DEVOPS Theory Assignment

Setup a Jenkins and Docker on Ubuntu 22.04

Jenkins Installation

To set up Jenkins on Ubuntu 22.04, we first need to install Java as it's a prerequisite for Jenkins:

sudo apt-get update

sudo apt install openjdk-17-jdk

After installing Java, we'll add the Jenkins repository and install Jenkins:

sudo wget -O /usr/share/keyrings/jenkins-keyring.asc https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key

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

sudo apt-get update

sudo apt-get install Jenkins

Once installed, check the Jenkins service status:

sudo systemctl status jenkins.service

To access Jenkins for the first time, you'll need the initial admin password:

sudo cat /var/lib/jenkins/secrets/initialAdminPassword

This command displays a one-time auto-generated password created during Jenkins installation that you'll use for the initial setup.

Open up any browser on your ubuntu system and hit the URL localhost:8080 or your systems IP address:8080 example 172.16.50.85:8080 to access Jenkins.

Docker Installation

For Docker installation, run the following commands:

sudo apt update

sudo apt install -y docker.io

sudo systemctl enable docker

sudo systemctl start docker

Verify Docker is running properly:

sudo systemctl status docker

docker –version

Apache and PHP Installation

To set up Apache web server with PHP support:

sudo apt update

sudo apt install apache2 -y

sudo systemctl status apache2

sudo apt install php libapache2-mod-php php-mysql -y

sudo systemctl restart apache2

Tomcat Installation

For Tomcat 9 installation and configuration:

sudo apt-get update

sudo apt install tomcat9

sudo apt install tomcat9-admin

Check Tomcat service status:

sudo systemctl status tomcat9.service

Configure Tomcat by editing the server.xml file:

sudo nano /etc/tomcat9/server.xml

Add or modify the connector configuration:

<Connector **port="8000"** protocol="HTTP/1.1"**change the port number to 8000 or any other**

address="172.16.50.85" Add the system IP Address

connectionTimeout="20000"

redirectPort="8443" />

Set up a Tomcat user for Jenkins access:

sudo nano /etc/tomcat9/tomcat-users.xml

Add the following user configuration:

<user username="**jenkins**" password="**Tomcat@123**" roles="**manager-script**"/>

Add the above line in the last line of the tomcat-users.xml file

Restart and check the status of the tomcat server

sudo systemctl restart tomcat9.service

sudo systemctl status tomcat9.service

If required Install JDK and restart Tomcat:

sudo apt install default-jdk –y

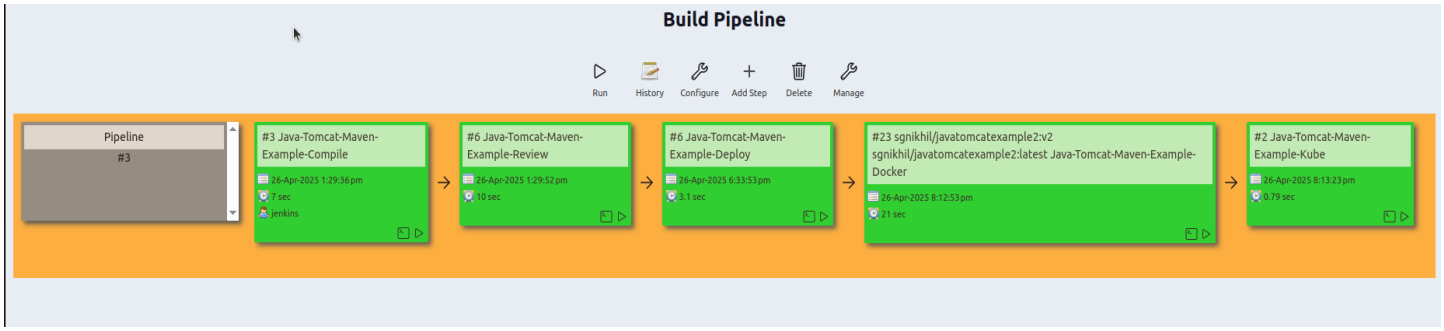
sudo systemctl restart tomcat9.service

sudo systemctl status tomcat9.service

Java-Tomcat-Maven-Example CI-CD Pipeline

Prerequisites

- Jenkins server up and running
- Required plugins installed: Git, Maven Integration, Docker, Kubernetes, and Deploy to Container (for Tomcat)
- Access to a Tomcat server (local or remote) for deployments
- Docker and Kubernetes cluster access if those steps are needed



Pipeline Steps Overview

- Compile
- Review
- Deploy
- Docker Build and Push, Run
- Kubernetes Deploy

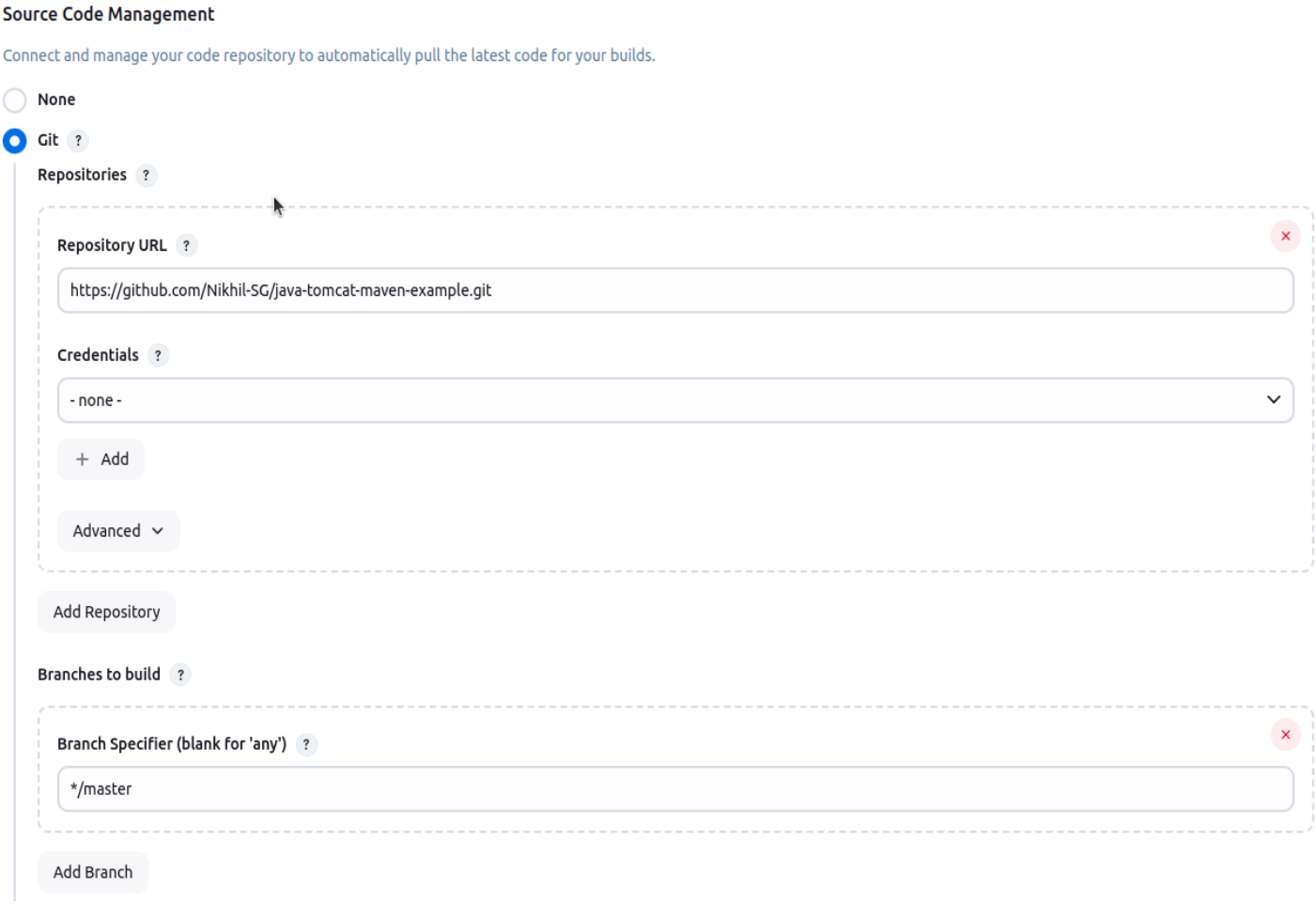
Java-Tomcat-Maven-Example-Compile

Source Code Management

Select Git.

Enter the repo URL: https://github.com/Nikhil-SG/java-tomcat-maven-example.git

Set branch to build: */master.



Build Steps

Add a build step: Invoke top-level Maven targets.

Choose Maven version: MyMaven.

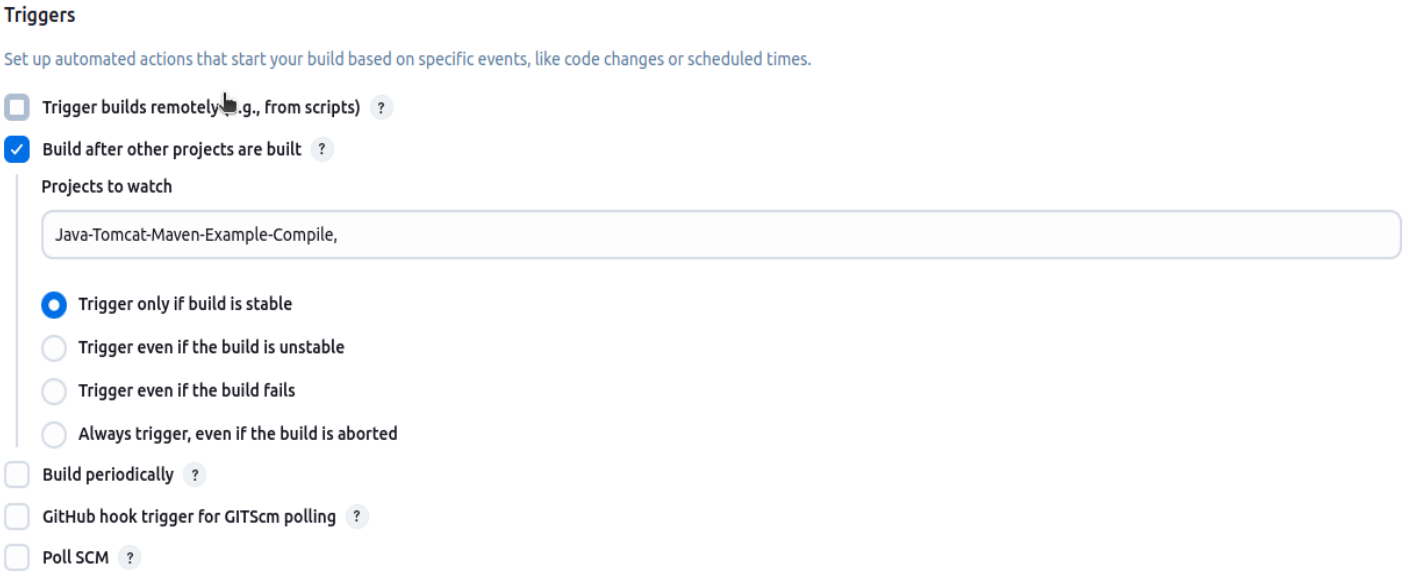
Set goals: clean compile.

