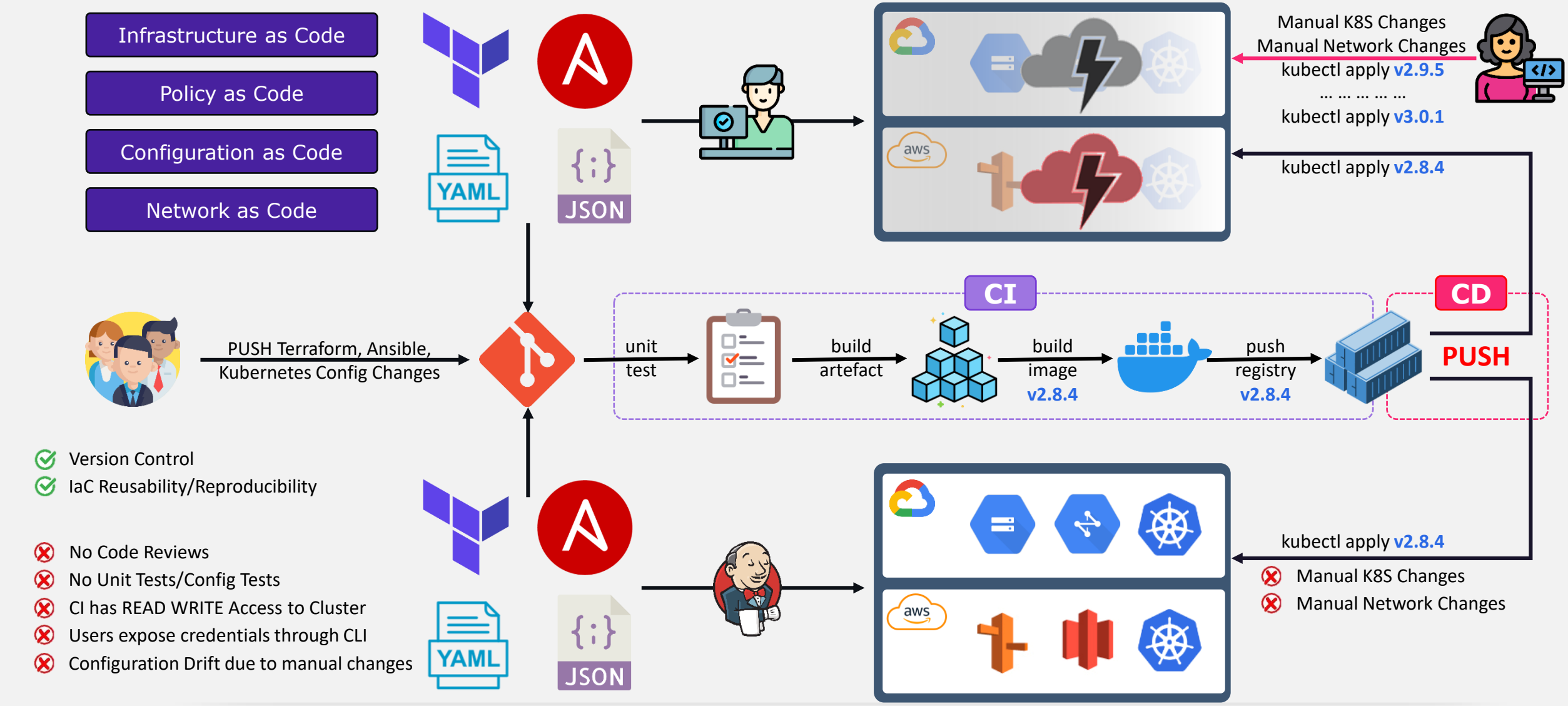
KodeKloud

KodeKloud

Meeting Task Dash

DevOps Team

Meeting Task Dash DevOps Team & Challenges

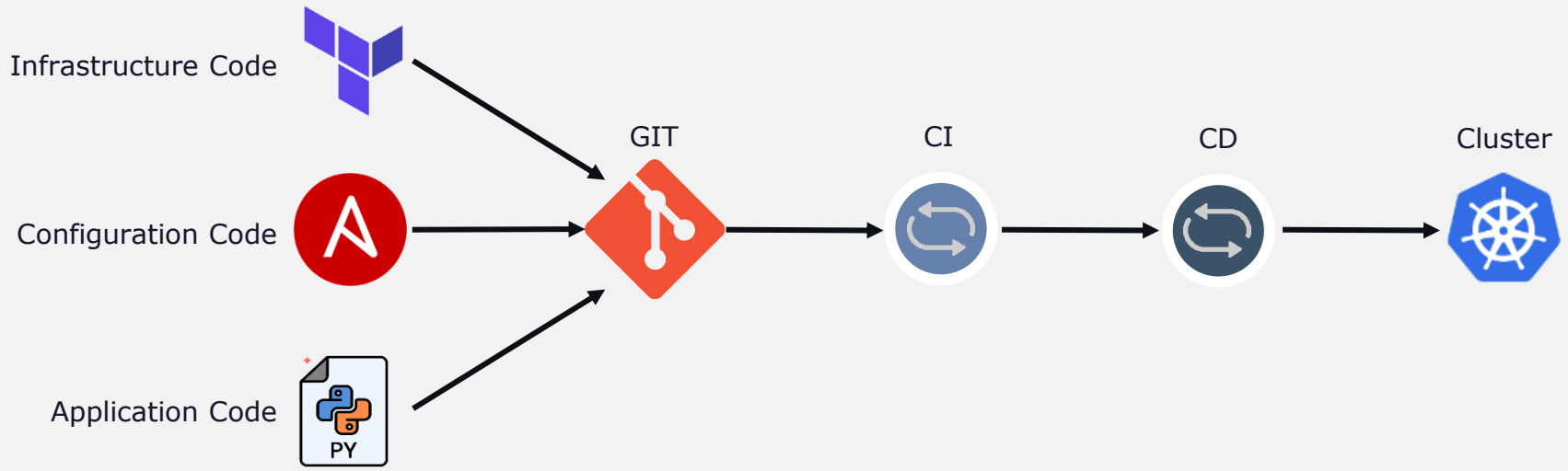


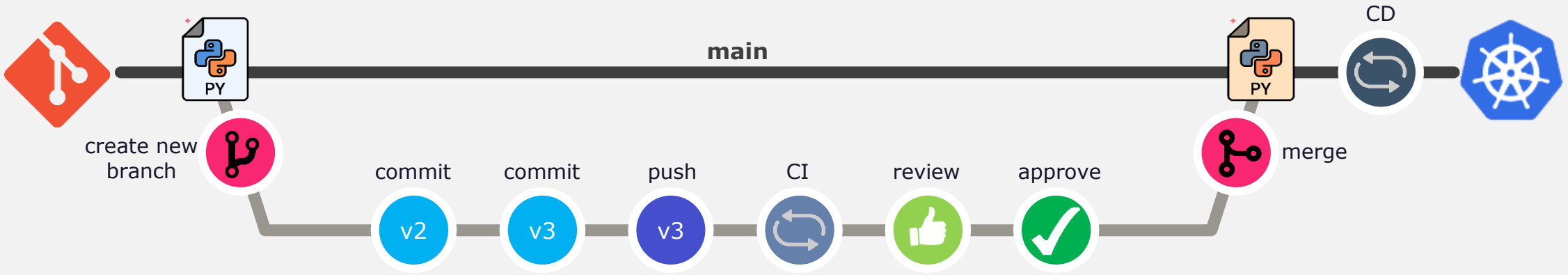
KodeKloud

What is GitOps?

GitOps

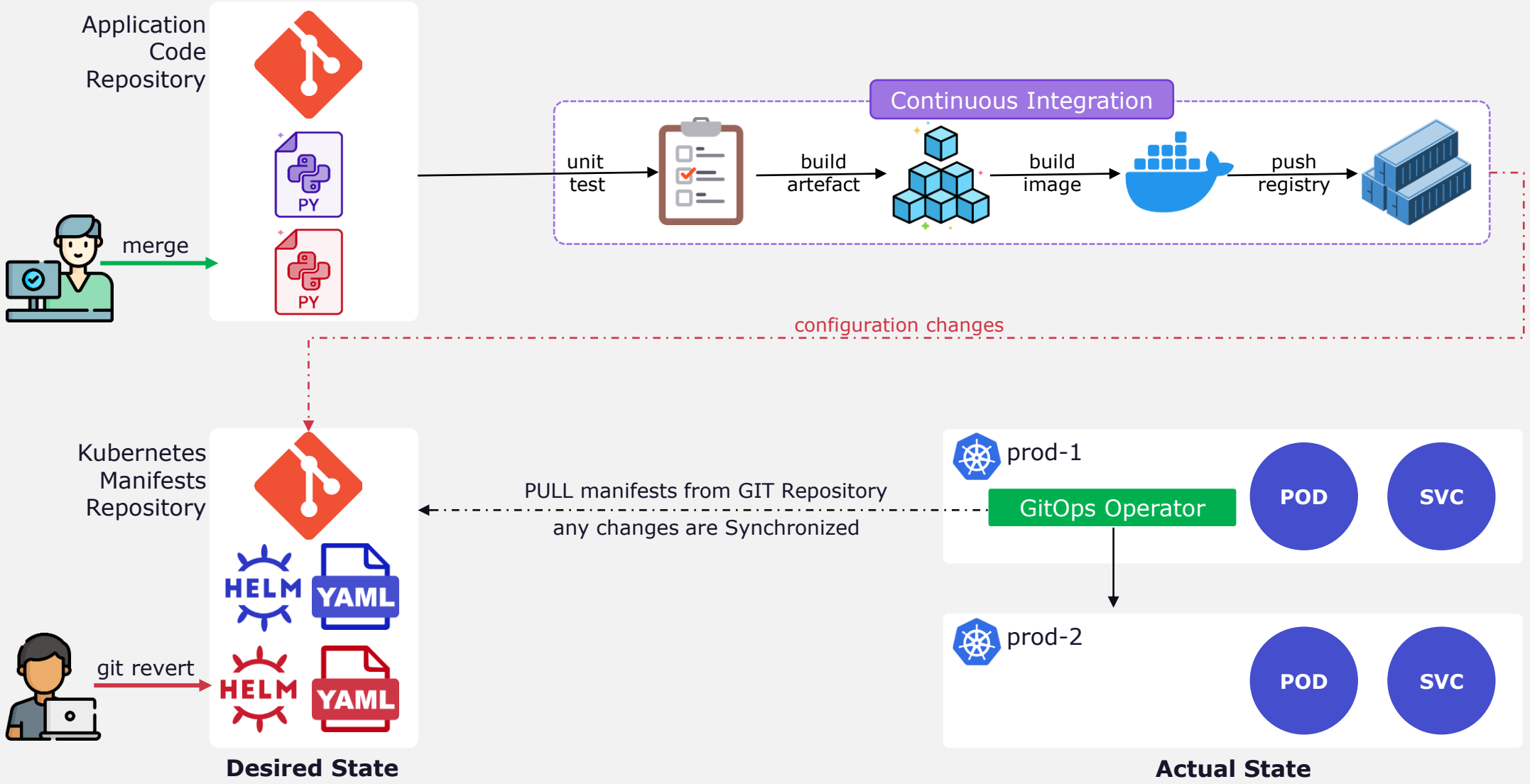
GitOps can be considered an extension of Infrastructure as Code (IaC) that uses Git as the version control system





