OpenShift GitOps

Hands-On Workshop

Arunkumar Hariharan
Senior Specialist Solution Architect

Geoff Allen
Senior Specialist Solution Architect

Red Hat

Agenda

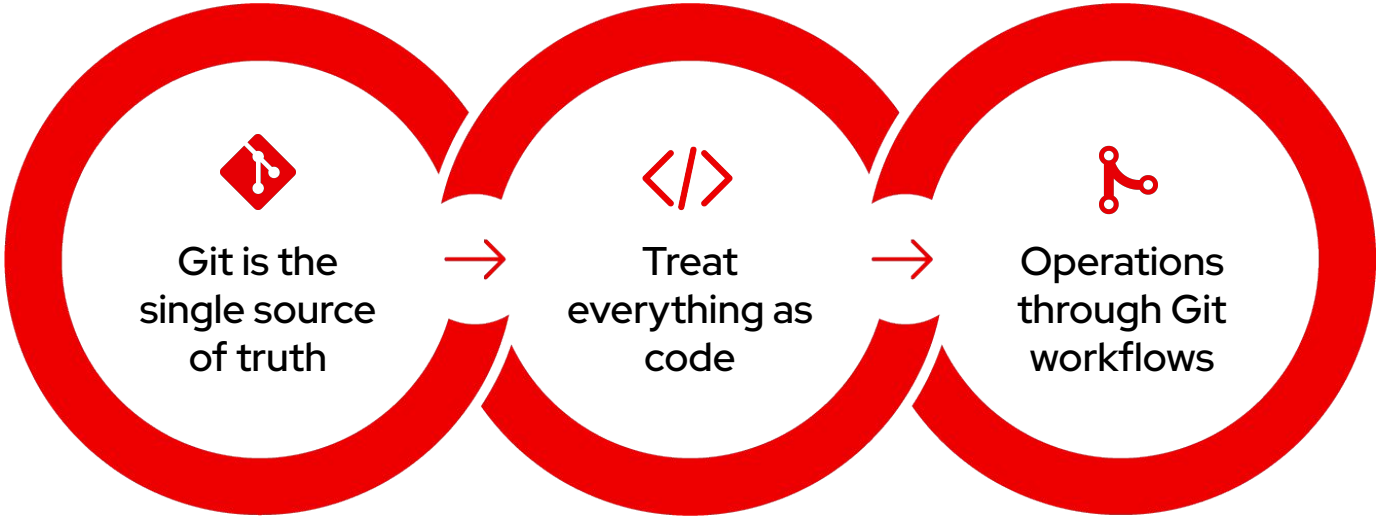
- Current CoBank CI/CD
- GitOps Introduction
- WorkShop Overview
- Hands-on Workshop
- Lunch
- Catalyst Activities - CoBank Specific Use Cases

Current CoBank CI/CD Pipeline

What is GitOps?

An **developer-centric** approach to **C**ontinuous **D**elivery and **infrastructure** operation

Developers



Operations

Red Hat

GitOps Principles

- ***Declarative***
  A system managed by GitOps must have its desired state expressed declaratively.

- ***Versioned and immutable***
  The desired state is stored in a way that enforces immutability and versioning and retains a complete version history.

- ***Pulled automatically***
  Software agents automatically pull the desired state declarations from the source.

- ***Continuously reconciled***
  Software agents continuously observe the actual system state and attempt to apply the desired state.

