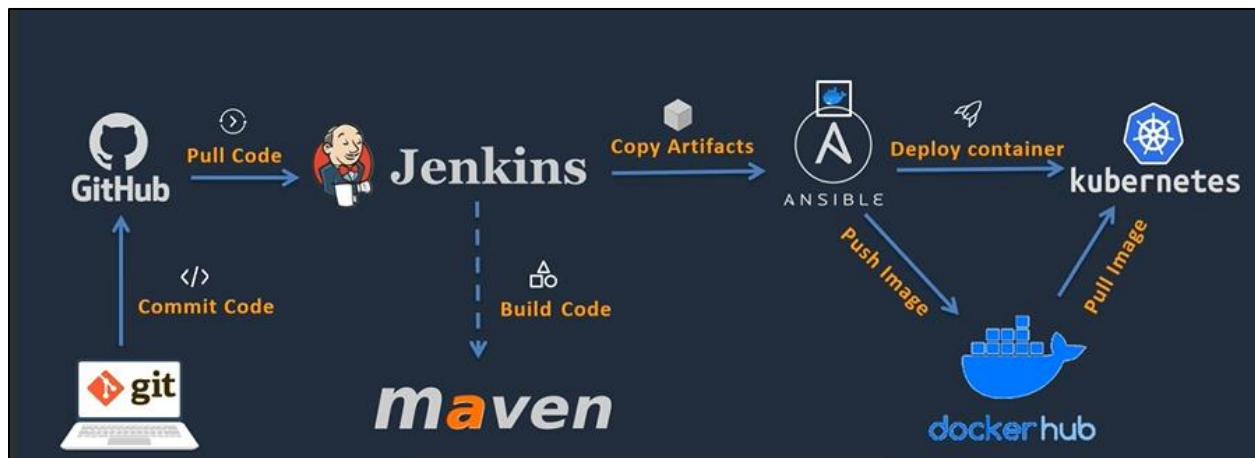


DevOps Project 1

Overview

This project involved setting up a Continuous Integration (CI) and Continuous Deployment (CD) pipeline to automate the deployment process of our applications.

Project Details



In this project, our objective is to establish a CI/CD pipeline to streamline the deployment process. Whenever alterations are made to the source code, they are promptly committed to our GitHub repository.

Subsequently, Jenkins is configured to fetch the code, utilize Maven for building, resulting in the generation of artifacts in the form of a .war file. These artifacts are then forwarded to the Ansible Server.

Concurrently, an Ansible playbook is executed to construct an image incorporating the artifacts, which is then committed to the Docker Hub repository.

Furthermore, we've implemented a CD job that can be initiated through Jenkins. This job triggers an Ansible Playbook, orchestrating the deployment and service files on our Kubernetes cluster.

Setup Jenkins Server

- Setup a Linux EC2 Instance.
- Install JAVA.
- Install Jenkins.
- Access Web UI on port 8080.



Steps to follow:

Create a 2 AWS EC2 Instance.

Commands:

```
sudo yum update -y  
sudo yum upgrade -y  
sudo nano /etc/hostname --- #change the hostname to Jenkins master.  
sudo init 6 --- #To restart the machine
```

Command to Install Java:

```
yum install java-17 -y
```

Command to Install Jenkins:

```
sudo wget -O/etc/yum.repos.d/jenkins.repo \https://pkg.jenkins.io/redhat-stable/jenkins.repo  
sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key  
sudo yum upgrade  
sudo yum install fontconfig java-17-openjdk  
sudo yum install jenkins  
sudo systemctl daemon-reload
```

```
jenkins --version  
systemctl start jenkins ---#To start the Jenkins
```

--# Now Jenkins is set up. Copy the Public key of the Jenkins Master and run on the browser.
Ex - 3.110.42.120:8080

```
cat /var/lib/jenkins/secrets/initialAdminPassword ---#Get the password.
```

Command to Install Git:

```
yum install git -y  
git init ---#To create the git repository  
ls -a ---# You will see the repository with .git
```

On Jenkins GUI -

Install the GitHub integration plugin.

The screenshot shows the 'Git installations' configuration page under 'Manage Jenkins > Tools'. A 'Git' entry is selected. The configuration fields are:

- Name: git
- Path to Git executable: git
- Install automatically:

A 'Add Git' button is at the bottom left.

Run Jenkins job to pull code from GitHub.

- Create a Job in Jenkins GUI.
- Add the Git Hub Repository - <https://github.com/amanduggal001/hello-world.git>

The screenshot shows the 'Configure' screen for a job named 'PullCodeFromGitHub' under 'Configuration'. The 'Source Code Management' section is active, showing 'Git' selected. The repository URL is set to <https://github.com/amanduggal001/hello-world.git>. The 'Credentials' dropdown is set to '- none -'.

Console Output

 **Console Output**

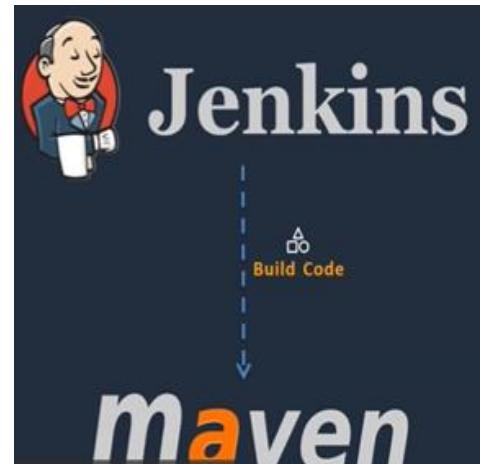
```

Started by user Aman Duggal
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/PullCodeFromGitHub
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/PullCodeFromGitHub/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/amanduggal001/hello-world.git # timeout=10
Fetching upstream changes from https://github.com/amanduggal001/hello-world.git
> git --version # timeout=10
> git --version # 'git version 2.40.1'
> git fetch --tags --force --progress -- https://github.com/amanduggal001/hello-world.git
+refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision aacc9fa39ba3ca8f46cf0019f35ce4511287375 (refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout -f aacc9fa39ba3ca8f46cf0019f35ce4511287375 # timeout=10
Commit message: "updated pom.xml file"
> git rev-list --no-walk aacc9fa39ba3ca8f46cf0019f35ce4511287375 # timeout=10
Finished: SUCCESS

```

Integrate Maven with Jenkins

- Setup Maven on Jenkins Server.
- Setup Environment Variables
JAVA_HOME, M2, M2_HOME
- Install Maven plugin.
- Configure Maven and Java.



Commands:

```

cd /opt
wget https://dlcdn.apache.org/maven/maven-3/3.9.6/binaries/apache-maven-3.9.6-bin.tar.gz --#To
download the maven.
tar -xvf apache-maven-3.9.6-bin.tar.gz --#To unzip the tar file.
rm -rf apache-maven-3.9.6-bin.tar.gz --#To remove the tar file.

```

When you download or clone raw code, it comes in a folder. This folder is inside Jenkins, and it contains all the code, like the POM.xml file and source code.

Next, when we ask Jenkins to build, it uses Maven (mvn) on that folder where the POM.xml file is. Maven reads the POM.xml file and follows the instructions inside. It builds, compiles, and tests the code accordingly.

Now, the question is, can we access Maven (mvn) on the entire terminal so far? To check follow the below steps.

```
cd /opt  
./mvn --version --#You can't access the mvn in/opt folder.  
cd /opt/apache-maven-3.9.6/bin  
./mvn --version --#You can access the mvn in bin folder.
```

It means the mvn is accessible only from the particular bin folder not on entire terminal.

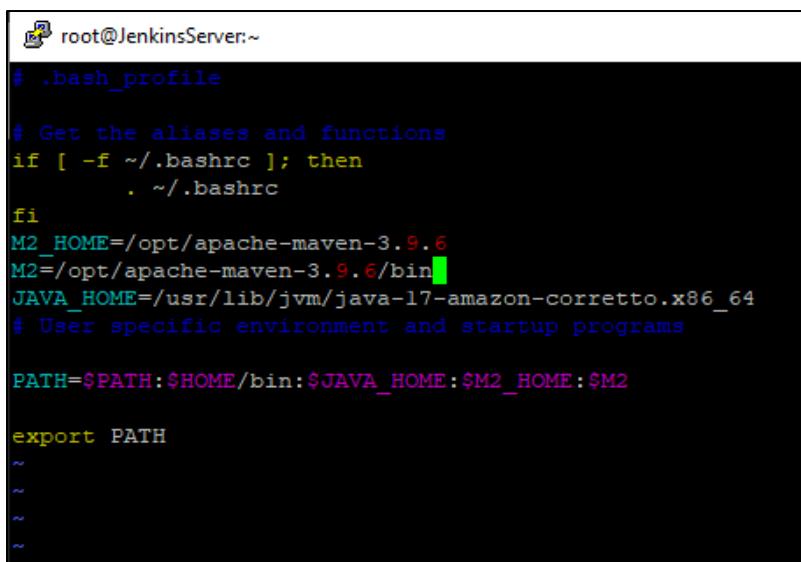
Now we need to define a path (Environment Variable), for mvn, so that the Jenkins can call the mvn from their location and start building.

```
cd ~  
vi .bash_profile --#Set the mvn path here  
  
find / -name java-17* --#To find location where the java is installed.
```

```
M2_HOME=/opt/apache-maven-3.9.6  
M2=/opt/apache-maven-3.9.6/bin  
JAVA_HOME=/usr/lib/jvm/java-17-amazon-corretto.x86_64  
# User specific environment and startup programs
```

```
PATH=$PATH:$HOME/bin:$JAVA_HOME:$M2_HOME:$M2
```

```
export PATH
```



A terminal window titled 'root@JenkinsServer:~' showing the configuration of the .bash_profile file. The window displays the following content:

```
# .bash_profile  
  
# Get the aliases and functions  
if [ -f ~/.bashrc ]; then  
    . ~/.bashrc  
fi  
M2_HOME=/opt/apache-maven-3.9.6  
M2=/opt/apache-maven-3.9.6/bin  
JAVA_HOME=/usr/lib/jvm/java-17-amazon-corretto.x86_64  
# User specific environment and startup programs  
  
PATH=$PATH:$HOME/bin:$JAVA_HOME:$M2_HOME:$M2  
  
export PATH
```

source .bash_profile --#It refresh the .bash_profile, what changes we made in the file. (Source command is used to read the file)

Mvn --version --# Validate if the mvn is calling on other folders now.

Integrate Git and Maven to Jenkins

- Go to Jenkins UI.
- Manage Jenkins > Plugins > Install Maven Integration.

The screenshot shows the Jenkins Plugins page. The navigation bar at the top says "Dashboard > Manage Jenkins > Plugins". Below the navigation, there's a search bar with the text "maven". On the left, there are four buttons: "Updates", "Available plugins" (which is highlighted in a light gray box), "Installed plugins", and "Advanced settings". On the right, there's a list of available plugins. The "Maven Integration 3.23" plugin is selected, indicated by a checked checkbox. The "Build Tools" section below it contains the following text:
This plugin provides a deep integration between Jenkins and Maven. It adds automatic triggers between projects, and provides configuration of various Jenkins parameters for Maven builds.

- Manage Jenkins> Tools > Add JDK and Maven Details.

The screenshot shows the Jenkins Tools page. The navigation bar at the top says "Dashboard > Manage Jenkins > Tools". Below the navigation, there's a button labeled "Add JDK". A dashed-line box encloses the configuration for "java 17". Inside the box, there are fields for "Name" (containing "java 17") and "JAVA_HOME" (containing "/usr/lib/jvm/java-17-amazon-corretto.x86_64"). A red warning message below the JAVA_HOME field states: "!! /usr/lib/jvm/java-17-amazon-corretto.x86_64 doesn't look like a JDK directory". There's also a checkbox labeled "Install automatically" with a question mark icon next to it. At the bottom of the box is another "Add JDK" button.

Maven installations

Add Maven

Maven

Name
Maven

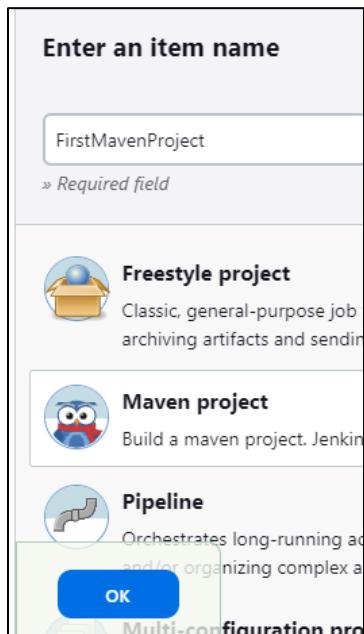
MAVEN_HOME
/opt/apache-maven-3.9.6

Install automatically ?

Add Maven

Build a java project using Jenkins

- Go to Jenkins UI > New Item > Create a new Job using Maven Project.



- First Select the Git.

The screenshot shows the Jenkins configuration interface for a project named "FirstMavenProject". Under the "Source Code Management" section, the "Git" option is selected. The "Repository URL" field contains the value "https://github.com/amanduggal001/hello-world.git". Other fields like "Credentials" and "Advanced" are shown below.

- Enter the command for maven build in the goals and option.

Clean install --# This command creates the packages and stores in the local repository.

The screenshot shows the Jenkins configuration interface under the "Build" section. The "Goals and options" field contains the value "clean install". Other fields like "Root POM" and "Advanced" are shown below.

- After build the .war file is created under the below location.

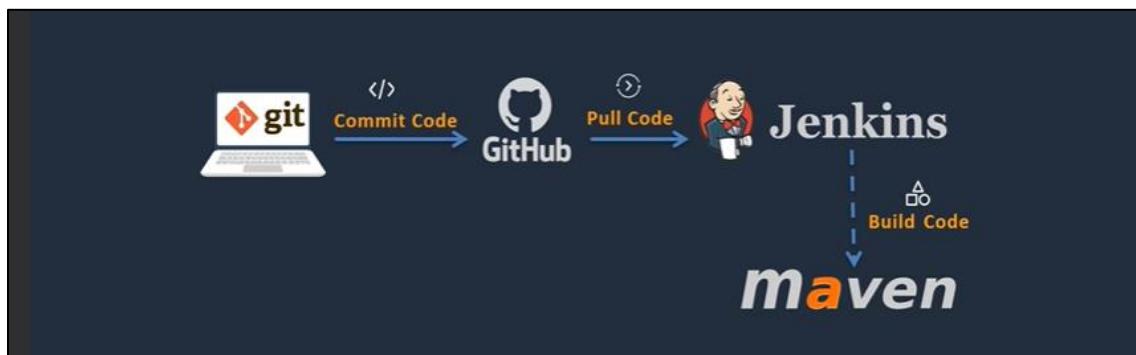
/var/lib/jenkins/workspace/FirstMavenProject/webapp/target/webapp.war

Console Output

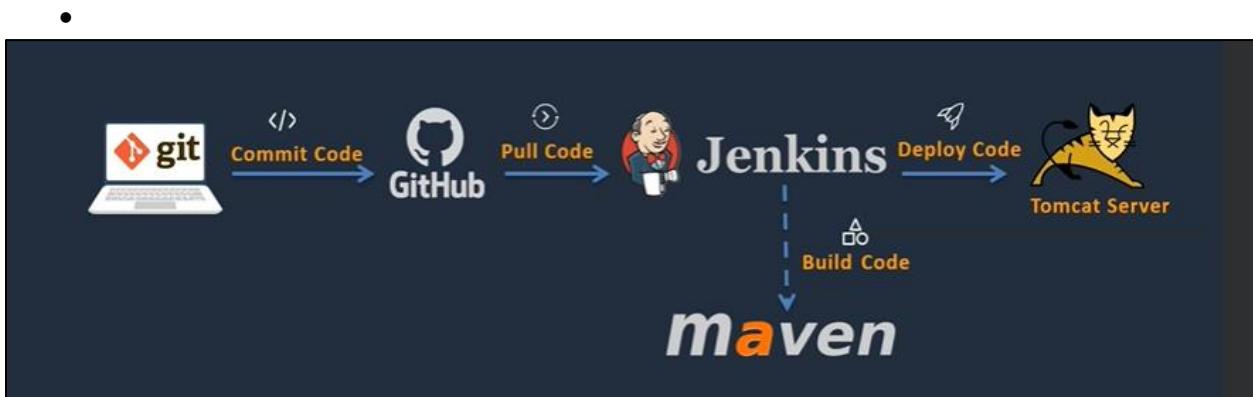
```

[INFO] ..... SUCCESS [ 18.038 s]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 18.038 s
[INFO] Finished at: 2024-03-08T17:38:27Z
[INFO] -----
Waiting for Jenkins to finish collecting data
[JENKINS] Archiving /var/lib/jenkins/workspace/FirstMavenProject/webapp/pom.xml to com.example.maven-project/webapp/1.0-SNAPSHOT/webapp-1.0-SNAPSHOT.pom
[JENKINS] Archiving /var/lib/jenkins/workspace/FirstMavenProject/webapp/target/webapp.war to com.example.maven-project/webapp/1.0-SNAPSHOT/webapp-1.0-SNAPSHOT.war
[JENKINS] Archiving /var/lib/jenkins/workspace/FirstMavenProject/server/pom.xml to com.example.maven-project/server/1.0-SNAPSHOT/server-1.0-SNAPSHOT.pom
[JENKINS] Archiving /var/lib/jenkins/workspace/FirstMavenProject/server/target/server.jar to com.example.maven-project/server/1.0-SNAPSHOT/server-1.0-SNAPSHOT.jar
[JENKINS] Archiving /var/lib/jenkins/workspace/FirstMavenProject/pom.xml to com.example.maven-project/maven-project/1.0-SNAPSHOT/maven-project-1.0-SNAPSHOT.pom
channel stopped
Finished: SUCCESS

```



Deployment Artifacts on a Tomcat Server



Till Now, we have build our code. Now how we can deploy our code on the target environment.

As a target environment we are going to use the **Tomcat Server**.

Setup Tomcat Server

- Setup a Linux EC2 Instance.
- Install Java.
- Configure Tomcat.
- Start Tomcat Server.
- Access Web UI on port 8080.



Steps to follow:

- Create an EC2 instance.

Commands:

```
sudo yum update -y  
sudo yum upgrade -y  
sudo nano /etc/hostname --- #change the hostname to Jenkins master  
sudo init 6 --- #To restart the machine
```

Install Java

```
yum install java-17 -y
```

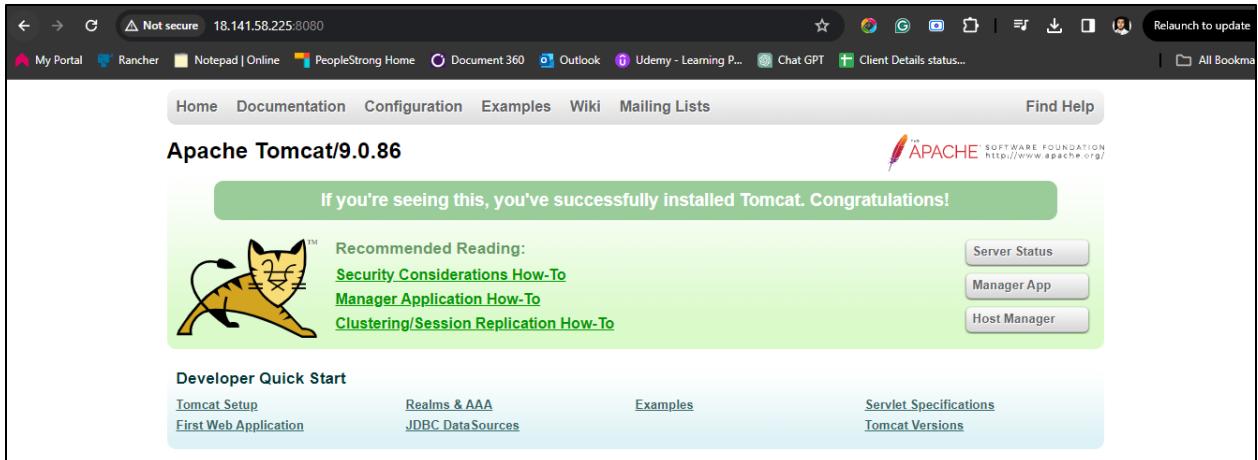
Install Tomcat

```
wget https://downloads.apache.org/tomcat/tomcat-9/v9.0.86/bin/apache-tomcat-9.0.86.tar.gz  
tar -xvzf apache-tomcat-9.0.86.tar.gz  
cd /apache-tomcat-9.0.86/bin  
.startup.sh --#To start the Tomcat Services
```

```
[root@TomcatServer bin]# ./startup.sh  
Using CATALINA_BASE:   /home/ec2-user/apache-tomcat-9.0.86  
Using CATALINA_HOME:   /home/ec2-user/apache-tomcat-9.0.86  
Using CATALINA_TMPDIR: /home/ec2-user/apache-tomcat-9.0.86/temp  
Using JRE_HOME:        /  
Using CLASSPATH:       /home/ec2-user/apache-tomcat-9.0.86/bin/bootstrap.jar:/home/ec2-user/apache-tomcat-9.0.86/bin/tomcat-juli.jar  
Using CATALINA_OPTS:  
Tomcat started.
```

- Now go to browser > Copy the Public IP of Tomcat Server

Example: 18.141.58.225:8080



- Click on Manager App.

You will get the 403 Forbidden. This is because by default the manager is only accessible from the browser running on the same machine as Tomcat.

403 Access Denied

You are not authorized to view this page.

By default the Manager is only accessible from a browser running on the same machine as Tomcat. If you wish to modify this restriction, you'll need to edit the Manager's `[context.xml]` file.

If you have already configured the Manager application to allow access and you have used your browser's back button, used a saved bookmark or similar then you may have triggered the cross-site request forgery (CSRF) protection that has been enabled for the HTML interface of the Manager application. You will need to reset this protection by returning to the `main Manager` page. Once you return to this page, you will be able to continue using the Manager application's HTML interface normally. If you continue to see this access denied message, check that you have the necessary permissions to access this application.

If you have not changed any configuration files, please examine the file `[conf/tomcat-users.xml]` in your installation. That file must contain the credentials to let you use this webapp.

For example, to add the `manager-gui` role to a user named `tomcat` with a password of `s3cret`, add the following to the config file listed above.

```
<role rolename="manager-gui">
<user username="tomcat" password="s3cret" roles="manager-gui"/>
```

Note that for Tomcat 7 onwards, the roles required to use the manager application were changed from the single `manager` role to the following four roles. You will need to assign the role(s) required for the functionality you wish to access.

- `manager-gui` - allows access to the HTML GUI and the status pages
- `manager-script` - allows access to the text interface and the status pages
- `manager-jmx` - allows access to the JMX proxy and the status pages
- `manager-status` - allows access to the status pages only

The HTML interface is protected against CSRF but the text and JMX interfaces are not. To maintain the CSRF protection:

- Users with the `manager-gui` role should not be granted either the `manager-script` or `manager-jmx` roles.
- If the text or JMX interfaces are accessed through a browser (e.g. for testing since these interfaces are intended for tools not humans) then the browser must be closed afterwards to terminate the session.

For more information - please see the [Manager App How-To](#).

To Access tomcat outside of the tomcat server. Follow the below steps.

find / -name context.xml --# To find the location on the file.

vi /home/ec2-user/apache-tomcat-9.0.86/webapps/host-manager/META-INF/context.xml

Comment Out below line which means that the tomcat will be allowed to run from the localhost only.

```
<!-- <Valve className="org.apache.catalina.valves.RemoteAddrValve"
allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:1" /> -->
```

```

root@TomcatServer:/home/ec2-user/apache-tomcat-9.0.86/bin
<?xml version="1.0" encoding="UTF-8"?>
<!--
  Licensed to the Apache Software Foundation (ASF) under one or more
  contributor license agreements. See the NOTICE file distributed with
  this work for additional information regarding copyright ownership.
  The ASF licenses this file to You under the Apache License, Version 2.0
  (the "License"). You may not use this file except in compliance with
  the License. You may obtain a copy of the License at

      http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License.
-->
<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|0:0:0:0:0:0:1" /> -->
  <Manager sessionAttributeValueClassNameFilter="java\.lang\.(\?Boolean|\?Integer|\?Long|\?Number|\?String)|org\.apache\.catalina\.filters\.CsrfPreventionFilter\?LruCache\?:\$1|\?java\.util\.(\?Linked)?HashMap"/>
</Context>
~
```

Same steps perform in another manager app.

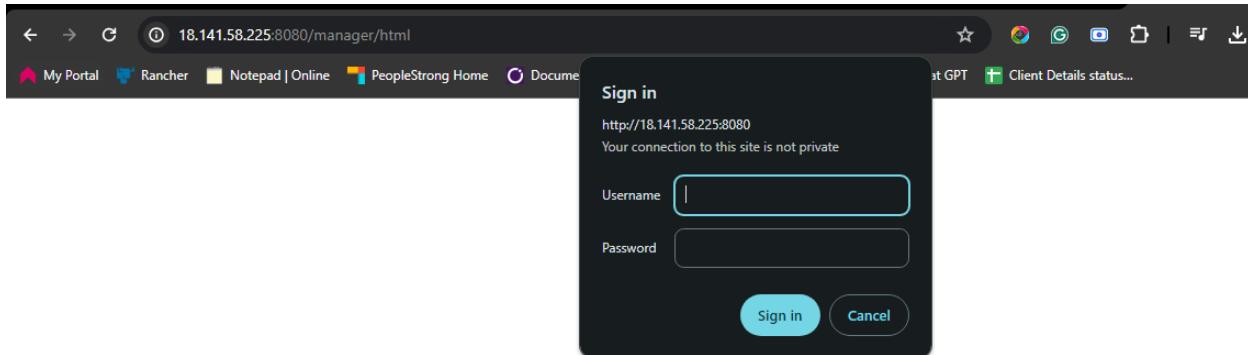
```
vi /home/ec2-user/apache-tomcat-9.0.86/webapps/manager/META-INF/context.xml
```

```
<!-- <Valve className="org.apache.catalina.valves.RemoteAddrValve"
allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:1" /> -->
```

Now restart the Tomcat services.

```
cd /apache-tomcat-9.0.86/bin
./shutdown.sh --#To Shutdown service
./startup.sh --#To Start the service
```

Now when you click on Manager App, it will ask you for credentials.



Update user's information in the tomcat-users.xml file. Go to tomcat home directory and add below users to conf/tomcat-users.xml file. --#Add roles and users in the user.xml file

```

<role rolename="manager-gui"/>
<role rolename="manager-script"/>
<role rolename="manager-jmx"/>
<role rolename="manager-status"/>
<user username="admin" password="Ammy@567" roles="manager-gui, manager-script, manager-jmx,
manager-status"/>--#Enter Password.
<user username="deployer" password="Ammy@567" roles="manager-script"/> --#Enter Password.
<user username="tomcat" password="Ammy@567" roles="manager-gui"/> --#Enter Password.

```

```

The sample user and role entries below are intended for use with the
examples web application. They are wrapped in a comment and thus are ignored
when reading this file. If you wish to configure these users for use with the
examples web application, do not forget to remove the <!.. ..> that surrounds
them. You will also need to set the passwords to something appropriate.
-->
<!--
<role rolename="tomcat"/>
<role rolename="role1"/>
<user username="tomcat" password="" roles="tomcat"/>
<user username="both" password="" roles="tomcat,role1"/>
<user username="role1" password="" roles="role1"/>
-->

<role rolename="manager-gui"/>
<role rolename="manager-script"/>
<role rolename="manager-jmx"/>
<role rolename="manager-status"/>
<user username="admin" password="Ammy@567" roles="manager-gui, manager-script, manager-jmx, manager-status"/>
<user username="deployer" password="Ammy@567" roles="manager-script"/>
<user username="tomcat" password="Ammy@567" roles="manager-gui"/>
-->
</tomcat-users>

```

```

cd /apache-tomcat-9.0.86/bin
./shutdown.sh --#To Shutdown service
./startup.sh --#To Start the service

```

Now login with:

```

tomcat
Ammy@567

```

Integrate Tomcat with Jenkins

- Install “Deploy to container”.
- Configure Tomcat server with Credentials.

Steps to follow:

- Go to Jenkins GUI > Manage Jenkins > Plugins > Install '**Deploy To container**' plugin.

