Kubernetes Community Days
Amsterdam 2023

GitOps Patterns for Managing Cloud Native Applications

Germán M. Yébenes | SUSE
Roberto Carratalá | Red Hat

Roberto Carratalá

EMEA Sr Cloud Services Black Belt, Red Hat

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

Germán M. Yébenes

Technical Marketing Manager, Edge Solutions, SUSE

- Technical Writer & Public Speaker
- Open Source contributor
- IoT & Edge playgrounder

Agenda

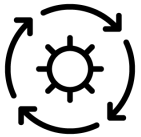
- GitOps Tools and Projects
- **Pattern 1** - Kustomize to the Rescue!
- **Pattern 2** - Sync Waves and Hooks to rule the world
- **Pattern 3** - App of Apps - The GitOps order awakens
- **Pattern 4** - ApplicationSet - The Rise of ArgoCD Applications
- **Pattern 5** - MultiCluster Management - How to rule them all!
- **Pattern 6** - Promotion Releases with GitOps without dying
- **Q/A**

GitOps Tools and Projects

GitOps Principles

The system is described declaratively

The desired state is versioned in Git

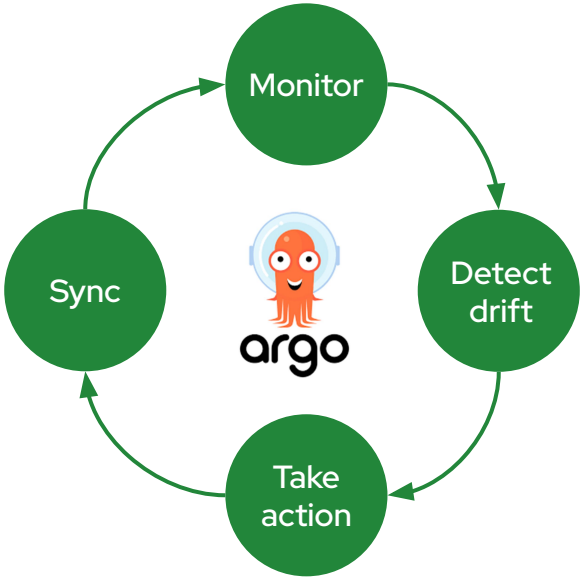


Approved changes can be applied automatically

A controller exists to detect and act on drift

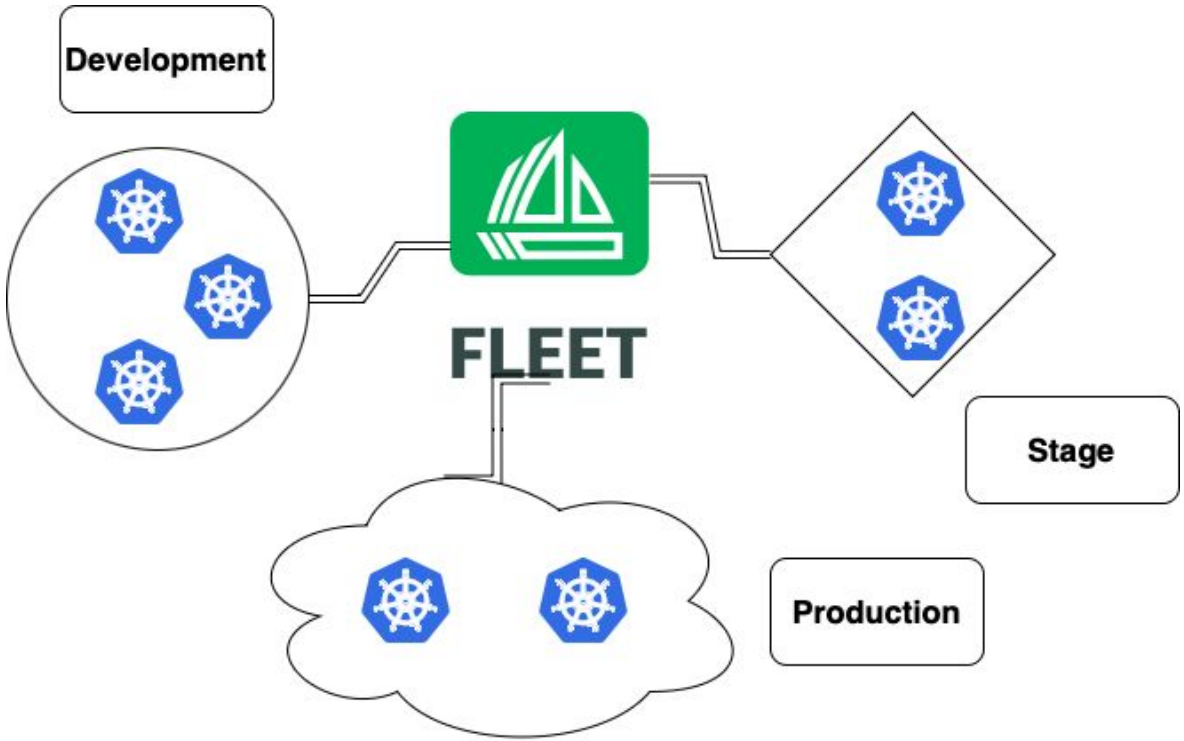
ArgoCD

- 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



Fleet

- Different environment requires different customisation
- Fleet is an open-source gitops solution for large number of clusters
- Clusters can be defined using a yaml on a GitHub Repo and grouping for easier deployment



