Pre-requisites and software requirements

Pre-requisites

Java programming Knowledge: Basic understanding of Java syntax, oops concept and familiarity with building Java applications.

Operating system: Windows, macOS or Linux (64-bit recommended).

Internet connectivity: Reliable Internet connection for downloading software, accessing GitHub and Azure Resources.

GitHub account: Everyone should have a GitHub account for version control and source code management (SCM).

Software requirements

Sl. No.	Software/tool	Version	Purpose	Installation notes
1	Java JDK.	17.0 / above (preferred)	Required for building and running Java based applications	Download from Oracle or open JDK distributions.
2	IntelliJ IDEA	2022.2 / above	Integrated development environment, IDE for Java development and project management.	Download from JetBrains
3	Eclipse for developers	2022-09 / above	Integrated development environment, IDE for Java development and project management.	Download from Eclipse Packages
4	Apache Tomcat server	10 / above	Local server for basic website deployment	Download from Apache Tomcat

5	GIT	Latest stable version	Version control system for managing source code and collaborating via Github	Download from GIT
6	Maven	Latest stable version	Build automation tool for Java projects	Download from Apache Maven
7	Gradle	Latest stable version	Alternative build automation tool for managing dependencies and tasks	Download from Gradle
8	Jenkins	Latest stable version	CI-CD tool for automating builds, tests and deployments	Download from Jenkins
9	Oracle VM virtual box.	Latest stable version	Virtualization tool for creating and managing virtual machines	Download from Virtualbox
10	Vagrant	Latest stable version	Tool for managing and provisioning virtual machine environments.	Download from vagrant
11	Ansible	Latest stable version	Configuration management and automation tool	Install via ansible documentation
12	Microsoft Azure	Free Tier account	Needed for DevOps CI-CD pipelines and deployments	Install via Microsoft Azure sign up

Experiment 1

Introduction to DevOps terminology definition and concepts, Understanding version control tool - Git, Overview of Build Automation tools, Key differences between Maven and Gradle.

1. What is DevOps?

DevOps is a set of cultural philosophies, practices, and tools that combine software development (Dev) and IT operations (Ops). Its goal is to shorten the systems development life cycle while delivering features, fixes, and updates frequently in close alignment with business objectives. DevOps promotes:

Collaboration: Breaking down silos between development and operations teams.

Automation: Automating repetitive tasks (builds, tests, deployments) to improve efficiency.

Continuous Integration/Delivery (CI/CD): Regularly integrating code changes and deploying them to production with minimal manual intervention.

Monitoring and Feedback: Constantly monitoring the performance and behavior of applications in production to rapidly address issues.

2. Why is DevOps Used?

Speed and Agility: Faster development cycles and quicker time-to-market.

Quality: Automated testing and integration help catch issues early.

Reliability: Frequent, smaller updates reduce the risk of large-scale failures.

Efficiency: Automation reduces manual errors and repetitive tasks.

Scalability: Processes and infrastructure can grow with the business.

3. Examples of DevOps in Action.

Continuous Integration (CI): Tools like Jenkins or Azure Pipelines automatically build and test code on every commit.

Continuous Deployment (CD): Systems automatically deploy tested code to production environments.

Monitoring: Tools such as Prometheus, Grafana, or New Relic continuously monitor application performance.

4. Understanding software development life cycle (SDLC).

Before diving into build tools like Maven and Gradle, it is essential to understand how software has traditionally been developed. The software development life cycle provides a structured approach to build software efficiently. One of the earliest models of SDLC is the waterfall model, which follows a linear and sequential approach, where each phase

must be fully completed before moving to the next phase. However, this model has several limitations which led to the adoption of agile and DevOps methodologies.

5. Drawbacks of Waterfall model

- Rigid & Inflexible – Changes cannot be made once a phase is completed.
- Late Bug Detection – Testing happens after coding, making bug fixes costly.
- Slow Development Process – Each phase must be completed before the next starts.
- Not Suitable for Modern Agile Environments – Does not support rapid iteration.

6. Introduction to agile methodology

To overcome Waterfall's limitations, the Agile Model was introduced. Agile promotes iterative and incremental development, allowing teams to deliver software faster with continuous feedback.

a) Key Agile Concepts

b) Sprints – Development happens in short cycles, usually lasting 7-14 days.

c) Scrum Master – Facilitates Agile processes and removes blockers for the team.

d) Daily Stand-up Meetings – Short meetings where team members discuss:

What they did yesterday?

What they plan to do today?

Any blockers?

e) Retrospective Meetings – At the end of each sprint, the team reflects on what worked well and what can be improved.

7. Transition from agile to DevOps

While agile improved development, deployment and operations remained slow This led to the birth of DevOps, which introduced:

- ► Continuous Integration and Deployment (CI/CD).
- ► Automation of Builds, Testing and Deployment.
- ► Faster and more reliable software releases.

Note: To make a DevOps sufficient, we need build tools like Meghan and Gradle. It simplifies project management.

8. Problems with manual Approach

- Time consuming - Downloading, adding and adding dependencies manually is inefficient.
- Error prone - Missing jar files or incorrect configurations can break the project
- Difficult to manage - Dependencies are not automatically updated.
- Not scalable - Every team member must manually configure their setup.

9. What is version control?

Version control also known as source control is a system that tracks and manages changes to files, especially code overtime following teams to collaborate efficiently and revert to previous version versions if needed. Before moving to Magan and Gradle, we must first understand version control systems. Like GIT partial control plays a crucial role in devops enabling

- ✓ efficient code management - track changes, revert to previous versions and collaborate seamlessly.
- ✓ Team collaboration - multiple developers can work on the same project without conflicts.
- ✓ Integration with CI/CD pipelines, automate spools, testing and deployment.

10. Setting Up Git on Local System

Step1: Check if get is already installed.

cmd prompt → git --version

Step2: If not installed download Git installer from https://git-scm.com/

Step3: Run installer and install git

Create GitHub Account

Step 1: Go to https://github.com/ and sign up for a new account
Step 2: Login to GitHub
Step 3: Create a new Repository (Private or Public)

Basic Git Commands

Step1: Create a new folder and open git bash and go to the folder location.

Step2: Run the Commands for git configuration

git config --global user.email " sampleGitHub@email.com"
git config --global user.name " sampleGitHub_username"

Step3: Initialize git using this command.

git init

Step4: Add some files in the folder by using this command.

git touch <filename1.txt>
git touch <filename2.html>
git touch <filename3.py>
git touch <filename4.js>

Step5: Run these commands to check status at files and commit your changes or updates.

git status [To check status]
git add <filename> [To add a particular file]
git add . [To add all the files]
git commit -m " Commit Message" [To commit all your changes]

git remote add origin “ github-url“ [To add remote repo link to local repo]
git push -u origin master [To push your local changes to remote]

11. Why do we need Build Tools (Maven & Gradle)?

- Automatically manage dependencies – No need to download JARs manually.
- Simplify project configuration – A single configuration file (pom.xml for Maven, build.gradle.kts for Gradle) handles everything.
- Enable easy build & testing – Run tests and package applications using simple commands.
- Ensure consistency – The same project setup works on different machines.

Note: Both Maven and Gradle are used to automate the build process and manage project dependencies. Here’ s why they are so popular in the DevOps ecosystem:

- Automated Build and Testing: They allow developers to compile code, run tests, and package applications without manual intervention.
- Consistent Build Environment: By enforcing standardized project structures and dependency management, they help avoid the “ it works on my machine” problem.
- Ease of Integration: Both tools integrate well with Continuous Integration (CI) servers like Jenkins, enabling automated pipelines.
- Dependency Resolution: They simplify the process of managing external libraries and ensure that all developers are using the same versions.

12. What is Maven?

Maven is a build automation and project management tool primarily used for Java projects. It uses a central configuration file known as the POM (Project Object Model),

written in XML, which defines the project structure, its dependencies, build order, and plugins.

Features

- Convention over Configuration: Maven enforces a standard directory structure (e.g.,src/main/java, src/test/java), which simplifies project setup.
- Dependency Management: Automatically downloads and manages external libraries and dependencies from repositories like Maven Central.
- Build Lifecycle: Defines a fixed lifecycle (e.g., compile, test, package, install, deploy) which standardizes the build process.
- Plugin Ecosystem: Provides a rich set of plugins to extend functionality (e.g., unit testing, code coverage, reporting).

13. What is Gradle?

Gradle is a modern build automation tool that is known for its flexibility and performance. It uses a Groovy or Kotlin DSL (Domain Specific Language) to define build logic, which allows for more dynamic and customizable configurations compared to Maven’ s XML-based approach.

Features

- Flexible Build Scripts: Instead of a rigid XML file, Gradle build scripts written in Groovy or Kotlin allow you to include logic, conditionals, and loops.
- Incremental Builds: Gradle tracks changes in source files and only rebuilds what is necessary, which can significantly speed up the build process.
- Multi-project Builds: Easily manages complex projects with multiple modules or subprojects.
- Extensibility: A robust plugin system that enables you to integrate various languages, frameworks, and tools.
- Parallel Execution: Can run tasks in parallel, optimizing build times for large projects.

14. Differences between Maven and Gradle

Aspect	Maven	Gradle
Build Script Language	Uses an XML-based configuration file (pom.xml).	Uses a DSL based on Groovy or Kotlin (build.gradle or build.gradle.kts).
Configuration Style	Declarative and rigid – follows strict conventions (convention over configuration).	Flexible and dynamic – allows you to write custom logic and conditions within the build script.
Build Lifecycle	Provides a fixed lifecycle (e.g., validate, compile, test, package, install, deploy).	Uses a task-based approach where tasks can be defined, customized, and linked in a flexible manner.
Dependency Management	Manages dependencies through the POM file; downloads them from central repositories (e.g., Maven Central).	Similar dependency management; supports dynamic version resolution and customizable dependency configurations.
Performance	Generally slower for large projects because of the fixed build lifecycle and less emphasis on incremental builds.	Often faster, thanks to incremental builds, caching, and parallel execution of tasks.

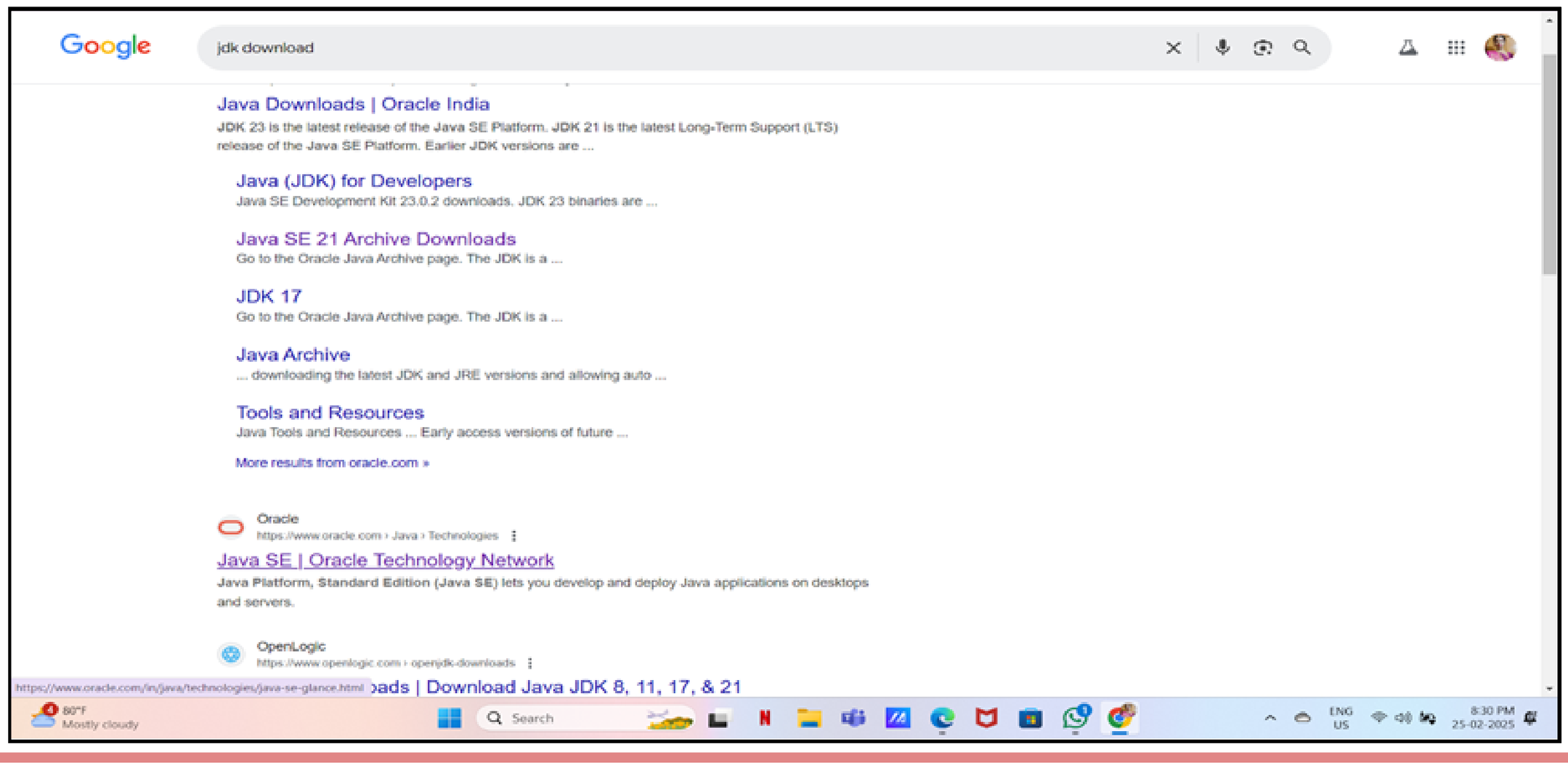
Extensibility & Plugins	Rich ecosystem of plugins but customization can be more challenging due to XML's verbosity and limitations in scripting.	Highly extensible through its scripting capabilities; writing custom tasks or plugins in Groovy/Kotlin is more straightforward.
Multi-Project Builds	Handles multi-module projects well but requires a strict directory layout and a parent POM for aggregating modules.	Excels in multi-project builds with simple configuration, allowing each subproject to be configured in a flexible manner.
Learning Curve	Easier for beginners due to its structured, convention-based approach but can become complex with large configurations.	More flexible but may have a steeper learning curve initially if you need to leverage its dynamic features and custom logic.
Community & Maturity	Has been around longer, so many legacy projects and extensive documentation exist.	Relatively newer; it has gained popularity due to its modern features, especially in Android and multi-language projects.
Integration with CI/CD	Well-supported by most CI/CD tools (like Jenkins, Azure Pipelines) with stable, predictable behavior.	Also integrates seamlessly with CI/CD tools and is often chosen for its faster build times and flexibility in pipeline scripting.

Experiment 2

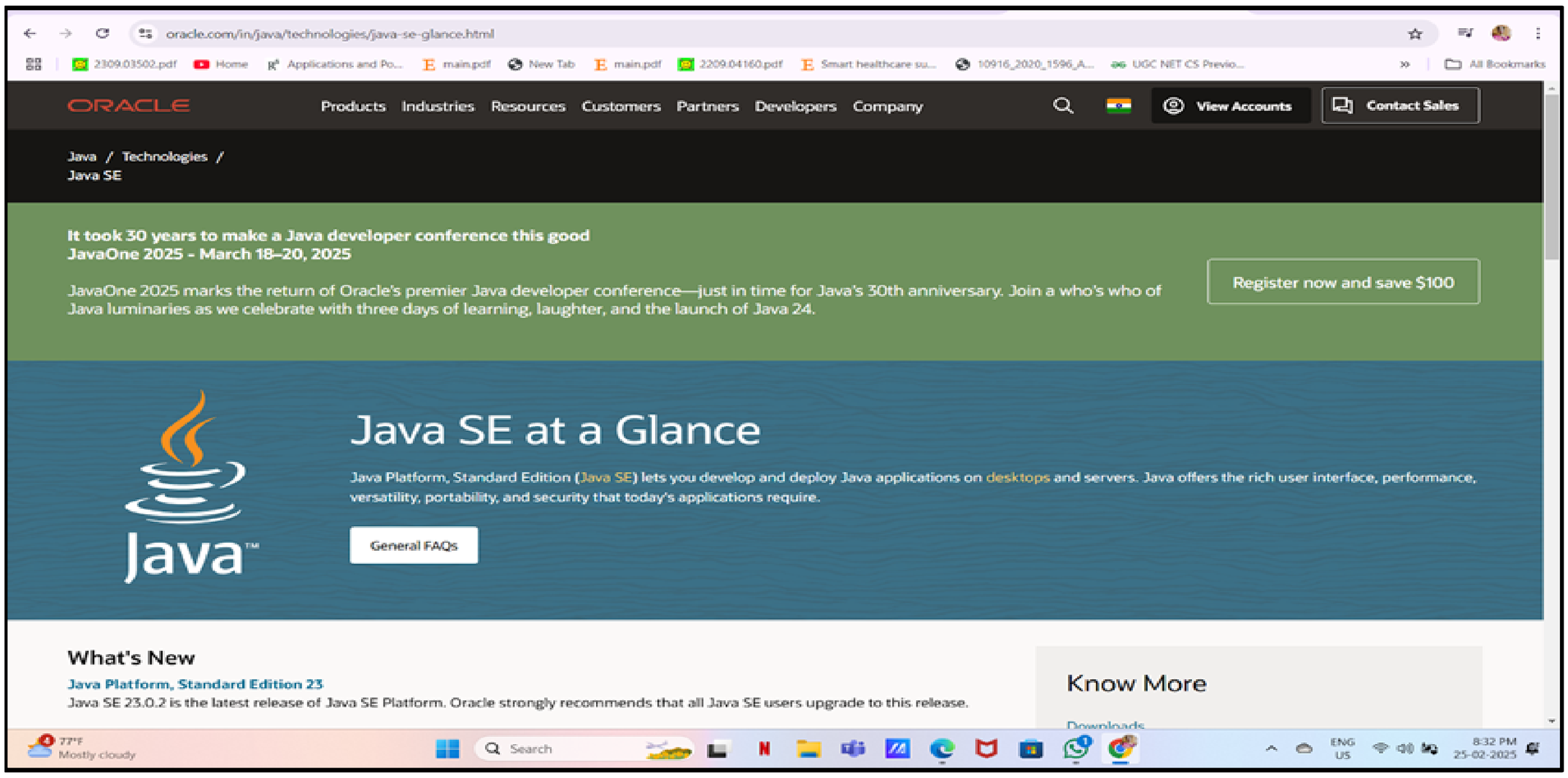
Installation and Setup of JDK-17, IDE (IntelliJ Idea), GIT, Maven, Gradle and Jenkins.

