EXPERIMENT NO: 1 **DATE:**

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

App.py

```
from flask import Flask, request, render_template

app = Flask(__name__)

@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':

        name = request.form['name']
        email = request.form['email']
        password = request.form['password']

        return render_template('success.html')

    return render_template('registration.html')

if __name__ == '__main__':
    app.run(host='0.0.0.0')
```

Register.html

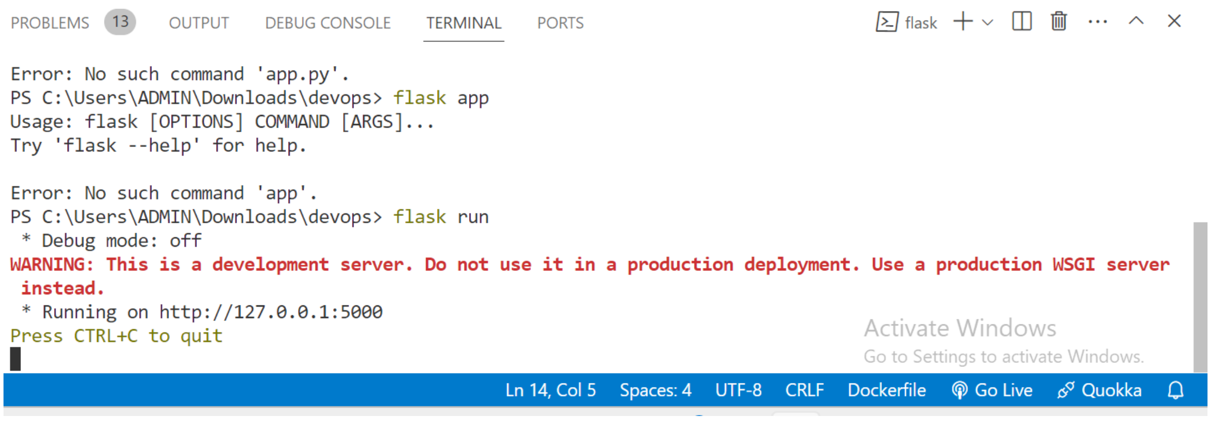
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Registration</title>
</head>
<body>
    <h2>Registration Form</h2>
    <form method="post">
        <input type="text" name="name" placeholder="Name">
        <input type="email" name="email" placeholder="Email">
        <input type="password" name="password" placeholder="Password">
        <input type="submit" value="Submit">
    </form>
</body>
</html>
```

Success.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>RegistrationSuccess</title>
</head>
<body>
    <h2>Registration Successful</h2>
</body>
</html>
```

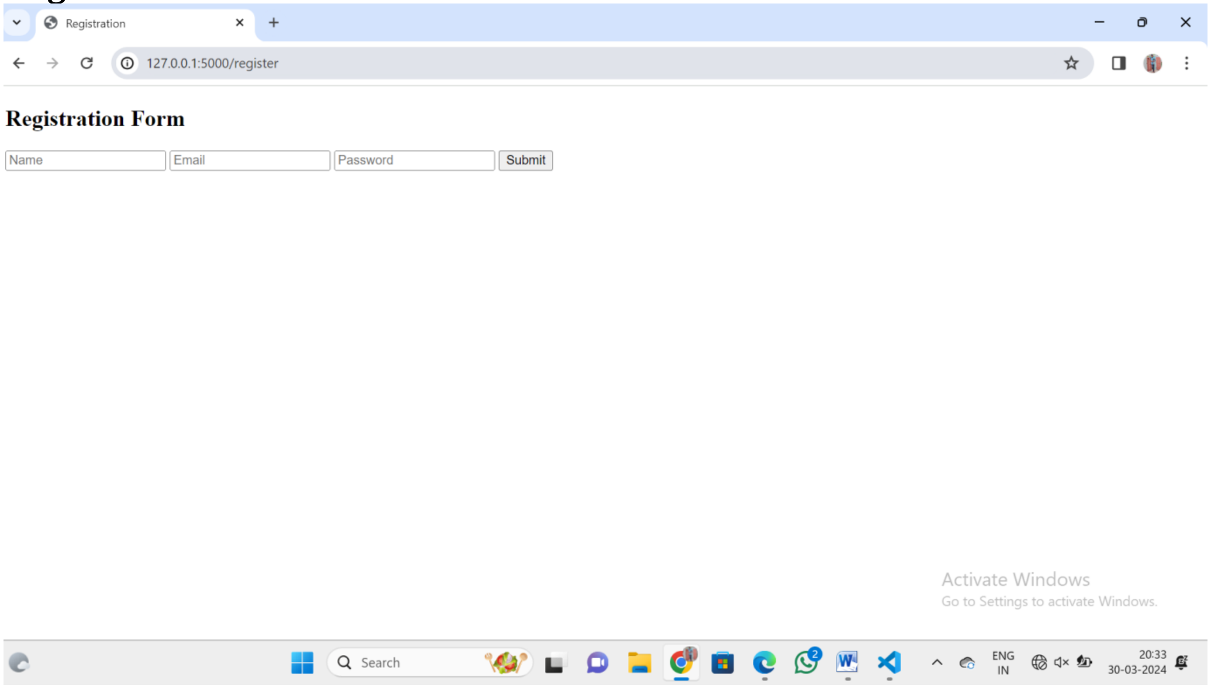
OUTPUT:

Flask run



```
Error: No such command 'app.py'.
PS C:\Users\ADMIN\Downloads\devops> flask app
Usage: flask [OPTIONS] COMMAND [ARGS]...
Try 'flask --help' for help.

Error: No such command 'app'.
PS C:\Users\ADMIN\Downloads\devops> flask run
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server
 instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

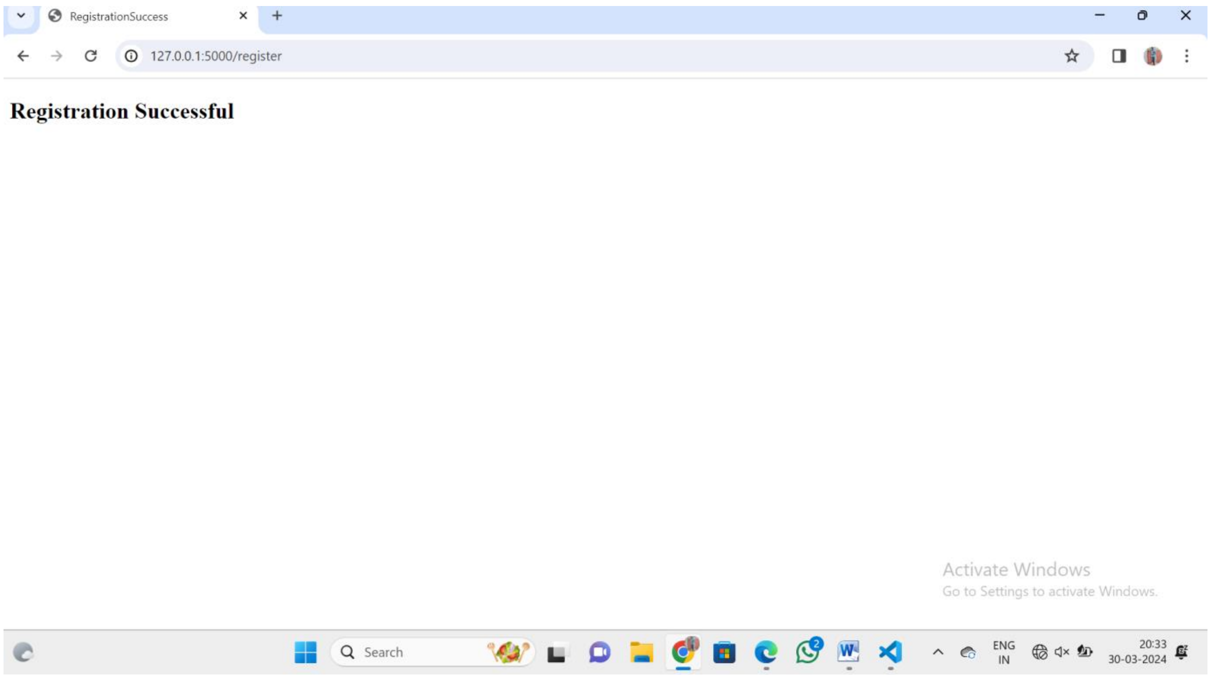
Register.html



Registration Form

Name	Email	Password	Submit

Success.html



RegistrationSuccess

127.0.0.1:5000/register

Registration Successful

EXPERIMENT NO: 2 **DATE:**

Program: Explore Git and GitHub commands

Git and GitHub are essential tools for version control and collaborative development. Below, I'll outline some common Git commands and how they interact with GitHub, the popular online platform for hosting Git repositories.

➢ Basic Git Commands:

1. **git init**: Initialize a new Git repository in the current directory.

2. **git clone [url]**: Clone a repository from a remote URL (like GitHub) to your local machine.

3. **git add [file]**: Add changes in a specific file to the staging area.

4. **git add .**: Add all changes in the current directory and its subdirectories to the staging area.

5. **git commit -m "message"**: Commit staged changes with a descriptive message.

6. **git status**: View the status of changes (untracked, modified, staged) in your repository.

7. **git diff**: Show changes between commits, commit and working tree, etc.

8. **git log**: View commit history.

9. **git branch**: List, create, or delete branches.

10. **git checkout [branch]**: Switch to a different branch.

11. **git merge [branch]**: Merge a branch into the current branch.

12. **git pull**: Fetch from and integrate with another repository or a local branch.

13. **git push**: Update remote references and associated objects.

➢ GitHub Commands:

1. **git remote add origin [url]**: Add a remote repository URL (typically on GitHub) as the origin.

2. **git remote -v**: List all remote repositories along with their URLs.

3. **git push -u origin [branch]**: Push the local branch to the remote repository (GitHub), setting the upstream branch.

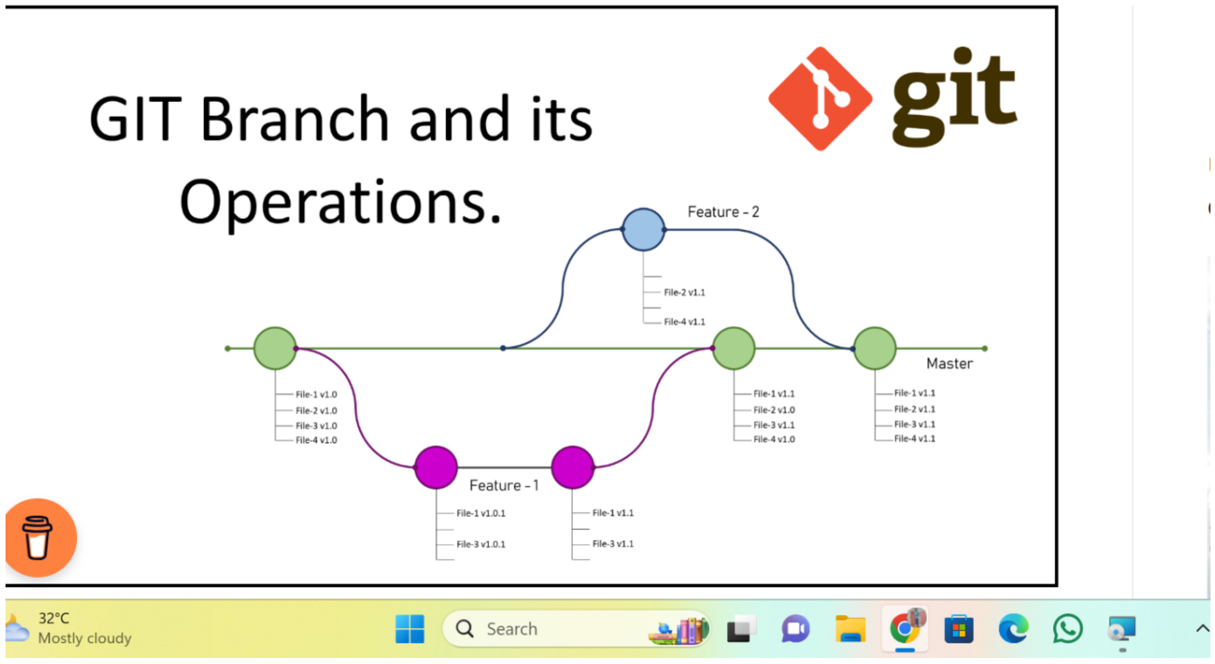
4. **git pull origin [branch]**: Pull changes from a remote repository into the current branch.

5. **git clone [url]**: Clone a repository from GitHub to your local machine.

6. **git fork**: Create a personal copy of someone else's repository on GitHub.

7. **git pull-request**: Open a pull request on GitHub to propose changes and start a code review process.

8. **git merge**: Merge changes from a pull request into the main branch.



EXPERIMENT NO: 3 **DATE:**

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

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

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

Account:-https://github.com/Kurmapulokesh

• Create a new repository on GitHub.

Repository Name:-demo

• Clone the repository to your local machine: $ git clone <repository-url>

$git clone https://github.com/Kurmapulokesh/demo11

• Move to the repository directory: $ cd <repository-name>

$cd demo

• Stage the changes: $ git add <file-name>

$git add app.py templates

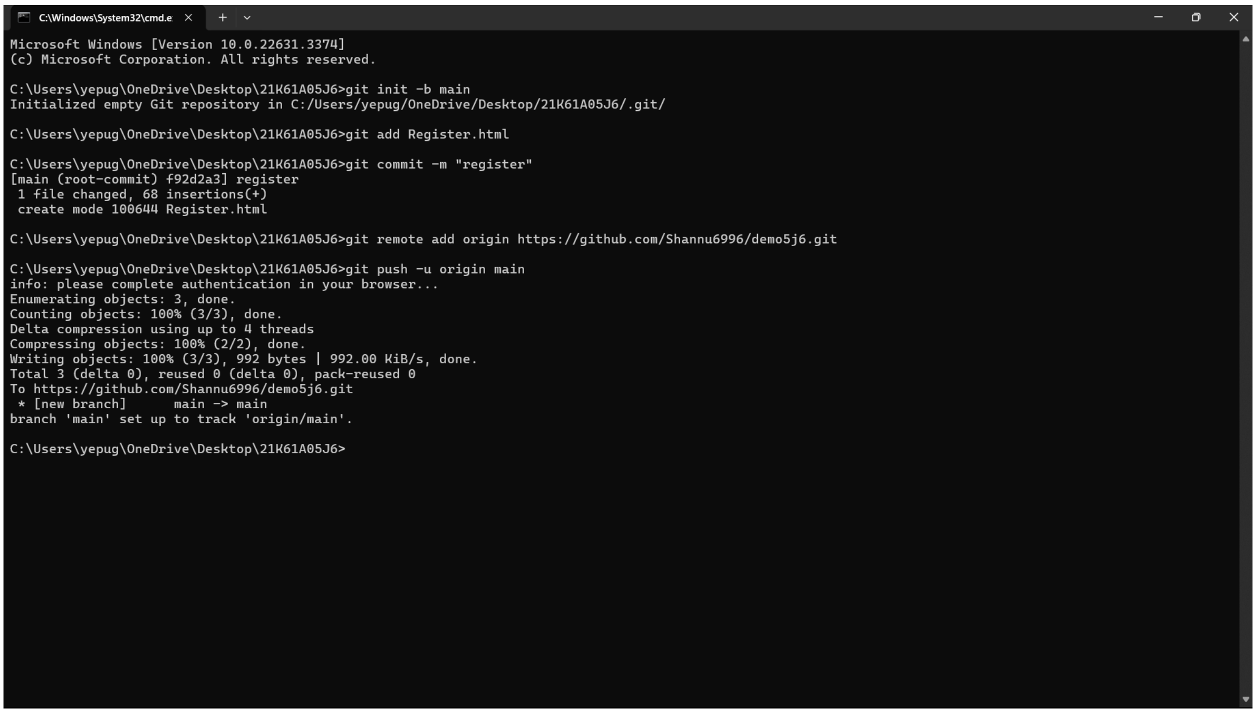
• Commit the changes: $ git commit -m "Added source code for a simple user registration form"

$git commit –m “added app.py and templates”

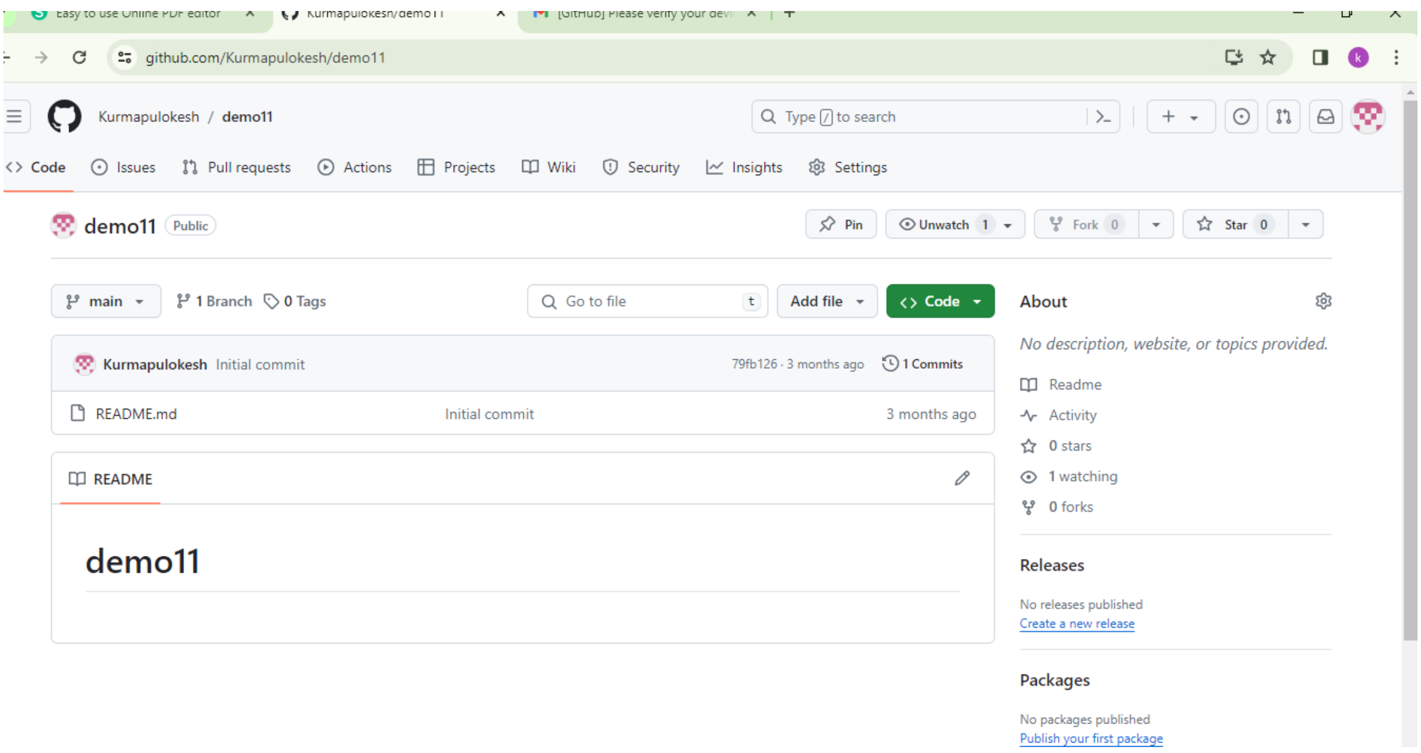
• Push the changes to the remote repository: $ git push origin master

$git push

Output:-



