

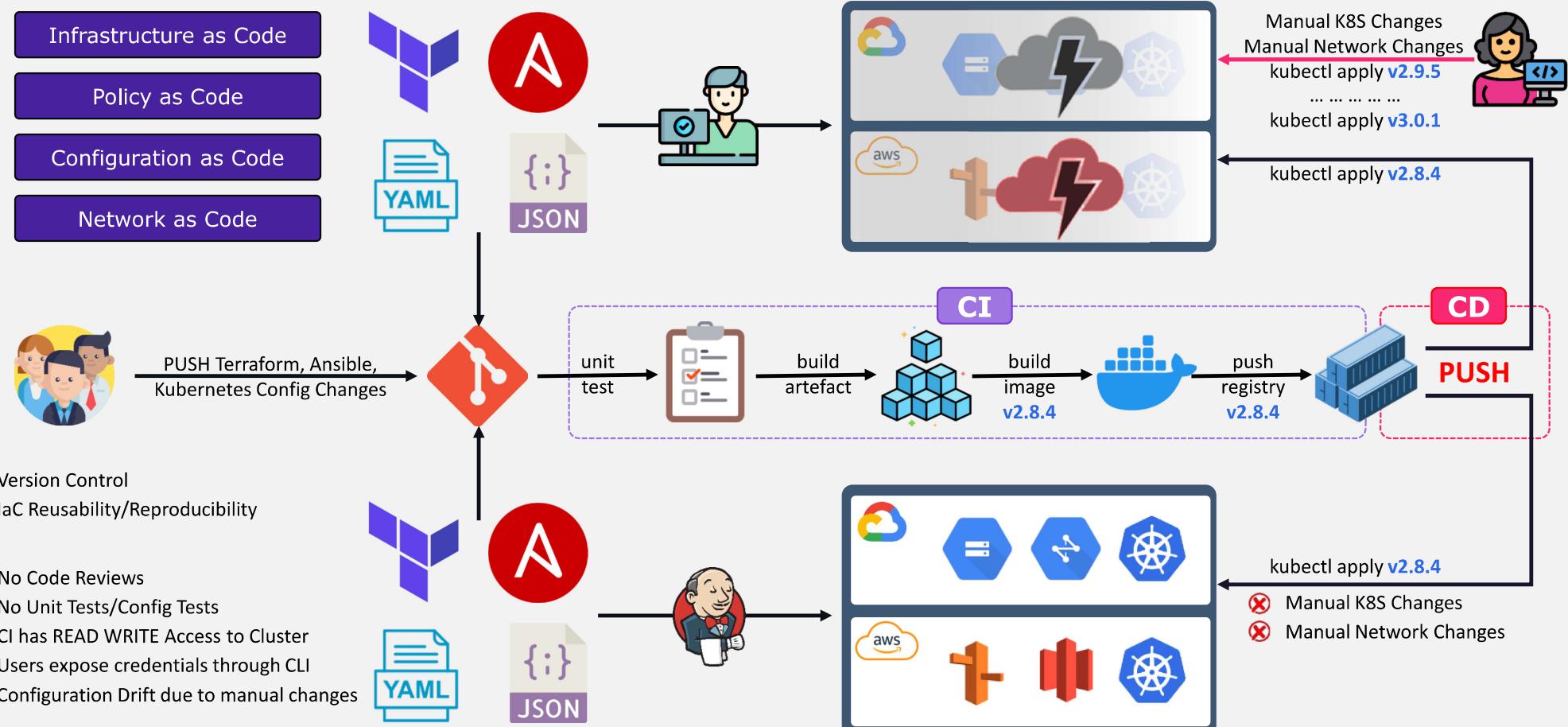


# KodeKloud

# Meeting Task Dash

## DevOps Team

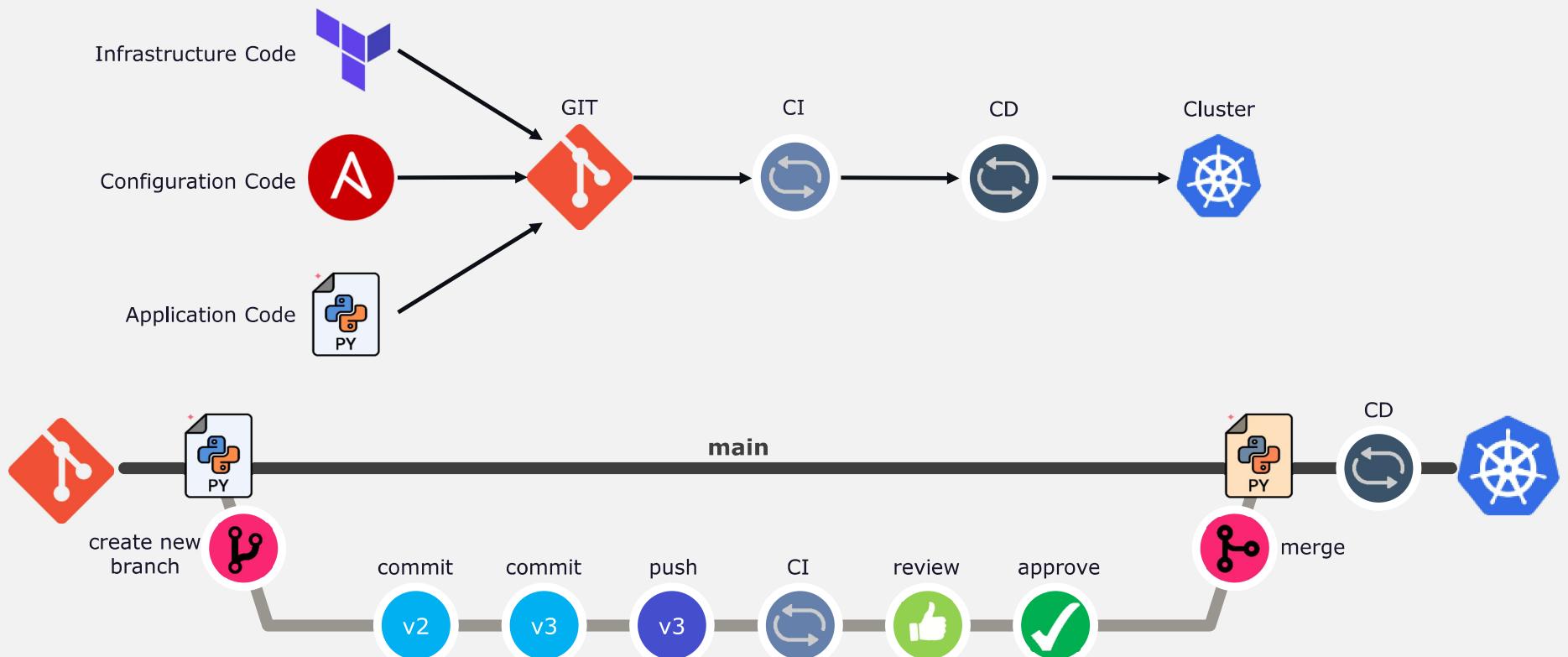
## Meeting Task Dash DevOps Team & Challenges



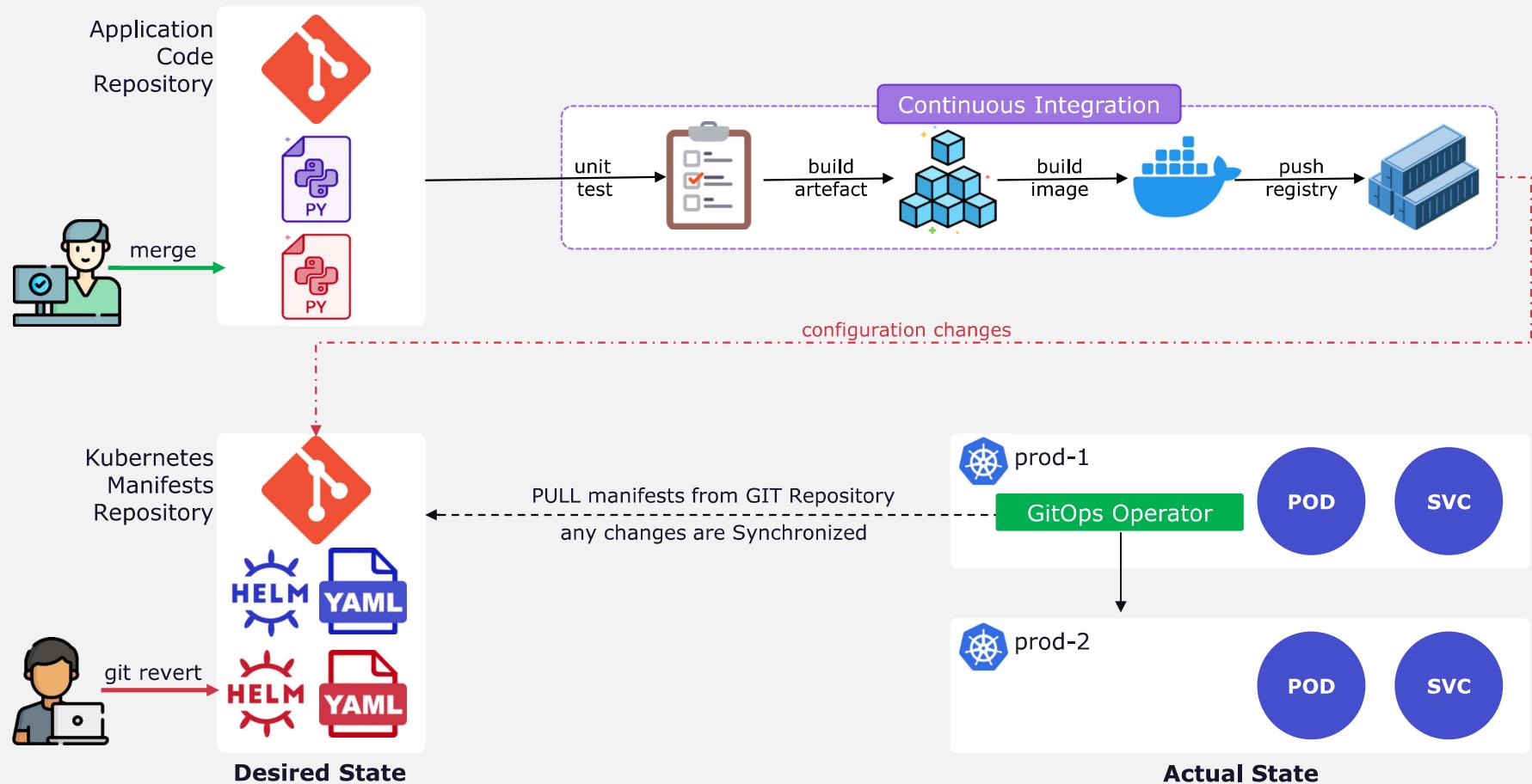
# What is GitOps?

