VIDYAVARDHAKA COLLEGE OF ENGINEERING

P.B. No.206, Gokulam, III - Stage, Mysore - 570 002, Karnataka, INDIA.
Phone: +91 821 4276201 / 202 / 225.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DEVOPS LABORATORY 20CS77
(ACADEMIC YEAR 2023-24)

LAB MANUAL

Prepared By:

Mr. Anilkumar B H
Assistant Professor
Dept. of CSE, VVCE

Department of Computer Science and Engineering

INSTITUTIONAL MISSION AND VISION

Vision

Vidyavardhaka College of Engineering shall be a leading institution in engineering and management education enabling individuals for significant contribution to the society.

Mission

1. Provide the best teaching-learning environment through competent staff and excellent infrastructure.
2. Inculcate professional ethics, leadership qualities, communication and entrepreneurial skills to meet the societal needs.
3. Promote innovation through research and development.
4. Strengthen industry-institute interaction for knowledge sharing.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Vision

The Department of Computer Science and Engineering shall create professionally competent and socially responsible engineers capable of working in global environment.

Mission

1. Promote best practices in teaching-learning, with dedicated staff and supportive infrastructure to create technically competent engineers.

2. Inculcate ethical and cultural values, communication and Entrepreneurial skills.

3. Provide exposure to tools and technologies necessary to pursue higher education and research.

4. Improve Industry-Institute relationship for mutual benefit.

PROGRAM SPECIFIC OUTCOMES (PSO)

Students graduating from the Department of Computer Science and Engineering will

PSO1: Apply algorithmic principles and computer science theory to perform complexity analysis in the modeling and design of computer, computer networks and computer-based systems using appropriate tools and techniques.

PSO2: Design efficient data models & databases for challenging applications.

PSO3: Demonstrate proficiency in design, development and testing software applications to comply with system specifications.

PROGRAM OUTCOMES (PO'S) BASED ON NBA GRADUATE ATTRIBUTES

Engineering Graduates will be able to:

PO1: Engineering Knowledge: Apply the knowledge of mathematics, science, engineering, fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern tool usage: Create, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment and sustainability: Understand the impact of the professional Engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8: Ethics: Apply ethical principles and commit professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in Multi-disciplinary environments.

PO12: Lifelong learning: Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change.

CONTENTS

Sl.no.	Experiments	Page No.
	A-Demonstration	
	A1. Demonstrate and Create project in local and remote repository using GitBash and GitHub and apply init, status, log, add, commit, push, config, clone and reset commands on repository.	
	A2. Demonstrate to create a project in remote repository and apply fork, merge, diff, merge conflict, branch and pull request concepts on repository using GitHub.	
	A3. Demonstrate the process of integration github repository with Jenkins to automate the project execution in CI/CD pipeline.	
B-Exercise		
	B1. Create a docker image for an application stored in local repository and run the application using docker image.	
	B2. Create a process to dockerize the image for mysql database, demonstrate the working of client and server model using docker.	
	B3. Create and configure Jenkins files for workflow and build of an application and push the image	
C-Structured Inquiry		
	C1. Create a maven projects with all dependencies required for the application in CI/CD pipeline.	
	C2. Integrate communication channel with Jenkins for status of project and also enable email notification for a build.	
	C2. Apply Xpath to locate elements in an application for automation testing in CI/CD pipeline.	
D-Open Ended Problem		
	D1. Create a Maven project and integrate all dependencies, integrate the project with github repository , integrate with docker and docker hub and create Continuous Integration / Continuous Deployment pipeline using Jenkins and enable email notification for status of build.	

DevOps

Git

Git track changes via a distributed version control system. This means that Git can track the state of different versions of your projects while you're developing them. It is distributed because you can access your code files from another computer – and so can other developers.

When you're building an open source project, you'll need a way to document or track your code. This helps make your work organized, and lets you keep track of the changes you've made.

Operation of Git

- Manage projects with **Repositories**
- **Clone** a project to work on a local copy
- Control and track changes with **Staging** and **Committing**
- **Branch** and **Merge** to allow for work on different parts and versions of a project
- **Pull** the latest version of the project to a local copy
- **Push** local updates to the main project

A1. Demonstrate and Create project in local and remote repository using GitBash and GitHub and apply init, status, log, add, commit, push, config, clone and reset commands on repository.

Git Init command

This command is used to create a local repository.

Syntax

1. $ git init Demo

```
HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop
$ git init Demo
Initialized empty Git repository in C:/Users/HiMaNshU/Desktop/Demo/.git/

HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop
$
```

Git status command

The status command is used to display the state of the working directory and the staging area. It also lists the files that you've changed and those you still need to add or commit.

Syntax

1. $ git status

```
HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/Git-example (master)
$ git status
On branch master
Your branch is based on 'origin/master', but the upstream is gone.
  (use "git branch --unset-upstream" to fixup)

nothing to commit, working tree clean
```

Git log Command

This command is used to check the commit history.

Syntax

1. $ git log

```
HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/gitexample2 (master)
$ git log
commit 1d2bc037a54eba76e9f25b8e8cf7176273d13af0 (HEAD -> master, origin/master,
origin/HEAD)
Author: ImDwivedi1 <52317024+ImDwivedi1@users.noreply.github.com>
Date:   Fri Aug 30 11:05:06 2019 +0530

    Initial commit
```

By default, if no argument passed, Git log shows the most recent commits first. We can limit the number of log entries displayed by passing a number as an option, such as -3 to show only the last three entries.

Git add command

This command is used to add one or more files to staging (Index) area.

Syntax

To add one file

1. $ git add Filename

To add more than one file

1. $ git add*

```
HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/Git-example (master)
$ git add README.md
```

Git commit command

Commit command is used in two scenarios. They are as follows.

Git commit -m

This command changes the head. It records or snapshots the file permanently in the version history with a message.

Syntax

1. $ git commit -m " Commit Message"

```
HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/Git-example (master)
$ git commit -a -m "Adding the key of c"
[master (root-commit) 758797a] Adding the key of c
 1 file changed, 2 insertions(+)
 create mode 100644 README.md
```

Git push Command

It is used to upload local repository content to a remote repository. Pushing is an act of transfer commits from your local repository to a remote repo. It's the complement to git fetch, but whereas fetching imports commits to local branches on comparatively pushing exports commits to remote branches.

Syntax

1. $ git push [variable name] master

```
HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/Git-example (mast
er)
$ git push origin master
```

Git config command

This command configures the user. The Git config command is the first and necessary command used on the Git command line.

Syntax

1. $ git config --global user.name "ImDwivedi1"
2. $ git config --global user.email "Himanshudubey481@gmail.com"

Git clone command

This command is used to make a copy of a repository from an existing URL. If I want a local copy of my repository from GitHub, this command allows creating a local copy of that repository on your local directory from the repository URL.

Syntax

1. $ git clone URL

```
HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/Git-example (master)
$ git clone https://github.com/ImDwivedi1/Git-Example.git
Cloning into 'Git-Example'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
```

Git Reset

The term reset stands for undoing changes. The git reset command is used to reset the changes.

1. $ git reset --hard

Consider the below output:

```
HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (test2)
$ git reset --hard
HEAD is now at 34c25eb Revert "CSS file "
```

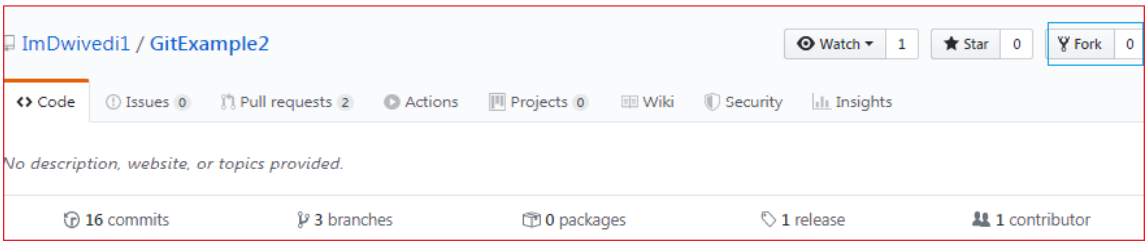
A2. Demonstrate to create a project in remote repository and apply fork, merge, diff, merge conflict, branch and pull request concepts on repository using GitHub.

Git Fork

A fork is a rough copy of a repository. Forking a repository allows you to freely test and debug with changes without affecting the original project.

It is a straight-forward process. Steps for forking the repository are as follows:

- Login to the GitHub account.
- Find the GitHub repository which you want to fork.
- Click the Fork button on the upper right side of the repository's page.



Git Merge

In Git, the merging is a procedure to connect the forked history. It joins two or more development history together. The git merge command facilitates you to take the data created by git branch and integrate them into a single branch.

1. $ git merge master

See the below output:

```
HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (test)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.

HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master)
$ git merge 2852e020909dfe705707695fd6d715cd723f9540
Updating 4a6693a..2852e02
Fast-forward
 newfile.txt  | 1 +
 newfile1.txt | 1 +
 2 files changed, 2 insertions(+)
 create mode 100644 newfile.txt
 create mode 100644 newfile1.txt

HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master)
$
```

Git Diff

It compares the different versions of data sources. The version control system stands for working with a modified version of files. So, the diff command is a useful tool for working with Git.

```
HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (test2)
$ git diff
diff --git a/newfile1.txt b/newfile1.txt
index ade63b7..41a6a9c 100644
--- a/newfile1.txt
+++ b/newfile1.txt
@@ -3,3 +3,4 @@ i am on test2 branch.
 git commit1
 git commit2
 git merge demo
+changes are made to understand the git diff command.
```

Git Merge Conflict

When two branches are trying to merge, and both are edited at the same time and in the same file, Git won't be able to identify which version is to take for changes. Such a situation is called merge conflict.

$ mkdir git-merge-test

$ cd git-merge-test

$ git init .

$ echo "this is some content to mess with" > merge.txt

$ git add merge.txt

$ git commit -am"we are commiting the inital content"

[main (root-commit) d48e74c] we are commiting the inital content

1 file changed, 1 insertion(+)

create mode 100644 merge.txt

$ git checkout -b new_branch_to_merge_later

$ echo "totally different content to merge later" > merge.txt

$ git commit -am"edited the content of merge.txt to cause a conflict"

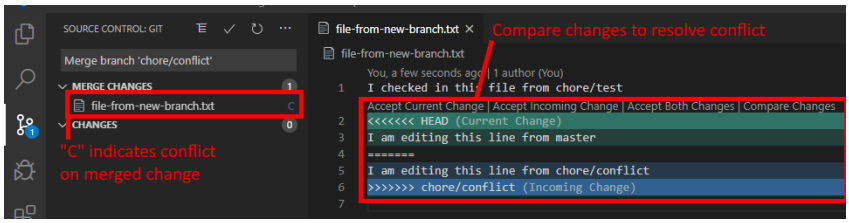
[new_branch_to_merge_later 6282319] edited the content of merge.txt to cause a conflict

1 file changed, 1 insertion(+), 1 deletion(-)

$ git merge new_branch_to_merge_later

Auto-merging merge.txt

CONFLICT (content): Merge conflict in merge.txt



Automatic merge failed; fix conflicts and then commit result

Git Pull

The term pull is used to receive data from GitHub. It fetches and merges changes from the remote server to your working directory. The **git pull command** is used to pull a repository.

```
HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master)
$ git pull
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/ImDwivedi1/GitExample2
   f1ddc7c..0a1a475  master     -> origin/master
Updating f1ddc7c..0a1a475
Fast-forward
 design2.css | 6 ++++++
 1 file changed, 6 insertions(+)
 create mode 100644 design2.css

HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master)
$
```

```
HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/Demo (master)
$ git pull https://github.com/ImDwivedi1/GitExample2.git
remote: Enumerating objects: 38, done.
remote: Counting objects: 100% (38/38), done.
remote: Compressing objects: 100% (25/25), done.
remote: Total 38 (delta 13), reused 19 (delta 7), pack-reused 0
Unpacking objects: 100% (38/38), done.
From https://github.com/ImDwivedi1/GitExample2
 * branch            HEAD       -> FETCH_HEAD

HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/Demo (master)
$
```

Git Branch

A branch is a version of the repository that diverges from the main working project. It is a feature available in most modern version control systems. A Git project can have more than one branch.

Create Branch

You can create a new branch with the help of the **git branch** command. This command will be used as:

Syntax:

1. $ git branch **<branch** name>

Output:

```
HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master)
$ git branch B1
```

List Branch

You can List all of the available branches in your repository by using the following command.

1. $ git branch

Output:

```
HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master)
$ git branch
  B1
  branch3
* master

HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master)
$ git branch --list
  B1
  branch3
* master
```

1. $ git branch -d<**branch** name>

Output:

```
HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master)
$ git branch -d B1
Deleted branch B1 (was 554a122).
```

A3. Demonstrate the process of integration github repository with Jenkins to automate the project execution in CI/CD pipeline.

