Installing and Operating Argo CD

Steve Buchanan
CLOUD ARCHITECT

@buchatech | www.buchatech.com

Overview

Requirements for Argo CD

Deploying Argo CD

Accessing the Argo CD API Server

Using the Argocd Command Line Interface (CLI)

Upgrading Argo CD

Setting up RBAC for Argo CD

Configuring User Management in Argo CD

Setting up Secrets Management with Argo CD

High Availability, Backup, and Disaster Recovery with Argo CD

Configuring Webhooks with Argo CD

Monitoring and Notifications with Argo CD

Requirements for Argo CD

Argo CD Requirements

kubectl command-line tool installed

-Kubernetes cluster to host Argo CD

-Cluster admin level access

-kubeconfig configured to connect to your Kubernetes cluster

Access to GitHub (or other repo)

Deploying Argo CD

Types of Argo CD Installs

Non High Availability

-Manifest = install.yaml

-Recommended for dev

-Single pods & replicas for Argo CD components

High Availability

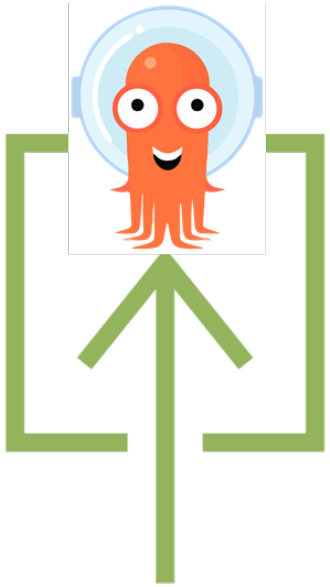
-Manifest = namespace-install.yaml

-Recommended for prod

-Tuned for high availability and resiliency

-Multiple replicas for supported components

Types of Argo CD Installs



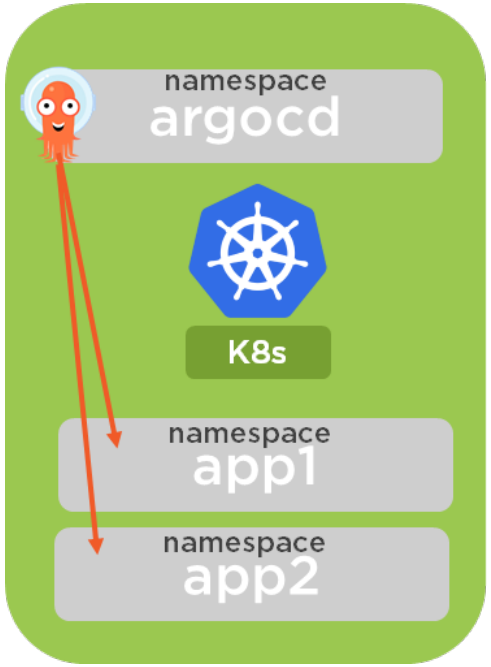
Core Install

-Manifest = core-install.yaml

-Use when multi-tenancy features (web ui, API server etc...) are not needed

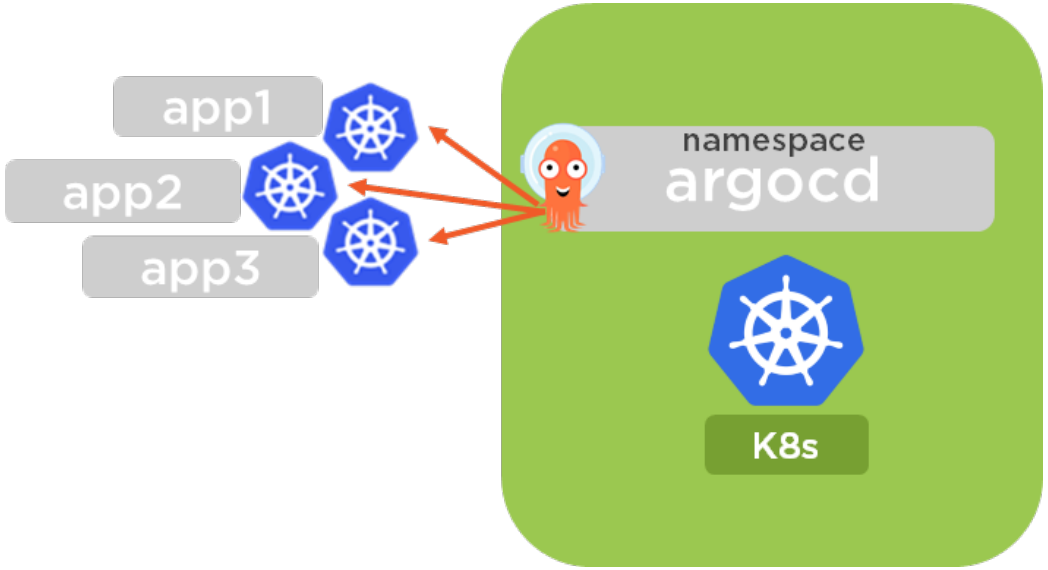
-installs the lightweight (non-HA) version of each component

Types of Argo CD Installs



Cluster Level

-Use when you have cluster level access & will deploy apps in the same K8s cluster that Argo CD runs on



Namespace Level

-Use when you have namespace level access & will deploy apps to external K8s clusters from where Argo CD is running