GitOps Patterns with ArgoCD

https://tinyurl.com/kcdamsgitops

Pattern 0 - Yet another K8s object
ArgoCD Application

- 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/pattern1/bgd/overlays/bgdk
    repoURL: https://github.com/rcarrata/kcd23ams-gitops-patterns
    targetRevision: main
  syncPolicy:
    automated:
      prune: true
      selfHeal: true
    syncOptions:
      - CreateNamespace=true
```

Pattern 1 - Kustomize to the rescue!

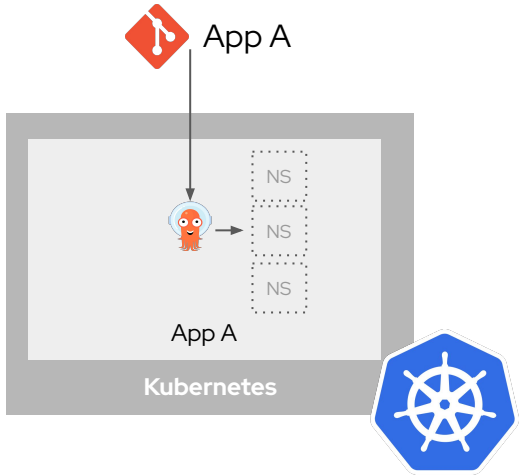
Kustomize traverses a Kubernetes manifest to add, remove or update configuration options without forking.

The principals 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.

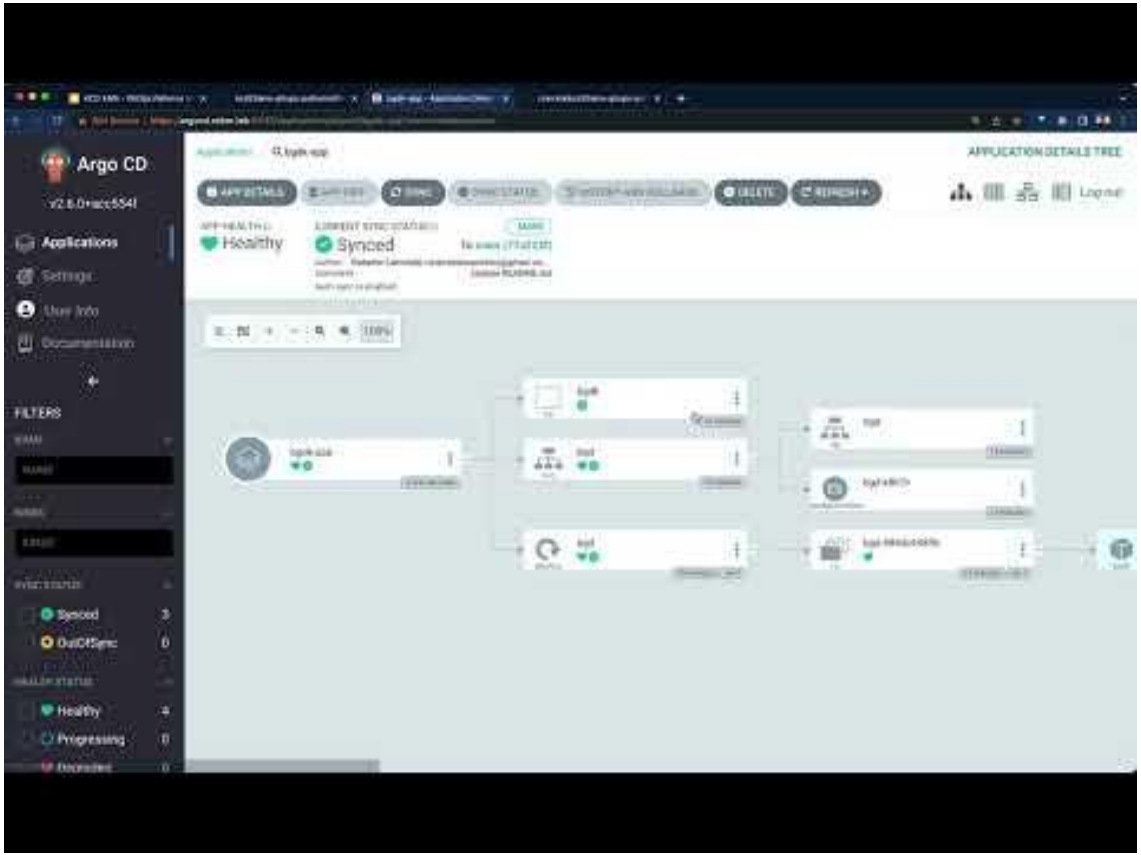
```
~/someApp
├── base
│   ├── deployment.yaml
│   ├── kustomization.yaml
│   └── service.yaml
└── overlays
    ├── development
    │   ├── cpu_count.yaml
    │   ├── kustomization.yaml
    │   └── replica_count.yaml
    └── production
        ├── cpu_count.yaml
        ├── kustomization.yaml
      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

Pattern 1 Demo - Kustomize to the rescue!

