THADOMAL SHAHANI TSEC ENGINEERING COLLEGE T22-94

Experiment - 1

SEPM ~~ASSIGNMENT~~ 2

Aim : To understand DevOps : Principles, practices DevOps Engineer role and responsibility

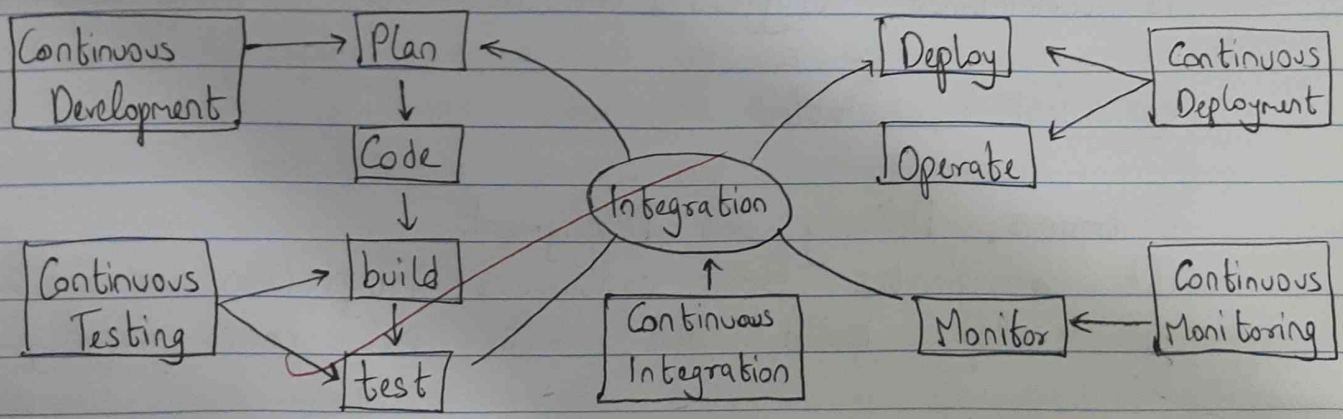
What is DevOps ?

DevOps is a collaborative approach where teams work together to build and deliver secure software efficiently. It combines software development (dev) and operations (ops) to decide how to accelerate through automations, collaboration, fast feedback and iterative improvement.

Core Principle :

1. Develop and test in production like environments
2. Deploy builds frequently
3. Continuously validate operational quality.

DevOps practices :



Continuous Development
This is the phase that involves planning and versioning and managing build of the software applications functionality

eg. git , github, maven, etc.

Continuous Testing
It is executing automated tests , continuously and repeatedly against the code base and the various deployment environments. It is a software testing methodology which focuses on achieving continuous quality and improvement

eg. Bamboo ,

Continuous Integration
It refers to the build and unit testing stages of the software releases process. Every revision that is committed triggers an automated build and test

eg. Jenkins , circled

Continuous Delivery and Deployment
It originates from continuous integration , a method to develop, build and test new code rapidly

with automation so that only code that is known to be good becomes part of a software product.

Infrastructure manggement
Without automation, building and maintaining large scale modern IT systems can be a resource-intensive undertaking and can lead to increased risk due to manual error configuration and resource management is an automated method for maintaining computer systems and software in an unknown consistent state.

Configuration Management
Infrastructure as code is the practise of describing all software runtime environment and networking setting used parameters in simple textual format that can be stored in your version control system and version on request.

Microservice Architecture
Dockers is a tool designed to make it easier to create, deploy and user application by using containers. Containers allows a developer to package up an application with all of the parts it needs such as libraries used other dependencies and deploy it as one package.

Cloud Base DevOps
DevOps automation is becoming cloud centric. Most public and private cloud computing provides support. DevOps systematically

on their platform, including continuous integration and continuous development tools.
eg. ~~Azo~~ Amazon Web Service, amazon lambda, google cloud

DevOps Engineer Roles:
- Implement development, testing and automation tools
- Set up infrastructure and tools
- Code review and responsibilities
- Bug fixing and trouble shooting
- Build and maintain CI/CD pipelines
- Security Implementation and monitoring.

Management Responsibilities
- Understand customer requirements and API's
- Plan team initiatives and activities.
- Manage Stakeholders
- Define development and Operational processes
- Coordinate team communication
- Monitor Customer experience
- Provide Periodic progress reports.
- Mentor Team Members.

A devOps engineer manages a company's IT infrastructure, bridging development, and operation.

Salif Shaikh
T22 – 2201094
TE – AI&DS

EXPERIMENT 2

TO UNDERSTAND VERSION CONTROL SYSTEM / SOURCE CODE MANAGEMENT, INSTALL GIT AND CREATE A GITHUB ACCOUNT

THEORY:

What Is Version Control?

A version control system, also known as source control or revision control, tracks changes made to files over time, allowing users to retrieve specific versions when needed. While it can be applied in various scenarios, it is most commonly used in software development.

What is Git?

- Git is a free and open-source distributed version control system designed to handle projects of all sizes with speed and efficiency.
- It enables collaborative software development, allowing multiple developers to modify source code while tracking changes.
- Git was initially designed and developed by Linus Torvalds for Linux kernel development in 2005.
- Every Git working directory is a full-fledged repository with complete history and version tracking, independent of network access or a central server.
- Git enables teams to work on the same files simultaneously, resolving conflicts and maintaining version history.

What Is GitHub?

GitHub is a web-based platform that hosts software development projects and integrates with Git for version control. It provides a user-friendly interface and collaborative tools to enhance project management.

- It allows developers to create, manage, and access repositories remotely.
- GitHub is essentially a collection of repositories that store project files and enable seamless collaboration.

Use of Version Control Software

- Allows users to track different versions of a project and revert changes when necessary.
- Helps in comparing versions, debugging, and recovering lost data.

Salif Shaikh
T22 – 2201094
TE – AI&DS

- Saves changes as patch files that can be applied to previous versions.
- Stores all versions on a central server, where developers can check out and update their work.

Use Cases of GitHub

1. Version Control – Allows developers to revert to previous versions when changes affect the project negatively.
2. Collaboration & Code Review – Enables teams to review each other's code, improving productivity and efficiency.
3. Issue Tracking – Developers can assign issues to team members and track progress.
4. Open-Source Development – GitHub is widely used for open-source projects, allowing public contributions.

Characteristics of Git

1. Strong Support for Non-Linear Development Supports rapid branching and merging.
   - Changes are merged frequently.
   - Lightweight branches make development faster.
2. Distributed Development
   - Every developer gets a local copy of the repository.
   - Changes can be merged efficiently between repositories.
3. Compatibility with Existing Systems
   - Git supports CVS server emulation, allowing existing CVS clients to access Git repositories.
4. Efficient Handling of Large Projects
   - Faster and more scalable than other version control systems.
   - Fetching data from a local repository is quicker than from a remote server.
5. Data Assurance
   - Git stores history in a way that ensures integrity—old versions cannot be changed unnoticed.
6. Automatic Garbage Collection
   - Git performs automatic garbage collection when enough loose objects are created.
   - Can be explicitly triggered using git gc --prune.
7. Periodic Explicit Object Packing
   - Git stores objects as separate files, later compressed into a packfile.
   - This improves performance but is computationally expensive.

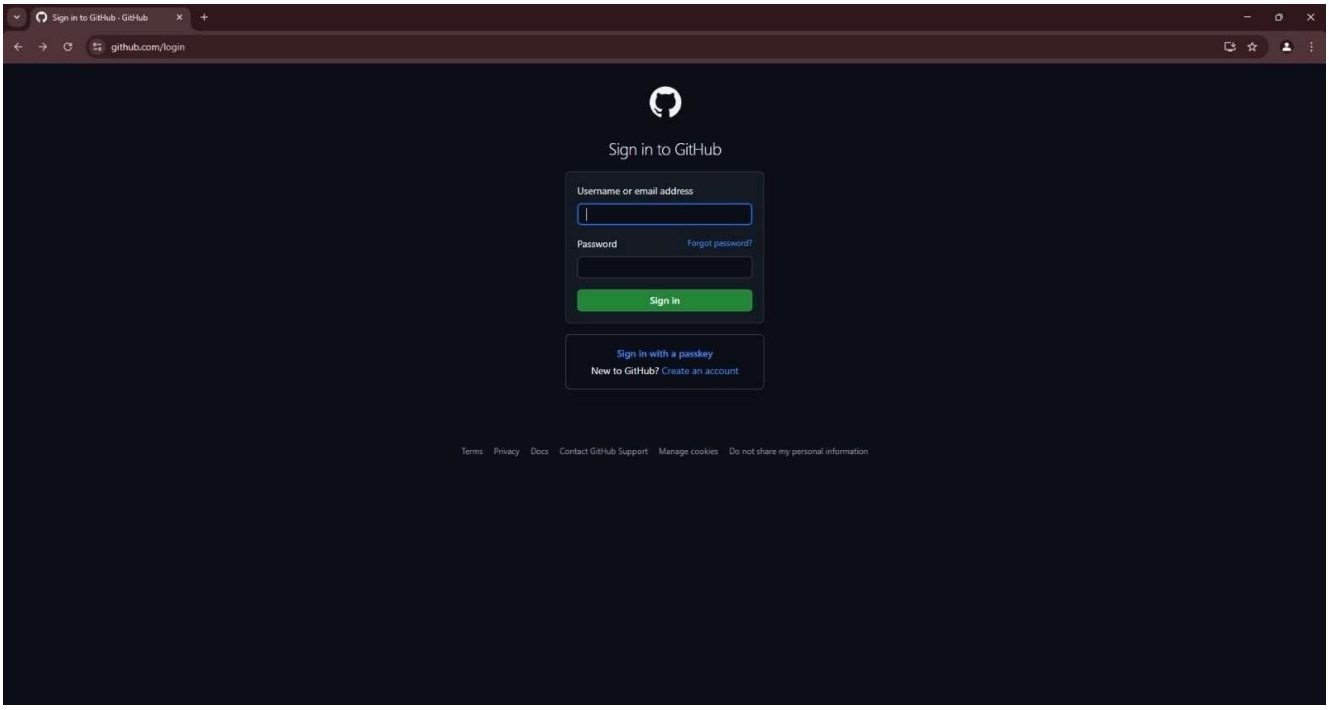
How Does Git Work?

- A Git repository is a key-value store where all objects are indexed by their SHA-1 hash.
- Objects include commits, files, tags, and tree nodes.
- Git works by taking snapshots of files at different points in time.

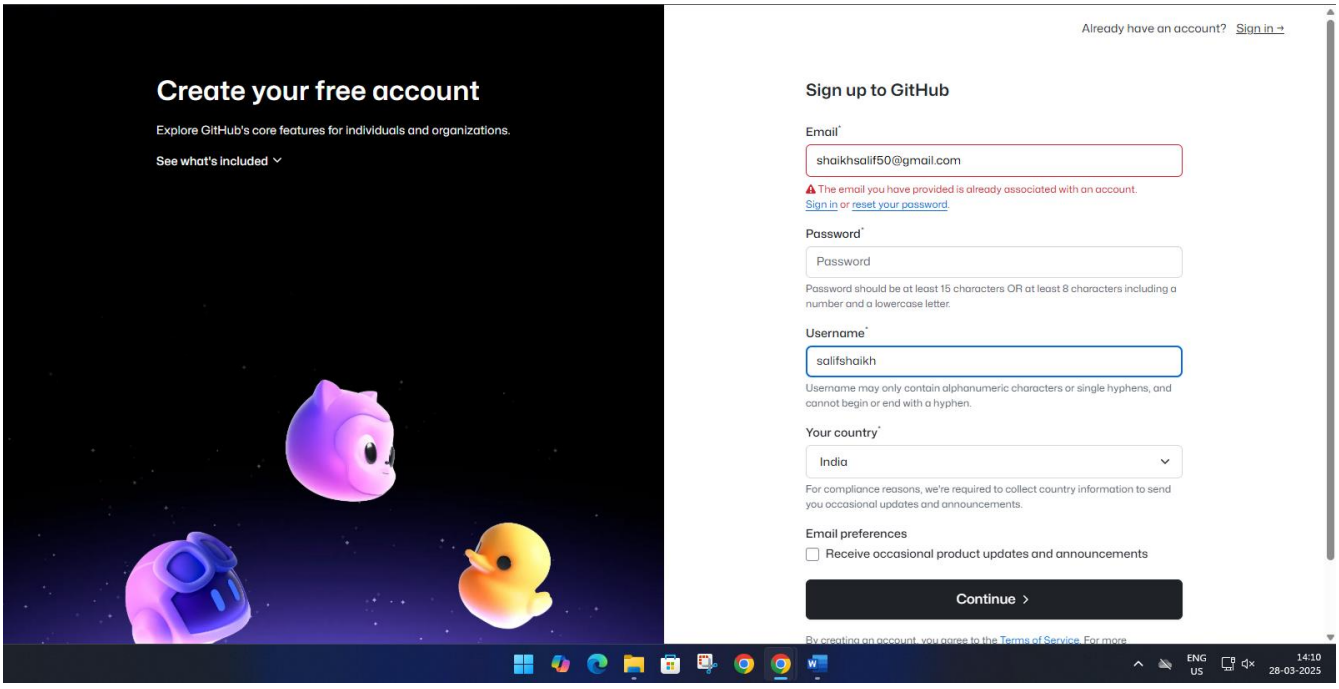
Steps to Creating a GitHub Account

1. Open a web browser and go to GitHub Signup.

Salif Shaikh
T22 – 2201094
TE – AI&DS

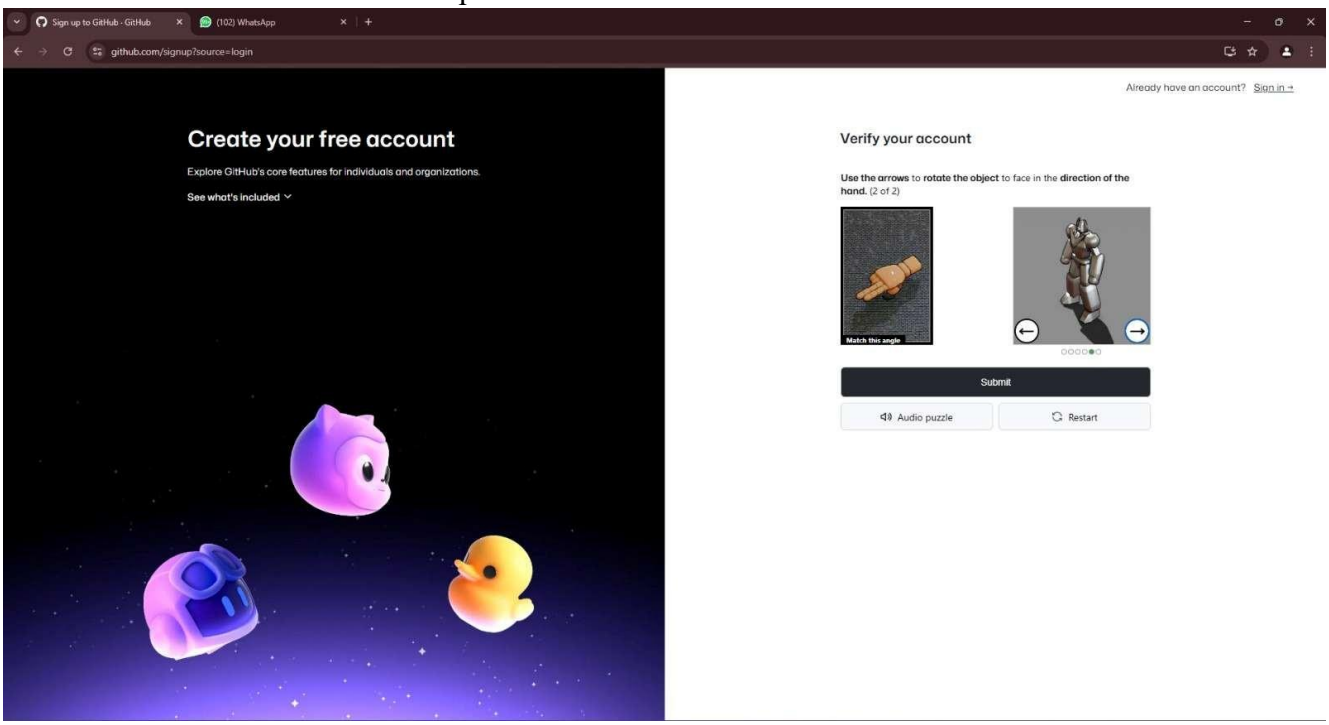


2. Enter a unique username. Provide a valid email address. Create a strong password.

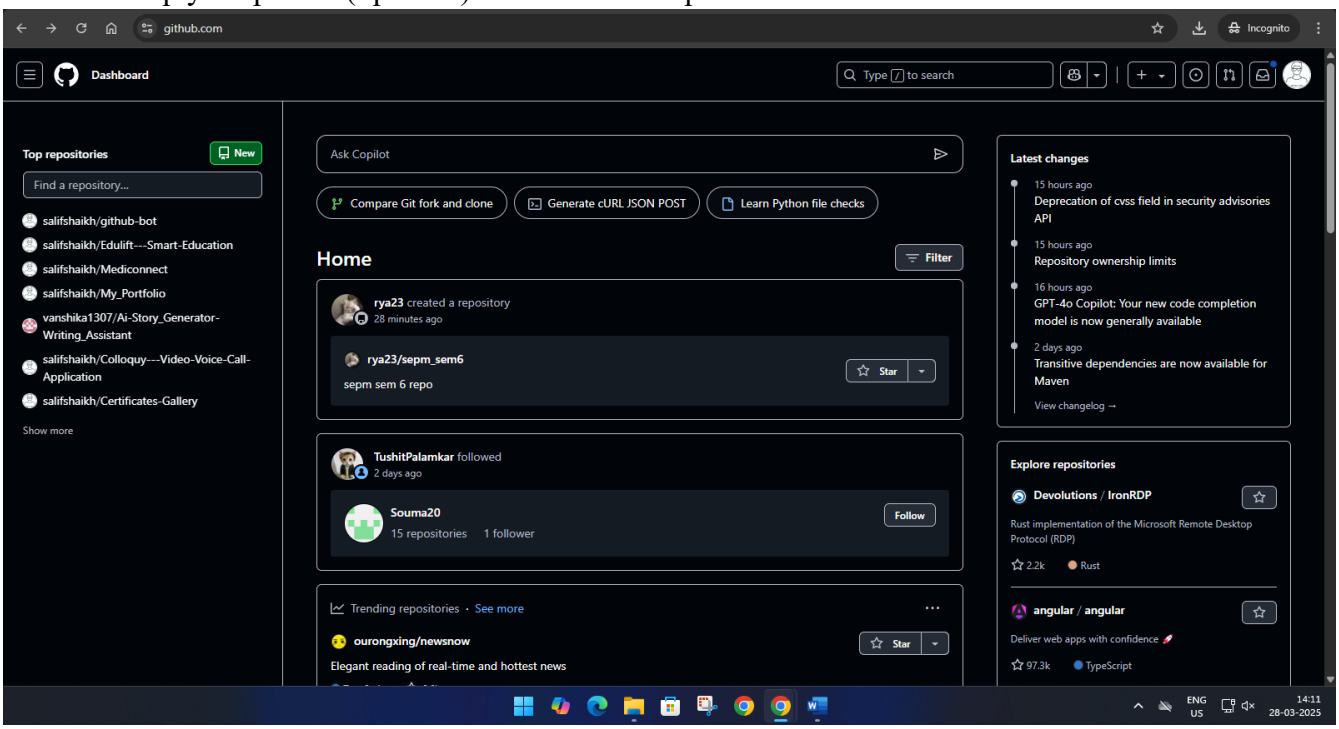


Vaibhav Worlikar
T23 – 2201124
TE – AI&DS

3. Click "Create account". Complete the CAPTCHA verification.



4. Choose your email preferences and click "Continue".
5. Verify your email by clicking the link sent to your registered email address.
6. Set up your profile (optional) and Select Free plan

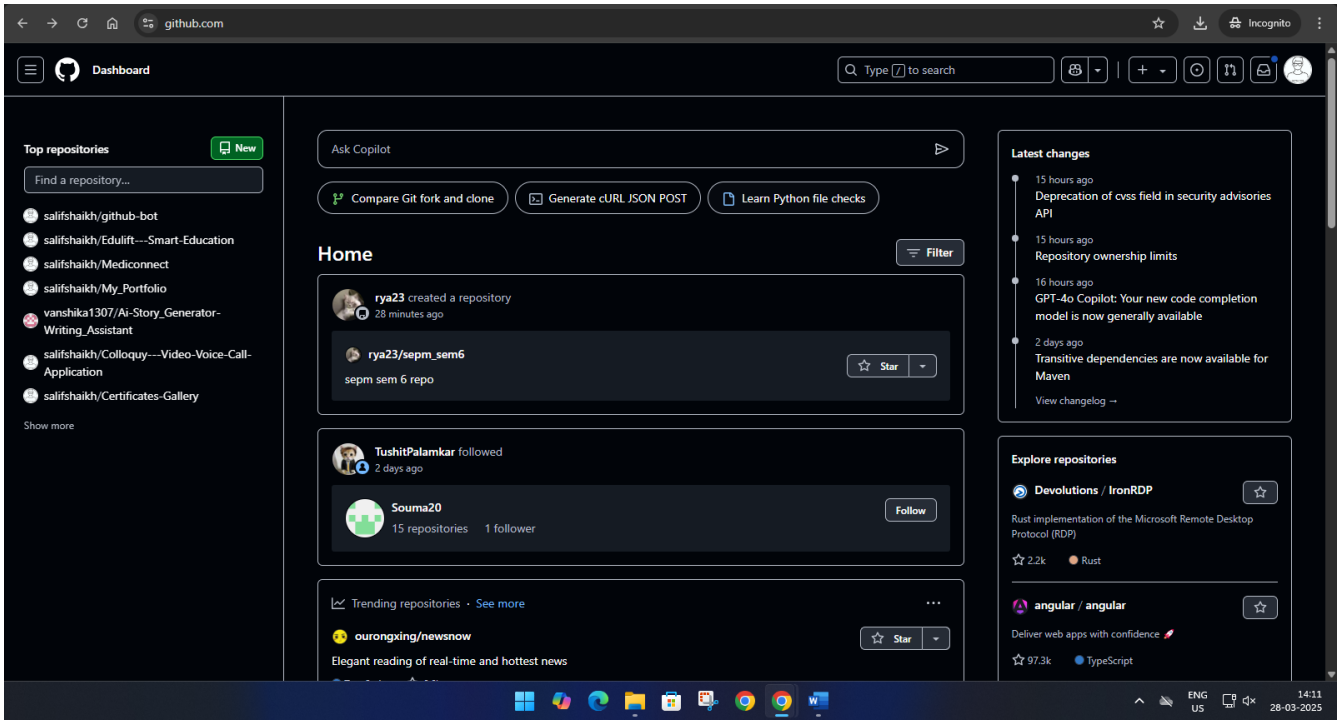


7. Your GitHub account is now ready to use!

Vaibhav Worlikar
T23 – 2201124
TE – AI&DS

Steps to Create a Repository on GitHub

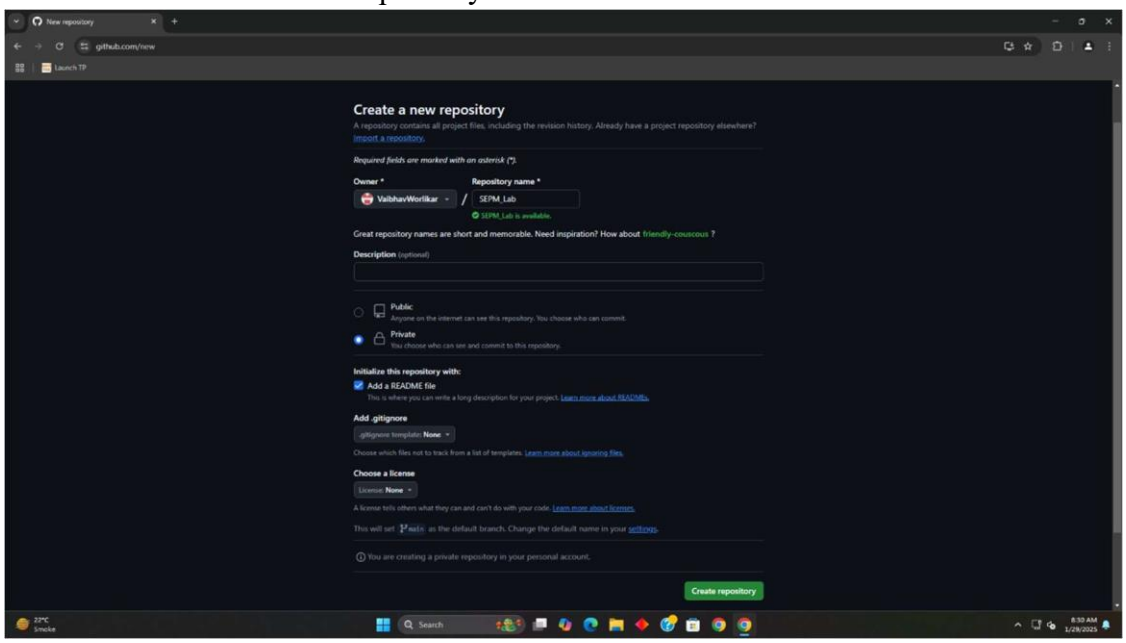
1. Log in to GitHub:
    - Go to GitHub and sign in to your account.
2. Navigate to Repositories:
    - Click on your profile icon (top-right corner) and select "Your repositories" from the dropdown.
    - Alternatively, click on the "+" icon in the top-right corner and select "New repository".



3. Create a New Repository:
    - Enter a repository name (e.g., my-project). o Optionally, add a description.
    - Choose the visibility:
        - Public (anyone can see your repository).
        - Private (only you and collaborators can access it).
4. Initialize the Repository (Optional):
    - You can check "Add a README file" to include a basic introduction. o Optionally, add a .gitignore file for excluding certain files from tracking.
    - Choose a license if needed.

Vaibhav Worlikar
T23 – 2201124
TE – AI&DS

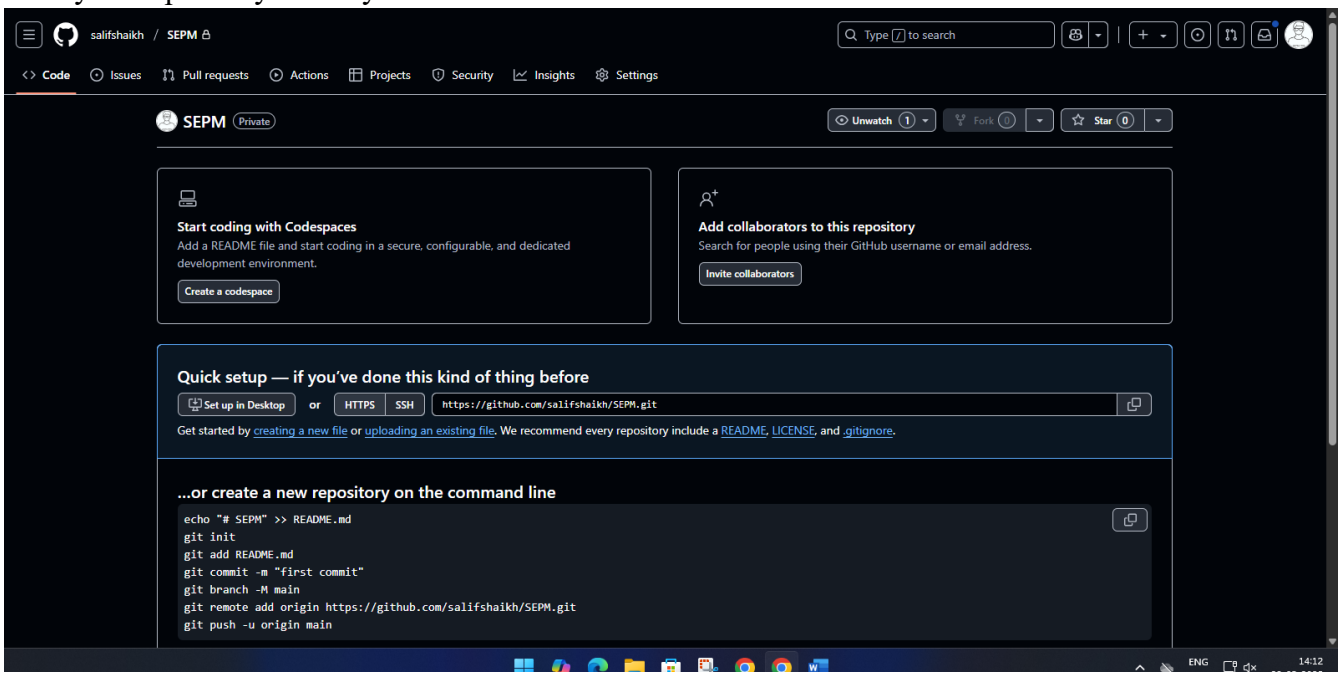
5. Create the Repository:
    - Click the "Create repository" button.



6. Set Up Locally (Optional):
    - Copy the repository URL and use Git to clone it:
    - git clone https://github.com/your-username/your-repository.git o Navigate to the folder and start working on your project!

Vaibhav Worlikar
T23 – 2201124
TE – AI&DS

Now your repository is ready to use!



Salif Shaikh
T22 – 2201094

EXPERIMENT 3

TO PERFORM VARIOUS GIT OPERATIONS ON LOCAL AND REMOTE REPOSITORIES USING GIT CHEAT SHEET

Theory:

Git is a distributed version control system that allows developers to track changes, collaborate, and manage source code efficiently. Git provides numerous commands to handle local and remote repositories.

1. Setting Up Git

Before performing Git operations, configure Git with your details:

git config --global user.name "Your Name"

git config --global user.email "your.email@example.com" Verify the configuration:

git config --list

2. Initializing a Git Repository To create a new Git repository: git init

This initializes a new repository in the current directory.

3. Cloning a Repository To clone a remote repository: git clone <repository_url> Example:

git clone https://github.com/your-username/repository.git

4. Staging and Committing Changes

- To check the status of the working directory:
- git status
- To add files to the staging area:
- git add <file_name> or to add all changes:

git add .

- To commit changes with a message:
- git commit -m "Your commit message" 5. Viewing Commit History To view commit logs:

git log

For a compact version:

git log --oneline

6. Branching in Git

- To create a new branch:
- git branch <branch_name>
- To switch to another branch: git checkout <branch_name>

Salif Shaikh
T22 – 2201094

- To create and switch to a new branch simultaneously:
- git checkout -b <branch_name> To view all branches:
- git branch

7. Merging Branches
   - First, switch to the main branch:
   - git checkout main
   - Merge a branch into the main branch:
   - git merge <branch_name> 8. Pushing Changes to Remote Repository To push changes to GitHub: git push origin <branch_name> If pushing for the first time:
   - git push --set-upstream origin <branch_name>

9. Pulling Changes from Remote Repository To fetch and merge changes from a remote repository:

git pull origin <branch_name>

10. Handling Merge Conflicts If a merge conflict occurs:

1. Open conflicting files and resolve issues manually.
2. Add resolved files to the staging area:
3. git add <file_name>
4. Commit the resolved changes:
5. git commit -m "Resolved merge conflict"

11. Undoing Changes
    - To undo changes before staging:
    - git checkout -- <file_name> To unstage a file:
    - git reset HEAD <file_name>
    - To revert the last commit:
    - git revert HEAD

12. Deleting a Branch
    - To delete a local branch: git branch -d <branch_name> To delete a remote branch:
    - git push origin --delete <branch_name>

13. Creating and Using a .gitignore File

A .gitignore file is used to ignore specific files or directories:

echo "node_modules/" >> .gitignore
git add .gitignore git commit -m
"Added .gitignore file"

14. Checking Differences in Files

Salif Shaikh
T22 – 2201094

- To compare working directory changes:
- git diff
- To compare staged changes:
- git diff --staged

15. Stashing Changes

To temporarily save uncommitted changes:

git stash

To apply the stashed changes:

git stash apply

Output:

```
Microsoft Windows [Version 10.0.26100.3476]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Lab805_4>git config --global user.name "salifshaikh"

C:\Users\Lab805_4>git config --global user.name shaikhsalif50@gmail.com

C:\Users\Lab805_4>git config --global --list
core.editor="C:\Users\Lab805_4\AppData\Local\Programs\Microsoft VS Code\bin\code" --w
ait
user.name=shaikhsalif50@gmail.com

C:\Users\Lab805_4>cat~/.gitconfig
'cat~' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\Lab805_4>git-demo-project1234
'git-demo-project1234' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\Lab805_4>git init
Initialized empty Git repository in C:/Users/Lab805_4/.git/

C:\Users\Lab805_4>Is -a
'Is' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\Lab805_4>
```

Salif Shaikh
T22 – 2201094

Conclusion

This experiment demonstrated various Git operations, including repository initialization, branching, merging, pushing, pulling, and resolving conflicts. These commands help in efficient version control and collaboration in software development projects.

Salif Shaikh
T22 – 2201094

EXPERIMENT 4

TO UNDERSTAND CONTINUOUS INTEGRATION, INSTALL AND CONFIGURE JENKINS WITH MAVEN / ANT / GRADLE TO SETUP A BUILD JOB

Theory:

1. Introduction to Continuous Integration (CI)

Continuous Integration (CI) is a software development practice where developers frequently integrate code changes into a shared repository. Each integration is automatically verified through a build and test process to detect errors early.

1.1 Benefits of Continuous Integration Early detection of defects.

- Faster debugging and deployment.
- Automated testing improves code quality.
- Reduces integration issues and conflicts.

1.2 Tools for Continuous Integration

- Jenkins – An open-source automation server.
- Maven, Gradle, Ant – Build tools to compile, test, and package applications.

---

2. Installing and Configuring Jenkins

2.1 Installing Jenkins

1. Download Jenkins:
    - Get Jenkins from Jenkins official site.
    - Install using the appropriate installer for your OS.
    - Start Jenkins using:
        - Windows: Run jenkins.exe
        - Linux/Mac: Run java -jar jenkins.war --httpPort=8080
2. Initial Setup:
    - Access Jenkins at http://localhost:8080/.
    - Enter the Administrator password (found in jenkins_home/secrets/initialAdminPassword).
    - Install suggested plugins and create an admin user.