This setup will pull the latest code from the master branch and compile it using Maven.

Java-Tomcat-Maven-Example-Review

Build Triggers

This job will start automatically after the Java-Tomcat-Maven-Example-Compile job finishes and only if that build was successful.



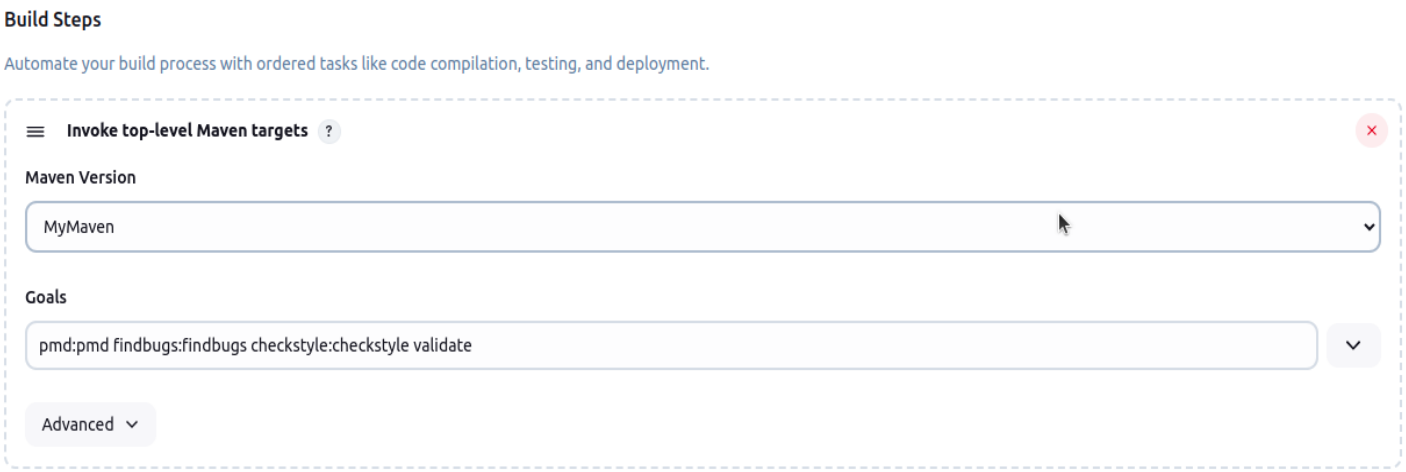
Build Steps

Uses Maven (MyMaven) to run code quality checks.

Maven goals:

pmd:pmd findbugs:findbugs checkstyle:checkstyle validate

This runs PMD, FindBugs, and Checkstyle tools to check for code issues and validate the project.



Post-build Actions

After the build, Jenkins collects and records results from static analysis tools:

CheckStyle (checks code style)

FindBugs (finds the bugs in the code)

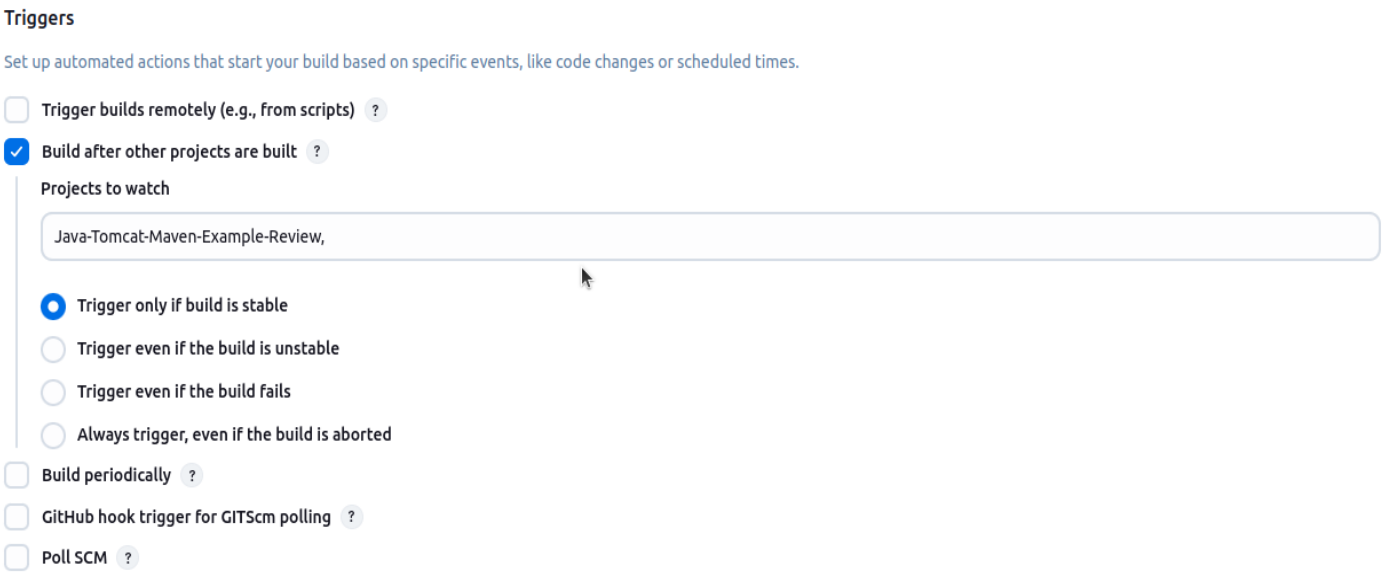
PMD (finds code problems)

This job runs code quality tools (PMD, FindBugs, Checkstyle) after a successful compile, and then records their results for you to review.

Java-Tomcat-Maven-Example-Deploy

Build Triggers

This job runs automatically after the Java-Tomcat-Maven-Example-Review job finishes successfully.

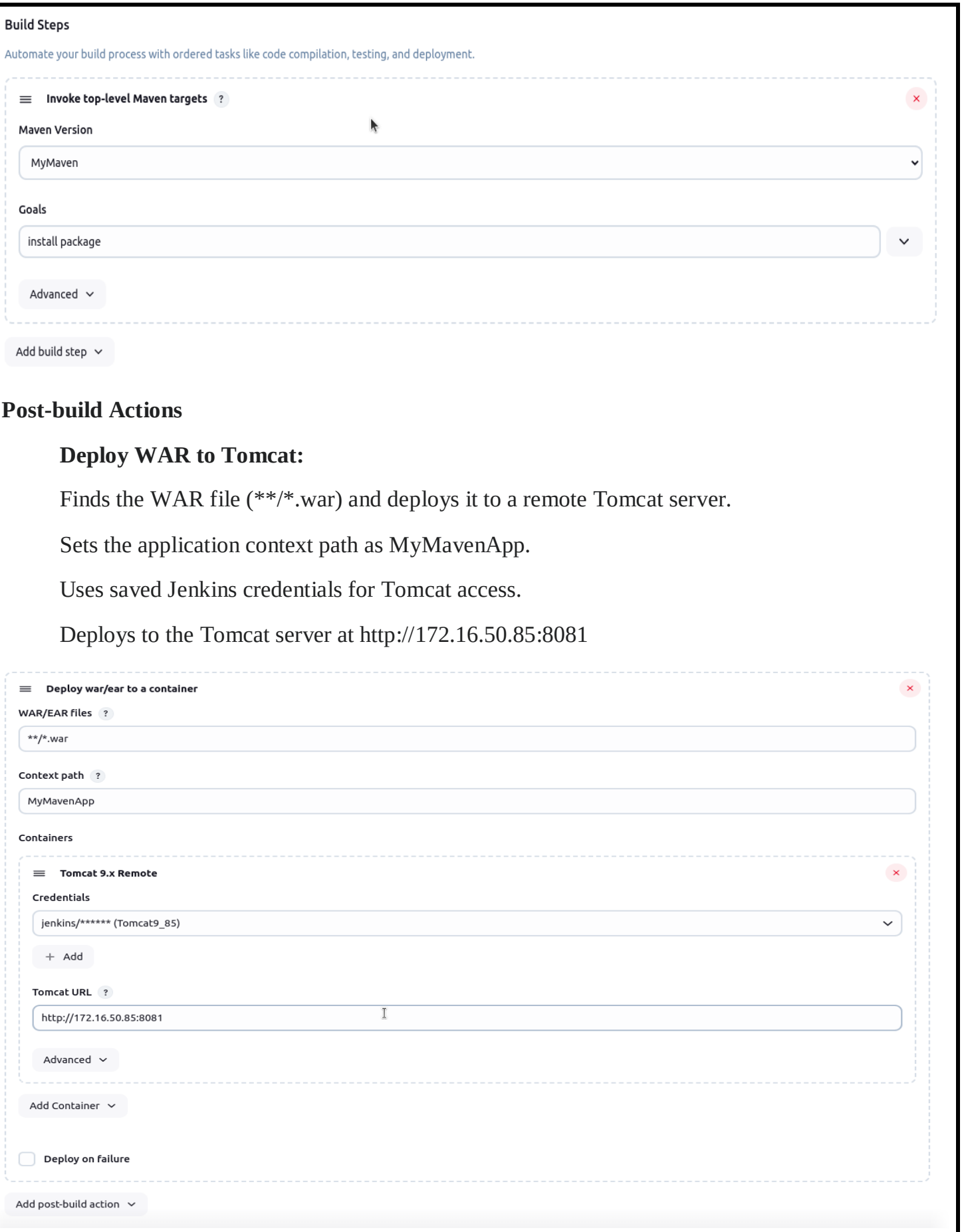


Build Steps:

Maven Build:

Uses Maven (MyMaven) to run install package goals.

This builds the project and creates a WAR file ready for deployment



Post-build Actions

Deploy WAR to Tomcat:

Finds the WAR file (**/*.war) and deploys it to a remote Tomcat server.

Sets the application context path as MyMavenApp.

Uses saved Jenkins credentials for Tomcat access.

