

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

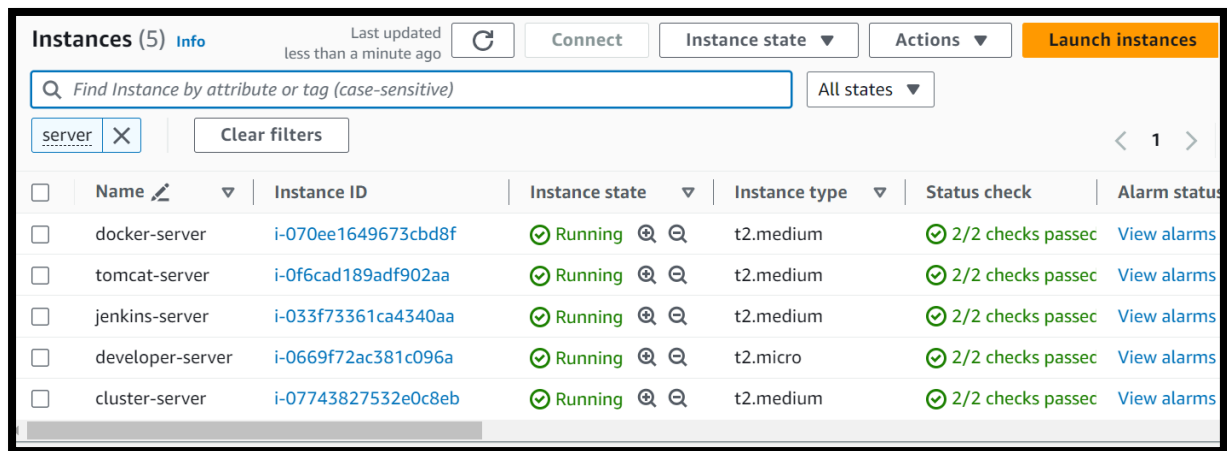
Create an end-to-end CI/CD pipeline in AWS platform using Jenkins as the orchestration tool, Github as the SCM, Maven as the Build tool, deploy in a docker instance and create a Docker image, Store the docker image in ECR, Achieve Kubernetes deployment using the ECR image. Build a sample java web app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



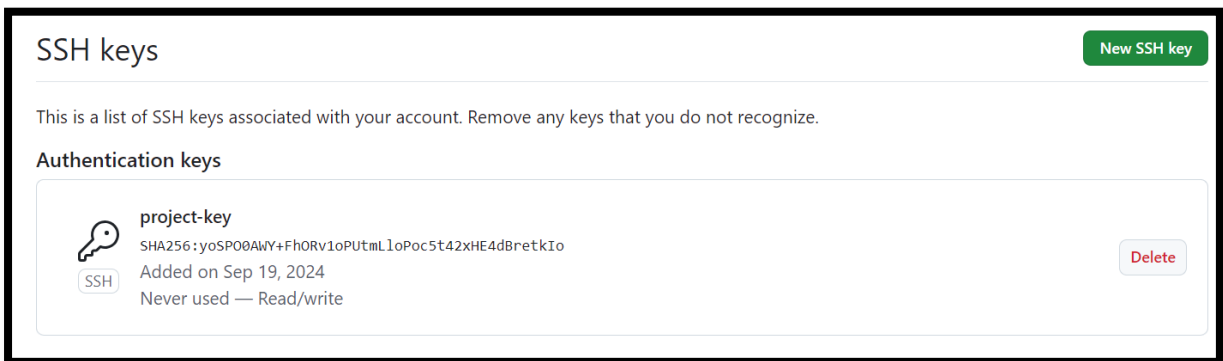
	Name	Instance ID	Instance state	Instance type	Status check	Alarm status
<input type="checkbox"/>	docker-server	i-070ee1649673cbd8f	Running	t2.medium	2/2 checks passed	View alarms
<input type="checkbox"/>	tomcat-server	i-0f6cad189adf902aa	Running	t2.medium	2/2 checks passed	View alarms
<input type="checkbox"/>	jenkins-server	i-033f73361ca4340aa	Running	t2.medium	2/2 checks passed	View alarms
<input type="checkbox"/>	developer-server	i-0669f72ac381c096a	Running	t2.micro	2/2 checks passed	View alarms
<input type="checkbox"/>	cluster-server	i-07743827532e0c8eb	Running	t2.medium	2/2 checks passed	View alarms

