OpenShift GitOps and ArgoCD at the Edge

Red Hat WI/MN User Groups

Ryan Etten
Architect, Container Infrastructure
Hybrid Cloud Adoption Squad, Red Hat

Red Hat

Who Am I?

- Member of the Red Hat Hybrid Cloud Adoption Squad (HCAS)
- Specialize in designing and implementing automated container platforms, deployment pipelines, and cloud-ready application architectures
- Advocate for technology that enables us to be best, most agile versions of ourselves, doing more together

Container Infrastructure Architect with Red Hat Consulting

Passionate about open-source, digital transformation strategies, photography, and music

What we'll discuss today

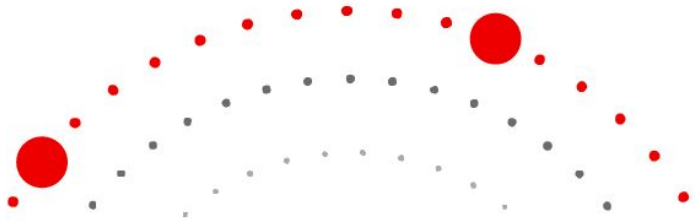
- Introduction
- Evolution of Automation
- GitOps Maturity Model
- OpenShift GitOps
- ArgoCD Architecture
- App of Apps Pattern
- Edge Cluster Scenario
- Edge Deployment Strategies
- ApplicationSets for Edge
- Securing ArgoCD

Introduction

Automation at the Edge

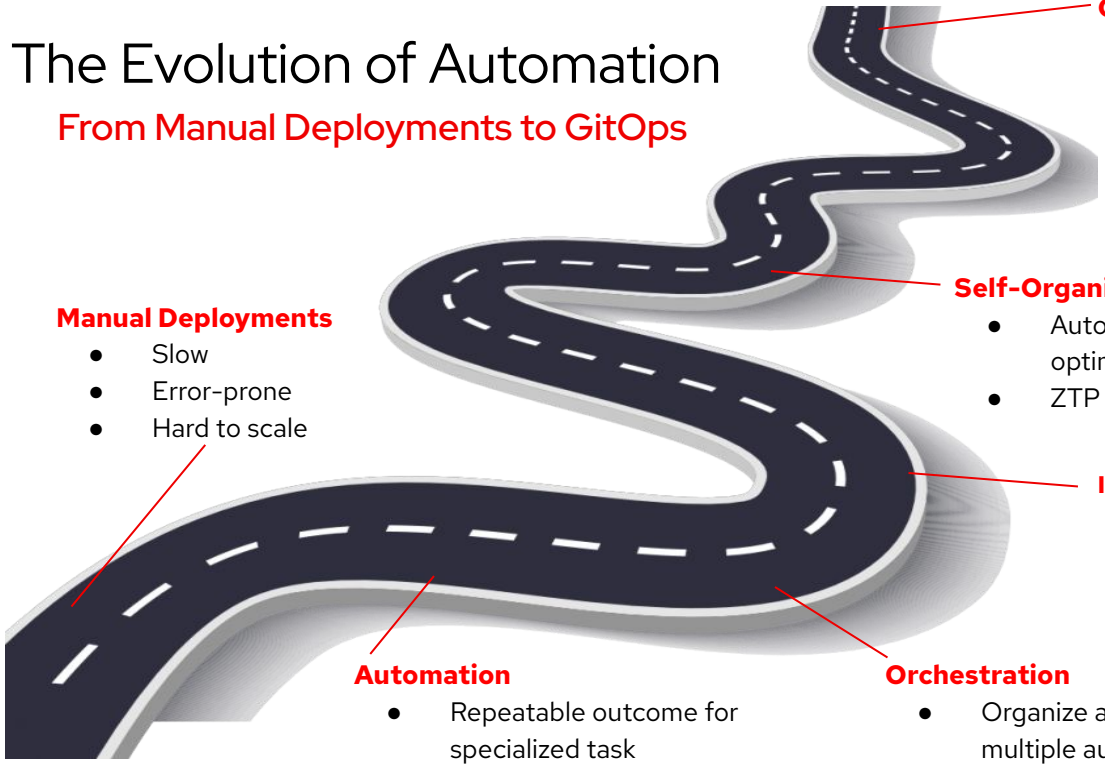
Organizations are doing more at the edge of the network, resulting in the need for consistent management and interoperability to reduce complexity

- OpenShift GitOps with ArgoCD
  - Helping organizations to build, deploy, scale, and manage end-to-end automation at the edge



The Evolution of Automation

From Manual Deployments to GitOps



Manual Deployments

- Slow
- Error-prone
- Hard to scale

Automation

- Repeatable outcome for specialized task
- No human intervention

Orchestration

- Organize and coordinate the execution of multiple automation tasks to define a workflow

Infrastructure as Code (IaC)

- Manage infrastructure with declarative model
- Follows DevOps practices
- ZTP for infrastructure

Self-Organizing Networks (SON)

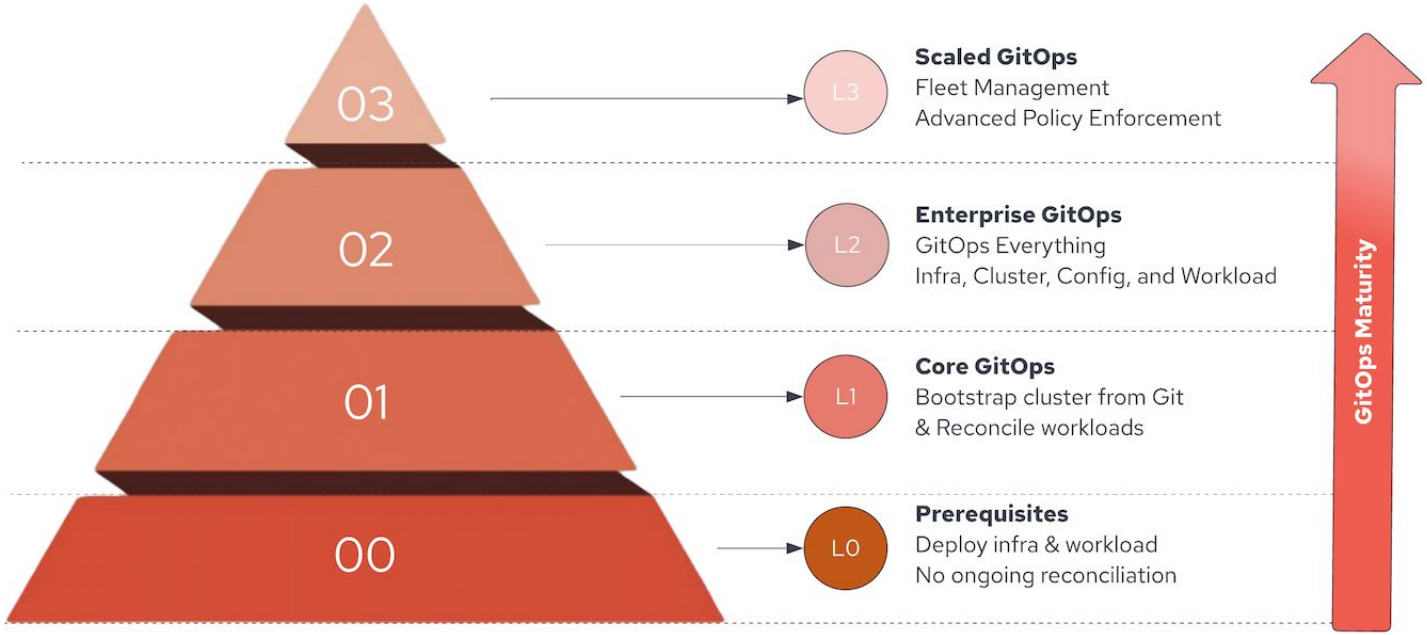
- Automation for configuration, optimization and healing of RAN
- ZTP for RU

