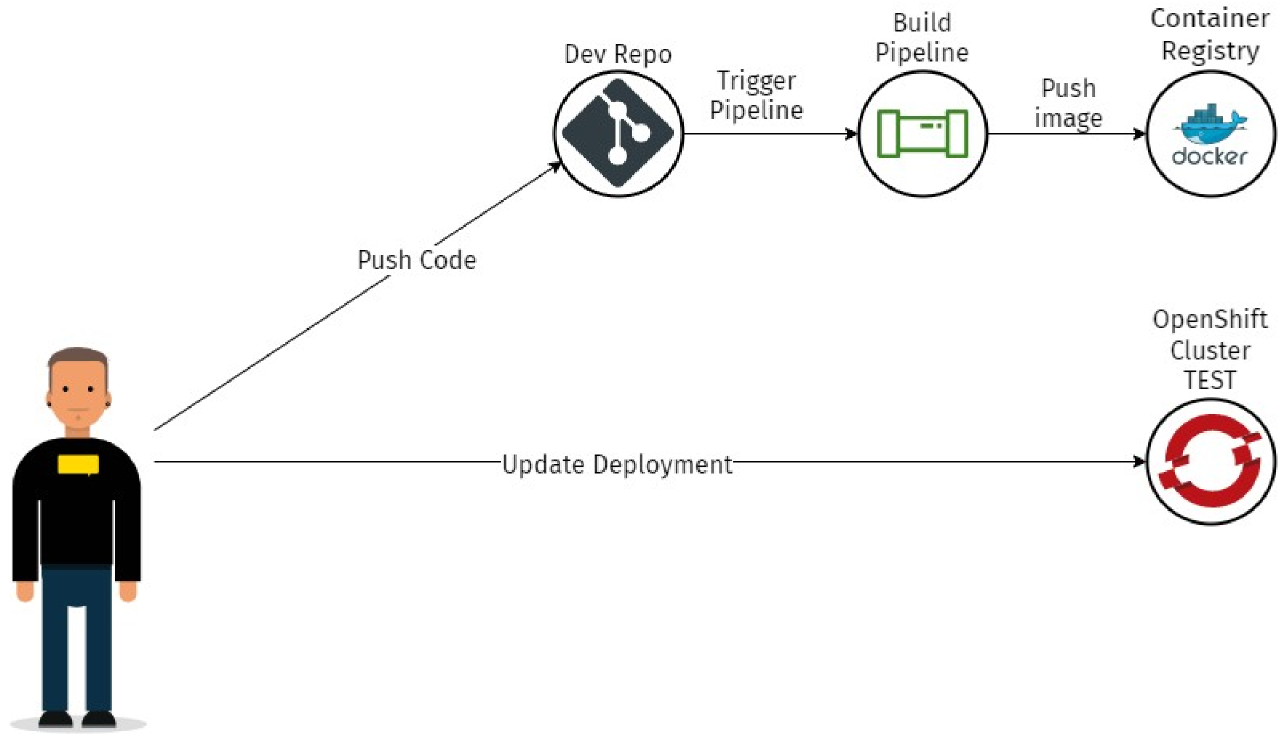
GitOps

Continuous X

Continuous Integration	The automation process for Developers. Source Code changes are continuous built, testet and merged to a shared Repository
Continuous Delivery	Continuous delivery means changes to an application are automatically bug tested and uploaded to a repository (like GitHub or a container registry), where they can then be deployed to a cluster
Continuous Deployment	Continuous deployment refer to automatically releasing a developer’s changes from the repository to production

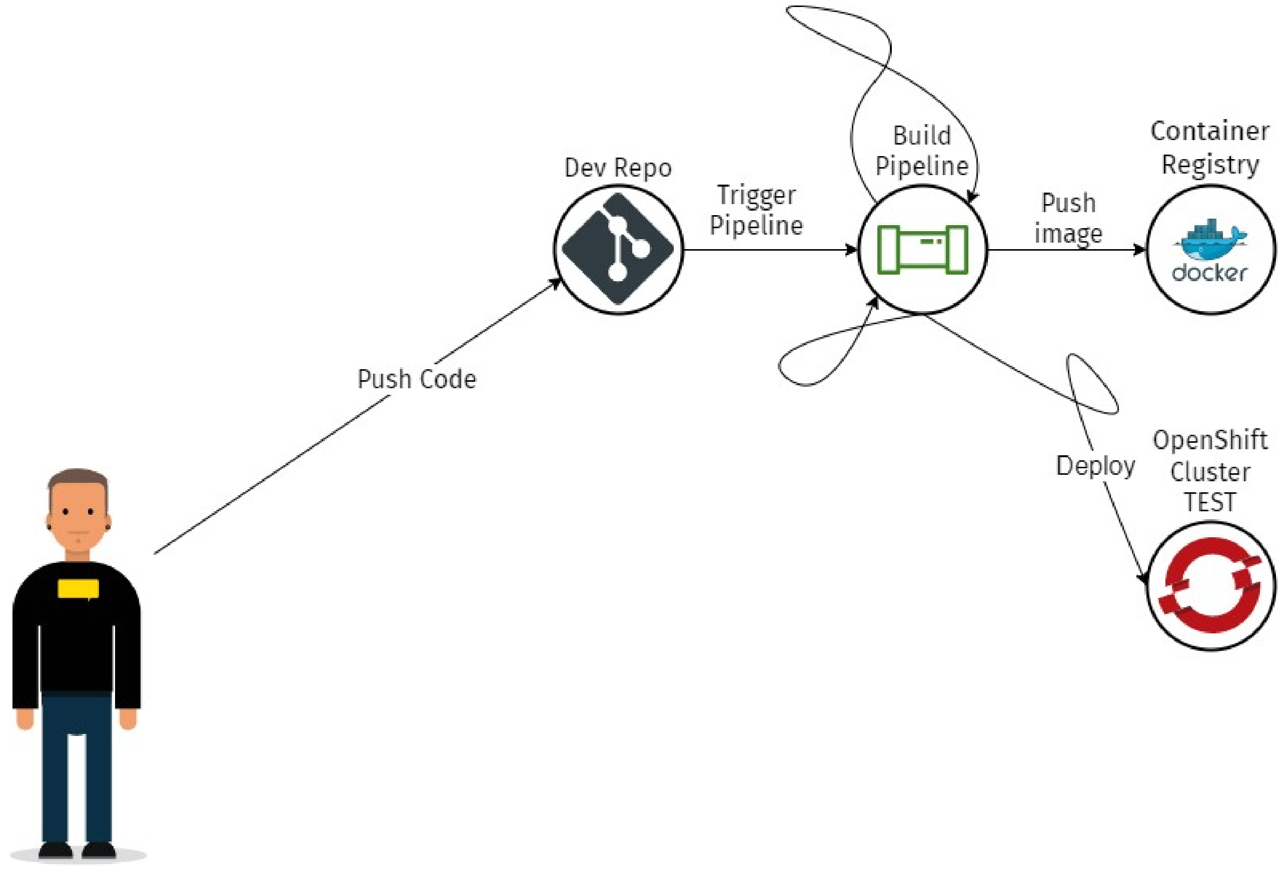
Introduction to GitOps

Continuous Delivery



Introduction to GitOps

Continuous Delivery Pipeline



GitOps

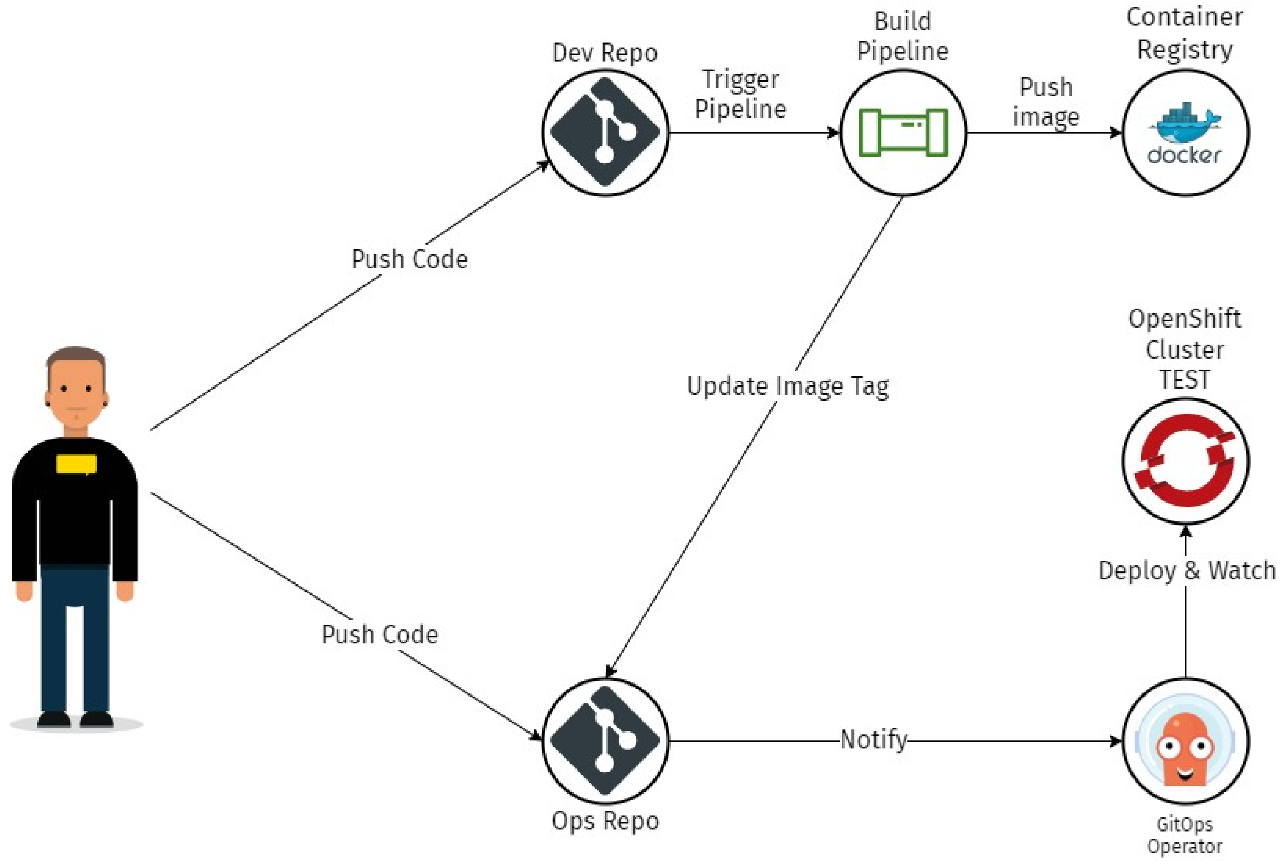
GitOps

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

Introduction to GitOps

Continuous Deployment with GitOps



GitOps

Introduction to Argo CD

Introduction to Argo CD

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

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.