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:yoSP00AWY+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]-----+
```



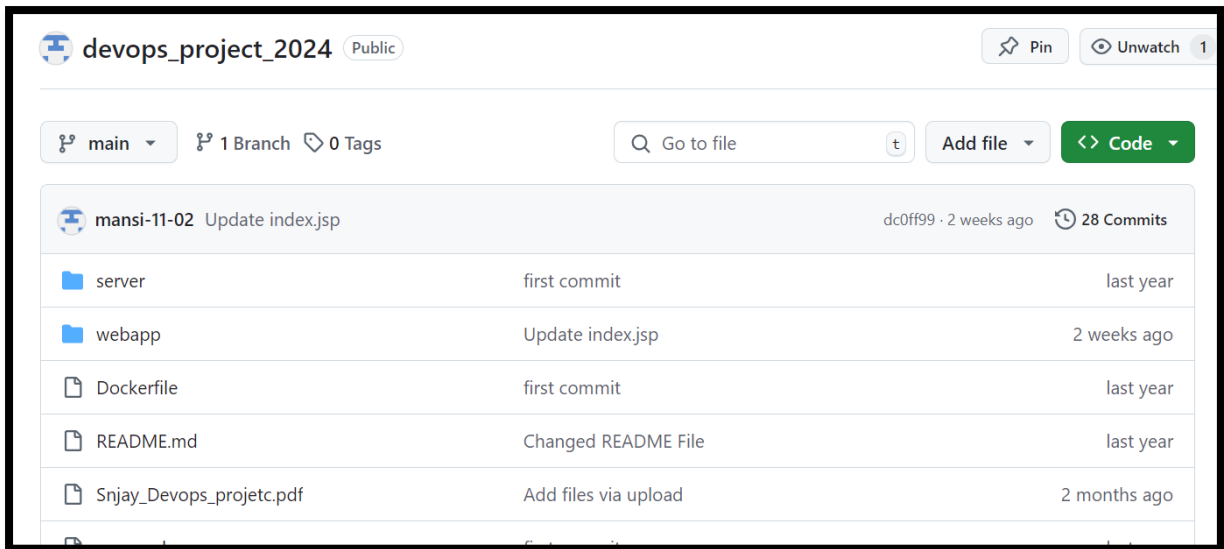
Cloning java project files on local server

```
[root@ip-172-31-22-170 ~]# git clone https://github.com/sanjayguruji/ci-cd-k8s-project.git
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, committed 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

```

1 cd /opt
2 dnf install java-17-amazon-corretto -y
3 wget -O /etc/yum.repos.d/jenkins.repo \https://pkg.jenkins.io/redhat-stable/jenkins.repo
4 rpm --import \https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key
5 dnf install jenkins -y
6 yum install maven -y
7 yum install git -y
8 systemctl enable jenkins
9 systemctl start jenkins
10 cat /var/lib/jenkins/secrets/initialAdminPassword
11 history

```

```

Installed:
git-2.40.1-1.amzn2023.0.3.x86_64
git-core-doc-2.40.1-1.amzn2023.0.3.noarch
perl-File-Find-1.37-477.amzn2023.0.6.noarch
perl-TermReadKey-2.38-9.amzn2023.0.2.x86_64
git-core-2.40.1-1.amzn2023.0.3.x86_64
perl-Error-1:0.17029-5.amzn2023.0.2.noarch
perl-Git-2.40.1-1.amzn2023.0.3.noarch
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/jenkins.service
[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

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

