1. Setup ARGO CD

Login into your k8s cluster and install ArgoCD

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

```
[root@master ~]# kubectl create namespace argocd
namespace/argocd created
[root@master ~]# kubectl apply -n argocd \
-f https://raw.githubusercontent.com/argoproj/argo-cd/stable/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
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
```

- **kubectl get pods -n argocd**

```
[root@master ~]# kubectl get pods -n argocd
NAME                                                READY   STATUS    RESTARTS   AGE
argocd-application-controller-0                     1/1     Running   0          3m12s
argocd-applicationset-controller-6dd89769b9-cq8sc   1/1     Running   0          3m12s
argocd-dex-server-5d887d57b9-jq9tk                  1/1     Running   0          3m12s
argocd-notifications-controller-f9847886b-pm7c4     1/1     Running   0          3m12s
argocd-redis-7c76ccfd48-dxzmd                       1/1     Running   0          3m12s
argocd-repo-server-5bf876bdfb-jrkkp                 1/1     Running   0          3m12s
argocd-server-69d5b9f8d5-6rm2d                      1/1     Running   0          3m12s
[root@master ~]#
```

- **kubectl port-forward svc/argocd-server -n argocd 8080:443**
- **kubectl patch svc argocd-server -n argocd **
  -p '{"spec": {"type": "NodePort"}}'

```
[root@master ~]# kubectl port-forward svc/argocd-server -n argocd 8080:443
Forwarding from 127.0.0.1:8080 -> 8080
Forwarding from [::1]:8080 -> 8080
kubectl get svc -n argocd
^C[root@master ~]kubectl get svc -n argocdcd
NAME                                      TYPE        CLUSTER-IP       EXTERNAL-IP
argocd-applicationset-controller          ClusterIP   10.106.172.215   <none>
argocd-dex-server                         ClusterIP   10.103.109.224   <none>
argocd-metrics                            ClusterIP   10.101.158.47    <none>
argocd-notifications-controller-metrics   ClusterIP   10.102.124.48    <none>
argocd-redis                              ClusterIP   10.101.83.232    <none>
argocd-repo-server                        ClusterIP   10.107.147.159   <none>
argocd-server                             ClusterIP   10.97.232.66     <none>
argocd-server-metrics                     ClusterIP   10.107.153.5     <none>
[root@master ~]# kubectl patch svc argocd-server -n argocd \
-p '{"spec": {"type": "NodePort"}}'
service/argocd-server patched
```

- **kubectl get svc argocd-server -n argocd**

```
[root@master ~]# kubectl get svc argocd-server -n argocd
NAME            TYPE       CLUSTER-IP     EXTERNAL-IP   PORT(S)                      AGE
argocd-server   NodePort   10.97.232.66   <none>        80:30722/TCP,443:32003/TCP   11m
```

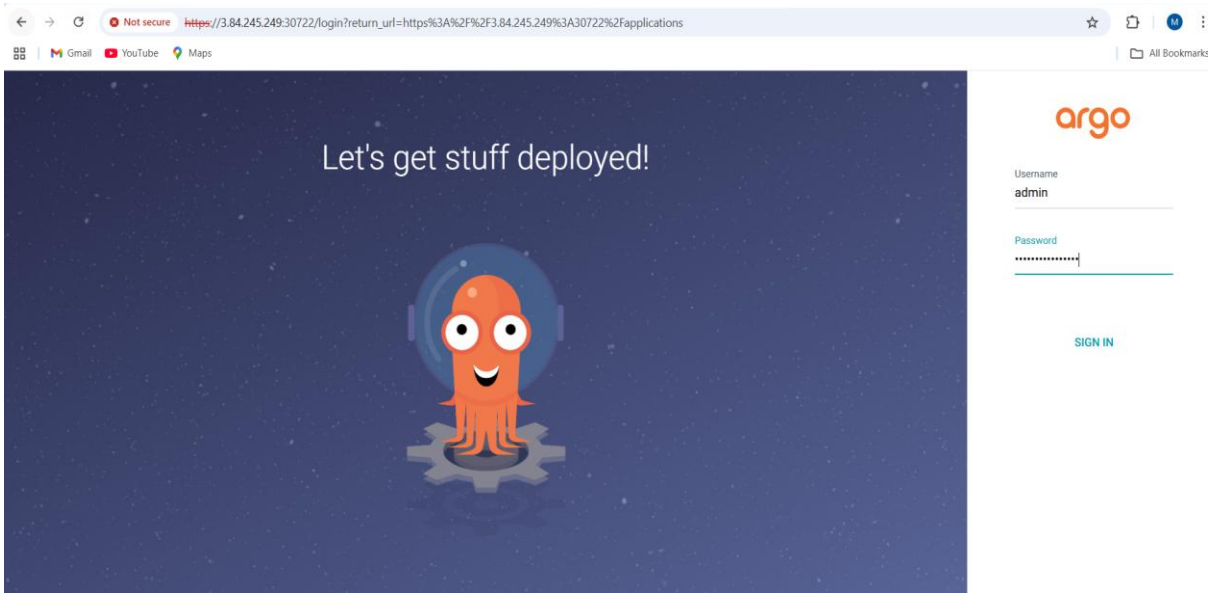
Access in browser with the ip and port number:30722

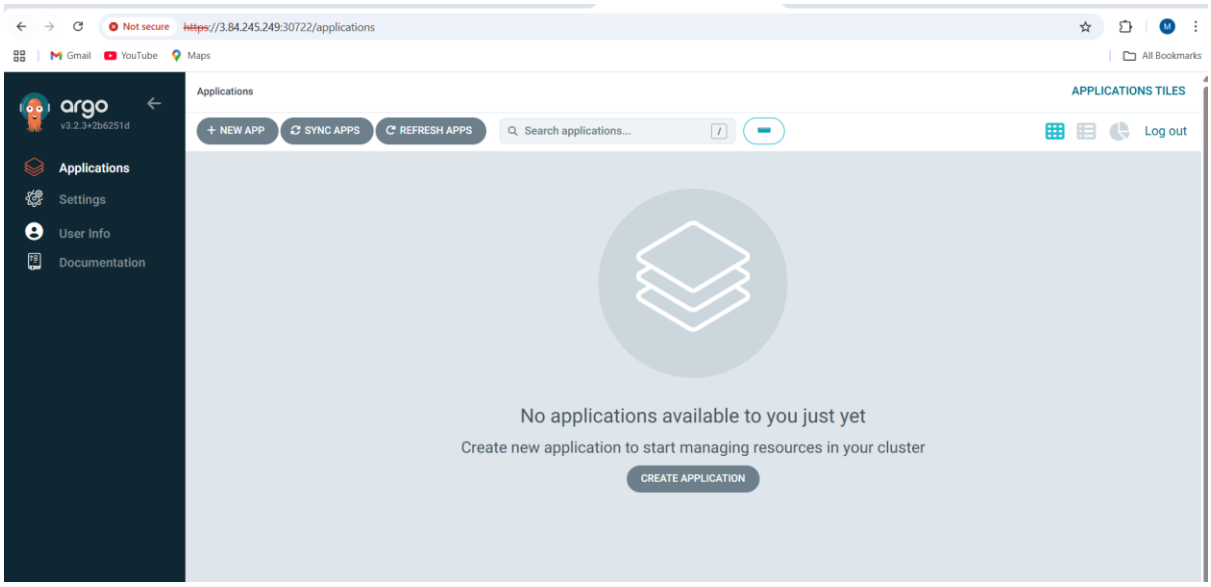
*** kubectl get secret argocd-initial-admin-secret -n argocd **

-o jsonpath="{.data.password}" | base64 –decode

```
[root@master ~]# kubectl get secret argocd-initial-admin-secret -n argocd \
-o jsonpath="{.data.password}" | base64 --decode
9VgucHD3U7-Xk4Az[root@master ~]#
```

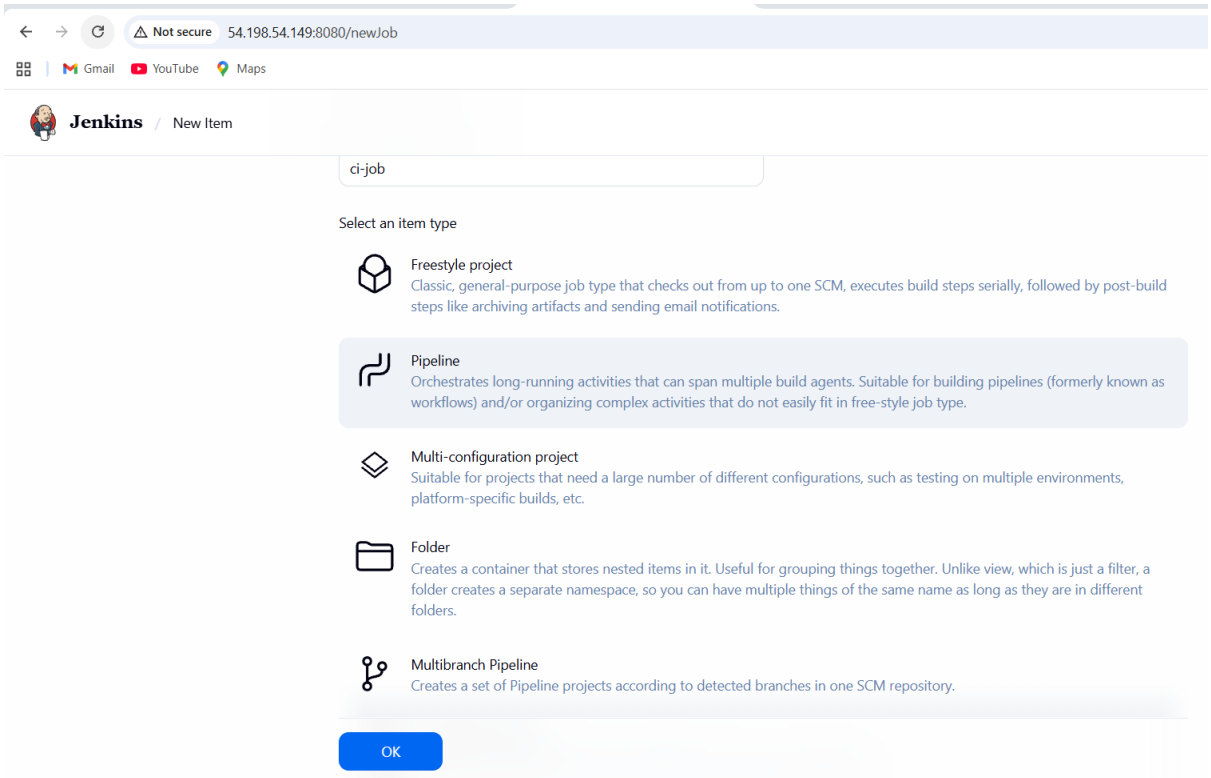
Username:admin password given in the above command we can access the argocd