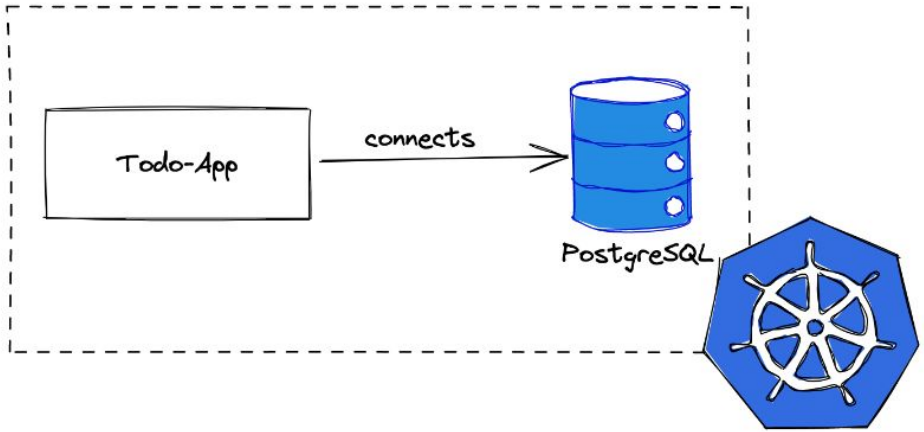


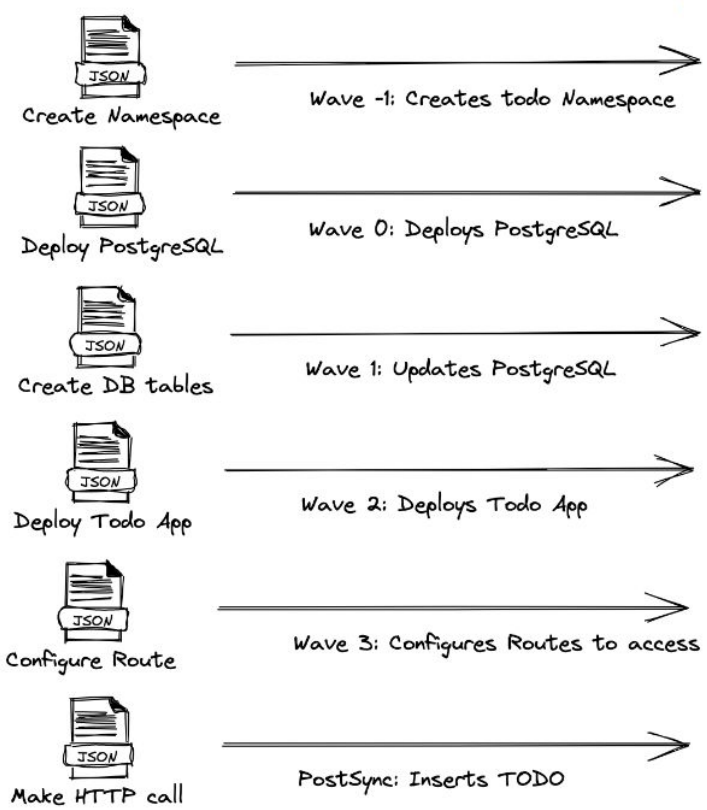
Application Scoped
(Pull)

Pattern 1 Demo - Kustomize to the rescue!



Pattern 2- Controlling Order within GitOps deployments





Pattern 2- Controlling Order within GitOps deployments

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.

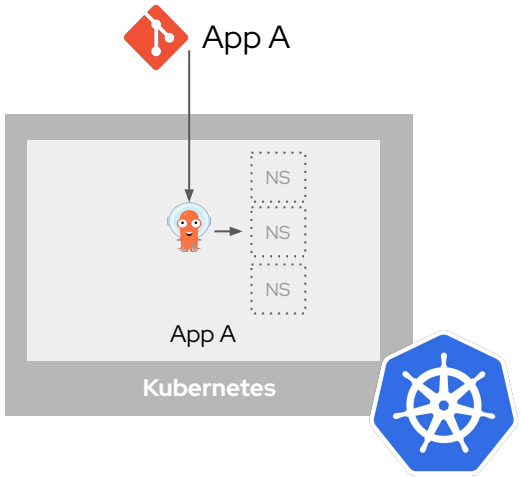
```yaml
apiVersion: v1
kind: Namespace
metadata:
  name: todo
  annotations:
    argocd.argoproj.io/sync-wave: "-1"
```