The screenshot shows the Jenkins 'Plugins' page. In the top navigation bar, 'Manage Jenkins' is selected. On the left, there's a sidebar with options: 'Updates', 'Available plugins' (which is highlighted in blue), 'Installed plugins', 'Advanced settings', and 'Download progress'. A search bar at the top right contains the text 'deploy to conta'. Below the search bar, a table lists the 'Deploy to container' plugin by 'Artifact Uploaders'. The table includes columns for 'Install' (with a checkbox), 'Name' (1.16), and a brief description: 'This plugin allows you to deploy a war to a container after a successful build. Glassfish 3.x remote deployment'. The 'Install' checkbox is currently unchecked.

- Manage Jenkins > Credentials > System > Global credentials (unrestricted) > Add Credentials

The screenshot shows the 'New credentials' page under 'Manage Jenkins > Credentials > System > Global credentials (unrestricted)'. The 'Kind' dropdown is set to 'Username with password'. The 'Scope' dropdown is set to 'Global (Jenkins, nodes, items, all child items, etc.)'. The 'Username' field contains 'deployer'. The 'Password' field contains '.....'. The 'ID' field contains 'tomcat_deployer'. The 'Description' field contains 'tomcat_deployer'. A 'Create' button is visible at the bottom.

- Now create new job '**BuildAndDeploy**' > Use Maven job.

The screenshot shows the 'Job Configuration' page for a new job. The 'Source Code Management' section is open, showing 'Git' selected. Under 'Git', the 'Repository URL' is set to 'https://github.com/amanduggal001/hello-world.git'. The 'Credentials' dropdown is set to '- none -'. There are 'Advanced' and '+ Add' buttons at the bottom of the section.

Build

Root POM ?
pom.xml

Goals and options ?
clean install

Advanced ▾

- Select '**Deploy war/ear to a container**' in Post Build Action under Build Settings.
- Enter the path where the .war file is created.
`/var/lib/jenkins/workspace/FirstMavenProject/webapp/target/webapp.war`
Enter only --**/*.war
- Now add '**Tomcat 8.x**' Remote in Containers.
- Select the credentials which we define for tomcat in Jenkins GUI.
- Now enter the tomcat URL.
<http://18.141.58.225:8080/>
- Apply > Save.
- Build now.

Dashboard > BuildAndDeploy > Configuration

BUILD SETTINGS

Configure

E-mail Notification

Post-build Actions

Deploy war/ear to a container

WAR/EAR files ?
**/*.war

War/ear files to deploy. Relative to the workspace root. You can also specify Ant-style GLOBs, like "*/*.war".
(from Deploy to container Plugin)

Context path ?

Containers

Tomcat 8.x Remote

Credentials
deployer/******** (tomcat_deployer)

+ Add +

Tomcat URL ?
http://18.141.58.225:8080/

Save **Apply**

In the tomcat server the .war file will be created in the below location:
/apache-tomcat-9.0.86/webapps

```
[root@TomcatServer apache-tomcat-9.0.86]# ls
bin           CONTRIBUTING.md  logs      RELEASE-NOTES  webapps
BUILDING.txt  lib              NOTICE    RUNNING.txt   work
conf          LICENSE          README.md temp
[root@TomcatServer apache-tomcat-9.0.86]# cd webapps
[root@TomcatServer webapps]# ls
docs  examples  host-manager  manager  ROOT  webapp  webapp.war
[root@TomcatServer webapps]#
```

Now in Tomcat the webapp is visible and when you click on it you will see the outputs.

Tomcat Web Application Manager

Message:		OK			
Manager					
List Applications		HTML Manager Help		Manager Help	
Applications					
Path	Version	Display Name	Running	Sessions	Commands
/	None specified	Welcome to Tomcat	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/docs	None specified	Tomcat Documentation	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/examples	None specified	Servlet and JSP Examples	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/host-manager	None specified	Tomcat Host Manager Application	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/manager	None specified	Tomcat Manager Application	true	1	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/webapp	None specified	Webapp	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

New user Register for DevOps Learning

Please fill in this form to create an account.

Enter Name	<input type="text"/>
Enter mobile	<input type="text"/>
Enter Email	<input type="text"/>
Password	<input type="password"/>
Repeat Password	<input type="password"/>

By creating an account you agree to our [Terms & Privacy](#).

Already have an account? [Sign in](#).

Thankyou, Happy Learning

Now if you want to change the code which you are viewing in the tomcat server.

- Pull this code onto the local workstation.
- Update the code and commit it on GitHub.
- Once it is committed, we can compile and build our code with the help of maven.
- Create a folder in your local system and copy the path in the Git Bash App.

cd D:/AWS/DevOps Project --#Change \ fwd slash to / bckwd slash, because it is treated as a Linux.

```
Aman.Duggal@AmanD-OEG MINGW64 ~
$ cd D:/AWS/DevopsProject

Aman.Duggal@AmanD-OEG MINGW64 /d/AWS/DevopsProject
$ |
```

- git clone <https://github.com/amanduggal001/hello-world.git>

```
Aman.Duggal@AmanD-OEG MINGW64 /d/AWS/DevopsProject
$ git clone https://github.com/amanduggal001/hello-world.git
Cloning into 'hello-world'...
remote: Enumerating objects: 333, done.
remote: Total 333 (delta 0), reused 0 (delta 0), pack-reused 333
Receiving objects: 100% (333/333), 35.85 KiB | 2.39 MiB/s, done.
Resolving deltas: 100% (74/74), done.
```

- cd /hello-world/webapp/src/main/webapp
- vi index.jsp
- Change the code - copy code from <https://www.geeksforgeeks.org/html-registration-form/>

git status

```
Aman.Duggal@AmanD-OEG MINGW64 /d/AWS/DevopsProject/hello-world/webapp/src/main/webapp (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
    (use "git restore <file>..." to discard changes in working directory)
      modified:   index.jsp

no changes added to commit (use "git add" and/or "git commit -a")
```

git add .

git status

```
Aman.Duggal@AmanD-OEG MINGW64 /d/AWS/DevopsProject/hello-world/webapp/src/main/webapp (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   index.jsp
```

```
git commit -m "updated index.jsp file"
```

```
Aman.Duggal@AmanD-OEG MINGW64 /d/AWS/DevopsProject/hello-world/webapp/src/main/webapp (master)
$ git commit -m "updated index.jsp file"
[master 0c63047] updated index.jsp file
Committer: Aman Duggal <Aman.Duggal@businessnext.com>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

  git config --global --edit

After doing this, you may fix the identity used for this commit with:

  git commit --amend --reset-author

1 file changed, 74 insertions(+), 37 deletions(-)
```

git push origin master --#To push the code

Once Push the code will be updated in the github.

- Now Build the job '**BuildAndDeploy**'
- Open the Tomcat Server > Manager App > Webapp
- New Registration form will be open.

The screenshot shows a registration form with the following fields:
First Name: [Input field]
Last Name: [Input field]
Email: [Input field]
Date of Birth: [Input field]
Password: [Input field]
Re-type Password: [Input field]
Re-Enter your password: [Input field]
Contact: [Input field]
Gender: Male
Submit

Now till here we are manually executing the job from jenkins whenever we are changing or update the code.

Our Purpose is whenever there is a change in GitHub it should automatically identify and execute the build job and deploy the code.

To do that Build triggers are used.

- Go to Jenkins > Configure the '**BuildAndDeploy**' job.
- In Build triggers select the **Poll SCM**.

Poll SCM will validate whether there is any changes anywhere in repository or not in GitHub. If there are no changes, build is not going to happen.
If there are some changes, build going to executed.

Difference in Build periodically and Poll SCM is in Build periodically irrespective of any code change the build will be executed in the defined time period and in Poll SCM it will execute only when there is some changes.

- Select Poll SCM and enter * * * * *

The screenshot shows the Jenkins 'Configuration' screen for a job named 'BuildAndDeploy'. On the left, a sidebar lists configuration sections: General, Source Code Management, Build Triggers (which is selected and highlighted in grey), Build Environment, Pre Steps, Build, Post Steps, Build Settings, and Post-build Actions. The main panel is titled 'Build Triggers' and contains several options:

- Build whenever a SNAPSHOT dependency is built ?
- Schedule build when some upstream has no successful builds ?
- Trigger builds remotely (e.g., from scripts) ?
- Build after other projects are built ?
- Build periodically ?
- GitHub Branches
- GitHub Pull Requests ?
- GitHub hook trigger for GITScm polling ?
- Poll SCM ?

 Below the 'Poll SCM' option is a 'Schedule' section with a dropdown menu set to 'xxxxx'. At the bottom of the panel are 'Save' and 'Apply' buttons.

Now whenever you made changes in code and push your code in GitHub the Jenkins will automatically build the job.

The screenshot shows a registration form titled 'Registration form' on a web browser. The URL in the address bar is 18.141.58.225:8080/webapp/?first=d&last=d&email=amanduggal103%40gmail.com&dob=2024-0. The form fields include:

- First Name:
- Last Name:
- Email:
- Date of Birth:
- Password:
- Re-type Password:
- Contact:
- Gender:
-

Setup Docker environment

Now we can deploy our code on Docker Container.



Setup Docker Host:

- Setup a Linux EC2 Instance.
- Install Docker.
- Start Docker Services.
- Basic docker commands.



Steps to follow:

- Create an EC2 instance for Docker.

Commands:

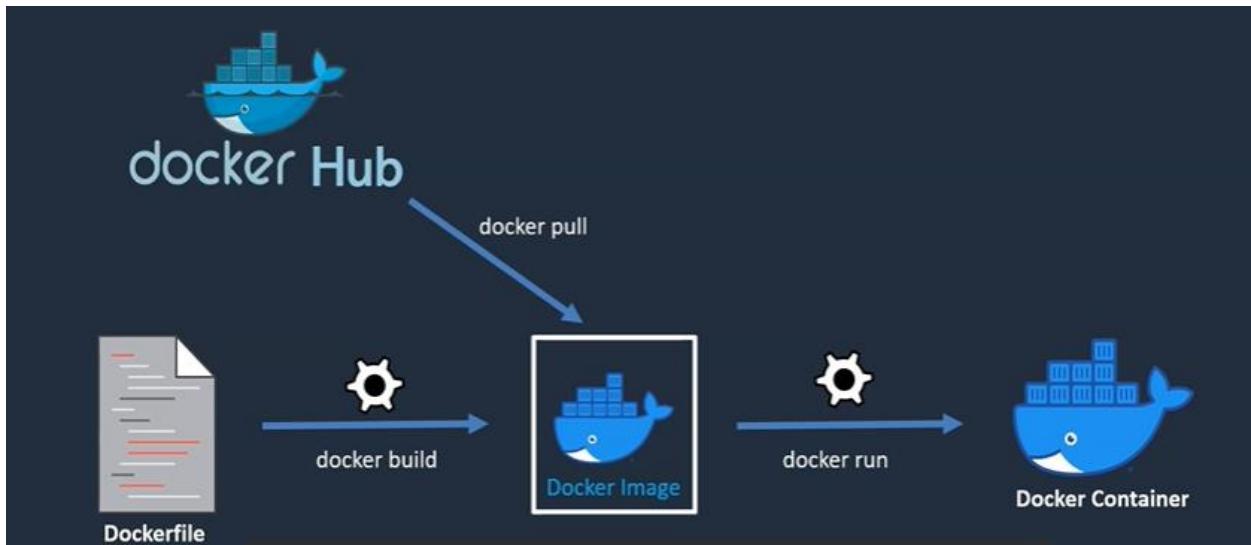
```
sudo yum update -y  
sudo yum upgrade -y  
sudo nano /etc/hostname --- #change the hostname to Jenkins master  
sudo init 6 --- #To restart the machine
```

Installing Docker

```
yum install docker -y  
docker --version
```

```
# start docker services  
service docker start  
service docker status
```

How to Create Docker Container



Docker pull tomcat

Docker images --#to view the images

```
[root@DockerServer ec2-user]# docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
tomcat          latest   405afe63d576  2 days ago   455MB
```

Now convert this image to container.

```
docker run -d --name tomcat-container -p 8081:8080 tomcat
```

By default docker run on port no. 8080, similar way we need to specify on which port no. container is running.

Container run inside the docker host, if you want to access it outside of the Docker host, then we need to map it to the external port, here we are using 8081.

It means that internally the tomcat is running on port number 8080, but externally we exposing it to the external network on port number 8081.

```
[root@DockerServer ec2-user]# docker run -d --name tomcat-container -p 8081:8080
tomcat
c2f3354c16e8f94f5b5alfe758676b05a502b8b2ab828d8970e7e01767cb2175
[root@DockerServer ec2-user]# docker ps
CONTAINER ID      IMAGE      COMMAND      CREATED      STATUS      PORTS
NAMES
c2f3354c16e8      tomcat     "catalina.sh run"    6 seconds ago   Up 5 seconds   0.0.
0.0:8081->8080/tcp, :::8081->8080/tcp      tomcat-container
[root@DockerServer ec2-user]#
```

Now you can access your container outside the docker host.

On browser 52.77.243.86:8081 (Public IP:Port No.)

HTTP Status 404 – Not Found

Type Status Report

Description The origin server did not find a current representation for the target resource or is not willing to disclose that one exists.

Apache Tomcat/10.1.19

To Fix this issue, get into the docker container.

```
docker exec -it tomcat-container /bin/bash
```

```
[root@DockerServer ec2-user]# docker exec -it tomcat-container /bin/bash
root@c2f3354c16e8:/usr/local/tomcat# ls
bin BUILDING.txt conf CONTRIBUTING.md lib LICENSE logs native-jni-lib NOTICE README.md RELEASE-NOTES RUNNING.txt temp webapps webapps.dist work
root@c2f3354c16e8:/usr/local/tomcat#
```

We are facing this issue since, whenever we try to access Tomcat from the browser it will look into the application on webapps directory.

But if you go inside the webapps directory you don't see any contents. But they are keeping all the information under the webapps.dist directory.

Now go inside the webapps.dist directory.

```
cd webapps.dist
```

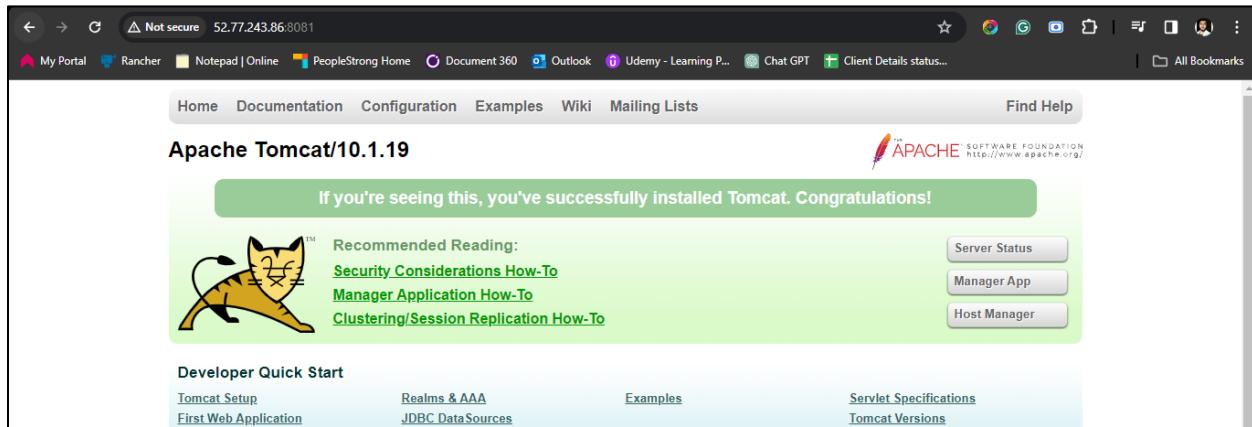
```
root@c2f3354c16e8:/usr/local/tomcat# cd webapps
root@c2f3354c16e8:/usr/local/tomcat/webapps# ls
root@c2f3354c16e8:/usr/local/tomcat/webapps# cd ..
root@c2f3354c16e8:/usr/local/tomcat# cd webapps.dist
root@c2f3354c16e8:/usr/local/tomcat/webapps.dist# ls
docs examples host-manager manager ROOT
root@c2f3354c16e8:/usr/local/tomcat/webapps.dist#
```

Copy the content into the webapps directory then only we can access our application.

```
cp -R * ../*webapps/ --#copying all files and directories from the current directory to the "webapps" directory.
```

```
root@c2f3354c16e8:/usr/local/tomcat/webapps.dist# cp -R * ../webapps/
root@c2f3354c16e8:/usr/local/tomcat/webapps.dist# cd ..
root@c2f3354c16e8:/usr/local/tomcat# cd webapps
root@c2f3354c16e8:/usr/local/tomcat/webapps# ls
docs examples host-manager manager ROOT
root@c2f3354c16e8:/usr/local/tomcat/webapps#
```

Now you can view your application.



The changes we have made to the container is temporary.

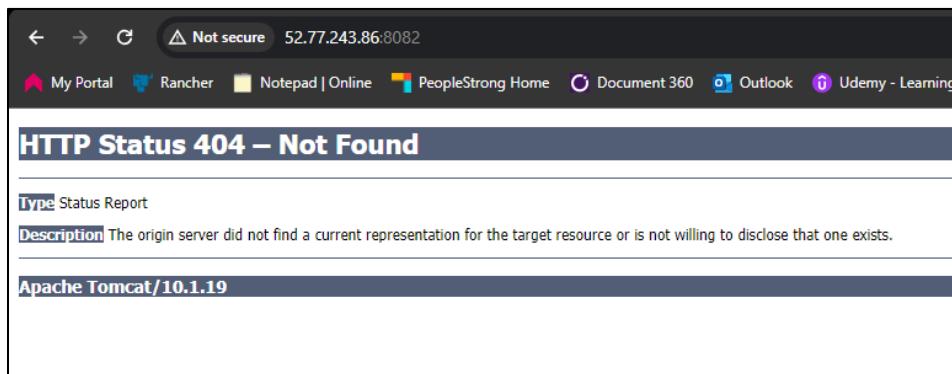
Let's look.

Exit --# To exit from existing container.

docker stop tomcat-container --#To stop the running container.

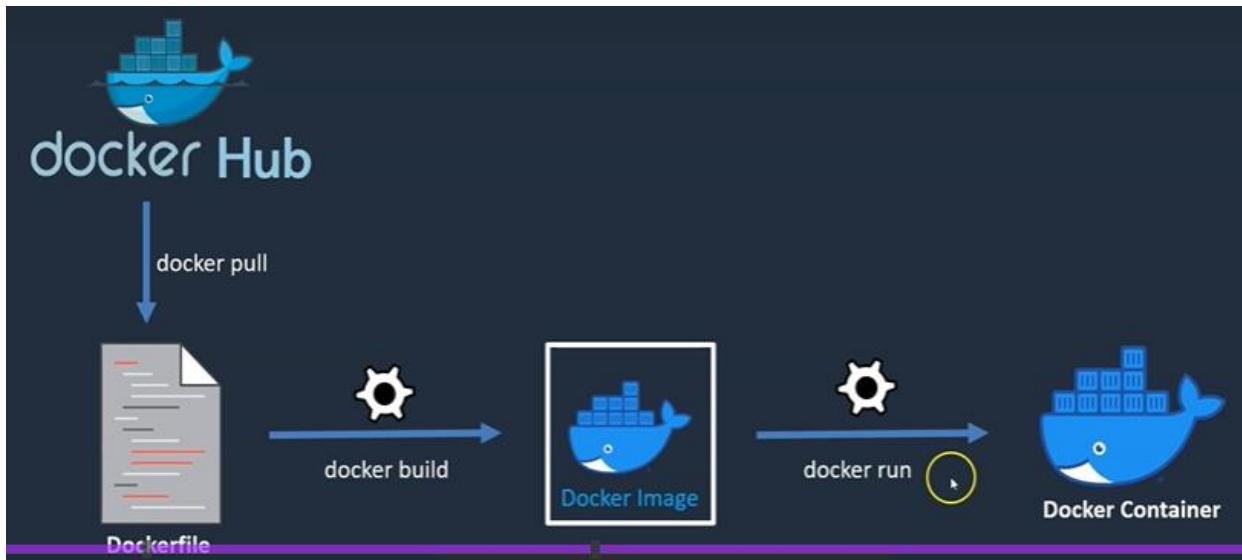
docker run -d --name tomcat2 -p 8082:8080 tomcat --#Create another container with the same image.

Now when you browse your container on browser you will get the same error, since we have fix the error only for the specific previous container.

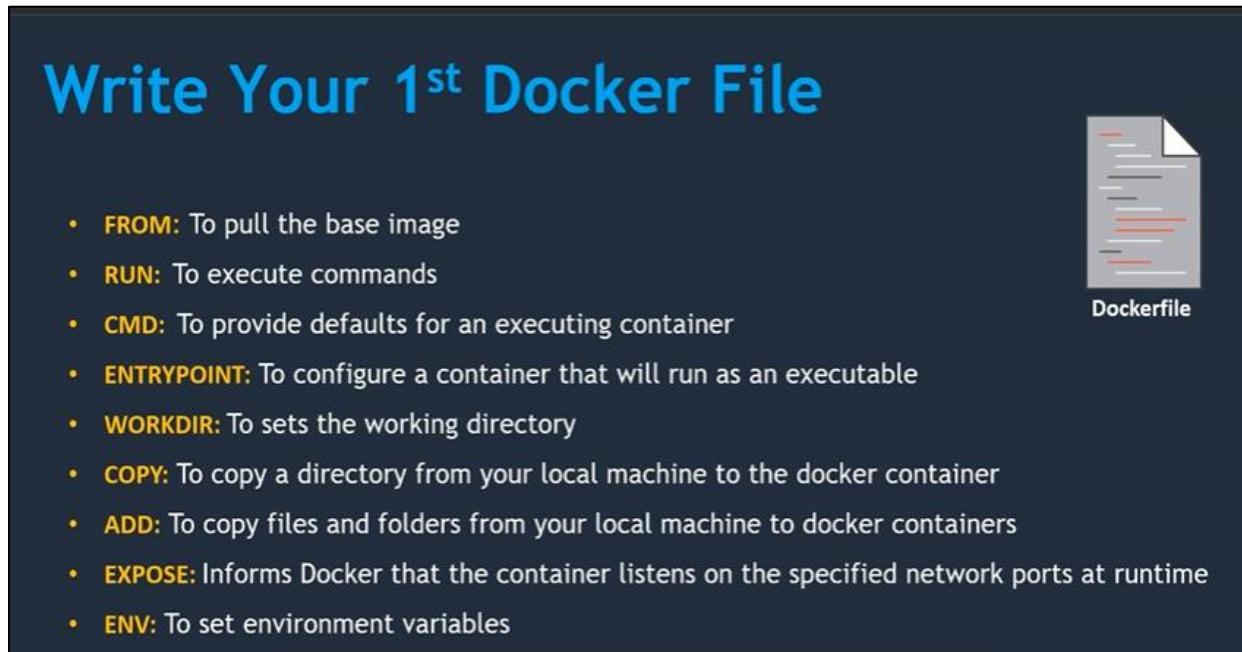


- To fix this permanently, we can create a docker file.
- Pull the tomcat image and will do the customization like copying the file from the webapp.dist to webapp so that the content will be accessible under webapps directory.
- Once, customization done we will create the docker image, so the changes will be captured.
- From the image we will launch the tomcat container.

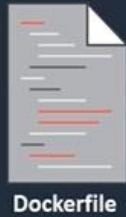
How to create Docker Container



Create a first Docker file



Install tomcat on Centos



- Pull centos from dockerhub - **FROM**
- Install java - **RUN**
- Create /opt/tomcat directory - **RUN**
- Change work directory to /opt/tomcat - **WORKDIR**
- Download tomcat packages - **ADD /RUN**
- Extract tar.gz file - **RUN**
- Rename to tomcat directory - **RUN**
- Tell to docker that it runs on port 8080 - **EXPOSE**
- Start tomcat services - **CMD**

--Note: This is the sample docker file for learning purpose we are using the centos instead of tomcat.
vi Dockerfile

```
FROM centos:latest

RUN cd /etc/yum.repos.d/
RUN sed -i 's/mirrorlist/#mirrorlist/g' /etc/yum.repos.d/CentOS-*
RUN sed -i 's|#baseurl=http://mirror.centos.org|baseurl=http://vault.centos.org|g'
/etc/yum.repos.d/CentOS-*

RUN yum -y install java
CMD /bin/bash
RUN mkdir /opt/tomcat
WORKDIR /opt/tomcat
ADD https://downloads.apache.org/tomcat/tomcat-9/v9.0.86/bin/apache-tomcat-9.0.86.tar.gz .
RUN tar -xvzf apache-tomcat-9.0.86.tar.gz
RUN mv apache-tomcat-9.0.86/* /opt/tomcat
EXPOSE 8080
CMD ["/opt/tomcat/bin/catalina.sh", "run"]
```

```

FROM centos:latest
RUN cd /etc/yum.repos.d/
RUN sed -i 's/mirrorlist/#mirrorlist/g' /etc/yum.repos.d/CentOS-
RUN sed -i 's|#baseurl=http://mirror.centos.org|baseurl=http://vault.centos.org|g' /etc/yum.repos.d/CentOS-
RUN yum -y install java
CMD /bin/bash
RUN mkdir /opt/tomcat
WORKDIR /opt/tomcat
ADD https://downloads.apache.org/tomcat/tomcat-9/v9.0.86/bin/apache-tomcat-9.0.86.tar.gz .
RUN tar -xvzf apache-tomcat-9.0.86.tar.gz
RUN mv apache-tomcat-9.0.86/* /opt/tomcat
EXPOSE 8080
CMD ["/opt/tomcat/bin/catalina.sh", "run"]
~
```

docker build -t mytomcat .

Open the browser and check publicip:portno, if the webpage is open or not.

Now create Docker file

```

FROM tomcat:latest
RUN cp -R /usr/local/tomcat/webapps.dist/* /usr/local/tomcat/webapps
```

```

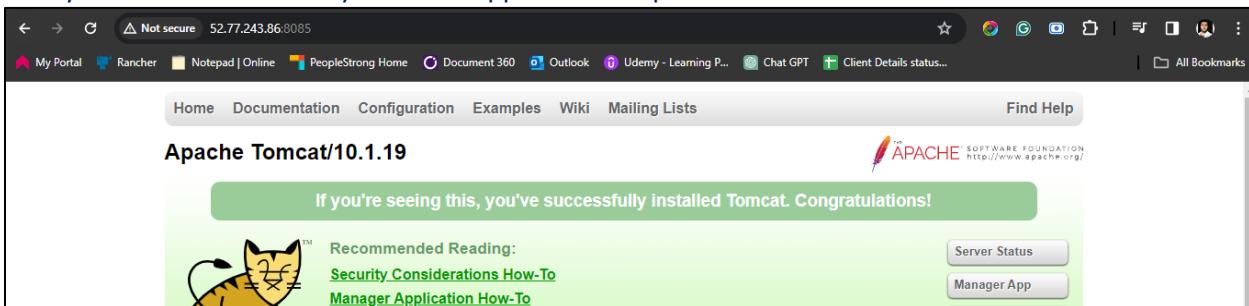
root@DockerServer:/home/ec2-user
FROM tomcat:latest
RUN cp -R /usr/local/tomcat/webapps.dist/* /usr/local/tomcat/webapps
~
```

docker build -t mytomcatserver .

docker images

docker run -d --name tomcatcontainer -p 8085:8080 mytomcatserver

Now you will see without any error the application is open.



Integrate Docker with Jenkins



- Create a dockeradmin user.
- Install “Publish Over SSH” plugin.
- Add Dockerhist to Jenkins “Configure systems”.

Commands:

cat /etc/passwd --#shows list of users.

cat /etc/group --#show list of groups.

Now, we should create a user and add him to the Docker group (Docker group is by default present)

useradd dockeradmin --#To add user.

passwd dockeradmin --#To create password for the user

(Enter the password)

id dockeradmin --#it shows the users belong to which group.

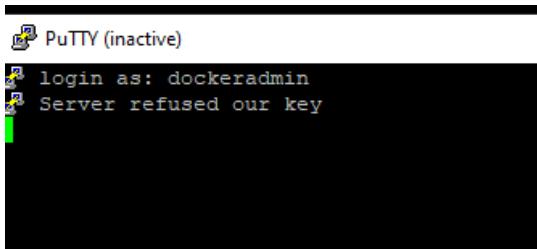
```
[root@DockerServer ec2-user]# id dockeradmin
uid=1001(dockeradmin) gid=1001(dockeradmin) groups=1001(dockeradmin)
[root@DockerServer ec2-user]#
```

By default, the dockeradmin user belongs to is default group i.e, dockeradmin. Now add the user to docker group.

usermod -aG docker dockeradmin --#This command will add the dockeradmin user to docker group.

