Experiment No:

1) Write code for a simple user registration form for an event.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Event Registration</title>
<style>
body {
font-family: Arial, sans-serif; margin: 0; padding: 0;
background-color: #f4f4f9;
}
.container { width: 50%;
margin: 50px auto; background: white; padding: 20px; border- radius: 8px;
box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}
h2 {
text-align: center;
}
.form-group {
margin-bottom: 15px;
}
label {
display: block;
margin-bottom: 5px;
}
input, select { width: 100%; padding: 10px;
border: 1px solid #ccc; border-radius:
5px;
```

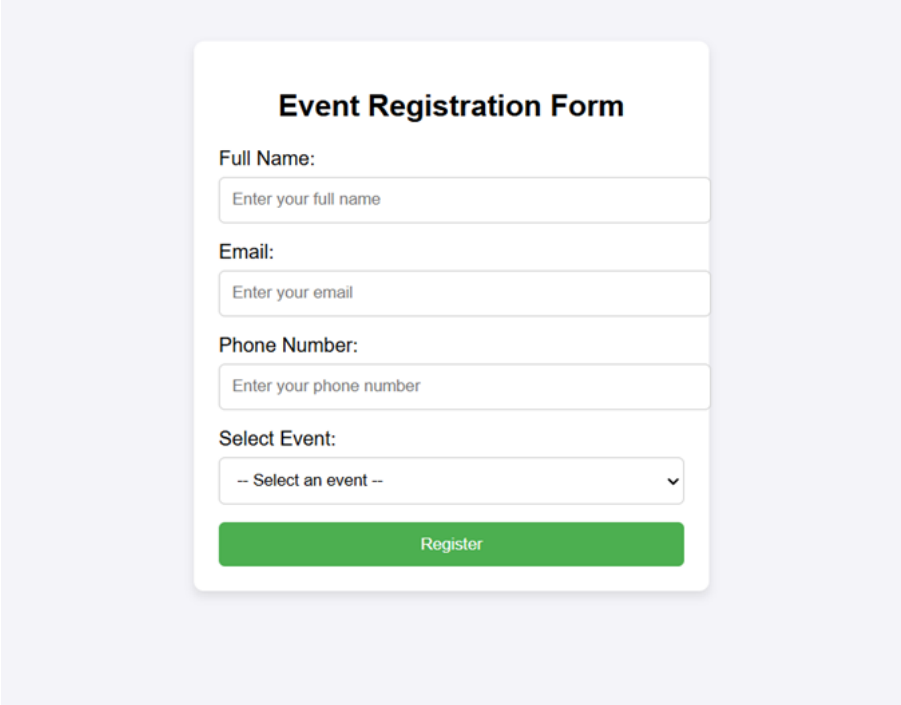
```
}.error {
color: red;
font-size: 0.9em;
}
button {
display: block; width: 100%; padding: 10px;
background-color: #4CAF50; color: white; border: none; border-radius: 5px;
cursor: pointer;
}
button:hover {
background-color: #45a049;
}
</style>
</head>
<body>
<div class="container">
<h2>Event Registration Form</h2>
<form id="registrationForm">
<div class="form-group">
<label for="name">Full Name:</label>
<input type="text" id="name" name="name" placeholder="Enter your full name"
required>
<span class="error" id="nameError"></span>
</div>
<div class="form-group">
<label for="email">Email:</label>
<input type="email" id="email" name="email" placeholder="Enter your email"
required>
<span class="error" id="emailError"></span>
</div>
```

```
<div class="form-group">
<label for="phone">Phone Number:</label>
<input type="tel" id="phone" name="phone" placeholder="Enter your phone number" required>
<span class="error" id="phoneError"></span>
</div>
<div class="form-group">
<label for="event">Select Event:</label>
<select id="event" name="event" required>
<option value="">-- Select an event --</option>
<option value="coding">Coding Workshop</option>
<option value="art">Art Class</option>
<option value="music">Music Concert</option>
</select>
<span class="error" id="eventError"></span>
</div>
<button type="button" onclick="validateForm()">Register</button>
</form>
</div>
<script>
function validateForm() { let isValid = true;
// Get form elements
const name = document.getElementById('name'); const email = document.getElementById('email'); const phone = document.getElementById('phone'); const event = document.getElementById('event');
// Clear previous errors
document.getElementById('nameError').innerText = '';
document.getElementById('emailError').innerText = '';
document.getElementById('phoneError').innerText = '';
document.getElementById('eventError').innerText = '';
// Validate name
```

