CMR COLLEGE OF ENGINEERING & TECHNOLOGY
An Autonomous institution
*Approved by AICTE *Permanently Affiliated to JNTUH *NBA Accreditation
Approved by NAAC Accreditation (A+-Grade)
Kandlakoya (V), Medchal Road, Hyderabad -S01401

(A405511) DEVOPS LABORATORY

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.

WEEK -1

Write Code for a Simple User Registration Form for an Event

To develop a basic user registration form using HTML and JavaScript, allowing users to submit their details for event registration.

Software Requirements:

- Code editor (VS Code or any)
- Web browser (Chrome, Firefox, etc.)

Lab Procedure:

1. Create HTML File for User Registration Form

- Create a file named registration.html.
- Add the following code:

```
<!DOCTYPE html>
<html>
<head>
  <title>User Registration Form</title>
  <script src="script.js"></script>
</head>
<body>
  <h2>Event Registration Form</h2>
  <form id="registrationForm">
    <label for="name">Name:</label>
    <input type="text" id="name" name="name" required><br><br>

    <label for="email">Email:</label>
    <input type="email" id="email" name="email" required><br><br>

    <label for="phone">Phone Number:</label>
    <input type="text" id="phone" name="phone" required><br><br>

    <button type="button" onclick="submitForm()">Register</button>
  </form>

  <p id="confirmation"></p>
</body>
</html>
```

2. Create JavaScript File for Form Handling

- Create a file named script.js.
- Add the following code:

```
function submitForm() {
  var name = document.getElementById('name').value;
  var email = document.getElementById('email').value;
  var phone = document.getElementById('phone').value;

  if (name && email && phone) {
    document.getElementById('confirmation').innerText =
      'Thank you, ' + name + '! You have registered successfully.';
    console.log('Registration Details:', { Name: name, Email: email, Phone: phone });
  } else {
    alert('Please fill all the fields.');
  }
}
```

---

3. Run and Test

- Open the registration.html file in a web browser.
- Fill in user details and click **Register**.
- Verify registration confirmation message is displayed.

---

4. Optional Enhancements

- Validate email and phone formats using JavaScript.
- Store registration data in local storage or backend.
- Add CSS for improved design.

---

5. Summary of Steps

- Create HTML form with input fields.
- Use JavaScript to handle form submission and validations.
- Display registration confirmation to the user.

OUTPUT:

Event Registration Form

Name: XYZABC

Email: XYZABC@gmail.com

Phone Number: 09861043010

Register

Thank you, XYZABC! You have registered successfully.

WEEK - 2 & WEEK -3

Explore Git and GitHub commands.

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

To explore and practice basic **Git** and **GitHub** commands for version control in a collaborative environment.

Software Requirements:

- Git installed (https://git-scm.com/downloads)
- GitHub account (https://github.com/)
- Code editor (VS Code or any)

Lab Procedure:

1. Configure Git (First Time Setup)

```
git config --global user.name "Your Name"
git config --global user.email "your-email@example.com"
git config --list
```

---

2. Initialize Local Repository

```
mkdir devops-lab
cd devops-lab
git init
```

---

3. Add Files and Commit Changes

```
echo "Hello DevOps" > readme.md
git status
git add readme.md
git commit -m "Initial commit: added readme"
```

---

4. View Commit History

```
git log
git log --oneline
```

---

5. Create and Manage Branches

```
git branch feature-1
git checkout feature-1
# or using
git switch feature-1
```

---

6. Update Files and Merge Branch

```
echo "Adding feature work" >> readme.md
git add .
git commit -m "Updated readme in feature-1 branch"

git checkout main
git merge feature-1
```

---

7. Push to GitHub

1. Create a repository on GitHub.
2. Connect local repo to GitHub:

```
git remote add origin https://github.com/your-username/your-repo-name.git
git branch -M main
git push -u origin main
```

---

8. Clone Repository

```
git clone https://github.com/your-username/your-repo-name.git
```

---

9. Pull Latest Changes

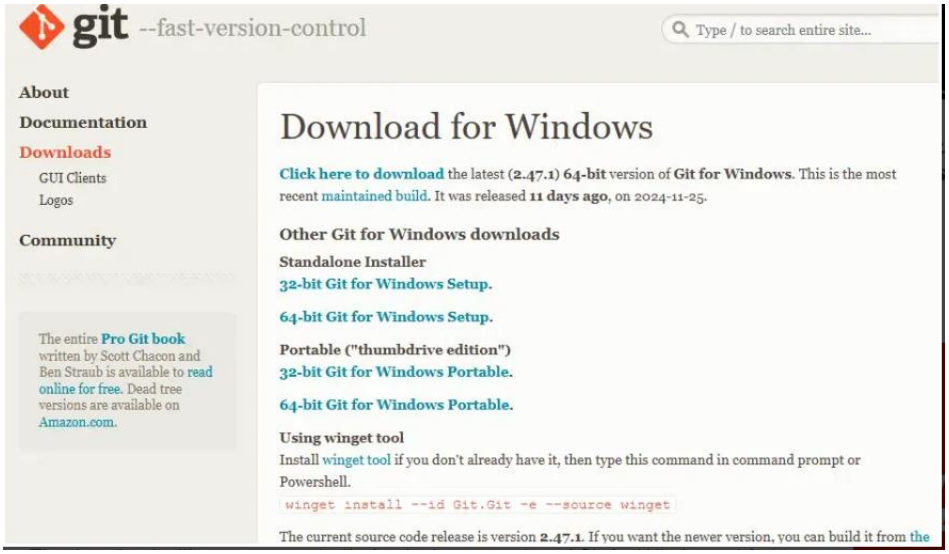
```
git pull origin main
```

---

10. Connect to GitHub and Push Code

git remote add origin https://github.com/your-username/your-repo-name.git
git branch -M main
git push -u origin main

OUTPUT:




```
C:\Users\GFG0536>git --version
git version 2.42.0.windows.1
```

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner * **Repository name** *

dfryer1193 / newrepo

Great repository names are short and memorable. Need inspiration? How about jubilant-memory?

Description (optional)

My new repo!

Public
Anyone on the internet can see this repository. You choose who can commit.

Private
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

Add a README file
This is where you can write a long description for your project. Learn more.

Add .gitignore
Choose which files not to track from a list of templates. Learn more.

Choose a license
A license tells others what they can and can't do with your code. Learn more.

Create repository

```
mnelson:Desktop mnelson$ cd ~/Desktop
mnelson:Desktop mnelson$ mkdir myproject
mnelson:Desktop mnelson$ cd myproject/
```

```
mnelson:myproject mnelson$ git init
Initialized empty Git repository in /Users/mnelson/Desktop/myproject/.git/
```

```
mnelson:myproject mnelson$ git status
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        mnelson.txt

nothing added to commit but untracked files present (use "git add" to track)
```

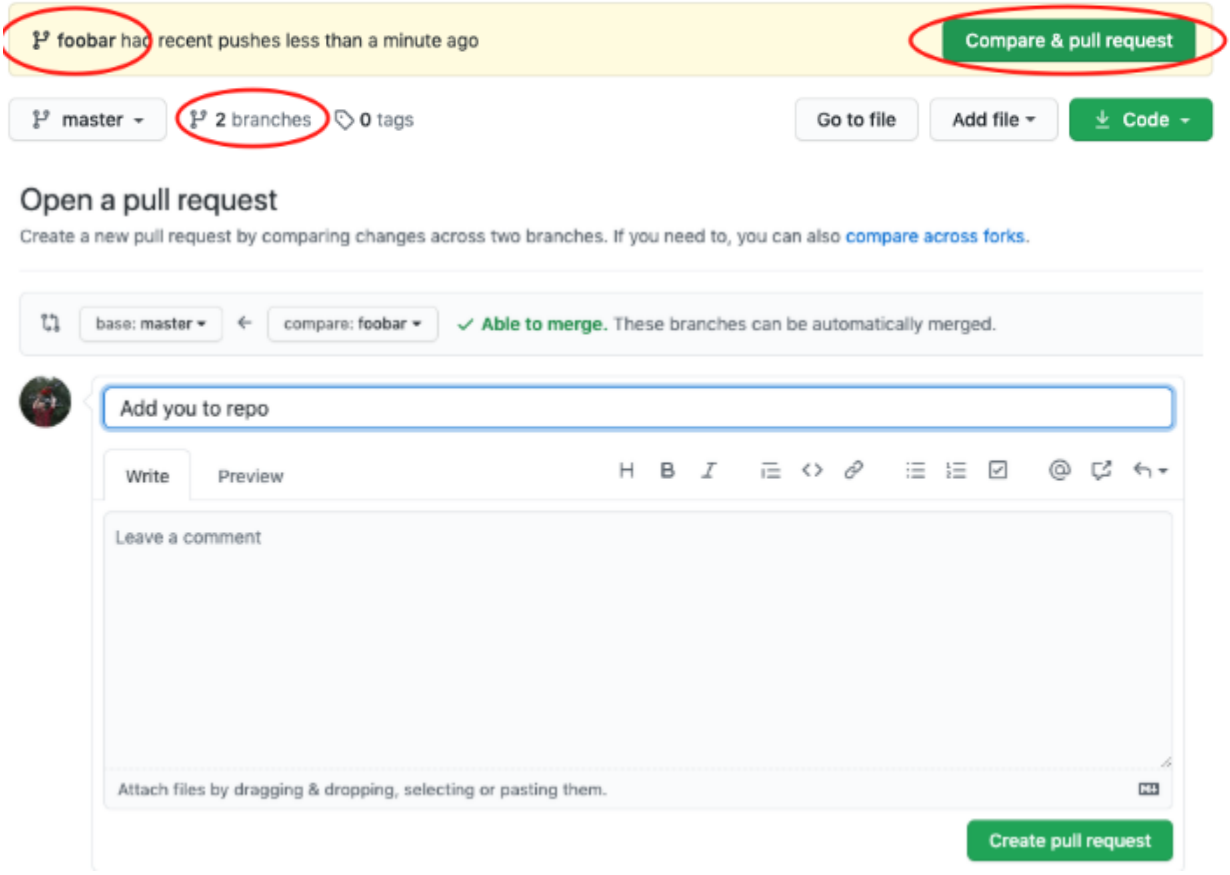
```
mnelson:myproject mnelson$ git status
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   mnelson.txt
```

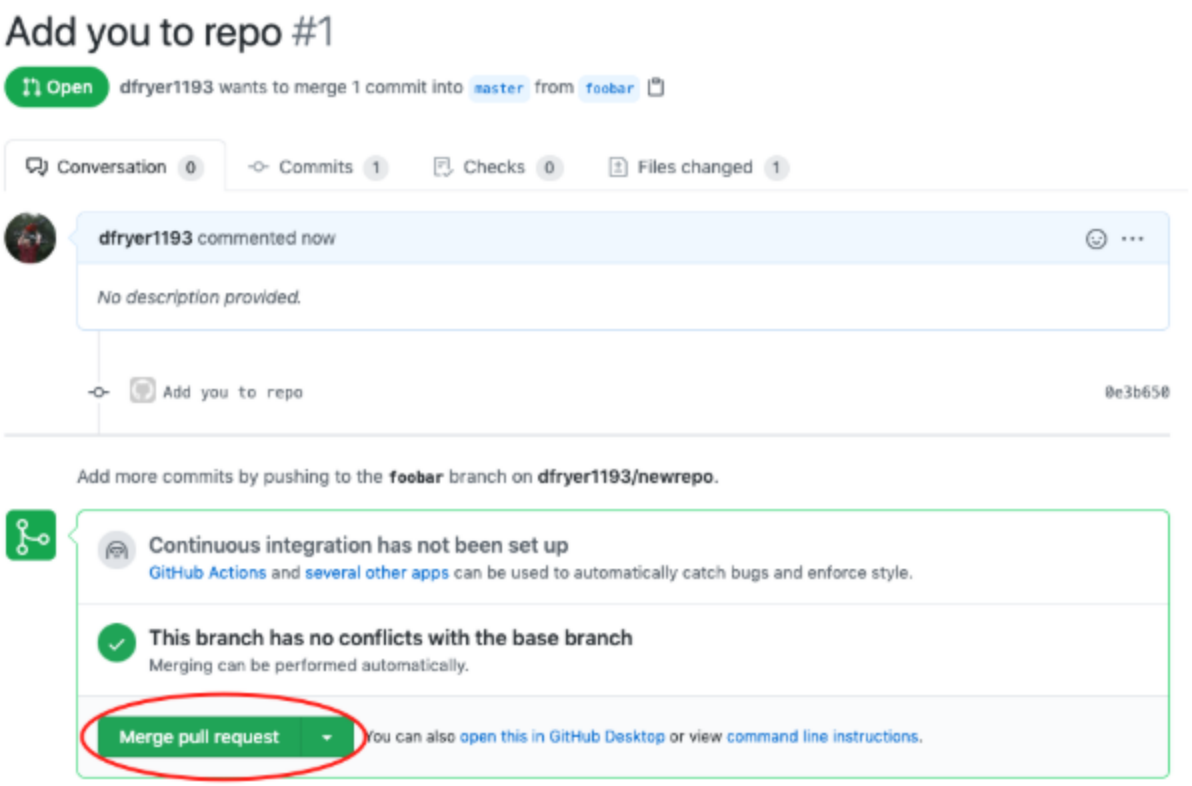
```
mnelson:myproject mnelson$ git commit -m "This is my first commit!"
[master (root-commit) b345d9a] This is my first commit!
 1 file changed, 1 insertion(+)
 create mode 100644 mnelson.txt
```

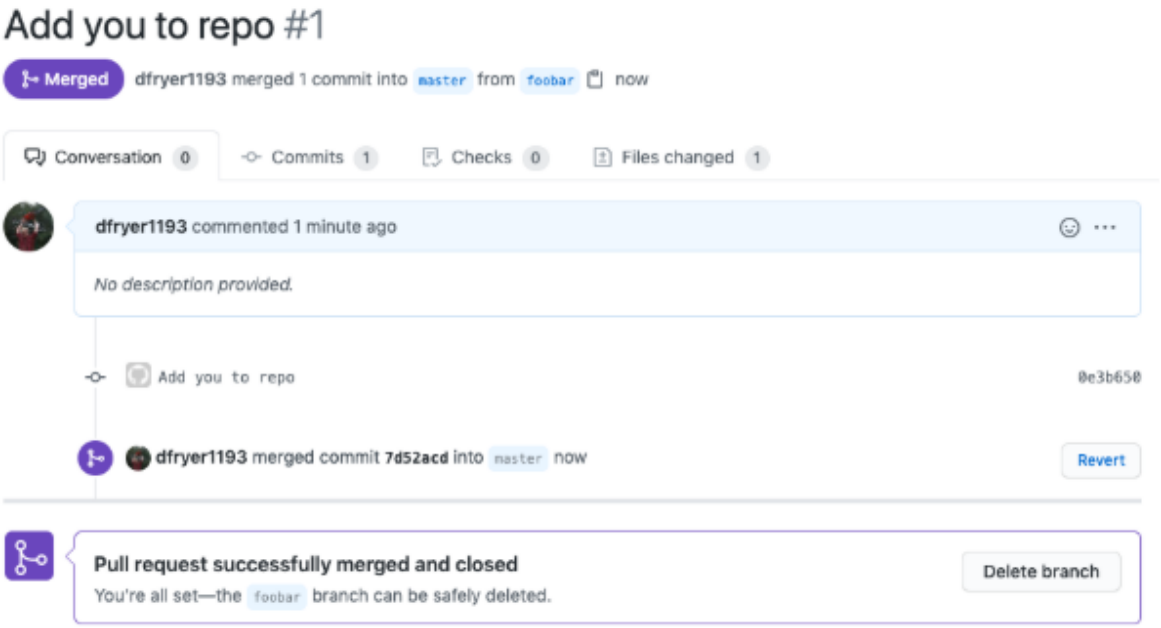
```
mnelson:myproject mnelson$ git branch
  master
* my-new-branch
```

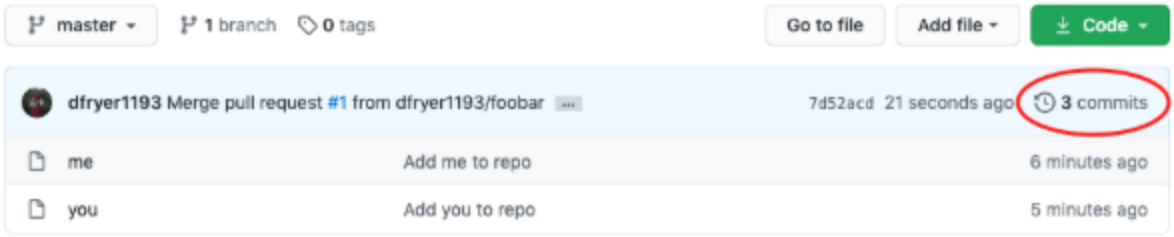
```
mnelson:myproject mnelson$ git remote add origin https://github.com/cubeton/mynewrepository.git
mnelson:myproject mnelson$ git push -u origin master
Counting objects: 3, done.
Writing objects: 100% (3/3), 263 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/cubeton/mynewrepository.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.
```

```
mnelson:myproject mnelson$ git push origin my-new-branch
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 313 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/cubeton/mynewrepository.git
 * [new branch]      my-new-branch -> my-new-branch
```




Add you to repo #1
Open
dfryer1193 wants to merge 1 commit into master from foobar
Conversation 0
Commits 1
Checks 0
Files changed 1
dfryer1193 commented now
No description provided.
Add you to repo
0e3b650
Add more commits by pushing to the foobar branch on dfryer1193/newrepo.
Continuous integration has not been set up
GitHub Actions and several other apps can be used to automatically catch bugs and enforce style.
This branch has no conflicts with the base branch
Merging can be performed automatically.
Merge pull request
You can also open this in GitHub Desktop or view command line instructions.


Add you to repo #1
Merged
dfryer1193 merged 1 commit into master from foobar now
Conversation 0
Commits 1
Checks 0
Files changed 1
dfryer1193 commented 1 minute ago
No description provided.
Add you to repo
0e3b650
dfryer1193 merged commit 7d52acd into master now
Revert
Pull request successfully merged and closed
You're all set—the foobar branch can be safely deleted.
Delete branch




```
mnelson:myproject mnelson$ git pull origin master
remote: Counting objects: 1, done.
remote: Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (1/1), done.
From https://github.com/cubeton/mynewrepository
 * branch            master     -> FETCH_HEAD
   b345d9a..5381b7c  master     -> origin/master
Merge made by the 'recursive' strategy.
 mnelson.txt | 1 +
 1 file changed, 1 insertion(+)
```

```
mnelson:myproject mnelson$ git log
commit 3e270876db0e5ffd3e9bfc5edede89b64b83812c
Merge: 4f1cb17 5381b7c
Author: Meghan Nelson <mnelson@hubspot.com>
Date:   Fri Sep 11 17:48:11 2015 -0400

    Merge branch 'master' of https://github.com/cubeton/mynewrepository

commit 4f1cb1798b6e6890da797f98383e6337df577c2a
Author: Meghan Nelson <mnelson@hubspot.com>
Date:   Fri Sep 11 17:48:00 2015 -0400

    added a new file

commit 5381b7c53212ca92151c743b4ed7dde07d9be3ce
Merge: b345d9a 1e8dc08
Author: Meghan Nelson <meghan@meghan.net>
Date:   Fri Sep 11 17:43:22 2015 -0400

    Merge pull request #2 from cubeton/my-newbranch

    Added some more text to my file

commit 1e8dc0830b4db8c93efd80479ea886264768520c
Author: Meghan Nelson <mnelson@hubspot.com>
Date:   Fri Sep 11 17:06:05 2015 -0400

    Added some more text to my file

commit b345d9a25353037afdeaa9fcaf9f330effd157f1
Author: Meghan Nelson <mnelson@hubspot.com>
Date:   Thu Sep 10 17:42:15 2015 -0400

    This is my first commit!
```

11. Summary of Important Git Commands

Command	Description
git init	Initialize local repo
git clone [URL]	Clone remote repo
git status	Show status of files
git add [file]	Stage file for commit
git commit -m “message”	Commit staged changes
git push	Push changes to GitHub
git pull	Fetch & merge from GitHub
git branch	List branches
git checkout [branch]	Switch branch
git merge [branch]	Merge specified branch
git log	View commit history

WEEK - 4

Jenkins installation and setup, explore the environment.

To install Jenkins, perform initial setup, and explore the Jenkins environment for automation tasks.

Software Requirements:

- Java installed (JDK 11 or higher)
- Jenkins package (https://www.jenkins.io/download/)
- Web browser

Lab Procedure:

1. Install Java (If Not Installed)

```
sudo apt update
sudo apt install openjdk-11-jdk
java -version
```

---

2. Install Jenkins

For Ubuntu/Debian:

```
wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -
sudo sh -c 'echo deb https://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'
sudo apt update
sudo apt install jenkins
```

For RedHat/CentOS:

Follow instructions from the official Jenkins website.

---

3. Start Jenkins Service

```
sudo systemctl start jenkins
sudo systemctl enable jenkins
sudo systemctl status jenkins
```

---

4. Access Jenkins Web Interface

- Open a web browser and navigate to:

http://localhost:8080

---

5. Unlock Jenkins

- Retrieve the administrator password:

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

- Paste the password into the Jenkins setup wizard.

---

6. Install Suggested Plugins

- Choose **Install suggested plugins** when prompted.
- Wait for installation to complete.

---

7. Create First Admin User

- Enter details like username, password, full name, and email address.

---

8. Explore Jenkins Dashboard

- View Jenkins dashboard interface.
- Explore options like **New Item**, **Manage Jenkins**, **Build History**, and **Credentials**.

---

9. Verify Installation

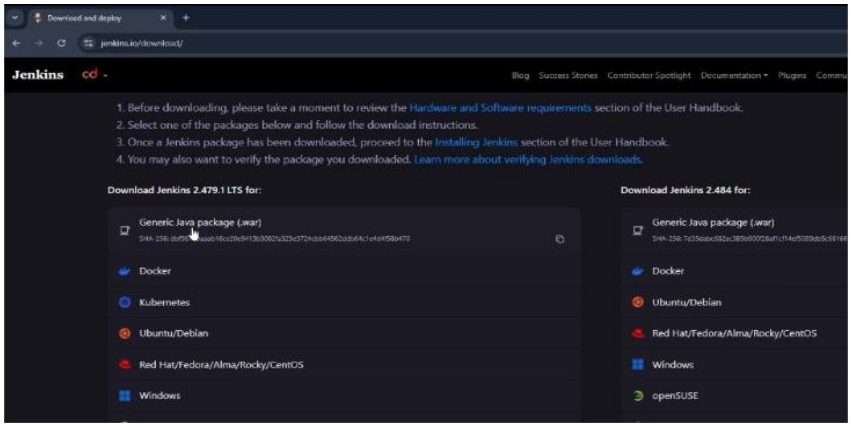
- Create a simple freestyle project.
- Run the project to verify Jenkins build functionality.

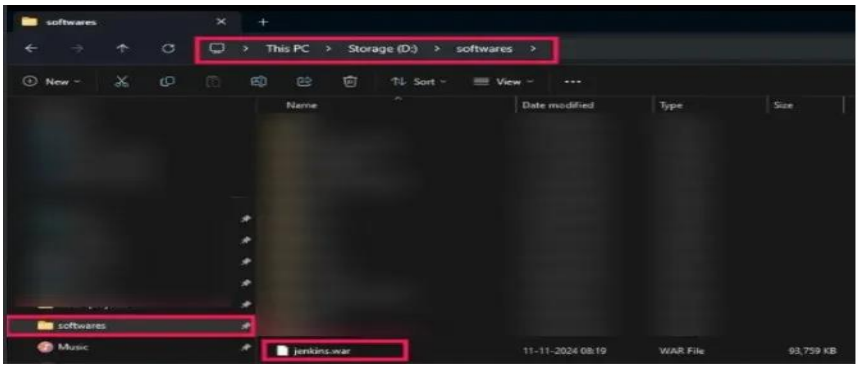
---

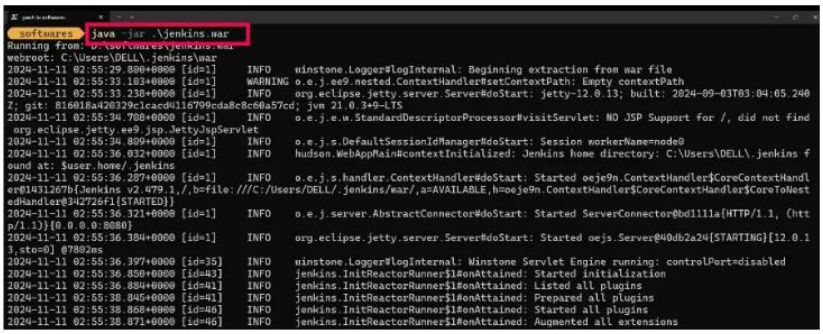
10. Summary of Important Commands

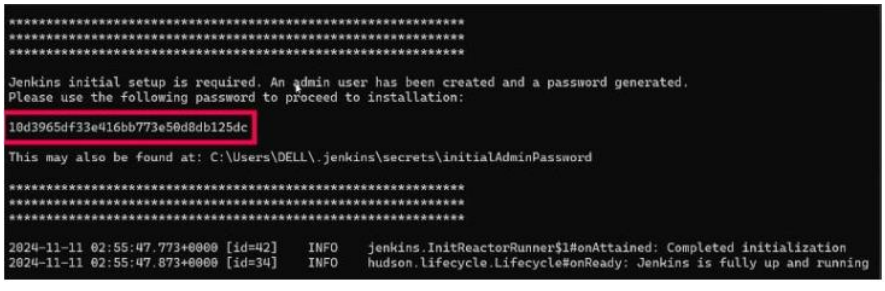
Command	Description
sudo apt install openjdk-11-jdk	Install Java
java -version	Check Java version
sudo apt install jenkins	Install Jenkins
sudo systemctl start jenkins	Start Jenkins service
sudo systemctl enable jenkins	Enable Jenkins on boot
sudo cat /var/lib/jenkins/secrets/initialAdminPassword	Retrieve setup password

OUTPUT:

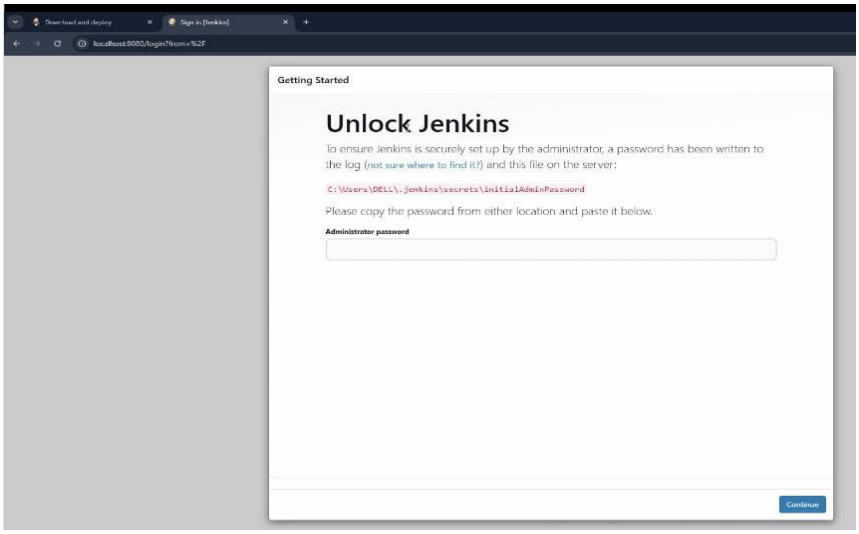








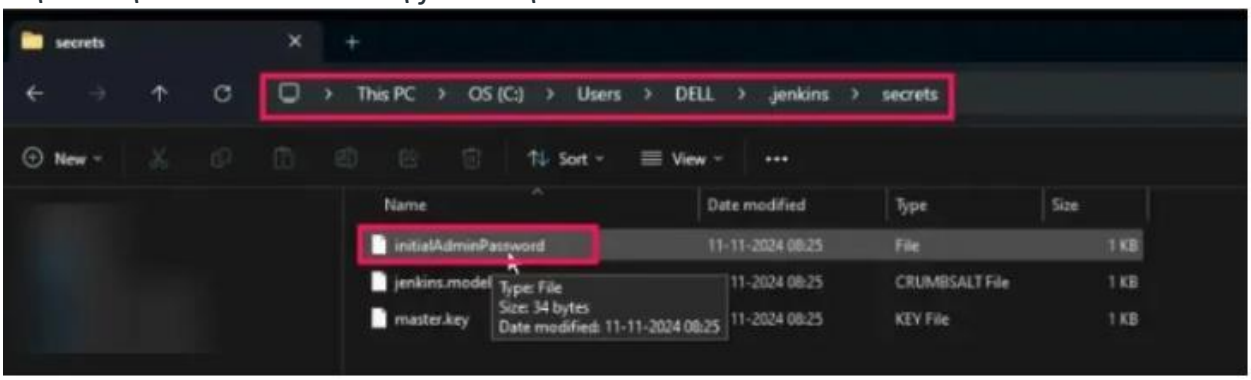
http://localhost:8080



If port 8080 is unavailable, Jenkins will automatically suggest an alternate port in the terminal output.

