THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

Nirjon Soni
106
T22

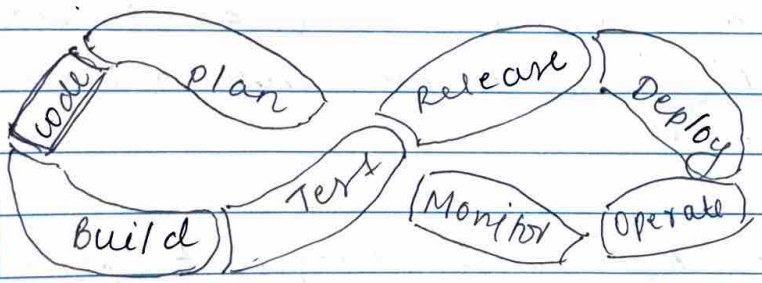
Experiment 1

Aim: To understand DevOps, principles, practices & DevOps roles & responsibilities.

Theory:

Definition:

- DevOps is the combination of 2 words, one is Development & the other is Operations. Its a culture to promote the development & operation process effectively.
- DevOps helps to increase organisation speed to deliver applications & services. It also allows organisations to serve their customers better & compete more strongly in the market.
- DevOps can also be defined as a sequence of development & IT operations with better communication & collaboration.
- DevOps has been one of the most valuable business disciplines for enterprises / organisations.



Build: without DevOps, the cost of the consumption resources was evaluated based on the pre-defined individual usage with fixed hardware allocation. And with DevOps, the usage of cloud, sharing of resources comes into the picture, & the build is dependent upon the user's need, which's a mechanism to control the usage of resources or capacity.

Code: Many good practices such as Git enables the code to be used, which ensures writing the code for business, helps to track changes, getting notified about the reason behind the difference in the actual and the expected output, and if necessary reverting to the original code developed

Test: The application will be ready for production after testing. In the case of manual testing, it consumes more time in testing and moving the code to the output. The testing can be automated, which decreases the time for testing so that the time to deploy the code to production can be reduced as automating the running of the scripts will remove many manual steps.

Plan: DevOps use Agile methodology to plan the development. With the operations & the development team in sync, it helps in organising the work to plan accordingly to increase productivity.

5. Monitor : Continuous monitoring is used to identify any risk & failure. Also, it helps in tracking the system accurately so that the health of the application can be checked. The monitoring becomes more comfortable with services where the log data may get monitored through many 3rd party tools such as Splunk.

6. Deploy : Many systems can support the scheduler for automated deployment. The cloud management platform enables users to capture accurate insights & view the optimization scenario, analytics on trends by the deployment of dashboards.

7. Operate : DevOps changes the traditional approach of developing & testing separately. The teams operate in a collaborative way where both the teams actively participate throughout the service lifecycle.

8. Release : Deployment to an environment can be done by automation. But when the deployment is made to the production environment, it is done by manual triggering.

Principles:
- collaboration
- data-based decision making
- customer-centric decision making
- constant improvement

- responsibility throughout the lifecycle.
- automation
- failure as a learning opportunity.

Advantages:

- DevOps is an excellent approach for quick development & deployment of applications.
- It responds faster to the market changes to improve business growth
- DevOps escalate business profit by decreasing software delivery time & transportation costs.
- DevOps clears the descriptive process, which gives clarity on product development & delivery.
- It improves customer experience & satisfaction.

Disadvantages:

- DevOps professional/experts developers are less available
- Developing with DevOps is so expensive.
- Adopting new DevOps tech into the industries is hard to manage in a short time.

Conclusion: Hence, we have known what DevOps is & its adv & disadv.

Experiment 02

Aim: To study and implement making a GIT account.

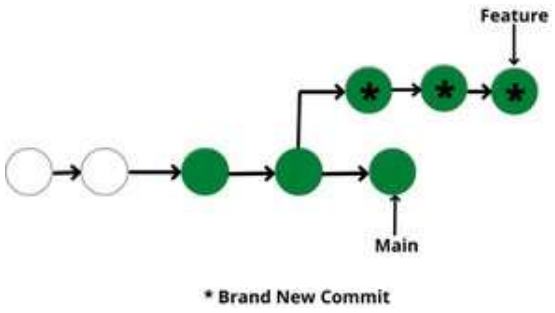
Theory:

Version control allows you to keep track of your work and helps you to easily explore the changes you have made, be it data, coding scripts, notes, etc. Version control systems are also called as revision control systems. Revision control systems work as independent standalone applications.

Applications like spreadsheets and word processors have control mechanisms. The unique features of version control system/ revision control system are as follows: Up to date history is available for the document and file types. It does not require any other repository systems. The repositories can be cloned as per the need and availability. This is extremely helpful in case of failure and accidental deletions. VCS includes tag system which helps in differentiating between alpha, beta or various release versions for different documents.

Use of Version Control System:

1. A repository: It can be thought of as a database of changes. It contains all the edits and historical versions (snapshots) of the project.
2. Copy of Work (sometimes called as checkout): It is the personal copy of all the files in a project. You can edit to this copy, without affecting the work of others and you can finally commit your changes to a repository when you are done making your changes.
3. Working in a group: Consider yourself working in a company where you are asked to work on some live project. You can't change the main code as it is in production, and any change may cause inconvenience to the user, also you are working in a team so you need to collaborate with your team to and adapt their changes. Version control helps you with the, merging different requests to main repository without making any undesirable changes. You may test the functionalities without putting it live, and you don't need to download and set up each time, just pull the changes and do the changes, test it and merge it back. It may be visualized as.



The various types of the version control systems are:

1. Local Version Control System

2. Centralized Version Control System

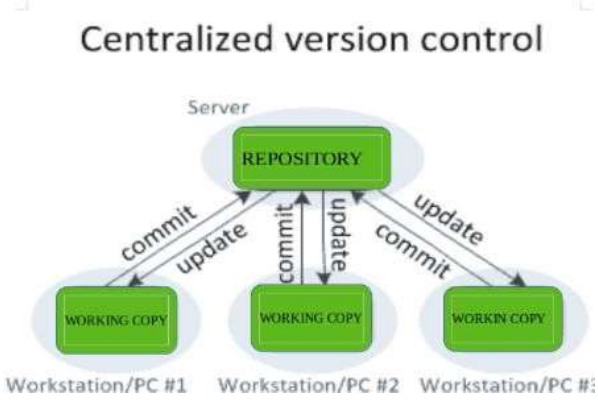
3. Distributed Version Control System

1. Local version control system: Local version control system maintains track of files within the local system. This approach is very common and simple. This type is also error prone which means the chances of accidentally writing to the wrong file is higher

2. Centralized Version Control System: In this approach, all the changes in the files are tracked under the centralized server. The centralized server includes all the information of versioned files, and list of clients that check out files from that central place.

Two things are required to make your changes visible to others which are:

- You commit
- They update



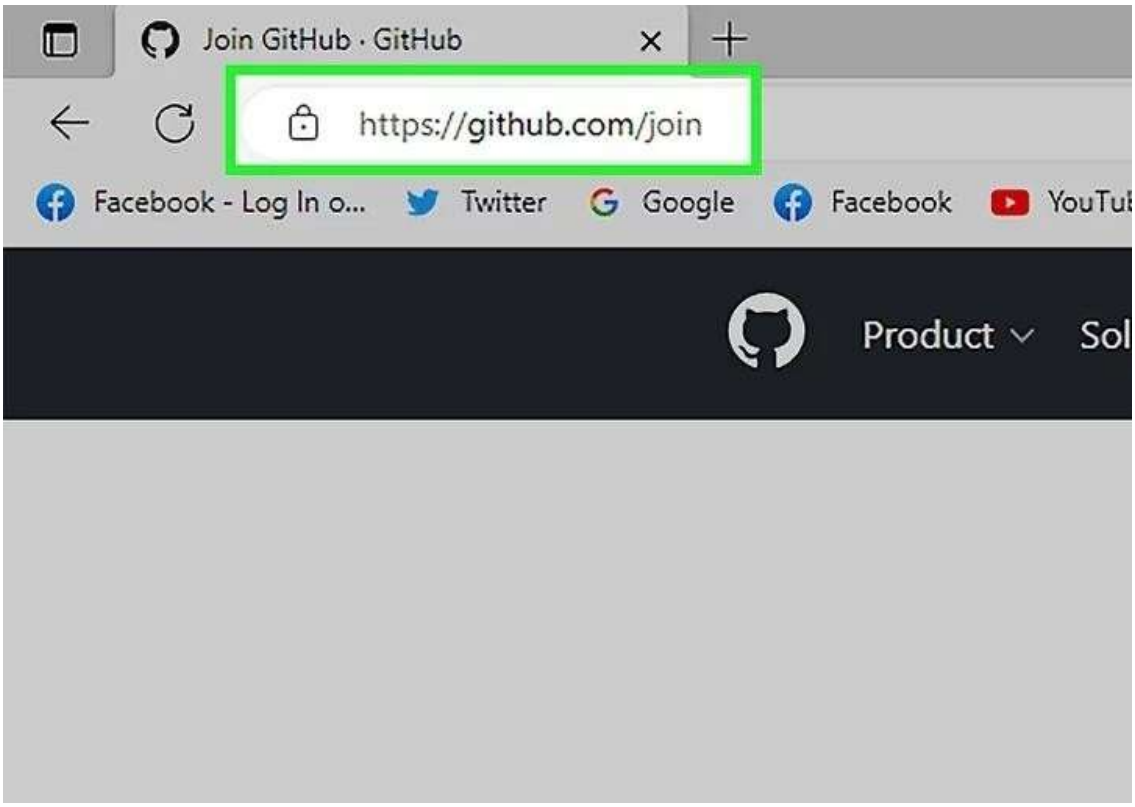
Signing up for a new personal account

1. Navigate to https://github.com/.
2. Click Sign up.
3. Follow the prompts to create your personal account.

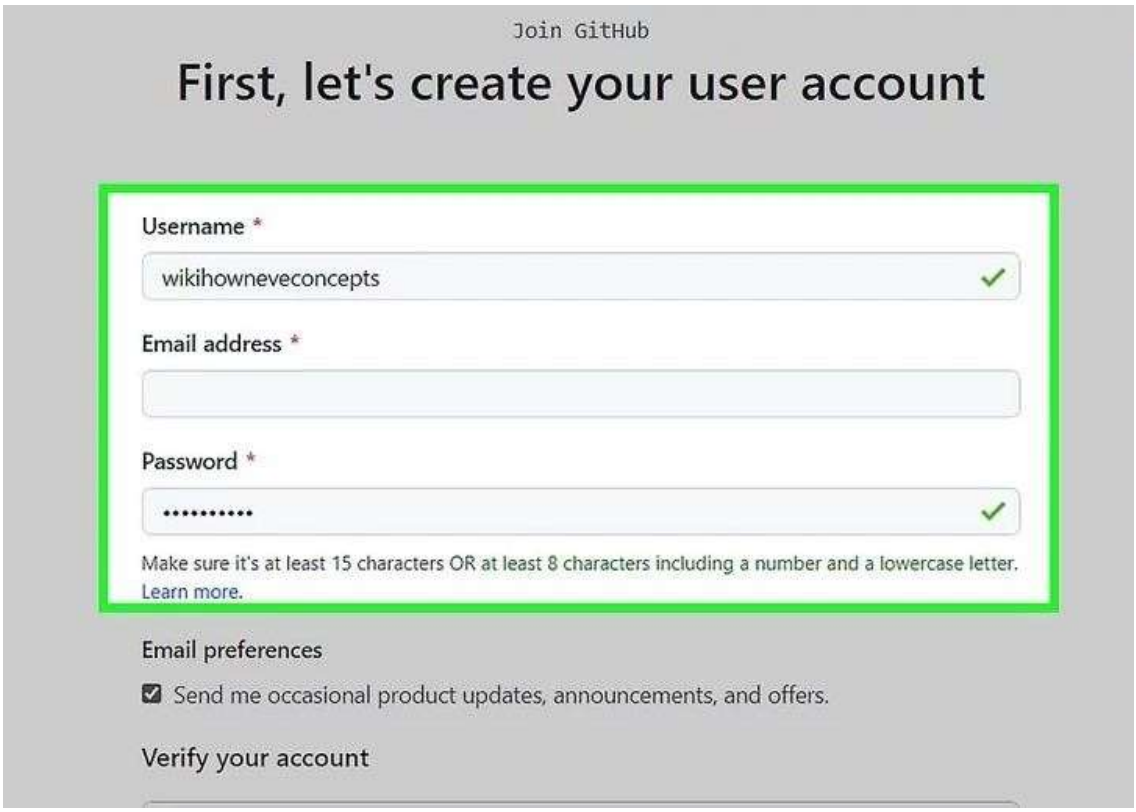
During sign up, you'll be asked to verify your email address. Without a verified email address, you won't be able to complete some basic GitHub tasks, such as creating a repository.

Some enterprises create managed user accounts for their users. You can't sign up for a personal account with an email address that's already verified for a managed user account.

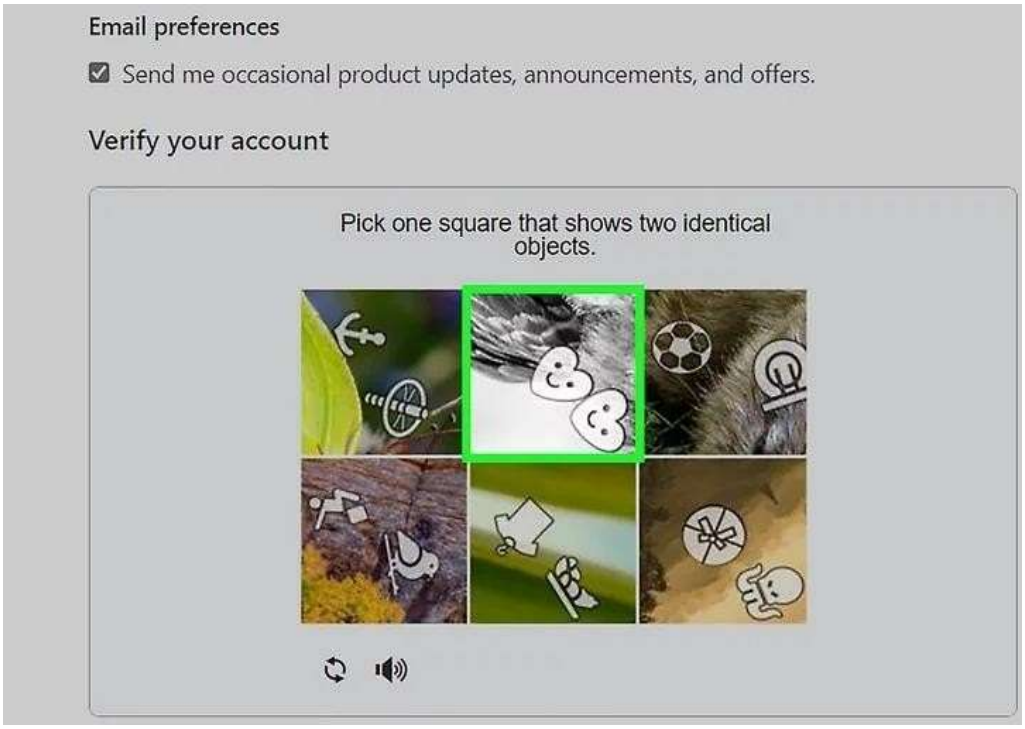
Step 01: Go to https://github.com/join in a web browser. You can use any web browser on your computer, phone, or tablet to join. Before you can create branches or make any pull requests, you'll need an account.



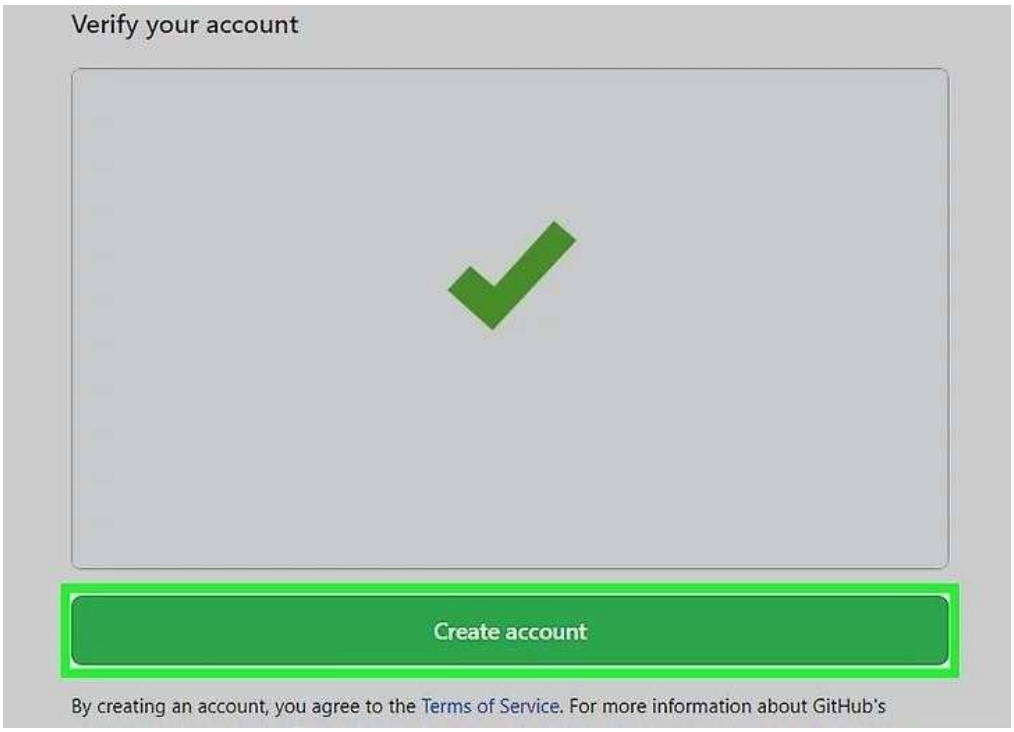
Step 02: Enter your personal details. In addition to creating a username and entering an email address, you'll also have to create a password.



