Thadomal Shahani Engineering College

Bandra (W.), Mumbai - 400 050.

CERTIFICATE

Certify that Mr./~~Miss~~ AYUSH SANJAY SHARMA

of I.T Department, Semester V with

Roll No. 117 has completed a course of the necessary

experiments in the subject DEVOPS LAB under my

supervision in the **Thadomal Shahani Engineering College**

Laboratory in the year 2025 - 2026

Teacher In- Charge

Head of the Department

Date 09/10/25

Principal

THADOMAL SHAHANI TSEC ENGINEERING SOLLEGE

CONTENTS

09/10/25

ASSIGNMENT- 1

(Introduction to DevOps)

AIM: To understand DevOps: Principles, Practices, and DevOps Engineer Role and Responsibilities.

THEORY:

DevOps is a set of practices, principles, and cultural philosophies that aim to unify software development (Dev) and IT operations (Ops). The goal is to shorten the software development life cycle and provide continuous delivery with high software quality.

CHARACTERISTICS OF DEVOPS

- **Collaboration & Communication:** Cross-functional teams (developers, QA, and IT operations) work together closely.

- **Automation-Driven:** Processes like code integration, testing, deployment, and infrastructure provisioning are automated.
- **Continuous Integration and Delivery (CI/CD):** Software is built, tested, and deployed automatically and frequently.
- **Monitoring & Feedback:** Real-time monitoring and proactive incident management for performance and reliability.
- **Infrastructure as Code (IaC):** Infrastructure is managed and provisioned through code, allowing repeatability and scalability.

KEY FEATURES OF DEVOPS

Features	Description
Agility	Rapid delivery of features and updates.
Scalability	Easily scale systems and processes.
Reliability	Frequent testing and monitoring ensure fewer bugs in production.
Security	DevSecOps integrates security into the DevOps lifecycle.
Automation	Minimizes manual processes, reduces errors, and increases speed.

COMMON DEVOPS TOOLS

1. Version Control Tools

These tools help manage source code and track changes over time.

- **Git:** A distributed version control system that allows developers to track code changes and collaborate. It's the backbone of modern DevOps.

- **GitHub / GitLab / Bitbucket:** Platforms built on Git that offer repository hosting, collaboration, pull requests, issue tracking, and integration with CI/CD pipelines.

2. Continuous Integration & Continuous Deployment (CI/CD) Tools

These automate building, testing, and deploying applications.

- **Jenkins:** Open-source automation server used to build, test, and deploy code. It supports hundreds of plugins.
  GitLab CI/CD: Built-in CI/CD tool in GitLab that automates code pipelines.
- **CircleCI:** Cloud-based CI/CD tool with fast integration and deployment cycles.
- **Travis CI:** Simple CI/CD tool integrated with GitHub for automated testing.

3. Configuration Management Tools

These tools automate server setup and manage system configurations.

- **Ansible:** Agentless tool that uses simple YAML scripts to automate configuration and deployment.
- **Puppet:** Uses declarative language to manage system configurations across multiple servers.
- **Chef:** Uses Ruby-based scripts to configure and manage infrastructure.

4. Containerization Tools

These allow packaging of applications with all dependencies into lightweight containers.

- **Docker:** The most widely used containerization platform; packages applications and dependencies in isolated environments.

- **Podman:** A daemonless container engine offering Docker-like features with enhanced security and flexibility.

5. Container Orchestration Tools

These manage and scale containerized applications.

- **Kubernetes:** Open-source system for automating deployment, scaling, and management of containerized apps.
- **OpenShift:** A Kubernetes-based container orchestration platform by Red Hat with enterprise features.

6. Monitoring and Logging Tools

These provide visibility into system performance and help in troubleshooting.

- **Prometheus:** Open-source monitoring and alerting toolkit, often used with Kubernetes.
- **Grafana:** Visualization tool that integrates with Prometheus and other data sources to display real-time metrics.
- **ELK Stack (Elasticsearch, Logstash, Kibana):** Combination for log aggregation, searching, and visualization.
- **Datadog:** Cloud-native monitoring platform that provides metrics, logs, and traces in one place.
- **Splunk:** Robust tool for log analysis and real-time data insights.

7. Infrastructure as Code (IaC) Tools

These manage infrastructure through code rather than manual processes.

- **Terraform:** Open-source IaC tool by HashiCorp that defines infrastructure using configuration files.

- **AWS CloudFormation:** AWS-specific tool for provisioning infrastructure as code using JSON or YAML templates.

8. Cloud Platforms

They provide scalable infrastructure and services for hosting applications.

- **AWS (Amazon Web Services):** Leading cloud provider offering a wide range of DevOps-friendly services.
- **Microsoft Azure:** Cloud platform with extensive tools and integrations for DevOps workflows.
- **Google Cloud Platform (GCP):** Offers container-native solutions and integrations with Kubernetes and open-source tools.

DEVOPS PRINCIPLES

1. **Collaboration and Communication**
    - DevOps breaks down silos between development, operations, QA, and other teams.
    - Encourages shared responsibility and transparency throughout the software lifecycle.
    - Enhances problem-solving and aligns teams with business goals.
2. **Automation**
    - Automates repetitive tasks like builds, testing, deployment, and infrastructure provisioning.
    - Reduces human error, increases speed, and ensures consistency.
    - Enables CI/CD workflows effectively.
3. **Continuous Integration (CI)**
    - Developers frequently merge code into a shared repository.

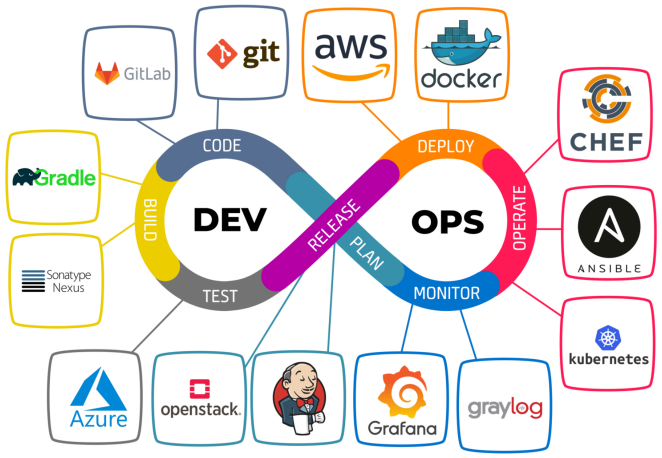
- Automated builds and tests verify each change to detect errors early. Improves software quality and reduces integration problems.

4. **Continuous Delivery/Deployment (CD)**
    - Code changes are automatically deployed to staging or production after successful integration.
    - Continuous Delivery ensures code is always in a deployable state.

    - Continuous Deployment automates production deployment without manual intervention.
5. **Infrastructure as Code (IaC)**
    - Infrastructure and configurations are managed via machine-readable definition files.
    - Ensures consistency, scalability, and repeatability.
    - Infrastructure changes can be version-controlled and audited like application code.
6. **Monitoring and Feedback**
    - Continuous monitoring provides real-time insights into performance and issues.
    - Feedback loops enable quick responses and improvements.
    - Supports proactive incident management and capacity planning.
7. **Security Integration (DevSecOps)**
    - Integrates security at every stage of the DevOps pipeline.
    - Automated security tests and compliance checks detect vulnerabilities early.
    - Ensures faster delivery without compromising security.

DEVOPS ENGINEER ROLE AND RESPONSIBILITIES

Area	Responsibility
CI/CD Implementation	Design and maintain automated build, test, and deployment pipelines.
Infrastructure Management	Provision and manage infrastructure using IaC tools (e.g., Terraform, CloudFormation).
Automation	Automate operational tasks like deployments, monitoring, and backups.
Monitoring and Logging	Set up tools to monitor performance and detect issues (e.g., Prometheus, ELK Stack).
Collaboration	Work closely with developers, QA, and operations teams to ensure smooth workflows.
Security	Implement security best practices across code, infrastructure, and deployment.
Cloud Services Management	Manage cloud environments (AWS, Azure, GCP) for scalability and availability.



CONCLUSION: This experiment provides a comprehensive understanding of DevOps, covering its principles, key practices, tools, and the roles and responsibilities of a DevOps engineer, highlighting how collaboration, automation, and continuous integration streamline software development and delivery.

LO MAPPING: LO1

ASSIGNMENT- 2

(Git Installation & Github Account)

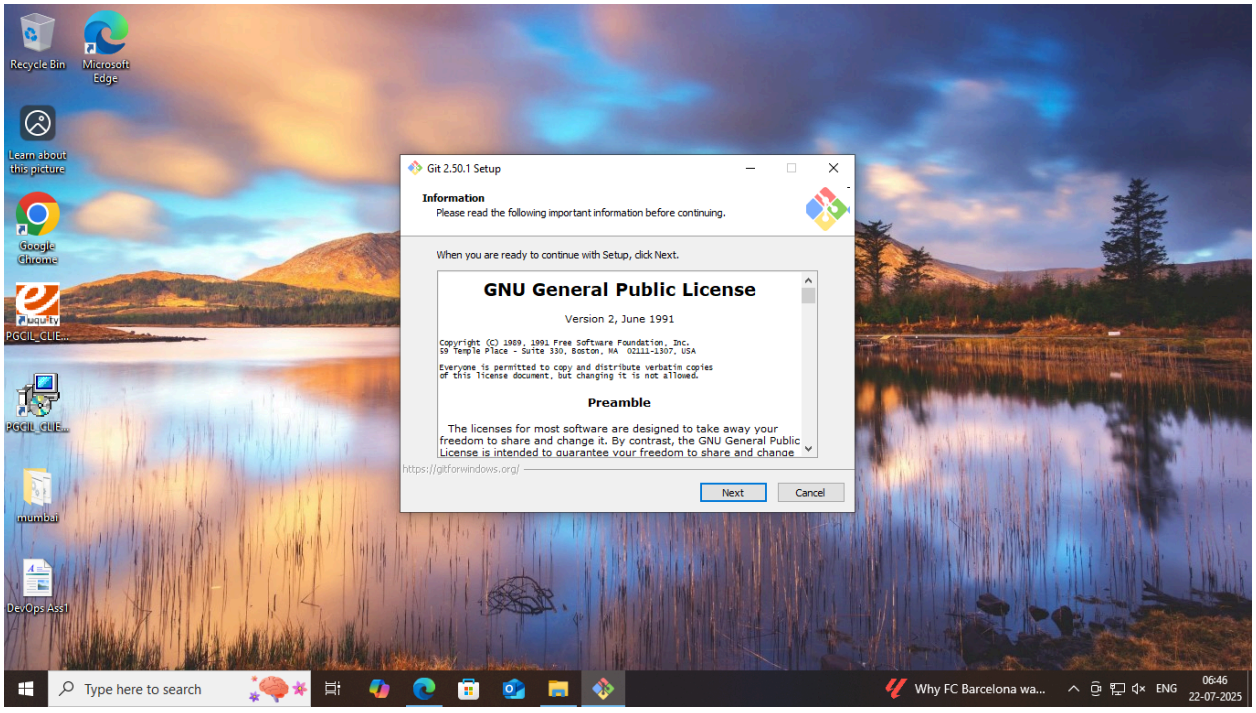
AIM: To understand Version Control System, Git installation & Github account.

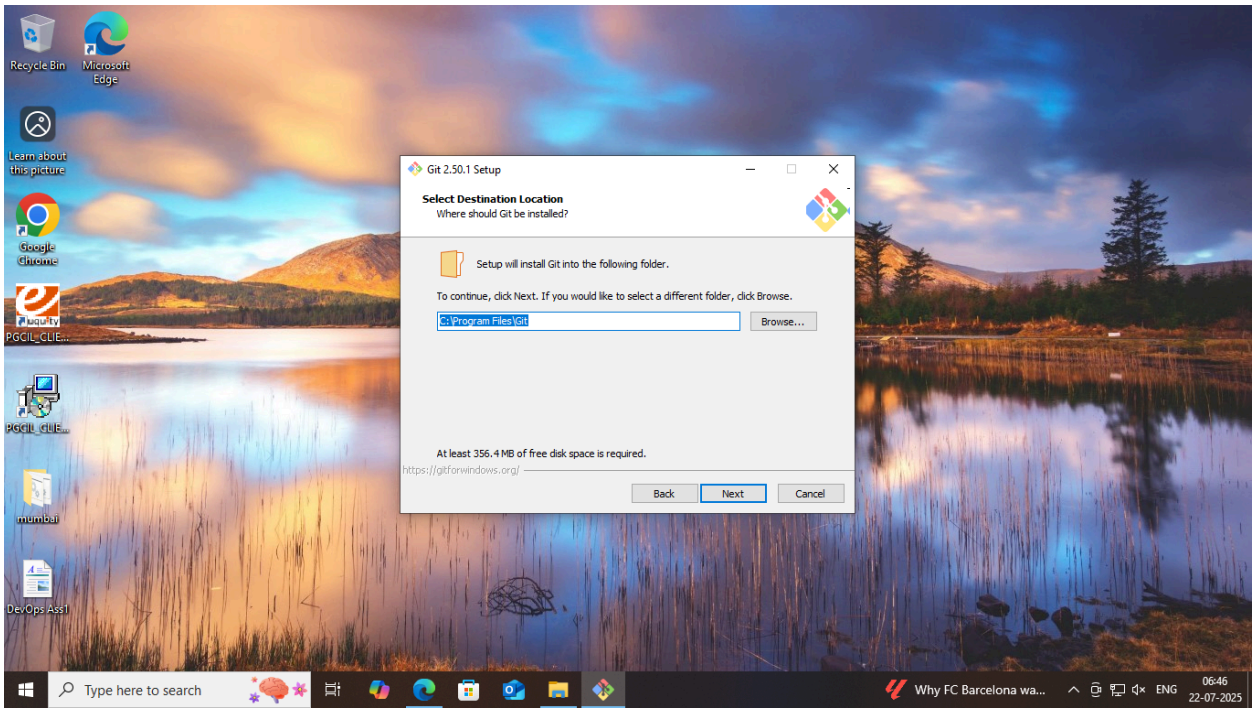
THEORY:

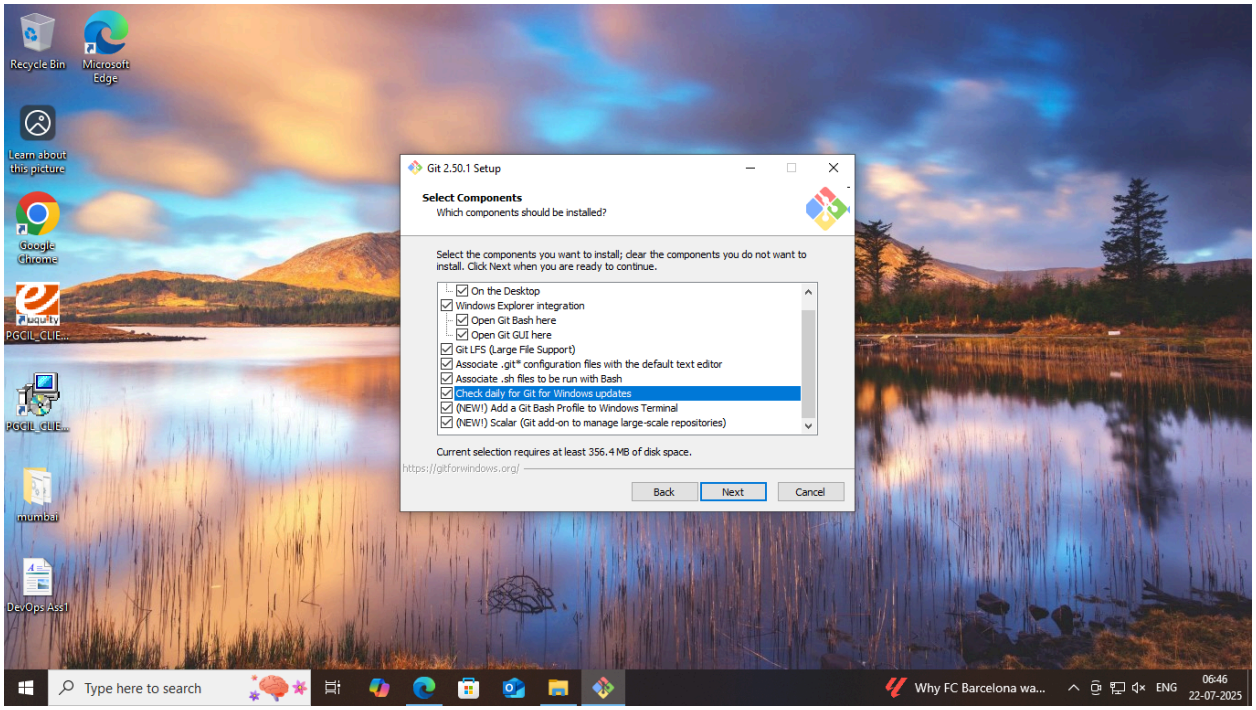
- **Version Control System (VCS)** is a tool that helps track and manage changes to files or code over time. It allows multiple people to work on a project without overwriting each other's work, supports reverting to previous versions, and maintains a history of changes.
- **Git** is a popular distributed VCS that stores the complete history of a project on every user's local machine, enabling offline work and efficient collaboration. Installing Git on a system provides the necessary command-line tools to create repositories, track changes, and manage branches.
- **GitHub** is a cloud-based hosting platform for Git repositories, enabling remote collaboration, code sharing, and integration with various development tools. Creating a GitHub account allows users to store repositories online, contribute to open-source projects, and manage collaborative workflows.

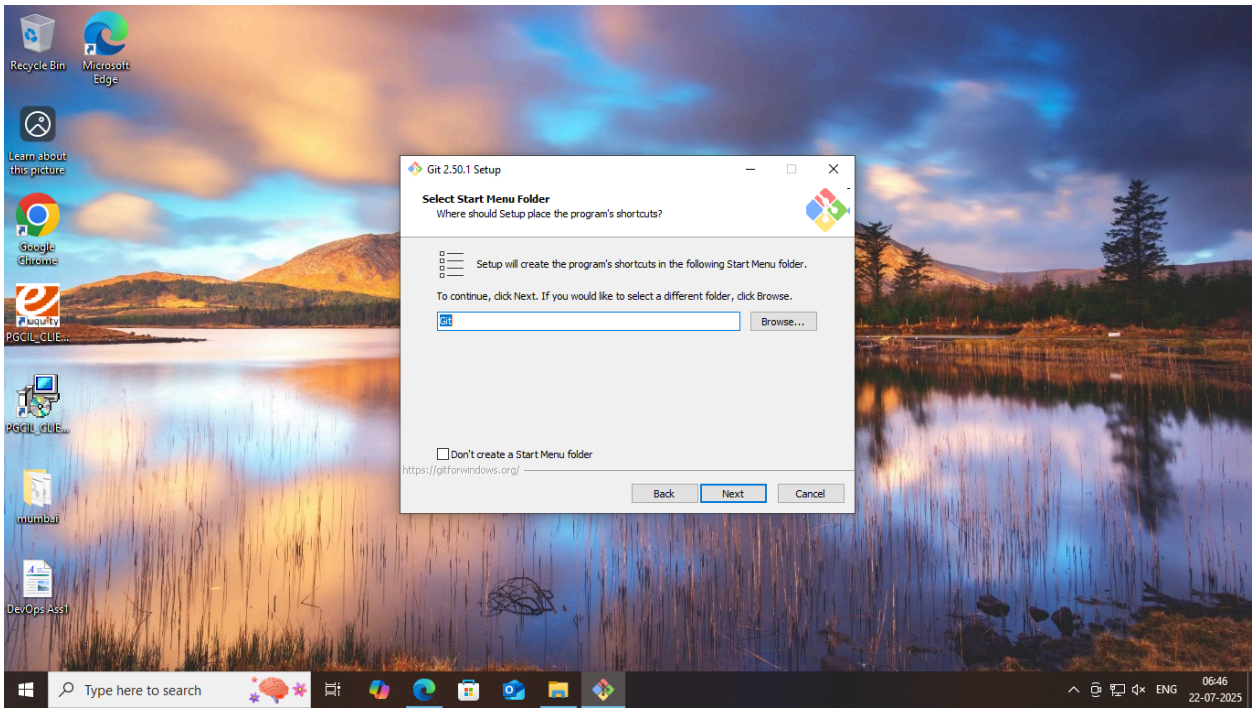
SCREENSHOTS:

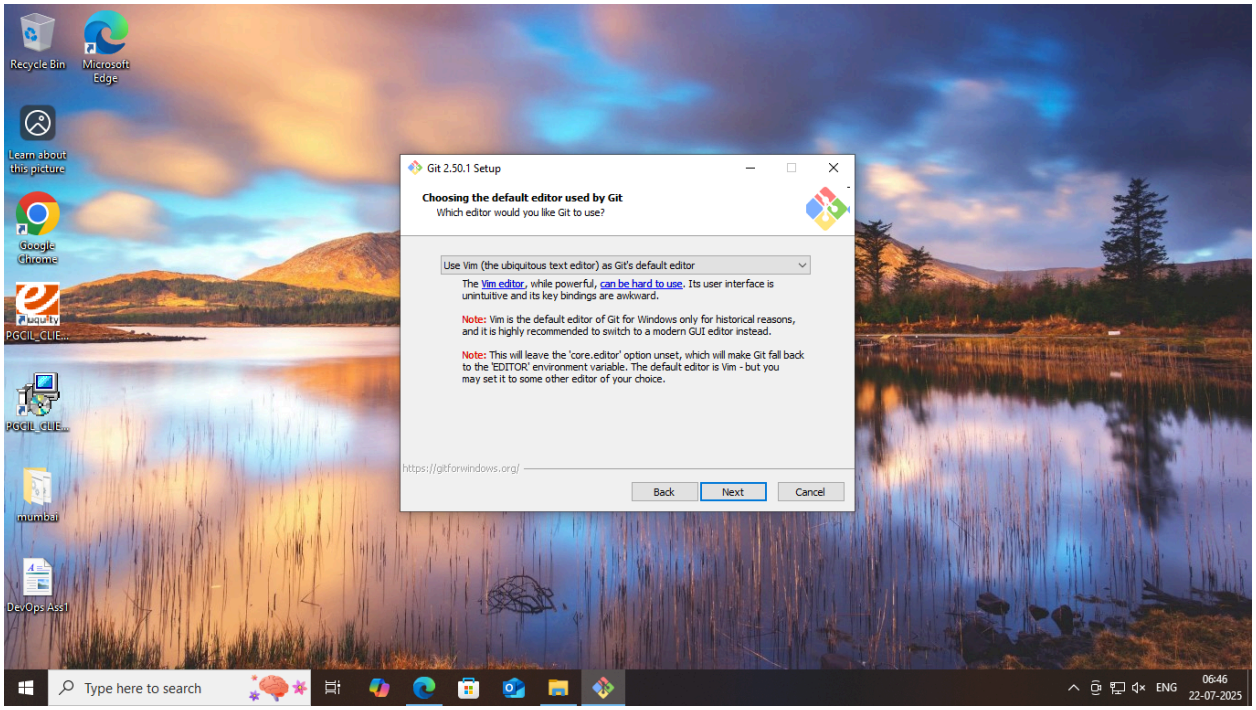
- **Git Installation**

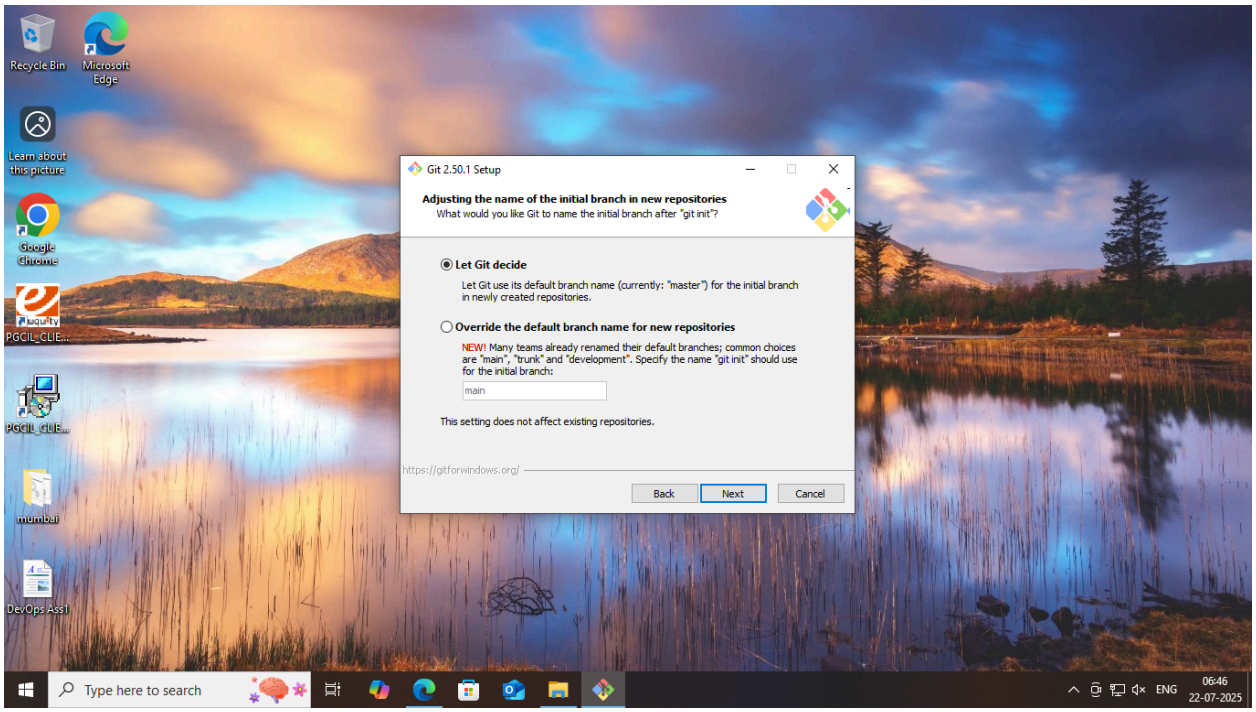


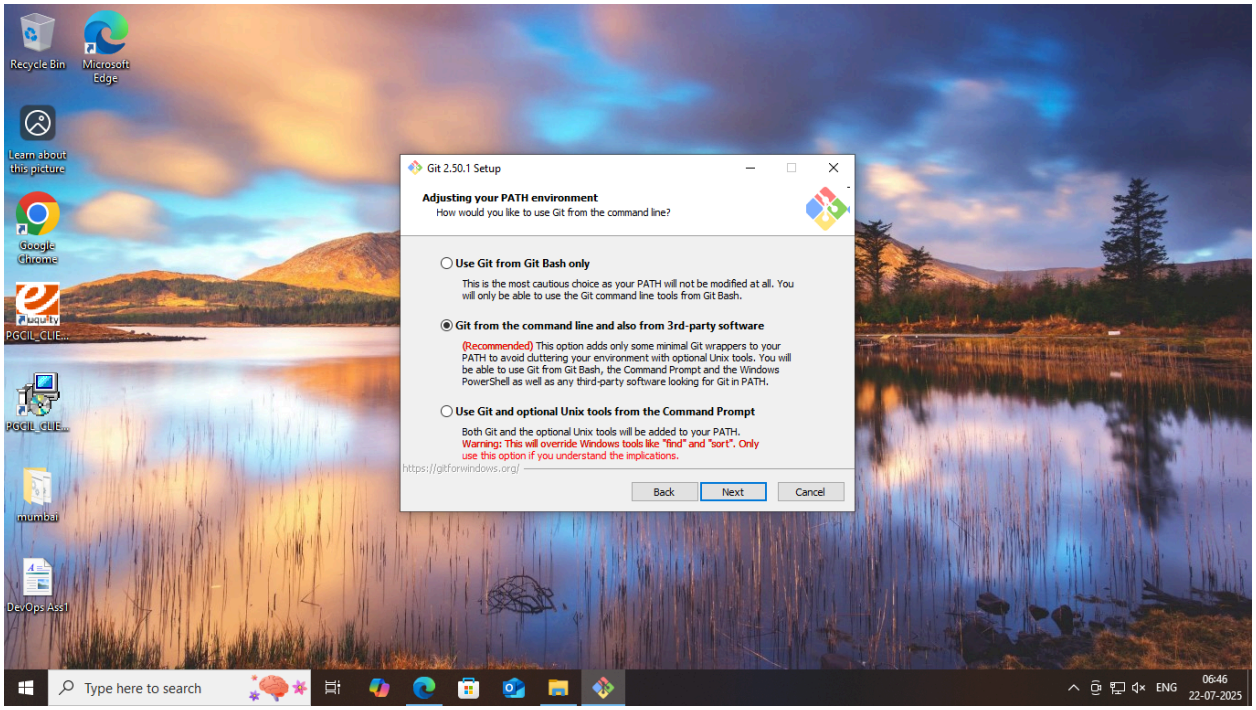


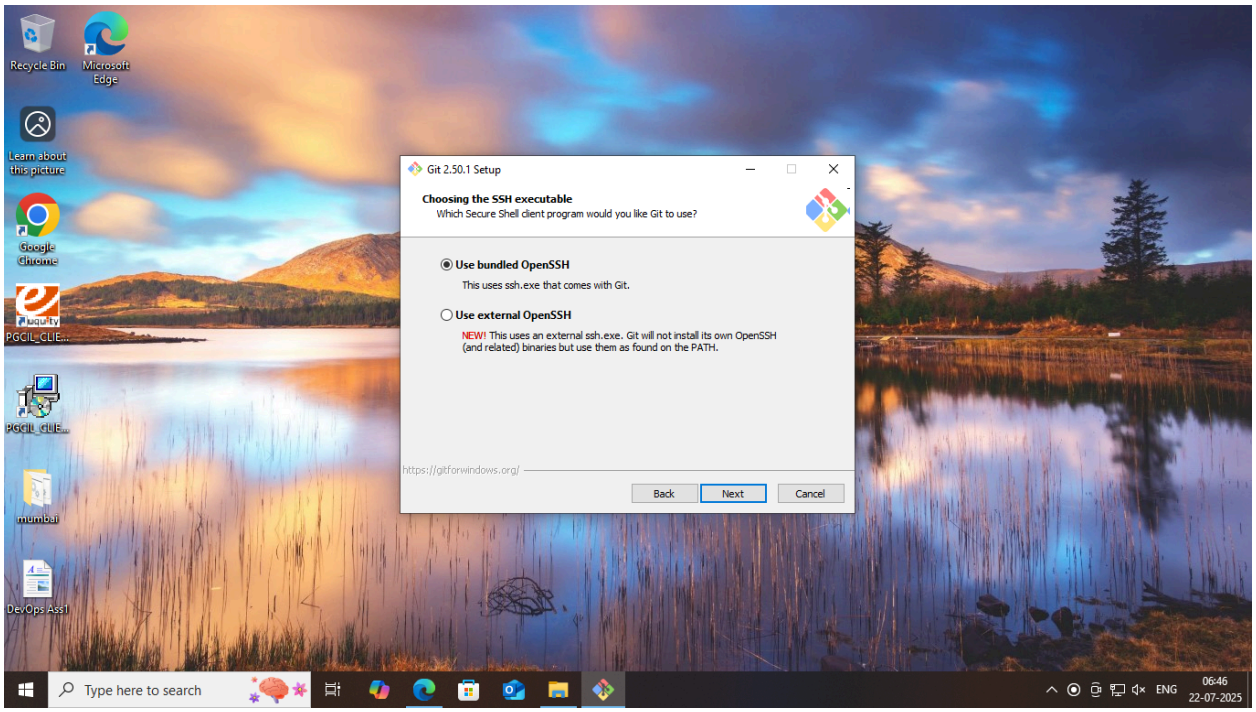

Recycle Bin
Microsoft Edge
Learn about this picture
Google Chrome
PGCIL_CLIE...
PGCIL_CLIE...
mumbai
DevOps Ass1
Git 2.50.1 Setup
Select Components
Which components should be installed?
Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.
On the Desktop
Windows Explorer integration
Open Git Bash here
Open Git GUI here
Git LFS (Large File Support)
Associate .git* configuration files with the default text editor
Associate .sh files to be run with Bash
Check daily for Git for Windows updates
(NEW!) Add a Git Bash Profile to Windows Terminal
(NEW!) Scalar (Git add-on to manage large-scale repositories)
Current selection requires at least 356.4 MB of disk space.
https://gitforwindows.org/
Back
Next
Cancel
Type here to search
Why FC Barcelona wa...
ENG
06:46
22-07-2025

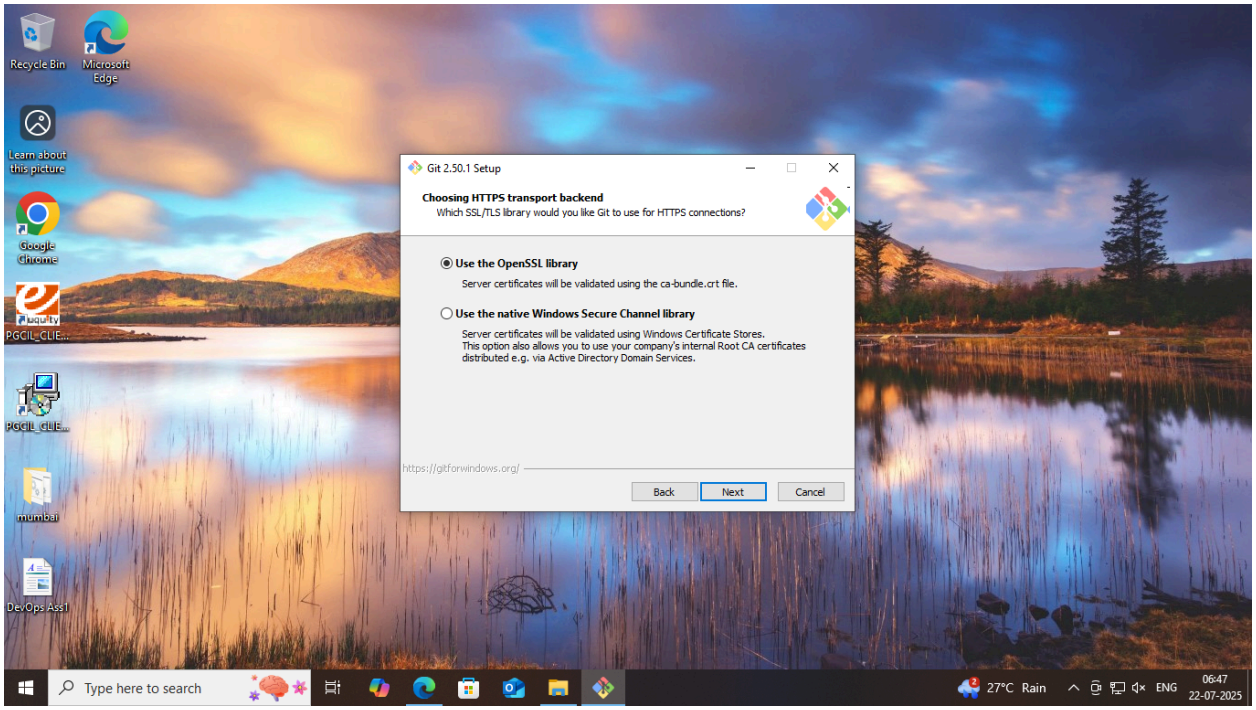

Recycle Bin
Microsoft Edge
Learn about this picture
Google Chrome
PGCIL_CLIE...
PGCIL_CLIE...
mumbai
DevOps Ass1
Git 2.50.1 Setup
Select Start Menu Folder
Where should Setup place the program's shortcuts?
Setup will create the program's shortcuts in the following Start Menu folder.
To continue, click Next. If you would like to select a different folder, click Browse.
Git
Browse...
Don't create a Start Menu folder
https://gitforwindows.org/
Back
Next
Cancel
Type here to search
ENG
06:46
22-07-2025

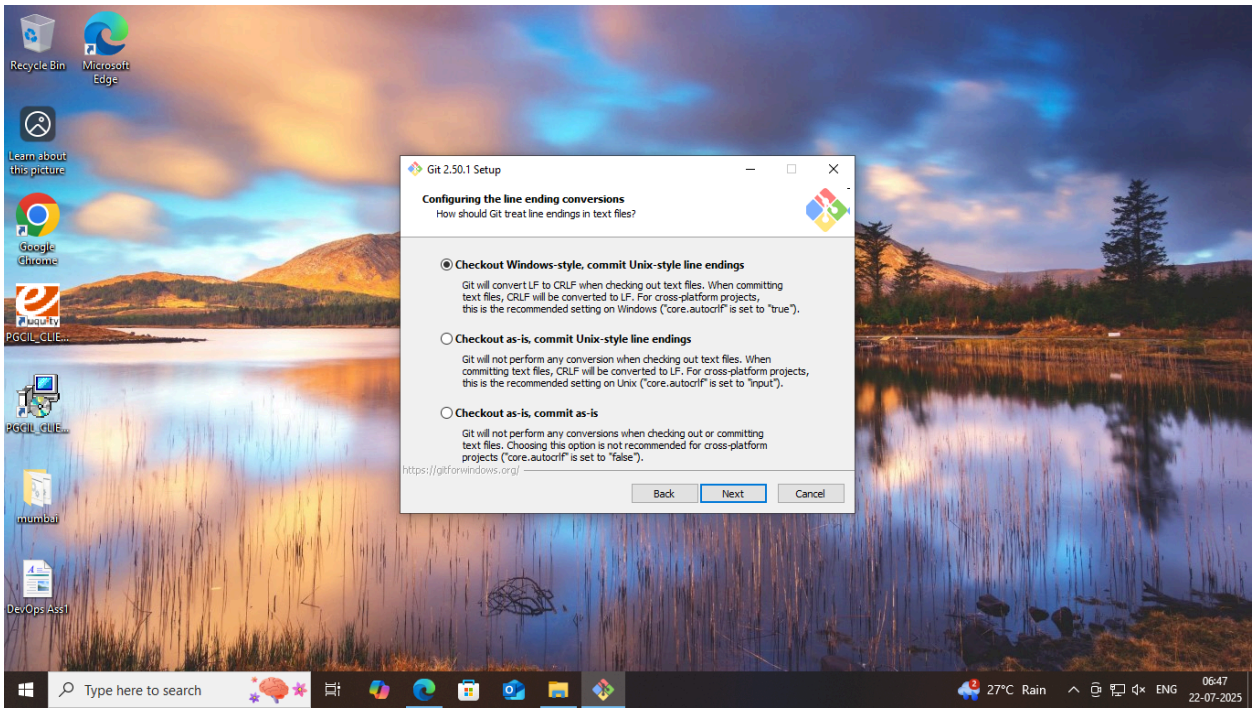

Recycle Bin
Microsoft Edge
Learn about this picture
Google Chrome
PGCIL_CLIE...
PGCIL_CLIE...
mumbai
DevOps Ass1
Git 2.50.1 Setup
Choosing the default editor used by Git
Which editor would you like Git to use?
Use Vim (the ubiquitous text editor) as Git's default editor
The Vim editor, while powerful, can be hard to use. Its user interface is unintuitive and its key bindings are awkward.
Note: Vim is the default editor of Git for Windows only for historical reasons, and it is highly recommended to switch to a modern GUI editor instead.
Note: This will leave the 'core.editor' option unset, which will make Git fall back to the 'EDITOR' environment variable. The default editor is Vim - but you may set it to some other editor of your choice.
https://gitforwindows.org/
Back
Next
Cancel
Type here to search
ENG
06:46
22-07-2025

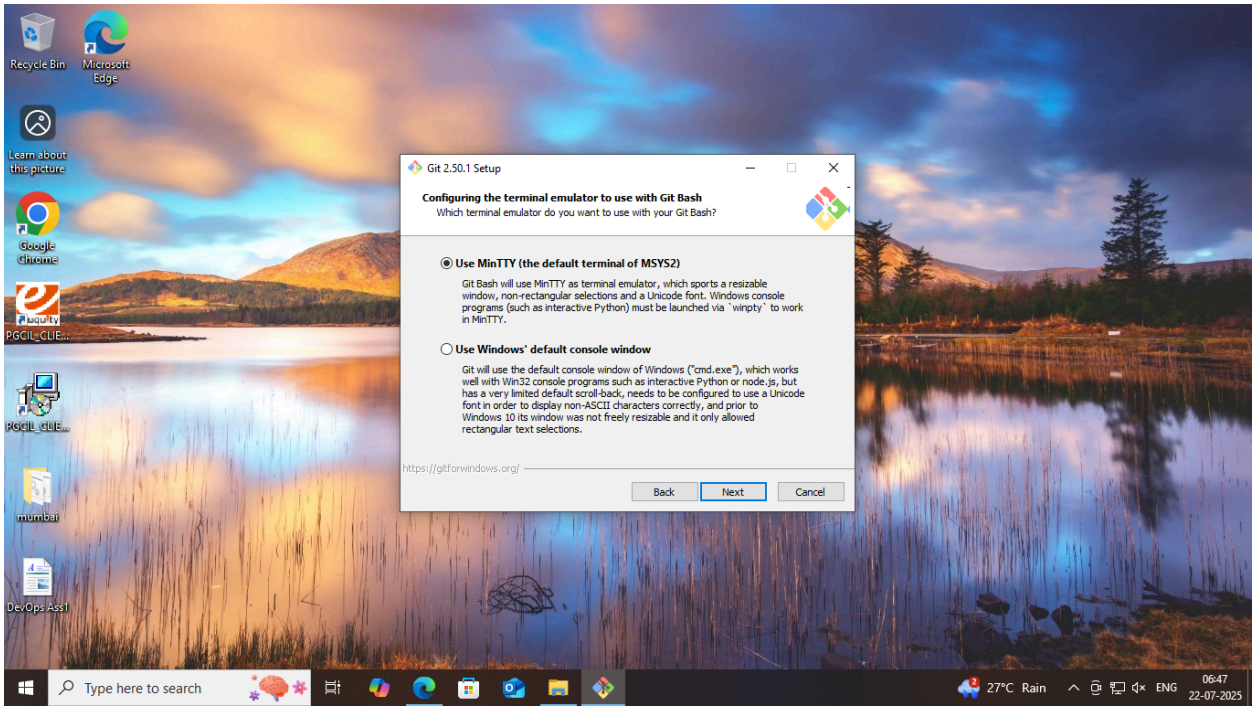

Recycle Bin
Microsoft Edge
Learn about this picture
Google Chrome
PGCIL_CLIE...
PGCIL_CLIE...
mumbai
DevOps Ass1
Git 2.50.1 Setup
Adjusting the name of the initial branch in new repositories
What would you like Git to name the initial branch after "git init"?
Let Git decide
Let Git use its default branch name (currently: "master") for the initial branch in newly created repositories.
Override the default branch name for new repositories
NEW! Many teams already renamed their default branches; common choices are "main", "trunk" and "development". Specify the name "git init" should use for the initial branch:
main
This setting does not affect existing repositories.
https://gitforwindows.org/
Back
Next
Cancel
Type here to search
ENG
06:46
22-07-2025

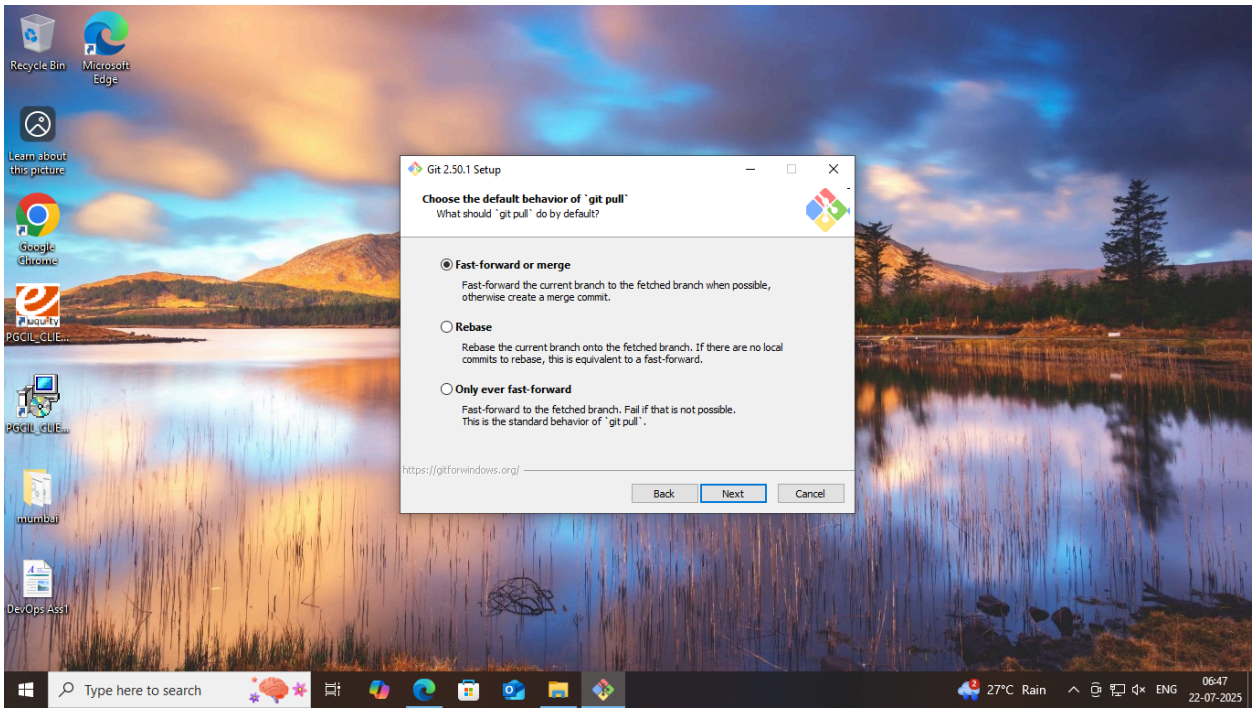

Recycle Bin
Microsoft Edge
Learn about this picture
Google Chrome
PGCIL_CLIE...
PGCIL_CLIE...
mumbai
DevOps Ass1
Git 2.50.1 Setup
Adjusting your PATH environment
How would you like to use Git from the command line?
Use Git from Git Bash only
This is the most cautious choice as your PATH will not be modified at all. You will only be able to use the Git command line tools from Git Bash.
Git from the command line and also from 3rd-party software
(Recommended) This option adds only some minimal Git wrappers to your PATH to avoid cluttering your environment with optional Unix tools. You will be able to use Git from Git Bash, the Command Prompt and the Windows PowerShell as well as any third-party software looking for Git in PATH.
Use Git and optional Unix tools from the Command Prompt
Both Git and the optional Unix tools will be added to your PATH.
Warning: This will override Windows tools like "find" and "sort". Only use this option if you understand the implications.
https://gitforwindows.org/
Back
Next
Cancel
Type here to search
ENG
06:46
22-07-2025