```
if (name.value.trim() === '') {
document.getElementById('nameError').innerText = 'Name is required.'; isValid = false;
}
// Validate email
if (email.value.trim() === '') {
document.getElementById('emailError').innerText = 'Email is required.'; isValid = false;
} else if (!/\S+@\S+\.\S+/.test(email.value)) {
document.getElementById('emailError').innerText = 'Invalid email format.'; isValid = false;
}
// Validate phone
if (phone.value.trim() === '') {
document.getElementById('phoneError').innerText = 'Phone number is required.'; isValid = false;
} else if (!/^\d{10}$/.test(phone.value)) {
document.getElementById('phoneError').innerText = 'Invalid phone number. Must be 10 digits.'; isValid = false;
}
// Validate event
if (event.value === '') {
document.getElementById('eventError').innerText = 'Please select an event.'; isValid = false;
}
if (isValid) {
alert('Registration successful!'); document.getElementById('registrationForm').reset();
}
}
</script></body></html>
```

Experiment No:

Output:



Experiment No:

2 Explore Git and GitHub commands.

Git

Git is a distributed version control system designed to track changes in source code during software development. It allows multiple developers to work on the same project, keeping track of every modification and enabling collaboration.

Key Features of Git:

1.**Version Control**: Tracks changes to files, allowing you to revisit or revert to earlier versions.

2.**Branching and Merging**: Developers can work on separate branches and later merge their changes.

3.**Distributed System**: Each user has a complete copy of the repository, enabling ofline work.

4.**Efficient Storage:** Git uses a combination of compression and deduplication, making it space- efficient.

5.**Speed:** Git performs operations like commits, diffs, and merges very quickly.

GitHub

GitHub is a web-based platform built on top of Git. It serves as a hosting service for Git repositories, providing additional collaboration tools, project management features, and integrations.

Key Features of GitHub:

1.**Remote Repository Hosting:** Stores your Git repositories in the cloud, making them accessible anywhere.

2.**Collaboration Tools:**

- Pull Requests: A mechanism to review and merge code changes.
- Issues: A system to track bugs, enhancements, and other tasks.

3.Documentation: Markdown support for README files and wikis.

4.Project Management: Built-in tools for kanban boards, milestones, and labels.

5.Social Coding:

- Forking: Copy repositories to your account.
- Starring: Bookmark repositories you like.

6.Integrations: Connects with CI/CD tools, analytics, and other services.

7.Community: A massive ecosystem of open-source projects and contributors.

Experiment No:

Git Basics

1.Initialize a Repository

git init

Initializes a new Git repository in your project directory.

2.Clone a Repository

git clone <repository-url>

Creates a local copy of a remote repository.

3.Check Repository Status

git status

Displays the current status of your working directory (untracked files, changes, etc.).

4.Track a File

git add <file>

git add . # Adds all files in the directory

Stages changes for the next commit.

5.Commit Changes

git commit -m "Commit message"

Saves a snapshot of the staged changes to the repository.

6.View Commit History

git log

Shows the commit history with details like hash, author, and date.

Branch Management

1.Create a New Branch

git branch <branch-name>

Creates a new branch.

2.Switch to a Branch

git checkout <branch-name>

Moves to a specific branch.

3.Create and Switch to a Branch

git checkout -b <branch-name>

Creates and switches to a new branch in one command.

4.Merge Branches

git merge <branch-name>

Merges the specified branch into the current branch.

5.Delete a Branch

git branch -d <branch-name>

Deletes a branch that has been merged.

Remote Repositories

1.Add a Remote

git remote add origin <repository-url>

Links a local repository to a remote repository.

2.View Remotes

git remote -v

Lists the remote repositories linked to the local repository.