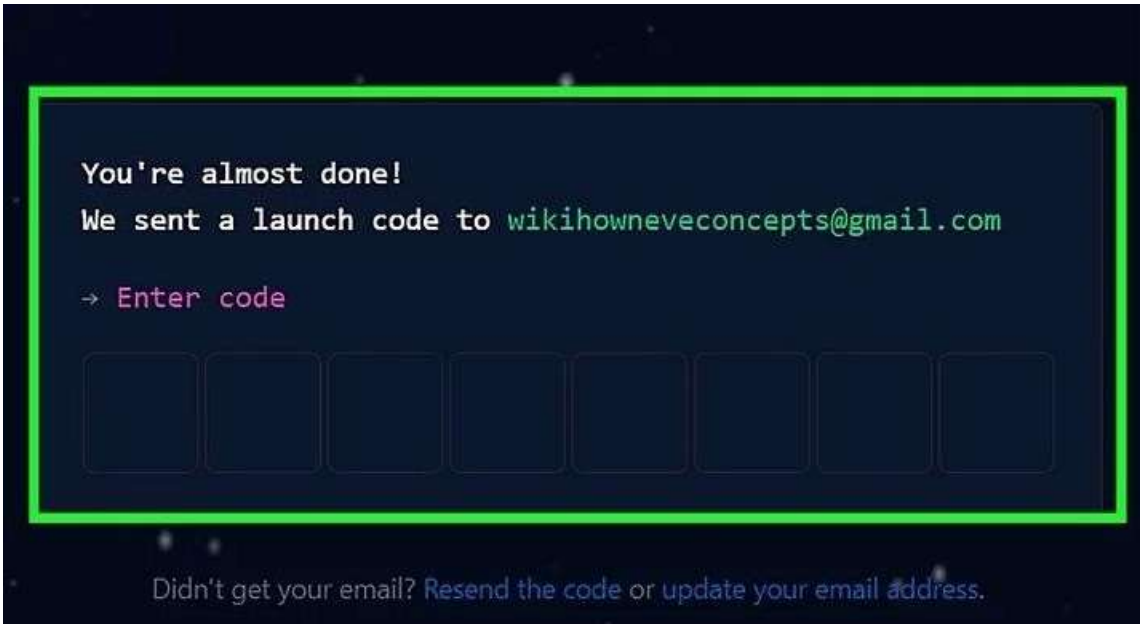
Step 03: Click Verify to start the verification puzzle. The instructions vary by puzzle, so just follow the on-screen instructions to confirm that you are a human. A green checkmark will appear after completing the puzzle.

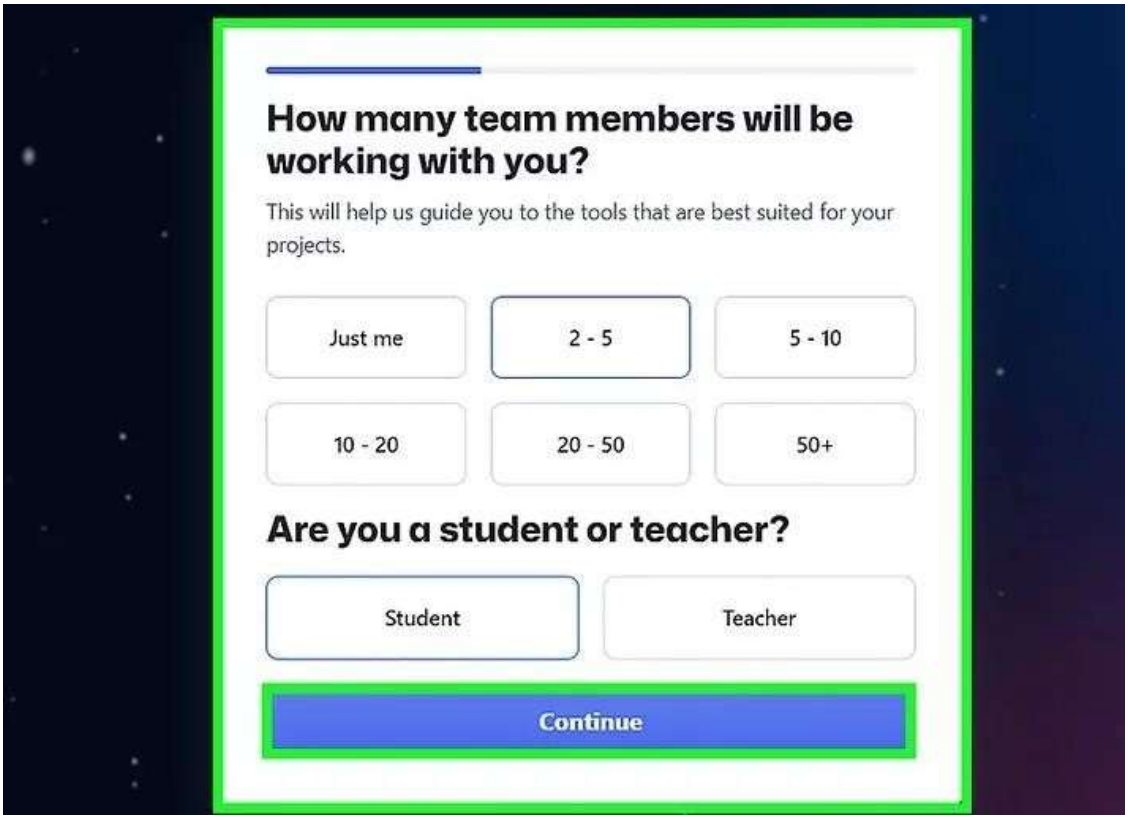


Step 04: Click the green Create account button. It's below the form, at the bottom of the page. This will take you to an email verification page.

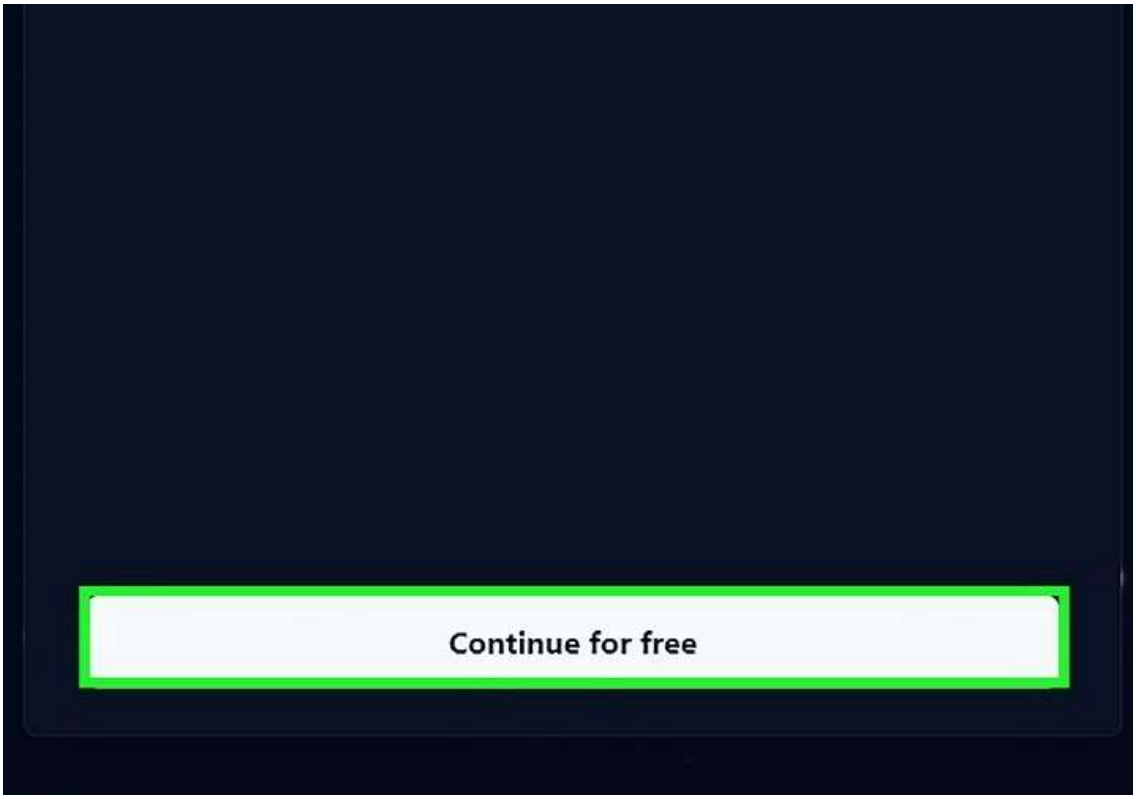


Step 05: Verify your email by entering the code. After clicking Create account, you'll receive an email with a code.





Step 06: Select the free plan. On the plan selection page, scroll down to click the button for choosing a free plan. This will immediately take you to your GitHub dashboard



Conclusion: Hence we studied and implemented creating a Github account.

TSEC

NIRJARA SONI
T22-106

Experiment 03

Aim: To Perform various GIT operations on local and Remote repositories

Theory:

The command mkdir git creates a new directory (folder) named "git" in the current working directory. This command is used to make a new directory in a Unix-like operating system.

The command cd git is used to change the current working directory to the directory named "git." After executing this command, any subsequent commands or file operations will occur within the "git" directory. "cd" stands for "change directory."

The git config --global user.name and git config --global user.email commands are used to set your global Git username and email address, respectively. They are part of the configuration settings in Git and are associated with the commits you make.

If you want to check your configuration settings, you can use the git config --list command to list all the settings Git can find at that point git commit -am "commit message"` stages and commits all changes in tracked files with a commit message in a single command.

The command nano index.html opens the Nano text editor for the file named "index.html." Nano is a simple command-line text editor that allows you to view and edit files directly in the terminal.

The command touch teststatus creates an empty file named "teststatus" in the current directory. The touch command is commonly used to update the timestamps of a file or create an empty file if it doesn't exist.

git checkout -- teststatus: Discards changes to the file "teststatus" in the working directory.This reverts the file to the state it has in the last commit.

The git add command is used to stage changes in the working directory for the next commit in Git. It prepares modifications, additions, or deletions to be included in the upcoming commit.

The git log command is used to display the commit history of a Git repository. It shows a chronological list of commits, including commit hashes, author information, timestamps, and commit messages.

The command git log --oneline displays a simplified and concise one-line representation of the commit history in a Git repository, showing only the commit SHA-1 hash and the commit message.

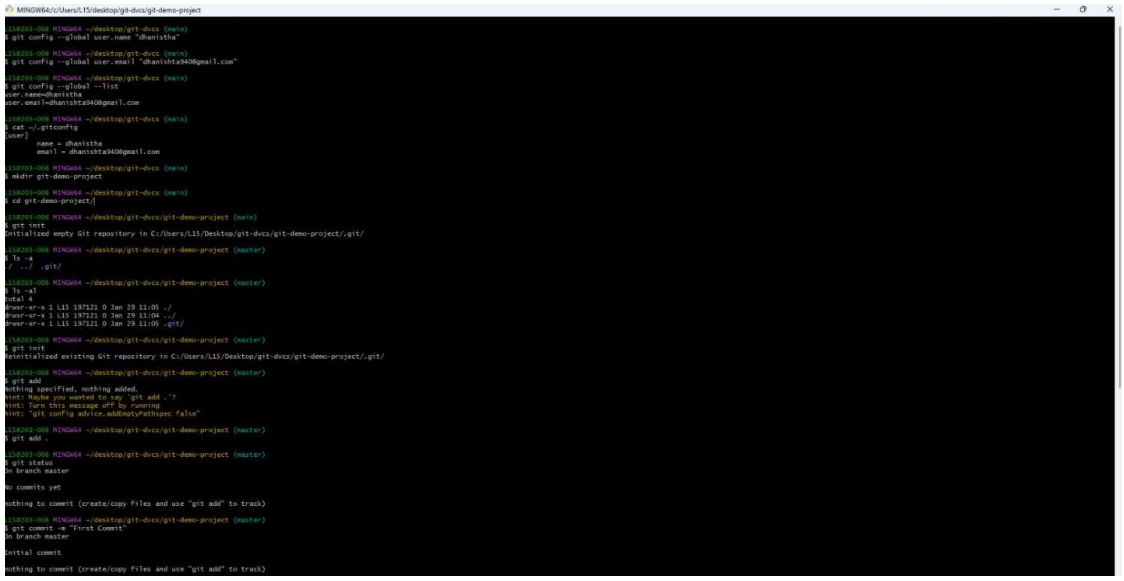
The git clone command is used to create a copy of a Git repository. When you run this command, it duplicates the entire repository, including its files, commit history, and branches, and downloads it to your local machine. This is often the initial step when you want to work with a project hosted on a remote Git repository.

The git pull command is used to fetch and integrate changes from a remote repository into the current branch of your local repository. It combines two actions: it fetches the changes from the remote repository, and then it automatically merges those changes into your local branch. This is a convenient way to update your local repository with the latest changes from the remote repository.

The git push command is used to upload or push the local changes in your Git repository to a remote repository. It updates the remote repository with the latest changes made in your local branch, making them accessible to others who share the same remote repository.

The git fetch command is used to retrieve changes from a remote repository. It fetches any new branches or changes made in the remote repository since your last interaction. However, it does not automatically merge these changes into your local branches. After using git fetch, you can inspect the changes and decide whether to integrate them using git merge or git rebase.

Output:



Conclusion:

Thus, we have successfully studied and performed various GIT operations on local and Remote repositories.

Experiment 04

Aim: To understand continuous integration, install and configure Jenkins with maven/ant/gradle to set up a build job.

Theory:

To install Jenkins following software packages are required:

1) GIT (git-scm.com)

2) Notepad++ (https://notepad-plus-plus.org/downloads/)

3) Latest Java development kit (JDK)

4) Jenkins

5) Apache Maven (Optional)

Step 1-: Install GIT

Step 2 -: Install Notepad++

Step 3 -: Install Java

Step 4 -: Install Jenkins

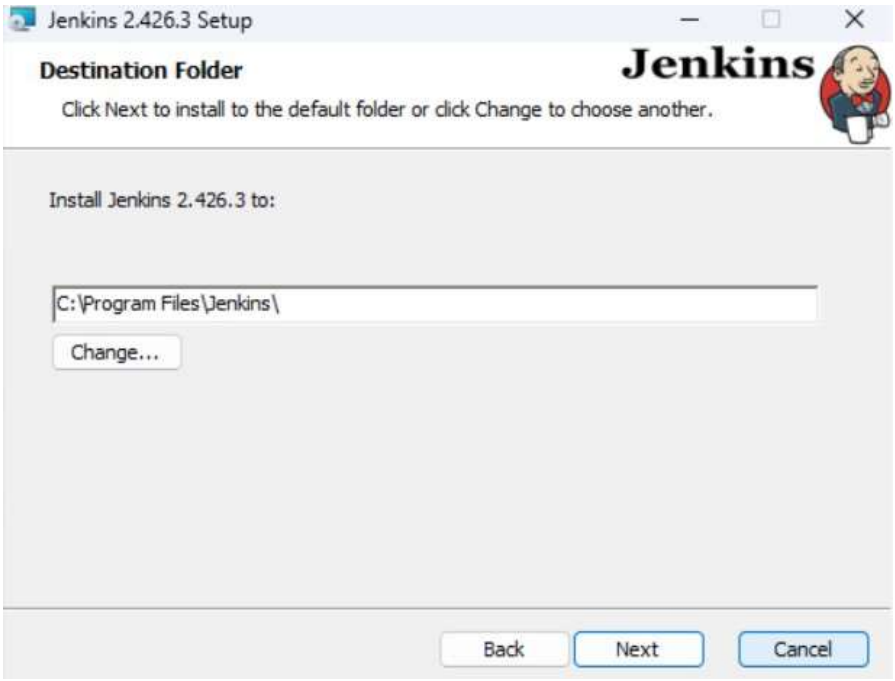
Step 5 -: Install Maven

Jenkins is an open source automation tool written in Java with plugins built for Continuous Integration purpose. Jenkins is used to build and test your software projects continuously making it easier for developers to integrate changes to the project, and making it easier for users to obtain a fresh build. It also allows you to continuously deliver your software by integrating with a large number of testing and deployment technologies.

