How to Install ArgoCD on Minikube and Deploy an App

In this article on we are going to cover Install Minikube on Ubuntu 22.04 LTS, How to Install ArgoCD on Minikube and Deploy an App on ArgoCD.

What is ArgoCD?

Argo CD is a declarative, GitOps continuous delivery (CD) tool for Kubernetes. It is part of the Argo Project, which is a collection of open-source tools for running and managing Kubernetes workloads and applications. Argo CD helps you manage and deploy your Kubernetes applications by syncing your desired application state stored in a Git repository with the actual state in your Kubernetes cluster.

What is Minikube in Kubernetes?

Minikube is a tool that allows you to run a single-node Kubernetes cluster on your local machine. It's designed for development and testing purposes, providing an easy way to set up a local Kubernetes environment without the need for a full-scale production cluster. Here are some key points about Minikube.

Ankita Lunawat

What is GitOps?

GitOps brings the best practices of DevOps, continuous delivery, and version control into infrastructure management, making it easier to deploy, manage, and monitor infrastructure and applications in a reliable and auditable way. By leveraging Git as the single source of truth and automating the deployment process, GitOps improves efficiency, consistency, and collaboration in managing cloud-native applications and infrastructure.

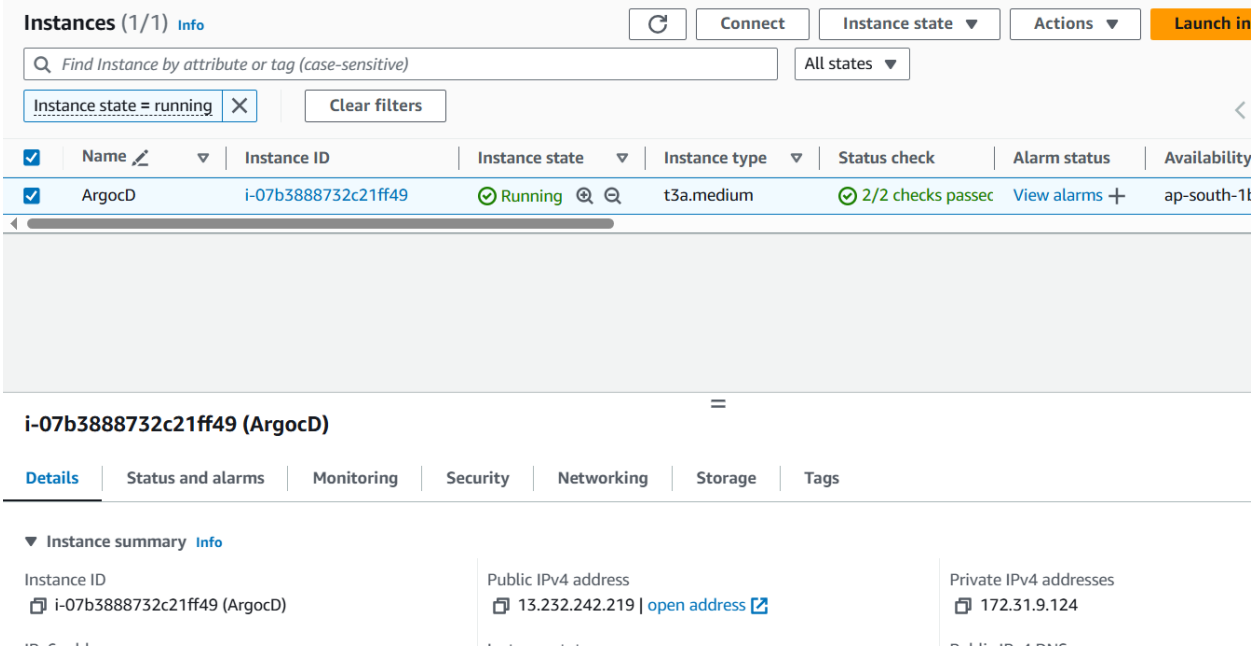
Prerequisites:

Before diving into the setup process, make sure you have the following prerequisites in place:
AWS EC2 Instance: Create an EC2 instance running Ubuntu.

- Minimum 2 CPU's or more
- Minimum 2GB of free memory
- Minimum 20GB of free disk space
- T3.medium instance type

SSH Access: Ensure you have SSH access to your EC2 instance.
Basic Linux Knowledge: Familiarity with basic Linux commands will be beneficial.
Launch an instance: Connect to Public IP to server.