GitOps

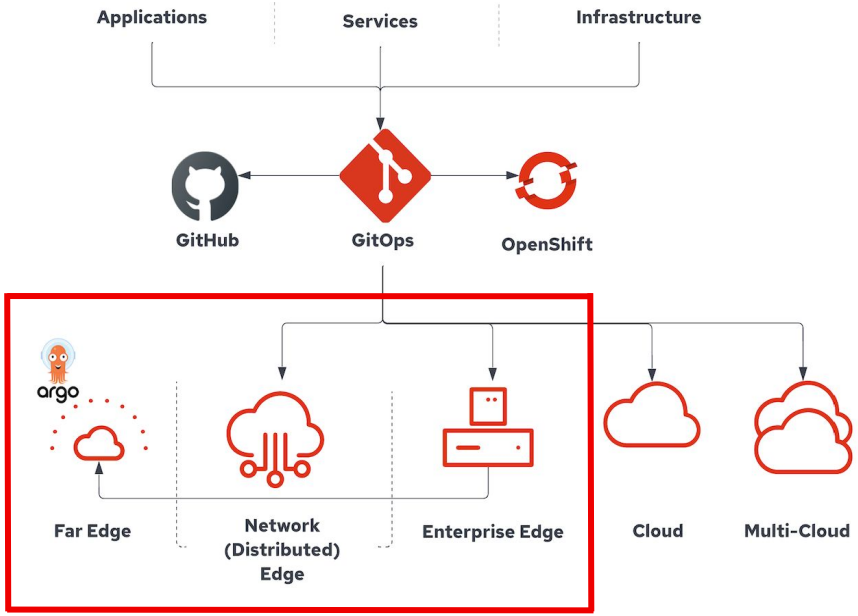
- Continuous reconciliation (self-healing)
- Declarative operations
- End-to-end zero-touch provisioning (ZTP)

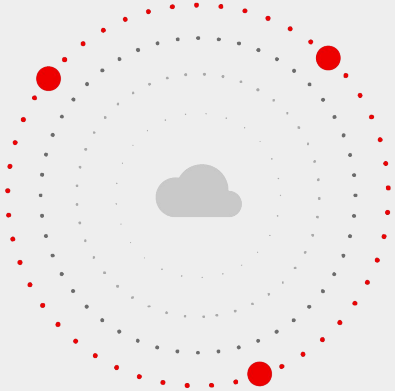
GitOps Maturity Model

Summary



GitOps in Edge Computing



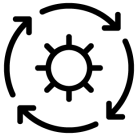


GitOps Principles

Defined in 2021

The system is described declaratively

The desired state is versioned in Git



Approved changes can be applied automatically

A controller exists to detect and act on drift

OpenShift GitOps

Powered by ArgoCD

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

Deployments and environments insights

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

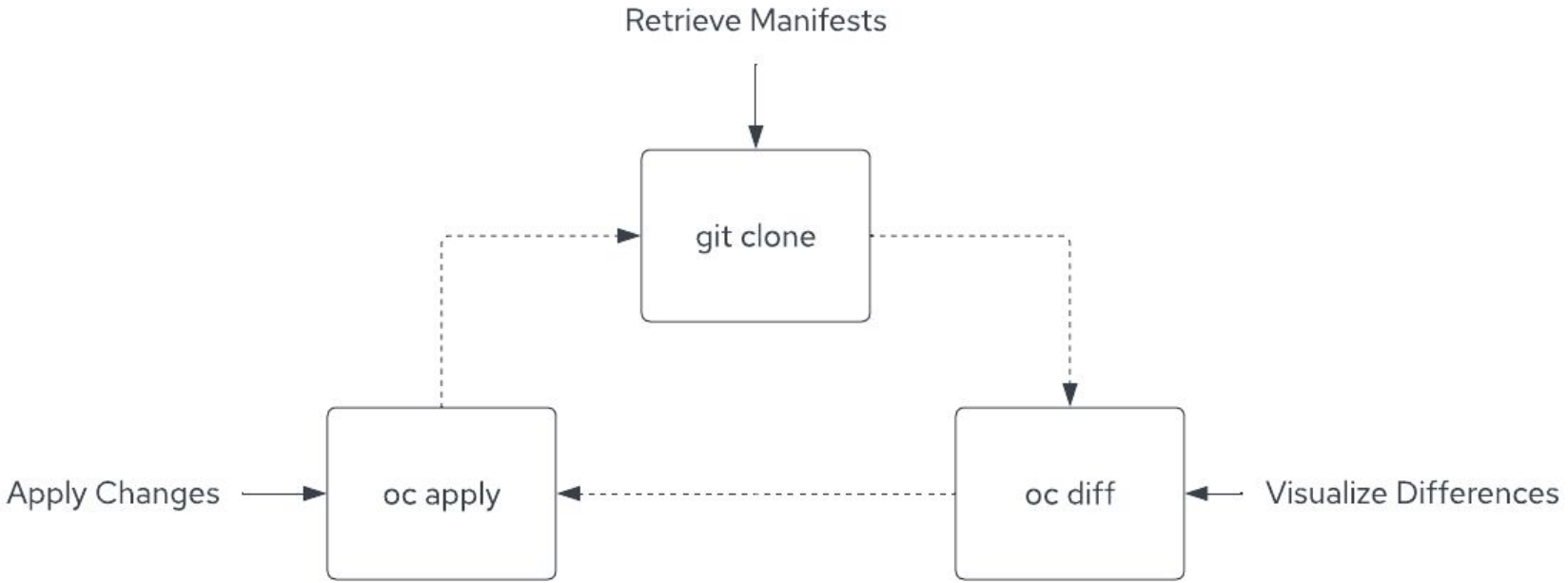
Understanding ArgoCD

OpenShift GitOps Made Simple

Argo CD offers the following key features and capabilities:

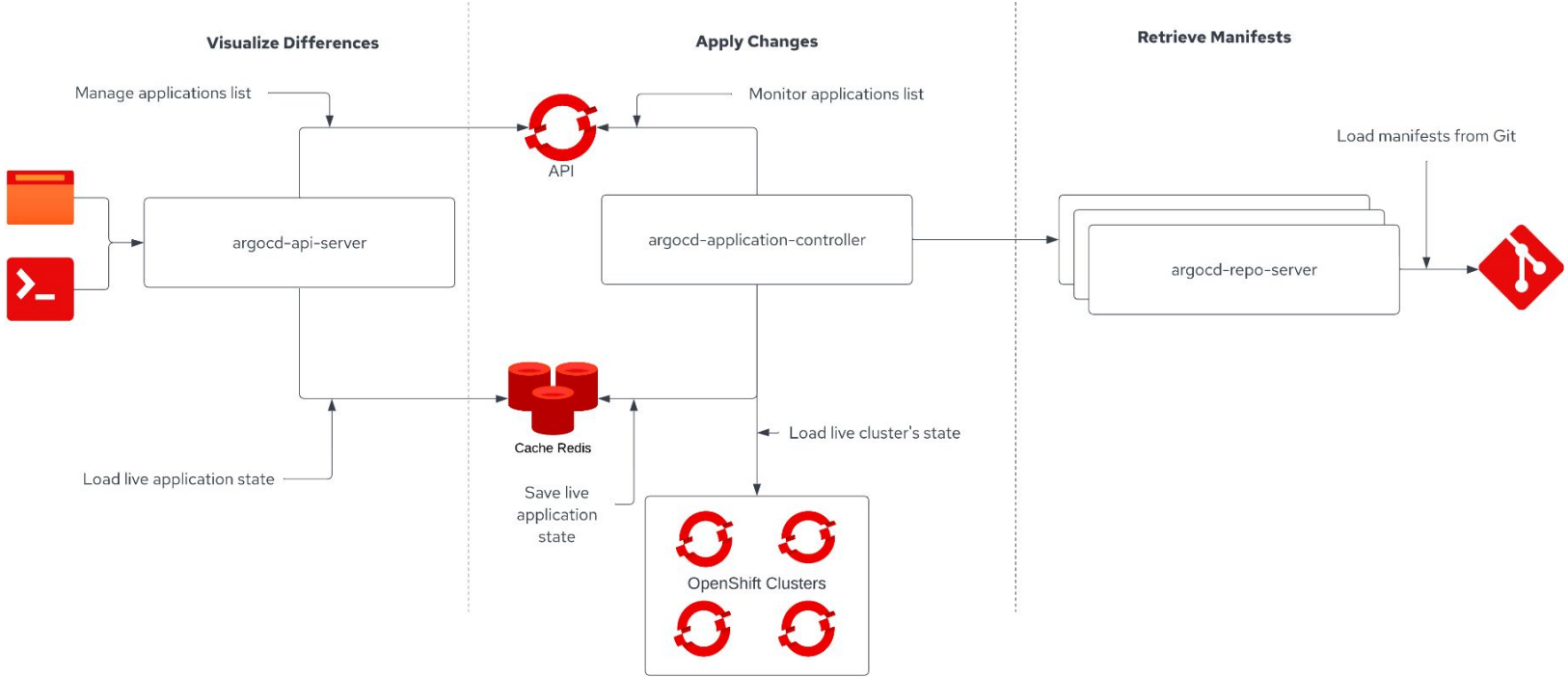
- Manual or automatic deployment of applications to OpenShift clusters
- Automatic sync of application state to the current version of declarative configuration
- Web user interface and command-line interface (CLI)
- Ability to visualize deployment issues, detect and remediate configuration drift
- Role-based access control (RBAC) enabling multi-cluster management
- Single sign-on (SSO) with providers such as GitLab, GitHub, Microsoft, OAuth2, and OIDC
- Support for webhooks triggering actions in GitLab, GitHub, and BitBucket

