DEVOPS ASSIGNMENT

1. Setup a Jenkins master and slave architecture environment. Create Continuous Integration Build pipeline project for any Web applications using suitable build utilities to pull code from version control, build, validate, compile, test, install dependencies, package and deploy application on Slave node which serves as a Build Environment Servers.

2. Create Jenkins Continuous Delivery project to containerize any web application along with databases on a Docker hosted Jenkins Slave nodes (Docker Host should server as Build or Staging or Pre-Production Servers) and also ensure you push image of applications to user's Docker hub repository

3. Create Jenkins Continuous Deployment project to deploy containerized application on Kubernetes cluster environment (Kubernetes Cluster should be your Production Environment)

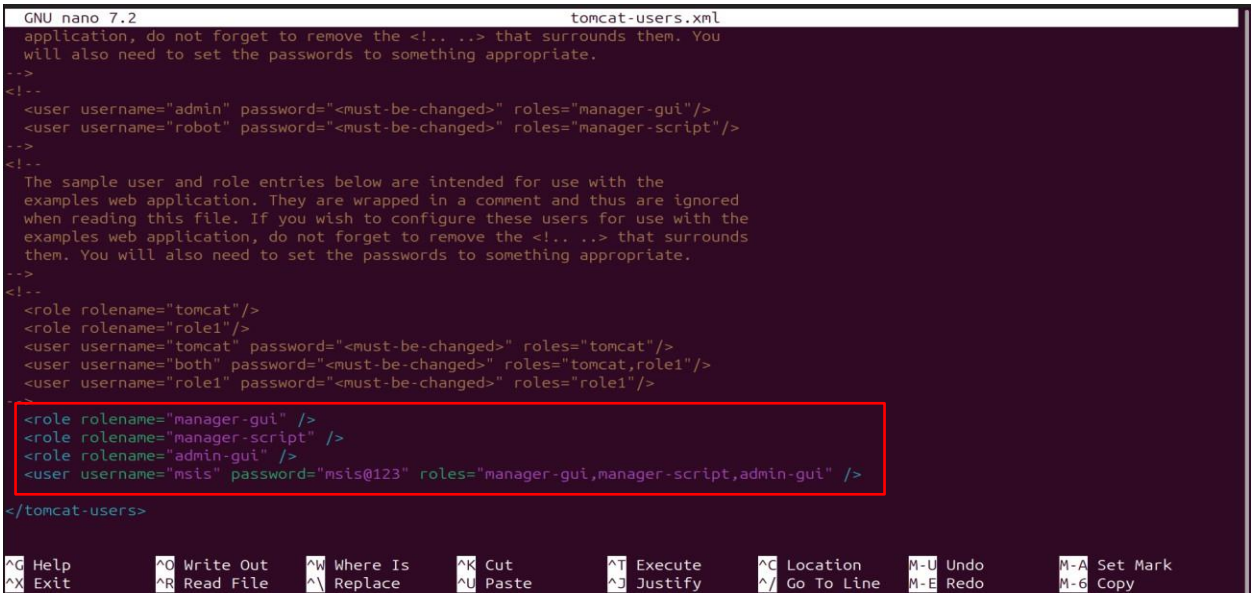
For Java

1.Implement CI/CD Build Pipeline for Java Application on Jenkins Server

Tomcat Installation reference link: https://docs.vultr.com/how-to-install-apache-tomcat-on-ubuntu-24-04

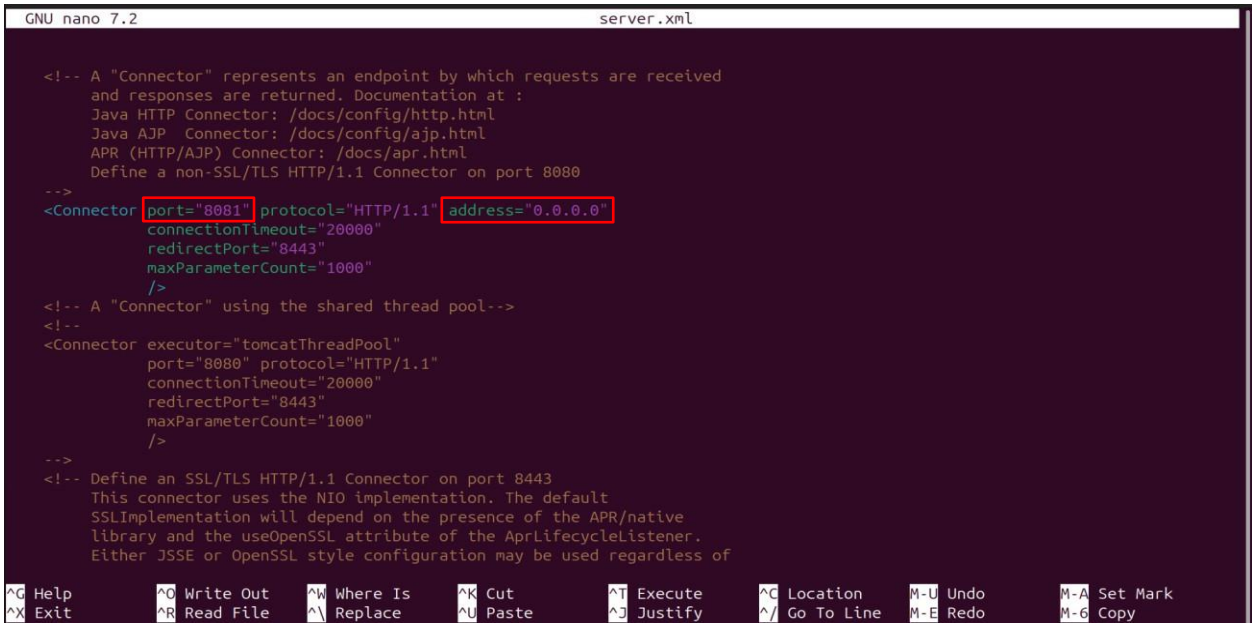
cd /opt/tomcat/conf

Sudo nano tomcat-users.xml



cd /opt/tomcat/conf

sudo nano server.xml



Plugins required:

- Maven Integration plugin

- Warnings: For Code review

- JUnit: For Testing (if required)

- Deploy to container

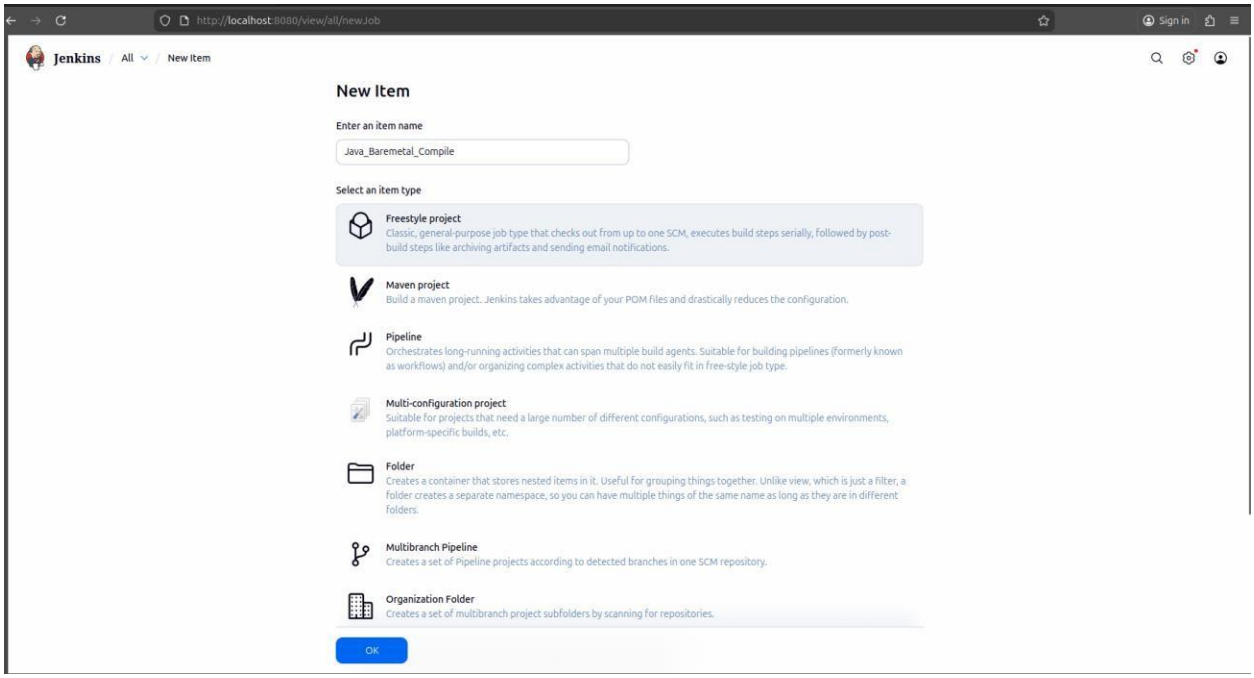
The above plugins can be installed by navigating to Manage Jenkins->Plugins.

GitHub Repo: https://github.com/sreepathysois/java-tomcat-maven-example.git

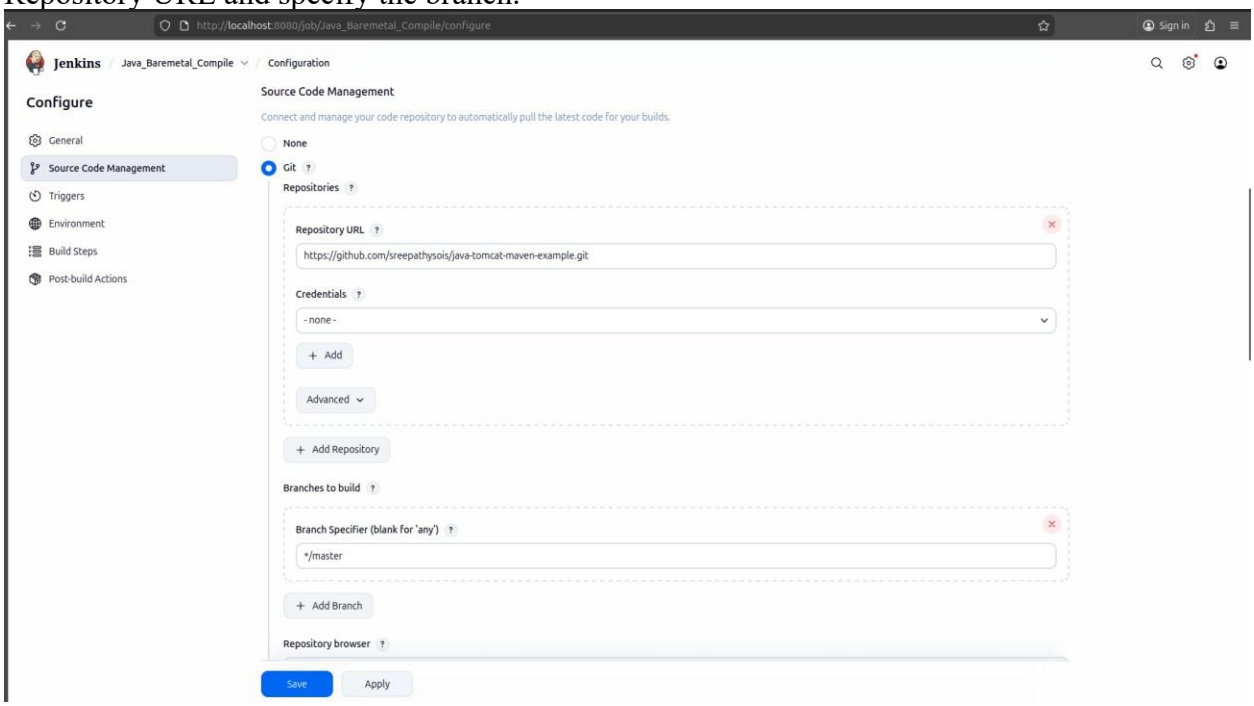
Log in to Jenkins
Server.

Step1: Compile

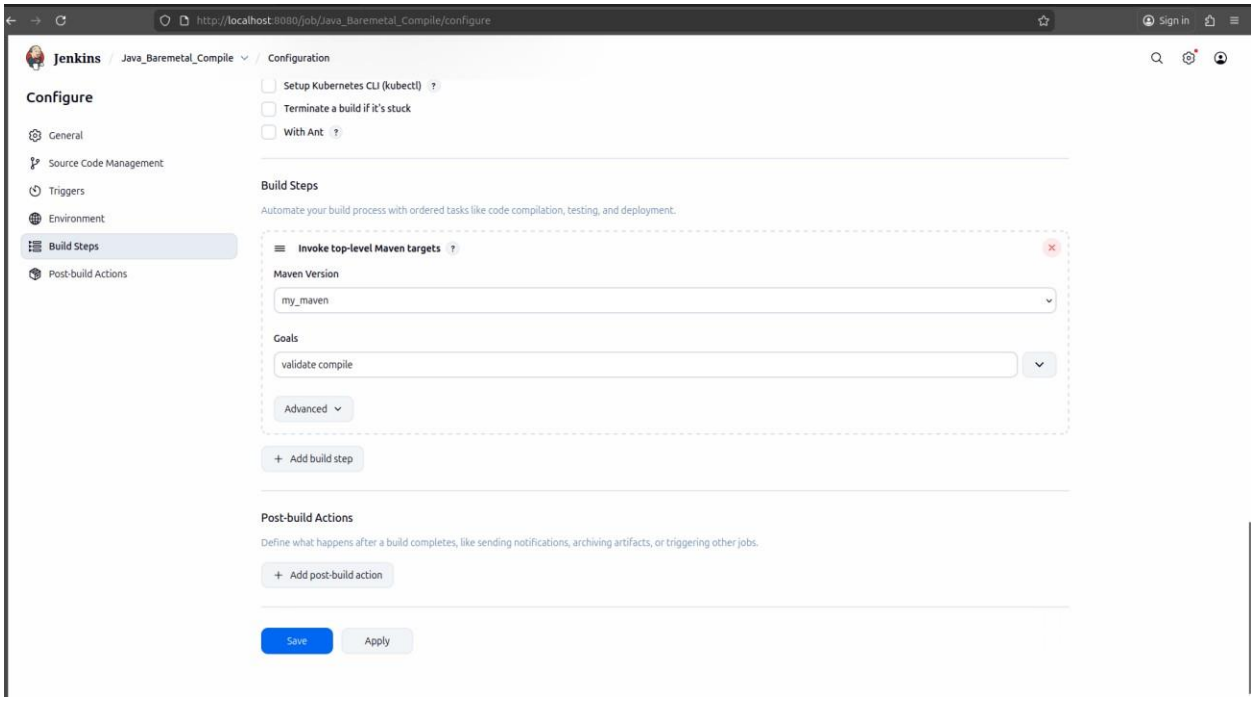
Create a Free Style Jenkins Project.
Provide a name to Project for Example: Java_Baremetal_compile then click Ok.



On the next page, select Source Code Management as Git, then provide the GitHub repository URL under Repository URL and specify the branch.

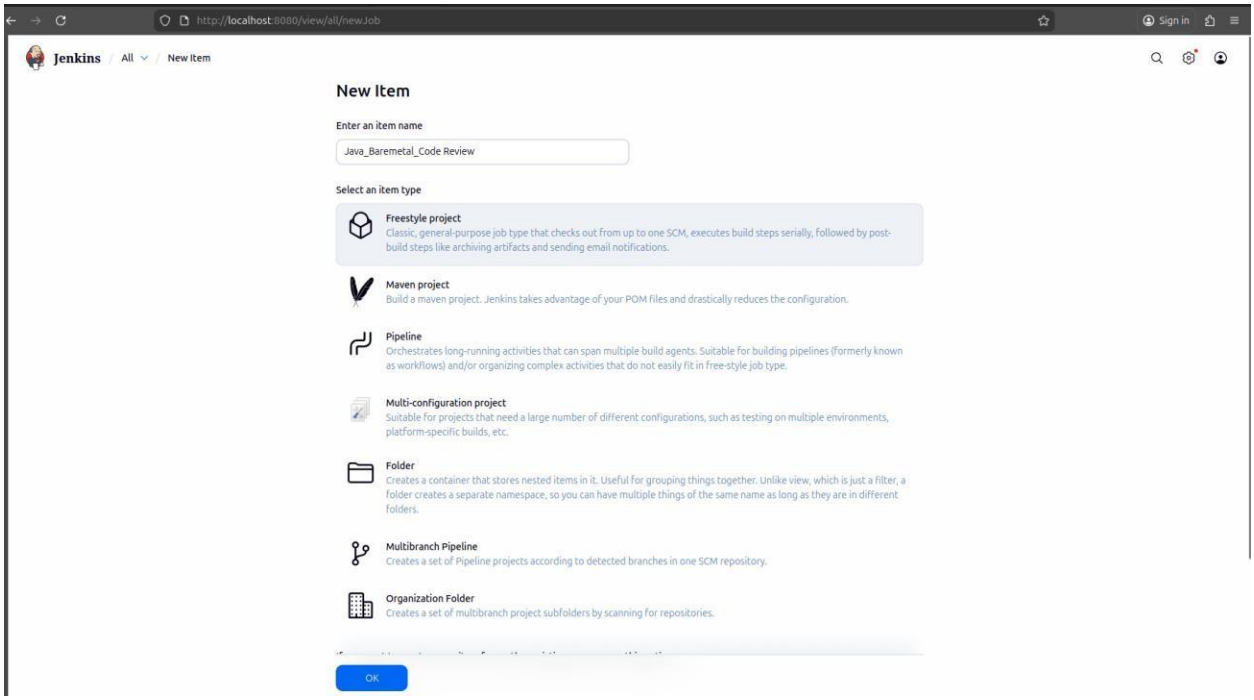


Under the Build step, select “Invoke Top-Level Maven Targets,” and under Goals, enter validate compile, then click Save.

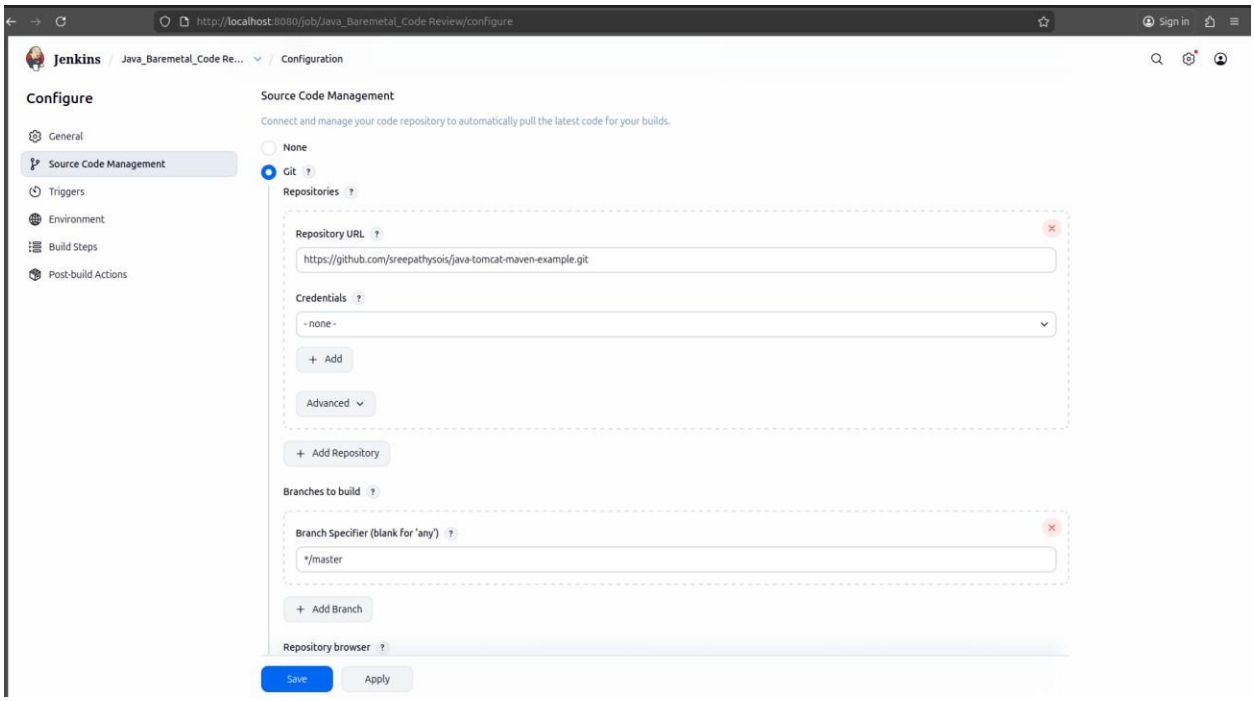


Step2: Code review

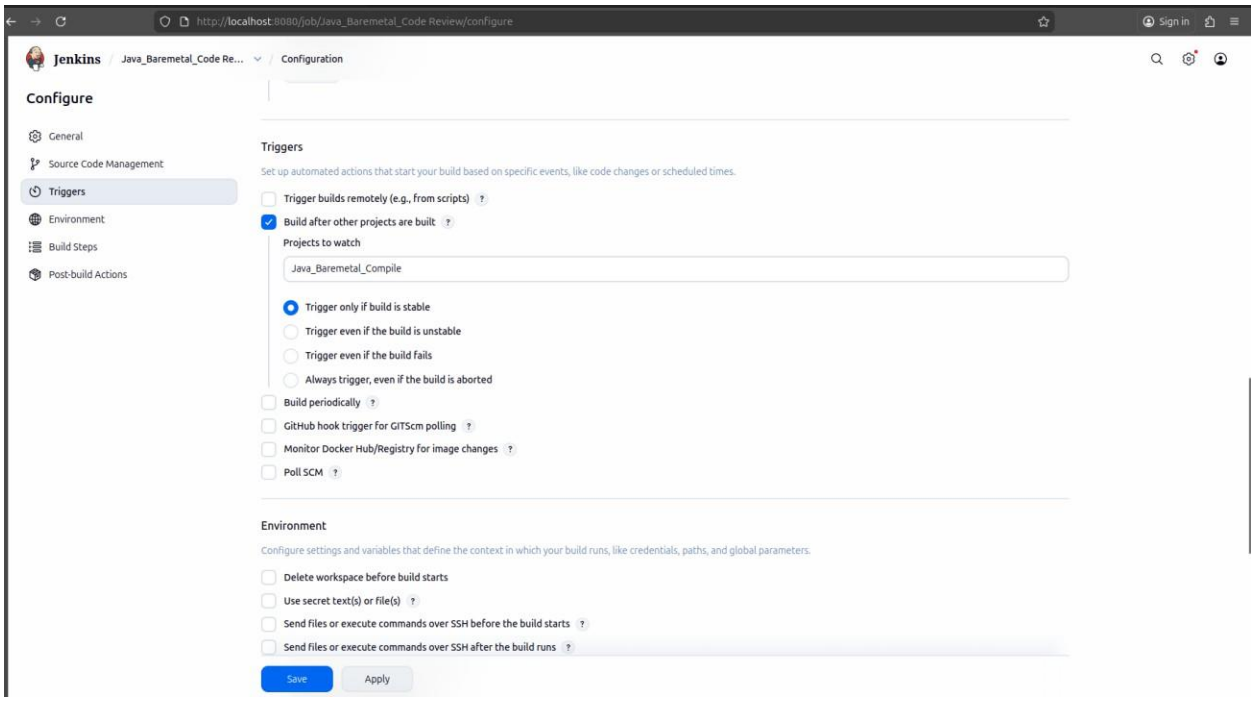
Create a Freestyle Jenkins project. Provide a name for the project, for example: Java_Baremetal_CodeReview, then click OK.



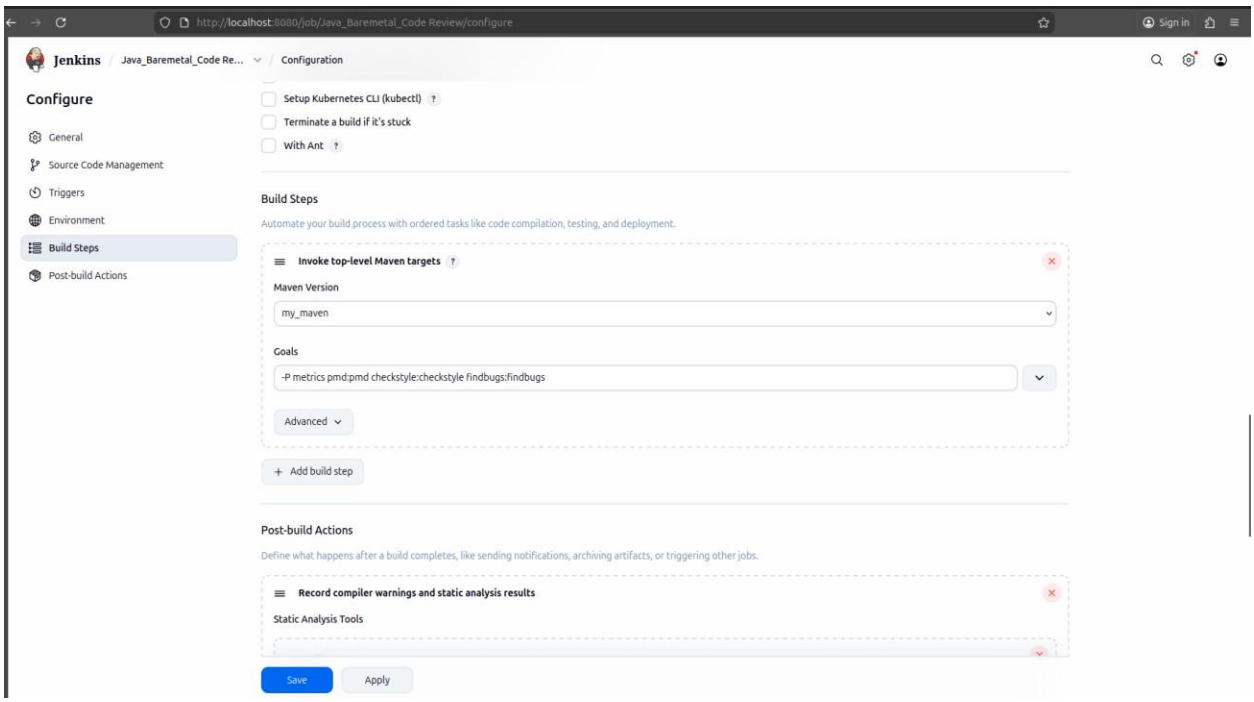
On the next page, select Source Code Management as Git, then provide the GitHub repository URL under Repository URL and specify the branch.



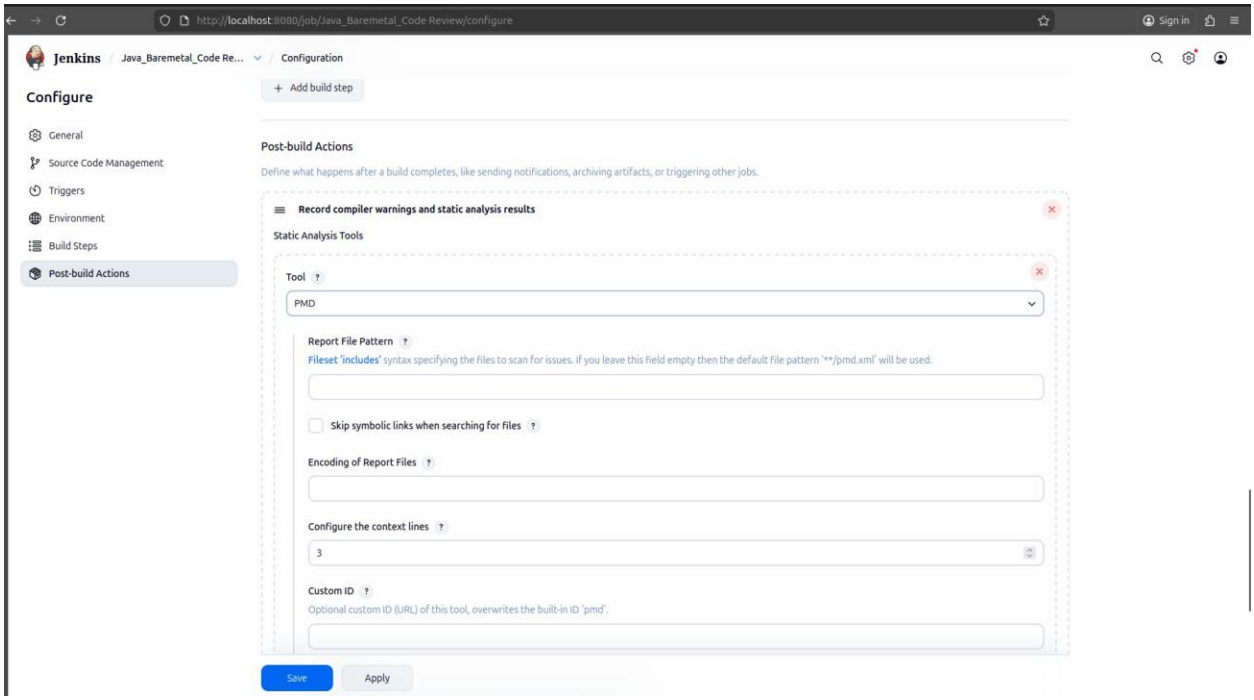
Under Triggers, select "Build after other projects are built," and under Projects to watch, enter the previous item name (Java_Baremetal_Compile). Then select "Trigger only if build is stable."

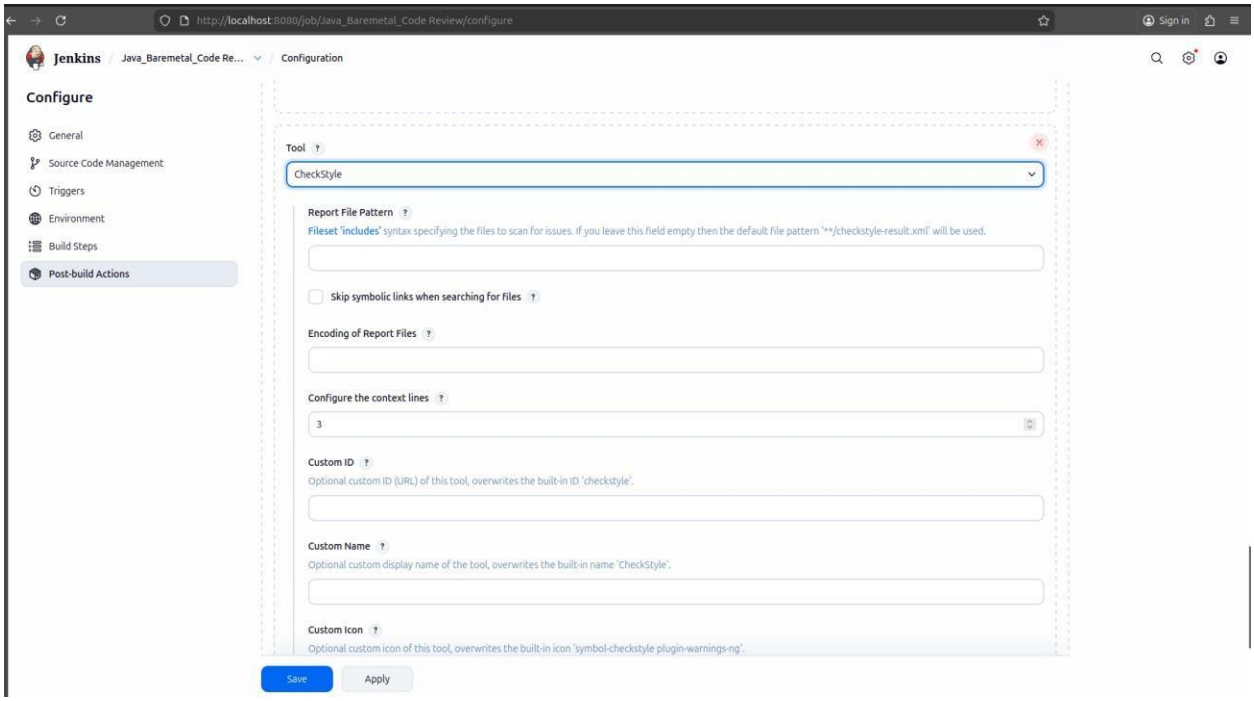


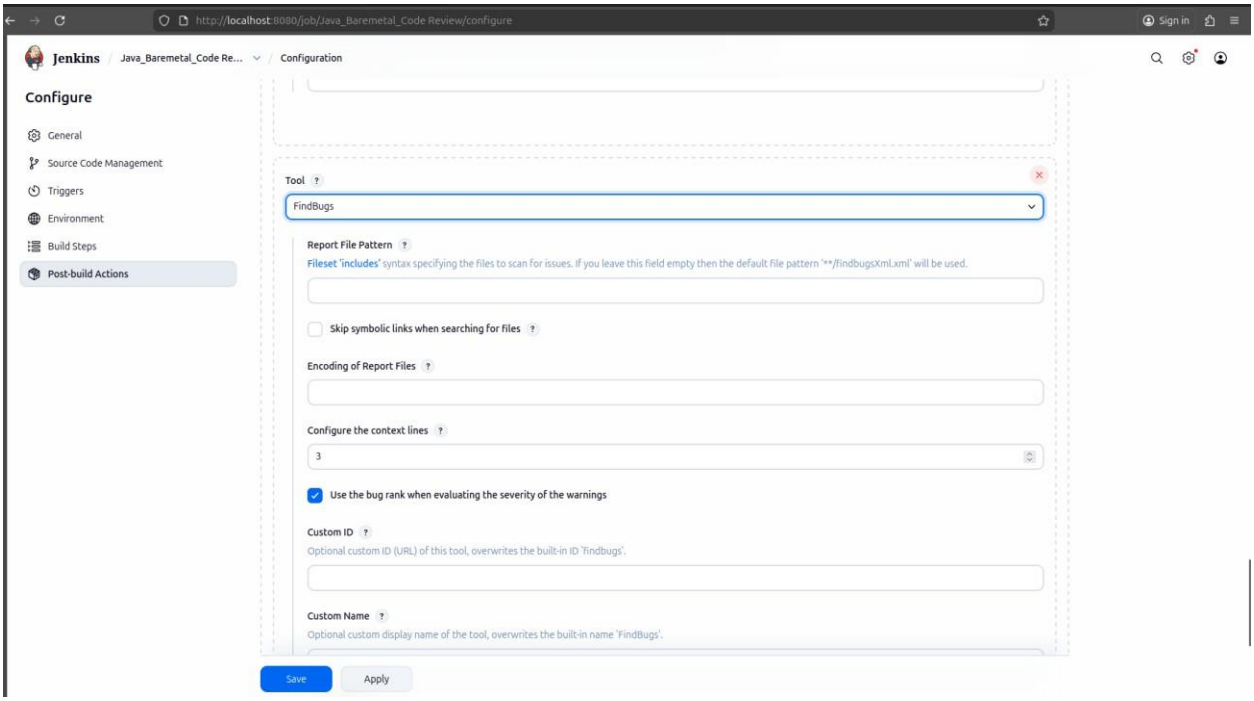
Under the Build step, select "Invoke Top-Level Maven Targets," and under Goals, enter -P metrics pmd:pmd checkstyle:checkstyle findbugs:findbugs, then click Save.

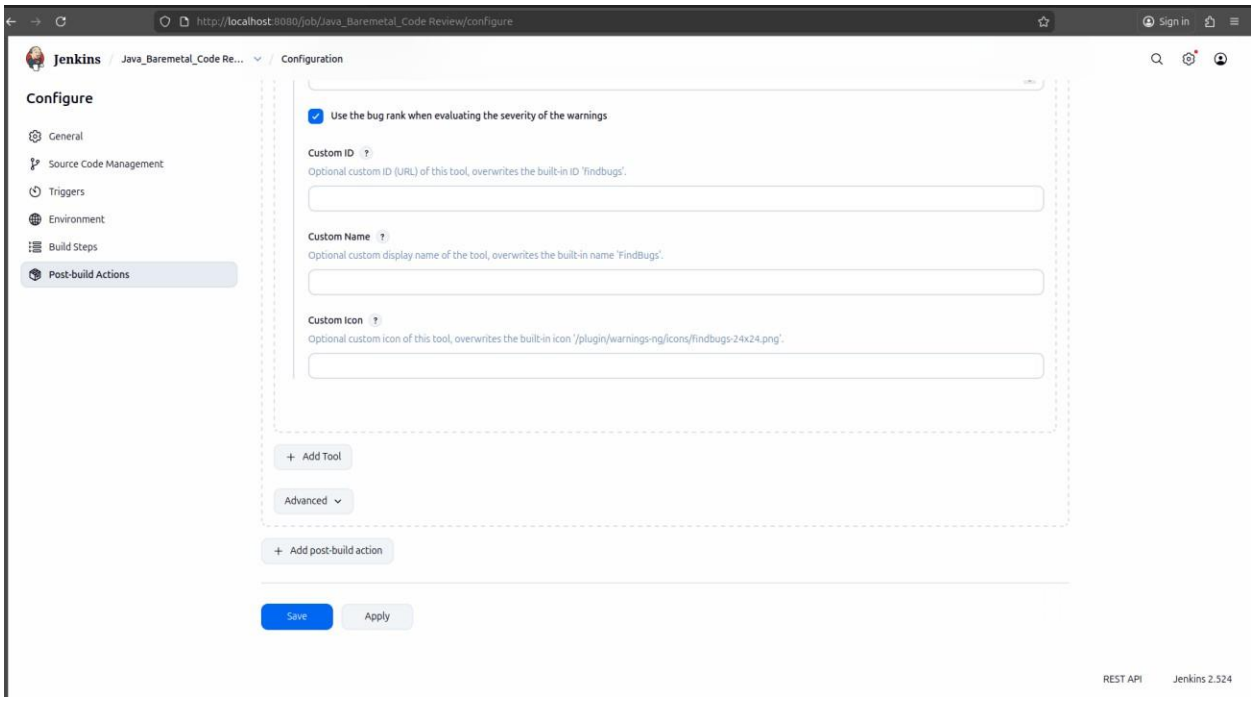


Under post-build, select "Record compiler warnings and static analysis results." Then select PMD, CheckStyle, and FindBugs, and click Save.