## GitOps

GitOps can be considered an extension of Infrastructure as Code (IaC) that uses Git as the version control system

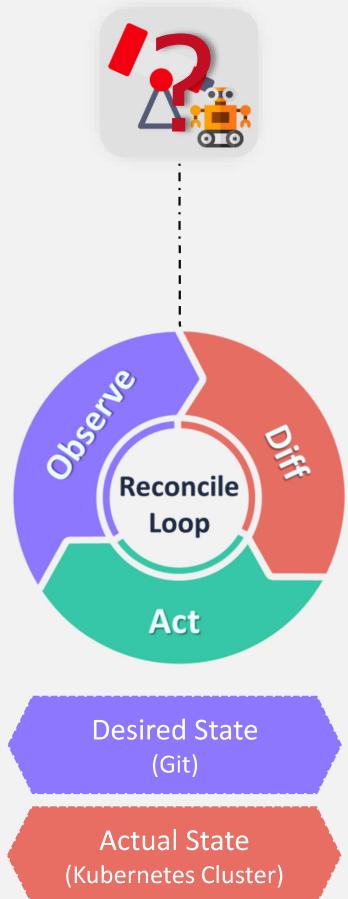
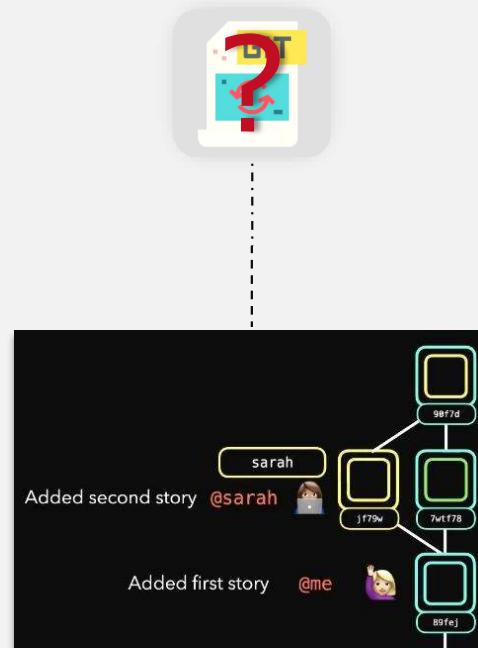


## GitOps



# GitOps Principles

## GitOps Principles

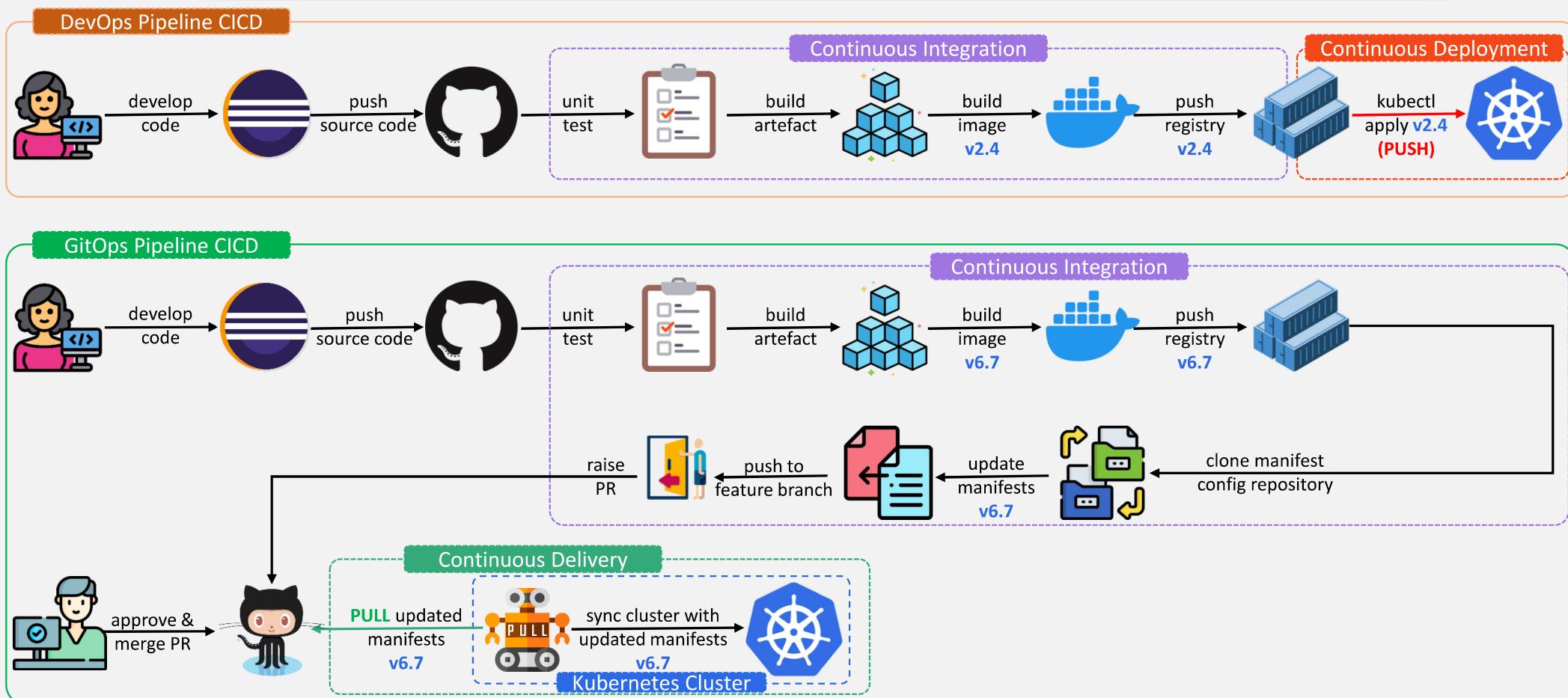




# KodeKloud

# DevOps vs GitOps

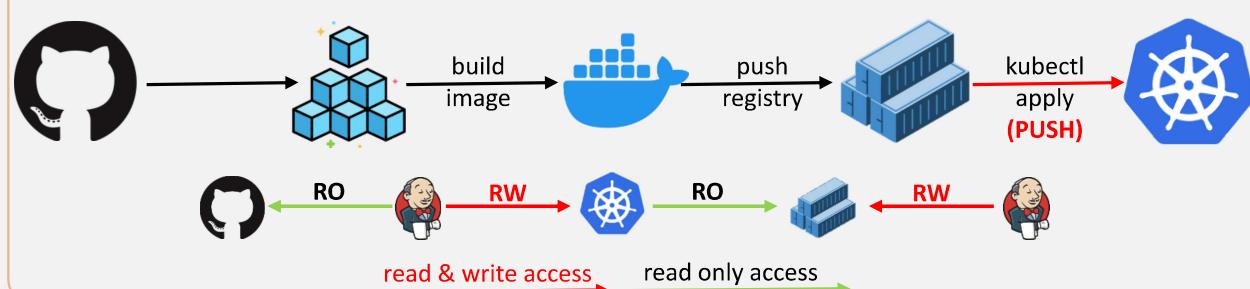
## DevOps vs GitOps



# **Push vs Pull based Deployments**

## Push vs Pull based Deployments

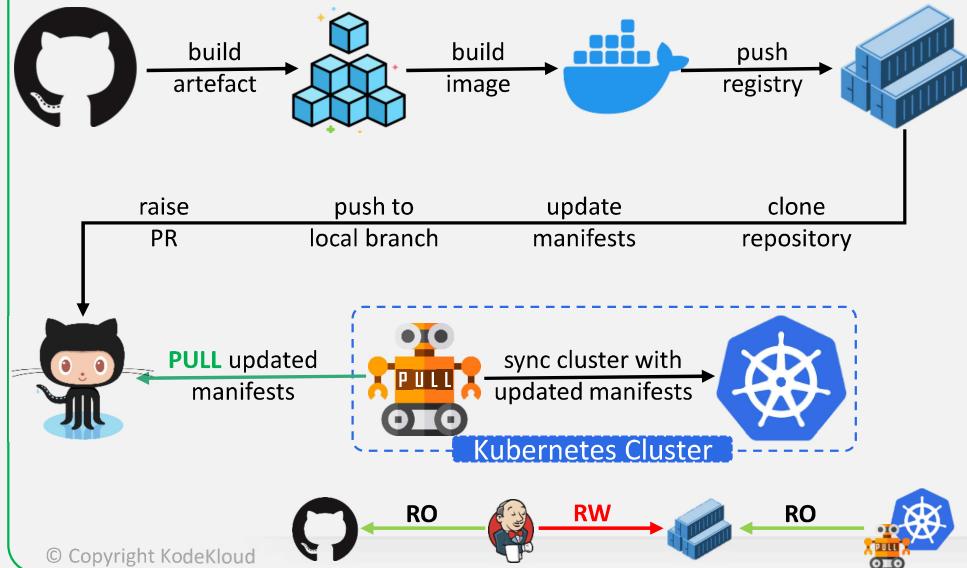
### PUSH based Deployment



- ✓ Deploying Helm Charts can be done easier
- ✓ Container version updates can be injected by build pipeline
- ✓ Secret management is easier

- ✗ Cluster config are embedded inside the CI system
- ✗ CI system has Read-Write access to the Cluster
- ✗ Deployment approach coupled to the CD system

### PULL based Deployment



- ✓ No external user/client has the right to modify the cluster
- ✓ Scan Container Register for new versions
- ✓ Secrets in Git Repository via HashiCorp Vault
- ✓ Deployment approach is not coupled to CD pipelines
- ✓ GitOps operators support multi-tenant model

- ✗ Managing secrets of Helm Chart deployments is harder
- ✗ Generic Secret management is complex

# GitOps Feature Set & Usecases

## GitOps Feature Set & UseCases

Single Source of Truth



Everything as a Code



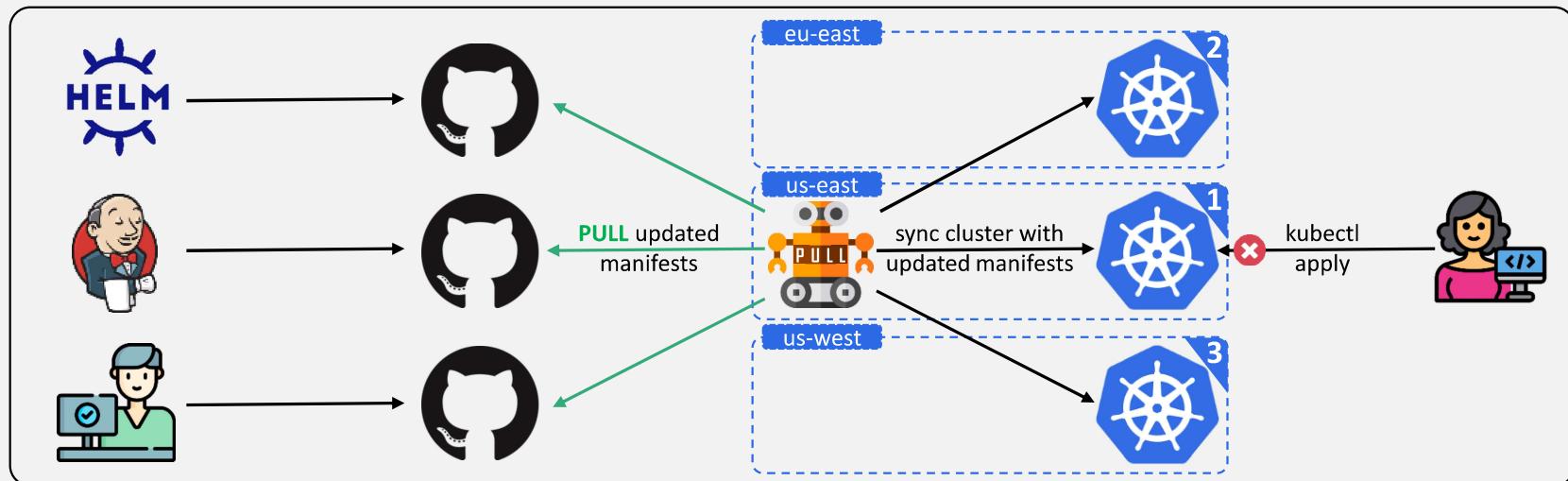
Rollback Application


  
Git Repo

Everything is Auditable



CI/CD Automation

Continuous Deployment Applications



Continuous Deployment Cluster Resources



Continuous Deployment Infrastructure



Detecting/Avoiding Configuration Drift



Multi-cluster Deployments



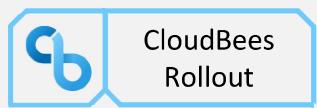
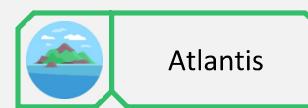
# GitOps Benefits & Challenges

