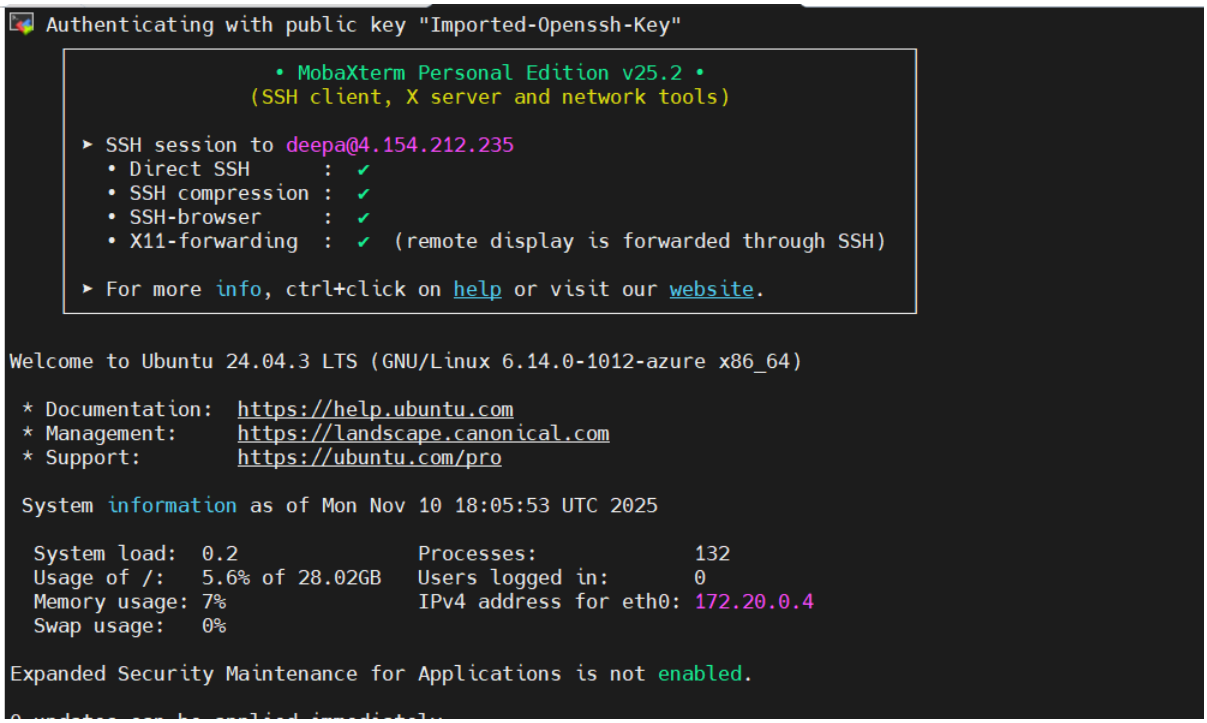
To set up ArgoCD with minikube:

Created VM in azure portal.



Update and install essentials:

sudo apt update

sudo apt upgrade -y

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

```
deepa@deepa-lin:~$ sudo apt update
sudo apt upgrade -y
sudo apt install -y curl wget git apt-transport-https conntrack
Hit:1 http://azure.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://azure.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://azure.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://azure.archive.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:5 http://azure.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:6 http://azure.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
Get:7 http://azure.archive.ubuntu.com/ubuntu noble/universe amd64 Components [3871 kB]
Get:8 http://azure.archive.ubuntu.com/ubuntu noble/universe amd64 c-n-f Metadata [301 kB]
Get:9 http://azure.archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [269 kB]
Get:10 http://azure.archive.ubuntu.com/ubuntu noble/multiverse Translation-en [118 kB]
Get:11 http://azure.archive.ubuntu.com/ubuntu noble/multiverse amd64 Components [35.0 kB]
Get:12 http://azure.archive.ubuntu.com/ubuntu noble/multiverse amd64 c-n-f Metadata [8328 B]
```

```
Selecting previously unselected package conntrack.
Preparing to unpack .../conntrack_1%3a1.4.8-1ubuntu1_amd64.deb ...
Unpacking conntrack (1:1.4.8-1ubuntu1) ...
Setting up apt-transport-https (2.8.3) ...
Setting up conntrack (1:1.4.8-1ubuntu1) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning candidates...
Scanning linux images...

Pending kernel upgrade!
Running kernel version:
  6.14.0-1012-azure
Diagnostics:
  The currently running kernel version is not the expected kernel version 6.14.0-1014-azure.

Restarting the system to load the new kernel will not be handled automatically, so you should consider rebooting.

Restarting services...

Service restarts being deferred:
 /etc/needrestart/restart.d/dbus.service
 systemctl restart networkd-dispatcher.service
 systemctl restart systemd-logind.service
 systemctl restart unattended-upgrades.service

No containers need to be restarted.

User sessions running outdated binaries:
 deepa @ session #2: sshd[1684]
 deepa @ session #4: sshd[1686]
 deepa @ user manager service: systemd[1690]

No VM guests are running outdated hypervisor (qemu) binaries on this host.
deepa@deepa-lin:~$
```

Install kubectl

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

chmod +x kubectl

sudo mv kubectl /usr/local/bin/

kubectl version --client

```
deepa@deepa-lin:~$ chmod +x kubectl
chmod: cannot access 'kubectl': No such file or directory
deepa@deepa-lin:~$ sudo mv kubectl /usr/local/bin/
kubectl version --client
^[[201~
mv: cannot stat 'kubectl': No such file or directory
deepa@deepa-lin:~$ curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
chmod +x kubectl
sudo mv kubectl /usr/local/bin/
kubectl version --client
curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
chmod +x kubectl
sudo mv kubectl /usr/local/bin/
kubectl version --client
curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
chmod +x kubectl
sudo mv kubectl /usr/local/bin/
kubectl version --client
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   138  100   138    0     0   1318      0 --:--:-- --:--:-- --:--:--  1326
100 57.7M  100 57.7M    0     0   114M      0 --:--:-- --:--:-- --:--:--  114M
Client Version: v1.34.1
Kustomize Version: v5.7.1
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   138  100   138    0     0   1230      0 --:--:-- --:--:-- --:--:--  1243
100 57.7M  100 57.7M    0     0  99.9M      0 --:--:-- --:--:-- --:--:-- 99.9M
Client Version: v1.34.1
Kustomize Version: v5.7.1
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   138  100   138    0     0   1312      0 --:--:-- --:--:-- --:--:--  1326
100 57.7M  100 57.7M    0     0   101M      0 --:--:-- --:--:-- --:--:--  101M
Client Version: v1.34.1
Kustomize Version: v5.7.1
deepa@deepa-lin:~$
```