Now, try to login with the dockeradmin user in docker server via putty.

Loginas: dockeradmin



Here, you are not allowed to login as dockeradmin user. This issue occurred, since the EC2 instances doesn't allow the password-based authentication. We should enable it.

--To enable it, Uncomment the **passwordauthentication yes** and comment the **passwordAuthentication no** in the **sshd_config** file.

```
vi /etc/ssh/sshd_config
```

```
# hostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes

# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication yes
#PermitEmptyPasswords no
#PasswordAuthentication no

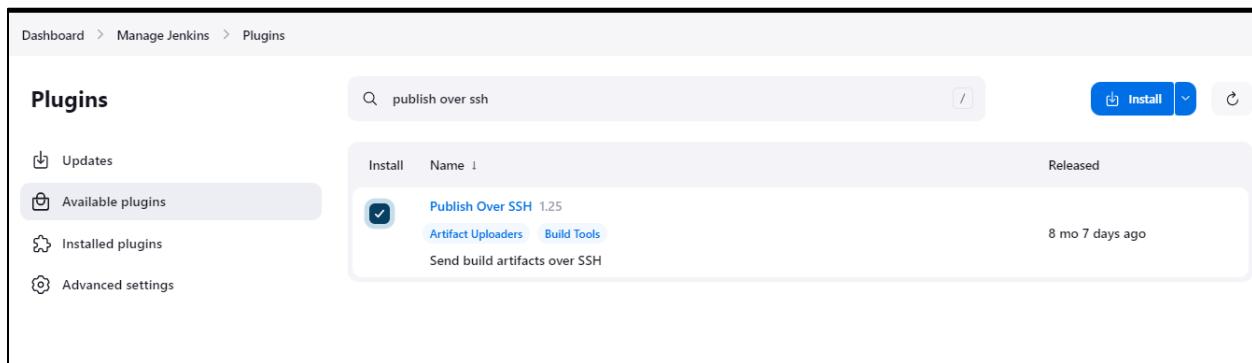
# Change to no to disable s/key passwords
#ChallengeResponseAuthentication yes
ChallengeResponseAuthentication no

# Kerberos options
#KerberosAuthentication no
#KerberosOrLocalPasswd yes
...
```

service sshd reload --#Reload the SSHD service.

Now login, with dockeradmin user you are able to login.

To integrate docker with the Jenkins.
Install the plugin "**Publish Over SSH**"



Manage Jenkins > System configuration > Publish Over SSH > SSH Server > Add

- **Name** - dockerhost
- **Hostname** - (Public Ip or private Ip docker server) (We can use both if they are in the same VPC)
- **Username** - dockeradmin --#The docker user which we have created
--#Click on Advance
Check mark -- Use password authentication, or use a different key
- **Passphrase / Password** - Ammy@567 --#Enter password

Click on test configuration.

SUCCESS -- Which means that I can able to authenticate from my Jenkins server to Docker host as a dockeradmin user.

- Apply > Save.

Dashboard > Manage Jenkins > System >

SSH Servers

SSH Server
Name ?
dockerhost
Hostname ?
172.31.25.42
Username ?
dockeradmin
Remote Directory ?

Advanced ^ Edited

Use password authentication, or use a different key ?

Passphrase / Password ?

.....



NOTE - Either you can use the password based authentication or the key based authentication. To see how we can generate the keys.

To Generate the Key -- Go to Terminal of docker

sudo su - dockeradmin --#Switch user to dockeradmin

ssh-keygen

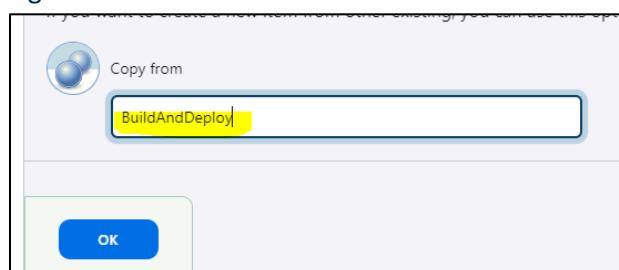
```
[dockeradmin@DockerServer ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/dockeradmin/.ssh/id_rsa):
Created directory '/home/dockeradmin/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/dockeradmin/.ssh/id_rsa.
Your public key has been saved in /home/dockeradmin/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:AAtSarAuox0hhipjxwGr3aeaK44yjXy5Wtkl+PaGgSs dockeradmin@DockerServer
The key's randomart image is:
+---[RSA 2048]---+
|ooo . |
|o+o. o |
|=+... . |
|*o.o. ...
|Booo...ooS
|=+..o+o..
|oo..oo...o
|*.oE ....
|+=*oo .. |
+---[SHA256]---+
[dockeradmin@DockerServer ~]$
```

The Keys gets generated in /home/dockeradmin

cd /home/dockeradmin

Till here the docker host is integrated with the Jenkins. Now we can create a new Jenkins job where we can build and deploy the artifacts on docker container.

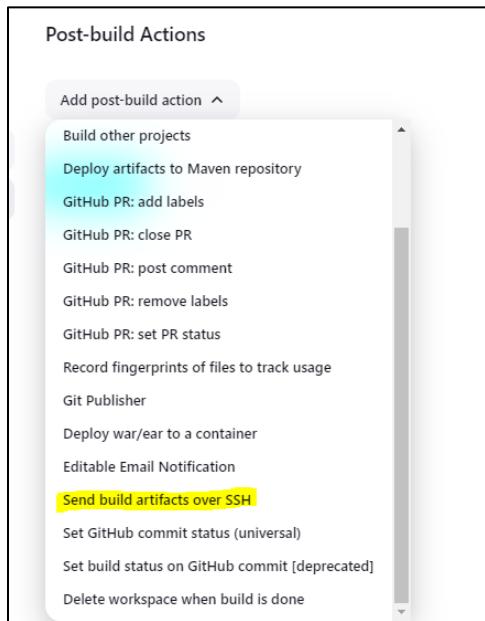
- Create a new job and copy the previous job i.e 'BuildAndDeploy' job to get all the previous job configurations.



- Delete the Existing Post-build Actions.



- Select 'Send build artifacts over SSH' in the Post-build Actions.



- **Source file** - webapp/target/*.war --#Enter the path where the .war file has been stored. By * means it copies all the .war file present in the target folder.
- **Remove Prefix** - webapp/target

Here we are copying the .war file and while copying the war file we don't require the prefix. We are copying the .war file to /home/dockeradmin location.

Post-build Actions

Send build artifacts over SSH ? X

SSH Publishers

SSH Server

Name ?

Advanced ▾

Transfers

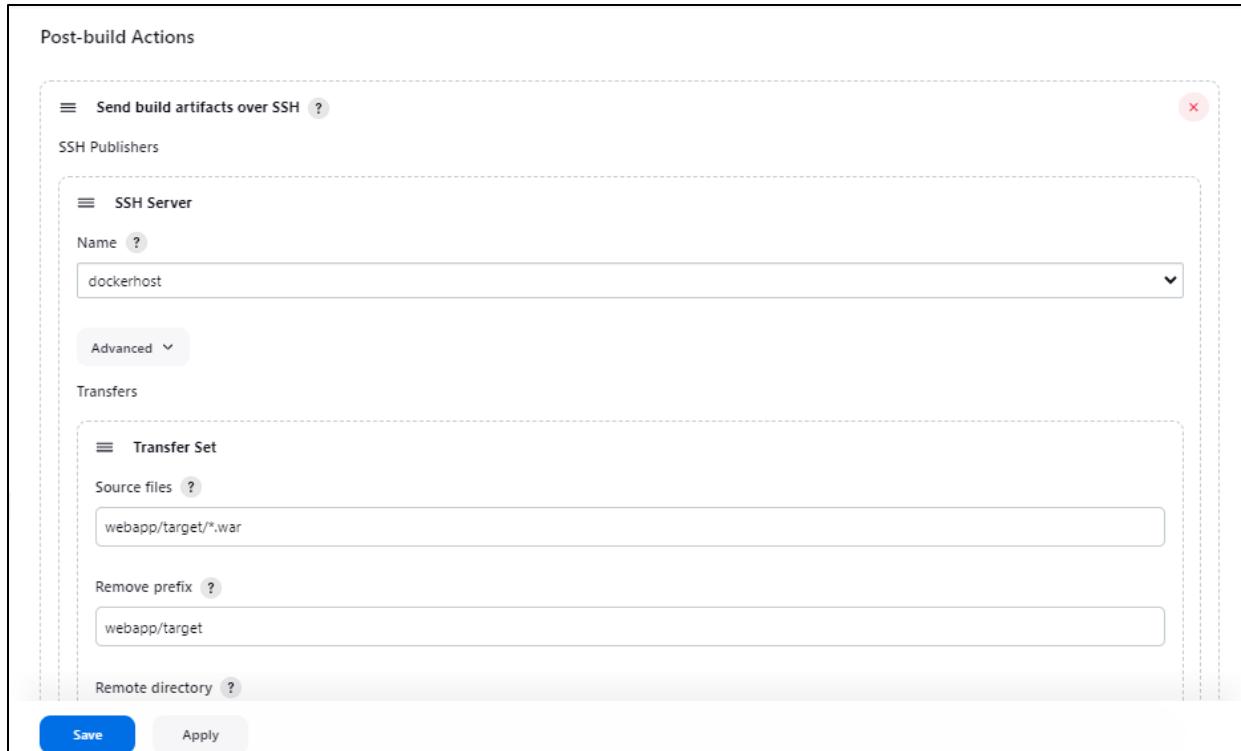
Transfer Set

Source files ?

Remove prefix ?

Remote directory ?

Save Apply



Build Now

Now check the docker terminal, and check whether the webapp.war file is copied or not.

```
[dockeradmin@DockerServer ~]$ ll
total 4
-rw-rw-r-- 1 dockeradmin dockeradmin 2582 Mar  9 18:41 webapp.war
[dockeradmin@DockerServer ~]$
```

This webapp.war file is owned by dockeradmin user and we would like to copy onto Docker Container. We can copy it manually. But what if we are copying it manually and if there are multiple builds how we make sure that your application is successfully deployed every time.

That is the reason we need to update our docker file to take this war file while creating a new container. That is how we can deploy our application along with the Tomcat.

Now create an Docker image along with the artifacts (war file) so that whenever we launch a new container it'll come up within your application.

Create a new directory to keep our Dockerfiles and artifacts, so that in future we can use that location to create our Docker images.

```
exit --#Exit from the dockeradmin user and come to the root user.
cd /opt
mkdir docker
```

```
[dockeradmin@DockerServer ~]$ exit
logout
[root@DockerServer ec2-user]# cd /opt
[root@DockerServer opt]# ls
aws  containerd  rh
[root@DockerServer opt]# mkdir docker
[root@DockerServer opt]# ls
aws  containerd  docker  rh
[root@DockerServer opt]#
```

Now give ownership of the docker directory to the dockeradmin user.

```
chown -R dockeradmin:dockeradmin docker
```

Now in the job, we will tell that instead of copying the artifacts into the dockeradmin home directory I would like to copy to the /opt/docker.

Basically, previously the webapp.war files are copied on the dockeradmin user only now we define the all the war files should be copied in the common location for which we created a docker directory and make the owner to dockeradmin user. By this we don't have manually copy the file from dockeradmin user to the root user.

Now move the Dockerfile which we have previously created to the /opt/docker directory.

```
cd /home/ec2-user/
mv Dockerfile /opt/docker/
```

Now Dockerfile is owned by the root and give the ownership of the Dockerfile to the dockeradmin as well.

```
cd /opt/docker/
chown -R dockeradmin:dockeradmin /opt/docker
[root@DockerServer docker]# ll
total 4
-rw-r--r-- 1 root root 90 Mar  9 14:28 Dockerfile
[root@DockerServer docker]# chown -R dockeradmin:dockeradmin /opt/docker
[root@DockerServer docker]# ll
total 4
-rw-r--r-- 1 dockeradmin dockeradmin 90 Mar  9 14:28 Dockerfile
[root@DockerServer docker]#
```

Now go to the Jenkins job and tell to don't copy the artifacts into the dockeradmin user.

Open the Job >Configure

Remote Directory - //opt//docker

The screenshot shows a configuration interface for a build step. On the left, a sidebar lists various configuration sections: General, Source Code Management, Build Triggers, Build Environment, Pre Steps, Build, Post Steps, Build Settings, and Post-build Actions. The 'Post-build Actions' section is currently selected and highlighted in grey. On the right, the configuration details for an 'SSH Server' named 'dockerhost' are displayed. Under the 'Transfers' section, a 'Transfer Set' is defined with 'Source files' set to 'webapp/target/*.war', 'Remove prefix' set to 'webapp/target', and 'Remote directory' set to '/opt/docker'. At the bottom of the screen, there are 'Save' and 'Apply' buttons.

Build Now

Now check the webapp.war file is copied or not?

```
[root@DockerServer docker]# ll
total 8
-rw-r--r-- 1 dockeradmin dockeradmin 90 Mar  9 14:28 Dockerfile
-rw-rw-r-- 1 dockeradmin dockeradmin 2582 Mar  9 19:28 webapp.war
```

Now, we need to create an container along with .war file. Then only we can access our application from the browser.

Edit the Dockerfile and mention to copy the .war file to the webapps directory.

```
COPY ./*.war /usr/local/tomcat/webapps
FROM tomcat:latest
RUN cp -R /usr/local/tomcat/webapps.dist/* /usr/local/tomcat/webapps
COPY ./*.war /usr/local/tomcat/webapps
~
```

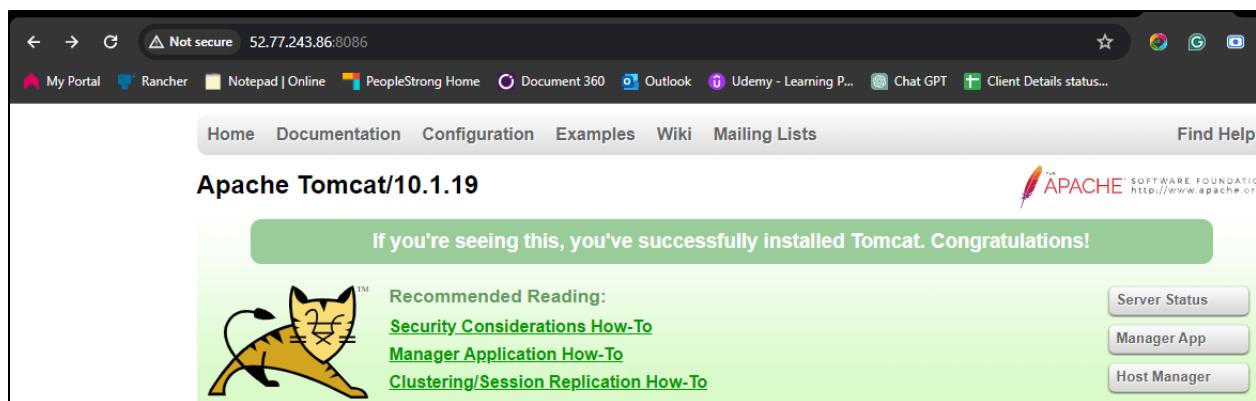
docker build -t tomcat.v1 . --#.represent to find the Dockerfile in the current directory.
docker images

```
[root@DockerServer docker]# docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
tomcat          v1       072d71c1ef07    6 seconds ago  460MB
demotomcat      latest   0ce94428fc68    5 hours ago   460MB
test1           latest   11d956292295    5 hours ago   511MB
mytomcat        latest   36dbd0afcfa2    6 hours ago   276MB
tomcat          latest   405afe63d576    3 days ago    455MB
centos          latest   5d0da3dc9764    2 years ago   231MB
[root@DockerServer docker]# docker run -d --name tomcatv1 -p 8086:8080 tomcat:v1
5ada457974a211f0b2dd47e79a11f71b8d4137a76812d631074435599e504589
[root@DockerServer docker]#
```

docker run -d --name tomacatv1 -p 8086:8080 tomcat:v1

Now check on browser, publicip:portno

52.77.243.86:8086



To View the application.

52.77.243.86:8086/webapp

First Name:	<input type="text" value="Enter your first name"/>
Last Name:	<input type="text" value="Enter your last name"/>
Email:	<input type="text" value="Enter your email"/>
Date of Birth:	<input type="text" value="mm/dd/yyyy"/> <input type="button" value="..."/>
Password:	<input type="password" value="Enter your password"/>
Re-type Password:	<input type="password" value="Re-Enter your password"/>
Contact:	<input type="text" value="Enter your Mobile Number"/>
Gender:	<input type="select" value="Male"/>
<input type="button" value="Submit"/>	

Here we can access our application through Docker container.

Till here, we created a job with the we could be able to build our code and copy the artifacts under the Docker host. But from there to create a container we have executed the commands like Docker build and Docker run.

This is not the right way, because we want to automate end-to-end pipeline. Means once we commit our code onto GitHub it should be able to build it and create artifact and copying the artifact onto Docker host creating a Docker image and creating a container out of it.

That is what should happen automatically.

Configure the Existing job '**BuildAndDeployContainer**'

Under Post-build Actions enter the predefined commands.

```
cd /opt/docker;
docker build -t regapp:v1 .;
docker run -d --name registerapp 8087:8080 regapp:v1
```

The screenshot shows the Jenkins job configuration page for 'BuildAndDeployContainer'. On the left, a sidebar titled 'Configure' lists various job settings: General, Source Code Management, Build Triggers, Build Environment, Pre Steps, Build, Post Steps, Build Settings, and Post-build Actions. The 'Post-build Actions' item is highlighted with a light gray background. The main panel is titled 'Transfers' and contains a 'Transfer Set' configuration. It includes fields for 'Source files' (set to 'webapp/target/*.war'), 'Remove prefix' (set to 'webapp/target'), 'Remote directory' (set to '//opt//docker'), and an 'Exec command' box containing the command: `cd /opt/docker;
docker build -t regapp:v1 .;
docker run -d --name registerapp 8087:8080 regapp:v1`. A note at the bottom of the transfer set panel states: 'All of the transfer fields (except for Exec timeout) support substitution of Jenkins environment variables'. At the bottom of the configuration panel are 'Save' and 'Apply' buttons.

Here, once the artifact has been copied, go inside the /opt/docker and build the image. Once Image is build, create a container out of it.

Now, before executing the job, stop and remove all the previous docker containers and images.

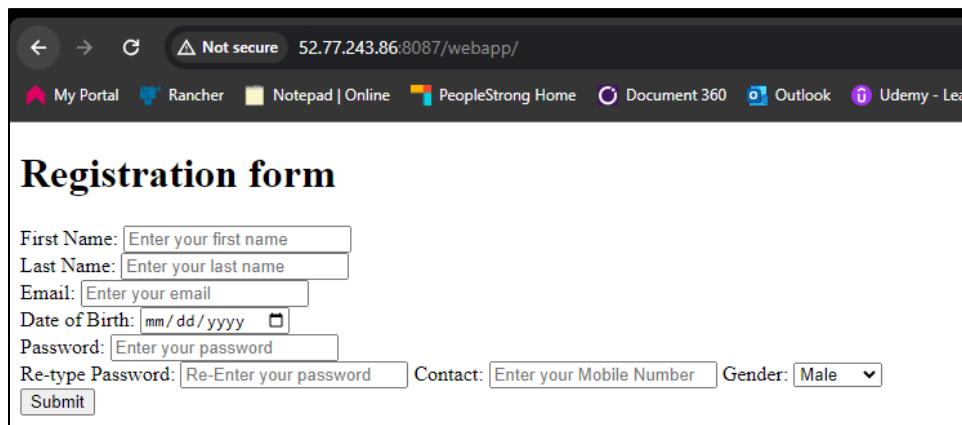
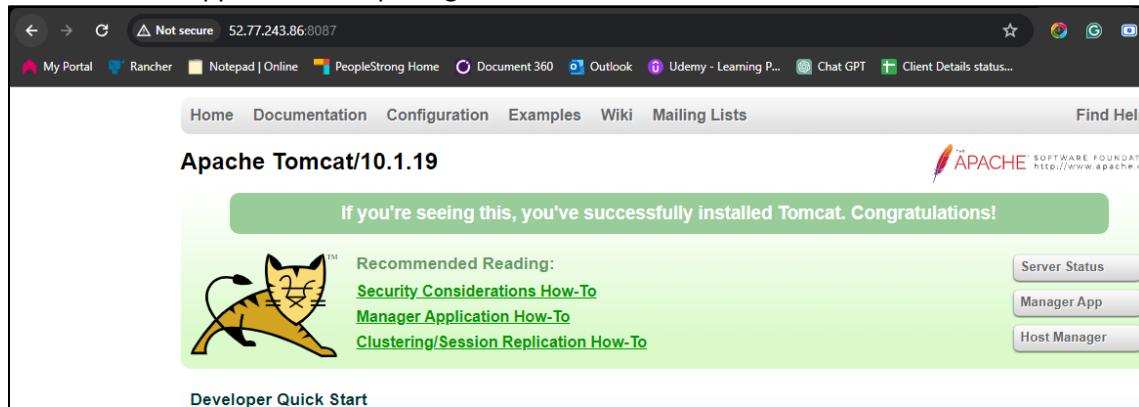
```
docker stop $(docker ps -q) --#To stop all the running containers at one go.  
docker container prune --# To remove/ delete all the container in one go.  
docker image prune -a --#To remove/delete all the images in one go.
```

Now execute the job.

Now you see the **webapp.war** file is updated by the time and all images and containers are created automatically without giving any command.

```
[root@DockerServer docker]# ll  
total 8  
-rw-r--r-- 1 dockerdocker dockeradmin 129 Mar  9 19:34 Dockerfile  
-rw-rw-r-- 1 dockerdocker dockeradmin 2582 Mar  9 20:17 webapp.war  
[root@DockerServer docker]# docker images  
REPOSITORY TAG IMAGE ID CREATED SIZE  
regapp v1 749506236639 About a minute ago 460MB  
tomcat latest 405afee3d576 3 days ago 455MB  
[root@DockerServer docker]# docker ps -a  
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES  
a963bcc85840 regapp:v1 "catalina.sh run" About a minute ago Up About a minute 0.0.0.0:8087->8080/tcp, :::8087->8080/tcp registerapp  
[root@DockerServer docker]# docker ps  
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES  
a963bcc85840 regapp:v1 "catalina.sh run" About a minute ago Up About a minute 0.0.0.0:8087->8080/tcp, :::8087->8080/tcp registerapp  
[root@DockerServer docker]#
```

Check the web application its opening.



Now let's try to update the hello-world program (source-code) from git and let's see if our jenkins job should trigger automatically without any manual intervention.

Open the Git Bash.

```
cd /d/AWS/DevopsProject/hello-world/webapp/src/main/webapp  
vi index.jsp  
--#Edit the code or just change the Heading
```

```
git status  
git add .  
git commit -m "updated index.jsp file"  
git push origin master
```

```
Aman.Duggal@AmanD-OEG MINGW64 /d/AWS/DevopsProject/hello-world/webapp/src/main/w  
ebapp (master)  
$ git commit -m "Updated index.jsp file"  
[master 3b6a759] Updated index.jsp file  
Committer: Aman Duggal <Aman.Duggal@businessnext.com>  
Your name and email address were configured automatically based  
on your username and hostname. Please check that they are accurate.  
You can suppress this message by setting them explicitly. Run the  
following command and follow the instructions in your editor to edit  
your configuration file:  
  
git config --global --edit  
  
After doing this, you may fix the identity used for this commit with:  
  
git commit --amend --reset-author  
  
1 file changed, 1 insertion(+), 1 deletion(-)  
  
Aman.Duggal@AmanD-OEG MINGW64 /d/AWS/DevopsProject/hello-world/webapp/src/main/w  
ebapp (master)  
$ git push origin master  
Enumerating objects: 13, done.  
Counting objects: 100% (13/13), done.  
Delta compression using up to 8 threads  
Compressing objects: 100% (5/5), done.  
Writing objects: 100% (7/7), 578 bytes | 578.00 KiB/s, done.  
Total 7 (delta 2), reused 0 (delta 0), pack-reused 0  
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.  
To https://github.com/amanduggal001/hello-world.git  
 d36fe04..3b6a759 master -> master  
  
Aman.Duggal@AmanD-OEG MINGW64 /d/AWS/DevopsProject/hello-world/webapp/src/main/w
```

The job will get failed, because in the post build action of the job we have defined the container name i.e. regapp, so whenever on the next build the job is trying to create the new container with the same name the conflict error occurred since the regapp container is already created.

To overcome this we can add the command to stop and delete the existing container before creating it.

```
cd /opt/docker;  
docker build -t regapp:v1 .;  
docker stop registerapp;  
docker rm registerapp;  
docker run -d --name registerapp -p 8087:8080 regapp:v1
```

Dashboard > BuildAndDeployContainer > Configuration

Configure

- General
- Source Code Management
- Build Triggers
- Build Environment
- Pre Steps
- Build
- Post Steps
- Build Settings
- Post-build Actions**

Remove prefix ? webapp/target

Remote directory ? //opt//docker

Exec command ?

```
cd /opt/docker;
docker build -t regapp:v1 .;
docker stop registerapp;
docker rm registerapp;
docker run -d --name registerapp -p 8087:8080 regapp:v1
```

All of the transfer fields (except for Exec timeout) support substitution of [Jenkins environment variables](#)

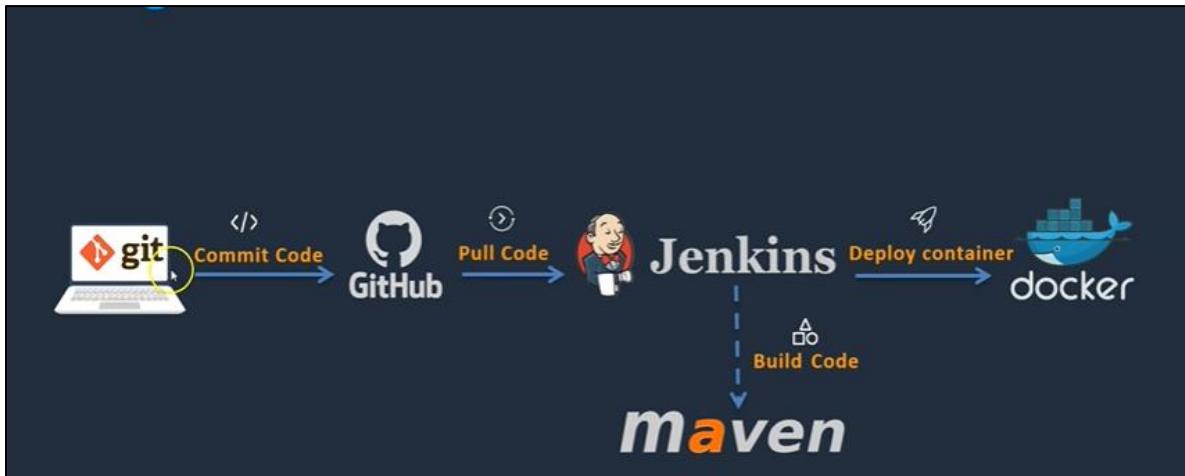
Advanced ▾

Now again make any changes in git and check if the job is automatically triggered without any error.

Till now we learn 't how to build and deploy the code by using jenkins. But if you see, this jenkins job looks ugly because we are trying to execute multiple commands. It is not right way to do this one.

That is where we can take advantage of deployment tools.

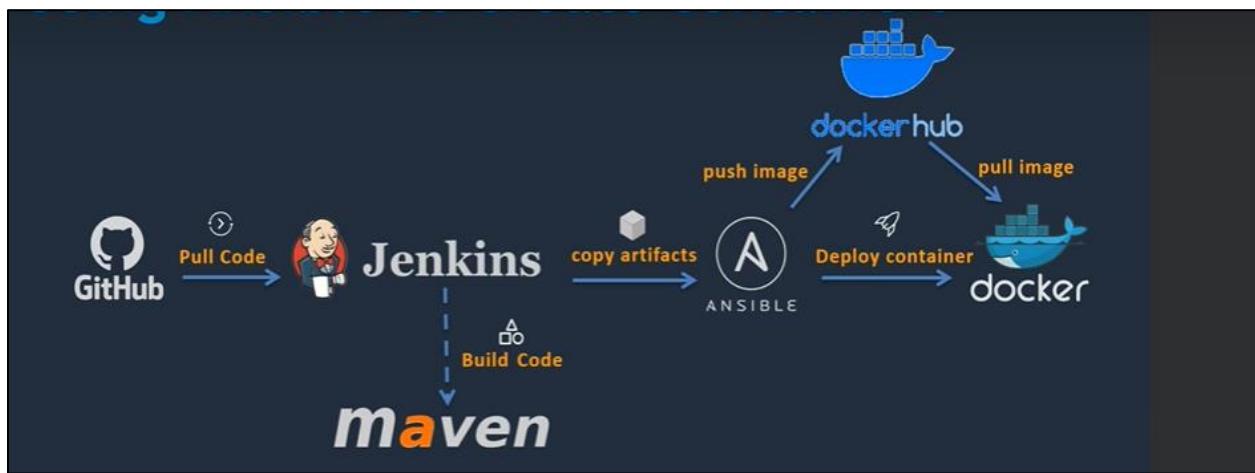
We can use Ansible as a deployment tool in our case. By Ansible we can execute all these commands with more efficient way.