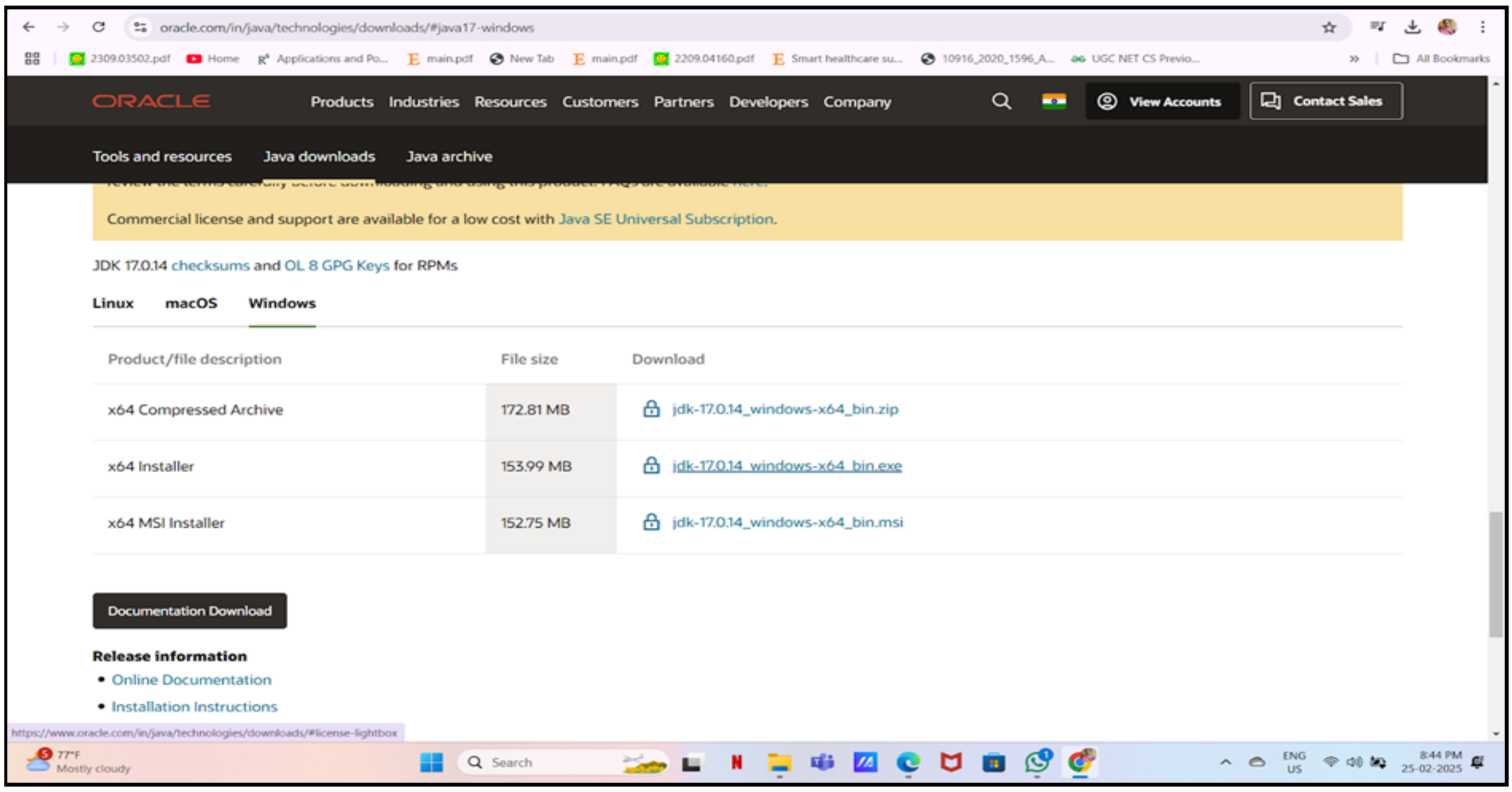
1. JDK Installation steps

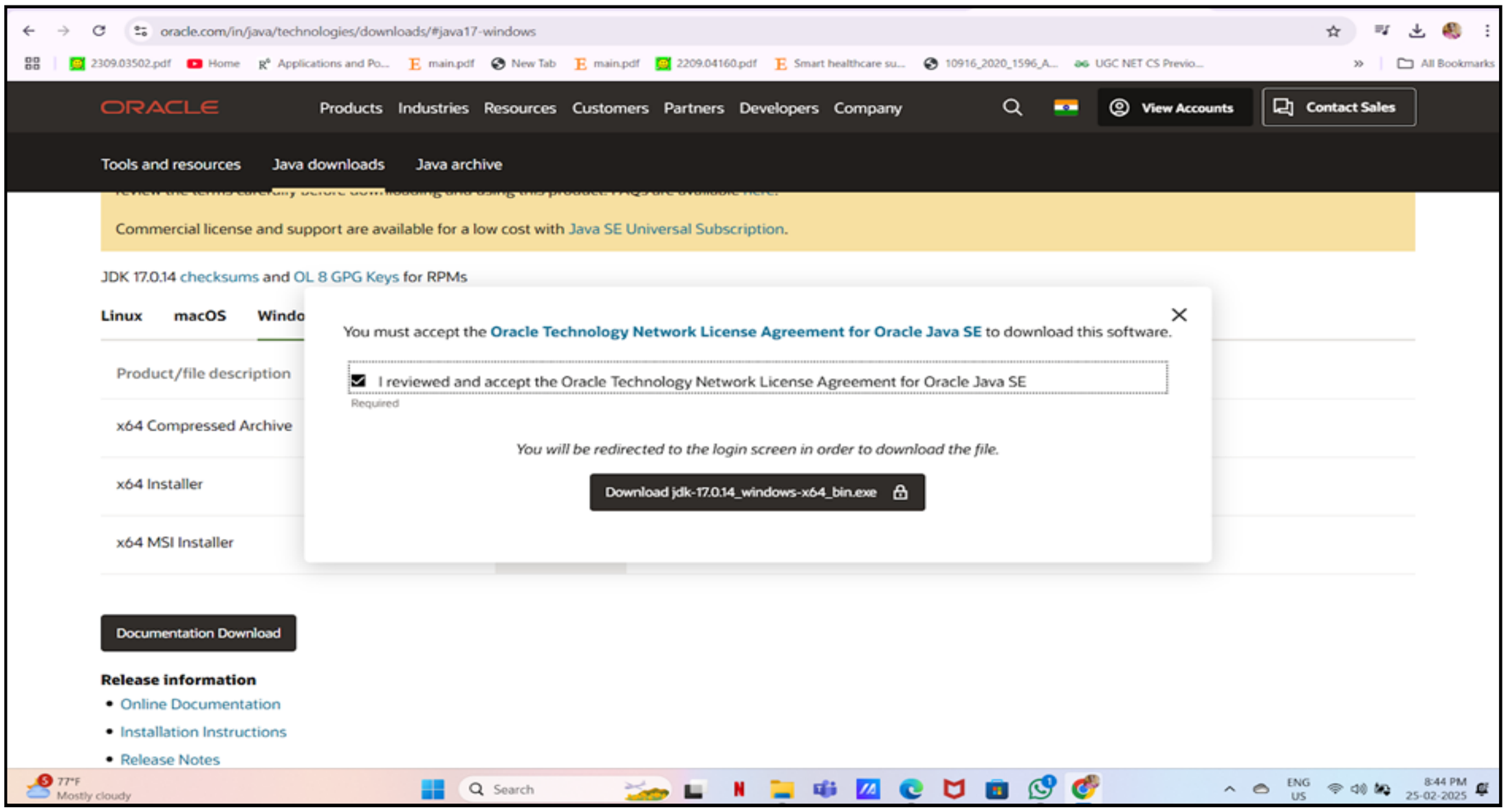
Goto — > Browser — > JDK download → Select-JAVA SE(Standard edition)



→


oracle.com/in/java/technologies/java-se-glance.html
ORACLE
Products Industries Resources Customers Partners Developers Company
View Accounts
Contact Sales
Java / Technologies / Java SE
It took 30 years to make a Java developer conference this good
JavaOne 2025 - March 18–20, 2025
JavaOne 2025 marks the return of Oracle's premier Java developer conference—just in time for Java's 30th anniversary. Join a who's who of Java luminaries as we celebrate with three days of learning, laughter, and the launch of Java 24.
Register now and save $100
Java SE at a Glance
Java Platform, Standard Edition (Java SE) lets you develop and deploy Java applications on desktops and servers. Java offers the rich user interface, performance, versatility, portability, and security that today's applications require.
General FAQs
What's New
Java Platform, Standard Edition 23
Java SE 23.0.2 is the latest release of Java SE Platform. Oracle strongly recommends that all Java SE users upgrade to this release.
Know More

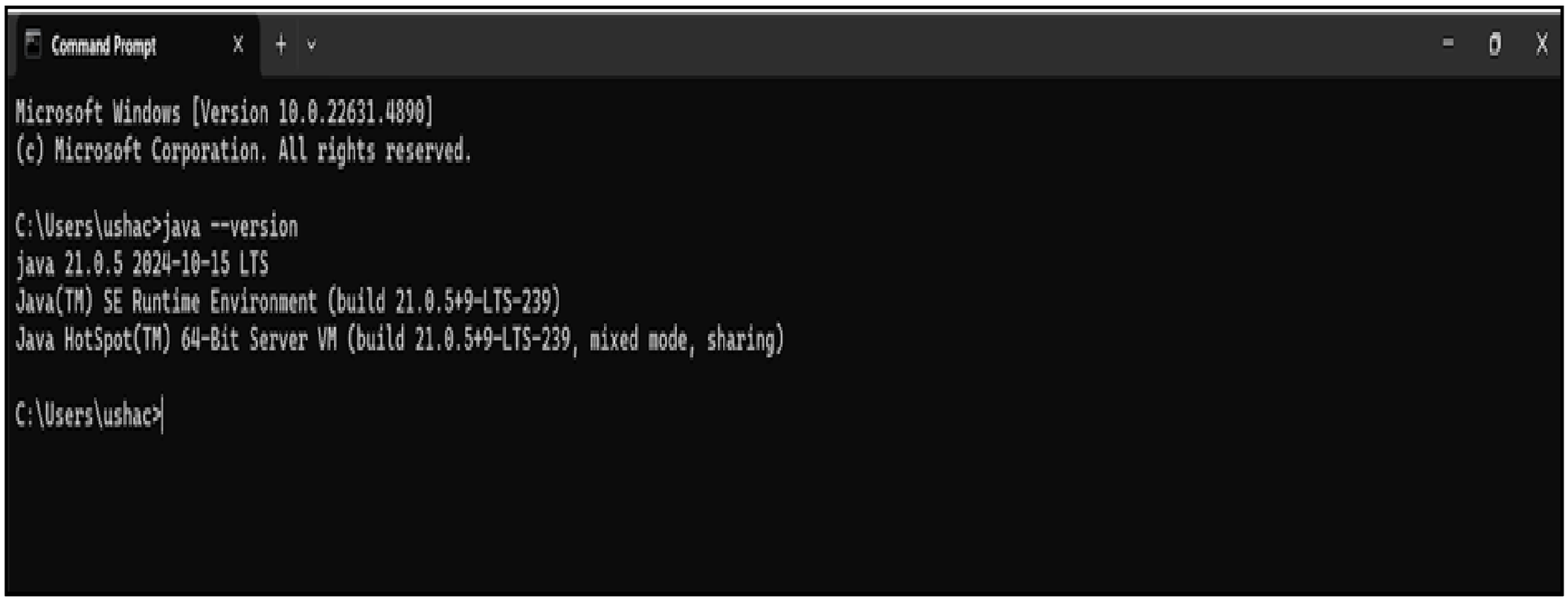

oracle.com/in/java/technologies/downloads/#java17-windows
ORACLE
Products Industries Resources Customers Partners Developers Company
View Accounts
Contact Sales
Tools and resources
Java downloads
Java archive
Commercial license and support are available for a low cost with Java SE Universal Subscription.
JDK 17.0.14 checksums and OL 8 GPG Keys for RPMs
Linux macOS Windows
Product/file description | File size | Download
x64 Compressed Archive | 172.81 MB | jdk-17.0.14_windows-x64_bin.zip
x64 Installer | 153.99 MB | jdk-17.0.14_windows-x64_bin.exe
x64 MSI Installer | 152.75 MB | jdk-17.0.14_windows-x64_bin.msi
Documentation Download
Release information
Online Documentation
Installation Instructions



JDK (java development kit) with the help of which we have compiler, JVM-convert java source code to byte code (with the help of which we can convert java program and make it compatible to any OS).

Check in -

cmd prompt → type java - - version , if installed shows the details of JDK

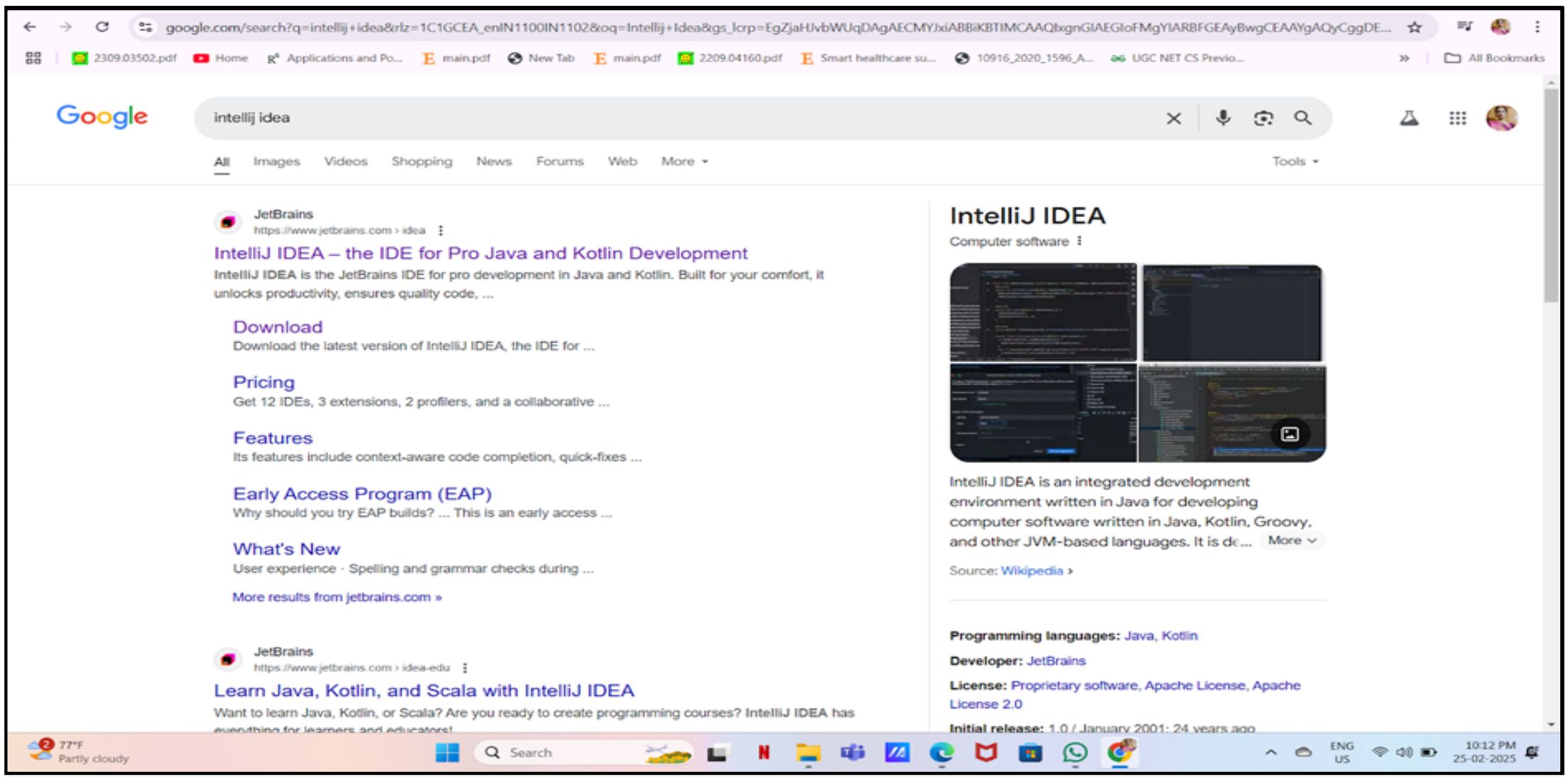


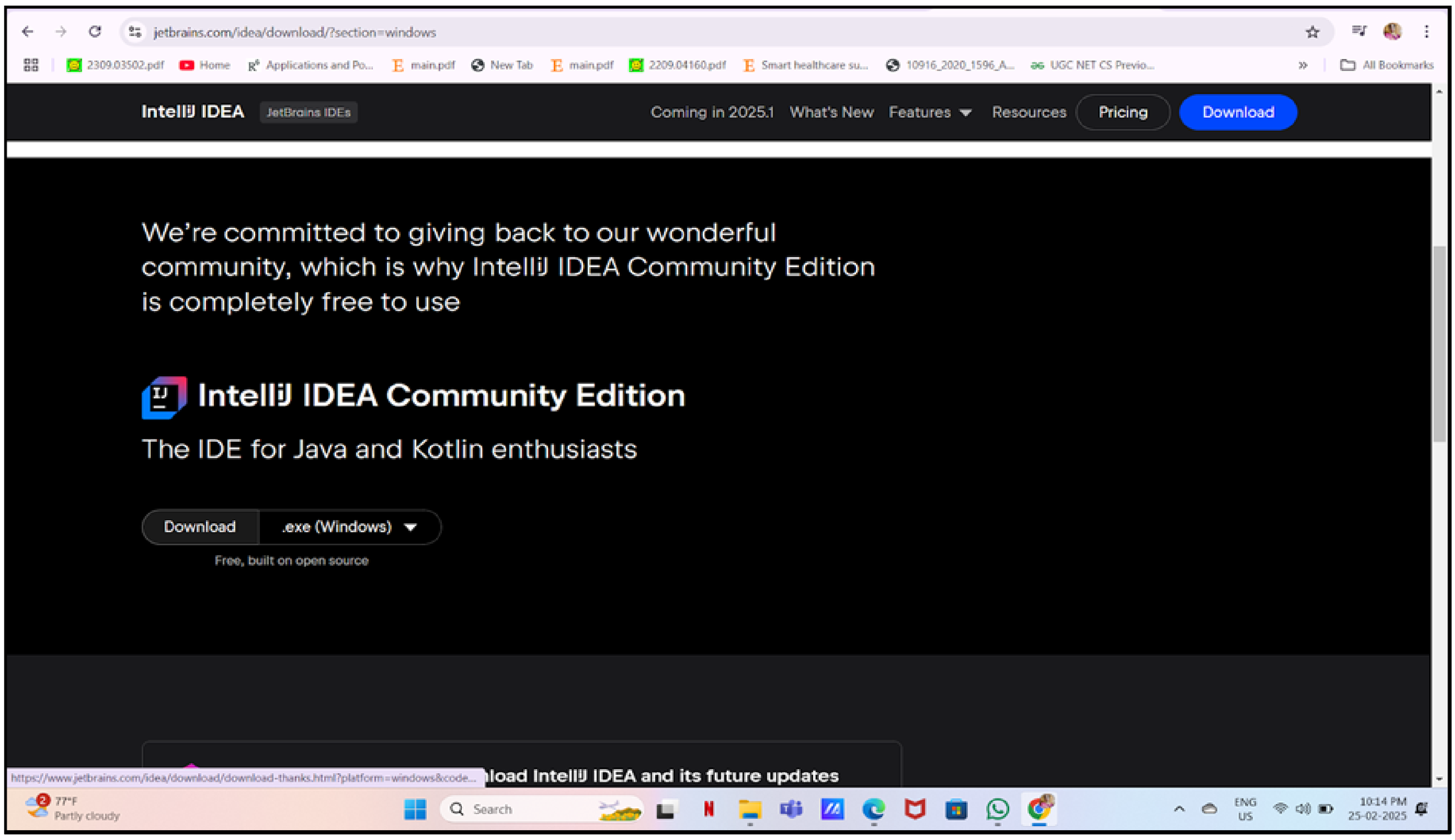
2. IDE (Integrated development Environment) used to write and execute JAVA programs.

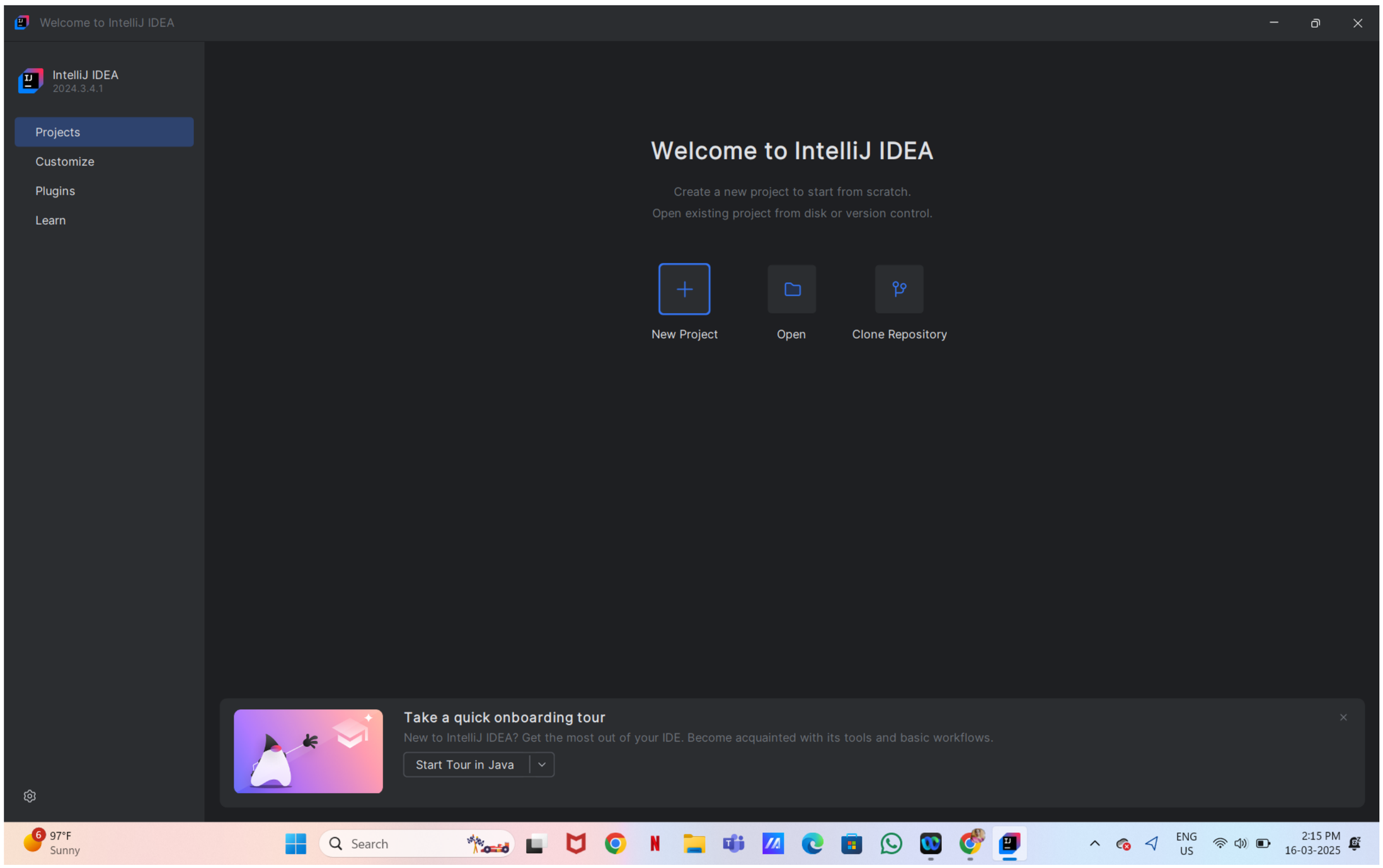
Types of IDE –

Eclipse, NetBeans, IntelliJ Idea (best totally dedicated for JAVA).

