Department of CSE

Laboratory Manual	
Course:	B.Tech.
Year & Semester:	III – I
Class:	CSE
Subject:	**DevOps Lab Manual**
Regulation:	R22

BALAJI INSTITUTE OF TECHNOLOGY AND SCIENCE (AUTONOMOUS)

B.Tech (Department of Computer Science & Engineering)

DEVOPS LAB

Course Outcomes:

- Understand the need of DevOps tools.
- Understand the environment for a software application development.
- Apply different project management, integration and development tools.
- Apply Docker Commands for content management.
- Use Selenium tool for automated testing of application.

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 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 for 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.

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

Step 1:

- ✓ Install Python from http://python.org
- ✓ Open powershell in windows
- ✓ Install Flask and MySQL with following commands
    - pip install flask
    - pip install flask-mysqldb

Step 2:

- ✓ Install xampp in windows
- ✓ Run xampp control panel then run apache and mysql
- ✓ Create database "mydb" in http://localhost/phpmyadmin
- ✓ Create a table

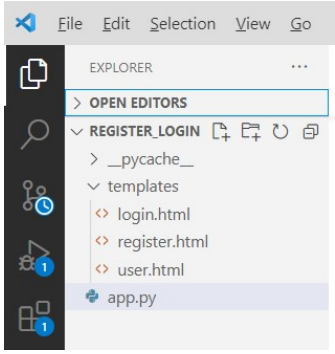
```
CREATE TABLE `user` (

  `userid` int(11) NOT NULL,

  `name` varchar(100) NOT NULL,

  `email` varchar(100) NOT NULL,

  `password` varchar(255) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

ALTER TABLE `user`

  ADD PRIMARY KEY (`userid`);
```

Step 3:

- ✓ Create a project (in powershell)
  >md Register_Login
- ✓ Change to Project
  >cd Register_Login
- ✓ Code . (it opens VS Code)

Step 4:

- ✓ Create app.py and write below code in the file
- ✓ Create templates folder under project
- ✓ Create register.html, login.html and user.html
- ✓ Write below code in respective fles
- ✓ Project Structure:



app.py

```
from flask import Flask, render_template, request, redirect, url_for, session

from flask_mysqldb import MySQL

import MySQLdb.cursors

import re

app = Flask(__name__)

app.secret_key = 'xyzsdfg'

app.config['MYSQL_HOST'] = 'localhost'

app.config['MYSQL_USER'] = 'root'

app.config['MYSQL_PASSWORD'] = ''

app.config['MYSQL_DB'] = 'mydb'

mysql = MySQL(app)

@app.route('/')
```

```
@app.route('/login', methods =['GET', 'POST'])

def login():

    mesage = ''

    if request.method == 'POST' and 'email' in request.form and 'password' in
request.form:

        email = request.form['email']

        password = request.form['password']

        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)

        cursor.execute('SELECT * FROM user WHERE email = % s AND password = % s',
(email, password, ))

        user = cursor.fetchone()

        if user:

            session['loggedin'] = True

            session['userid'] = user['userid']

            session['name'] = user['name']

            session['email'] = user['email']

            mesage = 'Logged in successfully !'

            return render_template('user.html', mesage = mesage)

        else:

            mesage = 'Please enter correct email / password !'

    return render_template('login.html', mesage = mesage)

@app.route('/logout')

def logout():
```

```
    session.pop('loggedin', None)
    session.pop('userid', None)
    session.pop('email', None)
    return redirect(url_for('login'))
@app.route('/register', methods =['GET', 'POST'])
def register():
    mesage = ''
    if request.method == 'POST' and 'name' in request.form and 'password' in
request.form and 'email' in request.form :
        userName = request.form['name']
        password = request.form['password']
        email = request.form['email']
        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
        cursor.execute('SELECT * FROM user WHERE email = % s', (email, ))
        account = cursor.fetchone()
        if account:
            mesage = 'Account already exists !'
        elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):
            mesage = 'Invalid email address !'
        elif not userName or not password or not email:
            mesage = 'Please fill out the form !'
        else:
```

```
            cursor.execute('INSERT INTO user VALUES (NULL, % s, % s, % s)', (userName,
email, password, ))
            mysql.connection.commit()
            mesage = 'You have successfully registered !'
    elif request.method == 'POST':
        mesage = 'Please fill out the form !'
    return render_template('register.html', mesage = mesage)


if __name__ == "__main__":
    app.run()
```

login.html:

```
<html>

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<title>User Login Form</title>

<link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/css/bootstrap.min.css">

</head>

<body>

<div class="container">

        <h2>User Login</h2>

        <form action="{{ url_for('login') }}" method="post">

           {% if mesage is defined and mesage %}

                        <div class="alert alert-warning">{{ mesage }}</div>

                {% endif %}

                <div class="form-group">

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

                        <input type="email" class="form-control" id="email"
name="email" placeholder="Enter email" name="email">

                </div>

                <div class="form-group">

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

```
                    <input type="password" class="form-control" id="password"
name="password" placeholder="Enter password" name="pswd">

            </div>

            <button type="submit" class="btn btn-primary">Login</button>

            <p class="bottom">Dont't have an account?  <a class="bottom"
href="{{url_for('register')}}"> Register here</a></p>

        </form>

</div>

</body>

</html>
```

rgister.html:

```
<html>

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<title>User Registeration Form</title>

<link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/css/bootstrap.min.css">

</head>

<body>

<div class="container">

        <h2>User Registration</h2>

        <form action="{{ url_for('register') }}" method="post">

      {% if mesage is defined and mesage %}

                        <div class="alert alert-warning">{{ mesage }}</div>

                {% endif %}

                <div class="form-group">

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

                        <input type="text" class="form-control" id="name" name="name"
placeholder="Enter name" name="name">

                </div>

                <div class="form-group">

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

```
                <input type="email" class="form-control" id="email"
name="email" placeholder="Enter email" name="email">

            </div>

            <div class="form-group">

                <label for="pwd">Password:</label>

                <input type="password" class="form-control" id="password"
name="password" placeholder="Enter password" name="pswd">

            </div>

            <button type="submit" class="btn btn-primary">Register</button>

            <p class="bottom">Already have an account?  <a class="bottom"
href="{{url_for('login')}}"> Login here</a></p>

        </form>

</div>

</body>

</html>
```

user.html:

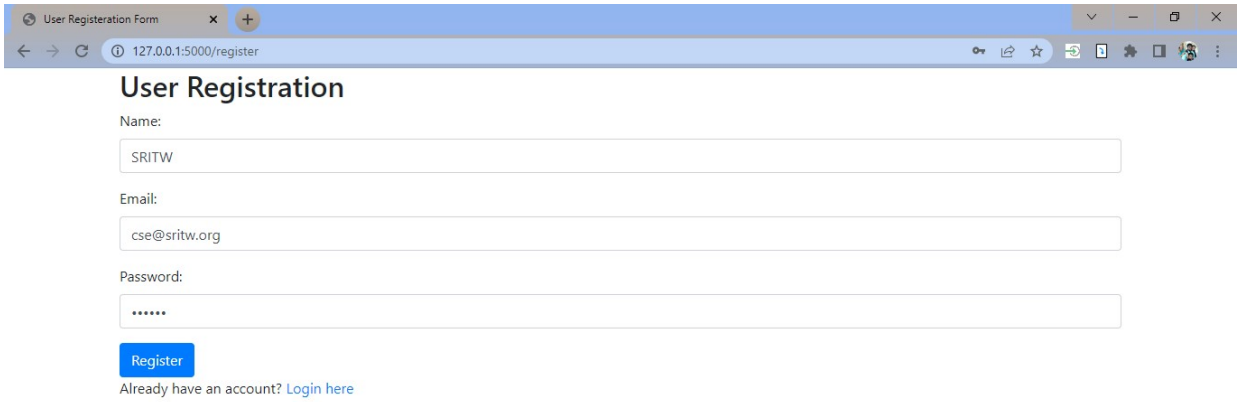
```
<html>

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<title>User Account</title>

<link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/css/bootstrap.min.css">

</head>

<body>

<div class="container">

        <div class="row">

                <h1>User Profile</h1>

        </div>

        <br>

        <div class="row">

                Logged in : <strong>{{session.name}} | <a href="{{ url_for('logout') }}">
Logout</a>

        </div>

        <br><br>

        <div class="row">


                <h2>Welcome to the user profile page...</h2>
```

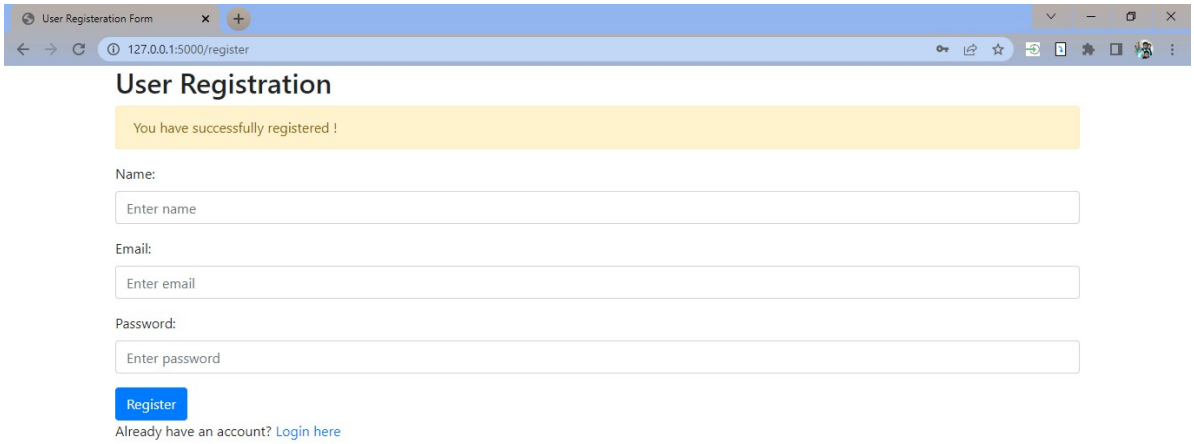
```
        </div>
</div>
</body>
</html>
```

Step5:

- ✓ Select app.py then run it.
- ✓ Open http://127.0.0.1:5000/ in browser

Output:


User Registeration Form
127.0.0.1:5000/register
User Registration
Name:
SRITW
Email:
cse@sritw.org
Password:
Register
Already have an account? Login here


User Registeration Form
127.0.0.1:5000/register
User Registration
You have successfully registered !
Name:
Enter name
Email:
Enter email
Password:
Enter password
Register
Already have an account? Login here


User Login Form
127.0.0.1:5000/login
User Login
Email:
cse@sritw.org
Password:
Login
Dont't have an account? Register here


User Account
127.0.0.1:5000/login
User Profile
Logged in :SRITW | Logout
Welcome to the user profile page...

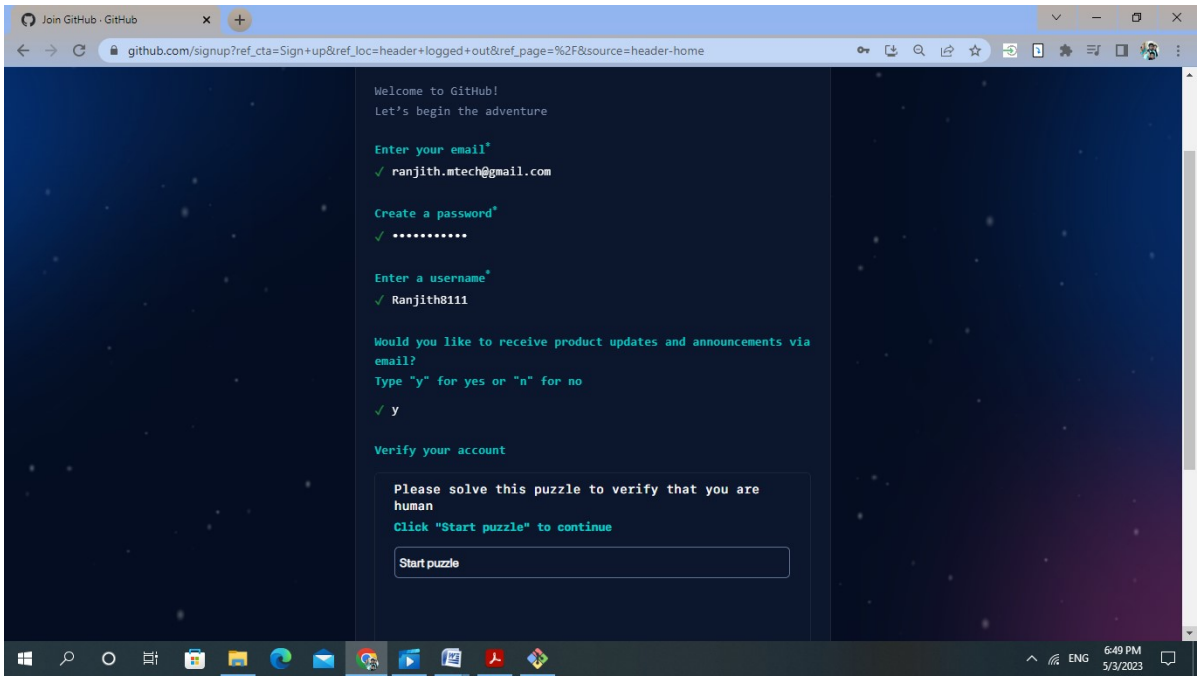
2. Explore Git and GitHub commands.

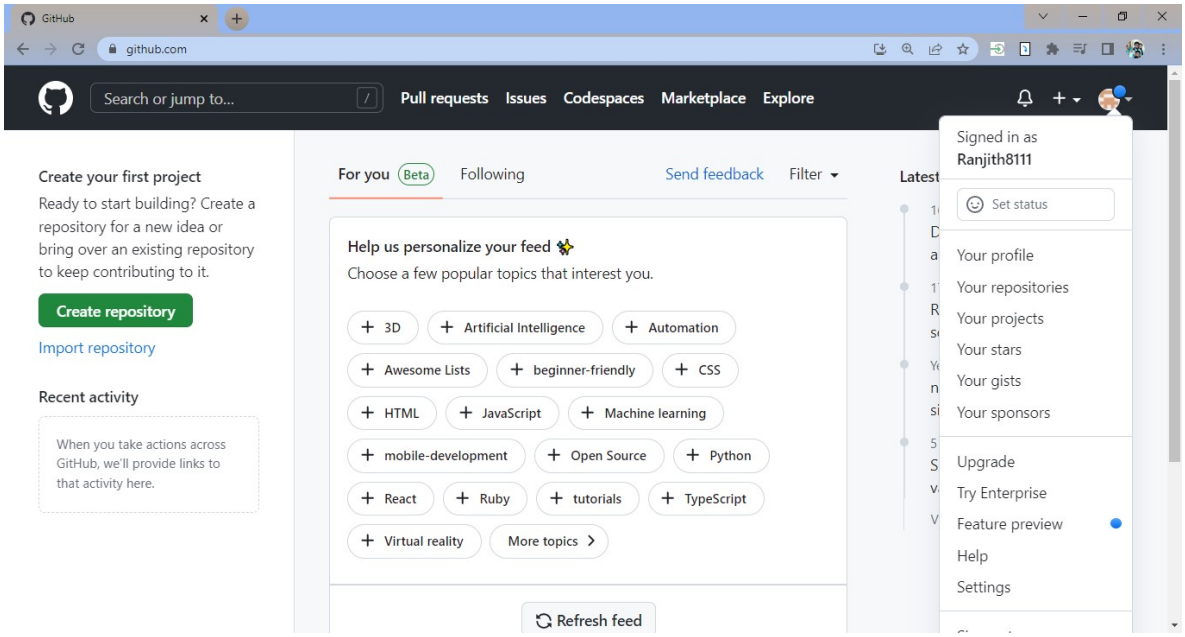
Step 1:

- ✓ Download gibash from https://git-scm.com/
- ✓ Install gitbash

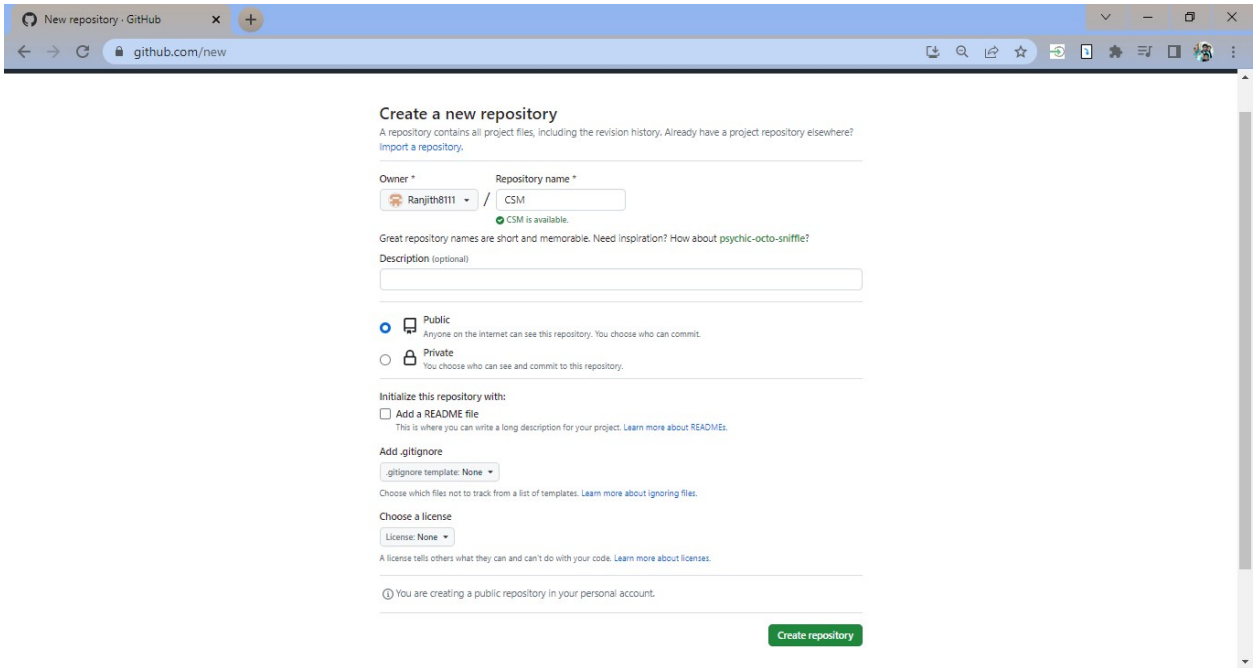
Step 2:

- ✓ Create an account with https://github.com/





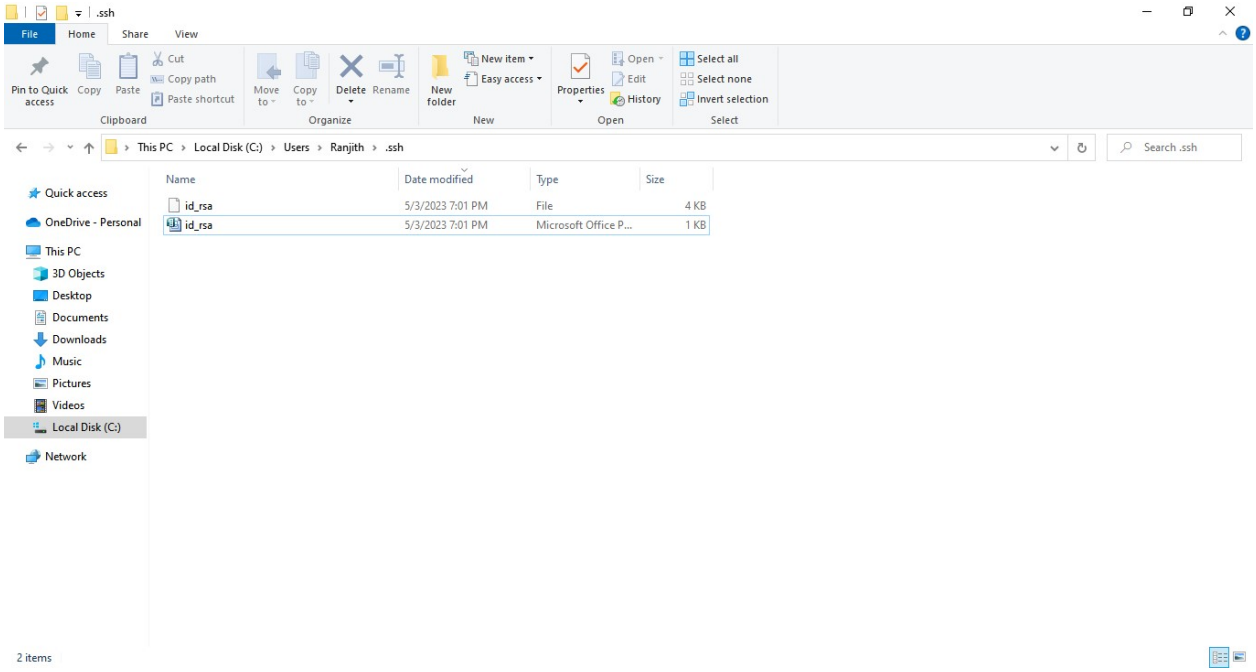
✓ Create a repository



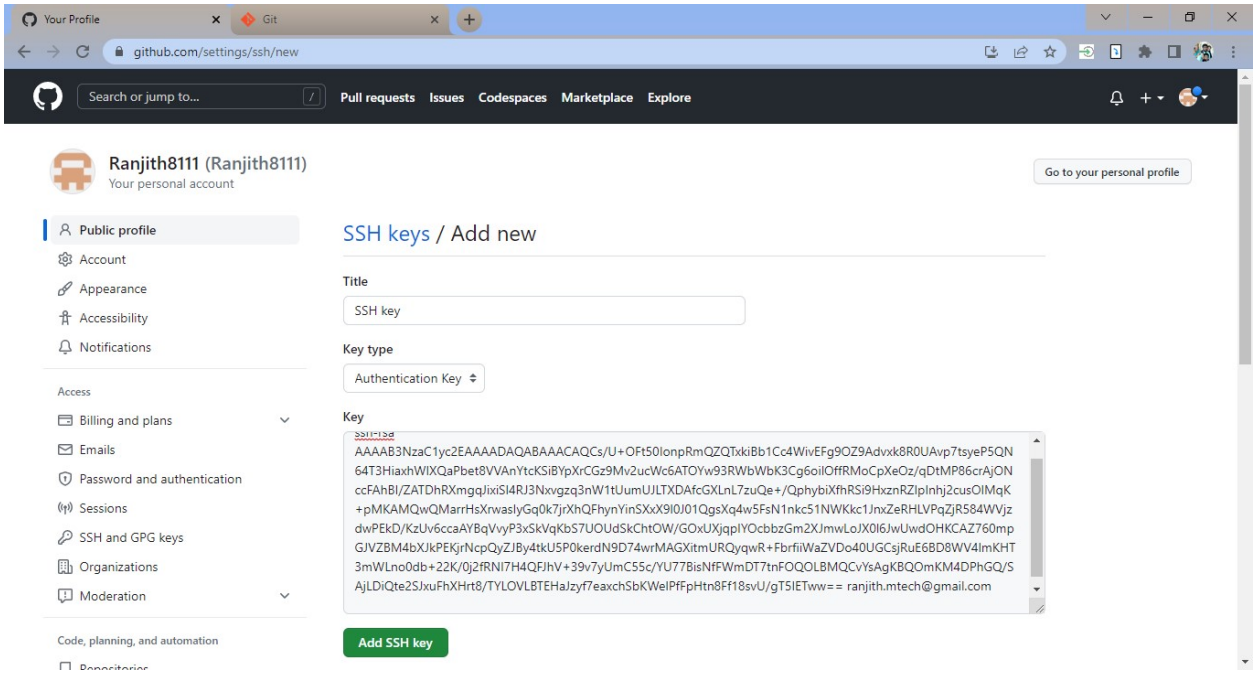
✓ Create SSH key in gitbash

```
MINGW64:/c/Users/Ranjith

Ranjith@DESKTOP-3AL783Q MINGW64 ~ (main)
$ ssh-keygen -t rsa -b 4096 -C "ranjith.mtech@gmail.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/Ranjith/.ssh/id_rsa):
Created directory '/c/Users/Ranjith/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/Ranjith/.ssh/id_rsa
Your public key has been saved in /c/Users/Ranjith/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:BYxOh3UN+zff6kPaEZye03EbbWChno/1V0f1R8z/yXU ranjith.mtech@gmail.com
The key's randomart image is:
+---[RSA 4096]----+
|     ..oooo oo.+o|
|      ..oo . o+.=|
|          . oo o*|
|         . . o++E|
|        S   o.+*&|
|             +**O|
|            .+.o=|
|            . o..|
|             .o. |
+----[SHA256]-----+

Ranjith@DESKTOP-3AL783Q MINGW64 ~ (main)
$
```

✓ Open and copy content of **id_rsa** from **.ssh** folder in user folder

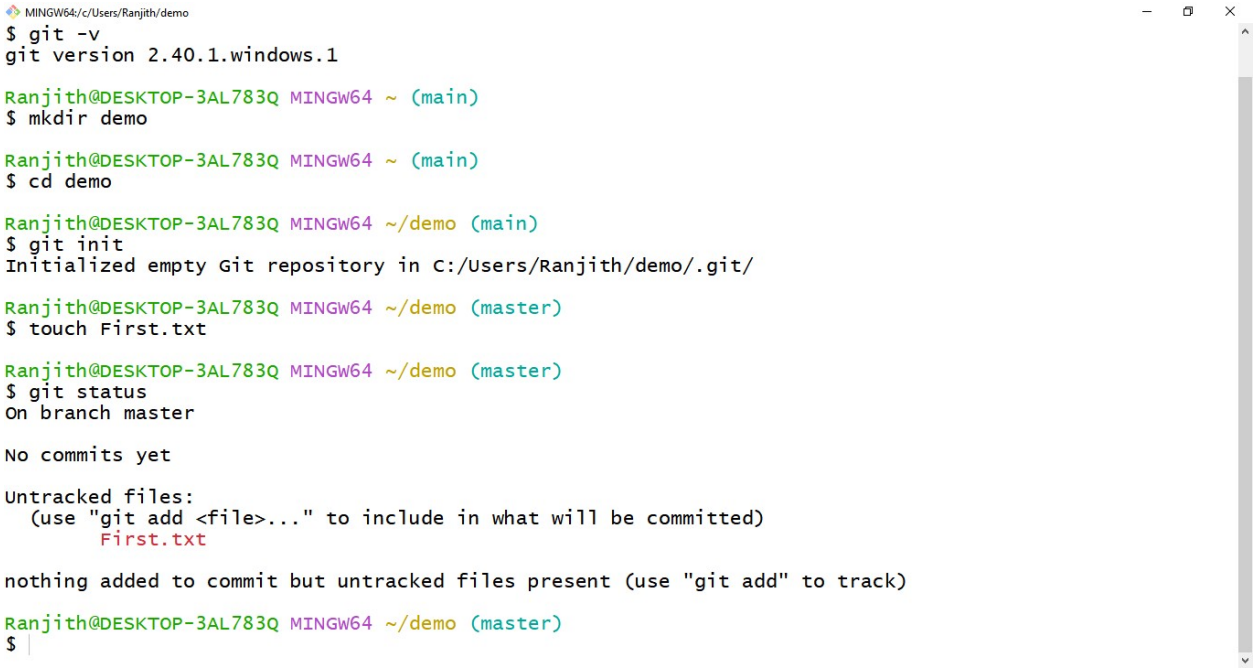


✓ Paste the content in key field in SSH keys/ Add new



Step 3:

✓ Do following git commands in gitbash



MINGW64:/c/Users/Ranjith/demo

```
Ranjith@DESKTOP-3AL783Q MINGW64 ~/demo (master)
$ git add First.txt

Ranjith@DESKTOP-3AL783Q MINGW64 ~/demo (master)
$ git status
On branch master

No commits yet

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


Ranjith@DESKTOP-3AL783Q MINGW64 ~/demo (master)
$ git log
fatal: your current branch 'master' does not have any commits yet

Ranjith@DESKTOP-3AL783Q MINGW64 ~/demo (master)
$
```

MINGW64:/c/Users/Ranjith/demo

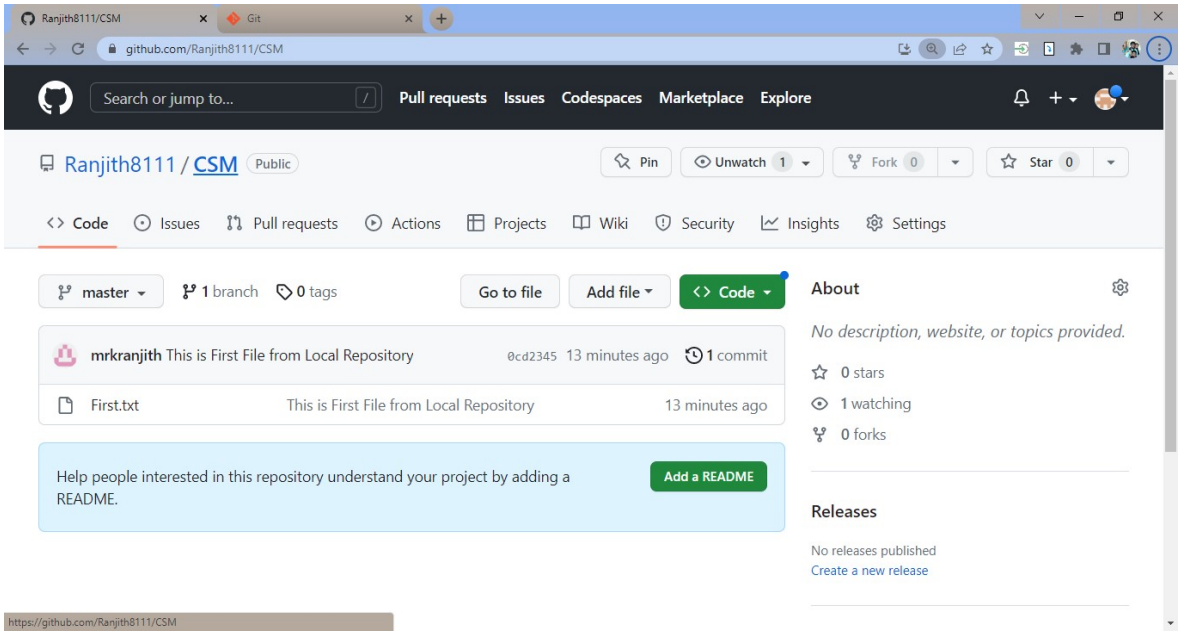
```
Ranjith@DESKTOP-3AL783Q MINGW64 ~/demo (master)
$ git commit -m "This is First File from Local Repository"
[master (root-commit) 0cd2345] This is First File from Local Repository
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 First.txt

Ranjith@DESKTOP-3AL783Q MINGW64 ~/demo (master)
$ git log
commit 0cd23455230d8f9350dec3c0ee065df8c53c0790 (HEAD -> master)
Author: M Ranjith Kumar <m_ranjithkumar@sritw.org>
Date:   Wed May 3 19:11:39 2023 +0530

    This is First File from Local Repository

Ranjith@DESKTOP-3AL783Q MINGW64 ~/demo (master)
$ git status
On branch master
nothing to commit, working tree clean

Ranjith@DESKTOP-3AL783Q MINGW64 ~/demo (master)
$
```

MINGW64:/c/Users/Ranjith/demo

```
Ranjith@DESKTOP-3AL783Q MINGW64 ~/demo (master)
$ git branch branch_1

Ranjith@DESKTOP-3AL783Q MINGW64 ~/demo (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        CSM/

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

Ranjith@DESKTOP-3AL783Q MINGW64 ~/demo (master)
$ git log
commit 0cd23455230d8f9350dec3c0ee065df8c53c0790 (HEAD -> master, branch_1)
Author: M Ranjith Kumar <m_ranjithkumar@sritw.org>
Date:   Wed May 3 19:11:39 2023 +0530

    This is First File from Local Repository

Ranjith@DESKTOP-3AL783Q MINGW64 ~/demo (master)
$ git checkout branch_1
Switched to branch 'branch_1'

Ranjith@DESKTOP-3AL783Q MINGW64 ~/demo (branch_1)
$ git log --oneline
0cd2345 (HEAD -> branch_1, master) This is First File from Local Repository

Ranjith@DESKTOP-3AL783Q MINGW64 ~/demo (branch_1)
$
```

MINGW64:/c/Users/Ranjith/demo

```
Ranjith@DESKTOP-3AL783Q MINGW64 ~/demo (branch_1)
$ git checkout master
Switched to branch 'master'

Ranjith@DESKTOP-3AL783Q MINGW64 ~/demo (master)
$ git remote add origin git@github.com:Ranjith8111/CSM.git

Ranjith@DESKTOP-3AL783Q MINGW64 ~/demo (master)
$ git push -u origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 243 bytes | 243.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:Ranjith8111/CSM.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.

Ranjith@DESKTOP-3AL783Q MINGW64 ~/demo (master)
$
```


Ranjith8111/CSM
Git
github.com/Ranjith8111/CSM
Search or jump to...
Pull requests
Issues
Codespaces
Marketplace
Explore
Ranjith8111 / CSM
Public
Pin
Unwatch 1
Fork 0
Star 0
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
master
1 branch
0 tags
Go to file
Add file
Code
mrkranjith This is First File from Local Repository
0cd2345 13 minutes ago
1 commit
First.txt
This is First File from Local Repository
13 minutes ago
Help people interested in this repository understand your project by adding a README.
Add a README
About
No description, website, or topics provided.
0 stars
1 watching
0 forks
Releases
No releases published
Create a new release
https://github.com/Ranjith8111/CSM

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

Do the following git commends

```
MINGW64:/c/Users/Ranjith/demo

Ranjith@DESKTOP-3AL783Q MINGW64 ~/demo (master)
$ cp -r ../Desktop/M.Ranjith_Kumar/DevOps/Lab/Exp1 .

Ranjith@DESKTOP-3AL783Q MINGW64 ~/demo (master)
$ ls
CSM/  Exp1/  First.txt

Ranjith@DESKTOP-3AL783Q MINGW64 ~/demo (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        CSM/
        Exp1/

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

Ranjith@DESKTOP-3AL783Q MINGW64 ~/demo (master)
$ git add Exp1/

Ranjith@DESKTOP-3AL783Q MINGW64 ~/demo (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   Exp1/__pycache__/app.cpython-311.pyc
        new file:   Exp1/app.py
        new file:   Exp1/templates/login.html
        new file:   Exp1/templates/register.html
        new file:   Exp1/templates/user.html

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


Ranjith@DESKTOP-3AL783Q MINGW64 ~/demo (master)
$
```

```
MINGW64:/c/Users/Ranjith/demo
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   Exp1/__pycache__/app.cpython-311.pyc
        new file:   Exp1/app.py
        new file:   Exp1/templates/login.html
        new file:   Exp1/templates/register.html
        new file:   Exp1/templates/user.html

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


Ranjith@DESKTOP-3AL783Q MINGW64 ~/demo (master)
$ git commit -m "This is DevOps Lab Exp1 Program"
[master c8d987e] This is DevOps Lab Exp1 Program
 5 files changed, 156 insertions(+)
 create mode 100644 Exp1/__pycache__/app.cpython-311.pyc
 create mode 100644 Exp1/app.py
 create mode 100644 Exp1/templates/login.html
 create mode 100644 Exp1/templates/register.html
 create mode 100644 Exp1/templates/user.html

Ranjith@DESKTOP-3AL783Q MINGW64 ~/demo (master)
$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

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

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

Ranjith@DESKTOP-3AL783Q MINGW64 ~/demo (master)
$
```

MINGW64:/c/Users/Ranjith/demo

```
Ranjith@DESKTOP-3AL783Q MINGW64 ~/demo (master)
$ git push -u origin master
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 4 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (10/10), 4.67 KiB | 2.33 MiB/s, done.
Total 10 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To github.com:Ranjith8111/CSM.git
   0cd2345..c8d987e  master -> master
branch 'master' set up to track 'origin/master'.

Ranjith@DESKTOP-3AL783Q MINGW64 ~/demo (master)
$
```

Ranjith8111/CSM | Git

github.com/Ranjith8111/CSM

Search or jump to... | Pull requests | Issues | Codespaces | Marketplace | Explore

Ranjith8111 / CSM Public

Pin | Unwatch 1 | Fork 0 | Star 0

Code | Issues | Pull requests | Actions | Projects | Wiki | Security | Insights | Settings

master | 1 branch | 0 tags | Go to file | Add file | Code

Ranjith8111 This is DevOps Lab Exp1 Program c8d987e 1 minute ago 2 commits

Exp1	This is DevOps Lab Exp1 Program	1 minute ago
First.txt	This is First File from Local Repository	22 minutes ago

Help people interested in this repository understand your project by adding a README. Add a README