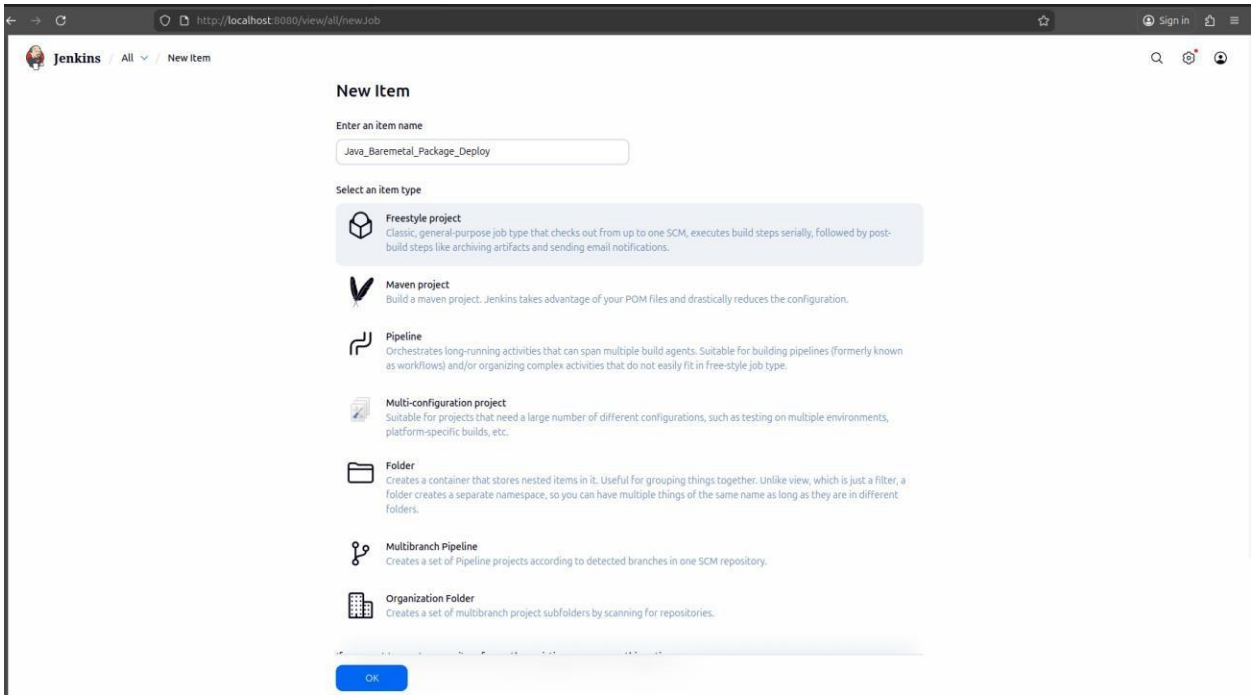

http://localhost:8080/job/Java_Baremetal_Code Review/configure
Sign in
Jenkins
Java_Baremetal_Code Re...
Configuration
Configure
General
Source Code Management
Triggers
Environment
Build Steps
Post-build Actions
Tool
CheckStyle
Report File Pattern
Fileset 'includes' syntax specifying the files to scan for issues. If you leave this field empty then the default file pattern '**/checkstyle-result.xml' will be used.
Skip symbolic links when searching for files
Encoding of Report Files
Configure the context lines
3
Custom ID
Optional custom ID (URL) of this tool, overwrites the built-in ID 'checkstyle'.
Custom Name
Optional custom display name of the tool, overwrites the built-in name 'CheckStyle'.
Custom Icon
Optional custom icon of this tool, overwrites the built-in icon 'symbol-checkstyle plugin-warnings-ng'.
Save
Apply


http://localhost:8080/job/Java_Baremetal_Code Review/configure
Sign in
Jenkins
Java_Baremetal_Code Re...
Configuration
Configure
General
Source Code Management
Triggers
Environment
Build Steps
Post-build Actions
Tool
FindBugs
Report File Pattern
Fileset 'includes' syntax specifying the files to scan for issues. If you leave this field empty then the default file pattern '**/findbugsXml.xml' will be used.
Skip symbolic links when searching for files
Encoding of Report Files
Configure the context lines
3
Use the bug rank when evaluating the severity of the warnings
Custom ID
Optional custom ID (URL) of this tool, overwrites the built-in ID 'findbugs'.
Custom Name
Optional custom display name of the tool, overwrites the built-in name 'FindBugs'.
Save
Apply

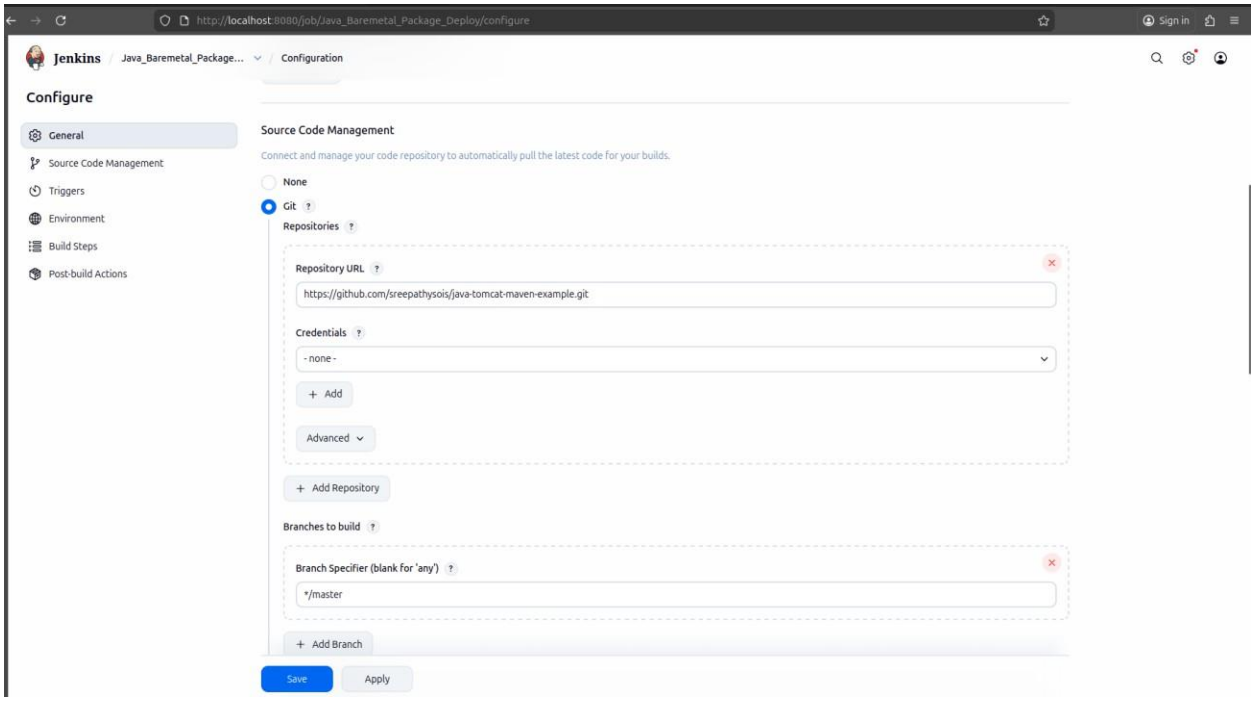


Step3: Package and Deploy

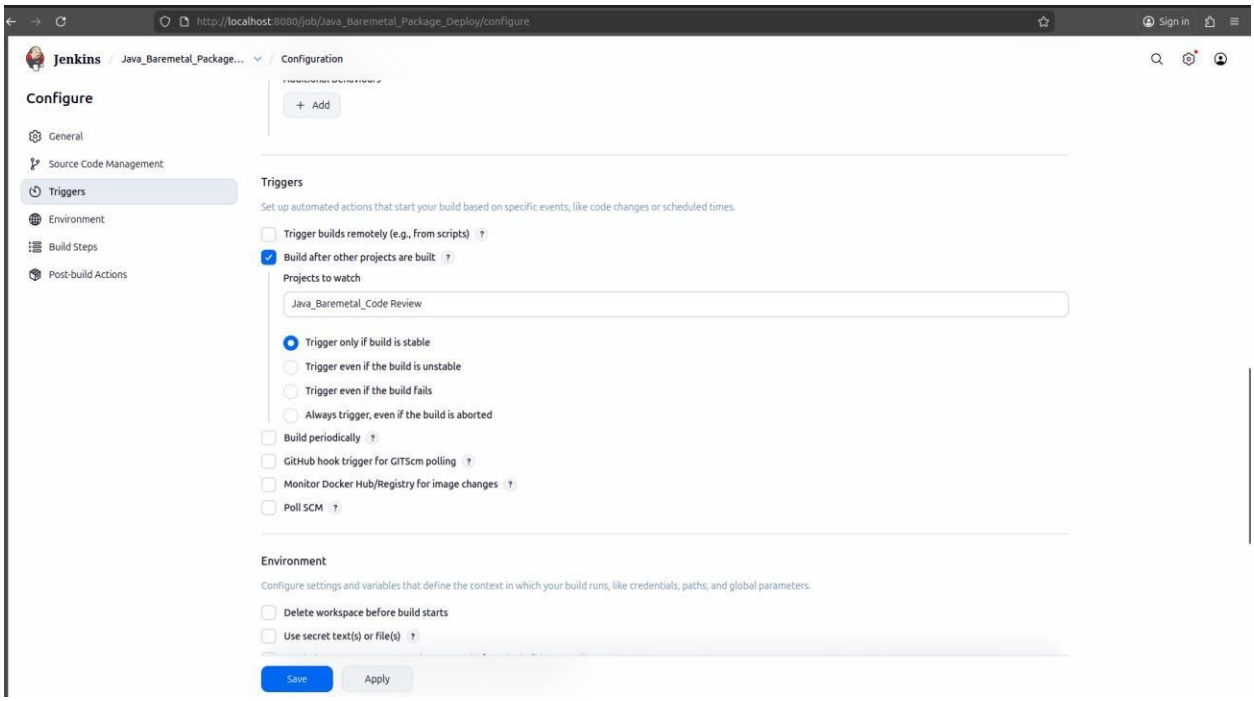
Create a **Freestyle Jenkins Project** and provide a name for the project, for example **Java_Baremetal_Package_Deploy**, then click **OK**.



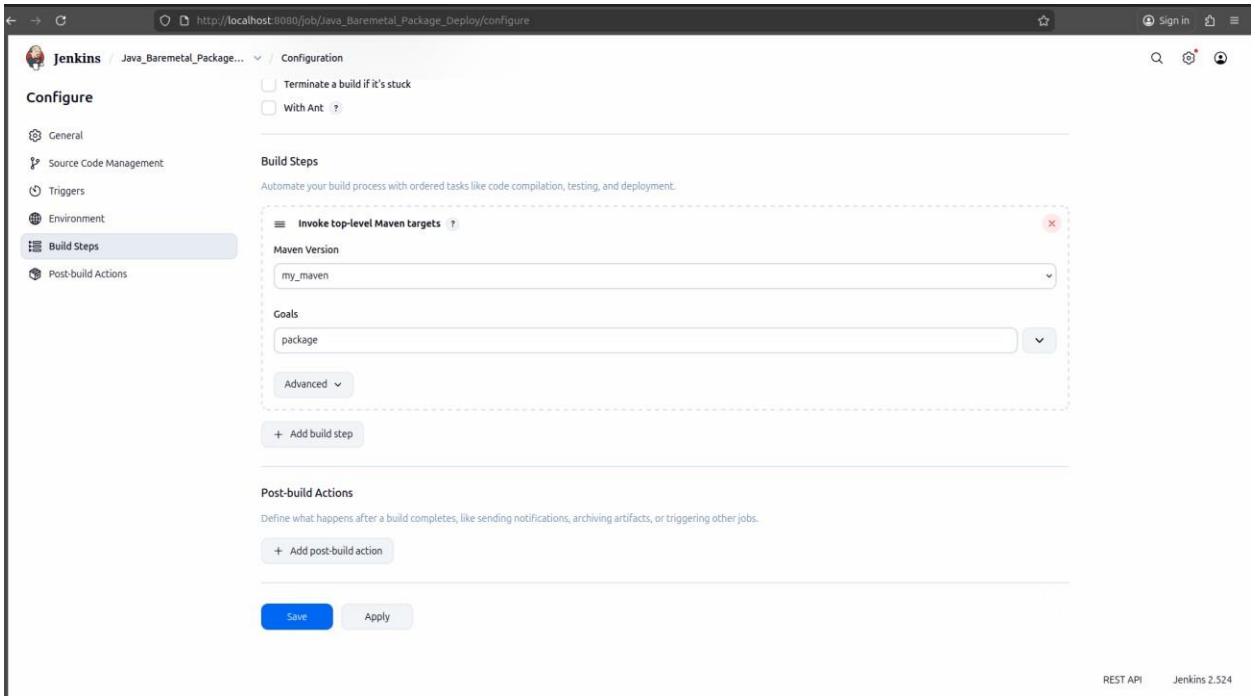
On the next page, select **Git** under **Source Code Management**, then enter the GitHub repository URL in the **Repository URL** field and specify the branch.



Under **Triggers**, select **"Build after other projects are built."** In the **Projects to watch** field, enter the name of the previous job (**Java_Baremetal_codeReview**). Then select **"Trigger only if build is stable."**

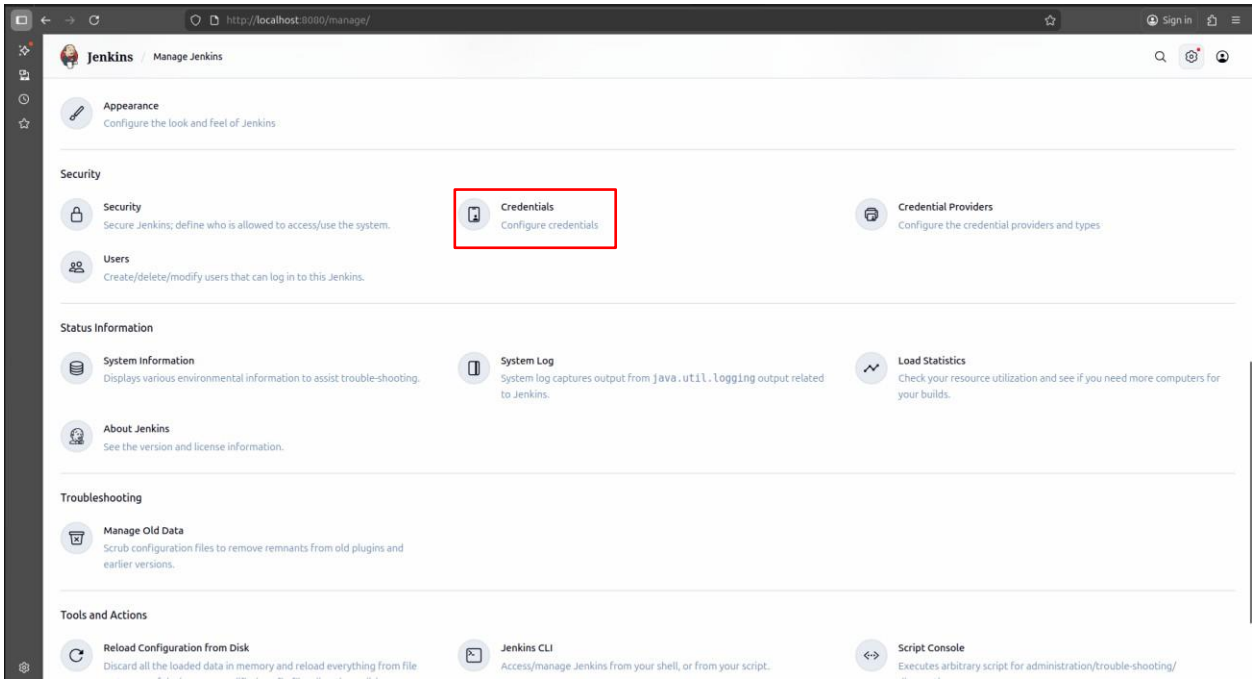


Under the **Build** step, select **Invoke Top-Level Maven Targets**, and in the **Goals** field, enter package. Then click **Save**.

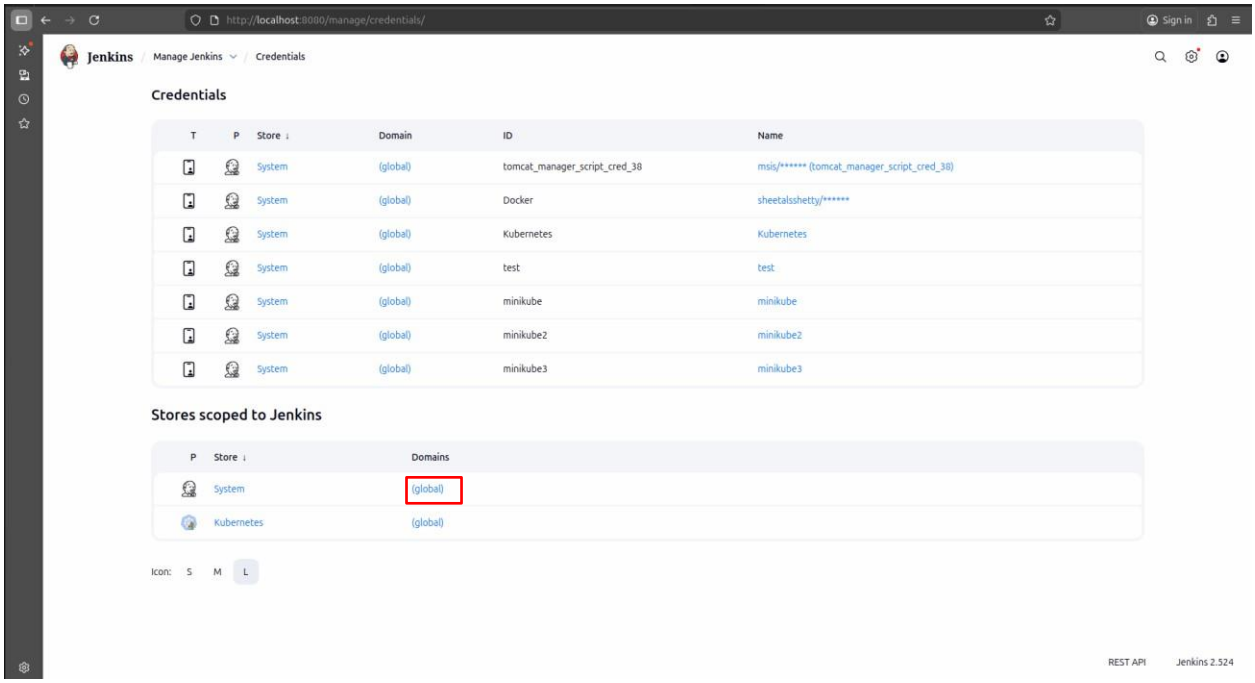


Under **Post-build Actions**, select **"Deploy to Container."**

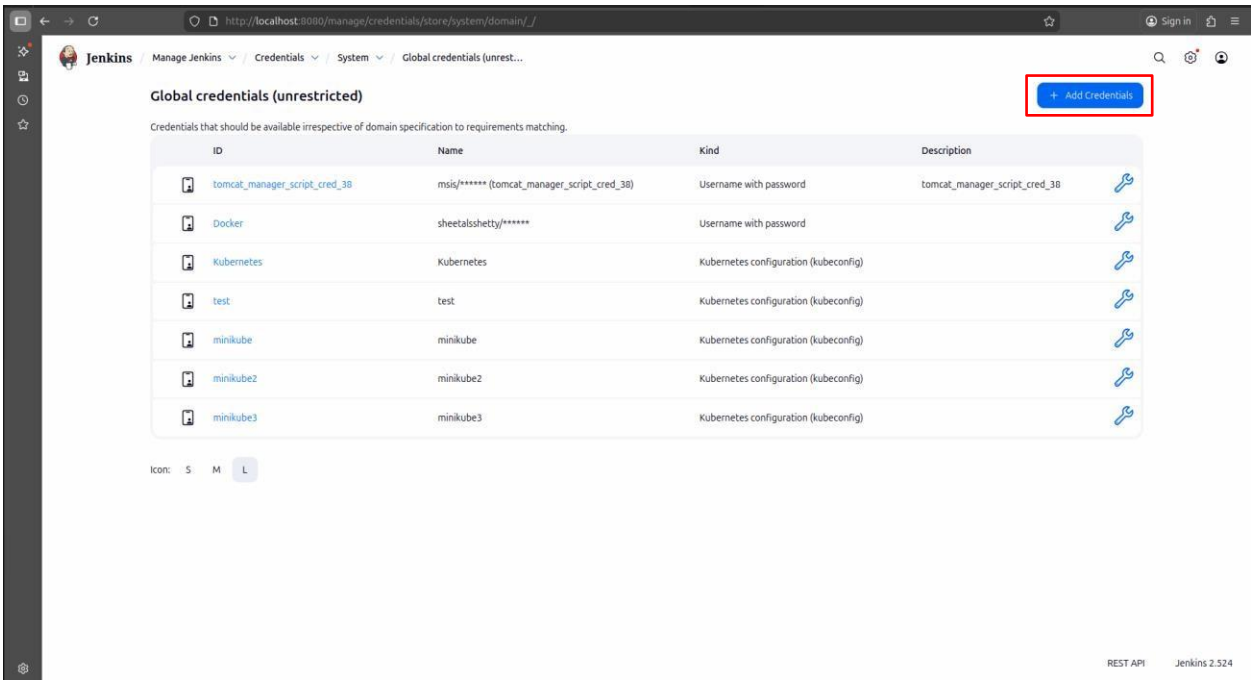
To configure Tomcat credentials in Jenkins, go to **Manage Jenkins → Credentials**.



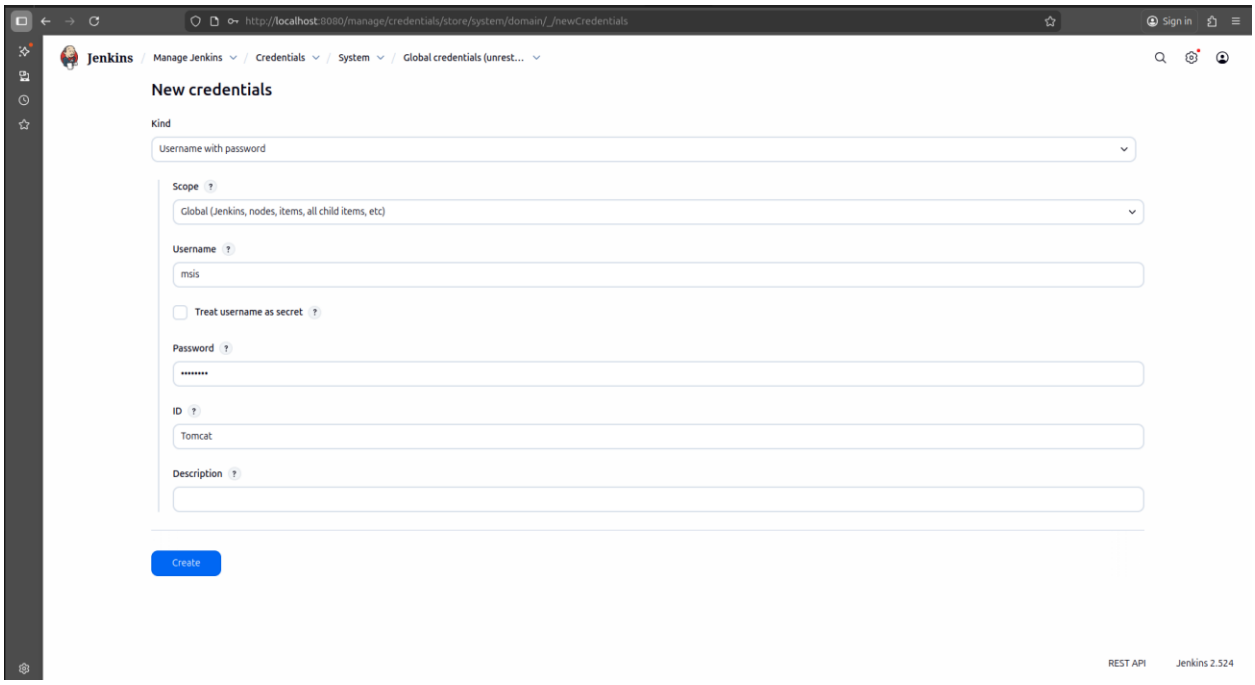
Click on Global.



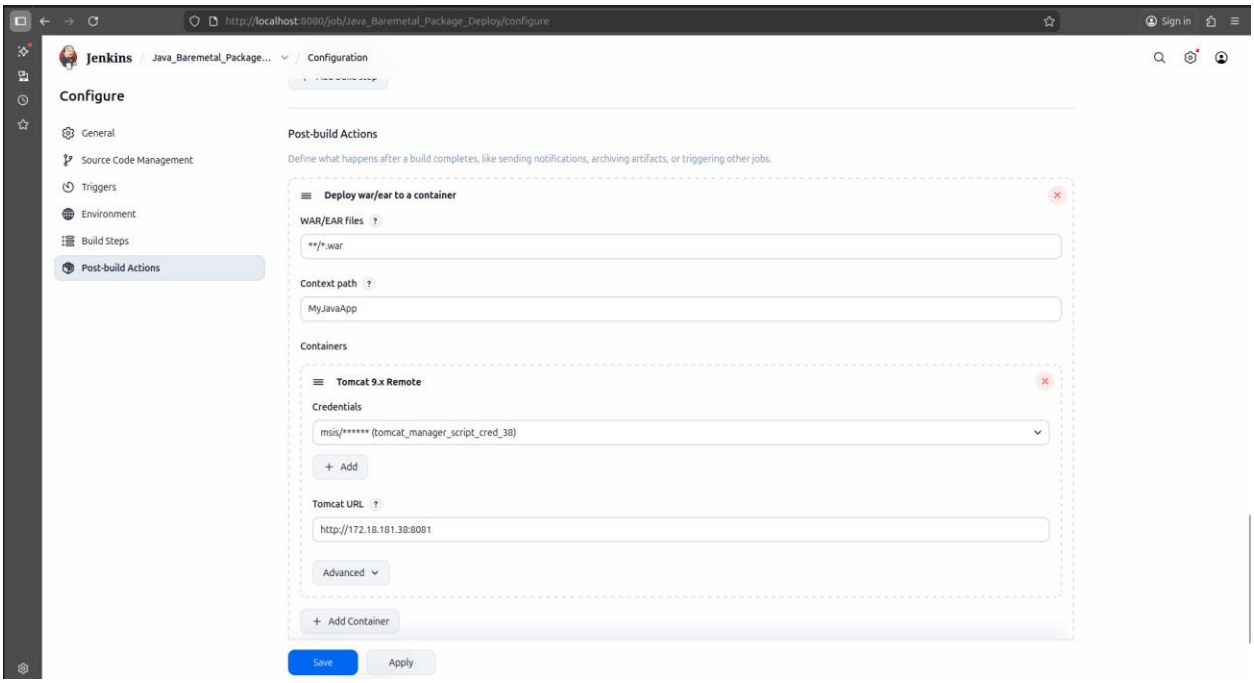
Click on Add Credential



Under **Kind**, select **"Username with Password."** Then enter the username and password that were configured in the tomcat-users.xml file.

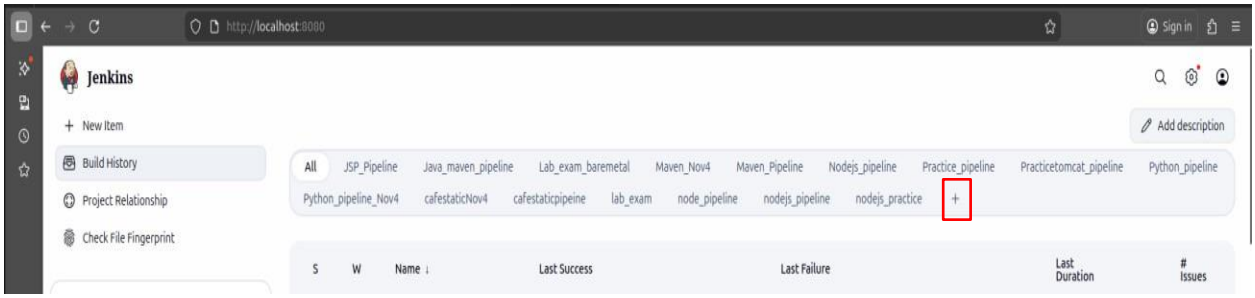


Go back to the Jenkins item **"Java_Baremetal_Package_Deploy."** Under **Deploy to container**, enter **/*.war in the **WAR/EAR Files** field, provide a name in the **Context path** (e.g., MyJavaApp), and select **Tomcat G.x Remote** as the container.
Choose the appropriate **Tomcat credential ID**, and in the **Tomcat URL** field, enter http://<IP>:<Tomcat_Port>.
Then click **Save**.

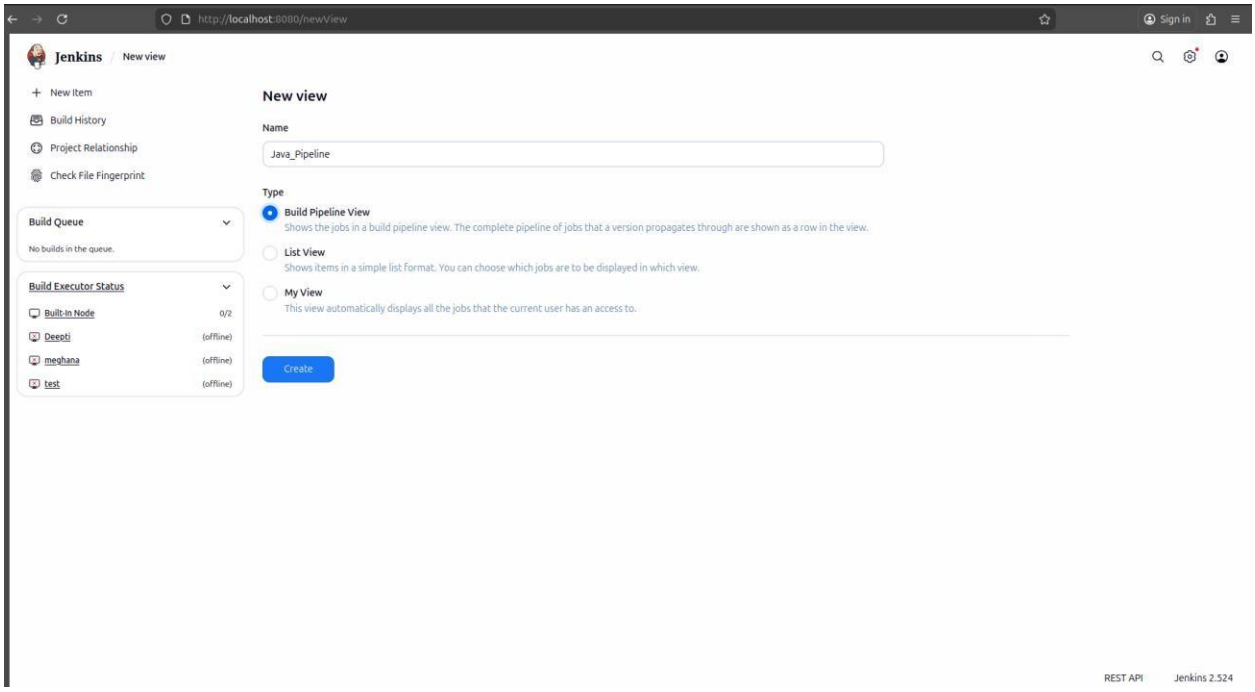


Now we will create the **Build Pipeline View**.

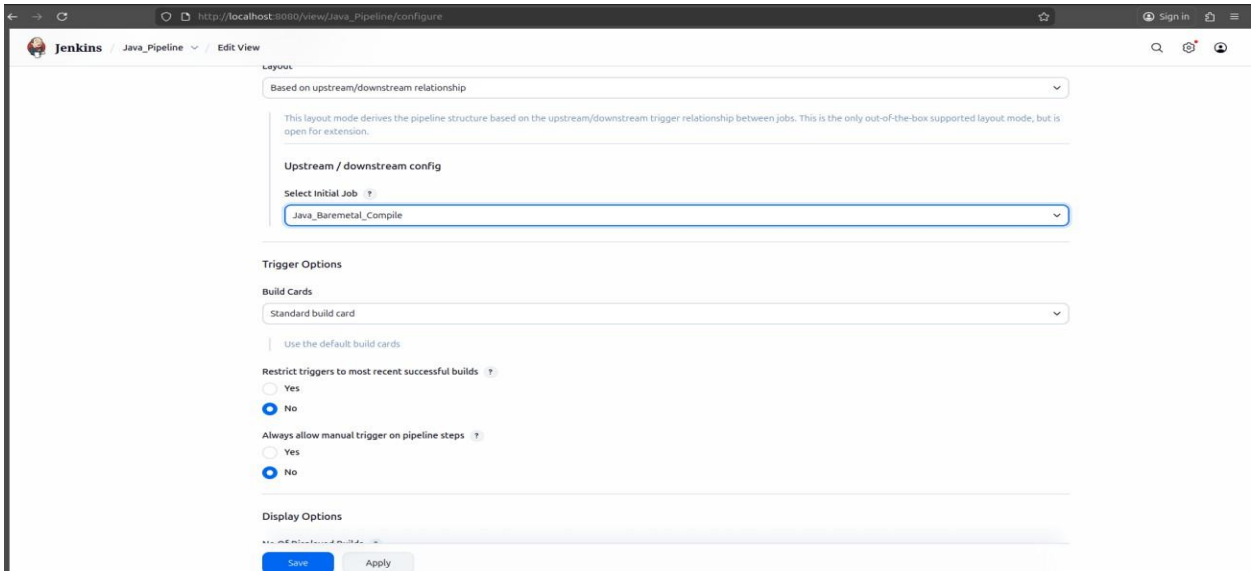
Navigate to the Jenkins home page and click on the **"+"** icon.



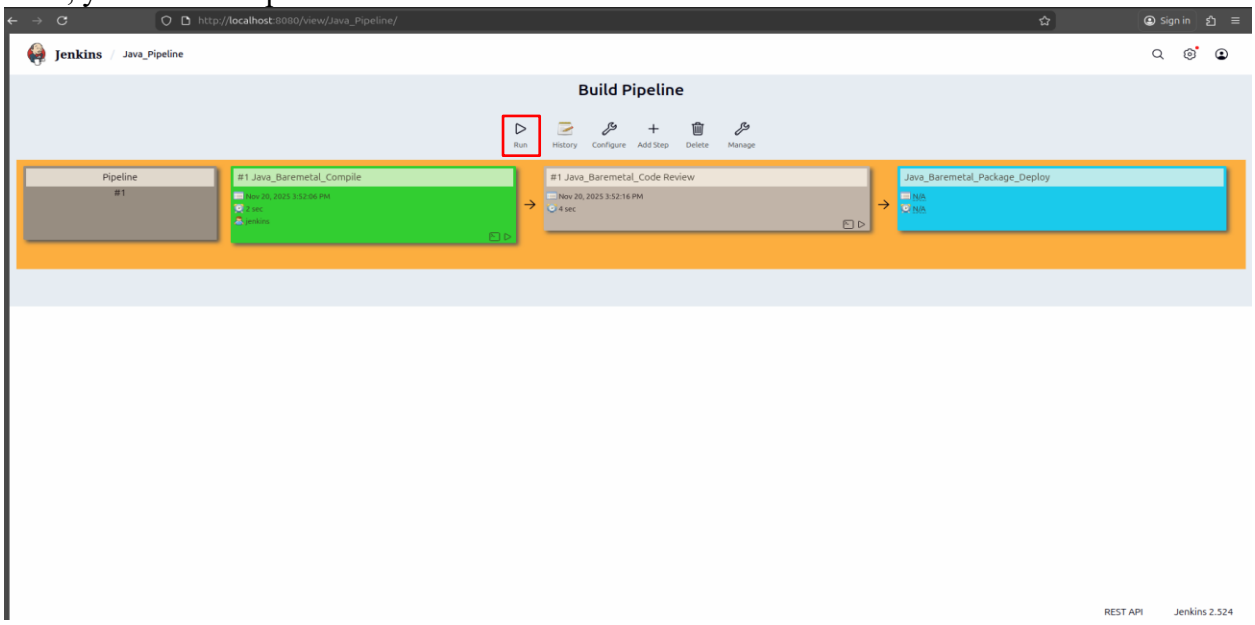
Provide a pipeline name (e.g., **Java_Pipeline**), select **Build Pipeline View** under **Type**, and then click **Create**.

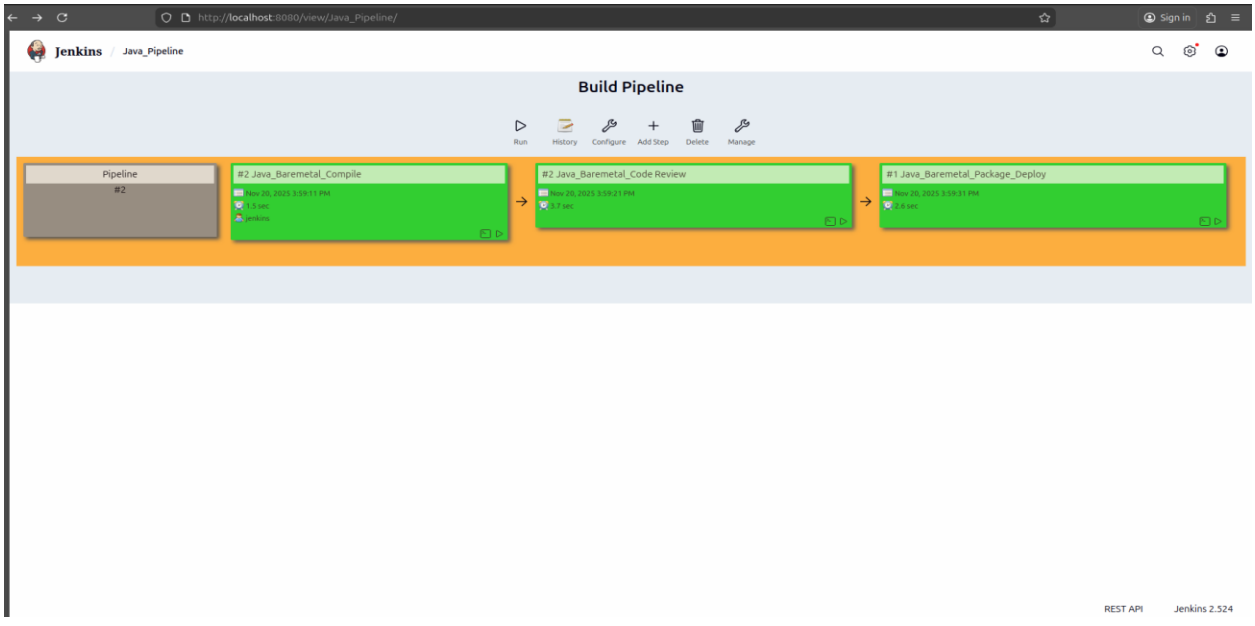


On the next page, under the **Upstream/Downstream config** section, enter the first job name (**Java_Baremetal_compile**) in the **Select Initial Job** field, then click **Save**.



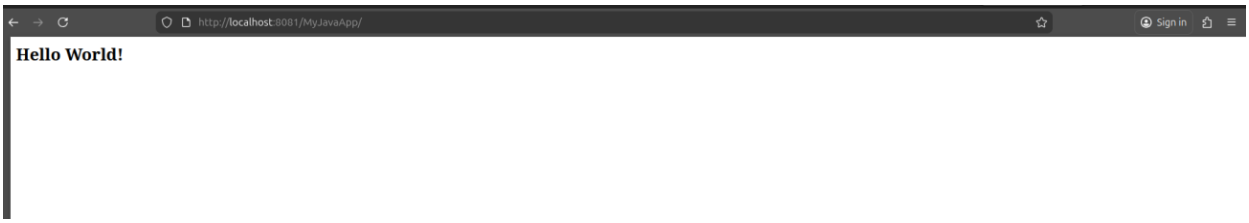
Now, you can see Pipeline with all the items. Click on Run button





Once, the Pipeline runs successfully we can access the page at

http://localhost:<Tomcat Port>/<Context Path> (for e.g http://localhost:8081/ MyJavaApp)



****

2: Containerizing the Java Application using Docker

Plugins: Install Docker relevant plugins

Below is the Dockerfile for Java tomcat application #Stage

1: Build WAR using Maven

FROM maven:3.9-eclipse-temurin-17 AS build WORKDIR /app COPY . . RUN mvn clean package -DskipTests

#Stage 2: Deploy to Tomcat FROM

tomcat:9.0-jdk17-temurin

RUN rm -rf /usr/local/tomcat/webapps/*

COPY --from=build /app/target/*.war /usr/local/tomcat/webapps/ROOT.war

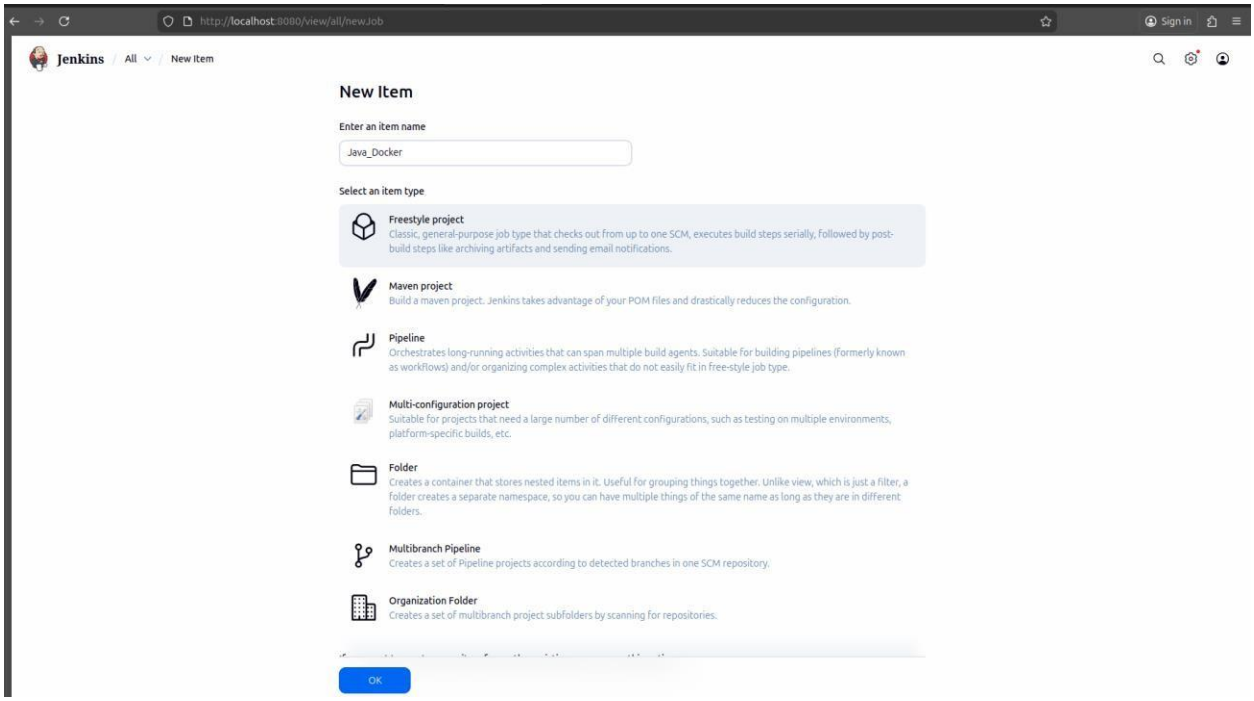
EXPOSE 8080

CMD ["catalina.sh", "run"]

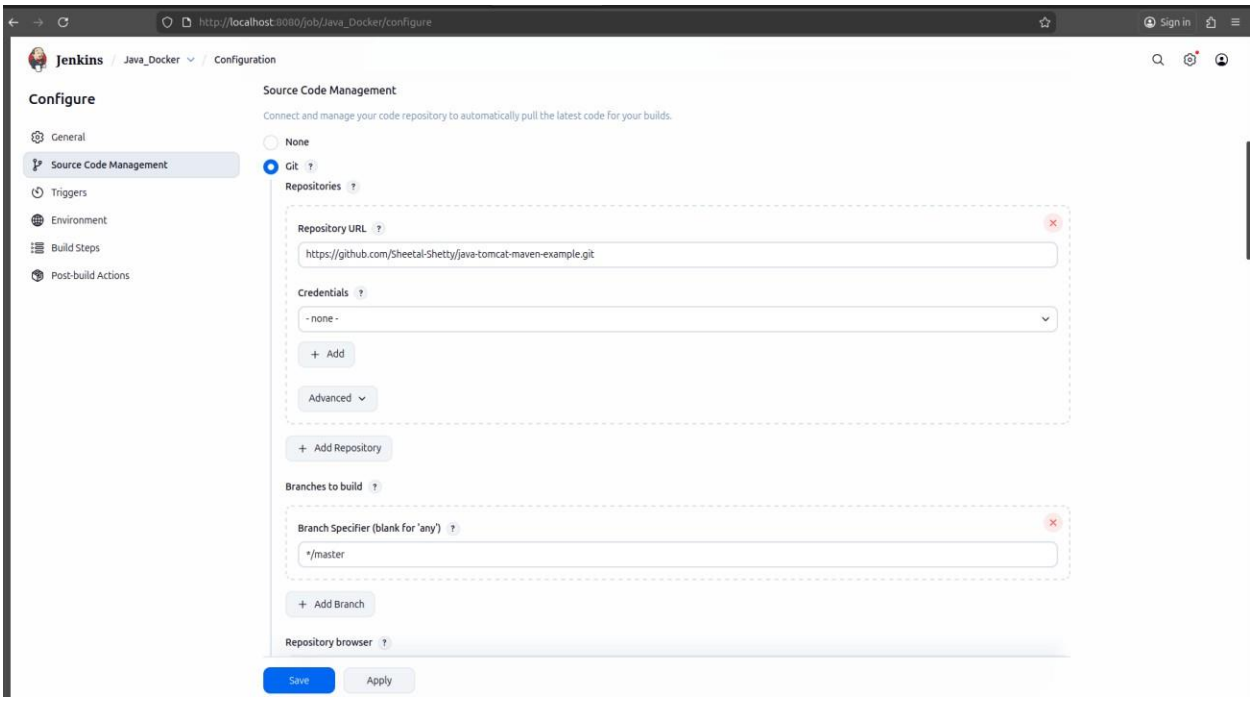
Pushed sir's code along with the Dockerfile to the new repository.