3.Push Changes

git push origin <branch-name>

Sends local commits to the remote repository.

4.Pull Changes

git pull origin <branch-name>

Fetches and merges changes from the remote repository into the current branch.

5.Fetch Changes

git fetch origin

Retrieves changes from the remote repository but does not merge them.

Experiment No:

3) Practice Source code management on GitHub. Experiment with the source code in exercise 1.

The sequence of commands for setting up Git, managing a repository, and pushing changes to a remote repository are :

Step 1: Install Git

Run these commands to ensure Git is installed on your system and set up your global Git configuration:

sudo apt update sudo apt install git git --version

sudo apt update: Updates the package list for available upgrades.

sudo apt install git: Installs Git.

git --version: Verifies the installed Git version.

Step 2: Configure Git

Set up your username and email for Git (this information will appear in your commits):

git config --global user.name "your username" git config -- global user.email "your email"

git config --list

•Replace "your username" and "your email" with your actual GitHub username and email.

•git config --list: Confirms the configuration.

Step 3: Create a Folder

1.Check your current directory:

pwd

2.Create a new folder:

mkdir <foldername>

Replace <foldername> with the desired folder name.

3.Navigate into the folder:

cd <foldername>

Step 4: Clone the Repository

1.Clone the remote repository into the folder:

git clone <repo link>

Experiment No:

Replace <repo link> with the URL of the repository you want to clone (e.g., from GitHub).

2.List the contents of the current directory:

ls

3.Navigate to the cloned repository folder:

cd <repo-folder-name>

Step 5. Create the HTML File

1.Create an HTML file for the event registration form:

touch index.html

2.Open the file in a text editor:

gedit index.html

Paste the code of question 1 Save and close the

file.

Step 6. Push Changes to GitHub

1.Check the status of your repository:

git status

You should see index.html as an untracked file.

2.Add the file to the staging area:

git add index.html

3.Commit the changes:

git commit -m "Add event registration form HTML file"

4.Push the changes to the main branch of your remote repository:

git push origin main

Experiment No:

4.Jenkins installation and setup, explore the environment.

Jenkins is an open-source automation server used to build, test, and deploy software. It helps automate parts of the software development lifecycle, making it easier to deliver high- quality software quickly and efficiently.

Key Features of Jenkins:

1.Automation: Automates repetitive tasks like building and testing code.

2.Extensibility: Supports a wide range of plugins for integration with various tools, including Git, Docker, Kubernetes, and Maven.

3.Continuous Integration/Continuous Delivery (CI/CD):

- CI: Integrates code changes into a shared repository multiple times a day.
- CD: Automates the deployment of applications after code changes.

4.Scalability: Supports distributed builds using multiple nodes, enabling parallel execution.

5.Flexibility: Supports different programming languages and build tools.

Installation of jenkins

Step 1: Run the following commands in terminal java version:- *sudo apt install openjdk-21-jdk*

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

iii)*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

Iv)*sudo apt-get update*

v)*sudo apt-get install fontconfig openjdk-17-jre*

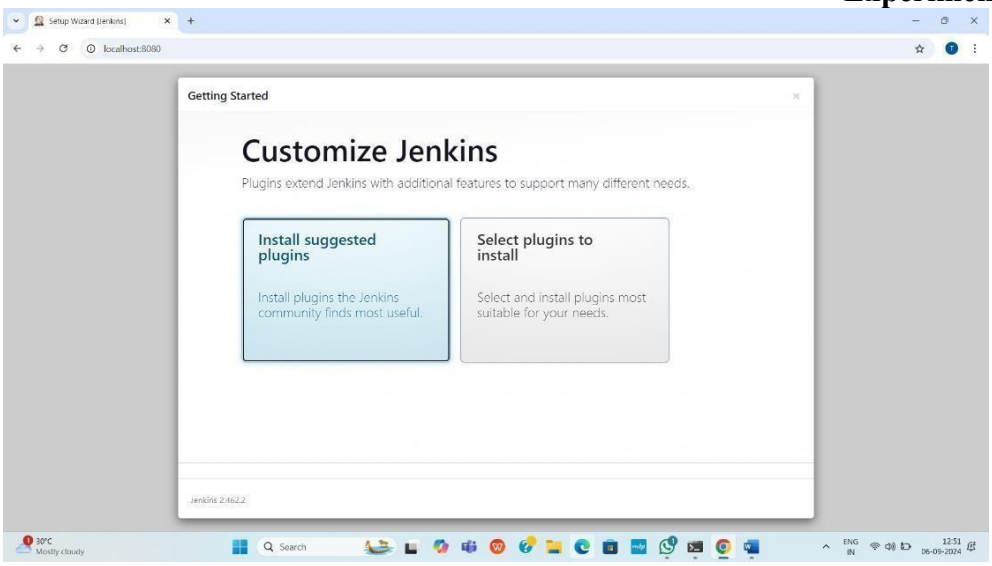
vi)*sudo apt-get install jenkins*

to check jenkins version :-

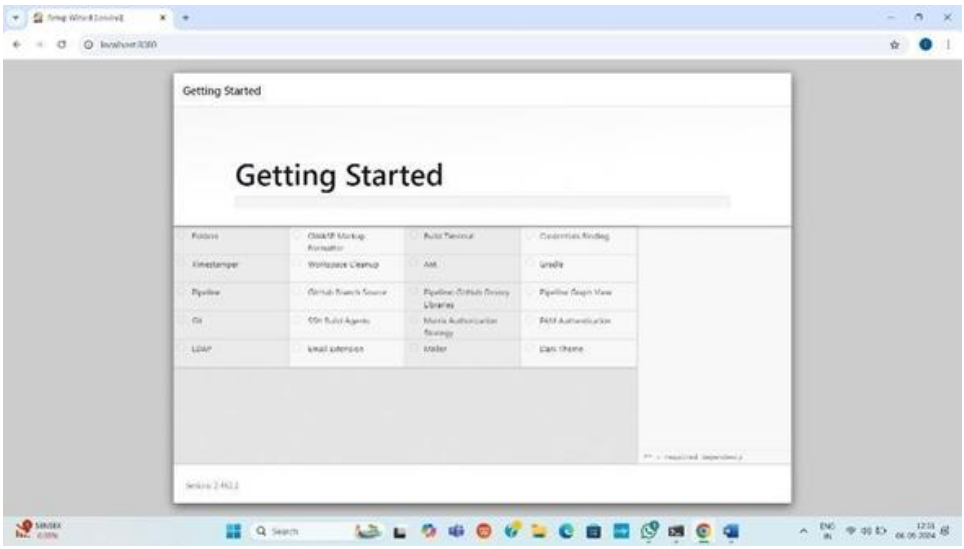
sudo systemctl status jenkins

Step2: Goto browser and type http://localhost:8080

Experiment No:

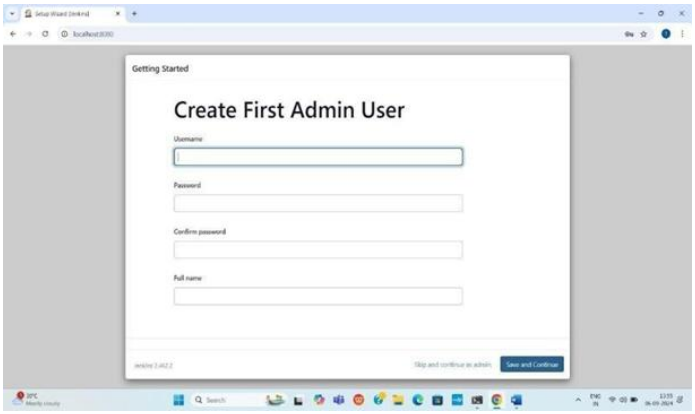


step3: click on install suggested plugins



wait for the green ticks in the above figure

Step4: The below figure will appear where you need click on skip and continue as admin



Experiment No:

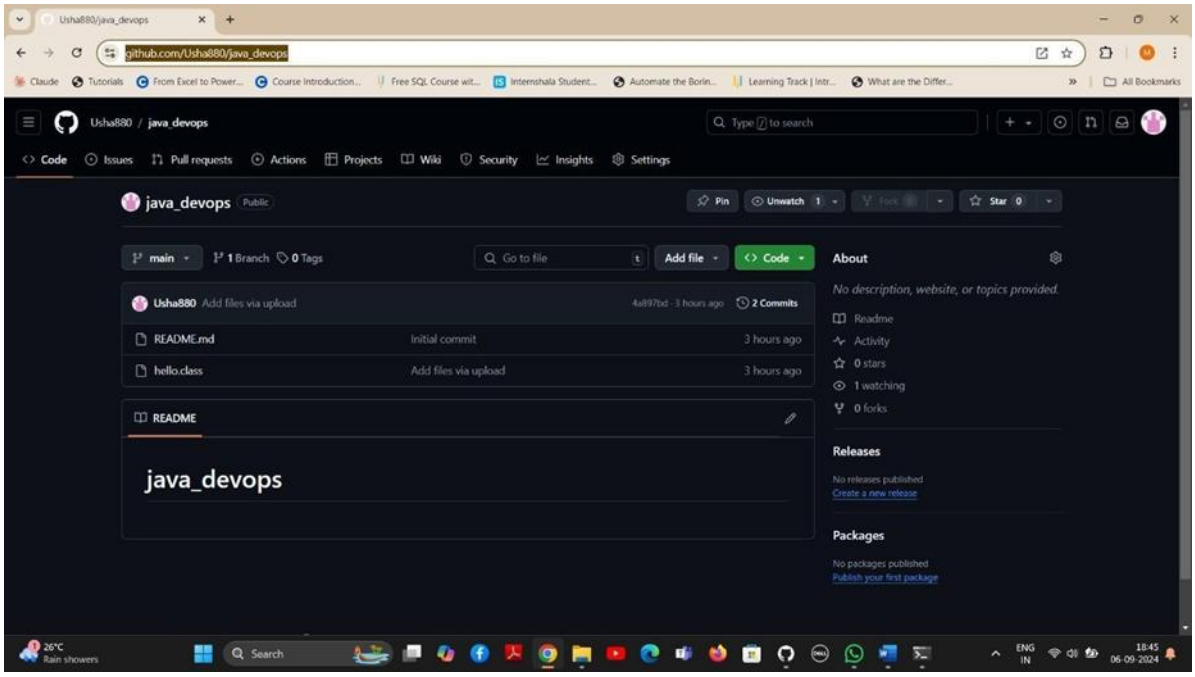
Step 5: Goto Manage jenkins > users > create user . Now create user according to your credentials and save from next time login using the saved credentials.

Note:remember the admin and user credentials

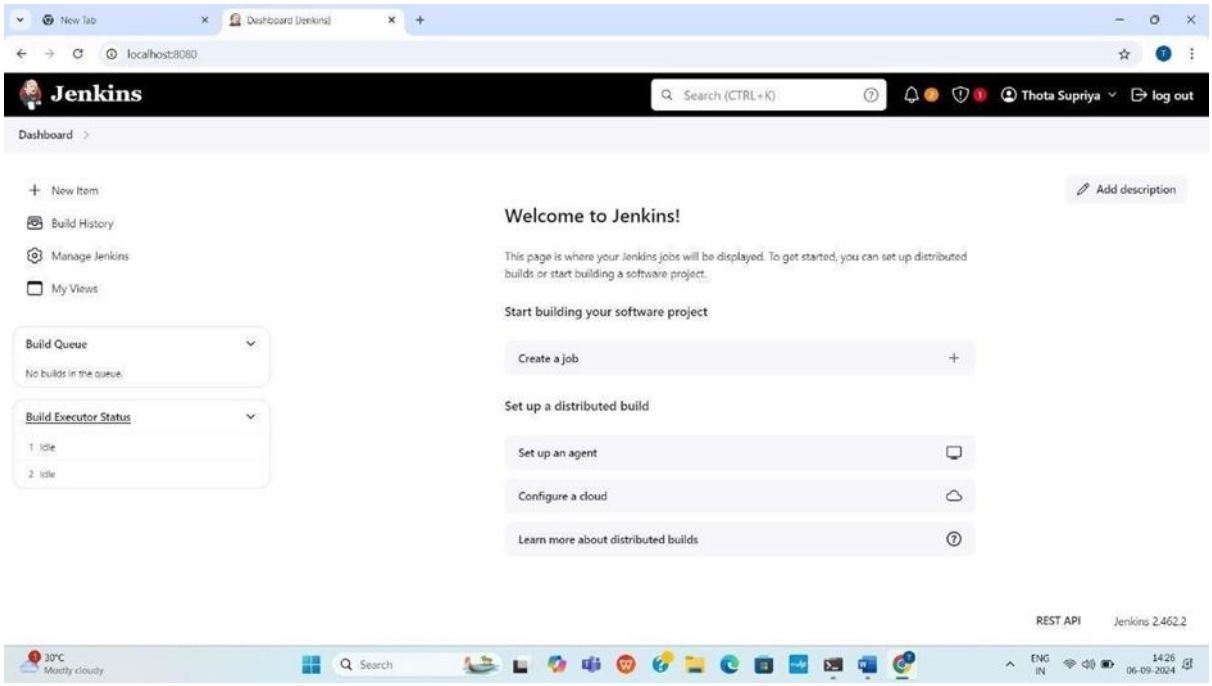
Experiment No:

5.Demonstrate continuous integration and development using Jenkins.

Step 1 :- copy the github url

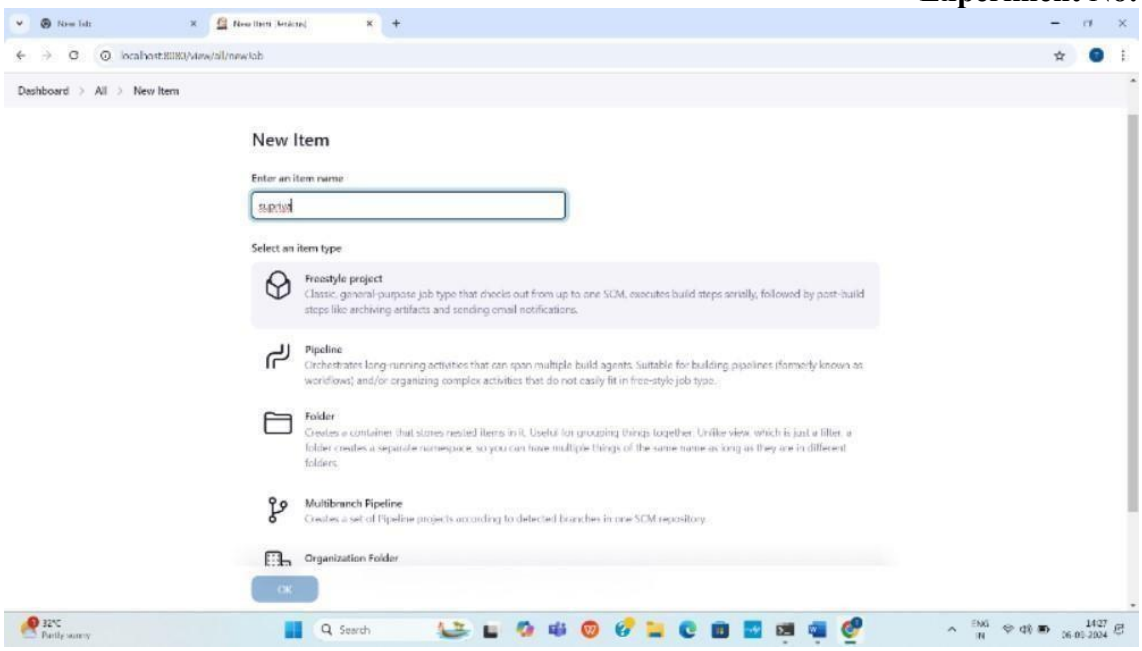


Step 2 :- Go the Google chrome and search for localhost:8080 .The below page will appear



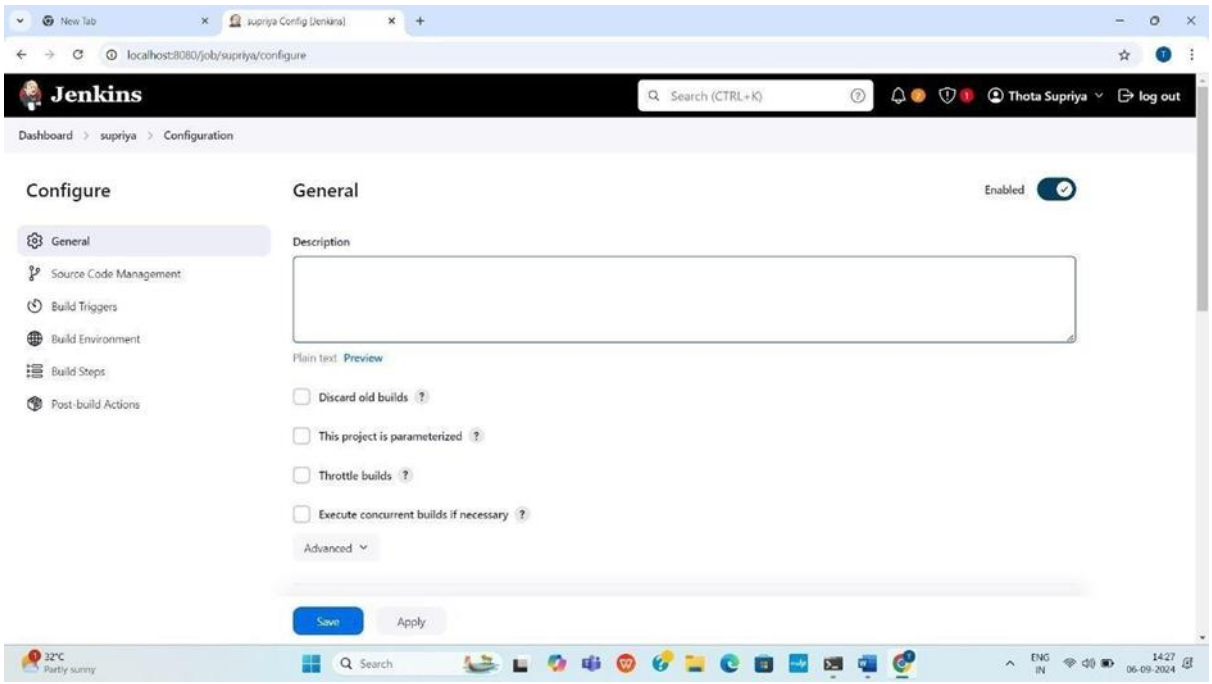
Step 3 :- Select new item and give one name for new item like Mounika (your wish).

Experiment No:



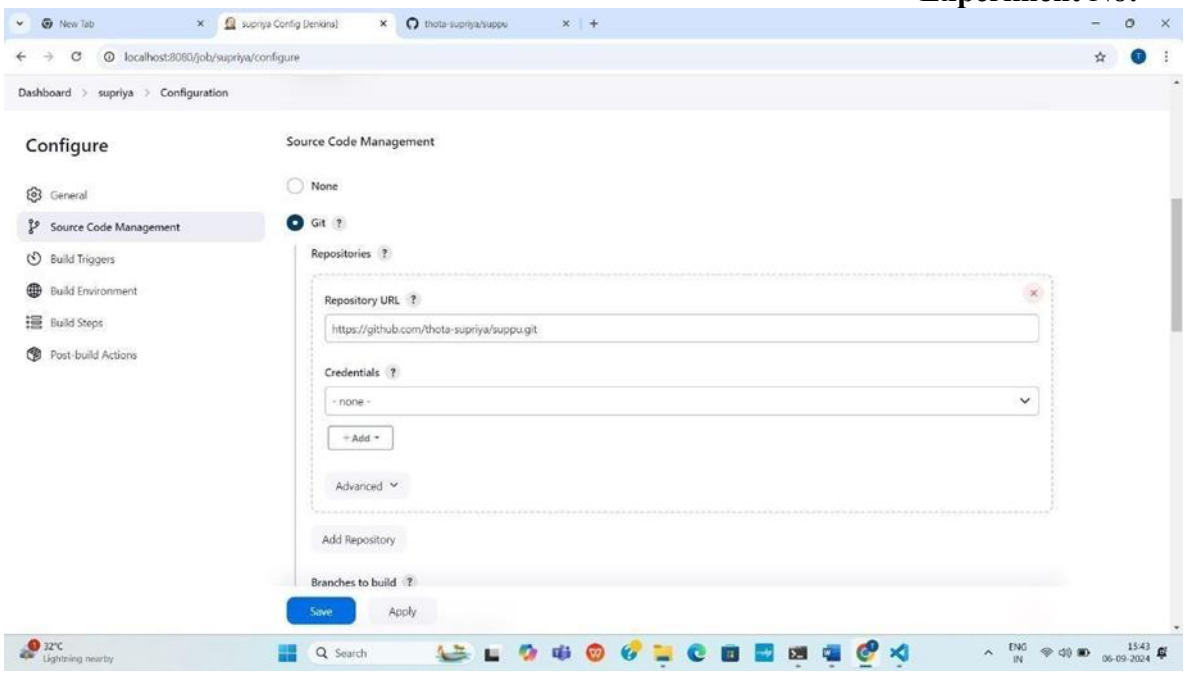
select Freestyle project and click ok.

Next you will see the below figure:

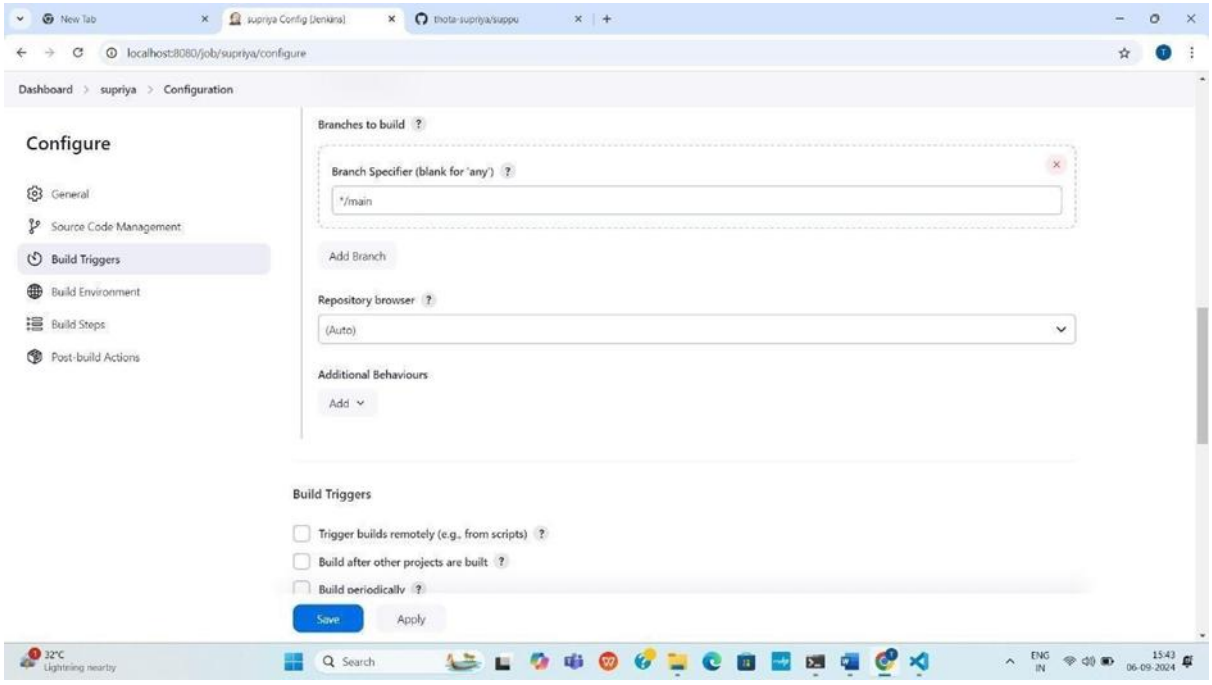


scroll up to see Git and click on it, paste your java git url in the box shown as below:

Experiment No:

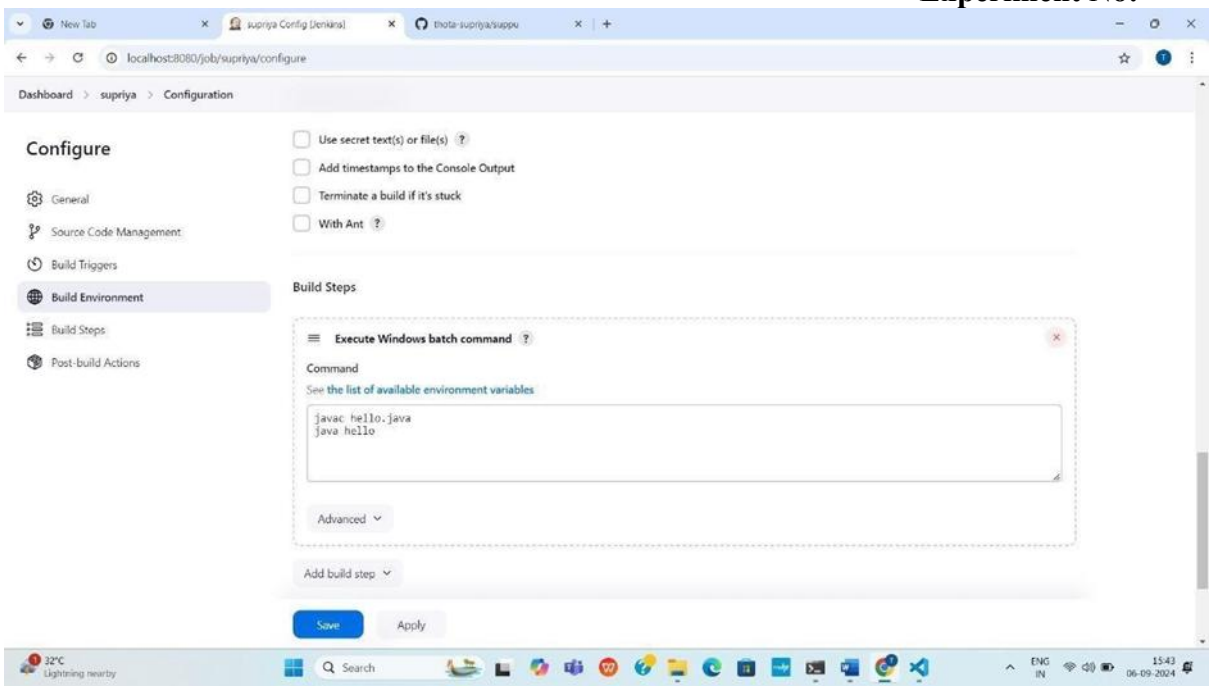


Step 4 :- In branches the name should be */main

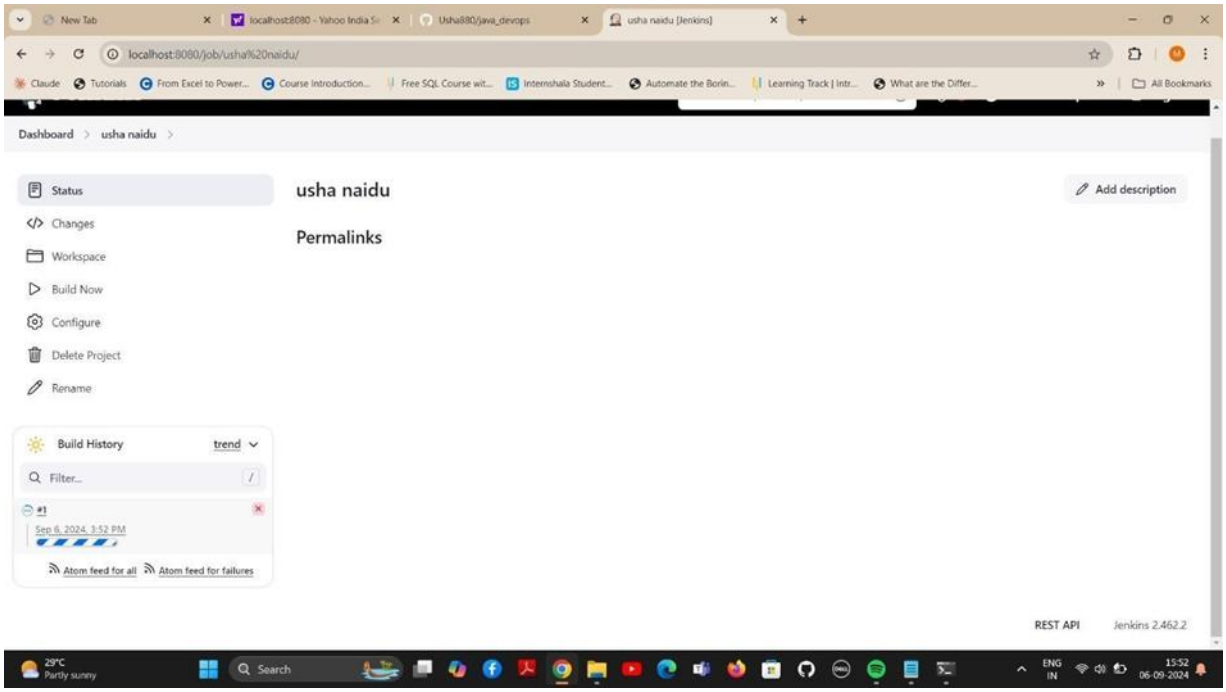


Step 5 :- Go to build steps : select Execute windows batch command ,enter compilation and run commands as shown in below figure:

Experiment No:



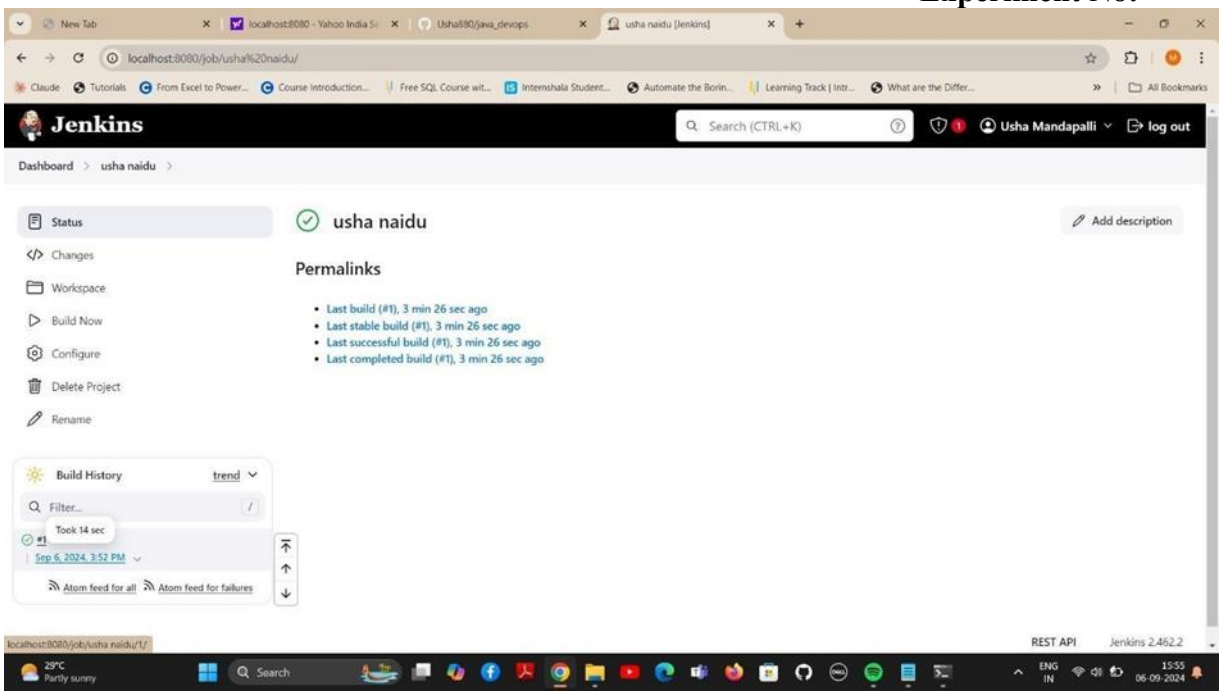
save and the below figure will appear:



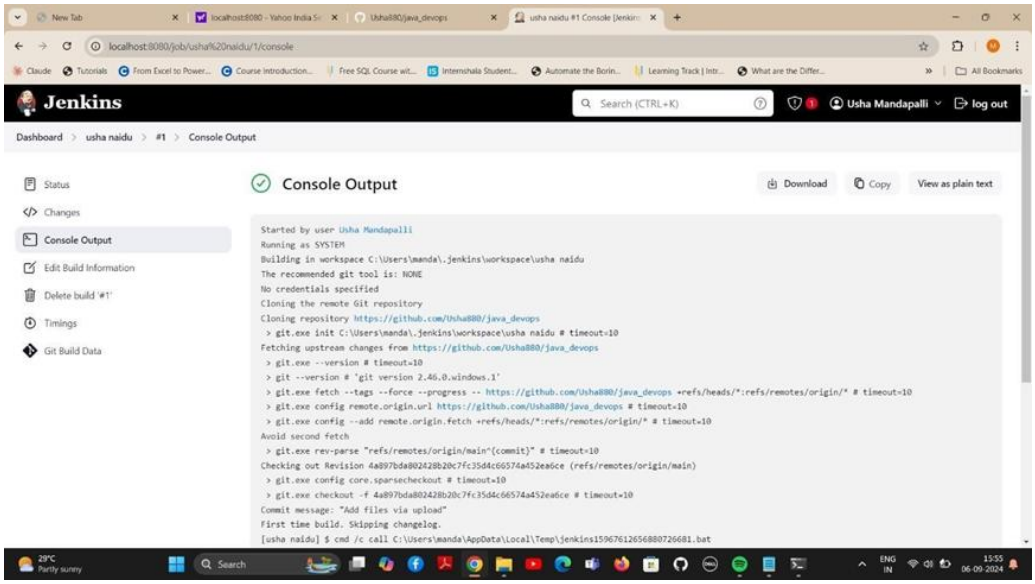
see at the left of the image, your build is setting up to ready (build history)

Step 6 :- click on sep6,2024 current date with time

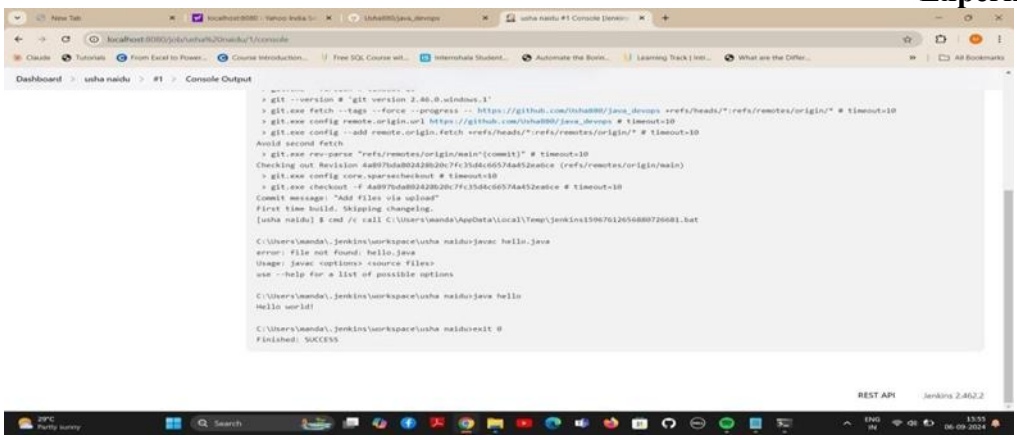
Experiment No:



on left side click on console output and your output will be shown like this:



Experiment No:



The Hello world is displayed.

Experiment No:

6.Explore Docker commands for content management.

Installation of Docker

Step 1*: sudo apt-get install ca-certificates curl*

Step 2: *sudo install -m 0755 -d /etc/apt/keyrings*

Step 3: *sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o*

/etc/apt/keyrings/docker.asc

Step 4: *sudo chmod a+r /etc/apt/keyrings/docker.asc*

Step 5: *echo \ "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \ $(./etc/os-release && echo "$VERSION_CODENAME") stable" | *

Step 6: *sudo tee /etc/apt/sources.list.d/docker.list > /dev/null*

Step 7: *sudo apt-get update*

Step 8*: sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker- compose-plugin*

Docker

Docker is an open-source platform designed to automate the development, deployment, and running of applications using containerization. Containers are lightweight, portable, and self sufficient environments that bundle an application with all its dependencies, libraries, and configurations, ensuring consistency across different computing environments.

Key Concepts of Docker

1.Containers

- Containers are isolated environments that run applications along with their dependencies.
- Unlike virtual machines, containers share the host operating system's kernel, making them lightweight and faster to start.

2.Images

- An image is a blueprint for a container.
- It contains the application code, runtime, libraries, and environment settings.
- You can create an image using a Dockerfile.

3.Dockerfile

- A Dockerfile is a script with a set of instructions to build a Docker image.
- It defines the application, dependencies, and configurations.

4.Docker Engine

- The core of Docker, it is responsible for creating and running containers.
- It consists of:
- Docker Daemon: Runs on the host machine and manages Docker objects.
- Docker CLI: A command-line interface to interact with Docker.

5.Docker Hub

- A public registry where Docker images are stored.
- Developers can pull existing images or push their own images.

Image Management

1.List Images

sudo docker images

2.Pull Images

sudo docker pull <image_name>:<tag>

//tag is used for pulling specific versions, if we need latest version remove tag and it will automatically installs latest version

3.Remove Images

- Remove a specific image:

  sudo docker rmi <image_id>

- Remove unused images:

  sudo docker image prune

4.Inspect a Image