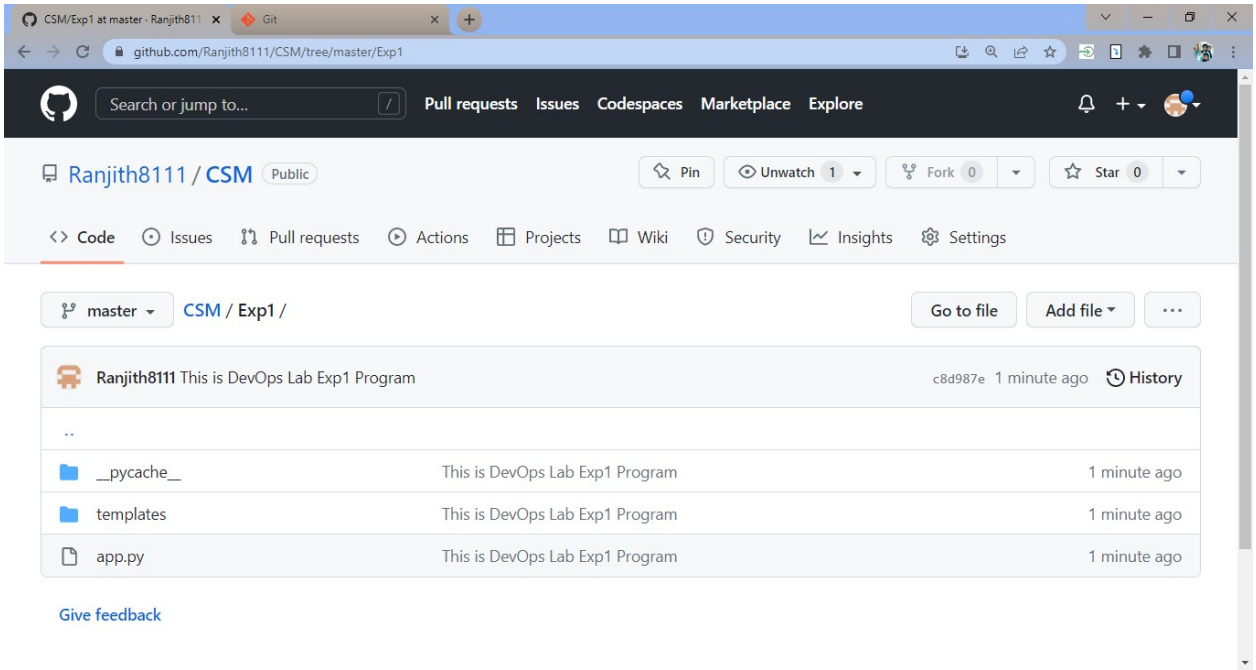
About

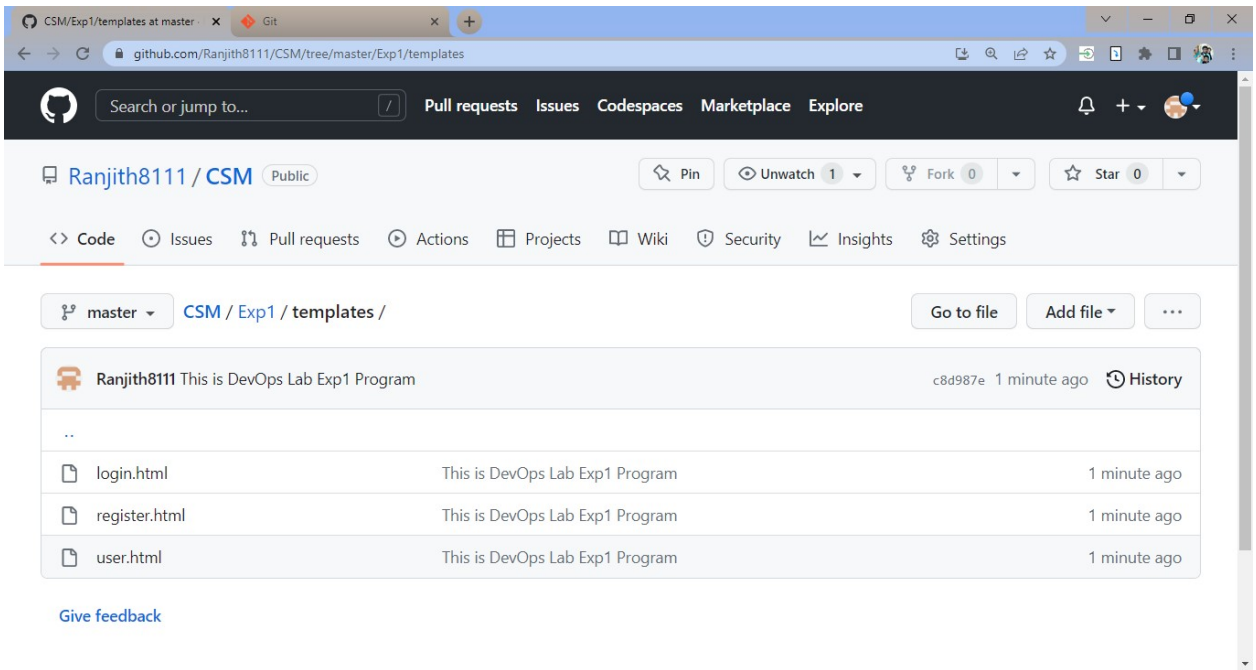
No description, website, or topics provided.

0 stars
1 watching
0 forks

Releases

No releases published
Create a new release


CSM/Exp1 at master · Ranjith811
Git
github.com/Ranjith8111/CSM/tree/master/Exp1
Search or jump to...
Pull requests
Issues
Codespaces
Marketplace
Explore
Ranjith8111 / CSM
Public
Pin
Unwatch 1
Fork 0
Star 0
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
master
CSM / Exp1 /
Go to file
Add file
Ranjith8111 This is DevOps Lab Exp1 Program
c8d987e 1 minute ago
History
..
__pycache__
This is DevOps Lab Exp1 Program
1 minute ago
templates
This is DevOps Lab Exp1 Program
1 minute ago
app.py
This is DevOps Lab Exp1 Program
1 minute ago
Give feedback


CSM/Exp1/templates at master
Git
github.com/Ranjith8111/CSM/tree/master/Exp1/templates
Search or jump to...
Pull requests
Issues
Codespaces
Marketplace
Explore
Ranjith8111 / CSM
Public
Pin
Unwatch 1
Fork 0
Star 0
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
master
CSM / Exp1 / templates /
Go to file
Add file
Ranjith8111 This is DevOps Lab Exp1 Program
c8d987e 1 minute ago
History
..
login.html
This is DevOps Lab Exp1 Program
1 minute ago
register.html
This is DevOps Lab Exp1 Program
1 minute ago
user.html
This is DevOps Lab Exp1 Program
1 minute ago
Give feedback

Experiment 4: Jenkins installation and setup, explore the environment

Prerequisites:

Before you proceed to install Jenkins in your windows system, there are some prerequisites for Jenkins to install Jenkins in your computer.

Hardware requirements:

- You need minimum 256 MB of RAM in your computer or laptop to install Jenkins
- You need at least 1 GB of space in your hard drive for Jenkins.

Software Requirements:

- Since Jenkins runs on Java, you need either latest version of Java Development Kit (JDK)11 above or Java Runtime Environment (JRE).

Release Types

Jenkins releases two types of versions based on the organization needs.

- Long-term support release
- Weekly release

Long term support release (LTS) :

Long-term support releases are available every 12 weeks. They are stable and are widely tested. This release is intended for end users.

Weekly release:

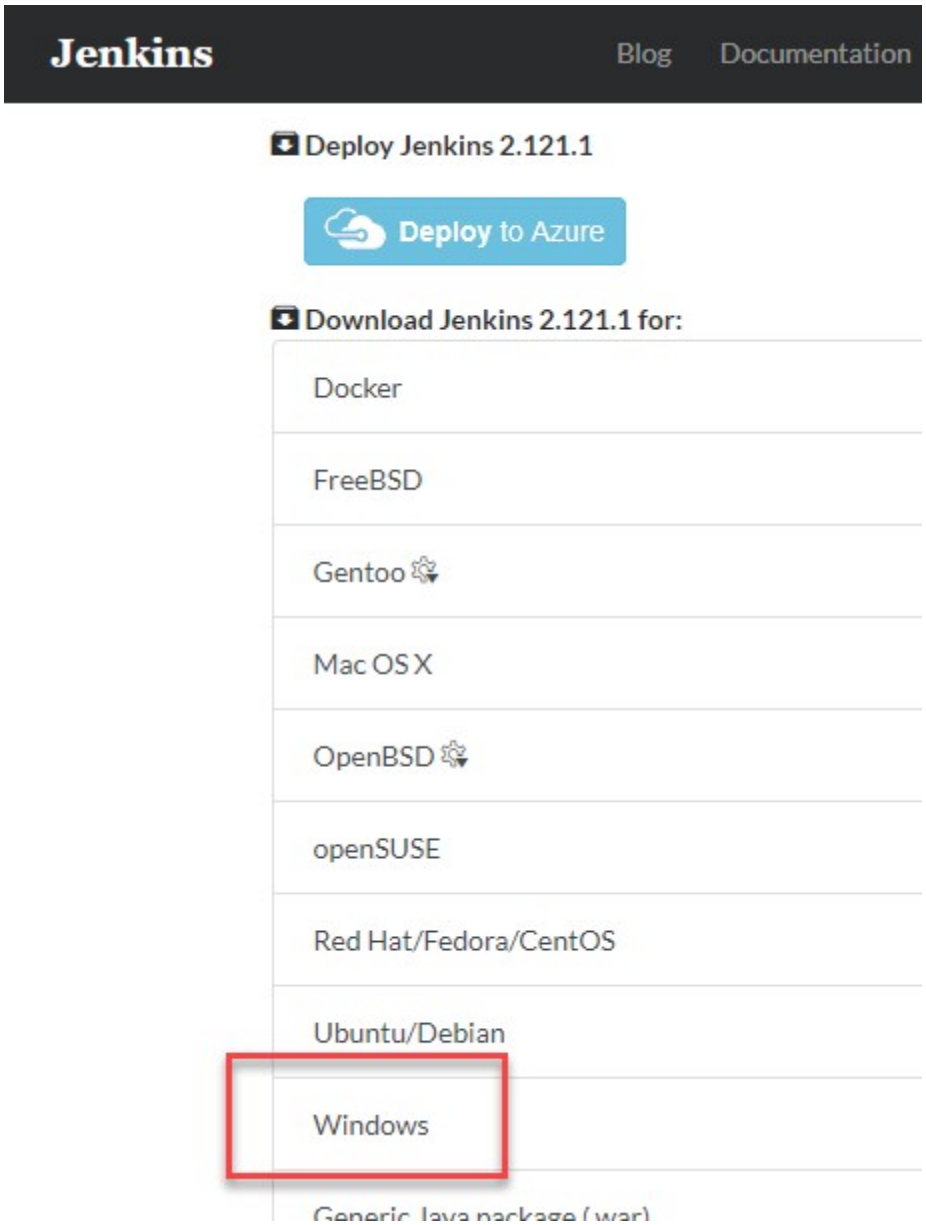
Weekly releases are made available every week by fixing bugs in its earlier version. These releases are intended towards plugin developers.

We will use the LTS release though the process remains the same for Weekly release.

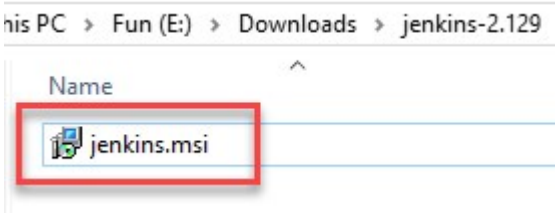
How to Download Jenkins?

Following steps should be followed so that to install Jenkins successfully:

Step 1) Got to https://www.jenkins.io/download/ and select the platform. In our case Windows

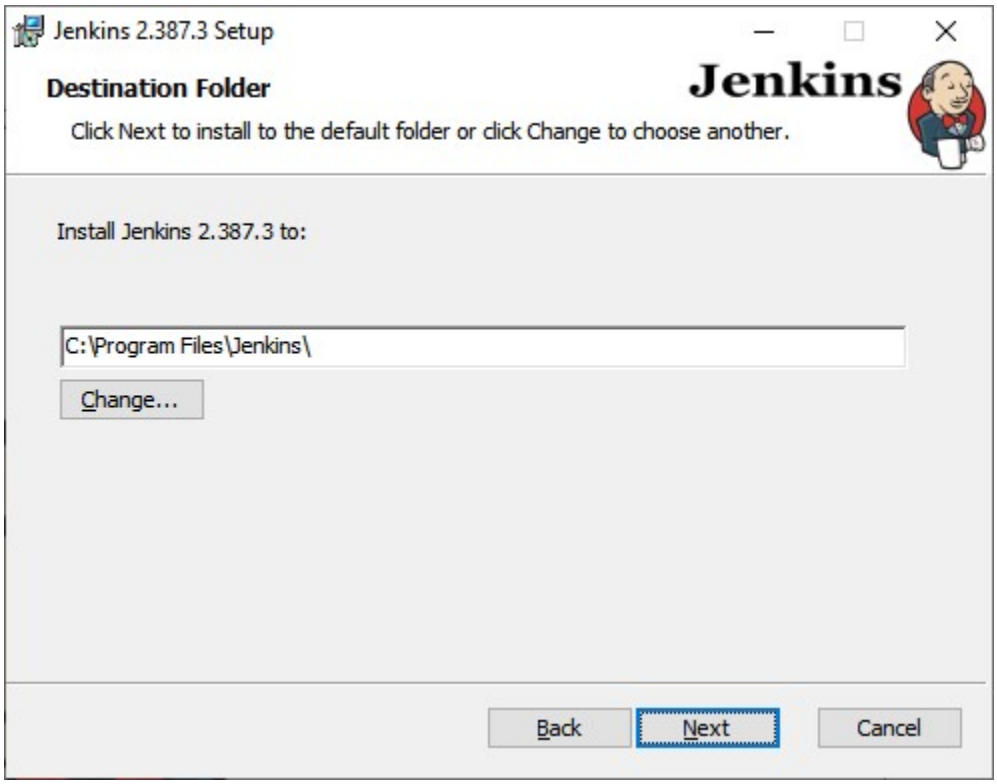


Step 2) Go to download location from local computer and unzip the downloaded package. Double-click on unzipped **jenkins.msi**. You can also Jenkin using a WAR (Web application ARchive) but that is not recommended.

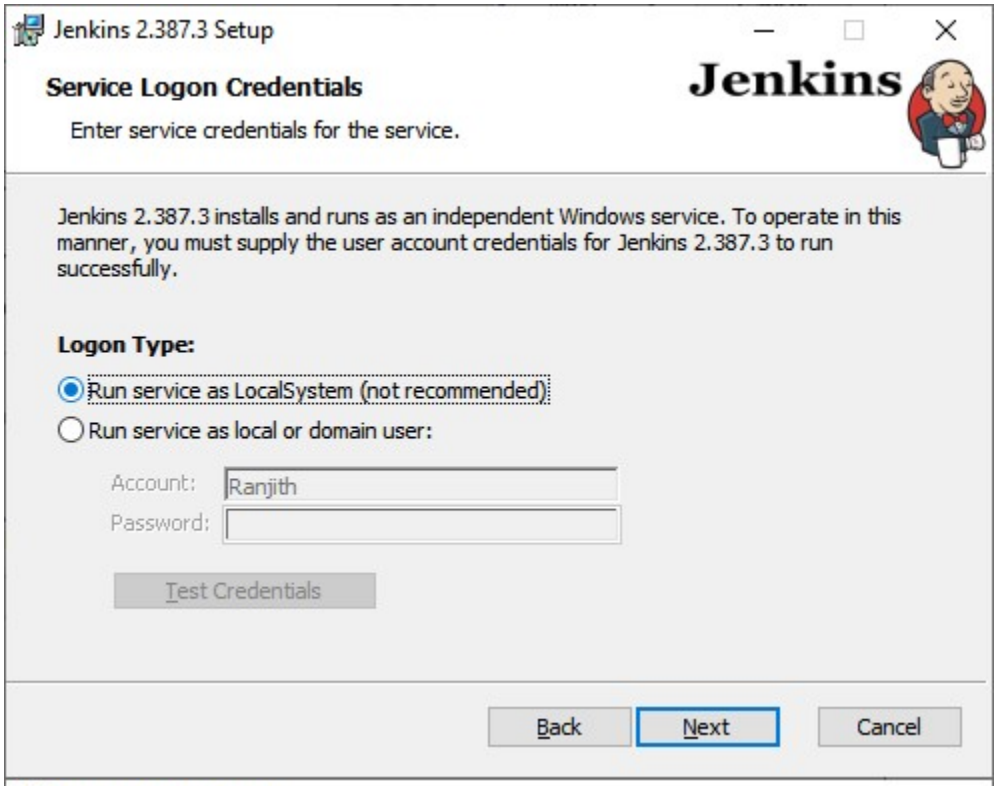


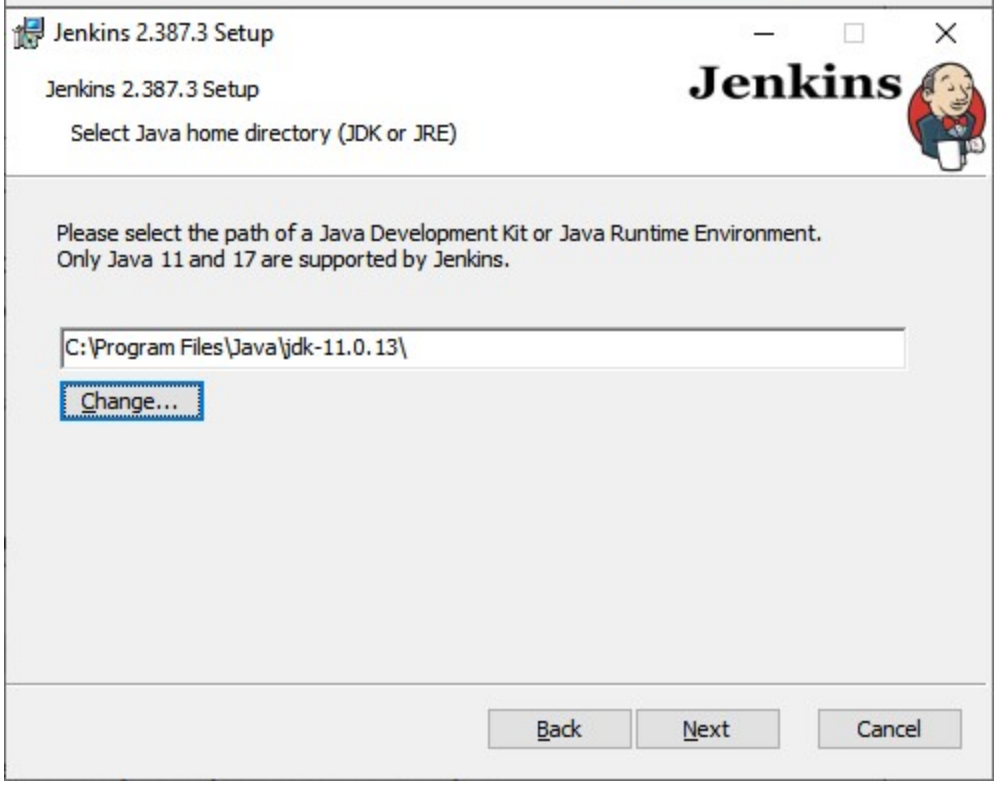
Step 3) In the Jenkin Setup screen, click Next.

