

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
clusterrole.rbac.authorization.k8s.io/argocd-applicationset-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 argocd
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**

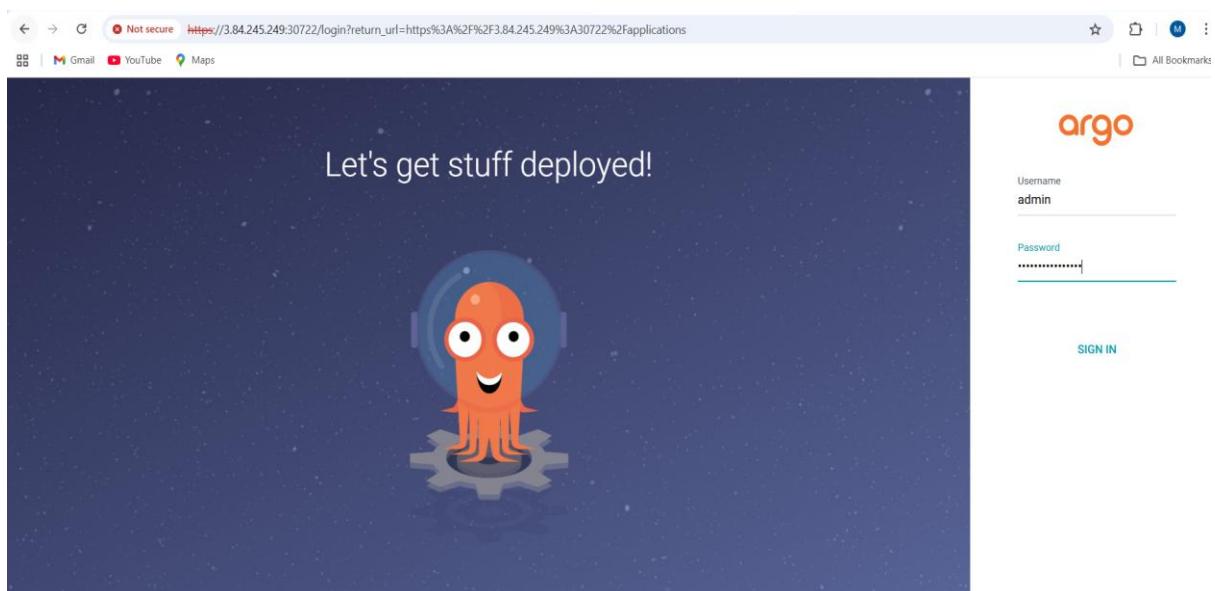
```
[root@master ~]# kubectl get svc argocd-server -n argocd
NAME      TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
argocd-server   NodePort    10.97.232.66 <none>        80:30722/TCP,443:32003/TCP   11m
```

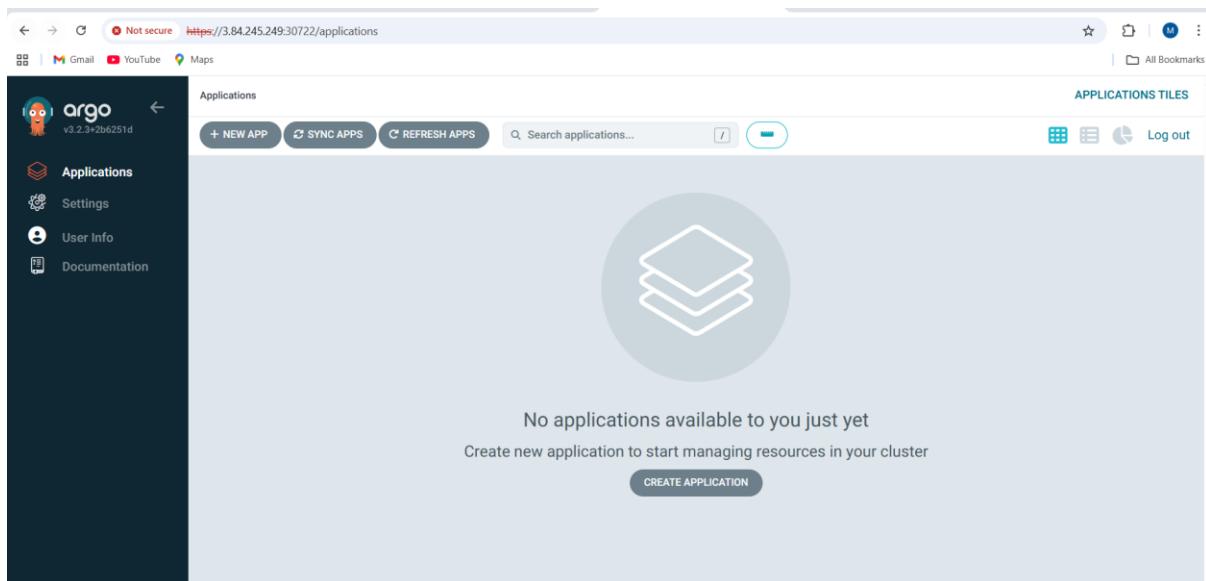
Access in browser with the ip and port number:30722

* **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.