sudo docker inspect <image_name_or_id>

Container Management

1.List Containers

- Shows Running containers:

  sudo docker ps

- Shows All containers (including stopped):

  sudo docker ps -a

2.Start, Stop, and Restart Containers

- Start a stopped container:

  sudo docker start <container_name_or_id>

- Stop a running container:

  sudo docker stop <container_name_or_id>

- Restart a container:

  sudo docker restart <container_name_or_id>

3.Run a Container

sudo docker run -d --name <container_name> <image_name>

- This will pull the image(if necessary),
- create a Container,
- starts the container,
- runs the default command

4.To run a container with specific configurations

sudo docker run --name <container_name> -e MYSQL_ALLOW_EMPTY_PASSWORD=yes -d mysql

- This is an environment variable that is set inside the container.
- MYSQL_ALLOW_EMPTY_PASSWORD=yes tells the MySQL container to allow the root user to have an empty password (i.e., no password).
- Note: Allowing an empty password can be a security risk in production, but it can be useful in development or testing environments.

5.Remove Containers

- Remove a specific container:

  sudo docker rm <container_name_or_id>

- Remove all stopped containers:

  sudo docker container prune

6.Inspect a Container

sudo docker inspect <container_name_or_id>

7.View Logs

sudo docker logs <container_name_or_id>

8.Create a Container without running

sudo docker create <image_name/Image_id> <image>

- Image name will be generated automatically and can read it using docker images
- <image> - like mysql,ngnix

Experiment No:

9.Create a container with custom name

sudo docker create –name <custom_container_name> <image>

- to start

  sudo docker start <custom_container_name>

- to stop

  sudo docker stop <custom_container_name>

- To forcefully kill a container

  sudo docker kill <container_name>

Experiment No:

7.Develop a simple containerized application using Docker.

1.Write the Java Application

Create a file named hi.java with the following content:

public class hi {

public static void main(String[] args) { System.out.println("Hello, World!");

}

}

2.Create the Dockerfile

In the same directory as the hi.java file, create a file named Dockerfile with the following content:

Use the official OpenJDK image as the base image

FROM openjdk:latest # Set the working directory inside the container WORKDIR /app

Copy the Java application into the container

COPY . /app

Compile the Java application

RUN javac hi.java

Command to run the Java application

CMD ["java", "hi"]

3.Build the Docker Image

Run the following command in the terminal (ensure you're in the same directory as the Dockerfile):

sudo docker build -t myimage .

- docker build: Builds a Docker image.
- -t myimage: Tags the image with the name myimage. “.” Specifies the current directory as the build context.

4.Run the Docker Container

Start a new container using the image you just built:

sudo docker run --name mycontainer myimage

- --name mycontainer: Names the container mycontainer. myimage: Specifies the image to use for the container.

Experiment No:

5.Verify the Output

To view the output of the container, use:

sudo docker logs mycontainer

This will display:

Hello, World!

Experiment No:

8.Integrate Kubernetes and Docker

Kubernetes

Kubernetes (often abbreviated as K8s) is an open-source container orchestration platform designed to automate the deployment, scaling, and management of containerized

applications. It helps manage microservices architectures by ensuring that applications run smoothly, scale efficiently, and are highly available.

Tools used:

minikube: Cluster setup & management tool for Kubernetes.

It is a tool that allows you to run a local Kubernetes cluster on your personal computer. It creates a single node Kubernetes cluster inside a docker or container on your local system,

making it an excellent option for developing,testing and learning kubernetes in a lightweight, isolated environment.

kubectl: Command-Line Tool for Kubernetes Management.