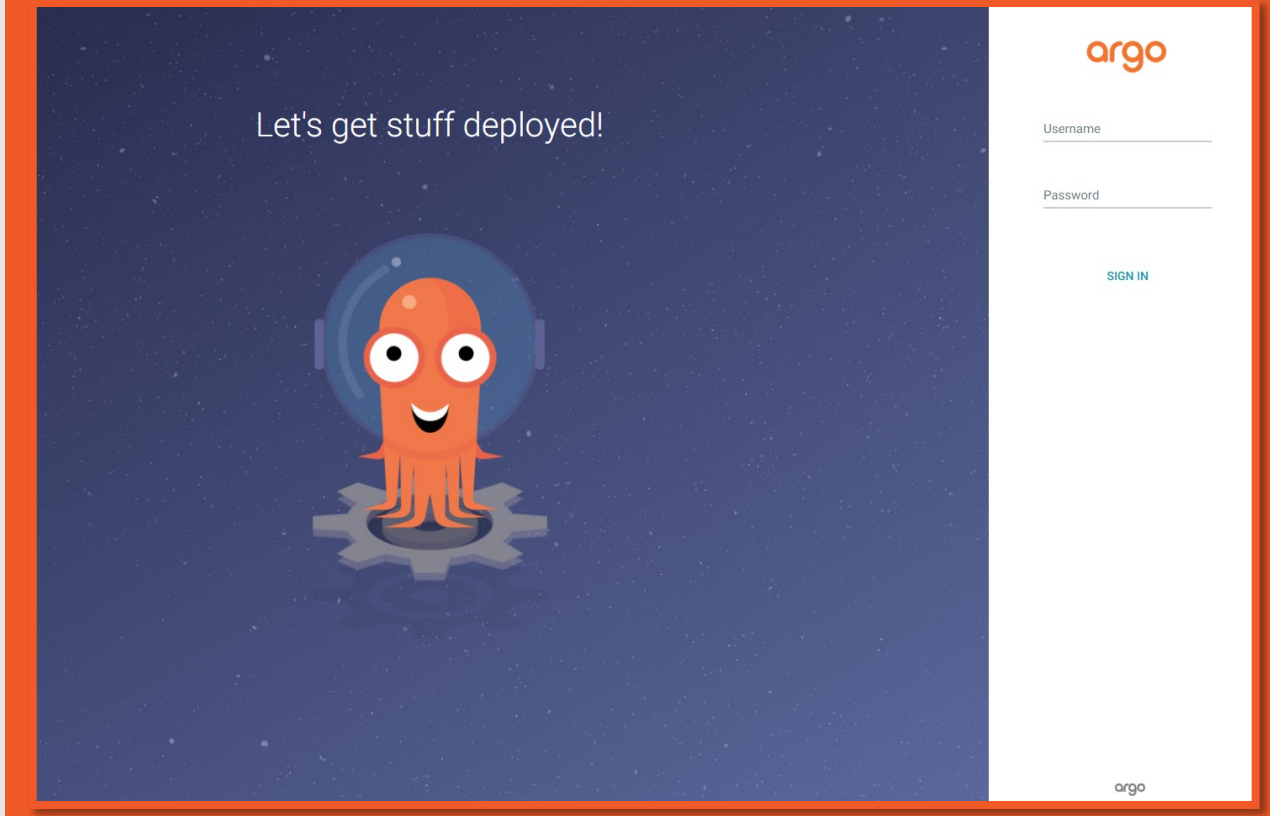
Install Argo CD

Create a namespace for Argo CD to deploy all of its components in:

```
kubectl create namespace argocd
```

Install Argo CD into the new namespace you created. Reference Argo CD's GitHub repository for the latest Argo CD operator:

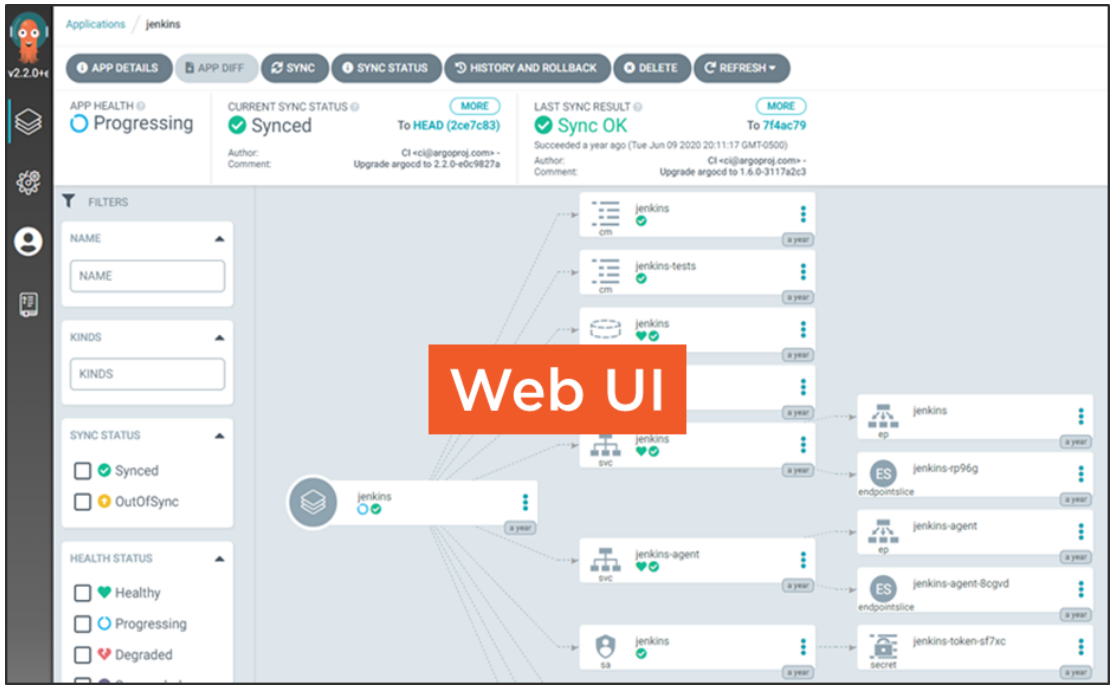
```
kubectl apply -n argocd -
f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml
```