https://opengitops.dev/

GitOps is for Everyone

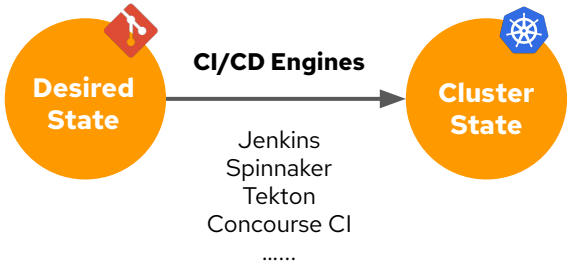
Developers

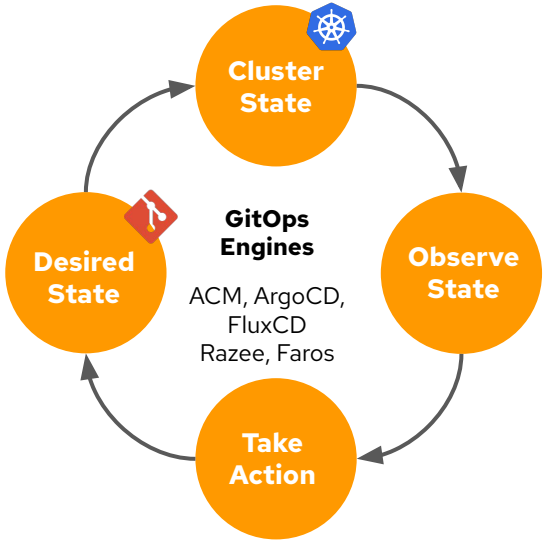
Operations

Source: GitOps-Cookbook-Red-Hat-Developer-ebook

Red Hat

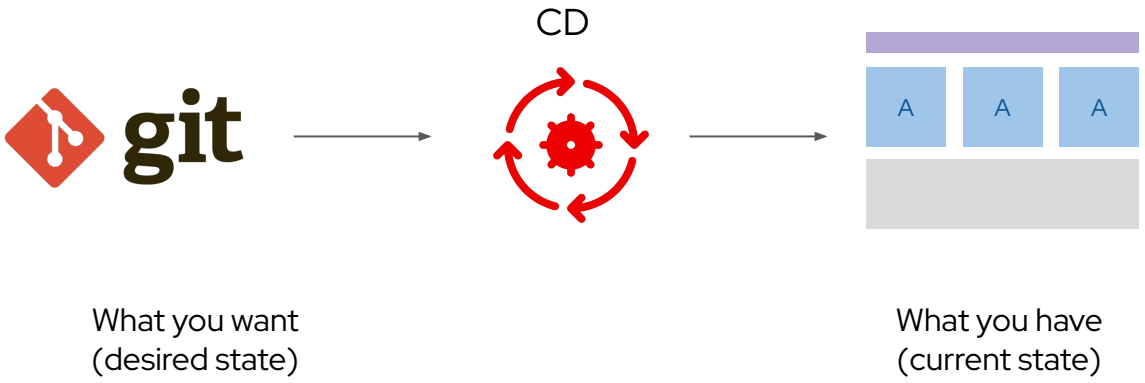
GitOps versus CI/CD





Red Hat

GitOps Workflow



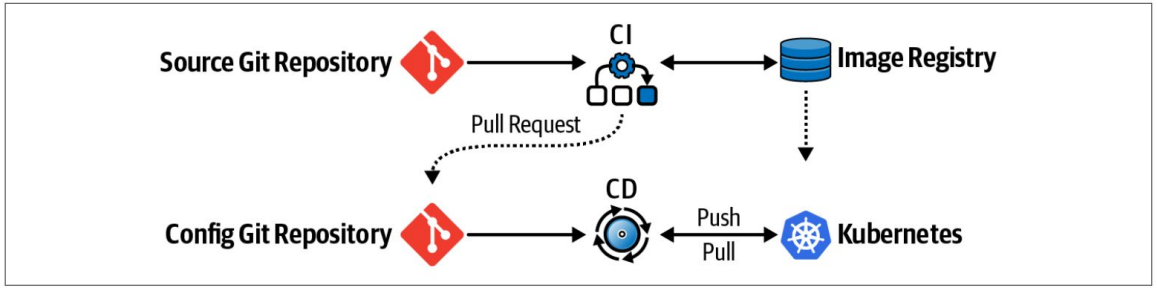


Figure 1-2. Application deployment model

Source: GitOps-Cookbook-Red-Hat-Developer-ebook

Why GitOps?

Standard Workflow

Familiar tools and Git workflows from application development teams

Enhanced Security

Review changes beforehand, detect configuration drifts, and take action

Visibility and Audit

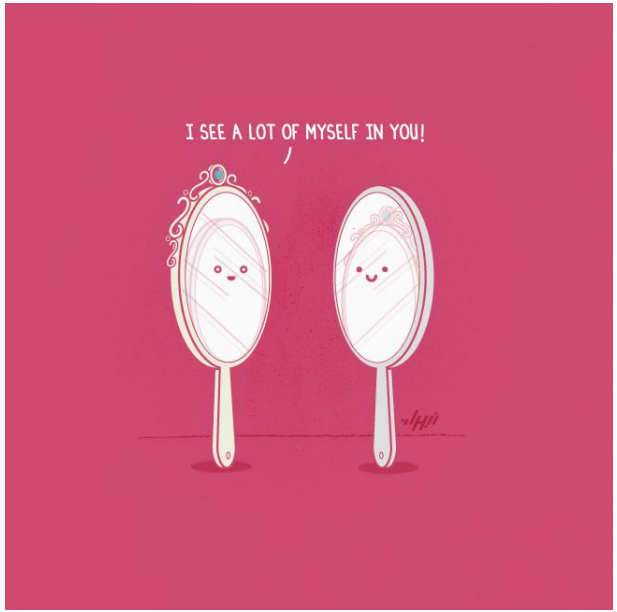
Capturing and tracing any change to clusters through Git history