KodeKloud

GitOps

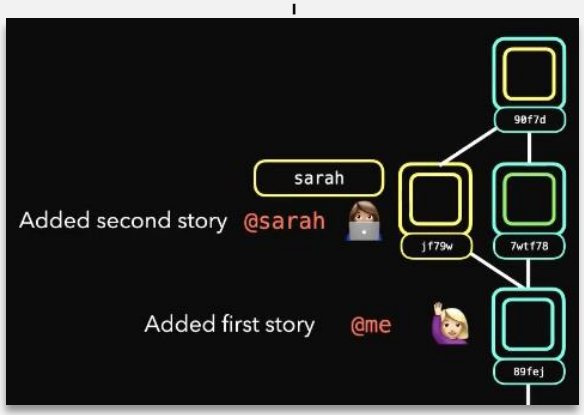


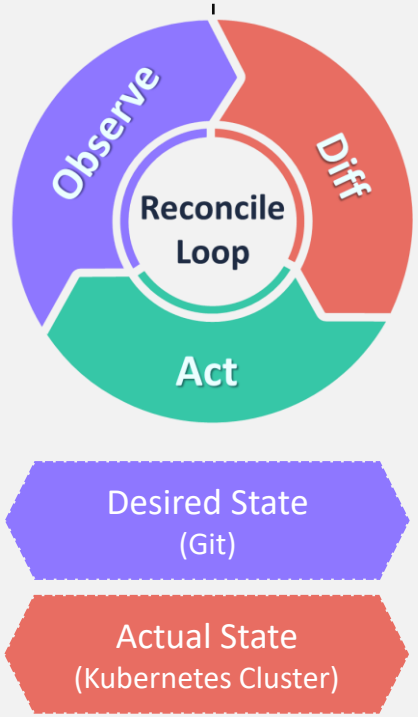
KodeKloud

GitOps Principles

GitOps Principles

```
>_

$ cat deployment.yaml

apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
```





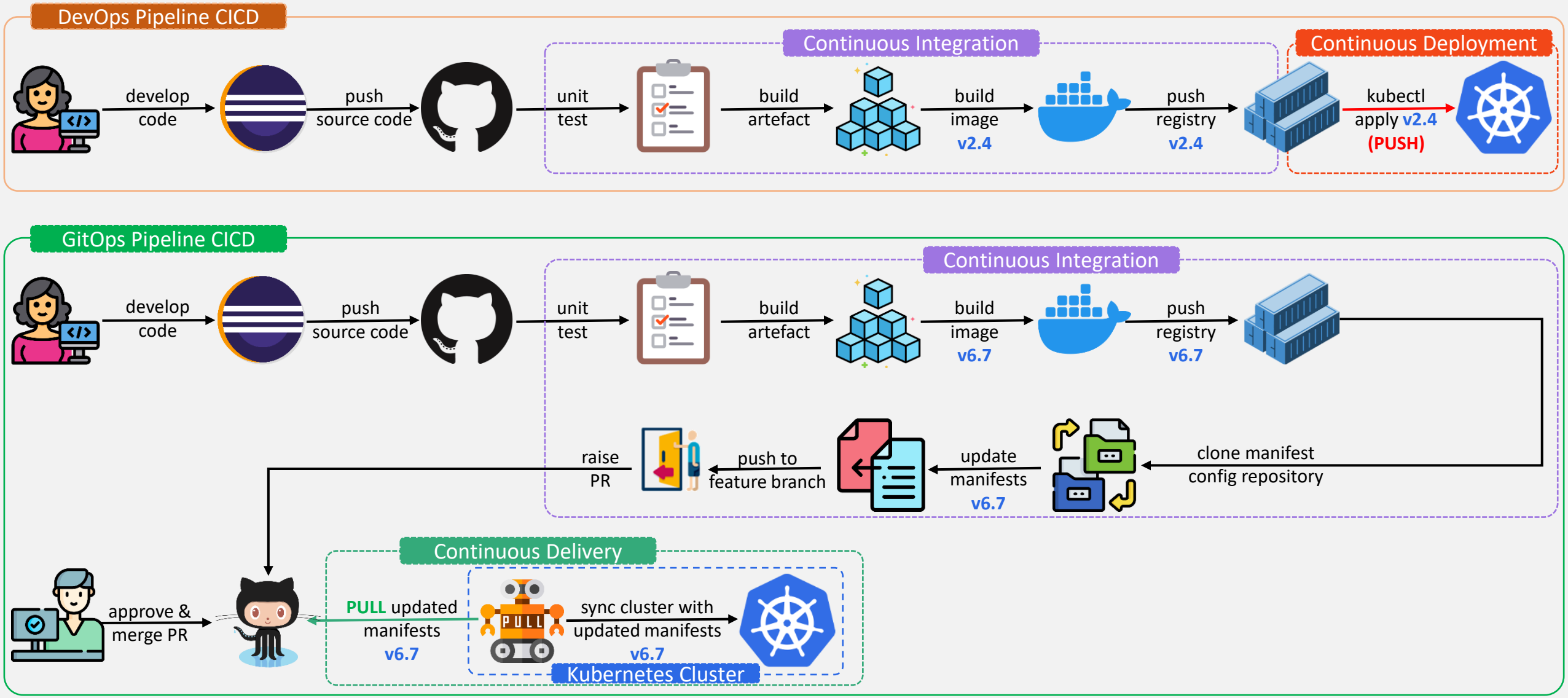
KodeKloud

KodeKloud

DevOps vs GitOps

KodeKloud

DevOps vs GitOps



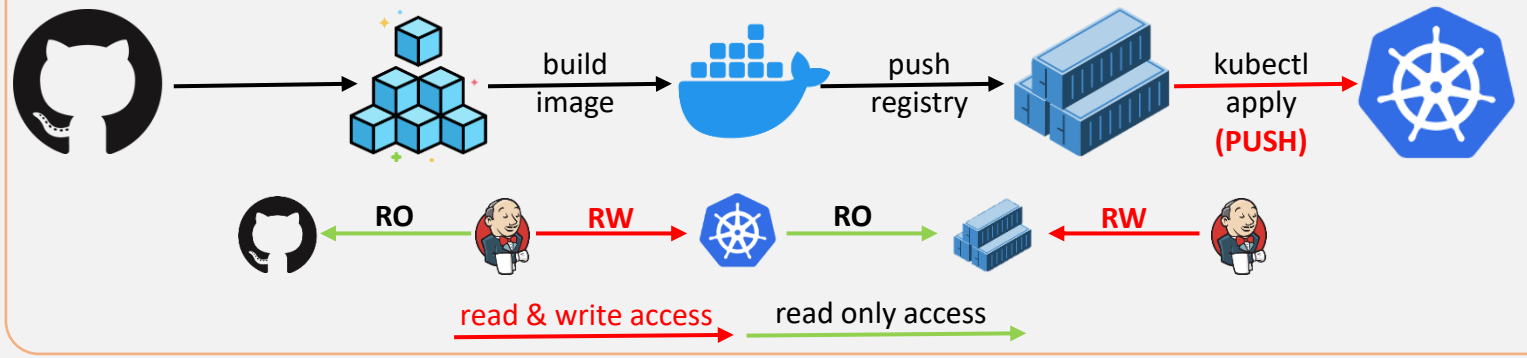
KodeKloud

Push vs Pull based Deployments

KodeKloud

Push vs Pull based Deployments

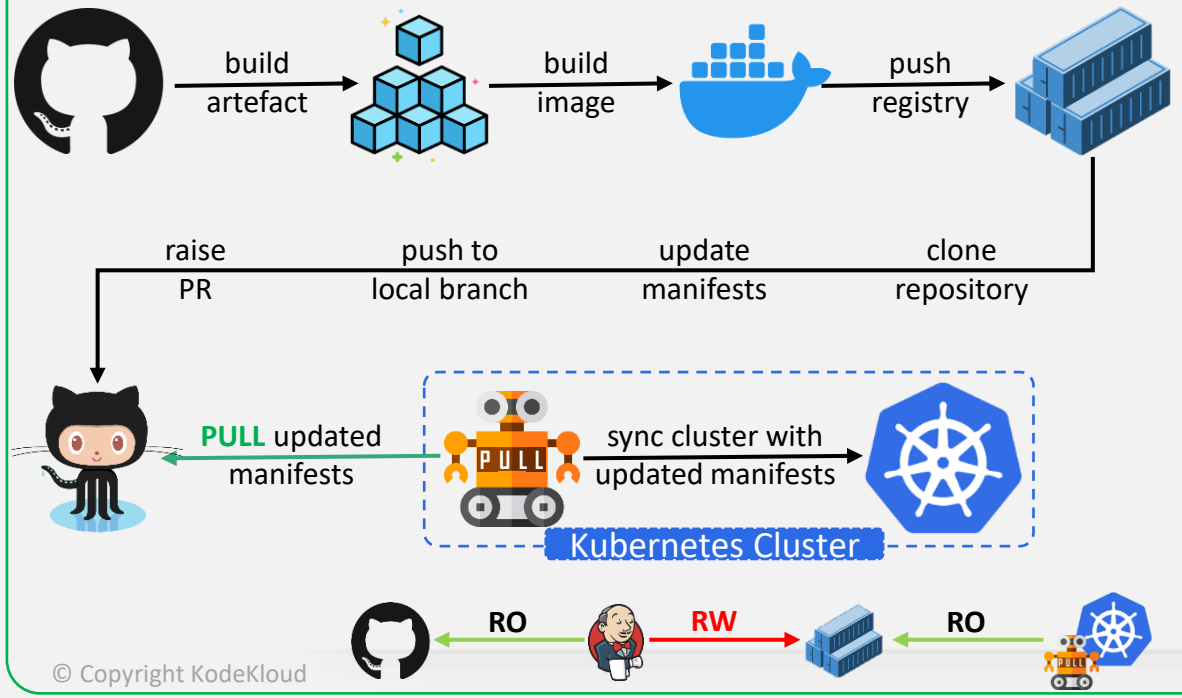
PUSH based Deployment



- 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

KodeKloud

GitOps Feature Set & Usecases

KodeKloud

GitOps Feature Set & Usecases

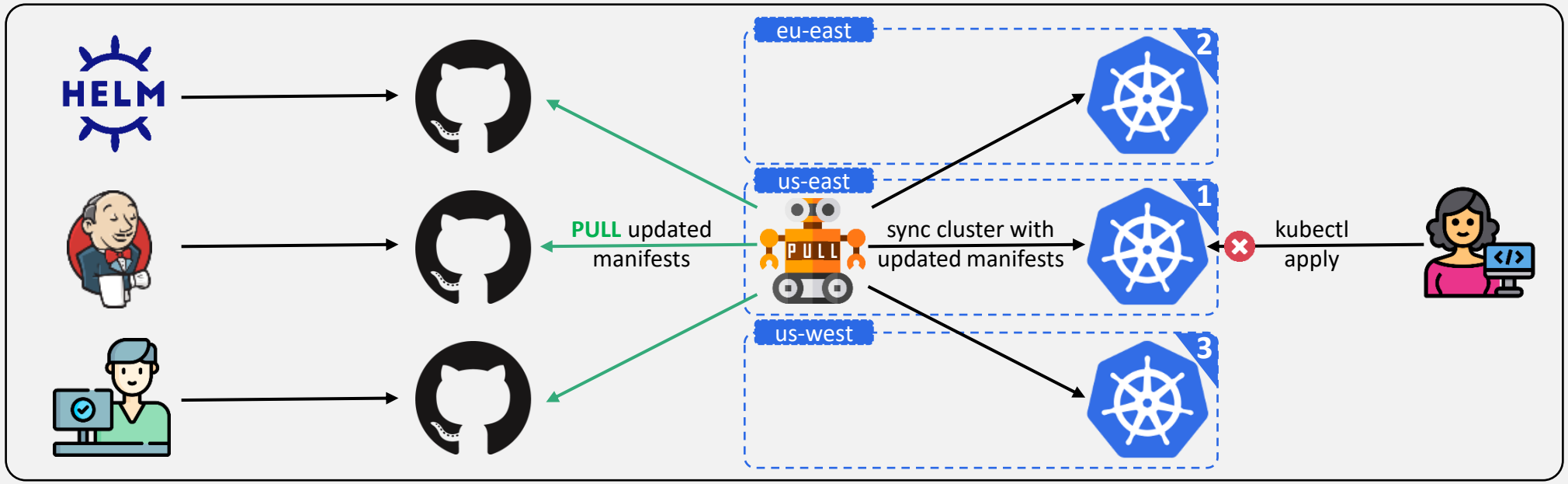
Single Source of
Truth

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

KodeKloud

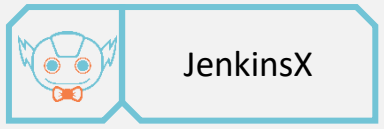
GitOps Benefits & Challenges

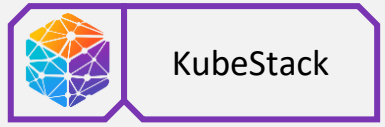
GitOps Benefits & Challenges

KodeKloud

GitOps Projects/Tools

GitOps Projects/Tools





KodeKloud

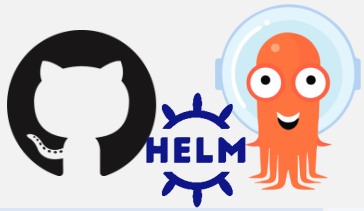
KodeKloud

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
	It can deploy to multiple clusters and is Enterprise-friendly (auditability, compliance, security, RBAC, SSO and lot more)



