End-to-End CI/CD Pipeline Documentation

This document provides a step-by-step guide to creating a CI/CD pipeline using Jenkins, GitHub, Maven, Docker, Amazon ECR, and Kubernetes. The pipeline will automate the process of building, testing, and deploying a sample Java web application.

PROBLEM STATEMENT

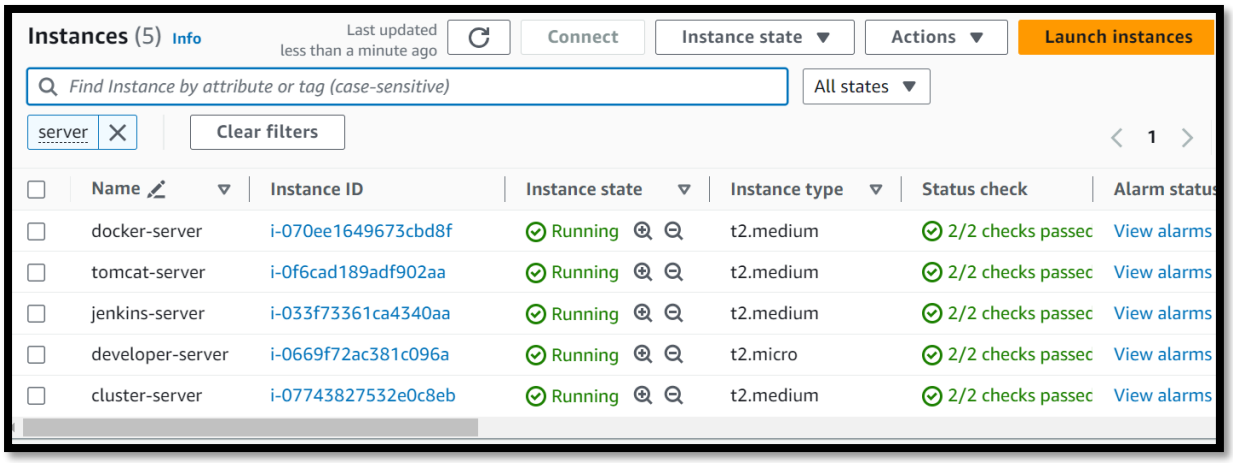
Create an end-to-end CI/CD pipeline in AWS platform using Jenkins as the orchestration tool, Github as the SCM, Maven as the Build tool, deploy in a docker instance and create a Docker image, Store the docker image in ECR, Achieve Kubernetes deployment using the ECR image. Build a sample java web app using maven.

Architecture Overview

The CI/CD pipeline consists of the following stages:

1. **Source Code Management**: Java Code is stored in a GitHub repository.
2. **Build**: Jenkins uses Maven plugin to build the Java application on tomcat server.
3. **Dockerization**: The application is packaged into a Docker image.
4. **Image Storage**: The Docker image is stored in Amazon ECR.
5. **Deployment**: The Docker image is deployed to a Kubernetes cluster.

Creating 5 instances in aws ec2 with same security group, same key-pair and same region.

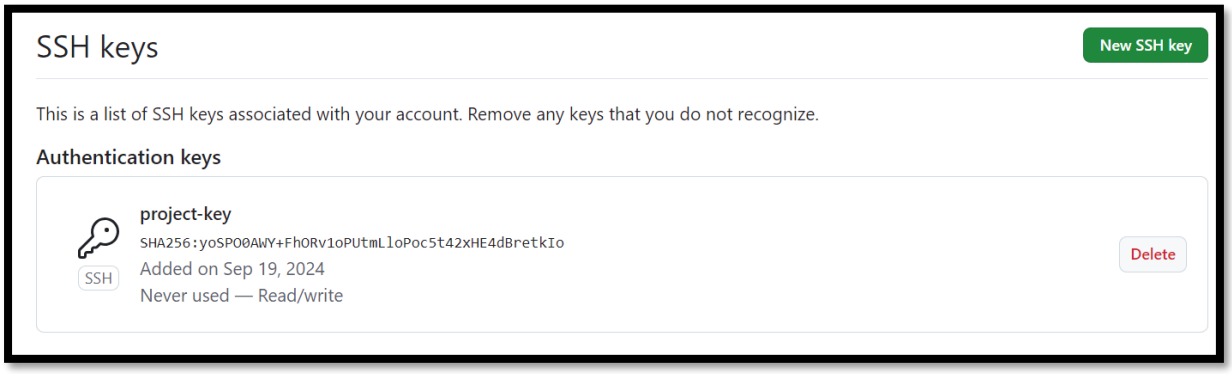


1.Source Code Management: Installing git in developer-server for local version control system.

```
[root@ip-172-31-16-30 data]# yum install git -y
Last metadata expiration check: 0:11:49 ago on Thu Sep 19 04:08:15 2024.
Dependencies resolved.
================================================================================
 Package                    Architecture        Version                    Repository
================================================================================
Installing:
 git                        x86_64              2.40.1-1.amzn2023.0.3      amazonlinux
Installing dependencies:
 git-core                   x86_64              2.40.1-1.amzn2023.0.3      amazonlinux
 git-core-doc               noarch              2.40.1-1.amzn2023.0.3      amazonlinux
 perl-Error                 noarch              1:0.17029-5.amzn2023.0.2   amazonlinux
```

Generating ssh-key to connect local server to our github account.

```
[root@ip-172-31-16-30 data]# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:yoSPO0AWY+FhORv1oPUtmLloPoc5t42xHE4dBretkIo root@ip-172-
The key's randomart image is:
+---[RSA 3072]----+
|  +++            |
| oB+ B .         |
| .o*= = .        |
|  +. * +         |
| oo = = S        |
| +.+ O +         |
|E B.B *          |
|   O.O           |
|    B..          |
+----[SHA256]-----+
```



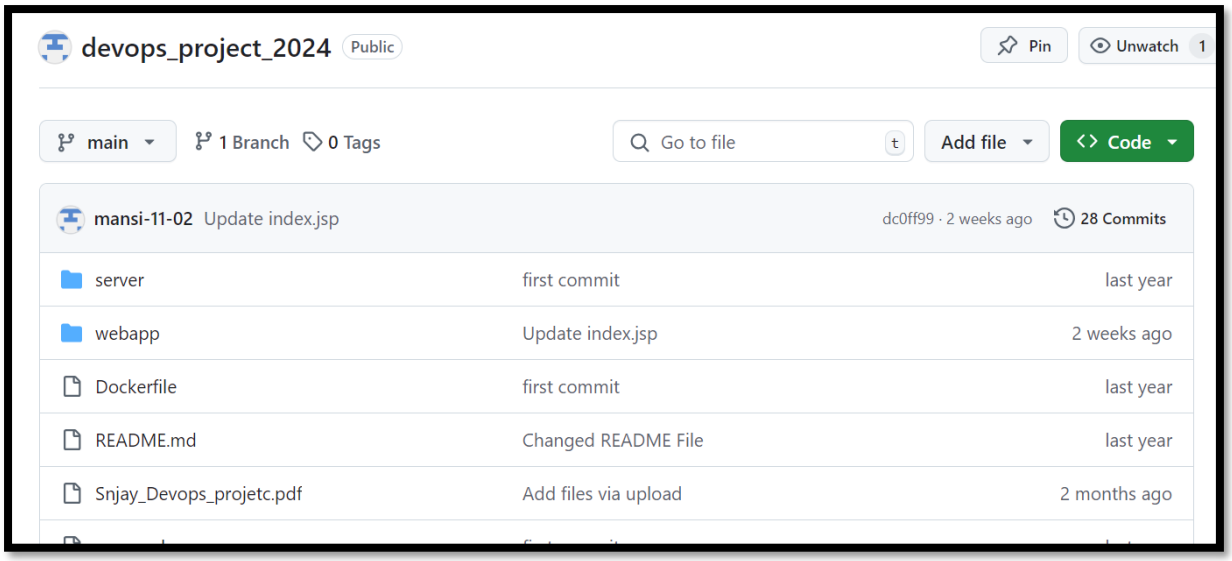
Cloning java project files on local server

```
[root@ip-172-31-22-170 ~]# git clone https://github.com/sanjayguruji/ci-cd-k8s-project.gi
Cloning into 'ci-cd-k8s-project'...
remote: Enumerating objects: 207, done.
remote: Counting objects: 100% (207/207), done.
remote: Compressing objects: 100% (89/89), done.
remote: Total 207 (delta 83), reused 189 (delta 77), pack-reused 0 (from 0)
Receiving objects: 100% (207/207), 38.11 KiB | 5.44 MiB/s, done.
Resolving deltas: 100% (83/83), done.
[root@ip-172-31-22-170 ~]# ls
ci-cd-k8s-project
```