Multi-cluster consistency

Reliably and consistently configure multiple Kubernetes clusters and deployment

Kubernetes and GitOps - A Perfect Match

- Kubernetes is a declarative environment
  - Application deployments are declared and Kubernetes scheduler makes it happen
  - OpenShift goes further in that cluster configuration is declared and Operators make it happen
- GitOps in **traditional environments requires automation/scripting**, declarative environment minimizes or eliminates this need
- Declarations are **yaml files** which are easily stored and managed in git



OpenShift GitOps

Multi-cluster config management

Declaratively manage cluster and application configurations across multi-cluster OpenShift and Kubernetes infrastructure with Argo CD

Automated Argo CD install and upgrade

Automated install, configurations and upgrade of Argo CD through OperatorHub

Opinionated GitOps bootstrapping

Bootstrap end-to-end GitOps workflows for application delivery using Argo CD and Tekton with GitOps Application Manager CLI

Deployments and environments insights

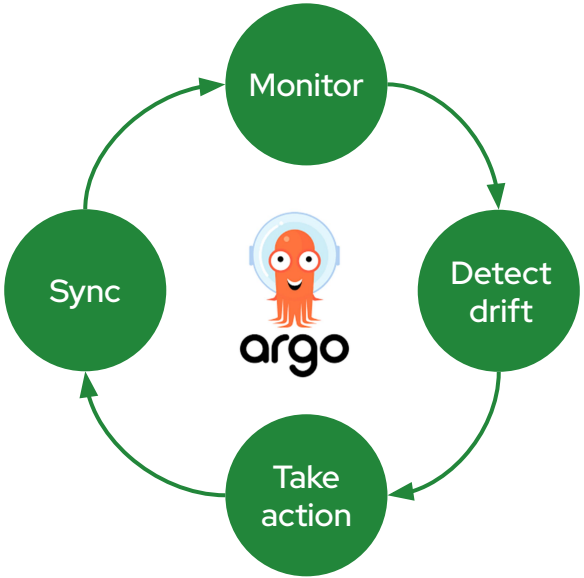
Visibility into application deployments across environments and the history of deployments in the OpenShift Console