GitOps Operator Functions

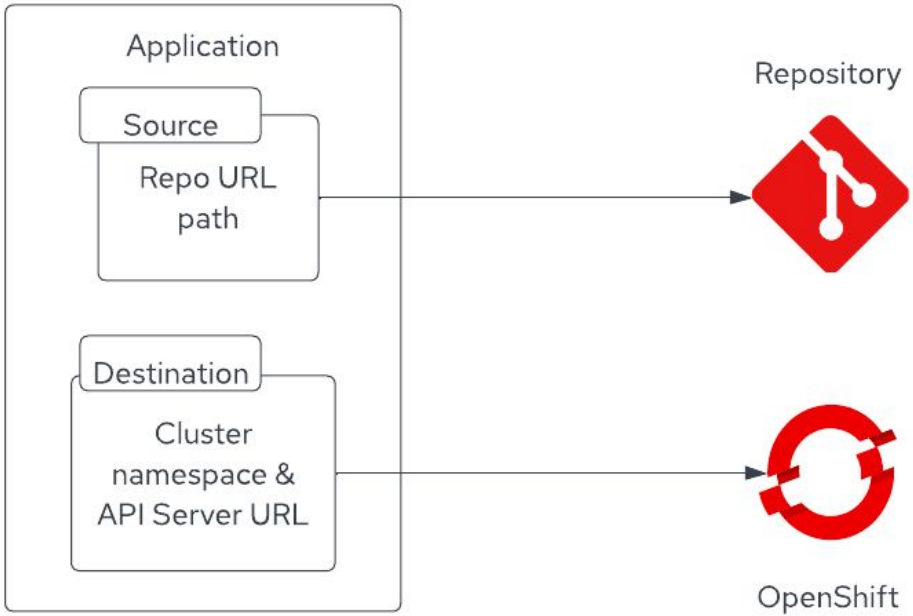


Red Hat

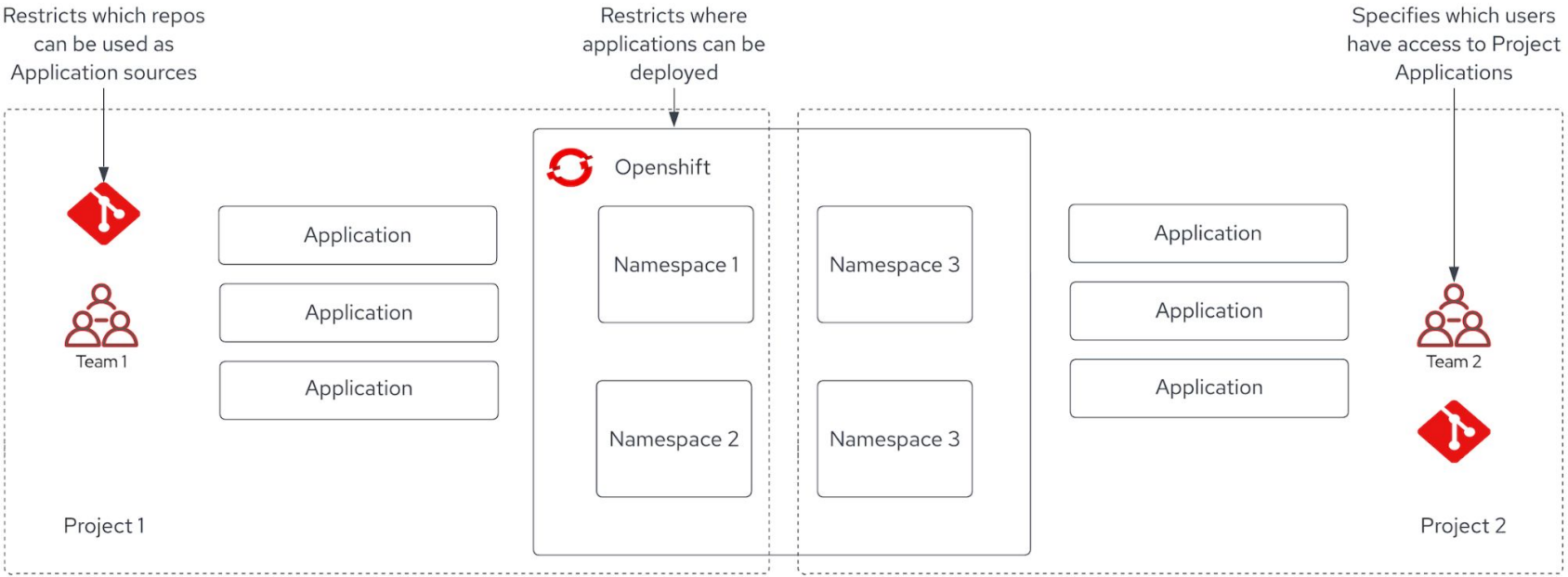
ArgoCD Architecture



ArgoCD Applications



ArgoCD Projects

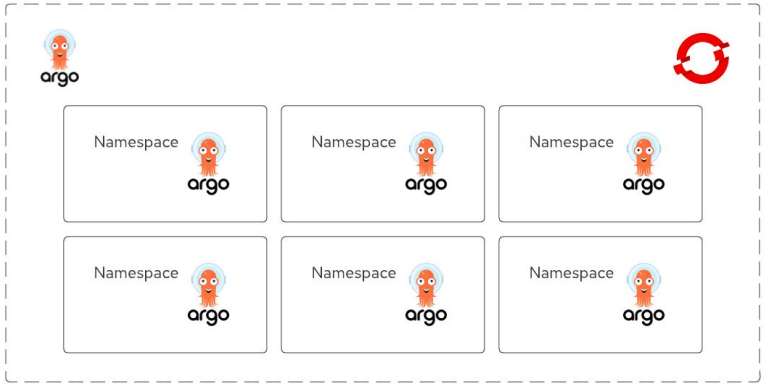


ArgoCD App of Apps Pattern

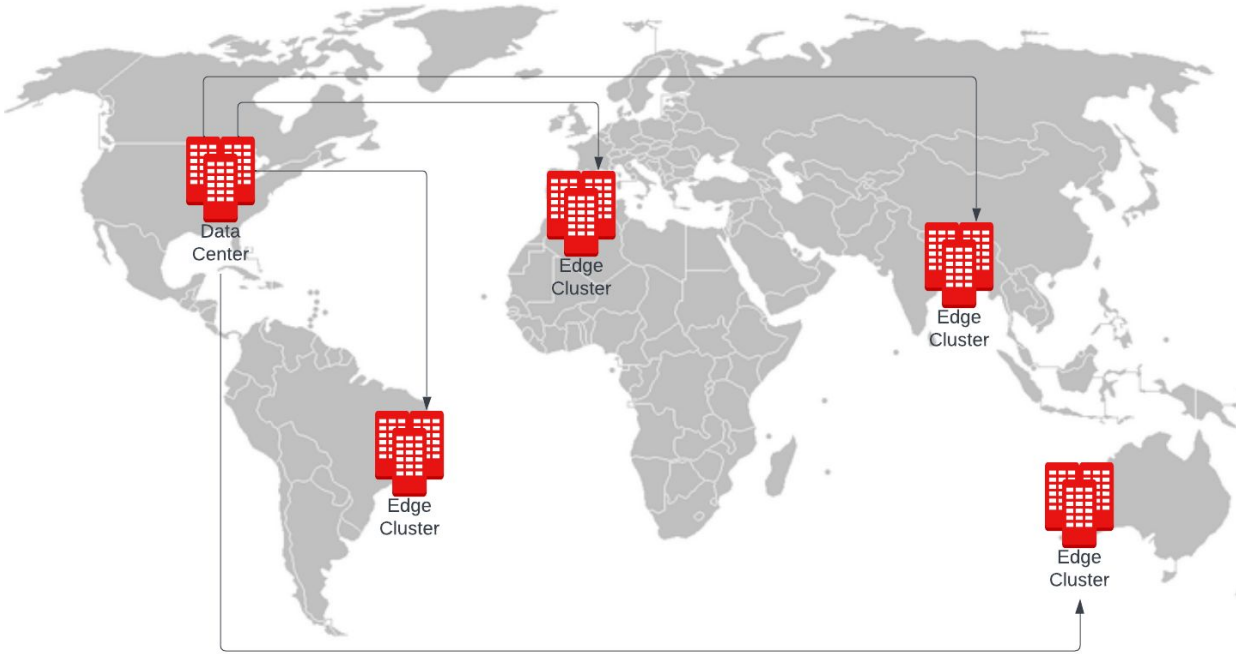
Precursor to ApplicationSets

- What are the challenges?
- What problems does the App of Apps pattern solve?
- When is it best to use App of Apps?
- How do we use the App of Apps pattern?
- How does App of Apps enable GitOps?

Let's look at an example...



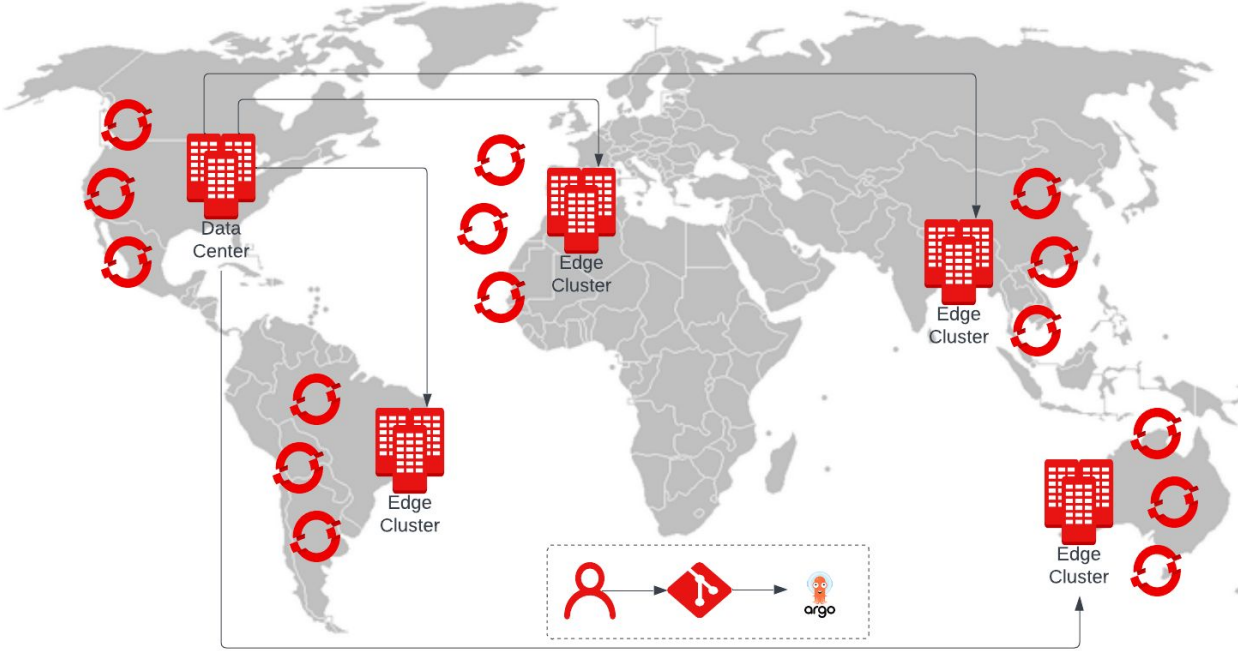
Global Edge Cluster Scenario



Overview

- Hundreds of global edge clusters
- Dozens of nodes on each
- Edge services
- Data Center
  - Management Control Plane
  - Service Data Plane

