
cdCon
+
GitOpsCon

cdCon + GitOpsCon

# Managing GitOps deployments in multi-cluster production environments

*Roberto Carratalá*
*Cloud Services Black Belt @ Red Hat*

# Speaker Info

## Roberto Carratalá

**Sr Cloud Services Black Belt, Red Hat**

- Kubernetes and DevOps enthusiast
- Open Source contributor & maintainer
- Tech Public Writer & Speaker
- rcarrata.com k8s blog

# Initial Scenario

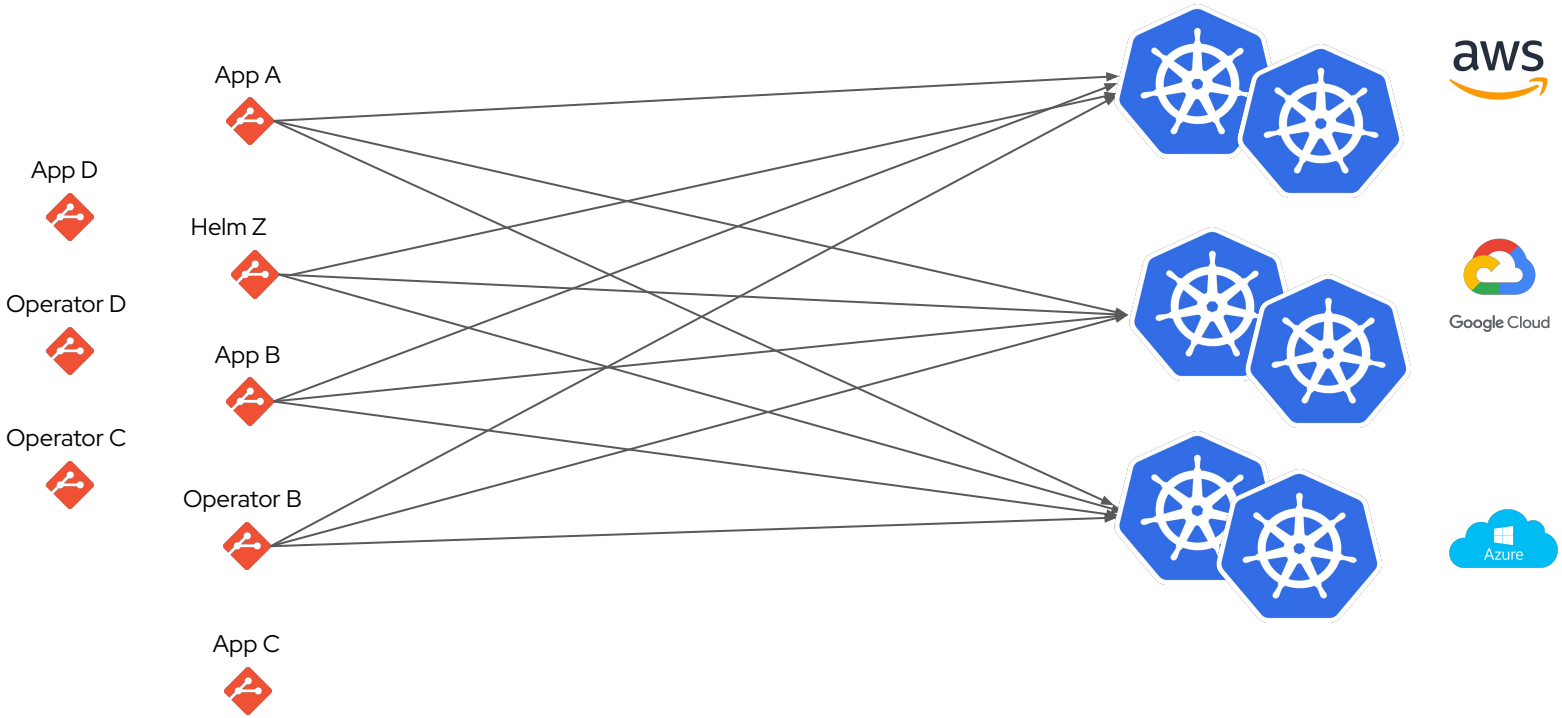


Deploy my apps in Multiple k8s Clusters in Multiple regions across the world

# Initial Scenario

cdCon + GitOpsCon

## Deploy my apps in Multiple k8s Clusters



# Initial Scenario



GitOps

# GitOps Multi-Cloud Patterns - Resources

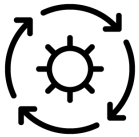
https://tinyurl.com/CdCon-GitOpsCon23

# GitOps Principles



The system is described declaratively

The desired state is versioned in Git



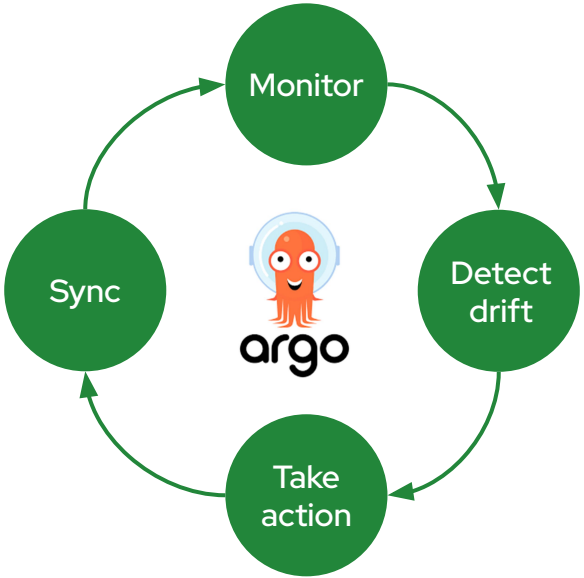
Approved changes can be applied automatically

A controller exists to detect and act on drift

# GitOps Tooling - ArgoCD



- 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



# Pattern 0 - Yet another K8s object

cdCon + GitOpsCon

- Argo CD applications, projects and settings **can be defined declaratively using Kubernetes manifests**.
- **Source** reference to the desired state in Git (repository, revision, path, environment)
- **Destination** reference to the target cluster and namespace.
- **SyncPolicy** - sync (manual or automatically) an application when it detects differences between the desired manifests in Git, and the live state in the cluster.

```yaml
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: bgdk-app
  namespace: argocd
spec:
  destination:
    namespace: bgdk
    server: https://kubernetes.default.svc
  project: default
  source:
    path: apps/demo1/bgd/overlays/bgdk
    repoURL: https://github.com/rcarrata/CdCon23-GitOpsMultiCluster.git
    targetRevision: main
  syncPolicy:
    automated:
      prune: true
      selfHeal: true
    syncOptions:
      - CreateNamespace=true
```

# Pattern 1 - GitOps Application with Kustomize

cdCon + GitOpsCon

Kustomize traverses a Kubernetes manifest to add, remove or update configuration options without forking.

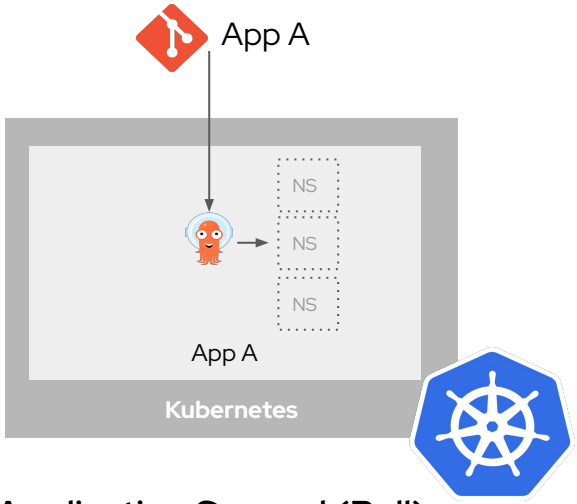
The principles of kustomize are:

- Purely declarative approach to configuration customization
- Manage an arbitrary number of distinctly customized Kubernetes configurations
- Every artifact that kustomize uses is plain YAML and can be validated and processed as such
- As a "templateless" templating system; it encourages using YAML without forking the repo.


```
~/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
```