2.2 Installing Required Plugins

- Navigate to Manage Jenkins > Plugin Manager.

Salif Shaikh
T22 – 2201094

- Install the following plugins:
    - Maven Integration Plugin (for Maven builds).
    - Gradle Plugin (for Gradle support).
    - Pipeline Plugin (for defining CI/CD pipelines).

---

3. Configuring Maven, Gradle, or Ant in Jenkins

3.1 Configuring Maven in Jenkins

1. Install Maven on the system (or use Jenkins' built-in Maven).
2. Go to Manage Jenkins > Global Tool Configuration.
3. Under Maven, click Add Maven, set a name (e.g., Maven-3.8.1), and provide the installation path.

3.2 Configuring Gradle in Jenkins

1. Install Gradle on your system.
2. In Jenkins, go to Global Tool Configuration.
3. Under Gradle, click Add Gradle, enter a name, and provide the installation path.

3.3 Configuring Ant in Jenkins

1. Install Apache Ant and set the environment variable (ANT_HOME).
2. In Global Tool Configuration, add Ant with its installation path.

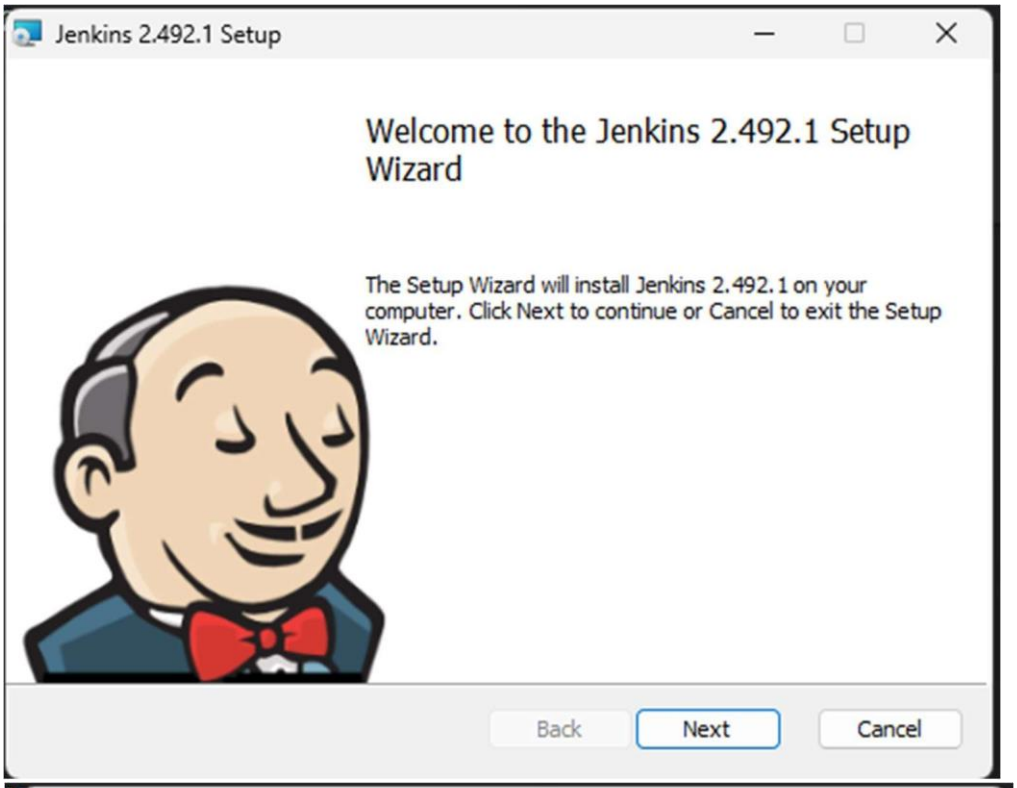
---

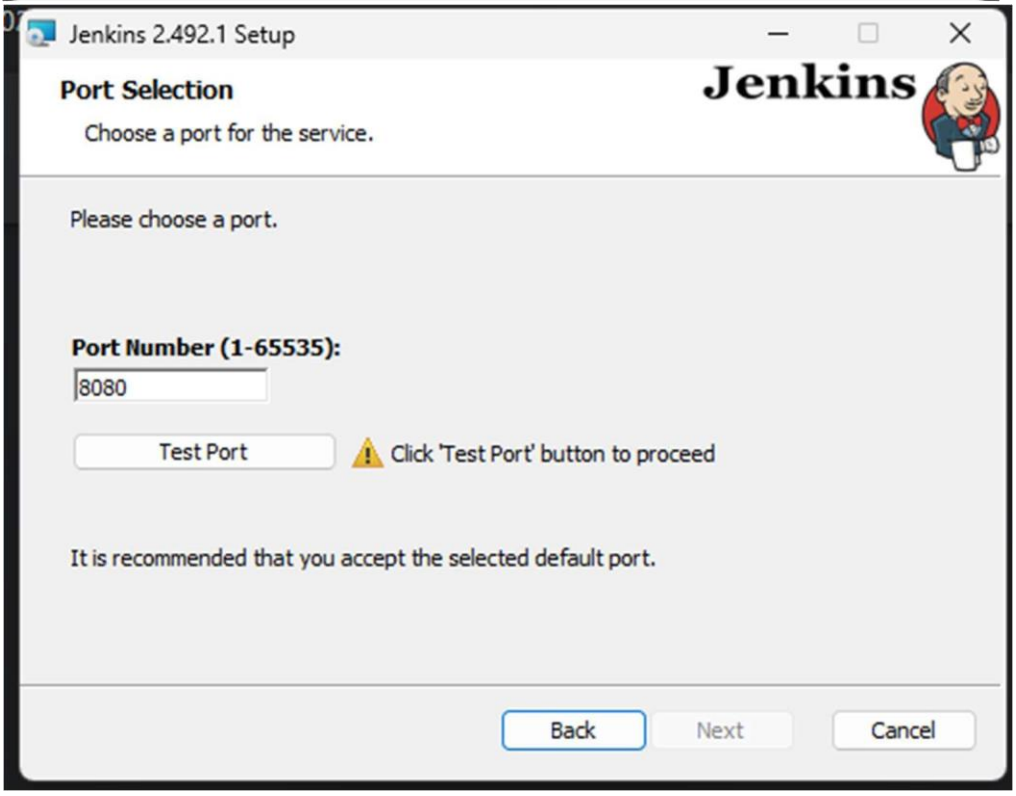
4. Creating a Build Job in Jenkins

4.1 Steps to Set Up a Build Job

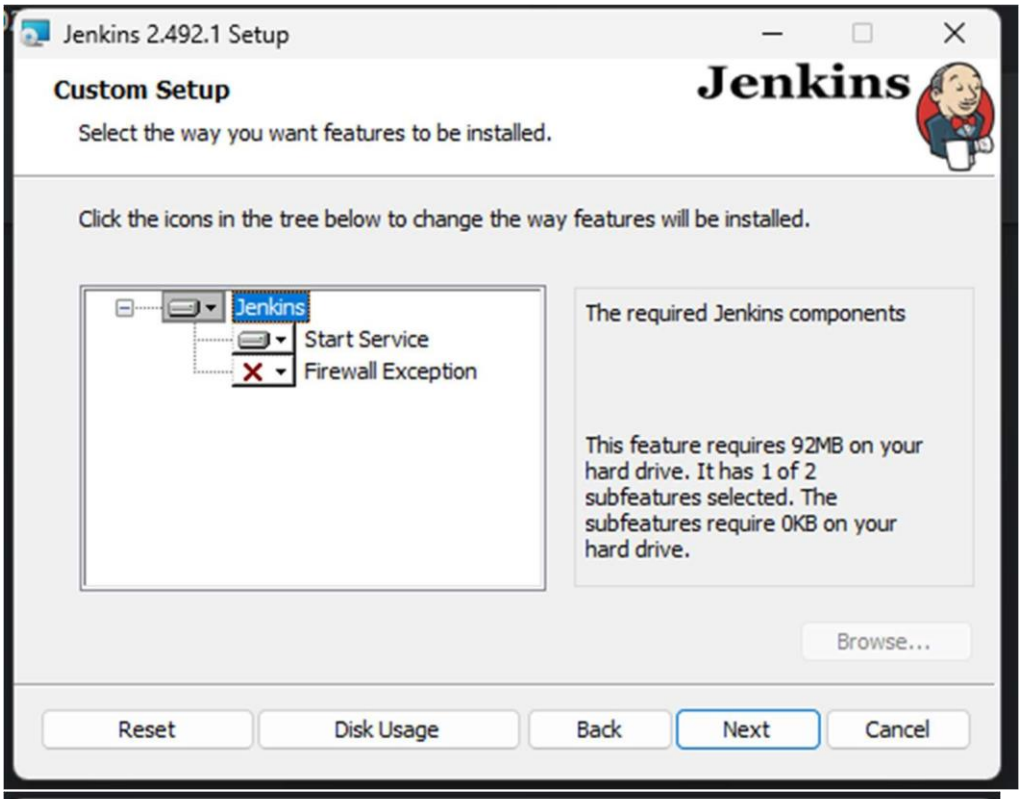
1. Create a New Job o Open Jenkins and click New Item.
    - Select Freestyle Project, enter a job name, and click OK.
2. Configure Source Code Management o Select Git and enter the repository URL. o Provide credentials if required.
3. Configure Build Steps o For Maven: Add a build step → Invoke Maven Targets → Enter clean package.
    - For Gradle: Add a build step → Invoke Gradle Script → Enter build.
    - For Ant: Add a build step → Invoke Ant → Enter build.xml.
4. Save and Build the Job o Click Save, then Build Now to execute.
    - Monitor logs in Console Output.

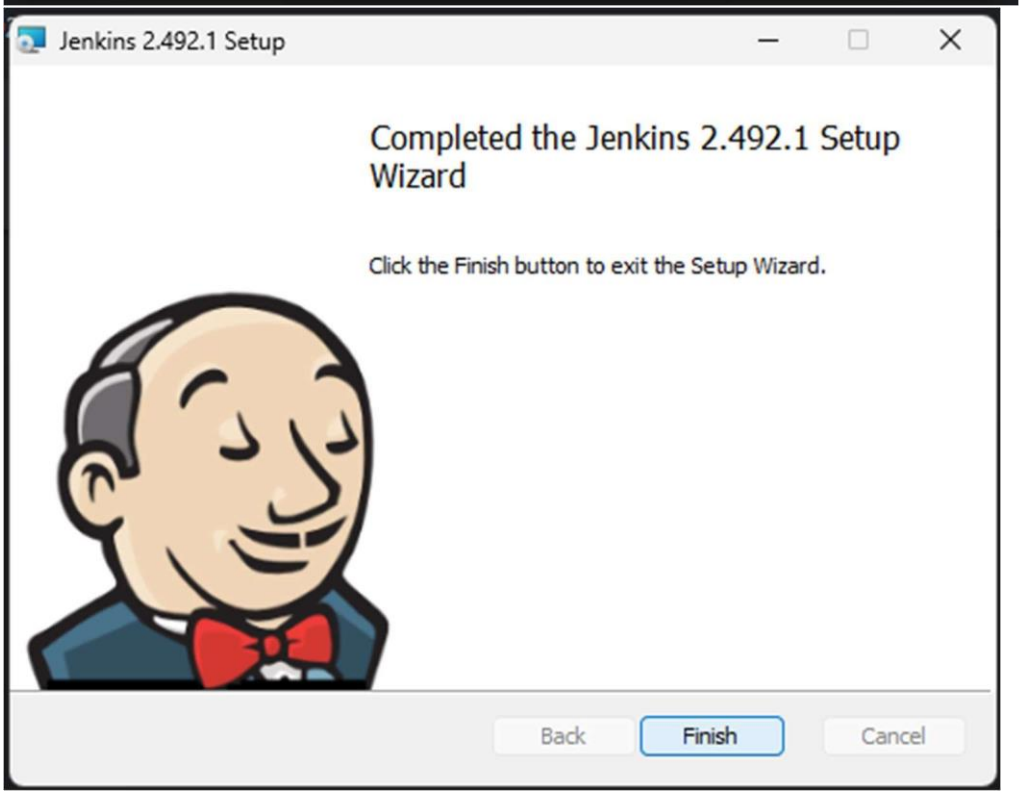
Salif Shaikh
T22 – 2201094



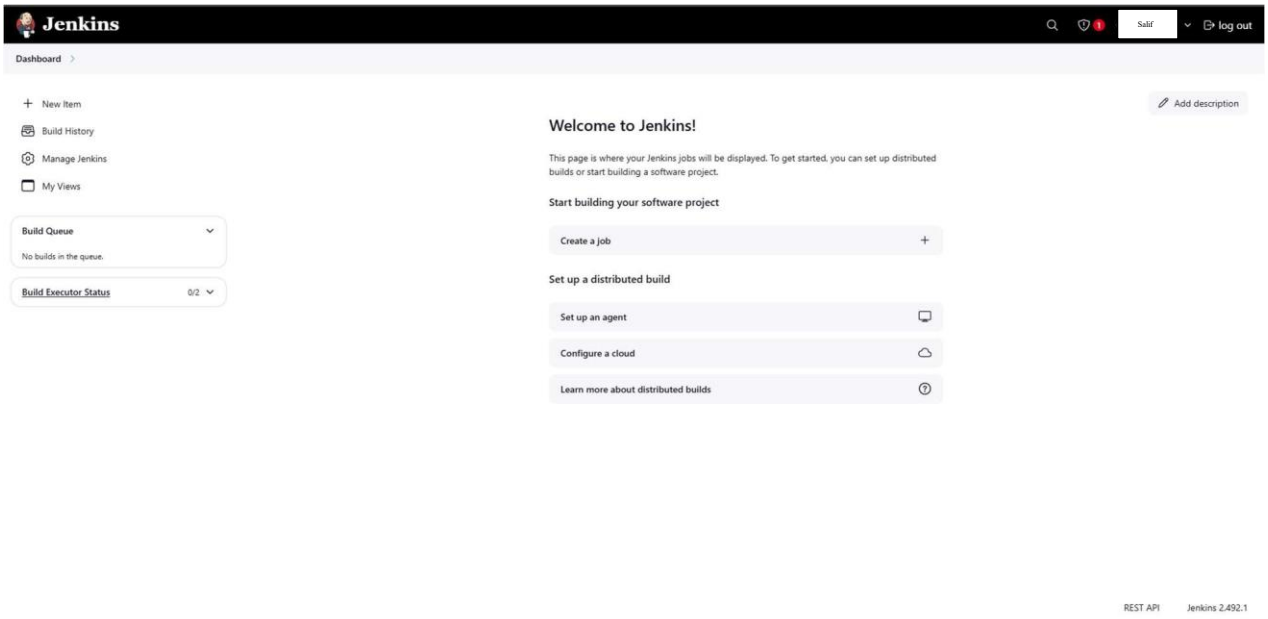


Salif Shaikh
T22 – 2201094





Salif Shaikh
T22 – 2201094



Conclusion

This experiment covered installing and configuring Jenkins for Continuous Integration, setting up build automation using Maven, Gradle, or Ant. By automating build jobs, we improve software quality, detect errors early, and enhance team collaboration.

SEPM EXP NO: 5

TO BUILD THE PIPELINE OF JOBS USING MAVEN / GRADLE / ANT IN JENKINS, CREATE A PIPELINE SCRIPT TO TEST AND DEPLOY AN APPLICATION OVER THE TOMCAT SERVER

Salif Shaikh T22-94 AI&DS

Theory:

1. Introduction to Jenkins and CI/CD

Jenkins is an open-source automation server used for continuous integration and continuous deployment (CI/CD). It helps automate the software development lifecycle by building, testing, and deploying applications.

1.1 CI/CD Concepts

- Continuous Integration (CI): Automatically integrates code changes into a shared repository and runs tests.
- Continuous Deployment (CD): Automates the process of deploying applications to production or staging environments.

1.2 Tools Used

- Jenkins – Automation server.
- Maven/Gradle/Ant – Build automation tools.
- Tomcat – A web server for deploying Java applications.

---

2. Installing and Configuring Jenkins

1. Download and Install Jenkins:
    - Download from Jenkins official site. o Install and start the Jenkins service.
    - Access Jenkins at http://localhost:8080/.
2. Install Required Plugins:

- Go to Manage Jenkins > Plugin Manager.
- Install Pipeline, Maven Integration, and Deploy to Container plugins.

---

3. Creating a Jenkins Pipeline

Jenkins pipelines define a series of automated steps for building, testing, and deploying applications.

3.1 Steps to Create a Pipeline

1. Open Jenkins Dashboard and click on New Item.
2. Select Pipeline and provide a project name.
3. Click OK and navigate to the Pipeline section.
4. Write a Pipeline script (Declarative or Scripted) to define the build and deployment process.

---

4. Writing a Jenkins Pipeline Script

The following script builds a Java application using Maven and deploys it to Tomcat:

```
groovy CopyEdit pipeline {  agent any  stages
{  stage('Checkout Code') {  steps {  git
'https://github.com/your-repository.git'
  }
 }  stage('Build with
Maven') {  steps {
sh 'mvn clean package'
  }
 }
 stage('Deploy to Tomcat') {  steps {  deploy adapters:
[tomcat8(credentialsId: 'tomcat-cred', path: '', url:
```

'http://yourtomcat-server:8080')], war: '**/*.war'

```
        }
    }
  }
}
```

---

5. Configuring Jenkins for Deployment

1. Configure Tomcat Server:
    - Install Tomcat and start the server. o Set up a user with deployment privileges in conf/tomcatusers.xml: xml CopyEdit

```
<role rolename="manager-gui"/>
<role rolename="manager-script"/>
<user username="admin" password="admin" roles="manager-gui,manager-script"/>
```

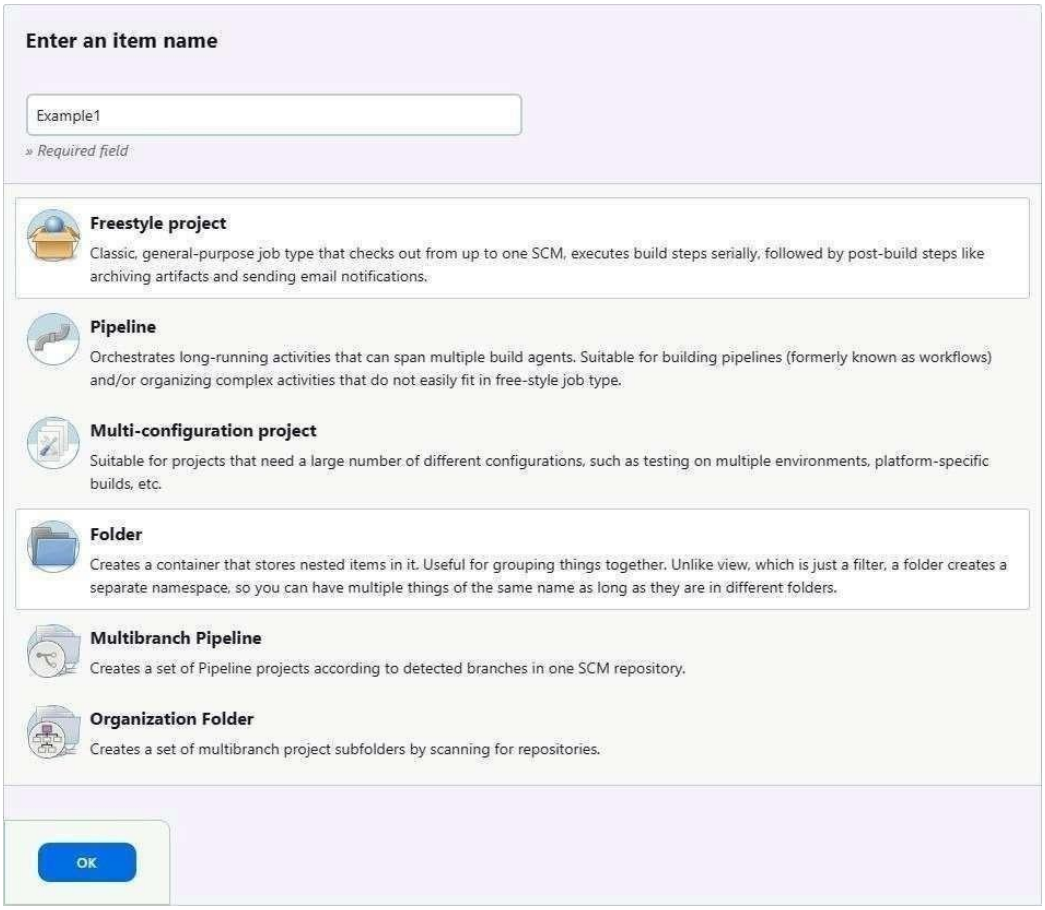
o Restart Tomcat.

2. Set Up Jenkins Credentials:
    - Go to Manage Jenkins > Credentials. o Add a username/password credential for Tomcat deployment.
3. Run the Pipeline in Jenkins:
    - Click Build Now to execute the pipeline. o Verify the deployment at http://your-tomcat-server:8080/your-app.

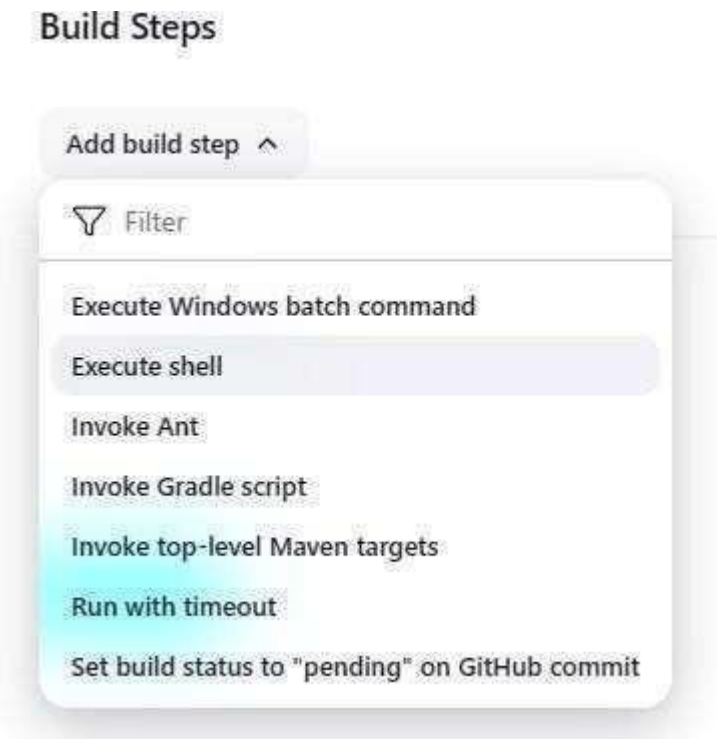
Example 1 :

Creating a job:

Naming the job and setting it as freestyle:

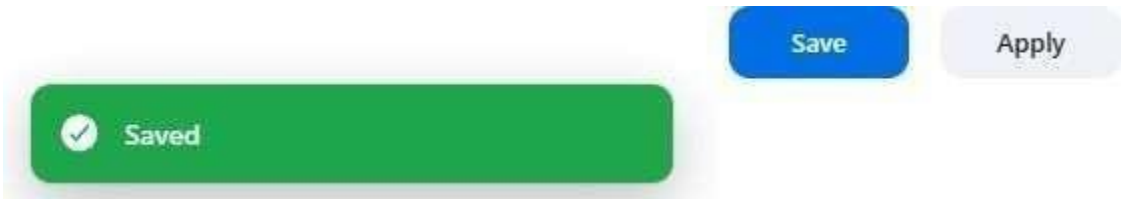


Selecting build type as “Execute shell”:



Entering a simple command for the shell execution:

Applying and saving the project configuration:

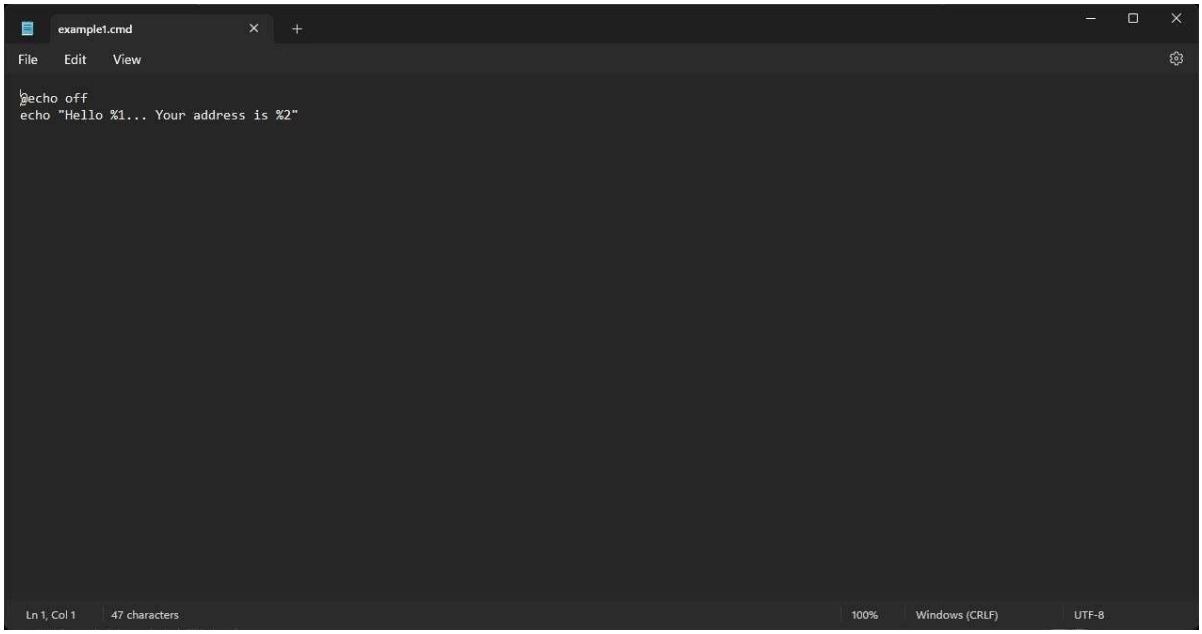


Building the project:

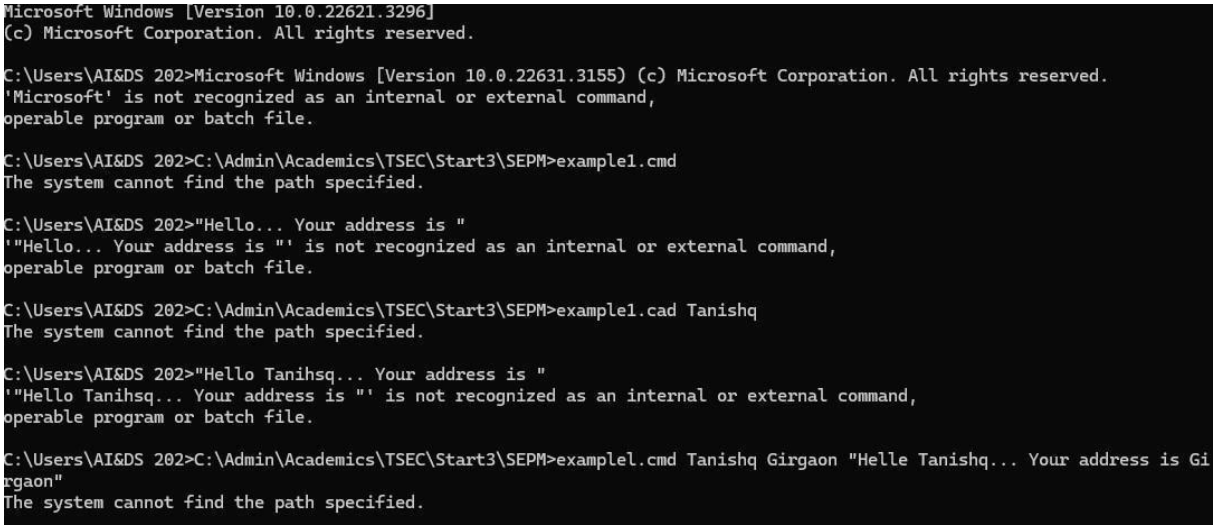
Console output (after building):

Example 1.2: Taking parameters through files Contents of script

example1.cmd:



Executing script example1.cmd on the terminal:

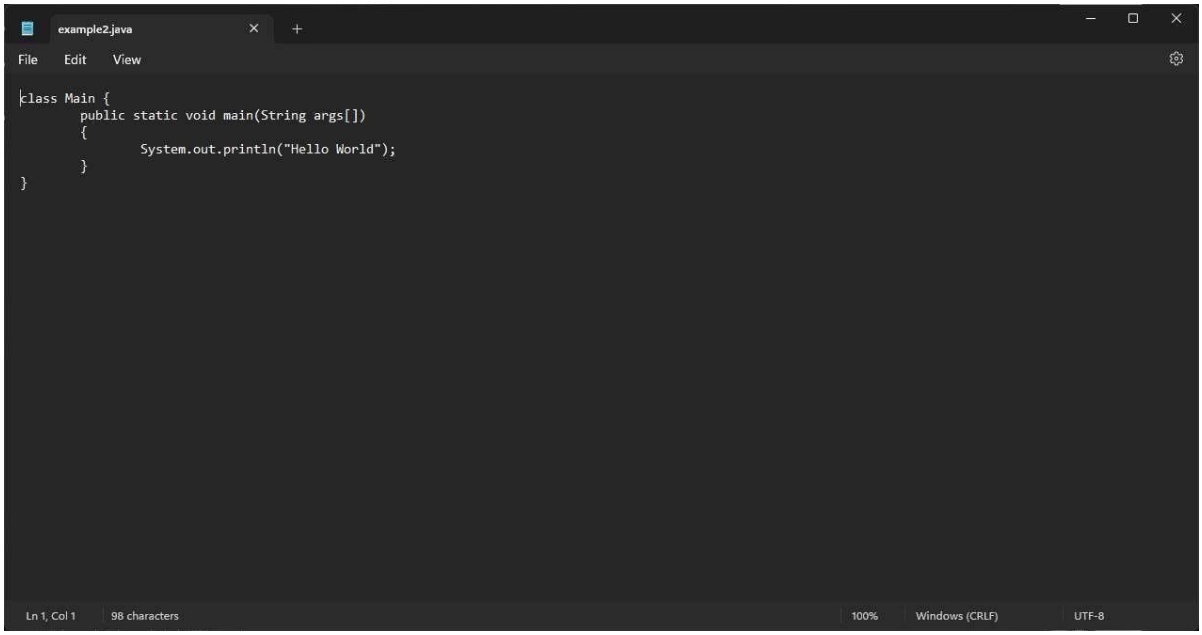


Modifying the Jenkins project to execute the script while supplying required parameters:

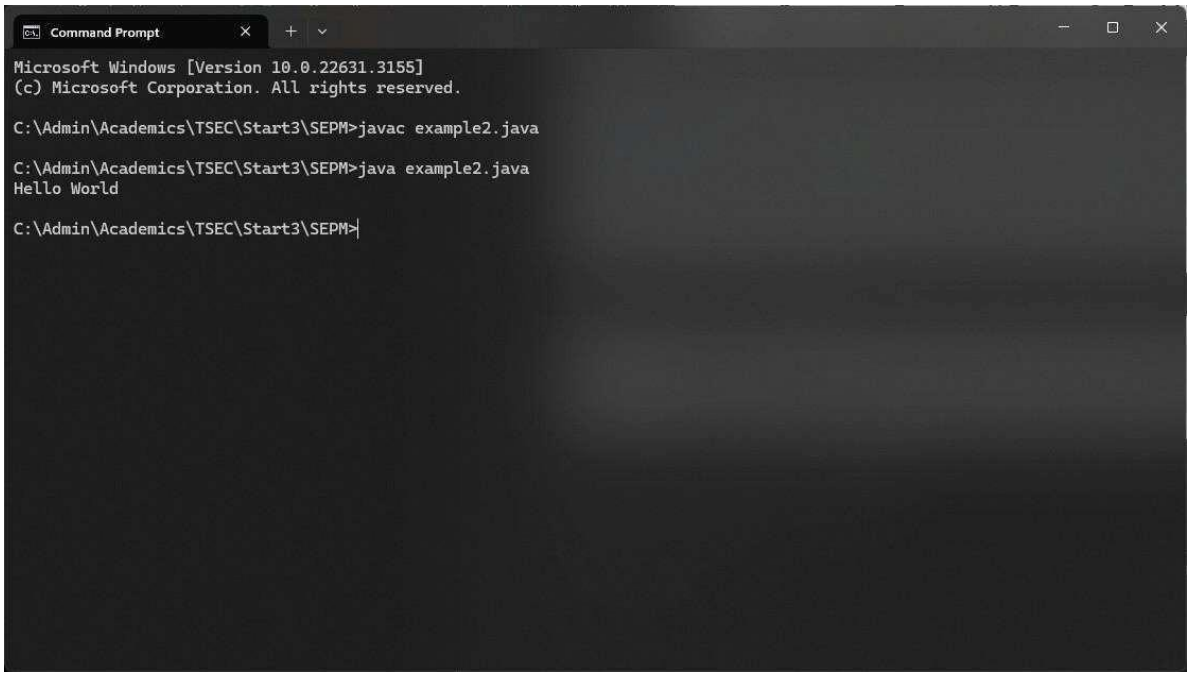
Console output after building the modified project:

Example 2 Example 2.1: Running a Java program under

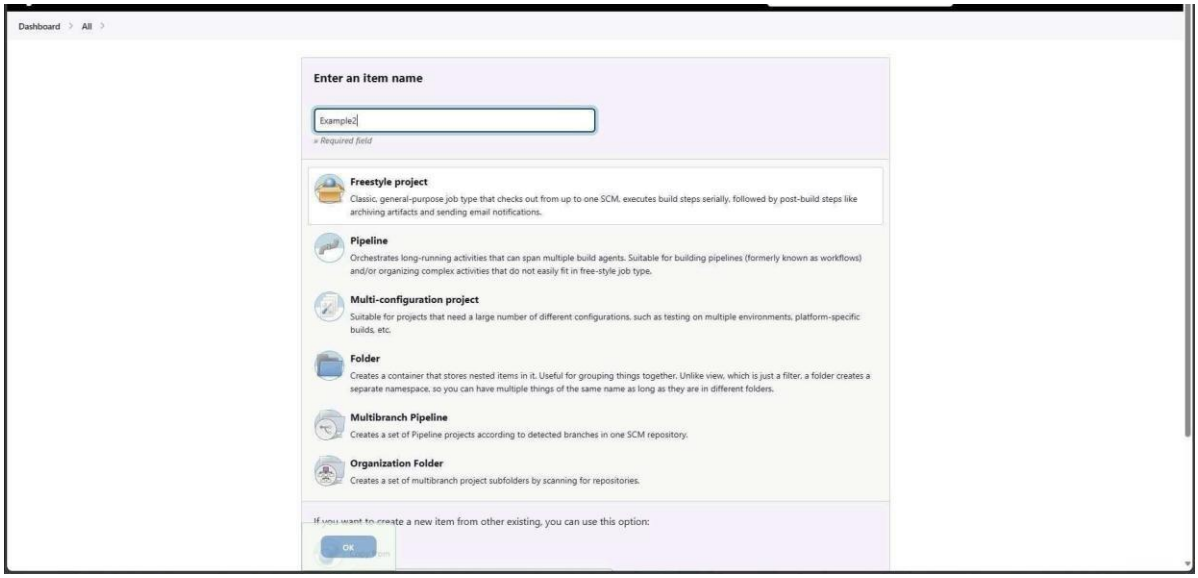
Jenkins Creating a simple Java program:



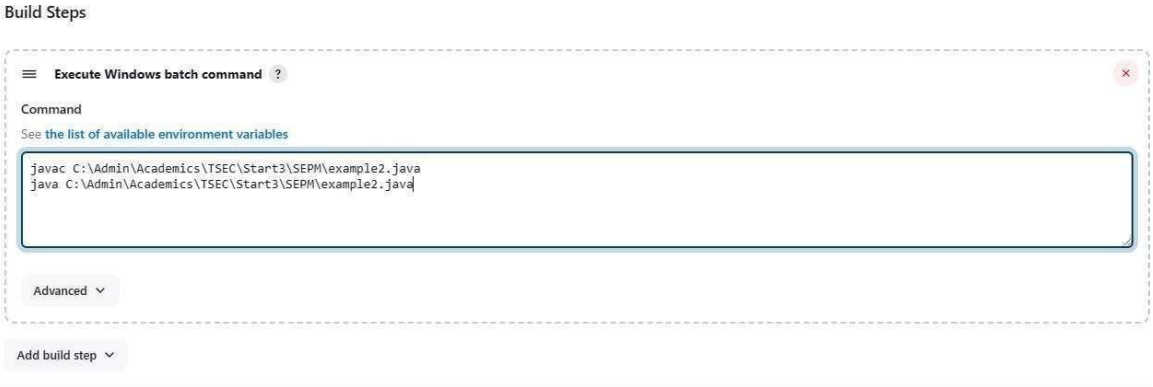
Compiling and running the program on the terminal:



Creating a new freestyle project:



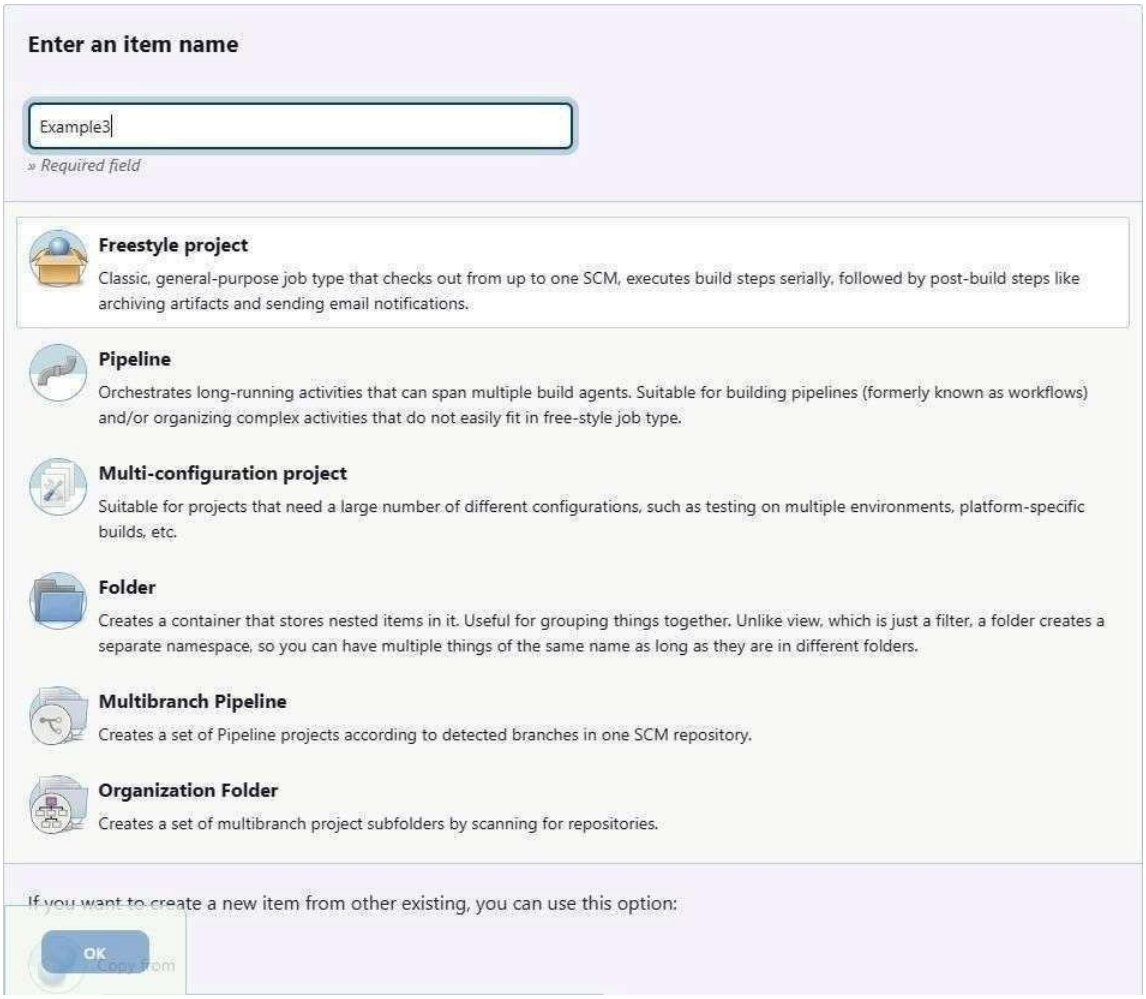
Configure new project:



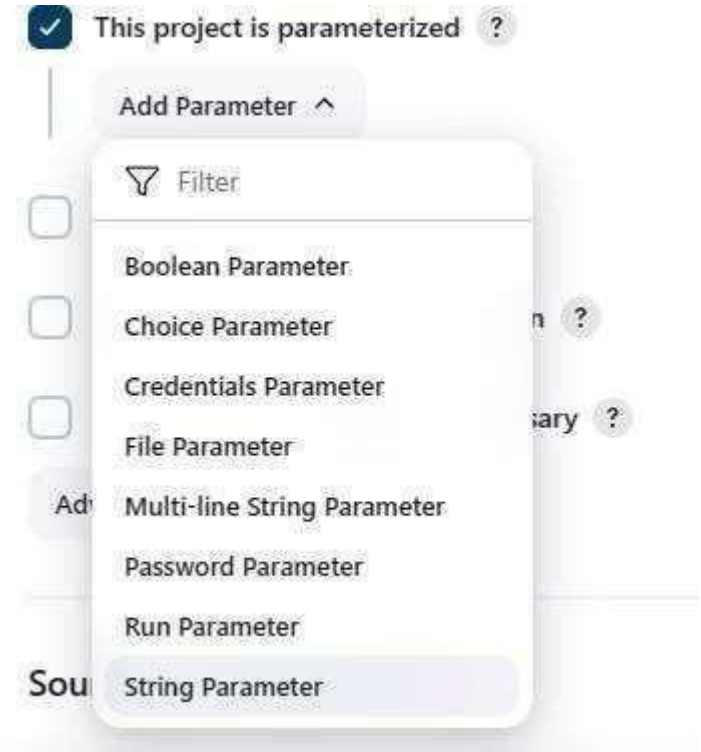
Console output after building:

Example 3 Example 3.1: Parameterise build

Creating a new freestyle project:



Enabling parameterisation and adding a String parameter:



Configuring the string parameter as Fname:

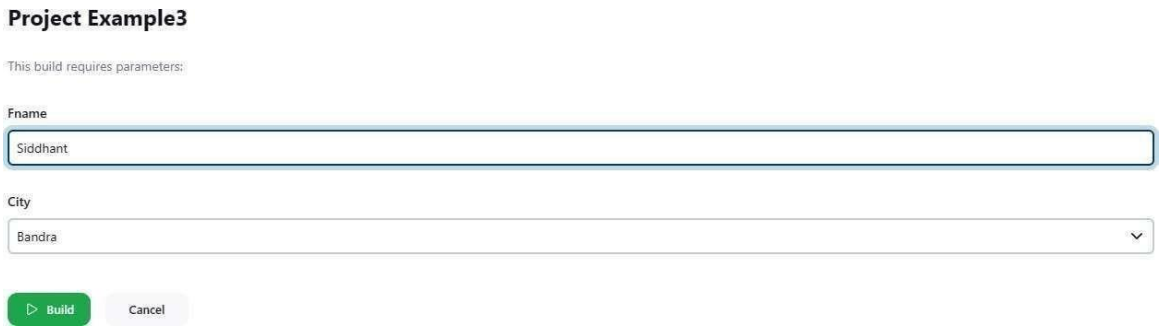
Adding a choice parameter and configuring it as City with the following choices:

Creating a script which takes 2 arguments for name and city:

```
C:\Users\AI&DS 202>Microsoft Windows [Version 10.0.22631.3155] (c) Microsoft Corporation. All rights reserved.
'Microsoft' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\AI&DS 202>C:\Admin\Academics\TSEC\Start3\SEPH>example3.cnd
The system cannot find the path specified.

C:\Users\AI&DS 202>Hello your name is and your city is
'Hello' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\AI&DS 202>C:\Admin\Academics\TSEC\Start3\SEPH example3.cmd Tanishq
The system cannot find the path specified.

C:\Users\AI&DS 202>Hello your name is Tanishq and your city is
'Hello' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\AI&DS 202>C:\Admin\Academics\TSEC\Start3\SEPM>example3.cmd Tansishq Bandra
The system cannot find the path specified.

C:\Users\AI&DS 202>Hello your name is Tanishq and your city is Bandra
'Hello' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\AI&DS 202>C:\Admin\Academics\TSEC\Start3\SEPH
```

Configuring build steps:

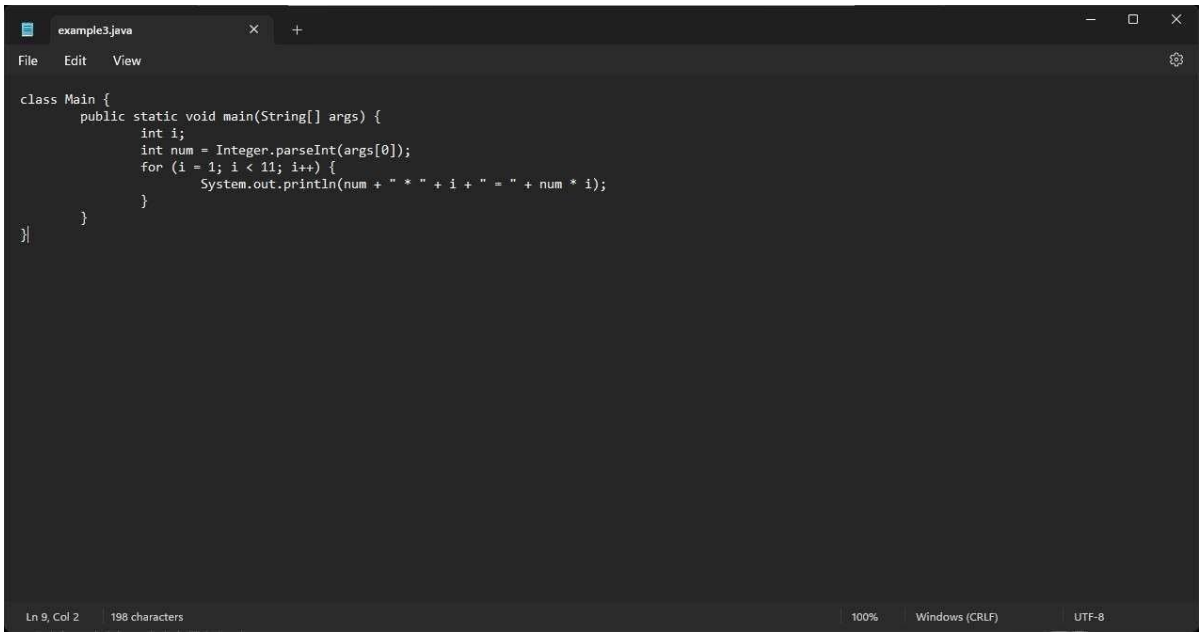
Entering parameters for build:



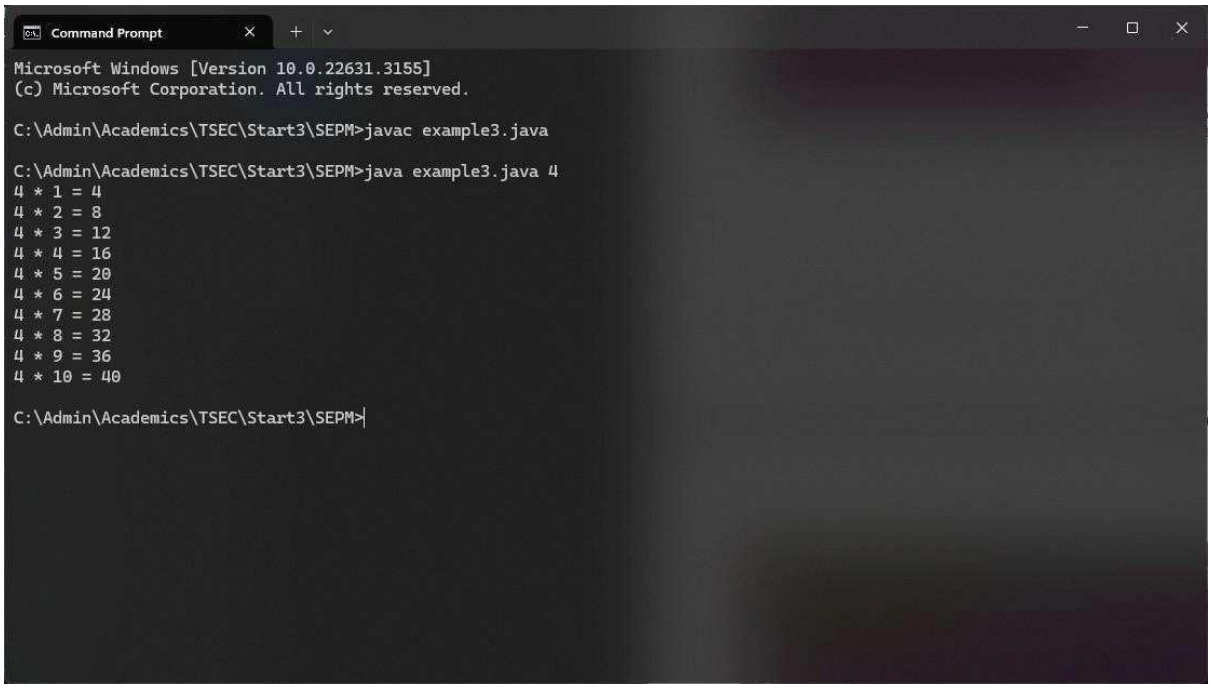
Console output after building:

Example 3.2: Running a Java program with parameters Creating

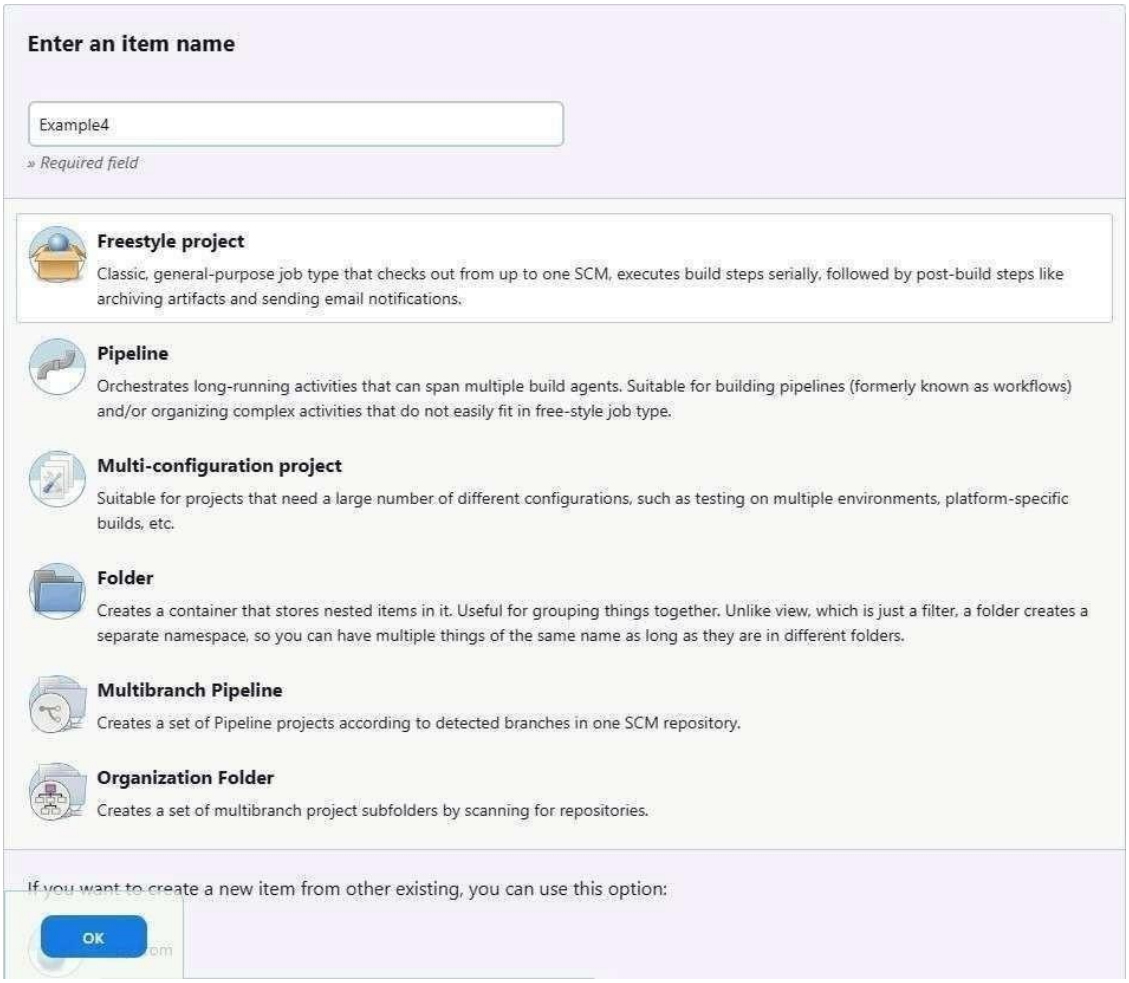
a Java program with an input argument:



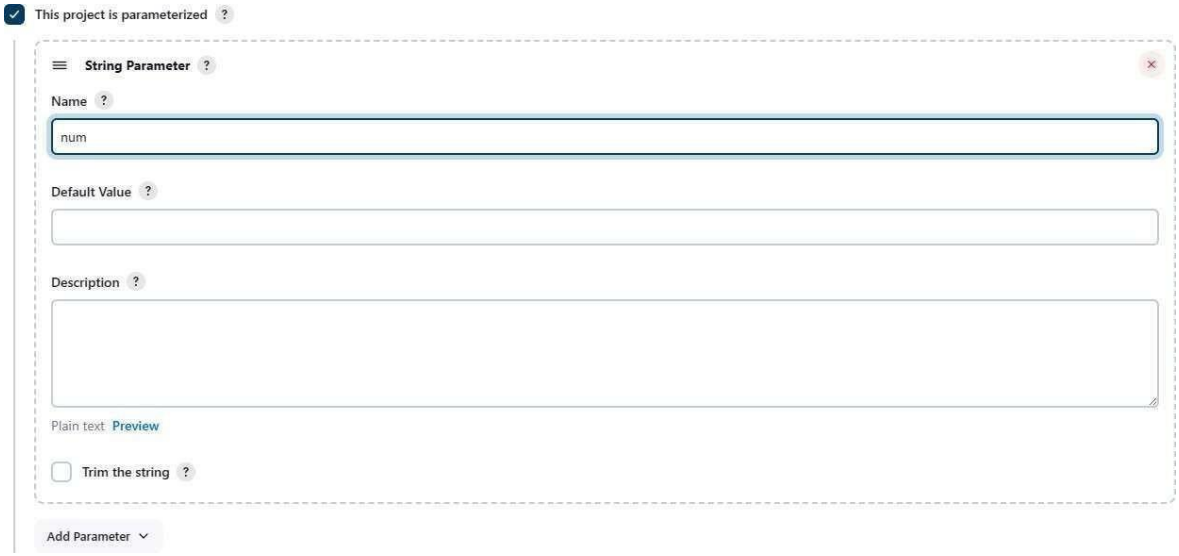
Testing the program on the terminal:



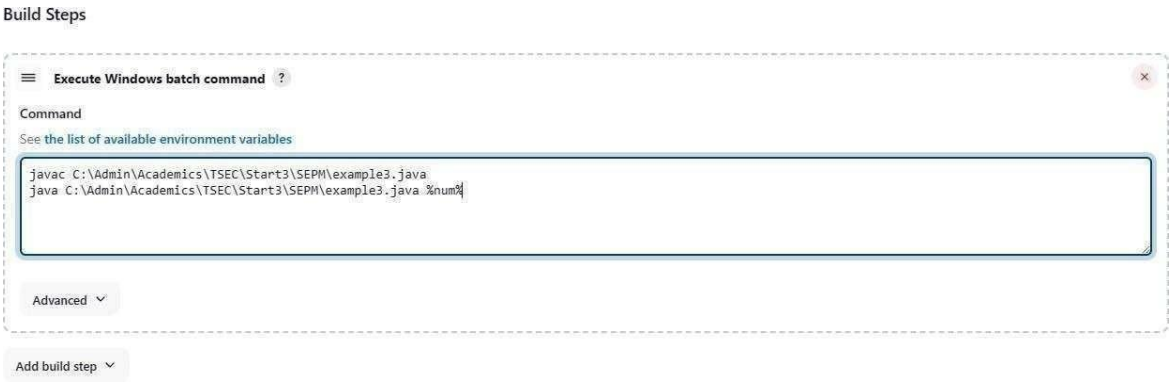
Creating a new freestyle project:



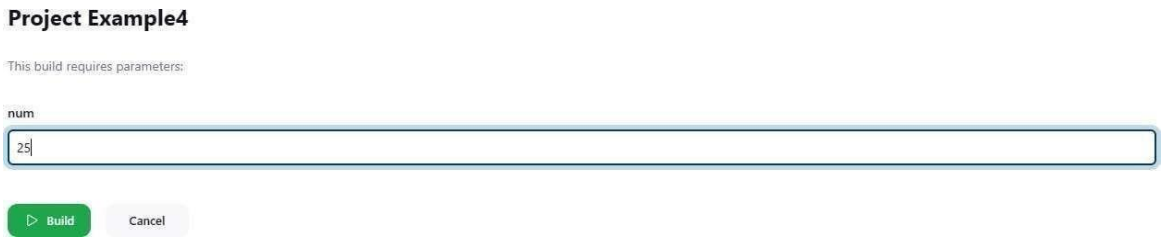
Parameterise the project by adding a string parameter as follows:



Configure the build steps:



Entering the parameter for the build:

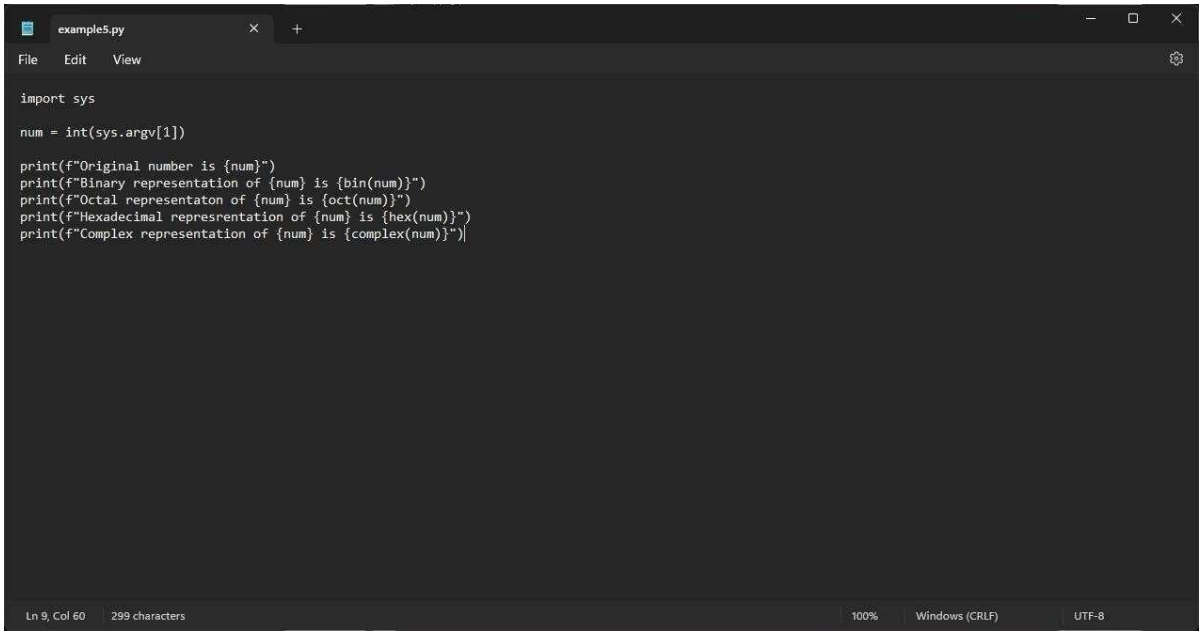


Console output after building:

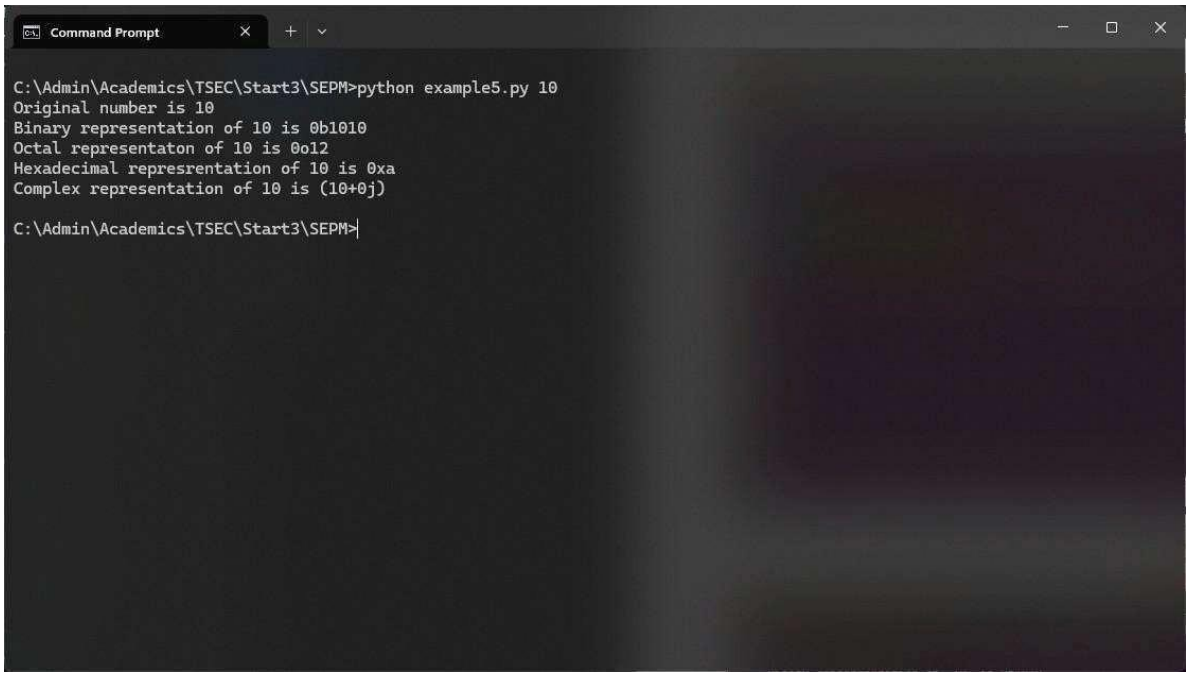
Example 5 Example

5.1: Running a Python program Creating a simple Python

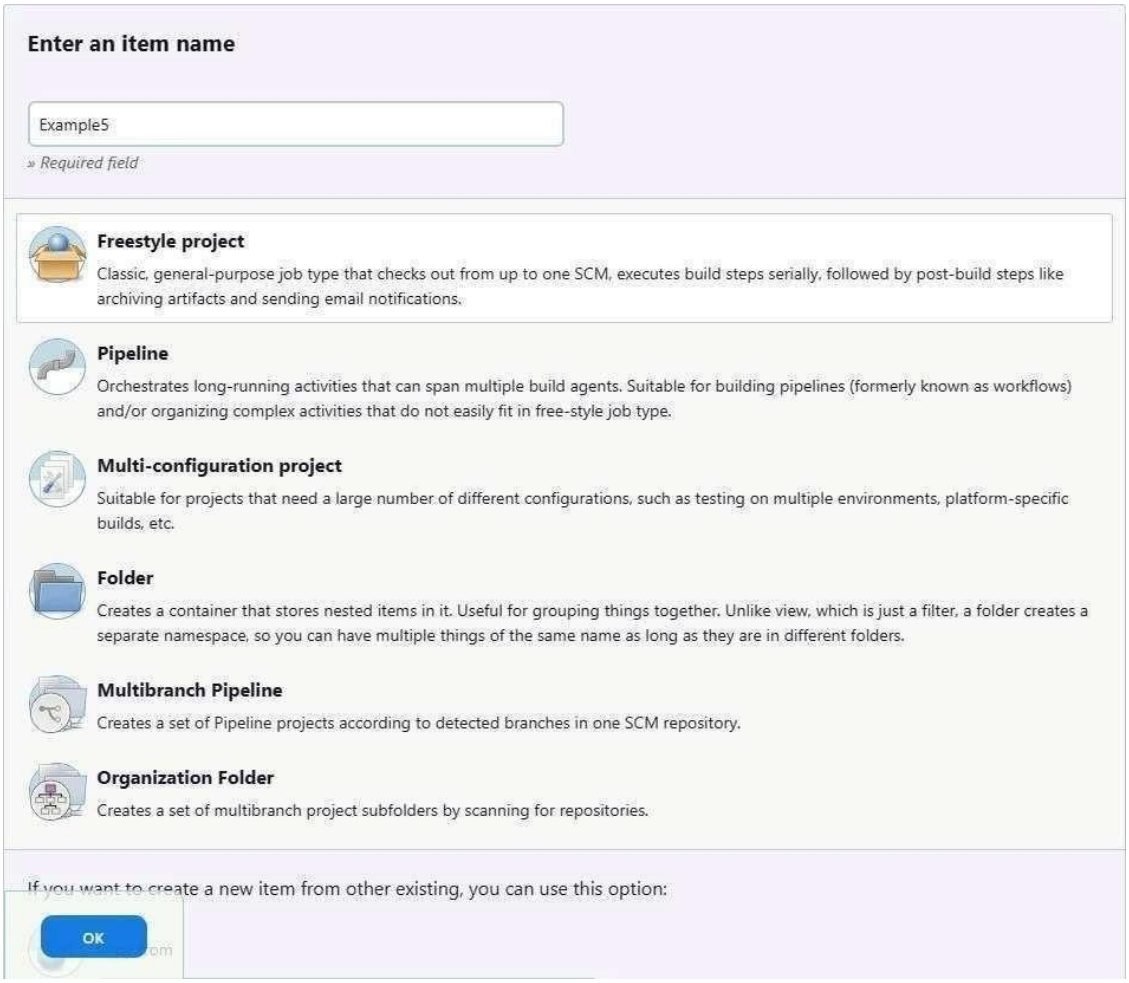
script:



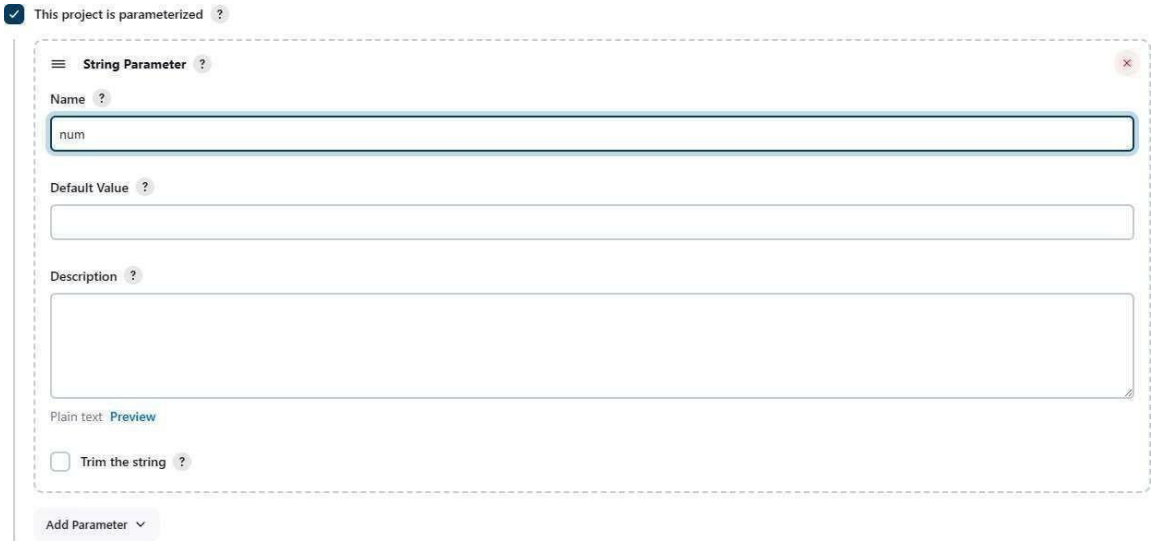
Running the Python script on the terminal:



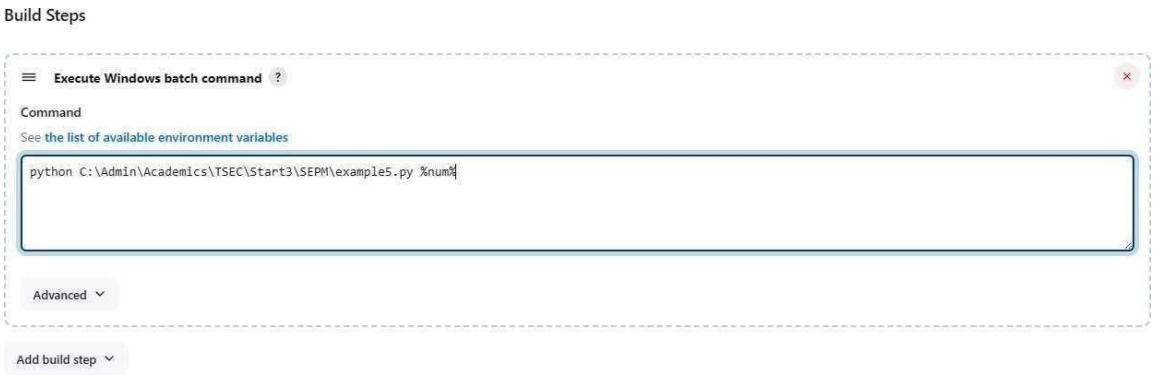
Creating a new freestyle project:



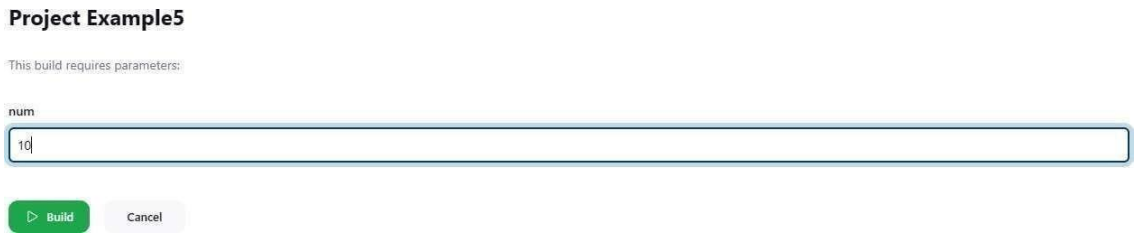
Parameterising the project with a string parameter as follows:



Configuring the build steps:



Setting the parameter for the build:



Console output after building:

Conclusion

This experiment demonstrated how to automate a software build and deployment process using Jenkins. By integrating Maven, Gradle, or Ant, we streamlined the compilation and testing of applications, while Jenkins facilitated continuous deployment to a Tomcat server.