```
/var/lib/jenkins/secrets/initialAdminPassword
```

Please copy the password from either location and paste it below.


Administrator password

Continue

Installing default packages.

Getting Started

✓ Folders	Formatter	Ant	Gradle	OWASP Markup Formatter
⌚ Timestamper	⌚ Workspace Cleanup	⌚ Pipeline: GitHub Groovy Libraries	⌚ Pipeline Graph View	** ASM API
⌚ Pipeline	⌚ GitHub Branch Source	⌚ Matrix Authorization Strategy	⌚ PAM Authentication	** JSON Path API
⌚ Git	⌚ SSH Build Agents	⌚ Mailer	⌚ Dark Theme	** Struts
⌚ LDAP	⌚ Email Extension			** Pipeline: Step API
				** Token Macro
				Build Timeout
				** bouncycastle API
				** Credentials
				** Plain Credentials
				** Variant
				** - required dependency

 **Jenkins**

Search (CTRL+K) ?

🛡️ 1 👤 Mansi Khamkar 🚪 log out

Dashboard >

+ New Item

📁 Build History

⚙️ Manage Jenkins

📌 My Views

Build Queue

No builds in the queue.

Build Executor Status

Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

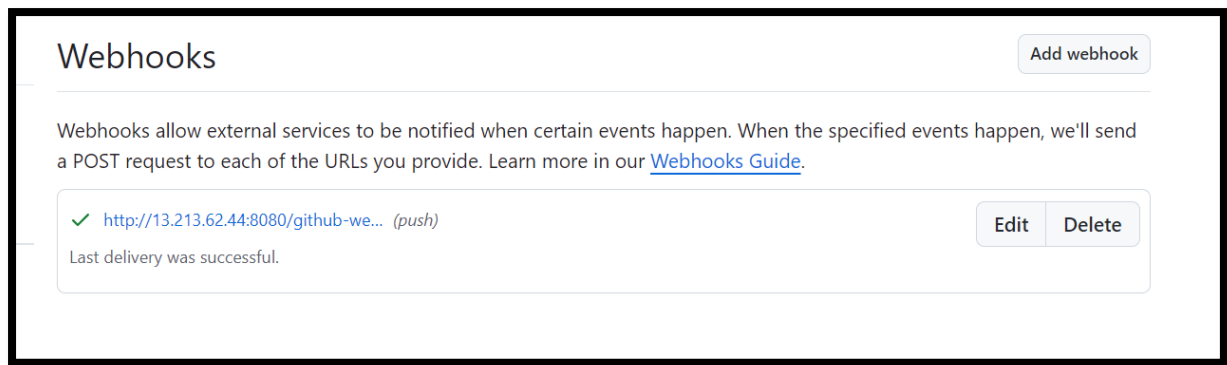
Create a job +

Set up a distributed build

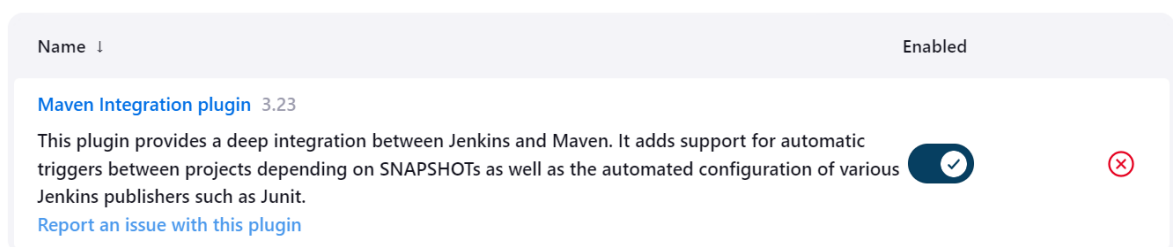
Add description

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

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

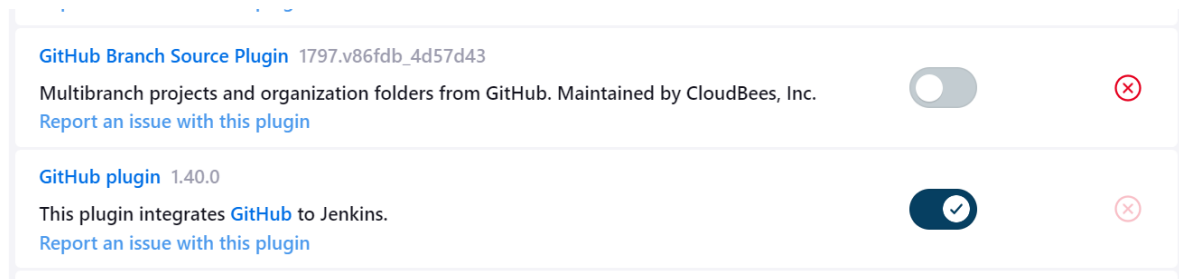


In Jenkins> dashboard > manage Jenkins > available plugins > maven integration >install



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",
```