## GitOps Benefits & Challenges

	Lightweight and vendor-neutral		Doesn't help with Secret Management
	Faster, Safer, Immutable, and Reproducible Deployments		Number of Git repositories
	Eliminating configuration drift		Challenges with programmatic updates
	Uses familiar tools and processes		Governance other than PR approval
	Revisions with history		Malformed YAML/Config Manifests

# GitOps Projects/Tools

## GitOps Projects/Tools





# KodeKloud

# What/Why/How ArgoCD

## What/Why/How ArgoCD

What is ArgoCD?	Argo CD is a declarative, GitOps continuous delivery tool for Kubernetes resources defined in a Git repository
	Continuously monitors running applications and comparing their live state to the desired state
	It reports the deviations and provides visualizations to help developers manually or automatically sync the live state with the desired state.
Why use ArgoCD?	 
	It extends the benefits of declarative specifications and Git-based configuration management
	It is the first step in achieving continuous operations based on monitoring, analytics, and automated remediation
How ArgoCD works?	 
	It follows the <b>GitOps</b> pattern by using Git repositories as the source of truth for the desired state of app and the target deployment envs.
	Kustomize applications      Helm charts      Ksonnet applications      Jsonnet files      YAML/JSON manifests
It automates the synchronization of the desired application state with each of the specified target environments	

# ArgoCD Concepts & Terminology

## ArgoCD Concepts & Terminology



<b>Application</b>	A group of Kubernetes resources as defined by a manifest.
<b>Application source type</b>	The tool is used to build the application. E.g. Helm, Kustomize or Ksonnet
<b>Project</b>	Provide a logical grouping of applications, which is useful when Argo CD is used by multiple teams.
<b>Target state</b>	The desired state of an application, as represented by files in a Git repository.
<b>Live state</b>	The live state of that application. What pods, configmap, secrets, etc are created/deployed in a Kubernetes cluster.
<b>Sync status</b>	Whether or not the live state matches the target state. Is the deployed application the same as Git says it should be?
<b>Sync</b>	The process of making an application move to its target state. E.g. by applying changes to a Kubernetes cluster.
<b>Sync operation status</b>	Whether or not a sync succeeded.
<b>Refresh</b>	Compare the latest code in Git with the live state. Figure out what is different.
<b>Health</b>	The health of the application, is it running correctly? Can it serve requests?

# ArgoCD Features

## ArgoCD Features

1

Automated deployment of applications to specified target environment in multiple clusters

2

Support for multiple config management/template tools (Kustomize, Helm, Ksonnet, Jsonnet, plain-YAML)

3

SSO Integration (OIDC, OAuth2, LDAP, SAML 2.0, GitHub, GitLab, Microsoft, LinkedIn)

4

Multi-tenancy and RBAC policies for authorization

5

Rollback/Roll-anywhere to any application configuration committed in Git repository

6

Health status analysis of application resources

7

Automated configuration drift detection and visualization

8

Out-of-the-box Prometheus metrics

9

Audit trails for application events and API calls

10

PreSync, Sync, PostSync hooks to support complex application rollouts (e.g. blue/green & canary upgrades)

11

Webhook integration (GitHub, BitBucket, GitLab)

12

CLI and access tokens for automation and CI integration

13

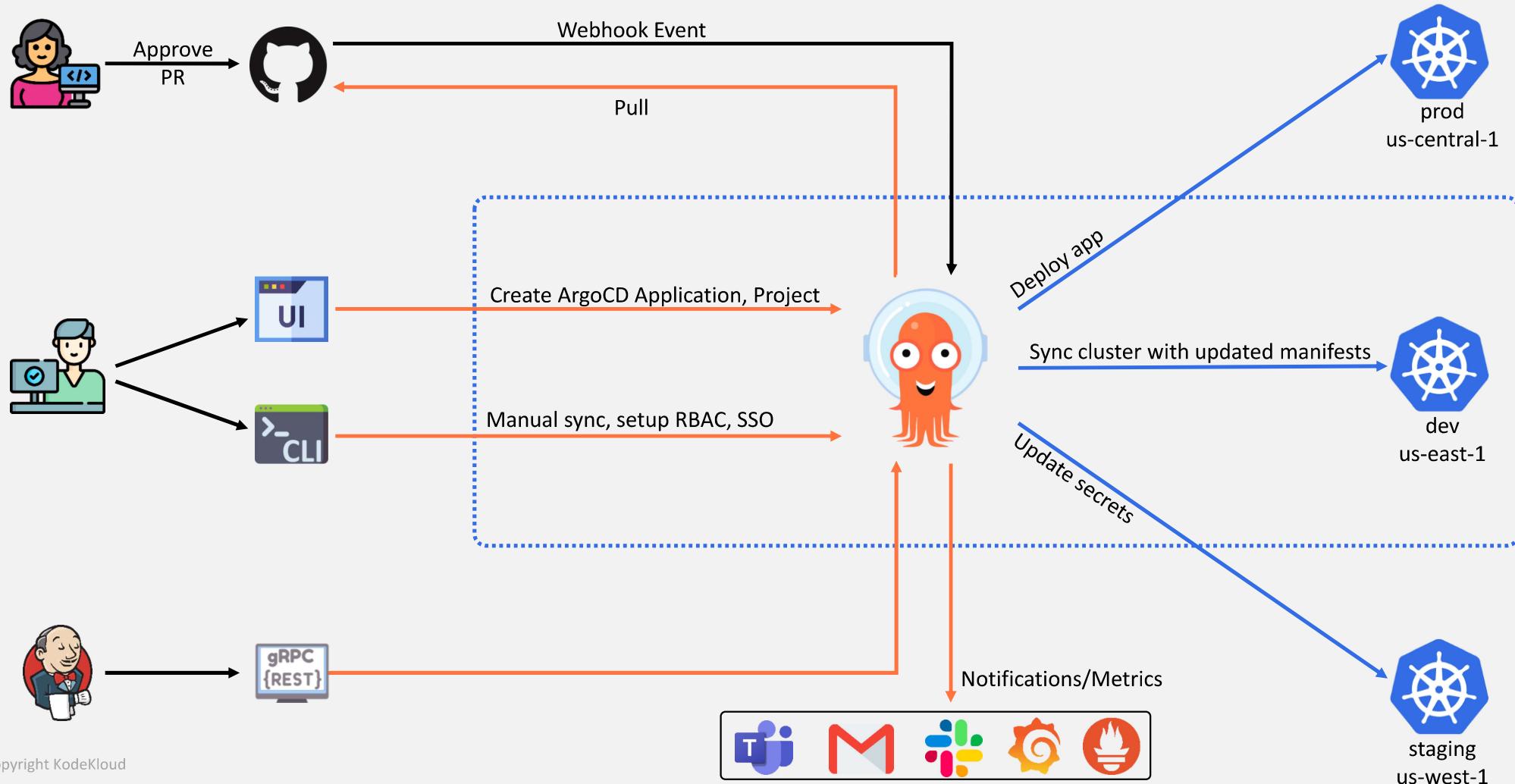
Web UI which provides real-time view of application activity

14

Automated or manual syncing of applications to its desired state

# ArgoCD Architecture

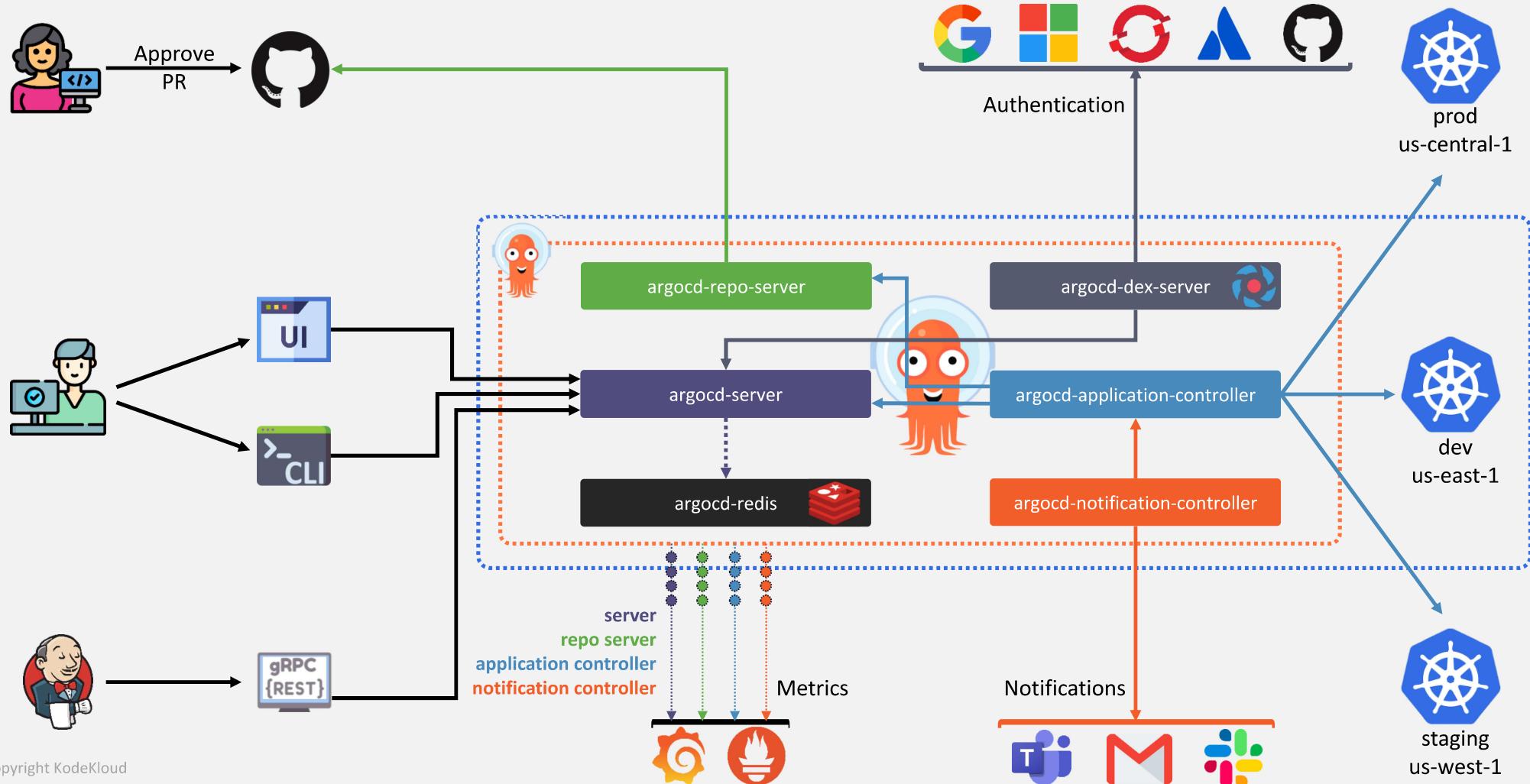
## ArgoCD Architecture



# ArgoCD Architecture

## Core Components

## ArgoCD Architecture - Core Components



# ArgoCD Installation Options

## Installation Options



>\_

```
$ kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml
```

>\_

```
$ helm repo add argo https://argoproj.github.io/argo-helm           ### Add repository
$ helm install my-argo-cd argo/argo-cd --version 4.8.0                ### Install chart
```

