

EXPERIMENT NO: 1

EXPERIMENT NO: 1. Write code for a simple user registration form for an event. Aim: Write code for a simple user registration form for an event.

```
<div class="container">

<h1>Event Registration Form</h1>

<form name="Registration" class="registration-form" onsubmit="return formValidation()">

<table>

<tr>

/td>

<td><label for="name">Name:</label></td>

<td><input type="text" name="name" id="name" placeholder="yourname"></td>

</tr>

<tr>

<td><label for="email">Email:</label></td>

<td><input type="text" name="email" id="email" placeholder="youremail"></td>

</tr>

<tr>

<td><label for="password">Password:</label></td>

<td><input type="password"

id="password"></td>

</tr>

<tr>

name="password"

<td><label for="phoneNumber">PhoneNumber:</label></td>

<td><input type="text" name="phoneNumber" id="phoneNumber"></td>

</tr>

<tr>

<td><label for="gender">Gender:</label></td>

<td>Male:<input type="radio" name="gender" value="male">Female: <input type="radio"

name="gender" value="female">
```

```

Other:<input type="radio" name="gender" value="other"></td>
</tr>
<tr>
<td><label for="language">language</label></td>
<td>
<select name="language" id="language">
<option value="">Select language</option>
<option value="English">English</option>
<optionvalue="Spanish">Spanish</option>
<optionvalue="Hindi">Hindi</option>
<optionvalue="Arabic">Arabic</option>
<optionvalue="Russian">Russian</option>
</select>
</td>
</tr>
<tr>
<td><label for="zipcode">ZipCode:</label></td>
<td><input type="number" name="zipcode" id="zipcode"></td>
</tr>
<tr>
<td><label for="about">About:</label></td>
<td><textarea name="about" id="about" placeholder="Writeaboutyourself..."
></textarea></td>
</tr>
<tr>
<td colspan="2"><input type="submit" class="submit" value="Register"/></td>
</tr></table>
</form>
</div>

```

Output:

Event Registration Form

Name:	<input type="text" value="your name"/>
Email:	<input type="text" value="your email"/>
Password:	<input type="password"/>
Phone Number:	<input type="text"/>
Gender:	Male: <input type="radio"/> Female: <input type="radio"/> Other: <input type="radio"/>
language	<input type="text" value="Select language v"/>
Zip Code:	<input type="text"/>
About:	<div>Write about yourself...</div>
<input type="button" value="Register"/>	

EXPERIMENT NO:2

EXPERIMENTNO:2. Explore Git and GitHub Commands

Aim: Explore Git and GitHub commands DESCRIPTION: Git and GitHub are two of the most popular tools used for version control and collaboration in software development.

Version control is a system that records/manages changes to documents, computer programs etc over time. It helps us tracking changes when multiple people work on the same project.

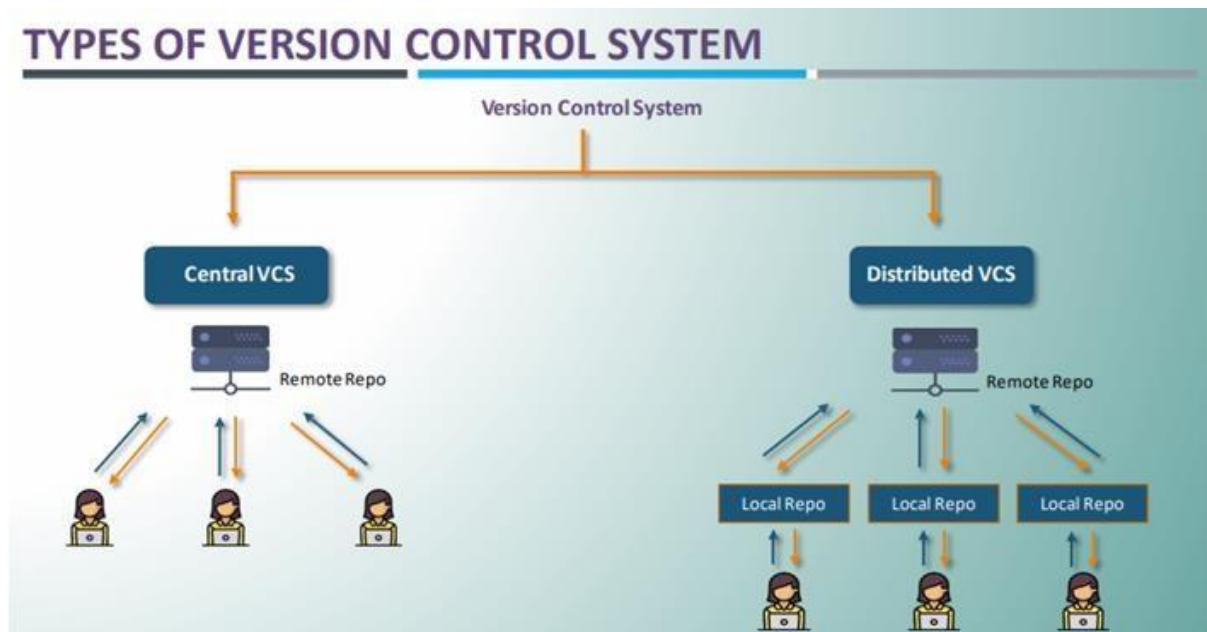
Advantages of Version Control:

Versioning is Automatic.

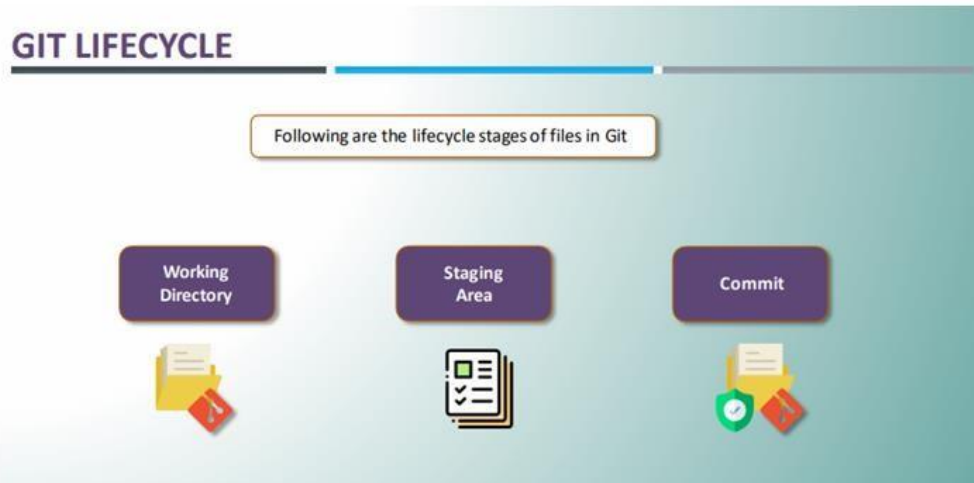
Team Collaboration is simple.

Easy Access to previous Versions.

Only modified code is stored across different versions, hence saves storage



Git is the most popular tool among all the DVCS tools. It is a version-control system for tracking changes in computer files and coordinating work on those files among multiple people. It is primarily used for source-code management in software development, but it can be used to keep track of changes in any set of files.



Working Directory: The place where your project resides in your local disk. This project may or may not be tracked by git. In either case, the directory is called the working directory. The project can be tracked by git, by using the command `git init`. By doing `git init`, it automatically creates a hidden `.git` folder.

GIT COMMANDS:

1. Configure Git

Now let Git know who you are. This is important for version control systems, as each Git commit uses this information: Command: `git config --global user.name "name"` `git config --global user.email "email Id"` EX: `git config --global user.name "John Doe"` `git config --global user.email sam@example.com`

2. Creating Git Folder:

i: `mkdir` makes a new directory.

EX: `mkdir myproject.`

ii . `cd` changes the current working directory.

EX: `cd myproject.`

3. Initialize Git:

Initializes a new Git repository in the current directory.

Command: `git init`

4. Add files:

Add one or more files to staging .

Command: git add <filename> (Or) git add *

5. It will **list** the files in the directory.

6. **Git status:**

Shows the current state of the working directory and staging area.

Command: git status

7. **Git Commit:**

Adding commits keep track of our progress and changes as we work.

Git considers each commit change point or "save point". Command: git commit -m "Commit message" EX: git commit -m "First release of Hello World!"

8. **Git Clone:**

Clones an existing repository from a URL to your local machine.

Command: git clone <https://github.com/username/repo.git>

9. **Git push:**

Pushes or sends your local commits to a remote repository.

Command: git push origin master

10. **Git pull:**

Fetches and merges changes from the main branch on origin to the current branch.

Command: git pull origin main

11. **Git branch:**

Lists all branches in the repository.

Command: git branch

12. **git branch <branch_name>**

Creates a new branch named feature.

Command: git branch <branch name>

13.git checkout:

Switches to a different branch Command:

git checkout <branchname>

14.git merge:

To merge a different branch into your active branch:

Command: git merge <branchname>

15.git log:

Displays a list of commits made in the repository's history.

Command:git log

16.git reset:

Reverts to a previous commit

Command: **git reset <file>**

EXPERIMENT NO-3

Name of the experiment : practice source code management on github
experiment with source code written in exercise 1.