A screenshot of a web browser showing the Jenkins 'New Item' creation dialog at the URL 54.198.54.149:8080/newJob. The title bar says 'Jenkins / New Item'. A search bar contains the text 'ci-job'. Below it, a section titled 'Select an item type' lists five options: 'Freestyle project', 'Pipeline', 'Multi-configuration project', 'Folder', and 'Multibranch Pipeline'. The 'Pipeline' option is highlighted with a light gray background and a blue border around its icon. A tooltip for 'Pipeline' describes it as 'Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.' At the bottom of the list is a blue 'OK' button.

Create repository in github and add Jenkins file.

 New repository

Type to search

Create a new repository

Repositories contain a project's files and version history. Have a project elsewhere? [Import a repository](#). Required fields are marked with an asterisk (*).

1 General

Owner *  **mujaheed00** / Repository name *  ci-job is available.

Great repository names are short and memorable. How about [supreme-train](#)?

Description
0 / 350 characters

2 Configuration

Choose visibility *  Public

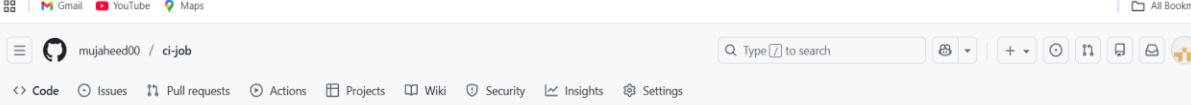
Choose who can see and commit to this repository

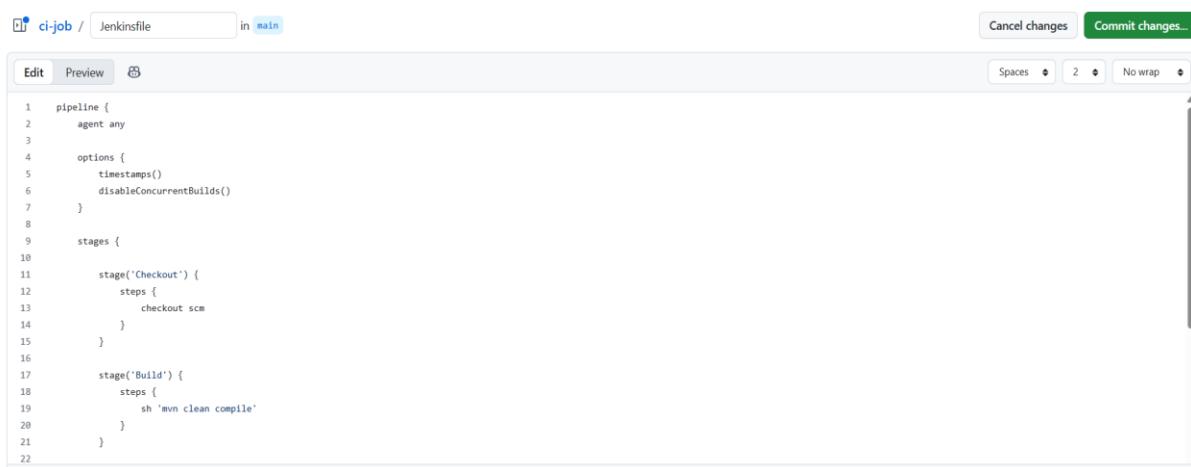
Add README On 

READMEs can be used as longer descriptions. [About READMEs](#)

Add .gitignore  No .gitignore

 github.com/mujaheed00/ci-job/new/main

 [Gmail](#) [YouTube](#) [Maps](#) [All Bookmarks](#)

 Jenkinsfile in main

Edit Preview

```

1 pipeline {
2     agent any
3
4     options {
5         timestamps()
6         disableConcurrentBuilds()
7     }
8
9     stages {
10         stage('Checkout') {
11             steps {
12                 checkout scm
13             }
14         }
15
16         stage('Build') {
17             steps {
18                 sh 'mvn clean compile'
19             }
20         }
21     }
22 }
```

Select git scm and give your repository

← → ⌂ Not secure 54.198.54.149:8080/job/ci-job/configure

Gmail YouTube Maps

 Jenkins / ci-job / Configuration

Configure

Definition

Pipeline script from SCM

SCM ?

Git

Repositories ?

Repository URL ?

