

How to Integrate jenkins with docker ci/cd

Step1: Launch Ubuntu 18.0 instance for Jenkins

❖ Installing the Default JRE/JDK

- `sudo apt update`
- `sudo apt install default-jdk`
- `java -version`

❖ Installing Jenkins

- **First, add the repository key to the system:**
`wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -`
- **append the Debian package repository address to the server's `sources.list`:**
`sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'`
- `sudo apt update`
- `sudo apt install jenkins`
- **Starting Jenkins**
`systemctl start jenkins`
- **To check the status**

```
root@ip-172-31-8-229:/home/ubuntu# systemctl status jenkins
● jenkins.service - LSB: Start Jenkins at boot time
   Loaded: (/etc/init.d/jenkins; generated)
   Active: active (exited) since Sun 2021-12-26 16:11:20 UTC; 2min 59s ago
     Docs: man:systemd-sysv-generator(8)
    Tasks: 0 (limit: 2347)
   CGroup: /system.slice/jenkins.service

Dec 26 16:11:19 ip-172-31-8-229 systemd[1]: Starting LSB: Start Jenkins at boot
Dec 26 16:11:19 ip-172-31-8-229 jenkins[7599]: Correct java version found
Dec 26 16:11:19 ip-172-31-8-229 jenkins[7599]: * Starting Jenkins Automation Se
Dec 26 16:11:19 ip-172-31-8-229 su[7650]: Successful su for jenkins by root
Dec 26 16:11:19 ip-172-31-8-229 su[7650]: + ??? root:jenkins
Dec 26 16:11:19 ip-172-31-8-229 su[7650]: pam_unix(su:session): session opened f
Dec 26 16:11:19 ip-172-31-8-229 su[7650]: pam_unix(su:session): session closed f
Dec 26 16:11:20 ip-172-31-8-229 jenkins[7599]: ...done.
Dec 26 16:11:20 ip-172-31-8-229 systemd[1]: Started LSB: Start Jenkins at boot t
lines 1-16/16 (END)
```

- `Install docker`

```
> apt install docker.io -y
> usermod -aG docker jenkins
> systemctl restart jenkins
```

❖ Opening the Firewall:

```
> ufw allow 8080
> ufw status
> ufw allow OpenSSH
> ufw enable
```

❖ Setting Up Jenkins:

To set up your installation, visit Jenkins on its default port, 8080, using your server domain name or IP address: http://your_server_ip_or_domain:8080

➤ use the `cat` command to display the password:

```
cat /var/lib/jenkins/secrets/initialAdminPas
```

- Copy the 32-character alphanumeric password from the terminal and paste it into the Administrator password field, then click Continue.
- The next screen presents the option of installing suggested plugins or selecting

Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

Install suggested plugins

Install plugins the Jenkins community finds most useful.

Select plugins to install

Select and install plugins most suitable for your needs.

❖ We'll click the **Install suggested plugins** option, which will immediately begin the installation.

Getting Started

✓ Folders	✓ OWASP Markup Formatter	✓ Build Timeout	✓ Credentials Binding	<pre> ** Pipeline: Milestone Step ** JavaScript GUI Lib: jQuery bundles (jQuery and jQuery UI) ** Jackson 2 API ** JavaScript GUI Lib: ACE Editor bundle ** Pipeline: SCM Step ** Pipeline: Groovy ** Pipeline: Input Step ** Pipeline: Stage Step ** Pipeline: Job ** Pipeline: Graph Analysis ** Pipeline: REST API ** JavaScript GUI Lib: Handlebars bundle ** JavaScript GUI Lib: Moment.js bundle Pipeline: Stage View ** Pipeline: Build Step ** Pipeline: Model API ** Pipeline: Declarative Extension Points API ** Apache HttpComponents Client 4.x API ** JSch dependency </pre>
✓ Timestampers	✓ Workspace Cleanup	✓ Ant	✓ Gradle	
🔄 Pipeline	🔄 GitHub Branch Source	🔄 Pipeline: GitHub Groovy Libraries	✓ Pipeline: Stage View	
🔄 Git	🔄 Subversion	🔄 SSH Slaves	🔄 Matrix Authorization Strategy	
🔄 PAM Authentication	🔄 LDAP	🔄 Email Extension	🔄 Mailer	