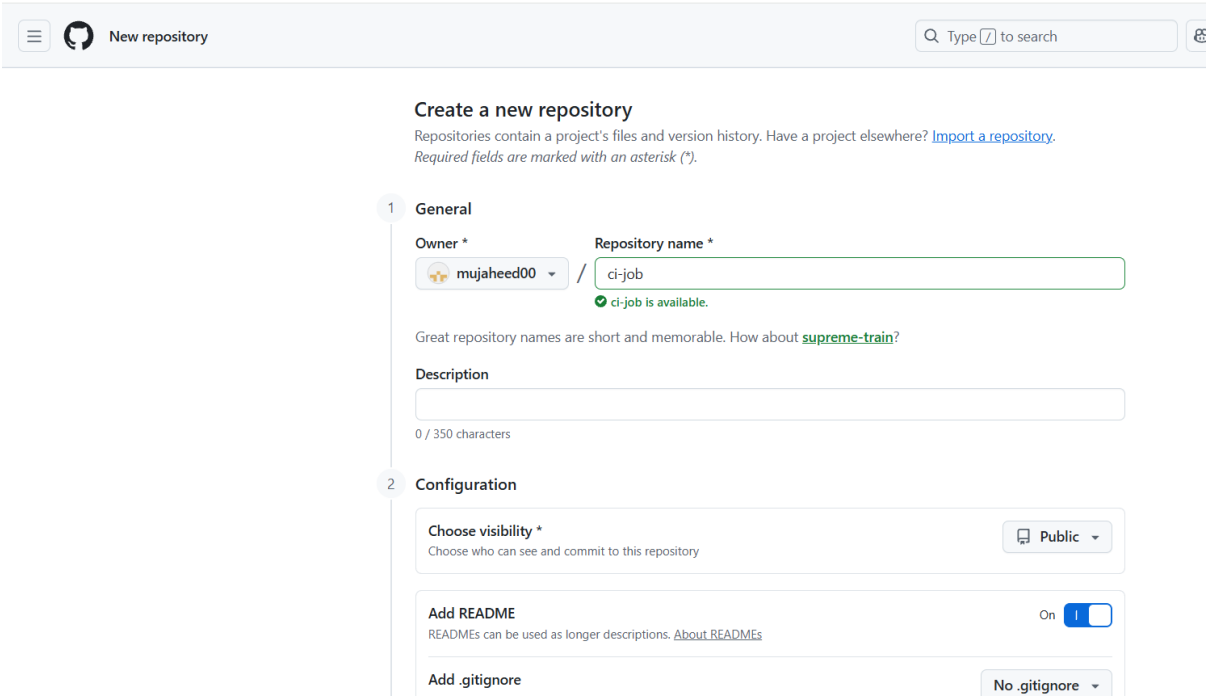


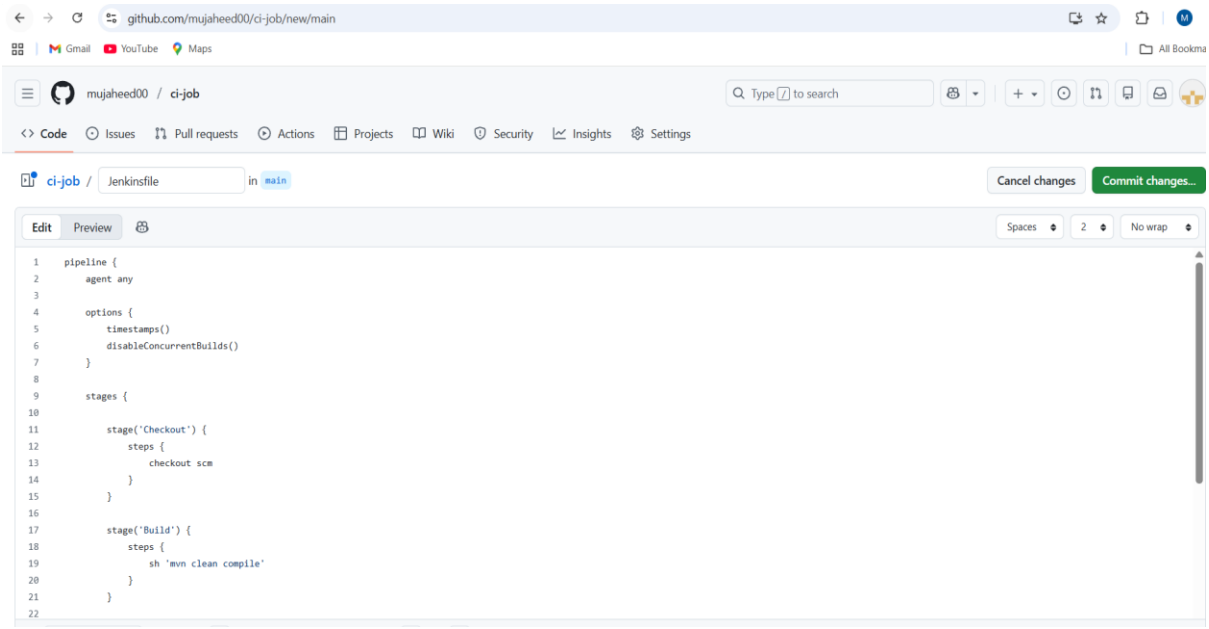
2. Create Jenkins job for CI.

Login into Jenkins server and build a new job with the name ci-job and select type as pipeline.