Global Edge Cluster



Overview

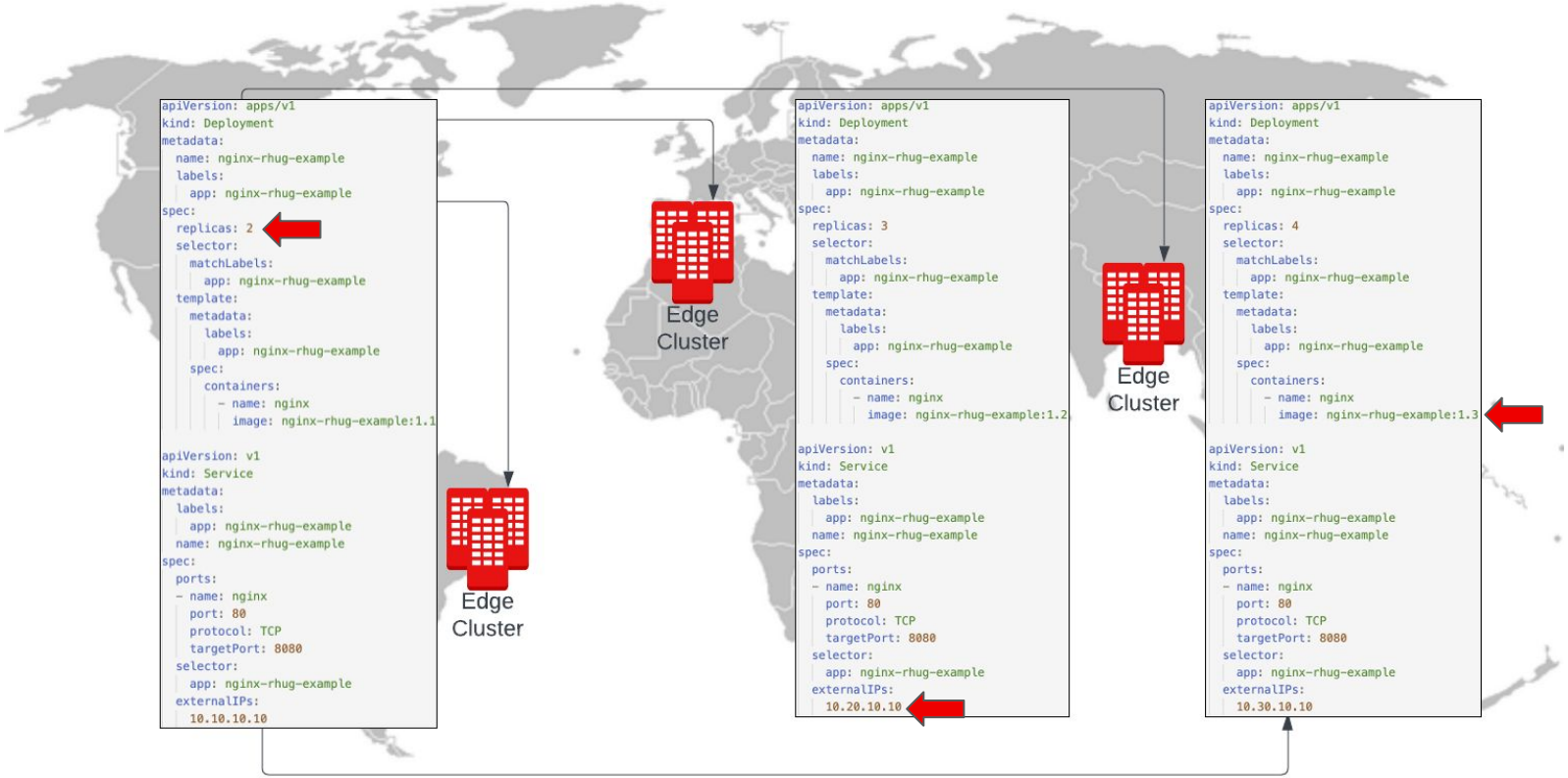
- Each edge is an OpenShift cluster
- GitOps from Data Center
- Developers push configuration to Git
- Central ArgoCD controller pulls the config from Git and Syncs to Edge OpenShift clusters

Global Edge Deployment Pattern

Overview

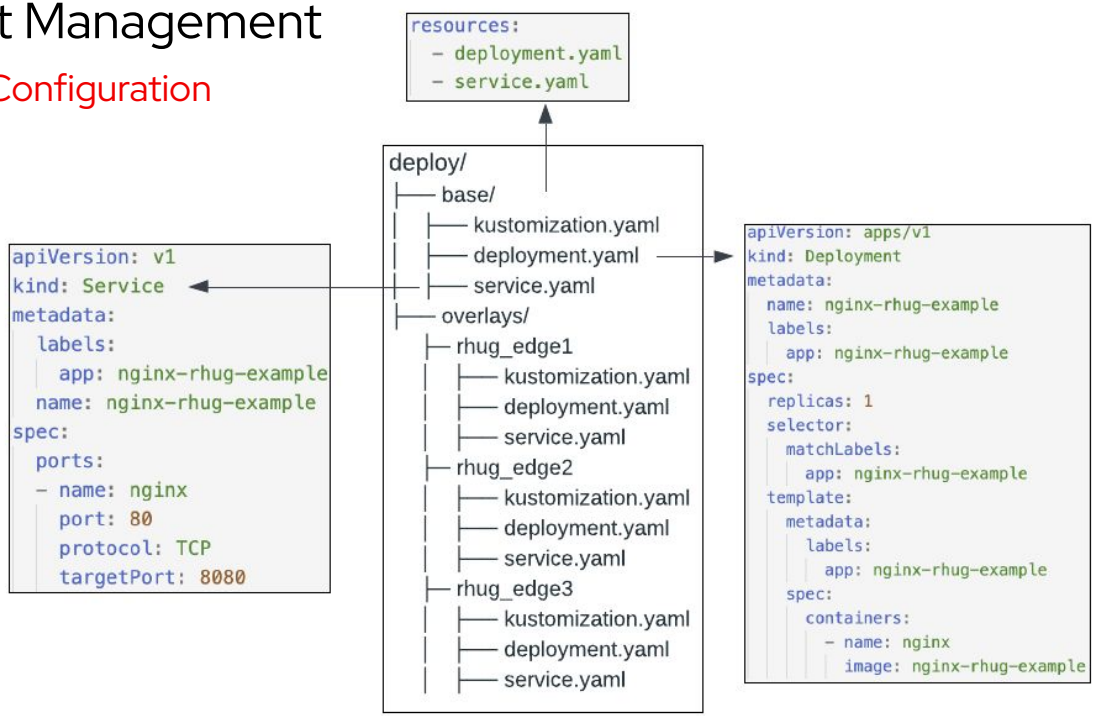
- An edge service is deployed to many edge clusters
- The service functionality and behavior are similar on all edge clusters
- The resulting OpenShift configuration pattern of an edge service:
  - Share large common configurations
  - Have cluster-specific configurations
    - Replica count, image tag, and external IPs

