

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 serve 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)

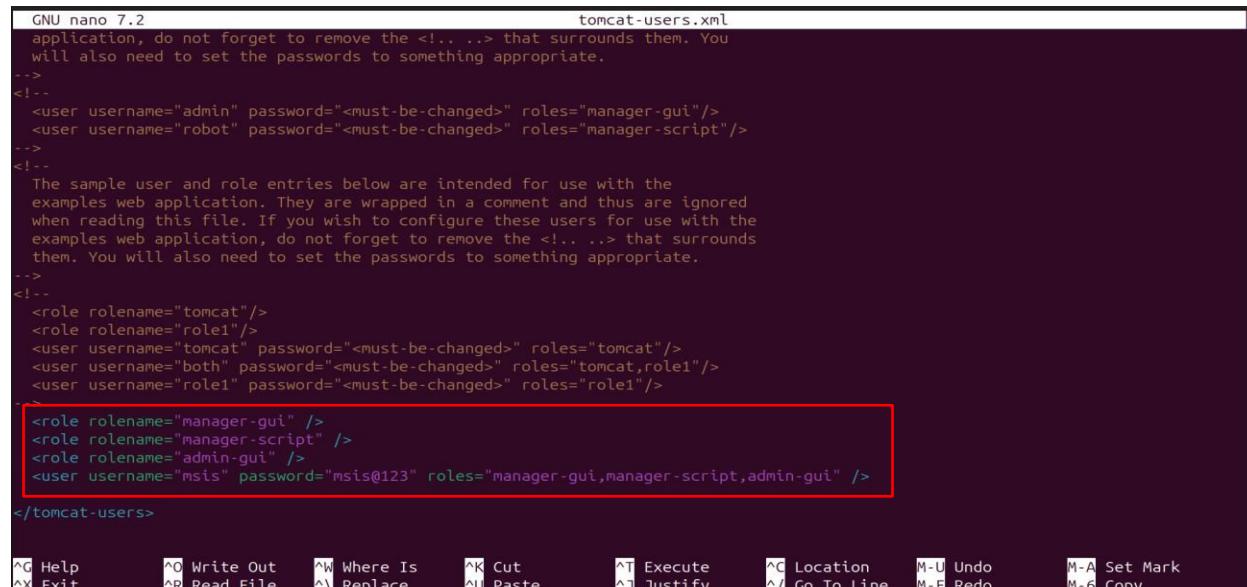
For Java

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
```



```
GNU nano 7.2                                     tomcat-users.xml
application, do not forget to remove the <!... ...> that surrounds them. You
will also need to set the passwords to something appropriate.
-->
<!--
<user username="admin" password="" roles="manager-gui"/>
<user username="robot" password="" roles="manager-script"/>
-->
<!--
The sample user and role entries below are intended for use with the
examples web application. They are wrapped in a comment and thus are ignored
when reading this file. If you wish to configure these users for use with the
examples web application, do not forget to remove the <!... ...> that surrounds
them. You will also need to set the passwords to something appropriate.
-->
<!--
<role rolename="tomcat"/>
<role rolename="role1"/>
<user username="tomcat" password="" roles="tomcat"/>
<user username="both" password="" roles="tomcat,role1"/>
<user username="role1" password="" roles="role1"/>
<!--
<role rolename="manager-gui" />
<role rolename="manager-script" />
<role rolename="admin-gui" />
<user username="msis" password="msis@123" roles="manager-gui,manager-script,admin-gui" />
-->
</tomcat-users>

^G Help      ^O Write Out   ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo    M-A Set Mark
^X Exit      ^R Read File   ^\ Replace    ^U Paste     ^J Justify   ^/ Go To Line M-E Redo    M-6 Copy
```