Until now, we were doing changes on our workstation, then committing those changes with help of Git onto GitHub. Once the latest code is available on Git, Jenkins could able to pull the latest code and build it with the help of Maven and it was creating Docker Images and also deploying Docker containers, and Docker host. In this case we could see Jenkins as a build and Deployment tool.

But is that any better way to handle it. Yes, that is where we can introduce deployment tool.

Setup Ansible



We can use Ansible as a deployment tool so that Jenkins need not to do the administrative kind of activities because Jenkins is more efficiently work as a build tool.

Along with Ansible we are using the docker hub. Our applications are containerized at this moment. We need a repository in case it is a non-containerized applications means the applications which we directly deploy on VM.

With the help of Ansible and Docker Hub we are going to make this CI-CD pipeline smoother.

In this case Jenkins is going to take the code from GitHub and build artifacts and copy those artifacts onto Ansible server. Now it is Ansible task to create images and deploy the containers.

Ansible is going to take the Artifacts and with the help of Docker file it creates a Docker image.

This Docker image we can commit it into the Docker Hub because Docker Hub is a repository to store the Docker images.

Now, whenever we execute any Ansible playbook to deploy a container this Docker host communicates with Docker hub that pull the image whatever we mentioned in our playbook and create a container out of it.

Setup Ansible Server:

- Setup EC2 instance.
 - Setup **Hostname**.
 - Create **ansibleadmin** user.
 - Add user to the **sudoers** file.
 - Generate **SSH keys**.
 - Enable **Password based login**.
 - Install Ansible.



Steps to follow:

Create an EC2 instance

```
useradd ansibleadmin --#To Create a new user.  
passwd ansibleadmin --#To setup the password.
```

Now add this user in the sudoers group to provide the admin privileges.

visudo

Add the user in the visudo file.

```
#* SERVICE MANAGEMENT UPPTS AND MOUNT  
# %sys ALL = NETWORKING, SOFTWARE, SERVICES, STORAGE, DELEGATING, PRIVATE, DRIVERS  
  
## Allows people in group wheel to run all commands  
%wheel ALL=(ALL)          ALL  
  
## Same thing without a password  
# %wheel          ALL=(ALL)          NOPASSWD: ALL  
ansbibleadmin    ALL=(ALL)          NOPASSWD: ALL  
## Allows members of the users group to mount and umount the  
## cdrom as root  
# %users  ALL=/sbin/mount /mnt/cdrom, /sbin/umount /mnt/cdrom
```

Now enable the password based authentication.

```
vi /etc/ssh/sshd_config
```

--#Uncomment the PasswordAuthentication yes

```
# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication yes
#PermitEmptyPasswords no
#PasswordAuthentication no

# Change to no to disable s/key passwords
#ChallengeResponseAuthentication yes
ChallengeResponseAuthentication no
```

```
service sshd reload --# To reload the SSHD service.  
sudo su - ansibleadmin --#Switch user to ansibleadmin.
```

Now generate ssh keys.
ssh-keygen
exit --#Come back to the root user.

amazon-linux-extras install ansible2 - --#Install ansible packages.
python --version --#Check if the python is installed or not as ansible required the python.
ansible --version --# Check the ansible version.

```
[root@AnsibleServer ec2-user]# python --version  
Python 2.7.18  
[root@AnsibleServer ec2-user]# ansible --version  
ansible 2.9.23  
  config file = /etc/ansible/ansible.cfg  
  configured module search path = [u'/root/.ansible/plugins/modules', u'/usr/share/ansible/plugins/modules']  
  ansible python module location = /usr/lib/python2.7/site-packages/ansible  
  executable location = /bin/ansible  
  python version = 2.7.18 (default, Dec 18 2023, 22:08:43) [GCC 7.3.1 20180712 (Red Hat 7.3.1-17)]  
[root@AnsibleServer ec2-user]#
```

Till now, we have successfully installed the ansible, now the next thing is we need to prepare our Ansible system to create docker images.

Integrate Docker with Ansible

We are going to write a playbook, that playbook is going to tell our DockerHost how to create a container.

Manage Docker Host with Ansible

- | | |
|--|---|
| <ul style="list-style-type: none">• On Docker Host<ul style="list-style-type: none">• Create ansadmin• Add ansadmin to sudoers files• Enable password based login | <ul style="list-style-type: none">• On Ansible Node<ul style="list-style-type: none">• Add to hosts file• Copy ssh keys• Test the connection |
|--|---|



For this we need to add our DockerHost as a client in Ansible system.

On Docker Host -

We need to perform some steps in docker host, first we create a user in docker host '**ansibleadmin**' same as we created in the ansible host.

Add the user to the sudoers group and enable the password-based authentication.

Same steps that we have done on the ansible server, because with the ansibleadmin user itself Ansible is going to manage our DockerHost.

```
useradd ansibleadmin --#To Create a new user.  
passwd ansibleadmin --#To setup the password.
```

Now add this user in the sudoers group to provide the admin privileges.

```
visudo
```

Add the user in the visudo file.

```
## Allows people in group wheel to run all commands  
%wheel    ALL=(ALL)        ALL  
  
## Same thing without a password  
# %wheel      ALL=(ALL)        NOPASSWD: ALL  
ansibleadmin  ALL=(ALL)        NOPASSWD: ALL  
## Allows members of the users group to mount and umount the  
## cdrom as root  
# %users      ALL=/sbin/mount /mnt/cdrom, /sbin/umount /mnt/cdrom
```

Now enable the password based authentication.

```
vi /etc/ssh/sshd_config
```

```
--#Uncomment the PasswordAuthentication yes
```

```
# To disable tunneled clear text passwords, change to no here!  
PasswordAuthentication yes  
#PermitEmptyPasswords no  
#PasswordAuthentication no  
  
# Change to no to disable s/key passwords  
#ChallengeResponseAuthentication yes  
ChallengeResponseAuthentication no
```

```
service sshd reload --# To reload the SSHD service.
```

On Ansible node-

- We need to add the **DockerHost IP** address in the inventory file (host file).
- We need to copy the SSH Keys, we will be copying the Ansible server public key onto Dockerhost **ansibleadmin** user so that the passwordless authentication get enabled.
- At last we can test the connection by executing some ping commands.

```
vi /etc/ansible/hosts  
--#Add the private ip of the Docker host in the host file of the ansible.
```

```
# here's another example of host ranges, this time there are  
# leading Os:  
  
## db-[99:101]-node.exa  
#  
#  
  
172.31.25.42
```

Now we need to copy our **ansibleadmin** user keys onto the target ansibleadmin user account (docker host).

On ansible node -

```
sudo su - ansibleadmin  
cd .ssh  
ls
```

You will see the public and private keys.

```
[ansibleadmin@AnsibleServer ~]$ cd .ssh  
[ansibleadmin@AnsibleServer .ssh]$ ls  
id_rsa  id_rsa.pub  
[ansibleadmin@AnsibleServer .ssh]$
```

Now copy this public key on the target system (docker host).

```
ssh-copy-id 172.31.25.42 --#Enter the private IP of the docker host.
```

```
[ansibleadmin@AnsibleServer .ssh]$ ssh-copy-id 172.31.25.42  
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/ansibleadmin/.ssh/id_rsa.pub"  
The authenticity of host '172.31.25.42 (172.31.25.42)' can't be established.  
ECDSA key fingerprint is SHA256:08ITcU0lTbsUg6ftOvnVeouP0pvqt/sQQsf9Q5PDw.  
ECDSA key fingerprint is MD5:74:35:bb:d8:32:50:14:16:fe:54:ae:32:78:71:1d:10.  
Are you sure you want to continue connecting (yes/no)? yes  
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed  
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys  
ansibleadmin@172.31.25.42's password:  
  
Number of key(s) added: 1  
  
Now try logging into the machine, with: "ssh '172.31.25.42'"  
and check to make sure that only the key(s) you wanted were added.
```

Now, check on the docker host the authorized_keys should be added. The authorized_keys file carries the public key of the ansible host.

```
[ansibleadmin@DockerServer .ssh]$ ls  
id_rsa id_rsa.pub  
[ansibleadmin@DockerServer .ssh]$ ls  
authorized_keys id_rsa id_rsa.pub  
[ansibleadmin@DockerServer .ssh]$
```

Now, test the connection between the ansible and the docker host.

ansible all -m ping --#This means, whatever hosts are there in the inventory file, try to connect to all those systems.

```
[ansibleadmin@AnsibleServer .ssh]$ ansible all -m ping  
[WARNING]: Platform linux on host 172.31.25.42 is using the  
could change this. See https://docs.ansible.com/ansible/2.9,  
172.31.25.42 | SUCCESS => {  
    "ansible_facts": {  
        "discovered_interpreter_python": "/usr/bin/python"  
    },  
    "changed": false,  
    "ping": "pong"  
}  
[ansibleadmin@AnsibleServer .ssh]$
```

Now ansible could be able to communicate with our DockerHost, without any credentials nothing but with the Passwordless Authentication.

Integrate Ansible with Jenkins



Now integrate Ansible with Jenkins, so that Jenkins can able to copy artifacts onto Ansible systems, Ansible can able to create image or it can deploy the containers on the Docker host.

By doing this, we can delegate the activities, like Jenkins can able to do only build activities, Ansible can take care of deployment activities.

Login to Jenkins server > Manage Jenkins > System configuration > Publish over SSH > Add.

Name - ansible-server

Hostname - Private Ip of the ansible server --#To check the private ip on the terminal use **ifconfig** command.

Username - ansibleadmin

Click on advance > Select '**Use password authentication, or use a different key**'

Password - Ammy@567 --#Password of ansibleadmin user.

The screenshot shows the Jenkins 'Manage Jenkins' interface under the 'System' section. A new SSH server configuration is being created. The 'Name' field contains 'ansible-server'. The 'Hostname' field contains '172.31.19.187'. The 'Username' field contains 'ansibleadmin'. There is a 'Remote Directory' field which is currently empty. Below these fields is a checkbox labeled 'Avoid sending files that have not changed' which is unchecked. Under the 'Advanced' section, there is a checked checkbox 'Use password authentication, or use a different key'. A password input field is shown below it, with a lock icon and the word 'Concealed'. At the bottom of the form are 'Save', 'Apply', and 'Change Password' buttons.

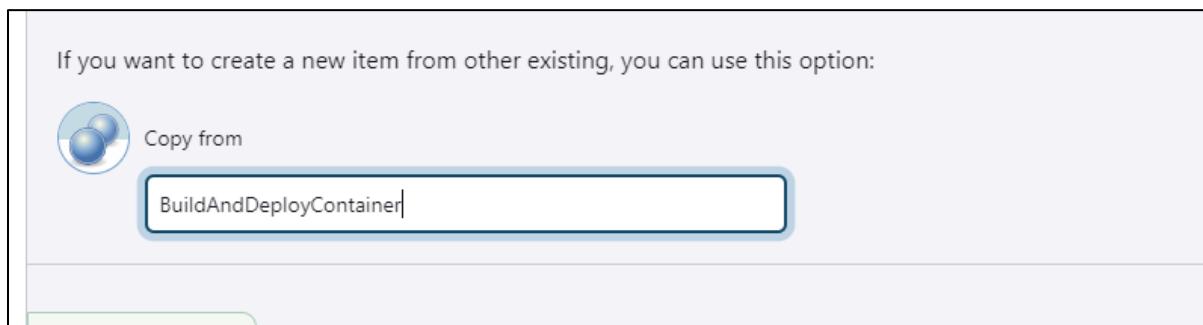
Till here, the integration of Ansible with Jenkins is successful.

Now we are going to create a new Jenkins job where we can build and copy the artifacts onto ansible system.

Create new item in Jenkins GUI.

Job Name - Copy_Artifacts_Onto_Anible

Copy the configurations from the previous job.



Disable the Poll SCM for now, we can enable it later whenever we need.

The screenshot shows a list of polling options under the 'Polling' section of a Jenkins job configuration. The 'Poll SCM' option is highlighted with a blue underline. Other options listed are 'GitHub Pull Requests' and 'GitHub hook trigger for GITScm polling'. Below this section is a 'Build Environment' section containing 'Delete workspace before build starts' and 'Use secret text(s) or file(s)'.

Select ansible-server under the SSH Server.

The screenshot shows the 'Post-build Actions' section of a Jenkins job. It includes a 'Send build artifacts over SSH' action and an 'SSH Publishers' section. Under 'SSH Publishers', there is a 'SSH Server' entry with a 'Name' field containing 'ansible-server'.

Remove the existing commands in the Exec command field. We can write the playbook for those commands in the ansible server.

The screenshot shows the 'Transfers' section of a Jenkins job. It includes a 'Transfer Set' configuration with fields for 'Source files' (containing 'webapp/target/*.war'), 'Remove prefix' (containing 'webapp/target'), 'Remote directory' (containing '//opt//docker'), and an 'Exec command' field which is currently empty. The 'Exec command' field is highlighted with a yellow background.

Apply > Save.

NOTE: Since the Remote directory i.e. **//opt//docker** mentioned in the job doesn't exists in the ansible server. Let's create this directory on ansible server.

On Ansible node -

```
sudo su - ansibleadmin
```

```
cd /opt
```

```
mkdir docker
```

```
||
```

```
~~ [ansibleadmin@AnsibleServer opt]$ ll
~~total 0
drwxr-xr-x 4 root root 33 Feb 23 05:26 aws
drwxr-xr-x 2 root root 6 Mar 10 11:23 docker
drwxr-xr-x 2 root root 6 Aug 16 2018 rh
[ansibleadmin@AnsibleServer opt]$
```

Now the docker directory is created but it is owned by the root user. The problem here is if it is owned by the root user while copying artifacts from Jenkins server it uses ansibleadmin user, so you will get the permission issue.

To overcome the problem, we must grant access to this directory to ansibleadmin user.

For that we need to change ownership.

```
sudo chown ansibleadmin:ansibleadmin docker
```

```
[ansibleadmin@AnsibleServer ~]$ cd /opt
[ansibleadmin@AnsibleServer opt]$ ll
total 0
drwxr-xr-x 4 root root 33 Feb 23 05:26 aws
drwxr-xr-x 2 root root 6 Mar 10 11:23 docker
drwxr-xr-x 2 root root 6 Aug 16 2018 rh
[ansibleadmin@AnsibleServer opt]$ sudo chown ansibleadmin:ansibleadmin docker
[ansibleadmin@AnsibleServer opt]$ ll
total 0
drwxr-xr-x 4 root         root         33 Feb 23 05:26 aws
drwxr-xr-x 2 ansibleadmin ansibleadmin 6 Mar 10 11:23 docker
drwxr-xr-x 2 root         root         6 Aug 16 2018 rh
[ansibleadmin@AnsibleServer opt]$
```

Now Build the job.

Once the job is successfully completed you can see the war file under the docker folder.

```
[ansibleadmin@AnsibleServer opt]$ cd docker  
[ansibleadmin@AnsibleServer docker]$ ls  
webapp.war  
[ansibleadmin@AnsibleServer docker]$
```

It means, we successfully copy the artifacts onto Ansible system. Now our Ansible job is creating an image by using this war file and push it into the Docker hub.

Build an image and create container on Ansible

Install docker into the ansible node.

```
sudo su -ansibleadmin  
sudo yum install docker -y
```

Now add ansibleadmin to the docker group, then only we can execute docker commands as ansibleadmin.

```
sudo usermod -aG docker ansibleadmin  
id ansibleadmin --#To check the group.
```

```
[ansibleadmin@AnsibleServer docker]$ sudo usermod -aG docker ansibleadmin  
[ansibleadmin@AnsibleServer docker]$ ll  
total 4  
-rw-rw-r-- 1 ansibleadmin ansible 2606 Mar 10 11:35 webapp.war  
[ansibleadmin@AnsibleServer docker]$ id ansibleadmin  
uid=1001(ansibleadmin) gid=1001(ansibleadmin) groups=1001(ansibleadmin),992(docker)  
[ansibleadmin@AnsibleServer docker] $
```

Now build the docker image, for docker image we need the docker file.

We have already created a Docker file on our docker host. So we can get it from there, instead of creating new docker file.(We are just copying the content of the docker file from the docker host and creating new docker file in the ansible host)

On Docker Node-

```
cd /opt/docker  
cat Dockerfile  
--# Copy the content.
```

On Ansible Node (under ansibleadmin user) -

```
sudo vi Dockerfile
```

```
FROM tomcat:latest  
RUN cp -R /usr/local/tomcat/webapps.dist/* /usr/local/tomcat/webapps
```

```
COPY ./*.war /usr/local/tomcat/webapps
```

```
ansibleadmin@AnsibleServer:/opt/docker
FROM tomcat:latest
RUN cp -R /usr/local/tomcat/webapps.dist/* /usr/local/tomcat/webapps
COPY ./*.war /usr/local/tomcat/webapps
~
```

Now we need to create an image of the Docker file.

```
sudo docker build -t regapp:v1 .
```

```
[ansibleadmin@AnsibleServer docker]$ sudo docker build -t regapp:v1 .
Sending build context to Docker daemon 5.632kB
Step 1/3 : FROM tomcat:latest
latest: Pulling from library/tomcat
23828d760c7b: Pull complete
e2670537dceb: Pull complete
c44aaba36909: Pull complete
c3b64924ef70: Pull complete
737fef02af8d: Pull complete
5d177a8f8e88: Pull complete
018443d5073c: Pull complete
3694d382f9eb: Pull complete
Digest: sha256:c83cd6449a4d34d6c5a9fell934c15a09c4762a2b0db46c916a4fba032e6b117
Status: Downloaded newer image for tomcat:latest
--> 405afe63d576
Step 2/3 : RUN cp -R /usr/local/tomcat/webapps.dist/* /usr/local/tomcat/webapps
--> Running in a14c897c1609
Removing intermediate container a14c897c1609
--> 07f0cc1e60d8
Step 3/3 : COPY ./*.war /usr/local/tomcat/webapps
--> b8a783ale25a
Successfully built b8a783ale25a
Successfully tagged regapp:v1
```

NOTE - If permission denied message come while build the docker image provide the privilege using the below command

```
sudo chmod 777 /var/run/docker.sock
```

```
docker images
```

```
docker run -it --name amancontainer -p 8081:8080 regapp:v1 --#This will create the container.
```

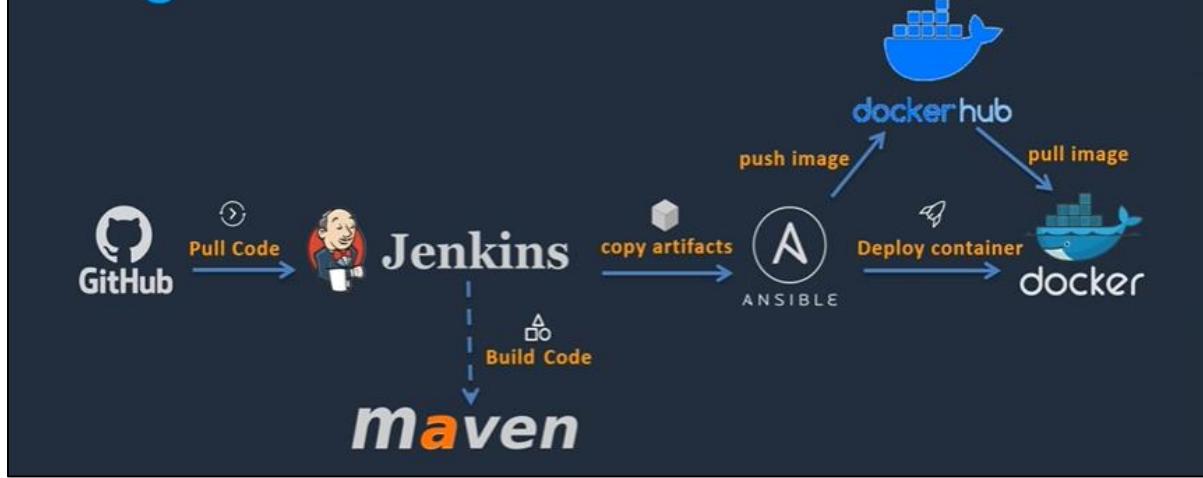
Now validate on browser if the application is opening.

```
Publicip of ansible:portno/webapp
```

The application is opening successfully. But here we are providing the manual commands on our Ansible.

Now next, we can create the Ansible Playbook which can do all these activities.

Ansible playbook to create image and container



So far, we have created an Ansible server, and we have integrated it with Jenkins so Jenkins could able to copy artifacts. With those artifacts we have created a Docker image by using manual commands.

Now this image is available on Ansible system, now how can we make available to our target environment, which means the Docker host, that is what the Docker Hub comes to our picture.

If we use Docker Hub, we can push our image onto Docker Hub.

This Docker Hub can be accessible by any Docker system.

Now, from Ansible we can instruct to this Docker host to go and pull the image from the Docker Hub and create a container.

This is how, Docker host can pull the image and create a container out of it.

Now this image is customized by Ansible which means that the artifacts which we have created, so we can see our application is running on Docker host and our end user can access this docker host.

Before creating an Ansible Playbook, let quickly see how Ansible Playbook does work.

To run the ansible playbook, we need to use the command,
ansible-playbook

We need to define where we need to run this ansible playbook for that we can provide with the -l option, that is inventory.

Now where does the inventory exist?

If we didn't give the -i option, it will take the default inventory which is in /etc/ansible/hosts

So, whatever servers are present in the hosts file it will execute by default.

In our case, we need to execute ansible playbook on Ansible server.

But in the hosts file we mentioned only the docker host IP address.

So, we can add our Ansible server IP address to the host file and we can tell to the Ansible server that they should execute only on Ansible Server.

Take the private IP of the Ansible Server.

ifconfig --#To check the IP address.

Add the private IP of the Ansible server and define the group on host's file.

By keeping the IP under group, so whenever I give ansible (group name), while executing command, it will take the IP under the group.

vi /etc/ansible/hosts

```
[dockerhost]
172.31.25.42
[ansible]
172.31.19.187
```

```
[dockerhost]
172.31.25.42
[ansible]
172.31.19.187
```

Now to make work with Ansible, you should copy your SSH keys onto Ansible server itself. Otherwise, you cannot be able to communicate with this system.

To Test the connection.

```
ansible all -a uptime
```

```
[ansibleadmin@AnsibleServer docker]$ ansible all -a uptime
The authenticity of host '172.31.19.187 (172.31.19.187)' can't be established.
ECDSA key fingerprint is SHA256:35MQToDDAOaVJxIqvxjxn616ynPw7mxjWyqfoxe8Bzg.
ECDSA key fingerprint is MD5:e5:1c:c5:58:d3:60:9f:b8:a0:09:42:44:b3:35:7c:ed.
Are you sure you want to continue connecting (yes/no)? [WARNING]: Platform linux on host 172.31.25.42 is using the discovered
future installation of another Python interpreter
could change this. See https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more informat
172.31.25.42 | CHANGED | rc=0 >>
14:07:35 up 1 day,  1:38,  1 user,  load average: 0.00, 0.00, 0.00

172.31.19.187 | UNREACHABLE! => {
    "changed": false,
    "msg": "Failed to connect to the host via ssh: Host key verification failed.",
    "unreachable": true
}
[ansibleadmin@AnsibleServer docker]$
```

The connection failed with ansible but connection successful with the dockerhost.

To resolve this, we should copy the ssh keys (authorized keys) to the ansible system.

```
ssh-copy-id 172.31.19.187
```

Now, check the connection.

```
ansible all -a uptime
```

```
[ansibleadmin@AnsibleServer docker]$ ansible all -a uptime
[WARNING]: Platform linux on host 172.31.25.42 is using the discovered Python
interpreter at /usr/bin/python, but future installation of another Python
interpreter could change this. See https://docs.ansible.com/ansible/2.9/referen
ce_appendices/interpreter_discovery.html for more information.
172.31.25.42 | CHANGED | rc=0 >>
17:02:10 up 1 day,  4:33,  1 user,  load average: 0.00, 0.00, 0.00
[WARNING]: Platform linux on host 172.31.19.187 is using the discovered Python
interpreter at /usr/bin/python, but future installation of another Python
interpreter could change this. See https://docs.ansible.com/ansible/2.9/referen
ce_appendices/interpreter_discovery.html for more information.
172.31.19.187 | CHANGED | rc=0 >>
17:02:10 up  7:54,  3 users,  load average: 0.08, 0.02, 0.01
[ansibleadmin@AnsibleServer docker]$
```

```
ssh-copy-id localhost --#For local host
```

Now write ansible playbook.

```
vi regapp.yml
```

```

---
- hosts: ansible
  tasks:
    - name: Create docker image
      command: docker build -t regapp:latest .
      args:
        chdir: /opt/docker

```

```

---
- hosts: ansible
  tasks:
    - name: Create docker image
      command: docker build -t regapp:latest .
      args:
        chdir: /opt/docker
~  

~  

~
```

ansible-playbook regapp.yml --check --#To dry run the command to check if the file is executing properly.

ansible-playbook regapp.yml
docker images

```

[ansibleadmin@AnsibleServer docker]$ docker images
REPOSITORY      TAG          IMAGE ID      CREATED       SIZE
regapp          latest        b8a783a1e25a   5 hours ago  460MB
regapp          v1           b8a783a1e25a   5 hours ago  460MB
tomcat          latest        405afe63d576   4 days ago   455MB
[ansibleadmin@AnsibleServer docker]$
```

This is how we can create Docker image by using ansible playbook.

Now let's see how we can commit or push this Docker image onto Docker Hub. So that our Docker host can be able to pull the image and create a container out of it.

To commit the image on the docker hub use the below steps:

docker login
Username: amanduggal001
Password: Ammy@56789

Now we cannot directly push the image to the docker hub account like,

docker push regapp:latest

```
[ansibleadmin@AnsibleServer docker]$ docker push regapp:latest
The push refers to repository [docker.io/library/regapp]
8c2cf7ee328b: Preparing
762d4b745489: Preparing
f57cd456a2a2: Preparing
88cal7fdb2cd: Preparing
6f37c5d016c8: Preparing
c1793f638462: Waiting
4df676480fd4: Waiting
d9d7e0852802: Waiting
cbb50874016a: Waiting
5498e8c22f69: Waiting
denied: requested access to the resource is denied
[ansibleadmin@AnsibleServer docker]$
```

Whenever we try to push directly it doesn't recognize that onto which account we can be able to push. That is the reason we should provide our username prefix to our image.

docker tag b8a783a1e25a amanduggal001/regapp:latest --# Added the tag of username to identify the dockerhub account.

docker images

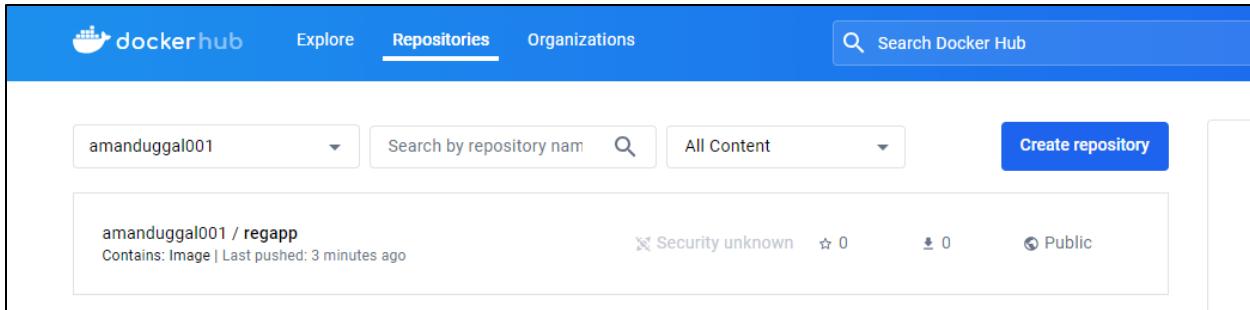
```
[ansibleadmin@AnsibleServer docker]$ docker tag b8a783a1e25a amanduggal001/regapp:latest
[ansibleadmin@AnsibleServer docker]$ docker images
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE
amanduggal001/regapp    latest   b8a783a1e25a  5 hours ago  460MB
regapp              latest   b8a783a1e25a  5 hours ago  460MB
regapp              v1      b8a783a1e25a  5 hours ago  460MB
tomcat              latest   405afe63d576  4 days ago   455MB
[ansibleadmin@AnsibleServer docker]$
```

docker push amanduggal001/regapp:latest --#Push the image

```
[ansibleadmin@AnsibleServer docker]$ docker push amanduggal001/regapp:latest
The push refers to repository [docker.io/amanduggal001/regapp]
8c2cf7ee328b: Pushed
762d4b745489: Pushed
f57cd456a2a2: Pushed
88cal7fdb2cd: Pushed
6f37c5d016c8: Pushed
c1793f638462: Pushed
4df676480fd4: Pushed
d9d7e0852802: Pushed
cbb50874016a: Pushed
5498e8c22f69: Pushed

latest: digest: sha256:bcff598dfa9beef73e2e2417bb5d267d72496af9759544aeda51e5bff63365ad size:
2414
[ansibleadmin@AnsibleServer docker]$
```

You can validate the image in the docker hub.



