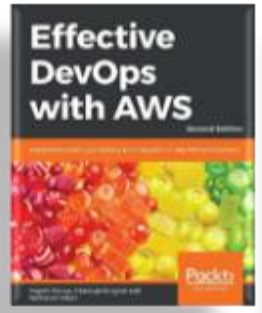
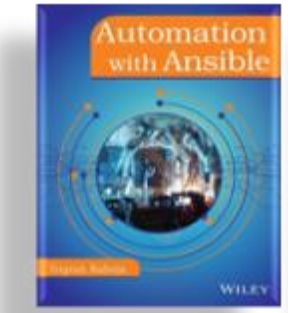
Flux CD for the Absolute Beginners - Hands-On

Yogesh Raheja





Core Concepts

Flux CD Controllers

- Source
- Kustomize
- Helm
- Image Reflector
- Image Automation
- Notification Controller

Application Deployment

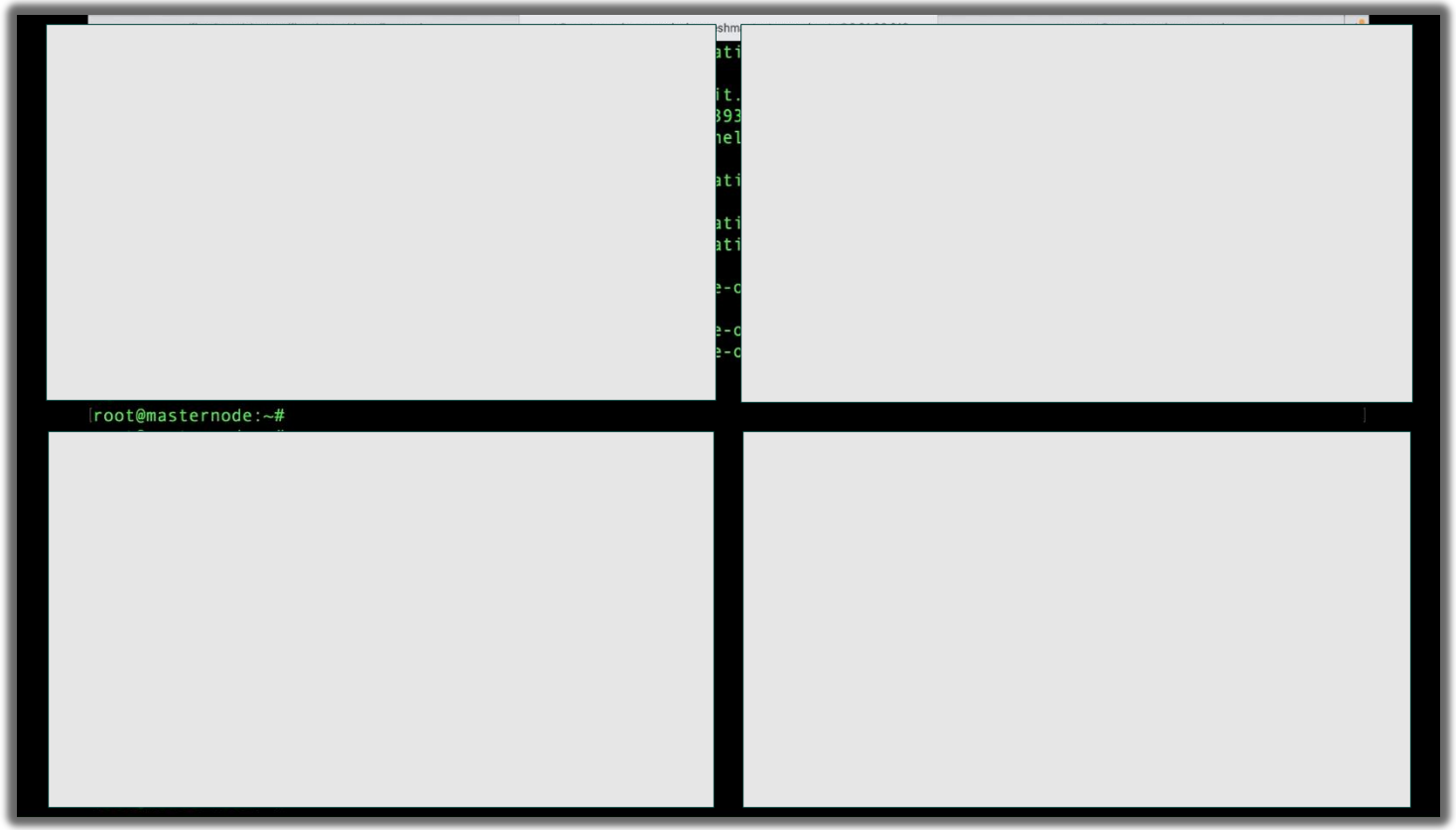
- Manifests
- Helm
- Kustomize

Course Workflow

Lectures

Live Demonstrations

Assignments



Course Workflow

- GitOps Principles
- Features & necessity of Flux CD
- Set up and manage Flux CD environments
- Flux CD architecture and core components
- Use Source, Kustomize, and Helm Controllers

Course Workflow

Application Deployment using GitOps Workflow

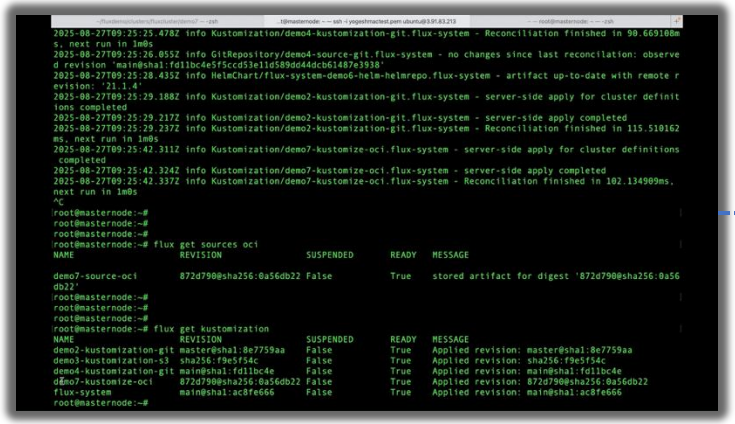
Private Git repositories

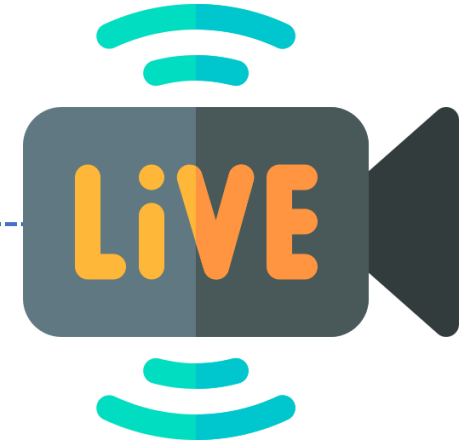
Automating image updates with the Image Automation Controller

Notification Controller

Monitoring & explore open-source UI options

Live Demonstrations





Lectures

Section: 1
Introduction to GitOps and Flux CD

Introduction to GitOps and Flux CD

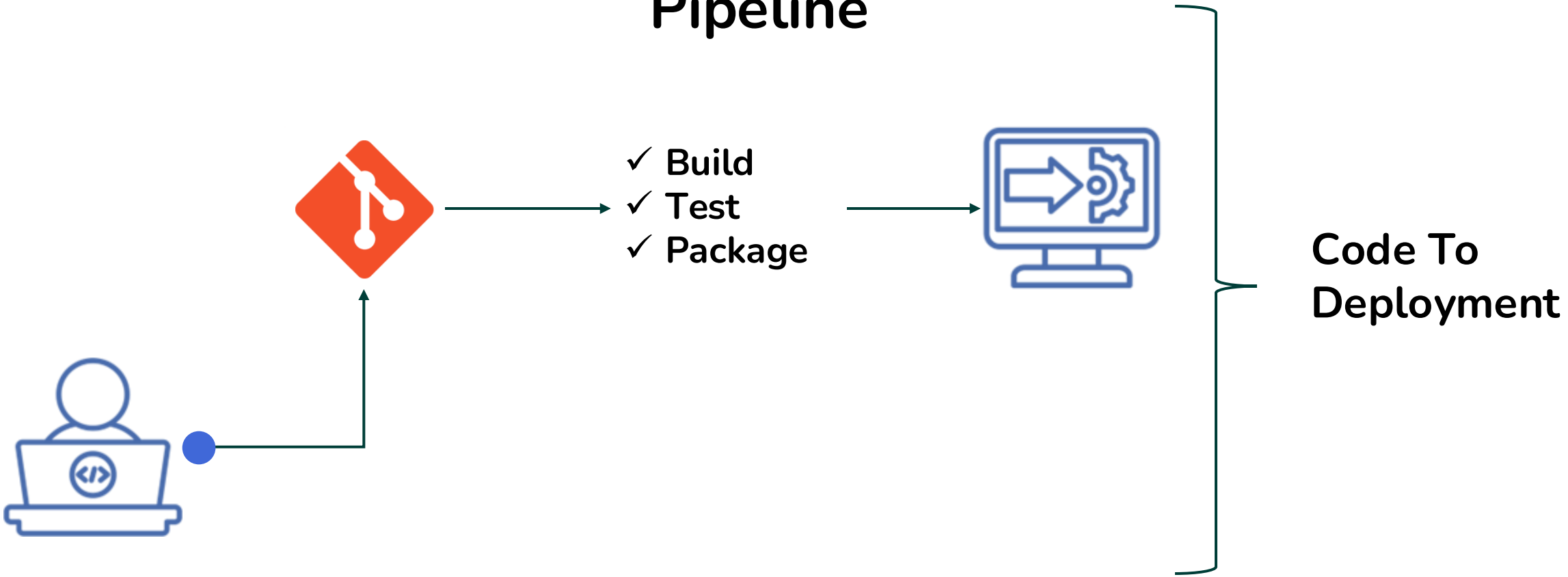
Section Overview

- Introduction to GitOps
- Flux CD
- Official Flux CD documentation

Introduction to GitOps

GitOps

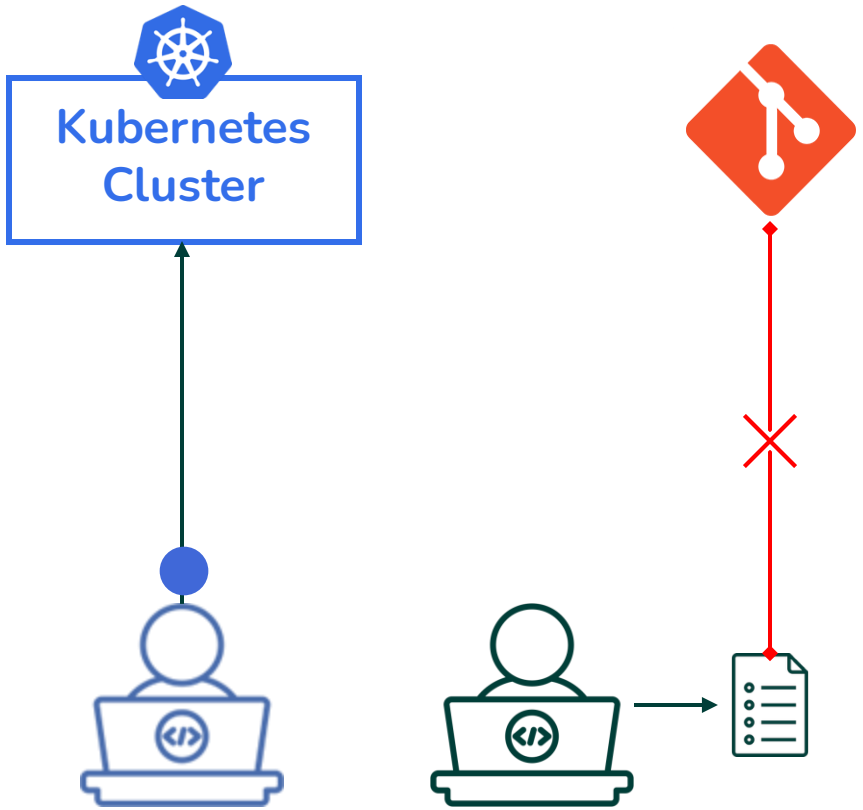
CI/CD Pipeline



Who is managing the infrastructure configurations?

How are changes to Kubernetes manifests or cluster configurations applied?