```
cd /opt/tomcat/conf
```

```
sudo nano server.xml
```

```

GNU nano 7.2                                server.xml

<!-- A "Connector" represents an endpoint by which requests are received
and responses are returned. Documentation at :
Java HTTP Connector: /docs/config/http.html
Java AJP Connector: /docs/config/ajp.html
APR (HTTP/AJP) Connector: /docs/apr.html
Define a non-SSL/TLS HTTP/1.1 Connector on port 8080
-->
<Connector port="8081" protocol="HTTP/1.1" address="0.0.0.0"
           connectionTimeout="20000"
           redirectPort="8443"
           maxParameterCount="1000"
           />
<!-- A "Connector" using the shared thread pool-->
<!--
<Connector executor="tomcatThreadPool"
            port="8080" protocol="HTTP/1.1"
            connectionTimeouts="20000"
            redirectPort="8443"
            maxParameterCount="1000"
            />
-->
<!-- Define an SSL/TLS HTTP/1.1 Connector on port 8443
This connector uses the NIO implementation. The default
SSLIImplementation will depend on the presence of the APR/native
library and the useOpenSSL attribute of the AprLifecycleListener.
Either JSSE or OpenSSL style configuration may be used regardless of

```

^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location M-U Undo
 ^X Exit ^R Read File ^V Replace ^U Paste ^J Justify ^Y Go To Line M-E Redo
 M-A Set Mark M-G Copy

Plugins required:

- Maven Integration plugin
- Warnings: For Code review
- JUnit: For Testing (if required)
- Deploy to container

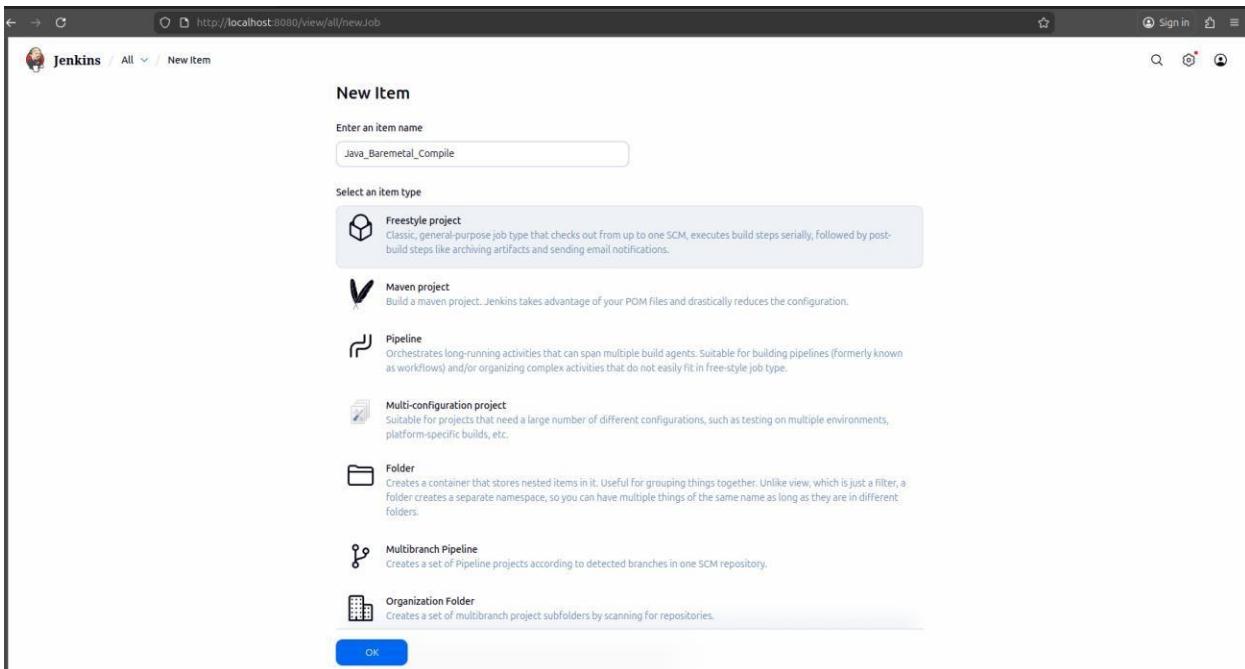
The above plugins can be installed by navigating to Manage Jenkins->Plugins.

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.

Configure

General

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/java-tomcat-maven-example.git

Credentials
-none-

+ Add

Advanced

+ Add Repository

Branches to build

Branch Specifier (blank for 'any')
*/master

+ Add Branch

Repository browser

Save

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

The screenshot shows the Jenkins configuration page for a job named "Java_Baremetal_Compile". The left sidebar includes options like General, Source Code Management, Triggers, Environment, Build Steps (which is selected), and Post-build Actions. The main content area is titled "Configuration" and contains sections for "Build Steps" and "Post-build Actions". Under "Build Steps", there is a step titled "Invoke top-level Maven targets" with fields for "Maven Version" (set to "my_maven") and "Goals" (set to "validate compile"). There is also an "Advanced" dropdown and a "+ Add build step" button. Under "Post-build Actions", there is a "+ Add post-build action" button. At the bottom are "Save" and "Apply" buttons.

Step2: Code review

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

The screenshot shows the "New Item" dialog in Jenkins. It has a title "New Item" and a sub-section "Enter an item name" with the value "Java_Baremetal_Code Review". Below it is a "Select an item type" section with several options: "Freestyle project" (selected), "Maven project", "Pipeline", "Multi-configuration project", "Folder", "Multibranch Pipeline", and "Organization Folder". Each option has a small icon and a brief description. At the bottom of the dialog is an "OK" button.

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

The screenshot shows the Jenkins configuration page for a job named "Java_Baremetal_Code Review". Under the "Source Code Management" section, "Git" is selected as the provider. The "Repository URL" is set to `https://github.com/sreepathysois/java-tomcat-maven-example.git`. The "Branches to build" field contains the specifier `*/*:master`. There are "Save" and "Apply" buttons at the bottom.

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.”

The screenshot shows the Jenkins configuration page for triggers. Under the "Triggers" section, "Build after other projects are built" is checked. In the "Projects to watch" field, "Java_Baremetal_Compile" is listed. Under "Trigger only if build is stable", the radio button is selected. There are "Save" and "Apply" buttons at the bottom.

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.

The screenshot shows the Jenkins job configuration page for 'Java_Baremetal_Code Review'. The left sidebar lists 'Configure' sections: General, Source Code Management, Triggers, Environment, Build Steps (selected), and Post-build Actions. The main content area has two expanded sections: 'Build Steps' and 'Post-build Actions'.
Build Steps: Contains a step titled 'Invoke top-level Maven targets'. It includes fields for 'Maven Version' (set to 'my_maven') and 'Goals' (set to '-P metrics pmd:pmd checkstyle:checkstyle findbugs:findbugs'). A 'Goals' dropdown shows additional options like 'metrics:report'. An 'Advanced' button is present.
Post-build Actions: Contains a step titled 'Record compiler warnings and static analysis results'. It includes a 'Static Analysis Tools' dropdown set to 'PMD'. Other fields include 'Tool' (PMD), 'Report File Pattern' (empty), 'Skip symbolic links when searching for files' (unchecked), 'Encoding of Report Files' (empty), 'Configure the context lines' (set to 3), and 'Custom ID' (empty).
At the bottom are 'Save' and 'Apply' buttons.

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

This screenshot shows the same Jenkins configuration page, but the 'Post-build Actions' section is now fully expanded. The 'Static Analysis Tools' dropdown is set to 'PMD'. Other visible fields include 'Tool' (PMD), 'Report File Pattern' (empty), 'Skip symbolic links when searching for files' (unchecked), 'Encoding of Report Files' (empty), 'Configure the context lines' (set to 3), and 'Custom ID' (empty). The 'Save' and 'Apply' buttons are at the bottom.

Jenkins / Java_Baremetal_Code Review / 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

This screenshot shows the Jenkins configuration page for a job named 'Java_Baremetal_Code Review'. The 'Post-build Actions' section is selected. A 'Tool' dropdown is set to 'CheckStyle'. The configuration includes a report file pattern, options for skipping symbolic links and specifying encoding, and settings for context lines, custom ID, name, and icon. Buttons for 'Save' and 'Apply' are at the bottom.

Jenkins / Java_Baremetal_Code Review / 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

This screenshot shows the Jenkins configuration page for a job named 'Java_Baremetal_Code Review'. The 'Post-build Actions' section is selected. A 'Tool' dropdown is set to 'FindBugs'. The configuration includes a report file pattern, options for skipping symbolic links and specifying encoding, and settings for context lines, a checked checkbox for using bug rank, custom ID, and name. Buttons for 'Save' and 'Apply' are at the bottom.

The screenshot shows the Jenkins configuration interface for a job named "Java_Baremetal_Code_Review". The left sidebar lists configuration sections: General, Source Code Management, Triggers, Environment, Build Steps, and Post-build Actions. The "Post-build Actions" section is currently selected. On the right, there's a panel for configuring a tool, specifically FindBugs. It includes fields for "Custom ID", "Custom Name", and "Custom Icon", each with a question mark icon for help. Below these fields are three buttons: "+ Add Tool", "Advanced", and "+ Add post-build action". At the bottom of the panel are "Save" and "Apply" buttons.

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**.

The screenshot shows the "New Item" creation page. In the "Select an item type" section, the "Freestyle project" option is selected and highlighted with a blue border. Other options listed are "Maven project", "Pipeline", "Multi-configuration project", "Folder", "Multibranch Pipeline", and "Organization Folder". Each option has a small icon and a brief description. At the bottom of the page is an "OK" button.

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.

The screenshot shows the Jenkins job configuration page for 'Java_Baremetal_Package_Deploy'. Under the 'Source Code Management' section, 'Git' is selected as the provider, and the repository URL is set to 'https://github.com/sreepathysois/java-tomcat-maven-example.git'. Under 'Branches to build', the branch specifier is set to '/master'. At the bottom, there are 'Save' and 'Apply' buttons.

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.”

The screenshot shows the Jenkins job configuration page for 'Java_Baremetal_Package_Deploy'. Under the 'Triggers' section, 'Build after other projects are built' is selected. In the 'Projects to watch' field, 'Java_Baremetal_Code Review' is listed. Under 'Trigger only if build is stable', the radio button for 'Trigger only if build is stable' is selected. Other options like 'Trigger even if the build is unstable' and 'Always trigger, even if the build is aborted' are available but not selected. At the bottom, there are 'Save' and 'Apply' buttons.

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

The screenshot shows the Jenkins job configuration page for 'Java_Baremetal_Package...' under the 'Configuration' tab. The 'Build Steps' section is highlighted. It contains a step titled 'Invoke top-level Maven targets' with 'Maven Version' set to 'my_maven' and 'Goals' set to 'package'. Below this is a 'Post-build Actions' section with a button '+ Add post-build action'.

REST API Jenkins 2.524

Under Post-build Actions, select “Deploy to Container.”

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

The screenshot shows the Jenkins 'Manage Jenkins' page. The 'Credentials' link under the 'Security' section is highlighted with a red box. Other sections like 'Appearance', 'Status Information', 'Troubleshooting', and 'Tools and Actions' are also visible.

Click on Global.

The screenshot shows the Jenkins 'Credentials' management page at <http://localhost:8080/manage/credentials/>. The page has a header with 'Manage Jenkins' and 'Credentials'. Below the header is a table titled 'Credentials' with columns: T, P, Store, Domain, ID, and Name. The table lists seven entries:

T	P	Store	Domain	ID	Name
System	System	(global)		tomcat_manager_script_cred_38	msis/***** (tomcat_manager_script_cred_38)
System	System	(global)		Docker	sheetalshetty/*****
System	System	(global)		Kubernetes	Kubernetes
System	System	(global)		test	test
System	System	(global)		minikube	minikube
System	System	(global)		minikube2	minikube2
System	System	(global)		minikube3	minikube3

Below the table is a section titled 'Stores scoped to Jenkins' with a table:

P	Store	Domains
System	System	(global)
Kubernetes	Kubernetes	(global)

At the bottom left are icons for S, M, and L. On the right are links for 'REST API' and 'Jenkins 2.524'.

Click on Add Credential

The screenshot shows the Jenkins 'Global credentials (unrestricted)' management page at http://localhost:8080/manage/credentials/stores/system/domain/_/. The page has a header with 'Manage Jenkins' and 'Global credentials (unrest...'. Below the header is a table titled 'Global credentials (unrestricted)' with columns: ID, Name, Kind, and Description. The table lists seven entries:

ID	Name	Kind	Description
tomcat_manager_script_cred_38	msis/***** (tomcat_manager_script_cred_38)	Username with password	tomcat_manager_script_cred_38
Docker	sheetalshetty/*****	Username with password	
Kubernetes	Kubernetes	Kubernetes configuration (kubeconfig)	
test	test	Kubernetes configuration (kubeconfig)	
minikube	minikube	Kubernetes configuration (kubeconfig)	
minikube2	minikube2	Kubernetes configuration (kubeconfig)	
minikube3	minikube3	Kubernetes configuration (kubeconfig)	

On the right side of the table is a blue button labeled '+ Add Credentials' with a red box around it. At the bottom left are icons for S, M, and L. On the right are links for 'REST API' and 'Jenkins 2.524'.

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

Jenkins Manage Jenkins Credentials System Global credentials (unrest...)

New credentials

Kind: Username with password

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

Username: msis

Treat username as secret

Password: *****

ID: Tomcat

Description: Tomcat

Create

REST API Jenkins 2.524

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**.

Jenkins / Java_Baremetal_Package... / Configuration

Configure

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

Post-build Actions

Define what happens after a build completes, like sending notifications, archiving artifacts, or triggering other jobs.

Deploy war/ear to a container

WAR/EAR file: **/*.war

Context path: MyJavaApp

Containers

Tomcat 9.x Remote

Credentials: msis/***** (tomcat_manager_script_cred_38)

Tomcat URL: http://172.18.181.38:8081

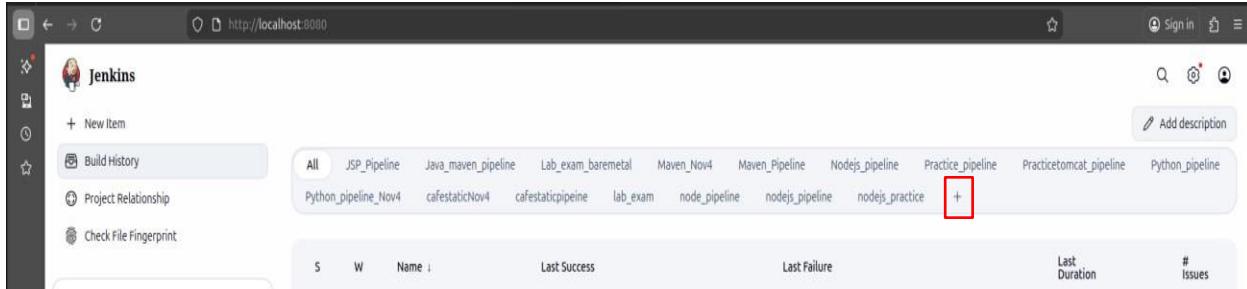
Advanced

+ Add Container

Save Apply

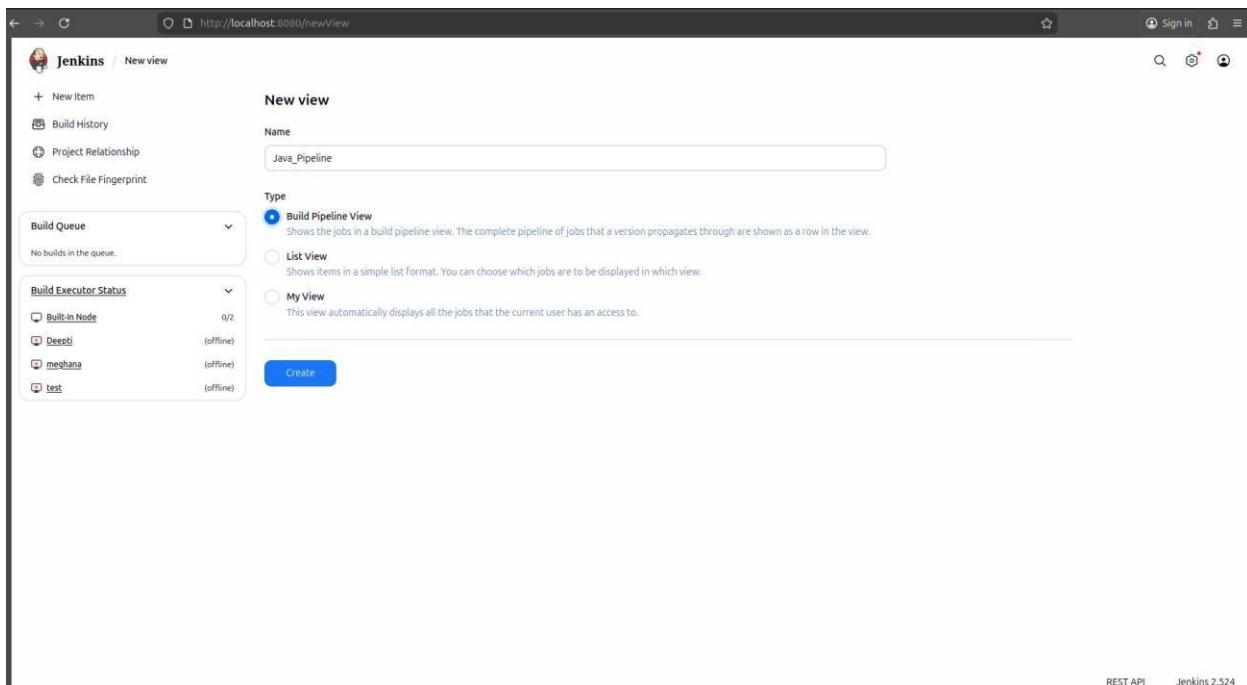
Now we will create the **Build Pipeline View**.

Navigate to the Jenkins home page and click on the “+” icon.



The screenshot shows the Jenkins home page at <http://localhost:8080>. The top navigation bar includes links for Sign in, Help, and a search bar. Below the header, there's a sidebar with options like '+ New item', 'Build History', 'Project Relationship', and 'Check File Fingerprint'. A main content area displays a list of pipelines: All, JSP_Pipeline, Java_maven_pipeline, Lab_exam_baremetal, Maven_Nov4, Maven_Pipeline, Nodejs_pipeline, Practice_pipeline, Practicetomcat_pipeline, Python_pipeline. A red box highlights the '+' icon located in the top right corner of the pipeline list area.

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



The screenshot shows the 'New view' configuration page at <http://localhost:8080/newView>. The left sidebar lists 'New item', 'Build History', 'Project Relationship', and 'Check File Fingerprint'. The main form is titled 'New view' and contains fields for 'Name' (set to 'Java_Pipeline') and 'Type' (radio button selected for 'Build Pipeline View'). Other options shown are 'List View' and 'My View'. At the bottom is a 'Create' button. On the right, there are sections for 'Build Queue' (empty) and 'Build Executor Status' (listing nodes: Built-in Node (0/2), Deepali (offline), meghana (offline), test (offline)). The footer indicates REST API and Jenkins 2.524.

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**.

http://localhost:8080/view/Java_Pipeline/configure

Jenkins Java_Pipeline / Edit View

Layout

Based on upstream/downstream relationship

This layout mode derives the pipeline structure based on the upstream/downstream trigger relationship between jobs. This is the only out-of-the-box supported layout mode, but is open for extension.

Upstream / downstream config

Select Initial Job

Trigger Options

Build Cards

Standard build card

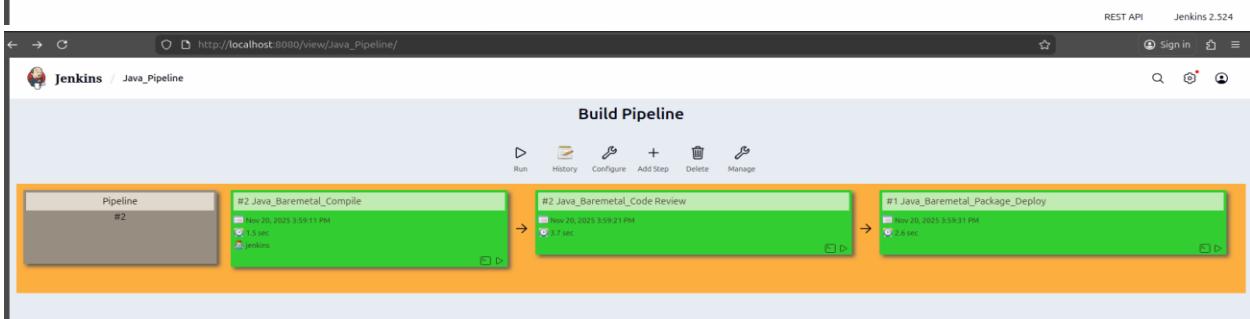
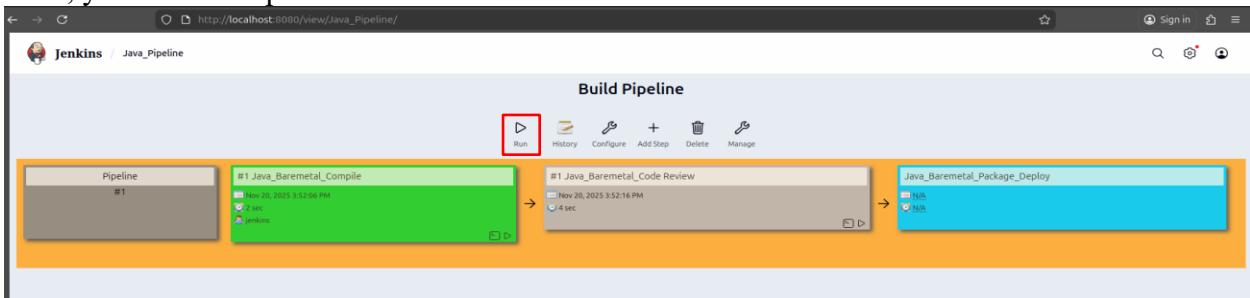
Use the default build cards

Restrict triggers to most recent successful builds Yes No

Always allow manual trigger on pipeline steps Yes No

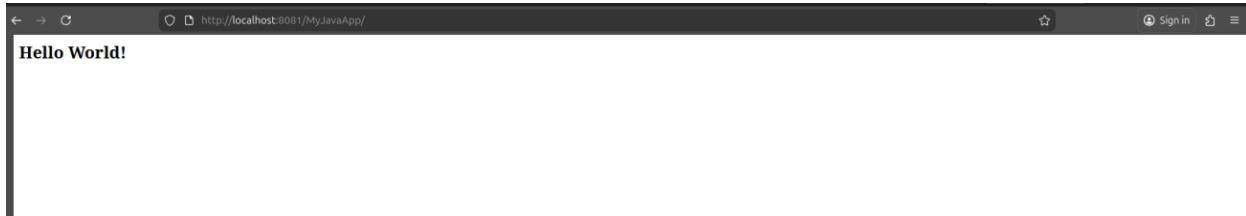
Display Options

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**.

New Item

Enter an item name

Java_Docker

Select an item type

- Freestyle project**
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.
- Maven project**
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.
- Pipeline**
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
- Multi-configuration project**
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.
- Folder**
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.
- Multibranch Pipeline**
Creates a set of Pipeline projects according to detected branches in one SCM repository.
- Organization Folder**
Creates a set of multibranch project subfolders by scanning for repositories.

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.

Configure

Source Code Management

None

Git

Repositories

Repository URL

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

Credentials

- none -

+ Add

Advanced

+ Add Repository

Branches to build

Branch Specifier (blank for 'any')

/master

+ Add Branch

Repository browser

Save

Apply

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**.

New credentials

Kind: Username with password

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

Username: sheetalshetty

Treat username as secret:

Password: (redacted)

ID: docker

Description: This is the Dockerhub credentials

Create

REST API Jenkins 2.524

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.

Configure

With Ant:

General

Source Code Management

Triggers

Environment

Build Steps

Post-build Actions

Build Steps

Docker Build and Publish

Repository Name: sheetalshetty/java

Tag: v3

Docker Host URI:

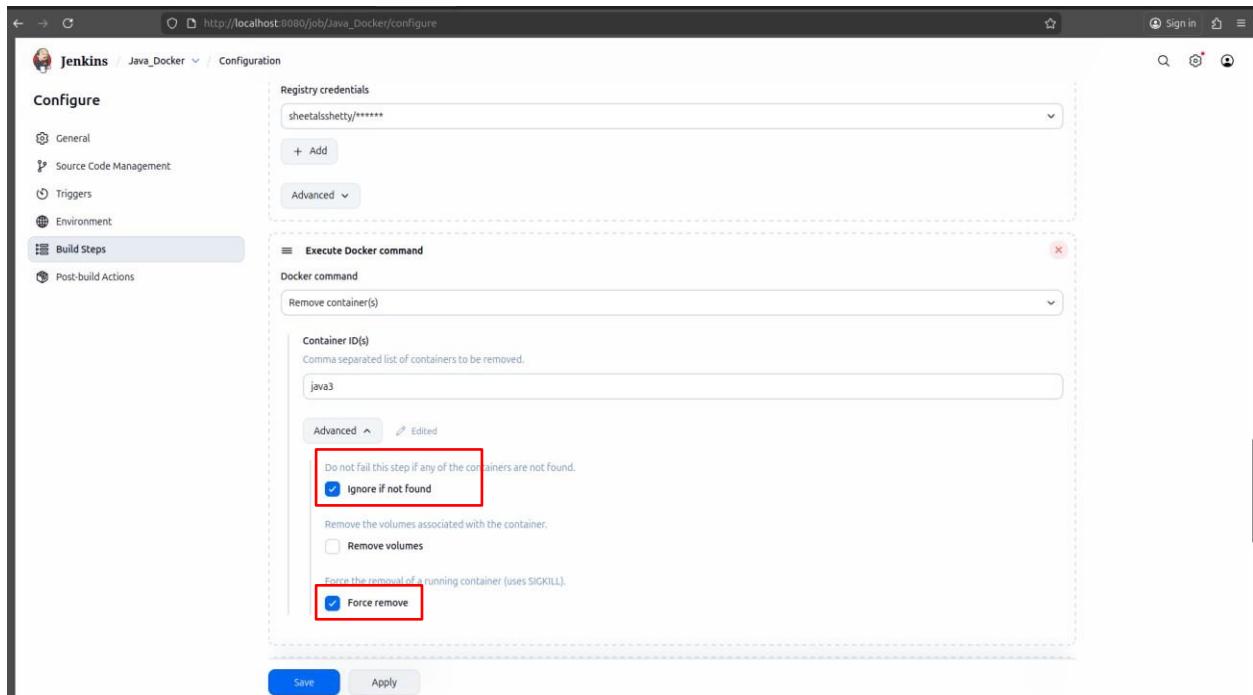
Server credentials: -none-

Docker registry URL:

Registry credentials: sheetalshetty/*****

Save Apply

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).

The screenshot shows the Jenkins job configuration page for a job named "Java_Docker". The "Post-build Actions" section is selected. It contains two "Execute Docker command" actions. The first action has the "Create container" option selected, with "Image name" set to "sheetalsshetty/java:v3", "Command" field empty, and "Hostname" field empty. The second action also has the "Create container" option selected. At the bottom, there are "Save" and "Apply" buttons.

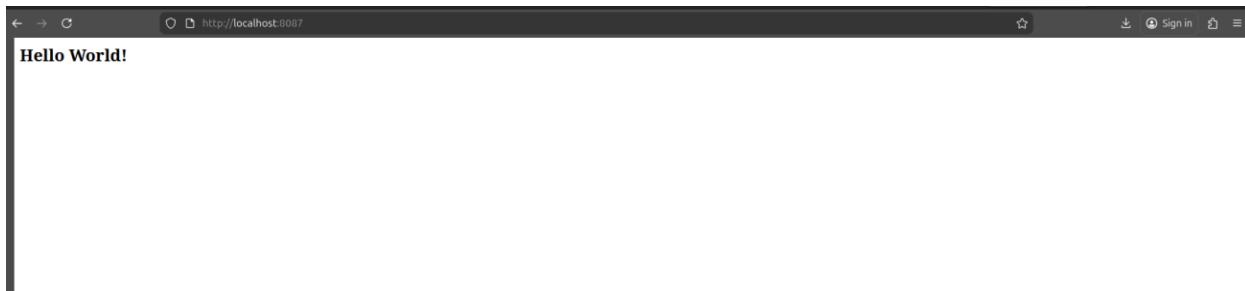
The screenshot shows the Jenkins job configuration page for a job named "Java_Docker". The "Post-build Actions" section is selected. It contains a "Docker command" action. Under "Port bindings", the value "8087:8080" is listed. There are "Check syntax" buttons next to both the port bindings and container ID fields. At the bottom, there are "Save" and "Apply" buttons.

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**.

The screenshot shows the Jenkins configuration page for a job named "Java_Docker". The "Post-build Actions" section is selected. It contains a single step: "Execute Docker command" with the command "Start container(s)" and the container ID "java". There are "Save" and "Apply" buttons at the bottom.

The screenshot shows the Jenkins summary page for the "Java_Docker" job. The "Build Now" button is highlighted with a red box. The "Builds" table shows one recent build: "#12 sheetalsshetty/java:v3 sheetalsshetty/java:latest" from today at 2:08 PM. The "Permalinks" section lists four builds: Last build (#12), Last stable build (#12), Last successful build (#12), and Last completed build (#12). There are "Save", "Apply", "REST API", and "Jenkins 2.524" buttons at the bottom.

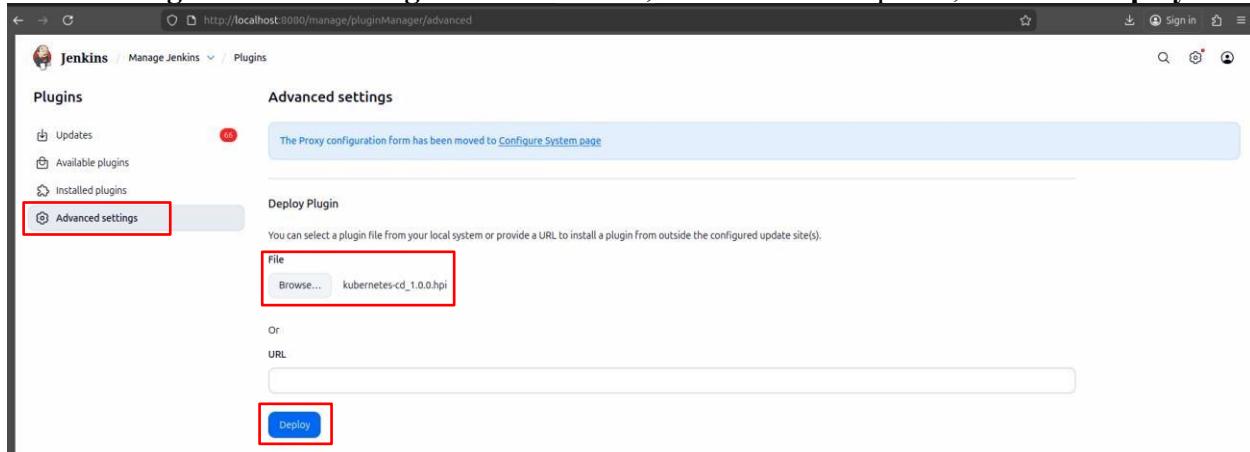
we can access the web page at [http://localhost:<Host Port>/](http://localhost:<Host Port>)



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**.

The screenshot shows the Jenkins 'New credentials' page. The 'Kind' dropdown is set to 'Kubernetes configuration (kubeconfig)'. The 'Scope' dropdown is set to 'Global (Jenkins, nodes, items, all child items, etc.)'. The 'ID' field contains 'kubernetes'. The 'Content' field displays a long base64-encoded kubeconfig string. Below the content, there are three radio button options: 'From a file on the Jenkins master', 'From a file on the Kubernetes master node', and 'Enter directly' (which is selected). A 'Create' button is at the bottom.

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

The screenshot shows the Jenkins 'New Item' page. In the 'Enter an item name' field, 'Java_Kubernetes' is typed. Under 'Select an item type', the 'Freestyle project' option is selected, shown with its icon and description: 'Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.' Other options like 'Maven project', 'Pipeline', 'Multi-configuration project', 'Folder', 'Multibranch Pipeline', and 'Organization Folder' are also listed. A 'OK' button is at the bottom.

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.

The screenshot shows the Jenkins job configuration page for a job named "Java_Kubernetes". The "Source Code Management" section is active, showing a Git repository configuration. The "Repository URL" is set to `https://github.com/sheetal-Shetty/java-tomcat-maven-example.git`. The "Branch Specifier" is set to `*/master`. Below these fields are "Save" and "Apply" buttons.

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**.

The screenshot shows the Jenkins job configuration page for a job named "Java_Kubernetes". The "Build Steps" section is active, showing a "Deploy to Kubernetes" step. The "Kubeconfig" dropdown is set to "Kubernetes". The "Config Files" dropdown contains `deployment.yaml, service.yaml`. The "Enable Variable Substitution in Config" checkbox is checked. Below these fields are "Verify Configuration", "Save", and "Apply" buttons.

Jenkins / Java_Kubernetes

Status: **Java_Kubernetes**

Build Now

Permalinks

- Last build (#8), 3 hr 56 min ago
- Last stable build (#8), 3 hr 56 min ago
- Last successful build (#8), 3 hr 56 min ago
- Last failed build (#4), 4 hr 18 min ago
- Last unsuccessful build (#4), 4 hr 18 min ago
- Last completed build (#8), 3 hr 56 min ago

Builds

Filter
#8 2:15 PM
#7 2:13 PM
#6 1:55 PM
#5 1:54 PM

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,

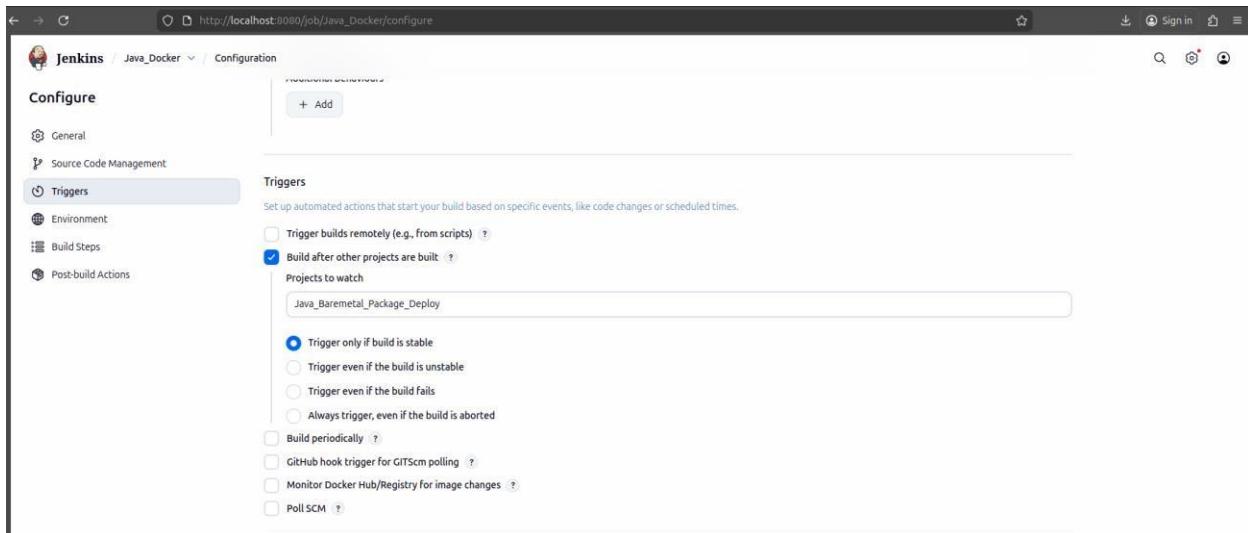
Hello World!

[http://localhost:<NodePort> or http://<cluster ip>:<Target Port>](http://localhost:32034/)

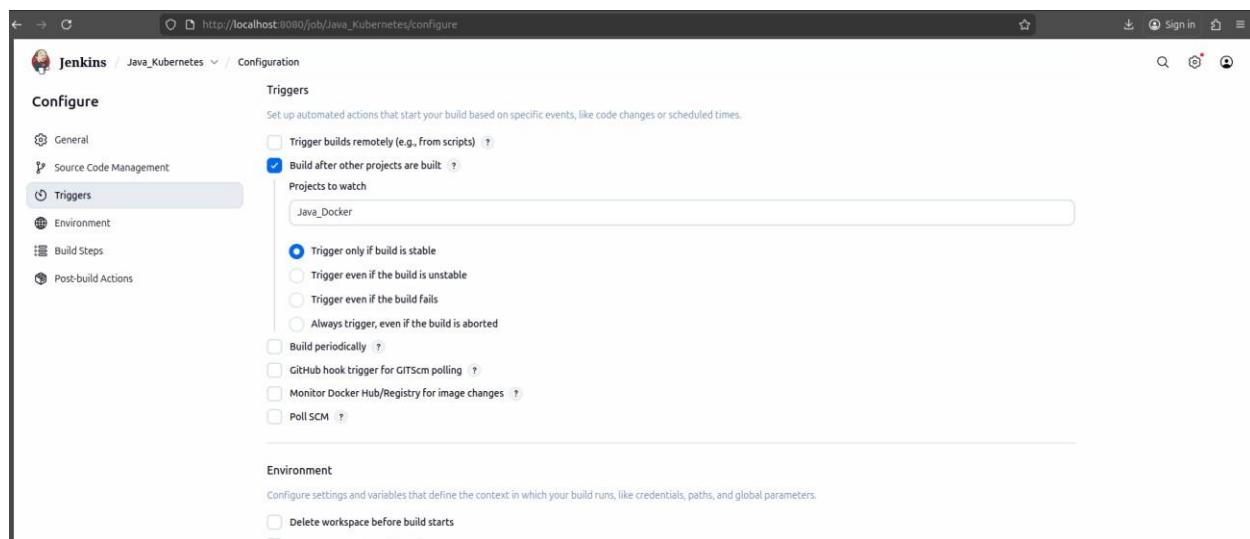
<http://localhost:32034/>

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

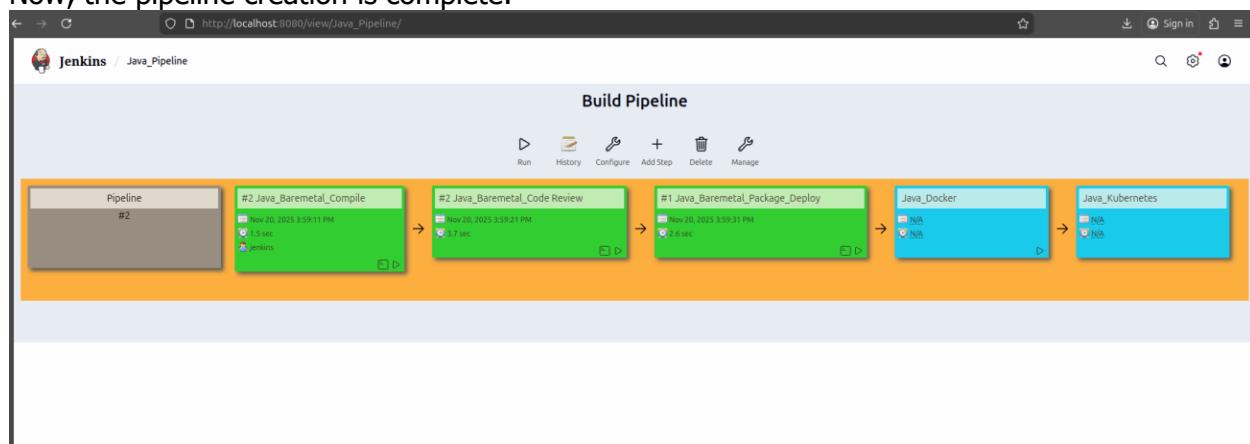
Since we have already created the pipeline for Java_Baremetal_Compile → Java_Baremetal_CodeReview → Java_Baremetal_Package_Deploy, we now need to add the remaining two stages (Java_Docker and Java_Kubernetes). Go to the item named Java_Docker. Under Triggers, select “Build after other projects are built.”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



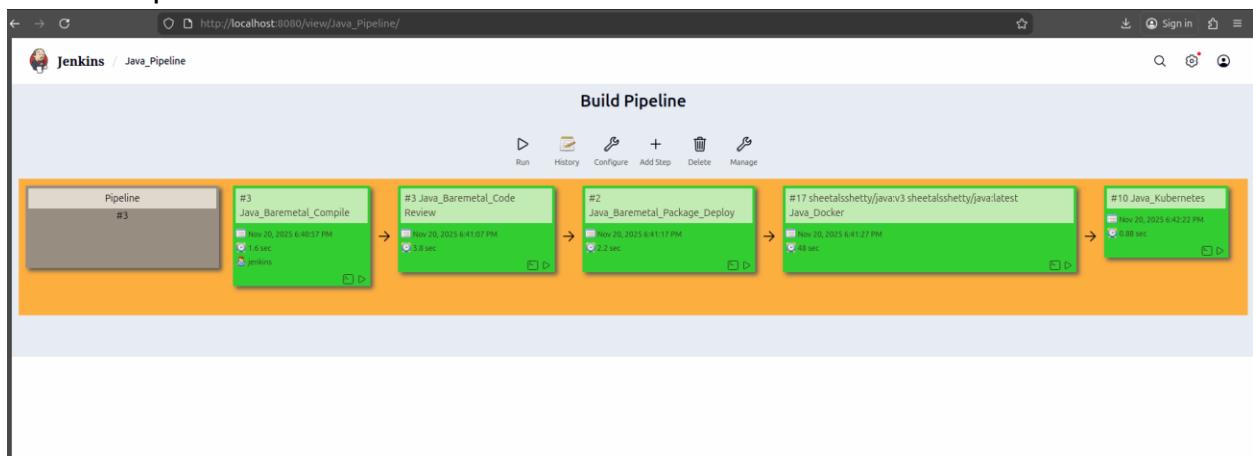
Go to the item named Java_Baremetal_Package_Deploy. Under Triggers, select "Build after other projects are built." Then, under Projects to watch, select Java_Docker.



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.

Jenkins / All / New Item

New Item

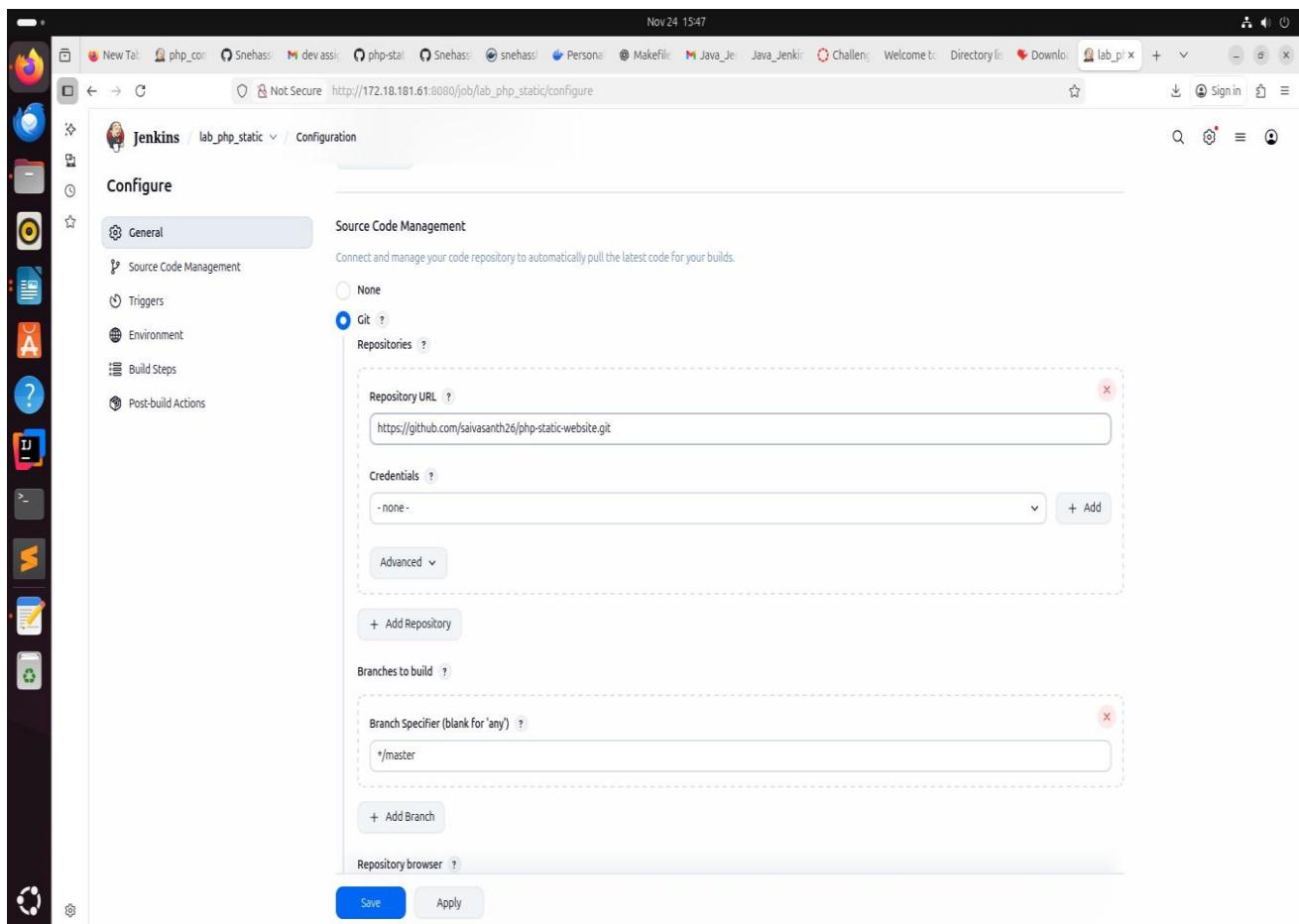
Enter an item name

Select an item type

- Freestyle project**
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.
- Maven project**
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.
- Pipeline**
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
- Multi-configuration project**
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.
- Folder**
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.
- Multibranch Pipeline**
Creates a set of Pipeline projects according to detected branches in one SCM repository.
- Organization Folder**
Creates a set of multibranch project subfolders by scanning for repositories.

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

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
    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**

Jenkins / lab_php_static / Configuration

Configure

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

Execute shell

Command

See the list of available environment variables