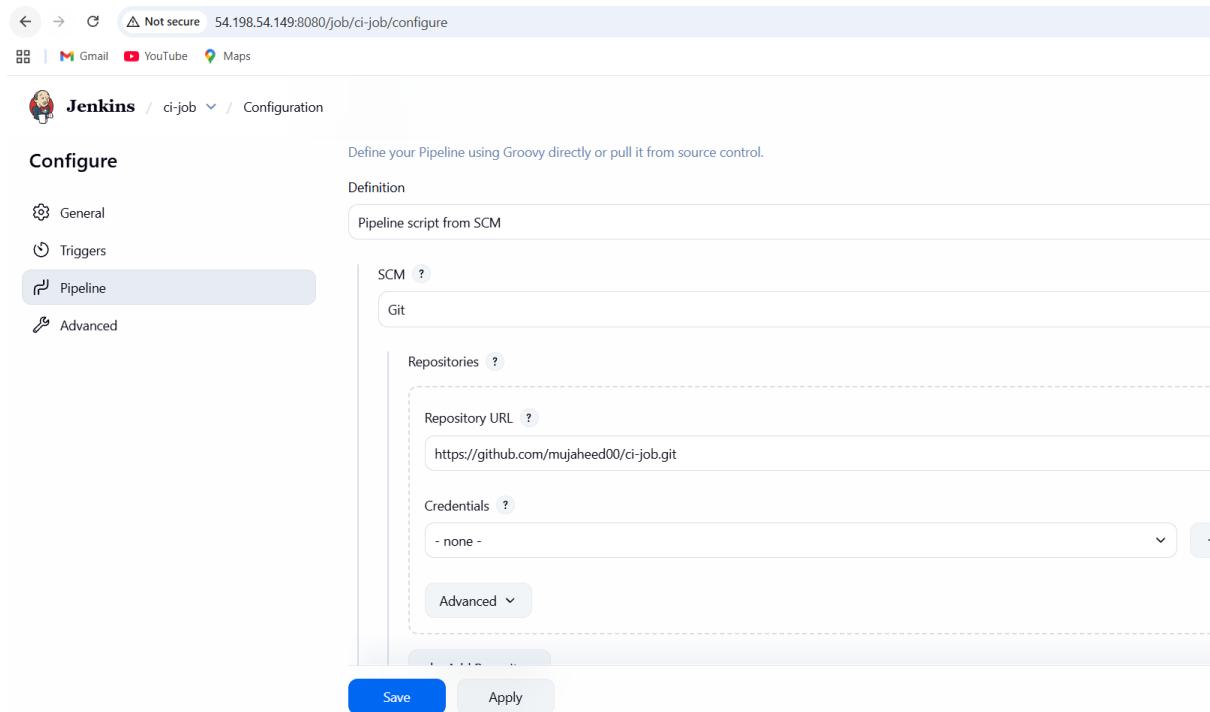
https://github.com/mujaheed00/ci-job.git

Credentials ?

- none -

Advanced ▾

Save Apply



← → ⌂ Not secure 54.198.54.149:8080/job/ci-job/configure

Gmail YouTube Maps

 Jenkins / ci-job / Configuration

Configure

General Triggers Pipeline Advanced

Repository URL

https://github.com/mujaheed00/ci-job.git

Credentials

- none -

Advanced ▾

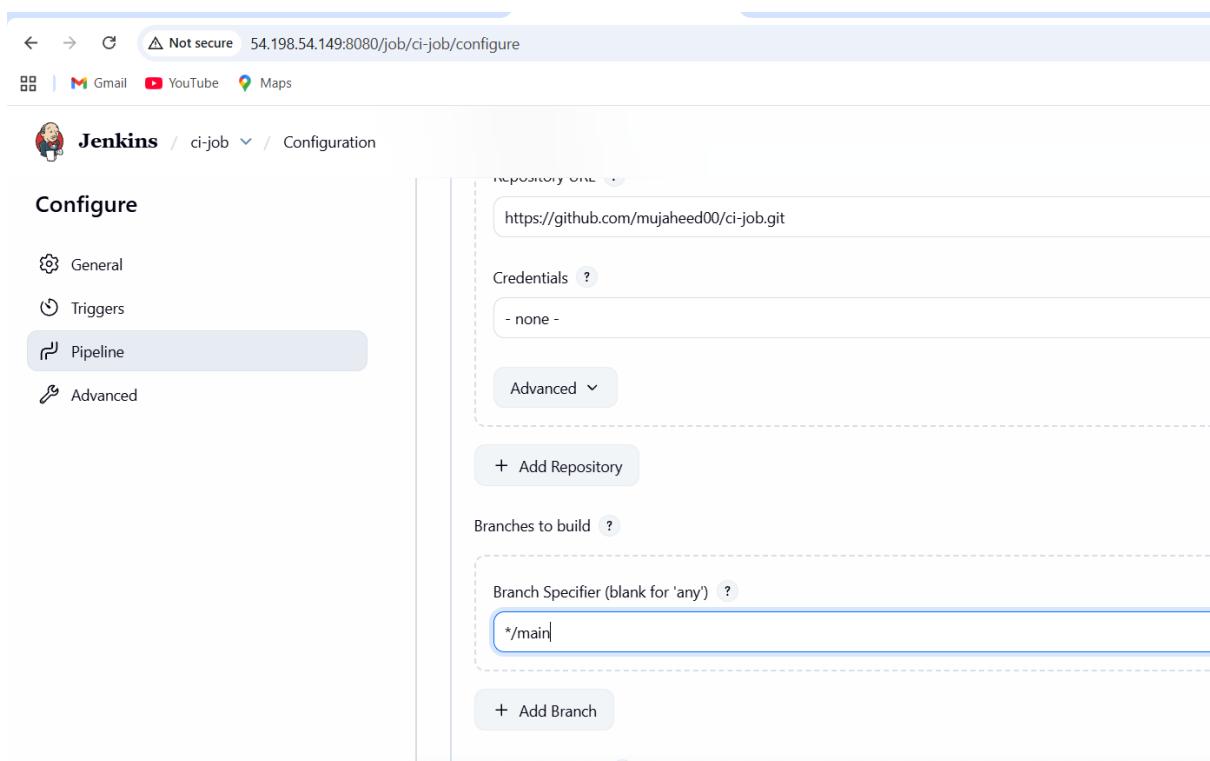
+ Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ?