Installation of IntelliJ Idea




jetbrains.com/idea/download/?section=windows
IntelliJ IDEA
JetBrains IDEs
Coming in 2025.1
What's New
Features
Resources
Pricing
Download
We're committed to giving back to our wonderful community, which is why IntelliJ IDEA Community Edition is completely free to use
IntelliJ IDEA Community Edition
The IDE for Java and Kotlin enthusiasts
Download
.exe (Windows)
Free, built on open source
https://www.jetbrains.com/idea/download/download-thanks.html?platform=windows&code...
load IntelliJ IDEA and its future updates
77°F
Partly cloudy
Search
ENG
US
10:14 PM
25-02-2025


Welcome to IntelliJ IDEA
IntelliJ IDEA
2024.3.4.1
Projects
Customize
Plugins
Learn
Welcome to IntelliJ IDEA
Create a new project to start from scratch.
Open existing project from disk or version control.
New Project
Open
Clone Repository
Take a quick onboarding tour
New to IntelliJ IDEA? Get the most out of your IDE. Become acquainted with its tools and basic workflows.
Start Tour in Java
97°F
Sunny
Search
ENG
US
2:15 PM
16-03-2025

3. Anatomy of JAVA Program

- Smallest building block in a java program is **Function.**
- What is a function?
    - Function is a block of code that performs a specific task.
    - For example, if a project is given to a team, then each team member performs a specific task similarly function do the task assigned to it.
    - Declaration of a function

```
Return Type Name () {
… … …
}
```

Some functions return values like number, date, time.
Some functions return nothing called **void**

```
Void Name (){
… … .
}
```

Here **Name** of the function should be meaningful like **sendEmail(), saveMessage()**.
We have **Parentheses ()** where we write **Arguments or Parameters**.
Within curly braces {} we write actual java code .

- Every java program should have at least one main() function, it is the entry point of a java program. Functions cannot exist on their own, It belongs to a class, class is a container that has one or more related functions.
- A class is basically used to organize our code.
- Every Java program should have at least one class that is - main () class.

```
Class Main()
{
Void main()
{
… ..
}
}
```

Here void main() is called a **method**.

- In JAVA, **function** is a part of the class called a **method** but in **PYTHON** we can have a method outside the class called a **function.**

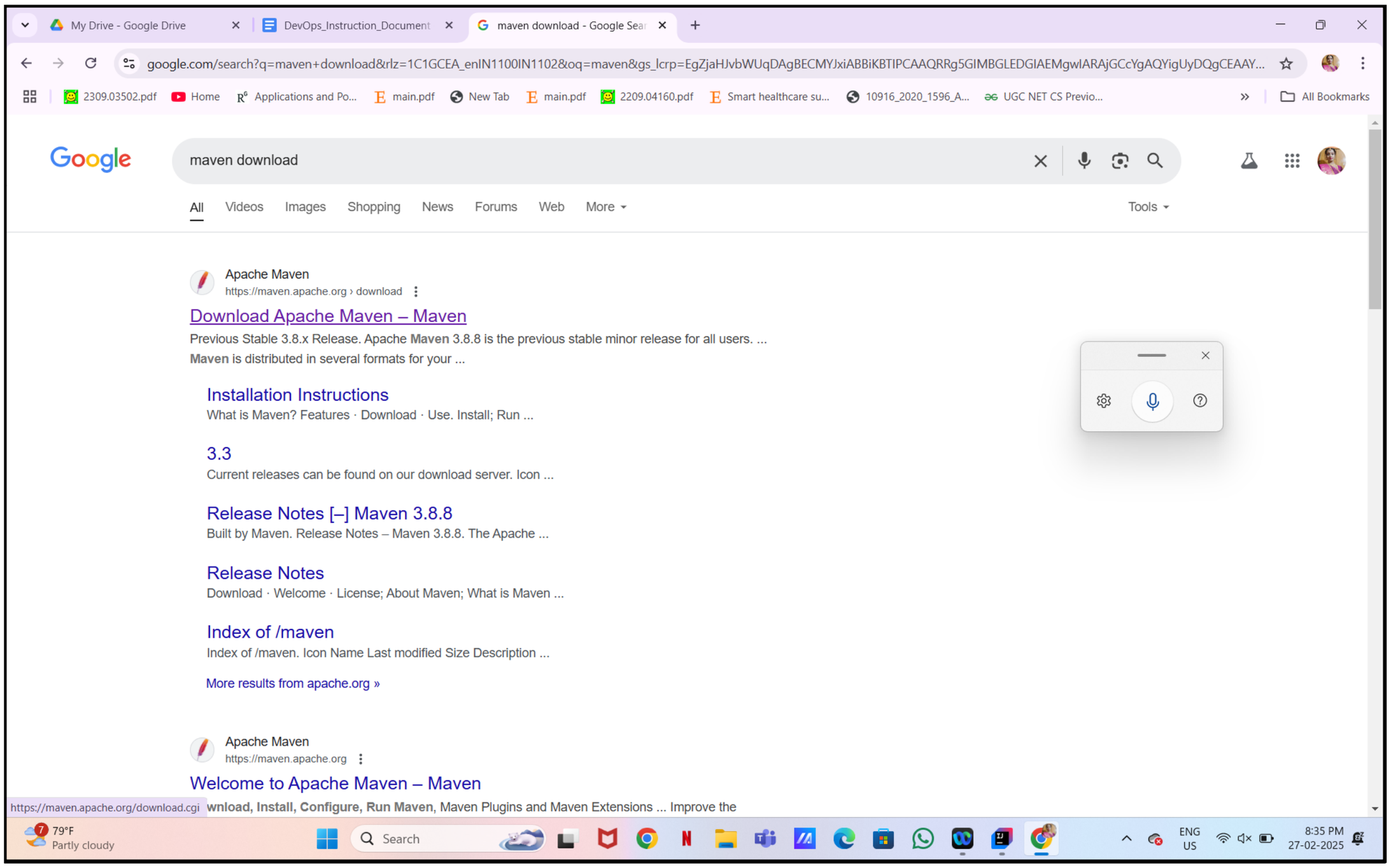
- In java every class and method should have an **Access modifier**.
- Access modifiers determine whether other classes and methods in this program can access this class method or not.
- Types of access modifiers are private, public and so on.

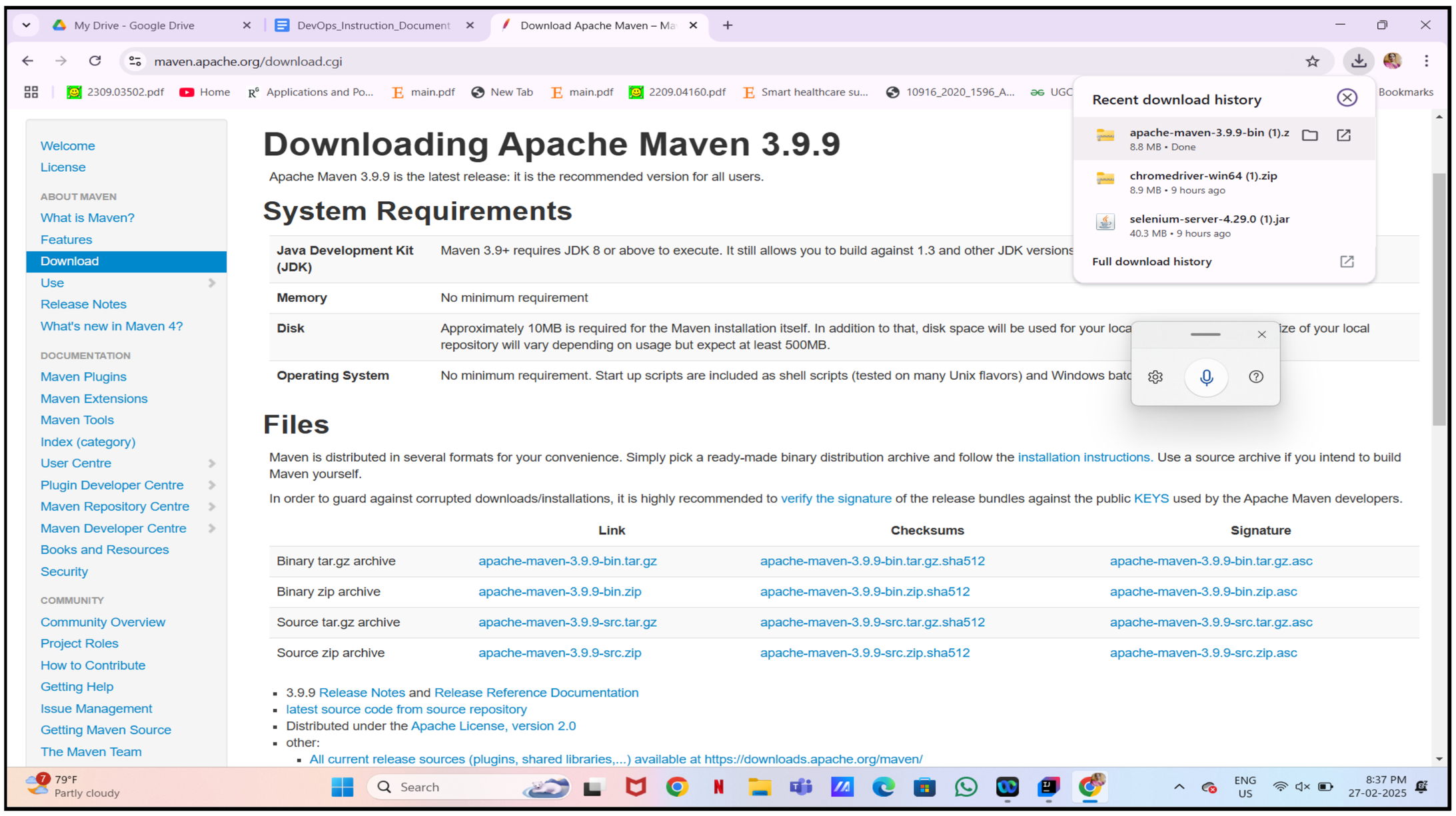
```
Public Class Main ()
{
Public Void main()
{
… ..
}
}
```

here the Main starts with the capital letter. This is because java uses a naming convention called **Pascal Naming Convention**(1st letter in the word should be upper) in contrast java uses camel Naming Convention, which means 1st letter of every word should be uppercase except for the first word(**main()**).

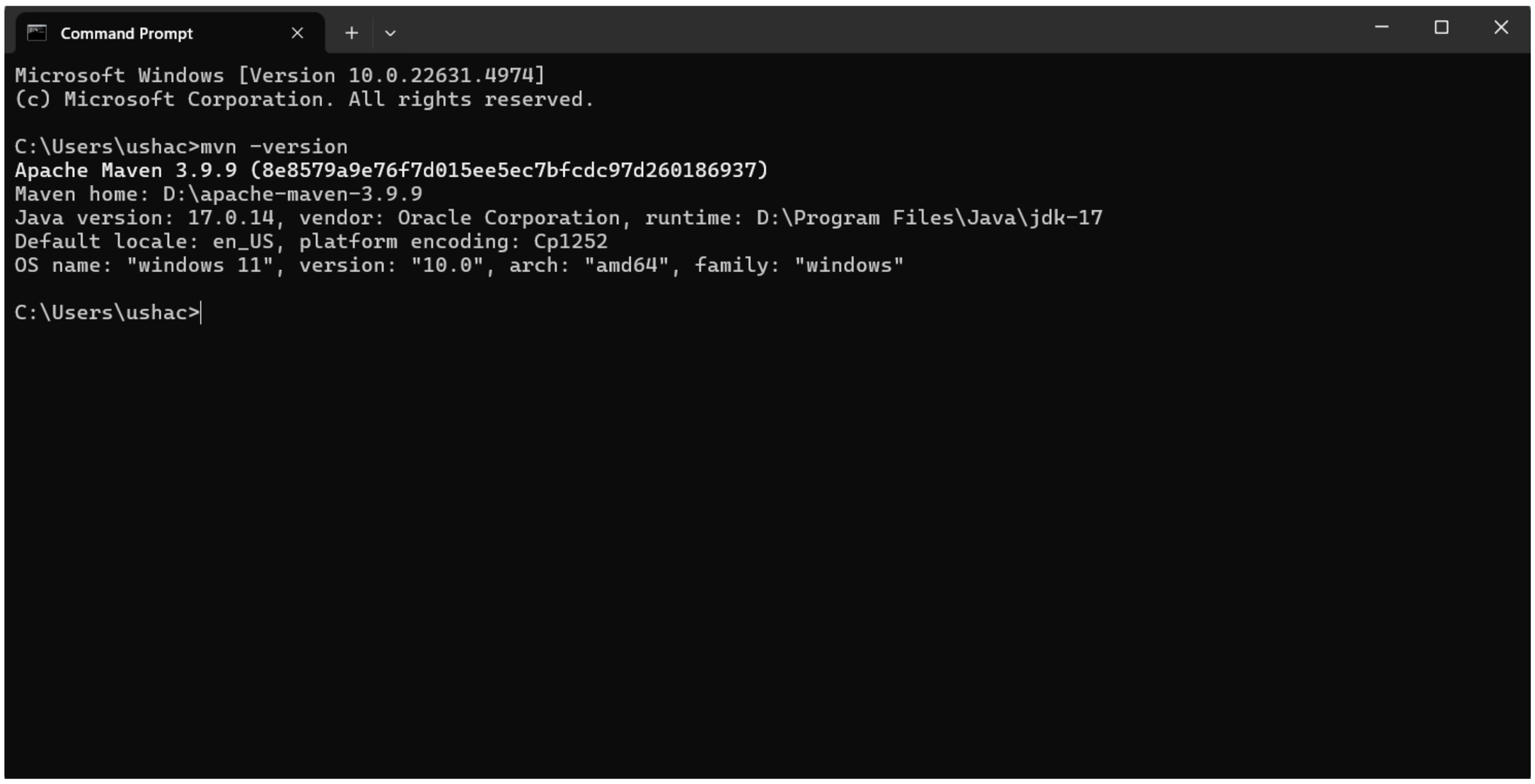
4. Installation of Maven

Browser → Maven download


My Drive - Google Drive
DevOps_Instruction_Document
maven download - Google Sear
google.com/search?q=maven+download&rlz=1C1GCEA_enIN1100IN1102&oq=maven&gs_lcrp=EgZjaHJvbWUqDAgBECMYJxiABBiKBTIPCAAQRRg5GIMBGLEDGIAEMgwIARAjGCcYgAQYigUyDQgCEAAY...
2309.03502.pdf
Home
Applications and Po...
main.pdf
New Tab
main.pdf
2209.04160.pdf
Smart healthcare su...
10916_2020_1596_A...
UGC NET CS Previo...
All Bookmarks
Google
maven download
All
Videos
Images
Shopping
News
Forums
Web
More
Tools
Apache Maven
https://maven.apache.org › download
Download Apache Maven – Maven
Previous Stable 3.8.x Release. Apache Maven 3.8.8 is the previous stable minor release for all users. ...
Maven is distributed in several formats for your ...
Installation Instructions
What is Maven? Features · Download · Use. Install; Run ...
3.3
Current releases can be found on our download server. Icon ...
Release Notes [–] Maven 3.8.8
Built by Maven. Release Notes – Maven 3.8.8. The Apache ...
Release Notes
Download · Welcome · License; About Maven; What is Maven ...
Index of /maven
Index of /maven. Icon Name Last modified Size Description ...
More results from apache.org »
Apache Maven
https://maven.apache.org
Welcome to Apache Maven – Maven
https://maven.apache.org/download.cgi
wnload, Install, Configure, Run Maven, Maven Plugins and Maven Extensions ... Improve the
79°F
Partly cloudy
Search
ENG
US
8:35 PM
27-02-2025


My Drive - Google Drive
DevOps_Instruction_Document
Download Apache Maven – Ma
maven.apache.org/download.cgi
Recent download history
apache-maven-3.9.9-bin (1).z
8.8 MB • Done
chromedriver-win64 (1).zip
8.9 MB • 9 hours ago
selenium-server-4.29.0 (1).jar
40.3 MB • 9 hours ago
Full download history
Welcome
License
ABOUT MAVEN
What is Maven?
Features
Download
Use
Release Notes
What's new in Maven 4?
DOCUMENTATION
Maven Plugins
Maven Extensions
Maven Tools
Index (category)
User Centre
Plugin Developer Centre
Maven Repository Centre
Maven Developer Centre
Books and Resources
Security
COMMUNITY
Community Overview
Project Roles
How to Contribute
Getting Help
Issue Management
Getting Maven Source
The Maven Team
Downloading Apache Maven 3.9.9
Apache Maven 3.9.9 is the latest release: it is the recommended version for all users.
System Requirements
Java Development Kit (JDK)
Maven 3.9+ requires JDK 8 or above to execute. It still allows you to build against 1.3 and other JDK versions
Memory
No minimum requirement
Disk
Approximately 10MB is required for the Maven installation itself. In addition to that, disk space will be used for your loca
ize of your local repository will vary depending on usage but expect at least 500MB.
Operating System
No minimum requirement. Start up scripts are included as shell scripts (tested on many Unix flavors) and Windows batc
Files
Maven is distributed in several formats for your convenience. Simply pick a ready-made binary distribution archive and follow the installation instructions. Use a source archive if you intend to build Maven yourself.
In order to guard against corrupted downloads/installations, it is highly recommended to verify the signature of the release bundles against the public KEYS used by the Apache Maven developers.
Link
Checksums
Signature
Binary tar.gz archive
apache-maven-3.9.9-bin.tar.gz
apache-maven-3.9.9-bin.tar.gz.sha512
apache-maven-3.9.9-bin.tar.gz.asc
Binary zip archive
apache-maven-3.9.9-bin.zip
apache-maven-3.9.9-bin.zip.sha512
apache-maven-3.9.9-bin.zip.asc
Source tar.gz archive
apache-maven-3.9.9-src.tar.gz
apache-maven-3.9.9-src.tar.gz.sha512
apache-maven-3.9.9-src.tar.gz.asc
Source zip archive
apache-maven-3.9.9-src.zip
apache-maven-3.9.9-src.zip.sha512
apache-maven-3.9.9-src.zip.asc
3.9.9 Release Notes and Release Reference Documentation
latest source code from source repository
Distributed under the Apache License, version 2.0
other:
All current release sources (plugins, shared libraries,...) available at https://downloads.apache.org/maven/
79°F
Partly cloudy
Search
ENG
US
8:37 PM
27-02-2025

After installing, check in cmd → mvn -version



5. Installation of Gradle

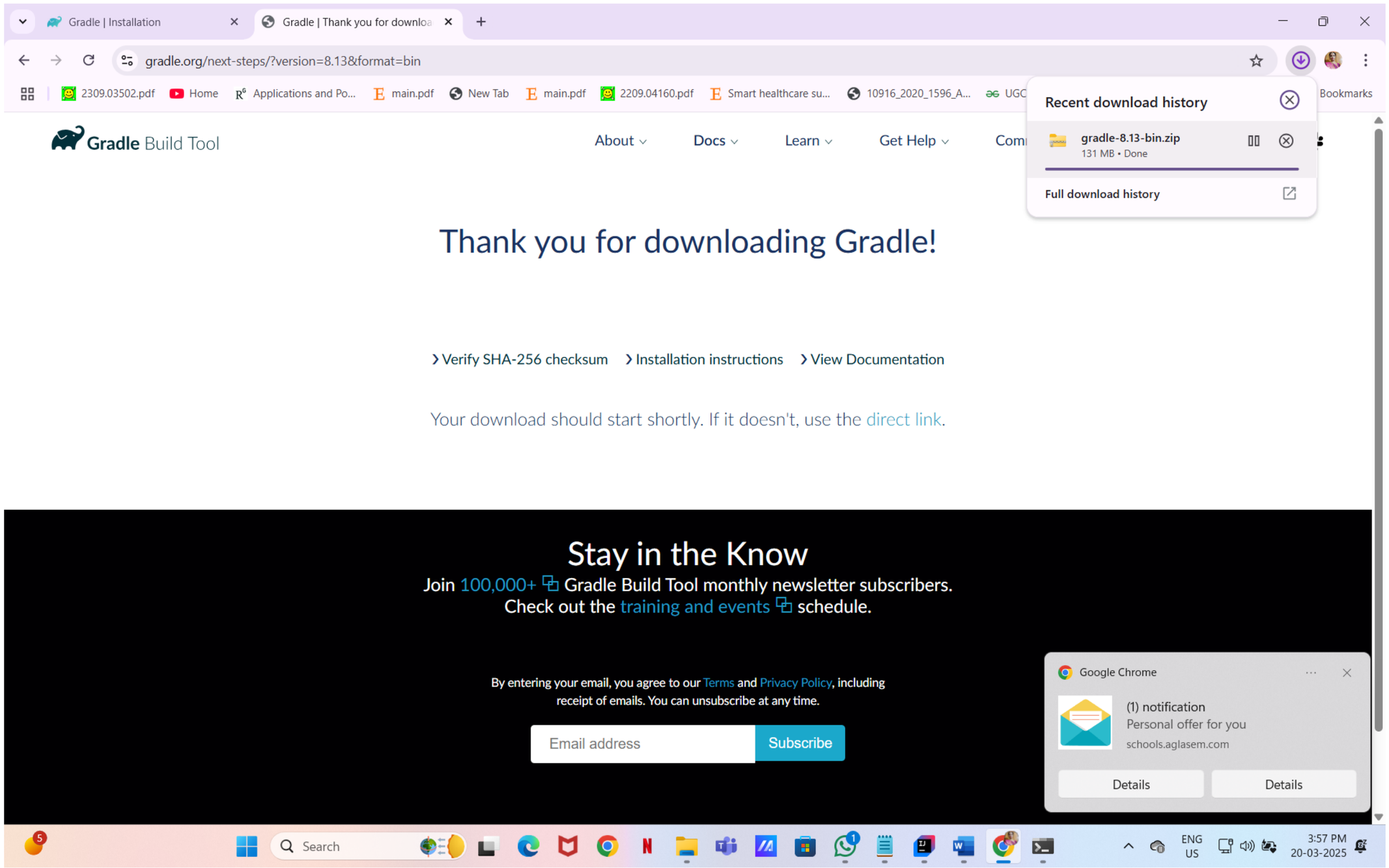
Step 1. Download the latest Gradle distribution

The current Gradle release is version 8.13, released on 25 Feb 2025. The distribution zip file comes in two flavors:

- Binary-only
- Complete, with docs and sources

If in doubt, choose the binary-only version and browse docs and sources online.