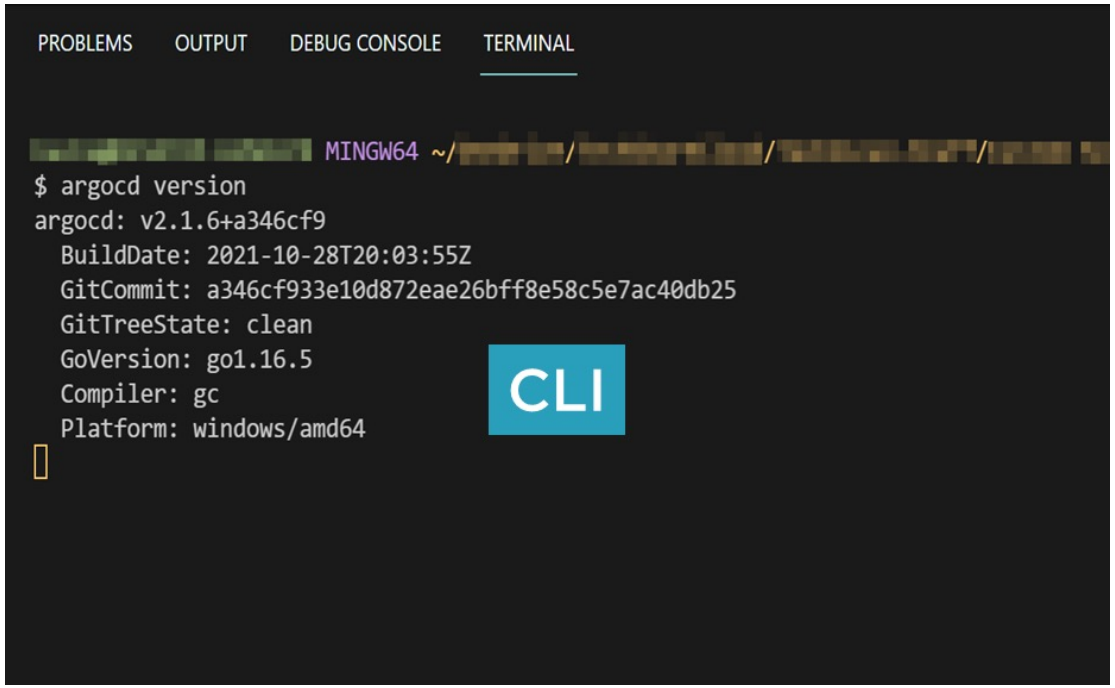


Accessing the Argo CD API Server

Accessing Argo CD

Two ways to access the Argo CD API Server





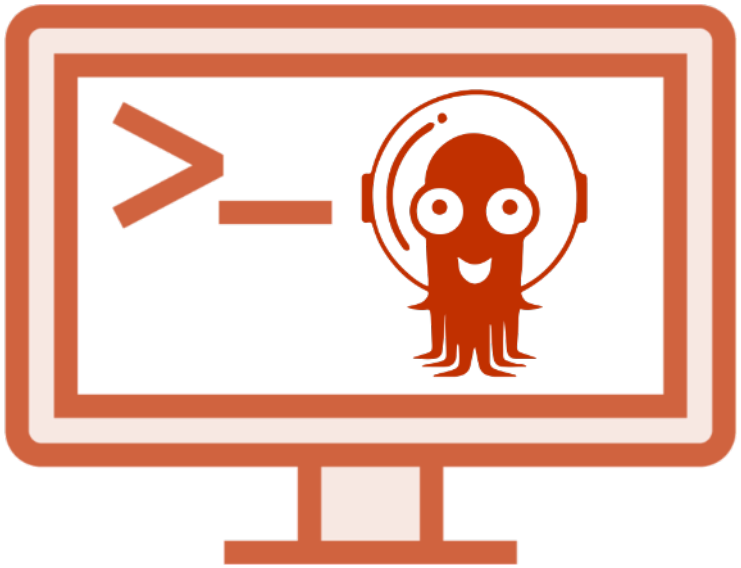
Using the ARGO CD Command Line Interface (CLI)

The Argo CD CLI

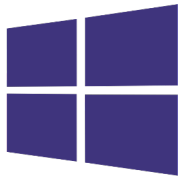
Argo CD has a Command Line Interface to Access the API Server

Some Activities can only be performed via the Argo CD CLI such as adding new clusters & managing user accounts