*/main

+ Add Branch



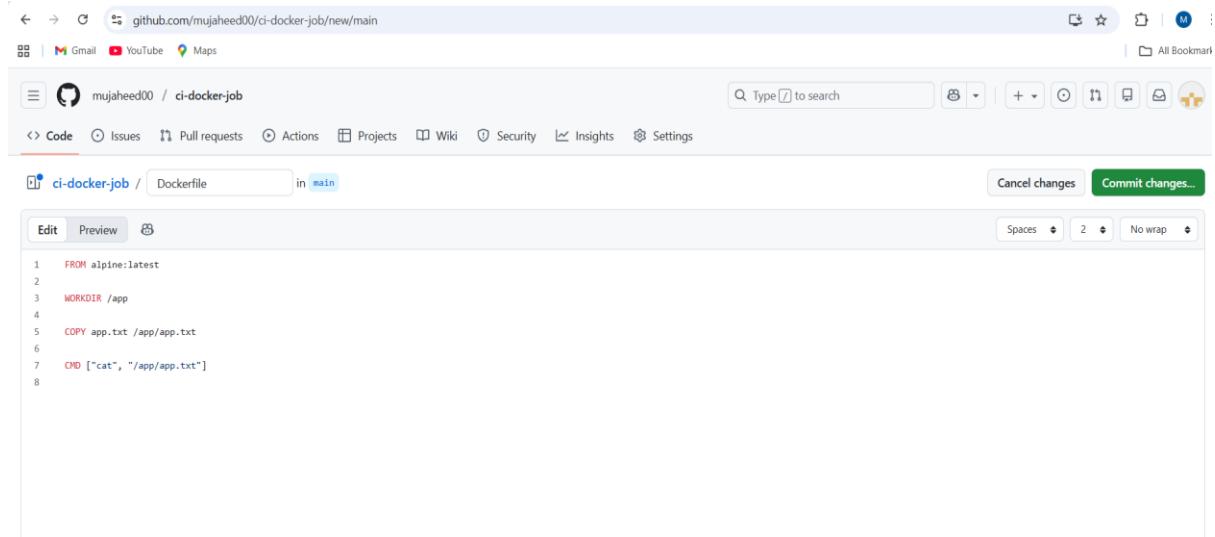
Click on build now

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

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

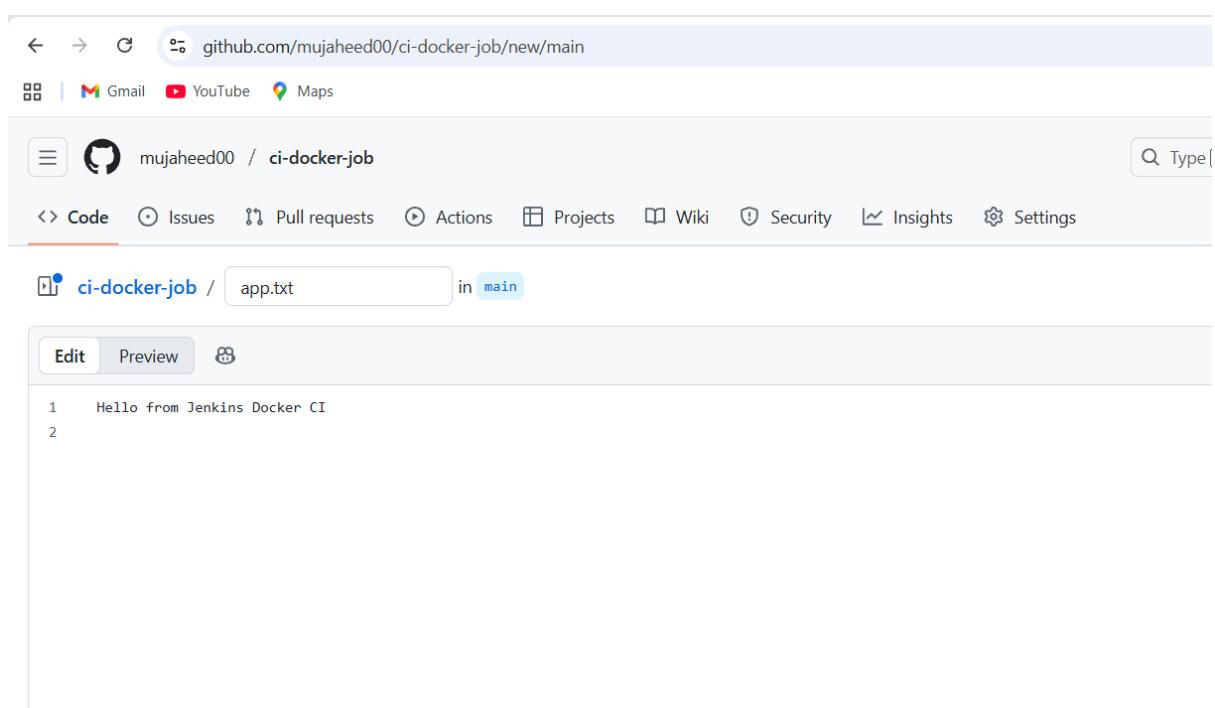
Create a new repository in github with the name ci-docker-job

Create a Dockerfile in that repository.