❖ create the user:

Getting Started

Create First Admin User

Username:

Password:

Confirm password:

Full name:

E-mail address:

Jenkins 2.121.1

[Continue as admin](#)

Save and Continue

- ❖ You will see an **Instance Configuration** page that will ask you to confirm the preferred URL for this instance. Confirm either the domain name for your server or your server's IP address:

Instance Configuration

Jenkins URL:

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the `BUILD_URL` environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

- ❖ After confirming the appropriate information, click **Save and Finish**. You will see a confirmation screen confirming that **“Jenkins is Ready”**

Jenkins is ready!

Your Jenkins setup is complete.

Start using Jenkins



❖ Launch ubuntu instance for Docker:

- Install of Docker:
- Apt update
- apt install docker.io -y
- usermod -aG docker ubuntu

❖ Create CI/CD pipeline Job:

Enter an item name

» Required field

**Freestyle project**

This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

**Pipeline**

Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

**Multi-configuration project**

Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

OK

❖ Write a pipeline Script:

Step1: Git Clone:

The screenshot shows the Jenkins configuration page for a job named 'java-web-app-docker'. The 'Advanced Project Options' tab is selected. Under the 'Pipeline' section, the 'Definition' is set to 'Pipeline script'. The 'Script' field contains the following YAML code:

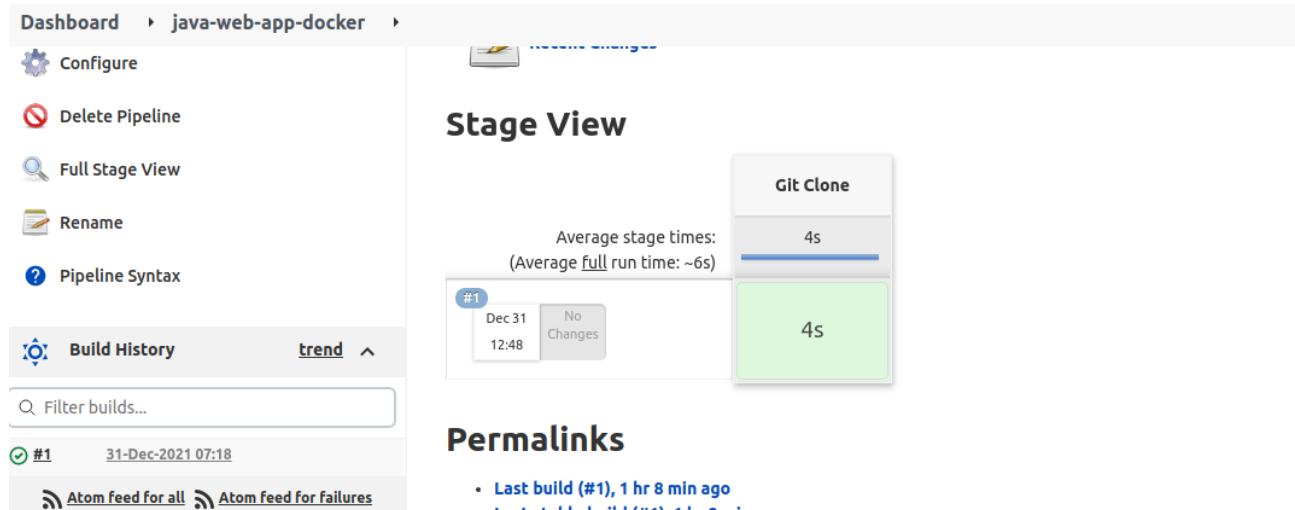
```
1 node{
2
3   stage("Git Clone"){
4     git url: 'https://github.com/muni77-sh/java-web-app-docker.git',branch: 'master'
5   }
6 }
7
```

At the bottom of the configuration area, there are 'Save' and 'Apply' buttons.

