List of Experiments:

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

2. Explore Git and GitHub commands.

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

4. Jenkins installation and setup, explore the environment.

5. Demonstrate continuous integration and development using Jenkins.

6. Explore Docker commands for content management.

7. Develop a simple containerized application using Docker.

8. Integrate Kubernetes and Docker

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

10. Install and Explore Selenium for automated testing.

11. Write a simple program in JavaScript and perform testing using Selenium.

12. Develop test cases for the above containerized application using selenium

Experiment 1: Write code for a simple user registration form for an event

Aim: Write code for a simple user registration form for an event.

Objective:

To create a basic user registration form for an event using HTML and CSS.

Procedure:

1. Open a text editor like Visual Studio Code.

2. Create a new HTML file named registration_form.html.

3. Write the HTML code for the registration form including fields like name, email, phone

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>User Registration Form</title>
<link rel="stylesheet" href="styles.css">
</head>
<body>
<h2>User Registration Form</h2>
<form>
<label for="name">Name:</label>
<input type="text" id="name" name="name" required><br>
<label for="email">Email:</label>
<input type="email" id="email" name="email" required><br>
<label for="phone">Phone:</label>
<input type="tel" id="phone" name="phone" required><br>
<input type="submit" value="Submit">
</form>
</body>
</html>
```

4. Style the form using CSS for better presentation.

