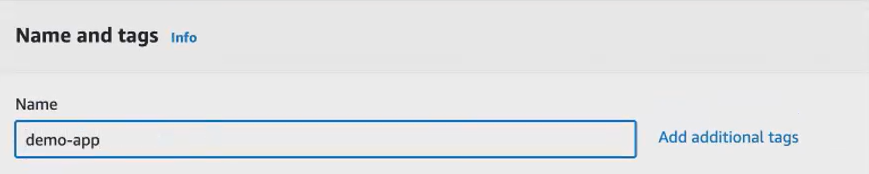
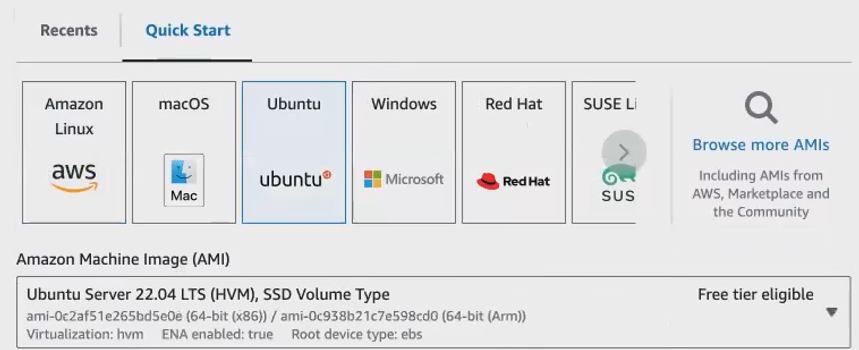
**Day 4- Kubernetes - Pods & Services [ in this build java application]**

<https://docs.google.com/document/d/1Is4h94KVFliaNxSuBZX9Spui98CYiFOIawjLfYa3wBU/edit> [this is very important.do this must]

<https://github.com/zen-class/zen-class-devops-documentation>

launch an instance & connect:





Select t2.medium.because install docker, Jenkins , Kubernetes cluster in single machine

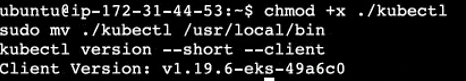


**3 tools are required to launch your Kubernetes cluster**

1. **Kubectl**
2. **Eksctl**
3. **Jenkins**
4. **Awscli**
5. **Create Kubernetes cluster**

**Kubectl:**





**Eksctl:**





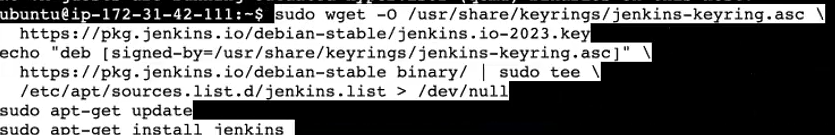
Duplicate the machine to install Jenkins ,docker:

1. install Jenkins [inside Jenkins install awscli & create Kubernetes cluster]
2. install docker

install Jenkins:

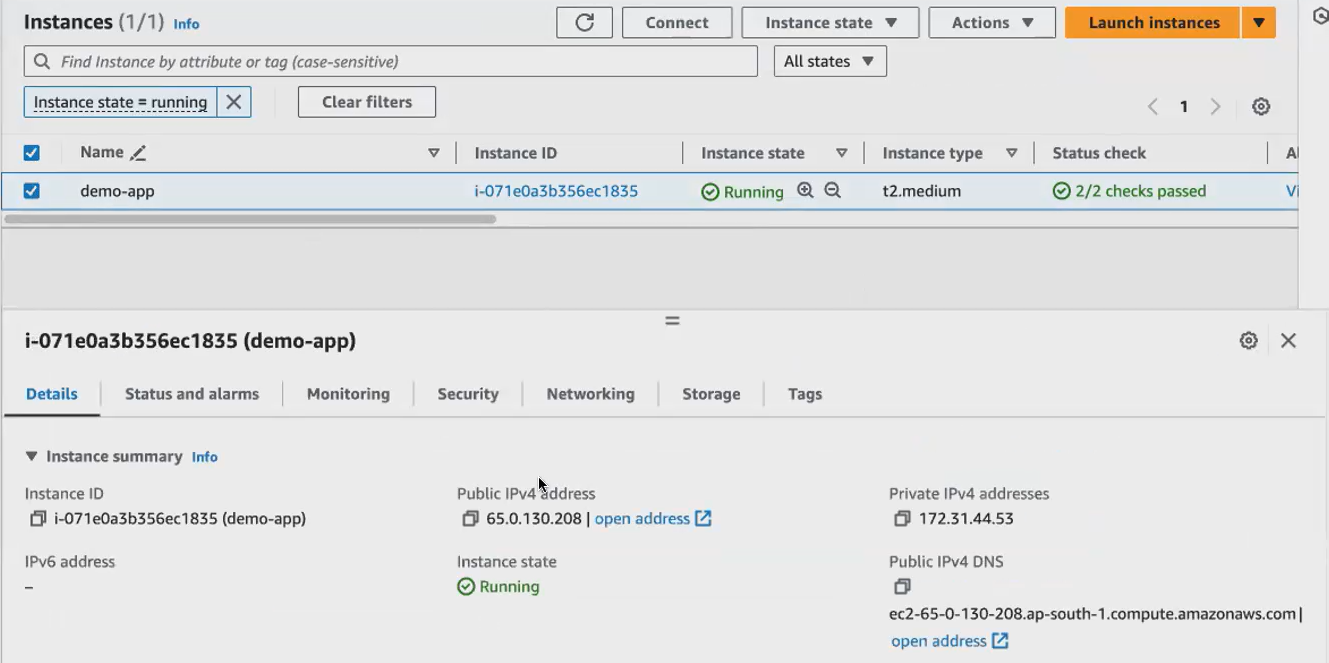




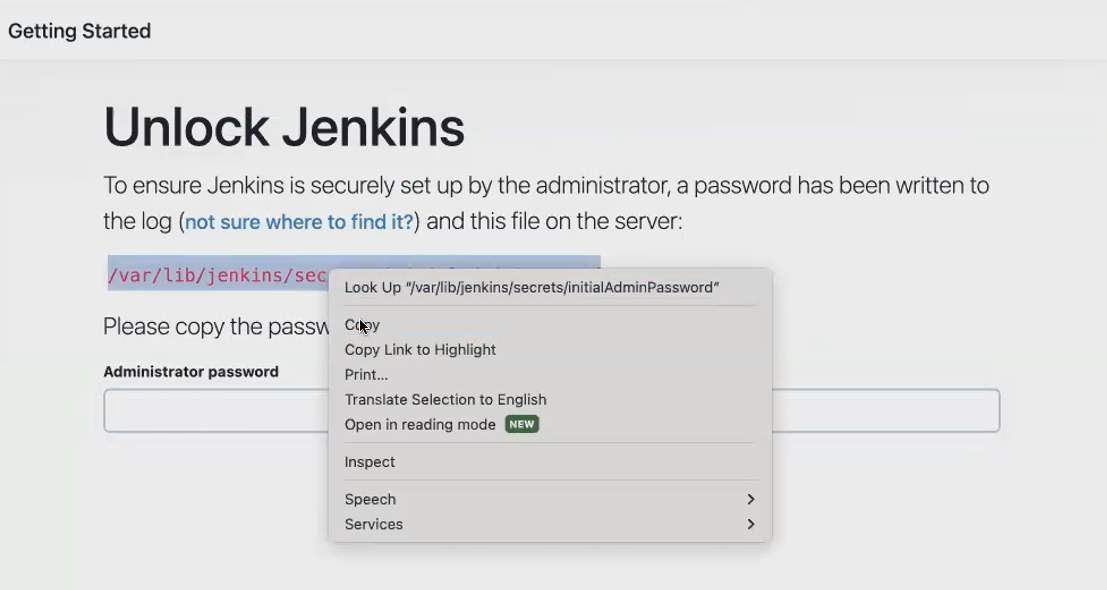


Duplicate the machine: [where we installed Jenkins]

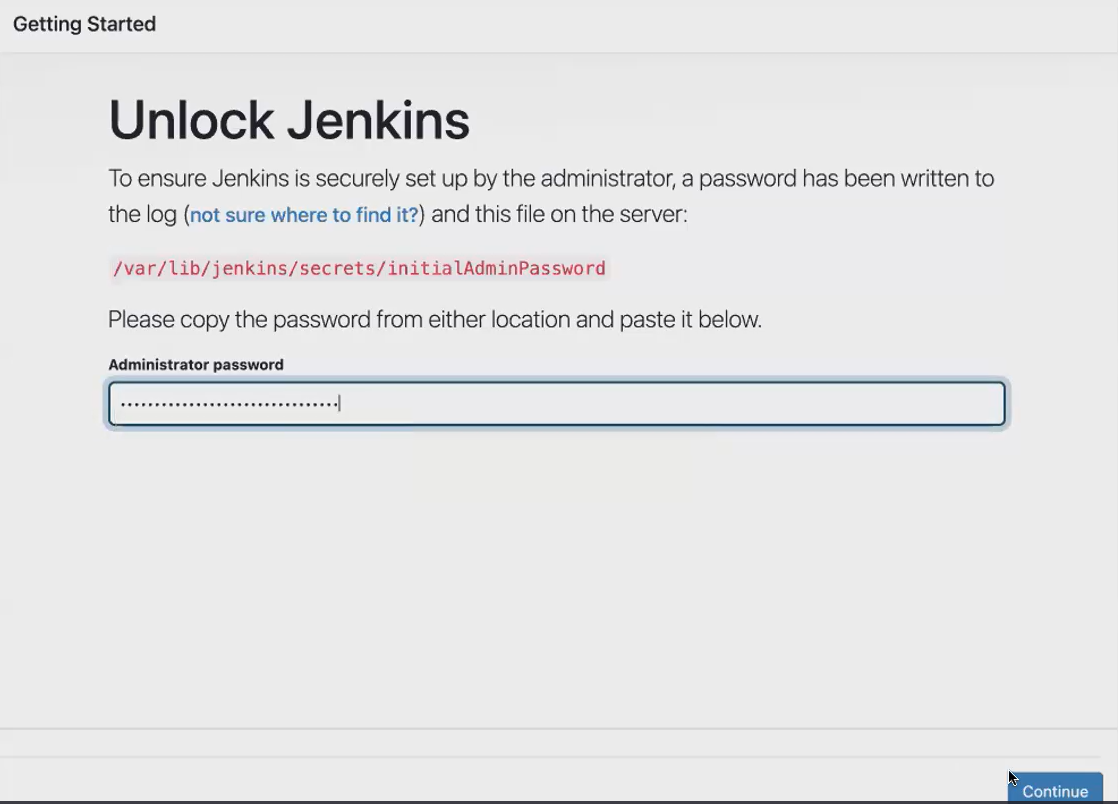
* So now you can see my Jenkins from the Jenkins user and launching my cluster.
* Okay, let it launch. I will take another session.
* So, in this machine we have installed Jenkins. Right, we will go and open up the port number [8080].