Aim :To practice source code management on github.

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

1. Create a GitHub account.
2. Create a new repository on GitHub.

3. After creating account and new repository.
4. Open Git Bash. Create a directory as shown

```
MINGW64:/c/Users/hari4/Devproject2/devops
hari4@Hari MINGW64 ~ (master)
$ mkdir Devproject2
hari4@Hari MINGW64 ~ (master)
$ cd Devproject2
```

5. Now initialize the directory and configure as shown below

```
hari4@Hari MINGW64 ~/Devproject2 (master)
$ git init
Initialized empty Git repository in C:/Users/hari4/Devproject2/.git/
hari4@Hari MINGW64 ~/Devproject2 (master)
$ git config --global user.email "bhanuprasadboddu99@gmail.com"
hari4@Hari MINGW64 ~/Devproject2 (master)
$ git config --global user.name "Hari"
```

6. Clone the repository to your local machine: `$ git clone`
7. Move to the repository directory: `$ cd <repository name>`

```
hari4@Hari MINGW64 ~/Devproject2 (master)
$ git clone http://gitlab.com/Hari_123/devops.git
Cloning into 'devops'...
warning: redirecting to https://gitlab.com/Hari_123/devops.git/
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
hari4@Hari MINGW64 ~/Devproject2 (master)
$ cd devops
```

8. Create a new file in the repository and add the source code written in exercise1.

.git	14-08-2024 11:18 AM	File folder	
fib	14-08-2024 11:18 AM	Python Source File	1 KB
README	14-08-2024 11:18 AM	Markdown Source ...	7 KB
registration	13-08-2024 07:28 PM	Chrome HTML Do...	2 KB

Stage the changes: `$ git add`

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

10. Push the changes to the remote repository: \$git push origin master

```
hari4@Hari MINGW64 ~/Devproject2/devops (main)
$ git add .

hari4@Hari MINGW64 ~/Devproject2/devops (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   registration.html

hari4@Hari MINGW64 ~/Devproject2/devops (main)
$ git commit -m "registration file is added to remote repository form local repository"
[main 0d8d6b7] registration file is added to remote repository form local repository
 1 file changed, 47 insertions(+)
 create mode 100644 registration.html

hari4@Hari MINGW64 ~/Devproject2/devops (main)
$ git push origin main
warning: redirecting to https://gitlab.com/Hari_123/devops.git/
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 864 bytes | 864.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To http://gitlab.com/Hari_123/devops.git
 7ba7687..0d8d6b7  main -> main
```

Verify that the changes are reflected in the repository on GitHub. These steps demonstrate how to use GitHub for source code management.

You can use the same steps to manage any source code projects on GitHub.

EXPERIMENT NO-4

Aim:Jenkins installation and setup,explore the environment.

Install Jenkins on Windows

1. Browse to the [official Jenkins download page](#). Under the Downloading Jenkins section is a list of installers for the long-term support (LTS) version of Jenkins. Click the Windows link to begin the download.

Downloading Jenkins

Jenkins is distributed as WAR files, native packages, installers, and Docker images. Follow these installation steps:

1. Before downloading, please take a moment to review the [Hardware and Software requirements](#) section of the User Handbook.
2. Select one of the packages below and follow the download instructions.
3. Once a Jenkins package has been downloaded, proceed to the [Installing Jenkins](#) section of the User Handbook.
4. You may also want to verify the package you downloaded. [Learn more about verifying Jenkins downloads.](#)

Download Jenkins 2.303.3 LTS for:

Generic Java package (.war) SHA-256: 8a5ae73b7755c3f31a050fa945f7a3991abdb43d941c7294cac99c1e27
Docker
Ubuntu/Debian
CentOS/Fedora/Red Hat
Windows
openSUSE
FreeBSD

Download Jenkins 2.319 for:

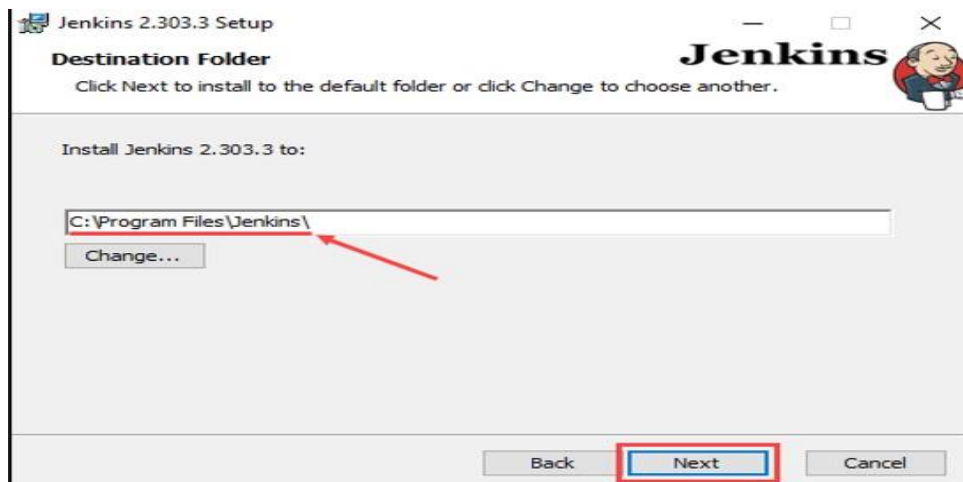
Generic Java package (.war) SHA-256: 50e9c8180da1bd3ba7e2a1e590d27a889bd527d5b0fc2d9ea944ce351c7105
Docker
Ubuntu/Debian
CentOS/Fedora/Red Hat
Windows
openSUSE
Arch Linux

2. Once the download is complete, run the **jenkins.msi** installation file.

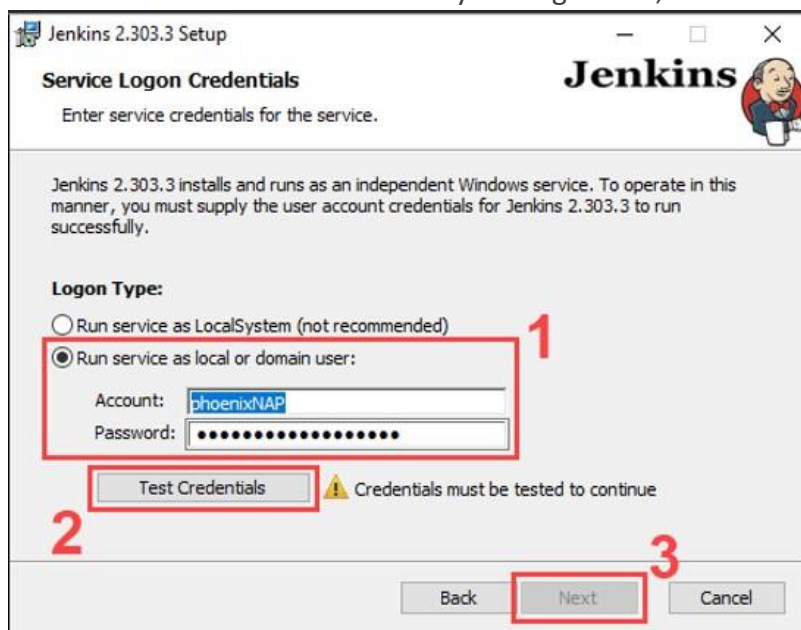
3. The setup wizard starts. Click **Next** to proceed.



4. Select the install destination folder and click **Next** to continue.



5. Under the **Run service as a local or domain user** option, enter the [domain](#) username and password for the user account you want to run Jenkins with. Click **Test Credentials** to verify the login data, then click **Next** to proceed.

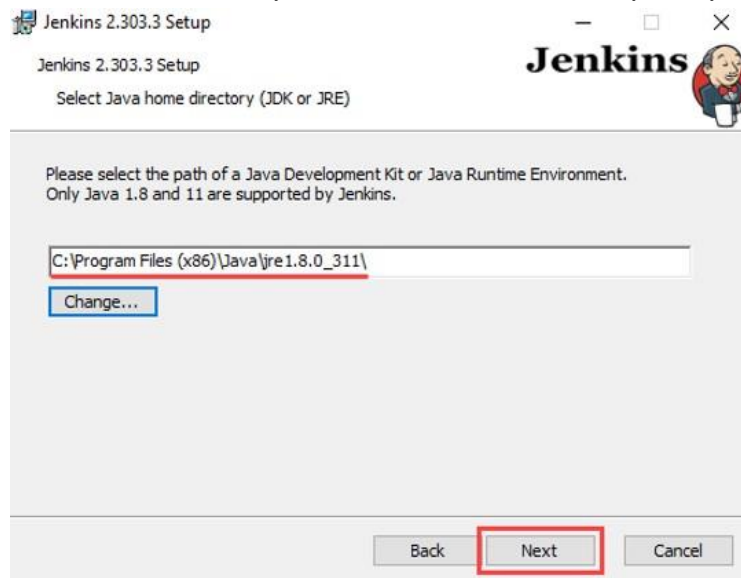


Using the **Run service as LocalSystem** option doesn't require you to enter user login data. Instead, it grants Jenkins full access to your system and services. Note that this is a less secure option and is thus not recommended.