Edge Deployment Pattern - Example



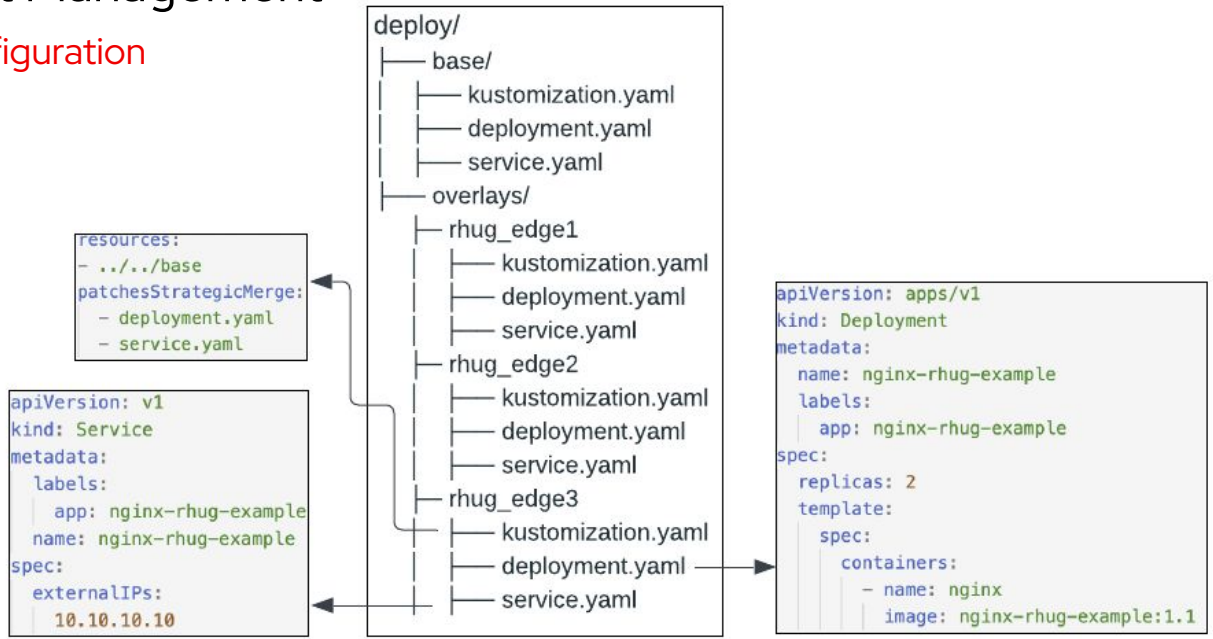
Edge Deployment Management

Kustomize Base Configuration



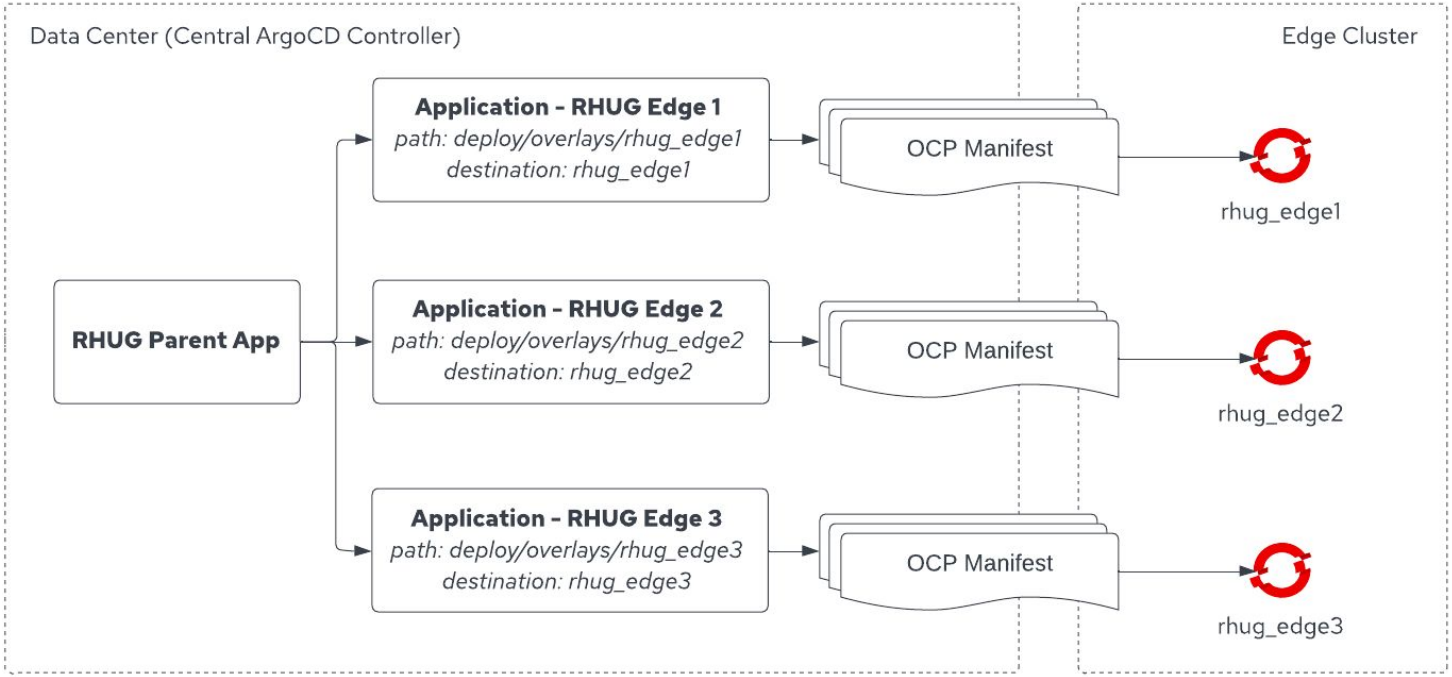
Edge Deployment Management

Overlay Configuration

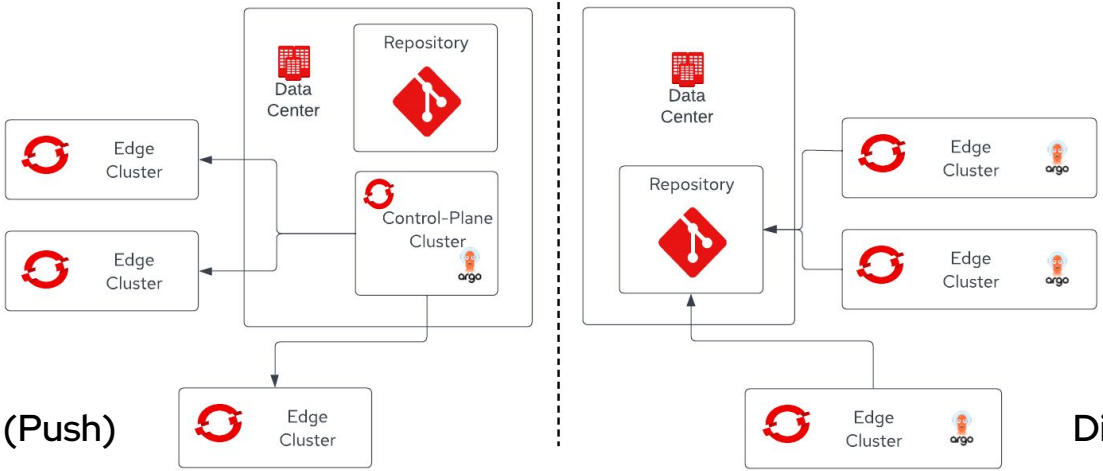


Red Hat

Edge Deployment Management - ArgoCD



Argo CD Edge Deployment Strategies



Centralized (Push)

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

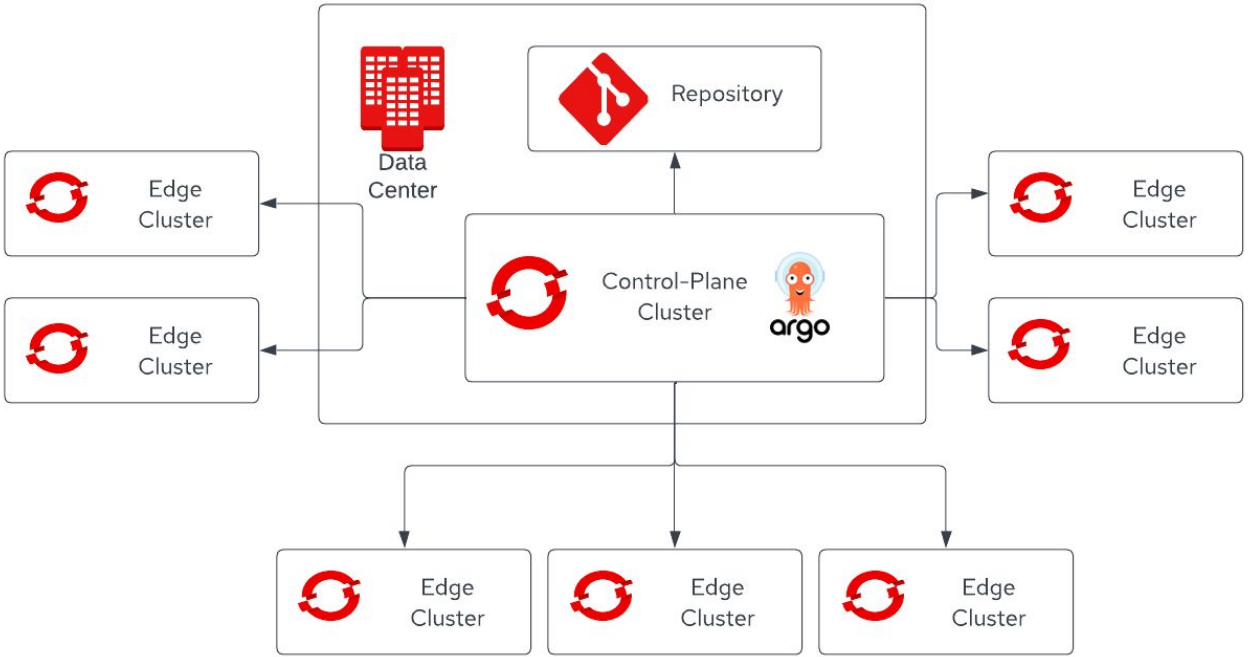
Distributed (Pull)