→ Save it

→ trigger the Build

→ Build is Success



Step2: Maven Package:

Global Tool Configuration:

→ Click the "Manage Jenkins"

→ Click the "Global Tool Configuration"

→ Click on Maven Installation and set the Maven path.

Maven installations

[Add Maven](#)

Maven	Name
	<input type="text" value="Maven"/>

☒ Install automatically [?](#)

Install from Apache

Version

▾

[Add Installer ▾](#)

[Delete Installer](#)

[Save](#) [Apply](#)

→ Finally click on save and your Maven configuration is done.

Then, goto “pipeline job” → Click on “Configure” →

Script ?

```
1 node{
2
3   stage("Git Clone"){
4       git url: 'https://github.com/muni77-sh/java-web-app-docker.git',branch: 'master'
5   }
6
7   stage('Maven Clean Package'){
8       def mavenHome= tool name: "Maven",type: "maven"
9       sh "${mavenHome}/bin/mvn clean package"
10  }
11
12 }
13
```

☒ Use Groovy Sandbox ?

Pipeline Syntax

Save

Apply

Save it → Trigger the build → Build is Success

Dashboard
>
java-web-app-docker
>

Changes

Build Now

Configure

Delete Pipeline

Full Stage View

Rename

Pipeline Syntax

Build History

trend ^

Filter builds...

#2

31-Dec-2021 08:51

#1

31-Dec-2021 07:18

Recent Changes

Stage View

Average stage times:

(Average full run time: ~12s)

#2

Dec 31 14:21

No Changes

#1

Dec 31 12:48

No Changes

Git Clone	Maven Clean Package
2s	18s
495ms	18s
4s	

Go to build job to verify to build is success → Click on “Console”

Dashboard
>
java-web-app-docker
>
#2

```

Progress (2/): 184 kB | 60 kB
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-interpolation/1.15/plexus-interpolation-1.15.jar (60 kB at 484 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-archiver/2.1/plexus-archiver-2.1.jar (184 kB at 1.4 MB/s)
[INFO] Packaging webapp
[INFO] Assembling webapp [java-web-app] in [/var/lib/jenkins/workspace/java-web-app-docker/target/java-web-app-1.0]
[INFO] Processing war project
[INFO] Copying webapp resources [/var/lib/jenkins/workspace/java-web-app-docker/src/main/webapp]
[INFO] Webapp assembled in [124 msecs]
[INFO] Building war: /var/lib/jenkins/workspace/java-web-app-docker/target/java-web-app-1.0.war
[INFO] WEB-INF/web.xml already added, skipping
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 12.485 s
[INFO] Finished at: 2021-12-31T08:51:55Z
[INFO] -----
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS

```

Step 3: Create a Docker Image: (in Pipeline Job)

Dashboard ▸ java-web-app-docker ▸

General Build Triggers Advanced Project Options **Pipeline**

Definition

Pipeline script ▾

Script ?

```

1 node{
2
3     def buildNumber = BUILD_Number
4     stage("Git Clone"){
5
6         git url: 'https://github.com/muni77-sh/java-web-app-docker.git',branch: 'master'
7     }
8
9     stage('Maven Clean Package'){
10         def mavenHome= tool name: "Maven",type: "maven"
11         sh "${mavenHome}/bin/mvn clean package"
12     }
13
14     stage("Build Docker Image"){
15         sh "docker build -t bhaskar77/java-jeb-app-docker:${buildNumber} ."
16     }
17 }
18

```

☒ Use Groovy Sandbox ?