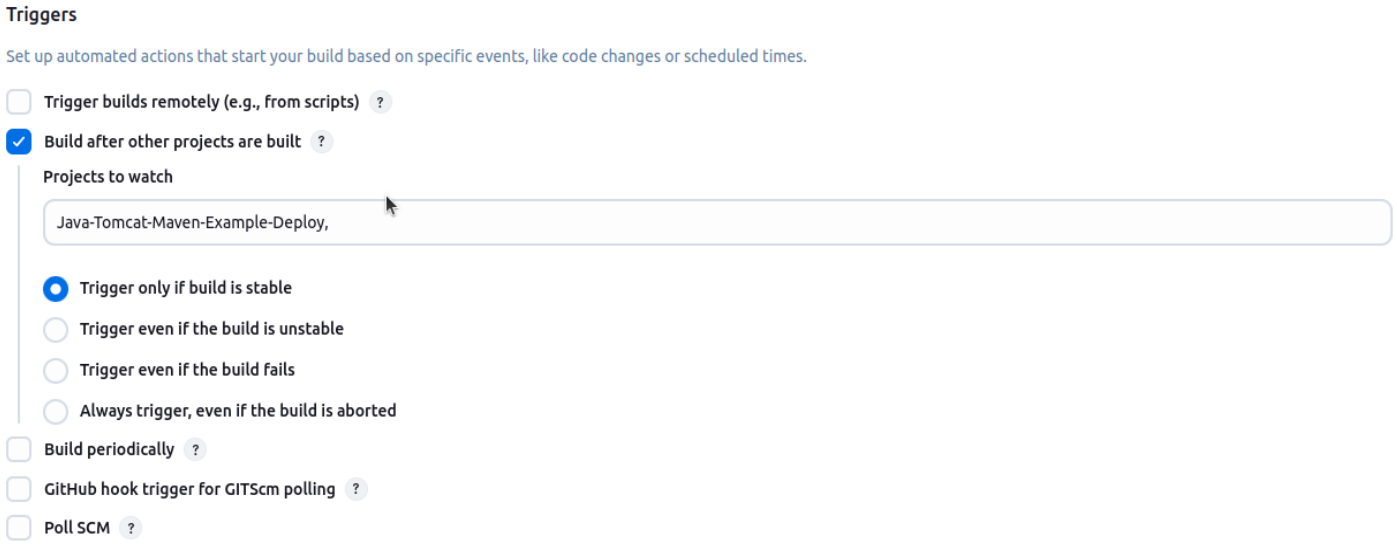
Deploys to the Tomcat server at http://172.16.50.85:8081

This job builds the project and automatically deploys the generated WAR file to a remote Tomcat server after the review stage is successful.

Java-Tomcat-Maven-Example-Docker

Build Triggers

This job runs automatically after the Java-Tomcat-Maven-Example-Deploy job finishes successfully.



Build Steps:

Maven Build

Uses Maven (MyMaven) with the goal:

package

This command packages your Java app into a WAR file.

Docker Build and Publish

Builds a Docker image using the generated WAR file.

Repository name: sgnikhil/javatomcatexample2

Tag: v2

Uses Docker credentials to push the image to Docker Hub.

Execute Shell

Runs a shell command to start a container from the new image

This job packages your app, builds a Docker image, pushes it to Docker Hub, and then runs the app in a Docker container.

Build Steps

Automate your build process with ordered tasks like code compilation, testing, and deployment.

Invoke top-level Maven targets

Maven Version

MyMaven

Goals

package

Advanced

Docker Build and Publish

Repository Name

sgnikhil/javatomcatexample2

Tag

v2

Docker Host URI

Server credentials

- none -

- none -

+ Add

Docker registry URL

Registry credentials

sgnikhil/****** (credentials to access nikhils docker)

+ Add

Advanced

Execute shell

Command

See the list of available environment variables

```
docker run -it -d -p 8087:8080 sgnikhil/javatomcatexample2:v2
```

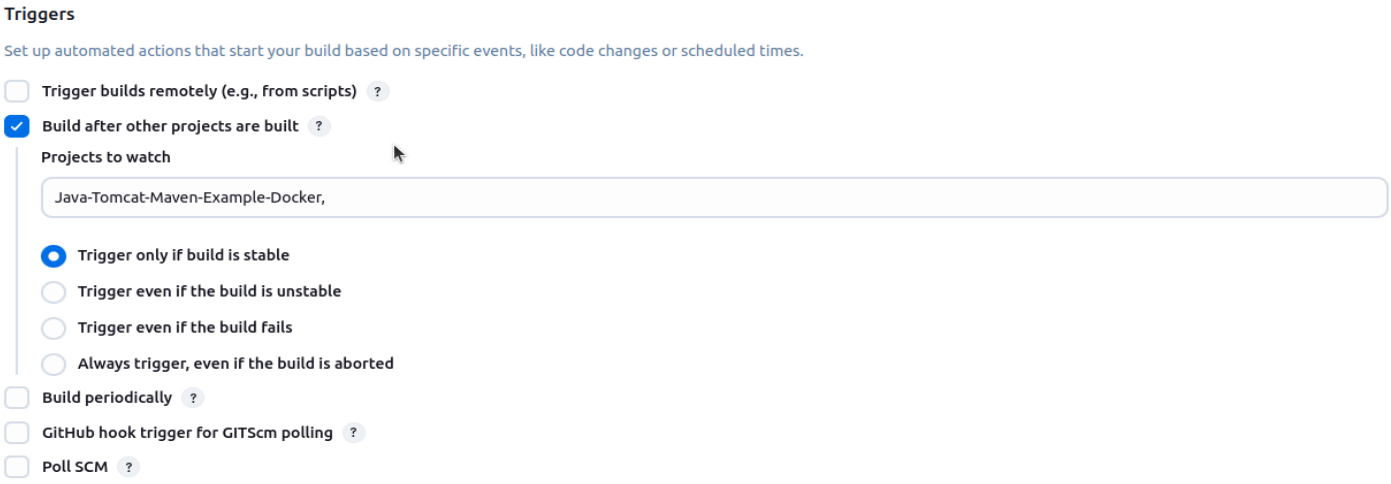
Advanced

Add build step

Java-Tomcat-Maven-Example-Kube

Build Triggers

This job runs automatically after the Java-Tomcat-Maven-Example-Docker job finishes successfully



Build Steps:

Deploy to Kubernetes Build Step

Kubeconfig:

Select the Kubernetes cluster credentials (here, "**47_49 (Master 47 Slave 49)**") so Jenkins can connect to your Kubernetes cluster.

Config Files:

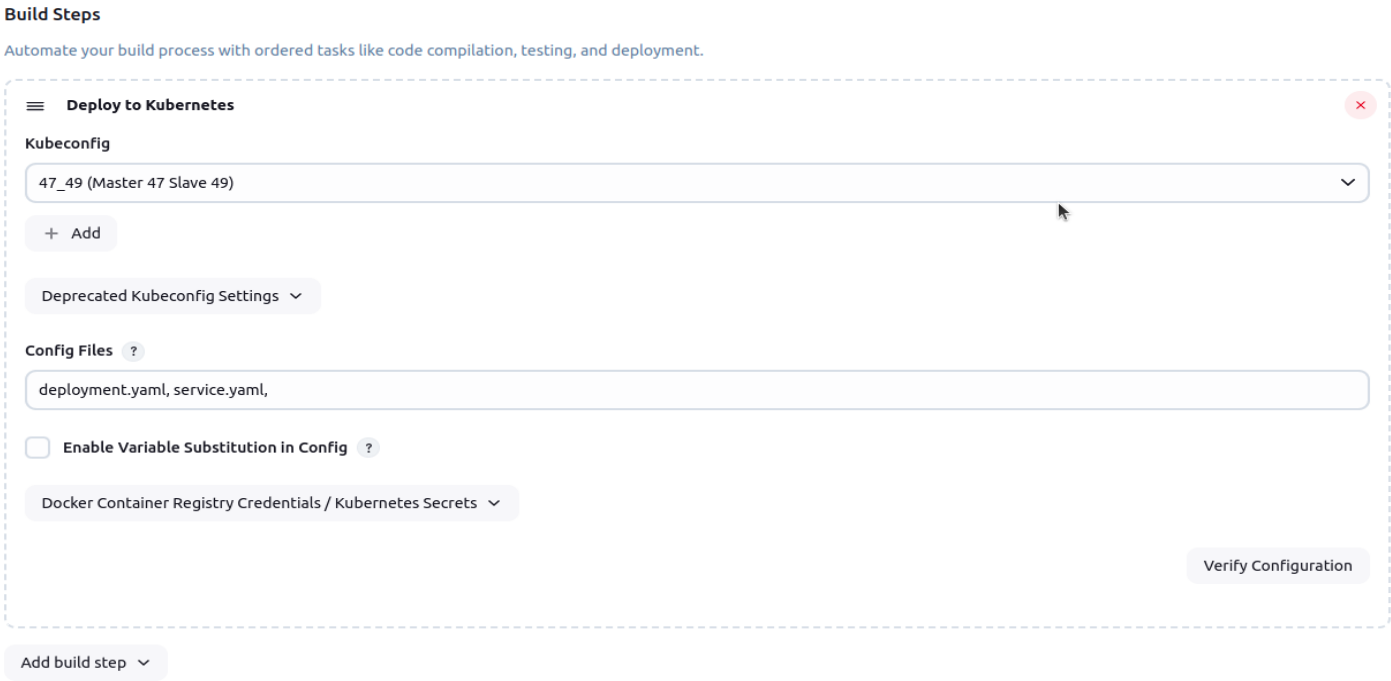
Lists the Kubernetes YAML files to apply:

deployment.yaml, service.yaml

These files define how your app will run and be exposed in the cluster.

Docker Container Registry Credentials / Kubernetes Secrets:

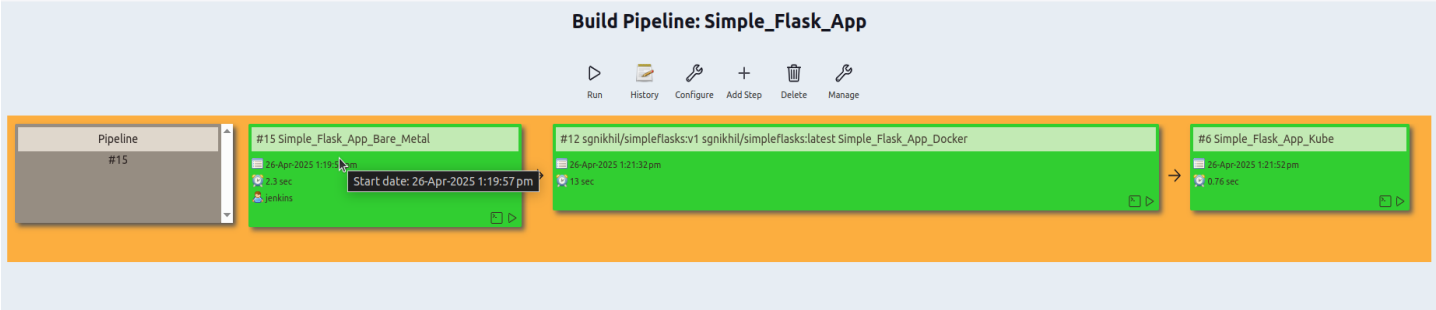
Let Jenkins use stored credentials if your deployment needs to pull images from a private Docker registry.