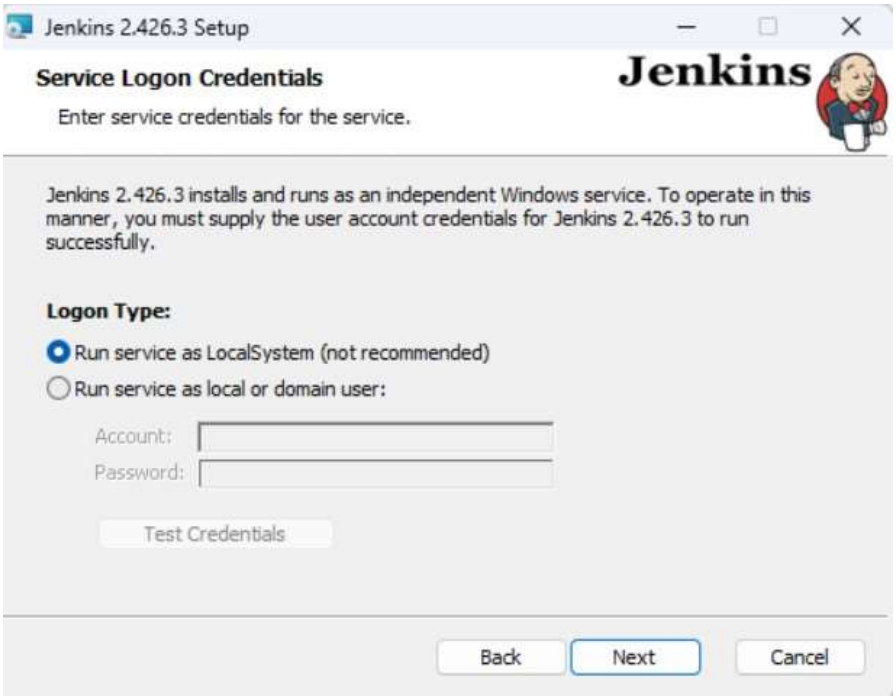
Step 1-: Open https://www.jenkins.io/doc/book/installing/windows/ and install Jenkins.

Open the installed .exe setup

Step 2: Locate the folder where you want to install Jenkins in the location path:

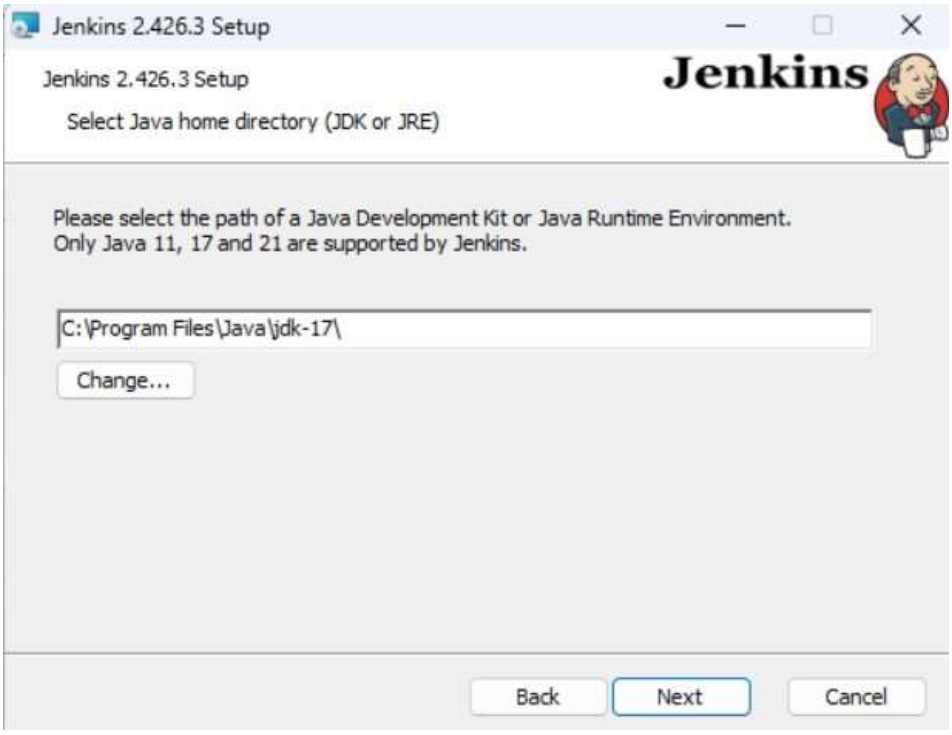


Step 3: Select service as Local System and proceed to Next.

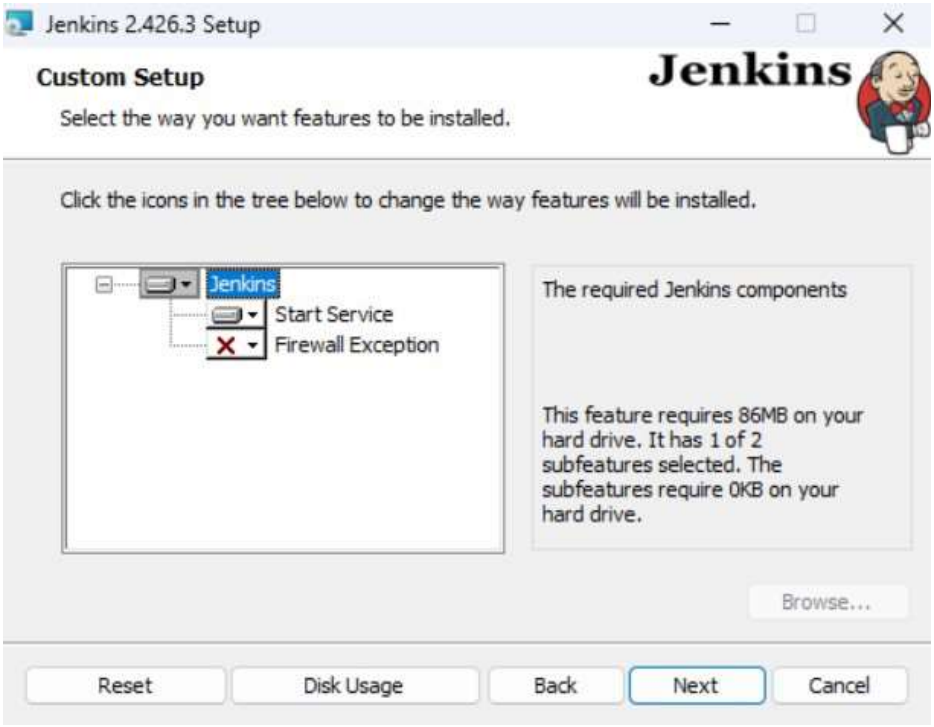


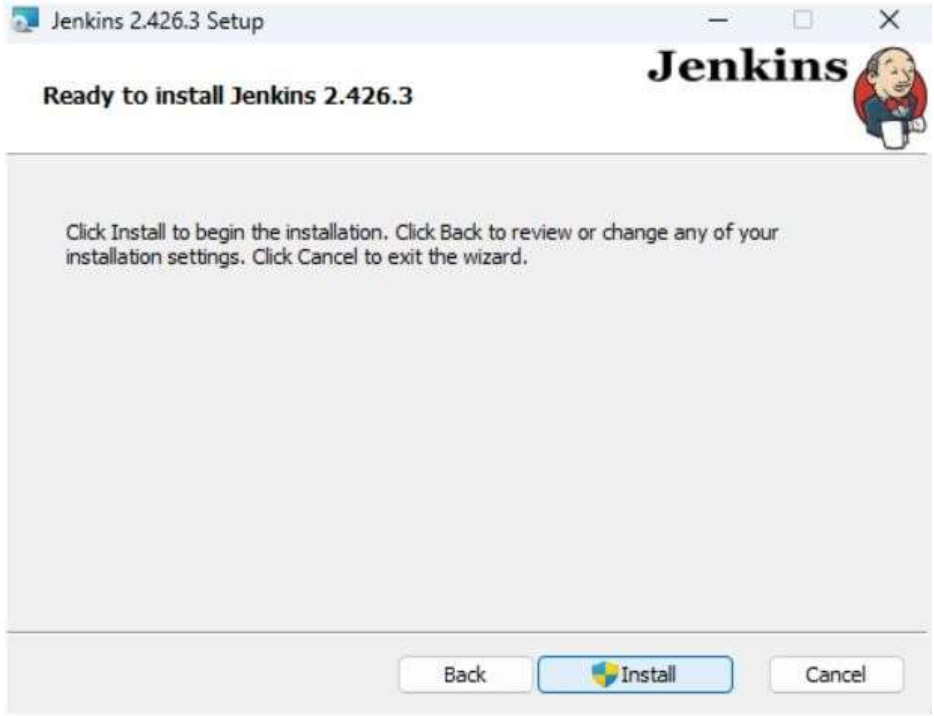
Step 4: Select the port 8080 and click Test Port button. The green tick will appear after which you can proceed to Next.

Step 5: Locate the folder where you have installed JDK in the location path:



Proceed to Next

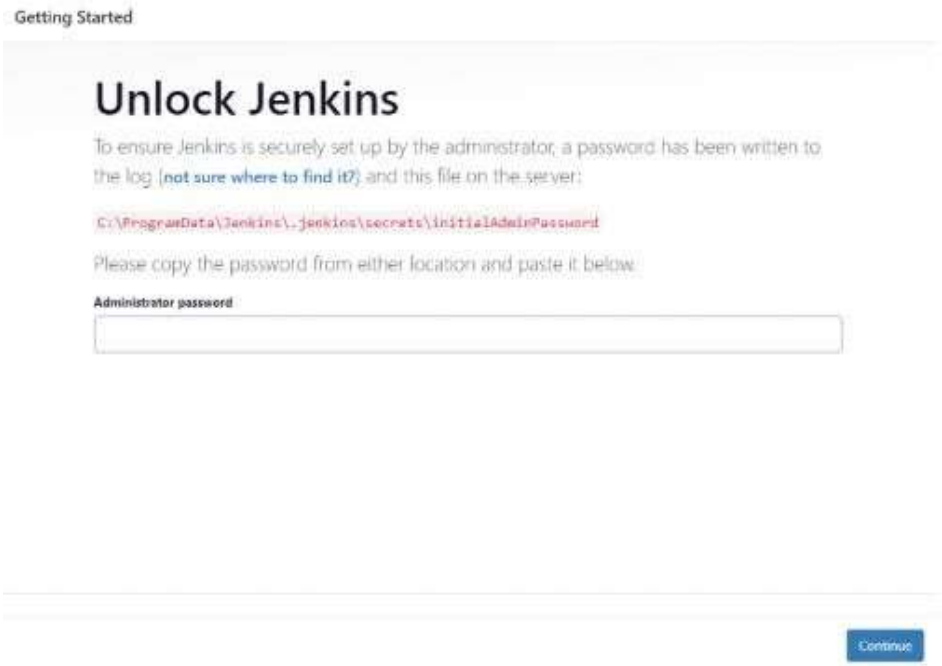




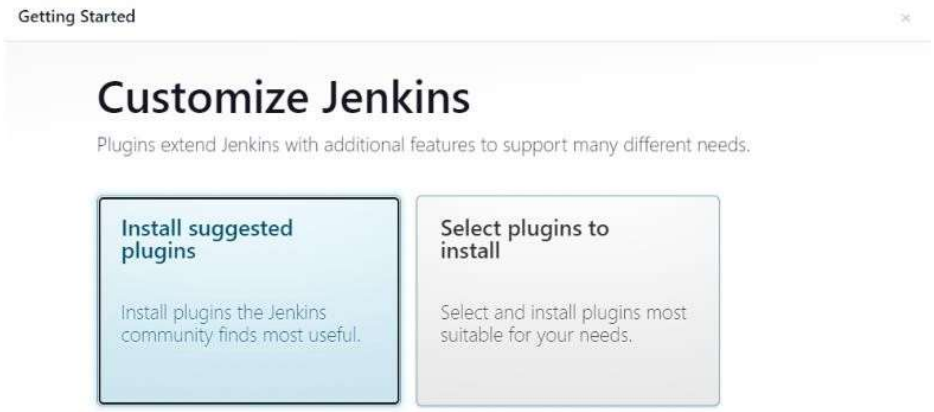
On clicking ‘Install’, installation is finished.

Step 6: Once Installation is done, you can test the Jenkins on http://localhost:8080 on the browser.

First time, when you open Jenkins portal it will ask to put admin default password which is stored in /var/lib/jenkins/secrets/initialAdminPassword file.

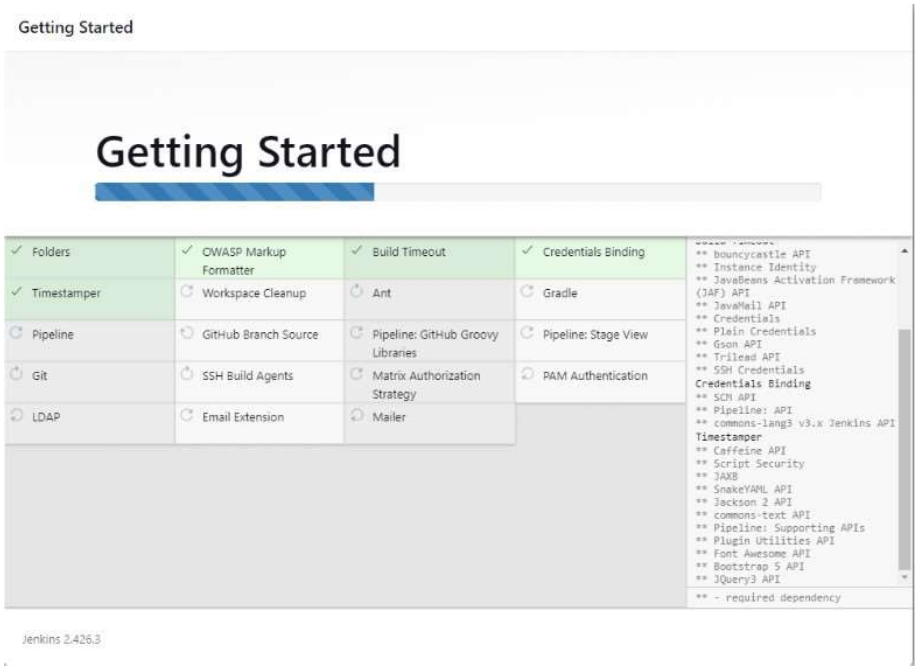


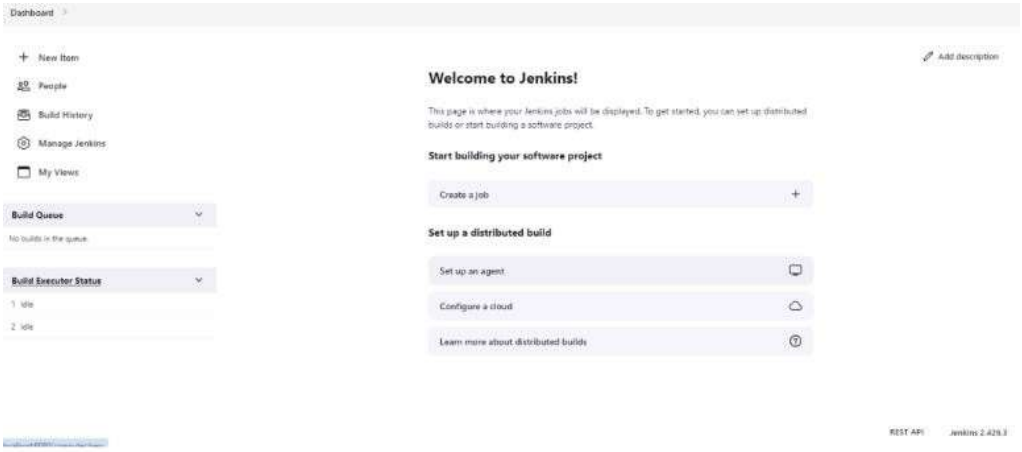
Step 7: On entering the password, you can continue to choose “Install Suggested Plugins”

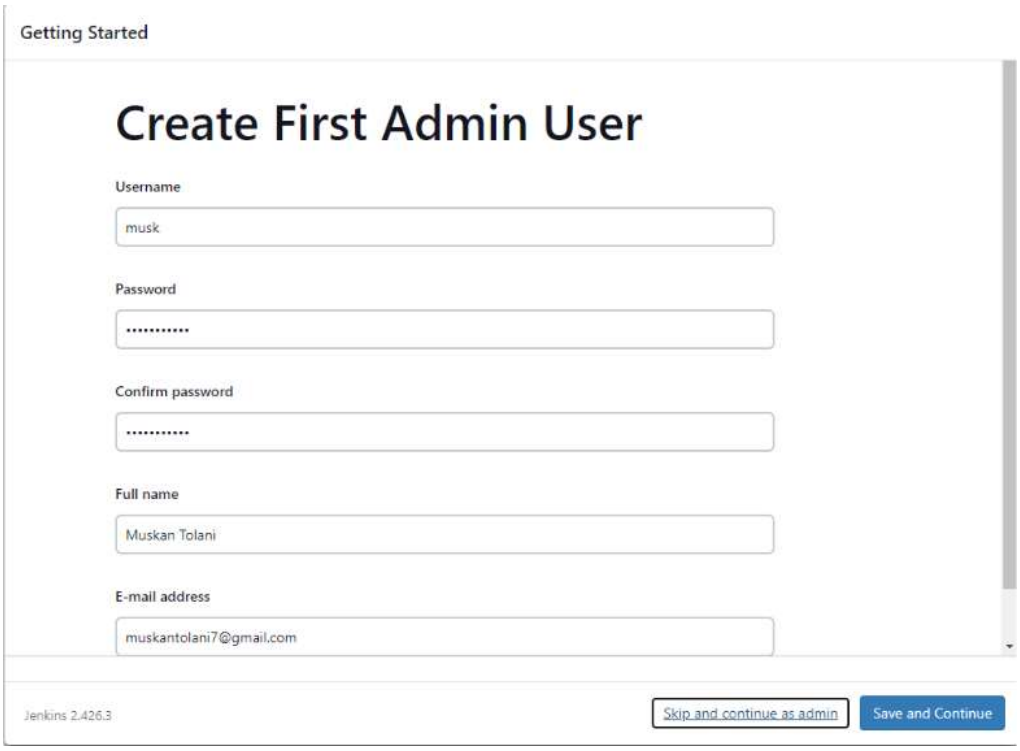


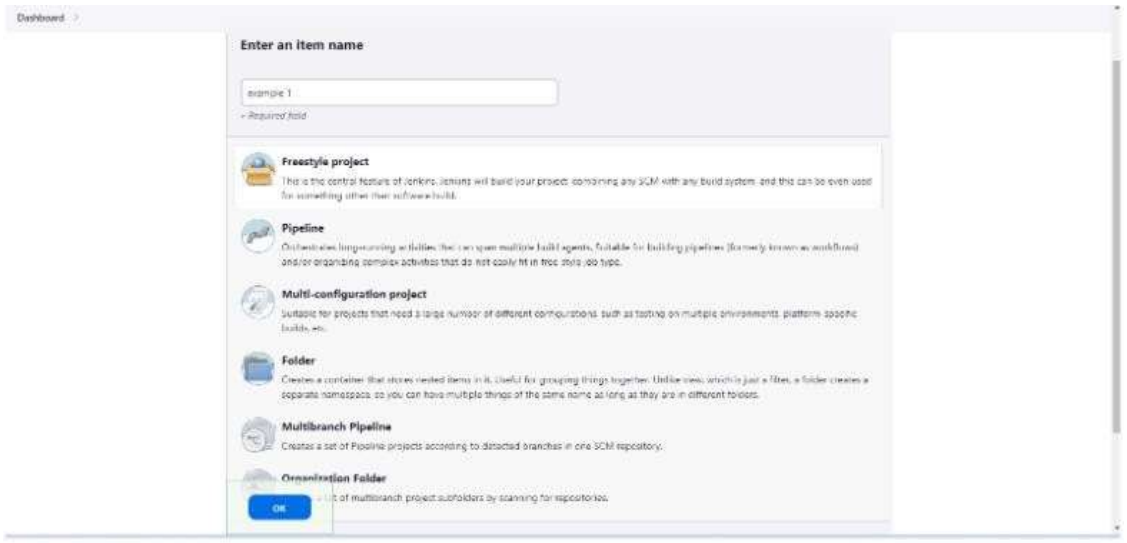
Once plugins are installed, click on next and specify the admin details along with the new password for Jenkins admin and click on finish to complete the installation.