JDK installations ^ Edited

Add JDK

JDK

Name

java

JAVA_HOME

/usr/lib/jvm/java-17-amazon-corretto.x86_64

Maven

Name

maven

MAVEN_HOME


/usr/share/maven


Create new item > type name > maven project

Enter an item name

mavenn

Select an item type

 **Freestyle project**
Classic, general-purpose job type that checks out from up to one SCM, executes steps like archiving artifacts and sending email notifications.

 **Maven project**
Build a maven project. Jenkins takes advantage of your POM files and drastically

Add your git repository link

Git ?

Repositories ?

Repository URL ? ✕

`https://github.com/mansi-11-02/milestone-project.git`

Credentials ?

- none - ▼

+ Add ▼

Change branch to main.

Branches to build ?

Branch Specifier (blank for 'any') ? ✕

`*/main`

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

```

SNAPSHOT/server-1.0-SNAPSHOT.pom
[JENKINS] Archiving /var/lib/jenkins/workspace/maven/server/target/server.jar to com.example.maven-
project/server/1.0-SNAPSHOT/server-1.0-SNAPSHOT.jar
[JENKINS] Archiving /var/lib/jenkins/workspace/maven/pom.xml to com.example.maven-project/maven-project/1.0-
SNAPSHOT/maven-project-1.0-SNAPSHOT.pom
channel stopped
Finished: SUCCESS

```

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


```

bash: wget: command not found
[root@ip-172-31-30-75 opt]# wget https://dldn.apache.org/tomcat/tomcat-9/v9.0.95/bin/apache-tomcat-9.0.95.tar.gz
--2024-09-20 09:42:53-- https://dldn.apache.org/tomcat/tomcat-9/v9.0.95/bin/apache-tomcat-9.0.95.tar.gz
Resolving dldn.apache.org (dldn.apache.org)... 151.101.2.132, 2a04:4e42::644
Connecting to dldn.apache.org (dldn.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 sessionAttributeClassNameFilter="java\.lang\.(?:Boolean|Integer|L
srfPreventionFilter\$LruCache(?:\$1)?|java\.util\.(?:Linked)?HashMap"/>
</Context>