Simple_Flask_App CI-CD Pipeline

Prerequisites

- Jenkins server is up and running
- Required plugins installed: Git, Python, Docker, Kubernetes
- Docker and Kubernetes cluster access



Pipeline Steps Overview

- Bare Metal Deploy
- Docker Build and Push, Run
- Kubernetes Deploy

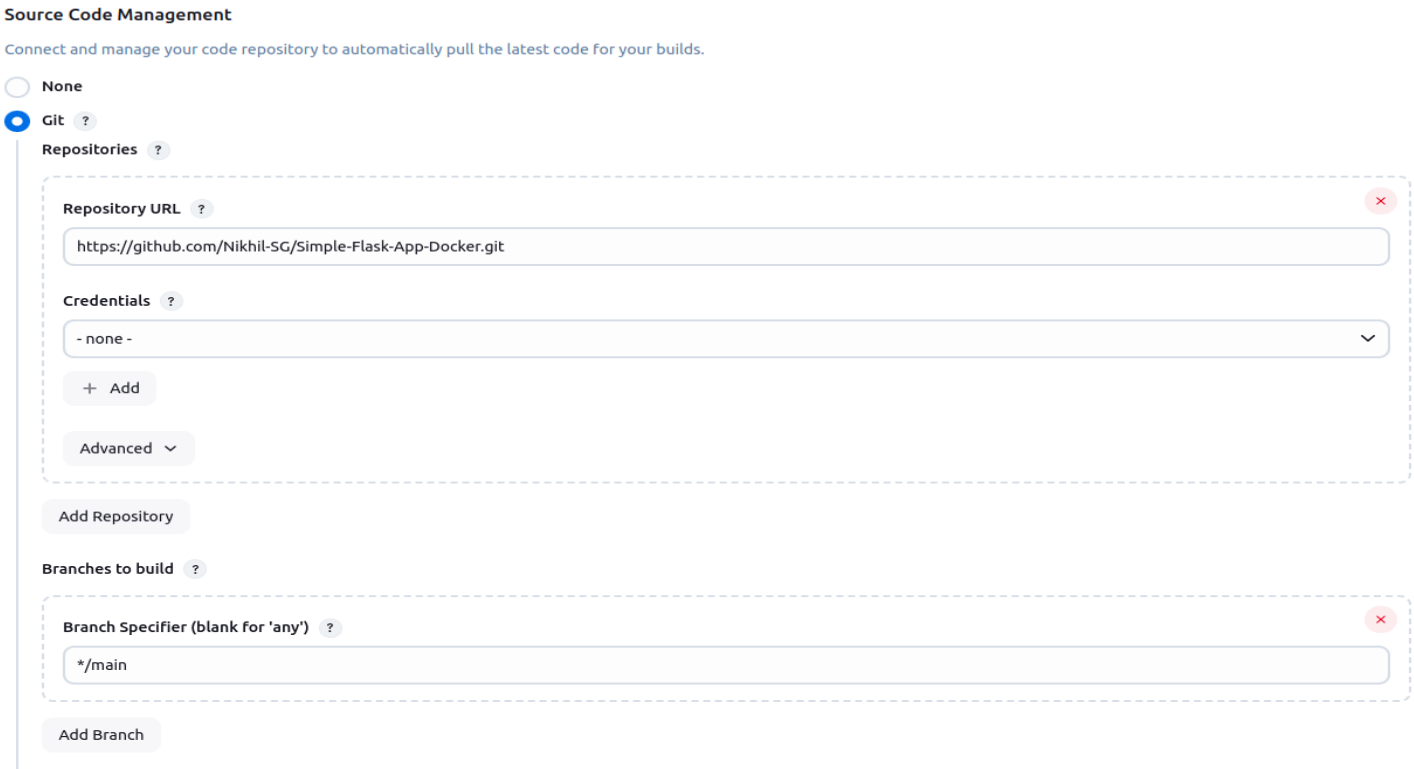
Simple_Flask_App_Bare_Metal

Source Code Management

Select Git.

Enter the repo URL: https://github.com/Nikhil-SG/Simple-Flask-App-Docker.git

Set branch to build: */main.



Build Steps:

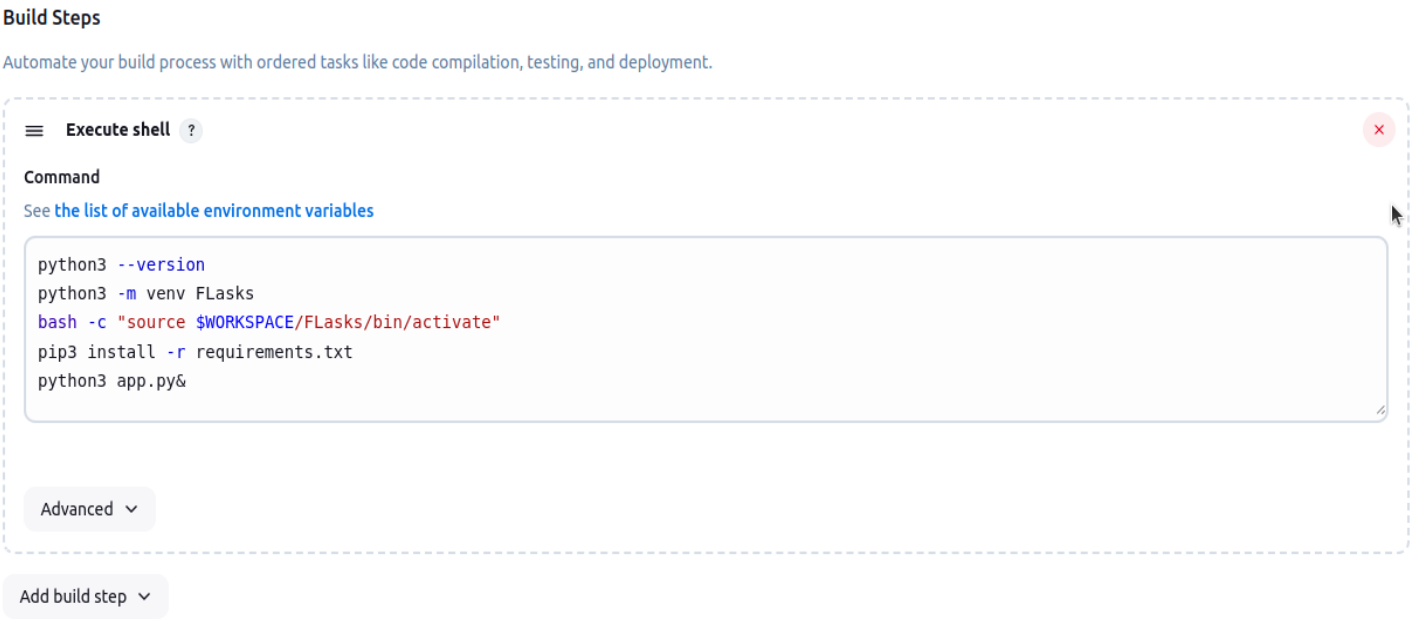
Checks Python version.

Creates a virtual environment called FLasks.

Activates the virtual environment.

Installs all dependencies from requirements.txt.

Starts the Flask app in the background with python3 app.py&

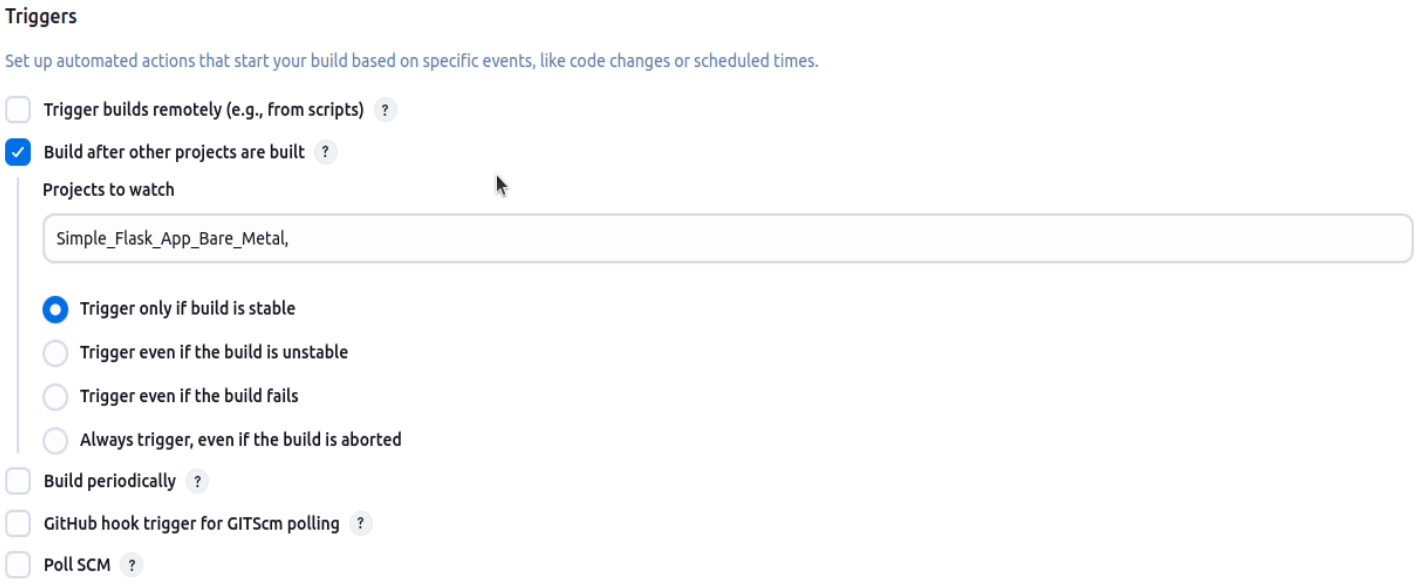


Jenkins downloads your Flask app code, sets up Python, installs needed packages, and runs the app directly on the server.

Simple_Flask_App_Docker

Build Triggers

This job runs automatically after the Simple_Flask_App_Bare_Metal job finishes successfully



Build Steps:

Docker Build and Publish

Builds a Docker image using the Dockerfile for gitrepo.