Powered by argo

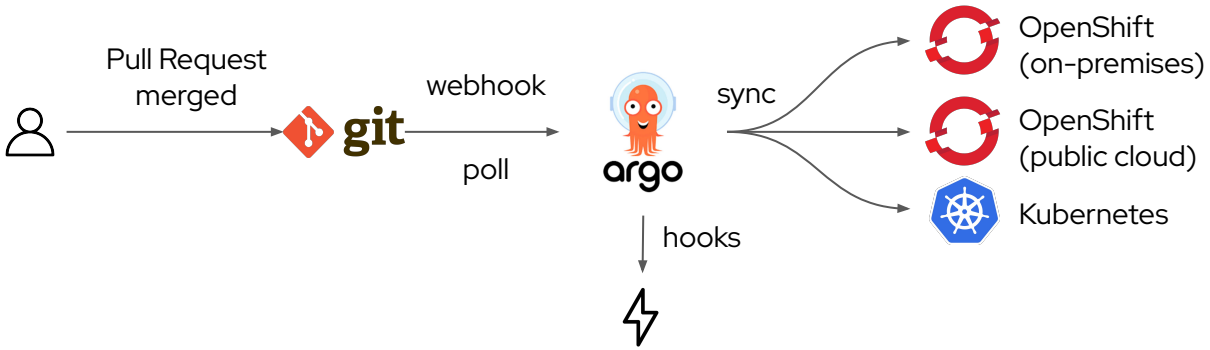
Red Hat

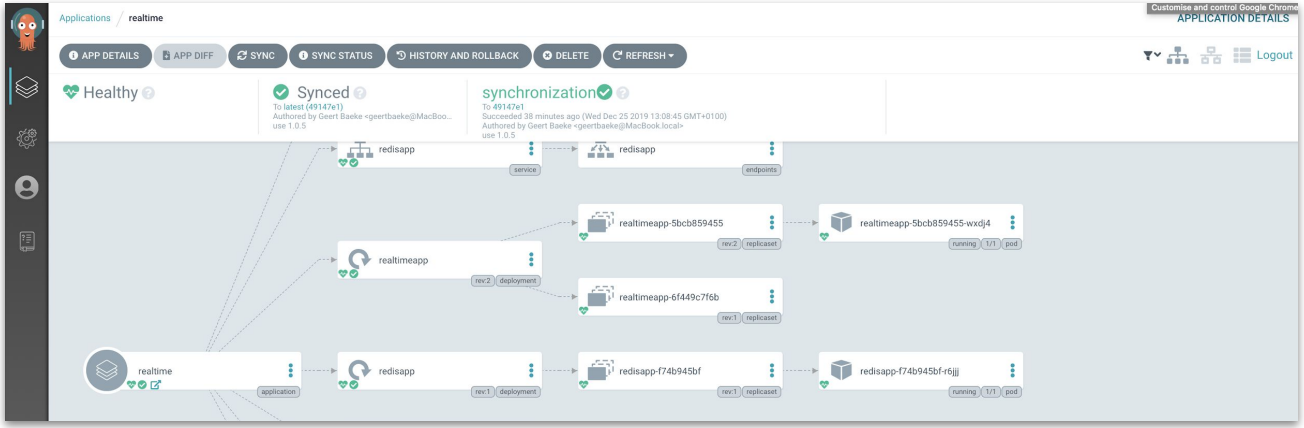
Argo CD

- Cluster and application configuration versioned in Git
- Automatically syncs configuration from Git to clusters
- Drift detection, visualization and correction
- Granular control over sync order for complex rollouts
- Rollback and rollforward to any Git commit
- Manifest templating support (Helm, Kustomize, etc)
- Visual insight into sync status and history



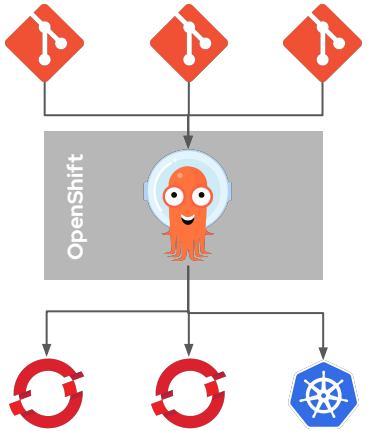
Argo CD





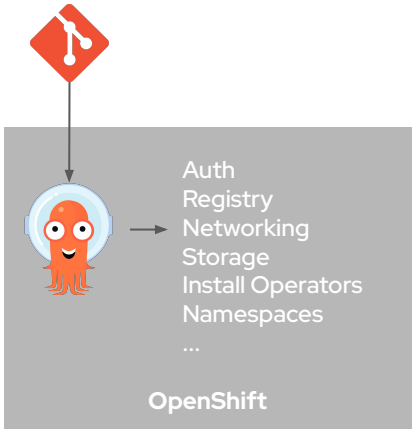
Red Hat

Flexible Deployment Strategies



Central Hub (Push)

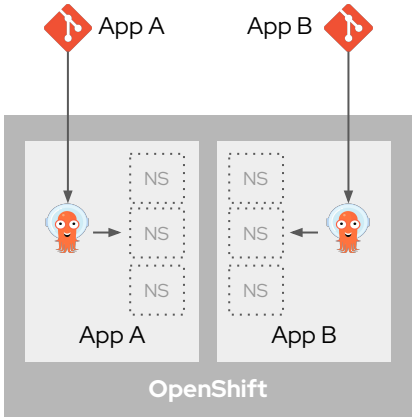
A central Argo CD pushes Git repository content to remote OpenShift and Kubernetes clusters



Cluster Scoped (Pull)

A cluster-scope Argo CD pulls cluster service configurations into the OpenShift cluster

Workshop Example



Application Scoped (Pull)

An application scoped Argo CD pulls application deployment and configurations into app namespaces

Red Hat

What is an Argo CD Application?

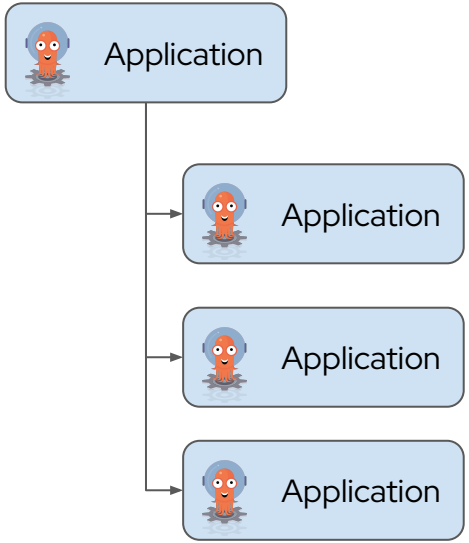
- Argo CD Application is a Custom Resource (CR) that defines the app in a declarative manner
- Application definition includes:
  - Name
  - Cluster
  - Git repository
  - Synchronization Policy
- Applications can be deployed from Argo CD GUI or CLI (argocd or kubectl or oc)

```yaml
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: product-catalog-dev
  namespace: argocd
spec:
  destination:
    namespace: argocd
    server: https://kubernetes.default.svc
  project: product-catalog
  source:
    path: manifests/app/overlays/dev-quay
    repoURL: https://github.com/gnunn-gitops/product-catalog.git
    targetRevision: master
  syncPolicy:
    automated:
      prune: false
      selfHeal: false
```