Need to work with an older version? See the releases page.



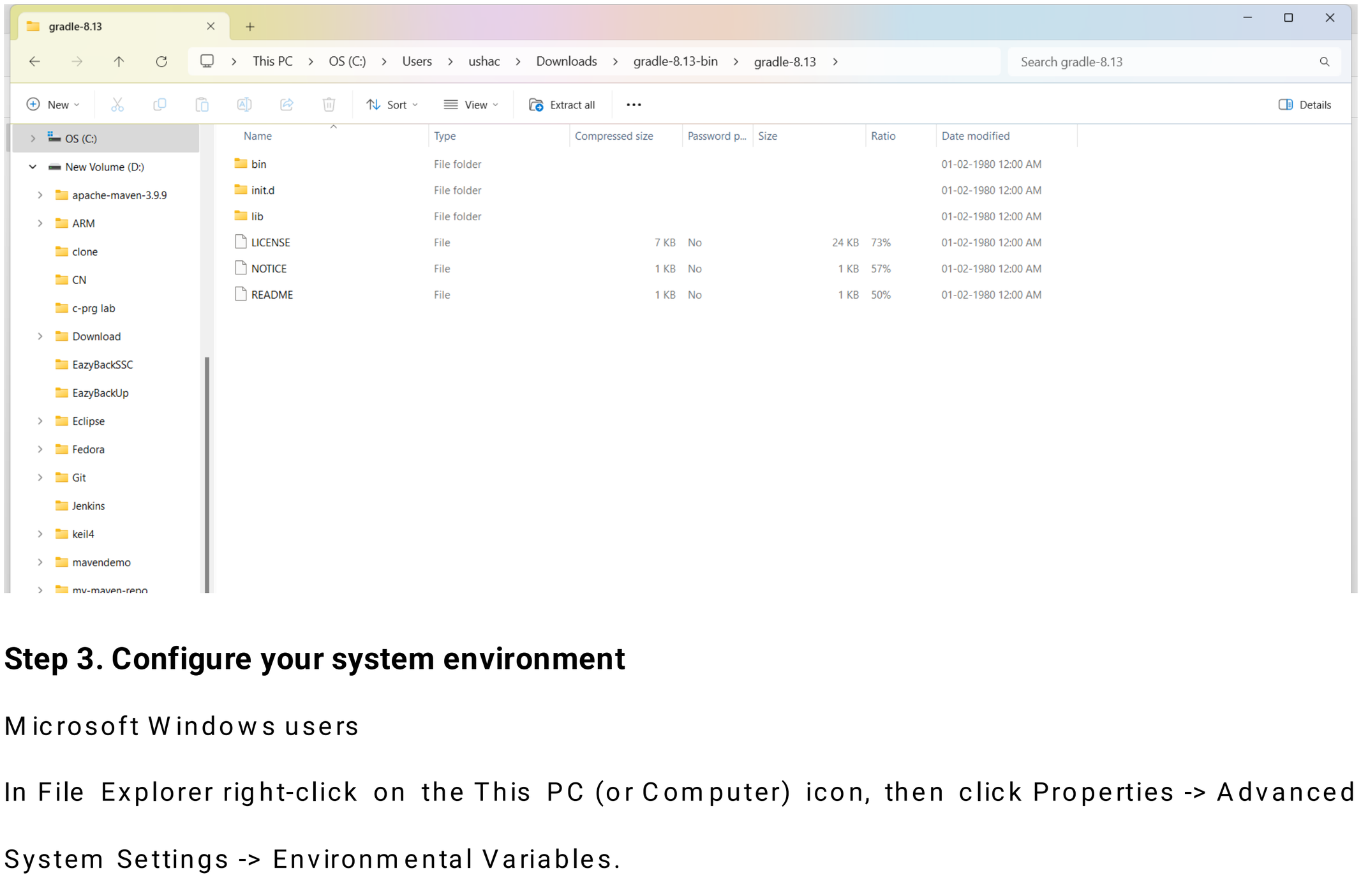
Step 2. Unpack the distribution

Microsoft Windows users

Create a new directory C:\Gradle with File Explorer.

Open a second File Explorer window and go to the directory where the Gradle distribution was downloaded. Double-click the ZIP archive to expose the content. Drag the content folder gradle-8.13 to your newly created C:\Gradle folder.

Alternatively you can unpack the Gradle distribution ZIP into C:\Gradle using an archiver tool of your choice.



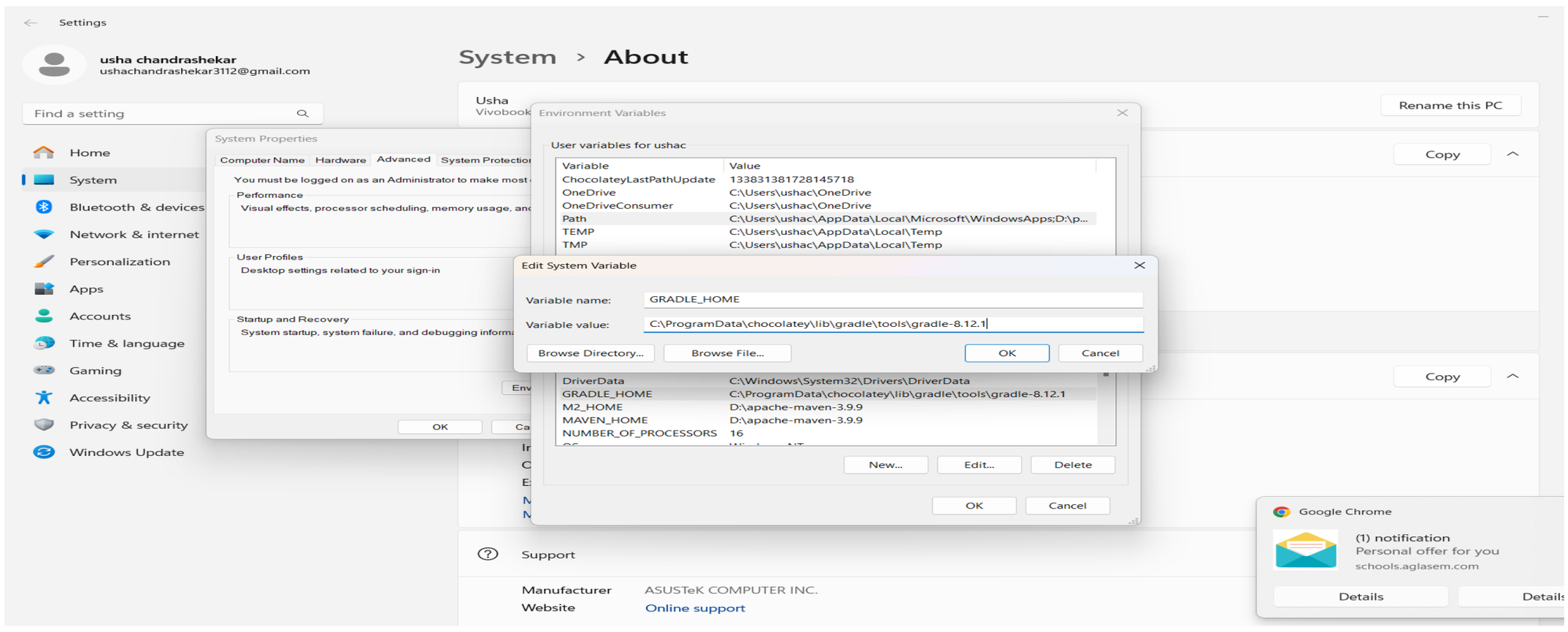
Step 3. Configure your system environment

Microsoft Windows users

In File Explorer right-click on the This PC (or Computer) icon, then click Properties -> Advanced System Settings -> Environmental Variables.

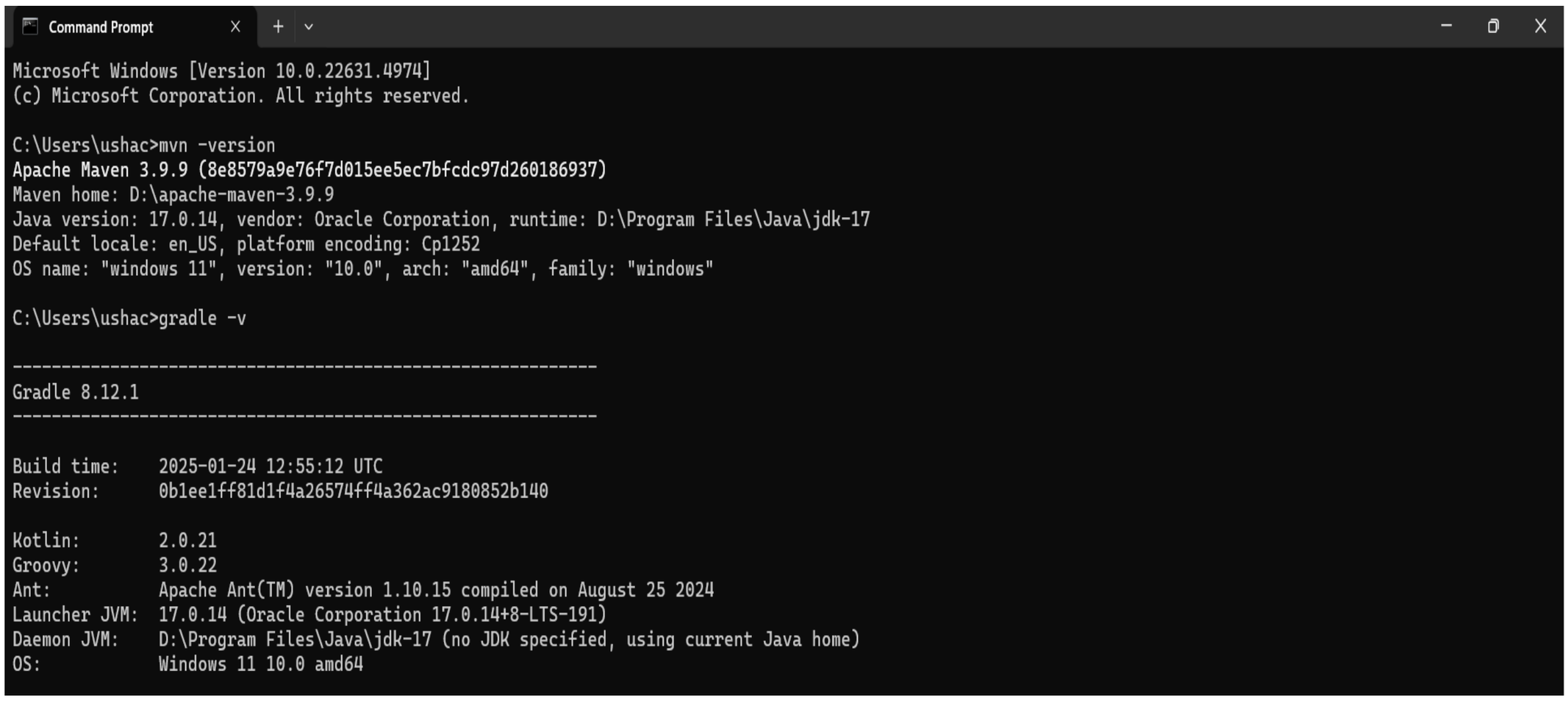
Under System Variables select Path, then click Edit. Add an entry for C:\Gradle\gradle-8.13\bin.

Click OK to save.



Step 4. Verify your installation

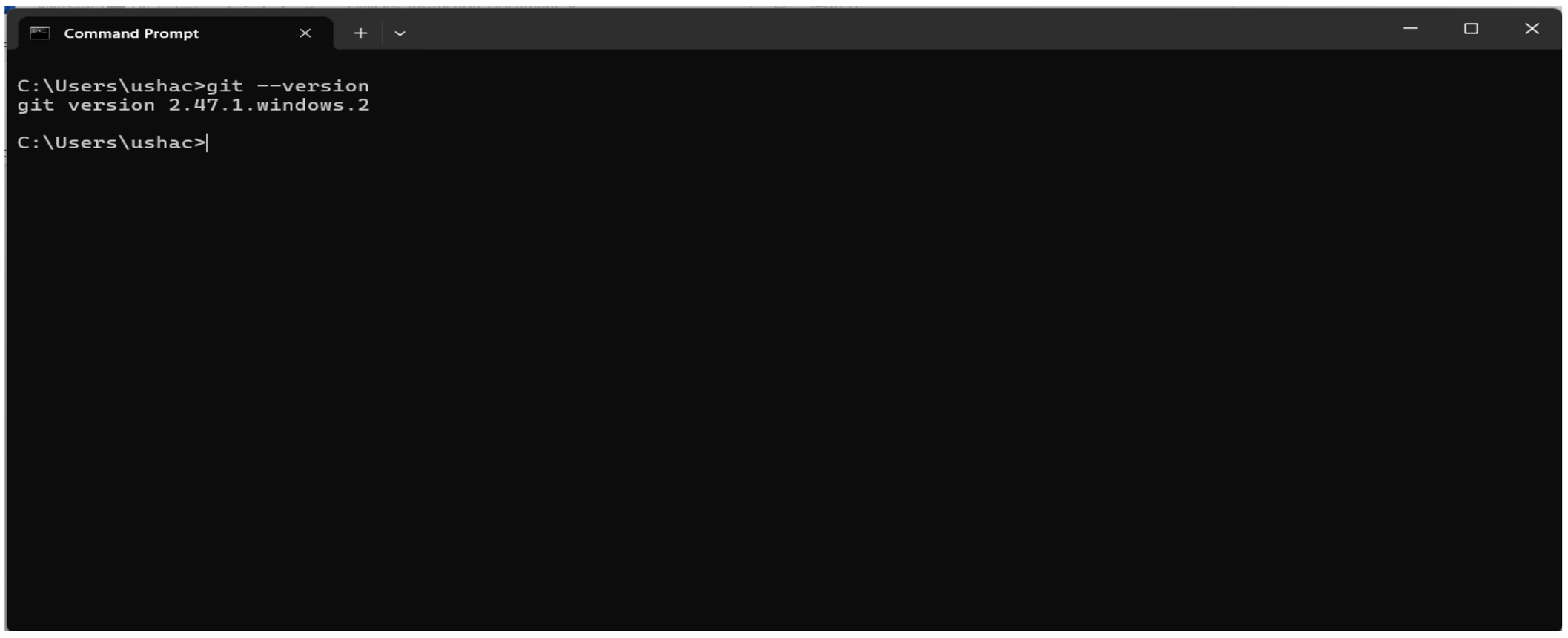
Open a console (or a Windows command prompt) and run gradle -v to run gradle and display the version.



6. Installation of GIT

Download GIT from git-scm.com and install.

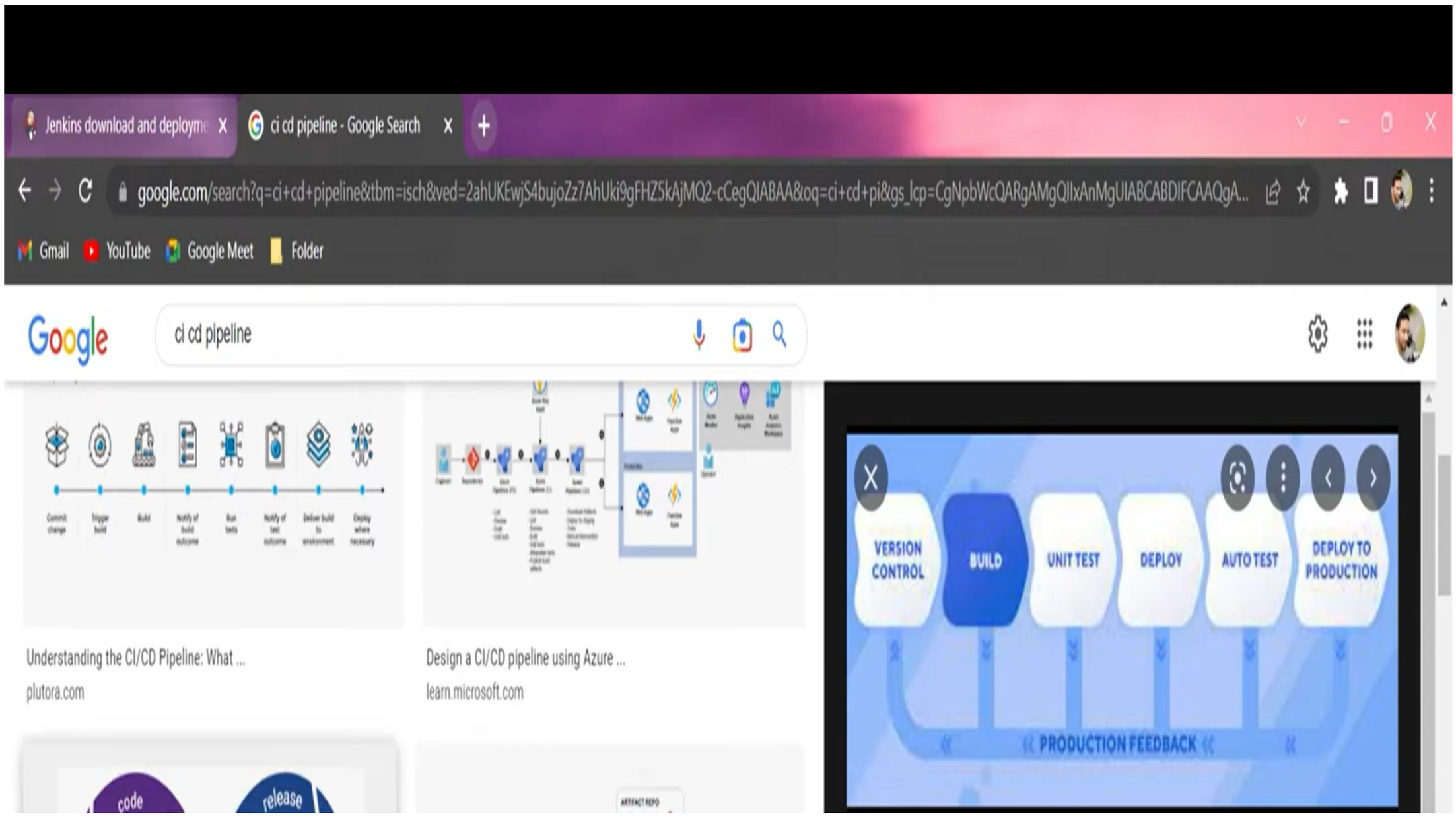
Check git --version in cmd



6. Installation of Jenkins

Goto Browser → jenkins.io → Download → Select Generic java package(.war), as shown ---

Create new folder and save the war file.