```
# 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**

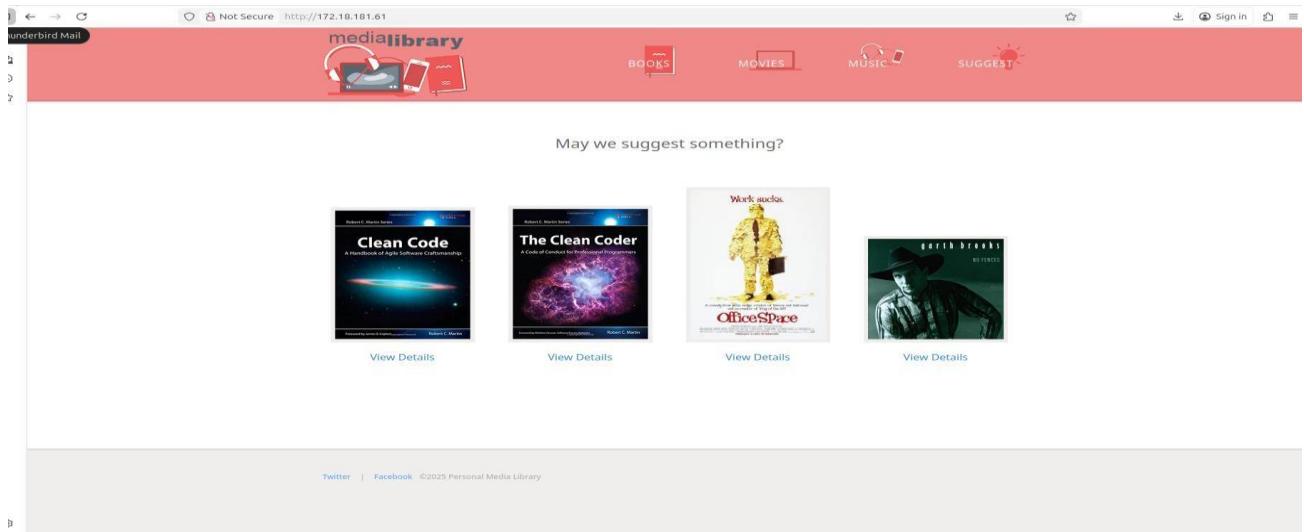
Under **Configure**. Click on **build now**