Install Minikube

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

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

minikube version

```
deepa@deepa-lin:~$
deepa@deepa-lin:~$ curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
sudo install minikube-linux-amd64 /usr/local/bin/minikube
minikube version
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  133M  100  133M    0     0  91.7M      0  0:00:01  0:00:01 --:--:-- 91.8M
minikube version: v1.37.0
commit: 65318f4cfff9c12cc87ec9eb8f4cdd57b25047f3
deepa@deepa-lin:~$
```


```
deepa@deepa-lin:~$
deepa@deepa-lin:~$ curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
sudo install minikube-linux-amd64 /usr/local/bin/minikube
minikube version
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  133M  100  133M    0     0  91.7M      0  0:00:01  0:00:01 --:--:-- 91.8M
minikube version: v1.37.0
commit: 65318f4cfff9c12cc87ec9eb8f4cdd57b25047f3
deepa@deepa-lin:~$
```


minikube start --driver=docker

```
deepa@deepa-lin:~$
deepa@deepa-lin:~$ minikube start --driver=docker
* minikube v1.37.0 on Ubuntu 24.04
* Using the docker driver based on user configuration

X The requested memory allocation of 3072MiB does not leave room for system overhead (total system memory: 3867MiB). You may face stability issues.
* Suggestion: Start minikube with less memory allocated: 'minikube start --memory=3072mb'

* Using Docker driver with root privileges
* Starting "minikube" primary control-plane node in "minikube" cluster
* Pulling base image v0.0.48 ...
* Downloading Kubernetes v1.34.0 preload ...
    > gcr.io/k8s-minikube/kicbase...:  488.51 MiB / 488.52 MiB  100.00% 106.80
    > preloaded-images-k8s-v18-v1...:  337.07 MiB / 337.07 MiB  100.00% 57.34 M
* Creating docker container (CPUs=2, Memory=3072MB) ...
* Preparing Kubernetes v1.34.0 on Docker 28.4.0 ...
* Configuring bridge CNI (Container Networking Interface) ...
* 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
deepa@deepa-lin:~$
```

kubectl get nodes

```
deepa@deepa-lin:~$ kubectl get nodes
NAME       STATUS   ROLES           AGE   VERSION
minikube   Ready    control-plane   52s   v1.34.0
deepa@deepa-lin:~$
```

Install Helm

curl https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3 | bash

helm version

```
deepa@deepa-lin:~$ curl https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3 | bash
helm version
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 11928  100 11928    0     0   161k      0 --:--:-- --:--:-- --:--:--  164k
Downloading https://get.helm.sh/helm-v3.19.0-linux-amd64.tar.gz
Verifying checksum... Done.
Preparing to install helm into /usr/local/bin
helm installed into /usr/local/bin/helm
version.BuildInfo{Version:"v3.19.0", GitCommit:"3d8990f0836691f0229297773f3524598f46bda6", GitTreeState:"clean", GoVersion:"go1.24.7"}
deepa@deepa-lin:~$
```

Verify Setup

```
deepa@deepa-lin:~$ docker ps              # Docker running
kubectl get nodes         # Minikube nodes
helm list -A              # Any Helm releases
CONTAINER ID   IMAGE                                COMMAND                  CREATED         STATUS         PORTS
                                                                                NAMES
c6c036239664   gcr.io/k8s-minikube/kicbase:v0.0.48   "/usr/local/bin/entr…"   3 minutes ago   Up 3 minutes   127.0.0.1:32768->22/tcp, 127.0.0.1:32769->2376/tcp, 127.0.0.1:3
2770->5000/tcp, 127.0.0.1:32771->8443/tcp, 127.0.0.1:32772->32443/tcp   minikube
NAME       STATUS   ROLES           AGE   VERSION
minikube   Ready    control-plane   3m6s   v1.34.0
NAME    NAMESPACE       REVISION        UPDATED STATUS  CHART   APP VERSION
deepa@deepa-lin:~$
```

Create a Namespace for Argo CD

kubectl create namespace argocd

```
deepa@deepa-lin:~$ kubectl create namespace argocd
namespace/argocd created
deepa@deepa-lin:~$
```

Deploy Argo CD

Apply the Argo CD manifests to your cluster:

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

This installs all Argo CD components: argocd-server, repo-server, controller, application-controller, etc.

```
deepa@deepa-lin:~$ kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml
Warning: unrecognized format "int64"
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
role.rbac.authorization.k8s.io/argocd-redis created
role.rbac.authorization.k8s.io/argocd-server created
clusterrole.rbac.authorization.k8s.io/argocd-application-controller created
clusterrole.rbac.authorization.k8s.io/argocd-applicationset-controller created
clusterrole.rbac.authorization.k8s.io/argocd-server created
rolebinding.rbac.authorization.k8s.io/argocd-application-controller created
rolebinding.rbac.authorization.k8s.io/argocd-applicationset-controller created
rolebinding.rbac.authorization.k8s.io/argocd-dex-server created
rolebinding.rbac.authorization.k8s.io/argocd-notifications-controller created
rolebinding.rbac.authorization.k8s.io/argocd-redis created
rolebinding.rbac.authorization.k8s.io/argocd-server created
clusterrolebinding.rbac.authorization.k8s.io/argocd-application-controller created
clusterrolebinding.rbac.authorization.k8s.io/argocd-applicationset-controller created
clusterrolebinding.rbac.authorization.k8s.io/argocd-server created
configmap/argocd-cm created
configmap/argocd-cmd-params-cm created
```

```
configmap/argocd-gpg-keys-cm created
configmap/argocd-notifications-cm created
configmap/argocd-rbac-cm created
configmap/argocd-ssh-known-hosts-cm created
configmap/argocd-tls-certs-cm created
secret/argocd-notifications-secret created
secret/argocd-secret created
service/argocd-applicationset-controller created
service/argocd-dex-server created
service/argocd-metrics created
service/argocd-notifications-controller-metrics created
service/argocd-redis created
service/argocd-repo-server created
service/argocd-server created
service/argocd-server-metrics created
deployment.apps/argocd-applicationset-controller created
deployment.apps/argocd-dex-server created
deployment.apps/argocd-notifications-controller created
deployment.apps/argocd-redis created
deployment.apps/argocd-repo-server created
deployment.apps/argocd-server created
statefulset.apps/argocd-application-controller created
networkpolicy.networking.k8s.io/argocd-application-controller-network-policy created
networkpolicy.networking.k8s.io/argocd-applicationset-controller-network-policy created
networkpolicy.networking.k8s.io/argocd-dex-server-network-policy created
networkpolicy.networking.k8s.io/argocd-notifications-controller-network-policy created
networkpolicy.networking.k8s.io/argocd-redis-network-policy created
networkpolicy.networking.k8s.io/argocd-repo-server-network-policy created
networkpolicy.networking.k8s.io/argocd-server-network-policy created
deepa@deepa-lin:~$
```