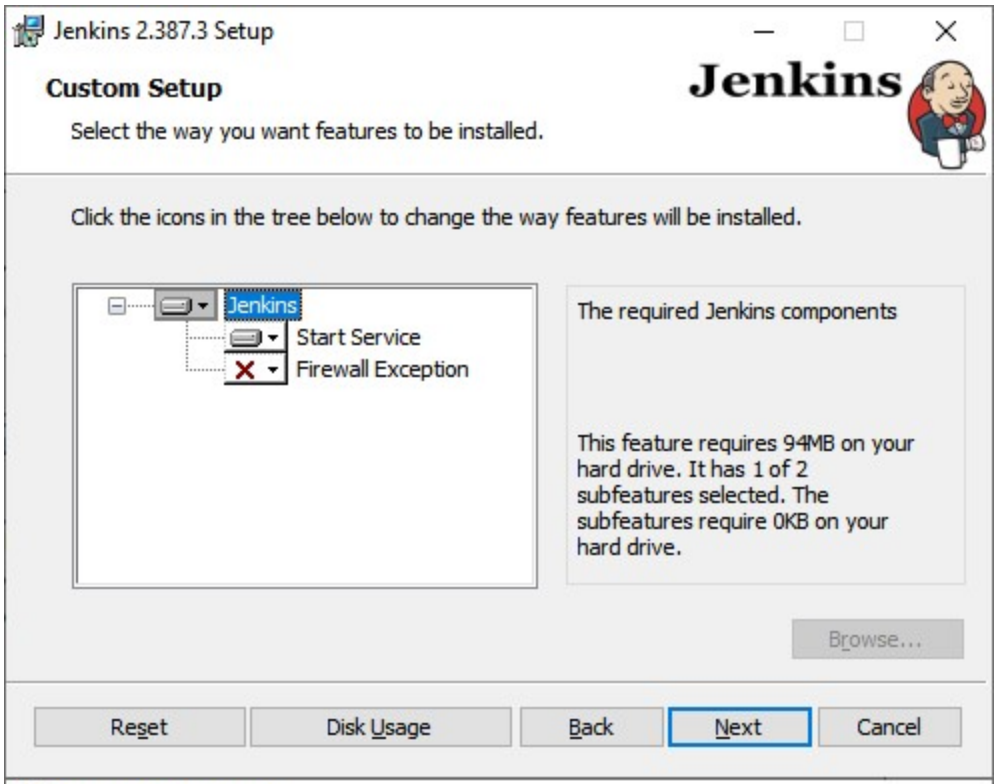
Step 4) Choose the location where you want to have the Jenkins instance installed (default location is C:\Program Files (x86)\Jenkins), then click on **Next** button.

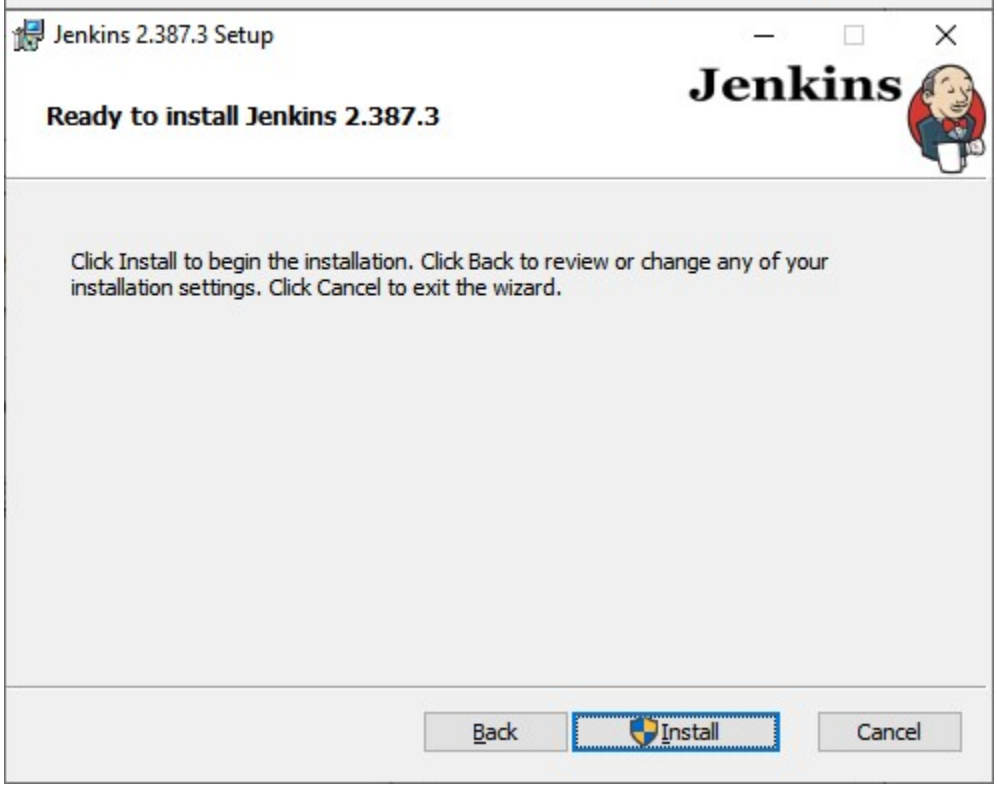


Step 5) Click on the Install button.

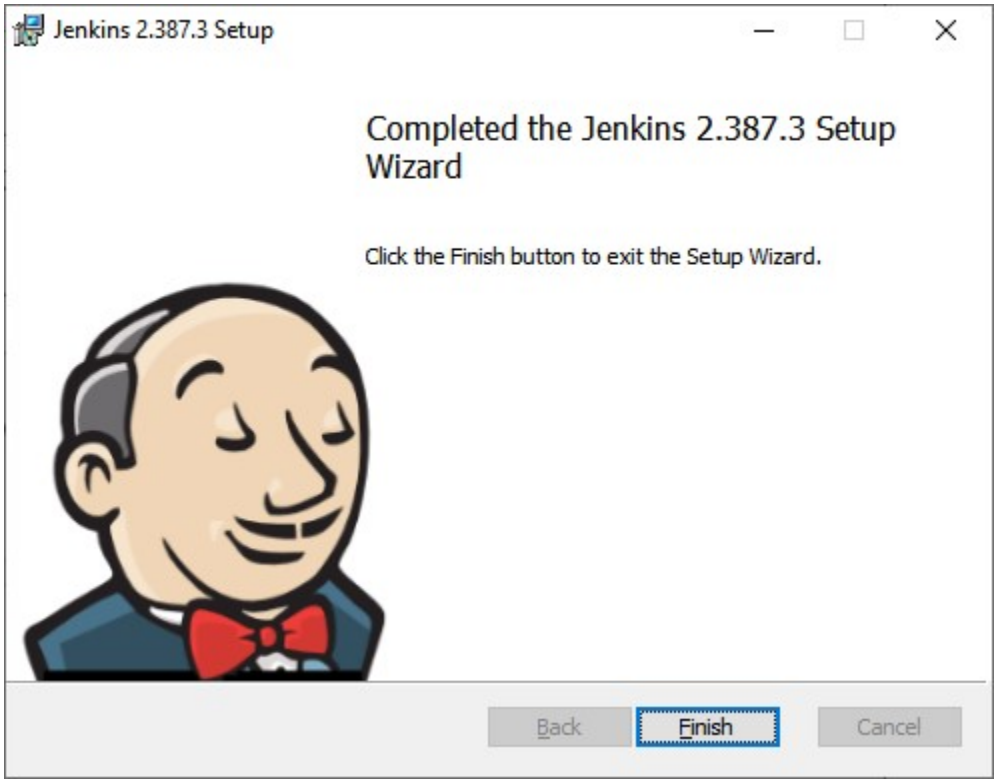




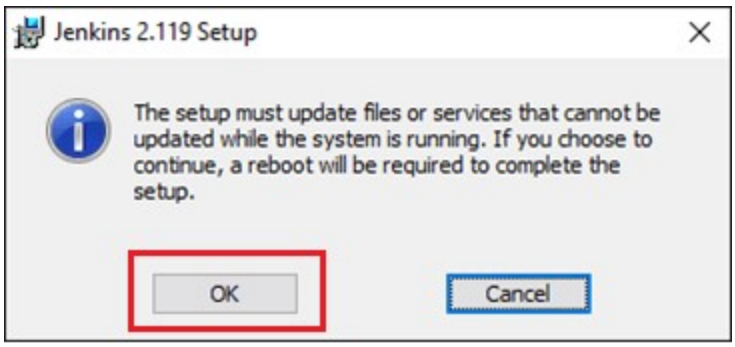

Jenkins 2.387.3 Setup
Custom Setup
Select the way you want features to be installed.
Jenkins
Click the icons in the tree below to change the way features will be installed.
Jenkins
Start Service
Firewall Exception
The required Jenkins components
This feature requires 94MB on your hard drive. It has 1 of 2 subfeatures selected. The subfeatures require 0KB on your hard drive.
Browse...
Reset
Disk Usage
Back
Next
Cancel


Jenkins 2.387.3 Setup
Ready to install Jenkins 2.387.3
Jenkins
Click Install to begin the installation. Click Back to review or change any of your installation settings. Click Cancel to exit the wizard.
Back
Install
Cancel

Step 6) Once install is complete, click Finish.



(Optional) Step 7) During the installation process an info panel may pop-up to inform the user that for a complete setup, the system should be rebooted at the end of the current installation. Click on OK button when the Info panel is popping-up:

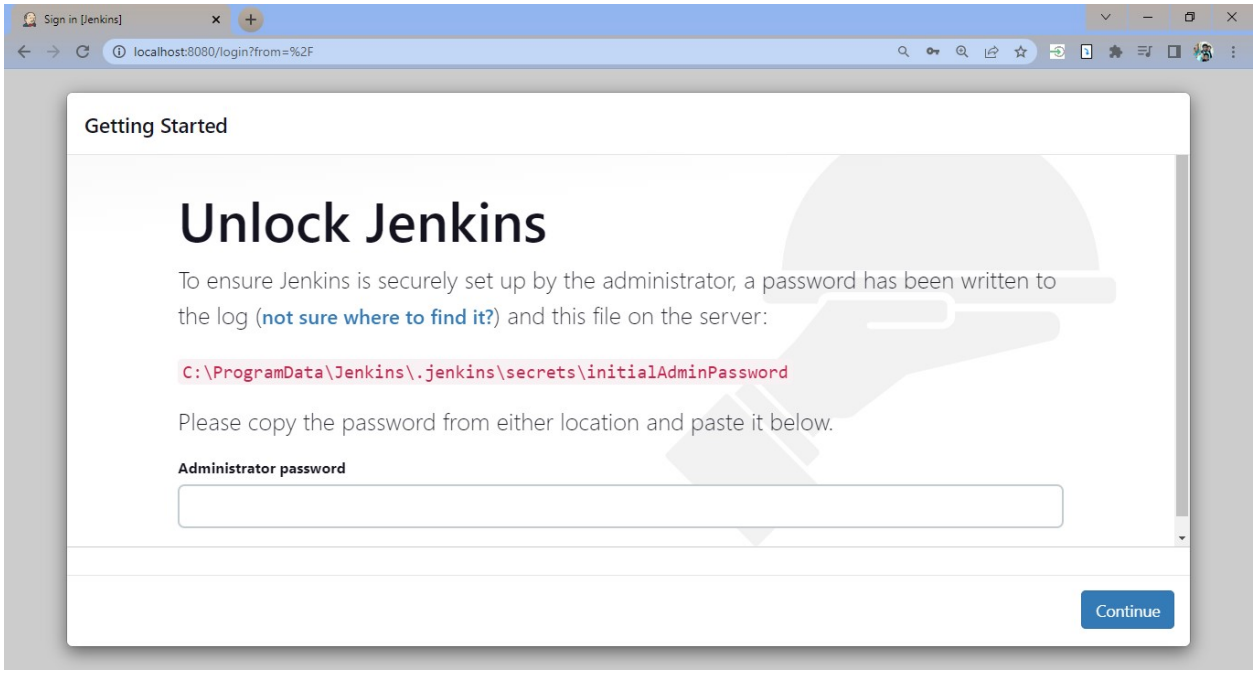


How to Unblock Jenkins?

After completing the Jenkins installation phase, you should proceed further and start its configuration. Next steps will guide you how you can unblock Jenkins application:

Step 1) After completing the Jenkins installation process, a browser tab will pop-up asking for the initial Administrator password. To access Jenkins, you need to go to browse the following path in your web browser. http://localhost:9000

If you can access the above URL, then it confirms that Jenkins is successfully installed in your system.



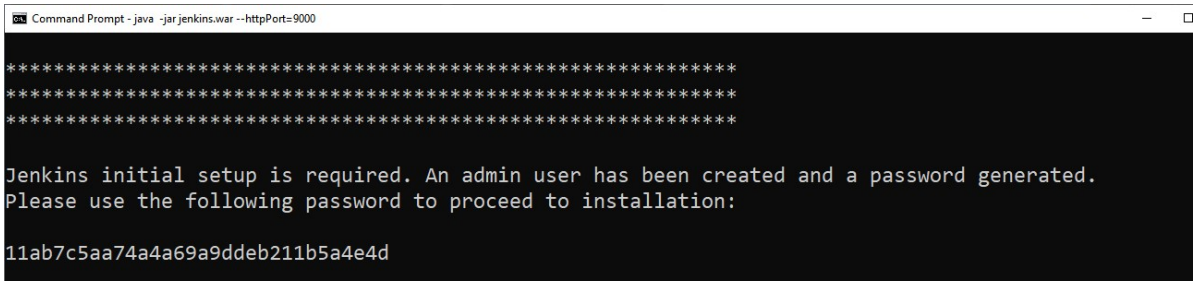
java -jar jenkins.war --httpPort=9000

```
Command Prompt - java -jar jenkins.war --httpPort=9000

C:\Program Files\Jenkins>java -jar jenkins.war --httpPort=9000
Running from: C:\Program Files\Jenkins\Jenkins.war
webroot: C:\Users\Ranjith\.jenkins\war
2023-05-23 06:15:57.174+0000 [id=1]     INFO    winstone.Logger#logInternal: Beginning extraction from
 war file
2023-05-23 06:15:57.252+0000 [id=1]     WARNING o.e.j.s.handler.ContextHandler#setContextPath: Empty c
ontextPath
2023-05-23 06:15:57.388+0000 [id=1]     INFO    org.eclipse.jetty.server.Server#doStart: jetty-10.0.13
; built: 2022-12-07T20:13:20.134Z; git: 1c2636ea05c0ca8de1ffd6ca7f3a98ac084c766d; jvm 11.0.13+10-LTS-3
70
2023-05-23 06:15:58.240+0000 [id=1]     INFO    o.e.j.w.StandardDescriptorProcessor#visitServlet: NO J
SP Support for /, did not find org.eclipse.jetty.jsp.JettyJspServlet
2023-05-23 06:15:58.362+0000 [id=1]     INFO    o.e.j.s.s.DefaultSessionIdManager#doStart: Session wor
kerName=node0
2023-05-23 06:15:59.154+0000 [id=1]     INFO    hudson.WebAppMain#contextInitialized: Jenkins home dir
ectory: C:\Users\Ranjith\.jenkins found at: $user.home/.jenkins
2023-05-23 06:15:59.353+0000 [id=1]     INFO    o.e.j.s.handler.ContextHandler#doStart: Started w.@511
d5d04{Jenkins v2.387.3,/,file:///C:/Users/Ranjith/.jenkins/war/,AVAILABLE}{C:\Users\Ranjith\.jenkins\w
ar}
2023-05-23 06:15:59.400+0000 [id=1]     INFO    o.e.j.server.AbstractConnector#doStart: Started Server
Connector@7494f96a{HTTP/1.1, (http/1.1)}{0.0.0.0:9000}
2023-05-23 06:15:59.429+0000 [id=1]     INFO    org.eclipse.jetty.server.Server#doStart: Started Serve
r@6bf0219d{STARTING}[10.0.13,sto=0] @3158ms
```

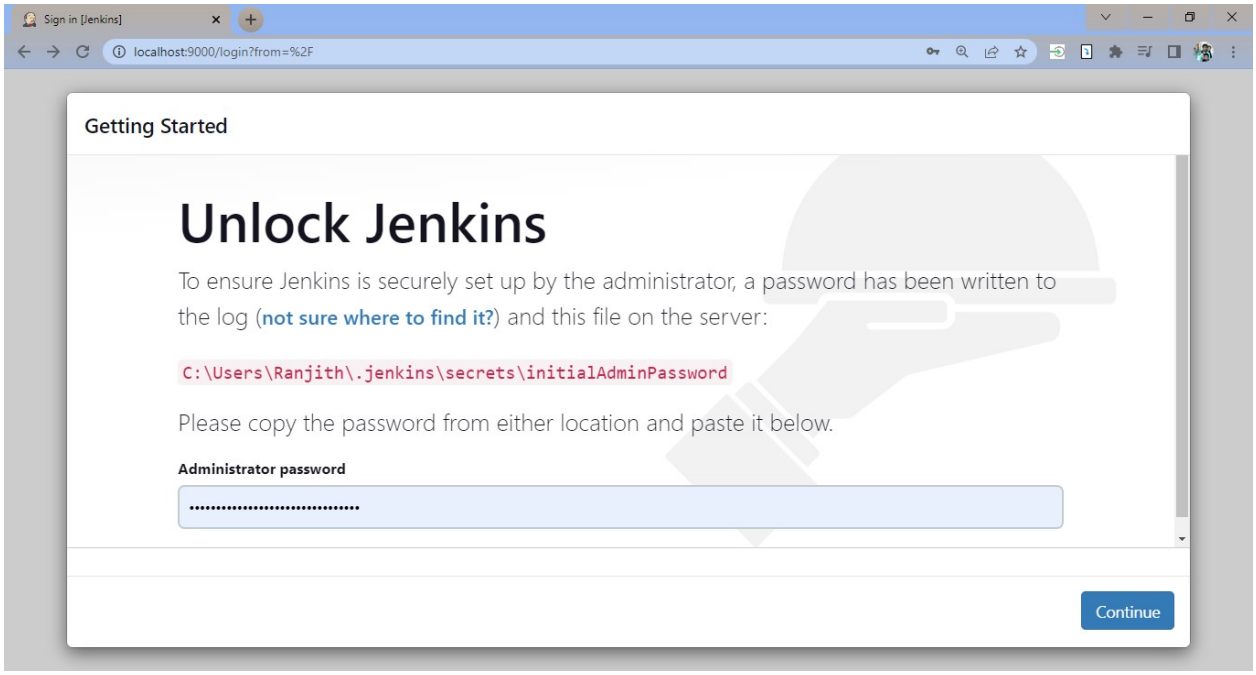
Step 2) The initial Administrator password should be found under the Jenkins installation path (set at Step 4 in Jenkins Installation).

```
Command Prompt - java -jar jenkins.war --httpPort=9000

*************************************************************
*************************************************************
*************************************************************

Jenkins initial setup is required. An admin user has been created and a password generated.
Please use the following password to proceed to installation:

11ab7c5aa74a4a69a9ddeb211b5a4e4d

This may also be found at: C:\Users\Ranjith\.jenkins\secrets\initialAdminPassword

*************************************************************
*************************************************************
*************************************************************

WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.codehaus.groovy.vmplugin.v7.Java7$1 (file:/C:/Users/Ranjith/
.jenkins/war/WEB-INF/lib/groovy-all-2.4.21.jar) to constructor java.lang.invoke.MethodHandles$Lookup(j
ava.lang.Class,int)
WARNING: Please consider reporting this to the maintainers of org.codehaus.groovy.vmplugin.v7.Java7$1
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
```

Step 3) Open the highlighted file and copy the content of the **initialAdminPassword** file.