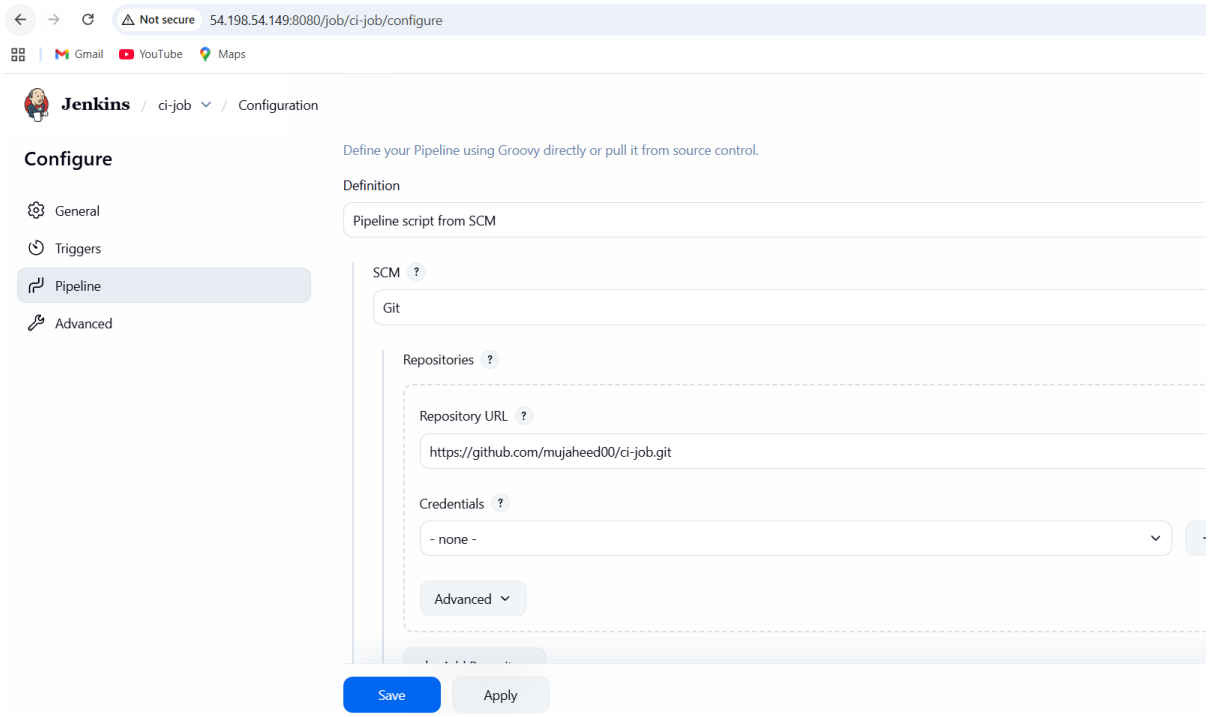


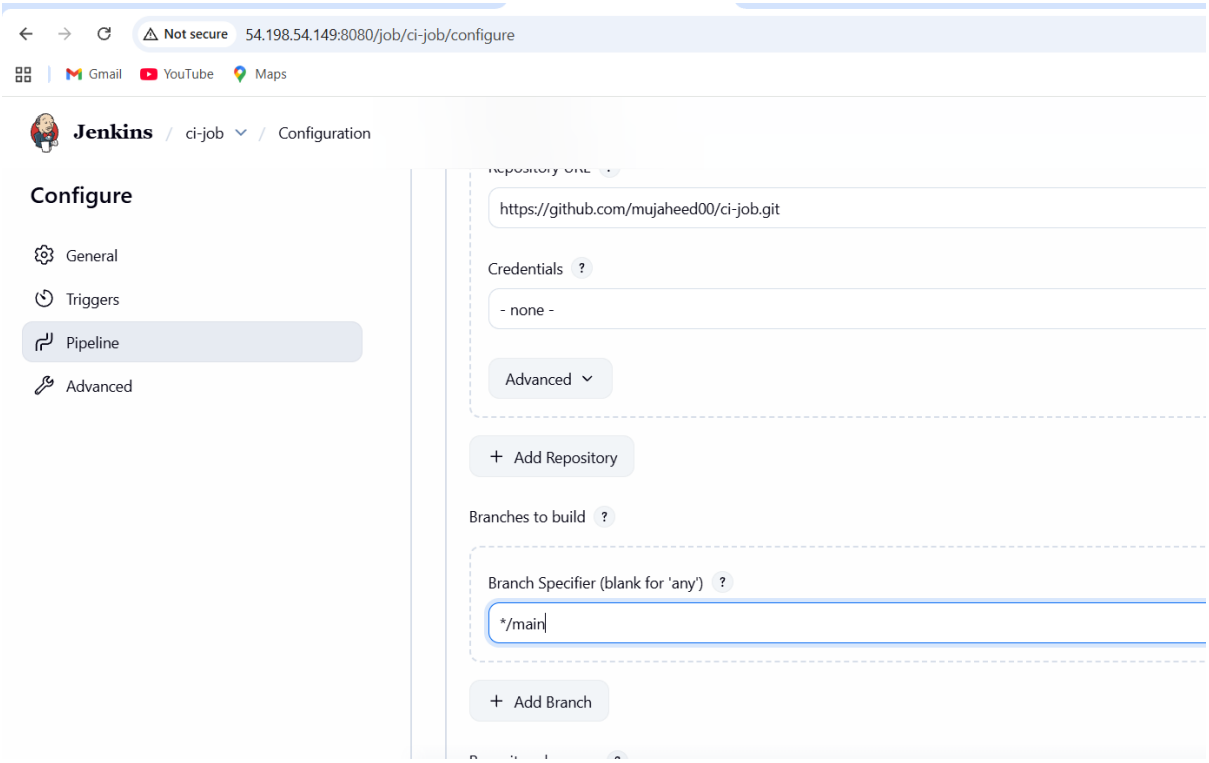
Create repository in github and add Jenkins file.



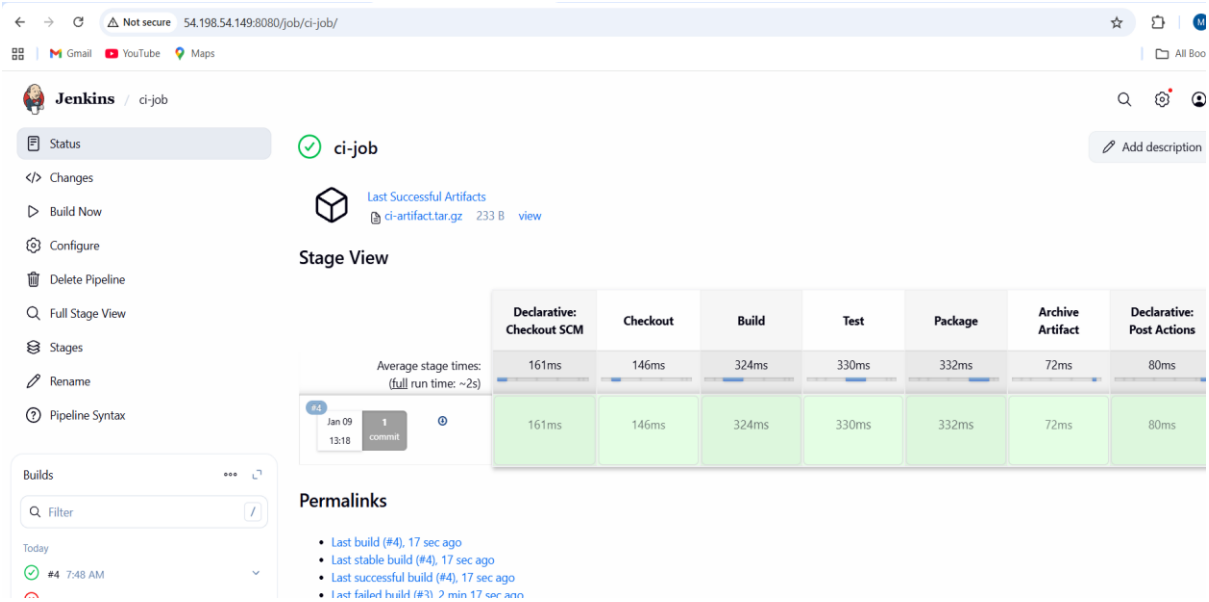


Select git scm and give your repository





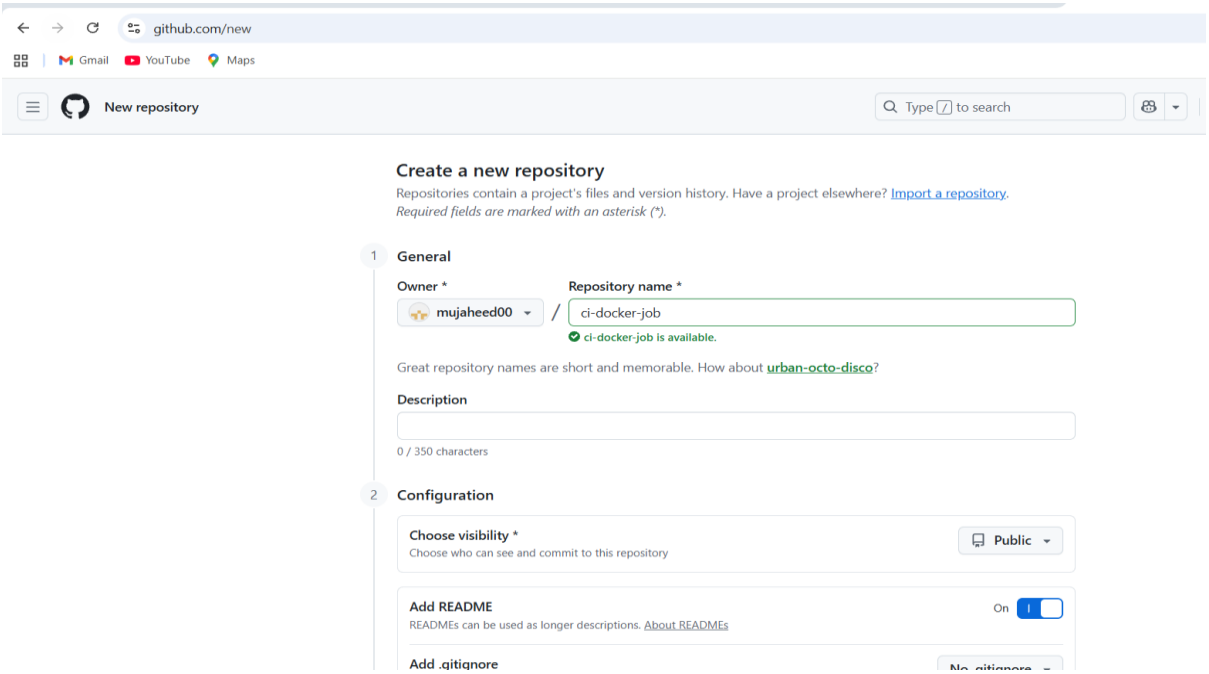
Click on build now



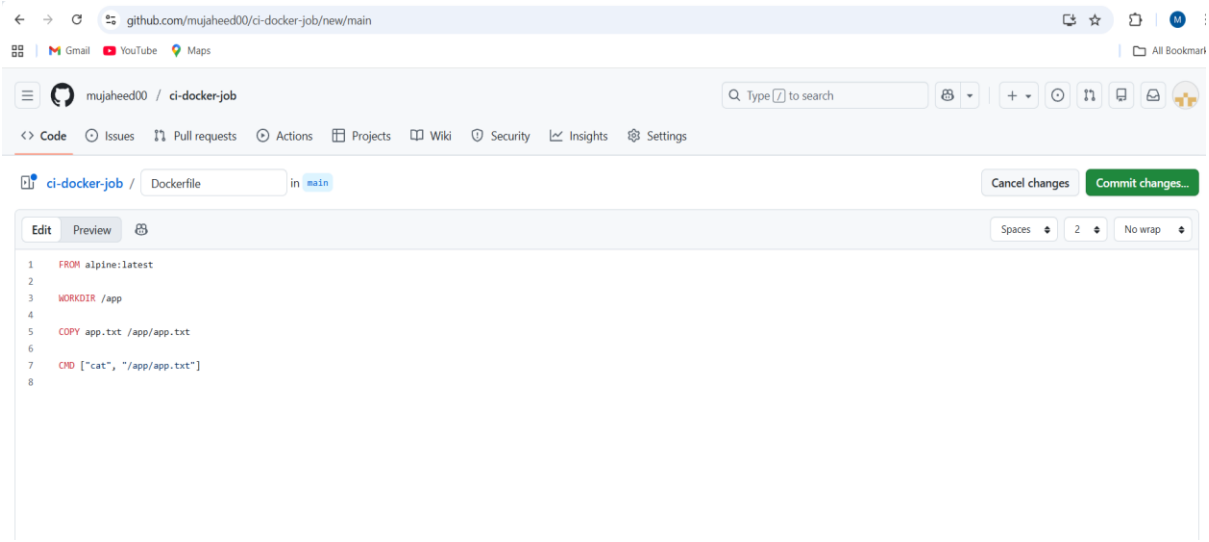
3. Create Jenkins job to create docker image and build image.