How ArgoCD works?	It follows the **GitOps** 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			

KodeKloud

ArgoCD Concepts & Terminology

ArgoCD Concepts & Terminology





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

KodeKloud

ArgoCD Features

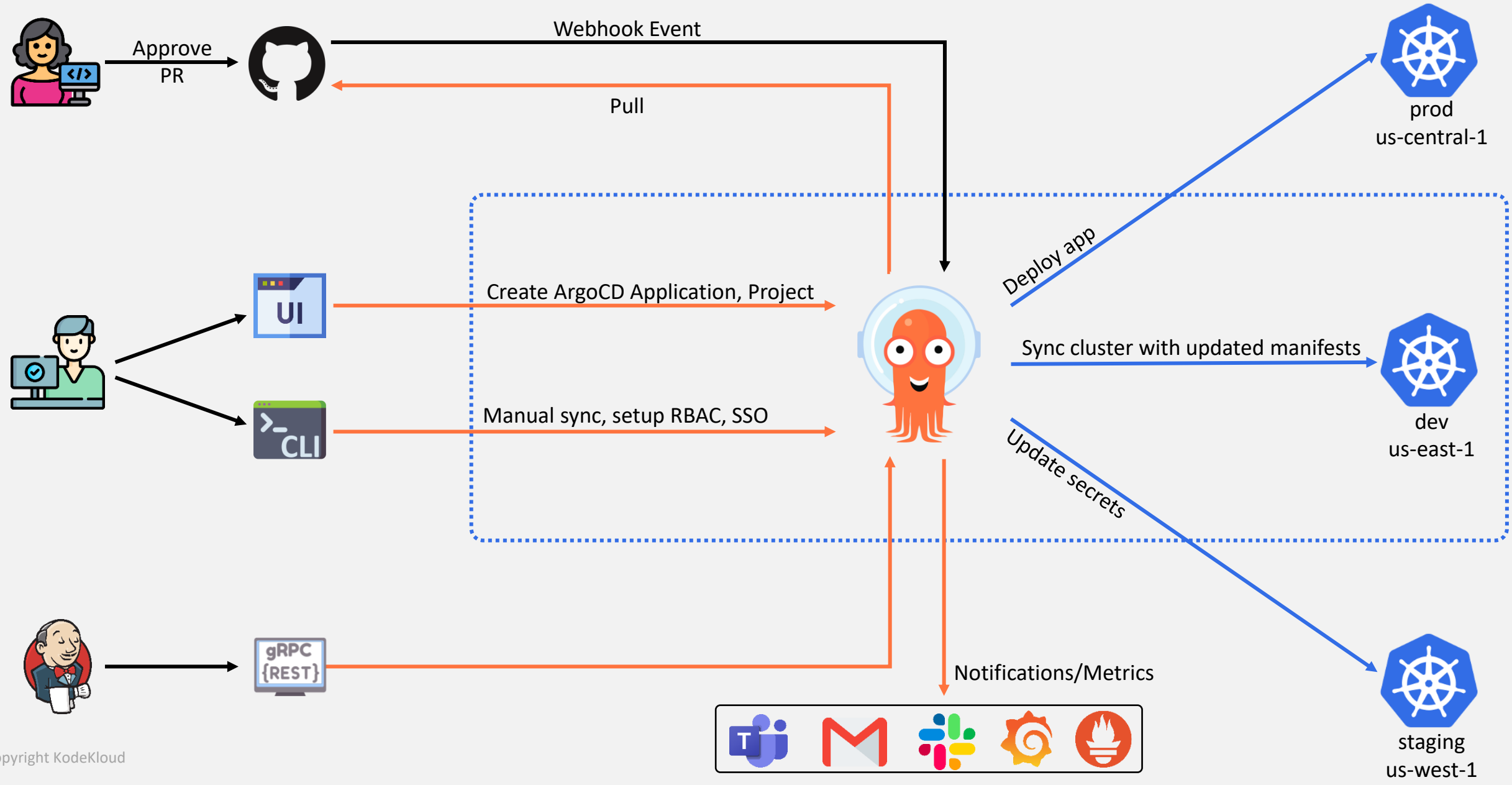
ArgoCD Features

1. Automated deployment of applications to specified target environment in multiple clusters
2. Support for multiple config management/templating 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

KodeKloud

ArgoCD Architecture

ArgoCD Architecture

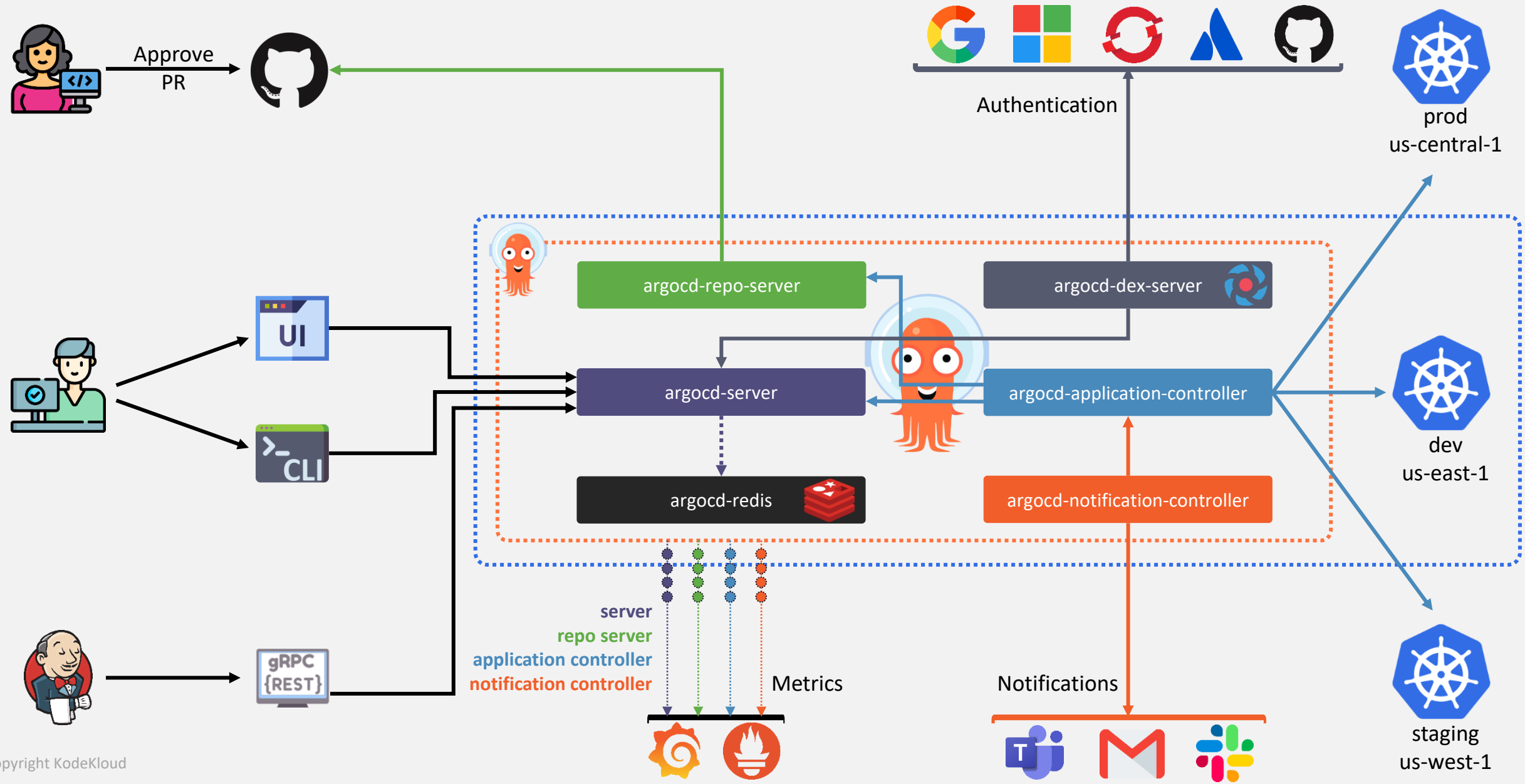


KodeKloud

ArgoCD Architecture
Core Components

KodeKloud

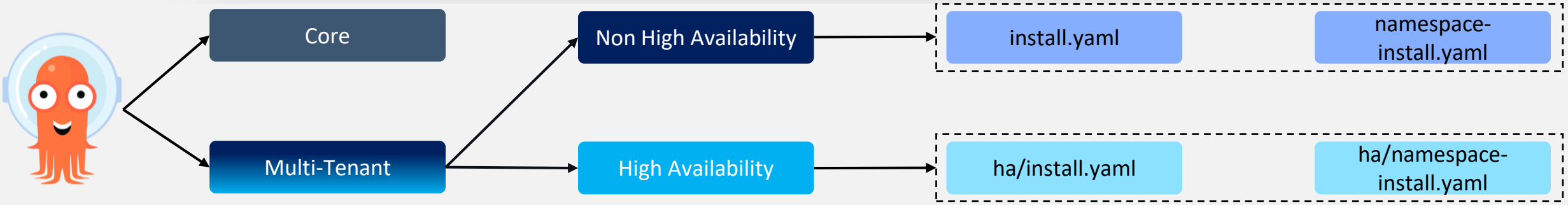
ArgoCD Architecture – Core Components



KodeKloud

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
```

KodeKloud

ArgoCD Application

KodeKloud

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

Project
default

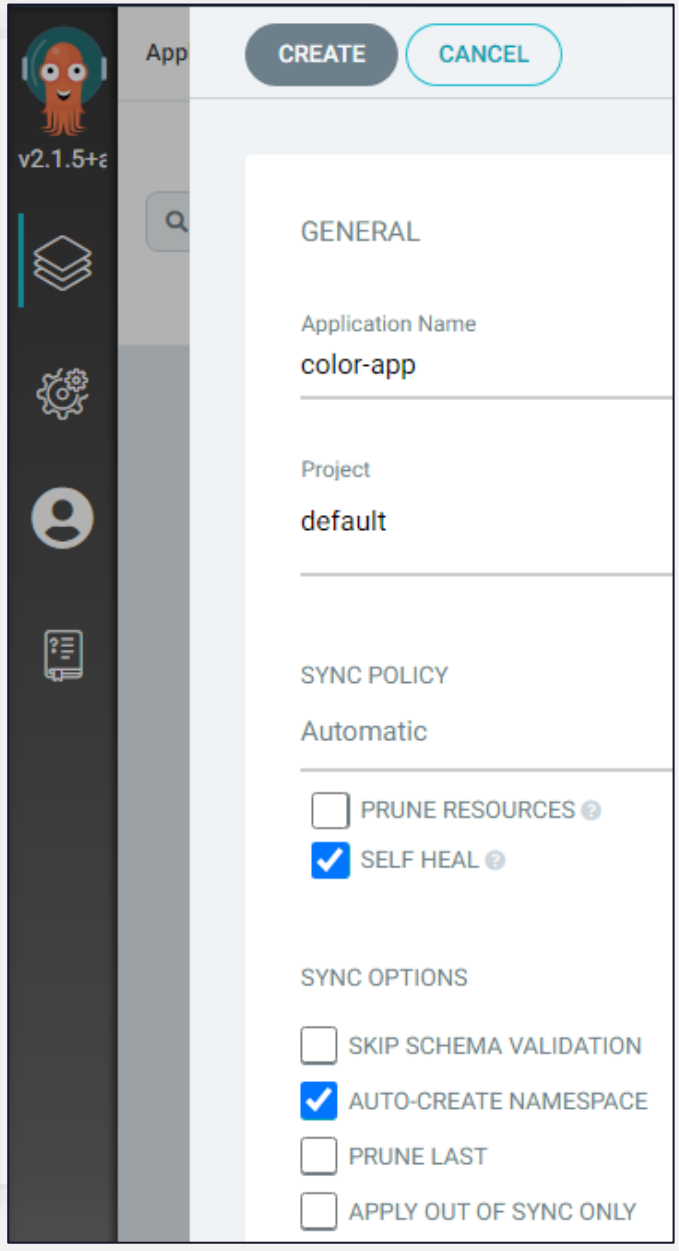
Sync Policy
Automated Options
Sync Options

Ignore Diff
group | kind | jsonPointers

