# CI/CD Deployment for Springboot Application

Prototype of the Application

Name : Shaik Akhila

GitHub : https://github.com/akhila31shaik/JSD_phase-5_project

This documentation will guide you through the steps involved in setting up a CI/CD pipeline for a Spring Boot application using Jenkins as the CI/CD tool and AWS EC2 instance as the hosting server. The application code will be fetched from a GitHub repository, and the deployment will be automated using Jenkins.

## Sprint Planning

The Implementation is done in Eight sprints which are mentioned below

Prerequisites:

A GitHub account to store the source code.
A Jenkins server with admin access.
An AWS account to create an EC2 instance.
Basic knowledge of Spring Boot, AWS EC2, and Jenkins.

Sprint involved:

### Sprint 1: Clone the GitHub repository:

Clone the repository containing the source code of the Spring Boot application to the local system. This will be used by Jenkins to build the application and create a deployment package.

### Sprint 2: Install necessary plugins:

Install the following plugins in the Jenkins server:

Maven Integration Plugin
Git plugin
These plugins will be used to build the application and fetch the source code from the GitHub repository.

### Sprint 3: Create a Jenkins job:

Create a new Jenkins job by navigating to Jenkins Dashboard > New Item > Freestyle project.

In the Source Code Management section, select Git and provide the URL of the GitHub repository where the code is stored.

In the Build section, select Invoke top-level Maven targets and provide the necessary Maven goals to build the application. For example, clean install.

**Sprint 4: Configure AWS EC2 instance:**

Create an EC2 instance in the AWS account and configure the security group to allow HTTP traffic on port 8080.

Install Java and Tomcat on the EC2 instance to run the Spring Boot application.

**Sprint 5: Install AWS CLI:**

Install the AWS CLI on the Jenkins server to automate the deployment process.

**Sprint 6: Configure AWS credentials:**

Add the AWS access key and secret key to the Jenkins server using the AWS CLI.

**Sprint 7: Create a deployment job:**

Create a new Jenkins job to deploy the Spring Boot application to the EC2 instance. In this job, configure the following steps:

Fetch the latest deployment package from the Jenkins workspace.
Copy the deployment package to the EC2 instance using the AWS CLI.
Start the Tomcat server on the EC2 instance to run the application.

**Sprint 8: Test the deployment:**

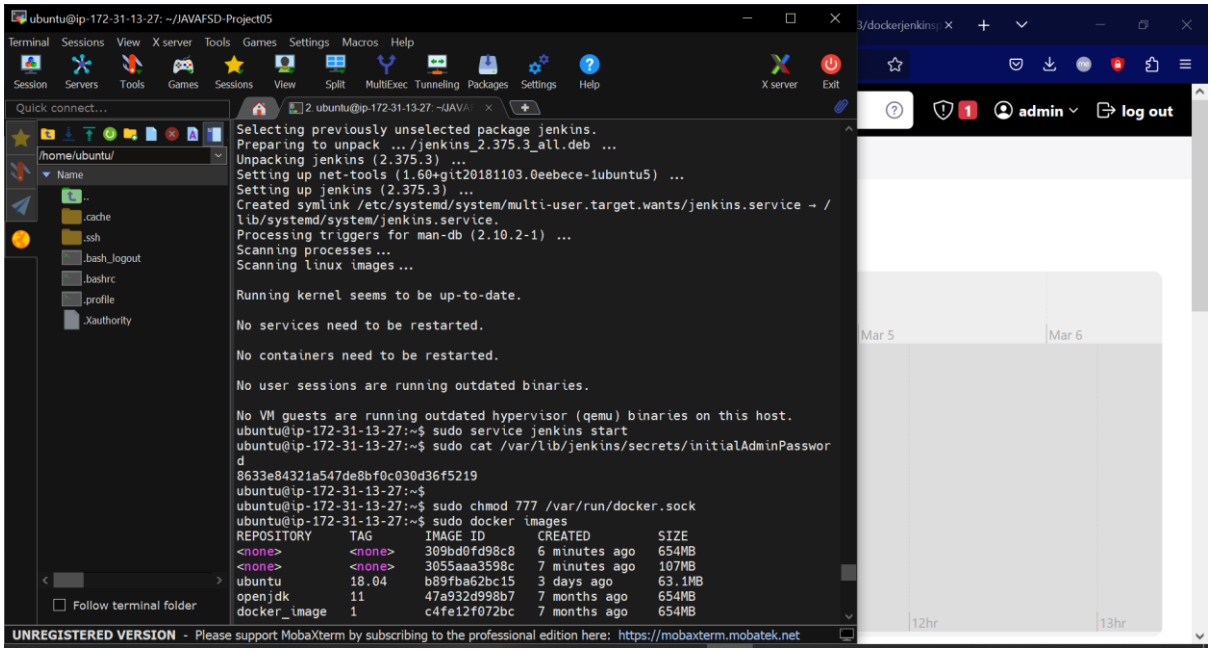
Run the deployment job and check if the application is deployed successfully on the EC2 instance. Access the application using the public IP address of the EC2 instance.

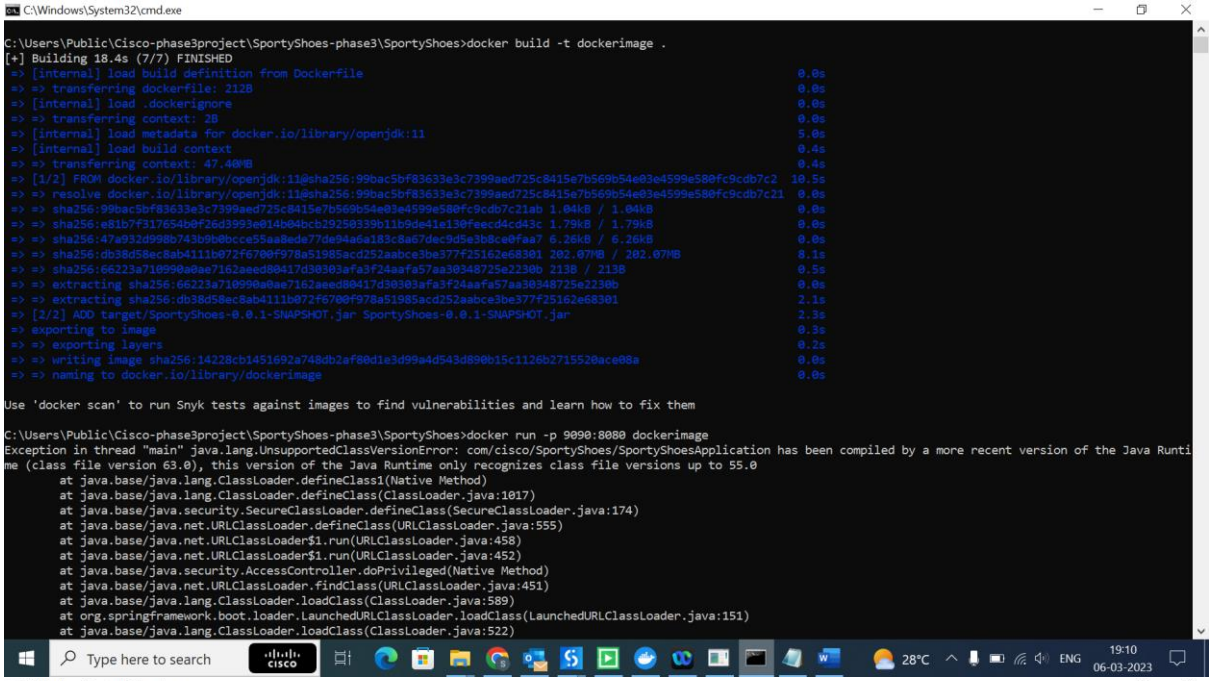
**Conclusion:**

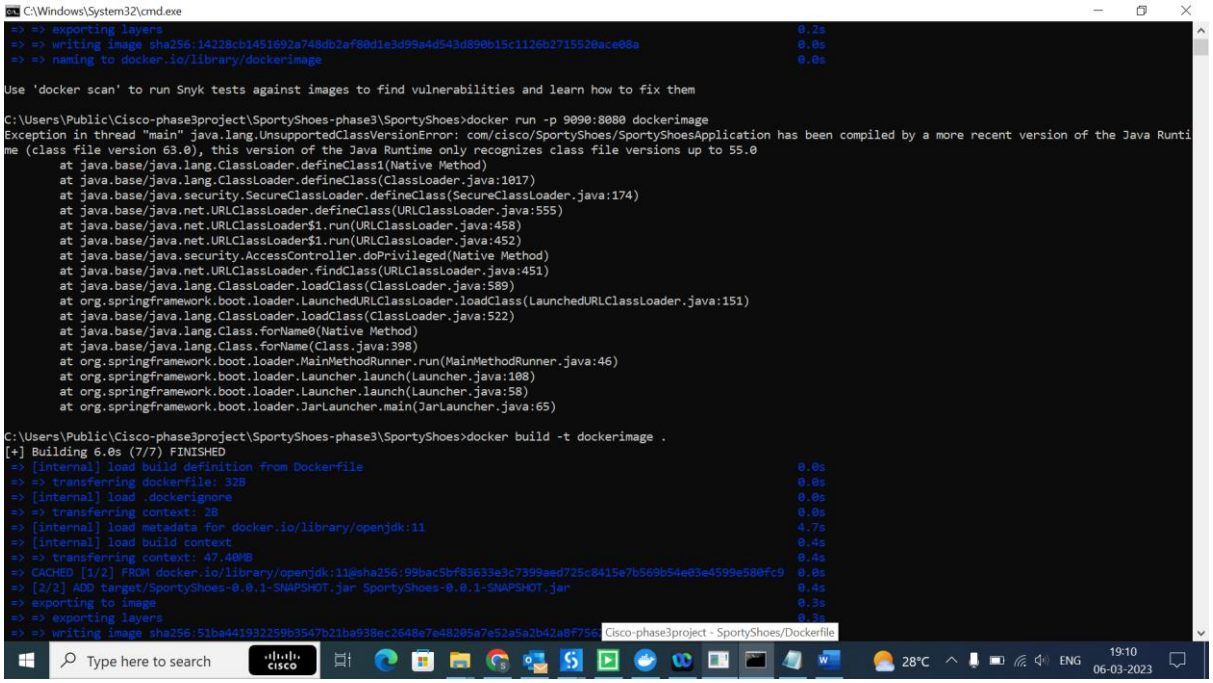
By following the above steps, you can automate the integration and deployment of a Spring Boot application using Jenkins and AWS EC2 instance. This will reduce the