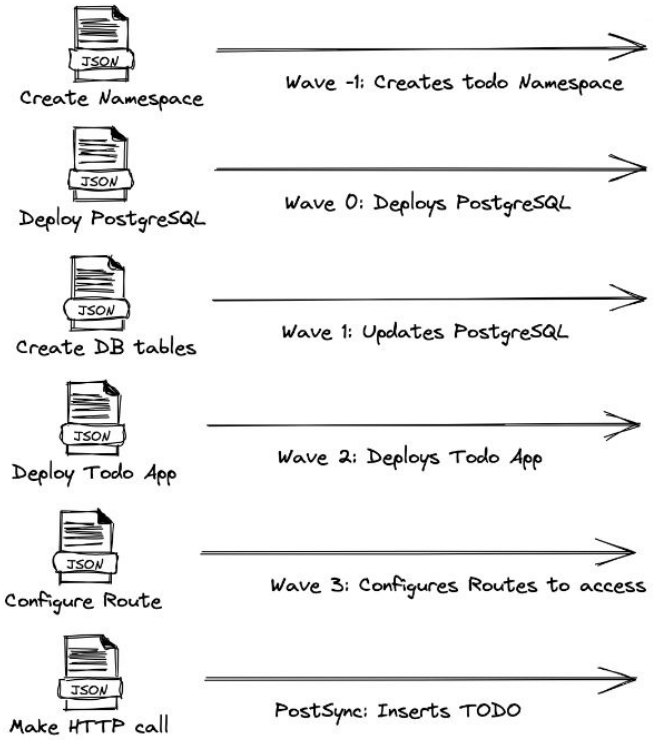
Hooks

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

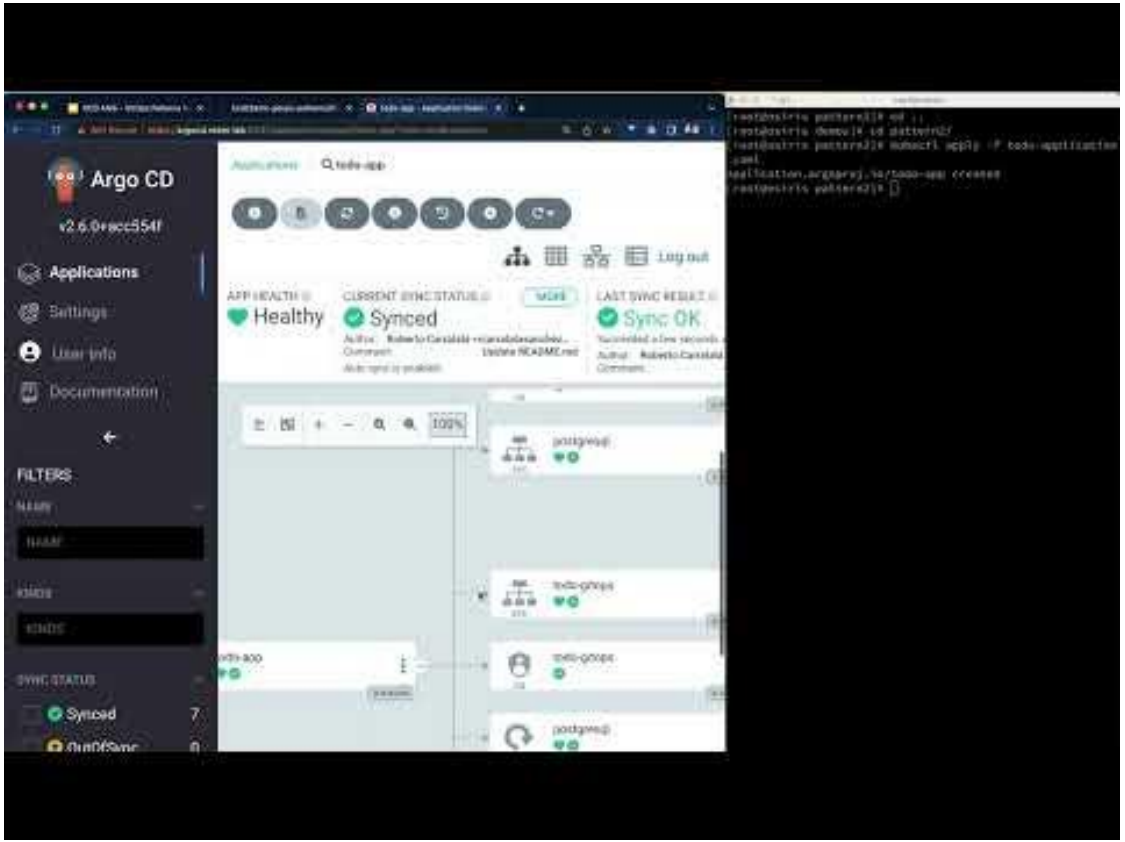
```yaml
metadata:
  annotations:
    argocd.argoproj.io/hook: PreSync
```

Pattern 2 Demo - Controlling Order within GitOps deployments



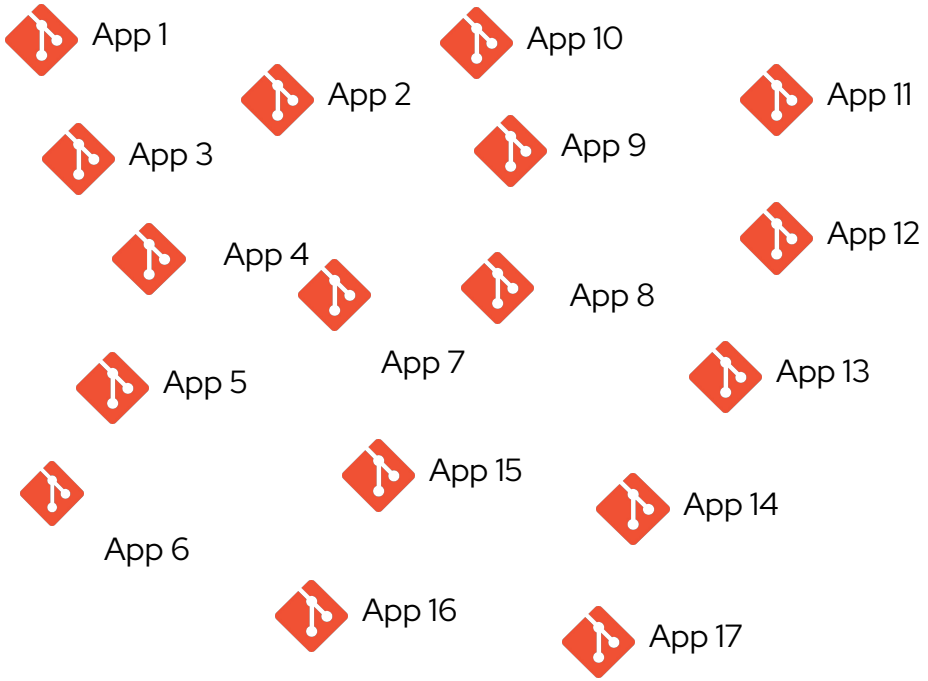


Pattern 2 Demo - Controlling Order within GitOps deployments



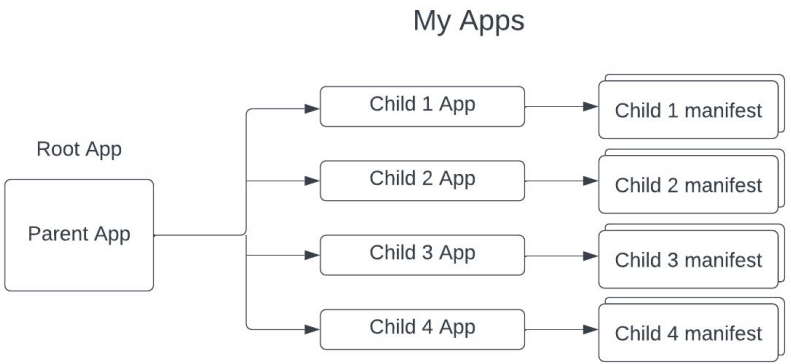
Pattern 3 - The GitOps Order Awakens!

What about deploying multiple related applications at once?