In security Group – Edit Inbound Rule with TCP -8080, https - 443



Step 1:Install Minikube on Ubuntu 22.04 LTS

```
ubuntu@ip-172-31-9-124:~$ sudo apt update -y
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Hit:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Fetched 126 kB in 1s (177 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
```

Install below packages for minikube

```
ubuntu@ip-172-31-9-124:~$ sudo apt install curl wget apt-transport-https -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
curl is already the newest version (8.5.0-2ubuntu10.1).
wget is already the newest version (1.21.4-1ubuntu4.1).
apt-transport-https is already the newest version (2.7.14build2).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

Install Docker on Ubuntu 22.04 LTS

```
ubuntu@ip-172-31-9-124:~$ sudo apt install docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base pigz runc ubuntu-fan
Suggested packages:
  ifupdown aufs-tools cgroupfs-mount | cgroup-lite debootstrap docker-buildx docker-compose-v2 docker-doc rinse zfs-fuse | zfsutils
The following NEW packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base docker.io pigz runc ubuntu-fan
0 upgraded, 8 newly installed, 0 to remove and 3 not upgraded.
Need to get 76.8 MB of archives.
After this operation, 289 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 pigz amd64 2.8-1 [65.6 kB]
Get:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 bridge-utils amd64 1.7.1-1ubuntu2 [33.9 kB]
Get:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 runc amd64 1.1.12-0ubuntu3 [8599 kB]
Get:4 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 containerd amd64 1.7.12-0ubuntu4 [38.6 MB]
Get:5 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 dns-root-data all 2023112702~willsync1 [4450 B]
Get:6 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 dnsmasq-base amd64 2.90-2build2 [375 kB]
Get:7 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 docker.io amd64 24.0.7-0ubuntu4 [29.1 MB]
Get:8 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 ubuntu-fan all 0.12.16 [35.2 kB]
Fetched 76.8 MB in 2s (44.0 MB/s)
Preconfiguring packages ...
Selecting previously unselected package pigz.
(Reading database ... 67743 files and directories currently installed.)
Preparing to unpack .../0-pigz_2.8-1_amd64.deb ...
Unpacking pigz (2.8-1) ...
Selecting previously unselected package bridge-utils.
Preparing to unpack .../1-bridge-utils_1.7.1-1ubuntu2_amd64.deb ...
Unpacking bridge-utils (1.7.1-1ubuntu2) ...
Selecting previously unselected package runc.
Preparing to unpack .../2-runc_1.1.12-0ubuntu3_amd64.deb ...
Unpacking runc (1.1.12-0ubuntu3) ...
Selecting previously unselected package containerd.
Preparing to unpack .../3-containerd_1.7.12-0ubuntu4_amd64.deb ...
Unpacking containerd (1.7.12-0ubuntu4) ...
Selecting previously unselected package dns-root-data.
Preparing to unpack .../4-dns-root-data_2023112702~willsync1_all.deb ...
Unpacking dns-root-data (2023112702~willsync1) ...
Selecting previously unselected package dnsmasq-base.
Preparing to unpack .../5-dnsmasq-base_2.90-2build2_amd64.deb ...
```

Configure to Run docker without sudo permission.

```
ubuntu@ip-172-31-9-124:~$ sudo usermod -aG docker $USER
ubuntu@ip-172-31-9-124:~$ sudo chmod 666 /var/run/docker.sock
ubuntu@ip-172-31-9-124:~$ egrep -q 'vmx|svm' /proc/cpuinfo && echo yes || echo no
no
ubuntu@ip-172-31-9-124:~$ sudo apt install qemu-kvm libvirt-clients libvirt-daemon-system bridge-utils virtinst libvirt-daemon
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Note, selecting 'qemu-system-x86' instead of 'qemu-kvm'
bridge-utils is already the newest version (1.7.1-1ubuntu2).
bridge-utils set to manually installed.
```

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

```
ubuntu@ip-172-31-9-124:~$ sudo adduser -a $USER libvirt
sudo adduser -a $USER libvirt-qemu
Option a is ambiguous (add-extra-groups, add_extra_groups, allow-all-names, allow-bad-names, allow-badname)
adduser [--uid id] [--firstuid id] [--lastuid id]
        [--gid id] [--firstgid id] [--lastgid id] [--ingroup group]
        [--add-extra-groups] [--encrypt-home] [--shell shell]
        [--comment comment] [--home dir] [--no-create-home]
        [--allow-all-names] [--allow-bad-names]
        [--disabled-password] [--disabled-login]
        [--conf file] [--extrausers] [--quiet] [--verbose] [--debug]
        user
    Add a normal user

adduser --system
        [--uid id] [--group] [--ingroup group] [--gid id]
        [--shell shell] [--comment comment] [--home dir] [--no-create-home]
        [--conf file] [--extrausers] [--quiet] [--verbose] [--debug]
        user
   Add a system user

adduser --group
        [--gid ID] [--firstgid id] [--lastgid id]
        [--conf file] [--extrausers] [--quiet] [--verbose] [--debug]
        group
addgroup
        [--gid ID] [--firstgid id] [--lastgid id]
        [--conf file] [--extrausers] [--quiet] [--verbose] [--debug]
        group
   Add a user group
```

Install the KVM and and other tools on Ubuntu 22.04 LTS

Add your user to libvert group

Reload Group

To download latest minikube setup refer minikube official download page.

