Red Hat Summit

Connect

GitOps Implementation and Takeaways in Multi-Cluster Environments

Mervan Ileri
Senior Architect

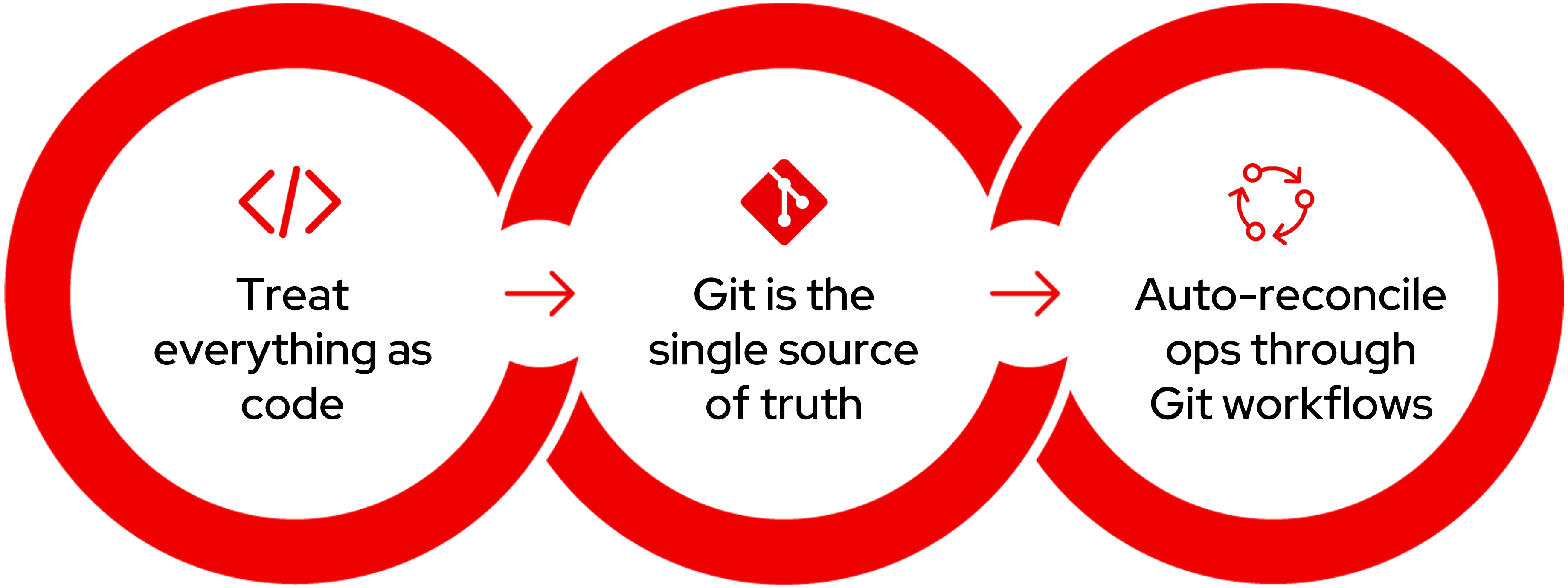
Red Hat

GitOps Overview

What is GitOps?

A developer-centric approach to Continuous Delivery and infrastructure operation.

GitOps unifies a collection of different topics in automation, application delivery, infrastructure management, and security.



Treat everything as code → Git is the single source of truth → Auto-reconcile ops through Git workflows

GitOps Principles

Declarative

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

Versioned and Immutable

Desired state is **stored** in a way that enforces immutability, versioning and retains a complete version history.

Pulled Automatically

Software agents automatically pull the desired state declarations from the source.

Continuously Reconciled

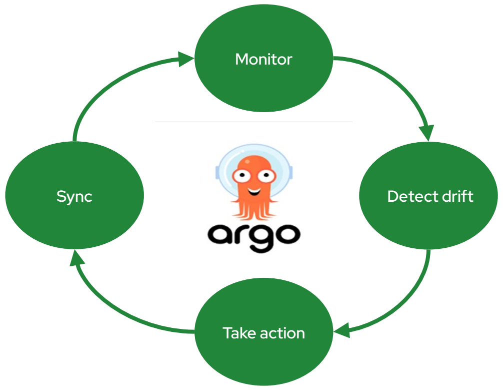
Software agents **continuously** observe actual system state and **attempt to apply** the desired state on drift.

https://opengitops.dev

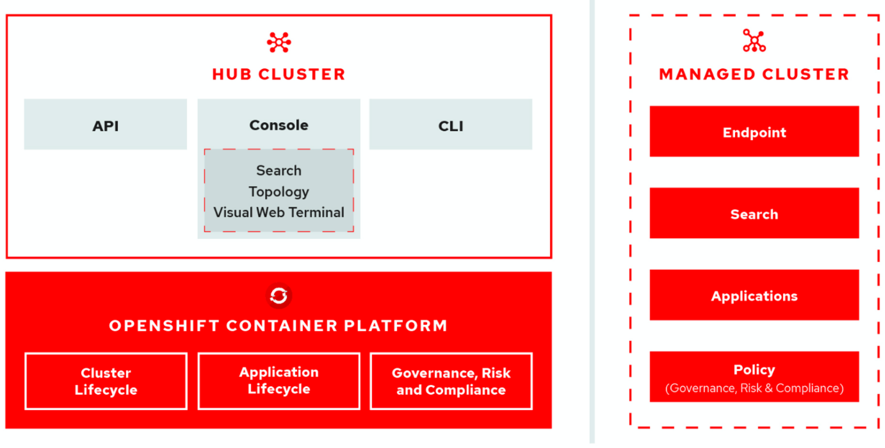
OpenShift GitOps and ACM Overview

Declarative GitOps for multi-cluster continuous delivery

OpenShift GitOps and Advanced Cluster Management (ACM) Overview



- Automatically syncs configuration from Git
- 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



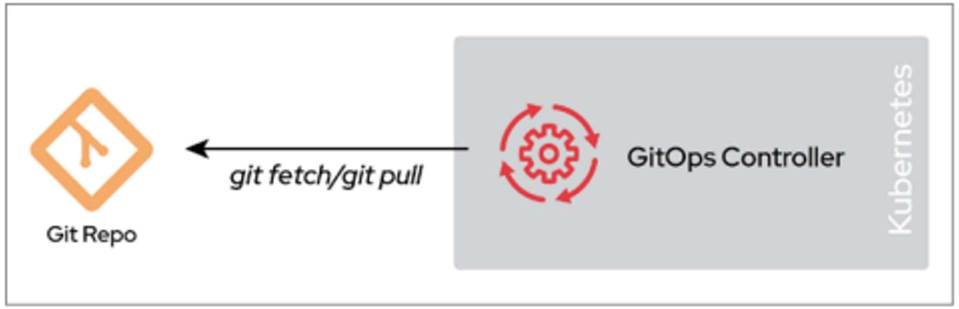
- Multicluster lifecycle management
- Policy driven governance, risk and compliance
- Advanced application lifecycle management
- Multicluster Observability and Search for health and optimization
- Multicluster networking for interconnecting apps

GitOps Repository Conventions

There is no one standard that works for everyone…

GitOps with Monorepo

Repository Considerations

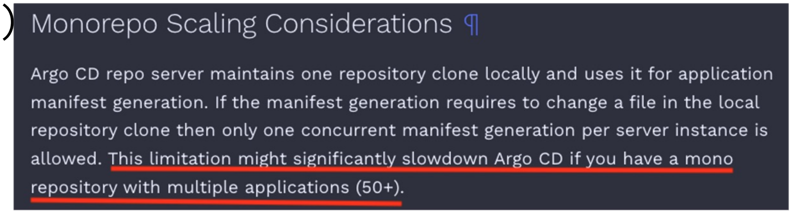


Advantages

- Provides a **central location** for configuration changes
- This simplicity enables straightforward **Git workflows** that will be **centrally visible to the entire organization**, making for a smoother and clearer approval process and merging

Disadvantages

- Scalability >> Increase complexity >> management
- Performance (Controller)
- ....



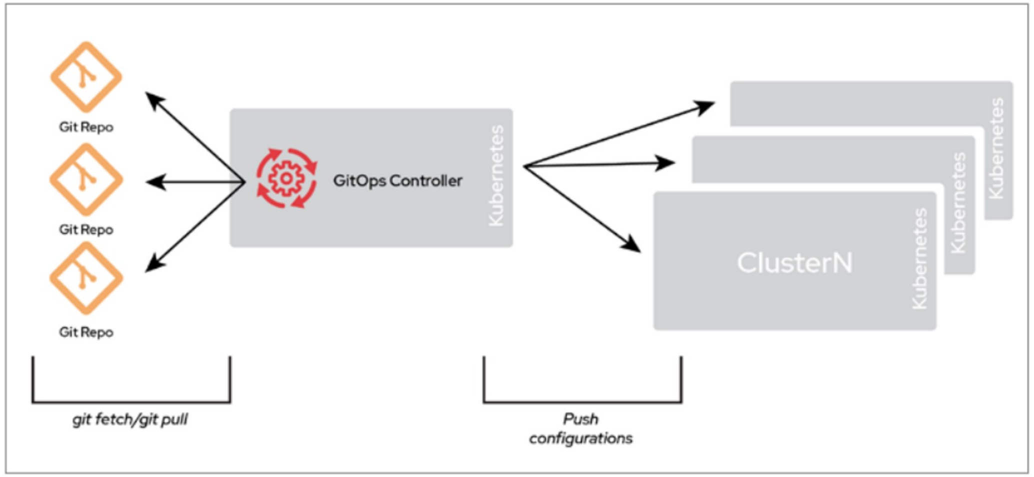
Monorepo Scaling Considerations ¶

Argo CD repo server maintains one repository clone locally and uses it for application manifest generation. If the manifest generation requires to change a file in the local repository clone then only one concurrent manifest generation per server instance is allowed. This limitation might significantly slowdown Argo CD if you have a mono repository with multiple applications (50+).