- Manually running **kubectl** commands
- Manually editing **YAML** files



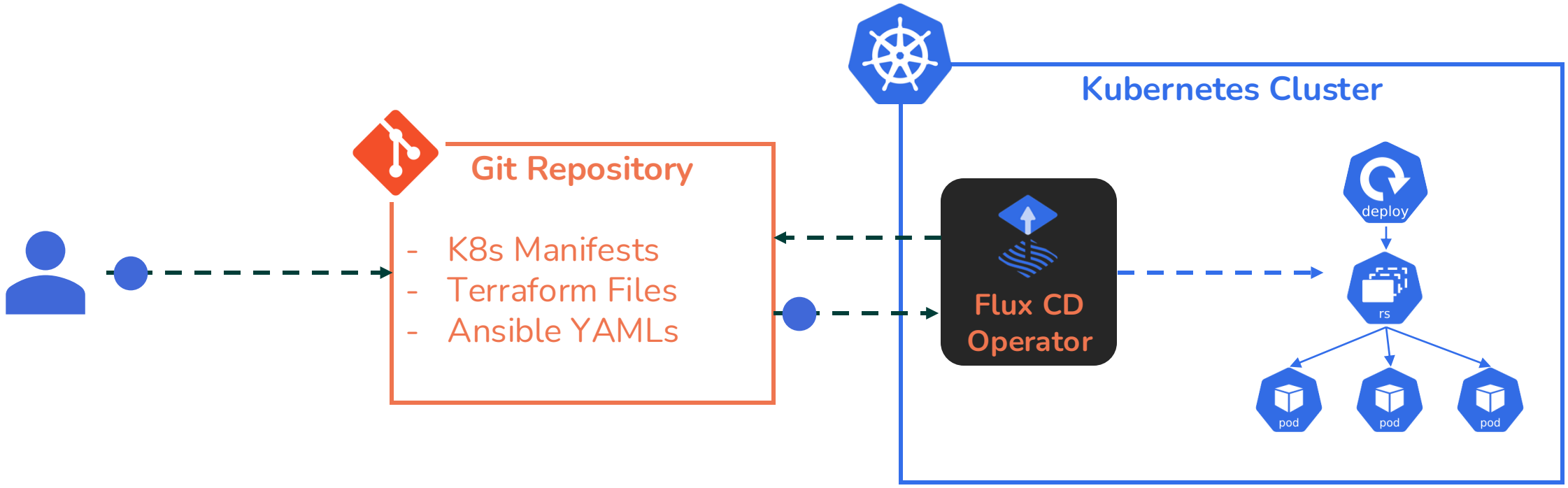
No Audit Trail

Configuration Drift

Disaster Recovery Nightmare

GitOps

Infrastructure & cluster configurations



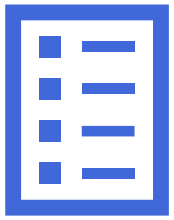
GitOps

- ✓ Every change to cluster is stored in Git, creating a single source of truth
- ✓ Everything is automated and versioned
- ✓ In case of failure, you can restore the cluster by syncing with Git
- ✓ GitOps operator ensures the cluster's live state matches what's in Git, preventing unwanted changes

GitOps

GitOps is a modern approach to continuous deployment that uses Git as the single source of truth for your application code & infrastructure configurations

Four Principles of GitOps



Declarative Configuration

Version Control

Automated Delivery

Continuous Reconciliation

- **Observe**
- **Diff**
- **Act**

Introduction to Flux CD

Introduction to Flux

Weaveworks 2016

CLOUD NATIVE
COMPUTING FOUNDATION

What is Flux CD?

Flux CD is a **GitOps-native continuous delivery tool** for Kubernetes. Its core purpose is to keep your cluster in perfect sync with the state defined in Git

✓ Keeps an eye on what's running in your cluster.

✓ Checks what's stored in Git

✓ And if there's any difference? Flux quietly steps in and fixes it, no questions asked

Why Flux CD?

Declarative & Version Controlled

Automation & Remediation

Scalability & Flexibility

How Flux CD Works?

Single source of truth

How Flux CD Works?

Fetching Configurations

Monitoring State

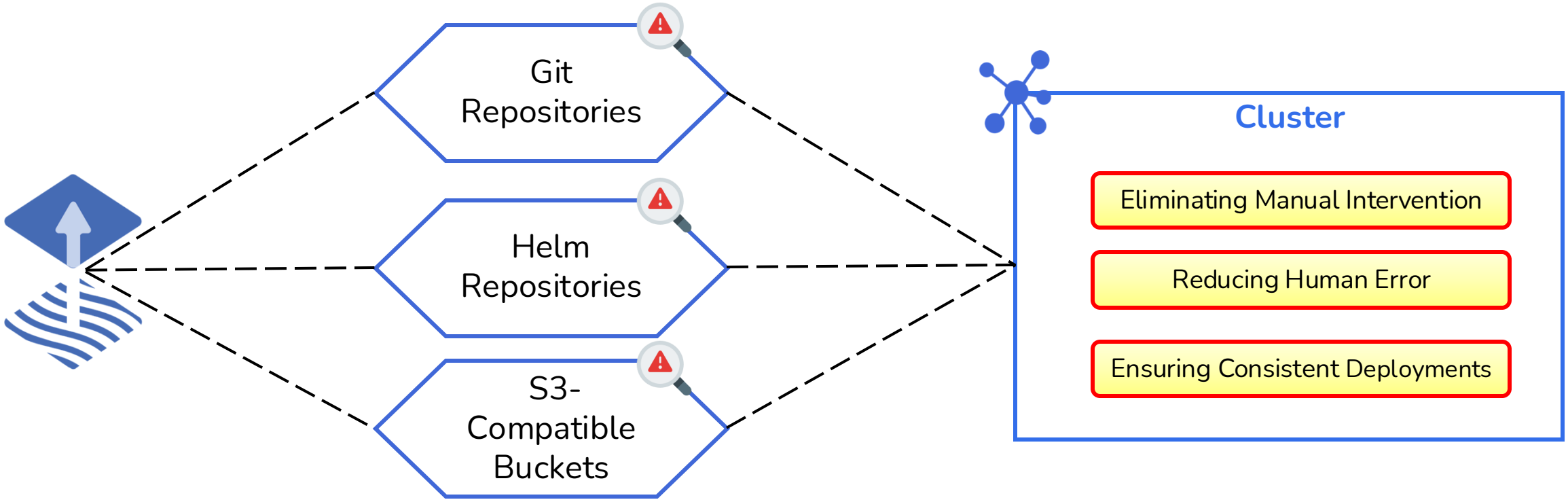
Reconciliation Loop

Modular Controllers

Features of Flux CD

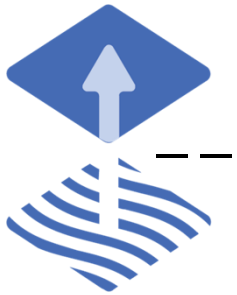
Features of Flux CD

1 Automated Application Deployment



Features of Flux CD

2 GitOps Workflow & Version Control



GitOps

Git

- K8s Manifests
- Helm charts
- Configurations

- ✓ Version Control
- ✓ Collaboration
- ✓ Auditability

Features of Flux CD

3 Progressive Delivery (with Flagger)

Flagger

- Canary Releases
- Blue/Green Deployments
- A/B testing

Features of Flux CD

4 Secure by Design

Pull-Based deployment model

Cluster fetches updates from Git instead of pushing them—reducing the attack surface

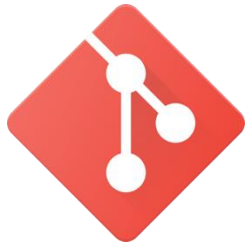
Principle of least privilege

Integrates with Kubernetes RBAC, and supports policy enforcement through **OPA** and **Kyverno**

Features of Flux CD

5 Broad Tooling Compatibility

Configuration Tools



Git providers

Container Registries

Policy Tools

Workflow Engines

Features of Flux CD

6 Health Assessment & Drift Detection

Checks the health of **clusters** and **workloads**. It detects drift between the live and **desired state**, automatically reconciling differences to maintain consistency

Features of Flux CD

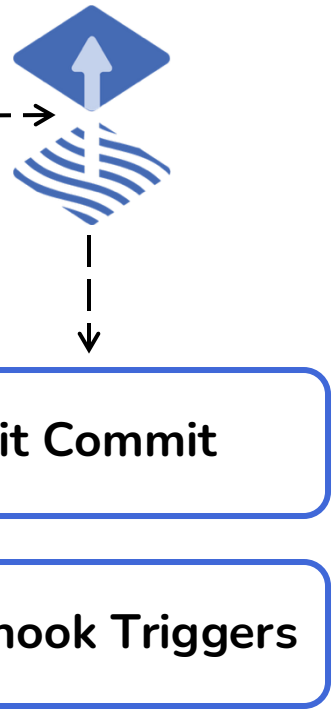
7 Dependency Management

Dependencies between **workloads resources**, ensuring deployments occur in the correct order and without conflicts

Features of Flux CD

8 Event-Triggered and Scheduled Reconciliation (webhook)

- ✓ **In addition to periodic checks**



Features of Flux CD

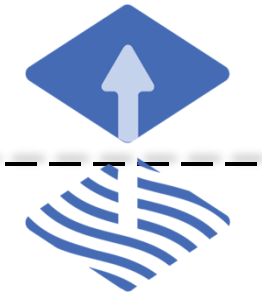
9 Automated Image Updates

- ✓ New image versions
- ✓ Update Kubernetes manifests in Git automatically
- ✓ Trigger Redeployments

Up-to-Date

Features of Flux CD

10 Alerting and Notifications



Integrates with external notification systems (**Slack, MS Teams, webhooks, etc**.) to send real-time alerts on **deployments**, **reconciliation events**, or **policy violations**

Demo

Flux CD Documentation

Section: 2
Understanding Flux CD Framework

Understanding Flux CD Framework

Section Overview

- Core Flux CD Terminologies

Core Flux CD Terminology

Git

Docker

Kubernetes

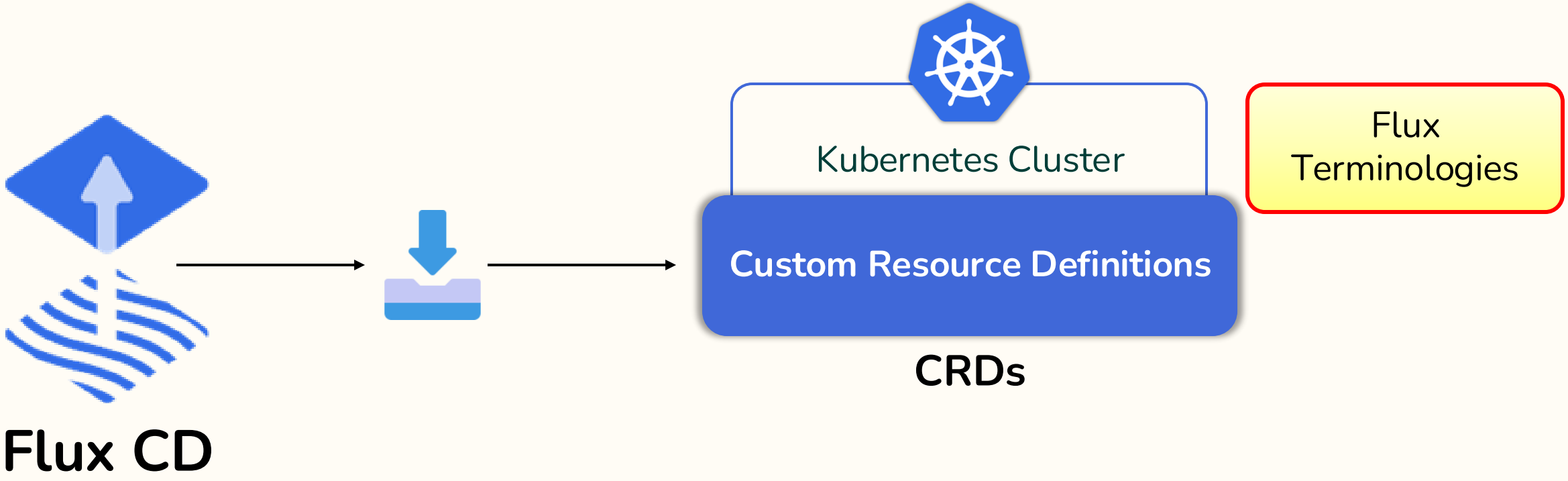
C/D

Continuous Delivery

GitOps

GitOps

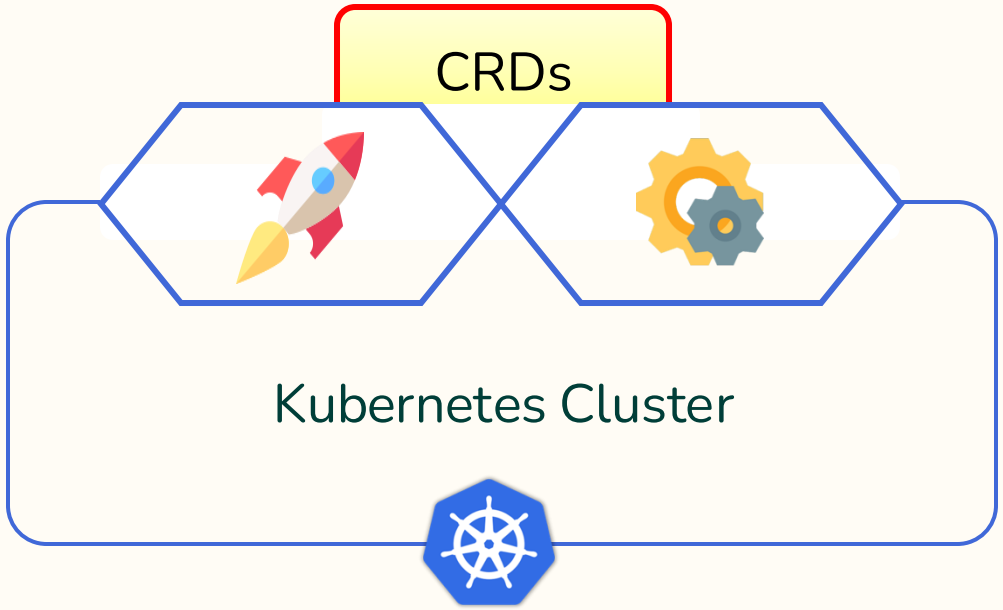
Core Flux CD Terminology



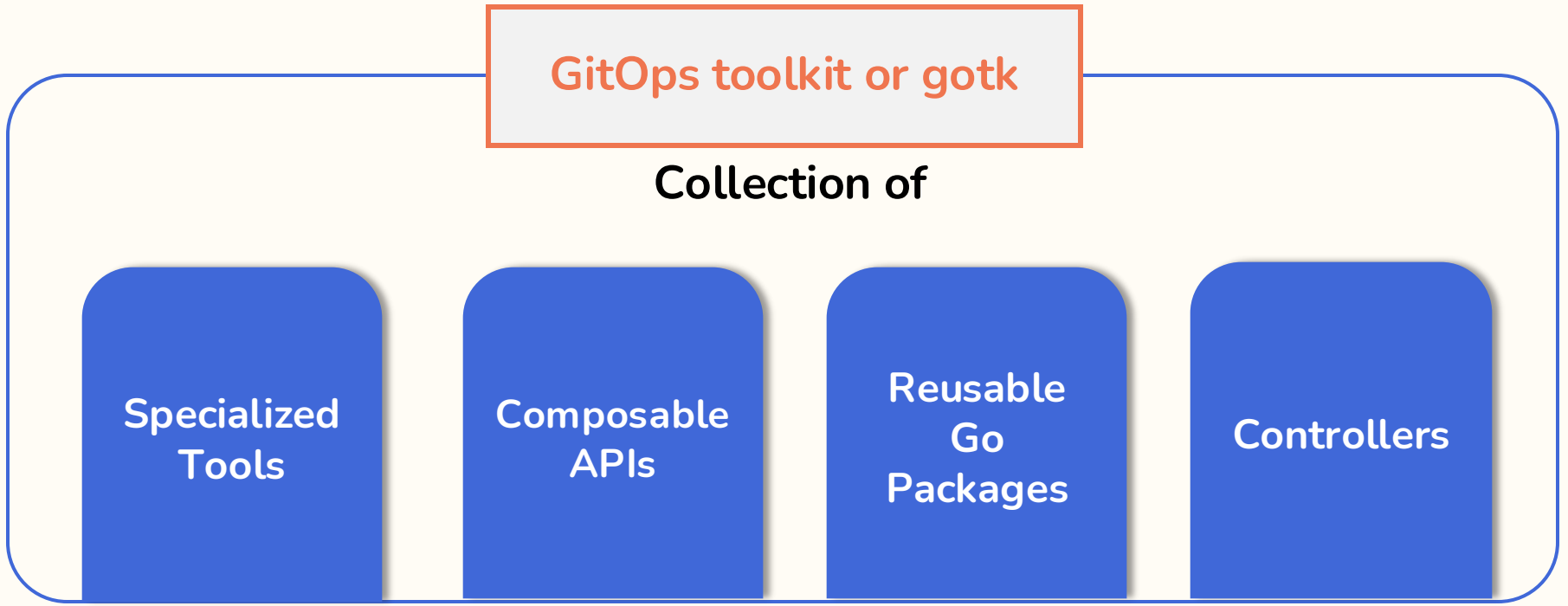
Core Flux CD Terminology

Bootstrap

Bootstrap process is how we install and set up Flux components



Core Flux CD Terminology



Core Flux CD Terminology

GitOps toolkit or gotk

Source Controller

Kustomize Controller

Helm Controller

Notification Controller

Image Automation Controller

Core Flux CD Terminology

Desired State

Desired state is the target configuration of your application as defined in your Git or equivalent repository