# Demo Pattern 1 - Deploying GitOps Application with Kustomize

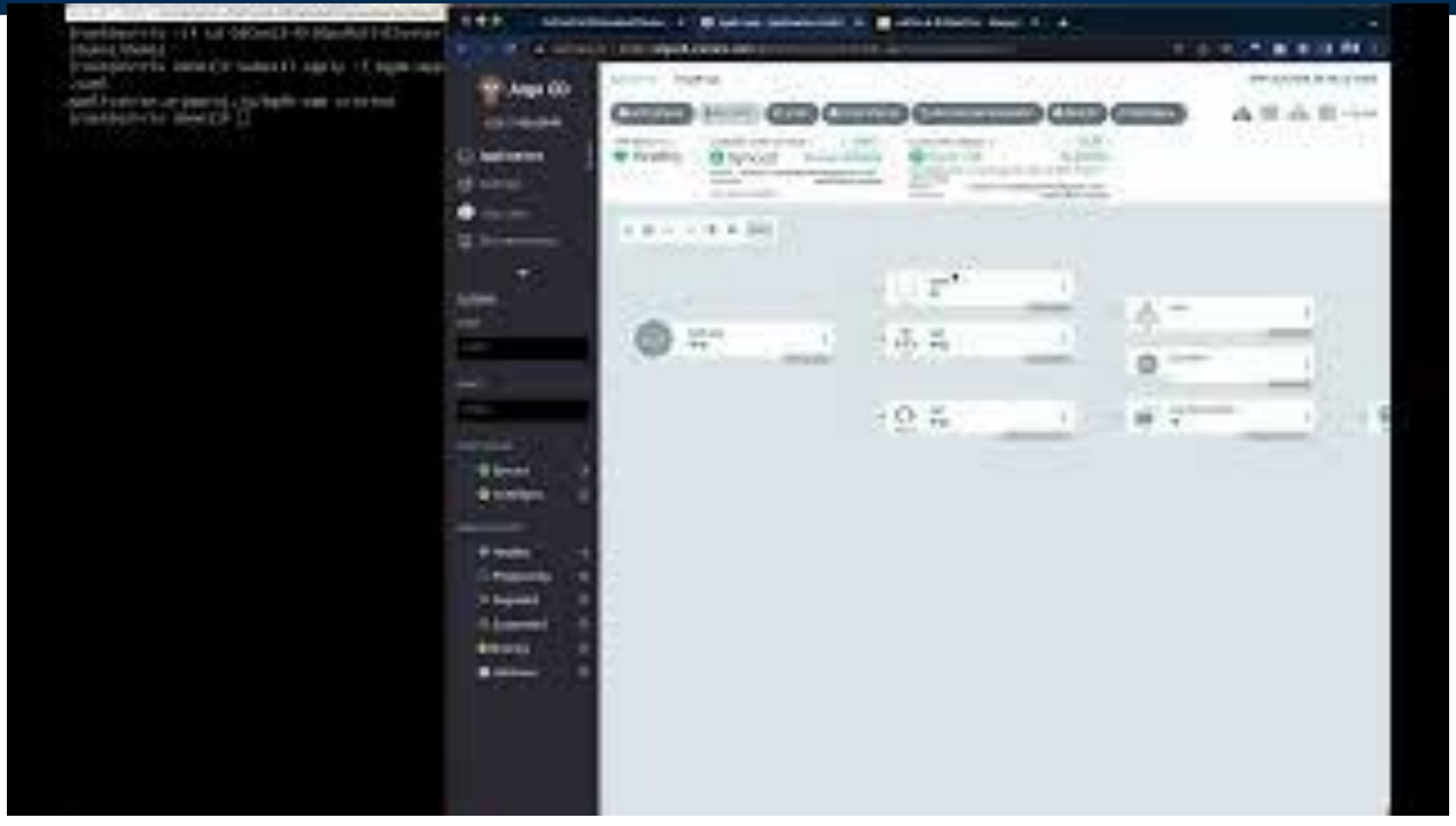
cdCon + GitOpsCon




Application Scoped (Pull)

Demo Resources

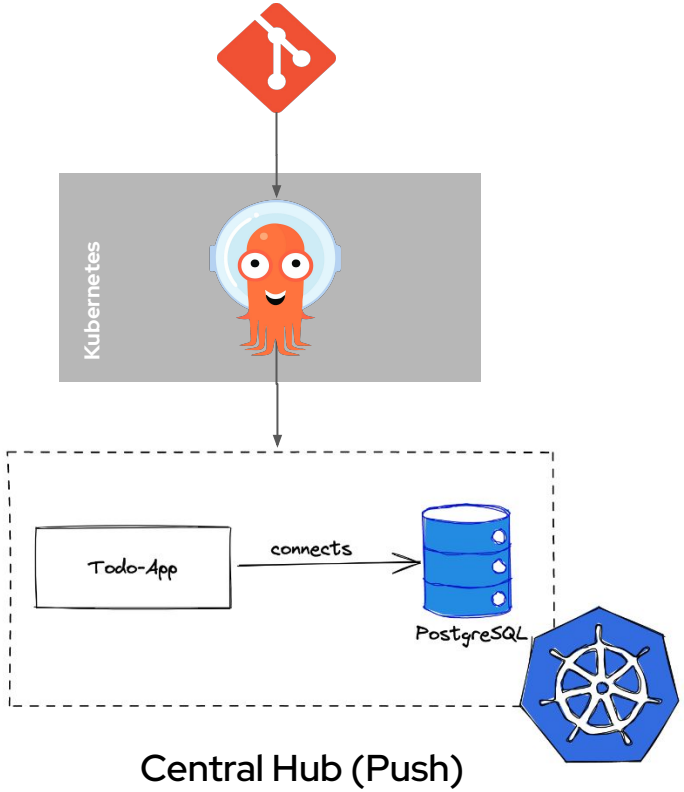
# Demo Pattern 1 - Deploying GitOps Application with Kustomize



# Pattern 2 - Deploying GitOps Apps in Remote Clusters

cdCon + GitOpsCon

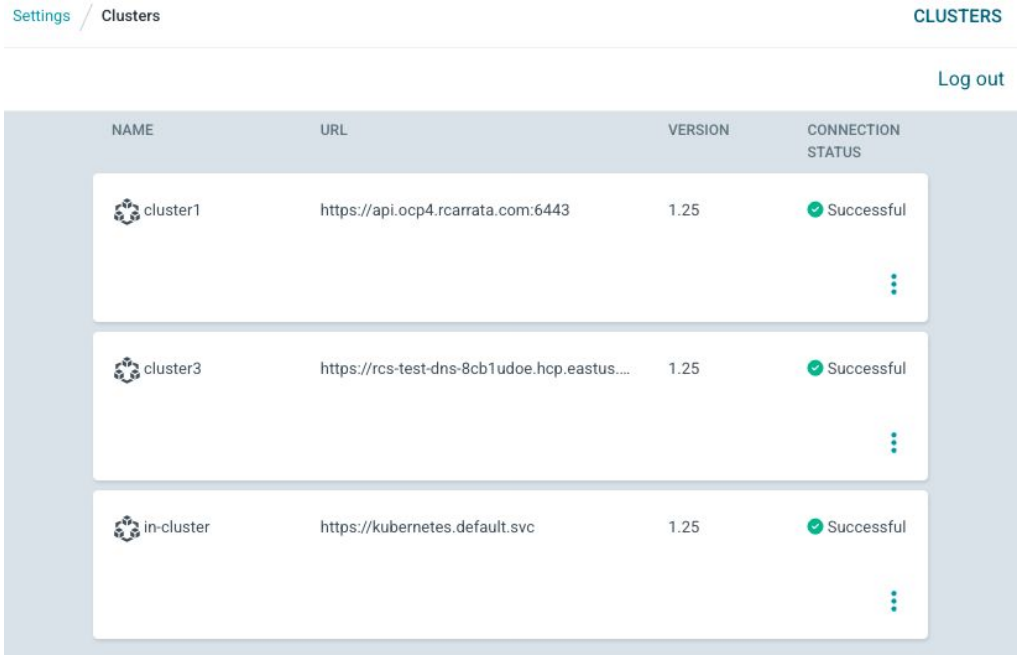
What about deploying my App into remote Clusters?