kubectl(Kubernetes Control) is the command-line tool used to interact with and manage Kubernetes clusters. It allows users to perform a wide range of tasks, such as creating,

managing, and troubleshooting Kubernetes resources (like pods, deployments, and services) in a Kubernetes cluster.

Key Concepts of Kubernetes:

1.Cluster:

A Kubernetes cluster is the foundation of a Kubernetes environment. It consists of a set of machines (either physical or virtual) that run Kubernetes, and it includes a control plane and one or more worker nodes.

2.Node:

A Node is a physical or virtual machine on which Kubernetes runs your application workloads. There are two types of nodes in Kubernetes:

- Master Node (Control Plane): Manages the cluster and makes global decisions (e.g., scheduling, scaling).
- Worker Node: Runs the application workloads, which include the pod.

3.Pod:

A Pod is the smallest and simplest unit in Kubernetes. It represents a single instance of a running process in a cluster.

4.Deployment:

Experiment No:

A Deployment is a Kubernetes resource used to manage replicas of a pod.

5.Service:

A Service is a Kubernetes resource that provides a stable IP address and DNS name for accessing a set of pods. Eg: NodePort,LoadBalancer,etc.

Installation of Kubernetes:

Step 1: sudo snap install microk8s --classic

Step 2: sudo usermod -a -G microk8s jntuhcse

Step 3: mkdir -p ~/.kube

Step 4: sudo chown -R jntuhcse ~/.kube

Step 5: newgrp microk8s

Step 6: microk8s kubectl get nodes

Step 7: microk8s kubectl get services

Step 8: sudo snap install kubectl --classic

Step 9: to check version

kubectl version --client

Step 10: to install minikube

curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64

sudo install minikube-linux-amd64 /usr/local/bin/minikube

Integration of Kubernetes and Docker.

Minikube commands:

1.Check minikube version:

minikube version

2.Start minikube with custom docker driver:

The below command starts a local Kubernetes cluster using docker as the virtualization driver instead of default virtual machine(VM)driver.

minikube start --driver=docker

3.Set a default driver:

The below command sets the default driver for Minikube to Docker, meaning that when you start a Minikube cluster, it will use Docker containers instead of a virtual machine (e.g., VirtualBox or Hyper-V) to run the Kubernetes cluster.

minikube config set driver docker

4.Status commands:

- Check the Status of Minikube Cluster: This command shows whether your Minikube cluster is running and other details.

  minikube status

- Check the Cluster Info: This command provides general information about the running cluster.

  minikube info

5.Minikube Service Proxy:

Create a Proxy for Minikube Services: To forward HTTP(S) requests to your Minikube services on a specific port

minikube service <service-name> --url [Eg:<service-name>=nginx]

6.Stopping Minikube:

Stop the Minikube Cluster: This will stop your Minikube cluster but keep the docker container.

minikube stop

7.Start minikube:

This command starts a local Kubernetes cluster on your machine.

minikube start

8.Delete Minikube:

Delete the Minikube Cluster: If you no longer need the Minikube cluster, you can delete it, which removes the docker container and all associated data.

minikube delete

9.Minikube Logs:

Get Logs of Minikube Cluster: You can check the logs of Minikube to troubleshoot or monitor the cluster.

minikube logs

kubectl commands:

1.Cluster Information:

View Cluster Info: Displays information about the Kubernetes cluster.

kubectl cluster-info

Experiment No:

2.Pods:

- Get Pods: Lists all the Pods in the current namespace.

  kubectl get pods

- Delete a Pod: Deletes a specific Pod.

  kubectl delete pod <pod-name>

- Describe a Pod: Shows detailed information about a specific Pod, including logs, events, etc.

  kubectl describe pod <pod-name>

3.Deployments:

- Get Deployments: Lists all deployments in the current namespace.

  kubectl get deployments

- Create a deployment:creates a deployment in the cluster.