Pods

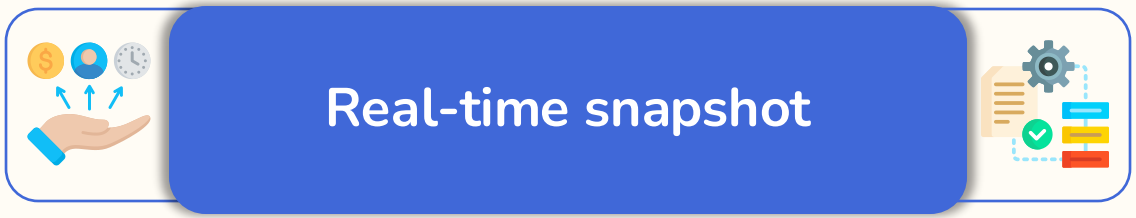
ConfigMaps

Secrets

Core Flux CD Terminology

Live (or Current) State

Live State is what's actually running in your cluster right now



Core Flux CD Terminology

Sources

Source defines where your manifests live

- Git Repositories
- OCI repositories
- Helm Repositories
- Object Storage Buckets

Core Flux CD Terminology

Kustomize or Helm

Kustomize Controller and **Helm Controller** are responsible for reconciling your cluster's live state with the desired state from your sources

Plain YAMLs

Kustomize Overlays

Helm chart

Core Flux CD Terminology

Reconciliation

Reconciliation is the process of making sure the live state in your cluster always matches the desired state in Git

Core Flux CD Terminology

Progressive Delivery

Progressive Delivery is about rolling out changes gradually

- Canary Deployments
- A/B Testing
- Blue-Green Deployments

Core Flux CD Terminology

Flux CLI

Flux CLI is your main tool for interacting with Flux

- Bootstrap Flux
- Manage Controllers
- Perform day-to-day operations

Understanding Flux CD Framework

Summary

- Key Terminologies in Flux CD

Section: 3
Setting up Flux CD

Setting up Flux CD

Section Overview

- Flux CD Installation Options and Prerequisities
- Overview of flux CLI
- Flux CLI Installation and CommonlyuUsed Commands
- Installing Flux CD on Kubernetes

Flux CD Installation Options

Bootstrapping with Flux CLI

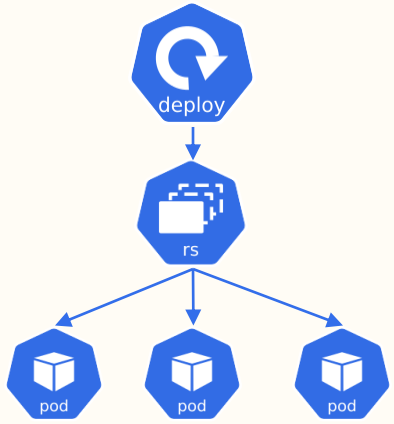
Bootstrapping with Terraform

```
flux_bootstrap_git
```

Bootstrapping with Helm

Flux CD Prerequisites

Kubernetes Cluster



Flux CLI

GitHub account & (PAT)

Git Repository

- Authenticating
- Managing

Overview of Flux CLI

Overview of Flux CLI

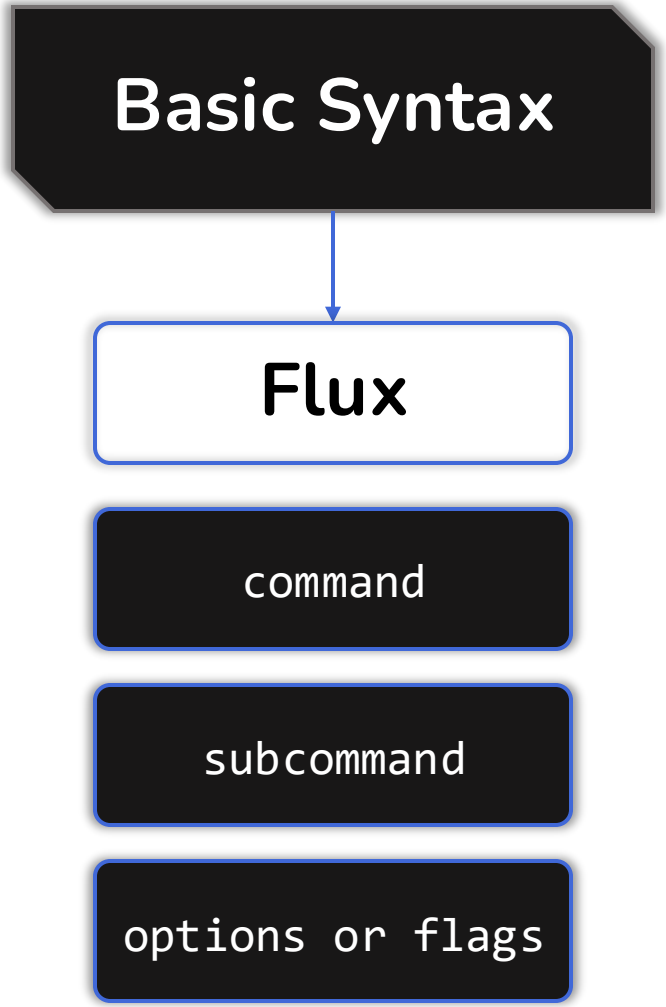
The **command-line utility** that powers Kubernetes continuous delivery pipelines

Overview of Flux CLI

Flux CLI

- Pre-Installation Checks
- Bootstrapping Flux
- Listing Resources
- Suspending or Resuming Resources
- Reconciling Changes
- Creating or Deleting Objects
- Viewing Logs

High-Level Overview of Flux CLI



High-Level Overview of Flux CLI

Flux

Command-Line Tool

`command`

Specifies the action you want to perform

- ✓ `bootstrap`
- ✓ `get`
- ✓ `list`
- ✓ `suspend, resume or delete`

`subcommand`

Gives more detail to the command

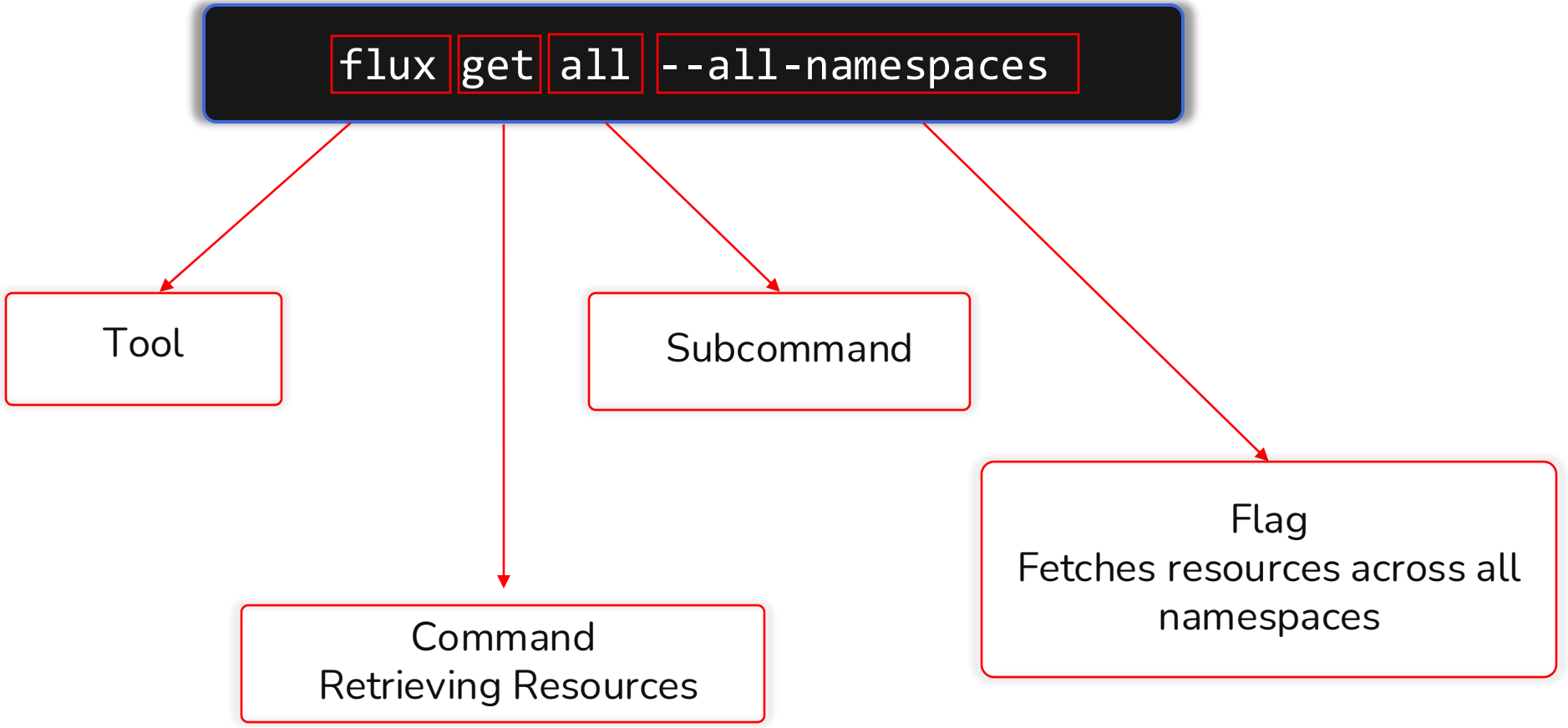
And *`flux get all`* retrieves everything at once