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

Red Hat

Edge Deployment Model: Centralized ArgoCD

ArgoCD Push



Red Hat

Edge Deployment Model: Centralized ArgoCD

Advantages

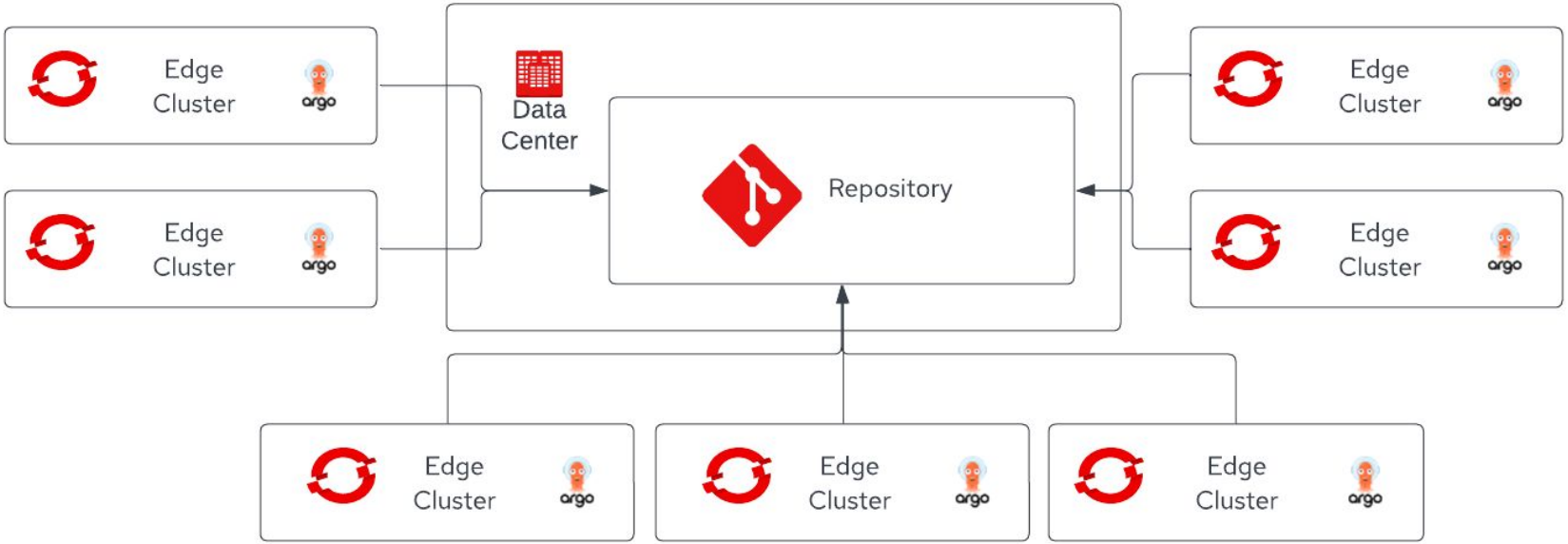
- Single pane of glass - better visibility across the organization
- Easiest to manage - single instance to maintain
- API & CLI integrations

Disadvantages

- ArgoCD performance may degrade with clusters as they scale
- Target cluster API's must be accessible to the central instance (external access)
- Single point of failure and single attack surface

Edge Deployment Model: Distributed ArgoCD

ArgoCD Pull



Red Hat

Edge Deployment Model: Distributed ArgoCD

Advantages

- Most scalable - application controller work is distributed to each edge cluster
- Increased security - OpenShift API servers can be made private

Disadvantages

- No control plane - lack of centralized visibility and management capabilities
- Difficult to manage - many ArgoCDs to access/configure/upgrade
- Git access - clusters require network access and authentication to the git repository

What if I want to...

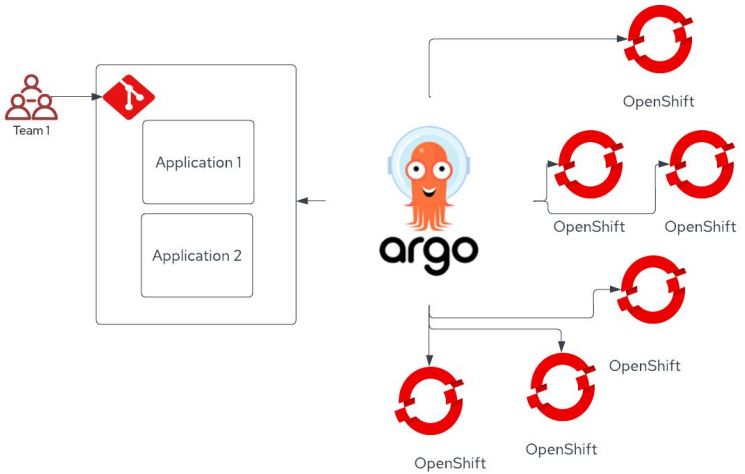
- Deploy ArgoCD applications to multiple OpenShift edge clusters at once
- Deploy multiple ArgoCD applications from a single monorepo
- Allow unprivileged users to deploy ArgoCD applications
- Deploy to different namespaces
- Deploy to different namespaces on a single OpenShift edge cluster
- Deploy from different Git repositories or folders/branches
- Do all of the above by only a single instance Kubernetes CRD

There's a solution for that...

ArgoCD ApplicationSets

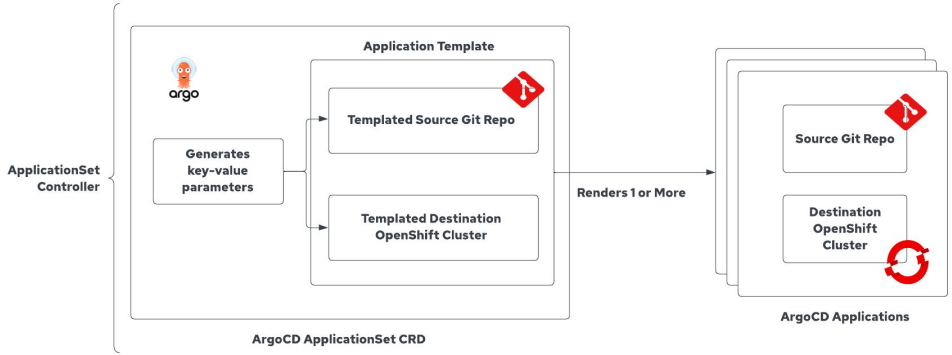
Addressing *Too Many Apps Problem*

- Automatically create and deploy applications based on discovery
  - Paths/files in a Git repo
  - Clusters registered to ArgoCD
  - Combination of the two
- Created as an alternative to App of Apps
- Works with either centralized or distributed deployment models
- Runs as its own controller



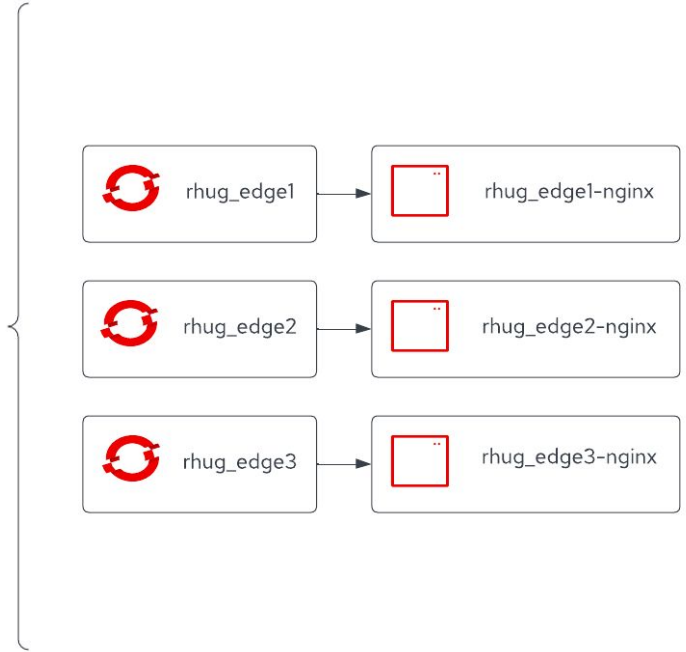
How does the ApplicationSet Controller for ArgoCD work?

```yaml
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: nginx-rhug-example
spec:
  generators:
  - list:
      elements:
      - cluster: rhug_edge1
        url: https://10.10.10.10
      - cluster: rhug_edge2
        url: https://10.20.10.10
      - cluster: rhug_edge3
        url: https://10.30.10.10
  template:
    metadata:
      name: '{{cluster}}-nginx'
    spec:
      project: default
      source:
        repoURL: https://github.com/imryanetten/rhug-examples.git
        targetRevision: HEAD
        path: applicationset/examples/list-generator/nginx/{{cluster}}
      destination:
        server: '{{url}}'
        namespace: nginx-rhug-example
```