Go to cmd → type – java -jar jenkins.war

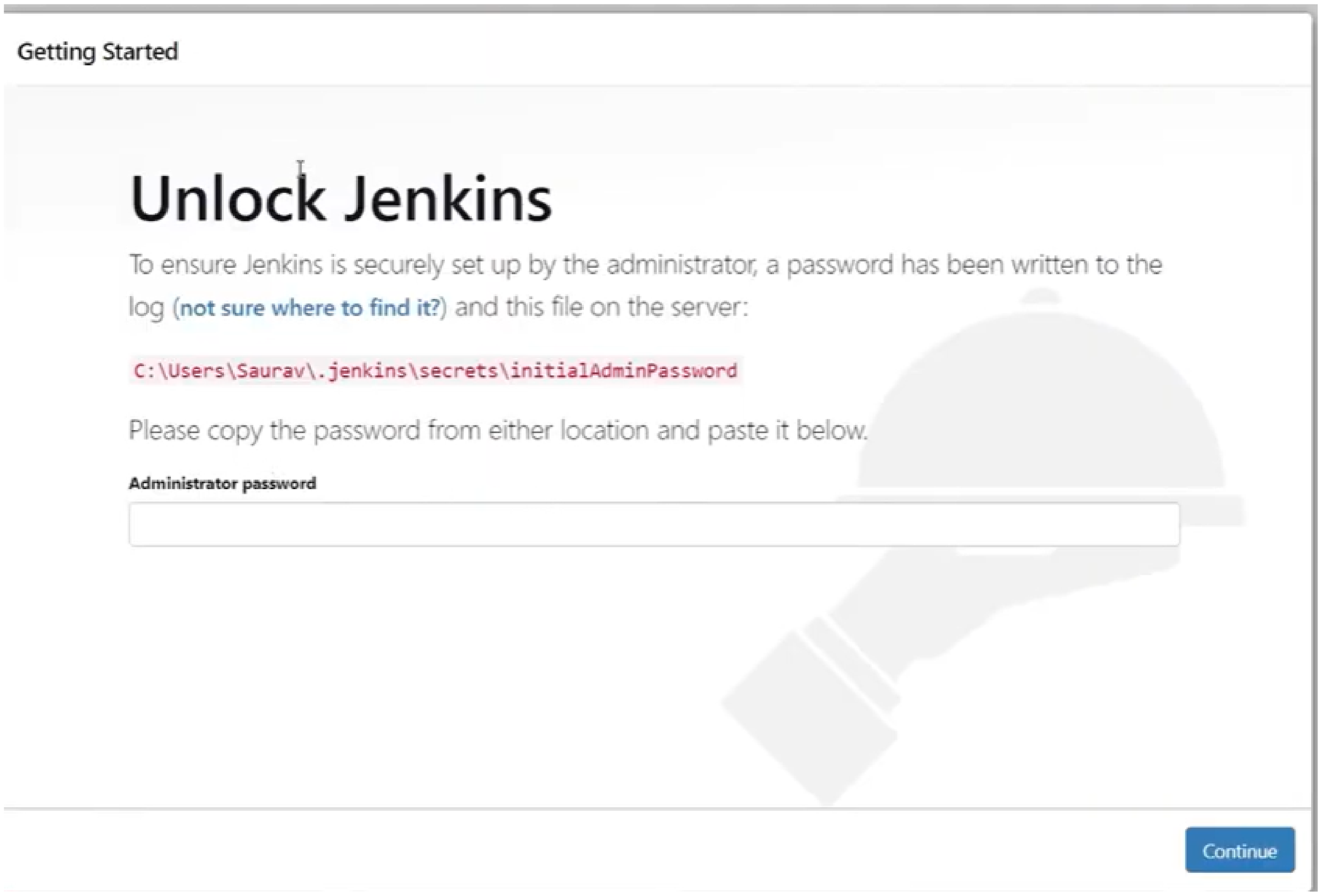
```
C:\Windows\System32\cmd.e
Microsoft Windows [Version 10.0.22631.4974]
(c) Microsoft Corporation. All rights reserved.

D:\Jenkins>java -jar jenkins.war
Running from: D:\Jenkins\jenkins.war
webroot: C:\Users\ushac\.jenkins\war
2025-03-24 06:14:16.033+0000 [id=1]     INFO    winstone.Logger#logInternal: Beginning extraction from war file
2025-03-24 06:14:16.080+0000 [id=1]     WARNING o.e.j.ee9.nested.ContextHandler#setContextPath: Empty contextPath
2025-03-24 06:14:16.120+0000 [id=1]     INFO    org.eclipse.jetty.server.Server#doStart: jetty-12.0.16; built: 2024-12-09T21:02:54.535Z; git: c3f88bafb4e393
f23204dc14dc57b042e84debc7; jvm 17.0.14+8-LTS-191
2025-03-24 06:14:16.553+0000 [id=1]     INFO    o.e.j.e.w.StandardDescriptorProcessor#visitServlet: NO JSP Support for /, did not find org.eclipse.jetty.ee9
.jsp.JettyJspServlet
2025-03-24 06:14:16.593+0000 [id=1]     INFO    o.e.j.s.DefaultSessionIdManager#doStart: Session workerName=node0
2025-03-24 06:14:16.962+0000 [id=1]     INFO    hudson.WebAppMain#contextInitialized: Jenkins home directory: C:\Users\ushac\.jenkins found at: $user.home/.
jenkins
2025-03-24 06:14:17.037+0000 [id=1]     INFO    o.e.j.s.handler.ContextHandler#doStart: Started oeje9n.ContextHandler$CoreContextHandler@6e9c413e{Jenkins v2
.479.3,/,b=file:///C:/Users/ushac/.jenkins/war/,a=AVAILABLE,h=oeje9n.ContextHandler$CoreContextHandler$CoreToNestedHandler@57a4d5ee{STARTED}}
2025-03-24 06:14:17.054+0000 [id=1]     INFO    o.e.j.server.AbstractConnector#doStart: Started ServerConnector@1bb9aa43{HTTP/1.1, (http/1.1)}{0.0.0.0:8080}
2025-03-24 06:14:17.064+0000 [id=1]     INFO    org.eclipse.jetty.server.Server#doStart: Started oejs.Server@7ed7259e{STARTING}[12.0.16,sto=0] @1454ms
2025-03-24 06:14:17.064+0000 [id=36]    INFO    winstone.Logger#logInternal: Winstone Servlet Engine running: controlPort=disabled
2025-03-24 06:14:17.207+0000 [id=42]    INFO    jenkins.InitReactorRunner$1#onAttained: Started initialization
2025-03-24 06:14:17.347+0000 [id=43]    INFO    jenkins.InitReactorRunner$1#onAttained: Listed all plugins
2025-03-24 06:14:19.535+0000 [id=55]    INFO    jenkins.InitReactorRunner$1#onAttained: Prepared all plugins
2025-03-24 06:14:19.555+0000 [id=63]    INFO    jenkins.InitReactorRunner$1#onAttained: Started all plugins
2025-03-24 06:14:19.557+0000 [id=63]    INFO    jenkins.InitReactorRunner$1#onAttained: Augmented all extensions
2025-03-24 06:14:19.848+0000 [id=71]    INFO    h.p.b.g.GlobalTimeOutConfiguration#load: global timeout not set
2025-03-24 06:14:20.550+0000 [id=64]    INFO    jenkins.InitReactorRunner$1#onAttained: System config loaded
2025-03-24 06:14:20.555+0000 [id=64]    INFO    jenkins.InitReactorRunner$1#onAttained: System config adapted
2025-03-24 06:14:20.589+0000 [id=66]    INFO    jenkins.InitReactorRunner$1#onAttained: Loaded all jobs
2025-03-24 06:14:20.604+0000 [id=62]    INFO    jenkins.InitReactorRunner$1#onAttained: Configuration for all jobs updated
2025-03-24 06:14:20.615+0000 [id=86]    INFO    hudson.util.Retrier#start: Attempt #1 to do the action check updates server
2025-03-24 06:14:20.623+0000 [id=65]    INFO    jenkins.InitReactorRunner$1#onAttained: Completed initialization
2025-03-24 06:14:20.648+0000 [id=34]    INFO    hudson.lifecycle.Lifecycle#onReady: Jenkins is fully up and running
```

Jenkins default port number is http://localhost/8080

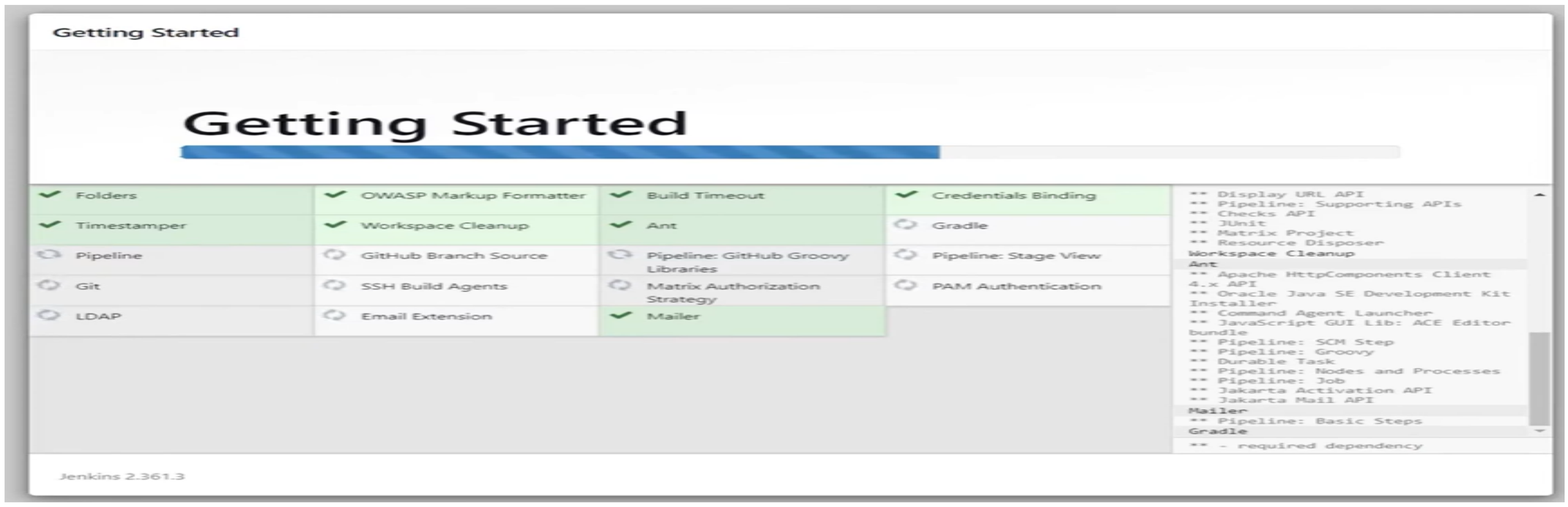
For the first time installing Jenkins remember the Checksum

```
2022-11-07 14:31:53.186+0000 [id=87]    INFO    hudson.model.AsyncPeriodicWork#lambda$doR
n$1: Started Download metadata
2022-11-07 14:31:53.196+0000 [id=87]    INFO    hudson.util.Retrier#start: Attempt #1 to d
o the action check updates server
2022-11-07 14:31:53.521+0000 [id=54]    INFO    jenkins.install.SetupWizard#init:

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

Jenkins initial setup is required. An admin user has been created and a password generated

Please use the following password to proceed to installation:

3c1e7ff37b0242aa979a9c1baf69edac
```

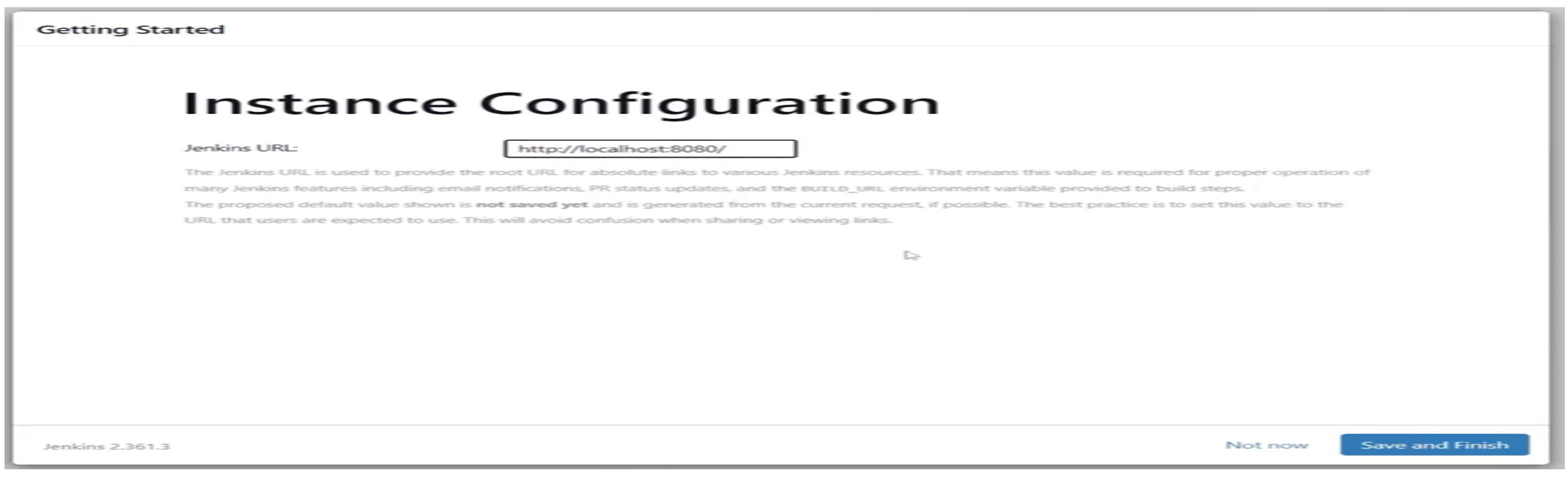
Open browser type localhost:8080, paste the checksum in Administrator password.

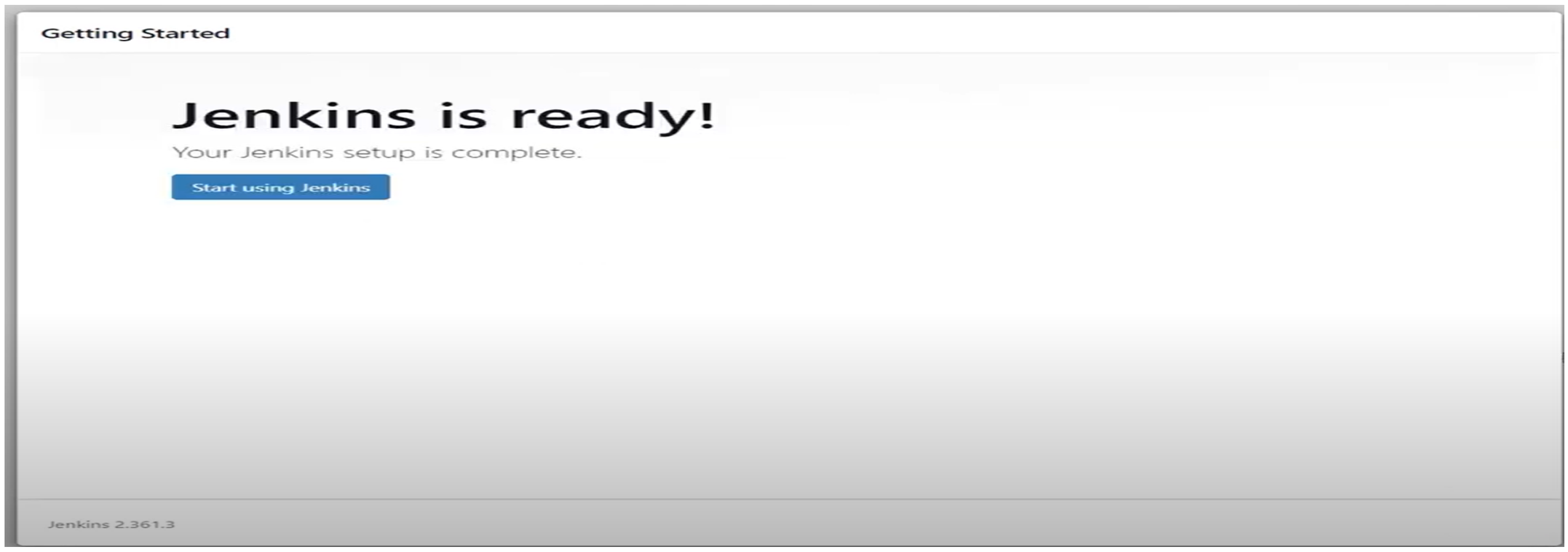


Select the install suggested plugins

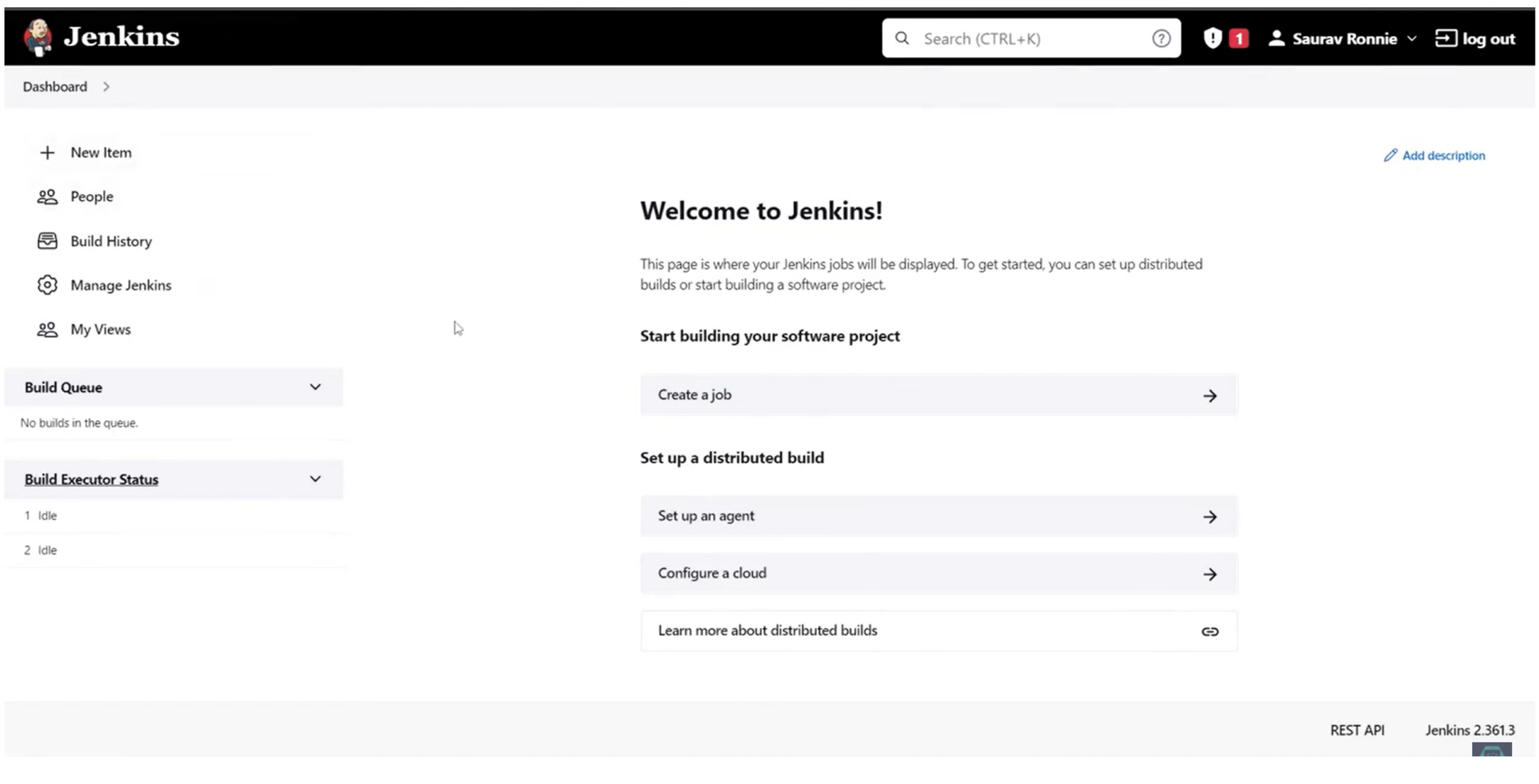

Getting Started
Getting Started
Folders
OWASP Markup Formatter
Build Timeout
Credentials Binding
Timestamper
Workspace Cleanup
Ant
Gradle
Pipeline
GitHub Branch Source
Pipeline: GitHub Groovy Libraries
Pipeline: Stage View
Git
SSH Build Agents
Matrix Authorization Strategy
PAM Authentication
LDAP
Email Extension
Mailer
** Display URL API
** Pipeline: Supporting APIs
** Checks API
** JUnit
** Matrix Project
** Resource Disposer
Workspace Cleanup
Ant
** Apache HttpComponents Client 4.x API
** Oracle Java SE Development Kit Installer
** Command Agent Launcher
** JavaScript GUI Lib: ACE Editor bundle
** Pipeline: SCM Step
** Pipeline: Groovy
** Durable Task
** Pipeline: Nodes and Processes
** Pipeline: Job
** Jakarta Activation API
** Jakarta Mail API
Mailer
** Pipeline: Basic Steps
Gradle
** - required dependency
Jenkins 2.361.3

Getting Started
Create First Admin User
Username:
Ronnie
Password:
Confirm password:
Full name:
E-mail address:
Jenkins 2.361.3
Skip and continue as admin
Save and Continue


Getting Started
Instance Configuration
Jenkins URL:
http://localhost:8080/
The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the BUILD_URL environment variable provided to build steps.
The proposed default value shown is not saved yet and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.
Jenkins 2.361.3
Not now
Save and Finish



Dashboard of Jenkins



Experiment 3

Build and run a Java application using Selenium - manual approach, create a Java application with Maven and migrate the same application to Gradle. Understanding the POM file, Dependency management and plugins. understanding build scripts in Gradle (Groovy and Kotlin DSL), Dependency management and Task automation.