Install Jenkins on Windows

1.Install Java Development Kit (JDK)

2.Download JDK 8 and choose windows 32-bit or 64-bit according to your system configuration. Click on "accept the license agreement." Set the Path for the Environmental Variable for JDK.

•Go to System Properties. Under the "Advanced" tab, select "Environment Variables."

•Under system variables, select "new." Then copy the path of the JDK folder and paste it in the corresponding value field. Similarly, do this for JRE.

•Under system variables, set up a bin folder for JDK in PATH variables.

•Go to command prompt and type the following to check if Java has been successfully installed:

2.Download and Install Jenkins

3.Download Jenkins. Under LTS, click on windows.

•After the file is downloaded, unzip it. Click on the folder and install it. Select "finish" once done.

•4. Run Jenkins on Localhost 8080

•Once Jenkins is installed, explore it. Open the web browser and type "localhost:8080".

•Enter the credentials and log in. If you install Jenkins for the first time, the dashboard will ask you to install the recommended plugins. Install all the recommended plugins.

5. Jenkins Server Interface

•New Item allows you to create a new project.

•Build History shows the status of your builds.

•Manage System deals with the various configurations of the system.

6. Build and Run a Job on Jenkins

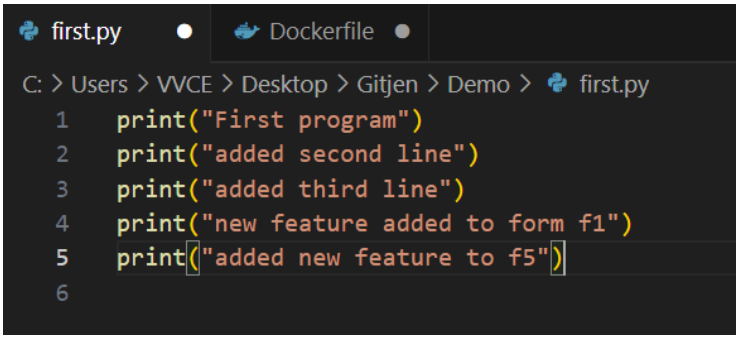
•Select a new item (Name - Jenkins_demo). Choose a freestyle project and click Ok.

•Under the General tab, give a description like "This is my first Jenkins job." Under the "Build Triggers" tab, select add built step and then click on the "Execute Windows" batch command.

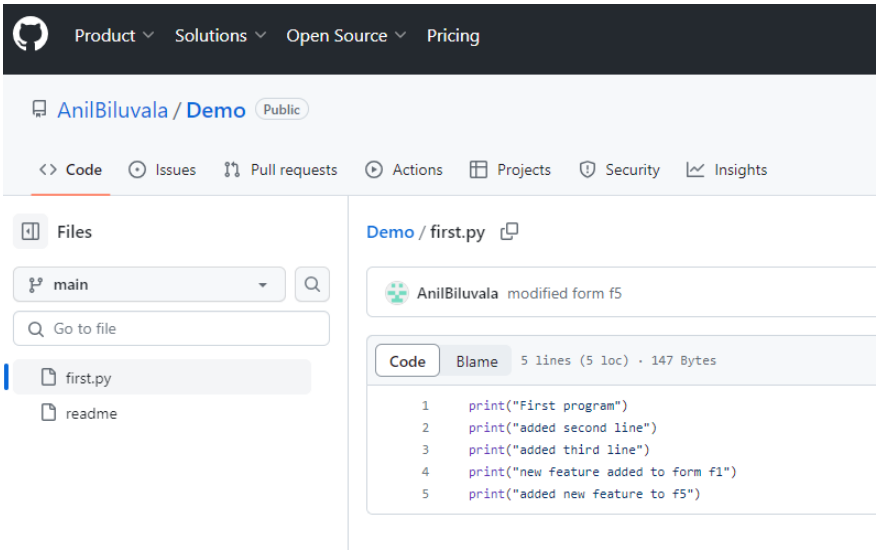
•In the command box, type the following: echo "Hello... This is my first Jenkins Demo: %date%: %time% ". Click on apply and then save.

•Select build now. You can see a building history has been created. Click on that. In the console output, you can see the output of the first Jenkins job with time and date. Project configuration

1. Install python interpreter on local machine.
2. Write python script and debug the program for any errors.



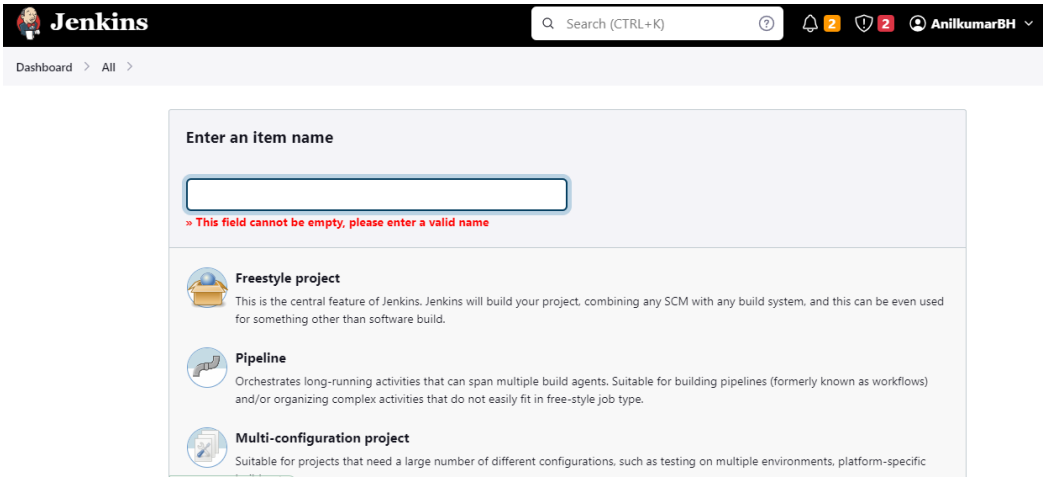
3. Push the project into git repository.



4. Launch the jenkins and configure the pipeline.

Configuration of Jenkins

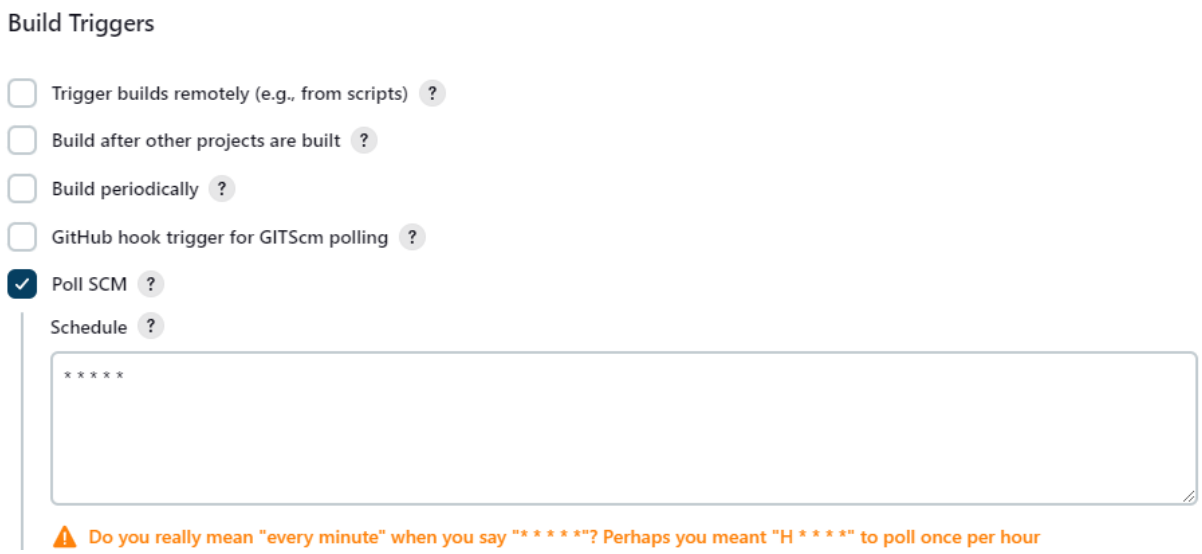
1. Create new project and provide name for the project. Choose “Freestyle Project”.



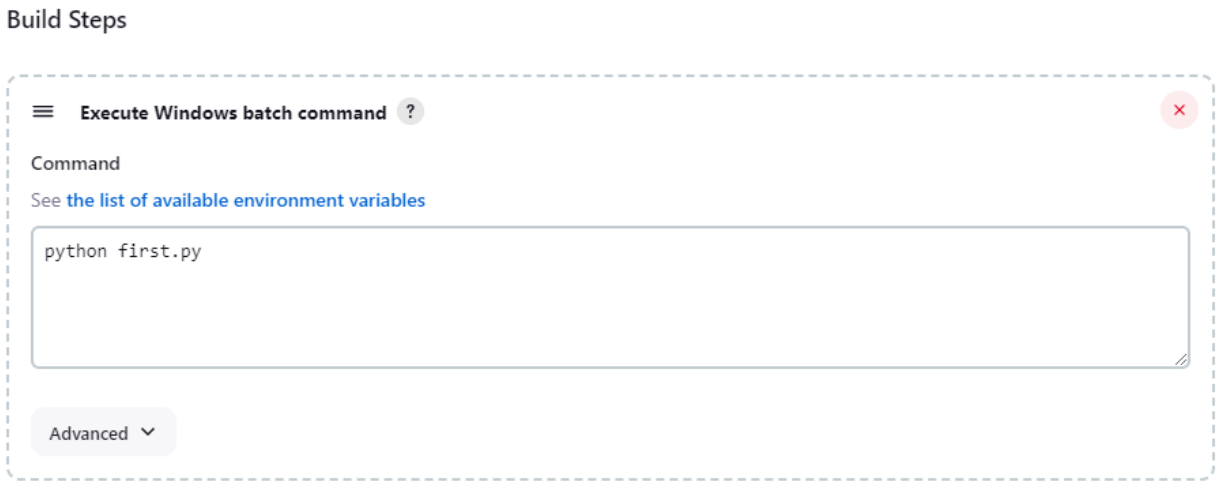
2. Click on newly created project and move to configuration. Under source Code Management provide git repository.

3. Provide branch name where project is located.

4. Under Build Triggers select Poll SCM and configure the schedule according to your project.



5. In build steps choose “Execute Windows batch command” and provide command for execution.



6. Apply and save your configuration.

7. Click on “Build Now” for detailed output click on “Console Output” and verify the process. Finally build status will be displayed according to your project implementation.

B-Exercise

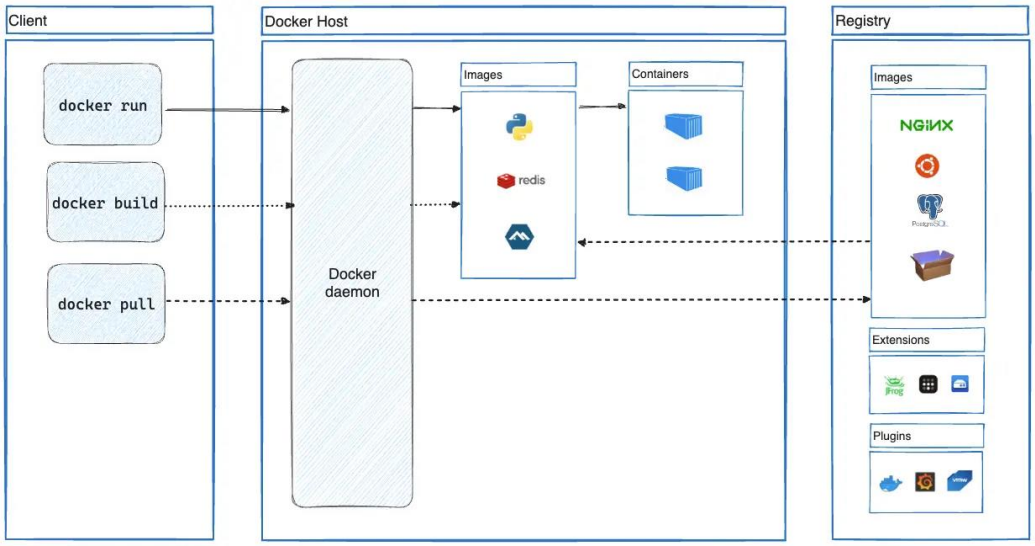
B1. Create a docker image for an application stored in local repository and run the application using docker image.

Introduction to Docker

Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications. By taking advantage of Docker's methodologies for shipping, testing, and deploying code, you can significantly reduce the delay between writing code and running it in production.

Docker architecture

Docker uses a client-server architecture. The Docker client talks to the Docker daemon, which does the heavy lifting of building, running, and distributing your Docker containers. The Docker client and daemon can run on the same system, or you can connect a Docker client to a remote Docker daemon. The Docker client and daemon communicate using a REST API, over UNIX sockets or a network interface. Another Docker client is Docker Compose, that lets you work with applications consisting of a set of containers.