  kubectl create deployment <deployment-name> --image=<image-name>

Eg:<image-name>=nginx, <deployment-name>=nginx

- Delete a deployment:

  kubectl delete deployment <deployment-name>

4.Services:

- Get Services: Lists all services in the current namespace.

  kubectl get services

- Describe a Service: Shows detailed information about a service.

  kubectl describe service <service-name>

- Expose a Pod or Deployment as a Service: Exposes a deployment or pod as a service.
  kubectl expose deployment <deployment-name> --port=80 --target-port=80 - type=LoadBalancer(or)kubectl expose deployment <deployment-name> --port=80 --target-port=80 - type=NodePort

5.Nodes:

- Get Nodes: Lists all nodes in the Kubernetes cluster.

  kubectl get nodes

- Describe a Node: Shows detailed information about a specific node.

kubectl describe node <node-name>

9.Automate the process of running containerized application for exercise 7 using Kubernetes.

Step 1: Create a Docker file for a java program.

Step 2: Build an image for the dockerfile and run it using docker commands.

Step 3: Docker login

Step 4: Create a new tag for an existing docker image by using the following command.

sudo docker tag <image-name> <docker user-name>/<image-name>

Step 5: Upload the above docker image from your local system to a docker registry using the following command.

sudo docker push <docker user-name>/<image-name>

Step 6: Create a deployment file with yaml extension.

deployment.yaml file: A deployment.yaml file is a Kubernetes configuration file used to define a deployment resource.This file is written in YAML format and contains all the instructions for K8S to create and maintain a group of pods based on the specified template.

//deployment.yaml apiVersion: apps/v1

kind: Deployment metadata:

name: <docker image-name> spec:

replicas: 1 selector:

matchLabels:

app: : <docker image-name> template:

metadata:

labels:

app: : <docker image-name> spec:

containers: : <docker image-name>

- name: : <docker image-name>

image: <docker user-name>/<docker image-name>:latest ports:

- containerPort: 8080

Step 7: Create a service file with yaml extension.

service.yaml: A service.yaml file is a Kubernetes configuration file used to define a service resource. This file is written in YAML format and contains all the details needed to configure the service.

//service.yaml apiVersion: v1 kind: Service metadata:

name: <docker image-name>-service spec:

selector:

app: <docker image-name> ports: - protocol: TCP

port: 80

targetPort: 8080 type: ClusterIP

Step 8:Use the below command to display the current context configured in the Kubernetes kubeconfig file.

kubectl config current-context

Step 9: Start the minikube cluster, which sets up a local K8s environment on your machine.

minikube start

Step 10: Use the below command to apply a Kubernetes configuration file (deployment.yaml) to the cluster.

kubectl apply -f deployment.yaml

Step 11: Use the below command to apply a K8s configuration file (service.yaml) to the cluster.

kubectl apply -f service.yaml

Step 12: The below command is used to retrieve and list information about the pods running in a Kubernetes cluster.

kubectl get pods

Step 13: Use the below command to retrieve and list all the services running in a Kubernetes cluster.

kubectl get services

Step 14: Use the following command to view the logs of a specific pod in a K8s cluster.

kubectl logs <pod-name>

Experiment No:

10) Install and Explore Selenium for Automated Testing

Selenium is an open-source tool used for automating web browser interactions. It allows testers to write scripts to perform web-based application testing, simulate user actions, and retrieve data from the browser.

Requirements

1.Software Requirements:

- JDK: Version 21
- VS Code: Latest version
- Chrome Browser: Installed
- ChromeDriver: Compatible with the Chrome browser

Folder Structure

Selenium

├── lib

│ ├── selenium-java.jar (All jar files)

│ ├── selenium-server.jar

├── bin

│ ├── chromedriver.exe

└── Main.java

Code: Basic Selenium Program (Main.java)

