



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 -501401

(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:

Email:

Phone Number:

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

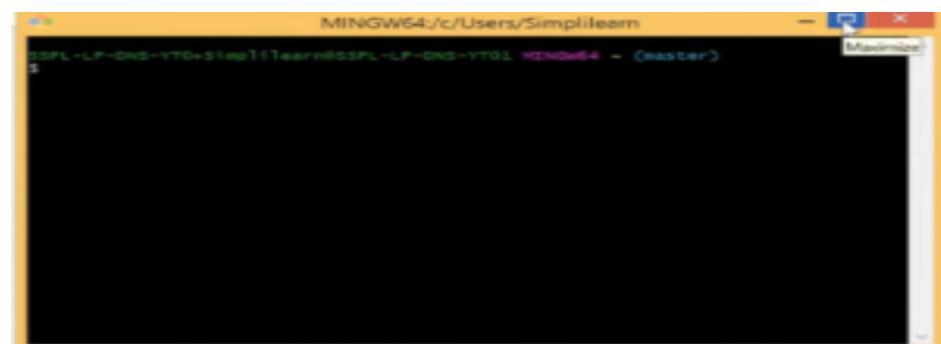
```
git remote add origin https://github.com/your-username/your-repo-name.git
```

```
git branch -M main
```

```
git push -u origin main
```

OUTPUT:

The screenshot shows the official Git website at git-scm.com. The main navigation bar includes links for About, Documentation, Downloads (selected), Community, and other resources. The 'Downloads' section is currently active, displaying options for GUI Clients and Logos. Below this, the 'Download for Windows' section is highlighted. It features a large button to 'Click here to download' the latest 64-bit version. It also lists other download options like Standalone Installer, 32-bit Git for Windows Setup, and 64-bit Git for Windows Setup. A sidebar on the left provides information about the Pro Git book.



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

A screenshot of a GitHub repository page for 'Airblader'. The repository has 5 commits and 5 pull requests. A context menu is open over the repository name, listing options such as 'New repository', 'Import repository', 'New ght', 'New organization', and 'Explore repository'. The GitHub interface includes a search bar, navigation links for Pull requests, Issues, Marketplace, and Explore, and a sidebar for repositories.

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.

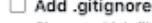
Initialize this repository with:

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

```

The screenshot shows a GitHub repository page. At the top, there is a message: "foobar had recent pushes less than a minute ago". To the right of this message is a green button labeled "Compare & pull request", which is circled in red. Below this, there are navigation links: "master" (with a dropdown arrow), "2 branches" (with a dropdown arrow), and "0 tags". To the right of these are buttons for "Go to file", "Add file", and "Code".

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

The screenshot shows the "Open a pull request" interface. At the top, there are dropdown menus for "base: master" and "compare: foobar", followed by a message: "✓ Able to merge. These branches can be automatically merged.". Below this is a search bar with the placeholder "Add you to repo". The main area contains a "Write" button, a "Preview" button, and a rich text editor toolbar with various icons. A text input field says "Leave a comment". At the bottom, there is a note: "Attach files by dragging & dropping, selecting or pasting them." and a green "Create pull request" button.

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.

-o 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.

-o 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](#)

master 1 branch 0 tags

Go to file Add file Code

dfryer1193 Merge pull request #1 from dfryer1193/foobar 7d52acd 21 seconds ago (3 commits)

me Add me to repo 6 minutes ago

you Add you to repo 5 minutes ago

master

Commits on Nov 6, 2020

Merge pull request #1 from dfryer1193/foobar
dfryer1193 committed 1 minute ago

Add you to repo
David Fryer committed 5 minutes ago

Add me to repo
David Fryer committed 7 minutes ago

```
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@megan.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 b345d9a25353037afdeaa9fcacf9f330effd157f1
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**.
-

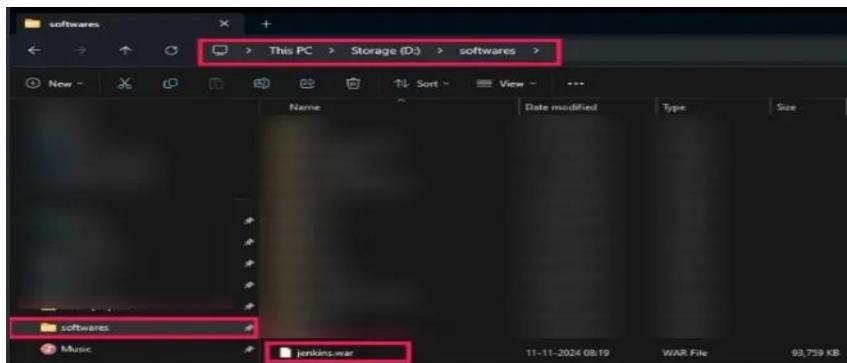
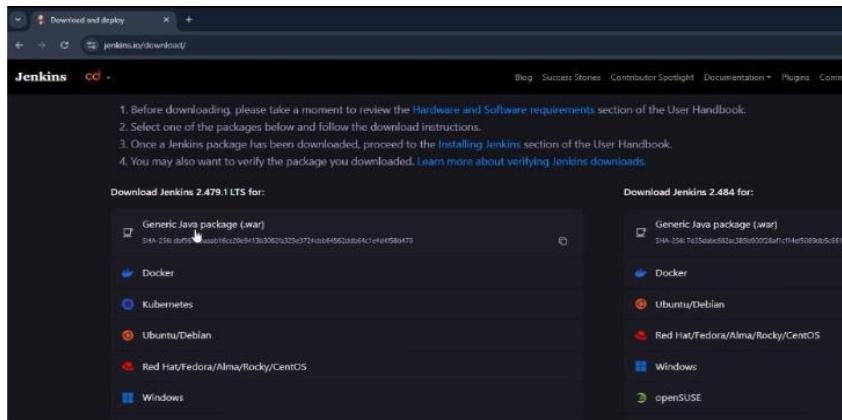
9. Verify Installation

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

10. Summary of Important Commands

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

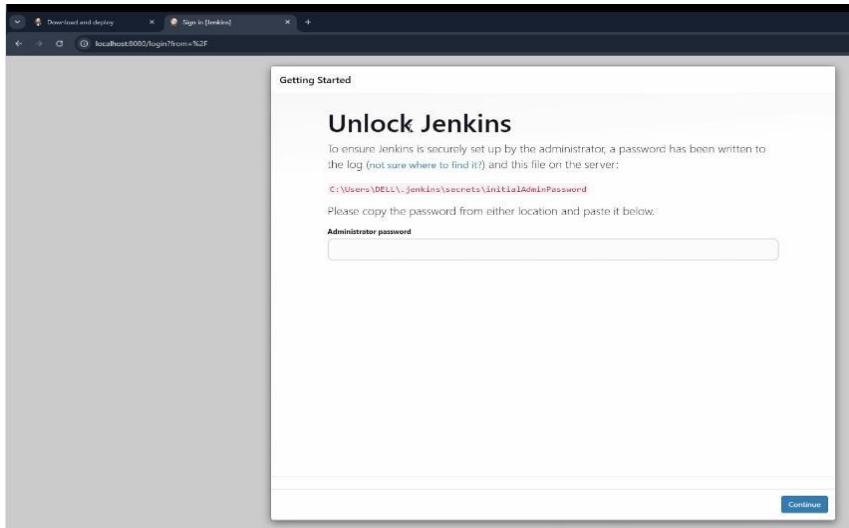
OUTPUT:



```
java -jar .\jenkins.war
Running from: C:\Users\DELL\.jenkins\war
webroot: C:\Users\DELL\.jenkins\war
2024-11-11 02:55:29.800+0000 [id=1] INFO  winstone.Logger#logInternal: Beginning extraction from war file
2024-11-11 02:55:33.103+0000 [id=1] WARNING o.e.j.e9.nested.ContextHandler#setContextPath: Empty contextPath
2024-11-11 02:55:33.238+0000 [id=1] INFO  org.eclipse.jetty.server.Server#doStart: jetty-12.0.13; built: 2024-09-03T03:04:05.240Z; git: 816018a420329e1cacd416799cdab8c8c6ea57cd; jvm 21.0.3+9-LTS
2024-11-11 02:55:33.240+0000 [id=1] INFO  o.e.j.u.StandardDescriptorProcessor#visitServlet: NO JSP Support for /, did not find
2024-11-11 02:55:33.240+0000 [id=1] INFO  o.e.j.u.StandardDescriptorProcessor#visitServlet: NO JSP Support for /, did not find
2024-11-11 02:55:34.809+0000 [id=1] INFO  o.e.j.s.DefaultSessionIdManager#doStart: Session workerName=node0
2024-11-11 02:55:36.032+0000 [id=1] INFO  hudson.WebAppMain#contextInitialized: Jenkins home directory: C:\Users\DELL\.jenkins found at: $user.home/.jenkins
2024-11-11 02:55:36.257+0000 [id=1] INFO  o.e.j.s.handler.ContextHandler#doStart: Started oejn9n.ContextHandler$CoreContextHandler@1451a26b[jenkins v2_479.1, /, b<file:///C:/Users/DELL/.jenkins/war/, a=AVAILABLE, h=oje9n.ContextHandler$CoreContextHandler$CoreToNest]
2024-11-11 02:55:36.257+0000 [id=1] INFO  o.e.j.s.handler.AbstractConnector#doStart: Started ServerConnector@bd111a[HTTP/1.1, (http://127.0.0.1:8080)]
2024-11-11 02:55:36.394+0000 [id=1] INFO  org.eclipse.jetty.server.Server#doStart: Started oejn9n.Server@40db2a24[STARTING][12.0.1]
3,stop=0] @782ms
2024-11-11 02:55:36.397+0000 [id=35] INFO  winstone.Logger#logInternal: Winstone Servlet Engine running; controlPort=disabled
2024-11-11 02:55:36.850+0000 [id=43] INFO  jenkins.InitReactorRunner$1#onAttained: Started initialization
2024-11-11 02:55:36.850+0000 [id=41] INFO  jenkins.InitReactorRunner$1#onAttained: Listed all plugins
2024-11-11 02:55:36.850+0000 [id=41] INFO  jenkins.InitReactorRunner$1#onAttained: Started all extensions
2024-11-11 02:55:38.088+0000 [id=46] INFO  jenkins.InitReactorRunner$1#onAttained: Started all plugins
2024-11-11 02:55:38.071+0000 [id=46] INFO  jenkins.InitReactorRunner$1#onAttained: Augmented all extensions
```

```
*****
Jenkins initial setup is required. An admin user has been created and a password generated.
Please use the following password to proceed to installation:
10d3965df33e416bb773e50d8db125dc
This may also be found at: C:\Users\DELL\.jenkins\secrets\initialAdminPassword
*****
2024-11-11 02:55:47.773+0000 [id=42] INFO  jenkins.InitReactorRunner$1#onAttained: Completed initialization
2024-11-11 02:55:47.873+0000 [id=34] INFO  hudson.lifecycle.Lifecycle#onReady: Jenkins is fully up and running
```

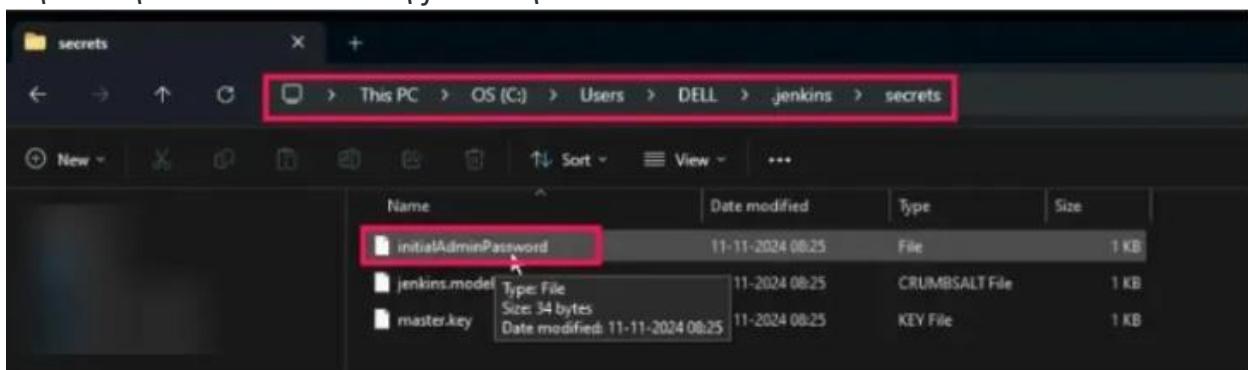
<http://localhost:8080>



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

Unlock Jenkins:

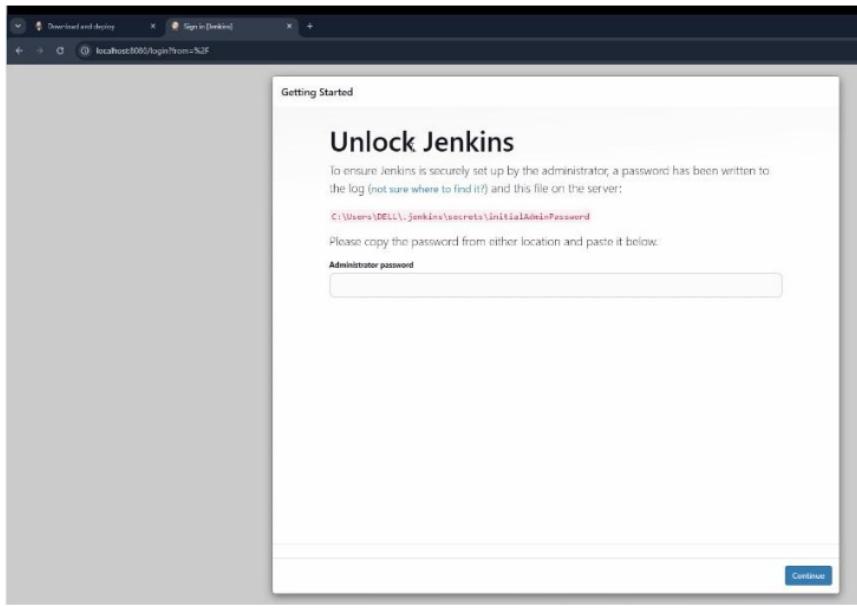
During the setup, Jenkins will generate a one-time admin password. Retrieve the password from the console logs or the initialAdminPassword file located in:
C:\Users<YourUsername>\.jenkins\secrets



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.

A screenshot of the Jenkins 'Customize Jenkins' setup page. The title is 'Customize Jenkins' and it says 'Plugins extend Jenkins with additional features to support many different needs.' Two options are shown: 'Install suggested plugins' (selected) and 'Select plugins to install'.

Getting Started

Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

Install suggested plugins

Install plugins the Jenkins community finds most useful.