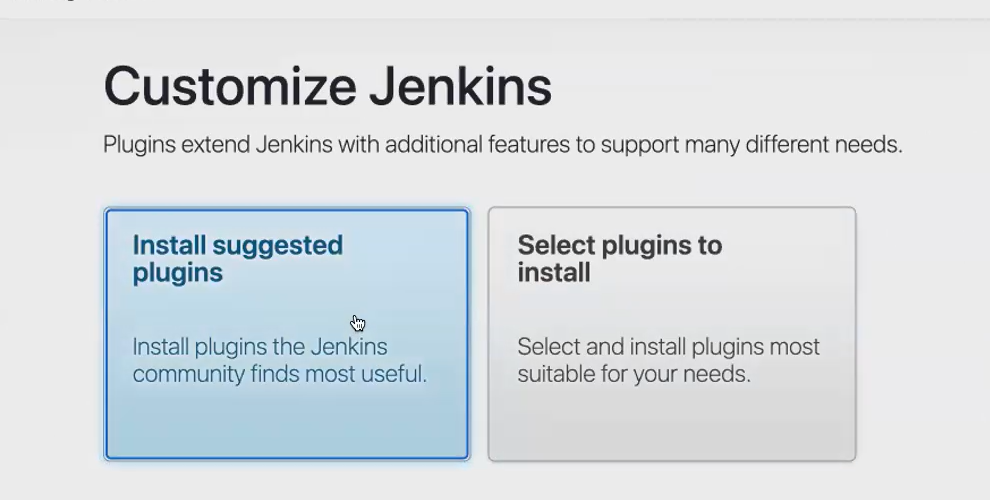


Jenkins page opened.









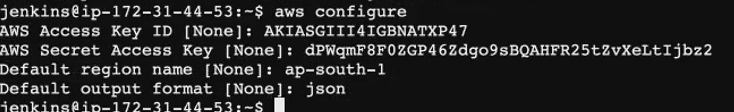
Get into the Jenkins user:

* 1. install awscli
  2. create Kubernetes cluster

get into Jenkins user [sudo su -jenkins]

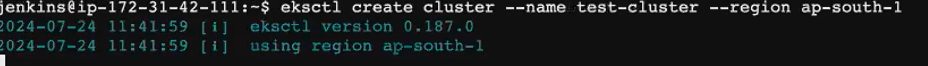


Awscli:

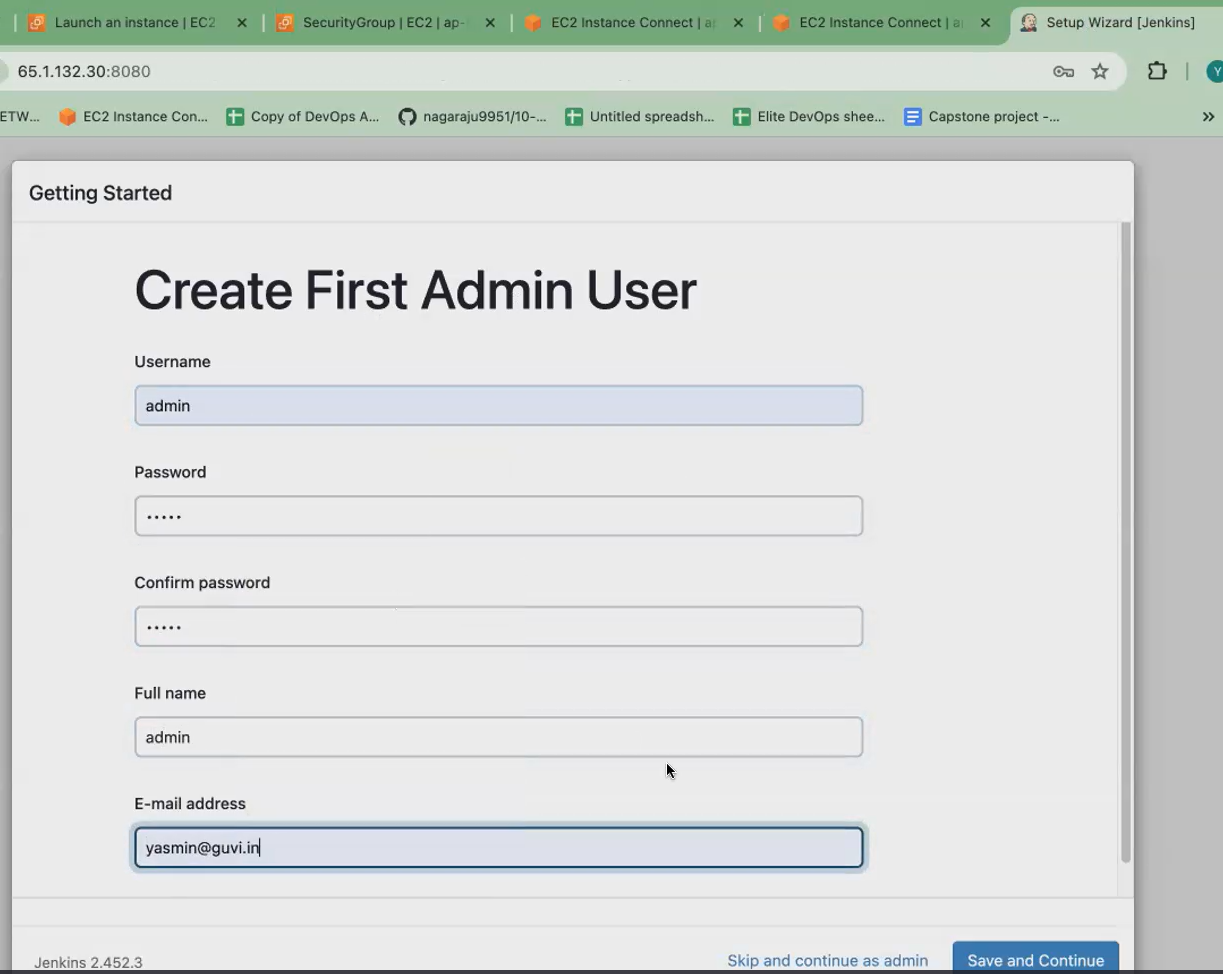


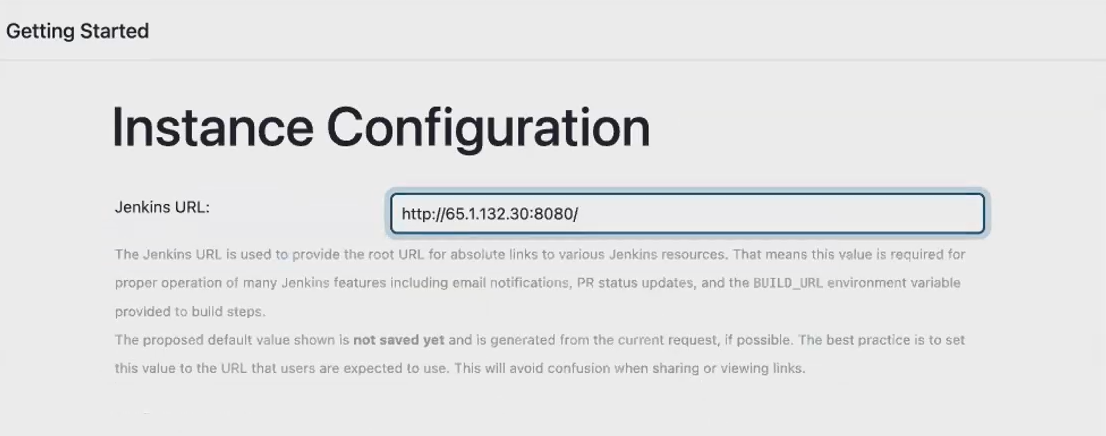
Create Kubernetes cluster inside Jenkins user:

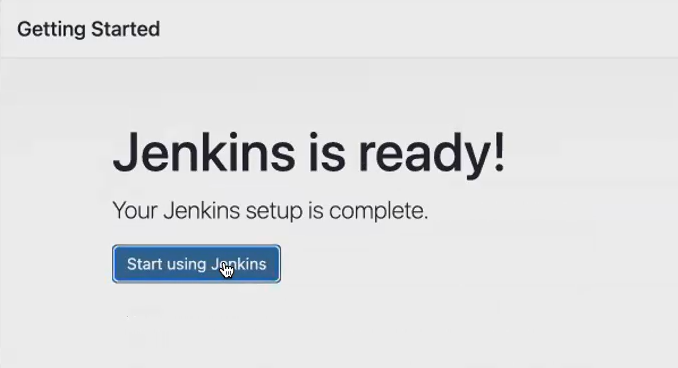
* Now you're inside the Jenkins user. from the Jenkins user Only, we have to launch the cluster.
* Only then your Jenkins can identify this cluster.
* Create cluster name of your cluster
* which region you want to launch a cluster
* type of worker node that you want to launch.
* worker node by default will be created with instance type t2.small. now the cluster, while the cluster is being created.