Builds

#	Time
#6	3:51 PM
#5	3:51 PM
#4	3:18 PM
#3	3:15 PM
#2	12:52 PM
#1	12:50 PM

Build scheduled
http://172.18.181.61:8080/job/lab_php_static/ws/

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:\\\\archive.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
```

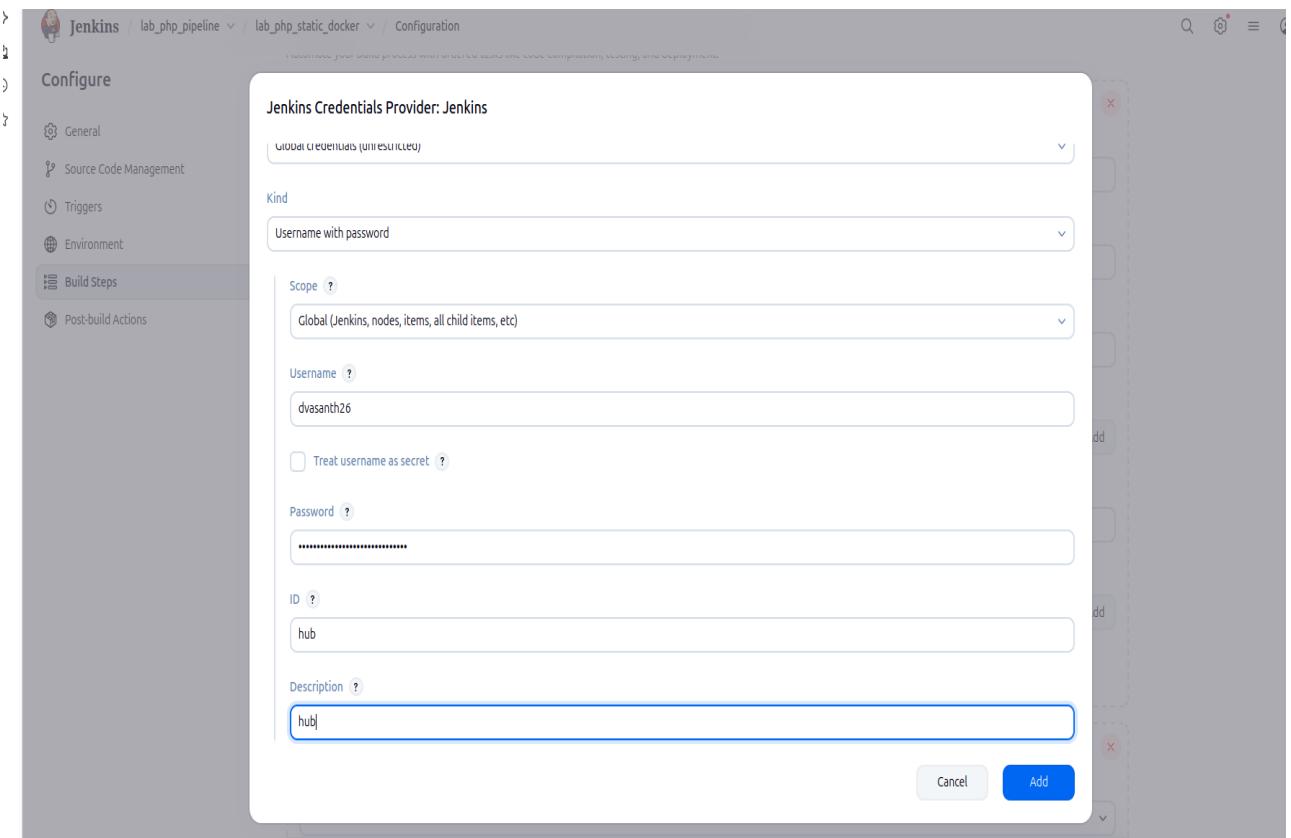
Create a **Freestyle Jenkins Project** and provide a name for the project, for example

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.

The screenshot shows the Jenkins configuration page for a job named 'lab_php_static_docker'. The 'General' tab is selected under 'Configure'. In the 'Source Code Management' section, 'Git' is chosen as the provider, and the repository URL is set to `https://github.com/saivasanth26/php-static-website.git`. The 'Branches to build' field contains the specifier `*/master`. At the bottom, there are 'Save' and 'Apply' buttons.

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.

The screenshot shows the Jenkins job configuration page for 'lab_php_static_docker'. The left sidebar lists configuration sections: General, Source Code Management, Triggers, Environment, Build Steps (selected), and Post-build Actions. The main area displays two build steps:

- Docker Build and Publish**:
 - Repository Name: dvasanth26/phpstatic
 - Tag: v1
 - Docker Host URI: (empty)
 - Server credentials: -none-
 - Docker registry URL: (empty)
 - Registry credentials: dvasanth26/******** (hub)
- Execute Docker command**:
 - Docker command: Create container

At the bottom are 'Save' and 'Apply' buttons.

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).

The screenshot shows the Jenkins job configuration page for 'lab_php_static_docker'. The left sidebar lists configuration sections: General, Source Code Management, Triggers, Environment, Build Steps, and Post-build Actions. 'Post-build Actions' is currently selected. The main panel displays the Docker configuration for the container. It includes fields for Network Mode (set to 'bridge'), Publish all ports (checked), Port bindings (125:80), Bind mounts (empty), Extended privileges (unchecked), and always restart (unchecked). A 'Check syntax' button is present in each of the syntax-validation boxes.

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**.

The screenshot shows the Jenkins job configuration page for 'lab_php_static_docker'. The left sidebar lists configuration sections: General, Source Code Management, Triggers, Environment, Build Steps, and Post-build Actions. 'Build Steps' is currently selected. A 'Docker command' step is added, with the 'Start container(s)' option selected and the 'Container ID(s)' field containing 'phpbook_'. Below the build steps, the 'Post-build Actions' section is visible, which allows for defining what happens after a build completes. At the bottom, there are 'Save' and 'Apply' buttons.

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

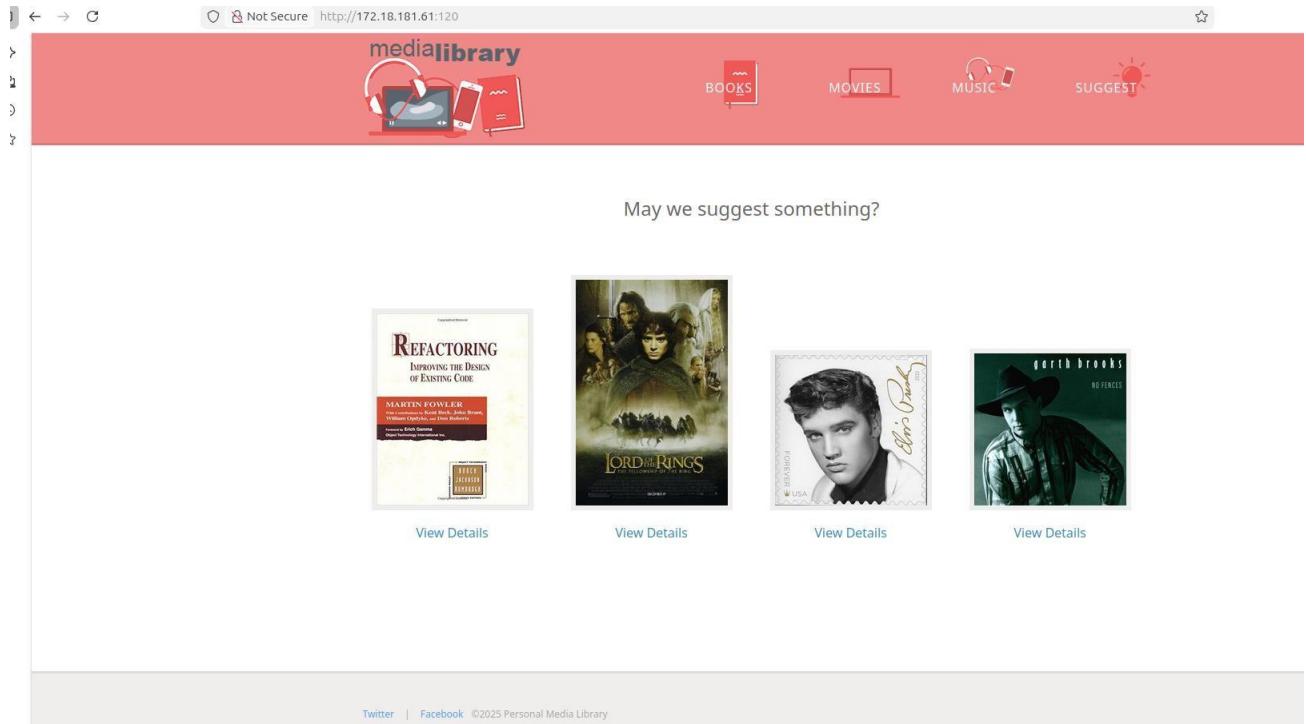
The screenshot shows the Jenkins job configuration page for 'lab_php_static_docker'. The left sidebar contains links for Status, Changes, Workspace, Build Now (highlighted with a yellow background), Configure, Delete Project, Rename, and Credentials. The main content area shows the job name 'lab_php_static_docker' with a green checkmark icon. Below it are sections for Upstream Projects ('lab_php_static') and Downstream Projects ('lab_php_static_kube'). A Permalinks section lists recent builds:

- Last build (#7 dvasanth26/phpstatic:v1 dvasanth26/phpstatic:latest), 38 sec ago
- Last stable build (#7 dvasanth26/phpstatic:v1 dvasanth26/phpstatic:latest), 38 sec ago
- Last successful build (#7 dvasanth26/phpstatic:v1 dvasanth26/phpstatic:latest), 38 sec ago
- Last unsuccessful build (#4 dvasanth26/phpstatic:v1 dvasanth26/phpstatic:latest), 4 days 0 hr ago
- Last completed build (#7 dvasanth26/phpstatic:v1 dvasanth26/phpstatic:latest), 38 sec ago

The Builds section shows a list of builds grouped by date:

- Today:
 - #7 dvasanth26/phpstatic:v1 dvasanth26/phpstatic:latest (3:56 PM)
 - #5 dvasanth26/phpstatic:v1 dvasanth26/phpstatic:latest (3:51 PM)
- November 20, 2025:
 - #4 dvasanth26/phpstatic:v1 dvasanth26/phpstatic:latest (3:18 PM)
 - #3 dvasanth26/phpstatic:v1 dvasanth26/phpstatic:latest (3:15 PM)
 - #2 dvasanth26/phpstatic:v1 dvasanth26/phpstatic:latest (3:03 PM)
 - #1 dvasanth26/phpstatic:v1 dvasanth26/phpstatic:latest (1:28 PM)

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



3. Deploy containerized application on Kubernetes cluster environment – Java

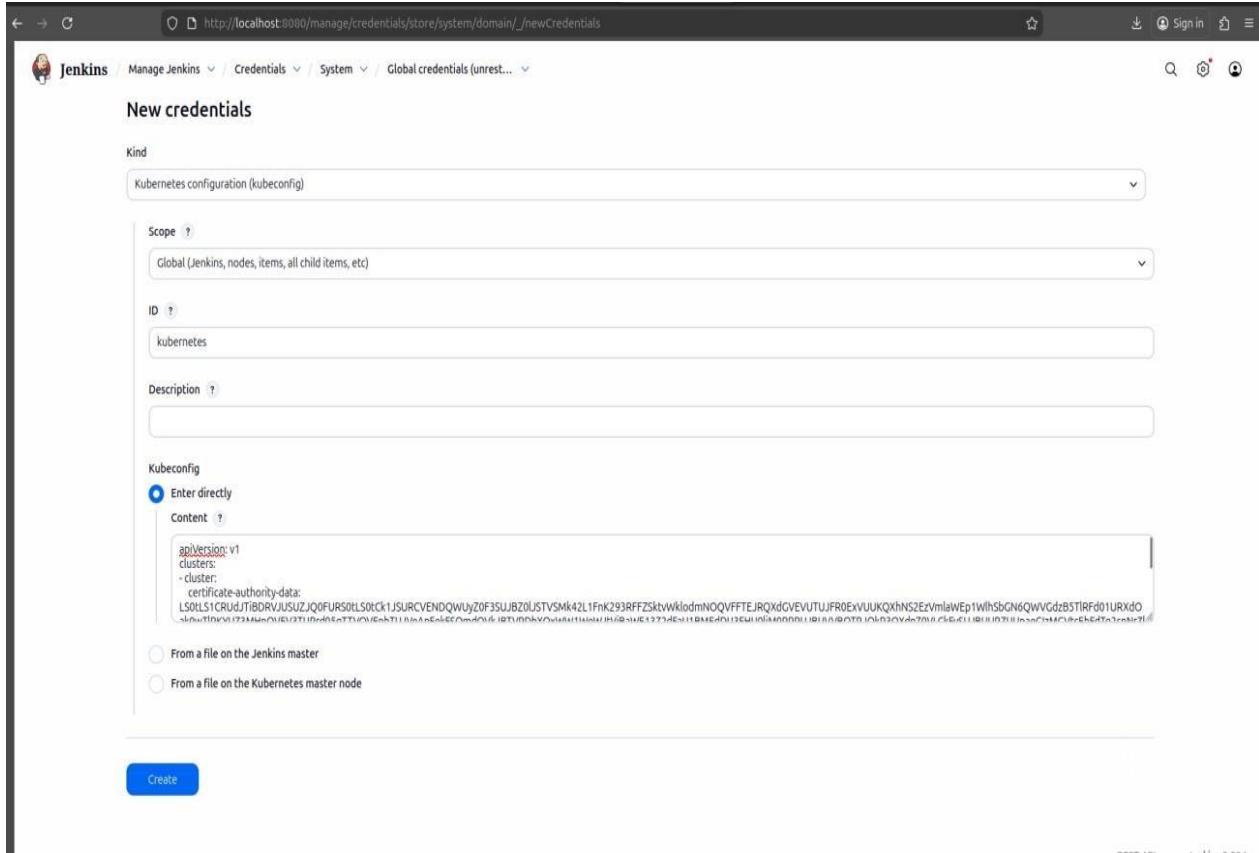
Plugins: Install the Kubernetes plugin using sir's kubernetes-cd_1.0.0.hpi file (available in the GitHub repository linked below).

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.

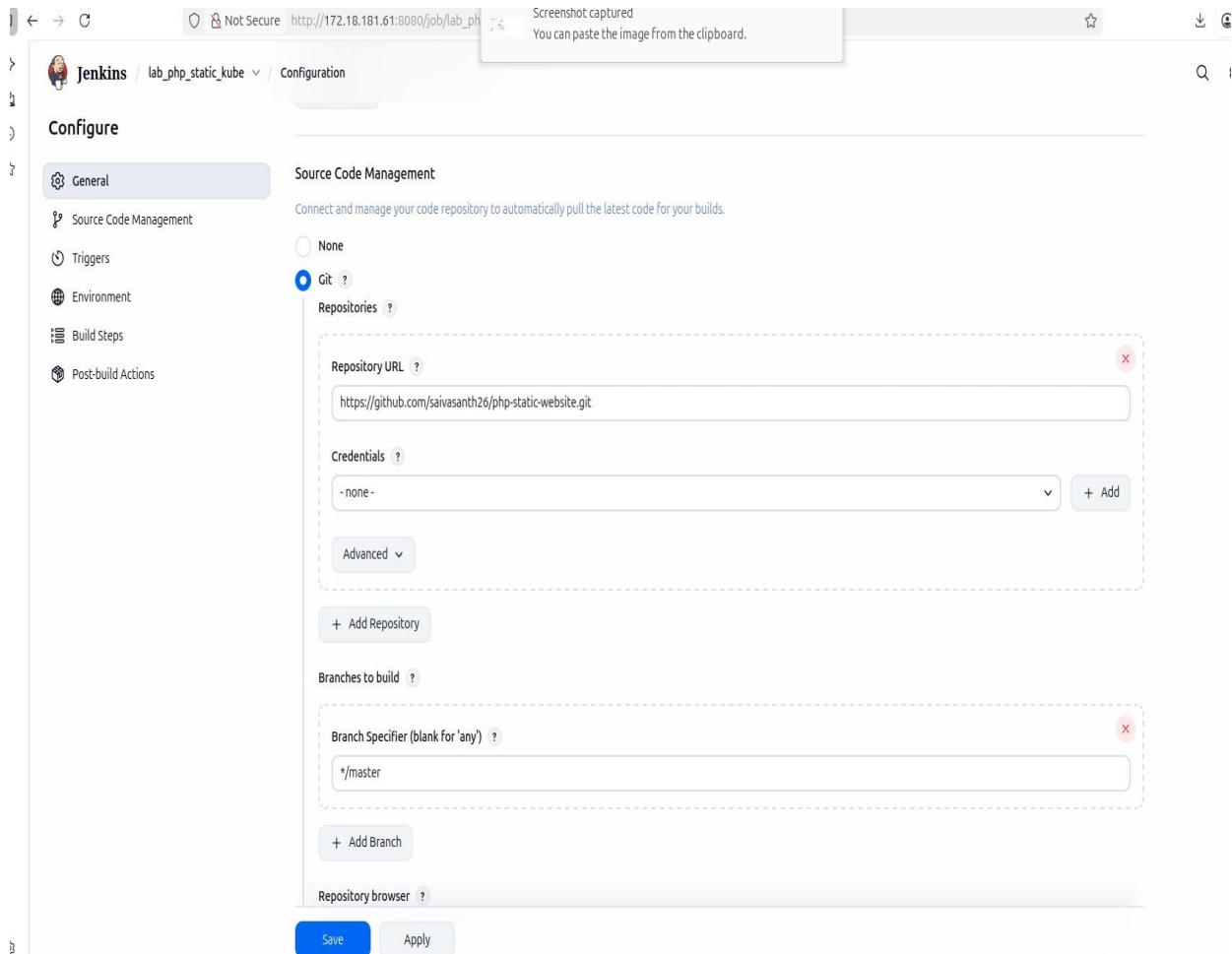
GitHub Repo: <https://github.com/saivasanth26/php-static-website.git>

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**.

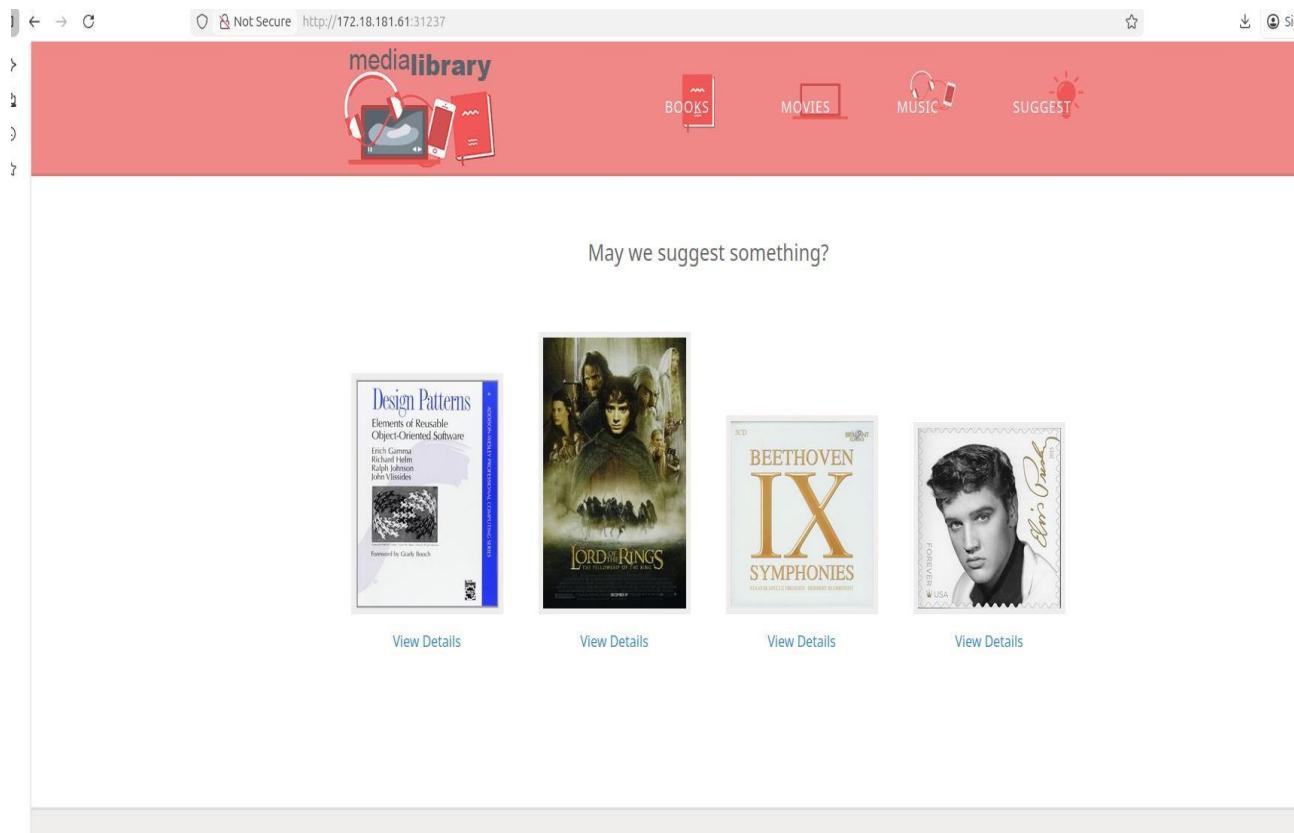
The screenshot shows the Jenkins configuration page for the job 'lab_php_static_kube'. The left sidebar lists configuration sections: General, Source Code Management, Triggers, Environment, Build Steps (which is selected and highlighted in grey), and Post-build Actions. The main content area is titled 'Configure' and contains a 'Deploy to Kubernetes' step. This step includes fields for 'Kubeconfig' (set to 'kubeconfig'), 'Config Files' (set to 'deployment.yaml,service.yaml'), and a checked checkbox for 'Enable Variable Substitution in Config'. A 'Verify Configuration' button is located at the bottom right of this step's panel. Below the configuration panels, there is a 'Post-build Actions' section with a '+ Add post-build action' button. At the bottom of the page are 'Save' and 'Apply' buttons.

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

The screenshot shows the Jenkins status page for the job 'lab_php_static_kube'. The left sidebar has links for Status (which is selected and highlighted in grey), Changes, Workspace, Build Now, Configure, Delete Project, Rename, and Credentials. The main content area displays the job name 'lab_php_static_kube' with a green checkmark icon. It shows 'Upstream Projects' with 'lab_php_static_docker' listed. Under 'Permalinks', it lists recent builds: Last build (#4), Last stable build (#4), Last successful build (#4), Last failed build (#1), and Last completed build (#4). The 'Builds' section shows a table of builds from today, with rows for '#4 3:56 PM' and '#3 3:51 PM'. A filter input field is at the top of the build table.

<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.

Jenkins / lab_php_static_kube / Configuration

Configure

Triggers (selected)

- Trigger builds remotely (e.g., from scripts) ?
- Build after other projects are built ?
 - Projects to watch:
lab_php_static_docker,
- Trigger only if build is stable
- Trigger even if the build is unstable
- Trigger even if the build fails
- Always trigger, even if the build is aborted

Build periodically ?

GitHub hook trigger for GITScm polling ?

Poll SCM ?

Environment

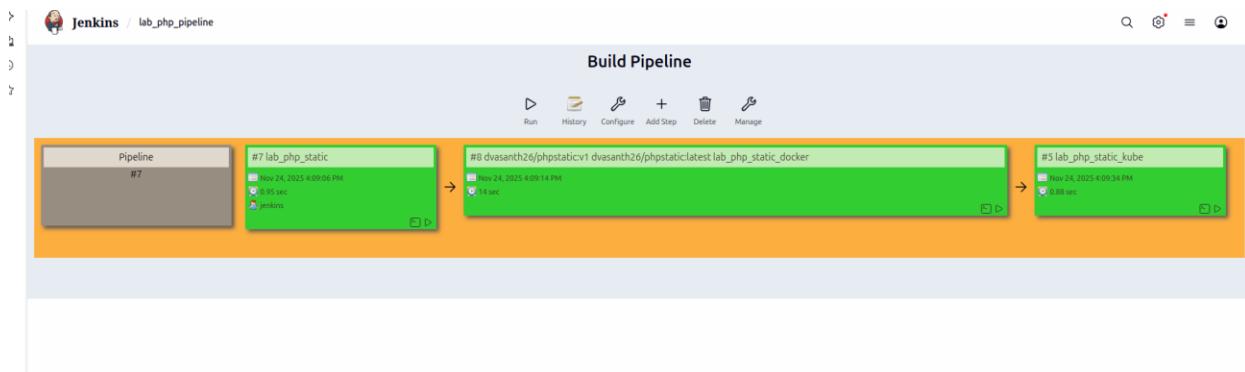
Configure settings and variables that define the context in which your build runs, like credentials, paths, and global parameters.

- Delete workspace before build starts
- Use secret text(s) or file(s) ?
- Add timestamps to the Console Output
- Build inside a Docker container ?
- Configure Kubernetes CLI (kubectl) (deprecated, use the multi credentials one instead) ?
- Configure Kubernetes CLI (kubectl) with multiple credentials
- Inspect build log for published build scans
- Setup Kubernetes CLI (kubectl) ?
- Terminate a build if it's stuck

Buttons: Save | Apply

Now, the pipeline creation is complete.

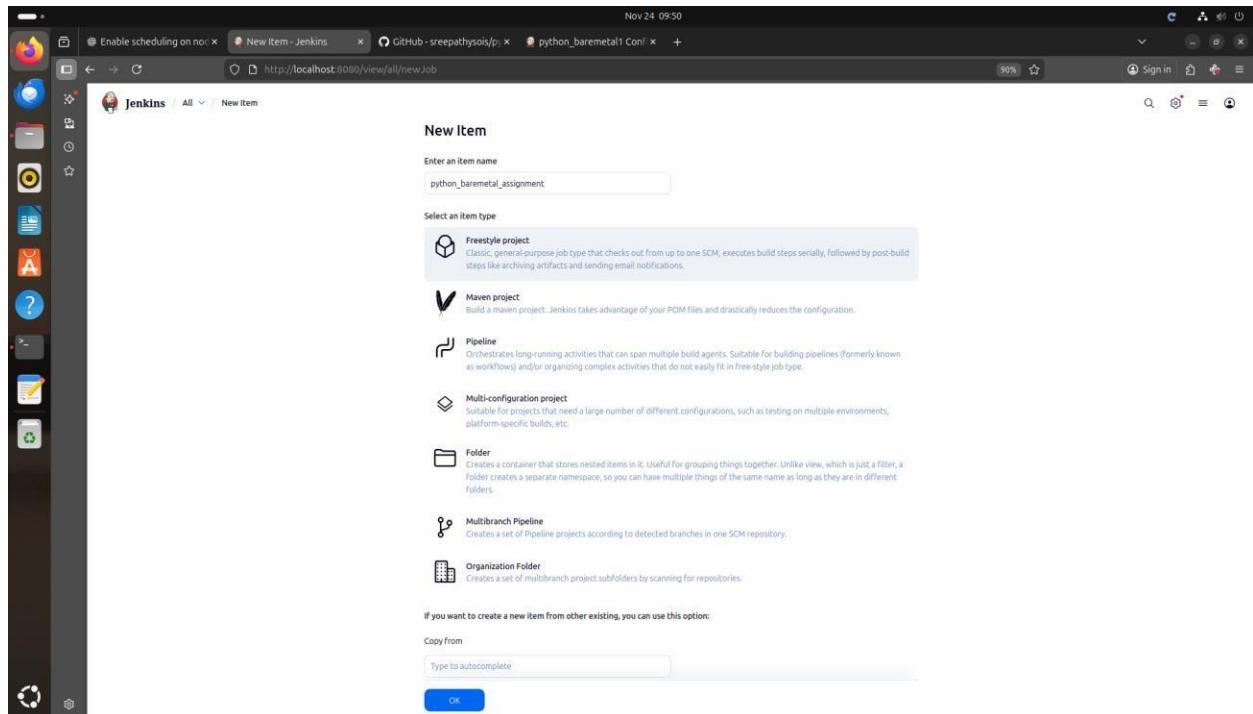
Run the Pipeline.



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



The screenshot shows the Jenkins configuration page for the 'python_baremetal_assignment' job. The 'Source Code Management' section is active, showing a 'Git' repository configuration. The 'Repository URL' is set to `https://github.com/sreepathyso/sreepathyso/python-demoapp-monitoring.git`. The 'Branch Specifier (blank for 'any')' is set to `*/*`. There are 'Save' and 'Apply' buttons at the bottom.

The screenshot shows the Jenkins configuration page for the 'python_baremetal_assignment' job. The 'Build Steps' section is active, containing a single 'Execute shell' step with the command:

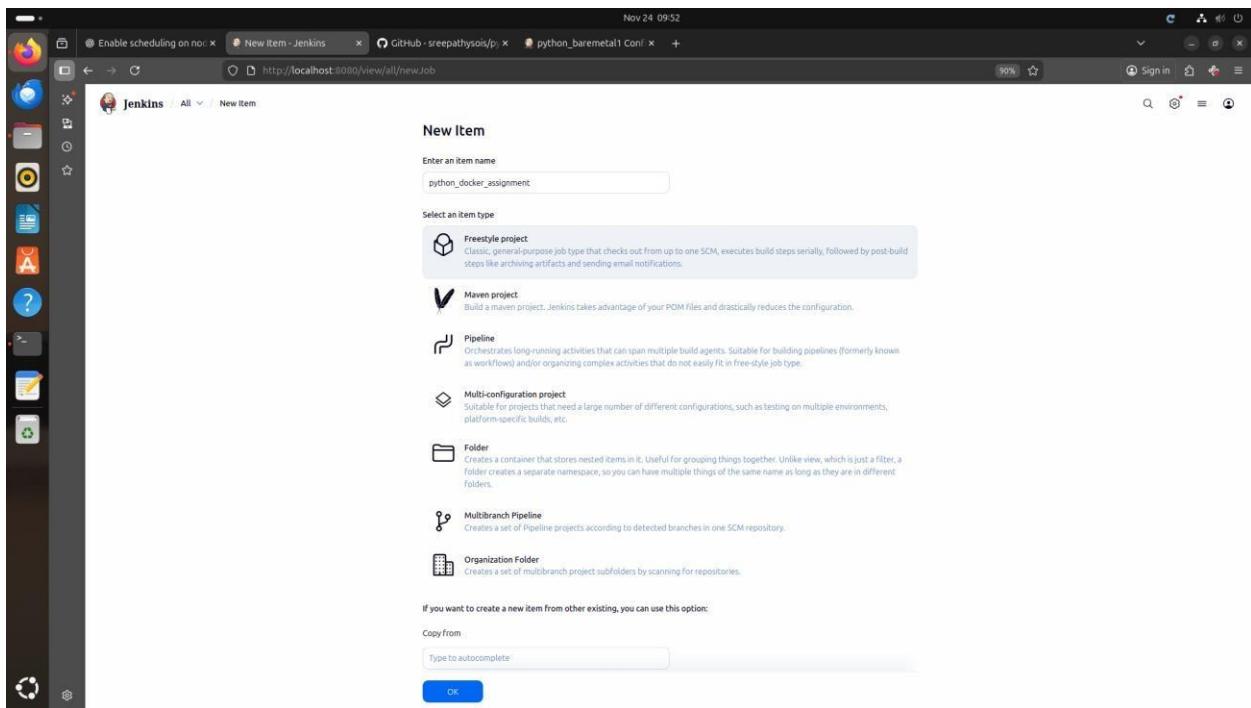
```
sudo make lint  
sudo make lint-fix  
sudo make test  
sudo make test-report
```

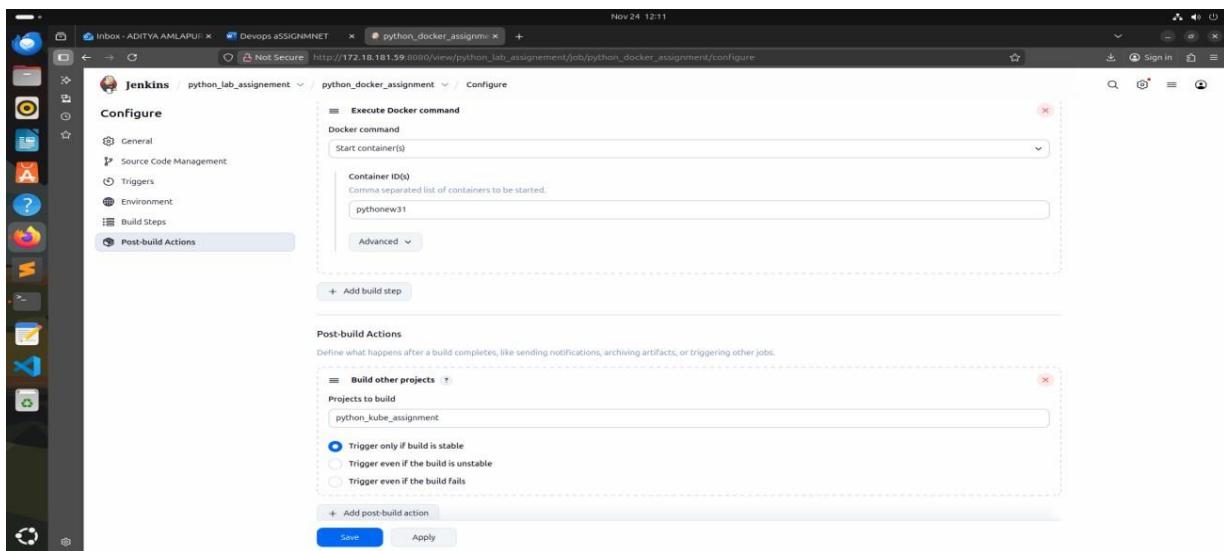
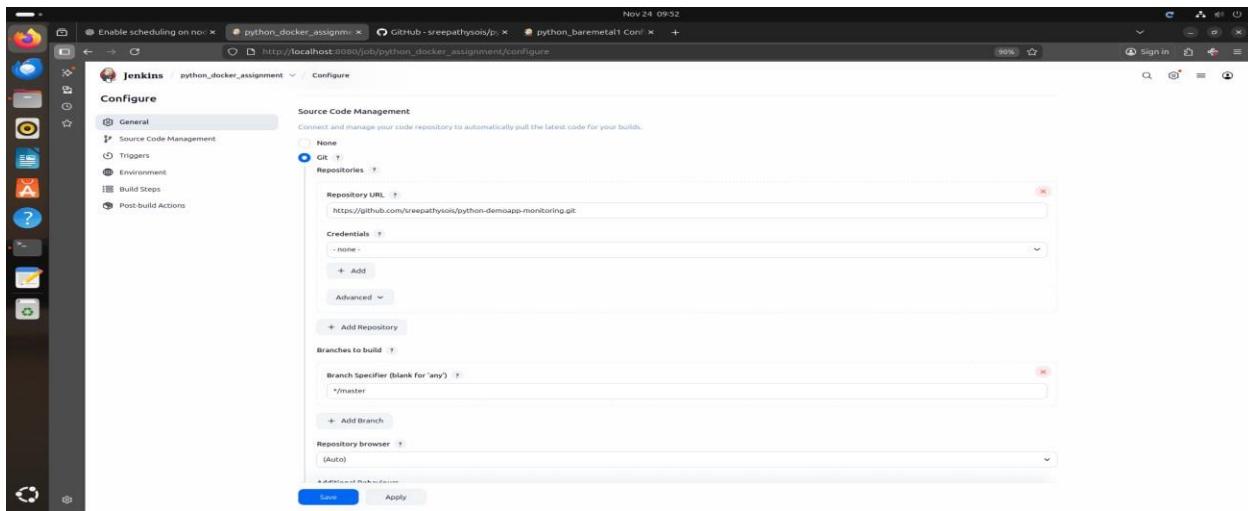
There are 'Save' and 'Apply' buttons at the bottom.

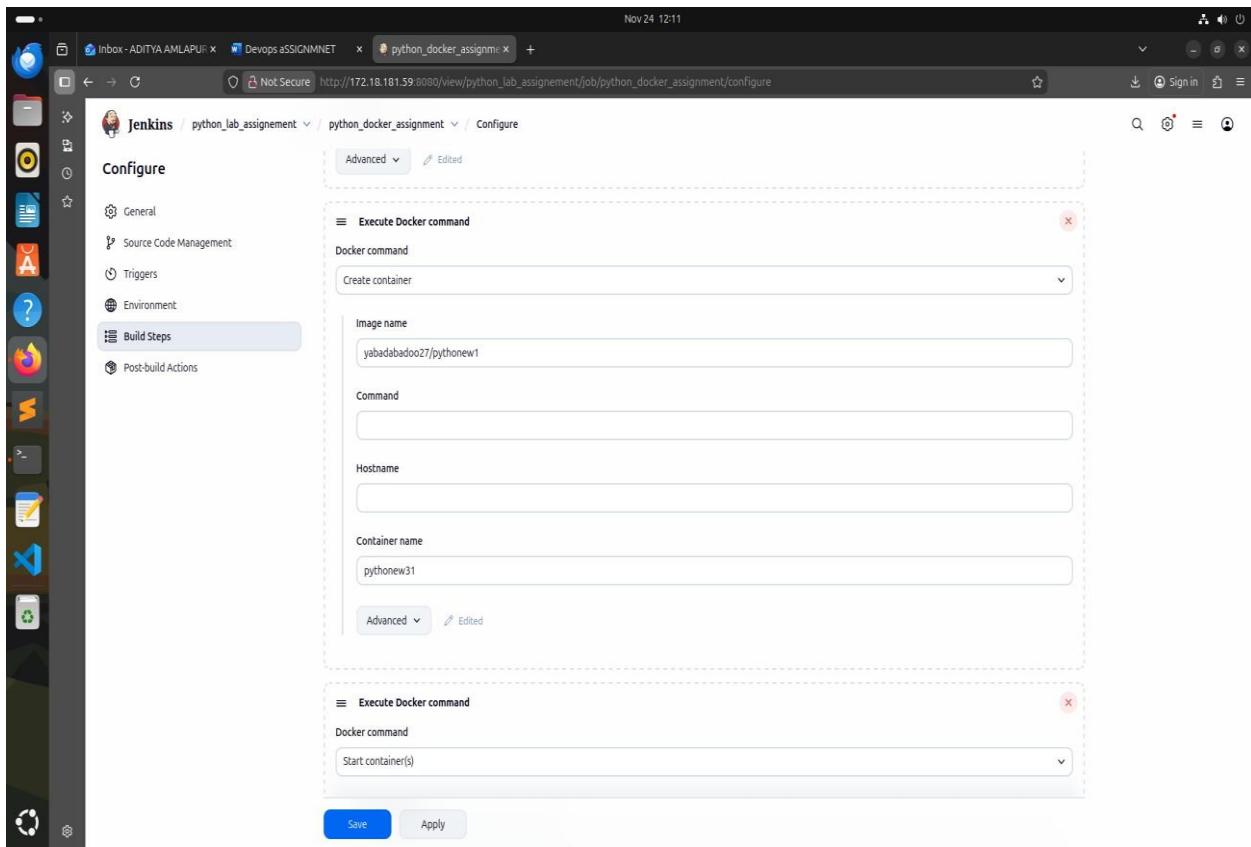
Step 2

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





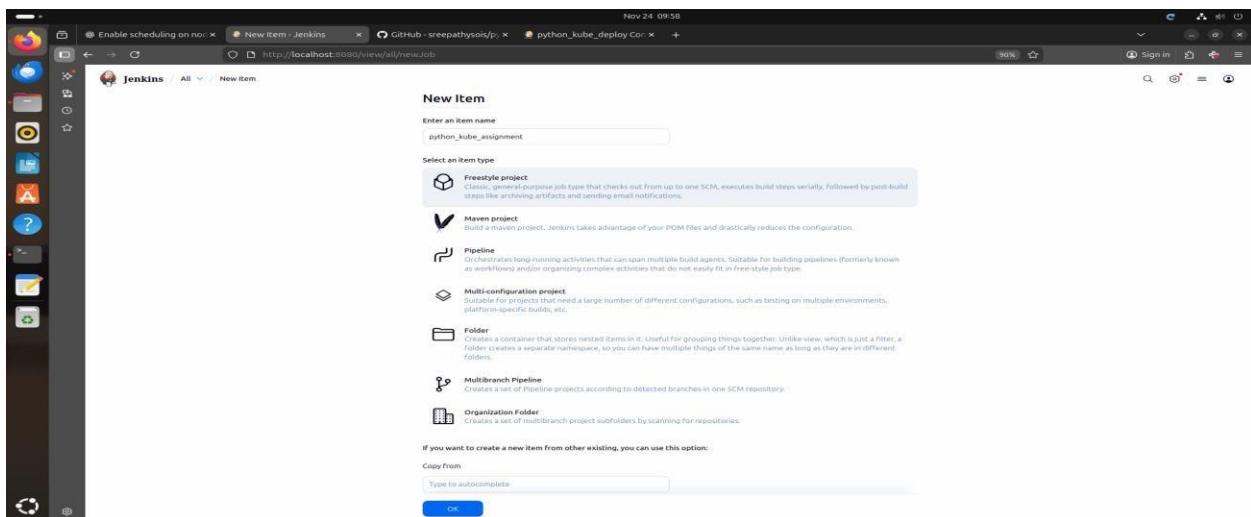


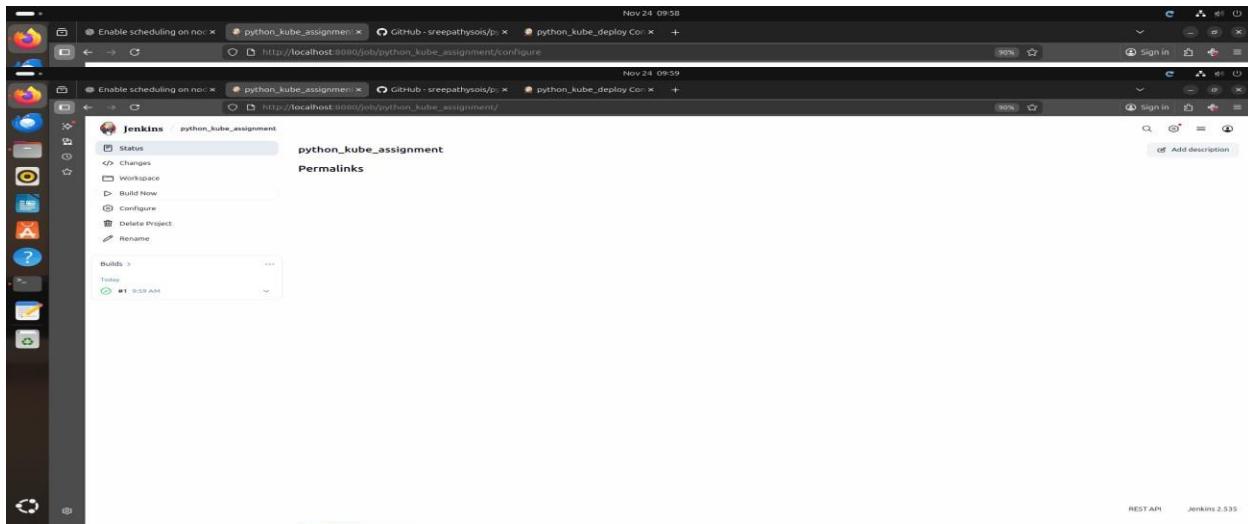
Step 3

Create one more freestyle project (kubernetes)

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

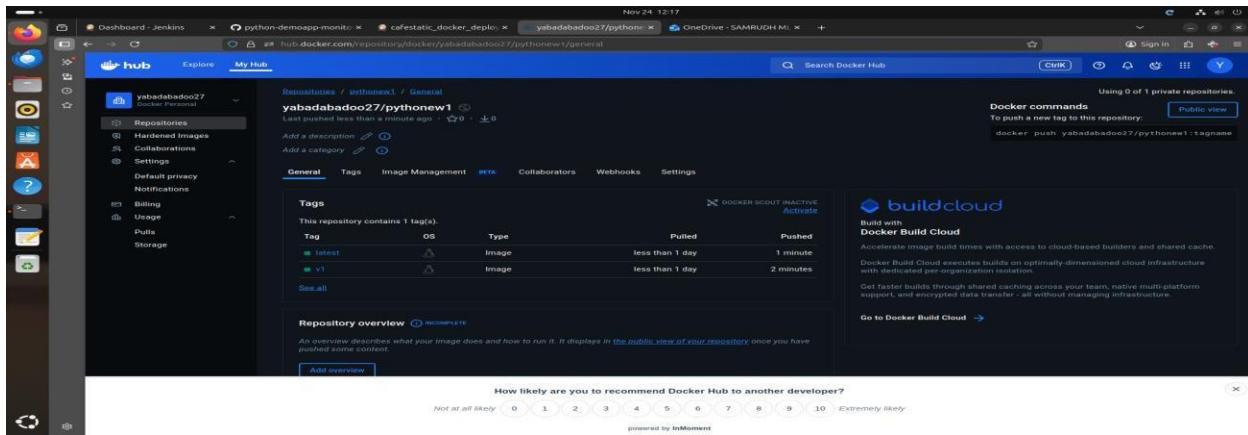
- Git
- Build steps – deploy to k8s(kubeconfig , files)



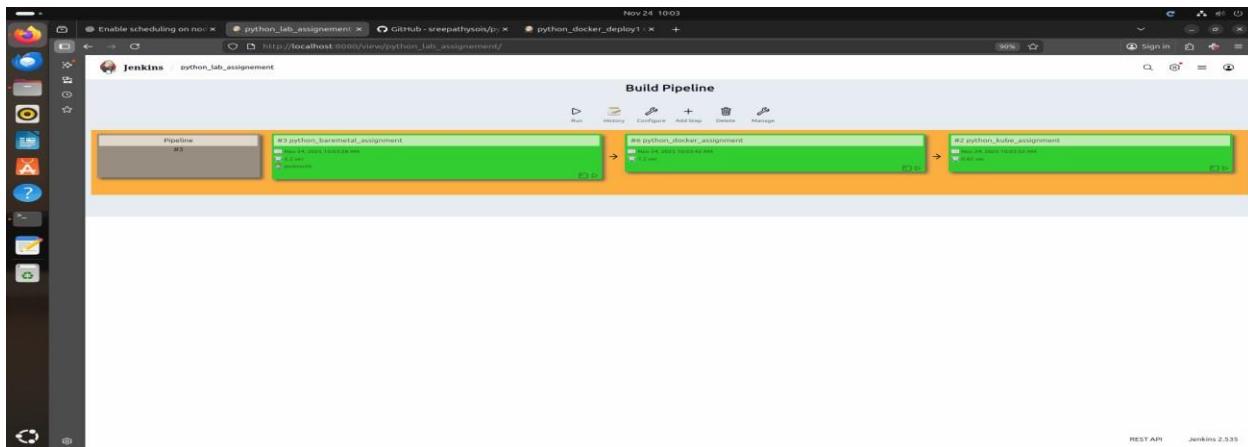


Results

1. Docker Hub – images pushed to doscker hub.



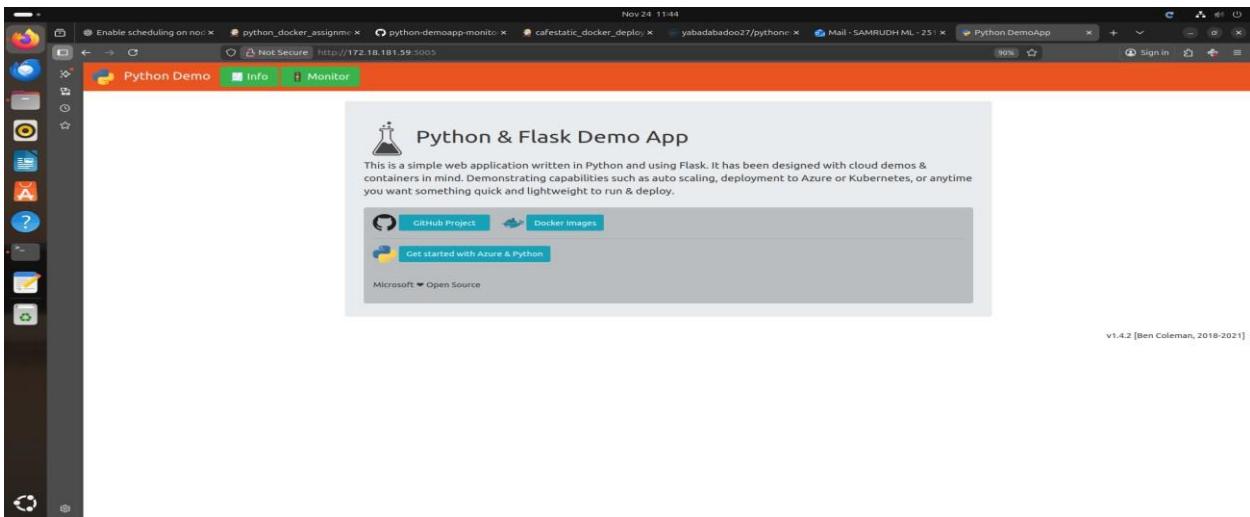
2. Pipeline view



3. Docker images running