GitHub Repo: https://github.com/Sheetal-Shetty/java-tomcat-maven-example.git Create a

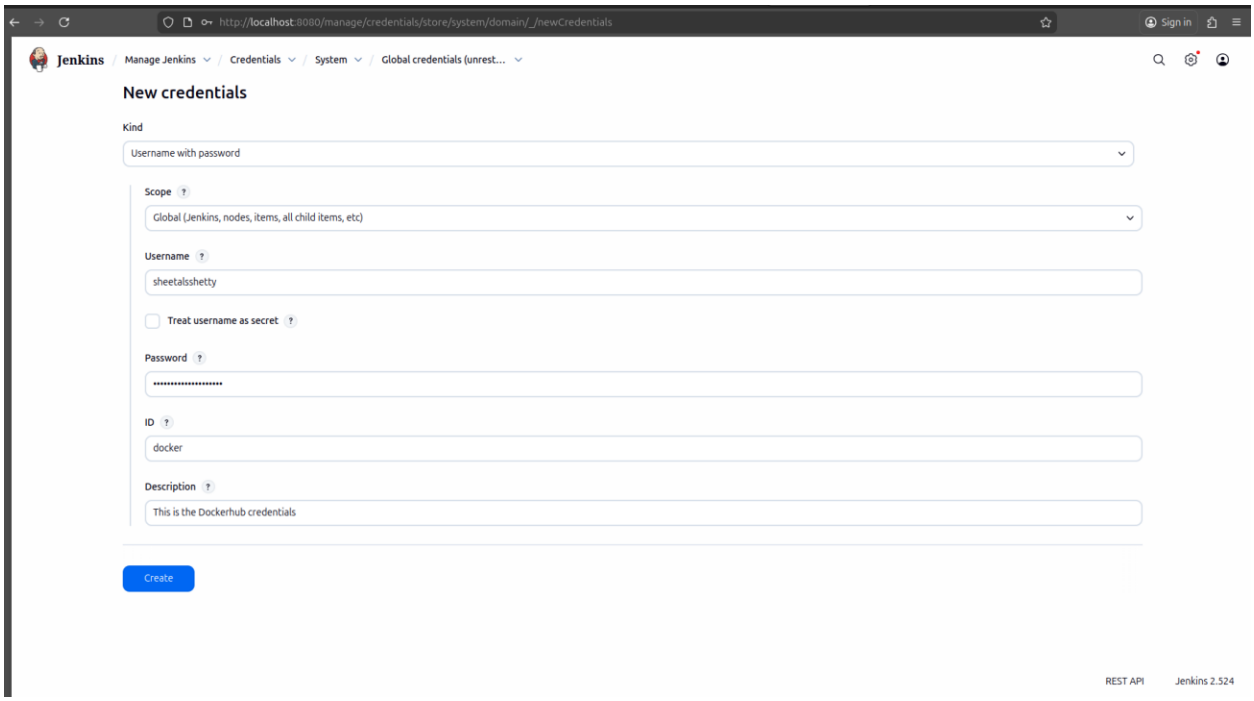
Freestyle Jenkins Project and provide a name for the project, for example **Java_Docker**, then click **OK**.



On the next page, select **Git** under **Source Code Management**, then enter the GitHub repository URL in the **Repository URL** field and specify the branch.

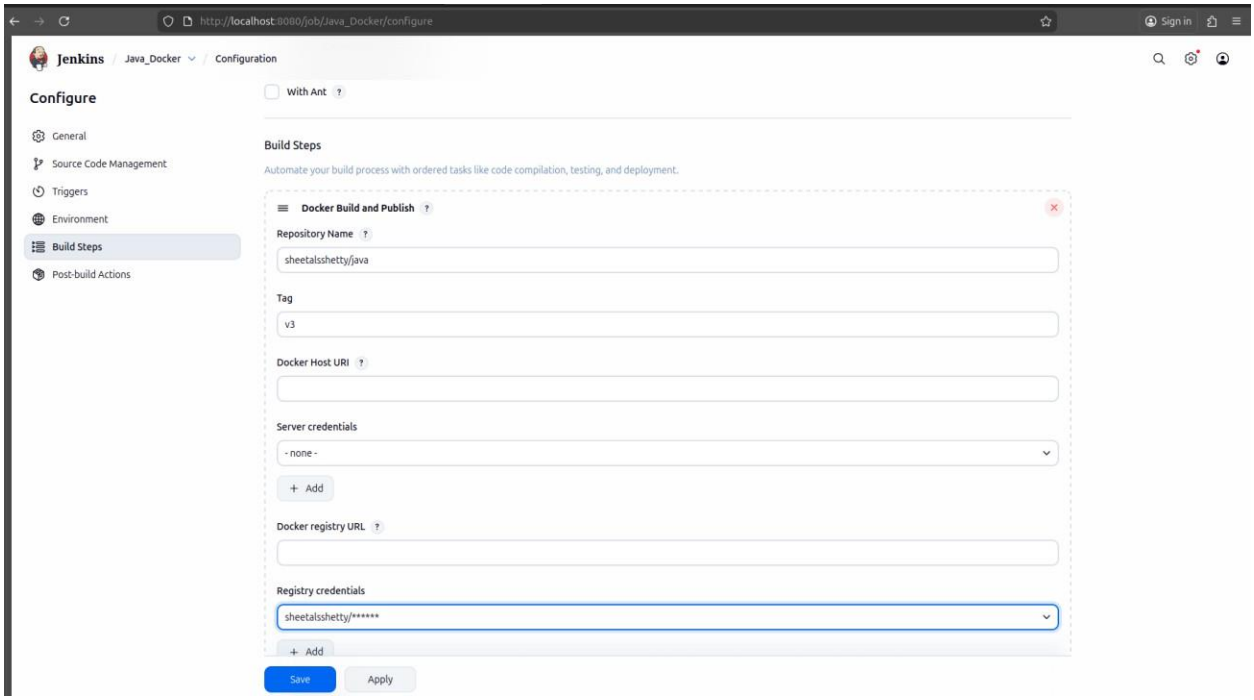


Add Docker credentials to Jenkins by navigating to **Manage Jenkins → Credentials**. Use your Docker Hub username as the **Username** and your Docker Hub access token as the **Password**.

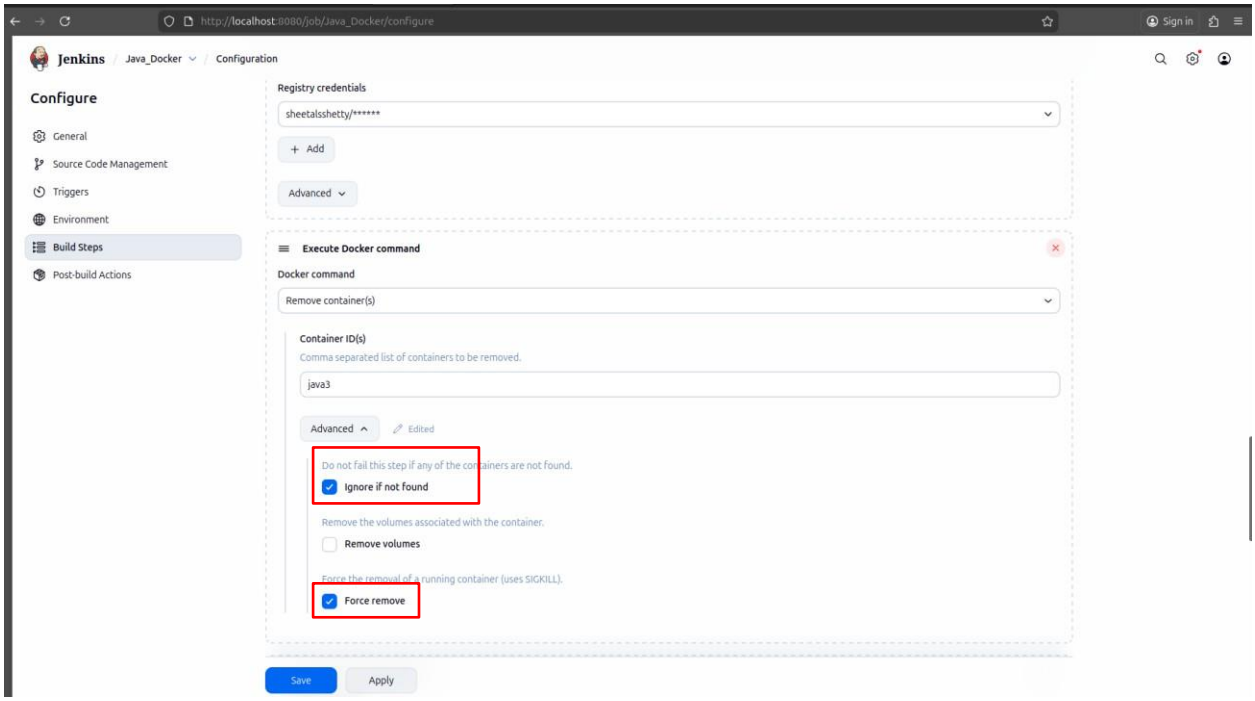


Under **Build Steps**, select **Docker Build and Publish**.
For **Repository Name**, enter <DockerHub_Username>/<image_name>.

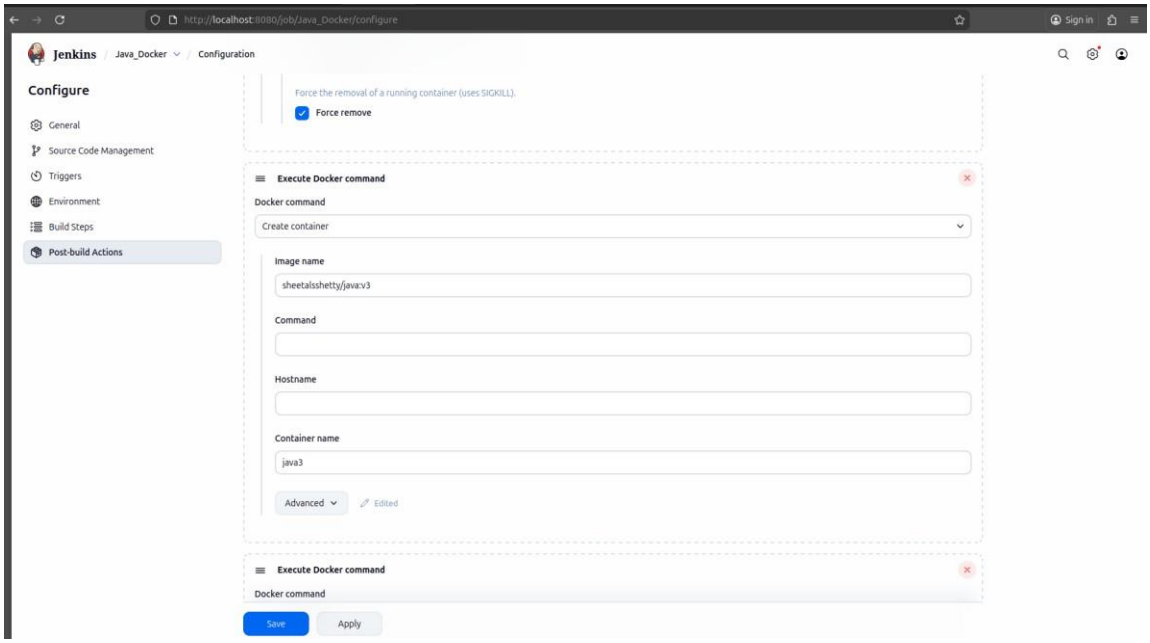
Set the **Tag** (e.g., v3 or any version you prefer), and under **Registry credentials**, select the Docker credential ID.

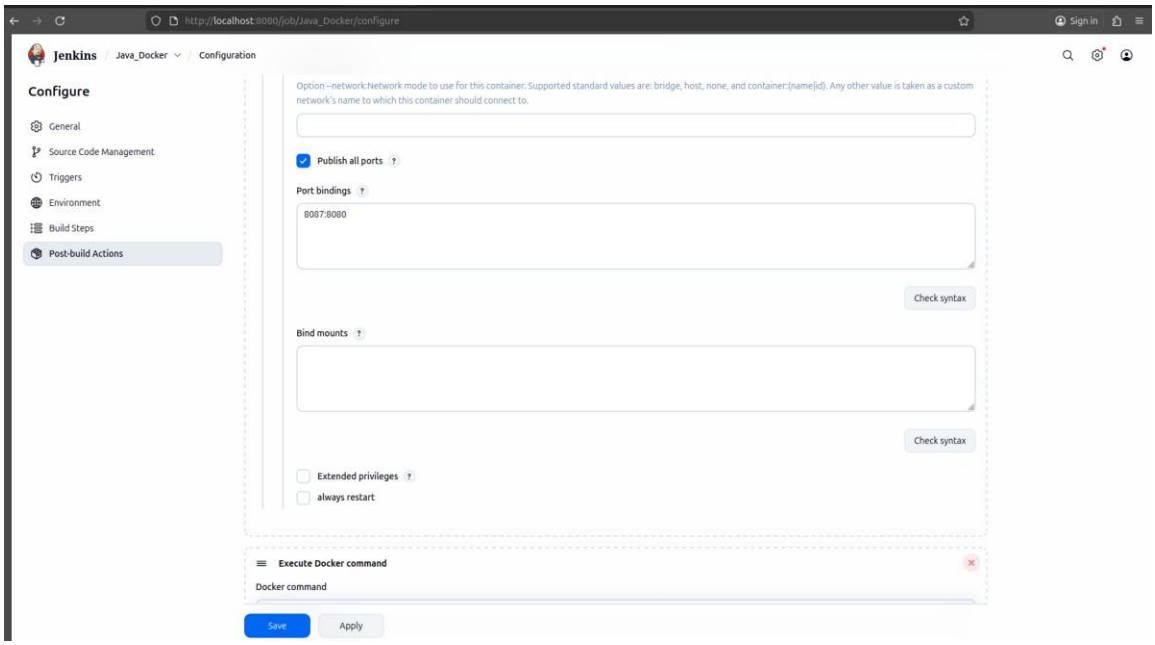


Click on **“+ Add Build Step”** to add another build step and select **“Execute Docker command.”** Under **Docker command**, choose **“Remove container(s).”** In the **Container ID(s)** field, enter the name of the container you plan to create (e.g., java3). Also select the checkboxes for **“Ignore if not found”** and **“Force remove.”**

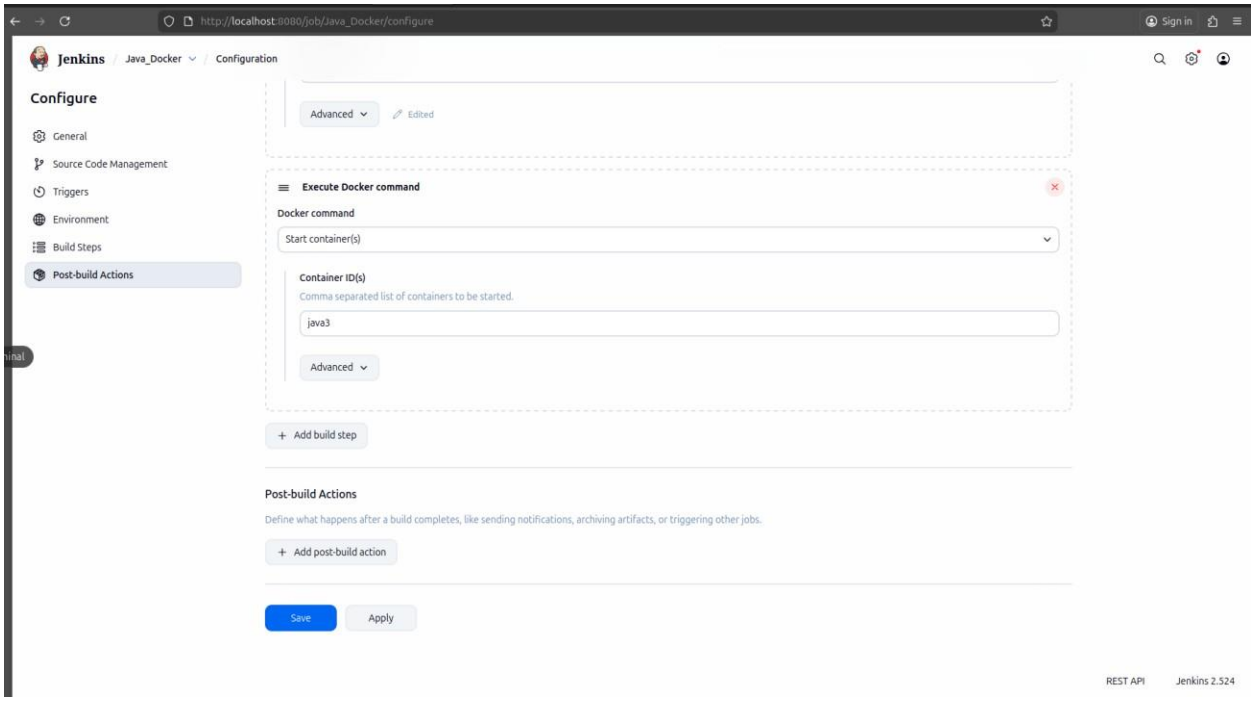


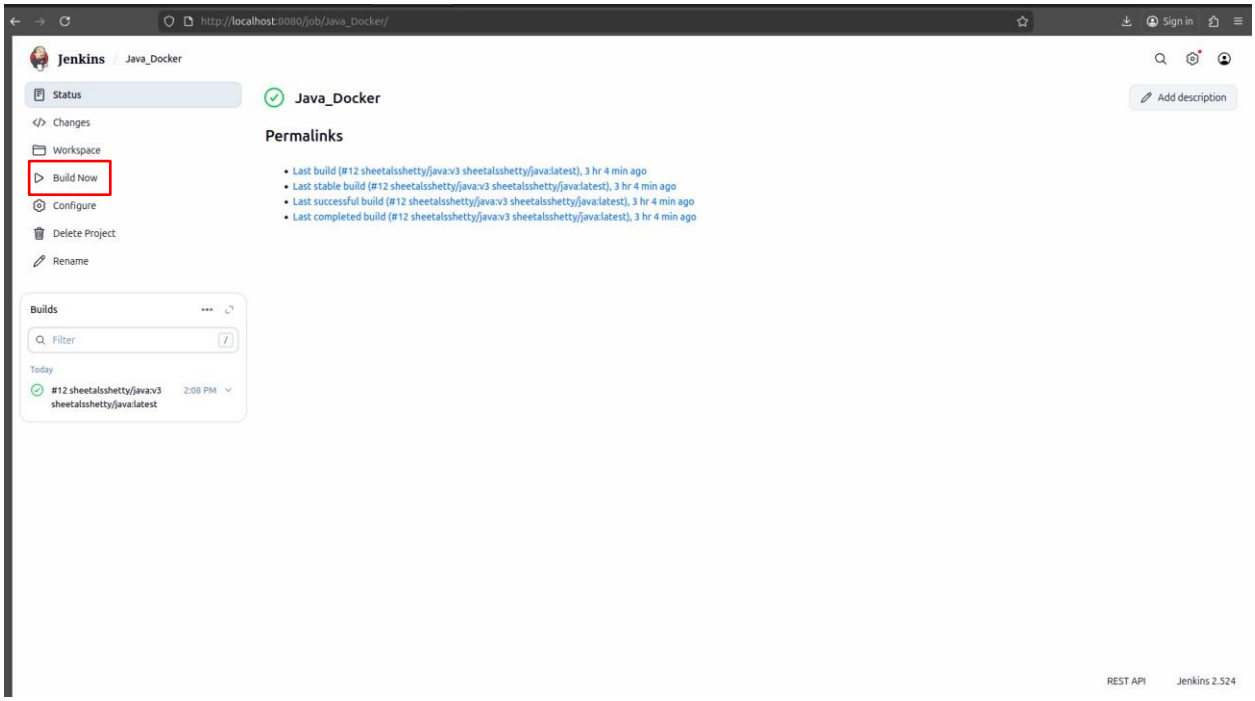
Click on **“+ Add Build Step”** to add another build step and select **“Execute Docker command.”**
Under **Docker command**, choose **“Create container.”**
In the **Image name** field, enter the name of the image you created in the previous build step along with its version (e.g., sheetalsshetty/java:v3).
In the **Container name** field, provide the name of the container you want to create (e.g., java3).
Click **Advanced**, select **“Publish all ports,”** and under **Port bindings**, map the ports in the format **Host_port:Container_port** (e.g., 8089:8080).



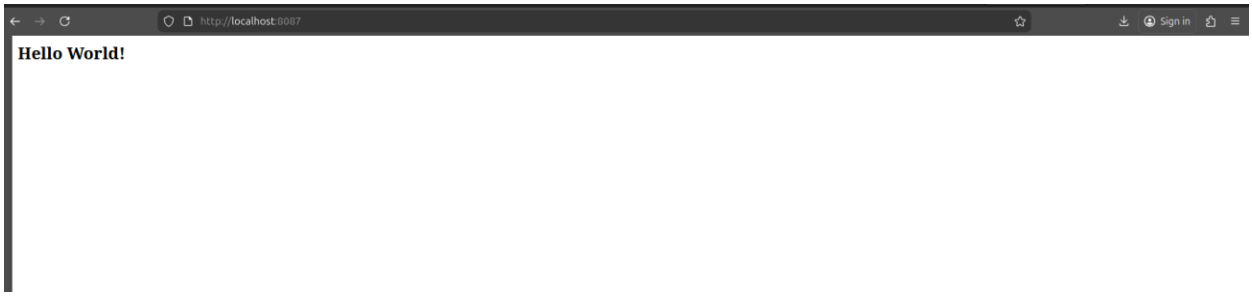


Click on **“+ Add Build Step”** to add another build step and select **“Execute Docker command.”** Under **Docker command**, choose **“Start container(s)”** and provide the name of the container you created in the **Container ID(s)** field (e.g., java3). Then click **Save**.





we can access the web page at http://localhost:<Host Port>/

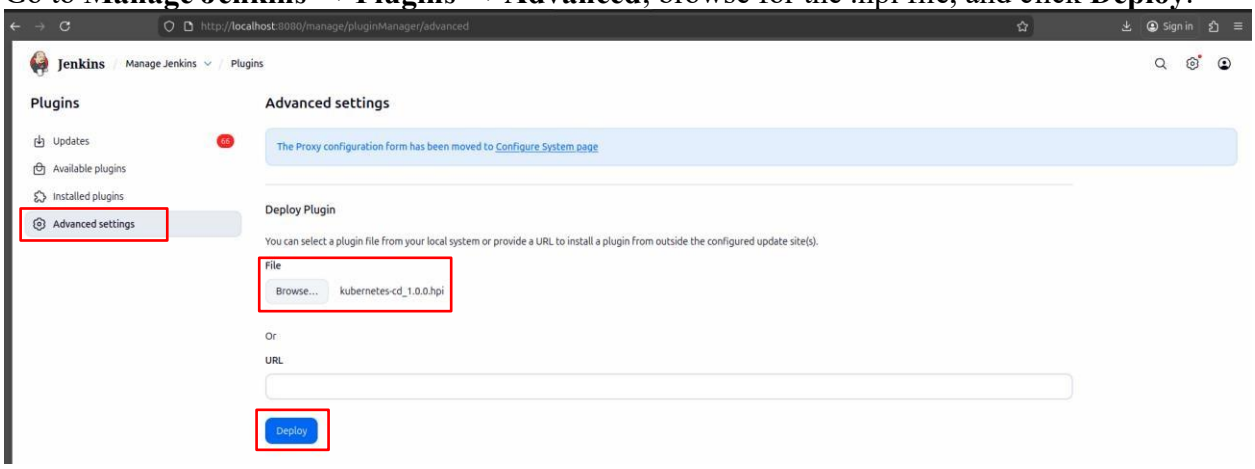


****

3. Deploy containerized application on Kubernetes cluster environment – Java

Plugins: Install the Kubernetes plugin using sir's kubernetes-cd_1.0.0.hpi file (available in the GitHub repository linked below).

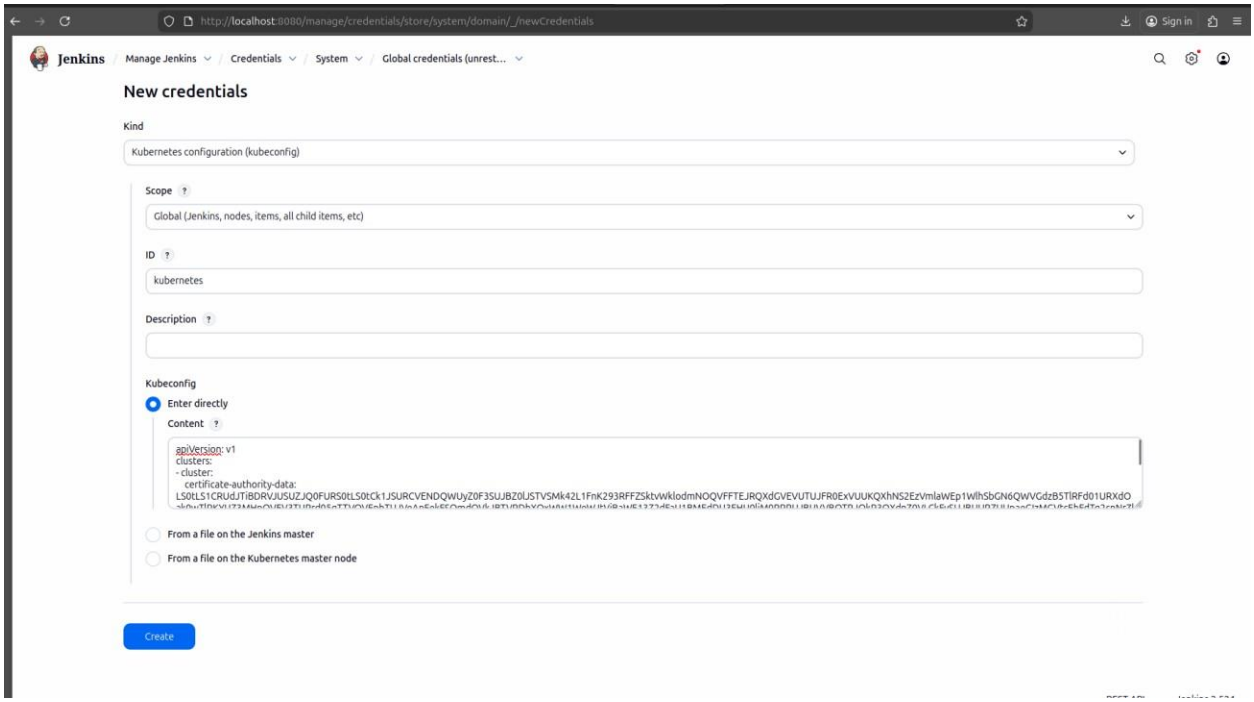
Go to **Manage Jenkins → Plugins → Advanced**, browse for the .hpi file, and click **Deploy**.



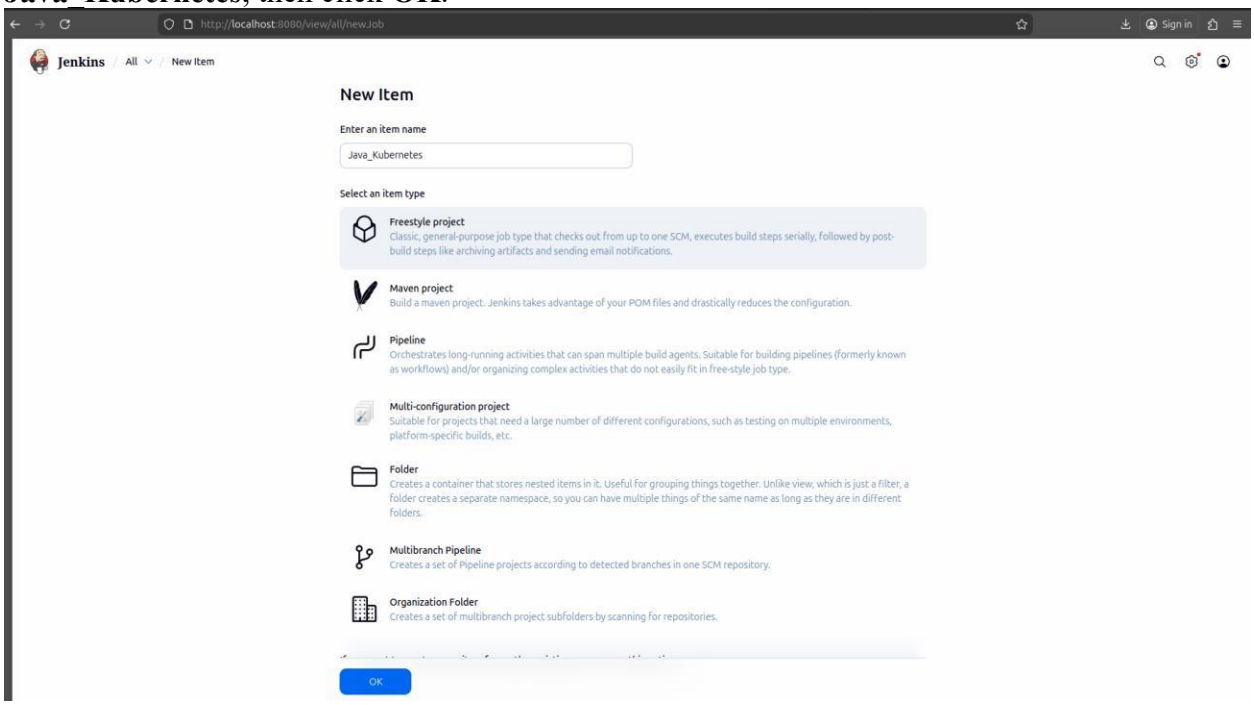
Create the Kubernetes manifest files (deployment.yaml and service.yaml) and add them to the GitHub repository.

GitHub Repo: https://github.com/Sheetal-Shetty/java-tomcat-maven-example.git

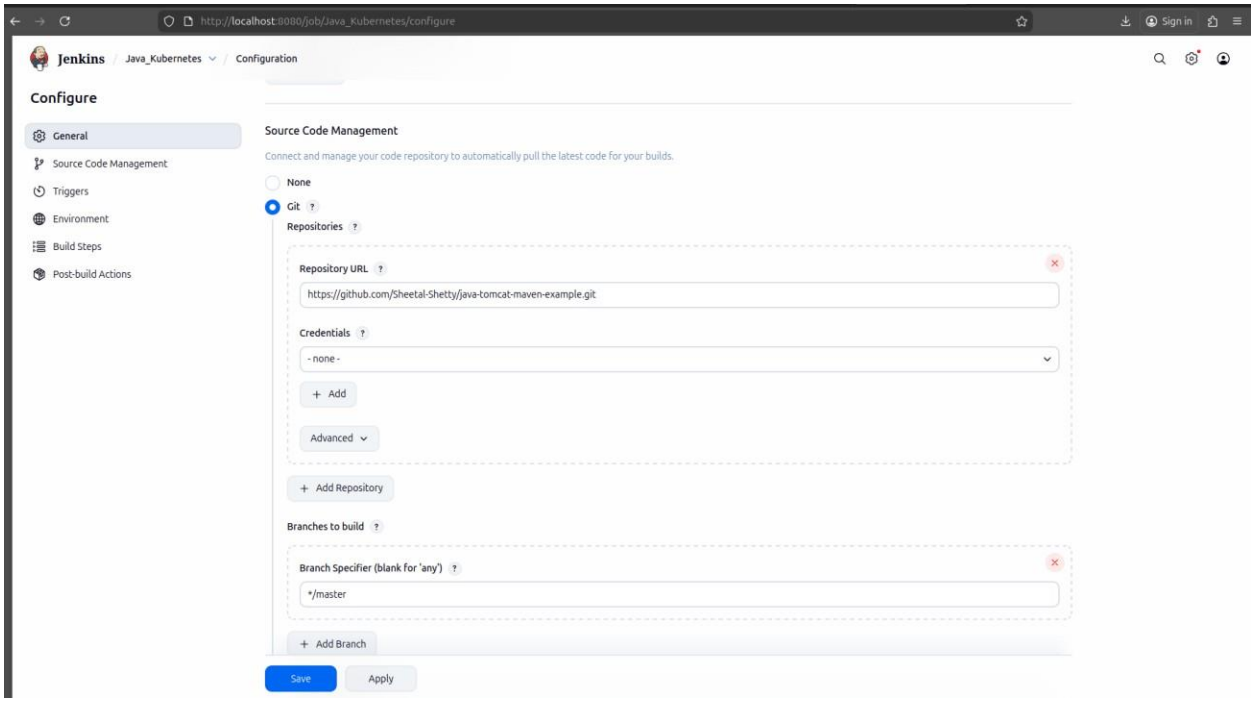
Add Kubernetes credentials to Jenkins by navigating to **Manage Jenkins → Plugins**. Then, under **Kind**, select **"Kubernetes configuration (kubeconfig)"** and provide a name for the credential under **ID**. For **Kubeconfig**, choose **"Enter directly"** and paste the output of cat ~/.kube/config from the master node. Finally, click **Create**.



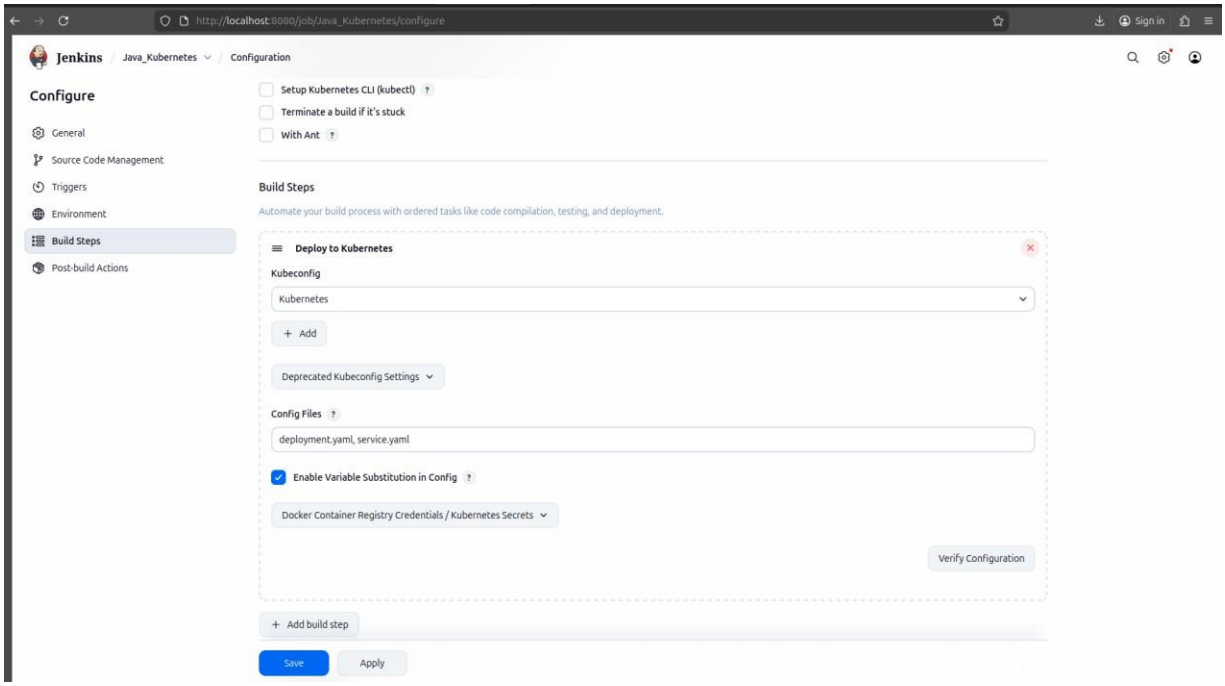
Create a **Freestyle Jenkins Project** and provide a name for the project, for example **Java_Kubernetes**, then click **OK**.

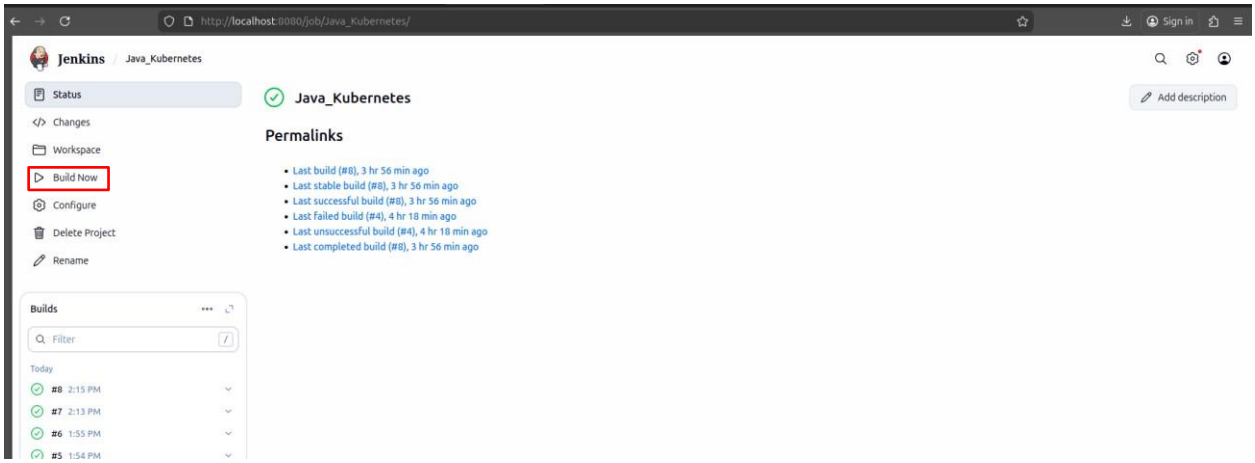


On the next page, select **Git** under **Source Code Management**, then enter the GitHub repository URL in the **Repository URL** field and specify the branch.



Under the **Build** step, select **"Deploy to Kubernetes."** Then choose the Kubernetes credential ID you added under **Kubeconfig**. In the **Config Files** section, specify the names of your deployment and service files (e.g., deployment.yaml, service.yaml). Finally, click **Save** and then **Build**.

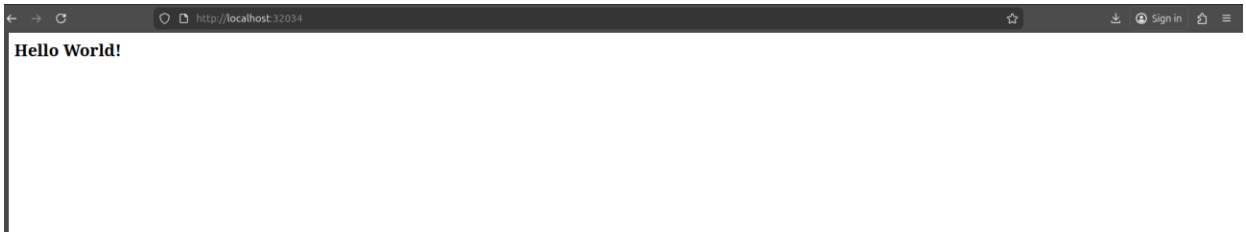




In terminal type kubectl get svc

```
msis@msis:~$ kubectl get svc javamaven
NAME        TYPE       CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
javamaven   NodePort   10.104.151.252   <none>        8081:32034/TCP   69s
msis@msis:~$
```

We can access the web page at,



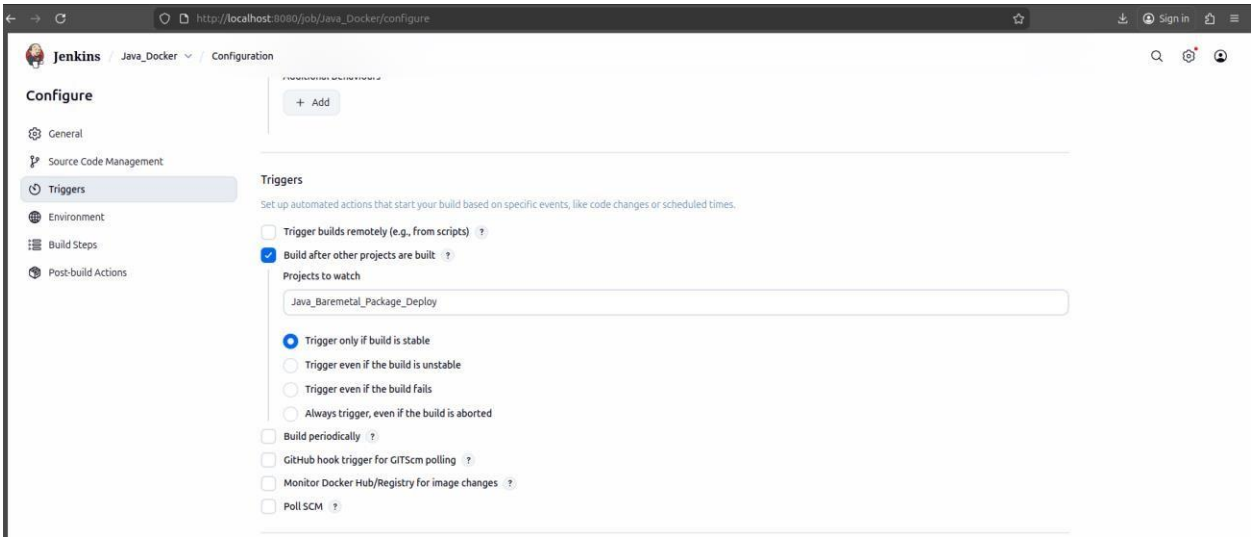
http://localhost:<NodePort> or http://<cluster ip>:<Target Port>

http://localhost:32034/

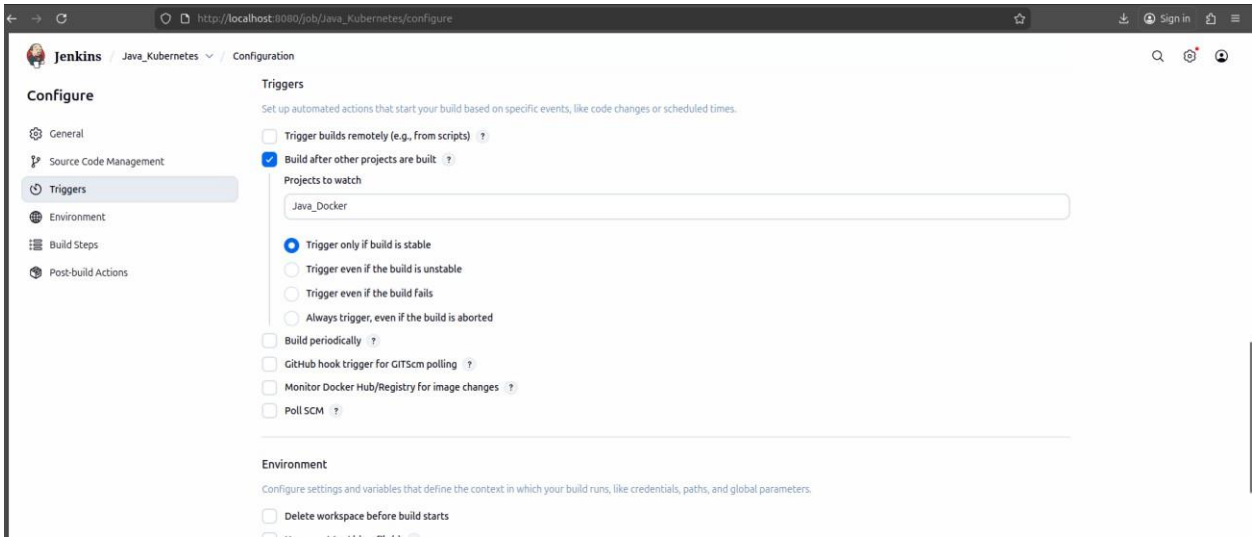
****

Create Pipeline for all these items (Baremetal->Docker->Kubernetes)

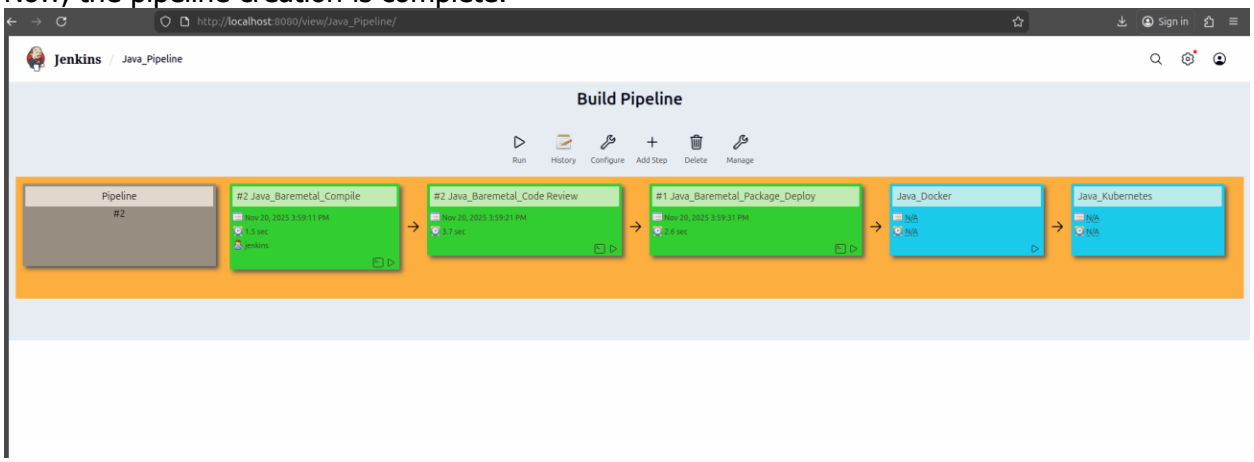
Since we have already created the pipeline for Java_Baremetal_Compile → Java_Baremetal_CodeReview → Java_Baremetal_Package_Deploy, we now need to add the The remaining two stages (Java_Docker and Java_Kubernetes).
Go to the item named Java_Docker. Under Triggers, select “Build after other projects are built.” Then, under Projects to watch, select Java_Baremetal_Package_Deploy.