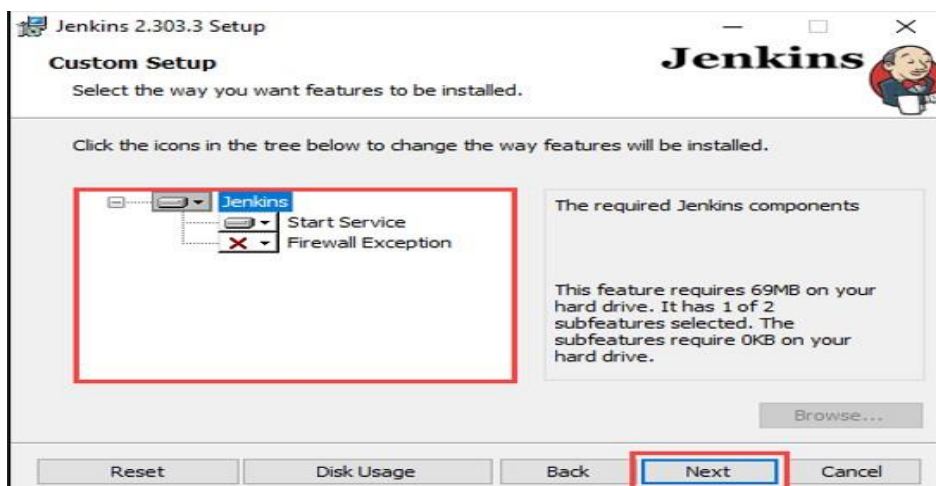
6. Enter the port number you want Jenkins to run on. Click **Test Port** to check if the selected port is available, then click **Next** to continue.



7. Select the directory where [Java is installed](#) on your system and click **Next** to proceed.



8. Select the features you want to install with Jenkins and click **Next** to continue.



1. Click

9. **Install** to start the installation process.

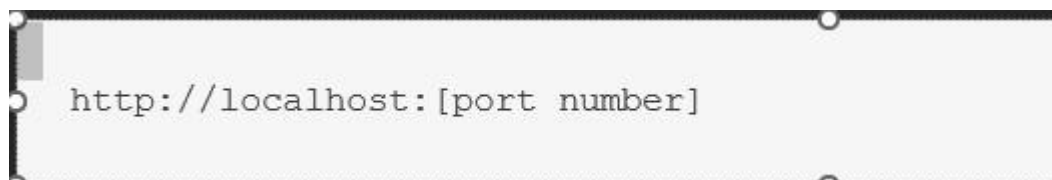


10. Once the installation is complete, click **Finish** to exit the install wizard.

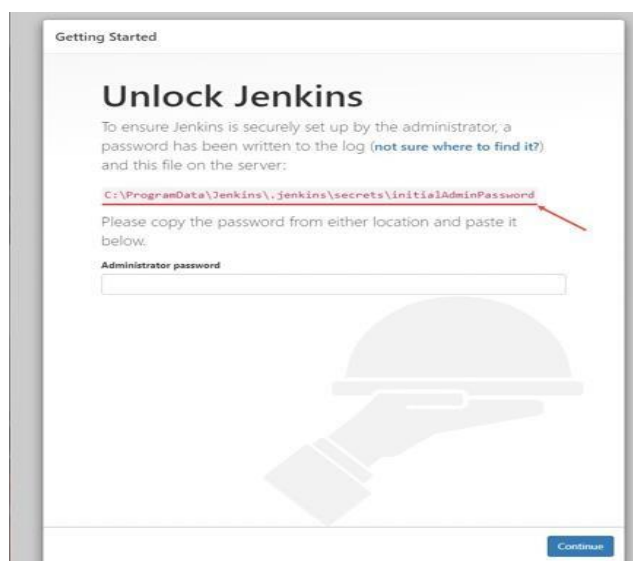
Unblock Jenkins

After completing the installation process, you have to unblock Jenkins before you can customize and start using it.

1. In your web browser, navigate to the port number you selected during the installation using the following address:



2. Navigate to the location on your system specified by the Unblock Jenkins page.



3. Open the **initialAdminPassword** file using a text editor such as Notepad.
4. Copy the password from the **initialAdminPassword** file.
5. Paste the password in the **Administrator password** field on the Unblock Jenkins page and click

Continue to proceed.

Customize Jenkins

Once you unlock Jenkins, customize and prepare the Jenkins environment.

1. Click the **Install suggested plugins** button to have Jenkins automatically install the most frequently used plugins.
2. After Jenkins finishes installing the plugins, enter the required information on the **Create First Admin User** page. Click **Save and Continue** to proceed.



Getting Started

Create First Admin User

Username:

Password:

Confirm password:

Full name:

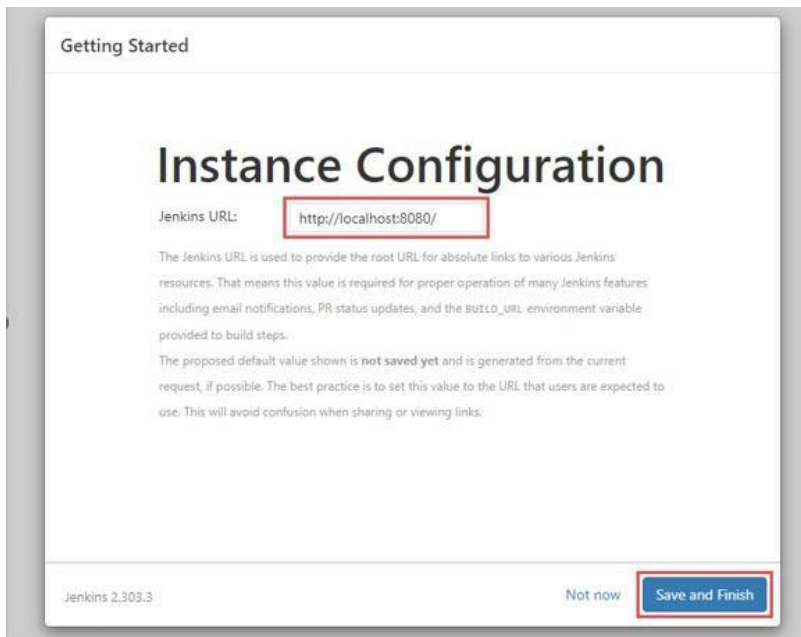
E-mail address:

Jenkins 2.303.3

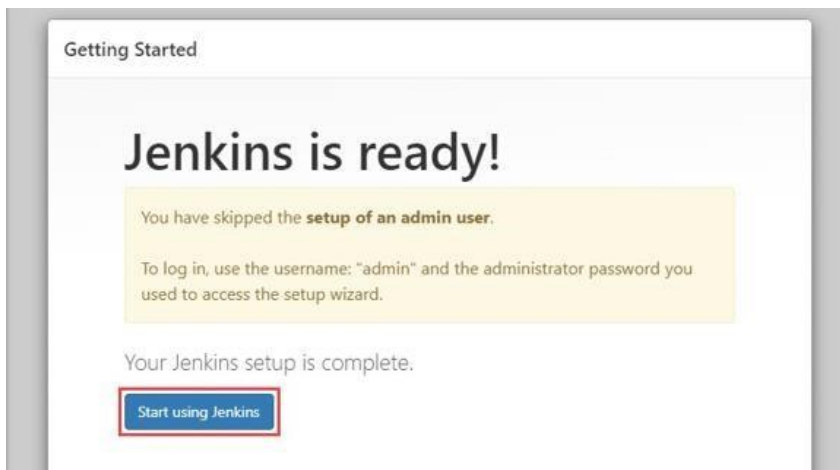
[Skip and continue as admin](#) [Save and Continue](#)

3. On the **Instance Configuration** page, confirm the port number you want Jenkins to use and click

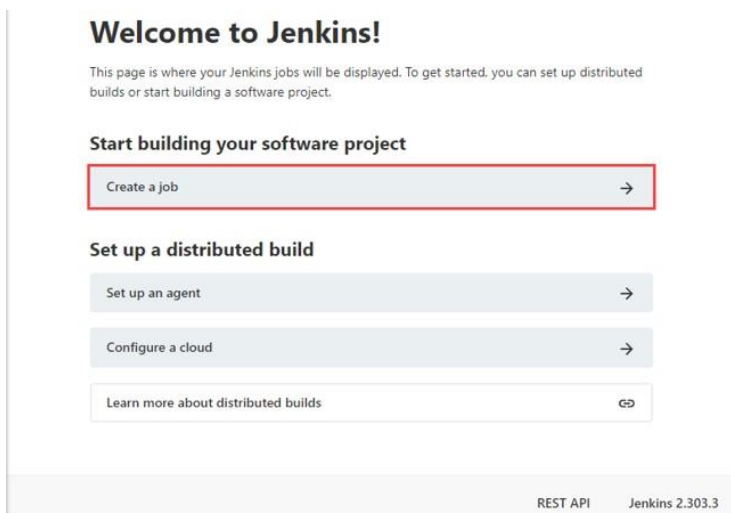
Save and Finish to finish the initial customization.



