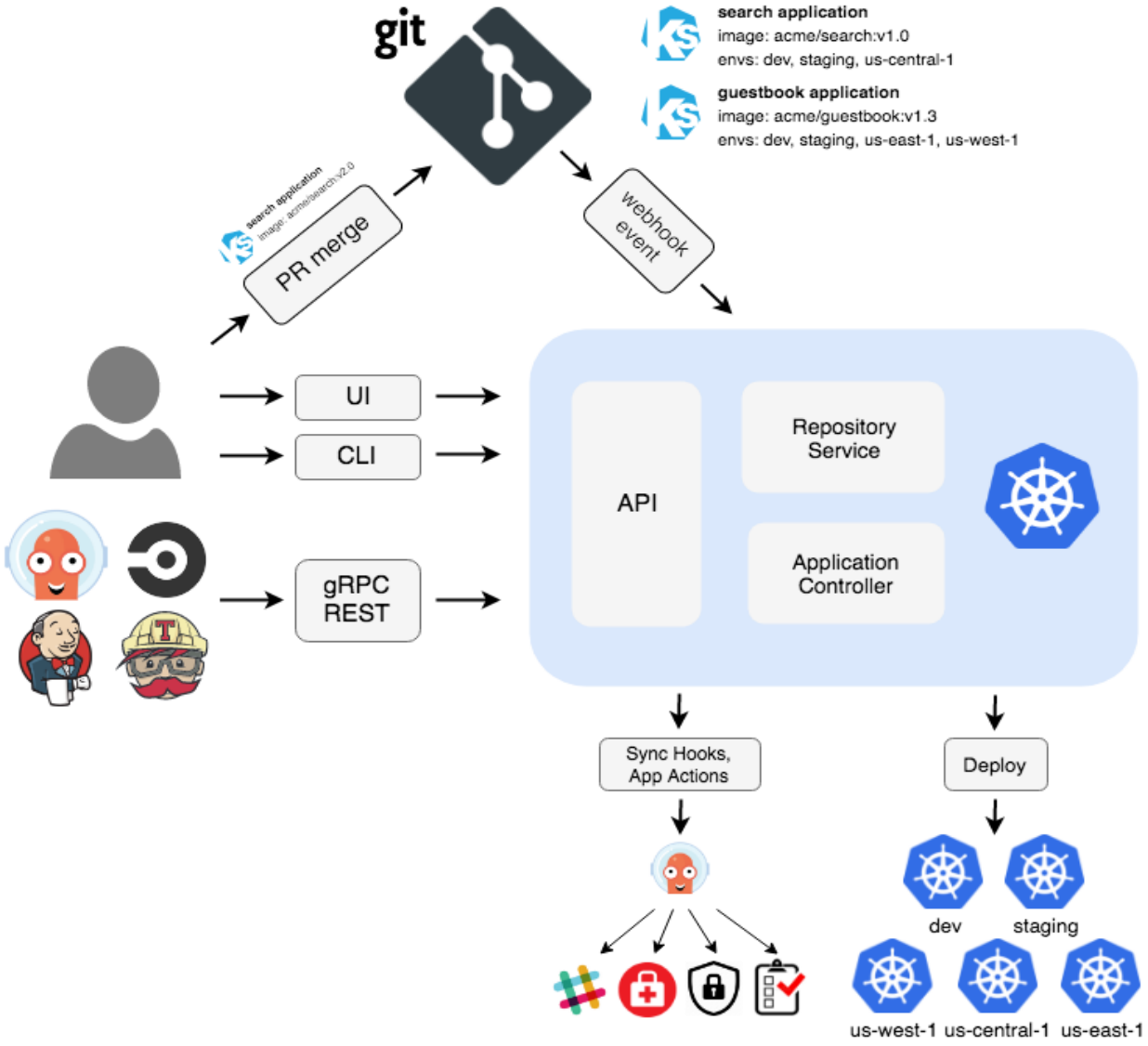
ARGO-CD


git
search application
image: acme/search:v1.0
envs: dev, staging, us-central-1
guestbook application
image: acme/guestbook:v1.3
envs: dev, staging, us-east-1, us-west-1
PR merge
webhook event
UI
CLI
gRPC
REST
API
Repository Service
Application Controller
Sync Hooks, App Actions
Deploy
dev
staging
us-west-1
us-central-1
us-east-1

Argo CD is an open-source, declarative, GitOps continuous delivery tool for Kubernetes. It is designed to help developers and DevOps teams automate the deployment and management of applications on Kubernetes clusters.