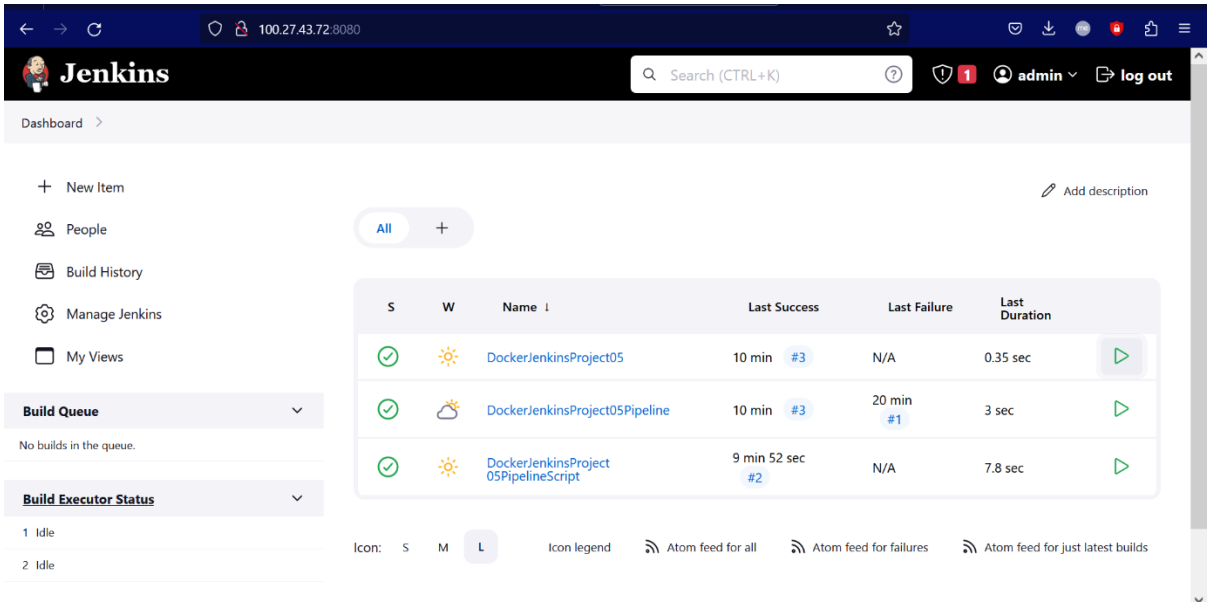
manual effort required for deployment and increase the efficiency of the development process.