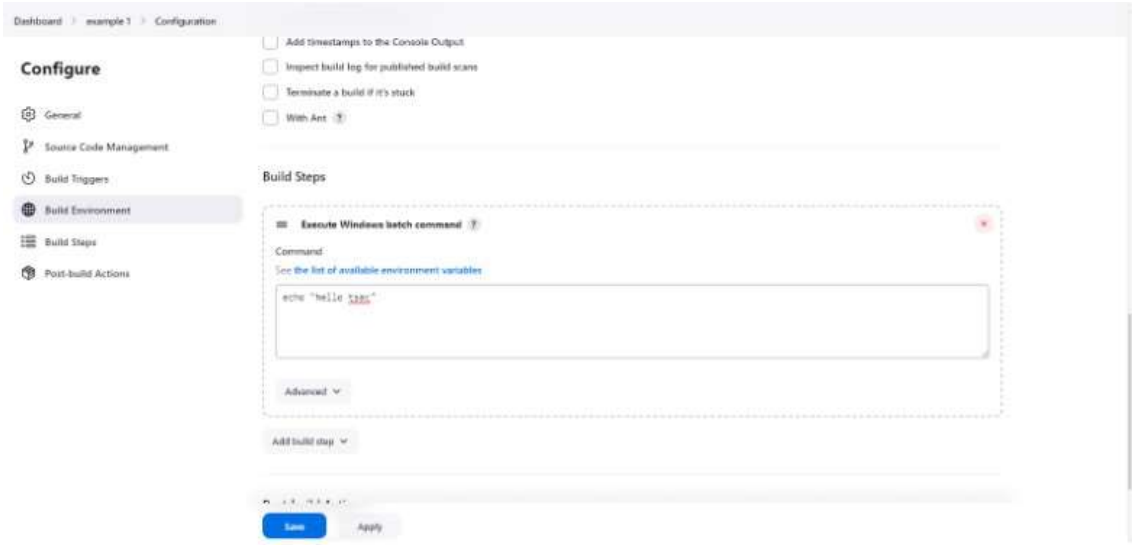
After filling the details, click on Save & Continue, you will be redirected to the dashboard.




Dashboard
New Item
People
Build History
Manage Jenkins
My Views
Build Queue
No builds in the queue.
Build Executer Status
1 Idle
2 Idle
Add description
Welcome to Jenkins!
This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.
Start building your software project
Create a job
Set up a distributed build
Set up an agent
Configure a cloud
Learn more about distributed builds
REST API
Jenkins 2.426.3

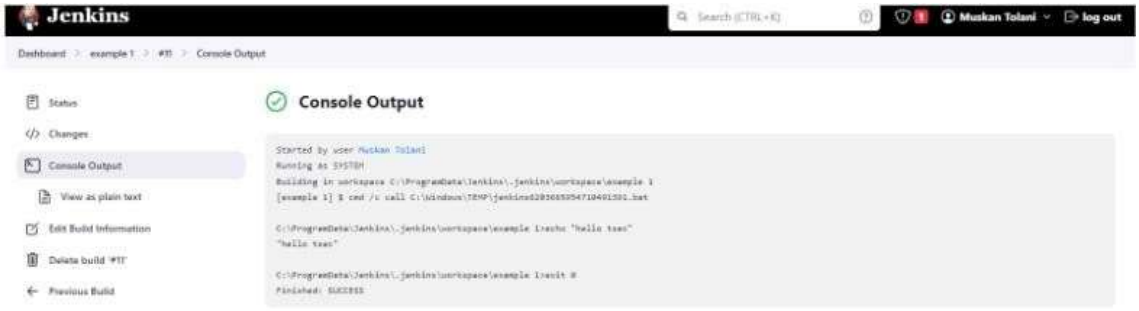

Getting Started
Create First Admin User
Username
musk
Password
Confirm password
Full name
Muskan Tolani
E-mail address
muskantolani7@gmail.com
Jenkins 2.426.3
Skip and continue as admin
Save and Continue


Dashboard
Enter an item name
example 1
» Required field
Freestyle project
Pipeline
Multi-configuration project
Folder
Multibranch Pipeline
Organization Folder
OK


Dashboard › example 1 › Configuration
Configure
General
Source Code Management
Build Triggers
Build Environment
Build Steps
Post-build Actions
Add timestamps to the Console Output
Inspect build log for published build scans
Terminate a build if it's stuck
With Ant
Build Steps
Execute Windows batch command
Command
See the list of available environment variables
echo "hello tsec"
Advanced
Add build step
Save
Apply


Jenkins
Search (CTRL+K)
Muskan Tolani
log out


Jenkins
Search (CTRL+K)
Muskan Tolani
log out
Dashboard › example 1 › #11
Status
Changes
Console Output
Edit Build Information
Delete build '#11'
Previous Build
#11 (Jan 31, 2024, 9:07:13 AM)
Keep this build forever
Add description
No changes.
Started by user Muskan Tolani



Conclusion: We have successfully installed and configured Jenkins with Maven/Ant/Gradle to setup a build Job and learnt about the implementation of Jenkins in open source continuous integration.

NIRJARA SONI
T22-106
TSEC

Experiment 05

Aim: Experiment 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

Theory:
Programming in Jenkins:

Continuous Integration is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible." In simple way, Continuous integration (CI) is the practice of frequently building and testing each change done to your code automatically.

Jenkins is a self-contained, open-source automation server which can be used to automate all sorts of tasks related to building, testing, and delivering or deploying software.

Our first job will execute the shell commands. The freestyle project provides enough options and features to build the complex jobs that you will need in your projects.

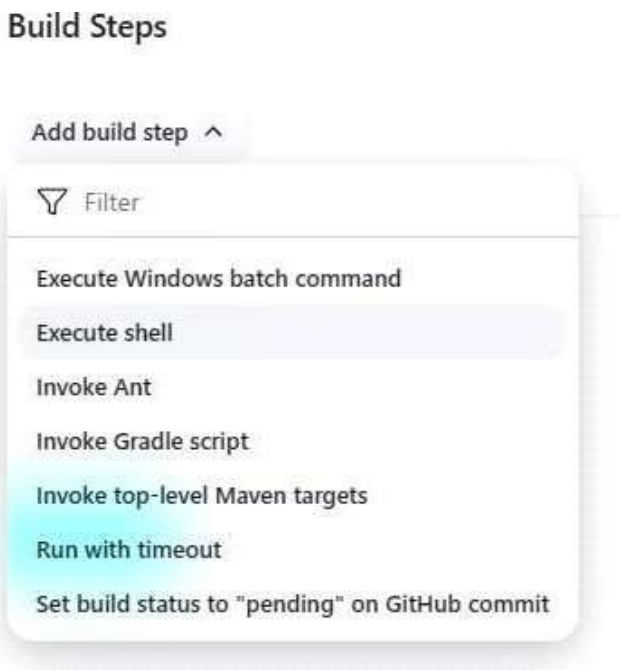
Example 1

Example 1.1: Deploying a freestyle app

in Jenkins Creating a job:

Naming the job and setting it as freestyle:

Selecting build type as “Execute shell”:

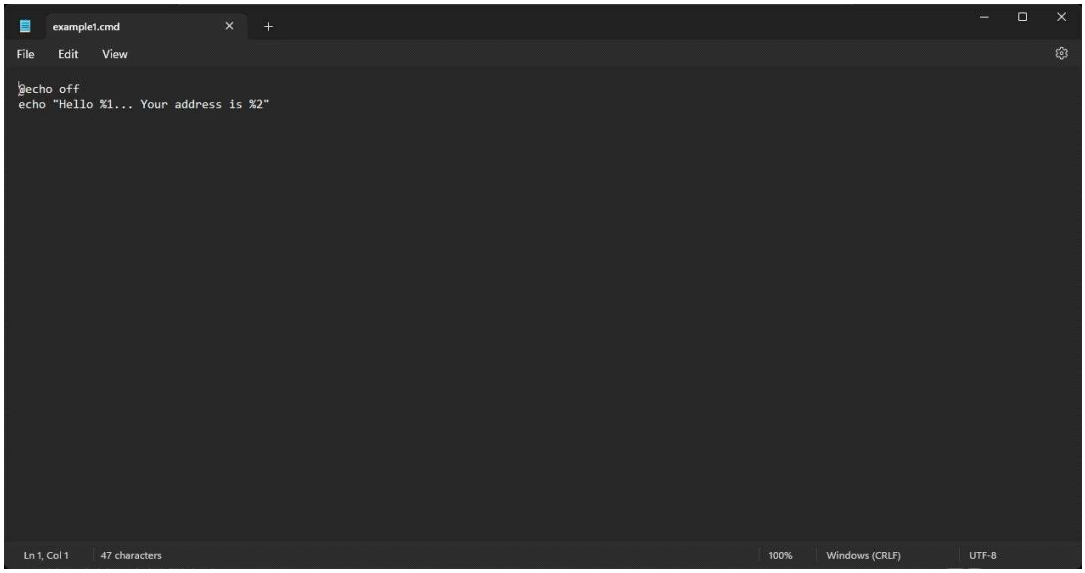


Entering a simple command for the shell execution:

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:

```
Microsoft Windows [Version 10.0.22621.3296]
(c) Microsoft Corporation. All rights reserved.

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\SEPM>example1.cmd
The system cannot find the path specified.

C:\Users\AI&DS 202>"Hello... Your address is "
'"Hello... Your address is "' 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>example1.cad Tanishq
The system cannot find the path specified.

C:\Users\AI&DS 202>"Hello Tanihsq... Your address is "
'"Hello Tanihsq... Your address is "' 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>example1.cmd Tanishq Girgaon "Helle Tanishq... Your address is Gi
rgaon"
The system cannot find the path specified.
```

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

Console output after building the modified project:

<u>Running a Java program under Jenkins</u>

Creating a simple Java program:

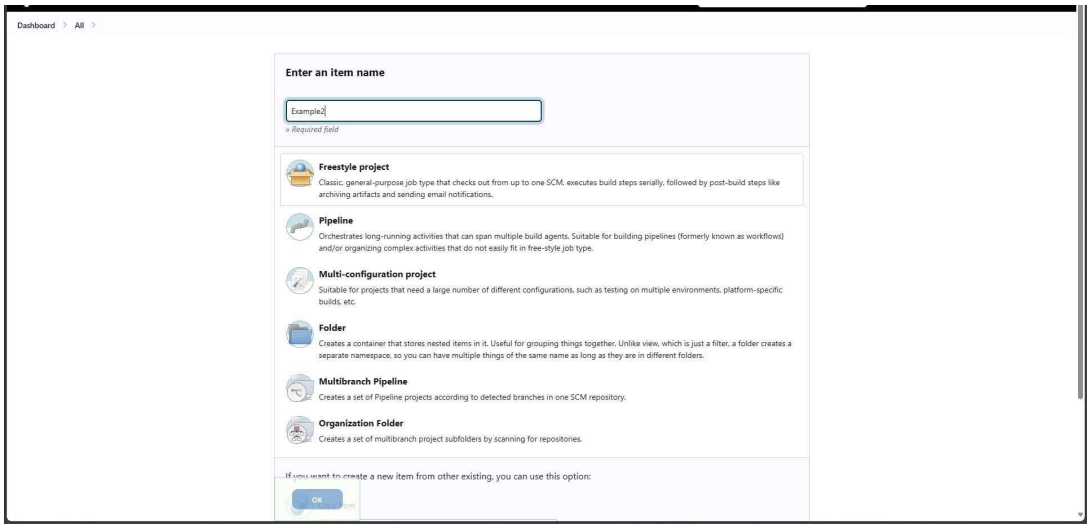
Compiling and running the program on the terminal:

```
C:\Users\richminds\Desktop\sepm>javac 24.java

C:\Users\richminds\Desktop\sepm>java 24.java
This is T12

C:\Users\richminds\Desktop\sepm>
```

Creating a new freestyle project:



Configure new project:

Command

See the list of available environment variables

```
javac C:\Users\richminds\Desktop\sepm\24.java
java C:\Users\richminds\Desktop\sepm\24.java
```

Console output after building:

Console Output Download Copy View as plain text

```
Started by user Aditya Dikonda
Running as SYSTEM
Building in workspace C:\ProgramData\Jenkins\.jenkins\workspace\24_34_31_45_40_42_37_41
[24_34_31_45_40_42_37_41] $ cmd /c call C:\WINDOWS\TEMP\jenkins3970528995341461278.bat

C:\ProgramData\Jenkins\.jenkins\workspace\24_34_31_45_40_42_37_41>javac C:\Users\richminds\Desktop\sepm\24.java

C:\ProgramData\Jenkins\.jenkins\workspace\24_34_31_45_40_42_37_41>java C:\Users\richminds\Desktop\sepm\24.java
This is T12

C:\ProgramData\Jenkins\.jenkins\workspace\24_34_31_45_40_42_37_41>exit 0
Finished: SUCCESS
```

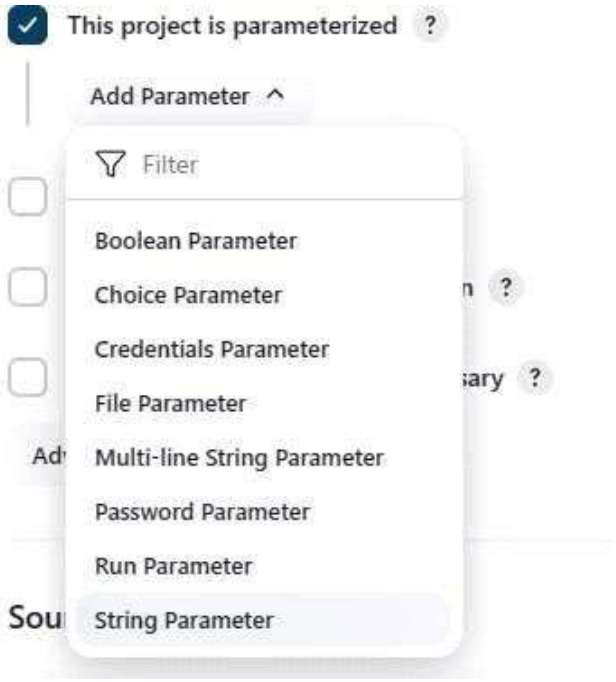
Example 3

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:

Configuring build steps:

Entering parameters for build:

Console output after building:

Console Output

```
Started by user Siddhant Chetlur
Running as SYSTEM
[EnvInject] - Loading node environment variables.
Building in workspace C:\ProgramData\Jenkins\.jenkins\workspace\Example3
[Example3] $ cmd /c call C:\WINDOWS\TEMP\jenkins1409453616515098615l.bat

C:\ProgramData\Jenkins\.jenkins\workspace\Example3>C:\Admin\Academics\TSEC\Start3\SEPM\example3.cmd Siddhant Bandra
Hello your name is Siddhant and your city is Bandra
Finished: SUCCESS
```

Example 5

example5.py

File Edit View

```
import sys

num = int(sys.argv[1])

print(f"Original number is {num}")
print(f"Binary representation of {num} is {bin(num)}")
print(f"Octal representaton of {num} is {oct(num)}")
print(f"Hexadecimal represrentation of {num} is {hex(num)}")
print(f"Complex representation of {num} is {complex(num)}")
```

