Name –Kamal D. Agrahari **TE IT - A | Batch – B**
ID – VU4F2223028 **Subject – DevOps Lab**

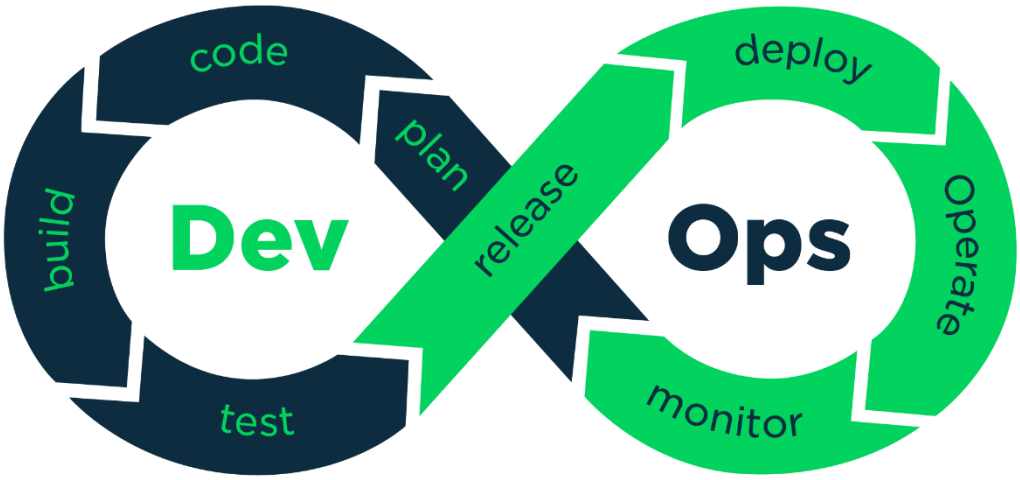
Experiment No. 1

Aim: Case Study on understand DevOps: Principles, Practices and DevOps Engineer Role and Responsibilities.

What is DevOps?

DevOps is a set of practices, principles, and cultural philosophies that combines software development (Dev) and IT operations (Ops). The primary goal of DevOps is to shorten the software development lifecycle and deliver high-quality software continuously. Key aspects of DevOps include:

Collaboration and Communication: Encouraging close cooperation between development and operations teams to improve workflow and productivity.
Automation: Automating repetitive and manual tasks such as testing, integration, deployment, and monitoring to increase efficiency and reduce errors.
Continuous Integration (CI): Frequently integrating code changes into a shared repository, followed by automated testing to detect issues early.
Continuous Delivery (CD): Ensuring that code changes are automatically tested and prepared for a release to production, enabling faster and more reliable software releases.
Infrastructure as Code (IaC): Managing and provisioning computing infrastructure through machine-readable scripts, rather than through manual processes.
Monitoring and Logging: Continuously monitoring applications and infrastructure to gain insights into performance and detect issues proactively.
Culture: Fostering a culture of collaboration, learning, and shared responsibility among all stakeholders involved in the software delivery process.



DevOps principles guide the integration of development and operations to achieve efficient and effective software delivery. Here are the key principles

- Culture of Collaboration and Communication:
  - Foster a culture where development, operations, and other stakeholders work together.
  - Promote open communication, shared goals, and a sense of shared responsibility.

- Automation:
  - Automate repetitive tasks to increase efficiency and reduce human error.
  - Use tools for automated testing, integration, deployment, and monitoring.

- Continuous Integration (CI):
  - Frequently merge code changes into a central repository.
  - Automatically test each integration to detect issues early.

- Continuous Delivery (CD):
  - Ensure that code is always in a deployable state.
  - Automate the release process to deliver software quickly and reliably.

- Infrastructure as Code (IaC):
  - Manage infrastructure using code and automation rather than manual processes.
  - Treat infrastructure configuration as software that can be versioned and tested.

- Monitoring and Logging:
  - Continuously monitor applications and infrastructure to gain insights into performance.
  - Use logs and metrics to identify and troubleshoot issues proactively.

- Lean and Agile Principles:
  - Apply lean principles to eliminate waste and improve processes.
  - Adopt agile methodologies for iterative development and continuous feedback.

- Measurement and Improvement:
  - Measure performance using key metrics to identify areas for improvement.
  - Continuously evaluate and refine processes to enhance efficiency and quality.

- Customer-Centric Action:
  - Focus on delivering value to customers quickly and efficiently.
  - Gather and act on customer feedback to improve the product continuously.

- Security:
  - Integrate security practices into the DevOps workflow.
  - Ensure that security is a shared responsibility among all team members (DevSecOps).

DevOps Model

Collaborative Culture:

Development and operations teams work together throughout the software development lifecycle (SDLC). Shared goals, open communication, and mutual accountability are emphasized.

Automation:

Automate as many processes as possible, including testing, integration, deployment, and monitoring.
Use tools and scripts to streamline workflows and reduce manual intervention.

Continuous Integration and Continuous Delivery (CI/CD):

Continuous Integration (CI): Developers frequently merge code changes into a shared repository. Each change is automatically tested to detect integration issues early.
Continuous Delivery (CD): Ensures that the code is always in a deployable state, allowing for frequent, reliable releases to production.

Infrastructure as Code (IaC):

Manage and provision infrastructure through code and automation tools.
Treat infrastructure configurations as code that can be versioned, tested, and reproduced.

Monitoring and Logging:

Continuously monitor applications and infrastructure for performance and availability.
Collect and analyze logs and metrics to gain insights and proactively address issues.

Feedback Loops:

Implement mechanisms for continuous feedback from development, operations, and end-users.
Use feedback to drive improvements in processes, tools, and product quality.

DevOps Practices

Version Control:

Use version control systems like Git to manage code changes.
Ensure all changes are tracked, and collaboration is facilitated through branching and merging.

Automated Testing:

Implement automated testing at various stages (unit, integration, system, and acceptance testing) to ensure code quality.
Use testing frameworks and tools to automate the execution and reporting of tests.

Continuous Integration:

Set up CI pipelines that automatically build and test code whenever changes are committed.
Use CI tools like Jenkins, Travis CI, or CircleCI to automate the process.

Continuous Delivery/Continuous Deployment:

Implement CD pipelines to automate the release process, ensuring that code is always in a deployable state.
Use tools like Jenkins, GitLab CI/CD, or Spinnaker for continuous delivery and deployment.

Configuration Management:
Use configuration management tools like Ansible, Puppet, or Chef to automate the setup and maintenance of infrastructure.
Ensure that configuration is consistent and reproducible across environments.

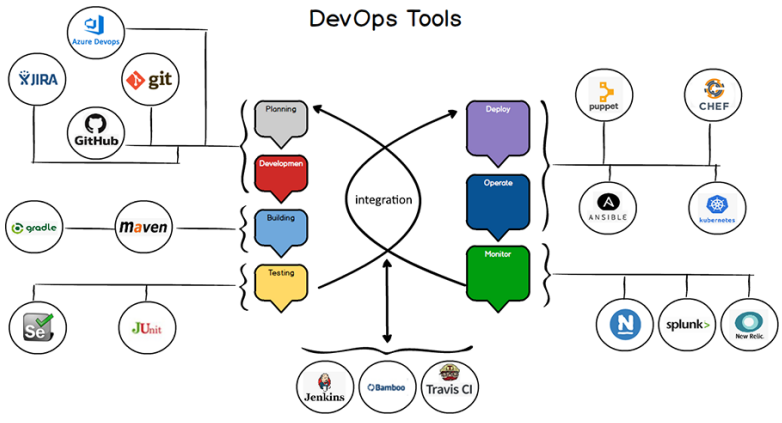
Infrastructure as Code (IaC):
Define and manage infrastructure using code and automation tools like Terraform, AWS CloudFormation, or Azure Resource Manager.
Enable version control, testing, and automation of infrastructure changes.

Monitoring and Logging:
Implement monitoring tools like Prometheus, Grafana, or Nagios to track system performance and health.
Use logging tools like ELK Stack (Elasticsearch, Logstash, Kibana), Splunk, or Fluentd to collect and analyze log data.

Containerization:
Use containerization technologies like Docker to package applications and their dependencies into containers.
Orchestrate containers using Kubernetes, OpenShift, or Docker Swarm for scalable and manageable deployments.

Security Integration (DevSecOps):
Integrate security practices into the DevOps workflow.
Use automated security testing, vulnerability scanning, and compliance checks as part of the CI/CD pipeline.

Collaboration and Communication Tools:
Use collaboration tools like Slack, Microsoft Teams, or Jira to enhance communication and project management.
Implement practices like daily stand-ups, retrospectives, and continuous feedback to foster a collaborative culture.



DevOps Engineer plays a crucial role in bridging the gap between development and operations teams to streamline and automate the processes involved in the software development lifecycle. Here are the key roles and responsibilities of a DevOps Engineer

Roles

Automation Expert:
Develop and maintain automation scripts and tools for building, testing, and deploying software.
Automate repetitive tasks to improve efficiency and reduce human error.

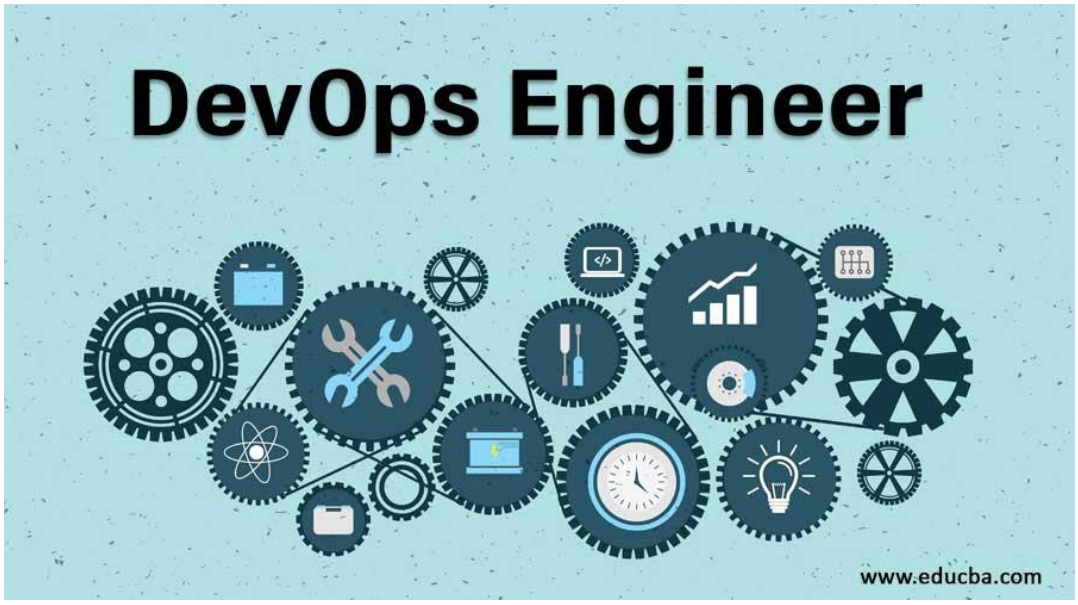
CI/CD Pipeline Developer:
Design, implement, and manage continuous integration and continuous delivery (CI/CD) pipelines.
Ensure that code changes are automatically tested and deployed to production environments.

Infrastructure Manager:
Use Infrastructure as Code (IaC) tools to manage and provision infrastructure.
Ensure that infrastructure is scalable, reliable, and secure.

Monitoring and Logging Specialist:
Set up and manage monitoring and logging systems to track application performance and system health.
Analyze logs and metrics to proactively identify and resolve issues.

Security Integrator:
Incorporate security best practices into the DevOps process (DevSecOps).
Implement automated security testing and vulnerability scanning.

Collaboration Facilitator:
Foster a culture of collaboration and communication between development, operations, and other stakeholders.
Ensure that all teams are aligned and working towards common goals.



Responsibilities

Build and Maintain CI/CD Pipelines:
Develop and manage CI/CD pipelines to automate the build, test, and deployment processes.
Ensure that pipelines are efficient, reliable, and scalable.

Automate Infrastructure Management:
Use IaC tools like Terraform, Ansible, or CloudFormation to automate the provisioning and management of infrastructure.
Ensure that infrastructure is consistent across development, testing, and production environments.

Monitor Systems and Applications:
Set up monitoring tools like Prometheus, Grafana, or Nagios to track the performance and availability of systems and applications.
Implement alerting mechanisms to notify relevant teams of issues.

Manage Configuration and Deployment:
Use configuration management tools like Chef, Puppet, or Ansible to manage system configurations.
Ensure that deployments are automated, reliable, and repeatable.

Implement and Enforce Security Practices:
Integrate security practices into the DevOps workflow, ensuring that security is a shared responsibility.
Conduct regular security assessments and vulnerability scans.

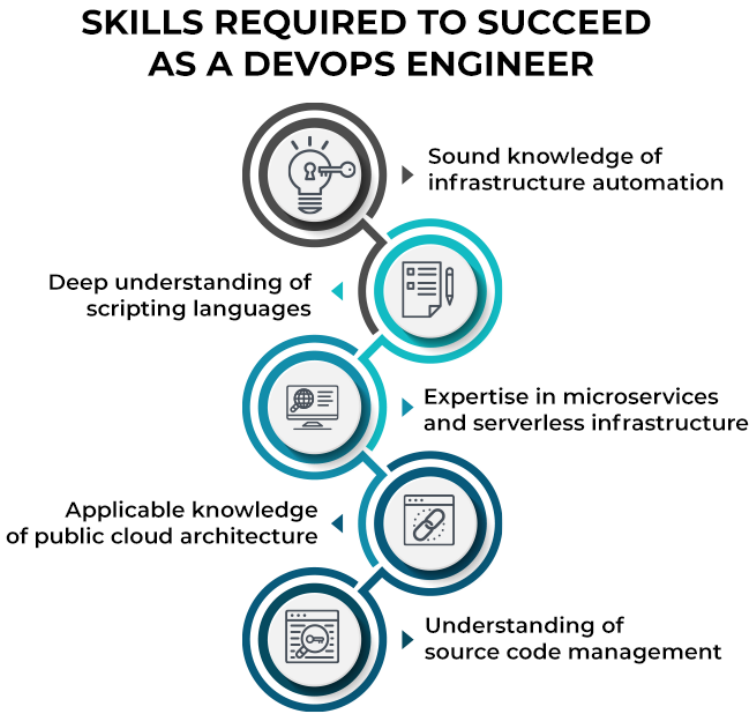
Collaborate with Development and Operations Teams:
Work closely with developers to understand application requirements and dependencies.
Collaborate with operations teams to ensure that infrastructure meets performance and availability requirements.

Optimize Performance and Efficiency:
Continuously evaluate and optimize processes, tools, and workflows to improve efficiency and performance.
Identify bottlenecks and implement solutions to address them.

Troubleshoot and Resolve Issues:
Investigate and resolve issues related to CI/CD pipelines, infrastructure, and application performance.
Use logs, metrics, and monitoring data to diagnose and fix problems.

Document Processes and Procedures:
Maintain comprehensive documentation of CI/CD pipelines, automation scripts, and infrastructure configurations.
Ensure that documentation is up-to-date and accessible to relevant teams.

Stay Updated with Industry Trends:
Keep abreast of the latest DevOps tools, technologies, and best practices.
Continuously learn and adopt new approaches to improve the DevOps process.



Name – Kamal D. Agrahari
ID -VU4F2223028

Subject – DevOps Lab
TE IT A | Batch B

Experiment No. 2

Aim - Study of Version Control System/Source Code Management, install git and create a GitHub Account.

Theory –

Version Control:

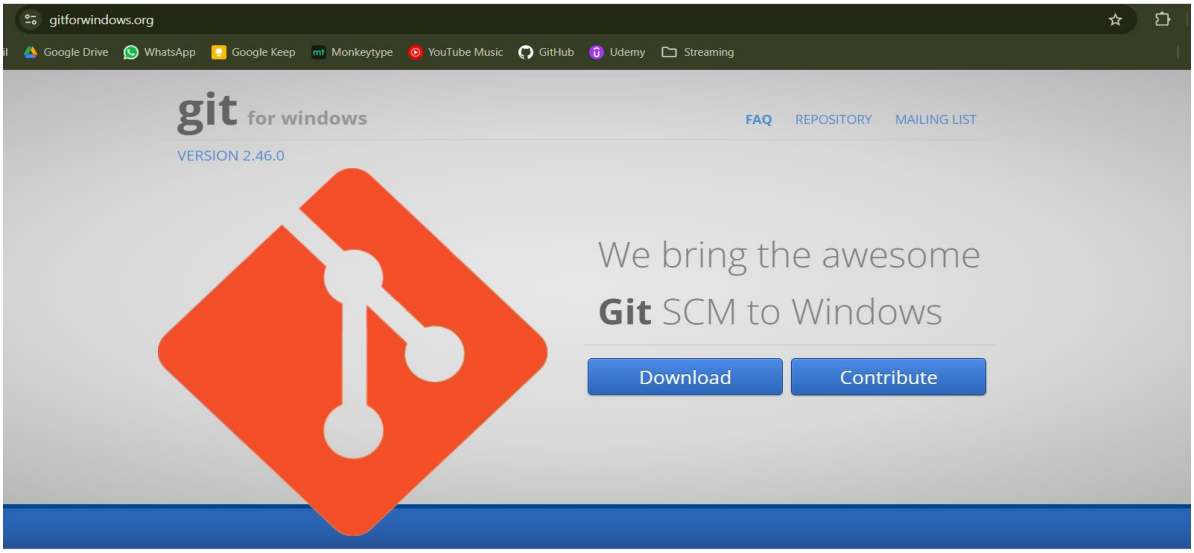
Version control is a system that manages changes to a project's files and directories over time. It allows multiple people to work on a project simultaneously, tracks history, and facilitates collaboration. Key features include:

1. Tracking Changes: Records modifications to files, making it possible to revert to previous versions if necessary.
2. Collaboration: Multiple contributors can work on the same project simultaneously without overwriting each other's work.
3. Branching and Merging: Facilitates experimentation by allowing users to create branches (copies of the project) and later merge changes back into the main branch.
4. Backup and Restore: Provides a backup of all versions of a project, making it easier to recover from data loss or corruption.
5. Code Review: Helps in peer reviewing code changes before integrating them into the main codebase.

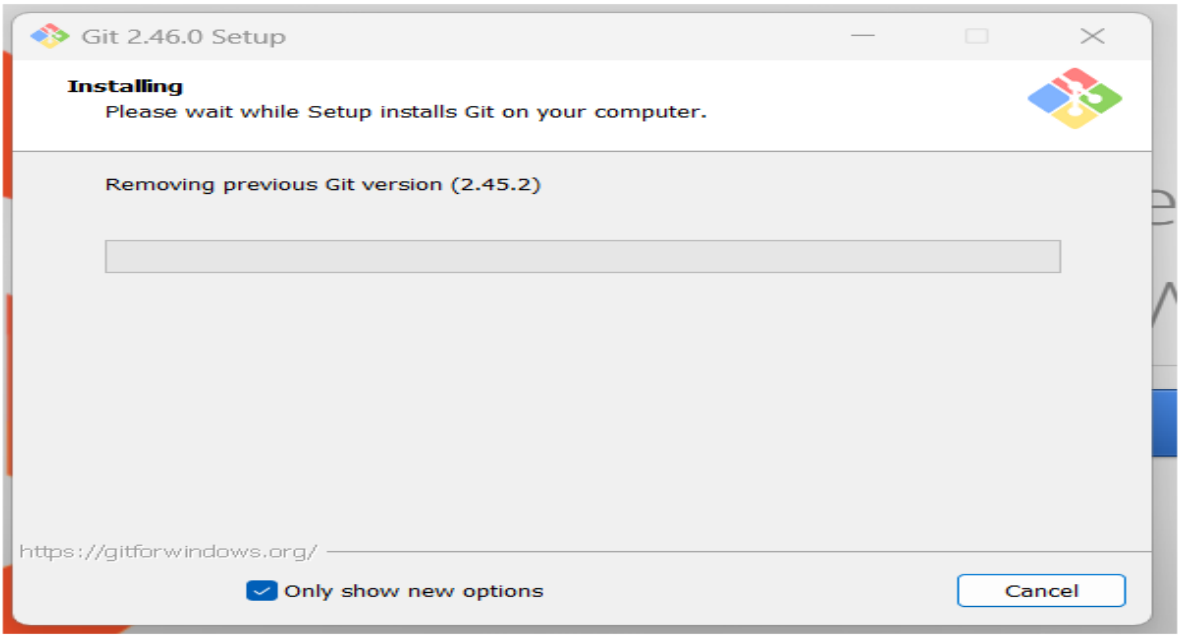
Popular version control systems include Git, Subversion (SVN), and Mercurial.

Steps for Installing of Git

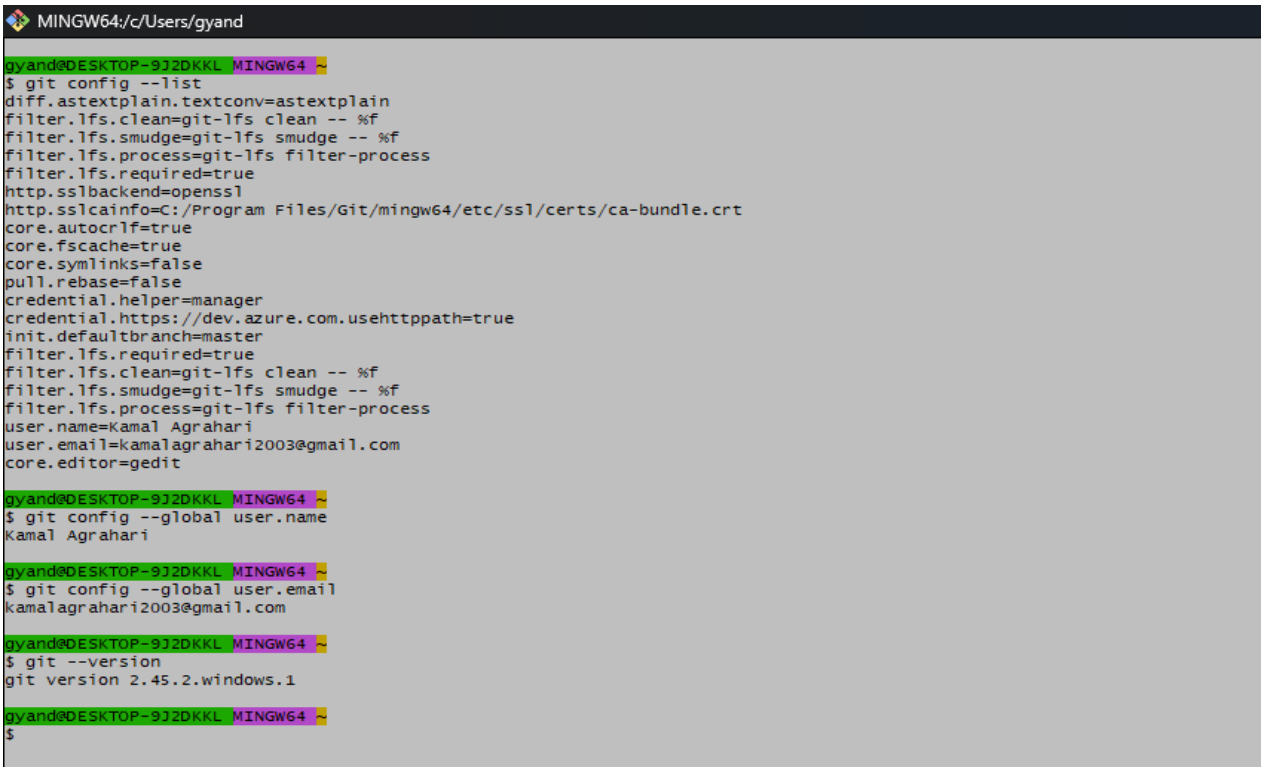
1. Go to the gitforwindows.org website then click on the DOWNLOAD button to download the installer

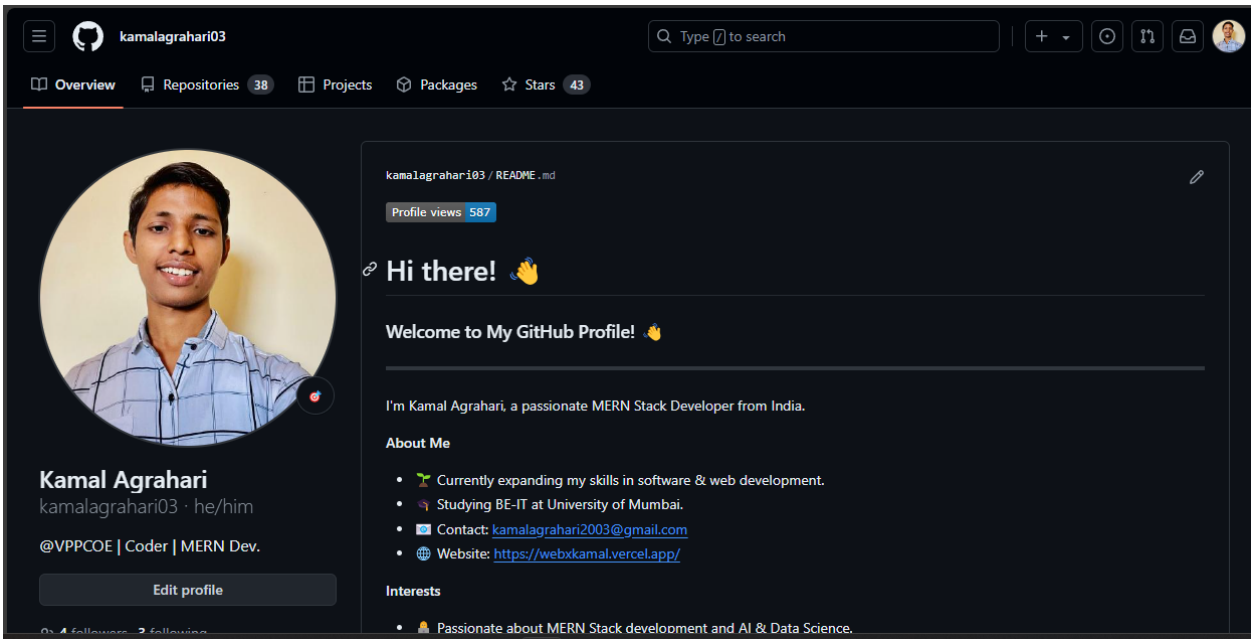


2. Run the installer and go throught the setup to install the git on your computer. It may ask some permissions click YES to allow the installer to run



3. See if git bash is installed, type command git –version to verify the installation




kamalagrahari03
Type / to search
Overview
Repositories 38
Projects
Packages
Stars 43
Kamal Agrahari
kamalagrahari03 · he/him
@VPPCOE | Coder | MERN Dev.
Edit profile
kamalagrahari03 / README.md
Profile views 587
Hi there! 👋
Welcome to My GitHub Profile! 👋
I'm Kamal Agrahari, a passionate MERN Stack Developer from India.
About Me
Currently expanding my skills in software & web development.
Studying BE-IT at University of Mumbai.
Contact: kamalagrahari2003@gmail.com
Website: https://webxkamal.vercel.app/
Interests
Passionate about MERN Stack development and AI & Data Science.