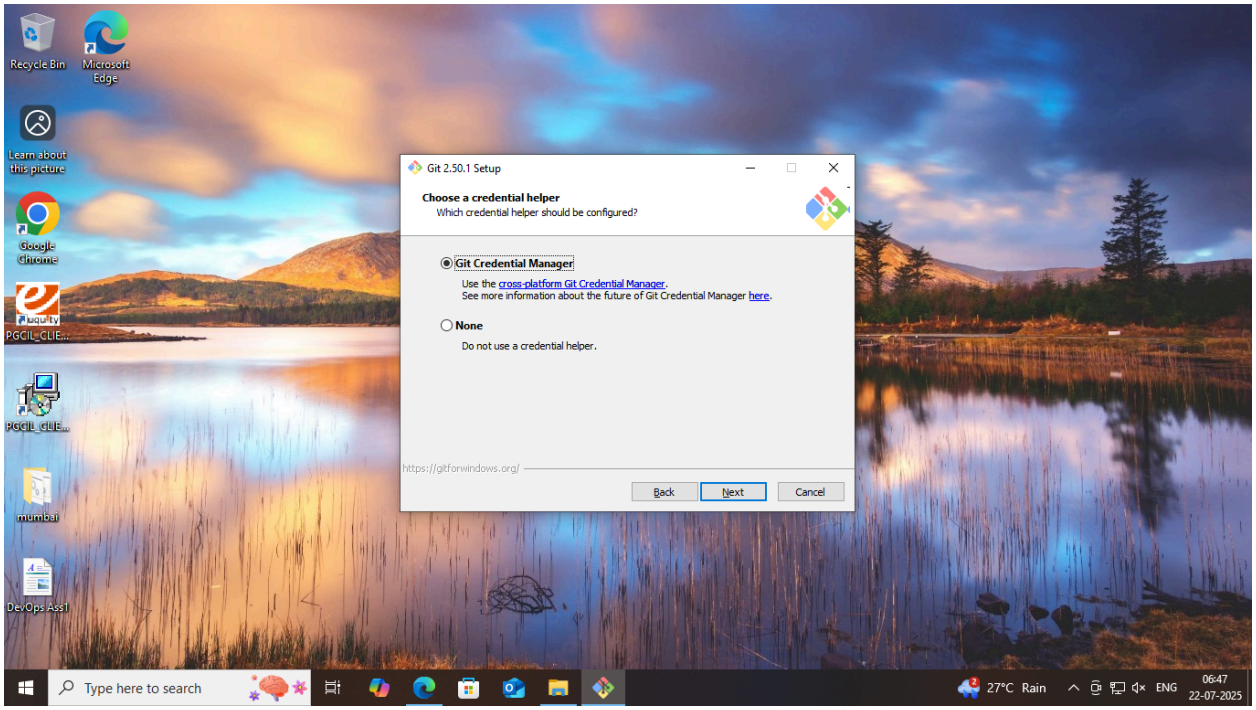

Recycle Bin
Microsoft Edge
Learn about this picture
Google Chrome
PGCIL_CLIE...
PGCIL_CLIE...
mumbai
DevOps Ass1
Git 2.50.1 Setup
Choosing the SSH executable
Which Secure Shell client program would you like Git to use?
Use bundled OpenSSH
This uses ssh.exe that comes with Git.
Use external OpenSSH
NEW! This uses an external ssh.exe. Git will not install its own OpenSSH (and related) binaries but use them as found on the PATH.
https://gitforwindows.org/
Back
Next
Cancel
Type here to search
ENG
06:46
22-07-2025

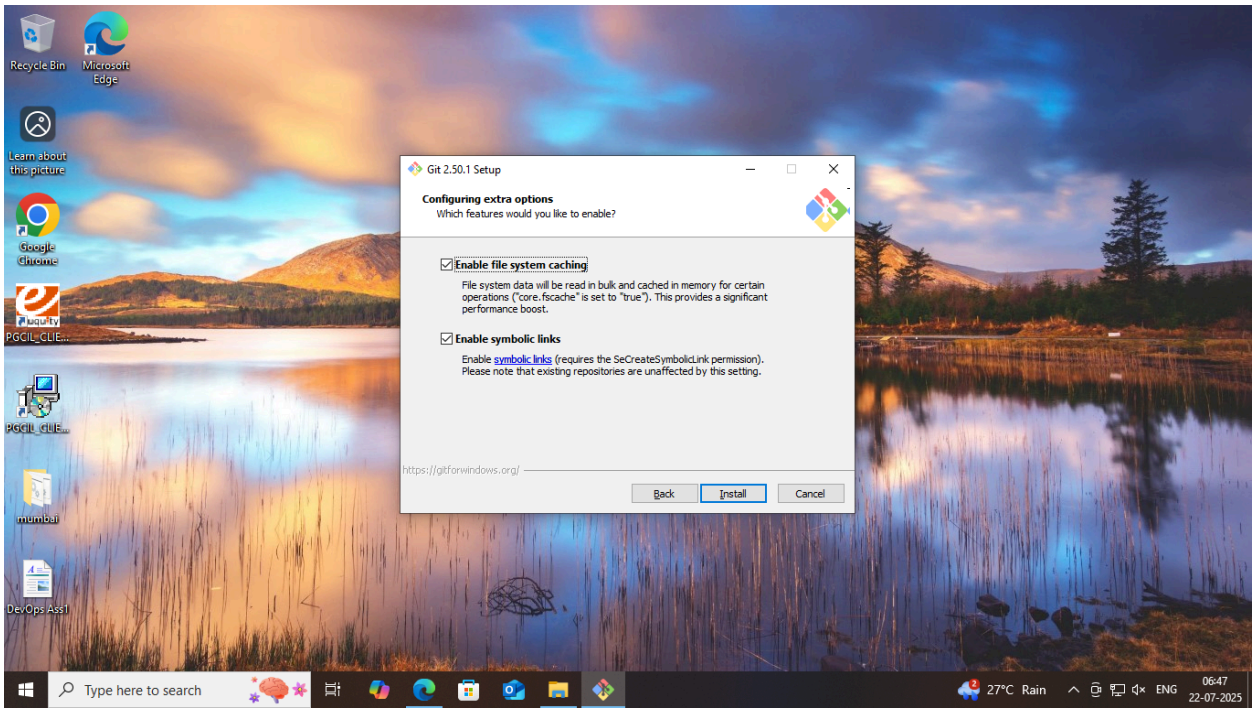

Recycle Bin
Microsoft Edge
Learn about this picture
Google Chrome
PGCIL_CLIE...
PGCIL_CLIE...
mumbai
DevOps Ass1
Git 2.50.1 Setup
Choosing HTTPS transport backend
Which SSL/TLS library would you like Git to use for HTTPS connections?
Use the OpenSSL library
Server certificates will be validated using the ca-bundle.crt file.
Use the native Windows Secure Channel library
Server certificates will be validated using Windows Certificate Stores. This option also allows you to use your company's internal Root CA certificates distributed e.g. via Active Directory Domain Services.
https://gitforwindows.org/
Back
Next
Cancel
Type here to search
27°C Rain
ENG
06:47
22-07-2025

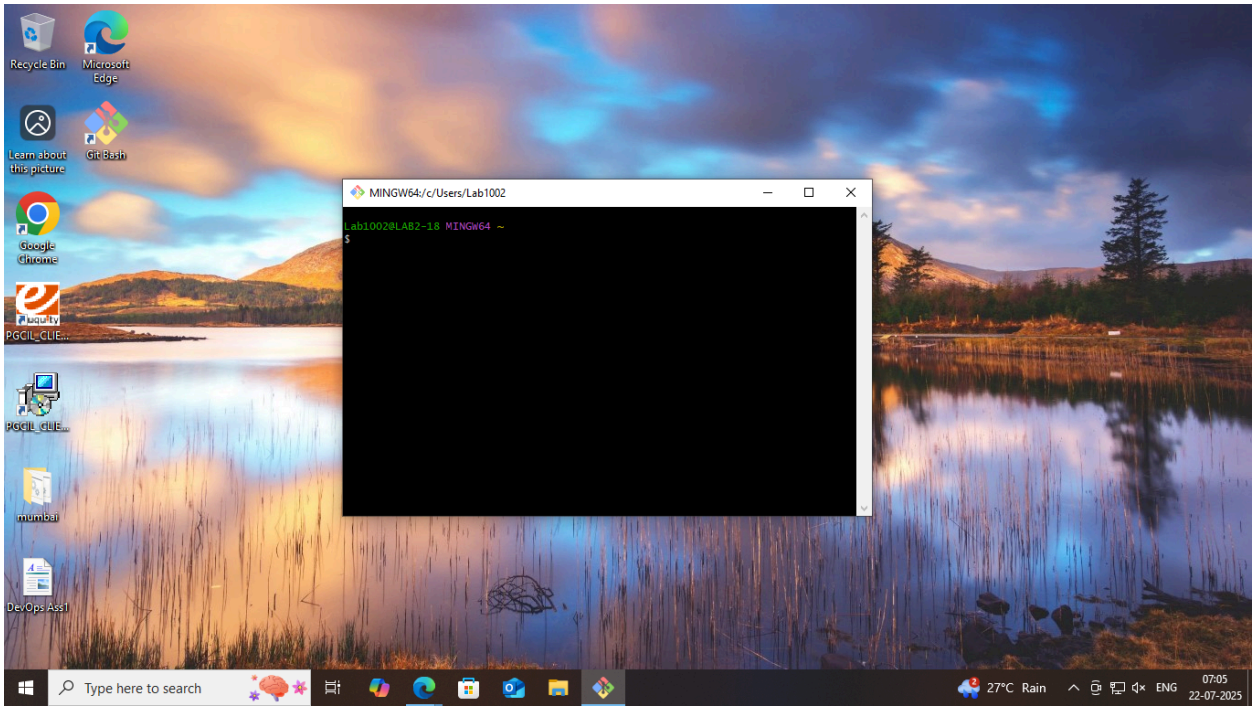

Recycle Bin
Microsoft Edge
Learn about this picture
Google Chrome
PGCIL_CLIE...
PGCIL_CLIE...
mumbai
DevOps Ass1
Git 2.50.1 Setup
Configuring the line ending conversions
How should Git treat line endings in text files?
Checkout Windows-style, commit Unix-style line endings
Git will convert LF to CRLF when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, this is the recommended setting on Windows ("core.autocrlf" is set to "true").
Checkout as-is, commit Unix-style line endings
Git will not perform any conversion when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, this is the recommended setting on Unix ("core.autocrlf" is set to "input").
Checkout as-is, commit as-is
Git will not perform any conversions when checking out or committing text files. Choosing this option is not recommended for cross-platform projects ("core.autocrlf" is set to "false").
https://gitforwindows.org/
Back
Next
Cancel
Type here to search
27°C Rain
ENG
06:47
22-07-2025


Recycle Bin
Microsoft Edge
Learn about this picture
Google Chrome
PGCIL_CLIE...
PGCIL_CLIE...
mumbai
DevOps Ass1
Git 2.50.1 Setup
Configuring the terminal emulator to use with Git Bash
Which terminal emulator do you want to use with your Git Bash?
Use MinTTY (the default terminal of MSYS2)
Git Bash will use MinTTY as terminal emulator, which sports a resizable window, non-rectangular selections and a Unicode font. Windows console programs (such as interactive Python) must be launched via `winpty` to work in MinTTY.
Use Windows' default console window
Git will use the default console window of Windows ("cmd.exe"), which works well with Win32 console programs such as interactive Python or node.js, but has a very limited default scroll-back, needs to be configured to use a Unicode font in order to display non-ASCII characters correctly, and prior to Windows 10 its window was not freely resizable and it only allowed rectangular text selections.
https://gitforwindows.org/
Back
Next
Cancel
Type here to search
27°C Rain
ENG
06:47
22-07-2025

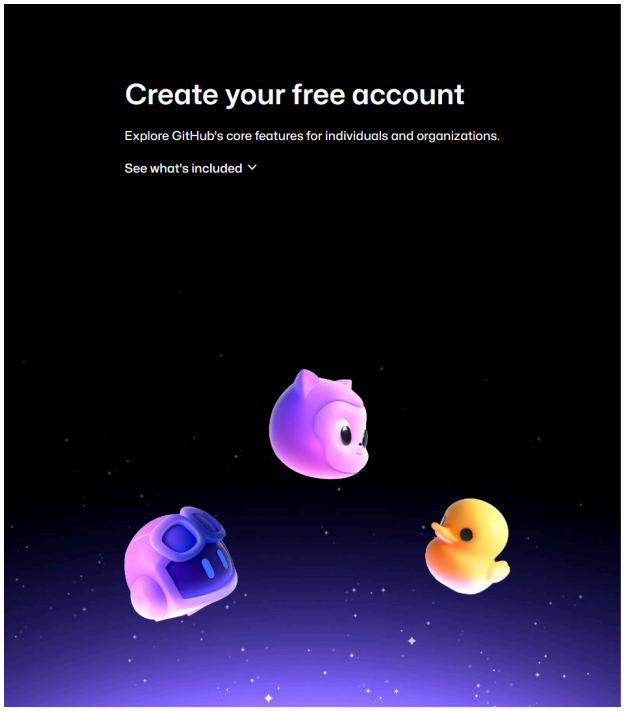

Recycle Bin
Microsoft Edge
Learn about this picture
Google Chrome
PGCIL_CLIE...
PGCIL_CLIE...
mumbai
DevOps Ass1
Git 2.50.1 Setup
Choose the default behavior of `git pull`
What should `git pull` do by default?
Fast-forward or merge
Fast-forward the current branch to the fetched branch when possible, otherwise create a merge commit.
Rebase
Rebase the current branch onto the fetched branch. If there are no local commits to rebase, this is equivalent to a fast-forward.
Only ever fast-forward
Fast-forward to the fetched branch. Fail if that is not possible. This is the standard behavior of `git pull`.
https://gitforwindows.org/
Back
Next
Cancel
Type here to search
27°C Rain
ENG
06:47
22-07-2025

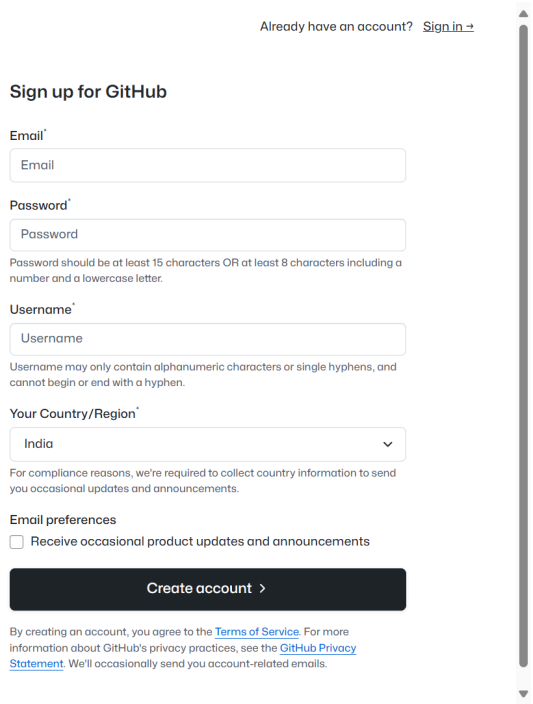

Recycle Bin
Microsoft Edge
Learn about this picture
Google Chrome
PGCIL_CLIE...
PGCIL_CLIE...
mumbai
DevOps Ass1
Git 2.50.1 Setup
Choose a credential helper
Which credential helper should be configured?
Git Credential Manager
Use the cross-platform Git Credential Manager.
See more information about the future of Git Credential Manager here.
None
Do not use a credential helper.
https://gitforwindows.org/
Back
Next
Cancel
Type here to search
27°C Rain
ENG
06:47
22-07-2025

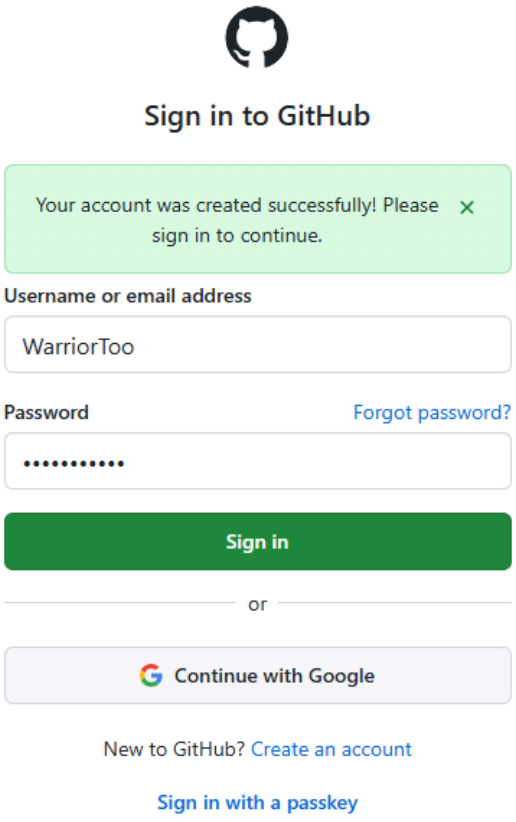

Recycle Bin
Microsoft Edge
Learn about this picture
Google Chrome
PGCIL_CLIE...
PGCIL_CLIE...
mumbai
DevOps Ass1
Git 2.50.1 Setup
Configuring extra options
Which features would you like to enable?
Enable file system caching
File system data will be read in bulk and cached in memory for certain operations ("core.fscache" is set to "true"). This provides a significant performance boost.
Enable symbolic links
Enable symbolic links (requires the SeCreateSymbolicLink permission).
Please note that existing repositories are unaffected by this setting.
https://gitforwindows.org/
Back
Install
Cancel
Type here to search
27°C Rain
ENG
06:47
22-07-2025

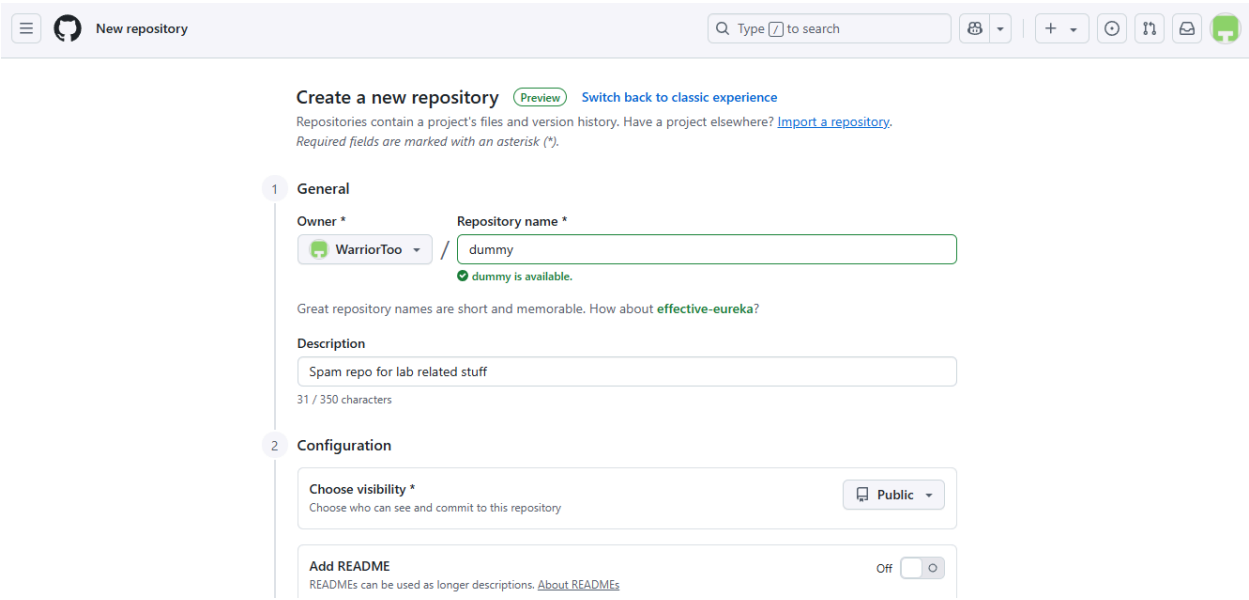

Recycle Bin
Microsoft Edge
Learn about this picture
Git Bash
Google Chrome
PGCIL_CLIE...
PGCIL_CLIE...
mumbai
DevOps Ass1
MINGW64:/c/Users/Lab1002
Lab1002@LAB2-18 MINGW64 ~
$
Type here to search
27°C Rain
ENG
07:05
22-07-2025

- **Github account creation & new repository setup**









CONCLUSION: This experiment successfully demonstrates the process of understanding Version Control Systems, installing Git, and creating a GitHub account, enabling efficient change tracking, seamless collaboration, and effective code management both locally and remotely.

LO MAPPING: LO1, LO2

ASSIGNMENT- 3

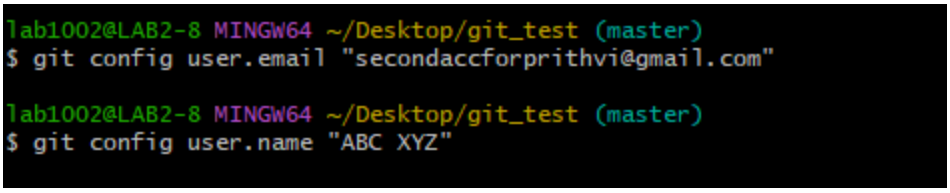
(Git Operations)

AIM: To perform various Git Operations.

THEORY: Git is a widely used distributed version control system that helps developers track changes in source code during software development. It allows multiple users to work on the same project simultaneously without interfering with each other's work. Git stores project history efficiently, enabling developers to revert to previous versions, compare changes, and collaborate effectively.

Common Git operations include creating repositories, cloning existing ones, staging and committing changes, branching, merging, and pushing or pulling code from remote repositories like GitHub or GitLab. Branching allows developers to work on new features or fixes independently, while merging integrates those changes back into the main codebase. These operations form the backbone of collaborative software development, ensuring consistency, traceability, and smooth integration of code from different contributors.