The files are added, commited and pushed in the github Repository by creating remote origin on main branch.

```
   11  git add .
   12  git commit
   13  git remote -v
   14  git branch -M main
   15  git branch
   16  git remote add origin git@github.com:mansi-11-02/devops_project_2024.git
   17  git push origin main
   18  git remote add origin git@github.com:mansi-11-02/devops_project_2024.git
   19  git push origin main
   20  history
[root@ip-172-31-19-163 devops-project]# git remote remove origin
[root@ip-172-31-19-163 devops-project]# git remote add origin git@github.com:mansi-11-02/devops
[root@ip-172-31-19-163 devops-project]# git push origin main
The authenticity of host 'github.com (20.205.243.166)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvCOqU.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
Enumerating objects: 143, done.
Counting objects: 100% (143/143), done.
Compressing objects: 100% (60/60), done.
Writing objects: 100% (143/143), 12.41 MiB | 2.41 MiB/s, done.
Total 143 (delta 39), reused 143 (delta 39), pack-reused 0
remote: Resolving deltas: 100% (39/39), done.
To github.com:mansi-11-02/devops_project_2024.git
 * [new branch]      main -> main
```

The files are successfully pushed in the git repository.



2. Build: Jenkins uses Maven plugin to build the Java application on tomcat server.

Download java, Jenkins and maven plugin on Jenkins-server. Starting Jenkins

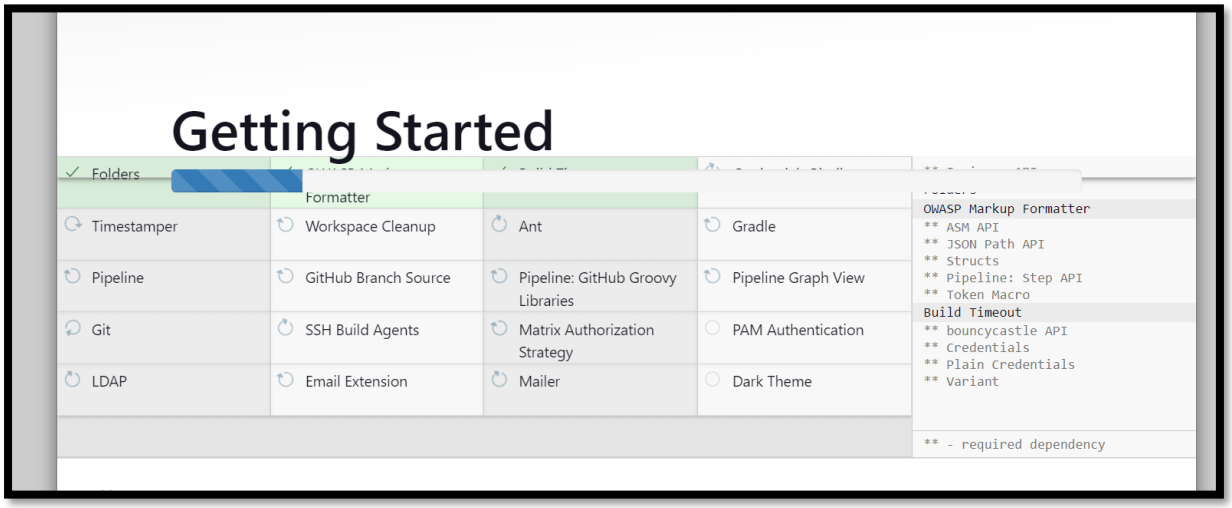
```
dnf install java-17-amazon-corretto -y
wget -O /etc/yum.repos.d/jenkins.repo \https://pkg.jenkins.io/redhat-stable/jenkins.repo
rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key
dnf install jenkins -y
yum install maven -y
yum install git -y
systemctl enable jenkins
systemctl start jenkins
cat /var/lib/jenkins/secrets/initialAdminPassword
history
```

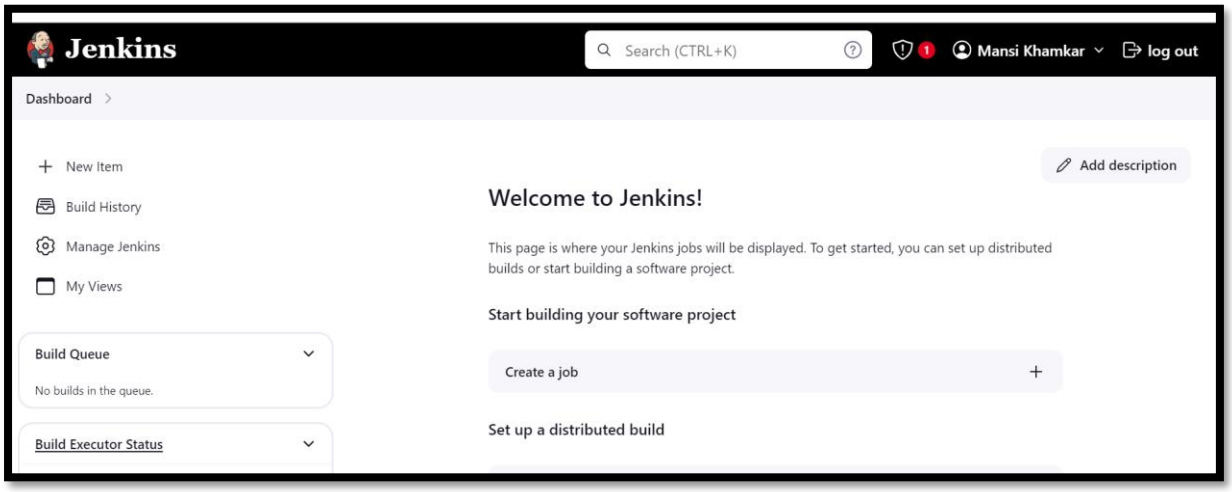
```
Installed:
  git-2.40.1-1.amzn2023.0.3.x86_64                          git-core-2.40.1-1.amzn2023.0.3.x86_64
  git-core-doc-2.40.1-1.amzn2023.0.3.noarch                 perl-Error-1:0.17029-5.amzn2023.0.2.noarch
  perl-File-Find-1.37-477.amzn2023.0.6.noarch               perl-Git-2.40.1-1.amzn2023.0.3.noarch
  perl-TermReadKey-2.38-9.amzn2023.0.2.x86_64               perl-lib-0.65-477.amzn2023.0.6.x86_64

Complete!
[root@ip-172-31-31-36 opt]# systemctl enable jenkins
Created symlink /etc/systemd/system/multi-user.target.wants/jenkins.service → /usr/lib/systemd/system/jen
[root@ip-172-31-31-36 opt]# systemctl start jenkins
```