Introduction to Argo CD

Argo CD tools

Kubernetes manifests can be specified 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

Introduction to Argo CD

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
  - **NOT the same as OpenShift Projects!**

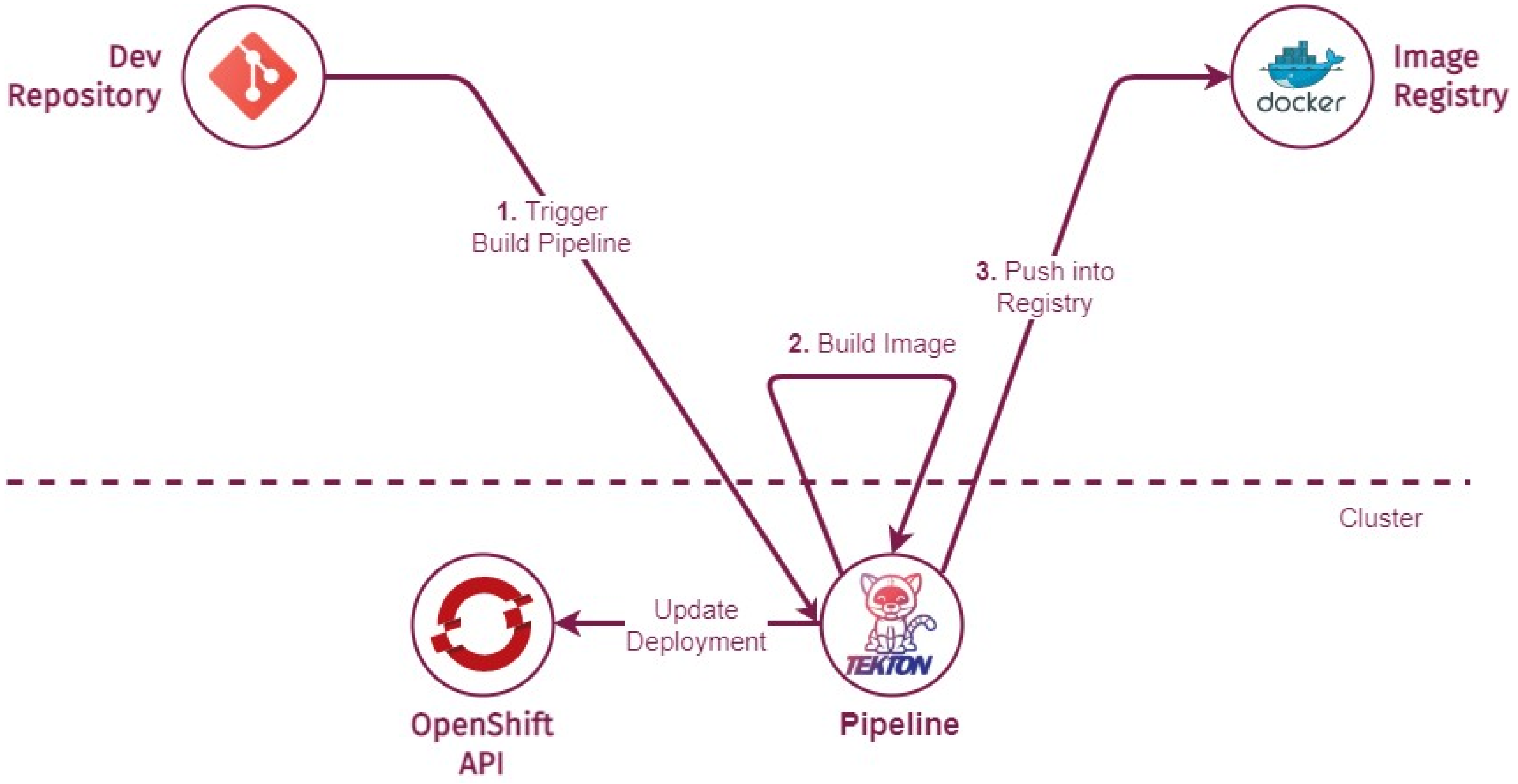
Introduction to Argo CD

CI integration

Can Argo CD be integrated with your CI pipeline?

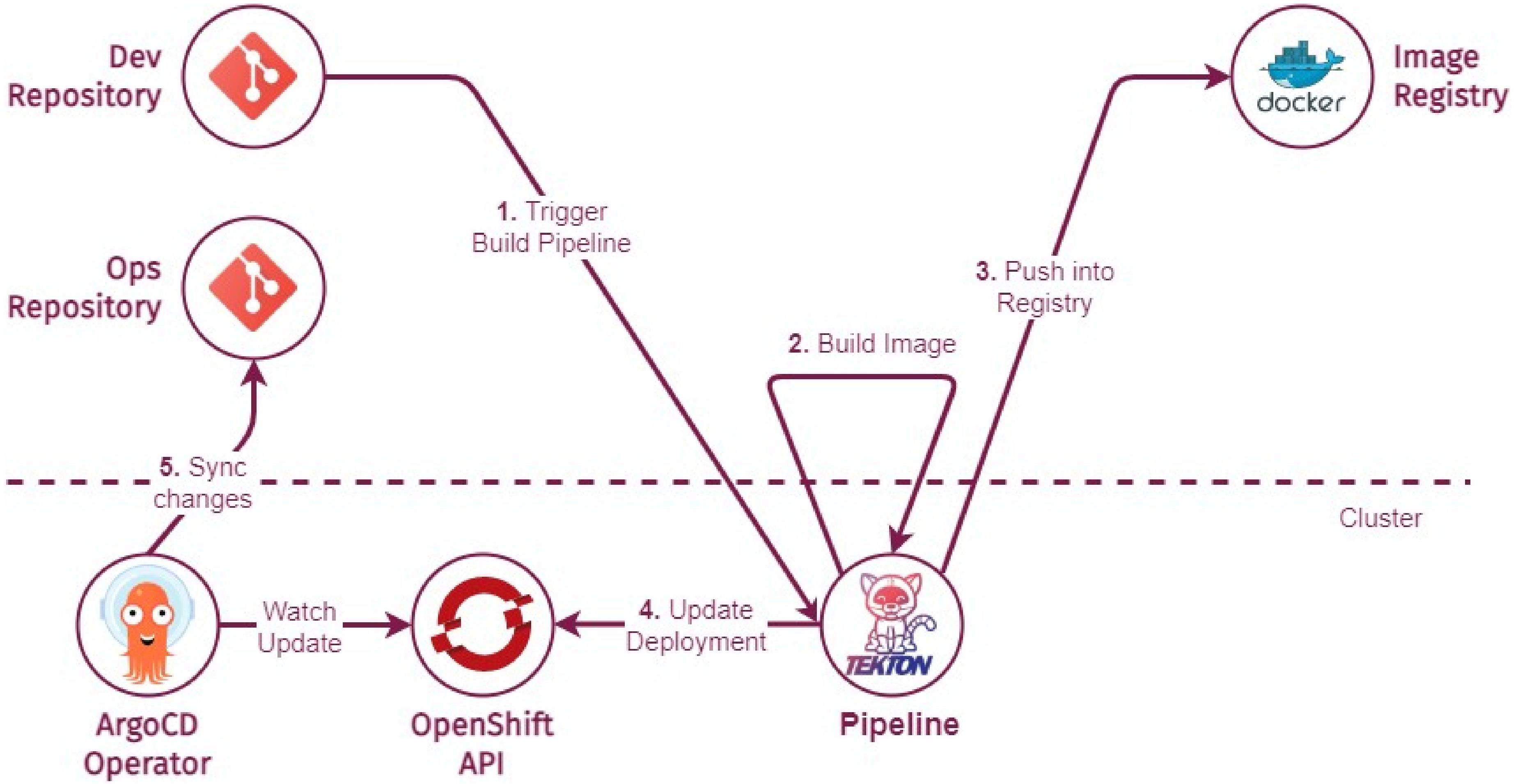
- Yes,
  - via Git
  - and webhooks
- … but …
  - its core principle is declarative,
  - not imperative
  - you have to do on your own