COMMANDS/SCREENSHOTS:



git config user.name: Set a name that is identifiable for credit

git config user.email: Set an email address to be associated with

```
lab1002@LAB2-8 MINGW64 ~/Desktop/git_test (master)
$ git config --global color.ui auto
```

git config --global color.ui auto: Set automatic command line for Git for easy reviewing

```
lab1002@LAB2-8 MINGW64 ~/Desktop (master)
$ cd git_test

lab1002@LAB2-8 MINGW64 ~/Desktop/git_test (master)
$ git init
Initialized empty Git repository in C:/Users/lab1002/Desktop/git_test/.git/

lab1002@LAB2-8 MINGW64 ~/Desktop/git_test (master)
$ git clone https://github.com/Zzz-0512/Trial.git
Cloning into 'Trial'...
warning: You appear to have cloned an empty repository.

lab1002@LAB2-8 MINGW64 ~/Desktop/git_test (master)
$ git add SimpleCalculator.java

lab1002@LAB2-8 MINGW64 ~/Desktop/git_test (master)
$ git status
On branch master

No commits yet
```

git init: Initialise existing directory as Git repository

git clone: Retrive an entire repository from hosted location using URL

git add: Add a file as it looks to next commit

git status: Show modified files in directory

```
lab1002@LAB2-8 MINGW64 ~/Desktop/git_test (master)
$ git commit -m "Adding Java file"
[master (root-commit) 28eac0f] Adding Java file
 1 file changed, 47 insertions(+)
 create mode 100644 SimpleCalculator.java
```

git commit: Commit staged content as new commit snapshot

```
lab1002@LAB2-8 MINGW64 ~/Desktop/git_test (master)
$ git diff

lab1002@LAB2-8 MINGW64 ~/Desktop/git_test (master)
$ git diff --staged
```

git diff: diff of what is changed but not staged

git diff --staged: diff of what is staged but not committed

```
lab1002@LAB2-8 MINGW64 ~/Desktop/git_test (master)
$ git branch
* master

lab1002@LAB2-8 MINGW64 ~/Desktop/git_test (master)
$ git branch branch1
```

git branch: list all branches

git branch branchname: create new branch at current commit

```
lab1002@LAB2-8 MINGW64 ~/Desktop/git_test (master)
$ git branch
  branch1
* master

lab1002@LAB2-8 MINGW64 ~/Desktop/git_test (master)
$ git checkout branch1
Switched to branch 'branch1'

lab1002@LAB2-8 MINGW64 ~/Desktop/git_test (branch1)
$ git checkout master
Switched to branch 'master'

lab1002@LAB2-8 MINGW64 ~/Desktop/git_test (master)
$ git merge branch1
Already up to date.

lab1002@LAB2-8 MINGW64 ~/Desktop/git_test (master)
$ git log
commit 28eac0f2ea6e54239ff039f796d2a605123e5fbd (HEAD -> master, branch1)
Author: ABC XYZ <secondaccforprithvi@gmail.com>
Date:   Thu Jul 31 15:13:16 2025 +0530

    Adding Java file
```

git checkout: switch to another branch

git merge: merge branch into current one

git log: show all commits in current branch history

```
lab1002@LAB2-8 MINGW64 ~/Desktop/git_test (master)
$ git log --follow SimpleCalculator.java
commit 28eac0f2ea6e54239ff039f796d2a605123e5fbd (HEAD -> master, branch1)
Author: ABC XYZ <secondaccforprithvi@gmail.com>
Date:   Thu Jul 31 15:13:16 2025 +0530

    Adding Java file
```

git log --follow: show commits that changed file even across renames

```
lab1002@LAB2-8 MINGW64 ~/Desktop/git_test (master)
$ git rm SimpleCalculator.java
rm 'SimpleCalculator.java'

lab1002@LAB2-8 MINGW64 ~/Desktop/git_test (master)
$ git log --stat -M
commit 28eac0f2ea6e54239ff039f796d2a605123e5fbd (HEAD -> master, branch1)
Author: ABC XYZ <secondaccforprithvi@gmail.com>
Date:   Thu Jul 31 15:13:16 2025 +0530

    Adding Java file

 SimpleCalculator.java | 47 +++++++++++++++++++++++++++++++++++++++++++++++
 1 file changed, 47 insertions(+)
```

git rm: delete file from project and stage removol from commit

git log --stat -M: show all commit logs with indication of paths

```
lab1002@LAB2-8 MINGW64 ~/Desktop/git_test (master)
$ git stash
Saved working directory and index state WIP on master: 28eac0f Adding Java file

lab1002@LAB2-8 MINGW64 ~/Desktop/git_test (master)
$ git stash list
stash@{0}: WIP on master: 28eac0f Adding Java file

lab1002@LAB2-8 MINGW64 ~/Desktop/git_test (master)
$ git stash pop
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    SimpleCalculator.java

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

no changes added to commit (use "git add" and/or "git commit -a")
Dropped refs/stash@{0} (d22b3305556451ffdd682fdd25a4e71ce65f72ff)

lab1002@LAB2-8 MINGW64 ~/Desktop/git_test (master)
$ git stash drop
No stash entries found.
```

git stash: Save modified and staged changes

git stash list: List stage-order of staged file changed

git stash pop: write working from top of stash stack

git stash drop: discard the changes from top of the stack

CONCLUSION: This experiment successfully demonstrates various Git operations, providing a clear understanding of version control, collaboration, and code management in software development.

LO MAPPING: LO1, LO2

ASSIGNMENT- 4

(Jenkins Installation)

AIM: To understand continuous integration and Jenkins installation.

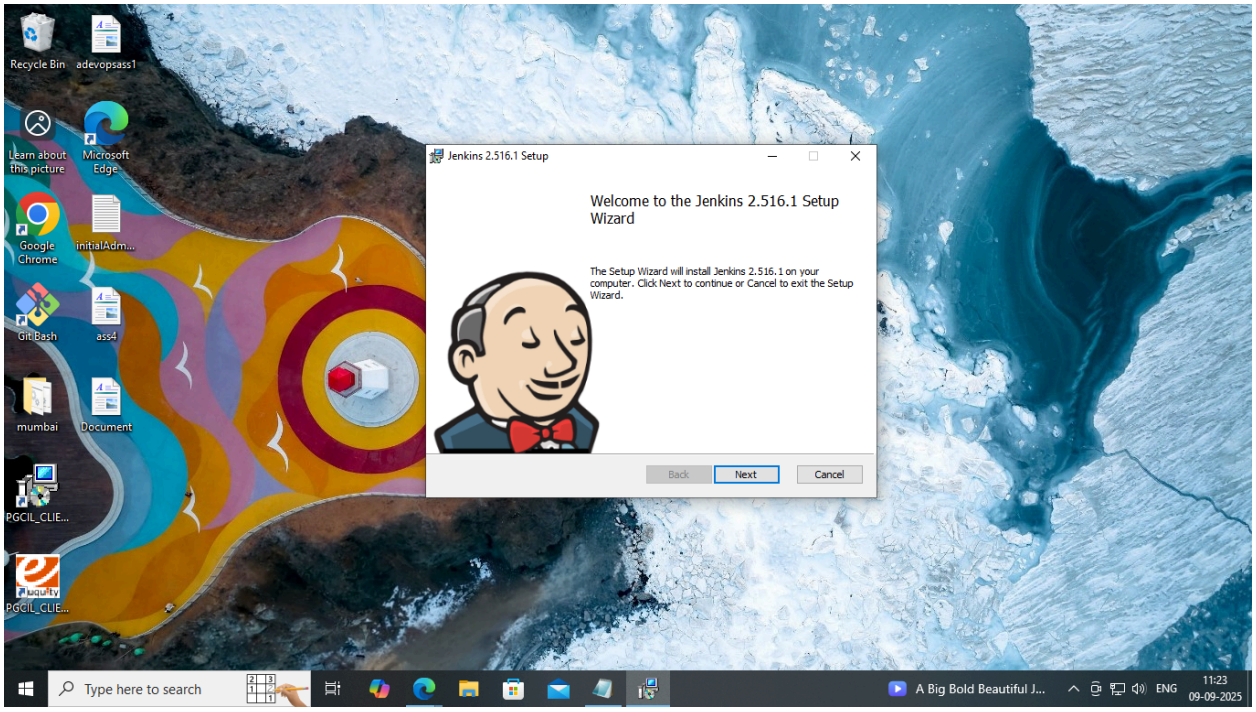
THEORY:

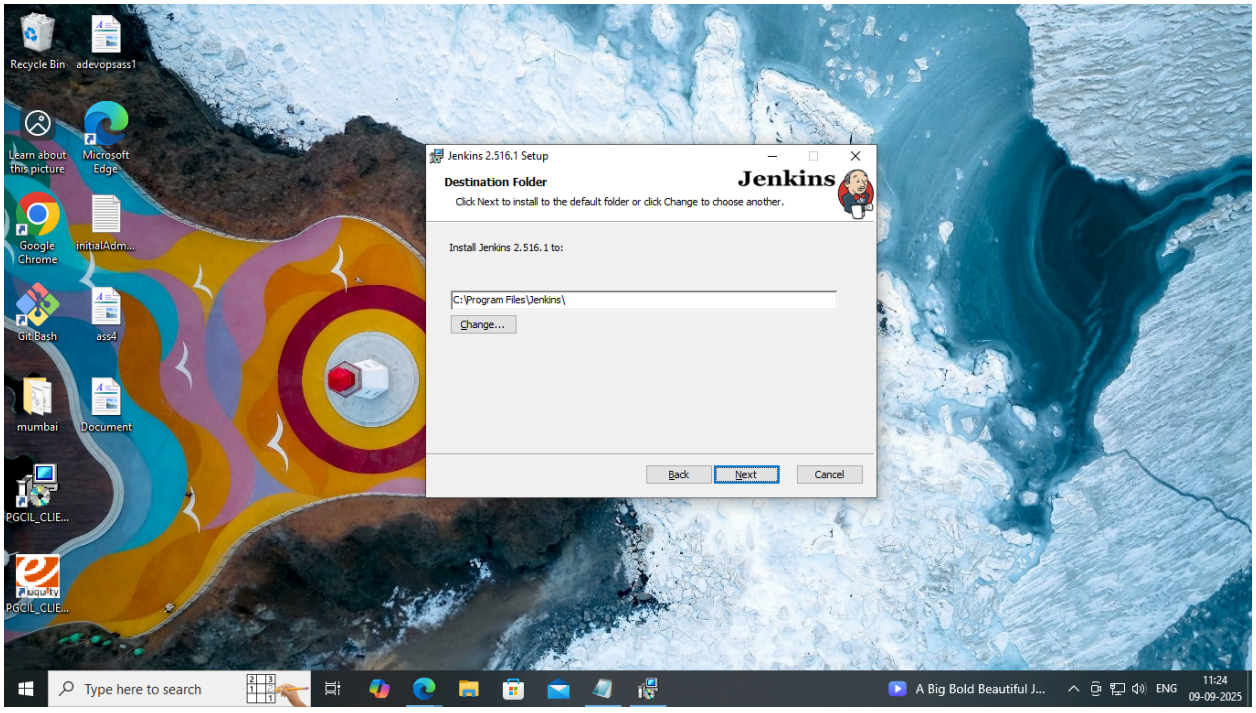
Continuous Integration (CI) is a key practice in modern software development that focuses on automatically integrating code changes from multiple developers into a shared repository several times a day. Each integration triggers an automated build and testing process to ensure that the new code does not break the existing functionality. This approach helps in identifying and fixing errors early, reduces integration issues, and promotes faster, more reliable software delivery. CI also encourages better collaboration among team members and ensures that the software is always in a deployable state.

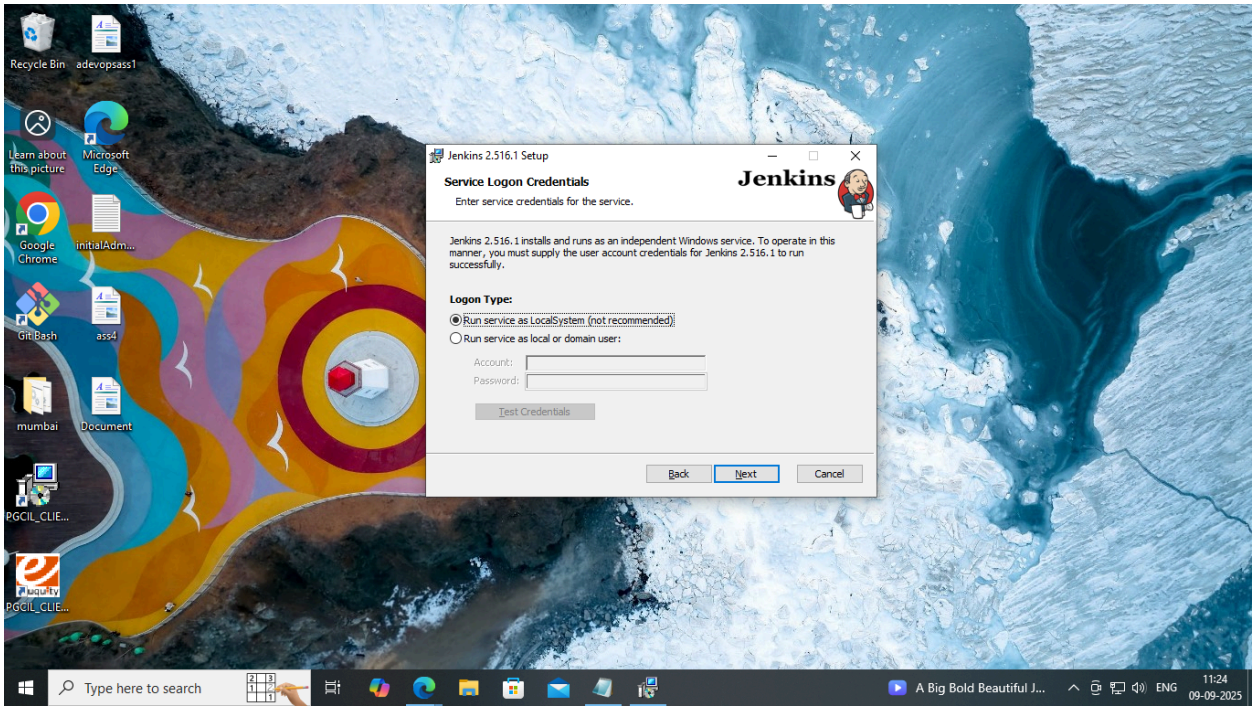
Jenkins is one of the most popular open-source tools used to implement Continuous Integration and Continuous Delivery (CI/CD). It automates the process of building, testing, and deploying applications, which saves time and minimizes manual effort. Jenkins supports a wide range of plugins that integrate with various tools like Git, Maven, Docker, and many others, making it highly flexible and customizable. The installation of Jenkins typically involves setting up the server (either locally or on the cloud), installing necessary plugins, and connecting it to a version control system. Once configured, Jenkins can automatically trigger builds and tests whenever changes are committed, ensuring smooth and efficient software development workflows.

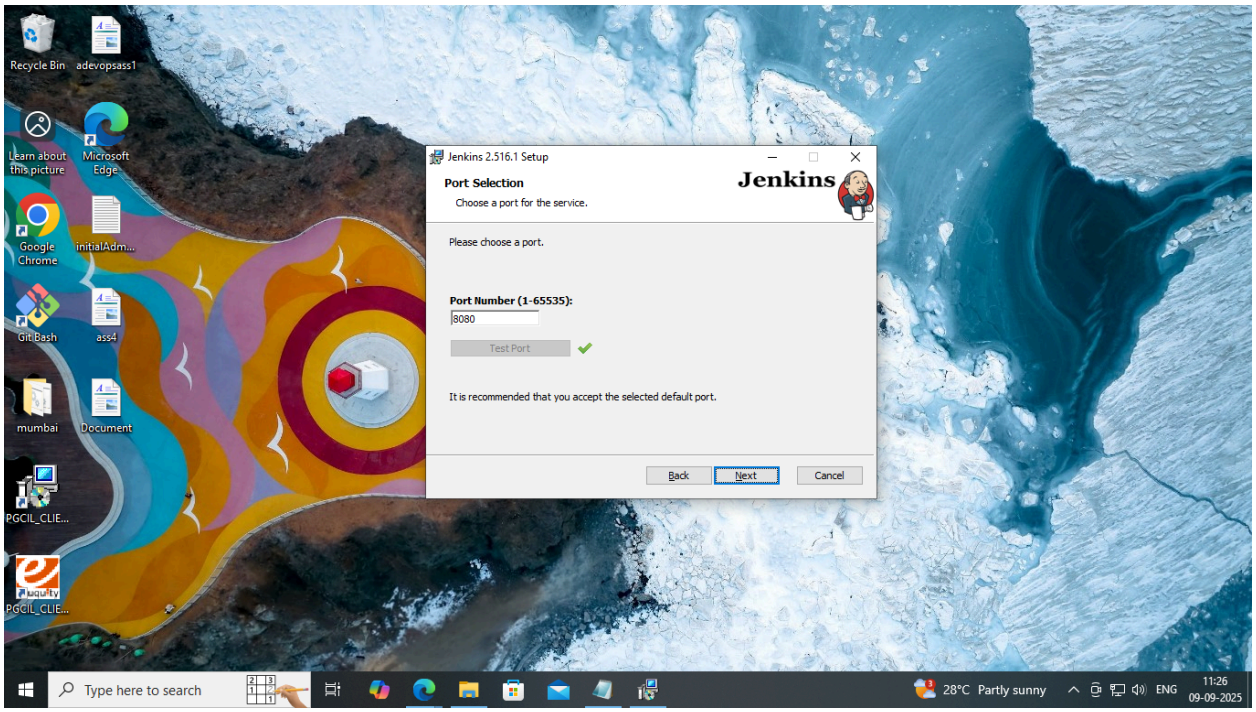
SCREENSHOTS:

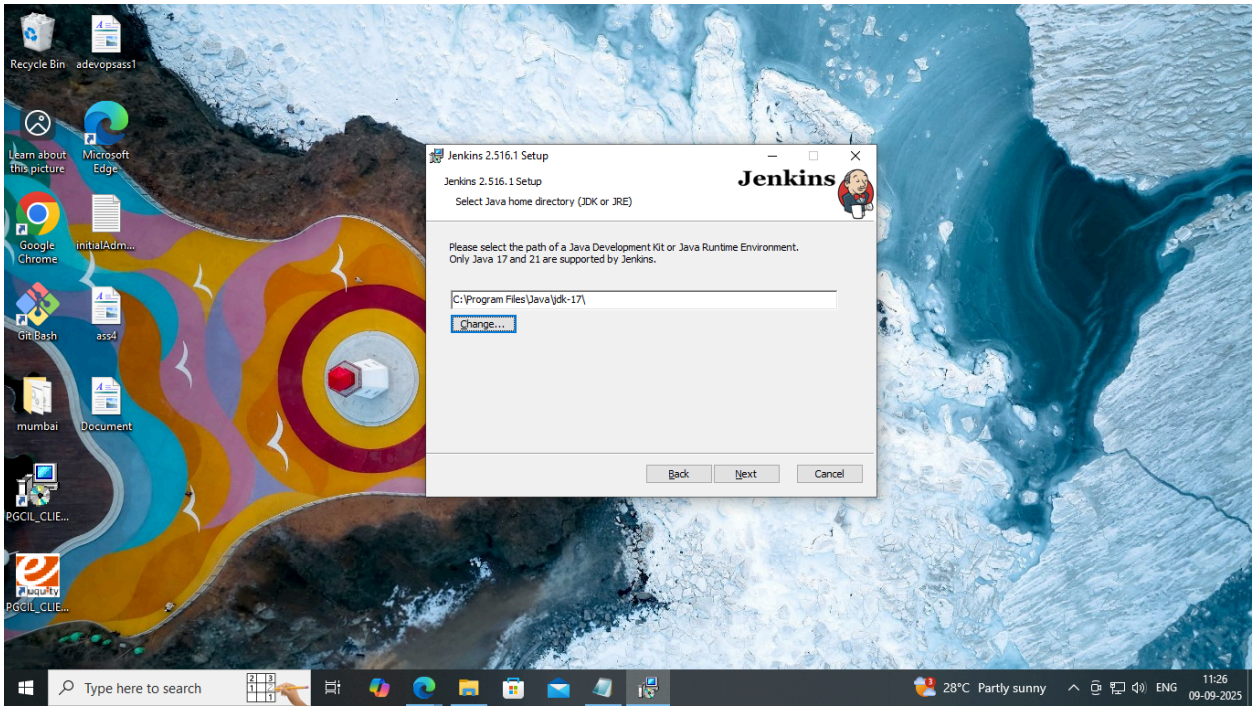
- Jenkins installation steps

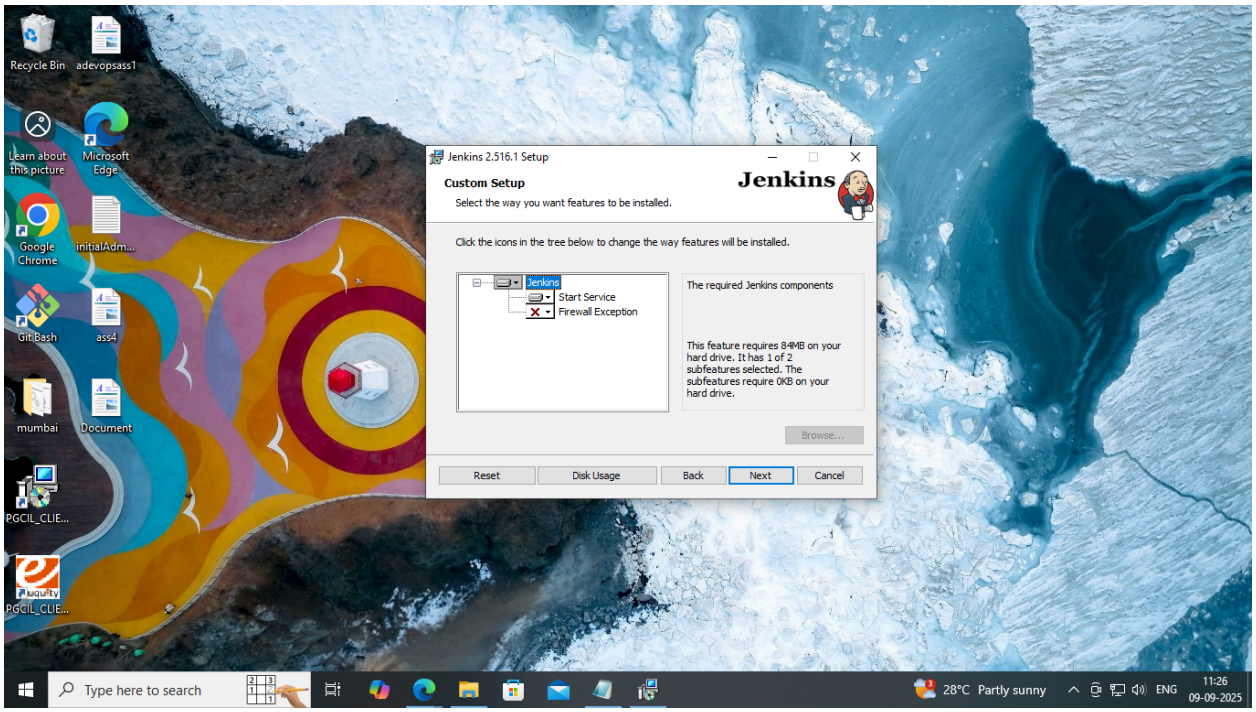


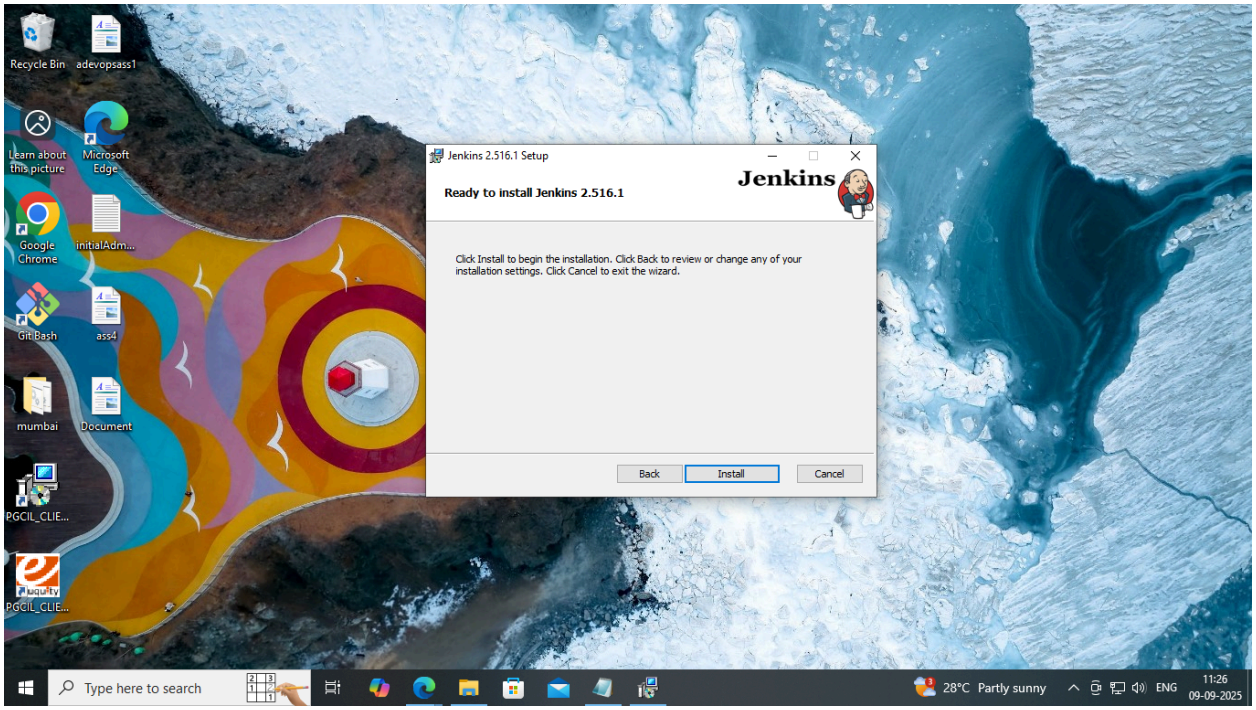


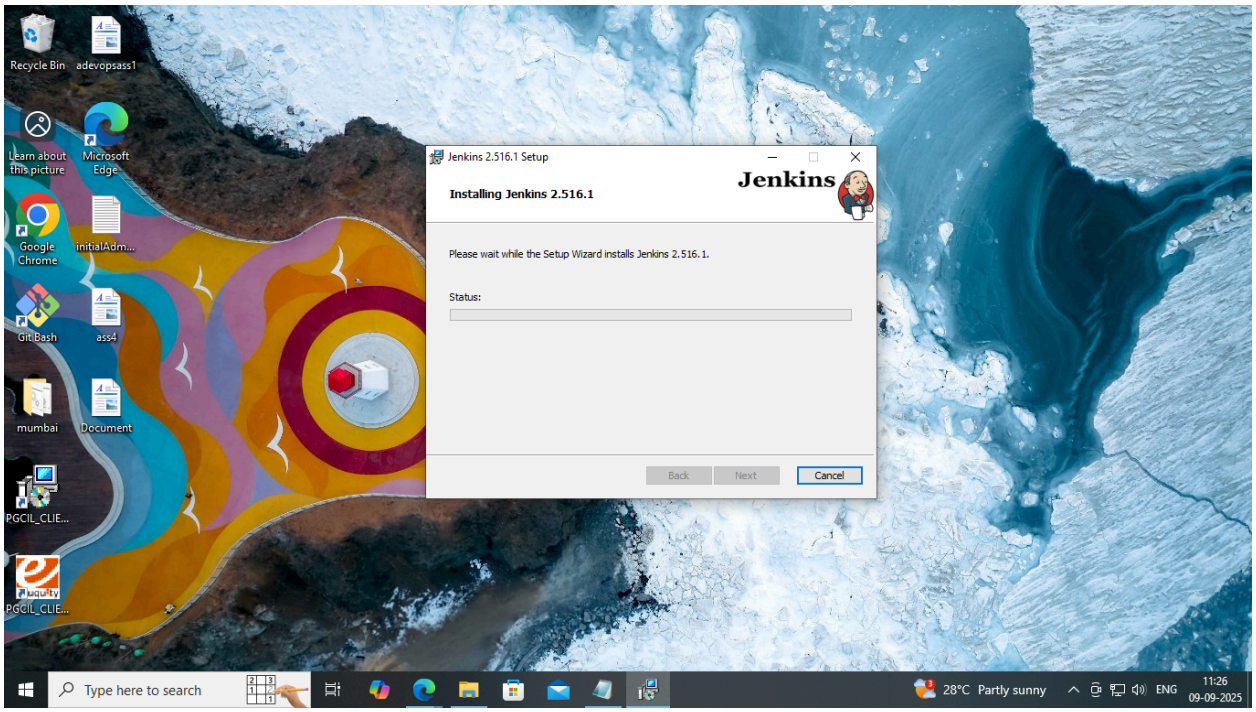

Recycle Bin
adevopsass1
Learn about this picture
Microsoft Edge
Google Chrome
initialAdm...
Git Bash
ass4
mumbai
Document
PGCIL_CLIE...
PGCIL_CLIE...
Jenkins 2.516.1 Setup
Service Logon Credentials
Enter service credentials for the service.
Jenkins
Jenkins 2.516.1 installs and runs as an independent Windows service. To operate in this manner, you must supply the user account credentials for Jenkins 2.516.1 to run successfully.
Logon Type:
Run service as LocalSystem (not recommended)
Run service as local or domain user:
Account:
Password:
Test Credentials
Back
Next
Cancel
Type here to search
A Big Bold Beautiful J...
ENG
11:24
09-09-2025

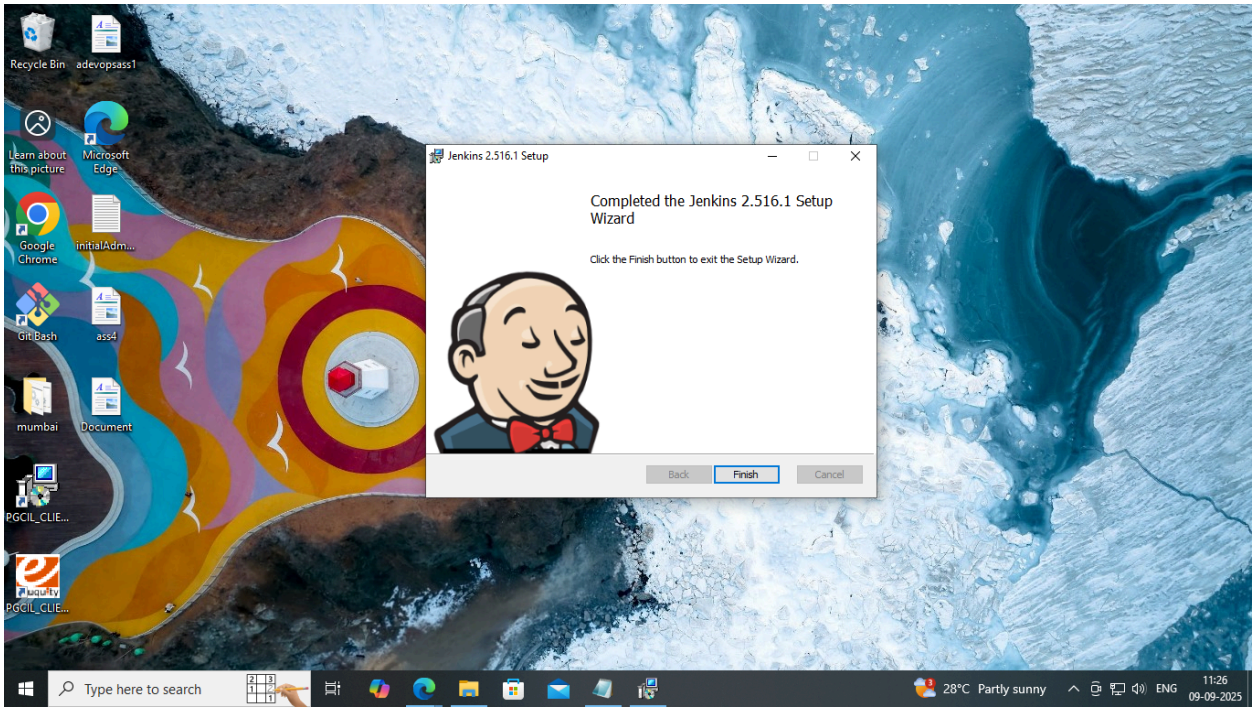

Recycle Bin
adevopsass1
Learn about this picture
Microsoft Edge
Google Chrome
initialAdm...
Git Bash
ass4
mumbai
Document
PGCIL_CLIE...
PGCIL_CLIE...
Jenkins 2.516.1 Setup
Port Selection
Choose a port for the service.
Jenkins
Please choose a port.
Port Number (1-65535):
8080
Test Port
It is recommended that you accept the selected default port.
Back
Next
Cancel
Type here to search
28°C Partly sunny
ENG
11:26
09-09-2025


Recycle Bin
adevopsass1
Learn about this picture
Microsoft Edge
Google Chrome
initialAdm...
Git Bash
ass4
mumbai
Document
PGCIL_CLIE...
PGCIL_CLIE...
Jenkins 2.516.1 Setup
Jenkins 2.516.1 Setup
Select Java home directory (JDK or JRE)
Jenkins
Please select the path of a Java Development Kit or Java Runtime Environment.
Only Java 17 and 21 are supported by Jenkins.
C:\Program Files\Java\jdk-17\
Change...
Back
Next
Cancel
Type here to search
28°C Partly sunny
ENG
11:26
09-09-2025


Recycle Bin
adevopsass1
Learn about this picture
Microsoft Edge
Google Chrome
initialAdm...
Git Bash
ass4
mumbai
Document
PGCIL_CLIE...
PGCIL_CLIE...
Jenkins 2.516.1 Setup
Custom Setup
Select the way you want features to be installed.
Jenkins
Click the icons in the tree below to change the way features will be installed.
Jenkins
Start Service
Firewall Exception
The required Jenkins components
This feature requires 84MB on your hard drive. It has 1 of 2 subfeatures selected. The subfeatures require 0KB on your hard drive.
Browse...
Reset
Disk Usage
Back
Next
Cancel
Type here to search
28°C Partly sunny
ENG
11:26
09-09-2025


Recycle Bin
adevopsass1
Learn about this picture
Microsoft Edge
Google Chrome
initialAdm...
Git Bash
ass4
mumbai
Document
PGCIL_CLIE...
PGCIL_CLIE...
Jenkins 2.516.1 Setup
Jenkins
Ready to install Jenkins 2.516.1
Click Install to begin the installation. Click Back to review or change any of your installation settings. Click Cancel to exit the wizard.
Back
Install
Cancel
Type here to search
28°C Partly sunny
ENG
11:26
09-09-2025


Recycle Bin
adevopsass1
Learn about this picture
Microsoft Edge
Google Chrome
initialAdm...
Git Bash
ass4
mumbai
Document
PGCIL_CLIE...
PGCIL_CLIE...
Jenkins 2.516.1 Setup
Jenkins
Installing Jenkins 2.516.1
Please wait while the Setup Wizard installs Jenkins 2.516.1.
Status:
Back
Next
Cancel
Type here to search
28°C Partly sunny
ENG
11:26
09-09-2025



CONCLUSION: This experiment successfully demonstrates the installation of Jenkins and provides a clear understanding of the concept of continuous integration.

LO MAPPING: LO1, LO3

ASSIGNMENT- 5

(Java Program using Jenkins)

AIM: To build a Java program using Jenkins.

THEORY: Jenkins is an open-source automation tool used for continuous integration and continuous delivery (CI/CD). It helps developers automate the building, testing, and deployment of their applications.

In this experiment, we use Jenkins to automatically build a simple Java program. The Java program (HelloWorld.java) contains a basic class that prints "Hello, World!" to the console. By configuring Jenkins to compile this Java file, we demonstrate how Jenkins can automate the build process.

When the Jenkins job is triggered, it will:

1. Fetch the Java source code.
2. Compile the HelloWorld.java file using the javac compiler.
3. Run the program to verify the output.

This shows how a Java application can benefit from automation using Jenkins.

STEP BY STEP PROCEDURE:

- **Open Jenkins Dashboard:-**

Go to http://localhost:8080 in your browser and log in.

- **Create a New Job:-**

Click “New Item” on the left sidebar.

Enter a job name (e.g., JavaBuild).

Select Freestyle project and click OK.

- **Configure Source Code Repository:-**

Scroll to Source Code Management section.

Select Git and enter your repository URL (e.g., GitHub repo with Java

code). Provide credentials if needed.

- **Set Build Environment:-**

Make sure JDK is configured in Jenkins (Manage Jenkins > Global Tool

Configuration). Select the JDK version under Build Environment or leave default.

- **Add Build Step:-**

Scroll to Build section.

Click Add build step > choose Invoke top-level Maven targets (if using Maven) or Execute shell (for manual Java compile commands).

For Maven, enter goals like clean install.

- **Save the Job:-**

Click Save at the bottom.

- **Build the Project:-**

Click Build Now on the left sidebar.

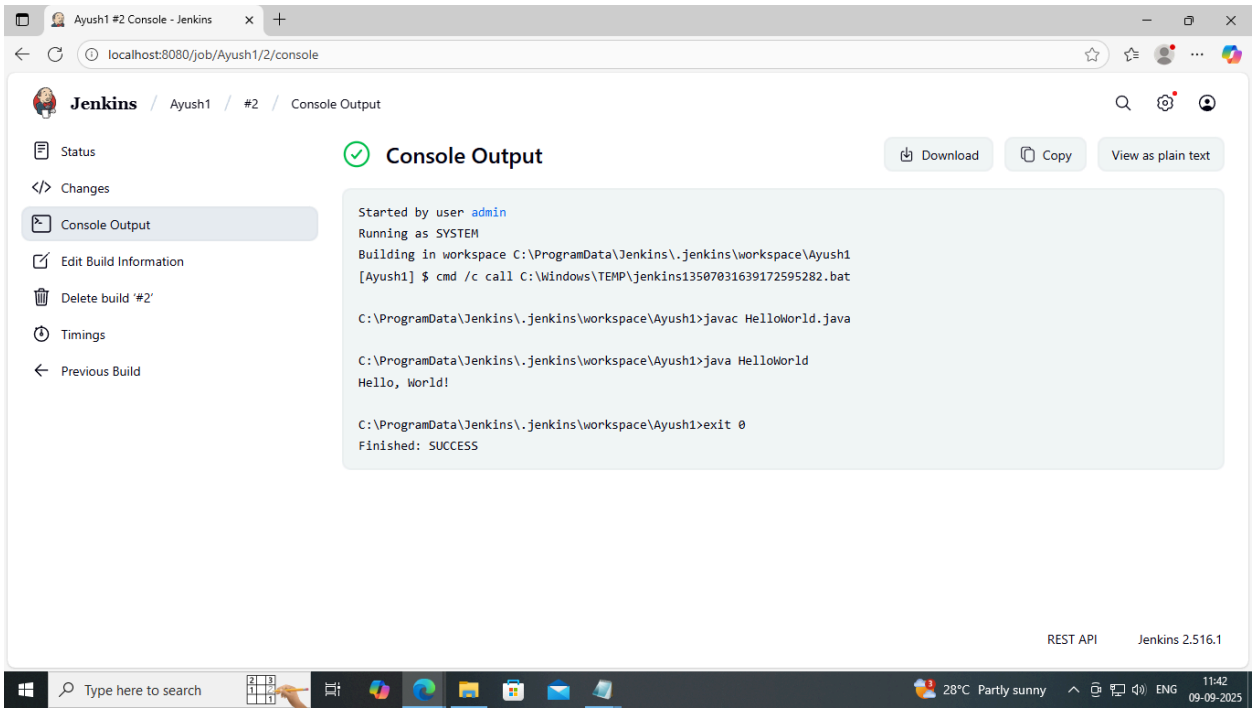
Monitor build progress in Build History and click the build number to see Console Output.

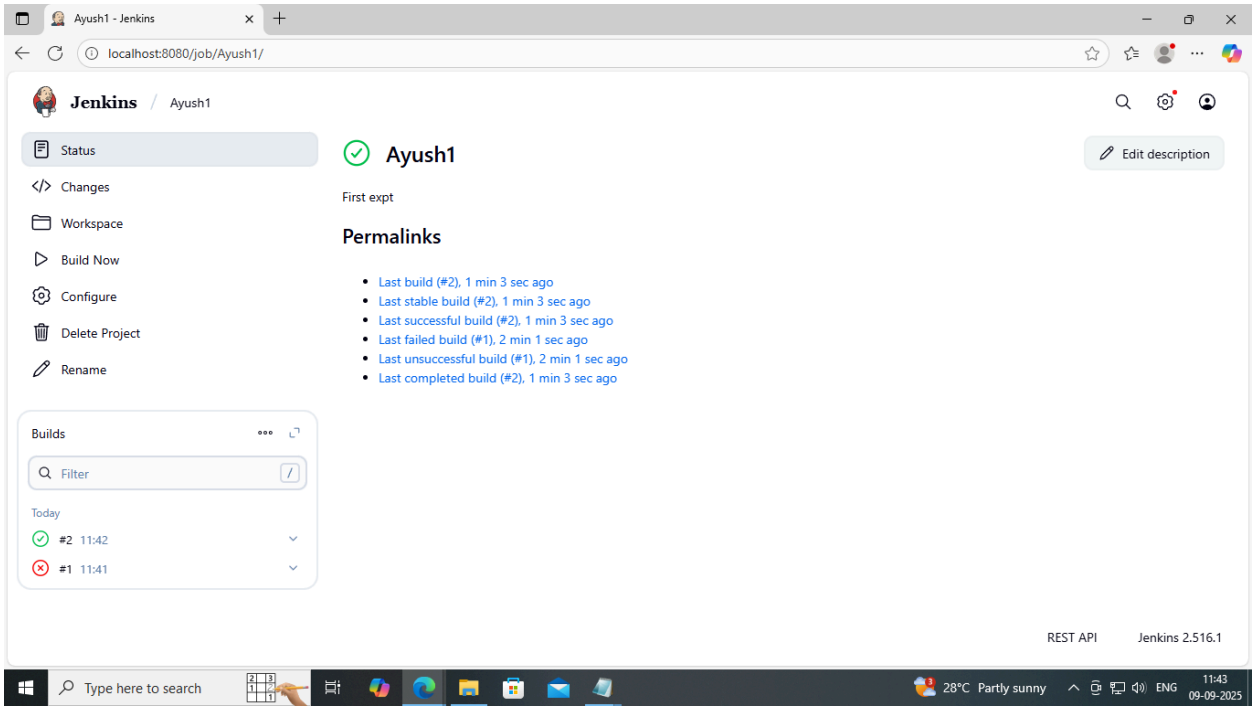
CODE:

```
public class HelloWorld {
    public static void main(String[] args) {

        System.out.println("Hello, World!");
    }
}
```

OUTPUT:





CONCLUSION: This experiment successfully builds a Java program using Jenkins.

LO MAPPING: LO1, LO3

ASSIGNMENT- 6

(Pipeline using Jenkins)

AIM: To build a pipeline using Jenkins.

THEORY: Jenkins Pipeline is a powerful tool that automates the process of building, testing, and deploying applications in a continuous integration/continuous delivery (CI/CD) environment. It allows the entire workflow to be defined as code, known as *Pipeline as Code*, which improves consistency, flexibility, and traceability. Pipelines make software delivery faster, more reliable, and easier to manage by automating repetitive tasks and integrating seamlessly with version control systems.

Features of Jenkins Pipeline:

- Allows automation of the entire CI/CD process.
- Supports Pipeline as Code using a Jenkinsfile.
- Provides visual representation of build stages for easy monitoring.
- Supports both Declarative and Scripted pipeline syntaxes.
- Can execute sequential and parallel tasks efficiently.
- Offers error handling and restart capabilities (resumes from where it stopped).
- Integrates with various plugins and tools like Git, Maven, Docker, etc.
- Enables version control of pipeline definitions.
- Provides scalability and supports distributed builds across multiple nodes.

There are two main ways to write a Jenkins Pipeline:

1. Declarative Pipeline: A structured and user-friendly syntax ideal for simple to moderate workflows, written within a Jenkinsfile.
2. Scripted Pipeline: A more flexible and advanced syntax written in Groovy, suitable for complex automation scenarios.

Overall, Jenkins Pipelines provide a robust, scalable, and automated approach to managing software build and deployment processes.

CODE:

- Declarative

Jenkinsfile (Declarative Pipeline)

```
pipeline {
  agent any
  stages {
    stage('Build') {
      steps {
        //
      }
    }
    stage('Test') {
      steps {
        //
      }
    }
    stage('Deploy') {
      steps {
```

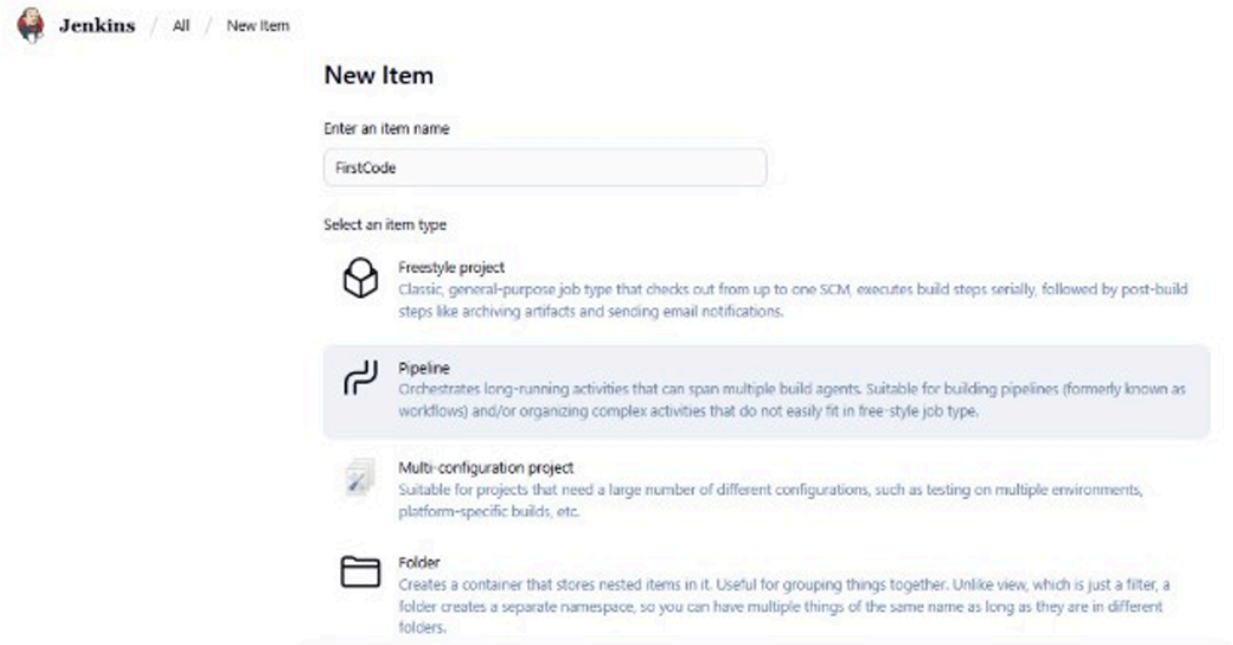
```
            //
          }
        }
    }
}
```

- Scripted

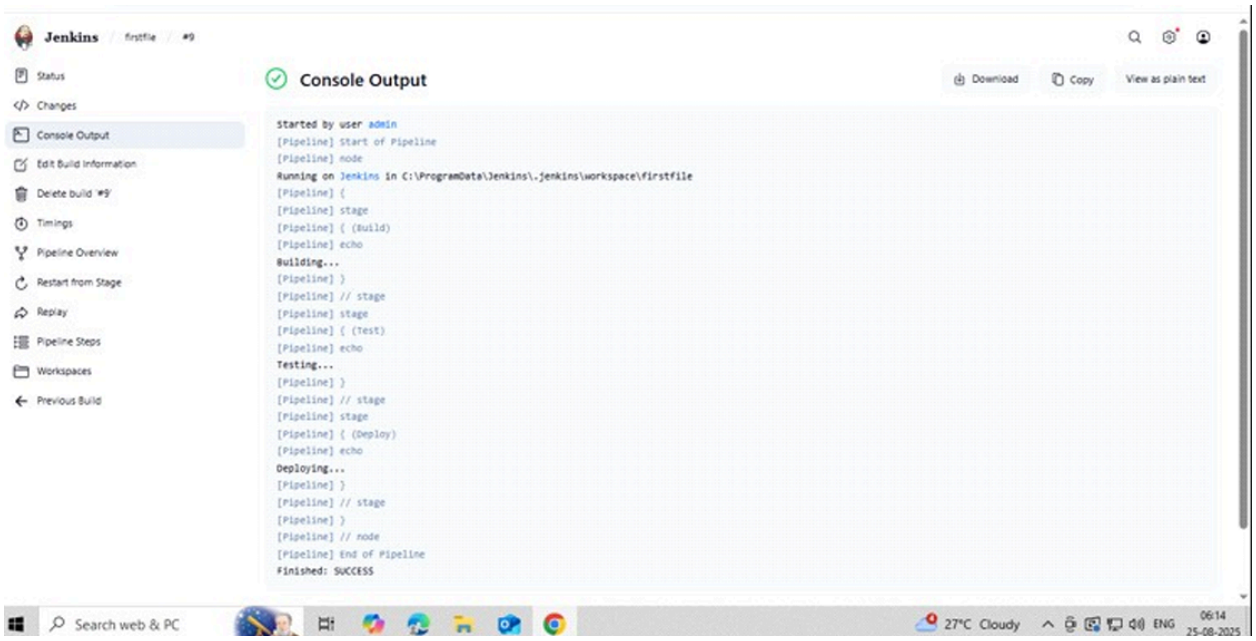
```
node {
stage('Build') {
//
}
stage('Test') {
//
}
stage('Deploy') {
//
}
}
```