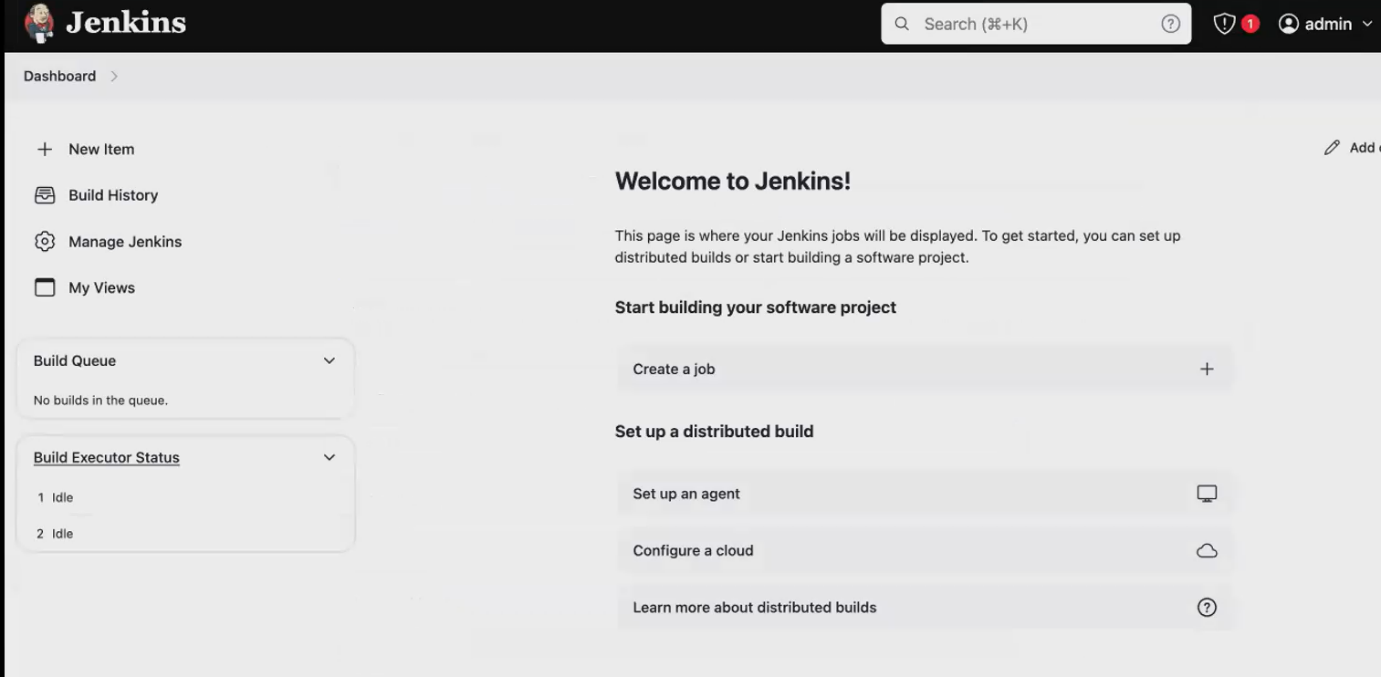


open Jenkins page:









**All are doing in the same machine.**

* so once you have launched the cluster, what you have to do.
* our application, will go and look into the application. Our application is a Java application**.**
* so, you have to build your application. That is, you have to artifact. You have to package your application, using Maven so for that what you need. You have to install, Maven.

**Install maven:**

* why, I'm installing in this machine.
* My Jenkins have to build the application. My Jenkins should build the Java application via Maven, so I have to give my Jenkins. I have to install maven and my Jenkins machine.
* So, I have installed Maven.
* so, I'm installed Maven, Java, Jenkins. Everything is running.
* We will also install Docker. So do we have to install





**Install docker:**



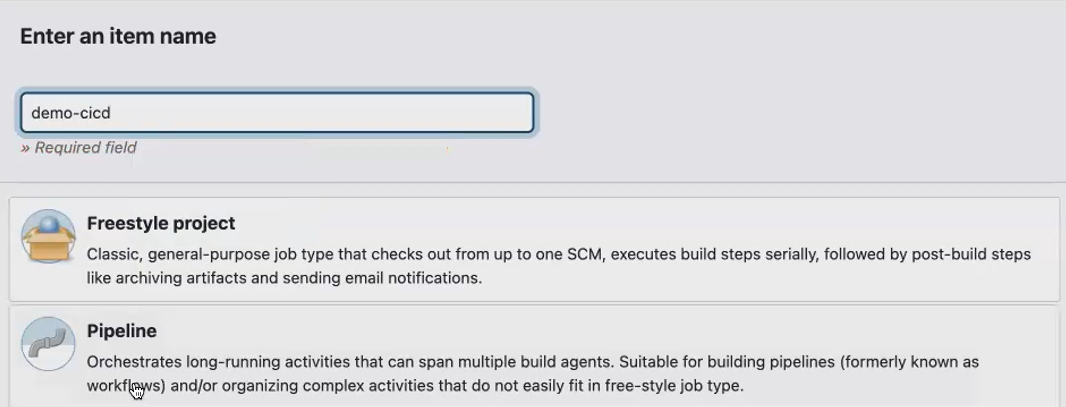


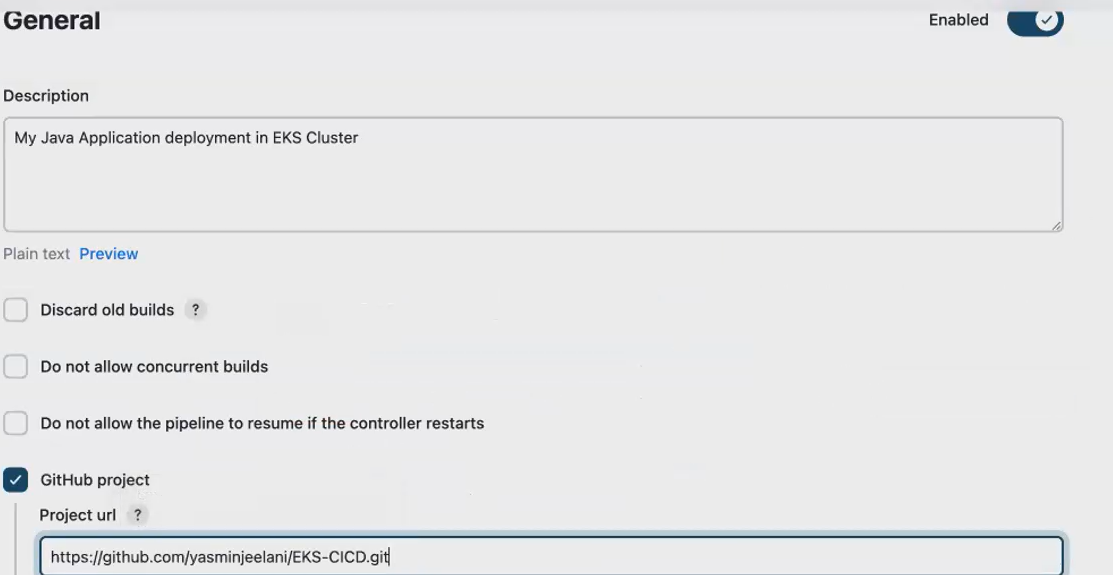
* after installing Docker, make sure you give permission for Jenkins and Ubuntu to the docker user.
* So you don't have to use the pseudo command to run your docker commands.



**Create a pipeline in Jenkins:**

**Click build project**





Later on do automation. don’t select poll scm



1. tools

2. environment

3. stages

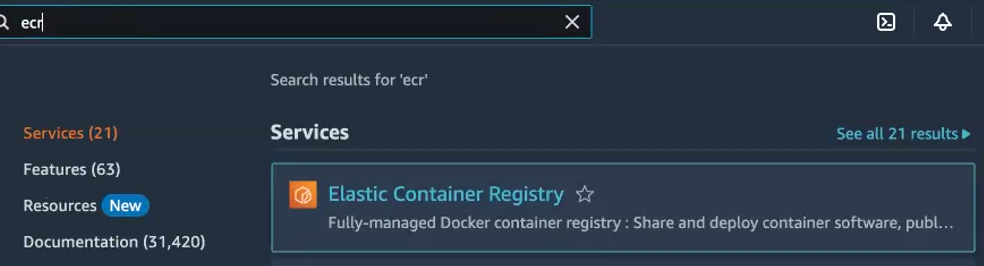
1st stage [checkout]

2nd stage [built the application using maven]