Install Docker Desktop on Windows

System requirements

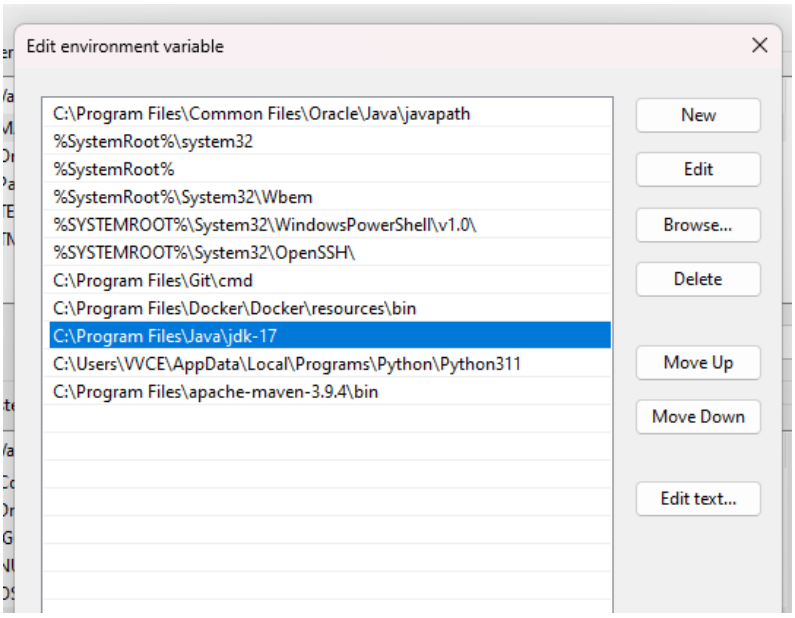
- WSL version 1.1.3.0 or later.
- Windows 11 64-bit: Home or Pro version 21H2 or higher, or Enterprise or Education version 21H2 or higher.
- Windows 10 64-bit:
- We recommend Home or Pro 22H2 (build 19045) or higher, or Enterprise or Education 22H2 (build 19045) or higher.
- Minimum required is Home or Pro 21H2 (build 19044) or higher, or Enterprise or Education 21H2 (build 19044) or higher.
- Turn on the WSL 2 feature on Windows. For detailed instructions, refer to the Microsoft documentationopen_in_new.

The following hardware prerequisites are required to successfully run WSL 2 on Windows 10 or Windows 11:

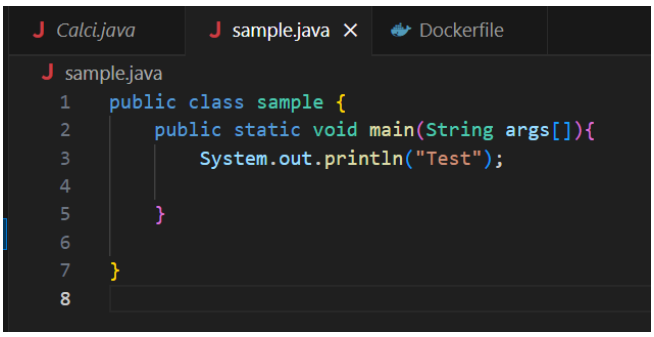
- 64-bit processor with Second Level Address Translation (SLAT)open_in_new
- 4GB system RAM
- Enable hardware virtualization in BIOS. For more information, see Virtualization.

Configuration of Java applications

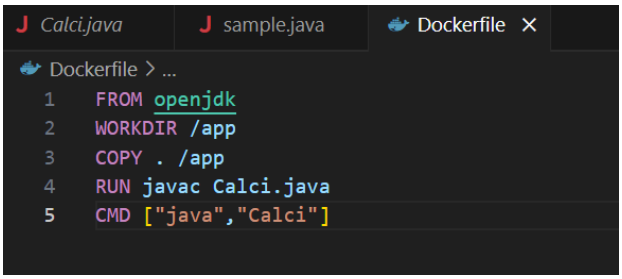
1. Install prerequisites for java applications like jdk tool.
2. Configure environment variables for jdk.



3. Create java application.

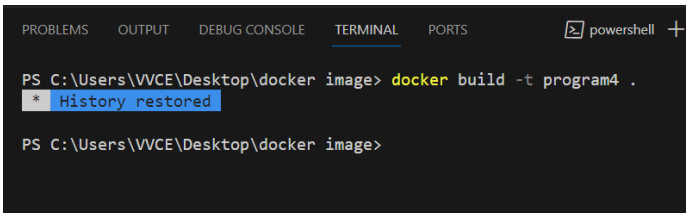


4. Create Dockerfile where configuration details are provided for creating image.

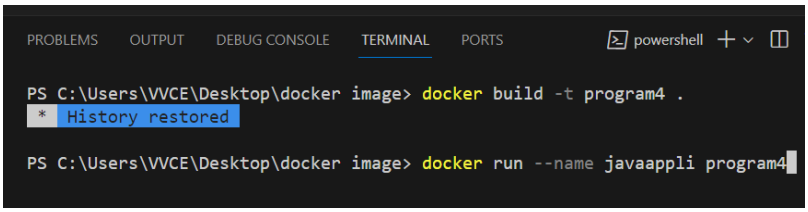


5. Complile the java program for any errors.

6. Create image for the application.



7. Run the application using docker image.



B2. Create a process to dockerize the image for mysql database, demonstrate the working of client and server model using docker.

This program is to demonstrate pipeline within the application. Mysql image contains built in client and server.

Procedure:

1. Run docker desktop application.
2. Go to command prompt and search mysql image in hub.

```
C:\Users\VVCE>docker search mysql
NAME                            DESCRIPTION                                     STARS     OFFICIAL   AUTOMATED
mysql                           MySQL is a widely used, open-source relation…   14767     [OK]
mariadb                         MariaDB Server is a high performing open sou…   5633      [OK]
percona                         Percona Server is a fork of the MySQL relati…   624       [OK]
phpmyadmin                      phpMyAdmin - A web interface for MySQL and M…   928       [OK]
bitnami/mysql                   Bitnami MySQL Docker Image                      106                  [OK]
bitnami/mysqld-exporter                                                         6
cimg/mysql                                                                      2
ubuntu/mysql                    MySQL open source fast, stable, multi-thread…   56
rapidfort/mysql                 RapidFort optimized, hardened image for MySQL   25
rapidfort/mysql8-ib             RapidFort optimized, hardened image for MySQ…   9
google/mysql                    MySQL server for Google Compute Engine          25                   [OK]
rapidfort/mysql-official        RapidFort optimized, hardened image for MySQ…   9
elestio/mysql                   Mysql, verified and packaged by Elestio         0
hashicorp/mysql-portworx-demo                                                   0
bitnamicharts/mysql                                                             0
newrelic/mysql-plugin           New Relic Plugin for monitoring MySQL databa…   1                    [OK]
databack/mysql-backup           Back up mysql databases to... anywhere!         105
linuxserver/mysql               A Mysql container, brought to you by LinuxSe…   41
mirantis/mysql                                                                  0
linuxserver/mysql-workbench                                                     54
vitess/mysqlctld                vitess/mysqlctld                                1                    [OK]
eclipse/mysql                   Mysql 5.7, curl, rsync                          1                    [OK]
drupalci/mysql-5.5              https://www.drupal.org/project/drupalci         3                    [OK]
drupalci/mysql-5.7              https://www.drupal.org/project/drupalci         0
datajoint/mysql                 MySQL image pre-configured to work smoothly …   2                    [OK]
```

3. Pull the image from hub.
4. Run the image

```
C:\Users\VVCE>docker run mysql
2024-01-14 14:02:46+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.2.0-1.el8 started.
2024-01-14 14:02:46+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
2024-01-14 14:02:46+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.2.0-1.el8 started.
2024-01-14 14:02:46+00:00 [ERROR] [Entrypoint]: Database is uninitialized and password option is not specified
    You need to specify one of the following as an environment variable:
    - MYSQL_ROOT_PASSWORD
    - MYSQL_ALLOW_EMPTY_PASSWORD
    - MYSQL_RANDOM_ROOT_PASSWORD
```

5. Configure password form mysql server

```
C:\Users\VVCE>docker run -e MYSQL_ROOT_PASSWORD="root123" -d mysql
24fa44019460bf7c2ac0871e68bb0d76a3d35ee90f1ba39bb1963409aa6dc02a
```

-e -> setting environment variable “MYSQL_ROOT_PASSWORD”

-d-> detach running in background.

6. Open another tab run client using linux command.

```
C:\Users\VVCE>docker run -it mysql /bin/bash
bash-4.4#
```

7. Required ip address of server to login from clinet. Inspec the server model.

```
C:\Users\VVCE>docker container ls
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS                 NAMES
be71b60ce703   mysql     "docker-entrypoint.s…"   2 minutes ago   Up 2 minutes   3306/tcp, 33060/tcp   objective_hofstad
ter
24fa44019460   mysql     "docker-entrypoint.s…"   4 minutes ago   Up 4 minutes   3306/tcp, 33060/tcp   quirky_euclid
```

8. Inspect contianer based on names

```
C:\Users\VVCE>docker inspect quirky_euclid
[
    {
        "Id": "24fa44019460bf7c2ac0871e68bb0d76a3d35ee90f1ba39bb1963409aa6dc02a",
        "Created": "2024-01-14T14:05:37.901160867",
        "Path": "docker-entrypoint.sh",
        "Args": [
            "mysqld"
        ],
        "State": {
            "Status": "running",
            "Running": true,
```

```
    "Gateway": "172.17.0.1",
    "IPAddress": "172.17.0.2",
    "IPPrefixLen": 16,
    "IPv6Gateway": "",
    "GlobalIPv6Address": "",
    "GlobalIPv6PrefixLen": 0,
    "DriverOpts": null
}
```

9. Note down the server ip address and in clinet enter in to server

```
C:\Users\VVCE>docker run -it mysql /bin/bash
bash-4.4# mysql -h 172.17.0.2 -u root -p
Enter password:
```

10. Logged in from client.

```
C:\Users\VVCE>docker run -it mysql /bin/bash
bash-4.4# mysql -h 172.17.0.2 -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.2.0 MySQL Community Server - GPL

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

B3. Create and configure Jenkins files for workflow and build of an application and push the image on docker hub.

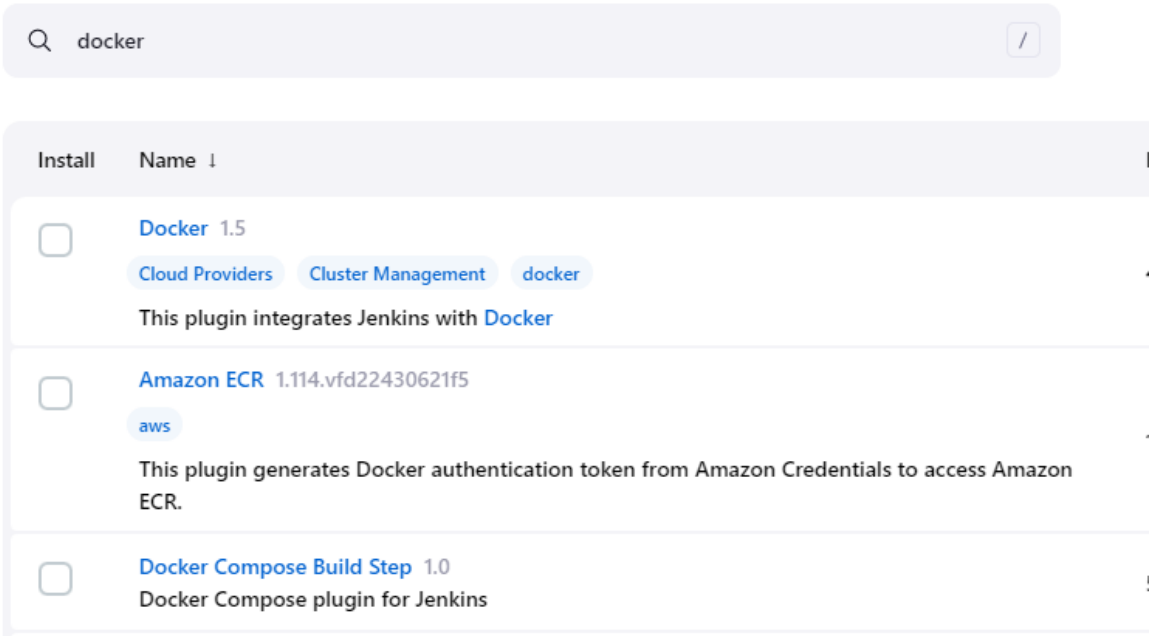
Configuration Steps:

1. Go to Jenkins and configure docker plug-ins.

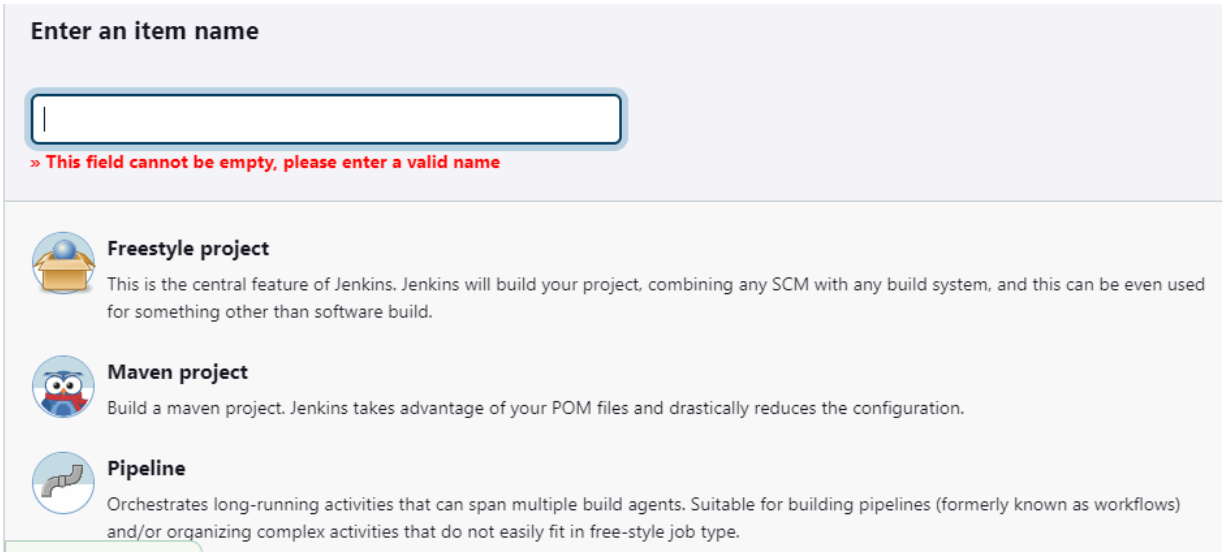
   Install

   1. CloudBees Docker Custom Build EnvironmentVersion1.7.3
   2. Docker
   3. Docker Compose Build Step

Make sure that these three plug-in must be installed.



2. Create a pipeline project



3. Go to project configuration add pipeline script under “Advanced project Options”