```

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.jar
Using CATALINA_OPTS:
Tomcat started.
```

Go to Jenkins in web browser:

To integrate tomcat 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



New credentials

Kind: Username with password

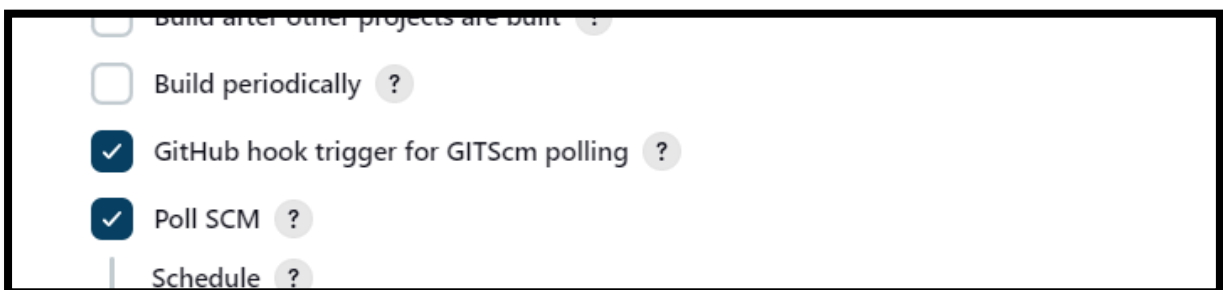
Scope: Global (Jenkins, nodes, items, all child items, etc)

Username: deployer

☐ Treat username as secret

Password:

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



☐ Build after other projects are built

☐ Build periodically

☒ GitHub hook trigger for GITScm polling

☒ Poll SCM

☐ Schedule

Deploy war/ear to a container

WAR/EAR files ?

**/*.war

Context path ?

Containers

Tomcat 8.x Remote

Credentials

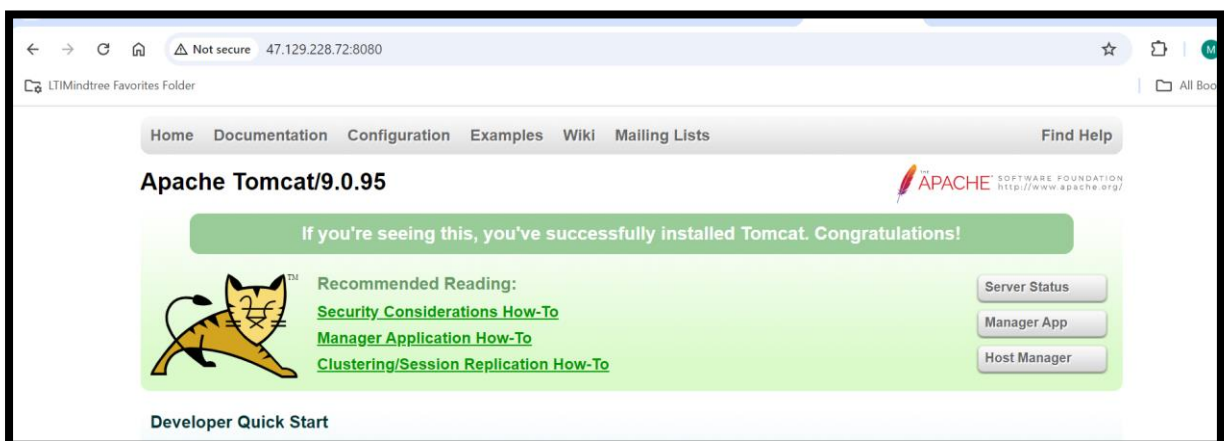
developer/*****

+ Add ▾

Tomcat URL ?

http://47.129.228.72:8080/

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



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

Sign in

http://47.129.228.72:8080

Your connection to this site is not private

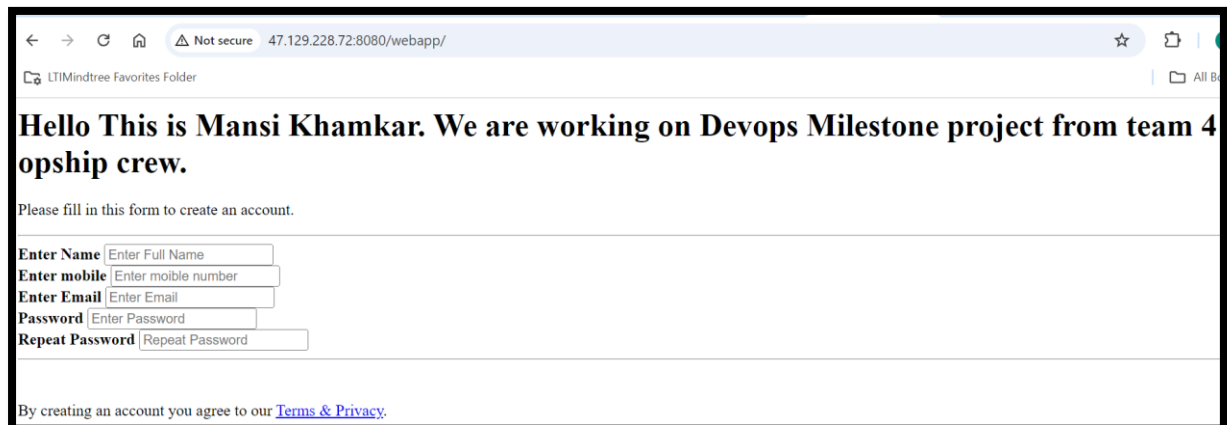
Username

Password

Open /webapp

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

Your web application should open.



The screenshot shows a web browser window with the address bar displaying "47.129.228.72:8080/webapp/". The page content includes a greeting: "Hello This is Mansi Khamkar. We are working on Devops Milestone project from team 4 opship crew." Below this is a registration form with the instruction "Please fill in this form to create an account." The form contains five input fields: "Enter Name" (placeholder: Enter Full Name), "Enter mobile" (placeholder: Enter mobile number), "Enter Email" (placeholder: Enter Email), "Password" (placeholder: Enter Password), and "Repeat Password" (placeholder: Repeat Password). At the bottom of the form, there is a link to "Terms & Privacy".

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/docker-ce.repo
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 authentication yes

Password authentication yes