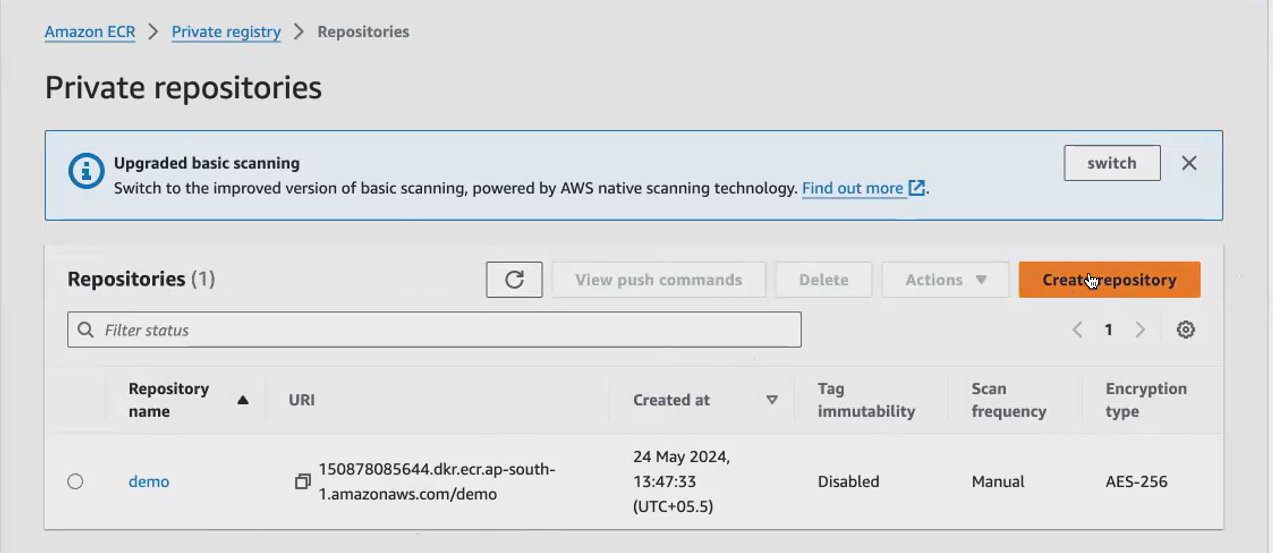
3rd stage [build docker ]

4th stage [ push the docker image to ecr registry]

1. Your Jenkins file always starts with pipeline. And after the pipeline, what you need.
2. you have to before agent. Before you specify any agent here,
3. we will be specifying the tools. So here I'm using a tool called Maven. So here I'm using a tool called Maven, and I'm naming that tool as Maven 3.
4. I will tell you where to configure this, Maven 3. you have installed Maven in your Jenkins machine, and you should also bring that Plugin. You have to install that Plugin Maven Plugin to your Jenkins
5. next is the environment. So here I will be using my Ecr registry. Once the docker build is ready, once the docker is ready. Once your docker image is ready, we will be pushing the docker image to the Ecr registry.
6. So where you get the Ecr registry, you know, aws, you have something. You can also push it to your Docker Hub registry. But since we have already done it, we have already seen configuring the Doctor Hub registry without Jenkins.

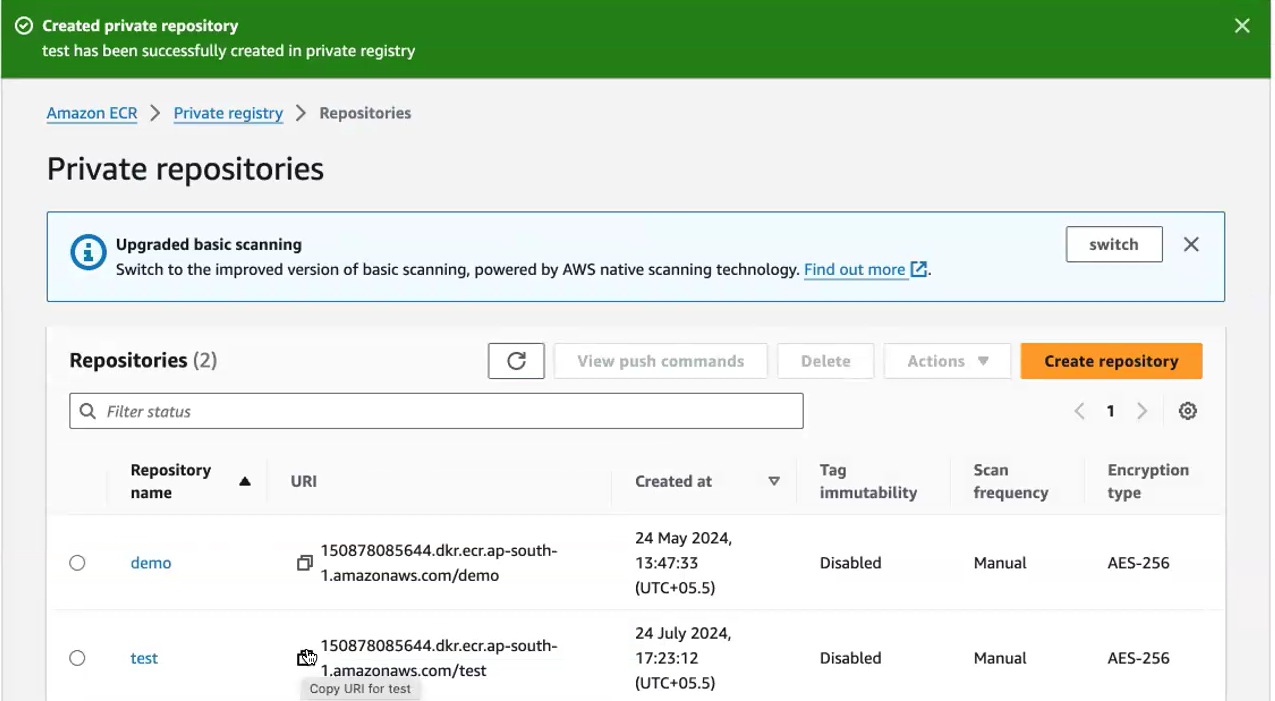


This is same as docker repository. We have push the docker image in this repo.



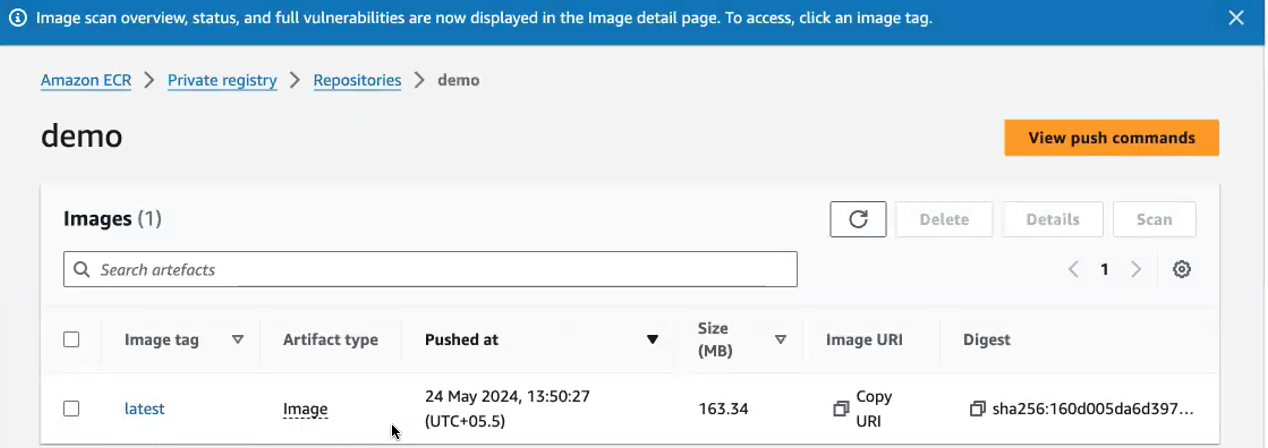


Click create repository



Copy the link and give it in Jenkins file.





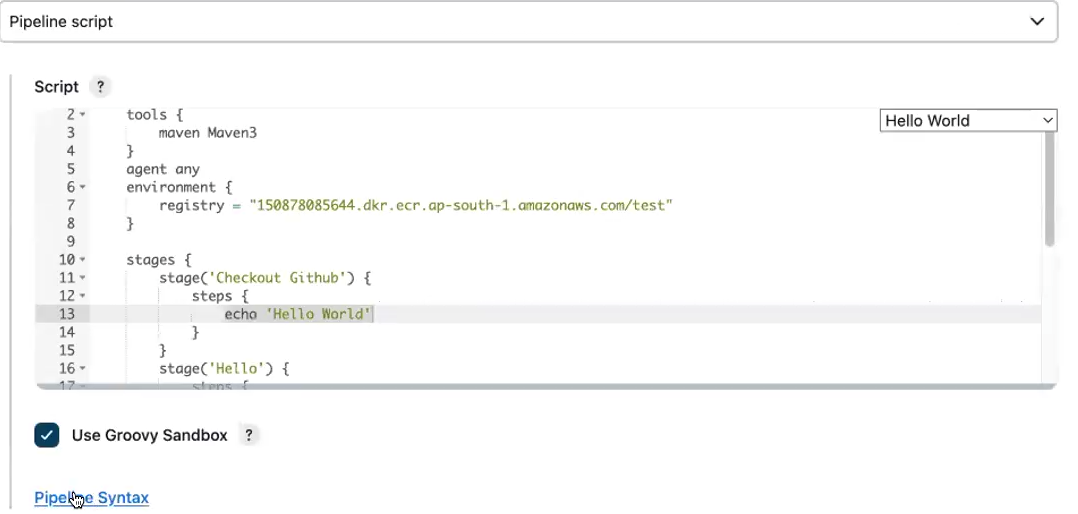
1. So, you create a registry and you pass the URL or Ecr registry URL. Here. Okay, so to this registry we will be pushing our docker image.