```
pipeline
{
 agent any
 environment
 {
  dockerImage=''
  registry='anilkumarbh/pythonapp'
  registryCredential='jenkin_docker_token'
 }
stages{
      stage('Checkout')
      {
       steps{
      checkout scmGit(branches: [[name: '*/main']], extensions: [], userRemoteConfigs: [[url: 'https://github.com/AnilBiluvala/dpipeline.git']])
      }
      }
     stage('Build Docker image')
      {
```

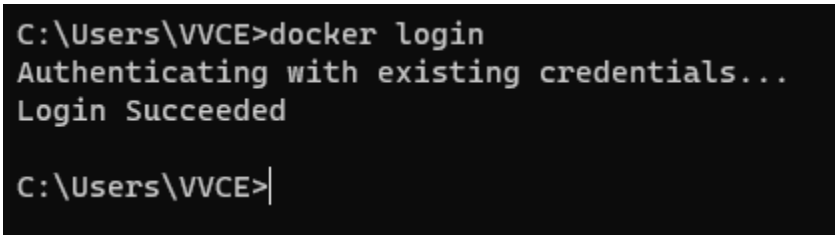
```
    steps{
     script{
          dockerImage=docker.build registry
        }
       }
   }

  }
}
```

In script add git url, docker hub credentias, docker hub image name.

Save the configuration and run the build. Image will be created.

4. Push the image to docker hub.



Provide docker hub crdentials.

5. Tag the image to be pushed.

```
C:\Users\VVCE>docker images
REPOSITORY                TAG       IMAGE ID       CREATED        SIZE
anilkumarbh/first_push    latest    08743a947625   7 days ago     470MB
javaappp4                 latest    21a1420fa6fe   7 weeks ago    470MB
build4                    latest    7ce1f5587702   7 weeks ago    1.02GB
anilkumarbh/javaapp       ver2      ea55569cb7c8   2 months ago   470MB
myjavaapp                 latest    ea55569cb7c8   2 months ago   470MB
mysql                     latest    a3b6608898d6   2 months ago   596MB
mysql                     <none>    b2013ac99101   3 months ago   577MB
phpmyadmin                latest    61dcc026c415   3 months ago   562MB
jenkins/jenkins           lts       3d0d48f61941   3 months ago   478MB
phpmyadmin/phpmyadmin     latest    933569f3a9f6   5 months ago   562MB
hello-world               latest    9c7a54a9a43c   8 months ago   13.3kB

C:\Users\VVCE>docker tag build4:latest anilkumarbh/javaapplication
```

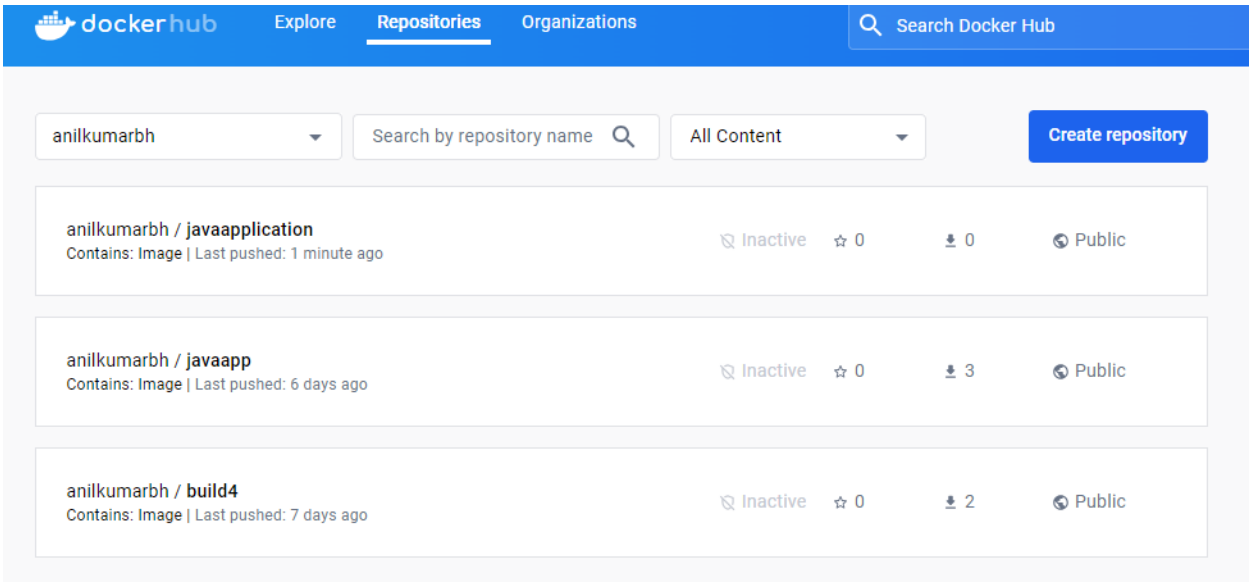
docker tag <reponame>:<tagname> <hubname>/<display name>

```
C:\Users\VVCE>docker images
REPOSITORY                TAG       IMAGE ID       CREATED        SIZE
anilkumarbh/first_push    latest    08743a947625   7 days ago     470MB
javaappp4                 latest    21a1420fa6fe   7 weeks ago    470MB
anilkumarbh/javaapplication   latest    7ce1f5587702   7 weeks ago    1.02GB
build4                    latest    7ce1f5587702   7 weeks ago    1.02GB
anilkumarbh/javaapp       ver2      ea55569cb7c8   2 months ago   470MB
myjavaapp                 latest    ea55569cb7c8   2 months ago   470MB
mysql                     latest    a3b6608898d6   2 months ago   596MB
mysql                     <none>    b2013ac99101   3 months ago   577MB
phpmyadmin                latest    61dcc026c415   3 months ago   562MB
jenkins/jenkins           lts       3d0d48f61941   3 months ago   478MB
phpmyadmin/phpmyadmin     latest    933569f3a9f6   5 months ago   562MB
hello-world               latest    9c7a54a9a43c   8 months ago   13.3kB
```

6. Push the image to hub.