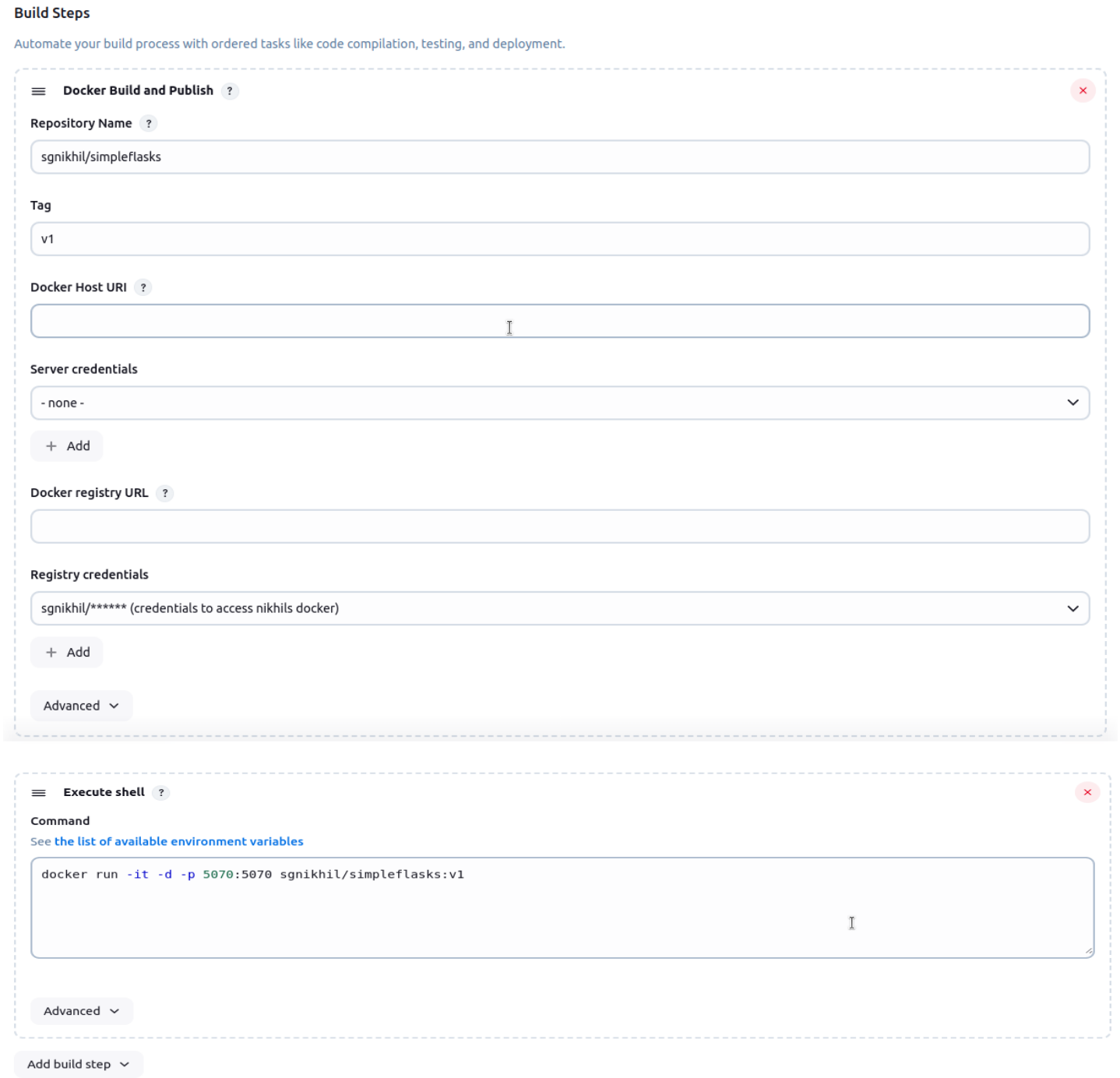
Repository name: sgnikhil/simpleflasks

Tag: v1

Uses Docker credentials to push the image to Docker Hub.

Execute Shell

Runs a shell command to start a container from the new image

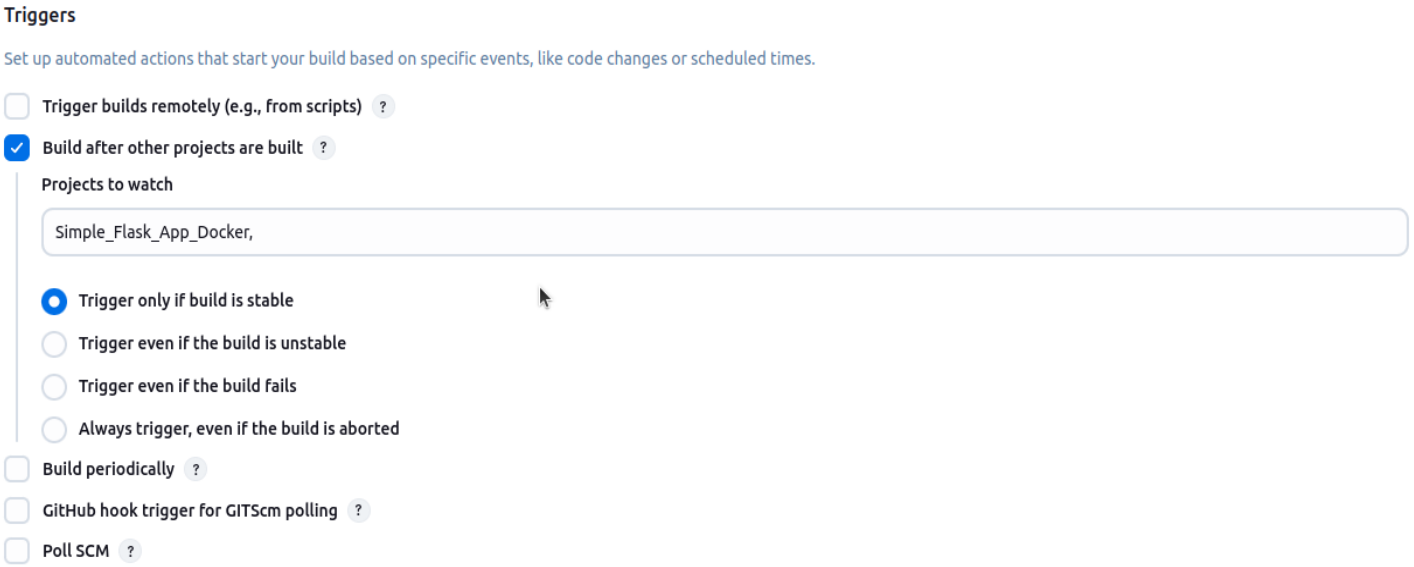


This job starts your Flask app in a Docker container on port 5070, right after the Bare Metal job finishes successfully.

Simple_Flask_App_Kube

Build Triggers

This job runs automatically after the Simple_Flask_App_Docker job finishes successfully



Build Steps:

Deploy to Kubernetes Build Step

Kubeconfig:

Select the Kubernetes cluster credentials (here, "**47_49 (Master 47 Slave 49)**") so Jenkins can connect to your Kubernetes cluster.

Config Files:

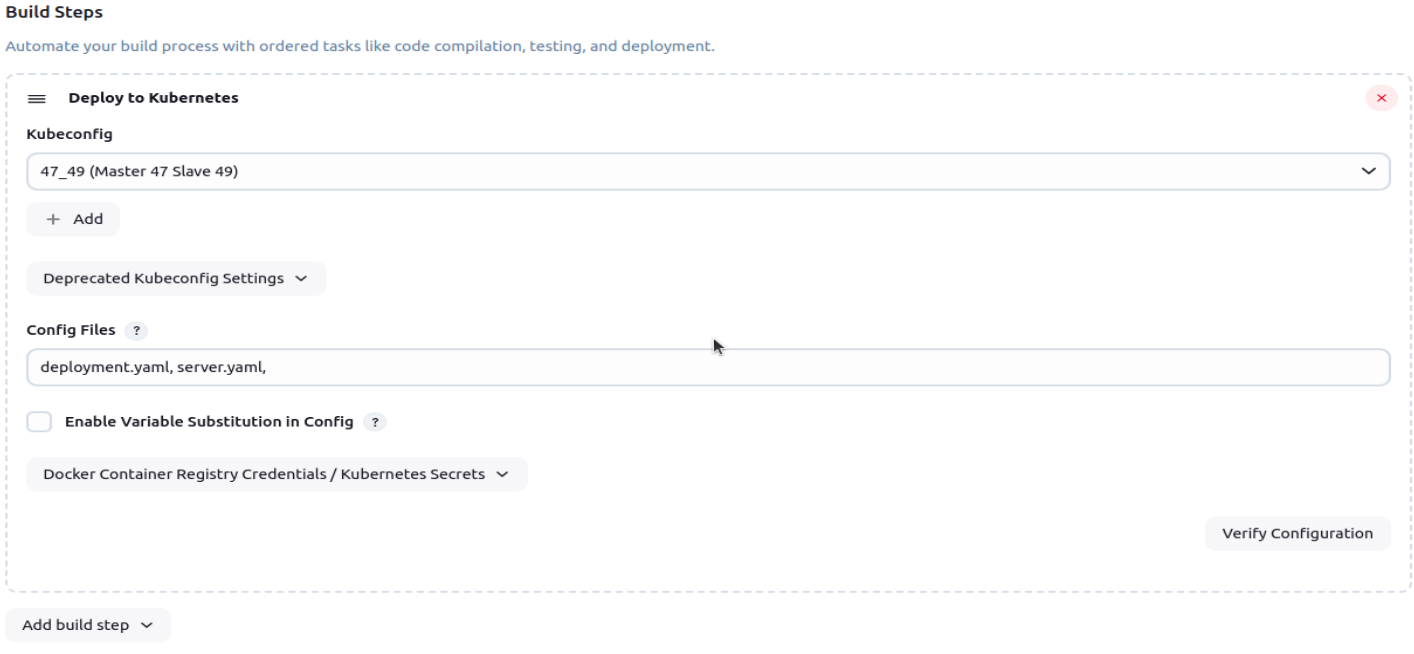
Lists the Kubernetes YAML files to apply:

deployment.yaml, service.yaml

These files define how your app will run and be exposed in the cluster.