```
$ 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

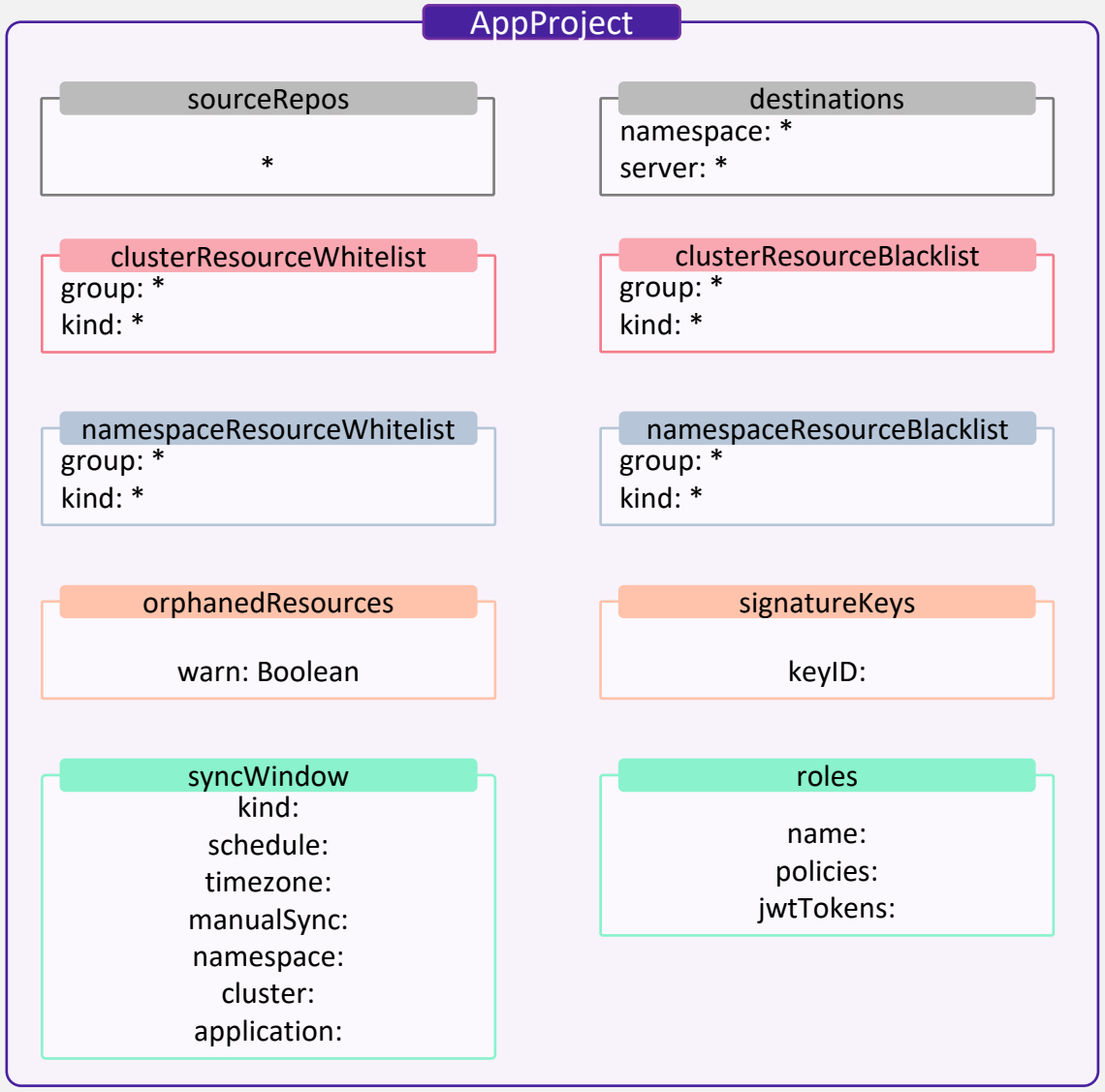
```yaml
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: color-app
  namespace: argocd
spec:
  project: default
  source:
    repoURL: https://github.com/sid/app-1.git
    targetRevision: HEAD
    path: team-a/color
  destination:
    server: https://kubernetes.default.svc
    namespace: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



KodeKloud

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

KodeKloud

Reconciliation Loop

Reconciliation Loop – Using Timeout

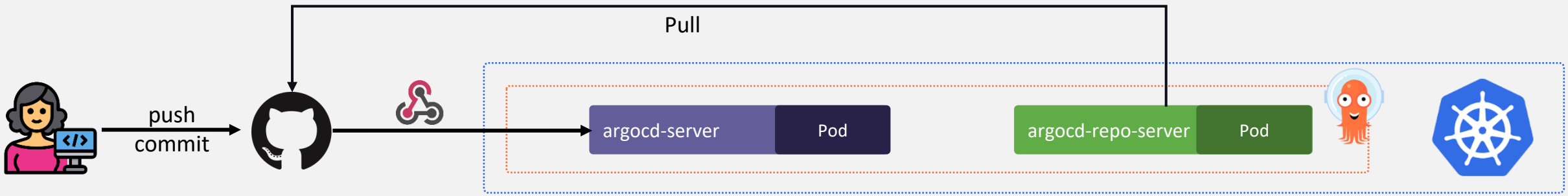

```
>_
$ kubectl -n argocd describe pod argocd-repo-server | grep -i "ARGOCD_RECONCILIATION_TIMEOUT:" -B1
 Environment:
  ARGOCD_RECONCILIATION_TIMEOUT: <set to the key 'timeout.reconciliation' of config map 'argocd-cm'> Optional: true
```

```
>_
$ kubectl -n argocd patch configmap argocd-cm --patch='{"data":{"timeout.reconciliation":"300s"}}'
  configmap/argocd-cm patched
```

```
>_
$ kubectl -n argocd rollout restart deploy argocd-repo-server
  deployment.apps/argocd-repo-server restarted
```

```
>_
$ kubectl -n argocd describe pod argocd-repo-server | grep -i "ARGOCD_RECONCILIATION_TIMEOUT:" -B1
  ARGOCD_RECONCILIATION_TIMEOUT=300s
```

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 *

https:// <<< argocd-ip >>> /api/webhook

Content type

application/json

SSL verification

By default, we verify SSL certificates when delivering payloads.

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

d1d5a79e-e49e-11ec-8ec1-39c21920f0c0

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
```

KodeKloud

Application Health Checks

KodeKloud

Application Health Checks

Healthy	All resources are 100% healthy
Progressing	Resource is unhealthy, but could still be healthy given time
Degraded	Resource status indicates a failure or an inability to reach a healthy state
Missing	Resource is not present in the cluster
Suspended	Resource is suspended or paused. Typical example is a paused Deployment
Unknown	Health assessment failed and actual health status is unknown

Service	Deployment, ReplicaSet, StatefulSet, DaemonSet
Load Balancer	Observed Gen = Deserved Gen
status.loadBalancer.ingress	updated replicas = desired replicas
hostname or IP	

Ingress	PersistentVolumeClaim
status.loadBalancer.ingress	status.phase = Bound
hostname or IP	

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

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

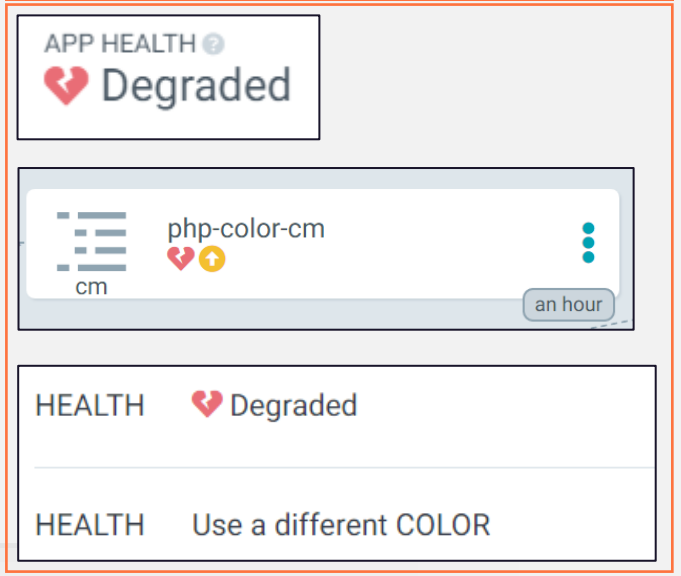
Format - *GROUP_RESOURCE*

apps_Deployment	Secret
batch_CronJob	Namespace
extensions_Ingress	Pod
networking.k8s.io_RuntimeClass	LimitRange
rbac.authorization.k8s.io_Role	PersistentVolumeClaim