Copy the public IP of your Jenkins server, allow port 8080 in the security group, and access it via http://<public-ip>:8080 in your web browser

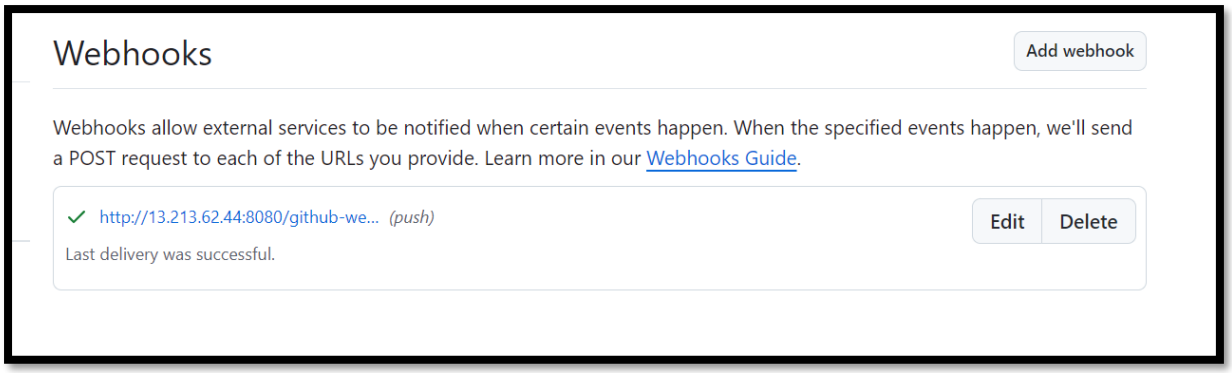
Installing default packages.



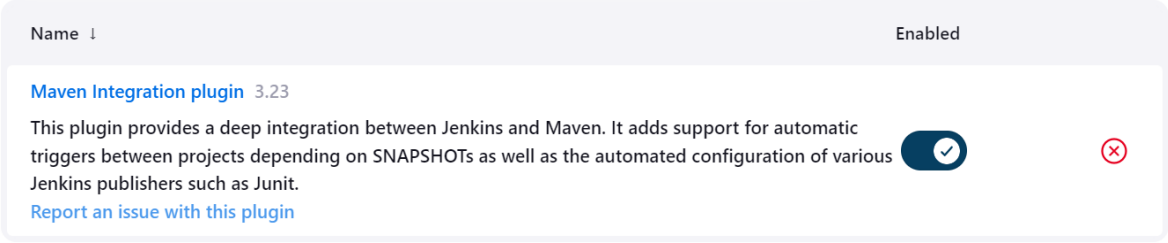


Connecting Jenkins with github by creating webhook in github repository settings.

In webhook the secret is pasted from Jenkins by generating tokens.

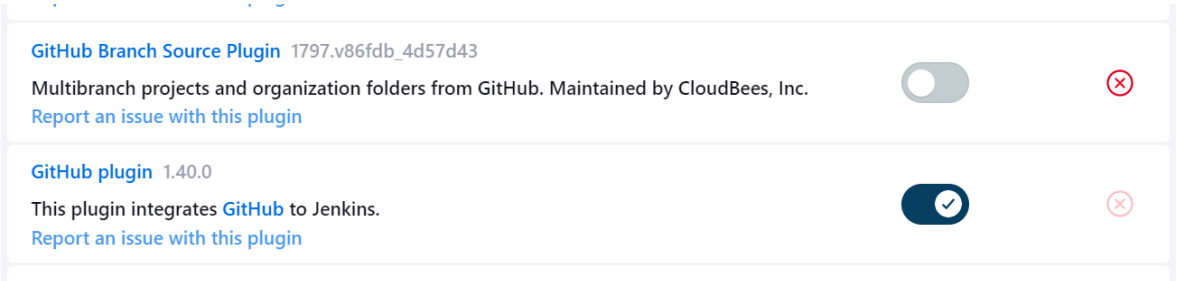


In Jenkins> dashboard > manage Jenkins > available plugins > maven integration >install



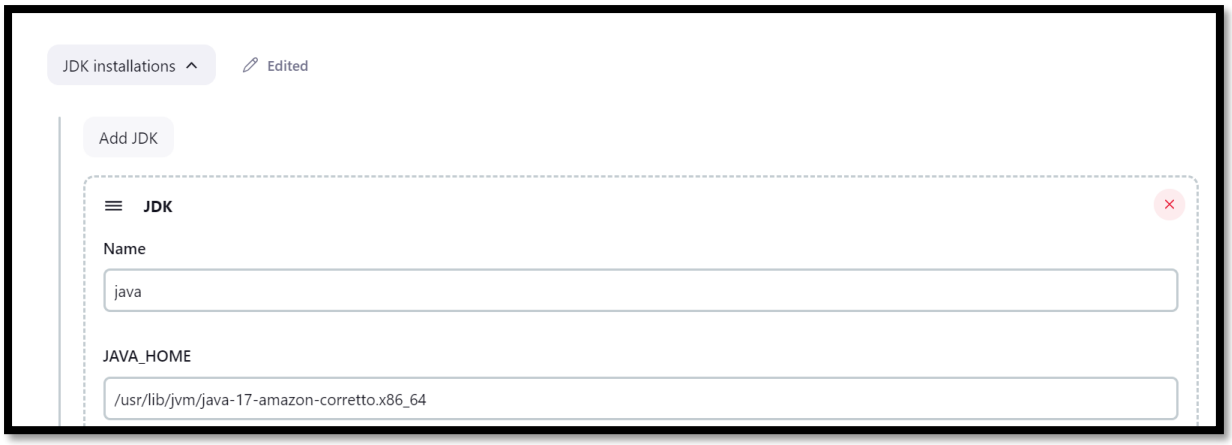
In installed plugins > type github > disable github branch source plugin and enable github plugin.