4. Click the **Start using Jenkins** button to move to the Jenkins dashboard.



5. Using the Jenkins dashboard, click **Create a job** to [build your first Jenkins software project](#)



EXPERIMENT NO-5

EXPERIMENTNO: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:
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.

Open Jenkins tool by typing localhost:8080 in a web browser

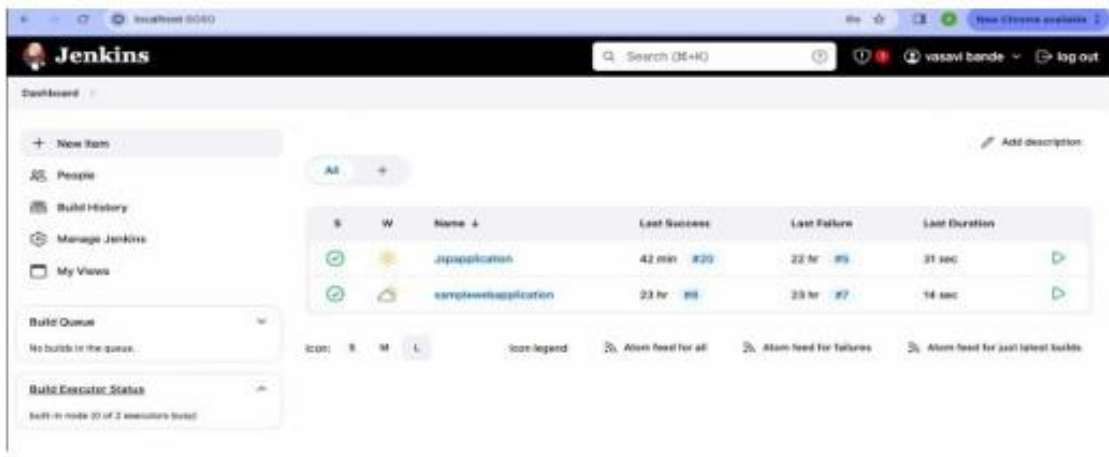


Login to your Jenkins server



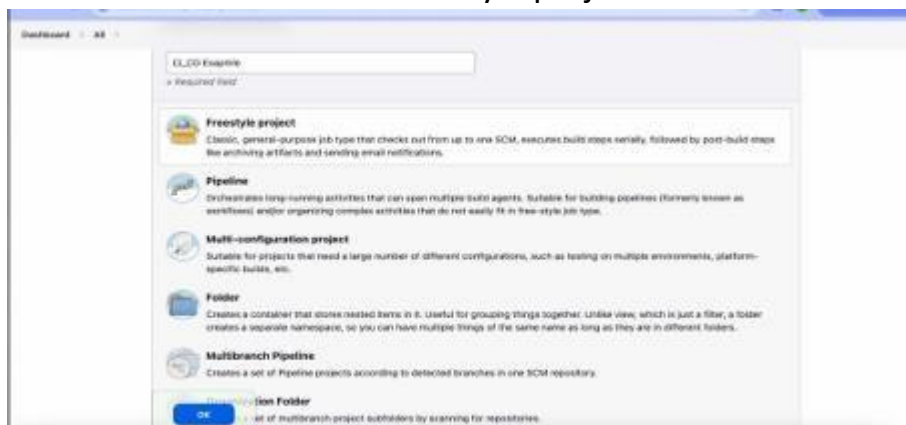
Jenkins page

Create a Jenkins job: Log in to the Jenkins web interface and create a new j



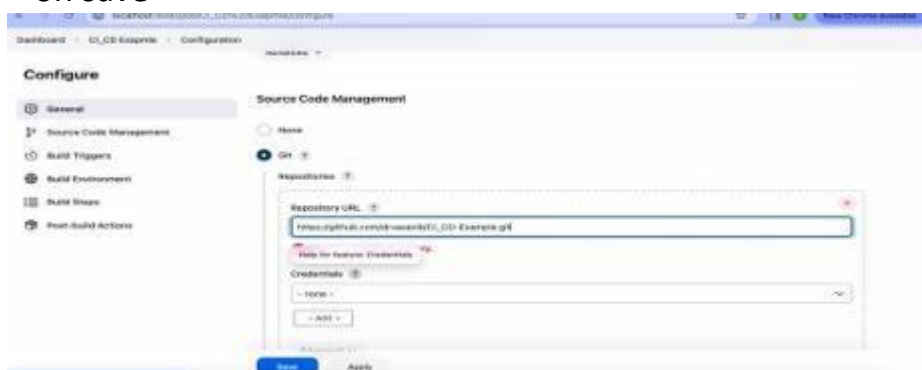
Click on Newitem

- Give a name and select freestyle project and then ok

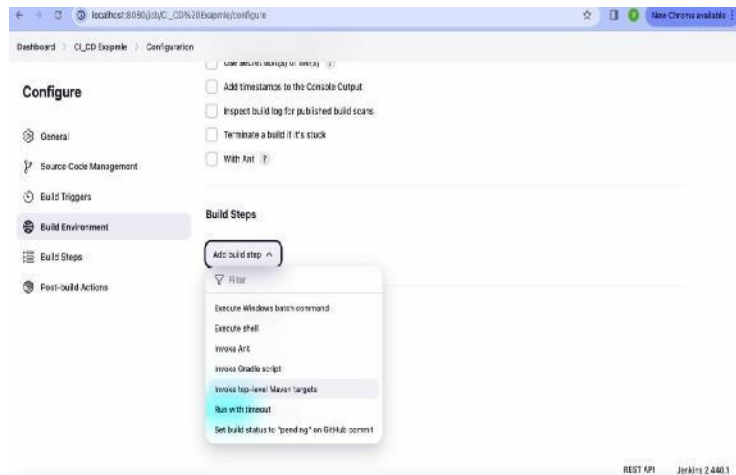


Configure the job to build the Java application from the Git repository.

- Select Git under source code management
- Give your git repository (CI_CD Example) URL in the Repository URL
- Click on save



Scroll down and goto Build steps and select Execute shell



Give shell commands as `javac filename.java` Java filename

Click on Buildnow. Check the output in console output.

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. 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

EXPERIMENT NO-6

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

AIM: Explore Docker commands for content management. **DESCRIPTION:**

Docker is a containerization technology that is widely used for managing application containers.

INSTALLING DOCKER FOR WINDOWS:

Step 1:

Download docker desktop for windows from the official docker website.com the official website docker.com we have the option

[Docker Desktop for Windows - x86_64](#).click on the above option mentioned.

Step :2

After downloading docker .exe file from official docker website run the installer and follow the on screen instructions.





Launch the docker desktop after installation

Step:3

After installation ,its time to configure and verify docker desktop .

To verify docker is installed correctly open a terminal(command prompt) and run:

`docker --version`

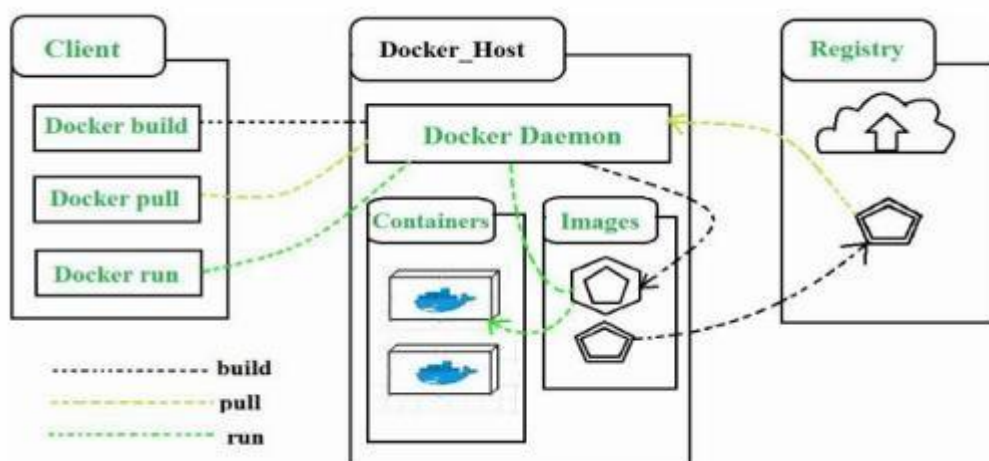
This command should displays the docker version and welcome message, indicating successful installation.

SIGN UP CREDENTIALS:

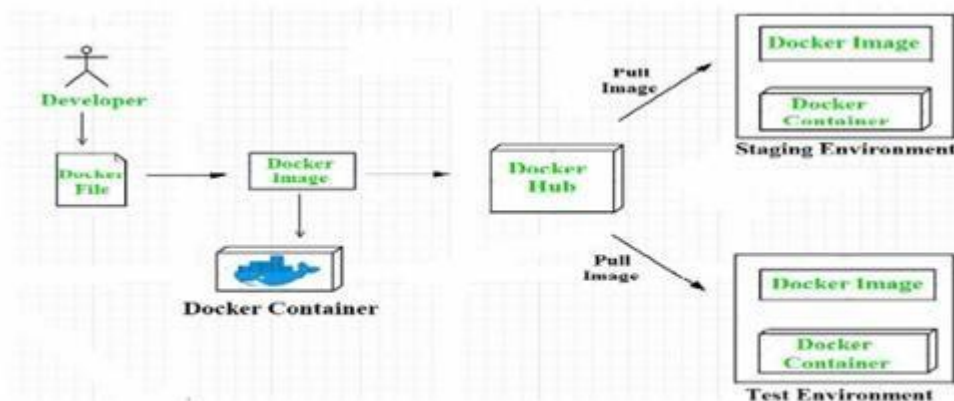
Open the docker desktop and click on sign up option and enter ypur email or userid and set up a password to sign in.