Salif Shaikh
T22 – 2201094

EXPERIMENT 6

STUDYING AGILE METHODOLOGY AND TEST CASE MANAGEMENT USING JIRA TOOL

Theory:

1. Introduction to Agile Methodology

Agile methodology is an iterative approach to software development and project management that emphasizes flexibility, collaboration, and customer feedback. It helps teams deliver high-quality software in incremental steps rather than following a rigid plan.

1.1 Agile Manifesto

Agile is based on four core values:

1. Individuals and interactions over processes and tools.
2. Working software over comprehensive documentation.
3. Customer collaboration over contract negotiation.
4. Responding to change over following a plan.

1.2 Agile Principles

Agile follows 12 principles, including:

- Customer satisfaction through early and continuous delivery.
- Embracing changing requirements for the customer's competitive advantage.
- Delivering working software frequently.
- Close collaboration between business and development teams.
- Motivated individuals and self-organizing teams.
- Continuous attention to technical excellence.

---

2. Agile Frameworks

Several frameworks exist under Agile, including:

1. Scrum – A framework for iterative and incremental product development.
2. Kanban – A visual workflow management system.
3. Extreme Programming (XP) – Focuses on engineering practices.
4. Lean Software Development – Focuses on minimizing waste and maximizing value.

2.1 Scrum Framework

Salif Shaikh
T22 – 2201094

Scrum is one of the most popular Agile frameworks. It consists of:

- Scrum Team: Product Owner, Scrum Master, Development Team.
- Scrum Artifacts: Product Backlog, Sprint Backlog, Increment.

- Scrum Events: Sprint Planning, Daily Stand-ups, Sprint Review, Sprint Retrospective.
- Sprint Cycle: A time-boxed period (usually 2-4 weeks) where a team completes selected backlog items.

---

3. Introduction to Jira

Jira is a popular tool developed by Atlassian for Agile project management, issue tracking, and test case management. It helps teams plan, track, and manage software development projects efficiently.

3.1 Features of Jira

- Customizable dashboards for tracking project progress.
- Issue tracking for creating, assigning, and monitoring tasks.
- Workflow automation to improve efficiency.
- Integration with various development and testing tools.

---

4. Test Case Management Using Jira

Jira can be used to manage test cases effectively by integrating with testing plugins like Zephyr and Xray.

4.1 Steps to Manage Test Cases in Jira

1. Creating a Test Case:
    - Navigate to the Jira project.
    - Click on Create Issue and select Test Case (if using Zephyr/Xray). o Define the test case with details such as Test Summary, Preconditions, Test Steps, and Expected Results.
    - Click Create to save the test case.
2. Executing a Test Case:
    - Open the created test case.
    - Click Execute and select the test cycle. o Update the test execution status (Pass, Fail, In Progress, Blocked).
3. Tracking Test Results:
    - Use Jira dashboards to generate reports on test execution progress.
    - Analyze defects by linking test cases to bug reports.
4. Integrating Jira with CI/CD Pipelines:

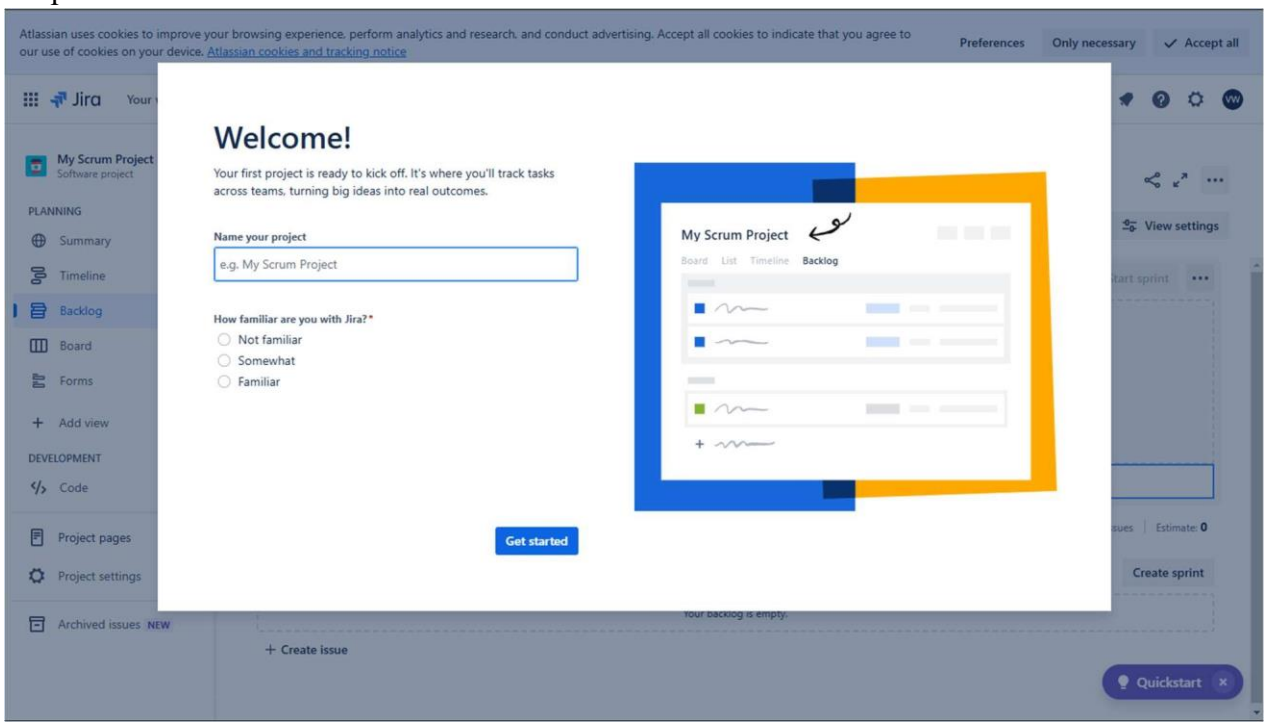
- Jira can integrate with Jenkins, Bamboo, and other DevOps tools to automate testing workflows.

---

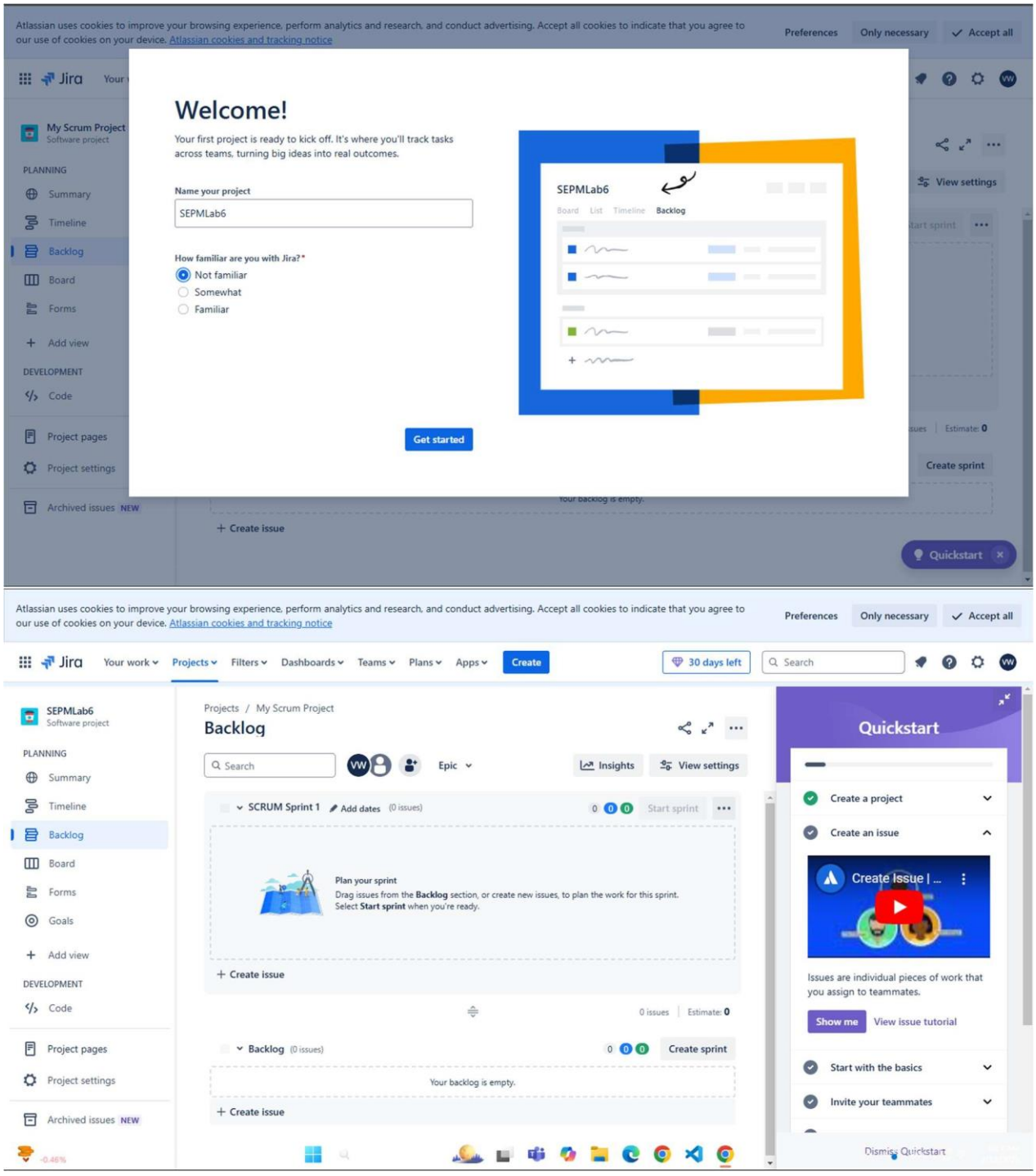
5. Benefits of Using Jira for Agile and Test Case Management

- Centralized Tracking: All tasks, sprints, and test cases are managed in one place.
- Collaboration: Teams can communicate and update task statuses in real-time.
- Customization: Jira can be tailored to fit any workflow.
- Automation: Reduces manual effort through workflow automation and integration with DevOps tools.

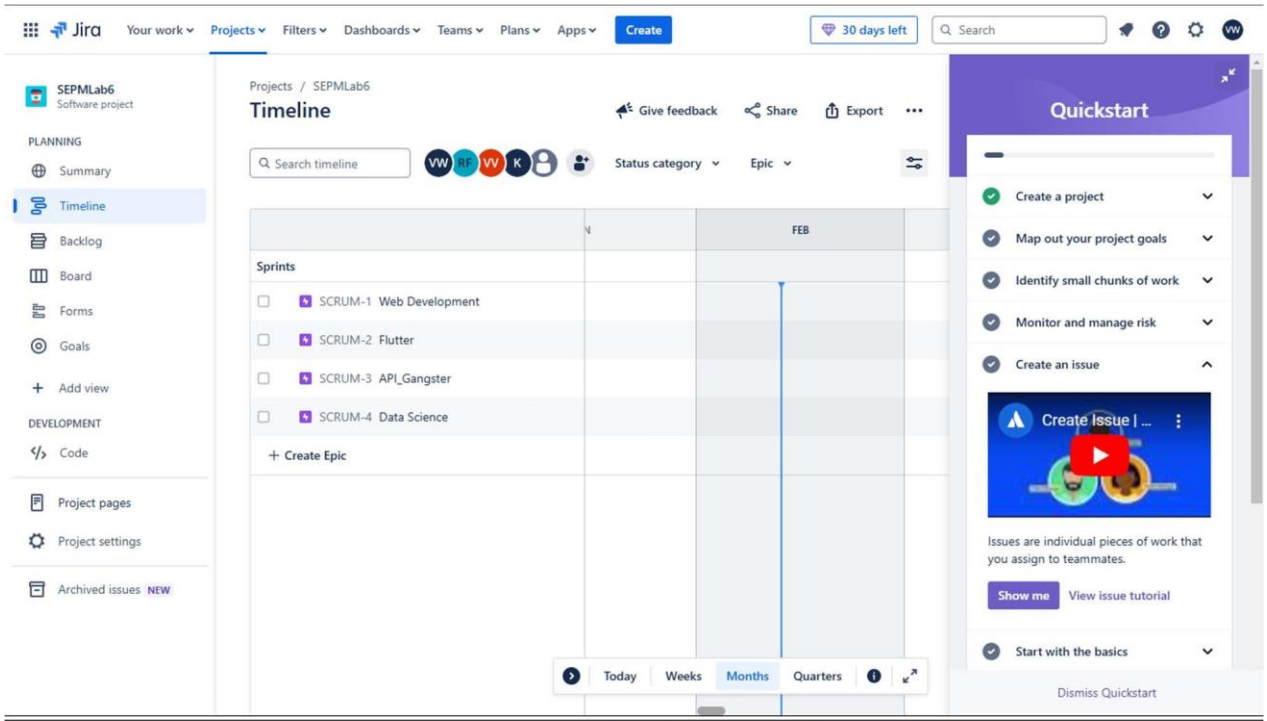
Output:

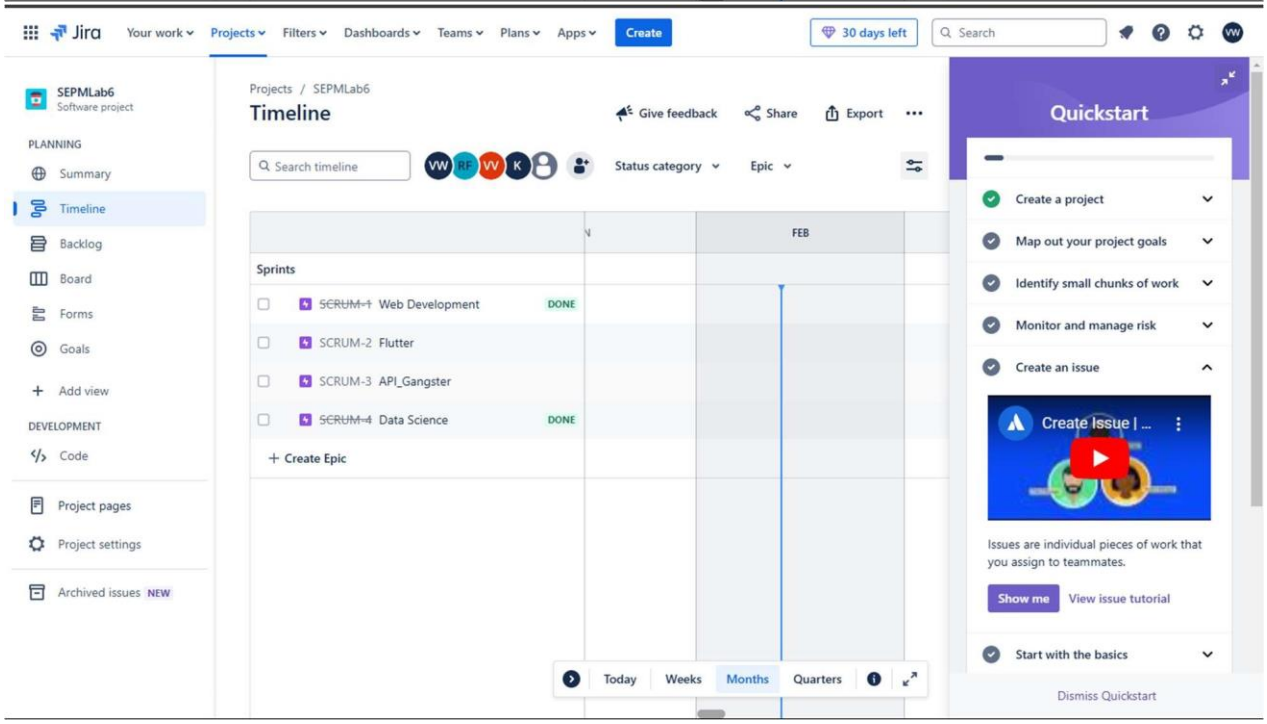


Salif Shaikh
T22 – 2201094

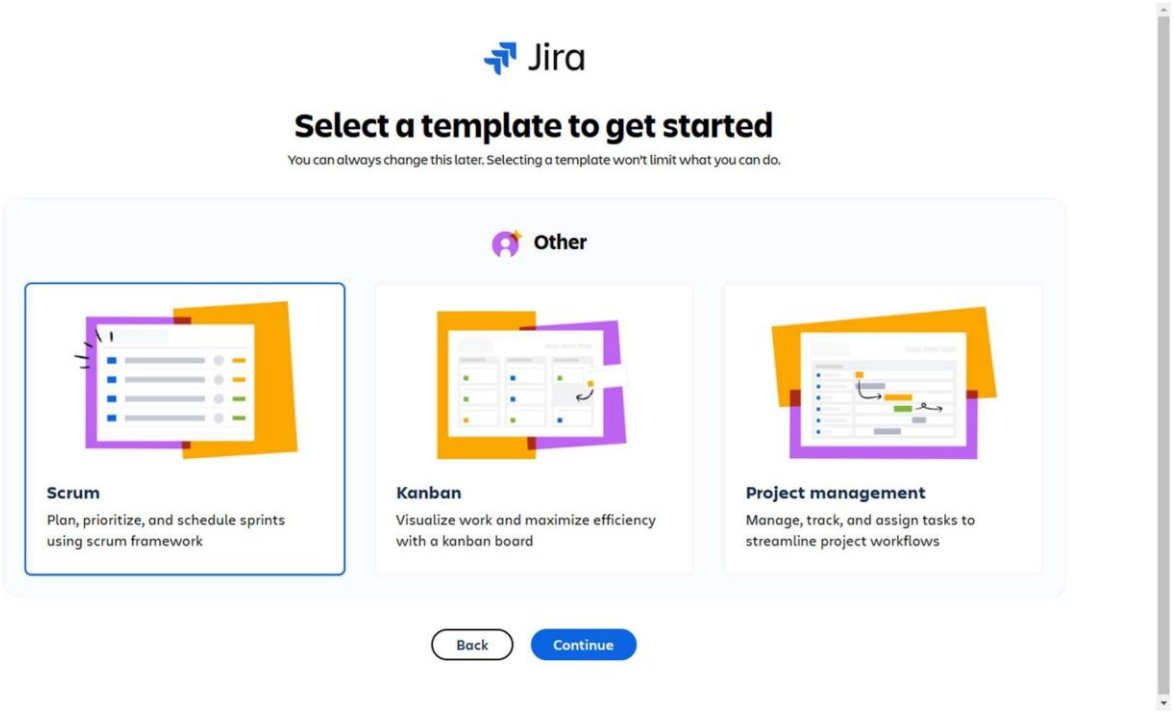


Salif Shaikh
T22 – 2201094





Salif Shaikh
T22 – 2201094



Conclusion

This experiment provided an overview of Agile methodology and how Jira is used for project and test case management. Jira enhances Agile practices by improving workflow efficiency, collaboration, and test case execution tracking, making it a valuable tool in modern software development.

SEPM:EXP7

To Create Scrum Board for Scrum Master using JIRA Tool.

Salif Shaikh -

BATCH:TE-T22

Introduction to Scrum Board in JIRA

A Scrum Board in JIRA is a visual representation of work progress within a Scrum team. It helps teams manage their backlog, sprint planning, and task execution efficiently. The Scrum Master uses the board to track progress, identify bottlenecks, and ensure smooth sprint execution.

Objectives

To understand how to create and configure a Scrum Board in JIRA.

To learn how to manage sprints and tasks effectively using JIRA.

Prerequisites

A JIRA account with administrator or project management access.

A JIRA project set up for Scrum methodology.

Basic understanding of Scrum framework including backlog, sprint, and user stories.

Steps to Create a Scrum Board in JIRA

Step 1: Log in to JIRA

Open a web browser and navigate to JIRA’s official website.

Enter your credentials and sign in.

Step 2: Create a New Scrum Project

1. Click on the JIRA Software option from the dashboard.
2. Select Projects > Create Project.
3. Choose Scrum Software Development as the project template.
4. Enter the project details including name, key, and team details.
5. Click Create to set up the project.

Step 3: Configure the Scrum Board

Navigate to Boards and select Create Board.

Choose Scrum Board.

Select whether to create the board from an existing project or a new project.

Assign the board to your Scrum project.

Click Create Board.

Step 4: Define and Prioritize the Backlog

Open the Scrum Board and navigate to the Backlog tab.

Click Create Issue to add user stories, tasks, and bugs.

Assign priority levels and categories to each item.

Organize backlog items based on priority.

Step 5: Create and Start a Sprint

Go to the Backlog tab.

Click Create Sprint.

Select user stories/tasks to include in the sprint.

Click Start Sprint and set the sprint duration.

Step 6: Manage and Track Sprint Progress

Move tasks across the board from To Do to In Progress and then Done.

Regularly update task statuses during daily stand-up meetings.

Use filters to view tasks assigned to different team members.

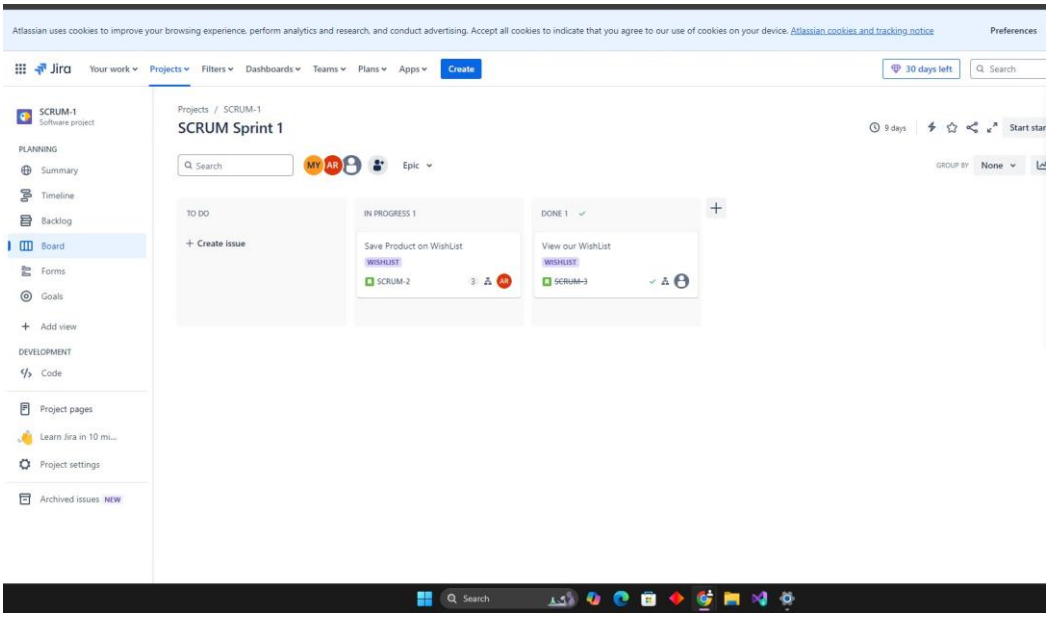
Monitor sprint burndown charts for progress tracking.

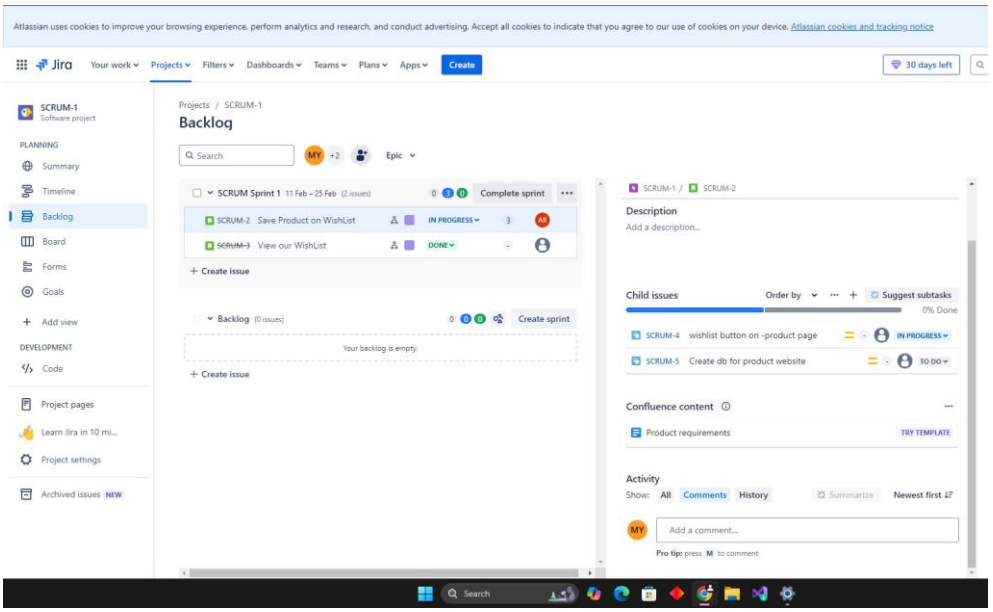
Step 7: Review and Close the Sprint

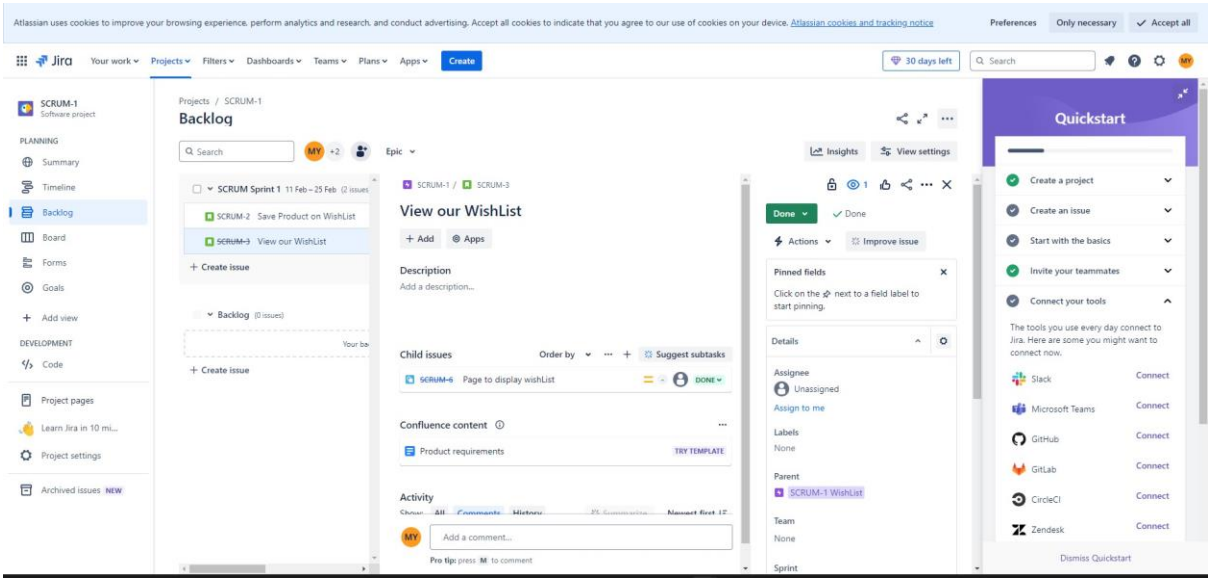
At the end of the sprint, review completed tasks.

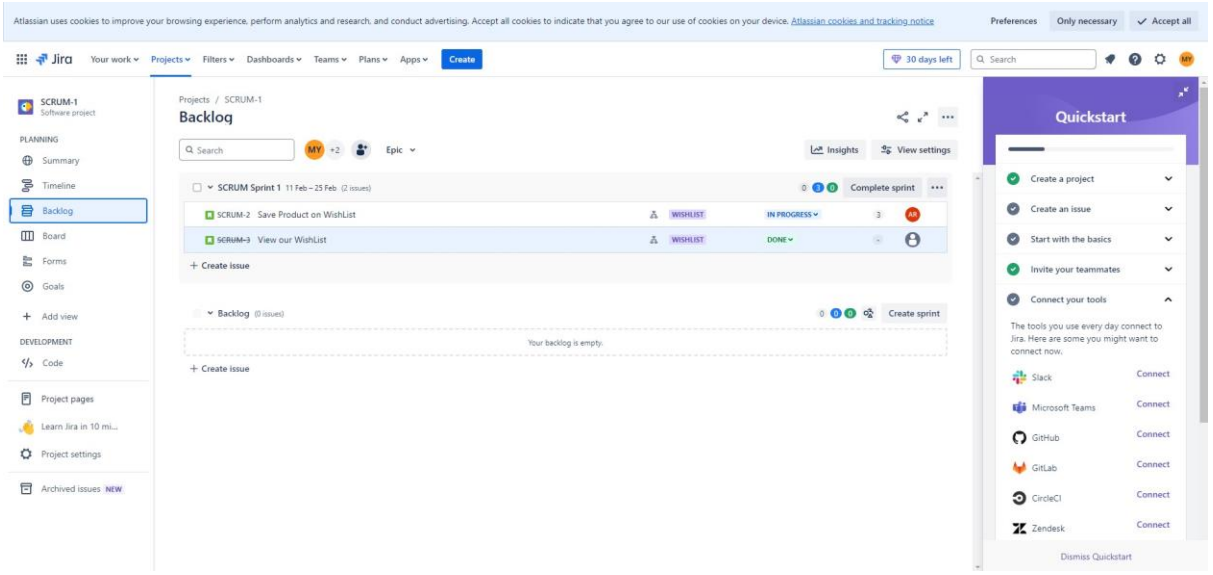
Conduct a Sprint Retrospective to discuss improvements.