Now, next we see how we can incorporate these manual steps with our Ansible playbook, so that in Ansible we can build tag and commit image onto the Docker hub and the docker host will pull this image and create a container out of it.

Edit the ansible playbook.

```
vi regapp.yml
```

```
---
- hosts: ansible
  tasks:
    - name: Create docker image
      command: docker build -t regapp:latest .
      args:
        chdir: /opt/docker

    - name: create docker tag.
      command: docker tag regapp:latest amanduggal001/regapp:latest

    - name: Push the docker image to docker hub
      command: docker push amanduggal001/regapp:latest
```

```
---
- hosts: ansible
  tasks:
    - name: Create docker image
      command: docker build -t regapp:latest .
      args:
        chdir: /opt/docker

    - name: create docker tag.
      command: docker tag regapp:latest amanduggal001/regapp:latest

    - name: Push the docker image to docker hub
      command: docker push amanduggal001/regapp:lates
```

```
ansible-playbook regapp.yml
```

```

PLAY [ansible] ****
TASK [Gathering Facts] ****
[WARNING]: Platform linux on host 172.31.19.187 is using the discovered Python interpreter at
/usr/bin/python, but future installation of another Python interpreter could change this. See
https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more
information.
ok: [172.31.19.187]

TASK [Create docker image] ****
changed: [172.31.19.187]

TASK [create docker tag.] ****
changed: [172.31.19.187]

TASK [Push the docker image to docker hub] ****
changed: [172.31.19.187]

PLAY RECAP ****
172.31.19.187 : ok=4    changed=3    unreachable=0    failed=0    skipped=0    re
scued=0    ignored=0

```

Till now, ansible playbook will build, commit and push the image to the docker hub. But we have to execute the ansible playbook manually to run those tasks.

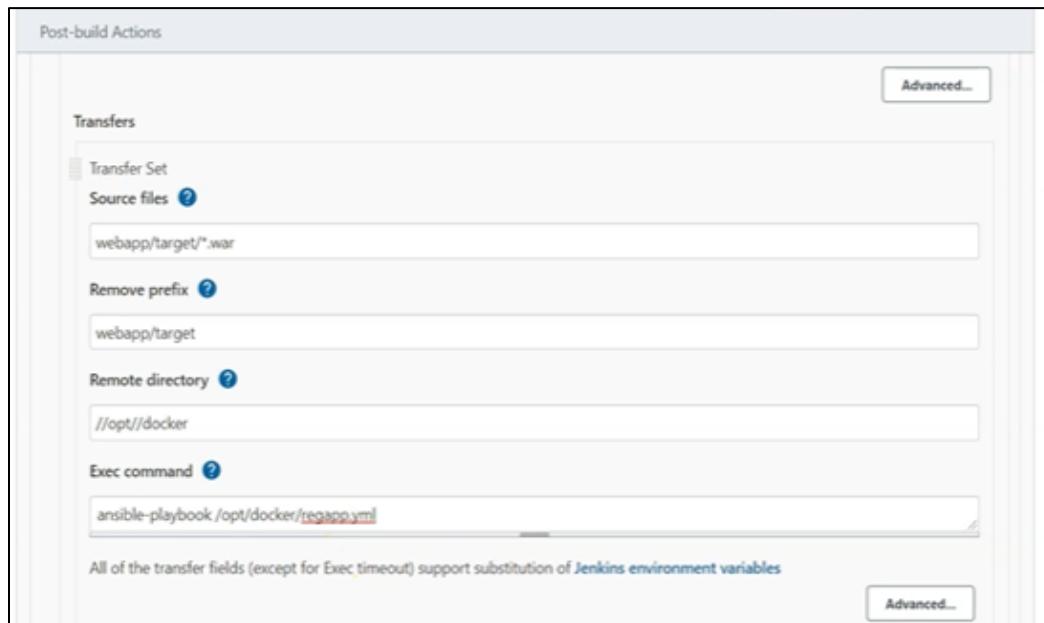
Now we will automate the execution of ansible playbook via Jenkins.

Jenkins server will initial this Ansible Playbook whenever there is new code come in. That is where we need to create a new image.

Go to Jenkins > configure the job '**Copy_Artifact_Onto_Ansible**'

Add the below command under the Exec command field.

ansible-playbook /opt/docker/regapp.yml



Enable the Poll SCM which ensures to execute the job in every minute and build if any new commit found.



Now, whenever any commit or any change is done the job gets executed. Once this job is executed our Ansible playbook should get executed.

Once that is done, we could be able to see the Docker images with the latest time stamps.

[Go to Git Bash](#)

```
cd D:/AWS/DevopsProject/hello-world/webapp/src/main/webapp
```

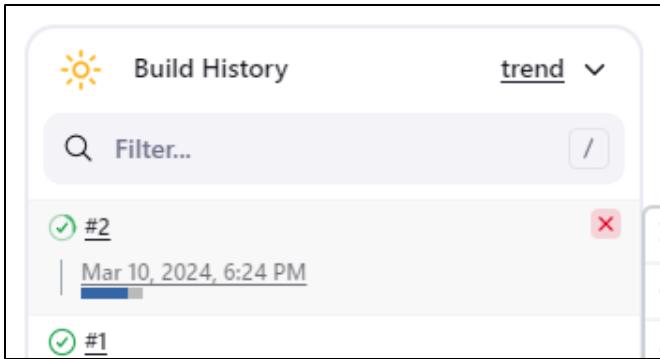
```
vi index.jsp
```

Change the source code.

```
git add .  
git commit -m "bugfix"  
git push origin master
```

```
Aman.Duggal@AmanD-OEG MINGW64 /d/AWS/DevopsProject/hello-world/webapp/src/main/webapp (master)  
$ git push origin master  
Enumerating objects: 13, done.  
Counting objects: 100% (13/13), done.  
Delta compression using up to 8 threads  
Compressing objects: 100% (5/5), done.  
Writing objects: 100% (7/7), 546 bytes | 273.00 KiB/s, done.  
Total 7 (delta 2), reused 0 (delta 0), pack-reused 0  
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.  
To https://github.com/amanduggal001/hello-world.git  
 07955e1..3f9724e master -> master
```

Build automatically starts once we push the code.



This means that once code change happens, Jenkins trigger job, copy artifacts onto Ansible, Ansible creates image.

Now we left out only with one step that is creating a container out of this image.

For that in Ansible Playbook we tell our docker host to go and connect to the Docker hub, pull this image, and create a container out of it.

Write one more Ansible playbook and mention to create a new container on the Docker host. Then the Docker host can understand the instructions and it goes and pull the image from the docker hub and create a container out of it.

```
vi deploy_regapp.yml
```

```
---
- hosts: dockerhost
  tasks:
    - name: create container
      command: docker run -d --name regapp-server -p 8082:8080 amanduggal001/regapp:latest
```

```
---
- hosts: dockerhost
  tasks:
    - name: create container
      command: docker run -d --name regapp-server -p 8082:8080 amanduggal001/regapp:latest
```

Test the playbook.

```
ansible-playbook deploy_regapp.yml --check
```

```
[ansibleadmin@AnsibleServer docker]$ ansible-playbook deploy_repapp.yml --check
PLAY [dockerhost] ****
TASK [Gathering Facts] ****
[WARNING]: Platform linux on host 172.31.25.42 is using the discovered Python interpreter at /usr/bin/python, but future installation of another Python interpreter could change this. See https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more information.
ok: [172.31.25.42]

TASK [create container] ****
skipping: [172.31.25.42]

PLAY RECAP ****
172.31.25.42 : ok=1    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
```

Now, login to the docker host system, we'll delete all our containers. Then we will execute this job manually so that we can see a new container is getting created.

On docker host node -

```
docker rm -f $(docker ps -q) --#To remove all the docker containers
docker rmi -f $(docker images -q) --#To remove all the docker images
```

```
[dockeradmin@DockerServer /]$ docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
[dockeradmin@DockerServer /]$ docker ps
CONTAINER ID      IMAGE      COMMAND      CREATED      STATUS      PORTS      NAMES
[dockeradmin@DockerServer /]$ docker ps -a
CONTAINER ID      IMAGE      COMMAND      CREATED      STATUS      PORTS      NAMES
```

On ansible host node –

ansible-playbook deploy_repapp.yml --# It will pull the image from the docker hub and create a container out of it.

```
[ansibleadmin@AnsibleServer docker]$ ansible-playbook deploy_repapp.yml

PLAY [dockerhost] ****
TASK [Gathering Facts] ****
[WARNING]: Platform linux on host 172.31.25.42 is using the discovered Python interpreter at /usr/bin/python, but future installation of another Python interpreter could change this. See https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more information.
ok: [172.31.25.42]

TASK [create container] ****
fatal: [172.31.25.42]: FAILED! => {"changed": true, "cmd": ["docker", "run", "-d", "--name", "regapp-server", "-p", "8082:8080", "amanduggal001/regapp:latest"], "delta": "0:00:00.063761", "end": "2024-03-10 19:22:30.319490", "msg": "non-zero return code", "rc": 126, "start": "2024-03-10 19:22:30.255729", "stderr": "docker: Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Post \\"http://$var$2Frun$2Fdocker.sock/v1.24/containers/create?name=regapp-server\\": dial unix /var/run/docker.sock: connect: permission denied.\nSee 'docker run --help'.", "stderr_lines": ["docker: Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Post \\"http://$var$2Frun$2Fdocker.sock/v1.24/containers/create?name=regapp-server\\": dial unix /var/run/docker.sock: connect: permission denied.", "See 'docker run --help'."], "stdout": "", "stdout_lines": []}

PLAY RECAP ****
172.31.25.42 : ok=1    changed=0    unreachable=0    failed=1    skipped=0    re
scued=0    ignored=0
```

You will get the exception due to permission issue on /var/run/docker.sock. The issue occurred because we are working as ansibleadmin user.

To grant permission go to **docker host node**.

chmod 777 /var/run/docker.sock

chmod: This is a command in Unix-like operating systems used to change the permissions of a file or directory.

777: This is a permission setting in octal notation. In the context of file permissions:

The first digit (7) corresponds to the owner's permissions.

The second digit (7) corresponds to the group's permissions.

The third digit (7) corresponds to others' (everyone else's) permissions.

In octal notation, each digit represents a combination of read (4), write (2), and execute (1) permissions.

The digits are added to represent the cumulative permissions.

So, 7 means read (4) + write (2) + execute (1), which is the highest level of permission.

ansible-playbook deploy_repapp.yml

```
[ansibleadmin@AnsibleServer docker]$ ansible-playbook deploy_reapp.yml

PLAY [dockerhost] ****
TASK [Gathering Facts] ****
[WARNING]: Platform linux on host 172.31.25.42 is using the discovered Python interpreter at /usr/bin/python, but fu
could change this. See https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more
ok: [172.31.25.42]

TASK [create container] ****
changed: [172.31.25.42]

PLAY RECAP ****
172.31.25.42 : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
[ansibleadmin@AnsibleServer docker]$
```

Once the playbook is executed check on **Docker Host**, image is pull from the docker hub and created a container out of it.

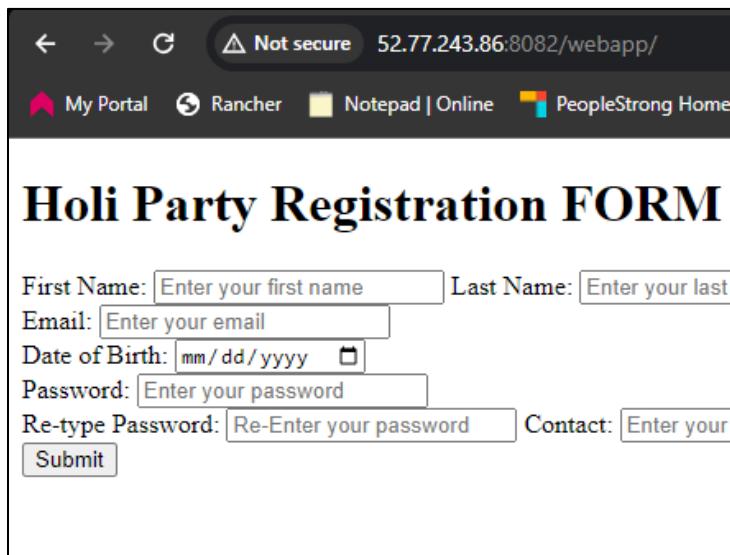
NOTE- We run ansible playbook on ansible server, and the images and containers are created on the docker server.

docker images
docker ps

```
[dockeradmin@DockerServer /]$ docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
amanduggal001/regapp  latest   5398975c833f  16 minutes ago  460MB
[dockeradmin@DockerServer /]$ docker ps
CONTAINER ID      IMAGE      COMMAND      CREATED      STATUS      PORTS      NAMES
762d10e6890a  amanduggal001/regapp:latest  "catalina.sh run"  17 seconds ago  Up 16 seconds  0.0.0.0:8082->8080/tcp, :::8082->8080/tcp  amancontainer
[dockeradmin@DockerServer /]$
```

Now check the application on browser,

Docker Host public ip: port no/webapp
52.77.243.86:8082/webapp/



Now, the problem arises whenever you execute the same playbook (deploy_repapp.yml) the conflict occurs. Since the container name is already used by another container and the port no is already in use.

```
[ansibleadmin@AnsibleServer docker]$ ansible-playbook deploy_repapp.yml

PLAY [dockerhost] ****

TASK [Gathering Facts] ****
[WARNING]: Platform linux on host 172.31.25.42 is using the discovered Python
interpreter at /usr/bin/python, but future installation of another Python
interpreter could change this. See https://docs.ansible.com/ansible/2.9/referen
ce_appendices/interpreter_discovery.html for more information.
ok: [172.31.25.42]

TASK [create container] ****
fatal: [172.31.25.42]: FAILED! => {"changed": true, "cmd": ["docker", "run", "-d",
"--name", "amancontainer", "-p", "8082:8080", "amanduggal001/regapp:latest"],
"delta": "0:00:00.064055", "end": "2024-03-11 16:56:36.596890", "msg": "non-zer
o return code", "rc": 125, "start": "2024-03-11 16:56:36.532835", "stderr": "doc
ker: Error response from daemon: Conflict. The container name \"/amancontainer\"
is already in use by container \"762d10e6898abcec0e21073fa9fe3cdd53f80fe9f60a38
442c55c194e05dfc9c\". You have to remove (or rename) that container to be able t
o reuse that name.\nSee 'docker run --help'.", "stderr_lines": ["docker: Error r
esponse from daemon: Conflict. The container name \"/amancontainer\" is already
in use by container \"762d10e6898abcec0e21073fa9fe3cdd53f80fe9f60a38442c55c194e0
5dfc9c\". You have to remove (or rename) that container to be able to reuse that
name.", "See 'docker run --help.'"], "stdout": "", "stdout_lines": []}

PLAY RECAP ****
172.31.25.42 : ok=1    changed=0    unreachable=0    failed=1    s
kipped=0    rescued=0    ignored=0
```

To overcome this problem, we need to execute the same steps that we have taken on our Jenkins sometime back.

That is first remove the existing container then remove the existing image. The question comes as to why we need to remove the image because if there is an existing image present it won't go to the Docker Hub to pull the latest image.

At last, create a new container.

Deploy Ansible Playbook

- Remove existing container.
- Remove existing image.
- Create a new container.



Once this is done, we can add our deployment process to our Jenkins so that Jenkins can take care of the entire Continuous Integration and Continuous deployment of our containers.

Edit the existing ansible playbook.

```
vi deploy_repapp.yml
---
- hosts: dockerhost
  tasks:
    - name: stop container
      command: docker stop amancontainer
      ignore_errors: yes      --#Incase if we have these containers stop it and in case these containers
                               doesn't exists don't fail the playbook.

    - name: remove the conatiner
      command: docker rm amancontainer

    - name: remove the image
      command: docker rmi amanduggal001/regapp:latest

    - name: create container
      command: docker run -d --name amancontainer -p 8082:8080 amanduggal001/regapp:latest
```

A screenshot of a terminal window titled "ansibleadmin@AnsibleServer:/opt/docker". The window displays the YAML code for the Ansible playbook. The code defines a single host group "dockerhost" with four tasks. The first task stops a container named "amancontainer" using the "docker stop" command. The second task removes the container using "docker rm". The third task removes the image "amancontainer" using "docker rmi". The fourth task creates a new container named "amancontainer" with port mapping "8082:8080" and links it to the image "amanduggal001/regapp:latest" using "docker run".

```
ansibleadmin@AnsibleServer:/opt/docker
---
- hosts: dockerhost
  tasks:
    - name: stop container
      command: docker stop amancontainer
      ignore_errors: yes      --#Incase if we have these containers stop it and in case these containers
                               doesn't exists don't fail the playbook.

    - name: remove the conatiner
      command: docker rm amancontainer

    - name: remove the image
      command: docker rmi amanduggal001/regapp:latest

    - name: create container
      command: docker run -d --name amancontainer -p 8082:8080 amanduggal001/regapp:latest
```

```
ansible-playbook deploy_repapp.yml
```

Check on docker server the container will be created.

Now, it's time to automate the process.

Go to Jenkins> Configure the job '**Copy_Artifacts_Onto_Ansible**'

Under Post-build Actions, enter the ansible command to execute the ansible playbook.

```
ansible-playbook /opt/docker/regapp.yml;  
sleep 10 --#To wait for 10 sec then the next command will execute.  
ansible-playbook /opt/docker/deploy_repapp.yml
```

The screenshot shows the Jenkins job configuration page for 'Copy_Artifacts_Onto_Ansible'. On the left, there is a sidebar with 'Configure' and several build steps: General, Source Code Management, Build Triggers, Build Environment, Pre Steps, Build, Post Steps, Build Settings, and Post-build Actions. The 'Post-build Actions' step is currently selected. On the right, under 'Transfer Set', there are fields for 'Source files' (webapp/target/*.war), 'Remove prefix' (webapp/target), 'Remote directory' (/opt/docker), and 'Exec command' (which contains the ansible command). A note at the bottom states: 'All of the transfer fields (except for Exec timeout) support substitution of Jenkins environment variables'.

Apply > Save.

Now make some changes in the source code, so that it's going to trigger the Jenkins job.

[Go to Git Bash](#)

```
cd D:/AWS/DevopsProject/hello-world/webapp/src/main/webapp
```

```
vi index.jsp
```

Change the source code.

```
git add .  
git commit -m "bugfix"  
git push origin master
```

Once pushed the image, the job gets initializing.

The screenshot shows the CircleCI build history interface. It displays four builds: #6 (Mar 11, 2024, 6:25 PM), #5 (Mar 11, 2024, 4:33 PM), #4 (Mar 11, 2024, 4:30 PM), and #3. Each build entry includes a checkbox, a red 'X' button, and up/down arrow buttons for reordering.

So, first thing what is going to happen is, it is going to update the image in our Ansible server and same thing updated over docker hub.

After that it should be able to delete the existing container on docker host and create a new container.

The screenshot shows the Docker Hub interface for the repository amanduggal001/regapp. It displays a grey cube icon, the repository name amanduggal001/regapp with a star icon, and the text "By amanduggal001 · Updated 3 minutes ago". Below this is a "Image" button.

New images and containers are created.

```
[root@DockerServer docker]# docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
amanduggal001/regapp    latest    726b3ead2dc6  About a minute ago   460MB
[root@DockerServer docker]# docker ps
CONTAINER ID      IMAGE                  COMMAND      CREATED      STATUS      PORTS      NAMES
19d482b2f077    amanduggal001/regapp:latest    "catalina.sh run"  52 seconds ago   Up 51 seconds  0.0.0.0:8082->8080/tcp, :::8082->8080/tcp   amancontainer
[root@DockerServer docker]#
```

Updated source code reflected on application.

Not secure 52.77.243.86:8082/webapp/

My Portal Rancher Notepad | Online PeopleStrong Home Document 360 Outlook Udemy - Learn

Final Registration Form (Last date 12 march 2024)

First Name: Enter your first name Last Name: Enter your last name

Email: Enter your email

Date of Birth: mm/dd/yyyy

Password: Enter your password

Re-type Password: Re-Enter your password Contact: Enter your Mobile Number Gender: Male

Submit

Till now we have modify our Jenkins job in such a way if someone changes the source code it should automatically build the code, create an image, create a container and we could able to access those changes from the browser.

Deploying on as a Container



But if you see the problem whenever there are some changes, we are terminating the existing container and creating a new container, during this time end user cannot be able to access the application.

Another thing is if our container is terminated, how we can come to know that it is not working? or how we can create a new container automatically?

Till now, we don't have such kind of mechanisms to handle these, that is where container management system comes.

We are using the container management service i.e. Kubernetes. Due to its multiple advantages, we are not going to deploy our application as the docker container, but we are deploy it as a part on Kubernetes environment.

Deploying on as a POD



Kubernetes Installation

We are using the AWS EKSCTL (command line utility) for managing the containers.

Pre- Requisites:

- Create an EC2 Instance.
- Install AWS CLI latest version.

This EC2 instance we are using it as a bootstrap image. We are going to setup Kubernetes with the help of this bootstrap image and this Ec2 instance is going to communicate with the EKS service through the AWS CLI.

Check the AWS Version

On Kubernetes Terminal

```
aws --version
```

```
[ec2-user@KubernetesServer:~]$ aws --version
aws-cli/1.18.147 Python/2.7.18 Linux/5.10.210-201.852.amzn2.x86_64 botocore/1.18.6
```

As a prerequisite, it is required to install the AWS CLI version 2.2.

Fargate – Linux	Managed nodes – Linux
<p>To create your cluster with Amazon EC2 Linux managed nodes</p> <p>Create your cluster and Linux managed node group. Replace <code>my-cluster</code> with your own value.</p> <p>Replace <code>your-key</code> with the name of an existing key pair. If you don't have a key pair, you can create one with the following command. If necessary, change <code>us-west-2</code> to the Region that you create your cluster in. Be sure to save the return output in a file on your local computer. For more information, see Creating or importing a key pair in the Amazon EC2 User Guide for Linux Instances. Though the key isn't required in this guide, you can only specify a key to use when you create the node group. Specifying the key allows you to SSH to nodes once they're created. To run the command, you need to have the AWS CLI version 2.37 or later or 1.20.40 or later. For more information, see Installing, updating, and uninstalling the AWS CLI in the AWS Command Line Interface</p>	

- **Download the AWS CLI Version 2** --<https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html>

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install
```

`sudo su -` --#This command is used to switch to the root user in a Linux system. The - option ensures that the environment variables are set as if the root user had logged in directly.

```
aws --version
```

```
[root@KubernetesServer ec2-user]# sudo su -
Last login: Mon Mar 11 20:33:58 UTC 2024 on pts/0
[root@KubernetesServer ~]# aws --version
aws-cli/2.15.27 Python/3.11.8 Linux/5.10.210-201.852.amzn2.x86_64 exe/x86_64.amzn.2 prompt/off
[root@KubernetesServer ~]#
```

- Now setup kubectl, check the AWS documentation for the latest versions and install it.

<https://docs.aws.amazon.com/eks/latest/userguide/install-kubectl.html>

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.0/2024-01-04/bin/linux/amd64/kubectl
```

```
ls
```

```
[root@KubernetesServer ~]# ls
kubectl
```

Now, we are downloading the kubectl packages. Once it is downloaded, we need to add execution permissions.

```
chmod +x kubectl
```

Move the kubectl file to /usr/local/bin.

mv kubectl /usr/local/bin --#This is the default path of Linux OS and by default whenever we execute kubectl command it will go and pick up from this location.

echo \$PATH --#We could see usr/local/bin in our path. So whenever we execute any command it is going to validate in this location.

```
[root@KubernetesServer ~]# echo $PATH  
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/root/bin
```

kubectl version --# To check if the kubectl is successfully installed or not.

```
[root@KubernetesServer ~]# kubectl version  
Client Version: v1.29.0-eks-5e0fdde  
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3  
The connection to the server localhost:8080 was refused - did you specify the right host or port?  
[root@KubernetesServer ~]#
```

Similarly, install EKSCTL. --<https://docs.aws.amazon.com/emr/latest/EMR-on-EKS-DevelopmentGuide/setting-up-eksctl.html>

curl --silent --location
[https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_\\$\(uname -s\)_amd64.tar.gz](https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz)" | tar xz -C /tmp --#The package downloaded in the /tmp path.

cd /tmp

```
[root@KubernetesServer ~]# cd /tmp  
[root@KubernetesServer tmp]# ls  
eksctl  systemd-private-41456dd44cb4465c96cac7e383ef003f-chronyd.service-cNVlJ3  
[root@KubernetesServer tmp]#
```

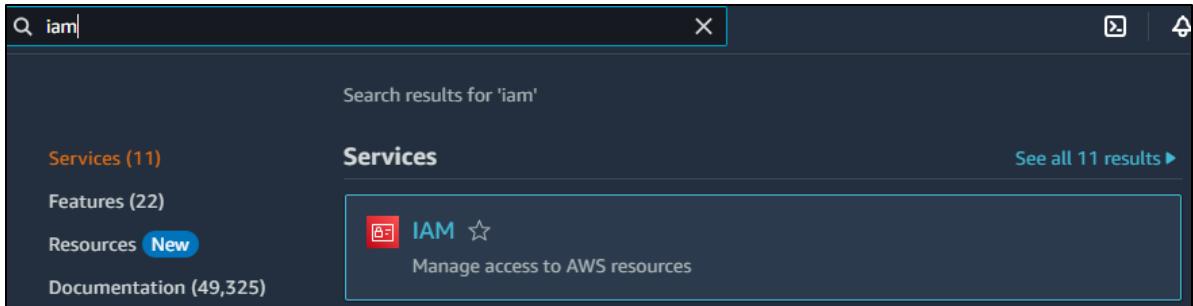
mv eksctl /usr/local/bin --#Move to default path.

eksctl version

```
[root@KubernetesServer tmp]# eksctl version  
0.173.0  
[root@KubernetesServer tmp]#
```

Now create an IAM role.

Go to AWS console and search for IAM service.



Select Roles > Create role.

A screenshot of the AWS IAM Roles page. The left sidebar shows 'Identity and Access Management (IAM)' with a 'Roles' tab selected. The main content area displays a table of existing IAM roles. The columns are 'Role name', 'Trusted entities', and 'Last activity'. The roles listed are: 'AWSServiceRoleForAutoScaling' (trusted by 'aws:autoscaling', last active 1 hour ago), 'AWSServiceRoleForAWSLicenseManagerRole' (trusted by 'aws:license-manager', last active -), 'AWSServiceRoleForElasticLoadBalancing' (trusted by 'aws:elasticloadbalancing', last active -), and 'AWSServiceRoleForMarketplaceLicenseManagement' (trusted by 'aws:marketplace-license-management', last active -). There are buttons for 'Create role' and 'Delete' at the top right of the table.

Select EC2 under Service or use case > Next.

A screenshot of a configuration dialog. The title is 'Use case'. Below it is a sub-section titled 'Service or use case' with a dropdown menu. The dropdown menu has 'EC2' selected, indicated by a yellow box. Other options in the dropdown are partially visible.