Ln 9, Col 60 | 299 characters | 100% | Windows (CRLF) | UTF-8

Example 5.1: Running a Python

program Creating a simple Python

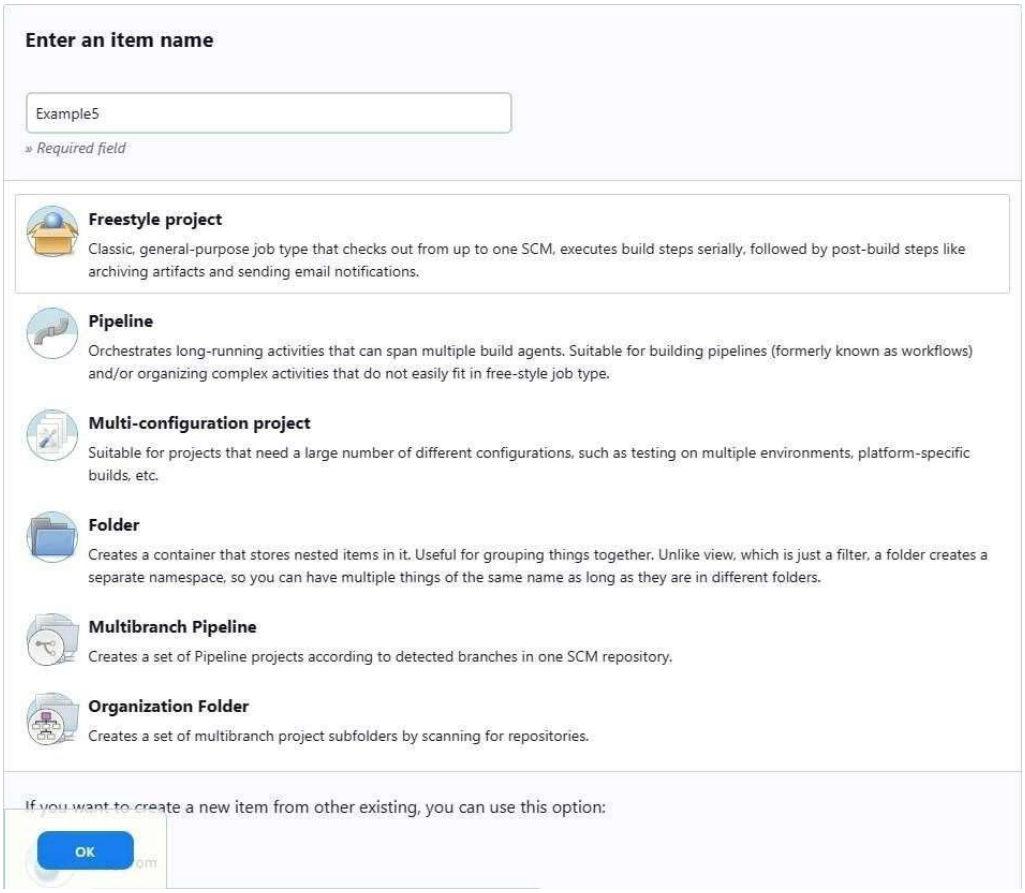
script:

Running the Python script on the terminal:

```
C:\Users\richminds\Desktop\sepm>python 24.py 10
Number is 10

C:\Users\richminds\Desktop\sepm>
```

Creating a new freestyle project:



Parameterising the project with a string parameter as follows:

Configuring the build steps:

Command

See the list of available environment variables

```
python C:\Users\richminds\Desktop\sepm\24.py
```

Setting the parameter for the build:

Conclusion: Thus, we have successfully studied Continuous Integration and installed, configured, and understood programming with Jenkins.

Experiment 06

Aim: To Study Agile Methodology and Test case Management using JIRA Tool

Theory:

Jira is a software development tool that helps team's plan, track, and release software projects. It's a popular tool among software development teams. JIRA is a widely used issue tracking and project management software developed by Atlassian.

The theory behind JIRA revolves around several key principles and concepts:

Issue Tracking: At its core, JIRA is designed to help teams track tasks, bugs, features, and other issues within a project. It provides a centralised platform for capturing, organising, and prioritising these items.

Agile Methodologies: JIRA supports agile methodologies such as Scrum and Kanban, enabling teams to plan, track, and deliver work in iterative cycles. Agile principles emphasise collaboration, adaptability, and delivering value to customers.

Workflow Management: JIRA allows teams to define custom workflows that represent the lifecycle of an issue from creation to completion. Workflows can include various statuses, transitions, and conditions to reflect the specific processes followed by the team.

Customization: One of the key strengths of JIRA is its flexibility and customization options. Teams can tailor JIRA to their specific needs by creating custom issue types, fields, workflows, and project configurations.

Collaboration and Communication: JIRA facilitates collaboration among team members by providing features such as commenting, mentioning, and attaching files to issues. It also integrates with other Atlassian products like Confluence and Bit bucket for seamless communication and knowledge sharing.

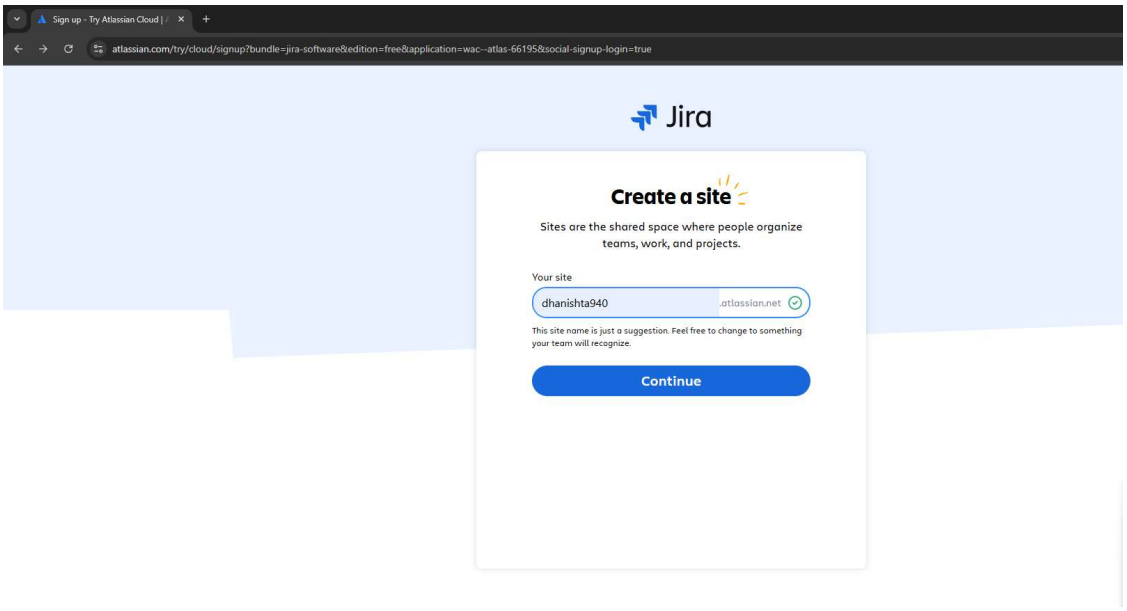
Reporting and Analytics: JIRA offers robust reporting and analytics capabilities, allowing teams to gain insights into their project progress, team performance, and other key metrics. Users can create custom dashboards, charts, and reports to monitor and analyse project data.

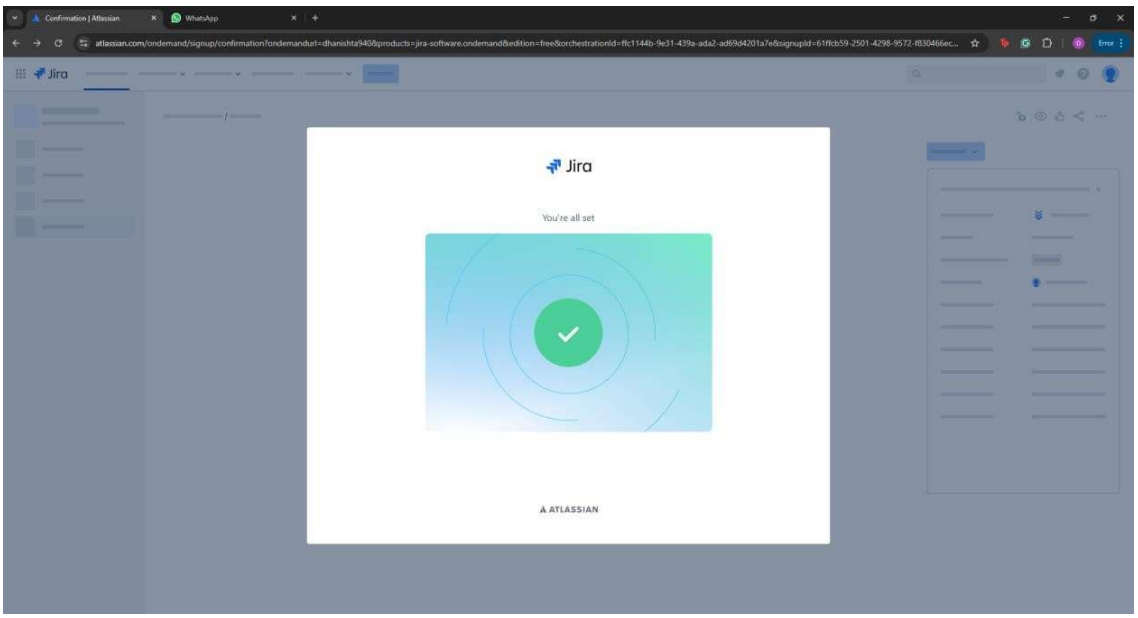
Integration: JIRA integrates with a wide range of third-party tools and services, including development tools, CI/CD pipelines, customer support systems, and more. This integration ecosystem enables teams to streamline their workflows and improve productivity.

Scalability and Performance: JIRA is designed to scale with the needs of growing teams and organisations. It can handle large volumes of data and users while maintaining performance and reliability.

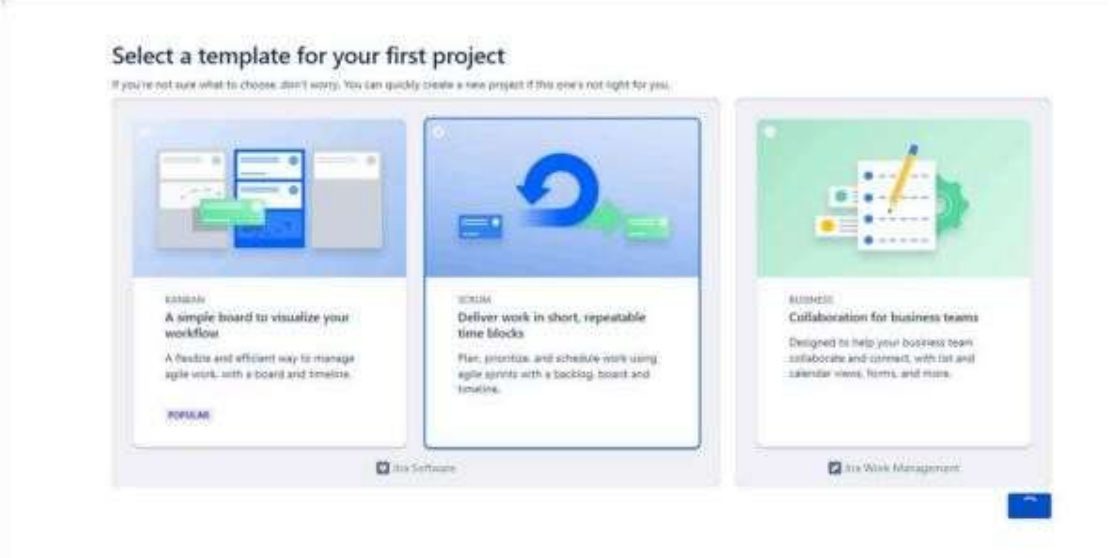
Steps:

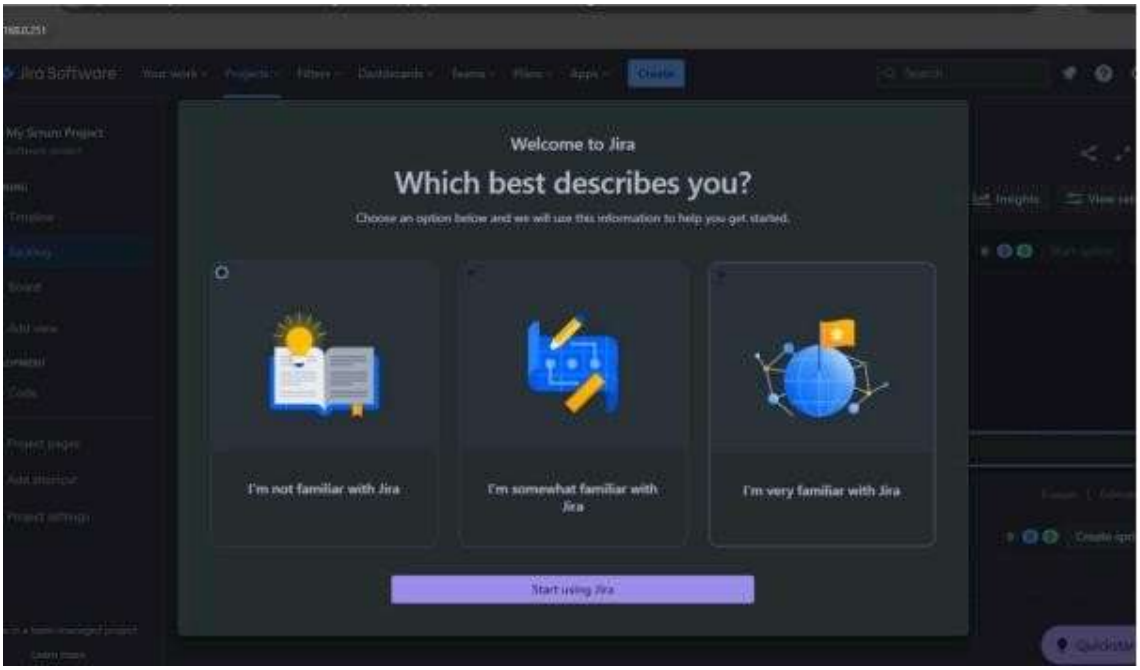
- Open altassian.com, select JIRA and Login with the required Credentials



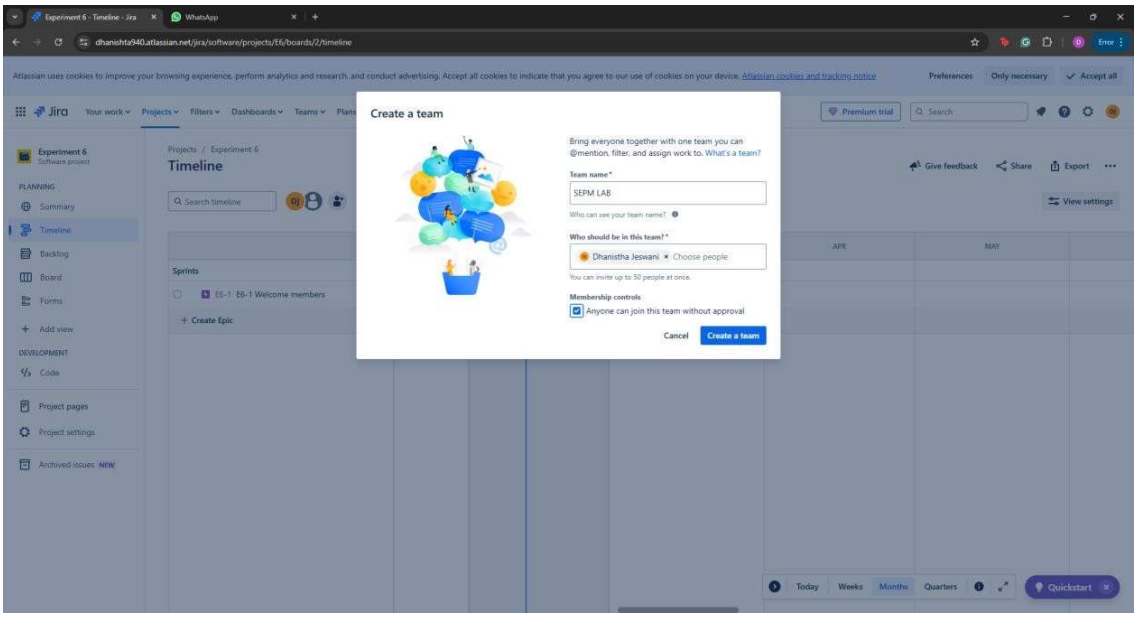



ATLASSIAN
One moment, your site is starting up
Thanks for signing up! Our robots are working on your Atlassian Cloud site.
This won't take more than a minute or two. You'll be taken there once it's ready.