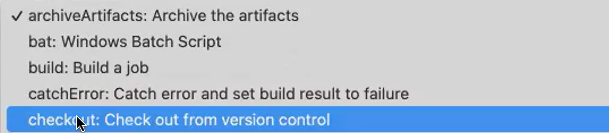
**1st stage [checkout]:**

1. So what is the 1st stage? 1st is your checkout stage. 1st you have where to check out.
2. Your Jenkins has to fetch the code from the Github Repository. Okay, so here you have to give the checkout link.

How-to get this path .steps given below:













Copy the path in 1st stage

**2nd stage [build stage]:**

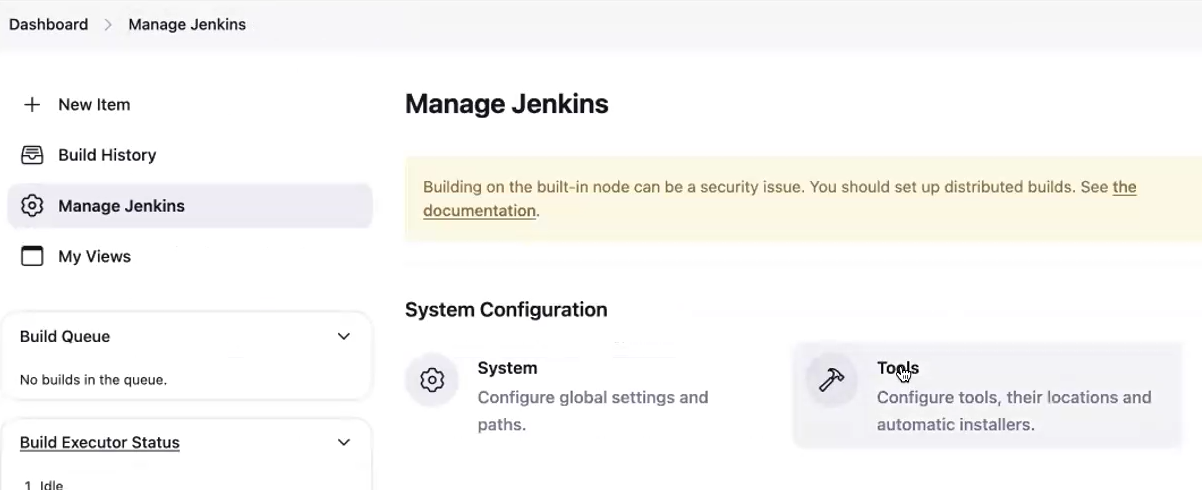
* So, we will be building it using maven. So you have to perform a maven build.
* To perform Maven, build what you need. You have to integrate your Jenkins with Maven. Since you're already installed it. Okay, you have already installed Maven. I will just enable the main plugin and this Gui.

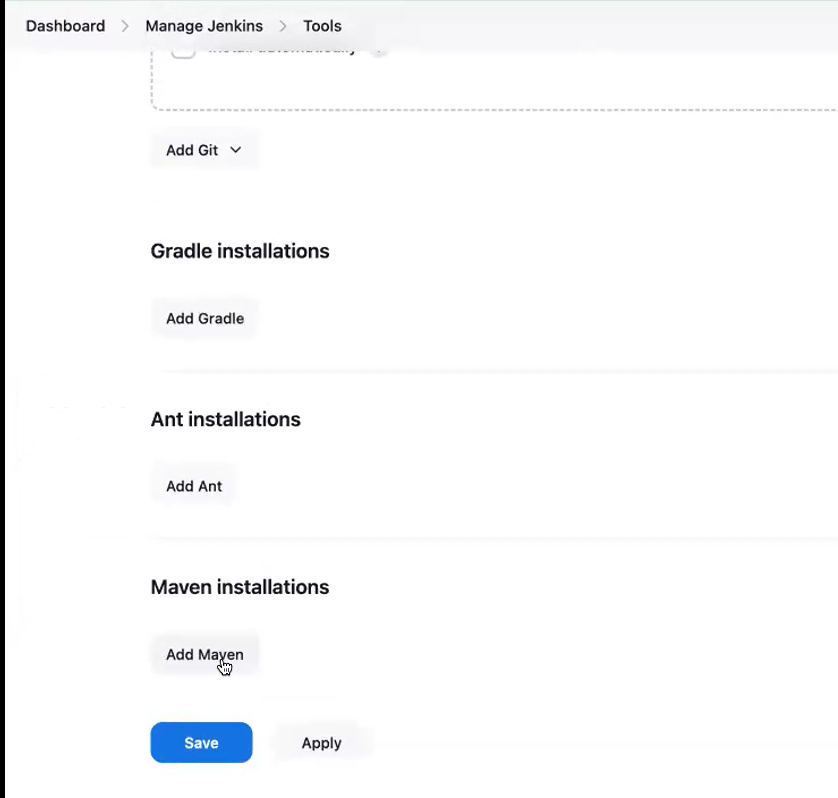


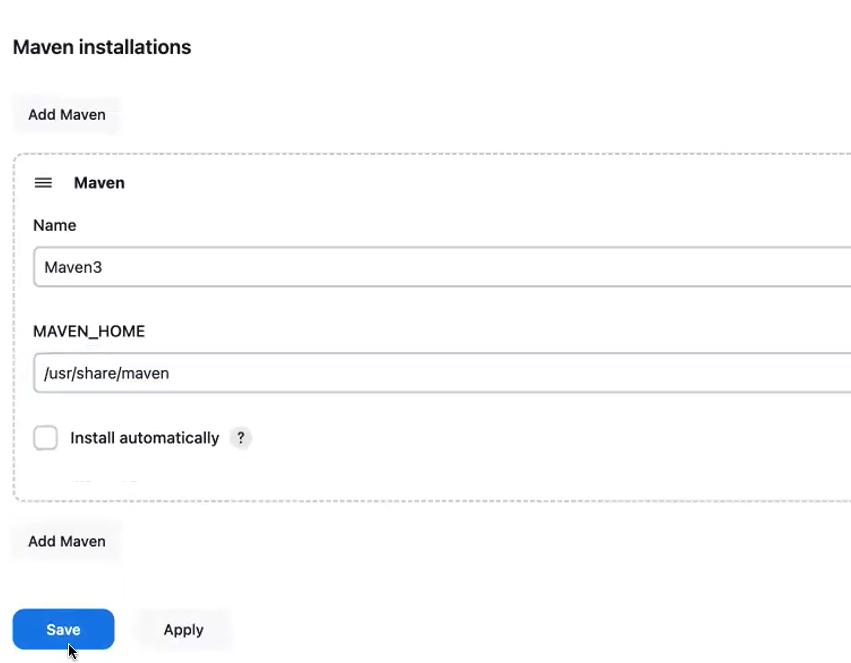
 I've already installed. If you haven't installed Maven, you can give installed automatically, it will install it in your Jenkins instance.

**Give maven plugin in Jenkins:**

**Open in new tab. set up the Maven tool in our Jenkins**







**3rd stage [build docker]:**

**then we will be building the application using docker.**

**Instead to give docker image name give ecr registry**

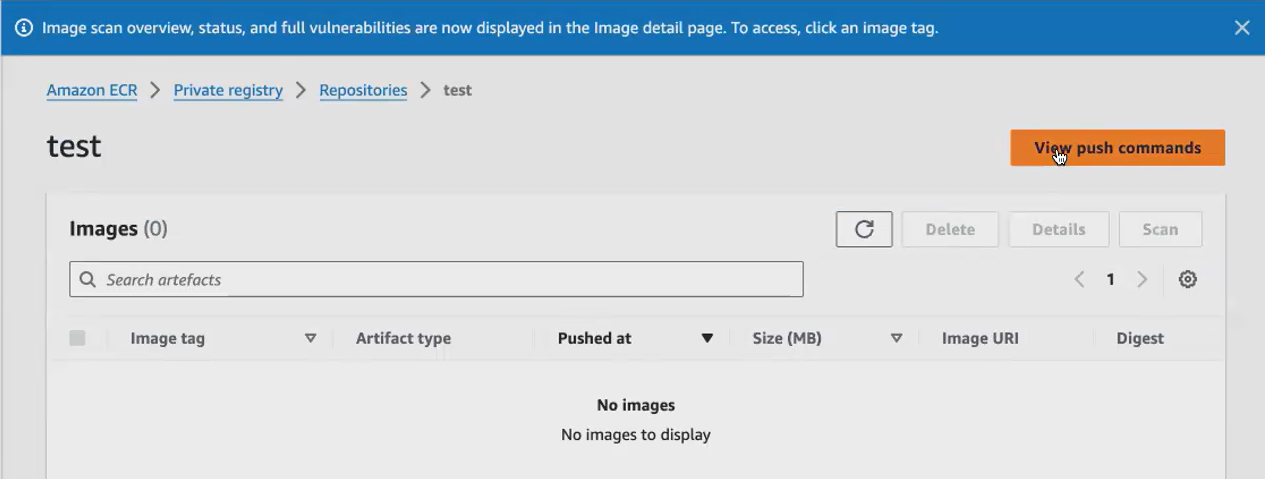


**4th stage [ push the docker image to ecr registry]:**



* so we will be pushing the docker image to the Ecr registry.
* So to before pushing your docker image to the docker hub registry. What do you do?
* You have to do a docker login
* after doing the docker login only we will be able to push the docker image to the docker Hub register.
* This Ecr belongs to Aws. It is. It doesn't belong to your docker hub. I
* it belongs to aws. So, using the Aws cli command, you will be logging into the Ecr registry after successful logging to ecr registry.
* you will be pushing your application; you will be pushing your docker image to the Ecr registry.