>\_

```
$ curl -sSL -o /usr/local/bin/argocd https://github.com/argoproj/argo-cd/releases/latest/download/argocd-linux-amd64
$ chmod +x /usr/local/bin/argocd
```

# ArgoCD Application

## ArgoCD Application

**Application**

**Source**



repoURL  
targetRevision  
path

**Source**



path  
version  
namePrefix

**Source**



version  
chartName  
releaseName  
values  
helm-parameters

**Source**



jsonnet  
extVars  
tlas  
recurse  
path

**Destination**



server  
namespace

>


```
_
```

```
$ argocd app create color-app \
--repo https://github.com/sid/app-1.git \
--path team-a/color-app \
--dest-namespace color \
--dest-server https://kubernetes.default.svc
```

*Application `color-app` created*

>


```
_
```

```
color-app.yaml
```

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: color-app
  namespace: argocd
spec:
  project: default
  source:
    repoURL: https://github.com/sid/app-1.git
    targetRevision: HEAD
    path: team-a/color
  destination:
    server: https://kubernetes.default.svc
    namespace: color
  syncPolicy:
    automated:
      selfHeal: true
    syncOptions:
      - CreateNamespace=true
```

**App**

**CREATE** **CANCEL**

**GENERAL**

Application Name: **color-app**

Project: **default**

**SYNC POLICY**

Automatic

PRUNE RESOURCES

SELF HEAL

**SYNC OPTIONS**

SKIP SCHEMA VALIDATION

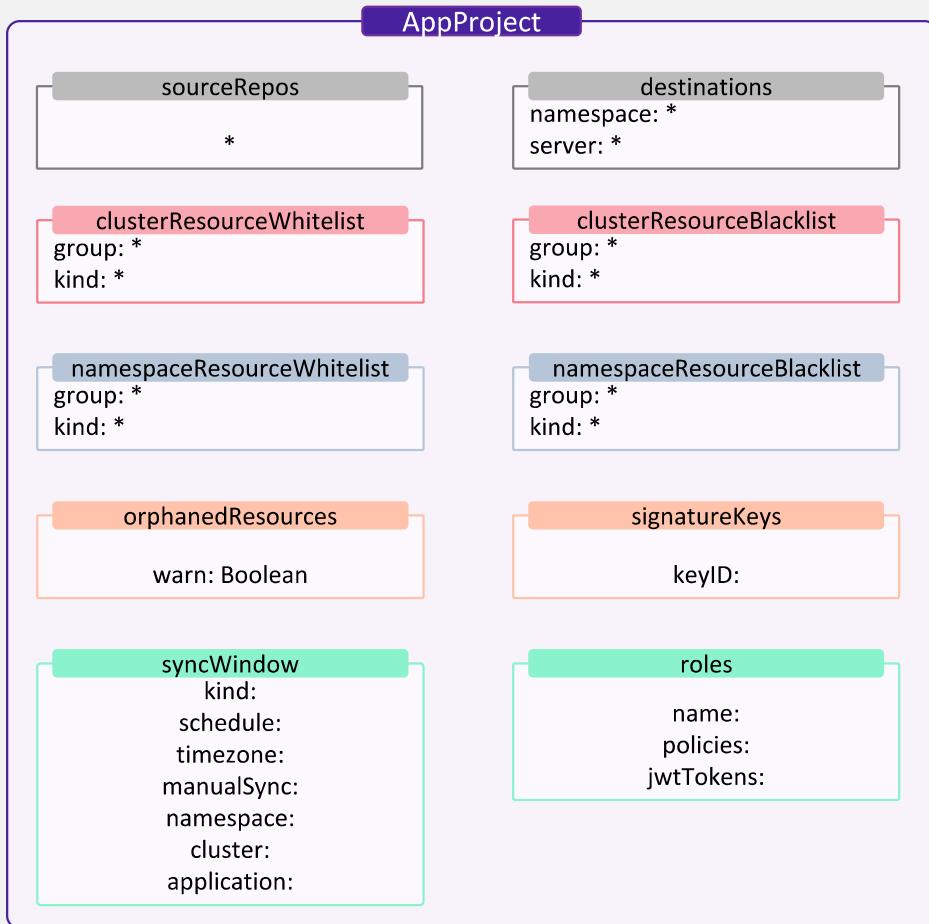
AUTO-CREATE NAMESPACE

PRUNE LAST

APPLY OUT OF SYNC ONLY

# ArgoCD AppProject

## ArgoCD AppProject



```

>_
$ kubectl get appproject -n argocd
NAME          AGE
default       10h

>_
$ kubectl get appproject default -o yaml -n argocd
apiVersion: argoproj.io/v1alpha1
kind: AppProject
metadata:
  name: default
  namespace: argocd
spec:
  clusterResourceWhitelist:
  - group: '*'
    kind: '**'
  destinations:
  - namespace: '*'
    server: '*'
  sourceRepos:
  - '*'

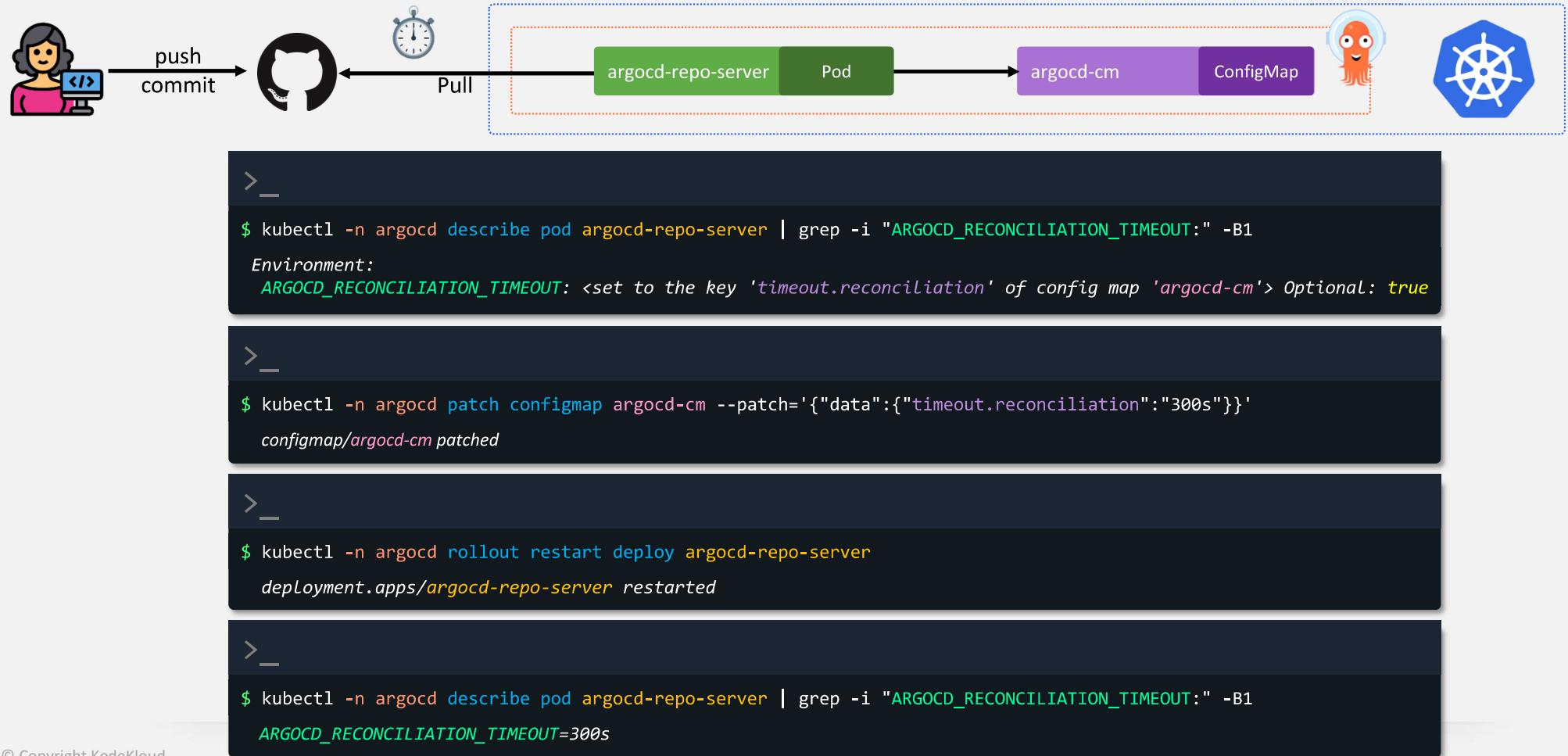
```



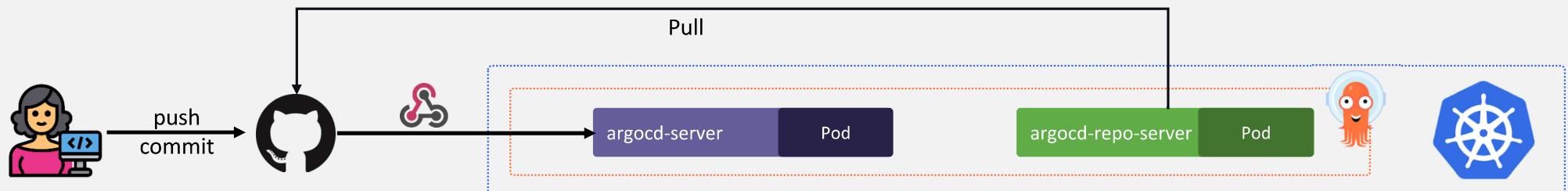
# KodeKloud

# Reconciliation Loop

## Reconciliation Loop - Using Timeout



## Reconciliation Loop - WebHook



### Webhooks / Manage webhook

[Settings](#) [Recent Deliveries](#)

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in our developer documentation.

**Payload URL \***

**Content type**

**SSL verification**  
 Enable SSL verification  Disable (not recommended)

**Which events would you like to trigger this webhook?**

- Just the push event.
- Send me everything.
- Let me select individual events.

Active  
 We will deliver event details when this hook is triggered.

[Update webhook](#) [Delete webhook](#)

[Request](#) [Response 200](#)

**Headers**

```

Request URL: https://40.78.145.130:32213/api/webhook
Request method: POST
Accept: */*
content-type: application/json
User-Agent: GitHub-Hookshot/b257cdf
X-GitHub-Delivery: d1d5a79e-e49e-11ec-8ec1-39c21920f0c0
X-GitHub-Event: push
X-GitHub-Hook-ID: 361799416
X-GitHub-Hook-Installation-Target-ID: 499894000
X-GitHub-Hook-Installation-Target-Type: repository
  
```

# Application Health Checks

## Application Health Checks

	Healthy	All resources are 100% healthy	Service	Deployment, ReplicaSet, StatefulSet, DaemonSet
	Progressing	Resource is unhealthy, but could still be healthy given time	Load Balancer status.loadBalancer.ingress	Observed Gen = Deserved Gen
	Degraded	Resource status indicates a failure or an inability to reach a healthy state	hostname or IP	updated replicas = desired replicas
	Missing	Resource is not present in the cluster	Ingress	PersistentVolumeClaim
	Suspended	Resource is suspended or paused. Typical example is a paused Deployment	status.loadBalancer.ingress hostname or IP	status.phase = Bound
	Unknown	Health assessment failed and actual health status is unknown		

Argo CD supports custom health checks written in Lua



1. Define a Custom Health Check in argocd-cm ConfigMap

2. Contribute a Custom Health Check