```
Microsoft Windows [Version 10.0.22631.3374]
(c) Microsoft Corporation. All rights reserved.

C:\Users\yepug\OneDrive\Desktop\21K61A05J6>git init -b main
Initialized empty Git repository in C:/Users/yepug/OneDrive/Desktop/21K61A05J6/.git/

C:\Users\yepug\OneDrive\Desktop\21K61A05J6>git add Register.html

C:\Users\yepug\OneDrive\Desktop\21K61A05J6>git commit -m "register"
[main (root-commit) f92d2a3] register
 1 file changed, 68 insertions(+)
 create mode 100644 Register.html

C:\Users\yepug\OneDrive\Desktop\21K61A05J6>git remote add origin https://github.com/Shannu6996/demo5j6.git

C:\Users\yepug\OneDrive\Desktop\21K61A05J6>git push -u origin main
info: please complete authentication in your browser...
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 992 bytes | 992.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Shannu6996/demo5j6.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.

C:\Users\yepug\OneDrive\Desktop\21K61A05J6>
```



Kurmapulokesh Initial commit 79fb126 · 3 months ago 1 Commits

README.md Initial commit 3 months ago

README

demo11

About

No description, website, or topics provided.

Readme
Activity
0 stars
1 watching
0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

EXPERIMENT NO: 4 **DATE:**

Program: Jenkins installation and setup, explore the environment.

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:

Download and install Jenkins:

- Download the Jenkins package for your operating system from the Jenkins website.
- Follow the installation instructions for your operating system to install Jenkins.

Start the Jenkins service:

- On Windows, use the Windows Services Manager to start the Jenkins service.
- On Linux, use the following command to start the Jenkins service:

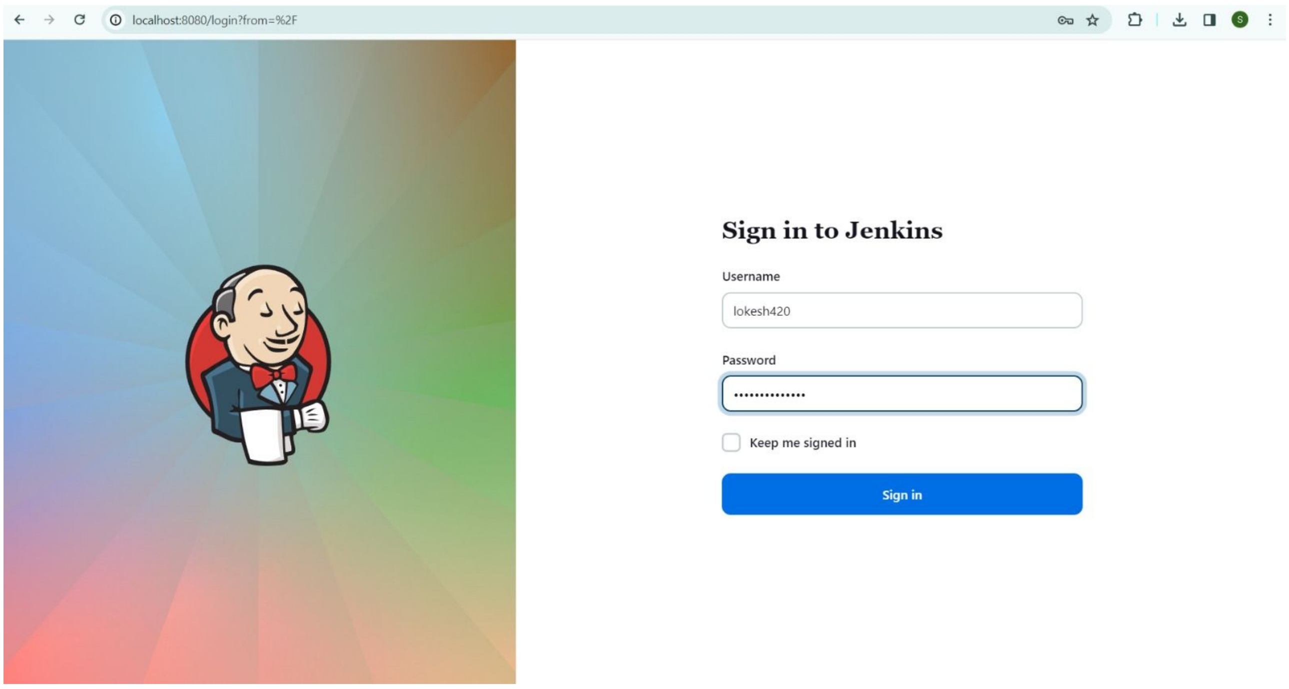
```
$ sudo service jenkins start
```

Access the Jenkins web interface:

- Open a web browser and navigate to http://localhost:8080 to access the Jenkins web interface.
- If the Jenkins service is running, you will see the Jenkins login page.

Initialize the Jenkins environment:

- Follow the instructions on the Jenkins setup wizard to initialize the Jenkins environment.
- This process involves installing recommended plugins, setting up security, and creating the first admin user.


localhost:8080/login?from=%2F

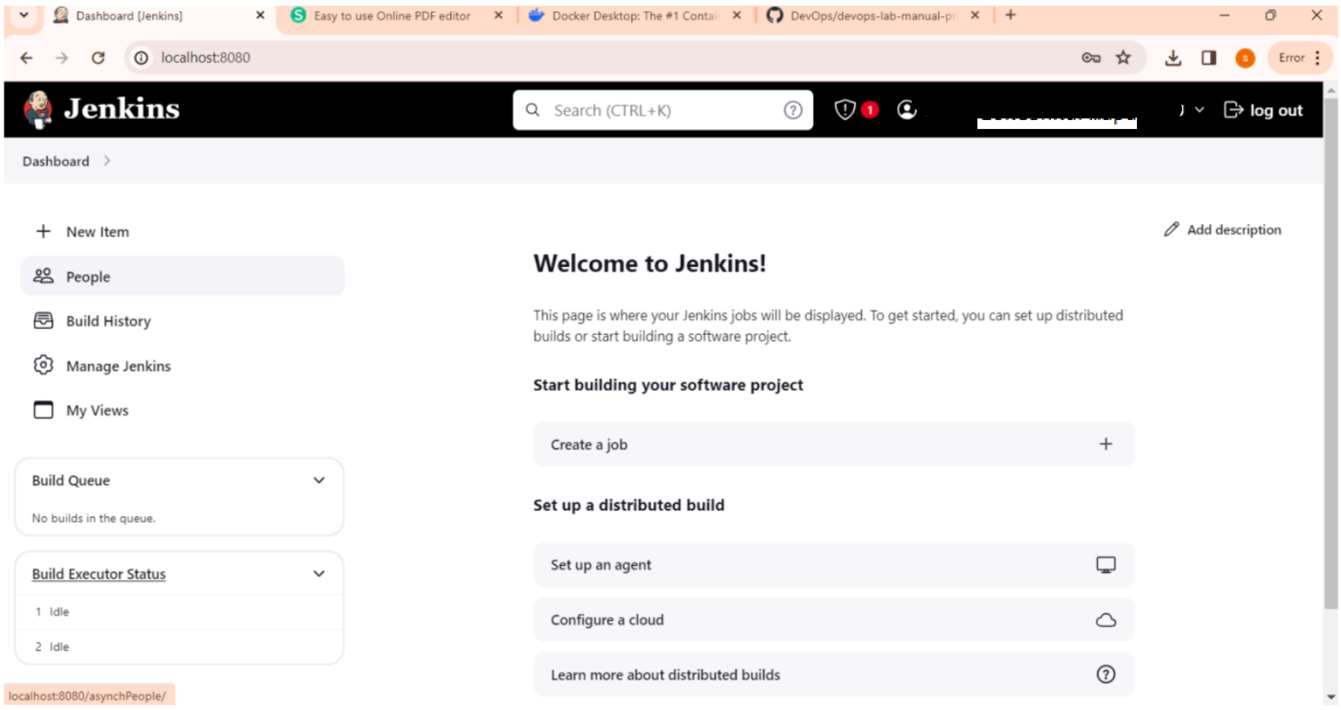
Sign in to Jenkins

Username: lokesh420

Password: ••••••••••••••

Keep me signed in

Sign in


Dashboard [Jenkins] | Easy to use Online PDF editor | Docker Desktop: The #1 Contai | DevOps/devops-lab-manual-pr

localhost:8080

Jenkins | Search (CTRL+K) | [redacted] | log out

Dashboard >

- New Item
- People
- Build History
- Manage Jenkins
- My Views

Build Queue
No builds in the queue.

Build Executor Status
1 Idle
2 Idle

Add description

Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

Create a job

Set up a distributed build

Set up an agent

Configure a cloud

Learn more about distributed builds

localhost:8080/asynchPeople/

EXPERIMENT NO: 5 **DATE:**

Program:-Demonstrate continuous integration and development using Jenkins.

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: Create a

simple Java application:

- Create a simple Java application that you want to integrate with Jenkins.
- The application should have some basic functionality, such as printing "Hello World" or performing simple calculations.