**Docker login:**





Copy and paste the login in Jenkins file

Docker build already given.

Docker push command copy and paste it in Jenkins file

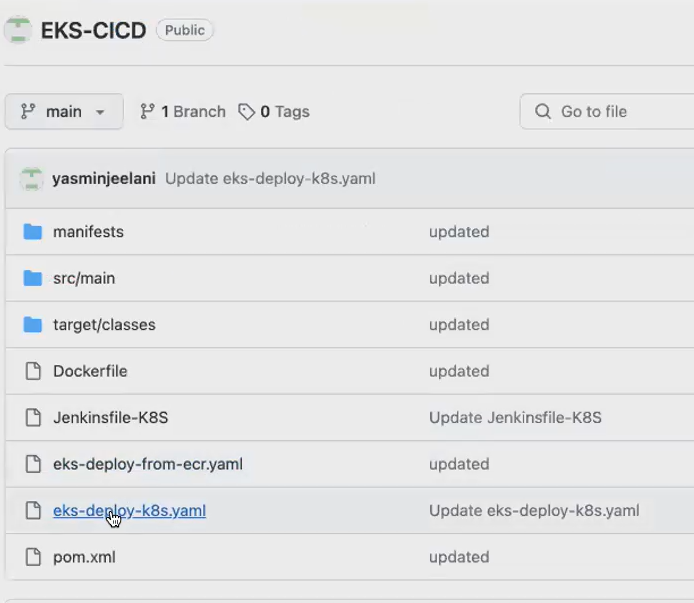




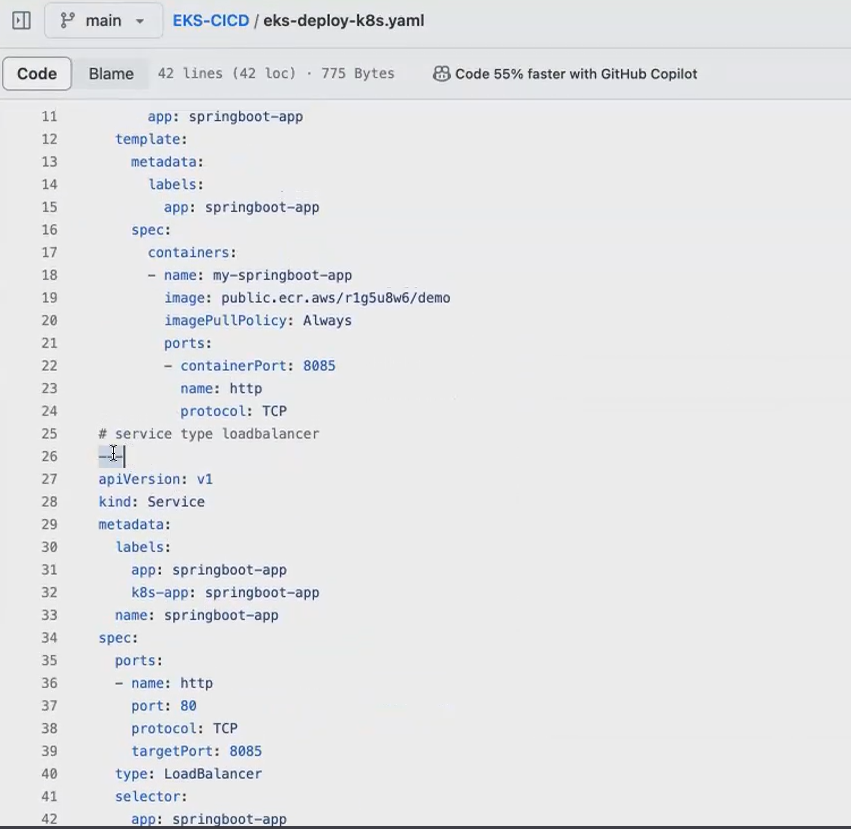


**5th stage** [**deploying the application to the Kubernetes cluster]:**

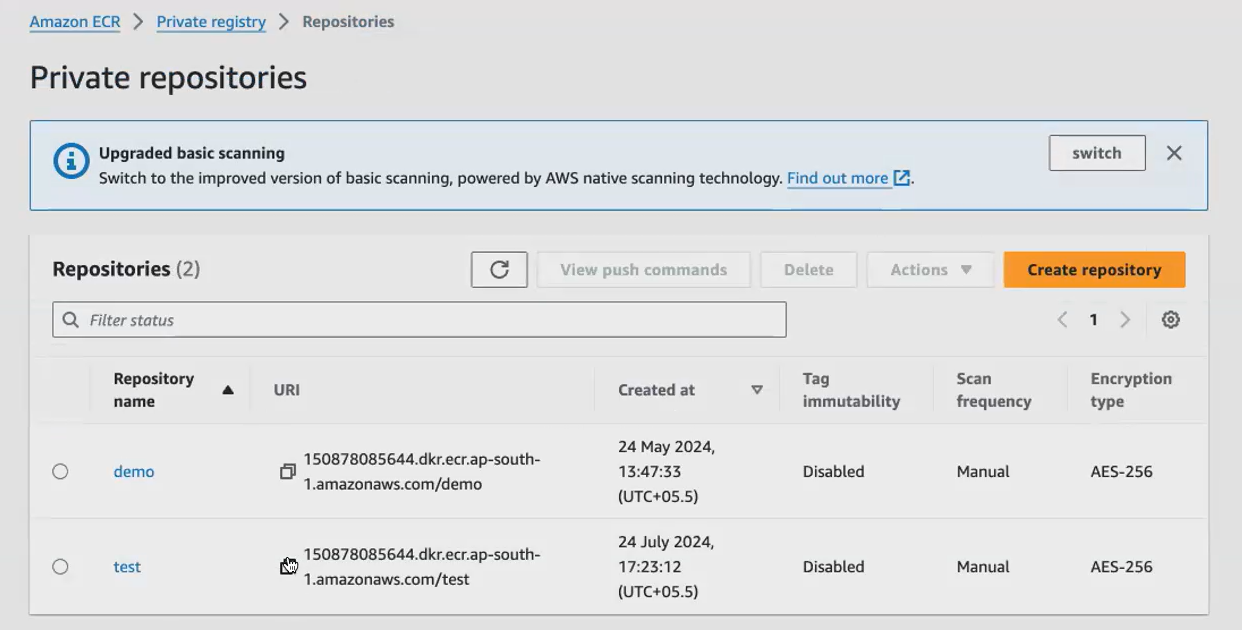
For deployment we have deployment file.in github Yasmin have deployment file and service file in single file. **Service file used to access the application via browser.**



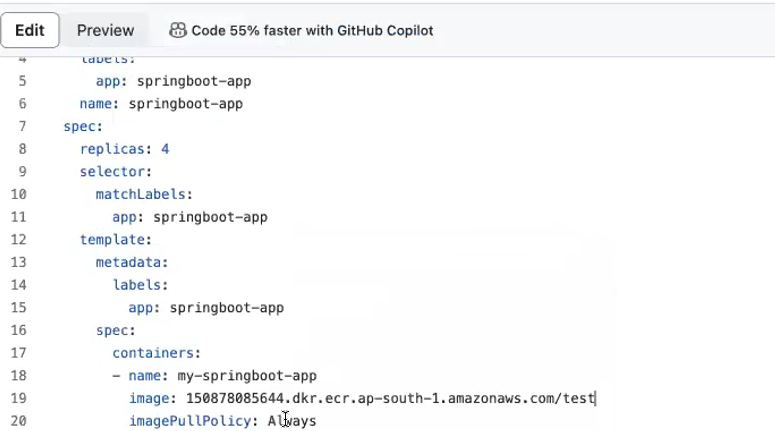
Both files separated by ---

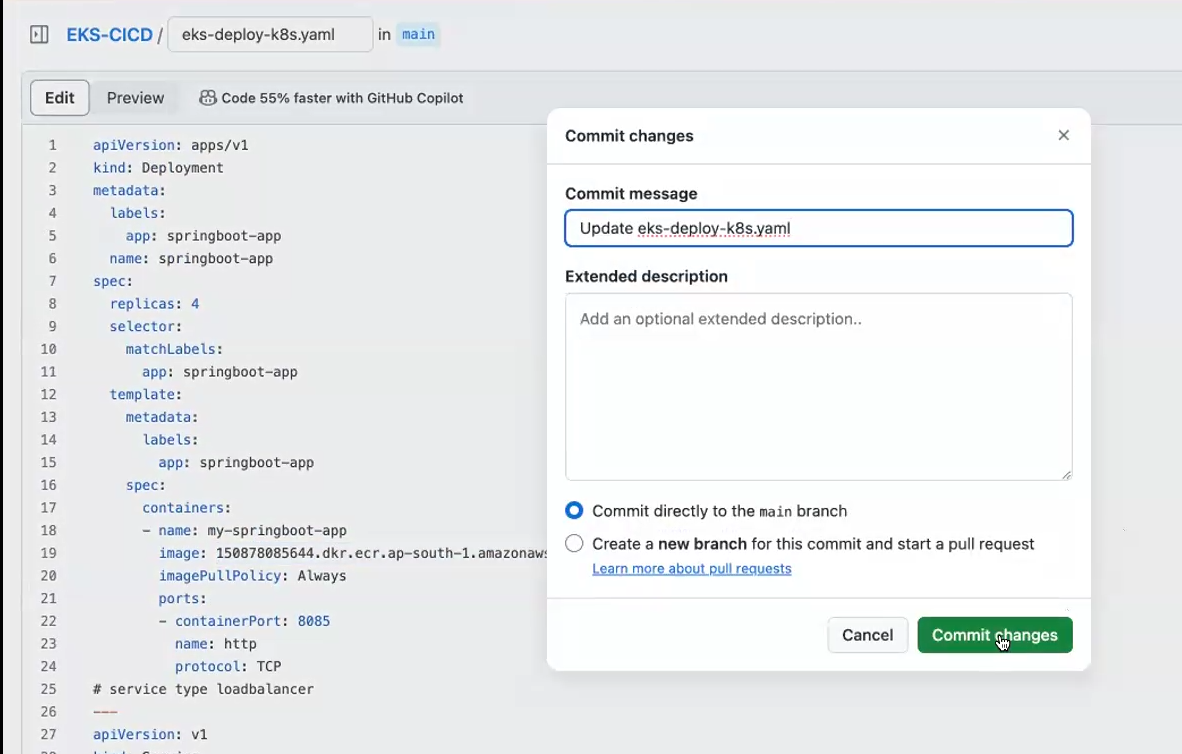


* **Docker image and you have pushed it to the Ecr registry. Assume that inside this registry you have your docker image.**