Script/Automate using Argo CD CLI

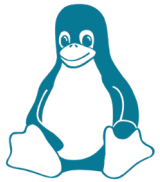


Ways to Install the Argo CD CLI



Windows: Chocolatey - choco install argocd-cli

Mac: Homebrew - brew install argocd



Linux: Homebrew - brew install argocd

Common Argo CD CLI Commands

argocd login

argocd account

argocd proj

argocd app

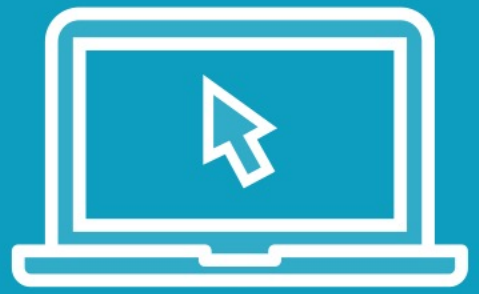
argocd repo

argocd version

argocd-util

argocd cluster

Demo



Demo: Tour of Argo CD Web User Interface

Upgrading Argo CD

Upgrading Argo CD

Argo CD follows Semantic Versioning

MAJOR.MINOR.PATCH

incrementing based on:

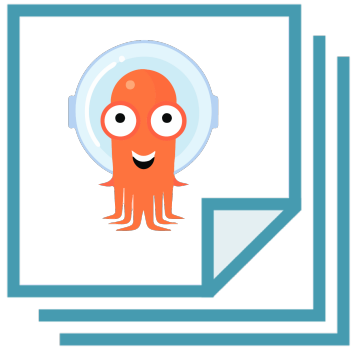
MAJOR version makes incompatible API changes	MINOR version adds functionality in a backwards compatible manner	PATCH version makes backwards compatible bug fixes
• Major introduces backward incompatible behavior changes with previous Argo CD versions	• Minor might introduce minor changes to Argo CD with a workaround	• patch such as Argo CD v1.6.1 to v1.6.3 does not introduce any breaking changes

Upgrading Argo CD

Running the following commands will upgrade your Argo CD

Non-HA:

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



HA:

```
kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/<version>/manifests/ha/install.yaml
```

Configuring User Management in Argo CD

Argo CD User Management

By default Argo CD has one built-in user (admin) when it is deployed

You have two options for users in Argo CD

Option 1: Local users

- This is good for small teams 5 or less & an API accounts for automation

Option 2: SSO Integration

- This is good for larger teams & integrating with external identity providers

Argo CD – Local Users

Argo CD local users are stored in a ConfigMap that is applied to Argo CD

Local Argo CD users do not have advanced features like groups, login history etc...

Each new user will need to be assigned a built in role

- readonly - read-only access to all resources
- admin - unrestricted access to all resources

Each new user will also need policy rules defined. Or they will default to policy.default These are:

- p, subject, resource, action, object, effect

```
apiVersion: v1

kind: ConfigMap

metadata:

  name: argocd-cm

  namespace: argocd

  labels:

    app.kubernetes.io/name: argocd-cm

    app.kubernetes.io/part-of: argocd

data:

  # add an additional local user with apiKey and login
capabilities

  #   apiKey - allows generating API keys

  #   login - allows to login using UI

  accounts.alice: apiKey, login

  # disables user. User is enabled by default

  accounts.alice.enabled: "false"
```

Argo CD – Local Users

- In Argo CD new users are created in a ConfigMap named argocd-cm

- Users can have the following capabilities:
  - apiKey - allows generating API keys
  - login - allows login using UI

- Managing users has to be done using the Argo CD CLI
  - Get users list: argocd account list
  - Get user details: argocd account get --account USERNAME
  - Set user password: argocd account update-password