CI integration

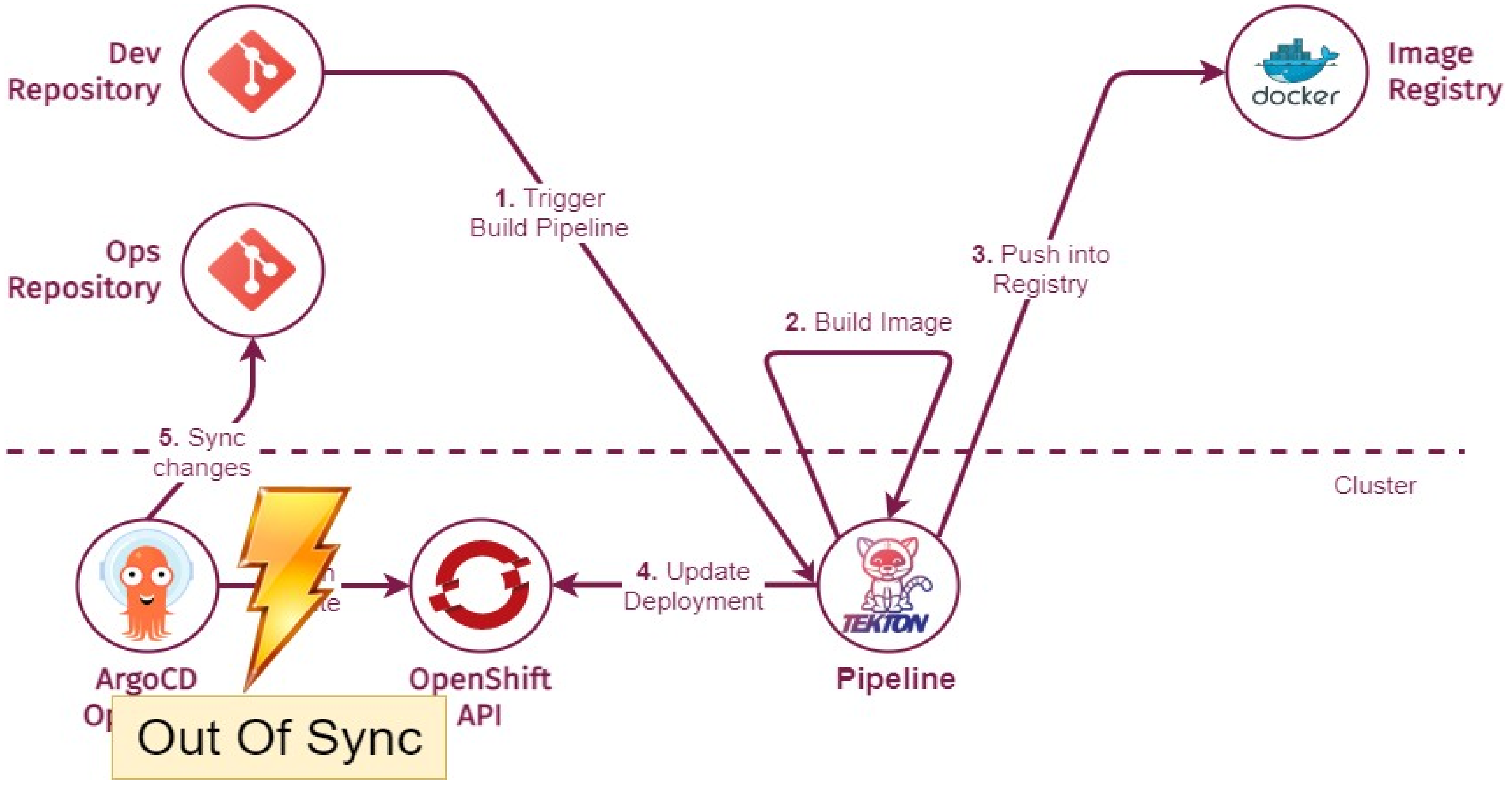


Introduction to Argo CD

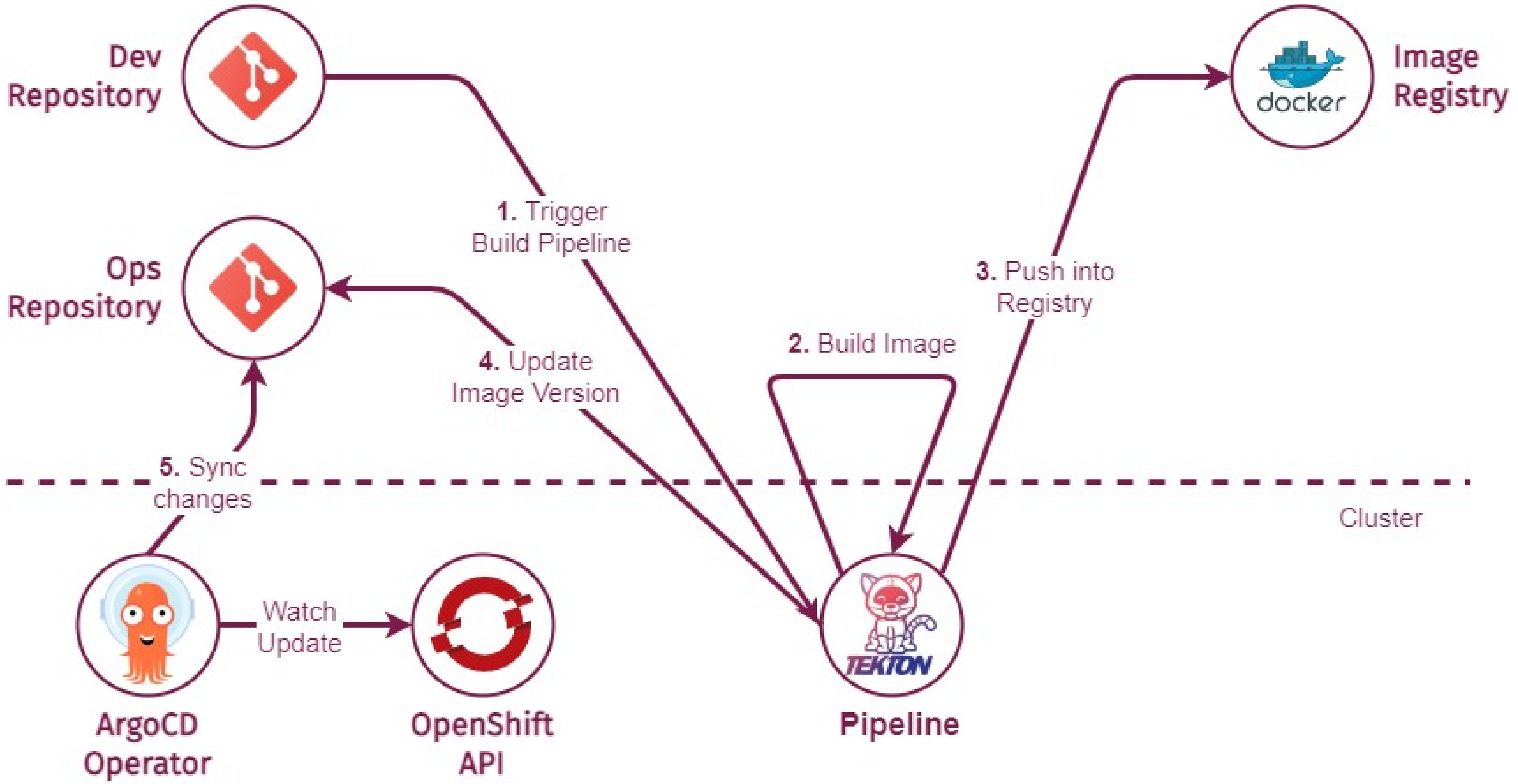
CI integration – Bad practice!



CI integration – Bad practice!



CI integration – Best practice!