Argo CD allows you to define the desired state of your applications and their resources in a Git repository, and it continuously monitors the cluster to ensure that the actual state matches the desired state.

ArgoCD follows the principles of GitOps, where the desired state of the application is defined in a Git repository and a continuous delivery system automatically synchronizes the actual state with the desired state.

ArgoCD is a powerful tool in the world of Kubernetes, to simplify the deployment and management of applications in a GitOps workflow. GitOps is a paradigm that leverages version control systems like Git to manage and automate the deployment of applications on Kubernetes clusters. It's becoming popular for its ability to ensure consistency, traceability, and collaboration in the world of container orchestration.

Features

1. GitOps Workflow: Argo CD follows the principles of GitOps, where the desired state of the application is defined in a Git repository. It continuously monitors the repository for changes and ensures that the actual state of the applications matches the desired state.

2. Declarative Application Definition/Configuration: ArgoCD follows the declarative approach, where you define your desired application state in a Git repository. This repository contains Kubernetes manifests, Helm charts, or Kustomize overlays that specify how your application should run on your clusters.

3. Automated Deployment/Synchronization: Argo CD automates the application deployment process by continuously synchronizing the desired state defined in Git with the actual state in the Kubernetes cluster. It detects changes and updates the applications accordingly.

ArgoCD continuously monitors your Git repository and the actual state of your Kubernetes clusters. When it detects changes in the repository, it automatically syncs the cluster to match the desired state, ensuring your applications are always up-to-date.

4. **Rollback and Rollout:** Argo CD provides rollback and rollout capabilities. It allows users to roll back to a previous known good state if issues occur during deployment. It also supports progressive deployments, such as canary and blue-green deployments.

5. Application Configuration Management: Argo CD manages application configuration using ConfigMaps, Secrets, and other Kubernetes resources. It tracks changes to the configuration and ensures that the deployed applications always use the correct configuration.

6. Multi-Environment Support/Multi-Cluster Management: Whether you're working with a single cluster or a multi-cluster environment, ArgoCD can handle it. It simplifies the complex task of managing applications across various clusters, making it easier to maintain consistency and compliance.

7. RBAC and Security: ArgoCD integrates with Kubernetes RBAC (Role-Based Access Control), providing fine-grained control over who can make changes to your applications. It helps maintain security by allowing you to define access rights and permissions.

8. Customization and Extensibility: It is highly customizable. You can extend its functionality by creating custom plugins and hooks to fit your specific use cases, making it adaptable to your unique requirements.

9. User-Friendly Web UI: It offers a user-friendly web-based dashboard, allowing both developers and operators to visualize the application deployment status and history, making it easier to troubleshoot and monitor deployments.

10. **Integration with Git Providers**: It supports various Git hosting services, including GitHub, GitLab, and Bitbucket, making it easy to integrate into your existing Git workflows.

11. Continuous Delivery Pipelines:It can be part of a broader continuous delivery pipeline, working seamlessly with tools like Jenkins, Tekton, or any CI/CD system of your choice.

Below are some of the concepts that are specific to Argo CD.

- Application A group of Kubernetes resources as defined by a manifest. This is a Custom Resource Definition (CRD).

- Application source type Which Tool is used to build the application.
- 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 etc are deployed.
- 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?
- Tool A tool to create manifests from a directory of files. E.g. Kustomize. See Application Source Type.
- Configuration management tool See Tool.
- Configuration management plugin A custom tool.