`options or flags`

Customize the behavior of a command

- ✓ `--namespace`
- ✓ `--verbose`
- ✓ `--help`

Practical Example



Practical Example

```
flux get sources git --all-namespaces
```

Structured & Powerful

Bootstrapping the System

Checking Health of Resources

Managing Deployments

Demo

Flux CLI Installation & Commonly Used Commands

Demo

Installing Flux CD on Kubernetes

Flux CD Architecture

Section Overview

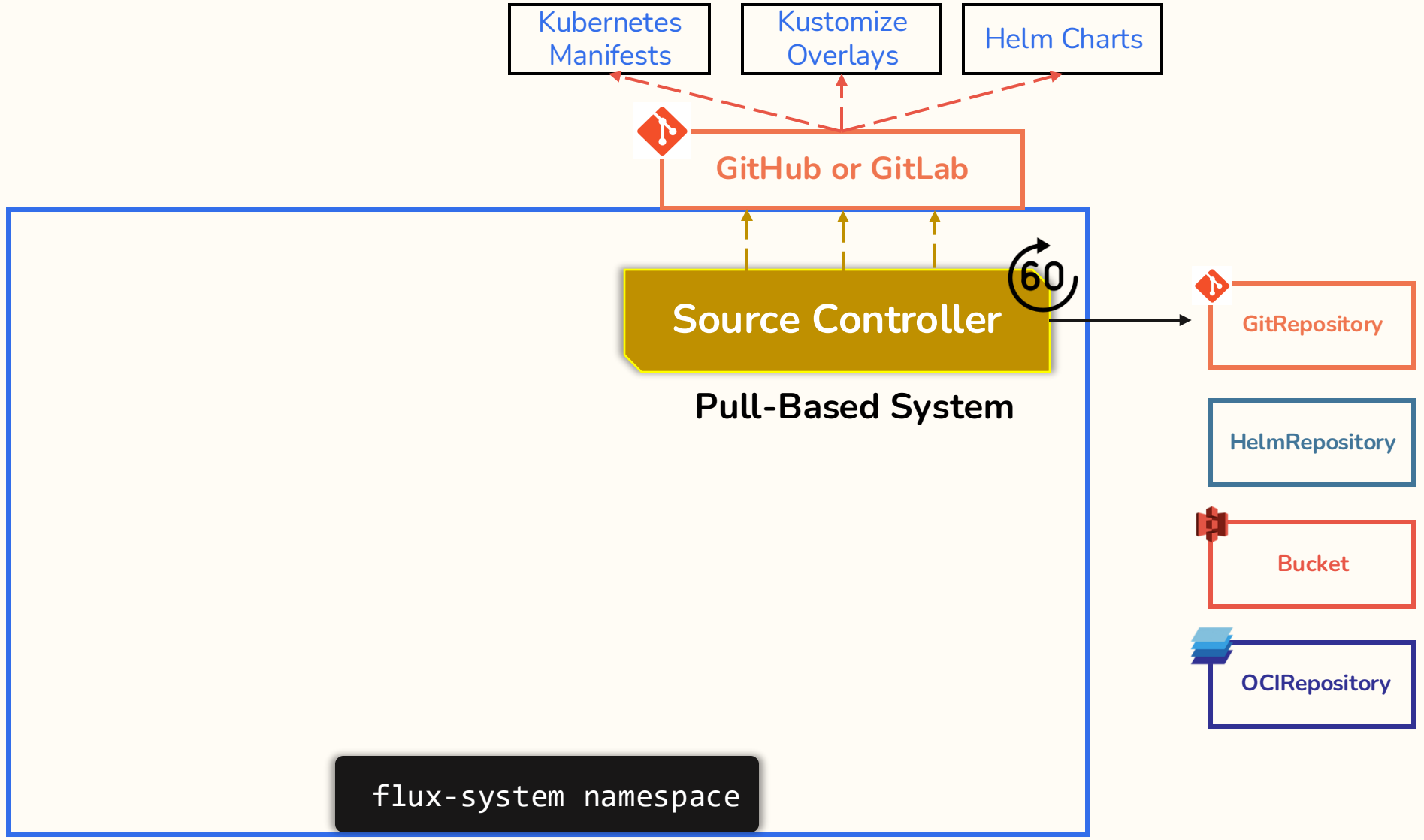
- Flux CD Architecture

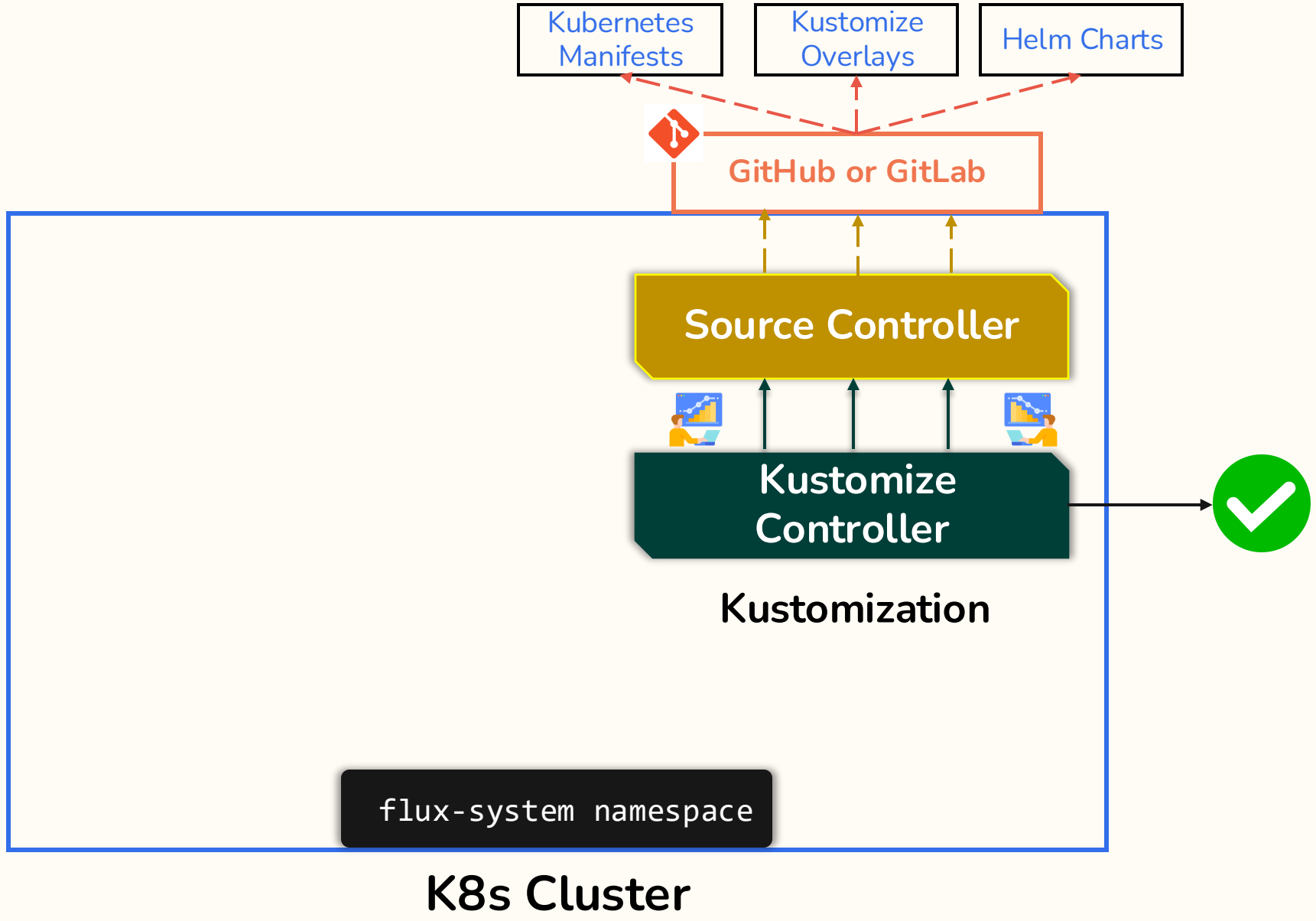
Flux CD Architecture

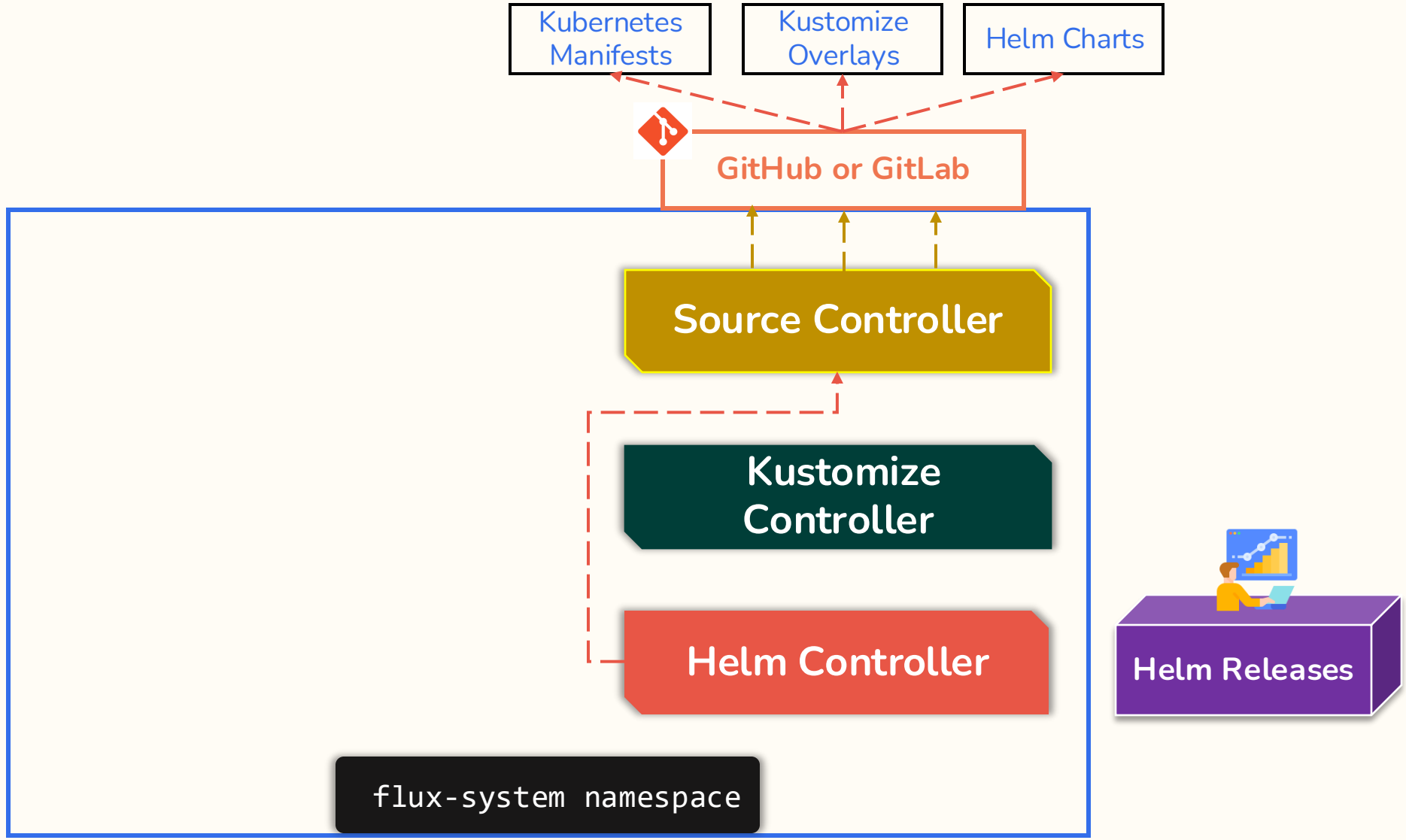
Flux CD Architecture

`flux-system namespace`

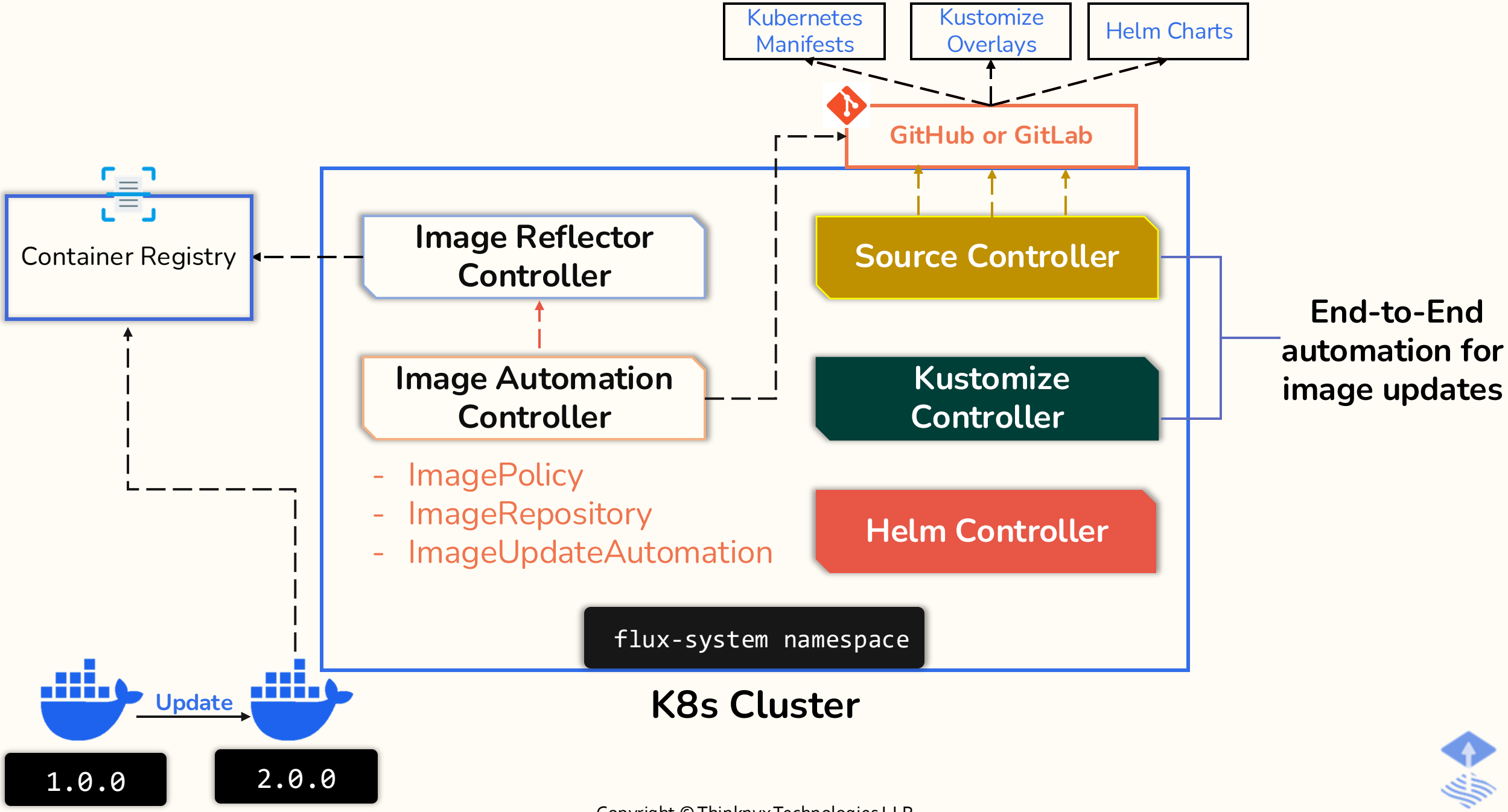
Custom Resource Definitions or **CRDs**

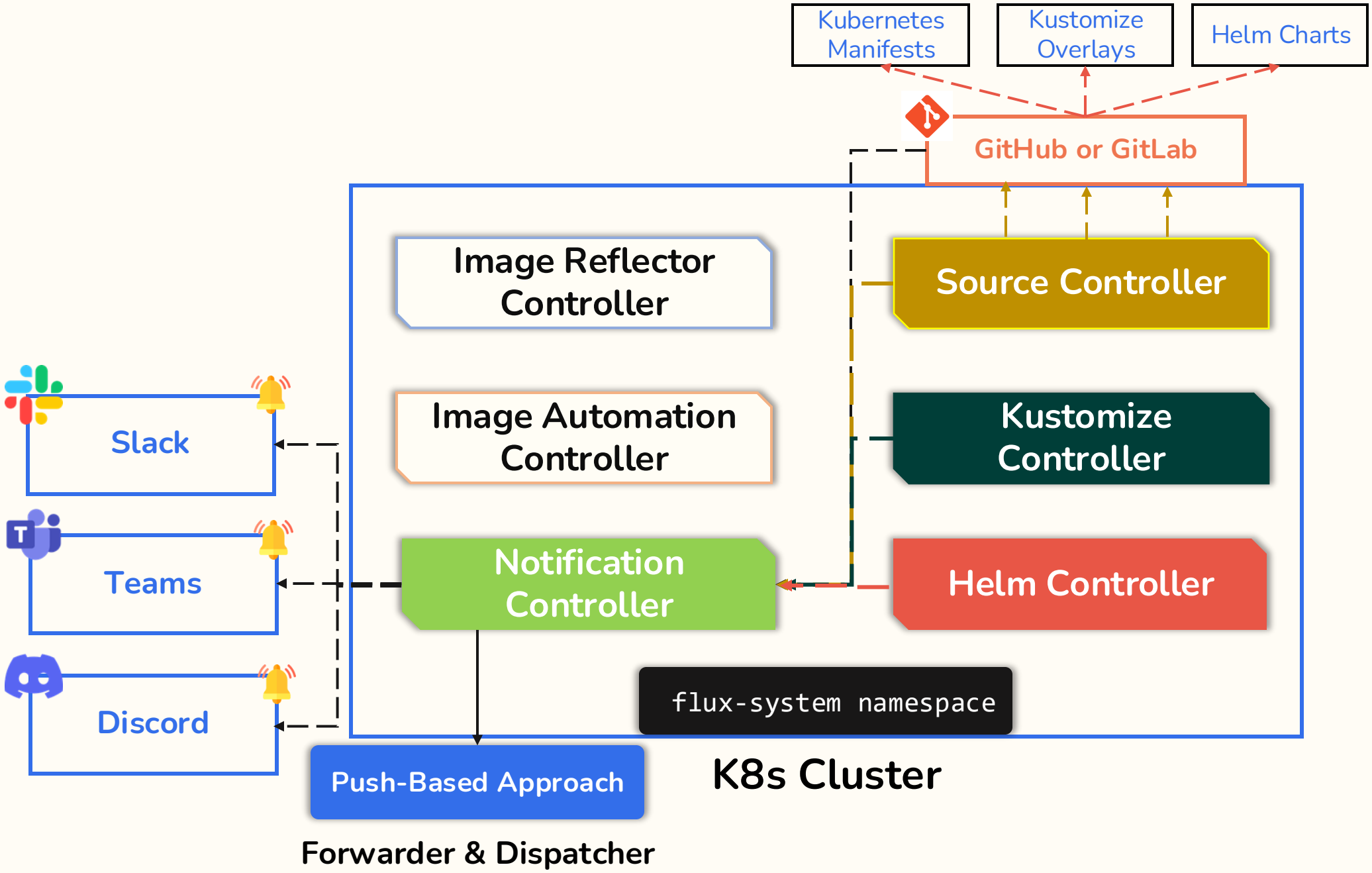


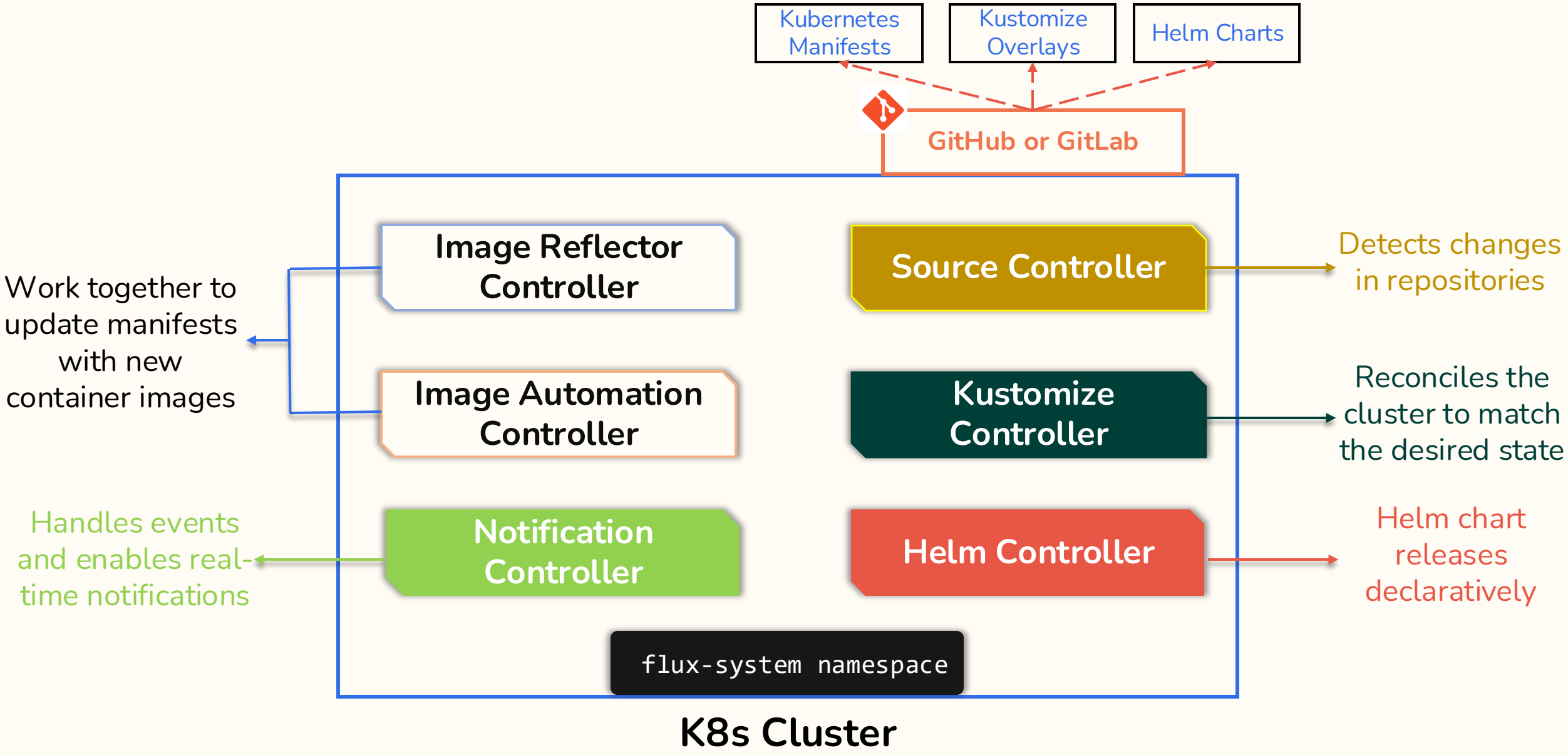




K8s Cluster







Section: 5
Working with Source & Kustomize controller

Working with Source and Kustomize Controller

Section Overview

- Getting started with Source Controller and Kustomize Controller
- Deploying First Application using Flux (using same Flux repository)
- suspend, resume and delete operations in Flux
- Source Controller and Kustomize Controller file creation
- Deploying application (Source, Kustomize file creation, kubernetes application manifests from external Repository)
- Integrating Private Repository
- Integrating Object Storage (s3) with Flux CD

Getting started with Source Controller and Kustomize Controller

Understanding Source Controller and Kustomize Controller

Single source of truth

GitLab

- Application Code