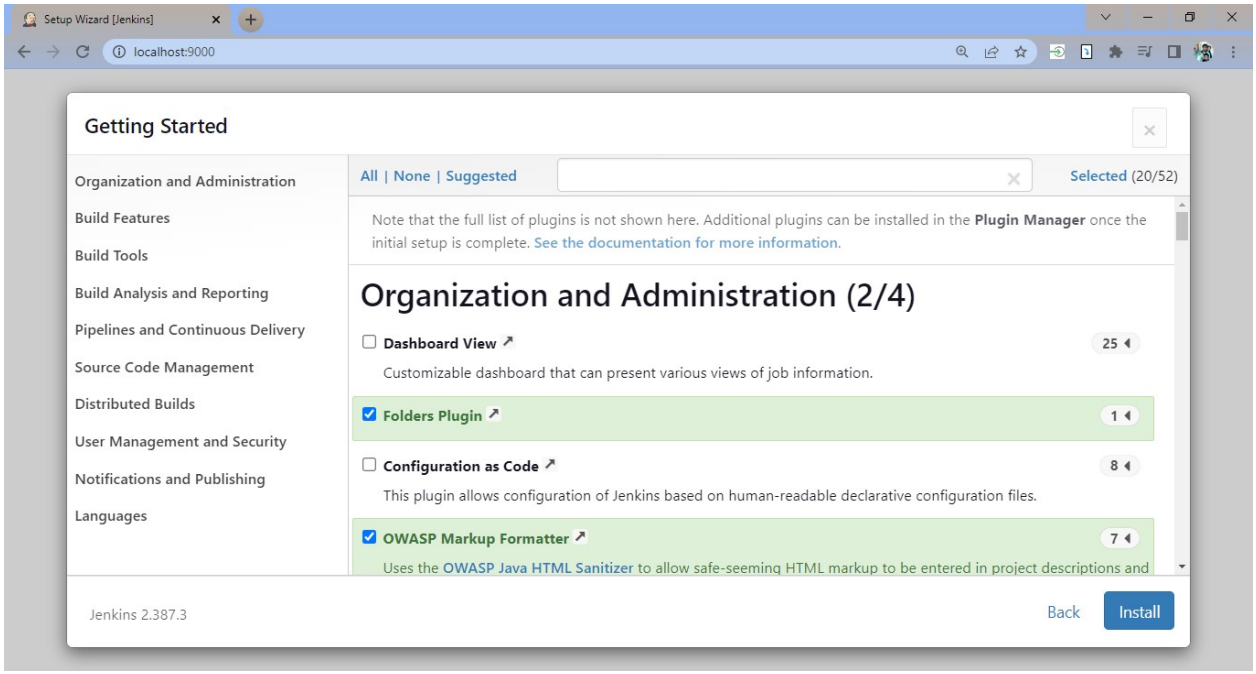
Step 4) Paste the password it into browser's pop-up tab (http://localhost:9000/login?form=%2F) and click on Continue button.

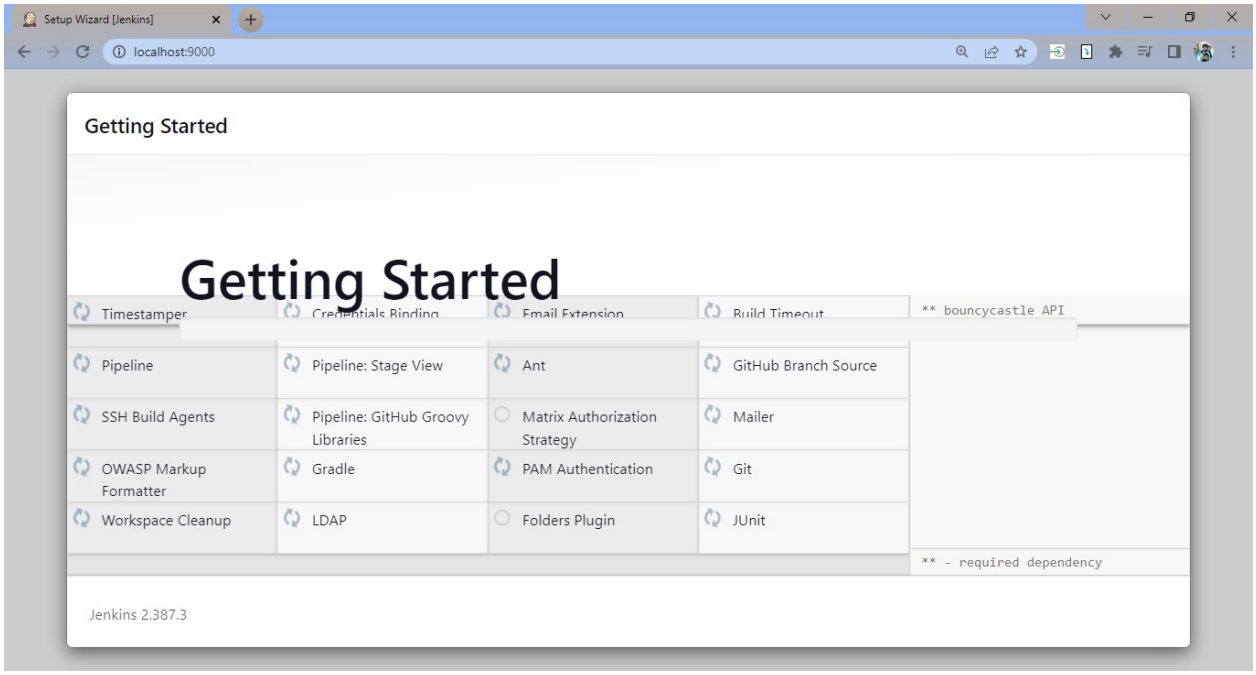


Customize Jenkins

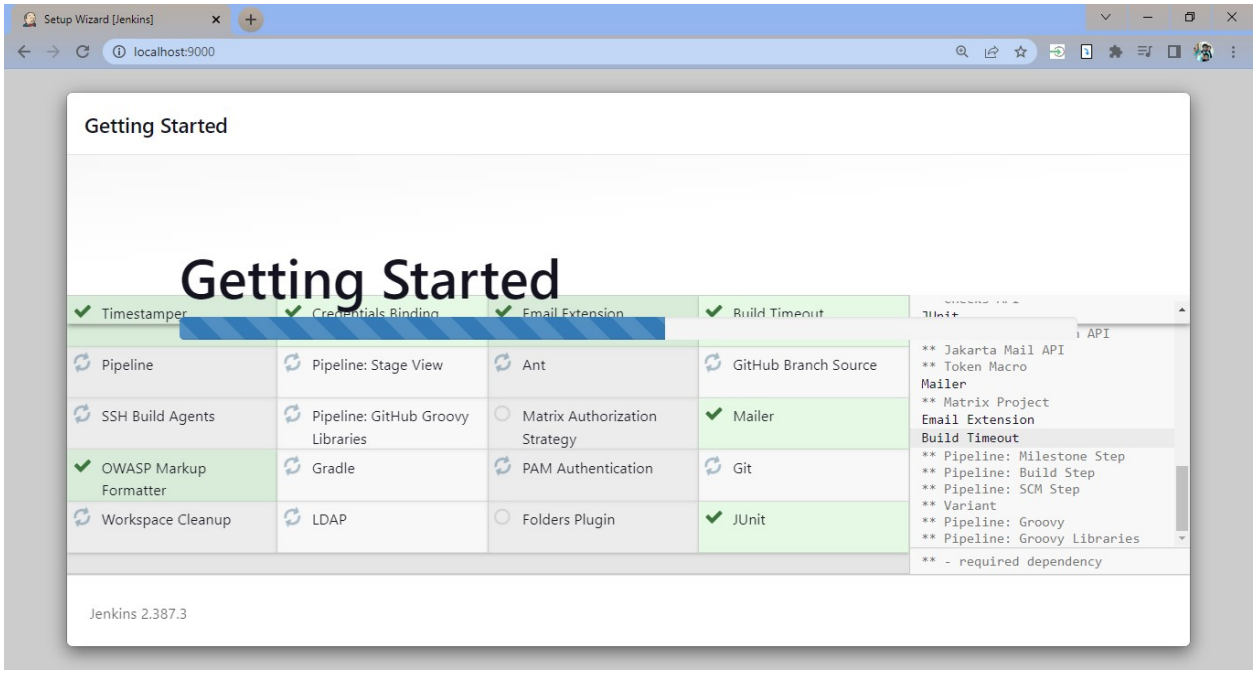
You can also customize your Jenkins environment by below-given steps:

Step 1) Click on the "Install suggested plugins button" so Jenkins will retrieve and install the essential plugins

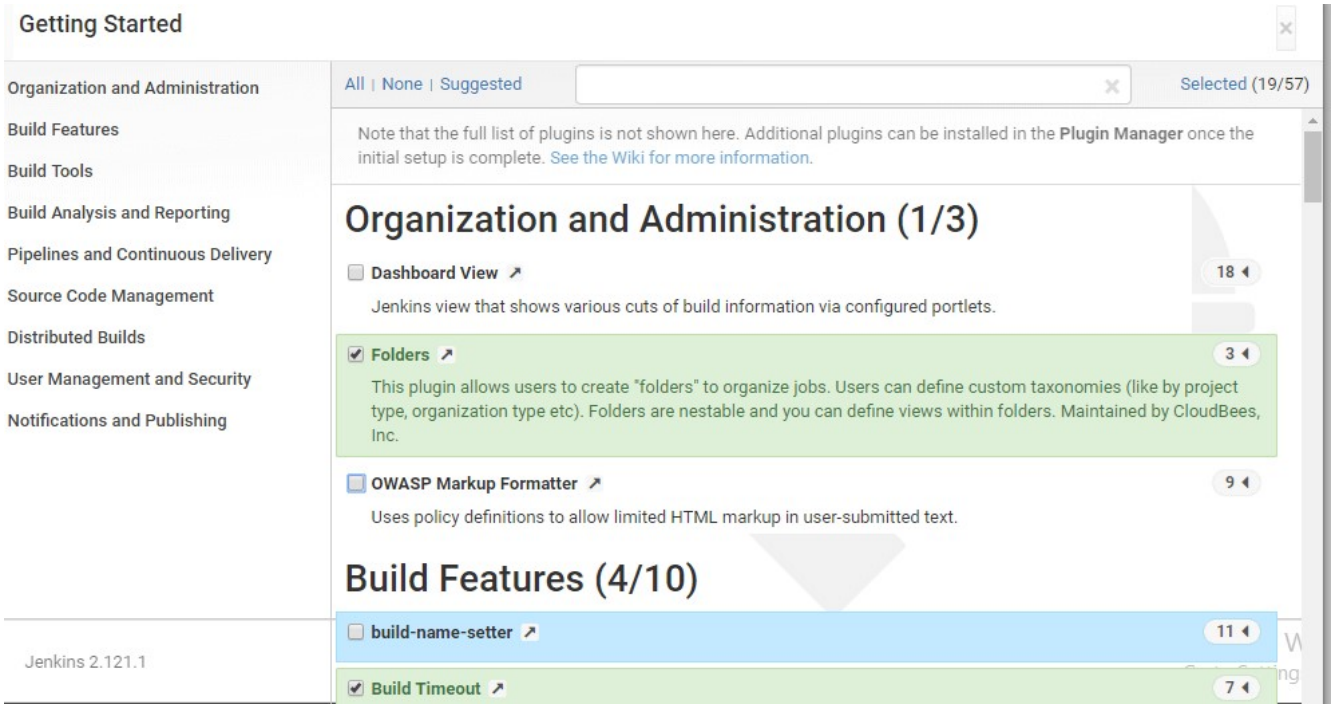




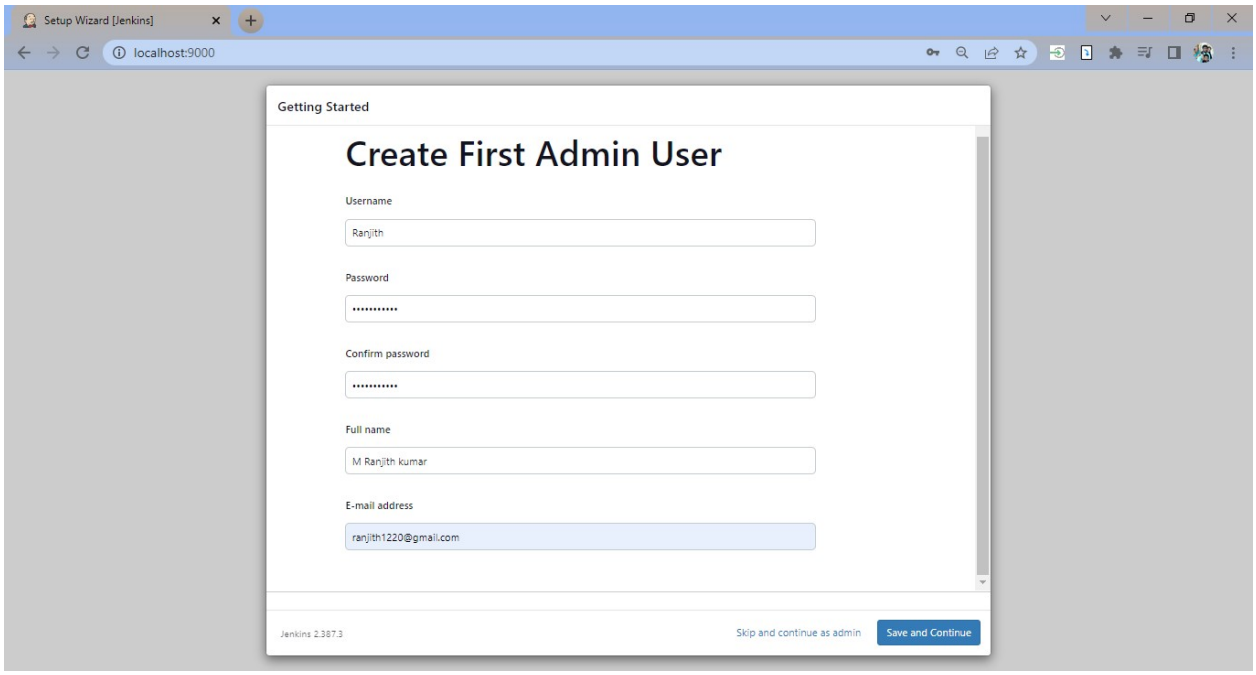
Jenkins will start to download and install all the necessary plugins needed to create new Jenkins Jobs.



Note: You can choose the Option "Select Plugins to Install" and select the plugins you want to install

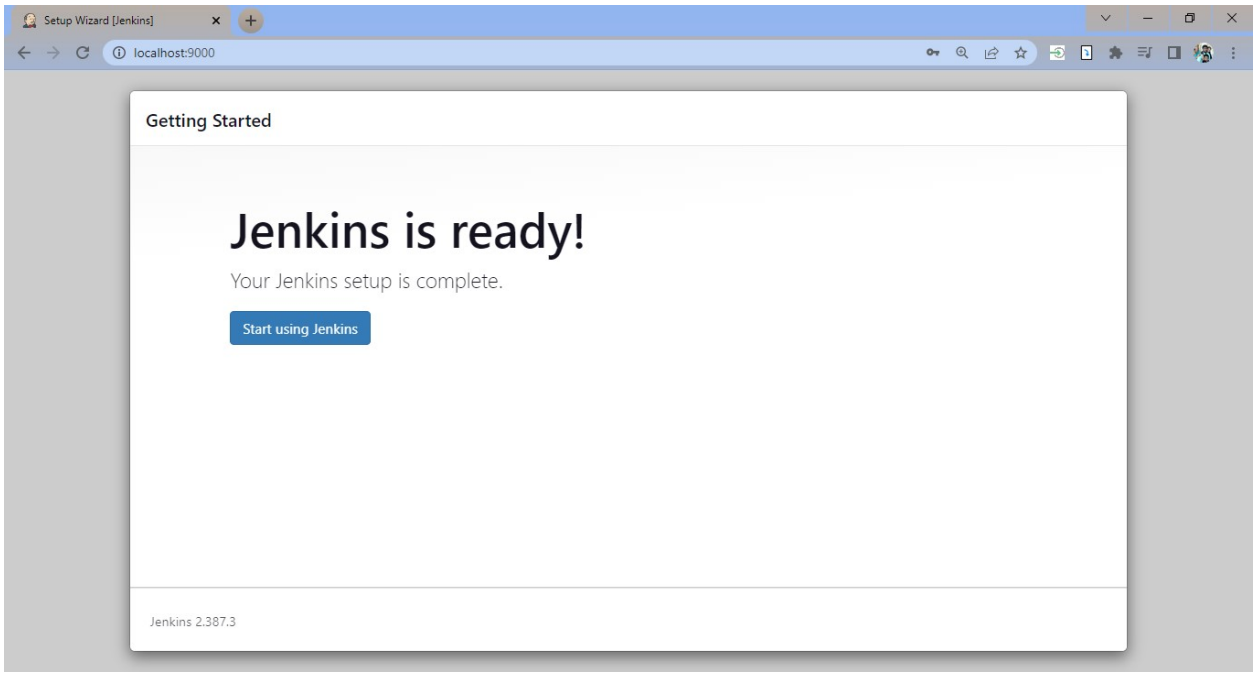


Step 2) After all suggested plugins were installed, the "Create First Admin User" panel will show up. Fill all the fields with desired account details and hit the "**Save and Finish**" button.