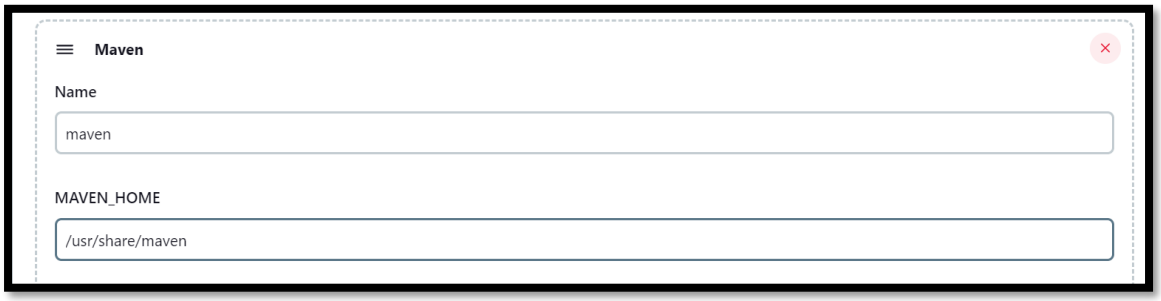
After installing restart Jenkins



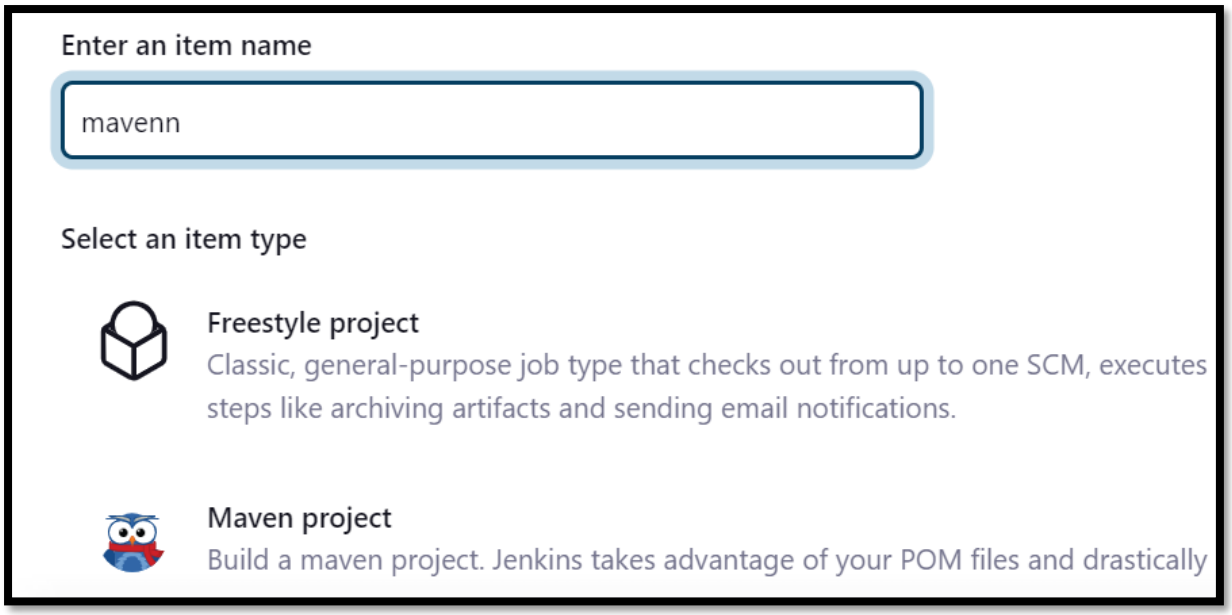
In Manage Jenkins add tools and paste the java and maven path from Jenkins-server

```
[root@ip-172-31-25-56 ~]# mvn -v
Apache Maven 3.8.4 (Red Hat 3.8.4-3.amzn2023.0.5)
Maven home: /usr/share/maven
Java version: 17.0.12, vendor: Amazon.com Inc., runtime: /usr/lib/jvm/java-17
Default locale: en, platform encoding: UTF-8
OS name: "linux", version: "6.1.109-118.189.amzn2023.x86_64", arch: "amd64",
```





Create new item > type name > maven project



Add your git repository link

Change branch to main.

Save, apply, and build now > The build should be successful.

Jenkins artifacts are visible, it means the build outputs or results are accessible.

```
[root@ip-172-31-25-56 ~]# cd /var/lib/jenkins/workspace/maven
[root@ip-172-31-25-56 maven]# ls
Dockerfile  Jenkinsfile  README.md  pom.xml  server  webapp
[root@ip-172-31-25-56 maven]# cd webapp/target
[root@ip-172-31-25-56 target]# ls
maven-archiver  surefire  webapp  webapp.war
```

Open tomcat-server in the terminal

Install java and apache tomcat in /opt directory

```
[root@ip-172-31-30-75 opt]# wget https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.95/bin/apache-tomcat-9.0.95.tar.gz
--2024-09-20 09:42:53--  https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.95/bin/apache-tomcat-9.0.95.tar.gz
Resolving dlcdn.apache.org (dlcdn.apache.org)... 151.101.2.132, 2a04:4e42::644
Connecting to dlcdn.apache.org (dlcdn.apache.org)|151.101.2.132|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 12715996 (12M) [application/x-gzip]
Saving to: 'apache-tomcat-9.0.95.tar.gz'

apache-tomcat-9.0.95.tar.gz   100%[===================================================>]  12.13M  67.0MB/s    in 0.2s

2024-09-20 09:42:53 (67.0 MB/s) - 'apache-tomcat-9.0.95.tar.gz' saved [12715996/12715996]

[root@ip-172-31-30-75 opt]# dnf install java-17-amazon-corretto -y
Last metadata expiration check: 0:21:09 ago on Fri Sep 20 09:22:10 2024.
Dependencies resolved.
================================================================================================================
 Package                                  Architecture    Version                          Repository          Size
================================================================================================================
Installing:
 java-17-amazon-corretto                  x86_64          1:17.0.12+7-1.amzn2023.1         amazonlinux         187 k
```

Unzip the apache tomcat folder. Enter in apache-tomcat folder. Open the bin file and find the path of all context.xml files.

```
[root@ip-172-31-30-75 bin]# find / -name context.xml
/opt/apache-tomcat-9.0.95/conf/context.xml
/opt/apache-tomcat-9.0.95/webapps/docs/META-INF/context.xml
/opt/apache-tomcat-9.0.95/webapps/examples/META-INF/context.xml
/opt/apache-tomcat-9.0.95/webapps/host-manager/META-INF/context.xml
/opt/apache-tomcat-9.0.95/webapps/manager/META-INF/context.xml
```

Change the last two files: comment valve statement in both the files <!-- -->

```
<Context antiResourceLocking="false" privileged="true" >
  <CookieProcessor className="org.apache.tomcat.util.http.Rfc6265CookieProcessor"
                   sameSiteCookies="strict" />
  <!--  <Valve className="org.apache.catalina.valves.RemoteAddrValve"
  allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1" />-->
  <Manager sessionAttributeValueClassNameFilter="java\.lang\.(?:Boolean|Integer|L
srfPreventionFilter\$LruCache(?:\$1)?|java\.util\.(?:Linked)?HashMap"/>
```

Open the conf/ tomcat-users.xml file and add users:

```
<role rolename="manager-gui"/>
<role rolename="manager-script"/>
<role rolename="manager-jmx"/>
<role rolename="manager-status"/>
<user username="admin" password="admin" roles="manager-gui, manager-script, manager-jmx, manager-status"/>
<user username="developer" password="developer" roles="manager-script"/>
<user username="tomcat" password="s3cret" roles="manager-gui"/>
</tomcat-users>
```

Starting tomcat

```
[root@ip-172-31-30-75 bin]# ./startup.sh
Using CATALINA_BASE:   /opt/apache-tomcat-9.0.95
Using CATALINA_HOME:   /opt/apache-tomcat-9.0.95
Using CATALINA_TMPDIR: /opt/apache-tomcat-9.0.95/temp
Using JRE_HOME:        /usr
Using CLASSPATH:       /opt/apache-tomcat-9.0.95/bin/bootstrap.ja
Using CATALINA_OPTS:
Tomcat started.
```

Go to Jenkins in web browser:

To integrate tomat server with Jenkins:

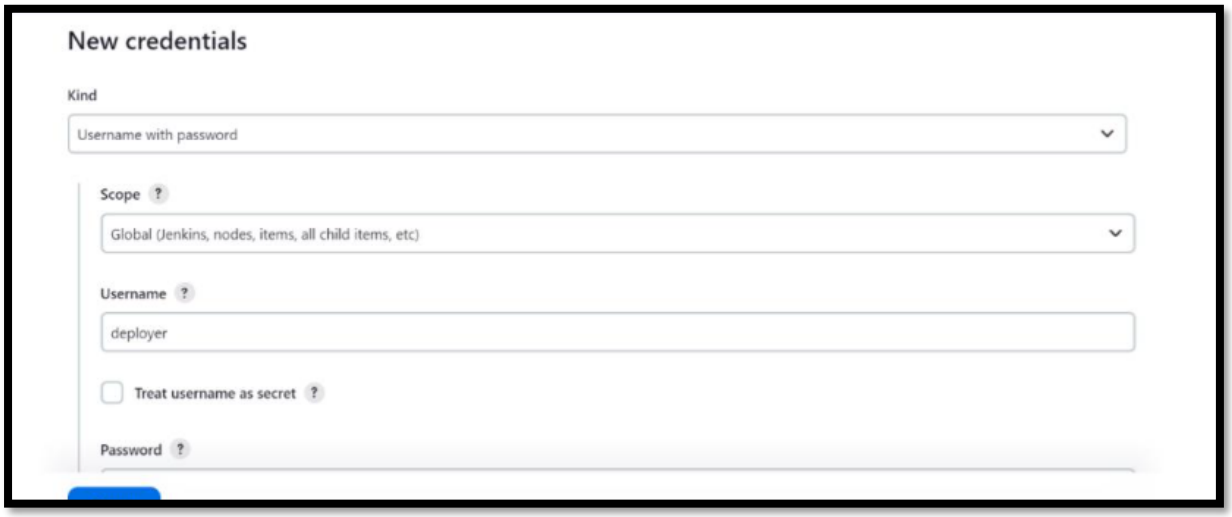
Go to Manage Jenkins > plugin > available plugins > deploy to container > install

Manage Jenkins > Credentials > system > global credentials > add

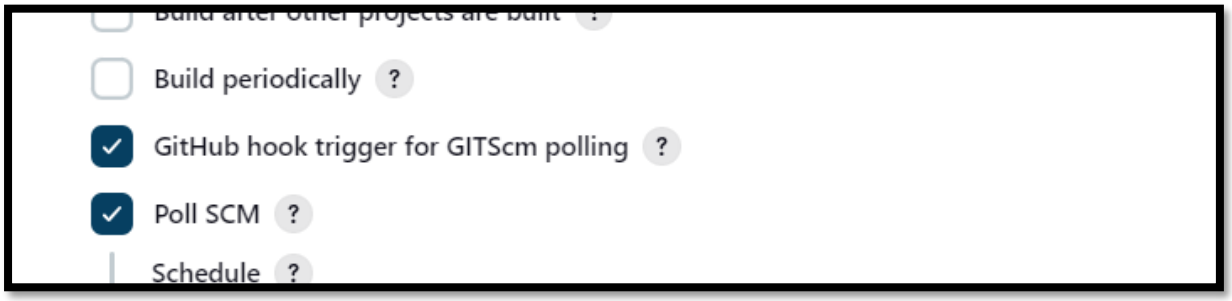
Username: developer

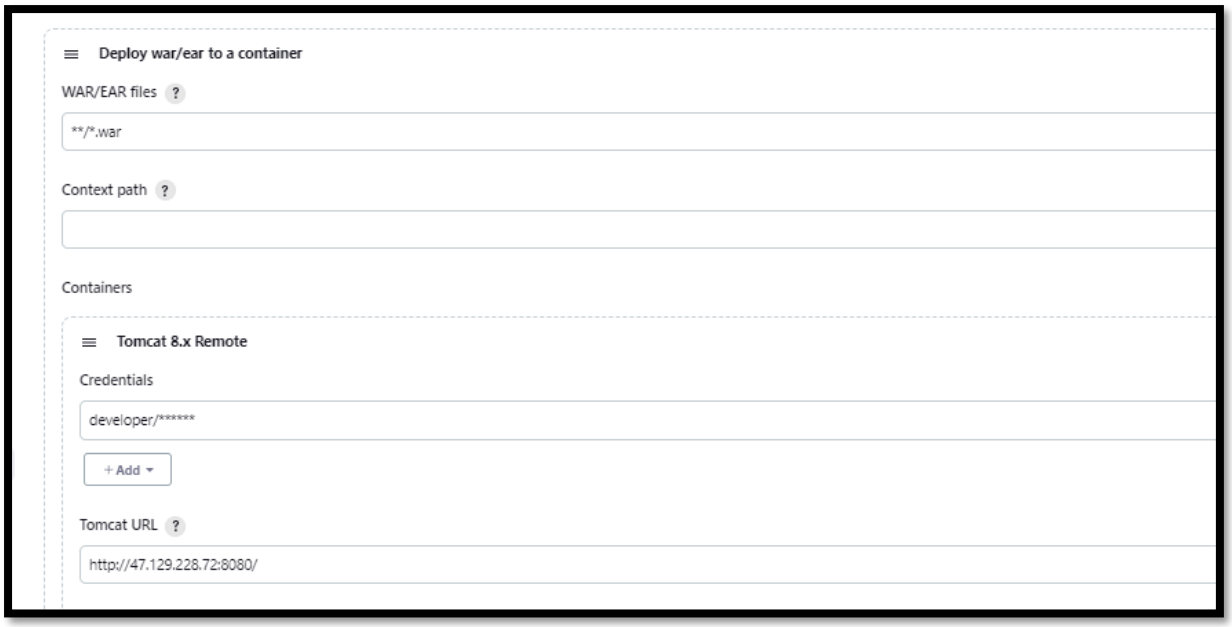
Password: developer

Id: tomcat-cred

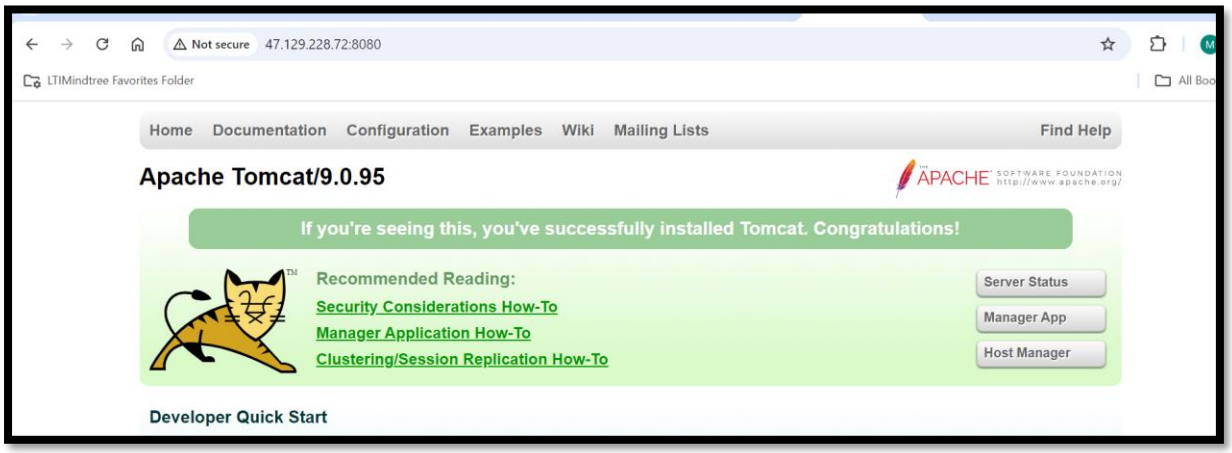


Create new item > type name > select maven project > add git repository link > change branch to main > Build trigger > Go to add post built action > deploy ear/war to a container > Save and apply > Build now

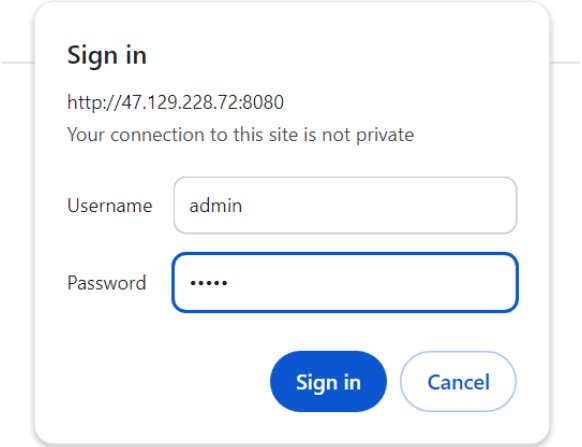




Open public ip of tomcat-server on web browser on port 8080



Go to manager app , add username: admin and password : admin



Open /webapp

/manager	*None specified*	Tomcat Manager Application	true	1
/webapp	*None specified*	Webapp	true	0

Your web application should open.