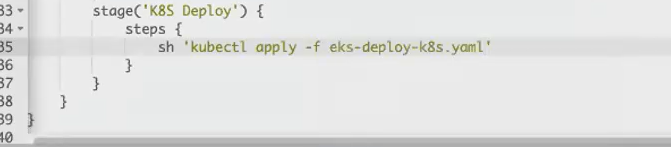


* **copy this registry name. Go to your deployment file and paste it here.**

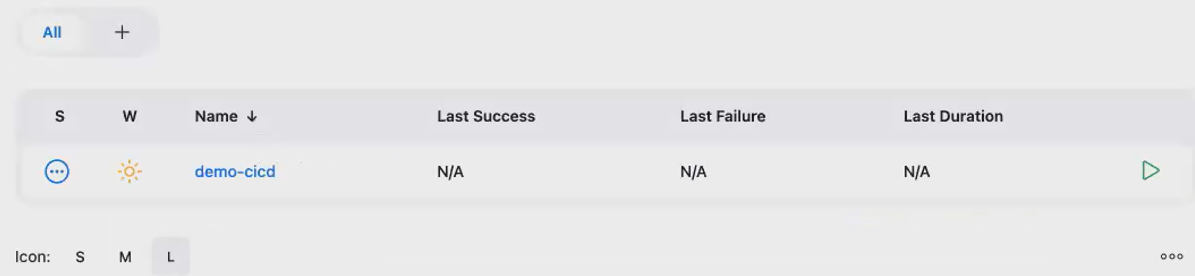




**Then give the deployment command with deployment file name [eks-deploy-k8s.yaml]**

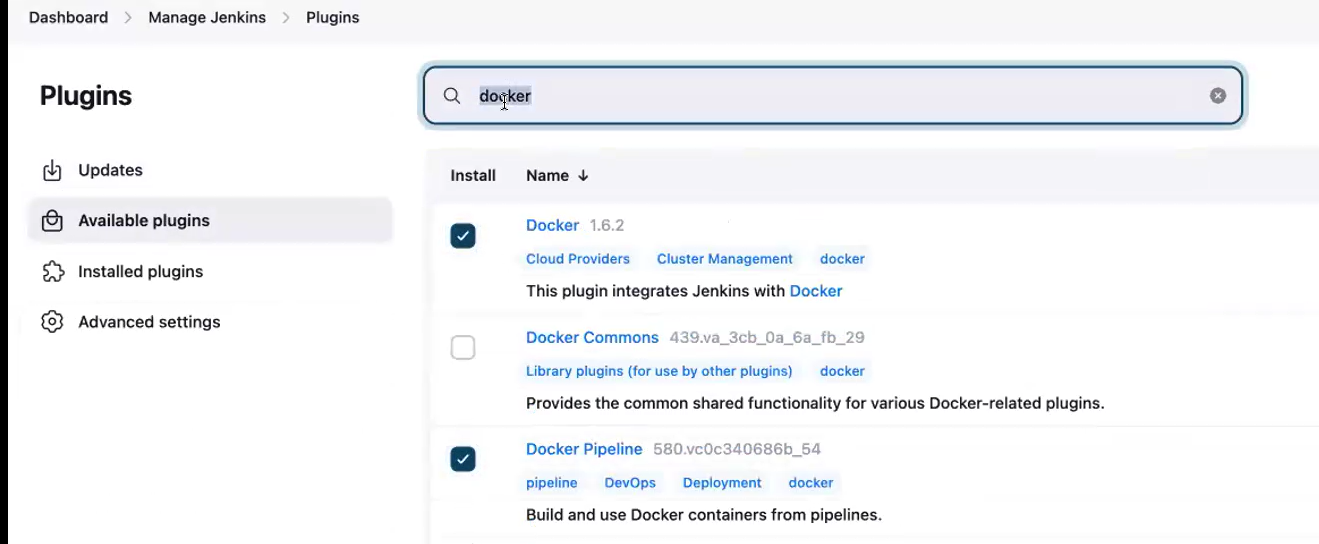


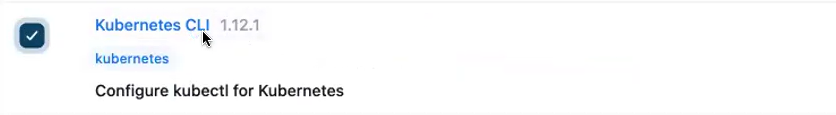
**Let save this file**

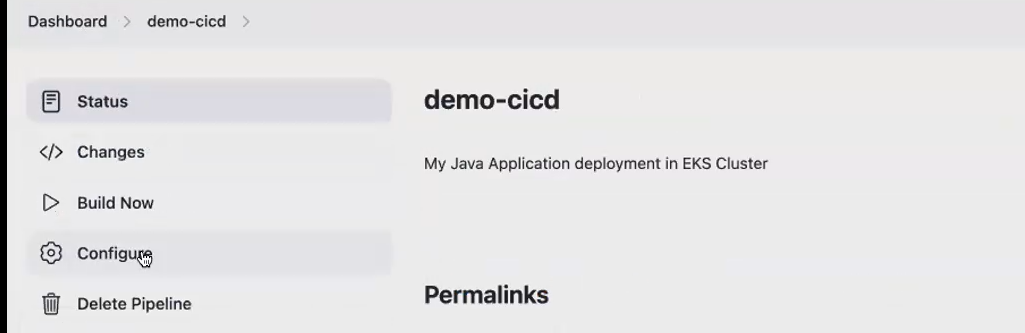


**Install plugins**

**Goto dashboard 🡪 manage Jenkins🡪plugins🡪 available Jenkins🡪search docker and Kubernetes cli 🡪select this 2 🡪click install**