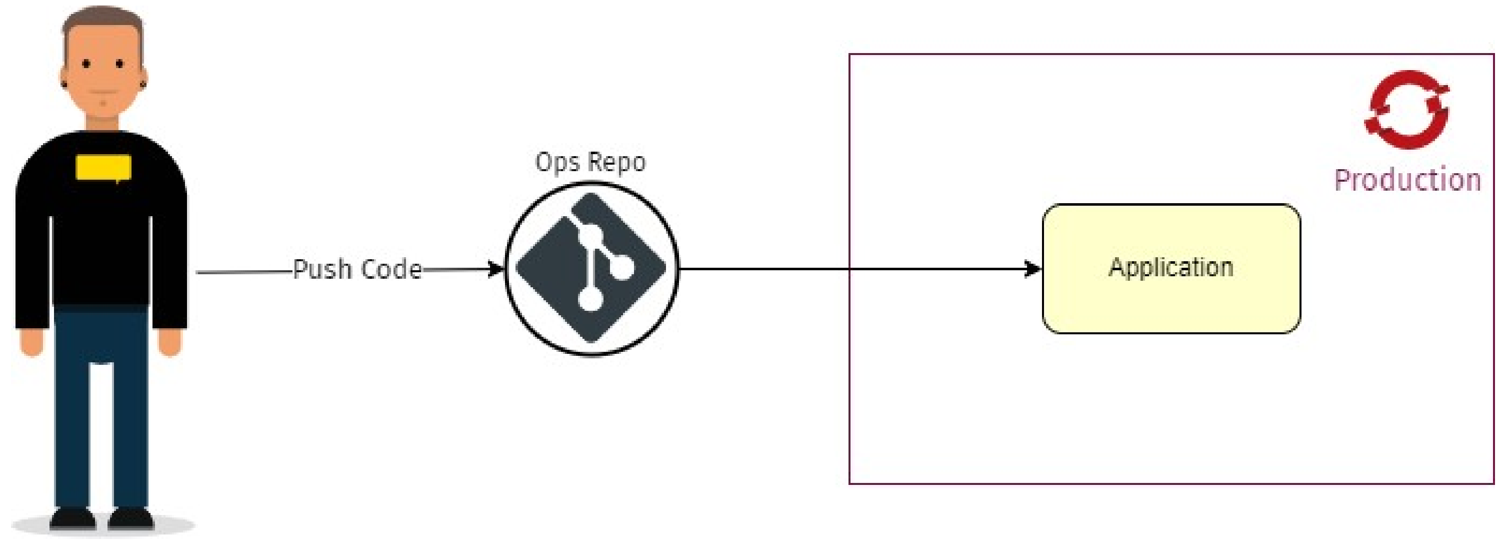


Argo CD

Simple Deployment Example

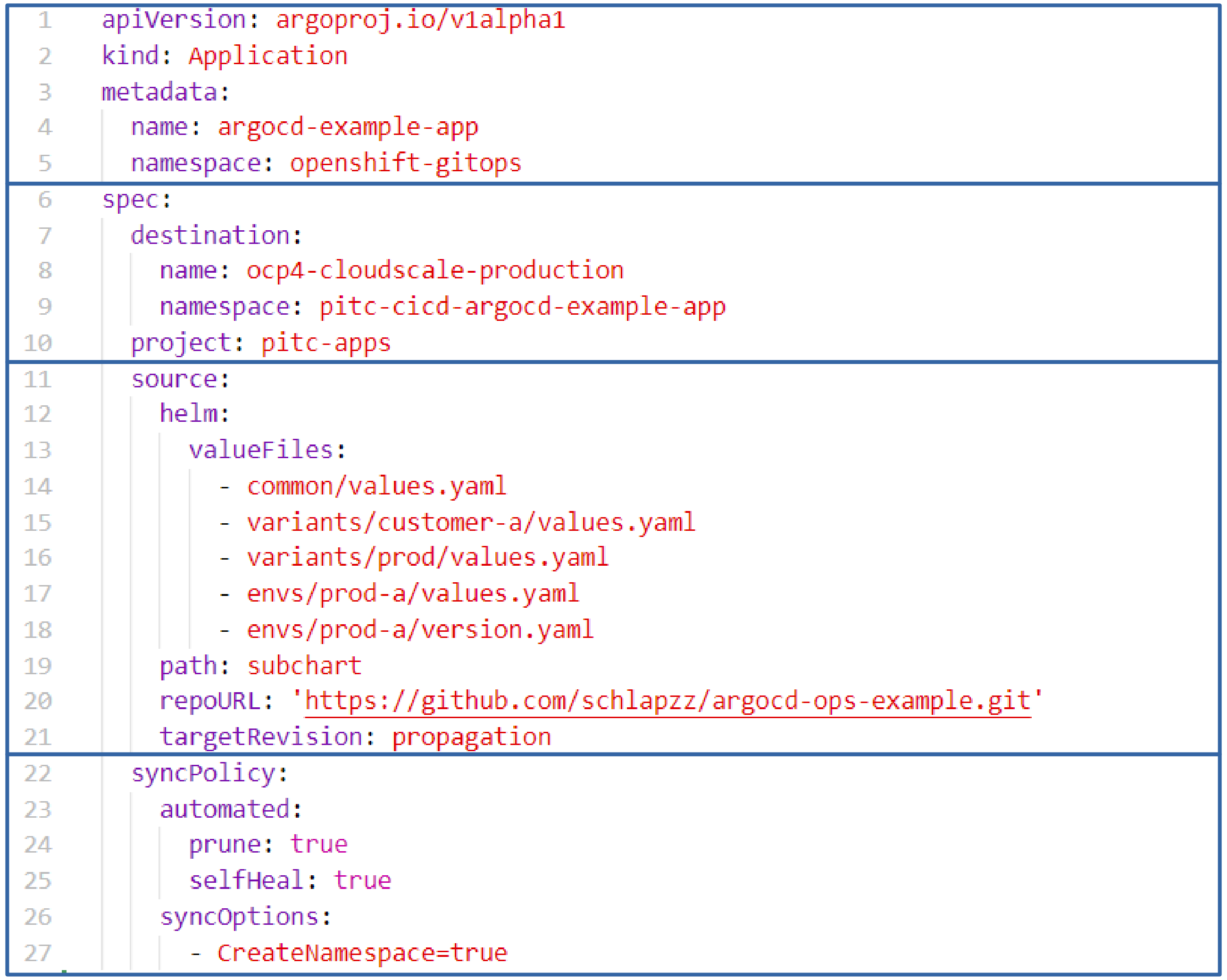
Example 1

Simple Deployment with Argo CD



Example 1

Simple Deployment with Argo CD



```yaml
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: argocd-example-app
  namespace: openshift-gitops
spec:
  destination:
    name: ocp4-cloudscale-production
    namespace: pitc-cicd-argocd-example-app
  project: pitc-apps
  source:
    helm:
      valueFiles:
        - common/values.yaml
        - variants/customer-a/values.yaml
        - variants/prod/values.yaml
        - envs/prod-a/values.yaml
        - envs/prod-a/version.yaml
    path: subchart
    repoURL: 'https://github.com/schlapzz/argocd-ops-example.git'
    targetRevision: propagation
  syncPolicy:
    automated:
      prune: true
      selfHeal: true
    syncOptions:
      - CreateNamespace=true
```

1 WHAT

2 WHERE

3 WHENCE

4 HOW

GitOps with Argo CD

Argo CD Patterns

GitOps

Argo CD Patterns

Dev-Ops Repos	How to structure your Code and Config
App Of Apps	Deploy multiple Apps with one App
ApplicationSet	Deploy one App in multiple Environments
Umbrella Chart	Deploy multiple Services with one Helm Chart
Secret Management	How to handle Secrets
RBAC	How to implement proper RBAC
Config Management	How to structure you Environment Configs

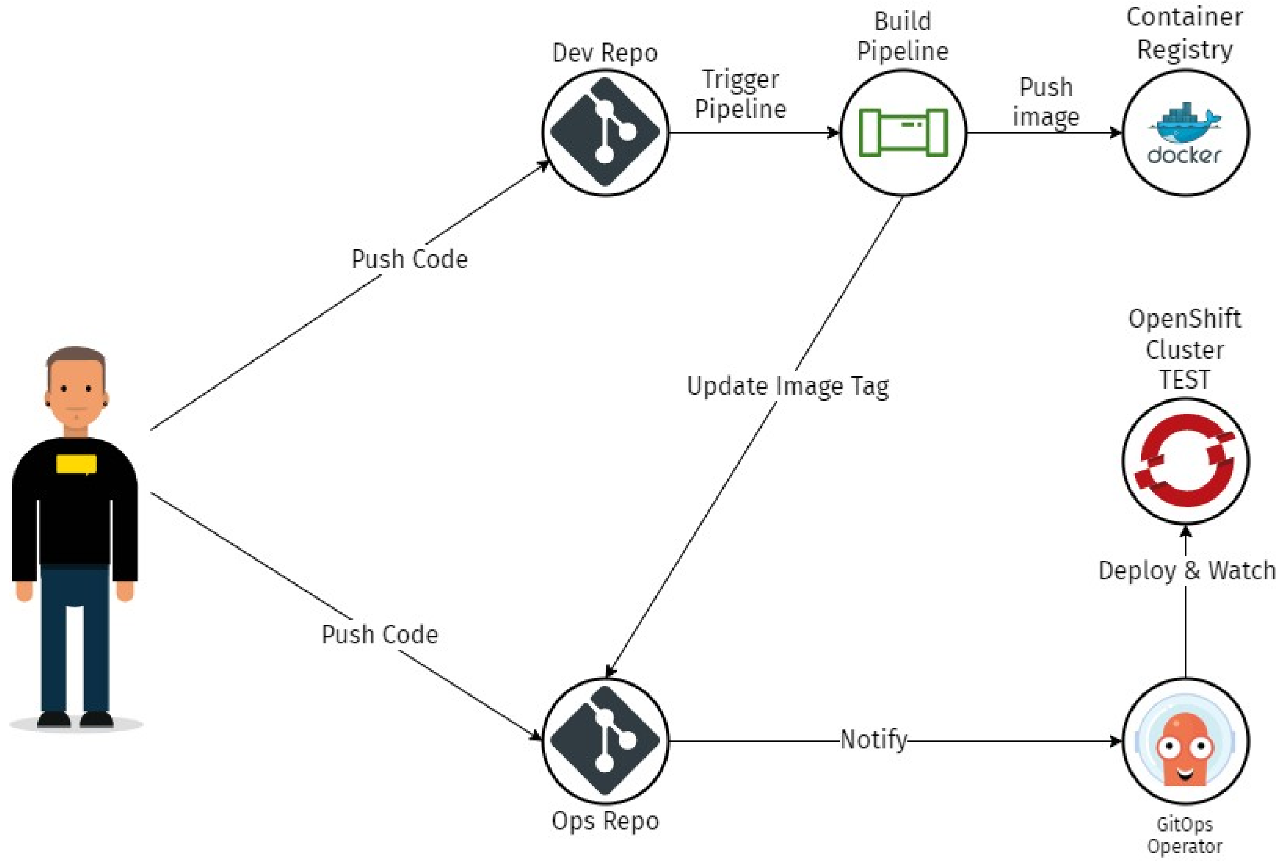
ArgoCD Patterns – Dev-Ops Repositories

Separating Config Vs. Source Code Repositories

- Highly recommended: Clean Separation of Code (Dev-Repo) and Configuration (Ops-Repo)
- 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

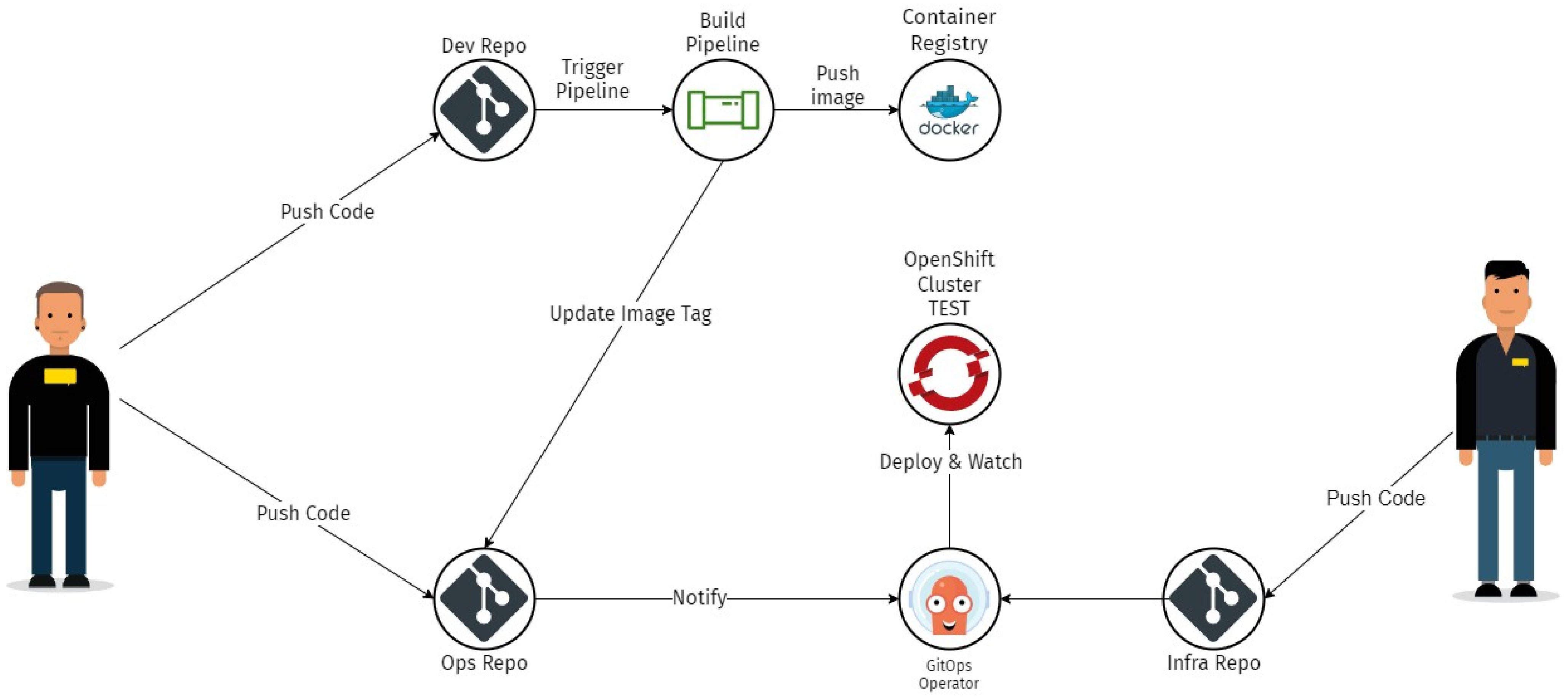
ArgoCD Patterns – Dev-Ops Repositories

Separating Repositories



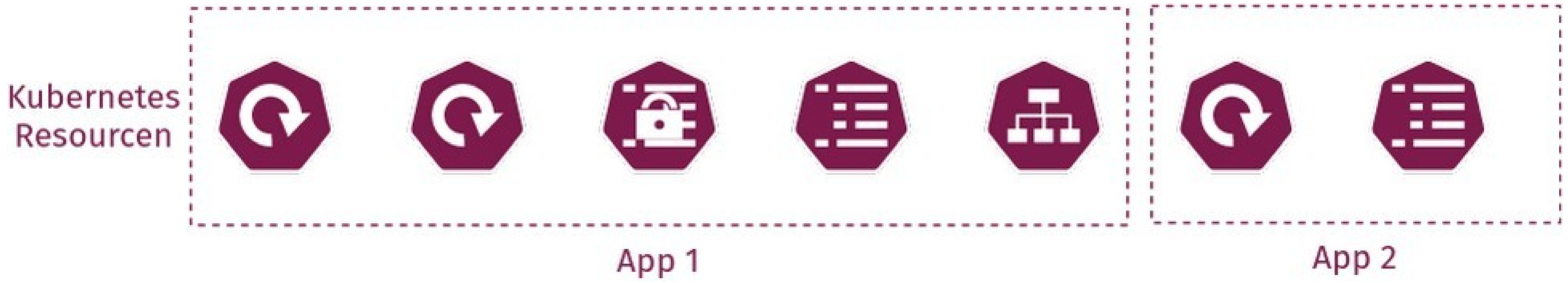
ArgoCD Patterns – Dev-Ops-Infra Repositories