Save Apply

→ save → Build is Success

Dashboard ▸ java-web-app-docker ▸

Build Now
 Configure
 Delete Pipeline
 Full Stage View
 Rename
 Pipeline Syntax

Build History **trend** ^

#3 31-Dec-2021 09:19

#2 31-Dec-2021 08:51

#1 31-Dec-2021 07:18

Atom feed for all
 Atom feed for failures

Recent Changes

Stage View

	Git Clone	Maven Clean Package	Build Docker Image
Average stage times: (Average full run time: ~16s)	1s	12s	17s
#3 Dec 31 14:49 No Changes	334ms	6s	17s
#2 Dec 31 14:21 No Changes	495ms	18s	
#1 Dec 31 12:48 No Changes	4s		

❖ Check the images are created in Jenkins server:

```
ubuntu@ip-172-31-2-207:~$ sudo docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
bhaskar77/java-jeb-app-docker  3                  b2a3d3ef0640       6 minutes ago      499MB
tomcat               8.0.20-jre8        e88a065848be       6 years ago        493MB
ubuntu@ip-172-31-2-207:~$
```

Step4: To Login In Docker Hub And Push :

Click on, Pipeline Syntax → To create Credentials (Secret) → Sample Step →

“withCredentials: Bind credentials to variables” → Create Secret Text → then,

“Generate pipeline script”

The screenshot shows the Jenkins Pipeline Syntax configuration page for a job named 'java-web-app-docker'. The breadcrumb navigation at the top reads 'Dashboard > java-web-app-docker > Pipeline Syntax'. The main section is titled 'Bindings' and contains the following elements:

- A 'Secret text' binding type is selected, indicated by a grid icon.
- A 'Variable' binding type is also shown with a question mark icon.
- A text input field contains the value 'Docker_Hub_pwd'.
- A 'Credentials' section with a question mark icon, containing a dropdown menu with 'Docker_Hub_password' and an 'Add' button with a plus icon.
- A red 'Delete' button is located at the bottom right of the bindings section.
- An 'Add' button with a dropdown arrow is at the bottom left of the bindings section.
- A blue 'Generate Pipeline Script' button is located below the bindings section.

Below the 'Generate Pipeline Script' button, the generated pipeline script is displayed:

```
withCredentials([string(credentialsId: 'Docker_Hub_pwd', variable: 'Docker_Hub_pwd')]) {
    // some block
}
```

Again, goto “pipeline job” → Click on “Configure” →






Script ?

```
1 node{
2
3     def buildNumber = BUILD_Number
4     stage("Git Clone"){
5
6         git url: 'https://github.com/muni77-sh/java-web-app-docker.git',branch: 'master'
7     }
8
9     stage('Maven Clean Package'){
10         def mavenHome= tool name: "Maven",type: "maven"
11         sh "${mavenHome}/bin/mvn clean package"
12     }
13
14     stage("Build Docker Image"){
15         sh "docker build -t bhaskar77/java-web-app-docker:${buildNumber} ."
16     }
17
18     stage("Docker Login And Push"){
19         withCredentials([string(credentialsId: 'DockerHubPwd', variable: 'DockerHubPwd')])
20         sh "docker login -u bhaskar77 -p ${DockerHubPwd}"
21     }
22     sh "docker push bhaskar77/java-web-app-docker:${buildNumber}"
23 }
24
25
26
```

Save

Apply

→ save it → trigger the build → build is success

 Changes Build Now Configure Delete Pipeline Full Stage View Rename Pipeline Syntax Build History

trend ^

Q Filter builds...