Unlock Jenkins:

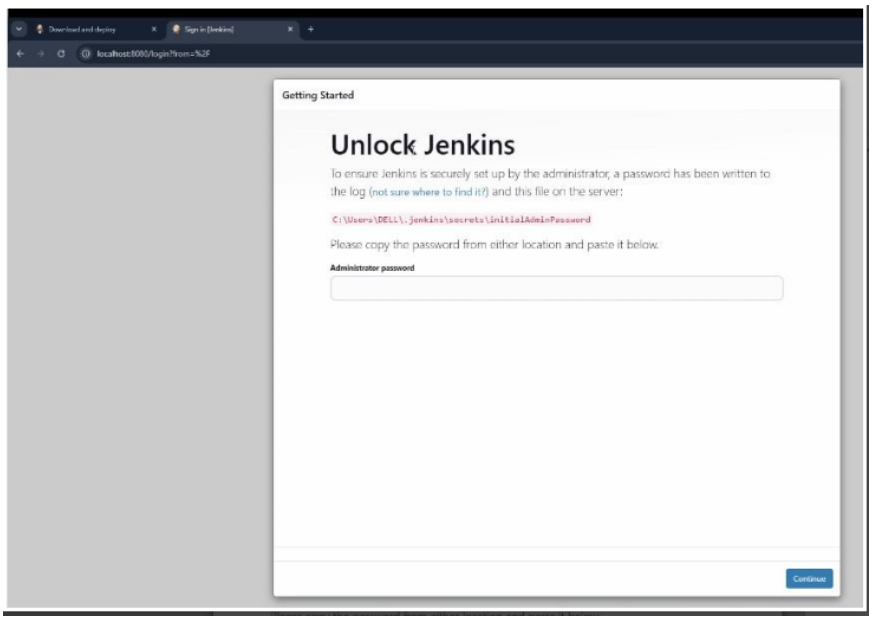
During the setup, Jenkins will generate a one-time admin password. Retrieve the password from the console logs or the initialAdminPassword file located in: *C:\Users\<YourUsername>\.jenkins\secrets*



initialAdminPassword file located

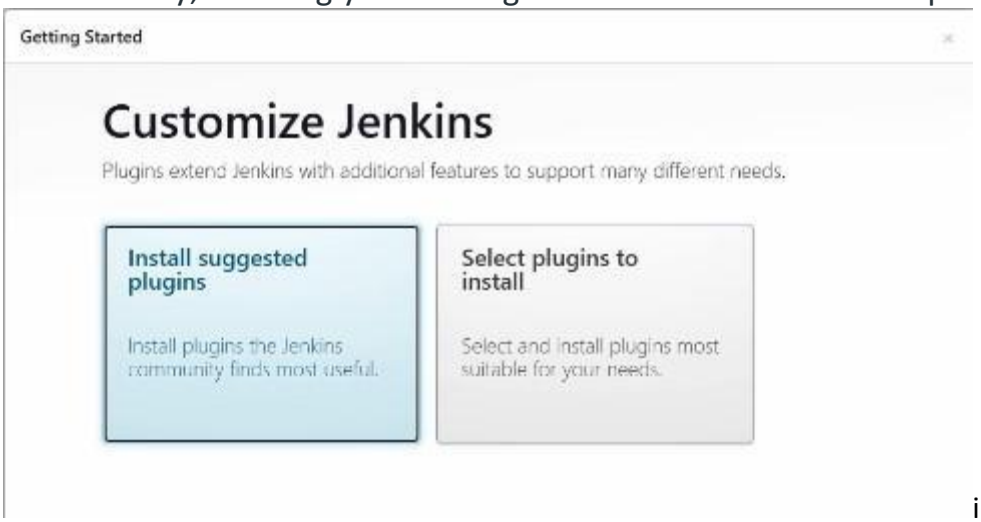
Enter the password on the Jenkins unlock page to proceed.

Enter the password on the Jenkins unlock page



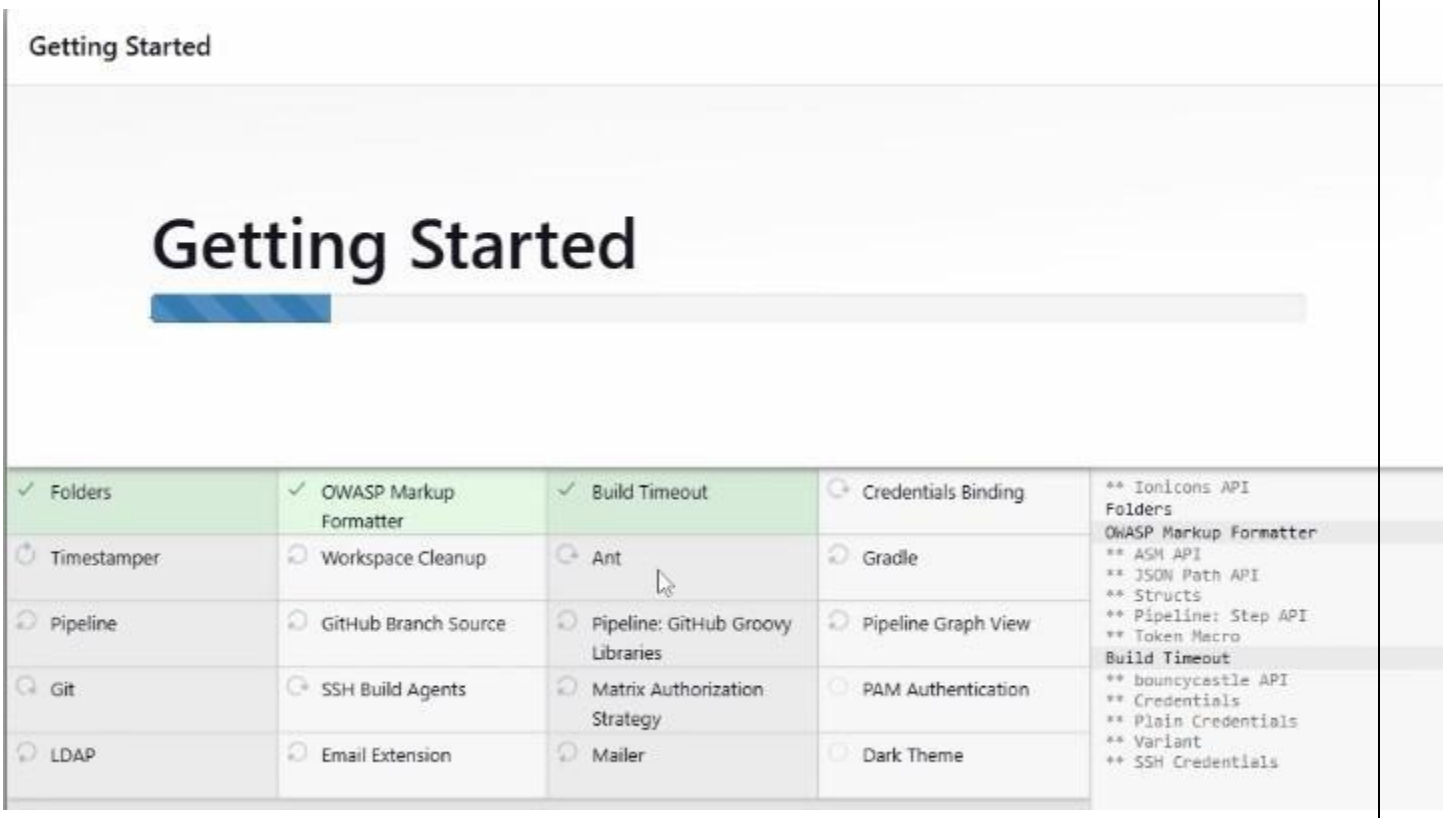
Install Suggested Plugins.

Jenkins will now prompt you to install plugins. Plugins in Jenkins provide extended functionality, allowing you to integrate with various tools and platforms.



install plugins

Choose Install Suggested Plugins for a quick setup. This will install commonly used plugins like Git, Pipeline, and Email Notification.

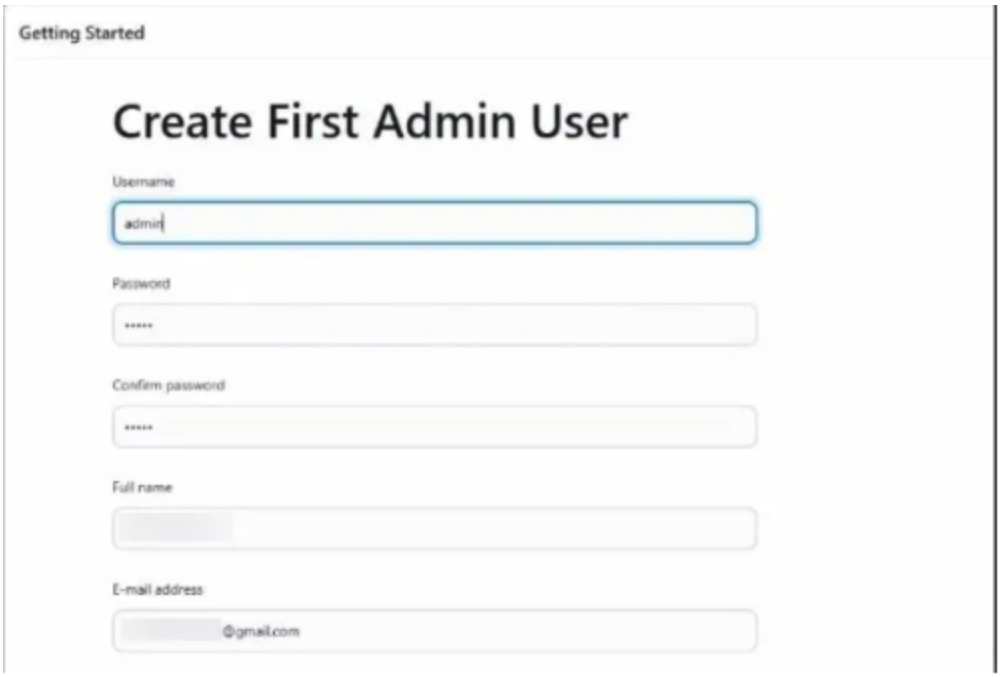


Install commonly used plugins like Git, Pipeline, and Email Notification.

Create an Admin User

Set up an admin user with a username, password, and email.

Create an Admin User

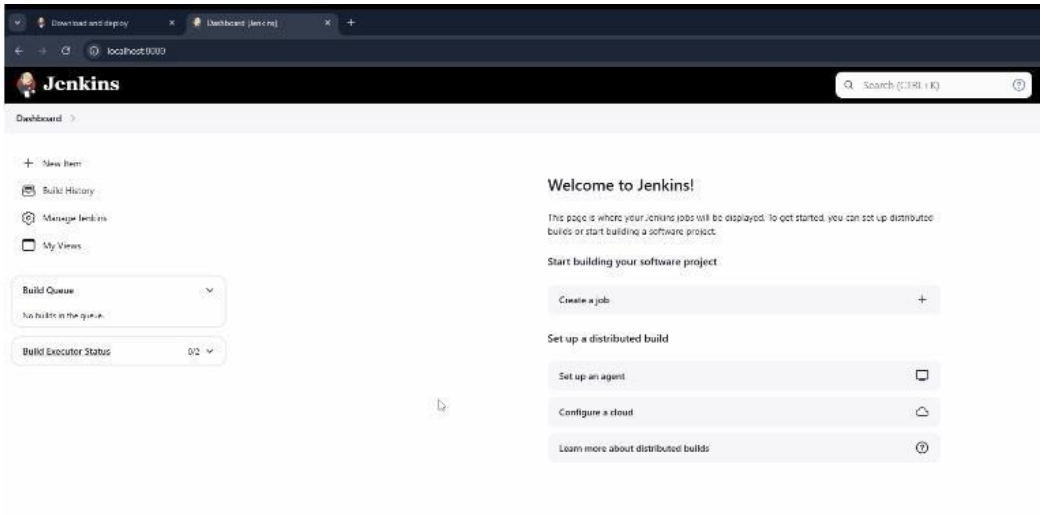


Create an Admin User

This user will be used for future logins.

Complete Installation

After setting up your admin user, Jenkins will finalize the installation and redirect you to the Jenkins dashboard.



WEEK -5

Demonstrate Continuous Integration and Development Using Jenkins

---

Software Requirements:

- Jenkins installed and configured (https://www.jenkins.io/download/)
- Java installed (JDK 11 or higher)
- Git installed (https://git-scm.com/downloads)
- A sample project (Java/Python/Node.js etc.)
- Web browser

Lab Procedure:

1. Configure Jenkins for CI/CD

- Ensure Jenkins is installed and accessible via http://localhost:8080.
- Log in to Jenkins dashboard.

---

2. Install Required Plugins

- Go to **Manage Jenkins > Manage Plugins**.
- Install the following plugins if not already installed:
    - Git Plugin
    - Pipeline Plugin
    - Any relevant build tool plugin (Maven, Gradle, etc.)

---

3. Create New Freestyle Project

- Click on **New Item**.
- Enter project name and select **Freestyle project**.
- Click **OK**.

---

4. Configure Source Code Management

- In the project configuration, go to **Source Code Management**.
- Select **Git**.
- Enter repository URL (e.g., https://github.com/your-username/sample-project.git).

---

5. Configure Build Triggers

- Enable **Poll SCM** or **Build periodically** as needed.
- Alternatively, configure **GitHub webhook** for automatic builds.

---

6. Define Build Steps

- Go to **Build** section.
- Select appropriate build step (e.g., Execute shell, Invoke Maven targets).
- Example build command:

```
mvn clean install
```

---

7. Optional: Post-build Actions

- Add post-build steps like:
  - Archive artifacts
  - Email notifications

---

8. Save and Build

- Save the project.
- Click **Build Now**.
- Monitor the build process in **Build History**.

---

9. View Console Output

- Click on build number.
- View **Console Output** to check build logs.

---

10. Automate Deployment (Optional)

- Add shell scripts or deployment tasks as build steps to automate deployment after build.

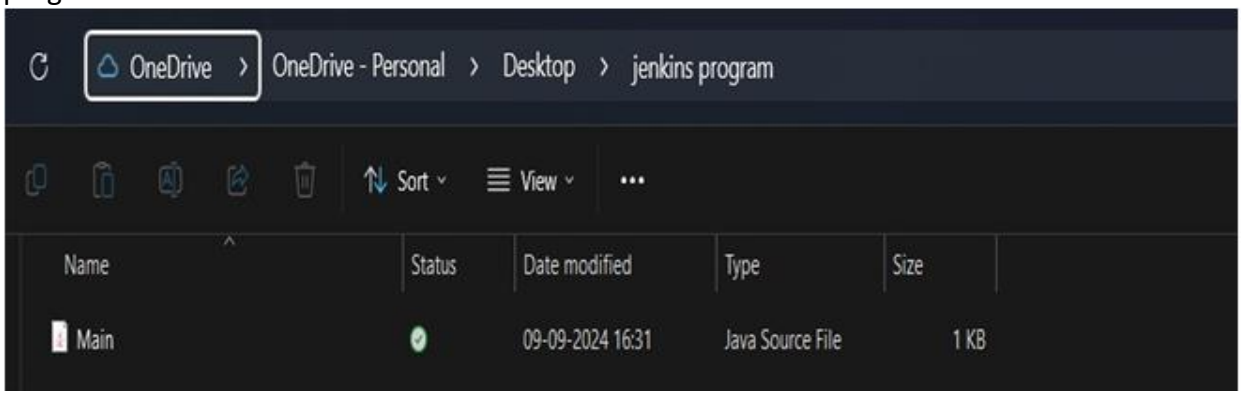
---

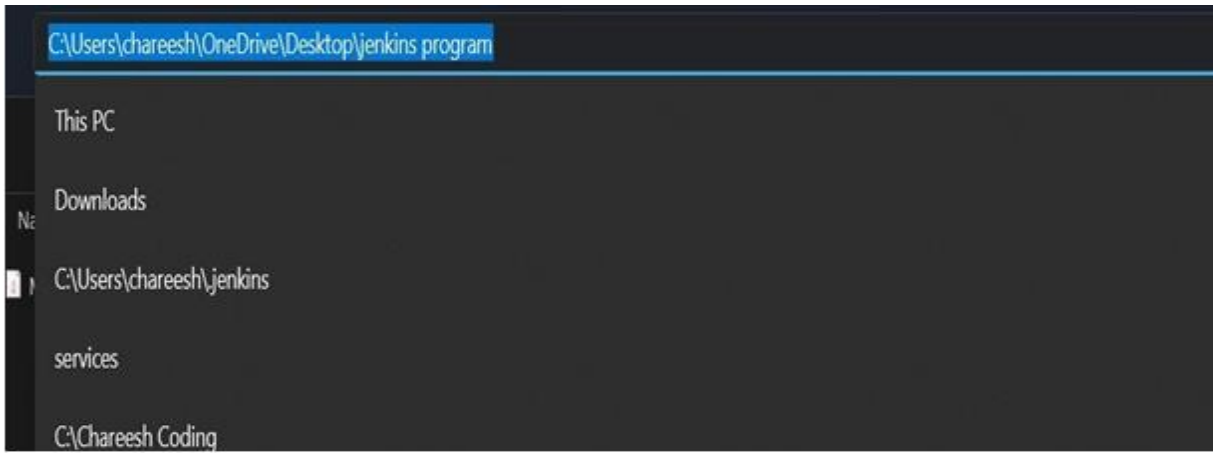
11. Summary of Important Jenkins Steps

Step	Description
Manage Plugins	Install necessary plugins
New Item	Create new project
Source Code Management	Connect to Git repository

Step	Description
Build Triggers	Automate build initiation
Build Steps	Define build instructions
Post-build Actions	Notifications, artifacts, deployments
Build Now	Start manual build
Console Output	Review logs and build process details

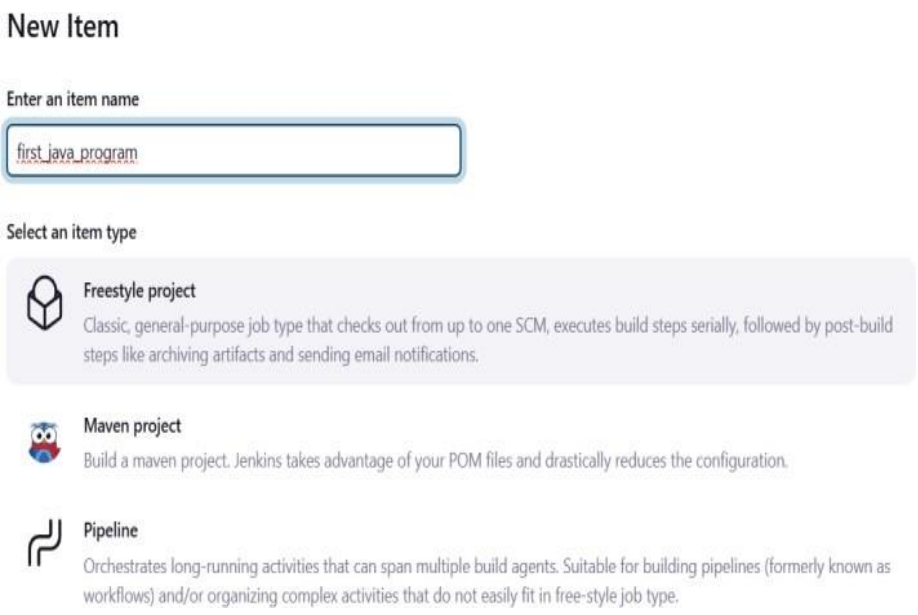
- Create a new folder and then open note pad and write a java program in itand save it in the new folder with .java extension. Now copy the new folderpath where this java program is saved.



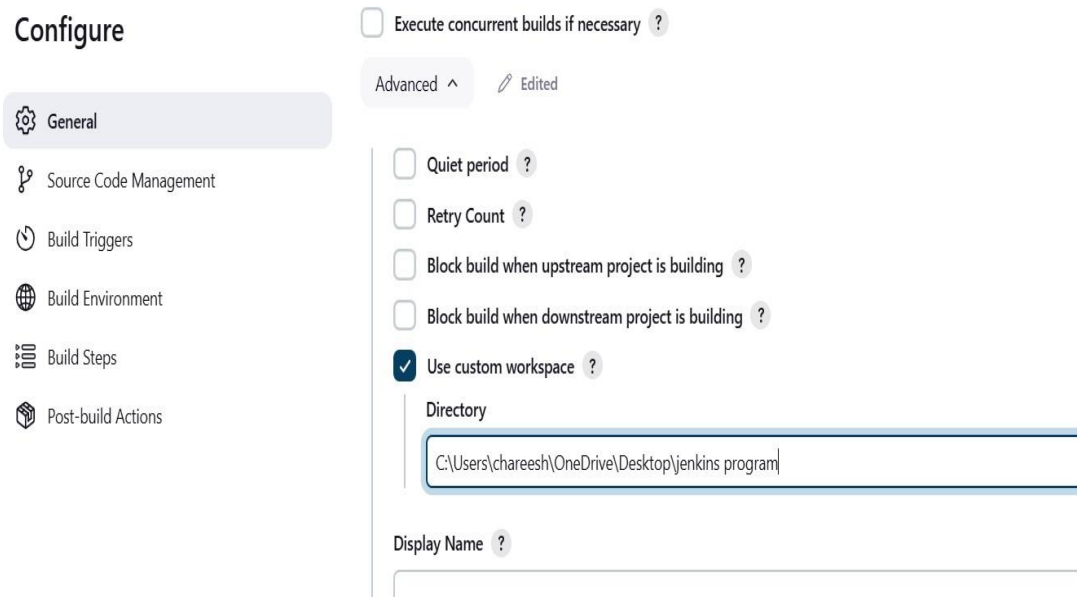


->open Jenkins and go to Dashboards and click on New item

->name the job and select freestyle project and click ok



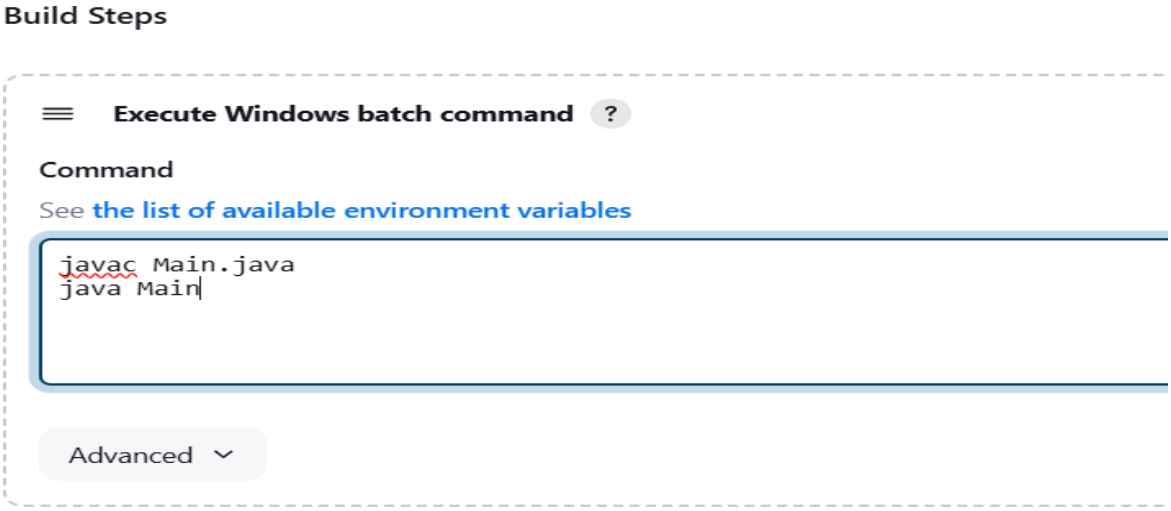
- Now in General section click on advanced and select use custom workspaceand past the path that you copied.