```
import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver; import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

public class Main {

public static void main(String[] args) throws Exception { System.out.println("Starting Selenium program..."); WebDriver driver

= new ChromeDriver();

try {

driver.get("https://www.google.com"); System.out.println("Opened Google homepage.");

WebElement searchBox = driver.findElement(By.name("q")); searchBox.sendKeys("Selenium WebDriver tutorial"); searchBox.submit();
```

System.out.println("Performed search operation.");

Thread.sleep(2000);

String pageTitle = driver.getTitle(); System.out.println("Title of the results page: " + pageTitle);

} finally { driver.quit();

System.out.println("Browser closed. Program completed.");

}

}

}

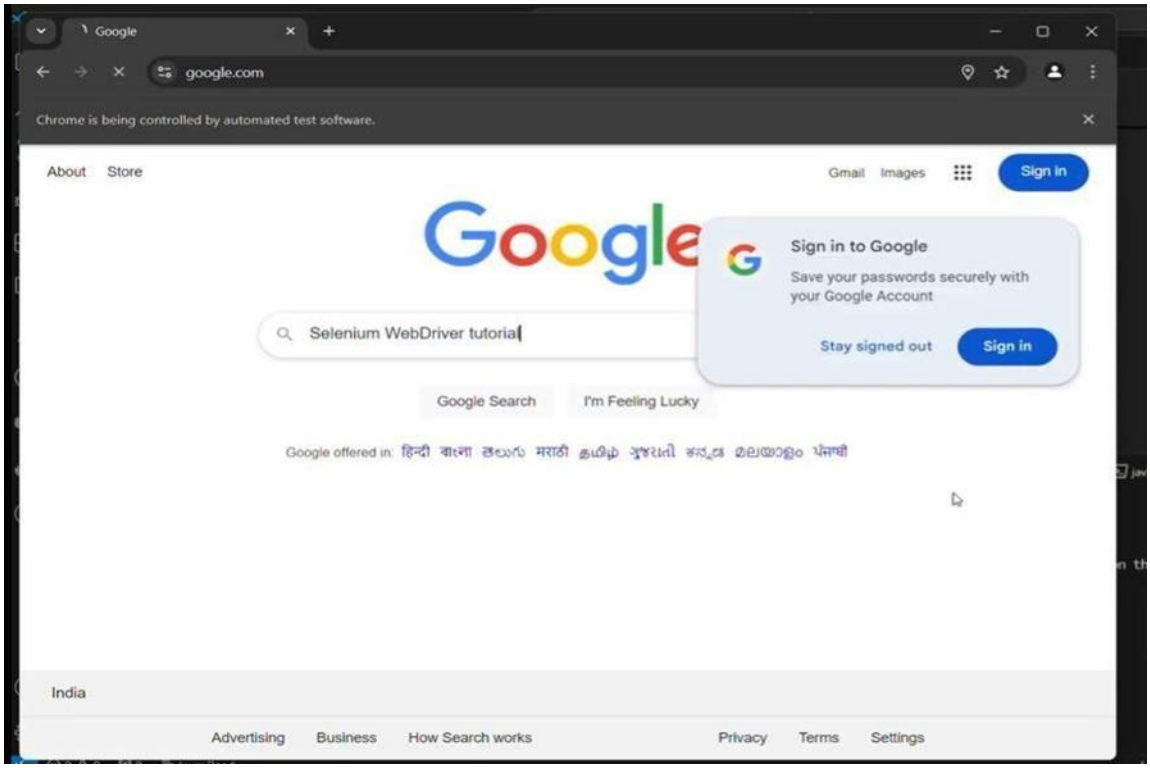
Execution Commands

1.Compile the Code:

javac -cp ".:lib/*" Main.java

2.Run the Program:

java -cp ".:lib/*" Main



Experiment No:

11.Write a Simple Program in JavaScript and Perform Testing Using Selenium

•Objective: Write a JavaScript program to open Google, perform a search, and test it using Selenium WebDriver.

•Concept: Selenium WebDriver enables browser automation using scripts. Here, Node.js is used for JavaScript execution.

Prerequisites:

1.Software Requirements:

- Node.js: Installed on your system.
- Selenium WebDriver: Installed via npm.
- ChromeDriver: Installed and compatible with your browser.
- oBrowser: (e.g., Google Chrome).

Folder Structure

Selenium

├── lib

│ ├── selenium-java.jar (All jar files)

│ ├── selenium-server.jar

├── bin

│ ├── chromedriver.exe

└── test.js

test.js

```
const { Builder, By, Key } = require('selenium-webdriver');

async function example() {

let driver = await new Builder().forBrowser('chrome').build(); try {

// Navigate to Google

await driver.get('https://www.google.com'); console.log('Opened Google homepage.');

// Find the search box and perform a search

let searchBox = await driver.findElement(By.name('q'));

await searchBox.sendKeys('Selenium WebDriver', Key.RETURN); console.log('Search performed successfully.');

} finally {
```

```
// Close the browser await driver.quit(); console.log('Browser closed.');
}
}example();
```

Execution Commands:

1.Install Dependencies:

npm init -y

npm install selenium-webdriver chromedriver

2.Run the Script:

node test.js