Commit the code to a Git repository:

- Create a Git repository for the application and commit the code to the repository.
- Make sure that the Git repository is accessible from the Jenkins server.

Create a Jenkins job:

- Log in to the Jenkins web interface and create a new job.
- Configure the job to build the Java application from the Git repository. Specify the build triggers, such as building after every commit to the repository.

Build the application:

- Trigger a build of the application using the Jenkins job.
- The build should compile the code, run any tests, and produce an executable jar file.

Monitor the build:

- Monitor the build progress in the Jenkins web interface.
- The build should show the build log, test results, and the status of the build.

Deploy the application:

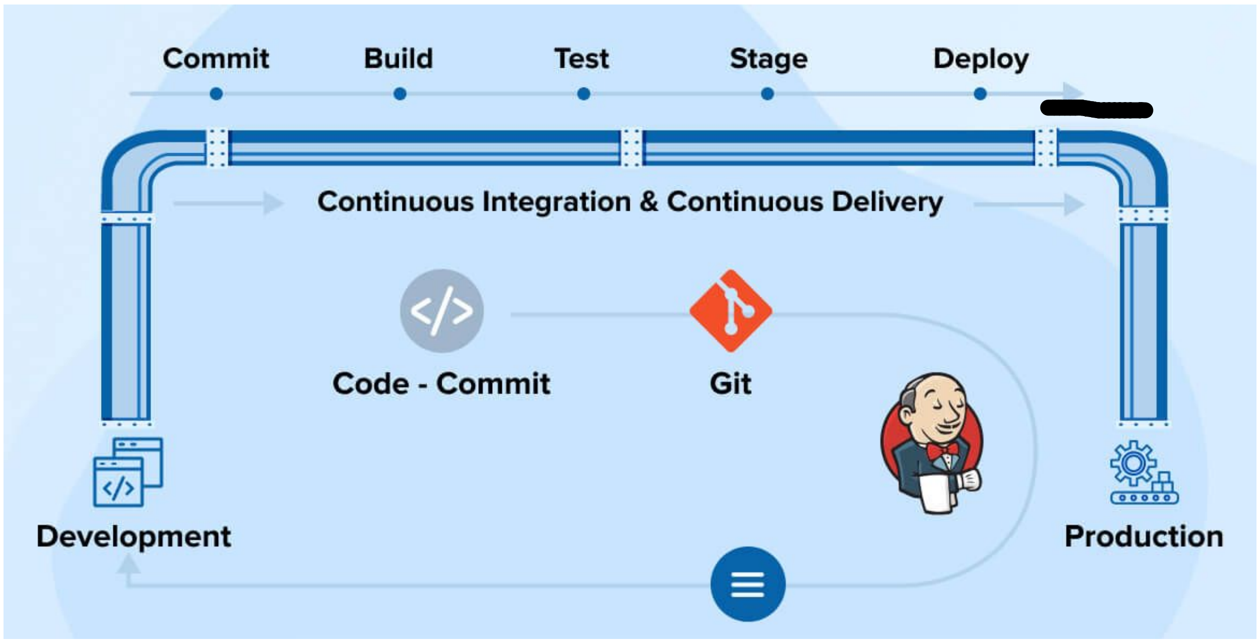
- If the build is successful, configure the Jenkins job to deploy the application to a production environment.
- The deployment could be as simple as copying the jar file to a production server or using a more sophisticated deployment process, such as using a containerization technology like Docker.

Repeat the process:

- Repeat the process for subsequent changes to the application.
- Jenkins should automatically build and deploy the changes to the production environment.

This is a basic example of how you can use Jenkins to demonstrate CI/CD in software development. In a real-world scenario, you would likely have more complex requirements, such as multiple environments, different types of tests, and a more sophisticated deployment process. However, this example should give you a good starting point for using Jenkins for CI/CD in your software development projects.

Output:-



EXPERIMENT NO: 6 **DATE:**

Program:-Explore Docker commands for content management.

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

- **Docker run:** Run a command in a new container.

For example: $ docker run --name mycontainer -it ubuntu:16.04 /bin/bash This command runs a new container based on the Ubuntu 16.04 image and starts a shell session in the container.

- **Docker start**: Start one or more stopped containers.

For example: $ docker start mycontainer

This command starts the container named "mycontainer".

- **Docker stop:** Stop one or more running containers.

For example: $ docker stop mycontainer

This command stops the container named "mycontainer".

- **Docker rm:** Remove one or more containers.

For example: $ docker rm mycontainer

This command removes the container named "mycontainer".

- **Docker ps:** List containers.

For example: $ docker ps

This command lists all running containers.

- **Docker images**: List images.

For example: $ docker images

This command lists all images stored locally on the host.

- **Docker pull**: Pull an image or a repository from a registry.

For example: $ docker pull ubuntu:16.04

This command pulls the Ubuntu 16.04 image from the Docker Hub registry.

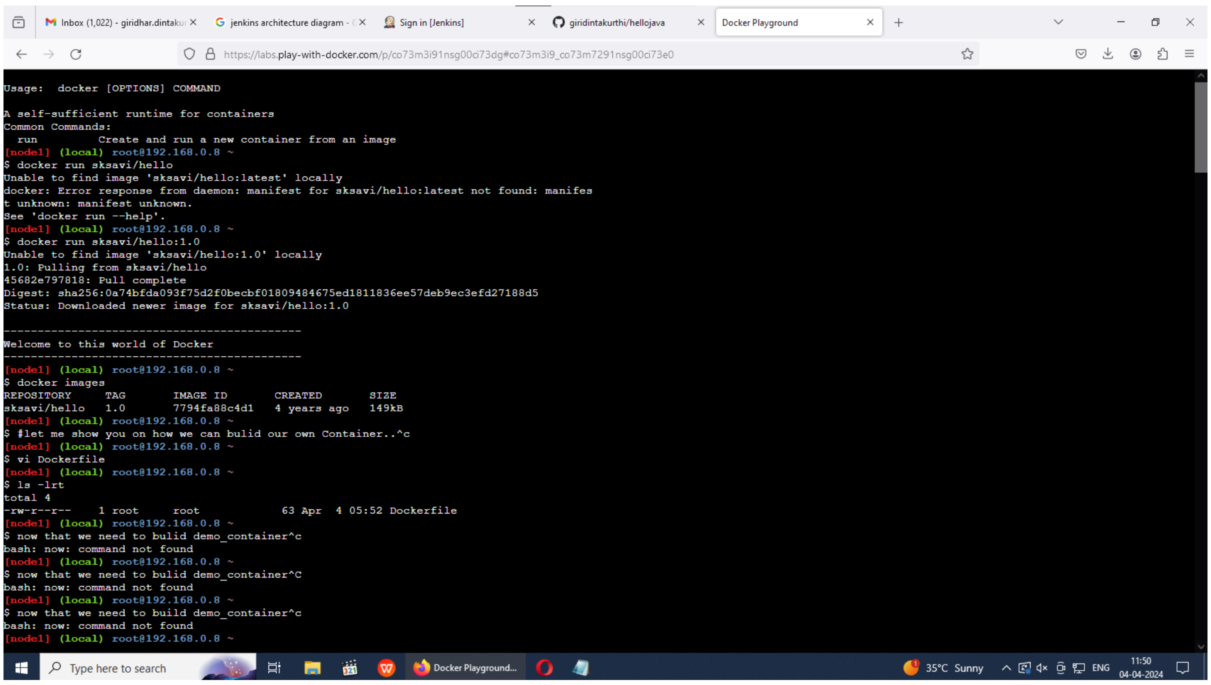
- **Docker push:** Push an image or a repository to a registry.

For example: $ docker push myimage

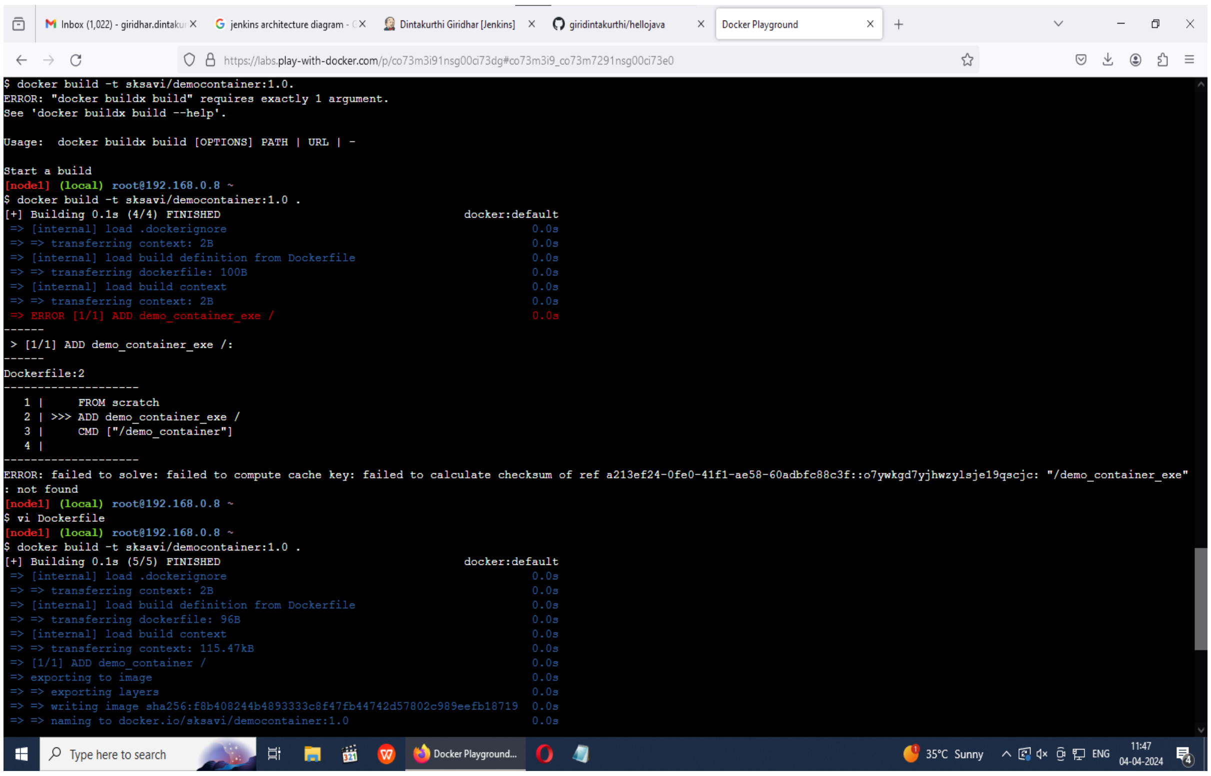
This command pushes the image named "myimage" to the Docker Hub registry.

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

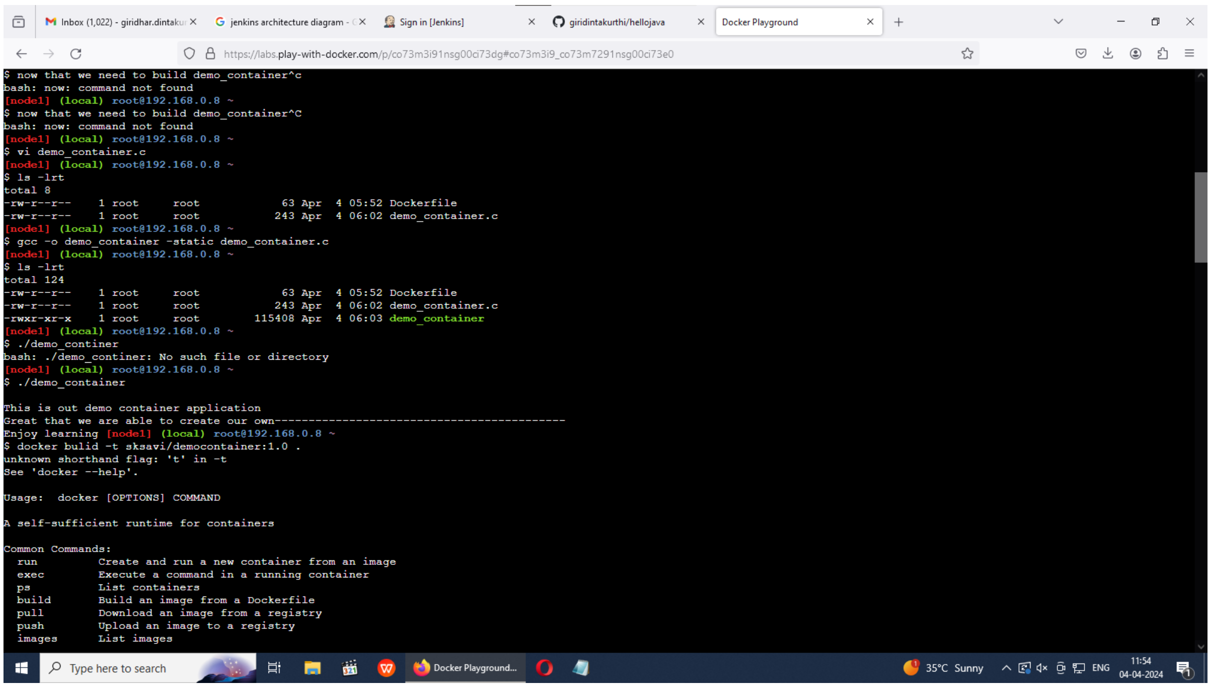
Output:-