INSTALLATION:

Update the system packages on Ubuntu 22.04 LTS

---

sudo apt update -y

install below packages for minikube

---

sudo apt install curl wget apt-transport-https -y

Install Docker on Ubuntu 22.04 LTS

---

sudo apt install docker.io

Configure to Run docker without sudo permission

---

sudo usermod -aG docker $USER

sudo chmod 666 /var/run/docker.sock

To check whether virtualization support is enabled on your machine or not

---

egrep -q 'vmx|svm' /proc/cpuinfo && echo yes || echo no

install the KVM and and other tools

---

sudo apt install qemu-kvm libvirt-clients libvirt-daemon-system bridge-utils virtinst libvirt-daemon

Add your user to libvert group

---

sudo adduser -a $USER libvirt

sudo adduser -a $USER libvirt-qemu

Reload Group:

---

newgrp libvirt

newgrp libvirt-qemu

To download latest minikube setup

---

curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64

Install Minikube

---

sudo install minikube-linux-amd64 /usr/local/bin/minikube

To check minikube version

==========

minikube version

Download kubectl binary with curl on Ubuntu using below command

==========

curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"

Make the kubectl binary executable

---

chmod +x ./kubectl

Make the kubectl binary executable

==========

sudo mv kubectl /usr/local/bin/

To check kubectl version on Ubuntu

---

kubectl version --client --output=yaml

Start the minikube Kubernetes cluster on Ubuntu

==========

minikube start --vm-driver docker

minikube status

here are several driver options that you can use to start a minikube cluster (virtualbox, docker, hyperv).We are using driver as docker

==========

minikube start --driver=docker

create namespace

---

kubectl create ns argocd

ArgoCD can be installed using its manifests

==========

kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/v2.5.8/manifests/install.yaml

Let's verify the installation by getting all the objects in the ArgoCD namespace.

---

kubectl get all -n argocd

By default, the ArgoCD server is not exposed outside the cluster. You can expose it using port-forwarding to access the ArgoCD UI.