Name – Kamal D. Agrahari TE-IT - A
ID – VU4F2223028 Batch – B
Sub – DevOps Lab

Experiment 03

Aim - Demonstration of various git operations on local and remote repository using git cheat-sheet.

Theory –

Some git commands are –

1. git --version: Displays the installed version of Git.
2. mkdir <dir name>: Creates a new directory named.
3. git init : Initialize the new git repository in the current directory.
4. nano <file name> : Opens the nano text editor to create or edit the file.
5. git status : Shows the status of the working directory and staging area, including untracked files.
6. git add –all : Adds all changes in the working directory to the staging area.
7. git commit -m "some message" : Commits the staged changes with the some message.
8. git remote add origin <url> : Adds a remote repository named origin with the specified URL.
9. git push -u origin main : Pushes the committed changes to the main branch of the remote repository and sets the upstream branch.
10. git log : Displays the commit history.
11. ls : Lists the files in the current directory.
12. git rm --cached <file name>: Removes the file from the staging area but keeps it in the working directory.

Program

```
Kamal Agrahari@Desktop MINGW64 ~
$ git --version
git version 2.46.0.windows.1

Kamal Agrahari@Desktop MINGW64 ~
$ mkdir exp3

Kamal Agrahari@Desktop MINGW64 ~
$ cd exp3/

Kamal Agrahari@Desktop MINGW64 ~/exp3
$ git init
Initialized empty Git repository in C:/Users/Kamal Agrahari/ exp3.git/

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ ls -a
./  ../  .git/

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
```

```
$ nano school.txt college.txt

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git add .
warning: in the working copy of 'college.txt', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'school.txt', LF will be replaced by CRLF the next time Git touches it

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git commit -m "Created files school.txt and college.txt"
[master (root-commit) c0f6c34] Created files school.txt and college.txt
 2 files changed, 2 insertions(+)
 create mode 100644 college.txt
 create mode 100644 school.txt

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git remote add origin https://github.com/Kamalagrahari03/devops-exp3.git

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git push origin master
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 346 bytes | 346.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Kamalagrahari03/devops-exp3.git
 * [new branch]      master -> master

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git status
On branch master
nothing to commit, working tree clean

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git log
commit c0f6c34a528c5108e79f17287071933ba0f9f3d5 (HEAD -> master, origin/master)
Author: Kamal Agrahari <kamalagrahari2003@gmail.com>
Date:   Fri Aug 2 09:28:48 2024 +0530

    Created files school.txt and college.txt

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ ls -a
./  ../  .git/  college.txt  school.txt

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git rm --cached school.txt
```

```
rm 'school.txt'

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    school.txt

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


Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git commit -m "Deleted the file school.txt"
[master d9e4f5f] Deleted the file school.txt
 1 file changed, 1 deletion(-)
 delete mode 100644 school.txt

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git push origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 16 threads
Compressing objects: 100% (1/1), done.
Writing objects: 100% (2/2), 254 bytes | 254.00 KiB/s, done.
Total 2 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Kamalagrahari03/devops-exp3.git
   c0f6c34..d9e4f5f  master -> master

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        school.txt

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

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git log
commit d9e4f5f206d678a739891a49ec8015cae2bf18b8 (HEAD -> master, origin/master)
Author: Kamal Agrahari <kamalagrahari2003@gmail.com>
Date:   Fri Aug 2 09:36:30 2024 +0530

    Deleted the file school.txt
```

```
commit c0f6c34a528c5108e79f17287071933ba0f9f3d5
Author: Kamal Agrahari <kamalagrahari2003@gmail.com>
Date:   Fri Aug 2 09:28:48 2024 +0530

    Created files school.txt and college.txt

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git checkout d9e4f school.txt
error: pathspec 'school.txt' did not match any file(s) known to git

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git checkout c0f6c school.txt
Updated 1 path from 7511864

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   school.txt


Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git commit -m "restore the delted file"
[master aab0dda] restore the delted file
 1 file changed, 1 insertion(+)
 create mode 100644 school.txt

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git push origin master
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 326 bytes | 326.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Kamalagrahari03/devops-exp3.git
   d9e4f5f..aab0dda  master -> master

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ ls
college.txt  school.txt

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git status
On branch master
nothing to commit, working tree clean
```

```
Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git log
commit aab0ddaa394ebd84a538d4521b997b8e67271ce8 (HEAD -> master, origin/master)
Author: Kamal Agrahari <kamalagrahari2003@gmail.com>
Date:   Fri Aug 2 09:40:07 2024 +0530

    restore the delted file

commit d9e4f5f206d678a739891a49ec8015cae2bf18b8
Author: Kamal Agrahari <kamalagrahari2003@gmail.com>
Date:   Fri Aug 2 09:36:30 2024 +0530

    Deleted the file school.txt

commit c0f6c34a528c5108e79f17287071933ba0f9f3d5
Author: Kamal Agrahari <kamalagrahari2003@gmail.com>
Date:   Fri Aug 2 09:28:48 2024 +0530

    Created files school.txt and college.txt

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ nano myhistory.txt

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git add .
warning: in the working copy of 'myhistory.txt', LF will be replaced by CRLF the next time Git touches it

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git commit -m "Adding the file myhistory.txt"
[master c94bef3] Adding the file myhistory.txt
 1 file changed, 1 insertion(+)
 create mode 100644 myhistory.txt

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git push origin master
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 345 bytes | 345.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Kamalagrahari03/devops-exp3.git
   aab0dda..c94bef3  master -> master

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git log
commit c94bef371475664fb78ce5c5c724b2d07b98f138 (HEAD -> master, origin/master)
Author: Kamal Agrahari <kamalagrahari2003@gmail.com>
```

```
Date:   Fri Aug 2 09:42:12 2024 +0530

    Adding the file myhistory.txt

commit aab0ddaa394ebd84a538d4521b997b8e67271ce8
Author: Kamal Agrahari <kamalagrahari2003@gmail.com>
Date:   Fri Aug 2 09:40:07 2024 +0530

    restore the delted file

commit d9e4f5f206d678a739891a49ec8015cae2bf18b8
Author: Kamal Agrahari <kamalagrahari2003@gmail.com>
Date:   Fri Aug 2 09:36:30 2024 +0530

    Deleted the file school.txt

commit c0f6c34a528c5108e79f17287071933ba0f9f3d5
Author: Kamal Agrahari <kamalagrahari2003@gmail.com>
Date:   Fri Aug 2 09:28:48 2024 +0530

    Created files school.txt and college.txt

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ nano school.txt myhistory.txt

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   myhistory.txt
        modified:   school.txt

no changes added to commit (use "git add" and/or "git commit -a")

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git diff
warning: in the working copy of 'myhistory.txt', LF will be replaced by CRLF the next time Git touches it
diff --git a/myhistory.txt b/myhistory.txt
index 8c22244..d8e450a 100644
--- a/myhistory.txt
+++ b/myhistory.txt
@@ -1 +1,2 @@
 World war 3
+It happening right now
diff --git a/school.txt b/school.txt
index e25d7b1..f294038 100644
```

```
--- a/school.txt
+++ b/school.txt
@@ -1 +1,2 @@
 Hello from Smt . Ramkalidevi School
+Checkout myhistory text file

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git add .
warning: in the working copy of 'myhistory.txt', LF will be replaced by CRLF the next time Git touches it

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git log --oneline
c94bef3 (HEAD -> master, origin/master) Adding the file myhistory.txt
aab0dda restore the delted file
d9e4f5f Deleted the file school.txt
c0f6c34 Created files school.txt and college.txt

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git commit -m "Updated the file myhistory.txt and school.txt"
[master dbf4c23] Updated the file myhistory.txt and school.txt
 2 files changed, 2 insertions(+)

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git log --oneline
dbf4c23 (HEAD -> master) Updated the file myhistory.txt and school.txt
c94bef3 (origin/master) Adding the file myhistory.txt
aab0dda restore the delted file
d9e4f5f Deleted the file school.txt
c0f6c34 Created files school.txt and college.txt

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git diff HEAD HEAD~
diff --git a/myhistory.txt b/myhistory.txt
index d8e450a..8c22244 100644
--- a/myhistory.txt
+++ b/myhistory.txt
@@ -1,2 +1 @@
 World war 3
-It happening right now
diff --git a/school.txt b/school.txt
index f294038..e25d7b1 100644
--- a/school.txt
+++ b/school.txt
@@ -1,2 +1 @@
 Hello from Smt. Ramkalidevi School
-Checkout myhistory text file

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
```

```
$ git show
commit dbf4c2385576bebe78540695fa1028a53bfe381c (HEAD -> master)
Author: Kamal Agrahari <kamalagrahari2003@gmail.com>
Date:   Fri Aug 2 09:45:24 2024 +0530

    Updated the file myhistory.txt and school.txt

diff --git a/myhistory.txt b/myhistory.txt
index 8c22244..d8e450a 100644
--- a/myhistory.txt
+++ b/myhistory.txt
@@ -1 +1,2 @@
 World war 3
+It happening right now
diff --git a/school.txt b/school.txt
index e25d7b1..f294038 100644
--- a/school.txt
+++ b/school.txt
@@ -1 +1,2 @@
 Hello from Smt. Ramkalidevi School
+Checkout myhistory text file

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git blame school.txt
aab0ddaa (Kamal Agrahari2024-08-02 09:40:07 +0530 1) Hello from Smt. Ramkalidevi School
dbf4c238 (Kamal Agrahari2024-08-02 09:45:24 +0530 2) Checkout myhistory text file

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git reflog
dbf4c23 (HEAD -> master) HEAD@{0}: commit: Updated the file myhistory.txt and school.txt
c94bef3 (origin/master) HEAD@{1}: commit: Adding the file myhistory.txt
aab0dda HEAD@{2}: commit: restore the delted file
d9e4f5f HEAD@{3}: commit: Deleted the file school.txt
c0f6c34 HEAD@{4}: commit (initial): Created files school.txt and college.txt

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git diff HEAD HEAD~
diff --git a/myhistory.txt b/myhistory.txt
index d8e450a..8c22244 100644
--- a/myhistory.txt
+++ b/myhistory.txt
@@ -1,2 +1 @@
 World war 3
-It happening right now
diff --git a/school.txt b/school.txt
index f294038..e25d7b1 100644
--- a/school.txt
+++ b/school.txt
```

```
@@ -1,2 +1 @@
 Hello from Smt. Ramkalidevi School
-Checkout myhistory text file

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git show HEAD
commit dbf4c2385576bebe78540695fa1028a53bfe381c (HEAD -> master)
Author: Kamal Agrahari <kamalagrahari2003@gmail.com>
Date:   Fri Aug 2 09:45:24 2024 +0530

    Updated the file myhistory.txt and school.txt

diff --git a/myhistory.txt b/myhistory.txt
index 8c22244..d8e450a 100644
--- a/myhistory.txt
+++ b/myhistory.txt
@@ -1 +1,2 @@
 World war 3
+It happening right now
diff --git a/school.txt b/school.txt
index e25d7b1..f294038 100644
--- a/school.txt
+++ b/school.txt
@@ -1 +1,2 @@
 Hello from Smt. RamkalideviSchool
+Checkout myhistory text file

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ nano myhistory.txt

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git commit -m "Added the content to myhistory.txt"
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   myhistory.txt

no changes added to commit (use "git add" and/or "git commit -a")

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git add .
warning: in the working copy of 'myhistory.txt', LF will be replaced by CRLF the next time Git touches it

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git commit -m "Added the content to myhistory.txt"
[master ca79b93] Added the content to myhistory.txt
 1 file changed, 1 insertion(+)
```

```
Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git tag -a
usage: git tag [-a | -s | -u <key-id>] [-f] [-m <msg> | -F <file>] [-e]
            [(--trailer <token>[(=|:)<value>])...]
            <tagname> [<commit> | <object>]
  or: git tag -d <tagname>...
  or: git tag [-n[<num>]] -l [--contains <commit>] [--no-contains <commit>]
            [--points-at <object>] [--column[=<options>] | --no-column]
            [--create-reflog] [--sort=<key>] [--format=<format>]
            [--merged <commit>] [--no-merged <commit>] [<pattern>...]
  or: git tag -v [--format=<format>] <tagname>...

   -l, --list          list tag names
   -n[<n>]             print <n> lines of each tag message
   -d, --delete        delete tags
   -v, --verify        verify tags

Tag creation options
   -a, --[no-]annotate   annotated tag, needs a message
   -m, --message <message>
                         tag message
   -F, --[no-]file <file>
                         read message from file
   --trailer <trailer>   add custom trailer(s)
   -e, --[no-]edit       force edit of tag message
   -s, --[no-]sign       annotated and GPG-signed tag
   --[no-]cleanup <mode> how to strip spaces and #comments from message
   -u, --[no-]local-user <key-id>
                         use another key to sign the tag
   -f, --[no-]force      replace the tag if exists
   --[no-]create-reflog  create a reflog

Tag listing options
   --[no-]column[=<style>]
                         show tag list in columns
   --contains <commit>   print only tags that contain the commit
   --no-contains <commit>
                         print only tags that don't contain the commit
   --merged <commit>     print only tags that are merged
   --no-merged <commit>  print only tags that are not merged
   --[no-]omit-empty     do not output a newline after empty formatted refs
   --[no-]sort <key>     field name to sort on
   --[no-]points-at <object>
                         print only tags of the object
   --[no-]format <format>
                         format to use for the output
   --[no-]color[=<when>] respect format colors
```

-i, --[no-]ignore-case
sorting and filtering are case insensitive

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git log
commit ca79b930475cd4b8b759fd46c4ad488bb119636e (HEAD -> master)
Author: Kamal Agrahari <kamalagrahari2003@gmail.com>
Date: Fri Aug 2 09:48:17 2024 +0530

Added the content to myhistory.txt

commit dbf4c2385576bebe78540695fa1028a53bfe381c
Author: Kamal Agrahari <kamalagrahari2003@gmail.com>
Date: Fri Aug 2 09:45:24 2024 +0530

Updated the file myhistory.txt and school.txt

commit c94bef371475664fb78ce5c5c724b2d07b98f138 (origin/master)
Author: Kamal Agrahari <kamalagrahari2003@gmail.com>
Date: Fri Aug 2 09:42:12 2024 +0530

Adding the file myhistory.txt

commit aab0ddaa394ebd84a538d4521b997b8e67271ce8
Author: Kamal Agrahari <kamalagrahari2003@gmail.com>
Date: Fri Aug 2 09:40:07 2024 +0530

restore the delted file

commit d9e4f5f206d678a739891a49ec8015cae2bf18b8
Author: Kamal Agrahari <kamalagrahari2003@gmail.com>
Date: Fri Aug 2 09:36:30 2024 +0530

Deleted the file school.txt

commit c0f6c34a528c5108e79f17287071933ba0f9f3d5
Author: Kamal Agrahari <kamalagrahari2003@gmail.com>
Date: Fri Aug 2 09:28:48 2024 +0530

Created files school.txt and college.txt

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git log --oneline --decorate
ca79b93 (HEAD -> master) Added the content to myhistory.txt
dbf4c23 Updated the file myhistory.txt and school.txt
c94bef3 (origin/master) Adding the file myhistory.txt
aab0dda restore the delted file

```
d9e4f5f Deleted the file school.txt
c0f6c34 Created files school.txt and college.txt

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ ls -a
./  ../  .git/  college.txt  myhistory.txt  school.txt

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git branch
* master

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git branch newbie

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git checkout newbie
Switched to branch 'newbie'

Kamal Agrahari@Desktop MINGW64 ~/exp3 (newbie)
$ nano newbie.txt

Kamal Agrahari@Desktop MINGW64 ~/exp3 (newbie)
$ git add .
warning: in the working copy of 'newbie.txt', LF will be replaced by CRLF the next time Git touches it

Kamal Agrahari@Desktop MINGW64 ~/exp3 (newbie)
$ git commit -m "Created the newbie branch"
[newbie a80c5bf] Created the newbie branch
 1 file changed, 2 insertions(+)
 create mode 100644 newbie.txt

Kamal Agrahari@Desktop MINGW64 ~/exp3 (newbie)
$ git log --oneline
a80c5bf (HEAD -> newbie) Created the newbie branch
ca79b93 (master) Added the content to myhistory.txt
dbf4c23 Updated the file myhistory.txt and school.txt
c94bef3 (origin/master) Adding the file myhistory.txt
aab0dda restore the delted file
d9e4f5f Deleted the file school.txt
c0f6c34 Created files school.txt and college.txt

Kamal Agrahari@Desktop MINGW64 ~/exp3 (newbie)
$ git checkout master
Switched to branch 'master'

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git push
fatal: The current branch master has no upstream branch.
```

To push the current branch and set the remote as upstream, use

    git push --set-upstream origin master

To have this happen automatically for branches without a tracking
upstream, see 'push.autoSetupRemote' in 'git help config'.

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git push origin master
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 16 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (7/7), 820 bytes | 820.00 KiB/s, done.
Total 7 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Kamalagrahari03/devops-exp3.git
   c94bef3..ca79b93 master -> master

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git log --oneline
ca79b93 (HEAD -> master, origin/master) Added the content to myhistory.txt
dbf4c23 Updated the file myhistory.txt and school.txt
c94bef3 Adding the file myhistory.txt
aab0dda restore the delted file
d9e4f5f Deleted the file school.txt
c0f6c34 Created files school.txt and college.txt

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ ls -a
./ ../ .git/ college.txt myhistory.txt school.txt

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git checkout newbie
Switched to branch 'newbie'

Kamal Agrahari@Desktop MINGW64 ~/exp3 (newbie)
$ git log --oneline
a80c5bf (HEAD -> newbie) Created the newbie branch
ca79b93 (origin/master, master) Added the content to myhistory.txt
dbf4c23 Updated the file myhistory.txt and school.txt
c94bef3 Adding the file myhistory.txt
aab0dda restore the delted file
d9e4f5f Deleted the file school.txt
c0f6c34 Created files school.txt and college.txt

Kamal Agrahari@Desktop MINGW64 ~/exp3 (newbie)
$ git branch saap

```
Kamal Agrahari@Desktop MINGW64 ~/exp3 (newbie)
$ git checkout saap
Switched to branch 'saap'

Kamal Agrahari@Desktop MINGW64 ~/exp3 (saap)
$ nano saap.txt

Kamal Agrahari@Desktop MINGW64 ~/exp3 (saap)
$ git status
On branch saap
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        saap.txt

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

Kamal Agrahari@Desktop MINGW64 ~/exp3 (saap)
$ git add .
warning: in the working copy of 'saap.txt', LF will be replaced by CRLF the next time Git touches it

Kamal Agrahari@Desktop MINGW64 ~/exp3 (saap)
$ git commit -m "Created new file in saap branch"
[saap b7c6609] Created new file in saap branch
 1 file changed, 1 insertion(+)
 create mode 100644 saap.txt

Kamal Agrahari@Desktop MINGW64 ~/exp3 (saap)
$ git checkout master
Switched to branch 'master'

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git push origin master
Everything up-to-date

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git log --oneline
ca79b93 (HEAD -> master, origin/master) Added the content to myhistory.txt
dbf4c23 Updated the file myhistory.txt and school.txt
c94bef3 Adding the file myhistory.txt
aab0dda restore the delted file
d9e4f5f Deleted the file school.txt
c0f6c34 Created files school.txt and college.txt

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git branch
* master
  newbie
```

saap

```
Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git checkout saap
Switched to branch 'saap'

Kamal Agrahari@Desktop MINGW64 ~/exp3 (saap)
$ git log --oneline
b7c6609 (HEAD -> saap) Created new file in saap branch
a80c5bf (newbie) Created the newbie branch
ca79b93 (origin/master, master) Added the content to myhistory.txt
dbf4c23 Updated the file myhistory.txt and school.txt
c94bef3 Adding the file myhistory.txt
aab0dda restore the delted file
d9e4f5f Deleted the file school.txt
c0f6c34 Created files school.txt and college.txt

Kamal Agrahari@Desktop MINGW64 ~/exp3 (saap)
$ git checkout master
Switched to branch 'master'

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git merge saap
Updating ca79b93..b7c6609
Fast-forward
 newbie.txt | 2 ++
 saap.txt   | 1 +
 2 files changed, 3 insertions(+)
 create mode 100644 newbie.txt
 create mode 100644 saap.txt

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git log --oneline
b7c6609 (HEAD -> master, saap) Created new file in saap branch
a80c5bf (newbie) Created the newbie branch
ca79b93 (origin/master) Added the content to myhistory.txt
dbf4c23 Updated the file myhistory.txt and school.txt
c94bef3 Adding the file myhistory.txt
aab0dda restore the delted file
d9e4f5f Deleted the file school.txt
c0f6c34 Created files school.txt and college.txt

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git branch -d saap
Deleted branch saap (was b7c6609).

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git branch
```

```
* master
  newbie

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git checkout newbie
Switched to branch 'newbie'

Kamal Agrahari@Desktop MINGW64 ~/exp3 (newbie)
$ ls
college.txt  myhistory.txt  newbie.txt  school.txt

Kamal Agrahari@Desktop MINGW64 ~/exp3 (newbie)
$ git merge master
Updating a80c5bf..b7c6609
Fast-forward
 saap.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 saap.txt

Kamal Agrahari@Desktop MINGW64 ~/exp3 (newbie)
$ git log --oneline
b7c6609 (HEAD -> newbie, master) Created new file in saap branch
a80c5bf Created the newbie branch
ca79b93 (origin/master) Added the content to myhistory.txt
dbf4c23 Updated the file myhistory.txt and school.txt
c94bef3 Adding the file myhistory.txt
aab0dda restore the delted file
d9e4f5f Deleted the file school.txt
c0f6c34 Created files school.txt and college.txt

Kamal Agrahari@Desktop MINGW64 ~/exp3 (newbie)
$ git checkout master
Switched to branch 'master'

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git log --oneline
b7c6609 (HEAD -> master, newbie) Created new file in saap branch
a80c5bf Created the newbie branch
ca79b93 (origin/master) Added the content to myhistory.txt
dbf4c23 Updated the file myhistory.txt and school.txt
c94bef3 Adding the file myhistory.txt
aab0dda restore the delted file
d9e4f5f Deleted the file school.txt
c0f6c34 Created files school.txt and college.txt

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git branch beta
```

```
Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git branch
  beta
* master
  newbie

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ nano beta.txt

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git add .
warning: in the working copy of 'beta.txt', LF will be replaced by CRLF the next time Git touches it

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git commit -m "Created beta branch and added new file"
[master 784c4e0] Created beta branch and added new file
 1 file changed, 2 insertions(+)
 create mode 100644 beta.txt

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git log --oneline
784c4e0 (HEAD -> master) Created beta branch and added new file
b7c6609 (newbie, beta) Created new file in saap branch
a80c5bf Created the newbie branch
ca79b93 (origin/master) Added the content to myhistory.txt
dbf4c23 Updated the file myhistory.txt and school.txt
c94bef3 Adding the file myhistory.txt
aab0dda restore the delted file
d9e4f5f Deleted the file school.txt
c0f6c34 Created files school.txt and college.txt

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ ls
beta.txt  college.txt  myhistory.txt  newbie.txt  saap.txt  school.txt

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git status
On branch master
nothing to commit, working tree clean

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
$ git checkout newbie
Switched to branch 'newbie'

Kamal Agrahari@Desktop MINGW64 ~/exp3 (newbie)
$ git checkout beta
Switched to branch 'beta'
```

```
Kamal Agrahari@Desktop MINGW64 ~/exp3 (beta)
$ nano gaana.txt

Kamal Agrahari@Desktop MINGW64 ~/exp3 (beta)
$ git log --oneline
b7c6609 (HEAD -> beta, newbie) Created new file in saap branch
a80c5bf Created the newbie branch
ca79b93 (origin/master) Added the content to myhistory.txt
dbf4c23 Updated the file myhistory.txt and school.txt
c94bef3 Adding the file myhistory.txt
aab0dda restore the delted file
d9e4f5f Deleted the file school.txt
c0f6c34 Created files school.txt and college.txt

Kamal Agrahari@Desktop MINGW64 ~/exp3 (beta)
$ git stash list

Kamal Agrahari@Desktop MINGW64 ~/exp3 (beta)
$ git add .
warning: in the working copy of 'gaana.txt', LF will be replaced by CRLF the next time Git touches it

Kamal Agrahari@Desktop MINGW64 ~/exp3 (beta)
$ git stash list

Kamal Agrahari@Desktop MINGW64 ~/exp3 (beta)
$ git commit -m "created gaana file in beta branch"
[beta 807e5eb] created gaana file in beta branch
 1 file changed, 7 insertions(+)
 create mode 100644 gaana.txt

Kamal Agrahari@Desktop MINGW64 ~/exp3 (beta)
$ git stash list

Kamal Agrahari@Desktop MINGW64 ~/exp3 (beta)
$ git diff stash
fatal: ambiguous argument 'stash': unknown revision or path not in the working tree.
Use '--' to separate paths from revisions, like this:
'git <command> [<revision>...] -- [<file>...]'

Kamal Agrahari@Desktop MINGW64 ~/exp3 (beta)
$ ls
college.txt  gaana.txt  myhistory.txt  newbie.txt  saap.txt  school.txt

Kamal Agrahari@Desktop MINGW64 ~/exp3 (beta)
$ git checkout master
Switched to branch 'master'

Kamal Agrahari@Desktop MINGW64 ~/exp3 (master)
```

```
$ git push origin master
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 16 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (9/9), 901 bytes | 901.00 KiB/s, done.
Total 9 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/Kamalagrahari03/devops-exp3.git
   ca79b93..784c4e0  master -> master
```

Name – Kamal D. Agrahari
ID -VU4F2223028

Subject – DevOps Lab
TE IT A | Batch B

Experiment No. 4

Aim - : To Install Jenkins on Windows.

