welcome.

devoteam

Introduction to GitOps on OpenShift.

How to stop worrying about environment reproductibility

Vincent Brobald – Senior ICT Consultant / 2020-01-24

Innovative technology consulting for business.

1

What is GitOps?

Something old,
something new

GitOps

GitOps is an operating model for **K8S** (Kubernetes) **cloud-native** environments.

This model centers on the use of the **Git** distributed version control system to provide a workflow for **infrastructure as code**, **environment as code** and **configuration as code**.

While this model naturally emerged from the automation and Continuous Delivery during the last two decades, it really found its rooting with the emergence of K8S and and its declarative model.

The term GitOps itself was coined by Weaveworks in 2017 as they wanted to formalize the processes for successful Git-based operations.

2

Principles of GitOps

Something borrowed

Main principles

Declarative description

VIRTUALSERVICE

PERSISTENVOLUMECLAIM

IMAGESTREAM

SERVICEACCOUNT

MACHINESET

ROLEBINDING

DEPLOYMENTCONFIG

Configuration
Environment
Infrastructure

= CODE

RESOURCEQUOTA

NETWORKPOLICY

CRD

TEMPLATE

STORAGECLASS

NAMESPACE

ROUTE

DAEMONSET

BUILDCONFIG

SERVICE

CONFIGMAP

DESTINATIONRULE

Main principles

One versioned source of truth (scoped)

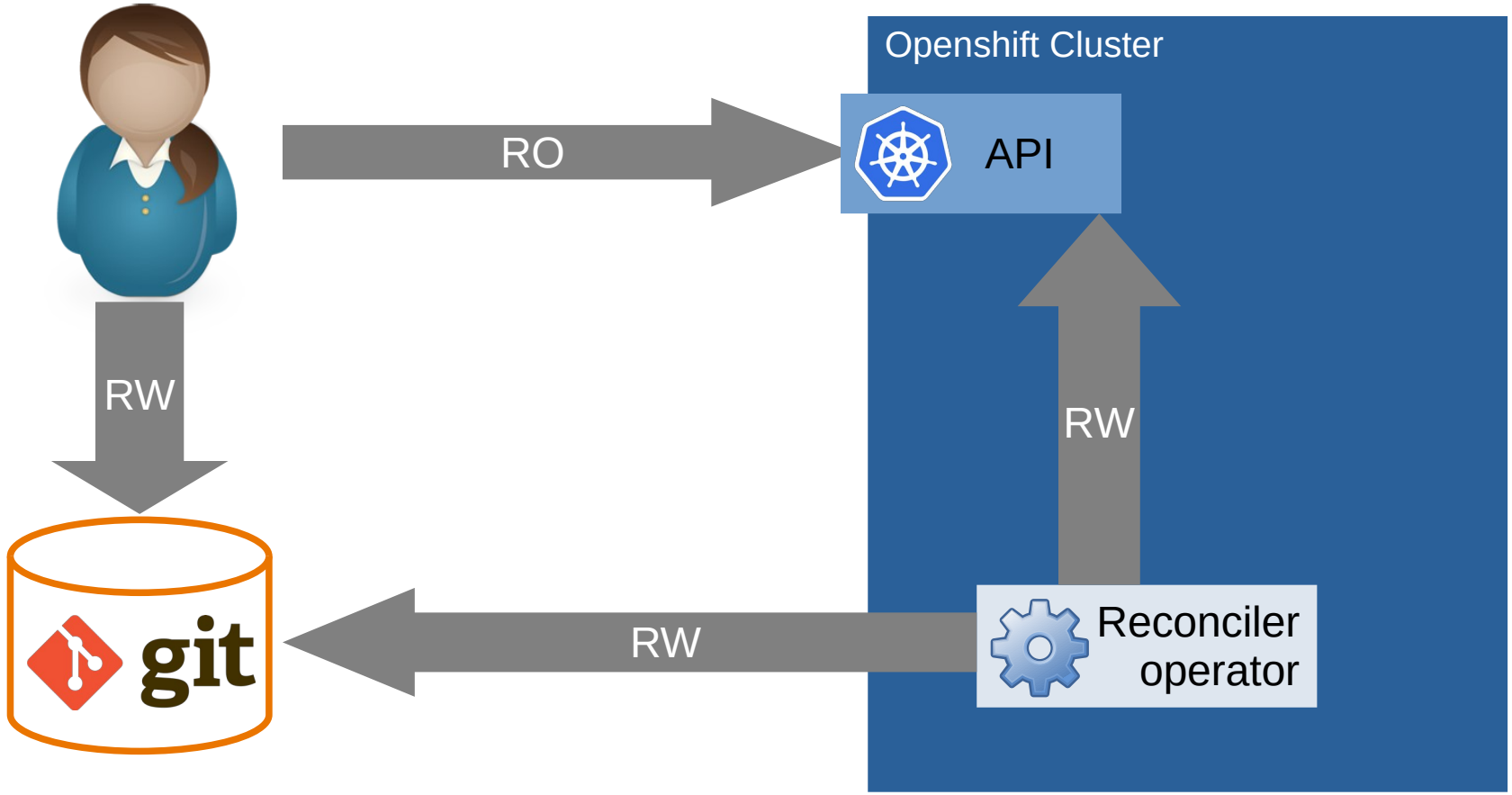
Configuration
Environment
Infrastructure