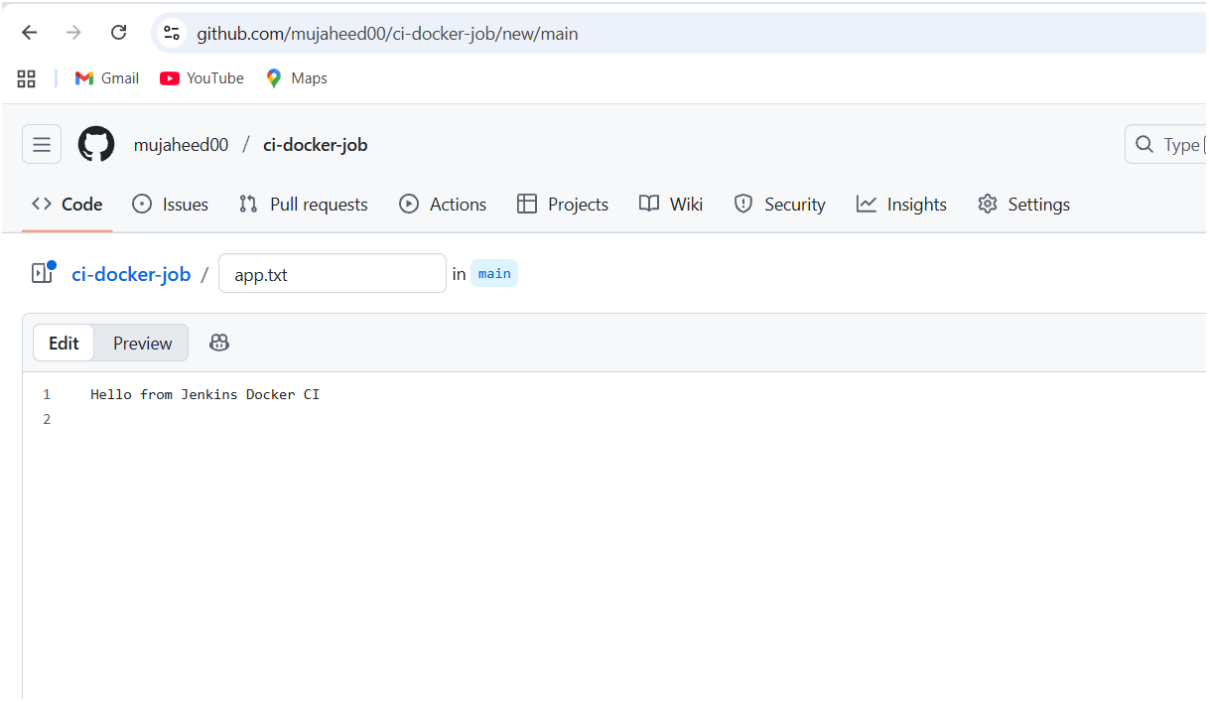
Make sure that your Jenkins server will have docker service up and running.

Create a new repository in github with the name ci-docker-job

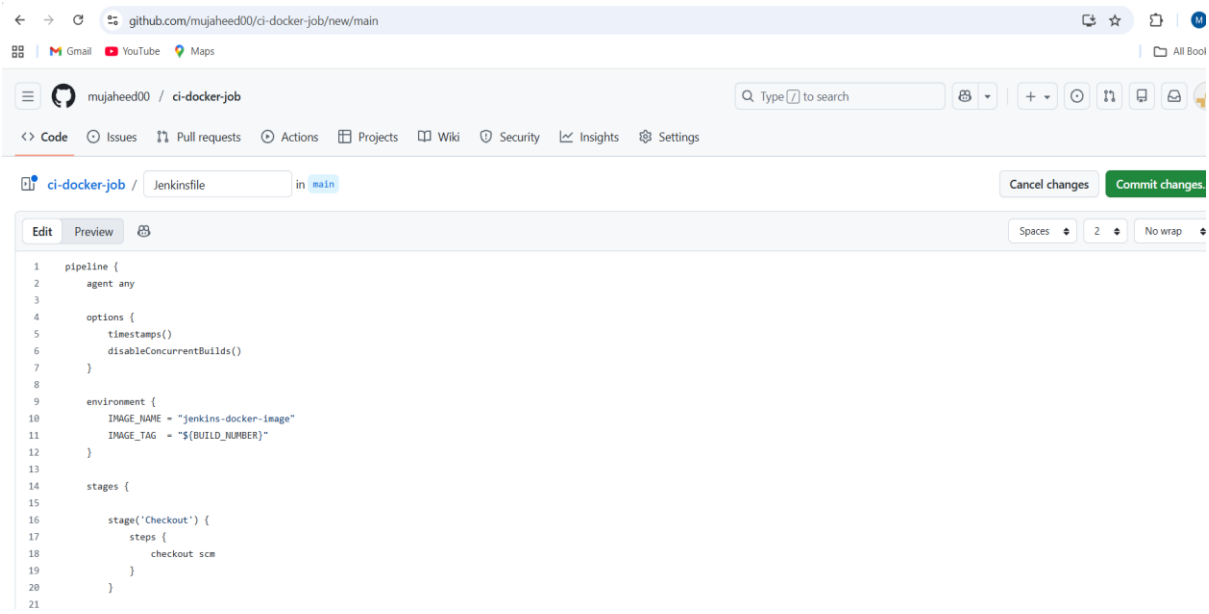


Create a Dockerfile in that repository.