```
>_ color-cm.yaml
```

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: php-color-cm
data:
  TRIANGLE_COLOR: white
```

Format - GROUP_RESOURCE	
apps.Deployment	Secret
batch.CronJob	Namespace
extensions.Ingress	Pod
networking.k8s.io.RuntimeClass	LimitRange
rbac.authorization.k8s.io.Role	PersistentVolumeClaim

© Copyright KodeKloud

```
>_ argocd-cm.yaml
```

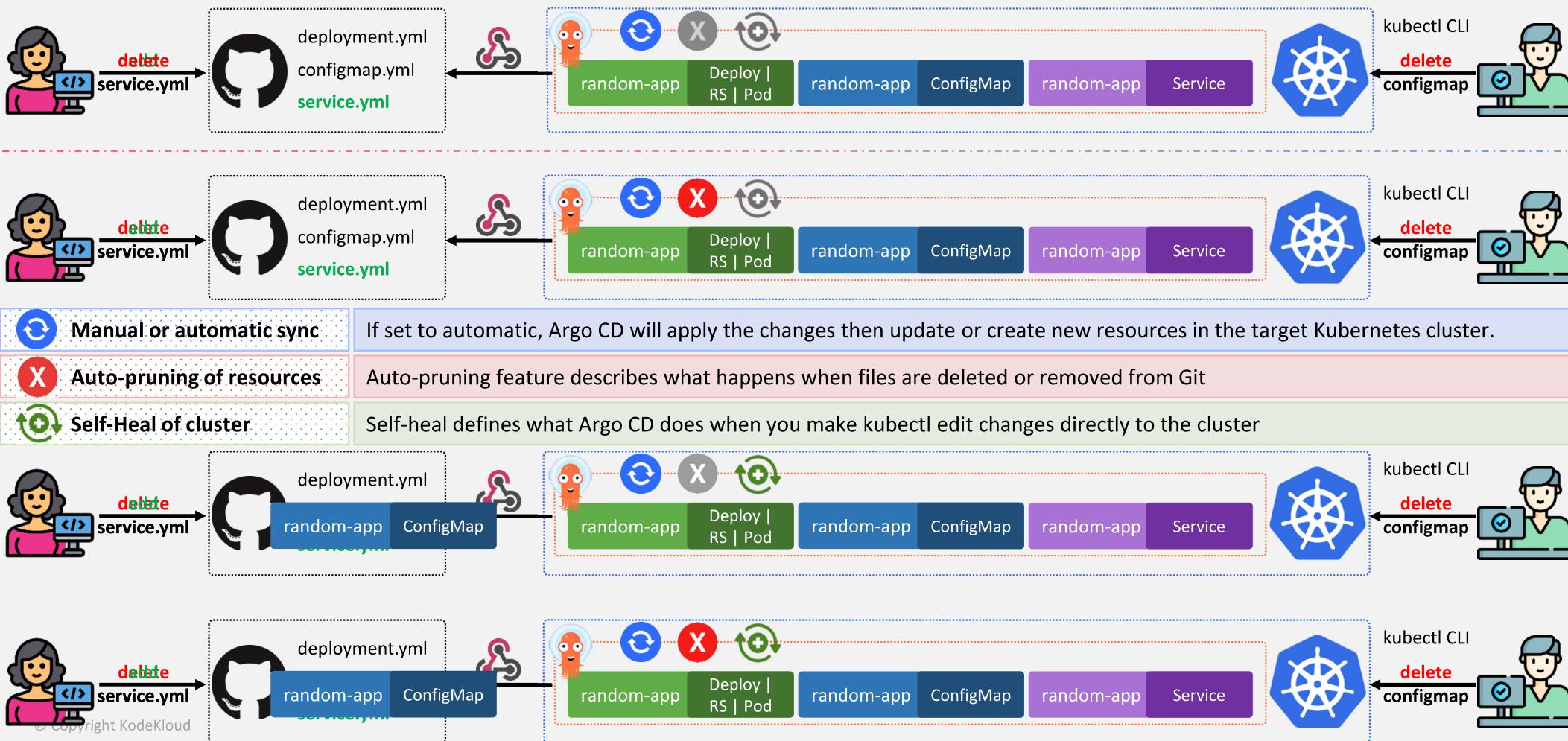
```
apiVersion: v1
kind: ConfigMap
metadata:
  name: argocd-cm
data:
  resource.customizations.health.ConfigMap: |
    hs = {}
    hs.status = "Healthy"
    if obj.data.TRIANGLE_COLOR == "white" then
      hs.status = "Degraded"
      hs.message = "Use a different COLOR"
    end
    return hs
  timeout.reconciliation: 300s
```

ArgoCD UI

APP HEALTH		Degraded
php-color-cm		cm
an hour		
HEALTH		Degraded
HEALTH	Use a different COLOR	

# ArgoCD Sync Strategies

## Sync Strategies



# Declarative Setup Mono Application

## Declarative Setup - Mono-Application

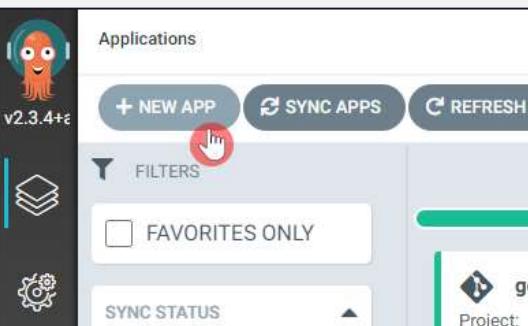
```
>_ tree structure of Git Repository
```

```
.
  └── declarative
      └── manifests
          └── geocentric-model
              ├── deployment.yml
              └── service.yml
      └── mono-app
          └── geocentric-app.yml
```

```
>_ mono-app/geocentric-app.yml
```

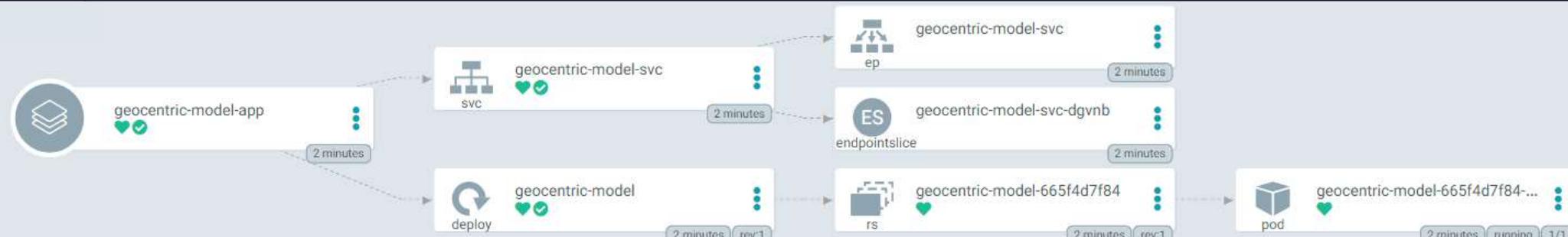
```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: geocentric-model-app
  namespace: argocd
spec:
  project: default
  source:
    repoURL: https://github.com/sidd-harth/test-cd.git
    targetRevision: HEAD
    path: ./declarative/manifests/geocentric-model
  destination:
    server: https://kubernetes.default.svc
    namespace: geocentric-model
  syncPolicy:
    syncOptions:
      - CreateNamespace=true
  automated:
    selfHeal: true
```

```
>_
$ kubectl apply -f mono-app/geocentric-app.yml
application.argoproj.io/geocentric-model-app created
```



**geocentric-model-app**

Project:	default
Labels:	
Status:	Healthy Synced
Repository:	<a href="https://github.com/sidd-harth/test-cd.git">https://github.com/sidd-harth/test-cd.git</a>
Target R...	HEAD
Path:	./declarative/manifests/geocentric-model
Destinati...	in-cluster
Namesp...	geocentric-model



# Declarative Setup

## App-of-Apps

## Declarative Setup - App-of-Apps

**Git Repository**

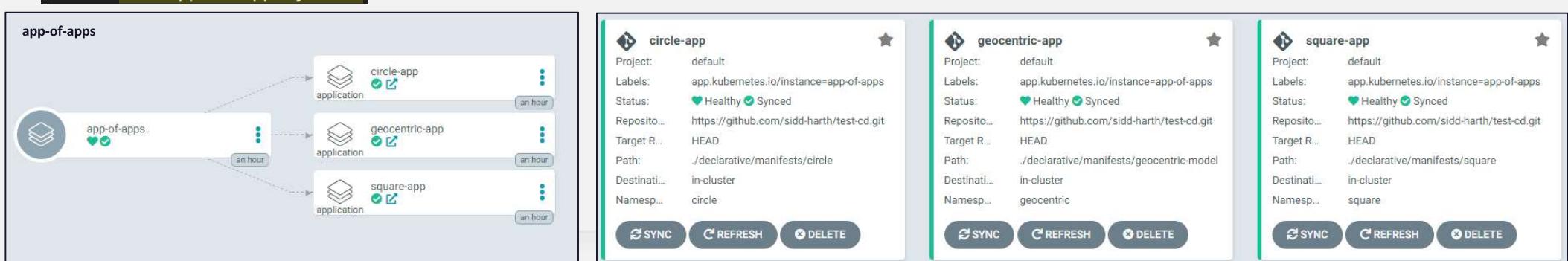
```
>_ declarative
  └── app-of-apps
      ├── circle-app.yml
      ├── geocentric-app.yml
      └── square-app.yml
  └── manifests
      └── circle
          ├── deployment.yml
          ├── service.yml
          └── geocentric-model
              ├── deployment.yml
              ├── service.yml
              └── square
                  ├── deployment.yml
                  └── service.yml
  └── mono-app
      └── geocentric-app.yml
  └── multi-app
      └── app-of-apps.yml
```

**declarative/multi-app/app-of-apps.yml**

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: app-of-apps
spec:
  project: default
  source:
    repoURL: https://github.com/sidd-harth/test-cd.git
    targetRevision: HEAD
    path: ./declarative/app-of-apps
  destination:
    server: https://kubernetes.default.svc
    namespace: argocd
  syncPolicy:
    automated:
      prune: true
      selfHeal: true
```

**declarative/app-of-apps/circle-app.yml**

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: circle-app
  namespace: argocd
spec:
  project: default
  source:
    repoURL: https://github.com/sidd-harth/test-cd.git
    targetRevision: HEAD
    path: ./declarative/manifests/circle
  destination:
    server: https://kubernetes.default.svc
    namespace: circle
  syncPolicy:
    syncOptions:
      - CreateNamespace=true
```



# ArgoCD - Deploy HELM Chart

## ArgoCD - Deploy HELM Chart

```
>_ Git Repository
>_
helm-chart
  --> Chart.yaml
  --> templates
    --> NOTES.txt
    --> _helpers.tpl
    --> configmap.yaml
    --> deployment.yaml
    --> service.yaml
  --> values.yaml
```

```
$ argocd app create random-shapes \
--repo https://github.com/sidd-harth/test-cd.git \
--path helm-chart \
--helm-set replicaCount=2 \
--helm-set color.circle=pink \
--helm-set color.square=violet \
--helm-set service.type=NodePort \
--dest-namespace default \
--dest-server https://kubernetes.default.svc

application 'random-shapes' created
```

```
$ argocd app create nginx \
--repo https://charts.bitnami.com/bitnami \
--helm-chart nginx \
--revision 12.0.3 \
--values-literal-file values.yaml \
--dest-namespace default \
--dest-server https://kubernetes.default.svc

application 'nginx' created
```