Select plugins to install

Select and install plugins most suitable for your needs.

install plugins

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

Getting Started

Getting Started

<input checked="" type="checkbox"/> Folders	<input checked="" type="checkbox"/> OWASP Markup Formatter	<input checked="" type="checkbox"/> Build Timeout	<input type="checkbox"/> Credentials Binding	Ionicons API
<input type="checkbox"/> Timestamper	<input type="checkbox"/> Workspace Cleanup	<input type="checkbox"/> Ant	<input type="checkbox"/> Gradle	Folders
<input type="checkbox"/> Pipeline	<input type="checkbox"/> GitHub Branch Source	<input type="checkbox"/> Pipeline: GitHub Groovy Libraries	<input type="checkbox"/> Pipeline Graph View	OWASP Markup Formatter
<input type="checkbox"/> Git	<input type="checkbox"/> SSH Build Agents	<input type="checkbox"/> Matrix Authorization Strategy	<input type="checkbox"/> PAM Authentication	** ASM API
<input type="checkbox"/> LDAP	<input type="checkbox"/> Email Extension	<input type="checkbox"/> Mailer	<input type="checkbox"/> Dark Theme	** JSON Path API

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

Getting Started

Create First Admin User

Username

Password

Confirm password

Full name

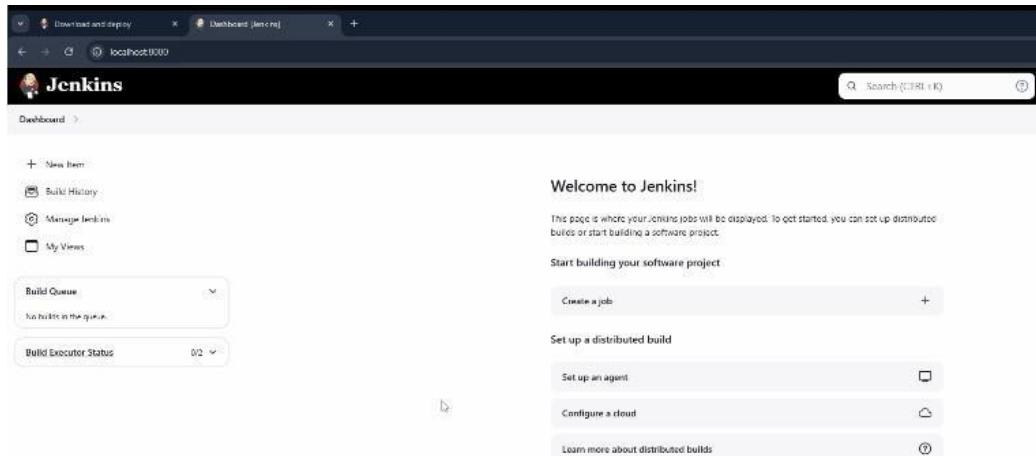
E-mail address

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)

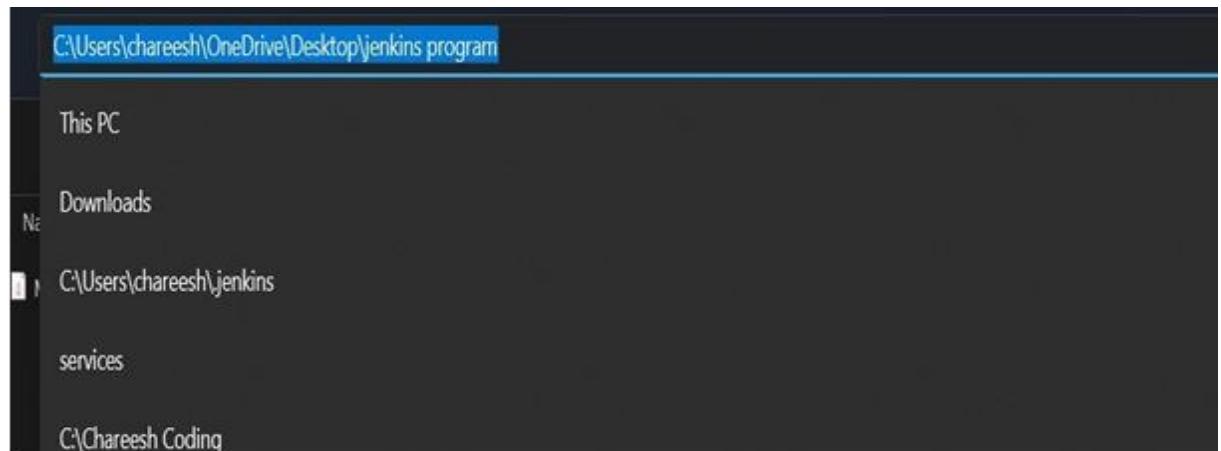
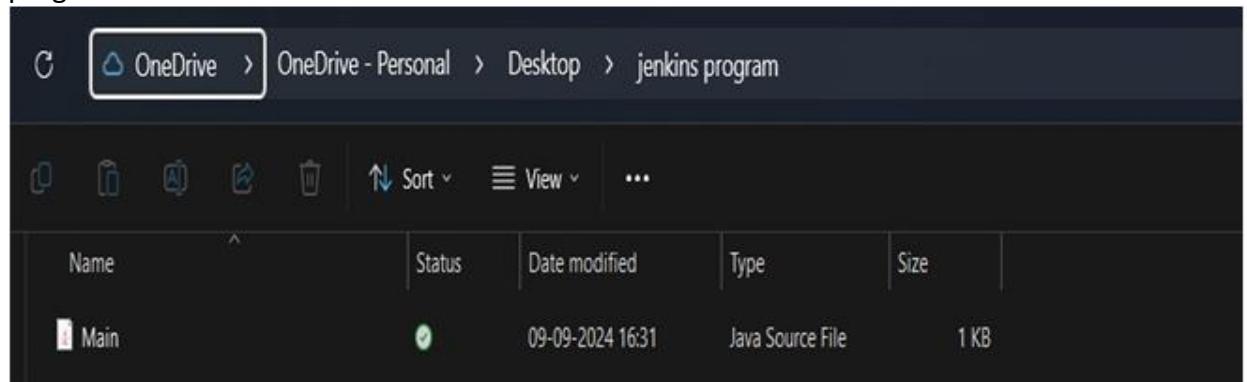
- Add shell scripts or deployment tasks as build steps to automate deployment after build.
-

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 it and save it in the new folder with .java extension. Now copy the new folder path 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

New Item

Enter an item name

first.java.program

Select an item type



Freestyle project

Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.



Maven project

Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.



Pipeline

Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

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

Configure

General

Source Code Management

Build Triggers

Build Environment

Build Steps

Post-build Actions

Execute concurrent builds if necessary ?

Advanced ^ Edited

Quiet period ?

Retry Count ?

Block build when upstream project is building ?

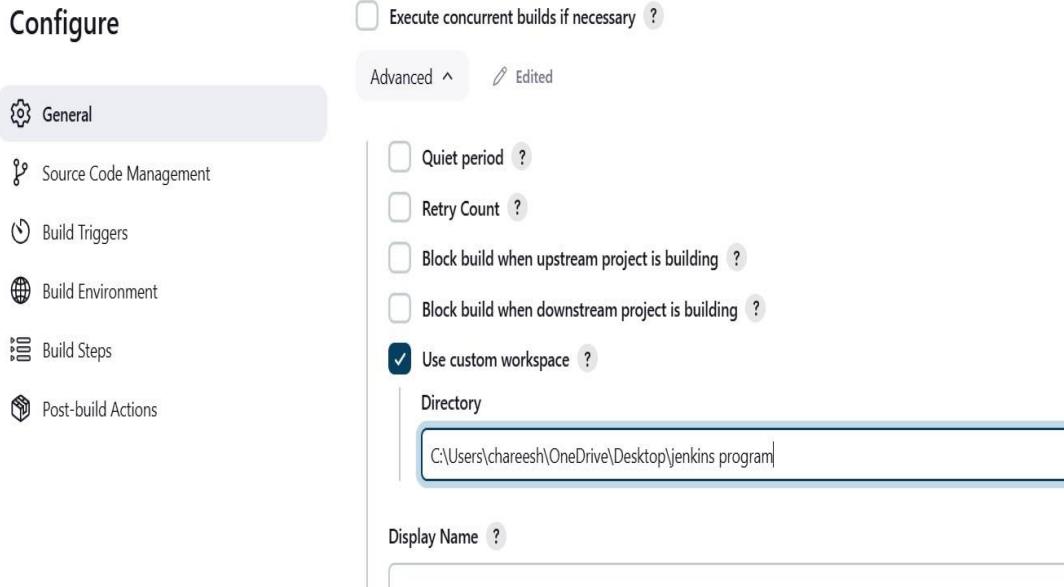
Block build when downstream project is building ?

Use custom workspace ?

Directory

C:\Users\chareesh\OneDrive\Desktop\jenkins program

Display Name ?



- 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)

Build Steps

≡ Execute Windows batch command ?

Command

See [the list of available environment variables](#)

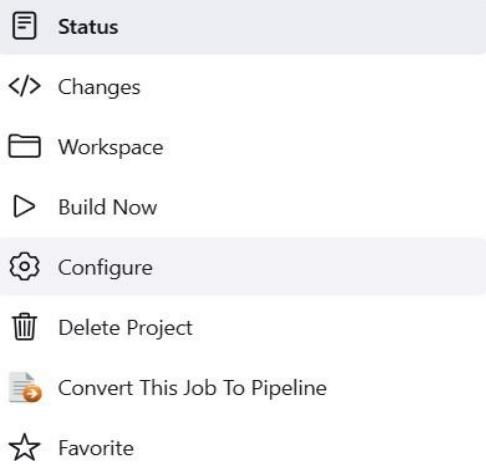
```
javac Main.java
java Main
```

Advanced ^



->Now click on apply and then save

->Now click on Build Now



first_java_program

java program

Permalinks

- Last build (#2), 12 sec ago
- Last stable build (#2), 12 sec ago
- Last successful build (#2), 12 sec ago
- Last failed build (#1), 1 min 34 sec ago
- Last unsuccessful build (#1), 1 min 34 sec ago
- Last completed build (#2), 12 sec ago

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

A screenshot of the Jenkins build history page for job 'first_java_program'. It shows two builds: #2 (successful) and #1 (unsuccessful). The successful build (#2) is dated Sep 9, 2024, at 4:45 PM. The unsuccessful build (#1) is dated Sep 9, 2024, at 4:44 PM. Below the builds are links for Atom feed for all and Atom feed for failures.

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

A screenshot of the Jenkins console output for build #2. The output shows the command to build the Java program and the resulting 'Hello world.' message.

```
Started by user admin
Running as SYSTEM
Building in workspace C:\Users\chareesh\OneDrive\Desktop\jenkins program
[jenkins program] $ cmd /c call C:\Users\chareesh\AppData\Local\Temp\jenkin
C:\Users\chareesh\OneDrive\Desktop\jenkins program>javac Main.java
C:\Users\chareesh\OneDrive\Desktop\jenkins program>java Main
Hello world.

C:\Users\chareesh\OneDrive\Desktop\jenkins program>exit 0
Finished: SUCCESS
```

Create a new item (Pipeline)

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

Dashboard > All > New Item

New Item

Enter an item name
My Basic Pipeline

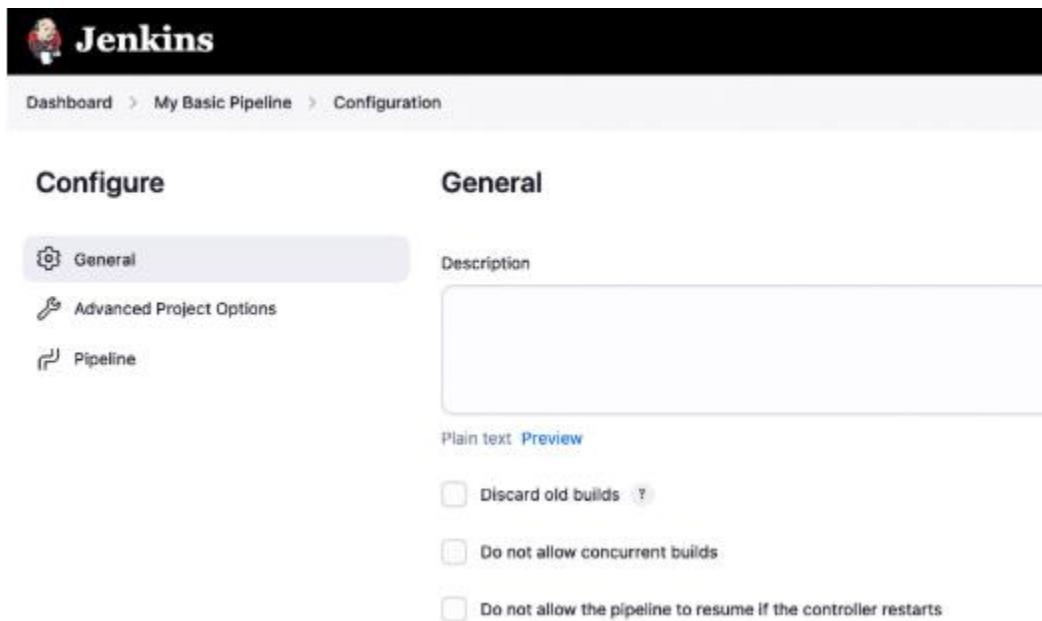
Select an item type

- Freestyle project**
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.
- Pipeline**
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
- Multi-configuration project**
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.
- Folder**
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.
- Multibranch Pipeline**
Creates a set of Pipeline projects according to detected branches in one SCM repository.
- Organization Folder**
Creates a set of multibranch project subfolders by scanning for repositories.

If you want to create a new item from other existing, you can use this option:

OK

Set the general pipeline configuration options



The screenshot shows the Jenkins Pipeline Configuration page for a project named "My Basic Pipeline". The top navigation bar includes links for "Dashboard", "My Basic Pipeline", and "Configuration". On the left, a sidebar titled "Configure" lists "General", "Advanced Project Options", and "Pipeline". The main "General" tab is selected, showing a "Description" field which is currently empty. Below the field are two links: "Plain text" and "Preview". Under the "Description" field, there are three checkboxes:

- Discard old builds ?
- Do not allow concurrent builds
- Do not allow the pipeline to resume if the controller restarts

Define the pipeline

1. Pipeline Script
2. Pipeline Script from SCM

Dashboard > My Basic Pipeline > Configuration

Configure Advanced Project Options

General Advanced Project Options Pipeline

Display Name ?