Theory –

Jenkins:

Jenkins is a Java based open-source application, which is one of the most popular tools for continuous integration and continuous delivery on any platform.

Jenkins is a self-contained server that can be used to automate all sorts of tasks related to building, testing, and delivering or deploying software. It has many plugins for automating almost everything at the infrastructure level. The use of Jenkins has widely increased due to a rich set of functionalities, which it provides in the form of plugins.

The Jenkins project has two release lines: Stable (LTS) and regular (Weekly). Depending upon the scenario and need, one can be chosen:

- Stable (LTS): Long-Term Support (LTS) release baselines are chosen every 12 weeks from the stream of regular releases.
- Regular releases (Weekly): This delivers bug fixes and new features rapidly (generally weekly) to users and plugin developers who need them.

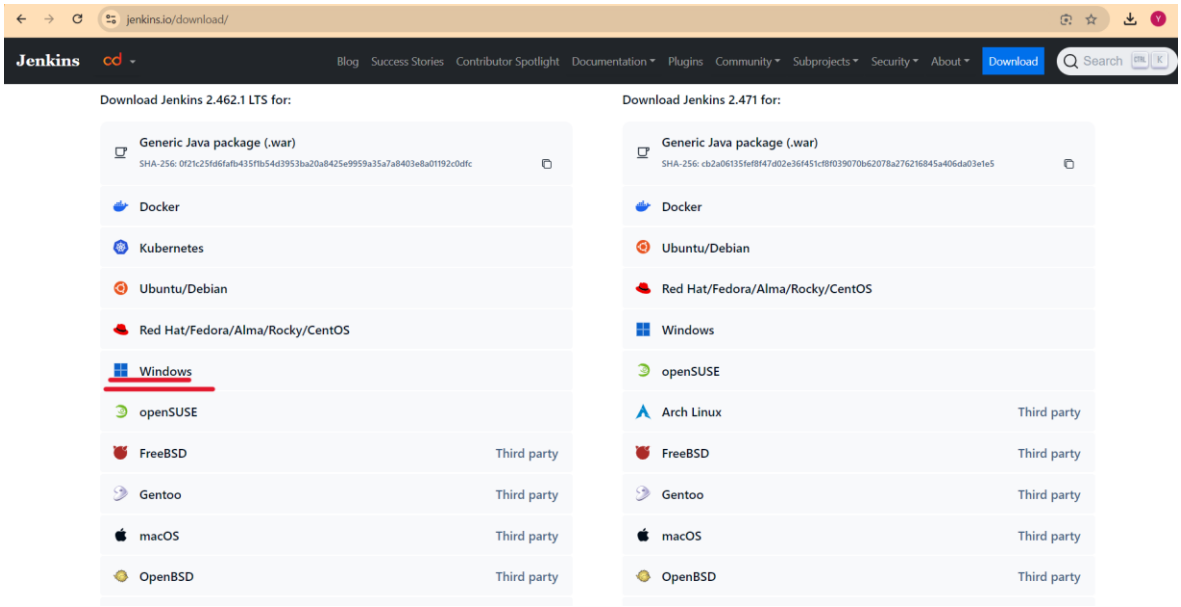
Jenkins can be installed through native system packages, Docker, or even run standalone by any machine with a Java Runtime Environment (JRE) installed.

Steps for Installing of Jenkins:

Step 1: Install JDK 17 or JDK 21 as they are supported Java versions for jenkins. After successful installation of JDK you will get the following respond in command prompt.

```
Microsoft Windows [Version 10.0.19045.4651]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ADMIN>java --version
java 17.0.12 2024-07-16 LTS
Java(TM) SE Runtime Environment (build 17.0.12+8-LTS-286)
Java HotSpot(TM) 64-Bit Server VM (build 17.0.12+8-LTS-286, mixed mode, sharing)
```

Step 2: Navigate to the “jenkins.io/download/” website and under Jenkins LTS system, click on Windows for the latest Jenkins package. This will download “jenkins.msi” installer.



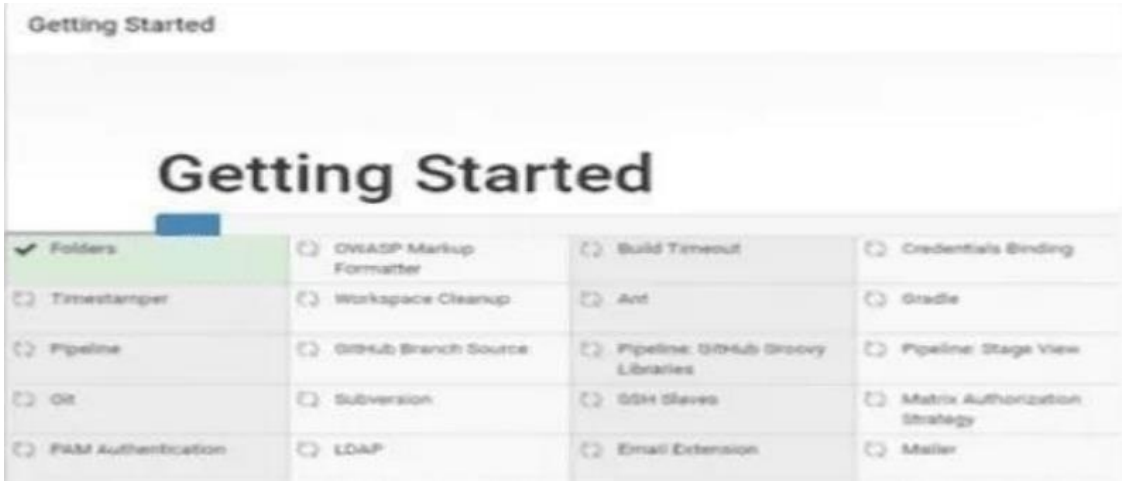
Step 3: Once the jenkins.msi is downloaded open it and proceed. Then choose the Jenkins Path. Give Logon type as “Run Service as Local System” and test the recommended port 8080 in next step and proceed the installation.

Step 4: You will be redirected to a local Jenkins page by default. If page is not loaded by default, paste the URL http://localhost:8080 , with the chosen port in the browser.

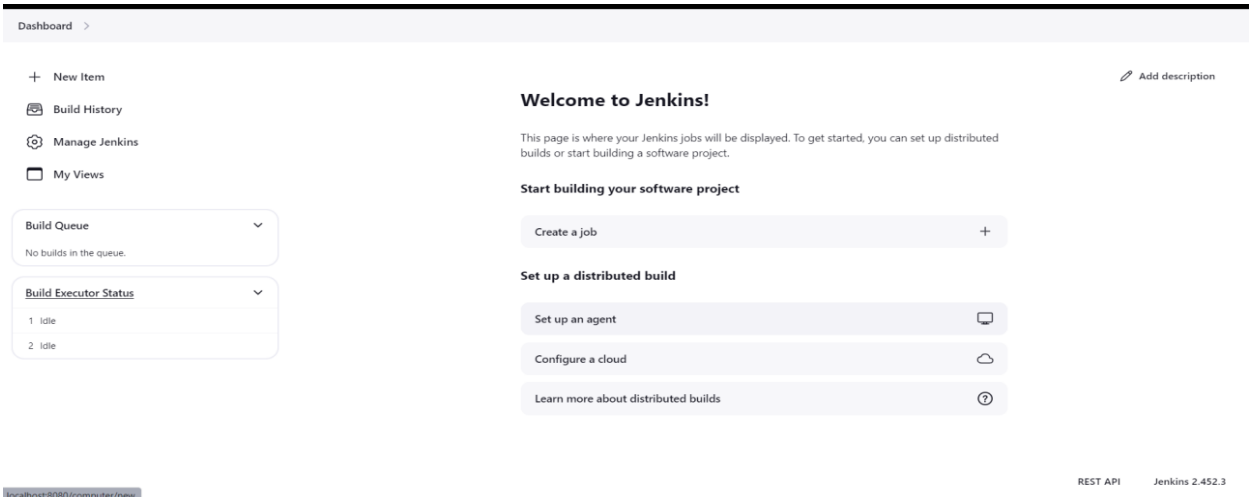
After completing the Jenkins installation phase, we will proceed with its configuration set up.

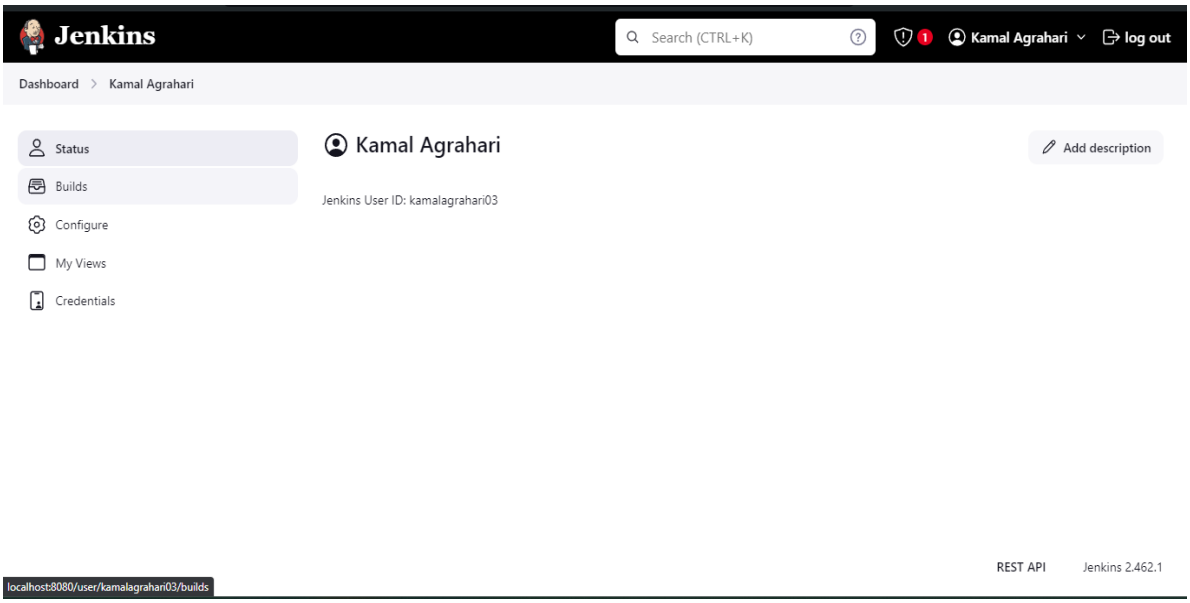
Step 5 : For Unlocking Jenkins, copy the password from the file at C:\Windows\System32\config\systemprofile\AppData\Local\Jenkins\.jenkins\secrets (shown on screen) and paste it in the Administrator password field. Click on Continue.

Step 6 : Choose "Install suggested plugins" and let the plugins install completely.



Step 7 : Finally the default Jenkins page will be shown as below.





Conclusion:

Hence we have perform Installation of Jenkins on Windows.

Name – Kamal D. Agrahari
ID -VU4F2223028

Subject – DevOps Lab
TE IT A | Batch B

Experiment No. 5

Aim - : Executing java sequence programming and integration of GitHub with Jenkins.

Theory –

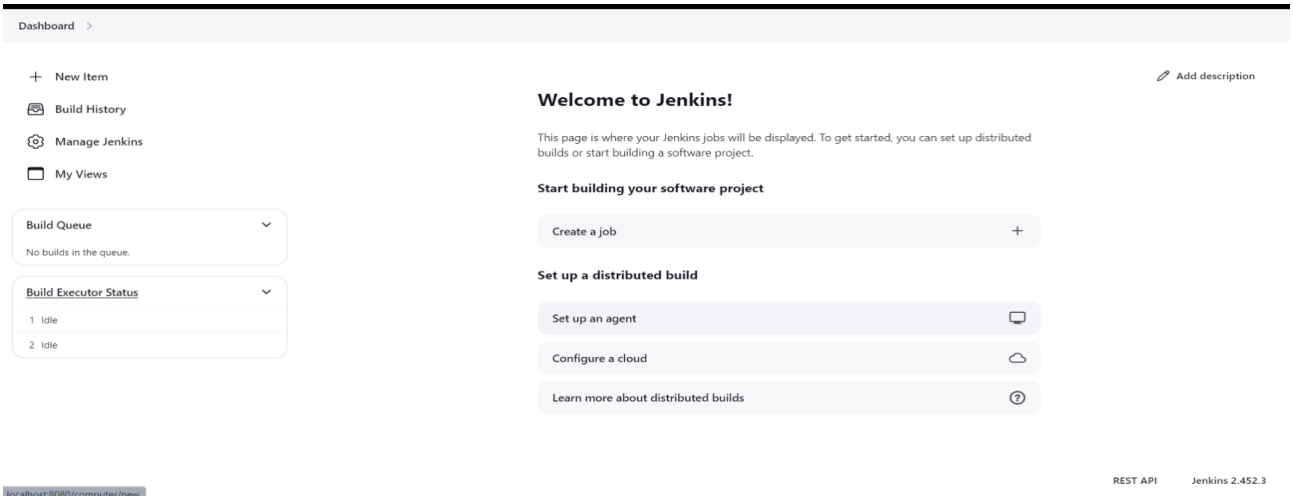
Jenkins:

Jenkins comes with a pretty basic setup, so you will need to install the required plugins to enable respective third-party application support.

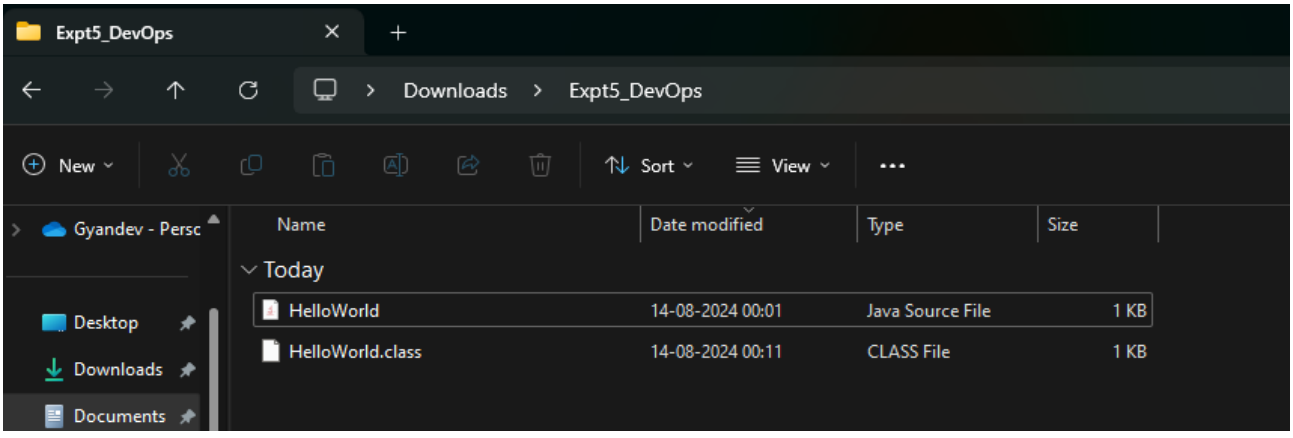
GitHub is a web-based repository of code which plays a major role in DevOps. It provides a common platform for multiple developers working on the same code/project to upload and retrieve updated code, thereby facilitating continuous integration.

Jenkins needs to have GitHub plugin installed to be able to pull code from the GitHub repository.

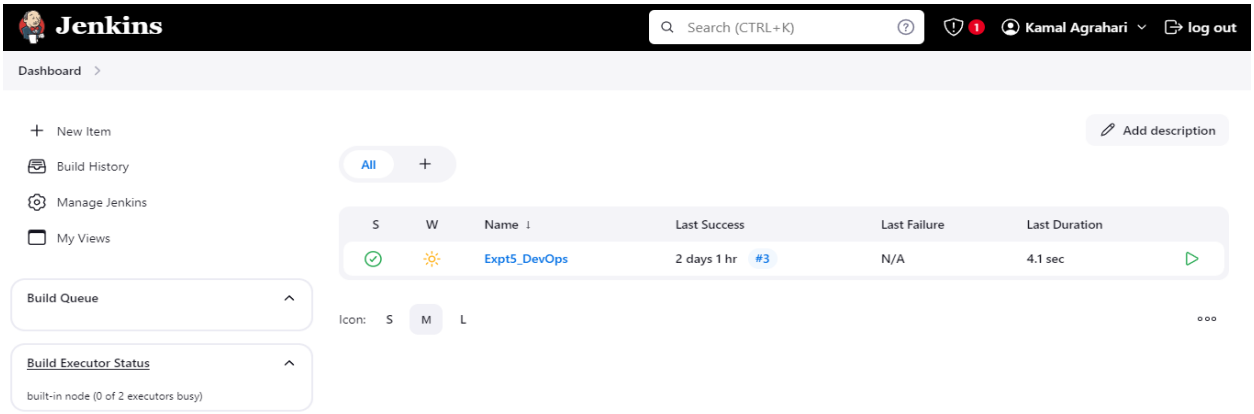
Step 1: Login to Jenkins.



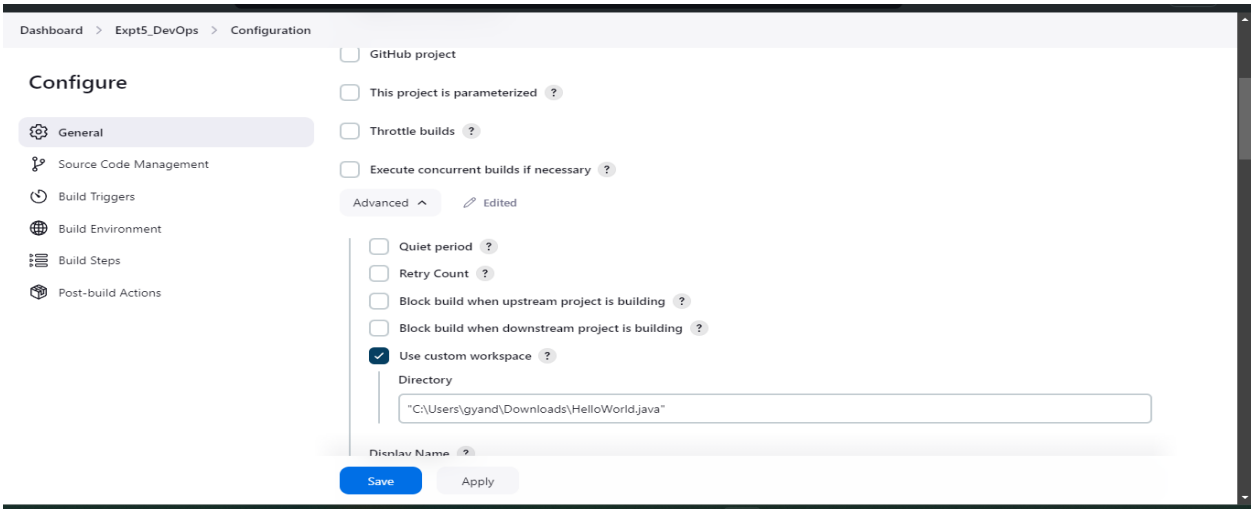
Step 2: Go to any Folder and Create one java File.



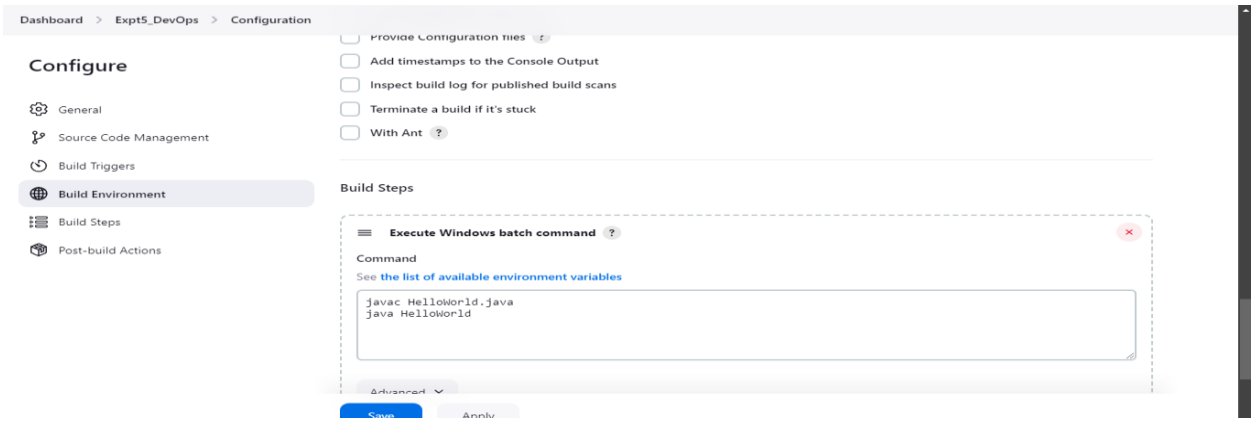
Step 3: Now create new Item.

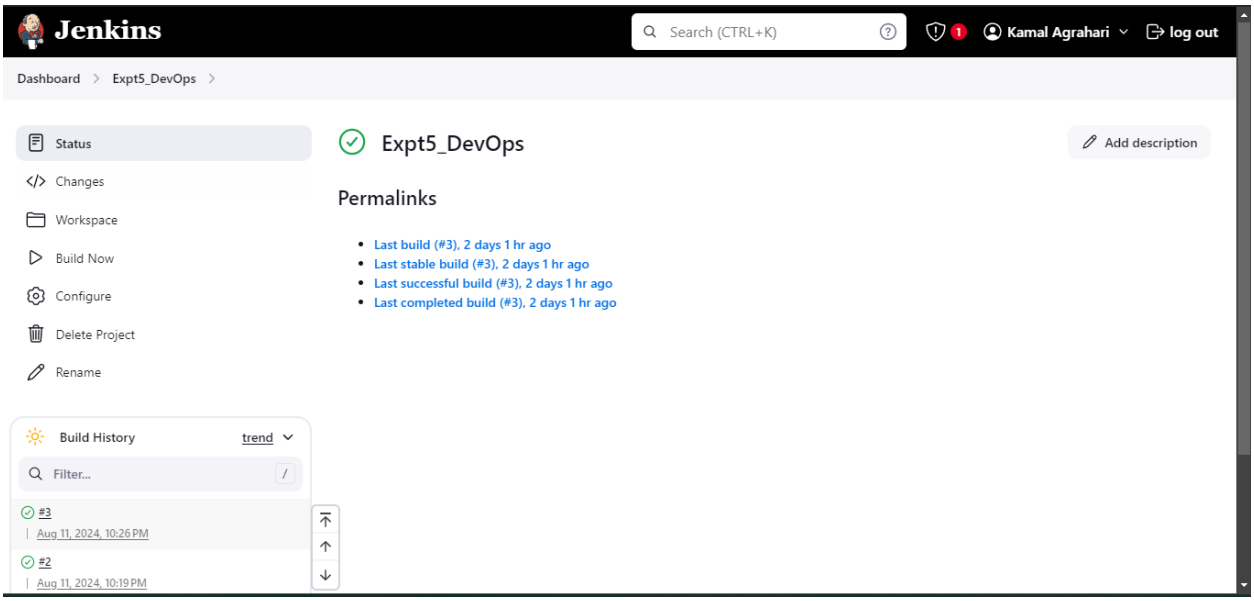


Step 4: In General section. Click on “Advanced” button. Enter the location of your Java code stored in local directory in custom workspace.