**Connect repo using HTTPS**

Type	helm
Name	bitnami-charts
Project	default
Repository URL	https://charts.bitnami.com/bitnami

```
$ helm ls
NAME NAMESPACE REVISION UPDATED STATUS CHART APP VERSION

$ argocd app get nginx
Name: nginx
Project: default
Server: https://kubernetes.default.svc
Namespace: default
URL: https://10.99.148.201/applications/nginx
Repo: https://charts.bitnami.com/bitnami
Target: 12.0.3
SyncWindow: Sync Allowed
Sync Status: Synced to 12.0.3
Health Status: Healthy

GROUP KIND NAMESPACE NAME STATUS HEALTH MESSAGE
          Service default nginx Synced Healthy service/nginx created
apps   Deployment default nginx Synced Healthy deployment.apps/nginx created
```

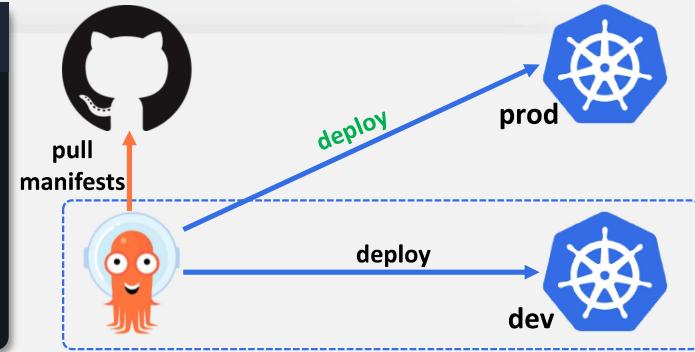
# ArgoCD Multi-Cluster Deployment

## Multi-Cluster Deployment

```
>_           adding Kubernetes Cluster to kubeconfig
$ kubectl config set-cluster prod --server=https://1.2.3.4 --certificate-authority=prod.crt
Cluster "prod" set.

$ kubectl config set-credentials admin --client-certificate=admin.crt --client-key=admin.key
User "admin" set.

$ kubectl config set-context admin-prod --cluster=prod --user=admin --namespace=prod-app
Context "admin-prod" set.
```



```
>
$ argocd cluster add admin-prod
WARNING: This will create a service account `argocd-manager` on the cluster referenced by context `admin-prod` with full cluster level admin
privileges. Do you want to continue [y/N]? y
INFO[0011] ServiceAccount "argocd-manager" created in namespace "kube-system"
INFO[0011] ClusterRole "argocd-manager-role" created
INFO[0011] ClusterRoleBinding "argocd-manager-role-binding" created
Cluster 'https://1.2.3.4' added
```

```
>
$ argocd cluster list


| SERVER                         | NAME       | VERSION | STATUS     | MESSAGE | PROJECT |
|--------------------------------|------------|---------|------------|---------|---------|
| https://1.2.3.4                | admin-prod | 1.21    | Successful |         |         |
| https://kubernetes.default.svc | in-cluster | 1.20    | Successful |         |         |


$ kubectl describe secret cluster-1.2.3.4-1827028835 -n argocd
....Data
config: 3017 bytes      #user token/cert
name:          54 bytes  #context name
server:        21 bytes  #server url
```



# KodeKloud

# User Management

## User Management

### New Users

New users can be defined in argocd-cm ConfigMap

accounts.USERNAME: apikey, login

accounts.SID.enabled: "false"

```
>_
$ argocd account list
NAME      ENABLED   CAPABILITIES
admin     true      Login
jai       true      apiKey, Login
ali       true      apiKey, Login
```

```
>_
$ kubectl -n argocd patch configmap argocd-cm --patch='{"data": {"accounts.jai": "apiKey,login"}}'
configmap/argocd-cm patched
$ kubectl -n argocd patch configmap argocd-cm --patch='{"data": {"accounts.ali": "apiKey,login"}}'
configmap/argocd-cm patched
```

```
>_
$ argocd account update-password --account jai
*** Enter password of currently logged in user (admin):
*** Enter new password for user jai:
*** Confirm new password for user jai:
Password updated
```

```
>_
$ argocd account update-password \
--account jai \
--new-password j@i_p@ssw0rd \
--current-password @dmin_p@$$w0rd
Password updated
```

### Default Roles

By default, all new users have no access

role:readonly and role:admin

policy.default from the argocd-rbac-cm ConfigMap



Alice  
(admin)



Jai  
(Add Clusters)



Ali  
(Project Kia Admin)



Kia Team

```
>_
$ kubectl -n argocd patch configmap argocd-rbac-cm --patch='{"data": {"policy.default": "role:readonly"}}'
configmap/argocd-rbac-cm patched
```

# **RBAC - Role Based Access Control**

## RBAC - Role Based Access Control

RBAC Policies	Applications, logs, and exec (which belong to a project)	p, <role/user/group>, <resource>, <action>, <project>/<object>
	All other resources:	p, <role/user/group>, <resource>, <action>, <object>

```
>_
$ kubectl -n argocd patch configmap argocd-rbac-cm \
--patch='{"data":{"policy.csv":"p, role:create-cluster, clusters, create, *, allow\ng, jai, role:create-cluster"}}'
configmap/argocd-rbac-cm patched
```

```
>_
$ argocd account can-i create clusters '*'
yes          #Logged in as - jai
```

```
>_
$ argocd account can-i delete clusters '*'
no           #Logged in as - jai
```

```
>_
$ kubectl -n argocd patch configmap argocd-rbac-cm \
--patch='{"data":{"policy.csv": "p, role:kia-admins, applications, *, kia-project/*, allow\ng, ali, role:kia-admins"}}'
configmap/argocd-rbac-cm patched
```

```
>_
$ argocd account can-i sync applications kia-project/*
yes          #Logged in as - ali
```



Alice  
(admin)



Jai  
(Add Clusters)



Ali  
(Project Kia Admin)



Kia Team

# Dex GitHub Connector



## Dex GitHub Connector



Client App  
ArgoCD



IdP  
Okta

**Okta - Create a new app integration**

**General Settings**

- App name:** ArgoCD
- App logo (optional):** A small orange octopus icon.
- App visibility:**
  - Do not display application icon to users
  - Do not display application icon in the Okta Mobile app

**SAML Settings**

- Single sign-on URL:** https://40.78.145.130:32213/api/dex/callback
- Audience URI (SP Entity ID):** https://40.78.145.130:32213/api/dex/callback
- Default RelayState:** Unspecified
- Name ID format:** Unspecified
- Application username:** Okta username
- Update application username on:** Create and update

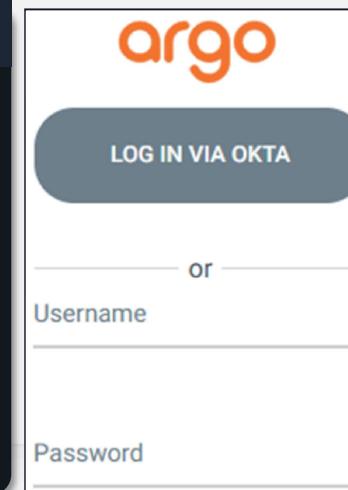
**Assignment**

Priority	Assignment
1	kia-team No description

1 Identity Provider Single Sign-On URL:  
https://dev-9725117.okta.com/app/dev-972511

3 X.509 Certificate:

```
-----BEGIN CERTIFICATE-----
MIIDpJCCAoBgAwIBAgIAYmy8voMA0GCSqGSIb3DQE
A1UECAkQZG9pZ2cmIgYtIEWNBQQA1UEBwNNU2FvIEZ
MBIGA1UECwwlU1PUHJvdmlkZXIKFDASgNVBAhM
FgipbmZvG9ydgEuY2otb84XOTIyMDYxNTA5NTYwFA
BAYTA1TMRMWEYDQVQ1QApDIWxZm9ybm1hMRYwFA1
VQKQDAPa3RMRBQwEgIDQQLDAtTU09cm2aNRlcjE
BgkqhkiG9w0BCQEwdWlu2m9Ab2tOIS5jb20wgEiMA0
AQQUKHKVQDCT1QVOSFTVkg+o6TGN1ejjo1nPKg2
RN0d9qg0KLbuLnkh1QfP2AzrgRtgJQ4BhVn2CVj
r
...
dex.config: |
  connectors:
    - type: saml
      id: okta
      name: Okta
      config:
        ssoURL: <okta-idp-sso-url>
        caData: |
          <base64encoded X.509 Certificate>
        usernameAttr: name
        emailAttr: email
        groupsAttr: groups
-----END CERTIFICATE-----
```



\$ kubectl -n argocd edit configmap argocd-rbac-cm

...

**data:**

```
policy.csv: |-
  p, role:crudApps, applications, *
    kia-project/*, allow
  g, kia-team, role:crudApps
  ...
  configmap/argocd-rbac-cm edited
```

# Bitnami Sealed Secret with ArgoCD

## Bitnami Sealed Secret with ArgoCD

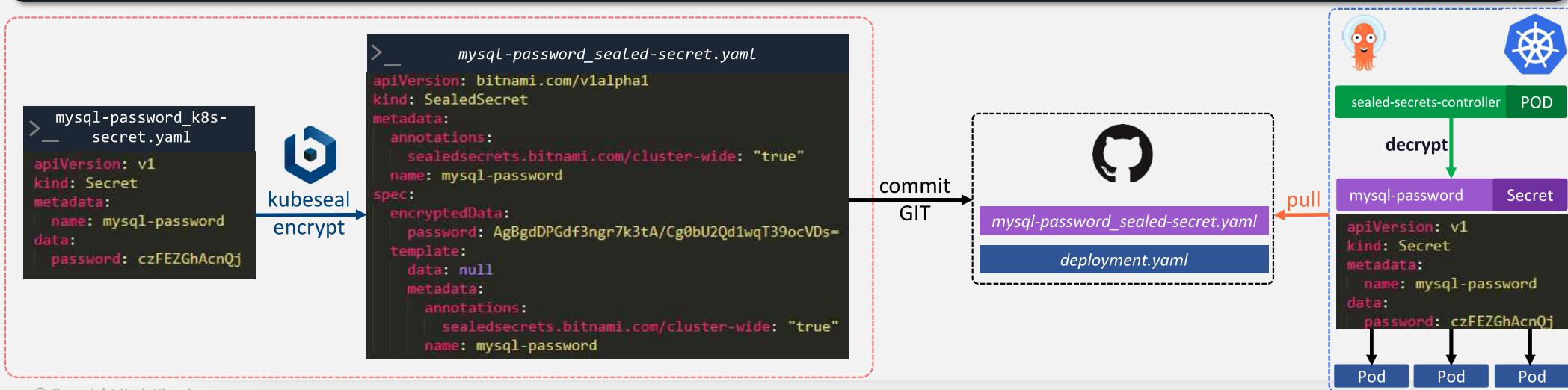
>\_

```
$ kubectl create secret generic mysql-password --from-literal=password=s1Ddh@rt# --dry-run=client -o yaml > mysql-password_k8s-secret.yaml

$ argocd app create sealed-secrets --repo https://bitnami-labs.github.io/sealed-secrets --helm-chart sealed-secrets --revision 2.2.0 --dest-namespace kube-system --dest-server https://1.2.3.4
application 'sealed-secrets' created

$ wget https://github.com/bitnami-labs/sealed-secrets/releases/download/v0.18.0/kubeseal-0.18.0-linux-amd64.tar.gz -O kubeseal && sudo install -m 755 kubeseal /usr/local/bin/kubeseal
2022-09-13 18:50:05 (111 MB/s) - 'kubeseal' saved [18116498/18116498]

$ kubeseal -o yaml --scope cluster-wide --cert sealedSecret.crt < mysql-password_k8s-secret.yaml > mysql-password_sealed-secret.yaml
```