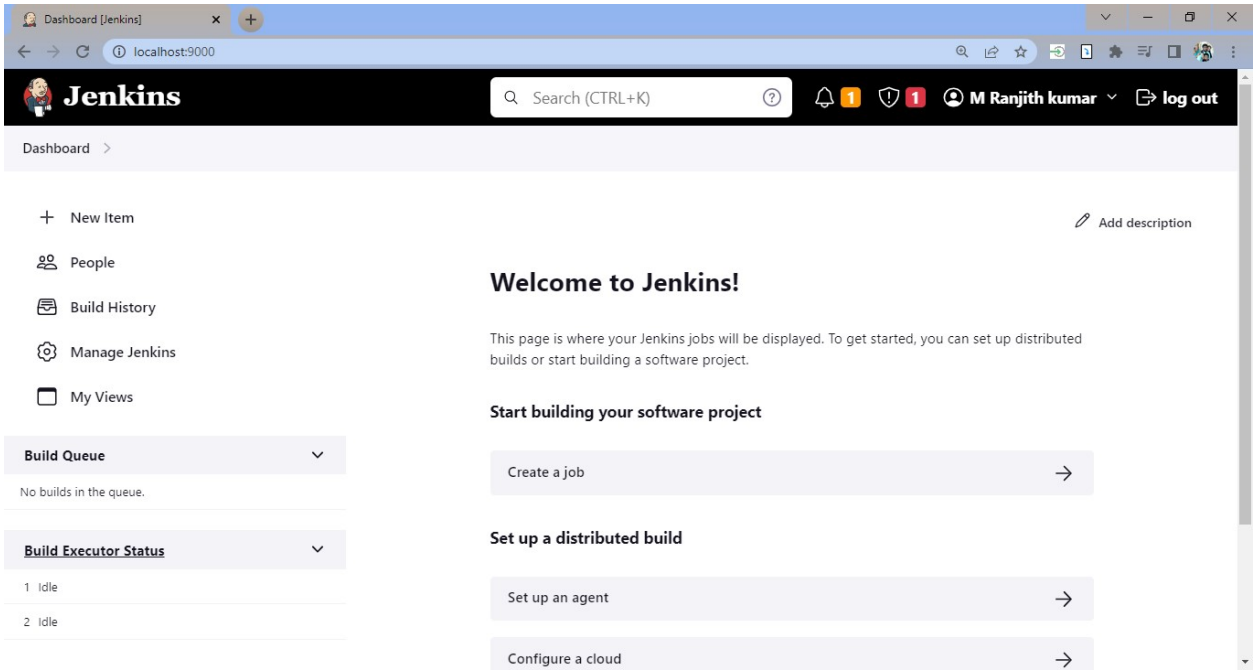


Step 3) Once you have filled the above data, finally it will ask for URL information where you can configure the default instance path for Jenkins. Leave it as it is to avoid any confusions later. However, if another application is already using 8080 port, you can use another port for Jenkins and finally save the settings, and you are done with installation of Jenkins. Hit the "**Save and Continue**" button:

Congratulations! We have successfully installed a new Jenkins Server. Hit the "Start using Jenkins" button.



Below you can find the Jenkins instance up and run, ready to create first Jenkins jobs:



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

Aim: Demonstrate continuous integration and development using Jenkins.

DESCRIPTION

Continuous Integration (CI) and Continuous Development (CD) are important practices in software development that can be achieved using Jenkins. Here's an example of how you can demonstrate CI/CD using Jenkins:

Create a simple Java application:

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

Commit the code to a Git repository:

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

Create a Jenkins job:

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

Build the application:

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

Monitor the build:

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

Deploy the application:

- If the build is successful, configure the Jenkins job to deploy the application to a production environment.
- The deployment could be as simple as copying the jar file to a

production server or using a more sophisticated deployment process, such as using a containerization technology like Docker.

Repeat the process:

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

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

DEVOPS LAB MANUAL

EXPERIMENT NO.: 6. Explore Docker commands for contentmanagement.

AIM: Explore Docker commands for content management.

DESCRIPTION

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

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

For example: $ docker run --name mycontainer -it ubuntu:16.04 /bin/bash

This command runs a new container based on the Ubuntu 16.04image and starts a shell session in the container.

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

For example: $ docker start mycontainer

This command starts the container named "mycontainer".

- **Docker stop:** Stop one or more running containers.
- For example: $ docker stop mycontainer

This command stops the container named "mycontainer".

- **Docker rm:** Remove one or more containers.
- For example: $ docker rm mycontainer

This command removes the container named "mycontainer".

- **Docker ps:** List containers.

For example: $ docker ps

This command lists all running containers.

- **Docker images:** List images.

For example: $ docker images

This command lists all images stored locally on the host.

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

For example: $ docker pull ubuntu:16.04

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

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

For example: $ docker push myimage

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

- **Build the Docker image:**

Run the following command to build the Docker image:

$ docker build -t myimage .

This command builds a new Docker image using the Dockerfile and tags

These are some of the basic Docker commands for managing containers andimages. There are many other Docker commands and options that you can use for more advanced use cases, such as managing networks, volumes, andconfiguration.

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

AIM: Develop a simple containerized application using Docker DESCRIPTION

Here's an example of how you can develop a simple containerized application using Docker in the following 5 steps:

Step1: **Choose an application**

Step2: **Write a Dockerfile**

Step3: **Build the Docker image**

Step4: **Run the Docker container**

Step5: **Verify the output**

Choose an application:

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

Write a Dockerfile:

- Create a file named "Dockerfile" in the same directory as the application.

In the Dockerfile, specify the base image, copy the application into the container, and specify the command to run the application. Here's an example Dockerfile for a Python script:

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

# Copy the Python script into the container
 COPY hello.py /app/

# Set the working directory to /app/ WORKDIR /app/
# Run the Python script when the container starts
CMD ["python", "hello.py"]
```

Build the Docker image:

Run the following command to build the Docker image:

```
C:\Users\Raziya>docker build -t myimage .
```

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

Run the Docker container:

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

```
C:\Users\Raziya>docker run --name mycontainer myimage
```

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

Verify the output:

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

```
C:\Users\Raziya>docker logs mycontainer
```

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

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

EXPERIMENT NO.: 8. Integrate Kubernetes and Docker

AIM: Integrate Kubernetes and Docker

DESCRIPTION:

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

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

Here's a high-level overview of the steps to integrate Kubernetes and Docker:

Build a Docker image:

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

- Push the Docker image to a registry:

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

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

Monitor and manage the containers:

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

- Continuously integrate and deploy changes:

Use a continuous integration and deployment (CI/CD) pipeline to automatically build, push, and deploy changes to the Docker image and the Kubernetes cluster.

This makes it easier to make updates to the applicationand ensures that the latest version is always running in the cluster.

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

ERIMENT NO.: 9. Automate the process of running containerizedapplication developed in exercise 7 using Kubernetes

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

DESCRIPTION

To automate the process of running the containerized application developedin exercise 7 using Kubernetes, you can follow these steps:

- Create a Kubernetes cluster:

Create a Kubernetes cluster using a cloud provider, such as Google Cloud or Amazon Web Services, or using a local installation of Minikube.

- Push the Docker image to a registry:

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

- Create a deployment:

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

```
apiVersion: apps/v1
kind: Deployment
metadata:
name: myappspec:
  replicas: 3
  selector:
    matchLabels:
      app: myapptemplate:
      metadata:
      labels:
        app: myappspec:
      containers:
      - name: myapp
      image: myimage
      ports:
        - containerPort: 80
```

- Create a service:

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

```
apiVersion: v1kind: Service metadata:
  name: myapp-service
spec:
  selector:
    app: myappports:
  - name: http
    port: 80
    targetPort: 80
    type: ClusterIP
```

- Apply the deployment and service to the cluster:

Apply the deployment and service to the cluster using the kubectl command-line tool. For example:

```
$ kubectl apply -f deployment.yaml
$ kubectl apply -f service.yaml
```

- Verify the deployment:

Verify the deployment by checking the status of the pods and the service. For example:

```
$ kubectl get pods
$ kubectl get services
```

This is a basic example of how to automate the process of running a containerized application using Kubernetes. In a real-world scenario, youwould likely have more complex requirements, such as managing persistent data, scaling, and rolling updates, but this example should give you a good starting point for using Kubernetes to manage your containers.

Steps to work with selenium

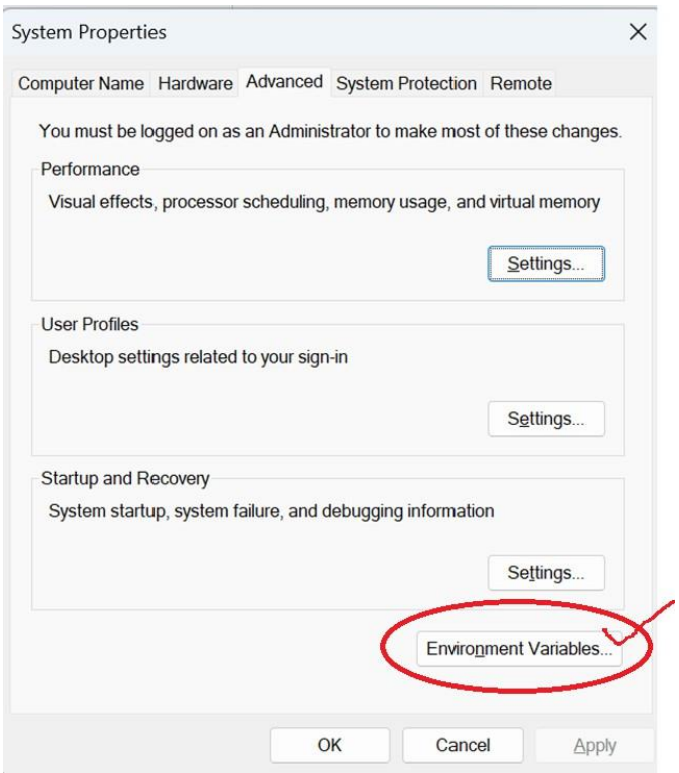
Step 1:

Download and Install Java 17(recommended) from the oracle website

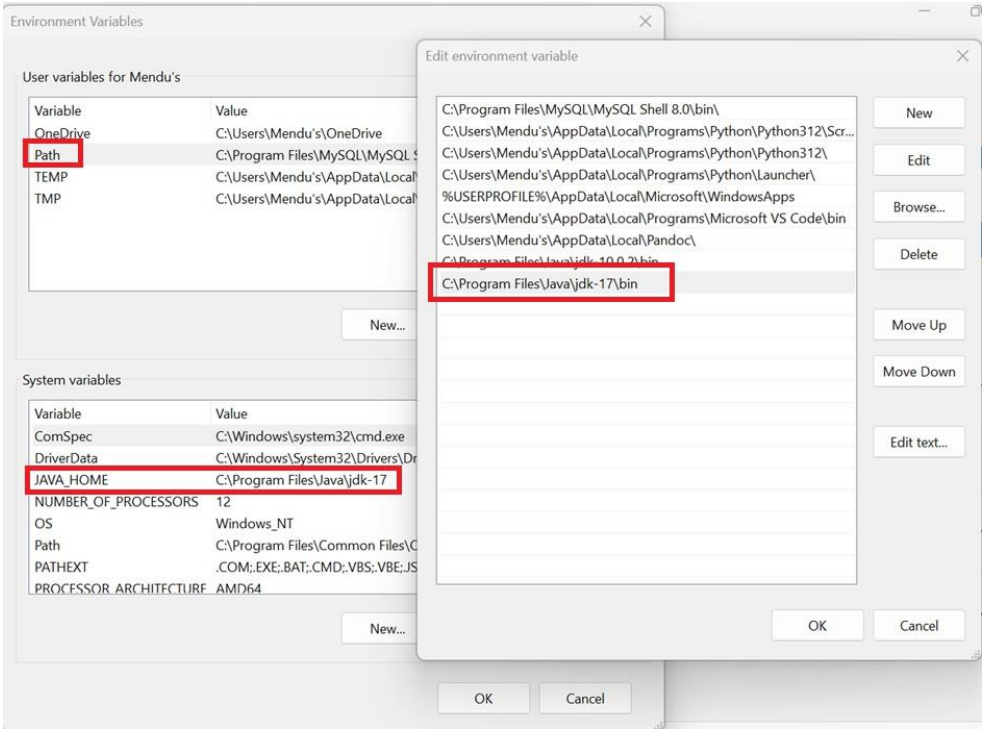
https://download.oracle.com/java/17/archive/jdk-17.0.10_windows-x64_bin.msi

2.Set path for java in environment variables

Goto→start and type edit environment variables and the below window will be opened for you



Click on Environment variables and select the path variable and edit it as shown below also set the JAVA_HOME variable.



Step 2:

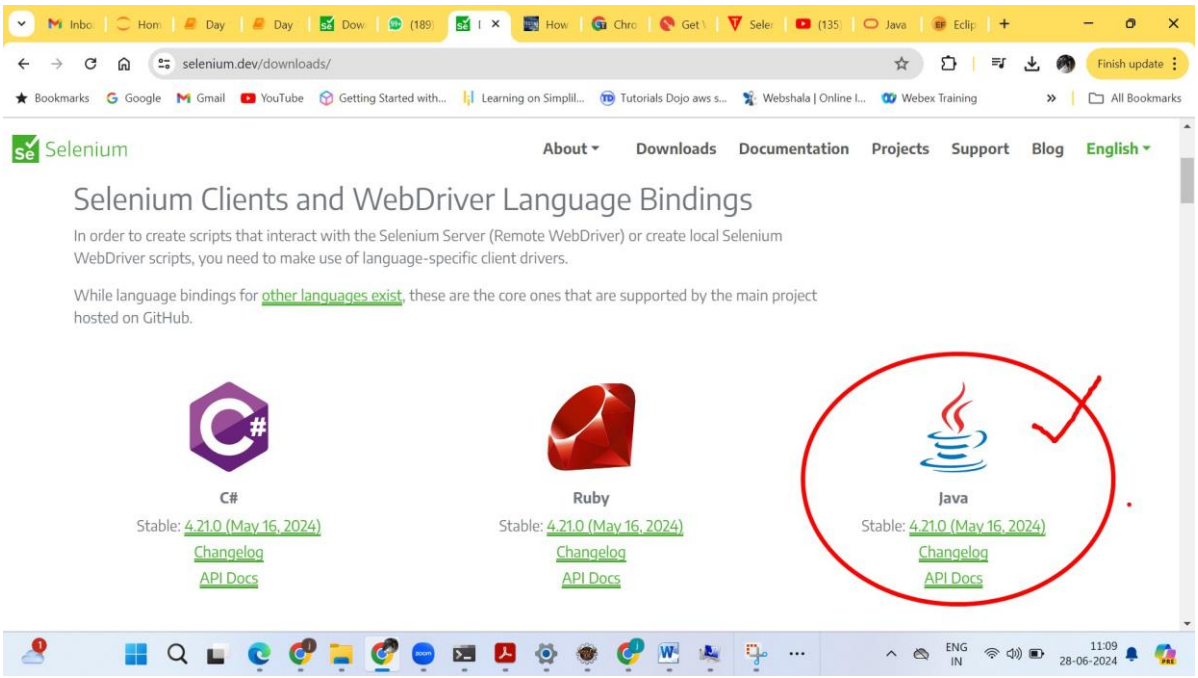
Download and Install Eclipse IDE

https://www.eclipse.org/downloads/download.php?file=/oomph/epp/2024-06/R/eclipse-inst-jre-win64.exe

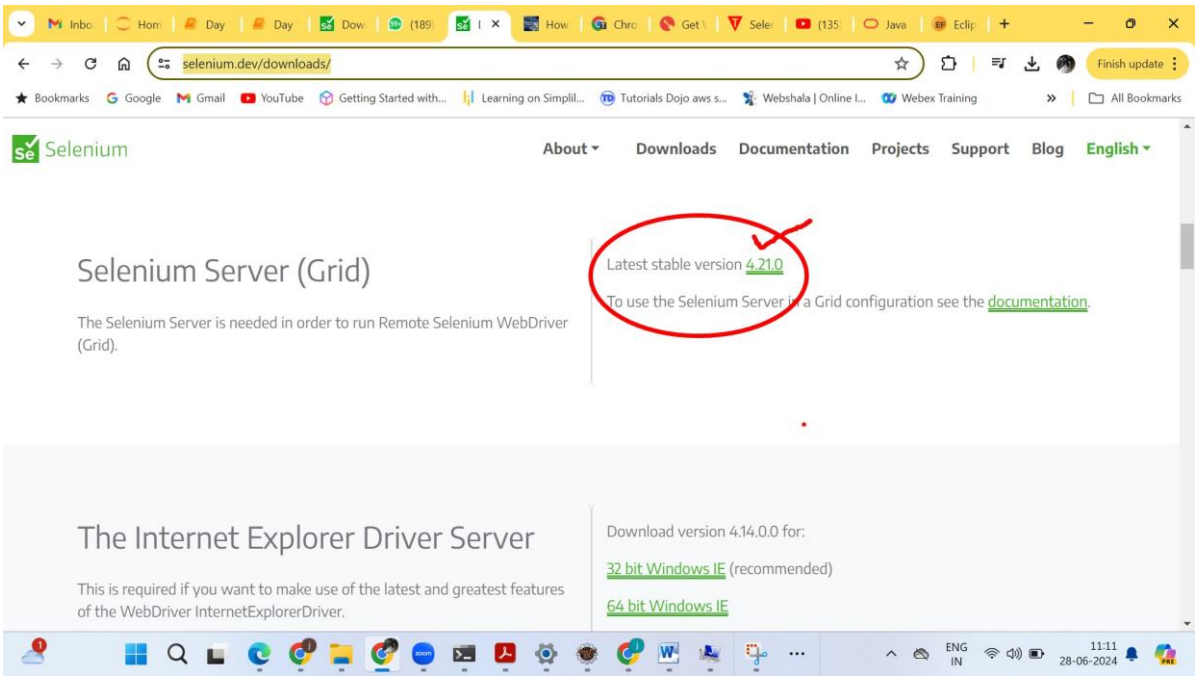
Step 3:

Download Selenium Web Driver for Java

https://www.selenium.dev/downloads/

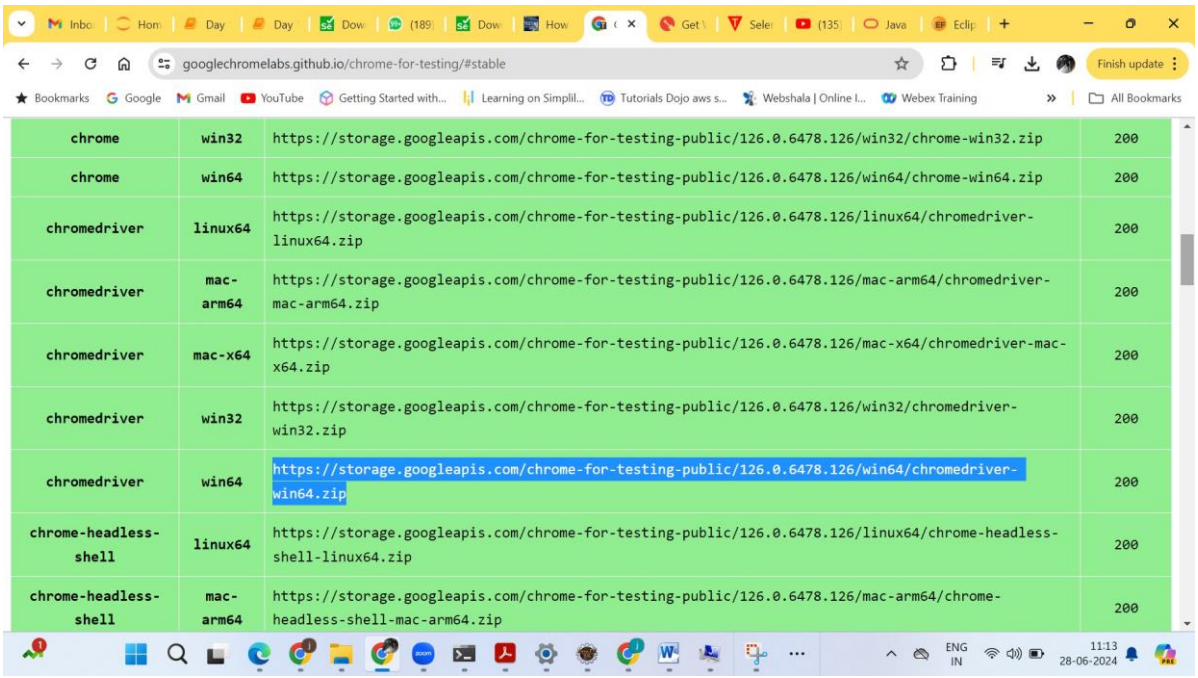


Also download Selenium Server as show below from the same link



Download Chrome Driver from the below link

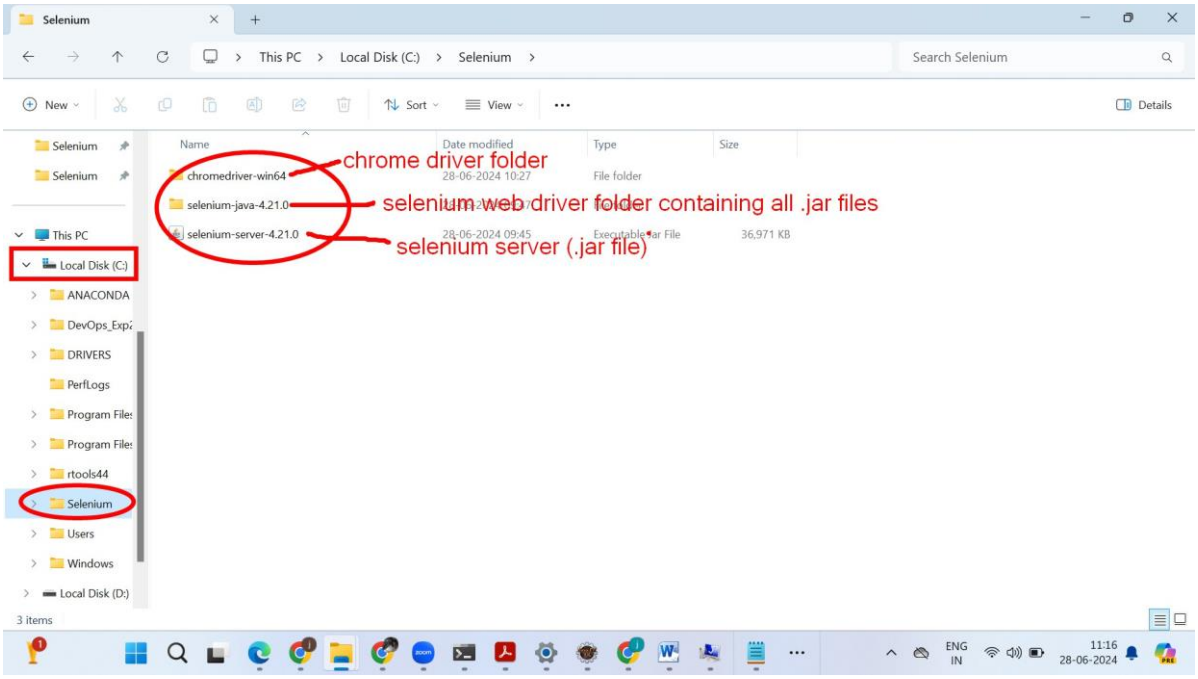
https://storage.googleapis.com/chrome-for-testing-public/126.0.6478.126/win64/chromedriver-win64.zip



chrome	win32	https://storage.googleapis.com/chrome-for-testing-public/126.0.6478.126/win32/chrome-win32.zip	200
chrome	win64	https://storage.googleapis.com/chrome-for-testing-public/126.0.6478.126/win64/chrome-win64.zip	200
chromedriver	linux64	https://storage.googleapis.com/chrome-for-testing-public/126.0.6478.126/linux64/chromedriver-linux64.zip	200
chromedriver	mac-arm64	https://storage.googleapis.com/chrome-for-testing-public/126.0.6478.126/mac-arm64/chromedriver-mac-arm64.zip	200
chromedriver	mac-x64	https://storage.googleapis.com/chrome-for-testing-public/126.0.6478.126/mac-x64/chromedriver-mac-x64.zip	200
chromedriver	win32	https://storage.googleapis.com/chrome-for-testing-public/126.0.6478.126/win32/chromedriver-win32.zip	200
chromedriver	win64	https://storage.googleapis.com/chrome-for-testing-public/126.0.6478.126/win64/chromedriver-win64.zip	200
chrome-headless-shell	linux64	https://storage.googleapis.com/chrome-for-testing-public/126.0.6478.126/linux64/chrome-headless-shell-linux64.zip	200
chrome-headless-shell	mac-arm64	https://storage.googleapis.com/chrome-for-testing-public/126.0.6478.126/mac-arm64/chrome-headless-shell-mac-arm64.zip	200

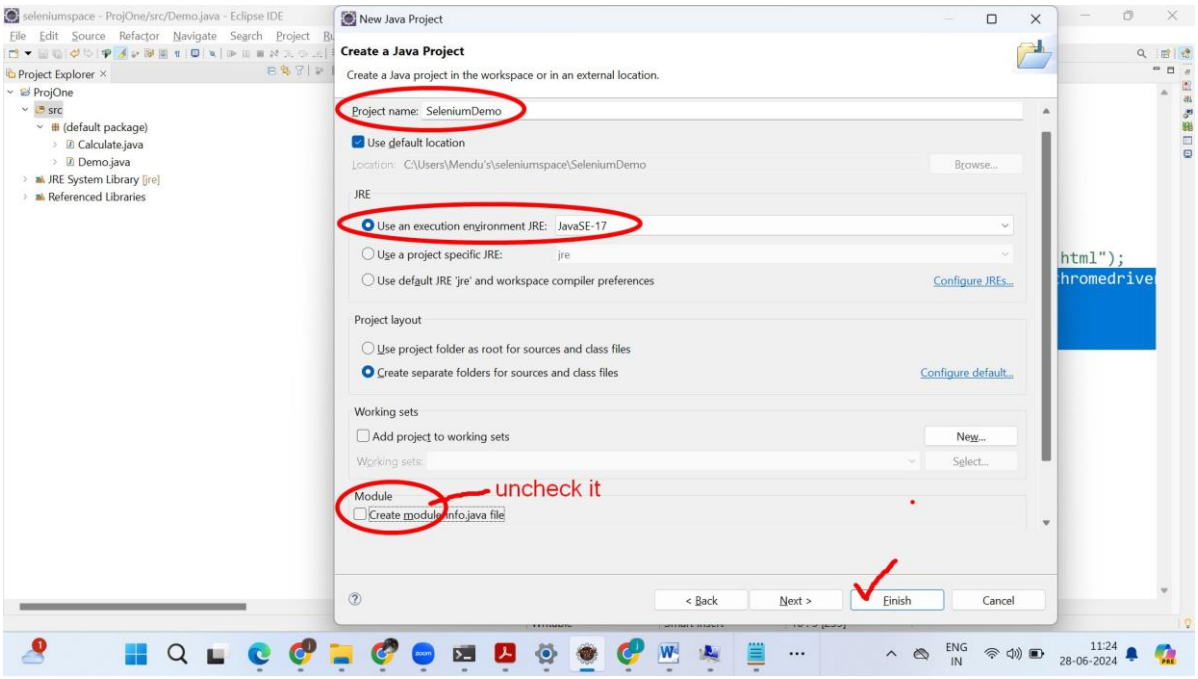
Step 4:

Create a folder Selenium in the c:\ drive and unzip all the above softwares and paste into it as shown below

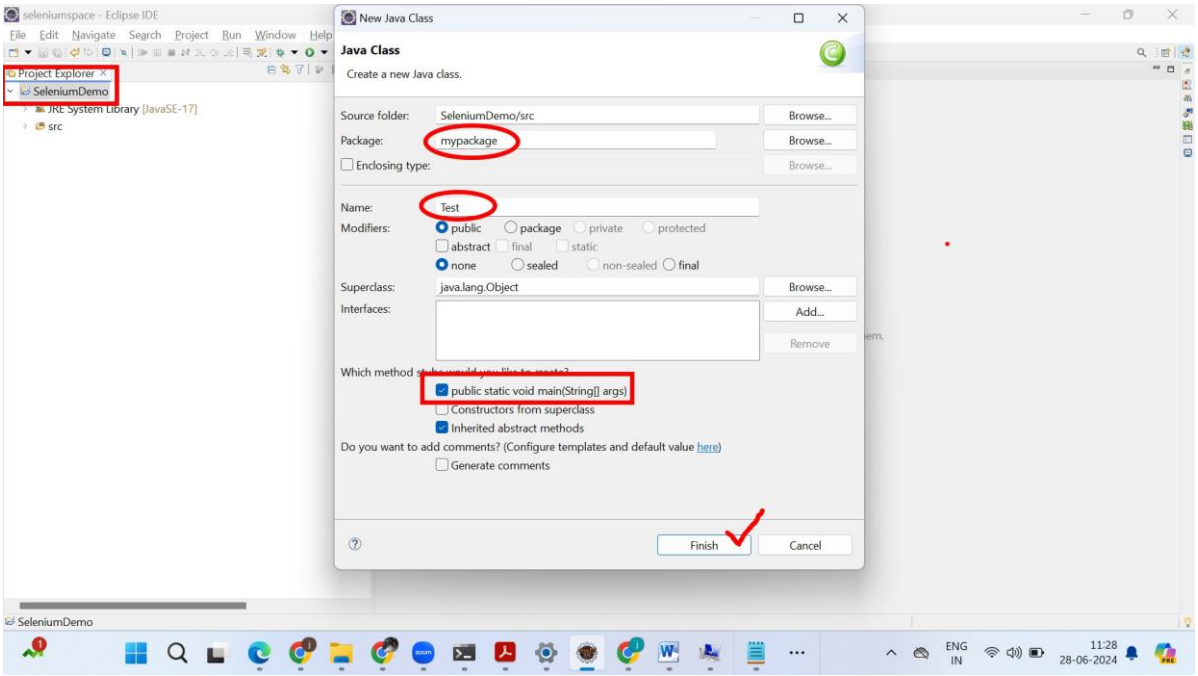


Step 5:

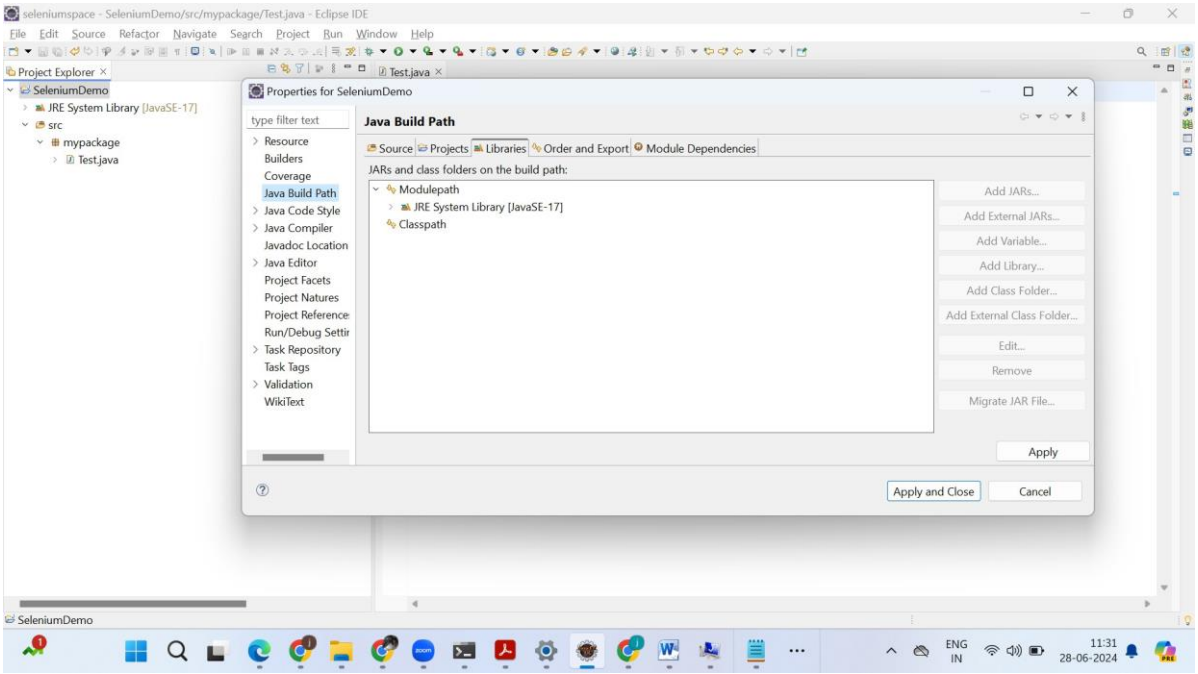
Now Open Eclipse and Take a Java Project SeleniumDemo as shown below



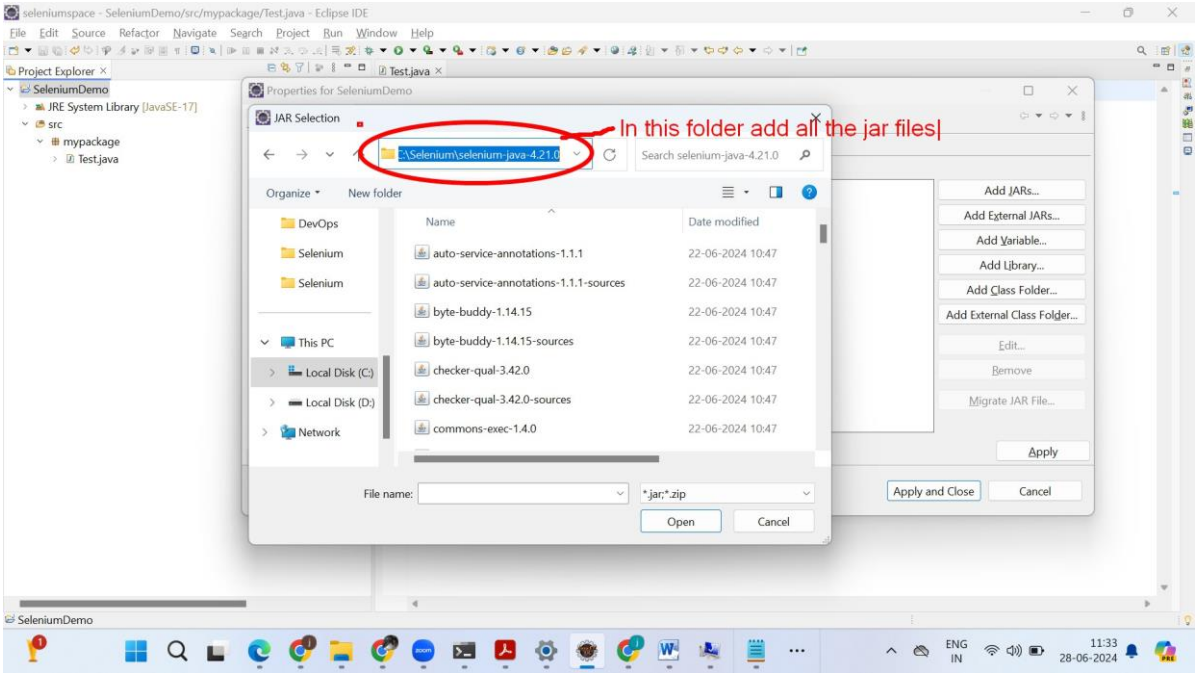
Add a class Test.java to the project on right clicking it in the project explorer



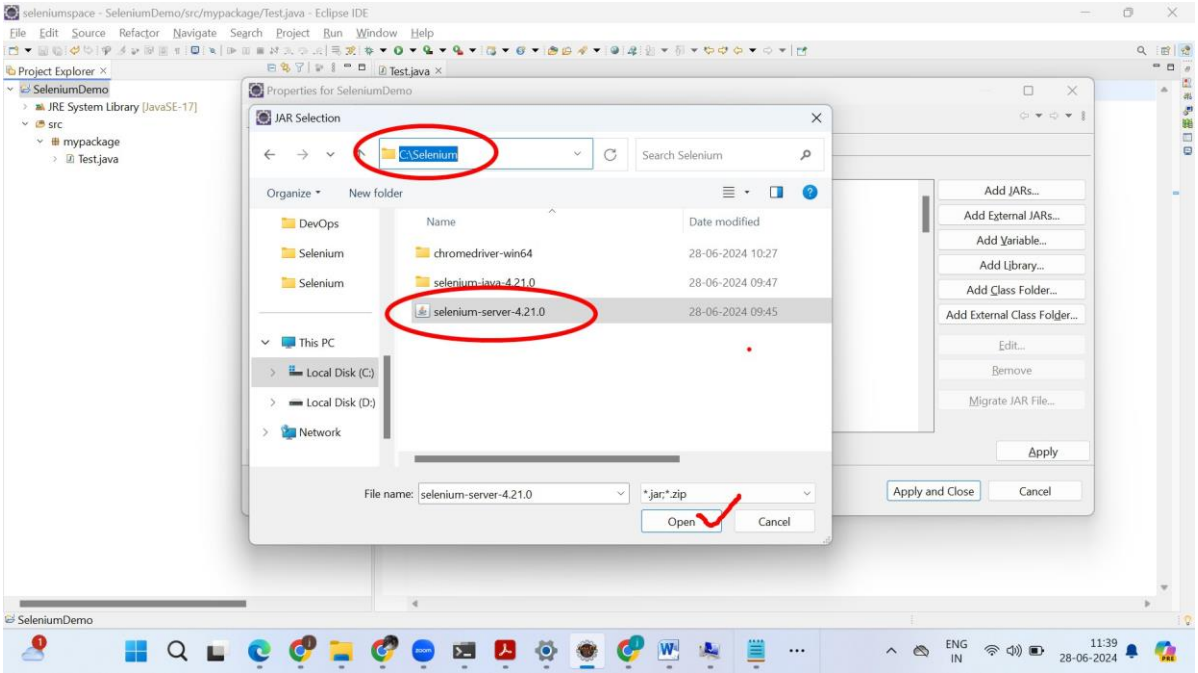
Rightclick on the SeleniumDemo project and goto Build Path and select configure build path and you can see the below screen



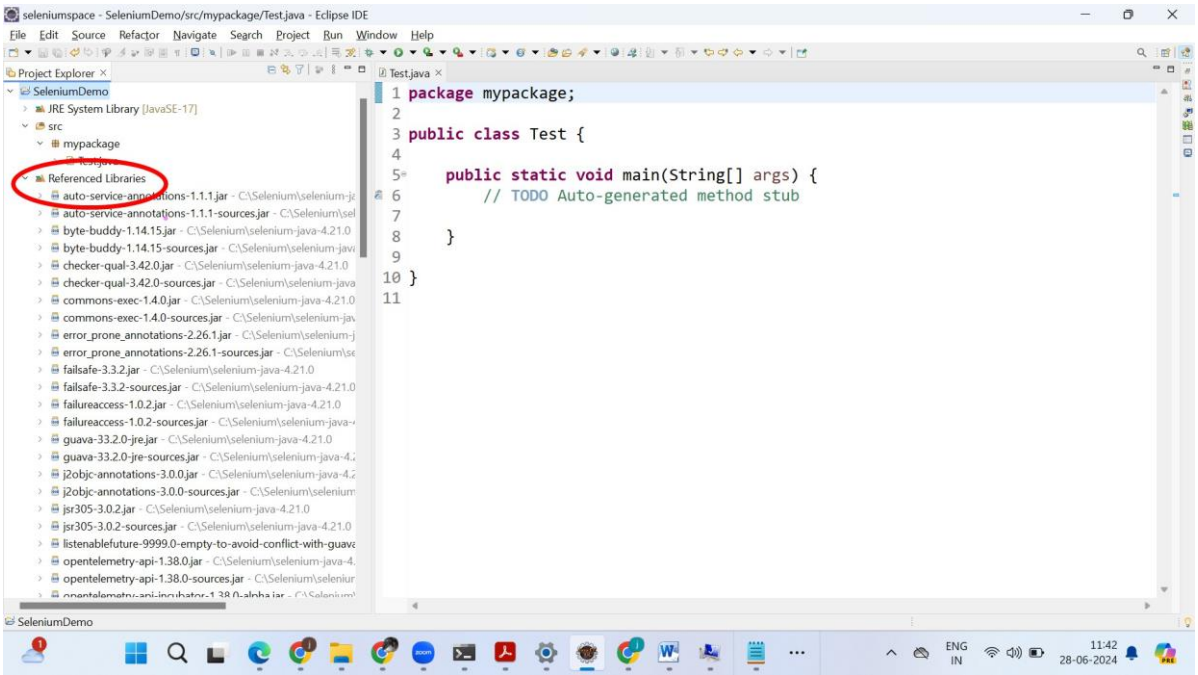
In the libraries tab select the class path and add External JARs and select all the jar files visible in the location and click open



Also Add Selenium-Server jar file as shown below

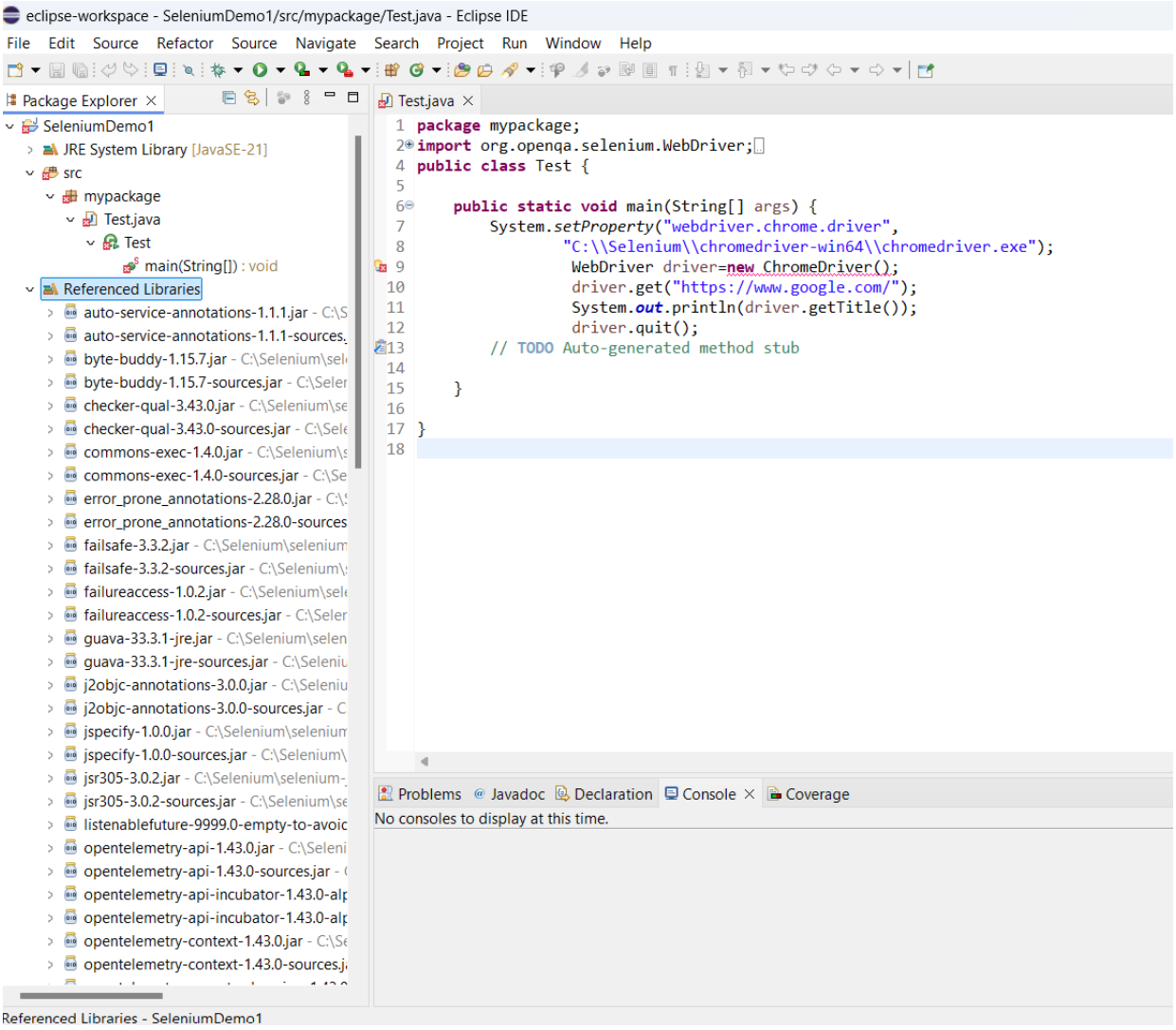


After adding the Selenium jar files you will be able to see the Referenced Libraries as shown below

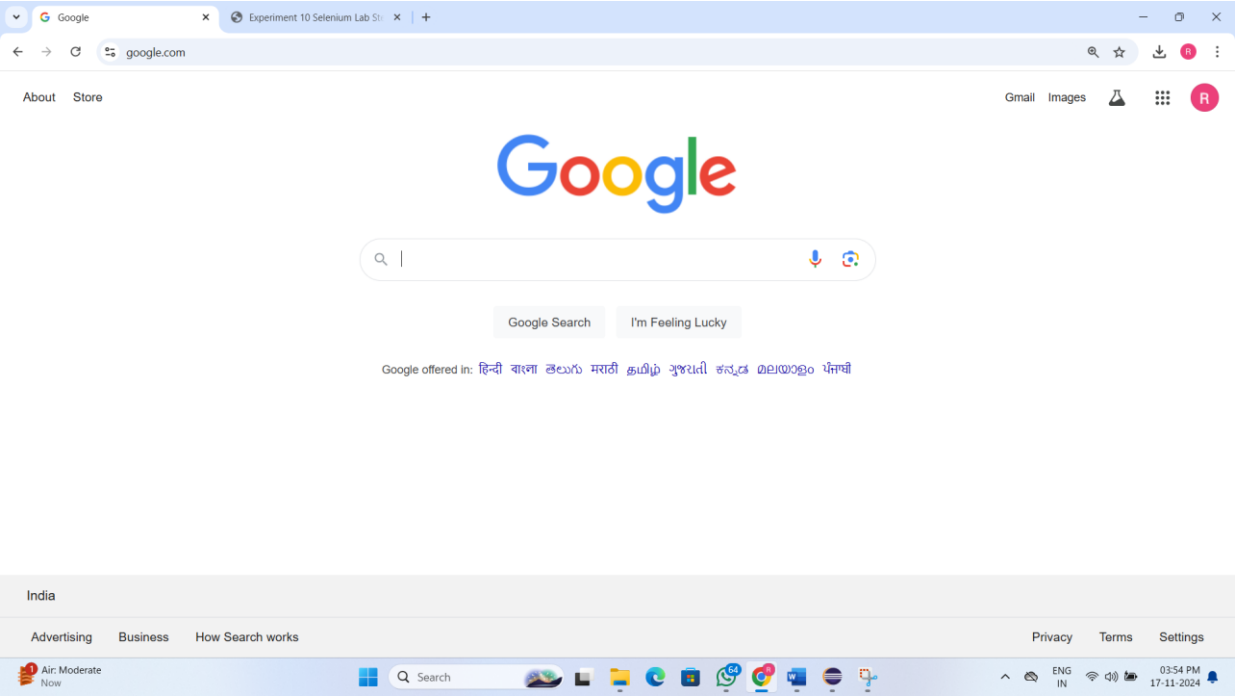


Write the below code in Test.java

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
public class Test {

    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver",
"C:\\Selenium\\chromedriver-win64\\chromedriver.exe");
        WebDriver driver=new ChromeDriver();
        driver.get("http://www.google.com");
        System.out.println(driver.getTitle());
        driver.quit();
    }
}
```