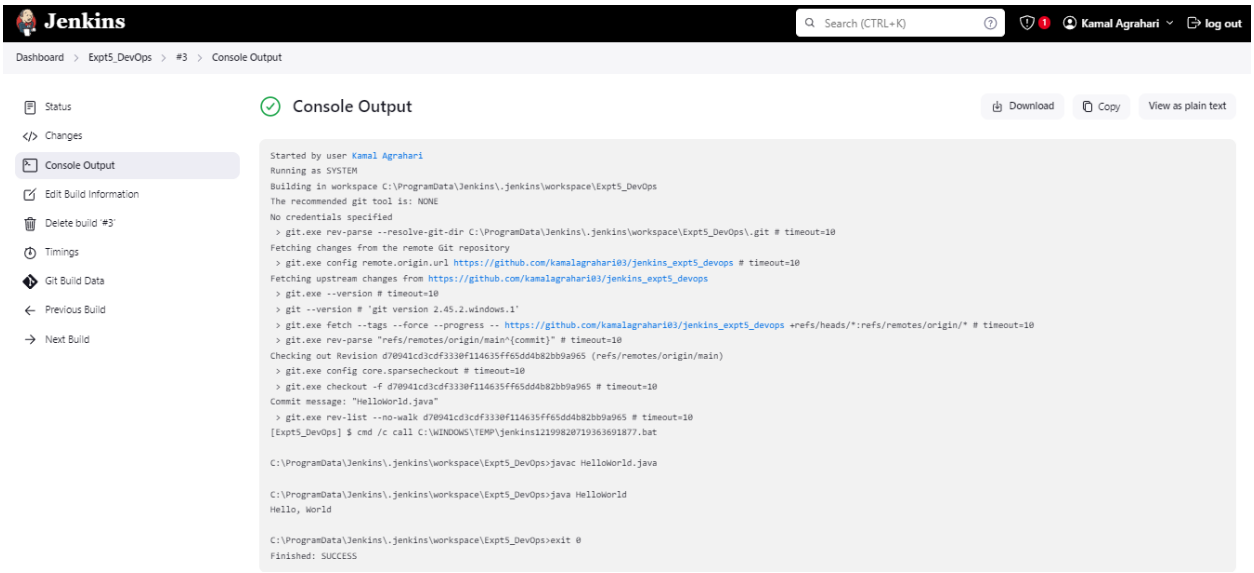


Step 5: Go to “Build” section and add build step “Execute Windows batch command”





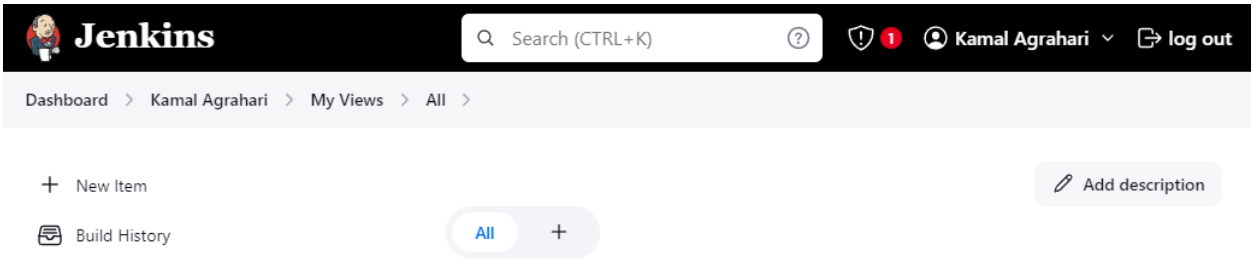
Step 6 : Execute “Hello java” by clicking on “Build Now” link. If the job is successful, then it will show Blue ball or Green tick otherwise Red ball under “Build History”



How to Integrate Jenkins With GitHub:

Firstly you need to install required Git plugins. Let’s proceed to the next steps for integrating Jenkins with GitHub.

Step 1: Create a new job in Jenkins, open the Jenkins dashboard with your Jenkins URL. For example, http://localhost:8080/ Click on create new item:

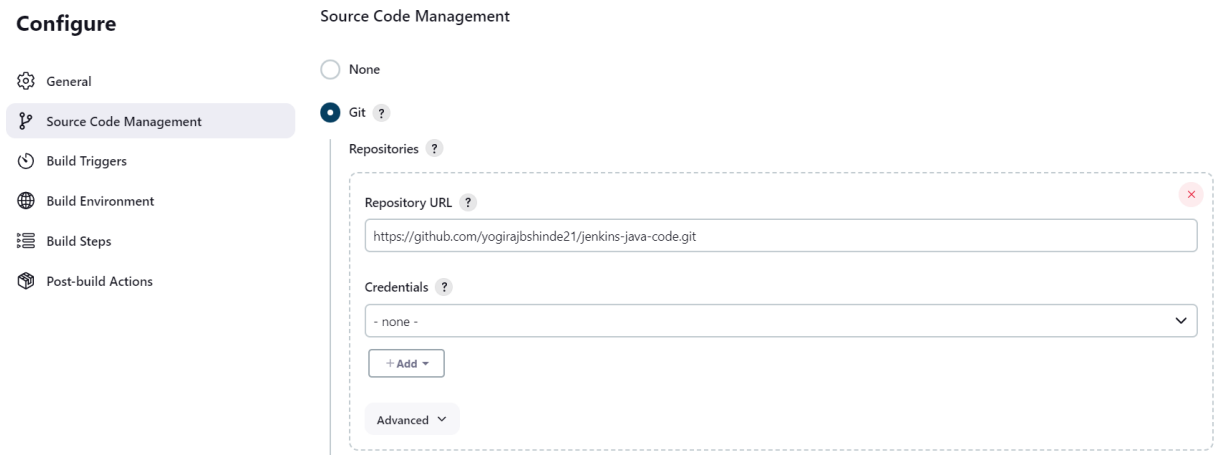


Step 2: Enter the item name select job type and click **OK**. We shall create a Freestyle project as an example.

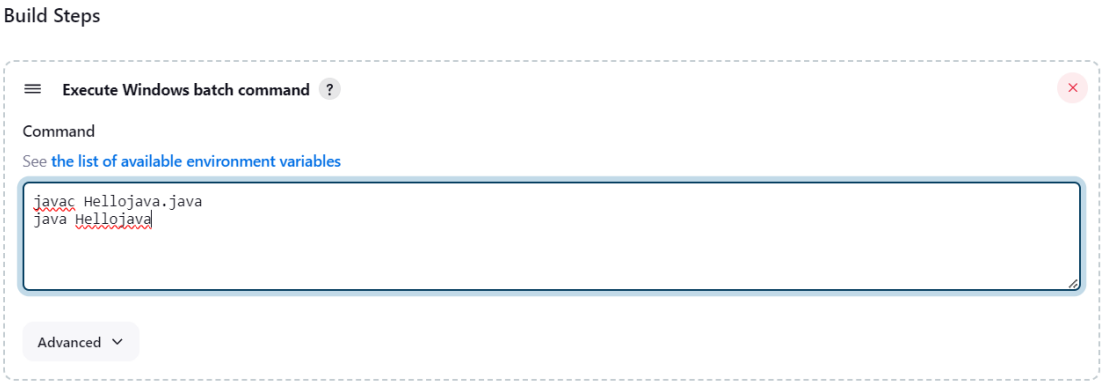
Step 3: Once you click **OK** then the page will be redirected to its project form. Here you will need to enter the project information.

Step 4: You will see a **Git** option under **Source Code Management** if your Git plugin has been installed in Jenkins.

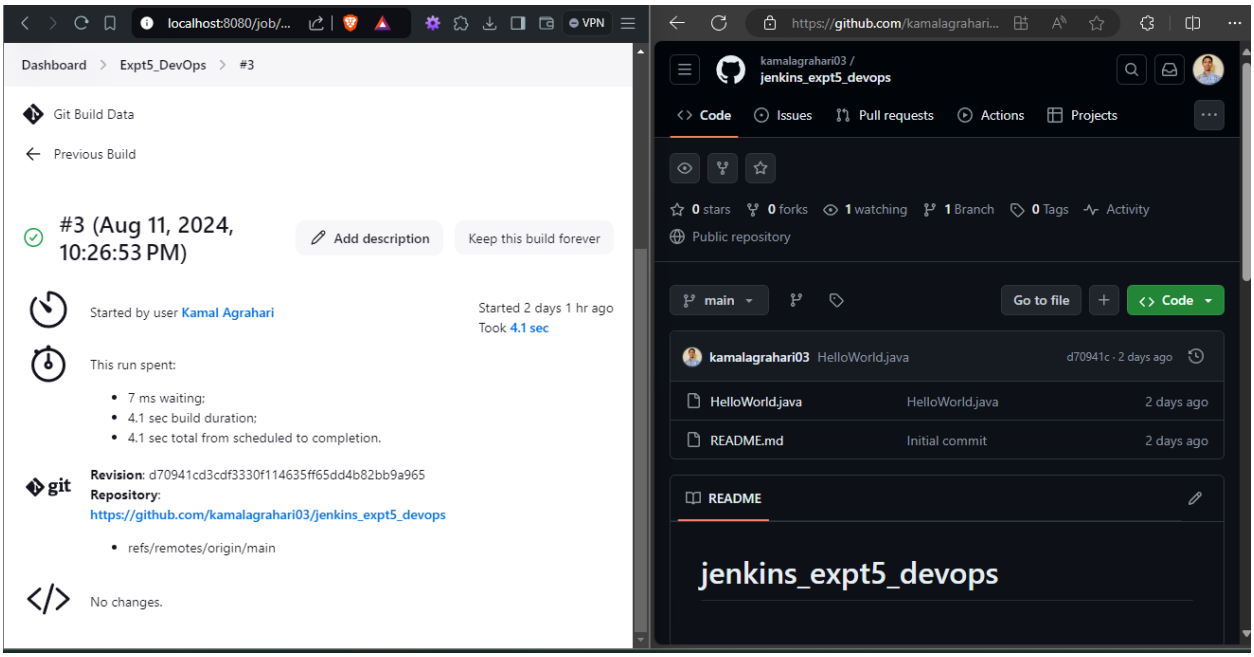
Step 5: Enter the Git repository URL to pull the code from GitHub:



Step 6 : Go to “Build” section and add build step “Execute Windows batch command”.



Step 7 : Click on “Build Now”. . If the job is successful, then it will show Blue ball or Green tick otherwise Red ball under “Build History”



Conclusion:

Hence, successfully executed java sequence program IN Jenkins integrated from GitHub.

Name – Kamal Dilip Kumar Agrahari
ID -VU4F2223028

Subject – DevOps Lab
TE IT A | Batch B

Experiment No. 6

Aim - : Executing java pipeline programming and integration of GitHub with Jenkins.

Theory –

Java pipeline programming refers to automating the process of building, testing, and deploying Java applications using Jenkins' Pipeline feature. Jenkins Pipelines are defined in a Jenkinsfile, which is a script that specifies the stages and steps of the continuous integration (CI) and continuous delivery (CD) process.

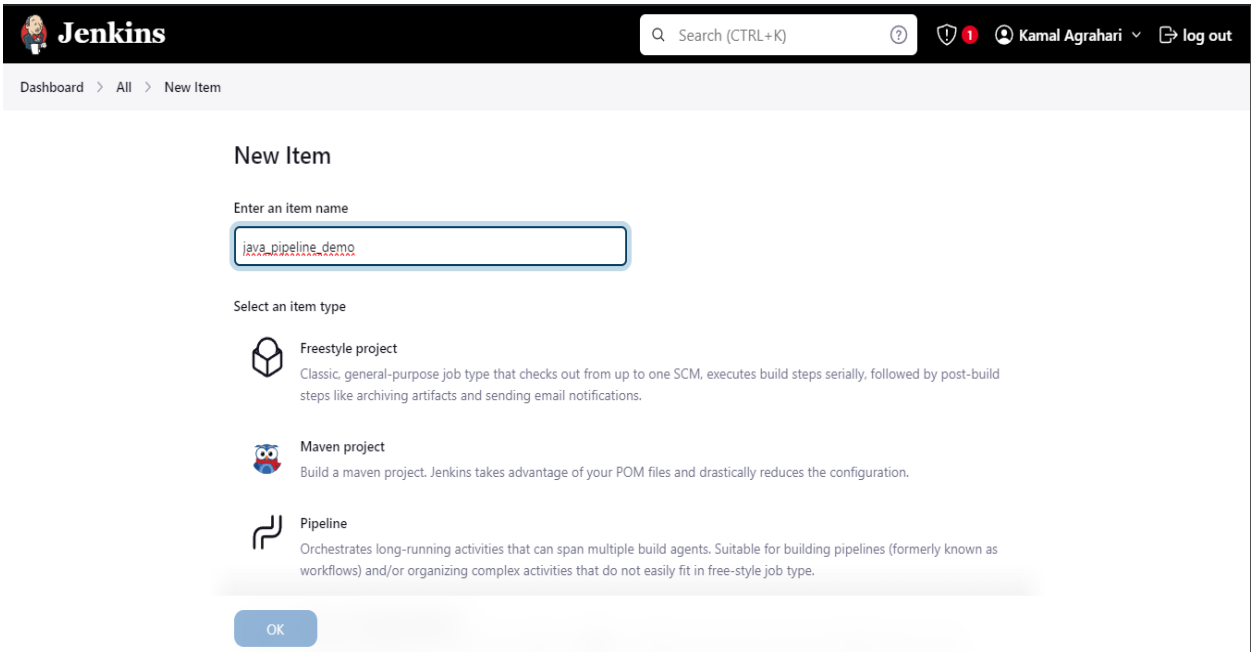
The JenkinsFile (Script file) is text file that defines the pipeline code, using Groovy-base syntax. It contains the stages and steps to execute for a Java Project

In our pipline, there are three stages

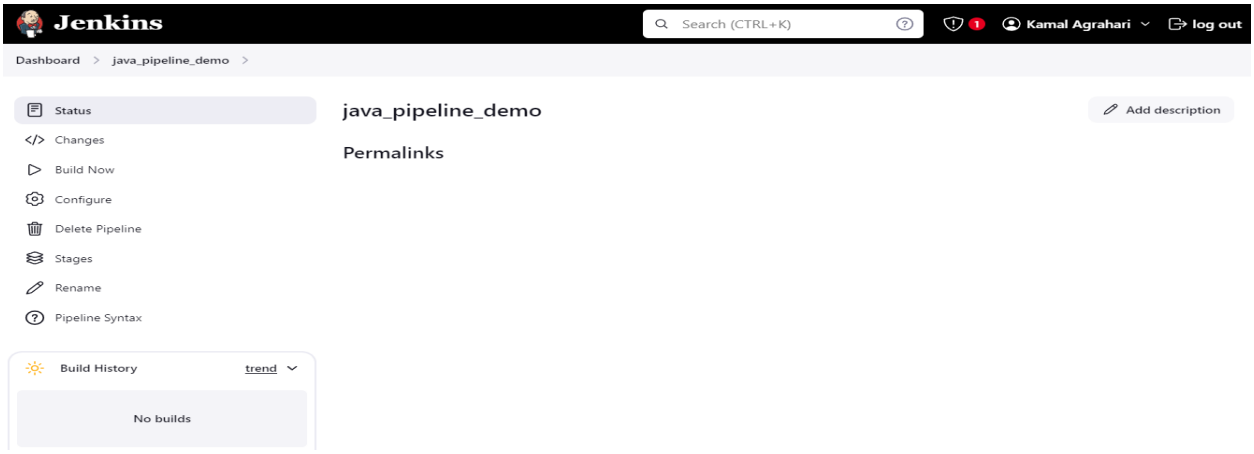
1. Checkout: In this stage the java program is fetched from our github repository
2. Build: In this stage the java porgram is compiled.
3. Test & Run: In this stage the compiled java program is executed and output is logged

Steps

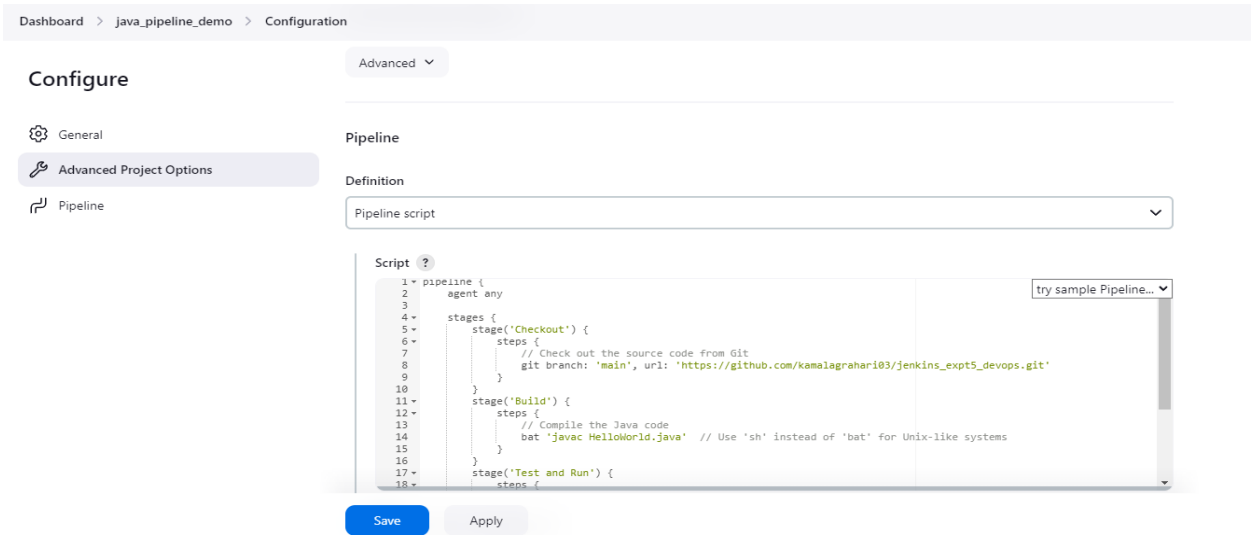
1. Open Jenkins and create a new item/job. Name it anything you want and then select the pipeline option



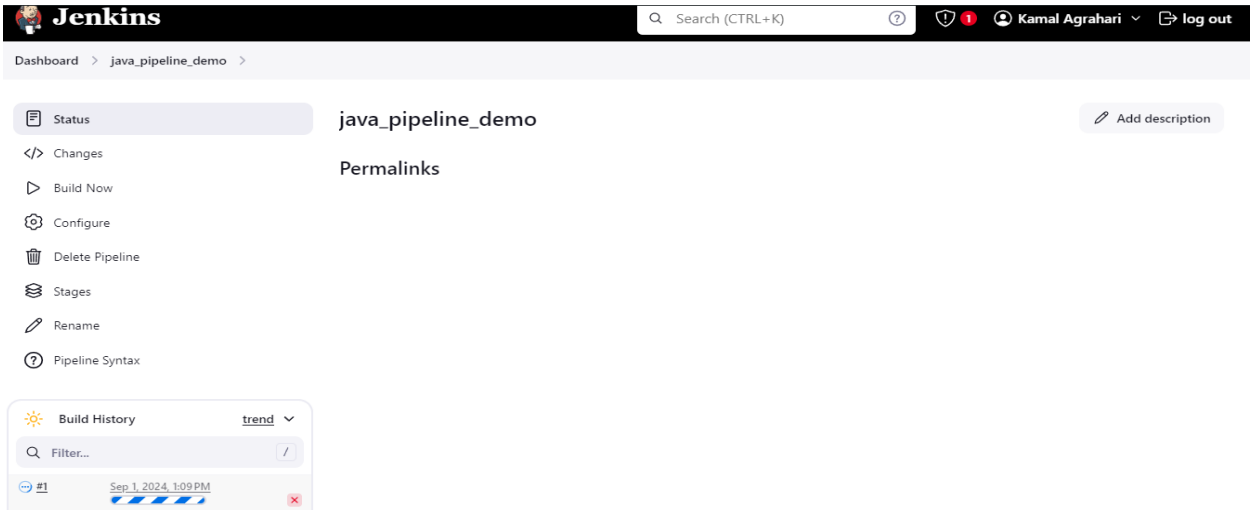
2. Open your pipeline item



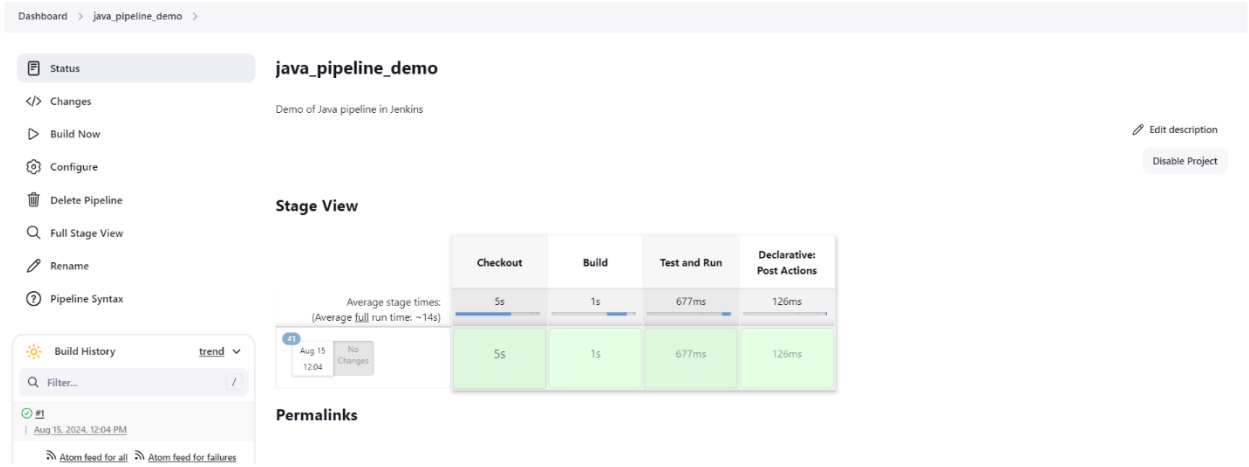
3. Go to Configure tab > then inside of it into Advance Project Options > In pipeline, in definition select pipeline script then in script area put your pipeline script.