Red Hat

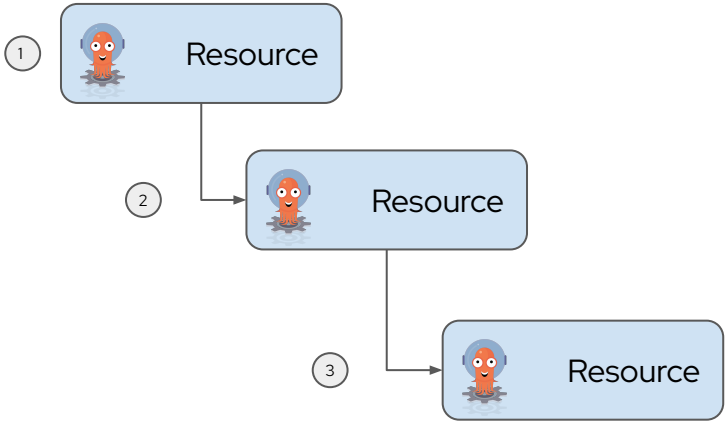
Argo CD “App of Apps”

- App of Apps is a common pattern where one Argo CD Application points to a repo that only contains other Argo CD Applications
- Very useful to be able to provision and manage a group of related applications together
- Pattern evolved over time, ApplicationSets are now available as well.



Argo CD Sync Waves

- Sometimes you need to deploy resources in a specific order due to dependencies, pre-reqs, etc
- Argo CD enables you to order the deployment using Sync Waves. Next resource will not deploy until the first resource is “healthy”
- Very useful when resources have hard dependencies versus eventual consistency



GitOps - Avoiding Duplication

GitOps enables deployment across multiple environments and clusters, awesome!

Wait, how do we manage configuration without copying and pasting yaml everywhere?

Helm is a package manager for Kubernetes applications that manages manifests via **templating**

define, install and update applications

Kustomize is a tool for customizing Kubernetes manifests via **patching**, it is built natively into the “kubectl/oc” cli with a standalone “kustomize” cli for advanced features.

Red Hat

What does it look like?

Kustomization

A file that declares any resources, and any customization to apply to them, e.g. add a common label or namespace