- Now in build steps section click on add build step and select **Execute Windows batch command. Now write**

javac Main.java

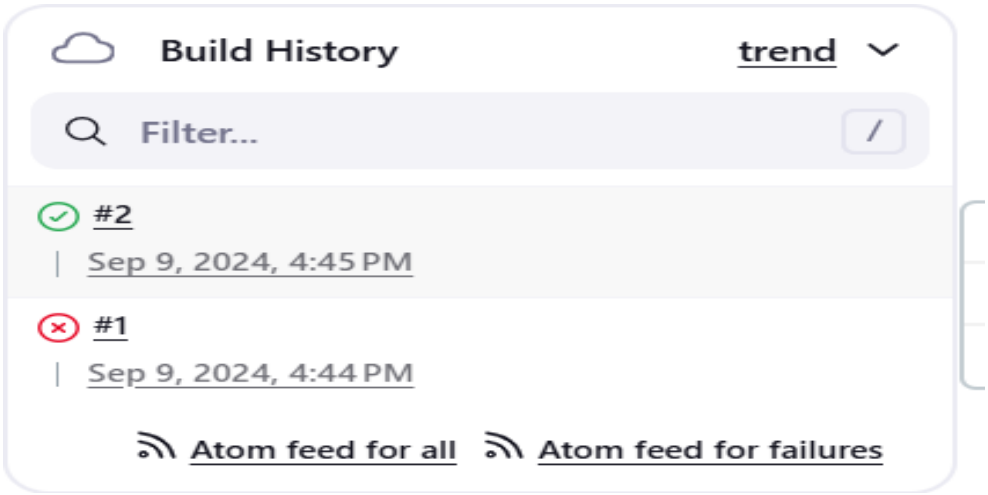
java Main (to run the java program)



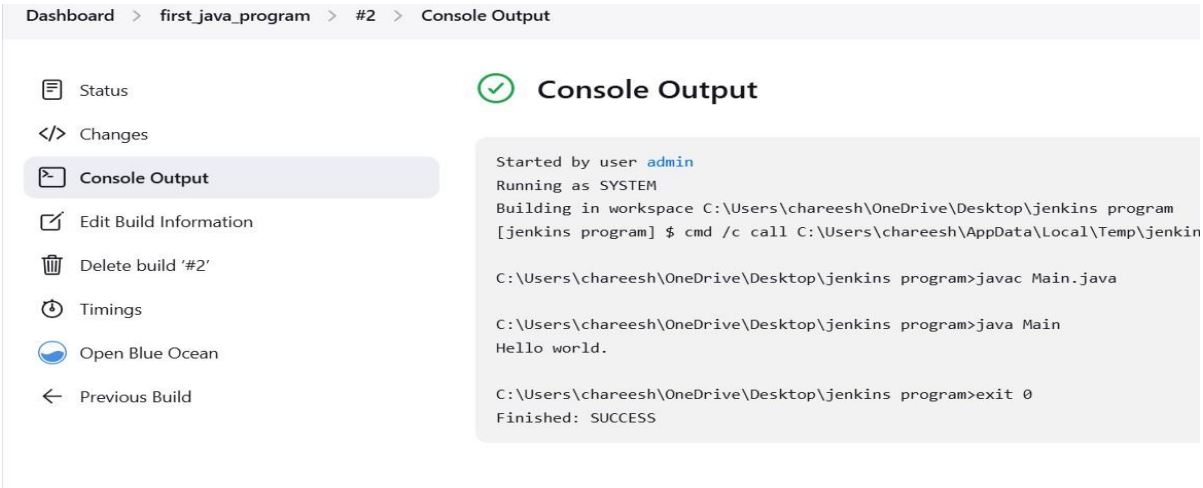
->Now click on apply and then save

->Now click on Build Now

->If you get a tick mark then your job was finished successfully

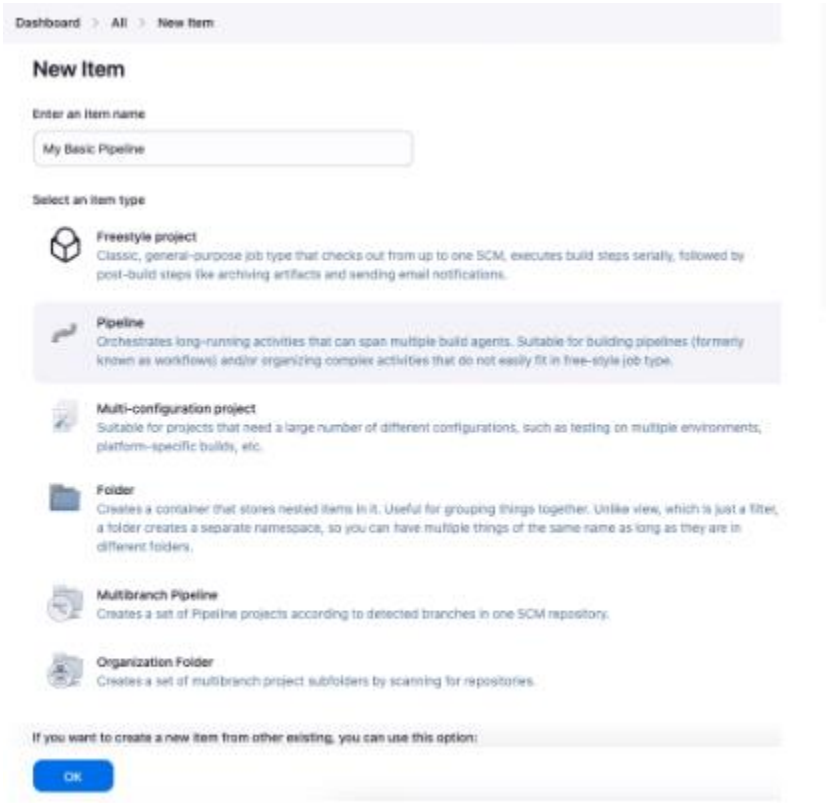


->click on that tick mark to see the output of the program

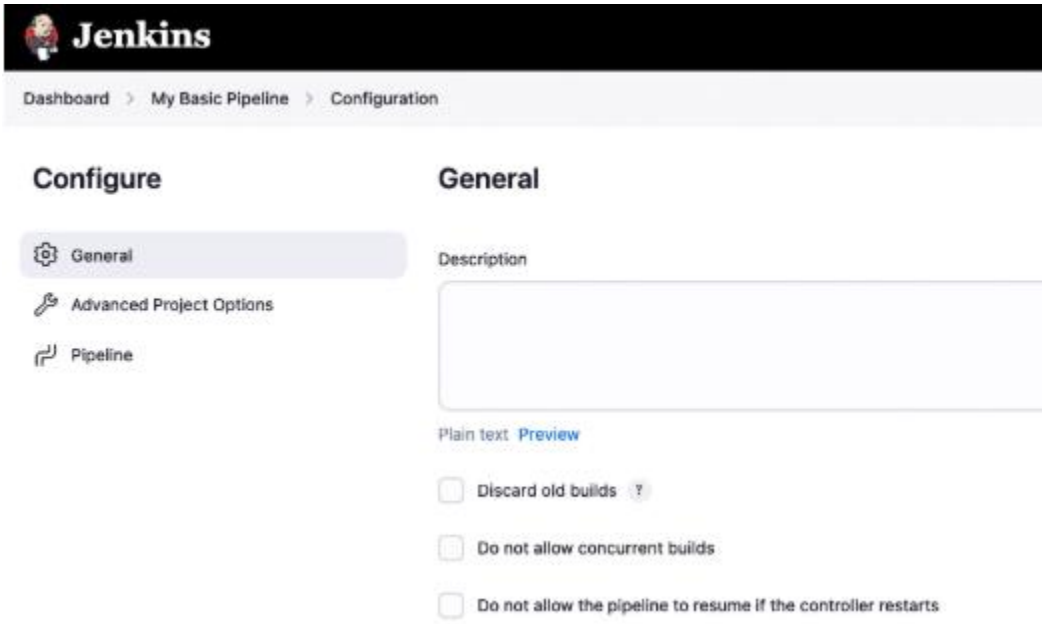


Create a new item (Pipeline)

From the Jenkins Dashboard, click on the *+ New Item* option to open the form below.

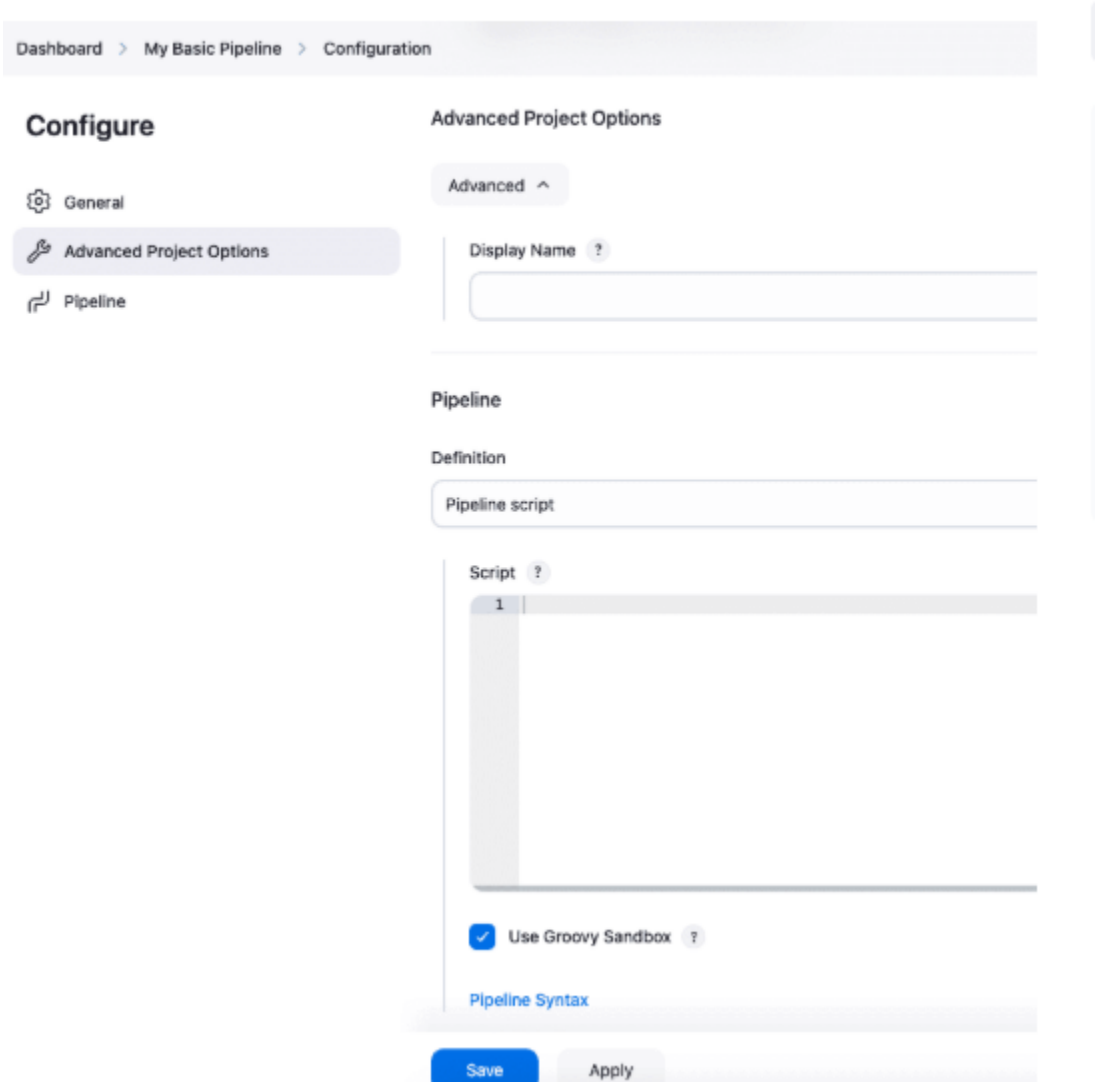


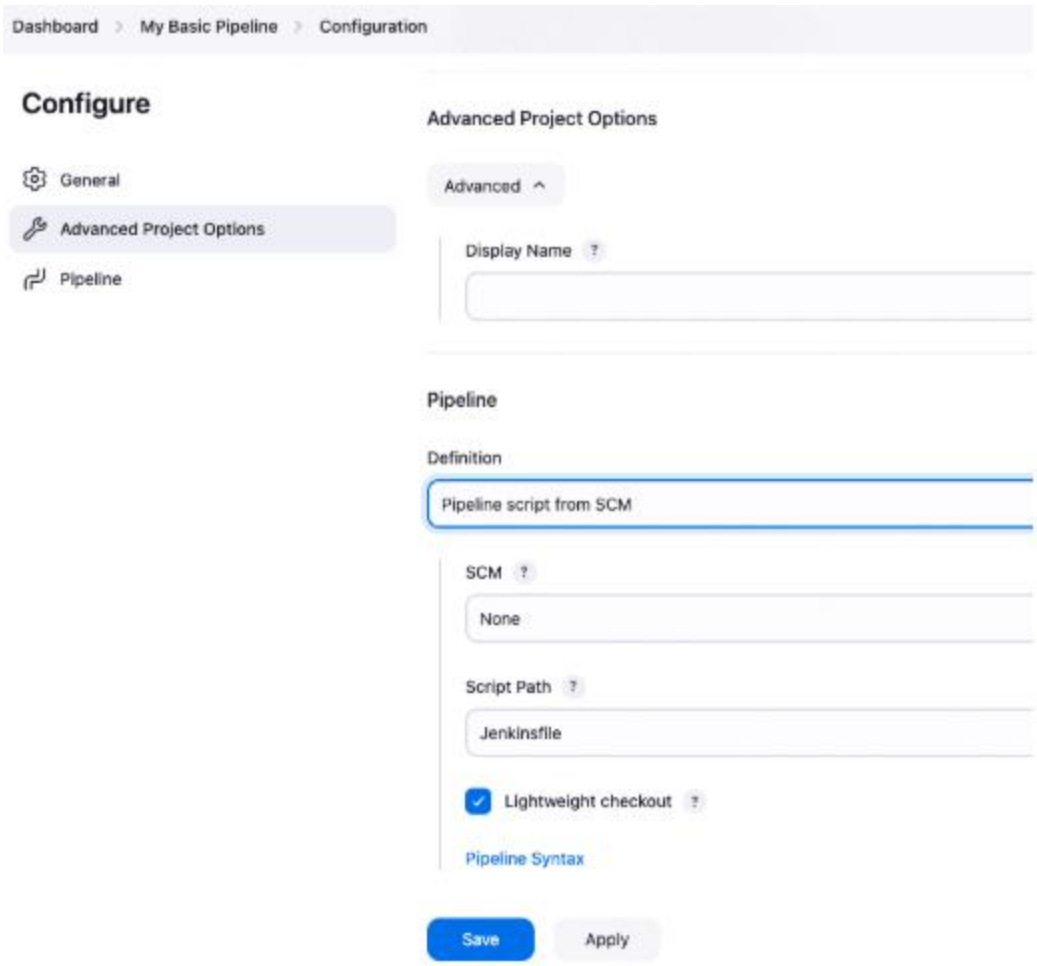
Set the general pipeline configuration options



Define the pipeline

1. Pipeline Script
2. Pipeline Script from SCM





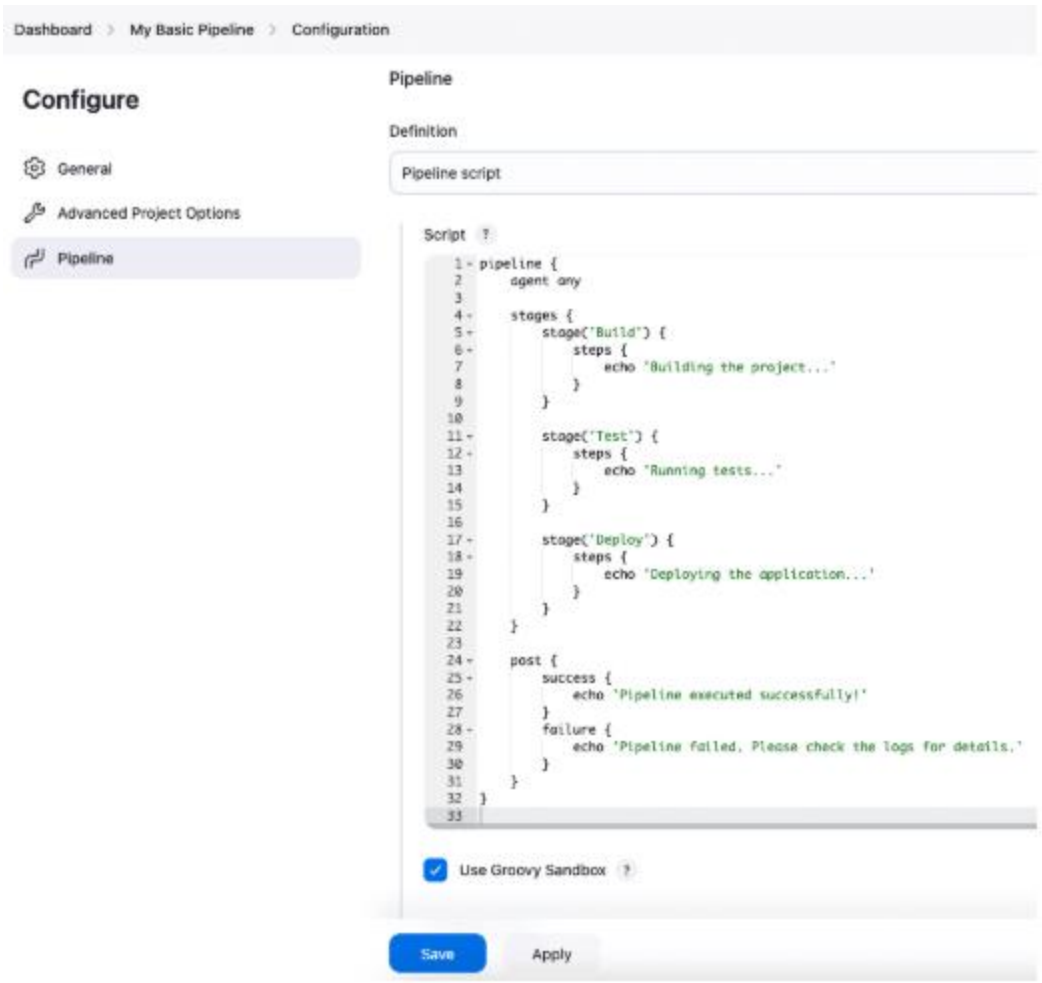
Script the pipeline

This example demonstrates how Jenkins automates tasks, providing clear feedback on execution outcomes.

```
pipeline {
  agent any

  stages {
    stage('Build') {
      steps {
        echo 'Building the project...'
              // bash/shell commands
      }
```

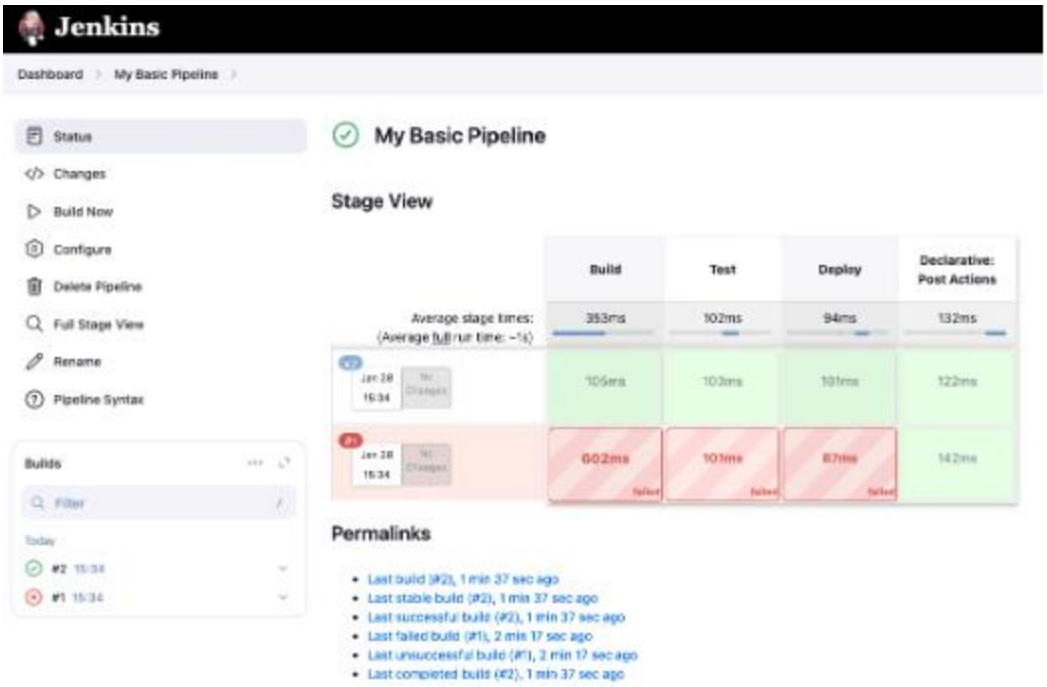
```
    }

    stage('Test') {
      steps {
        echo 'Running tests...'
              // bash/shell commands
      }
    }

    stage('Deploy') {
      steps {
        echo 'Deploying the application...'
              // bash/shell commands
      }
    }
  }

  post {
    success {
      echo 'Pipeline executed successfully!'
    }
    failure {
      echo 'Pipeline failed. Please check the logs for details.'
    }
  }
}
```

To save this script in Jenkins, click the “Save” button at the bottom of the configuration page. Once saved, Jenkins will use the script for the pipeline whenever the job is trigger.

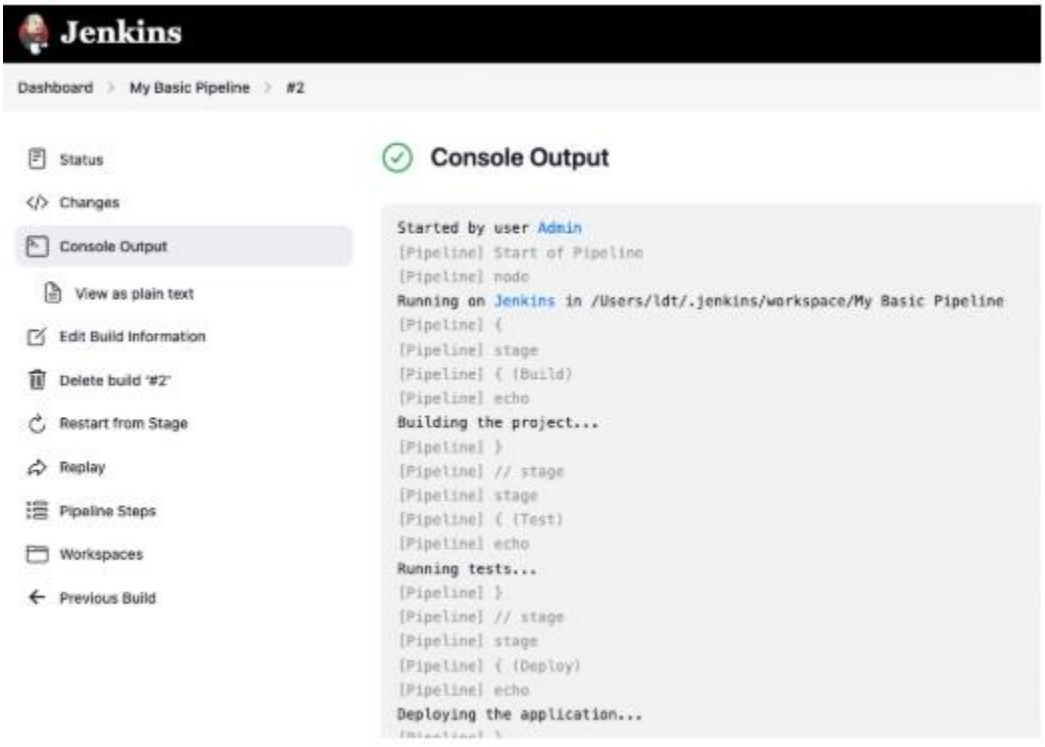


Save the pipeline

Run the pipeline



Access the logs



WEEK -6

Explore Docker commands for content management.

To develop and deploy a simple containerized application using Docker, understanding basic containerization concepts and workflows.

Software Requirements:

- Docker installed (https://www.docker.com/products/docker-desktop/)
- Code editor (VS Code or any)
- Sample application code (Python/Node.js etc.)