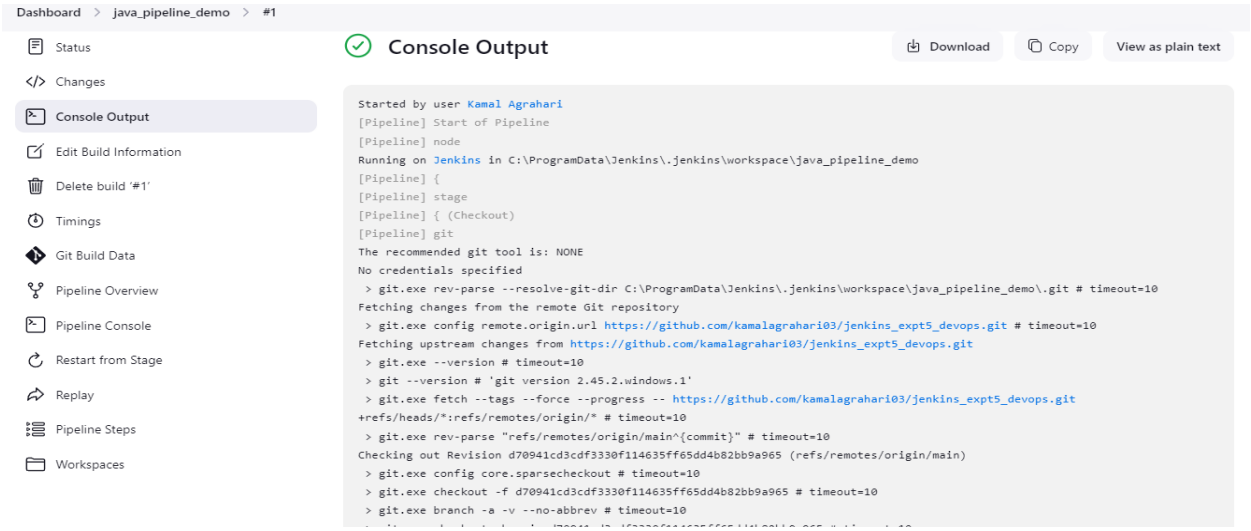
4. Then save your changes and click on Build Now to execute your pipeline



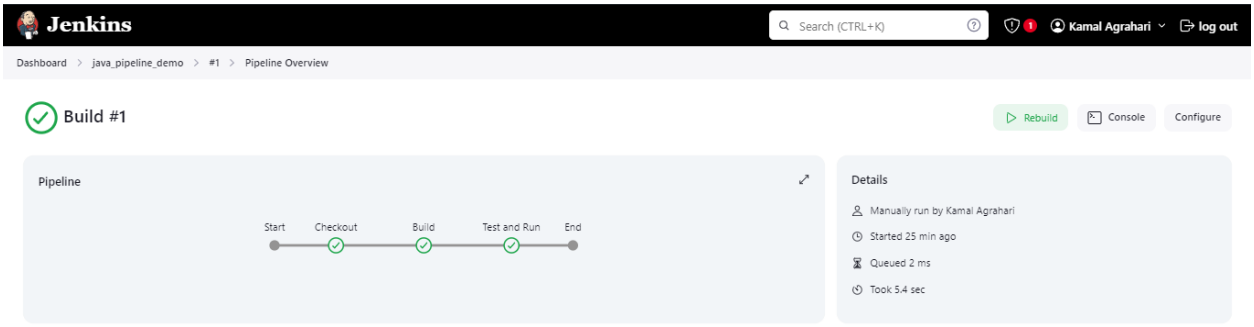
5. After you build is finished the overview results will be visible in Stage Vi

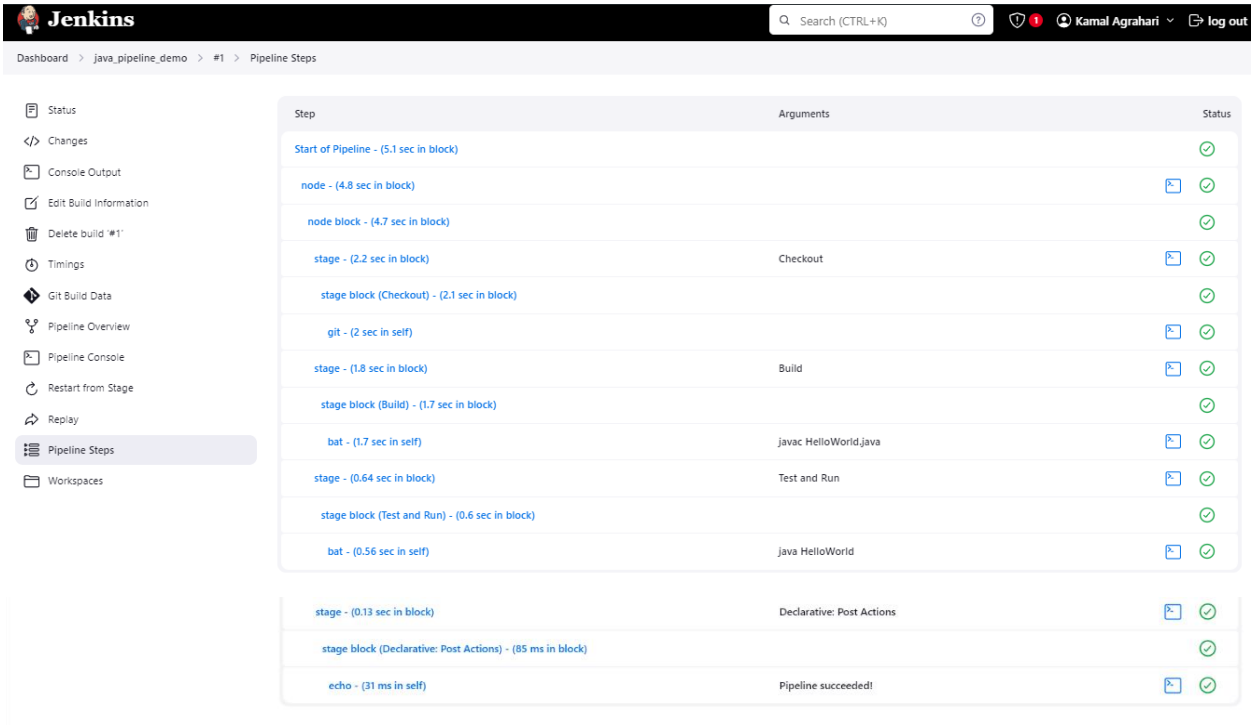


6. You can click on the build to see your console output



7. You can also checkout you step by step pipeline output in Pipeline Steps Tab





Conclusion:

Hence, successfully executed java pipeline programming in Jenkins integrated from GitHub.

Name: Kamal Agrahari
ID: VU4F2223028

Lab: DevOps Lab
TE IT A | Batch B

Experiment No. 7

Aim - : Execute a simple Maven project and integrate it with Jenkins using GitHub.

Theory –

Maven is a build automation tool primarily used for Java projects, hosted by the Apache Software Foundation. It simplifies the build process and provides a uniform system for managing project dependencies and configurations through the Project Object Model (POM), defined in the `pom.xml` file.

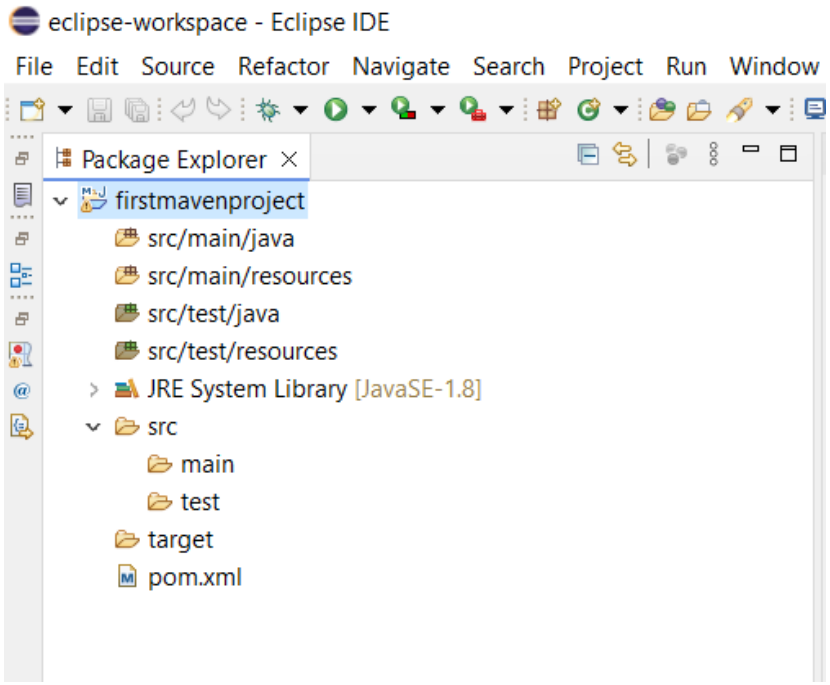
Tools and Technologies Used

- **Eclipse Neon**
- **Maven** - 3.5.3
- **JDK** - 1.8

Steps

1. Set Up Maven Project in Eclipse:

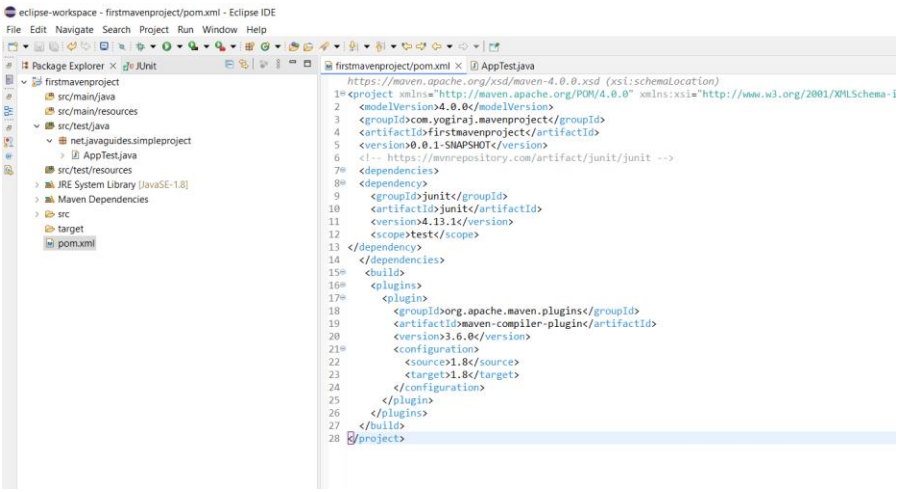
- Install Eclipse IDE from Eclipse Downloads.
- Open Eclipse and create a new Maven project (`File -> New -> Maven Project`).
- Select options for a simple project and use the default workspace location.
- Enter `GroupId` and `ArtifactId`.



2. Configure `pom.xml`:

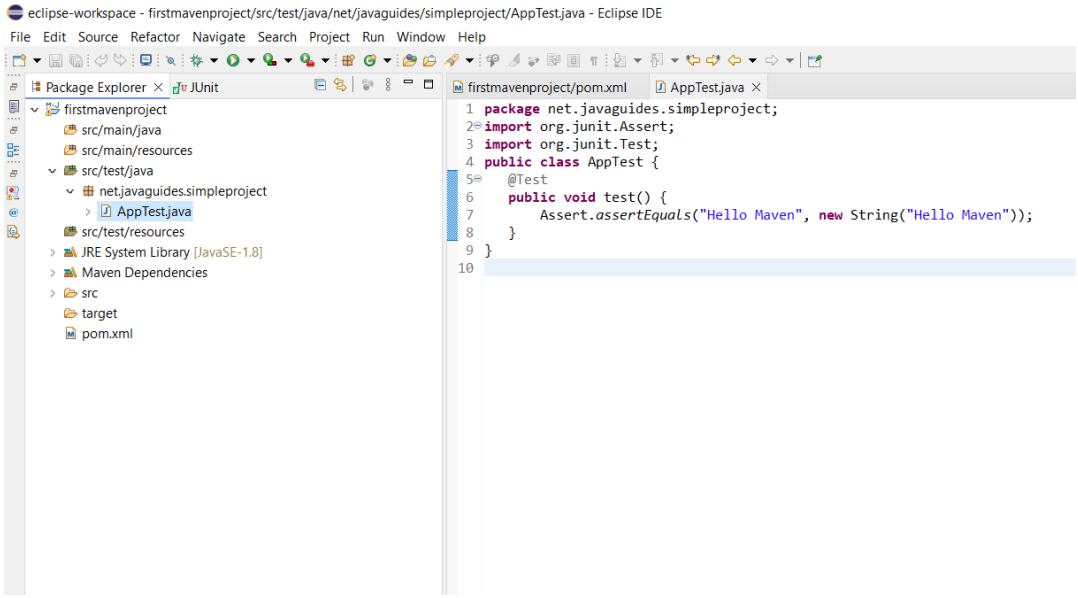
- Add the necessary dependencies (e.g., JUnit) and configure the Maven compiler plugin:

```
<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
https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.yogiraj.mavenproject</groupId>
  <artifactId>firstmavenproject</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <!-- https://mvnrepository.com/artifact/junit/junit -->
  <dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.13.1</version>
    <scope>test</scope>
</dependency>
  </dependencies>
   <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.6.0</version>
        <configuration>
          <source>1.8</source>
          <target>1.8</target>
        </configuration>
      </plugin>
    </plugins>
 </build>
</project>
```



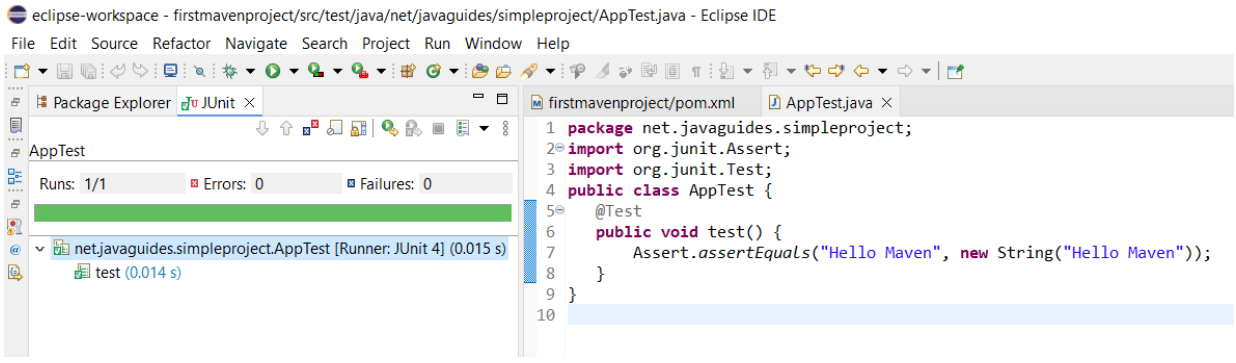
3. Create a JUnit Test:

- Create a package `net.javaguides.simpleproject` under `src/test/java`.
- Add `AppTest.java` and write a simple test:



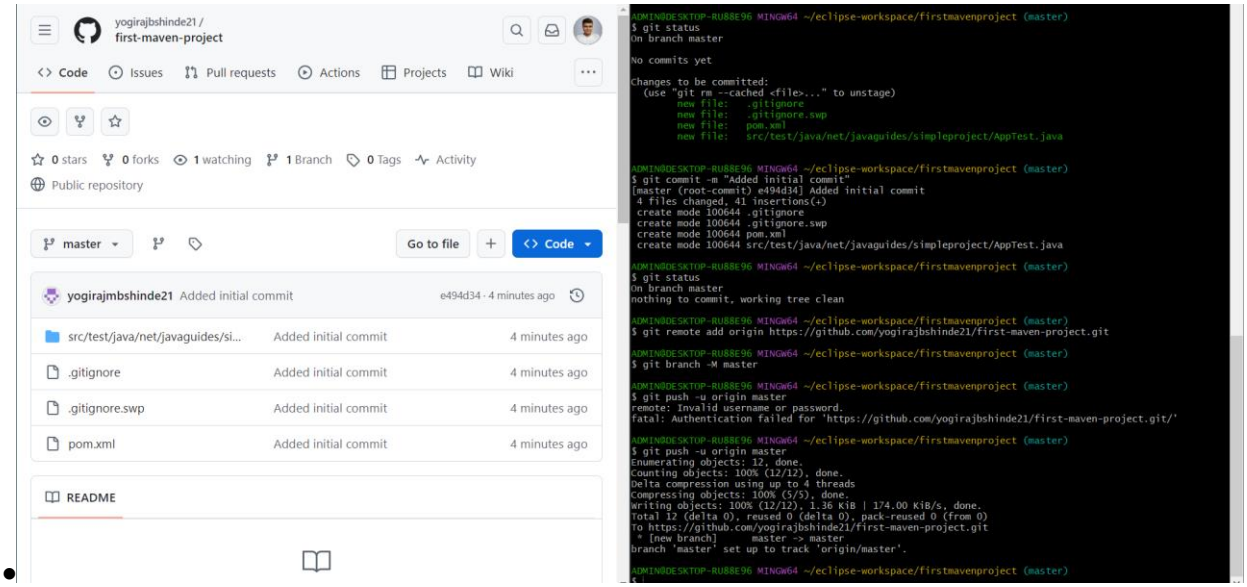
4. Run the Maven Project:

- Right-click on `AppTest.java` and select `Run as -> JUnit Test`.



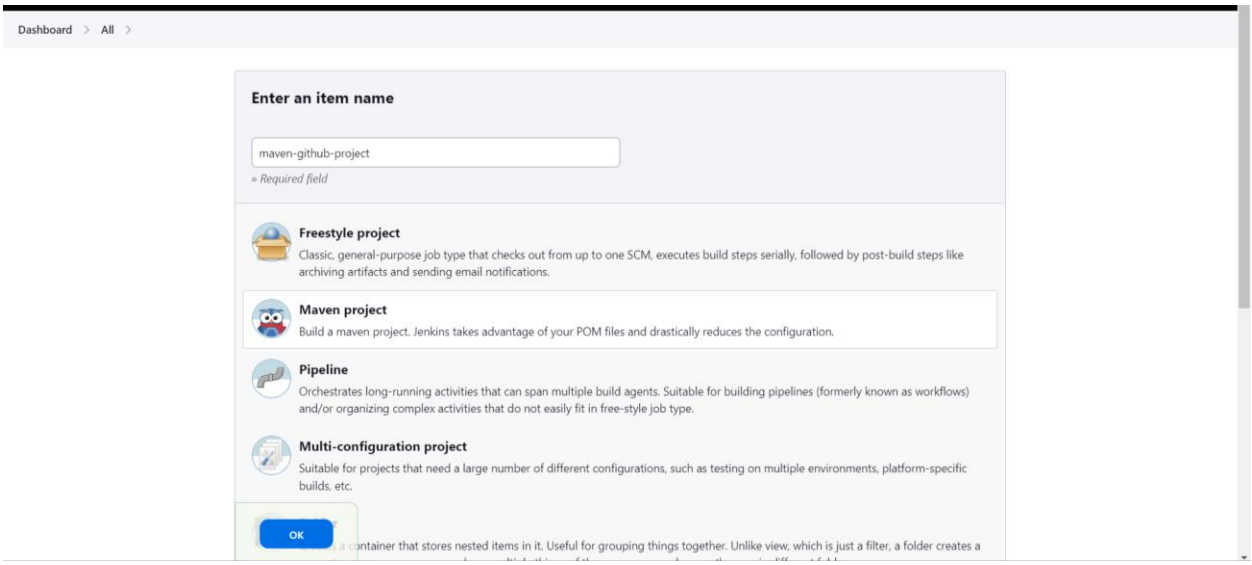
5. Push the Maven Project to GitHub:

- Commit and push the project to your GitHub repository.

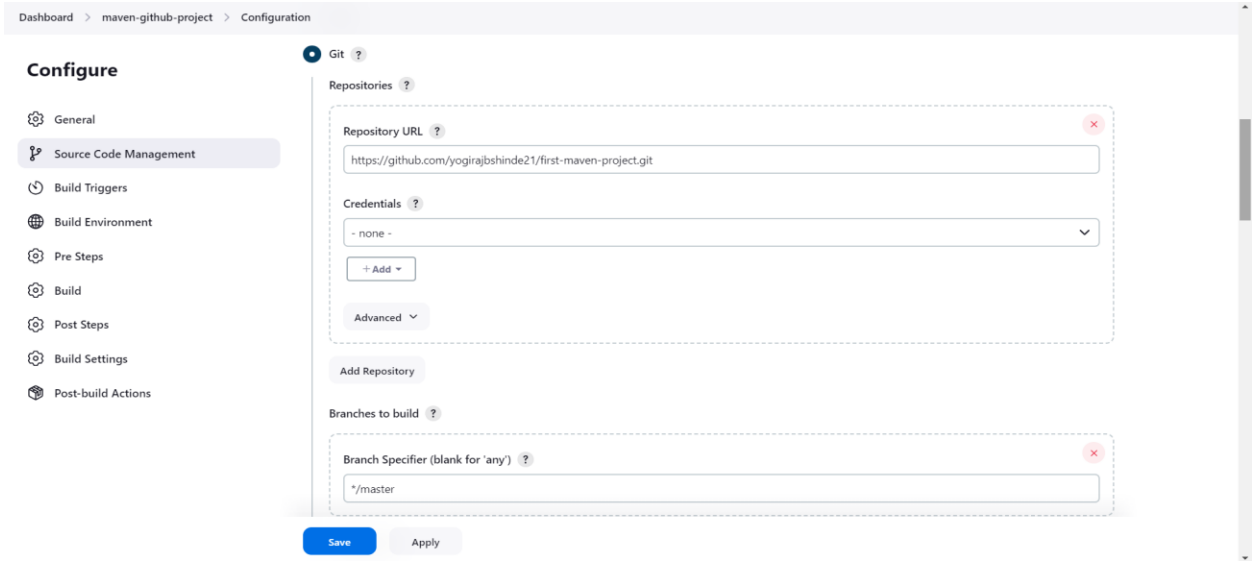


6. Integrate with Jenkins:

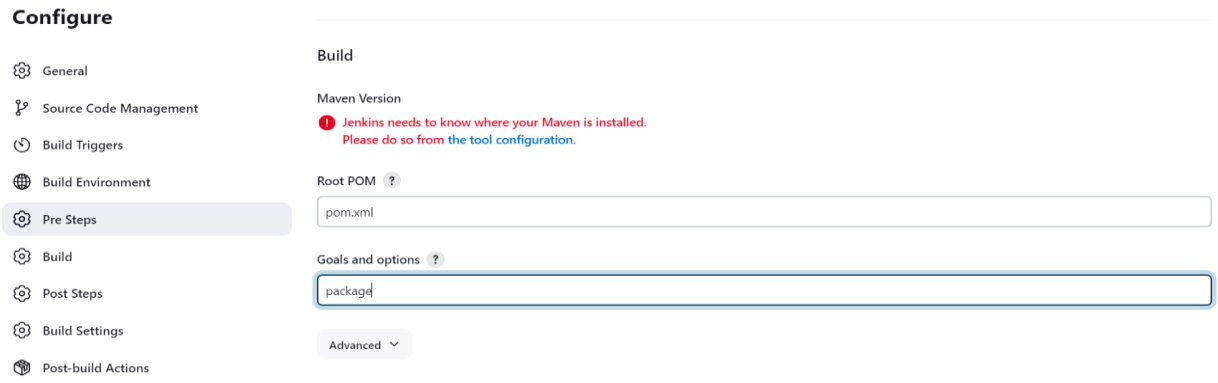
- Create a new job in Jenkins and select **Maven Project**.



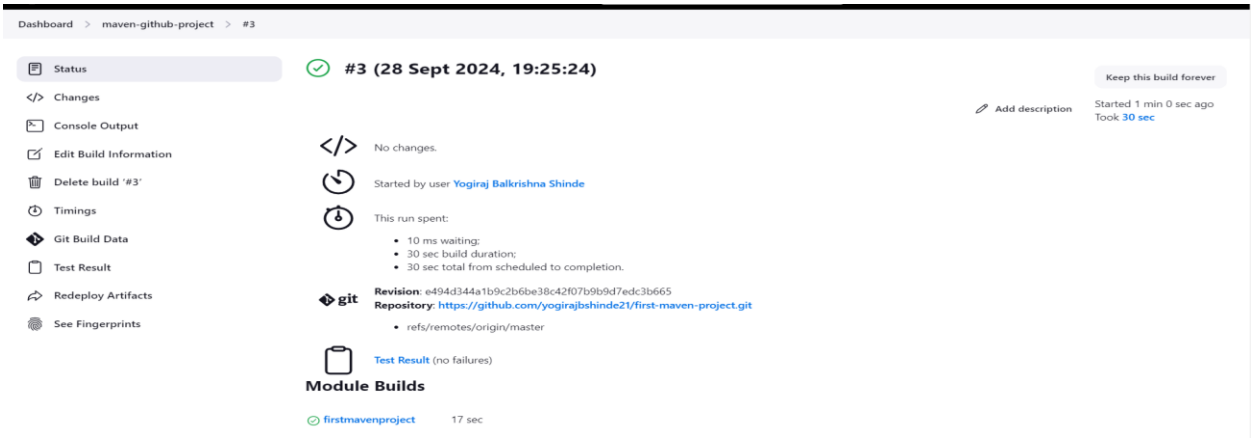
- Configure GitHub repository details in the Source Code Management section.

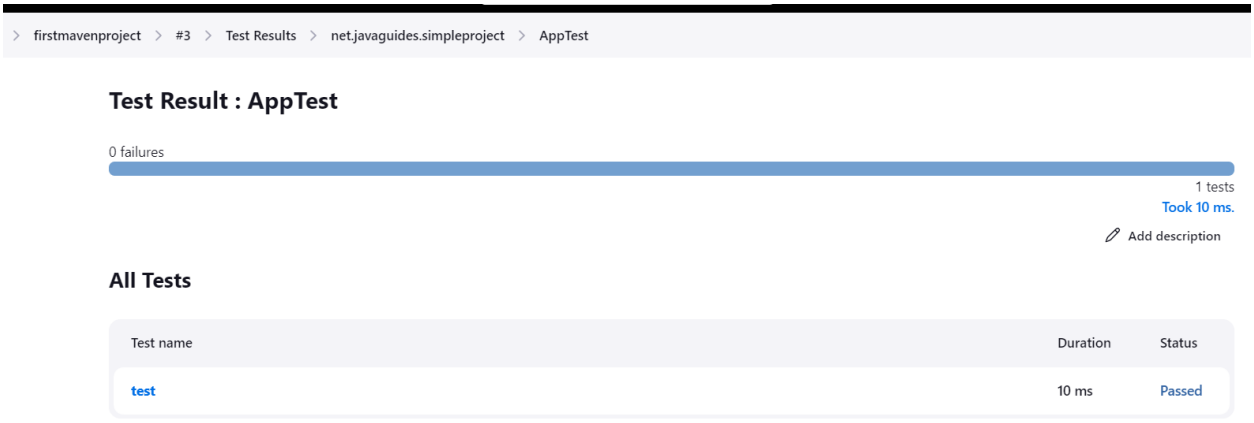


- Set build triggers and goals in the build section (e.g., `clean install`).



- Build the project.





Conclusion:

Successfully executed a simple Maven project in Eclipse, pushed it to GitHub, and integrated it with Jenkins for automated builds.