```
C:\Users\VVCE>docker push anilkumarbh/javaapplication
Using default tag: latest
The push refers to repository [docker.io/anilkumarbh/javaapplication]
4623bd9cd185: Mounted from anilkumarbh/build4
b7c8324b261e: Mounted from anilkumarbh/build4
701d0b971f5f: Mounted from anilkumarbh/build4
619584b251c8: Mounted from anilkumarbh/build4
ac630c4fd960: Mounted from anilkumarbh/build4
86e50e0709ee: Waiting
12b956927ba2: Waiting
266def75d28e: Preparing
29e49b59edda: Waiting
1777ac7d307b: Waiting
```

7. Image will be available in docker hub.



C-Structured Inquiry

C1. Create a maven projects with all dependencies required for the application in CI/CD pipeline.

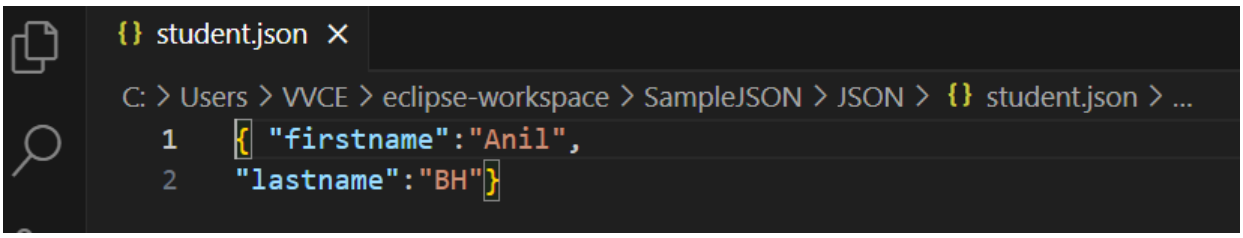
Required tools:

1. Eclipse IDE

2. Maven repository
3. JSON

Reading data from JSON file.

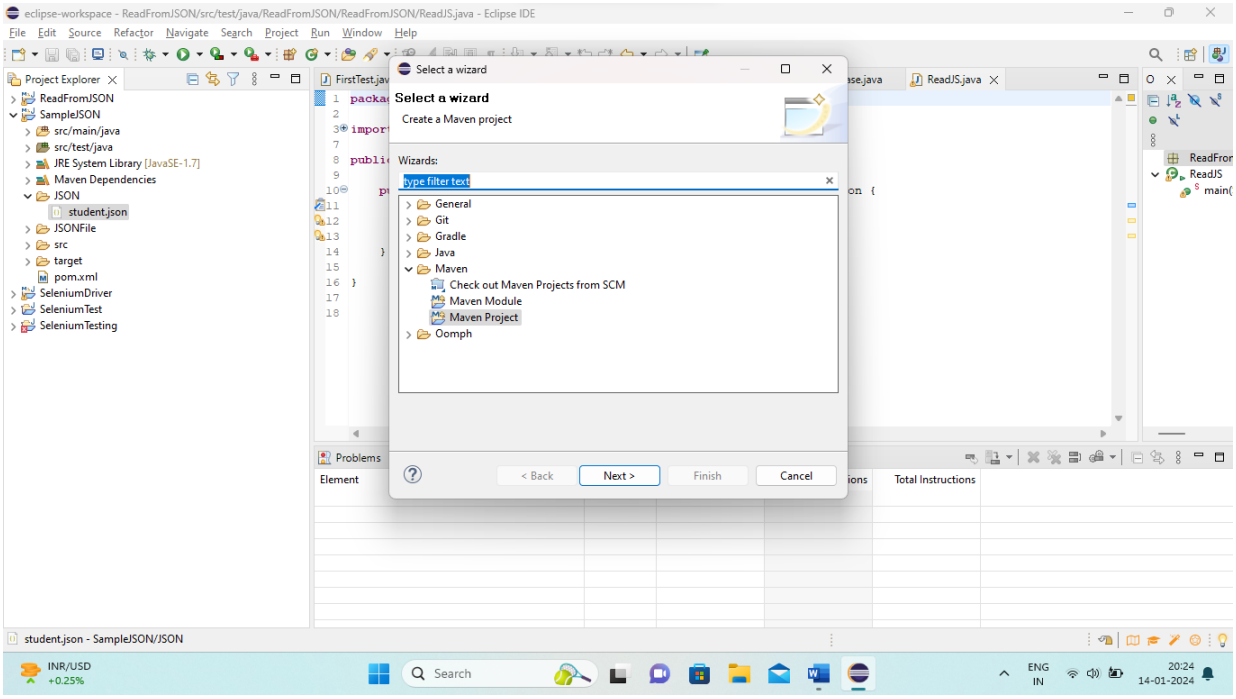
Create JSON file



Save the data.

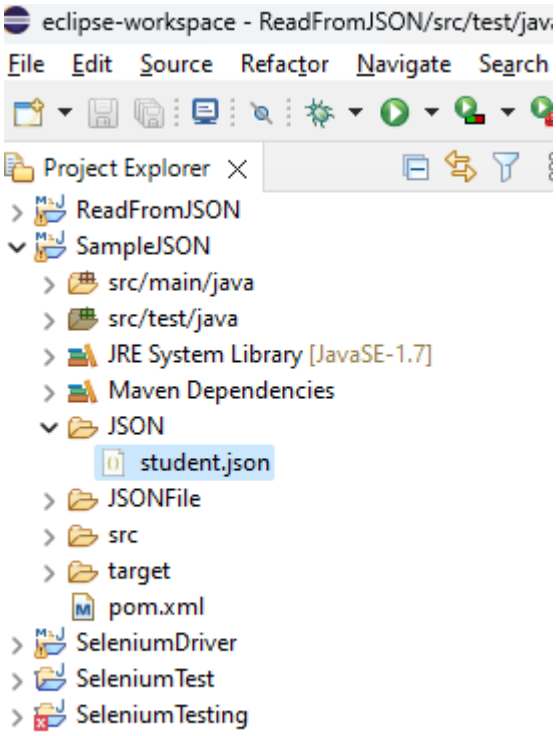
1. Create maven project

New Project -> Others -> select maven project



Provide project name

2. Create folder “JSON” and add .json file to it.

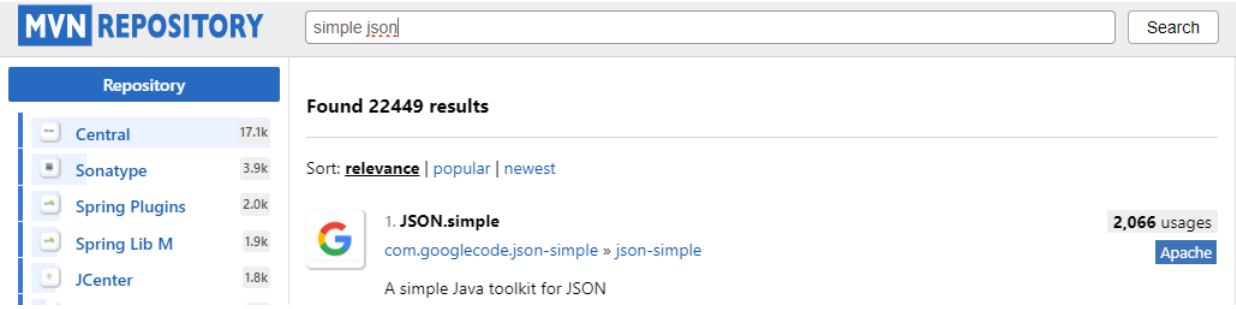


3. Adding JSON dependency to the project in POM.xml file.

   Procedure to add dependency

   Step1 : Go to maven repository

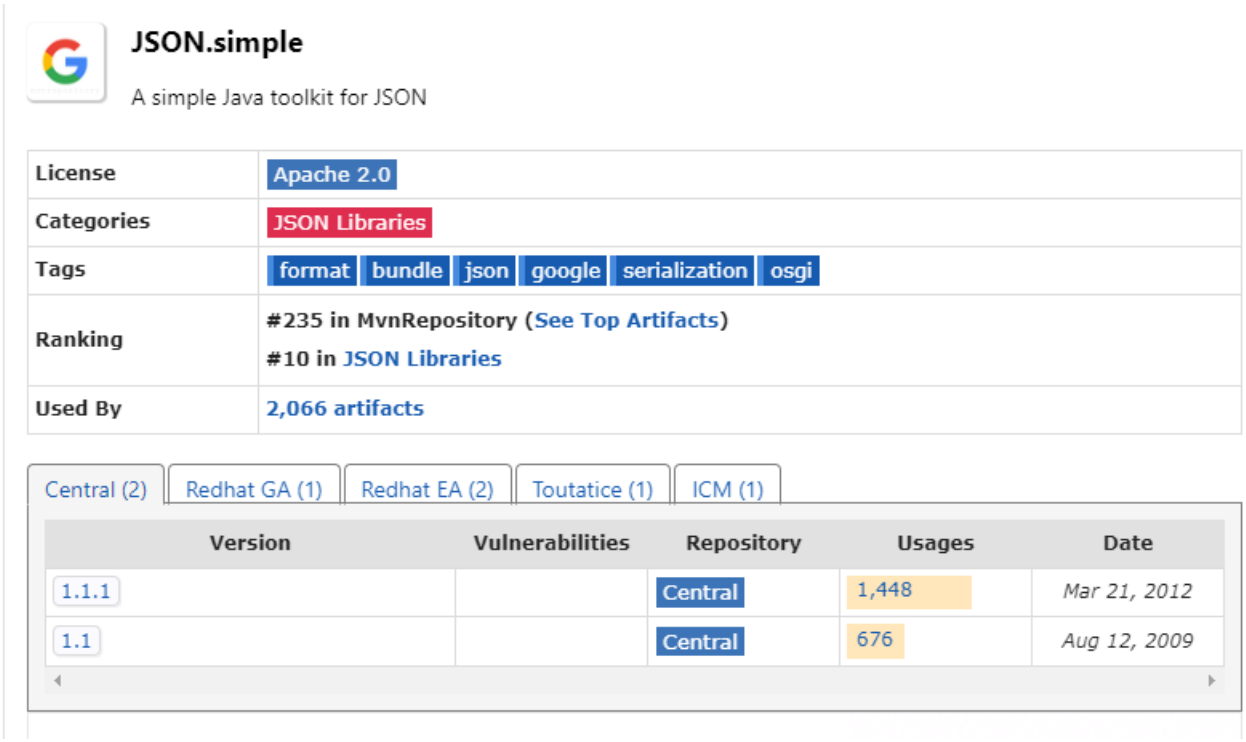
   Stept2 : search simple json



Step3 : choose relevant based on highest usages.

Setp4: choose version

Stept5: copy the dependency.



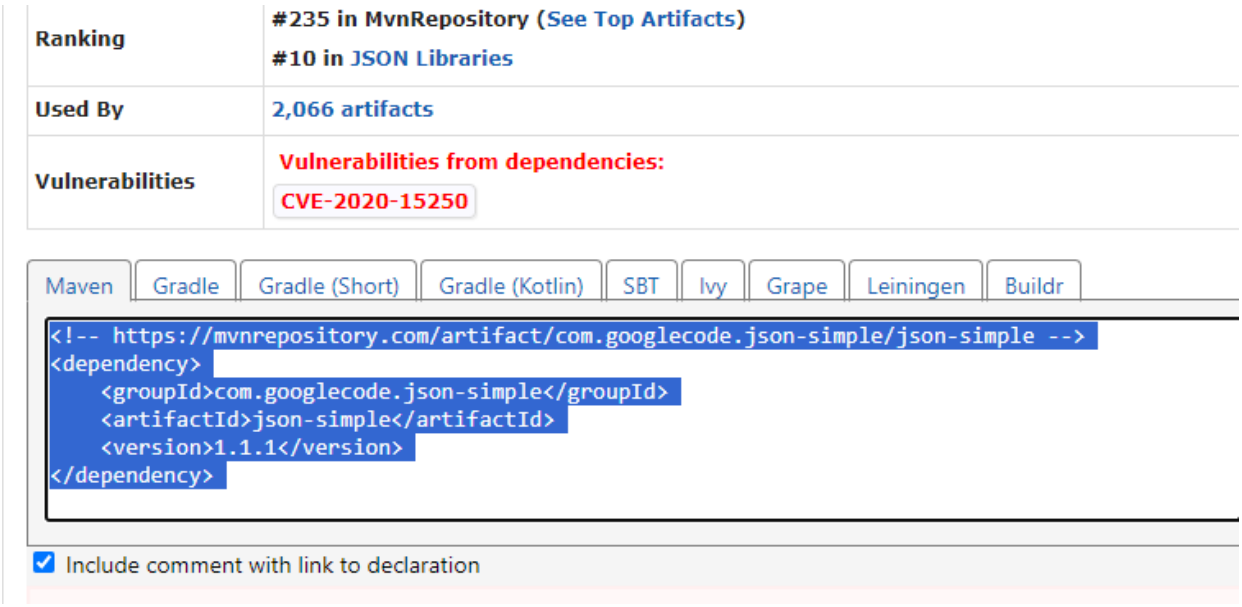
JSON.simple

A simple Java toolkit for JSON

License	Apache 2.0
Categories	JSON Libraries
Tags	format bundle json google serialization osgi
Ranking	#235 in MvnRepository (See Top Artifacts) #10 in JSON Libraries
Used By	2,066 artifacts

Central (2) | Redhat GA (1) | Redhat EA (2) | Toutatice (1) | ICM (1)

Version	Vulnerabilities	Repository	Usages	Date
1.1.1		Central	1,448	Mar 21, 2012
1.1		Central	676	Aug 12, 2009



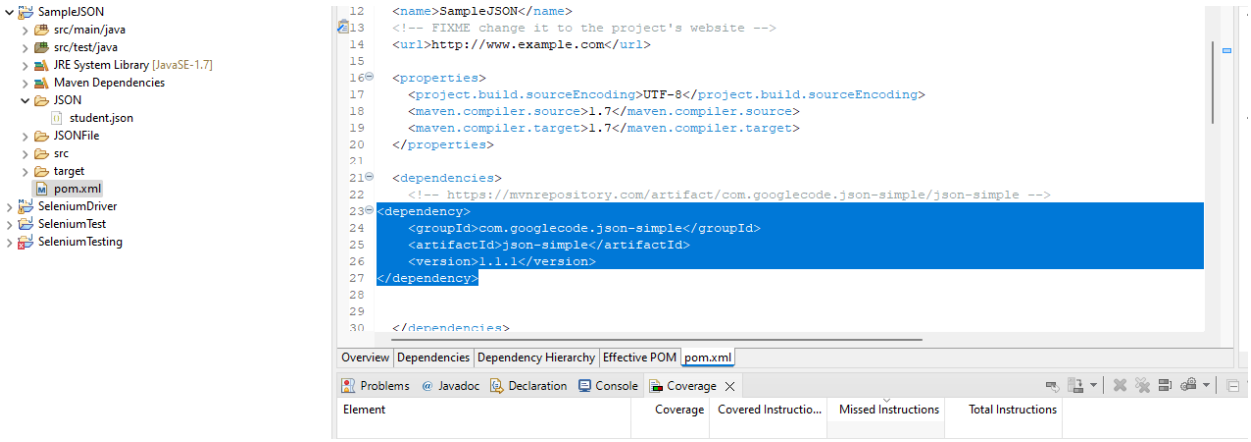
Ranking	#235 in MvnRepository (See Top Artifacts) #10 in JSON Libraries
Used By	2,066 artifacts
Vulnerabilities	Vulnerabilities from dependencies: CVE-2020-15250

Maven | Gradle | Gradle (Short) | Gradle (Kotlin) | SBT | Ivy | Grape | Leiningen | Buildr

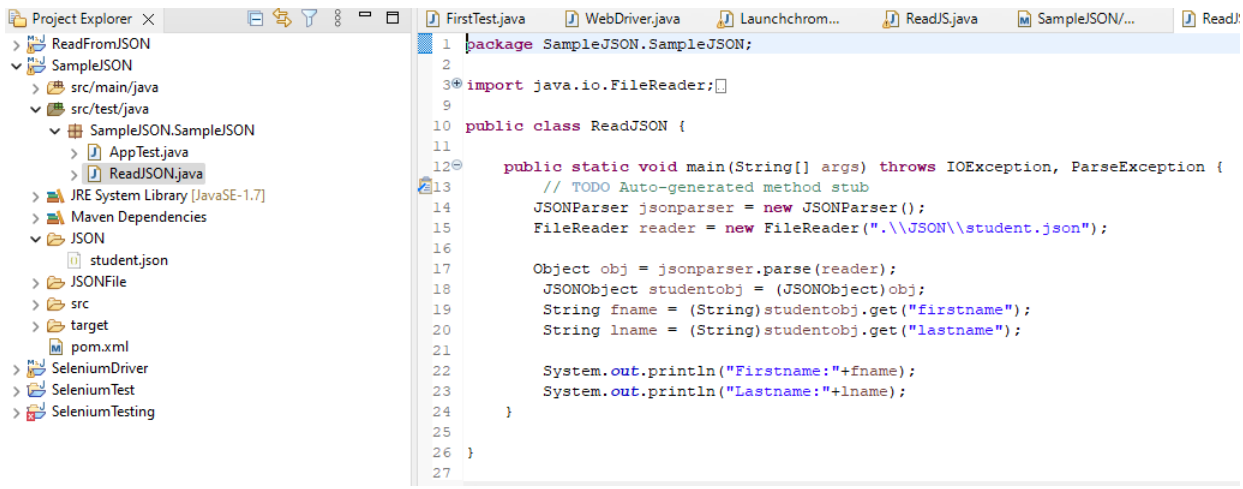
```
<!-- https://mvnrepository.com/artifact/com.googlecode.json-simple/json-simple -->
<dependency>
    <groupId>com.googlecode.json-simple</groupId>
    <artifactId>json-simple</artifactId>
    <version>1.1.1</version>
</dependency>
```

☑ Include comment with link to declaration

Step6: Open POM.xml and add dependency.



Program to read data from JSON. Create a class under src/test/java



Run the program it will fetch data from JSON file