Lab Procedure:

1. Install Docker

- Download and install Docker from the official Docker website.
- Verify installation:

```
docker --version
docker info
```

---

2. Create Application Source Code

- Example: Simple Python application (app.py):

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello():
   return "Hello, Docker!"

if __name__ == "__main__":
   app.run(host='0.0.0.0', port=5000)
```

- Create a requirements.txt file:

```
flask
```

---

3. Write Dockerfile

- Create a file named Dockerfile:

```
# Use official Python image
FROM python:3.9

# Set working directory
WORKDIR /app

# Copy application files
COPY . /app

# Install dependencies
RUN pip install -r requirements.txt

# Expose port
EXPOSE 5000

# Run the application
CMD ["python", "app.py"]
```

---

4. Build Docker Image

```
docker build -t my-docker-app .
```

---

5. Run Docker Container

```
docker run -d -p 5000:5000 my-docker-app
```

- Access the application at:

http://localhost:5000

---

6. Verify Running Containers

```
docker ps
```

---

7. Stop and Remove Containers (Optional)

```
docker stop <container_id>
docker rm <container_id>
```

---

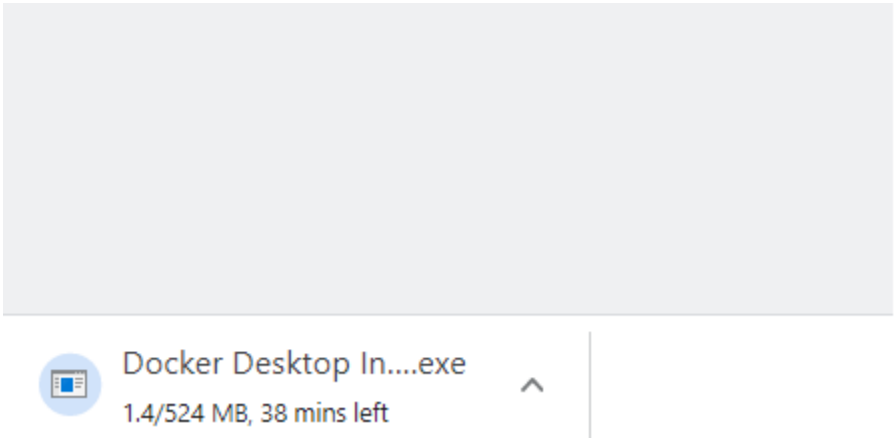
8. Summary of Important Docker Commands

Command	Description
docker –version	Check Docker version
docker build -t name .	Build Docker image
docker run -d -p host:container image	Run containerized app
docker ps	List running containers
docker stop <container_id>	Stop running container
docker rm <container_id>	Remove container
docker images	List Docker images

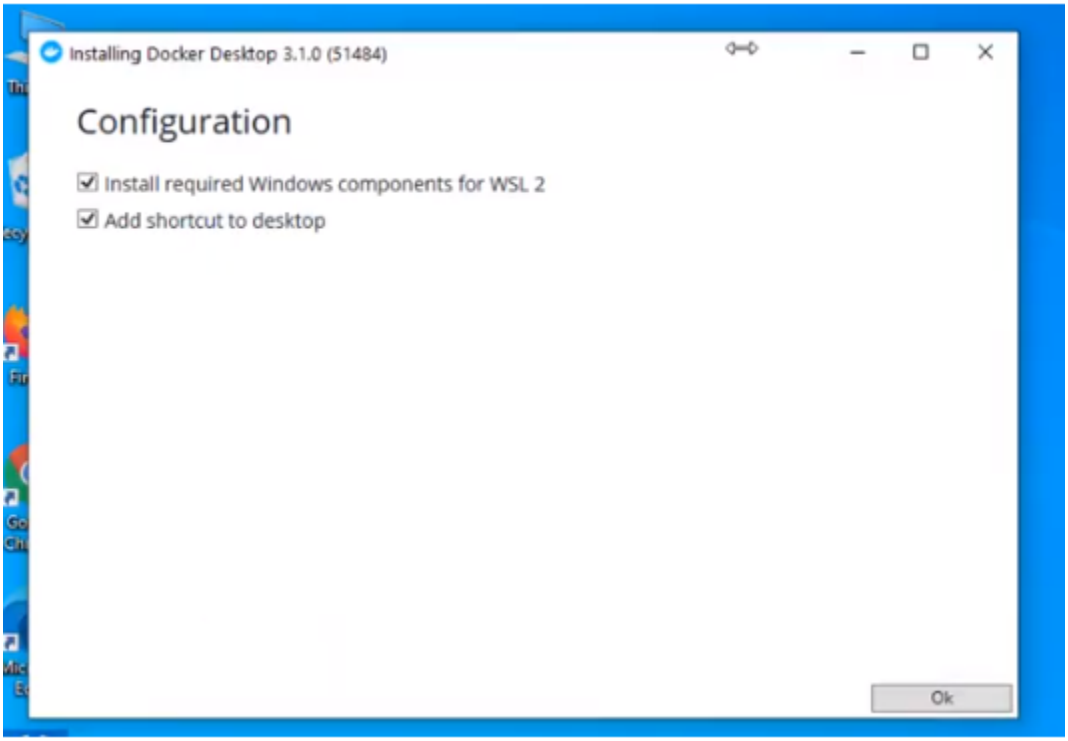
Installation:

- Open your preferred web browser (e.g., Chrome).
- Then search in the browser by typing s "Docker download" and press Enter.
- Click on the first link that appears in the search results.

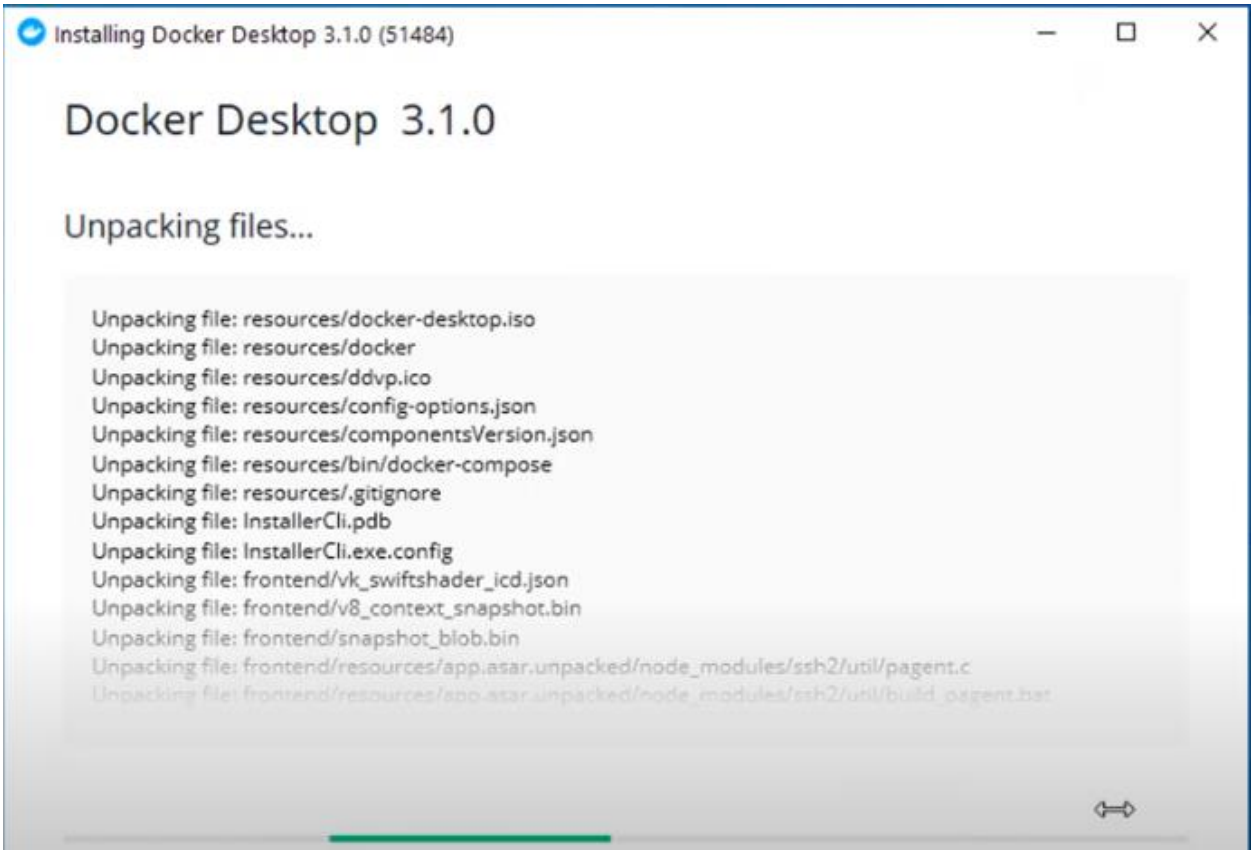
Start the Download



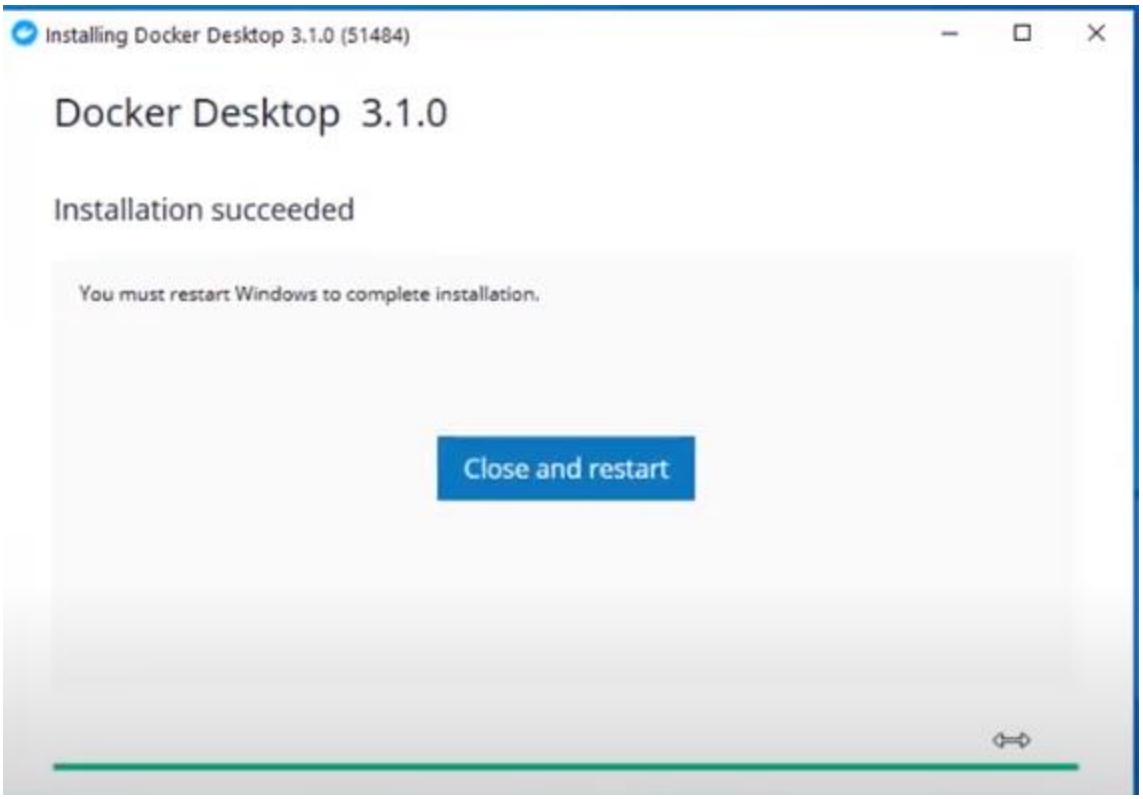
After installation, open Docker Desktop.



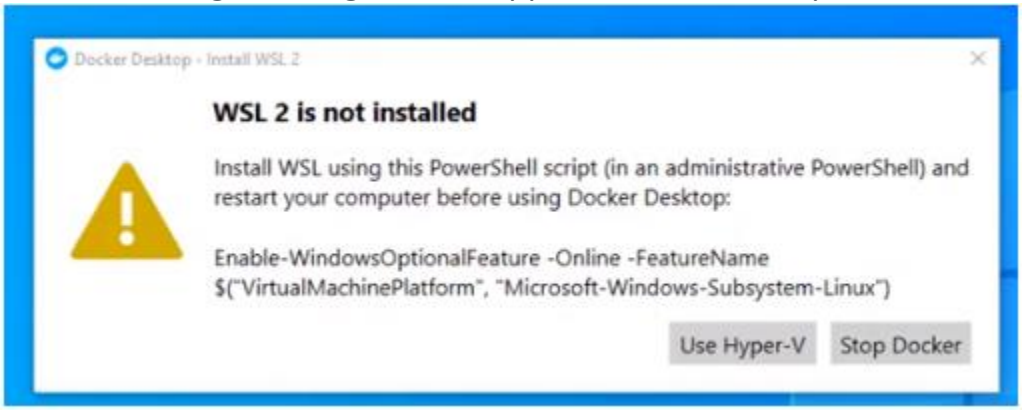
After clicking "OK," the installation will start.



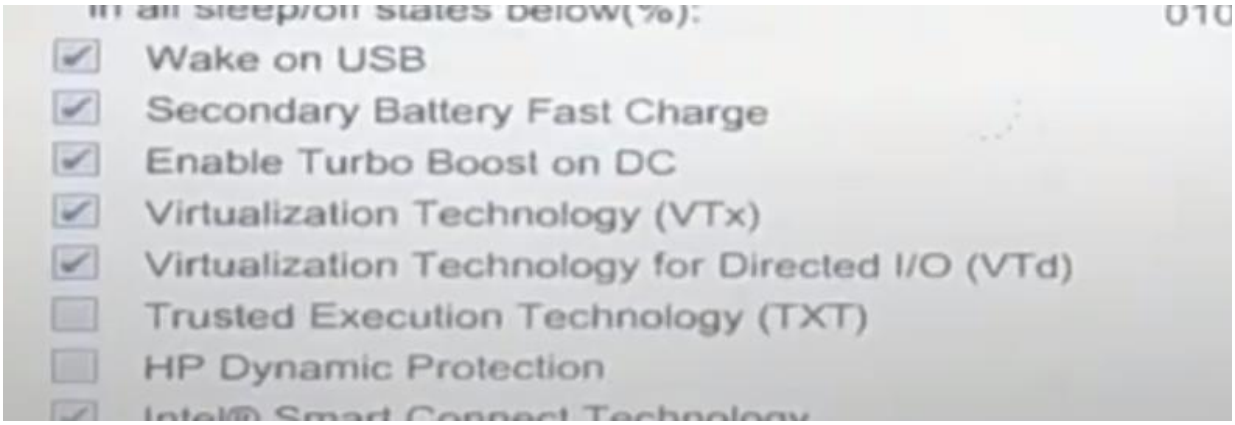
After installation completes, it will show a confirmation screen.



- Restart your PC to install WSL 2 (Windows Subsystem for Linux), a compatibility layer for running Linux binary executables natively on Windows.
- After restarting, a dialog box will appear. Click the "Stop Docker" button.

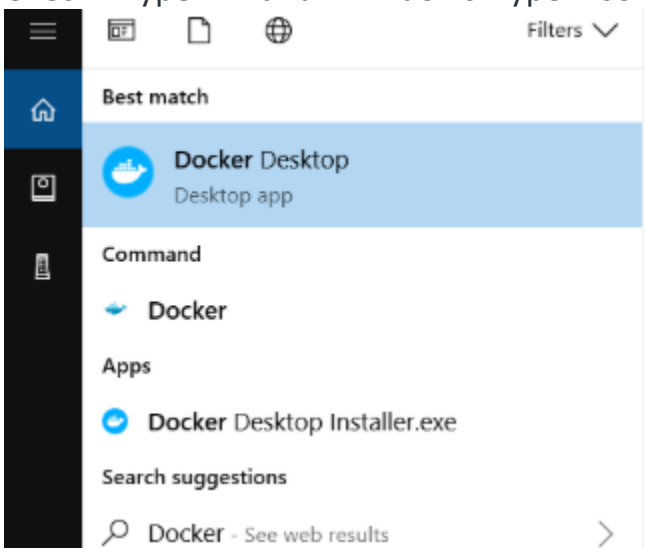


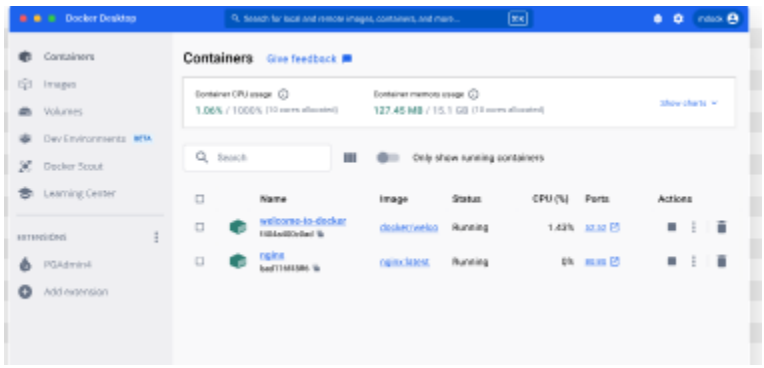
- Restart your PC and enter the BIOS setup:
  - Navigate to Settings > Update and Security > Recovery > Advanced Setup > Device Configuration.
  - Ensure the "Enable Turbo Boost on DC" option is marked. Save and exit.



Step 10: Activate Hyper-V

- Go to Control Panel > Turn Windows Features on or off.
- Check "Hyper-V" and "Windows Hypervisor Platform."





DOCKER COMMANDS:

1. Get help regarding Docker commands: 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
  login       Log in to a registry
  logout      Log out from a registry
  search      Search Docker Hub for images
  version     Show the Docker version information
  info        Display system-wide information

Management Commands:
  builder     Manage builds
  buildx*     Docker Buildx
  checkpoint  Manage checkpoints
  compose*    Docker Compose
  container   Manage containers
  context     Manage contexts
  debug*      Get a shell into any image or container
  desktop*    Docker Desktop commands (Alpha)
  dev*        Docker Dev Environments
  extension*  Manages Docker extensions
```

Docker run command: docker run <image_name>

This command is used to run a container from an image.

```
C:\Users\mojha>docker run redis
1:C 27 Sep 2022 04:55:32.494 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
1:C 27 Sep 2022 04:55:32.494 # Redis version=7.0.5, bits=64, commit=00000000, modified=
1:C 27 Sep 2022 04:55:32.494 # Warning: no config file specified, using the default con
1:M 27 Sep 2022 04:55:32.494 * monotonic clock: POSIX clock_gettime
1:M 27 Sep 2022 04:55:32.495 * Running mode=standalone, port=6379.
1:M 27 Sep 2022 04:55:32.495 # Server initialized
1:M 27 Sep 2022 04:55:32.495 # WARNING overcommit_memory is set to 0! Background save ma
to /etc/sysctl.conf and then reboot or run the command 'sysctl vm.overcommit_memory=1' f
1:M 27 Sep 2022 04:55:32.496 * Ready to accept connections
```

Docker ps: docker ps

Lists all the running containers

```
C:\Users\mojha>docker ps
CONTAINER ID   IMAGE                  COMMAND                  CREATED       STATUS       P
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
```

Docker Pull: docker pull <image_name>

This command allows you to pull any image which is presentin the docker hub.

```
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:b4e56cd71c74e379198b66b3db4dc415f42e8151a18da68d1b61f55fcc7af3e0
Status: Downloaded newer image for redis:latest
docker.io/library/redis:latest
```

Docker Stop: docker stop <container_ID>

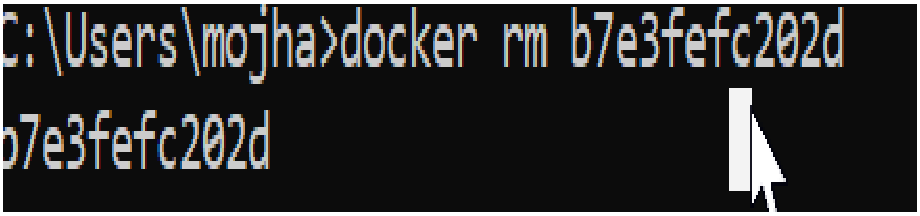
This command allows you to stop a container if it has crashedor you want to switch to another one.

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

Docker Start: docker start <container_ID>

Suppose you want to start the stopped container again, youcan do it with the help of this command.

Docker rm: docker rm <container_name or ID>To delete a container.



Docker Images: docker images

```
C:\Users\mojha>docker images
REPOSITORY                TAG       IMAGE ID       CREATED         SIZE
redis                     latest    9da089657551   4 days ago      117MB
docker/getting-started    latest    cb90f98fd791   5 months ago    28.8MB
docker101tutorial         latest    9a419a57dcb8   6 months ago    28.8MB
ubuntu                    latest    2b4cba85892a   6 months ago    72.8MB
alpine/git                latest    c6b70534b534   10 months ago   27.4MB
hello-world               latest    feb5d9fea6a5   12 months ago   13.3kB
```

Lists all the pulled images which are present in your system.

1. **Docker exec:** docker exec <flag>

   Some important flags:

-d flag: for running the commands in the background.

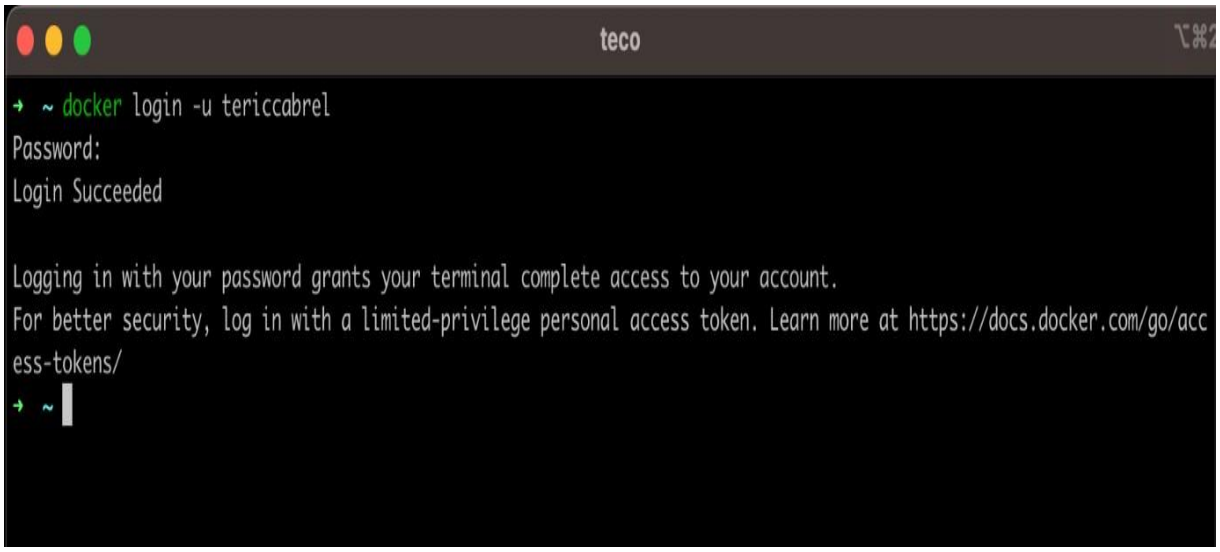
-i flag: it will keep STDIN open even when not attached.

-e flag: sets the environment variables

This command allows us to run new commands in a runningcontainer.

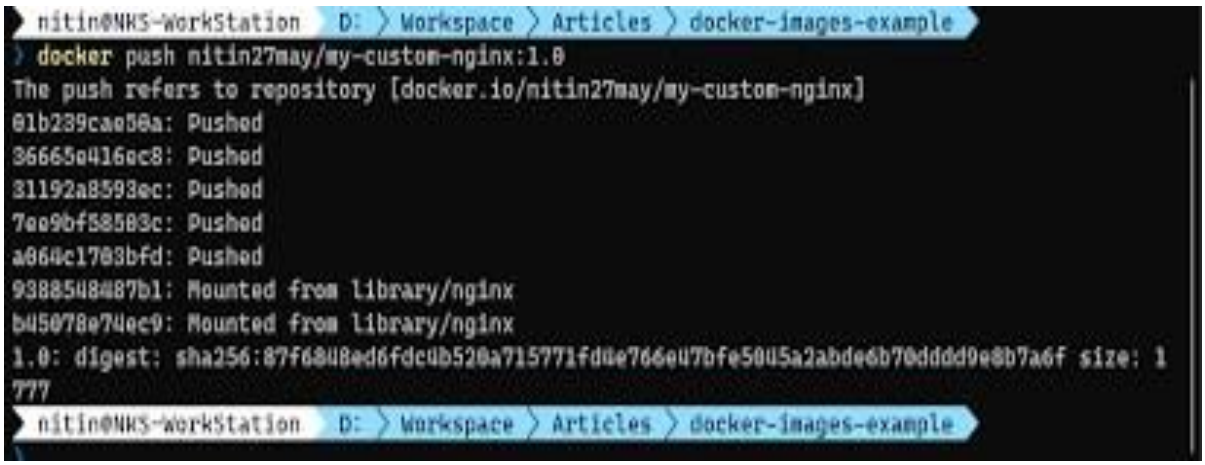
Docker Login: docker login

The Docker login command will help you to authenticate withthe Docker hub by which you can push and pull your images.



Docker Push: docker push <Image name/Image ID>

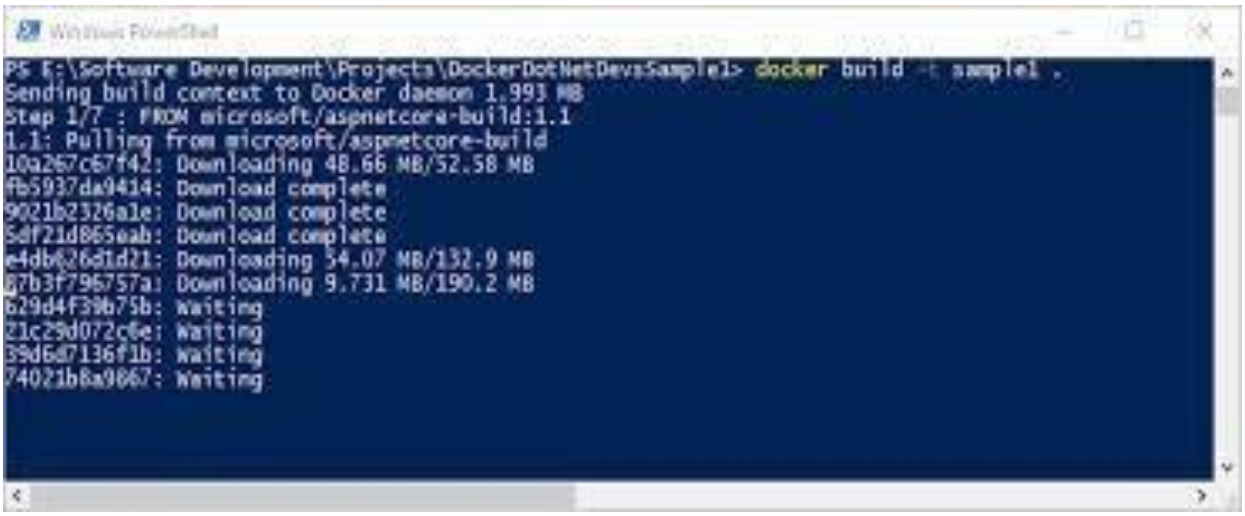
Once you build your own customized image by using Dockerfile you need to store the image in the remote registrywhich is DockerHub for that you need to push your image byusing the following command.



Docker Build: docker build -t image_name:tag .

In the place of image_name use the name of the image you build with and give the tag number and . “dot” represents thecurrent directory.

The docker build command is used to build the docker imageswith the help of Dockerfile.



Docker Stop: docker stop <container_name or id>

You can stop and start the docker containers where you cando the maintenance for containers. To stop and start specificcontainers you can use the following commands.

Stop Multiple Containers: docker stop <container1><container2> <container3>

Instead of stopping a single container. You can stop multiplecontainers at a time by using the following commands.

Docker Restart:

docker restart <container_name_or_id>

While running the containers in Docker you may face some errors and containers fails to start. You can restart the containers.

```
# docker restart $(docker ps -q)
745c51ea5908
aff44fb7b589
351f84d026f1
309c38398750
fa5457424722
9a0c10316d4c
87dbd6caab32
594d408e64ce
60f001a97db6
```

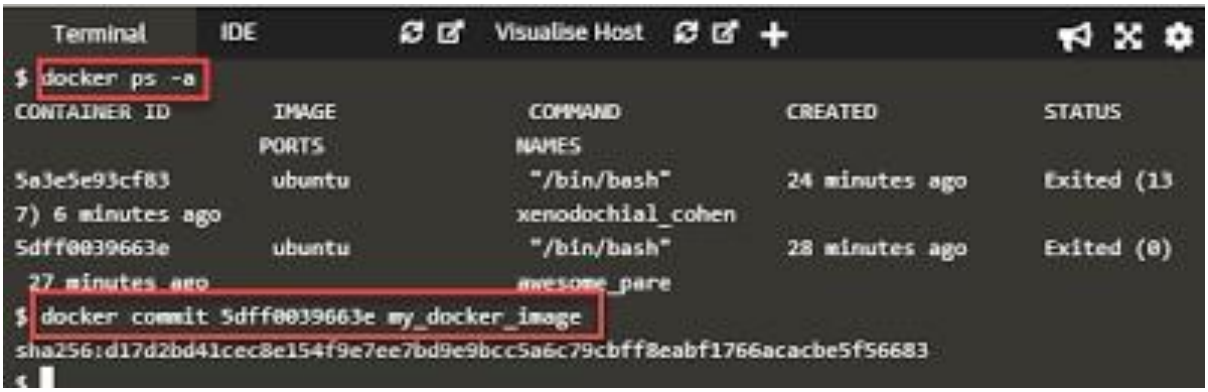
Docker Commit command:

docker commit <container_name_or_id>

<new_image_name>:tag

After running the containers by using the current image you can make the updates to the containers by interacting with

the containers from that containers you can create an image by using the following commands.