- Infrastructure Definitions
- Dockerfiles
- Kubernetes Manifests
- Helm Charts

Kustomize Controller

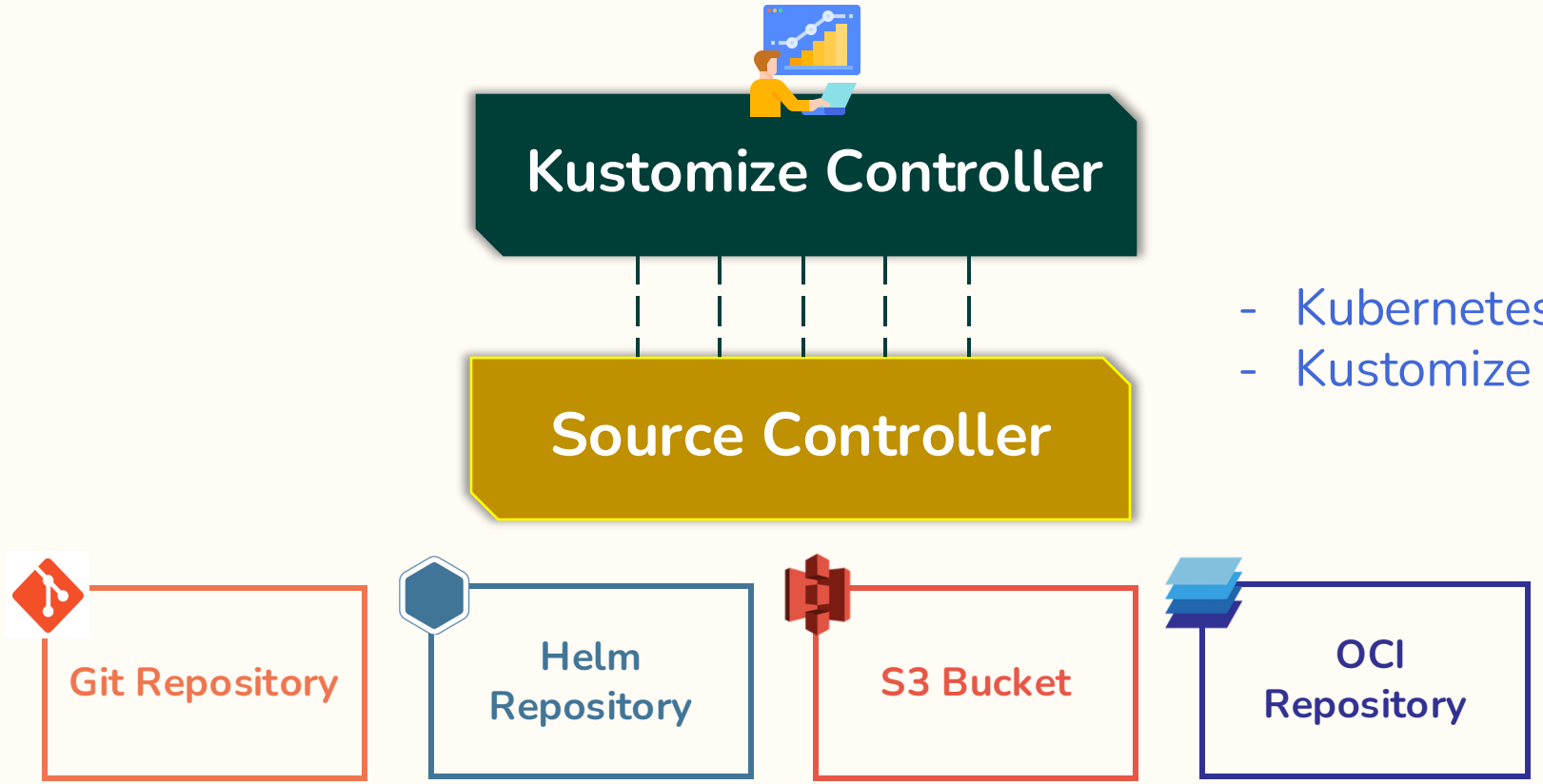
Source Controller

Git Repository

Helm Repository

S3 Bucket

OCI Repository



- Kubernetes manifest files
- Kustomize overlays

Understanding Source Controller and Kustomize Controller

- ✓ The ***Source Controller*** detects changes in the configured repositories
- ✓ The ***Kustomize Controller*** picks up those changes and reconciles the cluster to bring it back in sync with the desired state

Demo

Deploying First Application using Flux (using same Flux repository)

Demo

suspend, resume and delete operations in Flux

Source Controller and Kustomize Controller File Creation

Source Controller and Kustomize Controller File Creation

Imperative Approach

```
flux create source git demo2-source-git --
url=https://github.com/yogeshraheja/KubernetesVotingApp --branch=main
```

✓ Creates a **GitRepository** resource

Source Controller and Kustomize Controller File Creation

```
flux create kustomization demo2-kustomization-git --source=GitRepository/demo2-source-git --prune=true --target-namespace=devopsinaction --path=./
```

- ✓ No checks
- ✓ No traceability of changes
- ✓ Troubleshooting issues later can become time-consuming

Source Controller and Kustomize Controller File Creation

Declarative Approach

```
kubectl explain <CRD-Name>
```

You can manually write Flux resource manifests step by step — but that requires deeper familiarity with the **CRDs**

```
--export option
```