File → New Project → SeleniumAutomationwithJava

Src → New Package → com.example → New Java class → 
FirstSelenium.java
,


GetTitleAndUrl.java


BrowserInSelenium.java
,


logintest.java

File → New Project → SeleniumAutomationwithJava → right click → new Directory → Lib and drivers (paste the jar and chrome drivers file in directories respectively)

FirstSelenium.java

```
package com.example;

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

public class FirstSelenium {
    static WebDriver driver;

    public static void main(String[] args) {
        System.setProperty("web driver.chrome.driver","./Drivers/chromedriver.exe");
        driver =new ChromeDriver();
        driver.get("https://www.google.com");
        //driver.navigate().to("https://www.google.com");
```

```
        driver.quit();
    }

}
```

GetTitleAndUrl.java

```
package com.example;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class GetTitleAndUrl {
   static WebDriver driver;

 public static void main(String[] args) {
      System.setProperty("web driver.chrome.driver", "./Drivers/chromedriver.exe");
      driver = new ChromeDriver();
      driver.get("https://www.google.com");
      driver.navigate().to("https://www.bietdvg.edu/");
      String url = driver.getCurrentUrl();
      String title = driver.getTitle();
      System.out.println("The url is: "   +url );
      System.out.println("The title is: " + title);
      driver.quit();
   }
}
```

BrowserInSelenium.java

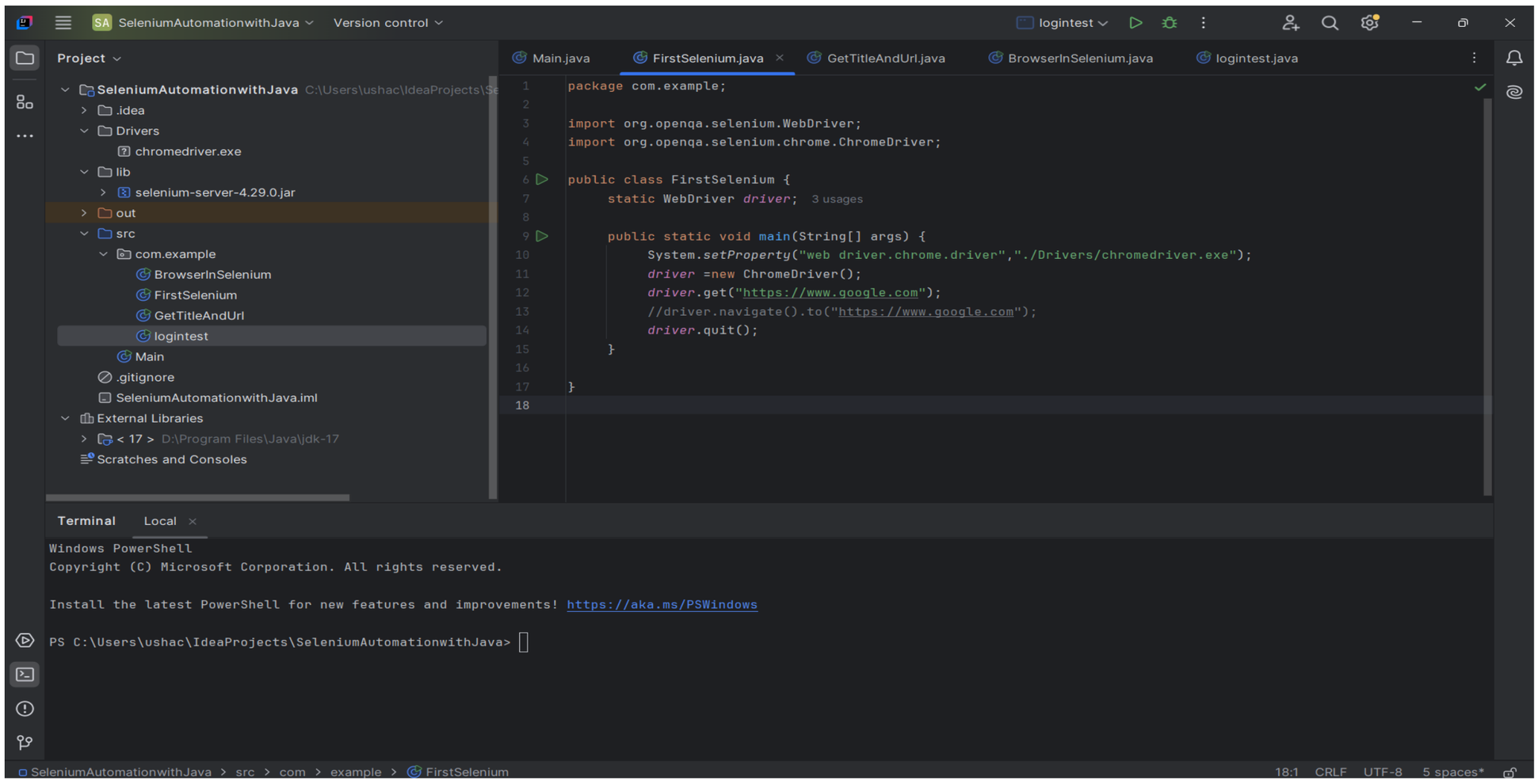
```
package com.example;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class BrowserInSelenium {
   static WebDriver driver;

   public static void main(String[] args) throws InterruptedException {
```

```
        System.setProperty("web driver.chrome.driver", "./Drivers/chromedriver.exe");
        driver = new ChromeDriver();
        driver.navigate().to("https://www.google.com/");
        Thread.sleep(2000);// Hard wait not recommended
        driver.navigate().to("https://www.selenium.dev/");
        Thread.sleep(2000);
        driver.navigate().to("https://www.saucedemo.com/");
        Thread.sleep(2000);
        driver.navigate().back();
        Thread.sleep(2000);
        driver.navigate().forward();
        driver.findElement(By.name("user-name")).sendKeys("Usha");
        Thread.sleep(2000);
        driver.navigate().refresh();
        Thread.sleep(1000);
        driver.close();
    }
}
```

loginTest.java

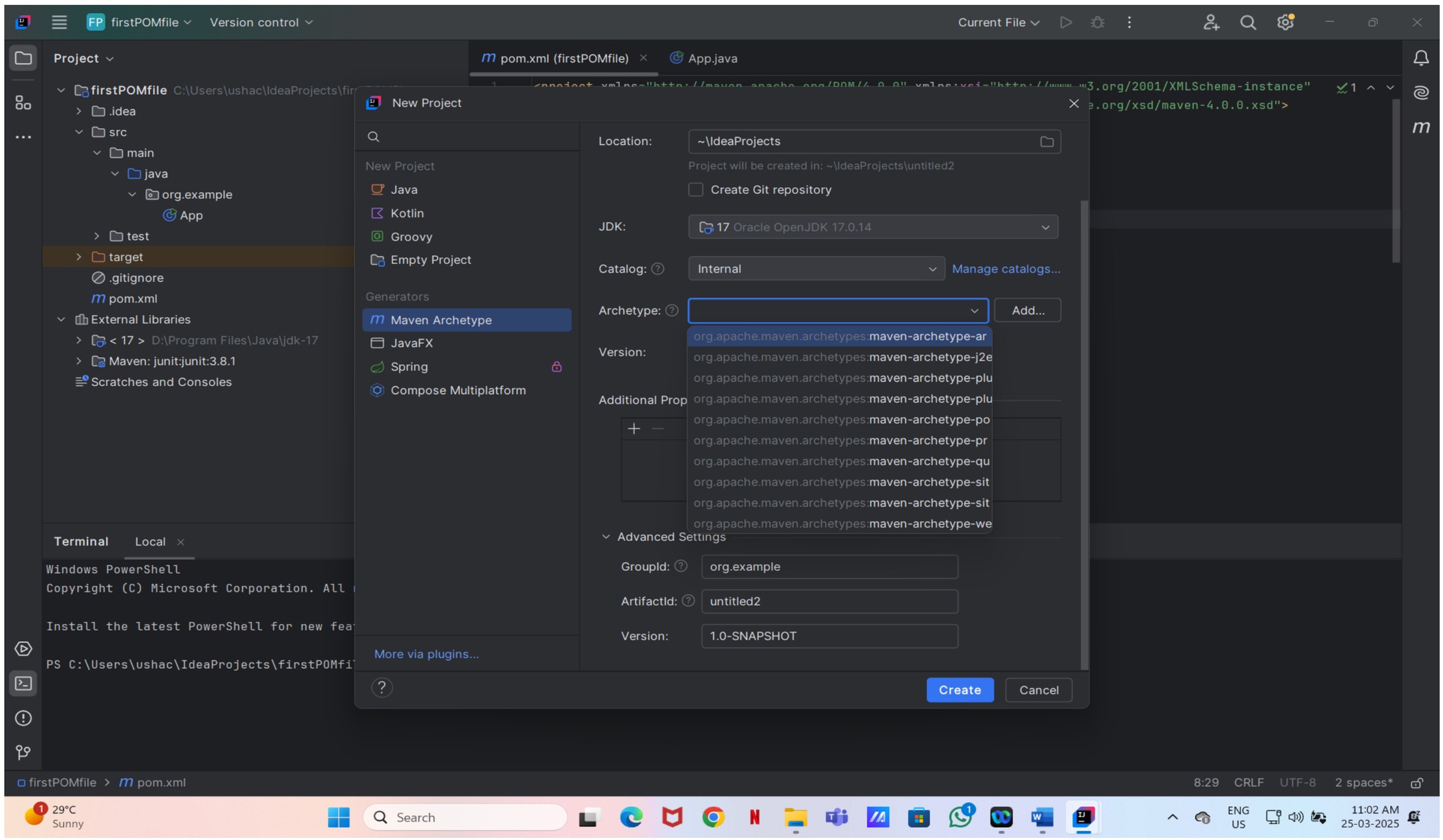
```
package com.example;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class logintest {
    public static void main(String[] args) throws InterruptedException {
        WebDriver driver = new ChromeDriver();
        driver.get("https://www.saucedemo.com/");
        driver.manage().window().maximize();
        Thread.sleep(2000);
        driver.findElement(By.id("user-name")).sendKeys("standard_user");
        Thread.sleep(2000);
        driver.findElement(By.id("password")).sendKeys("secret_sauce");
        Thread.sleep(2000);
        driver.findElement(By.id("login-button")).click();
        Thread.sleep(2000);
        driver.quit();
    }
}
```

Note: Sauce Demo is a sample website created by Sauce Labs, a cloud-based testing platform, designed to allow users to practice browser automation and test their code against a variety of browsers and operating systems



MAVEN PROJECT

File → new project → maven archetype → firstPOMfile → select maven-archetype-QuickStart → create



Pom.xml(firstPOMfile)

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>org.example</groupId>
  <artifactId>firstPOMfile</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>firstPOMfile</name>
  <url>http://maven.apache.org</url>

  <properties>
   <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>

  <dependencies>
   <dependency>
```

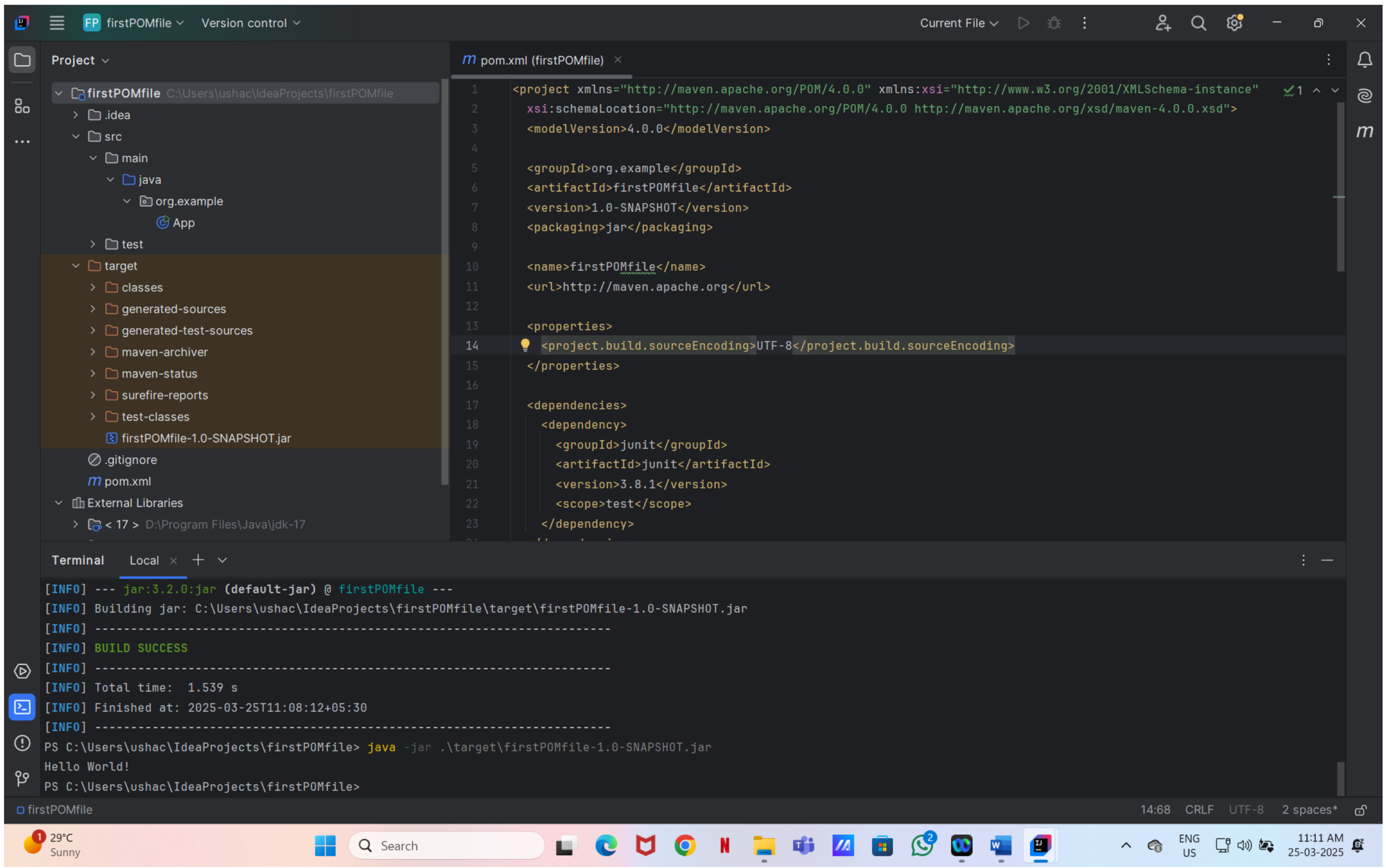
```
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>3.8.1</version>
    <scope>test</scope>
  </dependency>
</dependencies>
<build>
  <plugins>
    <!-- Compiler Plugin -->
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>3.8.1</version>
      <configuration>
        <source>1.8</source>
        <target>1.8</target>
</configuration>
</configuration>
    </plugin>

    <!-- Jar Plugin -->
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-jar-plugin</artifactId>
      <version>3.2.0</version>
      <configuration>
        <archive>
          <manifest>
            <mainClass>org.example.App</mainClass>
          </manifest>
        </archive>
      </configuration>
    </plugin>
  </plugins>
 </build>
</project>
```

In Terminal execute these commands

mvn clean compile
mvn package
java -jar .\target\firstPOMfile-1.0-SNAPSHOT.jar

Output

Hello world!



File → new project → SeleniumMaven → new directory → resources → 3files → index.html, logo.png, style.css (paste)

POM.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.example</groupId>
  <artifactId>seleniuminmaven</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>jar</packaging>
```

```
<name>seleniuminmaven</name>
<url>http://maven.apache.org</url>

<properties>
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>

<dependencies>
 <dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>3.8.1</version>
  <scope>test</scope>
 </dependency>

  <dependency>
   <groupId>org.seleniumhq.selenium</groupId>
   <artifactId>selenium-java</artifactId>
   <version>4.29.0</version>
  </dependency>

 <dependency>
  <groupId>org.testng</groupId>
  <artifactId>testng</artifactId>
  <version>7.11.0</version>
  <scope>test</scope>
 </dependency>
</dependencies>
<build>
<plugins>
 <plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-resources-plugin</artifactId>
  <version>3.2.0</version>
  <executions>
   <execution>
    <phase>prepare-package</phase>
    <goals>
     <goal>copy-resources</goal>
    </goals>
    <configuration>
```

```
        <outputDirectory>${project.basedir}/docs</outputDirectory>
        <!-- Deploy to /docs folder -->
        <resources>
         <resource>
          <directory>src/main/resources</directory>
          <includes>
           <include>/*</include> <!-- Copy all files in src/main/resources -->
          </includes>
         </resource>
        </resources>
       </configuration>
      </execution>
     </executions>
      </plugin>
    </plugins>
    </build>
</project>
```

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Computer Science and Engineering(Data Science)</title>
    <link rel="stylesheet" href="styles.css">
    <link rel="icon" type="image/x-icon" href="images/favicon.ico">
</head>
<body>

<!-- Header Section -->
<header>
    <div class="container">
        <h1>Vishveshwaraya Technological University</h1>
        <img src="logo.png" alt="VTU Logo" title="VTU">
        <nav>
            <a href="#Courses">Courses</a>
            <a href="#address">Address</a>
```

```
      </nav>
    </div>
</header>

<!-- Content Section -->
<div class="content">
    <!-- Services Section -->
    <section id="Courses">
      <h2>Courses</h2>
      <p>
        Bachelor of Engineering,Computer Applications and IT,Sciences, Management and
Business Administration,  Animation and Design.....
      </p>
    <div class="Courses">
      <div class="Courses">
        <h3>UG</h3>
        <p>
          Bachelor of Engineering.
        </p>
      </div>
      <div class="Courses">
        <h3>PG</h3>
        <p>
          Post Graduate.
        </p>
      </div>

    </div>
  </section>

    <!-- Address Section -->
    <section id="address">
      <h2>Address Info</h2>
      <p>
        QFH7+497 Jnana Sangama, VTU Main Rd, Visvesvaraya Technological University,
        Machhe, Belagavi, Karnataka 590018<br>
        <strong>Telephone:</strong> 0831 249 8100<br>
        <strong>Email:</strong> <a href="registrar@vtu.ac.in">registrar@vtu.ac.in.</a>
      </p>
    </section>
</div>
```

Style.css

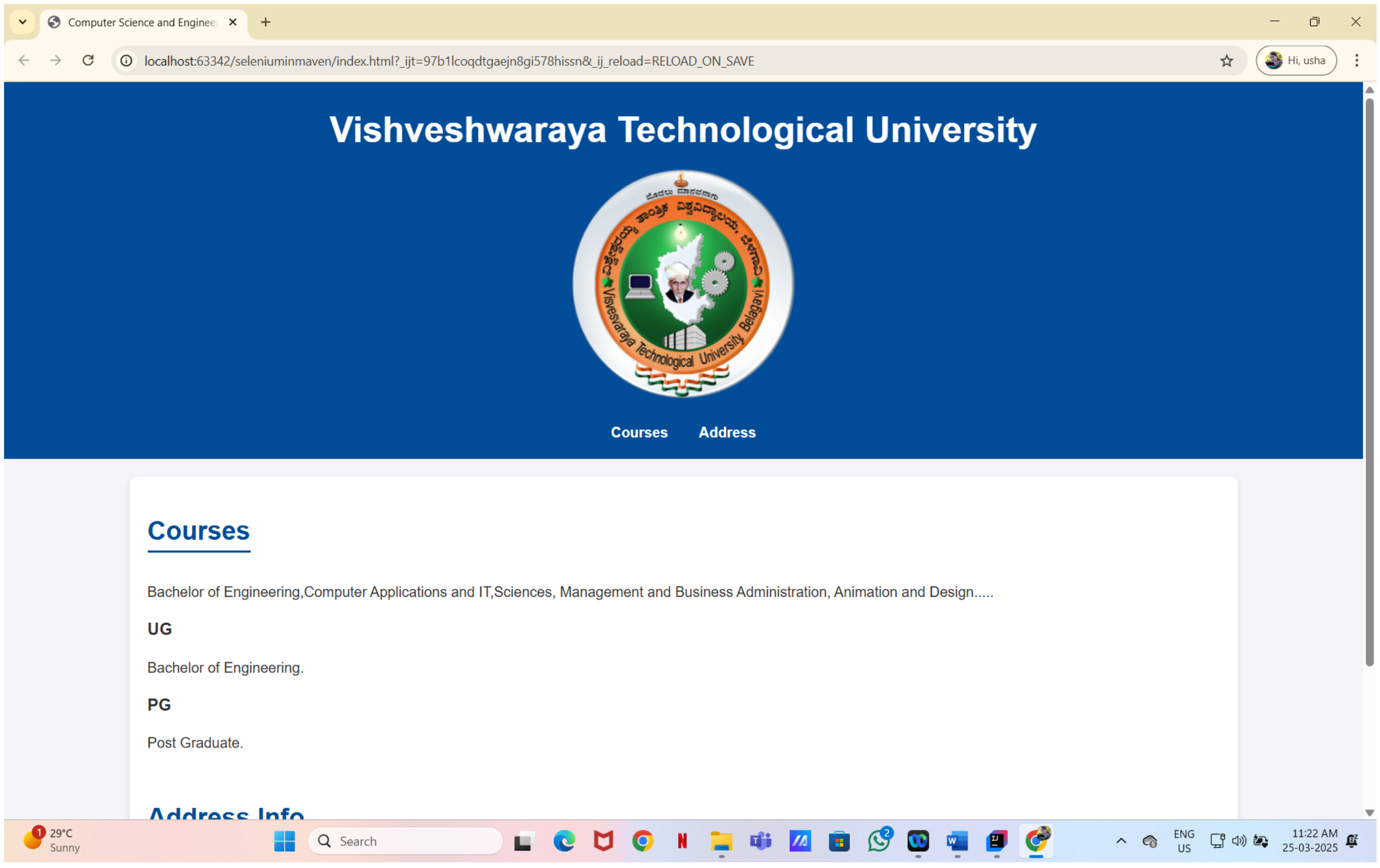
```
/* General Styles */
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f4f4f9;
    color: #333;
}