A screenshot of a GitHub code editor interface. The URL in the address bar is `github.com/mujaheed00/ci-docker-job/main`. The repository name is `mujaheed00 / ci-docker-job`. The file being edited is `Dockerfile`, located in the `main` branch. The code in the Dockerfile is:

```
1 FROM alpine:latest
2
3 WORKDIR /app
4
5 COPY app.txt /app/app.txt
6
7 CMD ["cat", "/app/app.txt"]
8
```



A screenshot of a GitHub code editor interface. The URL in the address bar is `github.com/mujaheed00/ci-docker-job/main`. The repository name is `mujaheed00 / ci-docker-job`. The file being edited is `app.txt`, located in the `main` branch. The code in the app.txt file is:

```
1 Hello from Jenkins Docker CI
2
```

Add another file Jenkinsfile

The screenshot shows a GitHub repository page for 'ci-docker-job'. The 'Code' tab is selected. A search bar at the top right contains the text 'Type / to search'. Below the search bar are several icons for repository management. The main area displays the Jenkinsfile content:

```

1 pipeline {
2     agent any
3
4     options {
5         timestamps()
6         disableConcurrentBuilds()
7     }
8
9     environment {
10        IMAGE_NAME = "jenkins-docker-image"
11        IMAGE_TAG = "${BUILD_NUMBER}"
12    }
13
14    stages {
15        stage('Checkout') {
16            steps {
17                checkout scm
18            }
19        }
20    }
21

```

At the bottom right of the code editor, there are buttons for 'Cancel changes' and 'Commit changes'.

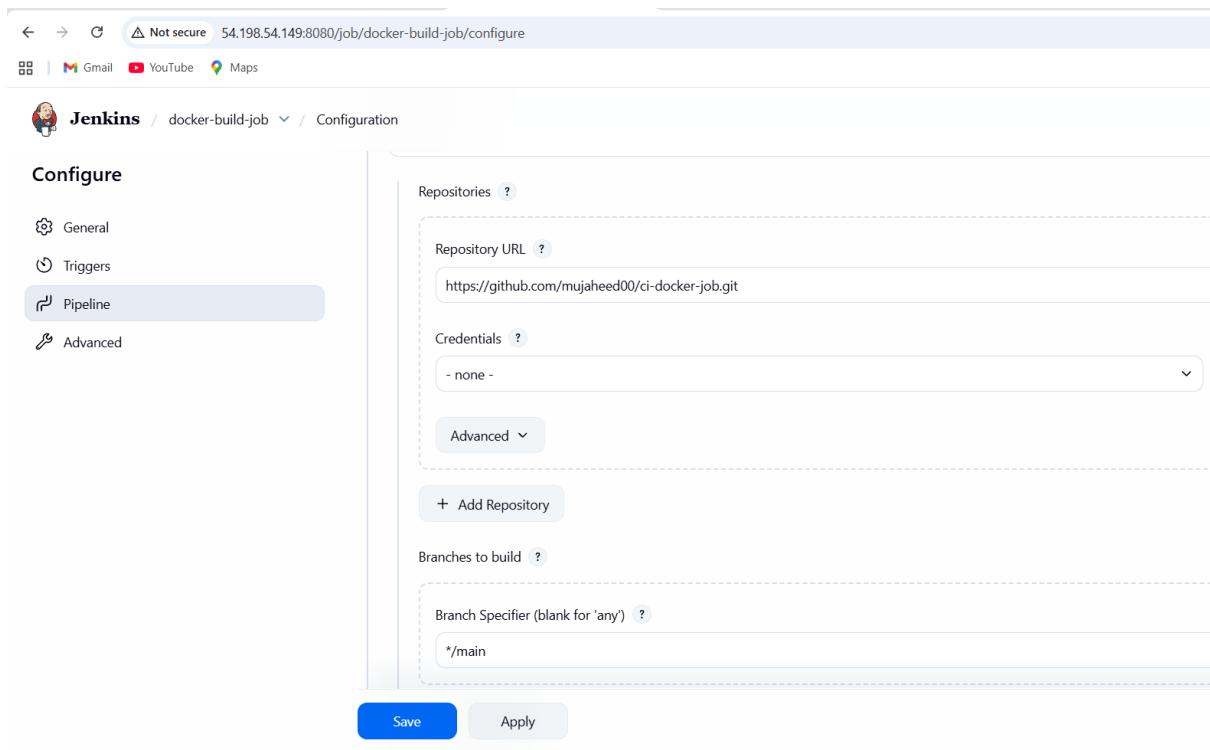
Build a new job with the name docker-build-job and type as pipeline

The screenshot shows the Jenkins 'New Item' creation interface. The title bar says 'New Item'. A text input field is labeled 'Enter an item name' and contains the value 'docker-build-job'. Below it, a section titled 'Select an item type' lists three options:

- Freestyle project**: Described as a classic, general-purpose job type.
- Pipeline**: Described as orchestrating long-running activities suitable for building pipelines.
- Multi-configuration project**: Described as suitable for projects with many configurations.
- Folder**: Described as a container for nested items.