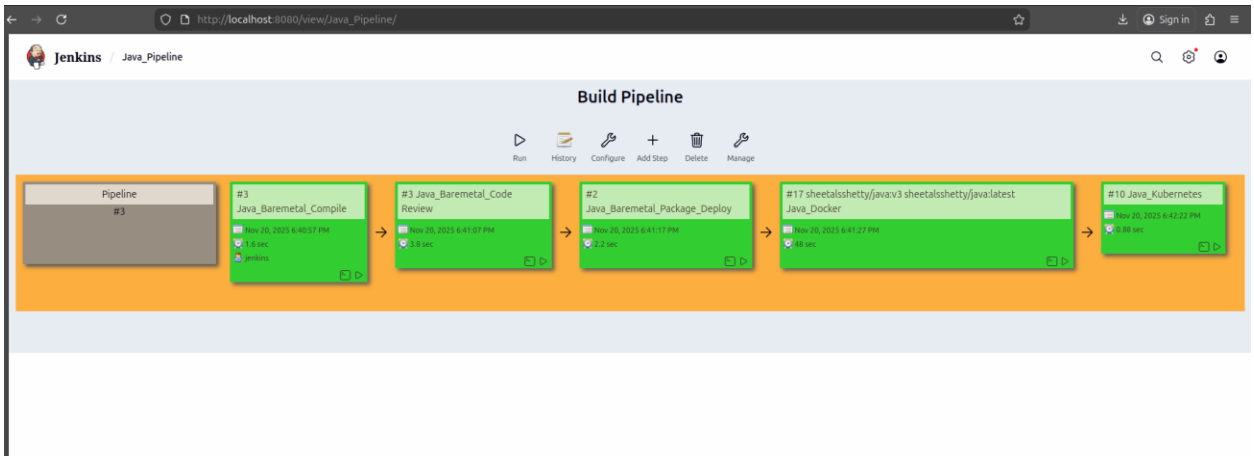
Go to the item named Java_Baremetal_Package_Deploy. Under Triggers, select "Build after other projects are built." Then, under Projects to watch, select Java_Docker.



Now, the pipeline creation is complete.



Run the Pipeline.



For php

1.Implement CI/CD Build Pipeline for php Application on Jenkins Server

Plugins required:

-Deploy to kubernetes

- Deploy to container

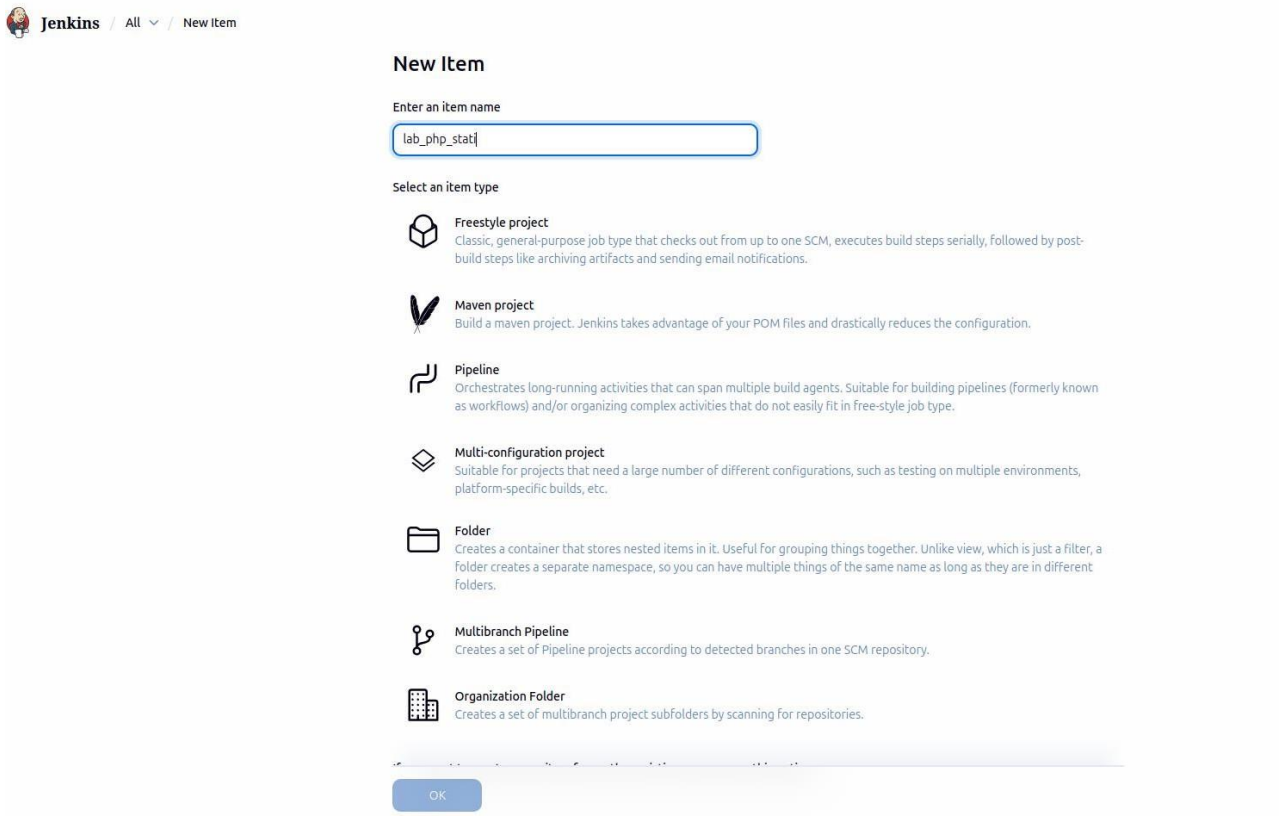
The above plugins can be installed by navigating to Manage Jenkins->Plugins.

GitHub Repo: https://github.com/saivasanth26/php-static-website.git

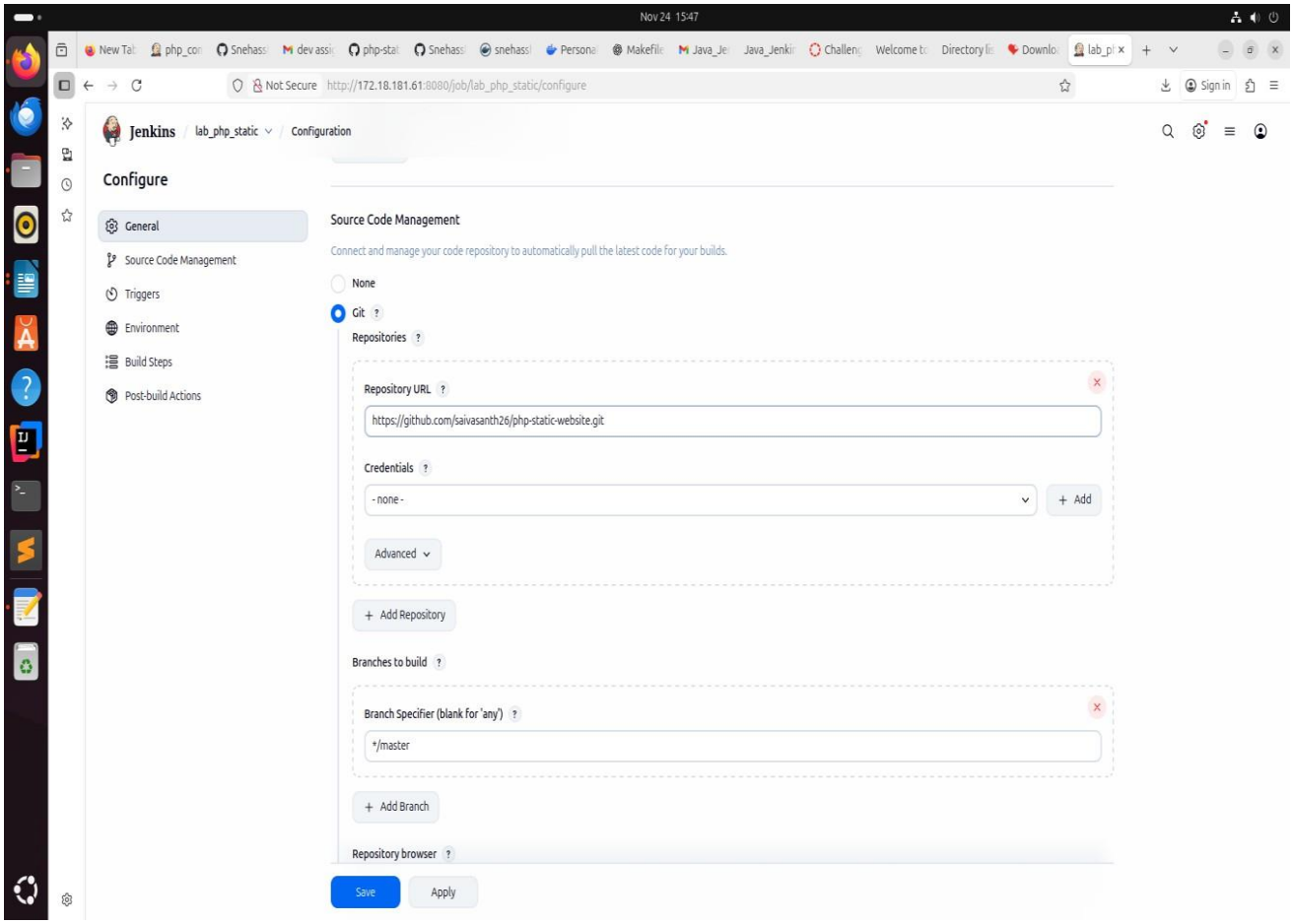
Log in to Jenkins Server.

Create a Free Style Jenkins Project.

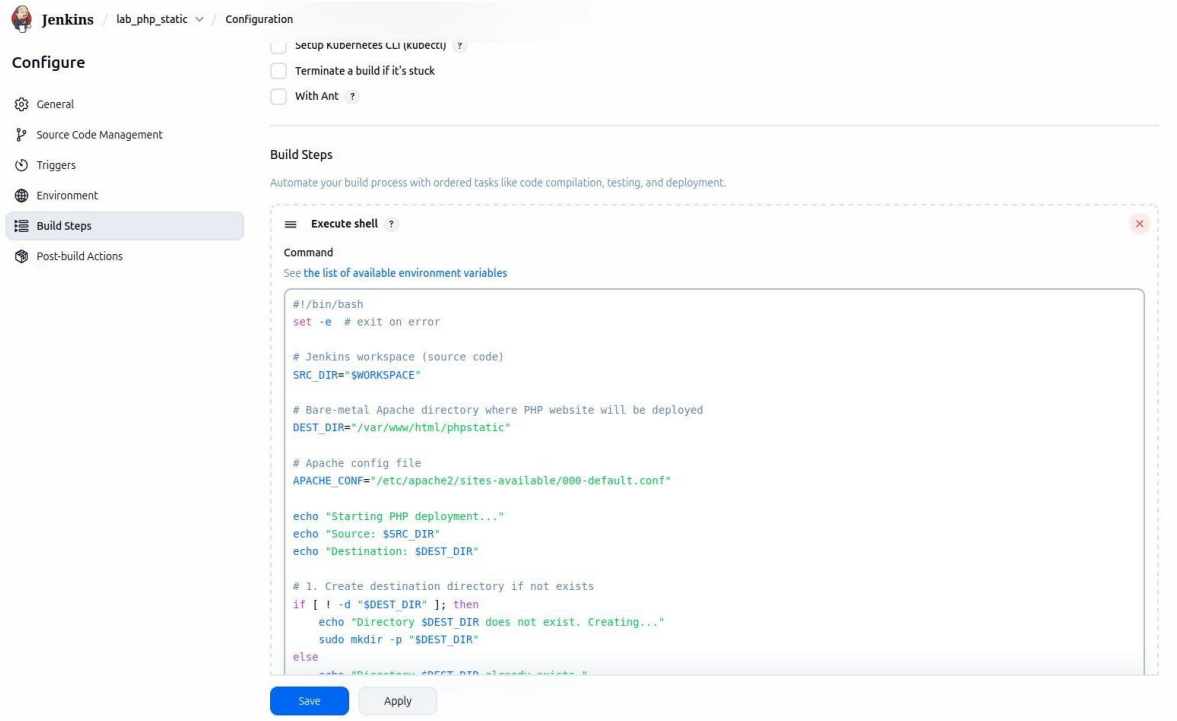
Provide a name to Project for Example: lab_php_static compile then click Ok.

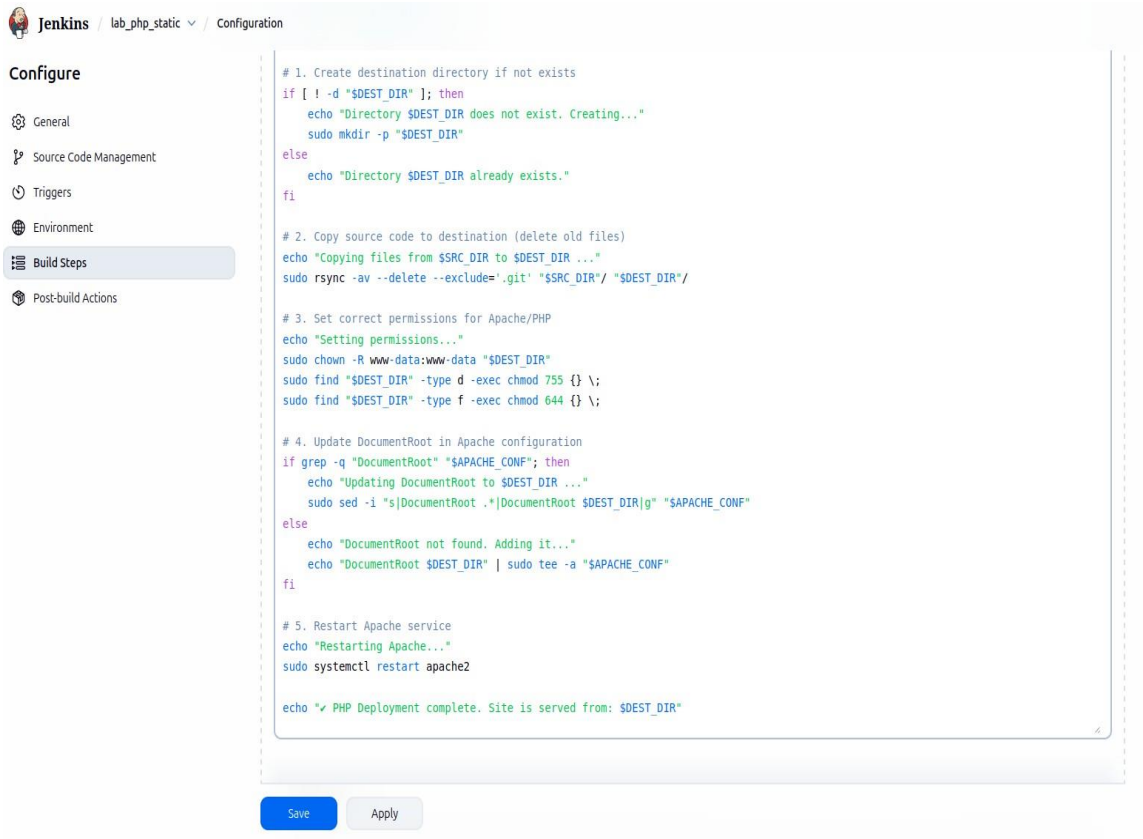


On the next page, select Source Code Management as Git, then provide the GitHub repository URL under Repository URL and specify the branch.

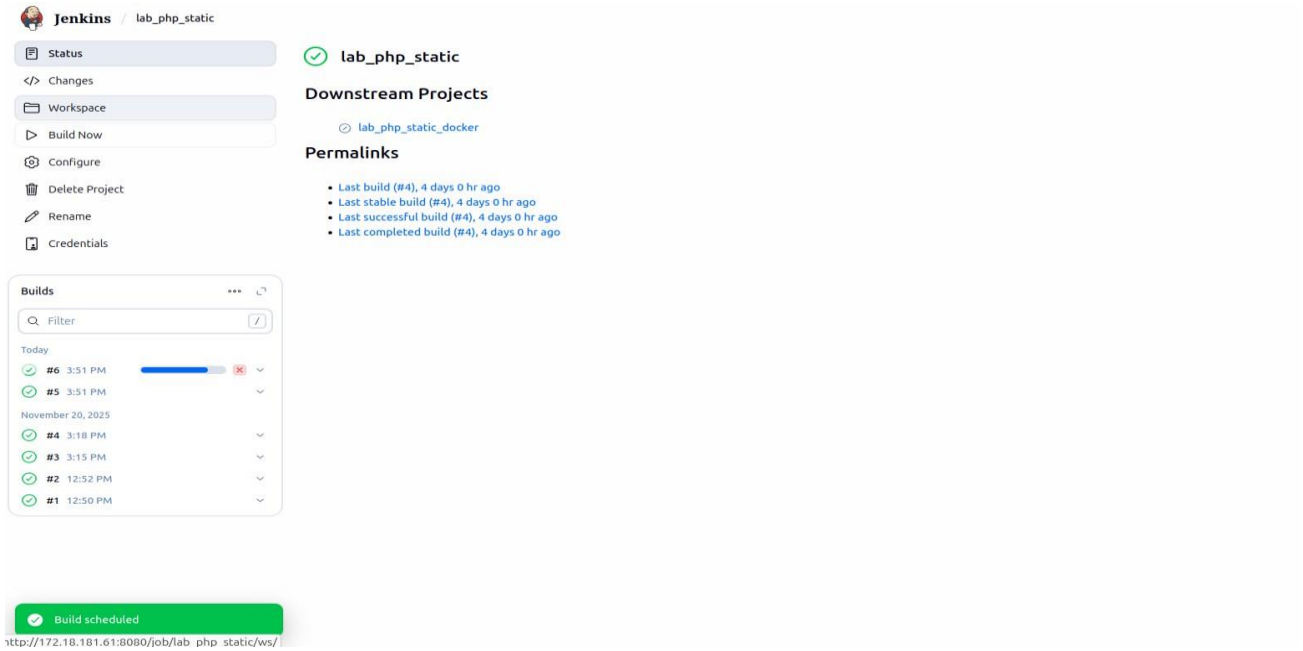


Under the **Build** step, select **Execute shell** and paste the shell script . Then click **Save.**

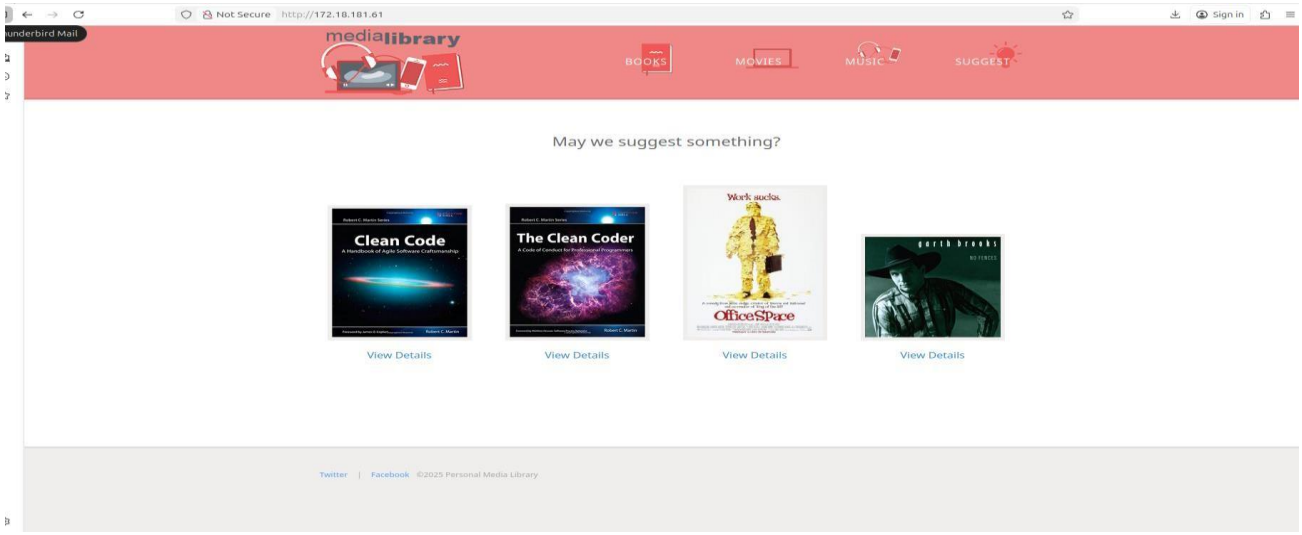

Jenkins / lab_php_static / Configuration
Configure
General
Source Code Management
Triggers
Environment
Build Steps
Post-build Actions
Terminate a build if it's stuck
With Ant
Build Steps
Automate your build process with ordered tasks like code compilation, testing, and deployment.
Execute shell
Command
See the list of available environment variables
#!/bin/bash
set -e # exit on error
# Jenkins workspace (source code)
SRC_DIR="$WORKSPACE"
# Bare-metal Apache directory where PHP website will be deployed
DEST_DIR="/var/www/html/phpstatic"
# Apache config file
APACHE_CONF="/etc/apache2/sites-available/000-default.conf"
echo "Starting PHP deployment..."
echo "Source: $SRC_DIR"
echo "Destination: $DEST_DIR"
# 1. Create destination directory if not exists
if [! -d "$DEST_DIR"]; then
echo "Directory $DEST_DIR does not exist. Creating..."
sudo mkdir -p "$DEST_DIR"
else
Save
Apply


Jenkins / lab_php_static / Configuration
Configure
General
Source Code Management
Triggers
Environment
Build Steps
Post-build Actions
# 1. Create destination directory if not exists
if [! -d "$DEST_DIR"]; then
echo "Directory $DEST_DIR does not exist. Creating..."
sudo mkdir -p "$DEST_DIR"
else
echo "Directory $DEST_DIR already exists."
fi
2. Copy source code to destination (delete old files)
echo "Copying files from $SRC_DIR to $DEST_DIR ..."
sudo rsync -av --delete --exclude='.git' "$SRC_DIR"/ "$DEST_DIR"/
# 3. Set correct permissions for Apache/PHP
echo "Setting permissions..."
sudo chown -R www-data:www-data "$DEST_DIR"
sudo find "$DEST_DIR" -type d -exec chmod 755 {} \;
sudo find "$DEST_DIR" -type f -exec chmod 644 {} \;
# 4. Update DocumentRoot in Apache configuration
if grep -q "DocumentRoot" "$APACHE_CONF"; then
echo "Updating DocumentRoot to $DEST_DIR ..."
sudo sed -i "s|DocumentRoot .*|DocumentRoot $DEST_DIR|g" "$APACHE_CONF"
else
echo "DocumentRoot not found. Adding it..."
echo "DocumentRoot $DEST_DIR" | sudo tee -a "$APACHE_CONF"
fi
5. Restart Apache service
echo "Restarting Apache..."
sudo systemctl restart apache2
echo "✔ PHP Deployment complete. Site is served from: $DEST_DIR"
Save
Apply

Under **Configure**. Click on **build now**



we can access the web page at http://localhost:<Host Port>/



2: Containerizing the php Application using Docker

Plugins: Install Docker relevant plugins

Below is the Dockerfile for phpapplication

```
FROM ubuntu:18.04
ARG DEBIAN_FRONTEND=noninteractive
#RUN sed -i
's/http:\/\/archive.ubuntu.com\/ubuntu\//https:\/\/arch ive.ubuntu.com\/ubuntu\//g'
/etc/apt/sources.list
RUN sed -i
's/http:\/\/archive.ubuntu.com\//http:\/\/us.archive.ub
untu.com\//g' /etc/apt/sources.list
RUN apt-get update -y && apt-get install -y apache2
RUN apt-get install -y php libapache2-mod-php COPY
000-default.conf /etc/apache2/sites- available/000-
default.conf
ADD . /var/www/html/
EXPOSE 80
ENTRYPOINT apachectl -D FOREGROUND
```

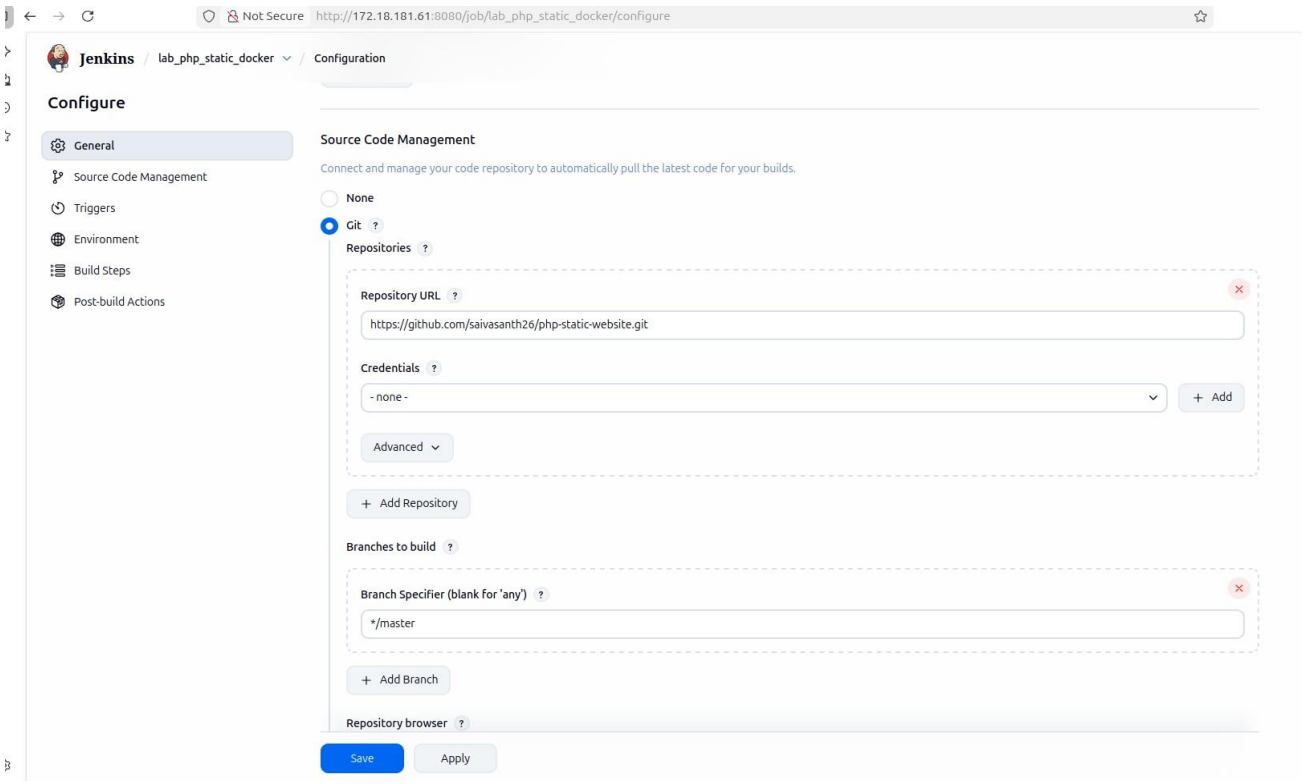
Create a **Freestyle Jenkins Project** and provide a name for the

project, for example

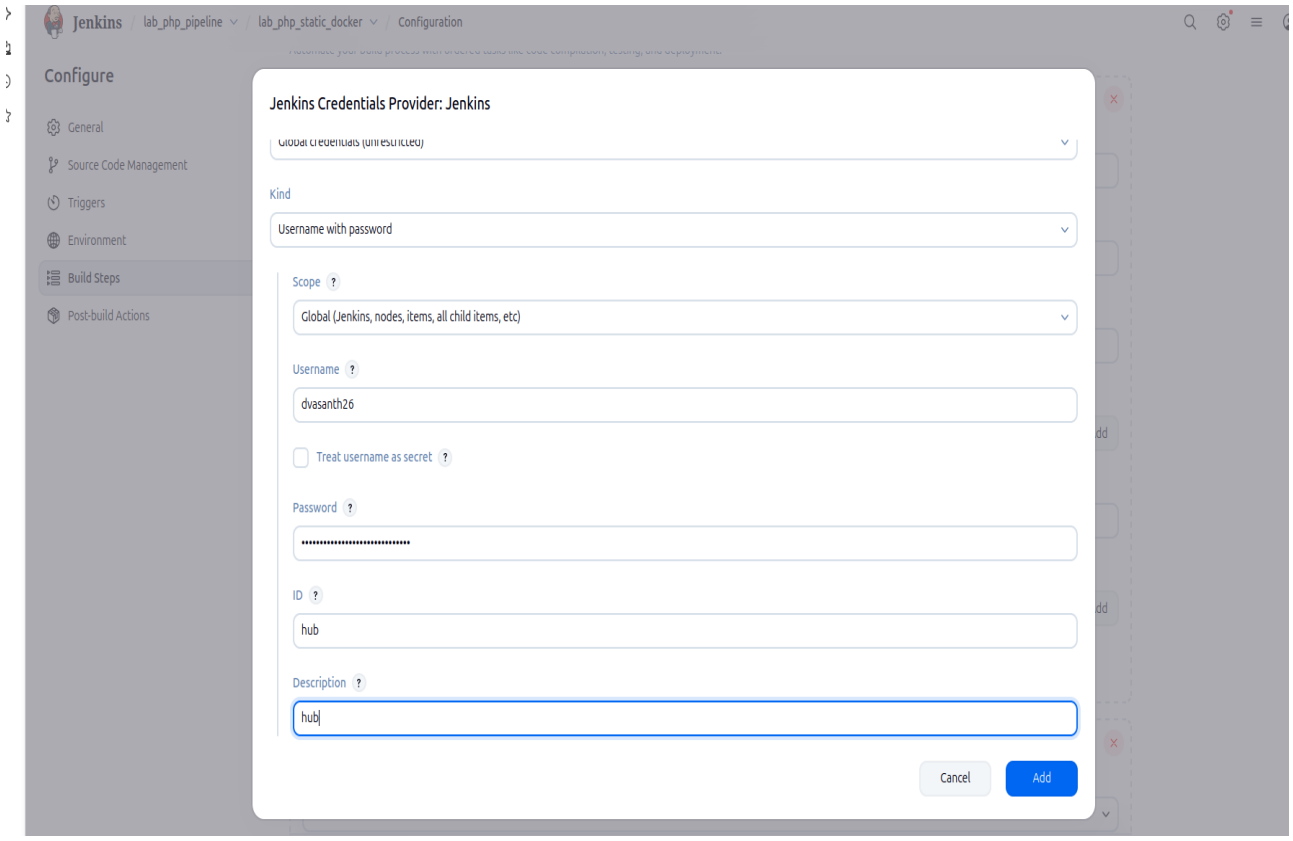
php_static_docker, then click **OK**.

On the next page, select **Git** under **Source Code Management**, then enter the GitHub repository URL in the

Repository URL field and specify the branch.



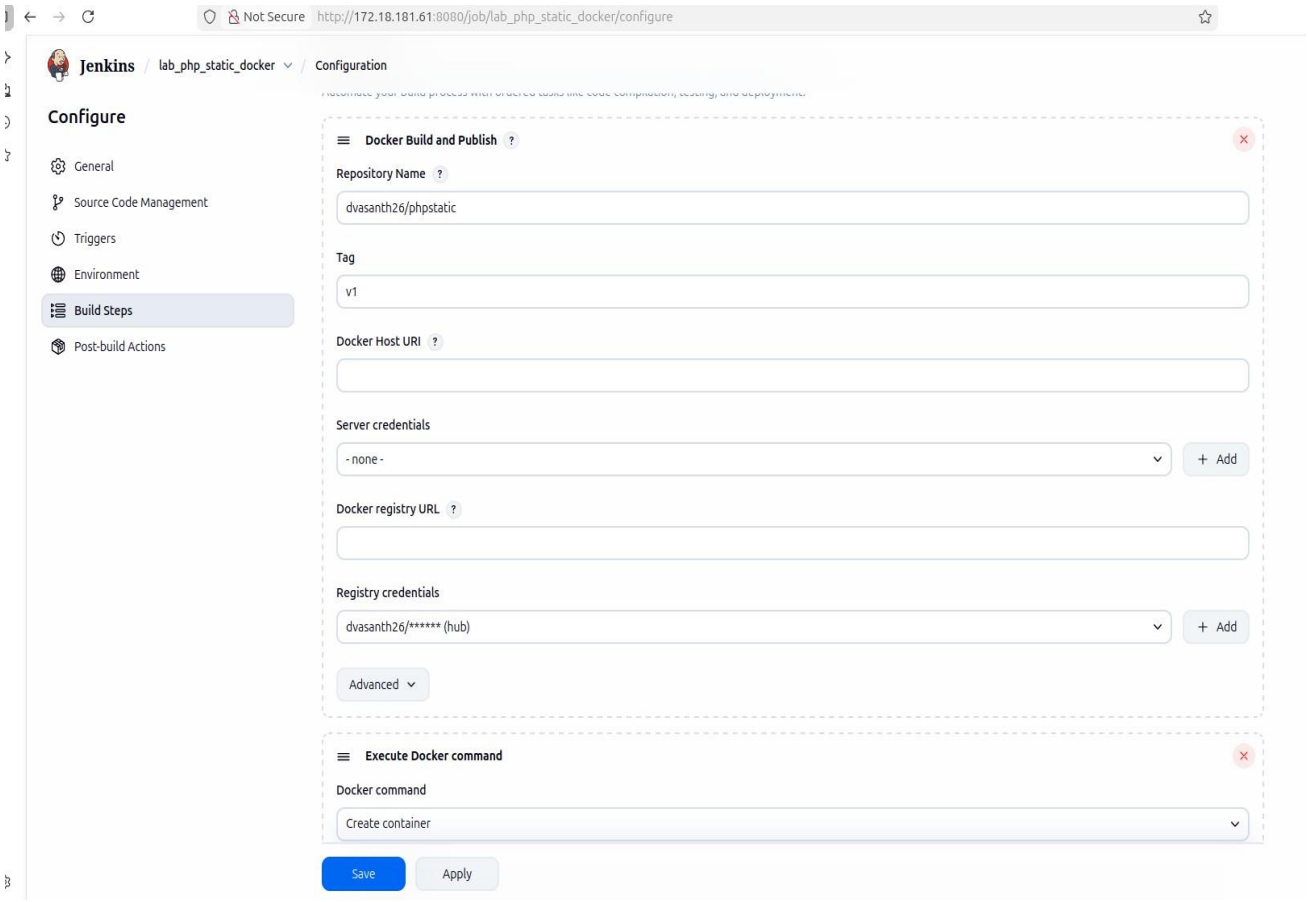
Add Docker credentials to Jenkins by navigating to **Manage Jenkins → Credentials**. Use your Docker Hub username as the **Username** and your Docker Hub access token as the **Password**.



Under **Build Steps**, select **Docker Build and Publish**.

For **Repository Name**, enter <DockerHub_Username>/<image_name>.

Set the **Tag** (e.g., v3 or any version you prefer), and under **Registry credentials**, select the Docker credential ID.



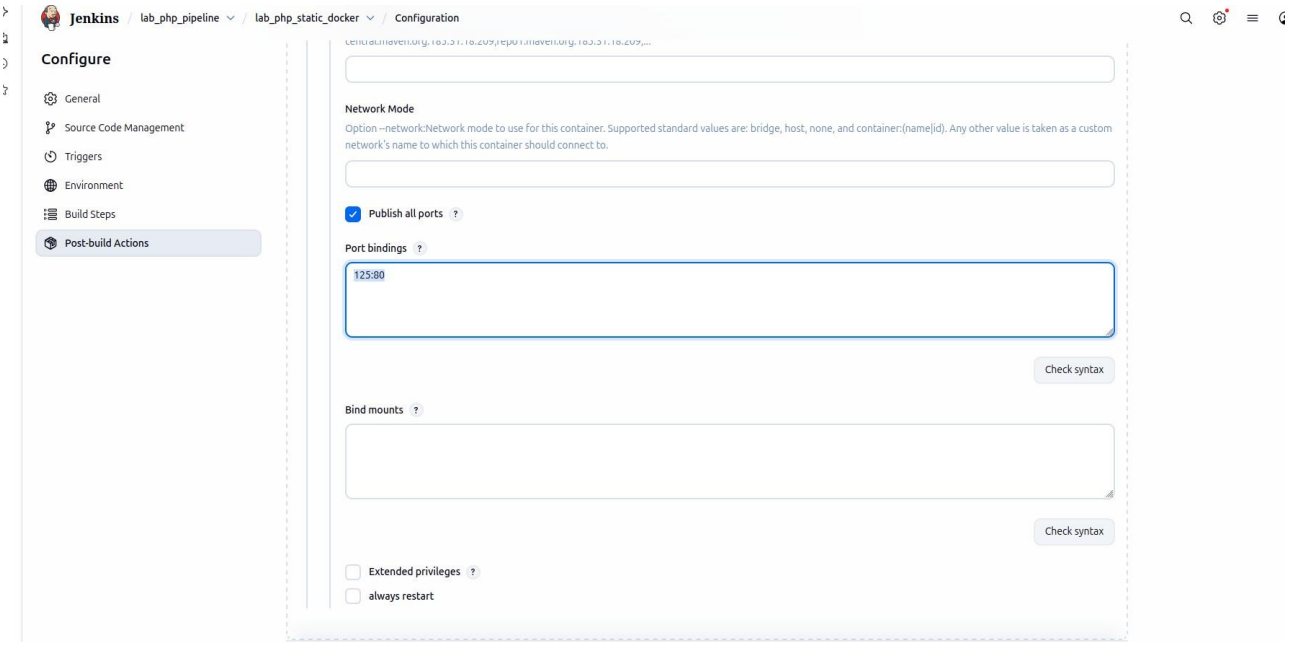
Click on **"+ Add Build Step"** to add another build step and select **"Execute Docker command."**

Under **Docker command**, choose **"Create container."**

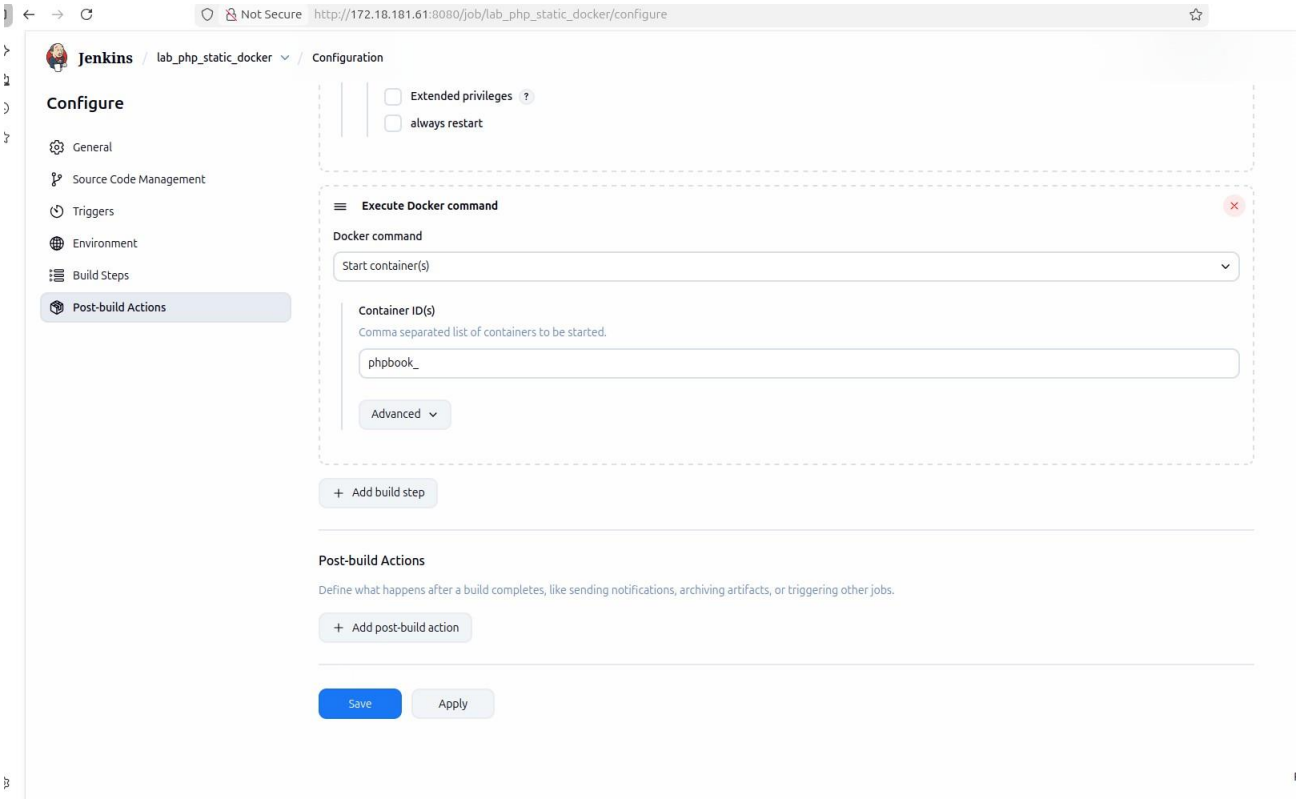
In the **Image name** field, enter the name of the image you created in the previous build step along with its version (e.g.dvasanth26/phpstatic:v1).