Accessing the Argo CD Server

Minikube (Local Cluster):

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

Use **port-forwarding** to access Argo CD locally:

```
deepa@deepa-lin:~$
deepa@deepa-lin:~$ kubectl port-forward svc/argocd-server -n argocd 8080:443
Forwarding from 127.0.0.1:8080 → 8080
Forwarding from [::1]:8080 → 8080
```

- Now Argo CD is accessible at: https://localhost:8080

Notice that the target port is **8080**, not 443. This means you are actually **forwarding port 8080 on the service** (HTTP), not 443 (HTTPS). That's why your browser is complaining when you try https://localhost:8080 — the service is serving **HTTP**, but your browser thinks it's HTTPS

Error:

argocd login localhost:8080 --insecure argocd: command not found

Install Argo CD CLI on Linux:

Also open the ports in Azure VM

Check ufw active?

sudo ufw status

```
deepa@deepa-lin:~$ sudo ufw status
Status: inactive
deepa@deepa-lin:~$
```

31086, 9073, 9443 and 80 ports opened, in portal.

Manually from VM:

sudo ufw allow 31086/tcp

sudo ufw reload

```
deepa@deepa-lin:~$ kubectl port-forward svc/argocd-server -n argocd 9443:443
Forwarding from 127.0.0.1:9443 → 8080
Forwarding from [::1]:9443 → 8080
```

It's listening **only on localhost (inside Azure VM)**
And forwarding **port 9443 on VM → port 8080 inside Argo CD service**

However, from outside (local computer):
still won't be able to reach it , because the port-forward is bound to 127.0.0.1 only.
External users (you, from outside Azure) can't hit that loopback interface.

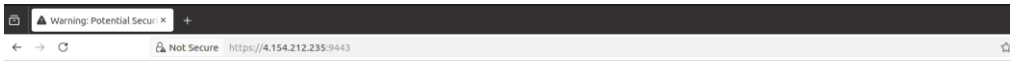
Step 1 — Allow external access
Stop the current port-forward (Ctrl + C), then run this instead:
kubectl port-forward --address 0.0.0.0 svc/argocd-server -n argocd 9443:443

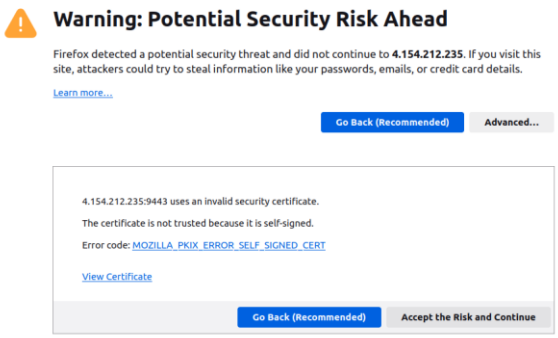
```
^Cdeepa@deepa-lin:~kubectl port-forward --address 0.0.0.0 svc/argocd-server -n argocd 9443:44343
Forwarding from 0.0.0.0:9443 → 8080
Handling connection for 9443
Handling connection for 9443
```

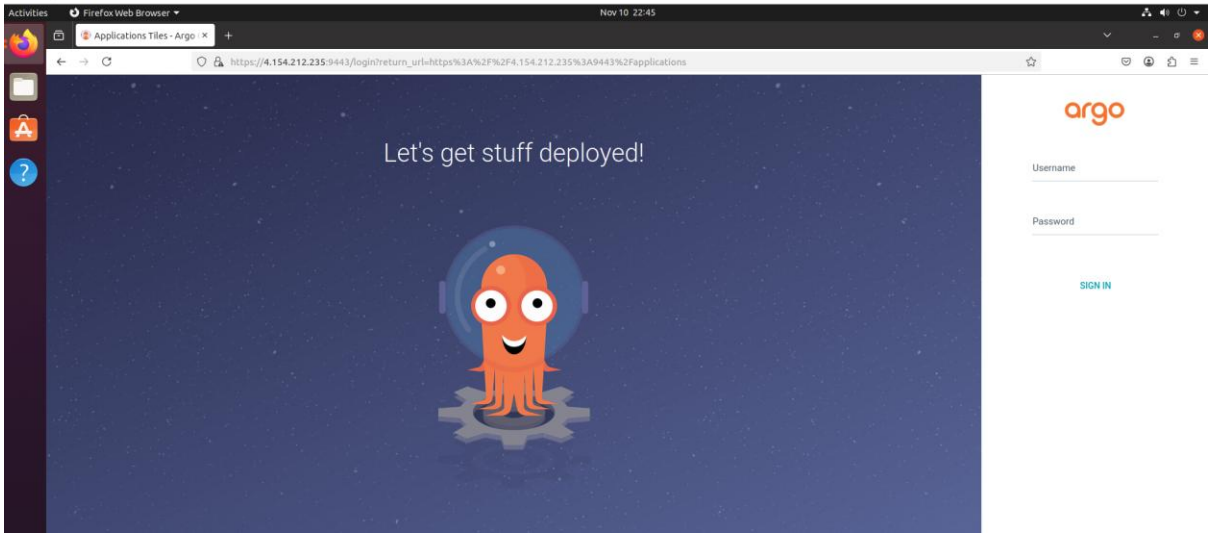
Step 2 — Make sure port 9443 is open in Azure NSG
Already opened port 31086 — now open 9443 as well:
In another terminal,

```
deepa@deepa-lin:~/K8s$ az network nsg rule create \
  --resource-group rg-deepa \
  --nsg-name deepa-lin-nsg \
  --name allow-argocd-portforward \
  --priority 1002 \
  --protocol Tcp \
  --destination-port-ranges 9443 \
  --access Allow \
  --direction Inbound \
  --description "Allow Argo CD port-forward access"
{
  "access": "Allow",
  "description": "Allow Argo CD port-forward access",
  "destinationAddressPrefix": "*",
  "destinationAddressPrefixes": [],
  "destinationPortRange": "9443",
  "destinationPortRanges": [],
  "direction": "Inbound",
  "etag": "W/\"9c0864f8-f139-4ddd-b3a5-7757dfc442ad\"",
  "id": "/subscriptions/fca64410-ff78-453e-b497-400ee8acf1a3/resourceGroups
ocd-portforward",
  "name": "allow-argocd-portforward",
  "priority": 1002,
  "protocol": "Tcp",
  "provisioningState": "Succeeded",
  "resourceGroup": "rg-deepa",
  "sourceAddressPrefix": "*",
  "sourceAddressPrefixes": [],
  "sourcePortRange": "*",
  "sourcePortRanges": [],
  "type": "Microsoft.Network/networkSecurityGroups/securityRules"
}
```