>_ argocd-cm.yaml

```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

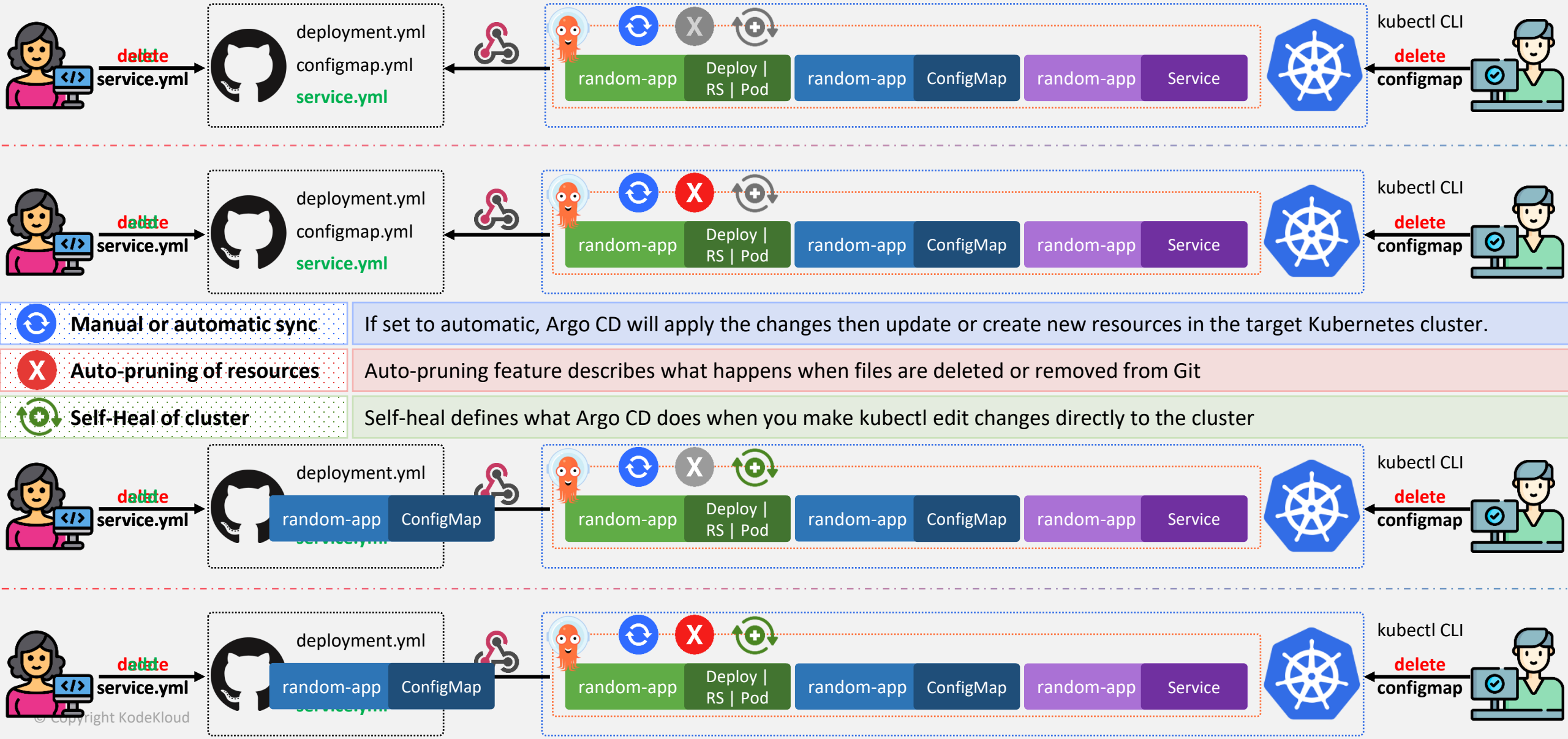


KodeKloud

ArgoCD Sync Strategies

KodeKloud

Sync Strategies



Manual or automatic sync	If set to automatic, Argo CD will apply the changes then update or create new resources in the target Kubernetes cluster.
Auto-pruning of resources	Auto-pruning feature describes what happens when files are deleted or removed from Git
Self-Heal of cluster	Self-heal defines what Argo CD does when you make kubectl edit changes directly to the cluster

KodeKloud

Declarative Setup Mono Application

KodeKloud

Declarative Setup - Mono-Application

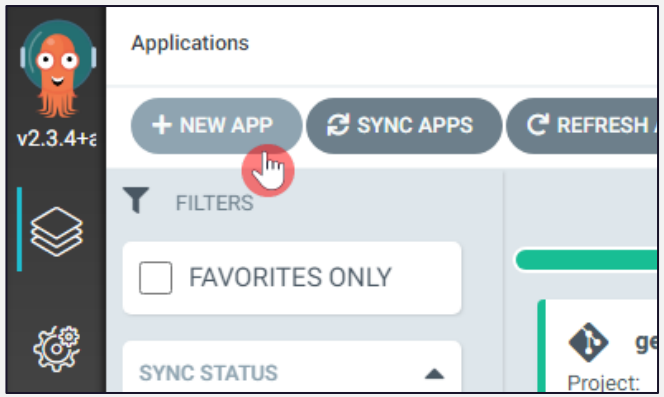
tree structure of Git Repository

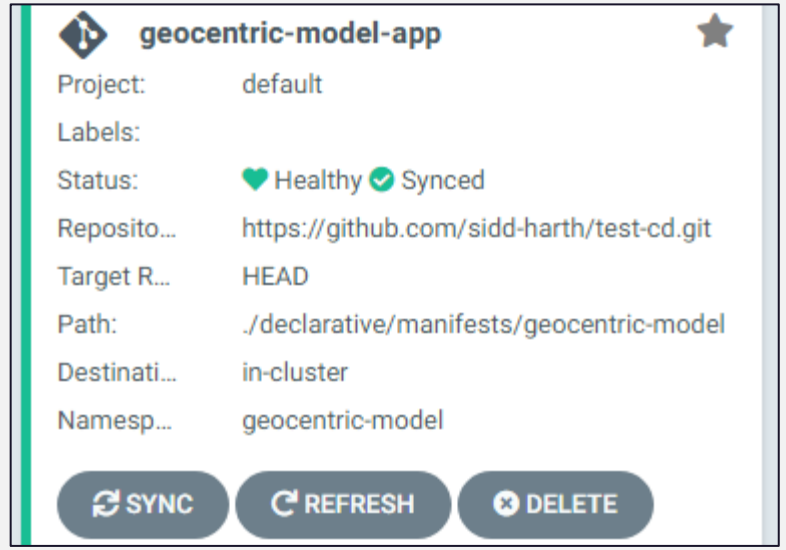
```
declarative
├── manifests
│   └── geocentric-model
│       ├── deployment.yml
│       └── service.yml
└── mono-app
    └── geocentric-app.yml
```

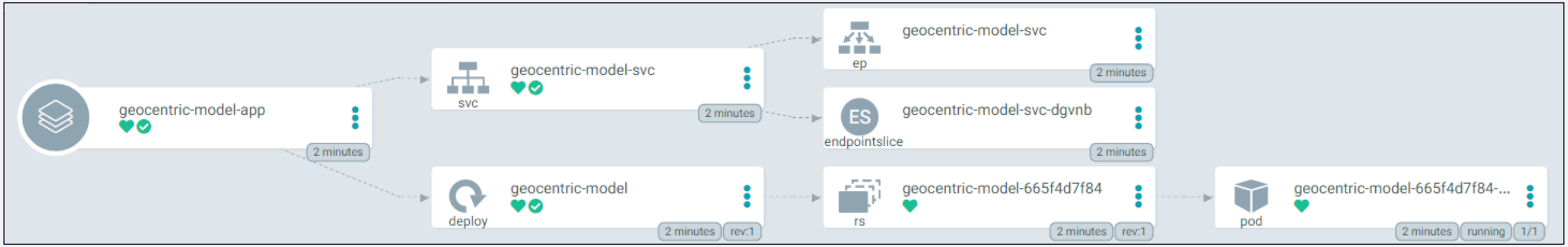
mono-app/geocentric-app.yml

```yaml
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
```







KodeKloud

Declarative Setup App-of-Apps

KodeKloud

Declarative Setup - App-of-Apps

Git Repository

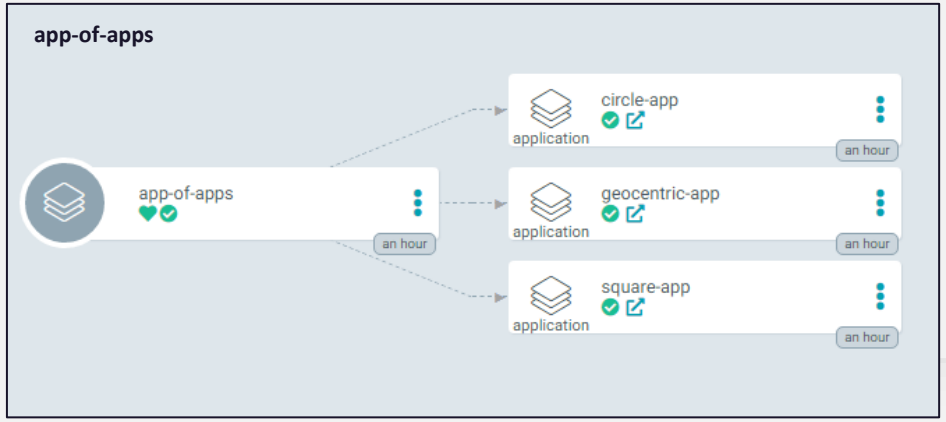
```
declarative
├── app-of-apps
│   ├── circle-app.yml
│   ├── geocentric-app.yml
│   └── square-app.yml
├── manifests
│   ├── circle
│   │   ├── deployment.yml
│   │   └── service.yml
│   ├── geocentric-model
│   │   ├── deployment.yml
│   │   └── service.yml
│   └── square
│       ├── deployment.yml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

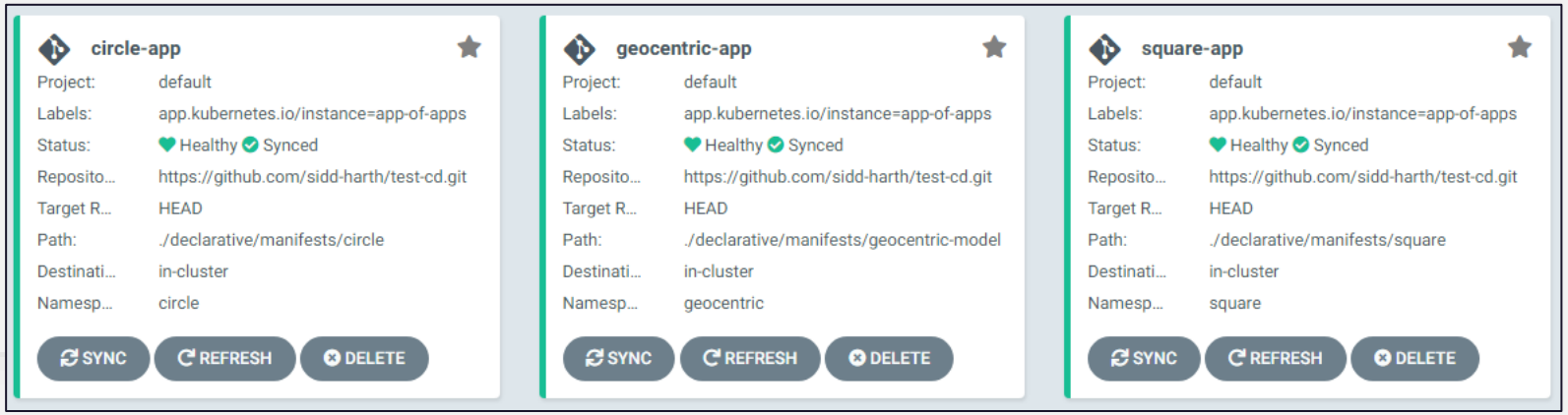
declarative/multi-app/app-of-apps.yml

```yaml
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

```yaml
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
```





KodeKloud

ArgoCD - Deploy HELM Chart

KodeKloud

ArgoCD - Deploy HELM Chart

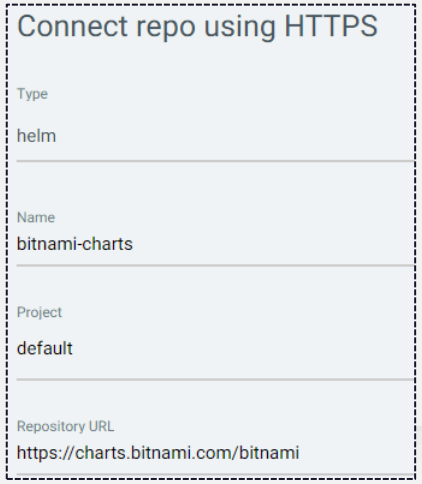
Git Repository

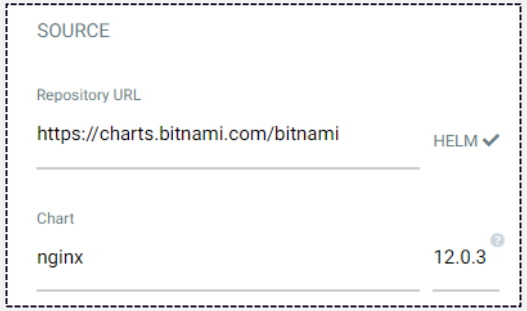
```
helm-chart
├── Chart.yaml
├── templates
│   ├── NOTES.txt
│   ├── _helpers.tpl
│   ├── configmap.yaml
│   ├── deployment.yaml
│   └── service.yaml
└── values.yaml
```

```
$ 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
```

```
$ 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
```

KodeKloud

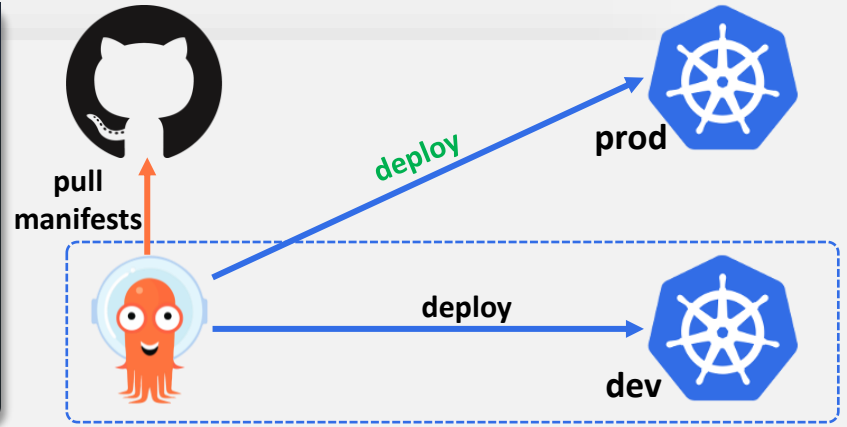
ArgoCD Multi-Cluster Deployment

KodeKloud

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

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
```