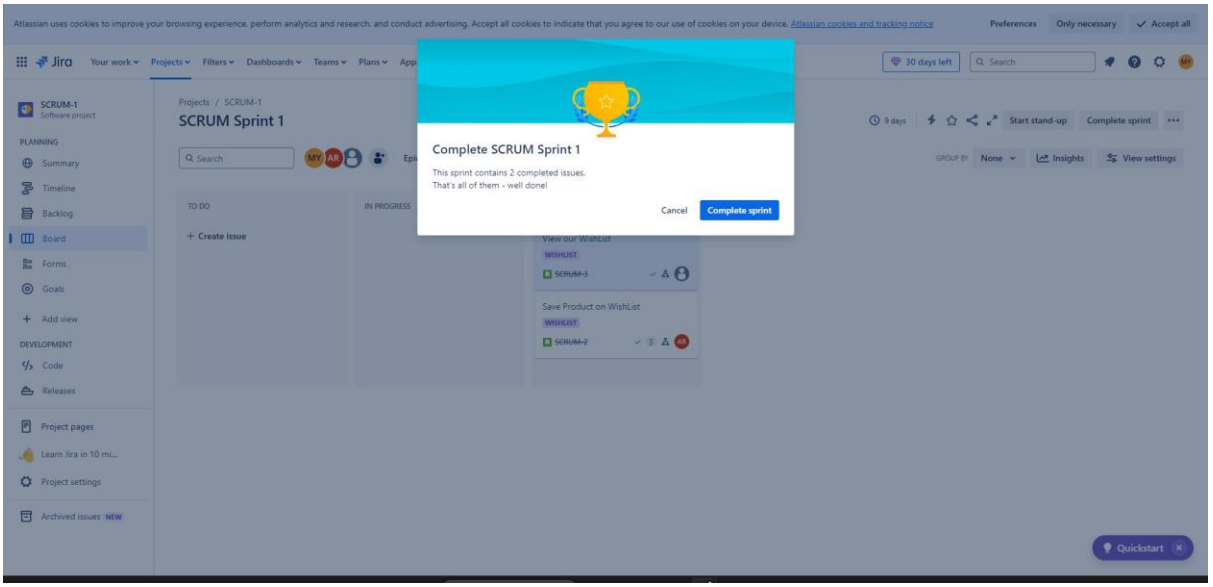
Close the sprint and move unfinished tasks to the next sprint.


Atlassian uses cookies to improve your browsing experience, perform analytics and research, and conduct advertising. Accept all cookies to indicate that you agree to our use of cookies on your device. Atlassian cookies and tracking notice
Preferences
Jira
Your work
Projects
Filters
Dashboards
Teams
Plans
Apps
Create
30 days left
Search
SCRUM-1
Software project
PLANNING
Summary
Timeline
Backlog
Board
Forms
Goals
Add view
DEVELOPMENT
Code
Project pages
Learn Jira in 10 mi...
Project settings
Archived issues NEW
Projects / SCRUM-1
SCRUM Sprint 1
9 days
Start star
Search
Epic
GROUP BY
None
TO DO
+ Create issue
IN PROGRESS 1
Save Product on WishList
WISHLIST
SCRUM-2
DONE 1
View our WishList
WISHLIST
SCRUM-3
Search


Atlassian uses cookies to improve your browsing experience, perform analytics and research, and conduct advertising. Accept all cookies to indicate that you agree to our use of cookies on your device. Atlassian cookies and tracking notice
Jira
Your work
Projects
Filters
Dashboards
Teams
Plans
Apps
Create
30 days left
SCRUM-1
Software project
PLANNING
Summary
Timeline
Backlog
Board
Forms
Goals
Add view
DEVELOPMENT
Code
Project pages
Learn Jira in 10 mi...
Project settings
Archived issues NEW
Projects / SCRUM-1
Backlog
Search
Epic
SCRUM Sprint 1 11 Feb – 25 Feb (2 issues)
Complete sprint
SCRUM-2 Save Product on WishList
IN PROGRESS
SCRUM-3 View our WishList
DONE
+ Create issue
Backlog (0 issues)
Create sprint
Your backlog is empty.
+ Create issue
SCRUM-1 / SCRUM-2
Description
Add a description...
Child issues
Order by
Suggest subtasks
0% Done
SCRUM-4 wishlist button on -product page
IN PROGRESS
SCRUM-5 Create db for product website
TO DO
Confluence content
Product requirements
TRY TEMPLATE
Activity
Show: All Comments History
Summarize
Newest first
Add a comment...
Pro tip: press M to comment
Search


Atlassian uses cookies to improve your browsing experience, perform analytics and research, and conduct advertising. Accept all cookies to indicate that you agree to our use of cookies on your device. Atlassian cookies and tracking notice
Preferences
Only necessary
Accept all
Jira
Your work
Projects
Filters
Dashboards
Teams
Plans
Apps
Create
30 days left
Search
SCRUM-1
Software project
PLANNING
Summary
Timeline
Backlog
Board
Forms
Goals
Add view
DEVELOPMENT
Code
Project pages
Learn Jira in 10 mi...
Project settings
Archived issues NEW
Projects / SCRUM-1
Backlog
Search
Epic
Insights
View settings
SCRUM Sprint 1 11 Feb – 25 Feb (2 issues)
SCRUM-2 Save Product on WishList
SCRUM-3 View our WishList
Create issue
Backlog (0 issues)
SCRUM-1 / SCRUM-3
View our WishList
Add
Apps
Description
Add a description...
Child issues
Order by
Suggest subtasks
SCRUM-6 Page to display wishList
DONE
Confluence content
Product requirements
TRY TEMPLATE
Activity
Add a comment...
Pro tip: press M to comment
Done
Actions
Improve issue
Pinned fields
Click on the next to a field label to start pinning.
Details
Assignee
Unassigned
Assign to me
Labels
None
Parent
SCRUM-1 WishList
Team
None
Sprint
Quickstart
Create a project
Create an issue
Start with the basics
Invite your teammates
Connect your tools
The tools you use every day connect to Jira. Here are some you might want to connect now.
Slack
Microsoft Teams
GitHub
GitLab
CircleCI
Zendesk
Connect
Dismiss Quickstart


Atlassian uses cookies to improve your browsing experience, perform analytics and research, and conduct advertising. Accept all cookies to indicate that you agree to our use of cookies on your device. Atlassian cookies and tracking notice
Preferences
Only necessary
Accept all
Jira
Your work
Projects
Filters
Dashboards
Teams
Plans
Apps
Create
30 days left
Search
SCRUM-1
Software project
PLANNING
Summary
Timeline
Backlog
Board
Forms
Goals
Add view
DEVELOPMENT
Code
Project pages
Learn Jira in 10 mi...
Project settings
Archived issues NEW
Projects / SCRUM-1
Backlog
Search
Epic
Insights
View settings
SCRUM Sprint 1 11 Feb – 25 Feb (2 issues)
Complete sprint
SCRUM-2 Save Product on WishList
WISHLIST
IN PROGRESS
SCRUM-3 View our WishList
WISHLIST
DONE
Create issue
Backlog (0 issues)
Create sprint
Your backlog is empty.
Create issue
Quickstart
Create a project
Create an issue
Start with the basics
Invite your teammates
Connect your tools
The tools you use every day connect to Jira. Here are some you might want to connect now.
Slack
Microsoft Teams
GitHub
GitLab
CircleCI
Zendesk
Connect
Dismiss Quickstart



;

Conclusion

Creating a Scrum Board in JIRA helps teams collaborate effectively and manage sprints systematically. The Scrum Master plays a crucial role in ensuring the board is updated and used efficiently to track progress, resolve blockers, and enhance team productivity.

Salif Shaikh
T22 – 2201094

EXPERIMENT 8

TO STUDY PROJECT SCHEDULING USING GNATT CHART IN CLICKUP

Theory:

1. Introduction to Project Scheduling

Project scheduling is the process of defining tasks, setting deadlines, assigning responsibilities, and tracking progress to ensure timely project completion. A Gantt chart is a visual tool used for planning and scheduling tasks over a timeline.

1.1 What is a Gantt Chart?

A Gantt chart is a bar chart that represents project tasks and their durations. It helps project managers:

- Visualize task dependencies and overlaps.
- Monitor progress against deadlines.
- Allocate resources effectively.
- Identify bottlenecks early.

1.2 Features of a Gantt Chart

- Task dependencies (finish-to-start, start-to-start, etc.).
- Milestones to mark key deliverables.
- Critical path analysis to determine the longest sequence of dependent tasks.
- Progress tracking using percentage completion.

---

2. Introduction to ClickUp

ClickUp is an all-in-one project management tool that offers Gantt charts for scheduling, tracking, and managing tasks efficiently.

2.1 Why Use ClickUp for Project Scheduling?

- Intuitive drag-and-drop interface for adjusting timelines.
- Supports task dependencies and rescheduling.
- Real-time collaboration with team members.
- Automation and notifications to streamline workflows.

---

3. Creating a Gantt Chart in ClickUp 3.1 Steps to Create a Gantt Chart in ClickUp

1. Login to ClickUp at ClickUp Website.
2. Create a New Project:
    - Click Spaces > Create New List or Folder.

- Name your project and define the project scope.

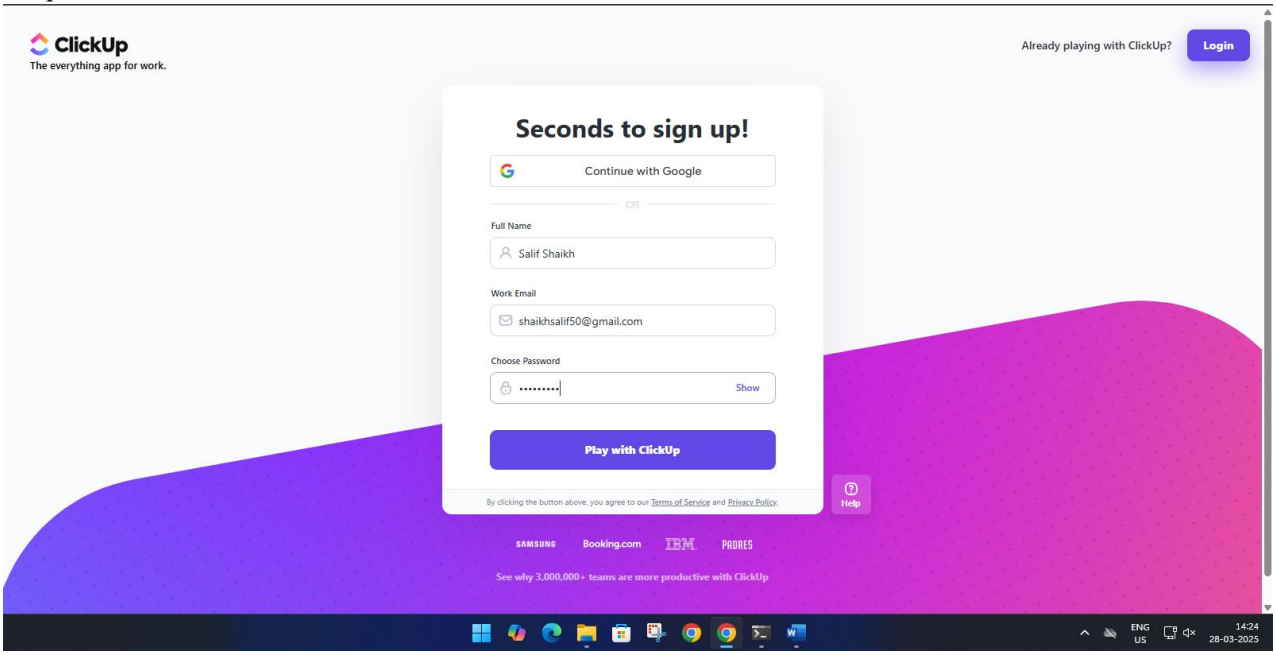
3. Add Tasks to the Project:
   - Click New Task, provide a task name, description, assignee, and due date.
   - Define task priorities and set dependencies.
4. Enable Gantt Chart View:
   - Go to View Options and select Gantt Chart.
   - Adjust start and due dates to organize the timeline.
5. Define Dependencies:
   - Click and drag connectors between tasks to create relationships (e.g., Task B starts after Task A).
6. Set Milestones:
   - Identify key deadlines and mark them as Milestones.

---

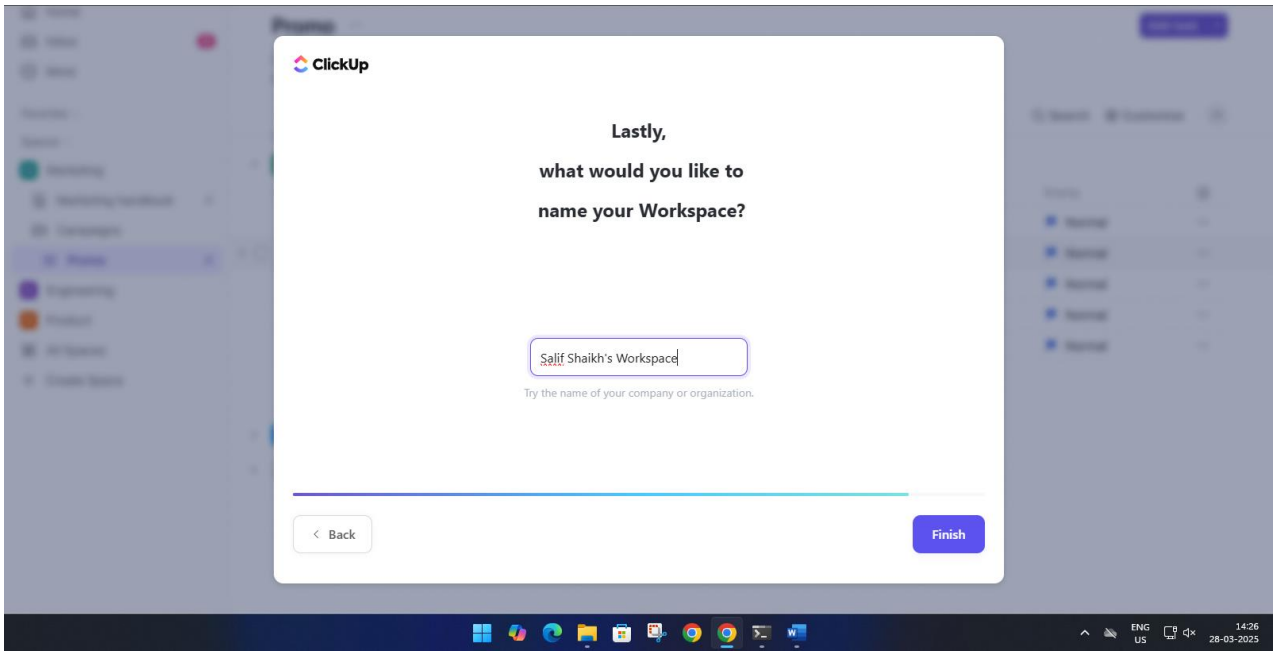
4. Tracking and Managing a Gantt Chart in ClickUp

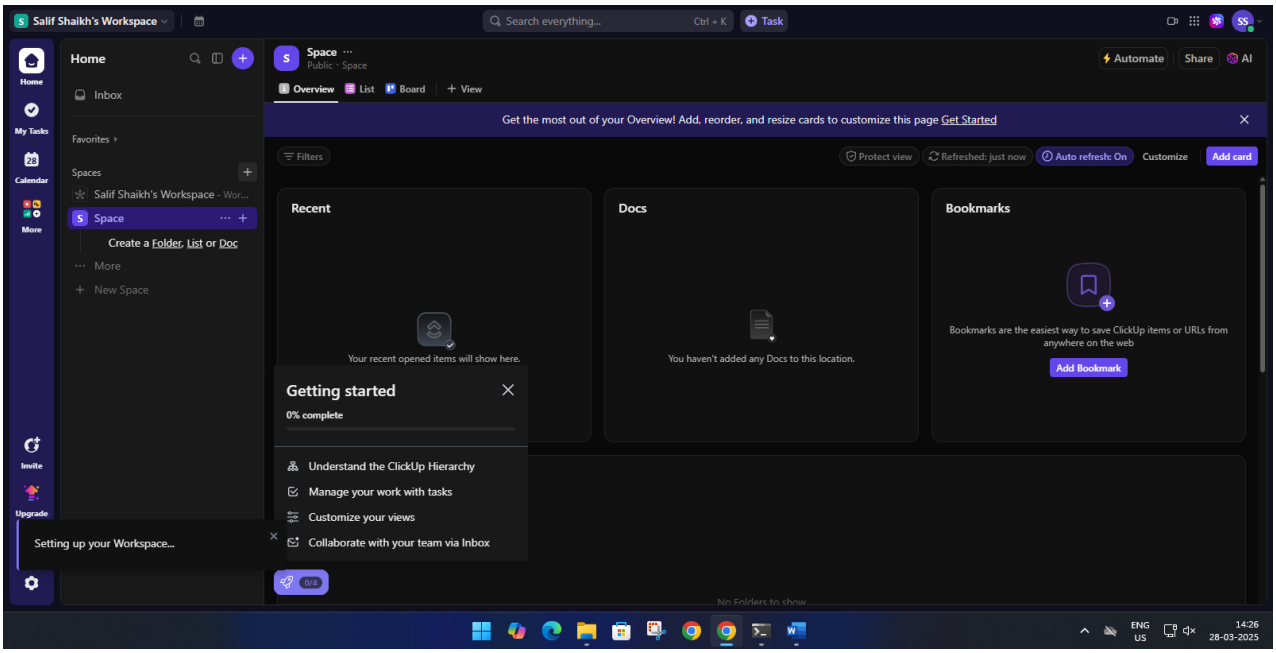
- Modify Timelines: Drag tasks to change deadlines dynamically.
- Monitor Progress: Use task completion percentages to track status.
- Adjust Dependencies: Update task sequences when project plans change.
- Generate Reports: Use ClickUp's reporting features to analyze workload and delays.

Output:

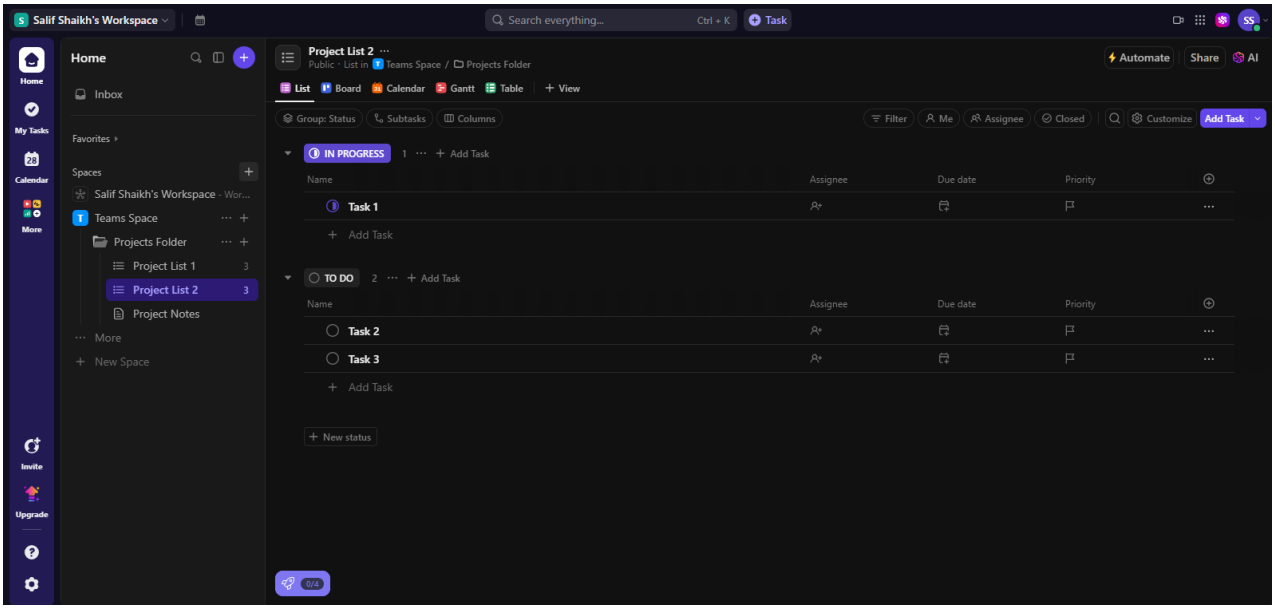


Salif Shaikh
T22 – 2201094





Salif Shaikh
T22 – 2201094



Conclusion

This experiment demonstrated the importance of project scheduling using a Gantt chart in ClickUp. By visualizing tasks, dependencies, and deadlines, teams can efficiently manage projects, avoid delays, and ensure smooth workflow execution.

Software Engineering & Project Management Lab Experiment No: - 09

Aim: To understand Docker Architecture and Container Life Cycle, install Docker and execute docker commands to manage images and interact with containers

Salif Shaikh

T22-94

AI&DS

Theory:

What is Docker?

Docker is a platform that allows developers to build, package, and deploy applications in lightweight, portable containers. These containers include everything needed to run an application, such as code, runtime, system tools, libraries, and dependencies.

Containerization Technology

Containers are isolated environments where applications run independently. Unlike traditional virtualization, which requires separate operating systems for each application, containers share the host OS kernel, making them faster and more efficient.

Docker Architecture

Docker follows a client-server architecture consisting of the following key components:

1. Docker Client

- It is the command-line interface (CLI) that allows users to interact with Docker.
- Commands such as docker run, docker build, and docker stop are executed through the client.

2. Docker Daemon (dockerd)

- It runs in the background and manages Docker containers, images, volumes, and networks.
- It listens for requests from the Docker Client and executes commands.

3. Docker Images

- A Docker image is a read-only template containing the application code, libraries, and dependencies.
- Images are created using Dockerfiles, which define the steps to build an image.
- Images are stored in Docker Hub or private repositories.

4. Docker Containers

- A container is an instance of a Docker image running as an isolated process on a host machine.
- Containers are lightweight, portable, and can be started, stopped, or removed as needed.

5. Docker Registry

- It is a storage system for Docker images.
- The public registry, Docker Hub, provides access to a vast collection of pre-built images.
- Users can also create private registries for security and control.

Docker Container Life Cycle

The **life cycle of a container** follows these steps:

1. **Create** – A container is created from an image using the docker create command.
2. **Start** – The container starts running using the docker start command.
3. **Run** – A new container can be started directly using docker run.
4. **Pause/Unpause** – Containers can be temporarily paused and resumed.
5. **Stop** – The container can be stopped using docker stop.
6. **Restart** – A stopped container can be restarted.
7. **Kill** – A container can be forcefully stopped using docker kill.
8. **Remove** – Containers that are no longer needed can be deleted using docker rm.

Software Engineering & Project Management Lab Experiment No: - 09

Aim: To understand Docker Architecture and Container Life Cycle, install Docker and execute docker commands to manage images and interact with containers

Benefits of Docker

1. Portability

- Containers can run on any platform that supports Docker.
- Applications behave consistently across different environments.

2. Efficiency

- Containers share the host OS kernel, reducing overhead and improving performance.
- They consume fewer resources compared to virtual machines.

3. Isolation

- Each container runs in its own isolated environment, preventing dependency conflicts.

4. Scalability

- Applications can be scaled up quickly by launching multiple containers.
- Docker enables automatic load balancing in large-scale deployments.

5. Consistency

- Ensures that the application runs the same way in development, testing, and production.
- Eliminates the "works on my machine" problem.

Docker Engine:

At the core of Docker is the Docker Engine, which is responsible for building, running, and managing containers. It consists of the Docker daemon, which manages containers, images, networks, and volumes, and the Docker client, which allows users to interact with the daemon through the Docker API.

Docker Images:

Software Engineering & Project Management Lab Experiment No: - 09

Aim: To understand Docker Architecture and Container Life Cycle, install Docker and execute docker commands to manage images and interact with containers

Docker images are read-only templates used to create containers. They contain the application code, runtime, libraries, dependencies, and other files needed to run the application. Images are built using Dockerfiles, which are text files that define the steps needed to create the image.

⛝ OUTPUT:-

Software Engineering & Project Management Lab Experiment No: - 09

Aim: To understand Docker Architecture and Container Life Cycle, install Docker and execute docker commands to manage images and interact with containers

```
C:\Users\202>docker run redis
Unable to find image 'redis:latest' locally
latest: Pulling from library/redis
8a1e25ce7c4f: Pull complete
8ab039a68e51: Pull complete
2b12a49dcfb9: Pull complete
cdf9868f47ac: Pull complete
e73ea5d3136b: Pull complete
890ad32c613f: Pull complete
4f4fb700ef54: Pull complete
ba517b76f92b: Pull complete
Digest: sha256:7dd707032d90c6eaafd566f62a00f5b0116ae08fd7d6cbbb0f311b82b47171a2
Status: Downloaded newer image for redis:latest
1:C 13 Mar 2024 03:19:03.928 * oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
1:C 13 Mar 2024 03:19:03.928 * Redis version=7.2.4, bits=64, commit=00000000, mod
1:C 13 Mar 2024 03:19:03.928 # Warning: no config file specified, using the defau
s.conf
1:M 13 Mar 2024 03:19:03.929 * monotonic clock: POSIX clock_gettime
1:M 13 Mar 2024 03:19:03.929 * Running mode=standalone, port=6379.
1:M 13 Mar 2024 03:19:03.929 * Server initialized
1:M 13 Mar 2024 03:19:03.929 * Ready to accept connections tcp
1:signal-handler (1710300105) Received SIGINT scheduling shutdown...
1:M 13 Mar 2024 03:21:45.877 * User requested shutdown...
1:M 13 Mar 2024 03:21:45.877 * Saving the final RDB snapshot before exiting.
1:M 13 Mar 2024 03:21:45.887 * DB saved on disk
1:M 13 Mar 2024 03:21:45.887 # Redis is now ready to exit, bye bye...
```

```
C:\Users\202>docker images
REPOSITORY   TAG       IMAGE ID       CREATED        SIZE
redis        latest    170a1e90f843   2 months ago   138MB
```

```
C:\Users\202>docker pull redis
Using default tag: latest
latest: Pulling from library/redis
Digest: sha256:7dd707032d90c6eaafd566f62a00f5b0116ae08fd7d6cbbb0f311b82b47171a2
Status: Image is up to date for redis:latest
docker.io/library/redis:latest
```

```
C:\Users\202>docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES

C:\Users\202>docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED              STATUS          PORTS      NAMES
052aecb0ee88   redis     "docker-entrypoint.s…"   About a minute ago   Up 4 seconds    6379/tcp   container121
1c4472744083   redis     "docker-entrypoint.s…"   6 minutes ago        Up 10 seconds   6379/tcp   modest_herschel

C:\Users\202>
```

Software Engineering & Project Management Lab Experiment No: - 09

Aim: To understand Docker Architecture and Container Life Cycle, install Docker and execute docker commands to manage images and interact with containers

```
C:\Users\202>docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED              STATUS          PORTS      NAMES
052aecb0ee88   redis     "docker-entrypoint.s…"   About a minute ago   Up 4 seconds    6379/tcp   container121
1c4472744083   redis     "docker-entrypoint.s…"   6 minutes ago        Up 10 seconds   6379/tcp   modest_herschel

C:\Users\202>
C:\Users\202>
C:\Users\202>docker stop 052aecb0ee88
052aecb0ee88

C:\Users\202>docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS      NAMES
1c4472744083   redis     "docker-entrypoint.s…"   9 minutes ago   Up 2 minutes   6379/tcp   modest_herschel
```

```
C:\Users\202>docker start 052aecb0ee88
052aecb0ee88

C:\Users\202>docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS      NAMES
052aecb0ee88   redis     "docker-entrypoint.s…"   5 minutes ago    Up 8 seconds    6379/tcp   container121
1c4472744083   redis     "docker-entrypoint.s…"   10 minutes ago   Up 3 minutes    6379/tcp   modest_herschel
```

```
C:\Users\202>docker rm 052aecb0ee88
052aecb0ee88
```

```
C:\Users\202>docker images
REPOSITORY   TAG       IMAGE ID       CREATED         SIZE
redis        latest    170a1e90f843   2 months ago    138MB
```

```
C:\Users\202>docker exec -d 1c4472744083 touch /tmp/execWorks

C:\Users\202>docker exec -it 1c4472744083 bash
root@1c4472744083:/data#
```

```
C:\Users\202>docker restart 1c4472744083
1c4472744083

C:\Users\202>docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS         PORTS      NAMES
1c4472744083   redis     "docker-entrypoint.s…"   13 minutes ago   Up 3 seconds   6379/tcp   modest_herschel
```

Software Engineering & Project Management Lab Experiment No: - 09

Aim: To understand Docker Architecture and Container Life Cycle, install Docker and execute docker commands to manage images and interact with containers

```
C:\Users\202>docker inspect 1c4472744083
[
    {
        "Id": "1c44727440831475b093dcbf93163064b819bdd9ad8378bb3a4fa847dc411d80",
        "Created": "2024-03-13T03:19:03.418741433Z",
        "Path": "docker-entrypoint.sh",
        "Args": [
            "redis-server"
        ],
        "State": {
            "Status": "running",
            "Running": true,
            "Paused": false,
            "Restarting": false,
            "OOMKilled": false,
            "Dead": false,
            "Pid": 2112,
            "ExitCode": 0,
            "Error": "",
            "StartedAt": "2024-03-13T03:32:13.750463204Z",
            "FinishedAt": "2024-03-13T03:32:13.145321277Z"
        },
        "Image": "sha256:170a1e90f8436daa6778aeea3926e716928826c215ca23a8dfd8055f663f9428",
        "ResolvConfPath": "/var/lib/docker/containers/1c44727440831475b093dcbf93163064b819bdd9a
```

```
C:\Users\202>docker commit 1c4472744083 new_image_name:redis2
sha256:33e4284a7e92a4a1331555d01f6e078fc496e3a3ed8eb7f84f2678261ad07e83

C:\Users\202>docker images
REPOSITORY       TAG       IMAGE ID       CREATED          SIZE
new_image_name   redis2    33e4284a7e92   4 seconds ago    138MB
new_image_name   tag       61ab016507fa   36 seconds ago   138MB
redis            latest    170a1e90f843   2 months ago     138MB
```

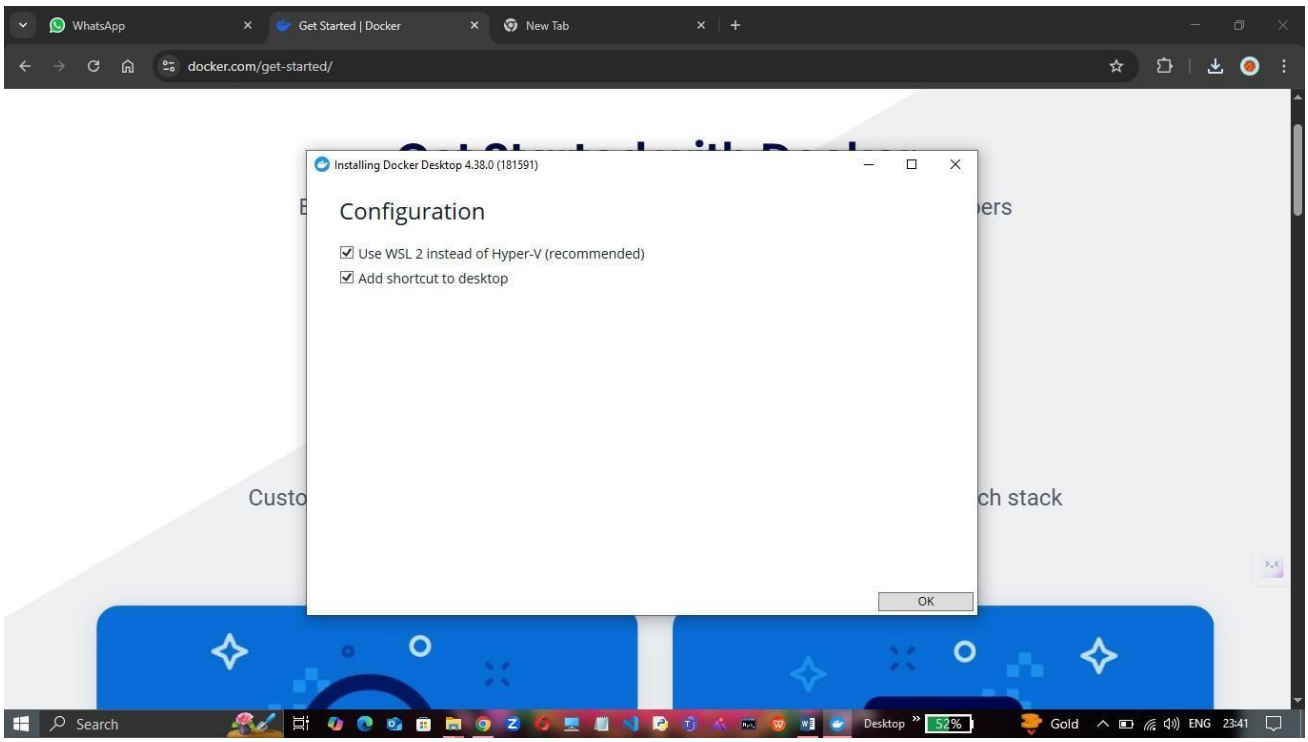
Software Engineering & Project Management Lab Experiment No: - 09

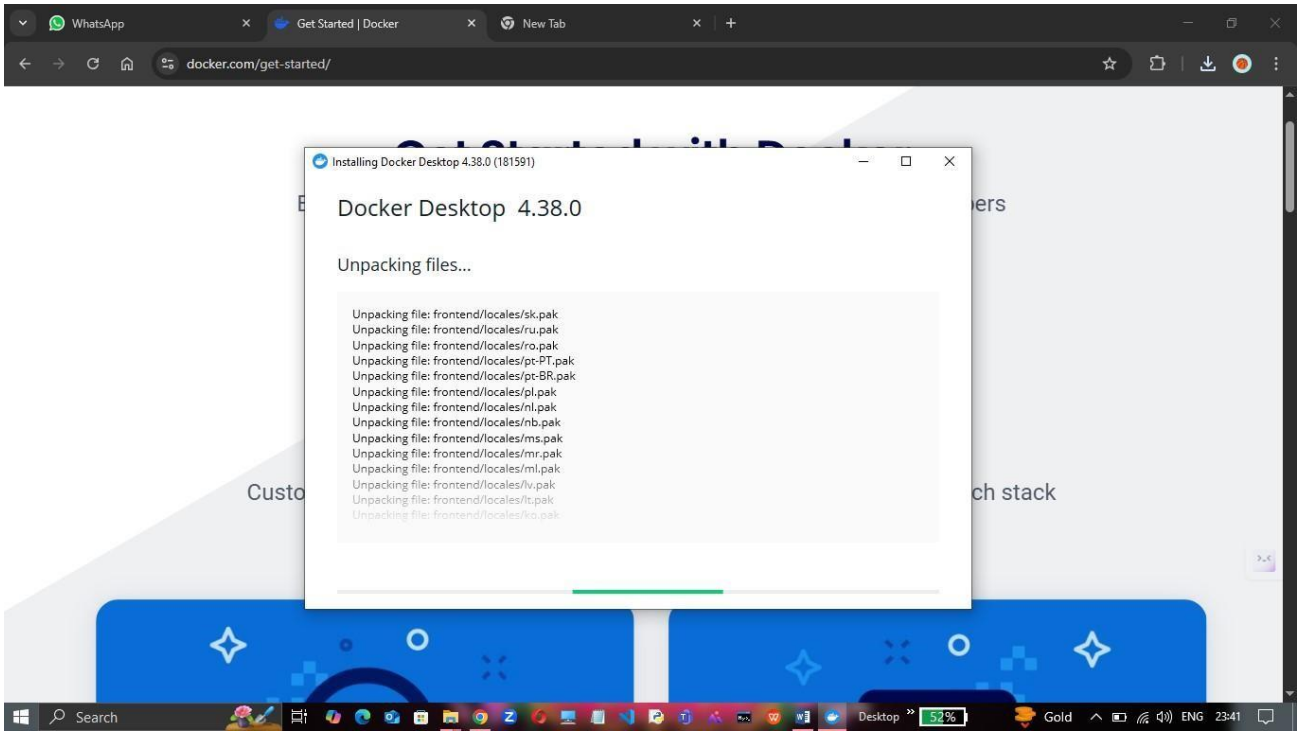
Aim: To understand Docker Architecture and Container Life Cycle, install Docker and execute docker commands to manage images and interact with containers