= CODE

DISTRIBUTED
VERSION
CONTROL

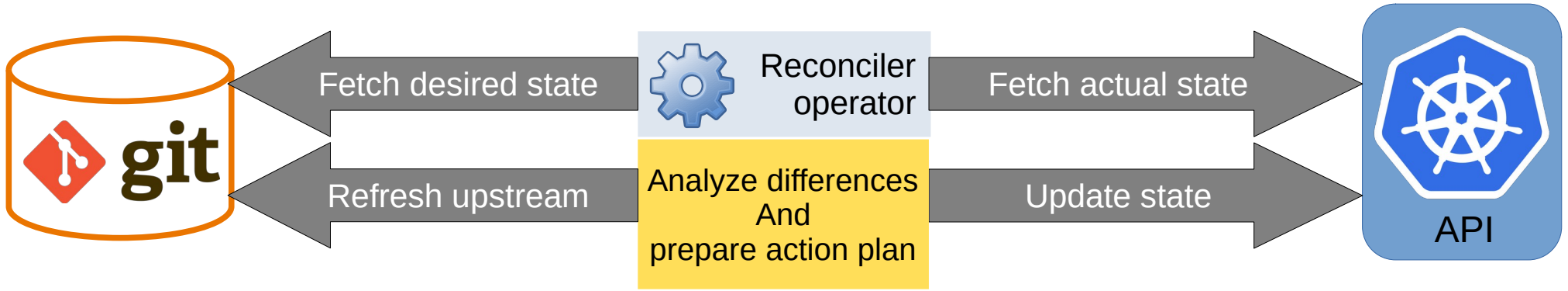
Main principles

All Ops are performed using the Git Workflow



Main principles

Continuous monitoring and reconciliation



3

Why GitOps

Just say no to bloated tools and dashboards

Why GitOps

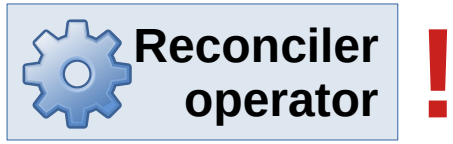
Workflow

You can reuse the best practices used for development workflow.
You benefit from the flexibility of the git collaborative model
Clear audit trail
You can use your git environment to enforce code review before merge.

Why GitOps

Self Healing and recovery - namespace

Namespace/Project



ConfigMap

```
apiVersion: v1
data:
  maven-template: |-
    <org.csanchez.jenkins.plugins.kubernetes.
      <inheritFrom></inheritFrom>
      <name>maven</name>
      <privileged>false</privileged>
      <alwaysPullImage>false</alwaysPullImage>
      <instanceCap>2147483647</instanceCap>
      <idleMinutes>0</idleMinutes>
      <label>maven</label>
      <serviceAccount>jenkins</serviceAccount>
      <nodeSelector></nodeSelector>
      <customWorkspaceVolumeEnabled>false
      <workspaceVolume class="org.csanchez
        <memory>false</memory>
      </workspaceVolume>
      <volumes />
      <containers>
        <org.csanchez.jenkins.plugins.kubernetes.
          <name>jnlp</name>
          <image>openshift/jenkins-agent-maven-
          <privileged>false</privileged>
          <alwaysPullImage>false</alwaysPullIm
```