```
ns1s@k8s-master-node: ~ $ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
db5008d3398d 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
c8f50fbcaab6 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 likhithi189/node-hello:latest "docker-entrypoint.s..." 2 days ago Up 2 days 0.0.0.0:3005->3000/tcp, [::]:3005->3000/tcp helloapp1
10649d2884fa likhithi189/node-hello:latest "docker-entrypoint.s..." 2 days ago Up 2 days 3000/tcp helloapp
cd71e78af29c likhithi189/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
```

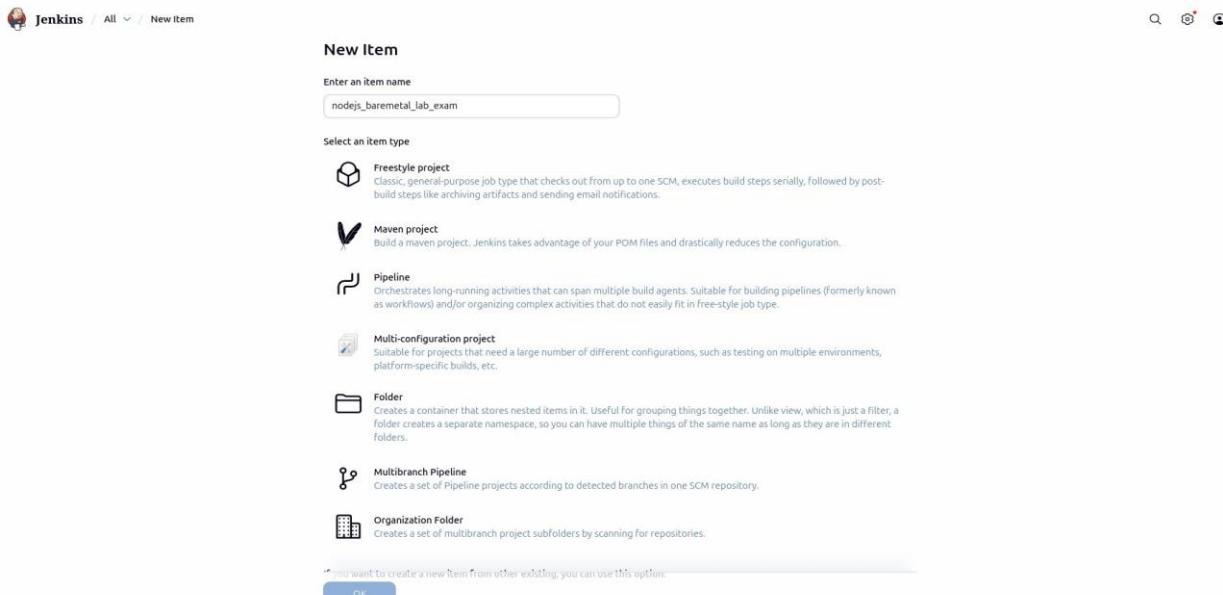
4. Python monitoring running on port:5005



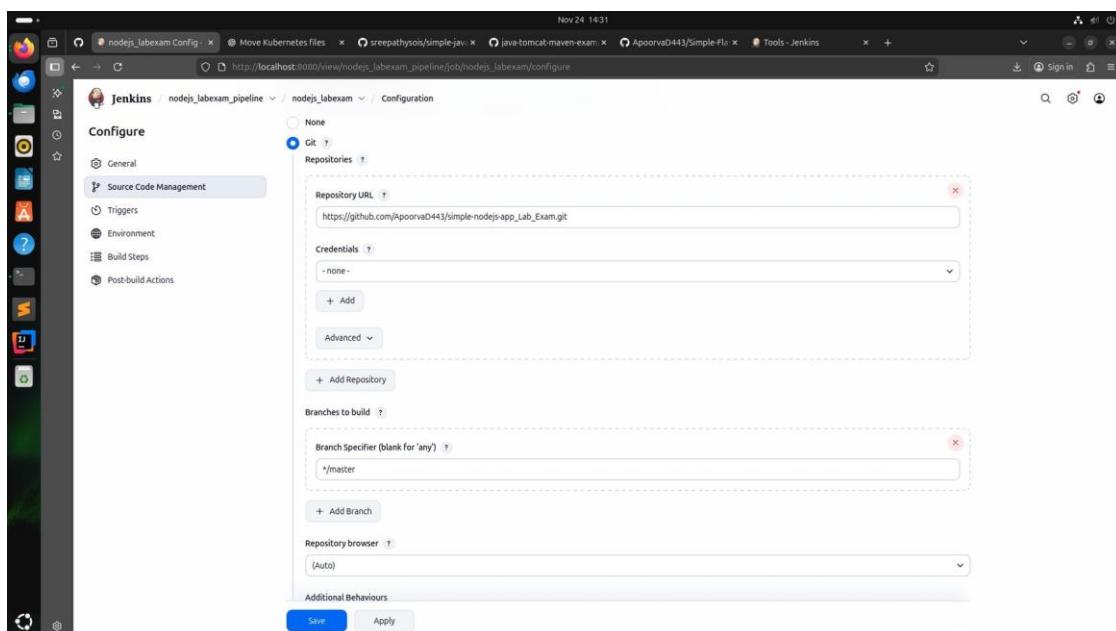
For Node js,

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

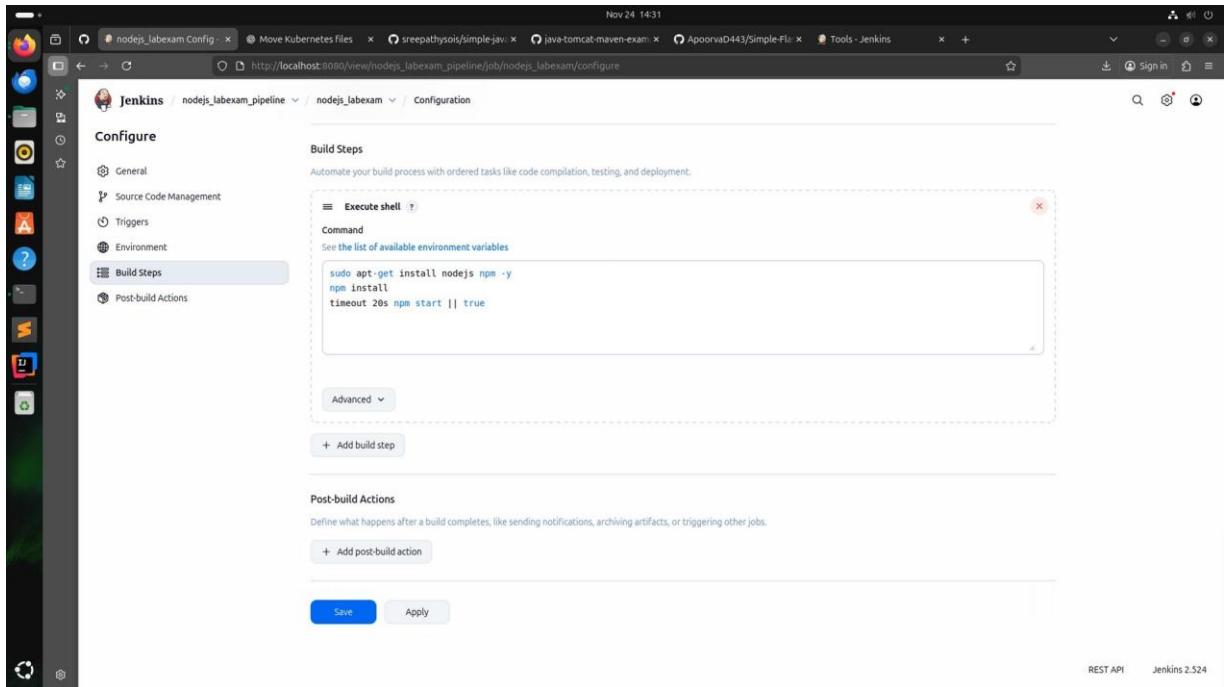
- Create a freestyle Jenkins Project
- Provide a name to Project for Example: nodejs_baremetal then click Ok



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



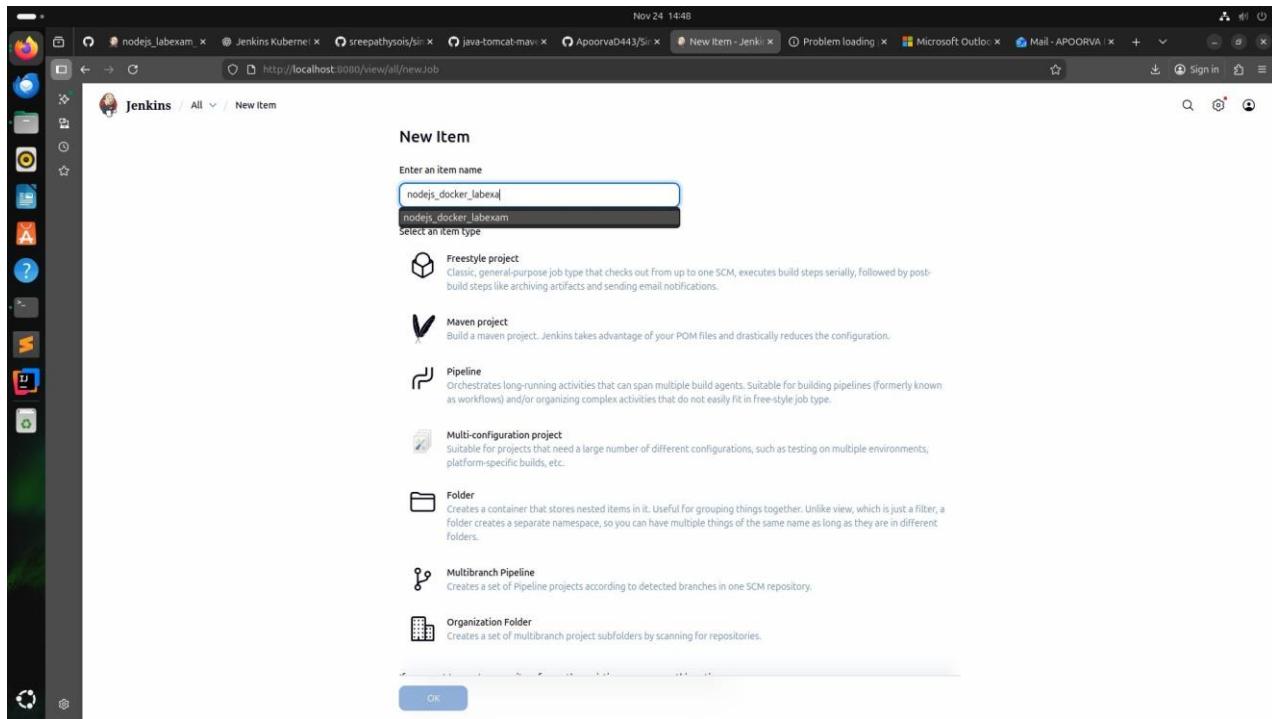
- Under the Build steps select execute shell and enter the below commands then click save.



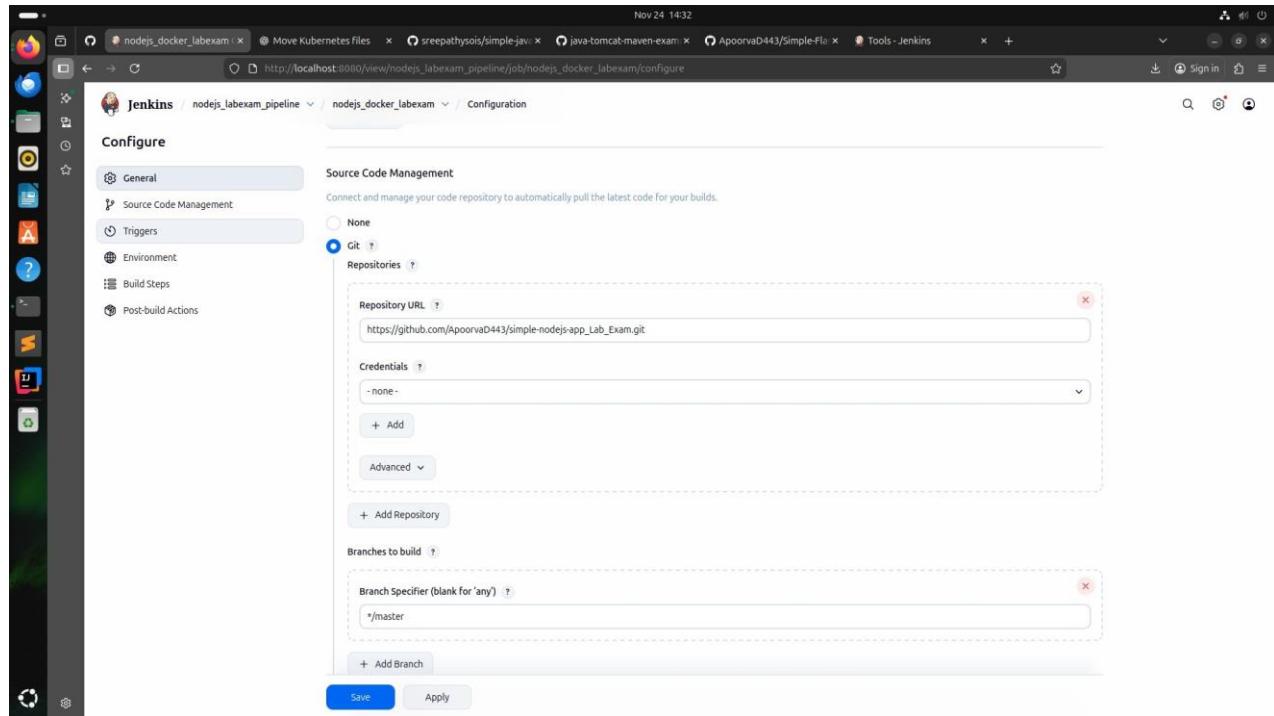
- Click on Build now.

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.

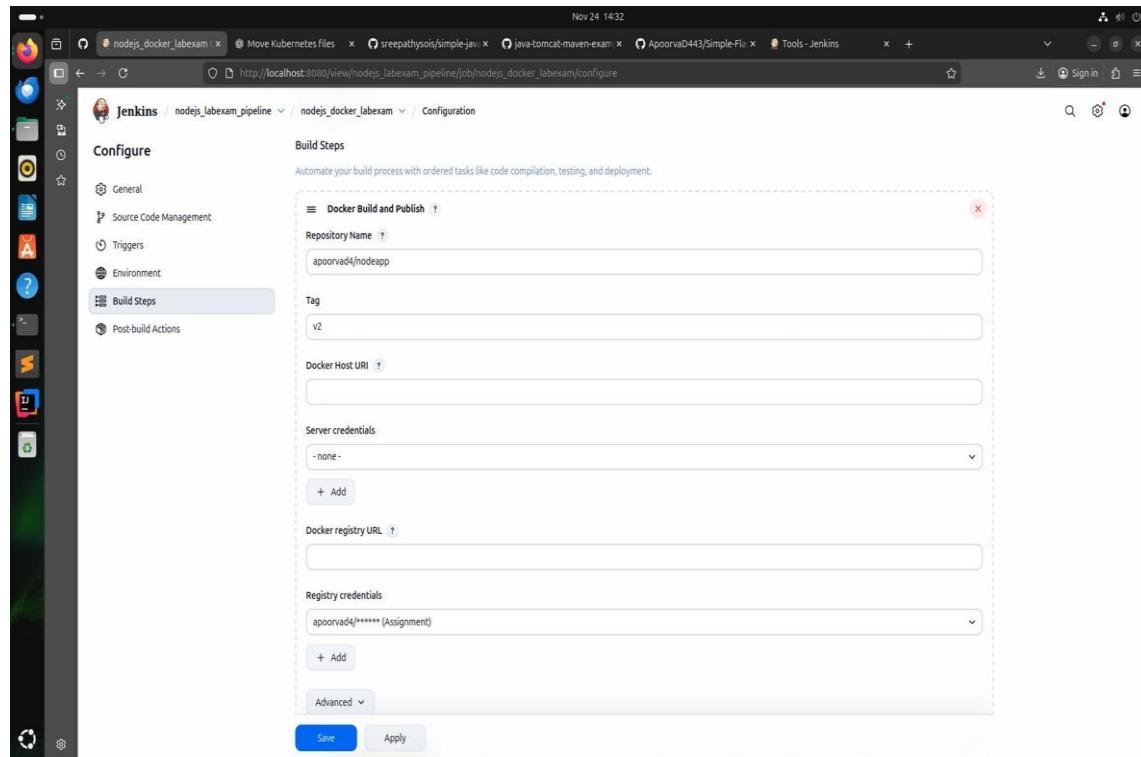


On the next page, select Source Code Management as Git, then provide the GitHub repository URL under Repository URL and specify the branch.

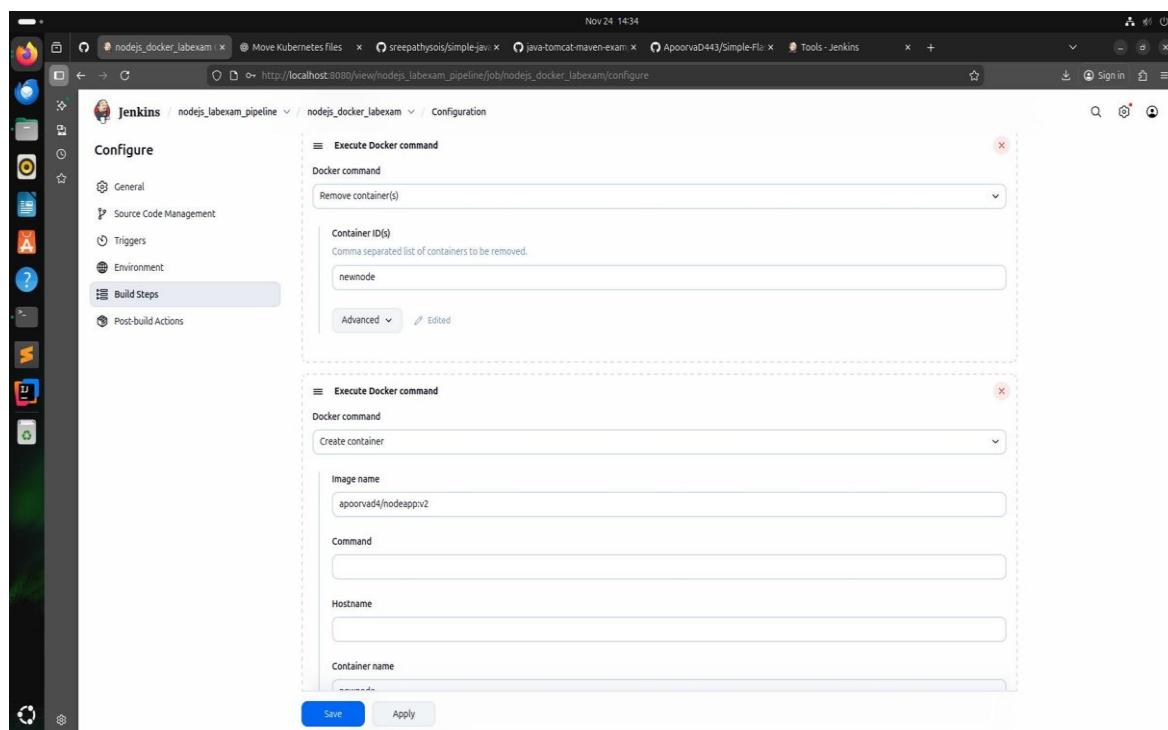


Under Build Steps, select Docker Build and Publish.
For Repository Name, enter <DockerHub_Username>/<image_name>

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).”



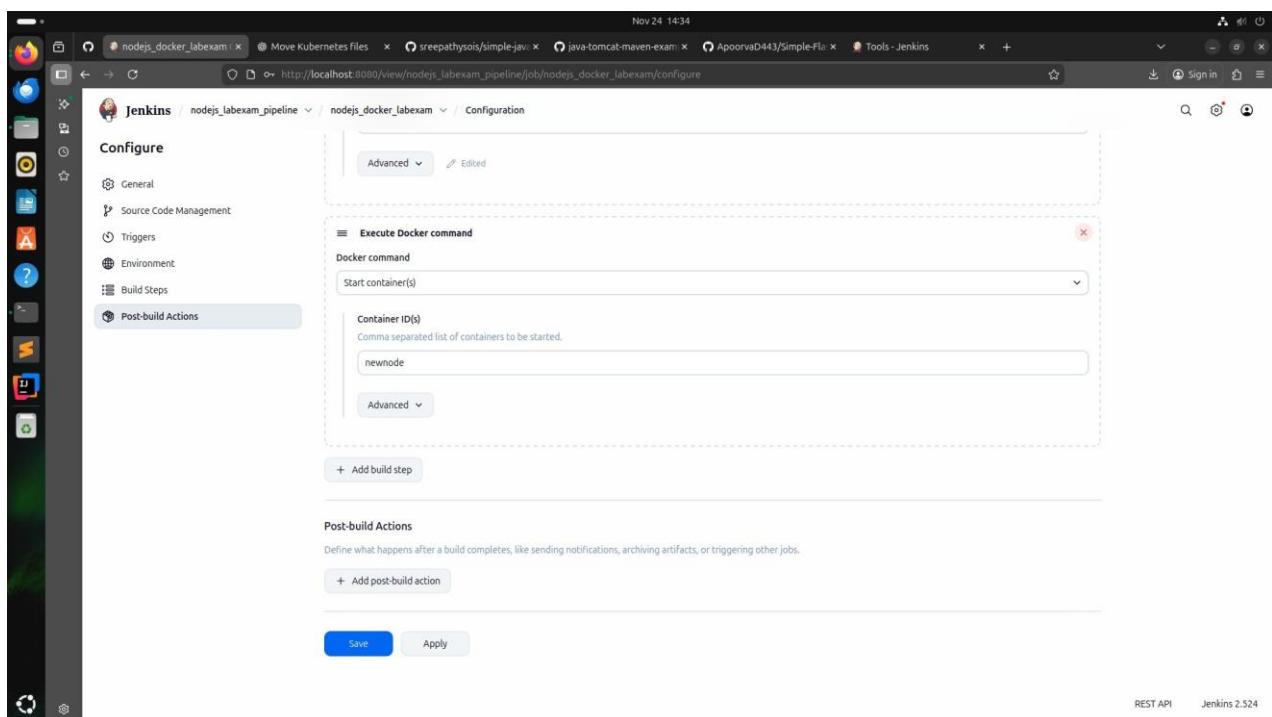
Click on “+ Add Build Step” to add another build step and select “Execute Docker command.”

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).

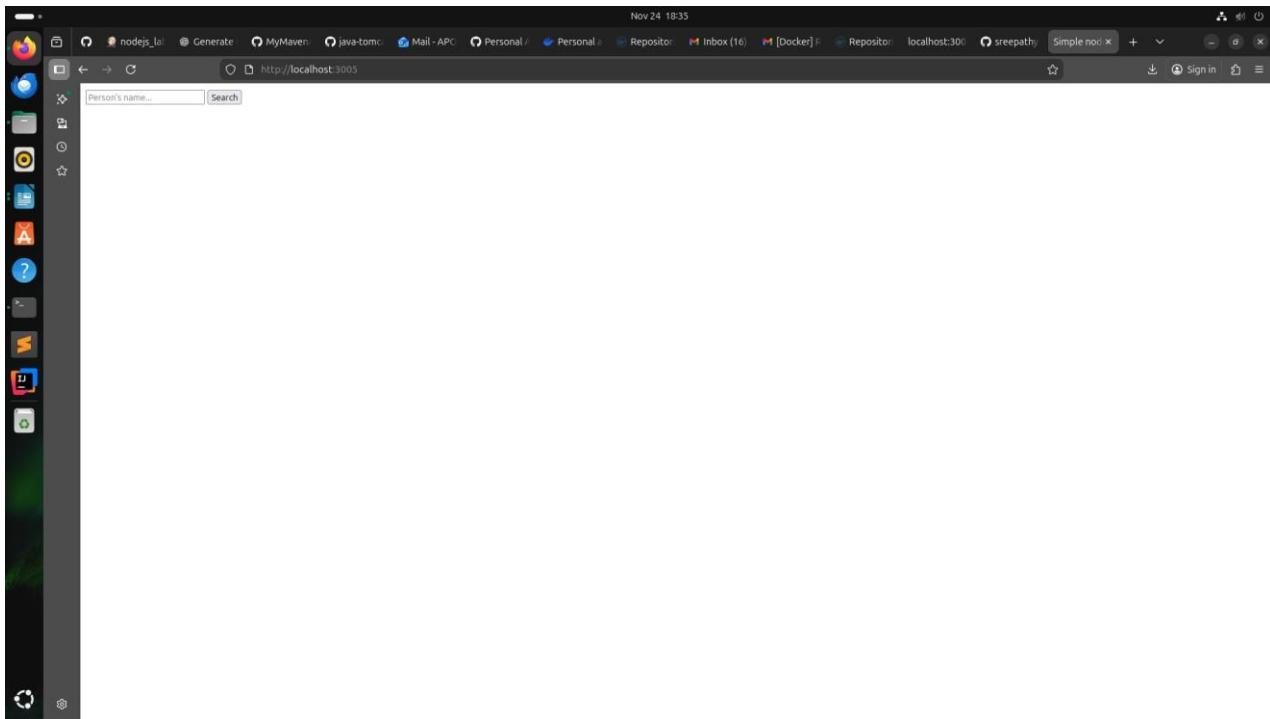
In the Container name field, provide the name of the container you want to create (e.g., new node)

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

Jenkins / nodejs_labexam_pipeline / nodejs_kubernetes_lab... / Configuration

Configure

General

Source Code Management

Triggers

Environment

Build Steps

Post-build Actions

Jenkins Credentials Provider: Jenkins

Kind: Kubernetes configuration (kubeconfig)

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

ID: kube123

Description:

Kubeconfig

Content: (radio button selected)