SCREENSHOTS:

Software Engineering & Project Management Lab Experiment No: - 09

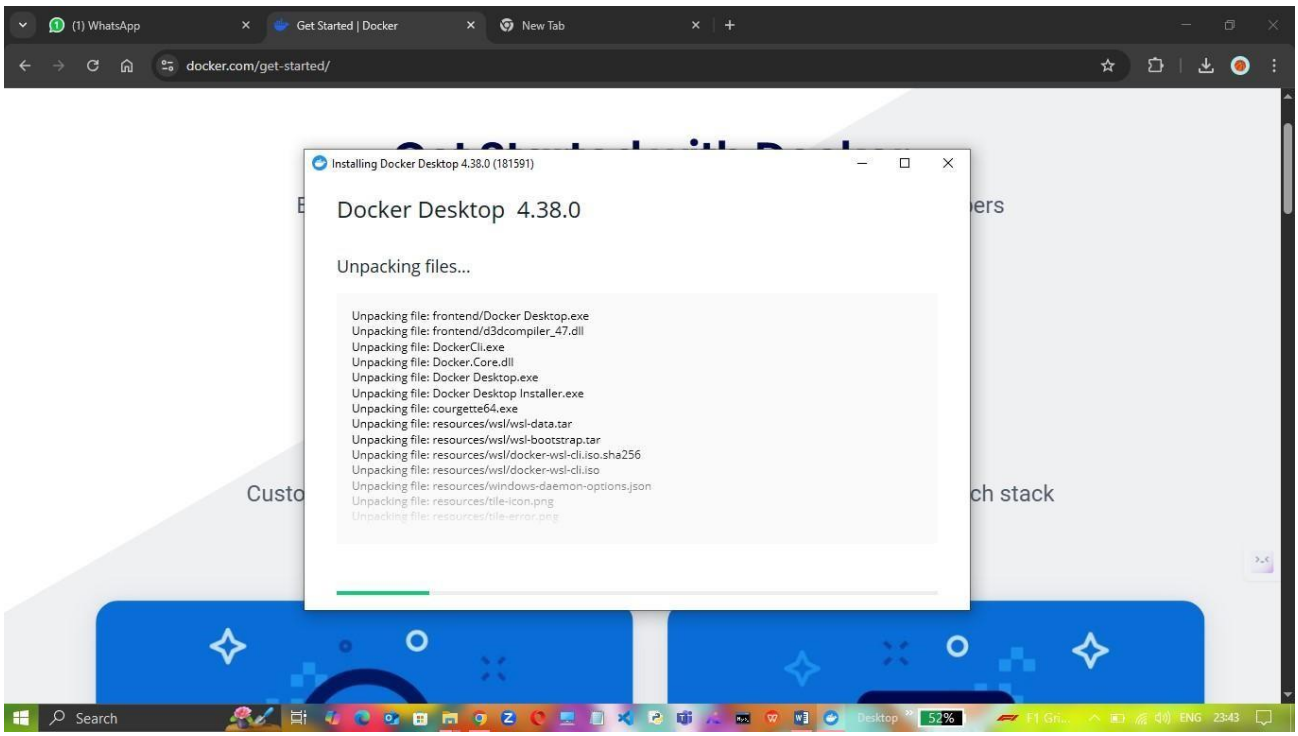
Aim: To understand Docker Architecture and Container Life Cycle, install Docker and execute docker commands to manage images and interact with containers

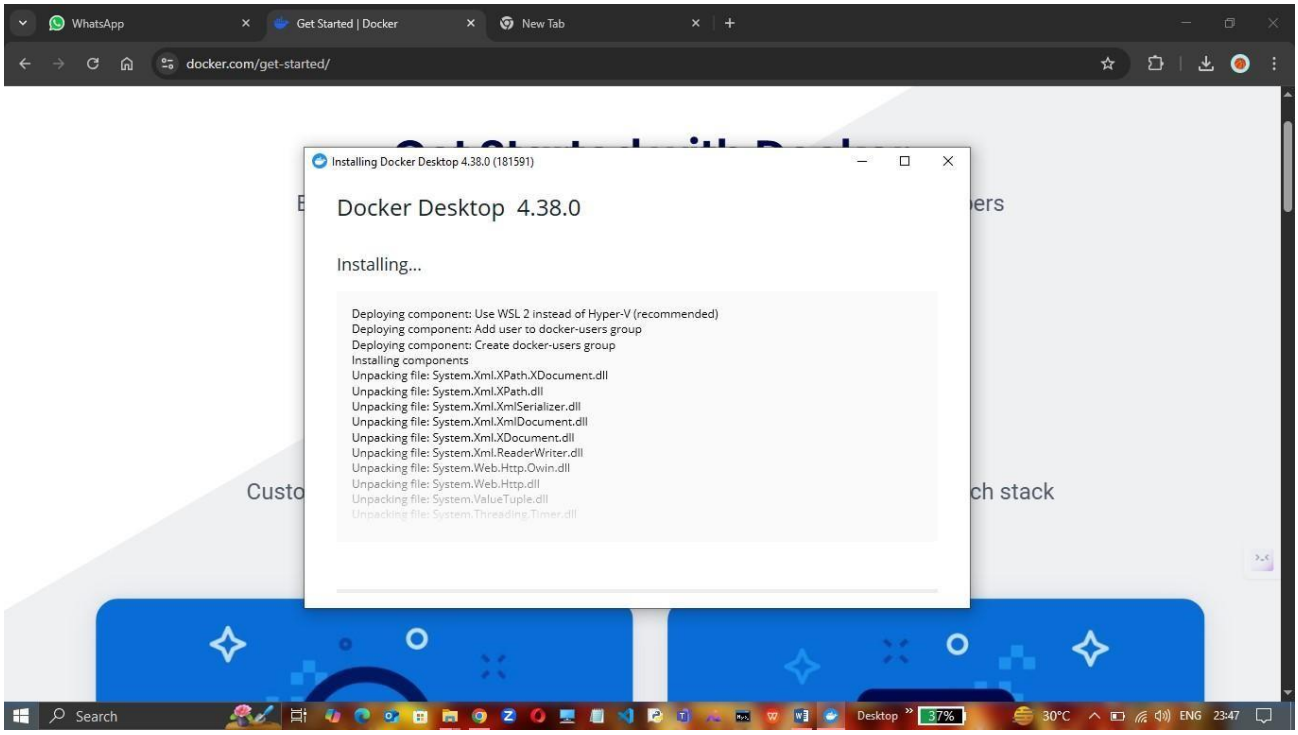




Software Engineering & Project Management Lab Experiment No: - 09

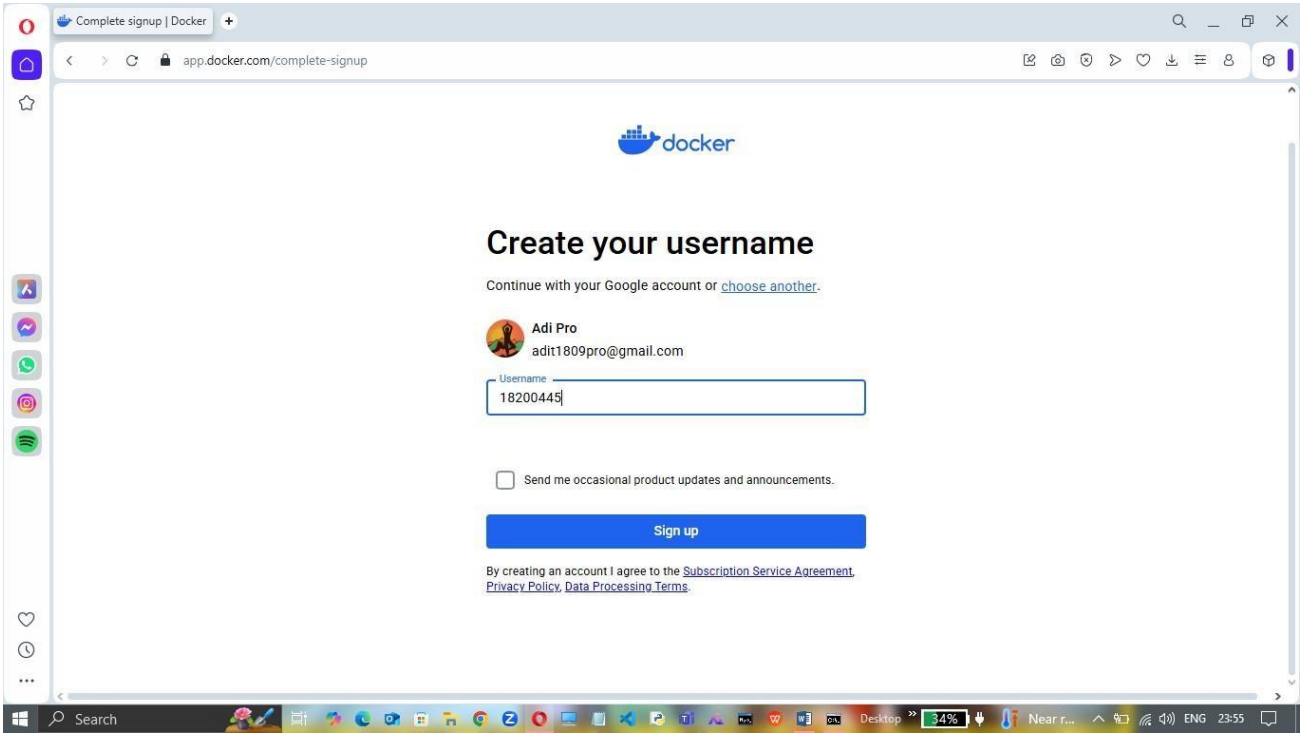
Aim: To understand Docker Architecture and Container Life Cycle, install Docker and execute docker commands to manage images and interact with containers





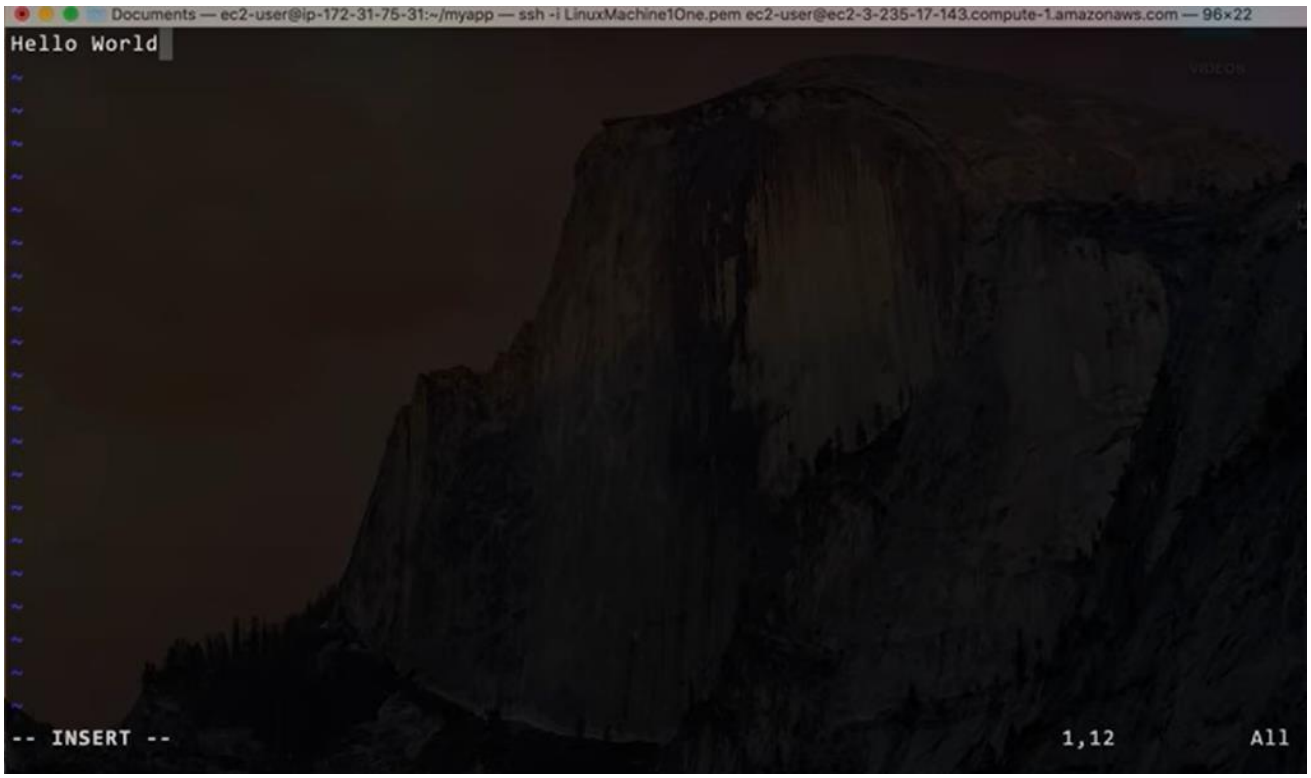
Software Engineering & Project Management Lab Experiment No: - 09

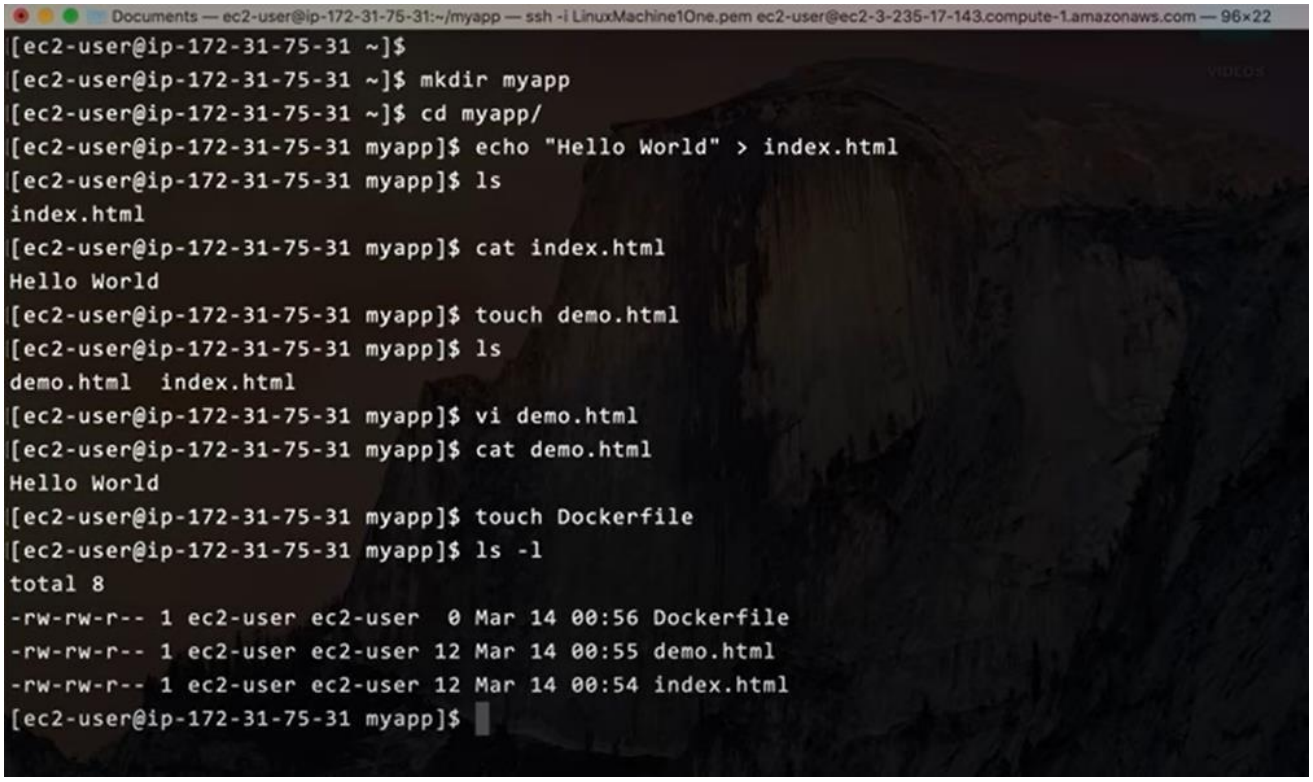
Aim: To understand Docker Architecture and Container Life Cycle, install Docker and execute docker commands to manage images and interact with containers



Software Engineering & Project Management Lab Experiment No: - 09

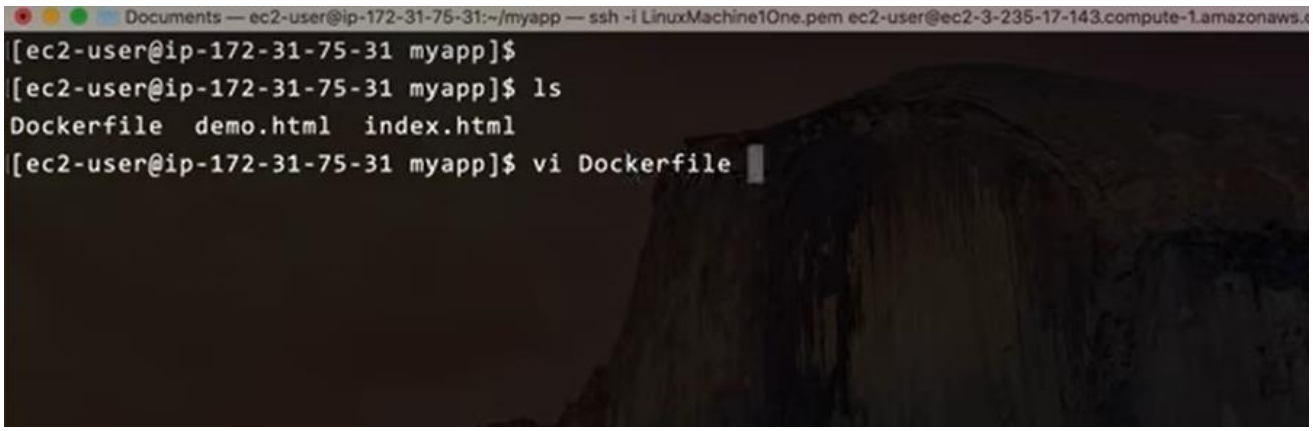
Aim: To understand Docker Architecture and Container Life Cycle, install Docker and execute docker commands to manage images and interact with containers

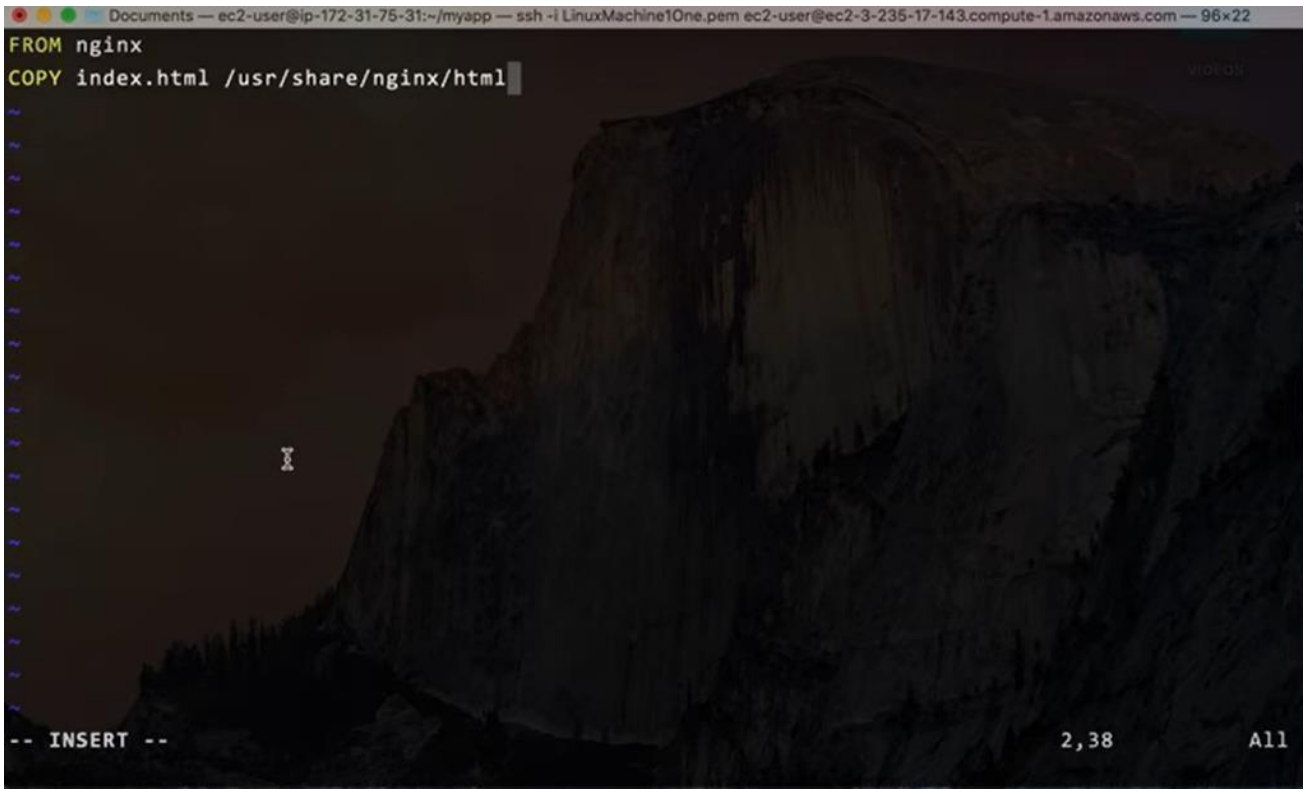




Software Engineering & Project Management Lab Experiment No: - 09

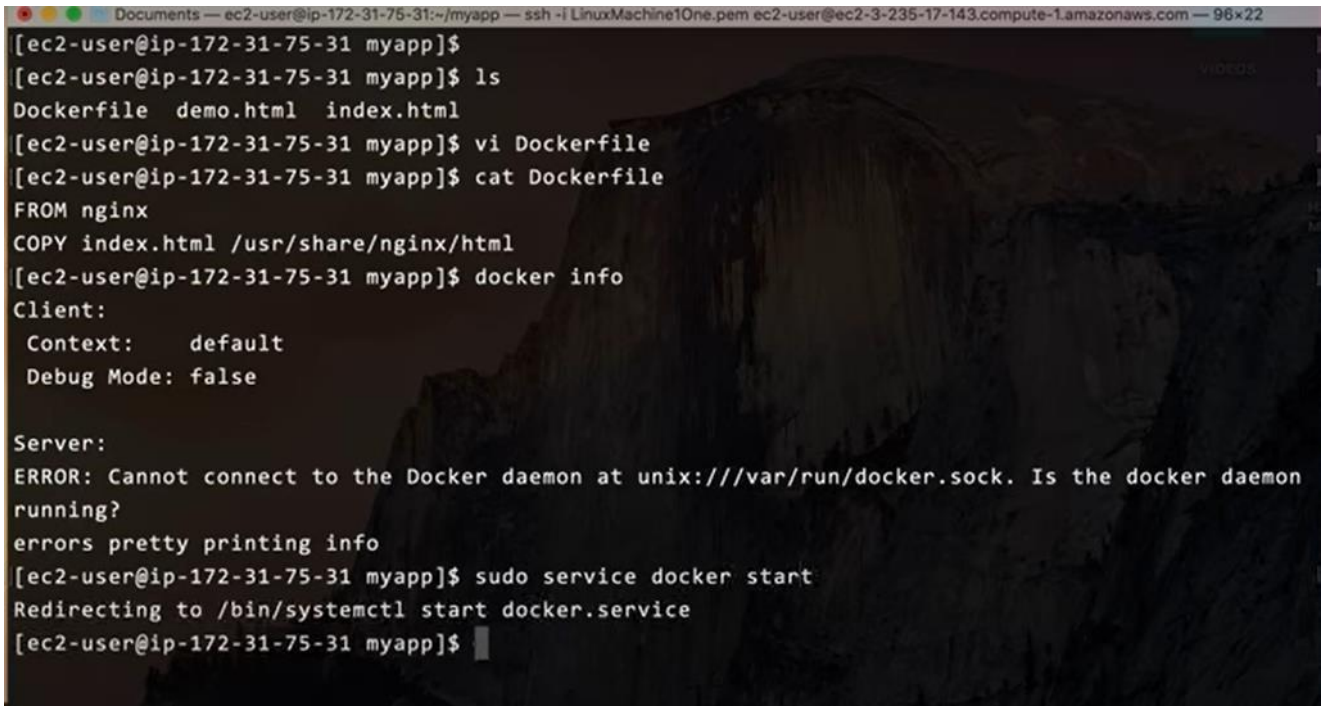
Aim: To understand Docker Architecture and Container Life Cycle, install Docker and execute docker commands to manage images and interact with containers

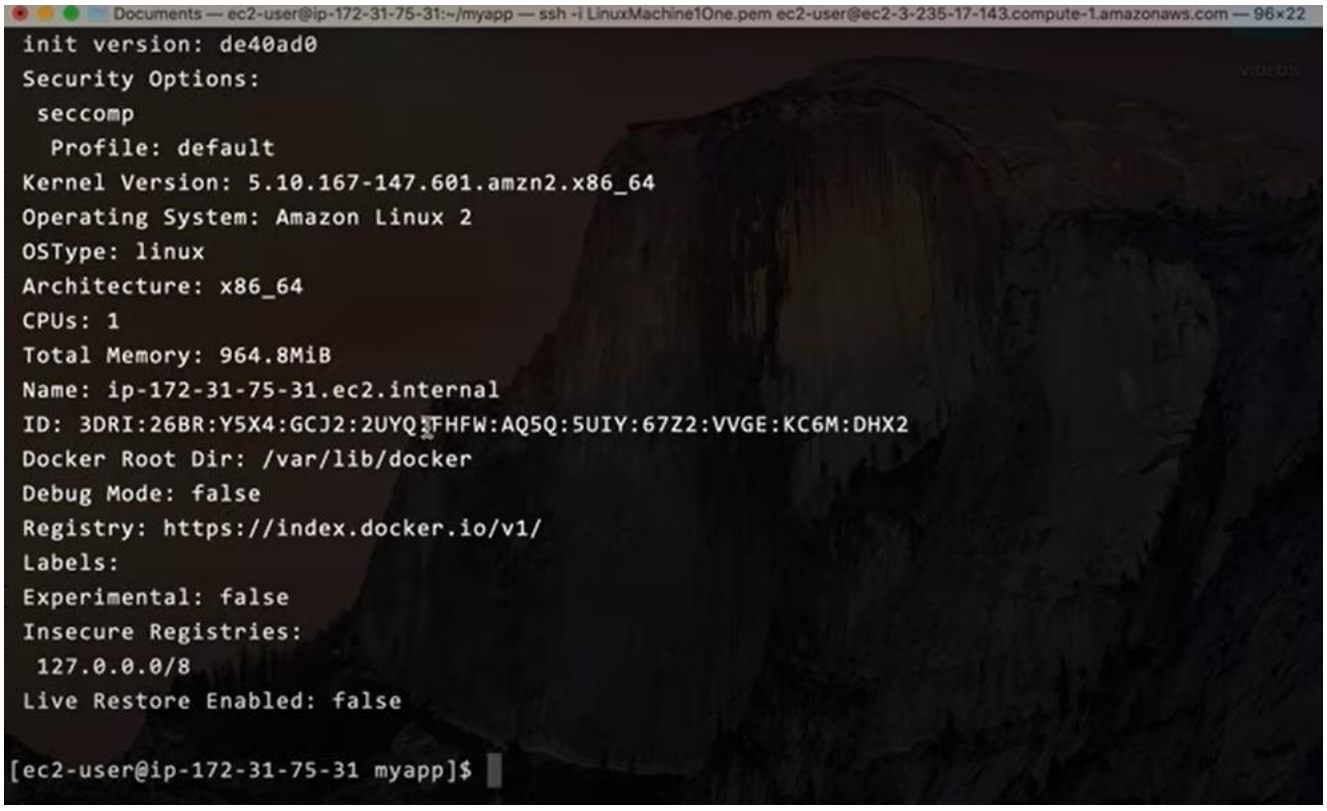




Software Engineering & Project Management Lab Experiment No: - 09

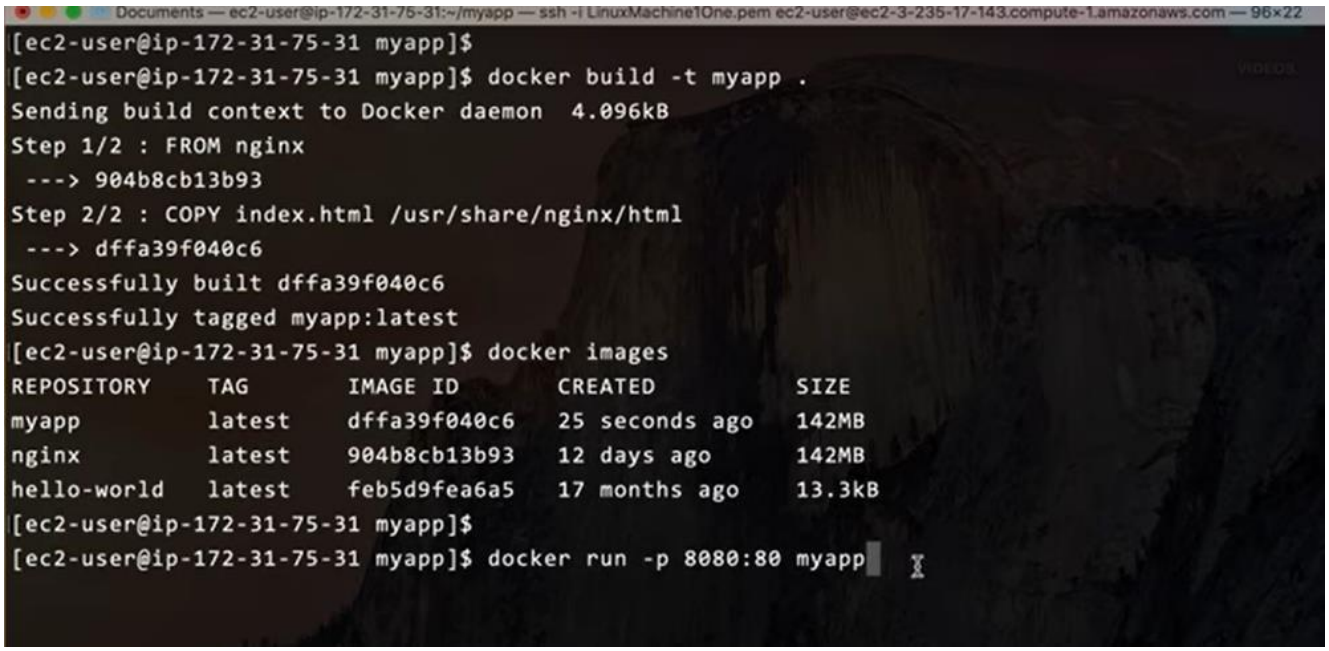
Aim: To understand Docker Architecture and Container Life Cycle, install Docker and execute docker commands to manage images and interact with containers