In the **Container name** field, provide the name of the container you want to create (e.g.phpbook).

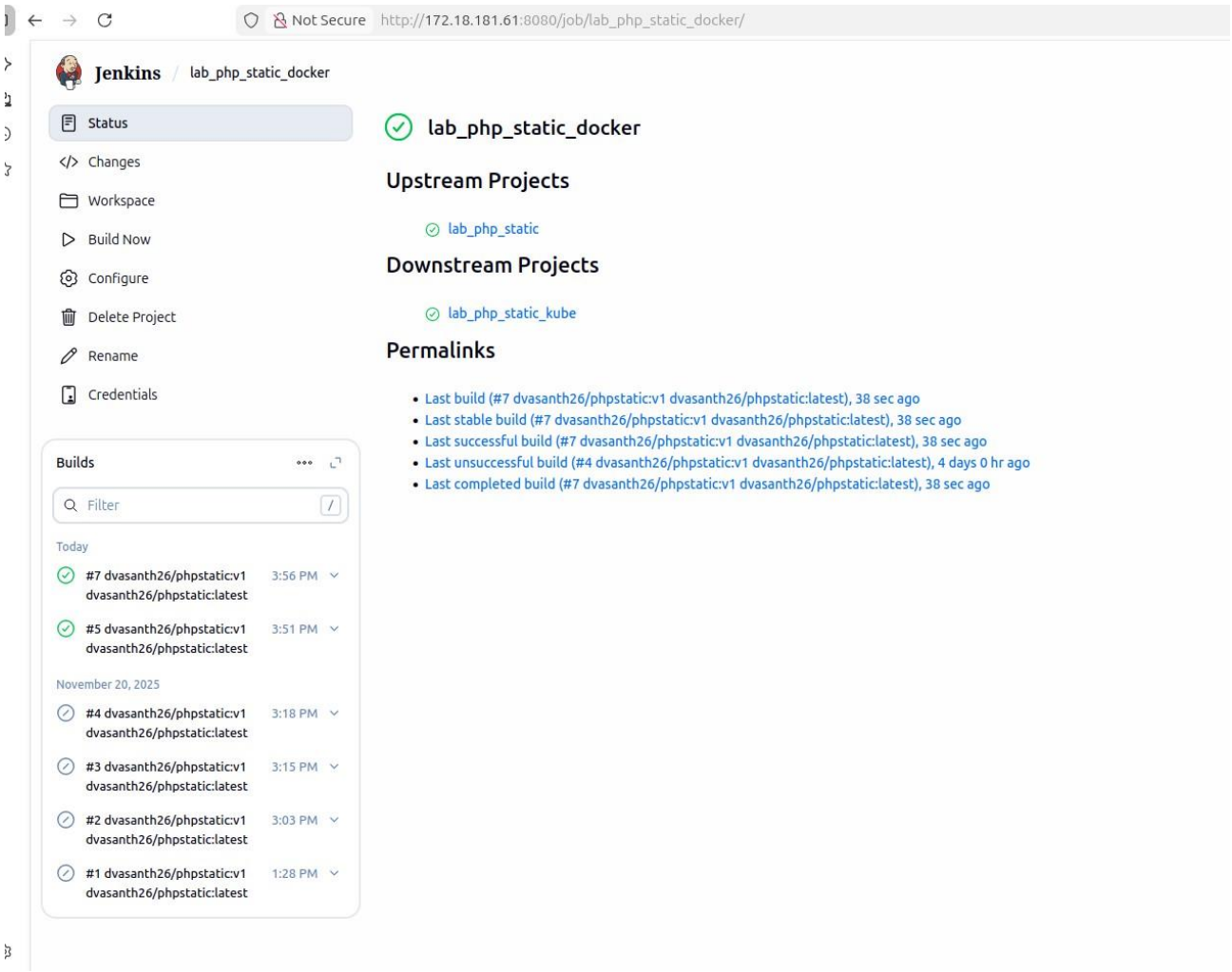
Click **Advanced**, select **"Publish all ports,"** and under **Port bindings**, map the ports in the format **Host_port:Container_port** (e.g., 125:80).



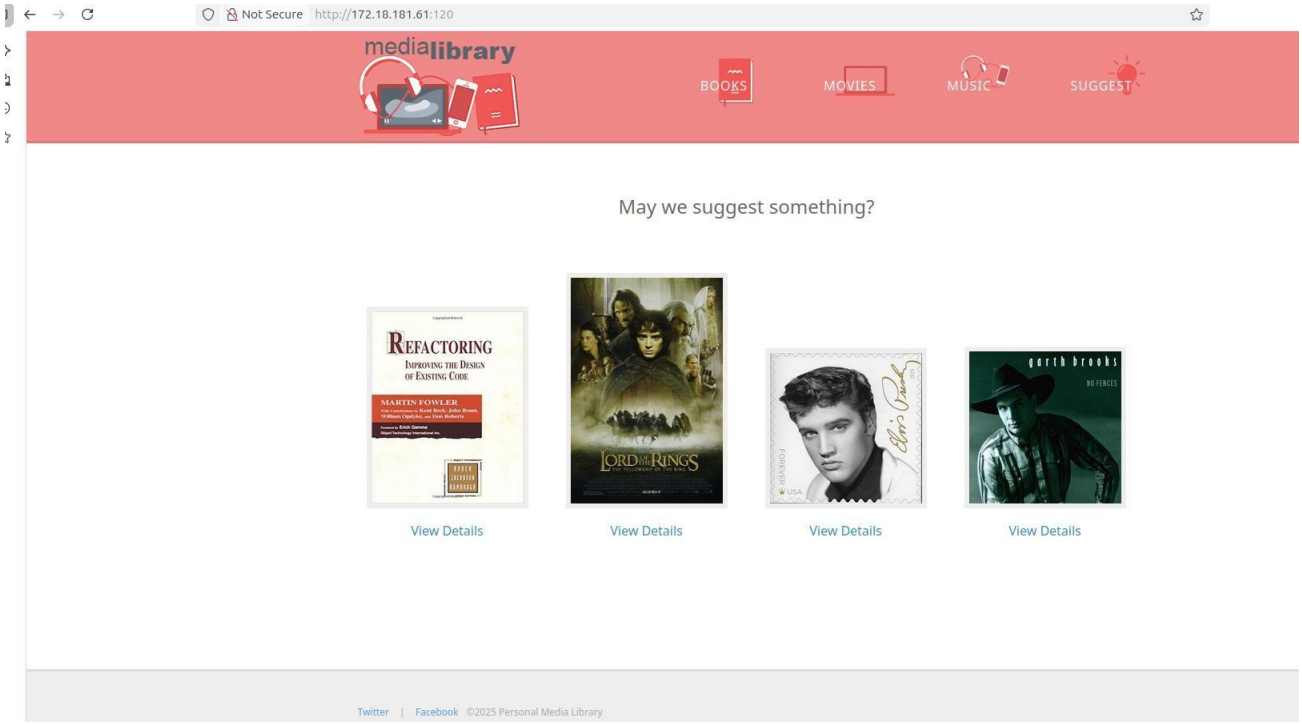
Click on **"+ Add Build Step"** to add another build step and select **"Execute Docker command."** Under **Docker command**, choose **"Start container(s)"** and provide the name of the container you created in the **Container ID(s)** field (e.g., phpbook). Then click **Save**.



Under **Configure** . Click on **build now.**



we can access the web page at http://localhost:<Host Port>/



****

3. Deploy containerized application on Kubernetes cluster environment – Java

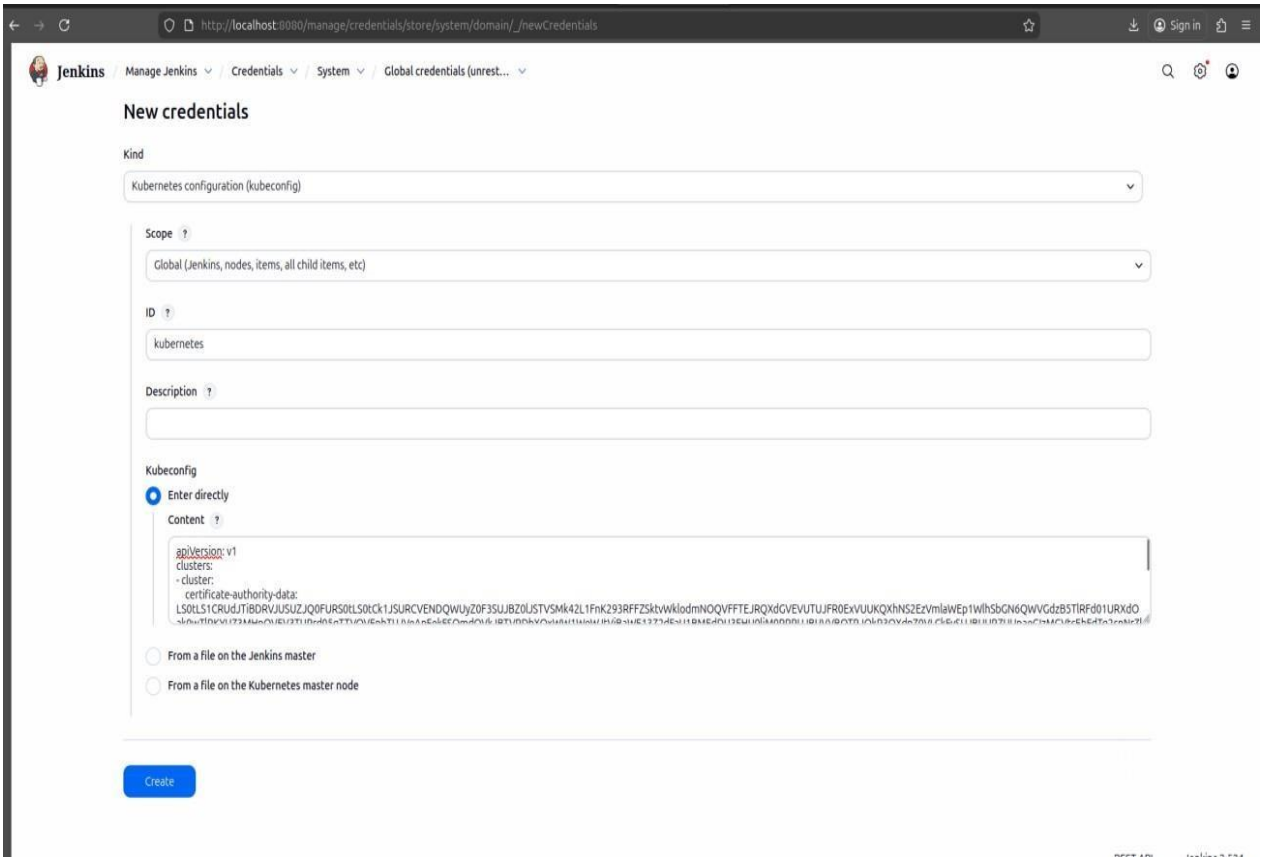
Plugins: Install the Kubernetes plugin using sir's kubernetes-cd_1.0.0.hpi file (available in the GitHub repository linked below).

Go to **Manage Jenkins → Plugins → Advanced**, browse for the .hpi file, and click **Deploy**.

Created the Kubernetes manifest files (deployment.yaml and service.yaml) and added them to the GitHub repository.

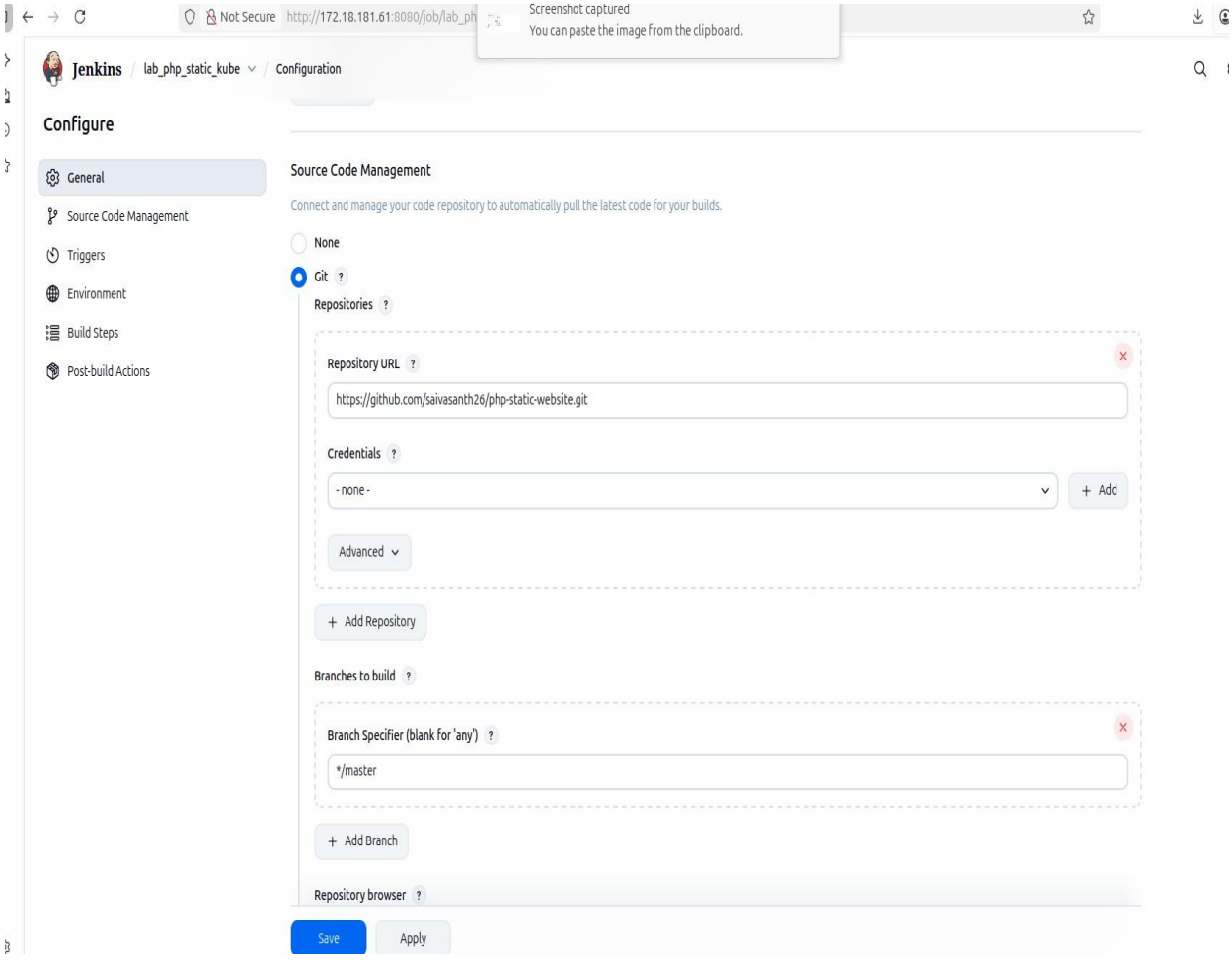
GitHub Repo: https://github.com/saivasanth26/php-static-website.git

Add Kubernetes credentials to Jenkins by navigating to **Manage Jenkins → Plugins**. Then,under **Kind**, select **"Kubernetes configuration (kubeconfig)"** and provide a name for the credential under **ID**. For **Kubeconfig**, choose **"Enter directly"** and paste the output of cat ~/.kube/config from the master node. Finally, click **Create**

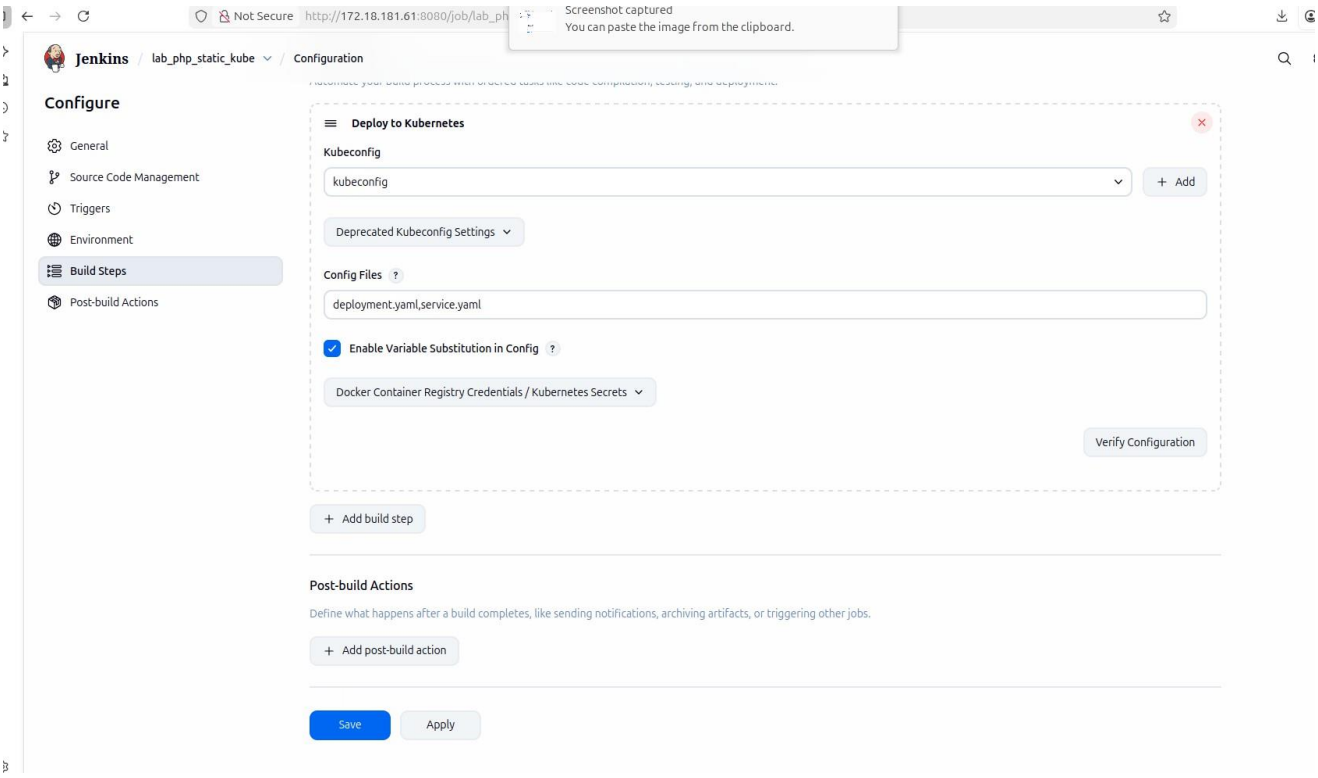


Create a **Freestyle Jenkins Project** and provide a name for the project, for example **lab_php_kube**, then click **OK**.

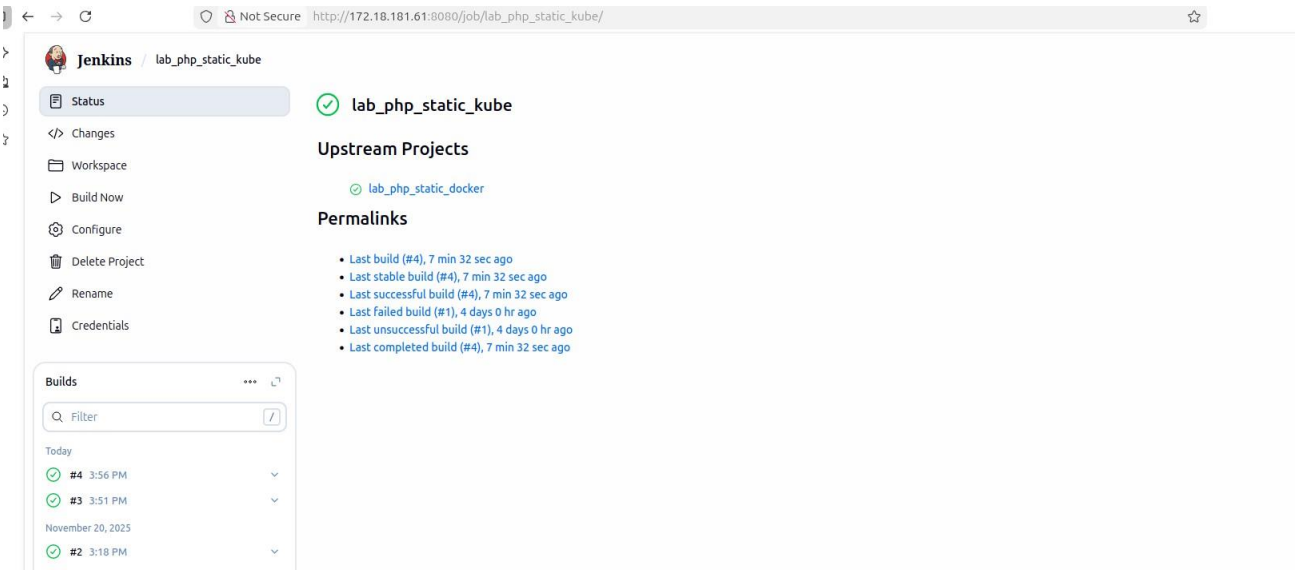
On the next page, select **Git** under **Source Code Management**, then enter the GitHub repository URL in the **Repository URL** field and specify the branch.



Under the **Build** step, select **"Deploy to Kubernetes."** Then choose the Kubernetes credential ID you added under **Kubeconfig**. In the **Config Files** section, specify the names of your deployment and service files (e.g., deployment.yaml, service.yaml). Finally, click **Save** and then **Build**.

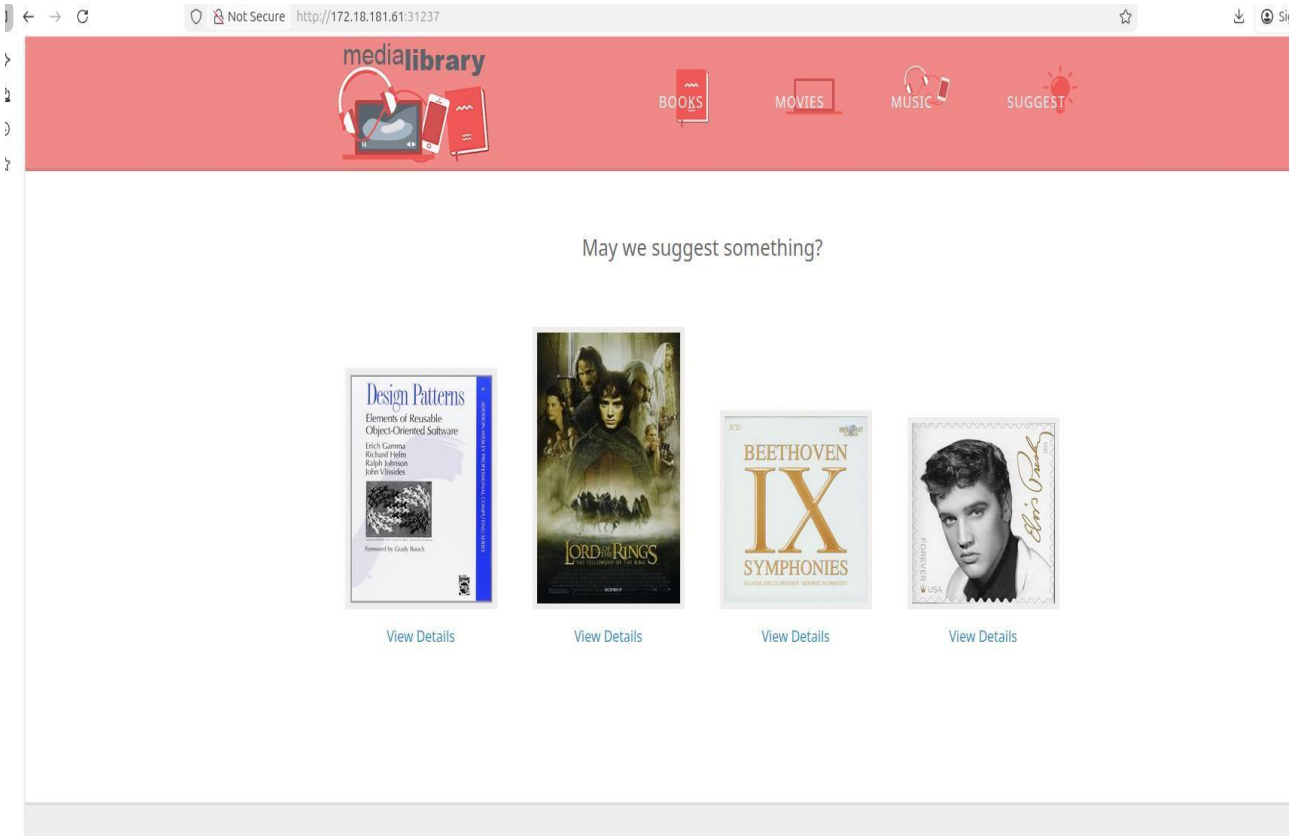


Under **Configure** . Click on **build now.**



http://localhost:<NodePort> or http://<cluster ip>:<Target Port>

http://localhost:31237/

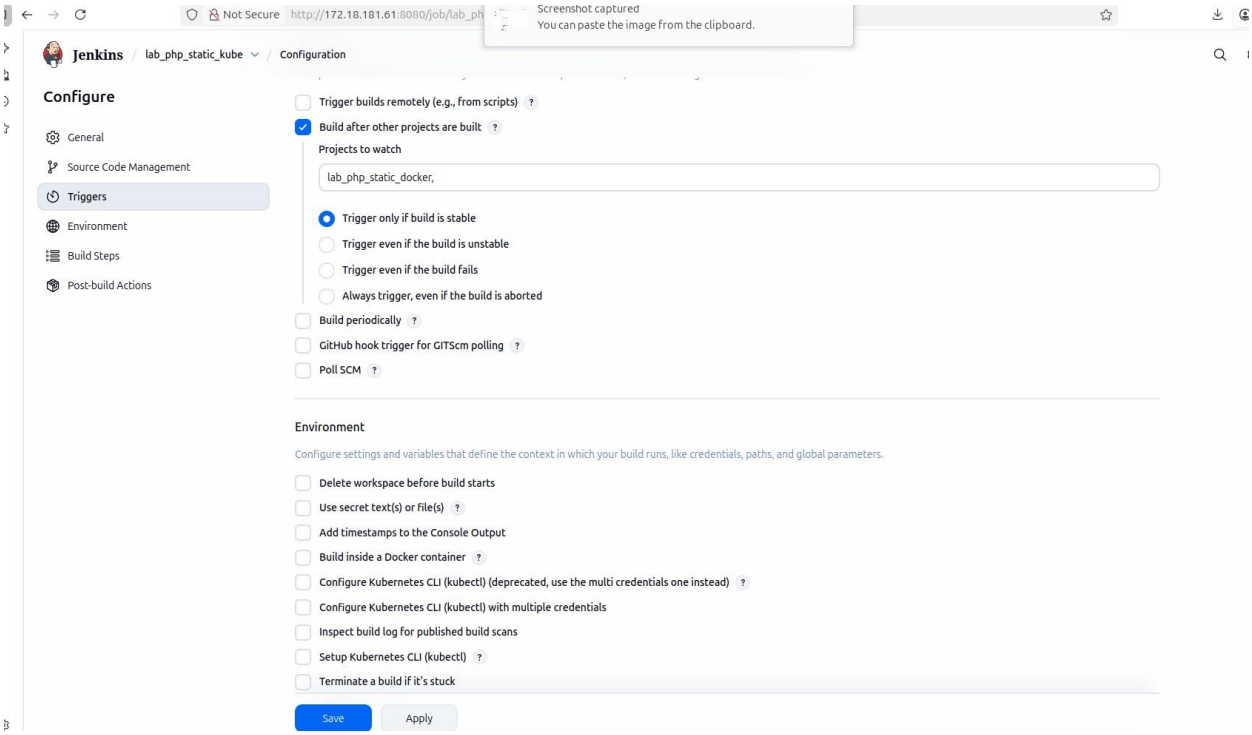


****

Create Pipeline for all these items (Baremetal->Docker->Kubernetes)

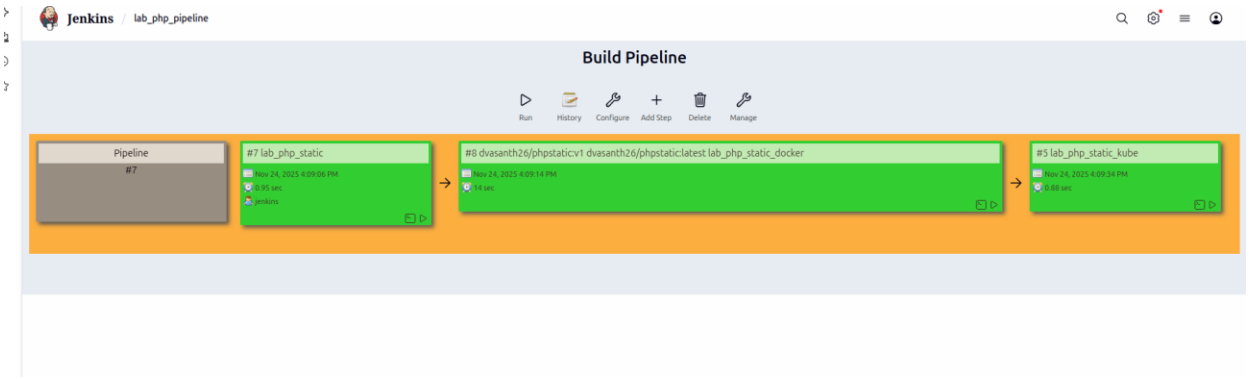
Since we have already created the pipeline for lab_php_static →
we now need to add the remaining two stages (Java_Docker and Java_Kubernetes).

Go to the item named lab_php_docker. Under Triggers, select "Build after other projects are built." Then, under Projects to watch, select lab_php_compile, and similarly go to project lab_php_kube and under Projects to watch, select lab_php_docker.



Now, the pipeline creation is complete.

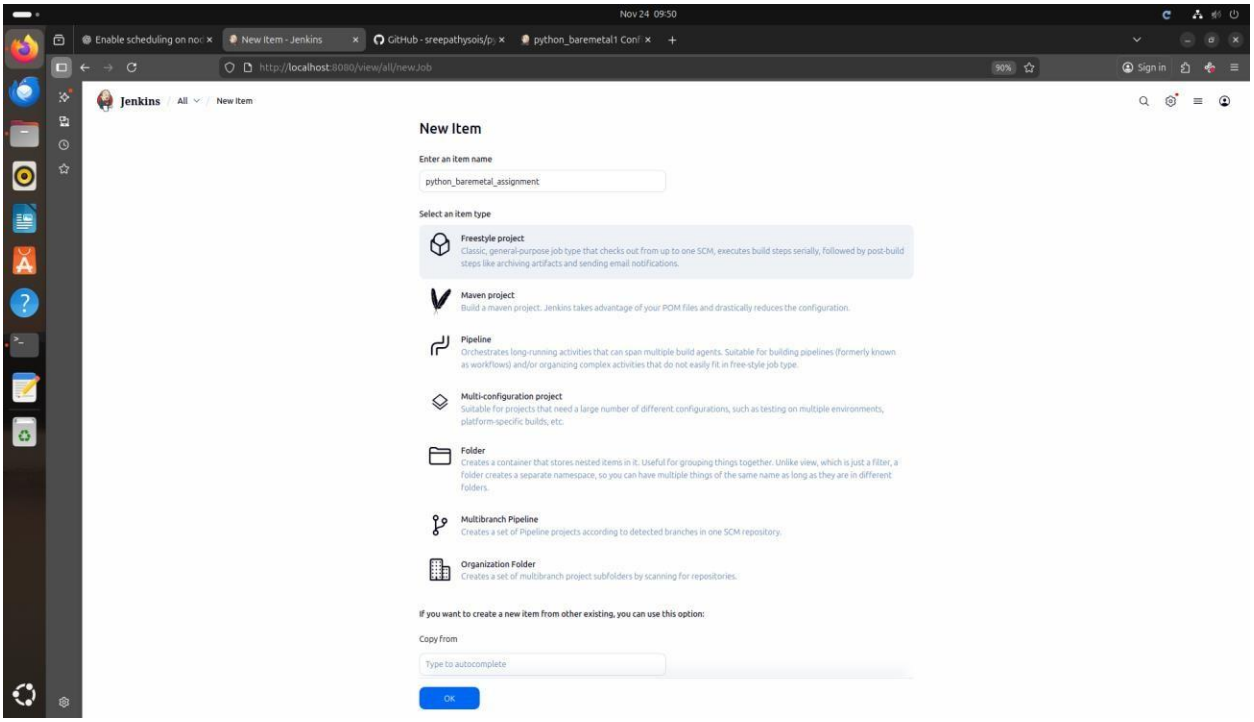
Run the Pipeline.

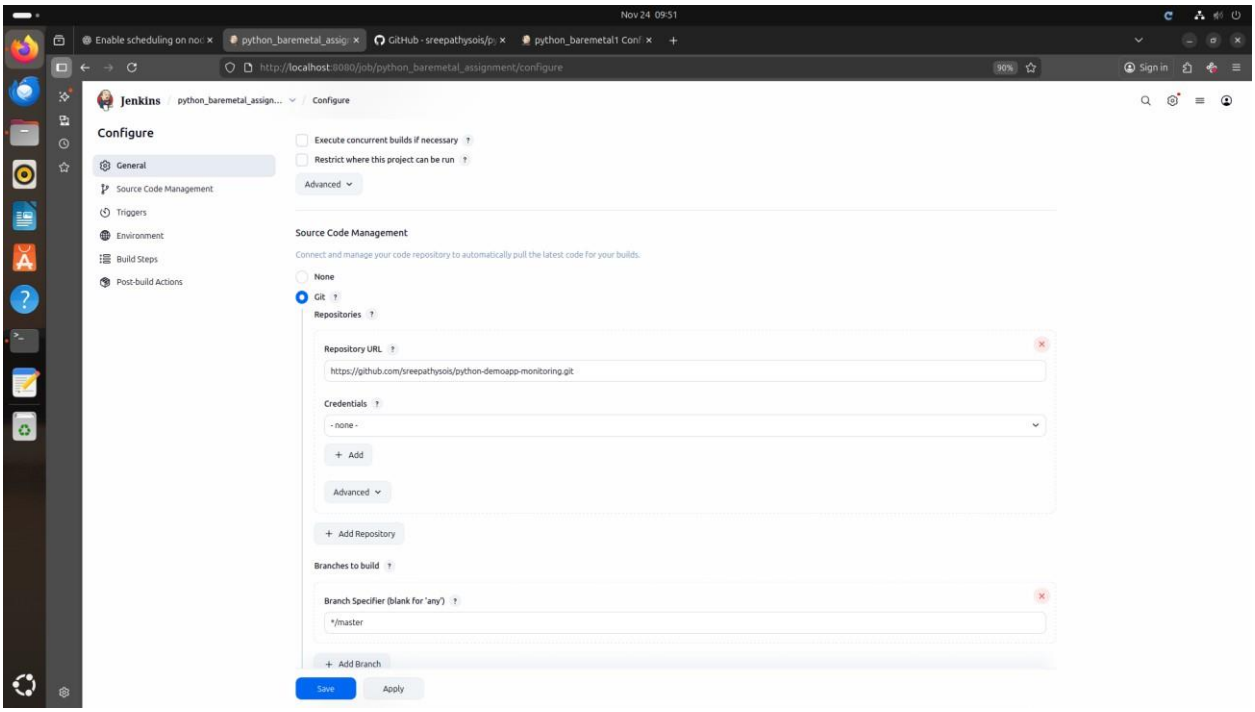


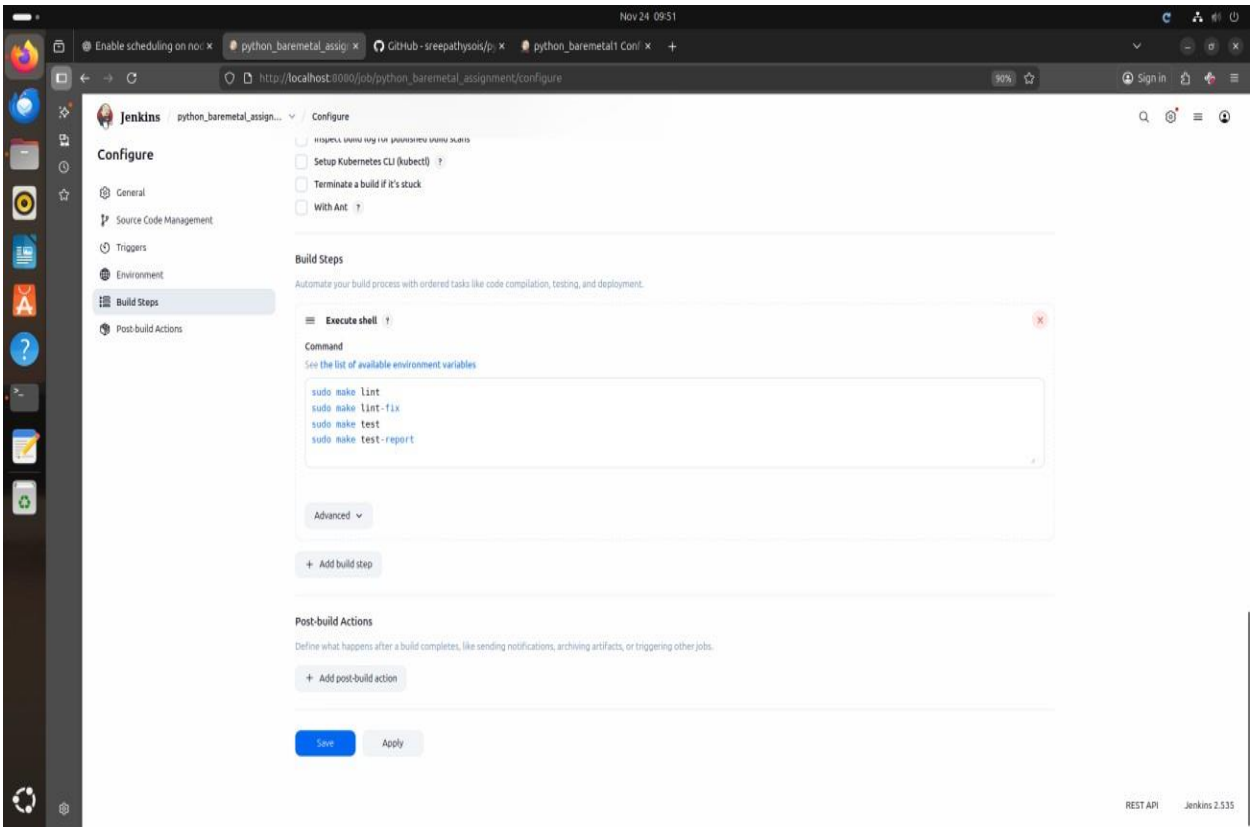
For Python ,

STEP 1

- CREATE A Freestyle project (baremetal)
- Add git url (select branch)
- Build steps – execute shell
- Post build actions – trigger only if build is stable



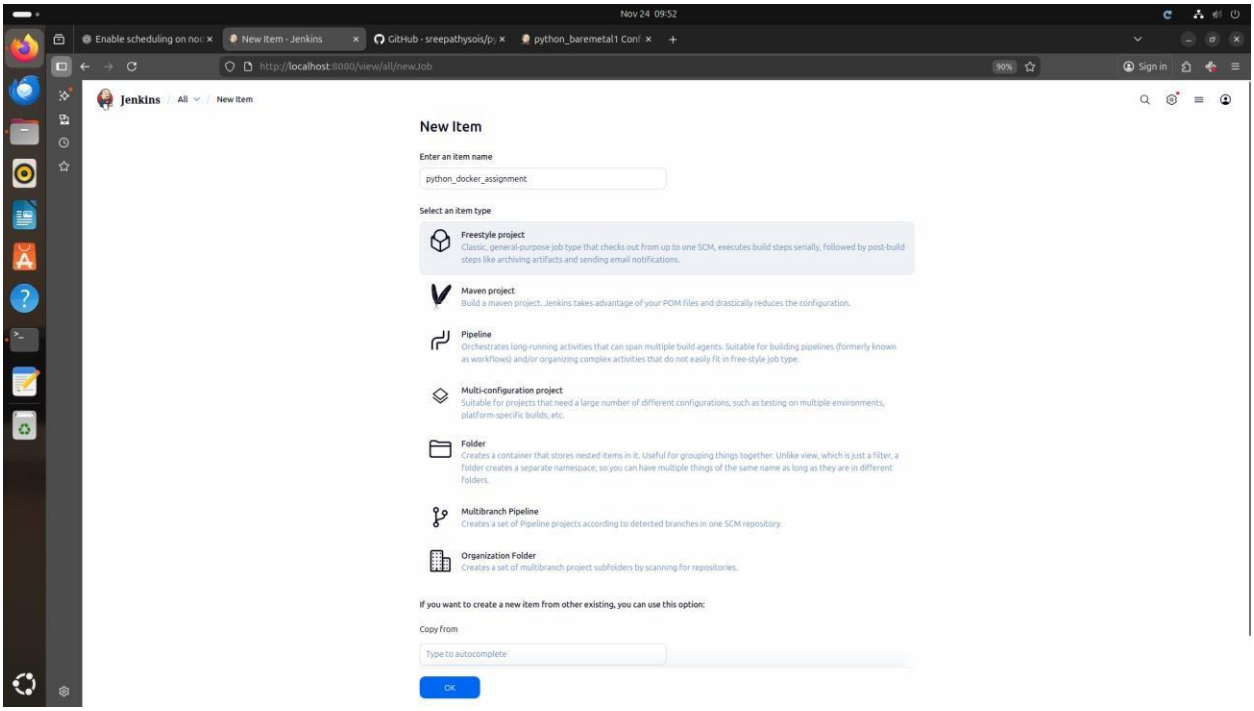

Nov 24 09:51
Enable scheduling on no
python_baremetal_assig
GitHub - sreepathysois/p
python_baremetal1 Conf
http://localhost:8080/job/python_baremetal_assignment/configure
90%
Sign in
Jenkins
python_baremetal_assign...
Configure
Configure
General
Source Code Management
Triggers
Environment
Build Steps
Post-build Actions
Execute concurrent builds if necessary
Restrict where this project can be run
Advanced
Source Code Management
Connect and manage your code repository to automatically pull the latest code for your builds.
None
Git
Repositories
Repository URL
https://github.com/sreepathysois/python-demoapp-monitoring.git
Credentials
- none -
+ Add
Advanced
+ Add Repository
Branches to build
Branch Specifier (blank for 'any')
*/master
+ Add Branch
Save
Apply