# Pattern 2 - Deploying GitOps Apps in Remote Clusters

cdCon + GitOpsCon

- Remote clusters can be added as Managed Clusters in ArgoCD
- Cluster credentials are stored in secrets same as repositories or repository credentials.
- Each secret must have label *argocd.argoproj.io/secret-type: cluster*




```
[root@osiris demos]# k get secret -n argocd --show-labels cluster-api.ocp4.rcarrata.com-1884399674
NAME                                        TYPE     DATA   AGE   LABELS
cluster-api.ocp4.rcarrata.com-1884399674    Opaque   3      46h   argocd.argoproj.io/secret-type=cluster
```

# Pattern 2 - Deploying GitOps Apps in Remote Clusters

cdCon + GitOpsCon

- Remote clusters can be referred into the ArgoCD Application
- **destination** reference to the target cluster and namespace.
- For the cluster one of ***server*** or ***name*** can be used, but not both (which will result in an error).
- Under the hood when the server is missing, it is calculated based on the name and used for any operations.

```yaml
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: todo-app
  namespace: argocd
spec:
  destination:
    name: cluster1
    namespace: todo
  project: default
  source:
    path: apps/demo2/todo
    repoURL: https://github.com/rcarrata/CdCon23-GitOpsMultiCluster.git
    targetRevision: main
  syncPolicy:
    automated:
      prune: true
      selfHeal: true
    syncOptions:
      - CreateNamespace=true
```

# Pattern 2 - Deploying GitOps Apps in Remote Clusters

cdCon + GitOpsCon

## SyncWaves

- A **Syncwave** is a way to order how Argo CD applies the manifests that are stored in git.
- All manifests have a wave of zero by default, but you can set these by using the `argocd.argoproj.io/sync-wave` annotation.

```yaml
apiVersion: v1
kind: Namespace
metadata:
  name: todo
  annotations:
    argocd.argoproj.io/sync-wave: "-1"
```

## Hooks

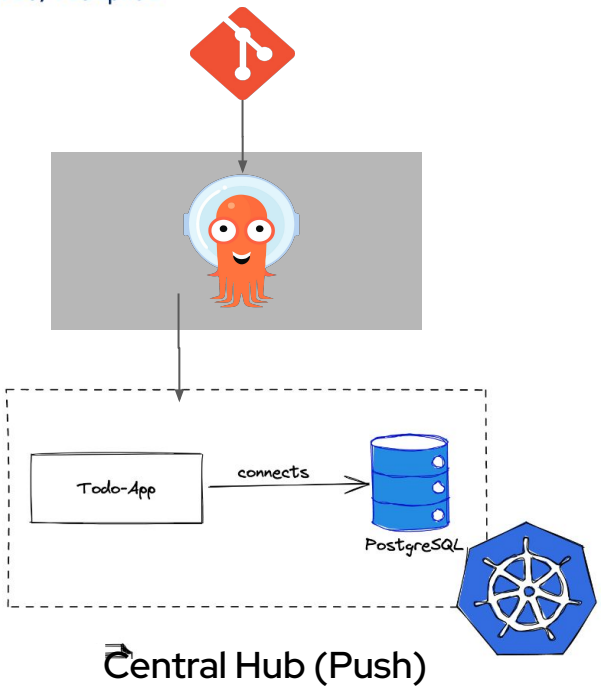
- Controlling your sync operation can be further redefined by using **hooks**.
- These hooks can run before, during, and after a sync operation.

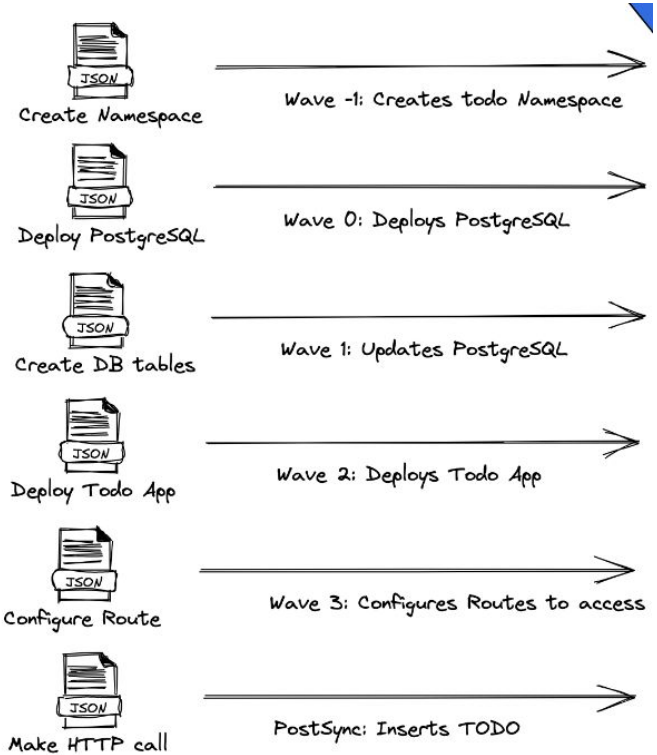
```yaml
metadata:
  annotations:
    argocd.argoproj.io/hook: PreSync
```

# Demo Pattern 2 - Deploying GitOps Apps in Remote Clusters

cdCon + GitOpsCon

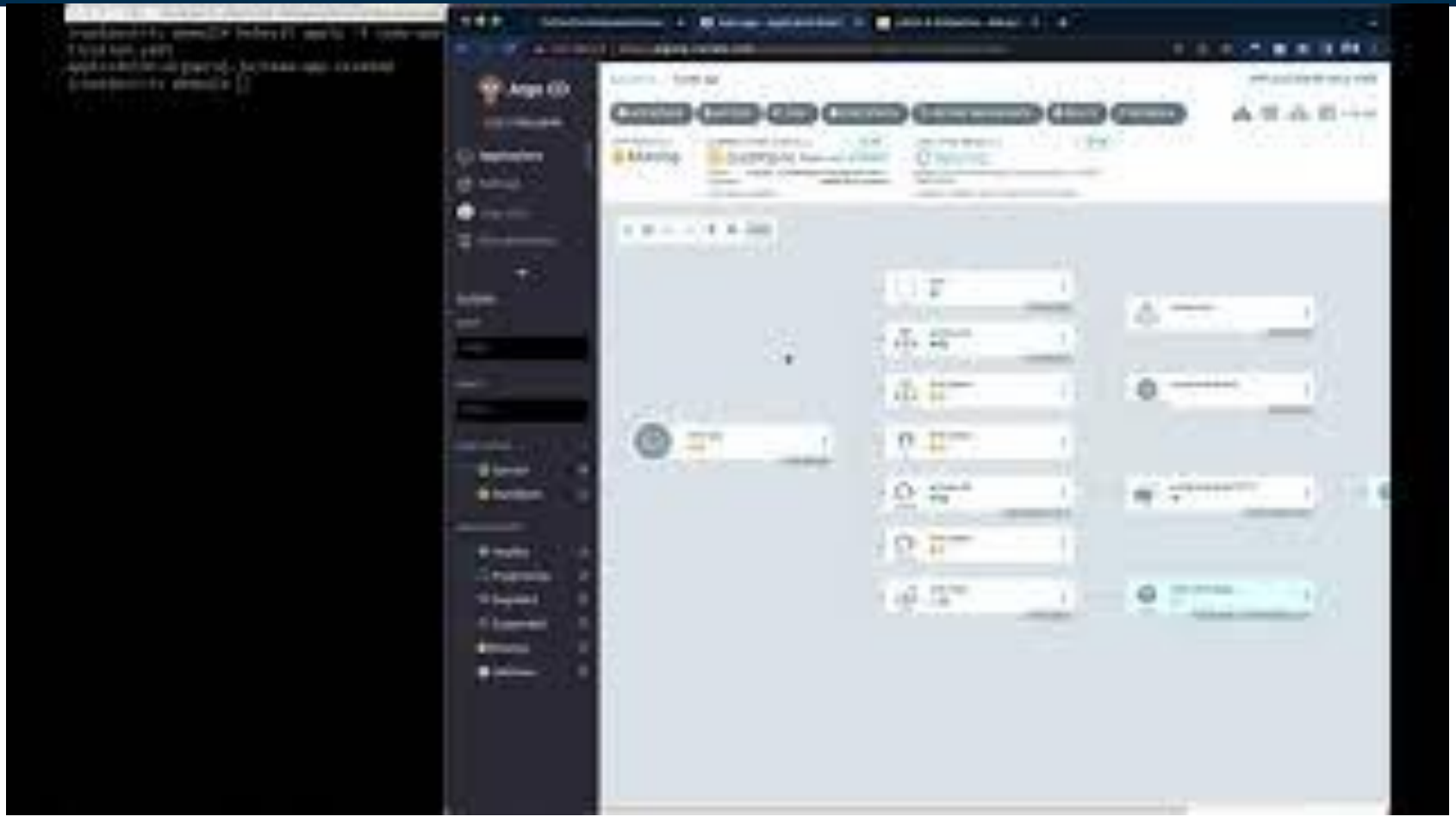
```yaml
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: todo-app
  namespace: argocd
spec:
  destination:
    name: cluster1
    namespace: todo
```