```java
public class ReadJSON {

    public static void main(String[] args) throws IOException, ParseException {
        // TODO Auto-generated method stub
       JSONParser jsonparser = new JSONParser();
       FileReader reader = new FileReader(".\\JSON\\student.json");

       Object obj = jsonparser.parse(reader);
        JSONObject studentobj = (JSONObject)obj;
        String fname = (String)studentobj.get("firstname");
        String lname = (String)studentobj.get("lastname");

        System.out.println("Firstname:"+fname);
        System.out.println("Lastname:"+lname);
    }

}
```

Problems @ Javadoc Declaration Console Coverage

<terminated> ReadJSON [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (14-Jan-2024, 8:36:02 pm – 8:36:02 pm) [pid:

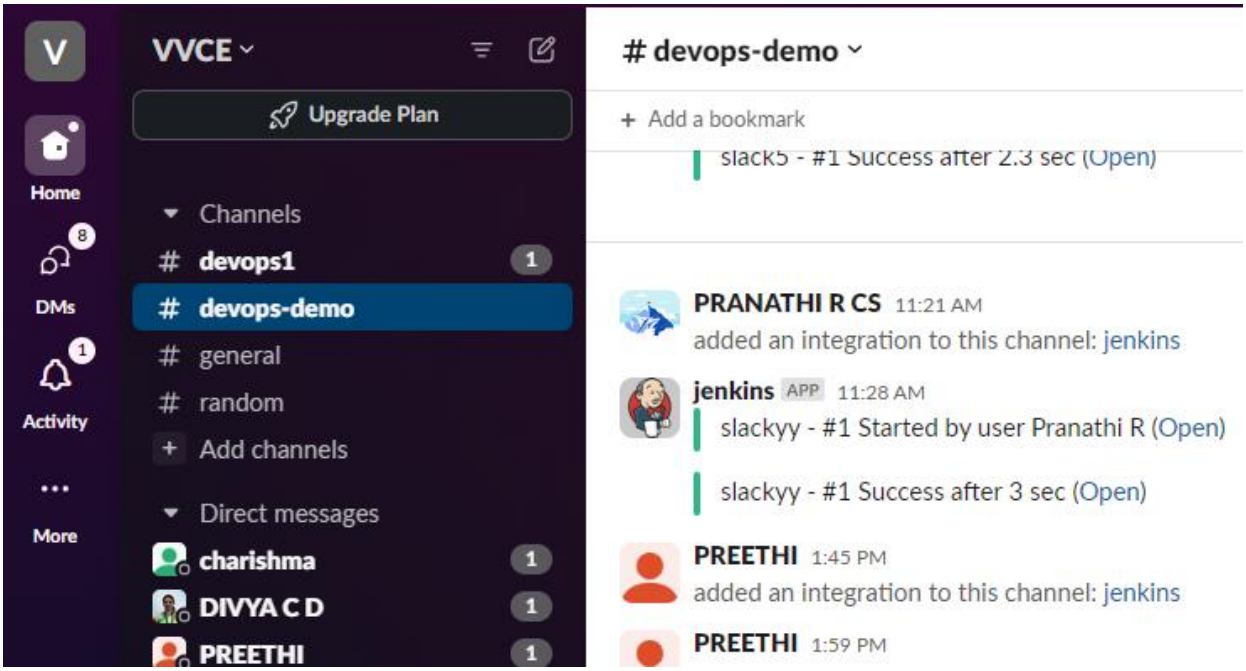
```
Firstname:Anil
Lastname:BH
```

C2. Integrate communication channel with Jenkins for status of project and also enable email notification for a build.

Communication Channel : Slack

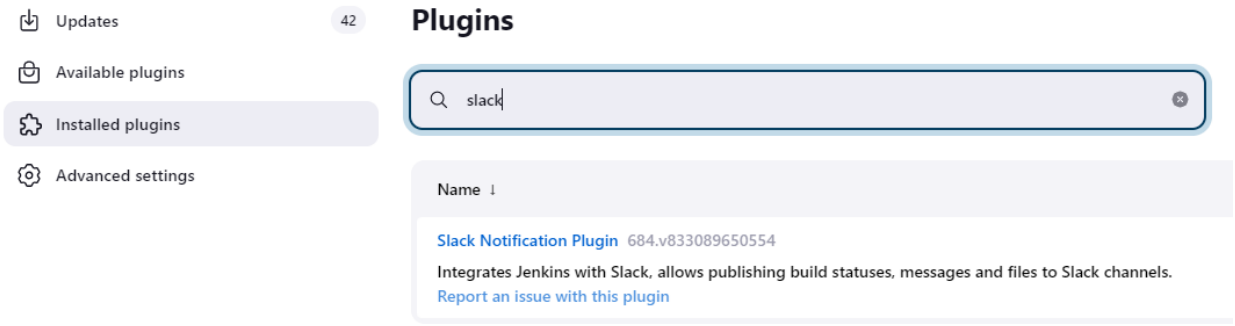
Slack installation :

1. Install slack application. Create account in slack.
2. Create new workgroup or use existing workgroup, add new channel under group.



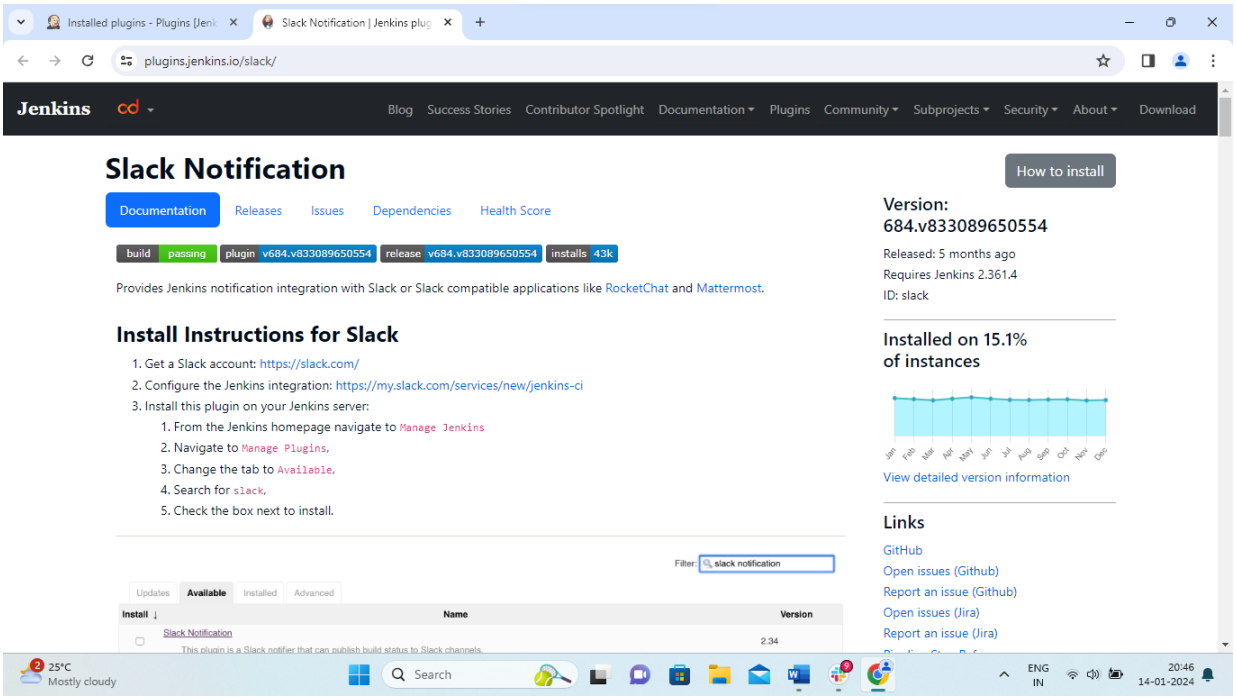