```
Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers
Common Commands:
  run         Create and run a new container from an image
[node1] (local) root@192.168.0.8 ~
$ docker run sksavi/hello
Unable to find image 'sksavi/hello:latest' locally
docker: Error response from daemon: manifest for sksavi/hello:latest not found: manifes
t unknown: manifest unknown.
See 'docker run --help'.
[node1] (local) root@192.168.0.8 ~
$ docker run sksavi/hello:1.0
Unable to find image 'sksavi/hello:1.0' locally
1.0: Pulling from sksavi/hello
45682e797818: Pull complete
Digest: sha256:0a74bfda093f75d2f0becbf01809484675ed1811836ee57deb9ec3efd27188d5
Status: Downloaded newer image for sksavi/hello:1.0

---------------------------------------------
Welcome to this world of Docker
---------------------------------------------
[node1] (local) root@192.168.0.8 ~
$ docker images
REPOSITORY     TAG       IMAGE ID       CREATED       SIZE
sksavi/hello   1.0       7794fa88c4d1   4 years ago   149kB
[node1] (local) root@192.168.0.8 ~
$ #let me show you on how we can bulid our own Container..^c
[node1] (local) root@192.168.0.8 ~
$ vi Dockerfile
[node1] (local) root@192.168.0.8 ~
$ ls -lrt
total 4
-rw-r--r--    1 root     root            63 Apr  4 05:52 Dockerfile
[node1] (local) root@192.168.0.8 ~
$ now that we need to bulid demo_container^c
bash: now: command not found
[node1] (local) root@192.168.0.8 ~
$ now that we need to bulid demo_container^C
bash: now: command not found
[node1] (local) root@192.168.0.8 ~
$ now that we need to build demo_container^c
bash: now: command not found
[node1] (local) root@192.168.0.8 ~
```



```
$ docker build -t sksavi/democontainer:1.0.
ERROR: "docker buildx build" requires exactly 1 argument.
See 'docker buildx build --help'.