See ArgoCD Monorepo Scaling Considerations https://argo-cd.readthedocs.io/en/stable/operator-manual/high_availability/#monorepo-scaling-considerations

GitOps with Multirepo

Repository Considerations



Advantages

- Allows **separating concerns** between different departments of an organization: a **repository** for the **security team**, a repository for the **operations team**, and one or more repositories for **application teams**.
- Allows **multitenancy**, where you have one repository per application.
- **Catalog** of what needs to go into an environment or cluster.

Disadvantages

- It can become hard to manage, but it scales very well and is flexible enough to fit almost any organization.

GitOps with Monorepo vs Multirepo

There is not a unique “right” way. There are design considerations.

- Maintain infrastructure, platform and applications in **different repositories**
- **Reflect organization’s operational model as different repos.** (e.g. a repo per operations team, per system, per application)
  - **Maintain a base repo system type**
  - **Separate artifacts definitions and base configuration** as stand-alone base
  - **Separate artifacts cluster or environment specific configurations** in stand alone or **environments repositories** (based on use cases)

Considerations

- **Monorepo** strategy may have an impact in the **performance and scalability of GitOps controllers***
- Monorepo can be challenging to manage access permissions to repo

GitOps Repository Structure

There is not a unique “right” way. There are best practices.

Considerations

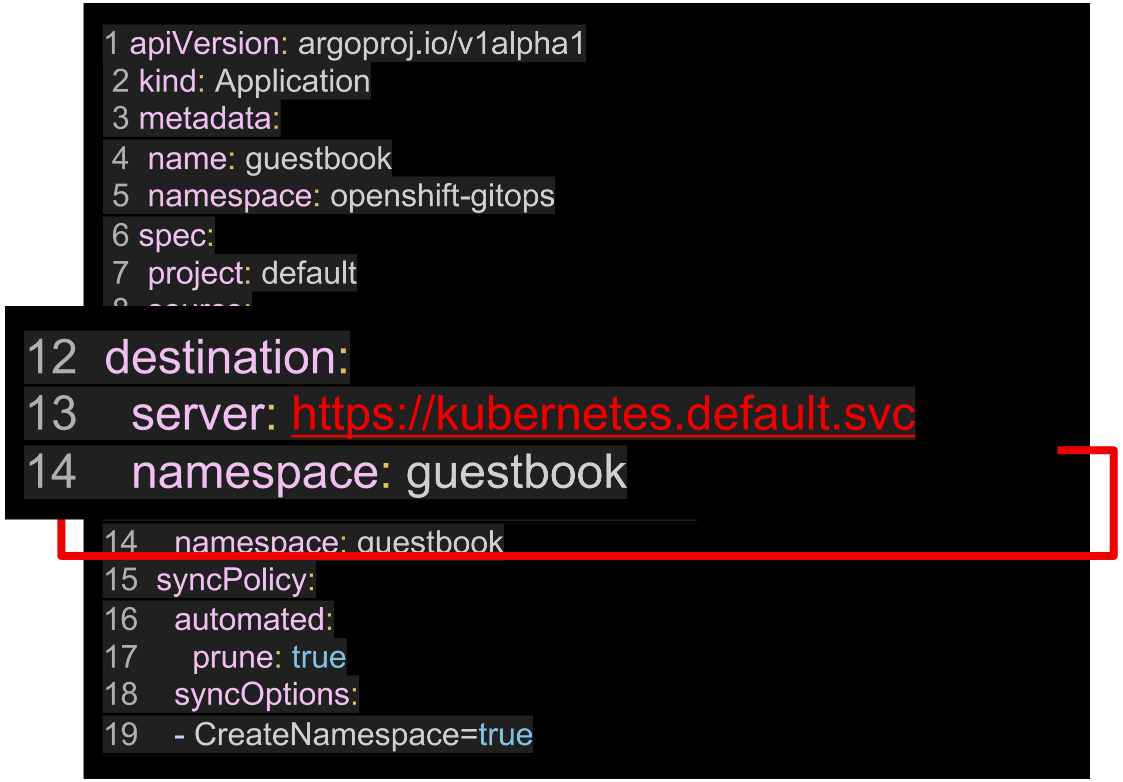
- **D**on’t **R**epeat **Y**ourself(**DRY**) principle
- **Structure generic** enough to deploy to many clusters
- “**Full DevOps**”: (both OCP administrators and OCP developers) are working together in the release process.
- Favor “**environment-per-folder**” approach, **not** “environment-per-branch” approach
- **Use tools to manage manifests**: Kustomize, Helm, Kustomize+Helm

Argo CD Concepts

Overview of Argo CD APIs

Applications

- Represents a deployed *application* instance in an environment.
- It is defined by two key pieces of information:
  - A *source* reference to the desired state in Git (repository, revision, path, environment)
  - A *destination* reference to the target cluster and namespace
- It includes options via *sync policy* to manage the synchronization between the desired state (source) and target state (destination)

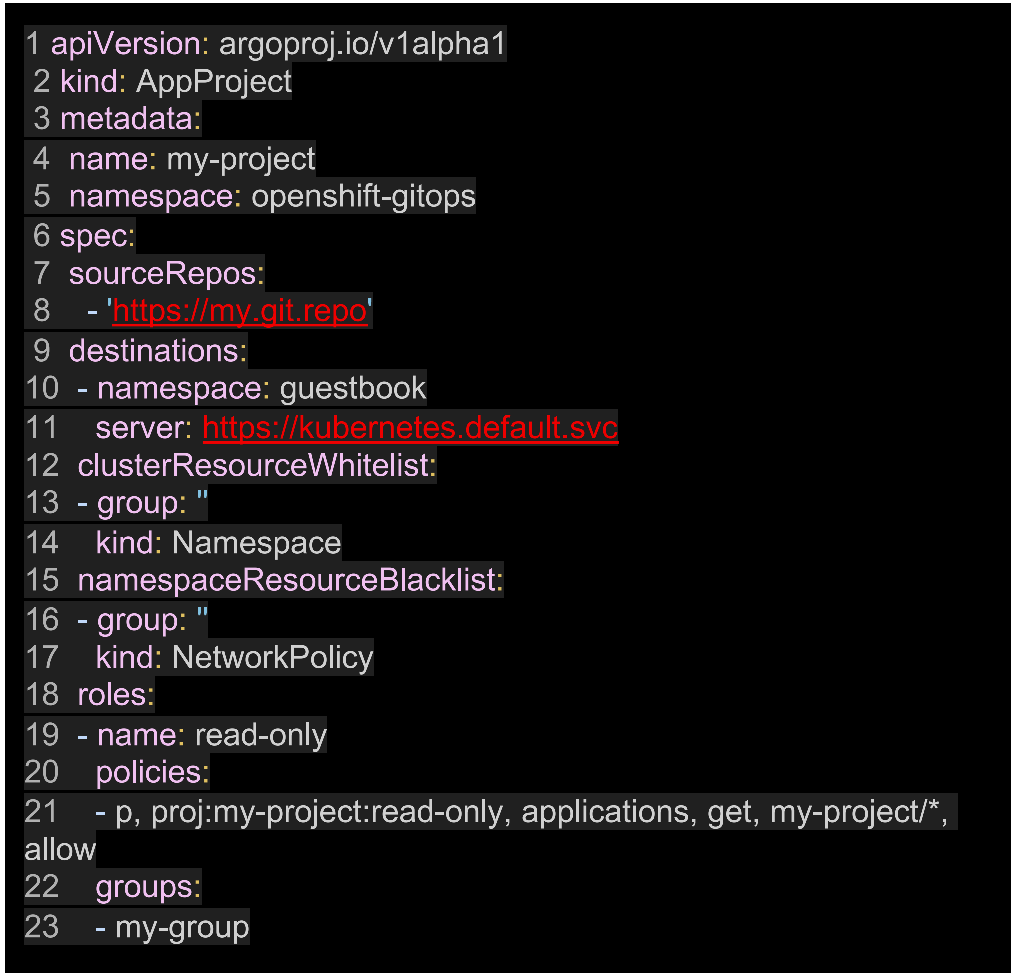


```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: guestbook
  namespace: openshift-gitops
spec:
  project: default
  source:
  destination:
    server: https://kubernetes.default.svc
    namespace: guestbook
    namespace: guestbook
  syncPolicy:
    automated:
      prune: true
    syncOptions:
    - CreateNamespace=true
```

https://argo-cd.readthedocs.io/en/stable/operator-manual/declarative-setup/#applications

Projects

- As a logical grouping of applications, AppProject holds three key pieces of information:
    - *sourceRepos* reference the repositories allowed to pull manifests from.
    - *destinations* reference to clusters and namespaces that applications within the project can deploy into
    - *roles* list of entities with definitions of their access to resources within the project.