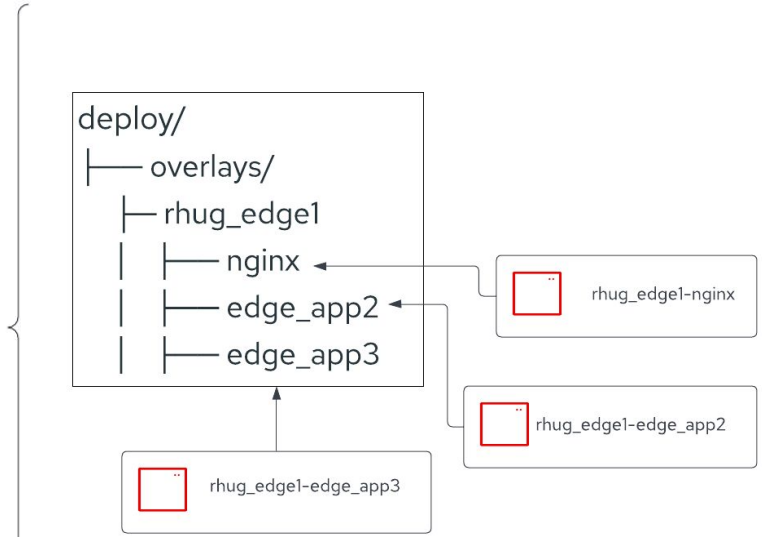
ApplicationSets - Centralized ArgoCD

```yaml
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: nginx-rhug-example
spec:
  generators:
  - clusters: {}
  template:
    metadata:
      name: '{{cluster}}-nginx'
    spec:
      project: default
      source:
        repoURL: https://github.com/imryanetten/rhug-examples.git
        targetRevision: HEAD
        path: deploy/overlays/{{cluster}}
      destination:
        server: '{{server}}'
        namespace: nginx-rhug-example
```



ApplicationSets - Distributed ArgoCD

```yaml
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: edge-cluster-add-ons
spec:
  generators:
  - git:
      repoURL: https://github.com/imryanetten/rhug-examples.git
      revision: HEAD
      directories:
      - path: deploy/overlays/rhug_edge1/*
  template:
    metadata:
      name: '{{path.basename}}'
    spec:
      project: default
      source:
        repoURL: https://github.com/imryanetten/rhug-examples.git
        targetRevision: HEAD
        path: {{path}
      destination:
        server: https://kubernetes.default.svc
```

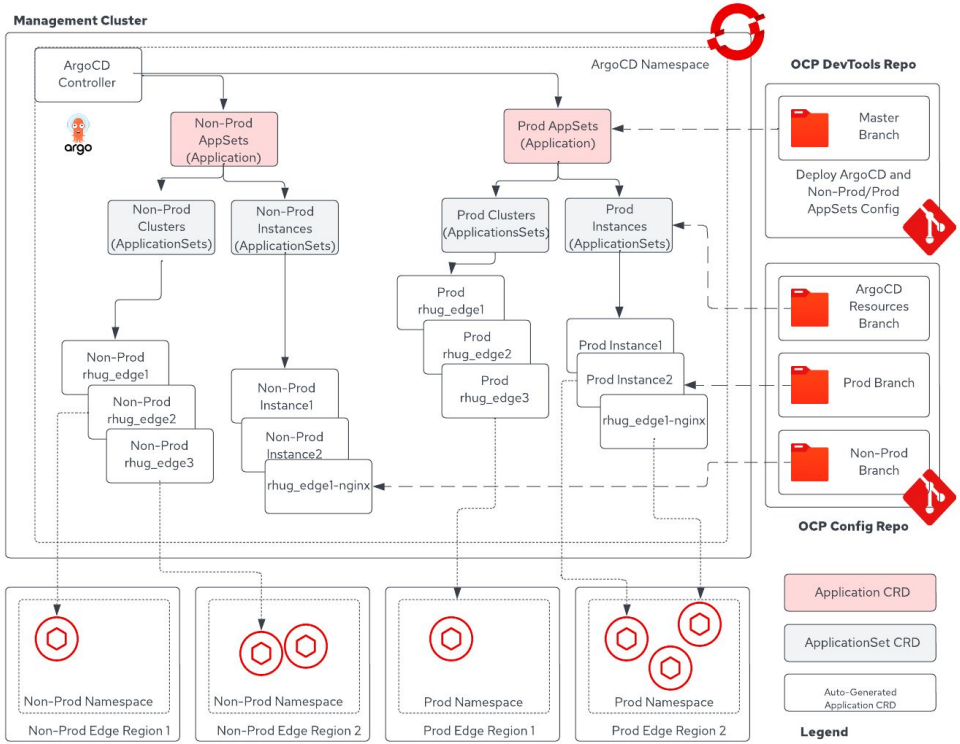


Red Hat

Edge Deployment

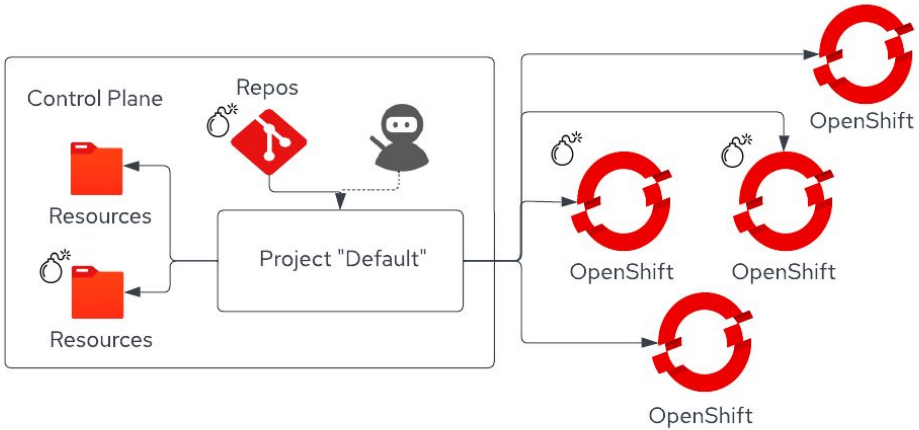
AppSets and Matrix Generator

```yaml
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: rhug-edge-cluster-git
spec:
  generators:
  # matrix 'parent' generator
  - matrix:
      generators:
        # cluster generator, 'child' #1
        - clusters: {}
        # git generator, 'child' #2
        - git:
            repoURL: https://github.com/imryanetten/rhug-examples.git
            revision: HEAD
            directories:
            - path: deploy/overlays/rhug_region1/*
```



Securing ArgoCD

- Use a dedicated project for the control plane
- Delete the “default” project
- Block ClusterRoleBindings in (most) projects
- Narrow roles on remote and edge clusters



Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make
Red Hat a trusted adviser to the Fortune 500.


linkedin.com/company/red-hat


youtube.com/user/RedHatVideos


facebook.com/redhatinc


twitter.com/RedHat

Red Hat