Separating Repositories



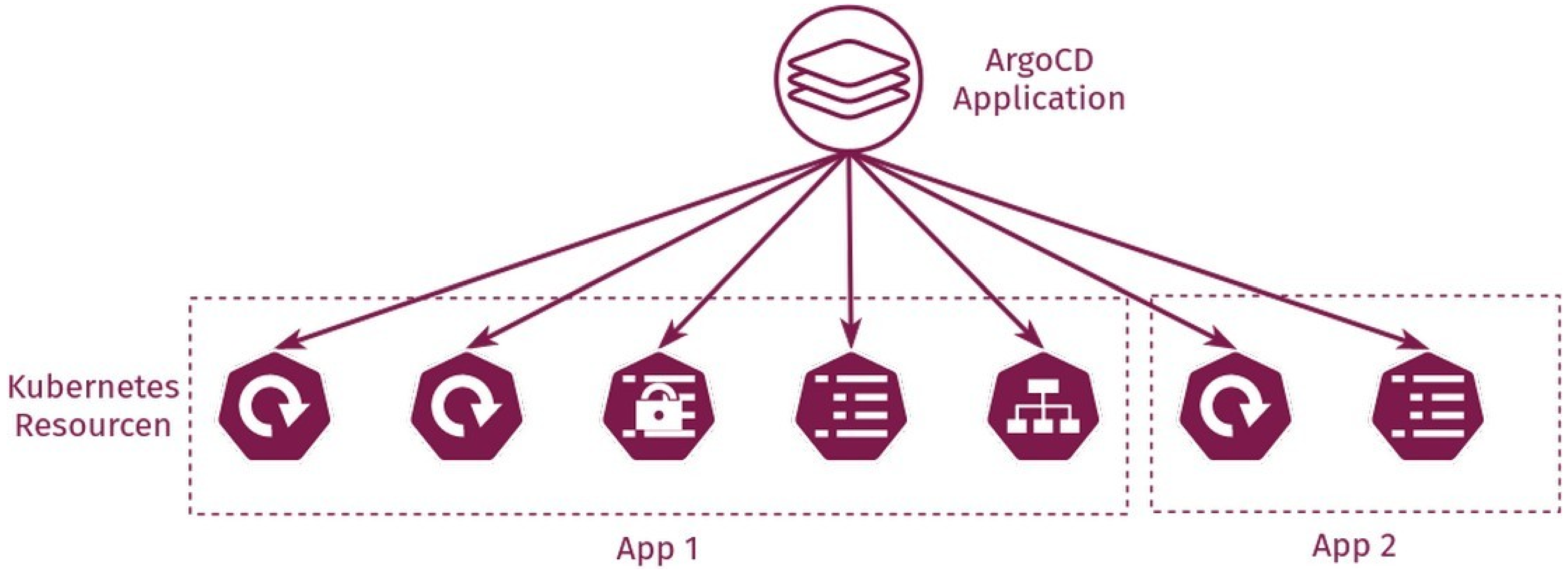
Argo CD Patterns

App Of Apps



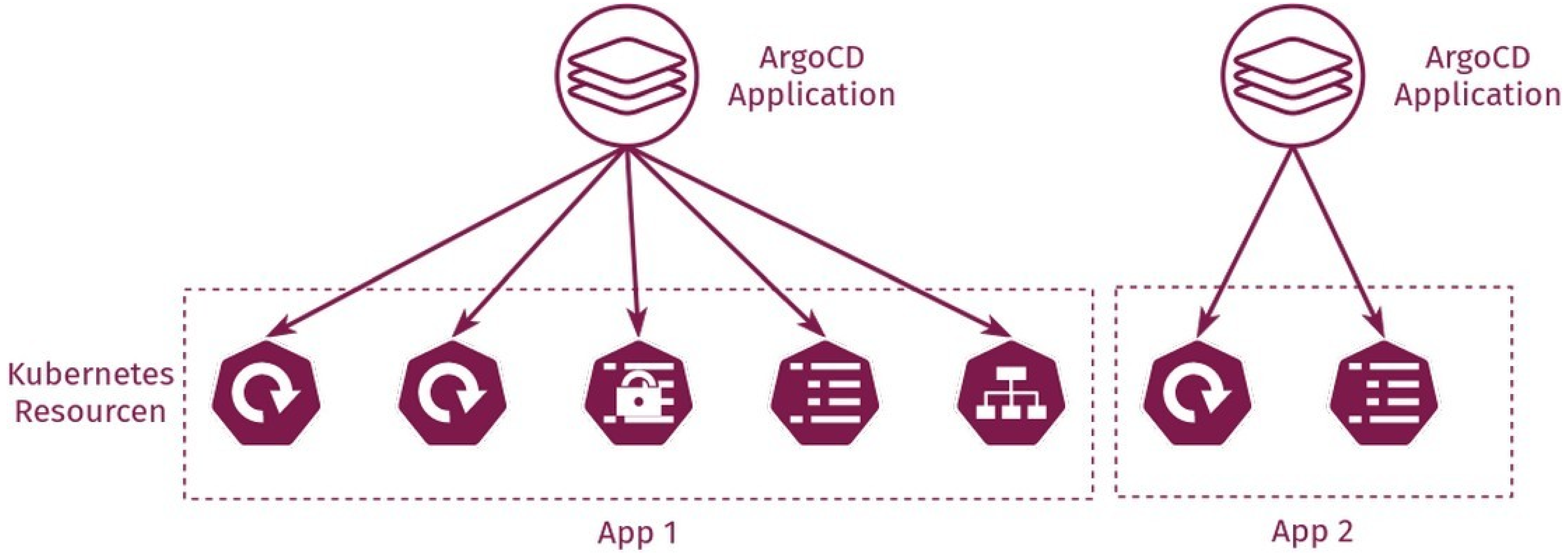
App Of Apps

Manage multiple Apps – BAD!



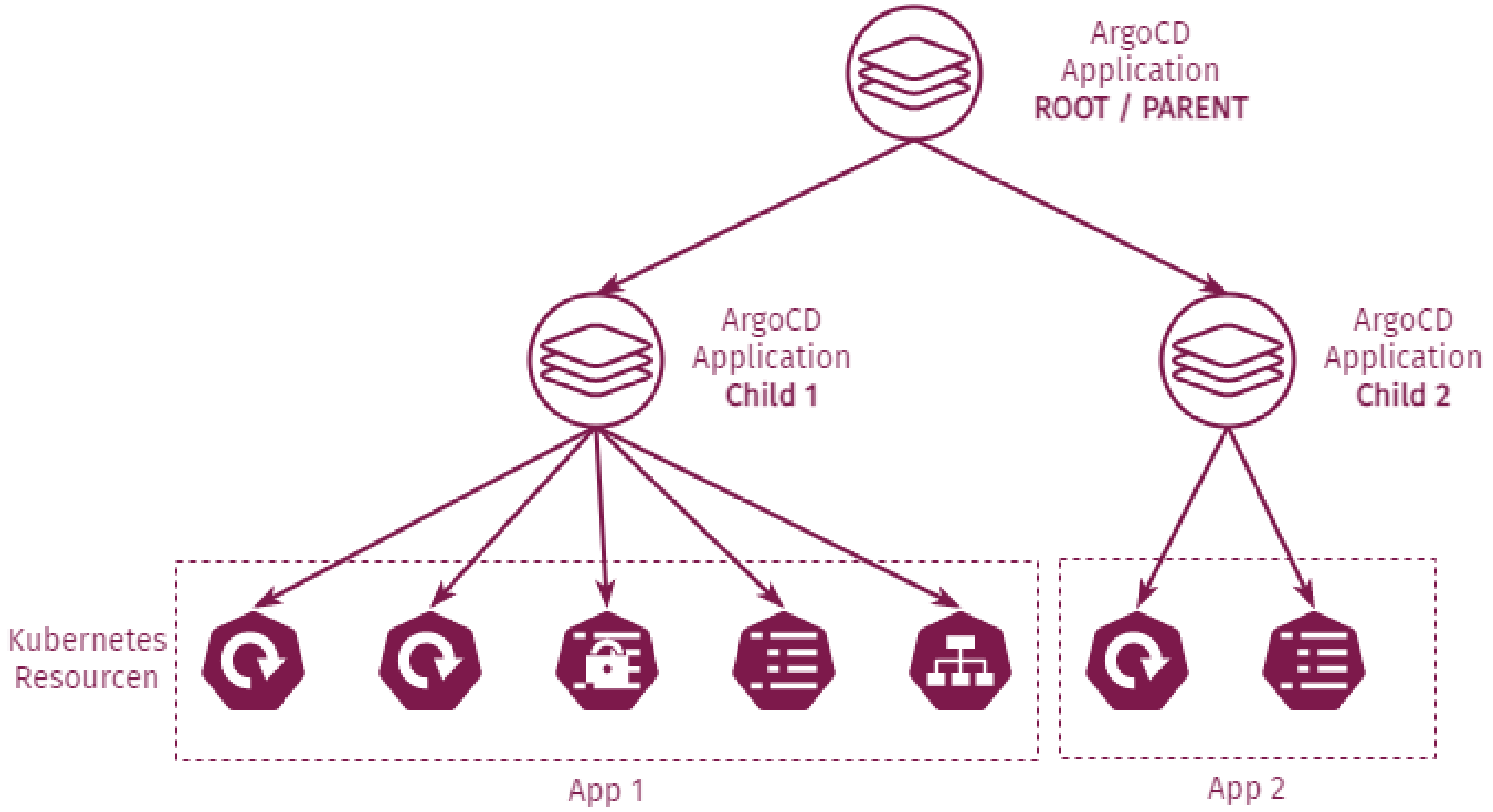
App Of Apps

Manage multiple Apps – GOOD!



App Of Apps

Manage multiple Apps – BEST!