Pipeline

Definition

Pipeline script

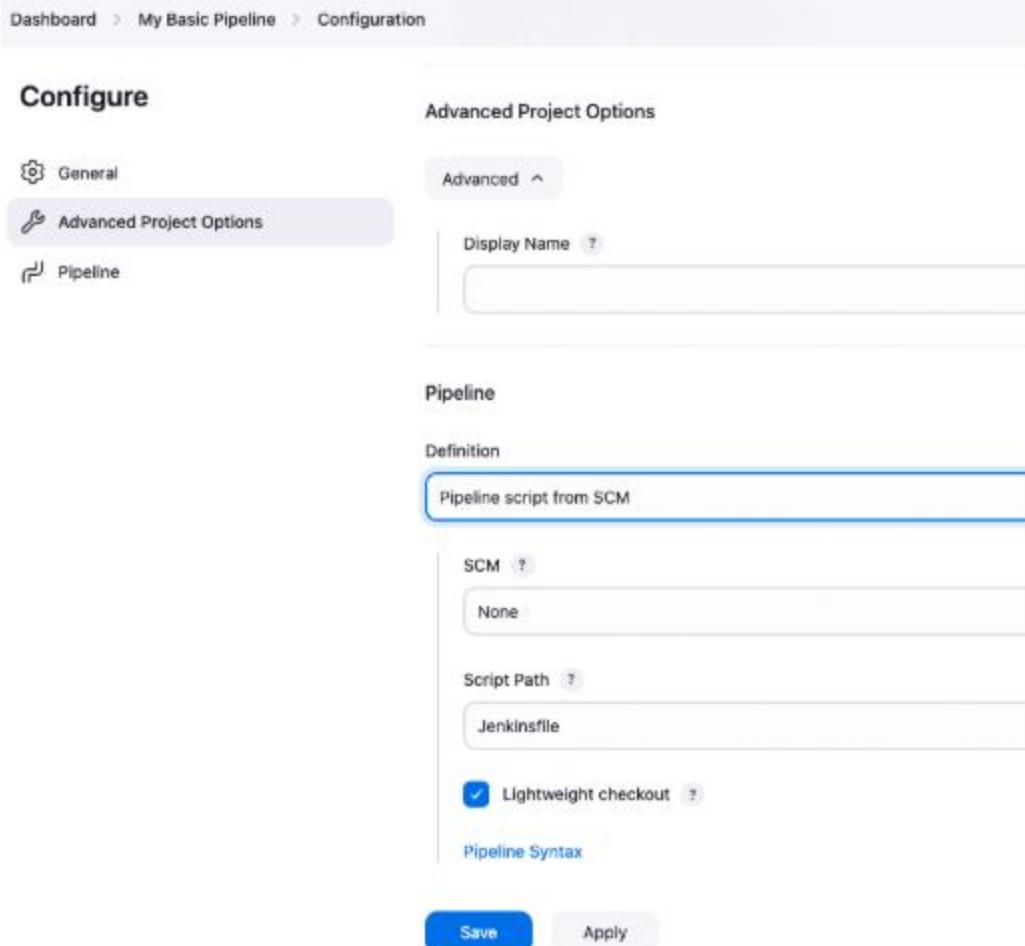
Script ?
1

Use Groovy Sandbox ?

Pipeline Syntax

Save Apply

The screenshot shows a configuration page for a pipeline named 'My Basic Pipeline'. The 'Advanced Project Options' tab is selected. In the 'Pipeline' section, the 'Definition' is set to 'Pipeline script'. A script editor shows the number '1'. A checkbox for 'Use Groovy Sandbox' is checked. At the bottom, there are 'Save' and 'Apply' buttons.



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 triggered.

Configure**Pipeline**

Definition

 General Advanced Project Options Pipeline

Pipeline script

```
Script ?  
1 pipeline {  
2     agent any  
3  
4     stages {  
5         stage('Build') {  
6             steps {  
7                 echo 'Building the project...'  
8             }  
9         }  
10        stage('Test') {  
11            steps {  
12                echo 'Running tests...'  
13            }  
14        }  
15        stage('Deploy') {  
16            steps {  
17                echo 'Deploying the application...'  
18            }  
19        }  
20    }  
21}  
22}  
23  
24 post {  
25     success {  
26         echo 'Pipeline executed successfully!'  
27     }  
28     failure {  
29         echo 'Pipeline failed. Please check the logs for details.'  
30     }  
31 }  
32 }  
33 }
```

 Use Groovy Sandbox ?**Save****Apply****Save the pipeline****Run the pipeline**

Jenkins

Dashboard > My Basic Pipeline >

Status **My Basic Pipeline** Build Now

Changes Configure Delete Pipeline Full Stage View Rename Pipeline Syntax

Builds Filter

Today

- Build #2 15:34
- Build #1 15:34

Stage View

Build	Test	Deploy	Declarative: Post Actions
Average stage times: (Average full run time: ~1s) Jan 28 15:34	353ms	102ms	94ms
	105ms	100ms	101ms
	602ms	101ms	87ms
	failed	failed	failed
	122ms	142ms	

Permalinks

- Last build (#2), 1 min 37 sec ago
- Last stable build (#2), 1 min 37 sec ago
- Last successful build (#2), 1 min 37 sec ago
- Last failed build (#1), 2 min 17 sec ago
- Last unsuccessful build (#1), 2 min 17 sec ago
- Last completed build (#2), 1 min 37 sec ago

Access the logs

Jenkins

Dashboard > My Basic Pipeline > #2

Status **Console Output** Build Now

Changes Console Output View as plain text Edit Build Information Delete build '#2' Restart from Stage Replay

Pipeline Steps Workspaces Previous Build

Started by user Admin

```
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /Users/ltd/.jenkins/workspace/My Basic Pipeline
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Build)
[Pipeline] echo
Building the project...
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Test)
[Pipeline] echo
Running tests...
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Deploy)
[Pipeline] echo
Deploying the application...
[Pipeline] }
```

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.

Docker Desktop

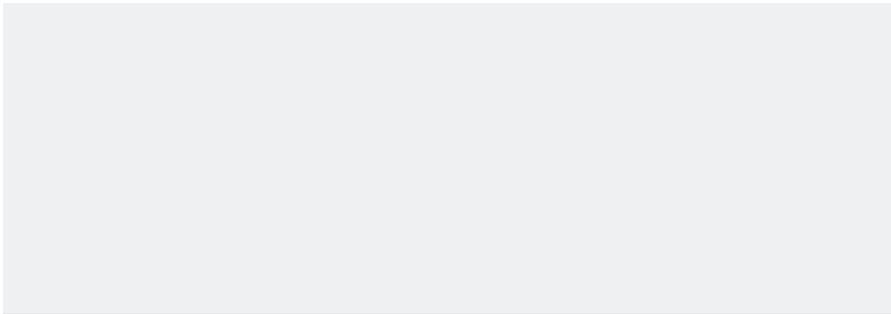
The fastest way to containerize applications on your desktop

[Download for Windows](#)

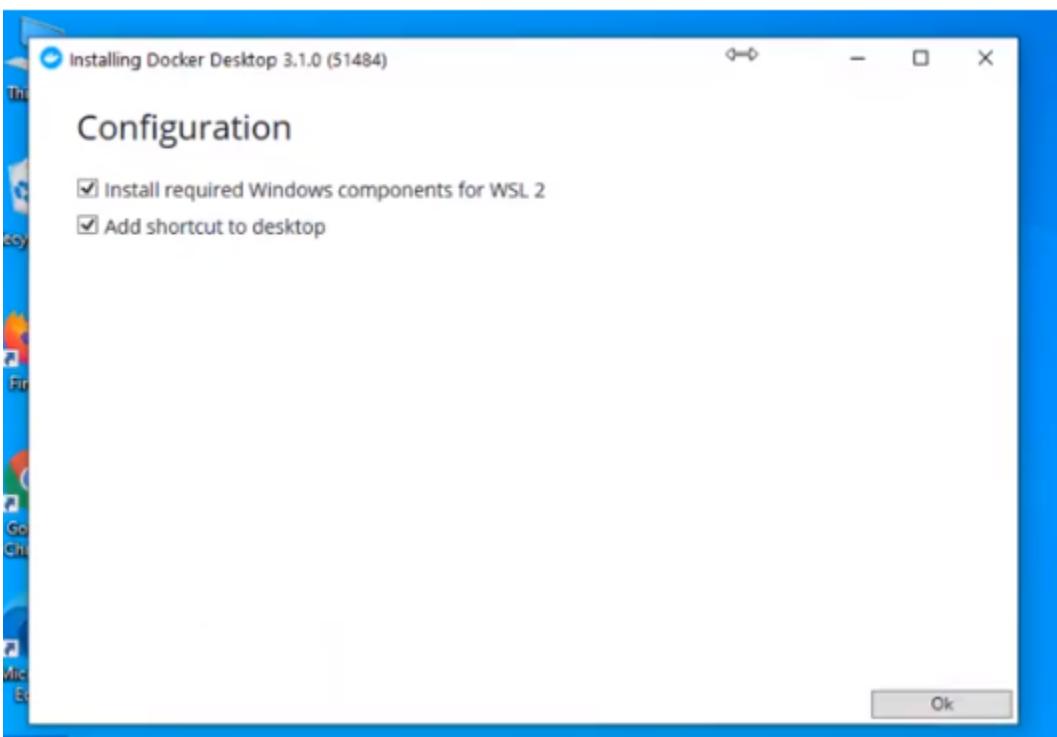
Also available for [Mac](#) and [Linux](#)

By downloading this, you agree to the terms of the [Docker Software End User License Agreement](#) and the [Docker Data Processing Agreement \(DPA\)](#).

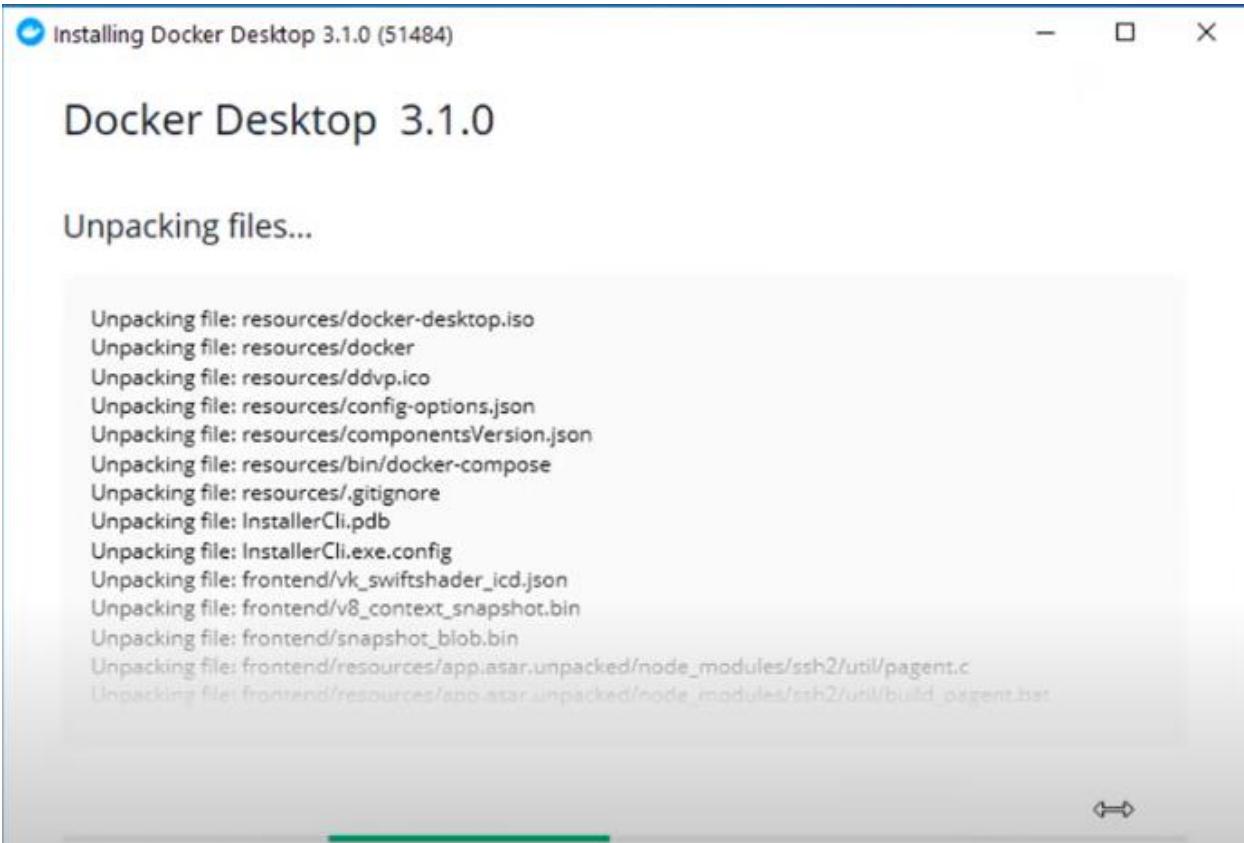
Start the Download



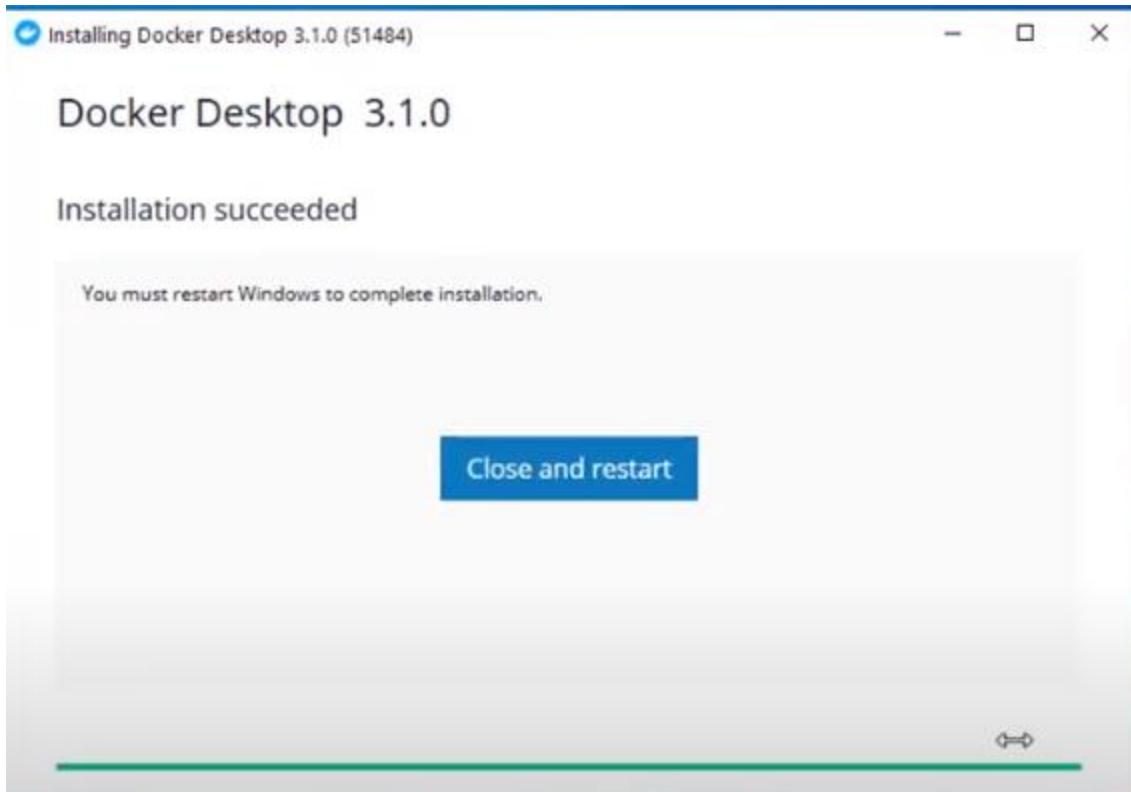
After installation, open Docker Desktop.