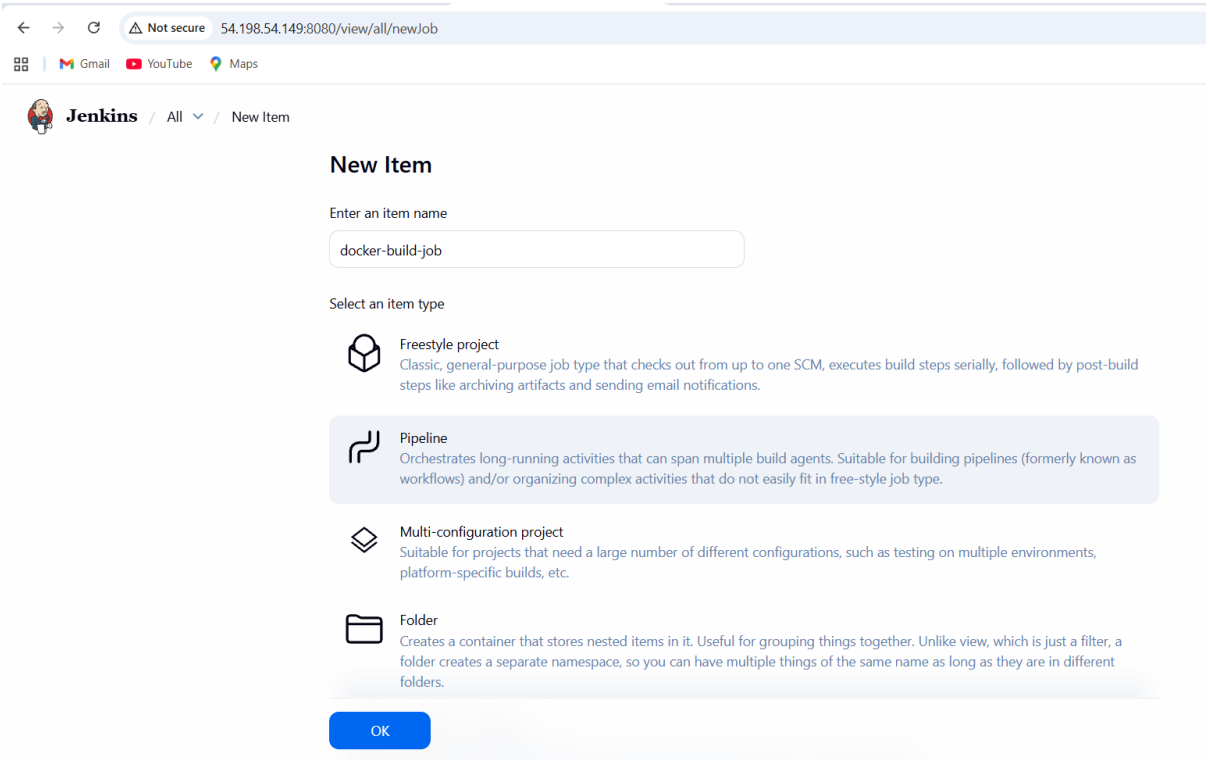




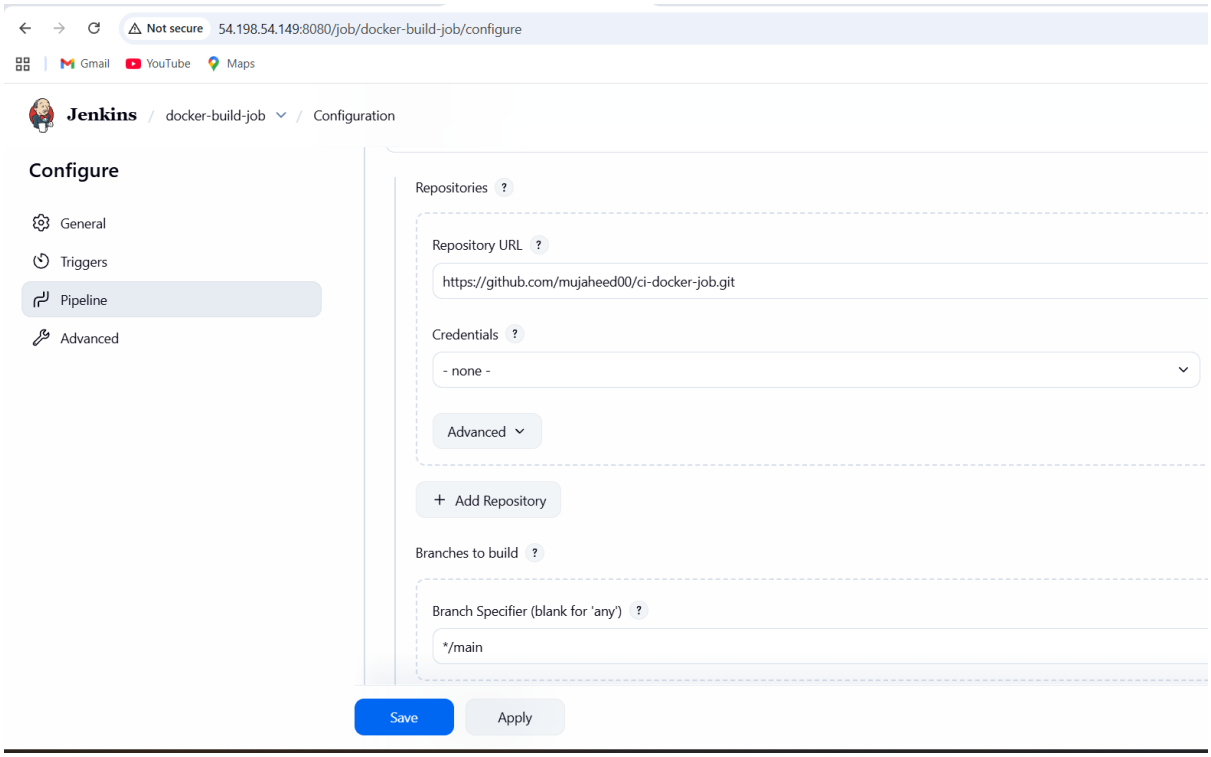
Add another file Jenkinsfile



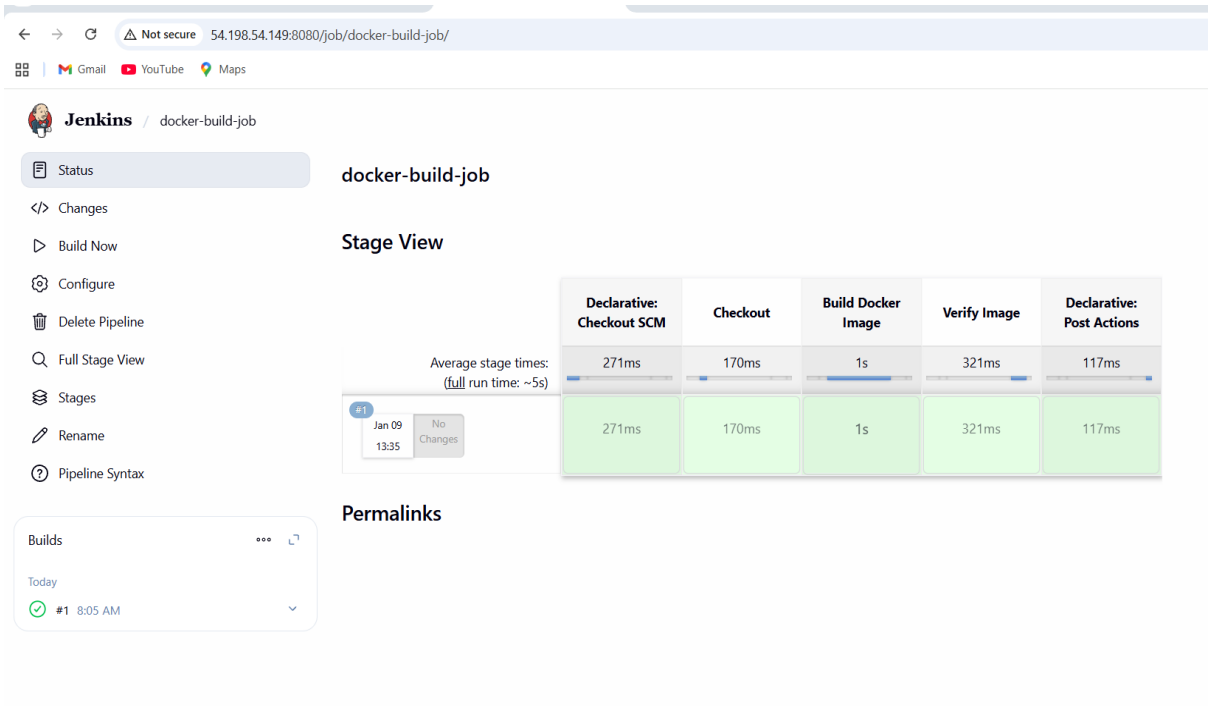
Build a new job with the name docker-build-job and type as pipeline



Give the repo which you have jenkinsfile and dockerfile and app.txt give that in git scm

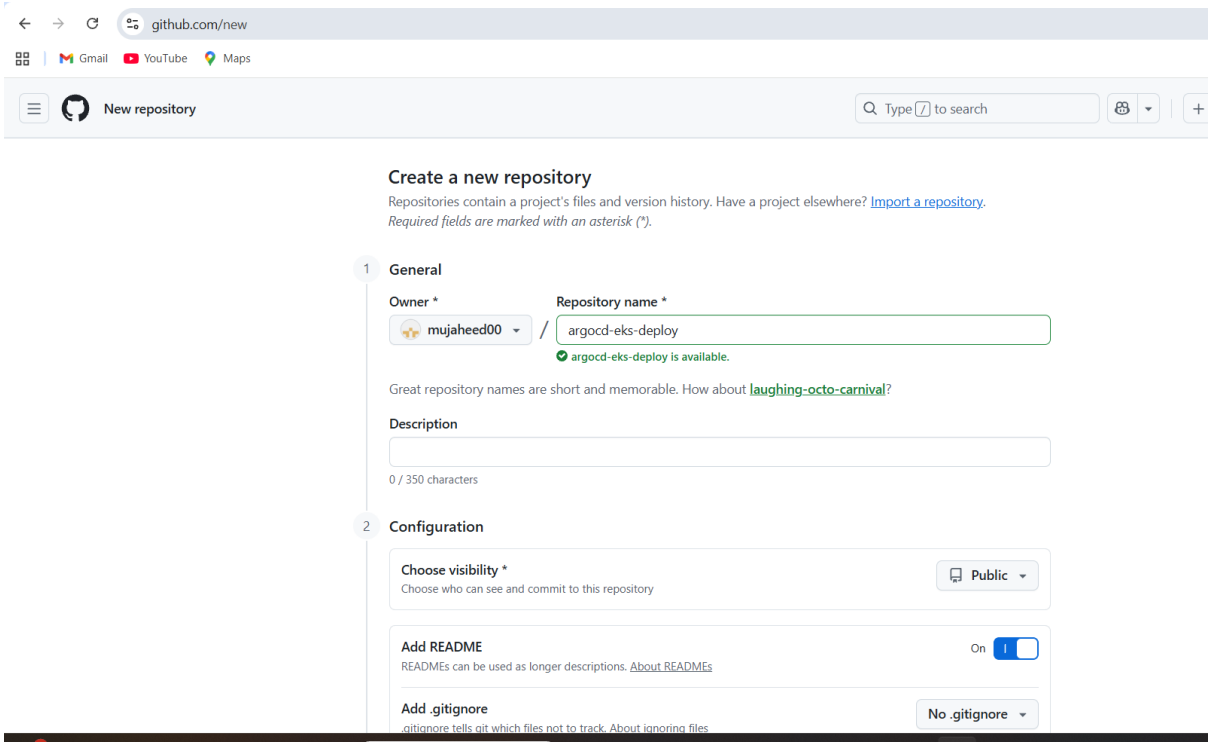


Click on build now.



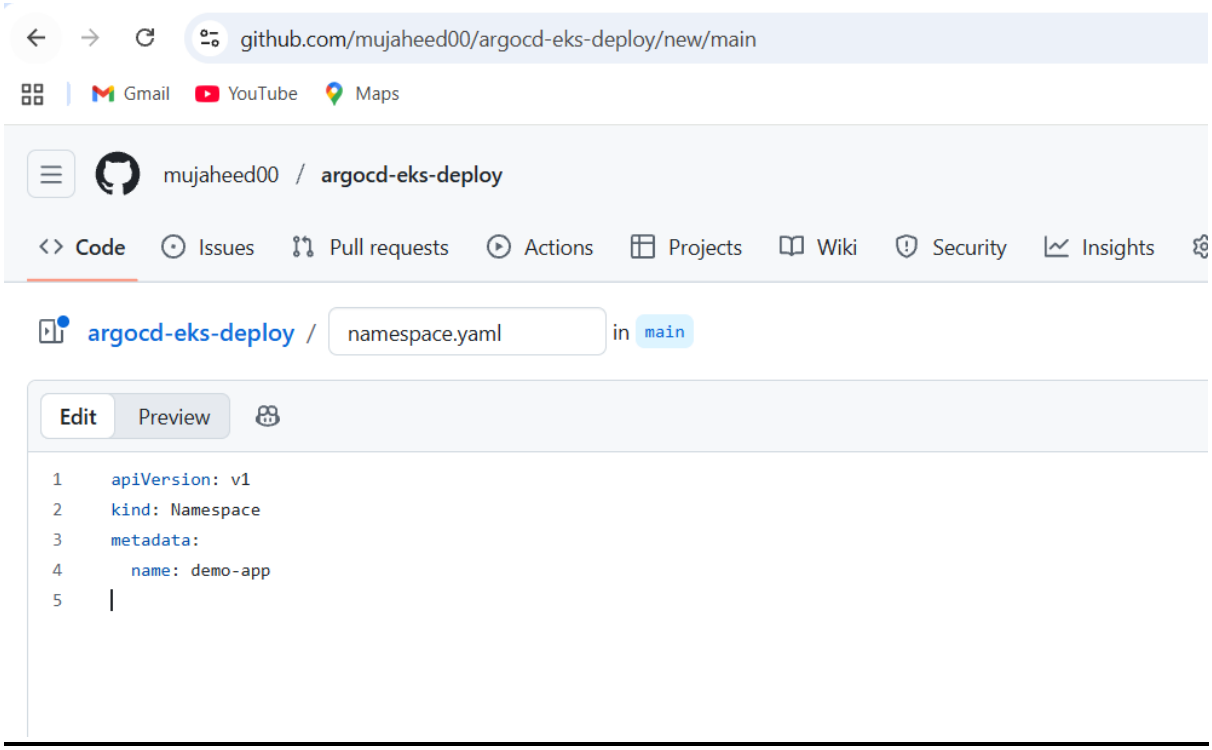
4. Create ArgoCD job to deploy on ek8s cluster.