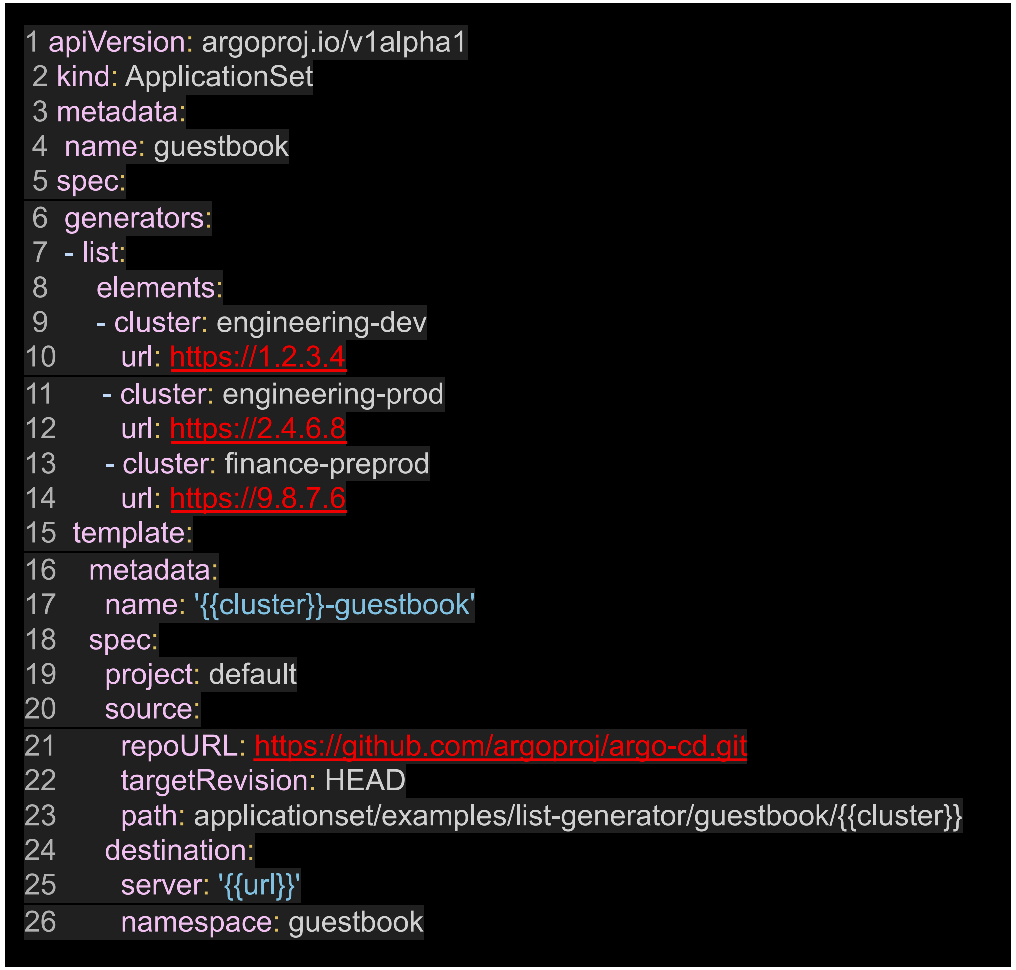


```yaml
apiVersion: argoproj.io/v1alpha1
kind: AppProject
metadata:
  name: my-project
  namespace: openshift-gitops
spec:
  sourceRepos:
    - 'https://my.git.repo'
  destinations:
  - namespace: guestbook
    server: https://kubernetes.default.svc
  clusterResourceWhitelist:
  - group: ''
    kind: Namespace
  namespaceResourceBlacklist:
  - group: ''
    kind: NetworkPolicy
  roles:
  - name: read-only
    policies:
    - p, proj:my-project:read-only, applications, get, my-project/*, allow
    groups:
    - my-group
```

https://argo-cd.readthedocs.io/en/stable/operator-manual/declarative-setup/#projects

ApplicationSet

- Dynamically generate a set of Application resources using templating with inputs provided by Generators
- Generators provide ways to parameterize the creation of Applications
- Different generator options: List Generator, Cluster Generator, Git Generator, Matrix Generator, Merge Generator

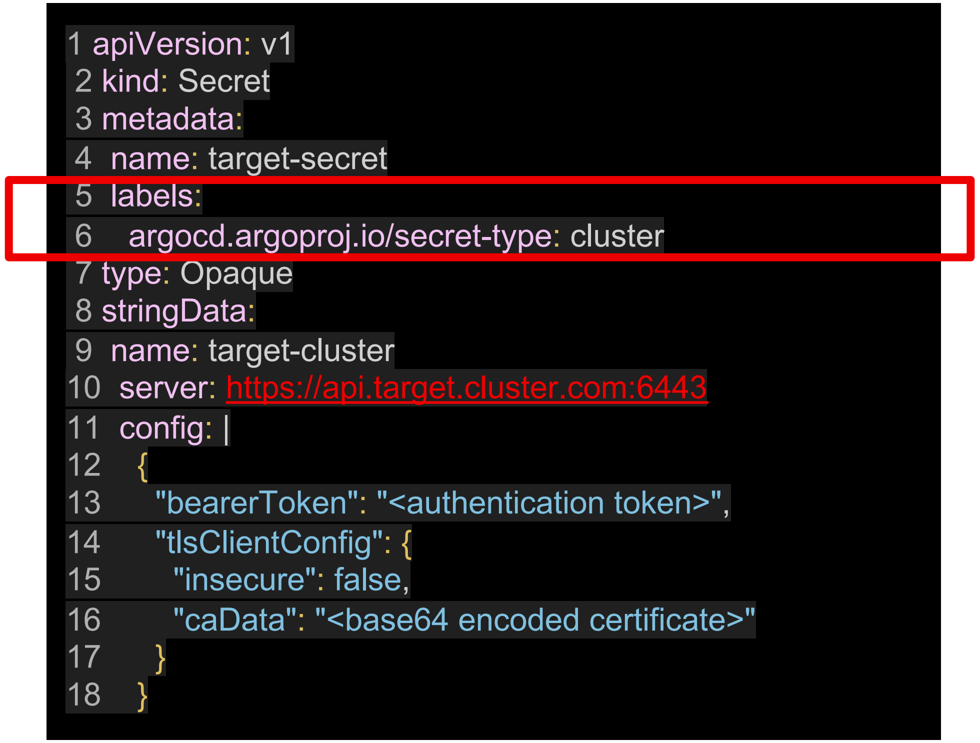


```yaml
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: guestbook
spec:
  generators:
  - list:
      elements:
      - cluster: engineering-dev
        url: https://1.2.3.4
      - cluster: engineering-prod
        url: https://2.4.6.8
      - cluster: finance-preprod
        url: https://9.8.7.6
  template:
    metadata:
      name: '{{cluster}}-guestbook'
    spec:
      project: default
      source:
        repoURL: https://github.com/argoproj/argo-cd.git
        targetRevision: HEAD
        path: applicationset/examples/list-generator/guestbook/{{cluster}}
      destination:
        server: '{{url}}'
        namespace: guestbook
```

https://argo-cd.readthedocs.io/en/stable/operator-manual/applicationset/

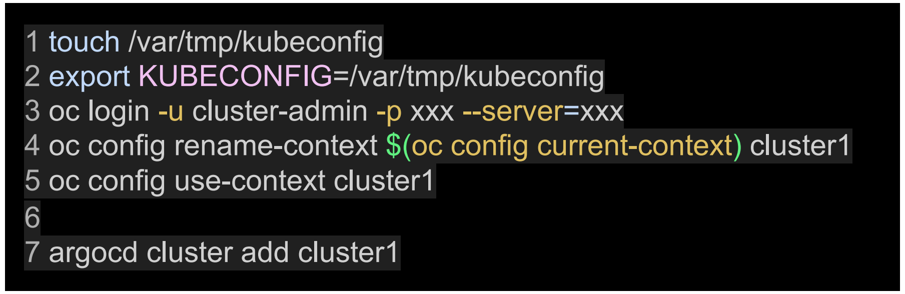
Defining Clusters in ArgoCD

Defining Cluster Destinations



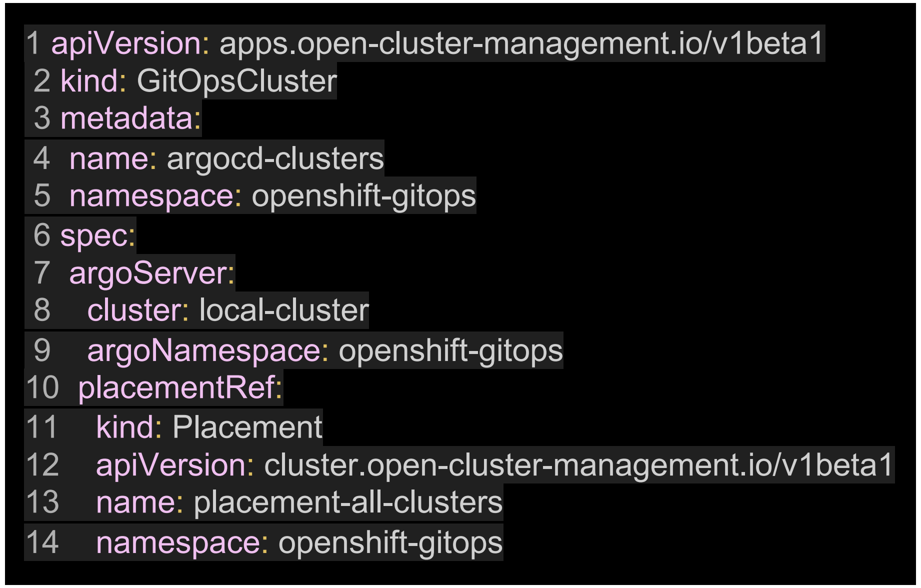
```yaml
apiVersion: v1
kind: Secret
metadata:
  name: target-secret
  labels:
    argocd.argoproj.io/secret-type: cluster
type: Opaque
stringData:
  name: target-cluster
  server: https://api.target.cluster.com:6443
  config: |
    {
      "bearerToken": "<authentication token>",
      "tlsClientConfig": {
        "insecure": false,
        "caData": "<base64 encoded certificate>"
      }
    }
```

or



```
touch /var/tmp/kubeconfig
export KUBECONFIG=/var/tmp/kubeconfig
oc login -u cluster-admin -p xxx --server=xxx
oc config rename-context $(oc config current-context) cluster1
oc config use-context cluster1

argocd cluster add cluster1
```