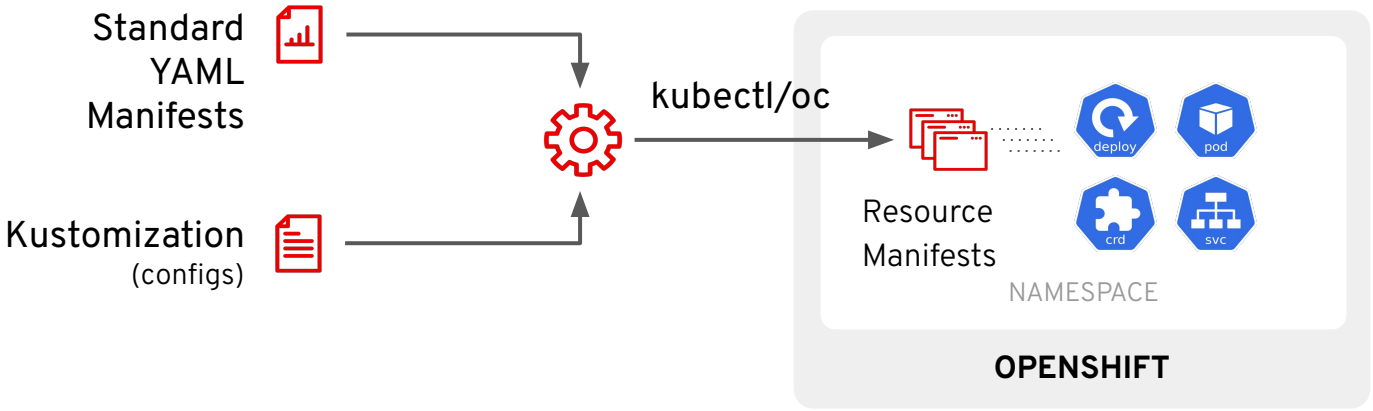
Overlay

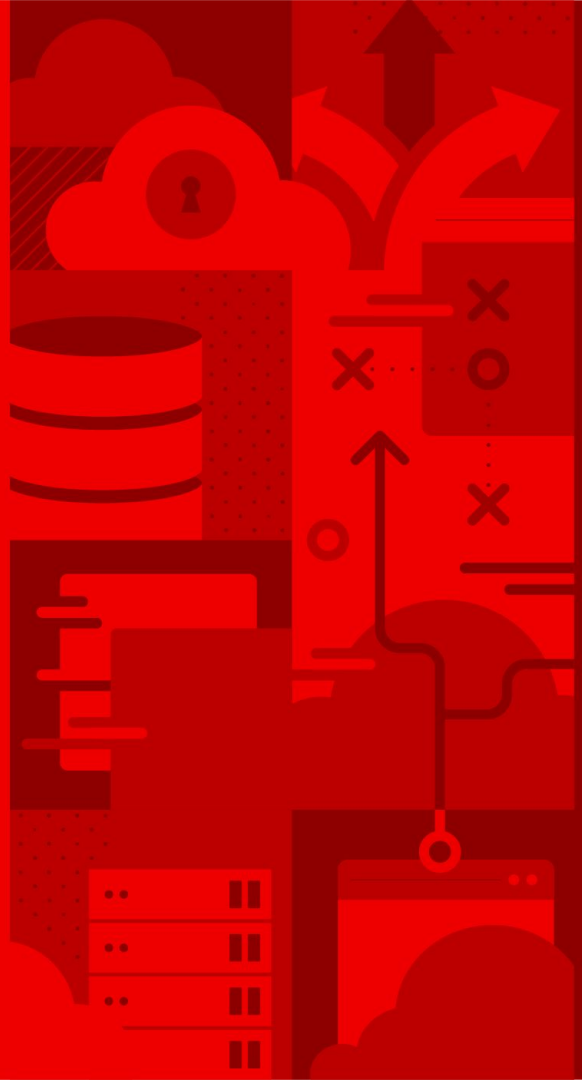
Manage variants of a configuration using overlays that modify a common base.

Example folder structure

```
~/someApp
├── base
│   ├── deployment.yaml
│   ├── kustomization.yaml
│   └── service.yaml
└── overlays
    ├── development
    │   ├── cpu_count.yaml
    │   ├── kustomization.yaml
    │   └── replica_count.yaml
    └── production
        ├── cpu_count.yaml
        ├── kustomization.yaml
        └── replica_count.yaml
```

How does it work?