```
WjdqexNkRgXgWnBPVfTMQmhNKZ0fGZUTXKbzj9WfT0QmNEBjERUfEGsBRPjZVNrmh4vPUenjOU12p031IVZERKVYETWbPQDwVfFVYzAIDzZENLZjHgDZLd...  
dChdR7CmWrdetN6cQUpMh1DfV0hDg7davwamP7C1hQbvUr7ewd8sUecf2z8sn91uhm2vCgavhnlf2zj8LbDpD2zQm2wVfVf5DODzCzDlWvVpbkDxE...  
RVBT7jhe1jkCv3t3YlwwoSU9GcbJkQhNDJSE5SkdJ0iVVmXoANUiUfUGFNEdHtJuao9tTNASt1f6E51Q1piS2ZGzmltApowENyYThrlcLTSHR0NzdmfimMkMrd0fNcVR0Dv  
id0f4ZxzuQoh4sVlZ/mxDRPjCzsWOkGyThMHFCfEJEUcVrL1v0JUKC20JKM1MMHvhWVHEjMWNJanRUU0S2YhhdytVeFFFOnsdR3NWMFhos1dr0LGHWVBemhzLm1vbIE  
KUDUlvXJR5r202fFKNDVzylccsrU3ZxVkyQVVSFA2k3d3eHMRXMX3cWNik2b7QWMyRG9j5Ernv1hZ7zk5QwpBwxF2U0tLSWJkd0tvcmsj3ltd2ZQuJpmoCtpNivWk0dXVI  
QyY2UrQJxJwbtsQ3VaeeI5ci0LS0tRUEifJTSBQUklwQVRfIEFW50L50tCq==
```

From a file on the Jenkins master

From a file on the Kubernetes master node

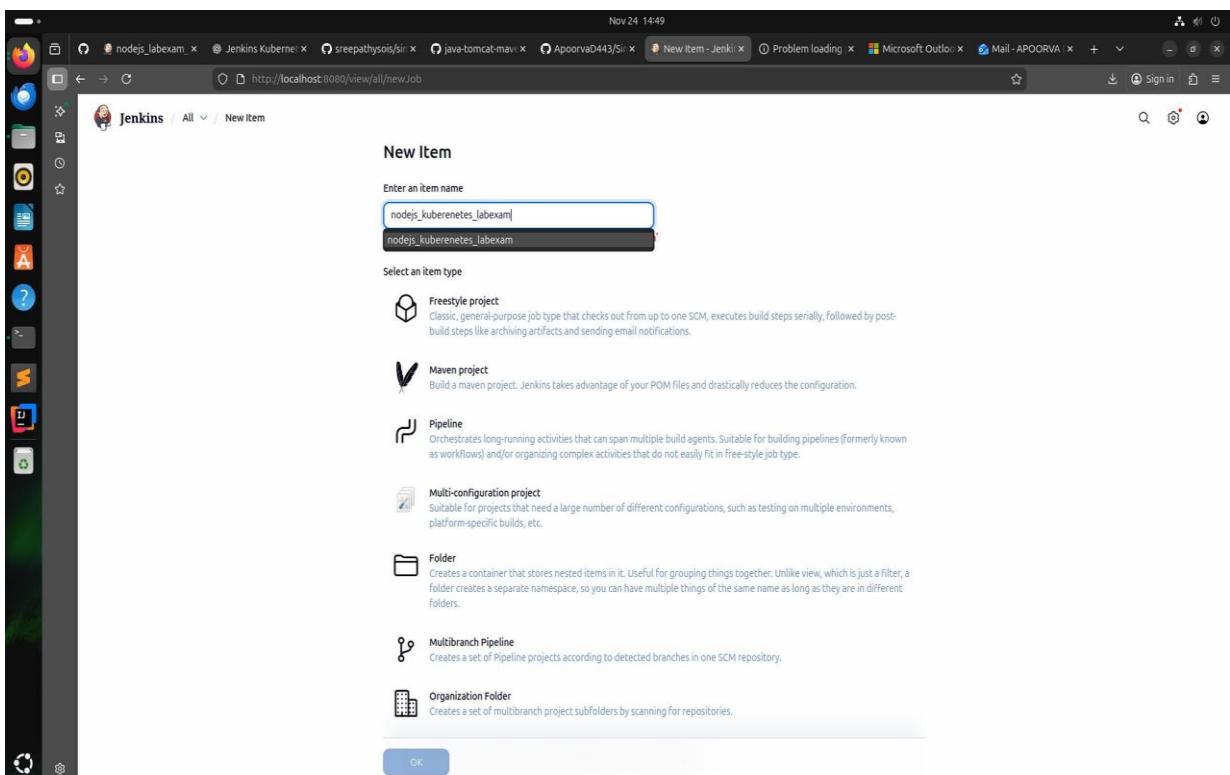
Save Apply