```
body {
font-family: Arial, sans-serif;
margin: 20px;
}
form {
border: 1px solid #ccc;
padding: 20px;
width: 300px;
}
input[type="text"],
input[type="email"],
input[type="tel"],
input[type="submit"] {
width: 100%;
padding: 10px;
margin: 5px 0;
box-sizing: border-box;
}
input[type="submit"] {
background-color: #4CAF50;
color: white;
border: none;
}
input[type="submit"]:hover {
background-color: #45a049;
}
```

Save the file as styles.css

5. Save the files and open registration_form.html in a web browser to test the form functionality.

EXPERIMENT NO: 2. Explore Git and GitHub commands

Aim: Explore Git and GitHub commands

Objective:

To understand the basic Git commands and how to use GitHub for version control.

the GIT full form is “Global Information Tracker,” a powerful version control system widely used for software development and other collaborative projects. GIT allows multiple developers to work on a project simultaneously while ensuring that their changes do not interfere with one another.

Procedure:

1.Create a GitHub account if you don't have one

Save your email id, user name, file name.

2.Install Git Bash on your system if not already installed.
For Linux: sudo apt-get installgit
For macOS: brew install git
For Windows: Download and install from Git website – 64 bit window exe setup.

3. These are some basic command of git/Linux before going into the git we need to know.

1)To check present working directory - pwd command

Write output-

2)To list the files and folders in the current directory – ls

Write output-

3)To change current directory to desktop - cd Desktop

Write output-

4)To check present working directory - pwd = c/user/rcp/desktop

Write output-

5)To create a directory- mkdir myproject

Write output-

6) Verify if the directory is created, list the files and folders in the current directory – ls

Write output-

7)To change directory to myproject - cd myproject

Write output-

8)To create a empty file in current directory - touch (filename)

Write output-

9)To edit file we will need editor – vi filename (which is similar to note pad)

Write output-

10)To get into the insert mode - type i (you will see --Insert--)

Write output-

11)To get out of insert mode - type ESC key

Write output-

12)To save the file - type :wq(where w is write and q is quit)

Write output-

13)To view the content of the file - cat filename.

Write output-

14)To configure your email id with git hub

git config --global user.email you@example.com

Write output-

15) To configure username with git hub

git config --global user.name "Your Name"

Write output-

16) To initialize current directory as a git local repository - git init

Write output-

17) To verify if local repository created or not - ls -a (you should see a directory called .git/)

Write output-

18) To check the status of git - git status(if the file are in red color then they are in working directory which change can be made)

Write output-

19) To send file from working directory to stageing - git add <file name>

Write output-

20) To check status if it is in green color it is ready to commit - git status

Write output-

21) To commit the files (for which is to stageing to local repository) - git commit -m "this is the updated file"

Write output-

22) again check the status - git status(On branch master nothing to commit, working tree clean)

Write output-

23) To send file from working directory to stageing -git add .

Write output-

24) To check status if it is in green color it is ready to commit - git status

Write output-

25) To commit the files (for which is to stageing to local repository) - git commit -m "this is the updated file"

Write output-

26)To check the Log – git log

Write output-

-Branching - to maintain main branch integrity of the main branch.

will create a new branch (feature branch) or sub branch and will do changes in it and will request to merge.

git branch - * symbol - to verify if the current branch is switched to new branch

24) $ git branch == it shows what are the local branches in our local repository

$ git branch

* master

25) creates a new branch from the new branch. = git branch <newbranchname>

[note - the above command creates a new branch only, it does not switch to new branch and stays in old branch only.]

26)git checkout <branchname> - to switch to new branch

27) ls - to see the content

28) $ git branch – to check where is the * and branch is

29) $ ls

30)touch <filename> to create a new file

31)git status

32)git add .

33) git commit -m “message”

34) $ git merge <submaster >– for merger you need to checkout to master first and merger the file of newbranch to master branch.

Open your GitHub account and created a new Repository

Experiment 3:

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

Aim: To practice source code management using Git and GitHub for the registration form created in Experiment 1. Procedure:

To practice source code management on GitHub, you can follow these steps:

• Create a GitHub account (if you don't already have one.)

• Create a new repository on GitHub.

Create a New Repository

1. **Log In to GitHub**:
    - Go to GitHub and log in with your new credentials.
2. **Navigate to Repositories**:
    - Click on your profile picture in the top-right corner, then select "Your repositories."
3. **Create a New Repository**:
    - Click the "New" button or "Create a new repository."

4 **Fill Out Repository Details**:

- **Repository name**: Enter a name for your repository.
- **Description**: (Optional) Add a description of your repository.