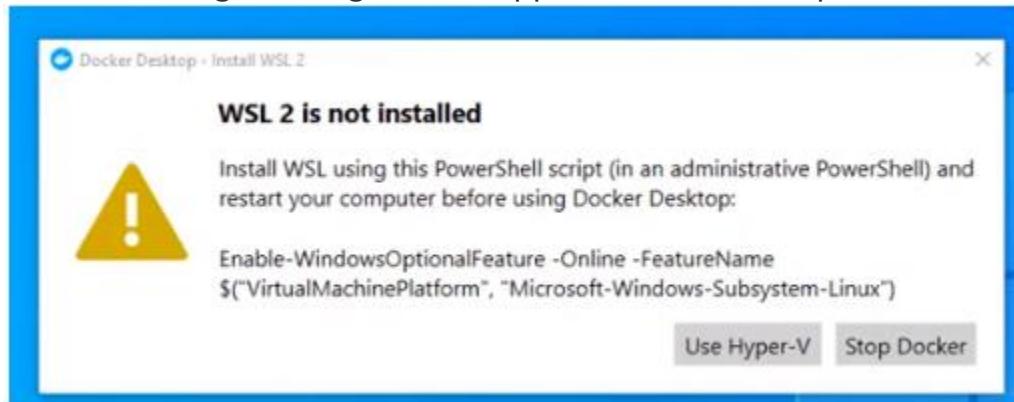
After clicking "OK," the installation will start.



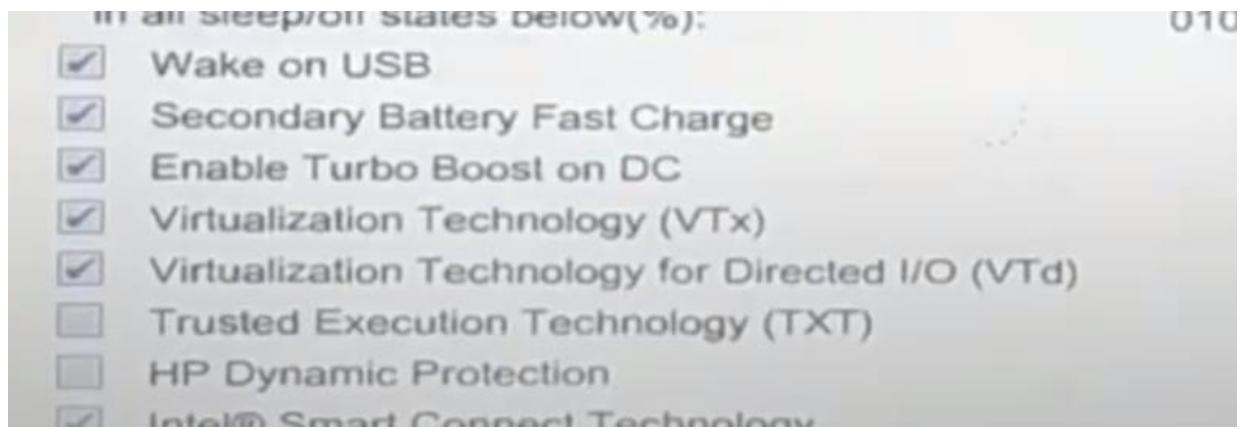
After installation completes, it will show a confirmation screen.



- 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.

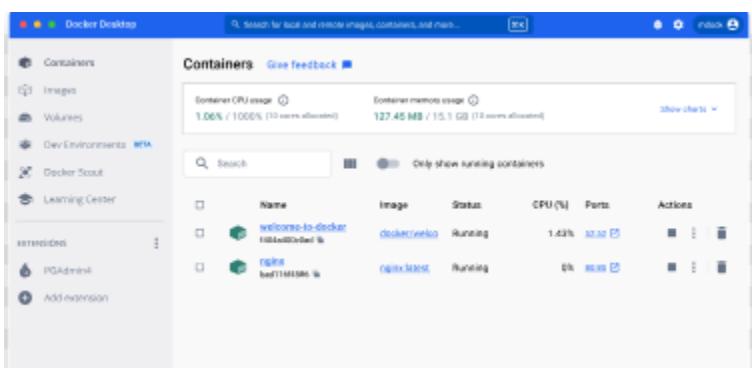
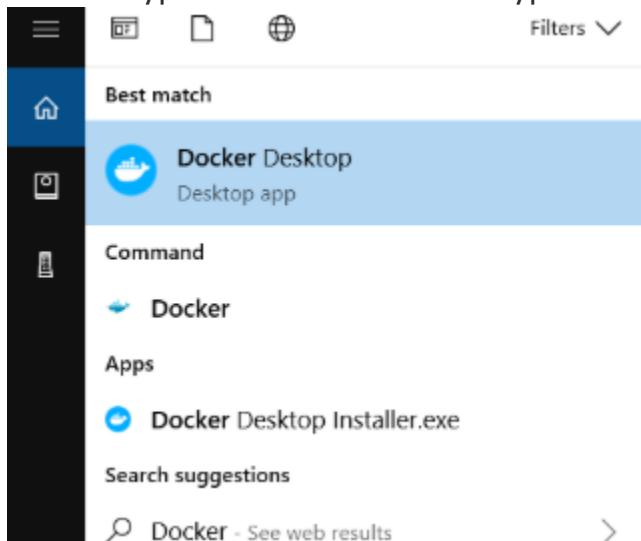


- Restart your PC and enter the BIOS setup:
 - Navigate to Settings > Update and Security > Recovery > Advanced Setup > Device Configuration.
 - Ensure the "Enable Turbo Boost on DC" option is marked. Save and exit.



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 # o000o000o000o Redis is starting o000o000o000o
1:C 27 Sep 2022 04:55:32.494 # Redis version=7.0.5, bits=64, commit=00000000, modified=0
1:C 27 Sep 2022 04:55:32.494 # Warning: no config file specified, using the default conf
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 m
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

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 present in 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 crashed or 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, you can do it with the help of this command.

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

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

```
b7e3fefc202d
```

Docker Images: 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.

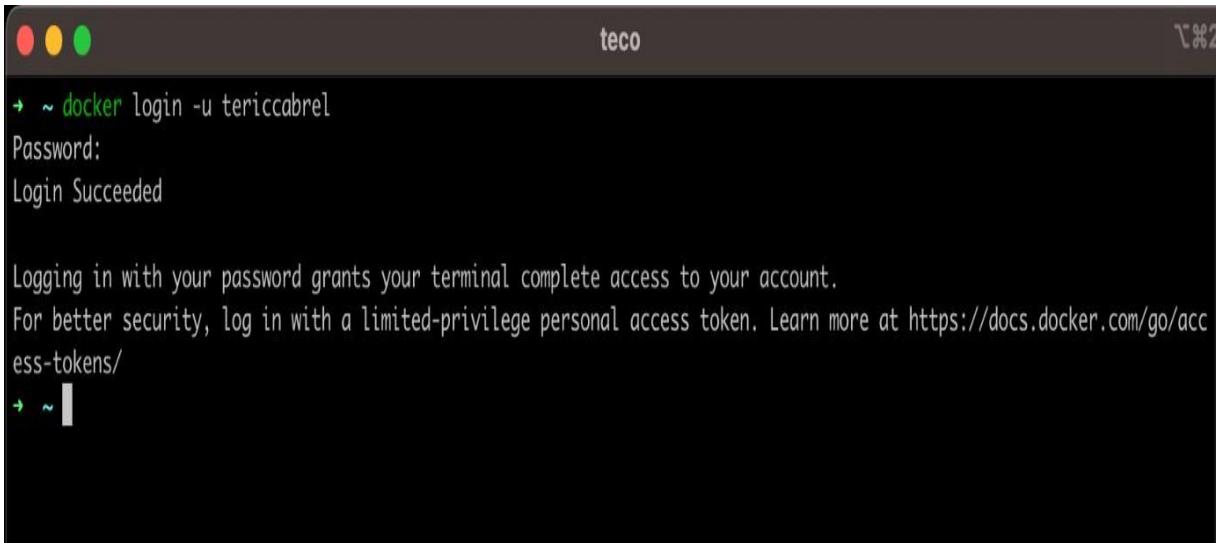
-e flag: sets the environment variables

This command allows us to run new commands in a running container.

```
C:\root@4396324406f8: /  
Microsoft Windows [Version 10.0.19043.1889]  
(c) Microsoft Corporation. All rights reserved.  
  
C:\Users\mojha>docker exec -d ubuntu_bash touch /tmp/execWorks  
  
C:\Users\mojha>docker exec -it ubuntu_bash bash  
root@4396324406f8:/#
```

Docker Login: docker login

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

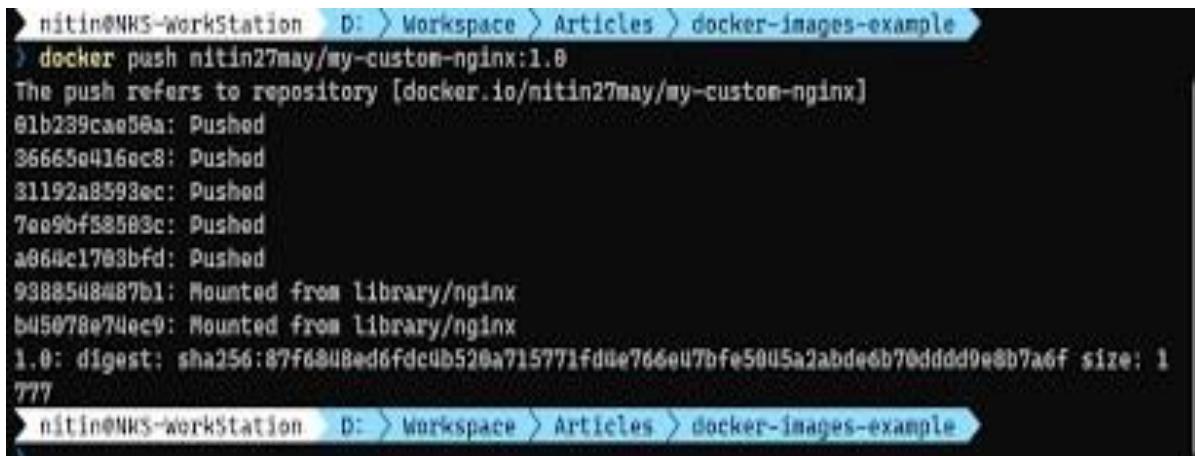


```
tericabrel
Password:
Login Succeeded

Logging in with your password grants your terminal complete access to your account.
For better security, log in with a limited-privilege personal access token. Learn more at https://docs.docker.com/go/access-tokens/
→ ~ |
```

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 registry which is DockerHub for that you need to push your image by using the following command.

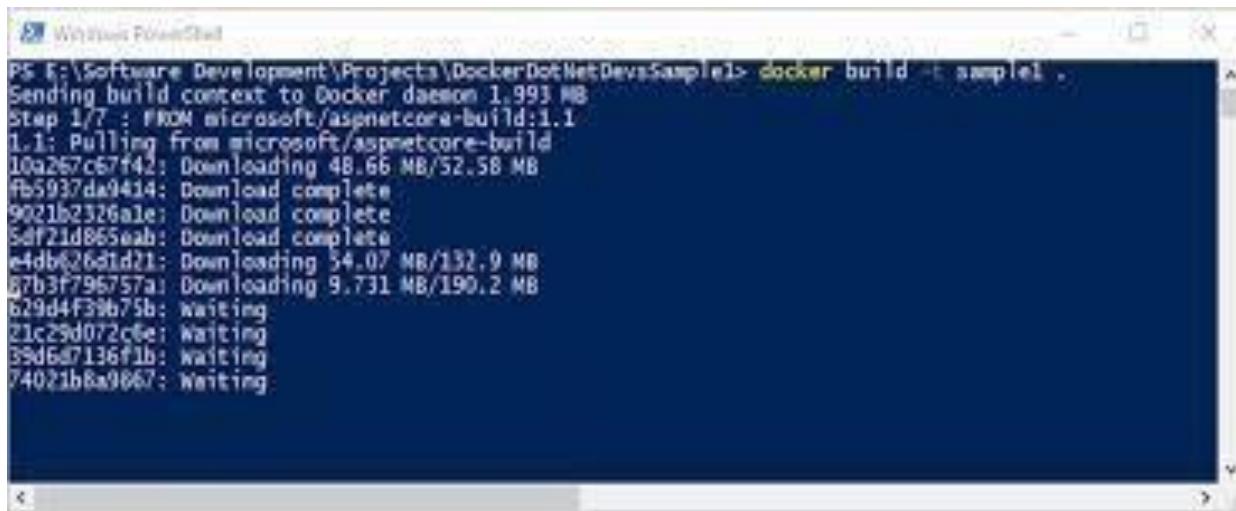


```
nitin@NHS-WorkStation D: > Workspace > Articles > docker-images-example
> docker push nitin27may/my-custom-nginx:1.0
The push refers to repository [docker.io/nitin27may/my-custom-nginx]
81b239cae08a: Pushed
36665e416ec8: Pushed
31192a8593ec: Pushed
7ee9bf58583c: Pushed
a064c1703bfd: Pushed
9388548487b1: Mounted from library/nginx
b45078e74ec9: Mounted from library/nginx
1.0: digest: sha256:87f6848ed6fdc4b520a715771fdde766e47bfe5005a2abde6b70ddbd9e8b7a6f size: 1
777
> nitin@NHS-WorkStation D: > Workspace > Articles > docker-images-example
```

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 the current directory.

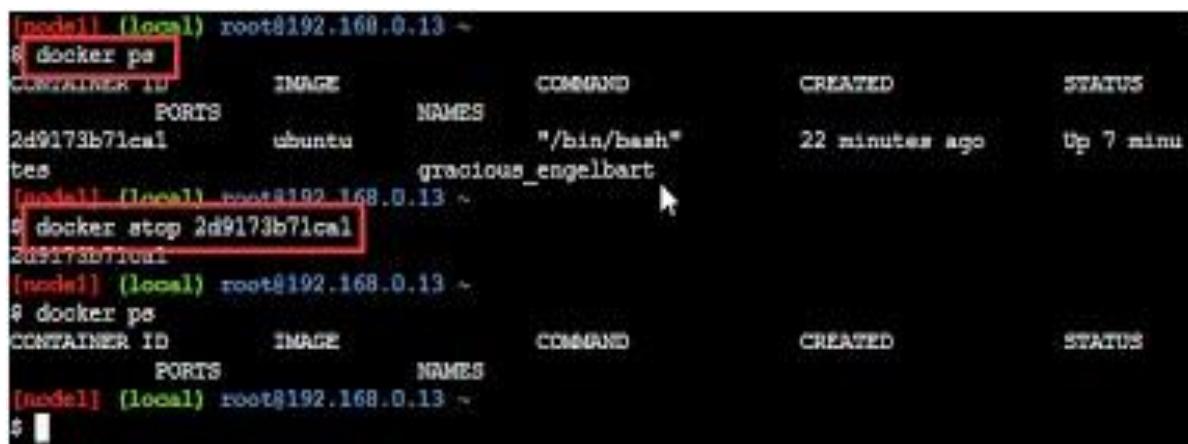
The docker build command is used to build the docker images with the help of Dockerfile.



```
PS E:\Software Development\Projects\DockerDotNetDevSample1> docker build -t sample1 .
Sending build context to Docker daemon 1.993 MB
Step 1/7 : FROM microsoft/aspnetcore-build:1.1
1.1: Pulling from microsoft/aspnetcore-build
10a267c67f42: Downloading 48.66 MB/52.58 MB
fb5937da9414: Download complete
9021b2326a1e: Download complete
5df21d865eab: Download complete
e4db626d1d21: Downloading 54.07 MB/132.9 MB
57b3f796757a: Downloading 9.731 MB/190.2 MB
629d4f39b75b: waiting
21c29d072c6e: Waiting
39d6d7136f1b: Waiting
74021b8a9867: Waiting
```

Docker Stop: docker stop <container_name or id>

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



```
[node1] (local) root@192.168.0.13 ~
$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS
2d9173b71cal      ubuntu              "/bin/bash"        22 minutes ago   Up 7 minutes
tea
[node1] (local) root@192.168.0.13 ~
$ docker stop 2d9173b71cal
2d9173b71cal
[node1] (local) root@192.168.0.13 ~
$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS
$
```

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

Instead of stopping a single container. You can stop multiple containers 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 fail to start. You can restart the containers.