Permit empty yes

Esc > :wq! > enter

Run `systemctl restart sshd`, `systemctl enable sshd`

Configure aws iam user:

<input type="checkbox"/>	Policy Name	Type	Attached via
<input type="checkbox"/>	AdministratorAccess	AWS managed - job function	Directly
<input type="checkbox"/>	AmazonEC2FullAccess	AWS managed	Directly
<input type="checkbox"/>	AmazonRoute53FullAccess	AWS managed	Directly
<input type="checkbox"/>	AmazonS3FullAccess	AWS managed	Directly
<input type="checkbox"/>	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
<div><div></div>project-11</div>	<div> 122610503177.dkr.ecr.ap-southeast-1.amazonaws.com/project-11</div>	September 19, 2024, 16:36:24 (UTC+05.5)	Mutable	AES-256


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

Publish Over SSH 1.25

Send build artifacts over SSH

[Report an issue with this plugin](#)

☒



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

Publish over SSH

Jenkins SSH Key ?

Passphrase ?

 Concealed

Path to key ?

Key ?

-----BEGIN OPENSsh PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAABG5vbmUAAAABbm9uZQAAAAAAAAABAAABlwAAAAadzC2gtcn
NhAAAAAwEAAQAAAYEAlrjC4P6Dn7dQSe25S06uAlikN/VXik57Q4alBcLckJzzu0QNkXjc
lv3+9HJ4SXEZR7z6aZQ+LBqDGCnqyM/DgAAaCO/y5ecyaBeiUXsODNkUzfRD9Vhl/PI+kJ
SqCIRyN3ZmpY7ljwa0LXVe2G/8G+y4lo9koB3WX34Xa7Lzc2QAHJ6rKdS4Ay5MJz5ioHf9

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

Name ?

jenkins

Hostname ?

172.31.25.56

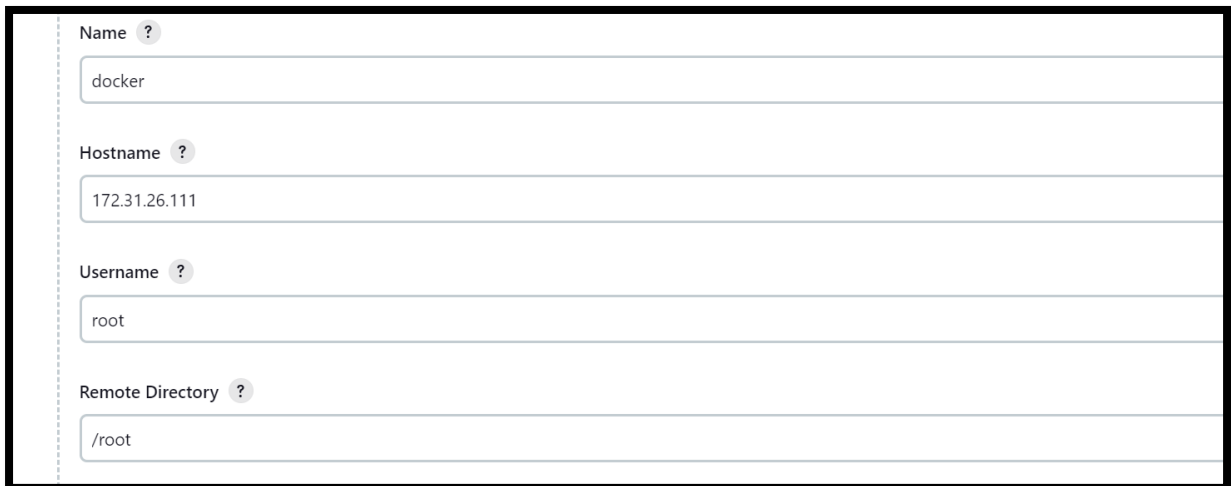
Username ?

root

Remote Directory ?

/root

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



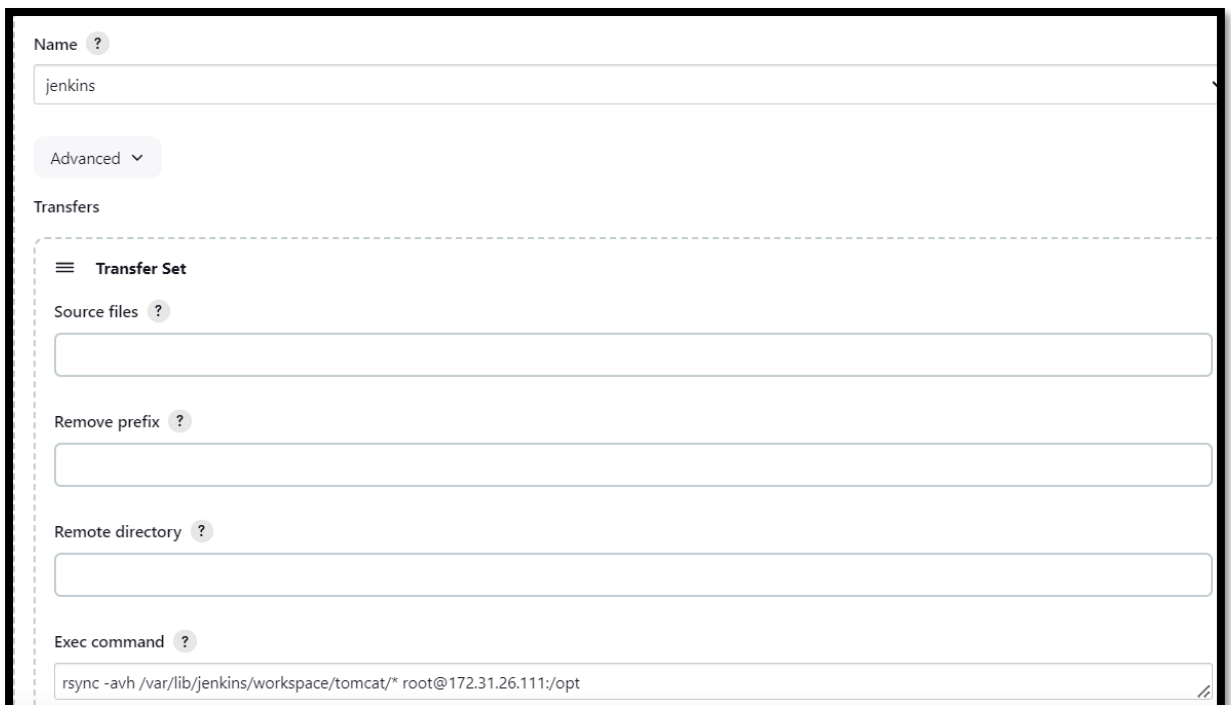
A screenshot of a Jenkins configuration form for an SSH server. The form has four input fields, each with a label and a help icon (a question mark in a circle). The fields are: 'Name' with the value 'docker', 'Hostname' with the value '172.31.26.111', 'Username' with the value 'root', and 'Remote Directory' with the value '/root'.

Name ?
docker
Hostname ?
172.31.26.111
Username ?
root
Remote Directory ?
/root