Run and see the below output. The website opens in the browser

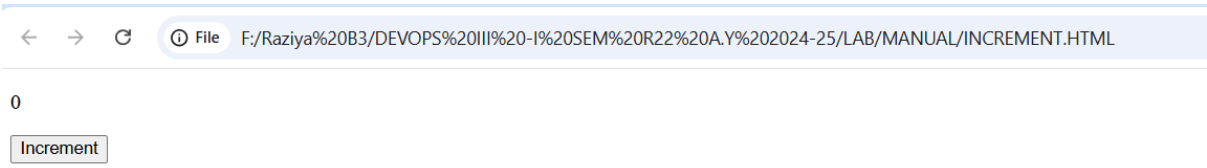


As we were trying to read the title of the page you can see the output read title of the website.

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

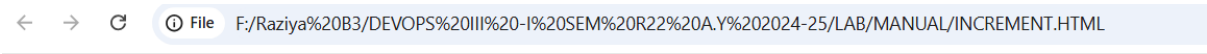
```html
//java script program to increment button value and display it on webpage
<!DOCTYPE html>
<html>
<head>
<title>Simple JavaScript Program</title>
</head>
<body>
<p id="output">0</p>
<button id="increment-button">Increment</button>
<script>
const output = document.getElementById("output");
const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

Output



0

Increment



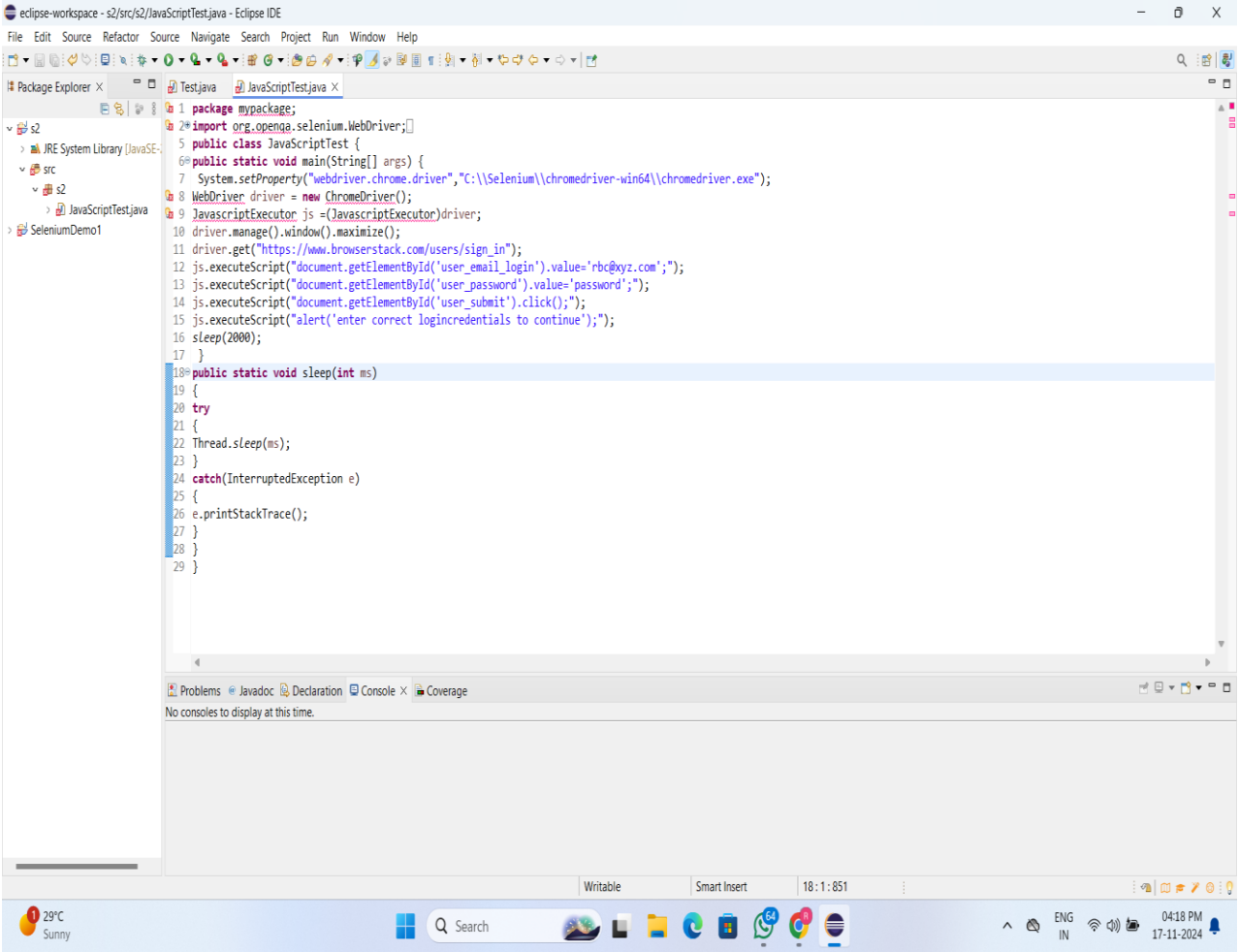
3

Increment

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

Add JavaScriptTest.java class to the earlier taken SeleniumDemo project in eclipseby right clicking on the project folder in project explorer

And write the below code



Program :

```
1  package mypackage;
2  import org.openqa.selenium.WebDriver;
5  public class JavaScriptTest {
6  public static void main(String[] args) {
7   System.setProperty("webdriver.chrome.driver","C:\\Selenium\\chromedriver-win64\\chromedriver.exe");
8  WebDriver driver = new ChromeDriver();
9  JavascriptExecutor js =(JavascriptExecutor)driver;
10 driver.manage().window().maximize();
11 driver.get("https://www.browserstack.com/users/sign_in");
12 js.executeScript("document.getElementById('user_email_login').value='rbc@xyz.com';");
13 js.executeScript("document.getElementById('user_password').value='password';");
14 js.executeScript("document.getElementById('user_submit').click();");
15 js.executeScript("alert('enter correct logincredentials to continue');");
16 sleep(2000);
17  }
18 public static void sleep(int ms)
19 {
20 try
21 {
22 Thread.sleep(ms);
23 }
24 catch(InterruptedException e)
25 {
26 e.printStackTrace();
27 }
28 }
29 }
```

Run the code and see the below output