```

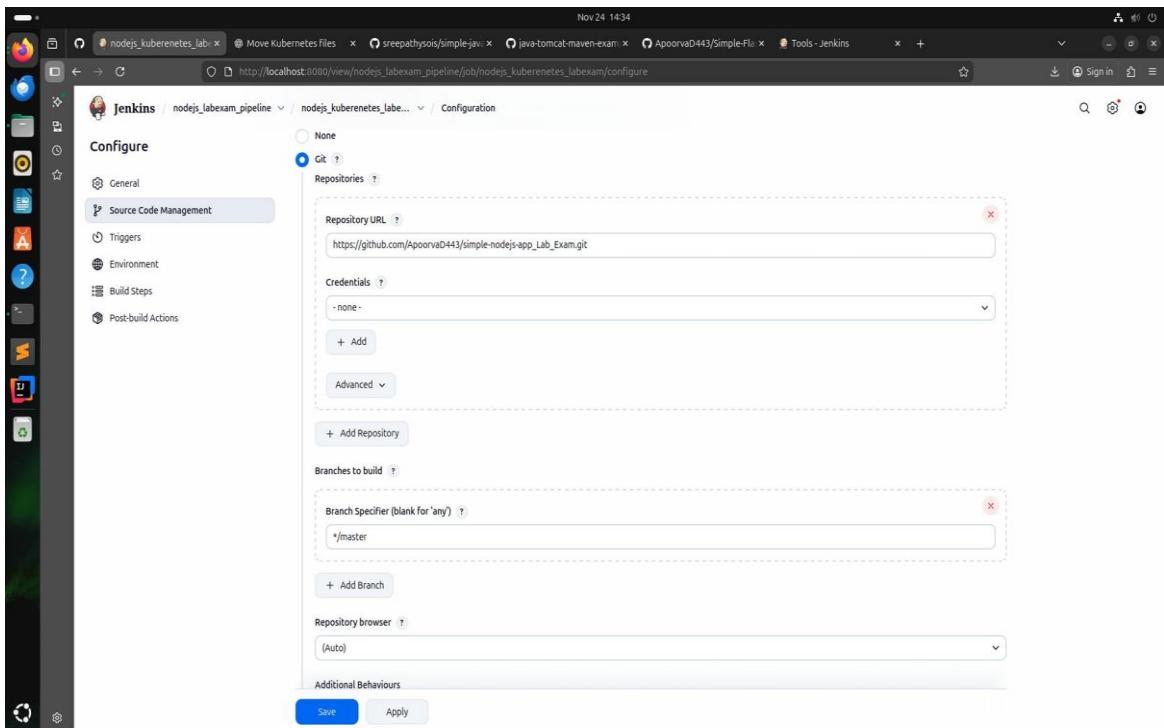
msis@k8s-worker-node-2:~$ cat ./kube/config
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: LS0tLS1CRUdJTBDRVJUSUZJQ0FURS0tL50tCk1JSURCVEND0WuYz0F3SUjBz0lJTLf4NDN4NRUT3NRFZSkvWklodnNOQVFTEJRQxDdGVETUJF0ExVUUKQXHNS2EzVnlawEp1WhlsBGN6QWVG
dz85TRFe1qQXhNeK1STkRaU23MHO0V4T1Vne165TN0RfPh1TUveApFekFS0d0V0k1BTVRD0bQxw1WkeWtVjbawE132zdFa1JBMEd0dU3FH01LM0RRRUJBUWVQTRJQkR3QxdnZ8VLCkFvSUJUWmdedscnM5fd1c1RSdhnUkd6Z1B2
U8gvgV0RvDhsNjRwojZRN0Qn1V3BRV0cvZlxodTaya04KR0W0EVrWDRMZDcyS2o0YnhzaE1u0DZWMERveEzAr2VNa1pCNDAbzD1tYkc4tn9BzTzJSt3e19kbGV02F3U0q0abFxbfdhcGxuNjVZXF0ZFN5dJFETLY4VLUV2V1xamovc2VB
TGHL2zd5k3d3EvbSy0H0uMlbt1eteUWn2FkCcxky9w5EtqcnzJNt1NTBNCpTJVGdHd1RD3WEVtaE15SEy0WmJheplu0m0rXpDLEVBZWN6dEdhN2ZLyNekap9yCg9uBFRTWRkzLNDWxJUTjYrTlvW01013ajhZwpb15L0hzcmYz
cET3eWRFxJ0Cj0cK3BOYtUzeA0P0kBRzPxeVxeR08ExWREd0vCL3dRUF35UReF0CkjhTz1Uk1CQY4RUJUQRUBUuvgtU0ExVREZ1FxQkJUSU91UmSGU3JNpHaE9U5wpiYU8z
ZmRnMEDsDg6eQVYKQm0d0vhSRUVeAkN2dwmXnxjbTvsZedWek1BMEdu3FH01M0RRRUJDU1B0tQRJF03RdLzPr1zjApkN0ZMbjy2vRzakYwM0wKwNkzbJ0UdwYUxFoxL1MDLsTeUya1BwWG9RnDqUSq5ynAxS1Y3VmtrYML
a1pnClVFN0Gz1T1VwMqdaPVveHvrdm1aVfFayy9SR2PbjY5Zm1JR0ZndnowTVZGbjhHNWtYWFVLHhtUmZ350t3L2UkDgtS3M3Q0R1T1B2MHTSKY1SmprD02k5cTJ0UkdocwtLrmtoZ3hMw12dTLz51VaWEzvMwv2MzJXNpHaE9U5wpiYU8z
RXJSUzlycxFM2kZ2uGZtdnpNOVzNexhTmhp2BSSEFM0EYTFxK3NQVFLndFPWmYwcE5ncVZWMXgrCkZr3hvbdnT3ZSEMezdEvhZ001UxP0q1pbwxEckJJTctGvY83czlN11kDt2LcxGd1UTrv0XZpRGVueCsKvk1vRV10QjuMG11
Ci0tL50tR5EIEF01URjRkL0DQVRFLs0tL50K
    server: https://17.18.181.34:6443
  name: kubernetes
contexts:
- context:
    cluster: kubernetes
    user: kubernetes-admin
  name: kubernetes-admin@kubernetes
current-context: kubernetes-admin@kubernetes
kind: Config
preferences: {}
users:
- name: kubernetes-admin
  user:
    client-certificate-data: LS0tLS1CRUdJTBDRVJUSUZJQ0FURS0tL50tCk1JSURVEND0WuYz0F3SUjBz0lJTLf4NDN4NRUT3NRFZSkvWklodnNOQVFTEJRQxDdGVETUJF0ExVUUKQXHNS2EzVnlawEp1WhlsBGN6QWVGdZB
5TRFe1qQXhNeK1STkRaU23MHO0V4T1Vne165TN0RfPh1TUveApFekFS0d0V0k1BTVRD0bQxw1WkeWtVjbawE132zdFa1JBMEd0dU3FH01LM0RRRUJBUWVQTRJQkR3QxdnZ8VLCkFvSUJUWmdedscnM5fd1c1RSdhnUkd6Z1B2
zRFFQkFRVFBNElCRHdd2dnRutb0b1CQVfb0dndjMkQgV053eBrekYm1hezexnbmZfa2z3YVNRd3Z0Wj1Ckwx0bkw1zRz5n0w8a1vElbIuH3R1U1MoPqz16MUE4Y4TMF1hmtXBzR04x0Hgxu1hNG5teh1HtD1Q0t1TTU
zbiRIdiH2U5Z0T16ST0RfBt0EUBF0T3hxc51Ck0zTnk0M9XSFhJbFNkV242ekw3SFdZTjdVazVTM28vNdr31xehVoSnVnd5vEc84bxJjN1hR0tq1zW1w1UWkUdHnV1/s/Vpdvdrwbk1z0D1wzkrnNFJUvhfjamJMMC9WREx02k9MNEQ2M1do0q0
zaZDahH0NhhFV1k3eJN3TQpByNFWpjhCNIckVERaYmTzRn9qdnVnb0t1evdn2x0N3hvW19hWcr13sdg9abk9r9eF3azB6akF3c4ptRytTCmCrG1FSG1xZcrtFpXRTF8201COLPHalzQz0lUN0TRHOTVZER3Rtvd1F0QxdJRM9eQVRC295
WSFNVRUREQUsKQndnckJnRuzCUNNEQipbTUJnTz1Uk1CQY4RUfQfQnf0j9hQfTVE13UvLNQfBRKzEzKwH1zLeF1QYXhndApXsk1Sox03uVphM1hNQTbHQ1nR1X1jyjNeeUUVCQ3dVQE8JSU8BUUhr1dBmjfwyz11Vg16Shp1Mu5nRukChB
6Ck91a1TzN2T0VgTRh1ZL4WmZLQWJfTrfJb1Mfb9v0gVp0vdaZtLgnkWn80Tg1ZzE5WjY1NvDx14WmZLZUf1NTYKGvNsYh0MfbukhQ5vpa0uNUZWRtdu2TuFv0dUc3dx0zu2QxyLBsq2vIUVJtS0ldQy9Q31ZQ5jU3dnlwpvTZVUL2V
RU0V3c1zv5JdJSTF1LMDBVsvg4RDNzQ1p5bzF0QnJLRDQSc5ebc9cdNz1VnL1N2pnRGRFUzFvTHcCkNxUnVytTk1M1Rup4WEhNzkd2b1LzT1dsNv8SRXUzRmY1ZE1Lz21Gckh1z2vRRIQvz02IMzhRc2H0MVR0eGnvb0MKEk3eDF552VfY1t1B
BeFFqMwxtcLqyNz0WERjcl0da3A0Z1zlnFvYcXhJQzFTUV1hsk5BRLsrxwJRCf10tL50tRUE1nfU1RjkL0DQVrFLs0tL50K
    client-key-data: LS0tLS1CRUdJTBDRVJUSUZJQ0FURS0tL50tCk1JSURVEND0WuYz0F3SUjBz0lJTLf4NDN4NRUT3NRFZSkvWklodnNOQVFTEJRQxDdGVETUJF0ExVUUKQXHNS2EzVnlawEp1WhlsBGN6QWVGdM
5UNBWhdVFB0RfJfUS2CJmRnTWRvnh1SjVzY21p0W13axJzTQ1Y1LkcnIw03Z0VNUx1LnRhdBRRHcvYzNSCnlpMhakzFwYn5wEdU0vhNf0EV/HdFfZbV25cm2tvjbawE10wDsdGfxNhdz0p1vErwR0tCJdtsWI
ad2QvUfZyDzH1VrWm5JmkSUDFReTdYenk9stQzTb0v0cJvUdC5nu5bNz2NUz1S1rDp0k4REFWnptZk1fbglvdzJxMzjV0Yk3N29Kwzkb691JnVrl0fZGzj3sNfb206GeYnFhlpnwTVZNSk5NHns281aH7rbmDRWhCm1jYznfBwLz0t1fJRE
RQJUb6b1lCQVFDt5tCV3iXuK13eJvUvY0qkVt1zNzEyZ1davOpnZjNreU1ydvJENTrXuLk1Udczak1zQx3pQjprcfpkMz28uJj0u1j3hjdrMhZ1QVgjcj1BRE0tZGh2NEQ2tEd40UrArVWdVbhzEh0vB8UMkd1Y1B5k01G6VYraGhqeJdQ33
D0kxQ0nFpL2NDTzN0MioRlVPNtC3UKtLJH0WzzasTaVDNFRtBMMFsawFTG0bxkr3Vl03dFF3Row0kv2k2ChM1VfcvXTFDu04zgptZk1tuXz1yeE13cUzWmDh13rUf2rX5nqydnvnhZQnQSThQnQvdnt4THzndXnhdHszM52VB
1Rzh0VER2C0RNYU9wNvYsd3RNzZcyw1LdemE5T1h3SFfLME9u0W43VnDyZty1tTURHSUxRMRtazFoa3FUGyyY28xTMk3a3ceKeu6algyduUbb0dQcQ1y0tDrFv09hd0bhkh1kR1zVzS2U5nQnd1R0zGbhW2HBmcZsVu0RveJz0Ccap
0d1tu2p1c1G5bmX3V1tbw14rMs2Tz9vNTRzN1RdVfE3Zlaca81k0WvRkAxR24z2XvRn1RwvRrje1p1CivsQ0Jz2d1z70Q2zysv51VnbfMz1zlmYzH0Jr1vZezOzzmfzbj1vaXkrbhlisaEdJLzr0hTb10w9H0kPPTHQKvNy0yztZ59
MYTRxMGlycLdRxxCzTJkRv1zdh1LkRjcvQnRlb1haGy1TlpzanE1QVr1c1n1a2Nym295QXN3YwpzTVFCaxFa0WZ5L3ZBckFscTdpY0w2cjjgxRkNqxFZ03pk1LzNgdVwfpu0xrnzGx40j1JMK3M0eTBTVd0cUSCCit503hCdEo1R0FDTfSrGt
oF9-Nm11T991-MM4v0u1u5Q1Inz5n1venc0h04RTd0s11V1dca2NnPy0okP0WMTKy0hN070TC71TYP0E7-0Y0T900-MEP1-d3P1H-0523D0P2Y71N1tMm4M1V7vun1H011-zM1-2T31V71DXYXY5TIV0Po0Va-YEh9eVz-1M1-2T31V71Dz7Gm-Sr0v

```

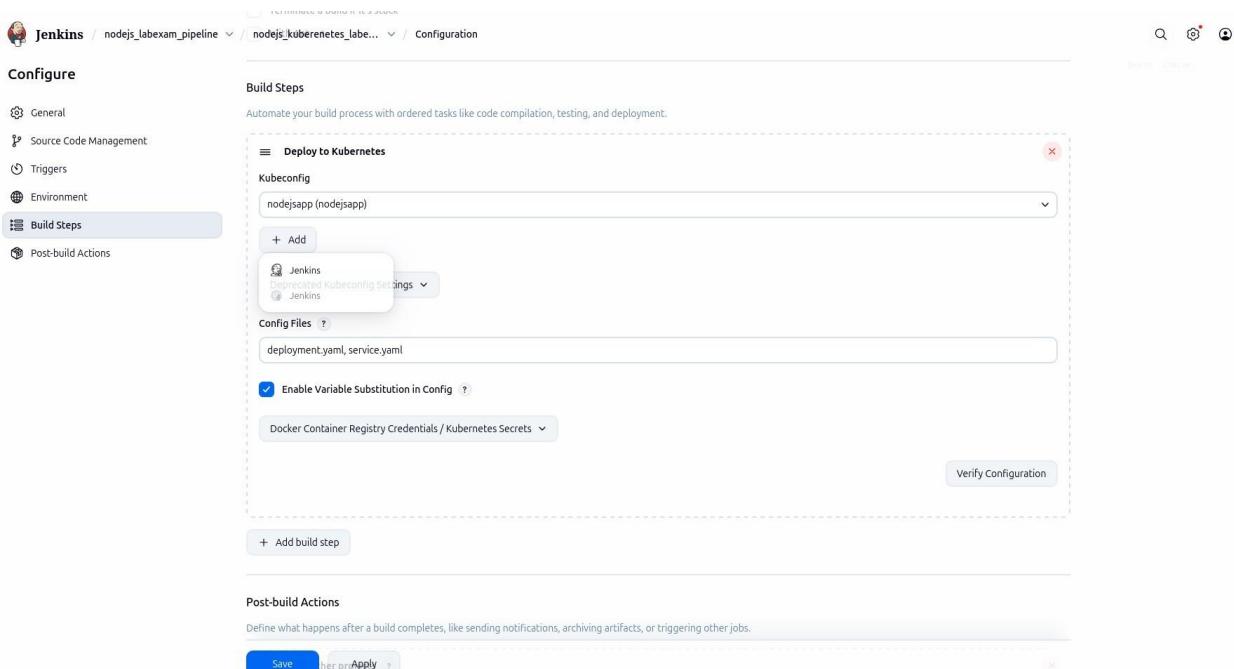
- Create a freestyle Jenkins Project
- Provide a name to Project for Example: nodejs_kubernetes then click Ok.



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

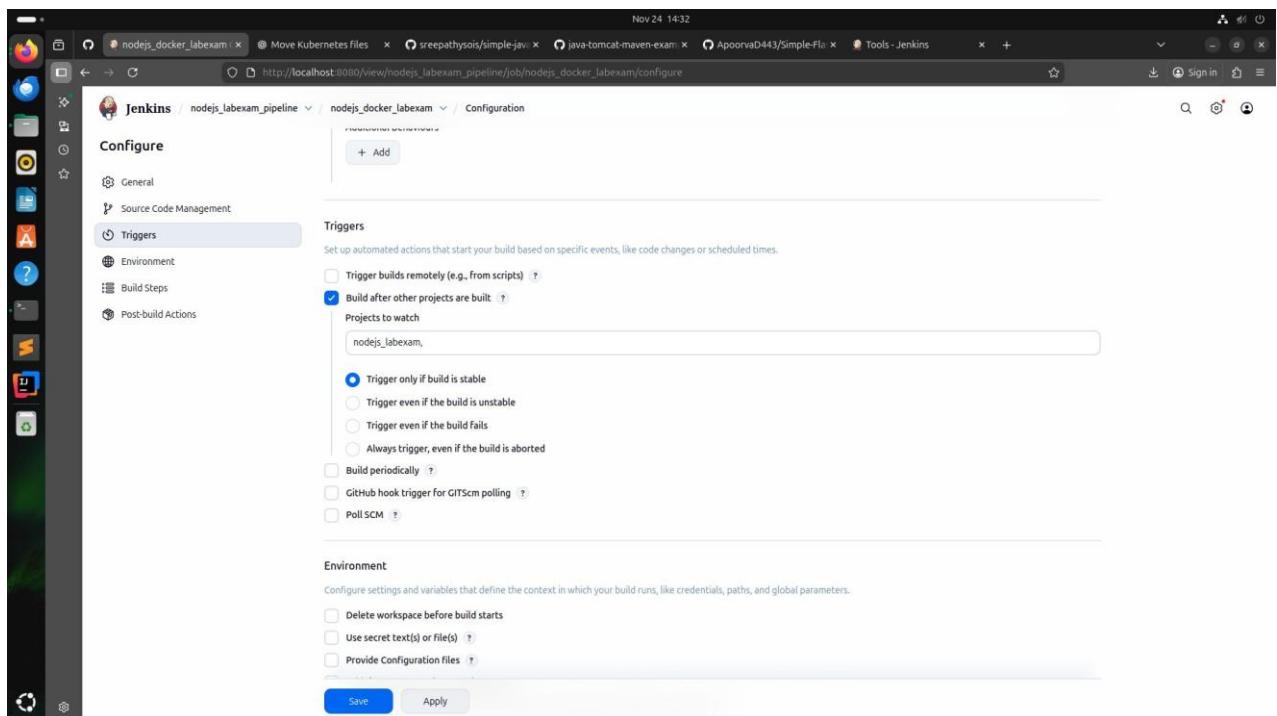
We can access the web page at,

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

Jenkins / nodejs_labexam_pipeline / nodejs_kubernetes_lab... / Configuration

Configure

Triggers

Set up automated actions that start your build based on specific events, like code changes or scheduled times.

Trigger builds remotely (e.g., from scripts) ?

Build after other projects are built ?

Projects to watch

nodejs_docker_labexam,

Trigger only if build is stable

Trigger even if the build is unstable

Trigger even if the build fails

Always trigger, even if the build is aborted

Build periodically ?

GitHub hook trigger for GITScm polling ?

Poll SCM ?

Environment

Configure settings and variables that define the context in which your build runs, like credentials, paths, and global parameters.

Delete workspace before build starts

Use secret text(s) or file(s) ?

Provide Configuration files ?

Add timestamps to the Console Output

Configure Kubernetes CLI (kubectl) (deprecated, use the multi credentials one instead) ?

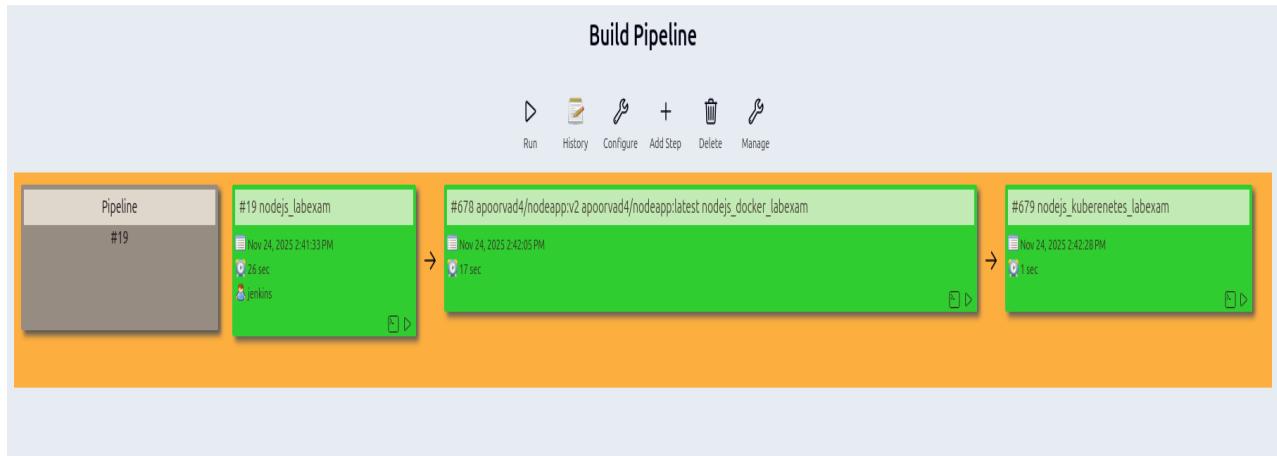
Configure Kubernetes CLI (kubectl) with multiple credentials

Inspect build log for published build scans

Provide Java home URL/Path to PATH

Save **netesApply(dect)** ?

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

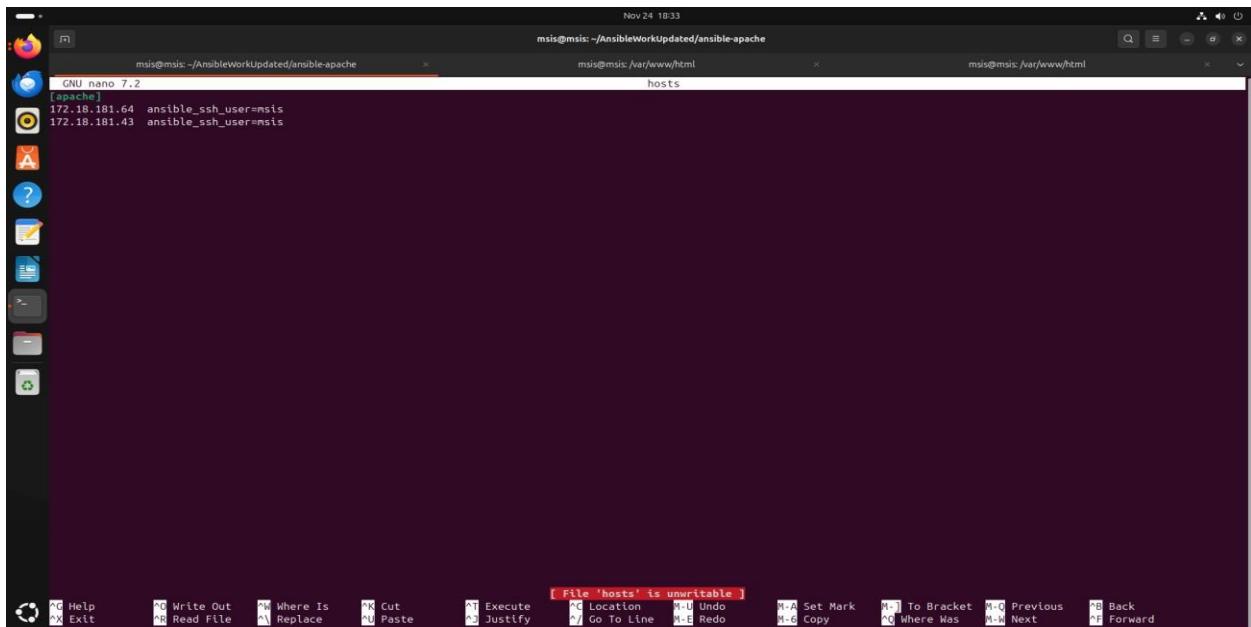
```
$ sudo git clone https://github.com/sreepathysois/AnsibleWorkUpdated.git
```

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
```

```
msis@msis:~/AnsibleWorkUpdated/ansible-apache
msis@msis:~/AnsibleWorkUpdated/ansible-apache$ sudo nano hosts
msis@msis:~/AnsibleWorkUpdated/ansible-apache$ sudo ansible-playbook apache.yml -i hosts -c ssh --ask-pass -K
SSH password:
BECOME password[defaults to SSH password]:  

PLAY [apache] *****
TASK [Gathering Facts] *****
ok: [172.18.181.64]
ok: [172.18.181.43]  

TASK [install apache2] *****
changed: [172.18.181.64]
changed: [172.18.181.43]  

PLAY RECAP *****
172.18.181.43      : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
172.18.181.64      : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

4. Run the lamp.yml

- go to lampansible