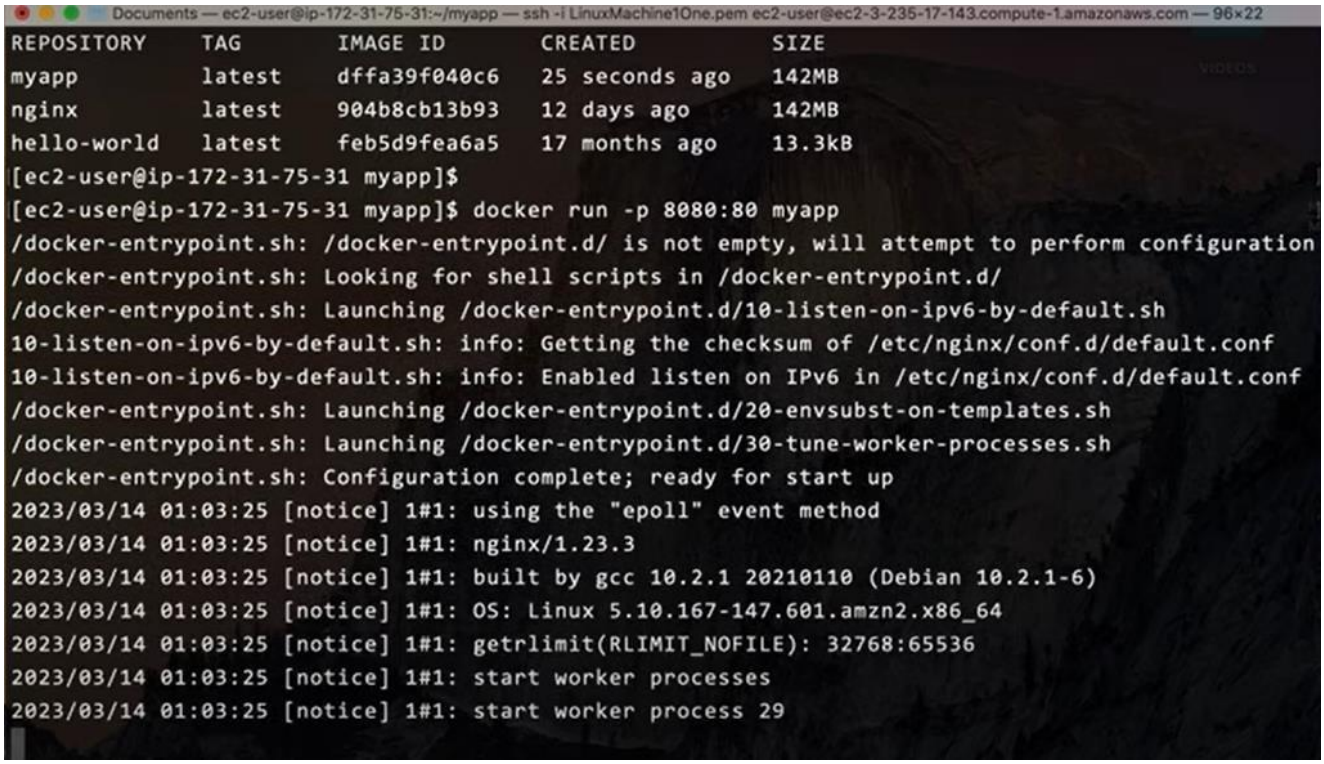




Software Engineering & Project Management Lab Experiment No: - 09

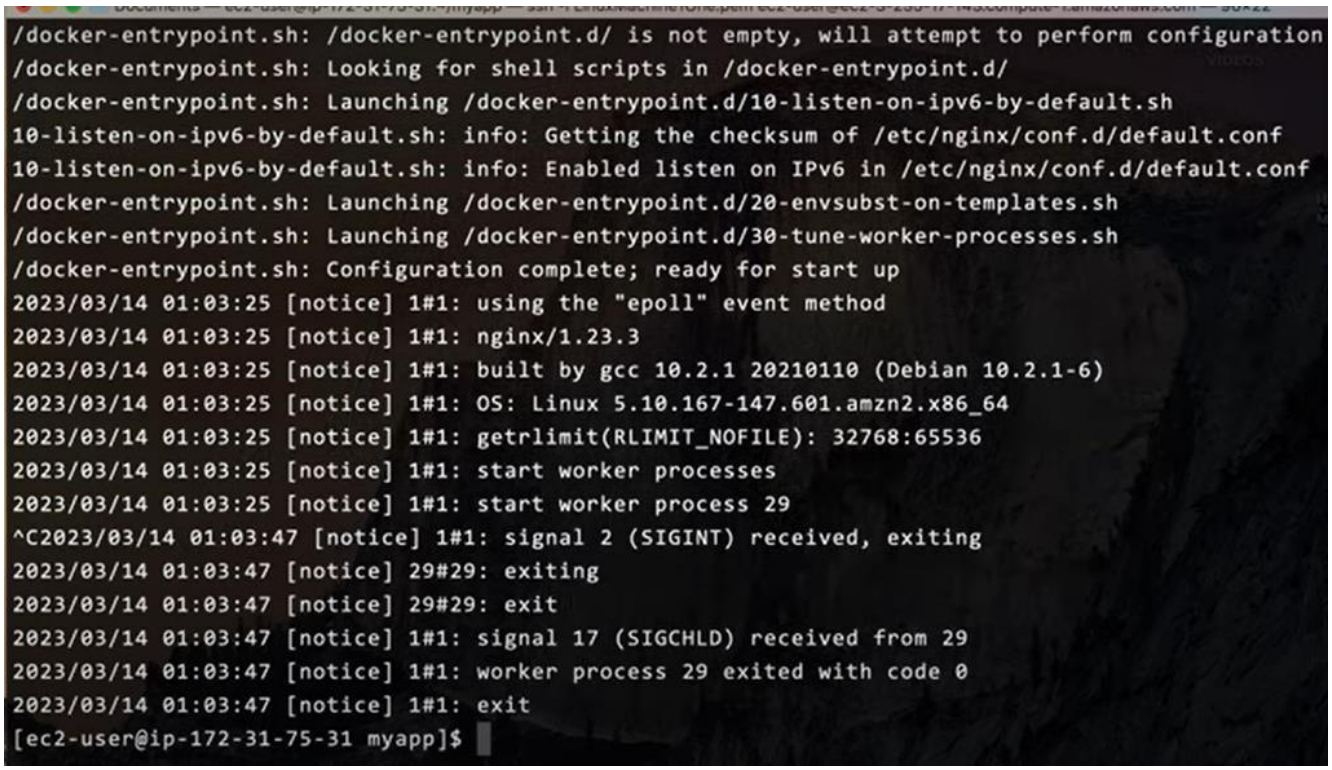
Aim: To understand Docker Architecture and Container Life Cycle, install Docker and execute docker commands to manage images and interact with containers





Software Engineering & Project Management Lab Experiment No: - 09

Aim: To understand Docker Architecture and Container Life Cycle, install Docker and execute docker commands to manage images and interact with containers



<u>Conclusion</u>: Thus, we have successfully installed Docker and execute docker commands to manage images and interact with containers.

Software Engineering & Project Management Lab Experiment No: - 10

Aim: To learn Dockerfile instructions, build an image for a sample web application using DOCKERFILE.

Salif Shaikh

T22-94

AI&DS

Theory:

What is Docker?

Docker is a platform that allows developers to build, package, and deploy applications in lightweight, portable containers. These containers include everything needed to run an application, such as code, runtime, system tools, libraries, and dependencies.

Benefits of Docker

1. Portability

- Containers can run on any platform that supports Docker.
- Applications behave consistently across different environments.

2. Efficiency

- Containers share the host OS kernel, reducing overhead and improving performance.
- They consume fewer resources compared to virtual machines.

3. Isolation

- Each container runs in its own isolated environment, preventing dependency conflicts.

4. Scalability

- Applications can be scaled up quickly by launching multiple containers.
- Docker enables automatic load balancing in large-scale deployments.

5. Consistency

- Ensures that the application runs the same way in development, testing, and production.
- Eliminates the "works on my machine" problem.

Software Engineering & Project Management Lab Experiment No: - 10

Aim: To learn Dockerfile instructions, build an image for a sample web application using DOCKERFILE.

DIRECTIVE argument

Although the DIRECTIVE is case-insensitive, it is recommended to write all directives in uppercase to differentiate them from arguments. A Dockerfile usually consists of multiple lines

of instructions that are executed sequentially by the Docker engine during the image-building process.

Now that you understand the purpose of a Dockerfile, let's get our hands dirty by building one for a sample Python application.

Building a Dockerfile for a Sample Python/Flask Application The application we'll be working with is a simple Flask app with only one home route that returns Hello, World!.

Let's start by setting up the Flask application. Open your favorite code editor and create a new directory for your project. In this directory, create a new file named app.py and add the following Python code:

```
from flask import Flask
app = Flask(__name__)
@app.route('/')
def hello():
    return 'Hello, World!'
```

Now, let's build the Dockerfile.

In the same directory as your app.py, create a new file named Dockerfile (with no file extension). This is where you'll write the instructions for Docker to build your image.

Now, follow the steps below to create the Dockerfile:

Software Engineering & Project Management Lab Experiment No: - 10

Aim: To learn Dockerfile instructions, build an image for a sample web application using DOCKERFILE.

#1: Specify the base image

The very first instruction you write in a Dockerfile must be the FROM directive, which specifies the base image. The base image is the image from which all other layers in your Docker image will be built. It's the foundation of your Docker image, much like the foundation of a building. Add the following instruction to your Dockerfile:

```
FROM python:3.11-slim
```

Here, we're telling Docker to use the official Python Docker image, and more specifically, the 3.11-slim version. This slim variant of Python Docker image is a minimal version that excludes some packages to make the image smaller. Note that the base image you specify will be downloaded from Docker Hub, Docker's official image registry. The reason we're using Python as the base image is that the application containerization is written in Python. The Python base image includes a Python interpreter, which is necessary to run Python code, as well as a number of commonly used Python utilities and libraries. By using the Python base image, we're ensuring that the environment within the Docker container is preconfigured for running Python code.

#2 Set the working directory

Once you've chosen the base image, the next step is to determine the working directory using the WORKDIR directive. Insert the following line after the FROM directive:

```
WORKDIR /app
```

Here, we're telling Docker to create a directory named app in the Docker container and use it as the current working directory. All instructions that FROM python:3.11-slim WORKDIR /app follow (like RUN, COPY, and CMD) will be run in this directory inside the container. Think of this as typing the command cd /app in a terminal to change the current working directory to /app. The difference here is that it's being done within the Docker container as part of the build process. A working directory within the container is necessary because it designates a specific

location for our application code within the container and determines where commands will be run from. If we don't set a working directory, Docker won't have a clear context for where your application is located, which would make it harder to interact with.

#3 Install dependencies Once the working directory is set, the next step is to install the dependencies. Our Python application relies on the Flask web framework, which manages requests, routes URLs, and handles other web-related tasks. To install Flask, add the following instruction in your Dockerfile just under the WORKDIR directive:

```
RUN pip install flask==2.3
```

Here, we're instructing Docker to use pip (a package installer for Python) to install the specific version of Flask we need for our application.

#4 Copy application files to the container After setting up the working directory and installing the necessary dependencies, we're now ready to copy the application files into the Docker container. To do this, add the following instruction just below the RUN directive:

```
COPY . /app
```

This line copies everything in the current directory (denoted by ".") on our host machine into the /app directory we previously set as our working directory within the Docker container. It's like using the cp command in the terminal to copy files from one directory to another, but in this context, it's copying files from your local machine to the Docker container. Why do we need to do this? It's simple. Without this step, the Docker container wouldn't have access to our application's code, making it impossible to run our app.

#5 Specify the environment variable Once the application files are copied, we need to set up the FLASK_APP environment variable for our Docker container using the ENV directive. Now, you may be wondering why we need this environment variable in the first place. In our app.py file, we create an instance of the Flask application and assign it to the variable app. This

application instance is what Flask needs to run, and it's located in the app.py file. When starting our Flask application using the flask run command (which we'll discuss in the next section), Flask must know where to locate the application instance to run. Flask uses the FLASK_APP environment variable to find this instance. Hence, we need to use the ENV directive to set the value of FLASK_APP to app.py. To do this, add the following line under the RUN directive:

```
ENV FLASK_APP=app.py
```

This line ensures Flask knows exactly where to find the application instance to run, which in our case is app.py.

#6 Define the default command The last instruction that we need for our application is to specify the default command that will be executed when the Docker container starts:

```
CMD ["flask", "run", "--host=0.0.0.0", "--port=5000"]
```

from the image, we'll build from this Dockerfile.

Insert the following instruction below the ENV directive:

Here's what each part of the argument passed to the CMD directive does:

- flask: This is the program that we want to run. In this case, it's the Flask commandline interface.
- run: This command instructs Flask to start a local development server.
- --host=0.0.0.0: This argument tells the Flask server to listen on all public IPs. In the context of Docker, this means the Flask application will be accessible on any IP address that can reach the Docker container.
- --port=5000: This argument specifies the port number that the Flask server will listen on. Port 5000 is the default port for Flask, but it's good practice to explicitly declare it for clarity.

After this, our Dockerfile is ready. It should look like this:

Aim: To learn Dockerfile instructions, build an image for a sample web application using DOCKERFILE.

```
FROM python:3.11-slim
WORKDIR /app
RUN pip install flask==2.3
COPY . /app
ENV FLASK_APP=app.py
CMD ["flask", "run", "--host=0.0.0.0", "--port=5000"]
```

It's worth noting that the directives we used in the Dockerfile for our Python app aren't the only ones available in Docker. But they are the ones you'll often encounter when working with Dockerfiles.

#7 Create a .dockerignore file Before we go ahead and build our Docker image, we need to take care of one last thing. Remember the following COPY directive?

```
COPY . /app
```

This line instructs Docker to copy everything from our current directory to the app directory inside the container, which includes the Dockerfile itself. But, the Dockerfile isn't required for our app to work—it's just for us to create the Docker image. So, we need to ensure that the Dockerfile doesn't get copied to the app directory in the container. Here's how we do it: Create a new file called .dockerignore in the same directory as your Dockerfile. This file works much like a .gitignore file if you're familiar with Git. Then, add the word Dockerfile to this file. This tells Docker to ignore the Dockerfile when copying files into the container. Now that we've prepared everything, it's time to build our Docker image, run a container from this image, and test our application to see if everything works as expected.

Aim: To learn Dockerfile instructions, build an image for a sample web application using DOCKERFILE.

Building and running the Docker Image

Open a terminal and navigate to the directory where your Dockerfile is located. Now, run the following command to create an image named sample-flask-app:v1 (you can name the image anything you prefer):

```
$ docker build . -t sample-flask-app:v1
```

In the command above, the dot (.) after the build command indicates that the current directory is the build context. We're using the -t flag to tag the Docker image with the name sampleflaskapp and version v1. After running this command, you'll see an output similar to this:

Aim: To learn Dockerfile instructions, build an image for a sample web application using DOCKERFILE.

```
$ docker build . -t sample-flask-app:v1
[+] Building 17.7s (10/10) FINISHED docker:default
 => [internal] load build definition from Do  0.0s
 => => transferring dockerfile: 192B          0.0s
 => [internal] load metadata for docker.io/l  3.5s
 => [auth] library/python:pull token for reg  0.0s
 => [internal] load .dockerignore             0.0s
 => => transferring context: 50B              0.0s
 => [1/4] FROM docker.io/library/python:3.11  9.0s
 => => resolve docker.io/library/python:3.11  0.0s
 => => sha256:1103112ebfc46e 3.51MB / 3.51MB  1.9s
 => => sha256:b4b80ef7128d 12.87MB / 12.87MB  2.7s
 => => sha256:a2eb07f336e4f1 1.65kB / 1.65kB  0.0s
 => => sha256:4bcdb5d5bc81ca 1.37kB / 1.37kB  0.0s
 => => sha256:15646a3fa12dde 6.93kB / 6.93kB  0.0s
 => => sha256:8a1e25ce7c4f 29.12MB / 29.12MB  4.8s
 => => sha256:cc7f04ac52f8a3bad5 243B / 243B  2.4s
 => => sha256:87b8bf94a2ace2 3.41MB / 3.41MB  3.3s
 => => extracting sha256:8a1e25ce7c4f75e372e  1.8s
 => => extracting sha256:1103112ebfc46e01c0f  0.2s
 => => extracting sha256:b4b80ef7128dc9bd114  1.0s
 => => extracting sha256:cc7f04ac52f8a3bad5b  0.0s
 => => extracting sha256:87b8bf94a2ace2b005d  0.7s
 => [internal] load build context             0.0s
 => => transferring context: 194B             0.0s
 => [2/4] WORKDIR /app                        0.2s
 => [3/4] RUN pip install flask==2.3          4.7s
 => [4/4] COPY . /app                         0.0s
 => exporting to image                        0.2s
 => => exporting layers                       0.2s
 => => writing image sha256:c6879156c7750c89  0.0s
 => => naming to docker.io/library/sample-fl  0.0s

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

To make sure the image sample-flask-app:v1 has been successfully created, run the following command to check the list of Docker images on your system:

```
$ docker image ls
```

Aim: To learn Dockerfile instructions, build an image for a sample web application using DOCKERFILE.

The resulting output should look something like this:

```
$ docker image ls
REPOSITORY          TAG       IMAGE ID       CREATED          SIZE
sample-flask-app    v1        c6879156c775   10 seconds ago   147MB
mongo               latest    24041ceefc56   6 days ago       755MB
```

In the list, you should see sample-flask-app:v1, which confirms the image is now in our system. Now, run the sample-flask-app:v1 image as a container by executing the following command:

```
$ docker container run -d -p 5000:5000 sample-flask-app:v1
```

The -d flag is short for --detach and runs the container in the background. The -p flag is short for --publish and maps port 5000 of the host to port 5000 of the Docker container. After running this command, you'll see an output like this:

```
$ docker container run -d -p 5000:5000 sample-flask-app:v1
ff37071dd4cef95cc1dc2ce7e145019339cfaec54575659f72aea4e560238f8c
```

The long string you see printed in the terminal is the container ID. To make sure the container is running, list the currently active Docker containers by running the following command:

```
$ docker container ls
```

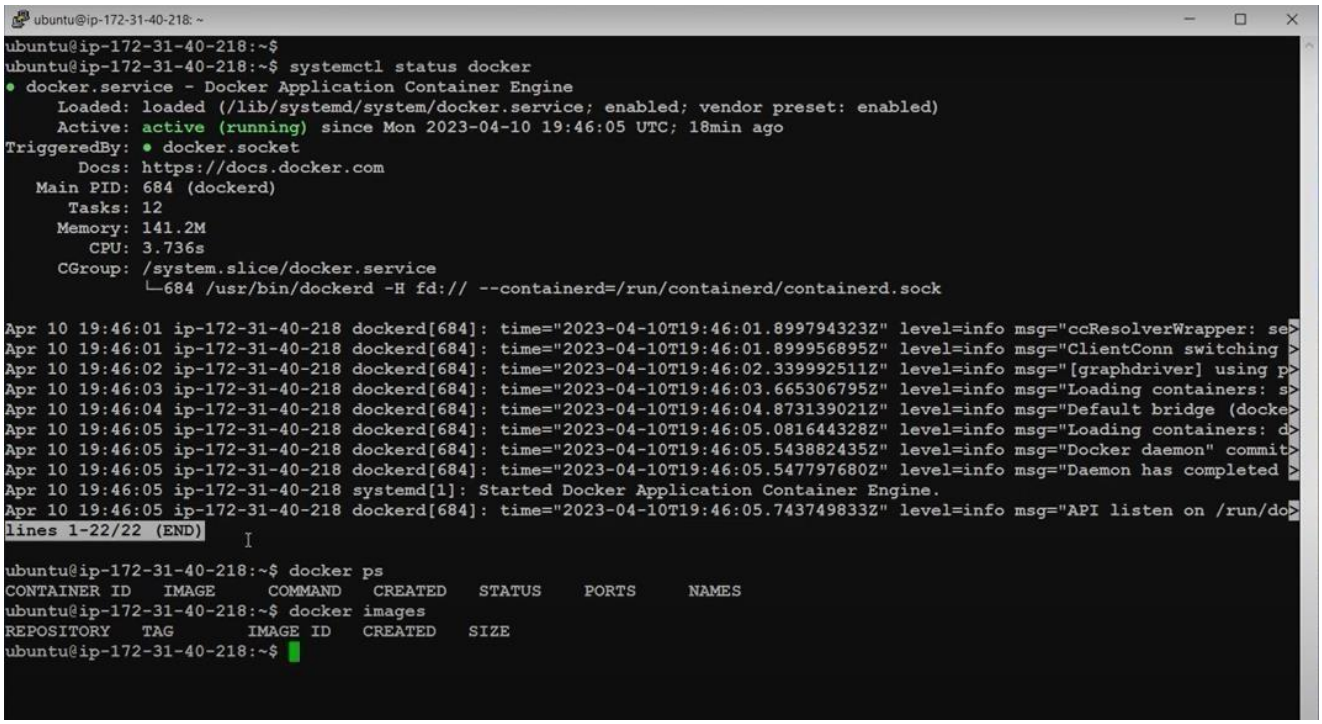
You should see something like this:

```
$ docker container ls
CONTAINER ID  IMAGE              COMMAND                  CREATED         STATUS         PORTS                    NAMES
c301c50152ac  sample-flask-app:v1  "flask run --host=0…"  14 seconds ago  Up 12 seconds  0.0.0.0:5000->5000/tcp  xenodochial_mendel
```

Aim: To learn Dockerfile instructions, build an image for a sample web application using DOCKERFILE.

The container is up and running as expected. Our Flask application is now running inside the container. To test it, open a web browser and go to http://localhost:5000. You should see the message Hello, World! displayed like this:

SCREENSHOTS:



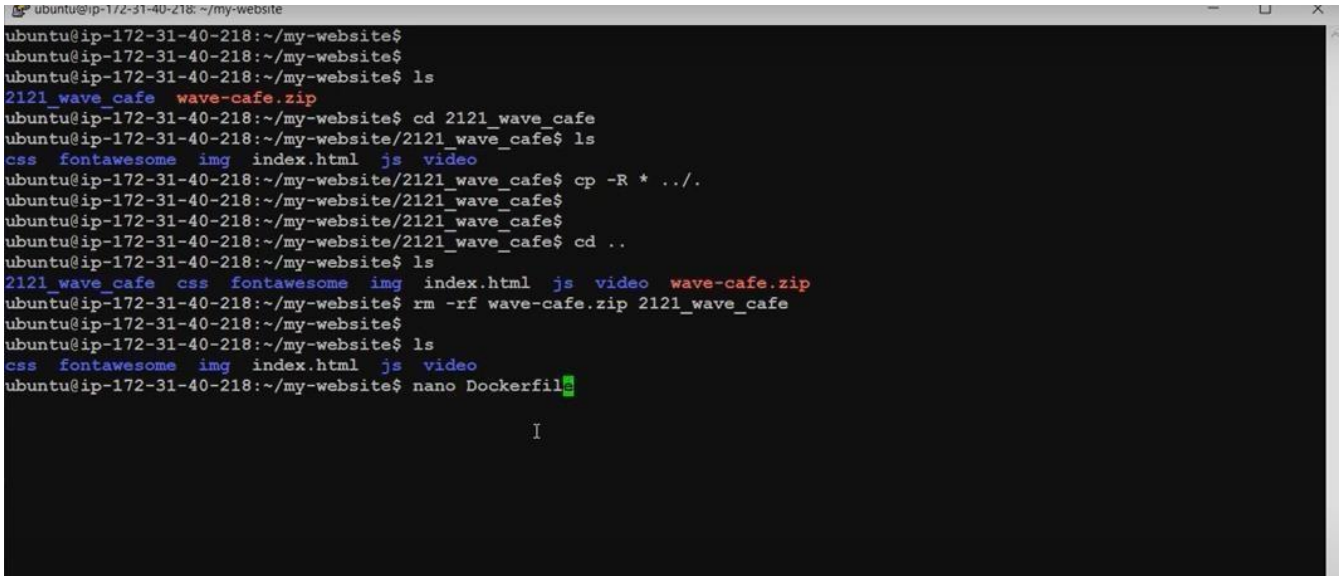
Software Engineering & Project Management Lab Experiment No: - 10

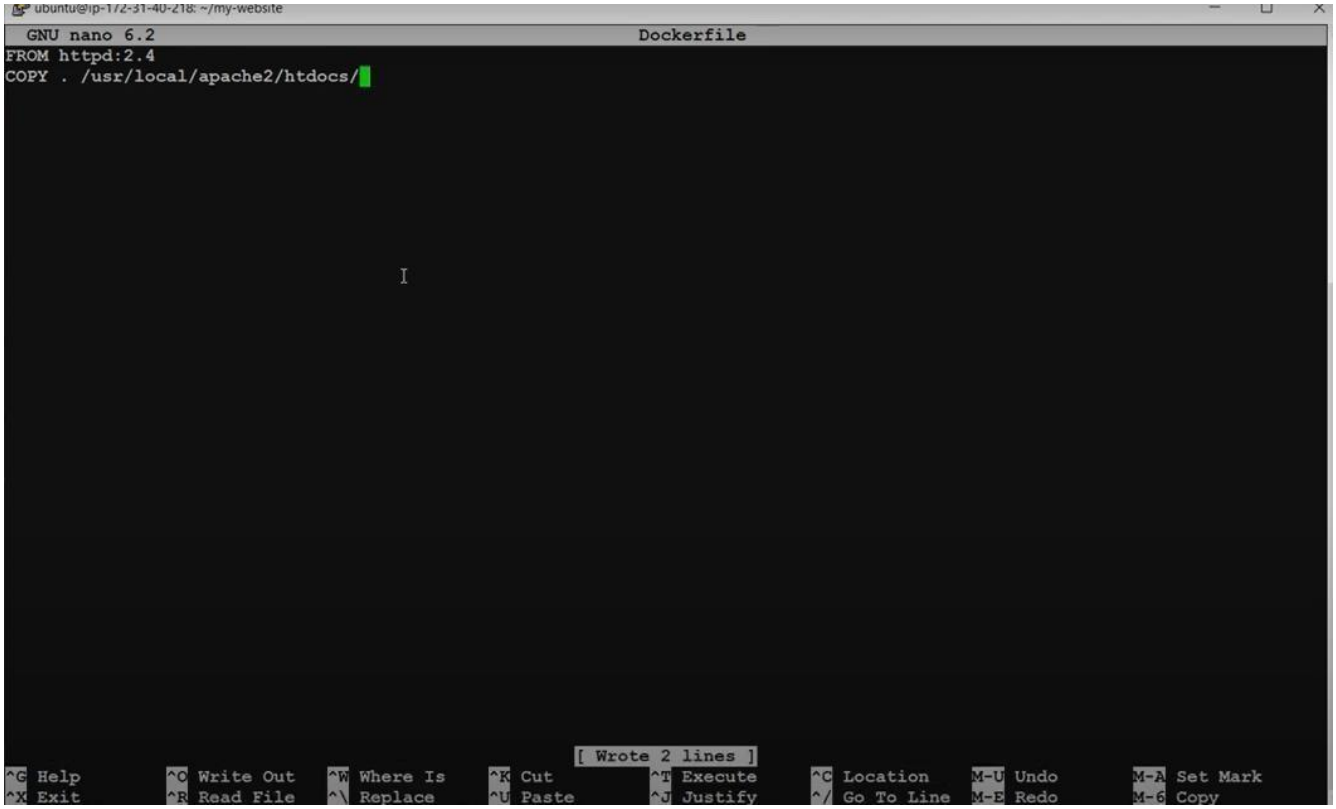
Aim: To learn Dockerfile instructions, build an image for a sample web application using DOCKERFILE.

```
ubuntu@ip-172-31-40-218:~$
ubuntu@ip-172-31-40-218:~$ pwd
/home/ubuntu
ubuntu@ip-172-31-40-218:~$ mkdir my-website
ubuntu@ip-172-31-40-218:~$ cd my-website/
ubuntu@ip-172-31-40-218:~/my-website$ wget https://www.free-css.com/assets/files/free-css-templates/download/page290/wave-c
afe.zip
--2023-04-10 20:06:14--  https://www.free-css.com/assets/files/free-css-templates/download/page290/wave-cafe.zip
Resolving www.free-css.com (www.free-css.com)... 217.160.0.242, 2001:8d8:100f:f000::28f
Connecting to www.free-css.com (www.free-css.com)|217.160.0.242|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 11896390 (11M) [application/zip]
Saving to: 'wave-cafe.zip'

wave-cafe.zip                 100%[=================================================>]  11.34M  6.08MB/s    in 1.9s

2023-04-10 20:06:17 (6.08 MB/s) - 'wave-cafe.zip' saved [11896390/11896390]

ubuntu@ip-172-31-40-218:~/my-website$
ubuntu@ip-172-31-40-218:~/my-website$
ubuntu@ip-172-31-40-218:~/my-website$ ls
wave-cafe.zip
ubuntu@ip-172-31-40-218:~/my-website$ unzip wave-cafe.zip
```