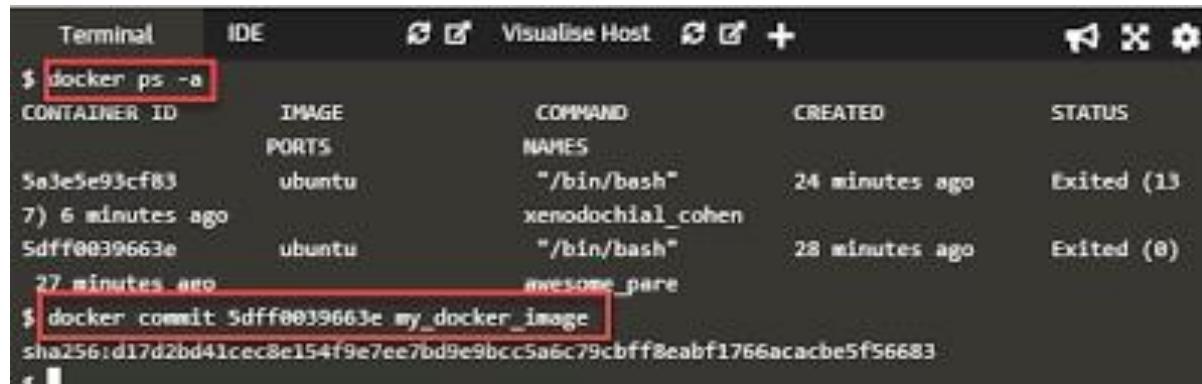
```
# docker restart $(docker ps -q)
745c51ea5908
aff44fb7b589
351f84d026f1
309c38398750
fa5457424722
9a0c10316d4c
87dbd6caab32
594d408e64ce
60f001a97db6
```

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.



The screenshot shows a terminal window with the following content:

```
Terminal IDE Visualise Host + 
$ docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS
5a3e5e93cf83        ubuntu              "/bin/bash"        24 minutes ago   Exited (13
7) 6 minutes ago
5dff0039663e        ubuntu              "/bin/bash"        28 minutes ago   Exited (0)
27 minutes ago
$ docker commit 5dff0039663e my_docker_image
sha256:d17d2bd41cec8e154f9e7ee7bd9e9bcc5a6c79cbff8eabf1766acacbe5f56683
```

WEEK -7

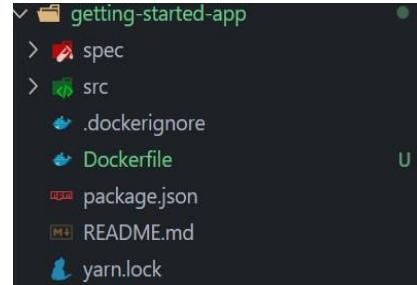
Develop a simple containerized application using Docker

- Clone the [getting-started-app](https://github.com/docker/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 following code

```
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-app directory. Replace /path/to/getting-started-app with the path to your getting-started-app directory.

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

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

Build the Image

Run the following commands in the terminal:

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

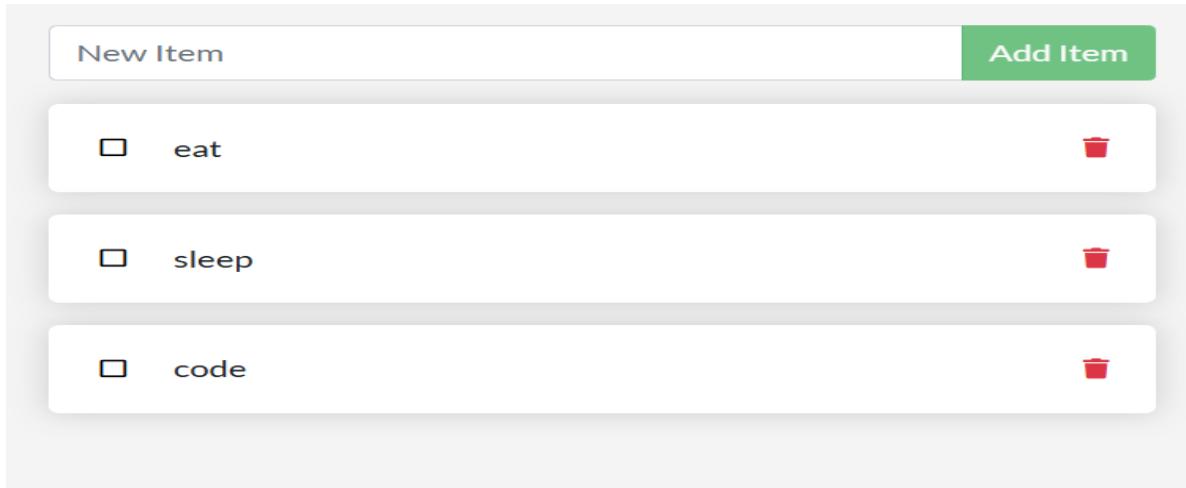
```
PS C:\Users\Virendra\OneDrive\Desktop\DevOps Lab> cd ..\getting-started-app\  
PS C:\Users\Virendra\OneDrive\Desktop\DevOps Lab\getting-started-app> docker build -t getting-started .  
[+] Building 63.3s (10/10) FINISHED  
=> [internal] load build definition from Dockerfile  
=> => transferring dockerfile: 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 can find the **getting-started** image. Also you can check the image in the docker desktop images section.

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

- > 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.



Products Developers Pricing Blog About us Partners

Docker Desktop

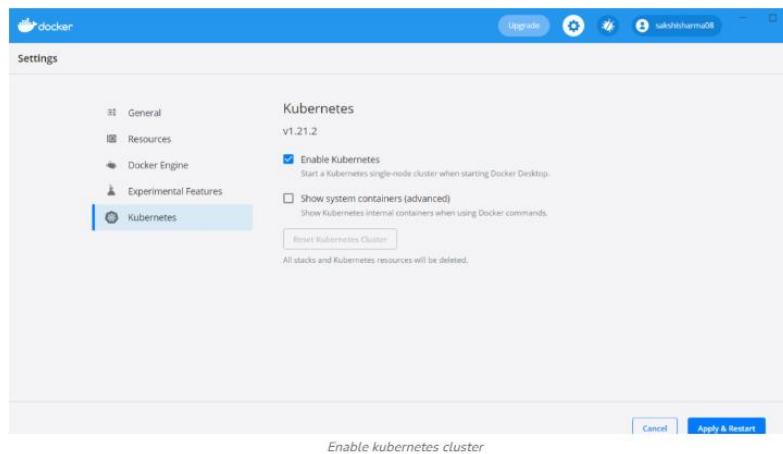
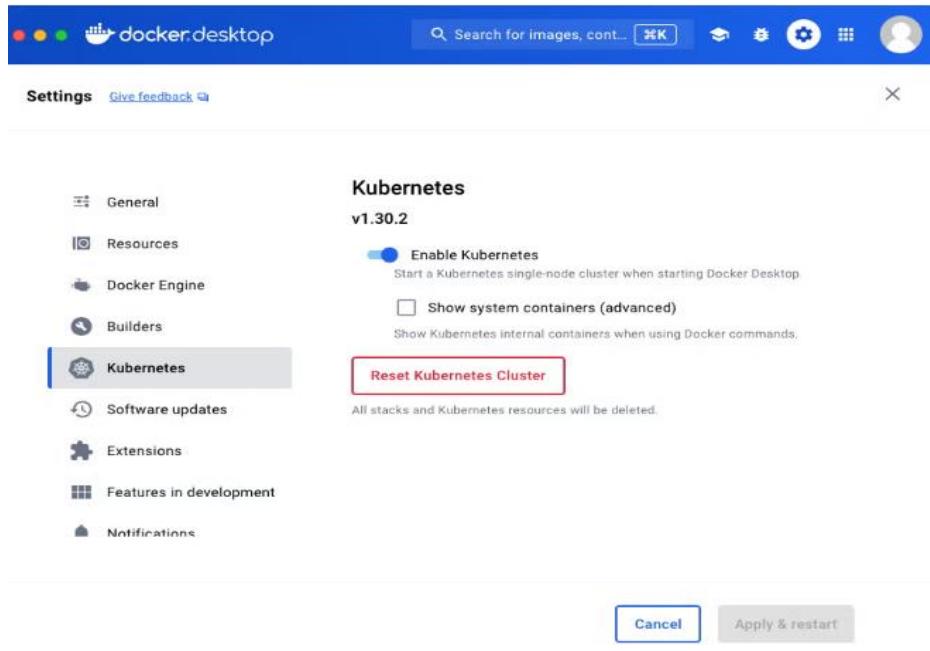
The fastest way to containerize applications on your desktop

[Download for Windows](#)

Also available for [Mac](#) and [Linux](#)

By downloading this, you agree to the terms of the Docker Software End User License Agreement and the Docker Data Processing Agreement (DPA).

Enable Kubernetes



To download Kubernetes CLI

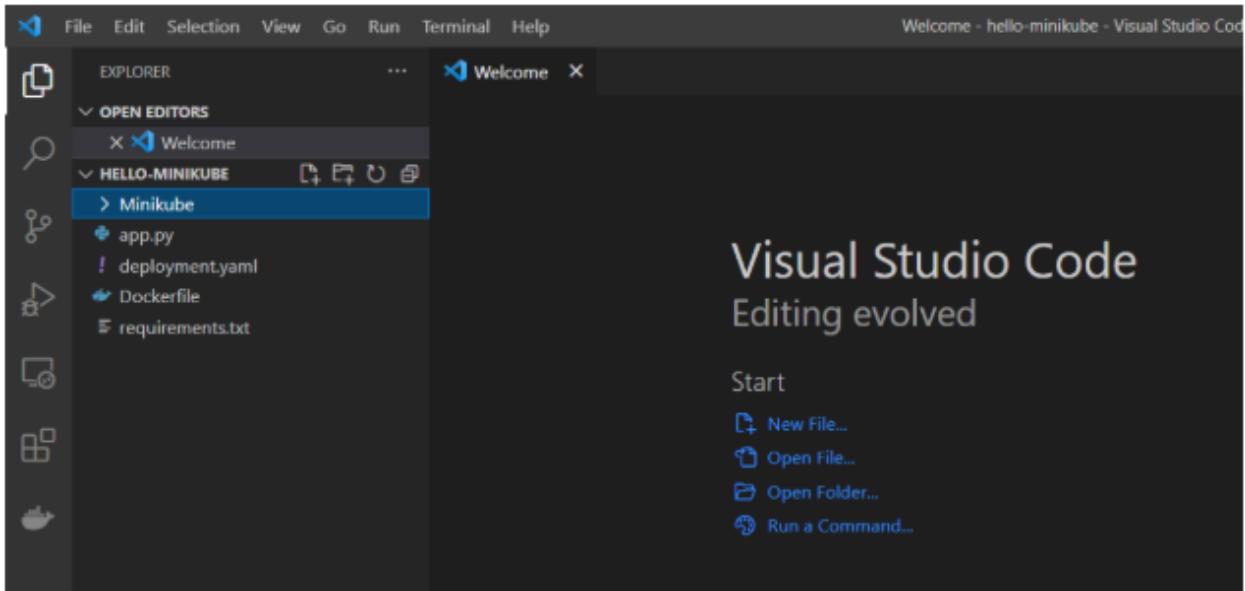
The screenshot shows the Kubernetes Documentation website. The top navigation bar includes links for Documentation, Kubernetes Blog, Training, Partners, Community, Case Studies, Versions, and English. The main content area is titled "Install Tools" and contains sections for "kubectl", "kind", and "minikube". On the left, there's a sidebar with navigation links for Home, Getting started, Concepts, Tasks, and various sub-sections under "Install Tools" such as "Install and Set Up kubectl on Linux", "Install and Set Up kubectl on macOS", and "Install and Set Up kubectl on Windows". A search bar is located at the top left.

Install Minikube

The screenshot shows the minikube documentation website. The top navigation bar includes links for Documentation, Get Started!, Handbook, Basic controls, Deploying apps, Kubectl, Accessing apps, Addons, Configuration, Dashboard, Pushing images, Proxies and VPNs, Registries, Certificates, Offline usage, Host access, Persistent Volumes, Mounting filesystems, and File Sync. The main content area is titled "minikube start" and provides instructions for running local Kubernetes. It lists requirements like Docker and a host machine with 2 CPUs and 2GB of memory. A "What you'll need" section and an "Installation" step are also shown.

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: 6080
    targetPort: 5080
    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: 5080
```

```
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 (3)

- Cron Jobs
- Daemon Sets
- Deployments
- Jobs
- Pods
- Replica Sets
- Replication Controllers
- Stateful Sets

Service (2)

- Ingresses
- Services

Config and Storage (3)

- Config Maps
- Persistent Volume Claims
- Secrets

Storage Classes

Workload Status

Deployments

Pods

Replica Sets

Deployments

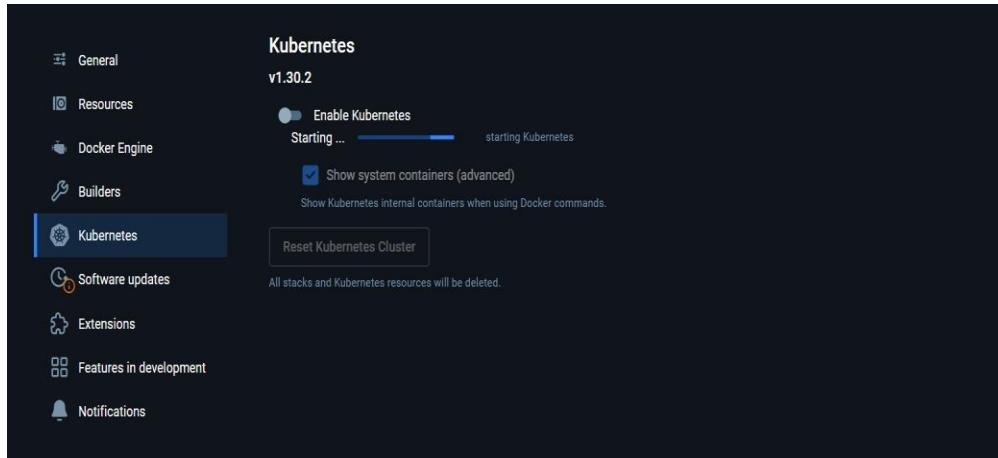
Name	Namespace	Labels	Pods	Created	Images
hello-world	default	app:hello-world	1 / 1	28 minutes ago	k8s.gcr.io/echoserver:1.4
hello-minikube	default	app:hello-minikube	1 / 1	36 minutes ago	k8s.gcr.io/echoserver:1.4
bad-test-app	default	-	5 / 5	42 minutes ago	hello-world

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

New Item

Enter an item name

> This field cannot be empty, please enter a valid name

Select an item type

Pipeline Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

OK

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

localhost:8080/view/all/job/kube/configure

Dashboard > All > kube > Configuration

Definition

Configure

General

Advanced Project Options

Pipeline

SCM ?