# Demo 2 Pattern - Deploying GitOps Apps in Remote Clusters

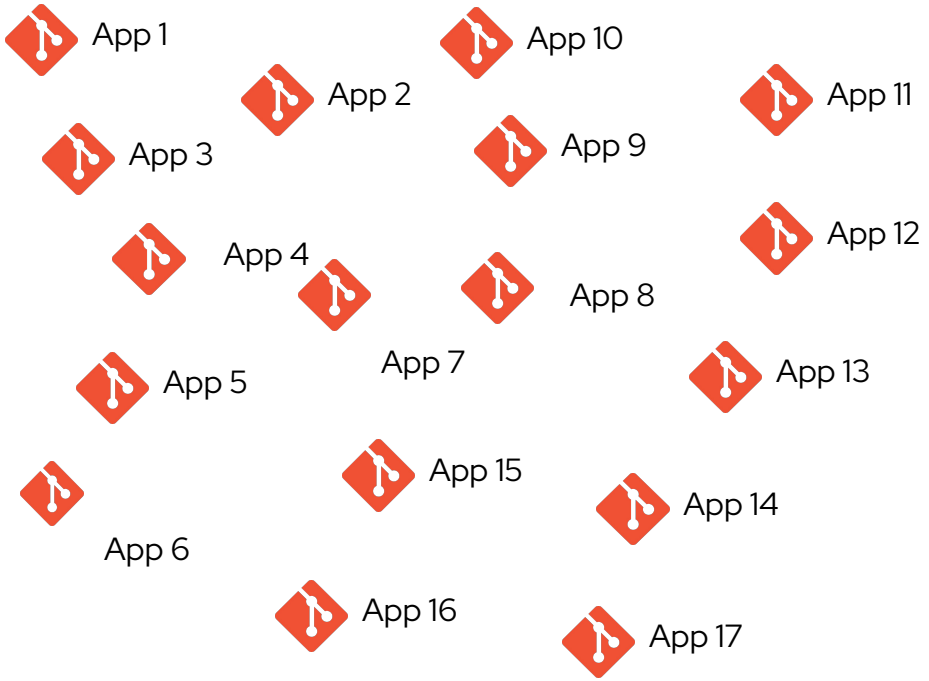
cdCon + GitOpsCon



# Pattern 3 - Managing GitOps Apps at scale

cdCon + GitOpsCon

What about deploying multiple related applications at once?

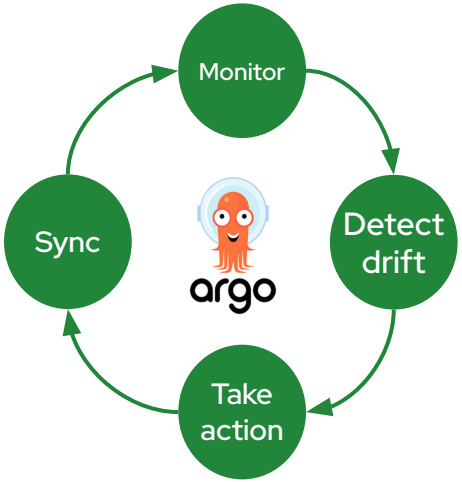


# Pattern 3 - Managing GitOps Apps at scale

cdCon + GitOpsCon

## ArgoCD ApplicationSets

- Use a single Kubernetes manifest to **target multiple Kubernetes clusters** with Argo CD
- Use a single Kubernetes manifest to **deploy multiple applications from one or multiple Git repositories** with Argo CD
- Improved support for monorepos: **multiple Argo CD Application resources defined within a single Git repository**
- Within multitenant clusters, **improves the ability of individual cluster tenants to deploy applications** using Argo CD

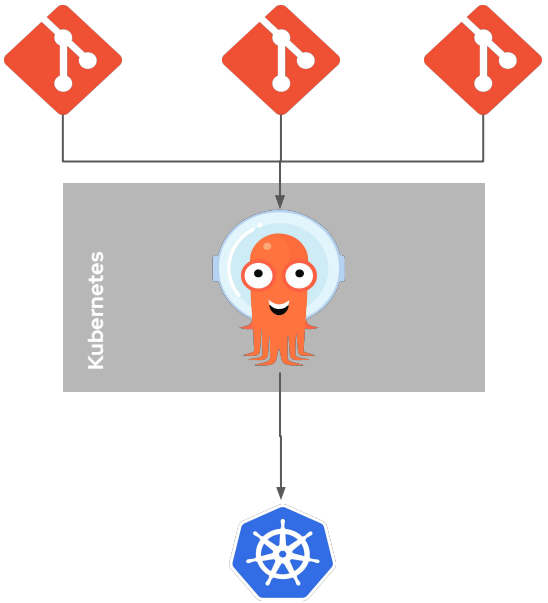


# Pattern 3 - Managing GitOps Apps at scale

cdCon + GitOpsCon

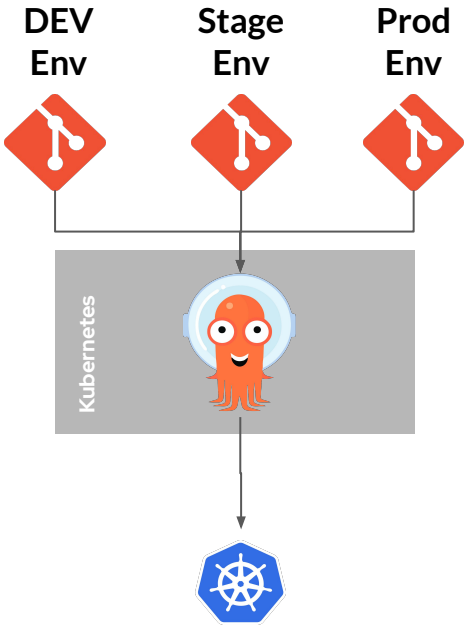
## Argo CD ApplicationSet Generators

- **List generator:** Generates parameters based on a fixed list of cluster name/URL values, as seen in the example above.
- **Cluster generator:** Automatically generates cluster parameters based on the clusters that are defined within Argo CD.
- **Git generator:** Generates parameters based on files or folders that are contained within the Git repository defined within the generator resource.
- **Matrix generator:** Combines the generated parameters of two other generators.