Docker Container Registry Credentials / Kubernetes Secrets:

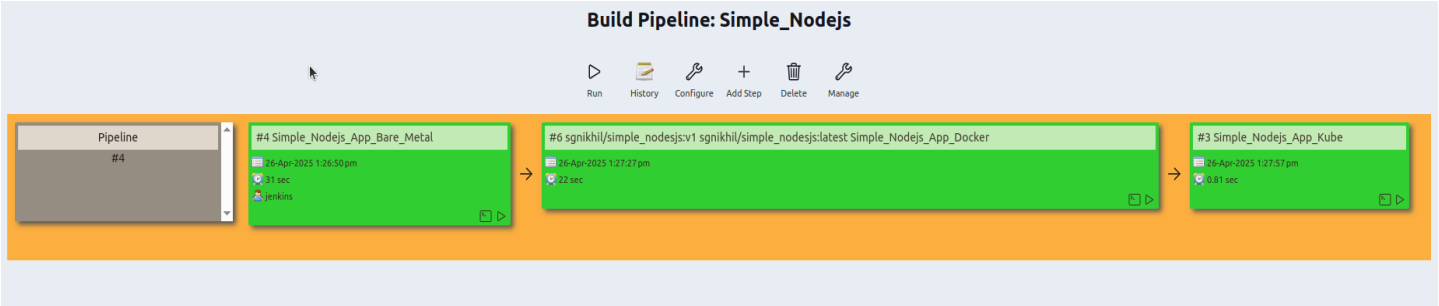
Let Jenkins use stored credentials if your deployment needs to pull images from a private Docker registry.



Simple_Nodejs CI-CD Pipeline

Prerequisites

- Jenkins server is up and running
- Required plugins installed: Git, Node.js, npm, Docker, Kubernetes
- Docker and Kubernetes cluster access



Pipeline Steps Overview

- Bare Metal Deploy
- Docker Build and Push, Run
- Kubernetes Deploy

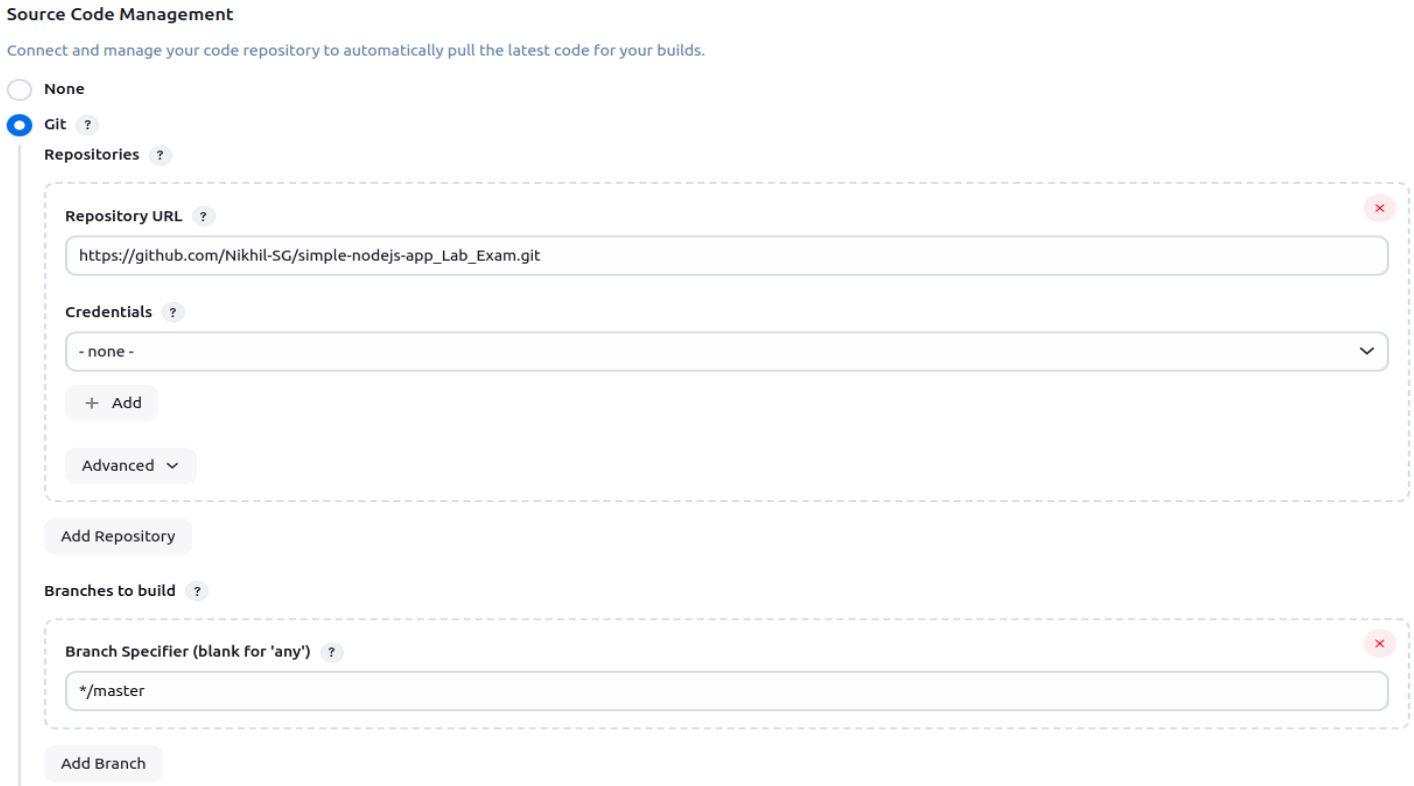
Simple_Nodejs_App_Bare_Metal

Source Code Management

Select Git.

Enter the repo URL: https://github.com/Nikhil-SG/Simple-Flask-App-Docker.git

Set branch to build: */master.



Build Steps:

Execute Shell

Updates the server's package list.

Removes any old Node.js and npm versions.

Installs Node.js 18.x (latest stable version).

Installs all Node.js dependencies with npm install.

Installs pm2 globally (a Node.js process manager).

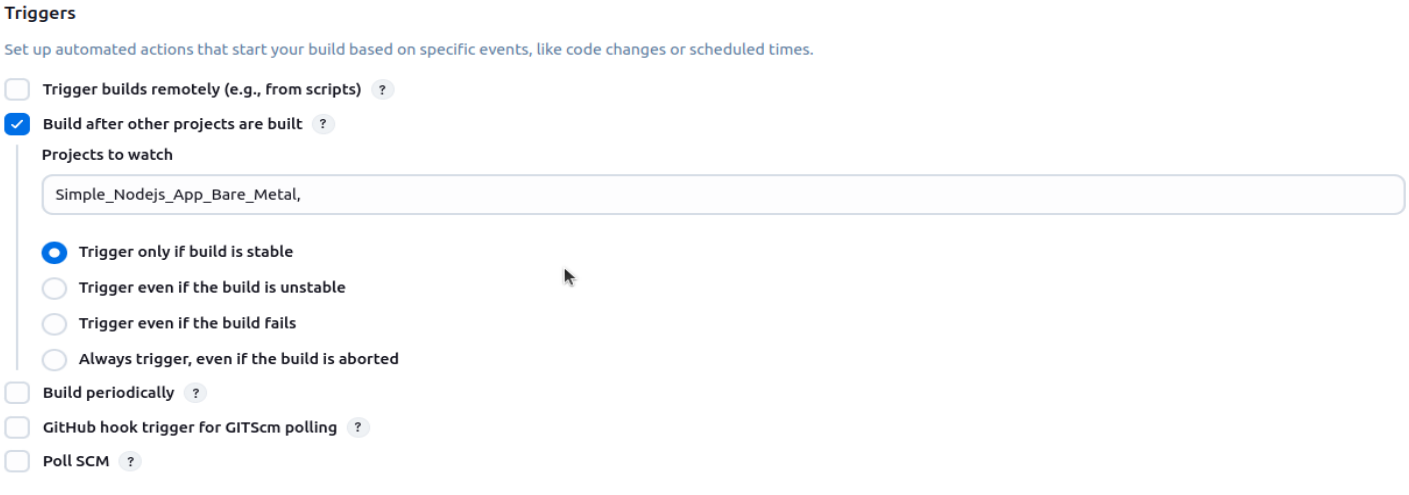
Restarts the app using pm2 with index.js as the entry point

Jenkins downloads your Node.js app, sets up the latest Node.js, installs dependencies, and runs the app using pm2 for better process management.

Simple_Nodejs_App_Docker

Build Triggers

This job runs automatically after the Simple_Nodejs_App_Bare_Metal job finishes successfully



Build Steps:

Docker Build and Publish

Builds a Docker image using the Dockerfile for gitrepo.

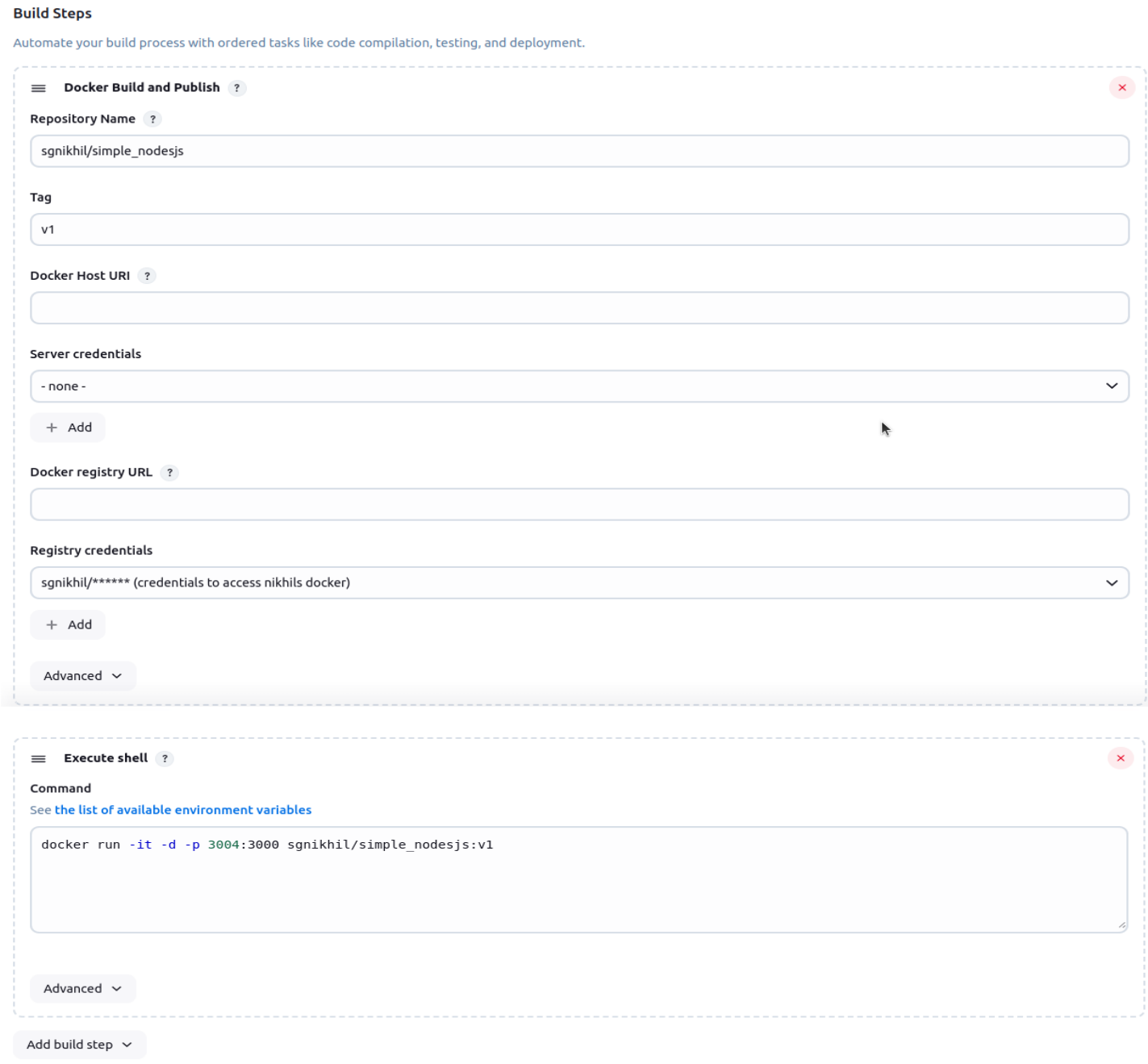
Repository name: sgnikhil/simple_nodejs

Tag: v1

Uses Docker credentials to push the image to Docker Hub.