Git

Repositories ?

Repository URL ?

https://github.com/Wilsonbolledula/kube.git

Credentials ?

wilsonbolledula/*****

+ Add

Advanced

Add Repository

Save Apply

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

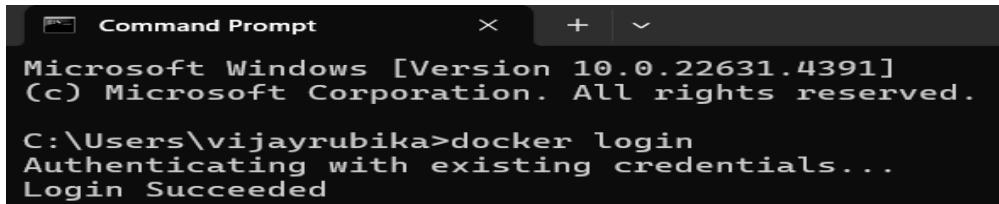
The screenshot shows the Jenkins interface for a job named 'kube'. On the left, there's a sidebar with options like Status, Changes, Build Now, Configure, Delete Pipeline, Stages, Rename, and Pipeline Syntax. The main area has a title 'kube' with a checkmark icon. Below it is a section titled 'Permalinks' containing a bulleted list of recent builds. To the right is a 'Build History' card showing two builds: #2 (Oct 25, 2024, 8:29 PM) and #1 (Oct 25, 2024, 8:27 PM). At the bottom of the card are links for 'Atom feed for all' and 'Atom feed for failures'.

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

The screenshot shows the Docker Desktop interface. On the left, there's a sidebar with options for Containers, Images, Volumes, Builds, Docker Scout, and Extensions. The main area shows the 'Images' tab with a list of images. One image is selected: 'my-kube:latest' (f62af06937). The details pane shows 'CREATED 3 days ago' and 'SIZE 919.50 MB'. There are buttons for 'Run' and 'Delete'. Below this, the 'Layers (20)' section lists the layers of the image, each with a file path and size. At the bottom, there's a note about Docker Scout analysis and a 'Start analysis' button.

Step-3:

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



```
Command Prompt
Microsoft Windows [Version 10.0.22631.4391]
(c) Microsoft Corporation. All rights reserved.

C:\Users\vijayrubika>docker login
Authenticating with existing credentials...
Login Succeeded
```

> 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:

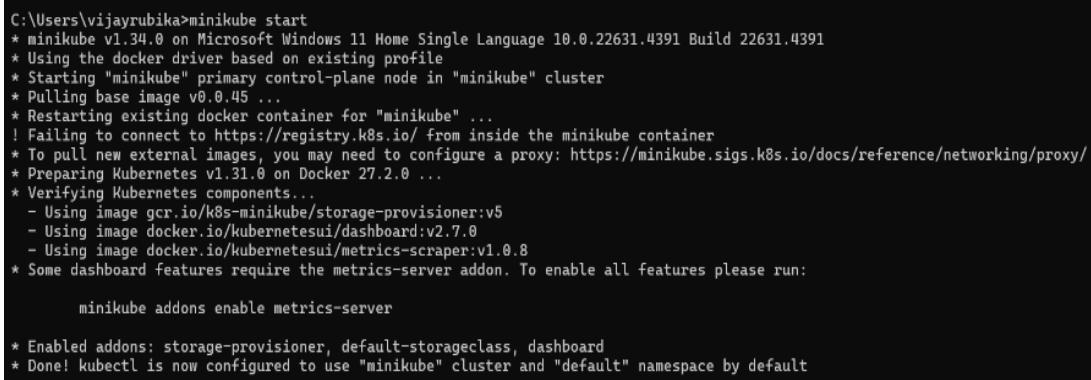
> Start the

Kubernetes

Syntax :

minikube

start



```
C:\Users\vijayrubika>minikube start
* minikube v1.34.0 on Microsoft Windows 11 Home Single Language 10.0.22631.4391 Build 22631.4391
* Using the docker driver based on existing profile
* Starting "minikube" primary control-plane node in "minikube" cluster
* Pulling base image v0.0.45 ...
* Restarting existing docker container for "minikube" ...
! Failing to connect to https://registry.k8s.io/ from inside the minikube container
* To pull new external images, you may need to configure a proxy: https://minikube.sigs.k8s.io/docs/reference/networking/proxy/
* Preparing Kubernetes v1.31.0 on Docker 27.2.0 ...
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
  - Using image docker.io/kubernetesui/dashboard:v2.7.0
  - Using image docker.io/kubernetesui/metrics-scraper:v1.0.8
* Some dashboard features require the metrics-server addon. To enable all features please run:
  minikube addons enable metrics-server

* Enabled addons: storage-provisioner, default-storageclass, dashboard
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```

>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”

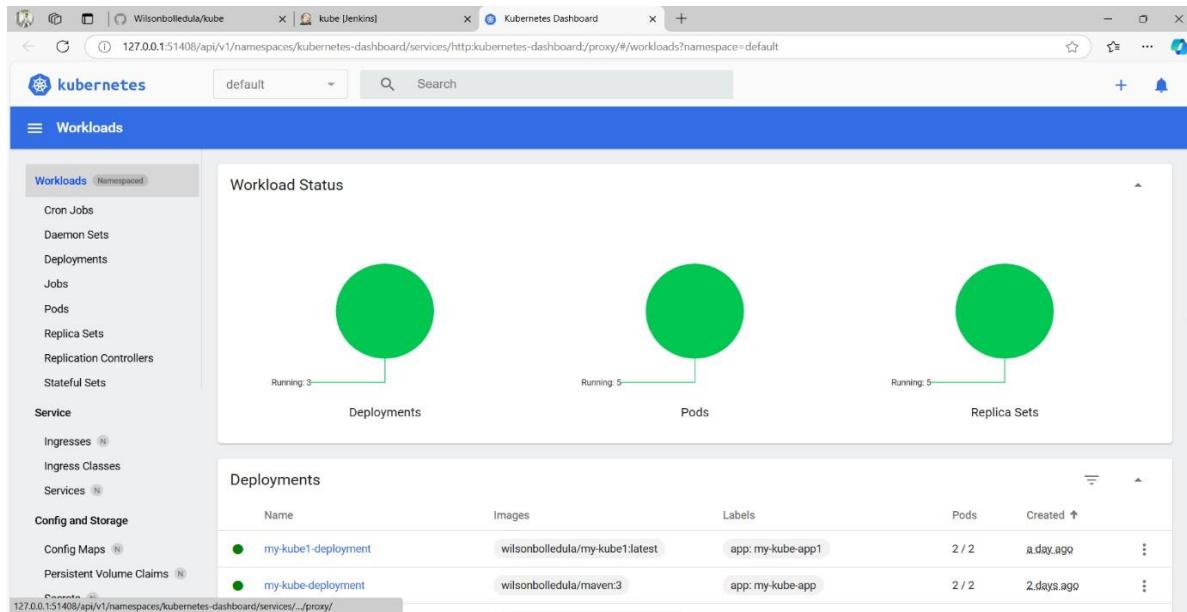
```
C:\Users\vijayrubika>kubectl get pods
NAME                               READY   STATUS    RESTARTS   AGE
my-kube-deployment-784d8954f7-6qwbm  1/1     Running   6 (6m40s ago)  47h
my-kube-deployment-784d8954f7-7f8df  1/1     Running   6 (6m40s ago)  47h
my-kube1-deployment-75db75cf4d-6s4fm 1/1     Running   2 (6m40s ago)  46h
my-kube1-deployment-75db75cf4d-hrzzn  1/1     Running   2 (6m40s ago)  46h
no

C: Microsoft Windows [Version 10.0.22631.4391]
NA (c) Microsoft Corporation. All rights reserved.
ku
my C:\Users\vijayrubika>docker login
my Authenticating with existing credentials...
my Login Succeeded
no

Command Prompt
```

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

That will open Kubernetes dashboard automatically on your default browser



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.

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)



```
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

NEXT WORK JakartaOne Livestream 2024

Join us on December 3 for our virtual conference that will give you a look at features from Jakarta EE 11 and other cloud native Java innovations.

Register

Learn More

Install your favorite desktop IDE packages

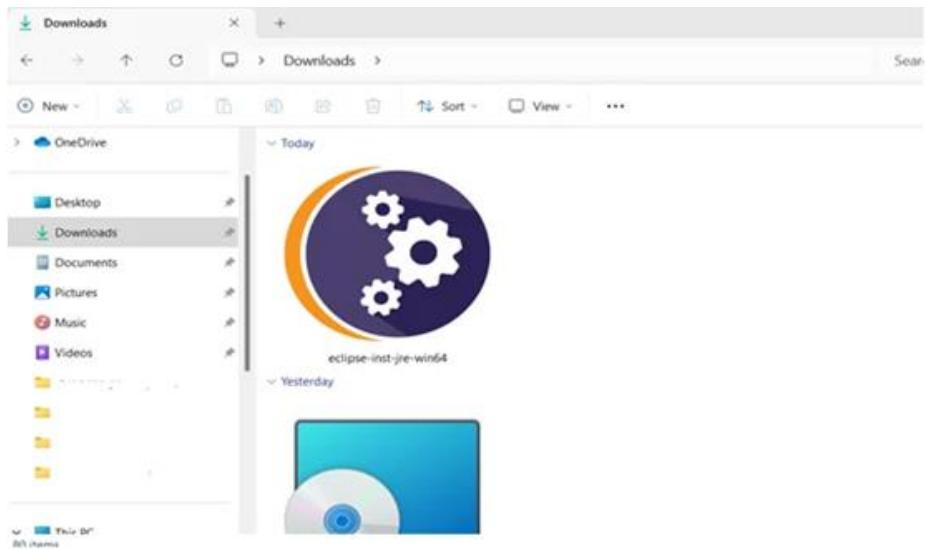
Learn More **Download v86_64**

Download Packages | Need Help

TEMURIN

The Eclipse Temurin™ project provides high-quality, TCK certified, OpenJDK runtimes and associated technology for use across the Java™ ecosystem.

Learn More **Download**



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 Developers

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 following 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

C:\Users\HP\eclipse\java-2025-09

 create start menu entry create desktop shortcut**INSTALLING**

BACK



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](#)

★ SPONSOR

eclipseinstaller by Oomph



Eclipse IDE for Java Developers

[details](#)

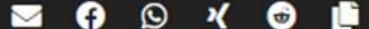
The essential tools for any Java developer, including a Java IDE, a Git client, XML Editor, Maven and Gradle integration.

Java 21+ VM

C:\Program Files\Java\jdk-23

Installation Folder

C:\eclipse



create start menu entry

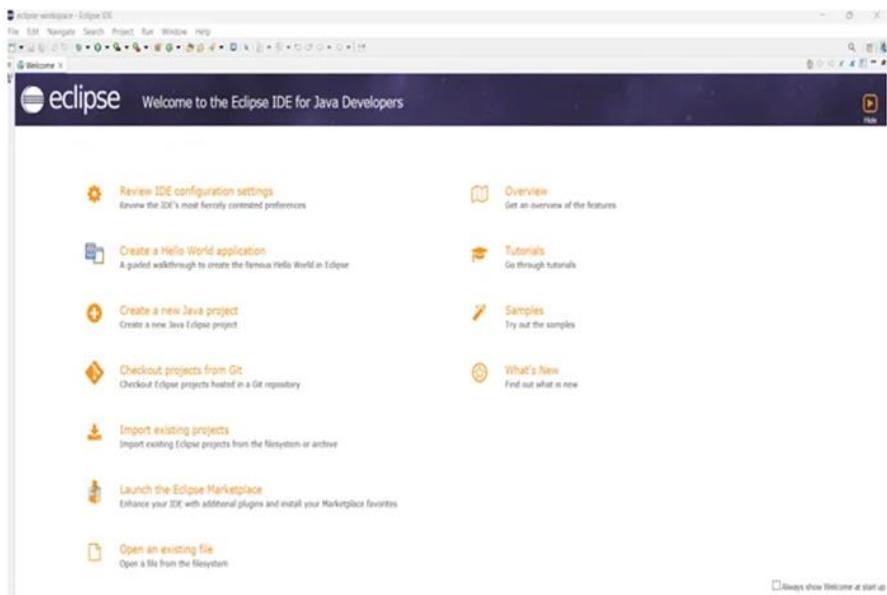
create desktop shortcut

► LAUNCH

[show readme file](#)

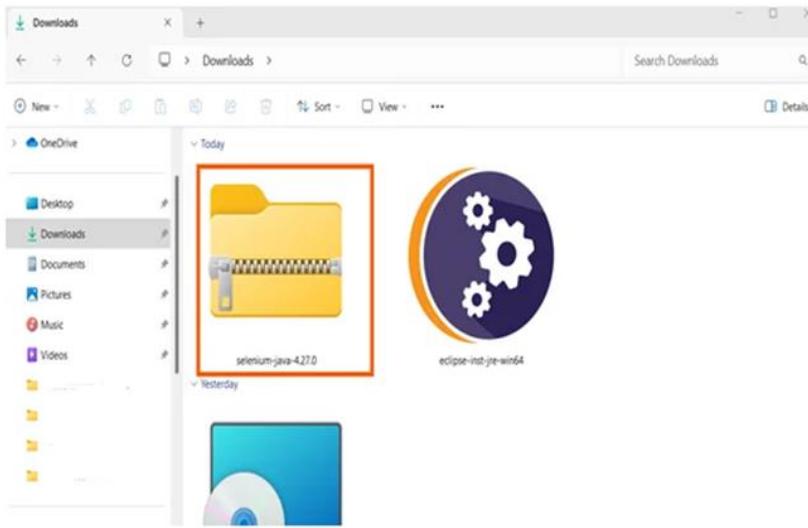
[open in system explorer](#)

[keep installer](#)