/* Header Styles */
header {
    background-color: #004c99;
    color: white;
    padding: 20px 0;
    text-align: center;
    position: sticky;
    top: 0;
    z-index: 1000;
}
header .container {
    max-width: 1200px;
    margin: 0 auto;
    text-align: center;
}
header h1 {
    margin: 10px 0;
    font-size: 2.5rem;
}
header img {
    max-width: 250px;
    margin: 10px 0;
    border-radius: 05px;
}
header nav {
    margin-top: 15px;
}
header nav a {
    color: white;
```

```
    margin: 0 15px;
    text-decoration: none;
    font-weight: bold;
    font-size: 1rem;
}
header nav a:hover {
    text-decoration: underline;
}

/* Content Section */
.content {
    max-width: 1200px;
    margin: 20px auto;
    padding: 20px;
    background-color: white;
    border-radius: 8px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}
.content section {
    margin-bottom: 30px;
}
.content section h2 {
    color: #004c99;
    border-bottom: 3px solid #004c99;
    display: inline-block;
    padding-bottom: 5px;
    font-size: 1.8rem;
    margin-bottom: 15px;
}
.content section p {
    line-height: 1.6;
    font-size: 1rem;
    text-align: justify;
}

/* Services Section */
.services {
    display: flex;
    flex-wrap: wrap;
    gap: 20px;
}
```

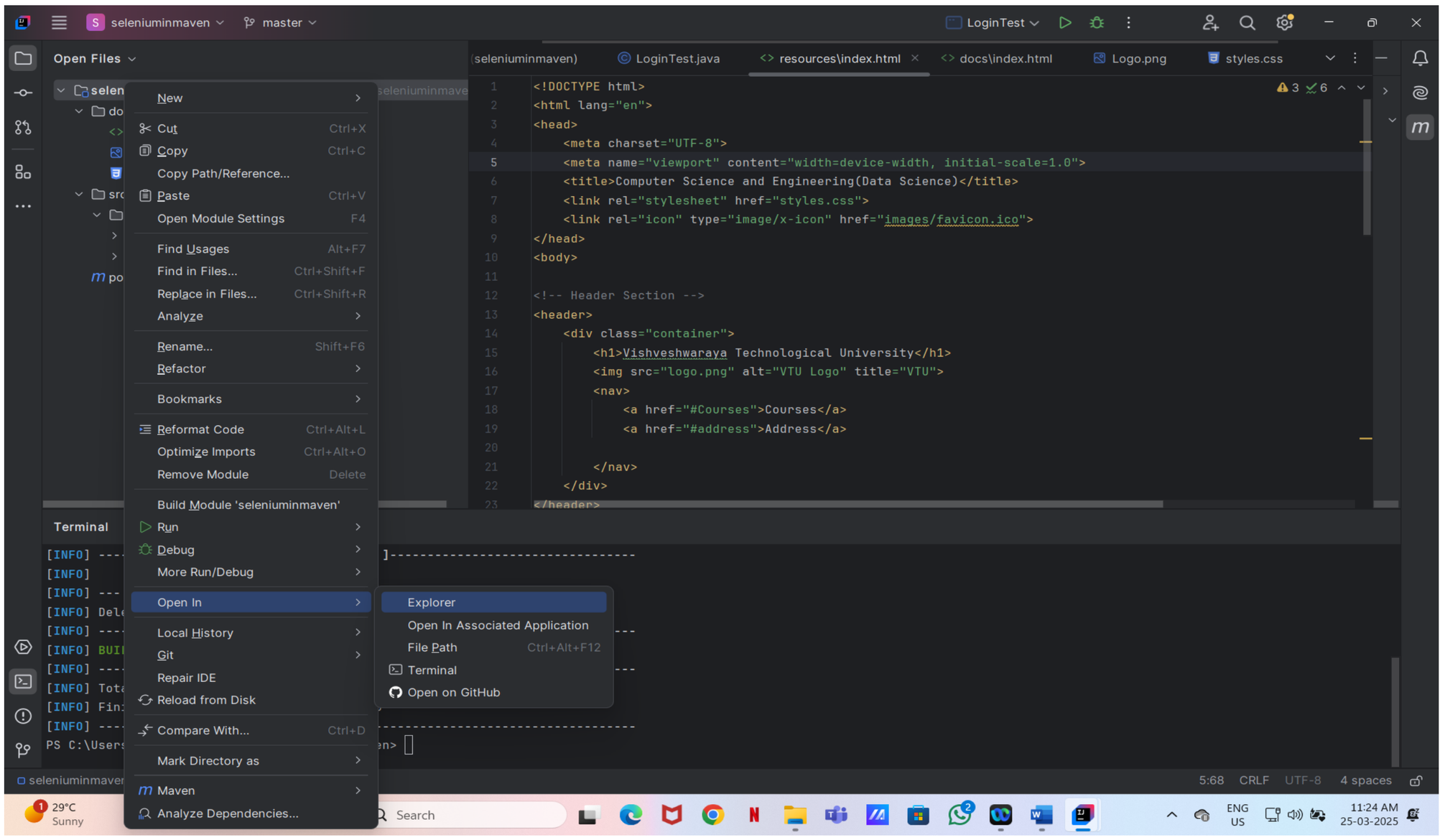
```
.service {
    flex: 1 1 calc(33.333% - 20px);
    background-color: #eaf3ff;
    padding: 15px;
    border-radius: 8px;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}
.service h3 {
    color: #004c99;
    font-size: 1.3rem;
    margin-bottom: 10px;
}
.service p {
    font-size: 0.95rem;
}

/* Address Section */
#address a {
    color: #004c99;
    text-decoration: none;
    font-weight: bold;
}
#address a:hover {
    text-decoration: underline;
}

/* Footer Styles */
footer {
    background-color: #004c99;
    color: white;
    text-align: center;
    padding: 15px 0;
    font-size: 0.9rem;
}
footer a {
    color: white;
    text-decoration: none;
    font-weight: bold;
}
footer a:hover {
    text-decoration: underline;
```

```
}

/* Responsive Styles */
@media (max-width: 768px) {
    .services {
        flex-direction: column;
    }
    .service {
        flex: 1 1 100%;
    }
    header nav a {
        display: block;
        margin: 5px 0;
    }
}
```

Logo.png



Right click on project → open in → explorer → open seleniummaven folder → right click → open git bash here



git init

git status

git add .

git commit -m “ selenium test in maven”

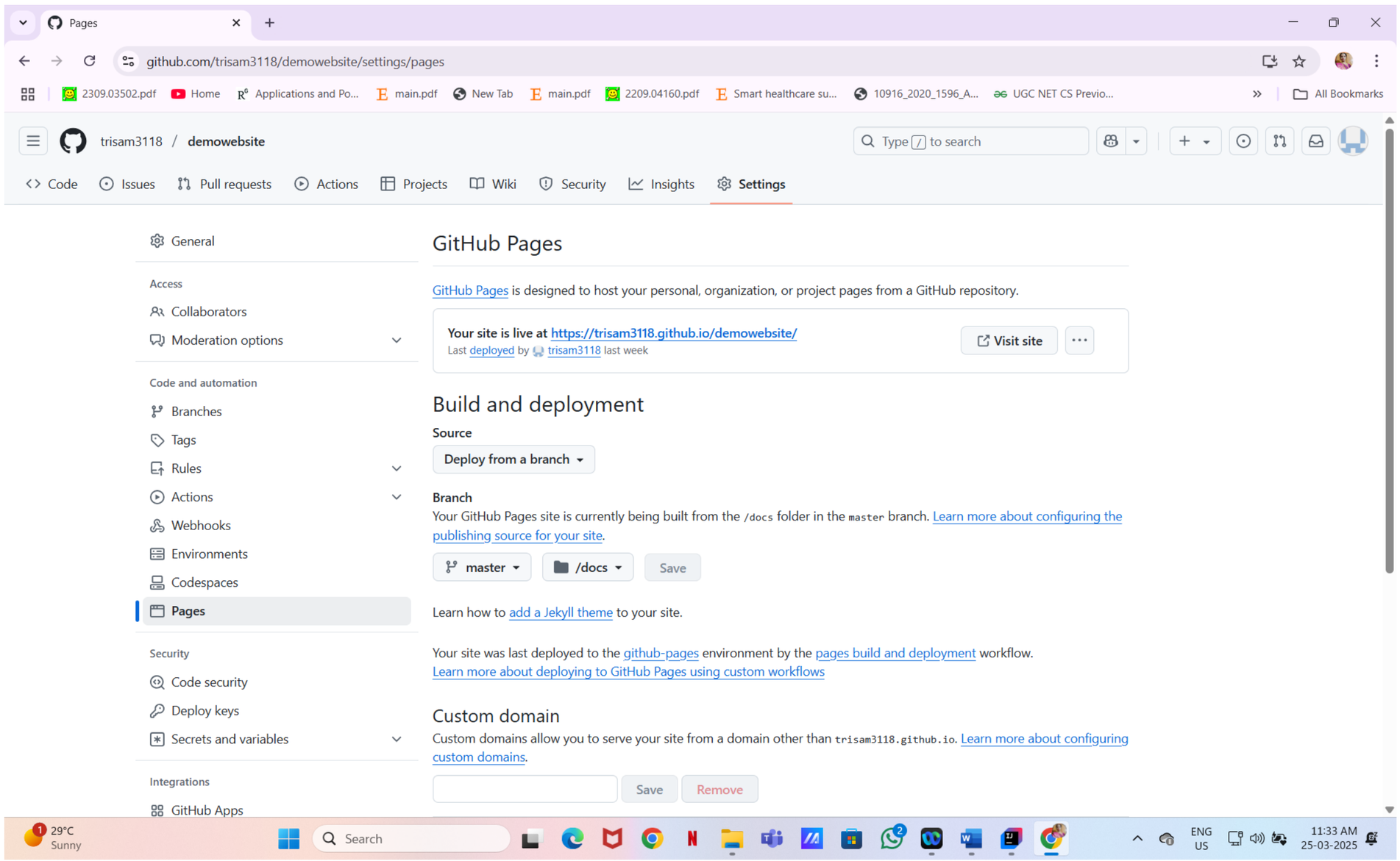
git status

git remote add origin URL

git push -u origin master

In GITHUB → SETTINGS → PAGES → SOURCE (DEPLOY FROM A BRANCH)

BRANCH → MASTER
ROOT →/docs → save



Migrate Maven project to Gradle

POM.XML

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
 <modelVersion>4.0.0</modelVersion>

 <groupId>com.example</groupId>
 <artifactId>FinalMavenToGradle</artifactId>
 <version>1.0-SNAPSHOT</version>
 <packaging>jar</packaging>

 <name>FinalMavenToGradle</name>
 <url>http://maven.apache.org</url>

 <properties>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
```

```
</properties>

<dependencies>
 <dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>3.8.1</version>
  <scope>test</scope>
 </dependency>
</dependencies>
<build>
<plugins>
<!-- Compiler Plugin -->
<plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-compiler-plugin</artifactId>
<version>3.8.1</version>
<configuration>
<source>1.8</source>
<target>1.8</target>
</configuration>
</plugin>
 <!-- Jar Plugin -->
 <plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-jar-plugin</artifactId>
  <version>3.2.0</version>
  <configuration>
   <archive>
    <manifest>
     <mainClass>com.example.App</mainClass>
    </manifest>
   </archive>
  </configuration>
 </plugin>
</plugins>
</build>
</project>
```

Build.gradle

```
plugins {
    id 'java'
}

group = 'com.example'
version = '1.0-SNAPSHOT'

repositories {
    mavenCentral()
}

dependencies {
    testImplementation 'junit:junit:4.13.2'
}
jar {
    manifest {
        attributes(
            'Main-Class': 'com.example.App'
        )

    }
}
```

App.java

```
package com.example;

/**
 * Hello world!
 *
 */
public class App
{
    public static void main( String[] args )
    {
        System.out.println( "Hello from Gradle!" );
    }
}
```

In Terminal run these commands

1. mvn clean install
2. mvn package
3. java -jar .\target\FinalMavenToGradle-1.0-SNAPSHOT.jar

   Output- Hello from Gradle!
4. gradle init --type pom
5. Select build script DSL:

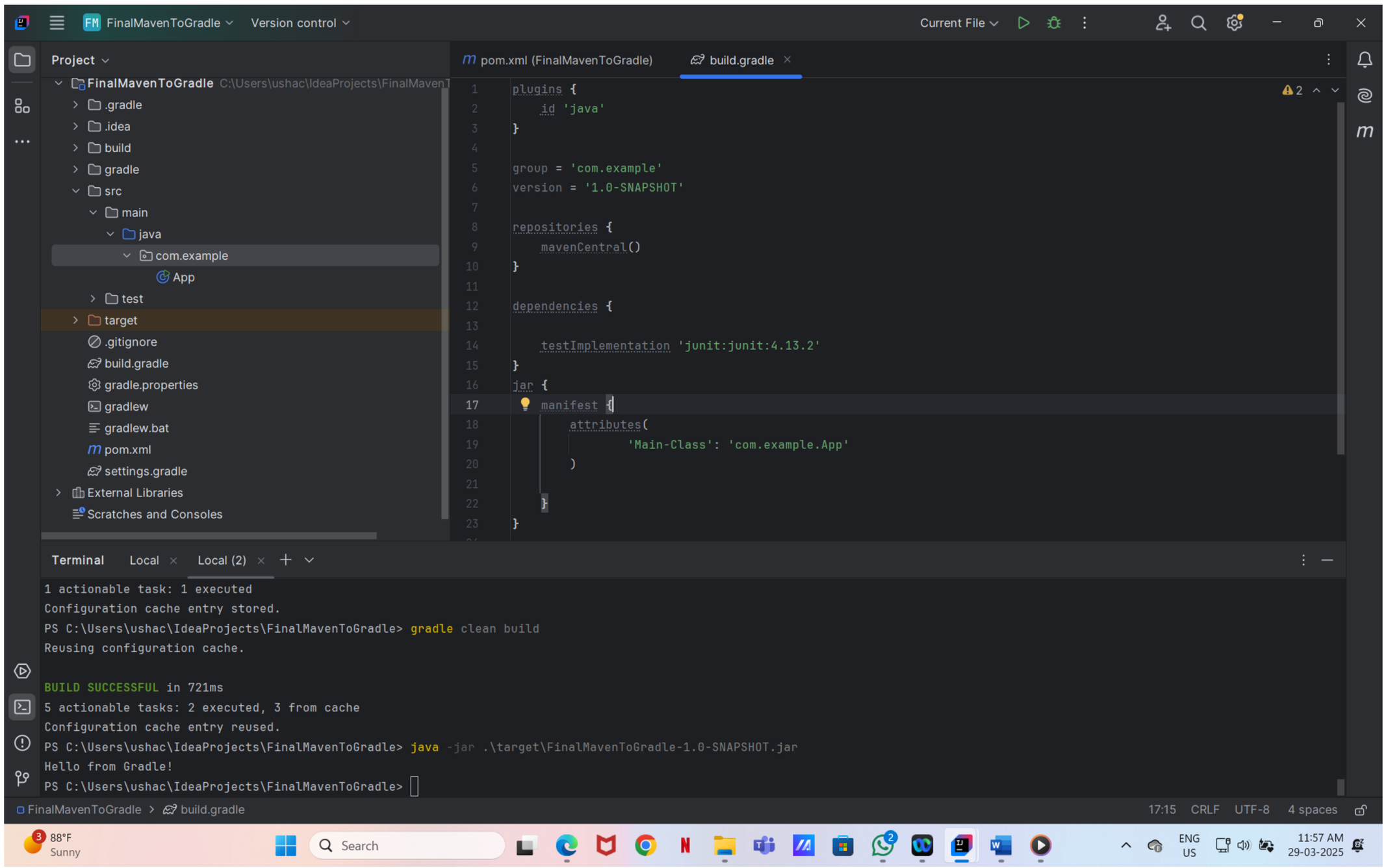
   1: Kotlin

   2: Groovy

   Select 2
6. Generate build using new APIs and behavior (some features may change in the next minor release)? (default: no) [yes, no]

   Yes
7. gradle clean build
8. java -jar .\target\FinalMavenToGradle-1.0-SNAPSHOT.jar

   Output – Hello from Gradle!



Gradle Project using Kotlin

File → New → project → select Gradle → choose → Kotlin DSL (it will generate buld.gradle.kts)→ Name the project → set JDK 17 → FINISH

buld.gradle.kts

```
import org.jetbrains.kotlin.gradle.tasks.KotlinCompile
plugins {
    kotlin("jvm") version "1.8.10"  // Use latest stable Kotlin version
    application
}
group = "org.example"
version = "1.0-SNAPSHOT"
repositories {
    mavenCentral()
}

dependencies {
    implementation(kotlin("stdlib"))  // Kotlin Standard Library
```

```
    testImplementation("org.junit.jupiter:junit-jupiter-api:5.8.2")

    testRuntimeOnly("org.junit.jupiter:junit-jupiter-engine:5.8.2")
}

tasks.test {
    useJUnitPlatform()
}

tasks.withType<KotlinCompile> {
    kotlinOptions.jvmTarget = "17"  // Match with your JDK version
}
tasks.register<Jar>("fatJar") {
    archiveClassifier.set("all")
    duplicatesStrategy = DuplicatesStrategy.EXCLUDE
    manifest {
        attributes["Main-Class"] = "MainKt"
    }
    from(configurations.runtimeClasspath.get().map { if (it.isDirectory) it else zipTree(it) })
    with(tasks.jar.get() as CopySpec)
}
tasks.register("hello") {
    doLast {
        println("Hello, Usha!")
    }
}


    application {
    mainClass.set("MainKt")  // Update this if using a package
}
```

Main.kt

```
fun main() {
    println("Hello, Gradle with Kotlin DSL")

}
```

Execute these commands in terminal

```
./gradlew build
./gradlew run
```

Output -

> Task :run

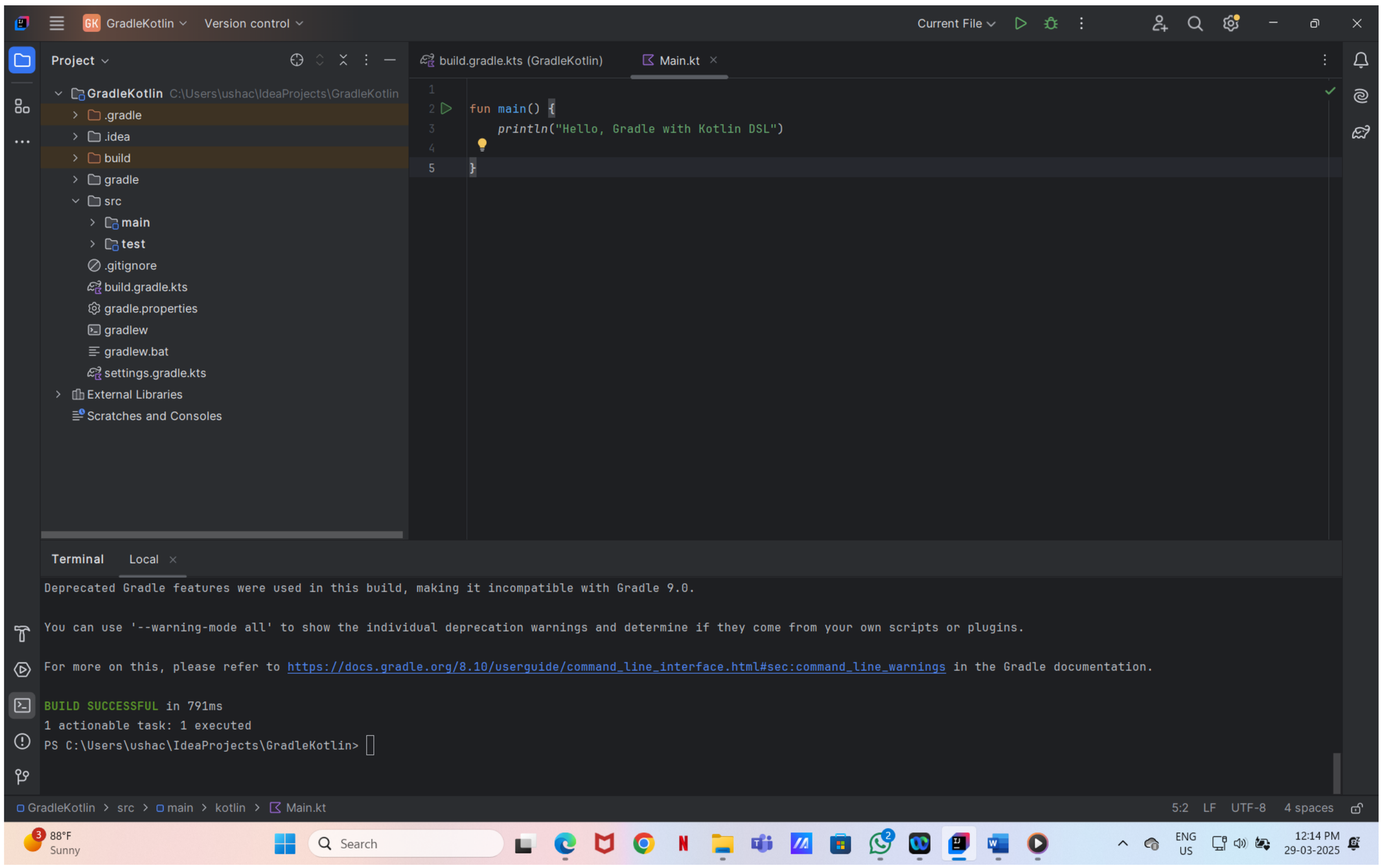
Hello, Gradle with Kotlin DSL

```
./gradlew fatJar
java -jar .\build\libs\GradleKotlin-1.0-SNAPSHOT.jar
./gradlew hello
```

Output-

> Task :hello

Hello, Usha!