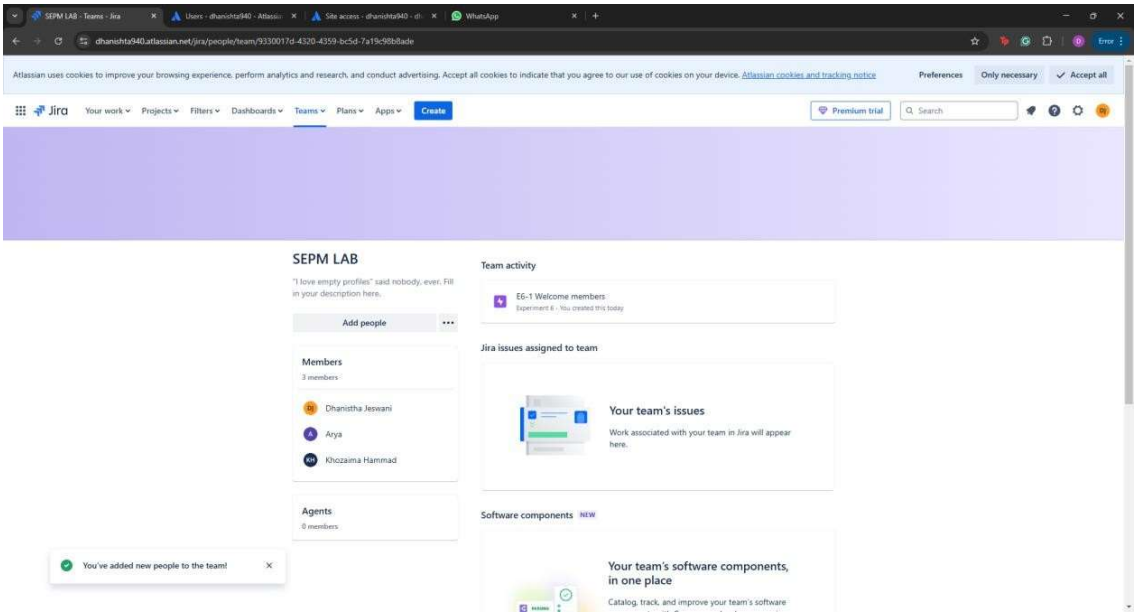

Select a template for your first project
A simple board to visualize your workflow
Deliver work in short, repeatable time blocks
Collaboration for business teams



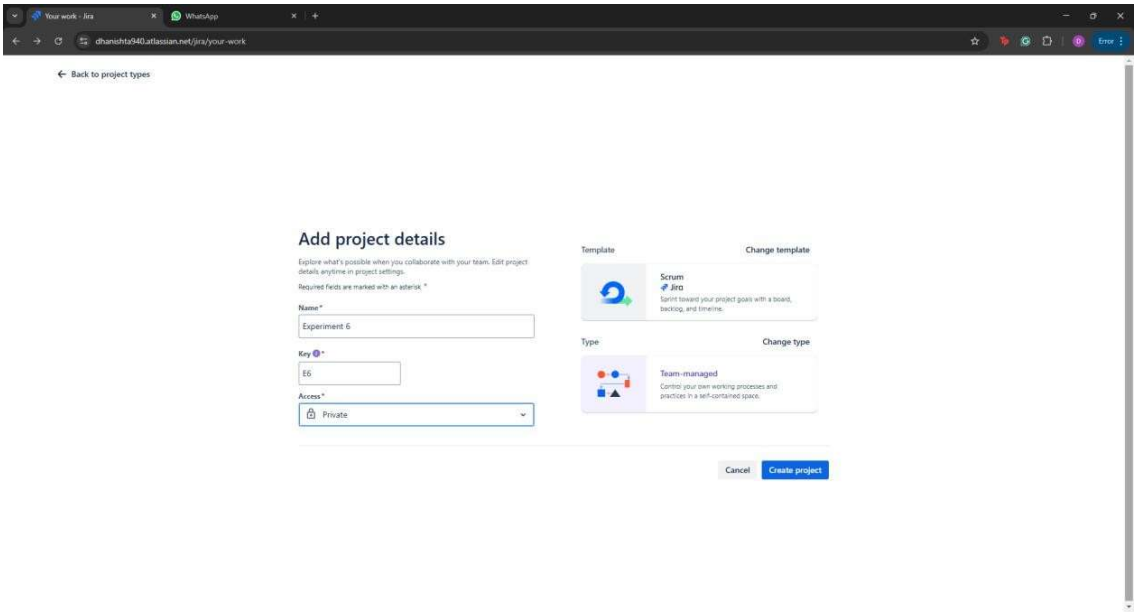
- Creating a team



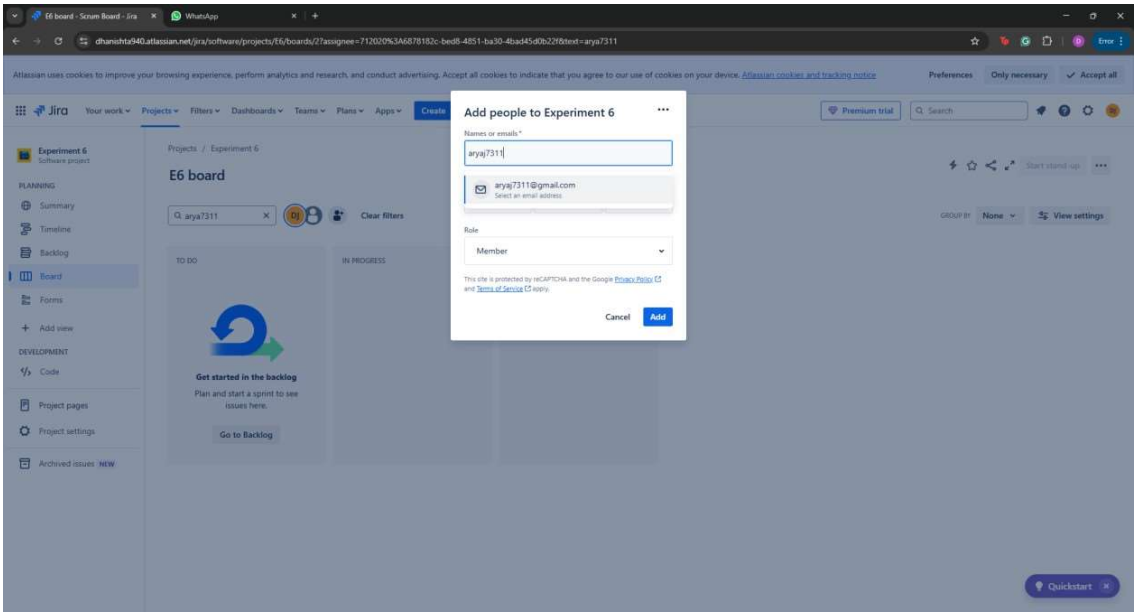
- Adding members to the team



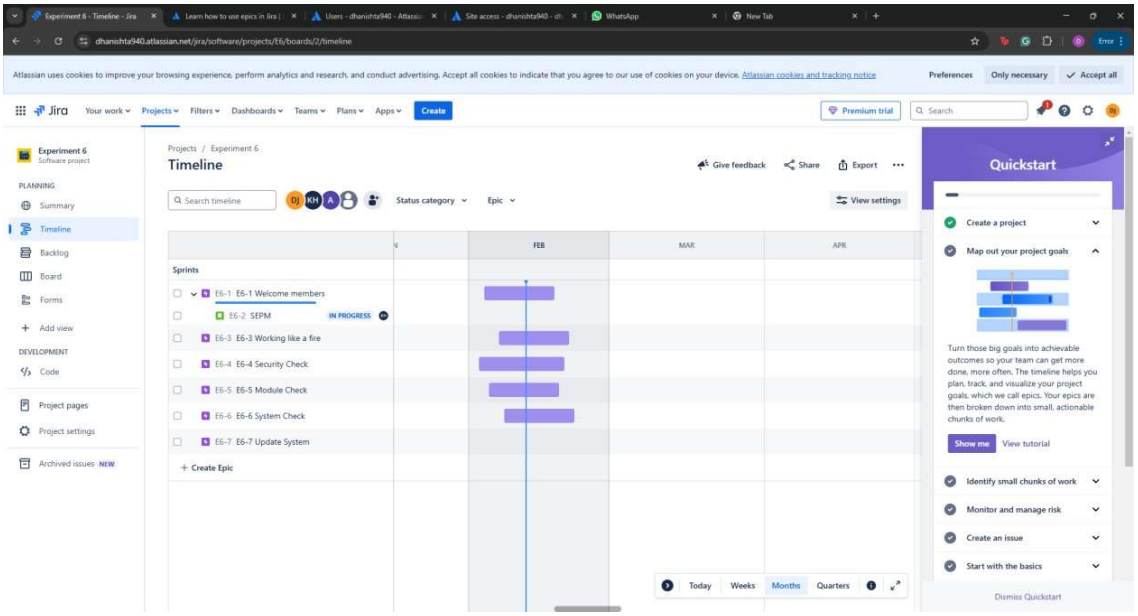
- Creating a project

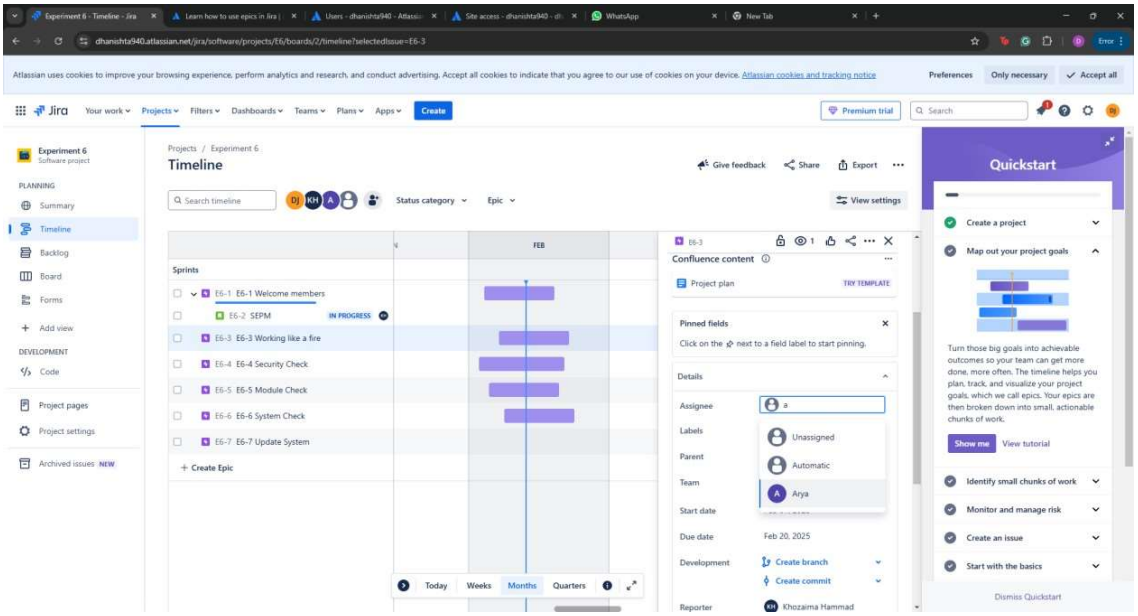


- Adding people to the project

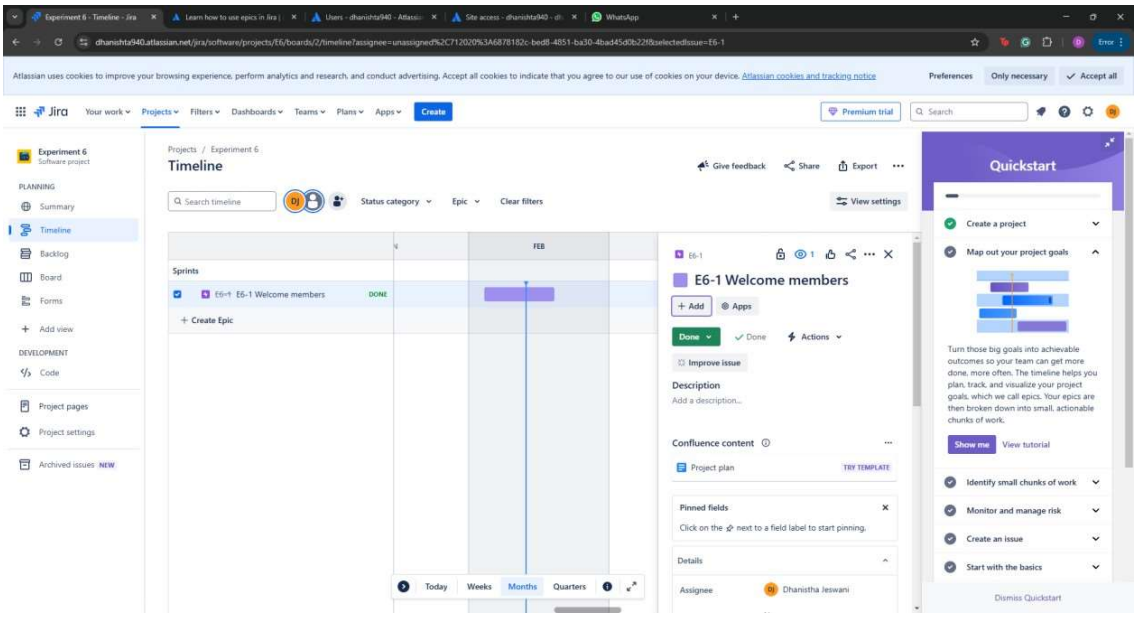


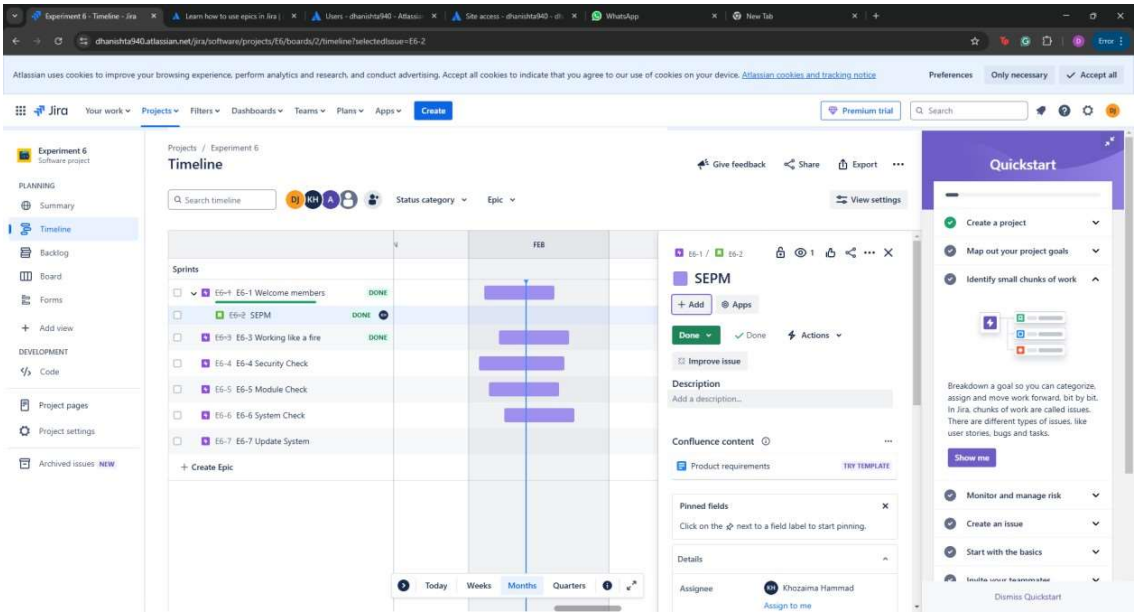
- Adding and assigning tasks





- Task completion





Conclusion: Hence, we have successfully implemented Agile Methodologies in JIRA.

NIRJARA SONI
T22-106

Experimnt 07

Aim: To Create Scrum Board for Scrum Master using JIRA Tool.

Theory:

To create a Scrum Board for a Scrum Master using the JIRA tool, you need to follow these steps:

Create a Scrum Project in JIRA:

Log in to your JIRA account and select a template from the library, choosing the Scrum template.

Once the project is created, you will land on the empty backlog, also known as the product backlog, containing a list of potential work items for the project

Create User Stories or Tasks in the Backlog:

In JIRA, work items like user stories, tasks, and bugs are referred to as "issues." Create user stories using the quick create option on the backlog.

User stories describe work items in a non-technical language from a user's perspective, following the format: "As a {type of user}, I want {goal} so that I {receive benefit}"

Prioritize User Stories in the Backlog:

After creating user stories, prioritize them by ranking or dragging and dropping them in the order they should be worked on.

The product owner usually prioritizes user stories, and the development team estimates the effort required to complete each story

Create a Scrum Board in JIRA:

Click on Search > View all boards and then Create board.

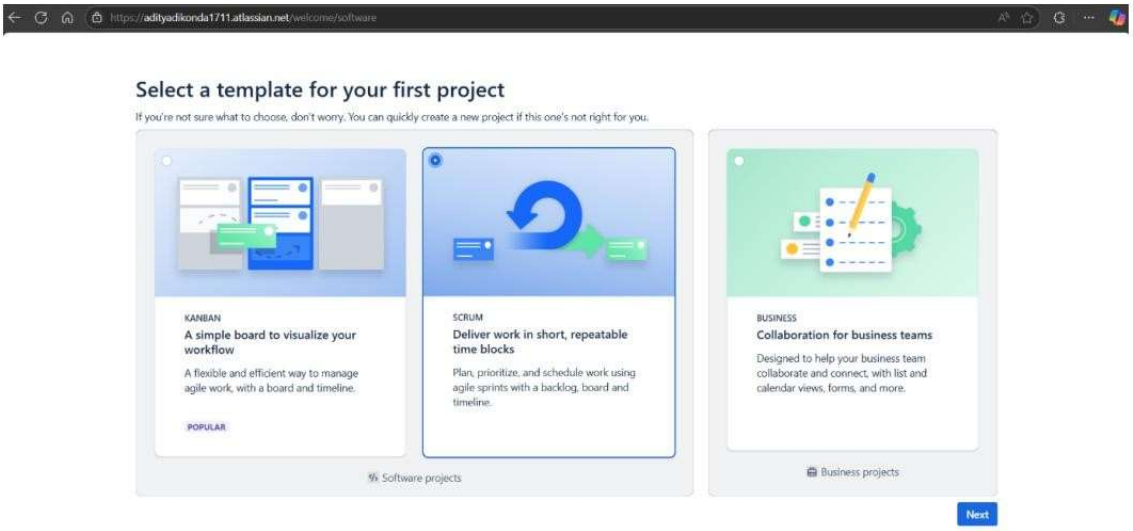
Select the board type as Scrum and choose whether to start with a new project template or add the board to existing projects.