# Demo Pattern 3 - Managing GitOps Apps at scale



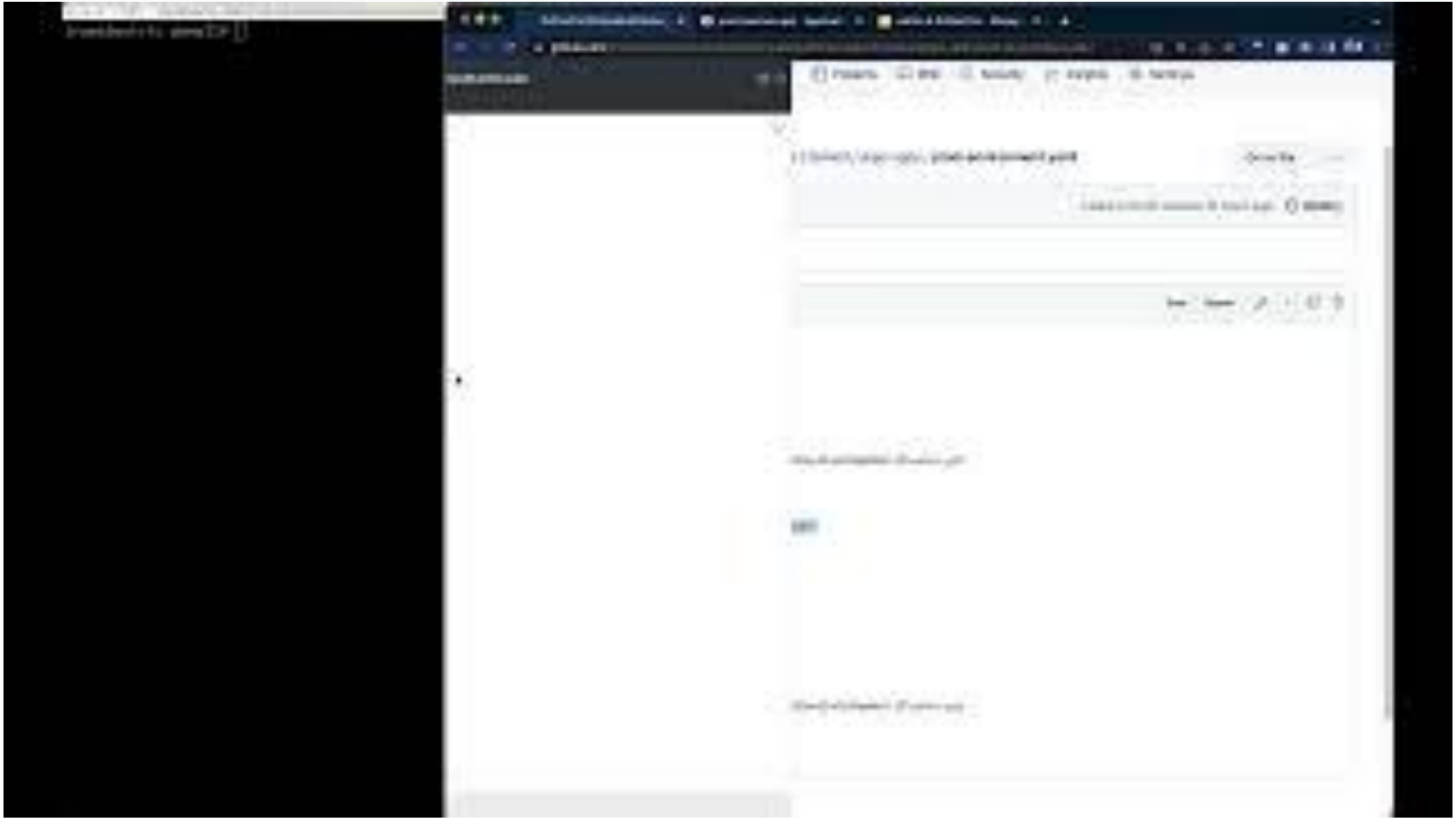



Central Hub (Push)

Demo Resources

# Demo Pattern 3 - Managing GitOps Apps at scale

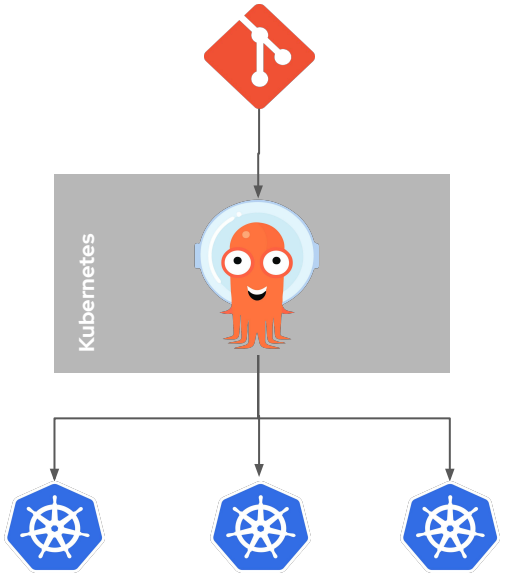
cdCon + GitOpsCon



# Pattern 4 - GitOps Multi Cluster Deployment Strategies

cdCon + GitOpsCon

- **List generator:** Generates parameters based on a fixed list of cluster name/URL values, as seen in the example above.
- **Cluster generator:** Automatically generates cluster parameters based on the clusters that are defined within Argo CD.
- **Git generator:** Generates parameters based on files or folders that are contained within the Git repository defined within the generator resource.
- **Matrix generator:** Combines the generated parameters of two other generators.

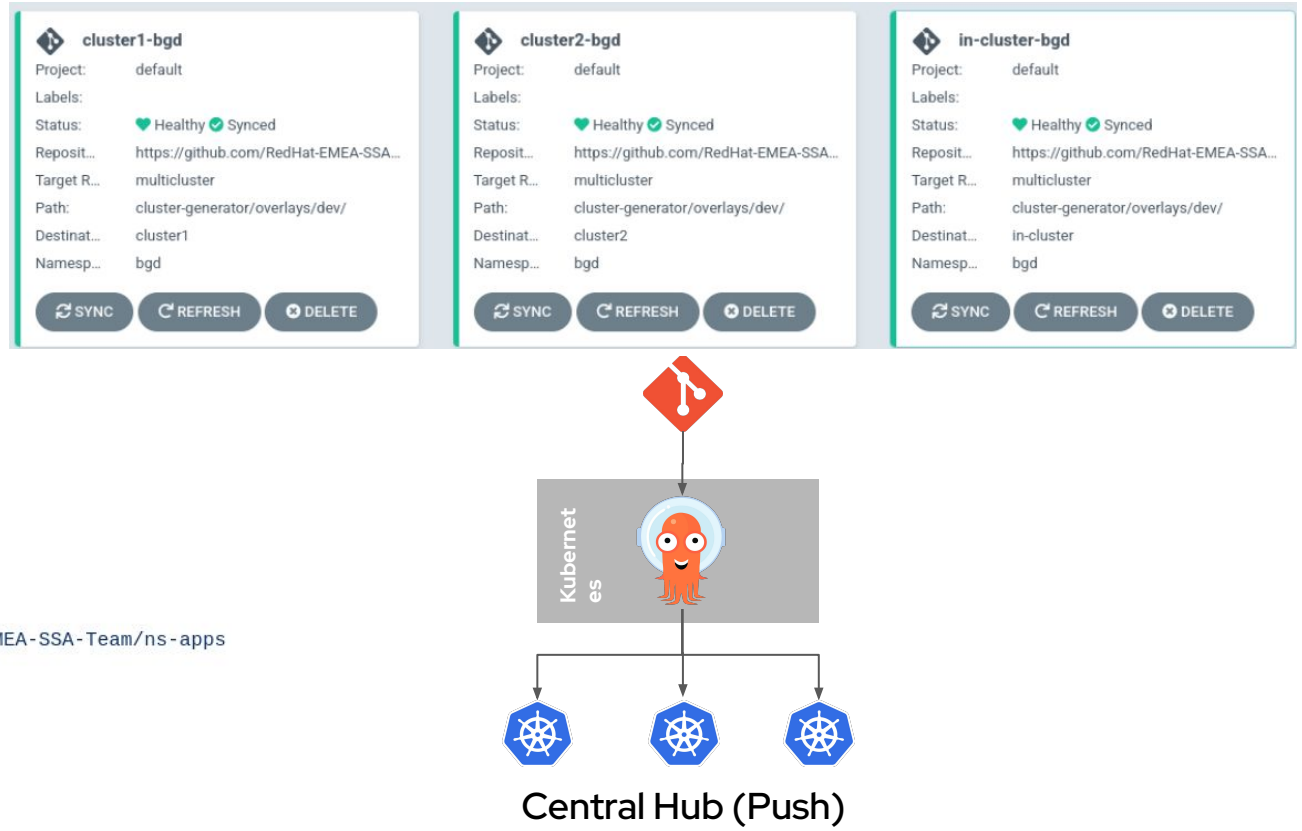



Central Hub (Push)