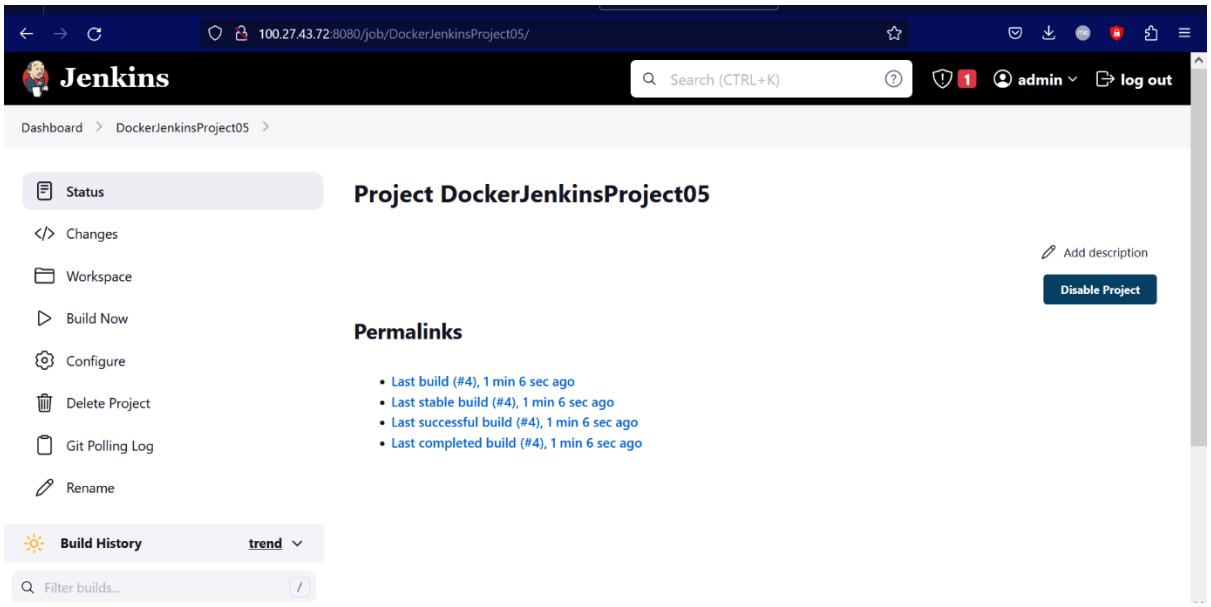
# Screenshot

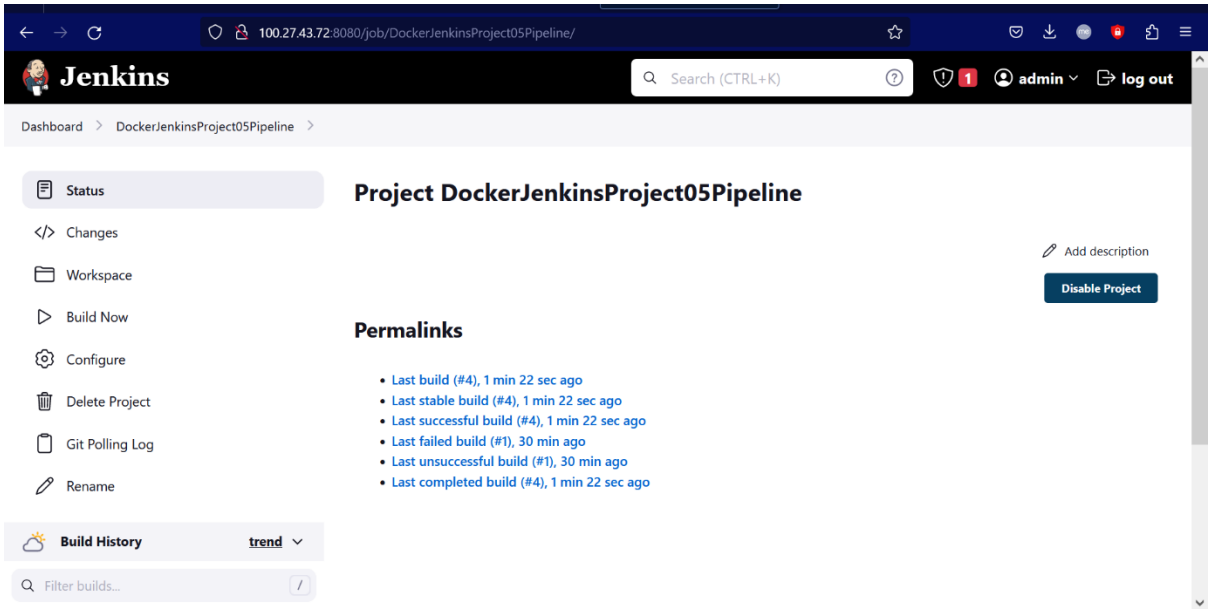


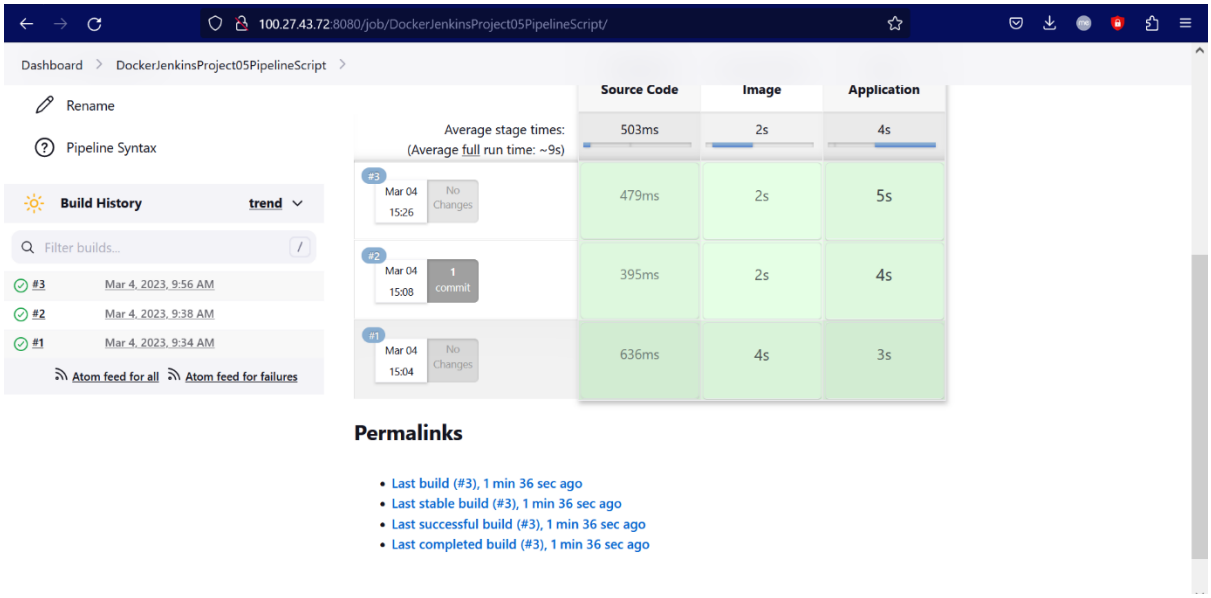

C:\Windows\System32\cmd.exe
C:\Users\Public\Cisco-phase3project\SportyShoes-phase3\SportyShoes>docker build -t dockerimage .
[+] Building 18.4s (7/7) FINISHED
=> [internal] load build definition from Dockerfile 0.0s
=> => transferring dockerfile: 212B 0.0s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [internal] load metadata for docker.io/library/openjdk:11 5.0s
=> [internal] load build context 0.4s
=> => transferring context: 47.40MB 0.4s
=> [1/2] FROM docker.io/library/openjdk:11@sha256:99bac5bf83633e3c7399aed725c8415e7b569b54e03e4599e580fc9cdb7c2 10.5s
=> => resolve docker.io/library/openjdk:11@sha256:99bac5bf83633e3c7399aed725c8415e7b569b54e03e4599e580fc9cdb7c21 0.0s
=> => sha256:99bac5bf83633e3c7399aed725c8415e7b569b54e03e4599e580fc9cdb7c21ab 1.04kB / 1.04kB 0.0s
=> => sha256:e81b7f317654b0f26d3993e014b04bcb20250339b11b9de41e130feecd4cd43c 1.79kB / 1.79kB 0.0s
=> => sha256:47a932d998b743b9b0bcce55aa8ede77de94a6a183c8a67dec9d5e3b8ce0faa7 6.26kB / 6.26kB 0.0s
=> => sha256:db38d58ec8ab4111b072f6700f978a51985acd252aabce3be377f25162e68301 202.07MB / 202.07MB 8.1s
=> => sha256:66223a710990a0ae7162aeed80417d30303afa3f24aafa57aa30348725e2230b 213B / 213B 0.5s
=> => extracting sha256:66223a710990a0ae7162aeed80417d30303afa3f24aafa57aa30348725e2230b 0.0s
=> => extracting sha256:db38d58ec8ab4111b072f6700f978a51985acd252aabce3be377f25162e68301 2.1s
=> [2/2] ADD target/SportyShoes-0.0.1-SNAPSHOT.jar SportyShoes-0.0.1-SNAPSHOT.jar 2.3s
=> exporting to image 0.3s
=> => exporting layers 0.2s
=> => writing image sha256:14228cb1451692a748db2af80d1e3d99a4d543d890b15c1126b2715520ace08a 0.0s
=> => naming to docker.io/library/dockerimage 0.0s
Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
C:\Users\Public\Cisco-phase3project\SportyShoes-phase3\SportyShoes>docker run -p 9090:8080 dockerimage
Exception in thread "main" java.lang.UnsupportedClassVersionError: com/cisco/SportyShoes/SportyShoesApplication has been compiled by a more recent version of the Java Runtime (class file version 63.0), this version of the Java Runtime only recognizes class file versions up to 55.0
at java.base/java.lang.ClassLoader.defineClass1(Native Method)
at java.base/java.lang.ClassLoader.defineClass(ClassLoader.java:1017)
at java.base/java.security.SecureClassLoader.defineClass(SecureClassLoader.java:174)
at java.base/java.net.URLClassLoader.defineClass(URLClassLoader.java:555)
at java.base/java.net.URLClassLoader$1.run(URLClassLoader.java:458)
at java.base/java.net.URLClassLoader$1.run(URLClassLoader.java:452)
at java.base/java.security.AccessController.doPrivileged(Native Method)
at java.base/java.net.URLClassLoader.findClass(URLClassLoader.java:451)
at java.base/java.lang.ClassLoader.loadClass(ClassLoader.java:589)
at org.springframework.boot.loader.LaunchedURLClassLoader.loadClass(LaunchedURLClassLoader.java:151)
at java.base/java.lang.ClassLoader.loadClass(ClassLoader.java:522)
Type here to search
28°C
ENG
19:10
06-03-2023

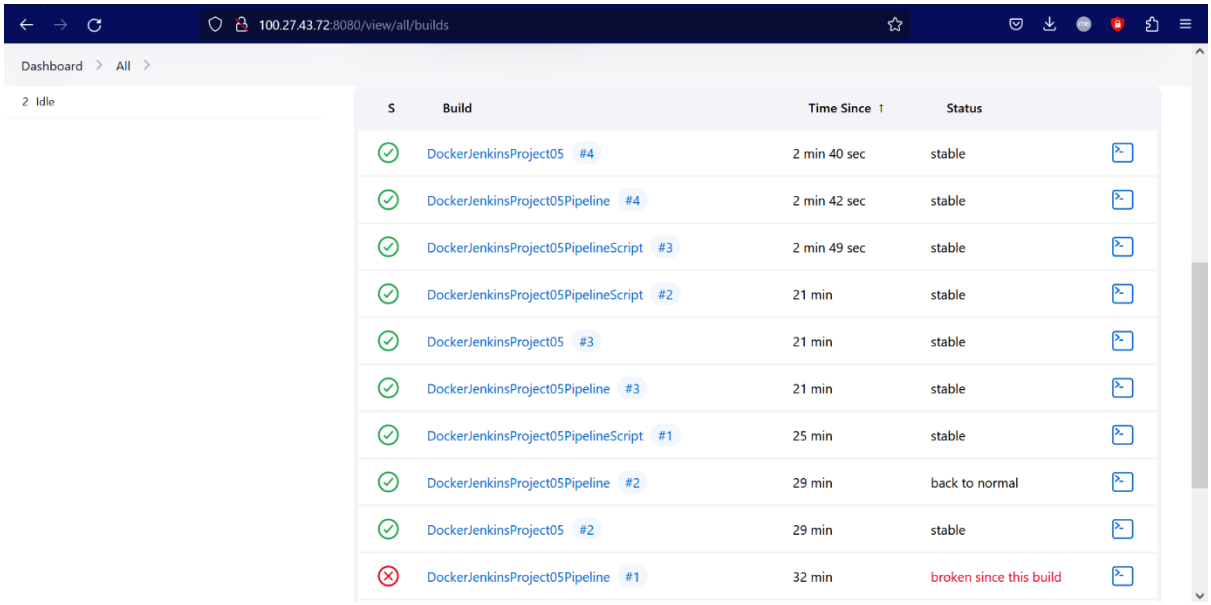

C:\Windows\System32\cmd.exe
=> => exporting layers 0.2s
=> => writing image sha256:14228cb1451692a748db2af80d1e3d99a4d543d890b15c1126b2715520ace08a 0.0s
=> => naming to docker.io/library/dockerimage 0.0s
Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
C:\Users\Public\Cisco-phase3project\SportyShoes-phase3\SportyShoes>docker run -p 9090:8080 dockerimage
Exception in thread "main" java.lang.UnsupportedClassVersionError: com/cisco/SportyShoes/SportyShoesApplication has been compiled by a more recent version of the Java Runtime (class file version 63.0), this version of the Java Runtime only recognizes class file versions up to 55.0
at java.base/java.lang.ClassLoader.defineClass1(Native Method)
at java.base/java.lang.ClassLoader.defineClass(ClassLoader.java:1017)
at java.base/java.security.SecureClassLoader.defineClass(SecureClassLoader.java:174)
at java.base/java.net.URLClassLoader.defineClass(URLClassLoader.java:555)
at java.base/java.net.URLClassLoader$1.run(URLClassLoader.java:458)
at java.base/java.net.URLClassLoader$1.run(URLClassLoader.java:452)
at java.base/java.security.AccessController.doPrivileged(Native Method)
at java.base/java.net.URLClassLoader.findClass(URLClassLoader.java:451)
at java.base/java.lang.ClassLoader.loadClass(ClassLoader.java:589)
at org.springframework.boot.loader.LaunchedURLClassLoader.loadClass(LaunchedURLClassLoader.java:151)
at java.base/java.lang.ClassLoader.loadClass(ClassLoader.java:522)
at java.base/java.lang.Class.forName0(Native Method)
at java.base/java.lang.Class.forName(Class.java:398)
at org.springframework.boot.loader.MainMethodRunner.run(MainMethodRunner.java:46)
at org.springframework.boot.loader.Launcher.launch(Launcher.java:108)
at org.springframework.boot.loader.Launcher.launch(Launcher.java:58)
at org.springframework.boot.loader.JarLauncher.main(JarLauncher.java:65)
C:\Users\Public\Cisco-phase3project\SportyShoes-phase3\SportyShoes>docker build -t dockerimage .
[+] Building 6.0s (7/7) FINISHED
=> [internal] load build definition from Dockerfile 0.0s
=> => transferring dockerfile: 32B 0.0s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [internal] load metadata for docker.io/library/openjdk:11 4.7s
=> [internal] load build context 0.4s
=> => transferring context: 47.40MB 0.4s
=> CACHED [1/2] FROM docker.io/library/openjdk:11@sha256:99bac5bf83633e3c7399aed725c8415e7b569b54e03e4599e580fc9 0.0s
=> [2/2] ADD target/SportyShoes-0.0.1-SNAPSHOT.jar SportyShoes-0.0.1-SNAPSHOT.jar 0.4s
=> exporting to image 0.3s
=> => exporting layers 0.3s
=> => writing image sha256:51ba441932259b3547b21ba938ec2648e7e48205a7e52a5a2b42a8f756
Cisco-phase3project - SportyShoes/Dockerfile
Type here to search
28°C
ENG
19:10
06-03-2023