Jenkins configuration:

1. Add slack notifications plug-in in jenkins.

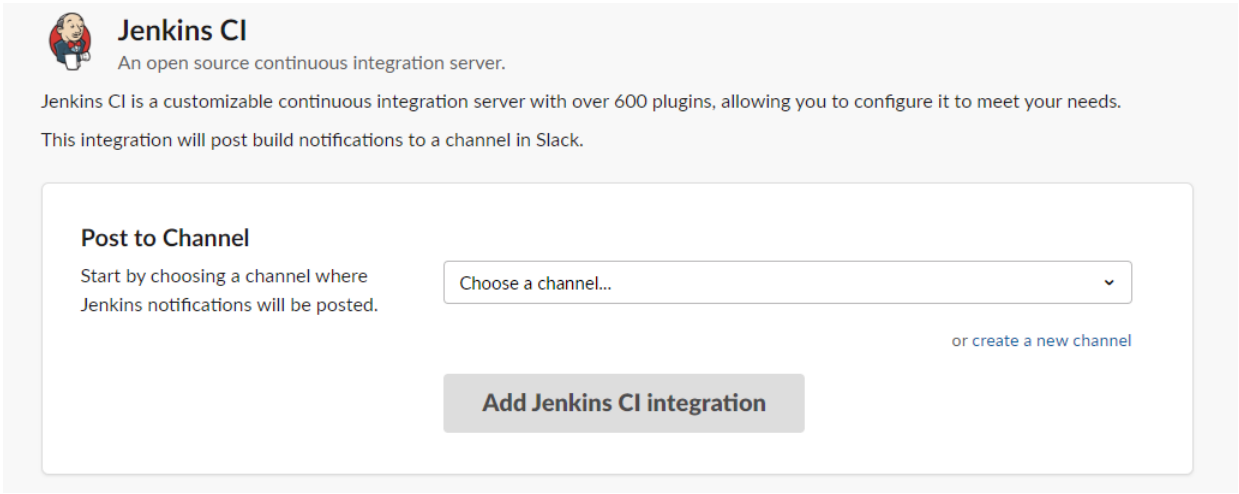


2. Click on installed plug-in

   It will redirect to notifications page



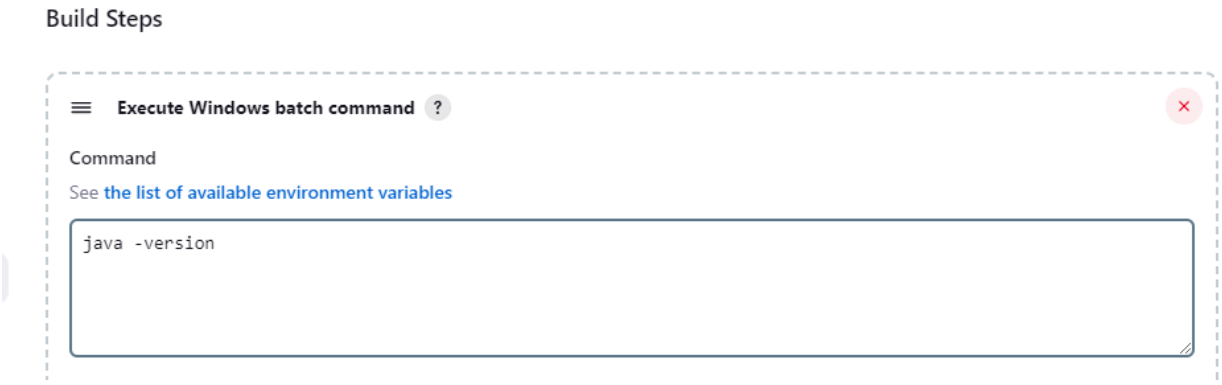
3. click on “Configure the Jenkins integration:” choose posting channel



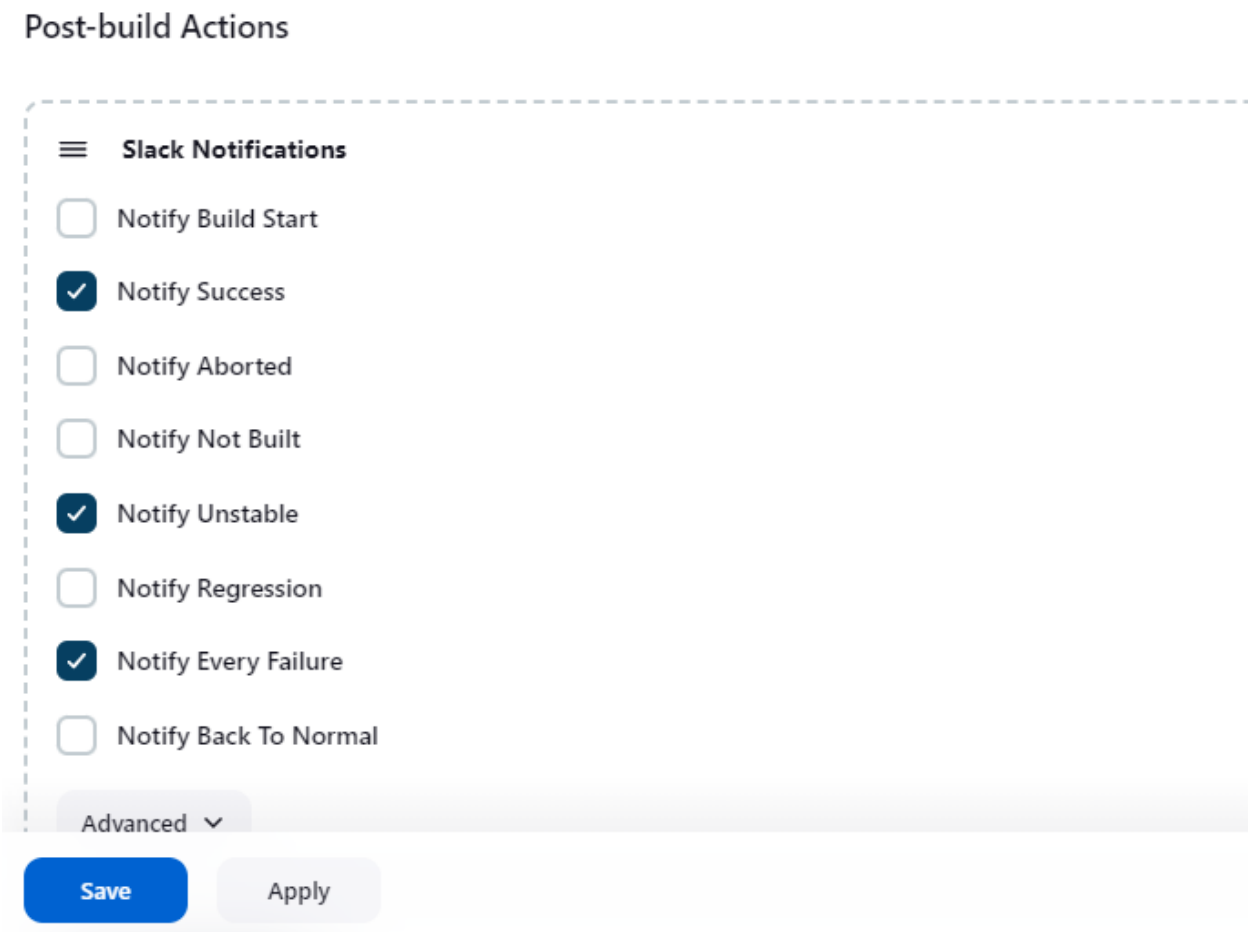
4. after choosing channel it will redirec to next page and note down “**Integration Token Credential ID”**

5. Create free style project.

6. Confiugation of project:

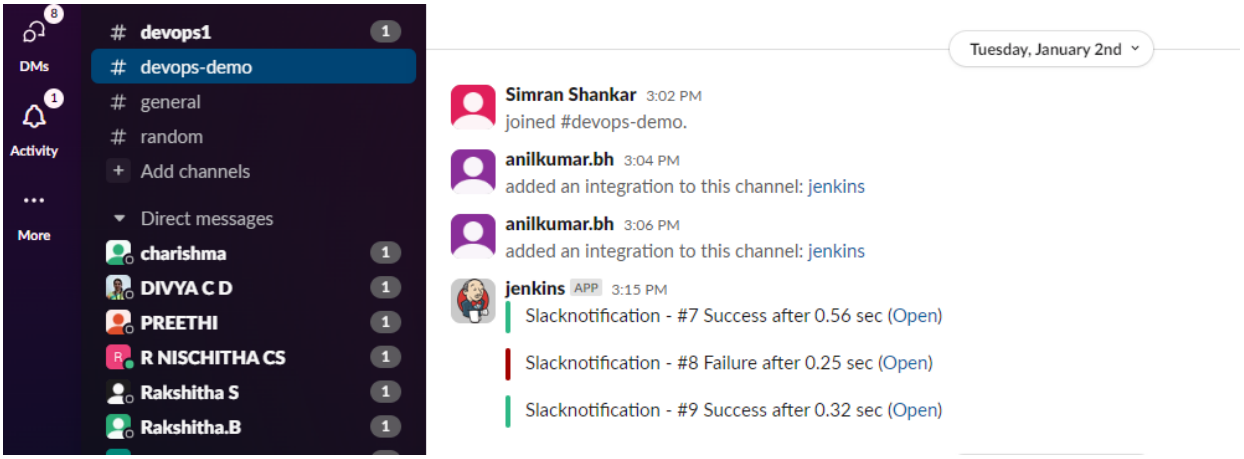


7. Add post build action and choose notifications.



8. Save the configuration and run the build.

9. Notification will be triggered from jenkins. User will get the notification in channel.



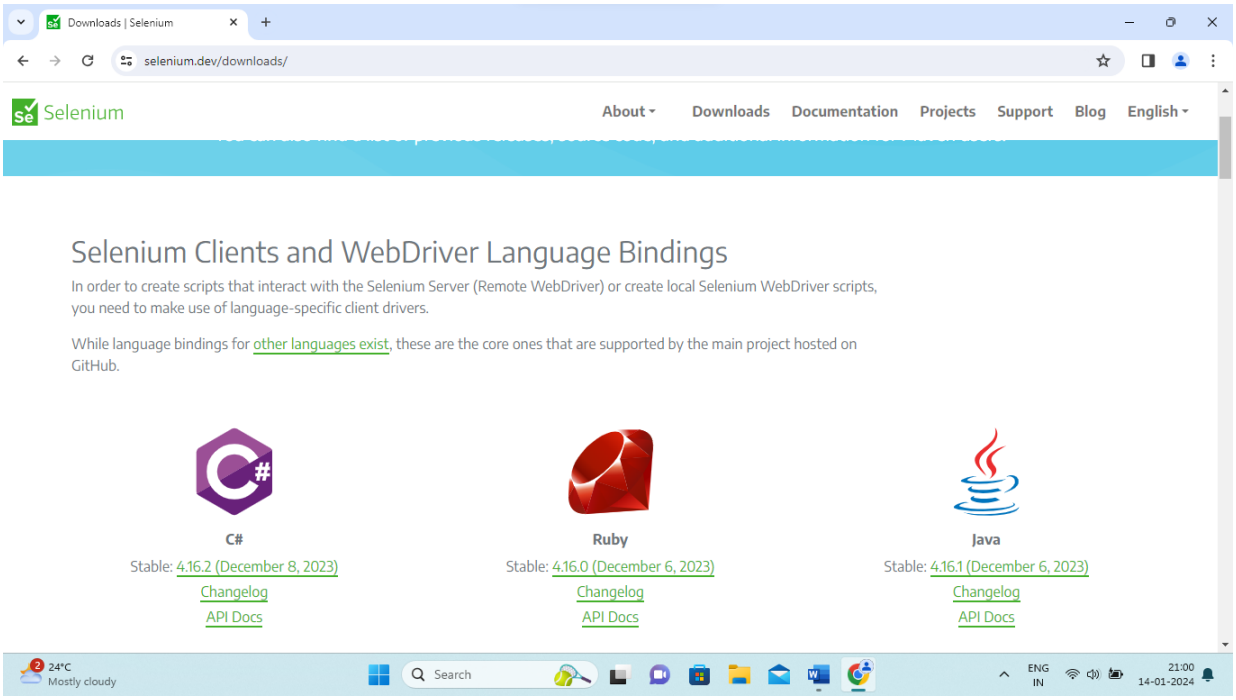
C3. Apply Xpath to locate elements in an application for automation testing in CI/CD pipeline.

Required tools:

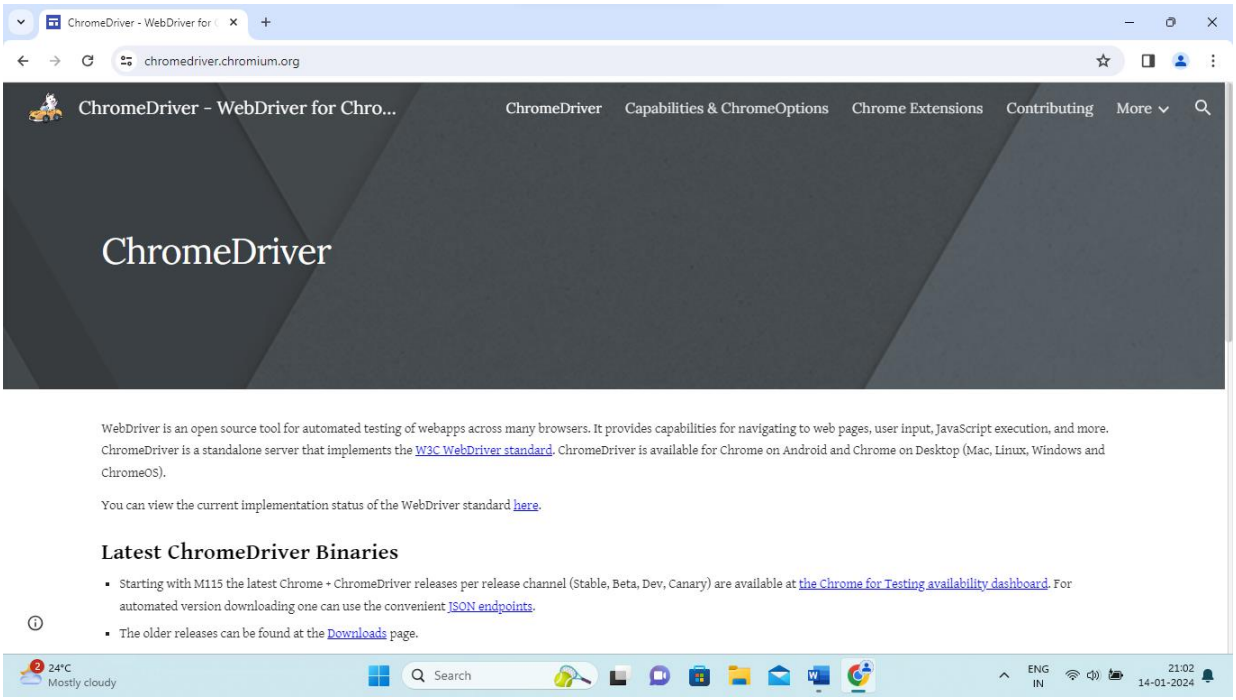
1. Selenium
2. Eclipse IDE
3. Browser for automation.

Procedure:

1. Downloading selenium webdriver, and choose the programming language : java

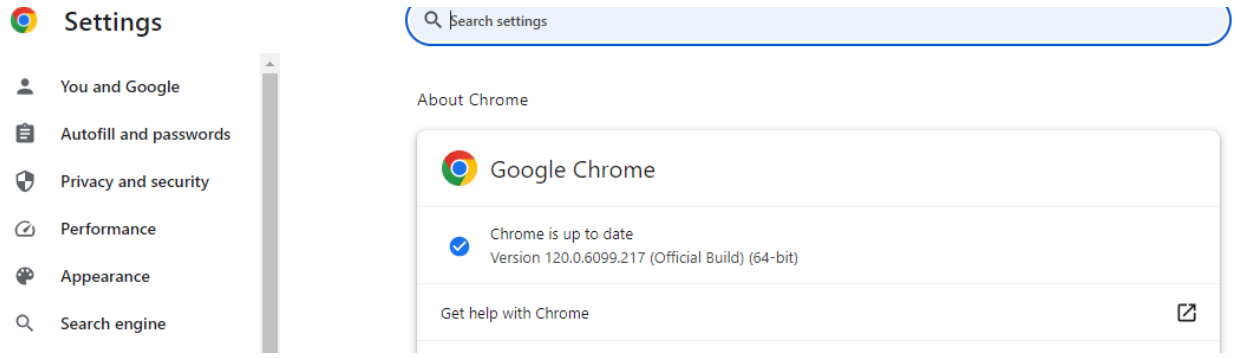


2. Select platform supported by selenium webdriver. Choose appropriate browser: chrome