ArgoCD Application ROOT / PARENT
ArgoCD Application Child 1
ArgoCD Application Child 2
Kubernetes Resourcen
App 1
App 2

App Of Apps

Manage multiple Apps

- The App of Apps pattern allows you to deploy multiple apps within one app definition
- One parent App creates multiple child Apps
- Suitable for:
  - Cluster Bootstraping
  - Manage multiple Apps as a group

Argo CD Application

```yaml
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: user1-app-of-apps-1
  namespace: argocd
  finalizers:
  - resources-finalizer.argocd.argoproj.io
spec:
  destination:
    namespace: user1
    name: in-cluster
  project: default
  source:
    path: app-of-apps/apps
    repoURL: https://github.com/acend/argocd-training-examples.git
    targetRevision: HEAD
```

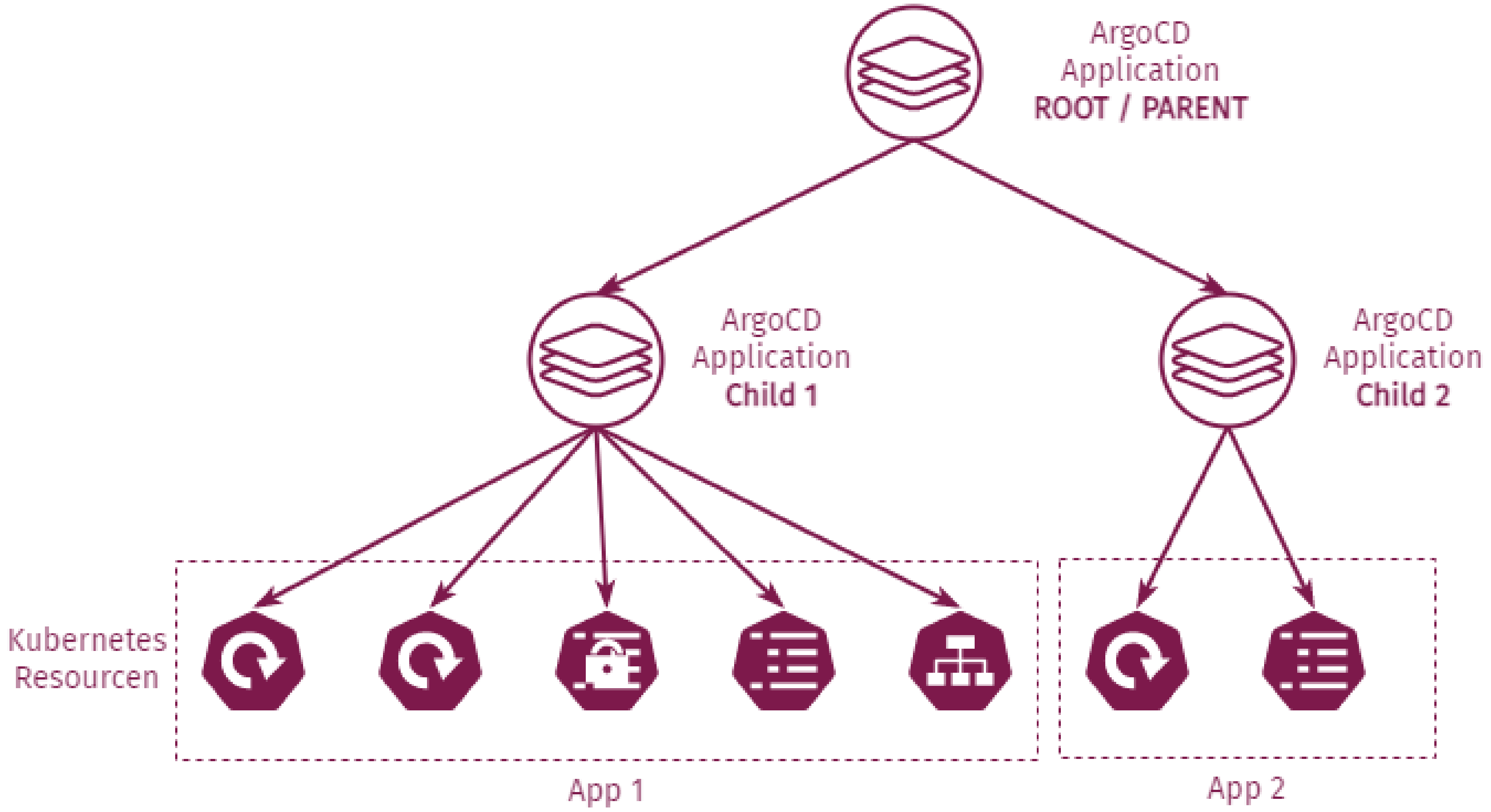
https://github.com/acend/argocd-training-examples

File structure

```
├── app-of-apps
│   ├── app1
│   │   └── deployment.yaml
│   ├── app2
│   │   └── deployment.yaml
│   ├── app3
│   │   └── deployment.yaml
│   └── apps                    #ArgoCD Application Definitions
│       ├── app1.yaml
│       ├── app2.yaml
│       └── app3.yaml
│
```

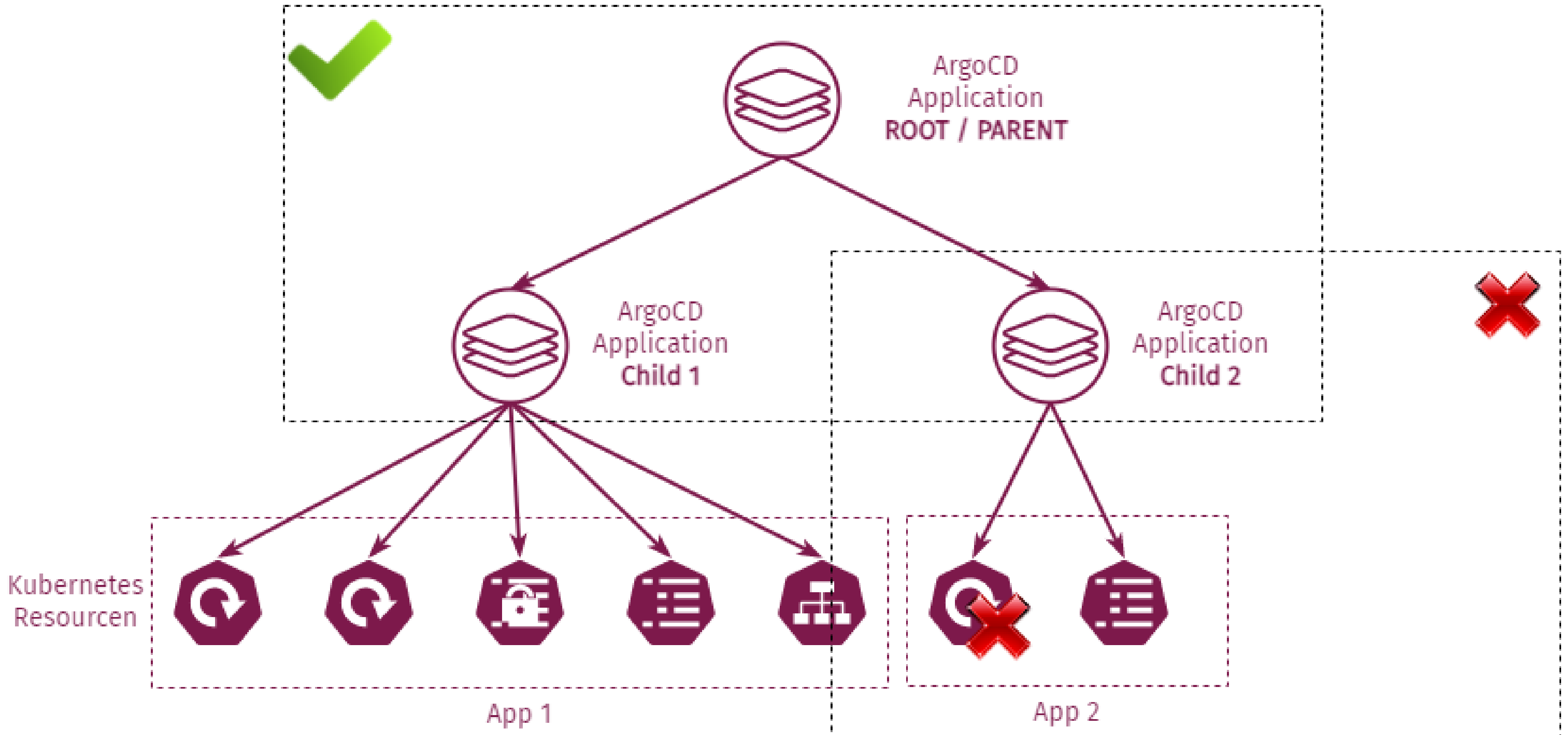
App Of Apps

Manage multiple Apps



App Of Apps

Manage multiple Apps – Scope!



ApplicationSet

ApplicationSets are an Argo CD Resource, suitable for multi deployments. It allows you to define Application templates and render them

- The ability to use a single Kubernetes manifest to target **multiple Kubernetes clusters**
- The ability to use a single Kubernetes manifest to deploy **multiple applications** from one or multiple Git repositories

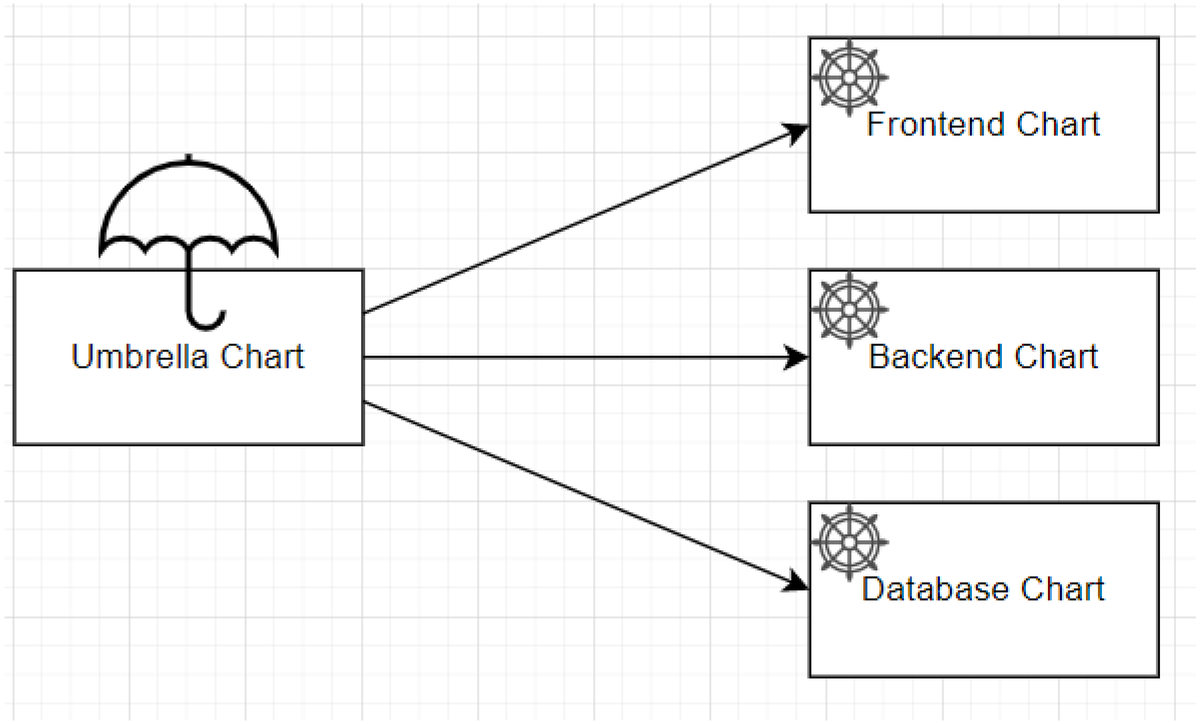
Umbrella Chart

- The Umbrella Chart pattern allows you to deploy multiple services within a single Helm Chart
- One parent Chart creates multiple child Services
- Suitable for:
  - Manage multiple Apps as a group
  - No Argo CD resources are involved