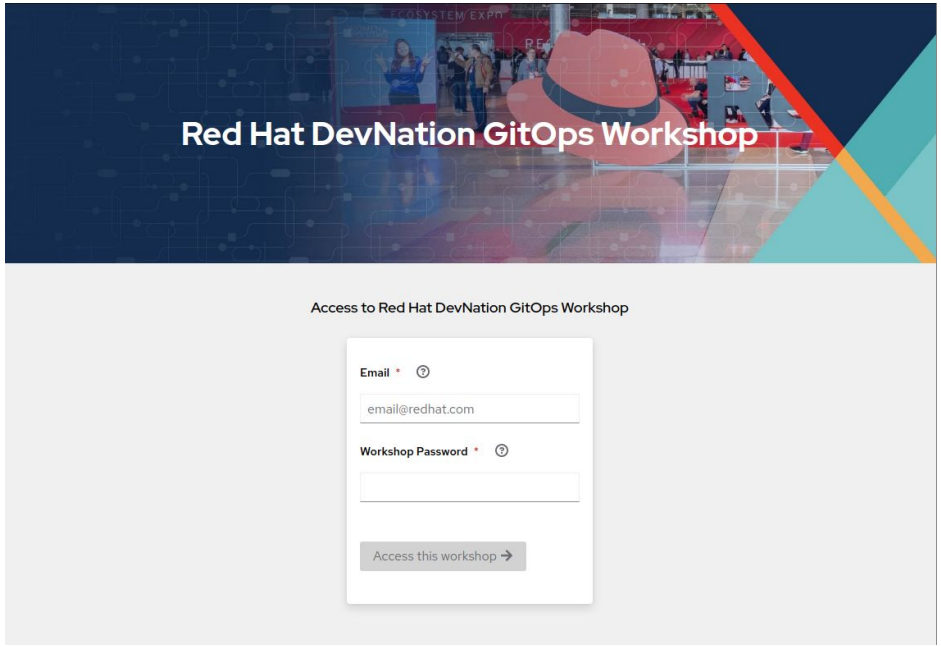


Workshop

Red Hat

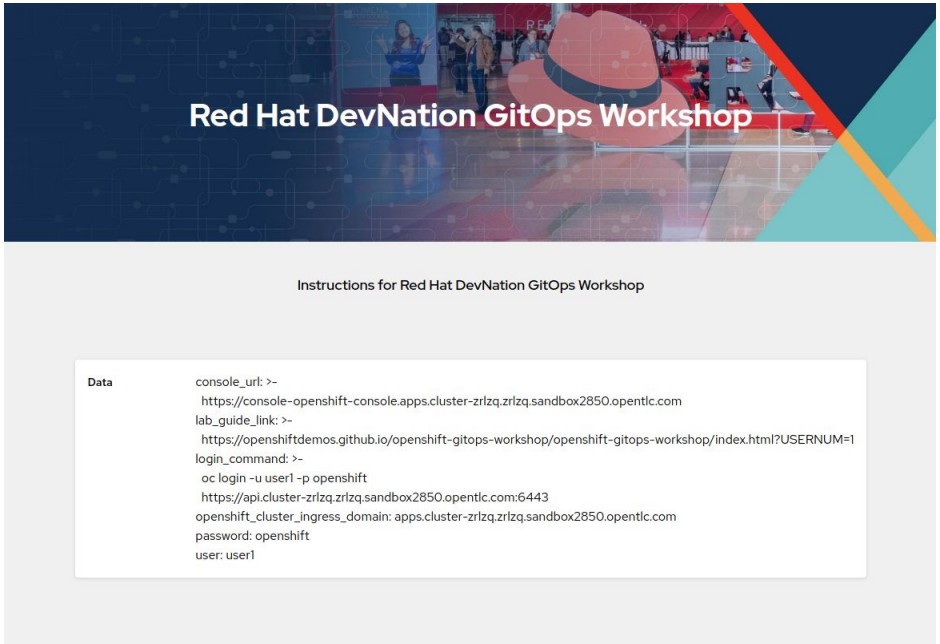
Register For Assigned User Number

- Register to receive an assigned user number (1,2,3, etc)
- You will be logging into the cluster with “userX” where X is your number
- The password for your user will be “openshift”



Register For Assigned User Number

- Once you register you will see the screen on the right
- It provides all of the information you need to access the workshop environment including:
  - Console URL
  - Lab Guide URL
  - Password
  - User (user1, user2, etc)



Links

console_url: >-
https://console-openshift-console.apps.cluster-hxt7v.hxt7v.sandbox364.opentlc.com

lab_guide_link: >-
https://openshiftdemos.github.io/openshift-gitops-workshop/openshift-gitops-workshop/index.html?USERNUM=1

login_command: >-
oc login --insecure-skip-tls-verify=false -u user1 -p openshift
https://api.cluster-hxt7v.hxt7v.sandbox364.opentlc.com:6443

openshift_cluster_ingress_domain: apps.cluster-hxt7v.hxt7v.sandbox364.opentlc.com

openshift_console_url: >-
https://console-openshift-console.apps.cluster-hxt7v.hxt7v.sandbox364.opentlc.com

password: openshift
user: user1

Red Hat

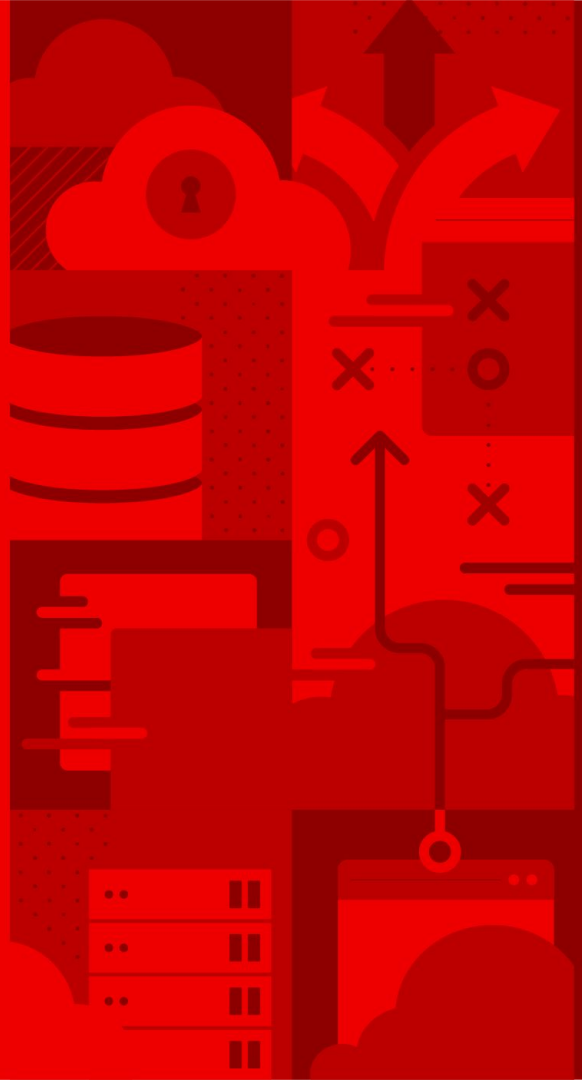
Workshop Information

URL to access this workshop is:

https://bit.ly/cobank-081623

Password to use to register for the workshop:

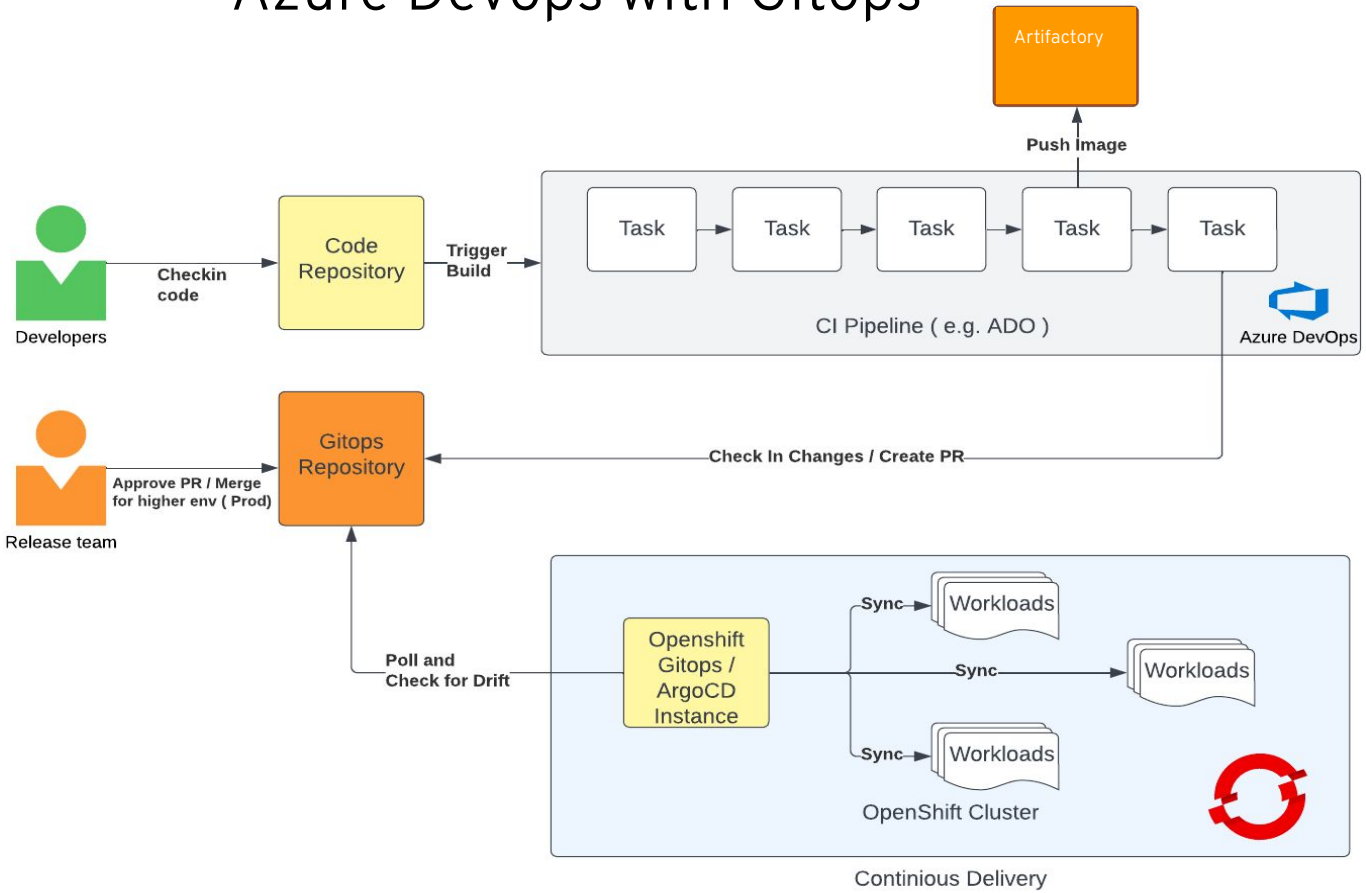
openshift



Catalyst

Red Hat

Azure Devops with Gitops



Red Hat

Questions?

linkedin.com/company/red-hat

youtube.com/user/RedHatVideos

facebook.com/redhatinc

twitter.com/RedHat

Red Hat