Name: Kamal Agrahari **Lab: DevOps**

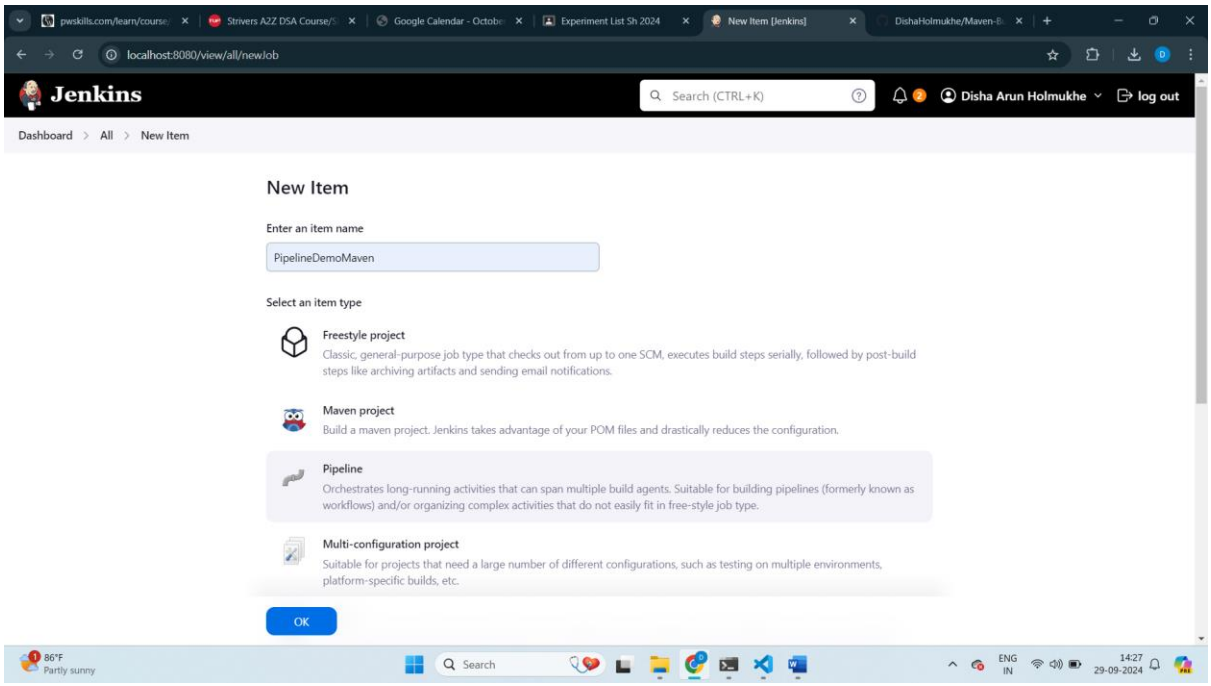
ID: VU4F2223028 **TE | IT | A**

Experiment:08

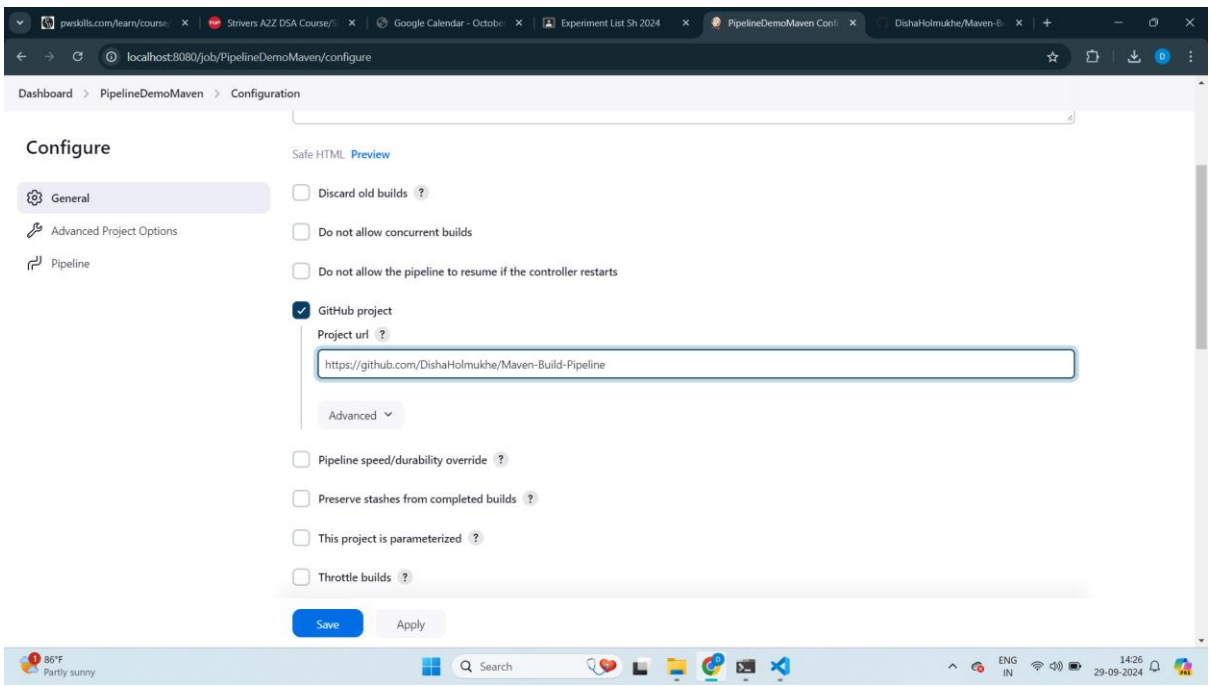
Aim: Demonstration to Create Maven Pipeline.

Theory:

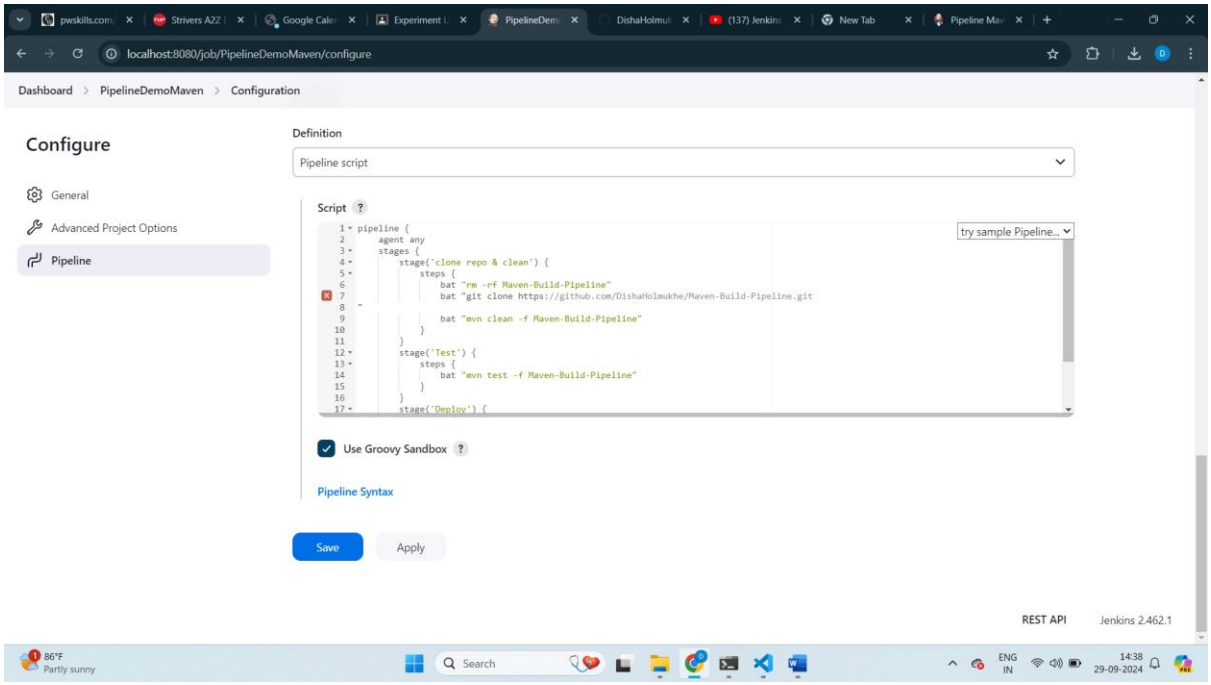
Step1: Create a new item/job with itme type as pipeline.



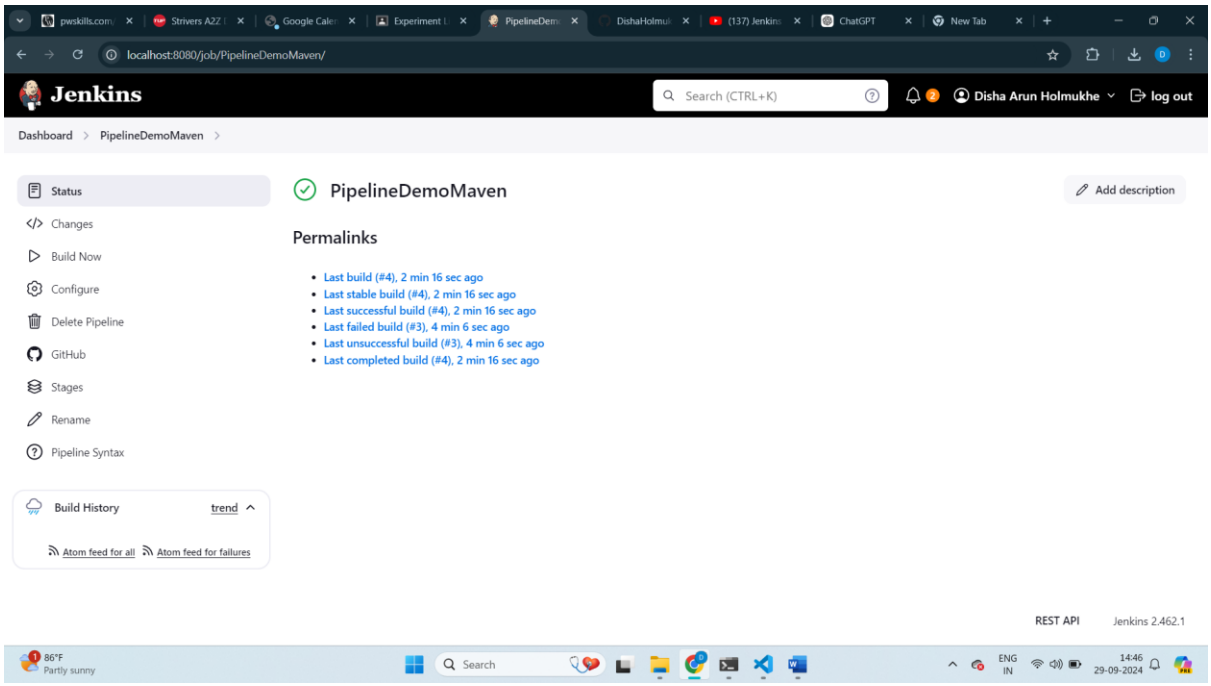
Step2: In General set github project with the github url to the maven project.



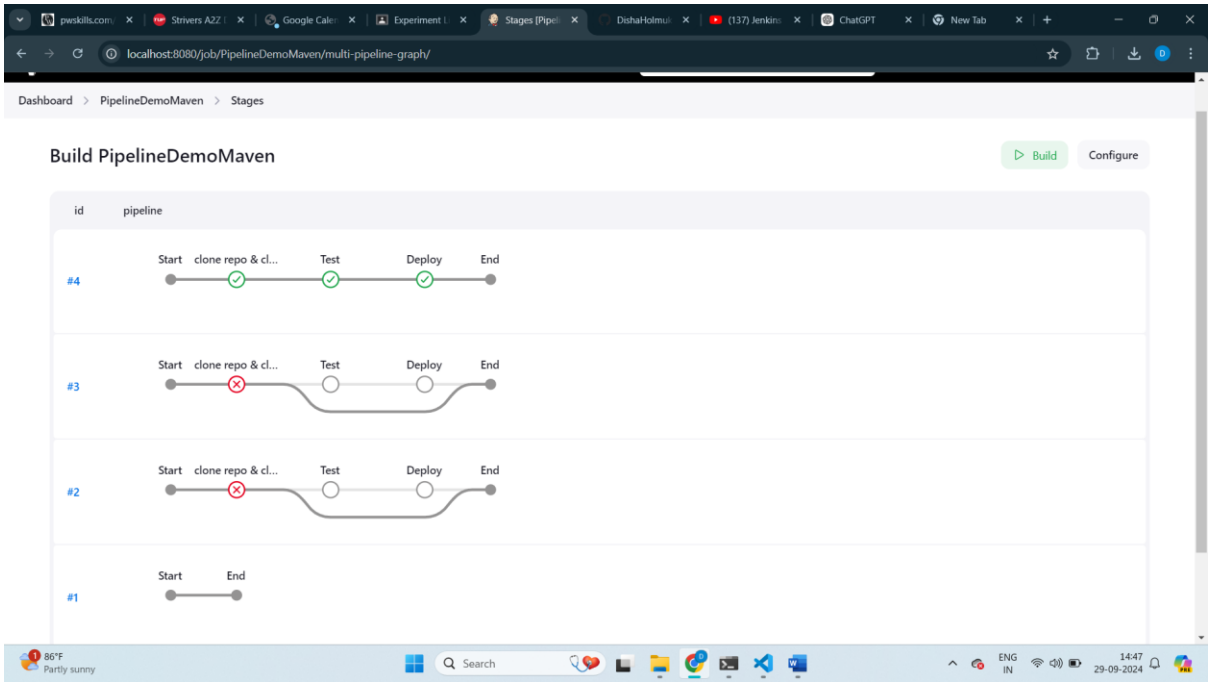
Step3: In Pipeline choose Pipeline Script



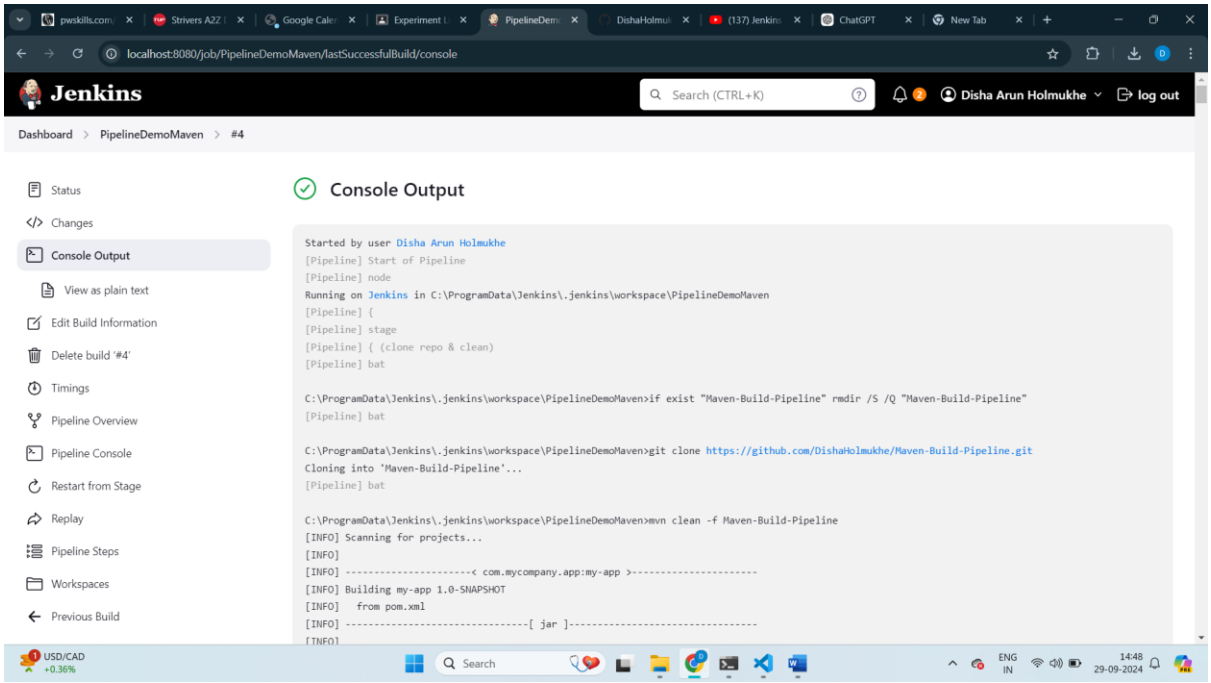
Step4: Click save and Build now.

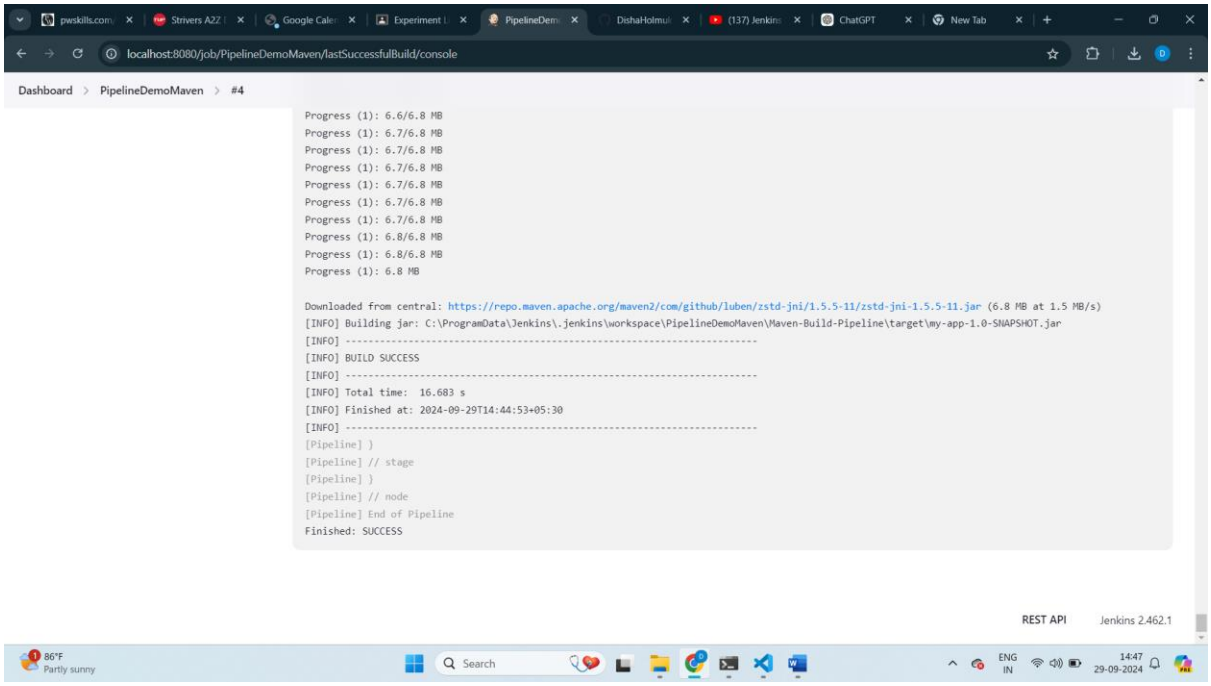


Step5: Click on stages to see the execution of pipeline in each stage.



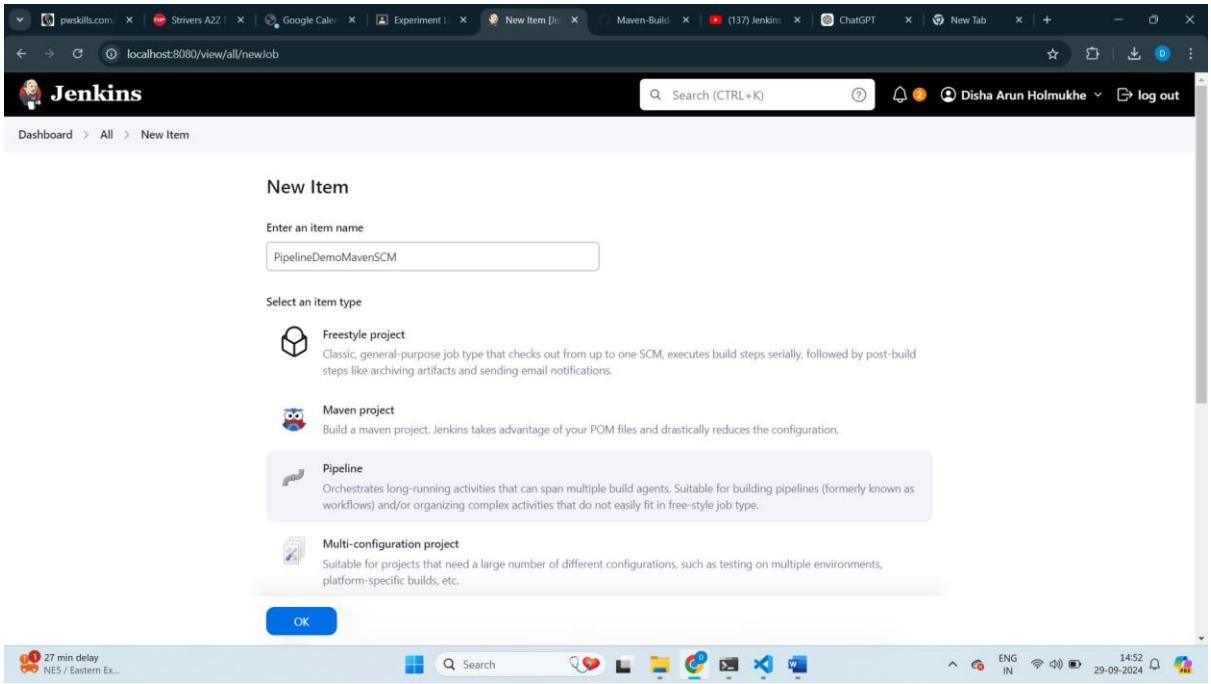
Step6: The console output displays “Finished:SUCCESS”. The pipeline is executed successfully.



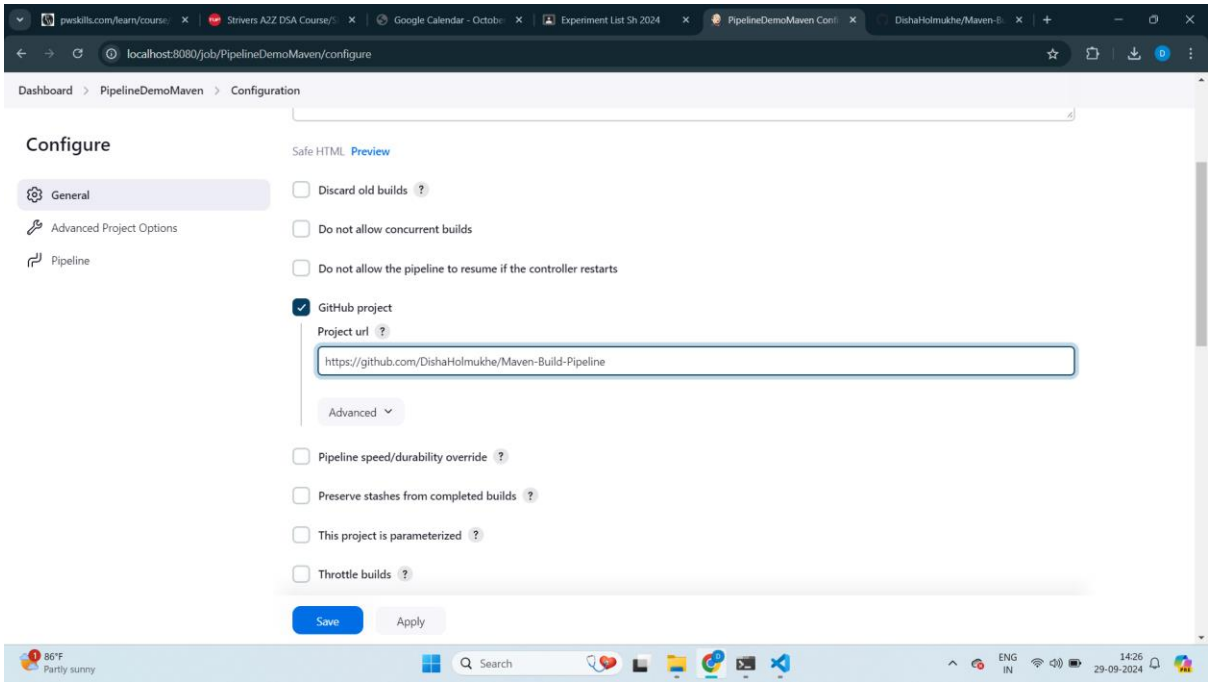


Building Maven pipeline using Pipeline script SCM

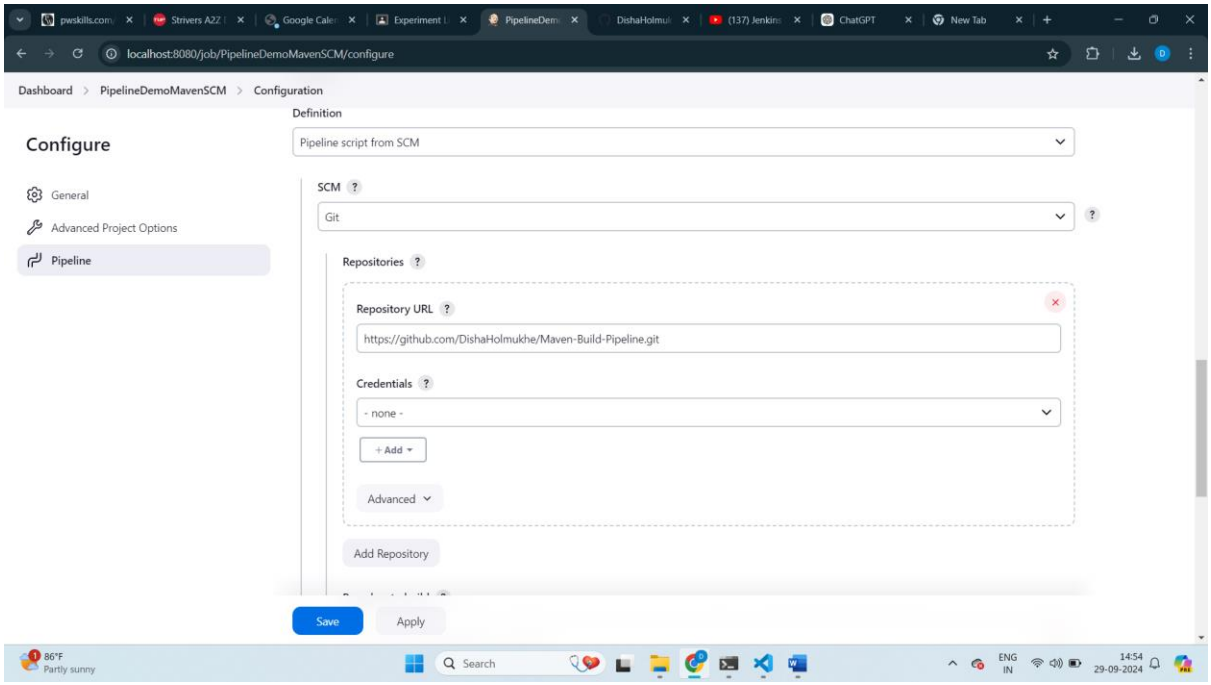
Step1: Create a new item/job with item type as pipeline.