Now in browser local machine,
https://4.154.212.235:9443
Click **“Advanced → Accept the risk and continue.”**
Should now see the **Argo CD login page**



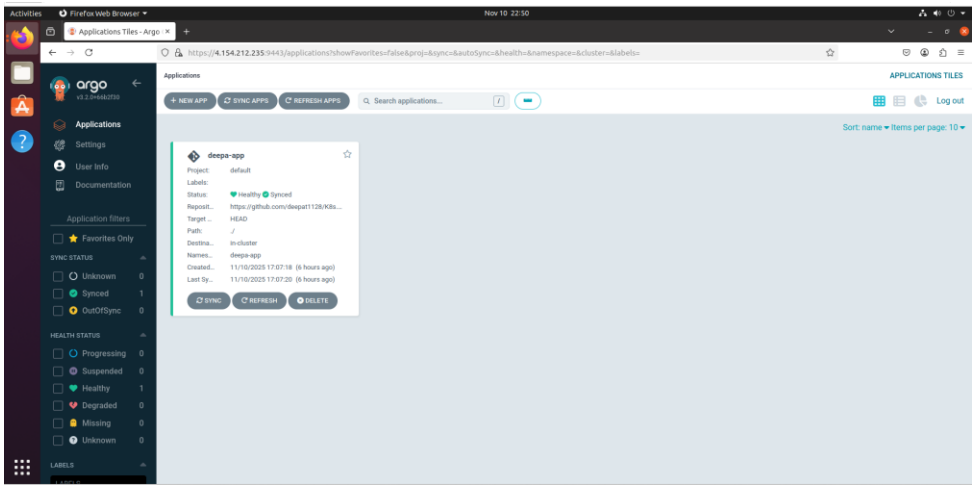





```
deepa@deepa-lin:~/K8s$ kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath="{.data.password}" | base64 -d; echo
vaOQttNWRadMCVmz
deepa@deepa-lin:~/K8s$
```

Username : admin

Password: vaOQttNWRadMCVmz



NodePort / port-forward working over HTTPS (9443)
NSG rules opened properly in Azure
Argo CD pods and service running in the argocd namespace
Now to practice:

Create new applications

- From GitHub (Helm/Kustomize/Manifest)
- Example: your repo → https://github.com/deepat1128/K8s.git

Enable auto-sync

- So that Argo CD keeps cluster in sync with Git changes.

Check health and status

- Explore the live state vs. desired state view.

(Optional) Set up RBAC, SSO, or connect Argo CD to multiple clusters.

Above screenshot shows, **application (deepa-app) is synced and healthy!**

verify it in the terminal:

```
deepa@deepa-lin:~/K8s$ kubectl get all -n deepa-app
NAME                                 READY   STATUS    RESTARTS        AGE
pod/deepadeploy-8495f7f86b-dxgm9     1/1     Running   1 (3h45m ago)   5h47m
pod/deepadeploy-8495f7f86b-kln77     1/1     Running   1 (3h45m ago)   5h47m
pod/deepadeploy-8495f7f86b-pqcrj     1/1     Running   1 (3h45m ago)   5h47m

NAME                   TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
service/deepadeploy    NodePort   10.102.60.114   <none>        80:30863/TCP   5h13m

NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/deepadeploy    3/3     3            3           5h47m

NAME                                       DESIRED   CURRENT   READY   AGE
replicaset.apps/deepadeploy-8495f7f86b     3         3         3       5h47m
deepa@deepa-lin:~/K8s$
deepa@deepa-lin:~/K8s$
```

http://192.168.49.2:30863 this still not connected. kubectl port-forward --address 0.0.0.0 svc/argocd-server -n argocd 9443:443,
https://4.154.212.235:9443 worked

“Listen on *all* network interfaces (0.0.0.0), not just localhost.”

Unless we use a minikube tunnel or LoadBalancer, those NodePorts are only reachable *inside the VM*.

If port forwarding is not required, then
Edit the svc,
kubectl edit svc argocd-server -n argocd
type cluster ip to LoadBalancer.
Then run, minikube tunnel
This will expose it with an external IP that you can always use in the browser.

Auto-sync Argo CD app with GitHub:
Whenever we push changes (like a new image tag or manifest update) to our GitHub repo,
Argo CD will **auto-detect and redeploy** your app.
Best if we want to practice *continuous delivery (CD)*.

Confirm your app name:
argocd app list

```
deepa@deepa-lin:~/K8s$
deepa@deepa-lin:~/K8s$ argocd app list
NAME            CLUSTER                         NAMESPACE  PROJECT  STATUS  HEALTH   SYNCPOLICY  CONDITIONS  REPO                                      PATH  TARGET
argocd/deepa-app  https://kubernetes.default.svc  deepa-app  default  Synced  Healthy  Auto        <none>      https://github.com/deepat1128/K8s.git  ./
deepa@deepa-lin:~/K8s$
```

SYNCPOLICY already shows Auto, then auto-sync is already enabled.

Otherwise, Enable Auto-Sync
argocd app set deepa-app --sync-policy automated

```
deepa@deepa-lin:~/K8s$
deepa@deepa-lin:~/K8s$ argocd app set deepa-app --sync-policy automated
deepa@deepa-lin:~/K8s$
deepa@deepa-lin:~/K8s$
```

Enable Auto-Prune and Self-Heal:
argocd app set deepa-app --auto-prune --self-heal
Auto-Prune → Deletes Kubernetes resources that were removed from Git.
Self-Heal → If someone manually changes a resource in the cluster, Argo CD reverts it to match Git.

```
deepa@deepa-lin:~/K8s$
deepa@deepa-lin:~/K8s$ argocd app set deepa-app --auto-prune --self-heal
deepa@deepa-lin:~/K8s$
```

Verify:

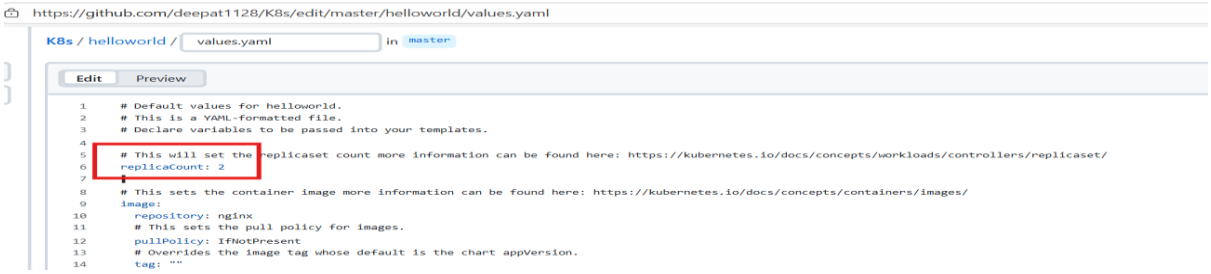
argocd app get deepa-app

```
deepa@deepa-lin:~/K8s$
deepa@deepa-lin:~/K8s$ argocd app get deepa-app
Name:               argocd/deepa-app
Project:            default
Server:             https://kubernetes.default.svc
Namespace:          deepa-app
URL:                https://192.168.49.2:31086/applications/deepa-app
Source:
- Repo:             https://github.com/deepat1128/K8s.git
  Target:
  Path:             ./
SyncWindow:         Sync Allowed
Sync Policy:        Automated (Prune)
Sync Status:        Synced to  (87e28b2)
Health Status:      Healthy

GROUP  KIND        NAMESPACE  NAME         STATUS  HEALTH   HOOK  MESSAGE
apps   Deployment  deepa-app  deepadeploy  Synced  Healthy        deployment.apps/deepadeploy created
deepa@deepa-lin:~/K8s$
deepa@deepa-lin:~/K8s$
```

Go to git repo:

https://github.com/deepat1128/K8s

Edit a file — for example, change the replica count in Deployment:



In Github changes replica 1 to 2 is committed.

But in ArgoCD , it is not triggered.

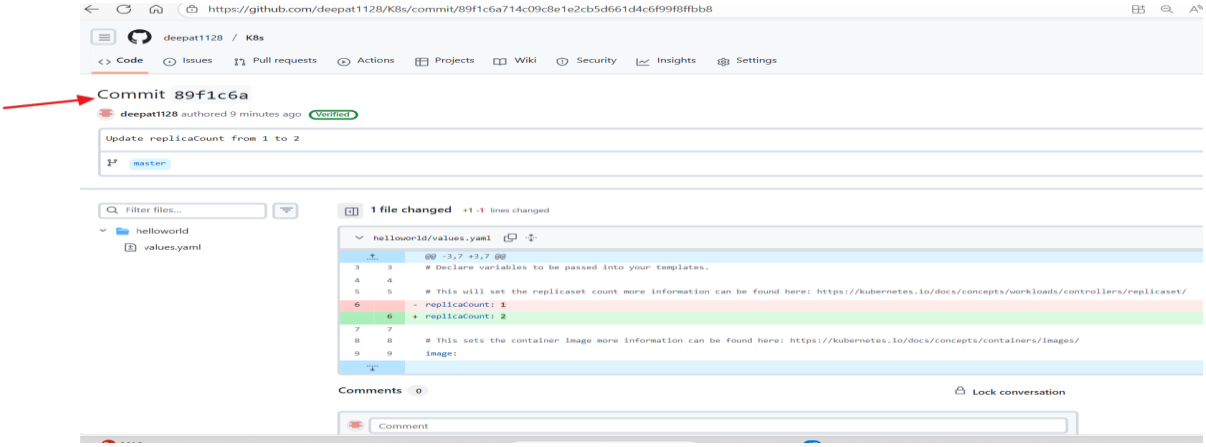
Force a quick refresh

argocd app refresh deepa-app

```
deepa@deepa-lin:~/K8s$
deepa@deepa-lin:~/K8s$
deepa@deepa-lin:~/K8s$ argocd app sync deepa-app --force
TIMESTAMP                  GROUP        KIND   NAMESPACE                  NAME    STATUS   HEALTH        HOOK  MESSAGE
2025-11-11T07:24:28+00:00   apps  Deployment   deepa-app           deepadeploy    Synced  Healthy

Name:               argocd/deepa-app
Project:            default
Server:             https://kubernetes.default.svc
Namespace:          deepa-app
URL:                https://192.168.49.2:31086/applications/deepa-app
Source:
- Repo:             https://github.com/deepat1128/K8s.git
  Target:
  Path:             ./
SyncWindow:         Sync Allowed
Sync Policy:        Automated (Prune)
Sync Status:        Synced to  (89f1c6a)
Health Status:      Healthy

Operation:          Sync
Sync Revision:      89f1c6a714c09c8e1e2cb5d661d4c6f99f8ffbb8
Phase:              Succeeded
Start:              2025-11-11 07:24:28 +0000 UTC
Finished:           2025-11-11 07:24:28 +0000 UTC
Duration:           0s
Message:            successfully synced (all tasks run)

GROUP  KIND        NAMESPACE  NAME         STATUS  HEALTH   HOOK  MESSAGE
apps   Deployment  deepa-app  deepadeploy  Synced  Healthy        deployment.apps/deepadeploy unchanged
deepa@deepa-lin:~/K8s$
```

Then check the app's status

argocd app get deepa-app



Github commit is updated in ArgoCD.

```
deepa@deepa-lin:~/K8s$ argocd app get deepa-app
Name:               argocd/deepa-app
Project:            default
Server:             https://kubernetes.default.svc
Namespace:          deepa-app
URL:                https://192.168.49.2:31086/applications/deepa-app
Source:
- Repo:             https://github.com/deepat1128/K8s.git
  Target:
  Path:             ./
SyncWindow:         Sync Allowed
Sync Policy:        Automated (Prune)
Sync Status:        Synced to  (89f1c6a)
Health Status:      Healthy

GROUP  KIND        NAMESPACE  NAME         STATUS  HEALTH   HOOK  MESSAGE
apps   Deployment  deepa-app  deepadeploy  Synced  Healthy        deployment.apps/deepadeploy unchanged
deepa@deepa-lin:~/K8s$
deepa@deepa-lin:~/K8s$
```

We can see on what commit it's running:

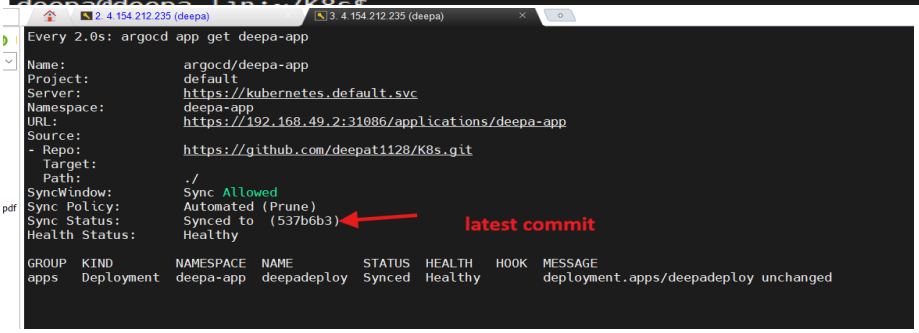
argocd app history deepa-app

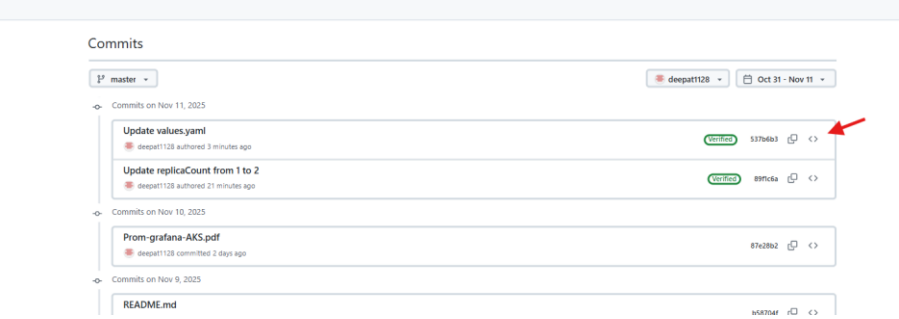
```
deepa@deepa-lin:~/K8s$
deepa@deepa-lin:~/K8s$ argocd app history deepa-app
SOURCE  https://github.com/deepat1128/K8s.git
ID      DATE                           REVISION
0       2025-11-11 01:07:20 +0000 UTC  (87e28b2)
1       2025-11-11 07:24:28 +0000 UTC  (89f1c6a)
deepa@deepa-lin:~/K8s$
deepa@deepa-lin:~/K8s$
```

Command to watch the app:

watch argocd app get deepa-app

```
deepa@deepa-lin:~/K8s$
deepa@deepa-lin:~/K8s$ watch argocd app get deepa-app
deepa@deepa-lin:~/K8s$
```





argocd app diff deepa-app

```
deepa@deepa-lin:~/K8s$
deepa@deepa-lin:~/K8s$ argocd app diff deepa-app
deepa@deepa-lin:~/K8s$
```

So, app is in sync.

What is Kustomize?

Kustomize is a Kubernetes-native configuration management tool.

We can create a **base** set of manifests (common to all environments) and then apply **overlays** (patches) for specific environments — without copying files.

Argo CD understands kustomization.yaml natively — no plugin needed.

Purpose: Multiple environment to be deployed with same config.

Step1: Set Up Your Git Repository Structure

In GitHub repo (https://github.com/deepat1128/K8s.git), create like this:

```
K8s/
 ├── base/
|    ├── deployment.yaml
|    ├── service.yaml
|    └── kustomization.yaml
└── overlays/
    ├── dev/
   |    ├── kustomization.yaml
   |    └── patch.yaml
    └── prod/
        ├── kustomization.yaml
        └── patch.yaml
```

Base/ contains common manifests – deployment, svc, cm etc, which are common across environments.

base/kustomization.yaml:

```
resources:
  - deployment.yaml
  - service.yaml
```

Overlays/dev : This is environment specific customization.

Create kustomization yaml

Other yaml files

Kubectl kustomize .

Overlays/pro : Production specific overrides.

Step2: Create overlays for each environment:

Three deployable apps inside ~/K8s repo that are ready for Argo CD GitOps automation:

- deepa-app/ → plain YAML (already working)
- helloworld/ → Helm chart (can be deployed via Argo CD next)
- prometheus/ → Helm chart (monitoring stack you can deploy later)

Commit and Push the Helm Chart
cd ~/K8s
git add .
git commit -m "Added helloworld Helm chart"
git push origin master

```
deepa@deepa-lin:~/K8s$ git push origin master
Password for 'https://deepat1128@github.com':
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 2 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 450 bytes | 450.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/deepat1128/K8s.git
   279273d..a2ca816  master → master
deepa@deepa-lin:~/K8s$
```

```
deepa@deepa-lin:~/K8s$ git config --global credential.helper store
deepa@deepa-lin:~/K8s$
deepa@deepa-lin:~/K8s$
deepa@deepa-lin:~/K8s$ argocd app get deepa-app
Name:               argocd/deepa-app
Project:            default
Server:             https://kubernetes.default.svc
Namespace:          deepa-app
URL:                https://192.168.49.2:31086/applications/deepa-app
Source:
- Repo:             https://github.com/deepat1128/K8s.git
  Target:
  Path:             ./
SyncWindow:         Sync Allowed
Sync Policy:        Automated (Prune)
Sync Status:        Synced to  (279273d)
Health Status:      Healthy

GROUP  KIND        NAMESPACE  NAME         STATUS  HEALTH   HOOK  MESSAGE
apps   Deployment  deepa-app  deepadeploy  Synced  Healthy        deployment.apps/deepadeploy unchanged
deepa@deepa-lin:~/K8s$
```

```
deepa@deepa-lin:~/K8s/deepa-app$ ls
deepa.yaml  deploy.yaml  kustomization.yaml  service.yaml
deepa@deepa-lin:~/K8s/deepa-app$ kubectl kustomize .
```

```
deepa@deepa-lin:~/K8s/deepa-app$ tree
.
├── deepa.yaml
├── deploy.yaml
├── kustomization.yaml
└── service.yaml

1 directory, 4 files
deepa@deepa-lin:~/K8s/deepa-app$
```

combined YAML output (namespace + labels + both manifests).

```
deepa@deepa-lin:~/K8s/deepa-app$ cp ~/K8s/deepa.yaml ~/K8s/deepa-app/deploy.yaml
deepa@deepa-lin:~/K8s/deepa-app$ ls
deepa.yaml  deploy.yaml  kustomization.yaml  service.yaml
deepa@deepa-lin:~/K8s/deepa-app$ kubectl kustomize .
apiVersion: v1
kind: Service
metadata:
  labels:
    app: deepadeploy
  name: deepadeploy
  namespace: deepa-app
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: deepadeploy
  type: NodePort
---
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: deepadeploy
  name: deepadeploy
  namespace: deepa-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: deepadeploy
  strategy: {}
  template:
    metadata:
      labels:
        app: deepadeploy
    spec:
      containers:
      - image: deepat1128/gitdemo1imagedt:1.0
        name: gitdemo1imagedt
        ports:
        - containerPort: 9090
        resources: {}
status: {}
deepa@deepa-lin:~/K8s/deepa-app$
```