```
ubuntu@ip-172-31-9-124:~$ newgrp libvirt
newgrp libvirt-qemu
ubuntu@ip-172-31-9-124:~$ curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 91.1M  100 91.1M    0     0  11.3M      0  0:00:08  0:00:08 --:--:-- 15.8M
ubuntu@ip-172-31-9-124:~$ sudo install minikube-linux-amd64 /usr/local/bin/minikube
ubuntu@ip-172-31-9-124:~$ minikube version
minikube version: v1.33.1
commit: 5883c09216182566a63dff4c326a6fc9ed2982ff
```

Install Minikube on Ubuntu 22.04 LTS

To check minikube version on Ubuntu

```
ubuntu@ip-172-31-9-124:~$ sudo install minikube-linux-amd64 /usr/local/bin/minikube
ubuntu@ip-172-31-9-124:~$ minikube version
minikube version: v1.33.1
commit: 5883c09216182566a63dff4c326a6fc9ed2982ff
ubuntu@ip-172-31-9-124:~$
```

Make the kubectl binary executable

```
^Cubuntu@ip-172-31-9-124:~$  cd /usr/local/bin/
ubuntu@ip-172-31-9-124:/usr/local/bin$ ls
kubectl  minikube
ubuntu@ip-172-31-9-124:/usr/local/bin$ kubectl version --client --output=yaml
clientVersion:
  buildDate: "2024-06-11T20:29:44Z"
  compiler: gc
  gitCommit: 39683505b630ff2121012f3c5b16215a1449d5ed
  gitTreeState: clean
  gitVersion: v1.30.2
  goVersion: go1.22.4
  major: "1"
  minor: "30"
  platform: linux/amd64
kustomizeVersion: v5.0.4-0.20230601165947-6ce0bf390ce3
```

Step 2: Install kubectl on Minikube

Download kubectl binary with curl on Ubuntu using below command.

```
ubuntu@ip-172-31-9-124:~$ curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   138  100   138    0     0    421      0 --:--:-- --:--:-- --:--:--   422
100 49.0M  100 49.0M    0     0  85.0M      0 --:--:-- --:--:-- --:--:-- 85.0M
ubuntu@ip-172-31-9-124:~$ chmod +x ./kubectl
ubuntu@ip-172-31-9-124:~$ sudo mv kubectl /usr/local/bin/
```

Step 3: Start MiniKube on Ubuntu 22.04

To check kubectl version on ubuntu

```
ubuntu@ip-172-31-9-124:~$ kubectl version --client --output=yaml
clientVersion:
  buildDate: "2024-06-11T20:29:44Z"
  compiler: gc
  gitCommit: 39683505b630ff2121012f3c5b16215a1449d5ed
  gitTreeState: clean
  gitVersion: v1.30.2
  goVersion: go1.22.4
  major: "1"
  minor: "30"
  platform: linux/amd64
kustomizeVersion: v5.0.4-0.20230601165947-6ce0bf390ce3
```

Start the minikube kubernates cluster on ubuntu.

```
ubuntu@ip-172-31-9-124:~$ minikube start --vm-driver docker
* minikube v1.33.1 on Ubuntu 24.04
* Using the docker driver based on user configuration
* Using Docker driver with root privileges
* Starting "minikube" primary control-plane node in "minikube" cluster
* Pulling base image v0.0.44 ...
* Downloading Kubernetes v1.30.0 preload ...
    > preloaded-images-k8s-v18-v1...:  342.90 MiB / 342.90 MiB  100.00% 15.04 M
    > gcr.io/k8s-minikube/kicbase...:  481.58 MiB / 481.58 MiB  100.00% 14.97 M
* Creating docker container (CPUs=2, Memory=2200MB) ...
* Preparing Kubernetes v1.30.0 on Docker 26.1.1 ...
  - Generating certificates and keys ...
  - Booting up control plane ...
  - Configuring RBAC rules ...
* Configuring bridge CNI (Container Networking Interface) ...
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Enabled addons: default-storageclass, storage-provisioner
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
ubuntu@ip-172-31-9-124:~$ minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured
```

To Check the status of Minikube.