✓ #12 Jan 1, 2022 2:03 PM

✓ #11 Dec 31, 2021 12:58 PM

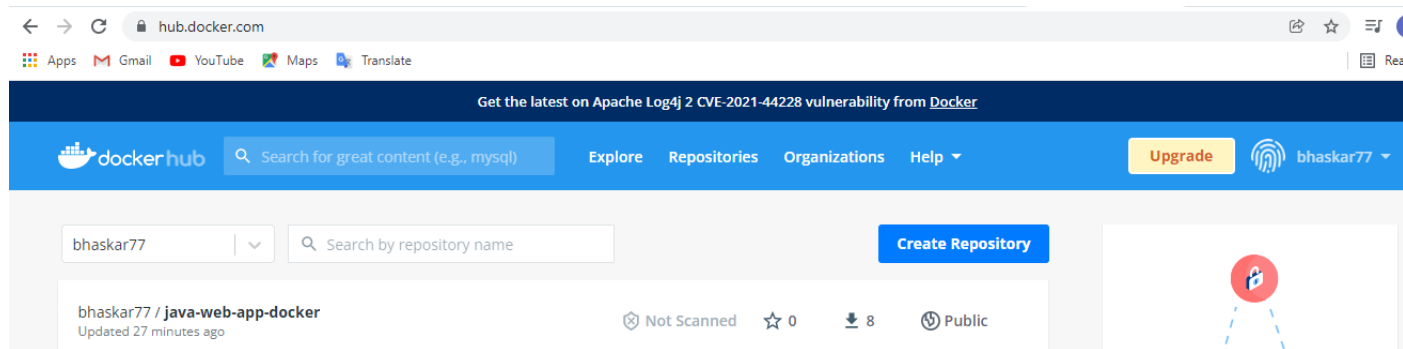


Recent Changes

Stage View

		Git Clone	Maven Clean Package	Build Docker Image	Docker Login And Push
Average stage times: (Average full run time: ~18s)		425ms	7s	2s	1s
#12	Jan 01 19:33 No Changes	997ms	8s	1s	6s
#11	Dec 31 18:28 No Changes	301ms	6s	810ms	4s

→ check images have been pushed into Docker Hub.



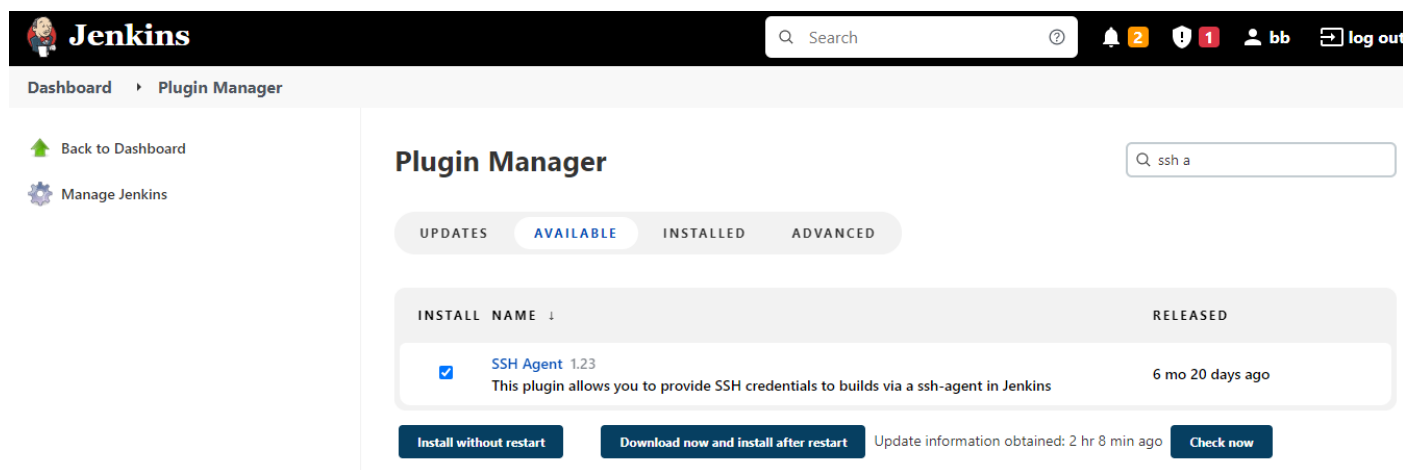
→ check if CLI docker images are created.

```
ubuntu@ip-172-31-2-207:~$ sudo docker images
REPOSITORY          TAG          IMAGE ID       CREATED        SIZE
bhaskar77/java-web-app-docker  12          c09794d40cdf  2 hours ago   499MB
tomcat               8.0.20-jre8  e88a065848be  6 years ago   493MB
ubuntu@ip-172-31-2-207:~$
```

Step5: To Connect to Deployment server From Jenkin_Server

→ Jenkins Dashboard → Manage Jenkins → Manage Plugin →

Click on “Available” → type “SSH Agent” install without restart.