vs



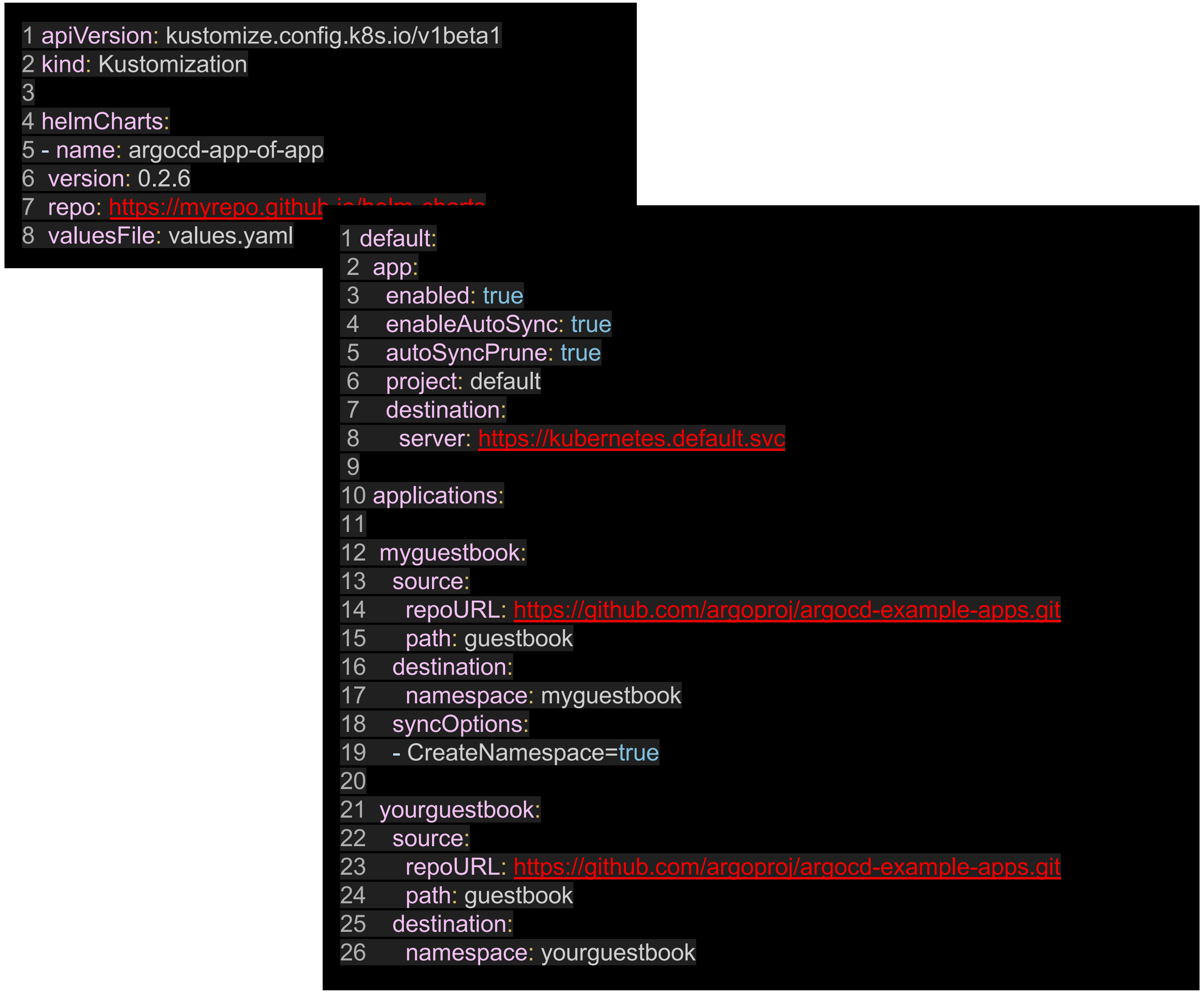
```yaml
apiVersion: apps.open-cluster-management.io/v1beta1
kind: GitOpsCluster
metadata:
  name: argocd-clusters
  namespace: openshift-gitops
spec:
  argoServer:
    cluster: local-cluster
    argoNamespace: openshift-gitops
  placementRef:
    kind: Placement
    apiVersion: cluster.open-cluster-management.io/v1beta1
    name: placement-all-clusters
    namespace: openshift-gitops
```

ACM's GitOpsCluster API simplifies the definition of clusters into ArgoCD with the assistance of placements…

App of Apps Pattern

App of Apps

- An application resource that creates other applications
- Manage a group of applications to be deployed in order.
- It is possible to deliver with a Helm chart which encapsulates application generation logic. Using it with kustomize enhances flexibility.



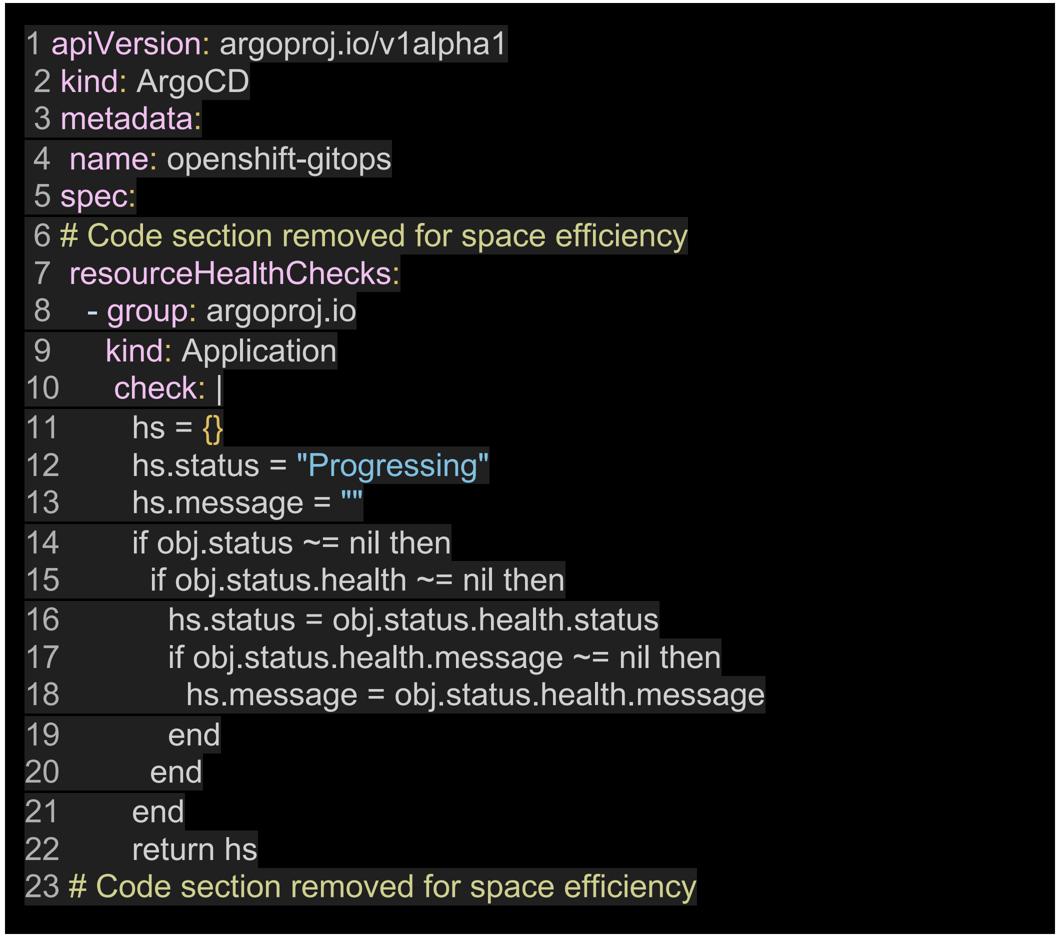
```yaml
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization

helmCharts:
- name: argocd-app-of-app
  version: 0.2.6
  repo: https://myrepo.github.io/helm-charts
  valuesFile: values.yaml
```

```yaml
default:
  app:
    enabled: true
    enableAutoSync: true
    autoSyncPrune: true
    project: default
    destination:
      server: https://kubernetes.default.svc

applications:

  myguestbook:
    source:
      repoURL: https://github.com/argoproj/argocd-example-apps.git
      path: guestbook
    destination:
      namespace: myguestbook
    syncOptions:
    - CreateNamespace=true

  yourguestbook:
    source:
      repoURL: https://github.com/argoproj/argocd-example-apps.git
      path: guestbook
    destination:
      namespace: yourguestbook
```

https://argo-cd.readthedocs.io/en/stable/operator-manual/declarative-setup/#app-of-apps

App of Apps

- Health assessment of argoproj.io/Application CRD removed in ArgoCD
- Restore health assessment if using sync waves for synchronization.