```
ubuntu@ip-172-31-9-124:~$ minikube start --driver=docker
* minikube v1.33.1 on Ubuntu 24.04
* Using the docker driver based on existing profile
* Starting "minikube" primary control-plane node in "minikube" cluster
* Pulling base image v0.0.44 ...
* Updating the running docker "minikube" container ...
* Preparing Kubernetes v1.30.0 on Docker 26.1.1 ...
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Enabled addons: storage-provisioner, default-storageclass
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```

Step #4: Install ArgoCD on Minikube

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

We are using driver as docker

Just like other kubernates tools, ArgoCD requires a namespace with its name. Therefore, we will create a namespace for argocd.

```
ubuntu@ip-172-31-9-124:~$ kubectl create ns argocd
namespace/argocd created
```

ArgoCD can be installed using its manifests. First, you'll need to download these manifests and apply them to your Minikube cluster.

```
ubuntu@ip-172-31-9-124:~$ kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/v2.5.8/manifests/install.yaml
customresourcedefinition.apiextensions.k8s.io/applications.argoproj.io created
customresourcedefinition.apiextensions.k8s.io/applicationsets.argoproj.io created
customresourcedefinition.apiextensions.k8s.io/appprojects.argoproj.io created
serviceaccount/argocd-application-controller created
serviceaccount/argocd-applicationset-controller created
serviceaccount/argocd-dex-server created
serviceaccount/argocd-notifications-controller created
serviceaccount/argocd-redis created
serviceaccount/argocd-repo-server created
serviceaccount/argocd-server created
role.rbac.authorization.k8s.io/argocd-application-controller created
role.rbac.authorization.k8s.io/argocd-applicationset-controller created
role.rbac.authorization.k8s.io/argocd-dex-server created
role.rbac.authorization.k8s.io/argocd-notifications-controller created
```

```
ubuntu@ip-172-31-9-124:~$ kubectl rollout restart deployment argocd-server -n argocd
deployment.apps/argocd-server restarted
ubuntu@ip-172-31-9-124:~$ kubectl describe service argocd-server-nodeport -n argocd
Name:                     argocd-server-nodeport
Namespace:                argocd
Labels:                   app=argocd-server
Annotations:              <none>
Selector:                 app=argocd-server
Type:                     NodePort
IP Family Policy:         SingleStack
IP Families:              IPv4
IP:                       10.101.39.213
IPs:                      10.101.39.213
Port:                     <unset>  80/TCP
TargetPort:               8080/TCP
NodePort:                 <unset>  30007/TCP
Endpoints:                <none>
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>
ubuntu@ip-172-31-9-124:~$ curl http://<NODE_IP>:30007
bash: NODE_IP: No such file or directory
ubuntu@ip-172-31-9-124:~$ curl http://<10.101.39.213>:30007
bash: 10.101.39.213: No such file or directory
ubuntu@ip-172-31-9-124:~$ kubectl rollout restart deployment argocd-server -n argocd
deployment.apps/argocd-server restarted
ubuntu@ip-172-31-9-124:~$ kubectl describe service argocd-server-nodeport -n argocd
Name:                     argocd-server-nodeport
Namespace:                argocd
Labels:                   app=argocd-server
Annotations:              <none>
Selector:                 app=argocd-server
Type:                     NodePort
IP Family Policy:         SingleStack
IP Families:              IPv4
IP:                       10.101.39.213
IPs:                      10.101.39.213
Port:                     <unset>  80/TCP
TargetPort:               8080/TCP
NodePort:                 <unset>  30007/TCP
Endpoints:                <none>
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>
```

This will create the necessary resources for ArgoCD within the argocd namespace.

```
ubuntu@ip-172-31-9-124:~$ kubectl get pods -n argocd
NAME                                                READY   STATUS    RESTARTS   AGE
argocd-application-controller-0                     1/1     Running   0          49m
argocd-applicationset-controller-765f86b44f-cwlrc   1/1     Running   0          49m
argocd-dex-server-56774b4f5b-kndjr                  1/1     Running   0          49m
argocd-notifications-controller-6c9cf9df94-4qn96    1/1     Running   0          49m
argocd-redis-6457544dd9-26bp4                       1/1     Running   0          49m
argocd-repo-server-5cdb9fbbb4-485rb                 1/1     Running   0          49m
argocd-server-f5c5487bf-wwkbh                       1/1     Running   0          49m
```