3.Dockerization: The application is packaged into a Docker image and **Image Storage**: The Docker image is stored in Amazon ECR.

```
[root@docker ~]# yum install -y yum-utils
Last metadata expiration check: 0:34:01 ago on Fri Sep 20 09:22:26 2024.
Package dnf-utils-4.3.0-13.amzn2023.0.4.noarch is already installed.
Dependencies resolved.
Nothing to do.
Complete!
[root@docker ~]# yum-config-manager --add-repo https://download.docker.com/linux/centos/docke
Adding repo from: https://download.docker.com/linux/centos/docker-ce.repo
[root@docker ~]# yum install docker -y
Docker CE Stable - x86_64
```

Set password: passwd root

Generate ssh-key: ssh-keygen

Open /etc/ssh/sshd_config file and make the following changes:

PermitRoot login yes

Uncomment Pubkey authenticatiom yes

Password authentication yes

Permit empty yes

Esc > :wq! > enter

Run systemctl restart sshd, systemctl enable sshd

Configure aws iam user:

Policy name	Type	Attached via
AdministratorAccess	AWS managed - job function	Directly
AmazonEC2FullAccess	AWS managed	Directly
AmazonRoute53FullAccess	AWS managed	Directly
AmazonS3FullAccess	AWS managed	Directly
IAMFullAccess	AWS managed	Directly

Copy jenkins public ssh key in docker

```
       valid_lft forever preferred_lft forever
[root@docker .ssh]# systemctl start sshd
[root@docker .ssh]# systemctl enable sshd
[root@docker .ssh]# ssh-copy-id root@172.31.25.56
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/root/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are a
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to in
root@172.31.25.56's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'root@172.31.25.56'"
and check to make sure that only the key(s) you wanted were added.
```

Copy docker public ssh key in jenkins

Perform all the same steps in jenkins-server from set password. There will be one more step in jenkins server of connecting jenkins-server with jenkins. For that copy public ssh key of jenkins-server in jenkins-server itself.

Go to aws.com and create a repository in Elastic Container registry

Amazon ECR > Private registry > Repositories

Private repositories

Create repository

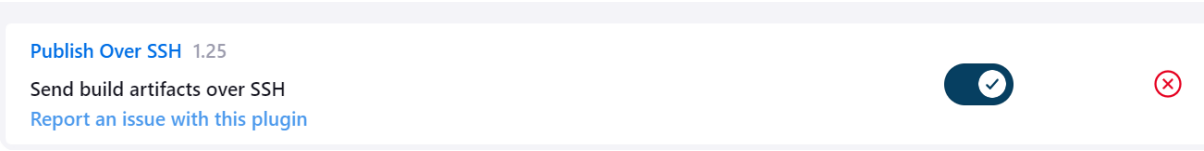
Repositories (1)

View push commands | Delete | Actions

Search by repository substring

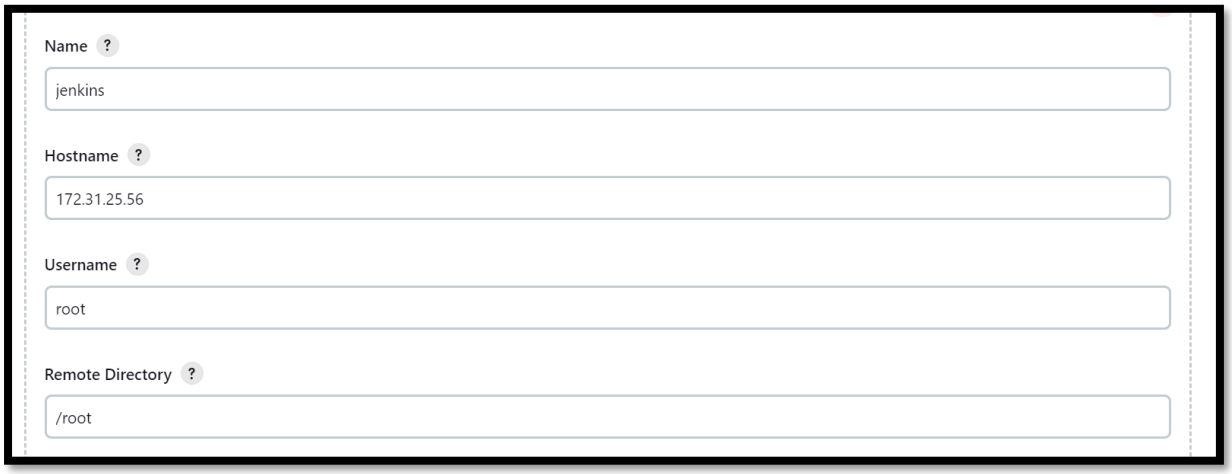
Repository name	URI	Created at	Tag immutability	Encryption type
project-11	122610503177.dkr.ecr.ap-southeast-1.amazonaws.com/project-11	September 19, 2024, 16:36:24 (UTC+05.5)	Mutable	AES-256

Go to jenkins > Manage jenkins > plugins > install **publish over ssh**

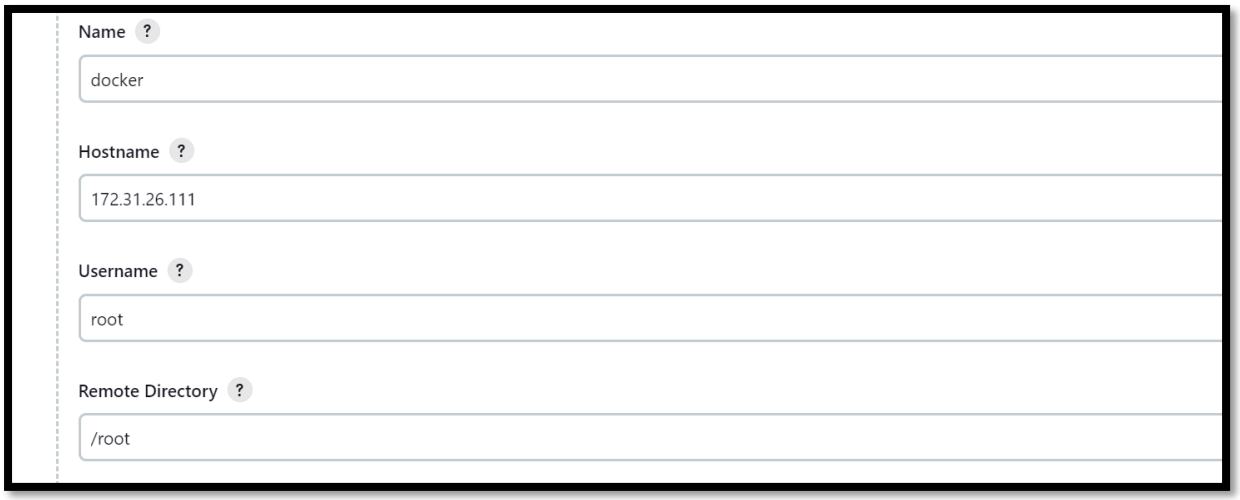


Manage jenkins > system > Go to publish over ssh tab and paste jenkins private key

Add ssh-server and paste jenkins ip by running ip a s command in jenkins-server > test configuration. It should display Success.