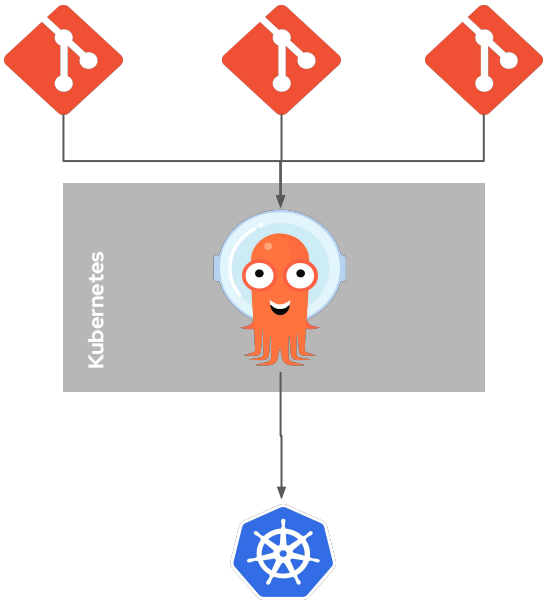


Pattern 3 - The GitOps Order Awakens!
ArgoCD App of Apps

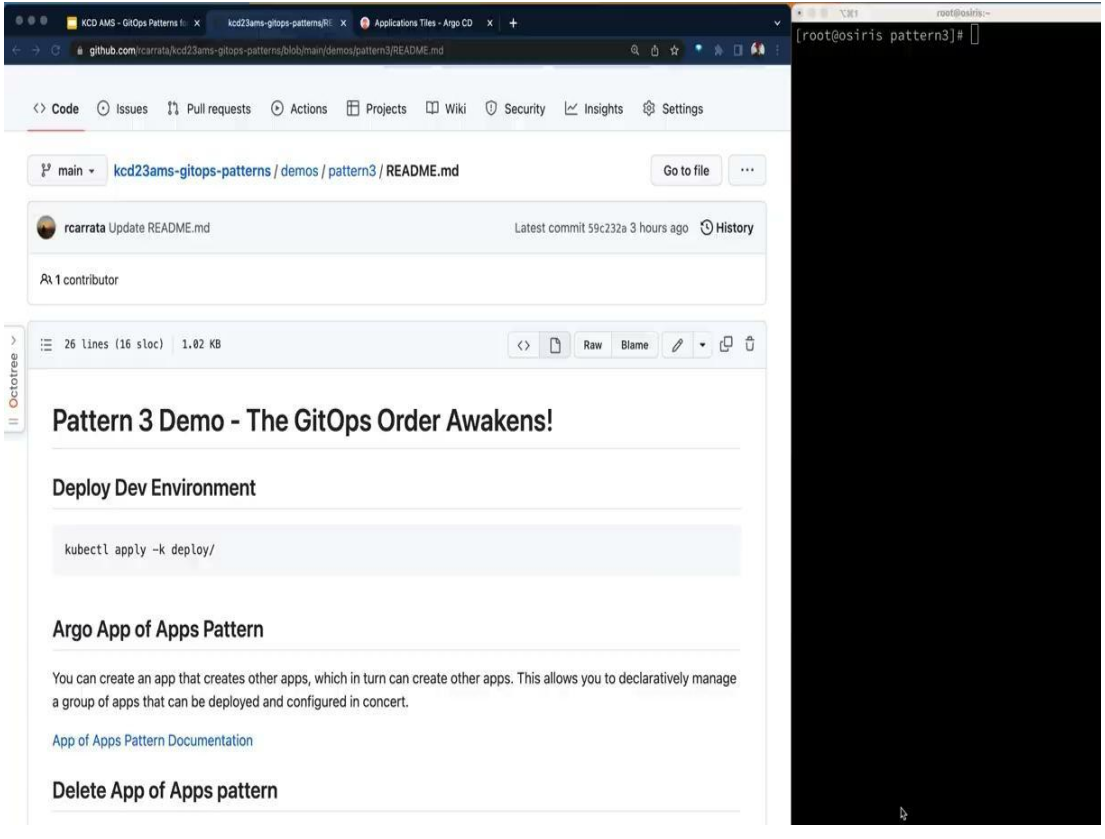
- Allows us to define a **root Argo CD application** that will itself define and sync multiple child applications.
- Root App points to a **folder in Git**, where we store the application manifests that define each application we want to create and deploy.
- This way we are able to **declare all our applications** inside a **single YAML manifest**



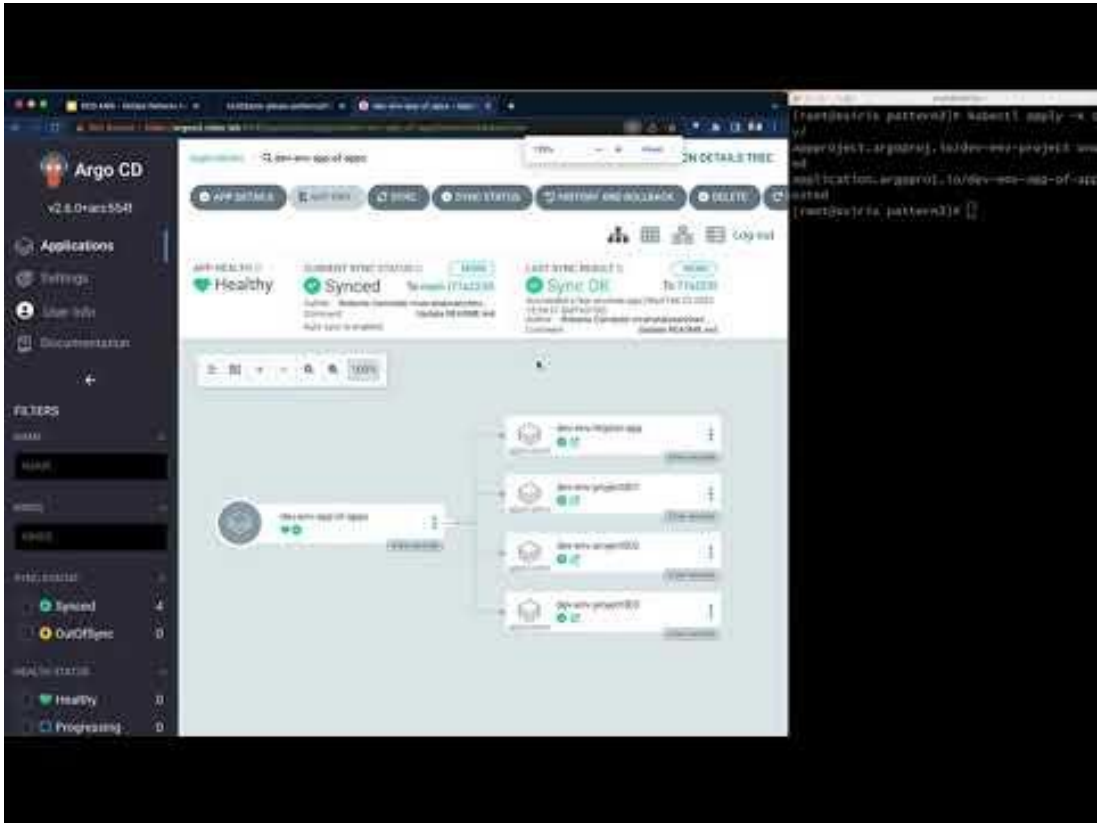
Pattern 3 Demo - The GitOps Order Awakens!