Go to github and click on new repository and name it as argocd-eks-deploy

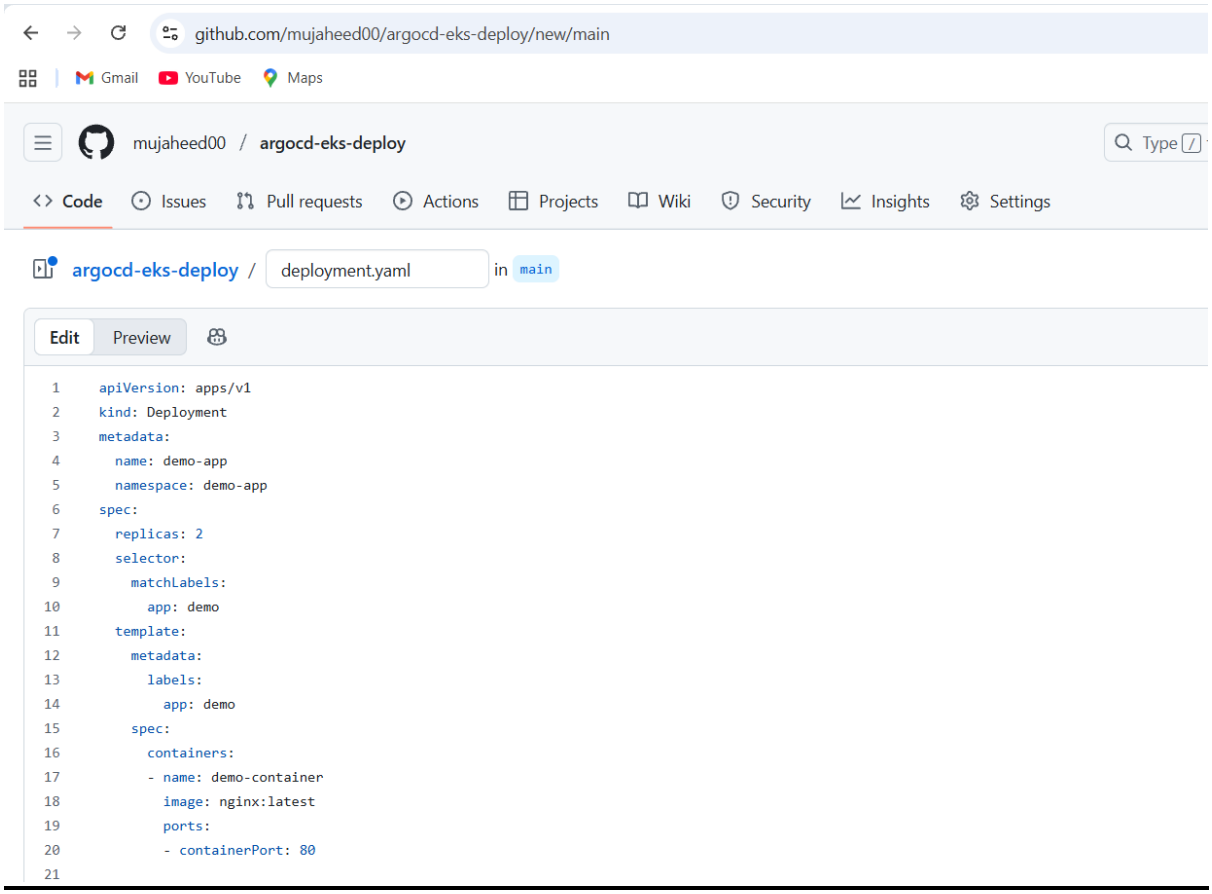


In that repository add manifest files.

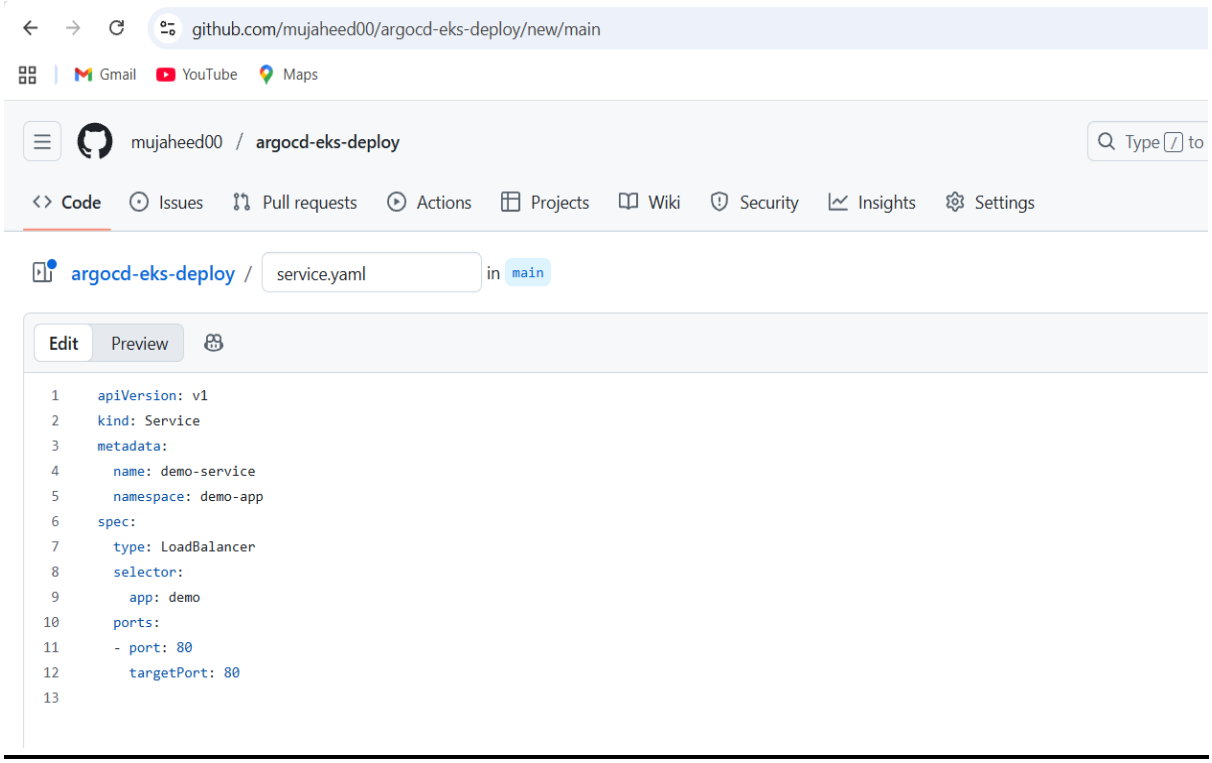
Namespace.yml:



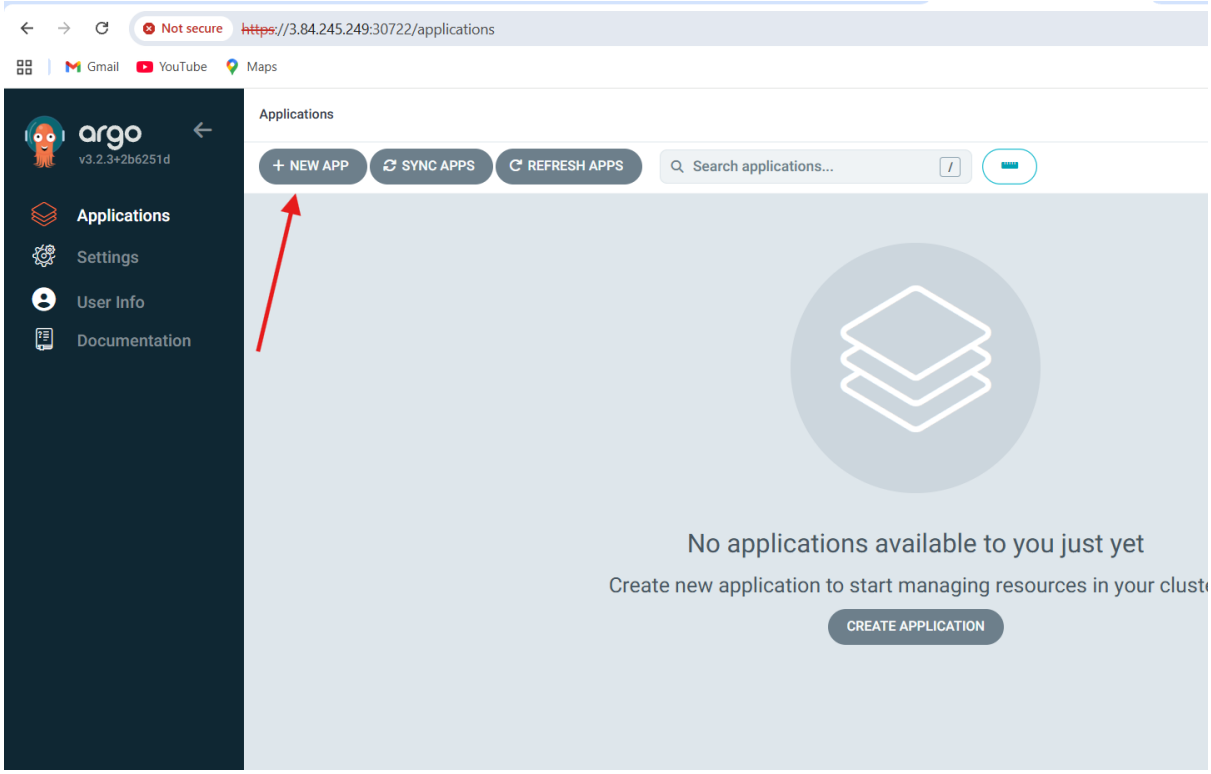
deployment.yaml

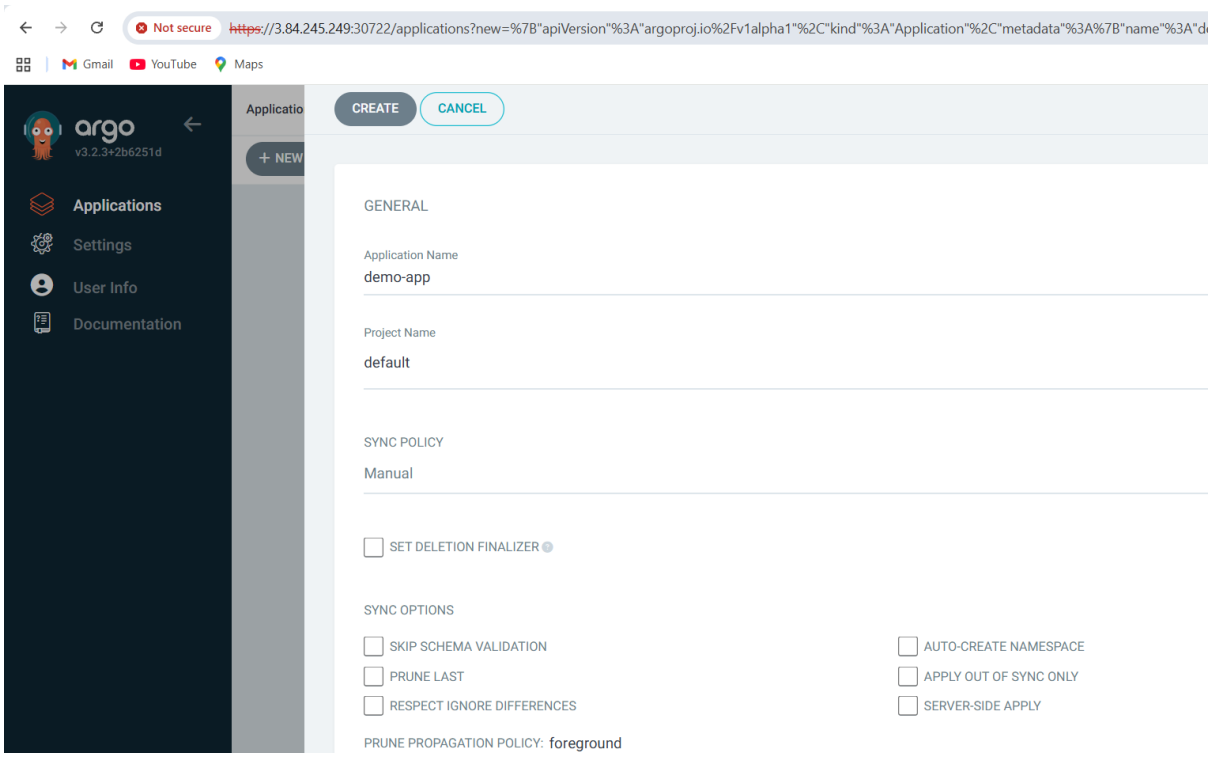


Service.yaml



Go to argocd GUI and click on new app

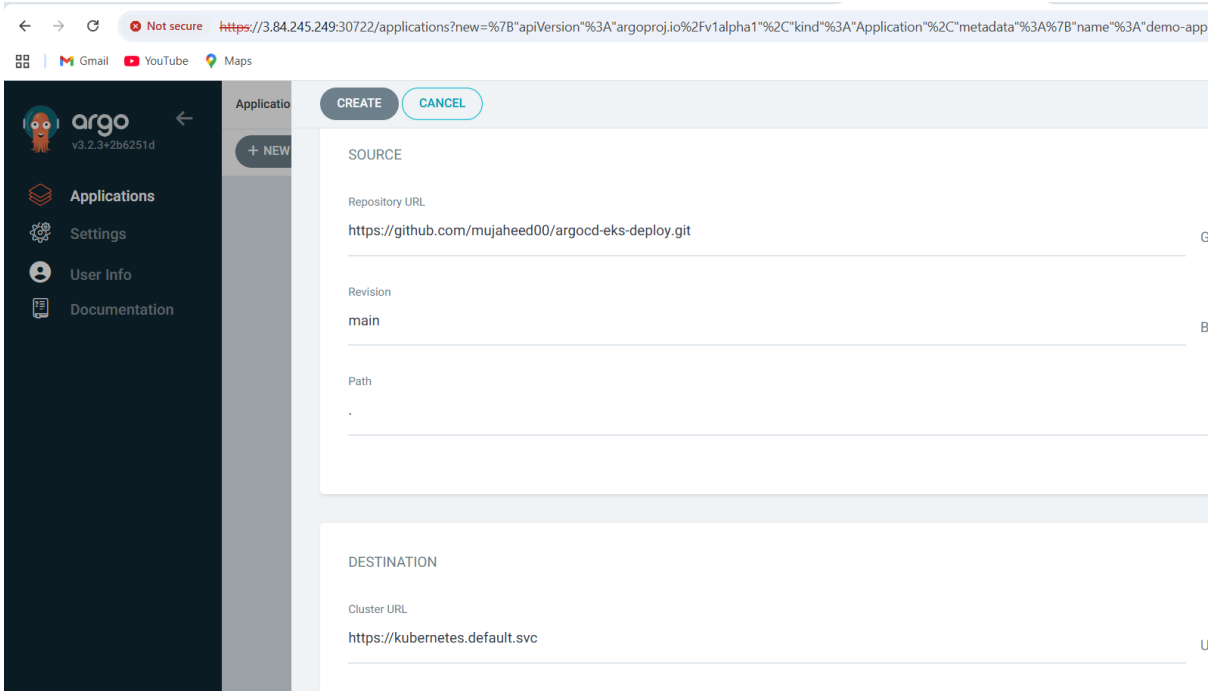


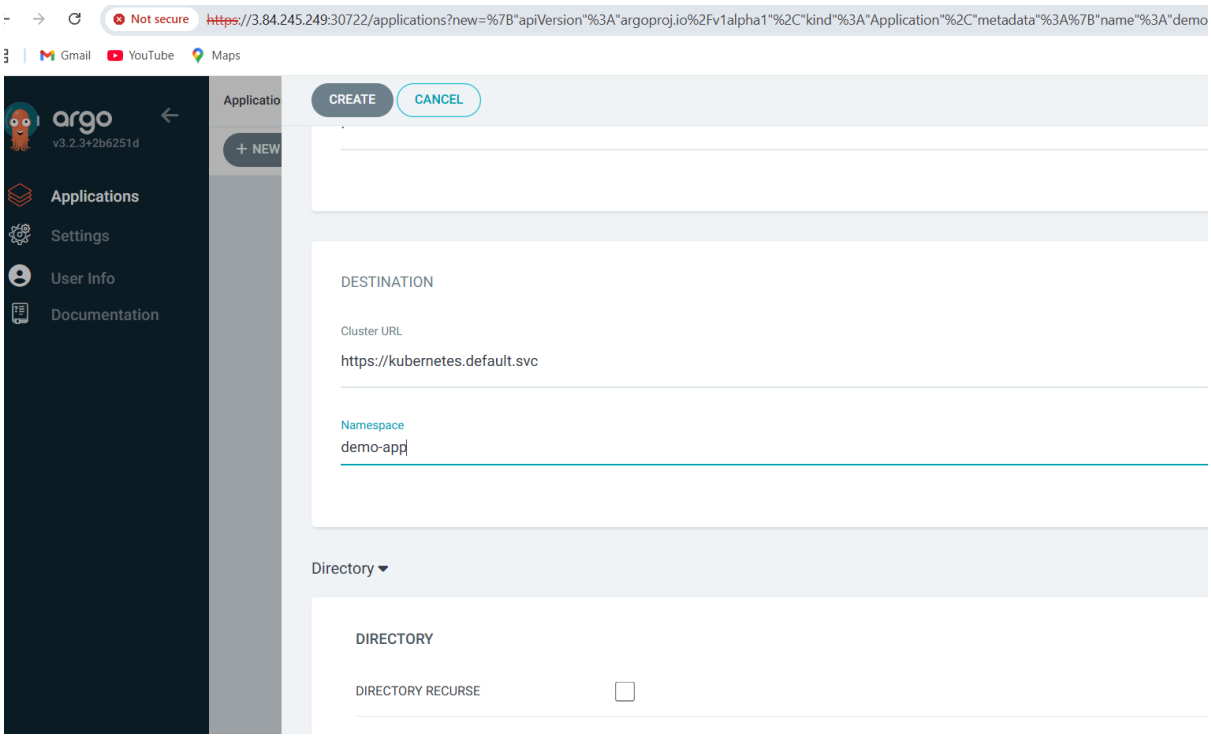


Source: which you had created for manifest files

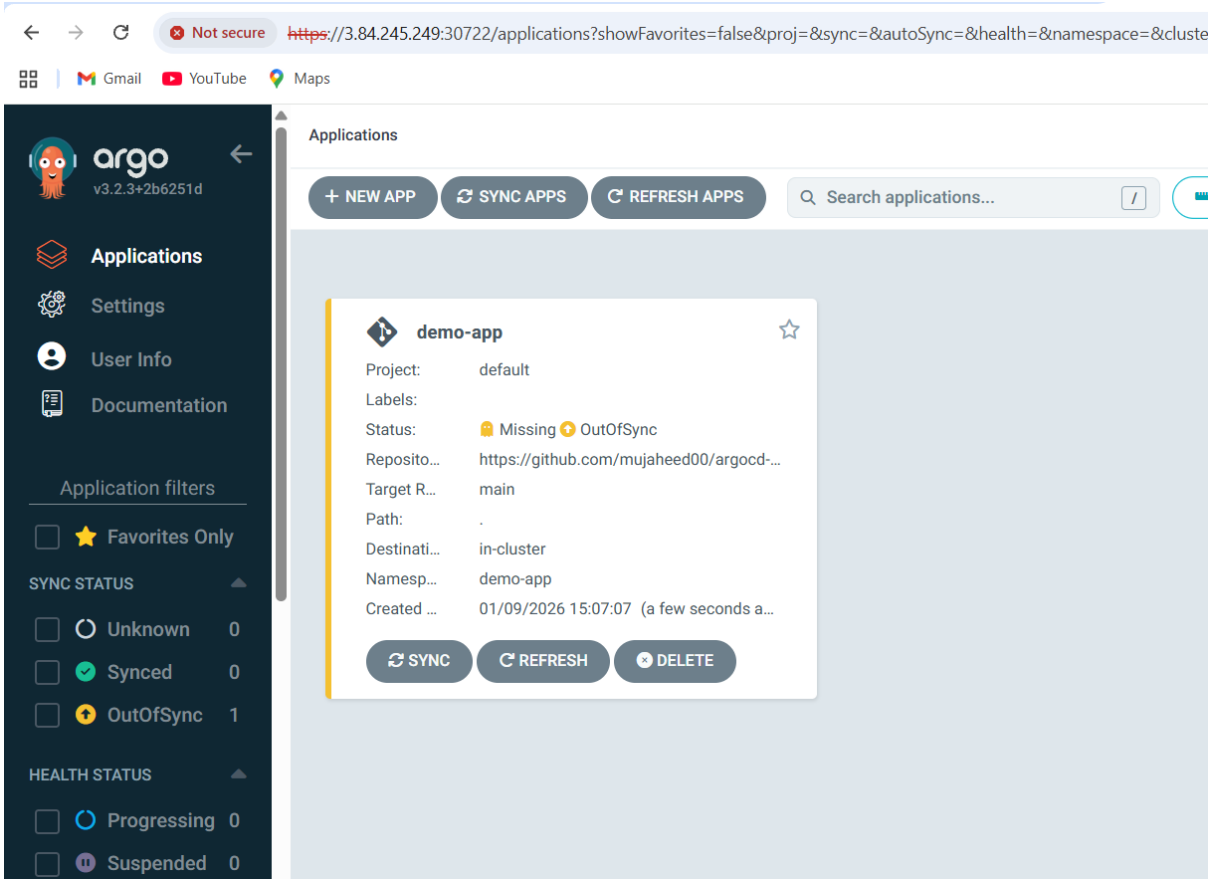
Branch: main

Path: .

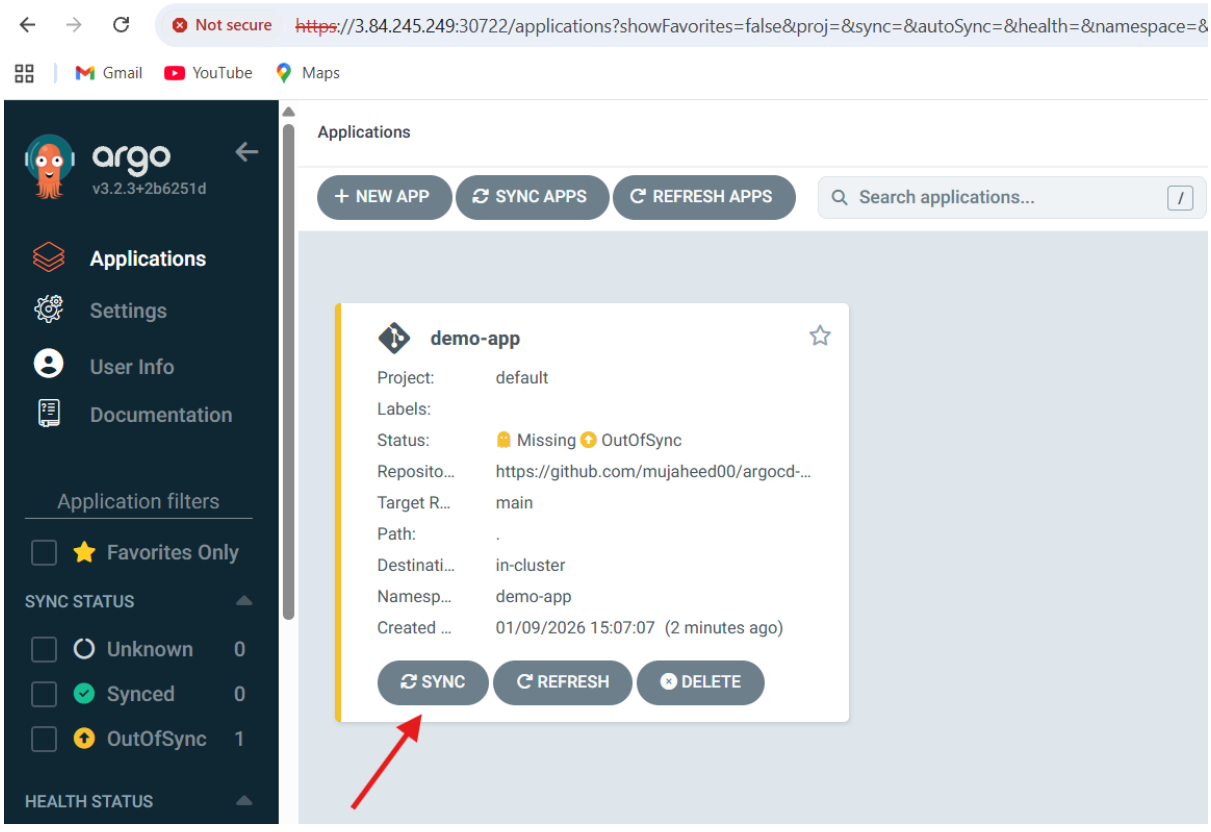