Alice
(admin)

```
$ 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
```

Jai
(Add Clusters)

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

Ali
(Project Kia Admin)

Default Roles | By default, all new users have no access | role:readonly and role:admin | policy.default from the argocd-rbac-cm ConfigMap

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

Kia Team

KodeKloud

RBAC - Role Based Access Control

KodeKloud

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>

Alice
(admin)

```
$ 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
```

Jai
(Add Clusters)

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

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

Ali
(Project Kia Admin)

```
$ 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
```

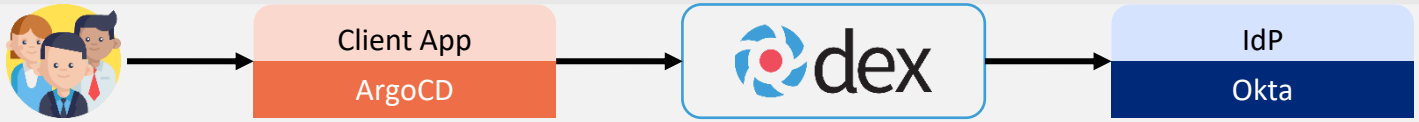
Kia Team

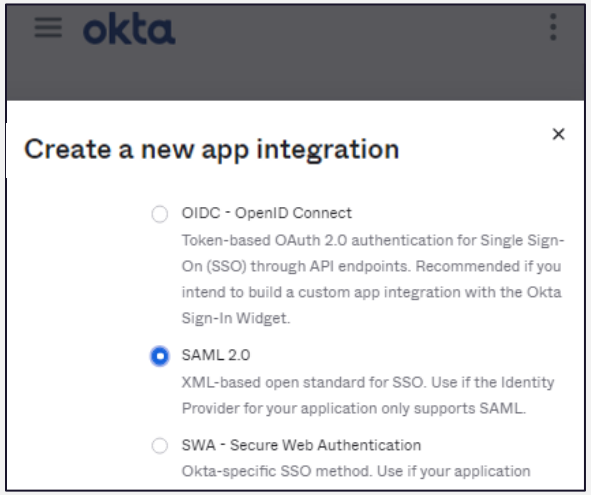
KodeKloud

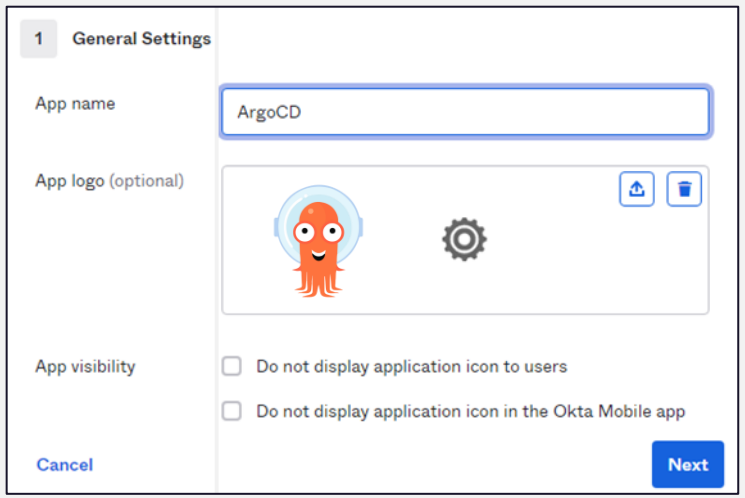
Dex GitHub Connector

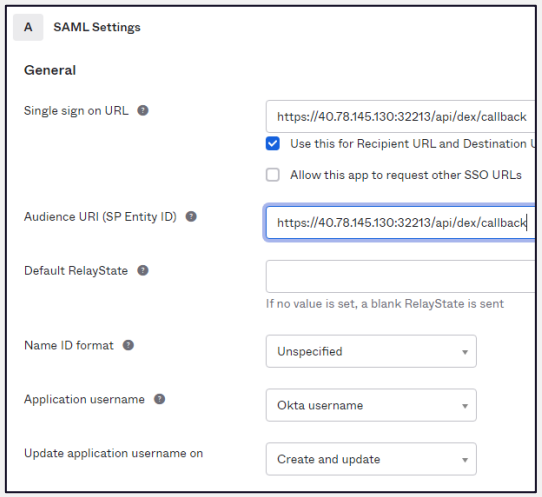
KodeKloud

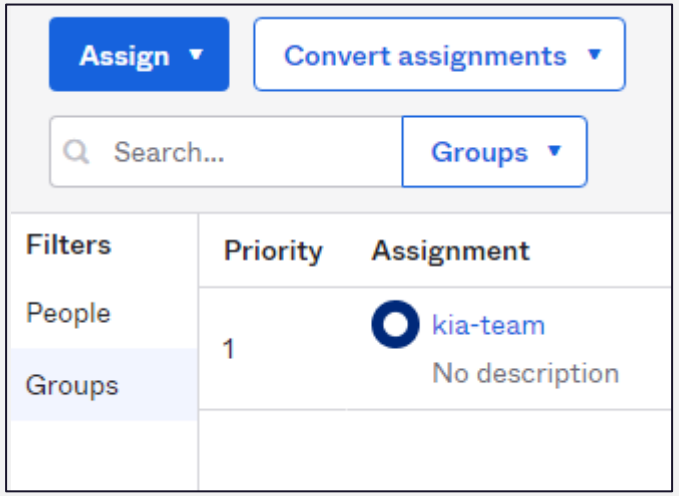
Dex GitHub Connector

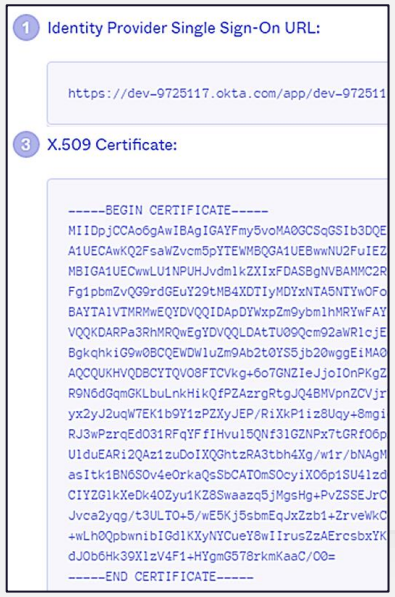




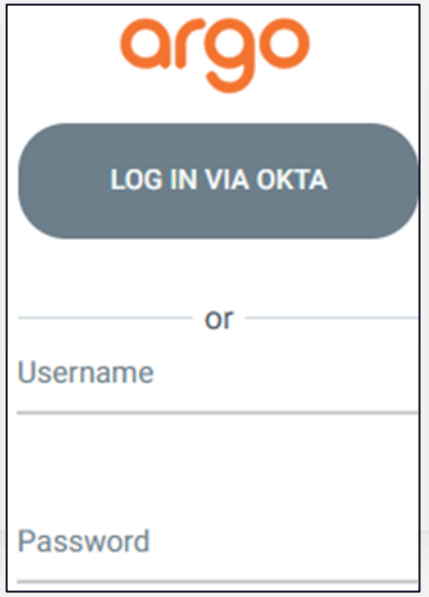









```
$ kubectl -n argocd edit configmap argocd-cm
… … …
dex.config: |
 connectors:
  - type: saml
    id: okta
    name: Okta
    config:
      ssoURL: <okta-idp-sso-url>
      caData: |
          <base64encoded X.509 Certificate>
      usernameAttr: name
      emailAttr: email
      groupsAttr: groups
```




```
$ 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
```

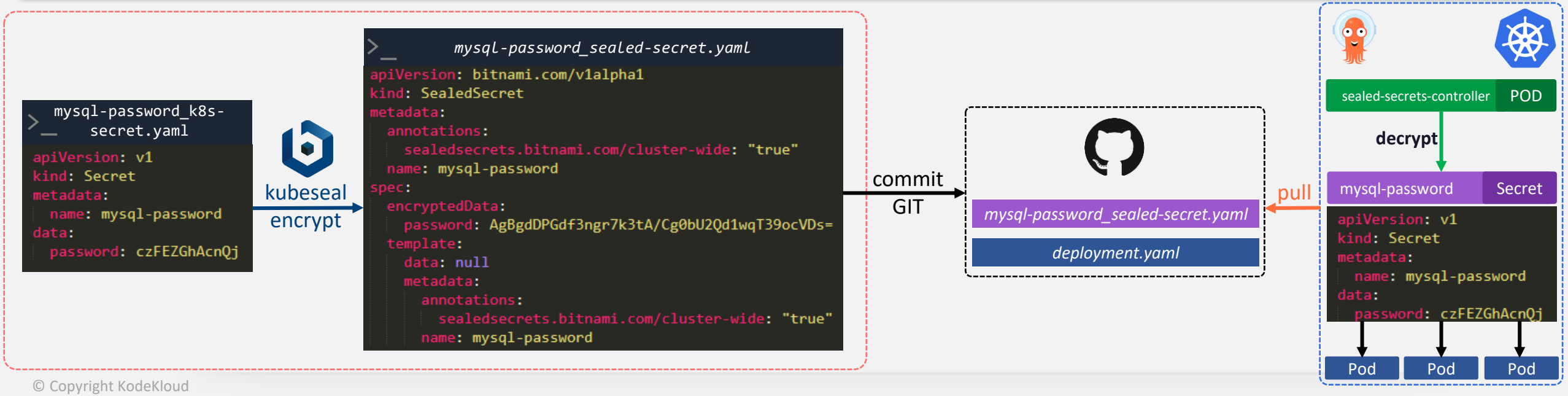
KodeKloud

Bitnami Sealed Secret with ArgoCD

KodeKloud

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
```



KodeKloud

HashiCorp Vault with ArgoCD Vault Plugin

KodeKloud

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:
    avp.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:
    avp.kubernetes.io/path: "crds/data/mysql"
type: Opaque
stringData:
  password: 1234567
```