DeploymentConfig

```
piVersion: apps.openshift.io/v1
kind: DeploymentConfig
metadata:
  annotations:
    openshift.io/generated-by: OpenShiftNewApp
    template.alpha.openshift.io/wait-for-ready: "
  creationTimestamp: 2019-08-22T20:04:26Z
  generation: 10
  labels:
    app: jenkins
    template: jenkins-ephemeral-template
  name: jenkins
  namespace: cicd-vincent
  resourceVersion: "40783466"
  selfLink: /apis/apps.openshift.io/v1/namespacic
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    name: jenkins
  strategy:
    activeDeadlineSeconds: 21600
    recreateParams:
      timeoutSeconds: 600
    resources: {}
    type: Recreate
```

Service

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    openshift.io/generated-by: OpenShiftNewApp
    service.alpha.openshift.io/dependencie
      "", "kind": "Service"}]'
    service.openshift.io/infrastructure: "true"
  creationTimestamp: 2019-08-22T20:04:26Z
  labels:
    app: jenkins-ephemeral
    template: jenkins-ephemeral-template
  name: jenkins
  namespace: cicd-vincent
  resourceVersion: "27052333"
  selfLink: /api/v1/namespaces/cicd-vincent/ser
  uid: 0b0d5b64-c518-11e9-b1c1-525400edd645
spec:
  clusterIP: 172.30.102.3
  ports:
  - name: web
    port: 80
    protocol: TCP
    targetPort: 8080
  selector:
    name: jenkins
  sessionAffinity: None
  type: ClusterIP
```

Why GitOps

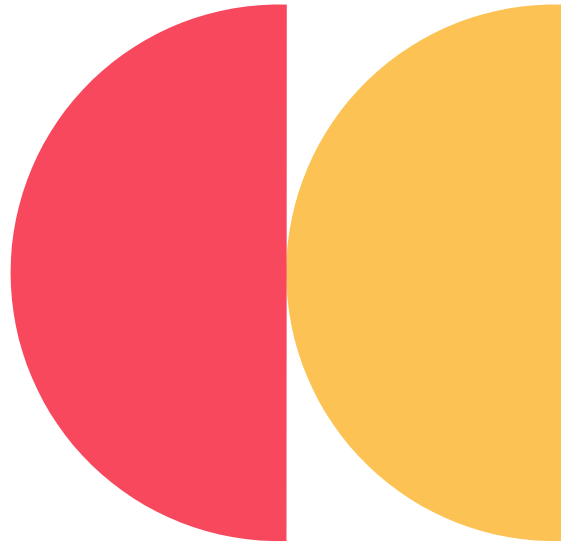
Allows low risk scrap & replace.

The continuous reconciliation has a very positive side-effects:

- Your configuration definition is tested continuously
- Therefore it is always current
- So you can reuse it with confidence to restart with a clean situation.

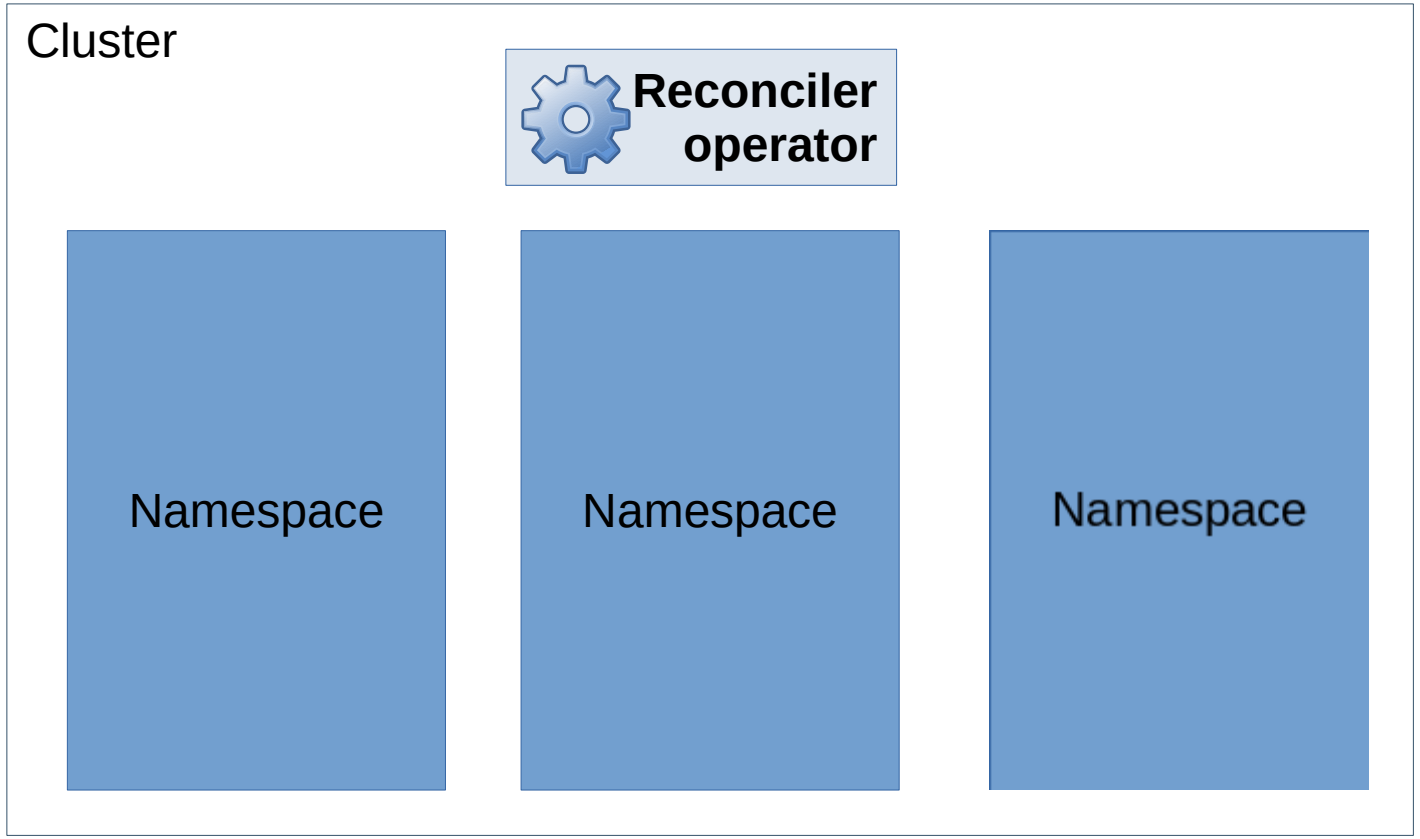
4

Topologies



Operational Models

Single reconcilier – Full cluster



Key Features:

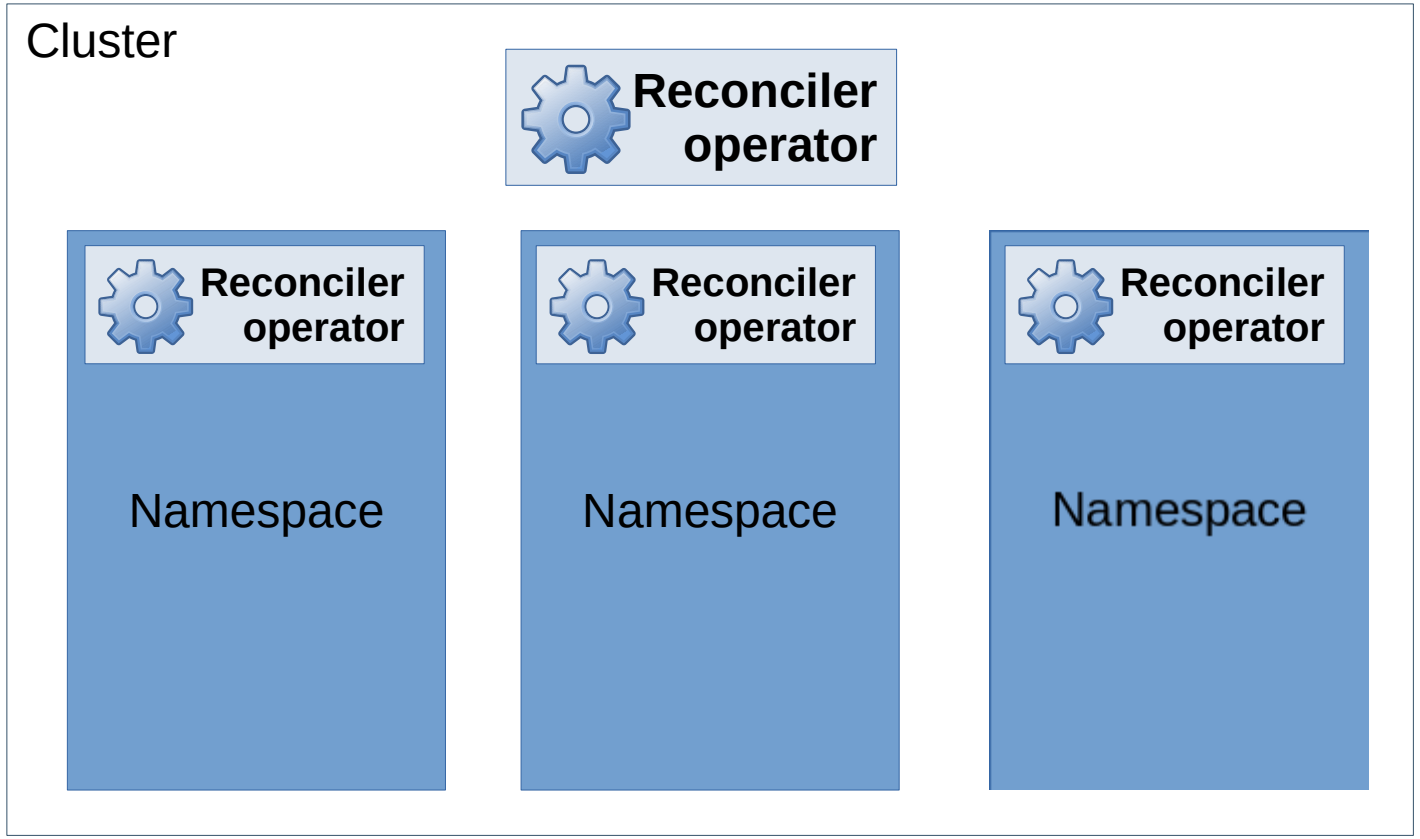
- All namespaces in one Git project
- One reconciler managing all namespaces

Useful for:

- Clusters needing tightly coupled lifecycle
- Lightweight clusters (decentralized/edge)

Operational Models

Cascading reconciler – Full cluster



Key Features:

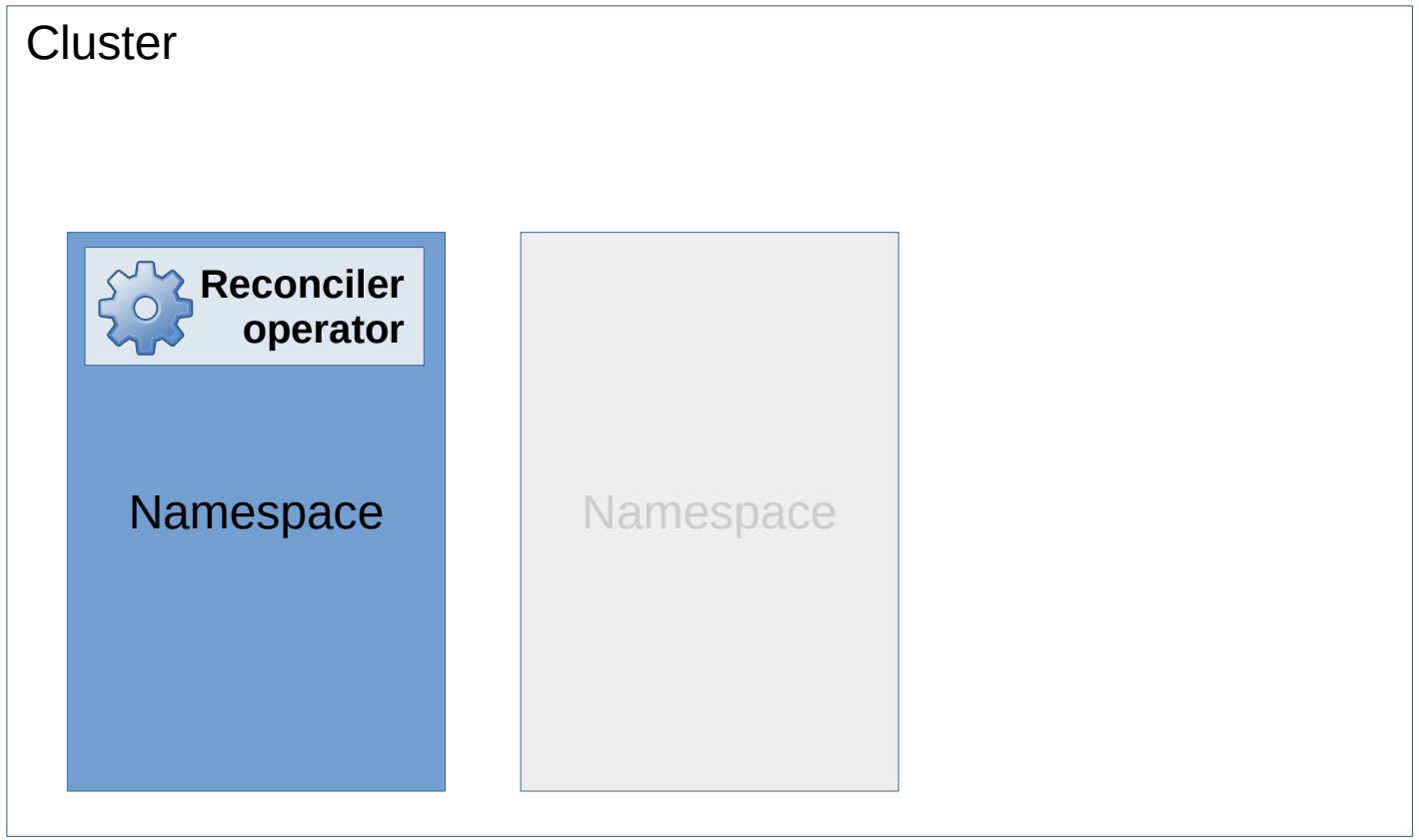