Click on, Pipeline Syntax → Sample Step →

“sshagent:SSH Agent” → To connect from One to another server → then,

“Generate pipeline script”

The screenshot shows the Jenkins Pipeline Syntax generator interface. The breadcrumb navigation is **Dashboard** > **java-web-app-docker** > **Pipeline Syntax**. On the left, there is a sidebar with links: [Declarative Online Documentation](#), [Steps Reference](#), [Global Variables Reference](#), [Online Documentation](#), [Examples Reference](#), and [IntelliJ IDEA GDSL](#). The main content area has a heading **Steps** and a subheading **Sample Step**. A dropdown menu shows **sshagent: SSH Agent**. Below this, there is a section for **sshagent** with a dropdown set to **ubuntu** and an **Add** button. There is also a checkbox for **Ignore missing credentials**. At the bottom, there is a **Generate Pipeline Script** button. Below the button, the generated script is shown:

```
sshagent(['Docker_Dev_Server_SSH']) {  
    // some block  
}
```

→ Go to back pipeline job → Click on “Configure” write the script for → Deploy Application

Docker Deployment Server:

The screenshot shows the Jenkins Pipeline configuration page. The breadcrumb navigation is **Dashboard** > **java-Web-App-Docker**. The **Pipeline** tab is selected. The pipeline script is displayed in a text area with line numbers 24 to 39. The script is as follows:

```