Argo CD – SSO Integration

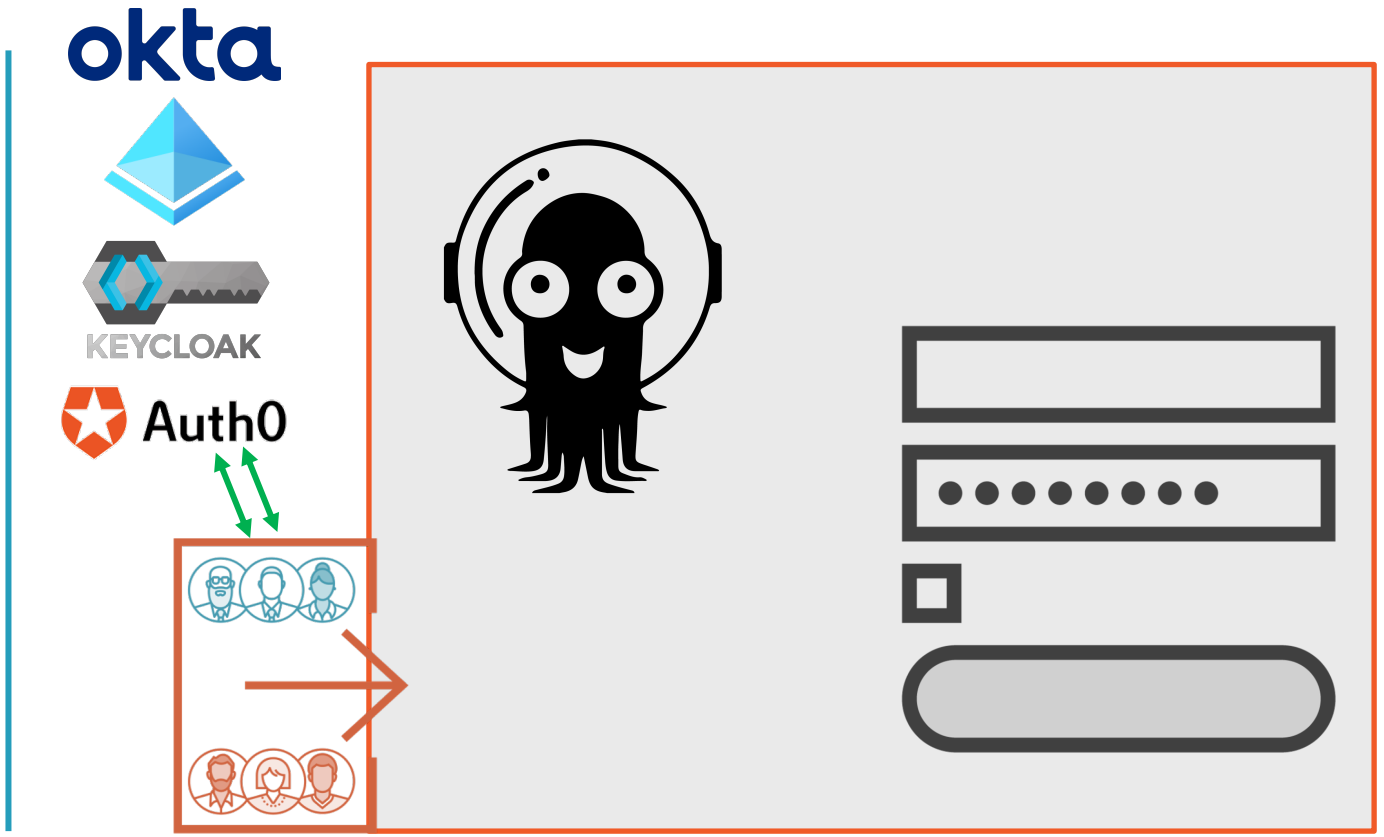
Argo CD is configured in the argocd-cm ConfigMap

Argo CD handles SSO via one of two options

1. Dex OIDC provider when your identity provider does not support OIDC i.e. SAML or LDAP

2. Existing OIDC provider. Supported OIDC providers are:

- Auth0
- Microsoft
- Okta
- OneLogin
- Keycloak
- OpenUnison
- Google



Setting up RBAC for Argo CD

Enabling Argo CD RBAC

Argo CD's RBAC feature enables restriction of access to Argo CD resources

Since Argo CD does not have its own user management system it requires SSO configuration or local users setup

After either SSO or local users is setup in Argo CD more RBAC roles can be defined, and local users or SSO users/groups can be mapped to these roles

Argo CD RBAC - Built-in Roles

role:readonly

read-only access to all resources

role:admin

unrestricted access to all resources

Argo CD RBAC - Resources & Actions

Resources:

clusters, projects, applications, repositories, certificates, accounts, gpgkeys

Actions:

get, create, update, delete, sync, override, action

Argo CD RBAC - Permissions

All resources except applications:

p, <role/user/group>, <resource>, <action>, <object>

Applications (tied to an AppProject):

p, <role/user/group>, <resource>, <action>, <appproject>/<object>

```
apiVersion: v1

kind: ConfigMap

metadata:

  name: argocd-rbac-cm

  namespace: argocd

data:

  policy.default: role:readonly

  policy.csv: |

    p, role:org-admin, applications, *, */*,
allow

    p, role:org-admin, clusters, get, *, allow

    p, role:org-admin, repositories, get, *,
allow

    p, role:org-admin, repositories, create, *,
allow

    p, role:org-admin, repositories, update, *,
allow
```

ArgoCD ConfigMap argocd-rbac-cm Example:

- Configures a custom role
- The role is assigned to users that belong to a my-github:teama group in Github
- Other users will get the default policy of role:readonly

Setting up Secrets management with Argo CD

Sensitive Information in Argo CD

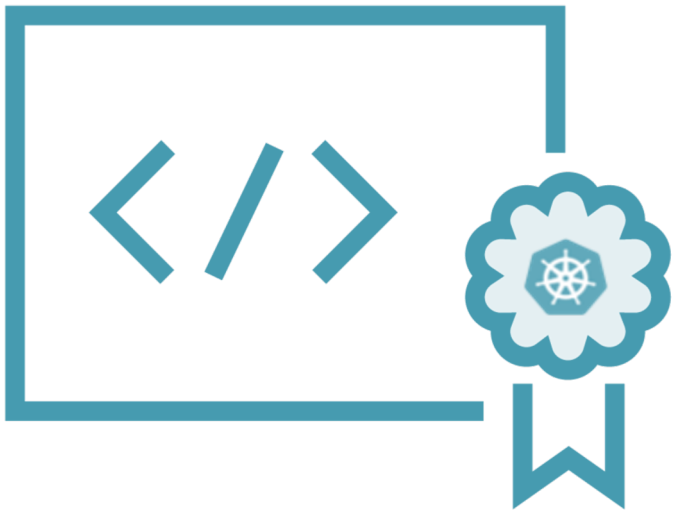
Argo CD never returns sensitive data from its API, and redacts all sensitive data in API payloads and logs. This includes:

- cluster credentials
- Git credentials
- OAuth2 client secrets
- Kubernetes Secret values

Securing External Cluster Credentials in Argo CD

Argo CD stores the credentials of the external cluster as a Kubernetes Secret in the argocd namespace

This secret contains the K8s API bearer token associated with the argocd-manager Service Account created during argocd cluster add



Secrets in Argo CD

Secret Management functionality is not built directly into Argo CD

We utilize a third-party solution to serve this need

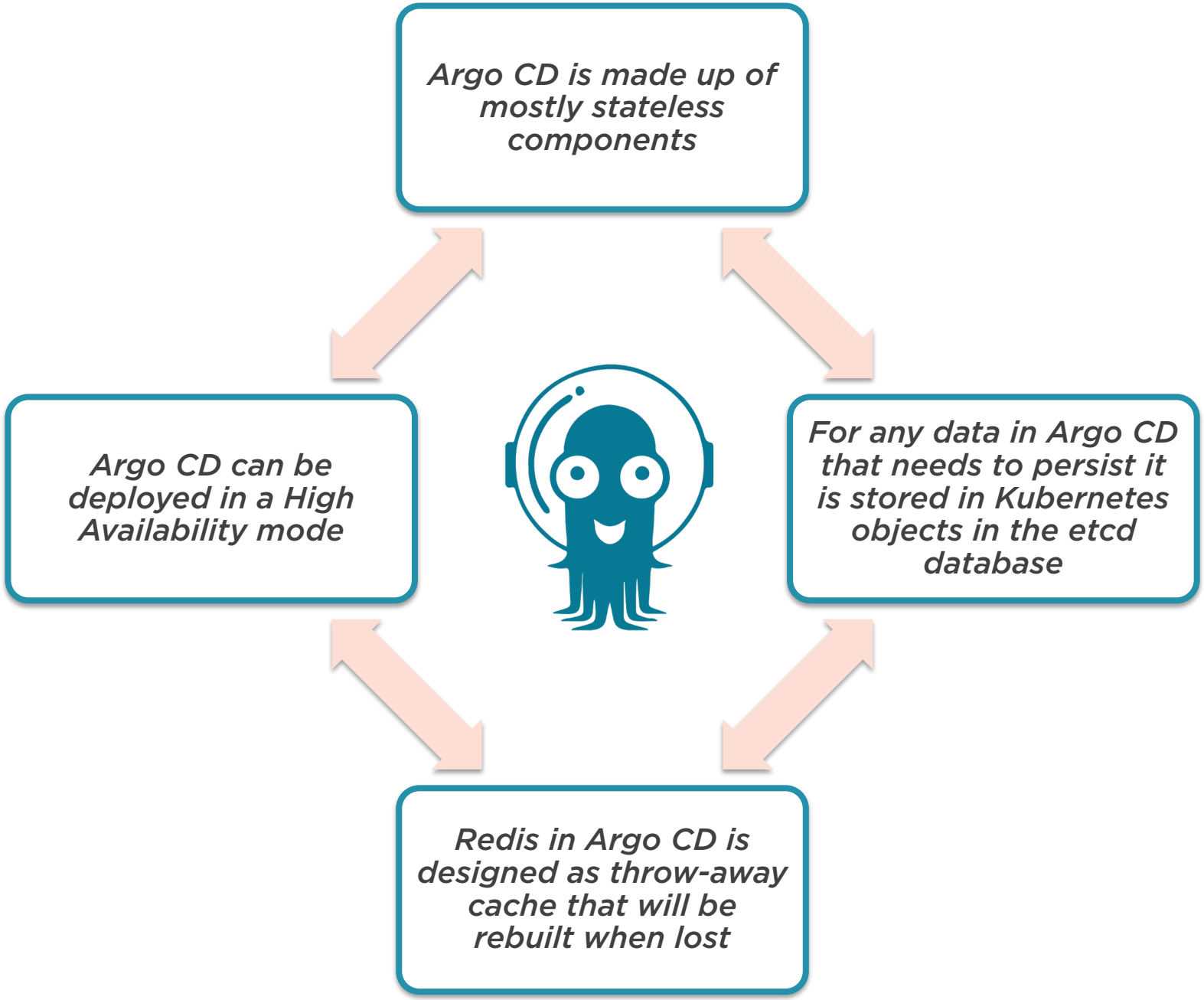
Secrets in Argo CD

GitOps Secret Management Solutions:

- Bitnami Sealed Secrets
- GoDaddy Kubernetes External Secrets
- External Secrets Operator
- Hashicorp Vault
- Banzai Cloud Bank-Vaults
- Helm Secrets
- Kustomize secret generator plugins
- aws-secret-operator
- KSOPS
- argocd-vault-plugin