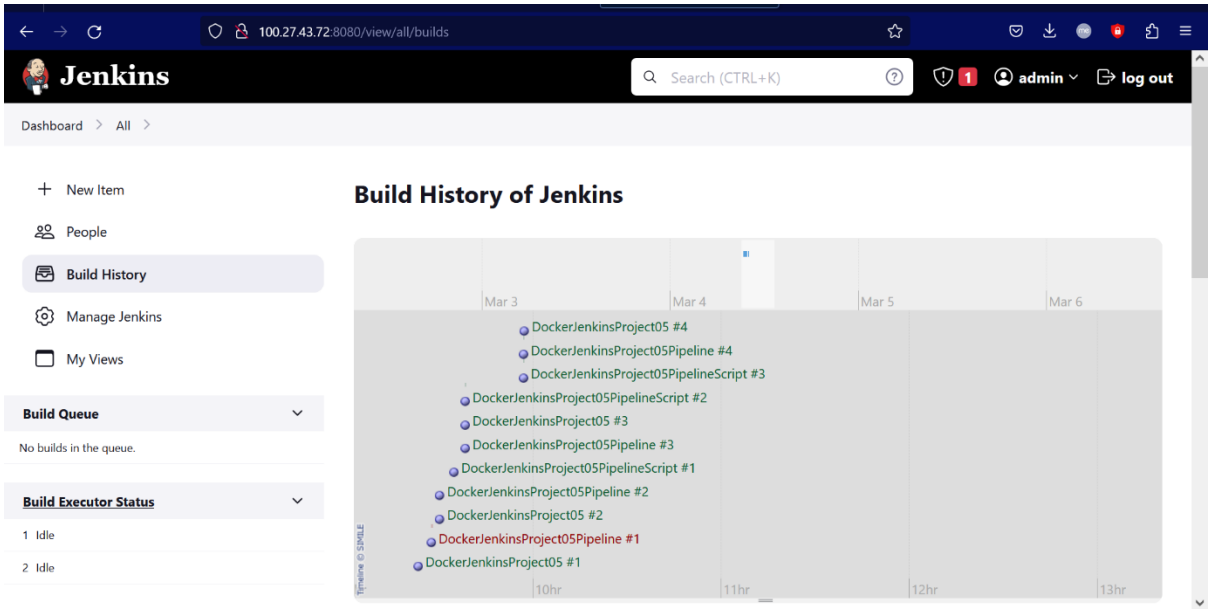

100.27.43.72:8080
Jenkins
Search (CTRL+K)
admin
log out
Dashboard
New Item
People
Build History
Manage Jenkins
My Views
Build Queue
No builds in the queue.
Build Executor Status
1 Idle
2 Idle
Add description
All
S
W
Name
Last Success
Last Failure
Last Duration
DockerJenkinsProject05
10 min #3
N/A
0.35 sec
DockerJenkinsProject05Pipeline
10 min #3
20 min #1
3 sec
DockerJenkinsProject 05PipelineScript
9 min 52 sec #2
N/A
7.8 sec
Icon: S M L
Icon legend
Atom feed for all
Atom feed for failures
Atom feed for just latest builds

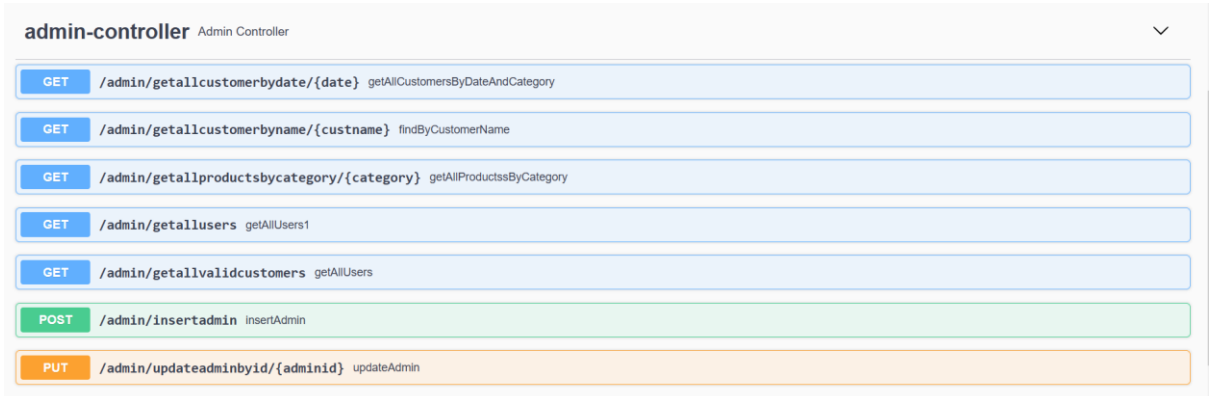

100.27.43.72:8080/job/DockerJenkinsProject05/
Jenkins
Search (CTRL+K)
admin
log out
Dashboard
DockerJenkinsProject05
Status
Changes
Workspace
Build Now
Configure
Delete Project
Git Polling Log
Rename
Build History
trend
Filter builds...
Project DockerJenkinsProject05
Add description
Disable Project
Permalinks
Last build (#4), 1 min 6 sec ago
Last stable build (#4), 1 min 6 sec ago
Last successful build (#4), 1 min 6 sec ago
Last completed build (#4), 1 min 6 sec ago


100.27.43.72:8080/job/DockerJenkinsProject05Pipeline/
Jenkins
Search (CTRL+K)
admin
log out
Dashboard
DockerJenkinsProject05Pipeline
Status
Changes
Workspace
Build Now
Configure
Delete Project
Git Polling Log
Rename
Build History
trend
Filter builds...
Project DockerJenkinsProject05Pipeline
Add description
Disable Project
Permalinks
Last build (#4), 1 min 22 sec ago
Last stable build (#4), 1 min 22 sec ago
Last successful build (#4), 1 min 22 sec ago
Last failed build (#1), 30 min ago
Last unsuccessful build (#1), 30 min ago
Last completed build (#4), 1 min 22 sec ago


100.27.43.72:8080/job/DockerJenkinsProject05PipelineScript/
Dashboard
DockerJenkinsProject05PipelineScript
Rename
Pipeline Syntax
Build History
trend
Filter builds...
#3 Mar 4, 2023, 9:56 AM
#2 Mar 4, 2023, 9:38 AM
#1 Mar 4, 2023, 9:34 AM
Atom feed for all
Atom feed for failures
Source Code
Image
Application
Average stage times:
(Average full run time: ~9s)
503ms
2s
4s
#3 Mar 04 15:26 No Changes
479ms
2s
5s
#2 Mar 04 15:08 1 commit
395ms
2s
4s
#1 Mar 04 15:04 No Changes
636ms
4s
3s
Permalinks
Last build (#3), 1 min 36 sec ago
Last stable build (#3), 1 min 36 sec ago
Last successful build (#3), 1 min 36 sec ago
Last completed build (#3), 1 min 36 sec ago


100.27.43.72:8080/view/all/builds
Dashboard
All
2 Idle
S
Build
Time Since
Status
DockerJenkinsProject05 #4 2 min 40 sec stable
DockerJenkinsProject05Pipeline #4 2 min 42 sec stable
DockerJenkinsProject05PipelineScript #3 2 min 49 sec stable
DockerJenkinsProject05PipelineScript #2 21 min stable
DockerJenkinsProject05 #3 21 min stable
DockerJenkinsProject05Pipeline #3 21 min stable
DockerJenkinsProject05PipelineScript #1 25 min stable
DockerJenkinsProject05Pipeline #2 29 min back to normal
DockerJenkinsProject05 #2 29 min stable
DockerJenkinsProject05Pipeline #1 32 min broken since this build


100.27.43.72:8080/view/all/builds
Jenkins
Search (CTRL+K)
admin
log out
Dashboard
All
New Item
People
Build History
Manage Jenkins
My Views
Build Queue
No builds in the queue.
Build Executor Status
1 Idle
2 Idle
Build History of Jenkins
Mar 3
Mar 4
Mar 5
Mar 6
DockerJenkinsProject05 #4
DockerJenkinsProject05Pipeline #4
DockerJenkinsProject05PipelineScript #3
DockerJenkinsProject05PipelineScript #2
DockerJenkinsProject05 #3
DockerJenkinsProject05Pipeline #3
DockerJenkinsProject05PipelineScript #1
DockerJenkinsProject05Pipeline #2
DockerJenkinsProject05 #2
DockerJenkinsProject05Pipeline #1
DockerJenkinsProject05 #1
10hr
11hr
12hr
13hr


admin-controller Admin Controller
GET /admin/getallcustomerbydate/{date} getAllCustomersByDateAndCategory
GET /admin/getallcustomerbyname/{custname} findByCustomerName
GET /admin/getallproductsbycategory/{category} getAllProductssByCategory
GET /admin/getallusers getAllUsers1
GET /admin/getallvalidcustomers getAllUsers
POST /admin/insertadmin insertAdmin
PUT /admin/updateadminbyid/{adminid} updateAdmin