Add ssh-server and paste docker ip by running ip a s command in docker-server > test configuration. It should display Success.



Apply > save

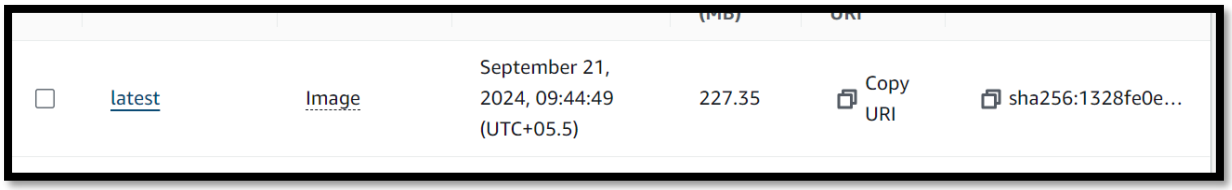
Go to your tomcat project in Jenkins > Configure > add post-built action > send artifacts over ssh:

The exec command is to synchronize files and directories from your local machine to the directory on the remote machine.

In exec commands we have enter the commands to create a docker image of the built maven project and push the image in aws ecr.

Apply > save > build Now

If build is successful image will be created and you can check it in your aws ecr repository.



4. **Deployment**: The Docker image is deployed to a Kubernetes cluster.

Create IAM roles in aws and attach policy with instance :

Policy name	Type	Attached entities
AmazonEKSClusterPolicy	AWS managed	2
AmazonElasticContainer...	AWS managed	1
AWSCloudFormationFull...	AWS managed	1
IAMFullAccess	AWS managed	2

Open cluster-server in terminal.

Set password: passwd root

Generate ssh-key: ssh-keygen

Open /etc/ssh/sshd_config file and make the following changes:

PermitRoot login yes

Uncomment Pubkey authenticatiom yes

Password authentication yes

Permit empty yes

Esc > :wq! > enter

Install eksctl and kubectl in cluster server

```
[root@kubernetes .ssh]# cd
[root@kubernetes ~]# curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(un
ame -s)_amd64.tar.gz" | tar xz -C /tmp
 sudo mv /tmp/eksctl /usr/local/bin
 eksctl version
0.190.0
[root@kubernetes ~]# curl -LO https://storage.googleapis.com/kubernetes-release/release/$(curl -s https://storage.google
apis.com/kubernetes-release/release/stable.txt)/bin/linux/amd64/kubectl
 sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
 kubectl version --client
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 53.7M  100 53.7M    0     0  10.4M      0  0:00:05  0:00:05 --:--:-- 13.3M
Client Version: v1.31.0
Kustomize Version: v5.4.2
```

Create cluster and nodegroup

```
[root@kubernetes ~]# eksctl create cluster --name cluster2 --region ap-south-1 --version 1.29 --vpc-public-subnets subne
t-0241e4185ea1b6efc,subnet-027ef83b728e2e522 --without-nodegroup
2024-09-20 12:34:24 [ℹ]  eksctl version 0.190.0
2024-09-20 12:34:24 [ℹ]  using region ap-south-1
2024-09-20 12:34:25 [✔]  using existing VPC (vpc-00ff47d27a0912365) and subnets (private:map[] public:map[ap-south-1a:{s
ubnet-0241e4185ea1b6efc ap-south-1a 172.31.32.0/20 0 } ap-south-1b:{subnet-027ef83b728e2e522 ap-south-1b 172.31.0.0/20 0
 }])
2024-09-20 12:34:25 [!]  custom VPC/subnets will be used; if resulting cluster doesn't function as expected, make sure t
o review the configuration of VPC/subnets
2024-09-20 12:34:25 [ℹ]  using Kubernetes version 1.29
2024-09-20 12:34:25 [ℹ]  creating EKS cluster "cluster2" in "ap-south-1" region with
2024-09-20 12:34:25 [ℹ]  if you encounter any issues, check CloudFormation console or try 'eksctl utils describe-stacks
--region=ap-south-1 --cluster=cluster2'
2024-09-20 12:34:25 [ℹ]  Kubernetes API endpoint access will use default of {publicAccess=true, privateAccess=false} for
 cluster "cluster2" in "ap-south-1"
2024-09-20 12:34:25 [ℹ]  CloudWatch logging will not be enabled for cluster "cluster2" in "ap-south-1"
2024-09-20 12:34:25 [ℹ]  you can enable it with 'eksctl utils update-cluster-logging --enable-types={SPECIFY-YOUR-LOG-TY
PES-HERE (e.g. all)} --region=ap-south-1 --cluster=cluster2'
2024-09-20 12:34:25 [ℹ]  default addons vpc-cni, kube-proxy, coredns were not specified, will install them as EKS addons
2024-09-20 12:34:25 [ℹ]
2 sequential tasks: { create cluster control plane "cluster2",
    2 sequential sub-tasks: {
        1 task: { create addons },
        wait for control plane to become ready,
    }
}
2024-09-20 12:34:25 [ℹ]  building cluster stack "eksctl-cluster2-cluster"
2024-09-20 12:34:25 [ℹ]  deploying stack "eksctl-cluster2-cluster"
2024-09-20 12:34:55 [ℹ]  waiting for CloudFormation stack "eksctl-cluster2-cluster"
2024-09-20 12:35:25 [ℹ]  waiting for CloudFormation stack "eksctl-cluster2-cluster"
```

```
2024-09-20 12:43:28 [✔]  all EKS cluster resources for "cluster2" have been created
2024-09-20 12:43:28 [✔]  created 0 nodegroup(s) in cluster "cluster2"
2024-09-20 12:43:28 [✔]  created 0 managed nodegroup(s) in cluster "cluster2"
2024-09-20 12:43:29 [ℹ]  kubectl command should work with "/root/.kube/config", try 'kubectl get nodes'
2024-09-20 12:43:29 [✔]  EKS cluster "cluster2" in "ap-south-1" region is ready
[root@kubernetes ~]# eksctl create nodegroup \
  --cluster cluster2 \
  --region ap-south-1 \
  --name my-node-group \
  --node-ami-family Ubuntu2004 \
  --node-type t2.small \
  --subnet-ids subnet-0241e4185ea1b6efc,subnet-027ef83b728e2e522 \
  --nodes 3 \
  --nodes-min 2 \
  --nodes-max 4 \
  --ssh-access \
  --ssh-public-key /root/.ssh/id_rsa.pub
2024-09-20 12:45:45 [ℹ]  will use version 1.29 for new nodegroup(s) based on control plane version
2024-09-20 12:45:46 [ℹ]  nodegroup "my-node-group" will use "ami-0ab6dcbf35da05038" [Ubuntu2004/1.29]
2024-09-20 12:45:46 [ℹ]  using SSH public key "/root/.ssh/id_rsa.pub" as "eksctl-cluster2-nodegroup-my-node-group-f0:59:
09:16:c5:35:7a:76:72:80:80:ea:a8:4f:c0:3e"
2024-09-20 12:45:46 [ℹ]  1 nodegroup (my-node-group) was included (based on the include/exclude rules)
2024-09-20 12:45:46 [ℹ]  will create a CloudFormation stack for each of 1 managed nodegroups in cluster "cluster2"
2024-09-20 12:45:46 [ℹ]
2 sequential tasks: { fix cluster compatibility, 1 task: { 1 task: { create managed nodegroup "my-node-group" } }
}
2024-09-20 12:45:46 [ℹ]  checking cluster stack for missing resources
2024-09-20 12:45:46 [ℹ]  cluster stack has all required resources
2024-09-20 12:45:46 [ℹ]  building managed nodegroup stack "eksctl-cluster2-nodegroup-my-node-group"
2024-09-20 12:45:47 [ℹ]  deploying stack "eksctl-cluster2-nodegroup-my-node-group"
```