High Availability, Backup, and Disaster Recovery with Argo CD

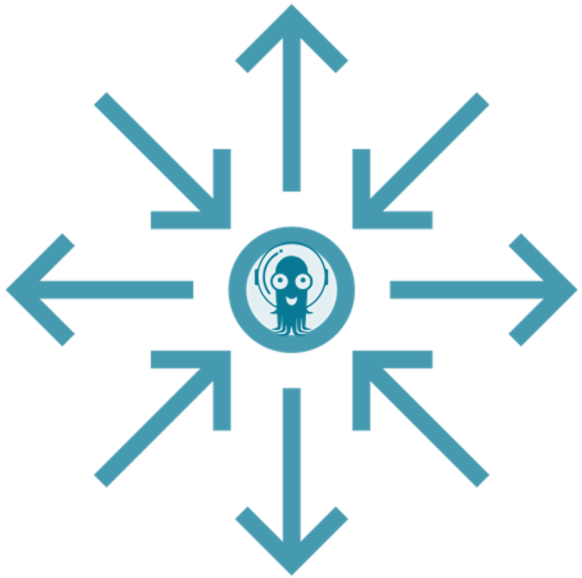
Argo CD High Availability



Argo CD High Availability

Argo CD's High Availability deployment requires at least three different nodes for pod anti-affinity roles

To deploy Argo CD in High Availability mode you will need to use the HA manifests



The HA manifests run more replicas (containers), & runs Redis in HA mode

Argo CD High Availability

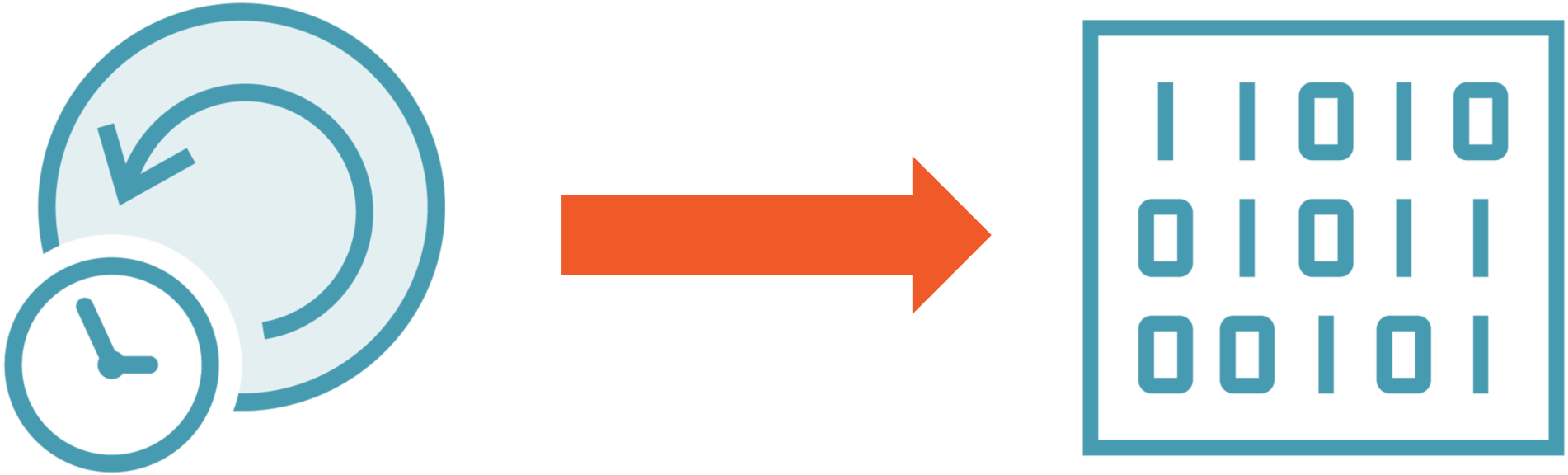
The HA manifests are:

ha/install.yaml: Deploys multiple replicas for supported Argo CD components. Use when you have cluster level access & will deploy apps in the same cluster that Argo CD runs on

ha/namespace-install.yaml: Deploys multiple replicas for supported Argo CD components. Use when you have namespace level access & will deploy apps to external clusters from where Argo CD is running

Argo CD Backup, and Disaster Recovery

argocd admin can export and import Argo CD data



Argo CD Backup, and Disaster Recovery

Backup

Export to a backup

```
docker run -v
~/.kube:/home/argocd/.kube --rm
argoproj/argocd:$VERSION argocd
admin export > backup.yaml
```

Restore

Import from a backup

```
docker run -i -v
~/.kube:/home/argocd/.kube --rm
argoproj/argocd:$VERSION argocd
admin import - < backup.yaml
```

Configuring Webhooks with Argo CD

Argo CD Webhooks

Argo CD supports Git webhook notifications from GitHub, GitLab, Bitbucket, & Gogs

The Argo CD API server can be configured to receive webhook events instead of relying on polling a Git repository

Monitoring and Notifications with Argo CD

Argo CD Monitoring

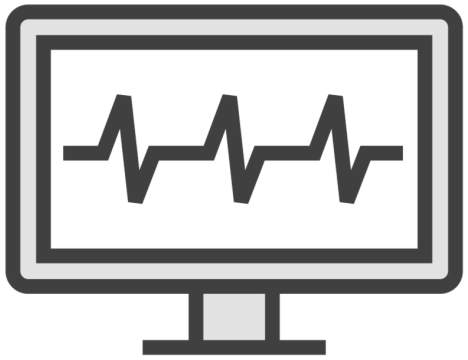
Argo CD exposes two sets of Prometheus metrics including: API Server Metrics & Application Metrics

Argo CD has a built-in health assessment that is surfaced up to the overall Application health status. This status comes from health checks on several standard K8s types:

- Deployment, ReplicaSet, StatefulSet DaemonSet
- Service
- Ingress
- PersistentVolumeClaim

Argo CD API Server Monitoring

API Server Metrics



request totals

response codes

etc....