## customer-controller Customer Controller

- PUT `/customer/addcustomerproducts/{custid}/{prodid}` updateProductInCustomer
- DELETE `/customer/deletecustomer/{userid}` deleteCustomer
- GET `/customer/getallcustomers` getAllCustomers
- GET `/customer/getcustomer/{userid}` getCustomer
- POST `/customer/insertcustomer/{id}` insertCustomer
- PUT `/customer/updatecustomerbyid/{userid}` updateCustomer

## product-controller Product Controller

- DELETE `/product/deleteproduct/{productid}` deleteProduct
- GET `/product/getallproducts` getAllProducts
- GET `/product/getproduct/{productid}` getProduct
- POST `/product/insertproduct` insertProduct
- PUT `/product/updateproductbyid/{productid}` updateProduct

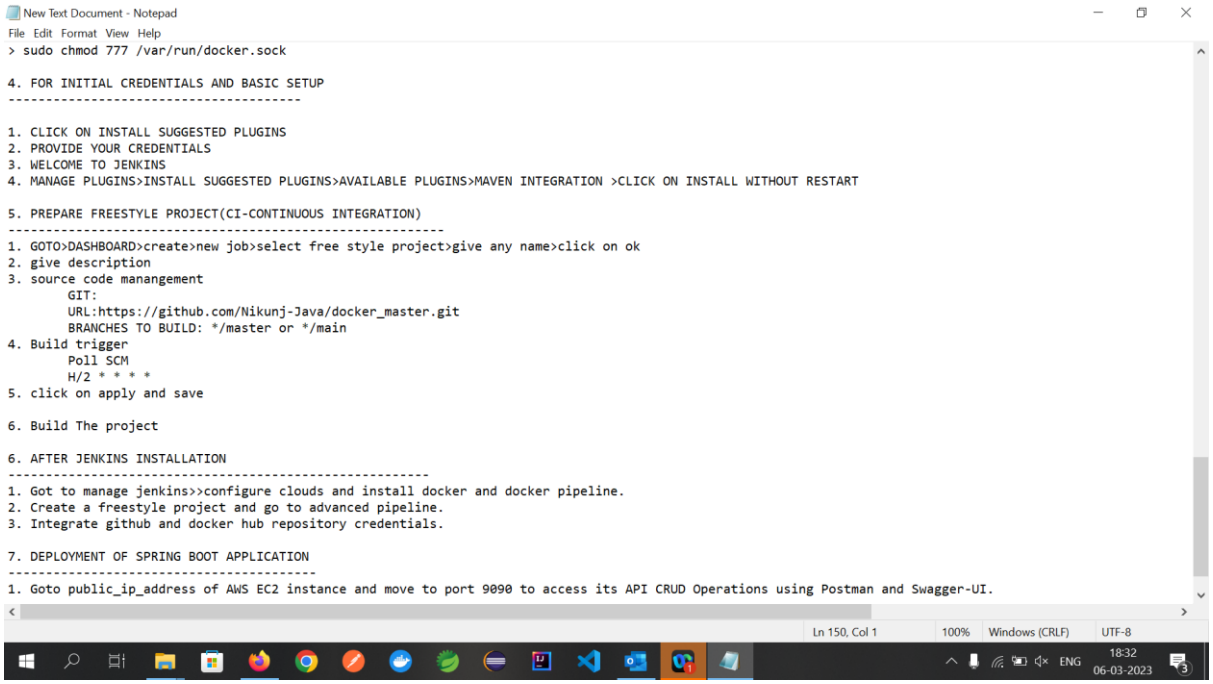
## purchased-product-controller Purchased Product Controller

- DELETE `/purchased/deletepurchasedproduct/{productid}` deletePurchasedProduct
- GET `/purchased/getallpurchasedproducts` getAllPurchasedProducts
- GET `/purchased/getpurchasedproduct/{productid}` getProduct
- POST `/purchased/insertpurchasedproduct` insertProduct
- PUT `/purchased/updatepurchasedproductbyid/{productid}` updatePurchasedProduct

## user-controller User Controller

- PUT `/user/addmoreuserproducts/{custid}/{prodid}` updateProductInCustomer
- GET `/user/getproducts/{userid}` getproducts
- POST `/user/insertproductinuser/{userid}/{prodid}` insertProductUserInDB
- POST `/user/signup` insertUser
- PUT `/user/updateuserbyid/{userid}` updateUser

# SOURCE CODE

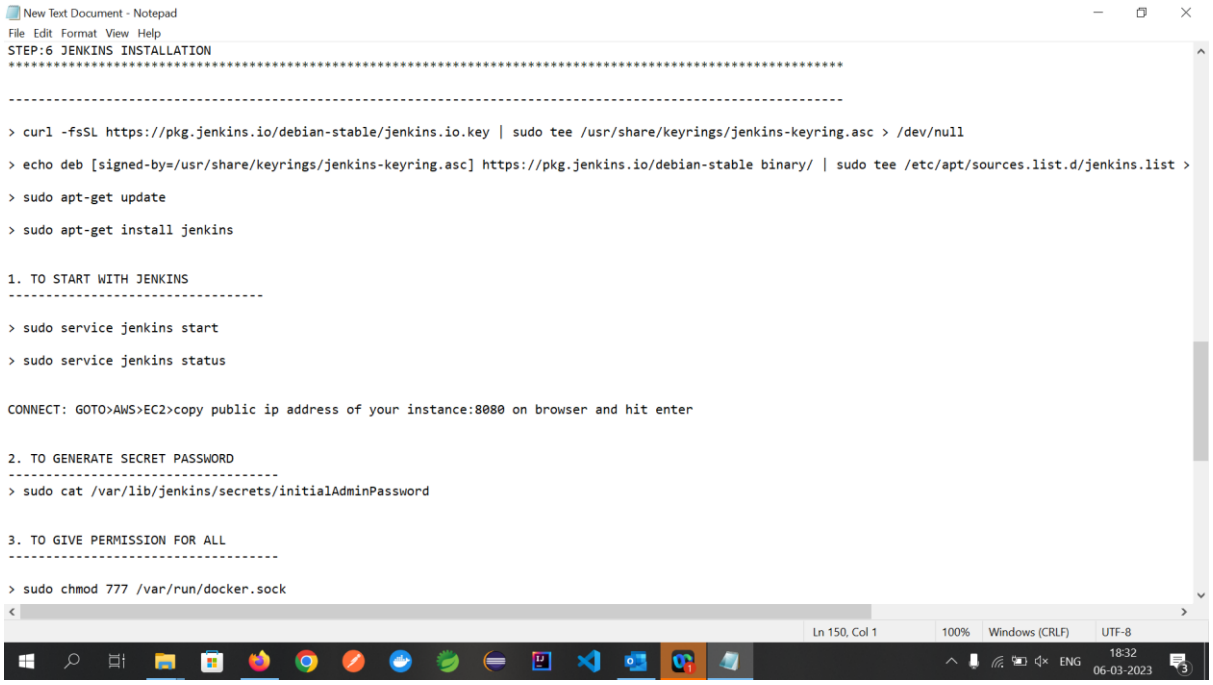


New Text Document - Notepad

File Edit Format View Help

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

4. FOR INITIAL CREDENTIALS AND BASIC SETUP
----------------------------------------

1. CLICK ON INSTALL SUGGESTED PLUGINS
2. PROVIDE YOUR CREDENTIALS
3. WELCOME TO JENKINS
4. MANAGE PLUGINS>INSTALL SUGGESTED PLUGINS>AVAILABLE PLUGINS>MAVEN INTEGRATION >CLICK ON INSTALL WITHOUT RESTART

5. PREPARE FREESTYLE PROJECT(CI-CONTINUOUS INTEGRATION)
----------------------------------------------------------
1. GOTO>DASHBOARD>create>new job>select free style project>give any name>click on ok
2. give description
3. source code manangement
       GIT:
       URL:https://github.com/Nikunj-Java/docker_master.git
       BRANCHES TO BUILD: */master or */main
4. Build trigger
       Poll SCM
       H/2 * * * *
5. click on apply and save

6. Build The project

6. AFTER JENKINS INSTALLATION
-----------------------------------------------------
1. Got to manage jenkins>>configure clouds and install docker and docker pipeline.
2. Create a freestyle project and go to advanced pipeline.
3. Integrate github and docker hub repository credentials.