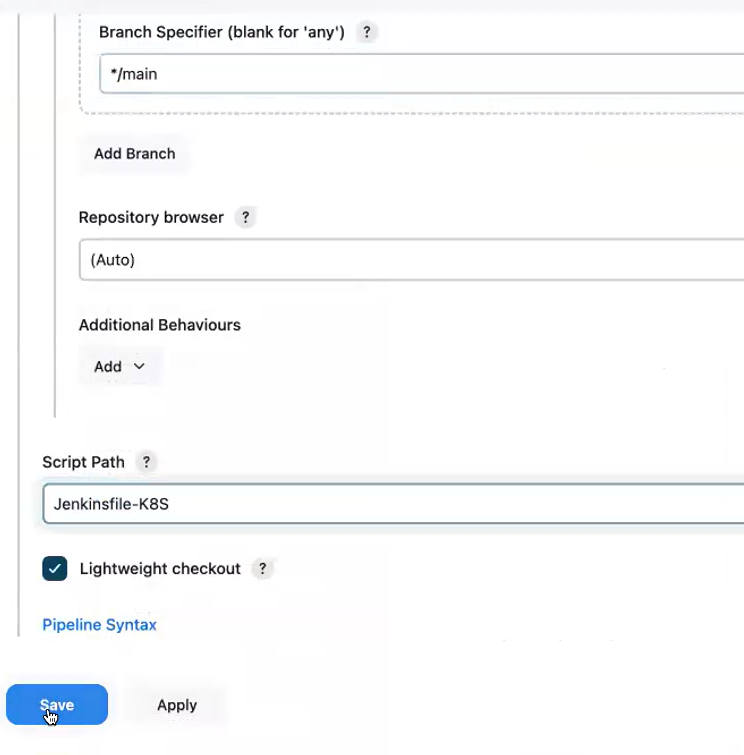


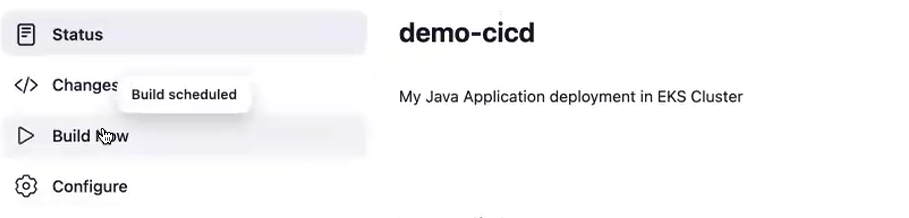


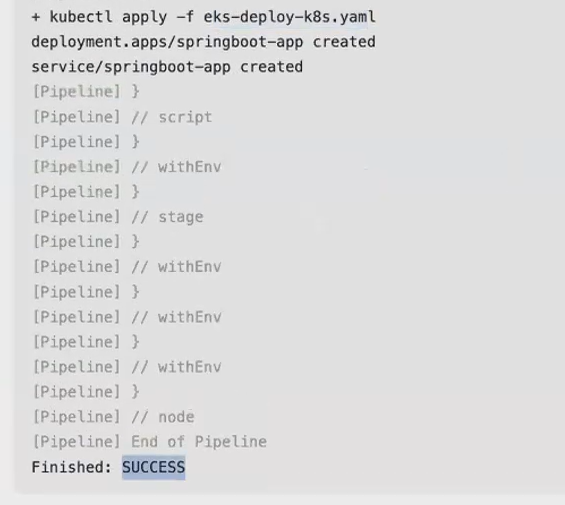


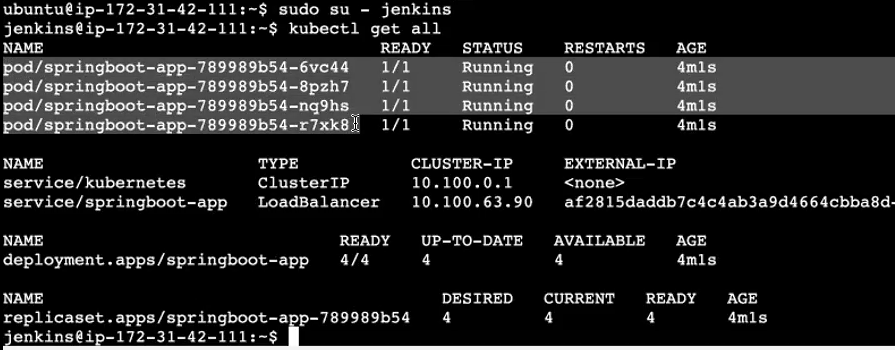
**We write the Jenkins file in github only.in Jenkins page it’s not recommended .so select pipeline script from scm[source code management]**





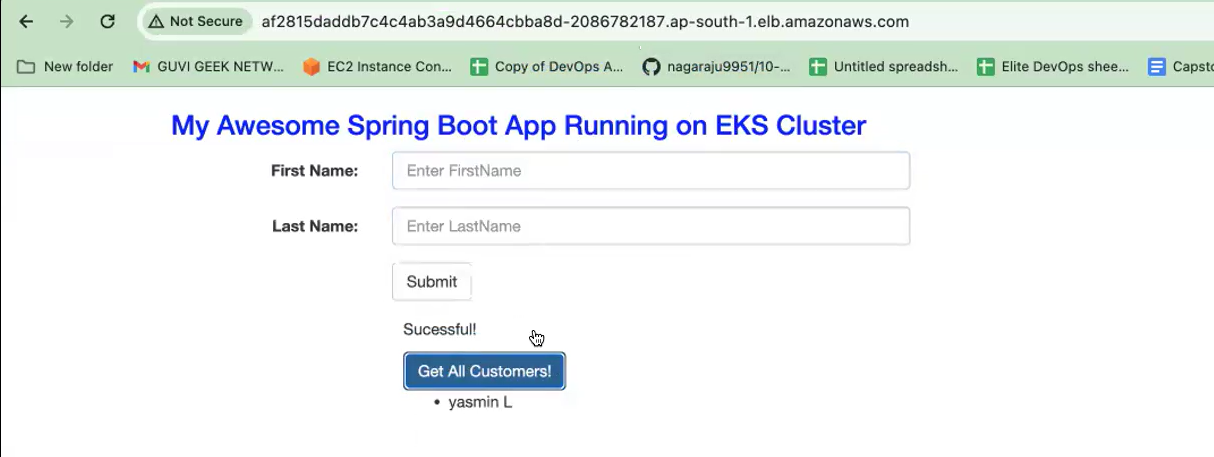








**Copy the load balancer ip and paste it in browser**

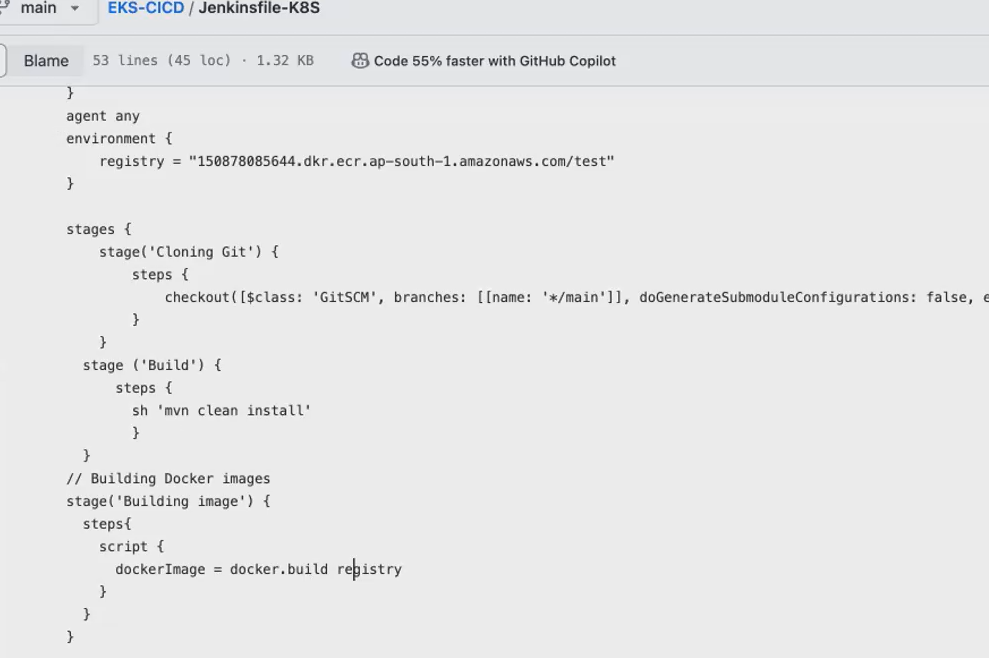


**In Jenkins file:**

**Instead of giving docker build -t my docker image name I have given as docker.dot build.**

**So docker.build is nothing but a docker plugin.**

**Okay, you install docker pipeline. if you have installed docker pipeline, you can simply give docker plugins will be enabled in your Jenkins. You can simply use this function. Docker.build what will happen? It will by default it will execute your docker build command.**

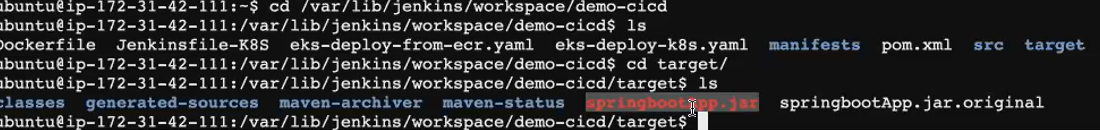


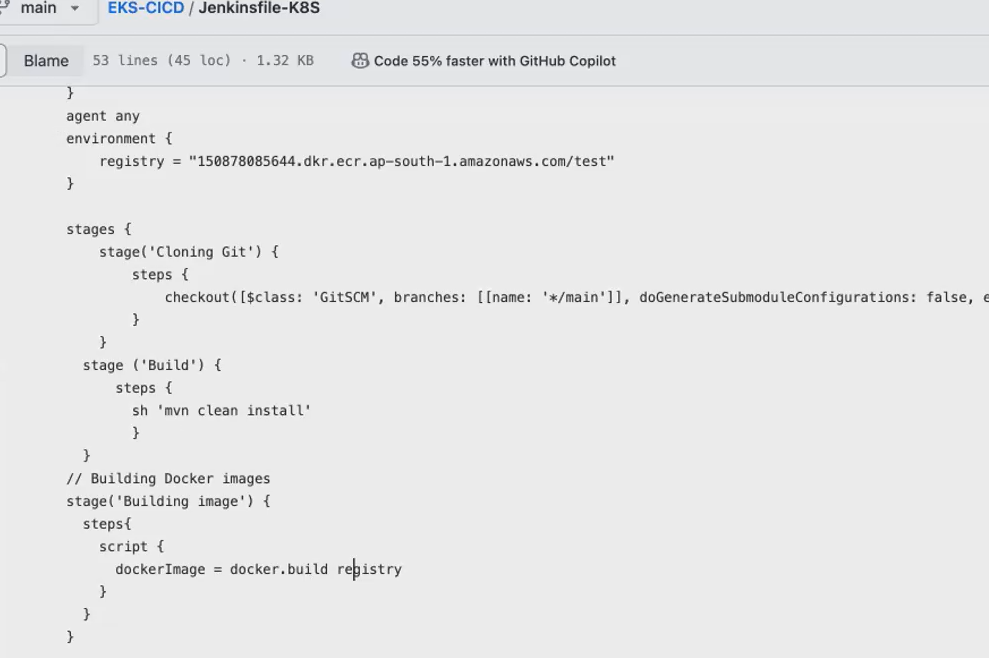


* **Mvn claen - mvn clean package command. Okay, what the mvn clean will do. It will remove the target folder. It will remove all the jar file. That is that it got created previously, and it will freshly create the artifact.**
* **So now, this artifact will be inside of Ec2.**



**Everything done in this work folder.**





**Repo Link -**

[**https://github.com/yasminjeelani/EKS-CICD.git**](https://github.com/yasminjeelani/EKS-CICD.git)

**repo link for full devops course—**

[**https://github.com/zen-class/zen-class-devops-documentation**](https://github.com/zen-class/zen-class-devops-documentation)

**documentation for this application by Yasmin--**

[**https://docs.google.com/document/d/1Is4h94KVFliaNxSuBZX9Spui98CYiFOIawjLfYa3wBU/edit**](https://docs.google.com/document/d/1Is4h94KVFliaNxSuBZX9Spui98CYiFOIawjLfYa3wBU/edit)