- **Public/Private**: Choose whether you want the repository to be public or private.
- **Initialize this repository with a README**: Optionally, check this box if you want to include a README file.
- Click the "Create repository" button to finalize.

Link your local repository to a GitHub repository using git remote add origin command.

1) $ git remote add origin <url>

Eg - $ git remote add origin https://github.com/vaishu-hub/sindhu.git

2)$ git remote -v

Eg - Output:-

origin https://github.com/Kirangaggs/dharma-new.git (fetch)

origin https://github.com/Kirangaggs/dharma-new.git (push)

---

clone :-

When you create a repository on GitHub, it exists as a remote repository. You can clone your repository to create a local copy on your computer and sync between the two locations.

Cloning a repository from GitHub:

Command - $ git clone <repository-url>

Write output-

Push :- The git push command is used to upload local repository content to a remote repository.

Pushing changes to a remote repository:

Command:- $ git push origin <branchname>

Write output-

==========

Pull:-

The git pull command is used to fetch and download content from a remote repository and immediately update the local repository to match that content.

Command - $ git pull origin <branch name>

Write output-

EXPERIMENT NO: 4. Jenkins installation and setup, explore the environment

Aim: Jenkins installation and setup, explore the environment

Jenkins is an automation server that helps to automate the build, test, and deployment of software

Jenkins is a popular open-source tool for Continuous Integration and Continuous Deployment (CI/CD) in software development. Here are the steps to install and set up Jenkins:

-Frist verify java is installed on machines or not

Open command prompt (cmd) - java --version

if version output given java is installed in machine -> Go for Jenkins installation

Jenkins support java 11,17,21 only so install these versions only

if not, error output is given -> Start with Java installation

- To install Jenkins on Windows 11, you can follow these

steps:

1. Download the Jenkins Installer: Visit the official Jenkins website (https://www.jenkins.io) and navigate to the Downloads page. Download the Windows installer package (usually an .msi file) compatible with your system.

2. Run the Installer: Locate the downloaded .msi file and double-click on it to run the Jenkins installer. You may be prompted to grant administrative permissions.

3. Select Installation Directory: In the installer window, select the installation directory for Jenkins. The default location is typically in the "Program Files" directory. You can choose a different directory if desired.

4. Choose Installation Options: On the next screen, you can select additional installation options, such as creating shortcuts or configuring Jenkins to run as a Windows service. Make the desired selections and proceed.

5. Customize Jenkins URL: In the subsequent screen, you can choose the Jenkins URL, which is the address you'll use to access the Jenkins web interface. By default, it will be http://localhost:8080. You can keep the default or specify a different URL if needed.

6. Complete the Installation: After configuring the installation options, click on the "Install" button to start the installation process. Wait for the installer to complete the installation of Jenkins on your Windows 11 system.

7. Launch Jenkins: Once the installation is finished, you can choose to launch Jenkins automatically by keeping the corresponding checkbox selected. Otherwise, you can manually launch Jenkins later from the Start menu or desktop shortcut.

8. Access Jenkins Web Interface: Open a web browser and enter the Jenkins URL (e.g., http://localhost:8080) in the address bar. The Jenkins web interface should load, and you'll be prompted to unlock Jenkins page by entering the initial administrative password.

9. Retrieve Initial Admin Password: To retrieve the initial administrative password, navigate to the Jenkins installation directory on your computer.

(i.e: c/programData/Jenkins/.jenkins/secrets/initialAdminPassword.

Look for a file called "initialAdminPassword" and open it using a text editor(like Notepad). Copy the password and paste it into the Jenkins web interface to proceed.

10. Follow Setup Wizard: The Jenkins setup wizard will guide you through the remaining configuration steps, including installing recommended plugins and creating the first administrative user. Follow the instructions on the screen to complete the setup. By following these steps, you can install Jenkins on Windows

Give your username and password in Jenkins as admin and admin.

11 and start using it for your continuous integration and continuous delivery (CI/CD) workflows. Note: Make sure your system meets the minimum requirements for running Jenkins, such as having Java Development Kit (JDK) installed.

If port test is not go tik then change to a different port no.

To kill something already running 8080 port

Step 1:

Open up cmd.exe (note: you may need to run it as an administrator, but this isn't always necessary), then run the below command:

netstat -ano | findstr :<PORT>

(Replace <PORT> with the port number you want, but keep the colon)

The area circled in red shows the PID (process identifier). Locate the PID of the process that's using the port you want

Step 2:

Next, run the following command:

taskkill /PID <PID> /F

(No colon this time)

==========