Argo CD Patterns

Umbrella Chart Pattern

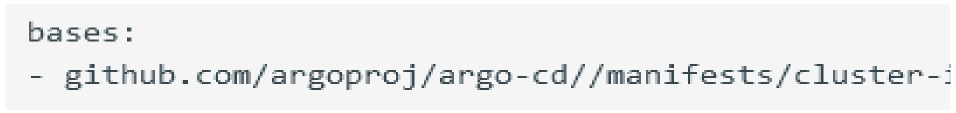
```
# Chart.yaml
dependencies:
- name: nginx
  version: "1.2.3"
  repository: "https://example.com/charts"
- name: memcached
  version: "3.2.1"
  repository: "https://another.example.com/charts"
```



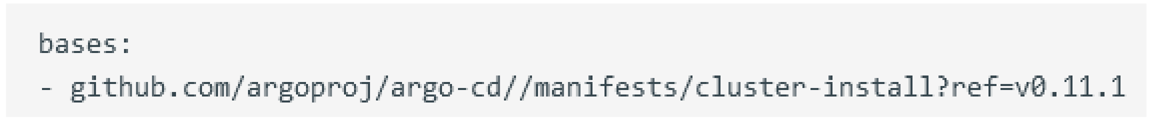
Argo CD Patterns

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-
```



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

Upstream might change

Version is fixed

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.

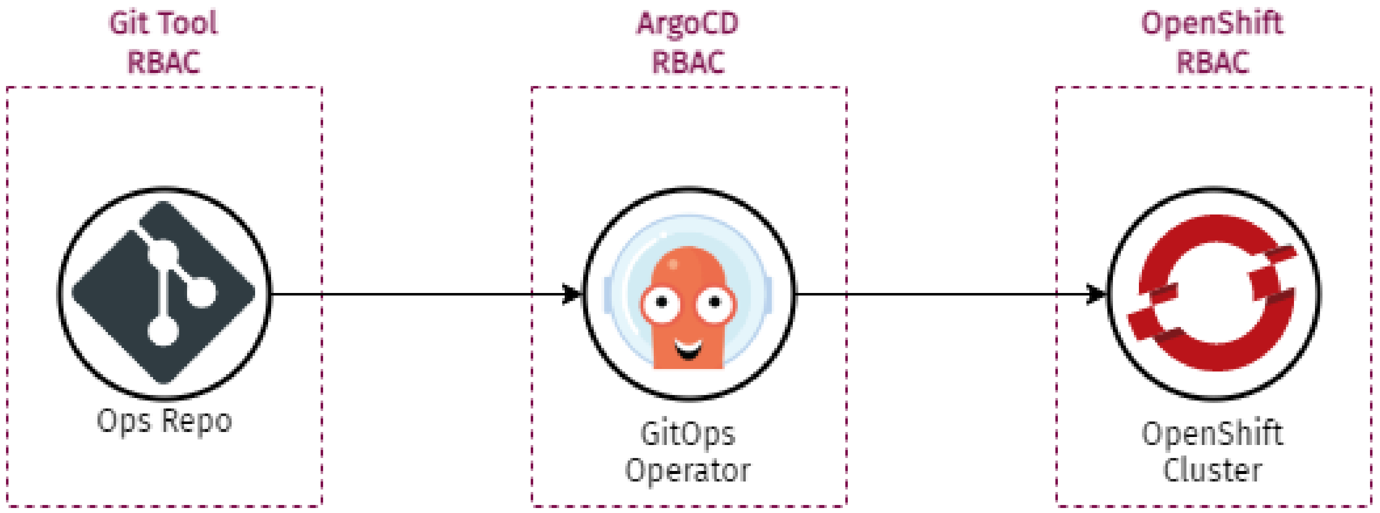
Tools for Secret Management

Mature tools to be used for managing secrets

- Bitnami Sealed Secrets
- Hashicorp Vault
- External Secrets
- Helm Secrets

Argo CD Patterns

RBAC



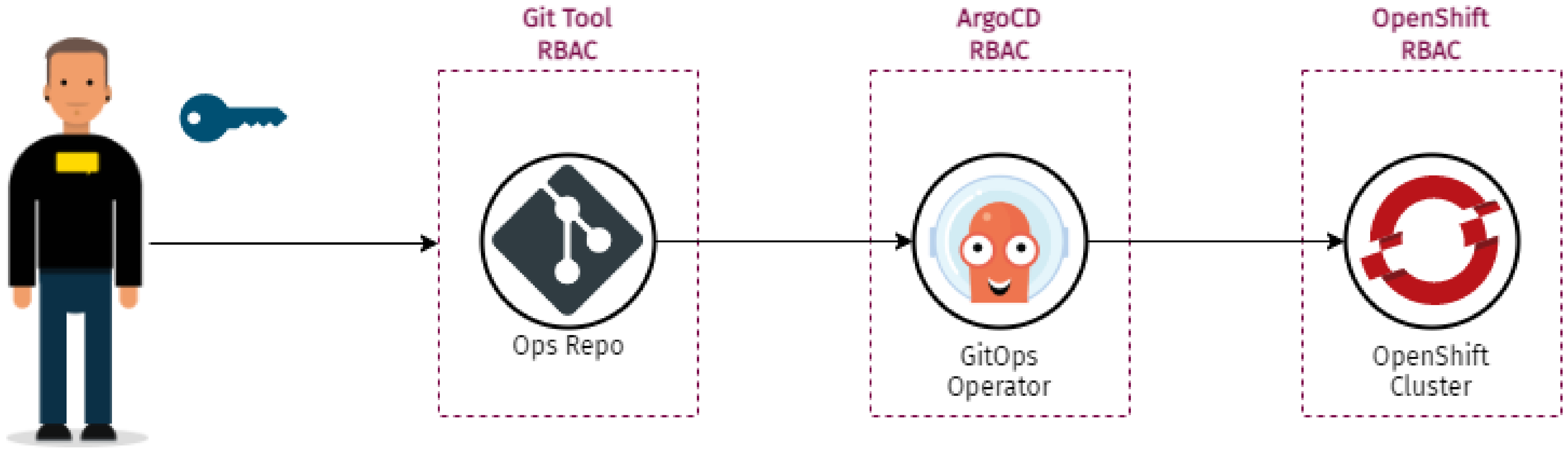
- Protected Projects
- Protected Branches
- User Access
- Pull / Merge Requests

- Projects
- Allowed repos
- Allowed Actions

- OpenShift RBAC

Argo CD Patterns

RBAC



- Sign your commits

- Protected Projects
- Protected Branches
- User Access
- Pull / Merge Requests

- Projects
- Allowed repos
- Allowed Actions

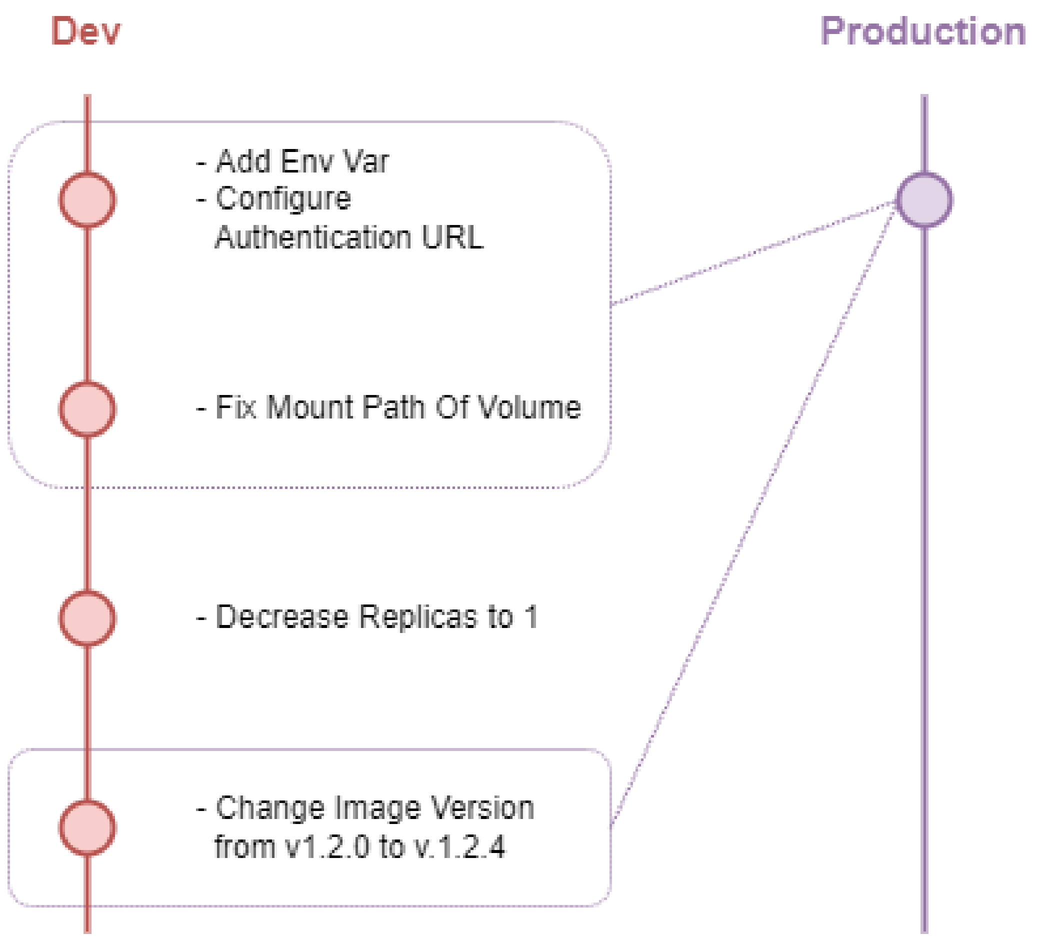
- OpenShift RBAC

Environment Management – Multi Branch

Using Git branches for modeling different environments is an anti-pattern.

- Pull requests and merges between different branches is problematic.
- As soon as you have a large number of environments, maintenance of all environments gets quickly out of hand.
- The branch-per-environment model goes against the existing Kubernetes ecosystem

Multi Branch merge problems


Dev
Production
- Add Env Var
- Configure Authentication URL
- Fix Mount Path Of Volume
- Decrease Replicas to 1
- Change Image Version from v1.2.0 to v.1.2.4

Config Management

Separate your Configuration

- Don’t put all your configs in a single File
- Better handling for Environment/Stage promotion
- Cleaner audit Log
- Easier to diff

Argo CD Patterns