# Pattern 4 - GitOps Multi Cluster Deployment Strategies

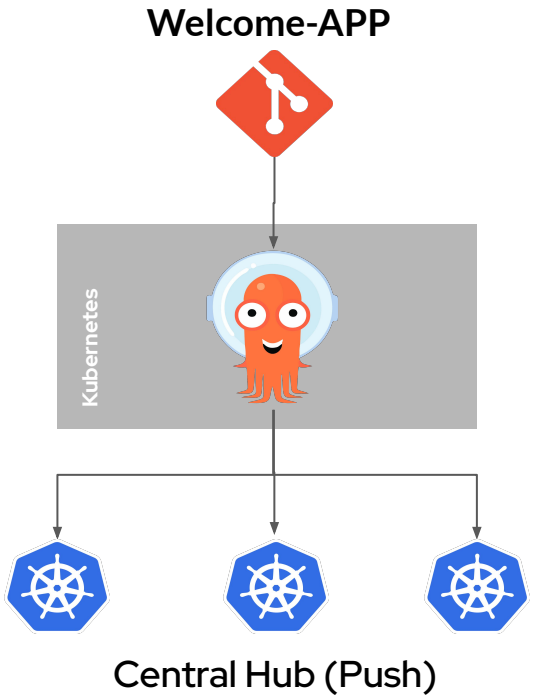
cdCon + GitOpsCon

```yaml
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: welcome-app-appset
  namespace: openshift-gitops
spec:
  generators:
    - clusters: {}
  template:
    metadata:
      name: "{{name}}-welcome-app"
    spec:
      project: default
      syncPolicy:
        automated:
          prune: true
          selfHeal: true
      source:
        repoURL: https://github.com/RedHat-EMEA-SSA-Team/ns-apps
        targetRevision: multicluster
        path: cluster-generator/base/
      destination:
        server: "{{server}}"
        namespace: welcome-app
```