Usage:  docker buildx build [OPTIONS] PATH | URL | -

Start a build
[node1] (local) root@192.168.0.8 ~
$ docker build -t sksavi/democontainer:1.0 .
[+] Building 0.1s (4/4) FINISHED                                 docker:default
 => [internal] load .dockerignore                                          0.0s
 => => transferring context: 2B                                            0.0s
 => [internal] load build definition from Dockerfile                       0.0s
 => => transferring dockerfile: 100B                                       0.0s
 => [internal] load build context                                          0.0s
 => => transferring context: 2B                                            0.0s
 => ERROR [1/1] ADD demo_container_exe /                                   0.0s
------
 > [1/1] ADD demo_container_exe /:
------
Dockerfile:2
--------------------
   1 |     FROM scratch
   2 | >>> ADD demo_container_exe /
   3 |     CMD ["/demo_container"]
   4 |
--------------------
ERROR: failed to solve: failed to compute cache key: failed to calculate checksum of ref a213ef24-0fe0-41f1-ae58-60adbfc88c3f::o7ywkgd7yjhwzylsje19qscjc: "/demo_container_exe"
: not found
[node1] (local) root@192.168.0.8 ~
$ vi Dockerfile
[node1] (local) root@192.168.0.8 ~
$ docker build -t sksavi/democontainer:1.0 .
[+] Building 0.1s (5/5) FINISHED                                 docker:default
 => [internal] load .dockerignore                                          0.0s
 => => transferring context: 2B                                            0.0s
 => [internal] load build definition from Dockerfile                       0.0s
 => => transferring dockerfile: 96B                                        0.0s
 => [internal] load build context                                          0.0s
 => => transferring context: 115.47kB                                      0.0s
 => [1/1] ADD demo_container /                                             0.0s
 => exporting to image                                                     0.0s
 => => exporting layers                                                    0.0s
 => => writing image sha256:f8b408244b4893333c8f47fb44742d57802c989eefb18719  0.0s
 => => naming to docker.io/sksavi/democontainer:1.0                        0.0s
```



```
$ now that we need to build demo_container^c
bash: now: command not found
[node1] (local) root@192.168.0.8 ~
$ now that we need to build demo_container^C
bash: now: command not found
[node1] (local) root@192.168.0.8 ~
$ vi demo_container.c
[node1] (local) root@192.168.0.8 ~
$ ls -lrt
total 8
-rw-r--r--    1 root     root            63 Apr  4 05:52 Dockerfile
-rw-r--r--    1 root     root           243 Apr  4 06:02 demo_container.c
[node1] (local) root@192.168.0.8 ~
$ gcc -o demo_container -static demo_container.c
[node1] (local) root@192.168.0.8 ~
$ ls -lrt
total 124
-rw-r--r--    1 root     root            63 Apr  4 05:52 Dockerfile
-rw-r--r--    1 root     root           243 Apr  4 06:02 demo_container.c
-rwxr-xr-x    1 root     root        115408 Apr  4 06:03 demo_container
[node1] (local) root@192.168.0.8 ~
$ ./demo_continer
bash: ./demo_continer: No such file or directory
[node1] (local) root@192.168.0.8 ~
$ ./demo_container

This is out demo container application
Great that we are able to create our own-------------------------------------------
Enjoy learning [node1] (local) root@192.168.0.8 ~
$ docker bulid -t sksavi/democontainer:1.0 .
unknown shorthand flag: 't' in -t
See 'docker --help'.

Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

Common Commands:
  run         Create and run a new container from an image
  exec        Execute a command in a running container
  ps          List containers
  build       Build an image from a Dockerfile
  pull        Download an image from a registry
  push        Upload an image to a registry
  images      List images
```

EXPERIMENT NO: 7 **DATE:**

Program:- Develop a simple containerized application using Docker.

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

Choose an application:

• Choose a simple application that you want to containerize. For example, a Python script that prints "Hello World".

Write a Dockerfile:

• Create a file named "Dockerfile" in the same directory as the application. In the Dockerfile, specify the base image, copy the application into the container, and specify the command to run the application.

Here's an example Dockerfile for a Python script:

```
# Use the official Python image as the base image

FROM python:3.9

 # Copy the Python script into the container

 COPY hello.py /app

# Set the working directory to

/app/ WORKDIR /app/

# Run the Python script when the container starts

CMD ["python", "hello.py"]
```

• Build the Docker image: Run the following command to build the Docker image: $ docker build -t myimage .

This command builds a new Docker image using the Dockerfile and tags the image with the name "myimage".

• Run the Docker container: Run the following command to start a new container based on the image:

```
$ docker run --name mycontainer myimage
```

This command starts a new container named "mycontainer" based on the "myimage" image and runs the Python script inside the container.

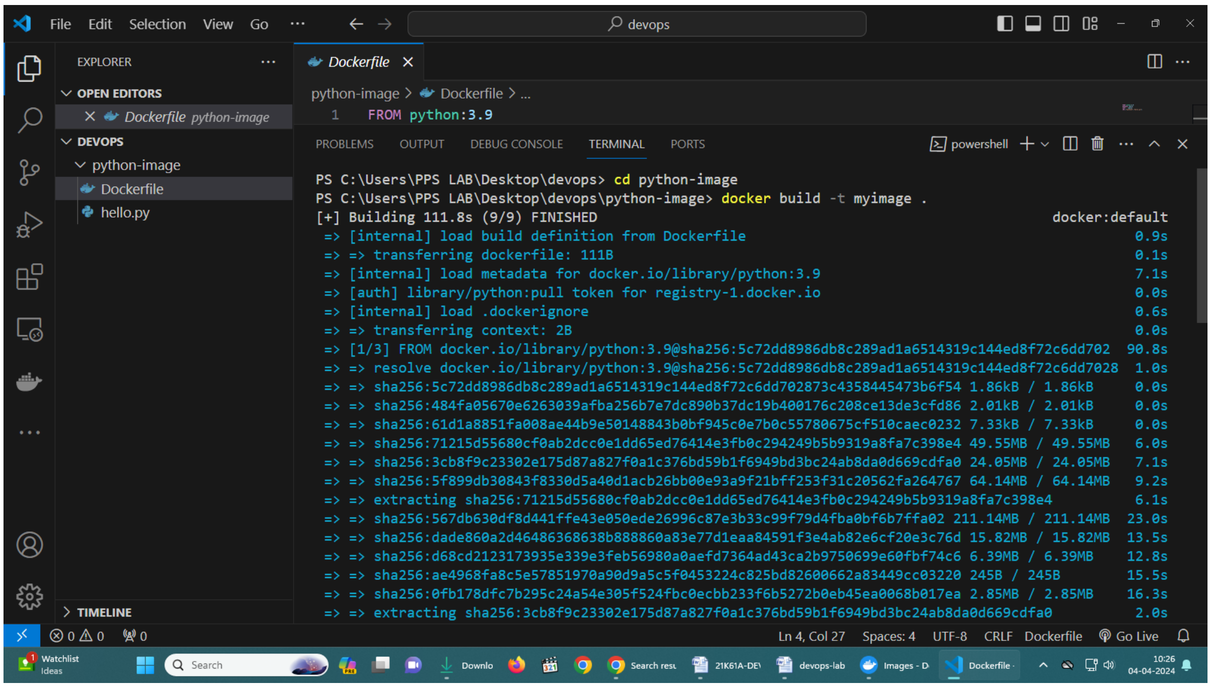
• Verify the output: Run the following command to verify the output of the container:

```
$ docker logs mycontainer
```

This command displays the logs of the container and should show the "Hello World" output.

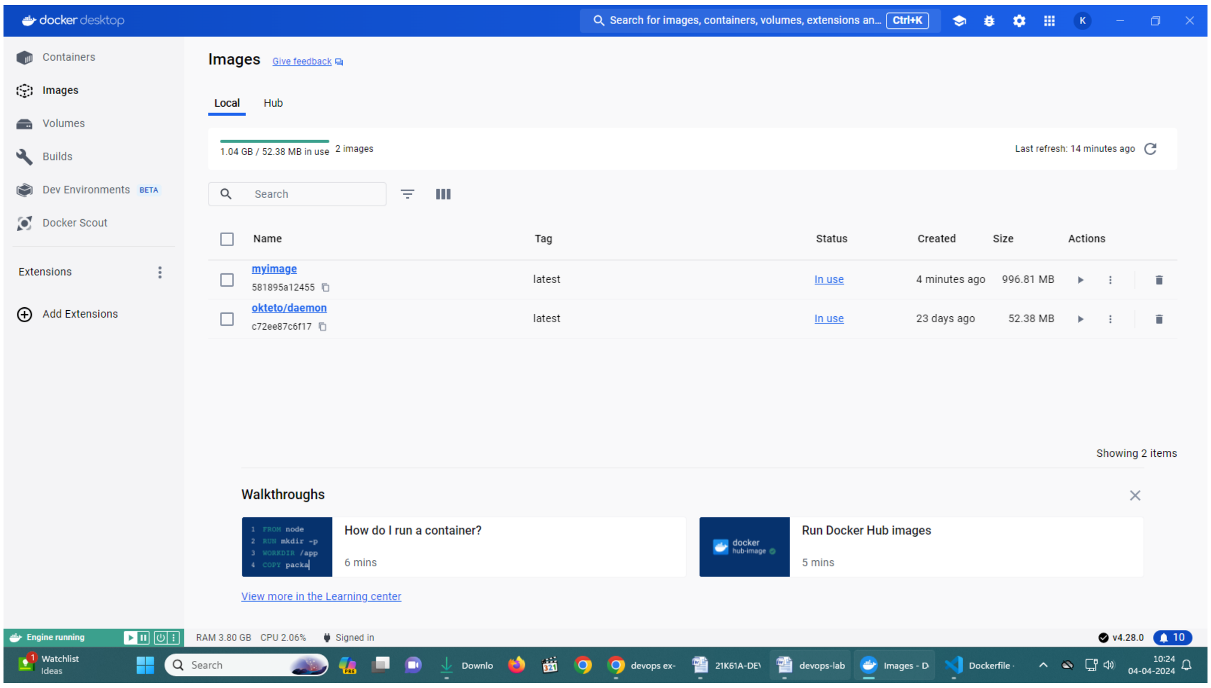
This is a simple example of how you can use Docker to containerize an application. In a real-world scenario, you would likely have more complex requirements, such as running multiple containers, managing network connections, and persisting data. However, this example should give you a good starting point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

Output:-



```
FROM python:3.9
```

```
PS C:\Users\PPS LAB\Desktop\devops> cd python-image
PS C:\Users\PPS LAB\Desktop\devops\python-image> docker build -t myimage .
[+] Building 111.8s (9/9) FINISHED                                                docker:default
 => [internal] load build definition from Dockerfile                                         0.9s
 => => transferring dockerfile: 111B                                                         0.1s
 => [internal] load metadata for docker.io/library/python:3.9                                7.1s
 => [auth] library/python:pull token for registry-1.docker.io                                0.0s
 => [internal] load .dockerignore                                                            0.6s
 => => transferring context: 2B                                                              0.0s
 => [1/3] FROM docker.io/library/python:3.9@sha256:5c72dd8986db8c289ad1a6514319c144ed8f72c6dd702  90.8s
 => => resolve docker.io/library/python:3.9@sha256:5c72dd8986db8c289ad1a6514319c144ed8f72c6dd7028  1.0s
 => => sha256:5c72dd8986db8c289ad1a6514319c144ed8f72c6dd702873c4358445473b6f54 1.86kB / 1.86kB    0.0s
 => => sha256:484fa05670e6263039afba256b7e7dc890b37dc19b400176c208ce13de3cfd86 2.01kB / 2.01kB    0.0s
 => => sha256:61d1a8851fa008ae44b9e50148843b0bf945c0e7b0c55780675cf510caec0232 7.33kB / 7.33kB    0.0s
 => => sha256:71215d55680cf0ab2dcc0e1dd65ed76414e3fb0c294249b5b9319a8fa7c398e4 49.55MB / 49.55MB   6.0s
 => => sha256:3cb8f9c23302e175d87a827f0a1c376bd59b1f6949bd3bc24ab8da0d669cdfa0 24.05MB / 24.05MB   7.1s
 => => sha256:5f899db30843f8330d5a40d1acb26bb00e93a9f21bff253f31c20562fa264767 64.14MB / 64.14MB   9.2s
 => => extracting sha256:71215d55680cf0ab2dcc0e1dd65ed76414e3fb0c294249b5b9319a8fa7c398e4         6.1s
 => => sha256:567db630df8d441ffe43e050ede26996c87e3b33c99f79d4fba0bf6b7ffa02 211.14MB / 211.14MB  23.0s
 => => sha256:dade860a2d46486368638b888860a83e77d1eaa84591f3e4ab82e6cf20e3c76d 15.82MB / 15.82MB  13.5s
 => => sha256:d68cd2123173935e339e3feb56980a0aefd7364ad43ca2b9750699e60fbf74c6 6.39MB / 6.39MB    12.8s
 => => sha256:ae4968fa8c5e57851970a90d9a5c5f0453224c825bd82600662a83449cc03220 245B / 245B        15.5s
 => => sha256:0fb178dfc7b295c24a54e305f524fbc0ecbb233f6b5272b0eb45ea0068b017ea 2.85MB / 2.85MB    16.3s
 => => extracting sha256:3cb8f9c23302e175d87a827f0a1c376bd59b1f6949bd3bc24ab8da0d669cdfa0         2.0s
```



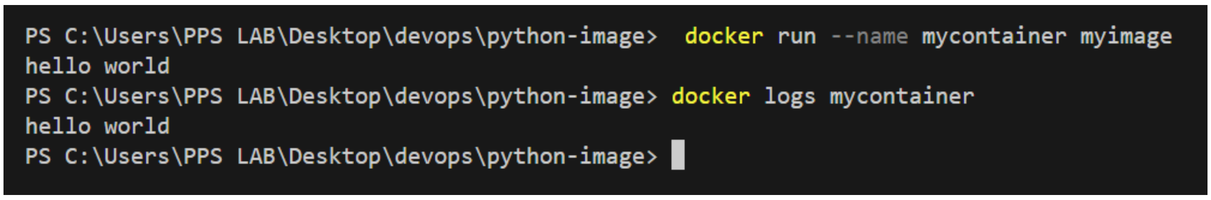
Images

1.04 GB / 52.38 MB in use 2 images

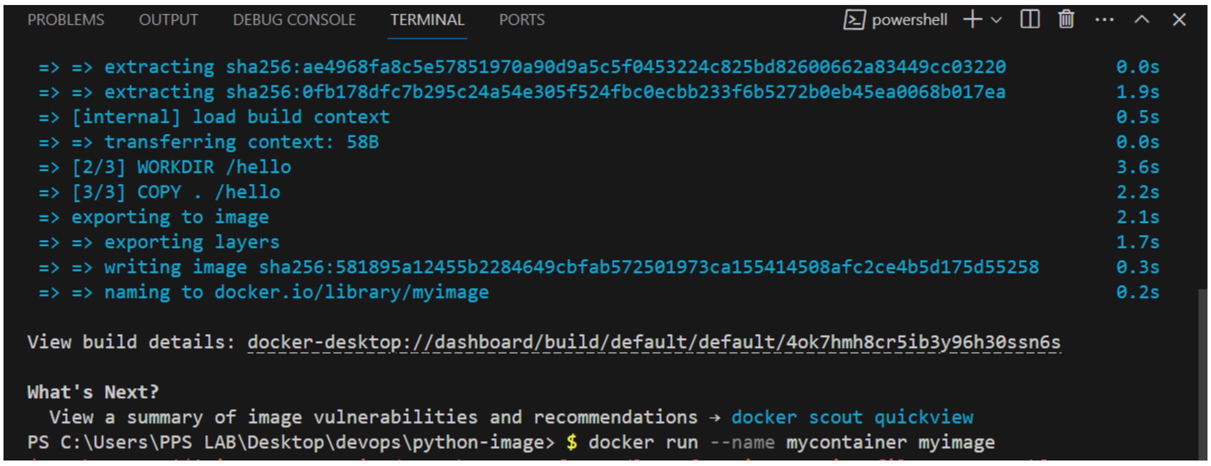
Last refresh: 14 minutes ago

Name	Tag	Status	Created	Size	Actions
myimage 581895a12455	latest	In use	4 minutes ago	996.81 MB	
okteto/daemon c72ee87c6f17	latest	In use	23 days ago	52.38 MB	

Showing 2 items



```
PS C:\Users\PPS LAB\Desktop\devops\python-image>  docker run --name mycontainer myimage
hello world
PS C:\Users\PPS LAB\Desktop\devops\python-image> docker logs mycontainer
hello world
PS C:\Users\PPS LAB\Desktop\devops\python-image> 
```



```
 => => extracting sha256:ae4968fa8c5e57851970a90d9a5c5f0453224c825bd82600662a83449cc03220        0.0s
 => => extracting sha256:0fb178dfc7b295c24a54e305f524fbc0ecbb233f6b5272b0eb45ea0068b017ea        1.9s
 => [internal] load build context                                                                0.5s
 => => transferring context: 58B                                                                 0.0s
 => [2/3] WORKDIR /hello                                                                         3.6s
 => [3/3] COPY . /hello                                                                          2.2s
 => exporting to image                                                                           2.1s
 => => exporting layers                                                                          1.7s
 => => writing image sha256:581895a12455b2284649cbfab572501973ca155414508afc2ce4b5d175d55258     0.3s
 => => naming to docker.io/library/myimage                                                       0.2s