Docker Architecture:

Docker architecture consists of Docker client, Docker Daemon running on Docker Host and DockerHub repository.



Components of Docker The main components of Docker include–Docker clients and servers, Docker images, Docker file, Docker Registries, and Docker containers.



DOCKER COMMANDS:

1.DOCKER RUN COMMAND:

This command is used to run a container from an image. The docker run command is a combination of the docker create and docker start commands. It creates a new container from the image specified and starts that container. If the docker image is not present, then the docker run pulls that.

```
$ docker run <image_name>
To give name of container
$ docker run --name <container_name> <image_name>
```

2.DOCKER PULL COMMAND:

This command allows you to pull any image which is present in the official registry of docker, Docker hub. By default, it pulls the latest image, but you can also mention the version of the image.

```
$ docker pull <image_name>
```

```
C:\Users\mojha>docker pull redis
Using default tag: latest
latest: Pulling from library/redis
31b3f1ad4ce1: Pull complete
ff29a33e56fb: Pull complete
b230e0fd0bf5: Pull complete
9469c4ab3de7: Pull complete
6bd1cefcc7a5: Pull complete
610e362ffa50: Pull complete
Digest: sha256:b4e56cd71c74e379198b66b3db4dc415f42e8151a18da68d1b61f55fcc7a
Status: Downloaded newer image for redis:latest
docker.io/library/redis:latest
```

3.DOCKER PS:

This command (by default) shows us a list of all the running containers. We can use various flags with it.

- **-a flag:** shows us all the containers, stopped or running.
- **-l flag:** shows us the latest container.
- **-q flag:** shows only the Id of the containers.

```
$ docker ps [options..]
```

```
C:\Users\mojha>docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
b7e3fefc202d	ubuntu	"bash"	11 hours ago	Up 11 hours	
1b470557a372	ubuntu	"sleep 1000000"	11 hours ago	Up 11 hours	
9088b39c68dd	docker/getting-started	"/docker-entrypoint...."	11 hours ago	Up 11 hours	0

4.Docker Stop

This command allows you to stop a container if it has crashed or you want to switch to another one.

```
$ docker stop <container_ID>
```

```
C:\Users\mojha>docker stop b7e3fefc202d
b7e3fefc202d
```

EXPERIMENT NO.: 7.

Develop a simple containerized application using Docker

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

1.CHOOSE AN APPLICATION:

- **Choose** a simple application that you want to containerize. We will be working with a simple todo list manager that runs on Node.js.
- Before you can run the application ,you need to get the application source code into your machine.
- To do that,create a new folder and name “dockerdev”.
- Open the folder in git bash.

- Next ,clone the getting-started-app repository using the following command:
- \$git clone <https://github.com/docker/getting-started-app.git>
- View the contents of the cloned repository.you should have the following files and sub-directories.

```

|— getting-started-app/
| |— .dockerignore
| |— package.json
| |— README.md
| |— spec/
| |— src/
| |— yarn.lock

```

2.BUILD APP'S IMAGE:

- To build the image, you'll need to use a Dockerfile. A Dockerfile is simply a text-based file with no file extension that contains a script of instructions. Docker uses this script to build a container image.
1. In the getting-started-app directory, the same location as the package.json file, create a file named Dockerfile with the following contents:

```
# syntax=docker/dockerfile:1
```

```
FROM node:lts-alpine
```

```
WORKDIR /app
```

```
COPY . .
```

```
RUN yarn install --production
```

```
CMD ["node", "src/index.js"]
```

```
EXPOSE 3000
```

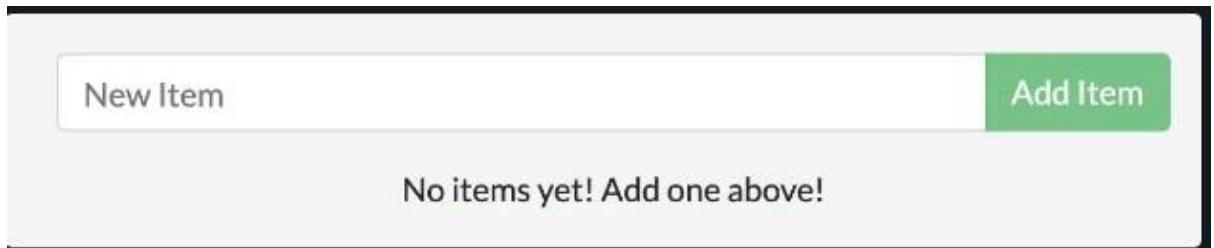
To build you need to use the command in the terminal:

```
$cd getting-started-app
```

```
$docker build t-getting-started-app .
```

3.START AN APP CONTAINER:

- you can run the application in a container using the docker run command
- `$docker run -d -p 127.0.0.1:3000:3000 getting-started`
- After a few seconds, open your web browser to <http://localhost:3000>. You should see your app.



4.UPDATE THE APPLICATION:

****Update the source code:**

1.The following steps are used to update the source code,you will change the “empty text” Go to following path file:

getting-started-app/src/static/js/app.js and open it in vscode editor.

2.Update line 56 to use the new empty text.

3.Now ,we should build updated version we need to stop and remove old container.

***Remove a container using CLI:**

4.Get the ID of the container by using docker ps command

```
$docker ps
```

5.Use the docker stop command to stop the container.replace the <container -id> with the id from docker ps

```
$docker stop <the container-id>
```

6.Once the container has stopped ,you can remove it by using the docker rm command.

```
$docker rm <the container_id>
```

7.Now start your updated app using the docker run command.

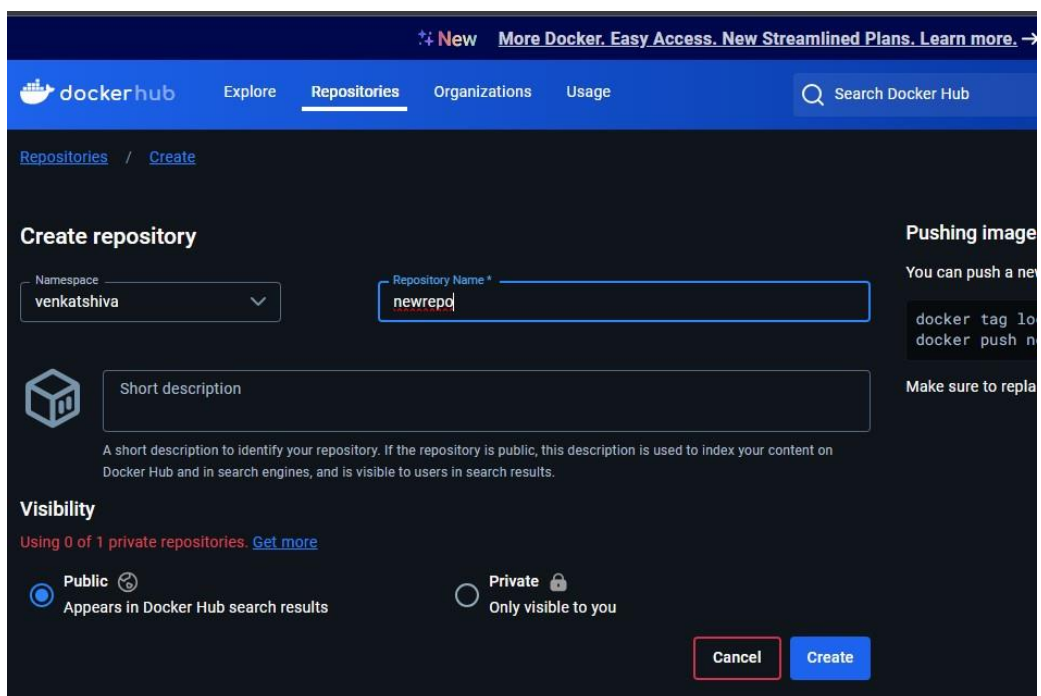
```
$docker run -dp 127.0.0.1:3000:3000 getting-started .
```


5.SHARE THE APPLICATION:

Create a repository:

To push an image, you first need to create a repository on Docker Hub.

1. [Sign up](#) or Sign in to [Docker Hub](#).
2. Select the **Create Repository** button.
3. For the repository name, use getting-started. Make sure the **Visibility** is **Public**.



The screenshot shows the Docker Hub 'Create repository' page. At the top, there's a navigation bar with 'dockerhub' logo and links for 'Explore', 'Repositories', 'Organizations', and 'Usage'. A search bar is also present. Below the navigation bar, the page title is 'Create repository'. The form includes a 'Namespace' dropdown menu set to 'venkatshiva', a 'Repository Name' text input field containing 'newrepo', and a 'Short description' text area. Below the description, there's a note about its purpose. The 'Visibility' section shows two options: 'Public' (selected with a radio button) and 'Private'. The 'Public' option is described as 'Appears in Docker Hub search results', while the 'Private' option is 'Only visible to you'. At the bottom right, there are 'Cancel' and 'Create' buttons.

4. Select **Create**.
5. Use the docker tag command to give the getting-started image a new name. Replace YOUR-USER-NAME with your Docker ID.