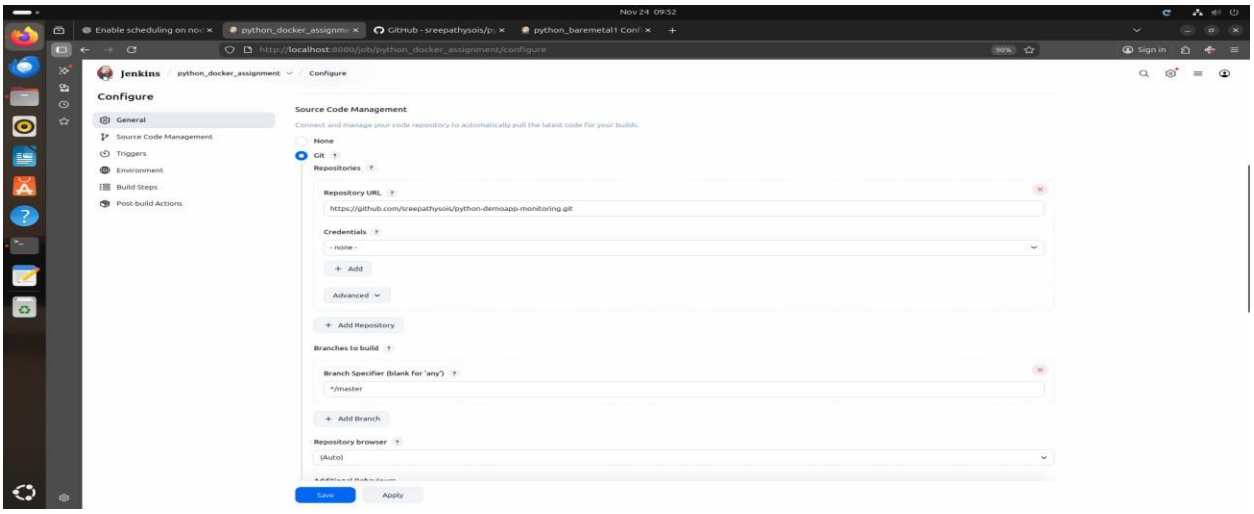

Nov 24 09:51
Enable scheduling on no
python_baremetal_assig
GitHub - sreepathysois/p
python_baremetal1 Conf
http://localhost:8080/job/python_baremetal_assignment/configure
90%
Sign in
Jenkins
python_baremetal_assign...
Configure
Configure
General
Source Code Management
Triggers
Environment
Build Steps
Post-build Actions
Setup Kubernetes CLI (kubectl)
Terminate a build if it's stuck
With Ant
Build Steps
Automate your build process with ordered tasks like code compilation, testing, and deployment.
Execute shell
Command
See the list of available environment variables
sudo make lint
sudo make lint-fix
sudo make test
sudo make test-report
Advanced
+ Add build step
Post-build Actions
Define what happens after a build completes, like sending notifications, archiving artifacts, or triggering other jobs.
+ Add post-build action
Save
Apply
REST API
Jenkins 2.535

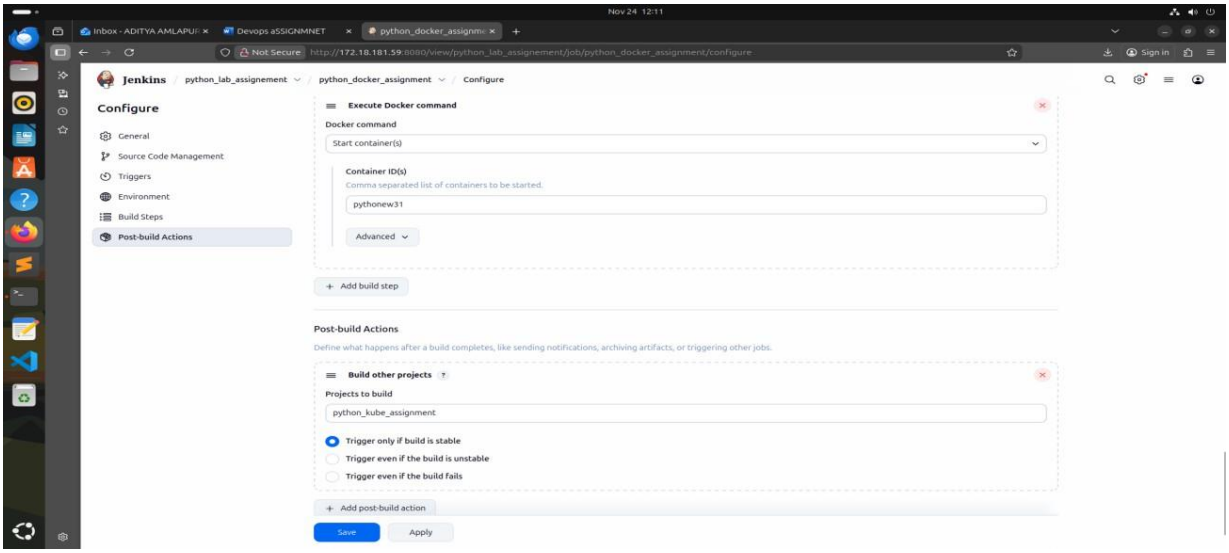
Step 2

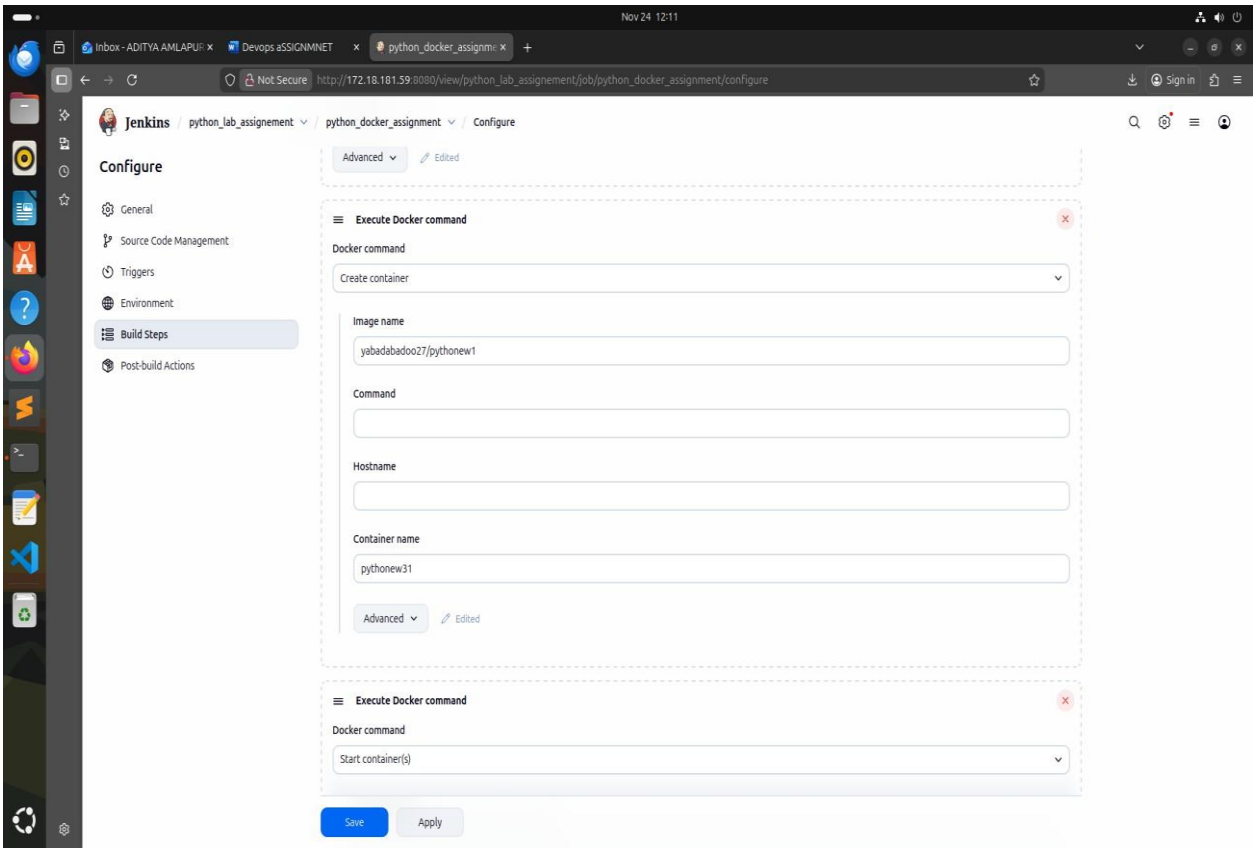
Create one more freestyle project (python docker)

- Git url – master branch
- Build steps – docker build n publish (repo name n cred)
- Execute docker command – create and start
- Post build actions – trigger only if build is stable




Nov 24 09:52
Jenkins
python_docker_assignment
Configure
Configure
General
Source Code Management
Triggers
Environment
Build Steps
Post-build Actions
Source Code Management
None
Git
Repositories
Repository URL
https://github.com/sreepathysois/python-demoapp-monitoring.git
Credentials
- none -
Add
Advanced
Add Repository
Branches to build
Branch Specifier (blank for 'any')
*/master
Add Branch
Repository browser
(Auto)
Save
Apply


Nov 24 12:11
Jenkins
python_lab_assignement
python_docker_assignment
Configure
Configure
General
Source Code Management
Triggers
Environment
Build Steps
Post-build Actions
Execute Docker command
Docker command
Start container(s)
Container ID(s)
Comma separated list of containers to be started.
pythonew31
Advanced
Add build step
Post-build Actions
Define what happens after a build completes, like sending notifications, archiving artifacts, or triggering other jobs.
Build other projects
Projects to build
python_kube_assignment
Trigger only if build is stable
Trigger even if the build is unstable
Trigger even if the build fails
Add post-build action
Save
Apply

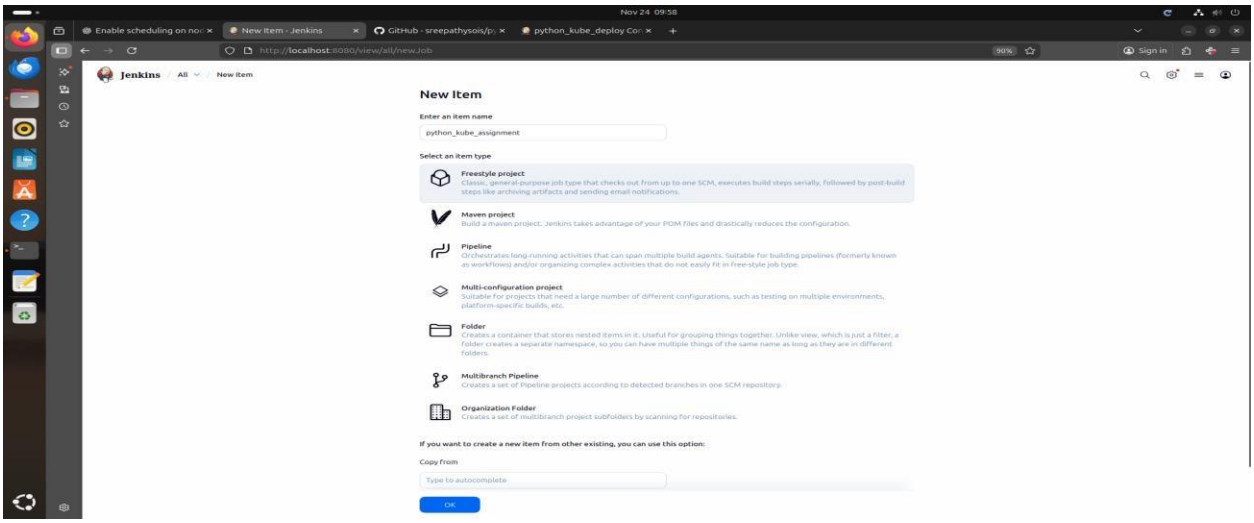


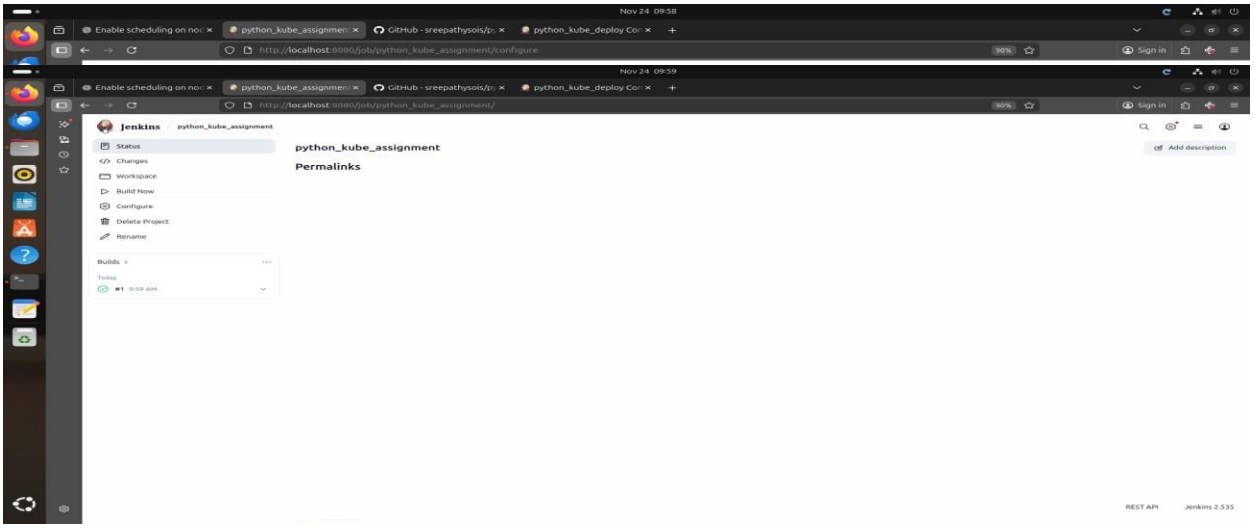
Step 3

Create one more freestyle project (kubernates)

Note for this : master slave should be up and running (node)

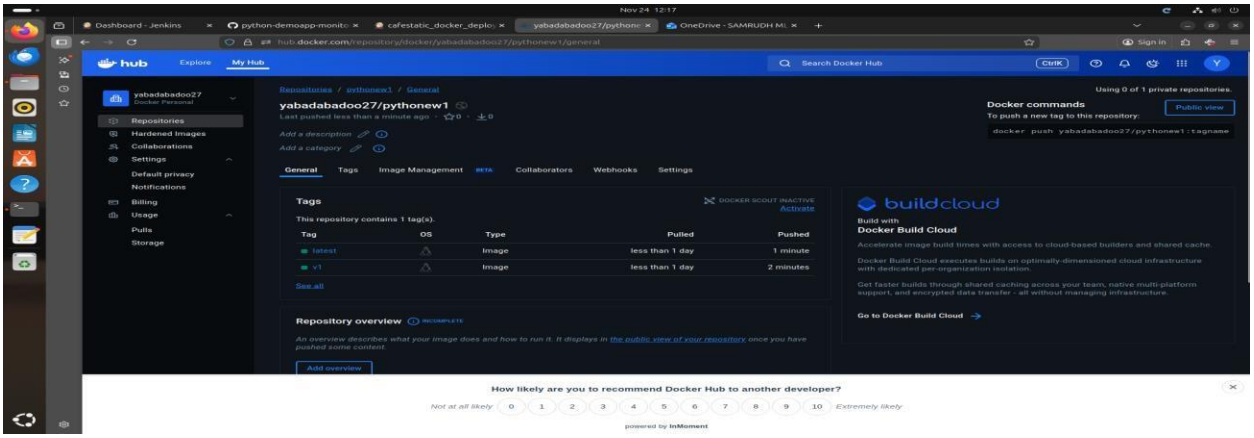
- Git
- Build steps – deploy to k8s(kubeconfig , files)



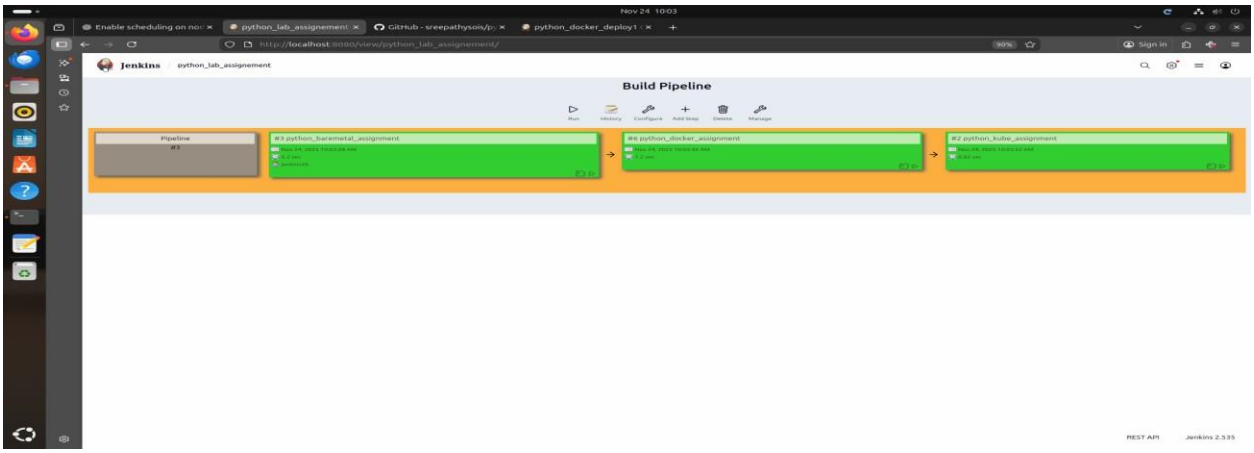


Results

1. Docker Hub – images pushed to doscker hub.



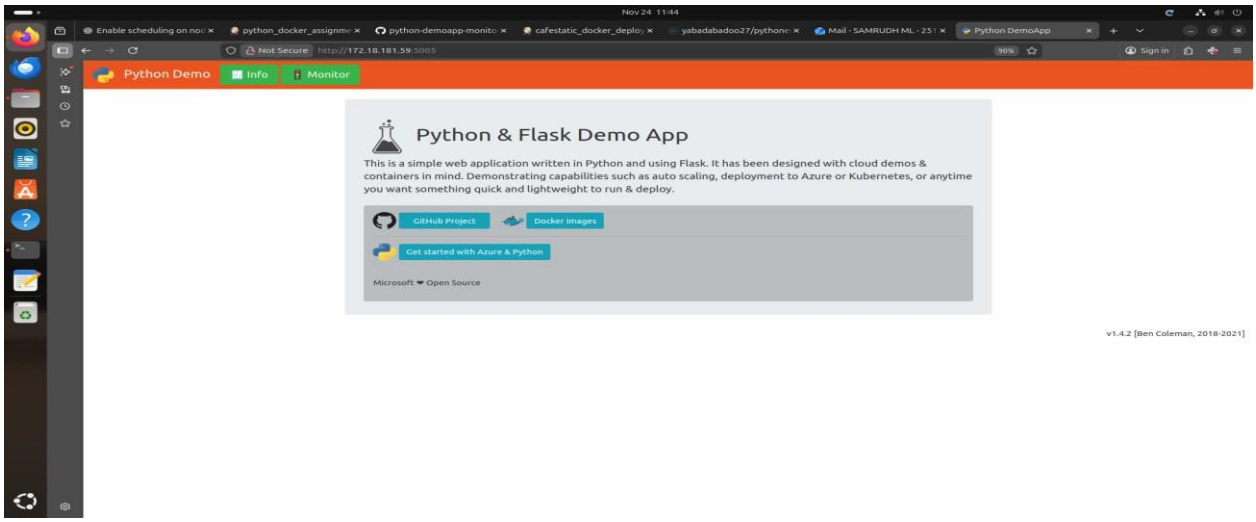
2. Pipeline view



3. Docker images running

```
msis@k8s-master-node:~$ docker ps
CONTAINER ID   IMAGE                          COMMAND                  CREATED          STATUS              PORTS                                                  NAMES
db5008d3390d   pythonmonitoringapp:v1         "gunicorn -b 0.0.0.0…"   About a minute ago   Up About a minute   5000/tcp, 0.0.0.0:3006->3000/tcp, [::]:3006->3000/tcp   pythonmonitoring
2
c8f58fbcaa8e   pymonitorapp:v1                "gunicorn -b 0.0.0.0…"   17 minutes ago   Up 17 minutes       5000/tcp, 0.0.0.0:3003->3000/tcp, [::]:3003->3000/tcp   pymonitoring
b8a092e29f3a   likhith189/node-hello:latest   "docker-entrypoint.s…"   2 days ago       Up 2 days           0.0.0.0:3005->3000/tcp, [::]:3005->3000/tcp             helloapp1
10649d2884fa   likhith189/node-hello:latest   "docker-entrypoint.s…"   2 days ago       Up 2 days           3000/tcp                                                helloapp
cd71e78af29c   likhith189/simple_flask:v1     "flask run --port 50…"   2 days ago       Up 2 days           0.0.0.0:5071->5070/tcp, [::]:5071->5070/tcp             simpleflaskapp
fb9cb31ad9ac   simplenodejs:v1                "npm start"              2 days ago       Up 2 days           0.0.0.0:3001->3000/tcp, [::]:3001->3000/tcp             simplenodejsapp
```

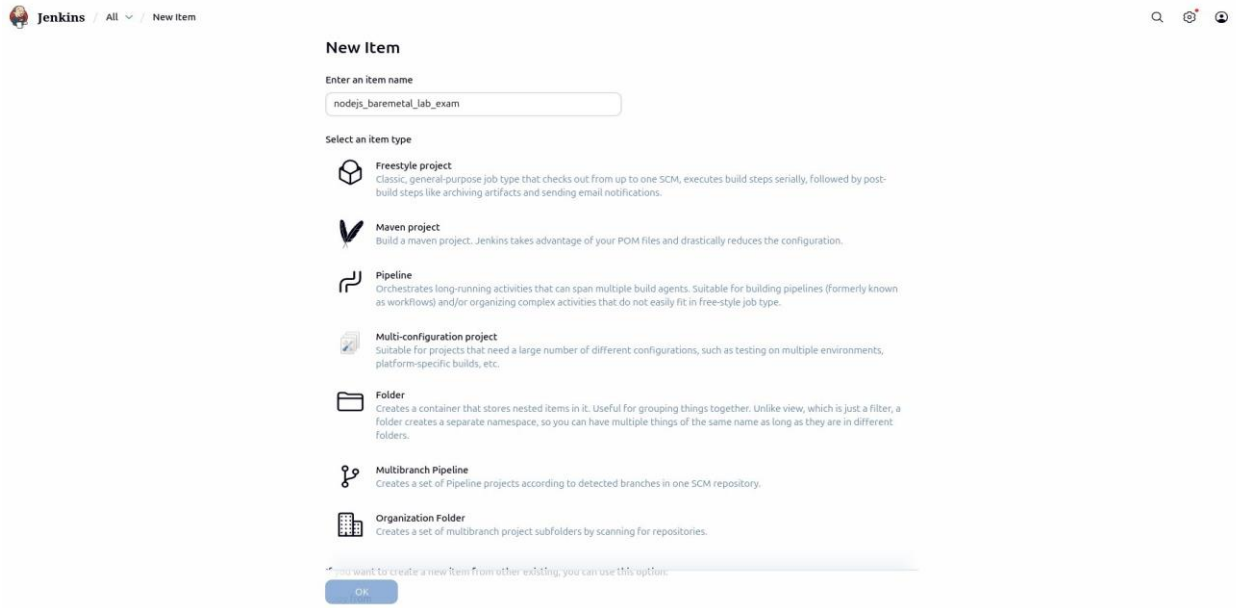
4. Python monitoring running on port:5005


Nov 24 11:44
Python DemoApp
Not Secure
http://172.18.181.59:5005
Sign in
Python Demo
Info
Monitor
Python & Flask Demo App
This is a simple web application written in Python and using Flask. It has been designed with cloud demos & containers in mind. Demonstrating capabilities such as auto scaling, deployment to Azure or Kubernetes, or anytime you want something quick and lightweight to run & deploy.
GitHub Project
Docker Images
Get started with Azure & Python
Microsoft ❤ Open Source
v1.4.2 [Ben Coleman, 2018-2021]

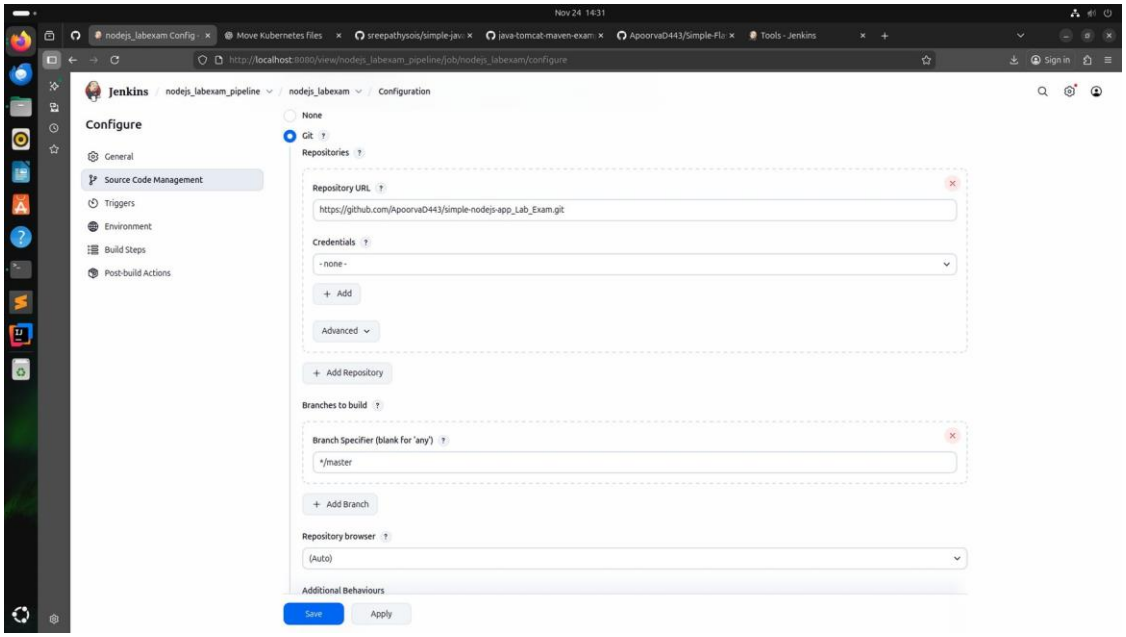
For Node js,

STEP 1: Implement CI/CD Build Pipeline for Nodejs Application on Jenkins Server

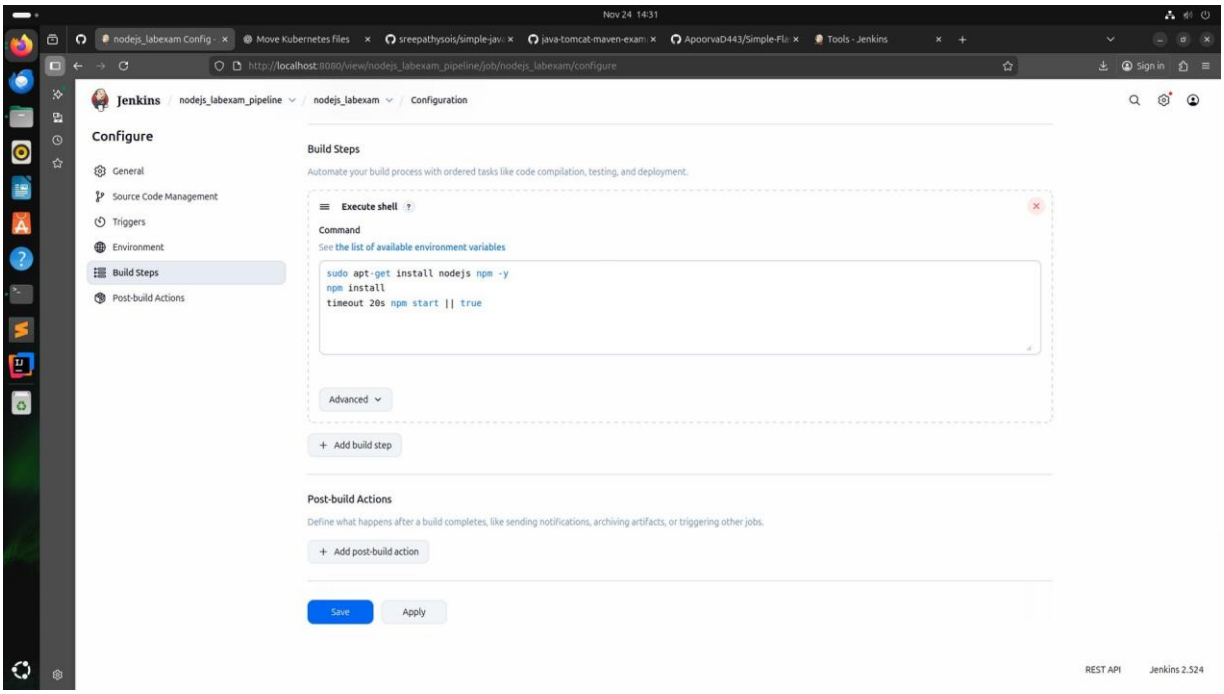
- Create a freestyle Jenkins Project
- Provide a name to Project for Example: nodejs_baremetal then click Ok



- On the next page, select Source Code Management as Git, then provide the GitHub repository URL under Repository URL and specify the branch.



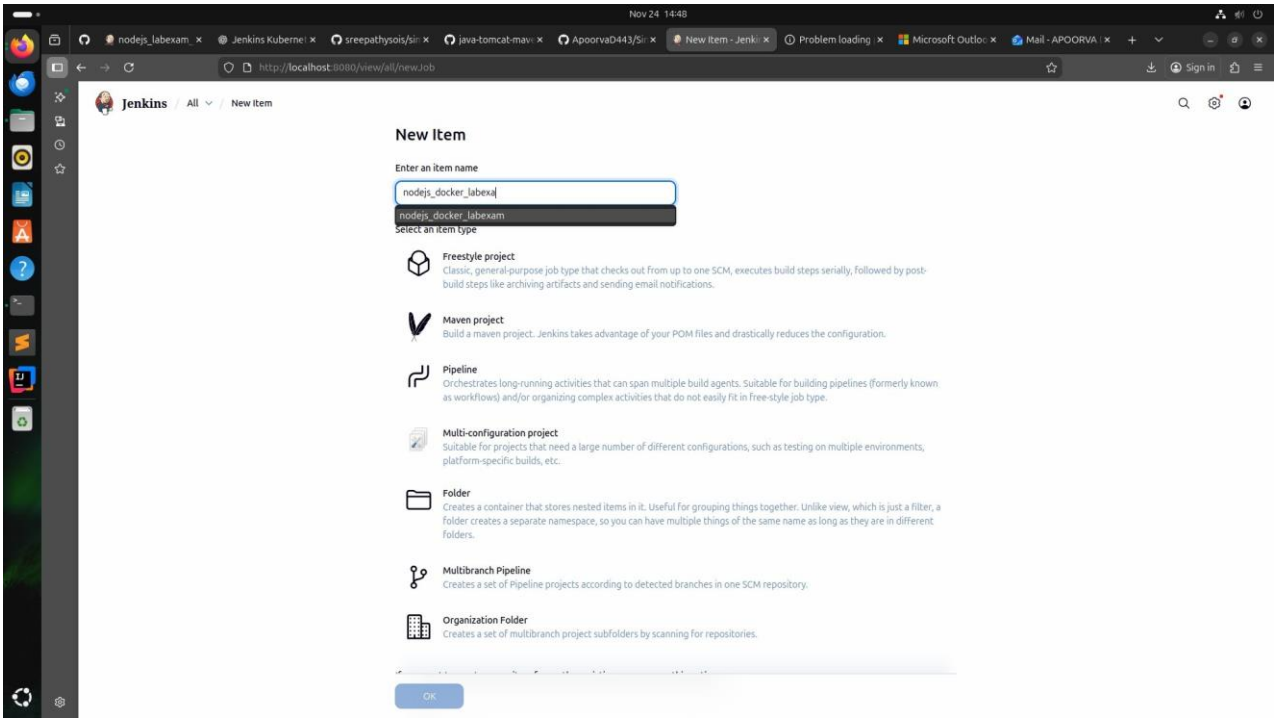
- Under the Build steps select execute shell and enter the below commands then click save.



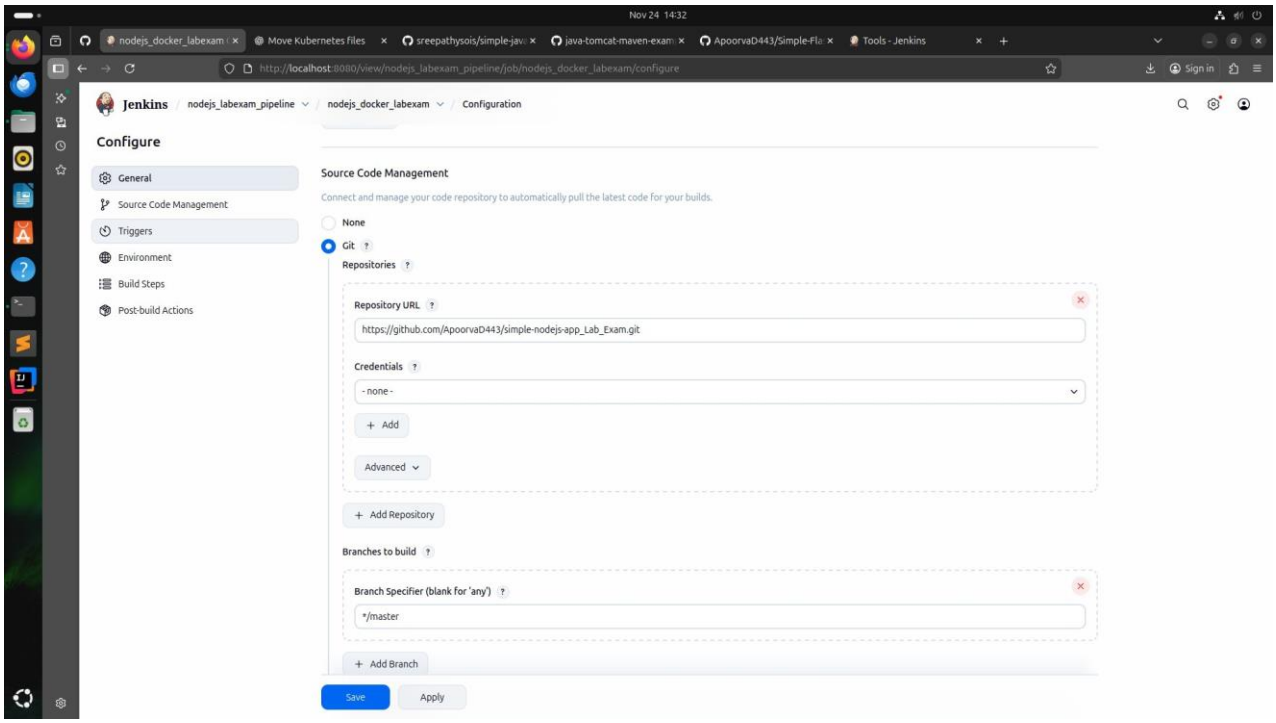
- Click on Build now.

STEP 2 : Containerizing the Node js Application using Docker

- Create a freestyle Jenkins Project
- Provide a name to Project for Example: nodejs_docker then click Ok.

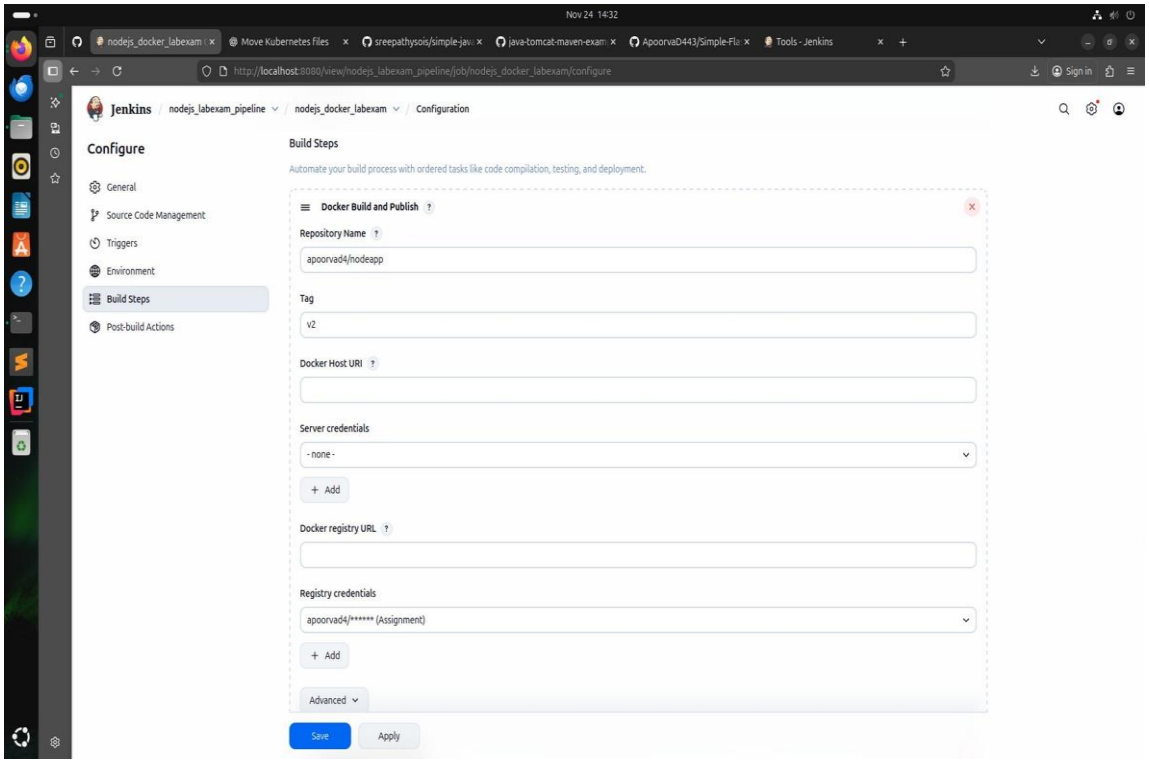


On the next page, select Source Code Management as Git, then provide the GitHub repository URL under Repository URL and specify the branch.

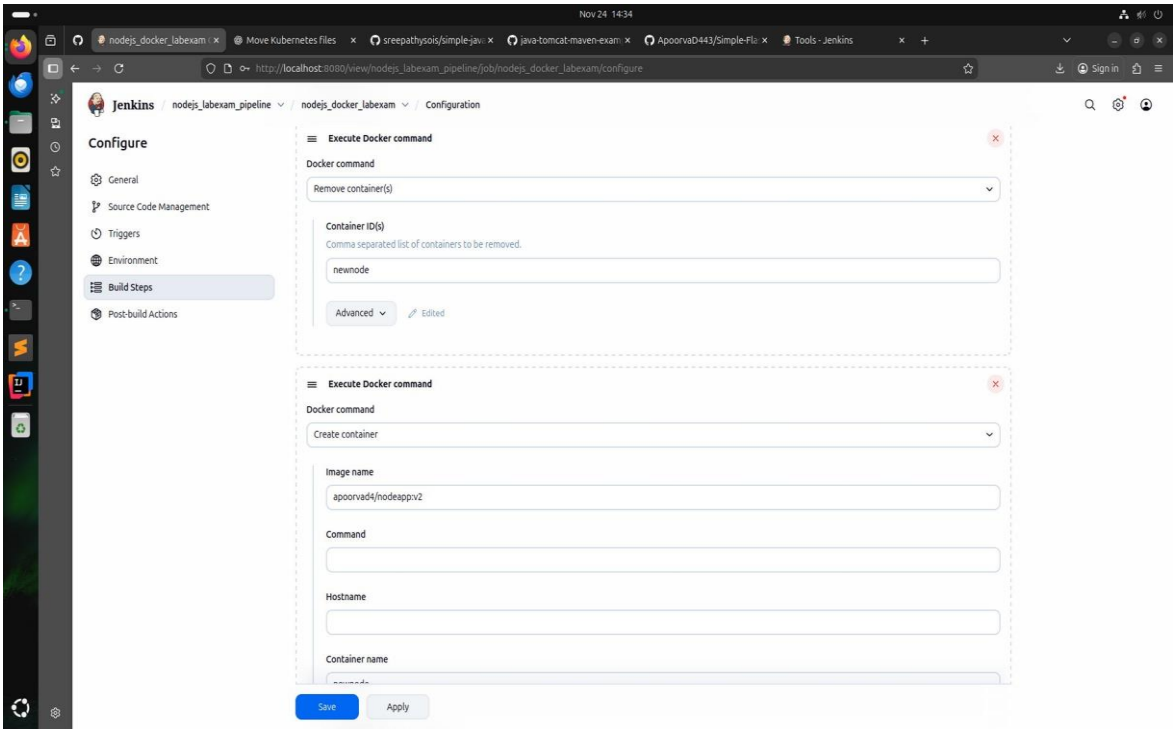


Under Build Steps, select Docker Build and Publish.
For Repository Name, enter <DockerHub_Username>/<image_name>

Set the Tag and under Registry credentials, select the Docker credential ID.