View build details: docker-desktop://dashboard/build/default/default/4ok7hmh8cr5ib3y96h30ssn6s

What's Next?
  View a summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\Users\PPS LAB\Desktop\devops\python-image> $ docker run --name mycontainer myimage
```

EXPERIMENT NO: 8 **DATE:**

Program:- Integrate Kubernetes and Docker.

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 abstractions 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 a high-level overview of the steps to integrate Kubernetes and Docker:

- Build a Docker image:

Use Docker to build a Docker image of your application. You can use a Dockerfile to specify the base image, copy the application into the container, and specify the command to run the application.

- Push the Docker image to a registry:

Push the Docker image to a container registry, such as Docker Hub or Google Container Registry, so that it can be easily accessed by Kubernetes. Deploy the Docker image to a Kubernetes cluster:

Use Kubernetes to deploy the Docker image to a cluster. This involves creating a deployment that specifies the number of replicas and the image to be used, and creating a service that exposes the deployment to the network.

Monitor and manage the containers:

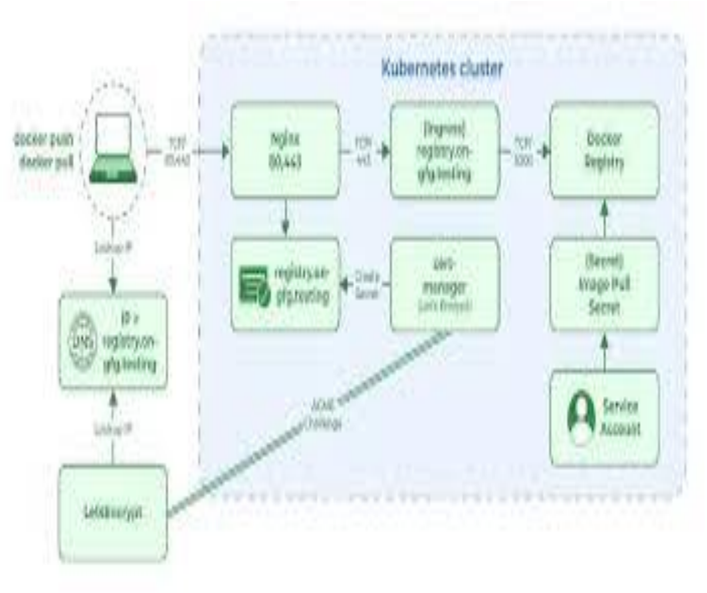
Use Kubernetes to monitor and manage the containers. This includes scaling the number of replicas, updating the image, and rolling out updates to the containers.

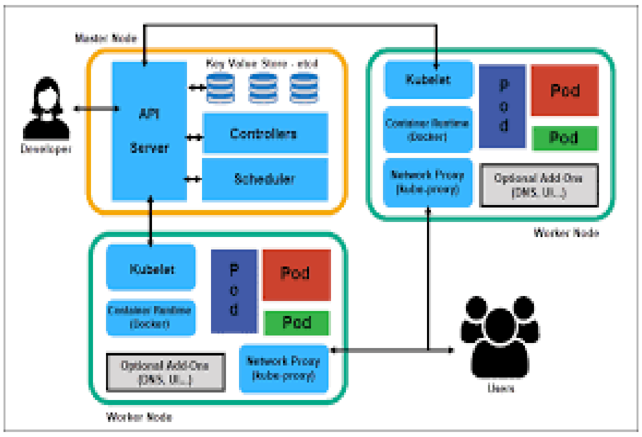
- Continuously integrate and deploy changes:

Use a continuous integration and deployment (CI/CD) pipeline to automatically build, push, and deploy changes to the Docker image and the Kubernetes cluster. This makes it easier to make updates to the application and ensures that the latest version is always running in the cluster.

By integrating Kubernetes and Docker, you can leverage the strengths of both technologies to manage containers in a scalable, reliable, and efficient manner.

Output:-


Kubernetes cluster
docker push
docker pull
TCP 80/443
Nginx 80,443
TCP 443
(Ingress) registry.on-gfg.testing
TCP 5000
Docker Registry
Lookup IP
IP > registry.on-gfg.testing
DNS
registry.on-gfg.testing
Create Secret
cert-manager (Let's Encrypt)
(Secret) Image Pull Secret
Service Account
ACME Challenge
Lookup IP
LetsEncrypt


Master Node
Key Value Store - etcd
API Server
Controllers
Scheduler
Developer
Kubelet
Container Runtime (Docker)
Network Proxy (kube-proxy)
Pod
Pod
Pod
Optional Add-Ons (DNS, UI...)
Worker Node
Kubelet
Container Runtime (Docker)
Pod
Pod
Pod
Optional Add-Ons (DNS, UI...)
Network Proxy (kube-proxy)
Worker Node
Users

EXPERIMENT NO: 9 **DATE:**

Program:- Install and Explore Selenium for automated testing.

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.

• Write a simple test:

• Once you have your IDE set up, you can write a simple test using the Selenium WebDriver. Here's an example in Java:

```
 import org.openqa.selenium.WebDriver;

 import org.openqa.selenium.chrome.ChromeDriver;

public class Main

{

public static void main(String[] args)

{

System.setProperty("webdriver.chrome.driver", "path/to/chromedriver");

WebDriver driver = new ChromeDriver();

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

 System.out.println(driver.getTitle()); driver.quit();

 }

 }
```

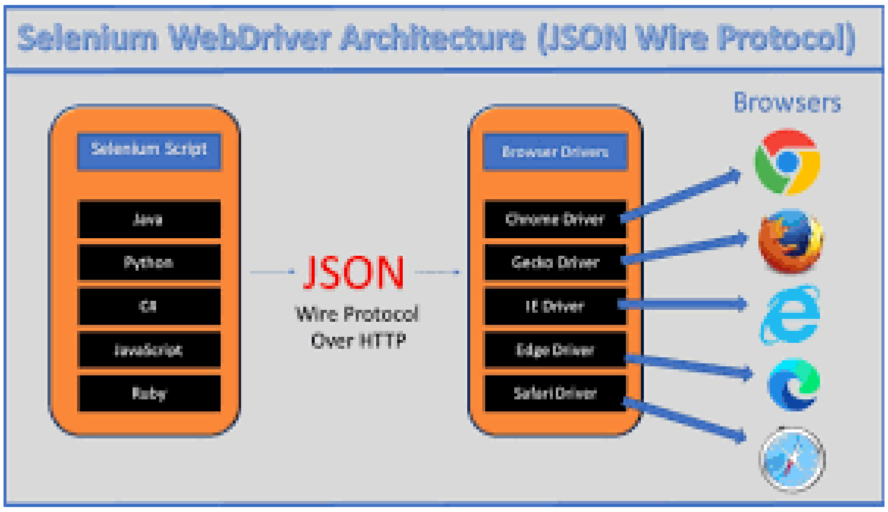
• Run the test:

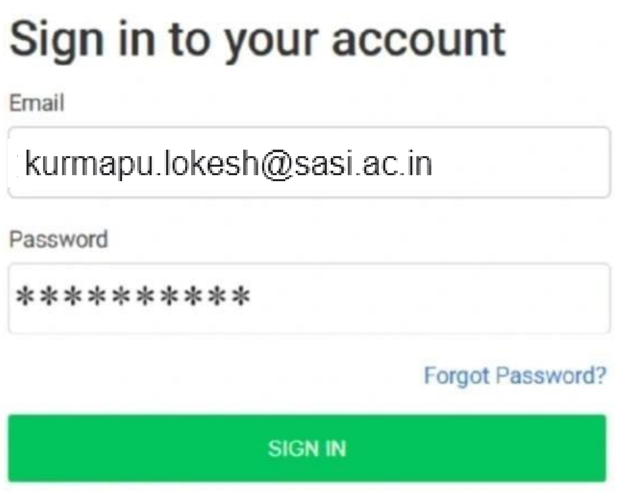
Run the test using your IDE or from the command line using the following command:

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

This is a basic example of how to get started with Selenium for automated testing. In a real-world scenario, you would likely write more complex tests and organize your code into test suites and test cases, but this example should give you a good starting point for exploring Selenium.

Output:-




Sign in to your account

Email

kurmapu.lokesh@sasi.ac.in

Password

****

EXPERIMENT NO: 10 **DATE:**

Program:- Write a simple program in JavaScript and perform testing using Selenium.

• Simple JavaScript program that you can test using Selenium

```
<!DOCTYPE >

<html>

<head>

<title>Simple Java Script Program</title>

</head>

<body>

<p id= “otput”>0</p>

<button id=”increment-button”>Increment</button>

<script>

Const output = document.getElementById(“output”);

Const increment 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
```

Output:-

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.