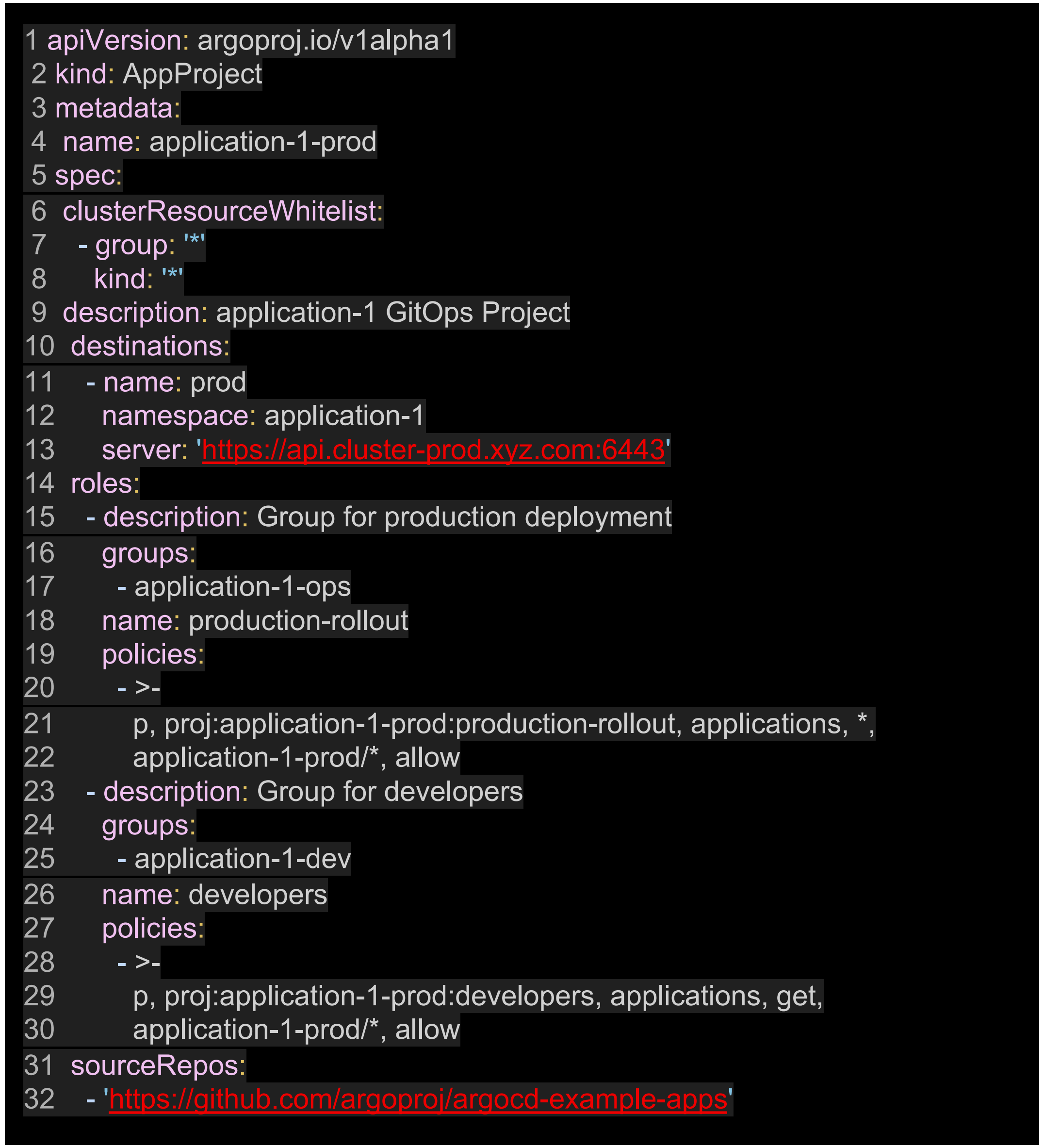


```
apiVersion: argoproj.io/v1alpha1
kind: ArgoCD
metadata:
  name: openshift-gitops
spec:
# Code section removed for space efficiency
  resourceHealthChecks:
    - group: argoproj.io
      kind: Application
       check: |
        hs = {}
        hs.status = "Progressing"
        hs.message = ""
        if obj.status ~= nil then
          if obj.status.health ~= nil then
            hs.status = obj.status.health.status
            if obj.status.health.message ~= nil then
              hs.message = obj.status.health.message
            end
          end
        end
        return hs
# Code section removed for space efficiency
```

https://argo-cd.readthedocs.io/en/stable/operator-manual/health/#argocd-app

Role Based Access Control

- Argo CD provides an internal RBAC for authorization to Argo resources
- It enables assigning *permissions* to *roles*, *users* or *groups*.
- Argo CD RBAC permissions and roles can be defined:
  - Globally scoped in the Argo CD CustomResource
  - Project scoped in the AppProject
- RBAC can be used to set team trust boundaries



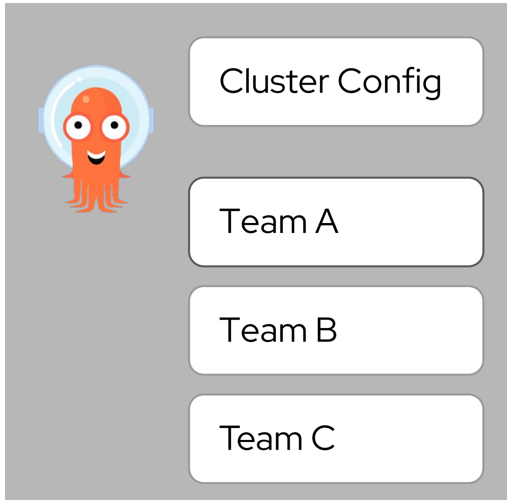
```yaml
apiVersion: argoproj.io/v1alpha1
kind: AppProject
metadata:
  name: application-1-prod
spec:
  clusterResourceWhitelist:
    - group: '*'
      kind: '*'
  description: application-1 GitOps Project
  destinations:
    - name: prod
      namespace: application-1
      server: 'https://api.cluster-prod.xyz.com:6443'
  roles:
    - description: Group for production deployment
      groups:
        - application-1-ops
      name: production-rollout
      policies:
        - >-
          p, proj:application-1-prod:production-rollout, applications, *,
          application-1-prod/*, allow
    - description: Group for developers
      groups:
        - application-1-dev
      name: developers
      policies:
        - >-
          p, proj:application-1-prod:developers, applications, get,
          application-1-prod/*, allow
  sourceRepos:
    - 'https://github.com/argoproj/argocd-example-apps'
```

https://argo-cd.readthedocs.io/en/stable/operator-manual/rbac/

Topology Models

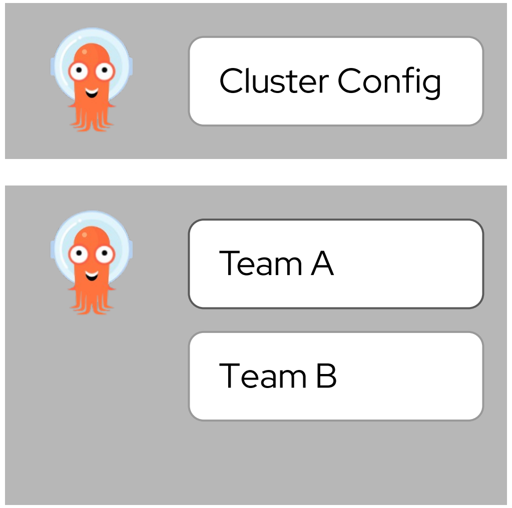
Selecting an Argo CD topology that is right for your organization

Logical Topologies



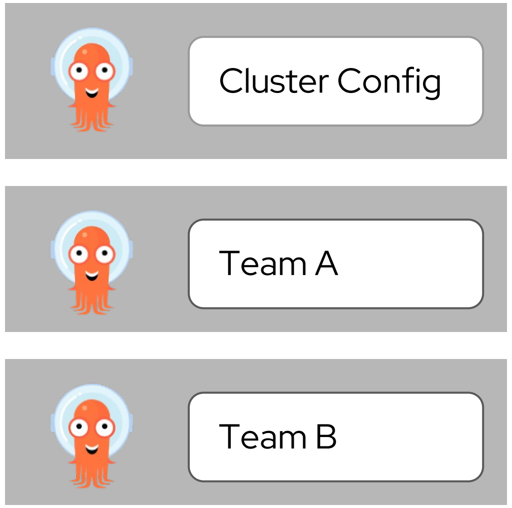
No Isolation

A single instance manages everything including cluster configuration and Applications



Partial Isolation

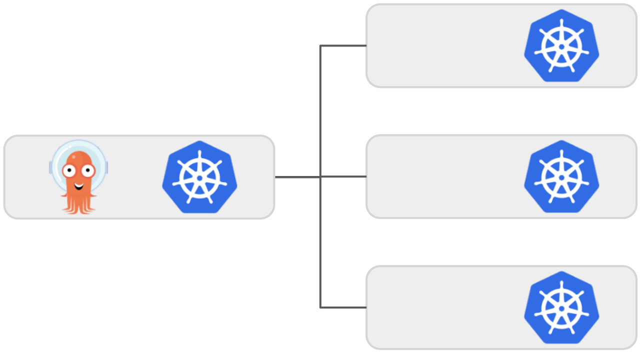
Separate instances for cluster configuration and applications. Both instances are cluster scoped



Maximum Isolation

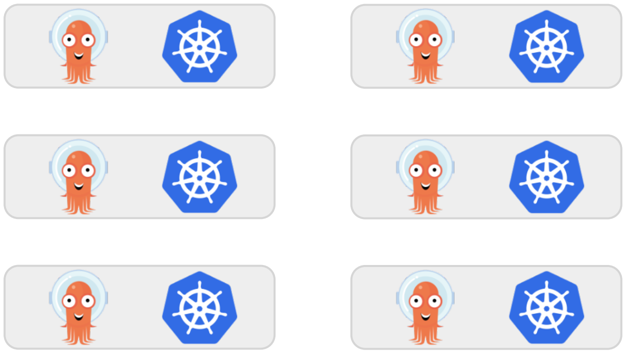
Separate instances across team trust boundaries. Team instances are namespace scoped, cluster config is cluster scoped