Let’s verify the Installation by getting all the objects in the ArgoCD namespace.

```
^Cubuntu@ip-172-31-9-124:~$ kubect get all -n argocd
NAME                                                READY   STATUS    RESTARTS   AGE
pod/argocd-application-controller-0                 1/1     Running   0          6m34s
pod/argocd-applicationset-controller-765f86b44f-cwlrc   1/1     Running   0          6m35s
pod/argocd-dex-server-56774b4f5b-kndjr              1/1     Running   0          6m35s
pod/argocd-notifications-controller-6c9cf9df94-4qn96    1/1     Running   0          6m35s
pod/argocd-redis-6457544dd9-26bp4                   1/1     Running   0          6m35s
pod/argocd-repo-server-5cdb9fbbb4-485rb             1/1     Running   0          6m35s
pod/argocd-server-f5c5487bf-wwkbh                   1/1     Running   0          6m35s

NAME                                          TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)                      AGE
service/argocd-applicationset-controller      ClusterIP   10.104.225.111   <none>        7000/TCP,8080/TCP            6m36s
service/argocd-dex-server                     ClusterIP   10.107.167.191   <none>        5556/TCP,5557/TCP,5558/TCP   6m36s
service/argocd-metrics                        ClusterIP   10.103.60.223    <none>        8082/TCP                     6m36s
service/argocd-notifications-controller-metrics   ClusterIP   10.100.139.141   <none>        9001/TCP                     6m36s
service/argocd-redis                          ClusterIP   10.108.87.139    <none>        6379/TCP                     6m35s
service/argocd-repo-server                    ClusterIP   10.109.245.119   <none>        8081/TCP,8084/TCP            6m35s
service/argocd-server                         ClusterIP   10.101.176.65    <none>        80/TCP,443/TCP               6m35s
service/argocd-server-metrics                 ClusterIP   10.107.208.110   <none>        8083/TCP                     6m35s

NAME                                           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/argocd-applicationset-controller   1/1     1            1           6m35s
deployment.apps/argocd-dex-server              1/1     1            1           6m35s
deployment.apps/argocd-notifications-controller    1/1     1            1           6m35s
deployment.apps/argocd-redis                   1/1     1            1           6m35s
deployment.apps/argocd-repo-server             1/1     1            1           6m35s
deployment.apps/argocd-server                  1/1     1            1           6m35s

NAME                                                      DESIRED   CURRENT   READY   AGE
replicaset.apps/argocd-applicationset-controller-765f86b44f   1         1         1       6m35s
replicaset.apps/argocd-dex-server-56774b4f5b              1         1         1       6m35s
replicaset.apps/argocd-notifications-controller-6c9cf9df94    1         1         1       6m35s
replicaset.apps/argocd-redis-6457544dd9                   1         1         1       6m35s
replicaset.apps/argocd-repo-server-5cdb9fbbb4             1         1         1       6m35s
replicaset.apps/argocd-server-f5c5487bf                   1         1         1       6m35s

NAME                                          READY   AGE
statefulset.apps/argocd-application-controller   1/1     6m35s
```

Step 5: Access ArgoCD UI on Browser

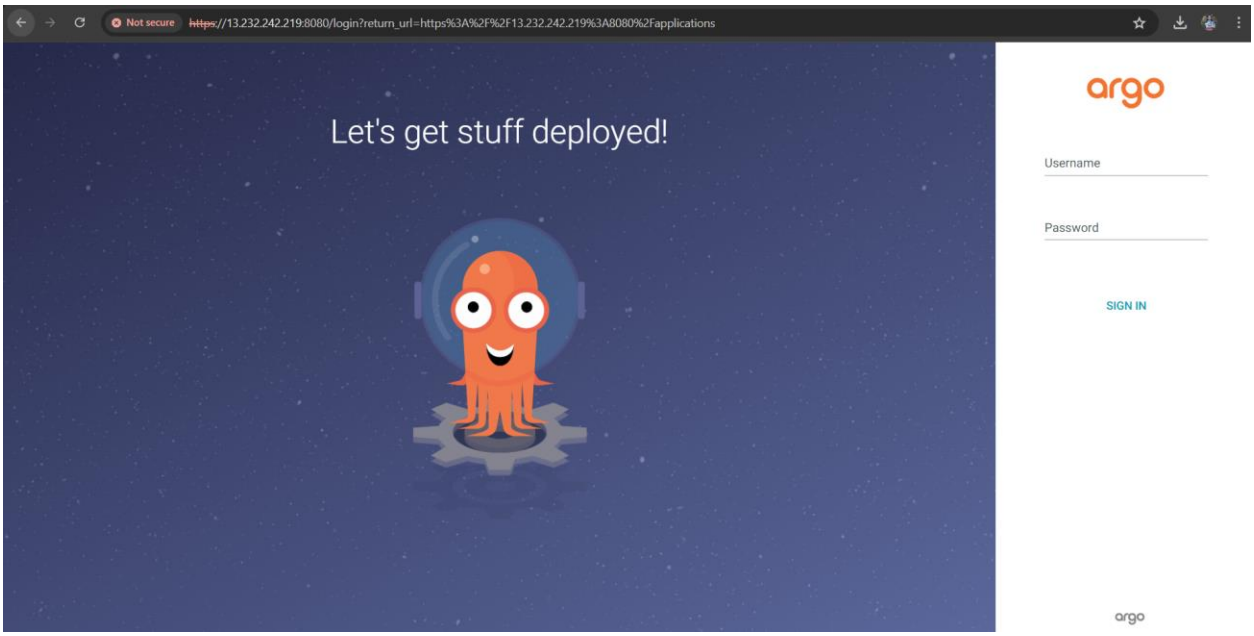
By default, the ArgoCD server is not exposed outside the cluster. You can expose it using port-forwarding to access the ArgoCD UI.

```
^Cubuntu@ip-172-31-9-124:~kubectl port-forward service/argocd-server 8090:80 -n argocdcd
Forwarding from 127.0.0.1:8090 -> 8080
Forwarding from [::1]:8090 -> 8080
```