- One Git project and one reconciler for the cluster-level resources and project bootstrapping.
- One Git project and one reconciler per namespace (or namespace group)

Typical use:

Large clusters hosting various different projects with dissociated lifecycle and managing teams.

Operational Models

Single reconciler – single namespace



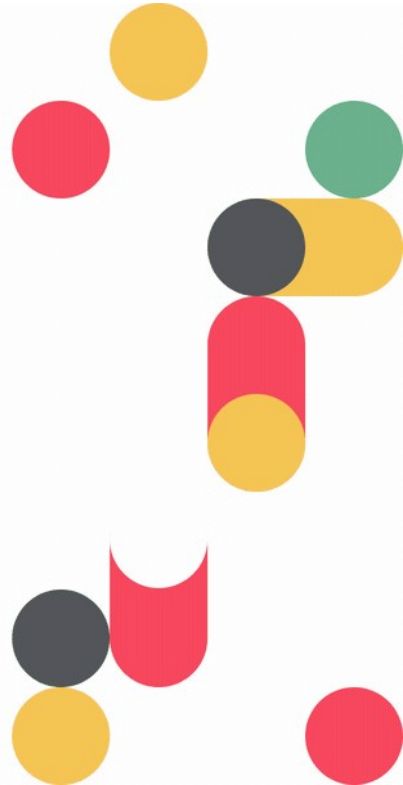
Key Features:

- One Git project and one namespace

Typical use:

Hosting one project on a shared platform where you have no cluster admin right

5 Attention points



Attention points

Your Git is your source of truth

MAKE BACKUPS

MAKE IT HIGHLY AVAILABLE

MAKE IT SECURE

USE PROPER GITFLOW

DO NOT TRY TO CIRCUMVENT IT

ENSURE IT CAN COPE WITH THE LOAD

Attention points

Prepare your content correctly

When possible, do not include values that will be overwritten by the cluster itself, like revision numbers, generation, status, ...

In Openshift, if you use deploymentConfigs with automatic rollout of new images, do not populate the image field in the deployment template, let the update trigger fill it for you.