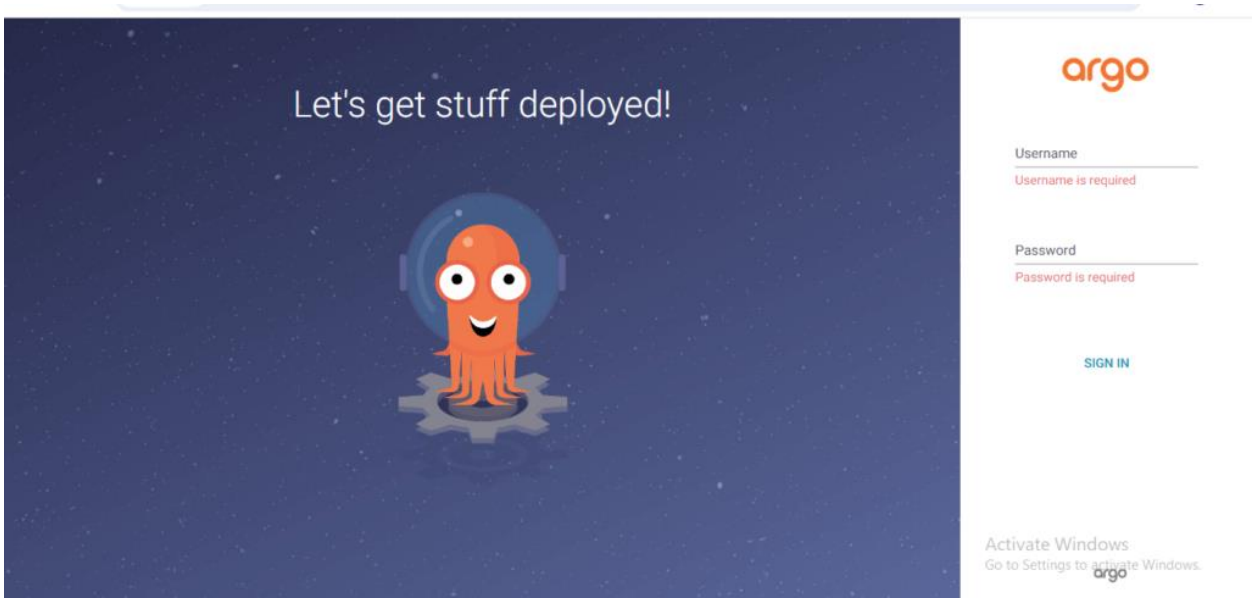
==========

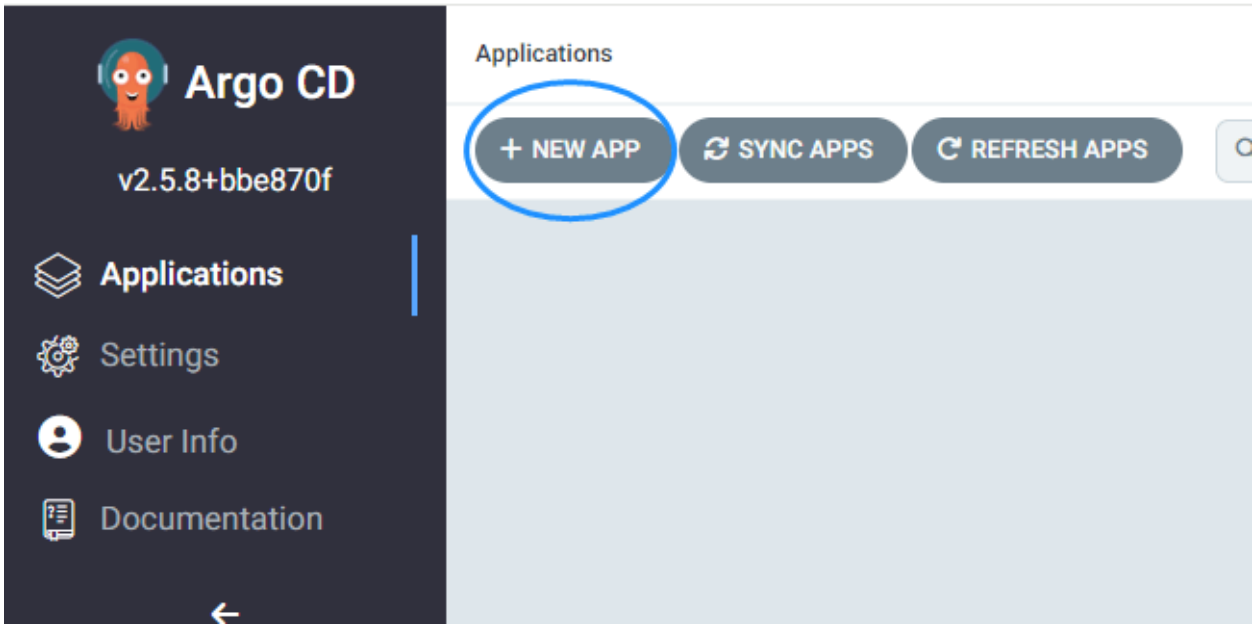
kubectl port-forward svc/argocd-server -n argocd --address 0.0.0.0 8080:443

The ArgoCD UI will be available at http://localhost/IP:8080. Access it through your web browser.Get the initial password for the admin user to log in