7. DEPLOYMENT OF SPRING BOOT APPLICATION
-----------------------------------------
1. Goto public_ip_address of AWS EC2 instance and move to port 9090 to access its API CRUD Operations using Postman and Swagger-UI.
```

Ln 150, Col 1 | 100% | Windows (CRLF) | UTF-8

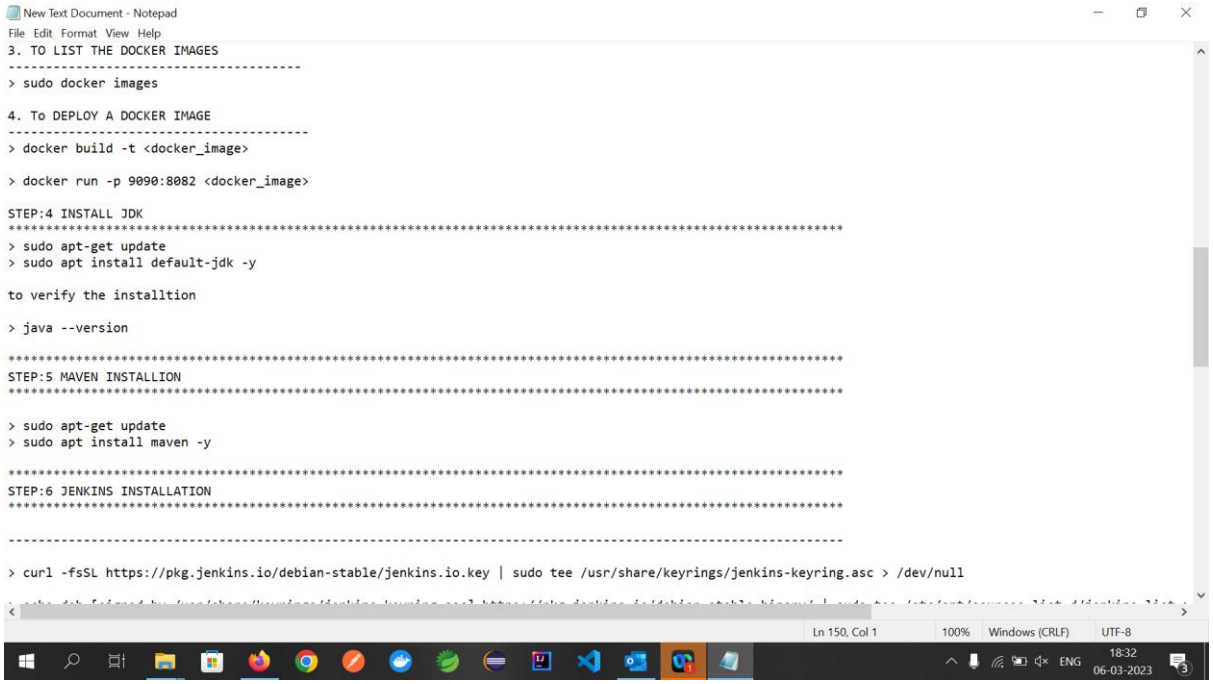


New Text Document - Notepad

File Edit Format View Help

```
STEP:6 JENKINS INSTALLATION
**********************************************************************************************************

----------------------------------------------------------------------------------------------------------

> curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo tee /usr/share/keyrings/jenkins-keyring.asc > /dev/null

> echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] https://pkg.jenkins.io/debian-stable binary/ | sudo tee /etc/apt/sources.list.d/jenkins.list >

> sudo apt-get update

> sudo apt-get install jenkins


1. TO START WITH JENKINS
---------------------------------

> sudo service jenkins start

> sudo service jenkins status


CONNECT: GOTO>AWS>EC2>copy public ip address of your instance:8080 on browser and hit enter


2. TO GENERATE SECRET PASSWORD
-------------------------------------
> sudo cat /var/lib/jenkins/secrets/initialAdminPassword


3. TO GIVE PERMISSION FOR ALL
-------------------------------------

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

Ln 150, Col 1 | 100% | Windows (CRLF) | UTF-8


New Text Document - Notepad
File Edit Format View Help
3. TO LIST THE DOCKER IMAGES
> sudo docker images
4. To DEPLOY A DOCKER IMAGE
> docker build -t <docker_image>
> docker run -p 9090:8082 <docker_image>
STEP:4 INSTALL JDK
> sudo apt-get update
> sudo apt install default-jdk -y
to verify the installtion
> java --version
STEP:5 MAVEN INSTALLION
> sudo apt-get update
> sudo apt install maven -y
STEP:6 JENKINS INSTALLATION
> curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo tee /usr/share/keyrings/jenkins-keyring.asc > /dev/null
Ln 150, Col 1
100%
Windows (CRLF)
UTF-8
ENG
18:32
06-03-2023


New Text Document - Notepad
File Edit Format View Help
link:https://docs.docker.com/engine/install/ubuntu/
> sudo apt-get update
> sudo apt-get install ca-certificates curl gnupg lsb-release
> sudo mkdir -m 0755 -p /etc/apt/keyrings
> curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
> echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" |
> sudo apt-get update
> sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
1. TO VERIFY THE INSTALLTION
> sudo docker -v
OUTPUT:Docker version 23.0.1, build a5ee5b1
> sudo docker --version
> sudo docker info
2. TO LIST DOCKER CONTAINERS
> sudo docker container ls
3. TO LIST THE DOCKER IMAGES
> sudo docker images
Ln 150, Col 1
100%
Windows (CRLF)
UTF-8
ENG
18:32
06-03-2023

New Text Document - Notepad

File Edit Format View Help

```
****************************************************************
STEP:1 AWS UBUNTU INSTANCE
****************************************************************
1. PREPARE AWS INSTANCE(UBUNTU SERVER 22.04 LTS (HVM) ,SSD VOLUME TYPE)
2. SERCURITY : ADD PORT NO :80 WITH CUSTOM TCP IP
3. DOWNLOAD  .PEM KEY TO CONNECT YOUR INSTANCE WITH YOUR LOCAL MACHINE USING MOBA X-TERM


****************************************************************
STEP:2 CONNECT USING MOBA X-TERM
****************************************************************
OPEN MOB X-TERM
> cd d:    //d: is my drive
> cd phase-5   //phase-5 is my folder where i have copied .pem key

> goto >aws>instance>choose your  instance>connect>ssh>copy the example key>

>open moba x-term >right click> enter

****************************************************************
STEP:3 DOCKER INSTALLATION ON UBUNTU OS
****************************************************************
goto>Google>Docker Installation on Ubuntu OS


link:https://docs.docker.com/engine/install/ubuntu/
----------------------------------------------------------------

> sudo apt-get update

> sudo apt-get install ca-certificates curl gnupg lsb-release

> sudo mkdir -m 0755 -p /etc/apt/keyrings
> curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
```

Ln 150, Col 1 | 100% | Windows (CRLF) | UTF-8

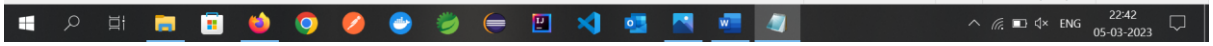
ENG 18:32 06-03-2023

01 Jenkins With Docker - Notepad

File Edit Format View Help

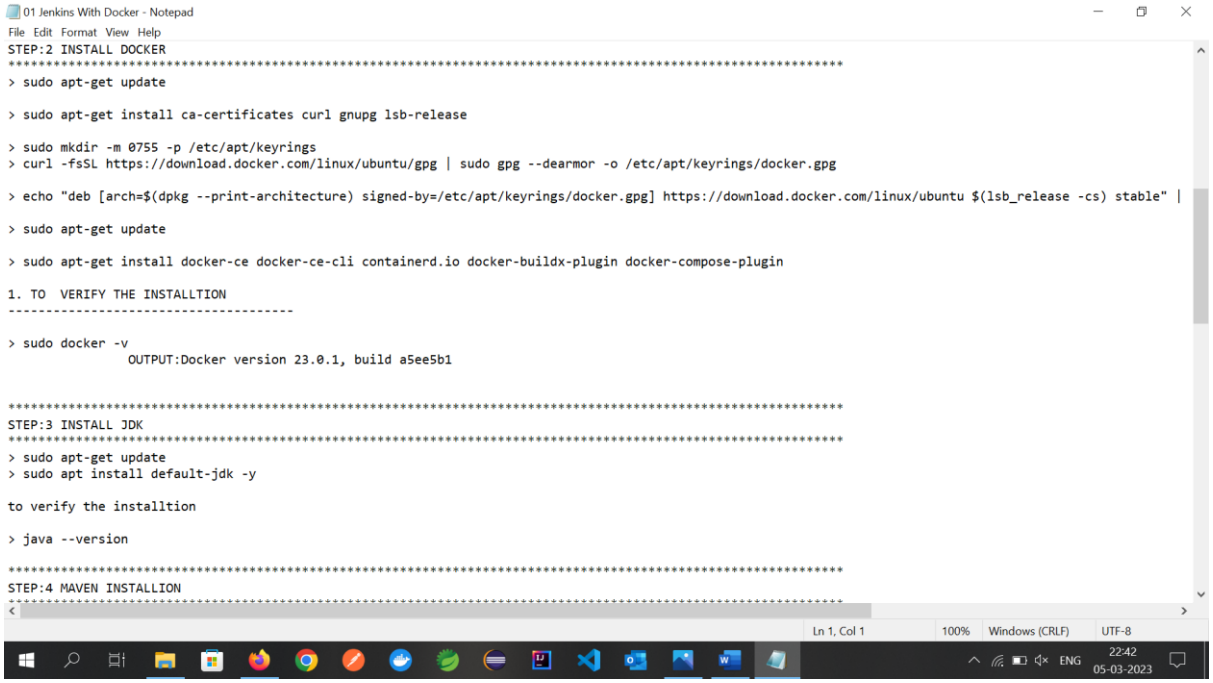
```


2. TO GENERATE SECRET PASSWORD
-----------------------------------
> sudo cat /var/lib/jenkins/secrets/initialAdminPassword


3. TO GIVE PERMISSION FOR ALL
-----------------------------------

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

4. FOR INITIAL CREDENTIALS AND BASIC SETUP
-----------------------------------------

1. CLICK ON INSTALL SUGGESTED PLUGINS
2. PROVIDE YOUR CREDENTIALS
3. WELCOME TO JENKINS
4. MANAGE PLUGINS>INSTALL SUGGESTED PLUGINS>AVAILABLE PLUGINS>MAVEN INTEGRATION >CLICK ON INSTALL WITHOUT RESTART

5. PREPARE FREESTYLE PROJECT(CI-CONTINUOUS INTEGRATION)
-------------------------------------------------------
1. GOTO>DASHBOARD>create>new job>select free style project>give any name>click on ok
2. give description
3. source code manangement
       GIT:
       URL:https://github.com/Nikunj-Java/docker_master.git
       BRANCHES TO BUILD: */master or */main
4. Build trigger
       Poll SCM
       H/2 * * * *
5. click on apply and save

6. Build The project
```

Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8



ENG 22:42 05-03-2023


01 Jenkins With Docker - Notepad
File Edit Format View Help
STEP:4 MAVEN INSTALLION
> sudo apt-get update
> sudo apt install maven -y
STEP:5 JENKINS INSTALLATION
> GOTO> GOOGLE> HOW TO INSTALL JENKINS IN UBUNTU
lINK: https://www.jenkins.io/doc/book/installing/linux/
> curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo tee /usr/share/keyrings/jenkins-keyring.asc > /dev/null
> echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] https://pkg.jenkins.io/debian-stable binary/ | sudo tee /etc/apt/sources.list.d/jenkins.list >
> sudo apt-get update
> sudo apt-get install jenkins
1. TO START WITH JENKINS
> sudo service jenkins start
> sudo service jenkins status
CONNECT: GOTO>AWS>EC2>copy public ip address of your instance:8080 on browser and hit enter
Ln 1, Col 1
100%
Windows (CRLF)
UTF-8
ENG
22:42
05-03-2023

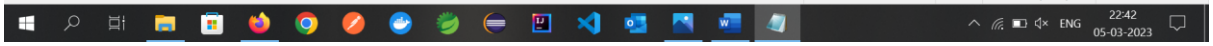

01 Jenkins With Docker - Notepad
File Edit Format View Help
STEP:2 INSTALL DOCKER
> sudo apt-get update
> sudo apt-get install ca-certificates curl gnupg lsb-release
> sudo mkdir -m 0755 -p /etc/apt/keyrings
> curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
> echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" |
> sudo apt-get update
> sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
1. TO VERIFY THE INSTALLTION
> sudo docker -v
OUTPUT:Docker version 23.0.1, build a5ee5b1
STEP:3 INSTALL JDK
> sudo apt-get update
> sudo apt install default-jdk -y
to verify the installtion
> java --version
STEP:4 MAVEN INSTALLION
Ln 1, Col 1
100%
Windows (CRLF)
UTF-8
ENG
22:42
05-03-2023

01 Jenkins With Docker - Notepad

File Edit Format View Help

```
**********************************************************************************************************************
JENKINS WITH DOCKER
**********************************************************************************************************************

STEP:1 CREATE AWS UBUNTU INSTANCE WITH PORT NO:8080

STEP:2 INSTALL DOCKER

STEP:3 INSTALL JDK

STEP:4 INSTALL MAVEN

STEP:5 INSTALL JENKINS

**********************************************************************************************************************
STEP:1 CREATE AWS UBUNTU INSTANCE WITH PORT NO:8080
**********************************************************************************************************************
> prepare AWS Ubuntu INSTANCE (Ubuntu Server 22.04 LTS (HVM) ,SSD Volume Type)
> Security : add port no :8080 with custom TCP rule
> Download.pem key to local machine and connect using moba x-term


> open moba x-term
> cd d:
> cd phase-5

> goto>aws>instance>your instance>connect>ssh>copy example key

> goto >mobax-term terminal>right click>hit enter

or connect directly from web browser

**********************************************************************************************************************
STEP:2 INSTALL DOCKER
**********************************************************************************************************************
```

Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8

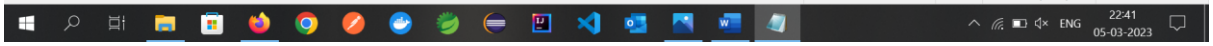


02 Push Docker Image to DockerHub - Notepad

File Edit Format View Help

```
**************************************************************************
STEP:1 LOOGED IN TO DOCKER HUB ACCOUNT
**************************************************************************

create an Account on dockerHub:

Link: https://hub.docker.com/
--------------------------------------------------------------------------

> sudo docker login

give username:
give password:
[Note: password is not visible]

if you are getting some permission denied error

> sudo chmod 666 /var/run/docker.sock

> sudo docker login

give username:
give password:
[Note: password is not visible]

**************************************************************************
STEP:2 PUSH AN IMAGE TO DOCKER HUB
**************************************************************************

EG: sudo docker tag <name of your image> <username>/<name of your directory-you can give any name>

> sudo docker tag dockerimage <docker_hub_name>/<image_name>
> sudo docker push <docker_hub_name>/<image_name>
```

Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8

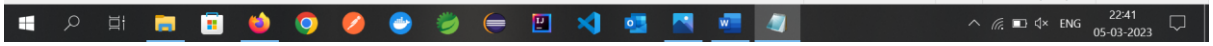


01 Docker Installation on EC2 Instance - Notepad

File Edit Format View Help

```
> sudo docker pull mysql

************************************************************
STEP:5 PULL GIT HUB IMAGES (CUSTOM IMAGES)
************************************************************

LINK: https://github.com/Nikunj-Java/docker_master.git

> git clone https://github.com/Nikunj-Java/docker_master.git
> ls (to check available folders)
> cd docker_master


1. LET'S PREPARE THE IMAGE IN A DOCKER CONTAINER
------------------------------------------------------------
> sudo docker build -t dockerimage . (. is mandatory)
> sudo docker images (to check the image is prepared or not?)

2. LET'S RUN THE IMAGE IN A DOCKER CONTAINER
------------------------------------------------------------

> sudo docker run -d --name mycontainer -p 80:80 dockerimage

> sudo docker container ls

3. TO CHECK WITH APP IS RUNNING OR NOT?
------------------------------------------------------------
> curl localhost

this will open 'index.php' page of your app

goto>AWS>Instance>Copy Public Ip Address
goto>webbrowser>ipAddress:80
```

Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8



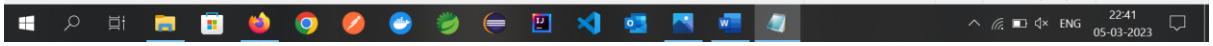
ENG 22:41 05-03-2023

01 Docker Installation on EC2 Instance - Notepad

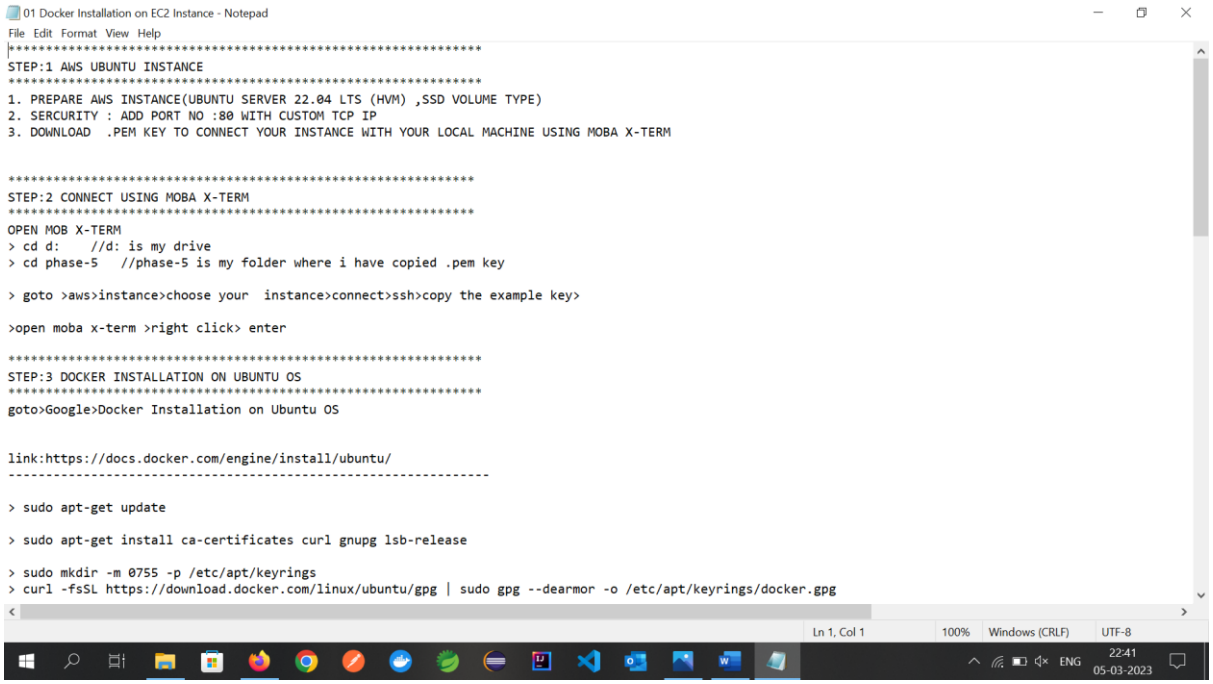
File Edit Format View Help

```
> curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

> echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" |

> sudo apt-get update

> sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

1. TO  VERIFY THE INSTALLTION
--------------------------------------

> sudo docker -v
              OUTPUT:Docker version 23.0.1, build a5ee5b1
> sudo docker --version

> sudo docker info

2. TO LIST DOCKER CONTAINERS
--------------------------------------
> sudo docker container ls

3. TO LIST THE DOCKER IMAGES
--------------------------------------
> sudo docker images

4. TO CHECK DOCKER VOLUME
--------------------------------------
> sudo docker volume ls

************************************************************
STEP:4 PULL DOCKER IMAGES
************************************************************

> sudo docker pull ubuntu
```