Click on "+ Add Build Step" to add another build step and select "Execute Docker command." Under Docker command, choose "Remove container(s)."



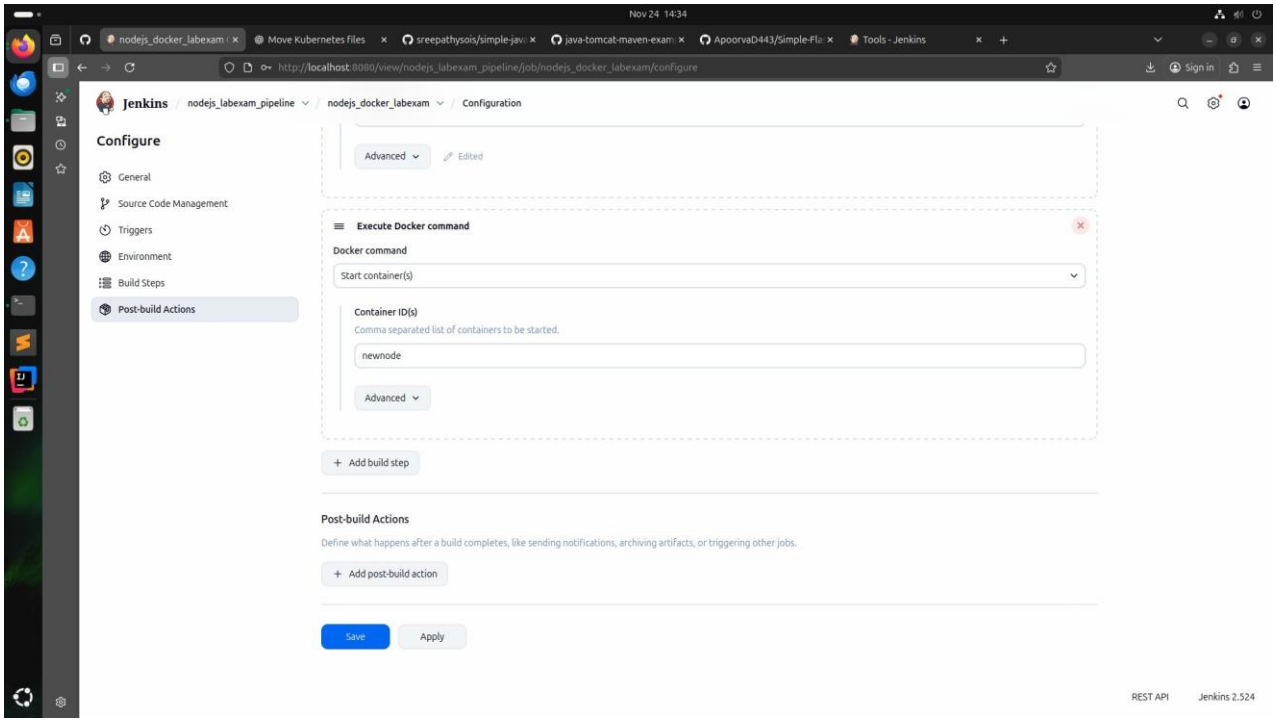
Click on “+ Add Build Step” to add another build step and select “Execute Docker command.”

Under Docker command, choose “Create container.”

In the Image name field, enter the name of the image you created in the previous build
step along with its version (e.g., apoorvad4/nodeapp:v2).

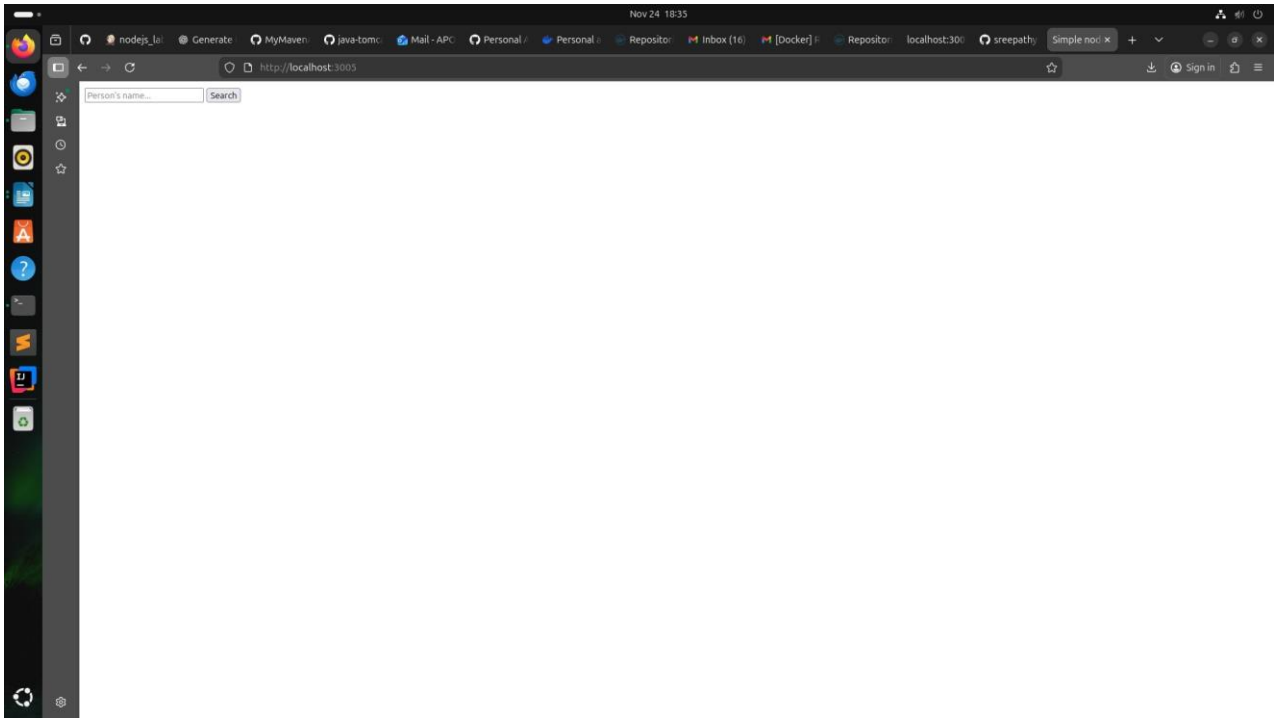
In the Container name field, provide the name of the container you want to create (e.g., new node)

Click Advanced, select “Publish all ports,” and under Port bindings, map the ports in the format Host_port:Container_port (e.g., 3005:3000).



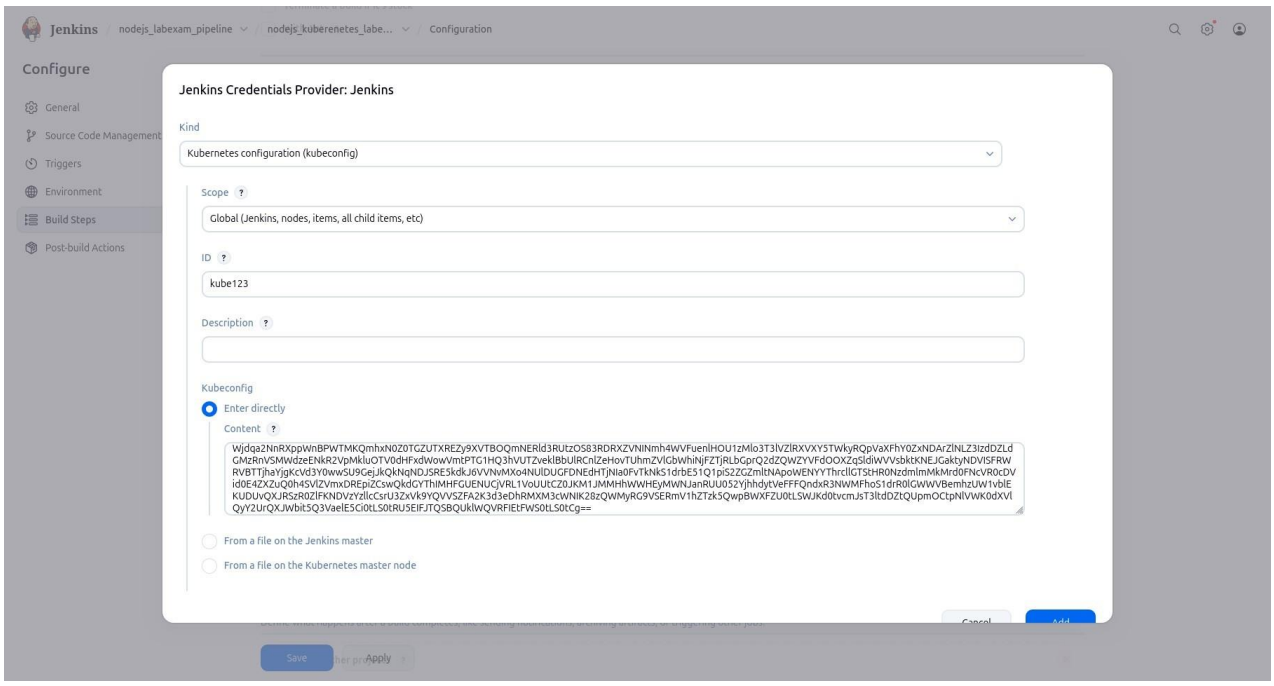
Click on “+ Add Build Step” to add another build step and select “Execute Docker command.” Under Docker command, choose “Start container(s)” and provide the name
of the container you created in the Container ID(s) field (e.g., newnode). Then click Save.

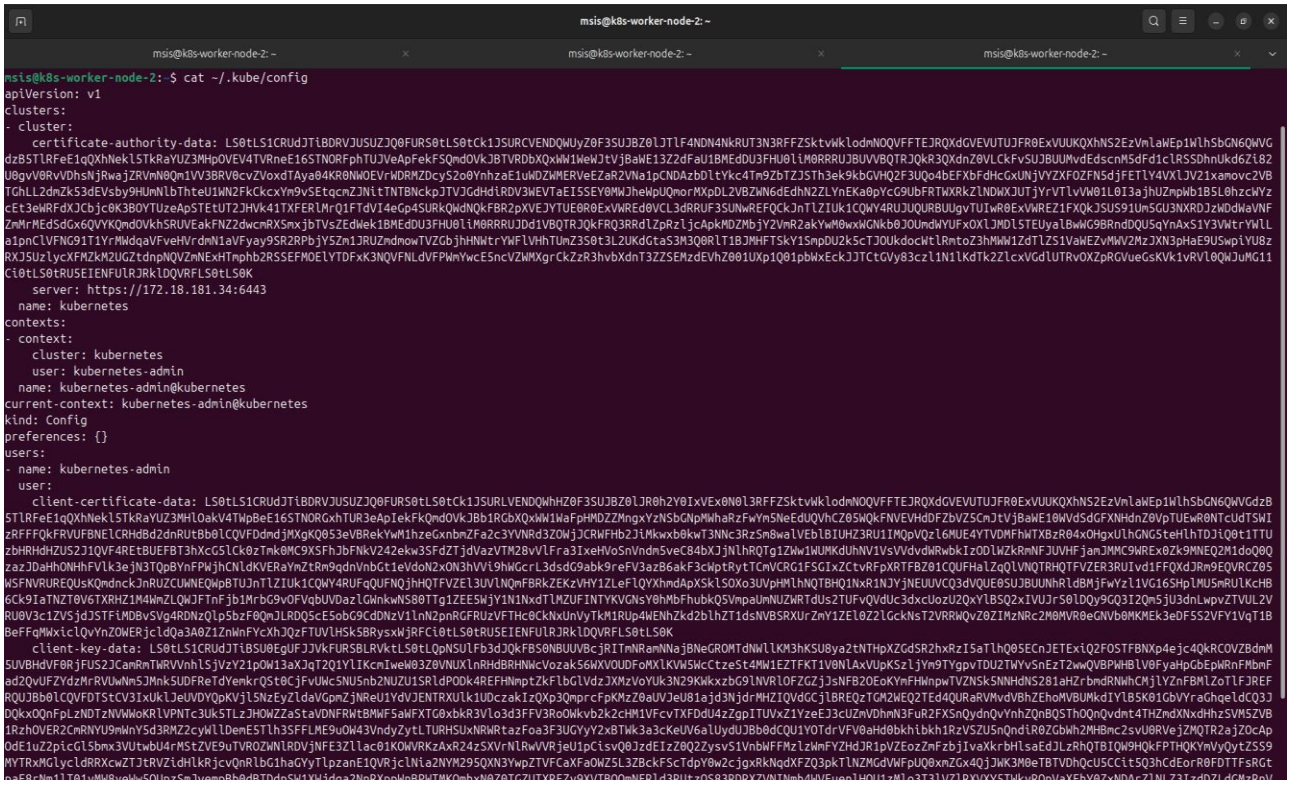
We can access the web page at http://localhost:<hostport>



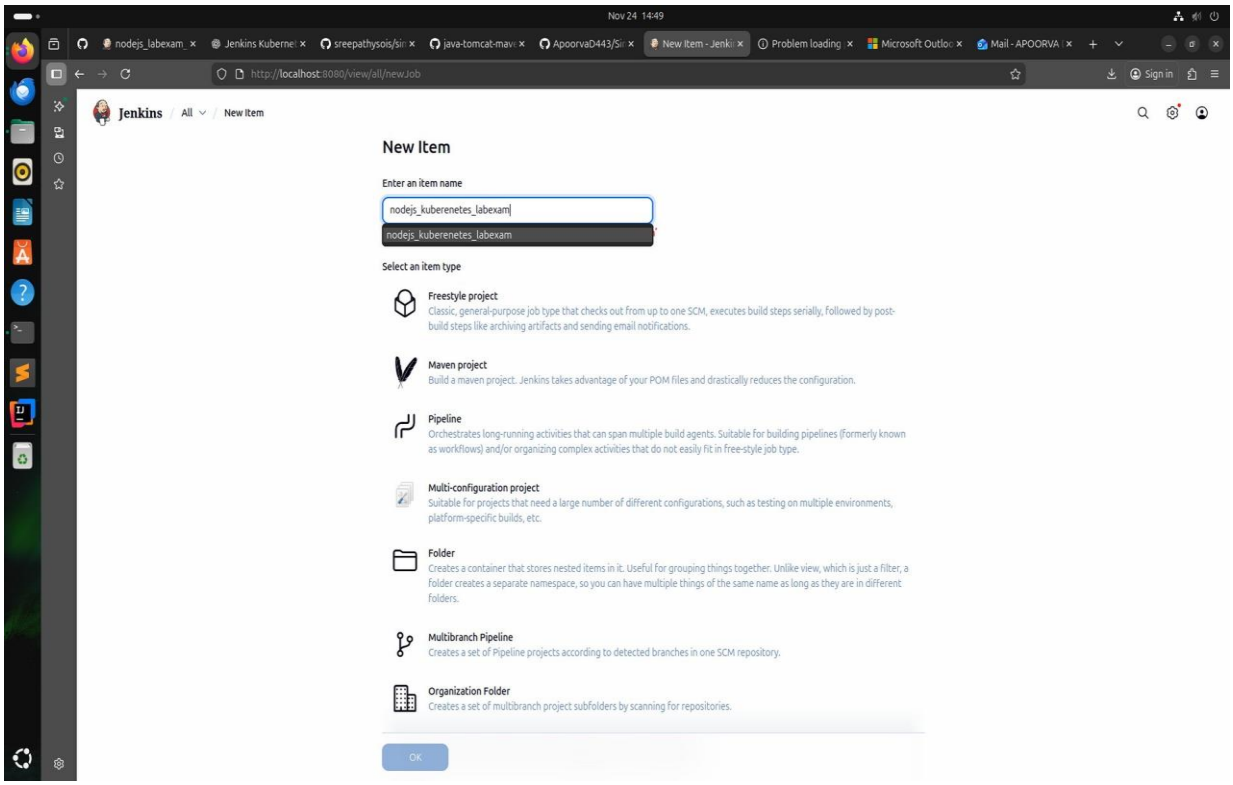
STEP 3 : Deploy containerized application on Kubernetes cluster environment

Add Kubernetes credentials to Jenkins by navigating to Manage Jenkins → Plugins. Then, under Kind, select “Kubernetes configuration (kubeconfig)” and provide a name for the credential under ID. For Kubeconfig, choose “Enter directly” and paste the output of cat ~/.kube/config from the master node. Finally, click Create

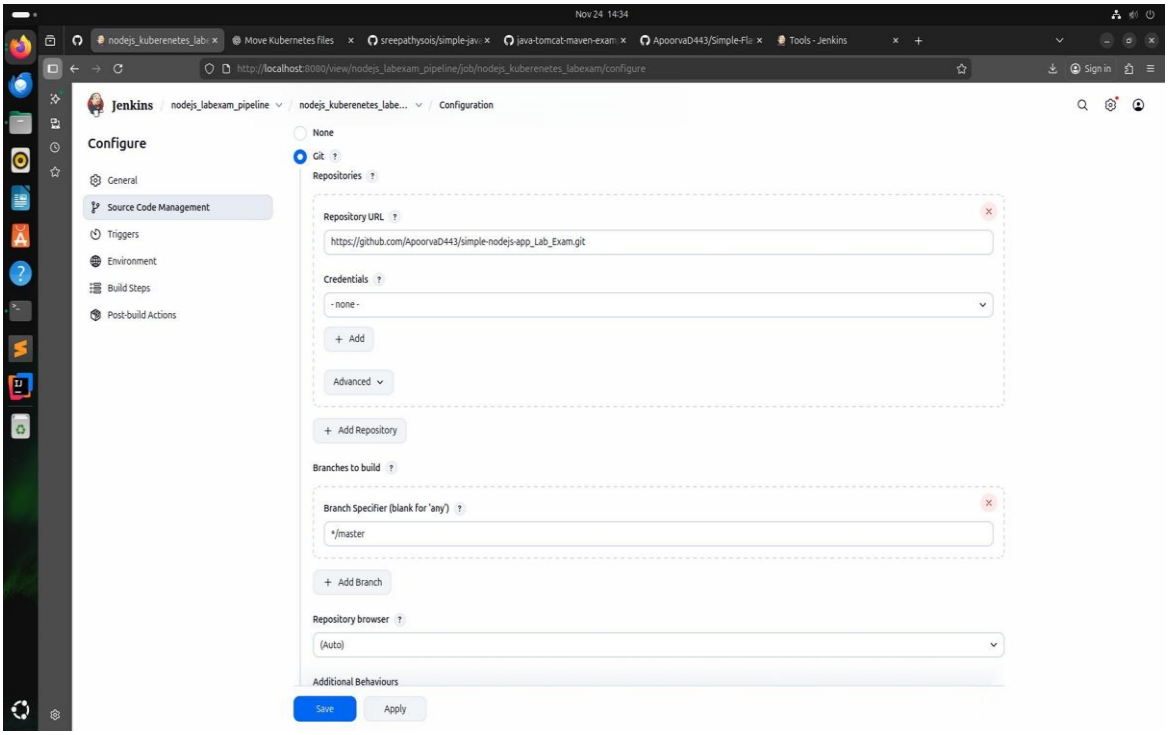




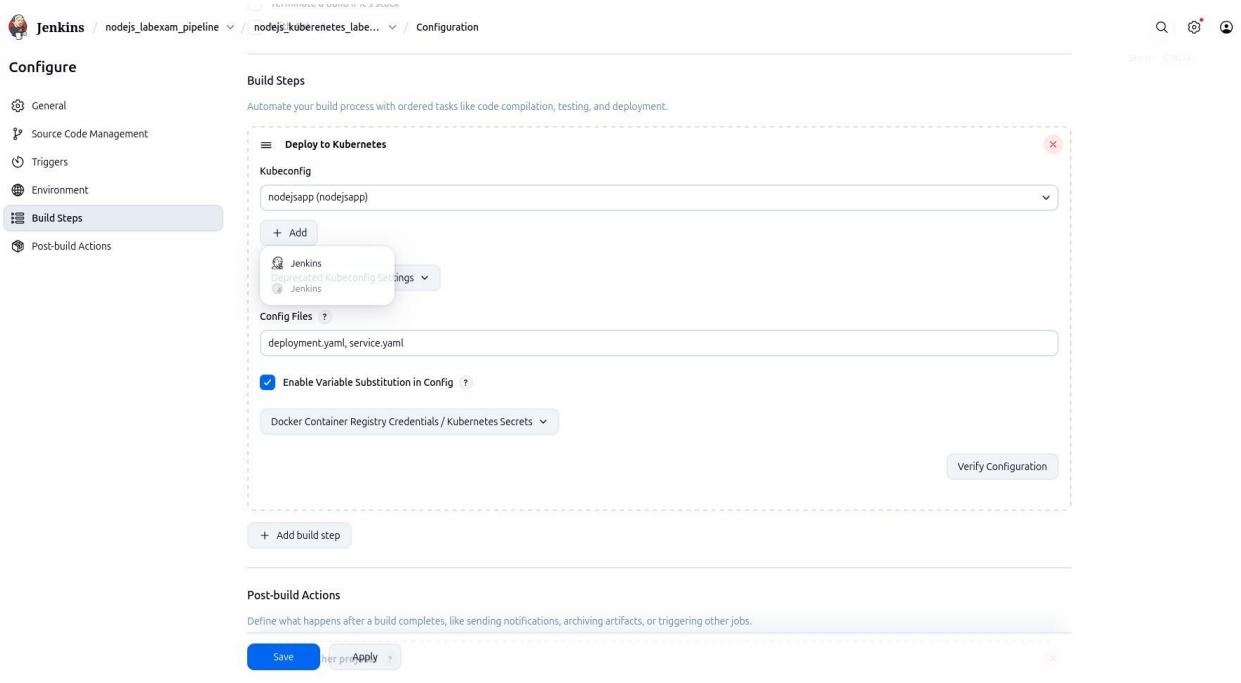
- Create a freestyle Jenkins Project
- Provide a name to Project for Example: nodejs_kubernetes then click Ok.



On the next page, select Source Code Management as Git, then provide the GitHub repository URL under Repository URL and specify the branch



Under the Build step, select “Deploy to Kubernetes.” Then choose the Kubernetes credential ID you added under Kubeconfig. In the Config Files section, specify the names of your deployment and service files (e.g., deployment.yaml, service.yaml). Finally, click Save and then Build.



In terminal type kubectl get svc

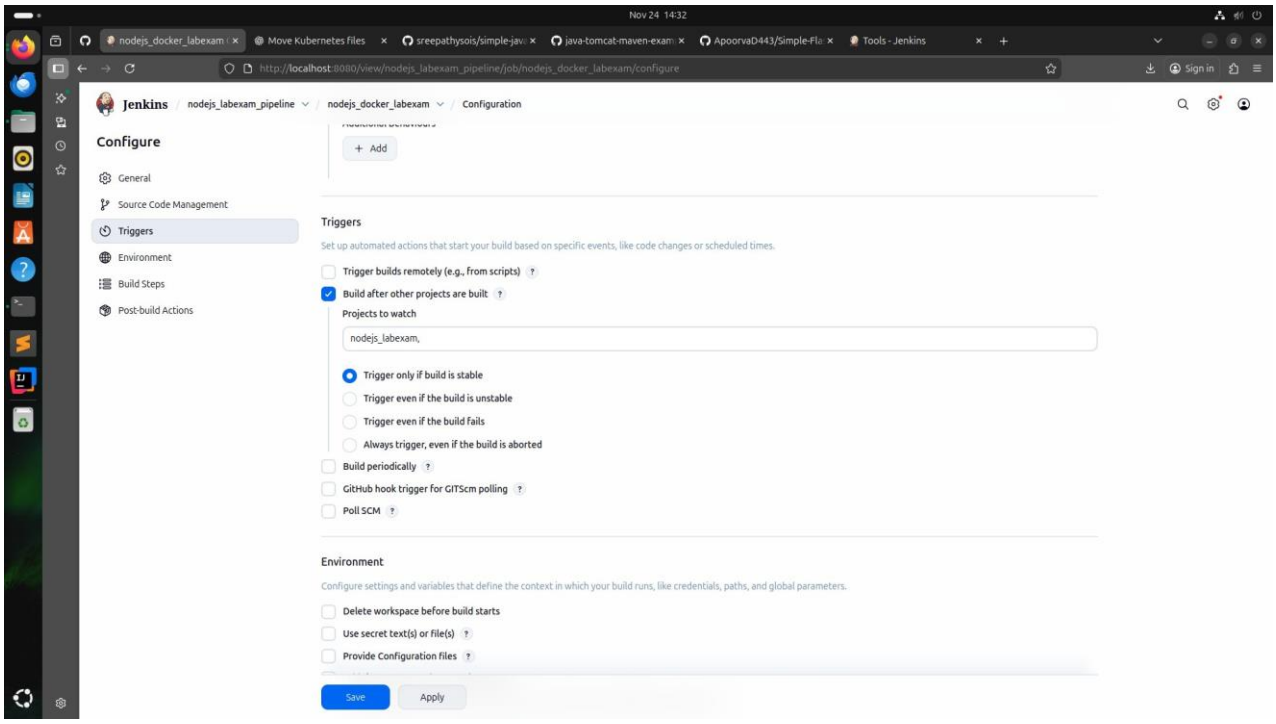
We can access the web page at,

http://localhost:<NodePort> or http://<cluster ip>:<Target Port>

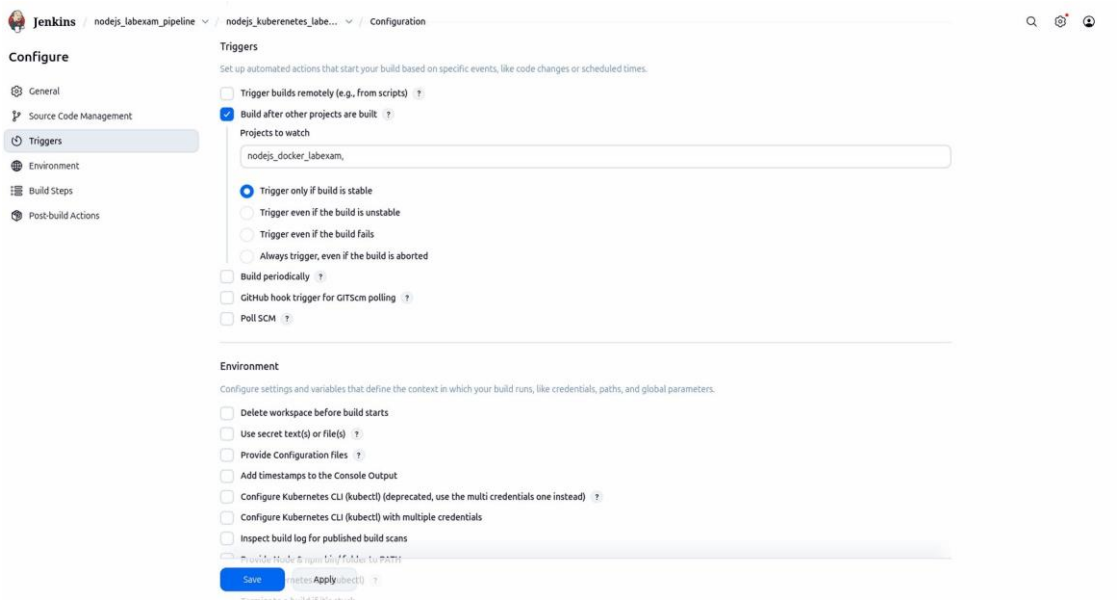
Create pipeline for all these items (Baremetal + Docker + Kubernetes)

Since we have already created the pipeline for Nodejs_Baremetal we now need to add the remaining two stages (nodejs_Docker and nodejs_Kubernetes)

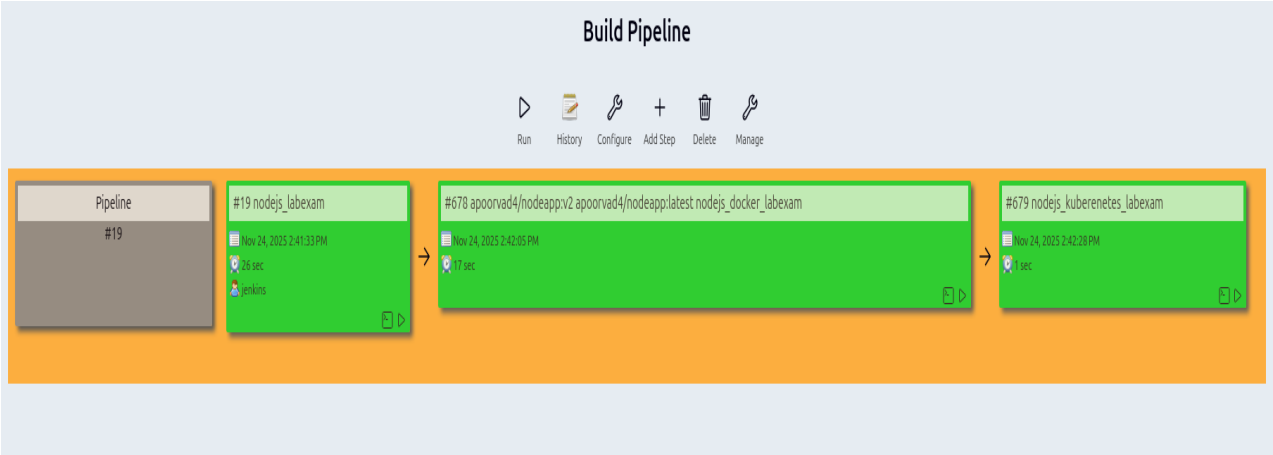
Go to the item named nodejs_docker select post build actions trigger only if build is stable. Then, under Projects to watch, select node_baremetal.



Go to the item named nodejs_kubernetes select post build actions trigger only if build is stable. Then, under Projects to watch, select node_docker



Now the pipeline creation is complete



ANSIBLE

Create Ansible playbook to create Build Server Environment or Docker Server Environment or Kubernetes Environment for the above-Mentioned Modules.

1. Purge apache2 in host systems

   $ sudo apt-get purge --autoremove apache2

2. Clone ansible repo in main system

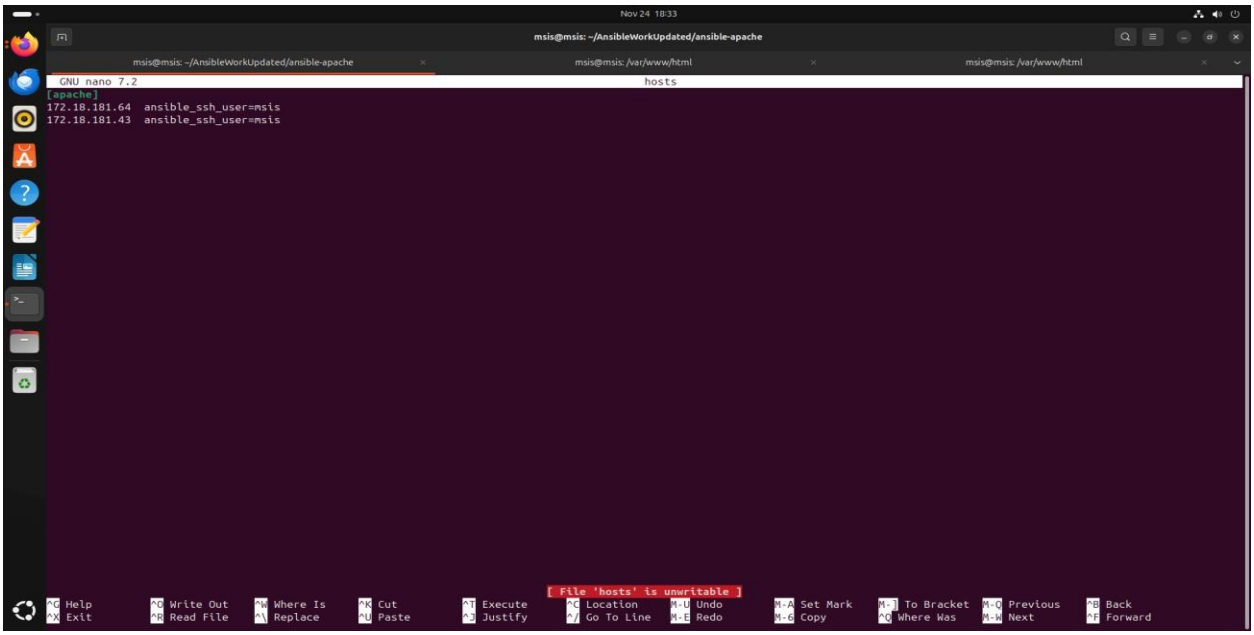
   $ sudo git clone https://github.com/sreepathysois/AnsibleWorkUpdated.git

3. Run the ansible-apache

   - Go to AnsibleWorkUpdated/ansible-apache

$cd AnsibleWorkUpdated/ansible-apache

   - Check hosts and make it as we want save it



   - Make the it passwordless to the slave nodes mentioned in hosts

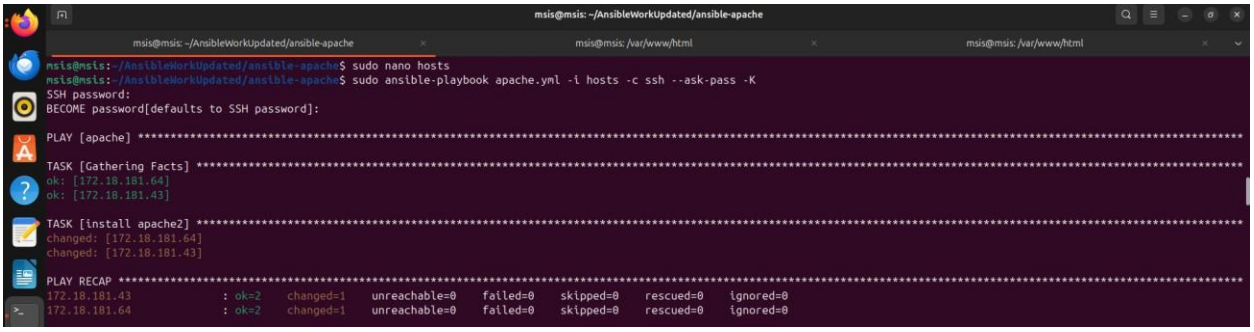
      $ sudo ssh-keygen -t RSA

      $ sudo ssh-copy-id -i msiskey.pub msis@172.18.181.64

      $ sudo ssh-copy-id -i msiskey.pub msis@172.18.181.43

- run the apache.yml

$ sudo ansible-playbook apache.yml -i hosts -c ssh --ask-pass –K

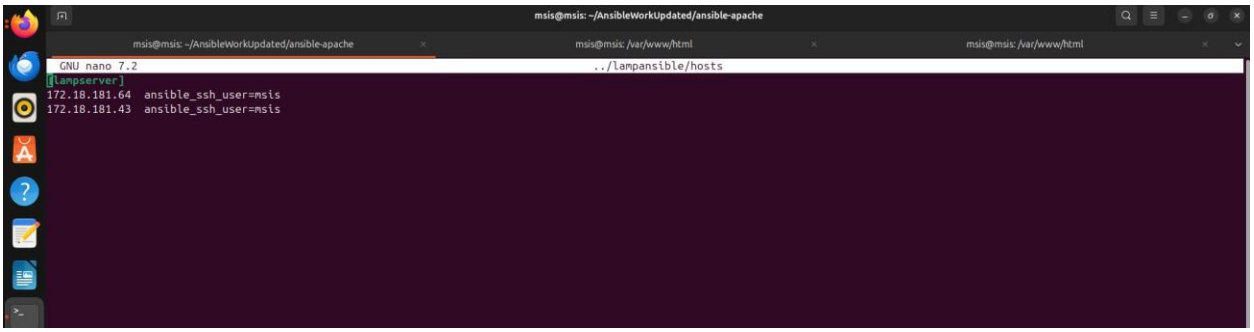


4. Run the lamp.yml

- go to lampansible

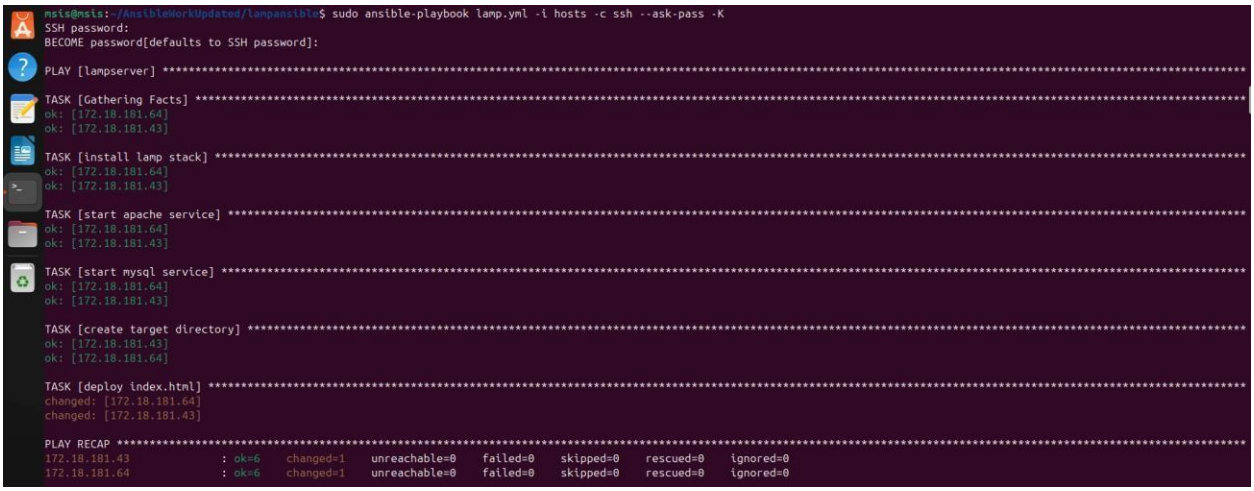
$cd ../lampansible

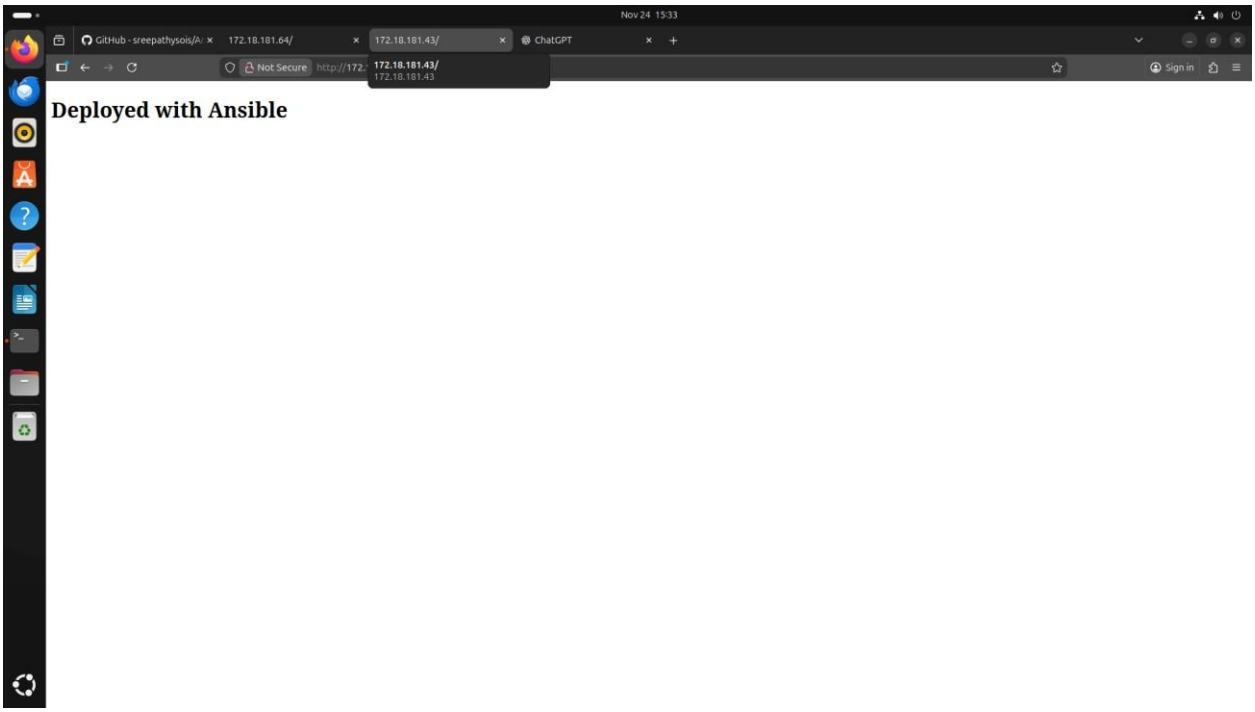
-confirm the hosts



- run the lamp.yml

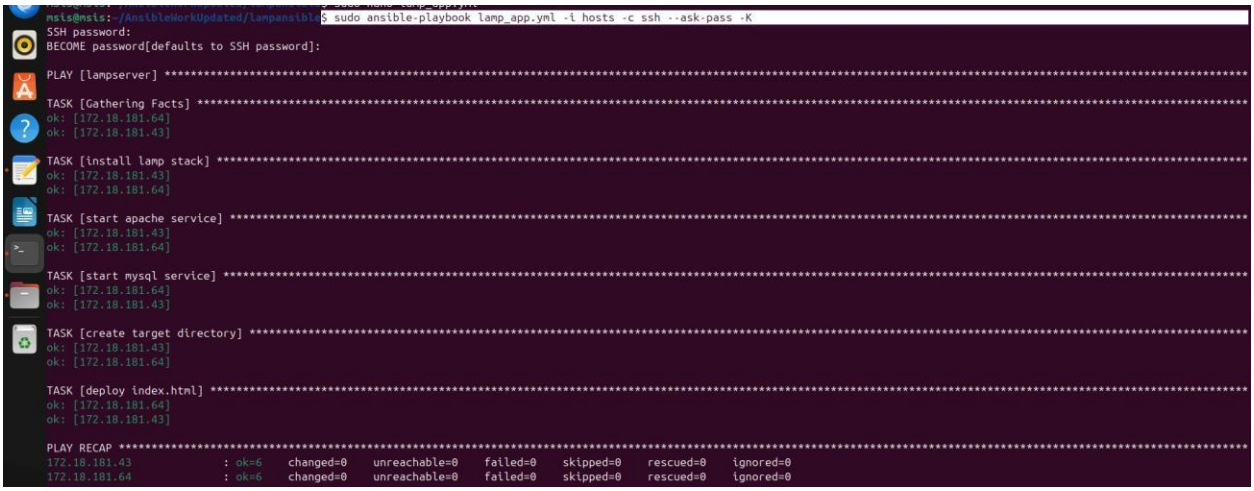
$ sudo ansible-playbook lamp_app.yml -i hosts -c ssh --ask-pass –K