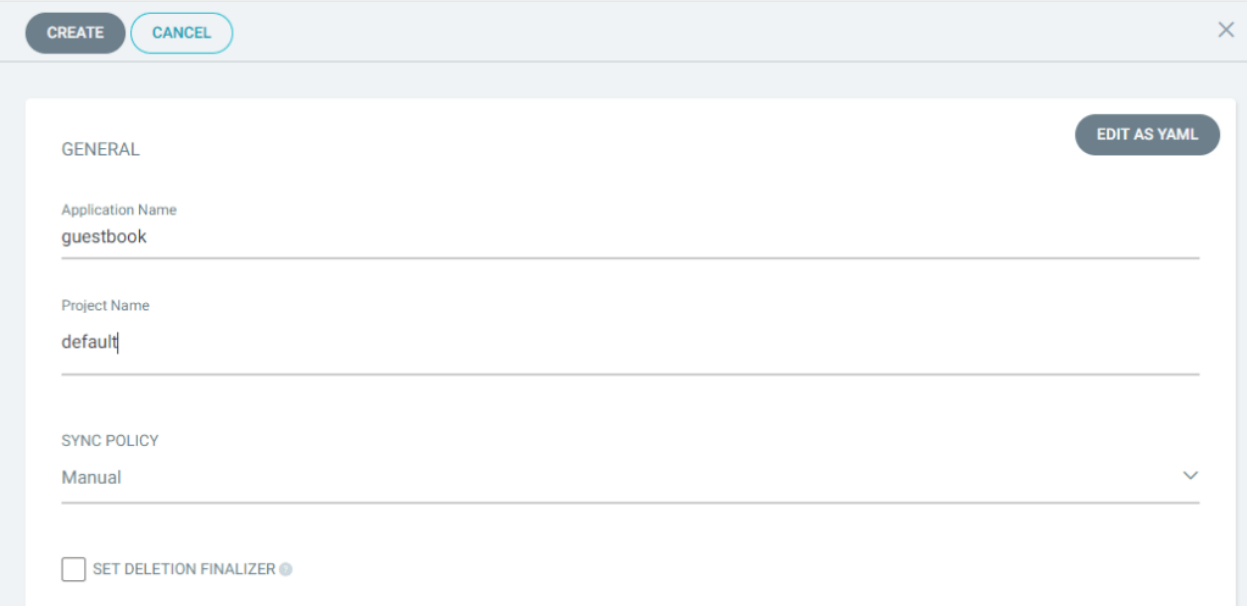
==========

kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath="{.data.password}" | base64 –d