Execute Shell

Runs a shell command to start a container from the new image



This job starts your Node.js application in a Docker container on port 5070, right after the Bare Metal job finishes successfully.

Simple_Nodejs_App_Kube

Build Steps:

Deploy to Kubernetes Build Step

Kubeconfig:

Select the Kubernetes cluster credentials (here, "**47_49 (Master 47 Slave 49)**") so Jenkins can connect to your Kubernetes cluster.

Config Files:

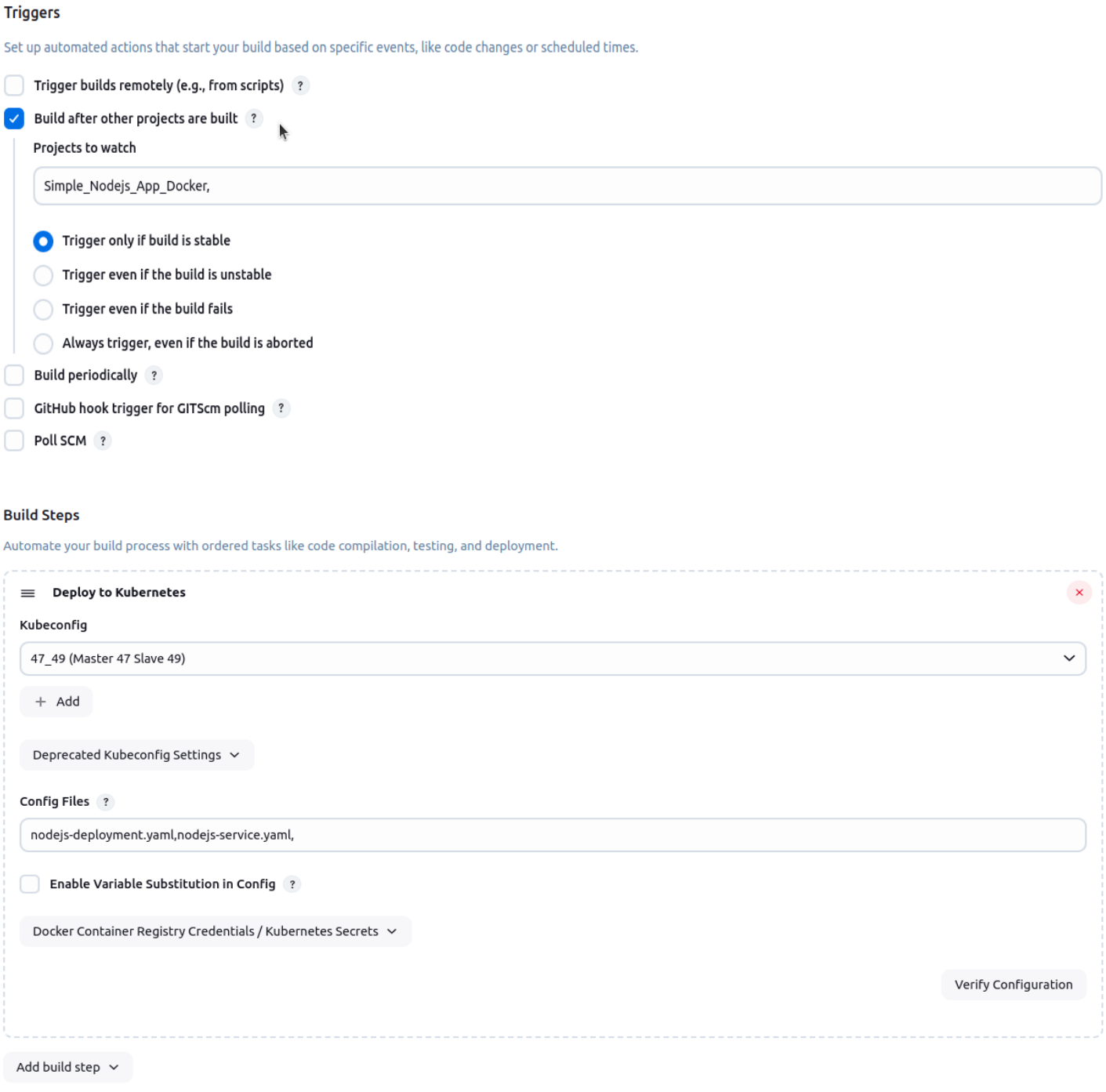
Lists the Kubernetes YAML files to apply:

deployment.yaml, service.yaml

These files define how your app will run and be exposed in the cluster.

Docker Container Registry Credentials / Kubernetes Secrets:

Let Jenkins use stored credentials if your deployment needs to pull images from a private Docker registry.



Bookalbum_php_CICD_Manual_Pipeline

Deploying on a Local Machine (Bare Metal)

Clone the Repository:

Download the PHP website project from GitHub to your local machine.

```
msis@msisBDA87:~$ git clone https://github.com/Nikhil-SG/basic-php-website.git
Cloning into 'basic-php-website'...
remote: Enumerating objects: 107, done.
remote: Counting objects: 100% (35/35), done.
remote: Compressing objects: 100% (15/15), done.
remote: Total 107 (delta 23), reused 20 (delta 20), pack-reused 72 (from 2)
Receiving objects: 100% (107/107), 439.25 KiB | 2.04 MiB/s, done.
Resolving deltas: 100% (41/41), done.
```

View Files:

Use ls to list files and folders.

Prepare Web Directory:

Create a new directory under /var/www/html/ for your project using sudo mkdir.

Copy Project Files:

Copy all files from basic-php-website to your new directory in /var/www/html/ using sudo cp.

```
msis@msisBDA87:~$ ls
 agent.jar          cafe-static-website   db-sample-schemas   Downloads        kmr_scripting                Music        Public                      Templates
 a.py               CMD                   Desktop             jenkins_85       MLBD                         php-apache2  sales_history_backup.zip    'Untitled Document 1'
 basic-php-website  CMD1                  Documents           jenkins_plugin   mongodb_backup_2025-04-03    Pictures     snap                        Videos
msis@msisBDA87:~$ cd basic-php-website/
msis@msisBDA87:~/basic-php-website$ ls
catalog.php  css  deployment.yaml  details.php  Dockerfile  img  inc  index.php  README.md  service.yaml  suggest.php
msis@msisBDA87:~/basic-php-website$ sudo mkdir /var/www/html/basic_php
[sudo] password for msis:
msis@msisBDA87:~/basic-php-website$ sudo cp -rf * /var/www/html/basic_php/
msis@msisBDA87:~/basic-php-website$ cd /var/www/html/basic_php/
msis@msisBDA87:/var/www/html/basic_php$ ls
catalog.php  css  deployment.yaml  details.php  Dockerfile  img  inc  index.php  README.md  service.yaml  suggest.php
```

Configure Apache:

Open the Apache config file (/etc/apache2/sites-available/000-default.conf) with sudo nano and update the DocumentRoot to point to your new project folder.

```
msis@msisBDA87:/var/www/html/basic_php$ sudo nano /etc/apache2/sites-available/000-default.conf
```

```
ServerAdmin webmaster@localhost
DocumentRoot /var/www/html/basic_php
```

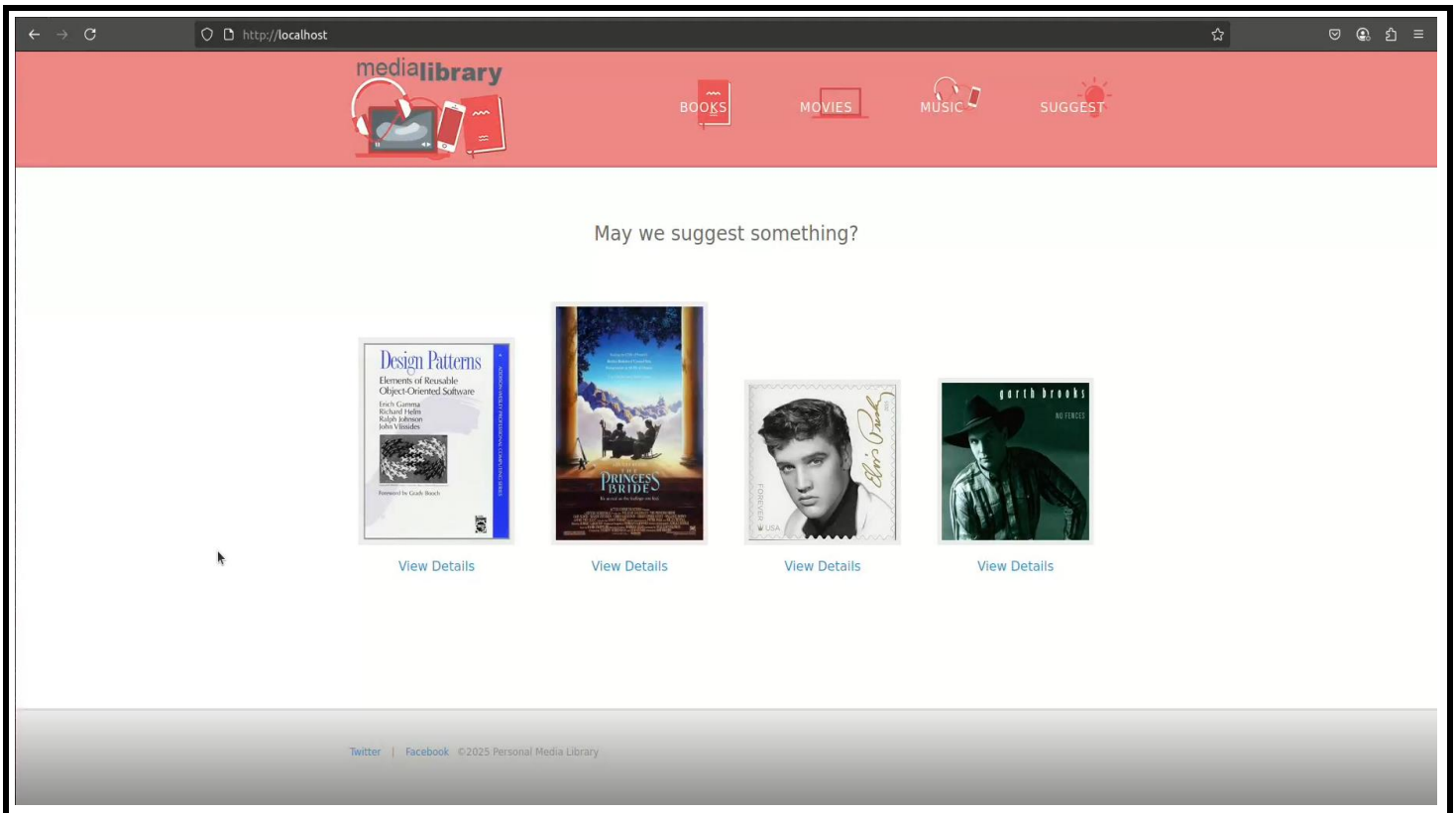
Enable Site & Restart Apache:

Enable the site with a2ensite and restart Apache with sudo systemctl restart

```
msis@msisBDA87:/var/www/html/basic_php$ sudo a2ensite 000-default.conf
Site 000-default already enabled
msis@msisBDA87:/var/www/html/basic_php$ sudo systemctl restart apache2
```

Access the Website:

Open a browser and go to localhost:80 or your system's IP address to view the website.



Deploying with Docker

Log in to Docker:

Authenticate to your Docker account. cd - change directory to basic-php-website folder

Build the Docker Image:

Navigate to the project folder, ensure a Dockerfile exists, and build the image using:

sudo docker build -t your_docker_username/item_name:version .

```
msis@msisBDA87:/var/www/html/basic_php$ cd
msis@msisBDA87:~$ sudo docker login
Authenticating with existing credentials...
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
msis@msisBDA87:~$ cd
msis@msisBDA87:~$ ls
agent.jar           cafe-static-website   db-sample-schemas   Downloads        kmr_scripting              Music        Public                    Templates
a.py                CMD                   Desktop             jenkins_85       MLBD                       php-apache2  sales_history_backup.zip  'Untitled Document 1'
basic-php-website   CMD1                  Documents           jenkins_plugin   mongodb_backup_2025-04-03  Pictures     snap                      Videos
msis@msisBDA87:~$ cd basic-php-website/
msis@msisBDA87:~/basic-php-website$ ls
catalog.php  css  deployment.yaml  details.php  Dockerfile  img  inc  index.php  README.md  service.yaml  suggest.php
msis@msisBDA87:~/basic-php-website$ sudo docker build -t sgnikhil/basic_phpweb:v1 .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/
```

Run the Docker Container:

Start the container with:

sudo docker run -it -d -p host_port:container_port your_docker_username/item_name:version

Push Image to Docker Hub:

Upload your image with:

sudo docker push your_docker_username/item_name:version

```
msis@msisBDA87:~/basic-php-website$ sudo docker run -it -d -p 82:80 sgnikhil/basic_phpweb:v1
db861c64f70f77f654e5e3196113c39bc7b9bc0b02b1f6368eb5fd109f106c7d
msis@msisBDA87:~/basic-php-website$ sudo docker push sgnikhil/basic_phpweb:v1
```

Access the Website:

In your browser, go to localhost:host_port or your_ip:host_port.

Deploying on Kubernetes

Access Kubernetes Master:

Log in and ensure all nodes are ready with kubectl get nodes.

Clone the Project:

Download your repository and check for necessary files (deployment.yaml, service.yaml).

```
msis@msisBDA87:~/basic-php-website$ cd
msis@msisBDA87:~$ sudo ssh msis@172.16.50.47
[sudo] password for msis:
msis@172.16.50.47's password:
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.8.0-51-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled.

178 updates can be applied immediately.
138 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

10 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

New release '24.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Sat Apr 26 16:37:01 2025 from 172.16.50.87
msis@msis:~$ kubectl get nodes
NAME                STATUS   ROLES           AGE    VERSION
msis.kubemaster.com   Ready    control-plane   101d   v1.30.8
msis.kubenode1.com    Ready    <none>          101d   v1.30.11
msis@msis:~$ git clone https://github.com/Nikhil-SG/basic-php-website.git
fatal: destination path 'basic-php-website' already exists and is not an empty directory.
msis@msis:~$ ls
airflow                custom-resources.yaml     Downloads                          kubelet.conf                          Pictures                          test.txt
basic-php-website      custom-resources.yaml.1   google-chrome-stable_current_amd64.deb  kubernetes_manifest_yml_files         'Progressive deployment(2).docx'  untitled.png
cafestaticbda          custom-resources.yaml.2   ingress.yaml                       Machine-Learning-Deployment-using-Docker  Public                        Videos
cafe-static-website    Desktop                   istio-1.24.2                       microservices-demo                    snap
components.yaml        Documents                 java-tomcat-maven-example          Music                                 Templates
msis@msis:~$ cd basic-php-website/
msis@msis:~/basic-php-website$ ls
catalog.php  css  deployment.yaml  details.php  Dockerfile  img  inc  index.php  README.md  service.yaml  suggest.php
```

Update Kubernetes Manifests:

Edit deployment.yaml and service.yaml to confirm the correct Docker image and container port.

```
msis@msis:~/basic-php-website$ sudo nano deployment.yaml
```

```
msis@msis:~/basic-php-website$ sudo nano service.yaml
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: bookalbum
  labels:
    app: bookalbum
spec:
  replicas: 1
  selector:
    matchLabels:
      app: bookalbum
  template:
    metadata:
      labels:
        app: bookalbum
    spec:
      containers:
      - name: bookalbum
        image: sgnikhil/basic_phpweb:v1
        ports:
        - containerPort: 80
```

```yaml
apiVersion: v1
kind: Service
metadata:
  labels:
    app: bookalbum
  name: bookalbum
spec:
  type: NodePort
  ports:
    - port: 80
      targetPort: 80
      protocol: TCP
  selector:
    app: bookalbum
```

Deploy to Cluster:

Apply the manifests using:

kubectl apply -f deployment.yaml kubectl apply -f service.yaml

Verify Deployment:

Check pods: kubectl get pods -o wide, Check service and port: kubectl get svc -o wide

```
msis@msis:~/basic-php-website$ kubectl apply -f deployment.yaml
deployment.apps/bookalbum configured
msis@msis:~/basic-php-website$ kubectl apply -f service.yaml
service/bookalbum unchanged
msis@msis:~/basic-php-website$ kubectl get pods
NAME                                READY   STATUS              RESTARTS   AGE
alertmanager-8bc75d9dd-8nwgc        0/1     ContainerCreating   0          4h44m
bookalbum-5fc666c98c-94qd8          1/1     Running             0          15s
cafestatic-6d5ccc5659-5xt76         1/1     Running             0          12m
cafeweb-6f94686c4d-7xnjw            1/1     Running             0          4h44m
java-tomcat-app-59c4b847df-rhkb5    1/1     Running             0          46m
mlapp-5fffdf5ff6-g89wp              1/1     Running             0          4h44m
mysql-7d4fc6574d-ts2rl              1/1     Running             0          4h44m
nodejs-deployment-6dfdfb747-8fxrq   1/1     Running             0          4h44m
nodejs-deployment-6dfdfb747-xhzzz   1/1     Running             0          4h44m
pythonweb-8658c6696d-78g48          1/1     Running             0          4h44m
simple-nodejs-app-f5dd4d56-vkr9n    1/1     Running             0          4h44m
msis@msis:~/basic-php-website$ kubectl get pods -o wide
NAME                                READY   STATUS              RESTARTS   AGE     IP              NODE                  NOMINATED NODE   READINESS GATES
alertmanager-8bc75d9dd-8nwgc        0/1     ContainerCreating   0          4h45m   <none>          msis.kubenode1.com    <none>           <none>
bookalbum-5fc666c98c-94qd8          1/1     Running             0          27s     192.168.227.58  msis.kubenode1.com    <none>           <none>
cafestatic-6d5ccc5659-5xt76         1/1     Running             0          12m     192.168.227.54  msis.kubenode1.com    <none>           <none>
cafeweb-6f94686c4d-7xnjw            1/1     Running             0          4h45m   192.168.227.20  msis.kubenode1.com    <none>           <none>
java-tomcat-app-59c4b847df-rhkb5    1/1     Running             0          46m     192.168.227.56  msis.kubenode1.com    <none>           <none>
mlapp-5fffdf5ff6-g89wp              1/1     Running             0          4h45m   192.168.227.36  msis.kubenode1.com    <none>           <none>
mysql-7d4fc6574d-ts2rl              1/1     Running             0          4h45m   192.168.227.37  msis.kubenode1.com    <none>           <none>
nodejs-deployment-6dfdfb747-8fxrq   1/1     Running             0          4h45m   192.168.227.12  msis.kubenode1.com    <none>           <none>
nodejs-deployment-6dfdfb747-xhzzz   1/1     Running             0          4h45m   192.168.227.44  msis.kubenode1.com    <none>           <none>
```

```
msis@msis:~/basic-php-website$ kubectl get svc -o wide
NAME             TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE    SELECTOR
bookalbum        NodePort       10.99.25.202     <none>        80:31547/TCP     17h    app=bookalbum
cafestatic       NodePort       10.103.112.3     <none>        80:30263/TCP     100d   app=cafestatic
cafeweb          NodePort       10.108.119.213   <none>        80:31732/TCP     101d   app=cafeapp,tier=frontend
java-tomcat-app  NodePort       10.98.107.6      <none>        8080:31731/TCP   46m    app=java-tomcat-app
kubernetes       ClusterIP      10.96.0.1        <none>        443/TCP          101d   <none>
mlapp            NodePort       10.99.89.70      <none>        5000:31553/TCP   2d21h  app=mlapp
mysql            ClusterIP      10.111.219.178   <none>        3306/TCP         101d   app=cafeapp
nodejs-service   LoadBalancer   10.111.12.211    <pending>     80:32193/TCP     2d17h  app=nodejs
pythonweb        NodePort       10.106.226.48    <none>        5070:31570/TCP   5h11m  app=pythonweb
simple-nodejs-app NodePort      10.99.87.118     <none>        3000:32673/TCP   2d17h  app=simple-nodejs-app
```

Access the Website:

Open a browser and go to node_ip:service_port to view your PHP website running on the cluster.