Add below Permission > Next.

- AmazonEC2FullAccess
- AWSCloudFormationFullAccess
- IAMFullAccess

Add permissions Info

Permissions policies (911) Info

Choose one or more policies to attach to your new role.

Filter by Type

Policy name	Type	Description
<input type="checkbox"/> AmazonEC2FullAccess	AWS managed	Provides full access to Amazon EC2 via...

▶ Set permissions boundary - optional

Enter role name > Click on 'Create a role'.

IAM > Roles > Create role

Step 1
Select trusted entity

Step 2
Add permissions

Step 3
Name, review, and create

Name, review, and create

Role details

Role name
Enter a meaningful name to identify this role.

Maximum 64 characters. Use alphanumeric and '+,-,@,_' characters.

Add this role to our Kubernetes EC2 instance.

Select EC2 instance > Actions > Security > Modify IAM.

Instances (1/5) Info

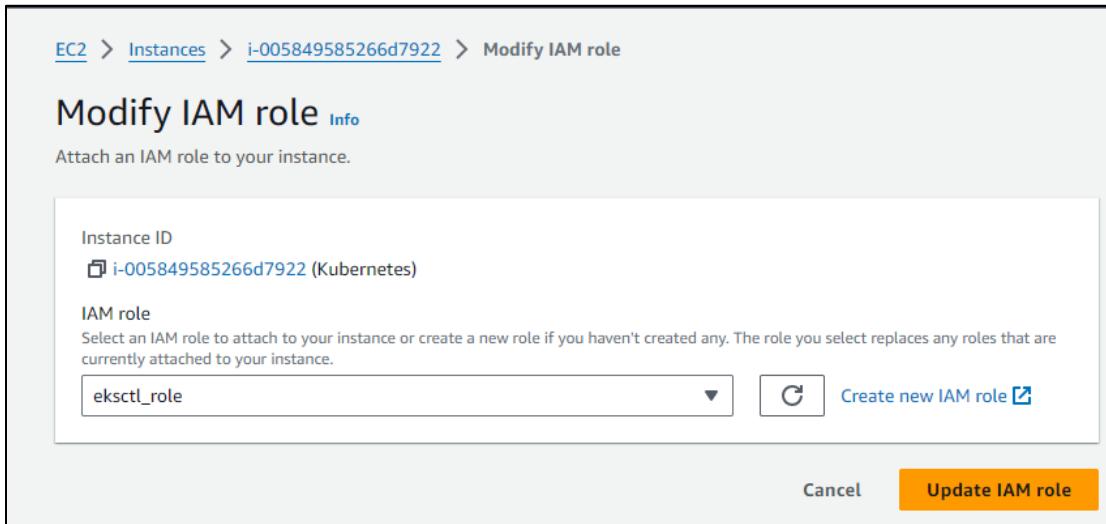
Find Instance by attribute or tag (case-sensitive)

Instance state = running Clear filters

Any state

Name	Instance ID	Instance state	Instance type	Status	Actions
Docker	i-06d33a96ff7c78f8b	Running	t2.micro	Change security groups Get Windows password	Connect View details Manage instance state Instance settings Networking Security Image and templates Monitor and troubleshoot
Ansible	i-0daa89af58feb9b9c	Running	t2.micro	Get Windows password	ability Zone outtheast-1a
Jenkins Server	i-0f49bb1e79ab458d9	Running	t2.micro	Modify IAM role	outeast-1a
Kubernetes	i-005849585266d7922	Running	t2.micro	2/2 checks passed View alarms +	ap-southeast-1a

Select your IAM role.



Now this Kubernetes server (Bootstrap server) have access to almost all the AWS services because we have given administrative privileges.

- **Now create a cluster.**

Format:

```
eksctl create cluster --name cluster-name \ --#Your name of the cluster.  
--region region-name \ --#On which region we wish to create the cluster.  
--node-type instance-type \ --#What type on instance we want to create.  
--nodes-min 2 \ --#Minimum nodes.  
--nodes-max 2 \--#Maximum nodes. If we don't specify by default it will create 2 instances.  
--zones <AZ-1>,<AZ-2> --#On which zones it can spread in this region.
```

Now update this command according to your requirement.

On Terminal (K8s)

```
eksctl create cluster --name amancluster --region ap-southeast-1 --node-type t2.small
```

--#NOTE: This command will take 20-25 min.

```
eksctl delete cluster --region=ap-southeast-1 --name=amancluster --#To delete the existing cluster. Not  
to run every time.
```

**NOTE: DELETE THE CLUSTER ONCE THE PROJECT IS COMPLETED, OTHERWISE UNNECESSARY BILL IS
GENERATED.**

eksctl is going to create our environment with help of cloud formation. Once you execute the command you will see the cloud formation template which created and executing to set up our EC2 instances.

Once the command is executed properly, 2 new instances are created.

Instances (7) Info						
		Connect	Instance state ▾	Actions ▾	Launch instances	▼
<input type="text"/> Find Instance by attribute or tag (case-sensitive)						
Instance state = running X		Clear filters				
Any state						
<input type="checkbox"/>	Name Edit	Instance ID	Instance state	Instance type	Status check	Alarm status
<input type="checkbox"/>	nikkicluster-ng-26d350b2-Node	i-00b5f7f5e397f18f4	Running View details Logs	t2.small	2/2 checks passed View alarms +	
<input type="checkbox"/>	Tomcat Server	i-0f32a823dcabc7b94	Running View details Logs	t2.micro	2/2 checks passed View alarms +	
<input type="checkbox"/>	Docker	i-06d33a96ff7c78f8b	Running View details Logs	t2.micro	2/2 checks passed View alarms +	
<input type="checkbox"/>	Ansible	i-0daa89af58feb9b9c	Running View details Logs	t2.micro	2/2 checks passed View alarms +	
<input type="checkbox"/>	Jenkins Server	i-0f49bb1e79ab458d9	Running View details Logs	t2.micro	2/2 checks passed View alarms +	
<input type="checkbox"/>	Bootstrap Server	i-0eabaa81c2a258fb8	Running View details Logs	t2.micro	2/2 checks passed View alarms +	
<input type="checkbox"/>	nikkicluster-ng-26d350b2-Node	i-0125899fcc7804880	Running View details Logs	t2.small	2/2 checks passed View alarms +	

Also check the cloud formation.

CloudFormation						
CloudFormation > Stacks						
Stacks		Stack name	Status	Created time	Description	Stack actions ▾
StackSets						
Exports						
Designer						
IaC generator						
▼ Registry						
Public extensions						
Activated extensions						
Publisher						
Stacks (2)						
		<input type="text"/> Filter by stack name	Active	View nested		◀ 1 ▶ ⌂
		Stack name	Status	Created time	Description	Stack actions ▾
<input type="radio"/>		eksctl-nikkicluster-nodegroup-ng-26d350b2	CREATE_COMPLETE	2024-03-12 21:43:58 UTC+0530	EKS Managed Node access: false) [creat	Create stack ▾
<input type="radio"/>		eksctl-nikkicluster-cluster	CREATE_COMPLETE	2024-03-12 21:32:57 UTC+0530	EKS cluster (dedicated IAM: true and managed by ek	

```
cat /root/.kube/config --#This config file is created, so whoever this file they can communicate with our clusters and do the activities.
```

```
[root@KubernetesServer ~]# cat /root/.kube/config
apiVersion: v1
clusters:
- cluster:
  certificate-authority-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURCVEND
  QWUyZ0F3SUJBZ01J2UtyZ1NCY3V1QTR3RFFZSktvWklodmNOQVFITEJRQXdGVEVUTUJFR0ExVUUKQXhN
  S2EzVm1aWEplWlhSbGN6QWVGdzB5TkRBek1URX1NVE00TXpsYUZ3MHpOREF6TURreU1UUxPNeMxhTUJV
  eApFekFSQmdOVkJBTVRDbXQxWW1WeWJtVjBaWE13Z2dFa1BMEdDU3FHU01iM0RRRUJBUVVBQTRJQkR3
  QXdz0VLCkFvSUJBURxT2hPTTBzODcyYkJPMFd1VjMvdStjT3VKTjNvREFuK1UvQXZJRHNKODR6eGE0
  M1g5ci9iMk1YN1YKR1NFMHVUSTdEUW1MSnFkZWZEAeTrU0pLSmNWVHRAUnpndDdBdm5xcGJPY1BhQk11
  OVf1bmh0dFhvK2tmNGRKYQovckpFeDNRNHBYcHR1YU5sNUZEMEzpS2pmbWh5QnViZ09RRytaUpKVfp1
  TnRPaUwxZU5XOsremtsS2pIdlJxCmpQbTFGSDB0UzJwRm9GOT10UWZpZGtqWTRJT3EzQXB1Y0drSE9U
  eXFpQ1Mzd1VkuUd4TTV6VFhSbTQ5VGN6ZHoKT0EzOEsydFRXTkFSW1lqRUM0TXFWWjd4Nk1oR0pHSXJS
  MXRncTYzUFkzY3RBSHIwcU9JaG9Nd1AwSSsrc0xhUgpuR3NhTThRWFZnRjcyelFvTjZEbnBFSGJHWWFO
  QWdNQkFBR2pXVEJYTUE0R0ExVWREd0VCL3dRRUF3SUNwREFQCKjnT1Z1Uk1CQWY4RUJUQURBUUgvTUIw
  R0ExVWREZ1FXQkJSUGt6a2ZyTzJDNE9zYmFKYUhqME9SVz11Q1V6QVYKQmdOVkhSRUVEakFNZ2dwcmRX
  SmxjbTVsZEdWek1BMEdDU3FHU01iM0RRRUJDd1VBQTRJQkFRRFRSa0tWND10OAo1YW1vV31pNGZTdzVz
  Z3hKdEVGVWBPaGYrMm0zM1Z0aFgl2ktmZUuzRWJcnUrNVJnTGJFUVJZS3pz2k9BbWFFCnNIOHY3RmpW
  eU1FQnNYaEtLK2FLSUxkdm45NUNkVT2vQUMrbmNzcGIreGtYVm51S0ZyeWhqbTdMeGxPeDdFUUYKQ1V
  MU95eXVkT1Erc1BQRFF2Ry9FUlhYdjhz2mFyc1NVMmpwUTd0VG15V1JtWUxiN11mWHZteE1IM1NQZitx
  eApHZUVQYVNiRGhYMit0WD2TaV1Ec1RYjBhNE0rSFhoNV1TS016aWtpSi9FWDFTU3clWGdYMW5LUkc3
  WVg1Q3JnCktSMFR1bk9CM0NHNFNKZU14Z3JINkRPSXh4a0kzNy9WRWk1U1RVN3k2U3dTb0VJYjNQUHMx
  cXdkTFJI2ms4VFoKak2rdGI1bml2WEpLCi0tLS0tRU5EIENFU1RJRk1DQVRFLS0tLS0K
    server: https://404CECFC1851612850C2E309382B3164.gr7.ap-southeast-1.eks.amazonaws.com
      name: newcluster.ap-southeast-1.eksctl.io
contexts:
- context:
  cluster: newcluster.ap-southeast-1.eksctl.io
  user: i-005849585266d7922@newcluster.ap-southeast-1.eksctl.io
  name: i-005849585266d7922@newcluster.ap-southeast-1.eksctl.io
current-context: i-005849585266d7922@newcluster.ap-southeast-1.eksctl.io
```

```
2024-03-12 16:17:04 [+] nodegroup "ng-26d350b2" has 2 node(s)
2024-03-12 16:17:04 [+] node "ip-192-168-16-10.ap-southeast-1.compute.internal" is ready
2024-03-12 16:17:04 [+] node "ip-192-168-58-155.ap-southeast-1.compute.internal" is ready
2024-03-12 16:17:05 [+] kubectl command should work with "/root/.kube/config", try 'kubectl get nodes'
2024-03-12 16:17:05 [+] EKS cluster "nikkicloud" in "ap-southeast-1" region is ready
[root@ip-172-31-27-151 tmp]# kubectl version
Client Version: v1.29.0-eks-5e0fdde
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
Server Version: v1.29.1-eks-508b6b3
[root@ip-172-31-27-151 tmp]# kubectl get nodes
NAME                      STATUS   ROLES      AGE     VERSION
ip-192-168-16-10.ap-southeast-1.compute.internal   Ready    <none>   2m57s   v1.29.0-eks-5e0fdde
ip-192-168-58-155.ap-southeast-1.compute.internal   Ready    <none>   2m50s   v1.29.0-eks-5e0fdde
[root@ip-172-31-27-151 tmp]#
```

Commands:

kubectl get nodes --#To see the pods.

kubectl get all --#To see what all services are running.

kubectl run webapp --image=httpd --#To create the pod.

kubectl delete pod webapp --#To delete pod.

```
[root@ip-172-31-27-151 tmp]# kubectl get all
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)    AGE
service/kubernetes  ClusterIP  10.100.0.1  <none>        443/TCP   52m
[root@ip-172-31-27-151 tmp]# kubectl run webapp --image=httpd
pod/webapp created
[root@ip-172-31-27-151 tmp]# kubectl get all
NAME        READY   STATUS    RESTARTS   AGE
pod/webapp  1/1     Running   0          87s
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)    AGE
service/kubernetes  ClusterIP  10.100.0.1  <none>        443/TCP   83m
[root@ip-172-31-27-151 tmp]#
```

Now, to understand how Kubernetes is going to help our applications and workloads more efficiently we are going to deploy the Nginx application.

Our intention is to run our register application on the k8s cluster so that in case if something happens to our pod or container it should be able to reprovision (recreated).

That is how we make sure that our application is highly available for the end user.

```
kubectl create deployment demo-nginx --image=nginx --port=80 --replicas=2
```

By using this command, we are going to pull the nginx image from the docker hub and create a deployment called demo-nginx. And, in the backend it is going to create a replica set of 2 pods. It exposes the nginx to port number 80.

```
kubectl get deployment
```

```
kubectl get replicaset
```

```
kubectl get pod
```

```
kubectl get all
```

```
[root@ip-172-31-27-151 tmp]# kubectl get deployment
NAME         READY   UP-TO-DATE   AVAILABLE   AGE
demo-nginx   2/2     2           2           22s
[root@ip-172-31-27-151 tmp]# kubectl get replicaset
NAME          DESIRED   CURRENT   READY   AGE
demo-nginx-5c97459668   2         2         2       40s
[root@ip-172-31-27-151 tmp]# kubectl get pod
NAME             READY   STATUS    RESTARTS   AGE
demo-nginx-5c97459668-lnbcq   1/1     Running   0          54s
demo-nginx-5c97459668-xtg7s   1/1     Running   0          54s
[root@ip-172-31-27-151 tmp]# kubectl get all
NAME             READY   STATUS    RESTARTS   AGE
pod/demo-nginx-5c97459668-lnbcq   1/1     Running   0          68s
pod/demo-nginx-5c97459668-xtg7s   1/1     Running   0          68s

NAME          TYPE        CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
service/kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP    98m

NAME         READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/demo-nginx   2/2     2           2           68s
NAME          DESIRED   CURRENT   READY   AGE
replicaset.apps/demo-nginx-5c97459668   2         2         2       68s
[root@ip-172-31-27-151 tmp]#
```

Expose the deployment as service. This will create an ELB in front of those 2 containers and allow us to publicly access them.

Now, we are going to expose this application to the external network for that we are going to use -

```
kubectl expose deployment demo-nginx --port=80 --type=LoadBalancer
```

The kubectl expose command is used in Kubernetes to create a new service and expose it to the external world or other parts of the cluster.

- **kubectl expose:** This is the main command to expose a resource in Kubernetes.
- **deployment demo-nginx:** Specifies the resource type (deployment) and the name of the deployment you want to expose (demo-nginx in this case).
- **--port=80:** Specifies the port on which the service will be exposed. In this case, the service will be exposed on port 80.
- **--type=LoadBalancer:** Specifies the type of service to create. In this example, it's set to LoadBalancer. This type is typically used in cloud environments where a cloud provider's load balancer will be provisioned to distribute external traffic among the pods in the service. The external IP address assigned to the load balancer will be used to access the service.

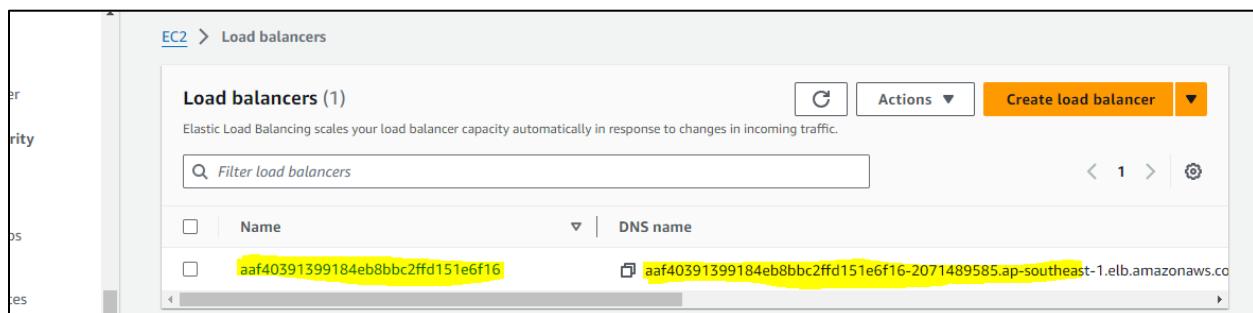
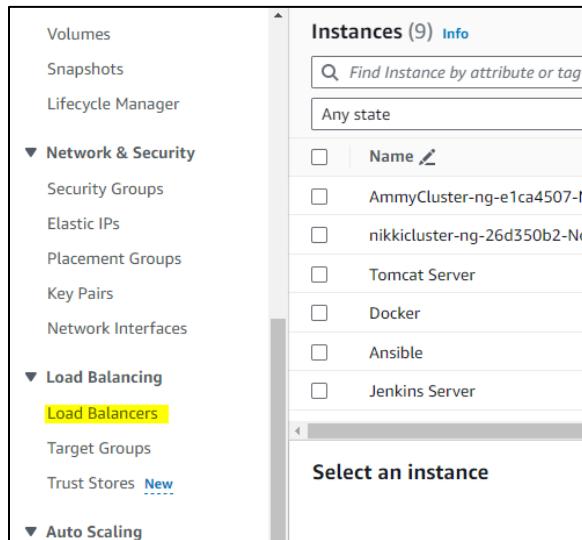
After running this command, Kubernetes will create a new service that forwards traffic to the pods managed by the demo-nginx deployment on port 80. If you are using a cloud provider that supports LoadBalancer services, Kubernetes will automatically provision a load balancer for you.

kubectl get services #To check the service.

```
[root@ip-172-31-27-151 tmp]# kubectl expose deployment demo-nginx --port=80 --type=LoadBalancer
service/demo-nginx exposed
[root@ip-172-31-27-151 tmp]# kubectl get services
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP          PORT(S)        AGE
demo-nginx     LoadBalancer 10.100.235.39  aaf40391399184eb8bbc2ffd151e6f16-2071489585.ap-southeast-1.elb.amazonaws.com  80:32038/TCP  8s
kubernetes     ClusterIP  10.100.0.1    <none>              443/TCP       106m
[root@ip-172-31-27-151 tmp]#
```

Once, we have created, we can access our application with the External IP (See the SS). But this will take a while to open, because we have used the Load Balancer. So it is going to create an actual Load Balancer in the AWS Cloud.

To check, Go to the AWS console > Load Balancers



You will see the load balancer is created. (Compare the name with the LB external ip from the terminal).

Also, the 2 instances are tagged with the load balancers.

The screenshot shows the 'Target instances' tab of the AWS CloudWatch Metrics interface. It displays two instances registered to a load balancer. The table includes columns for Instance ID, Name, Health status, and Health status description. Both instances are labeled 'In-service' and have 'Not applicable' in the health status description column. A search bar at the top allows filtering of target instances.

Instance ID	Name	Health status	Health status description
i-00b5f7f5e397f18f4	nikkicluster-nginx-26d350b2-Node	In-service	Not applicable
i-0125899fcc7804880	nikkicluster-nginx-26d350b2-Node	In-service	Not applicable

Now, check if our application is opening from browser. Copy the DNS address and paste it on the browser and you will see the nginx application page.

The screenshot shows a web browser window with the URL 'aaf40391399184eb8bbc2ffd151e6f16-2071489585.ap-southeast-1.elb.amazonaws.com'. The page content is 'Welcome to nginx!'. It includes a message stating 'If you see this page, the nginx web server is successfully installed and working. Further configuration is required.' Below that, it says 'For online documentation and support please refer to nginx.org. Commercial support is available at nginx.com'. At the bottom, it says 'Thank you for using nginx.'

Create 1st manifest file

Till now, we are manually giving single commands to create the deployments, services. But this is not the correct way.

Now we can see how we can create a Pod and service by using the manifest files. For that delete the existing configurations.

```
kubectl delete deployment demo-nginx -- #To delete the deployment.
```

```
kubectl delete service/demo-nginx --#To delete the services like load balancer.
```

```
vi pod.yml --#To create a manifest file.
```

```
apiVersion: v1
kind: Pod
metadata:
  name: demo-pod
spec:
  containers:
    - name: demo-nginx
      image: nginx
      ports:
        - name: demo-nginx
          containerPort: 80
```

```
apiVersion: v1
kind: Pod
metadata:
  name: demo-pod
spec:
  containers:
    - name: demo-nginx
      image: nginx
      ports:
        - name: demo-nginx
          containerPort: 80
~
```

Now, to expose our pod we need to create a service.

```
vi service.yml
```

```
apiVersion: v1
kind: Service
metadata:
  name: demo-service
spec:
  ports:
    - name: nginx-port
      port: 80
      targetPort: 80
  type: LoadBalancer
```

```

apiVersion: v1
kind: Service
metadata:
  name: demo-service
spec:
  ports:
    - name: nginx-port
      port: 80
      targetPort: 80

  type: LoadBalancer
~
```

kubectl apply -f pod.yml --#To execute the manifest file.
 kubectl apply -f service.yml
 kubectl get pods
 kubectl get services

```

[root@ip-172-31-27-151 tmp]# kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
demo-pod  1/1     Running   0          69s
```

```

[root@ip-172-31-27-151 tmp]# kubectl get services
NAME        TYPE           CLUSTER-IP   EXTERNAL-IP
demo-service  LoadBalancer  10.100.34.35  a2ad4aaaa96924436bdd06e8670b894b-1978937108.ap-southeast-1.elb.amazonaws.com
kubernetes   ClusterIP    10.100.0.1    <none>
[root@ip-172-31-27-151 tmp]#
```

Now, you will see the LB is created but the instances are out of service.

Instance ID	Name	Health status	Description	Security group
i-00b5f7f5e397f18f4	nikkicluster-ng-26d350b2-Node	Out-of-service	Instance has failed at l...	eks-cl
i-0125899fcc7804880	nikkicluster-ng-26d350b2-Node	Out-of-service	Instance has failed at l...	eks-cl

This is because we have created a pod and service and while creating a service, we got a Load Balancer and that LB we are accessing from the external network.

So, whenever we send the request from the external network to the service, how does it know that it has to send it to pod only. Because in the cluster there could be hundreds of pods. That is why we are using the labels in our manifest file.

In the service level, if we use the same label as of pods, then it means that any request comes to this service it looks for the Selector and this Selector have the label and whatever pods are having this label, it is going to forward req to that specific pod.

This is how it's work.

Setup Pod and Service



Now, add the label in pod.yml and service.yml.

vi pod.yml

```
apiVersion: v1
kind: Pod
metadata:
  name: demo-pod
  labels:
    app: demo-app
spec:
  containers:
    - name: demo-nginx
      image: nginx
      ports:
        - name: demo-nginx
          containerPort: 80
~
```

```
vi service.yml
```

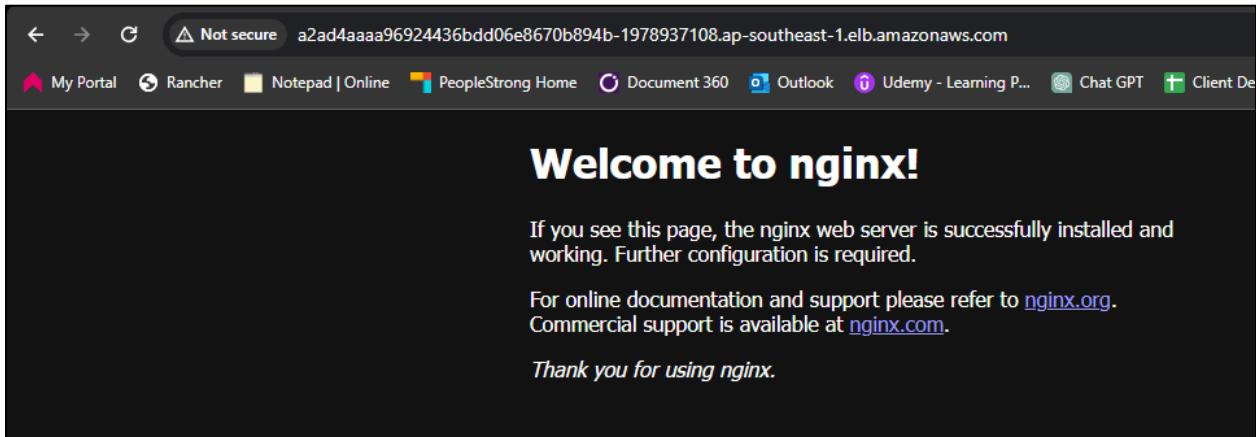
```
apiVersion: v1
kind: Service
metadata:
  name: demo-service
spec:
  ports:
    - name: nginx-port
      port: 80
      targetPort: 80
  selector:
    app: demo-app
  type: LoadBalancer
~
```

kubectl apply -f pod.yml --#It will update the existing pod configuration. no need to delete the pod before apply.

kubectl apply -f service.yml

kubectl describe service/demo-service --# It will shows the details and tell where it has to forward the request.

Now, check if our application is opening from browser. Copy the DNS address and paste it on the browser and you will see the nginx application page.



Till now, we have learnt how to create pods and services by using the manifest files. But when we delete pod it will not create the another pod after deletion. To overcome this we can use the deployment object in the manifest file.

First, delete the previous pods and services.

```
kubectl get all
kubectl delete service/demo-service
kubectl delete pod/demo-pod
```

```
[root@ip-172-31-27-151 ~]# kubectl get all
NAME           READY   STATUS    RESTARTS   AGE
pod/demo-pod   1/1     Running   0          2d18h

NAME                TYPE        CLUSTER-IP   EXTERNAL-IP
service/kubernetes ClusterIP  10.100.0.1   <none>
[root@ip-172-31-27-151 ~]# kubectl delete service/demo-service
service "demo-service" deleted
[root@ip-172-31-27-151 ~]# kubectl delete pod/demo-pod
pod "demo-pod" deleted
[root@ip-172-31-27-151 ~]# kubectl get all
NAME           TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes ClusterIP  10.100.0.1   <none>      443/TCP   2d20h
[root@ip-172-31-27-151 ~]#
```

Now, rewrite the manifest file, which pull the register app from the docker hub and create a pod out of it.

But while creating a pod we will make sure that in case something goes wrong with the pod it is going to recreate again and again. Also, incase we commit the latest image over dockerhub our kubernetes is going to terminate the existing pods and create the new pods with the latest image.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: valaxy-regapp
  labels:
    app: regapp

spec:
  replicas: 2
  selector:
    matchLabels:
      app: regapp

  template:
    metadata:
      labels:
        app: regapp
    spec:
      containers:
        - name: regapp
          image: valaxy/regapp
          imagePullPolicy: Always
          ports:
            - containerPort: 8080
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 1
```

Explanation:

<pre>apiVersion: apps/v1 kind: Deployment metadata: name: valaxy-regapp labels: app: regapp</pre>	<p>This is going to tell what is Deployment name and the Deployment Label.</p> <p>Deployment Name - Valaxy Registry App</p> <p>Label - Regapp</p> <p>Now, this deployment need a pod, because whenever we execute a Deployment, it is going to create pods in the backend.</p>
<pre>template: metadata: labels: app: regapp spec: containers: - name: regapp image: valaxy/regapp imagePullPolicy: Always ports: - containerPort: 8080</pre>	<p>This template tells, how to create a pod and what pod it has to launch.</p> <p>In this we have a pod definition. Then, this pod internally requires a container.</p> <p>Spec, is going to tell the container Definition. Container definition nothing but the details from where it can pull the image and what is the image pulling policy.</p> <p>Whenever we execute this deployment file it always pull the latest image from the GitHub repository.</p> <p>Next, on which port we can expose this container application.</p>
<pre>spec: replicas: 2 selector: matchLabels: app: regapp</pre>	<p>By using this information, it is going to create a replica set by matching the labels in the template.</p>
<pre>strategy: type: RollingUpdate rollingUpdate: maxSurge: 1 maxUnavailable: 1</pre>	<p>It represents, in case there is a new image in our GitHub, how it is going to create a pod with the new image.</p> <p>Let's take an example that her we have two replicas, it won't delete both. At a time, it is going to delete one container, and create a new container with new image. It will make sure that the new container is serving the application properly. Then it is going to terminate the 2nd pod and create a new container with the latest image.</p>

That is how we will make sure that our application is not going down.

This is how, we write the deployment file. To run the deployment file we need the service also. Let's take how we write the service file.