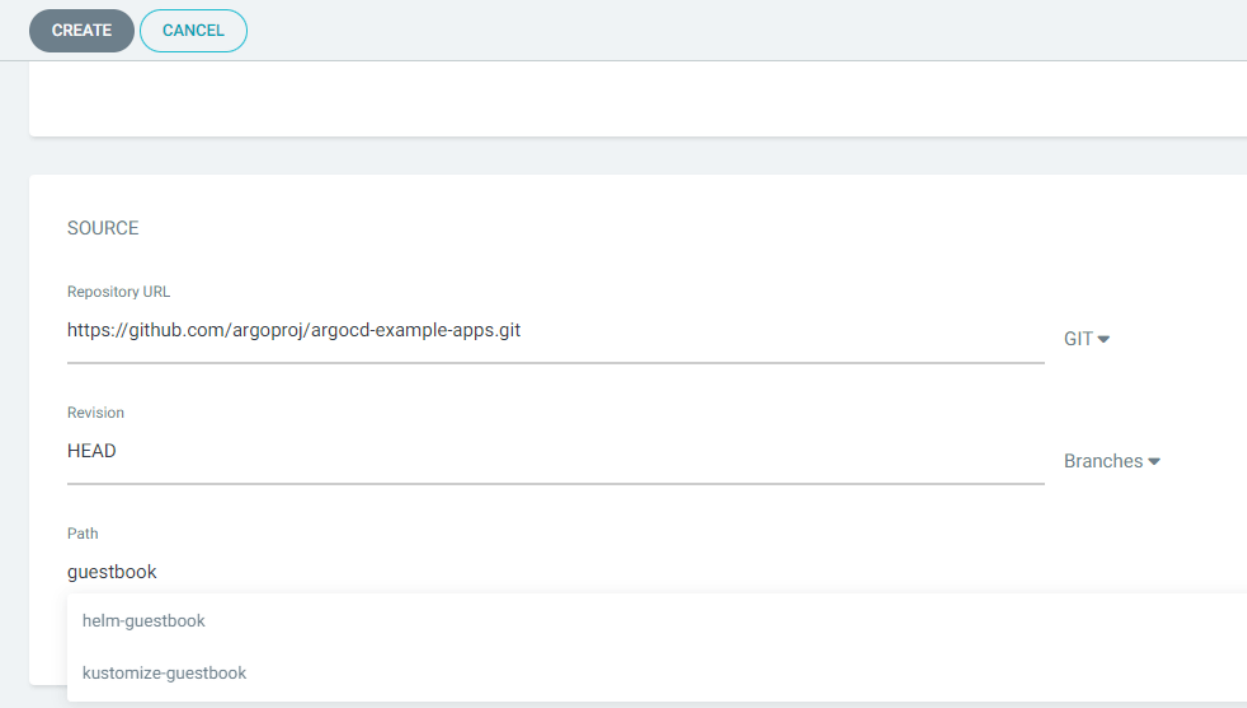




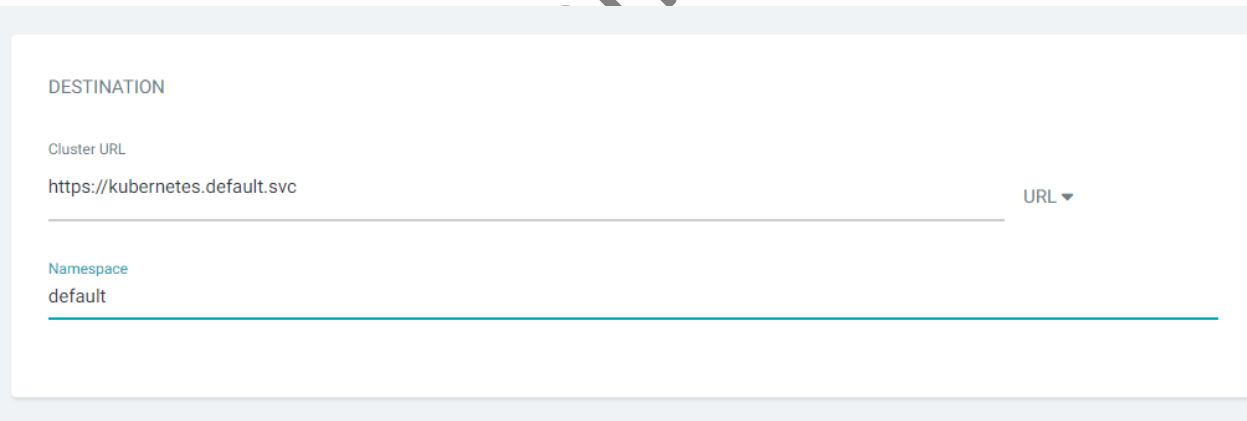
Give your app the name **`guestbook`**, use the project **`default`**, and leave the sync policy as **`Manual`**