Physical Topologies



Centralized

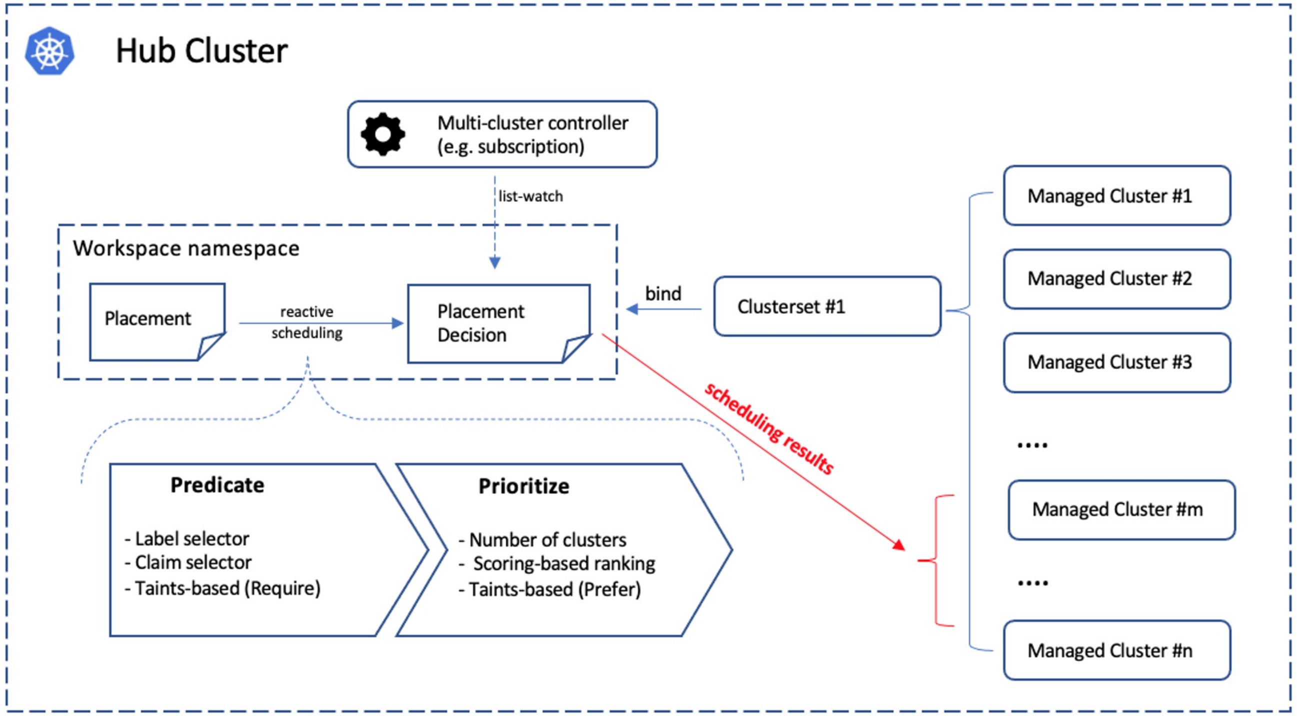
A centralized GitOps instance manages all clusters from a central Hub location

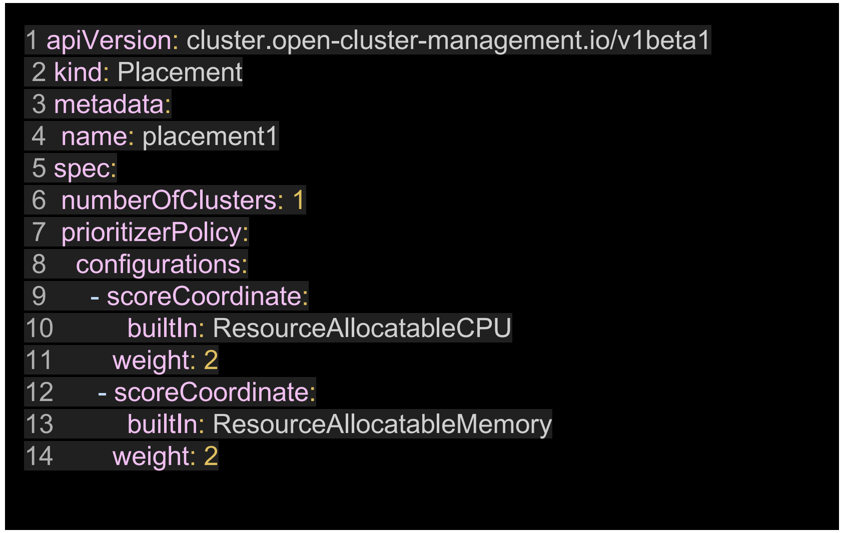


Distributed

Separate GitOps instances are deployed on each cluster and only manage resources on that cluster

Topology Models
ACM Placements



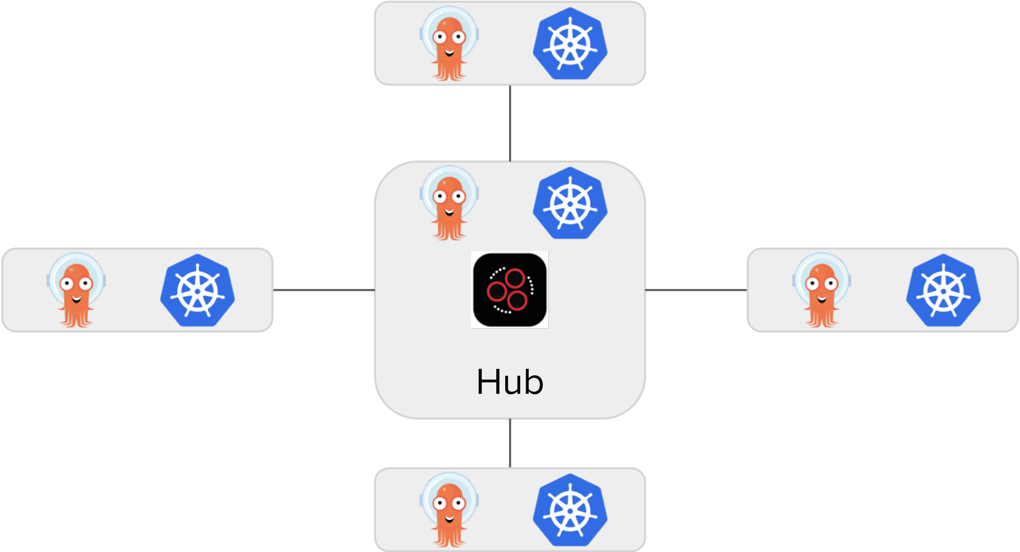


```yaml
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement1
spec:
  numberOfClusters: 1
  prioritizerPolicy:
    configurations:
      - scoreCoordinate:
          builtIn: ResourceAllocatableCPU
        weight: 2
      - scoreCoordinate:
          builtIn: ResourceAllocatableMemory
        weight: 2
```

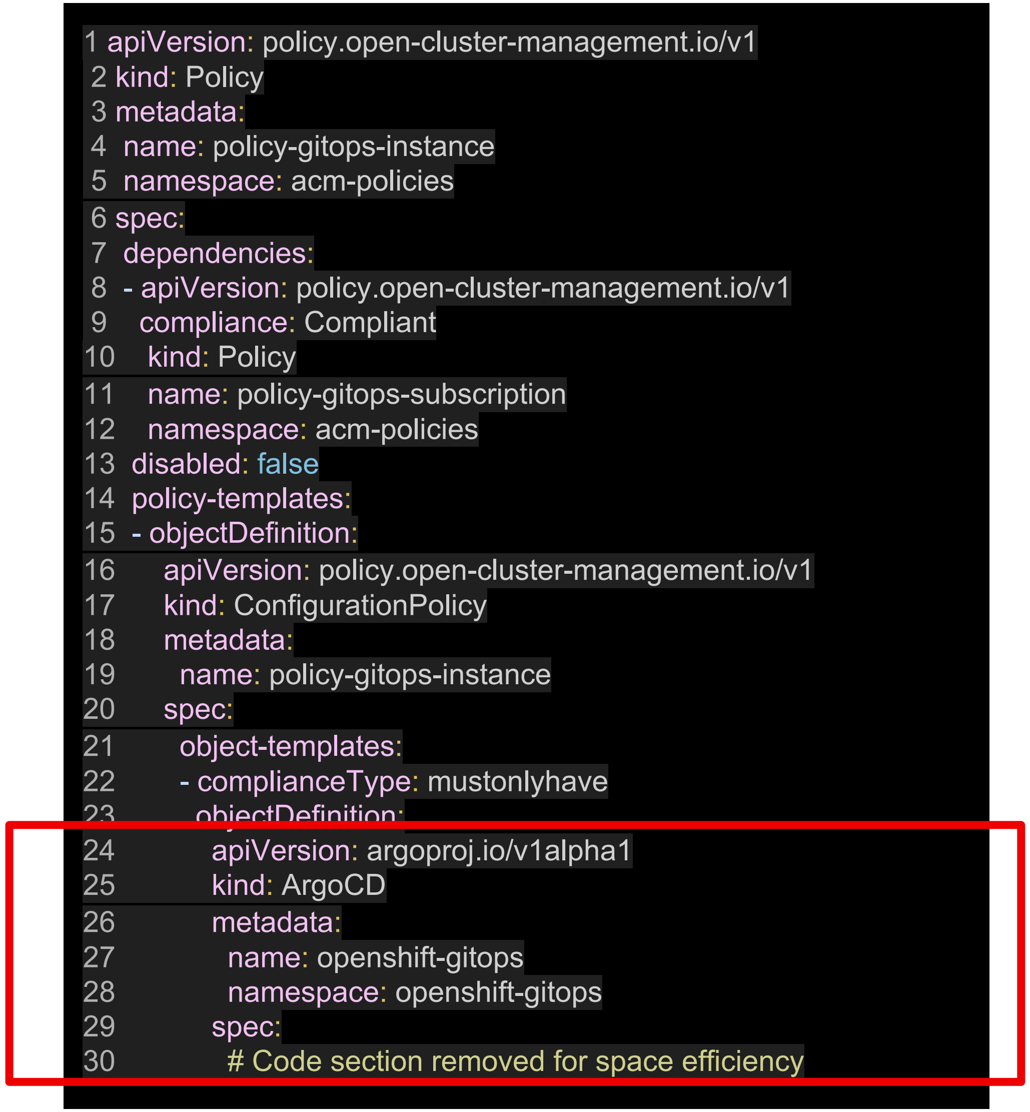
https://open-cluster-management.io/concepts/placement/