```
MINGW64:/c/Users/Dell/Desktop/docker/getting-started-app
Dell@DESKTOP-DU76PCK MINGW64 ~/Desktop/docker/getting-started-app (main)
$ docker tag getting-started rpranav09/getting-started

Dell@DESKTOP-DU76PCK MINGW64 ~/Desktop/docker/getting-started-app (main)
$ docker push rpranav09/getting-started
Using default tag: latest
The push refers to repository [docker.io/rpranav09/getting-started]
027a5805b4b1: Waiting
b903ba0882df: Waiting
d9f81725f57f: Waiting
da9db072f522: Waiting
2c683692384c: Waiting
7ae31c3108a2: Waiting
7e446ce5108f: Waiting
e7428cb4ffb3: Waiting
e7428cb4ffb3: Waiting
027a5805b4b1: Waiting
b903ba0882df: Waiting
```

\$docker tag getting-started YOUR-USER-NAME/getting-started

Example:\$docker tag getting-started xyz/newrepo

6.Now run the docker push command

\$docker push YOUR-USER-NAME/getting-started

Example :\$docker push xyz/newrepo

7.The image is pushed to the repository on docker.

EXPERIMENT NO-8

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 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.

EXPERIMENT NO-9

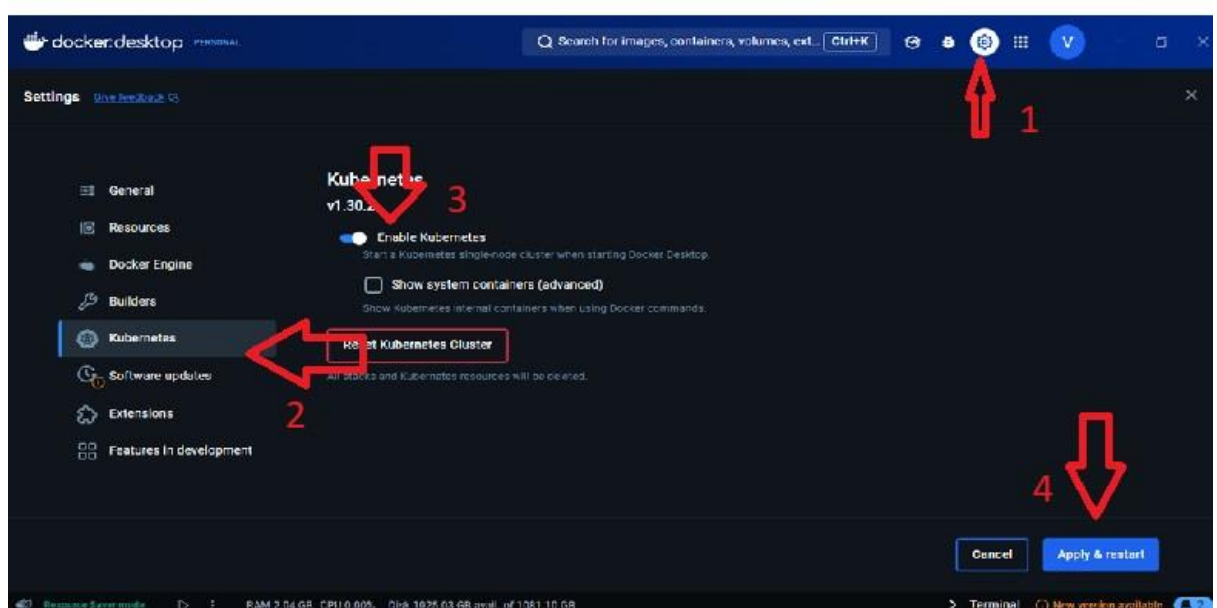
AIM: Automate the process of running containerized application developed in exercise 7 using Kubernetes

DESCRIPTION : To automate the process of running the containerized application developed in exercise 7 using Kubernetes, follow the below mentioned steps:

CREATE A KUBERNETES CLUSTER:

Open docker desktop in your local machine. On the right-side top we have settings, click on it.

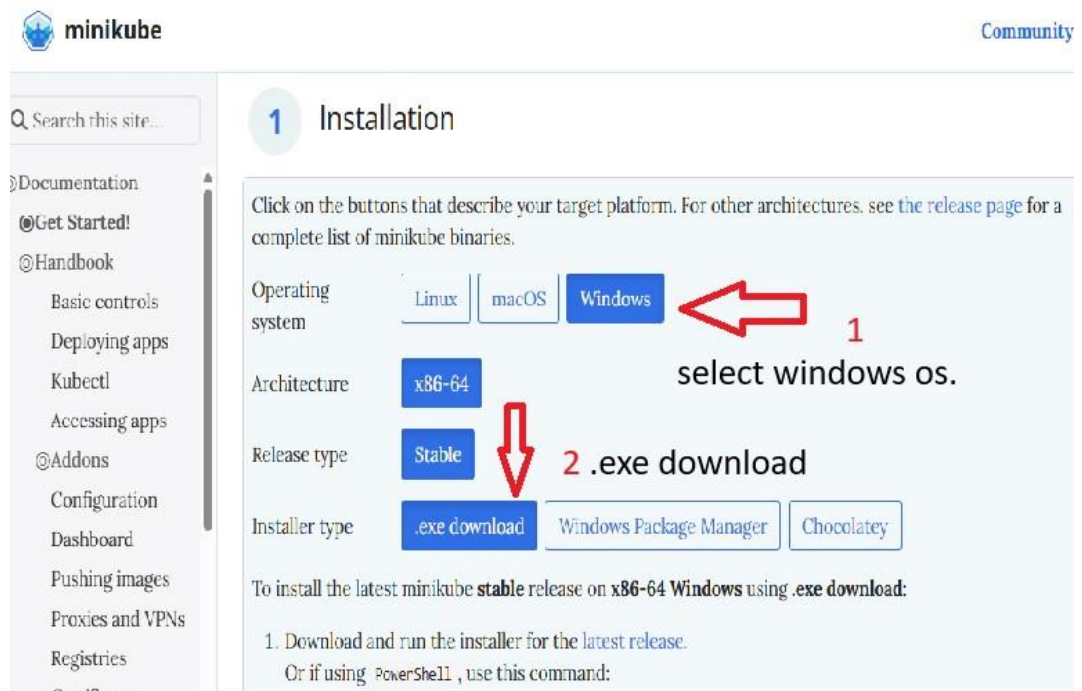
Next, select Kubernetes then click on enable Kubernetes and click on apply and restart.



Kubernetes cluster will be created.

MINIKUBE INSTALLATION:

- Visit the Minikube download page: Go to the Minikube download page.
- **Select your operating system:** Choose the appropriate version for your OS (Linux, macOS, or Windows)



- **Download the installer:** Click on the download link for your selected OS.
- **Install Minikube:** Once the download is complete, open the installer and follow the on-screen instructions to install Minikube on your system.

Containerize the application using Kubernetes:

Push the Docker image to a registry:

Push the Docker image of your application to a container registry, such as Docker Hub or Google Container Registry.

Create a deployment: Create a deployment in Kubernetes that specifies the number of replicas and the Docker image to use. Here's example of a deployment YAML file:

```

Yaml
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
          image: <image_name>
          ports:
            - containerPort: 8080

```

The deployment.yaml need to be stored in the getting-started-app folder which was created in the build process.

```

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
          image: <image_name>
          ports:
            - containerPort: 8080

```

Create a service: Create a service in Kubernetes that exposes the deployment to the network. Here's an example of a service YAML file:

The service.yaml need to be stored in the same folder where deployment.yaml is created.