Pattern 3 Demo - The GitOps Order Awakens!



Pattern 3 Demo - The GitOps Order Awakens!

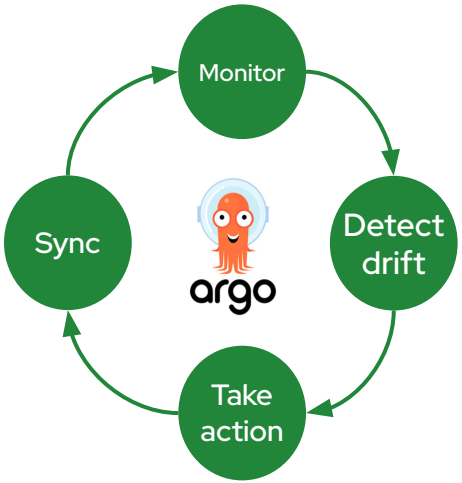


Do I need to define each ArgoCD App in the App of Apps?

Is there any way better to manage our Applications at scale?

Pattern 4 - Managing GitOps Apps at scale
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



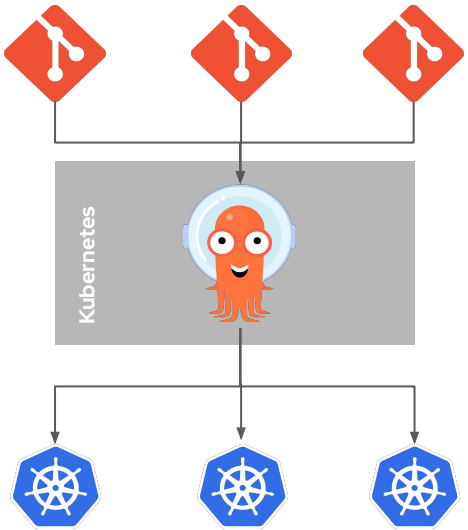
Pattern 4 - Managing GitOps Apps at scale

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.

Pattern 4 Demo - Managing GitOps Apps at scale

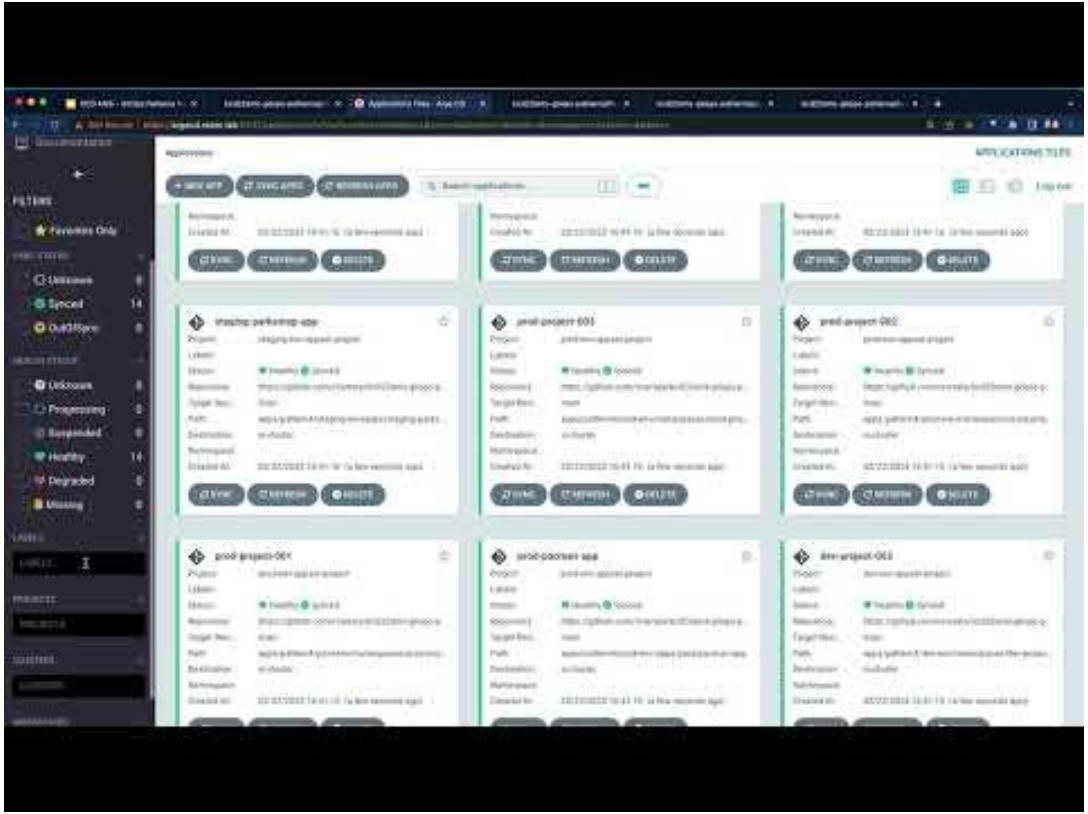
Argo CD ApplicationSet Generators



Central Hub (Push)