Physical Topologies

GitOps Control Plane



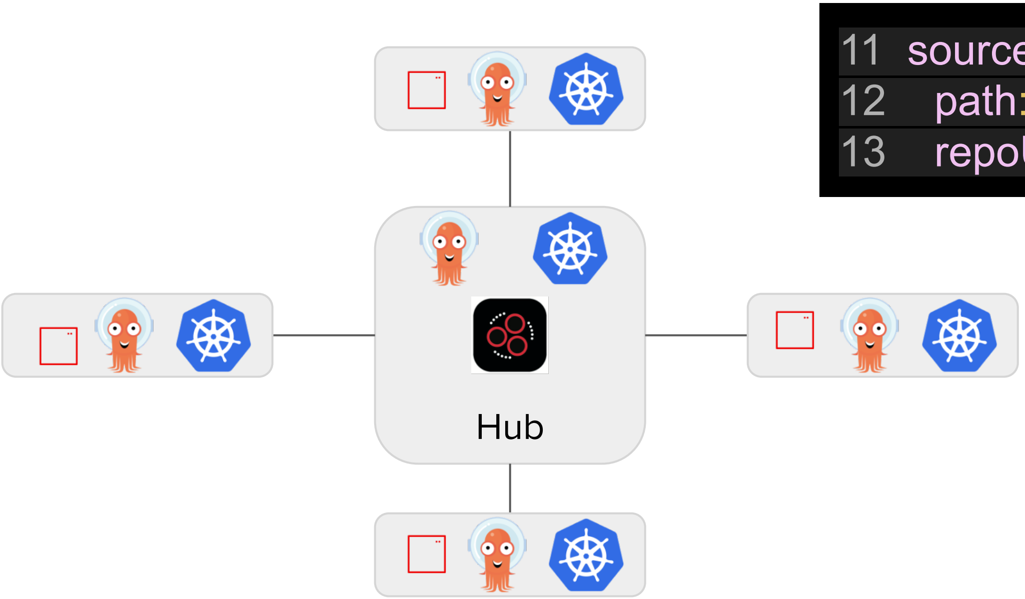
Distributed GitOps instances managed by a central control plane

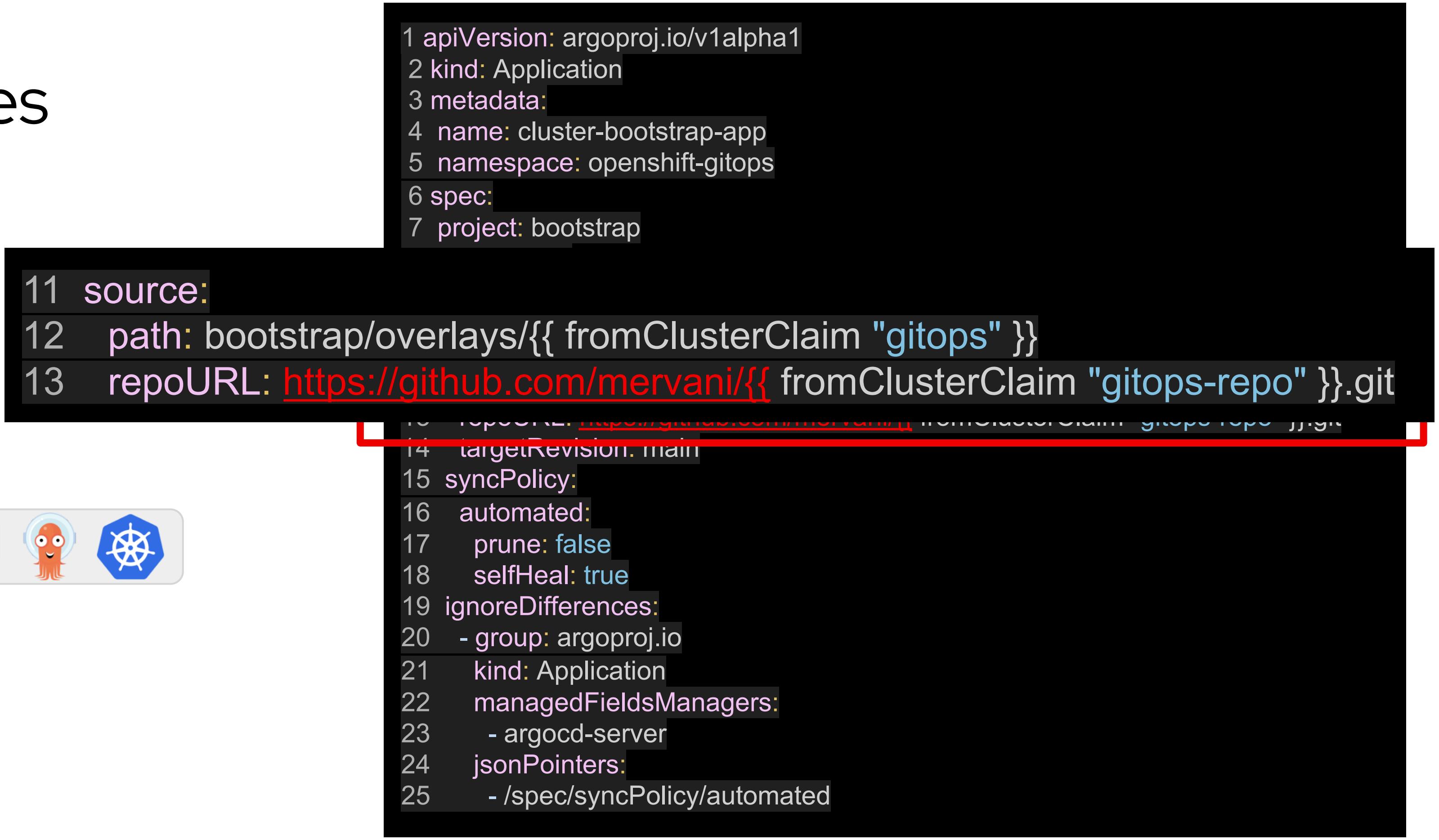


```yaml
apiVersion: policy.open-cluster-management.io/v1
kind: Policy
metadata:
  name: policy-gitops-instance
  namespace: acm-policies
spec:
  dependencies:
  - apiVersion: policy.open-cluster-management.io/v1
    compliance: Compliant
    kind: Policy
    name: policy-gitops-subscription
    namespace: acm-policies
  disabled: false
  policy-templates:
  - objectDefinition:
      apiVersion: policy.open-cluster-management.io/v1
      kind: ConfigurationPolicy
      metadata:
        name: policy-gitops-instance
      spec:
        object-templates:
        - complianceType: mustonlyhave
          objectDefinition:
            apiVersion: argoproj.io/v1alpha1
            kind: ArgoCD
            metadata:
              name: openshift-gitops
              namespace: openshift-gitops
            spec:
              # Code section removed for space efficiency
```

Physical Topologies

GitOps Control Plane





```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: cluster-bootstrap-app
  namespace: openshift-gitops
spec:
  project: bootstrap
  source:
    path: bootstrap/overlays/{{ fromClusterClaim "gitops" }}
    repoURL: https://github.com/mervani/{{ fromClusterClaim "gitops-repo" }}.git
    targetRevision: main
  syncPolicy:
    automated:
      prune: false
      selfHeal: true
  ignoreDifferences:
  - group: argoproj.io
    kind: Application
    managedFieldsManagers:
      - argocd-server
    jsonPointers:
      - /spec/syncPolicy/automated
```

Bootstrap applications can be distributed with policies in a dynamic manner with the help of ACM template functions