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.



A screenshot of a Jenkins configuration form for a post-built action. The form has a 'Name' field with the value 'jenkins' and an 'Advanced' dropdown menu. Below the dropdown is a 'Transfers' section with a 'Transfer Set' header. The 'Transfer Set' has four input fields: 'Source files', 'Remove prefix', 'Remote directory', and 'Exec command'. The 'Exec command' field contains the command 'rsync -avh /var/lib/jenkins/workspace/tomcat/* root@172.31.26.111:/opt'.

Name ?
jenkins
Advanced ▾
Transfers
Transfer Set
Source files ?
Remove prefix ?
Remote directory ?
Exec command ?
rsync -avh /var/lib/jenkins/workspace/tomcat/* root@172.31.26.111:/opt

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.

Name ?

docker

Advanced ▾

Transfers

≡ Transfer Set

Source files ?

Remove prefix ?

Remote directory ?

Exec command ?

cd /opt
aws ecr get-login-password --region ap-southeast-1 | docker login --username AWS --password-stdin 122610503177.dkr.ecr.ap-southeast-1.amazonaws.com
docker build -t project-11 .
docker tag project-11:latest 122610503177.dkr.ecr.ap-southeast-1.amazonaws.com/project-11:latest
docker push 122610503177.dkr.ecr.ap-southeast-1.amazonaws.com/project-11:latest

Apply > save > build Now

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

			(MB)	URI	
<input type="checkbox"/>	latest	Image	September 21, 2024, 09:44:49 (UTC+05.5)	227.35	Copy URI sha256:1328fe0e...

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

Create IAM roles in aws and attach policy with instance :

<input type="checkbox"/>	Policy name ↗	Type	Attached entities
<input type="checkbox"/>	AmazonEKSClusterPolicy	AWS managed	2
<input type="checkbox"/>	AmazonElasticContainer...	AWS managed	1
<input type="checkbox"/>	AWSCloudFormationFull...	AWS managed	1
<input type="checkbox"/>	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 authentication yes

Password authentication yes

Permit empty yes

Esc > :wq! > enter

Install eksctl and kubectl in cluster server