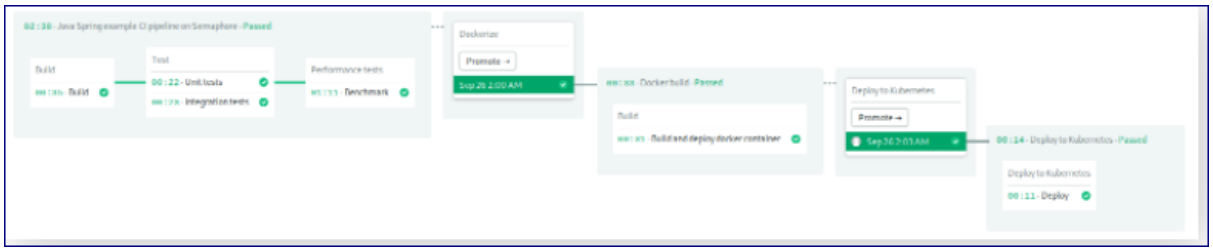
Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8

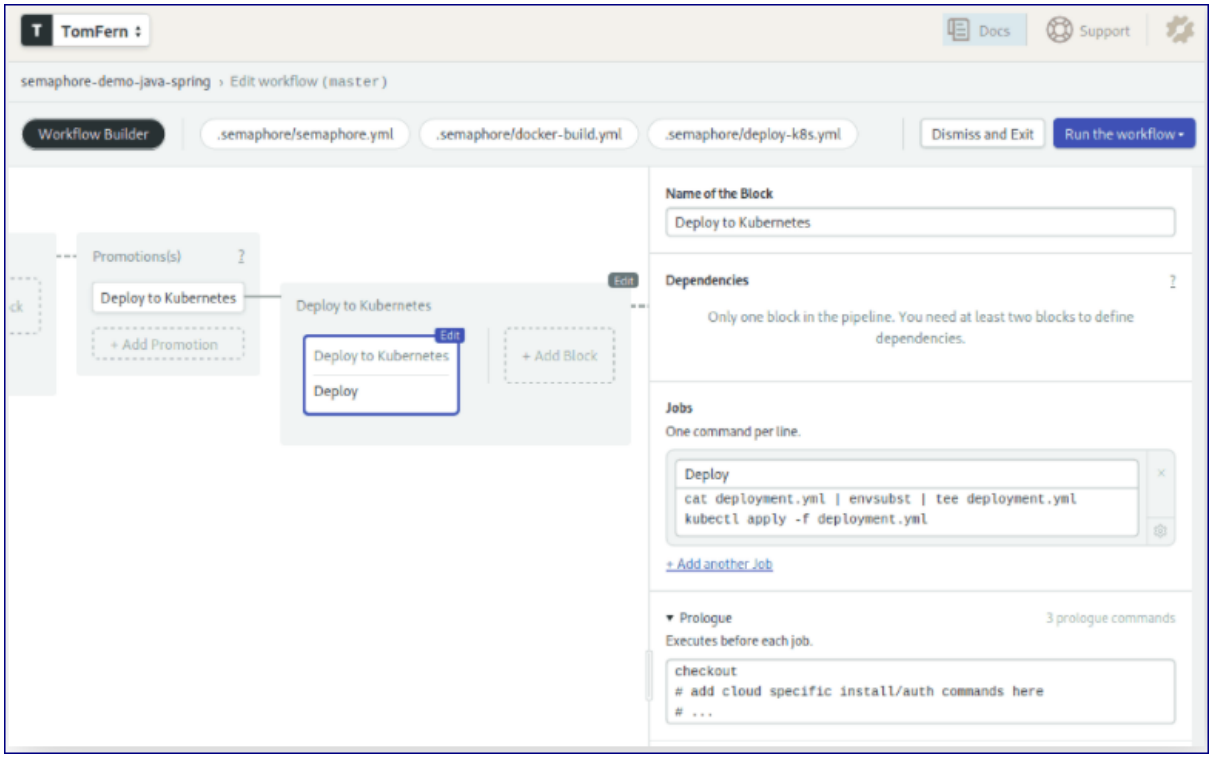


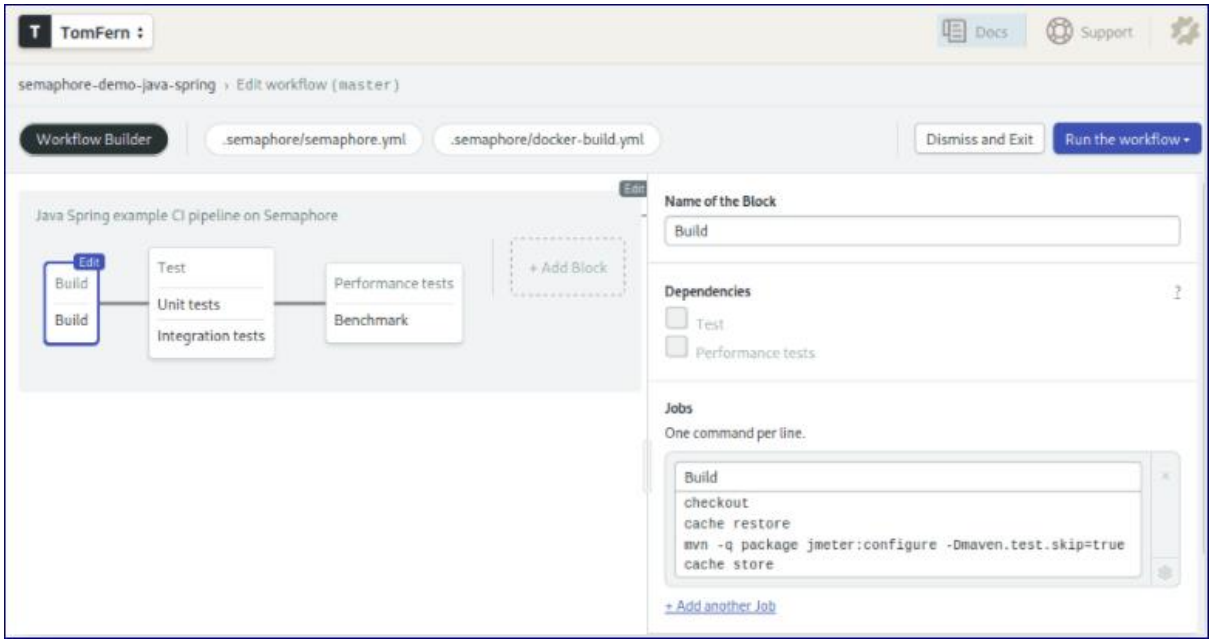
ENG 22:41 05-03-2023


01 Docker Installation on EC2 Instance - Notepad
File Edit Format View Help
**********
STEP:1 AWS UBUNTU INSTANCE
**********
1. PREPARE AWS INSTANCE(UBUNTU SERVER 22.04 LTS (HVM) ,SSD VOLUME TYPE)
2. SERCURITY : ADD PORT NO :80 WITH CUSTOM TCP IP
3. DOWNLOAD .PEM KEY TO CONNECT YOUR INSTANCE WITH YOUR LOCAL MACHINE USING MOBA X-TERM
**********
STEP:2 CONNECT USING MOBA X-TERM
**********
OPEN MOB X-TERM
> cd d: //d: is my drive
> cd phase-5 //phase-5 is my folder where i have copied .pem key
> goto >aws>instance>choose your instance>connect>ssh>copy the example key>
>open moba x-term >right click> enter
**********
STEP:3 DOCKER INSTALLATION ON UBUNTU OS
**********
goto>Google>Docker Installation on Ubuntu OS
link:https://docs.docker.com/engine/install/ubuntu/
> sudo apt-get update
> sudo apt-get install ca-certificates curl gnupg lsb-release
> sudo mkdir -m 0755 -p /etc/apt/keyrings
> curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
Ln 1, Col 1
100%
Windows (CRLF)
UTF-8
ENG
22:41
05-03-2023

# Flowchart




TomFern
Docs
Support
semaphore-demo-java-spring › Edit workflow (master)
Workflow Builder
.semaphore/semaphore.yml
.semaphore/docker-build.yml
.semaphore/deploy-k8s.yml
Dismiss and Exit
Run the workflow
Promotions(s)
Deploy to Kubernetes
+ Add Promotion
Deploy to Kubernetes
Deploy to Kubernetes
Deploy
+ Add Block
Name of the Block
Deploy to Kubernetes
Dependencies
Only one block in the pipeline. You need at least two blocks to define dependencies.
Jobs
One command per line.
Deploy
cat deployment.yml | envsubst | tee deployment.yml
kubectl apply -f deployment.yml
+ Add another Job
Prologue
3 prologue commands
Executes before each job.
checkout
# add cloud specific install/auth commands here
# ...


TomFern
Docs
Support
semaphore-demo-java-spring › Edit workflow (master)
Workflow Builder
.semaphore/semaphore.yml
.semaphore/docker-build.yml
Dismiss and Exit
Run the workflow
Java Spring example CI pipeline on Semaphore
Build
Build
Test
Unit tests
Integration tests
Performance tests
Benchmark
+ Add Block
Name of the Block
Build
Dependencies
Test
Performance tests
Jobs
One command per line.
Build
checkout
cache restore
mvn -q package jmeter:configure -Dmaven.test.skip=true
cache store
+ Add another Job

## Algorithm

Step 1> Start.

Step 2> The user must create an instance of AWS.

Step 3> Once the instance is created and running, then perform installation of both Jenkins and Docker given in the source code for guide.

Step 4> After Installation, user need to clone the spring boot application from GitHub repository.

Step 5> Create the corresponding docker image for that spring boot application.

Step 6> After docker image has been created user can deploy his application using docker container in respective port number.

Step 7> Install the necessary plugins and clouds such as docker and docker pipeline for Jenkins safe deployment.

Step 8> Stop