Apply :

```
deepa@deepa-lin:~/K8s/deepa-app$ kubectl apply -k .
service/deepadeploy configured
deployment.apps/deepadeploy configured
deepa@deepa-lin:~/K8s/deepa-app$
deepa@deepa-lin:~/K8s/deepa-app$
deepa@deepa-lin:~/K8s/deepa-app$
```

confirm it’s deployed:

kubectl get all -n deepa-app

```
deepa@deepa-lin:~/K8s/deepa-app$
deepa@deepa-lin:~/K8s/deepa-app$ kubectl get all -n deepa-app
NAME                                READY   STATUS    RESTARTS      AGE
pod/deepadeploy-8495f7f86b-dxgm9    1/1     Running   2 (12h ago)   19h
pod/deepadeploy-8495f7f86b-kln77    1/1     Running   2 (12h ago)   19h
pod/deepadeploy-8495f7f86b-pqcrj    1/1     Running   2 (12h ago)   19h

NAME                   TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
service/deepadeploy    NodePort   10.102.60.114   <none>        80:30863/TCP   19h

NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/deepadeploy    3/3     3            3           19h

NAME                                      DESIRED   CURRENT   READY   AGE
replicaset.apps/deepadeploy-8495f7f86b    3         3         3       19h
deepa@deepa-lin:~/K8s/deepa-app$
```

To access ArgoCD in browser:

```
deepa@deepa-lin:~/K8s/deepa-app$ minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured

deepa@deepa-lin:~/K8s/deepa-app$
```

Minikube cluster is running inside the VM (deepa-lin).
browser on **host machine** can't open
http://192.168.49.2:30863 — that IP belongs **inside** Ubuntu VM's virtual network

Open the port (e.g., 8080) in VM firewall / Azure NSG:
sudo ufw allow 8080

```
deepa@deepa-lin:~/K8s/deepa-app$ sudo ufw allow 8080
Rules updated
Rules updated (v6)
deepa@deepa-lin:~/K8s/deepa-app$
```

Able to connect outside the VM: in my local system

```
deepa@ubuntu:~/devops/github/K8s$ telnet 4.154.212.235 9443
Trying 4.154.212.235...
Connected to 4.154.212.235.
Escape character is '^]'.
```

```
deepa@deepa-lin:~$ kubectl port-forward --address 0.0.0.0 svc/argocd-server -n argocd 9443:443
Forwarding from 0.0.0.0:9443 → 8080
Handling connection for 9443
Handling connection for 9443
Handling connection for 9443
Handling connection for 9443
Handling connection for 9443
Handling connection for 9443
```

https://4.154.212.235:9443

```
deepa@deepa-lin:~/K8s$ git add .
git commit -m "deepa-app"
git push origin master
[master dea252b] deepa-app
 Committer: Ubuntu <deepa@deepa-lin.0npswh3u0twurijglpnxerazfg.xx.internal.cloudapp.net>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly:

    git config --global user.name "Your Name"
    git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

 4 files changed, 73 insertions(+)
 create mode 100644 deepa-app/deepa.yaml
 create mode 100644 deepa-app/deploy.yaml
 create mode 100644 deepa-app/kustomization.yaml
 create mode 100644 deepa-app/service.yaml
Password for 'https://deepat1128@github.com':
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 2 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 547 bytes | 547.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/deepat1128/K8s.git
   a2ca816..dea252b  master -> master
deepa@deepa-lin:~/K8s$ git config --global credential.helper store
deepa@deepa-lin:~/K8s$ git push origin master
Everything up-to-date
deepa@deepa-lin:~/K8s$
```

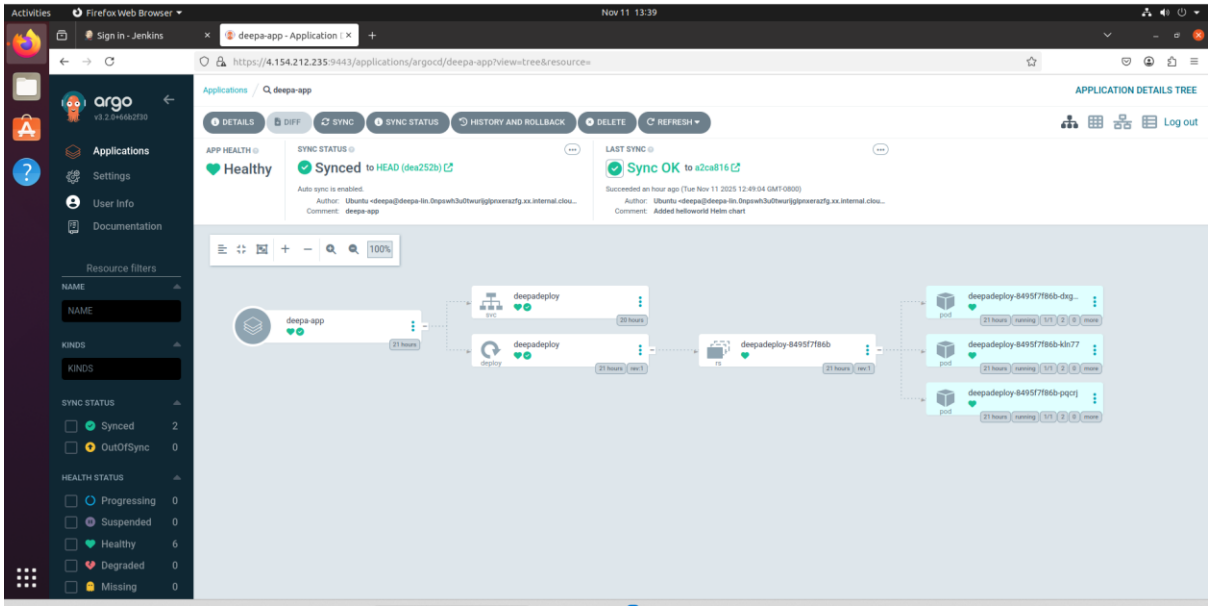
Stored the credentials.

```
deepa@deepa-lin:~/K8s$ argocd app get deepa-app
Name:               argocd/deepa-app
Project:            default
Server:             https://kubernetes.default.svc
Namespace:          deepa-app
URL:                https://192.168.49.2:31086/applications/deepa-app
Source:
- Repo:             https://github.com/deepat1128/K8s.git
  Target:
  Path:             ./
SyncWindow:         Sync Allowed
Sync Policy:        Automated (Prune)
Sync Status:        Synced to  (dea252b)
Health Status:      Healthy

GROUP  KIND        NAMESPACE  NAME         STATUS  HEALTH   HOOK  MESSAGE
apps   Deployment  deepa-app  deepadeploy  Synced  Healthy        deployment.apps/deepadeploy configured
       Service     deepa-app  deepadeploy  Synced  Healthy
deepa@deepa-lin:~/K8s$
```