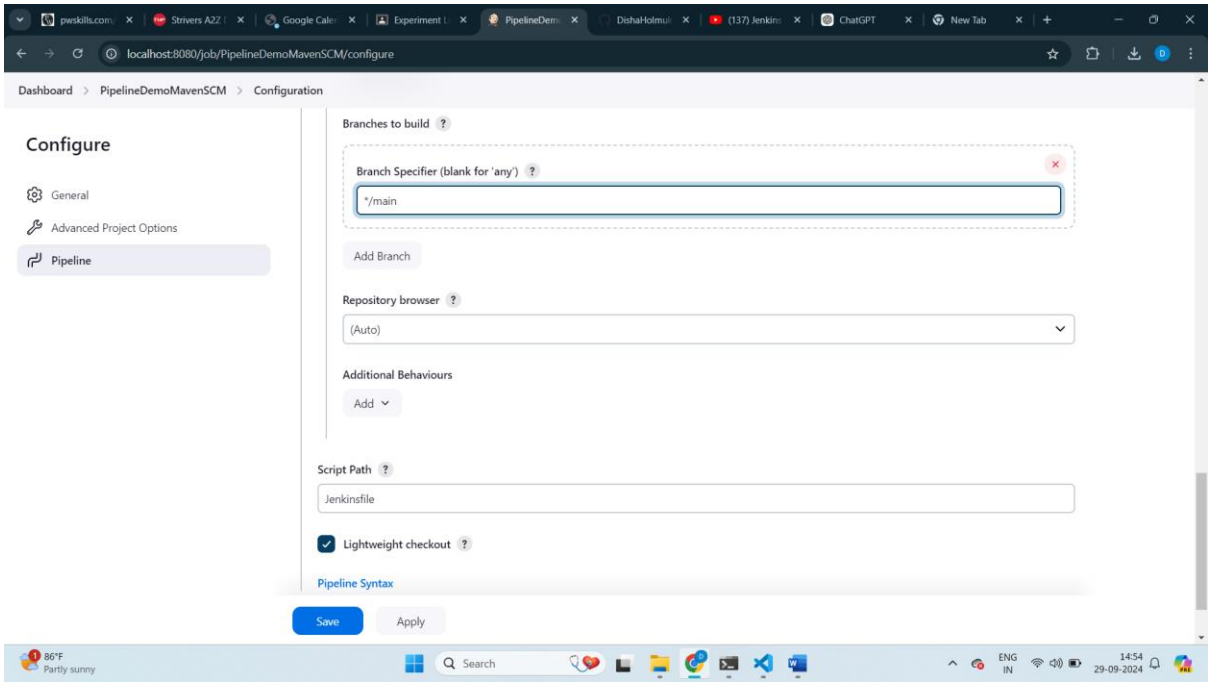
Step2: In General set github project with the github url to the maven project.



Step3: In pipeline select Pipeline Script from SCM.

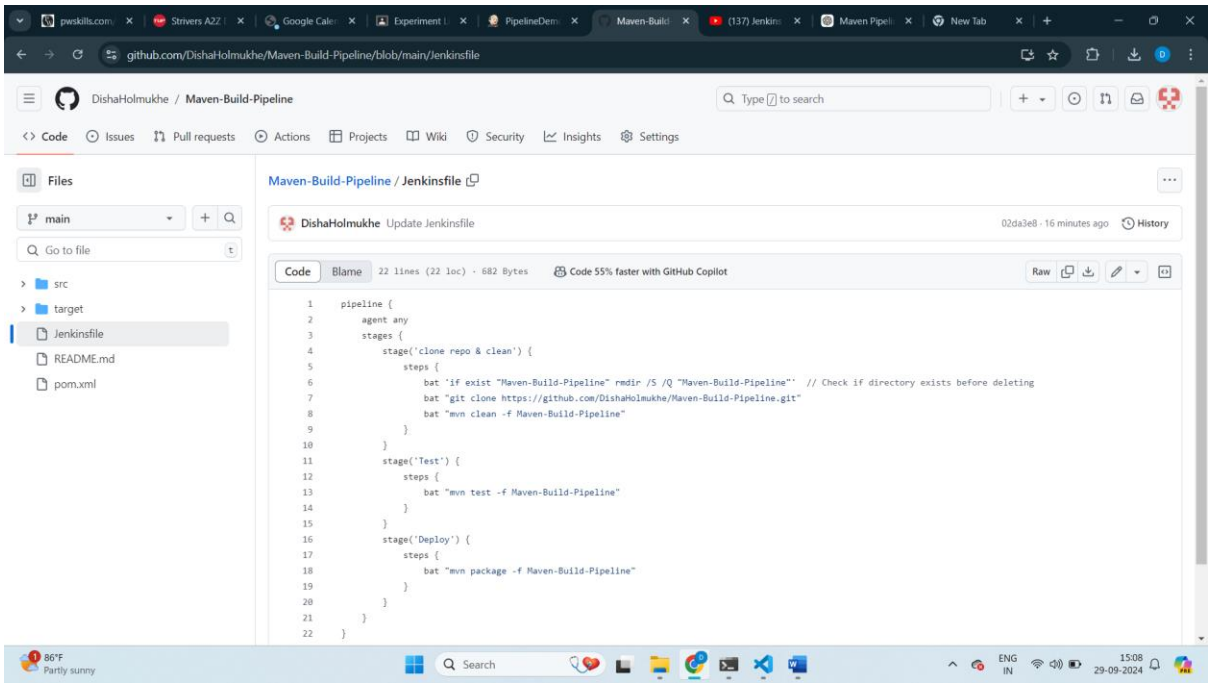


Step4: Set branch to main and Script path to the Jenkins file of your github repository.

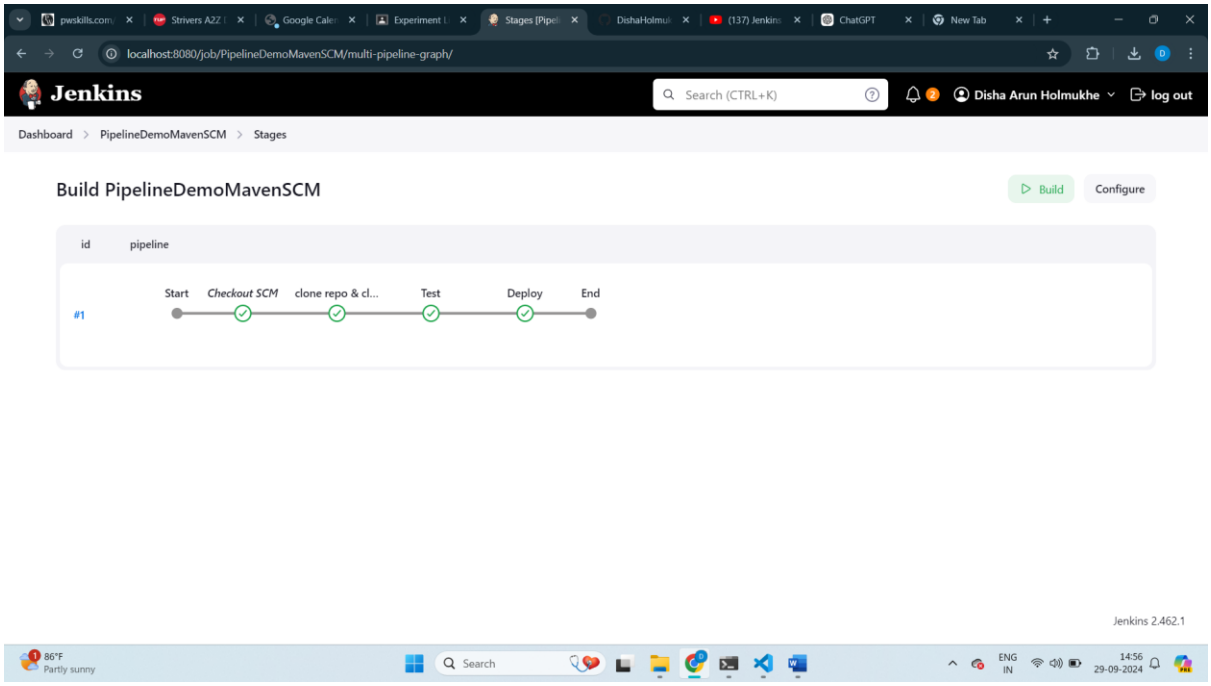


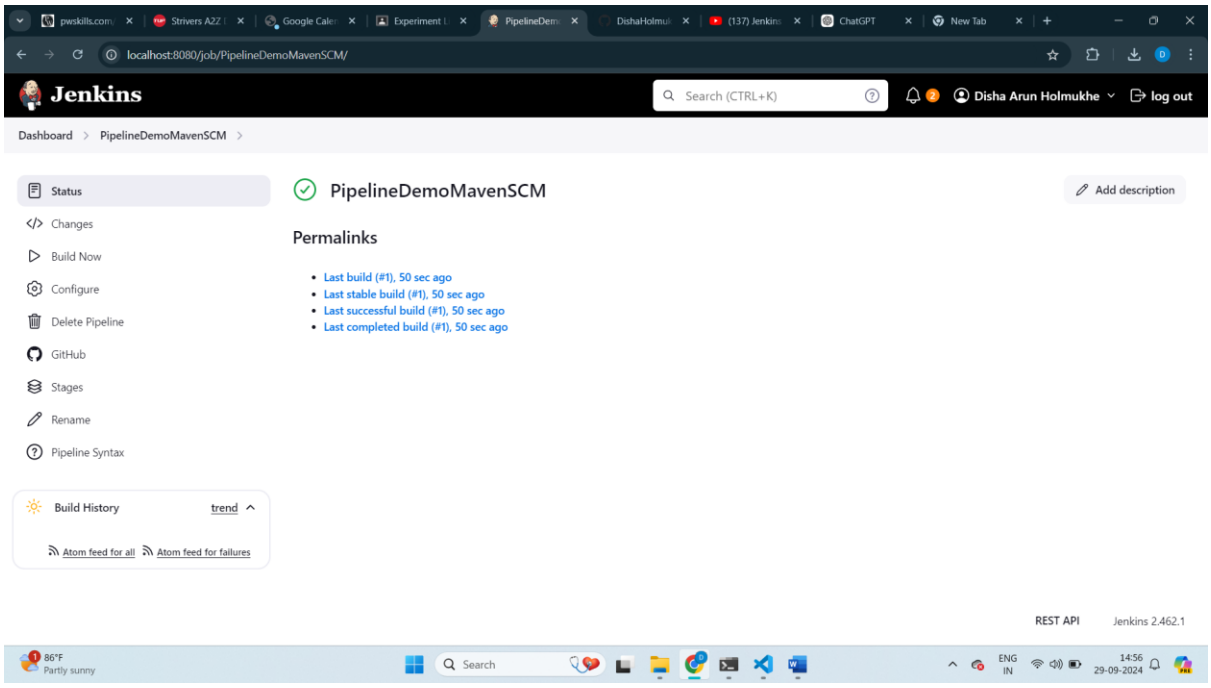
Step5: Click save and build now.

Step6: The Jenkinsfile should be in the Root of your repository.

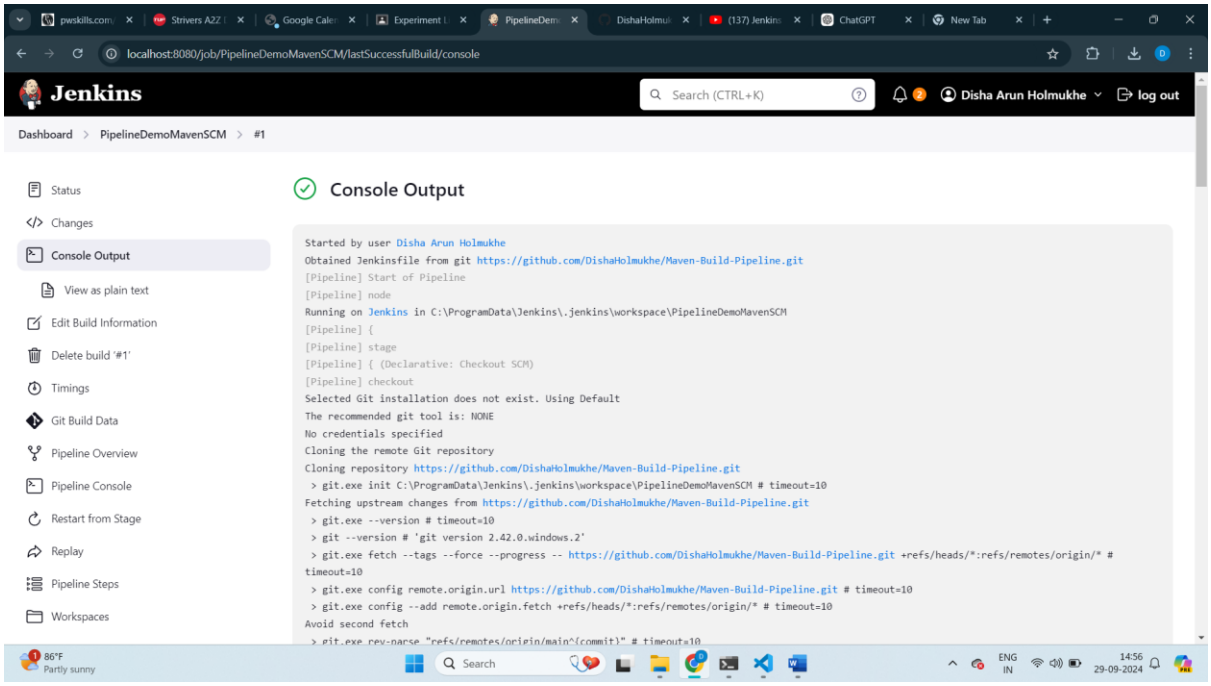


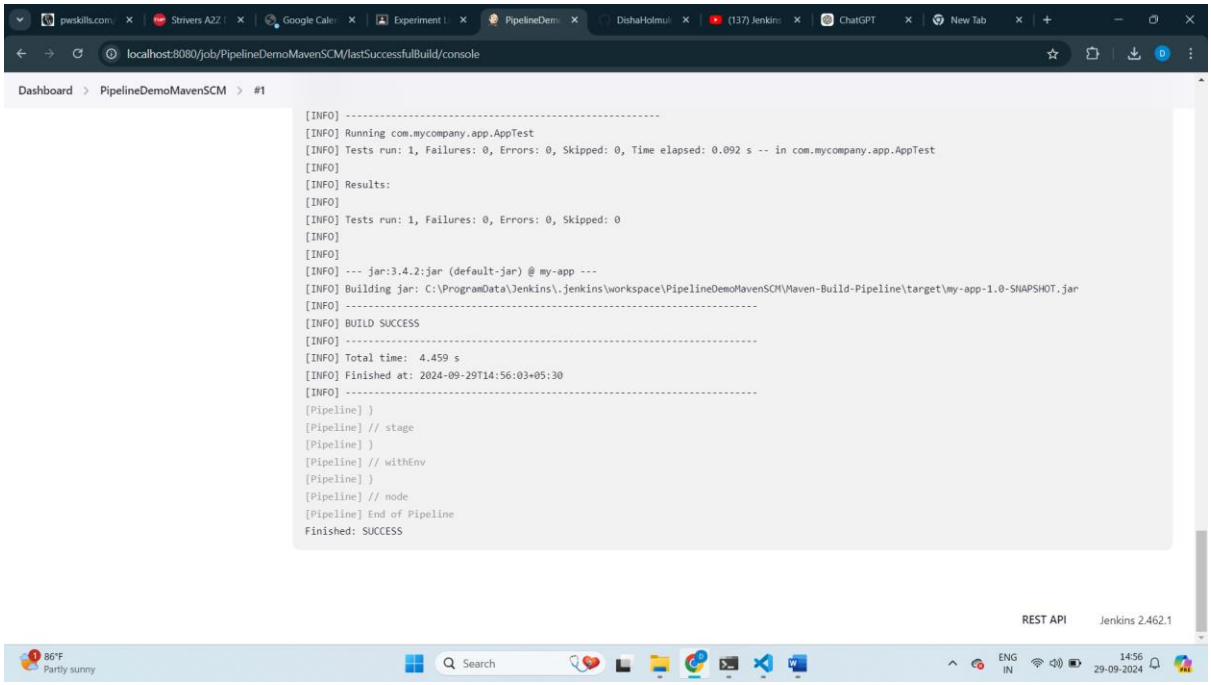
Step5: Click on stages to see the execution of pipeline in each stage.





Step6: The console output displays “Finished : SUCCESS”. The pipeline is executed successfully.





Conclusion: Successfully completed the demonstration to Create Maven Pipeline.

Name: Kamal Agrahari
ID: VU4F2223028

Lab: DevOps
Class: TE/A

Experiment no. 9

Aim : Running selenium test cases to test web-based UI components.

Theory :

Running Selenium test cases to test web-based UI components in Python involves several steps. Below, I'll outline a basic setup and provide an example to help you get started.

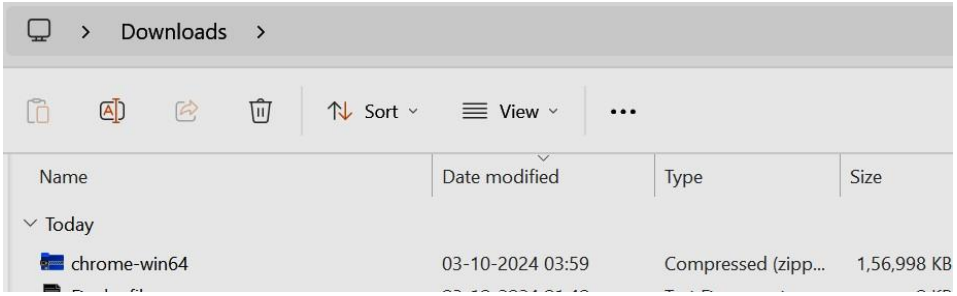
1. Install Required Packages

First, ensure you have Python installed on your machine. Then, install Selenium using pip:
pip install selenium

```
C:\Users\ADMIN>pip install selenium
Collecting selenium
  Downloading selenium-4.25.0-py3-none-any.whl.metadata (7.1 kB)
Requirement already satisfied: urllib3<3,>=1.26 in c:\users\admin\appdata\local\programs\python\python312\lib\site-packages (from urllib3[socks]<3,>=1.26->s
elenium) (2.2.0)
Collecting trio~=0.17 (from selenium)
  Downloading trio-0.26.2-py3-none-any.whl.metadata (8.6 kB)
Collecting trio-websocket~=0.9 (from selenium)
  Downloading trio_websocket-0.11.1-py3-none-any.whl.metadata (4.7 kB)
Requirement already satisfied: certifi>=2021.10.8 in c:\users\admin\appdata\local\programs\python\python312\lib\site-packages (from selenium) (2024.2.2)
Requirement already satisfied: typing_extensions~=4.9 in c:\users\admin\appdata\local\programs\python\python312\lib\site-packages (from selenium) (4.10.0)
Collecting websocket-client~=1.8 (from selenium)
  Downloading websocket_client-1.8.0-py3-none-any.whl.metadata (8.0 kB)
Collecting attrs>=23.2.0 (from trio~=0.17->selenium)
  Downloading attrs-24.2.0-py3-none-any.whl.metadata (11 kB)
Collecting sortedcontainers (from trio~=0.17->selenium)
  Downloading sortedcontainers-2.4.0-py2.py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: idna in c:\users\admin\appdata\local\programs\python\python312\lib\site-packages (from trio~=0.17->selenium) (3.6)
Collecting outcome (from trio~=0.17->selenium)
  Downloading outcome-1.3.0.post0-py2.py3-none-any.whl.metadata (2.6 kB)
Collecting sniffio>=1.3.0 (from trio~=0.17->selenium)
  Downloading sniffio-1.3.1-py3-none-any.whl.metadata (3.9 kB)
Requirement already satisfied: cffi>=1.14 in c:\users\admin\appdata\local\programs\python\python312\lib\site-packages (from trio~=0.17->selenium) (1.16.0)
Collecting wsproto>=0.14 (from trio-websocket~=0.9->selenium)
  Downloading wsproto-1.2.0-py3-none-any.whl.metadata (5.6 kB)
Collecting pysocks!=1.5.7,<2.0,>=1.5.6 (from urllib3[socks]<3,>=1.26->selenium)
  Downloading PySocks-1.7.1-py3-none-any.whl.metadata (13 kB)
Requirement already satisfied: pycparser in c:\users\admin\appdata\local\programs\python\python312\lib\site-packages (from cffi>=1.14->trio~=0.17->selenium)
 (2.21)
Collecting h11<1,>=0.9.0 (from wsproto>=0.14->trio-websocket~=0.9->selenium)
  Downloading h11-0.14.0-py3-none-any.whl.metadata (8.2 kB)
Downloading selenium-4.25.0-py3-none-any.whl (9.7 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.3/9.7 MB 685.6 kB/s eta 0:00:04
```

2. Set Up a WebDriver

Selenium requires a WebDriver to interface with the browser. For example, if you're using Chrome, you'll need ChromeDriver. Make sure you download the driver that matches your installed Chrome version from here.



3. Example Test Case

Here's a basic example of a Selenium test case that opens a web page, interacts with elements, and verifies the page title.

Example Code-

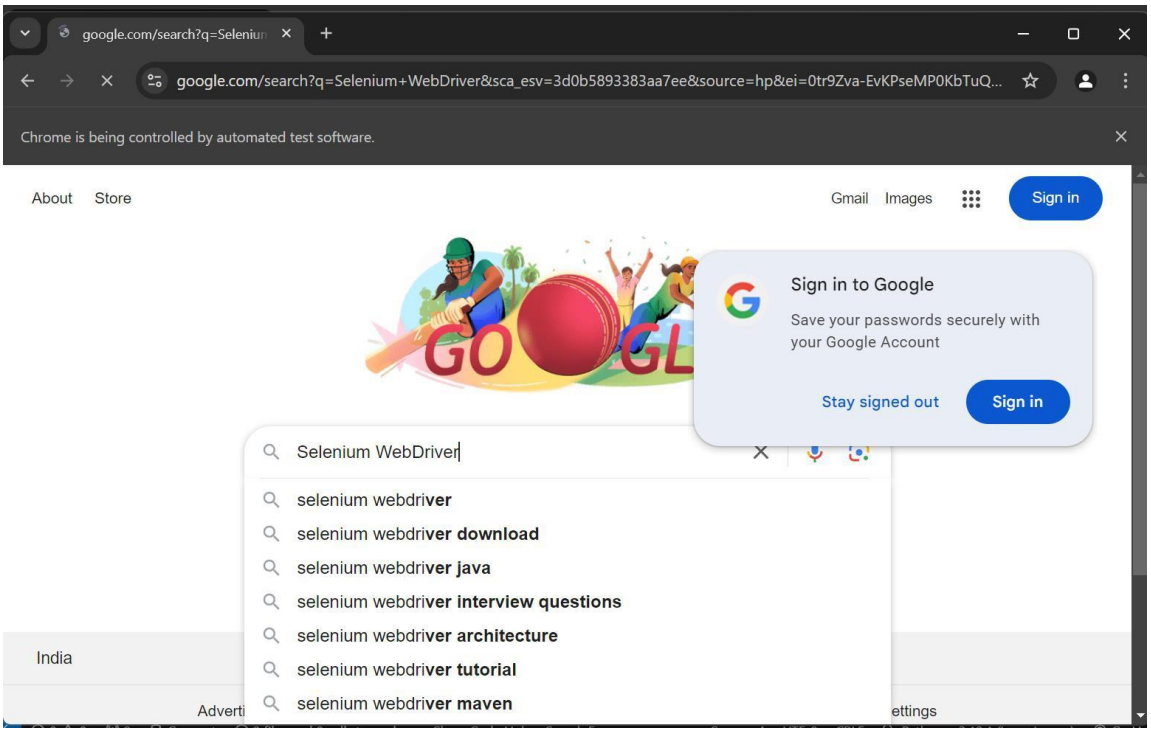
Test.py

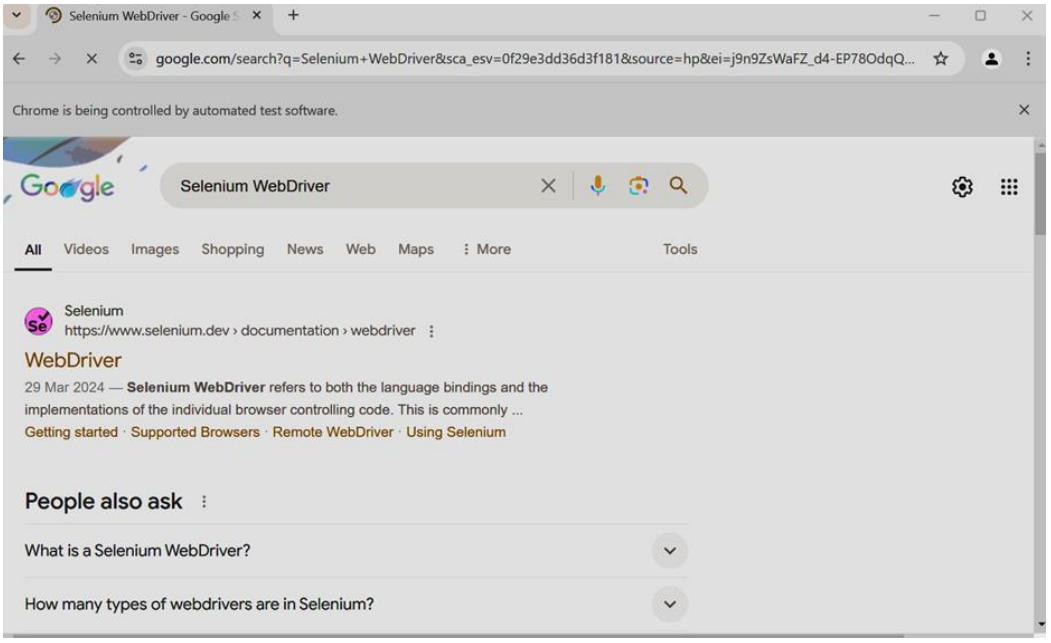
```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys from
selenium.webdriver.chrome.service import Service import
time

# Set the path to your WebDriver (e.g., ChromeDriver)
driver_path='C:\\Users\\ADMIN\\chromedriver-win64\\chromedriver.exe'
#yourChromedriverPath service
= Service(driver_path)
driver = webdriver.Chrome(service=service)
try:
  # Open a web page
  driver.get('http://google.com')  # Replace with your target URL

  # Wait for the page to load
  time.sleep(2)  # Implicit wait; you can also use WebDriverWait for better practices

  # Interact with elements
  search_box = driver.find_element(By.NAME, 'q')  # Example element
  search_box.send_keys('Selenium WebDriver' + Keys.RETURN)

  # Wait for results to load
  time.sleep(2)

  # Verify the title
  assert 'Selenium WebDriver' in driver.title

  print("Test Passed: Page title is correct.")
except Exception as e:
  print(f"Test Failed: {e}")
finally:
  # Close the browser
  driver.quit()
```

4. Run Your Test Case

1. Save the script in a .py file (e.g., test_script.py).
2. Run it using Python:

$ python test_script.py






```
C:\Users\ADMIN>python test.py

DevTools listening on ws://127.0.0.1:49242/devtools/browser/5e479cb9-4873-
49b2-8426-051f2a8c8b81
Test Passed: Page title is correct.
```

Conclusion: Hence we have performed selenium test on web-based UI components.