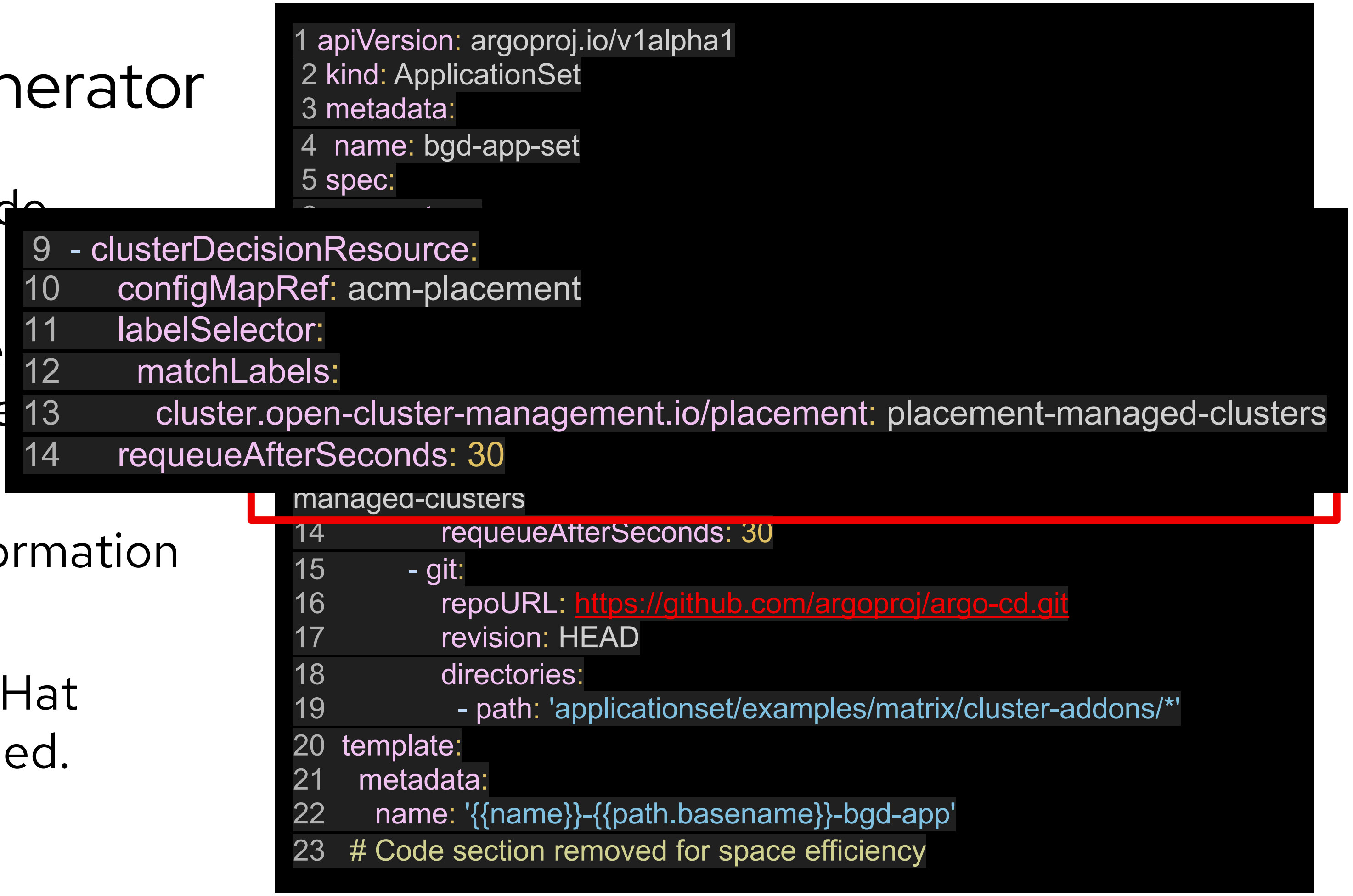
Topology Recommendations

- ★ Do not use the same GitOps instance for cluster configuration and application deployments
- ★ Do not have a separate GitOps instance for each individual application
- ★ Align the number of instances needed to support teams along trust boundaries
- ★ Use cluster scoped instances when there are more than a handful of namespaces being managed

ACM & Cluster Decision Resource Generator

Cluster Decision Resource Generator

- Takes the information from an outside source
- It uses this information to create the Applications provided by this outside source.
- The outside source provides the information via ConfigMap
- This is how the integration with Red Hat Advanced Cluster Manager is provided.



```
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: bgd-app-set
spec:
```

```
- clusterDecisionResource:
    configMapRef: acm-placement
    labelSelector:
      matchLabels:
        cluster.open-cluster-management.io/placement: placement-managed-clusters
    requeueAfterSeconds: 30
```

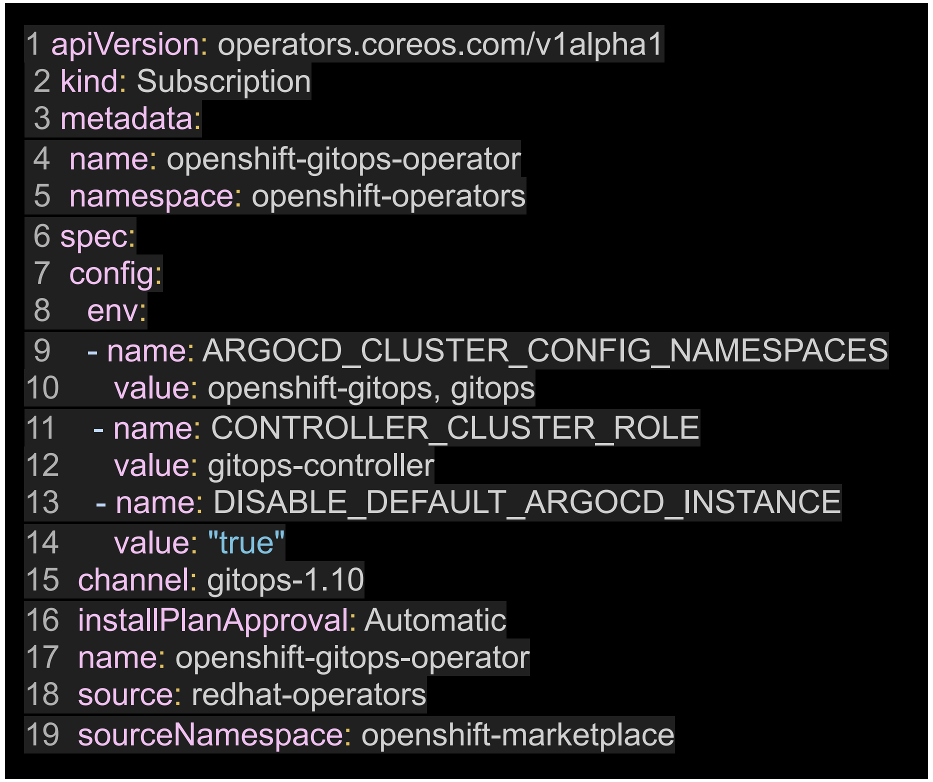
```
managed-clusters
          requeueAfterSeconds: 30
        - git:
          repoURL: https://github.com/argoproj/argo-cd.git
          revision: HEAD
          directories:
            - path: 'applicationset/examples/matrix/cluster-addons/*'
template:
  metadata:
    name: '{{name}}-{{path.basename}}-bgd-app'
  # Code section removed for space efficiency
```

https://argo-cd.readthedocs.io/en/stable/operator-manual/applicationset/Generators-Cluster-Decision-Resource/

Other Considerations

Environment Variables

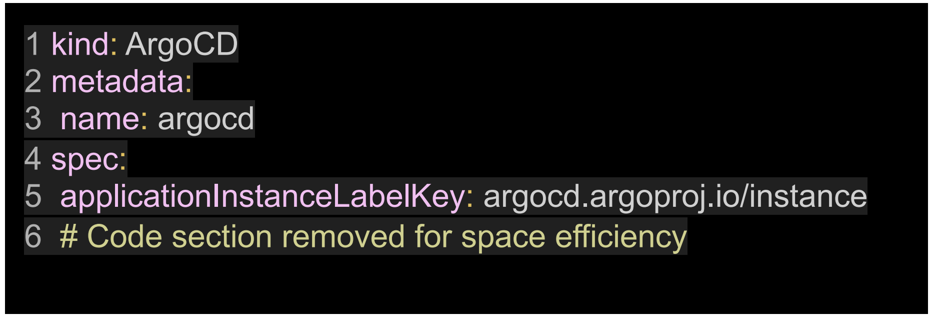
- ARGOCD_CLUSTER_CONFIG_NAMESPACES: List of namespaces of cluster-scoped Argo CD instances
- CONTROLLER_CLUSTER_ROLE: A common cluster role for all the managed namespaces. Can be customized to have a more strict role.
- DISABLE_DEFAULT_ARGOCD_INSTANCE: Disable the default installation of Argo CD in the 'openshift-gitops' namespace.



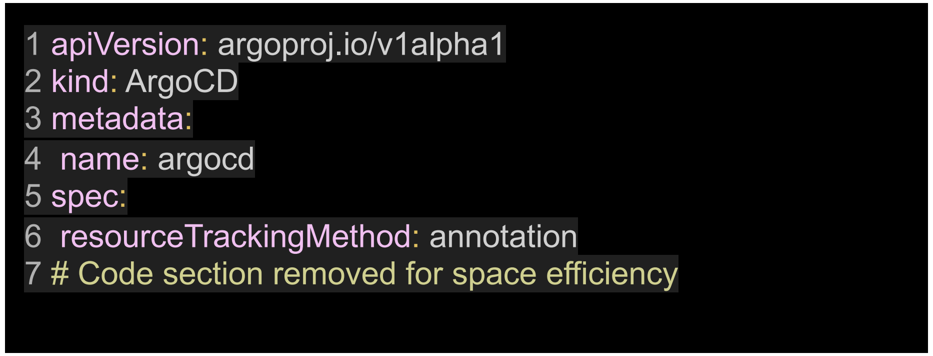
```yaml
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: openshift-gitops-operator
  namespace: openshift-operators
spec:
  config:
    env:
    - name: ARGOCD_CLUSTER_CONFIG_NAMESPACES
      value: openshift-gitops, gitops
    - name: CONTROLLER_CLUSTER_ROLE
      value: gitops-controller
    - name: DISABLE_DEFAULT_ARGOCD_INSTANCE
      value: "true"
  channel: gitops-1.10
  installPlanApproval: Automatic
  name: openshift-gitops-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace
```

Resource Tracking

- ArgoCD sets the ‘app.kubernetes.io/instance’ label to the application instance for resource tracking.
- Some external tools might write/append to this label.
- ArgoCD supports to use a custom label.
- Resource tracking can be changed so that an annotation is used.



```yaml
kind: ArgoCD
metadata:
  name: argocd
spec:
  applicationInstanceLabelKey: argocd.argoproj.io/instance
  # Code section removed for space efficiency
```



```yaml
apiVersion: argoproj.io/v1alpha1
kind: ArgoCD
metadata:
  name: argocd
spec:
  resourceTrackingMethod: annotation
# Code section removed for space efficiency
```

Red Hat Summit

Connect

Thank you

in linkedin.com/company/red-hat

f facebook.com/redhatinc

youtube.com/user/RedHatVideos

twitter.com/RedHat

Red Hat