```
triggers:
  - imageChangeParams:
      automatic: true
      containerNames:
      - jenkins
      from:
        kind: ImageStreamTag
        name: jenkins:2
        namespace: openshift
```

Attention points

What about secrets?

Never store them in plain text in Git!

Whenever possible, use a vault solution like those from Hashicorp or CyberArk

If not possible, use a tool like bitnami's sealed-secrets.

6

Tools

Tools

There are plenty...

Box/Kube-applier: simple controller, no pruning, can be a bit rough for complex environments. Pull model

Certified Jenklins: the jenkins provided along with Openshift can be easily scripted to match almost the functional level of the Kube-applier while also providing webhooks. Only push model

Weaveworks Flux: a cloud-native minded operator with some modularity. Pull model

Intuit Argo CD: a full-fledged operator-based solution with UI and API, Argo is well rooted in K8S best practices and can work in multicluster out of the box. Jenkins plugin. Pull and Push model

GitOps engine: a work-in-progress produced by authors of both Argo and Flux solutions.

Questions ?

Demo material

Kube-applier version used for the presentation:
- source: https://github.com/brobaldv/kube-applier
- oci image: https://hub.docker.com/r/vincentbrobald/oc-applier

Objects used for the demo:
- source: https://github.com/brobaldv/rhtd2020-gitops-democontent01.git