5.

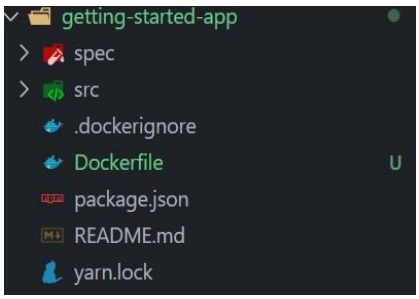
WEEK -7

Develop a simple containerized application using Docker

- 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 see the following files and sub-directories.




```
├── getting-started-app/
| ├── .dockerignore
| ├── package.json
| ├── README.md
| ├── spec/
| ├── src/
| └── yarn.lock
```

- Now create a Docker file in the same directory and add the followingcode

```
FROM
node:18-
alpine
WORKDIR
/app

COPY . .

RUN yarn install --
productionCMD ["node",
"src/index.js"] EXPOSE 3000
```

In the terminal, make sure you're in the getting-started-appdirectory. Replace /path/to/getting-started-appwith thepath to your getting-started-appdirectory.

- cd /path/to/getting-started-app

```
PS C:\Users\Virendra\OneDrive\Desktop\DevOps Lab> cd getting-started-app
PS C:\Users\Virendra\OneDrive\Desktop\DevOps Lab\getting-started-app>
```

Build the Image

Run the following commands in the terminaldocker build -t getting-started .

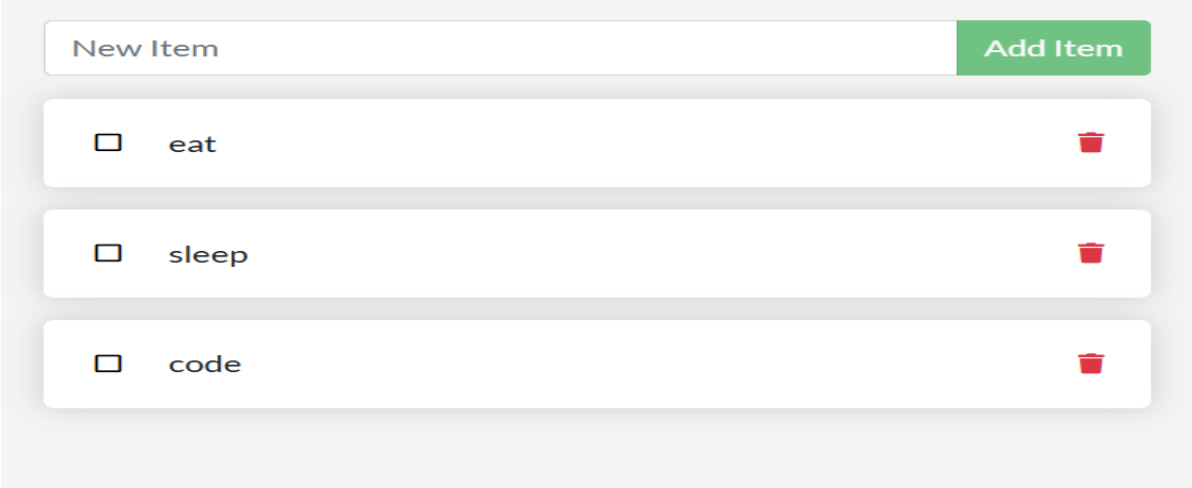
```
PS C:\Users\Virendra\OneDrive\Desktop\DevOps Lab> cd .\getting-started-app\
PS C:\Users\Virendra\OneDrive\Desktop\DevOps Lab\getting-started-app> docker build -t getting-started .
[+] Building 63.3s (10/10) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 156B
 => [internal] load metadata for docker.io/library/node:18-alpine
 => [auth] library/node:pull token for registry-1.docker.io
 => [internal] load .dockerignore
 => => transferring context: 66B
 => [1/4] FROM docker.io/library/node:18-alpine@sha256:02376a266c84acbf45bd19440e08e48b1c8b980374173340466
 => => resolve docker.io/library/node:18-alpine@sha256:02376a266c84acbf45bd19440e08e48b1c8b980374173340466
 => [internal] load build context
 => => transferring context: 6.42MB
 => CACHED [2/4] WORKDIR /app
```

You can verify whether the image is created or not by running the command docker images which will list all the images and you canfind the **getting-started** image. Also you can check the image in the docker desktop images section.

```
C:\Users\Virendra>docker images
REPOSITORY                                  TAG
IMAGE ID       CREATED          SIZE
getting-started                             latest
73024aea2f8b   5 minutes ago    346MB
my-nodejs-app                               latest
4538613e096b   9 days ago       1.37GB
virendra978/my-java-app                     latest
65ede62d20c2   2 weeks ago      673MB
```

> Now run the image using the command
docker run -dp 127.0.0.1:3000:3000 getting-started

> Open browser and type https://localhost:3000 to check the output the image is running on
port 3000.



WEEK -8

Integrate Kubernetes and Docker

Procedure to Install Kubernetes on Windows 11 OS

Step 1: Install Docker Desktop

5. Download and install Docker Desktop for Windows from: https://www.docker.com/products/docker-desktop/
6. Ensure Docker Desktop is running.
7. Enable Kubernetes within Docker Desktop:
    - Open Docker Desktop.
    - Go to **Settings** > **Kubernetes**.
    - Check **Enable Kubernetes**.
    - Click **Apply & Restart**.

Docker Desktop will install and configure a single-node Kubernetes cluster.

Step 2: Install kubectl

8. Download the latest kubectl release: https://kubernetes.io/docs/tasks/tools/install-kubectl-windows/
9. Alternatively, use PowerShell to download:

```
curl -LO "https://dl.k8s.io/release/v1.30.0/bin/windows/amd64/kubectl.exe"
```

10. Add the folder containing kubectl.exe to your **System PATH**:
    - Press Win + S and search for **Environment Variables**.
    - Click **Edit the system environment variables**.
    - In **System Properties**, click **Environment Variables…**.
    - Edit the **Path** variable under **System variables**.
    - Add the directory containing kubectl.exe.
11. Verify installation:

```
kubectl version --client
```

Step 3: Access Kubernetes Cluster

12. Use kubectl to check the status of the cluster:

kubectl get nodes

13. Kubernetes dashboard can be accessed via Docker Desktop:

- Open Docker Desktop.
- Go to **Dashboard**.
- View workloads, services, and more.

Optional: Install minikube (Alternative Approach)

14. Download minikube for Windows: https://minikube.sigs.k8s.io/docs/start/

15. Install minikube using PowerShell:

choco install minikube

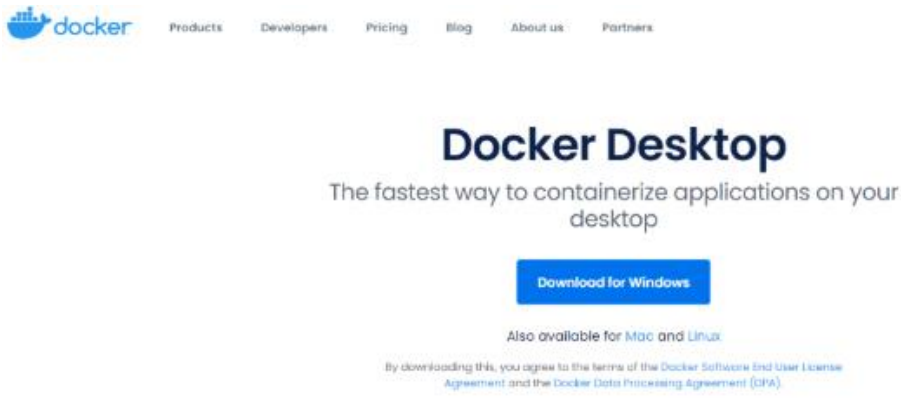
16. Start minikube cluster:

minikube start

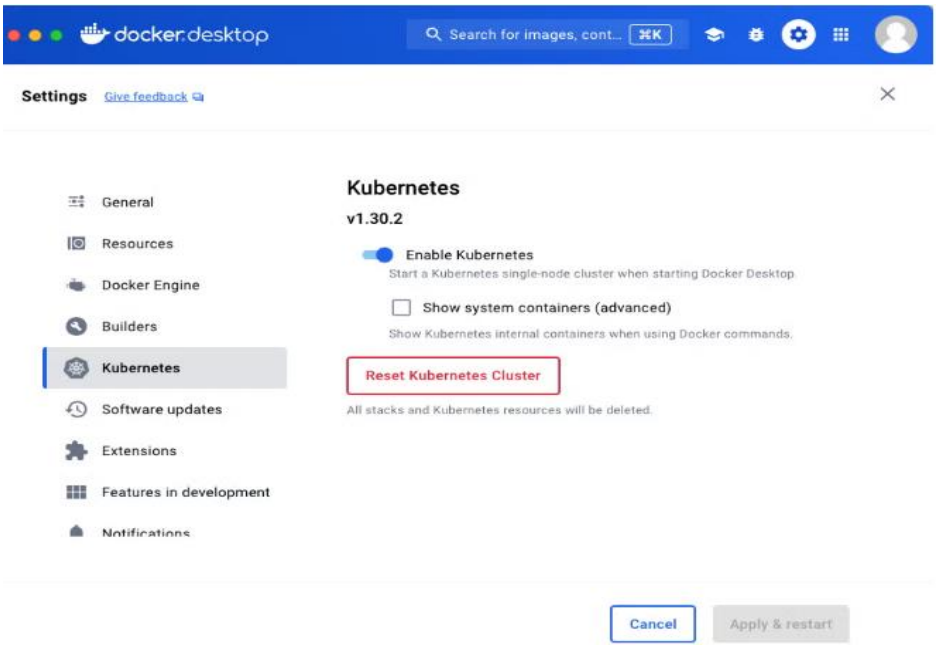
17. Verify minikube status:

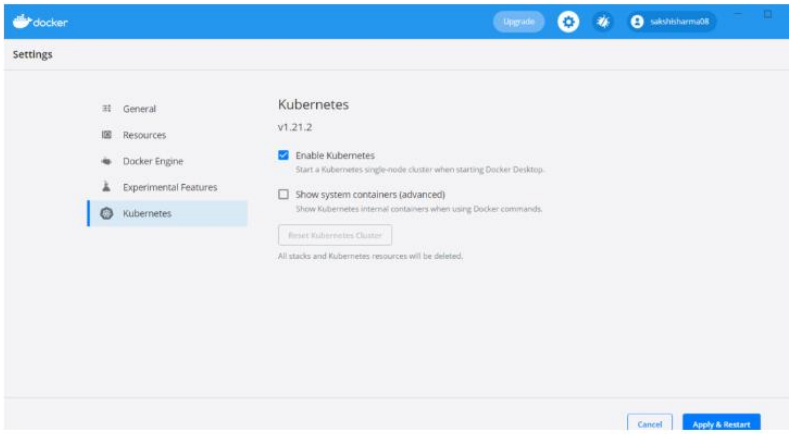
minikube status

Kubernetes is now installed and ready to use on your Windows 11 system.



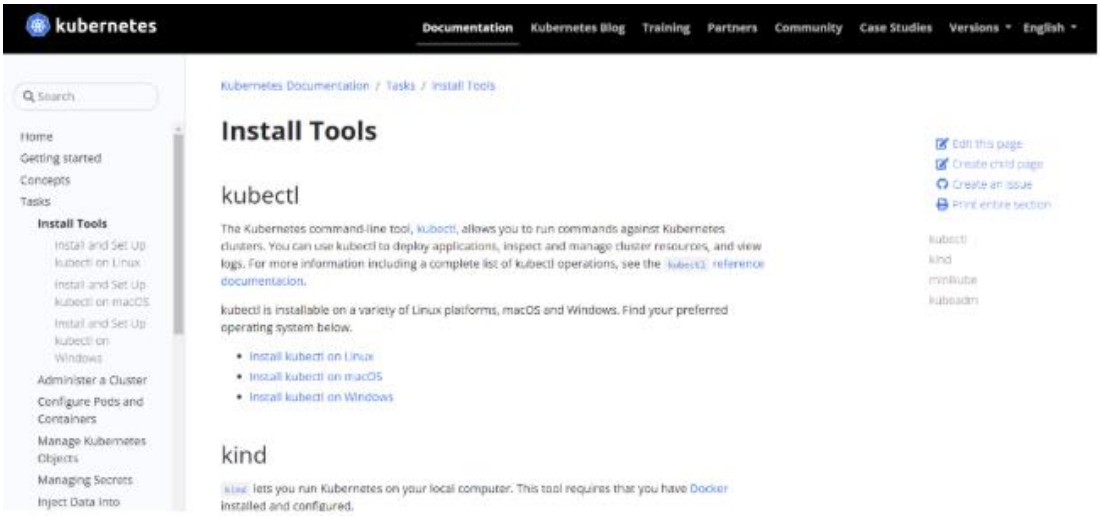
Enable Kubernetes



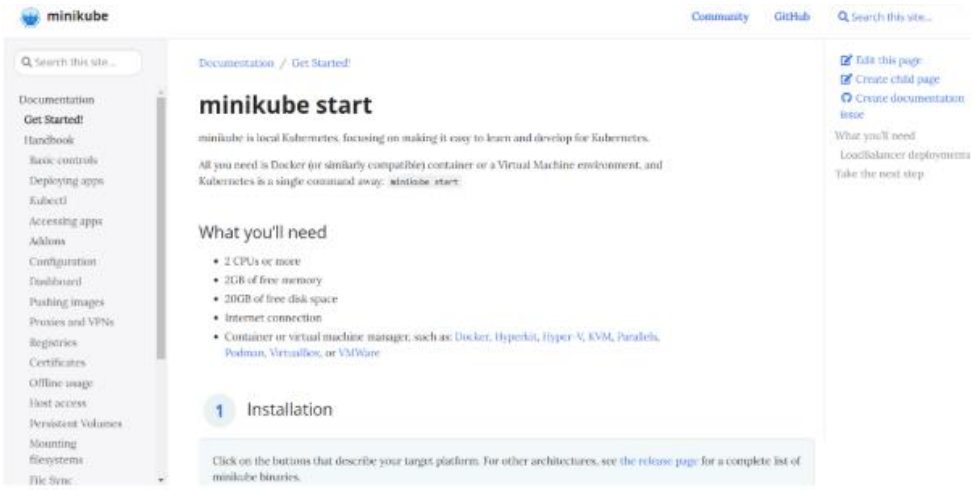


Enable kubernetes cluster

To download Kubernetes CLI

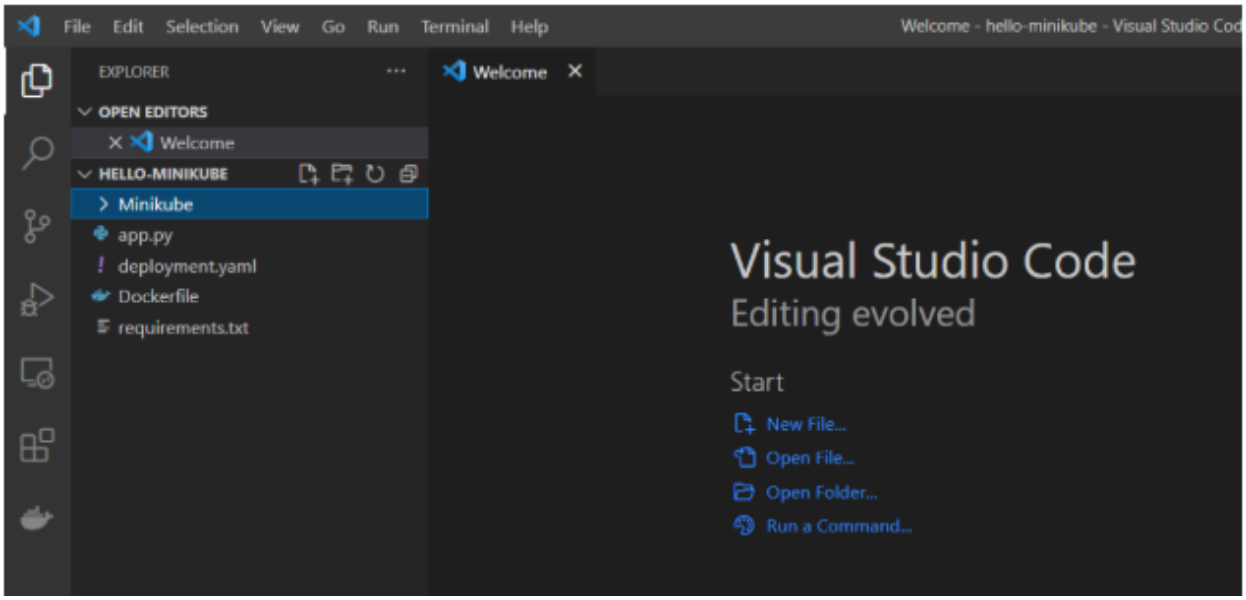


Install Minikube



Once all the installation process has been done we will now make a simple app on any editor.

- Make a folder(here hello-minikube) on your desktop.
- Open that folder in your Visual Studio Code(you can use the editor of your choice). The Project Structure looks as follows:



app.py

```
from flask import flask, jsonify
import time

app = flask(_name_)

app.router("/")
def hello_minikubegfg():
    return jsonify({"Time to Call":time.time()})

if _name_ == "_main_":
    app.run(host='0.0.0.0',debug=True)
```

- In requirement.txt file-> Just add the required dependency(here I have added flask)
- In Docker file -> Pull the docker image from docker hub, and also copy all the files which are there in the directory as mentioned below:

```
FROM python:3.7
RUN mkdir /app
WORKDIR /app/
ADD . /app/
RUN pip install -r requirements.txt
CMD ["python", "/app/app.py"]
```

- In the terminal add the below-mentioned commands:

```
docker build -t hello-minikube . (Press Enter)
docker images (Press Enter)
```

TAG	IMAGE ID	CREATED	SIZE
latest	ea70231c6aca	10 seconds ago	960MB
v0.0.25	8768eddc4356	4 weeks ago	1.1GB
latest	d1165f221234	5 months ago	13.3kB

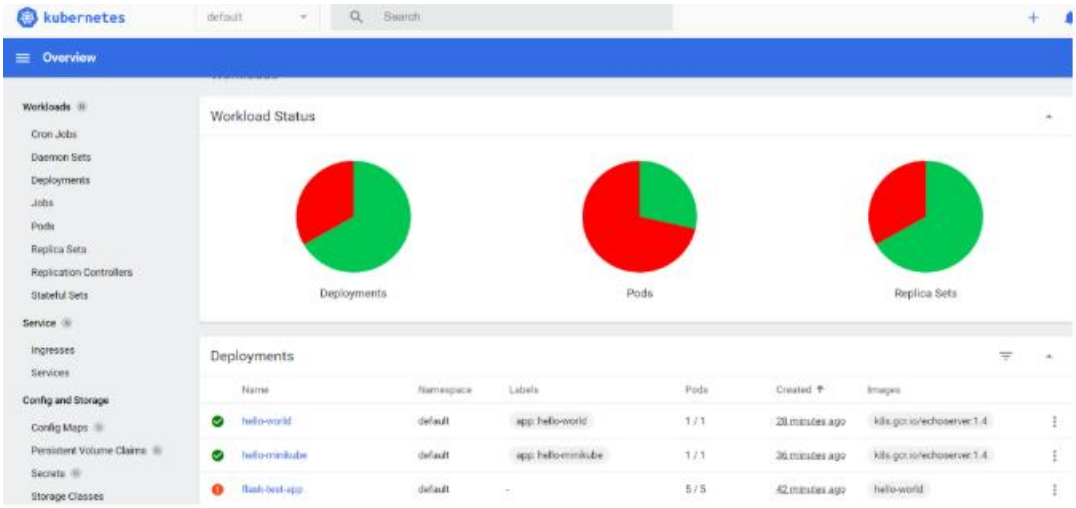
- Now add a new file name it '***deployment.yaml***' and add the below-mentioned code:

```
kind: Service
metadata:
  name: test-service
spec:
  selector:
    app: test-app
  ports:
  - protocol: "TCP"
    port: 6000
    targetPort: 5000
  type: LoadBalancer

---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello-minikube-test-app
spec:
  selector:
    matchLabels:
      app: hello-minikube-test-app
  replicas: 5
  template:
    metadata:
      labels:
        app: hello-minikube-test-app
    spec:
      containers:
      - name: hello-minikube-test-app
        image: hello-minikube
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 5000
```

```
minikube start (Press Enter)
kubectl apply -f deployment.yaml (Press Enter)
minikube dashboard (Press Enter)
```

Once the above commands are executed you will see your docker app has been deployed to Kubernetes and minikube dashboard will be opened on the browser.

Output: Therefore the docker app has been deployed to Kubernetes successfully.


kubernetes
default
Search
Overview
Workloads
Cron Jobs
Daemon Sets
Deployments
Jobs
Pods
Replica Sets
Replication Controllers
Stateful Sets
Service
Ingresses
Services
Config and Storage
Config Maps
Persistent Volume Claims
Secrets
Storage Classes
Workload Status
Deployments
Pods
Replica Sets
Deployments
Name
Namespace
Labels
Pods
Created
Images
hello-world
default
app: hello-world
1 / 1
28 minutes ago
k8s.gcr.io/echoserver:1.4
hello-minikube
default
app: hello-minikube
1 / 1
36 minutes ago
k8s.gcr.io/echoserver:1.4
flask-test-app
default
-
5 / 5
42 minutes ago
hello-world

WEEK -9

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

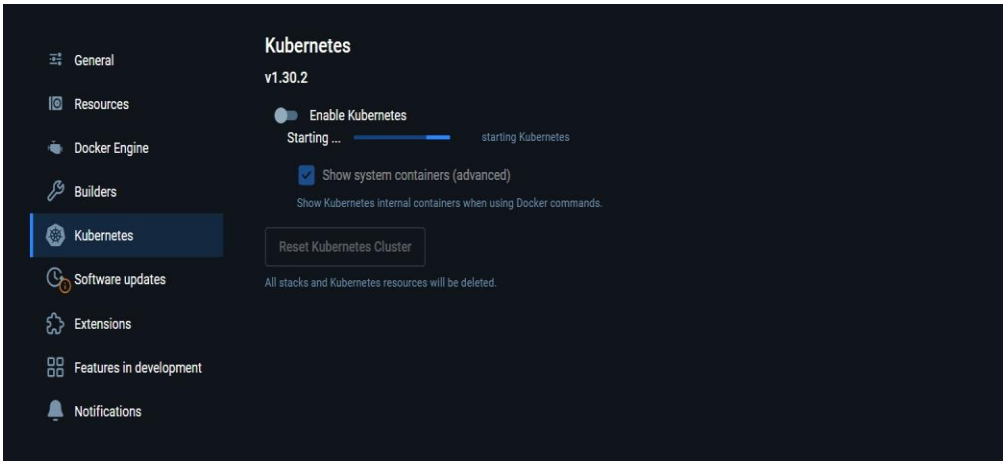
Step1:

1 Clone this repository to your local repository https://github.com/shiv4j/kube

2 Make sure that cloned repository consist of “node-app-deployment.yaml”,“node-app-service.yaml” files in folder.

3 Push this local repository to github (or) you can fork that repository from https://github.com/shiv4j /kube.git

4 After completion of pushing or forking of kube foleder into your github repository you should able to see like this that contains all the files.

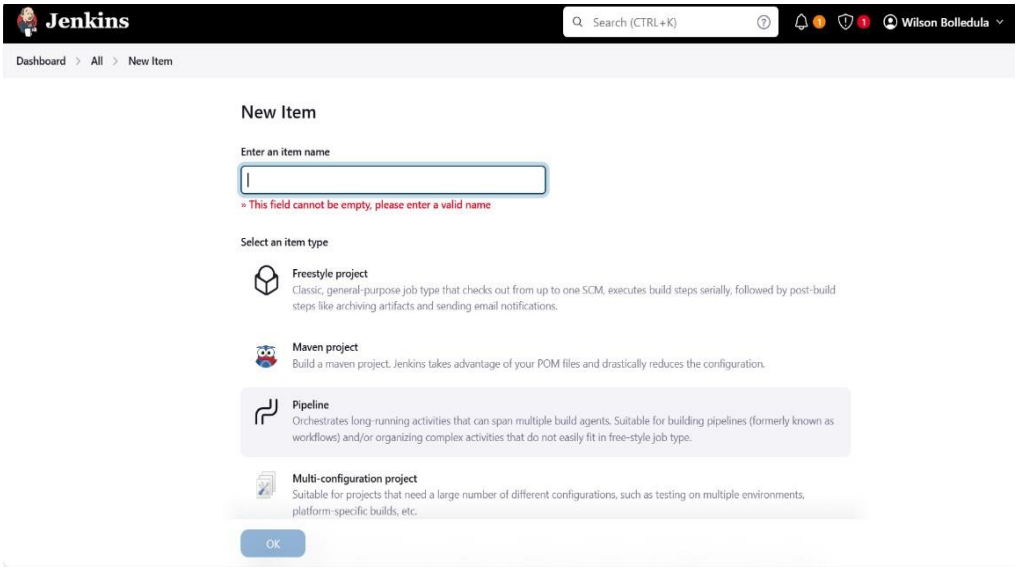


Step-2:

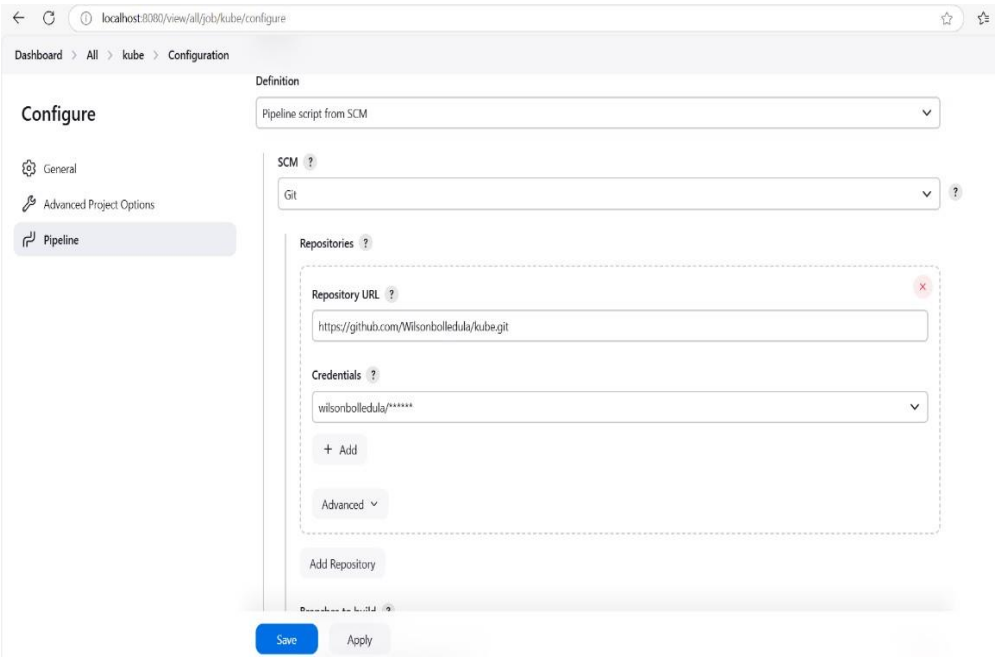
> Push your github repository to the Jenkins.