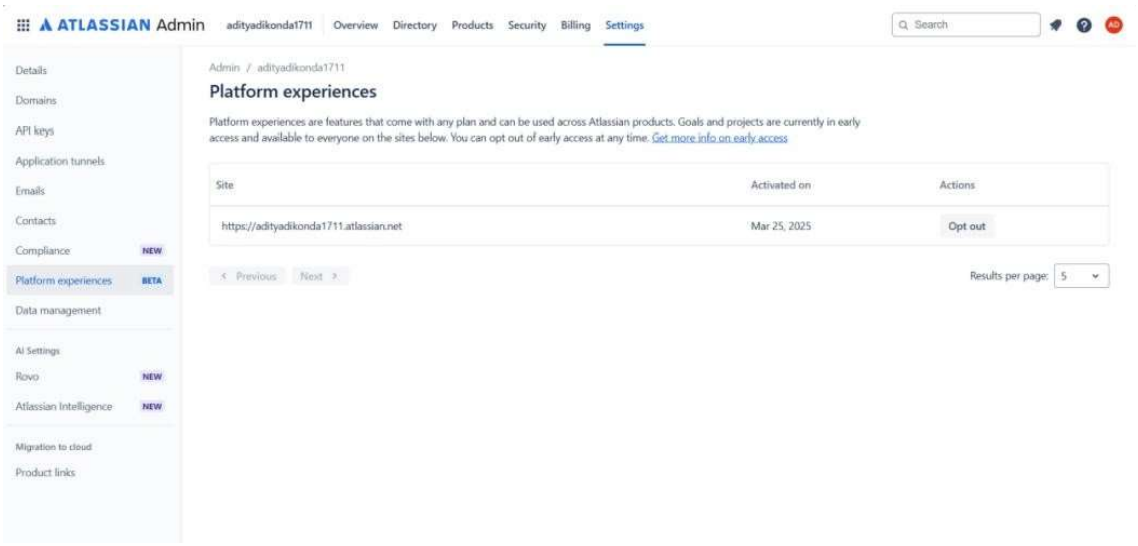
Configure columns and quick filters to reflect your team's process and focus on specific issues quickly

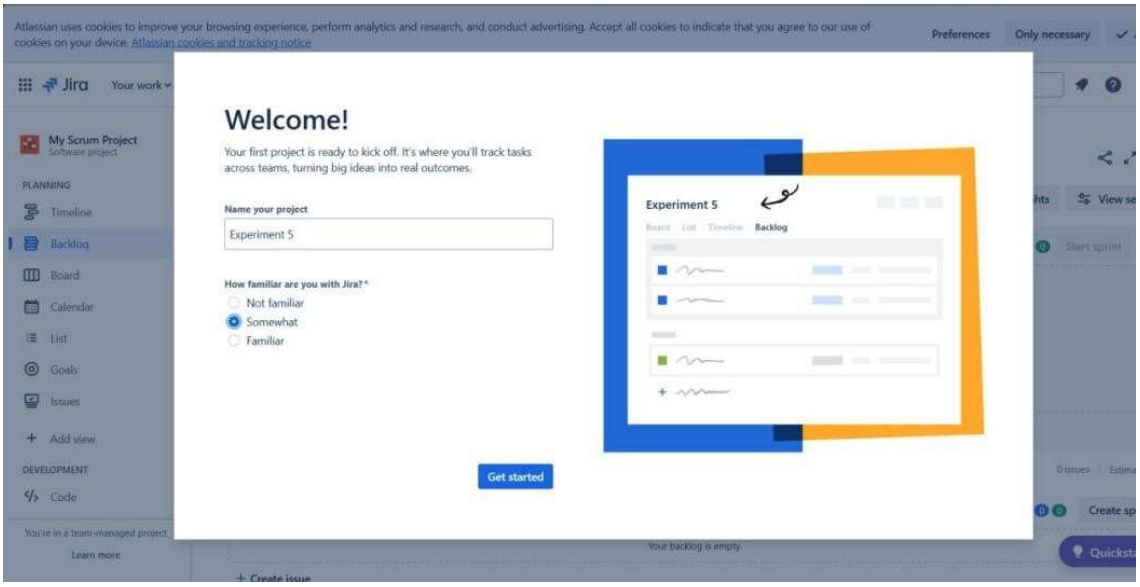
Navigate Between Boards:

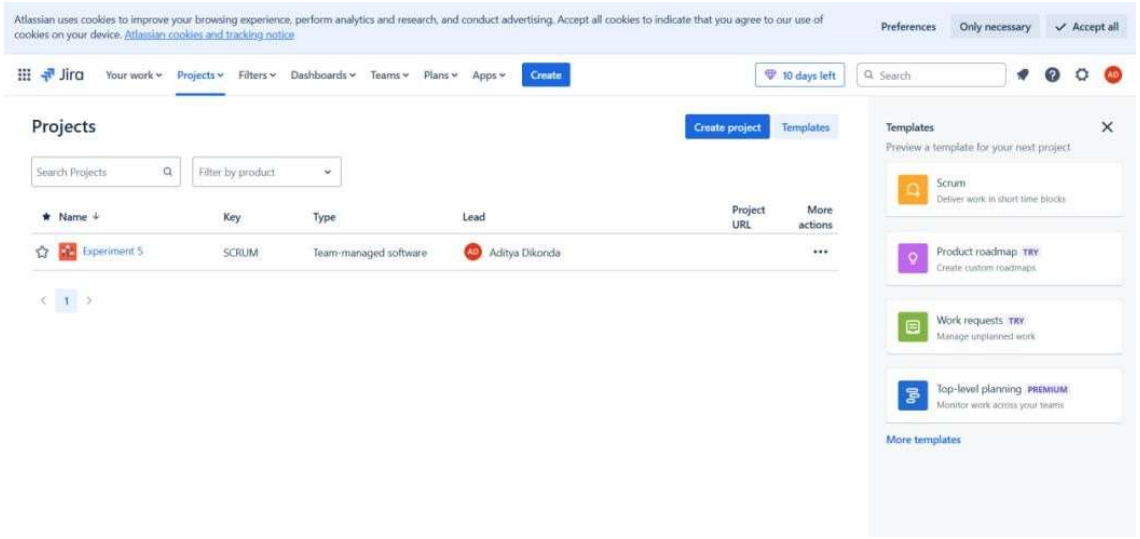
Use the board switcher located in the left-hand menu under the project name to move between different boards in JIRA

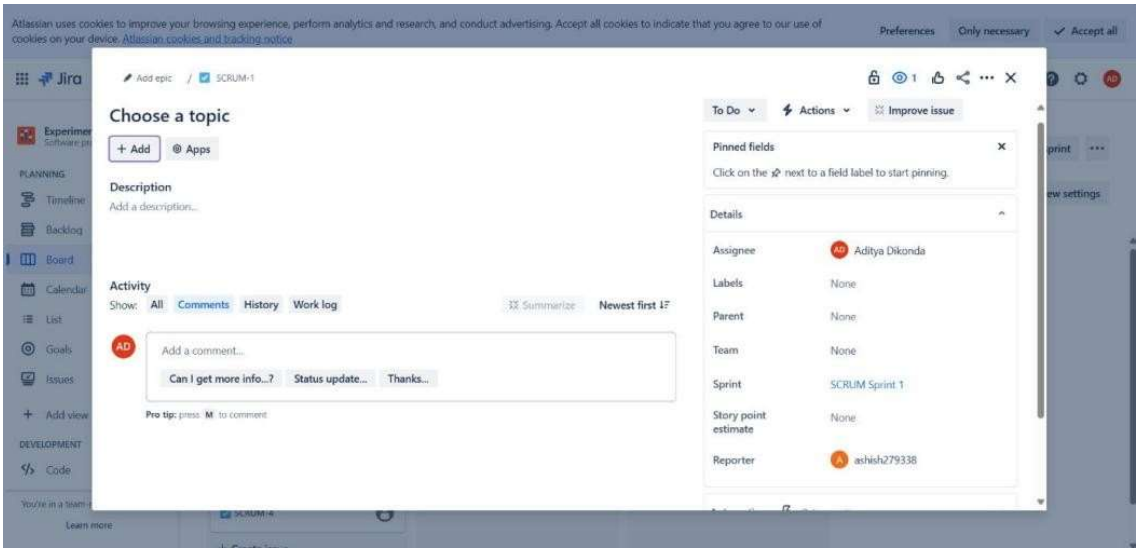
By following these steps, a Scrum Master can effectively create a Scrum Board in JIRA, enabling efficient management of tasks, user stories, and sprints within the agile project management framework.

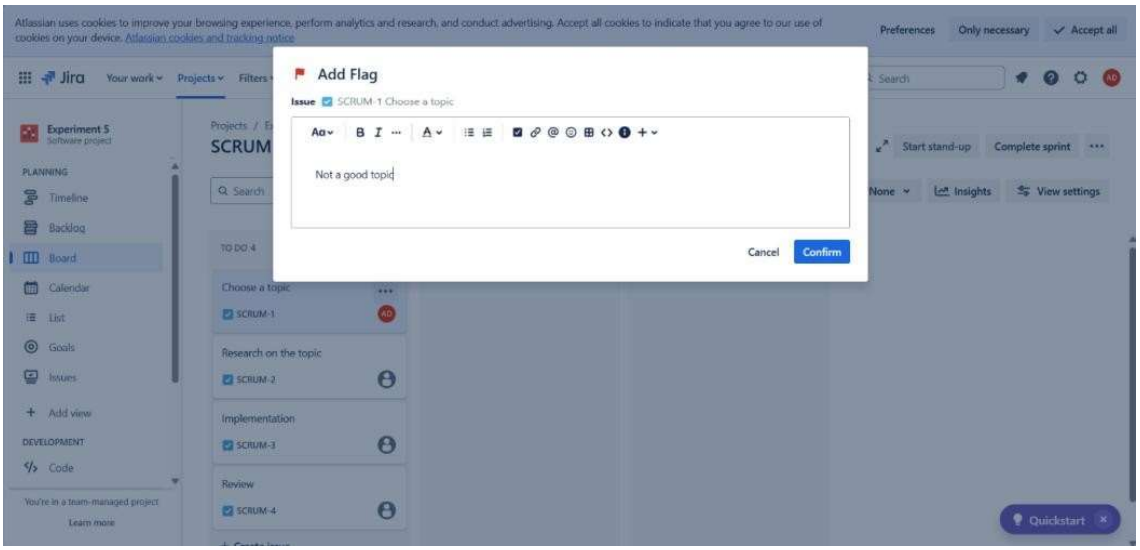


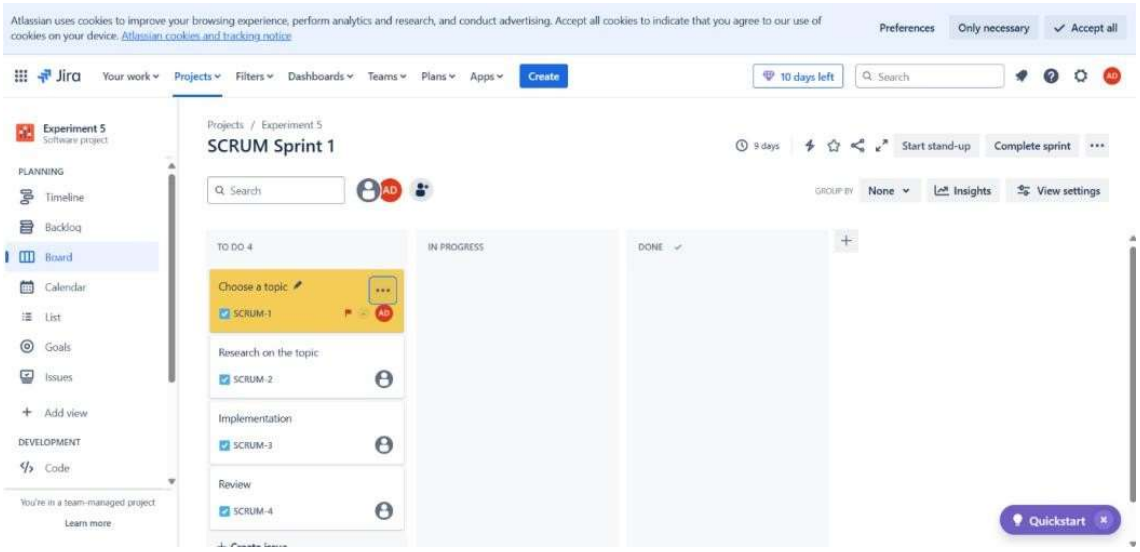


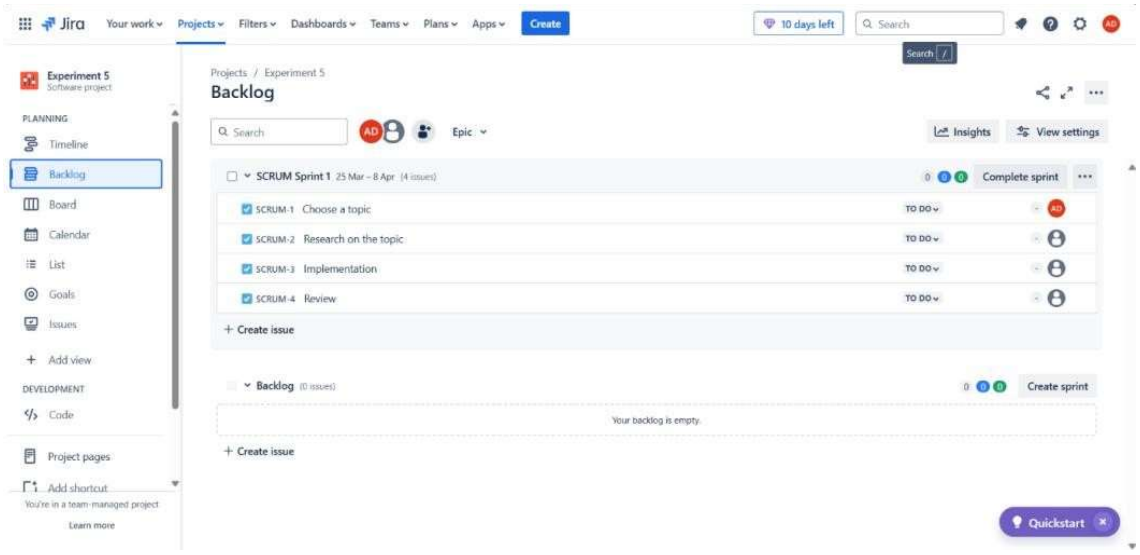

Welcome!
Your first project is ready to kick off. It's where you'll track tasks across teams, turning big ideas into real outcomes.
Name your project
Experiment 5
How familiar are you with Jira?*
Not familiar
Somewhat
Familiar
Get started
Experiment 5
Backlog

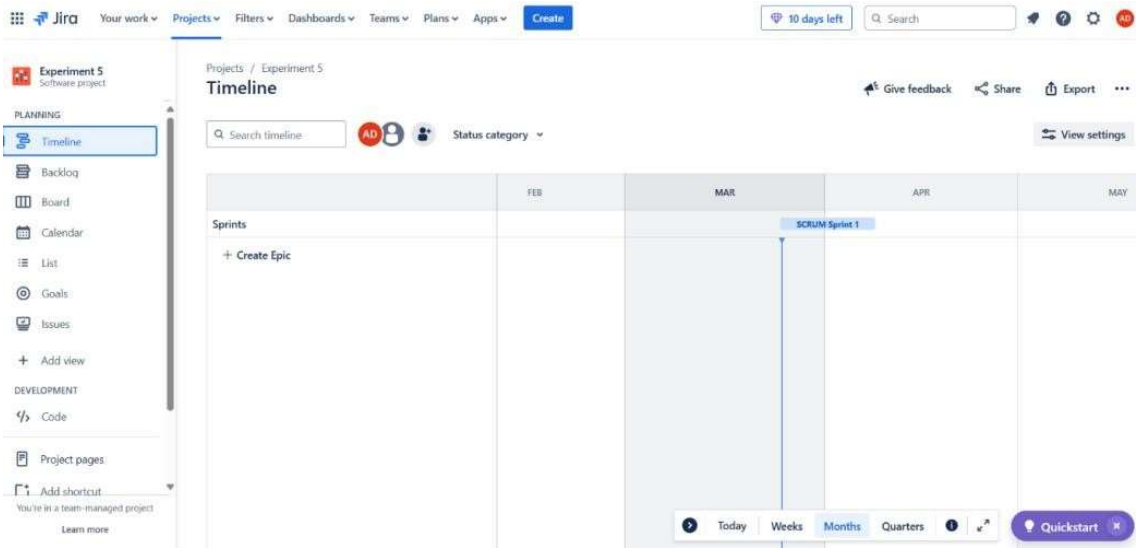

Projects
Create project
Templates
Search Projects
Filter by product
Name
Key
Type
Lead
Project URL
More actions
Experiment 5
SCRUM
Team-managed software
Aditya Dikonda
Templates
Preview a template for your next project
Scrum
Product roadmap TRY
Work requests TRY
Top-level planning PREMIUM
More templates