Metrics about API Server/API request & response activity scraped at the argocd-server-metrics:8083/metrics endpoint

API request and response activities include:

Argo CD App Monitoring

Application Metrics

Gauge for application health status

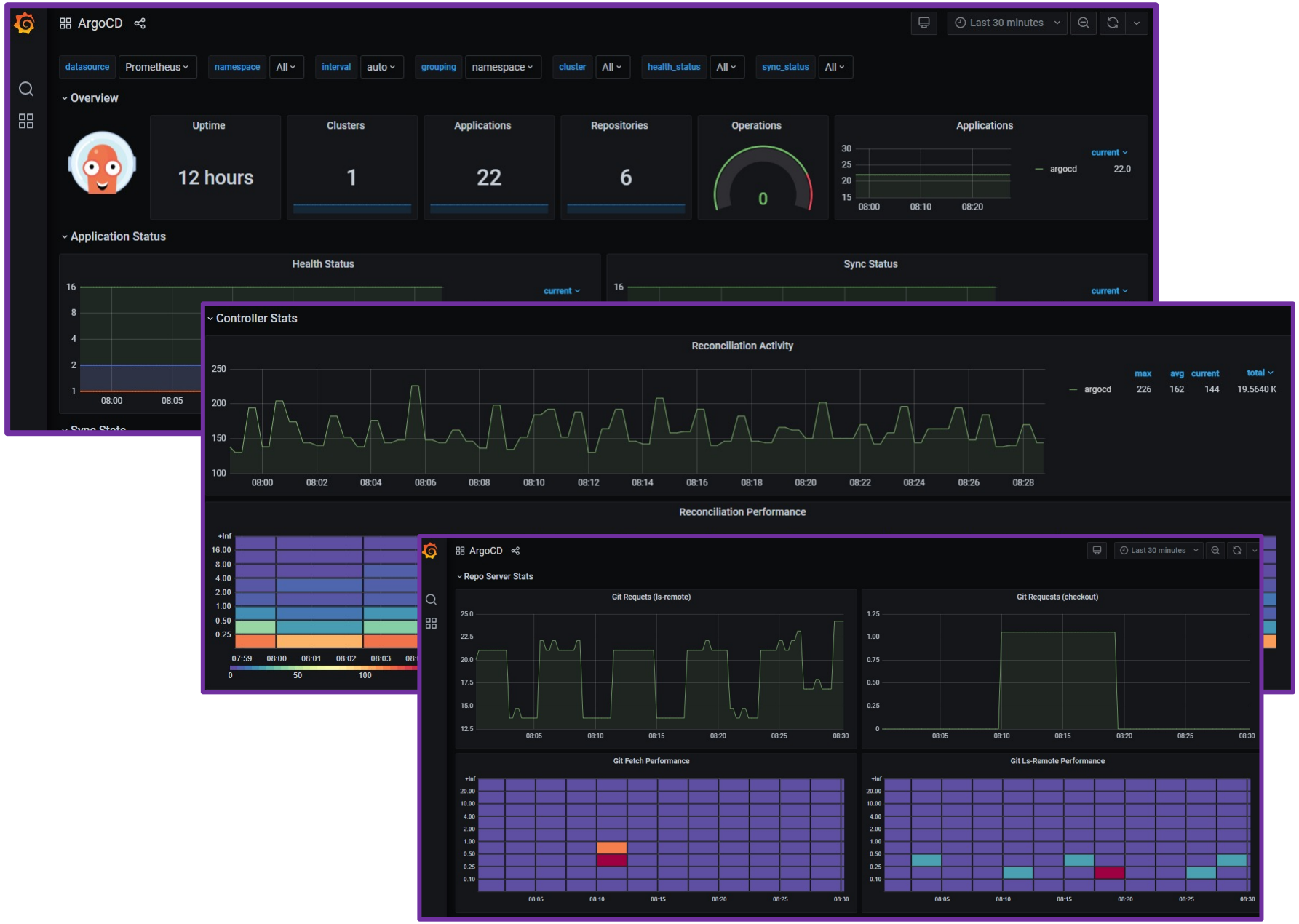
Gauge for application sync status

Counter for application sync history

Metrics about applications are scraped at the argocd-metrics:8082/metrics endpoint

Metrics include:

GRAFANA DASHBOARDS FOR ARGO CD



Argo CD Notifications

Notification functionality is not built directly into Argo CD

We utilize a third-party solution to serve this need

Argo CD Notifications

ArgoCD Notifications - Argo CD specific notification system that continuously monitors Argo CD applications & integrates with various notification services such as Slack, SMTP, Telegram, Discord, etc...

Argo Kube Notifier - generic Kubernetes resource controller that allows monitoring any Kubernetes resource and sends a notification when the configured rule is met

Kube Watch - a Kubernetes watcher that publishes notifications to Slack, hipchat, mattermost, & flock channels. It watches the cluster for resource changes and notifies the channels through webhooks

Summary

In this module we covered:

- We looked at the aspects of deploying & upgrading Argo CD, Accessing Argo CD & more
- We also looked at the Configuration aspects of Argo CD such as user management, RBAC, secrets, webhooks, monitoring & notifications

Why this is important:?

- This module sets the foundational knowledge needed to get your Argo CD up & running as well as configured to prepare the environment for applications