Config Management

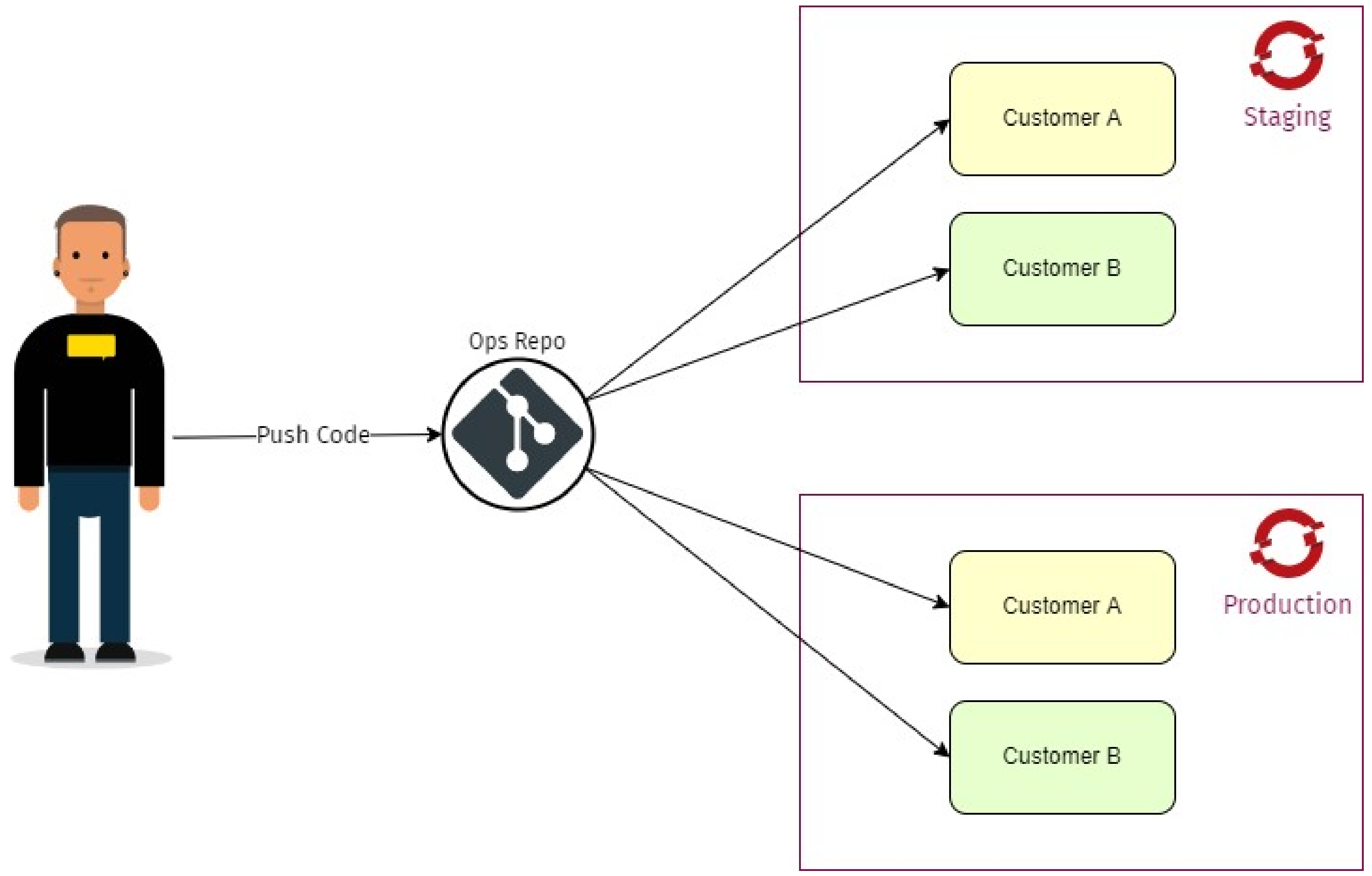
Application Version	**Kubernetes Settings**	**Static Business Settings**	**Non-Static Business Settings**
The application version in the form of the container tag used.	Kubernetes specific settings for your application. This includes the replicas of the application and other Kubernetes related information.	Settings that are unrelated to Kubernetes but have to do with the business of your application. **This are a settings that you never want to promote between environments**	This is the same thing as the previous point, **but it includes settings that you DO want to promote between environments**.

Argo CD

Multi Environment Deployment

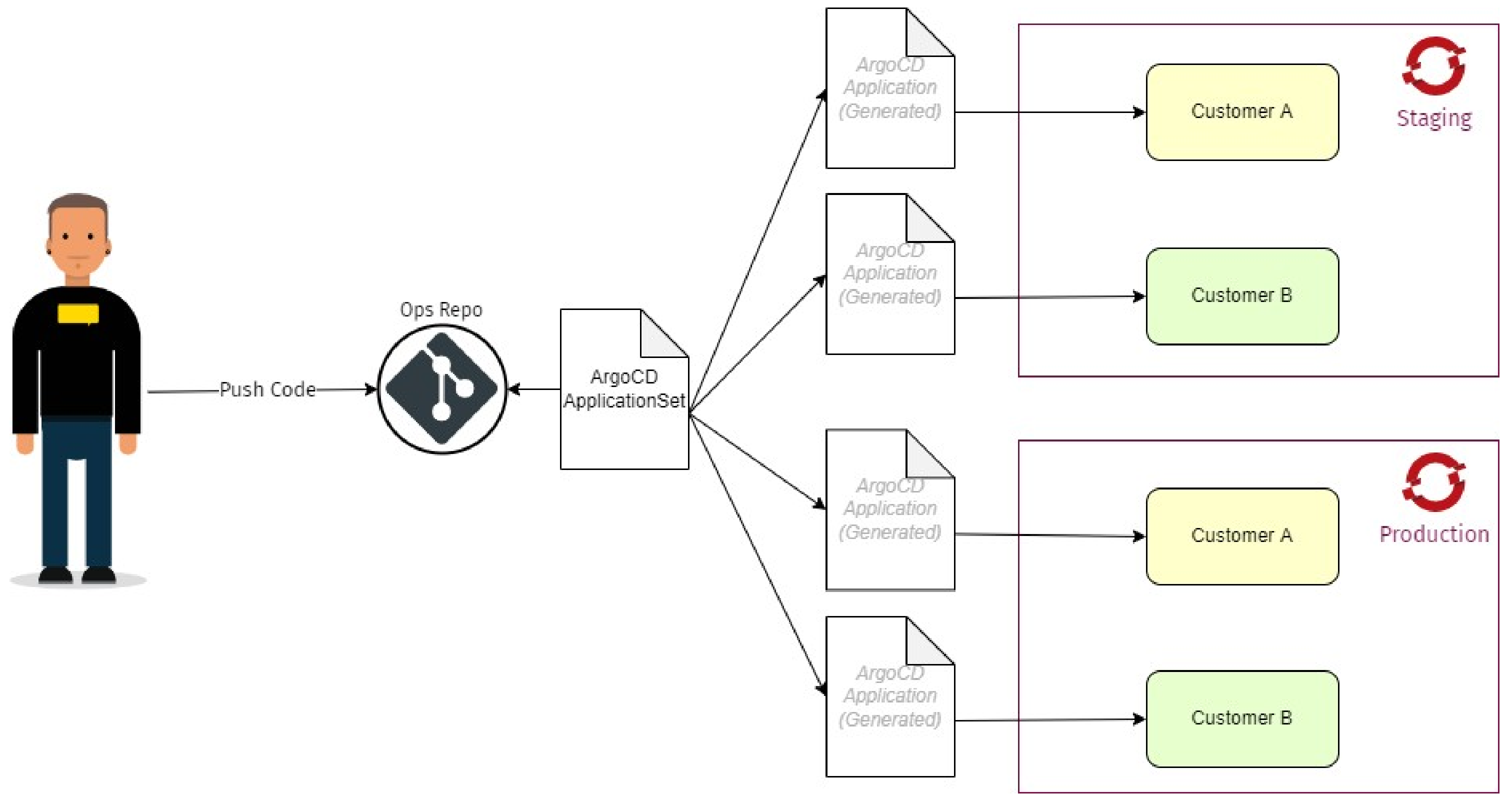
Argo CD - ApplicationSet

Multi Environment Deployment



Argo CD - ApplicationSet

Multi Environment Deployment


Push Code
Ops Repo
ArgoCD ApplicationSet
ArgoCD Application (Generated)
ArgoCD Application (Generated)
ArgoCD Application (Generated)
ArgoCD Application (Generated)
Customer A
Customer B
Staging
Customer A
Customer B
Production

Argo CD - ApplicationSet

Multi Environment Deployment

```yaml
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: exapp
spec:
```

1

```yaml
  generators:
    - list:
        elements:
          - customer: a
            stage: prod
            cluster: ocp4-cloudscale-production
          - customer: b
            stage: prod
            cluster: ocp4-cloudscale-production
```

2

```yaml
  template:
    metadata:
      name: exapp-{{stage}}-{{customer}}
    spec:
      source:
        repoURL: https://github.com/schlapzz/argocd-ops-example.git
        targetRevision: propagation
        path: subchart
        helm:
          valueFiles:
          - common/values.yaml
          - variants/customer-{{customer}}/values.yaml
```

https://github.com/schlapzz/argocd-ops-example

acend

acend

Acend Argo CD Basic Training

THX]

experience knowledge