> click on new item

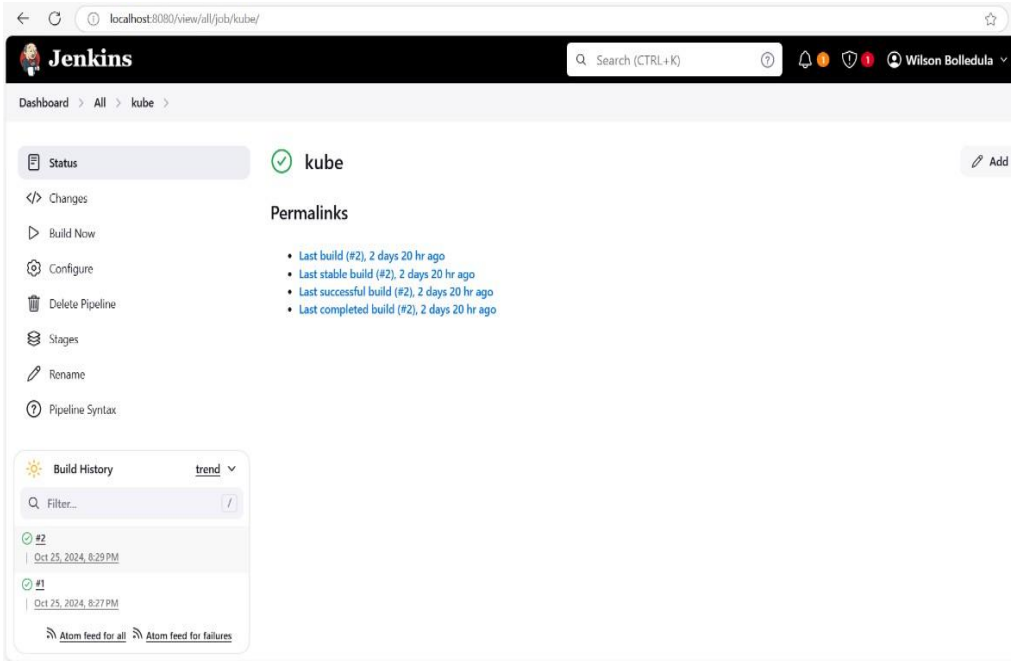
> Enter a name , select pipeline project and click on ok



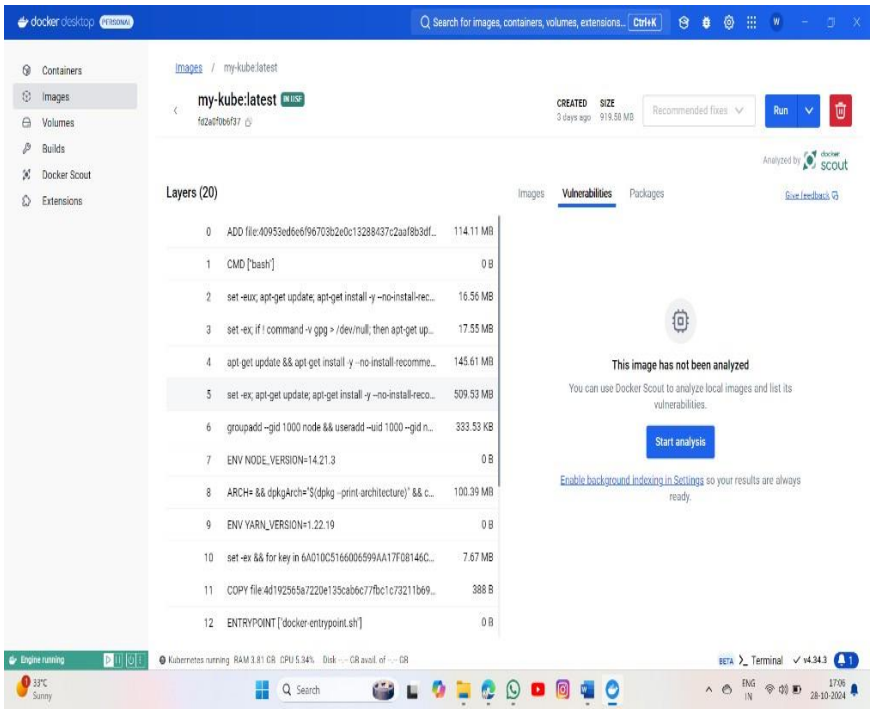
> In the configure paste your repository url and specify your branch whether itis main or master based on your github repository after that apply and save it.



> after creation of your Jenkins project build it, the build should be shown ingreen colour

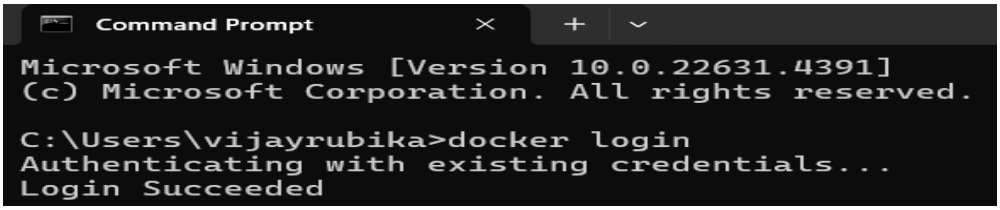


> you will get docker image for this project , like showed in the below



Step-3:

- Push the docker image into dockerhub
- open command prompt and run the command "**docker login**"



> tag your iamge using this syntax

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

yourusername/yourrepo:<tag> for example: docker tag my-kube:latest wilsonbolledula/my-kube:latest here username is your dockerhub account username

>push the image to dockerhub

docker push

yourusername/yourrepo:<tag> example:

docker push wilsonbolledula/my-kube:latest

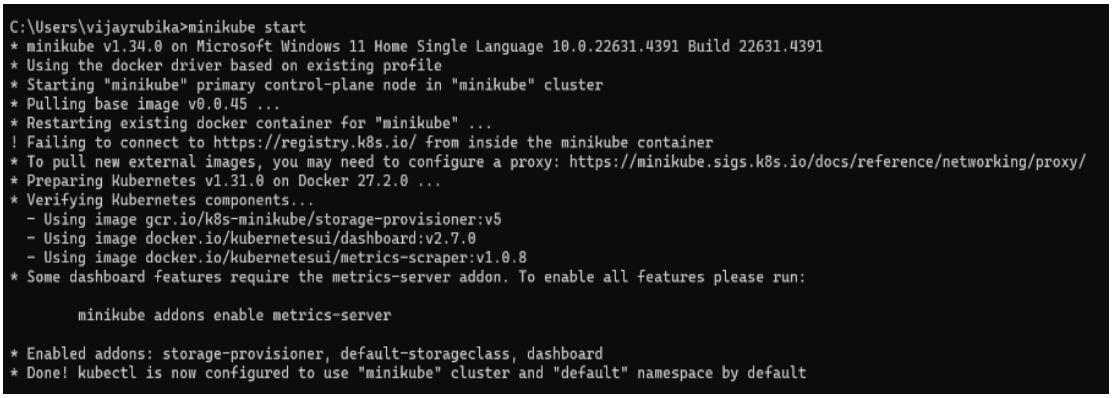
Step-4:

> Start the Kubernetes

Syntax **:**

minikube

start



>Apply node-app-deployment.yaml file

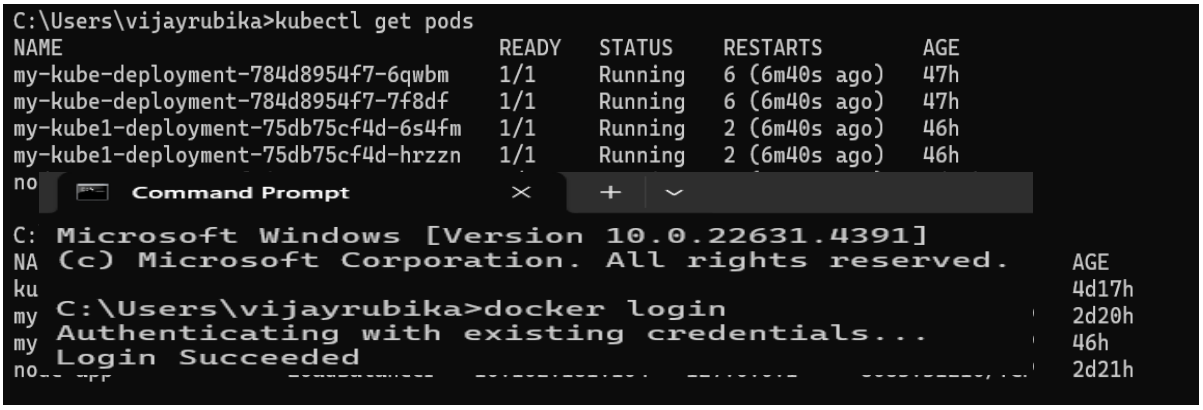
Syntax: kubectl apply -f node-app-deployment.yaml

> Apply node-app-service.yaml file

Syntax: kubectl apply -f node-app-service.yaml

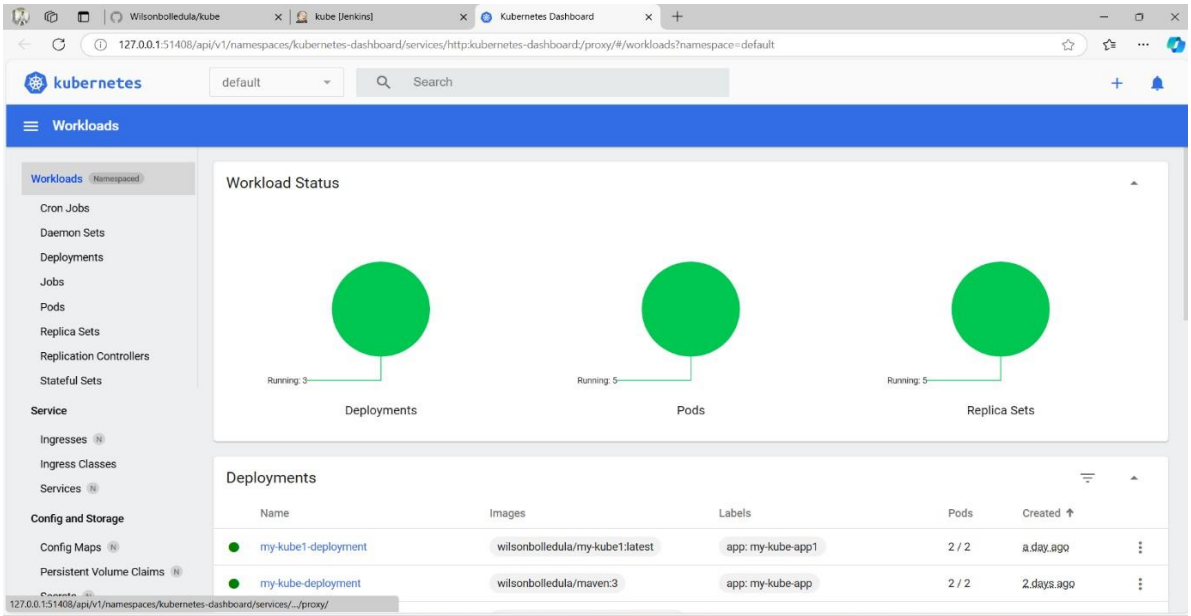
>that will apply to the deployments and services

>To check that type command “kubectl get pods” and “kubectl get service”



To open the Kubernetes dashboard type = “**minikube dashboard**”

That will open Kubernetes dashboard automatically on your default b r o w s e r



WEEK – 10

Install and Explore Selenium for automated testing

Step 1: Install Java Development Kit (JDK)

18. Download the latest JDK from the official Oracle website:
https://www.oracle.com/java/technologies/javase-downloads.html

19. Install the JDK by following the installation wizard.

20. Set up the JAVA_HOME environment variable:

- Press Win + S and search for **Environment Variables**.
- Click **Edit the system environment variables**.
- In **System Properties**, click **Environment Variables…**.
- Add a new **System variable** named JAVA_HOME pointing to your JDK installation directory.
- Edit the **Path** variable and add %JAVA_HOME%\bin.

21. Verify Java installation:

```
java -version
```

Step 2: Install Eclipse IDE (Optional)

22. Download Eclipse IDE from:
https://www.eclipse.org/downloads/

23. Install Eclipse by following the installation wizard.

24. Launch Eclipse IDE.

Step 3: Download Selenium WebDriver

25. Go to the official Selenium website:
https://www.selenium.dev/downloads/

26. Under **Selenium Client & WebDriver Language Bindings**, download the Java bindings.

27. Extract the downloaded ZIP file to a known location.

Step 4: Configure Selenium in Eclipse (Optional)

28. In Eclipse IDE:

- Create a new Java Project.
- Right-click the project and select **Build Path > Configure Build Path…**.
- Go to the **Libraries** tab.
- Click **Add External JARs…**.
- Add all the JAR files from the extracted Selenium folder.