Experiment 4

What is Jenkins? Installation of Jenkins on local host, Configuring Jenkins, setting up a CI pipeline, Integrating Jenkins with Maven or Gradle running automated builds and tests.

1. **Jenkins is an open-source automation server used for:**

   ✅ Continuous Integration (CI) – Automatically testing and integrating code changes

   ✅Continuous Deployment (CD) – Automating application deployment

   ✅Building Pipelines – Managing end-to-end software development workflows

   ✅ Plugin-Based Extensibility – Supporting tools like Maven, Gradle, Ansible, Docker, and Azure DevOps

2. **Why Use Jenkins?**

   ✔Automates builds and tests

   ✔Reduces manual intervention

   ✔Improves software quality

   ✔Works with multiple tools and platforms

3. **Installing Jenkins: Jenkins can be installed using multiple methods -**

   1️⃣Windows Installer (.msi) – Recommended for Windows

   2️⃣Linux Package Manager – Best for Linux Users

   3️⃣Jenkins WAR File – Universal method using Java

Steps to install Jenkins in windows

✅ Step 1: Download Jenkins

🔗 Download from: Download and deploy Choose Windows Installer (.msi) for an easy setup.

✅ Step 2: Install Jenkins

1️⃣ Run the downloaded .msi file.

2️⃣ Follow the installation wizard.

3️⃣ Select Run Jenkins as a Windows Service (recommended).

4️⃣ Choose the installation directory (default: C:\Program Files\Jenkins).

5️⃣ Click Install and wait for the setup to complete.

✅ Step 3: Start Jenkins

1️⃣ Open Services (services.msc) and ensure Jenkins is running.

2️⃣ Open a web browser and go to: 1 http://localhost:8080

✅ Step 4: Unlock Jenkins

1️⃣ Find the initial Admin Password in:

1. C:\Program Files\Jenkins\secrets\initialAdminPassword
2. Copy the password and paste it into the Jenkins setup page.

✅ Step 5: Install Recommended Plugins Jenkins will prompt you to install plugins. Click "Install Suggested Plugins".

✅ Step 6: Create Admin User

1️⃣ Set up a Username, Password, and Email.

2️⃣ Click Save and Finish.

Jenkins is now ready! Access it anytime at: 1 http://localhost:8080

Configuring Jenkins for First Use

✅ Understanding the Jenkins Dashboard After logging in, you will see:

- 🔹 New Item → Create Jobs/Pipelines
- 🔹 Manage Jenkins → Configure System, Users, and Plugins
- 🔹 Build History → View previous builds
- 🔹 Credentials → Store secure authentication details

✅ Installing Additional Plugins Jenkins supports plugins for various tools like Maven, Gradle, Docker, and Azure DevOps.

- 🔹 **To install a plugin:**

1️⃣ Go to Manage Jenkins → Manage Plugins

2️⃣ Search for the required plugin

3️⃣ Click Install without Restart

✅Setting Up Global Tool Configuration Configure Java, Maven, and Gradle in Jenkins:

1️⃣ Go to Manage Jenkins → Global Tool Configuration

2️⃣ Add paths for: JDK (C:\Program Files\Java\jdk-17)

Maven (C:\Maven\apache-maven-) Gradle (C:\Gradle\gradle-)

3️⃣ Click Save

Continuous Integration with Jenkins

1. Configuring Jenkins & Git Integration

Step 1: Verify Git Installation in Jenkins

1. Open Jenkins Dashboard → Manage Jenkins → Global Tool Configuration.
2. Under Git, verify the installation path (e.g., C:\Program Files\Git\bin\git.exe).
3. Click Save.

Step 2: Add GitHub Credentials in Jenkins

1. Navigate to Manage Jenkins → Manage Credentials.
2. Select Global credentials (unrestricted) → Click Add Credentials.
3. Choose Username with password or SSH Key, provide details, and click OK

2. Running a Selenium Java Test from a Local Maven Project

Step 1: Create a New Jenkins Job

1. Go to Jenkins Dashboard → Click New Item.
2. Enter a project name → Select Freestyle Project.
3. Click OK.

Step 2: Configure the Build Step

1. Scroll to Build → Click Add build step → Execute Windows Batch Command.
2. Enter the following commands (ensure correct navigation to project directory):

```
1 cd D:\Idea Projects\MVNGRDLDEMO
2 mvn test
```

3 Click Save → Click Build Now to execute the test.

3. Running Selenium Tests from a GitHub Repository via Jenkins

Step 1: Set Up a New Jenkins Job for GitHub Project

1.Go to Jenkins Dashboard → Click New Item.

2. Enter a project name → Select Freestyle Project.

3. Click OK.

Step 2: Configure Git Repository in Jenkins

1. Under Source Code Management, select Git.

2. Enter your GitHub repository URL (e.g., https://github.com/your-repo-name.git).
3. Select the Git credentials configured earlier.

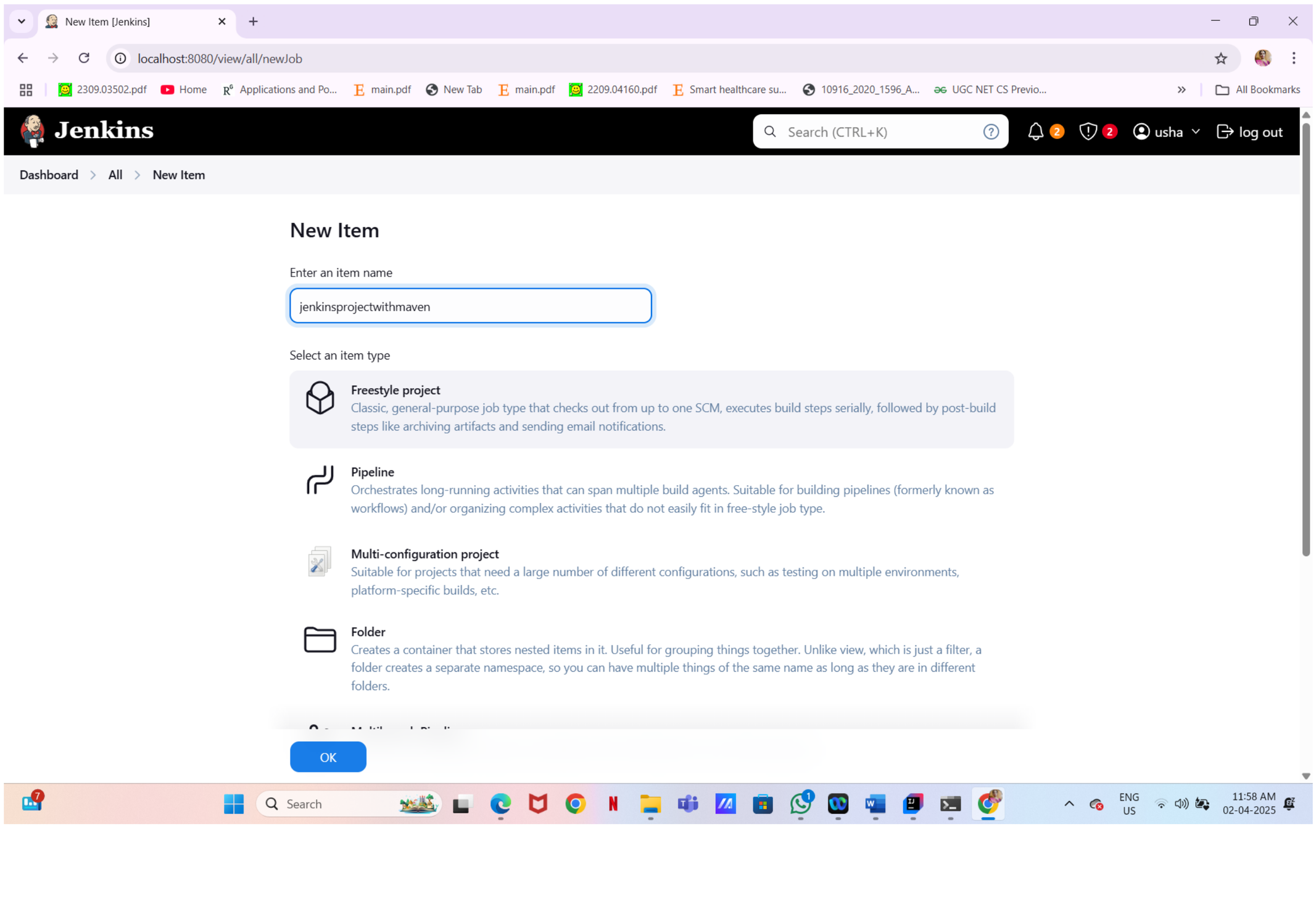
Step 3: Add Build Step for Maven

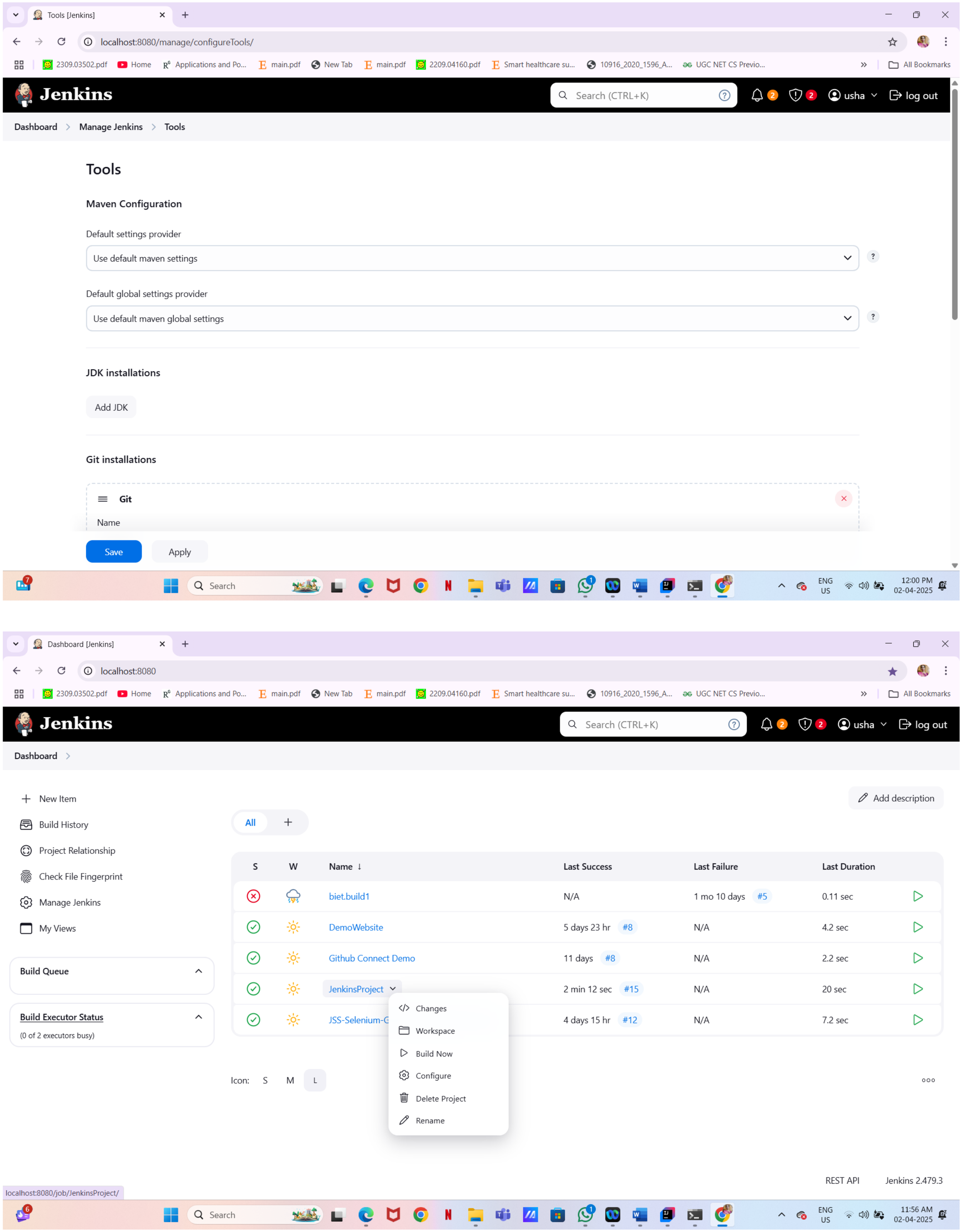
1. Scroll to Build → Click Add build step → Execute Windows Batch Command.
2. Enter the Maven test command: 1 mvn test 2
3. Click Save.

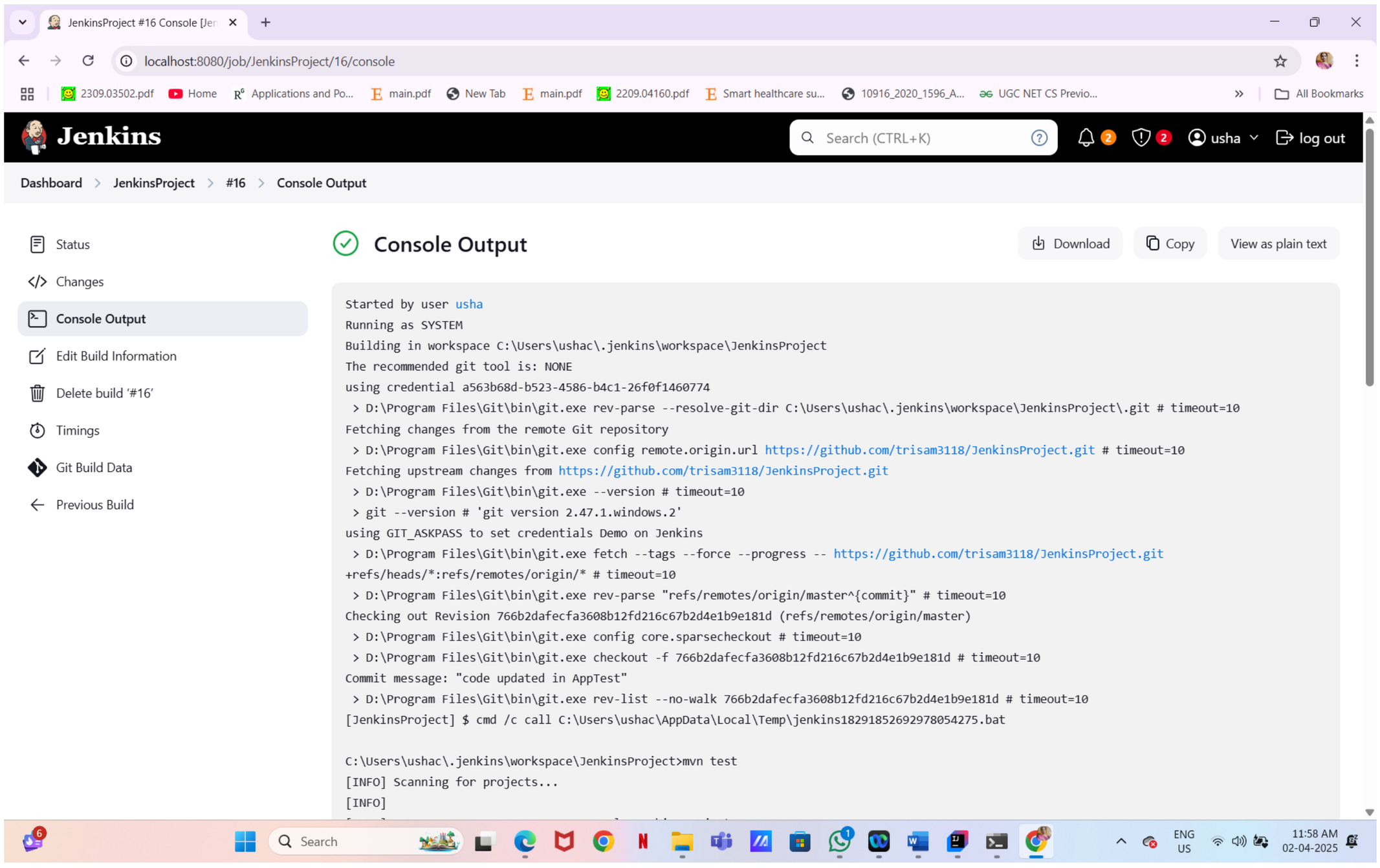
Step 4: Trigger the Build

1. Click Build Now to fetch the code from GitHub and execute the Selenium tests.

2. Check the Console Output to verify test execution.




Tools [Jenkins]
localhost:8080/manage/configureTools/
Jenkins
Search (CTRL+K)
usha
log out
Dashboard > Manage Jenkins > Tools
Tools
Maven Configuration
Default settings provider
Use default maven settings
Default global settings provider
Use default maven global settings
JDK installations
Add JDK
Git installations
Git
Name
Save
Apply
Dashboard [Jenkins]
localhost:8080
Dashboard
New Item
Build History
Project Relationship
Check File Fingerprint
Manage Jenkins
My Views
Build Queue
Build Executor Status
(0 of 2 executors busy)
Add description
All
S
W
Name
Last Success
Last Failure
Last Duration
biet.build1
N/A
1 mo 10 days #5
0.11 sec
DemoWebsite
5 days 23 hr #8
N/A
4.2 sec
Github Connect Demo
11 days #8
N/A
2.2 sec
JenkinsProject
2 min 12 sec #15
N/A
20 sec
JSS-Selenium-G
4 days 15 hr #12
N/A
7.2 sec
Changes
Workspace
Build Now
Configure
Delete Project
Rename
Icon: S M L
REST API
Jenkins 2.479.3
localhost:8080/job/JenkinsProject/



Pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.example</groupId>
  <artifactId>JenkinsProject</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>JenkinsProject</name>
  <url>http://maven.apache.org</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>
```

```
<dependencies>
 <dependency>
  <groupId>org.testng</groupId>

  <artifactId>testng</artifactId>
  <version>7.10.2</version>
  <scope>test</scope>
 </dependency>
 <dependency>
  <groupId>org.seleniumhq.selenium</groupId>
  <artifactId>selenium-java</artifactId>
  <version>4.25.0</version>
 </dependency>
 <dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>3.8.1</version>
  <scope>test</scope>
 </dependency>
</dependencies>
<build>
 <plugins>
     <plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-resources-plugin</artifactId>
    <version>3.2.0</version>
    <executions>
     <execution>
     <phase>prepare-package</phase> <!-- Before packaging -->
<goals>
 <goal>copy-resources</goal>
</goals>
 <configuration>
  <outputDirectory>${project.basedir}/docs</outputDirectory>
  <resources>
   <resource>
    <directory>src/main/resources</directory>
    <includes>
     <include>**/*</include> <!-- Copy all files in src/main/resources -->
    </includes>
   </resource>
```

```
        </resources>
    </configuration>
   </execution>
  </executions>
</plugin>
</plugins>
</build>
</project>
```

Now, **index.html, style.css and logo.png files** from **docs** folder is added to your Jenkins Project

Apptest.java

```
package org.test;
import org.openqa.selenium.WebDriver;
 import org.openqa.selenium.chrome.ChromeDriver;
 import org.testng.Assert;
 import org.testng.annotations.AfterTest;
 import org.testng.annotations.BeforeTest;
 import org.testng.annotations.Test;
 import static org.testng.Assert.assertTrue;

public class WebpageTest {
    private static WebDriver driver;

@BeforeTest
public void openBrowser() throws InterruptedException {
    driver = new ChromeDriver();
    driver.manage().window().maximize();
    Thread.sleep(2000);
    driver.get("https://trisam3118.github.io/demowebsite/");// here enter static website URL from
your GITHUB
}

@Test
public void titleValidationTest(){
    String actualTitle = driver.getTitle();
    String expectedTitle = "Computer Science and Engineering(Data Science)";
    Assert.assertEquals(actualTitle, expectedTitle);
    assertTrue(true, "Title should contain 'Computer Science and Engineering(Data Science)'");
```

```
}

@AfterTest
public void closeBrowser() throws InterruptedException {
    Thread.sleep(1000);
    driver.quit();
}

}
```

Note: Run this file it should open browser and validates the title.