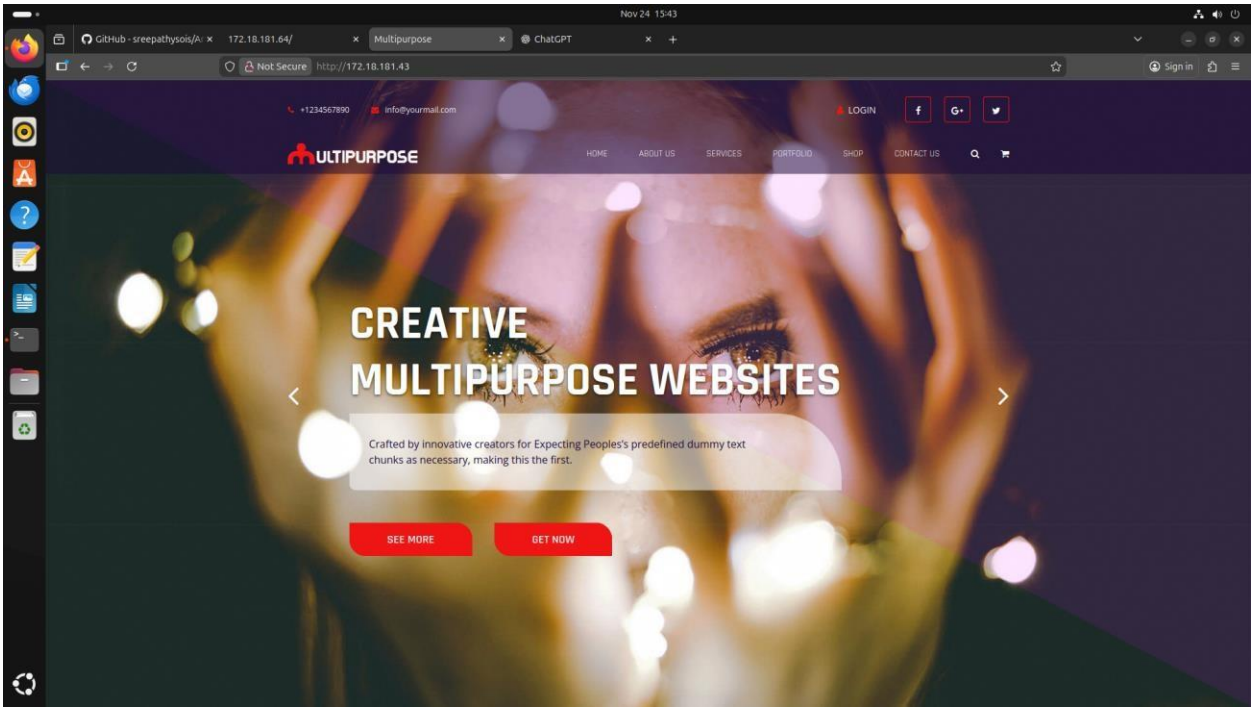




5. Run the lamp_app.yml

$ sudo ansible-playbook lamp_app.yml -i hosts -c ssh --ask-pass -K



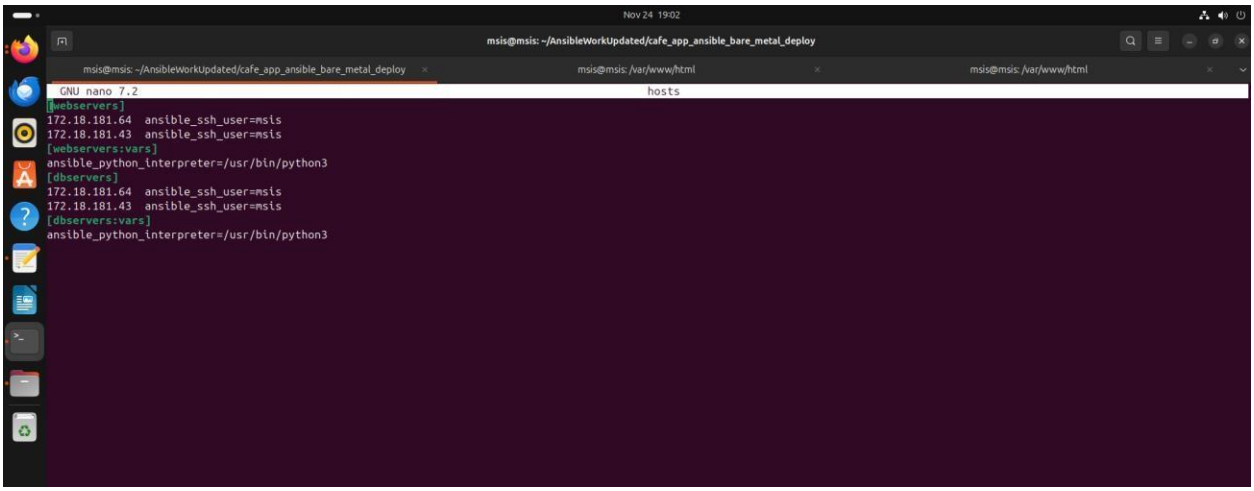


6. Run the baremetal

- go to cafe_app_ansible_bare_metal_deploy

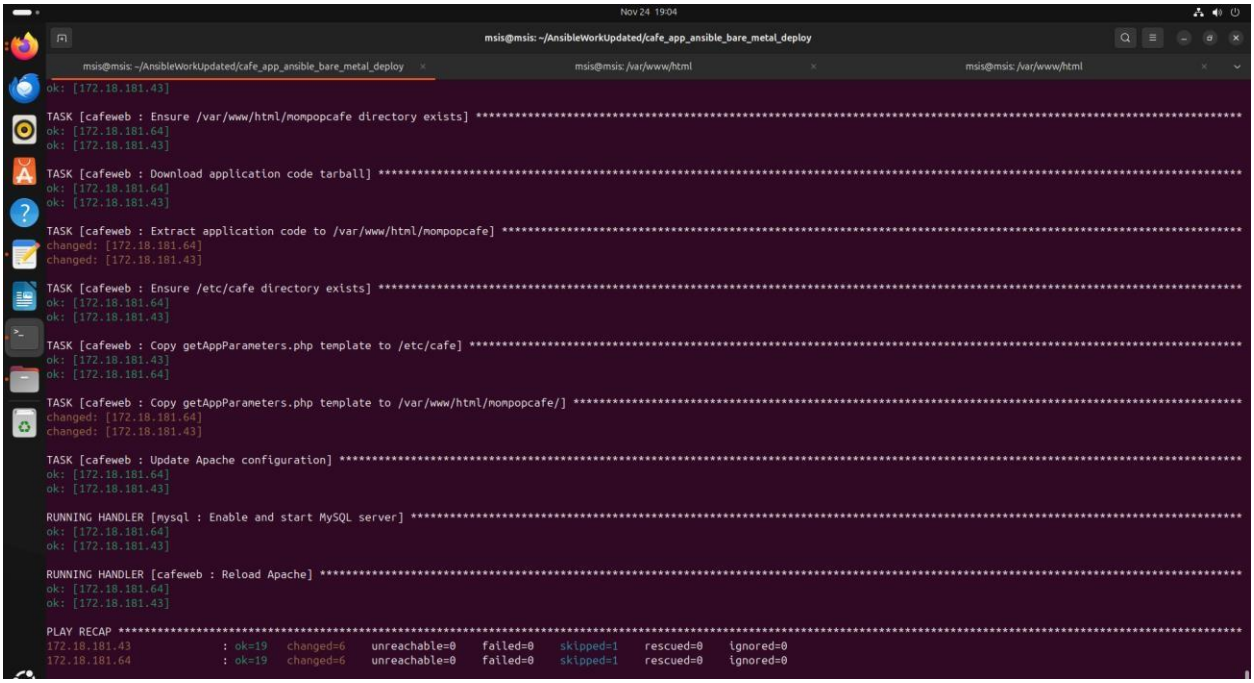
$cd ../ cafe_app_ansible_bare_metal_deploy

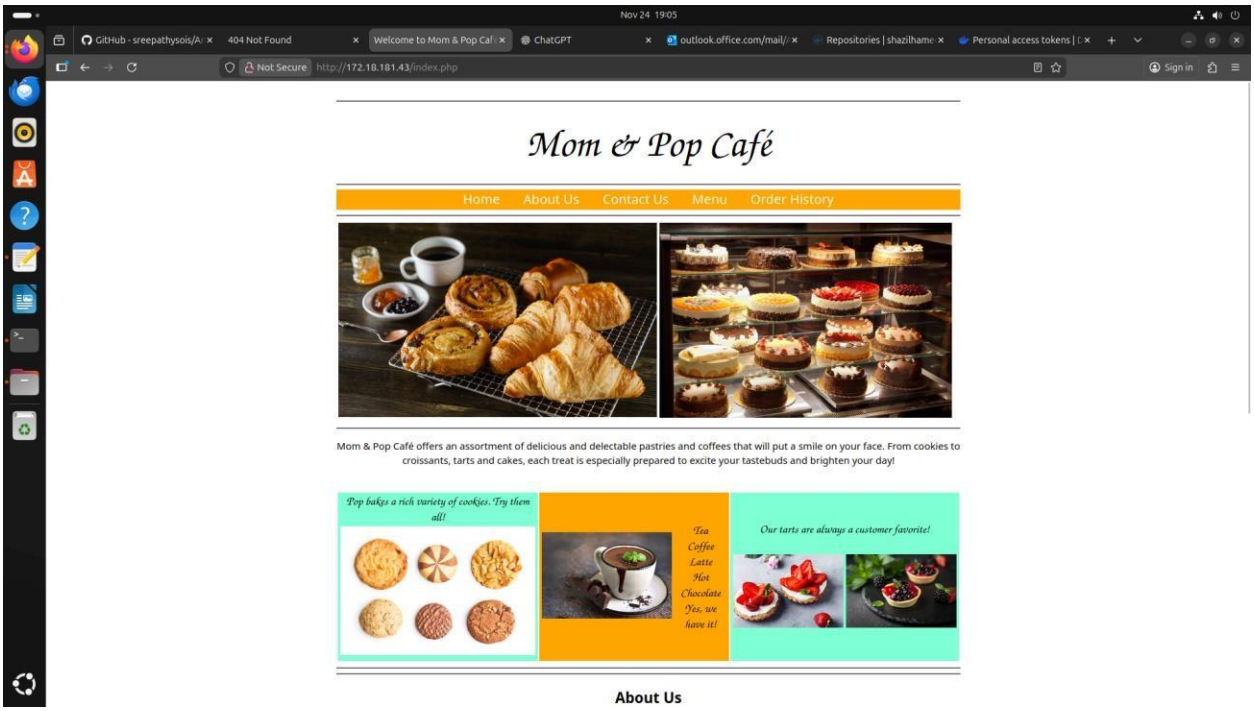
-confirm with hosts

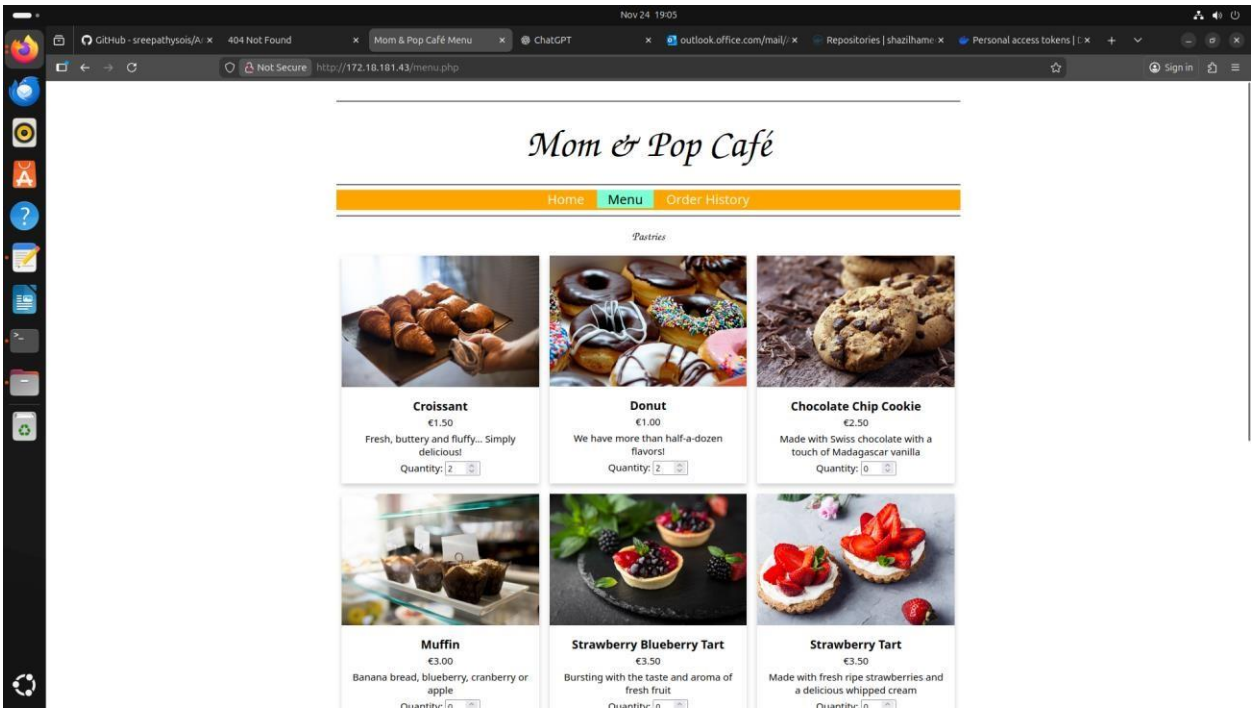


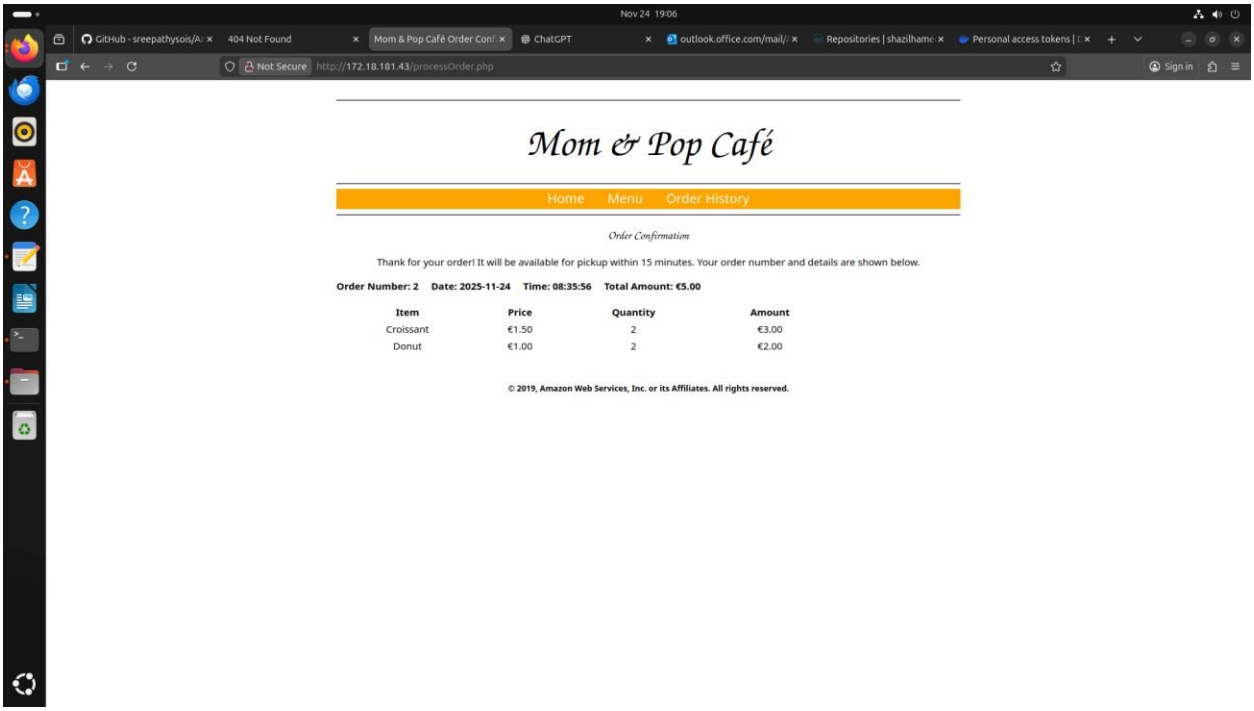
-run the baremetal

$sudo ansible-playbook playbook.yml -i hosts -c ssh --ask-pass -K




Nov 24 19:05
GitHub - sreepathysois/A
404 Not Found
Welcome to Mom & Pop Caf
ChatGPT
outlook.office.com/mail/
Repositories | shazilhame
Personal access tokens |
Not Secure
http://172.18.181.43/index.php
Sign in
Mom & Pop Café
Home
About Us
Contact Us
Menu
Order History
Mom & Pop Café offers an assortment of delicious and delectable pastries and coffees that will put a smile on your face. From cookies to croissants, tarts and cakes, each treat is especially prepared to excite your tastebuds and brighten your day!
Pop bakes a rich variety of cookies. Try them all!
Tea
Coffee
Latte
Hot Chocolate
Yes, we have it!
Our tarts are always a customer favorite!
About Us


Nov 24 19:05
GitHub - sreepathysois/A
404 Not Found
Mom & Pop Café Menu
ChatGPT
outlook.office.com/mail/
Repositories | shazilhame
Personal access tokens |
Not Secure
http://172.18.181.43/menu.php
Sign in
Mom & Pop Café
Home
Menu
Order History
Pastries
Croissant
€1.50
Fresh, buttery and fluffy... Simply delicious!
Quantity: 2
Donut
€1.00
We have more than half-a-dozen flavors!
Quantity: 2
Chocolate Chip Cookie
€2.50
Made with Swiss chocolate with a touch of Madagascar vanilla
Quantity: 0
Muffin
€3.00
Banana bread, blueberry, cranberry or apple
Strawberry Blueberry Tart
€3.50
Bursting with the taste and aroma of fresh fruit
Strawberry Tart
€3.50
Made with fresh ripe strawberries and a delicious whipped cream

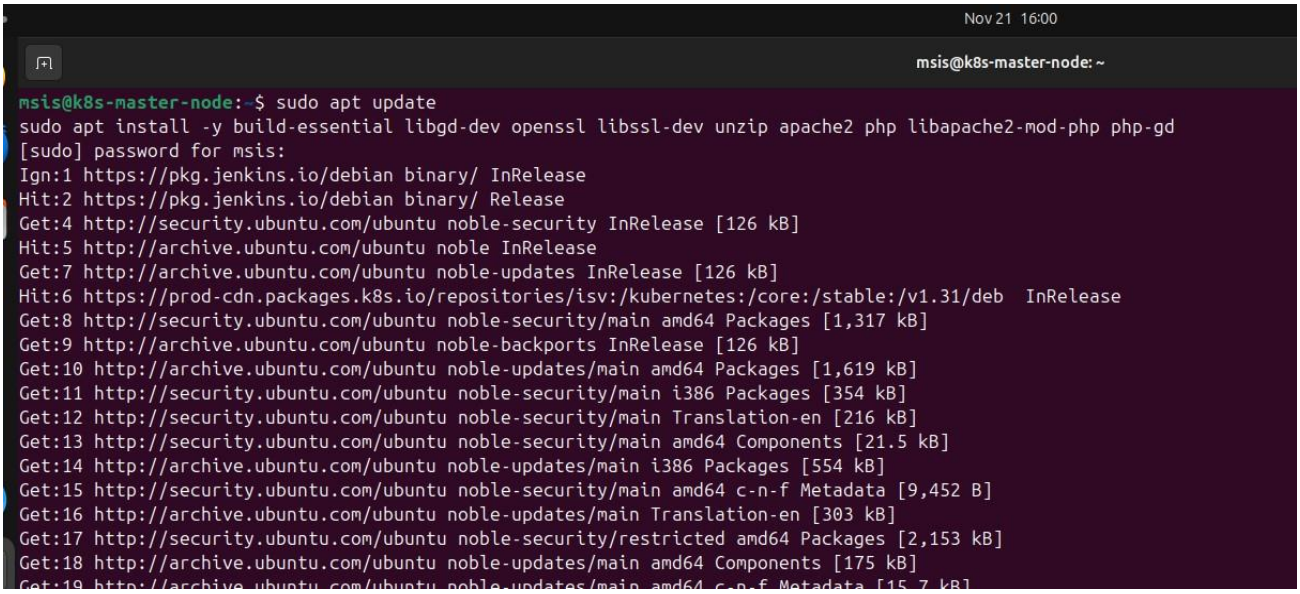

Nov 24 19:06
GitHub - sreepathysois/A
404 Not Found
Mom & Pop Café Order Con
ChatGPT
outlook.office.com/mail/
Repositories | shazilhame
Personal access tokens |
Not Secure
http://172.18.181.43/processOrder.php
Sign in
Mom & Pop Café
Home
Menu
Order History
Order Confirmation
Thank for your order! It will be available for pickup within 15 minutes. Your order number and details are shown below.
Order Number: 2 Date: 2025-11-24 Time: 08:35:56 Total Amount: €5.00
Item
Price
Quantity
Amount
Croissant
€1.50
2
€3.00
Donut
€1.00
2
€2.00
© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

CONTINUOUS MONITORING – NagiOS

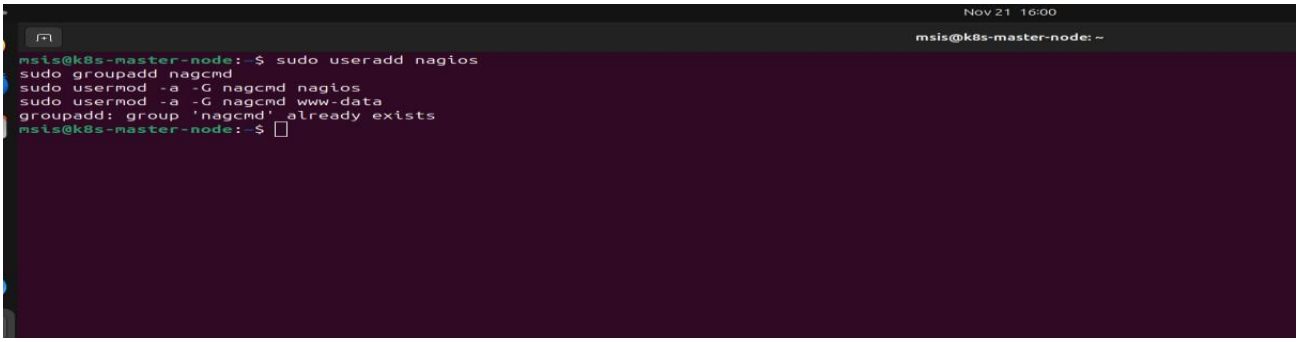
Create Continuous Monitoring Services using Nagios to Monitor Jenkins slave nodes (Build Server) or Docker hosts (Pre-Production Servers) or Kubernetes cluster nodes (Production Servers) which host web servers and databases. (Example: Monitor CPU load, Disk space, RAM usage, Processes running, Apache server running status, mysql connection etc.)

Install Nagios on Ubuntu 24.04 LTS

1. Install Nagios Dependencies packages on Ubuntu 24.04 LTS



2. Create a Nagios User and Group in Ubuntu 24.04 LTS



3. Download and Compile Nagios on Ubuntu 24.04 LTS

Configure and compile Nagios:

```
msis@k8s-master-node:/tmp/nagios-4.5.1$ ./configure --with-nagios-group=nagios --with-command-group=nagcmd
make all
checking for a BSD-compatible install... /usr/bin/install -c
checking build system type... x86_64-pc-linux-gnu
checking host system type... x86_64-pc-linux-gnu
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
checking whether make sets $(MAKE)... yes
checking whether ln -s works... yes
checking for strip... /usr/bin/strip
checking how to run the C preprocessor... gcc -E
checking for grep that handles long lines and -e... /usr/bin/grep
checking for egrep... /usr/bin/grep -E
checking for ANSI C header files... yes
checking whether time.h and sys/time.h may both be included... yes
checking for sys/wait.h that is POSIX.1 compatible... yes
```

4. Install Nagios on Ubuntu 24.04 LTS

```
msis@k8s-master-node:/tmp/nagios-4.5.1$ sudo make install
sudo make install-init
sudo make install-commandmode
sudo make install-config
sudo make install-webconf
cd ./base && make install
make[1]: Entering directory '/tmp/nagios-4.5.1/base'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/bin
/usr/bin/install -c -s -m 774 -o nagios -g nagios nagios /usr/local/nagios/bin
/usr/bin/install -c -s -m 774 -o nagios -g nagios nagiostats /usr/local/nagios/bin
make[1]: Leaving directory '/tmp/nagios-4.5.1/base'
cd ./cgi && make install
make[1]: Entering directory '/tmp/nagios-4.5.1/cgi'
make install-basic
make[2]: Entering directory '/tmp/nagios-4.5.1/cgi'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/sbin
for file in *.cgi; do \
        /usr/bin/install -c -s -m 775 -o nagios -g nagios $file /usr/local/nagios/sbin; \
done
make[2]: Leaving directory '/tmp/nagios-4.5.1/cgi'
make[1]: Leaving directory '/tmp/nagios-4.5.1/cgi'
cd ./html && make install
make[1]: Entering directory '/tmp/nagios-4.5.1/html'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/media
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/stylesheets
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/contexthelp
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/docs
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/docs/images
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/js
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/images
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/images/logos
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/includes
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/ssi
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/angularjs
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/angularjs/angular-1.3.9
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/angularjs/ui-utils-0.2.3
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/bootstrap-3.3.7
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/bootstrap-3.3.7/css
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/d3
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/spin
/usr/bin/install -c -m 664 -o nagios -g nagios ./robots.txt /usr/local/nagios/share
rm -f /usr/local/nagios/share/index.html
rm -f /usr/local/nagios/share/main.html
```

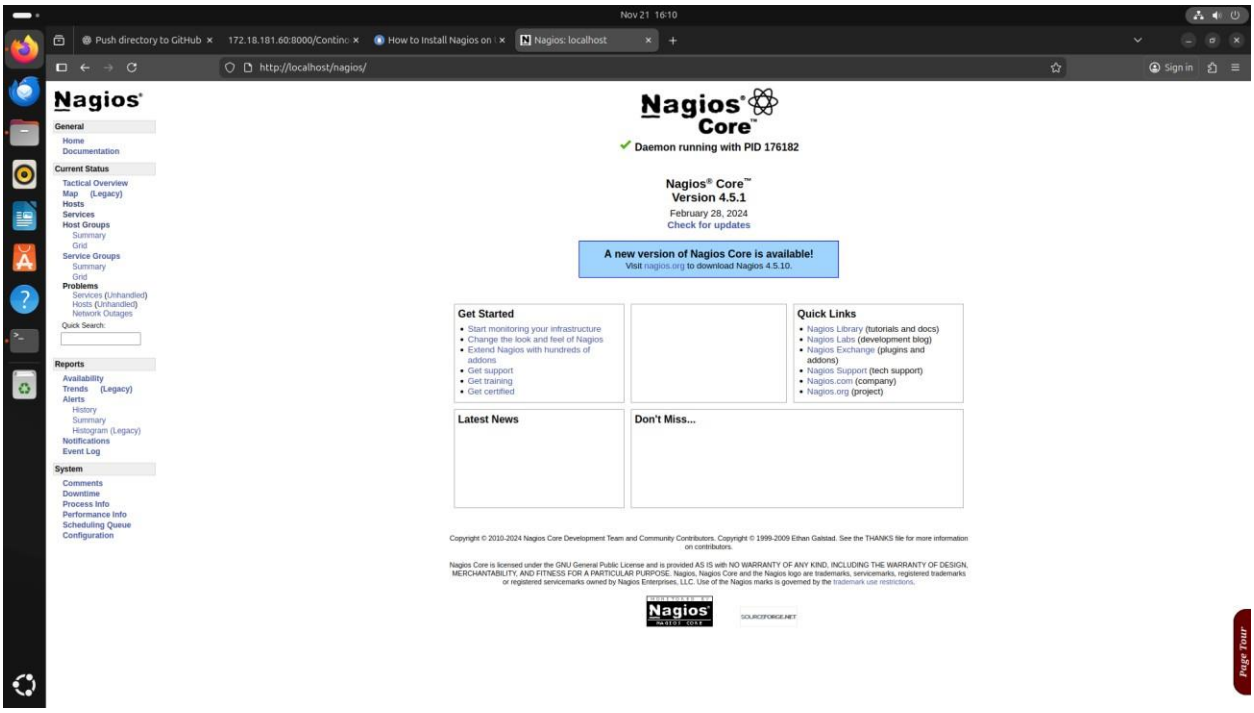
5. Install Nagios Plugins on Ubuntu 24.04 LTS

```
msis@k8s-master-node:/tmp/nagios-4.5.1$ cd /tmp
wget https://nagios-plugins.org/download/nagios-plugins-2.3.3.tar.gz
tar xzf nagios-plugins-2.3.3.tar.gz
cd nagios-plugins-2.3.3
./configure --with-nagios-user=nagios --with-nagios-group=nagios
make
sudo make install
--2025-11-21 16:04:16--  https://nagios-plugins.org/download/nagios-plugins-2.3.3.tar.gz
Resolving nagios-plugins.org (nagios-plugins.org)... 45.56.123.251
Connecting to nagios-plugins.org (nagios-plugins.org)|45.56.123.251|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2782610 (2.7M) [application/x-gzip]
Saving to: ‘nagios-plugins-2.3.3.tar.gz’

nagios-plugins-2.3.3.tar.gz                 100%[=========================================================================================>]   2.65M  1.36MB/s    in 1.9s

2025-11-21 16:04:19 (1.36 MB/s) - ‘nagios-plugins-2.3.3.tar.gz’ saved [2782610/2782610]

checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... /usr/bin/mkdir -p
checking for gawk... no
checking for mawk... mawk
checking whether make sets $(MAKE)... yes
checking whether to disable maintainer-specific portions of Makefiles... yes
checking build system type... x86_64-unknown-linux-gnu
checking host system type... x86_64-unknown-linux-gnu
checking for gcc... gcc
checking for C compiler default output file name... a.out
checking whether the C compiler works... yes
checking whether we are cross compiling... no
checking for suffix of executables...
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
```

6. Configure Apache on Ubuntu 24.04 LTS
7. Configure Nagios Web Interface on Ubuntu 24.04 LTS
8. Start Nagios Service on Ubuntu 24.04 LTS command line

```
msis@k8s-master-node:/tmp/nagios-plugins-2.3.3$ sudo a2enmod rewrite
sudo a2enmod cgi
sudo systemctl restart apache2
Module rewrite already enabled
Module cgi already enabled
Warning: The unit file, source configuration file or drop-ins of apache2.service changed on disk. Run 'systemctl daemon-reload' to reload units.
msis@k8s-master-node:/tmp/nagios-plugins-2.3.3$ systemctl daemon-reload
msis@k8s-master-node:/tmp/nagios-plugins-2.3.3$ sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin
New password:
Re-type new password:
Adding password for user nagiosadmin
msis@k8s-master-node:/tmp/nagios-plugins-2.3.3$ sudo systemctl start nagios
sudo systemctl enable nagios
msis@k8s-master-node:/tmp/nagios-plugins-2.3.3$
```

9. Access Nagios Web Interface on Ubuntu 24.04 LTS
   Log in with the username and the password you set



10. Add this line to /usr/local/nagios/etc/nagios.cfg

```
msis@k8s-master-node:/tmp/nagios-plugins-2.3.3$ sudo nano /usr/local/nagios/etc/nagios.cfg
msis@k8s-master-node:/tmp/nagios-plugins-2.3.3$
```

```
# You can also tell Nagios to process all config files (with a .cfg
# extension) in a particular directory by using the cfg_dir
# directive as shown below:

cfg_dir=/usr/local/nagios/etc/servers
#cfg_dir=/usr/local/nagios/etc/printers
#cfg_dir=/usr/local/nagios/etc/switches
#cfg_dir=/usr/local/nagios/etc/routers
```

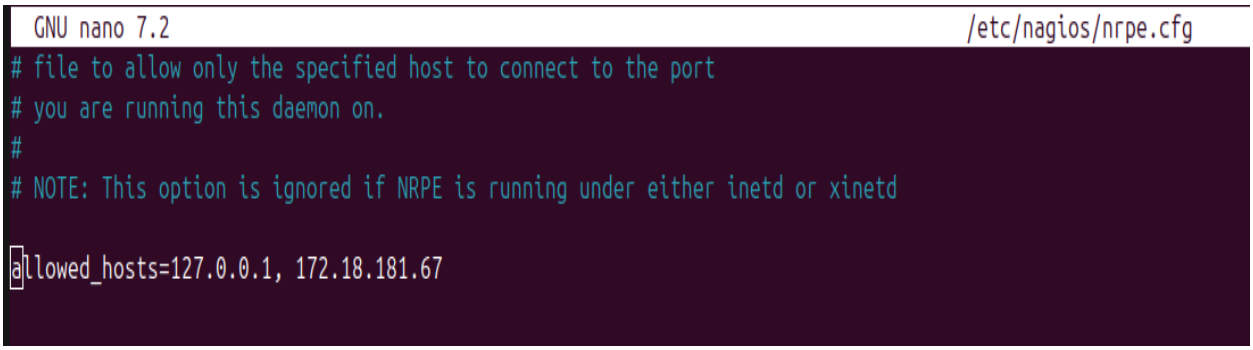
Monitor Remote Linux Host Using Nagios:

11. sudo apt update && sudo apt install nagios-nrpe-server nagios-plugins

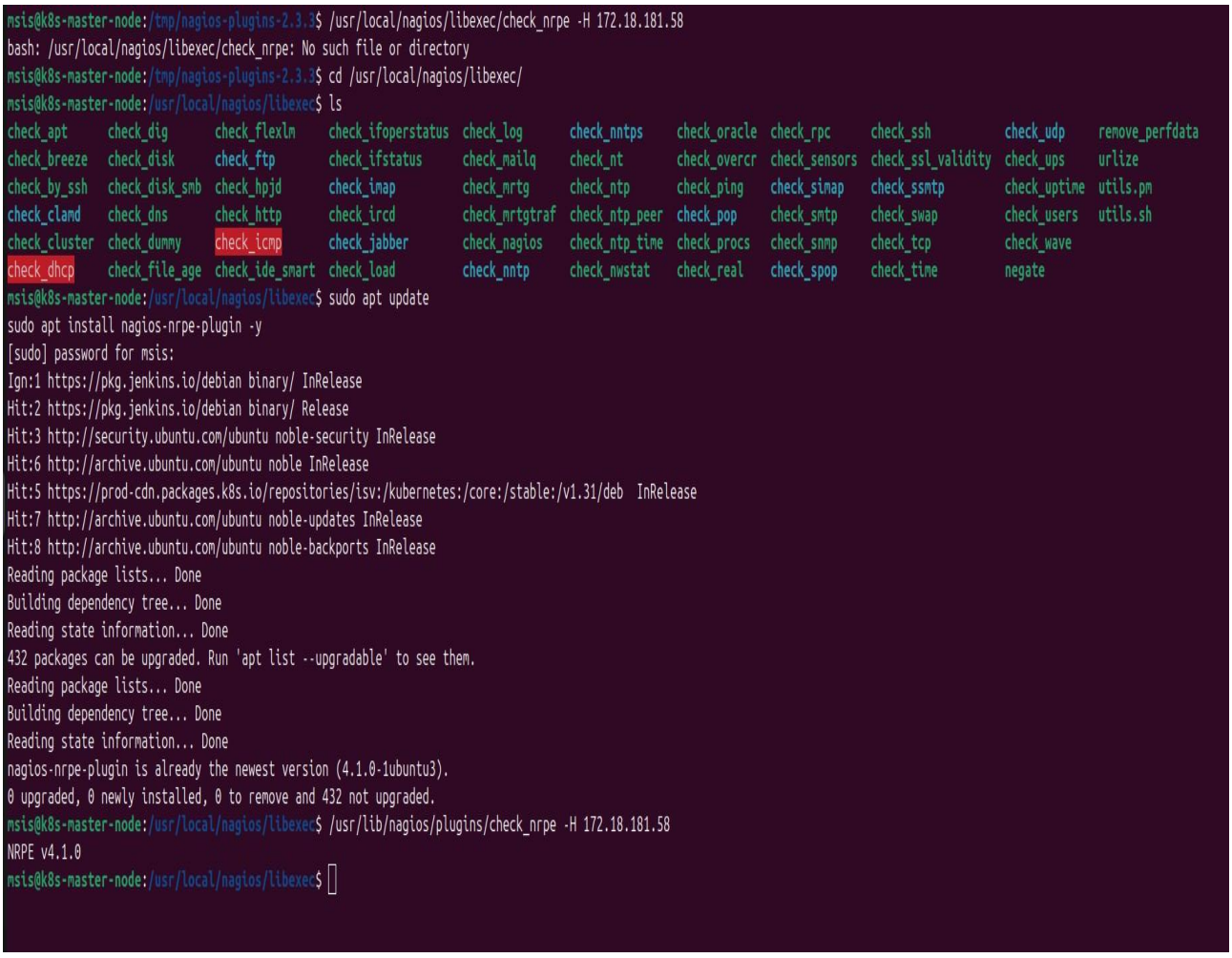
```
msis@k8s-worker-node-1:~$ sudo apt update && sudo apt install nagios-nrpe-server nagios-plugins
[sudo] password for msis:
Hit:1 https://brave-browser-apt-release.s3.brave.com stable InRelease
Hit:2 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  InRelease
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Hit:5 http://archive.ubuntu.com/ubuntu noble InRelease
Ign:3 https://pkg.jenkins.io/debian-stable binary/ InRelease
Get:6 http://archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Hit:7 https://pkg.jenkins.io/debian-stable binary/ Release
Get:9 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [1,317 kB]
Get:10 http://archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:11 http://security.ubuntu.com/ubuntu noble-security/main i386 Packages [354 kB]
Get:12 http://archive.ubuntu.com/ubuntu noble-updates/main i386 Packages [554 kB]
Get:13 http://security.ubuntu.com/ubuntu noble-security/main Translation-en [216 kB]
Get:14 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [1,619 kB]
Get:15 http://security.ubuntu.com/ubuntu noble-security/main amd64 Components [21.5 kB]
Get:16 http://security.ubuntu.com/ubuntu noble-security/main amd64 c-n-f Metadata [9,452 B]
Get:17 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Packages [2,153 kB]
Get:18 http://archive.ubuntu.com/ubuntu noble-updates/main Translation-en [303 kB]
Get:19 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 Components [175 kB]
Get:20 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 c-n-f Metadata [15.7 kB]
Get:21 http://security.ubuntu.com/ubuntu noble-security/restricted Translation-en [490 kB]
Get:22 http://archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Packages [2,307 kB]
Get:23 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Components [212 B]
Get:24 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [909 kB]
Get:25 http://security.ubuntu.com/ubuntu noble-security/universe i386 Packages [565 kB]
Get:26 http://archive.ubuntu.com/ubuntu noble-updates/restricted Translation-en [526 kB]
Get:27 http://security.ubuntu.com/ubuntu noble-security/universe Translation-en [205 kB]
Get:28 http://archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Components [212 B]
Get:29 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Components [52.2 kB]
Get:30 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [1,500 kB]
Get:31 http://security.ubuntu.com/ubuntu noble-security/universe amd64 c-n-f Metadata [19.4 kB]
Get:32 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Components [208 B]
Get:33 http://archive.ubuntu.com/ubuntu noble-updates/universe i386 Packages [989 kB]
Get:34 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 Components [378 kB]
Get:35 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 c-n-f Metadata [31.4 kB]
Get:36 http://archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Components [940 B]
Get:37 http://archive.ubuntu.com/ubuntu noble-backports/main amd64 Components [7,156 B]
Get:38 http://archive.ubuntu.com/ubuntu noble-backports/restricted amd64 Components [212 B]
Get:39 http://archive.ubuntu.com/ubuntu noble-backports/universe amd64 Components [11.0 kB]
Get:40 http://archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 Components [212 B]
Fetched 15.1 MB in 8s (1,857 kB/s)
```

12. vim /etc/nagios/nrpe.cfg

allowed_hosts=127.0.0.1, 172.18.181.67



13. /usr/local/nagios/libexec/check_nrpe -H 172.18.181.67



14. sudo vim /usr/local/nagios/etc/servers/MyLinuxHost001.cfg
15. nagios -v /usr/local/nagios/etc/nagios.cfg
    service nagios restart

```
msis@k8s-master-node:/usr/local/nagios/libexec$ sudo vim /usr/local/nagios/etc/servers/MyLinuxHost001.cfg
msis@k8s-master-node:/usr/local/nagios/libexec$ nagios -v /usr/local/nagios/etc/nagios.cfg
service nagios restart
Command 'nagios' not found, did you mean:
  command 'nagios4' from deb nagios4-core (4.4.6-4ubuntu0.24.04.1)
Try: sudo apt install <deb name>
msis@k8s-master-node:/usr/local/nagios/libexec$ service nagios4 restart
Failed to restart nagios4.service: Unit nagios4.service not found.
msis@k8s-master-node:/usr/local/nagios/libexec$ sudo vim /usr/local/nagios/etc/servers/MyLinuxHost001.cfg
msis@k8s-master-node:/usr/local/nagios/libexec$ sudo systemctl restart nagios
msis@k8s-master-node:/usr/local/nagios/libexec$
```

```
################################################
# Linux Host 001 configuration file
################################################

define host {
        use                     linux-server
        host_name               Linux_Host_001
        alias                   Linux Host 001
        address                 172.18.181.58
        register                1
}
define service{
      host_name                  Linux_Host_001
      service_description        PING
      check_command              check_ping!100.0,20%!500.0,60%
      max_check_attempts         2
      check_interval             2
      retry_interval             2
      check_period               24x7
      check_freshness            1
      contact_groups             admins
      notification_interval      2
      notification_period        24x7
      notifications_enabled      1
      register                   1
}


################################################
# END OF FILE
################################################
~
~
~
```

16. Check Host in Nagios Web Interface