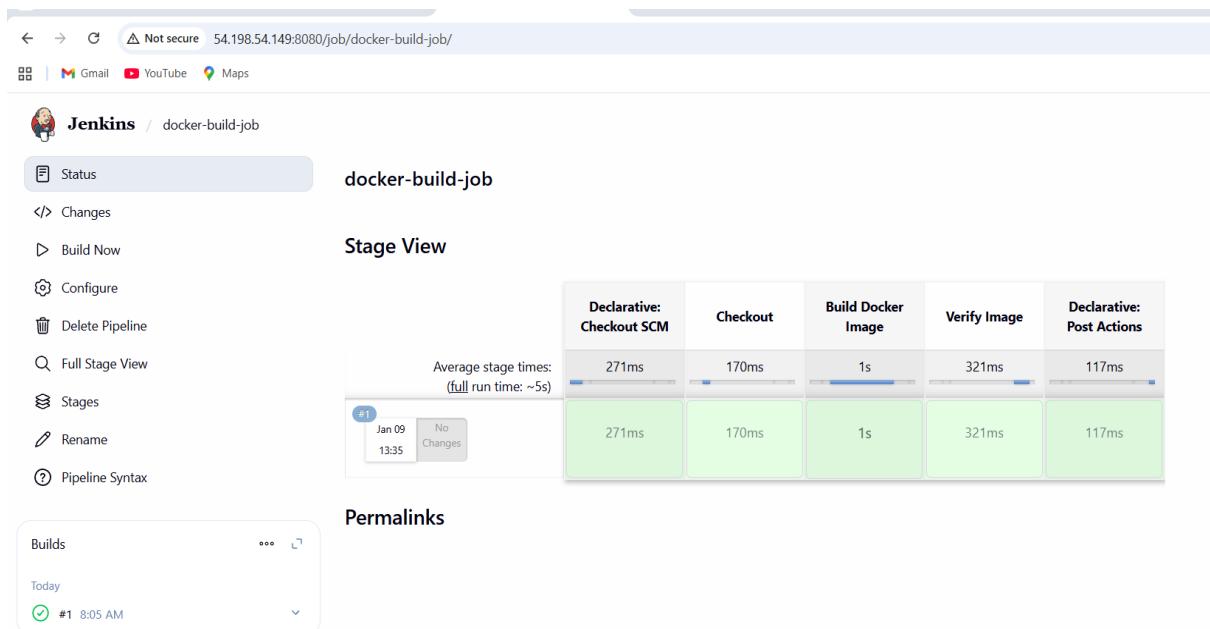
A blue 'OK' button is located at the bottom of the form.

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



The screenshot shows the Jenkins Pipeline configuration page for a job named "docker-build-job". The left sidebar has "Configure" selected, with sub-options: General, Triggers, Pipeline (which is highlighted), and Advanced. The main panel is titled "Repositories" and contains fields for "Repository URL" (set to <https://github.com/mujaheed00/ci-docker-job.git>) and "Credentials" (set to "- none -"). Below this is a "Branches to build" section with a "Branch Specifier" field containing `*/main`. At the bottom are "Save" and "Apply" buttons.

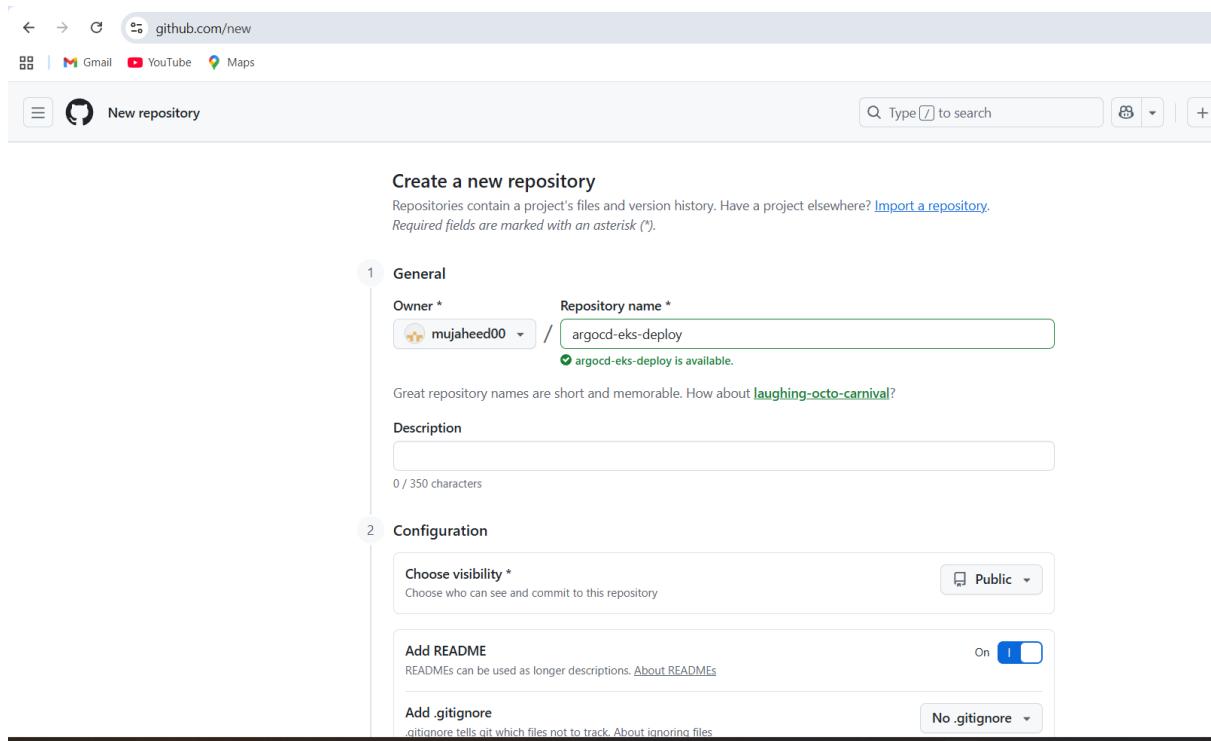
Click on build now.