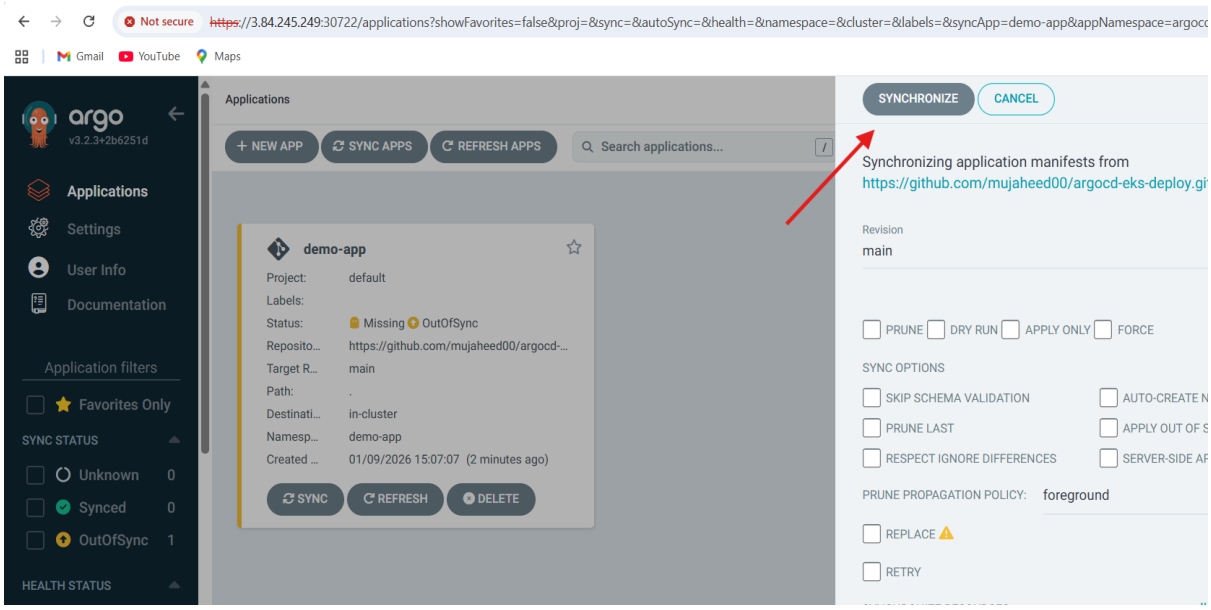


Click on create



Click on sync





Expose this demo-app in your cluster

- **kubectl get svc -n demo-app**

```
[root@master ~]# kubectl get svc -n demo-app
NAME           TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
demo-service   LoadBalancer   10.98.39.140    <pending>     80:30199/TCP   70s
[root@master ~]#
```

Access in your browser with your ip and port 30199