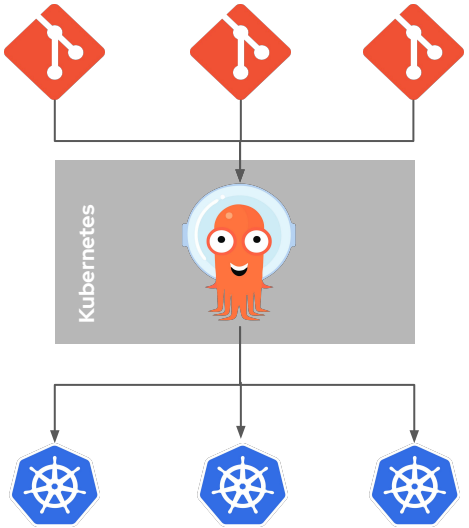
Pattern 4 Demo - Managing GitOps Apps at scale

Argo CD ApplicationSet Generators



Pattern 5 - GitOps Cluster Deployment Strategies

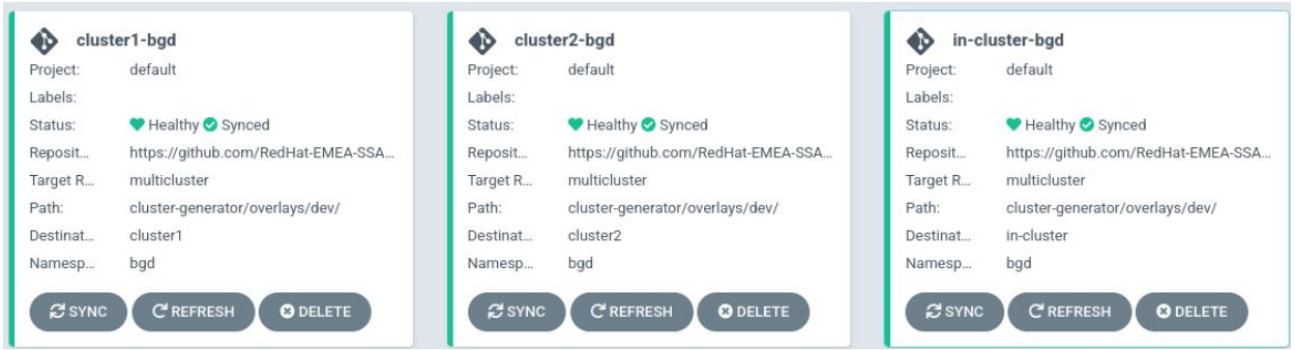
- **List generator:** Generates parameters based on a fixed list of cluster name/URL values, as seen in the example above.
- **Cluster generator:** Automatically generates cluster parameters based on the clusters that are defined within Argo CD.
- **Git generator:** Generates parameters based on files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

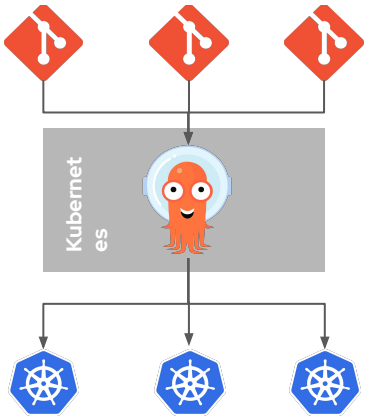


Central Hub (Push)

Pattern 5 - GitOps Cluster Deployment Strategies

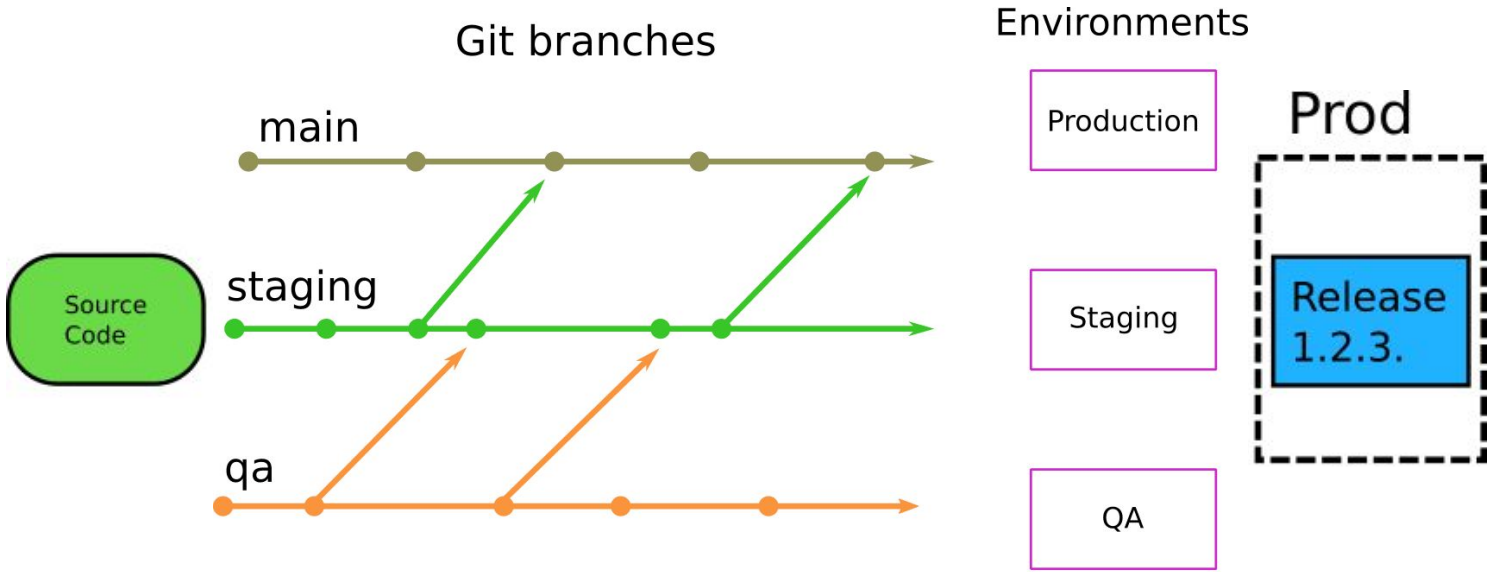
```yaml
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: welcome-app-appset
  namespace: openshift-gitops
spec:
  generators:
    - clusters: {}
  template:
    metadata:
      name: "{{name}}-welcome-app"
    spec:
      project: default
      syncPolicy:
        automated:
          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
```