```
Yaml
apiVersion: v1
kind: Service
metadata:
  name: my-app-service
spec:
  selector:
    app: my-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8080
  type: LoadBalancer
```

Service yaml file:

apiVersion: v1

kind: Service

metadata:

name: my-app-service

spec: selector:

app: my-app

ports:

- protocol: TCP

port: 80

targetPort: 8080

type: LoadBalancer

- Then open the terminal on the docker desktop at the bottom right-side.
- Apply the deployment and service to the cluster:
- Apply the deployment and service to the cluster using the kubectl command line tool.

For example:

○ \$ kubectl apply -f deployment.yaml

```
SAI KOUSHIK@DESKTOP-MHFJI2P MINGW64 ~/Desktop/docker/getting-started-app (main)
$ kubectl apply -f bb.yaml
deployment.apps/bb-demo created
service/bb-entrypoint created
```

- \$ kubectl apply -f service.yaml

Verify Deployment:

- Check the status of your deployment using :

-> kubectl get deployments

```
SAI KOUSHIK@DESKTOP-MHFJI2P MINGW64 ~/Desktop/docker/getting-started-app (main)
$ kubectl get deployments
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
bb-demo   1/1     1            1           71s
```

- Verify that pods and service are running using :

-> kubectl get pods

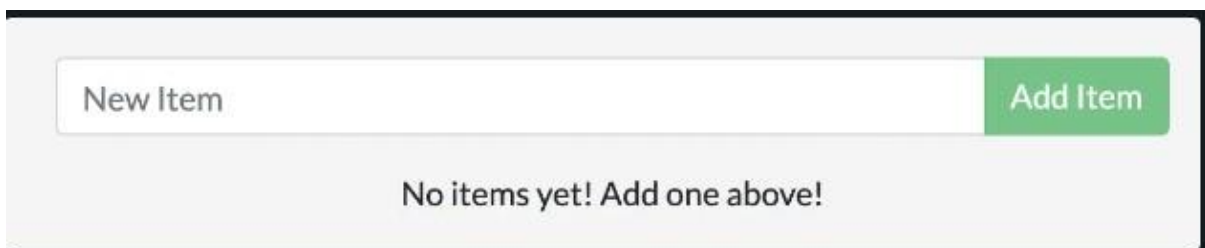
-> kubectl get services

```
SAI KOUSHIK@DESKTOP-MHFJI2P MINGW64 ~/Desktop/docker/getting-started-app (main)
$ kubectl get services
NAME            TYPE        CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
bb-entrypoint   NodePort    10.107.181.249 <none>         3000:30001/TCP   92s
kubernetes      ClusterIP   10.96.0.1     <none>         443/TCP          22h
```

- to view your application in a web browser using the port number:
- **Open Your Web Browser:** Launch any web browser like Chrome, Firefox, or Edge.
- **Enter the URL:** In the address bar, type `http://localhost:[port_number]` and press Enter. Replace `[port_number]` with the actual port number of your application.

For example, if your application is running on port 8080, you would type:

-> <http://localhost:8080>



The screenshot shows a web application interface. At the top, there is a light gray bar containing a text input field labeled 'New Item' and a green button labeled 'Add Item'. Below this bar, the main content area is light gray and displays the text 'No items yet! Add one above!' in a dark gray font.

EXPERIMENT-10

10. Install and Explore Selenium for automated testing.

AIM: Install and Explore Selenium for automated testing.

DESCRIPTION: Selenium is an open-source framework used for automating web browser interactions.

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

1. INSTALL JAVA JDK:

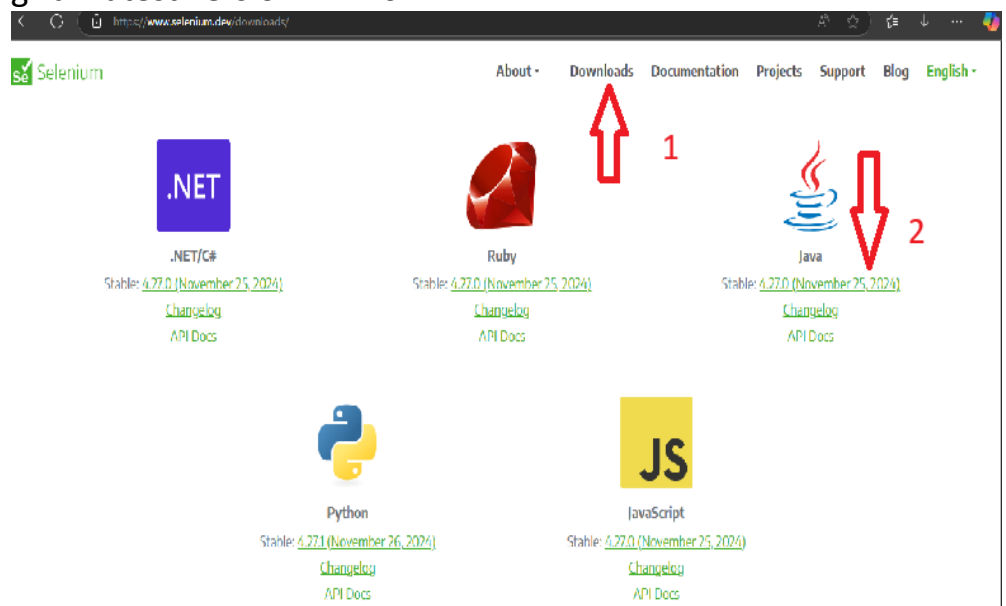
Go to the oracle jdk download page. Follow the installation instructions for your operating system.

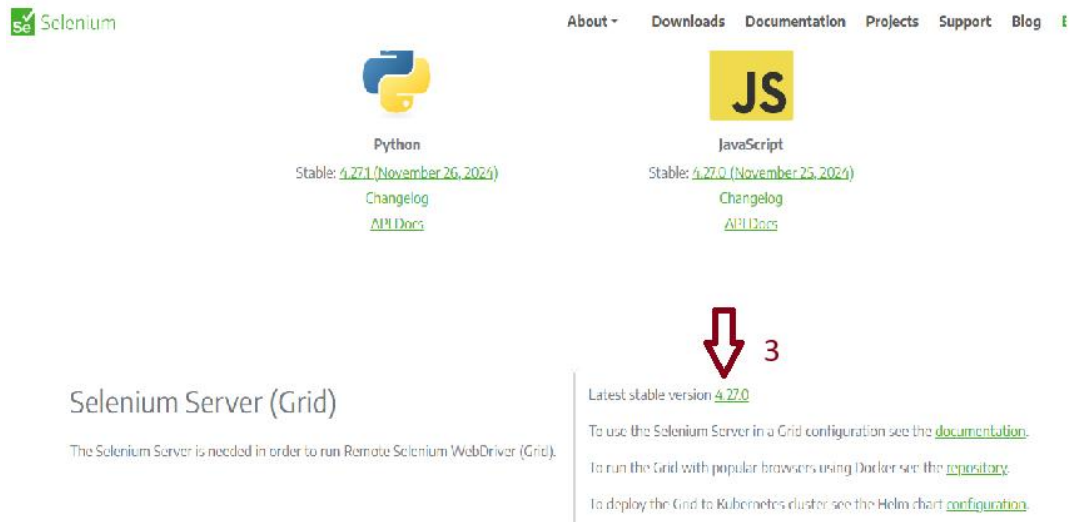
2. INSTALL ECLIPSE IDE:

Download the eclipse ide for java developers from the eclipse website.

3. INSTALL SELENIUM WEBDRIVER:

- Go to selenium.dev website and download option on the home task bar is visible, click on it.
- Then download selenium java zip file of latest version stable:4.27.0
- Next on the same website ,download selenium server grid. Latest version 4.27.0



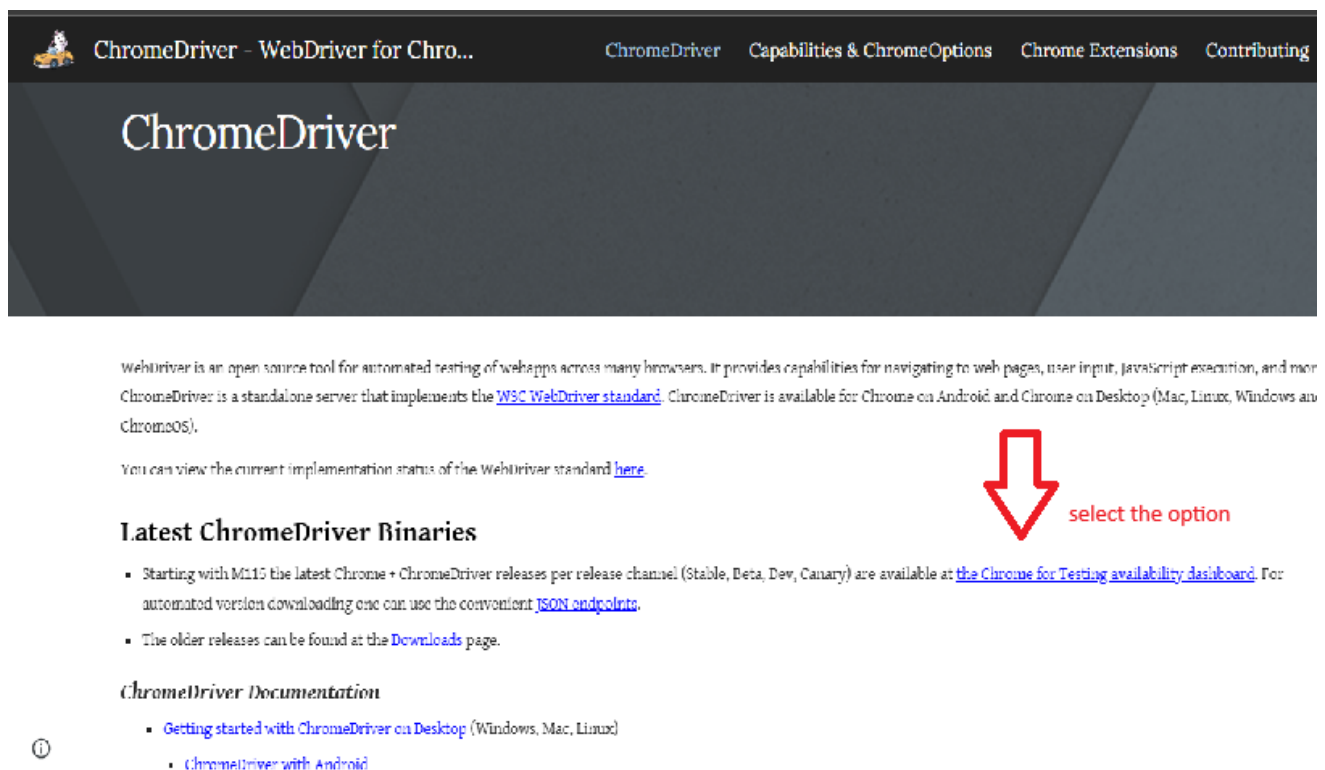


4.DOWNLOAD A BROWSER DRIVE:

If we are using chrome:

Download ChromeDriver:

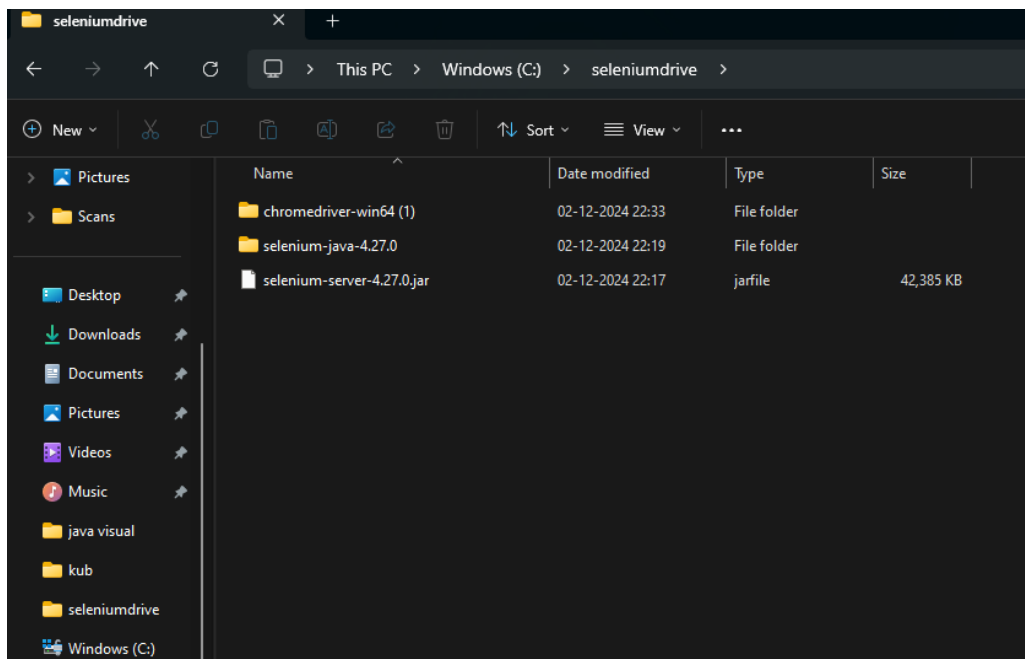
- Go to the ChromeDriver download page.
- Next click on the “testing availability dashboard”.



- Then select the chromedriver version your windows requires and download the chromedriver.

chrome	win32	https://storage.googleapis.com/chrome-for-testing-public/131.0.6778.85/win32/chrome-win32.zip
chrome	win64	https://storage.googleapis.com/chrome-for-testing-public/131.0.6778.85/win64/chrome-win64.zip
chromedriver	linux64	https://storage.googleapis.com/chrome-for-testing-public/131.0.6778.85/linux64/chromedriver-linux64.zip
chromedriver	mac-arm64	https://storage.googleapis.com/chrome-for-testing-public/131.0.6778.85/mac-arm64/chromedriver-mac-arm64.zip
chromedriver	mac-x64	https://storage.googleapis.com/chrome-for-testing-public/131.0.6778.85/mac-x64/chromedriver-mac-x64.zip
chromedriver	win32	https://storage.googleapis.com/chrome-for-testing-public/131.0.6778.85/win32/chromedriver-win32.zip
chromedriver	win64	https://storage.googleapis.com/chrome-for-testing-public/131.0.6778.85/win64/chromedriver-win64.zip
chrome-headless-shell	linux64	https://storage.googleapis.com/chrome-for-testing-public/131.0.6778.85/linux64/chrome-headless-shell-linux64.zip
chrome-headless-shell	mac-arm64	https://storage.googleapis.com/chrome-for-testing-public/131.0.6778.85/mac-arm64/chrome-headless-shell-mac-arm64.zip

- After downloading all the prerequisites, now create a folder in your C:/ folder and name it as seleniumDriver.
- Now the selenium web driver zip file which was downloaded should be extracted and copy the extracted file to the seleniumDriver folder created before.
- Next,ChromeDriver zip file should extracted same as selenium java and copied to folder created in c:drive



CONFIGURE YOUR PROJECT:

Open eclipse ide and create a new java project and create a package and class to the file.

Add selenium libraries:

- In eclipse, right click on your project, select build path>configure build path>libraries tab>add external jars and select selenium server and select apply and close.
- After successfully adding JAR files from the selenium directory to the project, now we can write your first selenium test script:

Simple test script:

```
package testing;
```

```
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://sriindu.ac.in/");
```

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

```
driver.quit();
```

```
}
```

```
}
```

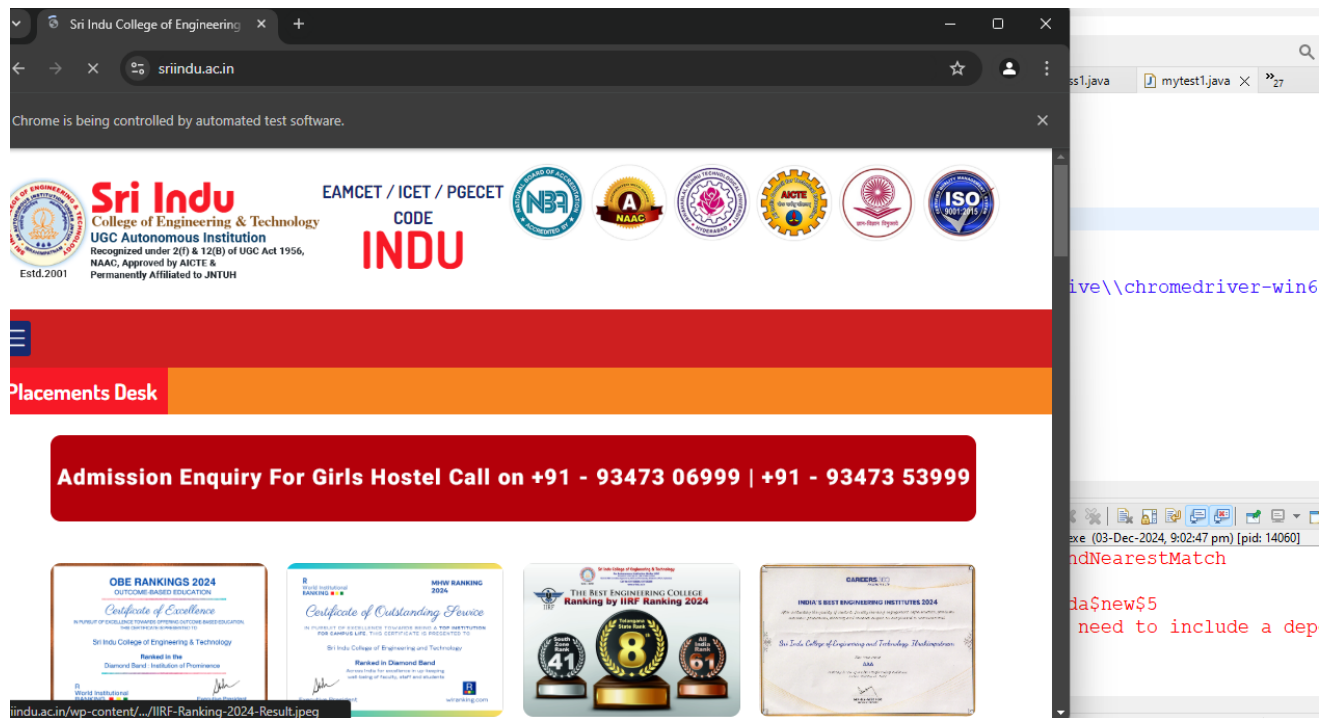
Test script:

```
1 package testing;  
2 import org.openqa.selenium.chrome.ChromeDriver;  
3 import org.openqa.selenium.WebDriver;  
4  
5  
6 public class mytest1 {  
7  
8     public static void main(String[] args) {  
9         System.setProperty("webdriver.chrome.driver", "C:\\seleniumdrive\\chromedriver-win64  
10         WebDriver driver= new ChromeDriver();  
11         driver.get("https://sriindu.ac.in/");  
12         System.out.println(driver.getTitle());  
13         driver.quit();  
14     }  
15 }  
16  
17 }
```

RUN YOUR TEST SCRIPT:

- Right click on your java file in eclipse and select run as and next click java application.
- Your browser should open and navigate to the specified URL.

OUTPUT:



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

EXPERIMENT NO-11

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:

PROGRAM: 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