```
ubuntu@ip-172-31-40-218: ~/my-website
  inflating: 2121_wave_cafe/fontawesome/webfonts/fa-regular-400.ttf
  inflating: 2121_wave_cafe/fontawesome/webfonts/fa-regular-400.woff
  inflating: 2121_wave_cafe/fontawesome/webfonts/fa-regular-400.woff2
  inflating: 2121_wave_cafe/fontawesome/webfonts/fa-solid-900.eot
  inflating: 2121_wave_cafe/fontawesome/webfonts/fa-solid-900.svg
  inflating: 2121_wave_cafe/fontawesome/webfonts/fa-solid-900.ttf
  inflating: 2121_wave_cafe/fontawesome/webfonts/fa-solid-900.woff
  inflating: 2121_wave_cafe/fontawesome/webfonts/fa-solid-900.woff2
   creating: 2121_wave_cafe/img/
  inflating: 2121_wave_cafe/img/about-1.png
  inflating: 2121_wave_cafe/img/about-2.png
  inflating: 2121_wave_cafe/img/hot-americano.png
  inflating: 2121_wave_cafe/img/hot-cappuccino.png
  inflating: 2121_wave_cafe/img/hot-espresso.png
  inflating: 2121_wave_cafe/img/hot-latte.png
  inflating: 2121_wave_cafe/img/iced-americano.png
  inflating: 2121_wave_cafe/img/iced-cappuccino.png
  inflating: 2121_wave_cafe/img/iced-espresso.png
  inflating: 2121_wave_cafe/img/iced-latte.png
  inflating: 2121_wave_cafe/img/smoothie-1.png
  inflating: 2121_wave_cafe/img/smoothie-2.png
  inflating: 2121_wave_cafe/img/smoothie-3.png
  inflating: 2121_wave_cafe/img/smoothie-4.png
  inflating: 2121_wave_cafe/img/special-01.jpg
  inflating: 2121_wave_cafe/img/special-02.jpg
  inflating: 2121_wave_cafe/img/special-03.jpg
  inflating: 2121_wave_cafe/img/special-04.jpg
  inflating: 2121_wave_cafe/img/special-05.jpg
  inflating: 2121_wave_cafe/img/special-06.jpg
  inflating: 2121_wave_cafe/index.html
   creating: 2121_wave_cafe/js/
  inflating: 2121_wave_cafe/js/jquery-3.4.1.min.js
   creating: 2121_wave_cafe/video/
  inflating: 2121_wave_cafe/video/wave-cafe-video-bg.mp4
ubuntu@ip-172-31-40-218:~/my-website$
ubuntu@ip-172-31-40-218:~/my-website$
ubuntu@ip-172-31-40-218:~/my-website$
ubuntu@ip-172-31-40-218:~/my-website$ clear
```

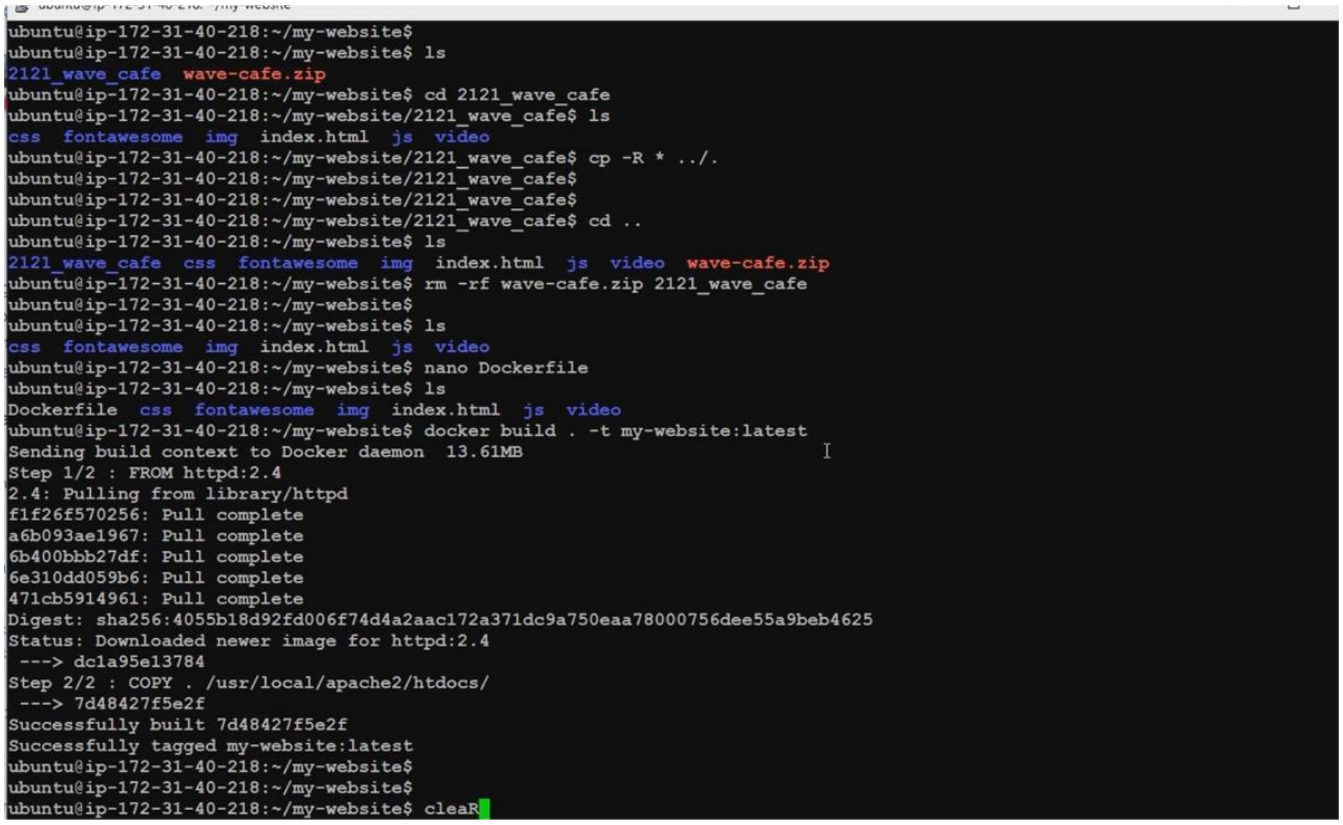
Aim: To learn Dockerfile instructions, build an image for a sample web application using DOCKERFILE.

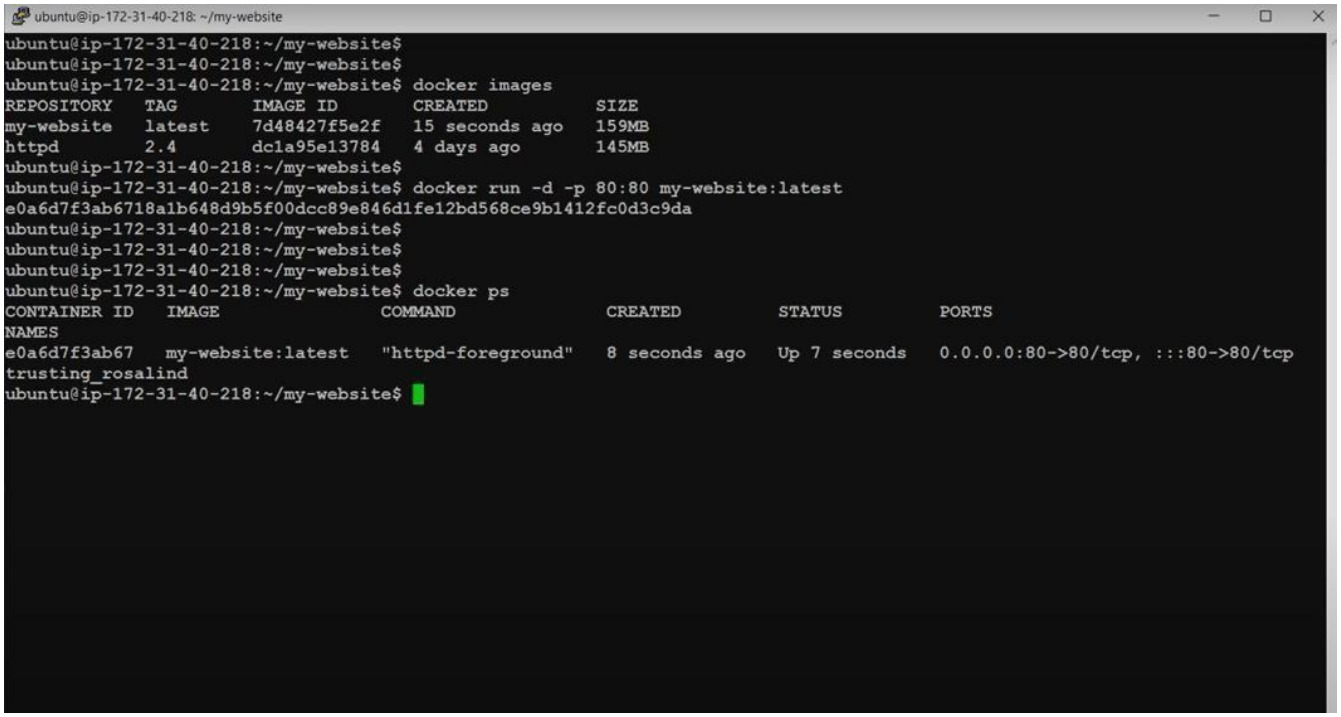




Software Engineering & Project Management Lab Experiment No: - 10

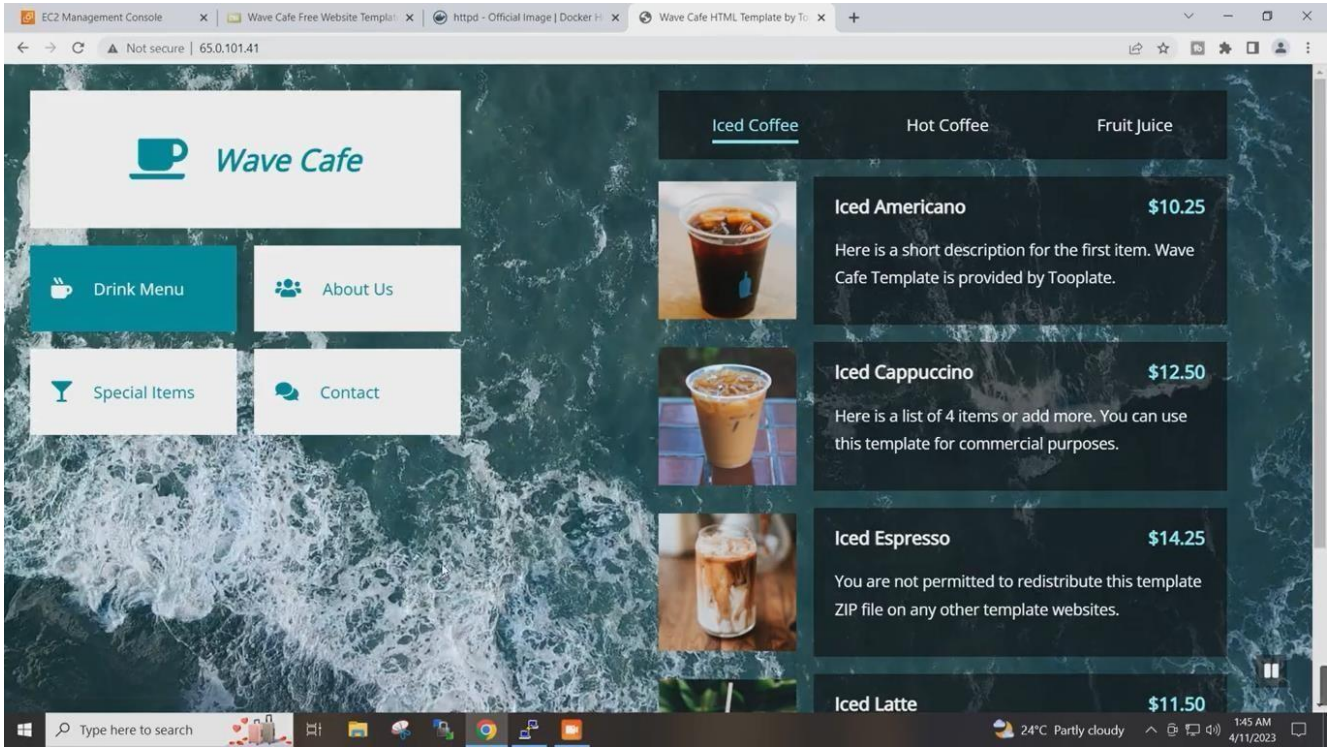
Aim: To learn Dockerfile instructions, build an image for a sample web application using DOCKERFILE.





Software Engineering & Project Management Lab Experiment No: - 10

Aim: To learn Dockerfile instructions, build an image for a sample web application using DOCKERFILE.



<u>Conclusion</u>: Thus, we have successfully learnt Dockerfile instructions & build an image for a sample web application using DOCKERFILE.

Salif Shaikh
T22 94

TSEC

SEPM ASSIGNMENT : 2

Q Differentiate between CPM and PERT

	PERT	CPM
a	PERT stands for project eval and review technique	CPM stands for critical path method
b	It is a technique of project management which is used to manage uncertain activities of any project.	It is a technique of project management used to only certain i.e (time is known) activities of any project.
c	It is a probability model	It is a deterministic model.
d	Appropriate for high precision time estimation	Appropriate for reasonable time estimation.
e	Non-repetative nature of job	Repetative nature of job.

Q. Explain the difference between Total Slack and Free slack.

Ans. Total Slack :

- It is the amount of time a task can be delayed without delaying the project overall completion date.
- It is calculated as the difference between late finish

and early finish of a task.

- If total slack is negative it means the project is behind schedule and needs compression techniques like crashing or fast tracking.

Free Slack:

- It is the amount of time a task can be delayed without delaying the start of any successor tasks.
- It is useful for identifying tasks that can be postponed without affecting dependent activities.
- If free slack is zero, any delay in the task will immediately affect atleast one successor task.

(ii) AON and AOA diagram.

→ Activity on Node (AON) diagram:
In AON diagrams, activities are represented by nodes and dependencies between them are shown with arrows

Key characteristics:-

- Node represents project activities.
- Arrows indicate dependencies between activities.
- Used in precedence diagramming method.

Activity on arrows (AOA)
In AOA, activities are represented by arrows, while nodes represent the start and end points of activities.

Key characteristics :-
- Arrows represent activities
- Nodes represent center
- Uses only finish to start relationships.

Q. Explain risk identification, risk projection, RMMM plan in detail.

Ans Risk identification is the process of of recognizing potential risks that could negatively impact a project system or organization.

Key steps:
→ Understanding project scopes
→ SWOT analysis
→ Checklist based approach
→ Historical Data analysis.

Risk projection also known as risk estimation or risk management involves analysing the identified risks in terms of their liklihood, impact and priority.

This helps in decision making regarding mitigation strategies.

Key aspects -

- Probability assessment
- Impact analysis
- Risk exposure calculation
- Risk mitigation

RMMM plan:

It is a structured approach to handling risks by reducing their probability and impact, monitoring their states and defining management strategies.

Components of RMM plan:

- Risk Mitigation
- Risk Monitoring
- Risk Management.

Q. Consider an xyz company undertake a project to computerized working of ABC city bank then
(i) Develop For the same project.
(ii) Develop responsibility matrix.

Ans(i) COBS divides the project into manageable sections ensuring a structured approach.

- Levelwise COBS for the project

THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

(a) Project initiation and planning
- Requirements analysis
- Feasibility study.
- Risk assessment and planning.
- Project scheduling and bugeting.

(b) System design and architecture
- Design a database
- Software and security architecture

(c) Software development
- Core banking system development
- Customer management module.

(d) Integration and Testing.
- System integration
- Fundamental testing.
- Security and Performance Testing.

(e) Responsibility Assignment Matrix.

Task / Activity	Project Manager	Business Analyst	Support Developer	Testers	IT Support
Requirement Analysis	R	A	C	-	-
Software development	R	I	A	-	-
System design	R	C	A	-	I
Testing Integration	R	C	A	C	I
Training & Documentation	R	A	C	-	I
Maintenance and Support	R	C	C	-	A

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Q. Explain Software Configuration Management in detail

Ans. • It is a systematic approach to managing changes in software throughout it is development lifecycle

- It ensures that software modifications are well tracked, recording and reporting the status of configuration items.

- Configuration Management is practised in form or other as part of any software engineering project where several individuals or organisations have to coordinate their activities.

- Software configuration Management is a system for managing the evaluation of software products, both during the initial stages of development and during all stages of maintenance.

- All supporting software used in development even though not part of the software product, should also be controlled by SCM.

- Advantages of SCM:

→ SCM provides significance benefits to all projects regardless of size, scope and complexity.

→ Provides a snapshot of dynamically changing software.

Q Explain the significance of Gantt charts in project Management.

Ans A gantt chart is a visual project management tool that represents the schedule of tasks over time. It helps in planning, tracking and managing tasks efficiently ensuring that project stay on schedule

Some of the significance of Gantt charts:

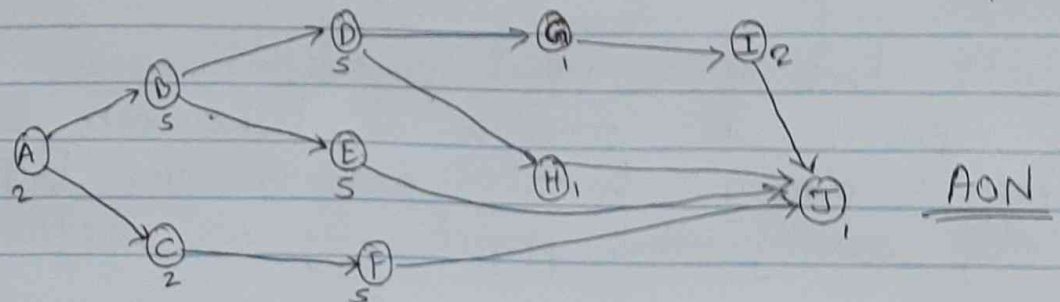
→ Visualising the project timelines:
Provides a clear picture of the projects program and structure helps stakeholders quickly understand deadline dependencies and bottlenecks

→ Task scheduling and deadlines:
Ensures that tasks are completed on time by setting clear starts and end dates. Helps manages, allocates resources effectively and avoid scheduling conflicts.

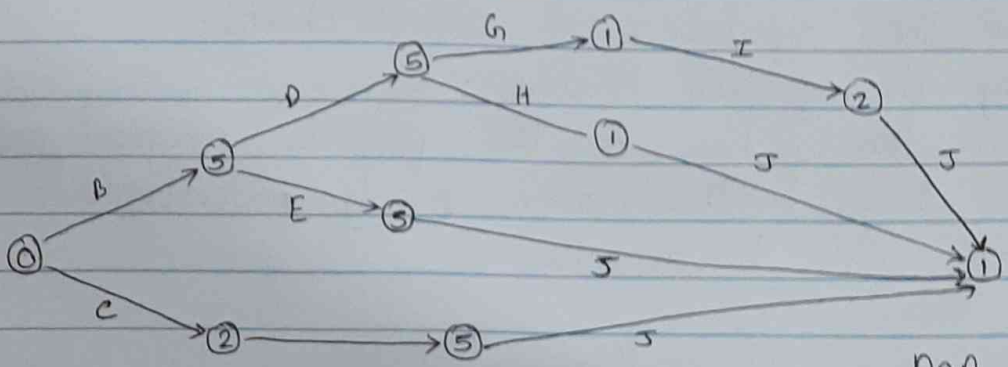
→ Managing Task dependencies.
Identifies which tasks rely on others, preventing delays in sequential tasks. Help in adjusting schedules when dependencies shift.

→ Risk identification and Mitigation
Highlights potential bottlenecks in the schedule helps in developing contingency plan for delays.

Q. Draw the AON and AOA network diagram for the following project and show critical path.

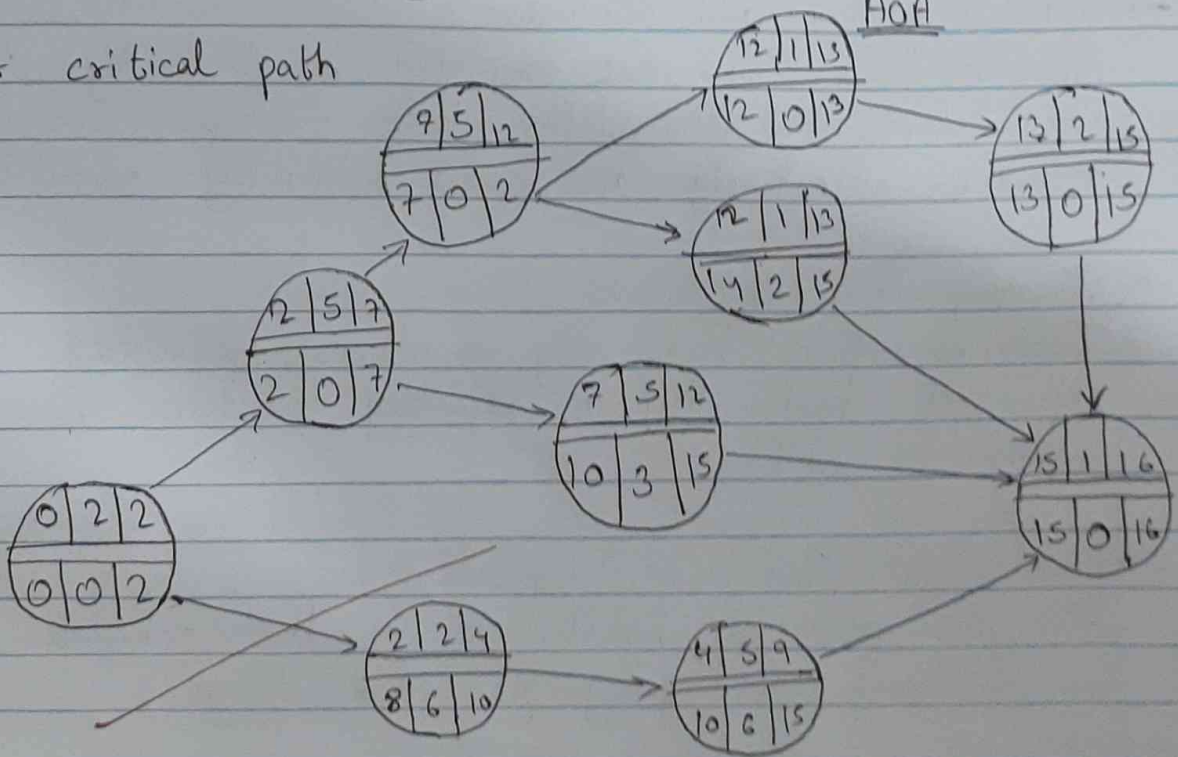


AON



AOA

For critical path



Path: A → B → D → G → I → J

SALIF
T22-99

A1

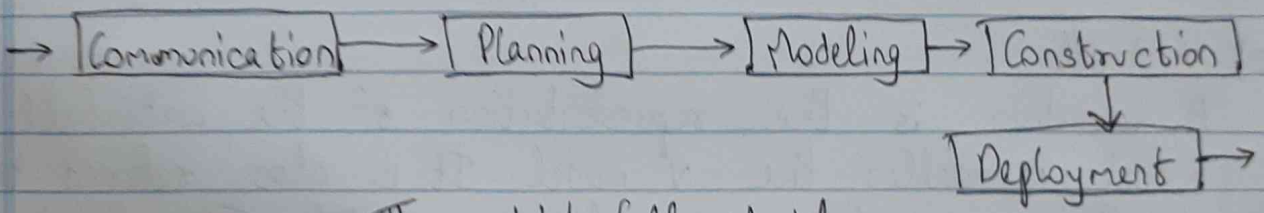
THADOMAL SHAHANI TSEC ENGINEERING COLLEGE T22-94

SEPM ASSIGNMENT - 1

1) Waterfall Model

The waterfall model, sometimes called the classic life cycle, suggests a systematic, sequential approach to software development begins with customer specifications of requirement and progress through planning, modelling, construction and deployment, eliminating in ongoing support of the completed software.

A variation is representation of the waterfall model is called v model



The Waterfall Model

Advantages:
- Simple and easy to understand
- Easy to manage.
- Best for smaller projects

Disadvantages:
- Late testing
- Lengthy development cycle.
- Not suitable for evolving projects.

eg. In a library management system, phases include requirement analysis, system design, implementation testing.

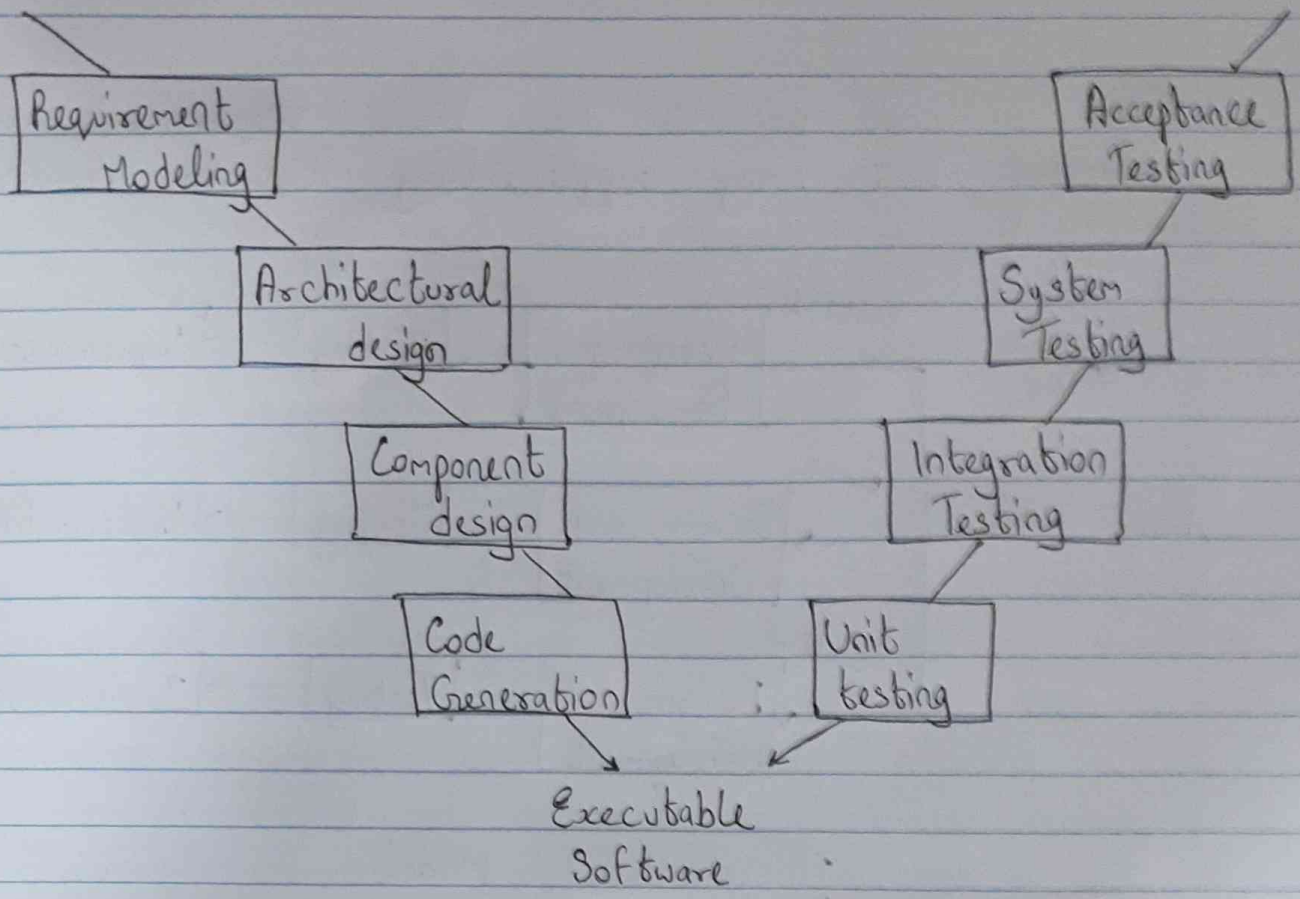
Where to use waterfall model ?
- Well understood requirement
- Very little changes expected
- Limited resources
- Small to medium size projects.

2) V Model

A variation is the representation of the waterfall model is called the V model. It is also refered to as the verification and validation model. It depicts the relationship of quality assurance actions to the actions associated with communication, modelling and early construction activities.

Where to use V Model ?
- Clear and stable requirements
- Defined testing phases.
- Strict quality assurance needs.

Advantages
- Easy to understand
- Saves a lot of time

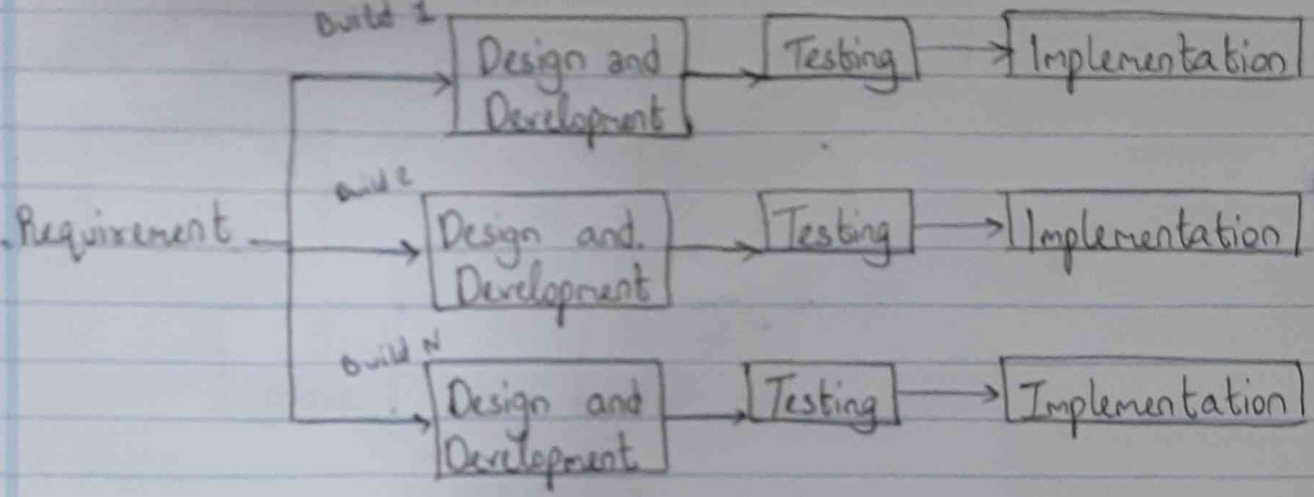


Disadvantages
- Rigid and least flexible
- Not good for complex projects

3) Incremental process Model:

The incremental model combines elements of linear and parallel process flow. It applies ~~series~~ sequence in a stagged fashion. as calender time progresses. When an incremental model is used, the first increment

after a core product i.e. basic requirement are addressed but many supplementary features remain underived.



Incremental Model

Advantages
- errors are easy to be recognised
- More flexible

Disadvantages
- Cost is high
- Need for good planning
- Well-defined module interfaces are needed.

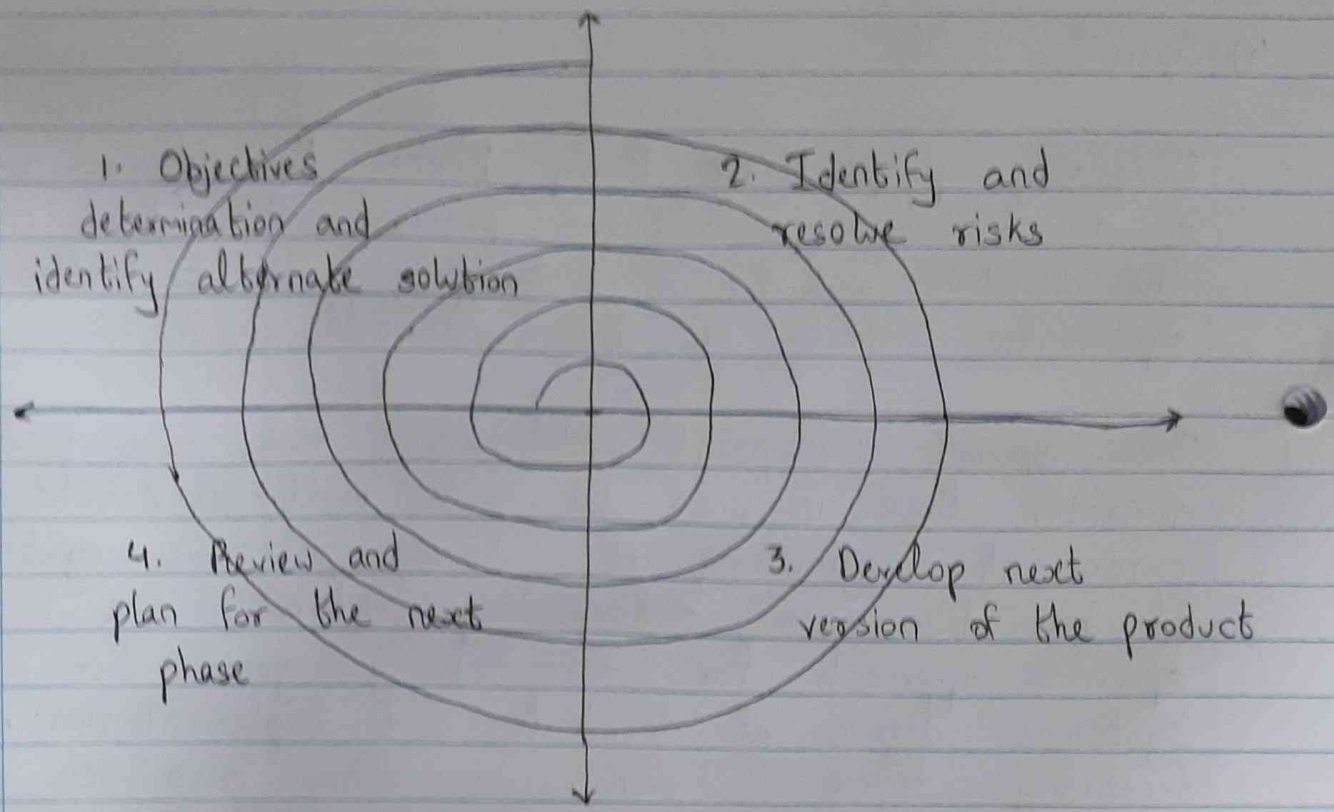
4) Special Model

Originally proposed by Barry Boehm, the special model is an evolutionary software process model that couples the iterative nature of prototyping with controlled, in systematic aspects of the waterfall model.

The spiral development model is a risk down model generator that is used to guide multi-stakeholder concurrent engineering of software intensive systems. It has two main distinguishing features; one is a specific approach for incrementally growing a system's degree of definition and implementation while decreasing degree of risk. The other is a set of anchor point milestone for ensuring stakeholder commitment to feasible and mutually satisfactory system solutions.

Advantages
- Risk handling
- Good for larger projects
- Improved quality

Disadvantages
- Complex
- Expensive
- Difficulty in time management



5) Rapid Prototyping

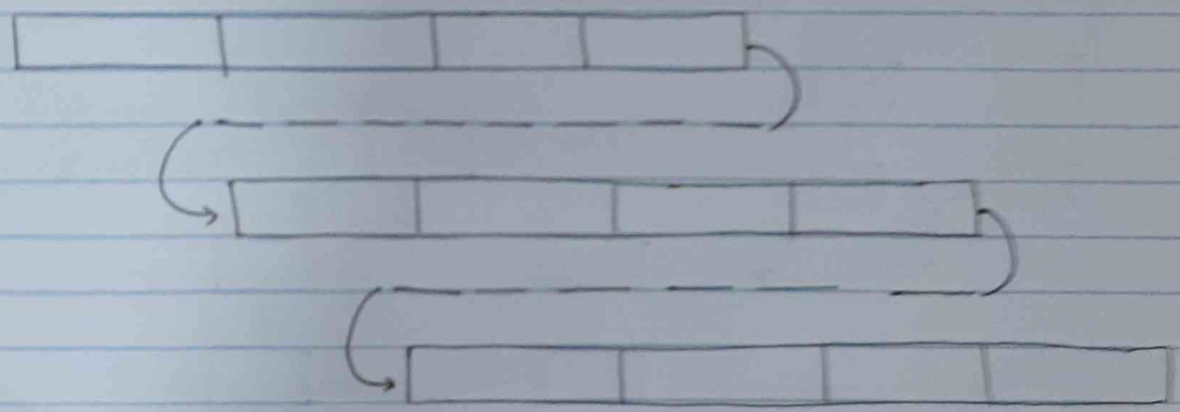
Methodology is similar to that of incremental or waterfall model the project is completed within the time and all requirement all collected before starting project. It is very fast.

Phases

- Business Modelling
- Data Modelling

Process Modelling

Advantages
- reduces time taken in development
- Components can be reused
- Flexible and easy to make changes



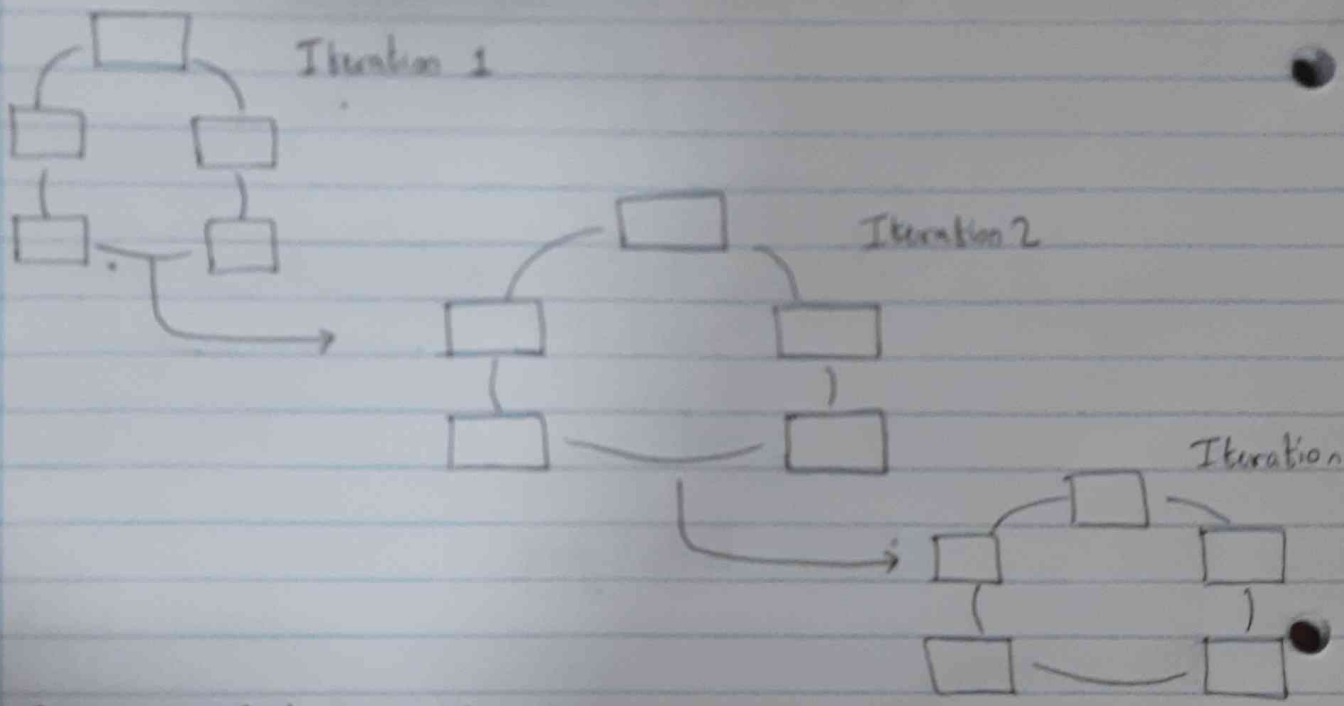
Rapid Prototyping

Disadvantages
- need highly skilled developers and designers
- very difficult to manage
- automated code generation is expensive.

6) Agile Model

It is a combination of iterative and incremental models.

Its focus is given to process adaptivity and customer satisfaction. It was created mainly to make changes in the middle of software development so that the software project can be completed quickly.



- Agile manifesto principle
  1) Individual and interactions
  2) Working software
  3) Customer collaboration

Advantages
- Project completed in very small time.
- Very realistic approach
- Provide flexibility to developers

TSEC T22-94

Disadvantages
- Cannot handle complex dependencies
- Mostly depends on customer representation.
- Due to lack of documentation, there can be confusion in development

Agile has the following models
1) Scrum
2) Extreme Programming

Model 2
XP