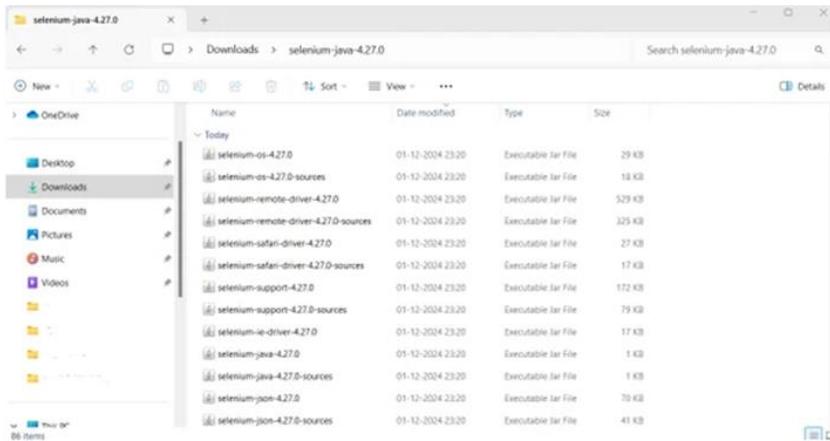


Install Selenium WebDriver –





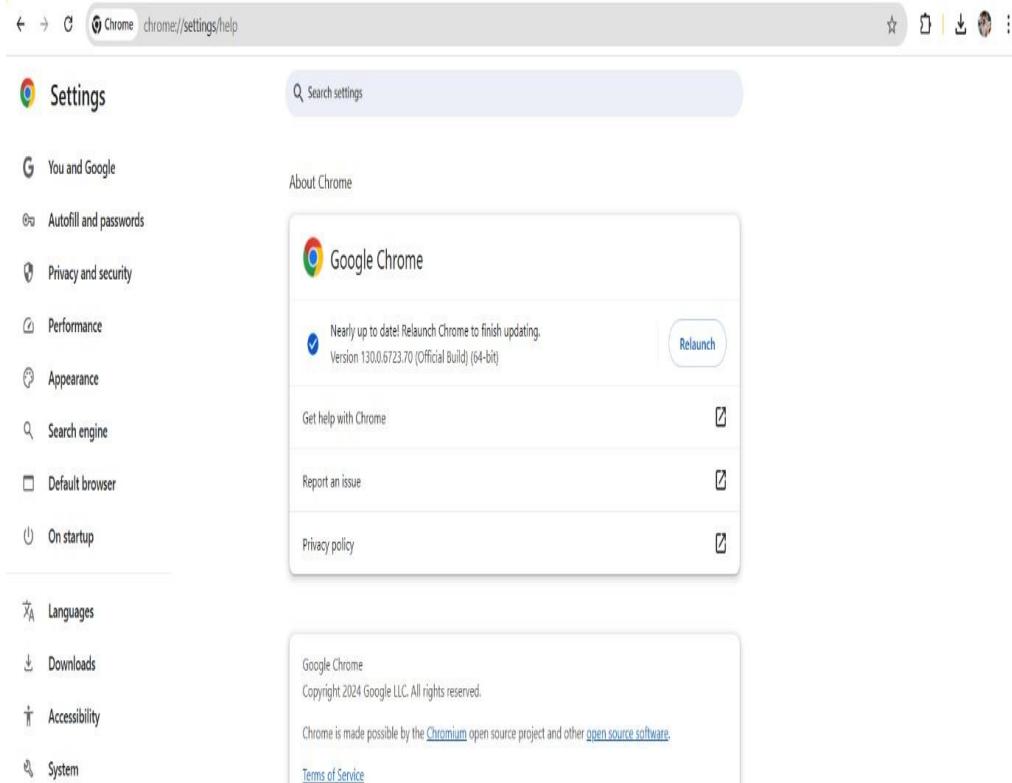
When the zip file is unzipped i.e. all the file present inside that zip are extracted, we will see the list of various JAR file which we will be using while configuring Selenium Webdriver and Eclipse IDE.



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.

The screenshot shows a browser window displaying the ChromeDriver download page from the Fugu showcase. The URL in the address bar is <https://fugu.showcase.cloud/chromedriver>. The page title is "Downloads" and the sub-section is "Current Releases". A warning message is displayed: "Warning: If you are using Chrome version 115 or newer, consult the [Chrome for Testing availability dashboard](#). This page provides convenient [JSON endpoints](#) for specific ChromeDriver version downloading." Below this, another bullet point states: "For lower versions of Chrome, see below for the version of ChromeDriver that supports it."

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



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

Chrome for Developers Get inspired Blog Docs ▾ New in Chrome Search

Capabilities Fugu showcase ChromeDriver

About ChromeDriver
Capabilities and ChromeOptions
Chrome Extensions
Contribute

Downloads

Stable Releases Canary Releases Version selection

Design documentation

Get started

Desktop **CLICK ON STABLE**

Version selection is the process of matching a Chrome binary of a given version to a compatible ChromeDriver binary.

For versions 115 and newer

Starting with M115 the ChromeDriver release process is integrated with that of Chrome. The latest Chrome + ChromeDriver releases per release channel (Stable, Beta, Dev, Canary) are available at the [Chrome for Testing \(CFT\) availability dashboard](#). As a result, you might no longer have a need for version selection — you could choose any available CFT version and download the correspondingly-versioned ChromeDriver binary.

For automated version downloading one can use the convenient [Cft JSON endpoints](#).

If you still have a need for version selection (e.g. to match a non-Cft Chrome binary with a compatible ChromeDriver binary), look up the Chrome binary's MAJOR.MINOR.BUILD version in the [latest-patch-versions-per-build JSON endpoints](#) to find the corresponding ChromeDriver version. If there is no entry for the MAJOR.MINOR.BUILD version yet, fall back to the [latest-versions-per-milestone JSON endpoint](#) instead. Alternatively, you can use the [LATEST_RELEASE_ endpoints](#) at the new location.

[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

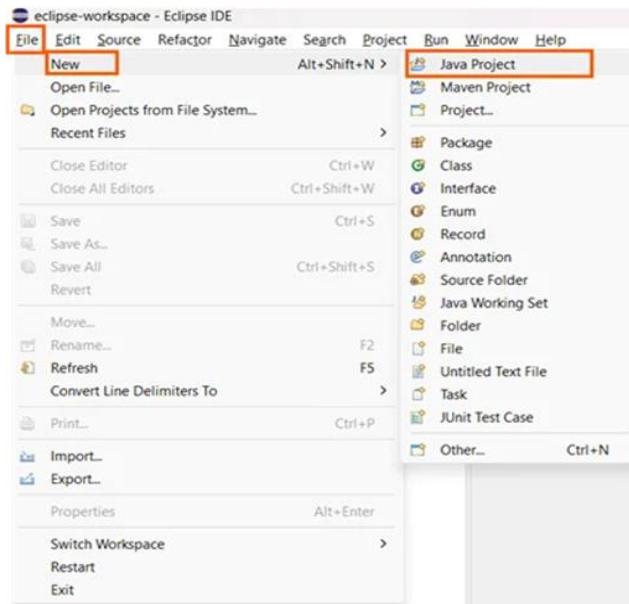
Channel	Version	Revision	Status
Stable	130.0.6723.91	r1356013	
Beta	131.0.6778.13	r1368529	
Dev	132.0.6793.2	r1372444	
Canary	132.0.6806.0	r1375157	

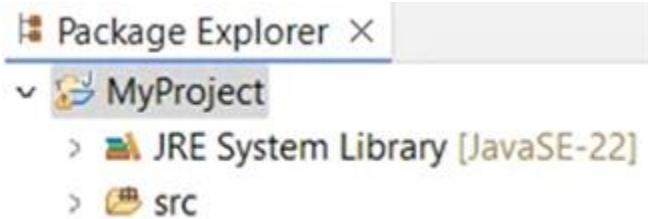
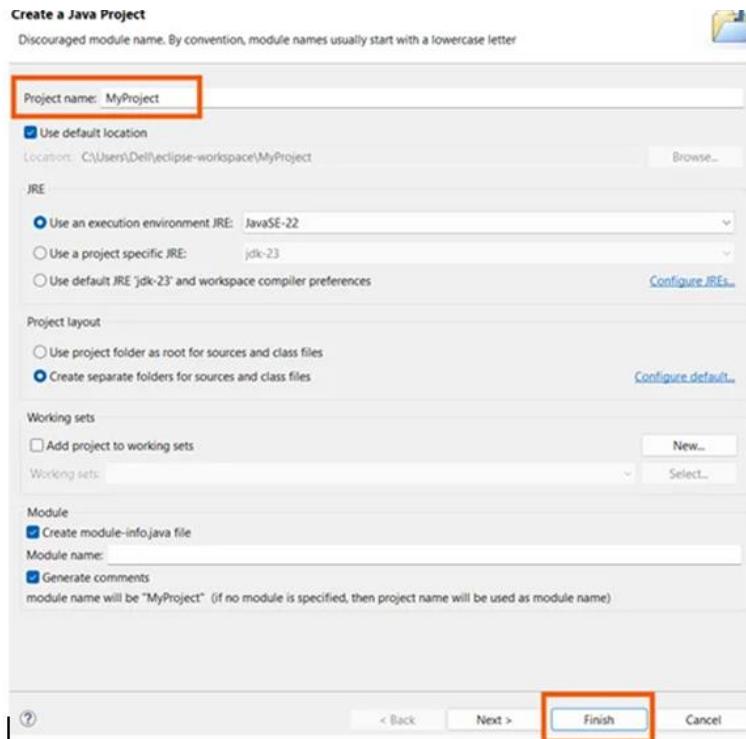
➤ Select according to your OS and Copy and paste chromedriver link in
chrome it will download automatically

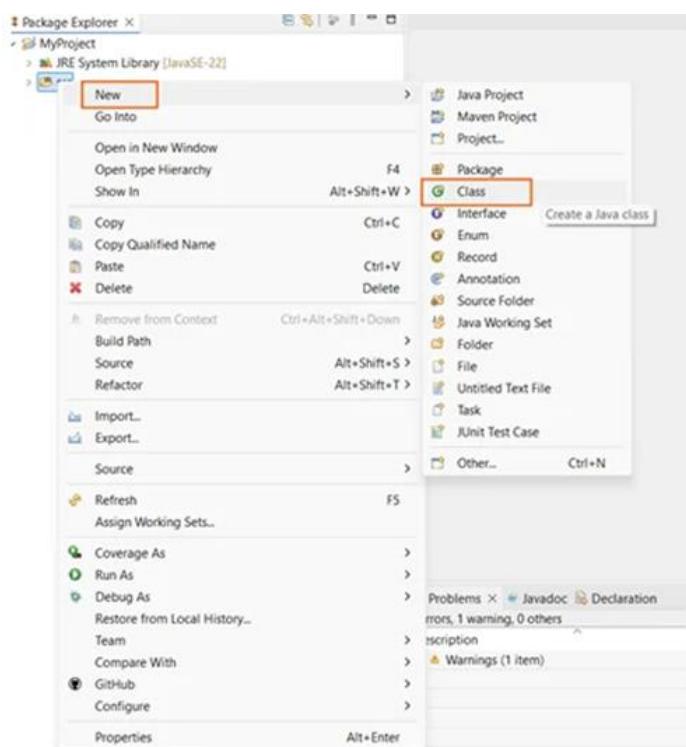
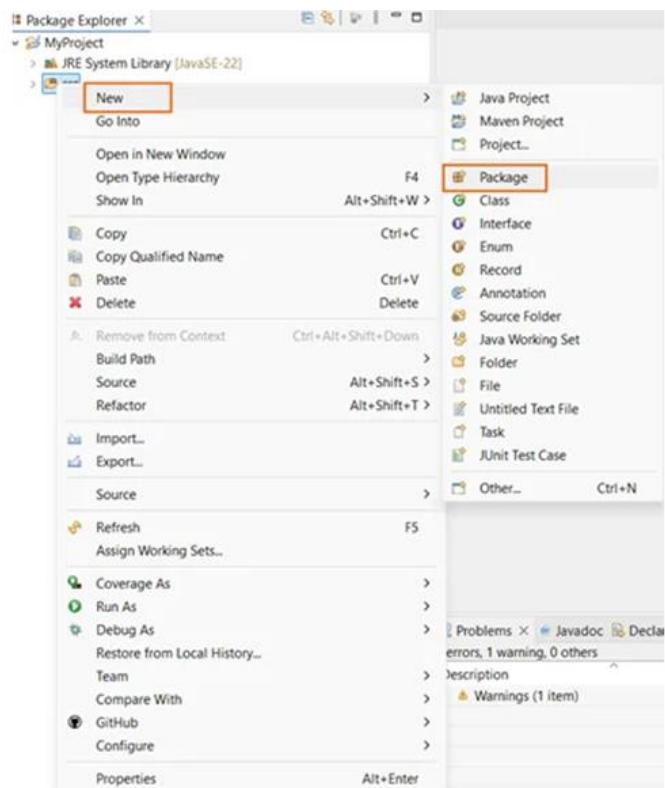
Version: 130.0.6723.91 (r1356013)

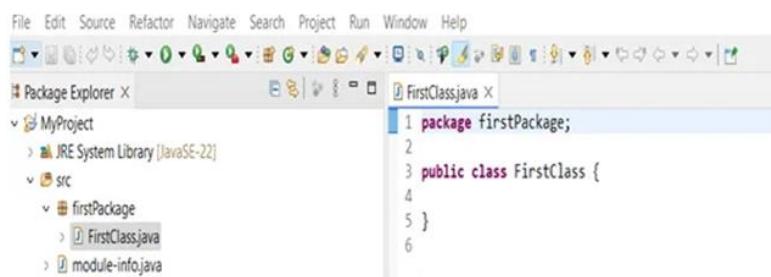
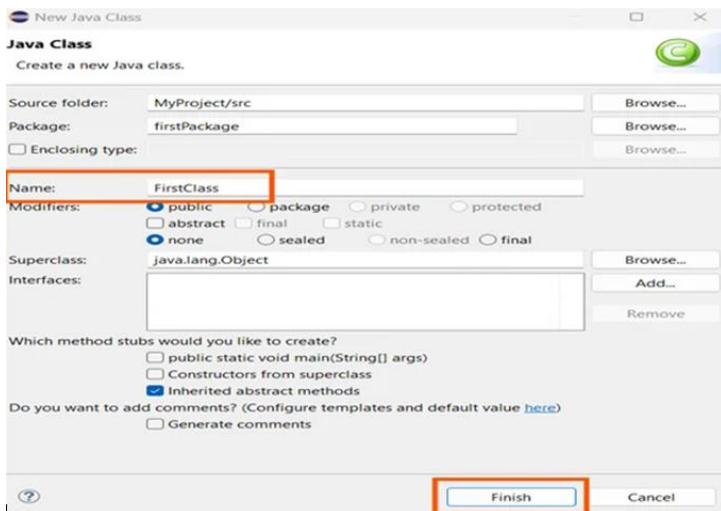
Binary	Platform	URL	HTTP status
chrome	linux64	https://storage.googleapis.com/chrome-for-testing-public/130.0.6723.91/linux64/chrome-linux64.zip	200
chrome	mac-arm64	https://storage.googleapis.com/chrome-for-testing-public/130.0.6723.91/mac-arm64/chrome-mac-arm64.zip	200
chrome	mac-x64	https://storage.googleapis.com/chrome-for-testing-public/130.0.6723.91/mac-x6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



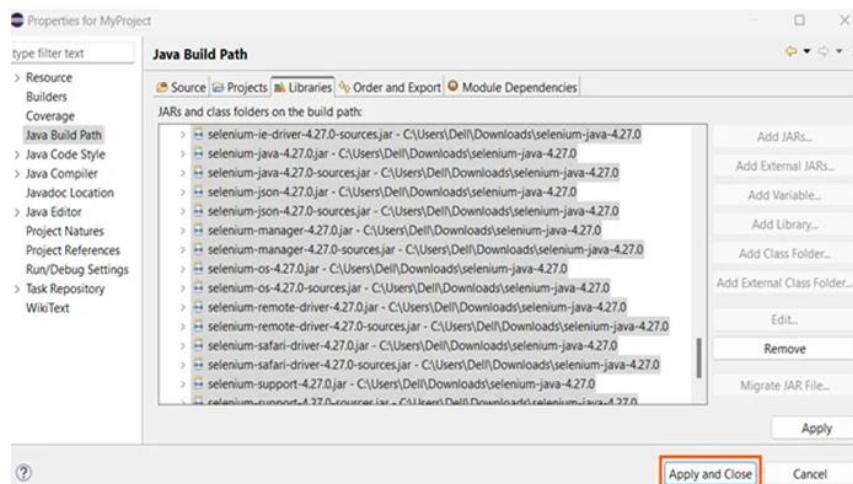






- Create a new Java Project.
- Right-click the project and select **Build Path > Configure Build Path....**
- Go to the **Libraries** tab.
- Click **Add External JARs....**

Add all the JAR files from the extracted Selenium folder.