# HashiCorp Vault with ArgoCD Vault Plugin

## HashiCorp Vault with ArgoCD Vault Plugin - Part 1

```
>_
$ vault secrets enable -path=crds kv-v2
Success! Enabled the kv secrets engine at: crds/

$ vault kv put crds/mysql MYSQL-PASSWORD=1234567
Key          Value
---          -----
created_time 2022-08-31T11:17:38.755927206Z
deletion_time n/a
destroyed     false
version       1
```

```
>_
$ cat mysql-secret.yaml
apiVersion: v1
kind: Secret
metadata:
  name: mysql-secret
  annotations:
    .argv.kubernetes.io/path: "crds/data/mysql"
type: Opaque
stringData:
  password: <MYSQL-PASSWORD>
```

```
>_
$ curl -Lo argocd-vault-plugin https://github.com/argoproj-labs/argocd-vault-plugin/releases/download/v1.10.0/argocd-vault-plugin_v1.10.0_linux_amd64
$ chmod +x argocd-vault-plugin && mv argocd-vault-plugin /usr/local/bin
```

```
>_
$ cat vault.env
VAULT_ADDR=http://vault:8200
VAULT_TOKEN=s.aokHnABJZD3HnABJZ73nIozm9wosK02wQq
AVP_TYPE=vault
AVP_AUTH_TYPE=token
```

```
>_
$ argocd-vault-plugin generate -c vault.env - < mysql-secret.yaml
apiVersion: v1
kind: Secret
metadata:
  name: mysql-secret
  annotations:
    .argv.kubernetes.io/path: "crds/data/mysql"
type: Opaque
stringData:
  password: 1234567
```

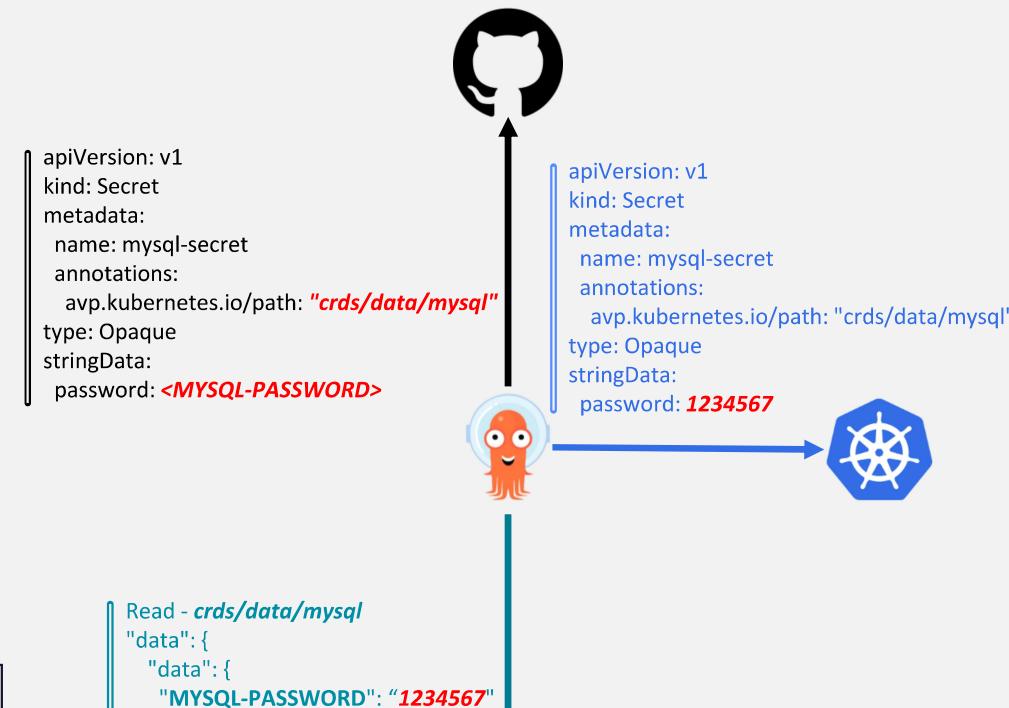
# ArgoCD with ArgoCD Vault Plugin

## HashiCorp Vault with ArgoCD Vault Plugin - Part 2

```
>_
...
containers:
  - name: argocd-repo-server
    volumeMounts:
      - name: custom-tools
        mountPath: /usr/local/bin/argocd-vault-plugin
volumes:
  - name: custom-tools
    emptyDir: {}
initContainers:
  - name: download-tools
    image: 'alpine:3.8'
    command: [- sh, -c ]
    args:
      - wget -O argocd-vault-plugin
        https://github.com/argocd/argocd-vault-plugin/v1.10.1 && chmod +x
        argocd-vault-plugin && mv argocd-vault-plugin /custom-tools/
volumeMounts:
  - mountPath: /custom-tools
    name: custom-tools
```

```
>_
...
data:
  configManagementPlugins: |-
    - name: argocd-vault-plugin
      generate:
        command: ["argocd-vault-plugin"]
        args: ["generate", "./"]
```

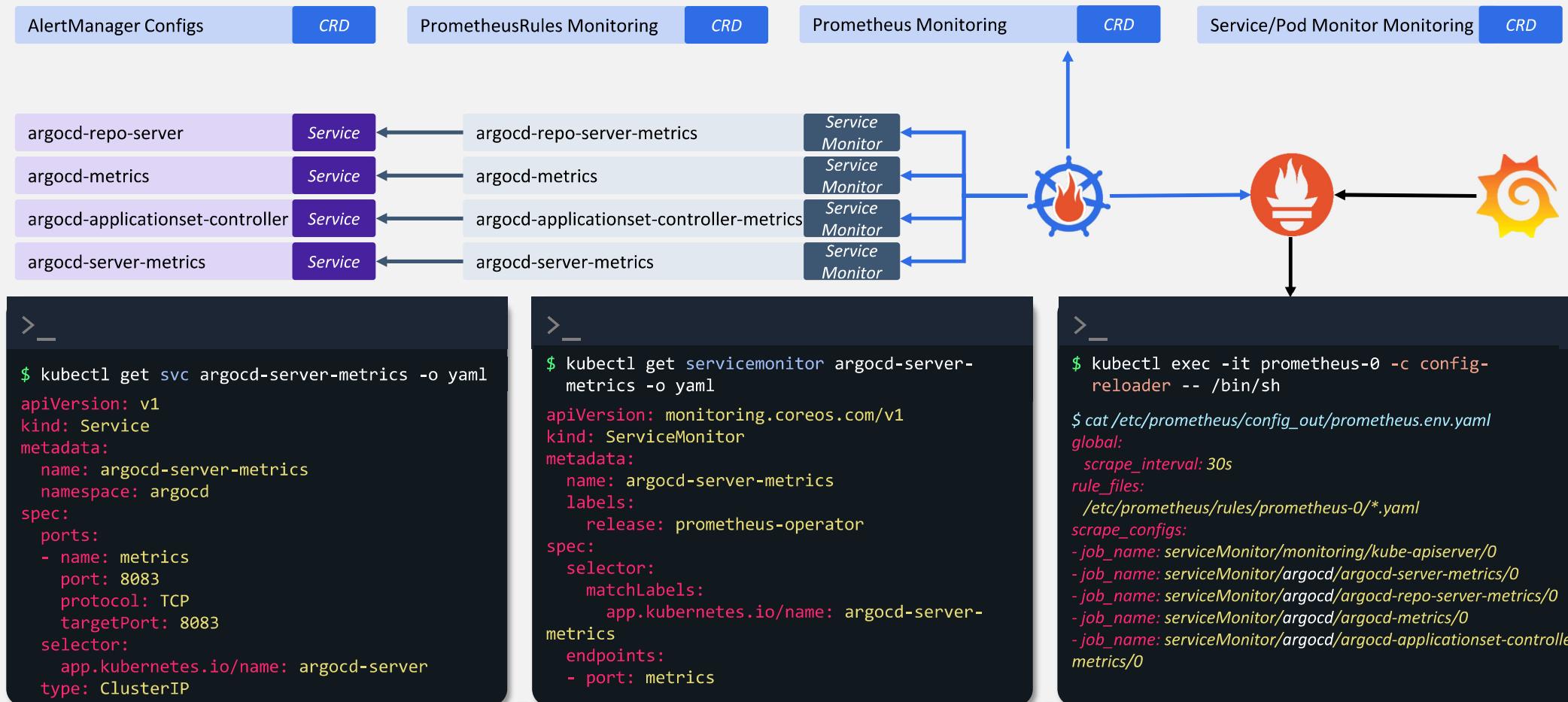
argocd-vault-plugin	
VAULT_ADDR	= http://vault:8200
VAULT_TOKEN	= s.aokHnABJZD3
AVP_TYPE	= vault
AVP_AUTH_TYPE	= token



# ArgoCD Metrics

# Prometheus + Grafana

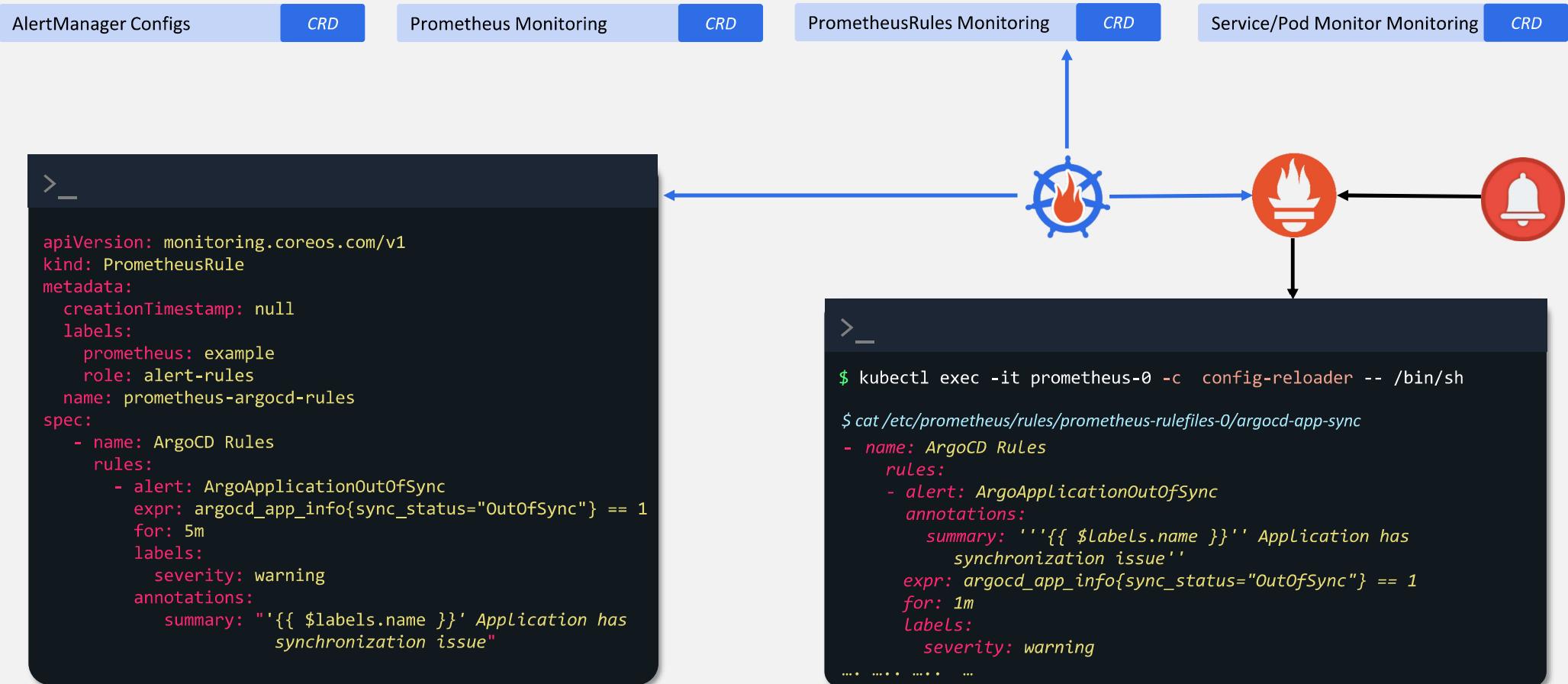
## Prometheus + Grafana – ArgoCD



# **ArgoCD Metrics**

# **Prometheus + AlertManager**

## Prometheus + Alertmanager - ArgoCD



# Notifications

## Notifications - ArgoCD

**Name app & choose workspace**

App Name: argocd-notification-app  
Don't worry - you'll be able to change this later.