```
apiVersion: v1
kind: Service

metadata:
  name: valaxy-service
  labels:
    app: regapp
spec:
  selector:
    app: regapp

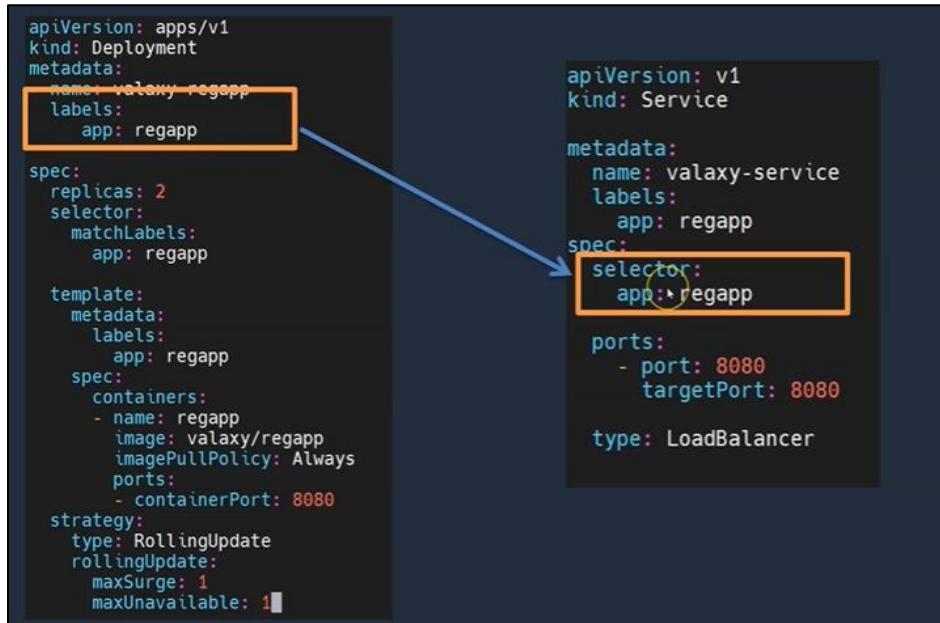
  ports:
    - port: 8080
      targetPort: 8080

  type: LoadBalancer
```

<code>apiVersion: v1</code>	This tells, what kind of resource we are creating.
<code>metadata:</code> <code>name: valaxy-service</code> <code>labels:</code> <code>app: regapp</code>	In this, we define the metadata. The name and what is the label.
<code>spec:</code> <code>selector:</code> <code>app: regapp</code>	In specification, here we are using the 'Selector'. It confirms which Deployment it should select. This selector information and Deployment label, should match.
<code>type: LoadBalancer</code>	We are using service type as a load balancer.
<code>ports:</code> <code>- port: 8080</code> <code>targetPort: 8080</code>	It tells, on which port no we are exposing our service at cluster level. Next, Target Port, whenever a new request comes to our

load balancer, on which port no. it is going to talk with the container.

This is how we need to write the service manifest file. Make sure the labels whatever we have given in the deployment file should match with the label of the service file. Similarly, the port number we mention in the Deployment file should match with the Service file.



Now create a deployment file.

```
vi regapp-deployment.yml
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: aman-regapp
  labels:
    app: regapp
spec:
  replicas: 2
  selector:
    matchLabels:
      app: regapp
  template:
    metadata:
      labels:
        app: regapp
```

```

spec:
  containers:
    - name: aman-regapp
      image: amanduggal001/regapp
      imagePullPolicy: Always
      ports:
        - containerPort: 8080
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 1

```

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: aman-regapp
  labels:
    app: regapp
spec:
  replicas: 2
  selector:
    matchLabels:
      app: regapp
  template:
    metadata:
      labels:
        app: regapp
    spec:
      containers:
        - name: aman-regapp
          image: amanduggal001/regapp
          imagePullPolicy: Always
          ports:
            - containerPort: 8080
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 1
~
~
~
```

kubectl apply -f regapp-deployment.yml
 kubectl get pods

NAME	READY	STATUS	RESTARTS	AGE
aman-regapp-7dfd865c78-wj9rs	1/1	Running	0	2m23s
aman-regapp-7dfd865c78-wlfqf	1/1	Running	0	2m23s

```
kubectl get all
```

NAME						
	READY	STATUS	RESTARTS	AGE		
pod/aman-regapp-7dfd865c78-wj9rs	1/1	Running	0	3m54s		
pod/aman-regapp-7dfd865c78-wlfqf	1/1	Running	0	3m54s		
NAME						
service/kubernetes	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE	
service/kubernetes	ClusterIP	10.100.0.1	<none>	443/TCP	2d23h	
NAME						
deployment.apps/aman-regapp	READY	UP-TO-DATE	AVAILABLE	AGE		
deployment.apps/aman-regapp	2/2	2	2	3m54s		
NAME						
replicaset.apps/aman-regapp-7dfd865c78	DESIRED	CURRENT	READY	AGE		
replicaset.apps/aman-regapp-7dfd865c78	2	2	2	3m54s		

kubectl get pod -o wide --#It tells each pod in which workload it is running.

NAME							NOMINATED NODE	READINESS GATES
	READY	STATUS	RESTARTS	AGE	IP	NODE		
aman-regapp-7dfd865c78-wj9rs	1/1	Running	0	5m4s	192.168.49.200	ip-192-168-58-155.ap-southeast-1.compute.internal	<none>	<none>
aman-regapp-7dfd865c78-wlfqf	1/1	Running	0	5m4s	192.168.22.150	ip-192-168-16-10.ap-southeast-1.compute.internal	<none>	<none>

Now, let's create a service in such a way that the traffic should flow through any of these 2 containers.

For that write the service manifest file.

```
vi regapp-service.yml
```

```
apiVersion: v1
kind: Service
metadata:
  name: app-service
  labels:
    app: regapp
spec:
  selector:
    app: regapp
  ports:
    - port: 8080
      targetPort: 8080
  type: LoadBalancer
```

```

apiVersion: v1
kind: Service
metadata:
  name: app-service
  labels:
    app: regapp
spec:
  selector:
    app: regapp
  ports:
    - port: 8080
      targetPort: 8080
  type: LoadBalancer
~
```

kubectl get all

```

[root@ip-172-31-27-151 tmp]# kubectl get all
NAME                           READY   STATUS    RESTARTS   AGE
pod/aman-regapp-7dfd865c78-wj9zs   1/1    Running   0          14m
pod/aman-regapp-7dfd865c78-wlfqf   1/1    Running   0          14m

NAME              TYPE        CLUSTER-IP   EXTERNAL-IP
service/app-service   LoadBalancer   10.100.182.238   a77413e9860ea4e43a555e4ed143f688-929134509.ap-southeast-1.elb.amazonaws.com   PORT(S)
                                                               8080:30508/TCP   29s
service/kubernetes   ClusterIP   10.100.0.1    <none>           443/TCP   3d

NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/aman-regapp   2/2     2           2          14m

NAME             DESIRED   CURRENT   READY   AGE
replicaset.apps/aman-regapp-7dfd865c78   2        2         2       14m
[root@ip-172-31-27-151 tmp]#
```

kubectl describe service/app-service --#This will tell about service where it is mapped and what are the backend parts it is going to send.

```

[root@ip-172-31-27-151 tmp]# kubectl describe service/app-service
Name:                   app-service
Namespace:              default
Labels:                 app=regapp
Annotations:            <none>
Selector:               app=regapp
Type:                   LoadBalancer
IP Family Policy:      SingleStack
IP Families:           IPv4
IP:                     10.100.182.238
IPs:                    10.100.182.238
LoadBalancer Ingress:  a77413e9860ea4e43a555e4ed143f688-929134509.ap-southeast-1.elb.amazonaws.com
Port:                   <unset>  8080/TCP
TargetPort:              8080/TCP
NodePort:                <unset>  30508/TCP
Endpoints:              192.168.22.150:8080,192.168.49.200:8080
Session Affinity:       None
External Traffic Policy: Cluster
Events:
  Type  Reason          Age    From            Message
  ----  ----          ----   ----            -----
  Normal EnsuringLoadBalancer 2m3s  service-controller  Ensuring load balancer
  Normal EnsuredLoadBalancer  2m   service-controller  Ensured load balancer
```

Validate the Load Balancer in the AWS Instance.

The screenshot shows the AWS CloudFormation console. On the left, a sidebar navigation includes 'Volumes', 'Snapshots', 'Lifecycle Manager', 'Network & Security' (with 'Security Groups', 'Elastic IPs', 'Placement Groups', 'Key Pairs', 'Network Interfaces'), 'Load Balancing' (selected), 'Load Balancers' (selected), 'Target Groups', 'Trust Stores New', and 'Auto Scaling' (with 'Auto Scaling Groups'). The main content area displays 'Load balancers (1/1)' with a table titled 'Load balancer: a77413e9860ea4e43a555e4ed143f688'. The table has columns: 'Instance ID', 'Name', 'Health status', 'Health status description', and 'Secure'. Two instances are listed: 'i-00b5f7f5e397f18f4' (nikkicloud-nginx-26d350b2-Node) and 'i-0125899fcc7804880' (nikkicloud-nginx-26d350b2-Node), both marked as 'In-service'.

Now, validate if your application is opening on browser or not.

Use DNSname:portnumber --#Since we have define the port number 8080 so we need to mention the port number to access the application.

Example: a77413e9860ea4e43a555e4ed143f688-929134509.ap-southeast-1.elb.amazonaws.com:8080

The screenshot shows a web browser window with the URL 'a77413e9860ea4e43a555e4ed143f688-929134509.ap-southeast-1.elb.amazonaws.com:8080'. The page title is 'Apache Tomcat/10.1.19'. The main content area features a green banner with the text 'If you're seeing this, you've successfully installed Tomcat. Congratulations!'. Below the banner is a cartoon cat icon. To the right of the cat, there is a list of 'Recommended Reading' links: 'Security Considerations How-To', 'Manager Application How-To', and 'Clustering/Session Replication How-To'. On the right side of the page, there are three buttons: 'Server Status', 'Manager App', and 'Host Manager'. At the bottom left, there is a link 'Developer Quick Start'.

Use DNSname:portnumber/webapp --#To access the application.

Final Registration Form (Last date 12 march 2024)

First Name: Enter your first name Last Name: Enter your last name

Email: Enter your email

Date of Birth: mm / dd / yyyy

Password: Enter your password

Re-type Password: Re-Enter your password

Contact: Enter your Mobile Number

Gender: Male Female

It is opening, which means we are successfully able to deploy our application on pod.

Now to validate, incase if any pod is deleted it should be reprovisioned and our application not get affected.

To Test, delete one pod.

```
kubectl delete pod aman-regapp-7dfd865c78-wj9rs
```

```
[root@ip-172-31-27-151 tmp]# kubectl get pods
NAME                  READY   STATUS    RESTARTS   AGE
aman-regapp-7dfd865c78-wj9rs   1/1     Running   0          31m
aman-regapp-7dfd865c78-wlfqf   1/1     Running   0          31m
[root@ip-172-31-27-151 tmp]# kubectl delete pod aman-regapp-7dfd865c78-wj9rs
pod "aman-regapp-7dfd865c78-wj9rs" deleted
[root@ip-172-31-27-151 tmp]# kubectl get pods
NAME                  READY   STATUS    RESTARTS   AGE
aman-regapp-7dfd865c78-gk617   1/1     Running   0          5s
aman-regapp-7dfd865c78-wlfqf   1/1     Running   0          32m
[root@ip-172-31-27-151 tmp]#
```

So, when we delete our pod, another pod is created immediately. That is because our replica set makes sure that we have 2 pods at any point of time.

Now, it's time to integrate our Kubernetes with our CI/CD pipeline. For that, first we need to integrate Kubernetes with the Ansible because we cannot run these kubectl commands manually.

We will tell Ansible to run the kubectl commands whenever we want to do the new deployment. That is how we use ansible to deploy the latest application on Kubernetes cluster.

Integrate Kubernetes bootstrap server with Ansible

- **On Bootstrap server**

- Create ansadmin
- Add ansadmin to sudoers files
- Enable password based login



- **On Ansible Node**

- Add to hosts file
- Copy ssh keys
- Test the connection



On bootstrap image server (Kubernetes server), create a user '**ansibleadmin**' and add that user to the Sudoers file.

```
useradd ansibleadmin  
passwd ansibleadmin
```

```
visudo --#Add the below user in the file.
```

```
ansibleadmin ALL=(ALL) NOPASSWD: ALL
```

```
##  
## The COMMANDS section may have other options added to it.  
##  
## Allow root to run any commands anywhere  
root    ALL=(ALL)      ALL  
ansibleadmin ALL=(ALL) NOPASSWD: ALL  
## Allows members of the 'sys' group to run networking, software  
## service management apps and more.  
# %sys ALL = NETWORKING, SOFTWARE, SERVICES, STORAGE, DELEGATING
```

Now, enable the password based authentication.

```
vi /etc/ssh/sshd_config --#Enable the password authentication.
```

```

# DON't read the user's ~/.ssh/ and /var/ssh keys
#IgnoreRhosts yes

# To disable tunneled clear text passwords, change to no here!
#PasswordAuthentication yes
PasswordAuthentication yes
#PermitEmptyPasswords no
#PasswordAuthentication no

# Change to no to disable s/key passwords
#ChallengeResponseAuthentication yes
ChallengeResponseAuthentication no

# Kerberos options
#KerberosAuthentication no
#KerberosOrLocalPasswd yes
#KerberosTicketCleanup yes
#KerberosGetAFSToken no
#KerberosGetAFSToken no

```

service sshd reload

Now login to ansible server.

cd /opt/docker

rewrite the deploy_repapp.yml file. Because till now we are deploying image on a Docker Container.
Now we need to deploy it on the Kubernetes as a pod.

Rename the existing .yml files for the meaningful name -

```

mv deploy_repapp.yml docker_deployment_repapp.yml
mv regapp.yml create_image_regapp.yml

```

```

[root@AnsibleServer docker]# ll
total 16
-rw-rw-r-- 1 ansibleadmin ansibleadmin 352 Mar 10 17:35 create_image_regapp.yml
-rw-rw-r-- 1 ansibleadmin ansibleadmin 388 Mar 11 17:08 docker_deployment_repapp.yml
-rw-r--r-- 1 root         root          129 Mar 10 11:53 Dockerfile
-rw-rw-r-- 1 ansibleadmin ansibleadmin 2628 Mar 11 18:31 webapp.war
[root@AnsibleServer docker]#

```

Now, we need to add Bootstrap image to the hosts file. In this case are creating the inventory file here itself.

vi hosts

localhost --#Incase if required

```

[kubernetes]
172.31.27.151 --#Kubernetes private Ip

```

```

[ansible]
172.31.19.187 --#Ansible private Ip

```

```

localhost

[kubernetes]
172.31.27.151

[ansible]
172.31.19.187
~
~
```

Now we have added the server and now we need to copy the keys on kubernetes server.

On ansible server

sudo su ansibleadmin

ssh-copy-id 172.31.27.151 --#Private Ip of Kubernetes server.

Are you sure you want to continue connecting (yes/no)? yes --#Put this line to yes.

```

[ansibleadmin@AnsibleServer docker]$ ssh-copy-id 172.31.27.151
/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/ansibleadmin/.ssh/id_rsa.pub"
The authenticity of host '172.31.27.151 (172.31.27.151)' can't be established.
ECDSA key fingerprint is SHA256:FopPQlUSwVFlY1Oy50+8nU/nVz/eUQ24o8bX/7Ioy2Q.
ECDSA key fingerprint is MD5:fb:a5:8c:6c:b6:64:93:69:3b:8e:be:fb:72:03:e6:22.
Are you sure you want to continue connecting (yes/no)? yes
/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
ansibleadmin@172.31.27.151's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh '172.31.27.151'"
and check to make sure that only the key(s) you wanted were added.

[ansibleadmin@AnsibleServer docker]$
```

ansible -i hosts all -a uptime --# -i hosts means in the current location we have hosts file and in this host file whatever server mentioned, ping to that servers.

```

[ansibleadmin@AnsibleServer docker]$ ansible -i hosts all -a uptime
[WARNING]: Platform linux on host 172.31.27.151 is using the discovered Python interpreter at /usr/bin/python, but future installation of another Python interpreter could change this. See https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more information.
172.31.27.151 | CHANGED | rc=0 >>
 18:38:00 up 3 days, 2:54, 2 users, load average: 0.00, 0.00, 0.00
[WARNING]: Platform linux on host localhost is using the discovered Python interpreter at /usr/bin/python, but future installation of another Python interpreter could change this. See https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more information.
localhost | CHANGED | rc=0 >>
 18:38:01 up 5 days, 9:30, 3 users, load average: 0.08, 0.02, 0.01
[WARNING]: Platform linux on host 172.31.19.187 is using the discovered Python interpreter at /usr/bin/python, but future installation of another Python interpreter could change this. See https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more information.
172.31.19.187 | CHANGED | rc=0 >>
 18:38:01 up 5 days, 9:30, 2 users, load average: 0.08, 0.02, 0.01
[ansibleadmin@AnsibleServer docker]$
```

It means, it can able to communicate with our Kubernetes server.

On Ansible server, we need to copy the ansibleadmin keys to the root user, otherwise it cannot able to login as a root user.

ssh-copy-id root@172.31.27.151 --#IP of Kubernetes server.

Enter the password for the Kubernetes root user.

Setup the password of the root user in the Kubernetes server.

(On Kubernetes server)

passwd root --#Enter the password.

```
[ansibleadmin@AnsibleServer docker]$ ssh-copy-id root@172.31.27.151
/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/ansibleadmin/.ssh/id_rsa.pub"
/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
root@172.31.27.151's password:
Permission denied, please try again.
root@172.31.27.151's password:
Permission denied, please try again.
root@172.31.27.151's password:

Number of key(s) added: 1

Now try logging into the machine, with:    "ssh 'root@172.31.27.151'"
and check to make sure that only the key(s) you wanted were added.

[ansibleadmin@AnsibleServer docker]$
```

Create ansible playbooks for deploy and service files

Till now we see how to integrate our bootstrap server with Ansible. Now let's execute our deployment and service files using Ansible. For that we have to create Ansible Playbooks for deployment and service files.

On Ansible Server, create a playbook.

```
vi kube_deploy.yml
```

```
---
- hosts: kubernetes
  become: true
  tasks:
    - name: deploy regapp on kubernetes
      command: kubectl apply -f /tmp/regapp-deployment.yml
```

```
---
- hosts: kubernetes
  become: true
  tasks:
    - name: deploy regapp on kubernetes
      command: kubectl apply -f /tmp/regapp-deployment.yml
~
```

Now, similarly create an playbook for service file.

```
vi kube_service.yml
---
- hosts: kubernetes
  become: true
  tasks:
    - name: deploy service
      command: kubectl apply -f /tmp/regapp-service.yml
```

```
---
- hosts: kubernetes
  become: true
  tasks:
    - name: deploy service
      command: kubectl apply -f /tmp/regapp-service.yml
~
```

Before executing the Ansible playbook, delete the exiting pods, deployments and services on Kubernetes server.

```
kubectl delete -f regapp-deployment.yml \
kubectl delete -f regapp-service.yml
kubectl get all
```

```
[root@ip-172-31-27-151 tmp]# kubectl get all
NAME                           READY   STATUS    RESTARTS   AGE
pod/aman-regapp-7dfd865c78-gk617  1/1    Running   0          137m
pod/aman-regapp-7dfd865c78-wlfqf  1/1    Running   0          169m

NAME              TYPE        CLUSTER-IP       EXTERNAL-IP
service/app-service LoadBalancer  10.100.182.238  a77413e9860ea4e43a55e4ed143f688-929134509.ap-southeast-1.elb.amazonaws.com
service/kubernetes ClusterIP  10.100.0.1     <none>

NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/aman-regapp  2/2     2           2          169m

NAME          DESIRED   CURRENT   READY   AGE
replicaset.apps/aman-regapp-7dfd865c78  2        2        2        169m
[root@ip-172-31-27-151 tmp]# kubectl delete -f regapp-deployment.yml
deployment.apps "aman-regapp" deleted
[root@ip-172-31-27-151 tmp]# kubectl delete -f regapp-service.yml
service "app-service" deleted
[root@ip-172-31-27-151 tmp]# kubectl get all
NAME              TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)   AGE
service/kubernetes ClusterIP  10.100.0.1     <none>        443/TCP   3d2h
[root@ip-172-31-27-151 tmp]#
```

Now, execute the playbook on the Ansible server.

```
ansible-playbook kube_deploy.yml
```

```
[ansibleadmin@AnsibleServer docker]$ ansible-playbook kube_deploy.yml
[WARNING]: Could not match supplied host pattern, ignoring: kubernetes

PLAY [kubernetes] ****
skipping: no hosts matched

PLAY RECAP ****
[ansibleadmin@AnsibleServer docker]$ ls
```

If we directly try to execute the Ansible playbook, it will skip and not execute the file properly. Since we have mentioned the hosts: kubernetes, so we need to provide the path of the hosts file where we have mentioned the Kubernetes group.

```
ansible-playbook -i /opt/docker/hosts kube_deploy.yml
```

```
[ansibleadmin@AnsibleServer docker]$ ansible-playbook -i /opt/docker/hosts kube_deploy.yml

PLAY [kubernetes] *****

TASK [Gathering Facts] *****
[WARNING]: Platform linux on host 172.31.27.151 is using the discovered Python
interpreter at /usr/bin/python, but future installation of another Python
interpreter could change this. See https://docs.ansible.com/ansible/2.9/referen
ce_appendices/interpreter_discovery.html for more information.
ok: [172.31.27.151]

TASK [deploy regapp on kubernetes] *****
fatal: [172.31.27.151]: FAILED! => {"changed": false, "cmd": "kubectl apply -f /root/regapp-deployment.yml", "msg": "[Errno 2] No such file or directory", "rc": 2}

PLAY RECAP *****
172.31.27.151 : ok=1    changed=0    unreachable=0    failed=1    s
kipped=0      rescued=0   ignored=0
```

I am getting this error. We should run this playbook as a root user. For that we need to mention in playbook the user is root, by this we didn't need to define the path also because by default it will check under the root directory.

```
vi kube_deploy.yml
```

```
---
- hosts: kubernetes
  user: root
  tasks:
    - name: deploy regapp on kubernetes
      command: kubectl apply -f regapp-deployment.yml
```

```
vi kube_service.yml
```

```
---
- hosts: kubernetes
  user: root
  tasks:
    - name: deploy service
      command: kubectl apply -f regapp-service.yml
```

Note: Make sure the files are present in the root location. If not copy the file and paste it in the root location.

```
[root@ip-172-31-27-151 tmp]# cp regapp-deployment.yml /root
[root@ip-172-31-27-151 tmp]# cp regapp-service.yml /root
[root@ip-172-31-27-151 tmp]# cd /root
[root@ip-172-31-27-151 ~]# ls
regapp-deployment.yml  regapp-service.yml
[root@ip-172-31-27-151 ~]#
```

Now, execute the playbooks.

```
ansible-playbook -i /opt/docker/hosts kube_deploy.yml
ansible-playbook -i /opt/docker/hosts kube_service.yml
```

Once executed, check on the Kubernetes server if the pods and services are created there after executing the playbooks in the ansible server.

kubectl get all

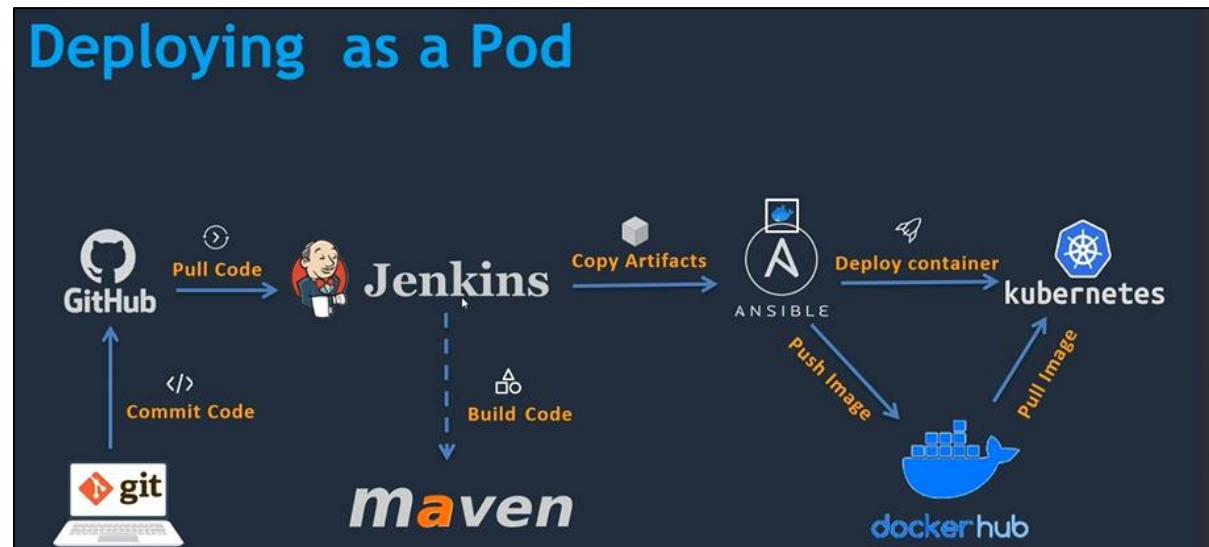
```
[root@ip-172-31-27-151 ~]# kubectl get all
NAME                                         READY   STATUS    RESTARTS   AGE
pod/aman-regapp-7dfd865c78-654c6            1/1     Running   0          3m7s
pod/aman-regapp-7dfd865c78-xgmn8            1/1     Running   0          3m7s

NAME                TYPE        CLUSTER-IP      EXTERNAL-IP
service/app-service LoadBalancer   10.100.180.125  a33cc63c5115b4e9acfcc3c2405dc3-620627145.ap-southeast-1.elb.amazonaws.com
service/kubernetes  ClusterIP   10.100.0.1     <none>
PORT(S)           AGE
8080:30940/TCP  89s
443/TCP          3d4h

NAME              READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/aman-regapp  2/2     2           2           3m7s

NAME             DESIRED   CURRENT   READY   AGE
replicaset.apps/aman-regapp-7dfd865c78  2        2        2       3m8s
[root@ip-172-31-27-151 ~]#
```

Create Jenkins deployment job for Kubernetes



Until now, we have set up our Kubernetes environment and we created deployment and service files and we have integrated with Ansible.

Now, we could able to initialize our deployment and service files with the help of Ansible.

Now, we are make use of Jenkins rather than executing those playbooks manually from Ansible. For that we will create a Jenkins job that is a deployment job. It initialize the Ansible playbook to initialize deploy and service files on Kubernetes.

- Login to Jenkins GUI.
- Create a new Job '**Deploy_On_Kubernetes**'
- Create a Freestyle project.