```
$ cd .../lampansible
```

-confirm the hosts

```
msis@msis:~/AnsibleWorkUpdated/ansible-apache
msis@msis:~/AnsibleWorkUpdated/ansible-apache$ ./lampansible/hosts
GNU nano 7.2
[lampserver]
172.18.181.64 ansible_ssh_user=msis
172.18.181.43 ansible_ssh_user=msis
```

- run the lamp.yml

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

```
A msis@mstis:~/AnsibleWorkUpdated/lampanstable$ sudo ansible-playbook lamp.yml -i hosts -c ssh --ask-pass -K
SSH password:
BECOME password[defaults to SSH password]:
PLAY [lampserver] *****

TASK [Gathering Facts] *****
ok: [172.18.181.64]
ok: [172.18.181.43]

TASK [install lamp stack] *****
ok: [172.18.181.64]
ok: [172.18.181.43]

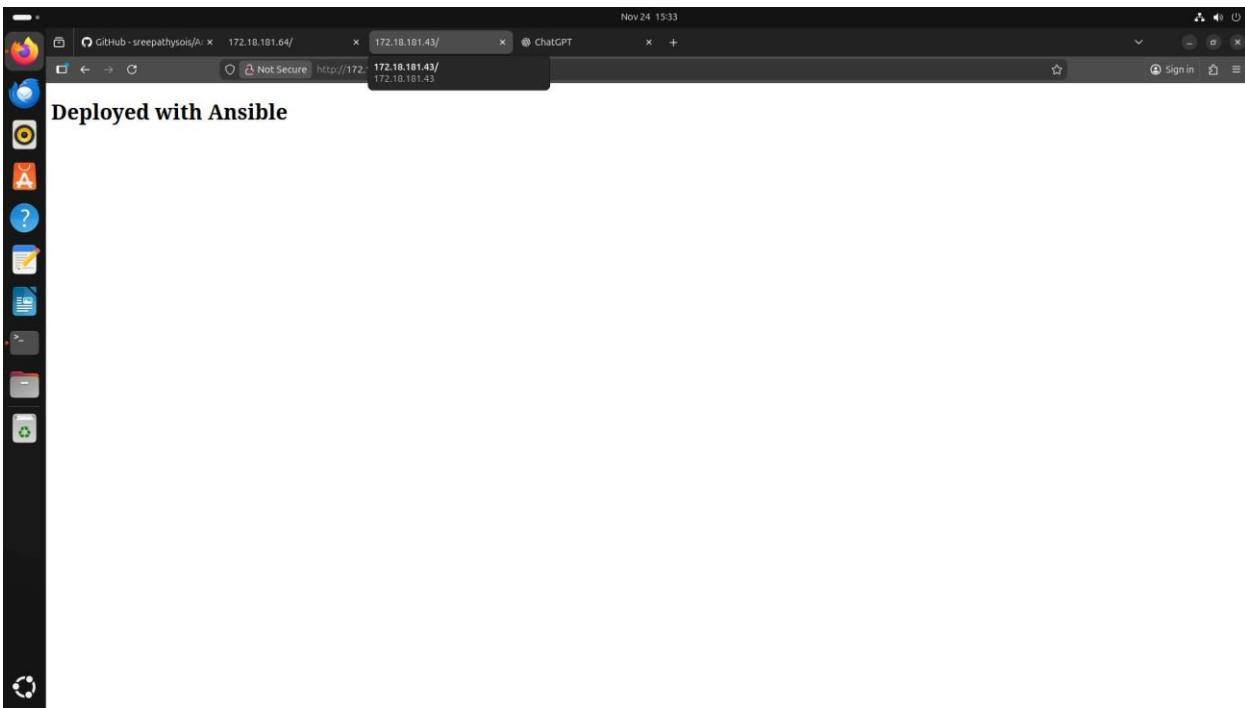
TASK [start apache service] *****
ok: [172.18.181.64]
ok: [172.18.181.43]

TASK [start mysql service] *****
ok: [172.18.181.64]
ok: [172.18.181.43]

TASK [create target directory] *****
ok: [172.18.181.43]
ok: [172.18.181.64]

TASK [deploy index.html] *****
changed: [172.18.181.64]
changed: [172.18.181.43]

PLAY RECAP *****
172.18.181.43      : ok=6    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
172.18.181.64      : ok=6    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```



5. Run the lamp_app.yml

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

```
msis@msis:~/AnsibleWorkUpdated/lampansible$ sudo ansible-playbook lamp_app.yml -i hosts -c ssh --ask-pass -K
BECOME password[defaults to SSH password]:  

PLAY [lampserver] *****  

TASK [Gathering Facts] *****  

ok: [172.18.181.64]  

ok: [172.18.181.43]  

TASK [Install lamp stack] *****  

ok: [172.18.181.43]  

ok: [172.18.181.64]  

TASK [start apache service] *****  

ok: [172.18.181.43]  

ok: [172.18.181.64]  

>_  

TASK [start mysql service] *****  

ok: [172.18.181.64]  

ok: [172.18.181.43]  

TASK [create target directory] *****  

ok: [172.18.181.43]  

ok: [172.18.181.64]  

TASK [deploy index.html] *****  

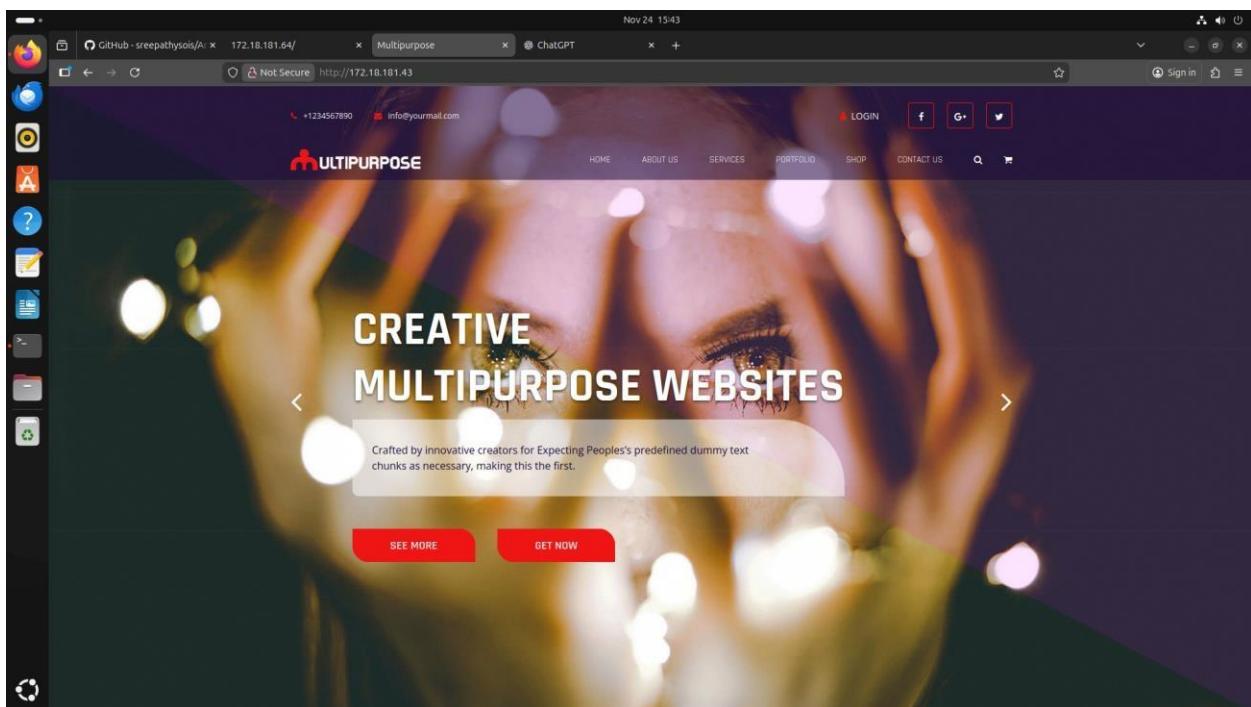
ok: [172.18.181.64]  

ok: [172.18.181.43]  

PLAY RECAP *****  

172.18.181.43 : ok=6    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0  

172.18.181.64 : ok=6    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```



6. Run the baremetal

- go to `cafe_app_ansible_bare_metal_deploy`
- ```
$cd ../cafe_app_ansible_bare_metal_deploy
```

-confirm with hosts

```
Nov 24 19:02
msis@msis: ~/AnsibleWorkUpdated/cale_app_ansible_bare_metal_deploy
GNU nano 7.2
[webservers]
172.18.181.64 ansible_ssh_user=msis
172.18.181.43 ansible_ssh_user=msis
[webservers:vars]
ansible_python_interpreter=/usr/bin/python3
[dbservers]
172.18.181.64 ansible_ssh_user=msis
172.18.181.43 ansible_ssh_user=msis
[dbservers:vars]
ansible_python_interpreter=/usr/bin/python3
```

-run the baremetal

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

```
Nov 24 19:04
msis@msis: ~/AnsibleWorkUpdated/cale_app_ansible_bare_metal_deploy
ok: [172.18.181.43]

TASK [cafeweb : Ensure /var/www/html/mompopcafe directory exists] ****
ok: [172.18.181.64]
ok: [172.18.181.43]

TASK [cafeweb : Download application code tarball] ****
ok: [172.18.181.64]
ok: [172.18.181.43]

TASK [cafeweb : Extract application code to /var/www/html/mompopcafe] ****
changed: [172.18.181.64]
changed: [172.18.181.43]

TASK [cafeweb : Ensure /etc/cafe directory exists] ****
ok: [172.18.181.64]
ok: [172.18.181.43]

TASK [cafeweb : Copy getAppParameters.php template to /etc/cafe] ****
ok: [172.18.181.43]
ok: [172.18.181.64]

TASK [cafeweb : Copy getAppParameters.php template to /var/www/html/mompopcafe/] ****
changed: [172.18.181.64]
changed: [172.18.181.43]

TASK [cafeweb : Update Apache configuration] ****
ok: [172.18.181.64]
ok: [172.18.181.43]

RUNNING HANDLER [mysql : Enable and start MySQL server] ****
ok: [172.18.181.64]
ok: [172.18.181.43]

RUNNING HANDLER [cafeweb : Reload Apache] ****
ok: [172.18.181.64]
ok: [172.18.181.43]

PLAY RECAP ****
172.18.181.43 : ok=19 changed=6 unreachable=0 failed=0 skipped=1 rescued=0 ignored=0
172.18.181.64 : ok=19 changed=6 unreachable=0 failed=0 skipped=1 rescued=0 ignored=0
```

Nov 24 19:05

GitHub - sreepathysois/A: x 404 Not Found x Welcome to Mom & Pop Café x ChatGPT x outlook.office.com/mail/: x Repositories | shazilham: x Personal access tokens | : x +

Not Secure http://172.18.181.43/index.php

# 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!

*Top 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: x 404 Not Found x Mom & Pop Café Menu x ChatGPT x outlook.office.com/mail/: x Repositories | shazilham: x Personal access tokens | : x +

Not Secure http://172.18.181.43/menu.php

# Mom & Pop Café

Home Menu Order History

### Pastries

**Croissant**  
€1.50  
Fresh, buttery and fluffy... Simply delicious!  
Quantity:

**Donut**  
€1.00  
We have more than half-a-dozen flavors!  
Quantity:

**Chocolate Chip Cookie**  
€2.50  
Made with Swiss chocolate with a touch of Madagascar vanilla  
Quantity:

**Muffin**  
€3.00  
Banana bread, blueberry, cranberry or apple  
Quantity:

**Strawberry Blueberry Tart**  
€3.50  
Bursting with the taste and aroma of fresh fruit  
Quantity:

**Strawberry Tart**  
€3.50  
Made with fresh ripe strawberries and a delicious whipped cream  
Quantity:

The screenshot shows a Firefox browser window with multiple tabs open. The active tab displays an order confirmation for "Mom & Pop Café".

**Mom & Pop Café**

**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.

The browser's sidebar on the left contains various icons for other open tabs and windows.

# 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

```
Nov 21 16:00
msis@k8s-master-node: ~

msis@k8s-master-node: $ sudo apt update
sudo apt install -y build-essential libgd-dev openssl libssl-dev unzip apache2 php libapache2-mod-php php-gd
[sudo] password for msis:
Ign:1 https://pkg.jenkins.io/debian binary/ InRelease
Hit:2 https://pkg.jenkins.io/debian binary/ Release
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Hit:5 http://archive.ubuntu.com/ubuntu noble InRelease
Get:7 http://archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Hit:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb InRelease
Get:8 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [1,317 kB]
Get:9 http://archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:10 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [1,619 kB]
Get:11 http://security.ubuntu.com/ubuntu noble-security/main i386 Packages [354 kB]
Get:12 http://security.ubuntu.com/ubuntu noble-security/main Translation-en [216 kB]
Get:13 http://security.ubuntu.com/ubuntu noble-security/main amd64 Components [21.5 kB]
Get:14 http://archive.ubuntu.com/ubuntu noble-updates/main i386 Packages [554 kB]
Get:15 http://security.ubuntu.com/ubuntu noble-security/main amd64 c-n-f Metadata [9,452 B]
Get:16 http://archive.ubuntu.com/ubuntu noble-updates/main Translation-en [303 kB]
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 amd64 Components [175 kB]
Get:19 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 c-n-f Metadata [15.7 kB]
```

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

```
Nov 21 16:00
msis@k8s-master-node: ~

msis@k8s-master-node: $ sudo useradd nagios
sudo groupadd nagcmd
sudo usermod -a -G nagcmd nagios
sudo usermod -a -G nagcmd www-data
groupadd: group 'nagcmd' already exists
msis@k8s-master-node: $
```

### 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 C99... 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/types.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 nagiosstats /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 -d /usr/local/nagios/share/robots.txt
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

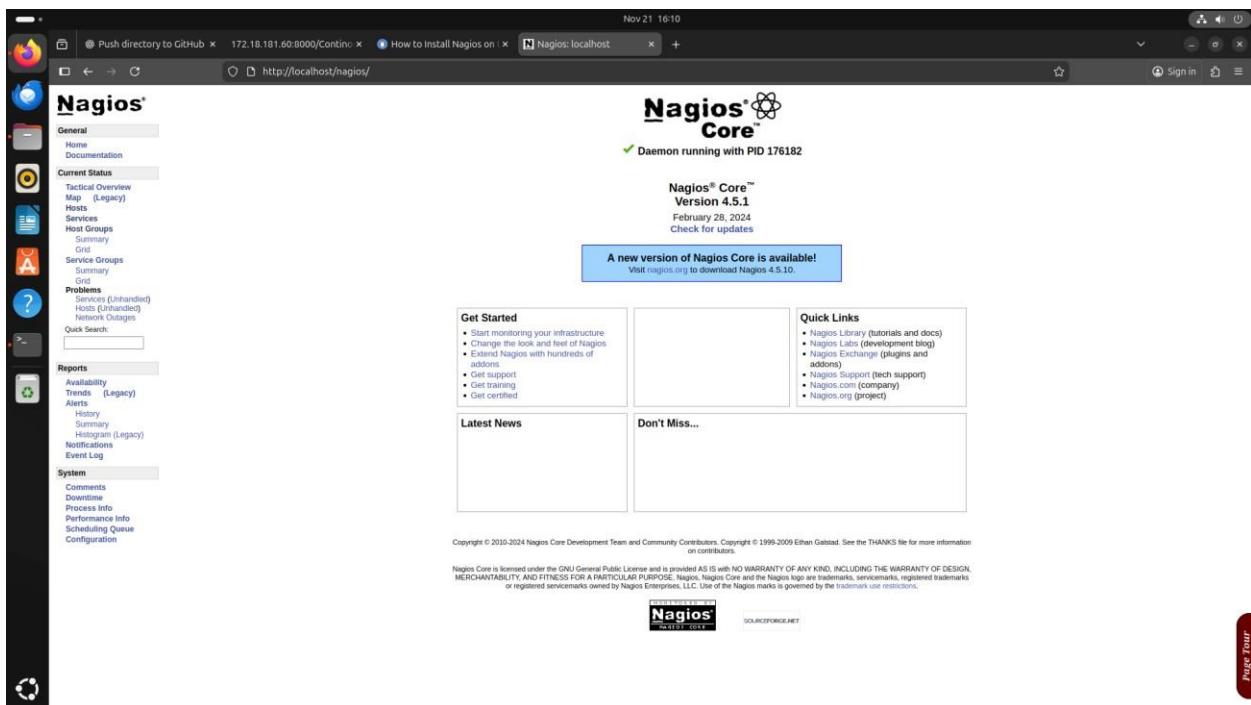
### 7. Configure Nagios Web Interface on Ubuntu 24.04 LTS

### 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:/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

```
GNU nano 7.2 /etc/nagios/nrpe.cfg

file to allow only the specified host to connect to the port
you are running this daemon on.

#
NOTE: This option is ignored if NRPE is running under either inetd or xinetd

[allowed_hosts=127.0.0.1, 172.18.181.67]
```

## 13. ./usr/local/nagios/libexec/check\_nrpe -H 172.18.181.67

```
msis@k8s-master-node:/tmp/nagios-plugins-2.3.3$./usr/local/nagios/libexec/check_nrpe -H 172.18.181.58
bash: ./usr/local/nagios/libexec/check_nrpe: No such file or directory
msis@k8s-master-node:/tmp/nagios-plugins-2.3.3$ cd /usr/local/nagios/libexec/
msis@k8s-master-node:/usr/local/nagios/libexec$ ls
check_apt check_dig check_flexlm check_ifoperstatus check_log check_ntp check_oracle check_rpc check_ssh check_udp remove_perfdata
check_breeze check_disk check_ftp check_ifstatus check_mailq check_nt check_overcr check_sensors check_ssl_validity check_ups urlize
check_by_ssh check_disk_smb check_hpjd check_imap check_mrtg check_ntp check_ping check_snmp check_ssntp check_uptime utils.pm
check_clamd check_dns check_http check_ircd check_mrtgtraf check_ntp_peer check_pop check_smtp check_swap check_users utils.sh
check_cluster check_dummy check_icmp check_jabber check_nagios check_ntp_time check_procs check_snmp check_tcp check_wave
check_dhcp check_file_age check_idc_smart check_load check_ntp check_ntpstat check_real check_spop check_time negate
msis@k8s-master-node:/usr/local/nagios/libexec$ sudo apt update
sudo apt install nagios-nrpe-plugin -y
[sudo] password for msis:
Ign:1 https://pkg.jenkins.io/debian binary/ InRelease
Hit:2 https://pkg.jenkins.io/debian binary/ Release
Hit:3 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:6 http://archive.ubuntu.com/ubuntu noble InRelease
Hit:5 https://prod-cdn.packages.k8s.io/repositories/isc:/kubernetes:/core:/stable:/v1.31/deb InRelease
Hit:7 http://archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:8 http://archive.ubuntu.com/ubuntu noble-backports InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
432 packages can be upgraded. Run 'apt list --upgradable' to see them.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
nagios-nrpe-plugin is already the newest version (4.1.0-1ubuntu3).
0 upgraded, 0 newly installed, 0 to remove and 432 not upgraded.
msis@k8s-master-node:/usr/local/nagios/libexec$./usr/lib/nagios/plugins/check_nrpe -H 172.18.181.58
NRPE v4.1.0
msis@k8s-master-node:/usr/local/nagios/libexec$ []
```

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

The screenshot shows the Nagios web interface with the following details:

**Current Network Status**  
Last Updated: Fri Nov 21 16:34:28 IST 2025  
Updated every 90 seconds  
Nagios Core: 4.5.1 - www.nagios.org  
Logged in as nageasadmin

**Host Status Totals**  
Up: 2, Down: 0, Unreachable: 0, Pending: 0  
All Problems: 2, All Types: 2

**Service Status Totals**  
Ok: 9, Warning: 0, Unknown: 0, Critical: 1, Pending: 0  
All Problems: 1, All Types: 1

**Host Status Details For All Host Groups**

| Host           | Status | Last Check          | Duration      | Status Information                        |
|----------------|--------|---------------------|---------------|-------------------------------------------|
| Linux_Host_001 | UP     | 11-21-2025 16:32:25 | 0d 0h 0m 22s+ | PING OK - Packet loss = 0%, RTA = 0.86 ms |
| localhost      | UP     | 11-21-2025 16:32:45 | 0d 0h 26m 3s  | PING OK - Packet loss = 0%, RTA = 0.09 ms |

**Results 1 - 2 of 2 Matching Hosts**

**Host Groups**

- Tactical Overview
- Map (Legacy)
- Hosts
- Services
- Host Groups
  - Summary
  - Grid
- Service Groups
  - Summary
- Problems
  - Services (Unhandled)
  - Hosts (Unhandled)
  - Network Outages
- Quick Search:

**Reports**

- Availability
- Trends (Legacy)
- Alerts
- History
- Summary
- Histogram (Legacy)
- Notifications
- Event Log

**System**

- Comments
- Downs
- Process Info
- Performance Info
- Scheduling Queue
- Configuration