Pick a workspace to develop your app in:  
mcd-level2

[Cancel](#) [Create App](#)

**Scopes**

**Bot Token Scopes**

Scopes that govern what your app can access.

OAuth Scope	Description
chat:write	Send messages as @slackargov2
chat:write.customize	Send messages as @slackargov2 with a

**Slash Commands**

**Workflow Steps**

**OAuth & Permissions**

Event Subscriptions

User ID Translation

App Manifest [NEW](#)

Beta Features

**OAuth Tokens for Your Workspace**

These tokens were automatically generated when you  
You can use these to authenticate your app. [Learn more](#)

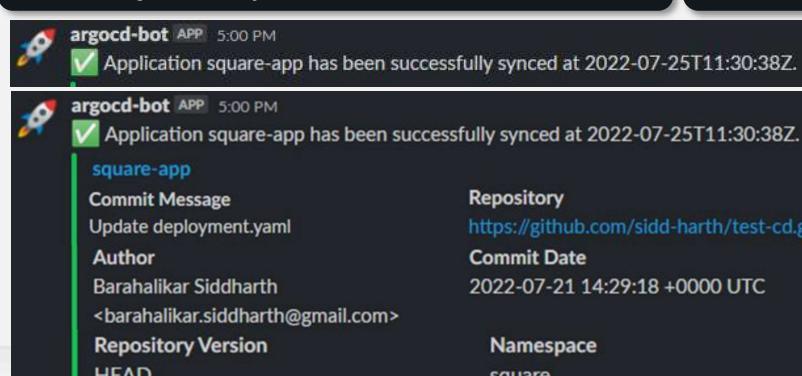
**Bot User OAuth Token**

xoxb-3323832857318-3839459687381-qtenS6po

```
$ kubectl edit configmap argocd-notifications-cm
apiVersion: v1
kind: ConfigMap
metadata:
  name: argocd-notifications-cm
data:
  service.slack: |
    token: $slack-token
    username: argocd-bot
    icon: ":rocket:"
  trigger.on-sync-succeeded: |
    - when: app.status.sync.status == 'Synced'
      send: [app-sync-succeeded-slack]
  template.app-sync-succeeded-slack: |
    message: |
      Application {{.app.metadata.name}} sync is
      {{.app.status.sync.status}}.
  slack:
    attachments:
      [
        "title": "{{ .app.metadata.name}}",
...
```

```
$ kubectl edit secret argocd-notifications-secret
apiVersion: v1
kind: Secret
metadata:
  name: argocd-notifications-secret
stringData:
  slack-token: xoxb-3323832857318-383945967381...
secret/argocd-notifications-secret edited
```

```
$ kubectl edit app argocd-application
apiVersion: argoproj.io/v1alpha1
kind: Application
name: square-app
namespace: argocd
metadata:
  annotations:
    notifications.argoproj.io/subscribe.
    on-sync-succeeded.slack: <slack-channel>
```



**Notification Catalog**

- app-created
- app-deleted
- app-health-degraded
- app-sync-status-unknown
- app-sync-running
- app-deployed
- app-sync-failed
- app-sync-succeeded

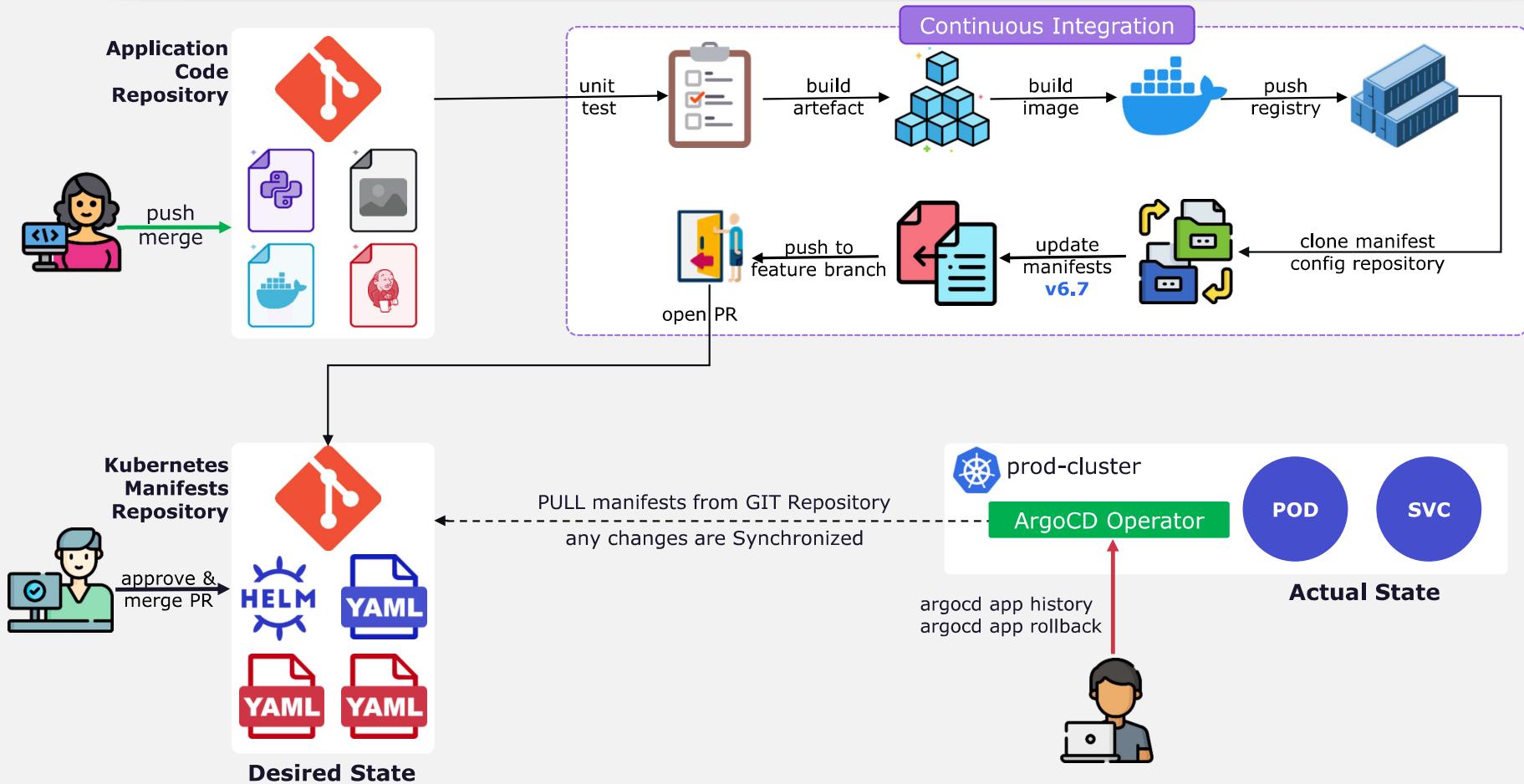




# KodeKloud

# CI/CD with GitOps

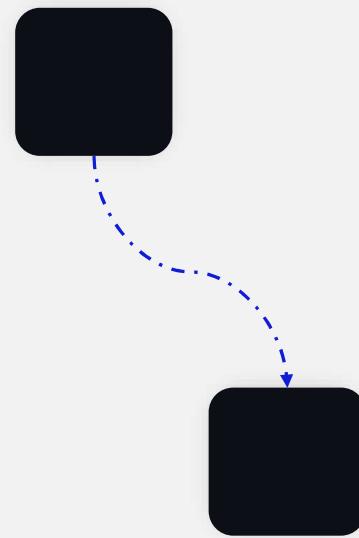
## CI/CD with GitOps





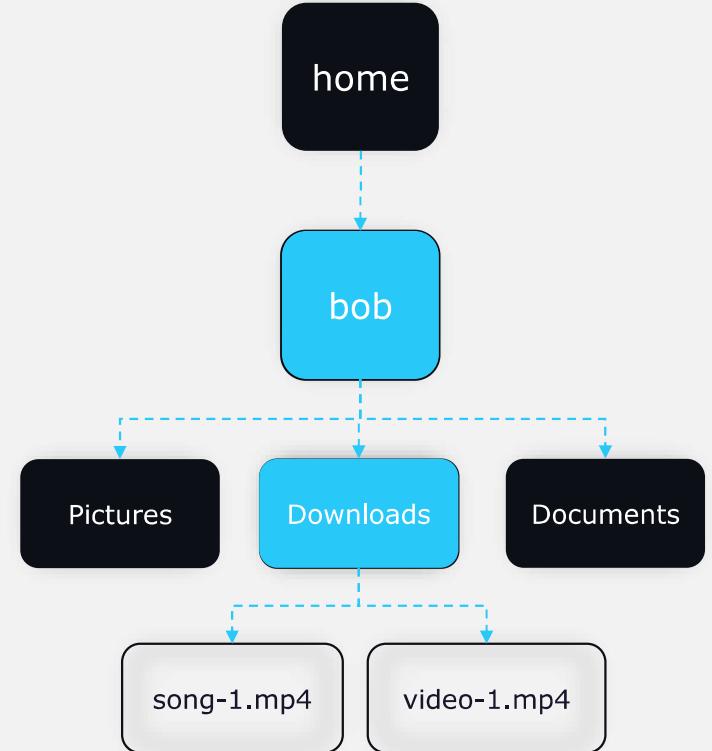
# KodeKloud

## Navigating the File System

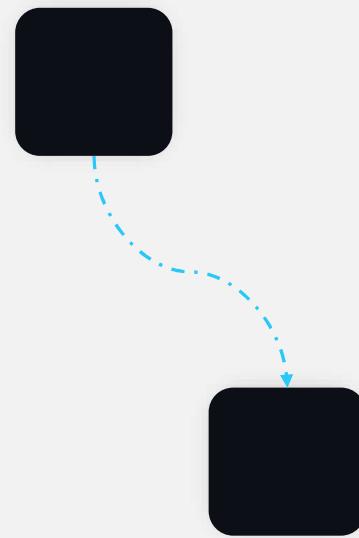


## Navigating the Filesystem

```
>_  
  
$ pwd  
/home/bob  
  
$ ls  
Pictures Downloads Documents  
  
$ cd Downloads  
  
$ ls  
song-1.mp4 video-1.mp4  
  
$ cd ..
```



Creating/Deleting Files



## Command & Illustration

>\_

```
$ ls  
File1 file2 file3
```

dir1

file1

file2

file3





# KodeKloud