24 sh "docker push bhaskar77/java-web-app-docker:${buildNumber}"  
25 }  
26 }  
27 }  
28 }  
29 }  
30 stage("Deploy Application As Docker In Docker Deployment Server"){  
31 }  
32 }  
33 sshagent(['Docker_Dev_Server_SSH']) {  
34 sh "ssh -o StrictHostKeyChecking=no ubuntu@172.31.17.31 docker rm -f javawebappcontainer || true"  
35 sh "ssh -o StrictHostKeyChecking=no ubuntu@172.31.17.31 docker run -d -p 8080:8080 --name javawebappcontainer bhaskar77/java-web-app-docker:${buildNumber}"  
36 }  
37 }  
38 }  
39 }
```



```

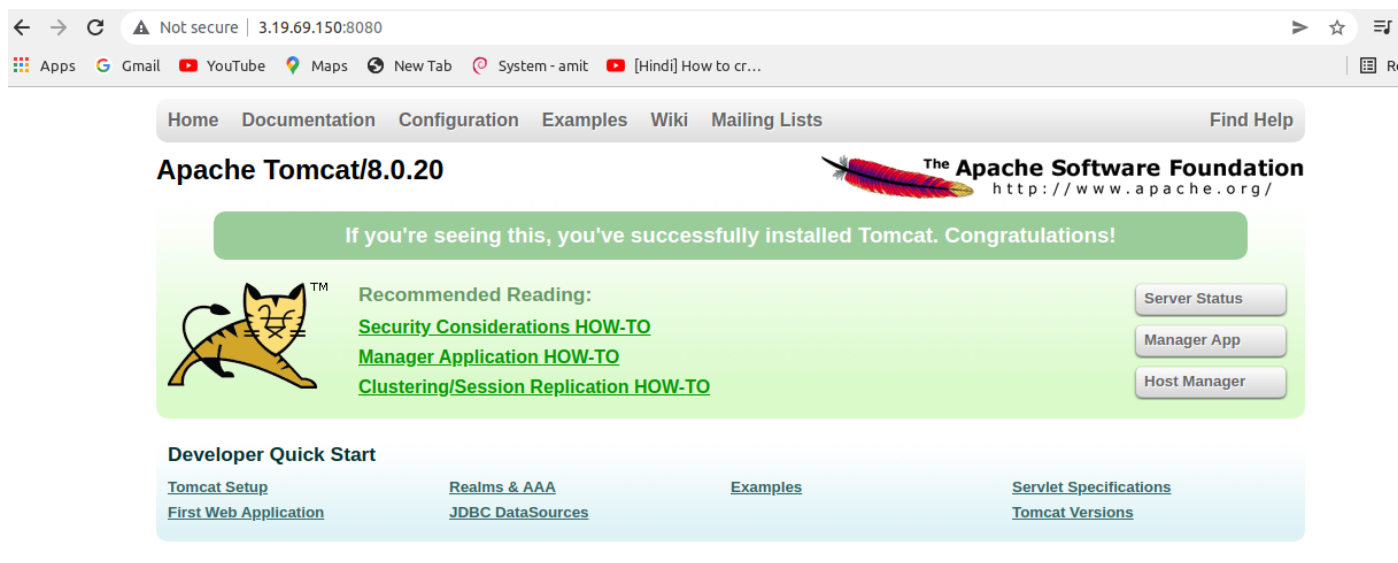
stage("Deploy Application As Docker In Docker Deployment Server"){

    sshagent(['Docker_Dev2_Server_SSH']) {
        sh "ssh -o StrictHostKeyChecking=no ubuntu@172.31.2.250 docker rm -f javawebappcontainer || true"
        sh "ssh -o StrictHostKeyChecking=no ubuntu@172.31.2.250 docker run -d -p 8080:8080 --name javawebappcontainer bhaskar77/java-web-app-docker:${buildNumber}"
    }
}

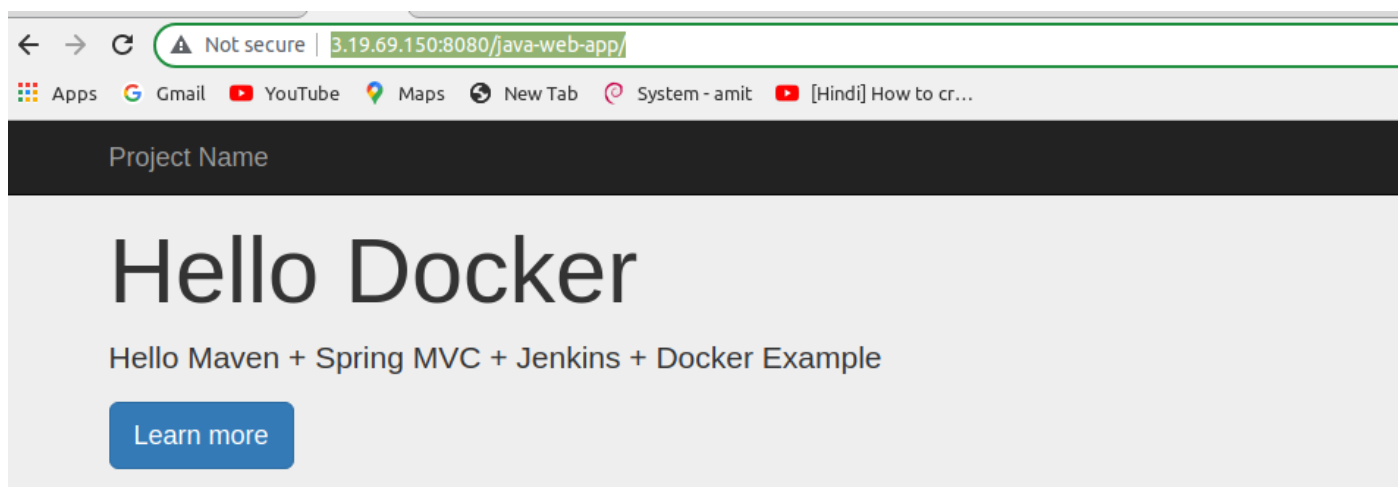
```

→ we are trying access the Application→ from “ Deployment-server-docker”

With public-IP “http://3.19.69.150:8080/”



To check my application context is “<http://3.19.69.150:8080/java-web-app/>”



Again, make changes in code and again rebuild the your “jenkins job”

Project Name

Hello Docker

Hello Maven + Spring MVC + Jenkins + Docker Example

[Learn more](#)

Welcome to bhasker home page

Devops Technology