Step 5: Write and Run Selenium Test Cases

29. Create a new Java Class.

30. Write your Selenium test script using WebDriver.

31. Example basic code:

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

public class SeleniumTest {
   public static void main(String[] args) {
      System.setProperty("webdriver.chrome.driver", "path_to_chromedriver.exe");
      WebDriver driver = new ChromeDriver();
      driver.get("https://www.google.com");
      driver.quit();
   }
}
```

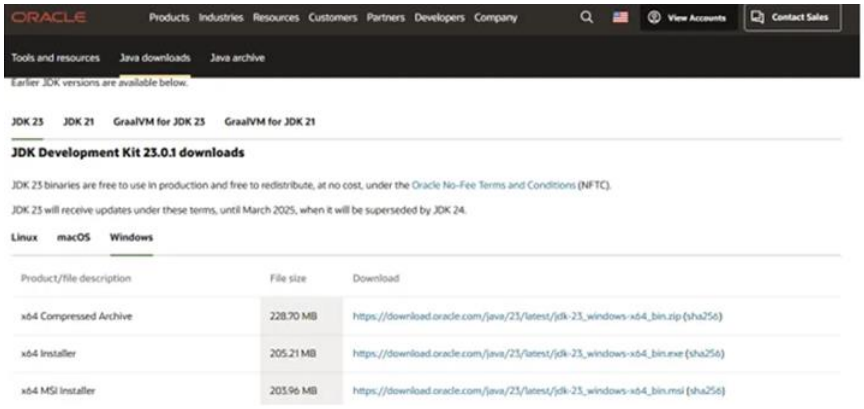
Step 6: Download ChromeDriver

32. Go to:
https://sites.google.com/chromium.org/driver/

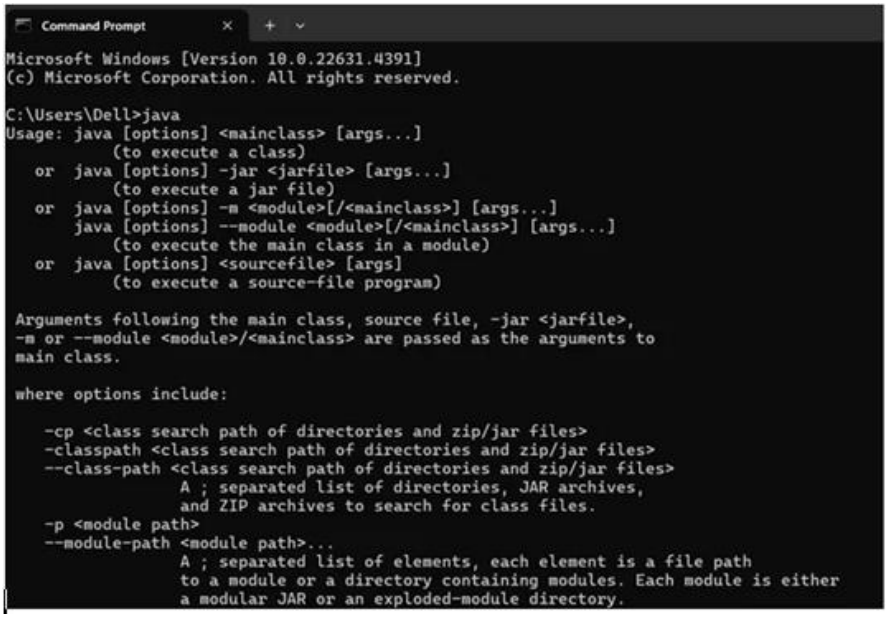
33. Download the version matching your Chrome browser.

34. Place chromedriver.exe in a known directory and reference its path in your Selenium script.

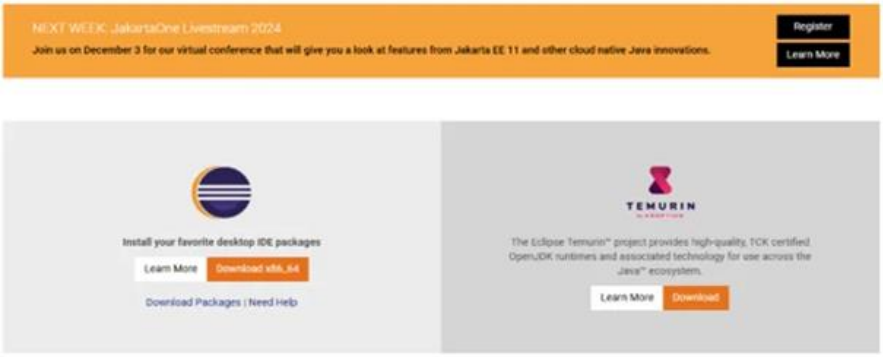
Selenium is now installed and ready for use on your Windows 11 system.

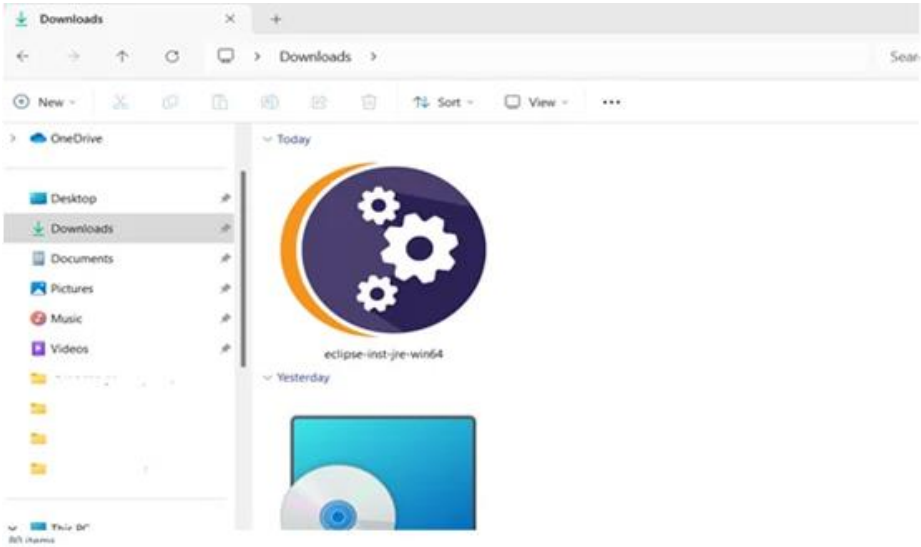

ORACLE
Products Industries Resources Customers Partners Developers Company
View Accounts
Contact Sales
Tools and resources
Java downloads
Java archive
Earlier JDK versions are available below.
JDK 23
JDK 21
GraalVM for JDK 23
GraalVM for JDK 21
JDK Development Kit 23.0.1 downloads
JDK 23 binaries are free to use in production and free to redistribute, at no cost, under the Oracle No-Fee Terms and Conditions (NFTC).
JDK 23 will receive updates under these terms, until March 2025, when it will be superseded by JDK 24.
Linux
macOS
Windows
Product/file description
File size
Download
x64 Compressed Archive
228.70 MB
https://download.oracle.com/java/23/latest/jdk-23_windows-x64_bin.zip (sha256)
x64 Installer
205.21 MB
https://download.oracle.com/java/23/latest/jdk-23_windows-x64_bin.exe (sha256)
x64 MSI Installer
203.96 MB
https://download.oracle.com/java/23/latest/jdk-23_windows-x64_bin.msi (sha256)


Java(TM) SE Development Kit 23.0.1 (64-bit) - Complete
Java
ORACLE
Java(TM) SE Development Kit 23.0.1 (64-bit) Successfully Installed
Click Next Steps to access tutorials, API documentation, developer guides, release notes and more to help you get started with the JDK.
Next Steps
Close


Command Prompt
Microsoft Windows [Version 10.0.22631.4391]
(c) Microsoft Corporation. All rights reserved.
C:\Users\Dell>java
Usage: java [options] <mainclass> [args...]
(to execute a class)
or java [options] -jar <jarfile> [args...]
(to execute a jar file)
or java [options] -m <module>[/<mainclass>] [args...]
java [options] --module <module>[/<mainclass>] [args...]
(to execute the main class in a module)
or java [options] <sourcefile> [args]
(to execute a source-file program)
Arguments following the main class, source file, -jar <jarfile>,
-m or --module <module>/<mainclass> are passed as the arguments to
main class.
where options include:
-cp <class search path of directories and zip/jar files>
-classpath <class search path of directories and zip/jar files>
--class-path <class search path of directories and zip/jar files>
A ; separated list of directories, JAR archives,
and ZIP archives to search for class files.
-p <module path>
--module-path <module path>...
A ; separated list of elements, each element is a file path
to a module or a directory containing modules. Each module is either
a modular JAR or an exploded-module directory.

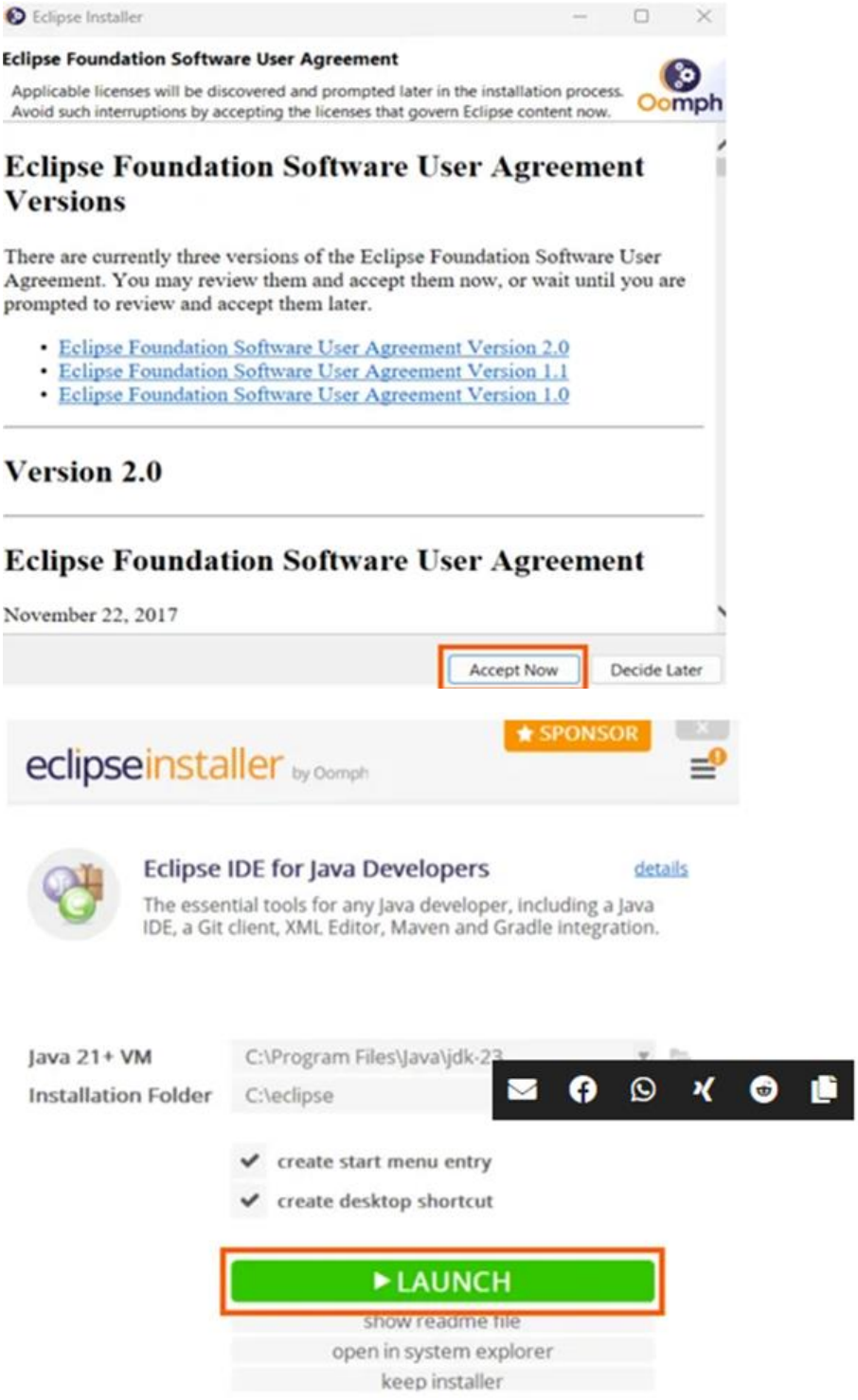
Download Eclipse Technology that is right for you

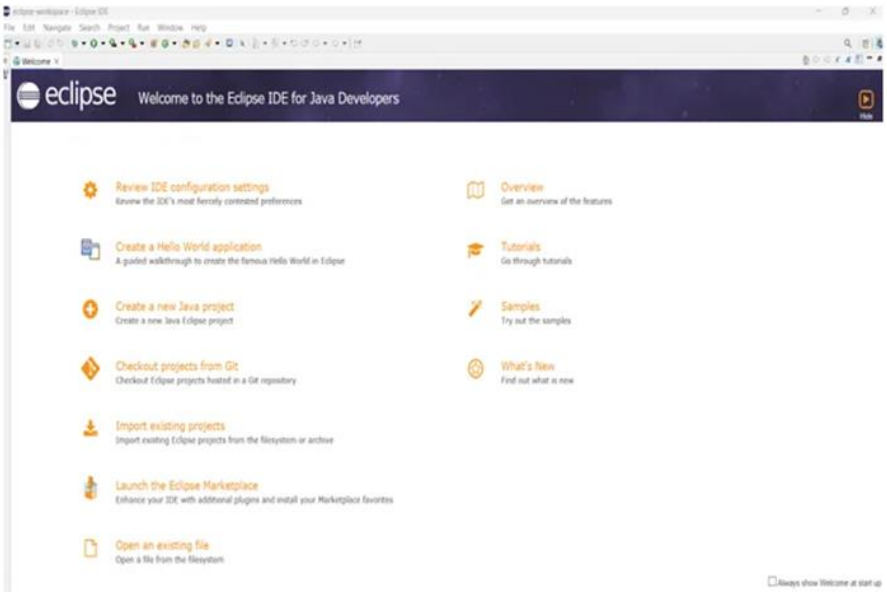




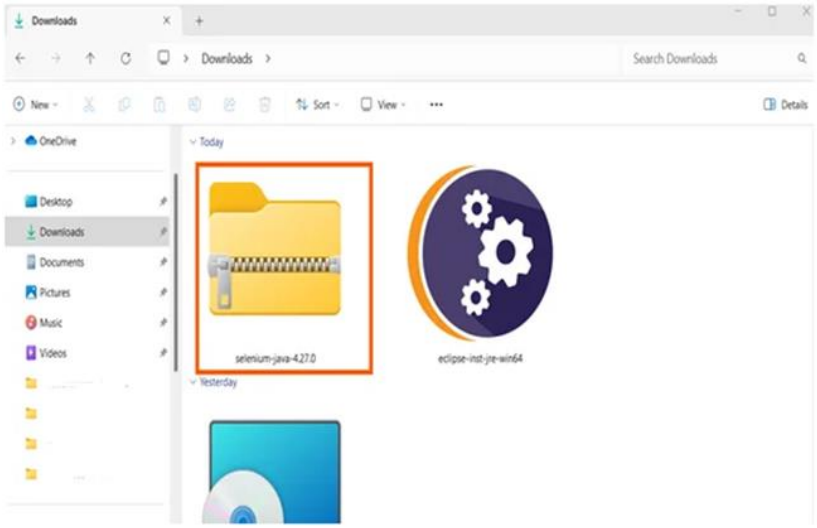

eclipseinstaller by Oomph
SPONSOR
type filter text
Eclipse IDE for Java Developers
The essential tools for any Java developer, including a Java IDE, a Git client, XML Editor, Maven and Gradle integration
Eclipse IDE for Enterprise Java and Web Develc
Tools for developers working with Java and Web applications, including a Java IDE, tools for JavaScript, TypeScript, JavaServer...
Eclipse IDE for C/C++ Developers
An IDE for C/C++ developers.
Eclipse IDE for Embedded C/C++ Developers
An IDE for Embedded C/C++ developers. It includes managed cross build plug-ins (Arm and RISC-V) and debug plug-ins...
Eclipse IDE for PHP Developers


Artifact download is progressing very slowly from the follo wing hosts: https://download.eclipse.org, https://mirrors.nju .edu.cn, https://ftp.yz.yamagata-u.ac.jp
Eclipse IDE for Java Developers
details
The essential tools for any Java developer, including a Java IDE, a Git client, XML Editor, Maven and Gradle integration.
Java 21+ VM
jres/21/updates/release/latest
Installation Folder
:\Users\HP\eclipse\java-2025-09
create start menu entry
create desktop shortcut
INSTALLING
BACK

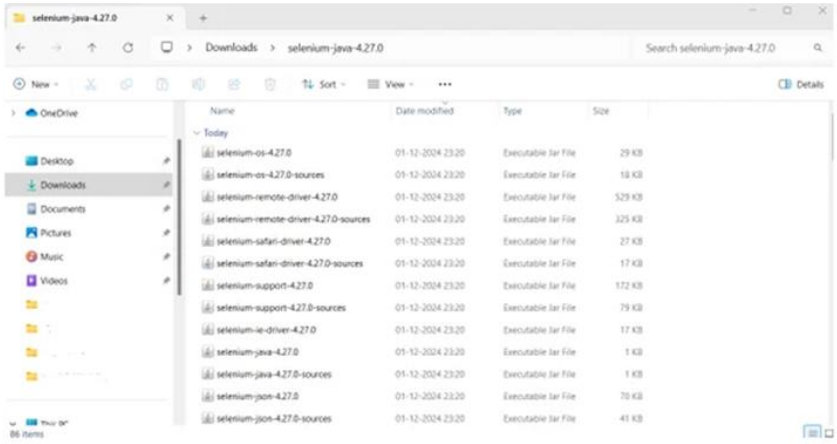

Eclipse Installer
Eclipse Foundation Software User Agreement
Applicable licenses will be discovered and prompted later in the installation process.
Avoid such interruptions by accepting the licenses that govern Eclipse content now.
Oomph
Eclipse Foundation Software User Agreement Versions
There are currently three versions of the Eclipse Foundation Software User Agreement. You may review them and accept them now, or wait until you are prompted to review and accept them later.
Eclipse Foundation Software User Agreement Version 2.0
Eclipse Foundation Software User Agreement Version 1.1
Eclipse Foundation Software User Agreement Version 1.0
Version 2.0
Eclipse Foundation Software User Agreement
November 22, 2017
Accept Now
Decide Later
eclipseinstaller by Oomph
SPONSOR
Eclipse IDE for Java Developers
details
The essential tools for any Java developer, including a Java IDE, a Git client, XML Editor, Maven and Gradle integration.
Java 21+ VM
C:\Program Files\Java\jdk-23
Installation Folder
C:\eclipse
create start menu entry
create desktop shortcut
LAUNCH
show readme file
open in system explorer
keep installer

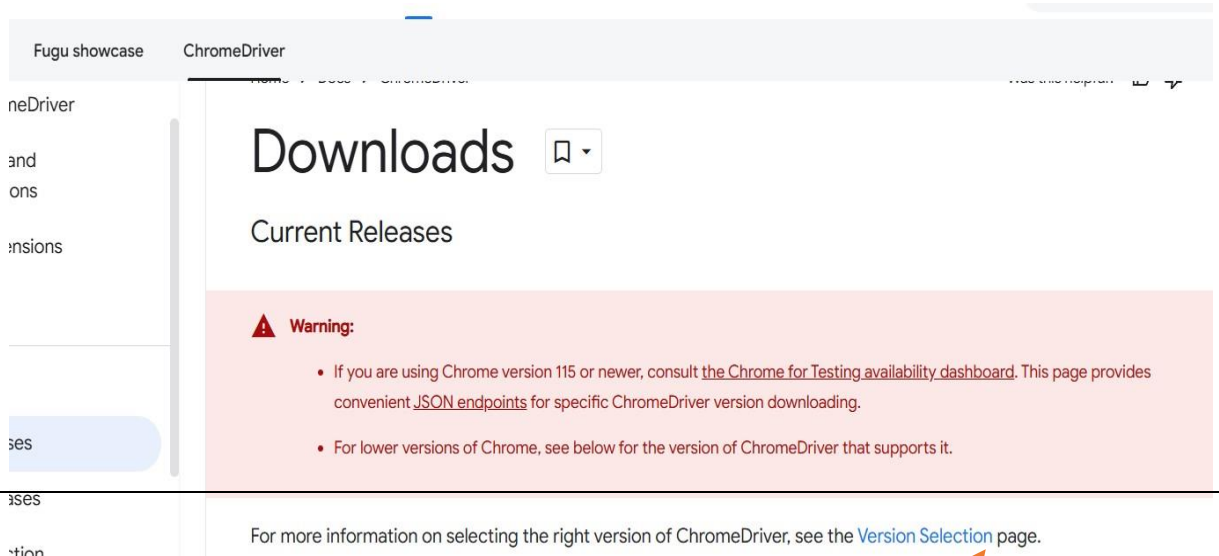


Install Selenium WebDriver –

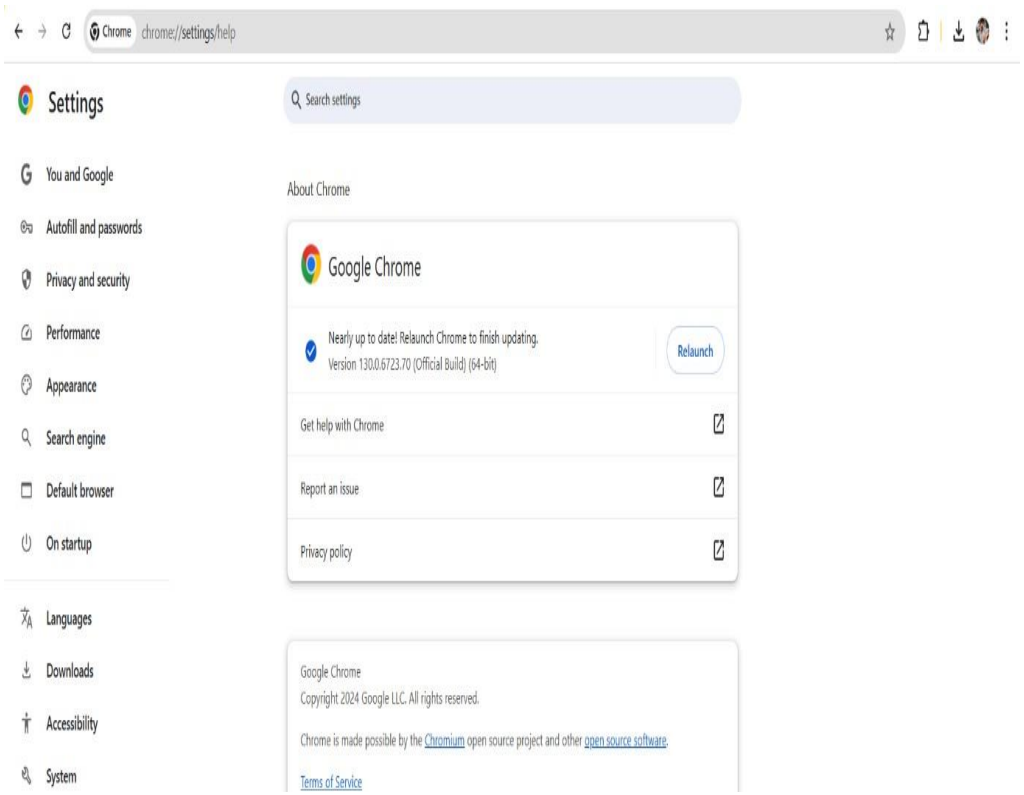


When the zip file is unzipped i.e. all the file present inside that zip are extracted, we will see the list of various JAR file which we will be using while configuring Selenium Webdriver and Eclipse IDE.

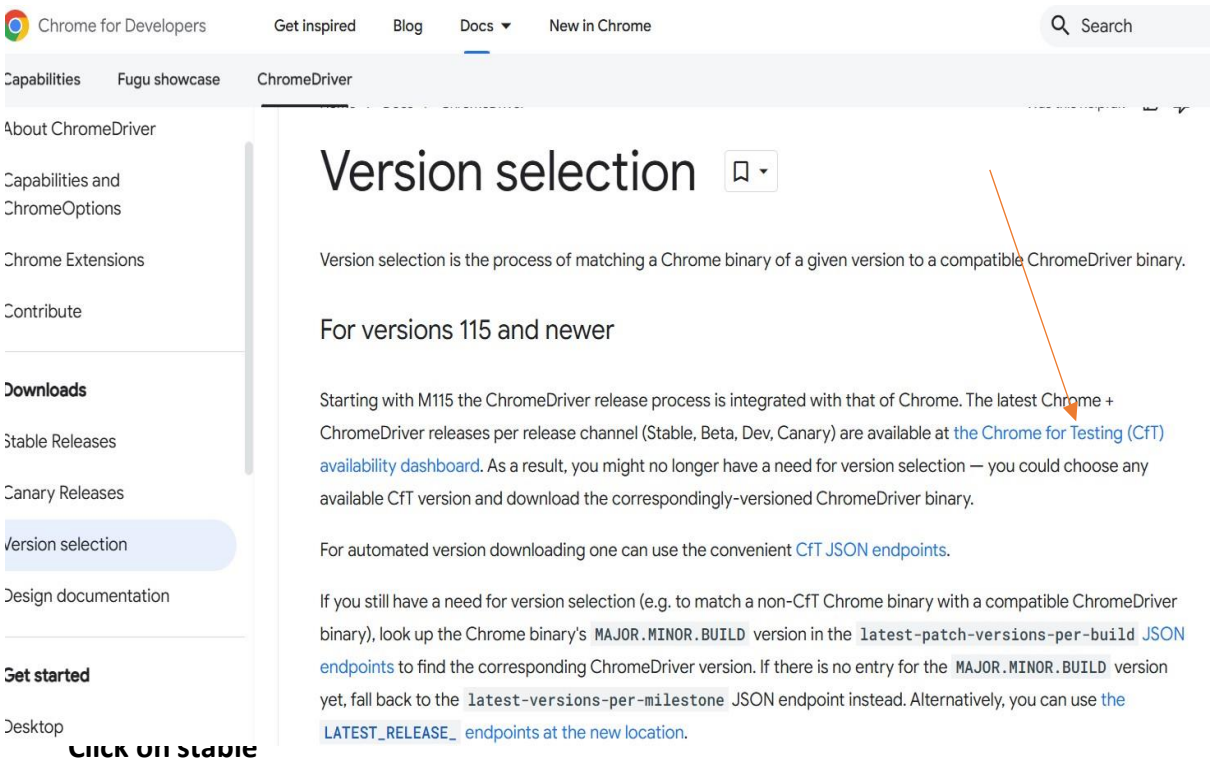




- **Click on version selection and check your version in chrome by chrome://settings/help**



- **Click on** the Chrome for Testing (CfT) availability dashboard.



➢ **Click on stable**

Chrome for Testing availability

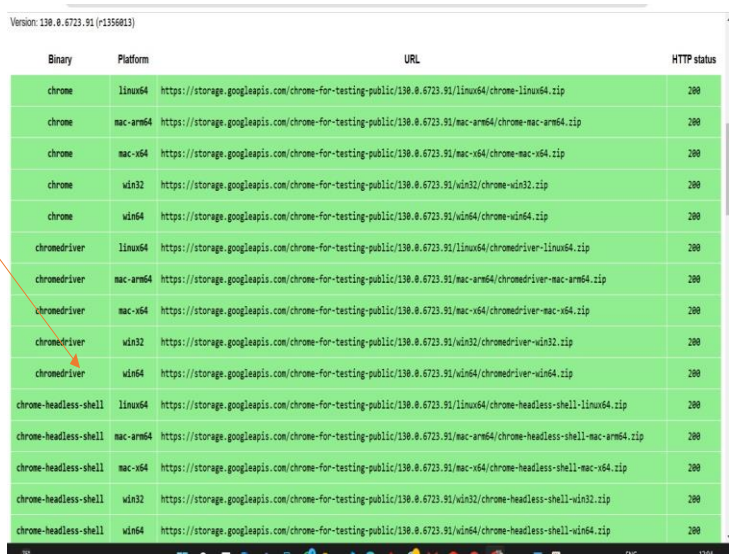
This page lists the latest available cross-platform Chrome for Testing versions and assets per Chrome release channel.

Consult our JSON API endpoints if you're looking to build automated scripts based on Chrome for Testing release data.

Last updated @ 2024-10-30T05:09:34.045Z

Channel	Version	Revision	Status
Stable	130.0.6723.91	r1356013	✓
Beta	131.0.6778.13	r1368529	✓
Dev	132.0.6793.2	r1372444	✓
Canary	132.0.6806.0	r1375157	✓

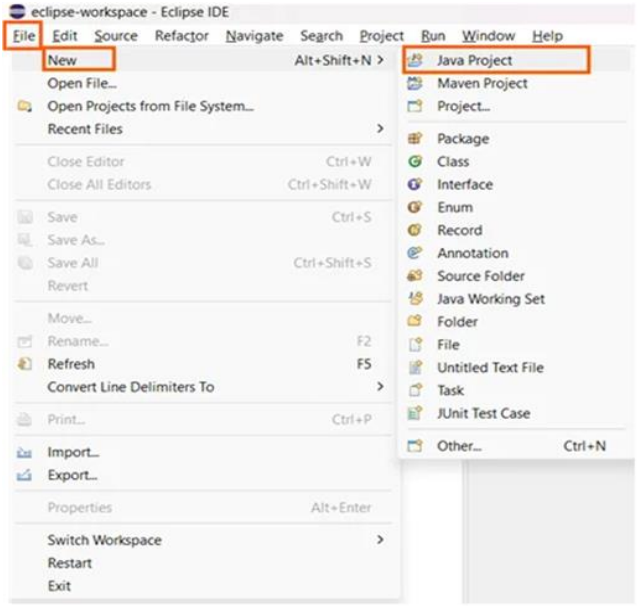
➢ **Select according to you os and Copy and paste chromedriver link in chromeit will download automatically**

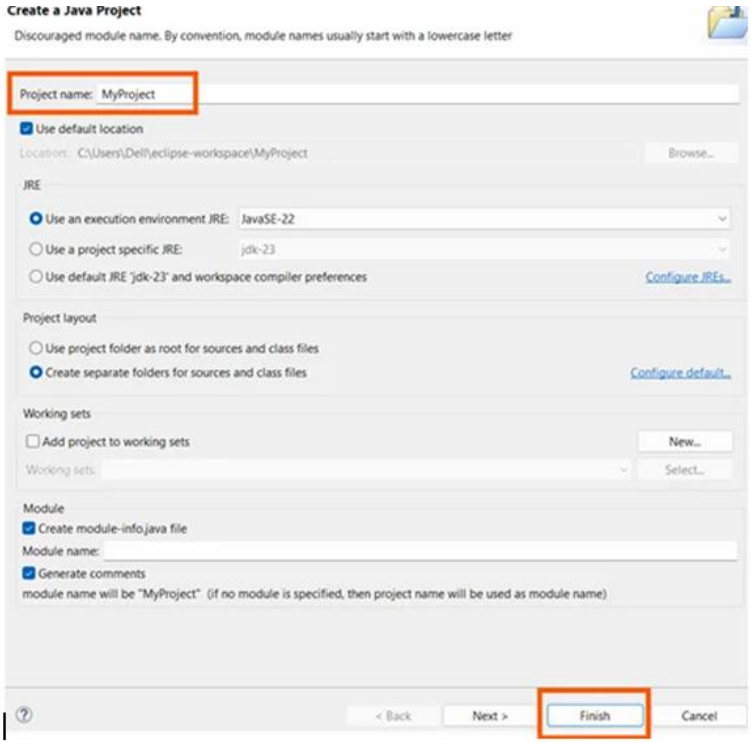


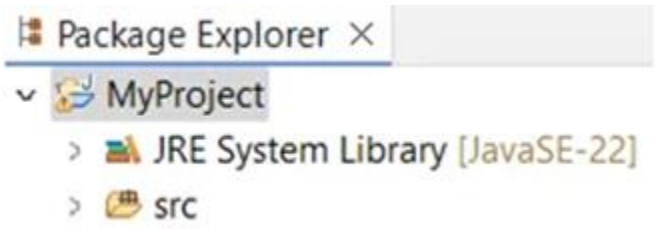
Version: 130.0.6723.91 (r1356013)

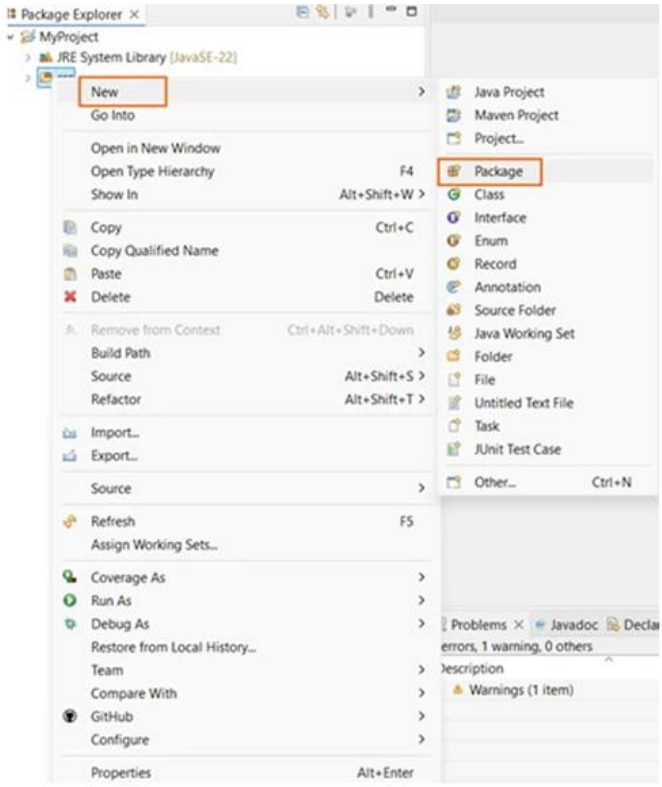
Binary	Platform	URL	HTTP status
chrome	linux64	https://storage.googleapis.com/chrome-for-testing-public/130.0.6723.91/linux64/chrome-linux64.zip	200
chrome	mac-arm64	https://storage.googleapis.com/chrome-for-testing-public/130.0.6723.91/mac-arm64/chrome-mac-arm64.zip	200
chrome	mac-x64	https://storage.googleapis.com/chrome-for-testing-public/130.0.6723.91/mac-x64/chrome-mac-x64.zip	200
chrome	win32	https://storage.googleapis.com/chrome-for-testing-public/130.0.6723.91/win32/chrome-win32.zip	200
chrome	win64	https://storage.googleapis.com/chrome-for-testing-public/130.0.6723.91/win64/chrome-win64.zip	200
chromedriver	linux64	https://storage.googleapis.com/chrome-for-testing-public/130.0.6723.91/linux64/chromedriver-linux64.zip	200
chromedriver	mac-arm64	https://storage.googleapis.com/chrome-for-testing-public/130.0.6723.91/mac-arm64/chromedriver-mac-arm64.zip	200
chromedriver	mac-x64	https://storage.googleapis.com/chrome-for-testing-public/130.0.6723.91/mac-x64/chromedriver-mac-x64.zip	200
chromedriver	win32	https://storage.googleapis.com/chrome-for-testing-public/130.0.6723.91/win32/chromedriver-win32.zip	200
chromedriver	win64	https://storage.googleapis.com/chrome-for-testing-public/130.0.6723.91/win64/chromedriver-win64.zip	200
chrome-headless-shell	linux64	https://storage.googleapis.com/chrome-for-testing-public/130.0.6723.91/linux64/chrome-headless-shell-linux64.zip	200
chrome-headless-shell	mac-arm64	https://storage.googleapis.com/chrome-for-testing-public/130.0.6723.91/mac-arm64/chrome-headless-shell-mac-arm64.zip	200
chrome-headless-shell	mac-x64	https://storage.googleapis.com/chrome-for-testing-public/130.0.6723.91/mac-x64/chrome-headless-shell-mac-x64.zip	200
chrome-headless-shell	win32	https://storage.googleapis.com/chrome-for-testing-public/130.0.6723.91/win32/chrome-headless-shell-win32.zip	200
chrome-headless-shell	win64	https://storage.googleapis.com/chrome-for-testing-public/130.0.6723.91/win64/chrome-headless-shell-win64.zip	200

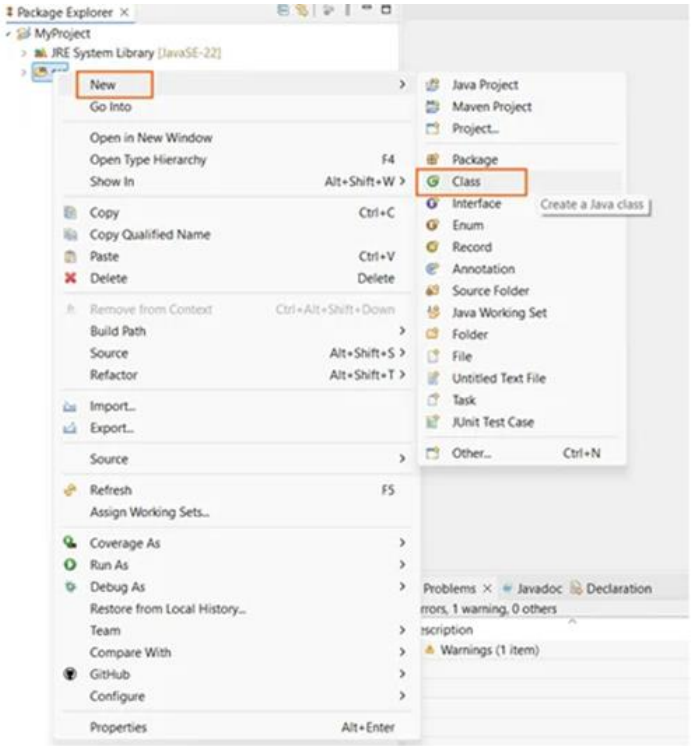
- the file will be downloaded , extract the files and set the path

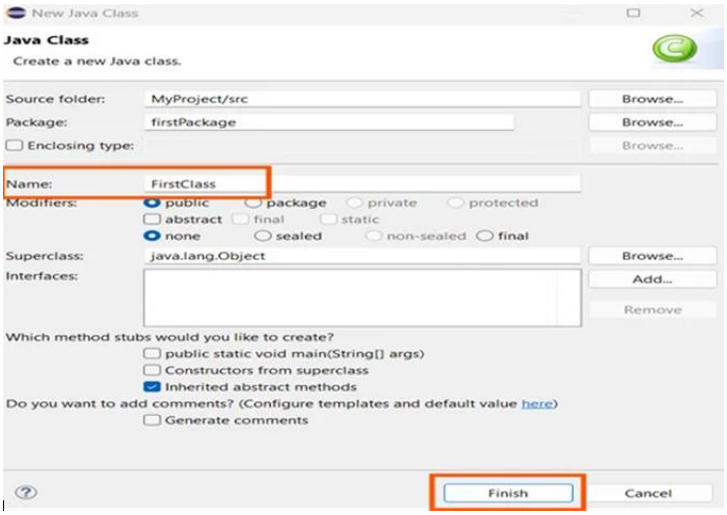


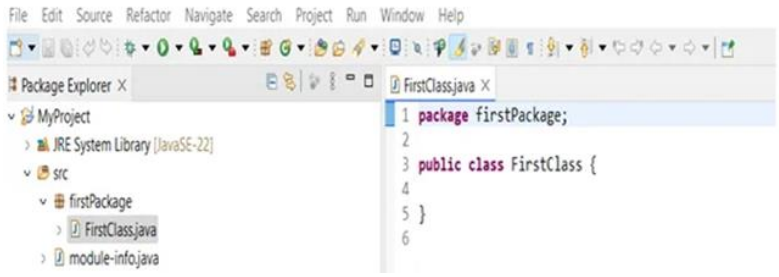

Create a Java Project
Discouraged module name. By convention, module names usually start with a lowercase letter
Project name: MyProject
Use default location
Location: C:\Users\Dell\eclipse-workspace\MyProject
Browse...
JRE
Use an execution environment JRE: JavaSE-22
Use a project specific JRE: jdk-23
Use default JRE 'jdk-23' and workspace compiler preferences
Configure JREs...
Project layout
Use project folder as root for sources and class files
Create separate folders for sources and class files
Configure default...
Working sets
Add project to working sets
New...
Working sets:
Select...
Module
Create module-info.java file
Module name:
Generate comments
module name will be "MyProject" (if no module is specified, then project name will be used as module name)
< Back
Next >
Finish
Cancel


Package Explorer
MyProject
JRE System Library [JavaSE-22]
src


Package Explorer
MyProject
JRE System Library [JavaSE-22]
New
Go Into
Open in New Window
Open Type Hierarchy F4
Show In Alt+Shift+W
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Remove from Context Ctrl+Alt+Shift+Down
Build Path
Source Alt+Shift+S
Refactor Alt+Shift+T
Import...
Export...
Source
Refresh F5
Assign Working Sets...
Coverage As
Run As
Debug As
Restore from Local History...
Team
Compare With
GitHub
Configure
Properties Alt+Enter
Java Project
Maven Project
Project...
Package
Class
Interface
Enum
Record
Annotation
Source Folder
Java Working Set
Folder
File
Untitled Text File
Task
JUnit Test Case
Other... Ctrl+N
Problems
Javadoc
errors, 1 warning, 0 others
Description
Warnings (1 item)

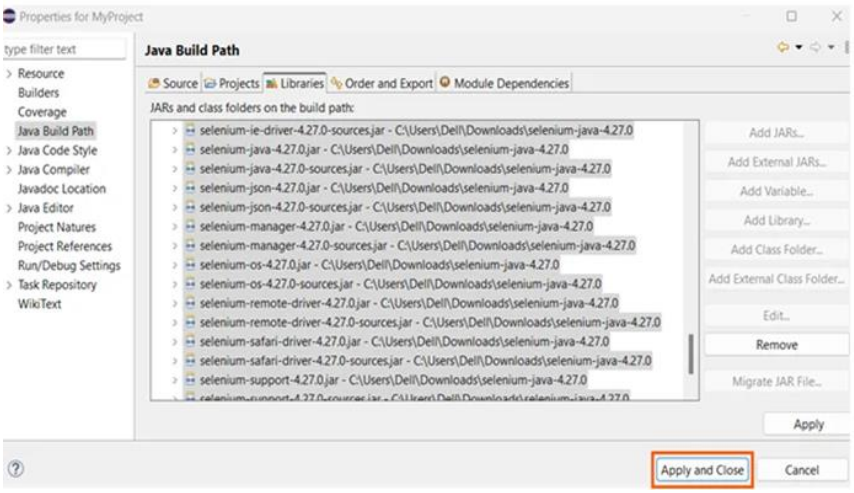

Package Explorer
MyProject
JRE System Library [JavaSE-22]
New
Go Into
Open in New Window
Open Type Hierarchy F4
Show In Alt+Shift+W
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Remove from Context Ctrl+Alt+Shift+Down
Build Path
Source Alt+Shift+S
Refactor Alt+Shift+T
Import...
Export...
Source
Refresh F5
Assign Working Sets...
Coverage As
Run As
Debug As
Restore from Local History...
Team
Compare With
GitHub
Configure
Properties Alt+Enter
Java Project
Maven Project
Project...
Package
Class
Interface
Create a Java class
Enum
Record
Annotation
Source Folder
Java Working Set
Folder
File
Untitled Text File
Task
JUnit Test Case
Other... Ctrl+N
Problems
Javadoc
Declaration
rrors, 1 warning, 0 others
Warnings (1 item)

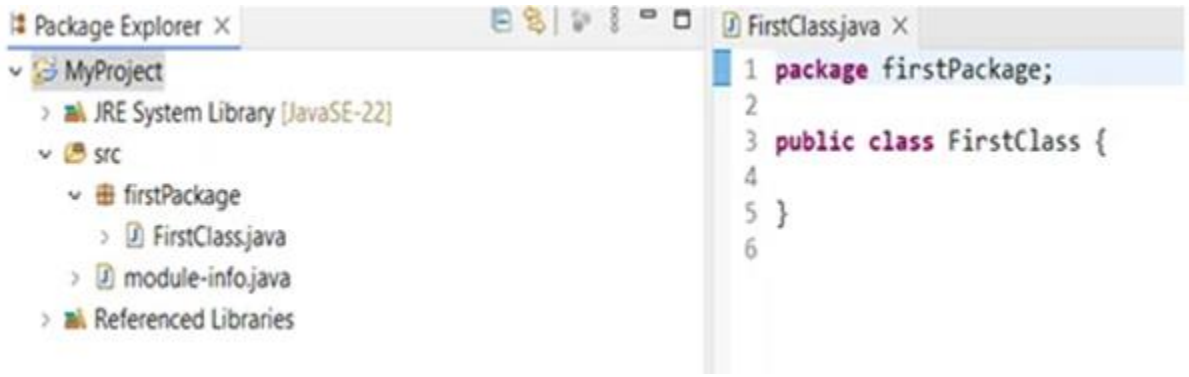




- Create a new Java Project.
- Right-click the project and select **Build Path > Configure Build Path…**.
- Go to the **Libraries** tab.
- Click **Add External JARs…**.

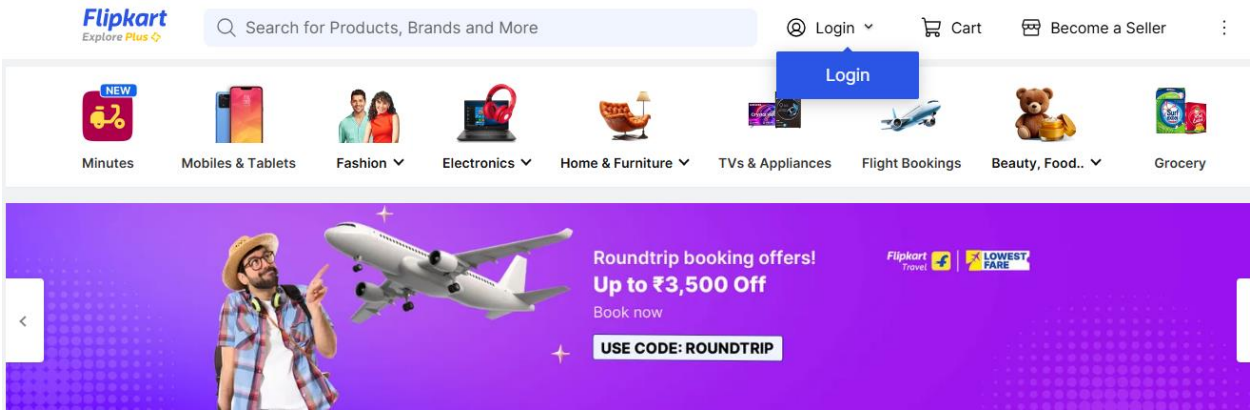
Add all the JAR files from the extracted Selenium folder.





Run the code:

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

public class SeleniumTest {
   public static void main(String[] args) {
      System.setProperty("webdriver.chrome.driver", "path_to_chromedriver.exe");
      WebDriver driver = new ChromeDriver();
      driver.get("https://www.flipkart.com");
      driver.quit();
   }
}
```