# Demo Pattern 4 - GitOps Multi Cluster Deployment Strategies

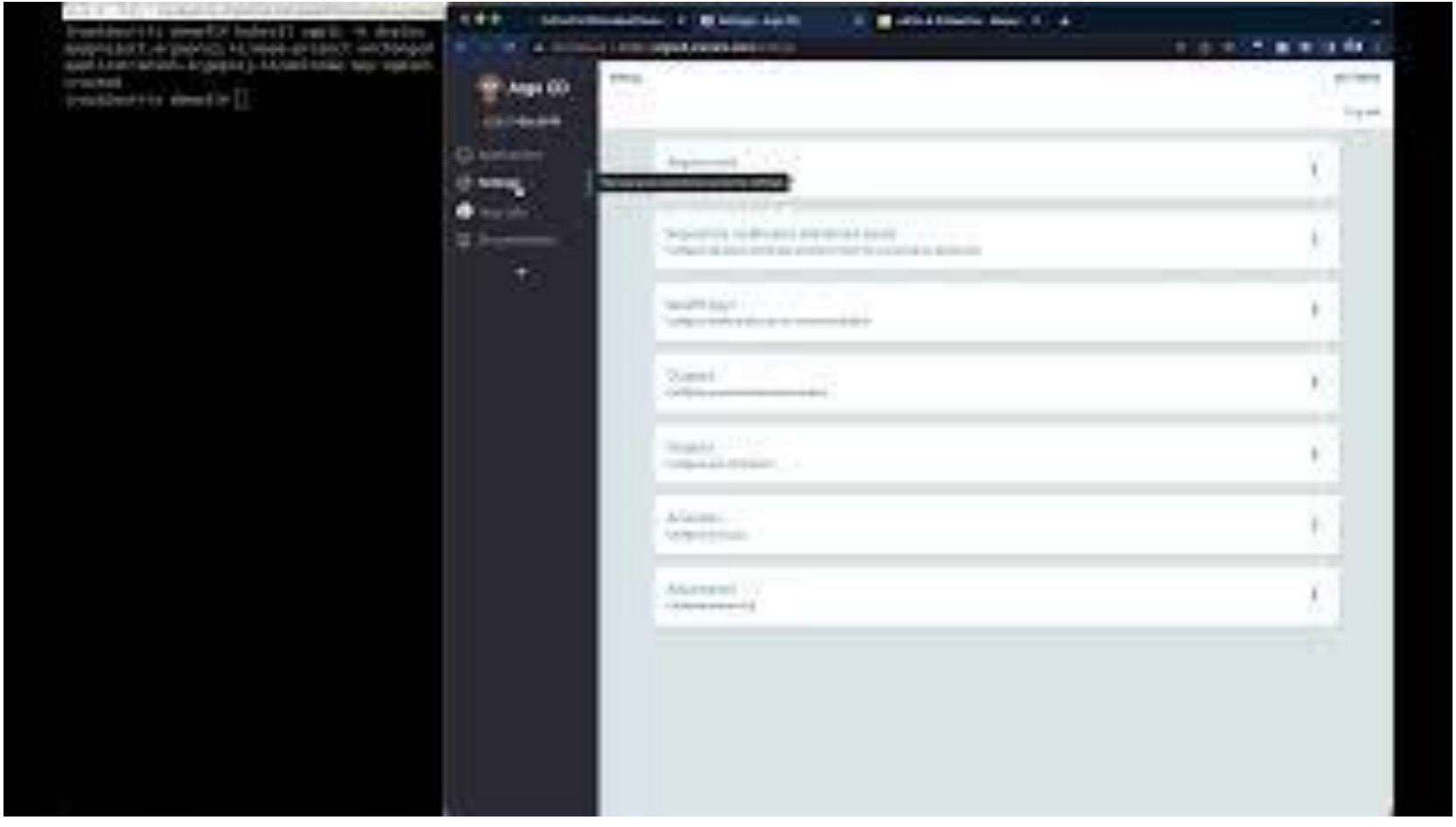





Demo Resources

# Demo Pattern 4 - GitOps Multi Cluster Deployment Strategies





# Pattern 5 - GitOps Multi Cluster & Multi-Environment Strategies

cdCon + GitOpsCon

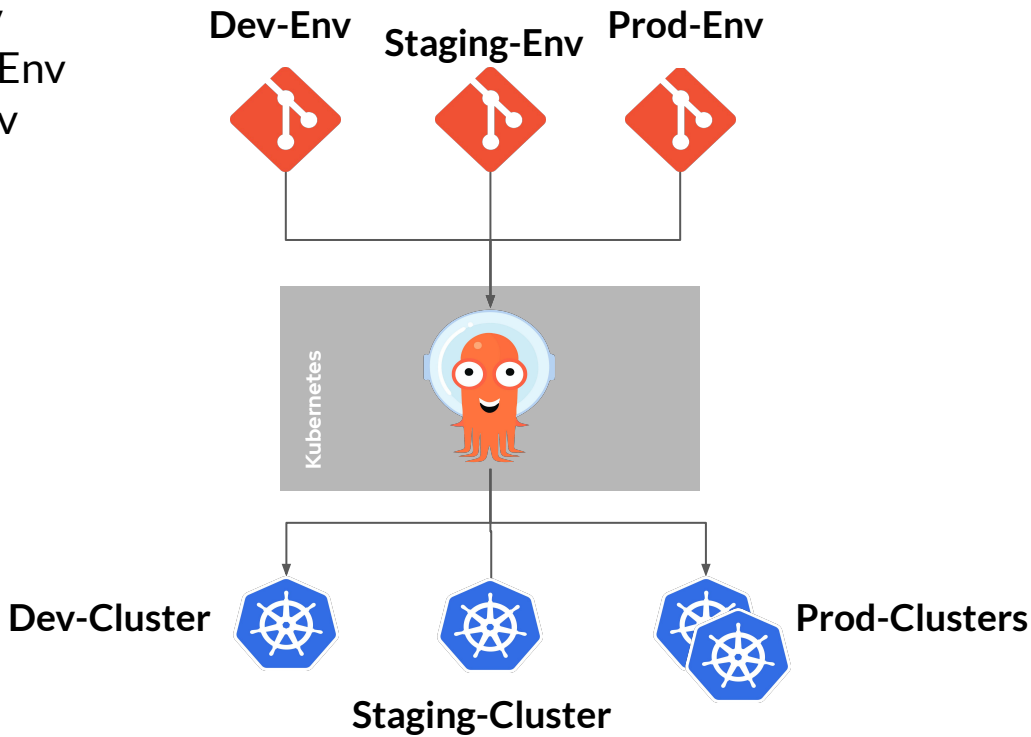
**Multiple Environments**

- Dev-Env
- Staging-Env
- Prod-Env

**What about deploying my multiple environments into multiple Clusters?**

**Multiple Clusters**

- Dev-Cluster
- Staging-Cluster
- Prod-Clusters




WORKED FINE IN DEV
OPS PROBLEM NOW
memegenerator.net

# Pattern 5 - GitOps Multi Cluster & Multi-Environment Strategies

cdCon + GitOpsCon

- A **label selector** may be used to narrow the scope of targeted clusters to only those matching a specific label
- This would requires **to match** the Argo CD cluster secret with the label defined in the ApplicationSet selector

```yaml
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: dev-env-multicluster-app
  namespace: argocd
spec:
  generators:
  - clusters:
      selector:
        matchLabels:
          dev: "true"
  template:
    metadata:
      name: "{{name}}-dev-env-welcome-app"
```

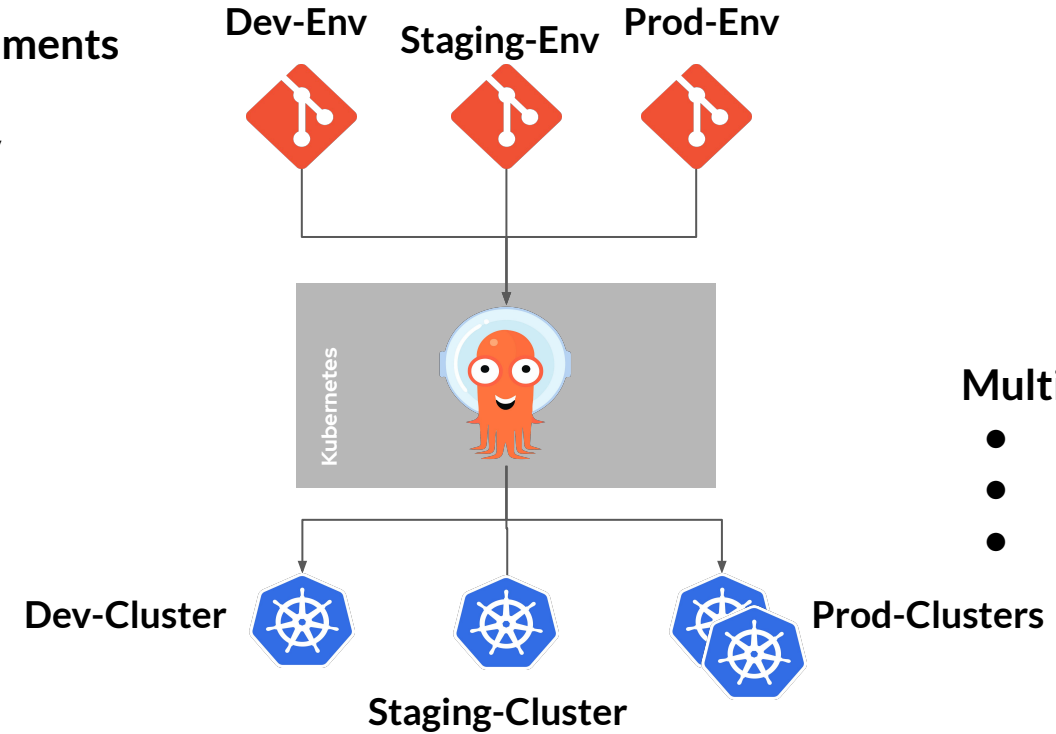
```
[root@osiris demo4]# kubectl get secret -n argocd $DEV -o jsonpath={.metadata.labels} | jq -r .
{
  "argocd.argoproj.io/secret-type": "cluster",
  "dev": "true"
}
```

# Demo Pattern 5 - GitOps Multi Cluster & Multi-Environment Strategies

cdCon + GitOpsCon

**Multiple Environments**

- Dev-Env
- Staging-Env
- Prod-Env




**Multiple Clusters**

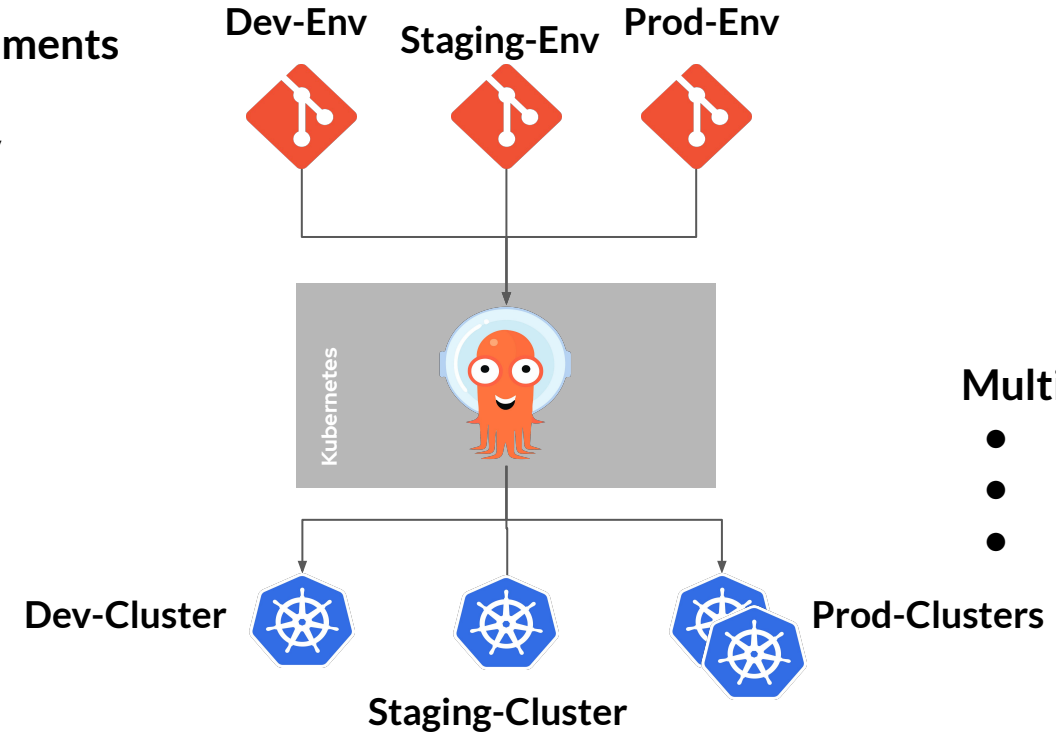
- Dev-Cluster
- Staging-Cluster
- Prod-Clusters