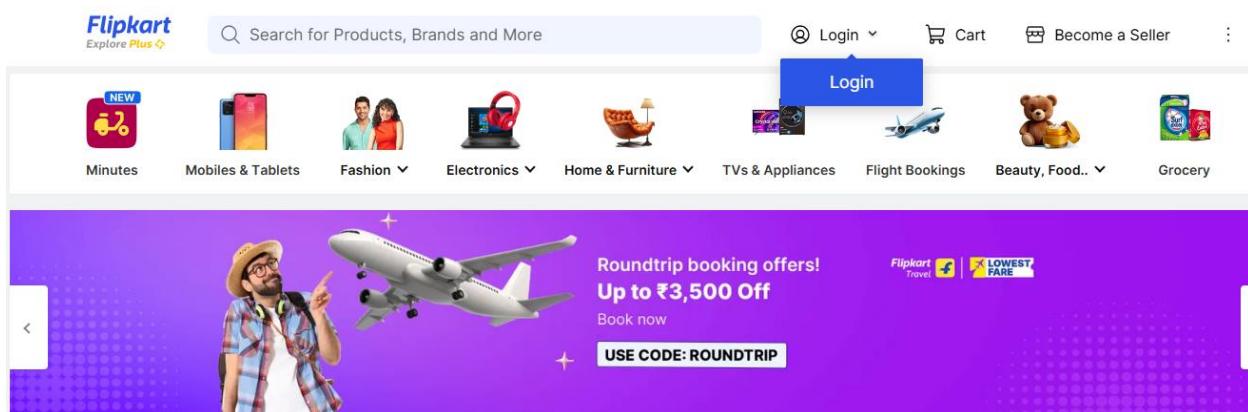


```
1 package firstPackage;
2
3 public class FirstClass {
4
5 }
6
```

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...');

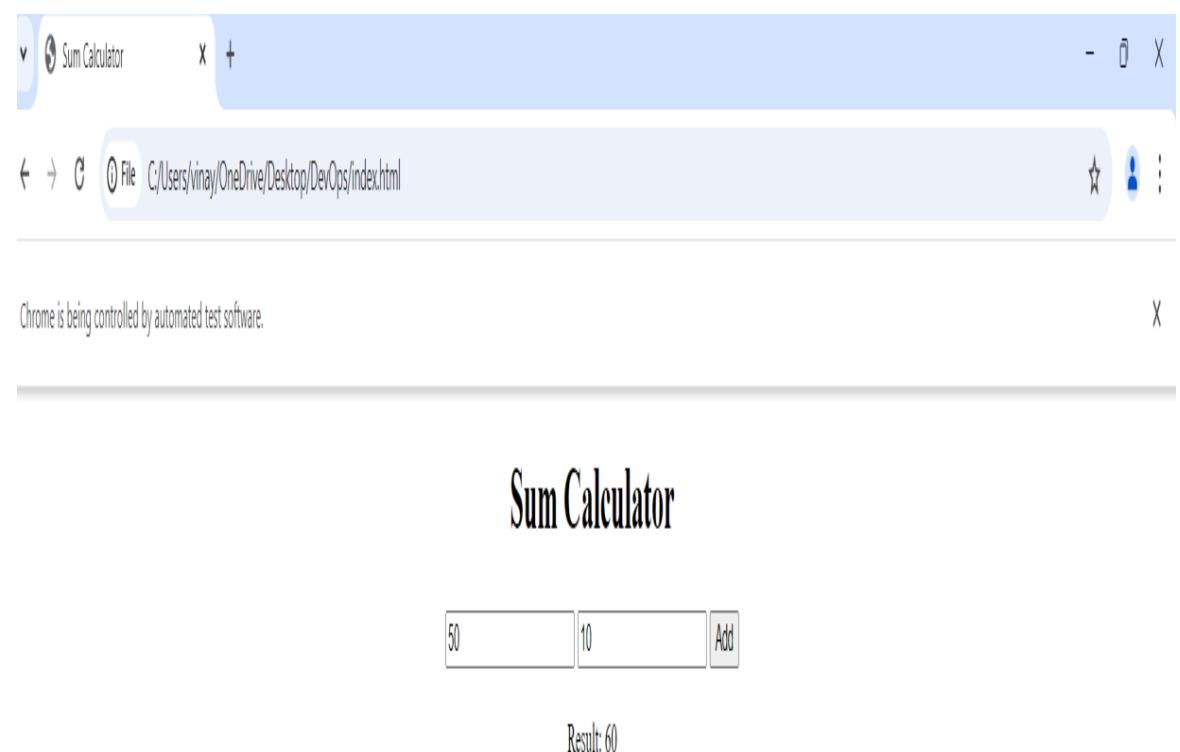
  process.stdin.setRawMode(true);
  process.stdin.resume();
  process.stdin.on('data', async () =>
  {

    await
    driver.
    quit();
  }
}
```

```
process  
s.exit(  
0);  
  
});  
}  
}  
  
runTest();
```

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

Output:



WEEK -12

Develop test cases for the above containerized application using selenium.

Prerequisites :

Docker Desktop

Python (for running scripts
locally if needed)
(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 try again.</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
WebDriver driver =
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 server time.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 used**docker-compose up --build**

This command will:

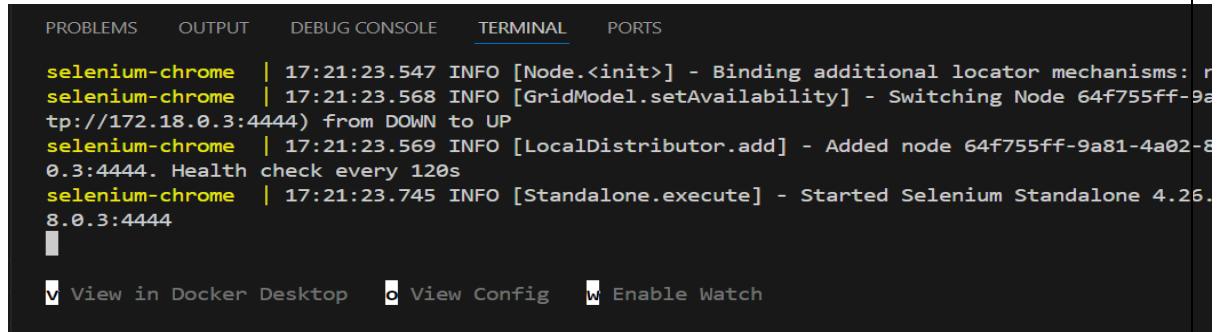
Build the Docker image for your HTML server.

Pull the Selenium standalone Chrome image.

Start both services.

Press **v** to navigate to docker.

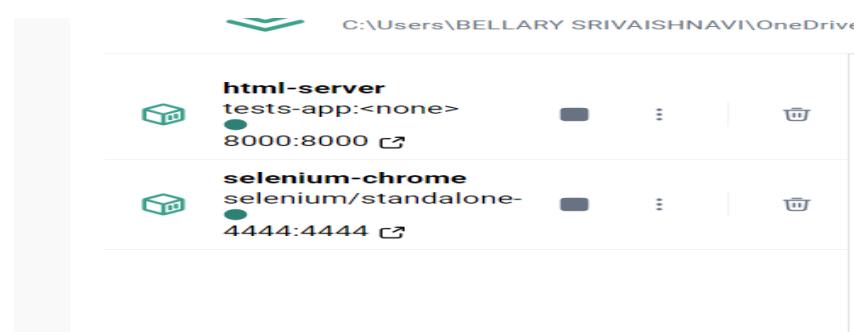
And Click on the links.



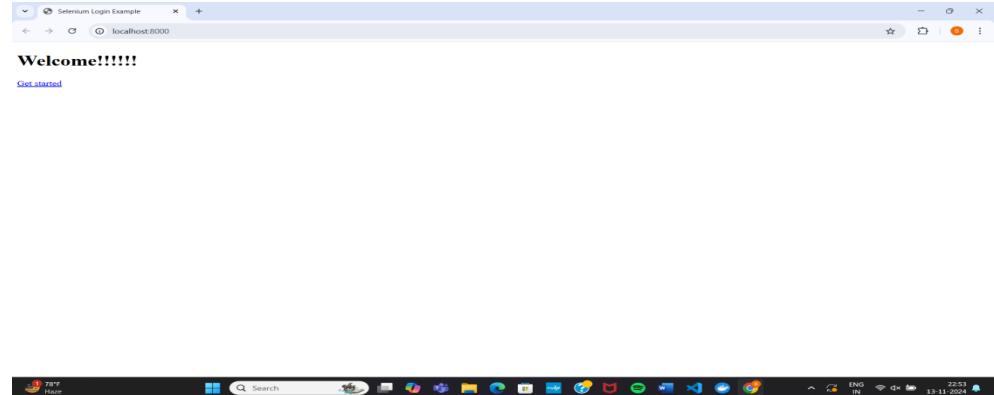
```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

selenium-chrome | 17:21:23.547 INFO [Node.<init>] - Binding additional locator mechanisms: r
selenium-chrome | 17:21:23.568 INFO [GridModel.setAvailability] - Switching Node 64f755ff-9a
tp://172.18.0.3:4444) from DOWN to UP
selenium-chrome | 17:21:23.569 INFO [LocalDistributor.add] - Added node 64f755ff-9a81-4a02-8
0.3:4444. Health check every 120s
selenium-chrome | 17:21:23.745 INFO [Standalone.execute] - Started Selenium Standalone 4.26.
8.0.3:4444

View in Docker Desktop  View Config  Enable Watch
```

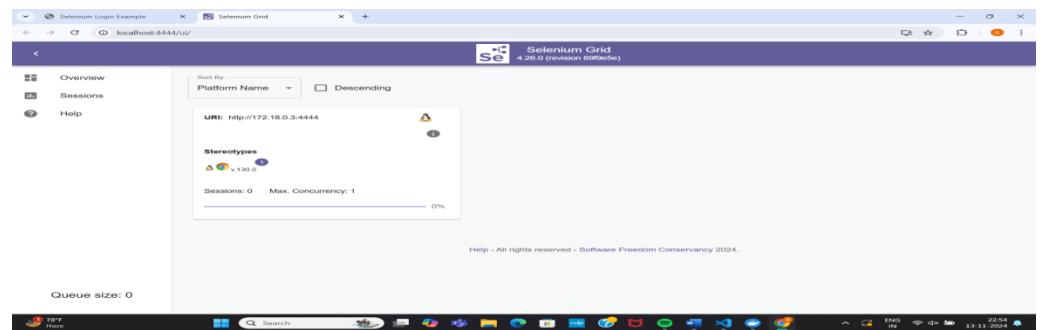


8000:8000



localhost:8000

4444:4444



Step 7:

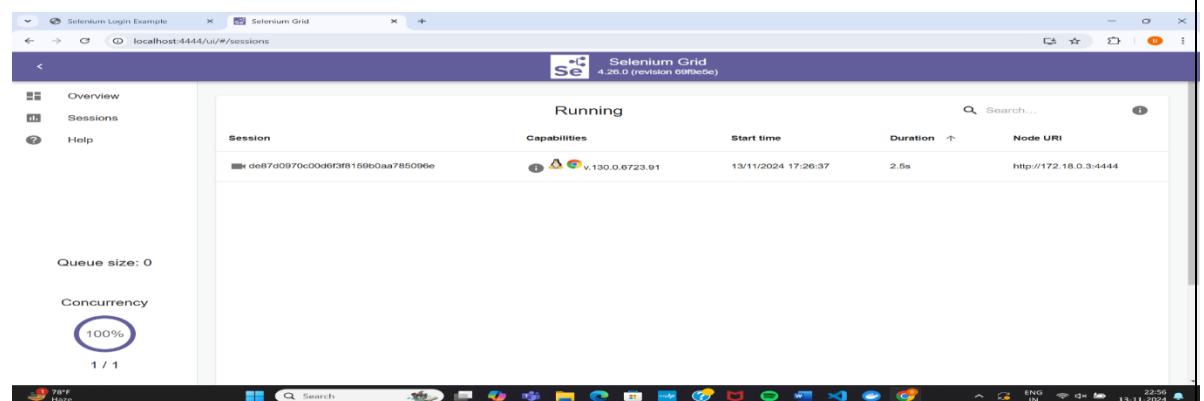
Run the Selenium Test Script

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

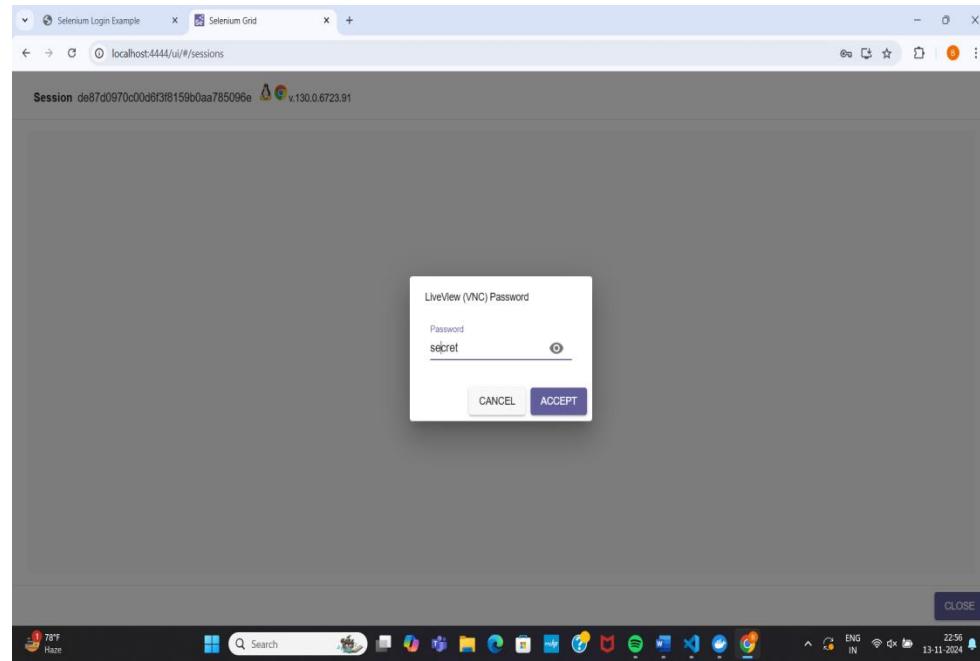
In terminal use, python SeleniumTest.py

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.)