Central Hub (Push)

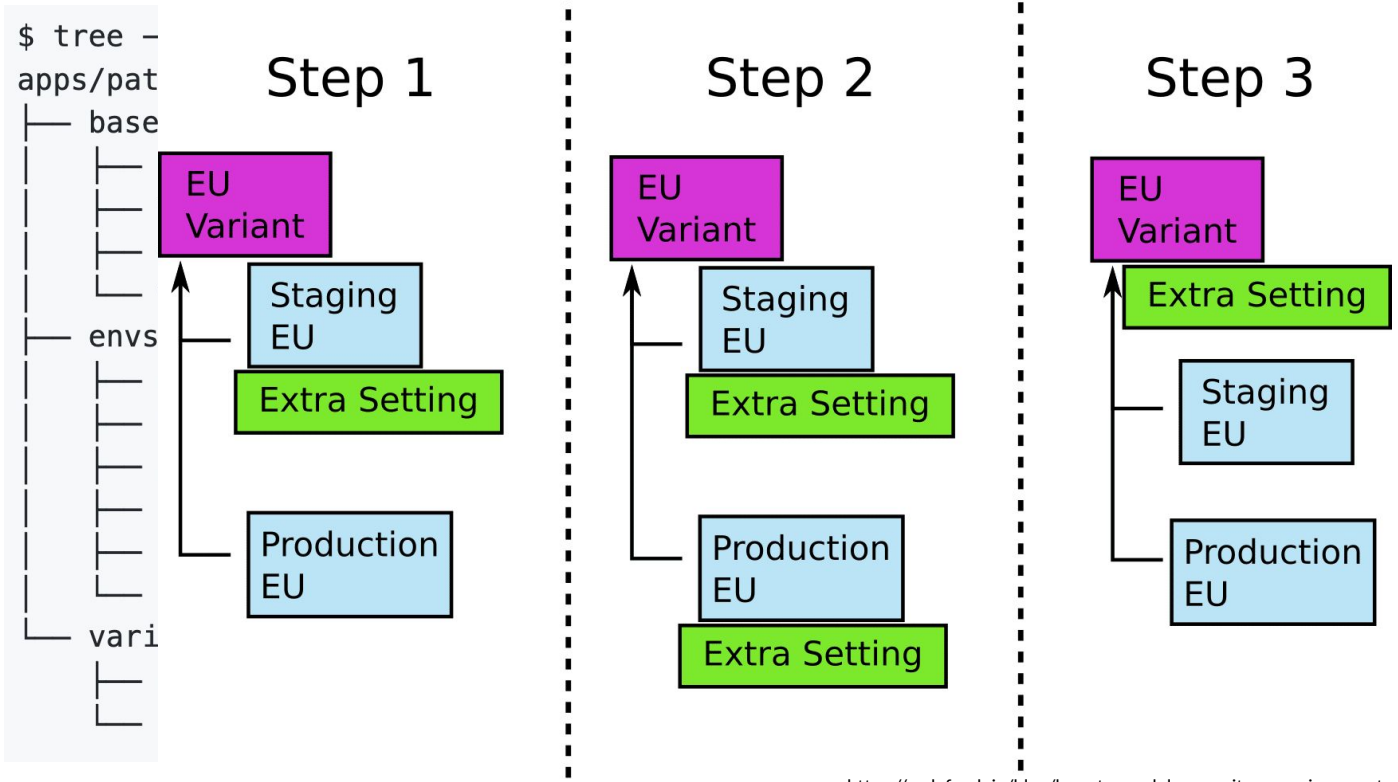
Pattern 6 - Promotion between GitOps environments



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

Pattern 6 - Promotion between GitOps environments

```
$ tree -
apps/pat
├── base
│   ├──
│   ├──
│   ├──
│   └──
├── envs
│   ├──
│   ├──
│   ├──
│   ├──
│   ├──
│   └──
└── vari
    ├──
    └──
```



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

Pattern 6 - Promotion between GitOps environments

main kcd23ams

rcarrata added patterr

1 contributor

10 lines (10 sloc) | 191

```
1  apiVersion: apps/v
2  kind: Deployment
3  metadata:
4    name: welcome-ap
5  spec:
6    template:
7      spec:
8        containers:
9          - name: we
10           image: q
```

main kcd23ams-gitops-patterns / apps / pattern6 / envs / prod-us / version.yaml

rcarrata added pattern6

1 contributor

10 lines (10 sloc) | 191 Bytes

```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: welcome-app
5  spec:
6    template:
7      spec:
8        containers:
9          - name: welcome-app
10           image: quay.io/rcarrata/welcome-app:1.0
```

Q/A

Thanks for Attending!