3. Click on " The older releases can be found at the Downloads page."

4. Find out the version of your browser



Choose webdrive based on version

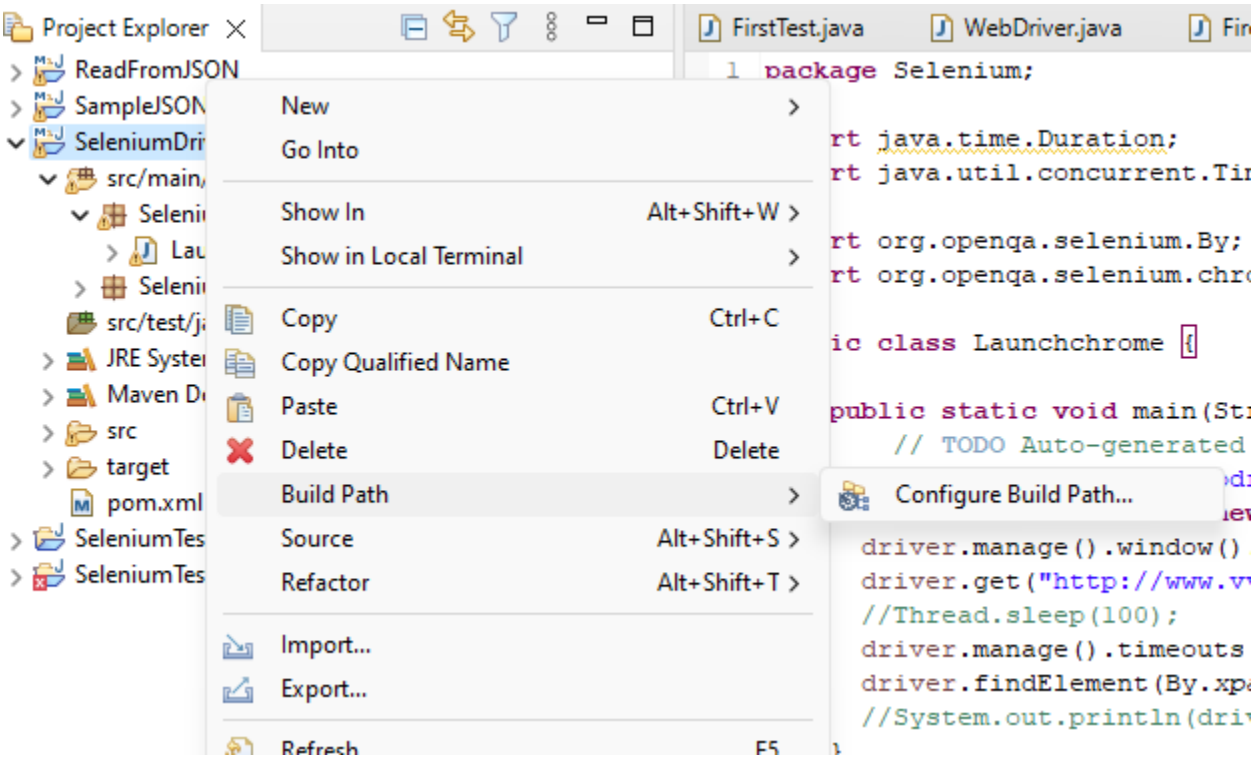
Channel	Version	Revision	Status
Stable	120.0.6099.109	r1217362	✅
Stable (upcoming)	120.0.6099.216	r1217362	❌

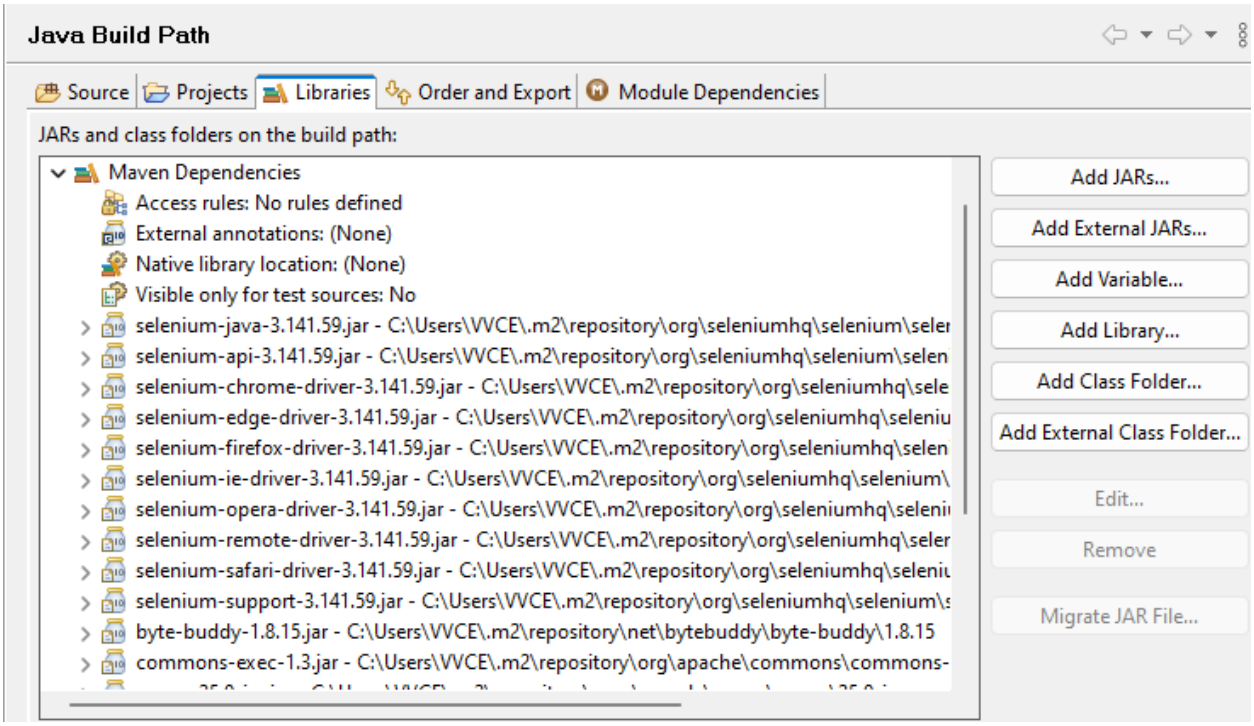
Download the perticular version driver.

Project configuration in Eclipse:

Step1: Create a Java project.

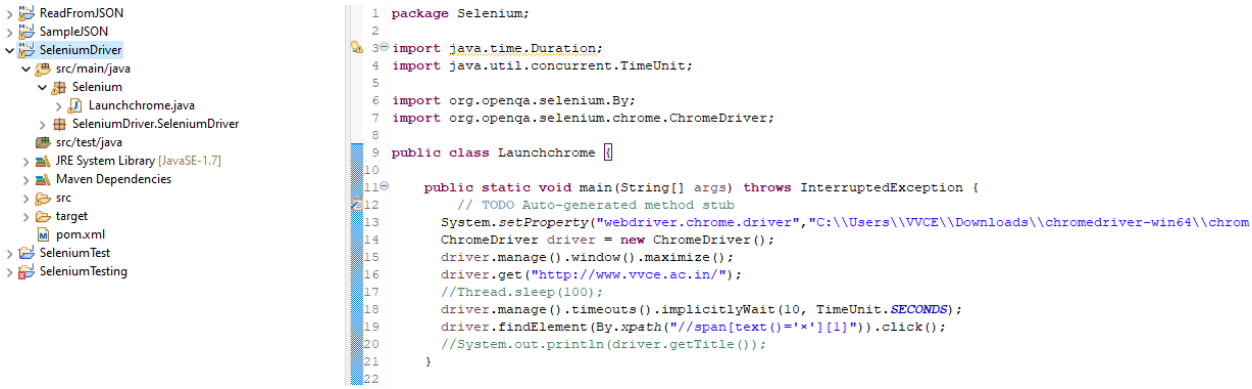
Step2: Add external jar files to project





Step3: Save the configuration.

Step4: Create class under “ src/main/”



Step5: add chromedriver path

Step6: Save the project.

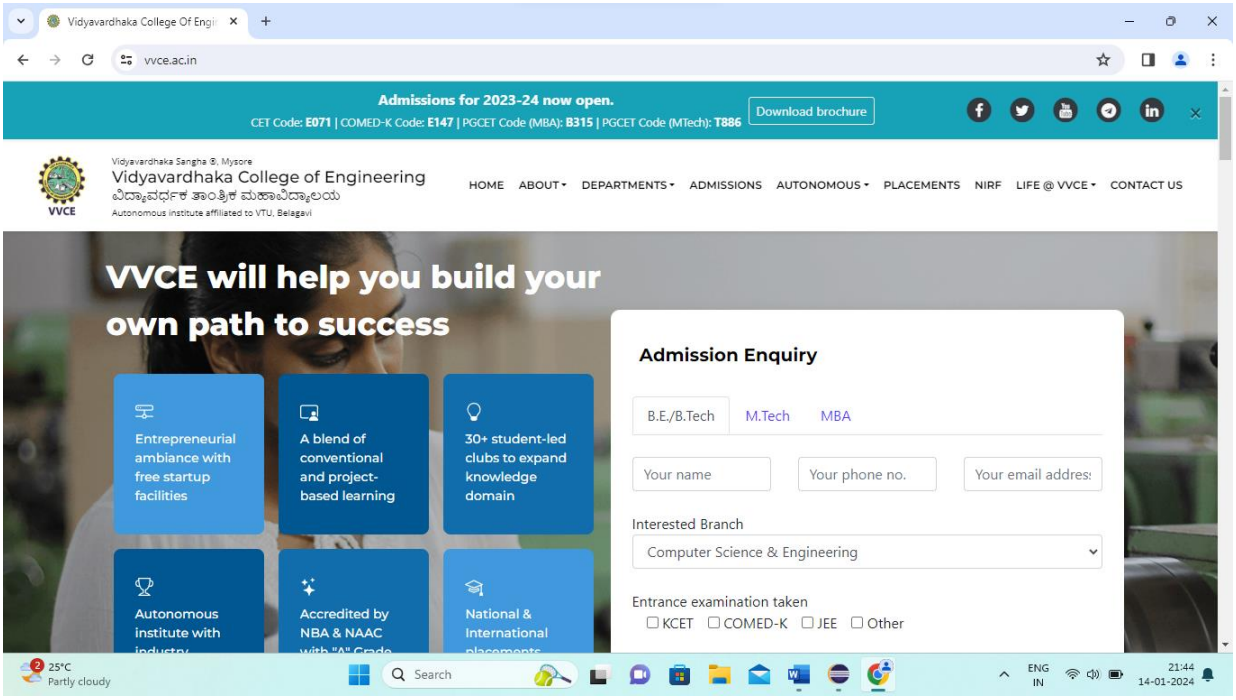
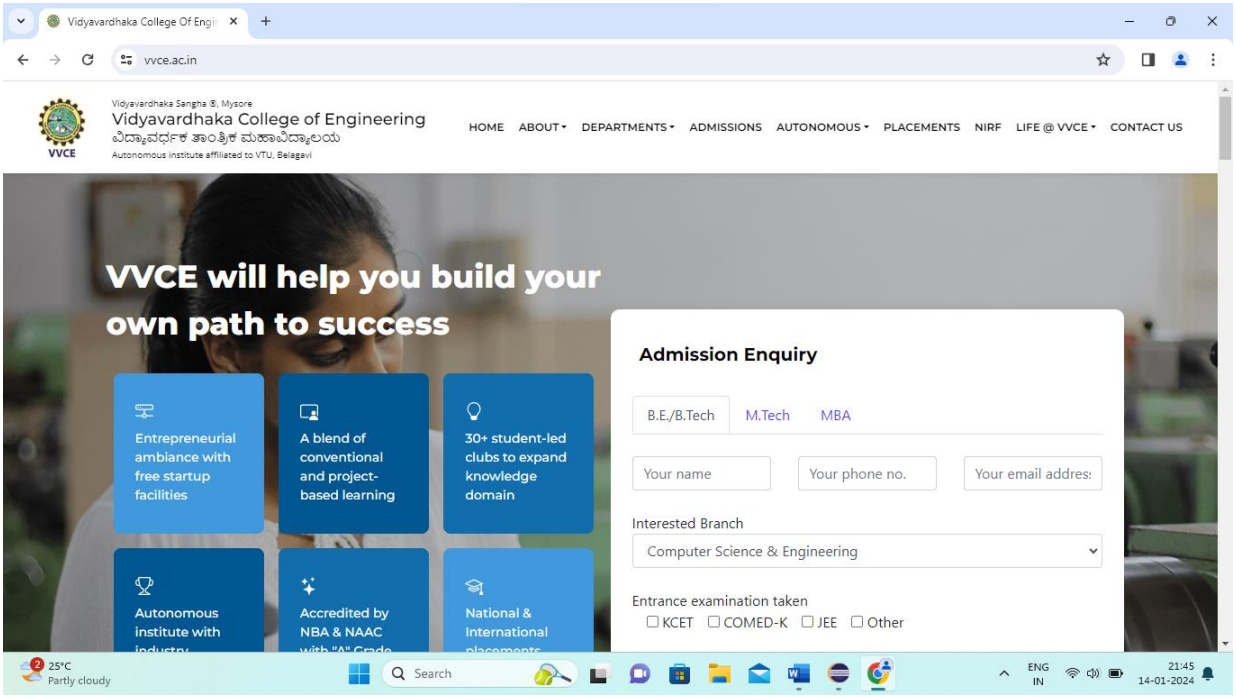
About Xpath:

- Absolute XPath: Begins from the root of the HTML document and specifies the complete path to the element. It's not as flexible and can break if the page structure changes.
- Relative XPath: Starts from a specific element and navigates through the DOM hierarchy to locate the desired element.

Relative Xpath:

//<Tag Name>[Type of element =”value”][indexing]

Example:

//span[text()='x'][1] xpath for close popup.

Step7: Run the application.