```
Default output format [none]: table
[root@kubernetes ~]# cd
[root@kubernetes ~]# curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz" | tar xz -C /tmp
sudo mv /tmp/eksctl /usr/local/bin
eksctl version
0.190.0
[root@kubernetes ~]# curl -LO https://storage.googleapis.com/kubernetes-release/release/$(curl -s https://storage.googleapis.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
[root@kubernetes ~]# eksctl create cluster --name cluster1 --region ap-south-1 --version 1.29 --unc-public-subnets subn
```

Create cluster and nodegroup

```
[root@kubernetes ~]# eksctl create cluster --name cluster2 --region ap-south-1 --version 1.29 --vpc-public-subnets subnet-0241e4185ealb6efc,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 [I] using existing VPC (vpc-00ff47d27a0912365) and subnets (private:map[] public:map[ap-south-1a:{subnet-0241e4185ealb6efc 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 [I] custom VPC/subnets will be used; if resulting cluster doesn't function as expected, make sure to 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-TYPE-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 [I] all EKS cluster resources for "cluster2" have been created
2024-09-20 12:43:28 [I] created 0 nodegroup(s) in cluster "cluster2"
2024-09-20 12:43:28 [I] 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 [I] 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-0241e4185ealb6efc,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-0ab6dcdf35da05038" [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

```
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.

<input type="checkbox"/>	latest	Image	September 21, 2024, 09:44:49 (UTC+05.5)	227.35	Copy URI	sha256:1328fe0e...
--------------------------	--------	-----------------------	---	--------	-------------	--------------------

```
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.

SSH Server

Name ?

Hostname ?

Username ?

Remote Directory ?

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 deployment and service.

Name ?

Advanced ▾

Transfers

Transfer Set

Source files ?

Remove prefix ?

Remote directory ?

Exec command ?

```
kubectl delete deployment app-webapp
kubectl apply -f deployment.yml
kubectl apply -f service.yml
```

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)
app-service	LoadBalancer	10.100.229.100	a32388ce2015b480db71ab6ed876f5cd-1905233686.ap-southeast-1.elb.amazonaws.com	8080:309
kubernetes	ClusterIP	10.100.0.1	<none>	443/TCP

The screenshot shows a web browser window with the address bar displaying the URL: a32388ce2015b480db71ab6ed876f5cd-1905233686.ap-southeast-1.elb.amazonaws.com:8080/webapp/. The page content includes a greeting: "Hello This is Mansi Khamkar. We are working on Devops Milestone project from team 4 opship crew." Below this is a registration form with the text "Please fill in this form to create an account." The form fields are: "Enter Name" (with placeholder "Enter Full Name"), "Enter mobile" (with placeholder "Enter moible number"), "Enter Email" (with placeholder "Enter Email"), "Password" (with placeholder "Enter Password"), and "Repeat Password" (with placeholder "Repeat Password"). Below the form is a "Register" button. At the bottom of the form, there is a link to "Terms & Privacy" and a link to "Sign in." The page concludes with the text "Thank You" and "Happy Learning. See You Again."

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