Enter an item name

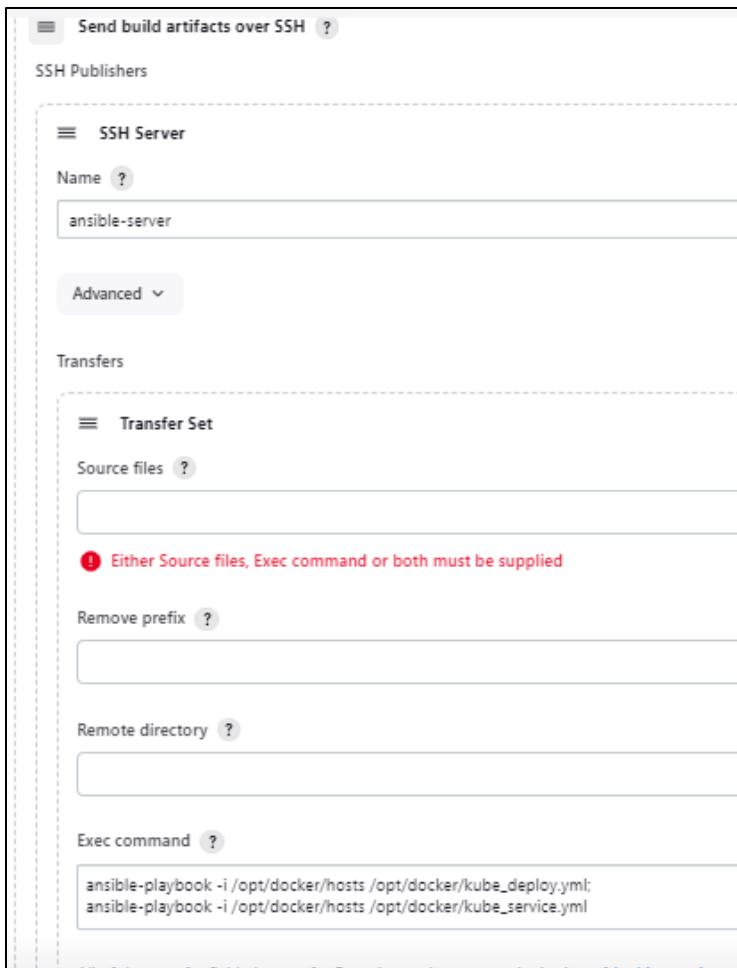
» Required field

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

- Add the Post build action '**Send build artifacts over SSH**'.
- Name: ansible-server

Exec Command -

```
ansible-playbook -i /opt/docker/hosts /opt/docker/kube_deploy.yml;  
ansible-playbook -i /opt/docker/hosts /opt/docker/kube_service.yml
```



So basically, we have created the Jenkins job just to execute these two commands to execute the ansible playbooks.

This job will execute the Ansible playbooks and Ansible playbook execute the Kubernetes manifest files which will create the deployments and services.

To test the job, first delete the existing deployment and services from the Kubernetes server.

```
kubectl delete deployment.apps/aman-regapp
kubectl delete service/app-service
kubectl get all
```

```
[root@ip-172-31-27-151 ~]# kubectl get all
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   3d4h
[root@ip-172-31-27-151 ~]#
```

- Now build the job '**Deploy_On_Kubernetes**'

The screenshot shows the Jenkins interface for a build named 'Deploy_On_Kubernetes' (Build #1). The 'Console Output' tab is selected. The log output is as follows:

```

Started by user Aman Duggal
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/Deploy_On_Kubernetes
SSH: Connecting from host [JenkinsServer]
SSH: Connecting with configuration [ansible-server] ...
SSH: EXEC: completed after 7,805 ms
SSH: Disconnecting configuration [ansible-server] ...
SSH: Transferred 0 file(s)
Finished: SUCCESS

```

Check the Kubernetes server, the deployments and services are created.

```

[root@ip-172-31-27-151 ~]# kubectl get all
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP  10.100.0.1 <none>        443/TCP   3d4h
[root@ip-172-31-27-151 ~]# kubectl get all
NAME             READY   STATUS    RESTARTS   AGE
pod/aman-regapp-7dfd865c78-m5fdd  1/1    Running   0          73s
pod/aman-regapp-7dfd865c78-xqpjd  1/1    Running   0          73s

NAME           TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/app-service   LoadBalancer  10.100.11.228  a6304d80ld89ad495aa68014ldcaf7453-218160373.ap-southeast-1.elb.amazonaws.com  8080:30205/TCP   70s
service/kubernetes   ClusterIP  10.100.0.1 <none>        443/TCP   3d4h

NAME             READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/aman-regapp  2/2     2           2           73s
NAME           DESIRED   CURRENT   READY   AGE
replicaset.apps/aman-regapp-7dfd865c78  2       2       2       73s

```

Now, instead of running two playbooks together, we can also merge the content of deployment and services just add the task of other playbook in any one of the file.

`vi kube_deploy.yml`

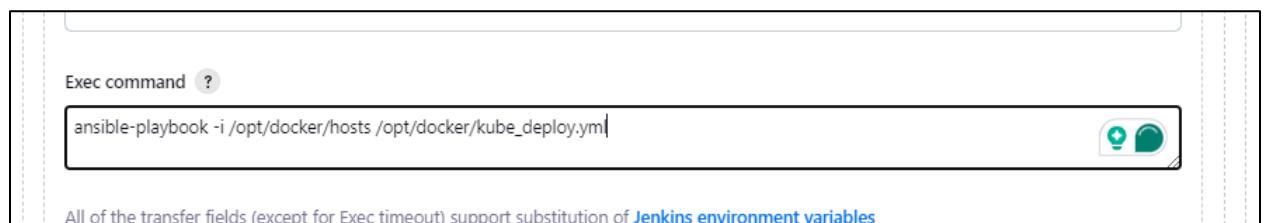
```

---
- hosts: kubernetes
  user: root
  tasks:
    - name: deploy regapp on kubernetes
      command: kubectl apply -f regapp-deployment.yml
    - name: create service on kubernetes
      command: kubectl apply -f regapp-service.yml

```

```
---  
- hosts: kubernetes  
  user: root  
  tasks:  
    - name: deploy regapp on kubernetes  
      command: kubectl apply -f regapp-deployment.yml  
    - name: create service on kubernetes  
      command: kubectl apply -f regapp-service.yml  
~  
~  
~
```

Go to Jenkins job and remove the other, as we don't required it more since all the content should be in kube_deploy.yml file.



Build the Job and test again.

Till now, we have created a job called '**Deploy_On_Kubernetes**' which helps us to execute the deployment and service file of Kubernetes.

But why we are executing those?

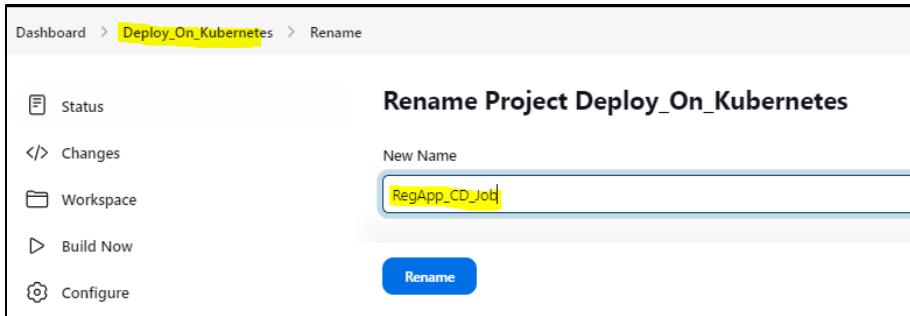
Because, if there is any changes to our source code, we need to build that job and it should create a latest image over Docker Hub. And with the latest image we need to deploy our pods.

Now, our intention is to update the image by changing the source code, so that our Kubernetes can deploy with the latest code.

That is how we can automate the entire CI/CD pipeline.

So far we have created the CD job which only creates the deployments, we need to create a CI job.

Just for better understanding, Rename the existing job i.e. '**Deploy_On_Kubernetes**' to '**RegApp CD job**'.



Now, create a CI job, we just need to pull the code from the GitHub and build in the Jenkins and run the Ansible playbook in our Ansible server which we already created a .yml file previously i.e '**create_image_regapp.yml**' in which it builds the image, retag the image and push it on the Docker Hub.

```
[ansibleadmin@AnsibleServer docker]$ cat create_image_regapp.yml
---
- hosts: ansible
  tasks:
    - name: Create docker image
      command: docker build -t regapp:latest .
      args:
        chdir: /opt/docker

    - name: create docker tag.
      command: docker tag regapp:latest amanduggal001/regapp:latest

    - name: Push the docker image to docker hub
      command: docker push amanduggal001/regapp:latest
[ansibleadmin@AnsibleServer docker]$
```

So, this file will create the latest image with the help of webapp.war file under the docker file and it commits on to the docker hub.

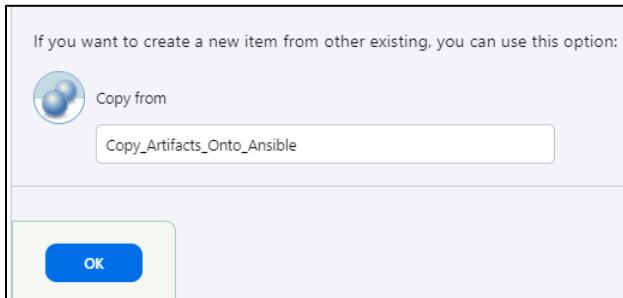
We previously created that Job on Jenkins i.e '**Copy_Artifacts_Onto_Ansible**'. This job will be get executed whenever there is change in source code.

Now we are creating the CI Job with the help of '**Copy_Artifacts_Onto_Ansible**' Job so that we can create an image and disable the previous job.

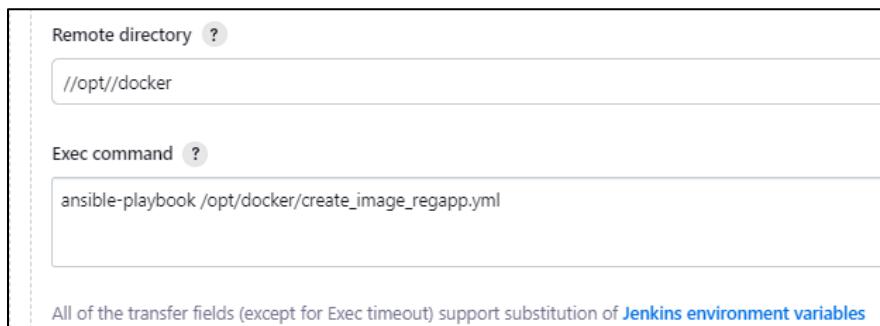
- Create a new Job ' RegAPP_CI_Job' and copy the configuration from the previous job i.e '**Copy_Artifacts_Onto_Ansible**'.

Enter an item name

» Required field



- **Description:** Build code with help of Maven and create an image on ansible and push it onto dockerhub.
- **Exec Command** - ansible-playbook /opt/docker/create_image_regapp.yml -#rename the yml file, since we renamed earlier and remove the other commands since we are deploying using the Kubernetes with other job.



First, we'll see until, if it is creating image successfully or not. For that change the source code from Git bash.

Go to Git Bash

```
cd D:/AWS/DevopsProject/hello-world/webapp/src/main/webapp
```

```
vi index.jsp
```

Change the source code.

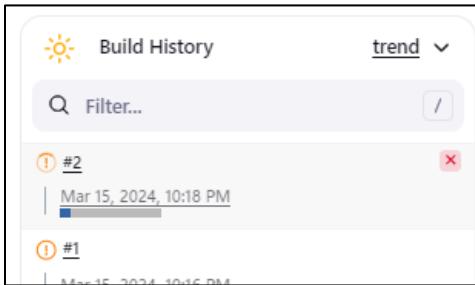
A screenshot of a terminal window titled 'MINGW64:d/AWS/DevopsProject/hello-world/webapp/src/main/webapp'. The window displays the content of the 'index.jsp' file, which contains HTML and Java code. The code includes a title with a placeholder 'Fill Until March', a link to a stylesheet, and a main div.

```
git add .
```

```
git commit -m "bugfix"
```

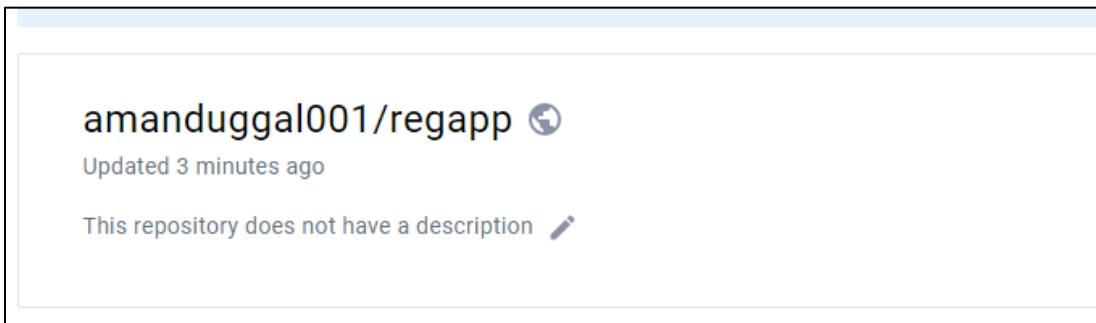
```
git push origin master
```

Once pushed the image, the job gets initializing.



Once the job is executed the images were updated.

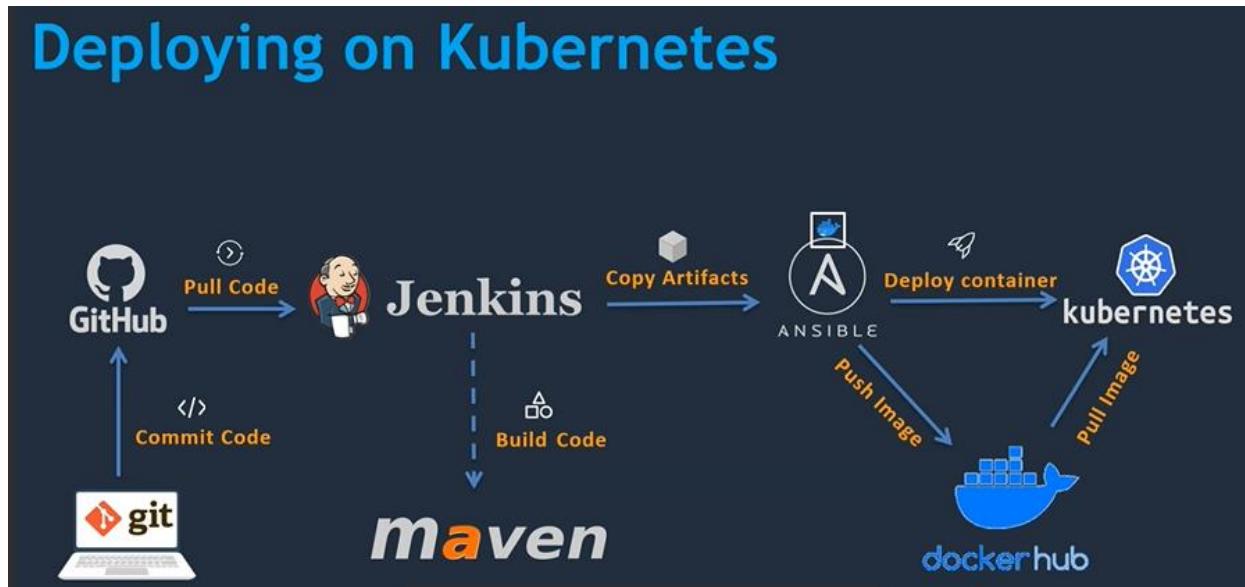
```
[ansibleadmin@AnsibleServer docker]$ docker images
REPOSITORY          TAG      IMAGE ID   CREATED        SIZE
amanduggal001/regapp    latest   5c63b010045b  56 seconds ago  460MB
regapp               latest   5c63b010045b  56 seconds ago  460MB
amanduggal001/regapp    <none>   726b3ead2dc6  4 days ago   460MB
amanduggal001/regapp    <none>   eldbf221af46  4 days ago   460MB
amanduggal001/regapp    <none>   5398975c833f  4 days ago   460MB
tomcat                latest   405afe63d576  9 days ago   455MB
[ansibleadmin@AnsibleServer docker]$
```



It means that our images are successfully created without any issues and committing onto the GitHub. Now we just need to integrate our CI and CD job so that what will happen, whenever the CI is successful it should initialize the CD job. And CD job should take the latest image from docker hub and perform the latest deployment.

NOTE: Please confirm if your docker service is started or not. My Job is getting failed and after validating the issue I found that the docker service is in active.

Enable rolling update to create pod from latest docker image



So far, what we have done, we are changing the code on our workstation and these changes we are committing onto our GitHub repository.

From there, our Jenkins is pulling the code and building with the help of Maven and it is generating the .war file, we call it as a Artifacts and pushing it onto the Ansible Server.

At the same time, it is also executing Ansible playbook, which can create an image with the artifacts and commit it into the docker hub repository.

Also, we have created a job which is CD job which can be initialized with the Jenkins and it is going to initialize the Ansible Playbook. This Ansible playbook executes the deployment and service files on Kubernetes cluster.

However, we are trying to integrate the CI and CD job so that whenever we do any changes over source code, it should be available under Kubernetes cluster.

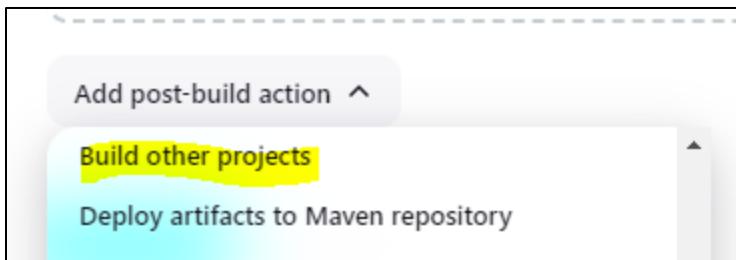
Let's go and integrate the both jobs.

- Open the Jenkins dashboard. Here we have our CI and CD jobs.

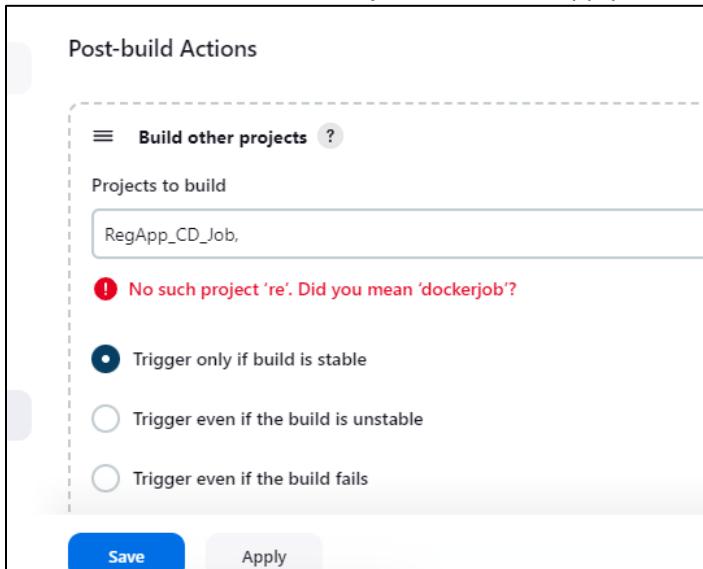
Stability: No recent builds failed	RegApp_CD_Job	100	11 hr	#1	N/A	8 sec	
	RegApp_CI_Job		24 min	#8	N/A	34 sec	

In CI job, there is an option to initialize another Jenkins job. That is where we will tell that initialize the CD job. So that once the CI job is completed, the CD job is going take the image and do the deployments.

- Configure the CI job > Add Post Build Action > Build other projects.



- Add the CD Job under the Projects to Build > Apply > Save.



Now, do some changes on the source code and commit it into the GitHub repository and this is going to initialize the Jenkins job and thus Jenkins job creates an Docker image and commit it over Docker Hub.

Then also, this Jenkins CI job initialize the CD job, which should be able to take the latest image from Docker Hub and do the deployment on our Kubernetes cluster.

Go to Git Bash

```
cd D:/AWS/DevopsProject/hello-world/webapp/src/main/webapp
```

```
vi index.jsp
```

Change the source code.

```
<!DOCTYPE html>
<html>

<head>
    <title>REGISTRATION FORM ---- Please (Fill Until March)</title>
    <link rel="stylesheet"
          href="style.css">
</head>

<body>
    <div class="main">
        <h1>Final Registration Form (Last date 12 march 2024)</h1>
```

```
git add .
git commit -m "bugfix"
git push origin master
```

Once pushed the image, the job gets initializing.

```
[root@ip-172-31-27-151 ~]# kubectl get all
NAME                                         READY   STATUS    RESTARTS   AGE
pod/aman-regapp-7dfd865c78-php5m           1/1     Running   0          50s
pod/aman-regapp-7dfd865c78-z5qrk           1/1     Running   0          50s

NAME                TYPE      CLUSTER-IP   EXTERNAL-IP
service/app-service LoadBalancer   10.100.101.50  aacd4e5466809499796367b3983e452e-774048406.ap-southeast-1.elb.amazonaws.com
service/kubernetes ClusterIP   10.100.0.1    <none>

NAME               READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/aman-regapp   2/2     2           2           50s

NAME              DESIRED   CURRENT   READY   AGE
replicaset.apps/aman-regapp-7dfd865c78  2        2         2       50s
```

There is an issue, the job could not able to recognize until or unless we do changes to our deployment and service files. Add the new entry in the deployment and service file.

(On Ansible server)

```
vi kube_deploy.yml
```

- name: update deployment with new pods if image updated in the docker hub.
 command: kubectl rollout restart deployment.apps/aman-regapp -#When you run this command, Kubernetes will restart the pods managed by the aman-regapp deployment, effectively deploying any changes made to the deployment configuration or restarting the pods for maintenance or updates.

```
---
- hosts: kubernetes
  user: root
  tasks:
    - name: deploy regapp on kubernetes
      command: kubectl apply -f regapp-deployment.yml
    - name: create service on kubernetes
      command: kubectl apply -f regapp-service.yml
    - name: update deployment with new pods if image updated in docker hub
      command: kubectl rollout restart deployment.apps/aman-regapp
~
```

So, this is going to replace our environment with the latest image.

Now, let's do a new commit, so the new commit is going to create the new image as well as new pod. So that we can see the new changes in the web browser.

Note: Whenever we commit new changes, it will be update over here.

This branch is 16 commits ahead of `yankils/hello-world:master`.

Commit	File	Date
Aman Duggal and Aman Duggal ne commit		065c9b4 · 26 minutes ago
server	Update pom.xml	3 years ago
webapp	ne commit	26 minutes ago

When our CI job should initiate, whenever it is running it should replace the .war file and create the latest image on the Docker Hub repository.

```
[ansibleadmin@AnsibleServer docker]$ ll
total 28
-rw-rw-r--. 1 ansibleadmin ansibleadmin 352 Mar 16 08:34 create_image_regapp.yml
-rw-rw-r--. 1 ansibleadmin ansibleadmin 388 Mar 11 17:08 docker_deployment_repa
pp.yml
-rw-r--r--. 1 root         root        129 Mar 10 11:53 Dockerfile
-rw-r--r--. 1 root         root        63 Mar 15 18:20 hosts
-rw-rw-r--. 1 ansibleadmin ansibleadmin 374 Mar 16 08:37 kube_deploy.yml
-rw-rw-r--. 1 ansibleadmin ansibleadmin 125 Mar 15 20:00 kube_service.yml
-rw-rw-r--. 1 ansibleadmin ansibleadmin 2648 Mar 16 08:17 webapp.war
```

Once this is done, our CD job should get executed and this CD job should terminate these pods one by one and should be able to launch the new pods with the new application.

```
[root@ip-172-31-27-151 ~]# kubectl get pods
NAME                  READY   STATUS    RESTARTS   AGE
aman-regapp-7dfd865c78-php5m   1/1     Running   0          33m
aman-regapp-7dfd865c78-z5qrk   1/1     Running   0          33m
```

Let's go and change our source code.

[Go to Git Bash](#)

```
cd D:/AWS/DevopsProject/hello-world/webapp/src/main/webapp
```

```
vi index.jsp
```

Change the source code.

```
</head>
<body>
    <div class="main">
        <h1>Final Registration Form (Last date 12 march 2024)</h1>
        <form action="">
            <label for="first">Engineer First Name:</label>
            <input type="text" id="first"
                   name="first"
                   placeholder="Enter your first name" required>
            <br>
            <label for="last">Last Name:</label>
            <input type="text" id="last"
                   name="last"
                   placeholder="Enter your last name" required>
            <br>
        </form>
    </div>

```

```
git add .
git commit -m "new commit"
git push origin master
```

Once pushed the image, the job gets initializing on both CI and CD.

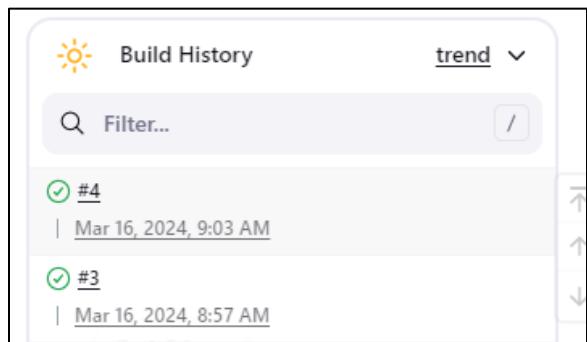
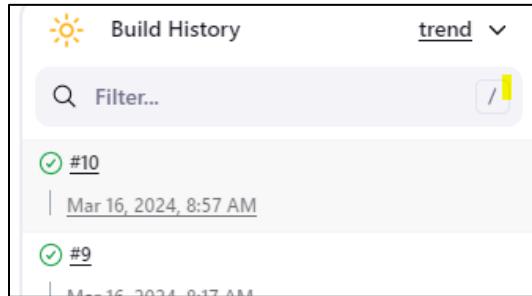


Image gets updated in the Docker Hub.

Add a short description for this repository
The short description is used to index your content on Docker Hub and in search engines. It's visible to users

amanduggal001/regapp

Updated 3 minutes ago

This repository does not have a description

Previous Pods are terminated, and new pods are created.

```
[root@ip-172-31-27-151 ~]# kubectl get all
NAME                                         READY   STATUS    RESTARTS   AGE
pod/aman-regapp-844796fb65-fc5p4   1/1     Running   0          113s
pod/aman-regapp-844796fb65-nm52n   1/1     Running   0          113s

NAME           TYPE      CLUSTER-IP   EXTERNAL-IP
service/app-service LoadBalancer  10.100.101.50  aacd4e5466809499796367b3983e452e-774048406.ap-southeast-1.elb.amazonaws.com
service/kubernetes ClusterIP   10.100.0.1    <none>
PORT(S)        AGE
8080:30964/TCP 47m
443/TCP       3d16h

NAME          READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/aman-regapp  2/2     2           2           47m

NAME          DESIRED  CURRENT   READY   AGE
replicaset.apps/aman-regapp-6564656bdf  0        0        0        7m53s
replicaset.apps/aman-regapp-7dfd865c78  0        0        0        47m
replicaset.apps/aman-regapp-844796fb65  2        2        2        113s
[root@ip-172-31-27-151 ~]#
```

webapp.war file gets updated with the latest .war file.

```
[ansibleadmin@AnsibleServer docker]$ ll
total 28
-rw-rw-r--. 1 ansibleadmin ansibleadmin 352 Mar 16 08:34 create_image_regapp.yml
l
-rw-rw-r--. 1 ansibleadmin ansibleadmin 388 Mar 11 17:08 docker_deployment_repa
pp.yml
-rw-r--r--. 1 root         root        129 Mar 10 11:53 Dockerfile
-rw-r--r--. 1 root         root        63 Mar 15 18:20 hosts
-rw-rw-r--. 1 ansibleadmin ansibleadmin 374 Mar 16 08:37 kube_deploy.yml
-rw-rw-r--. 1 ansibleadmin ansibleadmin 125 Mar 15 20:00 kube_service.yml
-rw-rw-r--. 1 ansibleadmin ansibleadmin 2619 Mar 16 09:03 webapp.war
[ansibleadmin@AnsibleServer docker]$
```

Application is working and opened in the web browser.

The screenshot shows a web browser window with the following details:

- Address Bar:** Not secure aacd4e5466809499796367b3983e452e-774048406.ap-southeast-1.elb.amazonaws.com:8080/webapp/
- Tab Bar:** My Portal, Rancher, Notepad | Online, PeopleStrong Home, Document 360, Outlook, Udemy - Learning P..., Chat GPT, Client Details status..
- Title:** DevOps Registrattion Form (Final)
- Text:** Please fill in this form to create an account.
- Input Fields:**
 - Enter Name: Enter Full Name
 - Enter mobile: Enter moible number
 - Enter Email: Enter Email
 - Password: Enter Password
 - Repeat Password: Repeat Password
- Text:** By creating an account you agree to our [Terms & Privacy](#).
- Buttons:** Register, Already have an account? [Sign in](#).