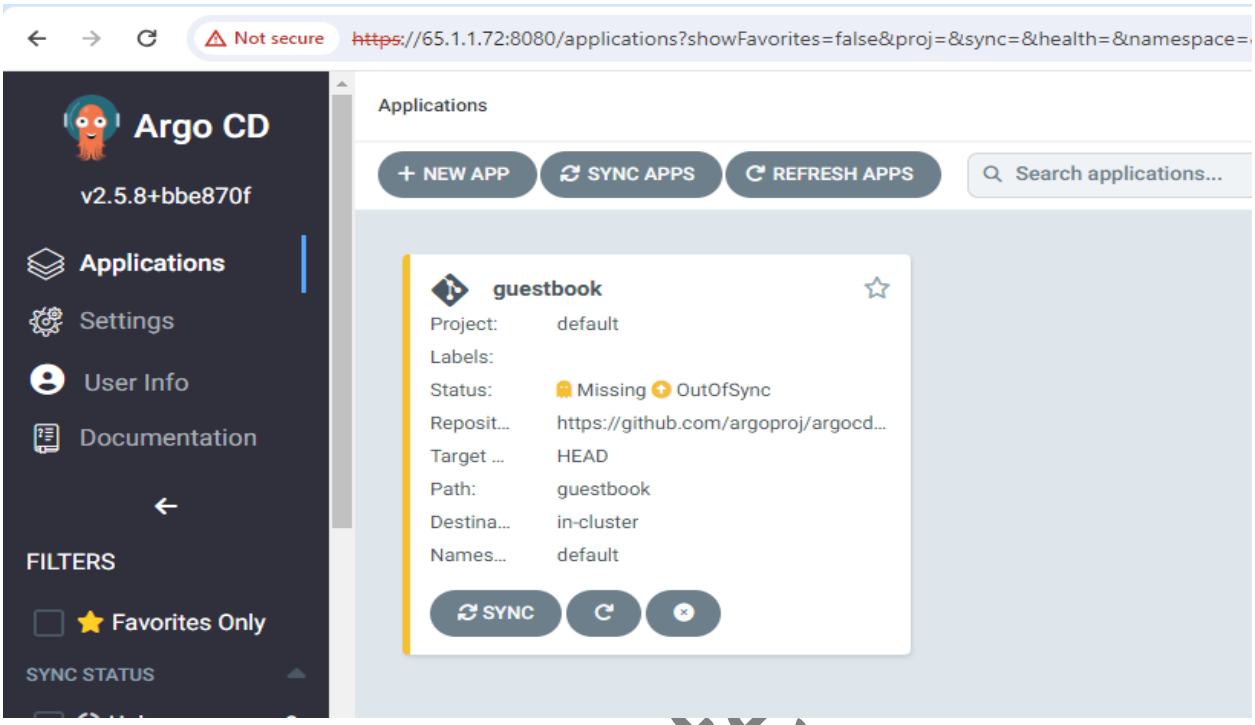


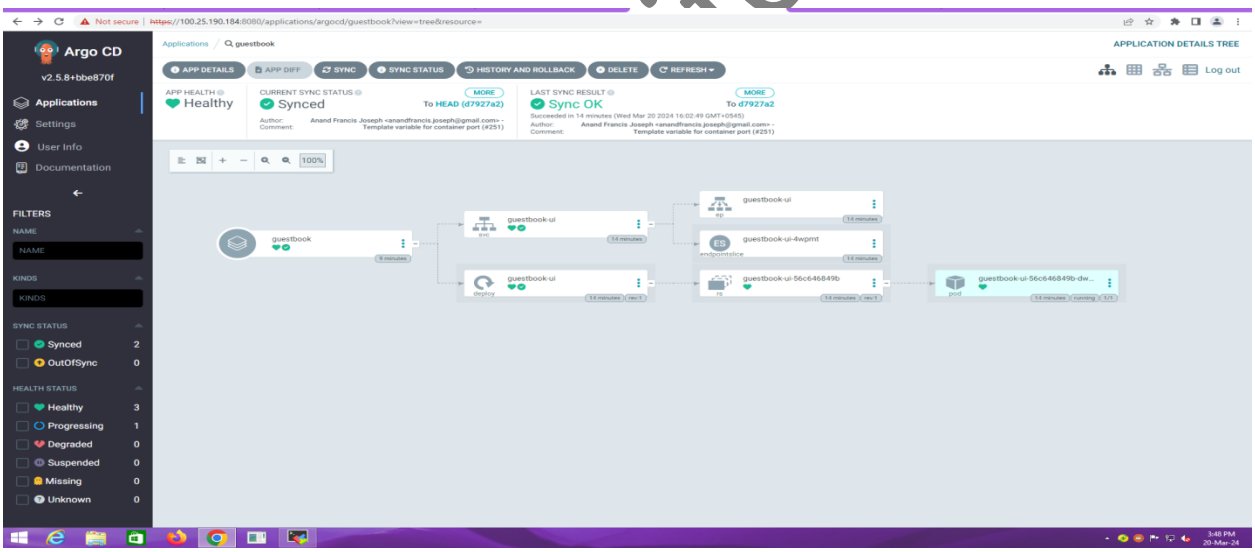
Connect the https://github.com/argoproj/argocd-example-apps.git repo to Argo CD by setting repository url to the github repo url, leave revision as **`HEAD`**, and set the path to **`guestbook`**:



For **Destination**, set cluster URL to `https://kubernetes.default.svc` (or `in-cluster` for cluster name) and namespace to **`default`**:




Not secure
https://65.1.1.72:8080/applications?showFavorites=false&proj=&sync=&health=&namespace=
Argo CD
v2.5.8+bbe870f
Applications
Settings
User Info
Documentation
FILTERS
Favorites Only
SYNC STATUS
Applications
+ NEW APP
SYNC APPS
REFRESH APPS
Search applications...
guestbook
Project: default
Labels:
Status: Missing OutOfSync
Reposit... https://github.com/argoproj/argocd...
Target ... HEAD
Path: guestbook
Destina... in-cluster
Names... default
SYNC


Argo CD
v2.5.8+bbe870f
Applications
Settings
User Info
Documentation
APP DETAILS
APP DIFF
SYNC
SYNC STATUS
HISTORY AND ROLLBACK
DELETE
REFRESH
APPLICATION DETAILS TREE
Log out
Healthy
Synced
Sync OK
guestbook
guestbook-ui
guestbook-ui-4wpmt
FILTERS
SYNC STATUS
Synced
OutOfSync
HEALTH STATUS
Healthy
Progressing
Degraded
Suspended
Missing
Unknown

PYTHON LIFE