EXPERIMENT NO: 5. Demonstrate continuous integration and development using Jenkins.

Aim: Demonstrate continuous integration and development using Jenkins.

DESCRIPTION

Continuous Integration (CI) and Continuous Development (CD) are important practices in software development that can be achieved using Jenkins.

Here's an example of how you can demonstrate CI/CD using Jenkins:

Setting Up Nginx on Windows with Jenkins and GitHub Integration

Step 1: Installing Nginx on Windows

Download Nginx:

Go to the official Nginx website and download the stable version of Nginx for Windows.

Extract the Nginx Files:

Extract the downloaded .zip file to a directory of your choice, for example, C:\nginx.

Start Nginx:

double click on Nginx --- C:\nginx-1.26.2

Nginx should now be running, and you can verify it by opening your browser and going to http://localhost. You should see the Nginx welcome page.

Step 2: Configure Nginx to Serve a Custom Web Page

Replace the Default Page:

The default page is located in the C:\nginx\html\index.html file.

Replace this with your own index.html [remove the index.html file and copy Registration_form.html and styles.css and rename registration_form.html to index.html] file by copying it into the C:\nginx\html\ directory:

Test Your Custom Page:

Refresh your browser at http://localhost to see your custom page served by Nginx.

Step 3: Set Up GitHub Repository

Create a GitHub Repository:

Create a new repository on GitHub.

Clone the repository to your local machine:

[in git bash]

#git clone https://github.com/yourusername/your-repo.git

Add your index.html file to the repository:

#cd jenkinsproject

#copy index.html and styles.css to Nginx/html [dirctory which we created early.]

#copy C:\path\to\your\index.html .

#git add index.html

#git commit -m "Initial commit"

#git push origin main

Step 4: Configuring Jenkins for Continuous Deployment

Create a Jenkins Job:

In Jenkins, create a new Freestyle project.

1) Under Source Code Management, select Git and enter the URL of your GitHub repository [new repository url].

2) Provide credentials - add-username and password of github.

3) Configure Build Triggers:

Enable GitHub hook trigger for GITScm polling under Build Triggers.

4) Set Up Build Steps:

In the Build section, add a Windows batch command step with the following commands:

#REM Pull the latest changes from GitHub

#git pull origin main

#REM Copy the updated index.html to the Nginx directory

#copy index.html C:\nginx-1.26.2\html\index.html

#copy styles.css C:\nginx-1.26.2\html\styles.css

Step 5: Manual build

1) Modify your index.html file a bit and push it to remote repository.

2) In Jenkins Project page click on "Build Now " to triggger the build manually.

3) Refresh the website hosted on localhost if the changes are deployed or not.

Step 6: Installing and Setting up Ngrok to expose our localhost:8080 to outside world(github wenhook prerequisite)

1) Download and Install ngrok

Download ngrok:

Visit the ngrok website and sign up for a free account.

After signing up, download the ngrok executable for Windows.

Install ngrok:

Extract the downloaded .zip file to a directory of your choice.

2) Connect Your ngrok Account

Open executable file and paste the auth token command given in your ngrok website:

Run the following command to authenticate ngrok with your account (you'll get an authtoken from the ngrok dashboard):

ngrok config add-authtoken 2lTQVCGs8mBCi6skY7H3FtBjtJa_47gtxMnH5L12HTVttjPJ2

3) Expose Jenkins on Port 8080:

Assuming Jenkins is running on port 8080, run the following command in Command Prompt:

#ngrok http 8080

This will create a public URL that tunnels traffic to http://localhost:8080.

Note the Public URL:

Once ngrok is running, it will display an HTTP and HTTPS URL. This URL is now accessible from the internet and will forward requests to your local Jenkins instance.

Example output might look like:

Forwarding https://abcd1234.ngrok.io -> http://localhost:8080

4) Verifying if ngrok setup is working or not:

Open the obtained URL in a new tab in chrome.

Select `visit site` if a certificate related issue is shown while accessing th website.

ER: Jenkins page should be accessible.

Step 7: Setting Up GitHub Webhook

Add a Webhook to Your GitHub Repository:

Go to Settings -> Webhooks in your GitHub repository.

Add a new webhook with the following details:

1) Payload URL: http://<your_localhost>:8080/github-webhook/

2) Content type: application/json

3) SSL verification: Disable(not recommended)

4) Trigger: Just the push event

Step 8: Testing the Setup

Make a Change to index.html:

Modify index.html in your local repository.

Commit and push the changes to GitHub:

#git add index.html

#git commit -m "Updated index.html"

#git push origin main

Check Jenkins:

Jenkins should automatically start a build when it detects the push.

After the build completes, check http://localhost to see the updated page served by Nginx.

EXPERIMENT NO.: 6. Explore Docker commands for content management.

AIM: Explore Docker commands for content management.

DESCRIPTION

Before diving into Docker commands, it's important to understand how Docker fits into content management.

- **Docker** is a platform that enables you to package applications into **containers**. These containers are isolated, lightweight environments that ensure consistency across different environments.
- In the context of content management, Docker can be used to containerize CMS applications (like WordPress, Joomla, etc.), media processing tools, file servers, or static website generators.