OUTPUT:

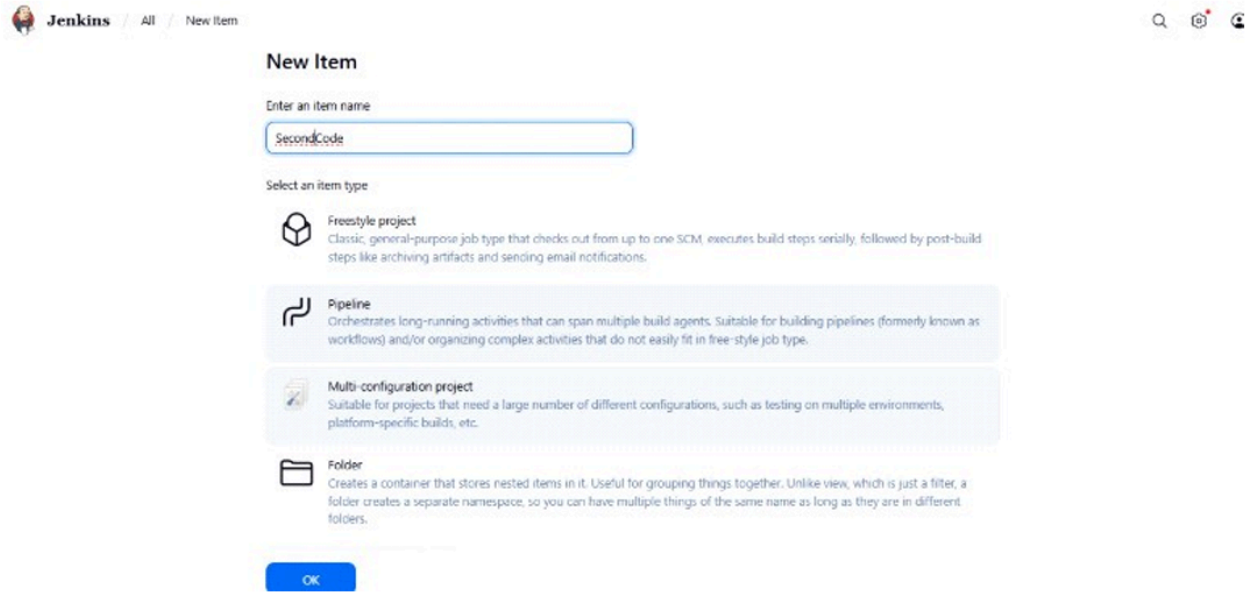
- Declarative pipeline creation



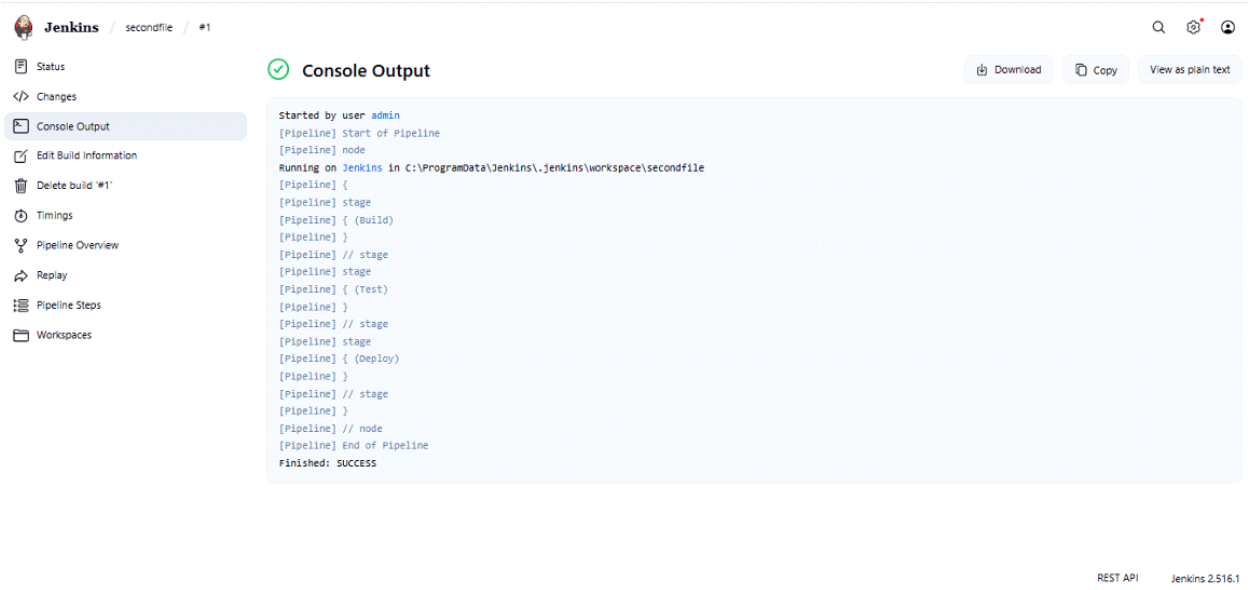
- Console output (declarative)



- Scripted pipeline creation



- Console output (scripted)



CONCLUSION: This experiment successfully demonstrates the creation of a Jenkins pipeline, providing a clear understanding of its features, structure, and role in automating the CI/CD process.

LO MAPPING: LO1, LO3

ASSIGNMENT- 7

(Docker)

AIM: To understand Docker architecture and container life cycle, install docker, deploy container in Docker.

THEORY: Docker is an open-source platform that enables developers to automate the deployment, scaling, and management of applications using containers. A container is a lightweight, standalone, and executable package that includes everything required to run an application — such as code, runtime, system tools, libraries, and configuration settings. Docker ensures consistent behavior of applications across different environments, improving development, testing, and deployment workflows.

Docker Architecture

Docker follows a **client-server architecture**, consisting of three main components:

1. Docker Client

The Docker Client is the main tool users use to interact with Docker. It sends commands (like docker run, docker build) to the Docker Daemon via the Docker CLI. The client can connect to Docker Daemons running locally or remotely.

2. Docker Daemon (dockerd)

The Docker Daemon runs on the host system and listens for API requests from the Docker Client. It manages the building, running, and monitoring of containers and other Docker objects.

Docker Objects:

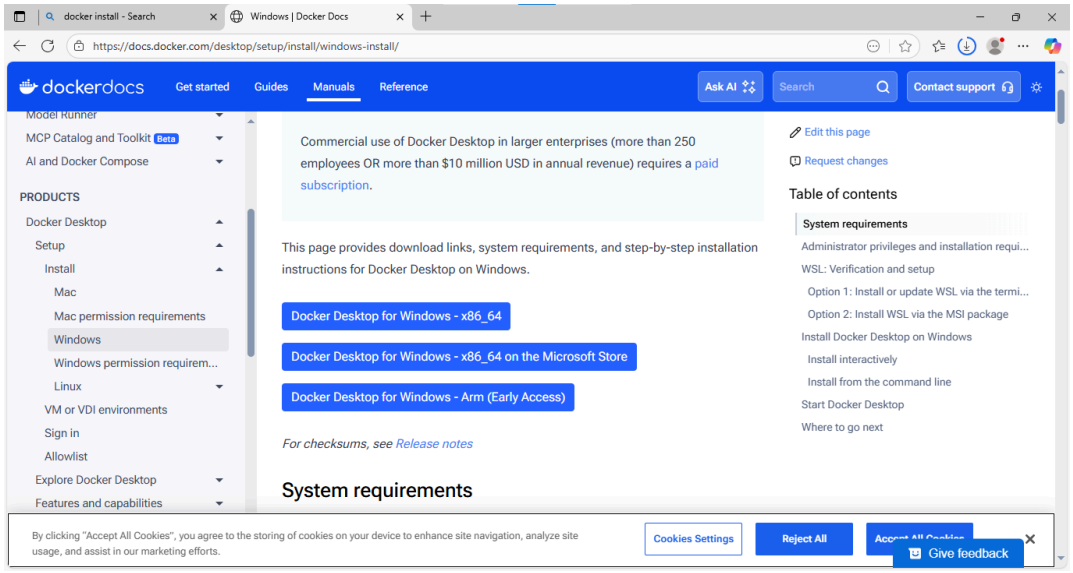
- **Images:** Read-only templates used to create containers.
- **Containers:** Running instances of Docker images.
- **Volumes:** Persistent storage used by containers.
- **Networks:** Enable communication between containers or between containers and the outside world.

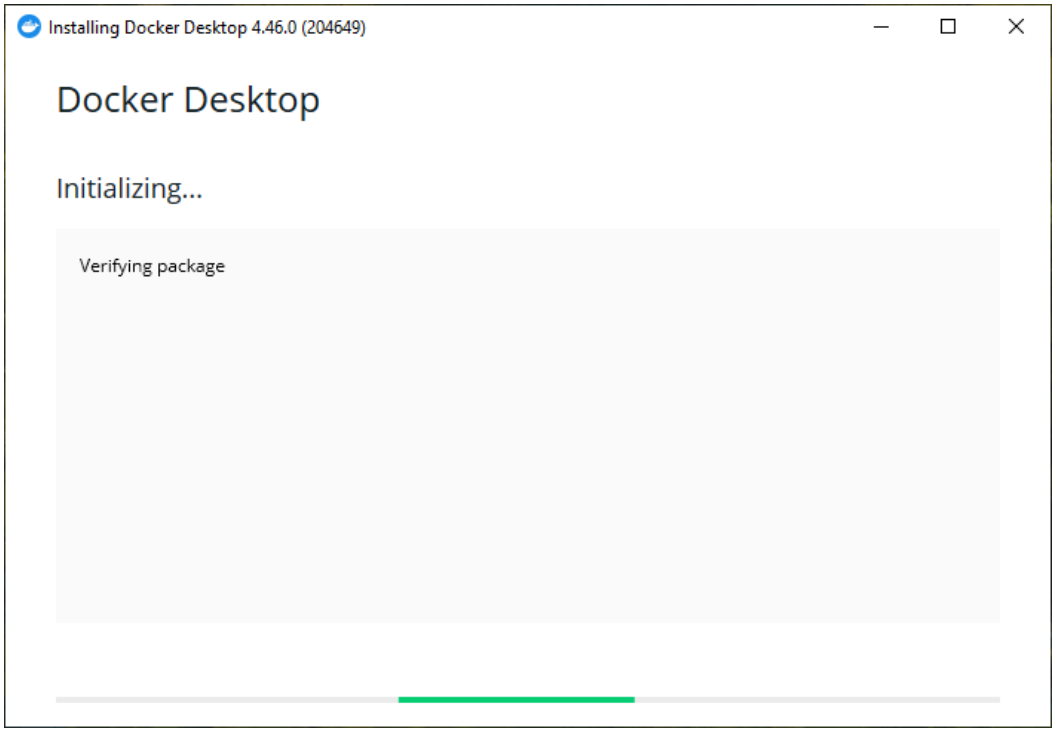
3. Docker Registry

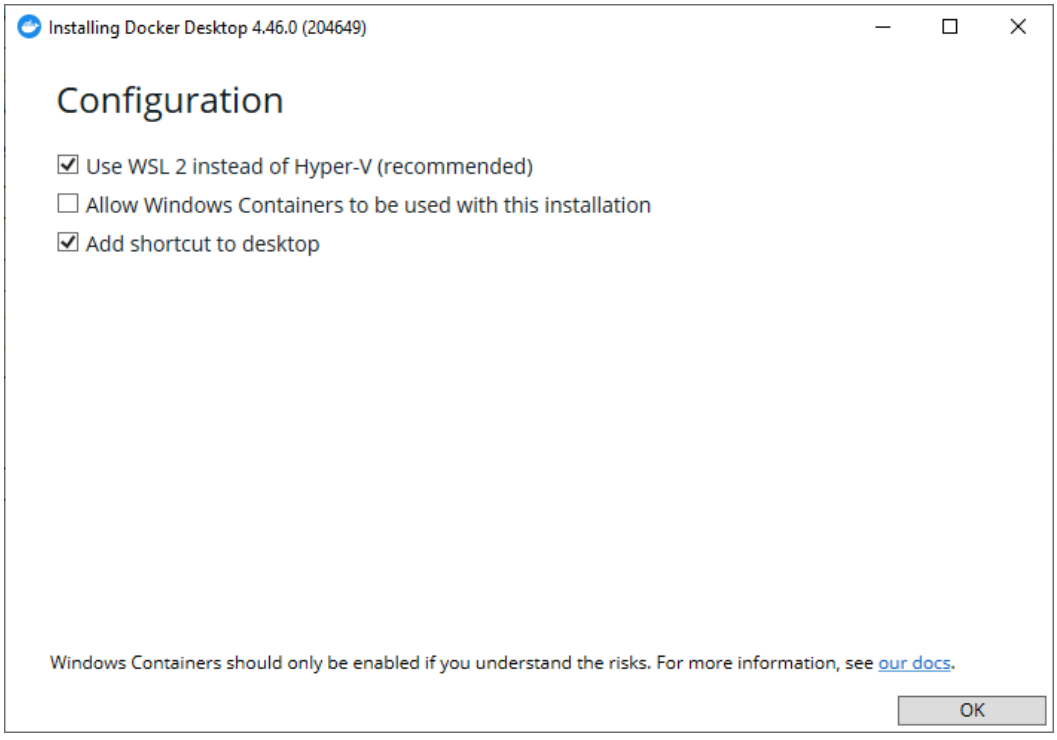
A Docker Registry stores and distributes Docker images. Docker Hub is the default public registry, but private registries can also be used. You can pull images from a registry or push your own images to it.

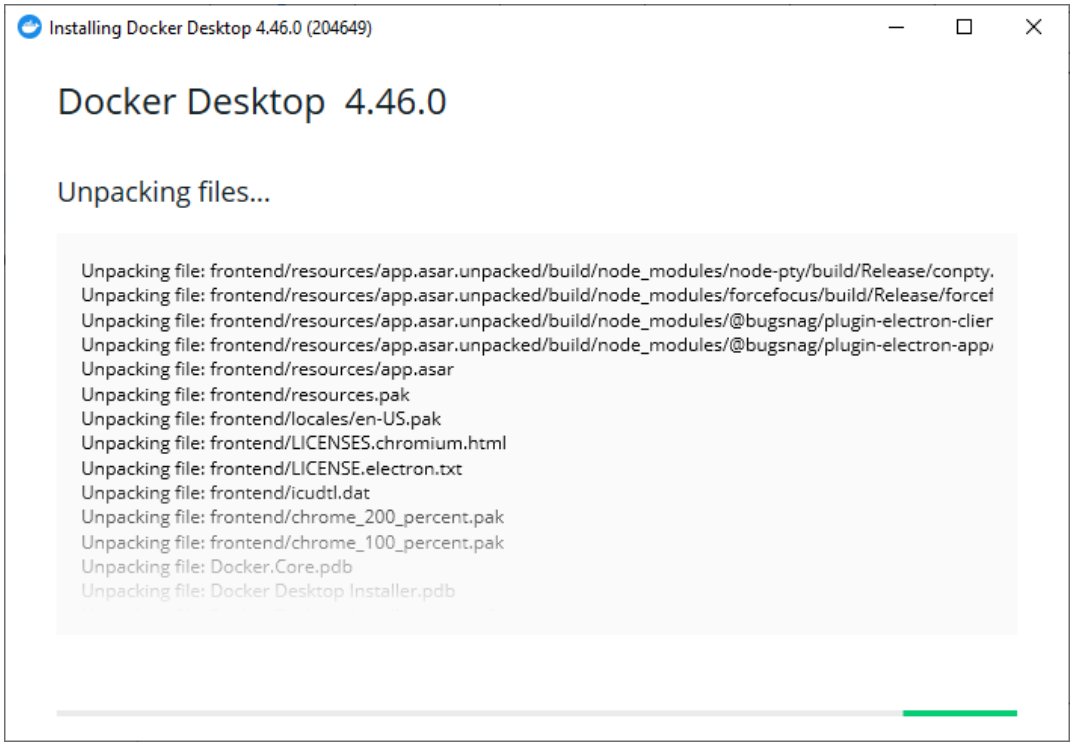
SCREENSHOTS:

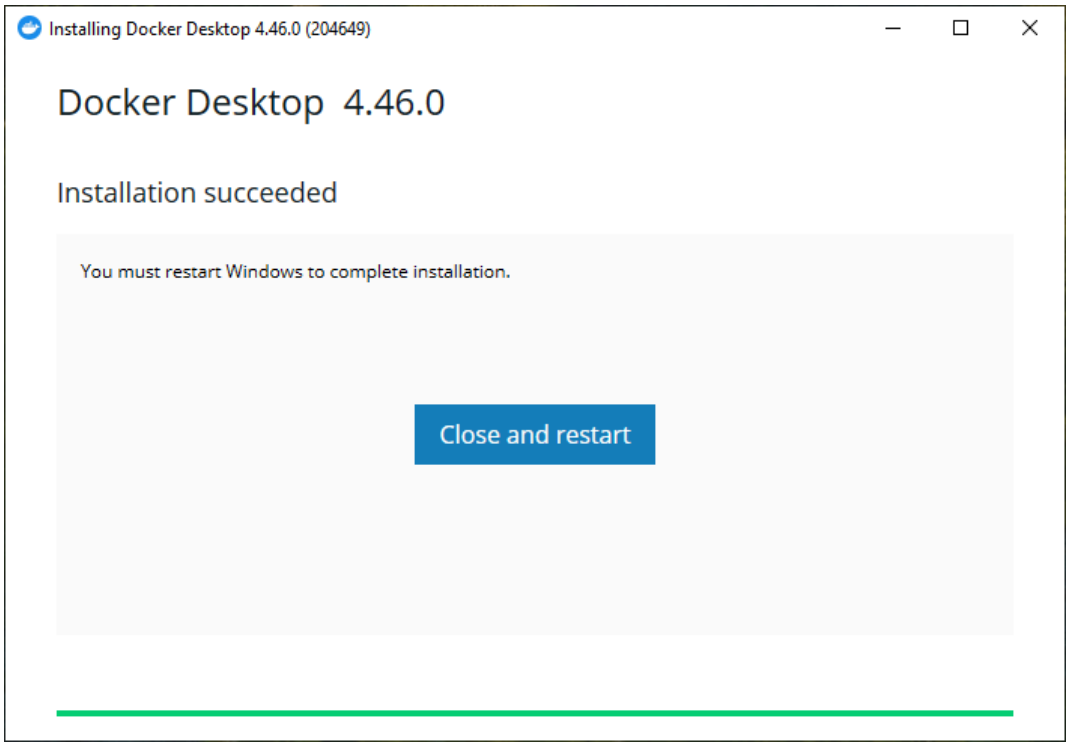
- Docker installation

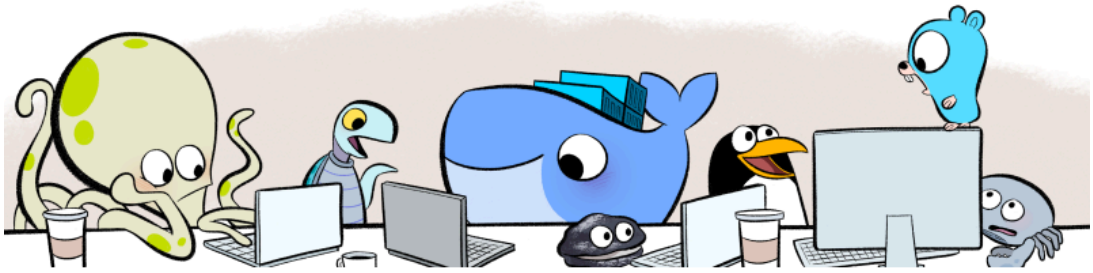



Installing Docker Desktop 4.46.0 (204649)
Docker Desktop
Initializing...
Verifying package


Installing Docker Desktop 4.46.0 (204649)
Configuration
Use WSL 2 instead of Hyper-V (recommended)
Allow Windows Containers to be used with this installation
Add shortcut to desktop
Windows Containers should only be enabled if you understand the risks. For more information, see our docs.
OK


Installing Docker Desktop 4.46.0 (204649)
Docker Desktop 4.46.0
Unpacking files...
Unpacking file: frontend/resources/app.asar.unpacked/build/node_modules/node-pty/build/Release/conpty.
Unpacking file: frontend/resources/app.asar.unpacked/build/node_modules/forcefocus/build/Release/forcef
Unpacking file: frontend/resources/app.asar.unpacked/build/node_modules/@bugsnag/plugin-electron-clier
Unpacking file: frontend/resources/app.asar.unpacked/build/node_modules/@bugsnag/plugin-electron-app
Unpacking file: frontend/resources/app.asar
Unpacking file: frontend/resources.pak
Unpacking file: frontend/locales/en-US.pak
Unpacking file: frontend/LICENSES.chromium.html
Unpacking file: frontend/LICENSE.electron.txt
Unpacking file: frontend/icudtl.dat
Unpacking file: frontend/chrome_200_percent.pak
Unpacking file: frontend/chrome_100_percent.pak
Unpacking file: Docker.Core.pdb
Unpacking file: Docker Desktop Installer.pdb


Installing Docker Desktop 4.46.0 (204649)
Docker Desktop 4.46.0
Installation succeeded
You must restart Windows to complete installation.
Close and restart



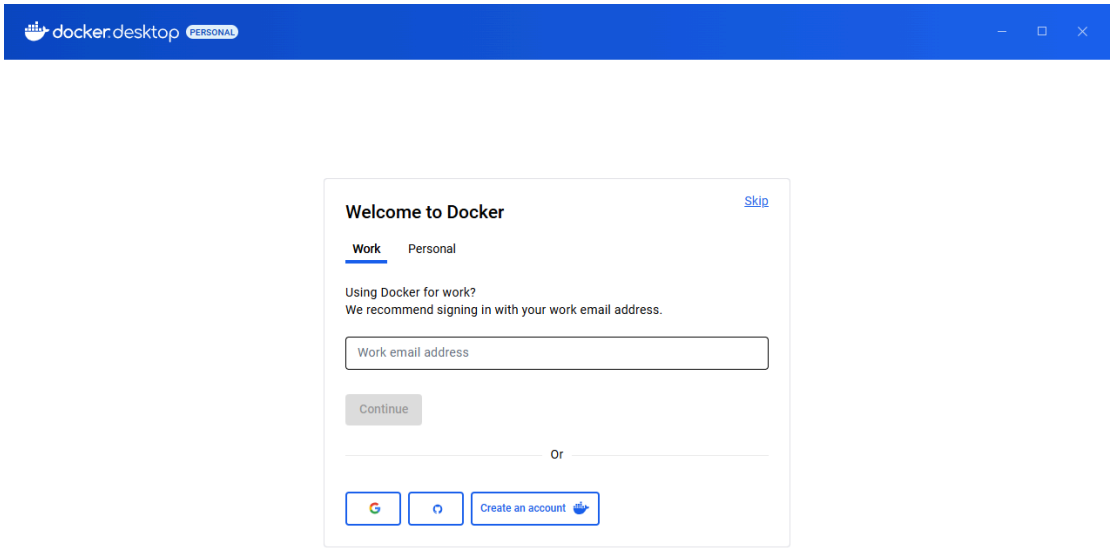
Docker Subscription Service Agreement

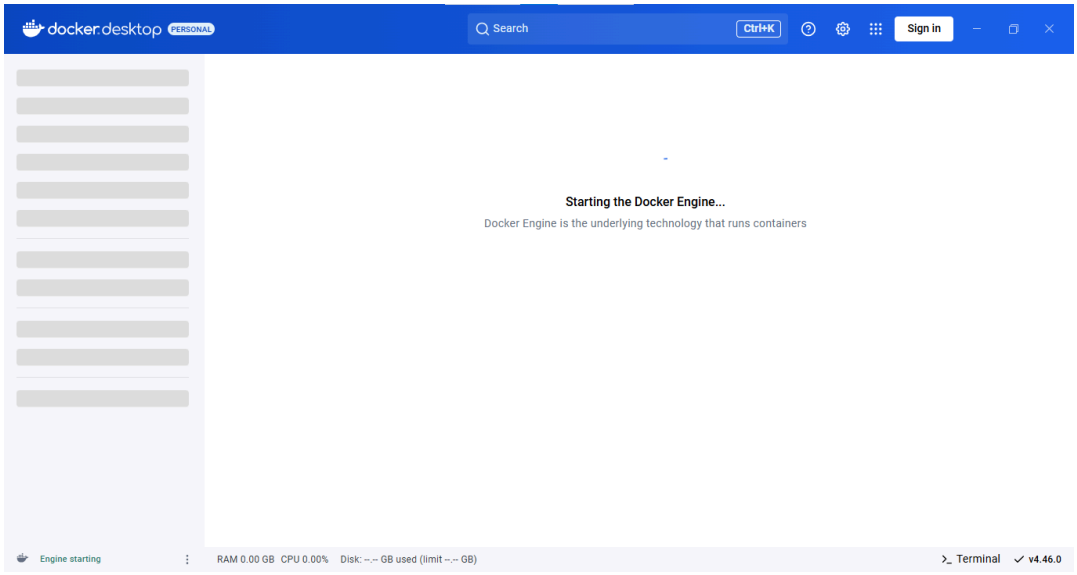
By selecting **accept**, you agree to the Subscription Service Agreement, the Docker Data Processing Agreement, the Data Privacy Policy and the Docker AI Supplemental Terms.