The ArgoCD UI will be available at http://localhost/IP:8080. Access it through your web

browser.Now we can go to a browser and open instance_ip:8080

You will see a privacy warning. Just ignore the warning, click on Advanced and then hit on Proceed to localhost (unsafe) to continue to the GUI interface.



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^Cubuntu@ip-172-31-9-124:~$ kubectl -n argocd get secret argocd-initial-
onpath="{.data.password}" kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath="{.data.password}" | base64 -d
ZSiOXy3bAzth0CFWubuntu@ip-kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath="{.data.password}" | base64 -ord}" | base64 -d
ZSiOXy3bAzth0CFWubuntu@ip-172-31-9-124:~$
```

Use the generated password to log in as the admin user in the ArgoCD UI.

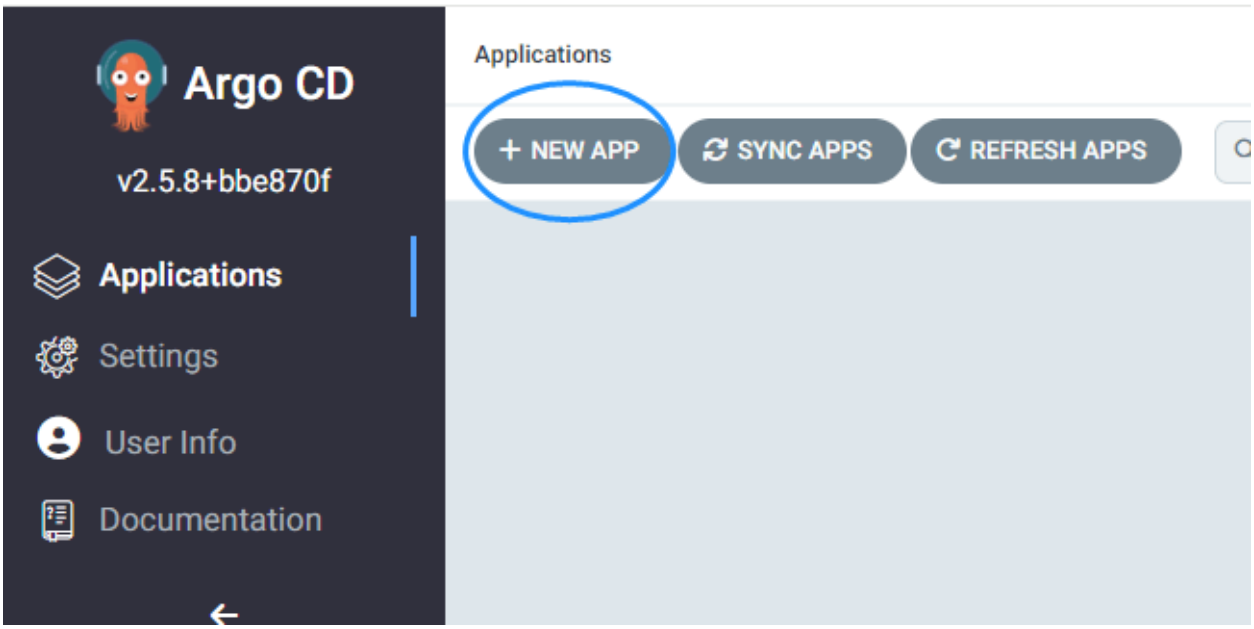
We have covered Install ArgoCD on minikube.

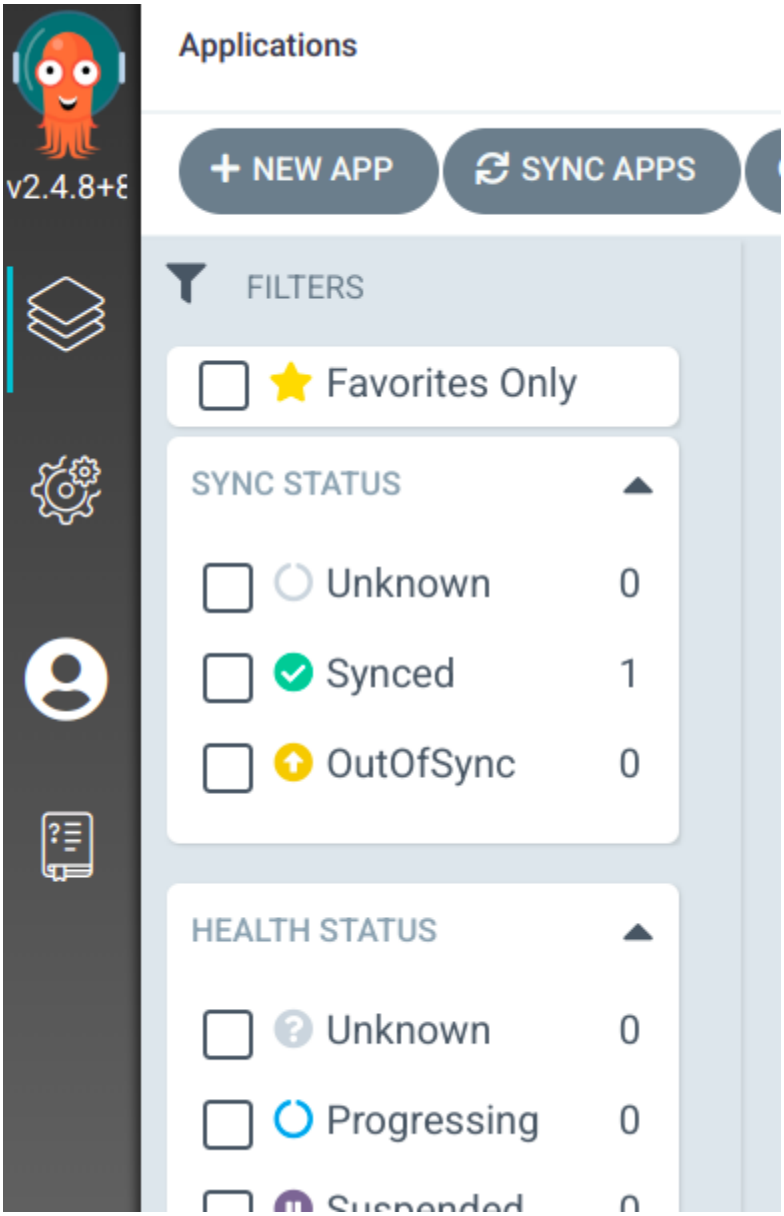
Step 6: Deploying an app on ArgoCD

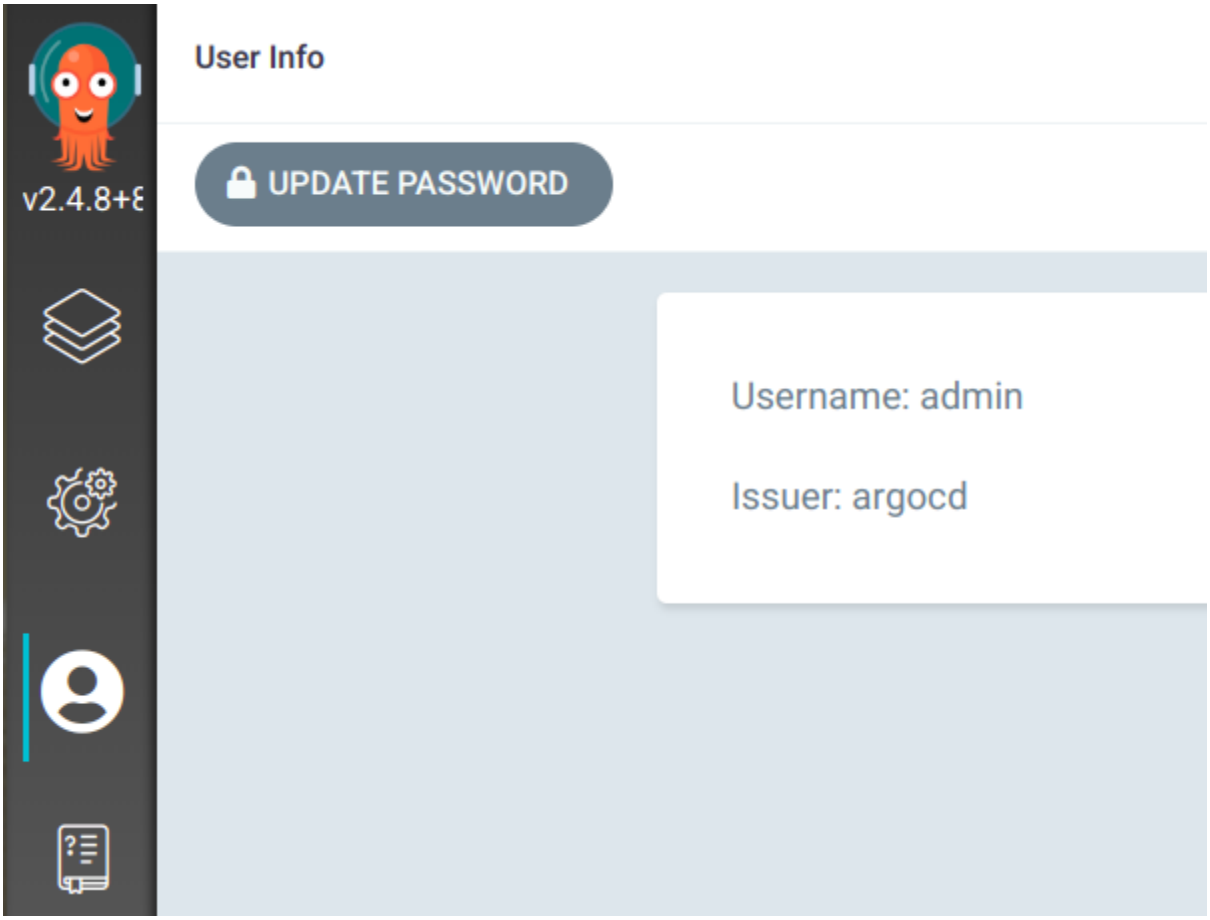
Now that we are in the user interface, we can create a new _app. The source code for my _application is in my

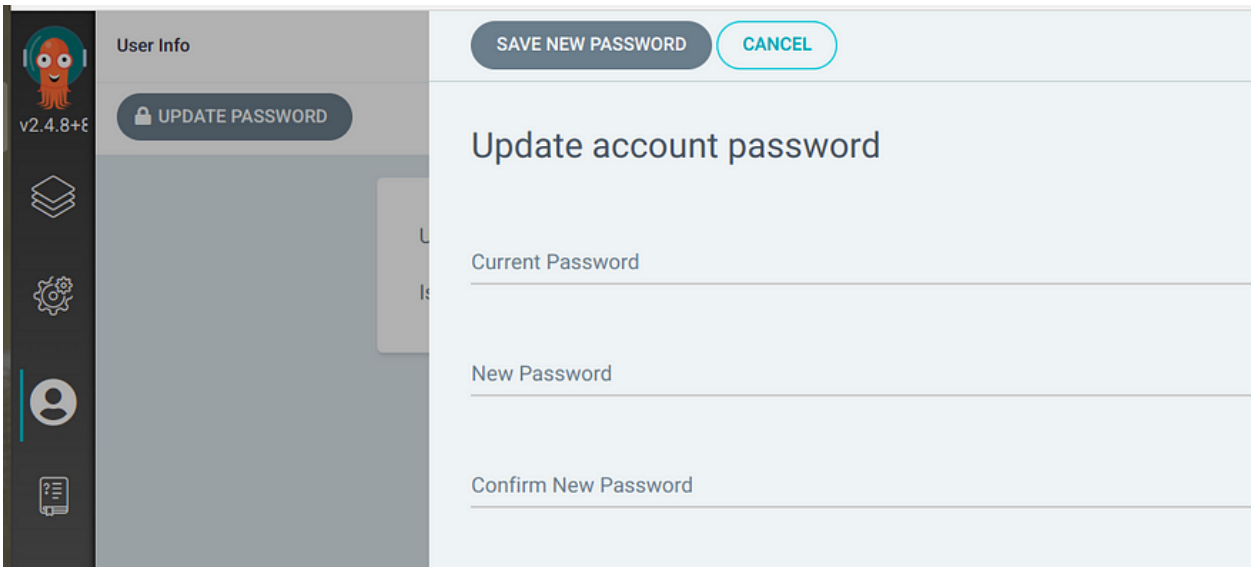
Github repo, so We will connect my Github repo to ArgoCD.

After logging in, click the **+ New App** button as shown below:




Applications
v2.4.8+8
+ NEW APP
SYNC APPS
FILTERS
Favorites Only
SYNC STATUS
Unknown 0
Synced 1
OutOfSync 0
HEALTH STATUS
Unknown 0
Progressing 0


User Info
v2.4.8+8
UPDATE PASSWORD
Username: admin
Issuer: argocd


User Info
v2.4.8+8
UPDATE PASSWORD
SAVE NEW PASSWORD
CANCEL
Update account password
Current Password
New Password
Confirm New Password

Conclusion

Congratulation's….! you have installed ArgoCD in local Ubuntu machine.