- Docker helps ensure that these systems are portable, easy to deploy, and scalable, enabling efficient content management workflows.
- **Container**: A lightweight, standalone, and executable package that includes everything needed to run a piece of software (code, runtime, system tools, libraries).
- **Image**: A read-only template used to create containers. You can think of it as a snapshot of a filesystem and applications.
- **Docker Hub**: A cloud-based repository where Docker images are stored and shared.

2. Install Docker

If Docker desktop is not installed on your system yet, follow these steps:

- **Windows/Mac**: Download and install Docker Desktop from Docker's official site.

Steps to install docker in windows:

Step-1: Verify if wsl is installed in windows or Not

Open CMD

$ wsl --version

Case-1:

If version is displayed wsl is installed.

Note:

We need wsl version 2 in order to install docker

To verify Default version of wsl use the following command

$ wsl --status

Sample Output:

Default Distribution: docker-desktop

Default Version: 2

If wsl version is not 2 but 1 use the following command

$ wsl --set-default-version 2

Case-2:

If wsl is not installed:

-> Open "turn windows feature on or off" from start menu

-> Click on Check box next to "Windows Subsystem for Linux"

-> Click OK

-> Wait for the process to complete

-> Then click on Restart Now Button to restart the system

After System Restarts:

Perform steps in case-1

Or

Wsl --status

Wsl --update

wsl --install –d Ubuntu

Step-2: Download docker desktop

To Install Docker Desktop:

-> In chrome browser: https://docs.docker.com/desktop/install/windows-install/

-> Click on "Docker Desktop for Windows - x86_64"

-> Download Begins.

-> After the Docker desktop Installer download completes.

-> Double-click Docker Desktop Installer.exe to run the installer. By default, Docker Desktop is installed at C:\Program Files\Docker\Docker.

-> When prompted, ensure the Use WSL 2 instead of Hyper-V option on the Configuration page is selected or not depending on your choice of backend.

-> Follow the instructions on the installation wizard to authorize the installer and proceed with the install.

-> When the installation is successful, select Close to complete the installation process.

-> Restart the system if required

Step-3: Account in Docker Hub

-> In Chrome Browser: hub.docker.com

-> Signup using any of the given options

Step-4 : Use Docker hub Account to login to Docker Desktop

Docker is a containerization technology that is widely used for managing application containers. Here are some commonly used Docker commands for content management:

Docker commands

1)docker pull: - To pull the image from docker hub

steps: -

(i)login to docker hub

(ii) Search nginx

(iii) docker pull nginx:latest(repo to local)

docker pull <image-name>:<tag-name>

2) docker image ls: To list the docker images in our system

3) docker run: To run the images available in our system

(i) Syntax: docker run -d -p 8080:80 --name nameofcontainer imgename:latest

Docker run -d <to run container in the background> -p <portforwarding> <hostport:containerport> --name <name of the container> <image-name>:tag

The container on which application is running cannot be directly connected to. To overcome this challenge docker has introduced concept of port forwarding, where one of the port from the host-system (the machine on which the docker is installed) can be mapped to the port of container where the application in running in the container.

4. docker ps : to view all the running containers

5. docker ps -a : to view all the running and non running containers

6. docker rm <container id/container name>: to delete the container which is in stopped state or nor running state.

7.docker stop <container id/container name> : to stop the running container and then you can remove the docker container.

8.docker rm -f nginx-2 : to force remove a running container

9.docker start <container id/container name> : to start the container which are in stopped state.

10.docker images: This command lists all images stored locally on the host.

11.docker tag: To Tag a existing image to a new name

docker tag nginx:latest kirangaggs1/kiran-dockerimage:1

docker tag <current-image:current-tag> <registry-url/reponame:tag-name>

12. docker login -u <docker hub username>: To login to docker account using CLI

Enter password:

13. docker push <registry-url>/<repo-name>:<tag>:1 - To push image into our docker repository

14. docker rmi <imageid/image-name>: To delete an image which is not in use with any container

To push an image from local to docker hub repo:

1. Create a private repo in hub.docker.com

2. Create a tag of the existing image using 11th command

3. Perform Docker login step (12th command)

4. Push the image using 13th command

These are some of the basic Docker commands for managing containers and images. There are many other Docker commands and options that you can use for more advanced use cases, such as managing networks, volumes, and configuration. However, these commands should give you a good starting point for using Docker for content management.

EXPERIMENT NO.: 7. Develop a simple containerized application using Docker

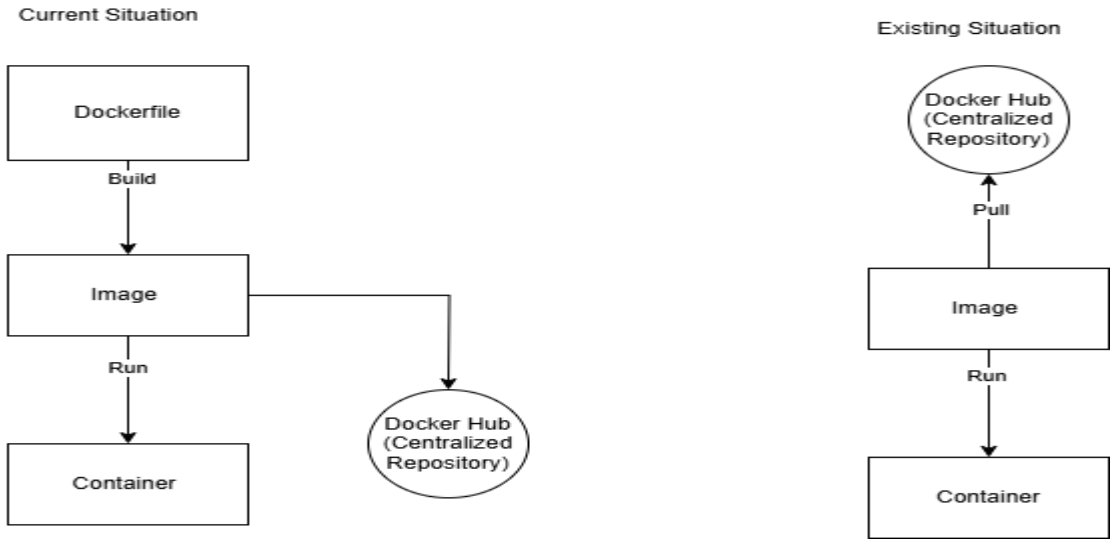
AIM: Develop a simple containerized application using Docker

DESCRIPTION:

Here's an example of how you can develop a simple containerized application using Docker:

Prerequisite:

The docker file and the index.html must be in the same location. Desktop preferably.