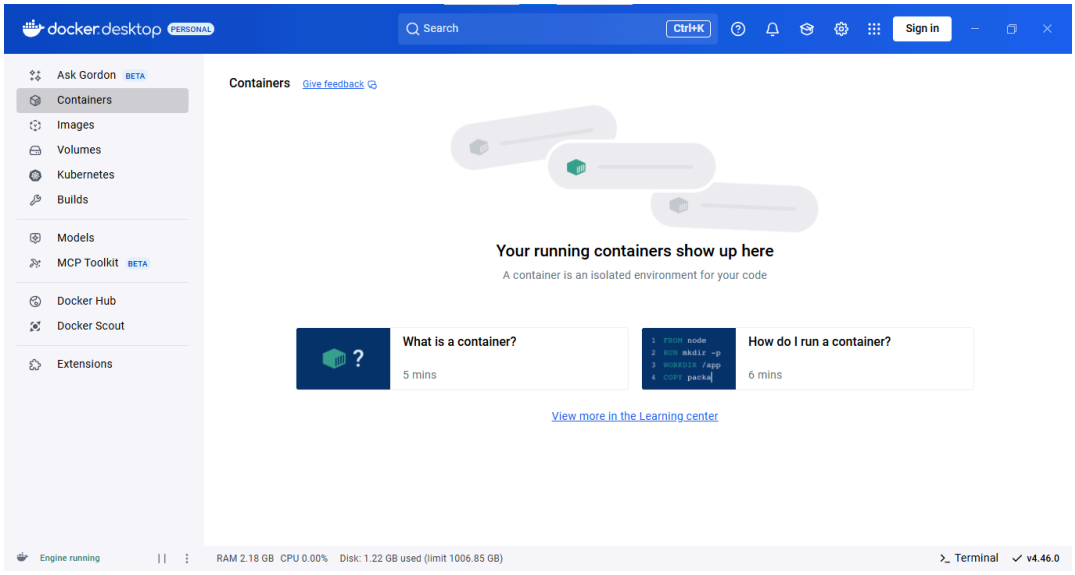
Commercial use of Docker Desktop at a company of more than 250 employees OR more than $10 million in annual revenue requires a paid subscription (Pro, Team, or Business). See subscription details

View Full Terms

Accept Close







CONCLUSION: This experiment successfully demonstrates Docker architecture, container lifecycle, installation, and deployment, highlighting how Docker simplifies and automates application management.

LO MAPPING: LO1, LO5

ASSIGNMENT- 8

(Web Application image using Dockerfile)

AIM: To build an image for a sample web application using Dockerfile.

THEORY:

1. Creating a Dockerfile: A Dockerfile is a text file that contains instructions for building a Docker image. It specifies the base image, sets up the environment, and defines how the application should be installed and configured within the container.

a. Base Image:

The first line of your Dockerfile should specify a base image. The base image provides the foundational operating system and environment for your application. You can use official images from Docker Hub or create custom base images.

Example:

```
FROM ubuntu:20.04
```

b. Setting the Working Directory:

Set a working directory within the container where your application's code will reside. This makes it easier to manage and run your application.

Example:

```
WORKDIR /app
```

c. Copying Files:

Use the COPY or ADD instructions to copy your application code and any necessary

files into the container.

Example:

```
COPY . /app
```

d. Installing Dependencies:

If your application requires dependencies like libraries or packages, you can use the container's package manager or run custom commands to install them.

Example (for a Node.js application):

```
RUN npm install
```

e. Exposing Ports:

If your web application listens on a specific port, use the EXPOSE instruction to specify the port that should be exposed when the container is running.

Example:

```
EXPOSE 80
```

f. Command to Start the Application:

Use the CMD or ENTRYPOINT instruction to define the command that should be executed when the container is started. This should be the command that launches your web application.

Example:

```
CMD ["node", "app.js"]
```

2. Docker Build: To build the Docker image, use the docker build command. Provide the path to the directory containing your Dockerfile and application code.

Example:

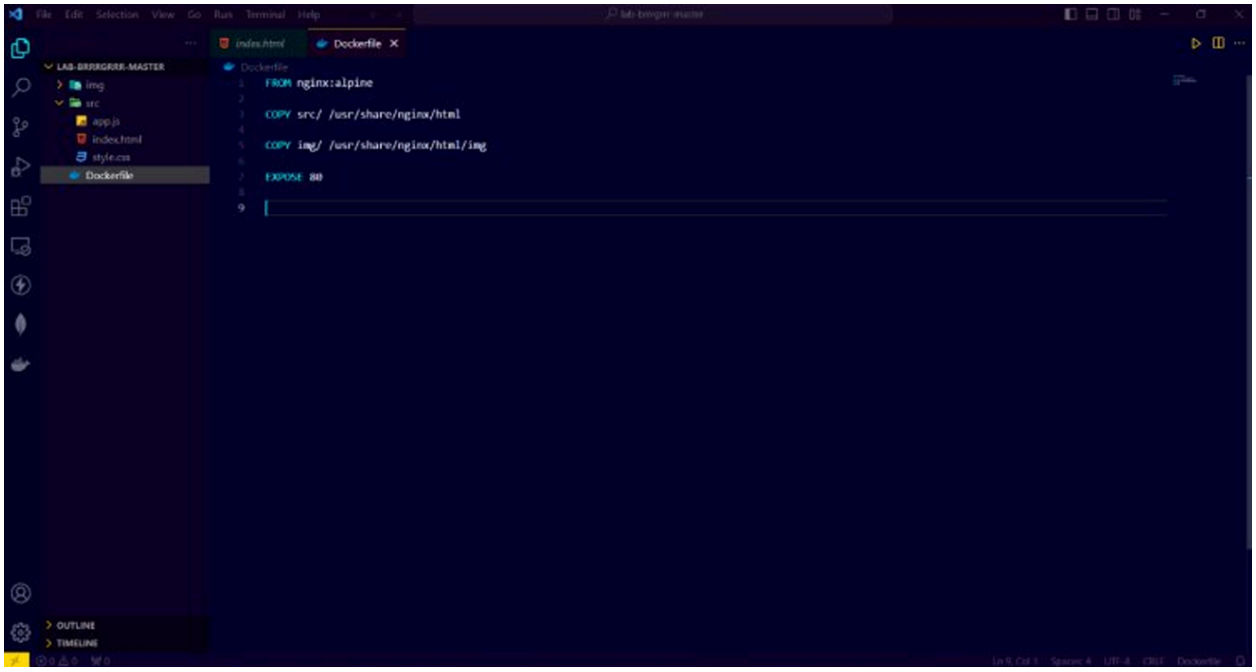
```
docker build -t my-web-app .
```

3. Docker Run: To run a container from the image you've built, use the docker run command. Map the container's exposed port to a port on your host machine if needed.

Example:

docker run -p 8080:80 my-web-app

COMMANDS/SCREENSHOTS:



```
C:\Windows\System32\cmd.e
Microsoft Windows [Version 10.0.22621.2283]
(c) Microsoft Corporation. All rights reserved.

E:\lab-brrrgrrr-master>code .

E:\lab-brrrgrrr-master>docker build -t sample-web-app .
[+] Building 9.3s (9/9) FINISHED                                docker:default
 => [internal] load .dockerignore                                          0.1s
 => => transferring context: 2B                                            0.0s
 => [internal] load build definition from Dockerfile                       0.1s
 => => transferring dockerfile: 145B                                       0.0s
 => [internal] load metadata for docker.io/library/nginx:alpine            4.3s
 => [auth] library/nginx:pull token for registry-1.docker.io               0.0s
 => [1/3] FROM docker.io/library/nginx:alpine@sha256:4c93a3bd8bf95412      4.5s
 => => resolve docker.io/library/nginx:alpine@sha256:4c93a3bd8bf95412      0.1s
 => => sha256:96526aa774ef0126ad0fe9e9a95764c5fc37f40 3.40MB / 3.40MB      1.1s
 => => sha256:f2004135e416117cc29b9fd1a5c217b19bd2555 1.90MB / 1.90MB      1.1s
 => => sha256:4c93a3bd8bf95412889dd84213570102176b605 1.65kB / 1.65kB      0.0s
 => => sha256:34b58b4f5c6d133d97298cbaae140283dc325ff 1.99kB / 1.99kB      0.0s
 => => sha256:d571254277f6a0ba9d0c4a88f29b944476dcd4 16.69kB / 16.69kB     0.0s
 => => sha256:fbf1cf5026c467c51d6532a304acb35164d5aaee73d 626B / 626B      0.5s
 => => sha256:38966af6931dff98fc0ff3f63f490938a895c2739b2 958B / 958B      0.9s
 => => sha256:c3ee70732c61e54665d4cd10d75c2962958b72d6dbe 370B / 370B      1.2s
 => => sha256:76cbc9ea6abf200d8089d7fe3c6ad19d6f0ce9e 1.40kB / 1.40kB      1.4s
 => => sha256:7e2fd992447a7948a6090f3c4eb2dd92ad37ae1 1.21kB / 1.21kB      1.4s
 => => extracting sha256:96526aa774ef0126ad0fe9e9a95764c5fc37f409ab9e      0.2s
 => => sha256:37f8bcf34db7931f3e1386852d3dde3d244cb 12.64MB / 12.64MB      3.3s
 => => extracting sha256:f2004135e416117cc29b9fd1a5c217b19bd25556f8f5      0.6s
 => => extracting sha256:fbf1cf5026c467c51d6532a304acb35164d5aaee73d5      0.0s
 => => extracting sha256:38966af6931dff98fc0ff3f63f490938a895c2739b20      0.0s
 => => extracting sha256:c3ee70732c61e54665d4cd10d75c2962958b72d6dbef      0.0s
 => => extracting sha256:7e2fd992447a7948a6090f3c4eb2dd92ad37ae1140d6      0.0s
 => => extracting sha256:76cbc9ea6abf200d8089d7fe3c6ad19d6f0ce9eb0519      0.0s
 => => extracting sha256:37f8bcf34db7931f3e1386852d3dde3d244cb54f28aa      0.8s
 => [internal] load build context                                          0.1s
 => => transferring context: 99.20kB                                       0.1s
 => [2/3] COPY src/ /usr/share/nginx/html                                  0.1s
 => [3/3] COPY img/ /usr/share/nginx/html/img                              0.1s
 => exporting to image                                                     0.1s
 => => exporting layers                                                    0.0s
```

```
C:\Windows\System32\cmd.e
 => => sha256:d571254277f6a0ba9d0c4a88f29b944476dcd4 16.69kB / 16.69kB  0.0s
 => => sha256:fbf1cf5026c467c51d6532a304acb35164d5aaee73d 626B / 626B   0.5s
 => => sha256:38966af6931dff98fc0ff3f63f490938a895c2739b2 958B / 958B   0.9s
 => => sha256:c3ee70732c61e54665d4cd10d75c2962958b72d6dbe 370B / 370B   1.2s
 => => sha256:76cbc9ea6abf200d8089d7fe3c6ad19d6f0ce9e 1.40kB / 1.40kB   1.4s
 => => sha256:7e2fd992447a7948a6090f3c4eb2dd92ad37ae1 1.21kB / 1.21kB   1.4s
 => => extracting sha256:96526aa774ef0126ad0fe9e9a95764c5fc37f409ab9e   0.2s
 => => sha256:37f8bcf34db7931f3e1386852d3dde3d244cb 12.64MB / 12.64MB   3.3s
 => => extracting sha256:f2004135e416117cc29b9fd1a5c217b19bd25556f8f5   0.6s
 => => extracting sha256:fbf1cf5026c467c51d6532a304acb35164d5aaee73d5   0.0s
 => => extracting sha256:38966af6931dff98fc0ff3f63f490938a895c2739b20   0.0s
 => => extracting sha256:c3ee70732c61e54665d4cd10d75c2962958b72d6dbef   0.0s
 => => extracting sha256:7e2fd992447a7948a6090f3c4eb2dd92ad37ae1140d6   0.0s
 => => extracting sha256:76cbc9ea6abf200d8089d7fe3c6ad19d6f0ce9eb0519   0.0s
 => => extracting sha256:37f8bcf34db7931f3e1386852d3dde3d244cb54f28aa   0.8s
 => [internal] load build context                                       0.1s
 => => transferring context: 99.20kB                                    0.1s
 => [2/3] COPY src/ /usr/share/nginx/html                               0.1s
 => [3/3] COPY img/ /usr/share/nginx/html/img                           0.1s
 => exporting to image                                                  0.1s
 => => exporting layers                                                 0.0s
 => => writing image sha256:713c7cdaaf784f542f0c4e31aae5b6951be50cdb7   0.0s
 => => naming to docker.io/library/sample-web-app                       0.0s

What's Next?
  View a summary of image vulnerabilities and recommendations → docker scout quickview

E:\lab-brrrgrrr-master>docker images
REPOSITORY      TAG       IMAGE ID       CREATED          SIZE
sample-web-app  latest    713c7cdaaf78   7 seconds ago    42.7MB
myimage         latest    438bb56a50a3   13 hours ago     122MB
ubuntu          latest    c6b84b685f35   6 weeks ago      77.8MB
hello-world     latest    9c7a54a9a43c   4 months ago     13.3kB

E:\lab-brrrgrrr-master>docker run -d -p 8080:80 sample-web-app
e4bf6da28d0642a0b54c01578e67f98e0221350032e1b60ea54bc901d8467dc3

E:\lab-brrrgrrr-master>
```

CONCLUSION: This experiment successfully demonstrates the process of building a Docker image for a sample web application using a Dockerfile. It provides a clear understanding of how Docker automates the packaging of applications along with their dependencies, ensuring consistent behavior across different environments. By writing Dockerfiles and building images, one can define base images, copy application files, install dependencies, and specify commands to run the application. This approach enables the deployment of scalable, portable, and reproducible containerized applications, which is a fundamental practice in modern DevOps workflows and continuous deployment pipelines.

LO MAPPING: LO1, LO5

ASSIGNMENT- 9

(Nagios)

AIM: To install Nagios on an Ubuntu system.

THEORY: Nagios is an open-source monitoring tool used to monitor systems, networks, and infrastructure. It allows administrators to track the health and performance of servers, applications, and network devices, and provides alerts when problems are detected. Nagios ensures IT infrastructure runs smoothly and helps prevent downtime by proactively notifying issues before they impact users.

Key Features of Nagios:

- Host and Service Monitoring: Monitors the status of servers, network devices, and applications.
- Alerting and Notifications: Sends alerts via email or SMS when a host or service fails or exceeds thresholds.
- Web Interface: Provides a graphical dashboard to view the status, performance, and history of monitored systems.
- Plugin Architecture: Supports custom scripts or programs to monitor various services such as HTTP, SSH, CPU usage, disk space, etc.
- Extensibility: Can integrate with third-party tools and automation scripts for advanced monitoring.

Nagios Architecture:

Nagios uses a client-server architecture consisting of the following components:

1. Nagios Core (Server): The central engine that performs monitoring, executes checks, and generates alerts.
2. Nagios Plugins: Scripts or programs executed by Nagios to check the status of hosts and services.
3. NRPE (Nagios Remote Plugin Executor): Allows monitoring of remote Linux/Unix systems.
4. Web Interface: Provides a user-friendly graphical interface for viewing system status, alerts, and reports.

Nagios is widely used in IT environments for proactive monitoring, helping reduce downtime and maintain service reliability by providing real-time alerts and performance metrics.

INSTALLATION PROCEDURE/SCREESHOTS:

1. Update System & Install Dependencies

```
sudo apt update
sudo apt install -y apache2 php libapache2-mod-php build-essential \
libgd-dev unzip curl wget libssl-dev daemon make gcc
```

2. Create Nagios User and Group

```
sudo useradd nagios
sudo groupadd nagcmd
sudo usermod -aG nagcmd nagios
sudo usermod -aG nagcmd www-data
```

3. Download and Compile Nagios Core

```
cd /tmp
wget https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.5.1.tar.gz
```

```
tar -xzf nagios-4.5.1.tar.gz
cd nagios-4.5.1
./configure --with-command-group=nagcmd
make all
sudo make install
sudo make install-commandmode
sudo make install-init
sudo make install-config
sudo make install-webconf
```

4. Set Up Web Interface Authentication

Create a login user for the web interface (e.g., admin):

```
sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users admin
```

Note: Choose a strong password and remember it.

5. Install Nagios Plugins

```
cd /tmp
wget https://nagios-plugins.org/download/nagios-plugins-2.3.3.tar.gz
tar -xzf nagios-plugins-2.3.3.tar.gz
cd nagios-plugins-2.3.3
./configure --with-nagios-user=nagios --with-nagios-group=nagios
make
sudo make install
```

6. Enable Required Apache Modules

```
sudo a2enmod rewrite cgi
sudo systemctl restart apache2
```

7. Start Nagios Service

sudo systemctl enable nagios

sudo systemctl start nagios

Accessing the Nagios Web Interface

1. Get your server IP:

ip a | grep inet

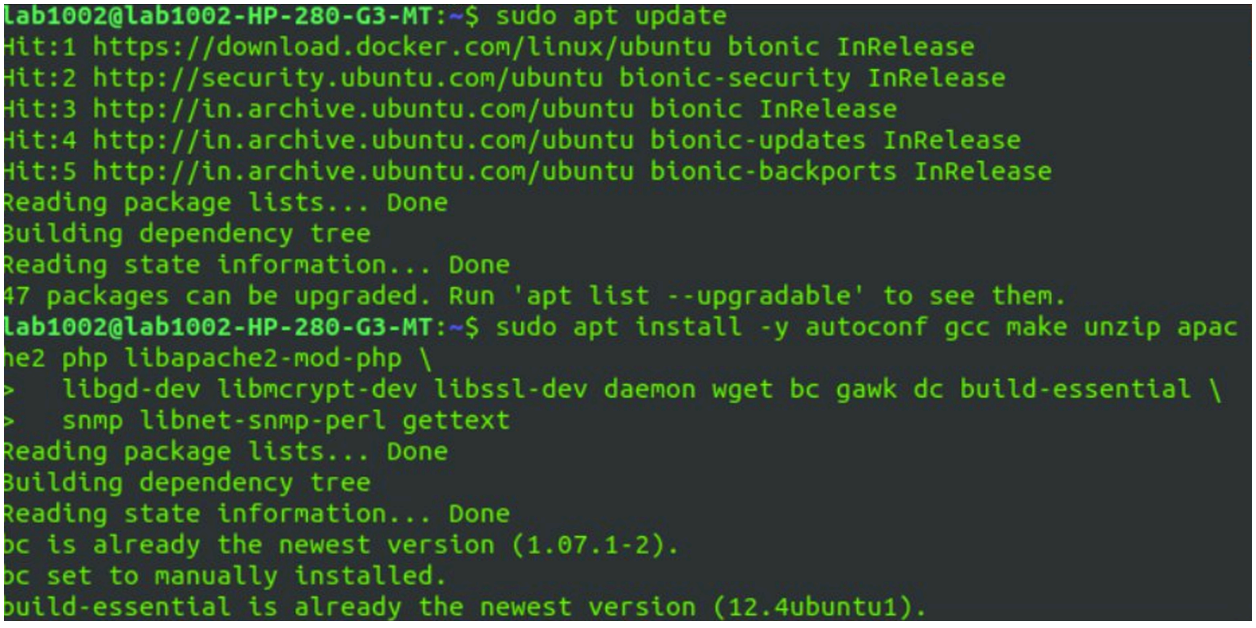
Example output: 192.168.4.104

2. Open a browser and visit:

http://192.168.4.104/nagios

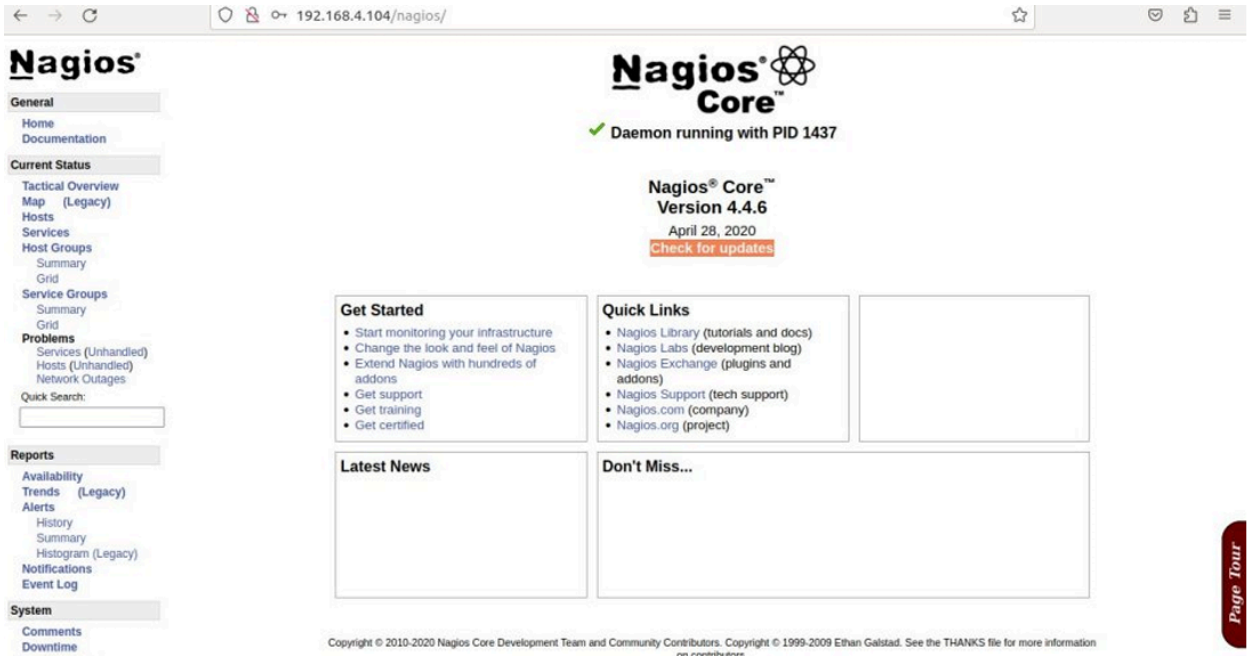
3. Login with:

- Username: admin
- Password: (the one you set earlier)



```
lab1002@lab1002-HP-280-G3-MT:~$ sudo usermod -a -G nagcmd nagios
lab1002@lab1002-HP-280-G3-MT:~$ sudo usermod -a -G nagcmd www-data
lab1002@lab1002-HP-280-G3-MT:~$ ip a | grep inet
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
    inet 192.168.4.104/24 brd 192.168.4.255 scope global dynamic noprefixroute e
np4s0
    inet6 fe80::59a0:c15b:f605:a620/64 scope link noprefixroute
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
lab1002@lab1002-HP-280-G3-MT:~$ ls /etc/apache2/sites-enabled/nagios.conf
/etc/apache2/sites-enabled/nagios.conf
lab1002@lab1002-HP-280-G3-MT:~$ sudo systemctl reload apache2
lab1002@lab1002-HP-280-G3-MT:~$ ls /usr/local/nagios/share
angularjs              includes            media
bootstrap-3.3.7        index.php           robots.txt
config.inc.php         infobox.html        side.php
contexthelp            js                  spin
d3                     jsonquery.html      ssi
docs                   locale              stylesheets
graph-header.html      main.php            trends-form.html
histogram-form.html    map-directive.html  trends-graph.html
histogram-graph.html   map-form.html       trends-host-yaxis.html
```

```
ab1002@lab1002-HP-280-G3-MT:~$ cat /usr/local/nagios/etc/htpasswd.users
dmin:$apr1$2XweygvH$ReiuYPl/sfqlhcYq9xuMn/
ab1002@lab1002-HP-280-G3-MT:~$ sudo a2enmod rewrite cgi
odule rewrite already enabled
odule cgi already enabled
ab1002@lab1002-HP-280-G3-MT:~$ sudo systemctl restart apache2
ab1002@lab1002-HP-280-G3-MT:~$ sudo htpasswd /usr/local/nagios/etc/htpasswd.use
s admin
ew password:
e-type new password:
pdating password for user admin
ab1002@lab1002-HP-280-G3-MT:~$ sudo systemctl restart apache2
```



CONCLUSION: This experiment successfully demonstrates the complete process of installing and configuring Nagios on an Ubuntu system. It provides a thorough understanding of how Nagios monitors the health and performance of servers, services, and network devices. The experiment covers creating necessary users and groups, compiling and installing Nagios Core, setting up web interface authentication, installing essential plugins, enabling required Apache modules, and starting the Nagios service. By completing these steps, one can effectively use Nagios to receive real-time alerts, view system status through a web interface, and ensure proactive management of IT infrastructure. This hands-on experience highlights the importance of monitoring tools in maintaining system reliability, minimizing downtime, and supporting efficient operations in IT environments.

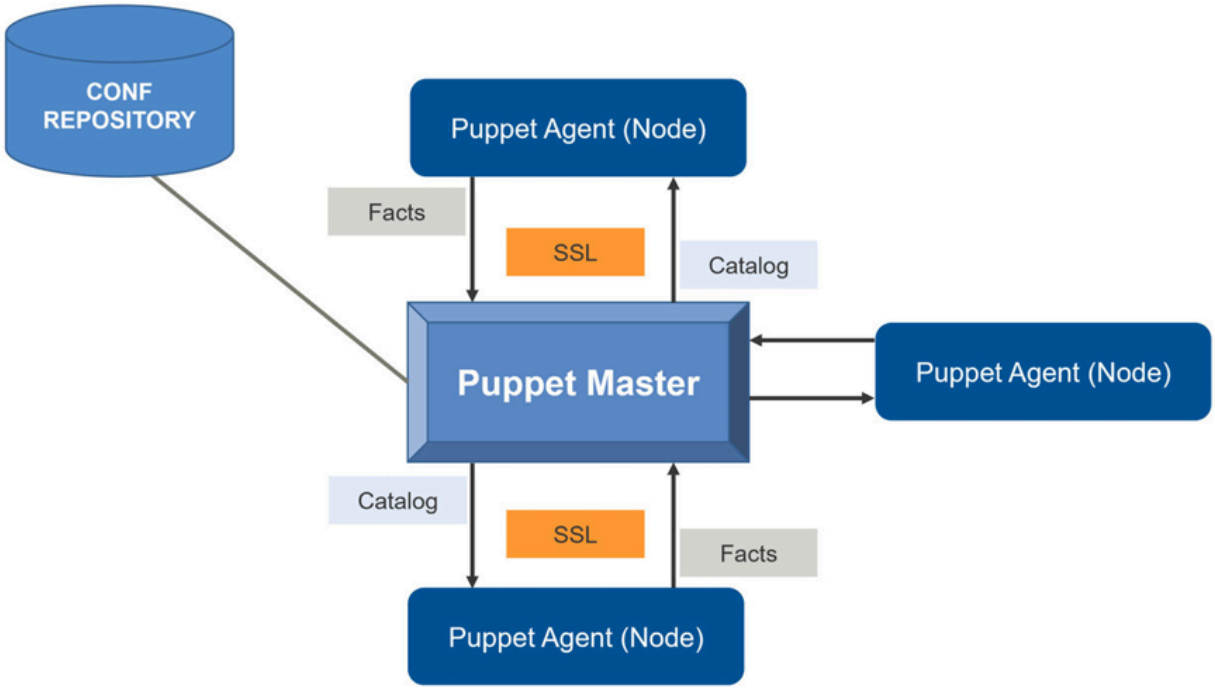
LO MAPPING: LO1, LO5

ASSIGNMENT- 10

(Puppet Tools)

AIM: To study puppet tools.

THEORY:



Puppet is an open-source configuration management and automation tool used to manage servers, infrastructure, and application deployments efficiently. It is widely adopted in DevOps practices to automate repetitive tasks, enforce system configurations, and ensure consistency across environments. Puppet helps organizations scale infrastructure, reduce manual errors, and improve system reliability. Puppet uses a declarative approach, where administrators define the *desired state* of the infrastructure, rather than specifying

step-by-step instructions. Puppet ensures that all managed systems converge to this desired state automatically.

For example, you can specify that a certain package must be installed, a service should be running, or a configuration file must have specific content, and Puppet will enforce these rules across multiple servers.

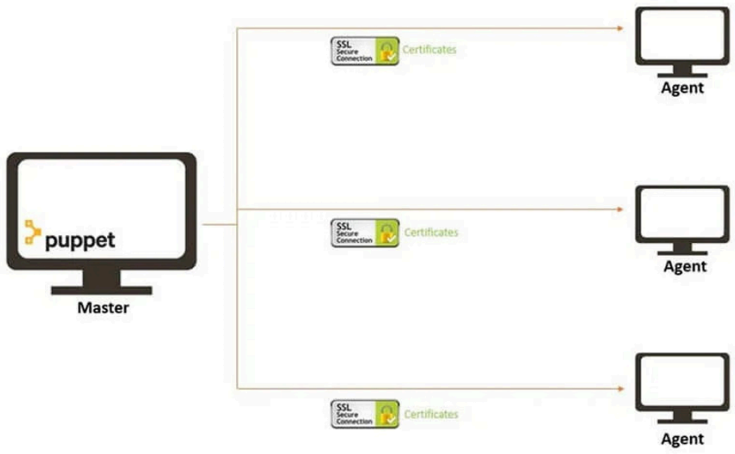
Architecture of Puppet:

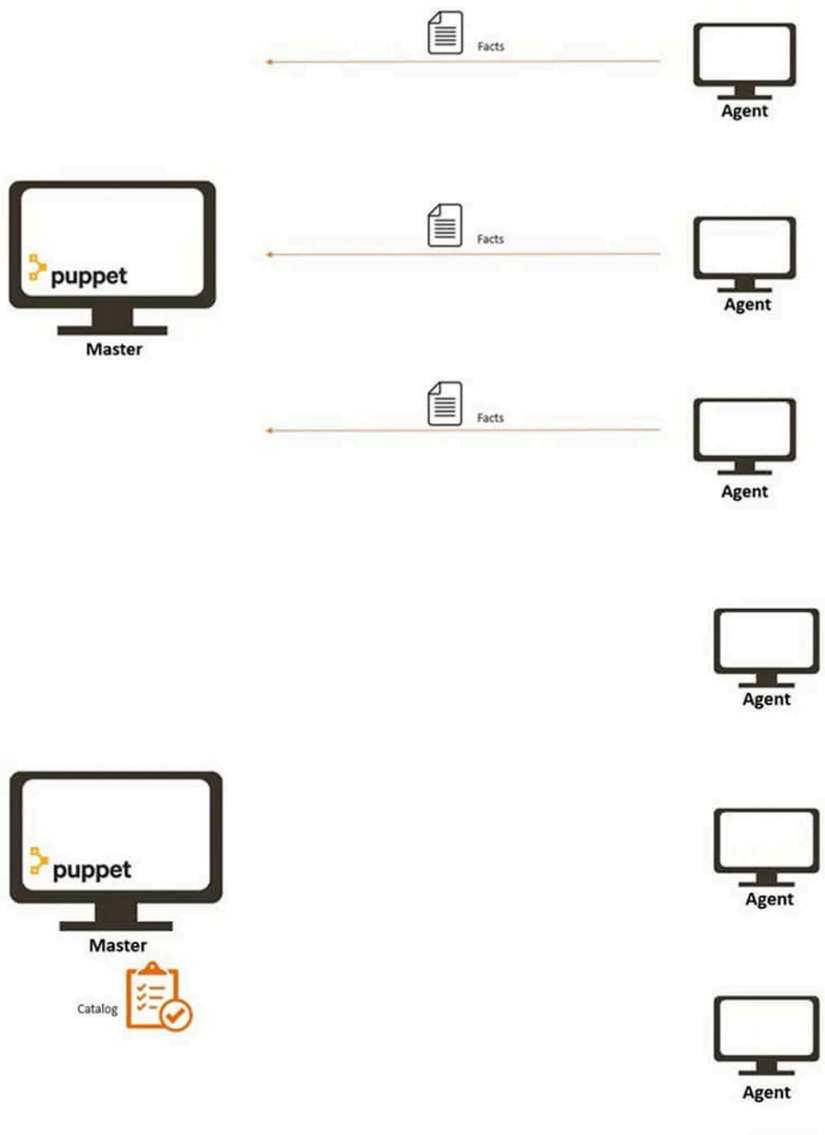
Puppet typically uses a client-server architecture, consisting of the following components:

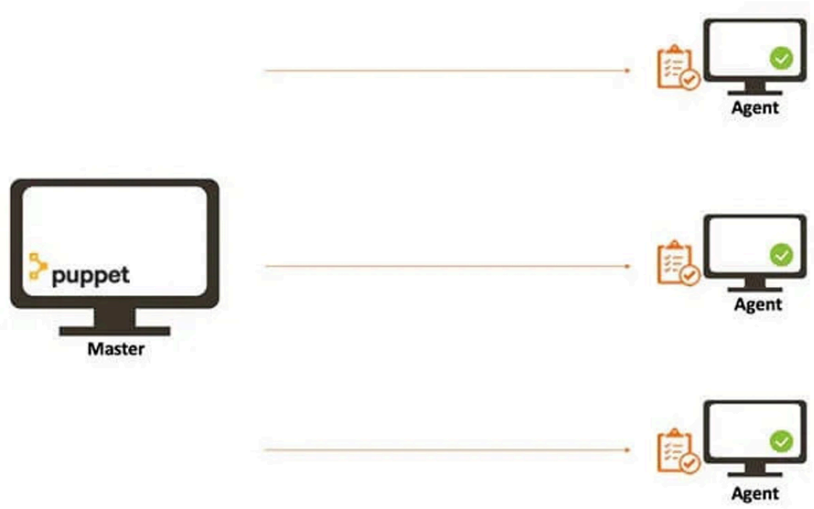
Component	Description
Puppet Master	Central server that stores configuration data and manifests; compiles catalogs for nodes.
Puppet Agent	Installed on managed nodes; applies configurations received from the Puppet Master and reports status.
Manifests	Files written in Puppet DSL that define the desired state of resources.
Modules	Collections of manifests and related files (templates, scripts) organized by functionality for reuse.
Facter	Collects system information (facts) like OS, IP address, and memory for dynamic configurations.
Catalog	Document compiled by Puppet Master containing the desired state for each resource on a node.

How Puppet Works:

1. **Defining Desired State:**
    - The administrator writes **manifests** or **modules** that describe the desired state of servers and applications.
2. **Fact Collection:**
    - Puppet Agents collect system information (facts) using **Facter**, which helps make configurations dynamic.
3. **Catalog Compilation:**
    - Puppet Master compiles a **catalog** based on the manifests and facts, detailing the desired state for each node.
4. **Catalog Application:**
    - Puppet Agents apply the catalog to the managed nodes, ensuring each resource matches the desired state.
5. **Reporting:**
    - After applying configurations, Puppet Agents send reports to the Puppet Master showing which changes were applied and if any errors occurred.
6. **Continuous Enforcement:**
    - Puppet periodically checks the system state to detect **drift** (changes made outside Puppet) and automatically corrects it to maintain consistency.




Facts
Agent
puppet
Master
Facts
Agent
Facts
Agent
Agent
puppet
Master
Catalog
Agent
Agent



Key Features of Puppet:

- **Automation:** Automates repetitive configuration and deployment tasks.
- **Consistency:** Ensures all systems converge to the desired state.
- **Scalability:** Efficiently manages hundreds or thousands of nodes.
- **Declarative Language:** Simplifies writing configuration code.
- **Cross-Platform Support:** Works with Linux, Windows, and cloud environments.
- **Reporting and Compliance:** Provides detailed reports on system state and changes.
- **Integration with DevOps Tools:** Works with CI/CD pipelines, Docker, Kubernetes, and cloud platforms.

Common Puppet Use Cases:

1. **System Configuration Management:**
    - Installing and managing software packages.
    - Configuring users, groups, and permissions.
    - Managing services, cron jobs, and scheduled tasks.
2. **Application Deployment:**

    - Automates deployment of applications across multiple servers.
    - Ensures consistent configuration in staging and production.
3. **Infrastructure as Code (IaC):**
    - Defines infrastructure in code to improve repeatability and reduce human error.
4. **Compliance and Security:**
    - Ensures security policies are consistently applied.
    - Detects configuration drift and corrects deviations automatically.

Benefits of Using Puppet:

- Reduces manual effort and errors in managing infrastructure.
- Enables faster and more reliable deployment of applications and services.
- Improves collaboration between development and operations teams.
- Ensures predictable and auditable changes to infrastructure.
- Supports hybrid cloud and multi-platform environments.

CONCLUSION: This experiment successfully provides a comprehensive understanding of Puppet tools, including its architecture, working mechanism, features, and practical applications, highlighting how it automates configuration management, enforces system consistency, and supports scalable and reliable DevOps practices.

LO MAPPING: LO1, LO6

AYUSH SHARMA
T22-117

(A) Friday 03/10/25

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

DEV OPS WRITTEN ASSIGNMENT-1

Q1. What is test automation or automation testing? State its advantages.

Ans. Automated testing means using special software for tasks that people usually do when checking & testing a software product. Nowadays many software projects use automation testing from start to end especially agile and devops. This means the engineering teams run tests automatically with the help of software tools. It will help the testing team to make the process faster. Continuous delivery (CD) quickly sends new code to users.

It is a ~~software testing~~ technique where the testers writes scripts independently and uses suitable software automation tools to test software. It is an automation process of a manual process. It allows for executing repetitive tasks without the ~~the~~ intervention of a manual testers.

→ It is used to automate the testing tasks that are difficult to perform manually.

→ Automation tests can be run at any time of the day as they use ~~simplified~~ scripted sequences to examine the software.

→ Automation tests can also enter test data. Compare ~~the~~ the expected result with the actual results and generate detailed test reports.

→ It is possible to record the test suite and replay it when required.

- Advantages of automation testing:

① Increase speed & efficiency:- Automated tests execute much faster than manual tests, significantly reducing the time required to complete testing cycles.

② Improved accuracy & reliability :- Automation eliminates the potential for human error that often occurs during repetitive manual tasks, ensuring consistent & precise test execution every time.

③ Broader test coverage: Automation allows for the execution of thousands of complex test cases that are impractical or impossible to run manually, leading to more comprehensive software quality assurance.

④ Cost & time savings :- While there's an initial investment in setup automation reduces ongoing labour costs and identifies bugs earlier in the development process which is significantly cheaper than fixing them later.

⑤ 24/7 test execution :- Automated tests can be run continuously without human intervention, enabling round the clock testing & feedback.

Q2. What is X path? Explain difference between single slash & double slash test in X path.

Ans. X path stands for XML path language. It is an expression language that is used to query or transform. We use it to traverse among elements and attributes in an XML document. The world wide web (WWW) consortium (W3C) declared x path in 1999.

X path is used for:

- Query or transform XML document.
- Traverse elements, attributes and text through in XML document.
- Find particular elements or attributes with matching pattern.

- Extract information from any part of an XML document.

- SYNTAX : //tagname [@attribute='value']

X path expression:

SYMBOL	DESCRIPTION
//	Select nodes in the document from the current node that match selection no matter where they are.
/	Select the root node.
tagname	tagname of the current node.
@	Select the attribute
attribute	attribute name of the node
value	value of the attribute

- Difference between '/' & '//':

PARAMETER	SINGLE SLASH /	DOUBLE SLASH //
• Meaning	select nodes directly from the root nodes.	select nodes from anywhere in the doc.
• Nature of path	Absolute path	Relative path.
• Search scope	Search only at one exact level starting from the root node.	Search recursively at all levels in the doc.
• Flexibility	Rigid - breaks if hierarchy breaks.	Flexible - works even if structure changes
• Performance	Faster	Slightly slower.

AYUSH SHARMA
T22-117

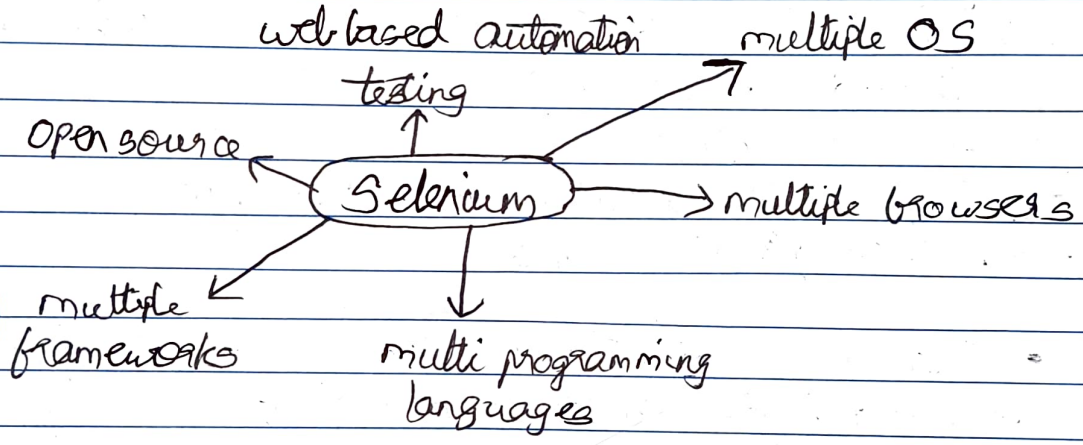
(A) Graded 09/10/25

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

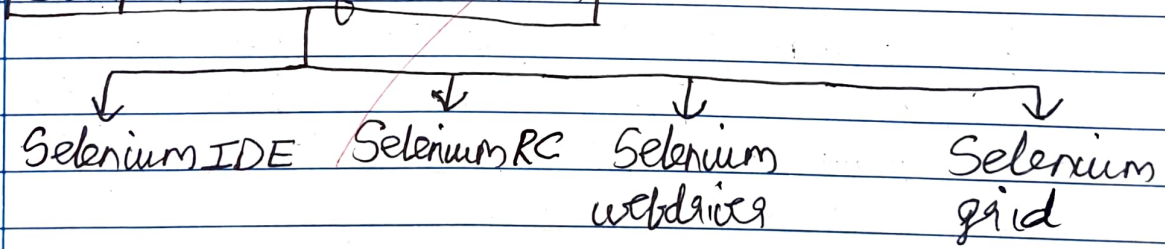
DEV OPS WRITTEN ASSIGNMENT-2

Q1. What is selenium? What are its components?

Ans. Selenium is a widely used tool for testing web based application that checks if they're working as expected. It is a primary preference amongst testers for cross browsers testing and is viewed as one of the most reliable system for wide application automation evaluation



- Components of selenium



- Selenium IDE :- Selenium IDE serves as an innovative toolkit for web testing allowing users to record interactions with web applications. Selenium IDE was initially created by Shinya Kasatani in 2006, it ~~also~~ helps to simplify testing process.

→ Record :- with selenium IDE, users can record how they use web application.

→ Playback :- Selenium IDE automatically repeats what you recorded earlier.

→ Browser check :- Selenium IDE works on various browsers for testing.

- Selenium RC :- Selenium remote control (RC) was one of the earliest selenium tools, providing web driver. It allowed testers to write automated web applications tests in various programming languages like C#, java, Python, etc. Webdriver was considered a better choice over selenium RC for the fol. reasons:
  → Improved API.
  → Support for modern web technologies
  → Better performance.

- Selenium web driver :- It is a robust open source framework for automating web browsers, primarily aimed at easing the testing and verification of web applications. It is an important part of selenium suite, webdriver offers a programming interface to interact with web browsers, allowing developers and testers to automate browser actions seamlessly.

- Selenium grid :- Selenium grid is a server that allows tests to use web browser instances running in remote machines with selenium grid, one server acts a a hub. Tests contact hub to obtain access to browser instances.

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Q2. What makes selenium such a widely used testing tool. Give reasons as to why it is advised to select selenium as a testing tool for web applications or systems

Ans. Selenium is a widely used testing tool for the following reasons:

- Open source & free of cost.

→ It is an open source project meaning anyone can use it without purchasing license.
→ Unlike commercial testing tools, it has zero cost barriers which make it highly attractive to startups & enterprises alike.
→ Large community contributions constantly improve selenium, keeping it updated with new web technologies.

- Web browser compatibility

→ Web applications must work seamlessly across browsers like Opera, chrome, safari, firefox, etc.
→ Selenium supports all major browsers & provides the capability to run the same test case across different browsers.
→ This eliminates the need of writing browser specific scripts.

- Cross platform support

→ Selenium tests can run on windows, linux, unix & macOS.
→ This flexibility ensures that developers and testers can work in their own environments while maintaining uniformity of results.

→ With Selenium grid, tests can be distributed across multiple environments simultaneously.

- Realistic web interaction:

→ Selenium mimics real user interactions:
  - Clicking, typing, scrolling, drag-drop
  - Handling pop-ups, alerts, checkboxes & dropdowns.

→ This enables that the test cases are as close to the real world usage as possible improving test accuracy.