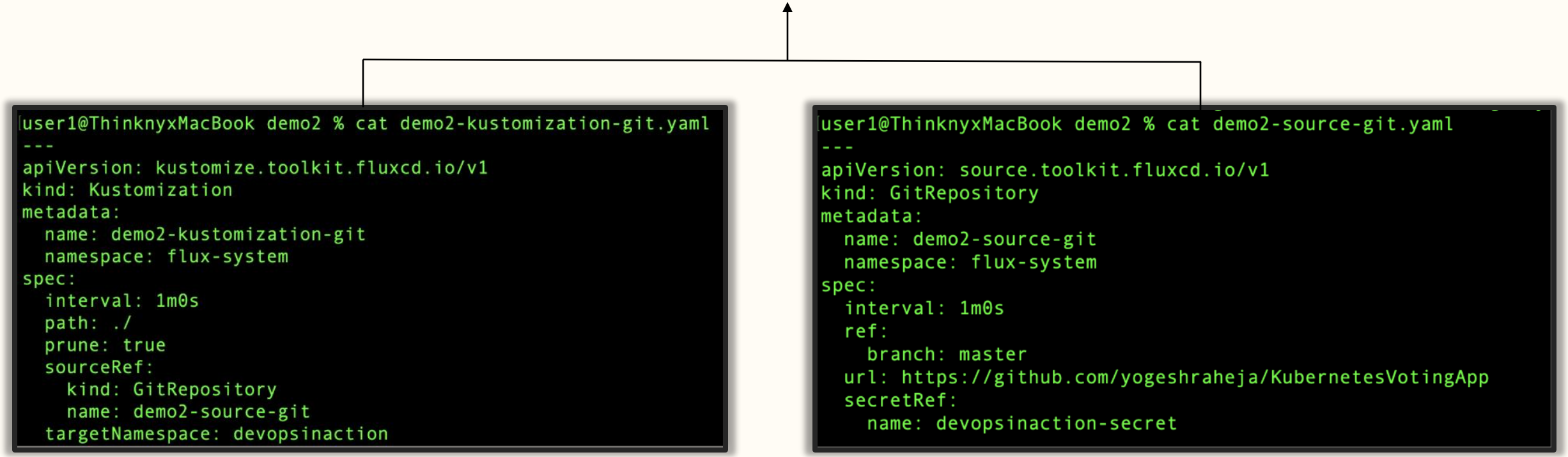
```
flux create source git demo2-source-git --
url=https://github.com/yogeshraheja/KubernetesVotingApp --branch=main --export >
demo2-source-git.yaml
```

```
user1@ThinknyxMacBook demo2 % cat demo2-source-git.yaml
---
apiVersion: source.toolkit.fluxcd.io/v1
kind: GitRepository
metadata:
  name: demo2-source-git
  namespace: flux-system
spec:
  interval: 1m0s
  ref:
    branch: master
  url: https://github.com/yogeshraheja/KubernetesVotingApp
  secretRef:
    name: devopsinaction-secret
```

```
flux create kustomization demo2-kustomization-git --source=GitRepository/demo2-source-git --prune=true --target-namespace=devopsinaction --path=./ --export > demo2-kustomization-git.yaml
```

```
user1@ThinknyxMacBook demo2 % cat demo2-kustomization-git.yaml
---
apiVersion: kustomize.toolkit.fluxcd.io/v1
kind: Kustomization
metadata:
  name: demo2-kustomization-git
  namespace: flux-system
spec:
  interval: 1m0s
  path: ./
  prune: true
  sourceRef:
    kind: GitRepository
    name: demo2-source-git
  targetNamespace: devopsinaction
```

Stored & Version-Controlled



```
user1@ThinknyxMacBook demo2 % cat demo2-kustomization-git.yaml
---
apiVersion: kustomize.toolkit.fluxcd.io/v1
kind: Kustomization
metadata:
  name: demo2-kustomization-git
  namespace: flux-system
spec:
  interval: 1m0s
  path: ./
  prune: true
  sourceRef:
    kind: GitRepository
    name: demo2-source-git
  targetNamespace: devopsinaction
```

```
user1@ThinknyxMacBook demo2 % cat demo2-source-git.yaml
---
apiVersion: source.toolkit.fluxcd.io/v1
kind: GitRepository
metadata:
  name: demo2-source-git
  namespace: flux-system
spec:
  interval: 1m0s
  ref:
    branch: master
  url: https://github.com/yogeshraheja/KubernetesVotingApp
  secretRef:
    name: devopsinaction-secret
```

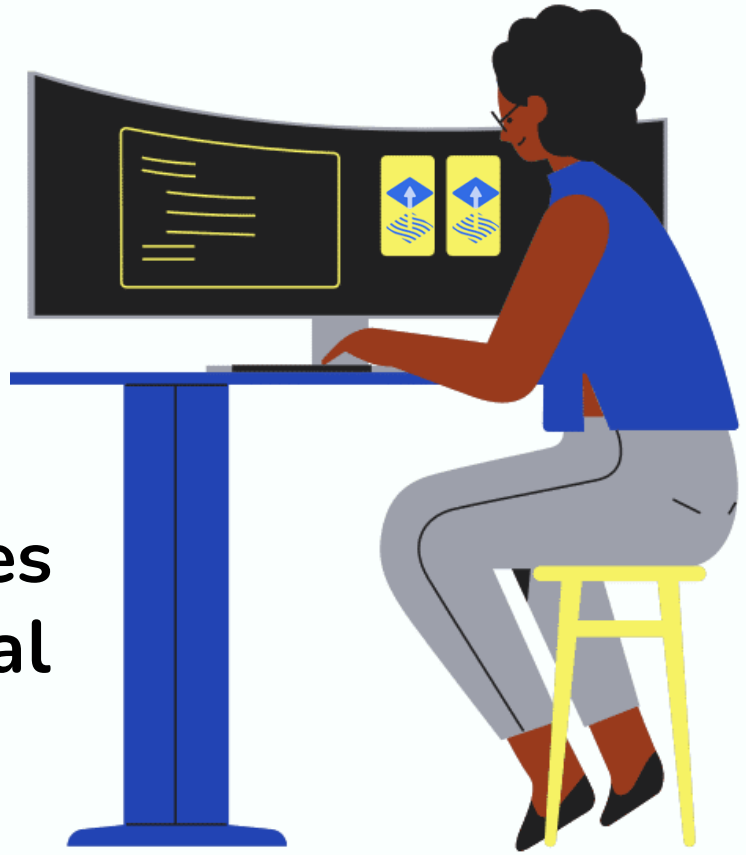
Fully Declarative

Auditable

Easier to Manage

Demo

Deploying application (Source, Kustomize file creation, kubernetes application manifests from external Repository)



Demo

Integrating Private Repository

Demo

Integrating Object Storage (s3) with Flux CD

Section: 6
Application Deployment using Kustomize and Helm

Application Deployment using Kustomize and Helm

Section Overview

- Application Deployment using Kustomize and Helm
- Deploying Application using Kustomize Overlay
- Understanding Helm Controller in Flux CD
- Deploying Application using Helm Charts (with Git as Source)
- Deploying Application from Helm Charts (with Helm Repository as Source)

Demo

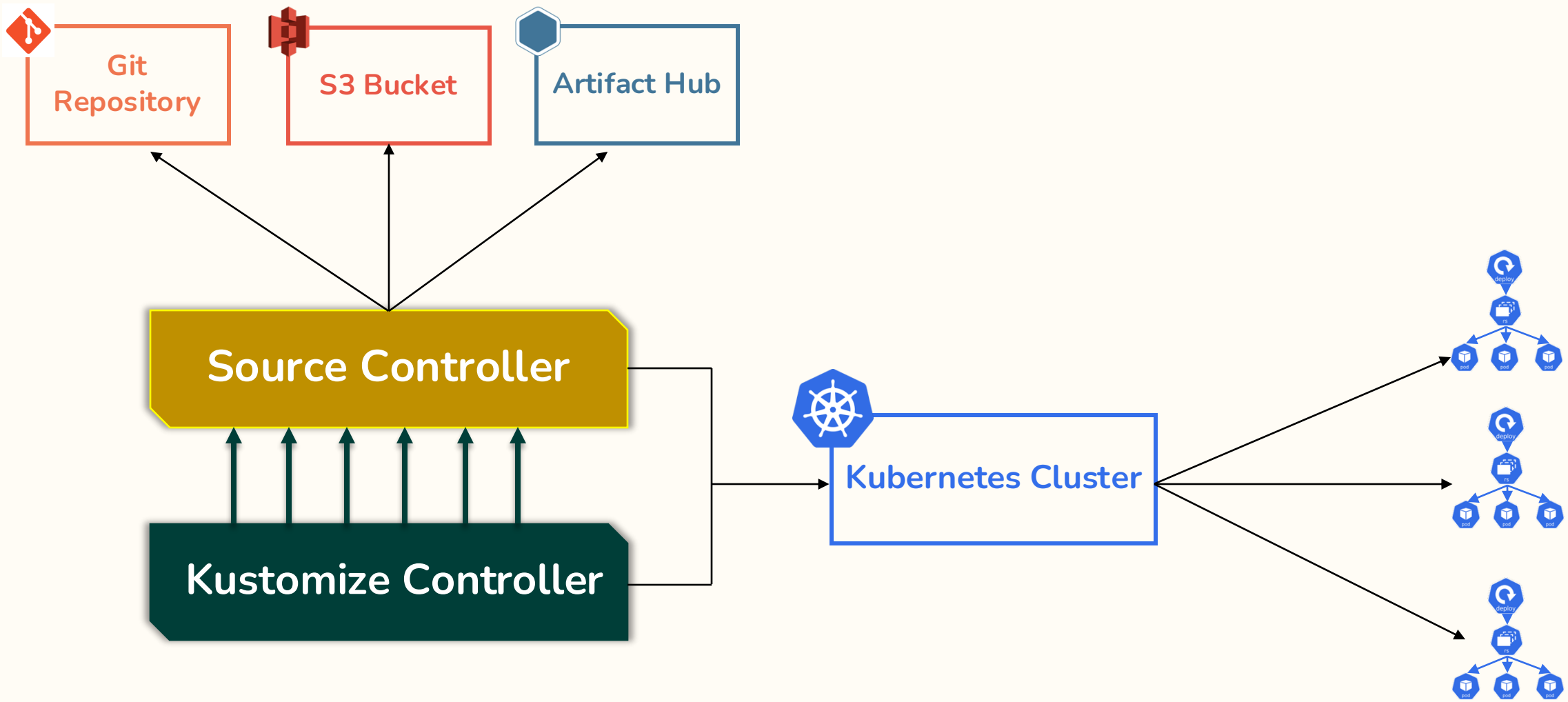
Application Deployment using Kustomize and Helm

Demo

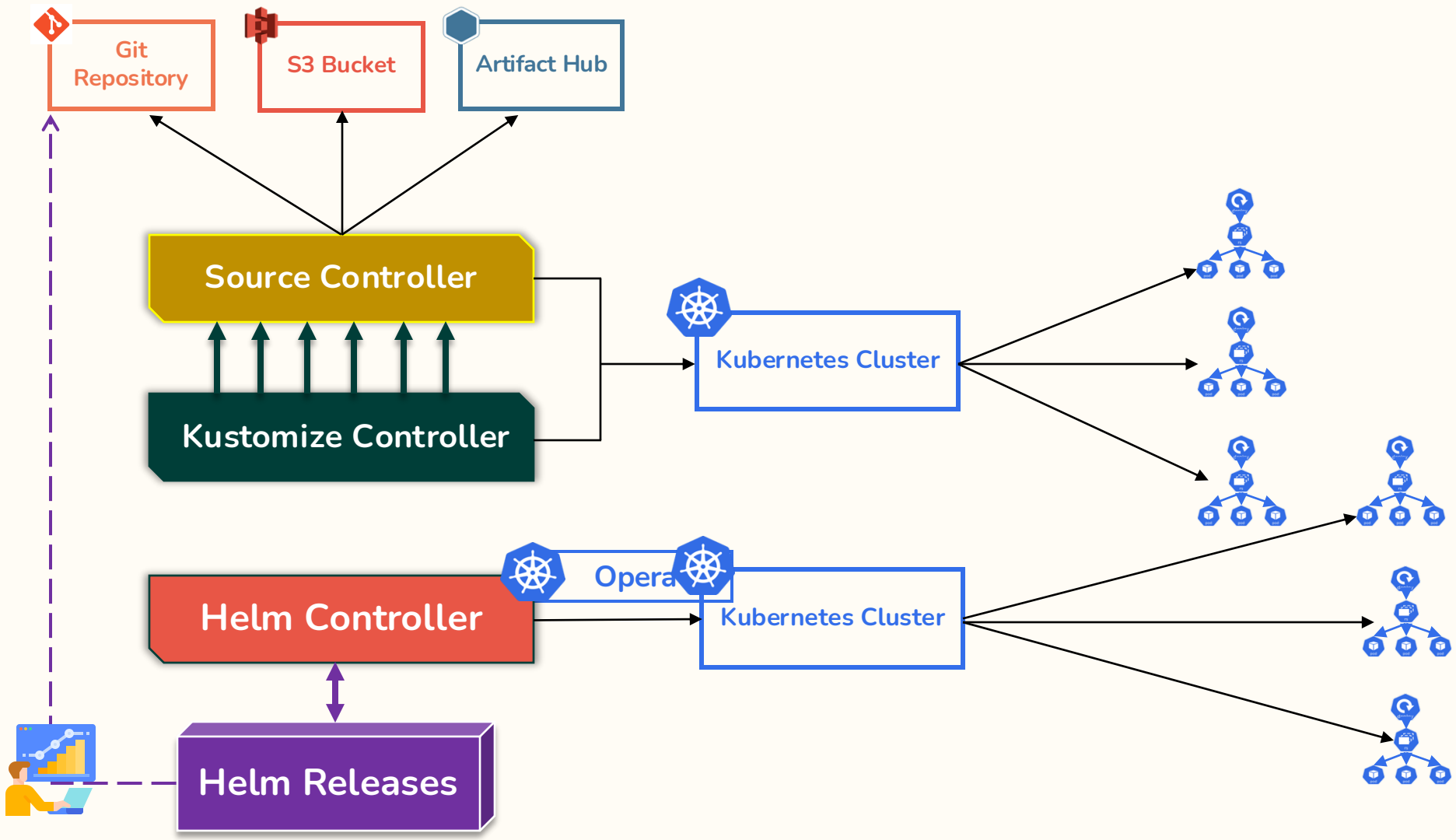
Deploying Application using Kustomize Overlay

Understanding Helm Controller in Flux CD

Understanding Helm Controller in Flux CD



Understanding Helm Controller in Flux CD



Understanding Helm Controller in Flux CD

Helm Releases

```
flux create hr --help
```

or

```
flux create helmrelease --help
```

Demo

Deploying Application using Helm Charts (with Git as Source)

Demo

Deploying Application from Helm Charts (with Helm Repository as Source)

Section: 7 Application Deployment from OCI Registry

Application Deployment from OCI Registry

Section Overview

- Understanding OCI Registry
- Pushing manifests and helm charts to OCI Registry
- Understanding OCI Repository Source Controller in Flux CD
- Deploying Application (Manifests based package/artifact) from OCI Registry
- Deploying Application (helm based package/artifact) from OCI Registry

Understanding OCI Registry

Understanding OCI Registry

CLOUD NATIVE COMPUTING FOUNDATION

Container Images	Helm Charts	Kubernetes Manifest Files
hub	Artifact Hub	Git Repository