WEEK -11

Write a simple program in JavaScript and perform testing using Selenium

- Open VS code and create html and javascript files
- Create html file and paste this code :

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Sum Calculator</title>
    <style>
        body{
text-align: center;
        }
    </style>
</head>
<body>
    <h1>Sum Calculator</h1>
    <input type="number" id="num1" placeholder="Enter first number">
    <input type="number" id="num2" placeholder="Enter second number">
    <button id="add">Add</button>
    <p>Result: <span id="result">0</span></p>

    <script>
        function
            calculateSum(a,
            b)
```

```
        {    return a + b;
        }

        document.getElementById('add').addEventListener('click', function()
            { const num1 =
            parseInt(document.getElementById('num1').value, 10);const
            num2 = parseInt(document.getElementById('num2').value, 10);
            const result = calculateSum(num1, num2);
            document.getElementById('result').textContent = result;
        });
    </script>
</body>
</html>
```

1. Create on javascript file and paste this code :

```
const webdriver = require('selenium-
    webdriver');const assert =
    require('assert');

const driver = new webdriver.Builder().forBrowser('chrome').build();


async function
    runTest() {
    try {
    // Open the HTML file in the browser
    await driver.get('file://' +_____dirname + '/index.html');
    // Find the input elements and enter values
```

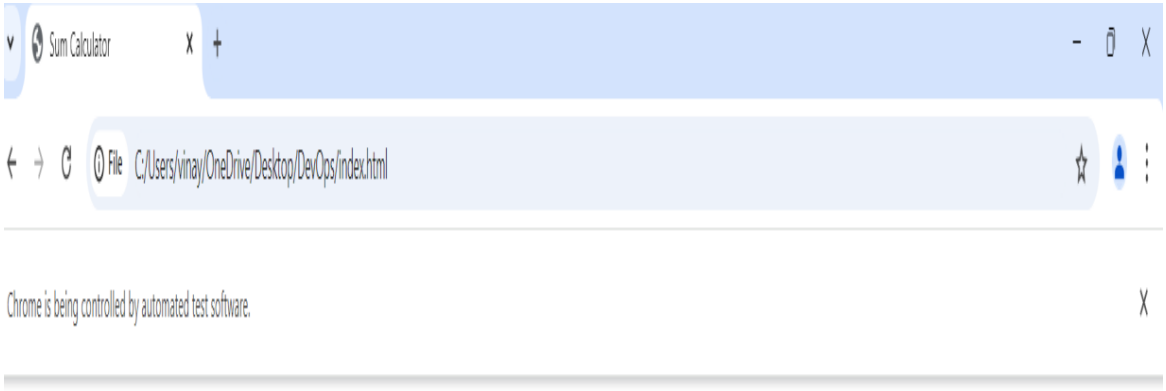
```
const num1 = await
driver.findElement(webdriver.By.id('num1'));await
num1.sendKeys('50');

const num2 = await driver.findElement(webdriver.By.id('num2'));
await num2.sendKeys('10');

// Click the "Add" button
const addButton = await driver.findElement(webdriver.By.id('add'));
await addButton.click();

// Get the result text and verify it
const result = await
driver.findElement(webdriver.By.id('result'));const text = await
result.getText();

assert.strictEqual(text, '60', 'Sum calculation is incorrect');
console.log('Test passed: Sum is correct');
} catch (error) {
    console.error('Test failed:', error);
} finally {
    // Wait for user input to close the
    browserconsole.log('Press any key
    to exit...');
    process.stdin.setRawMode(true);
    process.stdin.resume();
    process.stdin.on('data', async () =>
    {
        await
        driver.
        quit();
```

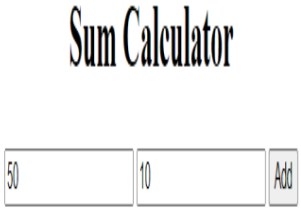
```
            proces
            s.exit(
            0);
        });
    }
}

runTest();
```

2. Open terminal

1. **npm install selenium-webdriver**
2. node test.js

Output:





Sum Calculator

Result: 60

WEEK -12

Develop test cases for the above containerized application using selenium.

Prerequisites :

Docker Desktop

Python (for running scripts

locally if needed)Selenium

(pip install selenium)

Steps for execution:

Clone this repository to your local repository https://github.com/Srivaishnavi08/tests

Or follow the below

stepsStep-1:

Create a directory named selenium-test and navigate to the current directory

path.tests/

```
├── Dockerfile
├── index.html
├── SeleniumTest.py
└── docker-compose.yml
```

Step 2: Create the index.html File

This is your sample web page that Selenium will interact with.

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Selenium Login Example</title>
</head>
<body>
    <!-- Homepage -->
    <div id="homepage">
        <h1>Welcome!!!!!!</h1>
        <!-- Get Started Free Button -->
        <a href="#loginPage" id="get-started" onclick="navigateToLogin()">Get
        started</a>
</div>
    <!-- Login Page -->
    <div id="loginPage" style="display: none;">
        <h2>Login to the Page</h2>
        <form onsubmit="return validateLogin()">
            <label for="user_email_login">Email:</label>
            <input type="email" id="user_email_login" name="user_email_login"
            required>
            <br><br>
```

```
            <label for="user_password">Password:</label>

            <input type="password" id="user_password" name="user_password"
            required>

            <br><br>

            <button type="submit" name="commit">Login</button>

        </form>

        <p id="error-message" style="color: red; display: none;">Invalid
credentials, please tryagain.</p>

    </div>

    <!-- Dashboard Section (only shown after successful login) -->

    <div id="dashboard" style="display: none;">

        <h2>Welcome to Your Dashboard!</h2>

        <p>This is the dashboard area you see after a successful login.</p>

    </div>

    <script>

        // Function to navigate

        to the login page

        function

        navigateToLogin() {

            document.getElementById('homepage').style.display = 'none';

            document.getElementById('loginPage').style.display = 'block';

        }

        // Function to validate
        login credentials
```

```
        function validateLogin()
        {

            const email =
            document.getElementById('user_email_login').value;
            const password =
            document.getElementById('user_password').value;


            if (email === "abc@gmail.com" && password === "password") {

                // Hide login page and display dashboard
                document.getElementById('loginPage').style.di
                splay = 'none';
                document.getElementById('dashboard').style.d
                isplay = 'block';return false; // Prevent actual
                form submission

            } else {

                // Show error message if credentials are incorrect
                document.getElementById('error-
                message').style.display = 'block';return false; //
                Prevent actual form submission

            }

        }
    </script>
</body>
    </html>
```

Step 3: Create the Dockerfile

This Dockerfile sets up a simple HTTP server to serve your index.html file.

```
    # Use the official Python image as the base image FROM python:3.9
```

 # Set the working directory

in the containerWORKDIR

/app

 # Copy the index.html file

to the containerCOPY

index.html .

 # Expose port 8000 for
the HTTP serverEXPOSE
8000

 # Start a simple HTTP server to serve the
index.html fileCMD ["python", "-m", "http.server",
"8000"]

Step-4: Create the Selenium Test Script

This script will automate testing of your HTML page using Selenium.

from selenium import webdriver

from selenium.webdriver.common.by import By

from selenium.webdriver.support.ui import WebDriverWait

from selenium.webdriver.support import
expected_conditions as ECimport time

print("Test Execution
Started") options =

```
webdriver.ChromeOpti
ons()

options.add_argument('--ignore-
ssl-errors=yes')
options.add_argument('--ignore-
certificate-errors')


# Start the Selenium
WebDriverdriver =
webdriver.Remote(

    command_executor='http://localhost:4444/
    wd/hub',options=options

)


# Maximize the window
size
driver.maximize_window(
) time.sleep(10)

driver.get("http://host.docker.internal:8000") # Access the
local servertime.sleep(10)


try:

    # Wait for the "Get started free" link to be
    clickablelink = WebDriverWait(driver, 30).until(

        EC.element_to_be_clickable((By.LINK_TEXT, "Get started"))
```

```
)
link.click()  # Click the link

time.sleep(10)  # Wait for any resulting page to load

WebDriverWait(driver, 10).until(
   EC.presence_of_element_located((By.ID,
   "user_email_login))

)

WebDriverWait(driver, 10).until(
   EC.presence_of_element_located((By.ID,
   "user_password"))

)
```

 # Enter login credentials

```
username = driver.find_element(By.ID,

"user_email_login")password =

driver.find_element(By.ID,

"user_password") login_button =

driver.find_element(By.NAME, "commit")


username.send_keys("abc@gmail.com") # Replace with

actual usernamepassword.send_keys("password") #

Replace with actual password login_button.click()


# Check for a post-login element (adjust to your page's unique element for
logged-in users)try:

   error_message = WebDriverWait(driver, 10).until(

      EC.visibility_of_element_located((By.ID, "error-message"))

   )
```

```
        time.sleep(10)

        print("Login failed:
    Incorrect credentials")except:

        # No error message found, proceed with checking for
        dashboardWebDriverWait(driver, 10).until(

            EC.visibility_of_element_located((By.ID, "dashboard"))  # Replace
    with actual post-login element ID)

        print("Login Successful!")
    except Exception as e:
        print(f"An error occurred while trying to click the link: {e}")
    finally:

        # Ensure the browser quits after execution
        driver.quit()

        print("Test Execution Completed!")
```

Step 5: **Create the docker-compose.yml File**

This file defines two services: your HTML server and the Selenium Chrome container.

.

```
    build:
      context: .
      dockerfile: Dockerfile
    container_name: html-server
    ports:
      - "8000:8000"
  selenium:
    image: selenium/standalone-
chrome
    container_name: selenium-chrome
    ports:
      - "4444:4444"
    depends_on:
       - app
```

Step-6:

Build and Run Your Docker Containers

In terminal use**docker-compose up –build**

This command will:

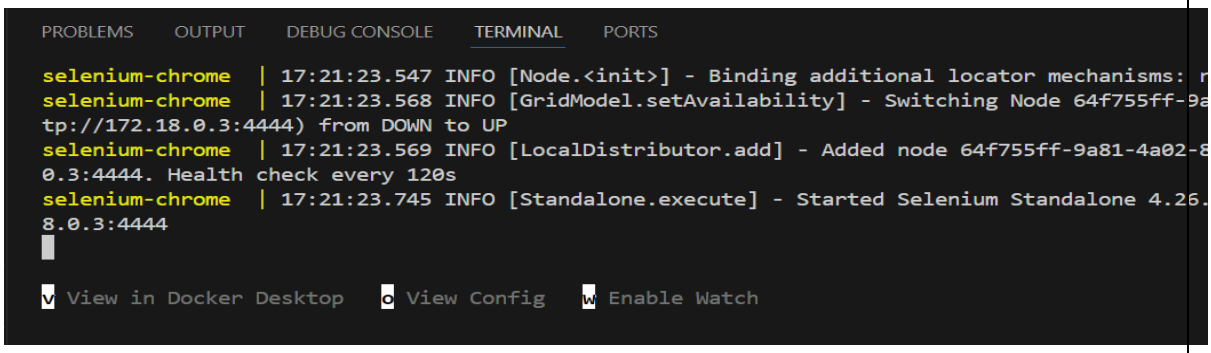
Build the Docker image for your HTML server.

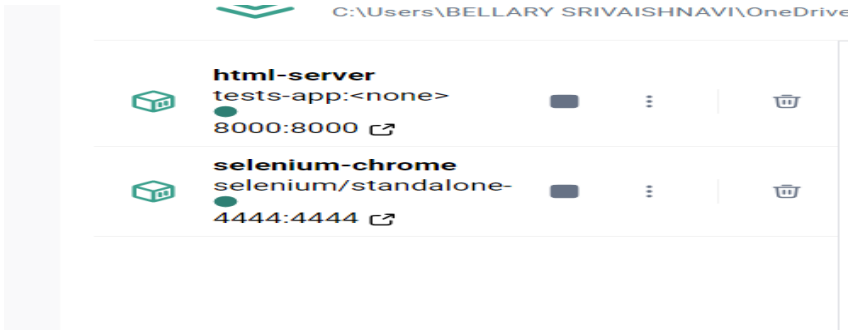
Pull the Selenium standalone Chrome image.

Start both services.

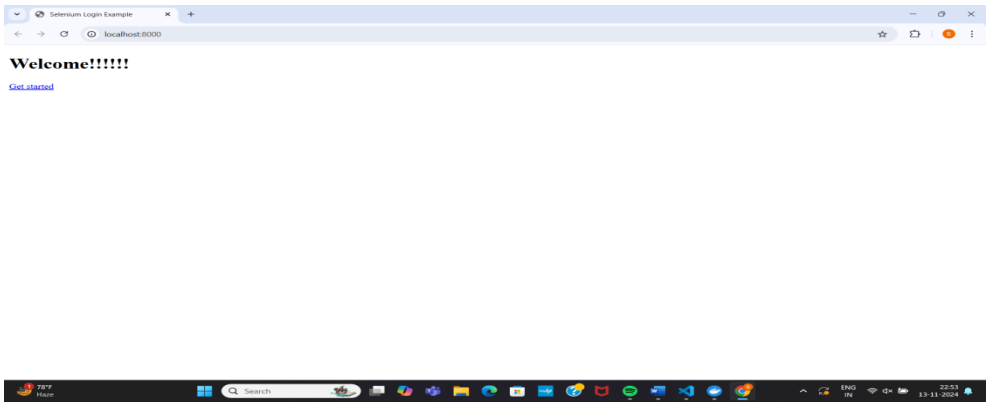
Press **v** to navigate to docker.

And Click on the links.

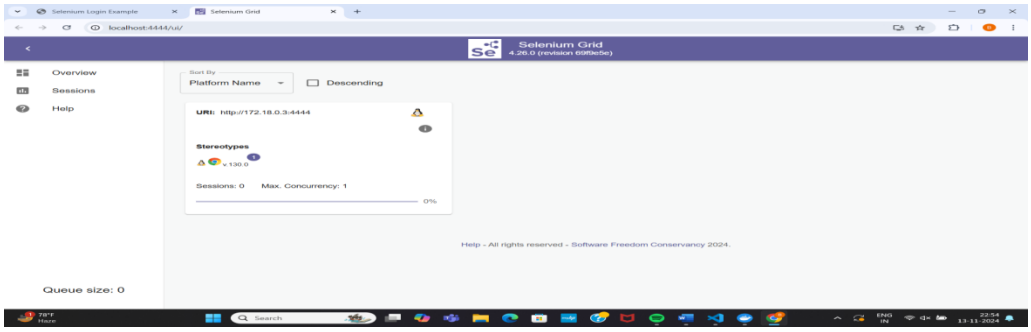




8000:8000



4444:4444



Step 7:

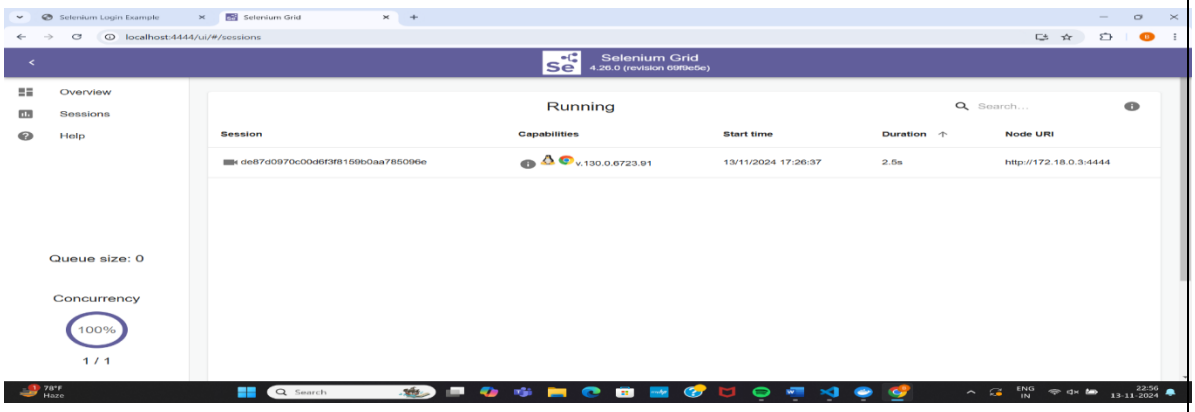
Run the Selenium Test Script

While your containers are running, open a new terminal and run:

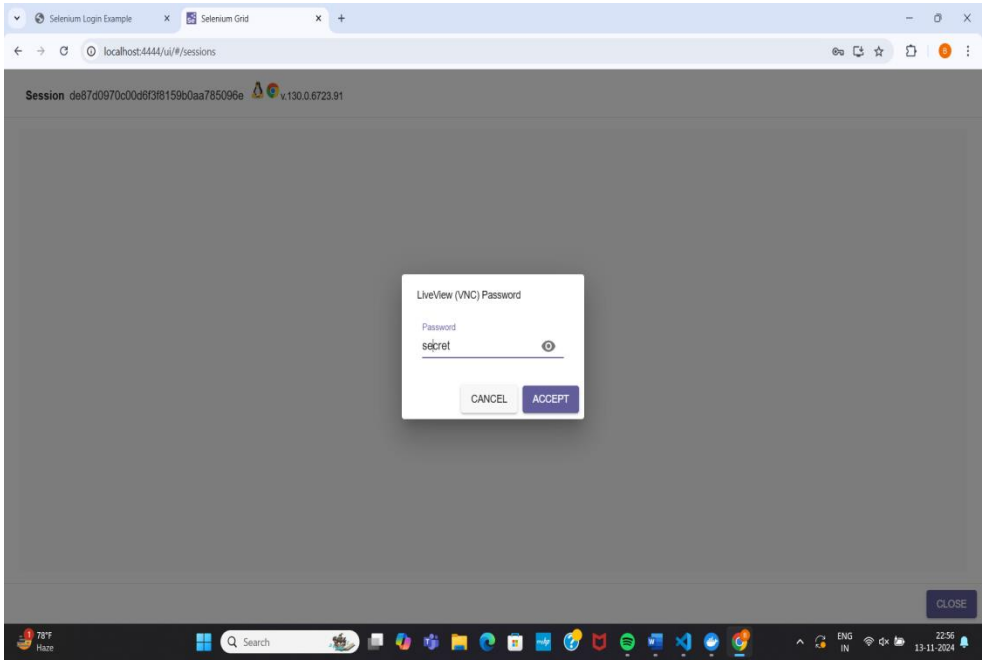
In terminal use, python SeleniumTest.py

Explanation :

Go to sessions, click on video icon.



password “secret” to view the video of automatic testing.



Step 8:

Stop the Docker Containers

Once you are done testing, stop the containers using:

docker-compose down

This command stops and removes the containers, networks, and any volumes associated with the services.

After running the test and clicking the video icon, the video will show how the test interacts with the web page (clicking buttons, entering passwords, etc.)