Create one deployment.yml file and service.yml file

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: app-webapp
  labels:
    app: webapp
spec:
  replicas: 2
  selector:
    matchLabels:
      app: webapp
  template:
    metadata:
      labels:
        app: webapp
    spec:
      containers:
      - name: webapp
        image: 122610503177.dkr.ecr.ap-southeast-1.amazonaws.com/project-11:latest
        ports:
        - containerPort: 8080
```

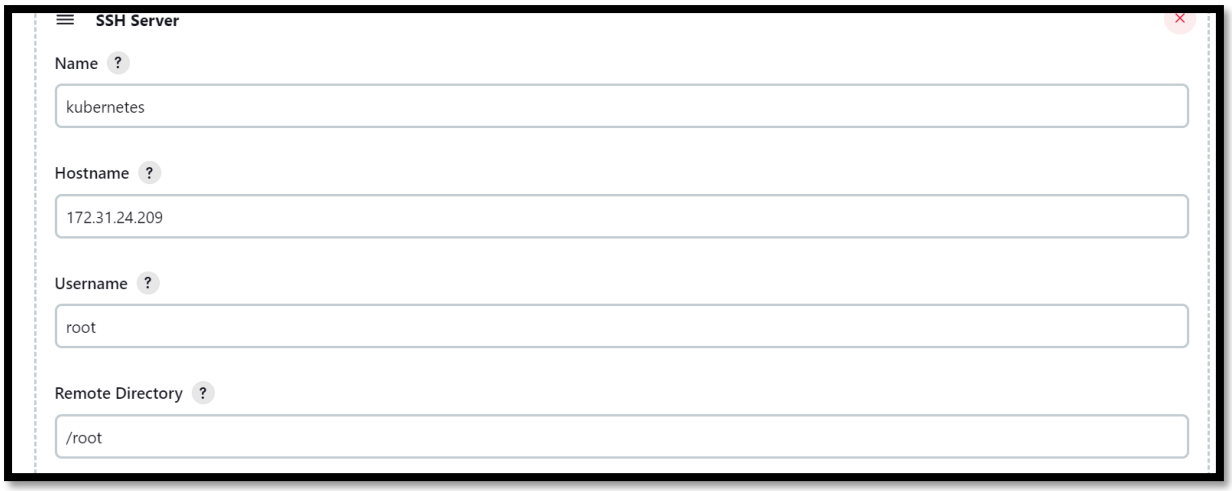
Paste the URI of the image in deployment.yml file.

☐	latest	Image	September 21, 2024, 09:44:49 (UTC+05.5)	227.35	Copy URI	sha256:1328fe0e...

```yaml
apiVersion: v1
kind: Service
metadata:
  name: app-service
  labels:
    app: webapp
spec:
  selector:
    app: webapp
  ports:
    - port: 8080
      targetPort: 8080

  type: LoadBalancer
```

Add ssh-server and paste cluster ip by running ip a s command in docker-server > test configuration. It should display Success.



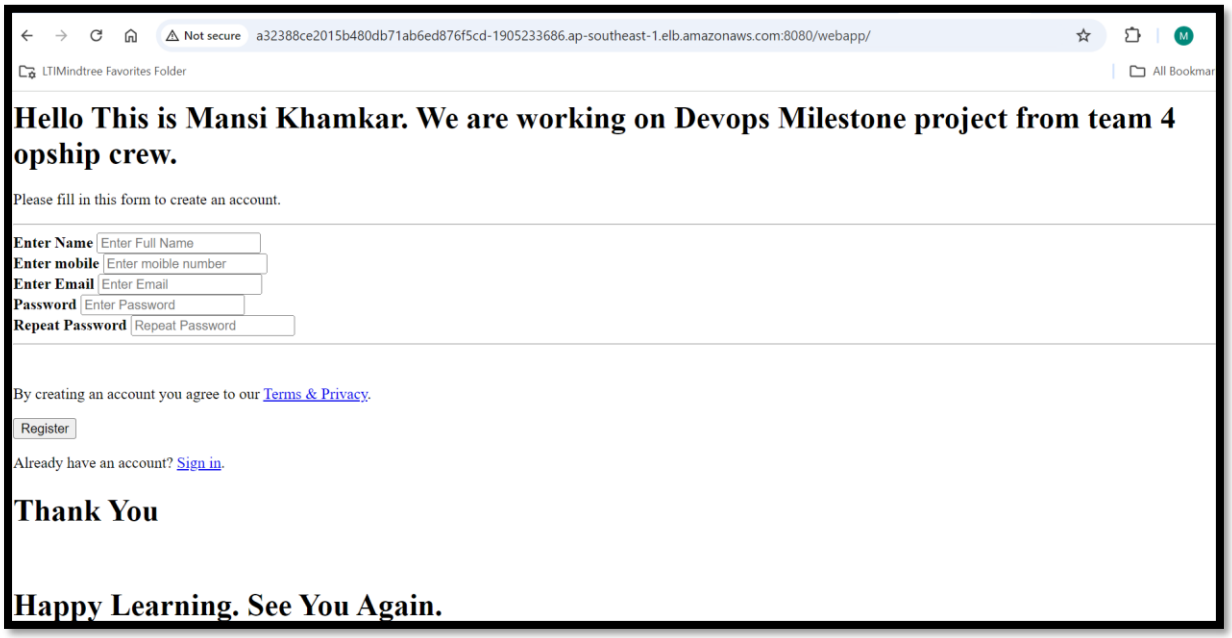
Apply > save

Go to your tomcat project in Jenkins > Configure > add post-built action > send artifacts over ssh:

These commands are to delete previous deployment and create new deploment and service.

In the cluster server run the following command to display current services and vcopy the external ip and run http://external-ip:8080/webapp in web browser to run your java application.

```
[root@ip-172-31-24-209 ~]# kubectl get svc
NAME          TYPE           CLUSTER-IP       EXTERNAL-IP                                                                PORT(S)
          AGE
app-service   LoadBalancer   10.100.229.100   a32388ce2015b480db71ab6ed876f5cd-1905233686.ap-southeast-1.elb.amazonaws.com   8080:309
26/TCP   15h
kubernetes    ClusterIP      10.100.0.1       <none>                                                                     443/TCP
          21h
```



In this way we have automated the whole process. If there are any modifications in the jsp file from github jenkins will automatically build the project and the changes will be reflected on the live webpage automatically.

If there are any errors in the modification the build will be failed and the webpage will not be updated. It will show the last successful built which will help in preventing our webpage from crashing.

-BY MANSI KHAMKAR