KodeKloud

ArgoCD with ArgoCD Vault Plugin

KodeKloud

HashiCorp Vault with ArgoCD Vault Plugin – Part 2

argocd-repo-server

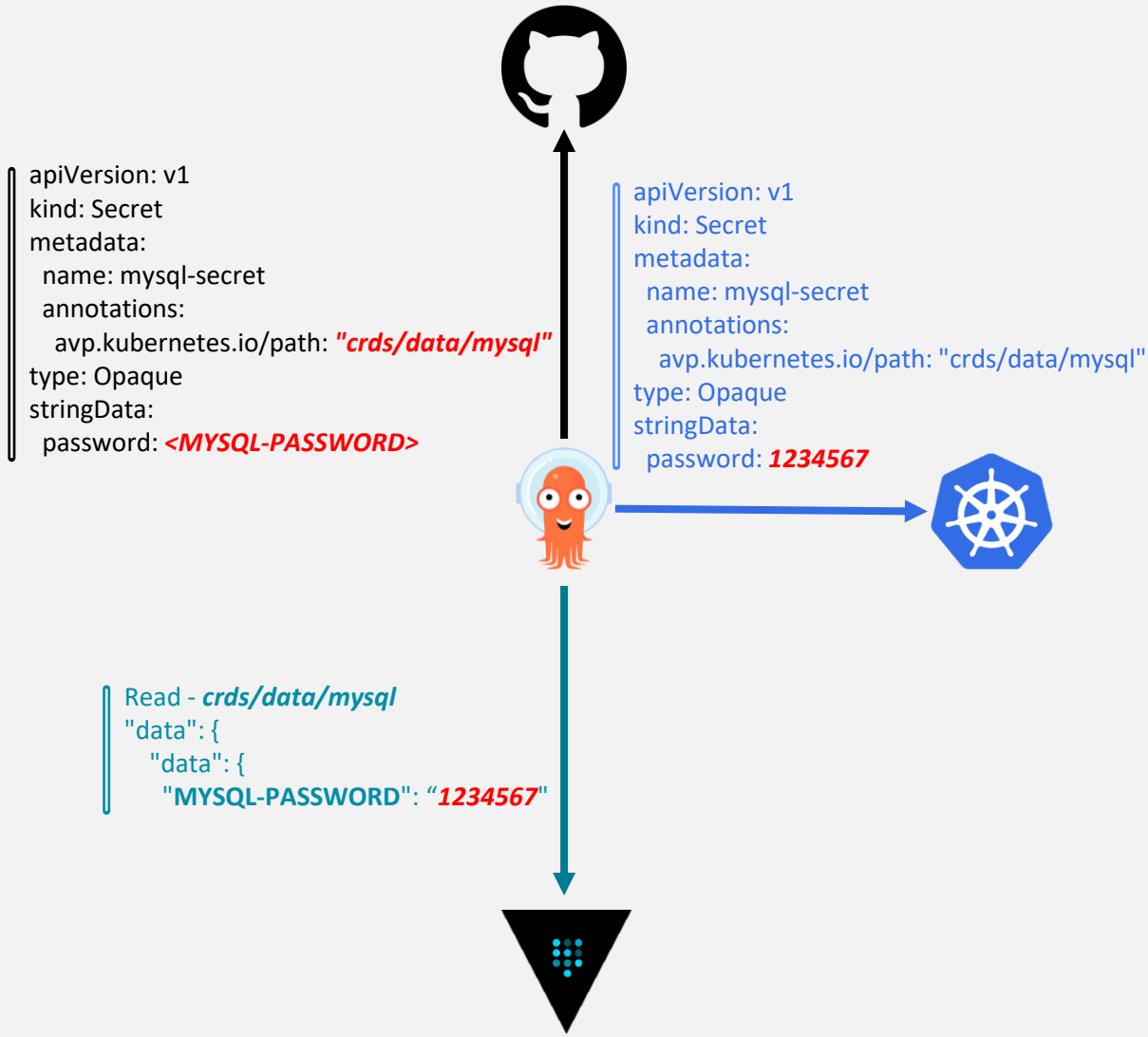
```
… … …
containers:
  - name: argocd-repo-server
    volumeMounts:
      - name: custom-tools
        mountPath: /usr/local/bin/argocd-vault-plugin
volumes:
  - name: custom-tools
    emptyDir: {}
initContainers:
  - name: download-tools
    image: 'alpine:3.8'
    command: [- sh,  -c ]
    args:
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

argocd-cm

```
… … …
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

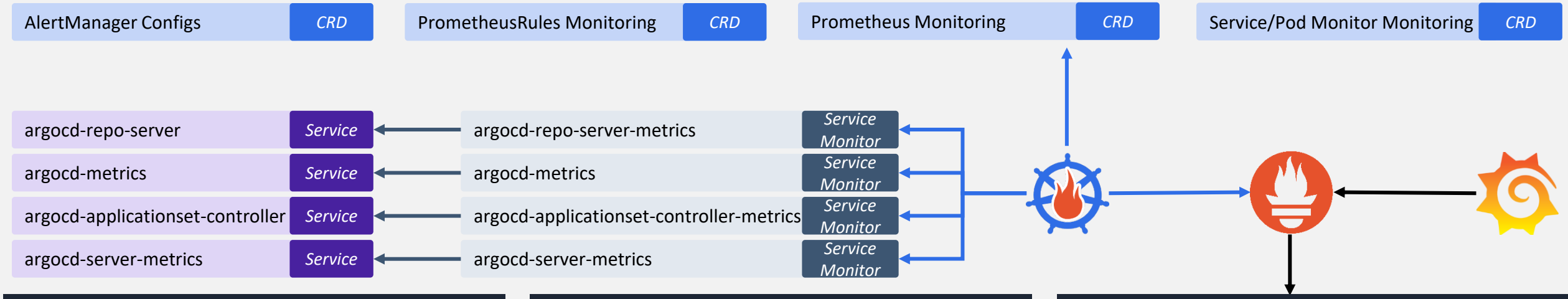


KodeKloud

ArgoCD Metrics
Prometheus + Grafana

KodeKloud

Prometheus + Grafana – ArgoCD




```
$ kubectl get svc argocd-server-metrics -o yaml
apiVersion: v1
kind: Service
metadata:
  name: argocd-server-metrics
  namespace: argocd
spec:
  ports:
  - name: metrics
    port: 8083
    protocol: TCP
    targetPort: 8083
  selector:
    app.kubernetes.io/name: argocd-server
  type: ClusterIP
```

```
$ kubectl get servicemonitor argocd-server-metrics -o yaml
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: argocd-server-metrics
  labels:
    release: prometheus-operator
spec:
  selector:
    matchLabels:
      app.kubernetes.io/name: argocd-server-metrics
  endpoints:
  - port: metrics
```

```
$ kubectl exec -it prometheus-0 -c config-reloader -- /bin/sh

$ cat /etc/prometheus/config_out/prometheus.env.yaml
global:
  scrape_interval: 30s
rule_files:
  /etc/prometheus/rules/prometheus-0/*.yaml
scrape_configs:
- job_name: serviceMonitor/monitoring/kube-apiserver/0
- job_name: serviceMonitor/argocd/argocd-server-metrics/0
- job_name: serviceMonitor/argocd/argocd-repo-server-metrics/0
- job_name: serviceMonitor/argocd/argocd-metrics/0
- job_name: serviceMonitor/argocd/argocd-applicationset-controller-metrics/0
```

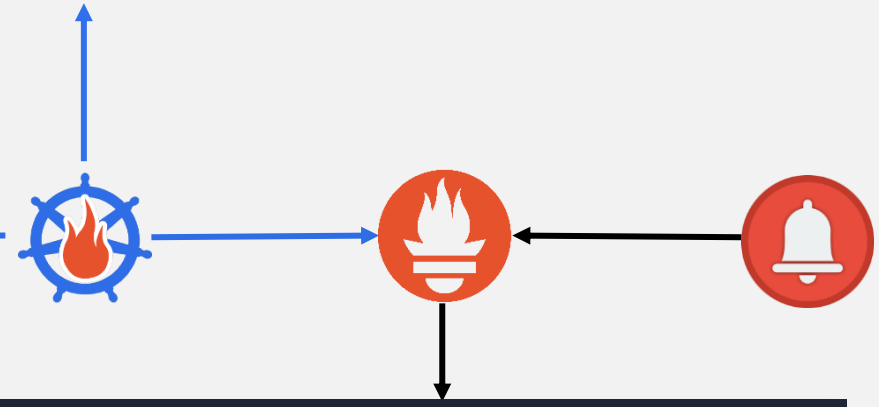
KodeKloud

ArgoCD Metrics
Prometheus + AlertManager

KodeKloud

Prometheus + Alertmanager – ArgoCD

AlertManager Configs CRD | Prometheus Monitoring CRD | PrometheusRules Monitoring CRD | Service/Pod Monitor Monitoring CRD



```
apiVersion: monitoring.coreos.com/v1
kind: PrometheusRule
metadata:
  creationTimestamp: null
  labels:
    prometheus: example
    role: alert-rules
  name: prometheus-argocd-rules
spec:
   - name: ArgoCD Rules
     rules:
       - alert: ArgoApplicationOutOfSync
         expr: argocd_app_info{sync_status="OutOfSync"} == 1
         for: 5m
         labels:
           severity: warning
         annotations:
            summary: "'{{ $labels.name }}' Application has
                      synchronization issue"
```

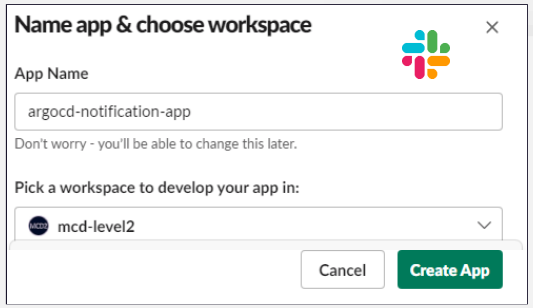
```
$ kubectl exec -it prometheus-0 -c  config-reloader -- /bin/sh

$ cat /etc/prometheus/rules/prometheus-rulefiles-0/argocd-app-sync
- name: ArgoCD Rules
    rules:
    - alert: ArgoApplicationOutOfSync
      annotations:
        summary: '''{{ $labels.name }}'' Application has
           synchronization issue''
      expr: argocd_app_info{sync_status="OutOfSync"} == 1
      for: 1m
      labels:
        severity: warning
…. …. …..  …
```

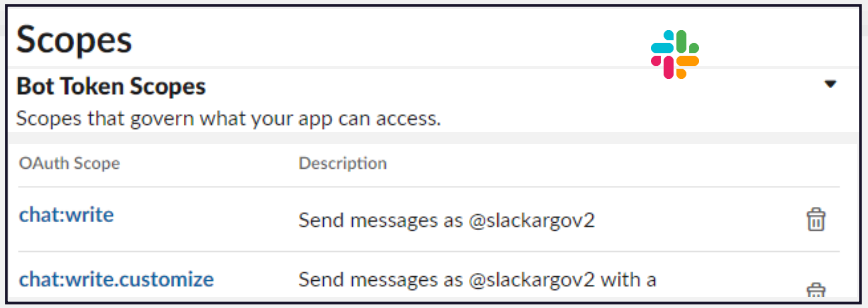
KodeKloud

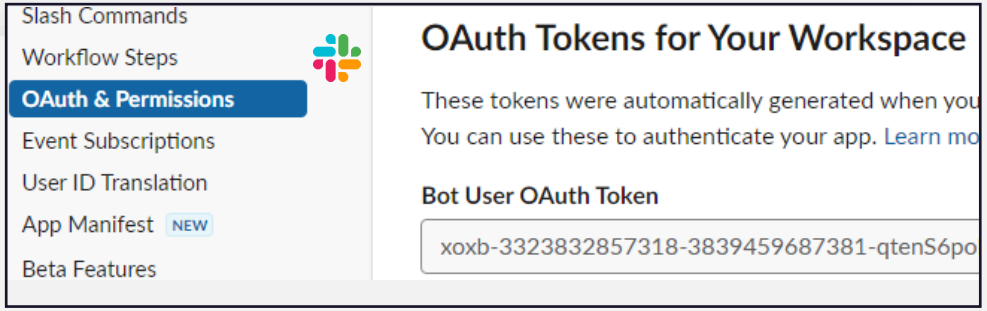
Notifications

KodeKloud

Notifications - ArgoCD



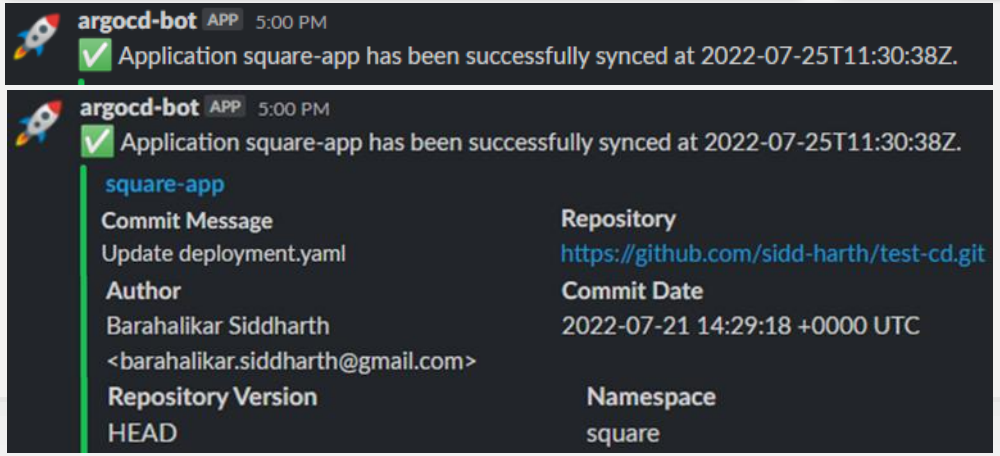


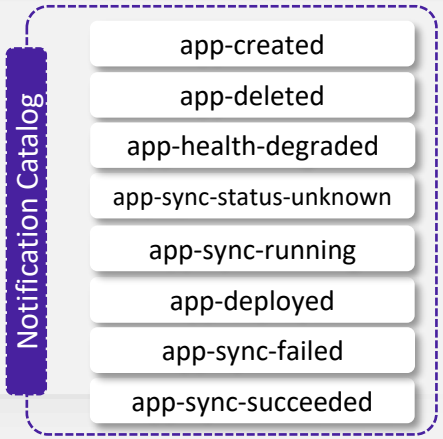



```
$ kubectl edit configmap argocd-notifications-cm
apiVersion: v1
kind: ConfigMap
metadata:
  name: argocd-notifications-cm
data:
  service.slack: |
    token: $slack-token
    username: argocd-bot
    icon: ":rocket:"
  trigger.on-sync-succeeded: |
        - when: app.status.sync.status == 'Synced'
          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>
```