Name: Kamal Agrahari
ID: VU4F2223028

Lab: DevOps
Class: TE / A

Experiment No. 10

Aim: Installation of Docker.

Theory:

- Docker

- Docker is an open-source platform that automates the deployment, scaling, and management of applications using containerization.
- Containers are lightweight, portable, and self-sufficient environments that package an application and all its dependencies, ensuring consistent operation across different environments.

- Key Concepts:

  - Container: A lightweight, standalone, executable package that includes everything needed to run a piece of software, including the code, runtime, libraries, and system tools.
  - Image: A read-only template used to create containers. Images can be thought of as the blueprint for a container.
  - Dockerfile: A text file containing a series of instructions on how to build a Docker image.
  - Docker Hub: A cloud-based repository for sharing Docker images, allowing you to download existing images or upload your own.
  - Docker Engine: The core component of Docker, responsible for running containers and managing container lifecycle.

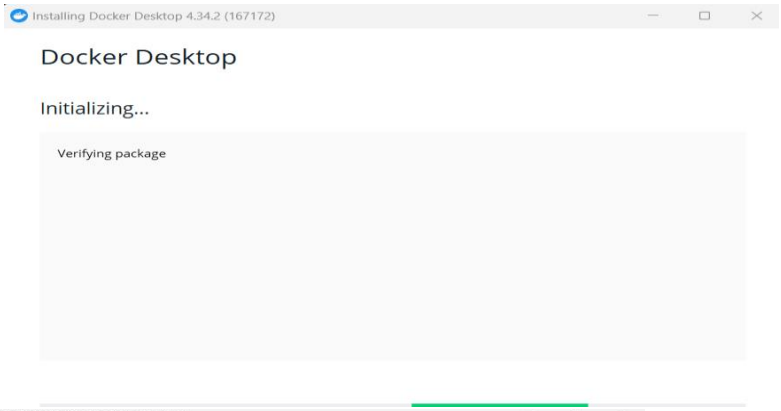
- Installation of Docker

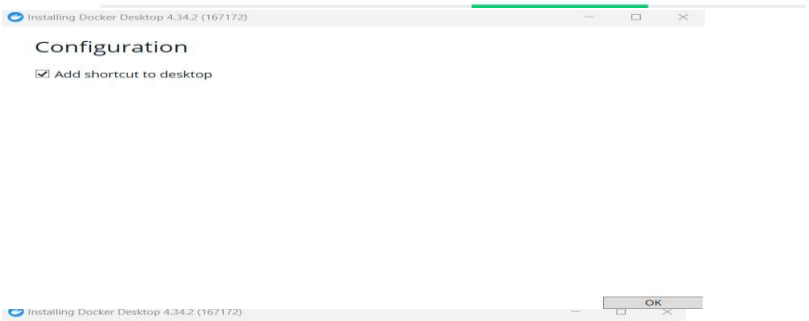
- Requirements: Windows 10 64-bit: Pro, Enterprise, or Education (Build 15063 or later).

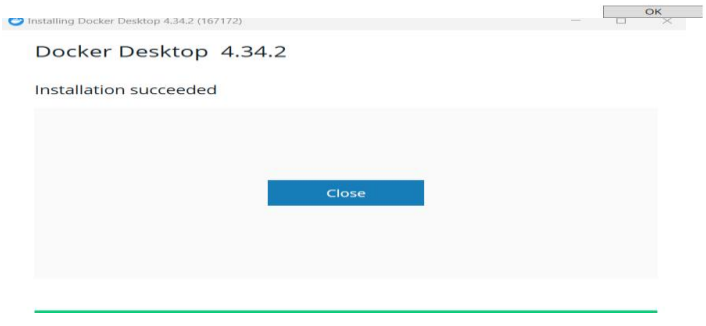
- Steps:

1. Download Docker Desktop: Go to the Docker Desktop for Windows page and download the installer.

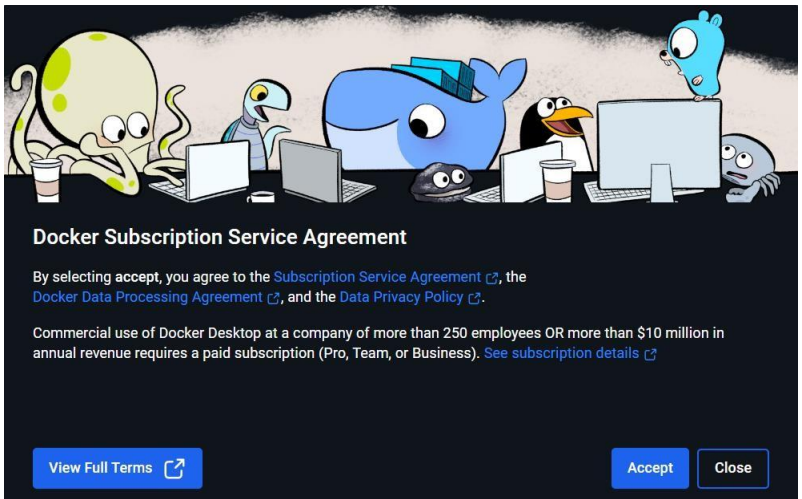
2. Run the Installer: Double-click the downloaded file and follow the installation instructions. Ensure the “Install required Windows components for WSL 2” option is checked if prompted.

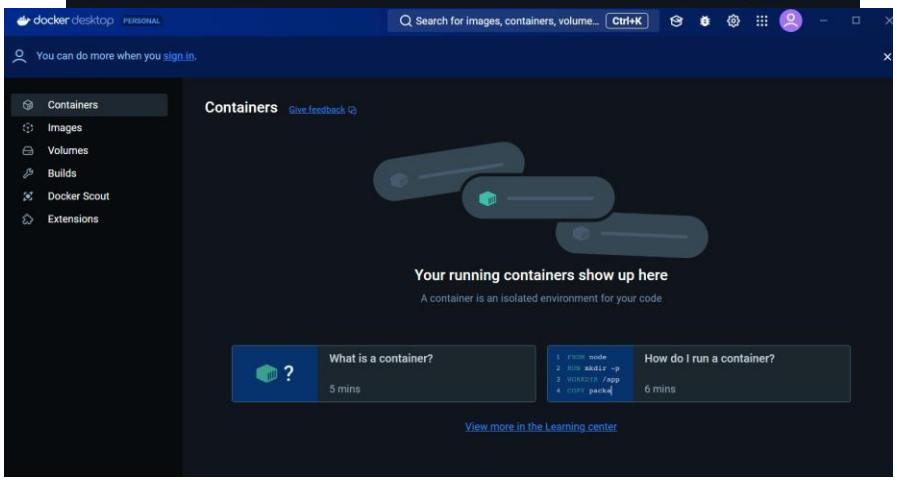




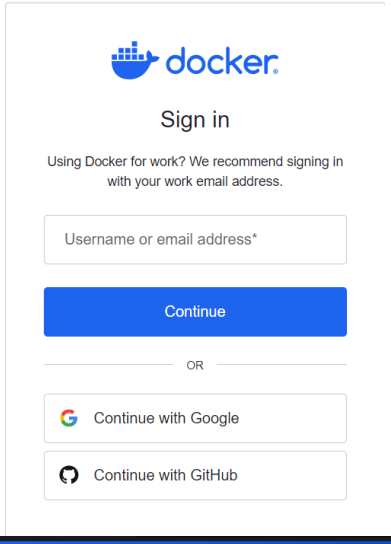


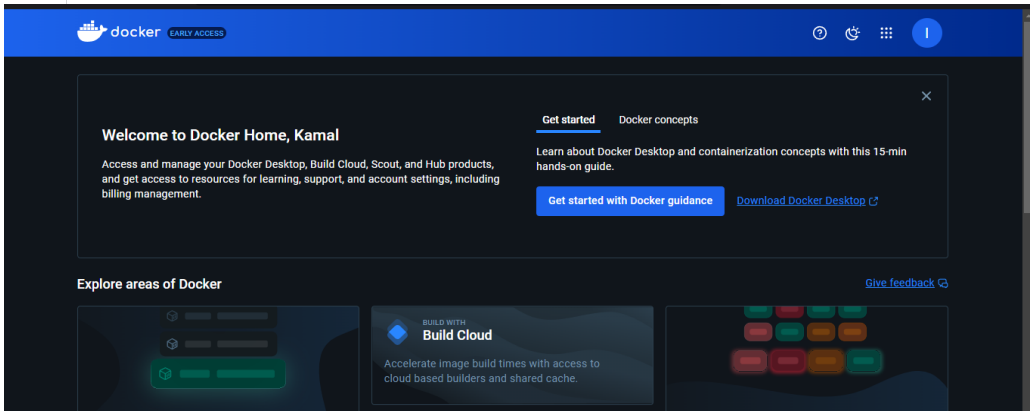
3. Start Docker Desktop: After installation, launch Docker Desktop from the Start menu.



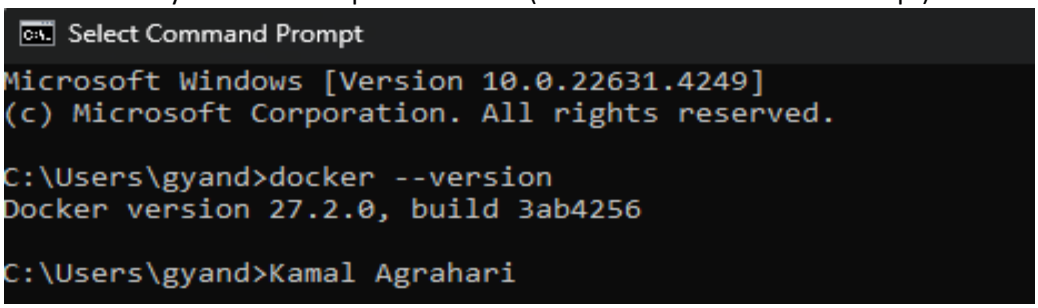


4. Configuration: Follow the onboarding instructions to complete the setup, including creating a Docker account if necessary.





5. Verify Installation: Open a terminal (PowerShell or Command Prompt) and run:



Conclusion:

Hence, we successfully installed the Docker.

Name: Kamal Agrahari
ID: VU4F2223028

Lab: DevOps
Class: TE / A

Experiment No. 11

Aim: Building Docker Image.

Theory:

 What is a Docker Image?

- A Docker image is a lightweight, stand-alone, and executable package that includes everything needed to run an application, including the code, runtime, libraries, environment variables, and configuration files.
- Images are immutable, meaning once they are created, they cannot be changed. Instead, new images are built on top of existing images.

 Key Characteristics of Docker Images:

- Layered File System: Docker images are composed of layers. Each layer represents a set of file changes or operations, such as adding files, modifying files, or installing software. This layering system allows for efficient storage and sharing since multiple images can share common layers.
- Read-Only: Once an image is created, it becomes read-only. Any changes made to a running container (an instance of the image) are stored in a new writable layer on top of the image.
- Versioning: Images can be tagged and versioned, allowing you to manage different versions of your application easily.

 Building a Docker Image

1. Create a Dockerfile

Dockerfile

```
#STEP1 sepcify the base image
FROM alpine

#STEP2 download and install dependencies
RUN apk add --update redis

#STEP3 setup the strtup command
CMD ["redis-server"]
```

2. Build the Docker Image

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS

PS C:\Users\Sayali Nimbalkar\Desktop\Docker File> docker build .
[+] Building 10.5s (6/6) FINISHED docker:desktop-linux => sha256:43c4264eed91be 3.62MB / 3.62MB  1.3s
 => [internal] load build definition from Do  0.0s
 => => transferring dockerfile: 211B          0.0s
 => [internal] load metadata for docker.io/l  5.0s
 => [internal] load .dockerignore             0.0s
 => => transferring context: 2B               0.0s
 => [1/2] FROM docker.io/library/alpine:late  1.7s
 => => resolve docker.io/library/alpine:late  0.1s
 => => sha256:43c4264eed91be 3.62MB / 3.62MB  1.3s
 => => extracting sha256:43c4264eed91be63b20  0.3s
 => [2/2] RUN apk add --update redis          2.9s
 => => resolve docker.io/library/alpine:late  0.1s
 => => sha256:43c4264eed91be 3.62MB / 3.62MB  1.3s
 => => extracting sha256:43c4264eed91be63b20  0.3s
 => => resolve docker.io/library/alpine:late  0.1s
 => => resolve docker.io/library/alpine:late  0.1s
 => => sha256:43c4264eed91be 3.62MB / 3.62MB  1.3s
 => => extracting sha256:43c4264eed91be63b20  0.3s
 => => resolve docker.io/library/alpine:late  0.1s
 => => sha256:43c4264eed91be 3.62MB / 3.62MB  1.3s
 => => resolve docker.io/library/alpine:late  0.1s
 => => sha256:43c4264eed91be 3.62MB / 3.62MB  1.3s
 => => exporting layers                       0.5s
 => => exporting manifest sha256:111f83d7a91  0.0s
 => => exporting layers                       0.5s
```

3. Verify the Built Image

```
C:\Users\gyand>docker images
REPOSITORY              TAG       IMAGE ID       CREATED         SIZE
nginx/nginx-ingress     latest    3f339f1b3bd8   47 hours ago    281MB

C:\Users\gyand>Kamal Agrahari_
```

4. Run the image by the “image ID” to check if it’s working correctly.

```
PS C:\Users\Sayali Nimbalkar\Desktop\Docker File> docker run a4e47e3c2b51
1:C 02 Oct 2024 12:00:09.836 * oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
1:C 02 Oct 2024 12:00:09.836 * Redis version=7.2.5, bits=64, commit=00000000, modified=0, pid=1, just started
1:C 02 Oct 2024 12:00:09.836 # Warning: no config file specified, using the default config. In order to specify a config file use redis-server /path/to/redis.conf
1:M 02 Oct 2024 12:00:09.837 * monotonic clock: POSIX clock_gettime
1:M 02 Oct 2024 12:00:09.838 * Running mode=standalone, port=6379.
1:M 02 Oct 2024 12:00:09.839 * Server initialized
1:M 02 Oct 2024 12:00:09.840 * Ready to accept connections tcp
1:signal-handler (1727870691) Received SIGINT scheduling shutdown...
1:M 02 Oct 2024 12:04:51.412 * User requested shutdown...
1:M 02 Oct 2024 12:04:51.413 * Saving the final RDB snapshot before exiting.
1:M 02 Oct 2024 12:04:51.423 * DB saved on disk
1:M 02 Oct 2024 12:04:51.424 # Redis is now ready to exit, bye bye...
```

5. Tagging a name to the image.

```
PS C:\Users\Sayali Nimbalkar\Desktop\Docker File> docker build -t sayali/redis:latest .
[+] Building 2.5s (6/6) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 211B
 => [internal] load metadata for docker.io/library/alpine:latest
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [1/2] FROM docker.io/library/alpine:latest@sha256:beefdbd8a1da6d2915566fde36db9db0b524eb737fc57cd1367effd16dc0d06d
 => => resolve docker.io/library/alpine:latest@sha256:beefdbd8a1da6d2915566fde36db9db0b524eb737fc57cd1367effd16dc0d06d
 => CACHED [2/2] RUN apk add --update redis
 => exporting to image
 => => exporting layers
 => => exporting manifest sha256:111f83d7a91f0b1bf7b7063368e9db52a9ae8e65b1ae22326c2ca71a9af8b7e9
 => => exporting config sha256:68f0da0c44ea6b47fc9c4d0a87c9924021dde87d654438d1608a67b539466420
 => => exporting attestation manifest sha256:8b664ca3422f4fd4b168c21ff2244db317356d2a8b326d40b2af69be27971748
 => => exporting manifest list sha256:0d53756c965006149bdfdf69017a91cd368ce1dd16a4295bb6d30a49eff076c9
 => => naming to docker.io/sayali/redis:latest
 => => unpacking to docker.io/sayali/redis:latest
```

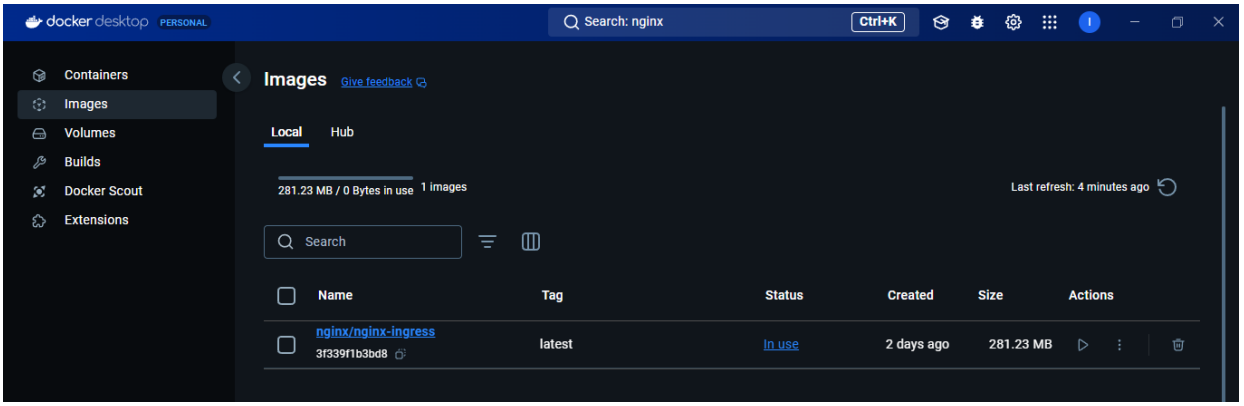
6. Running the image by given name.

```
PS C:\Users\Sayali Nimbalkar\Desktop\Docker File> docker run sayali/redis
1:C 02 Oct 2024 12:10:20.241 * oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
1:C 02 Oct 2024 12:10:20.241 * Redis version=7.2.5, bits=64, commit=00000000, modified=0, pid=1, just started
1:C 02 Oct 2024 12:10:20.241 # Warning: no config file specified, using the default config. In order to specify a config file use redis-server /path/to/redis.conf
1:M 02 Oct 2024 12:10:20.241 * monotonic clock: POSIX clock_gettime
1:M 02 Oct 2024 12:10:20.244 * Running mode=standalone, port=6379.
1:M 02 Oct 2024 12:10:20.247 * Server initialized
1:M 02 Oct 2024 12:10:20.251 * Ready to accept connections tcp
1:signal-handler (1727871074) Received SIGINT scheduling shutdown...
1:M 02 Oct 2024 12:11:14.978 * User requested shutdown...
1:M 02 Oct 2024 12:11:14.978 * Saving the final RDB snapshot before exiting.
1:M 02 Oct 2024 12:11:14.988 * DB saved on disk
1:M 02 Oct 2024 12:11:14.988 # Redis is now ready to exit, bye bye...
PS C:\Users\Sayali Nimbalkar\Desktop\Docker File>
```

7. Verifying it.

```
C:\Users\gyand>docker images
REPOSITORY              TAG       IMAGE ID       CREATED          SIZE
nginx/nginx-ingress     latest    3f339f1b3bd8   47 hours ago     281MB

C:\Users\gyand>Kamal Agrahari_
```

8. You can see the image in your docker desktop.



Conclusion:

Hence, we successfully build the Docker image.

Name: Kamal Agrahari Lab: DevOps

ID: VU4F2223028 TE | IT | A

Experiment No. 12

Title: Demonstration of ansible automation.

Theory:

Ansible is an open-source automation tool that simplifies the process of configuration management, application deployment, and task automation across a network of machines. It uses a simple and human-readable language (YAML) for writing automation scripts called **playbooks**. This demonstration outlines the steps to automate a basic task using Ansible.

Objective

In this demonstration, we will automate the installation of **Nginx** on multiple remote servers using Ansible.

Prerequisites

1. **Ansible Installed**: Ensure Ansible is installed on your control machine (the machine from which you will run Ansible).
2. **SSH Access**: Ensure that you have SSH access to the target machines and that they are listed in the Ansible inventory file.
3. **Target Machines**: Have at least two target machines (remote servers) ready for Nginx installation.

Steps to follow:

Step 1: Create an inventory file (hosts.ini) to define the target machines. This file specifies the IP addresses or hostnames of the servers where you want to deploy Nginx.

```
[webservers]
server1 ansible_host=192.168.1.10
server2 ansible_host=192.168.1.11
```

Step 2: Create a playbook file (install_nginx.yml) to describe the tasks needed to install Nginx. This playbook will include tasks for installing Nginx and starting the service.

```
---
- name: Install Nginx on web servers
hosts: webservers
become: yes  # Use sudo to install packages
tasks:
- name: Update package manager cache
apt:
update_cache: yes

- name: Install Nginx
apt:
name: nginx
state: present

- name: Start Nginx service
service:
name:  nginx
state: started
enabled: yes
```

Step 3: To execute the playbook and install Nginx on the target servers, run the following command in your terminal:

```
ansible-playbook -i hosts.ini install_nginx.yml
```

Step 4: After running the playbook, you can verify that Nginx is installed and running on the target servers. SSH into one of the servers and run:

```
systemctl status nginx
```

Ansible to automate the installation of Nginx on multiple servers. By defining the target servers in an inventory file and outlining the installation steps in a playbook, we simplified

the deployment process, ensuring consistency and reducing manual effort. Ansible's ability to manage multiple systems simultaneously makes it an invaluable tool for system administrators and DevOps engineers.

```
devesh@LAPTOP-AKG6B6TB:~$
devesh@LAPTOP-AKG6B6TB:~$ sudo apt install ansible
[sudo] password for devesh:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ansible is already the newest version (9.2.0+dfsg-0ubuntu5).
0 upgraded, 0 newly installed, 0 to remove and 6 not upgraded.
devesh@LAPTOP-AKG6B6TB:~$
```

```
devesh@LAPTOP-AKG6B6TB:~$ cat inventory.txt
target1 ansible_ssh_user=ubuntu ansible_host=54.224.151.191 ansible_ssh_private_key_file=~/learning.pem ansibl
e_python_interpreter=/usr/bin/python3
devesh@LAPTOP-AKG6B6TB:~$
devesh@LAPTOP-AKG6B6TB:~$
```

```
devesh@LAPTOP-AKG6B6TB:~$
devesh@LAPTOP-AKG6B6TB:~$ ansible -m ping all -i inventory.txt
The authenticity of host '54.224.151.191 (54.224.151.191)' can't be established.
ED25519 key fingerprint is SHA256:WYrT2uVrngX0d0LTQiyzh8kEXllAKB0jbdpiZG3PQok.
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:1: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
target1 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
devesh@LAPTOP-AKG6B6TB:~$
```

Conclusion:

Hence, we have successfully demonstrate an ansible automation