Understanding OCI Registry

Challenges

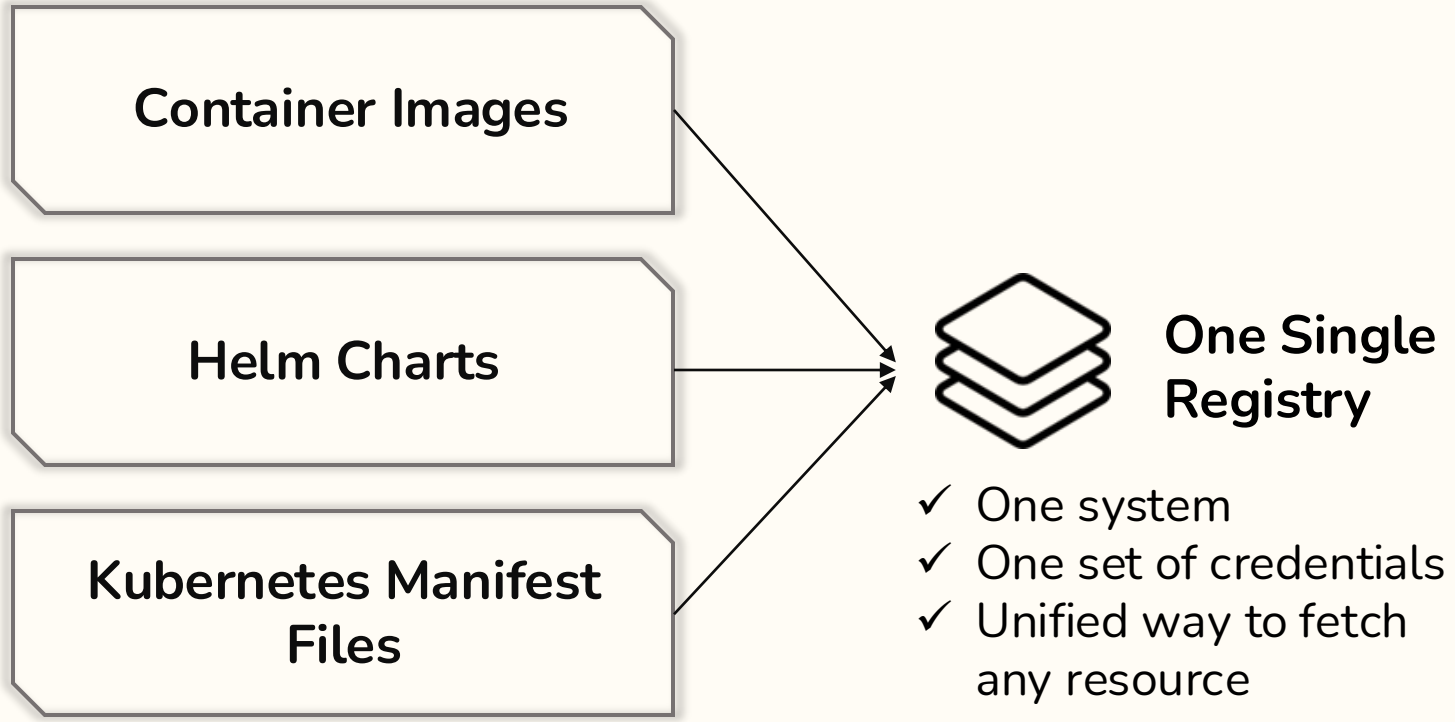
- Multiple URLs
- Multiple tools
- Different authentication and authorization mechanisms

Understanding OCI Registry

CLOUD NATIVE COMPUTING FOUNDATION

Challenges

- Multiple URLs
- Multiple tools
- Different authentication and authorization mechanisms



Understanding OCI Registry

OCI Artifacts & OCI Registries

Open Container Initiative

Open standards for container images, formats, & runtimes

Container images, but with OCI artifacts

Kubernetes manifests, Helm charts, and even Kustomize overlays

Understanding OCI Registry

ghcr.io

GitHub Container Registry

Converting **Kubernetes manifests** into OCI artifacts and pushing them to the registry

Doing the same with **Helm charts**

Understanding OCI Registry

Kubernetes Manifest Files

Docker CLI

```
flux push artifact
```

Helm Charts

HELM

Helm CLI

```
helm push
```

Demo

Pushing manifests and helm charts to OCI Registry

Understanding OCI Repository Source Controller in Flux CD

Understanding OCI Repository Source Controller in Flux CD

ghcr.io

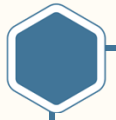
OCI Registry

Kubernetes Manifest Artifacts

Helm Chart Artifacts

Understanding OCI Repository Source Controller in Flux CD

GitRepository



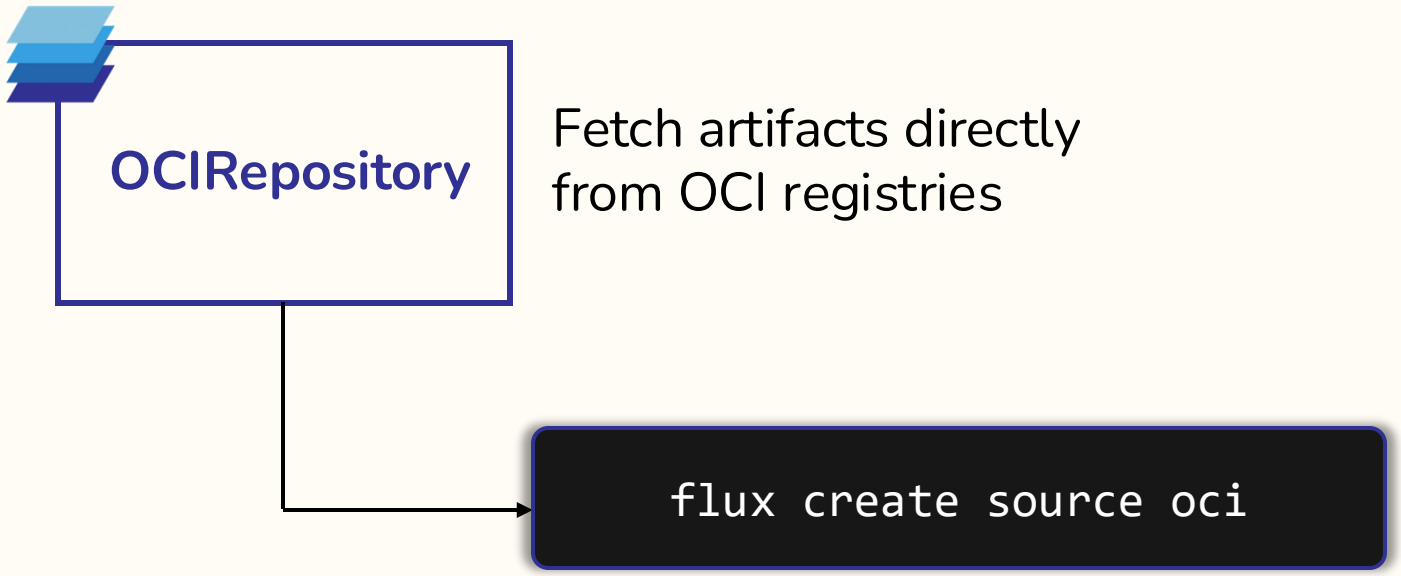
HelmRepository

Bucket

OCIRepository

Designed for
OCI-compliant registries

Understanding OCI Repository Source Controller in Flux CD



Understanding OCI Repository Source Controller in Flux CD

- ✓ The ***Kustomization resource*** to apply Kubernetes manifest artifacts
- ✓ The ***HelmRelease resource*** to apply Helm charts

Demo

Deploying Application (Manifests based package/artifact) from OCI Registry

Demo

Deploying Application (Helm based package) from OCI Registry

Section: 8
Image Reflector and Automation controllers in Flux CD

Application Deployment using Kustomize and Helm

Section Overview

- Understanding Flux CD Image Automation Controllers
- Installing Image Automation Controllers
- Flux CD Image Automation (ImageRepository, ImagePolicy, ImageUpdate Flux resources)

Understanding Flux CD Image Automation Controllers

Understanding Flux CD Image Automation Controllers

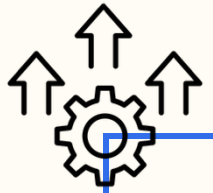
CI

Responsible for building container images

CD

Responsible for deploying the latest images into the environment

Understanding Flux CD Image Automation Controllers



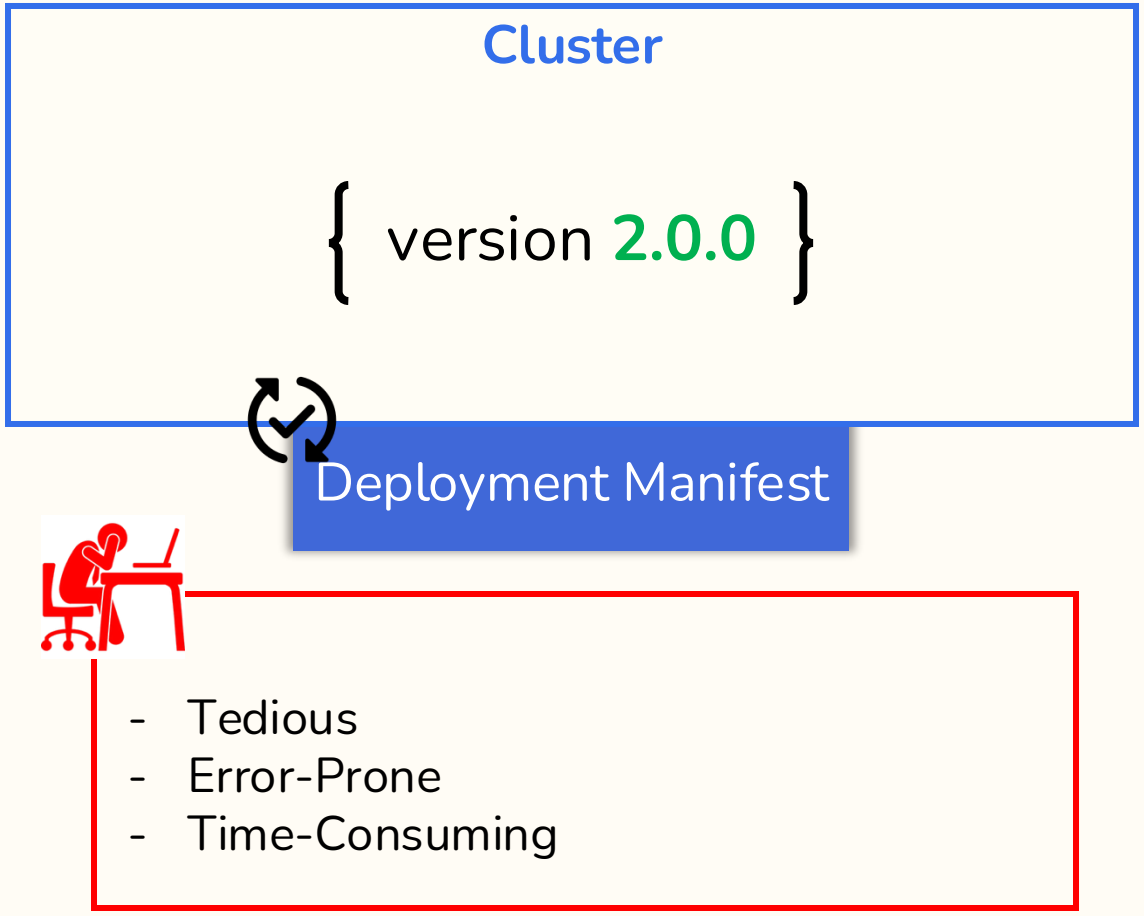
Container Registry

Cluster

{ version 1.0.0 }

- ✓ Add new features
- ✓ Merge their code
- ✓ Triggers the CI pipeline

Understanding Flux CD Image Automation Controllers



Understanding Flux CD Image Automation Controllers

Flux Image Automation

Image Reflector Controller

Image Automation Controller

Understanding Flux CD Image Automation Controllers

Image Reflector Controller

ImageRepository

Scans container registries and fetches image metadata, including tags and versions

ImagePolicy

Applies rules to filter and select the image versions discovered by the ImageRepository

Understanding Flux CD Image Automation Controllers

Image Automation Controller

ImageUpdateAutomation

- ✓ Clones the Git repository
- ✓ Updates the YAML manifests with the latest image version
- ✓ Commits and pushes the changes back to the repository

But how does it know which field in the YAML needs to be updated?