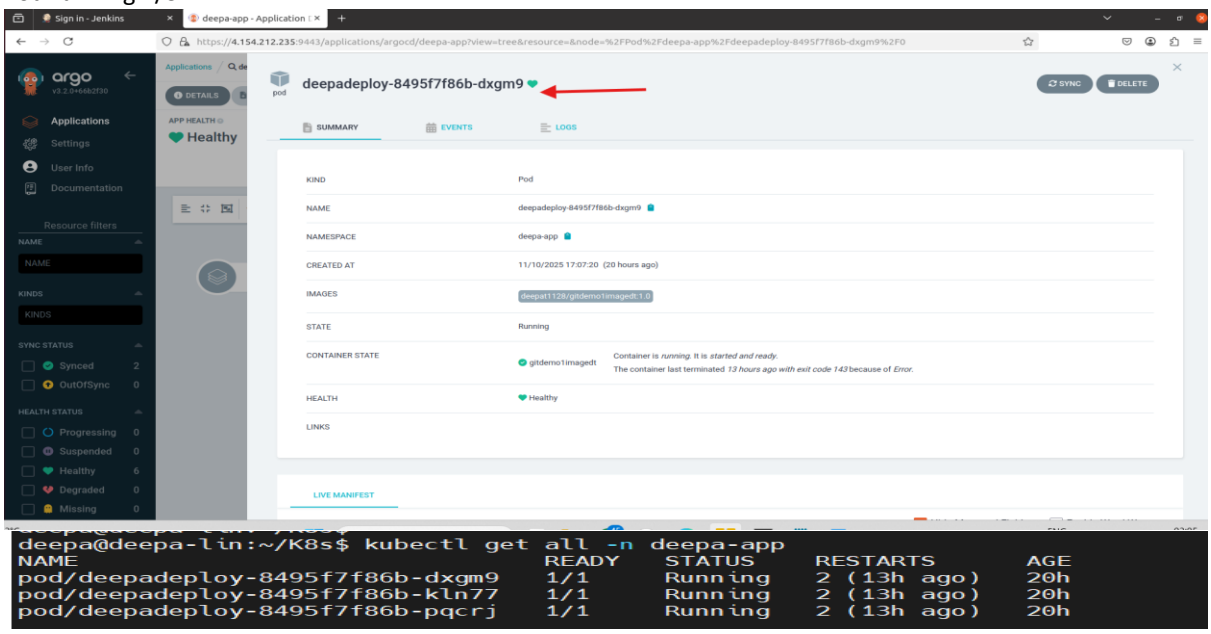
```
deepa@deepa-lin:~/K8s$ kubectl logs pod/deepadeploy-8495f7f86b-dxgm9 -n deepa-app

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.7.0)

2025-11-11 19:37:42.315  INFO 1 --- [           main] com.example.demo1.Demo1Application       : Starting Demo1Application v0.0.1-SNAPSHOT using Java 17.0.16 on deepadeplo
-8495f7f86b-dxgm9 with PID 1 (/app/app.jar started by root in /app)
2025-11-11 19:37:42.404  INFO 1 --- [           main] com.example.demo1.Demo1Application       : No active profile set, falling back to 1 default profile: "default"
2025-11-11 19:37:50.120  INFO 1 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2025-11-11 19:37:50.670  INFO 1 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 500 ms. Found 1 JPA repository
interfaces.
2025-11-11 19:37:54.115  INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080 (http)
2025-11-11 19:37:54.197  INFO 1 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2025-11-11 19:37:54.198  INFO 1 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.63]
2025-11-11 19:37:54.726  INFO 1 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2025-11-11 19:37:54.727  INFO 1 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 10479 ms
2025-11-11 19:37:54.858  INFO 1 --- [           main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
2025-11-11 19:37:56.358  INFO 1 --- [           main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
2025-11-11 19:37:56.394  INFO 1 --- [           main] o.s.b.a.h2.H2ConsoleAutoConfiguration    : H2 console available at '/demo1/h2'. Database available at 'jdbc:h2:mem:de
o1'
2025-11-11 19:37:57.288  INFO 1 --- [           main] o.hibernate.jpa.internal.util.LogHelper  : HHH000204: Processing PersistenceUnitInfo [name: default]
2025-11-11 19:37:57.720  INFO 1 --- [           main] org.hibernate.Version                    : HHH000412: Hibernate ORM core version 5.6.9.Final
2025-11-11 19:37:58.836  INFO 1 --- [           main] o.hibernate.annotations.common.Version   : HCANN000001: Hibernate Commons Annotations {5.1.2.Final}
2025-11-11 19:37:59.824  INFO 1 --- [           main] org.hibernate.dialect.Dialect            : HHH000400: Using dialect: org.hibernate.dialect.H2Dialect
2025-11-11 19:38:03.287  INFO 1 --- [           main] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transac
ion.jta.platform.internal.NoJtaPlatform]
2025-11-11 19:38:03.334  INFO 1 --- [           main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2025-11-11 19:38:05.175  WARN 1 --- [           main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries
may be performed during view rendering. Explicitly configure spring.jpa.open-in-view to disable this warning
2025-11-11 19:38:07.600  INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with context path ''
2025-11-11 19:38:07.617  INFO 1 --- [           main] com.example.demo1.Demo1Application       : Started Demo1Application in 32.063 seconds (JVM running for 48.238)
deepa@deepa-lin:~/K8s$
```

logs show that your **Spring Boot application** (Demo1Application) has started successfully inside the pod deepadeploy-8495f7f86b-dxgm9:

- Tomcat is running on port **8080**.
- H2 database is initialized and accessible at /demo1/h2.
- JPA and Hibernate have started correctly.
- Application startup completed without critical errors.



Pod running 1/3




```
deepa@deepa-lin:~/K8s$ kubectl get all -n deepa-app
NAME                                    READY   STATUS    RESTARTS      AGE
pod/deepadeploy-8495f7f86b-dxgm9        1/1     Running   2 (13h ago)   20h
pod/deepadeploy-8495f7f86b-kln77        1/1     Running   2 (13h ago)   20h
pod/deepadeploy-8495f7f86b-pqcrj        1/1     Running   2 (13h ago)   20h
```