# Demo Pattern 5 - GitOps Multi Cluster & Multi-Environment Strategies

cdCon + GitOpsCon

**Multiple Environments**

- Dev-Env
- Staging-Env
- Prod-Env



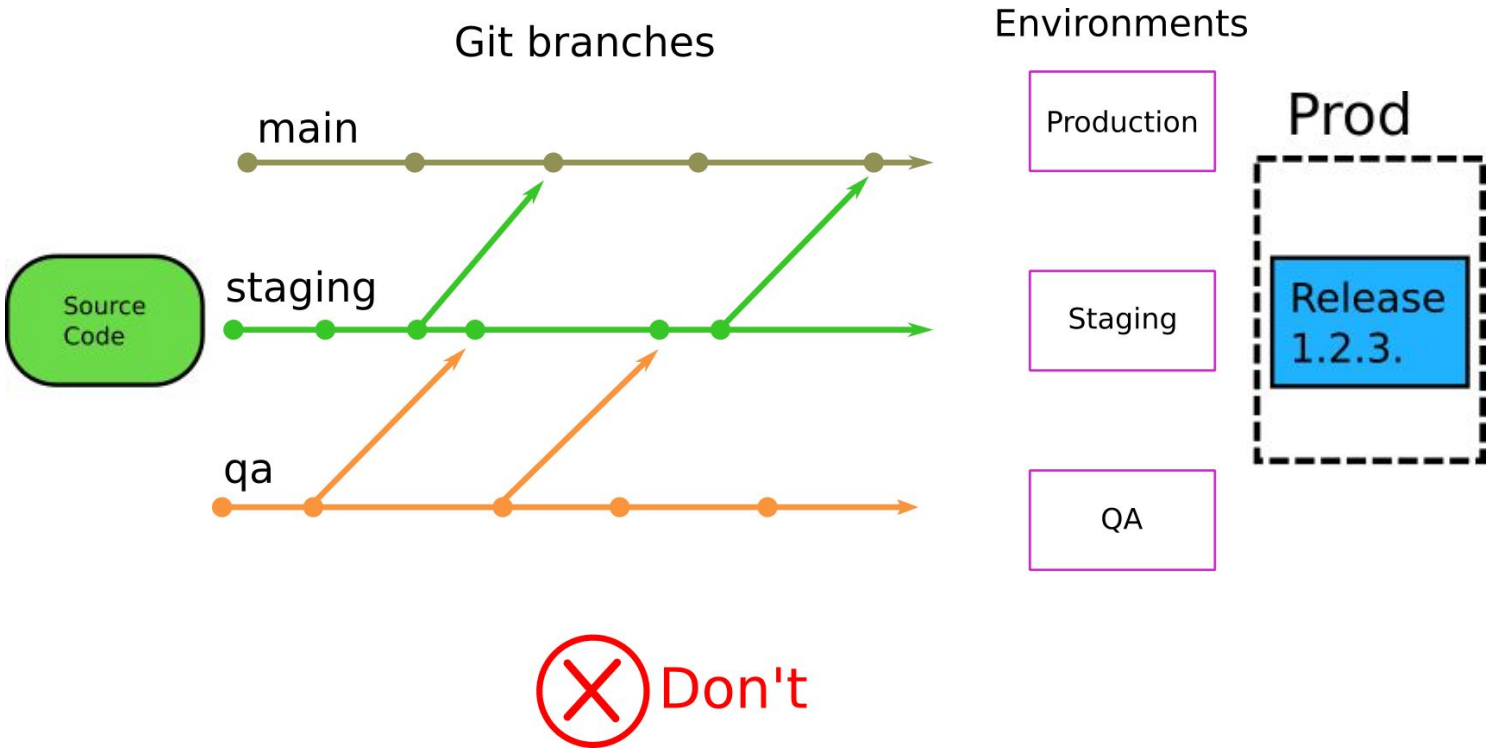

**Multiple Clusters**

- Dev-Cluster
- Staging-Cluster
- Prod-Clusters

# Demo Pattern 5 - GitOps Multi Cluster & Multi-Environment Strategies

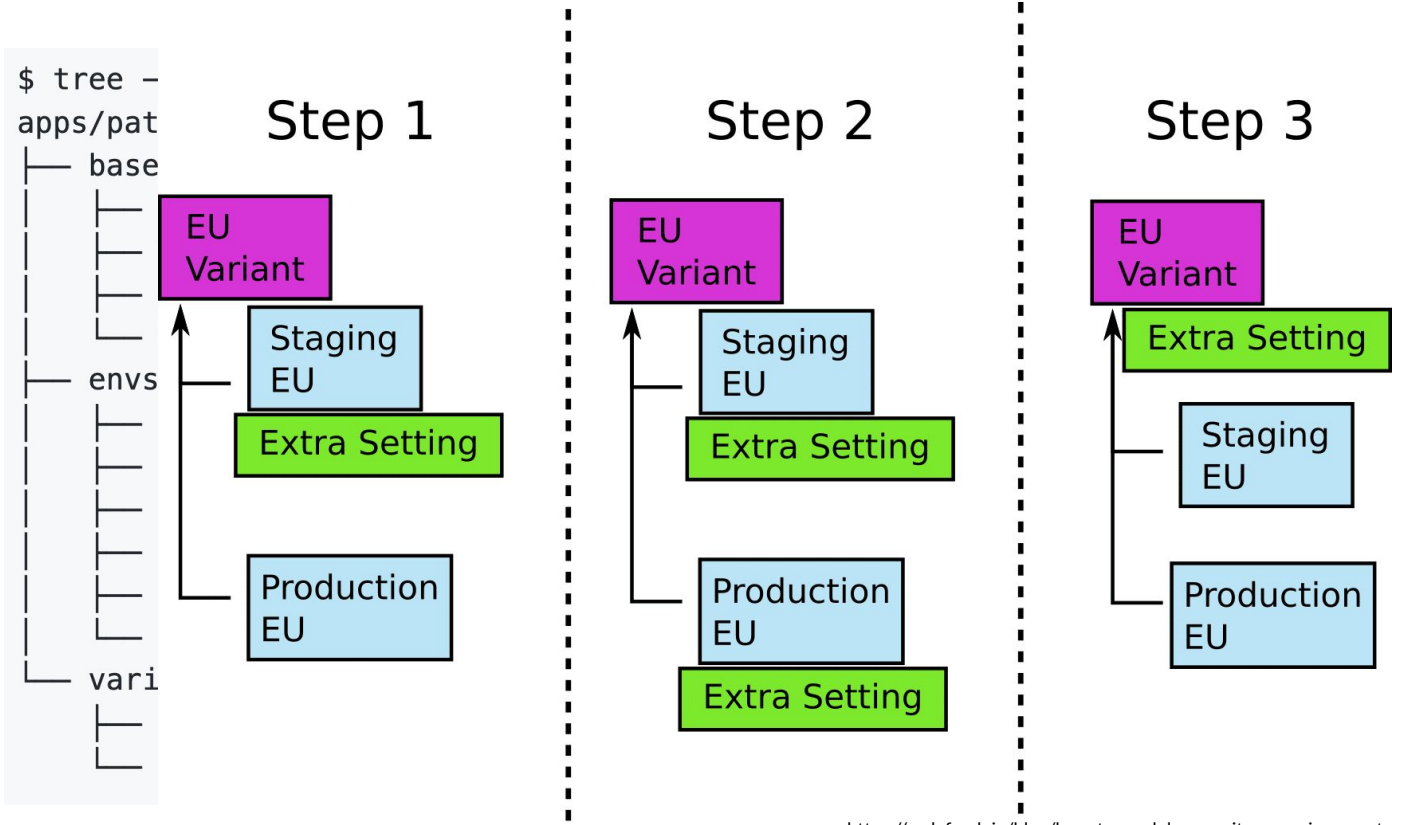
cdCon + GitOpsCon

# Pattern 6 - Promotion between GitOps environments




Source - https://codefresh.io/blog/how-to-model-your-gitops-environments-and-promote-releases-between-them/

# Pattern 6 - Promotion between GitOps environments




https://codefresh.io/blog/how-to-model-your-gitops-environments-and-promote-releases-between-them/

# Pattern 6 - Promotion between GitOps environments

cdCon + GitOpsCon


## Scenario 1 - Promote application version from Dev to Staging Environment in the US:

```
cp envs/dev-gpu/version.yaml envs/staging-us/version.yaml
```

## Scenario 2 - Promote application version from Staging to Prod Environment in the US:

```
cp envs/staging-us/version.yaml envs/prod-us/version.yaml
```

```
spec:
  containers:
    - name: welcome-app
      image: quay.io/rcarrat
```

```
template:
  spec:
    containers:
      - name: welcome-app
        image: quay.io/rcarrata/welcome-app:2.0
```

# Q/A

# Thanks for Attending!