Steps:

1. Choose a simple application that you want to containerize.

Eg:- create index.html file on desktop

2. To create own docker image we need a dockerfile.
3. To create a Dockerfile open a editor of your choice.
4. Type the following in the editor like notepad

FROM nginx:latest [a key word to set base image for the container]

RUN rm /usr/share/nginx/html/* [a key word to run a command inside the container]

COPY index.html /usr/share/nginx/html/index.html [a keyword to copy file from local machine to container]

Note: The words in capital letters are referred to as key words.

5. Save the file with the name as "Dockerfile" and save the file without any extension on Desktop.
6. To rename the extension to Dockerfile :- mv .\Dockerfile.txt Dockerfile
7. Execute the following command in terminal to build a image out of the Dockerfile.

docker build -t <name-of-the-image> .

Note -1: .(dot) Means current directory

Note-2: as we are not mentioning any filename in the command, docker will search for a file named:Dokerfile and build an image according to the instructions provided in that file.

8. Docker images
9. Now that the image is created, We can run the image to build a container out of it using the following command:

   docker run -d -p 8081:80 --name <container-name> <image-name-built-in-last-step>

10. Now goto localhost:8081 - you need to get the updated index page.

This is a simple example of how you can use Docker to containerize an application. In a real-world scenario, you would likely have more complex requirements, such as running multiple containers, managing network connections, and persisting data. However, this example should give you a good starting point for using Docker to containerize your applications.

EXPERIMENT NO.: 8. Integrate Kubernetes and Docker

AIM: Integrate Kubernetes and Docker

DESCRIPTION:

Kubernetes and Docker are both popular technologies for managing containers, but they are used for different purposes. Kubernetes is an orchestration platform that provides a higher-level abstraction for managing containers, while Docker is a containerization technology that provides a lower-level runtime for containers.

To integrate Kubernetes and Docker, you need to use Docker to build and package your application as a container image, and then use Kubernetes to manage and orchestrate the containers.

Here's how to integrate Kubernetes and Docker:

1. Install Docker on your local machine or server, if it's not already installed.

2. Install Kubernetes on your local machine or server, if it's not already installed. You can use a cloud provider like AWS or Google Cloud, or you can install Kubernetes on your own servers.

3. Build your Docker image using a Dockerfile, as described in the previous exercise.

4. Push the Docker image to a container registry, such as Docker Hub or a private registry.

5. Create a Kubernetes deployment by creating a YAML file that describes the deployment. The deployment YAML file should specify the Docker image, the number of replicas, and any other configuration options.

6. Use the Kubernetes command-line tool, kubectl, to create the deployment:

kubectl apply -f deployment.yaml

This command creates a Kubernetes deployment using the configuration defined in the YAML file.

7. Use kubectl to check the status of the deployment:

kubectl get deployments

This command lists all Kubernetes deployments in the current namespace, along with their current status.

8. Use kubectl to create a Kubernetes service that exposes the deployment:

kubectl expose deployment my-app --type=LoadBalancer --port=80 --target-port=80

This command creates a Kubernetes service that exposes the deployment on port 80. The service type is LoadBalancer, which means that Kubernetes will automatically provision a

load balancer for the service.

9. Use kubectl to check the status of the service:

kubectl get services

This command lists all Kubernetes services in the current namespace, along with their current status.

10. Access the application by opening a web browser and navigating to the IP address of the load balancer. You should see the application running in the browser.

That's it! You've just integrated Kubernetes and Docker to deploy and manage a containerized application at scale. You can modify the application code, rebuild the Docker image, and update the Kubernetes deployment to test your changes.

Regenerate response

EXPERIMENT NO.: 9. Automate the process of running containerized application developed in exercise 7 using Kubernetes

AIM: Automate the process of running containerized application developed in exercise 7 using Kubernetes.

DESCRIPTION:-

To automate the process of running the containerized application developed in exercise 7 using Kubernetes, we can use Kubernetes YAML files to define the deployment, service, and other resources needed to run the application.

Here are the steps to automate the process:

1. Create a deployment YAML file that defines the deployment for your containerized application. The YAML file should include the following details:

• The Docker image to be used

• The number of replicas to be created

• The port on which the container will listen

Here is an example of a deployment YAML file:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app-deployment
spec:
```

```
  replicas: 3
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
      - name: my-app-container
        image: my-app-image:latest
        ports:
        - containerPort: 8080
```