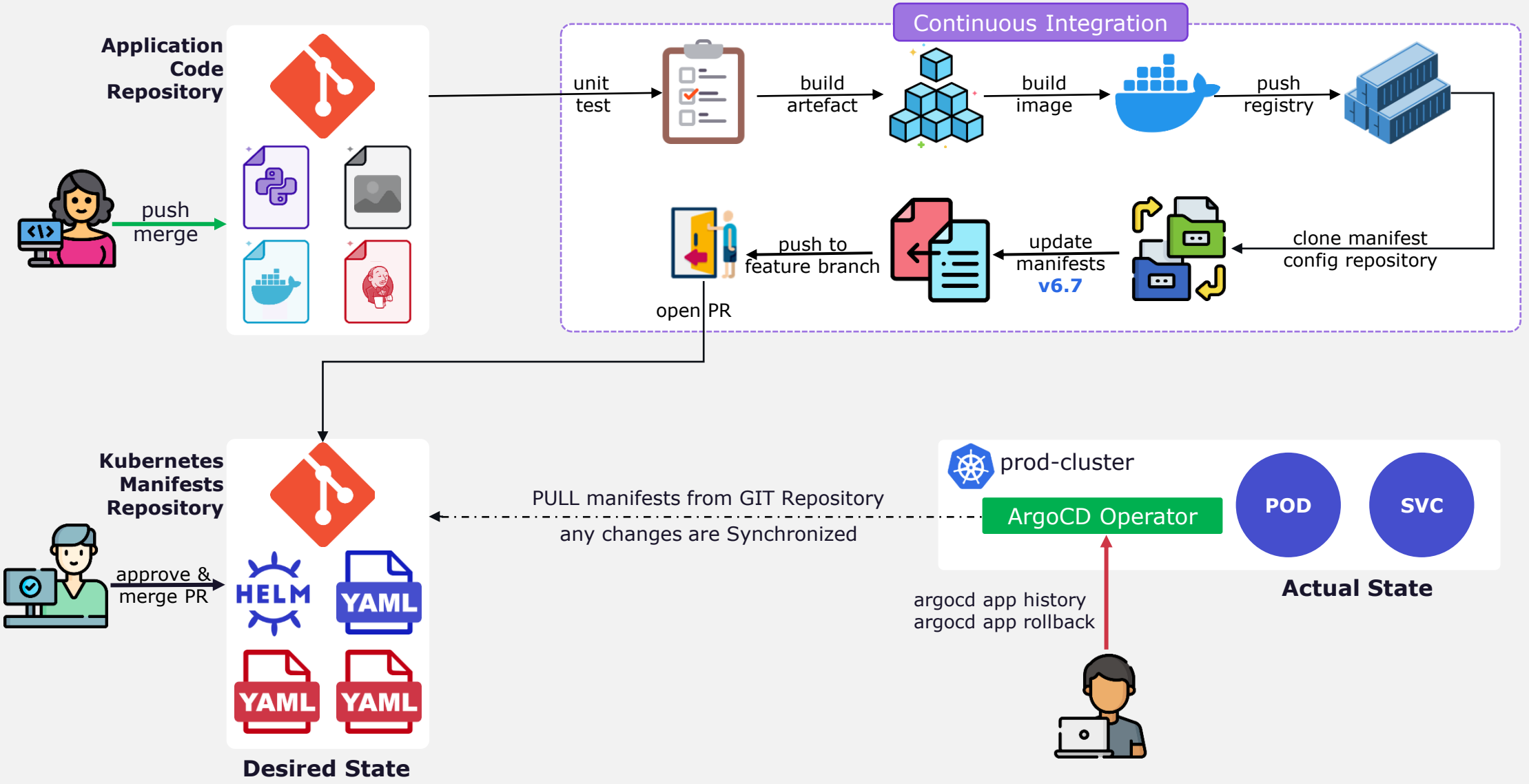
KodeKloud

KodeKloud

CI/CD with GitOps

KodeKloud

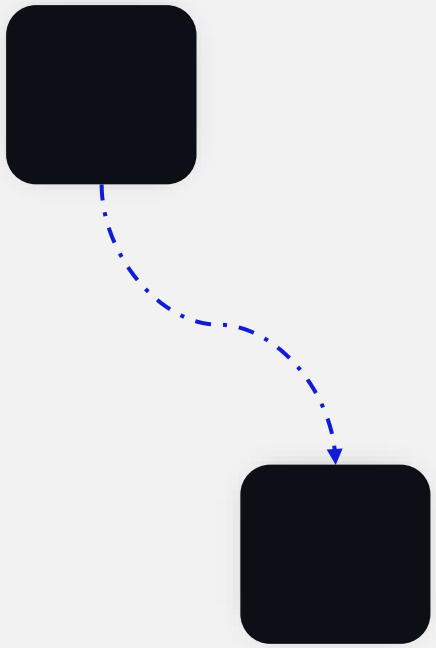
CI/CD with GitOps



KodeKloud

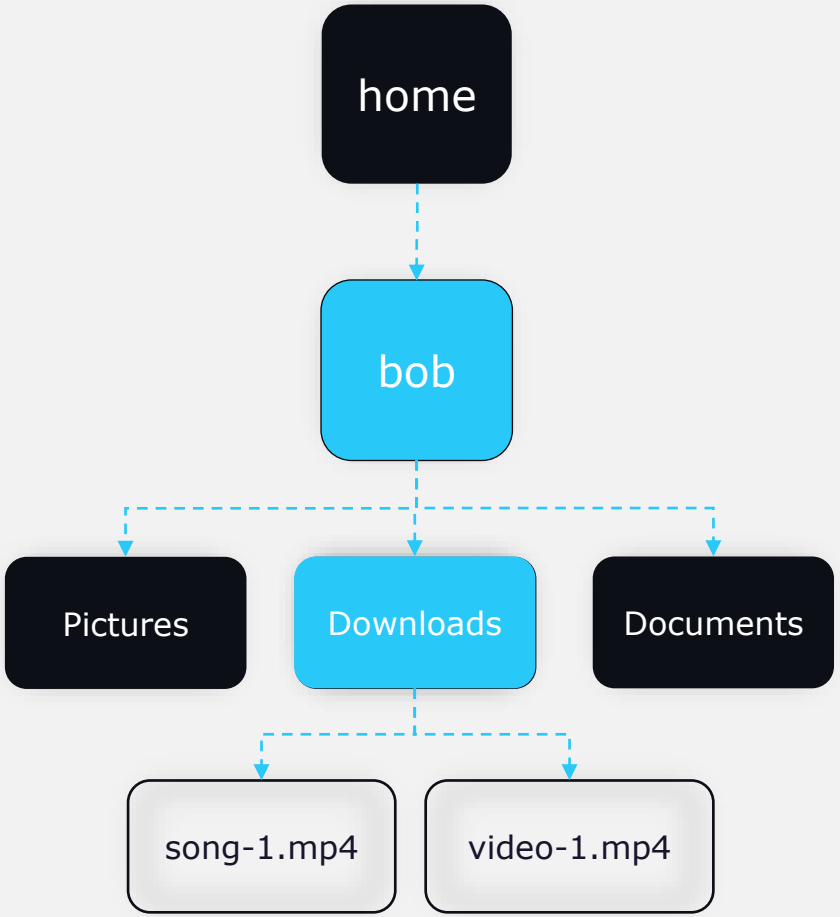
KodeKloud

Navigating the File System



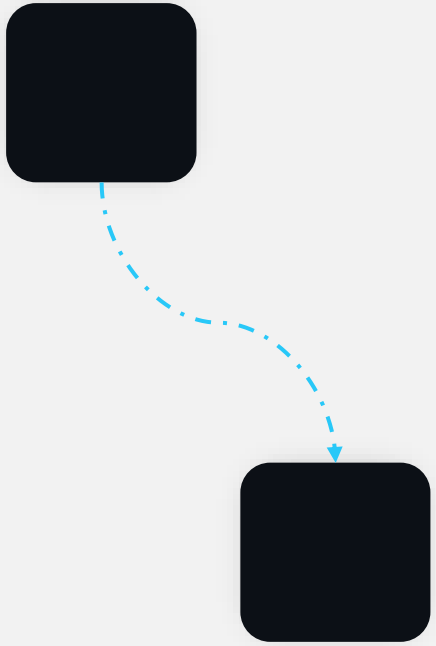
Navigating the Filesystem

```
>_

$ pwd
/home/bob

$ ls
Pictures Downloads Documents

$ cd Downloads

$ ls
song-1.mp4    video-1.mp4

$ cd ..
```



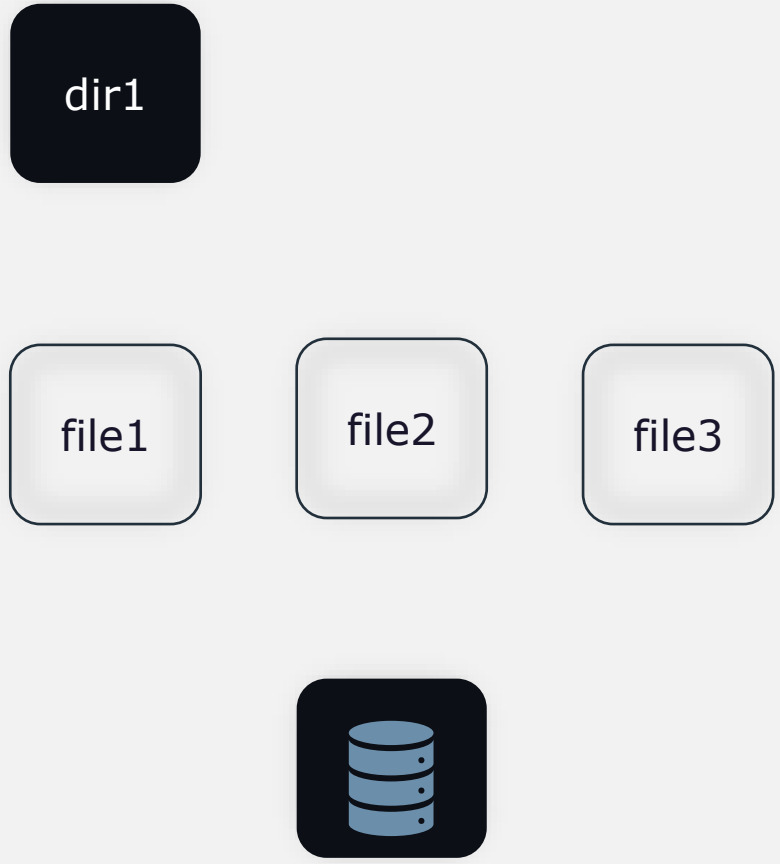
KodeKloud

Creating/Deleting Files



Command & Illustration

```
$ ls
File1 file2 file3
```



KodeKloud