The screenshot shows the Jenkins Job Stage View for "docker-build-job". The left sidebar has "Status" selected. The main area shows the "Stage View" with five stages: Declarative: Checkout SCM (271ms), Checkout (170ms), Build Docker Image (1s), Verify Image (321ms), and Declarative: Post Actions (117ms). A summary at the top indicates an average stage time of 271ms and a full run time of ~5s. The first stage is labeled "#1 Jan 09 13:35 No Changes". Below the stage view is a "Permalinks" section showing a build log entry for "#1 8:05 AM".

4. Create ArgoCD job to deploy on ek8s cluster.

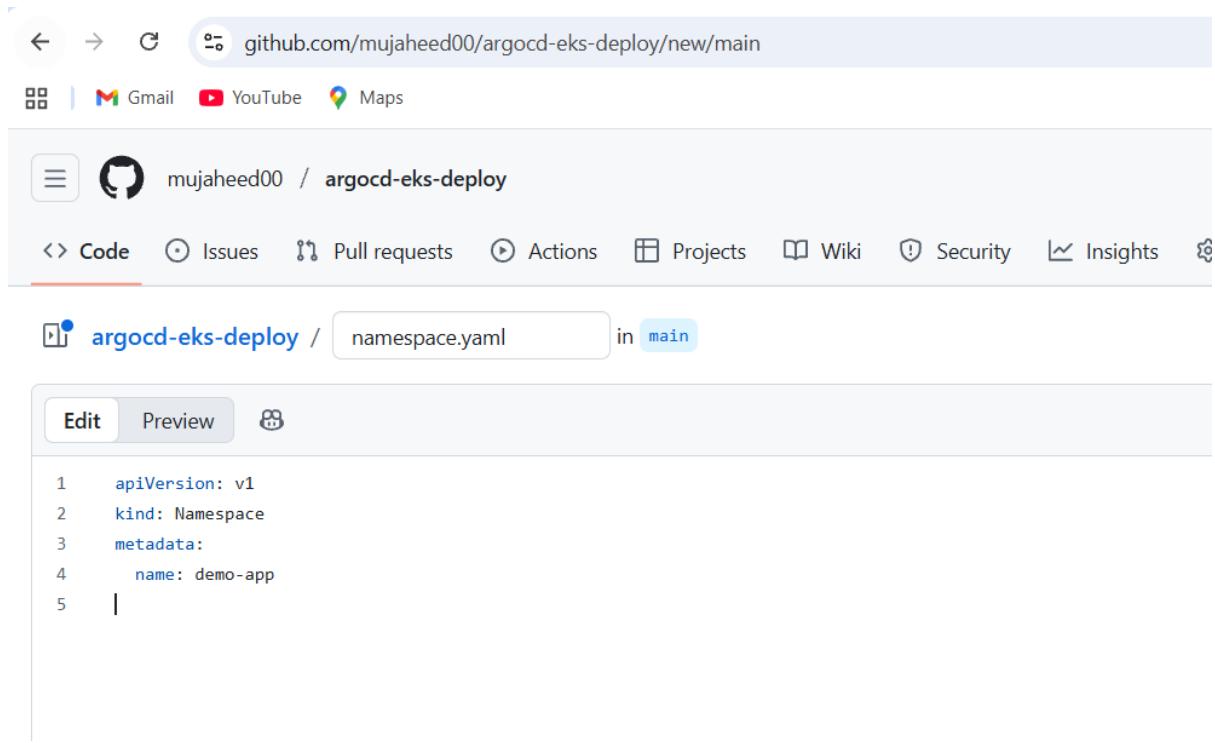
Go to github and click on new repository and name it as argocd-eks-deploy



The screenshot shows the GitHub 'Create a new repository' interface. The repository name is set to 'argocd-eks-deploy'. The configuration section includes visibility set to 'Public', 'Add README' (on), and 'Add .gitignore' (No .gitignore).

In that repository add manifest files.

Namespace.yaml:



The screenshot shows the 'argocd-eks-deploy' repository on GitHub. The 'Code' tab is selected, displaying the 'namespace.yaml' file. The file content is as follows:

```
1 apiVersion: v1
2 kind: Namespace
3 metadata:
4   name: demo-app
```

deployment.yaml

← → ⌂ github.com/mujaheed00/argocd-eks-deploy/new/main

Gmail YouTube Maps

mujaheed00 / argocd-eks-deploy

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

argocd-eks-deploy / deployment.yaml in main

Edit Preview

```
1 apiVersion: apps/v1
2 kind: Deployment
3 metadata:
4   name: demo-app
5   namespace: demo-app
6 spec:
7   replicas: 2
8   selector:
9     matchLabels:
10    app: demo
11   template:
12     metadata:
13       labels:
14         app: demo
15     spec:
16       containers:
17         - name: demo-container
18           image: nginx:latest
19         ports:
20           - containerPort: 80
21
```

Service.yaml

← → ⌂ github.com/mujaheed00/argocd-eks-deploy/new/main

Gmail YouTube Maps

mujaheed00 / argocd-eks-deploy

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

argocd-eks-deploy / service.yaml in main

Edit Preview

```
1 apiVersion: v1
2 kind: Service
3 metadata:
4   name: demo-service
5   namespace: demo-app
6 spec:
7   type: LoadBalancer
8   selector:
9     app: demo
10  ports:
11    - port: 80
12      targetPort: 80
13
```

Go to argocd GUI and click on new app

The screenshot shows the argocd GUI interface. On the left is a dark sidebar with icons for Applications, Settings, User Info, and Documentation. The main area is titled 'Applications' and features a large circular icon with three stacked squares. Below it, the text reads 'No applications available to you just yet' and 'Create new application to start managing resources in your cluster'. At the top of this section is a button labeled '+ NEW APP'. A red arrow points from the bottom left towards this button. Other buttons at the top include '+ SYNC APPS' and 'REFRESH APPS'. A search bar and a refresh icon are also present.