2. Create a service YAML file that defines the service for your containerized application.

The YAML file should include the following details:

• The port on which the service will listen

• The type of the service (ClusterIP, NodePort, or LoadBalancer)

The target port on which the container is listening

Here is an example of a service YAML file:

```
apiVersion: v1
kind: Service
metadata:
  name: my-app-service
spec:
  selector:
    app: my-app
  ports:
    - name: http
      port: 80
```

targetPort: 8080

type: LoadBalancer

3. Apply the deployment YAML file using the kubectl apply command:

kubectl apply -f deployment.yaml

4. Apply the service YAML file using the kubectl apply command:

kubectl apply -f service.yaml

5. Verify that the deployment and service are running using the kubectl get command:

kubectl get deployments

kubectl get services

6. Access the application using the IP address of the load balancer.

By automating the process of running the containerized application using Kubernetes, we can easily scale the application up or down, roll out updates, and manage the application with ease.

EXPERIMENT NO.: 10. Install and Explore Selenium for automated testing

AIM: Install and Explore Selenium for automated testing

DESCRIPTION:

To install and explore Selenium for automated testing, you can follow these steps:

Install Java Development Kit (JDK):

• Selenium is written in Java, so you'll need to install JDK in order to run it. You can download and install JDK from the official Oracle website.

• Install the Selenium WebDriver:

• You can download the latest version of the Selenium WebDriver from the Selenium website. You'll also need to download the appropriate driver for your web browser of choice (e.g. Chrome Driver for Google Chrome).

Install an Integrated Development Environment (IDE):

• To write and run Selenium tests, you'll need an IDE. Some popular choices include Eclipse, IntelliJ IDEA, and Visual Studio Code.