But how does it know which field in the YAML needs to be updated?

```
# {"$imagepolicy": "flux-system:demo9-image-policy"}
```

```
user1@ThinknyxMacBook demo9 % cat manifests/02_demoapp_dep.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: demoapp
  name: demoapp-dep
  namespace: demo9-ns
spec:
  replicas: 3
  selector:
    matchLabels:
      app: demoapp
  template:
    metadata:
      labels:
        app: demoapp
    spec:
      containers:
      - image: yogeshraheja/fluxdemo:1.0.0 # {"$imagepolicy": "flux-system:demo9-image-policy"}
        name: demoapp
```

- ✓ The ***Source Controller*** detects the change
- ✓ The ***Kustomize Controller*** then applies the updated manifest to the cluster

Demo

Installing Image Automation Controllers

Demo

Flux CD Image Automation (ImageRepository, ImagePolicy, ImageUpdate Flux resources)

Section: 9
Notification Controller in Flux CD

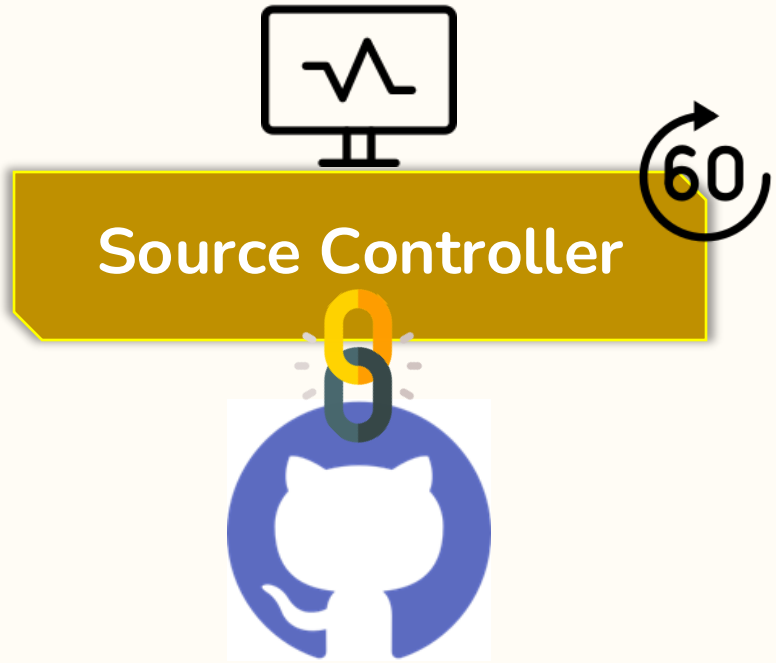
Notification Controller in Flux CD

Section Overview

- Understanding Flux CD Notification Controllers
- Webhooks with Flux CD (Receiver Flux resource)
- Integrating Flux CD with MS Teams for notifications (Providers and Alerts Flux resources)

Understanding Flux CD Notification Controllers

Notification Controller in Flux CD

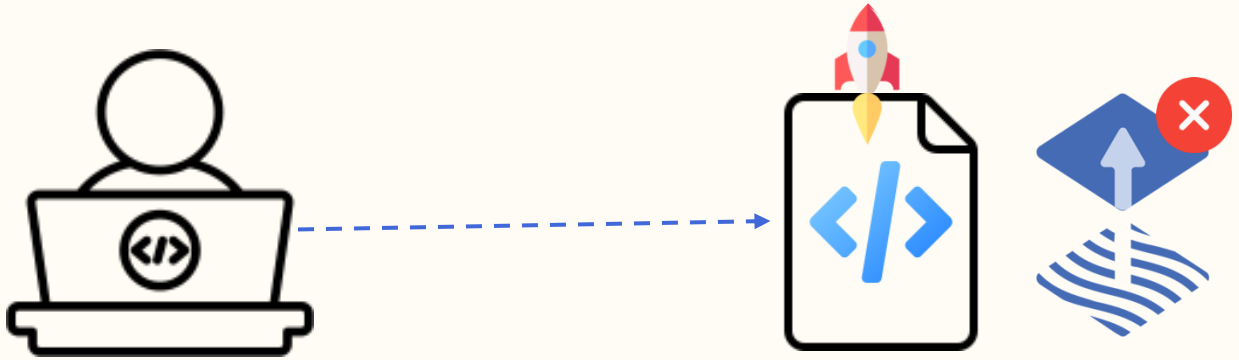


Notification Controller in Flux CD

Flux CD

Pull-Based Approach

But what if you don’t want to rely on polling every 60 seconds, and instead, you prefer a **push-based approach?**



Version Control System

Notification Controller

Event Forwarder

Notifies other Flux components to reconcile as soon as changes are detected in your repositories

Developers

Notification Dispatcher

Collects events from GitOps Toolkit controllers

On-Call Engineers

Source

Kustomize

Helm

Slack

Teams

Discord

Rocket Chat

Demo

Webhooks with Flux CD (Receiver Flux resource)

Demo

Integrating Flux CD with MS Teams for notifications (Providers and Alerts Flux resources)

Section: 10
Flux CD Monitoring and Dashboarding

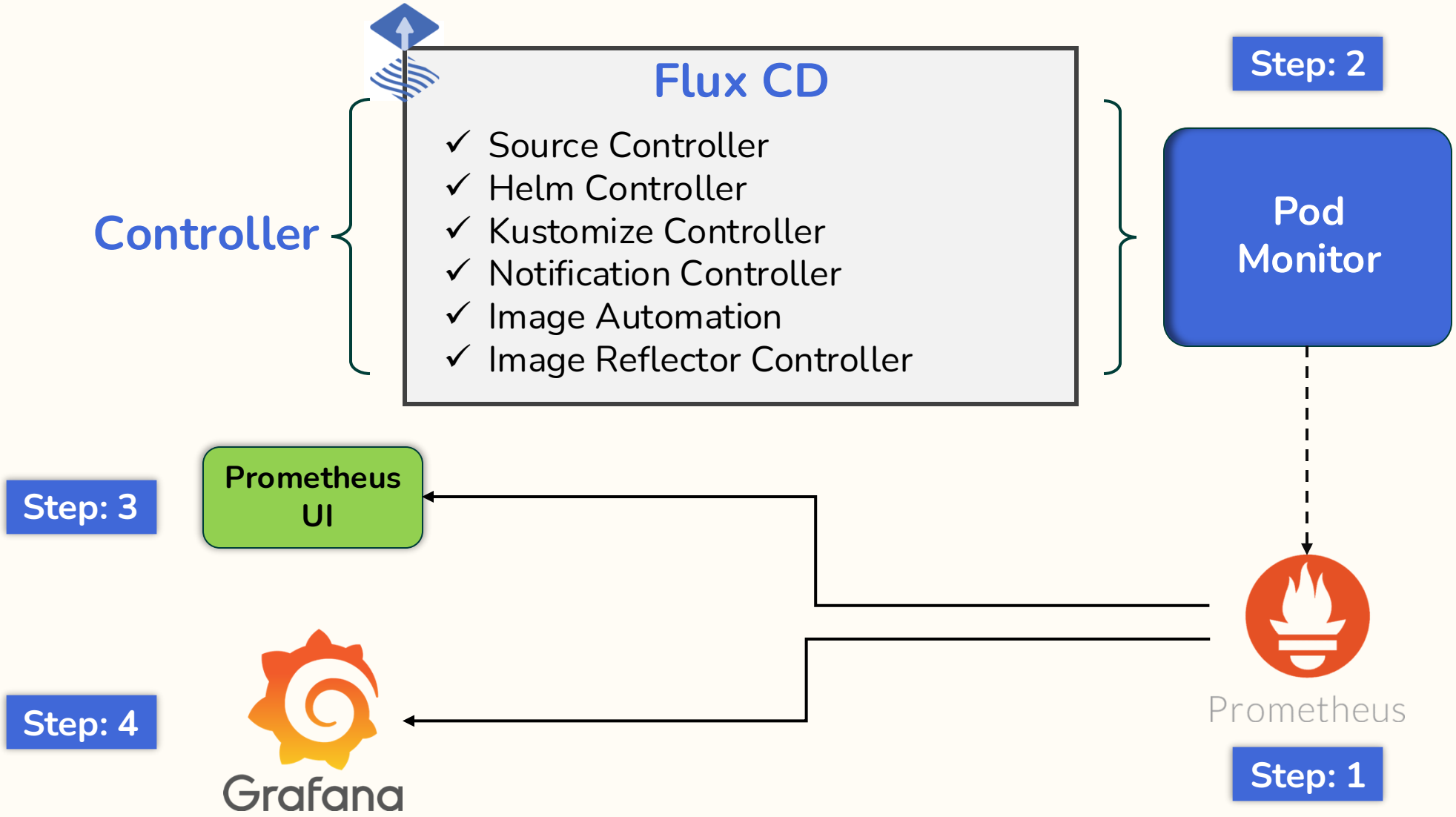
Flux CD Monitoring & Dashboarding

Section Overview

- Flux CD Monitoring and Dashboarding
- Setting up Prometheus and Grafana using Flux CD
- Monitoring Flux CD with Prometheus and Grafana

Flux CD Monitoring & Dashboarding

Flux CD Monitoring



Demo

Setting up Prometheus and Grafana using Flux CD

Demo

Monitoring Flux CD with Prometheus and Grafana

Section: 11
Flux CD UI

Flux CD UI

Section Overview

- Flux CD UI Options
- Demonstration - Flux UI with Capacitor
- Flux UI with weave-gitops

Flux CD UI Options

Flux CD UI Options

Argo CD

Flux CD

- ✓ Open-source
- ✓ Free project

Flux CD UI Options

- Weave GitOps Project
- VS Code GitOps Tools
- Freelens FluxCD Extension
- Capacitor

https://fluxcd.io/ecosystem/

Flux CD UI Options

https://fluxcd.io/ecosystem/

- Weave GitOps Project
- VS Code GitOps Tools
- Freelens FluxCD Extension
- Capacitor

Flux UIs / GUIs

These open source projects offer a dedicated graphical user interface for Flux.

Source	Description
gimlet-io/capacitor	Capacitor is a general purpose Flux UI to debug Flux and application issues. Apply these manifests to install it.
headlamp/flux-plugin	Headlamp is a Kubernetes UI extensible through plugins. The Flux plugin for Headlamp allows to visualize Flux and perform operations common operations like quick sync'ing, suspending/resuming, and others. See the plugin README for installation instructions.
freelensapp/freelens-fluxcd-extension	Freelens is a free and open-source user interface designed for managing Kubernetes clusters. It provides a standalone application compatible with macOS, Windows, and Linux operating systems. This extension visualizes Flux resources and allows to perform operations on them.
vmware-tanzu/kubeapps	Kubeapps is an in-cluster web-based application that enables users with a one-time installation to deploy, manage, and upgrade applications on a Kubernetes cluster. Read the documentation for managing Flux packages using Kubeapps.
weaveworks/vscode-gitops-tools	GitOps Tools for Visual Studio Code: provides an intuitive way to manage, troubleshoot and operate your Kubernetes environment following the GitOps operating model
weaveworks/weave-gitops	Weaveworks offered a free and open source GUI for Flux under the weave-gitops project. You can install the Weave GitOps UI using a Flux `HelmRelease`, please see the get started documentation for more details.

Flux CD UI Options

Flux UIs / GUIs

These open source projects offer a dedicated graphical user interface for Flux.

Source	Description
gimlet-io/capacitor	Capacitor is a general purpose Flux UI to debug Flux and application issues. Apply these manifests to install it.
headlamp/flux-plugin	Headlamp is a Kubernetes UI extensible through plugins. The Flux plugin for Headlamp allows to visualize Flux and perform operations common operations like quick sync'ing, suspending/resuming, and others. See the plugin README for installation instructions.
freelensapp/freelens-fluxcd-extension	Freelens is a free and open-source user interface designed for managing Kubernetes clusters. It provides a standalone application compatible with macOS, Windows, and Linux operating systems. This extension visualizes Flux resources and allows to perform operations on them.
vmware-tanzu/kubeapps	Kubeapps is an in-cluster web-based application that enables users with a one-time installation to deploy, manage, and upgrade applications on a Kubernetes cluster. Read the documentation for managing Flux packages using Kubeapps.
weaveworks/vscode-gitops-tools	GitOps Tools for Visual Studio Code: provides an intuitive way to manage, troubleshoot and operate your Kubernetes environment following the GitOps operating model
weaveworks/weave-gitops	Weaveworks offered a free and open source GUI for Flux under the weave-gitops project. You can install the Weave GitOps UI using a Flux `HelmRelease`, please see the get started documentation for more details.

- Weave GitOps Project
- VS Code GitOps Tools
- Freelens FluxCD Extension
- Capacitor

- ✓ Helps application operators quickly discover and resolve issues
- ✓ Offer an intuitive interface that provides a guided experience
- ✓ Making it easier for new users to get started

Demo

Flux UI with Capacitor

Demo

Flux UI with weave-gitops

Section: 12
More Concepts (Security Considerations)

More Concepts (Security Considerations)

Section Overview

- More Concepts (Security Considerations)

Demo

Security Practices for Flux CD

Section: 13
Conclusion

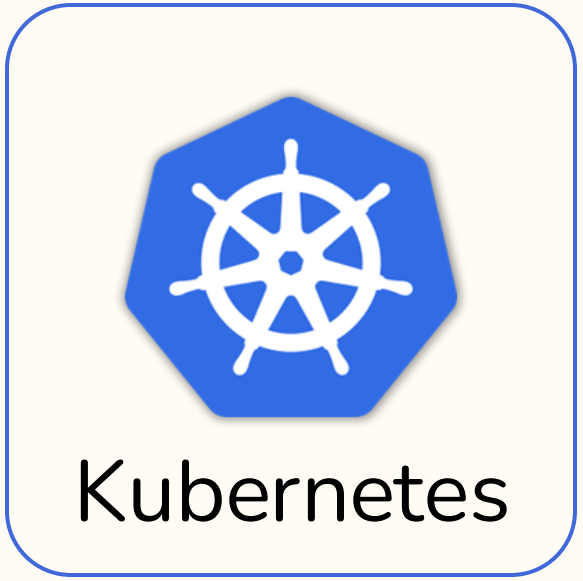
Demo

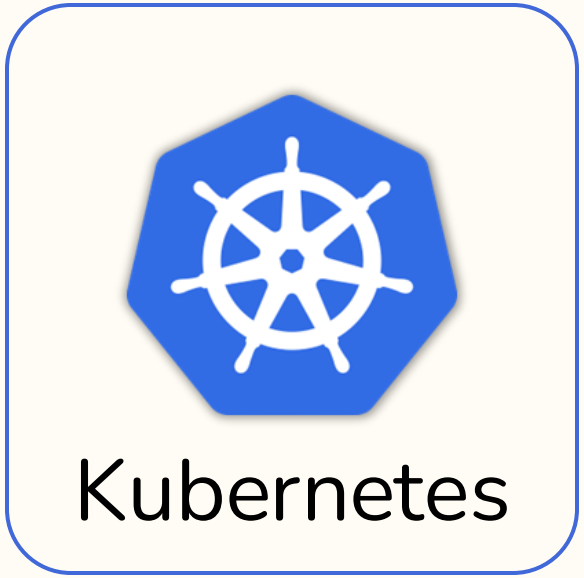
Tips using kubectl command for Flux CD resources

Conclusion

Section Overview

- Deep understanding of Flux CD
  - ✓ Deploy
  - ✓ Manage
  - ✓ Troubleshoot Flux CD
- Implement sync configurations
  - ✓ Source
  - ✓ Helm
  - ✓ Kustomize
  - ✓ Image Automation
  - ✓ Notification Controllers





Ever-evolving platform, and continuous learning is key to staying ahead