The screenshot shows the 'CREATE' dialog for a new application. The sidebar on the left remains the same. The main area has tabs for 'CREATE' and 'CANCEL', with 'CREATE' being active. The 'GENERAL' tab is selected, showing fields for 'Application Name' (set to 'demo-app') and 'Project Name' (set to 'default'). Below these are sections for 'SYNC POLICY' (set to 'Manual'), 'SYNC OPTIONS' (checkboxes for 'SKIP SCHEMA VALIDATION', 'PRUNE LAST', 'RESPECT IGNORE DIFFERENCES', 'AUTO-CREATE NAMESPACE', 'APPLY OUT OF SYNC ONLY', and 'SERVER-SIDE APPLY'), and 'PRUNE PROPAGATION POLICY' (set to 'foreground'). A red arrow points from the bottom left towards the 'CREATE' button.

Source: which you had created for manifest files

Branch: main

Path: .

Not secure <https://3.84.245.249:30722/applications?new=%7B%22apiVersion%22%3A%22argoproj.io%2Fv1alpha1%22,%22kind%22%3A%22Application%22,%22metadata%22%3A%7B%22name%22%3A%22demo-app%22,%22labels%22%3A%7B%22app%22%3A%22true%22%7D%7D>

G

B

U

CREATE CANCEL

SOURCE

Repository URL
https://github.com/mujaheed00/argocd-eks-deploy.git

Revision
main

Path
.

DESTINATION

Cluster URL
https://kubernetes.default.svc

Not secure <https://3.84.245.249:30722/applications?new=%7B%22apiVersion%22%3A%22argoproj.io%2Fv1alpha1%22,%22kind%22%3A%22Application%22,%22metadata%22%3A%7B%22name%22%3A%22demo-app%22,%22labels%22%3A%7B%22app%22%3A%22true%22%7D%7D>

3 Gmail YouTube Maps

CREATE CANCEL

DESTINATION

Cluster URL
https://kubernetes.default.svc

Namespace
demo-app

Directory ▾

DIRECTORY

DIRECTORY RECURSE

Click on create

The screenshot shows the Argo UI interface. On the left is a sidebar with the Argo logo and version v3.2.3+2b6251d. The sidebar includes links for Applications, Settings, User Info, Documentation, Application filters (Favorites Only, SYNC STATUS, HEALTH STATUS), and a SYNC APPS button. The main content area is titled 'Applications' and shows a card for 'demo-app'. The card details are:

Project:	default
Labels:	
Status:	Missing OutOfSync
Repository:	https://github.com/mujaheed00/argocd-
Target R...	main
Path:	.
Destinati...	in-cluster
Namesp...	demo-app
Created ...	01/09/2026 15:07:07 (a few seconds ago)

At the bottom of the card are three buttons: SYNC, REFRESH, and DELETE.

Click on sync

The screenshot shows the Argo UI interface, identical to the first one but with a red arrow pointing to the 'SYNC' button in the 'demo-app' application card. This indicates where the user should click to trigger the sync operation.

The screenshot shows the Argo UI interface. On the left is a sidebar with icons for Applications, Settings, User Info, and Documentation. Below these are sections for Application filters, SYNC STATUS, and HEALTH STATUS. The main content area is titled 'Applications' and contains a card for 'demo-app'. The card details include: Project: default, Labels: Missing OutOfSync, Status: Missing, Repository: https://github.com/mujaheed00/argocd-eks-deploy.git, Target R...: main, Path: ., Destination: in-cluster, Namespace: demo-app, and Created: 01/09/2026 15:07:07 (2 minutes ago). At the bottom of the card are three buttons: SYNC, REFRESH, and DELETE. To the right of the card is a 'SYNC' dialog box. The dialog has a 'Synchronize' button at the top right and a 'CANCEL' button. It displays the message 'Synchronizing application manifests from https://github.com/mujaheed00/argocd-eks-deploy.git'. Below this are sections for Revision (main), SYNC OPTIONS (with checkboxes for PRUNE, DRY RUN, APPLY ONLY, FORCE, SKIP SCHEMA VALIDATION, AUTO-CREATE NAMESPACES, PRUNE LAST, APPLY OUT OF ORDER, RESPECT IGNORE DIFFERENCES, SERVER-SIDE APPLIED, REPLACE, and RETRY), and PRUNE PROPAGATION POLICY (foreground). A red arrow points from the status bar at the top right to the 'SYNC' button in the top right corner of the main content area.

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

The screenshot shows a web browser window with the URL 3.84.245.249:30199. The page title is 'Welcome to nginx!'. The page content includes: 'If you see this page, the nginx web server is successfully installed and working. Further configuration is required.'; 'For online documentation and support please refer to nginx.org. Commercial support is available at nginx.com.'; and 'Thank you for using nginx.'