Install Python on your local machine if it's not already installed. You can download and install the latest version of Python from the official website.

2. Install Selenium using pip, the Python package manager:

pip install selenium

3. Download the web driver for the browser you want to automate. Selenium WebDriver is a collection of language-specific bindings that allow you to control a browser from your code. You can download the web driver for various browsers like Chrome, Firefox, Safari, etc. from the official Selenium website.

4. Set up the Selenium environment by creating a new Python file and importing the necessary modules:

from selenium import webdriver

5. Initialize a new instance of the WebDriver:

driver = webdriver.Chrome()

This will open a new instance of the Chrome browser.

6. Navigate to a webpage using the get method:

driver.get("https://www.google.com")

This will load the Google homepage in the Chrome browser.

7. Interact with elements on the webpage using the various methods provided by the WebDriver API. For example, to search for an element on the page and click on it, you can use the following code:

search_box = driver.find_element_by_name("q")

search_box.send_keys("selenium")

This will search for the term "selenium" on Google and display the search results.

8. Close the WebDriver instance using the quit method:

driver.quit()

This will close the Chrome browser.

By using Selenium for automated testing, you can write scripts to simulate user actions on a webpage, such as filling out forms, clicking on buttons, and navigating through pages. This can help you automate repetitive and time-consuming testing tasks and improve the quality and reliability of your web applications.

EXPERIMENT NO.: 11. Write a simple program in JavaScript and perform testing using Selenium

AIM: Write a simple program in JavaScript and perform testing using Selenium

Simple JavaScript program that you can test using Selenium

```
<!DOCTYPE html>
<html>
<head>
<title>Simple JavaScript Program</title>
</head>
<body>
<p id="output">0</p>
<button id="increment-button">Increment</button>
<script>
const output = document.getElementById("output");
const incrementButton =
document.getElementById("increment-button");
let count = 0;
incrementButton.addEventListener("click", function() {
count += 1;
output.innerHTML = count;
});
</script>
</body>
</html>
```

Write a test case for this program using Selenium

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
```

```
public class Main {
private WebDriver driver;
@Before
public void setUp() {
System.setProperty("webdriver.chrome.driver", "path/to/chromedriver");
driver = new ChromeDriver();
}
@Test
public void testIncrementButton() {
driver.get("file:///path/to/program.html");
driver.findElement(By.id("increment-button")).click();
String result = driver.findElement(By.id("output")).getText();
assert result.equals("1");
}
@After
public void tearDown() {
driver.quit();
}
}
```

You can run the test case using the following command:

```
$ javac Main.java
$ java Main
```

The output of the test case should be:

```
Time: 0.189
OK (1 test)
```

This output indicates that the test case passed, and the increment button was successfully clicked, causing the output to be incremented by 1.

EXPERIMENT NO.: 12. Develop test cases for the above containerized application using selenium

AIM: Develop test cases for the above containerized application using selenium

PROGRAM:

Here is an example of how you could write test cases for the containerized

Application on using Selenium

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
public class Main {
private WebDriver driver;
@Before
public void setUp() {
System.setProperty("webdriver.chrome.driver", "path/to/chromedriver");
driver = new ChromeDriver();
}
@Test
public void testHomePageLoads()
{
driver.get("http://localhost:8080");
String title = driver.getTitle();
assert title.equals("My Containerized Application");
}
@Test
public void testSubmitForm() {
driver.get("http://localhost:8080");
driver.findElement(By.name("name")).sendKeys("John Doe");
driver.findElement(By.name("email")).sendKeys("john.doe@example.co
```

```
m");
driver.findElement(By.name("submit")).click();
String result = driver.findElement(By.id("result")).getText();
assert result.equals("Form submitted successfully!");
}
@After
public void tearDown() {
driver.quit();
}
}
```

You can run the test cases using the following command:

```
$ javac Main.java
$ java Main
```

The output of the test cases should be:

```
..
Time: 1.135
OK (2 tests)
```

This output indicates that both test cases passed, and the containerized application is functioning as expected.