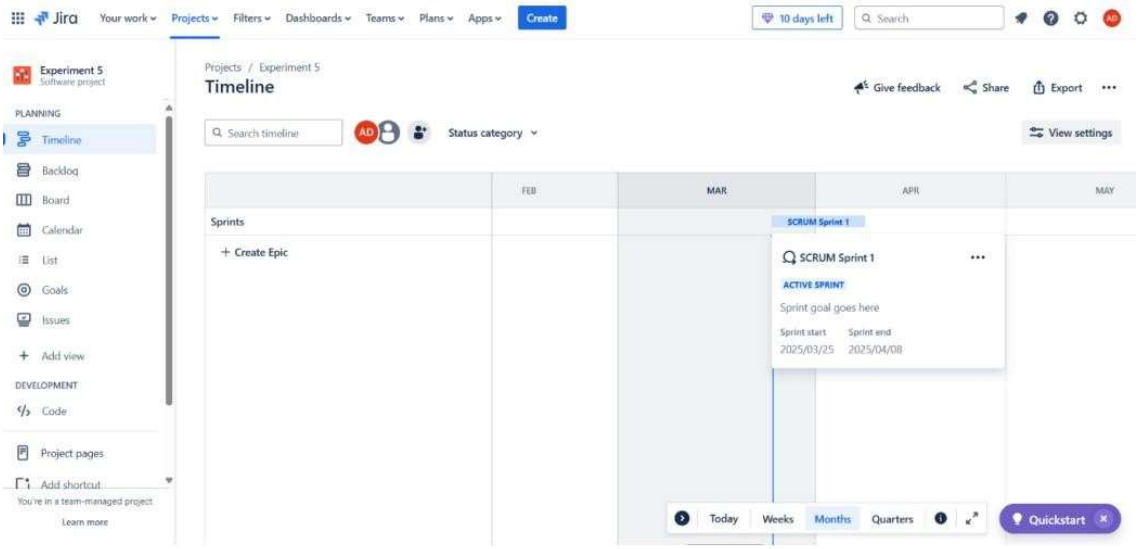

Choose a topic
+ Add
Apps
Description
Add a description...
Activity
Show: All Comments History Work log
Newest first
Add a comment...
Can I get more info...?
Status update...
Thanks...
Pro tip: press M to comment
To Do
Actions
Improve issue
Pinned fields
Click on the next to a field label to start pinning.
Details
Assignee
Aditya Dikonda
Labels
None
Parent
None
Team
None
Sprint
SCRUM Sprint 1
Story point estimate
None
Reporter
ashish279338


Add Flag
Issue SCRUM-1 Choose a topic
Not a good topic
Cancel
Confirm
Choose a topic
Research on the topic
Implementation
Review


SCRUM Sprint 1
TO DO 4
IN PROGRESS
DONE
Choose a topic
Research on the topic
Implementation
Review


Backlog
SCRUM Sprint 1 25 Mar – 8 Apr
SCRUM-1 Choose a topic
SCRUM-2 Research on the topic
SCRUM-3 Implementation
SCRUM-4 Review
Create issue
Backlog
Your backlog is empty.





Conclusion: Scrum project was implemented using JIRA.

Experiment 08

Aim: To Study Project Scheduling Using Gantt chart in ClickUp.

Theory:

Project Scheduling-

A project schedule is a timeline that outlines the tasks, milestones, deadlines, and resources required for completing a project. It helps project managers organise and plan the sequence of activities, allocate resources efficiently, set realistic timelines, and monitor progress. Having a clear project schedule is crucial for a project manager as it ensures tasks are completed on time, helps in managing resources effectively, allows for better coordination among team members, and assists in identifying potential issues or delays, enabling timely adjustments to keep the project on track.

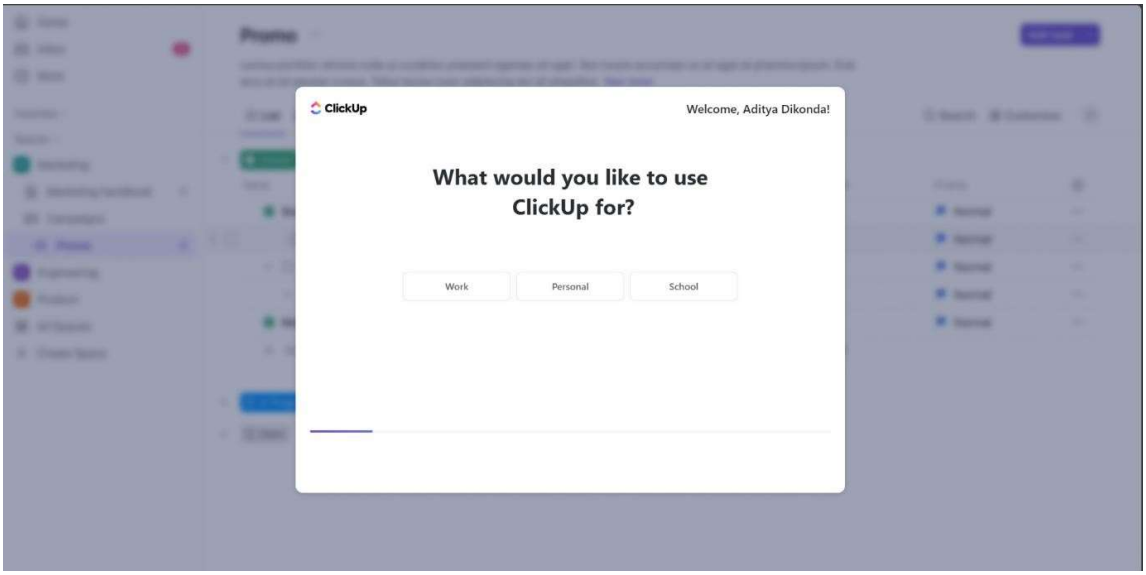
Gantt chart-

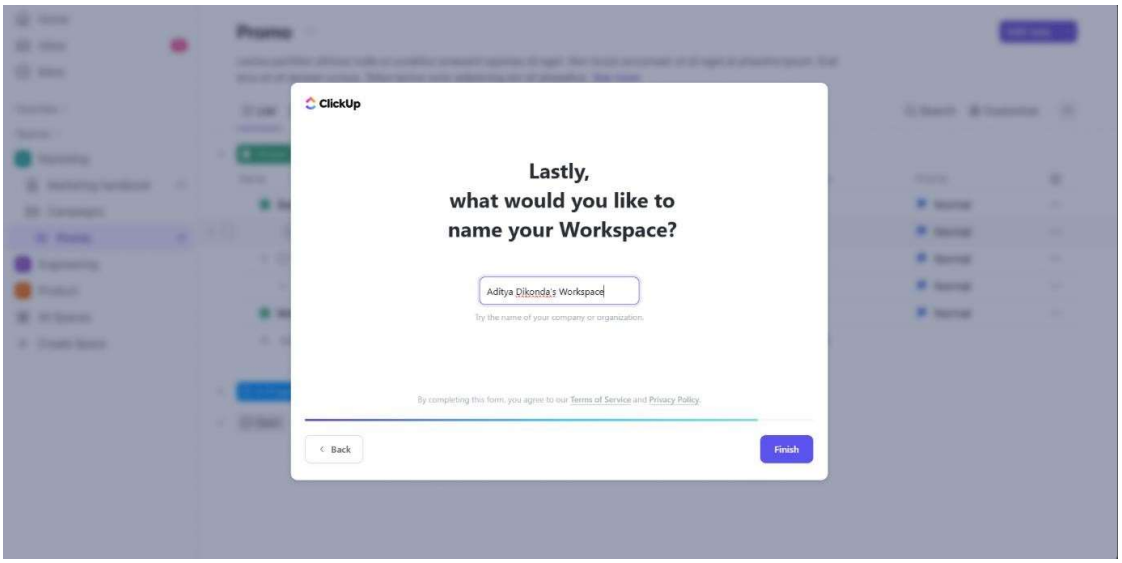
A Gantt chart, commonly used in project management, is one of the most popular and useful ways of showing activities (tasks or events) displayed against time. On the left of the chart is a list of the activities and along the top is a suitable time scale. Each activity is represented by a bar; the position and length of the bar reflects the start date, duration and end date of the activity.

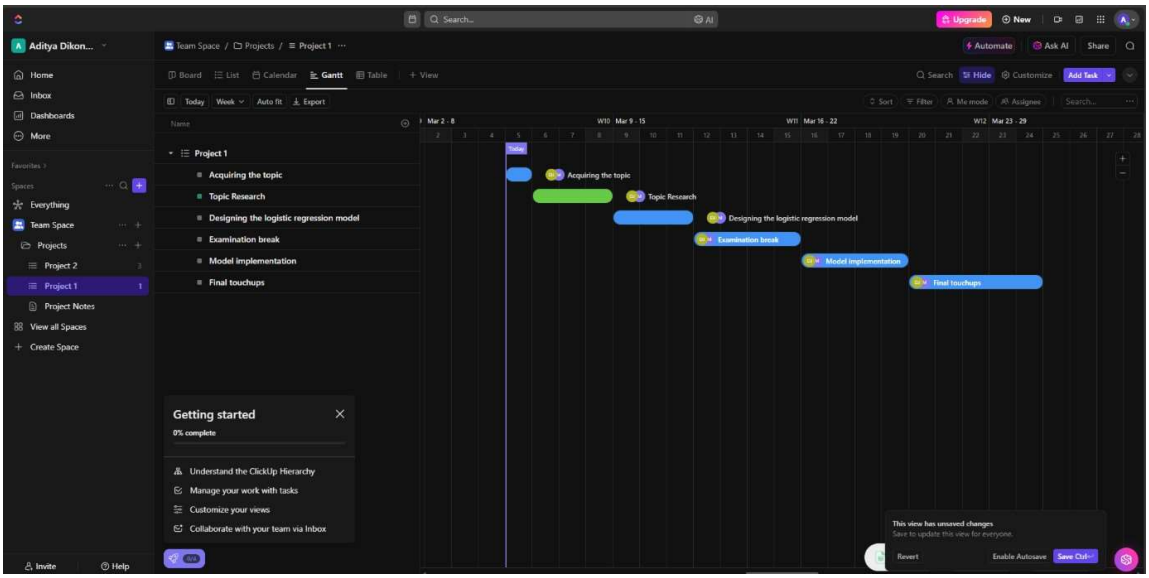
About the topic-

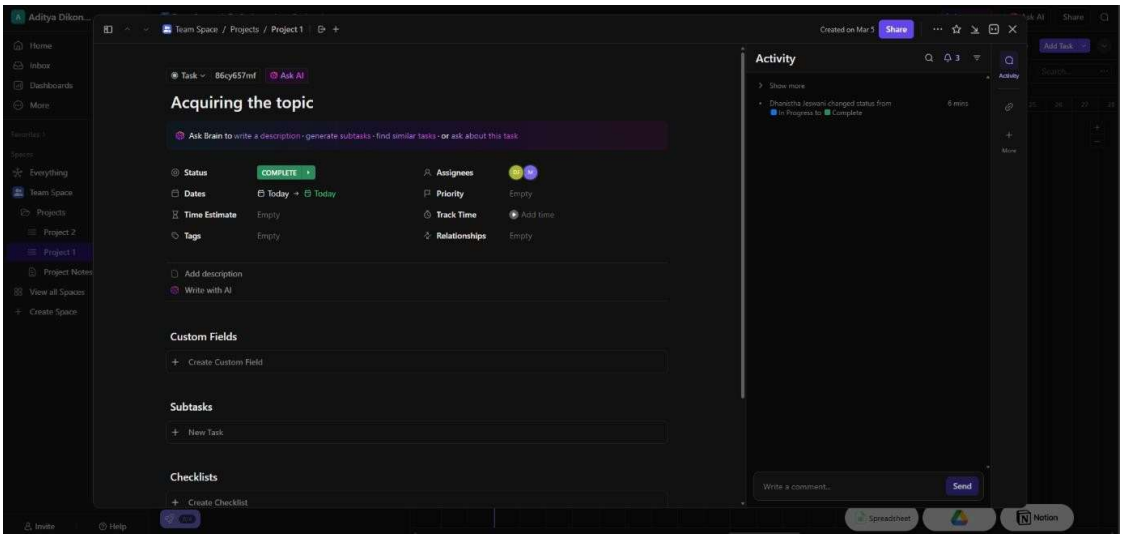
It is a Logistic regression Model designed on Agricultural decision making based on soil characteristics and various other environmental factors. Our dataset Encompasses of essential parameters such as soil composition (Nitrogen,potassium and phosphorus contents of the soil as well as the pH level of the soil) and location specific variables such as temperature, humidity and rainfall. Our model leverages these inputs and aims to predicts the most suitable crop for a given soil and environmental conditions.

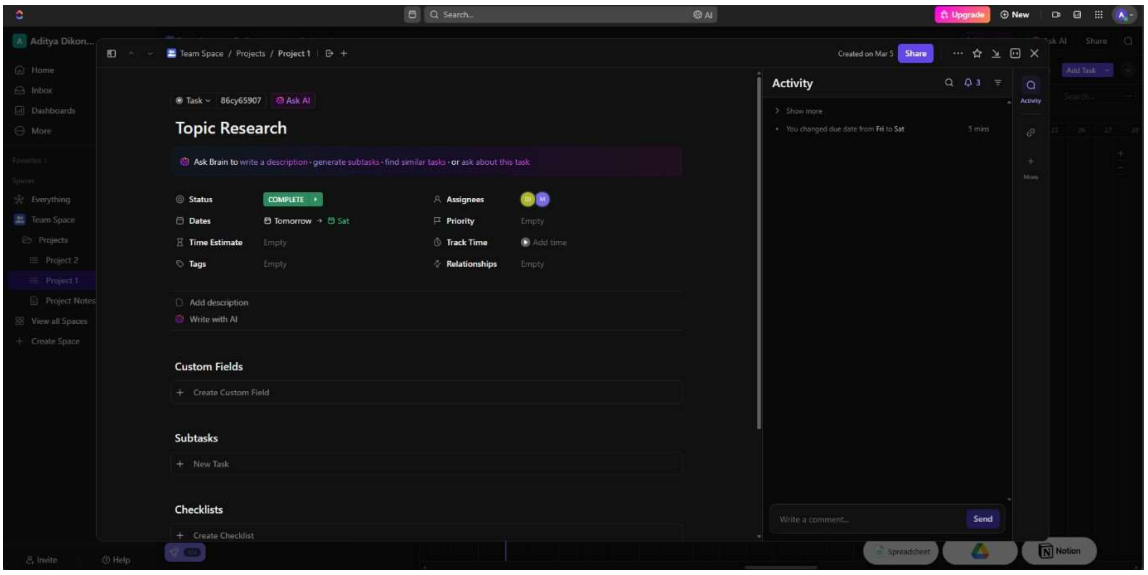
Output:

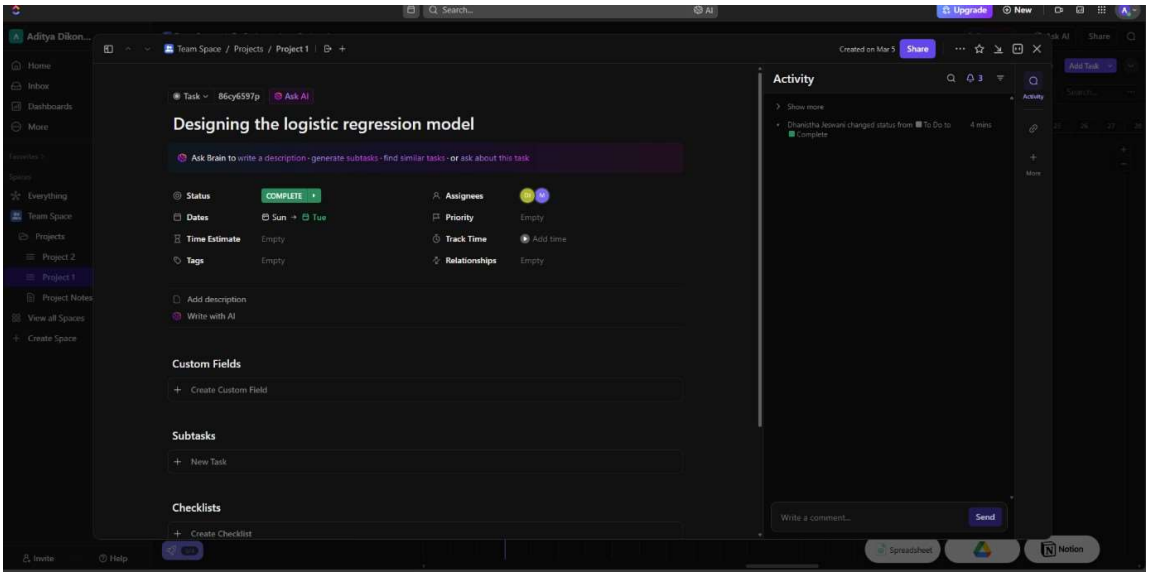

ClickUp
Welcome, Aditya Dikonda!
What would you like to use ClickUp for?
Work
Personal
School

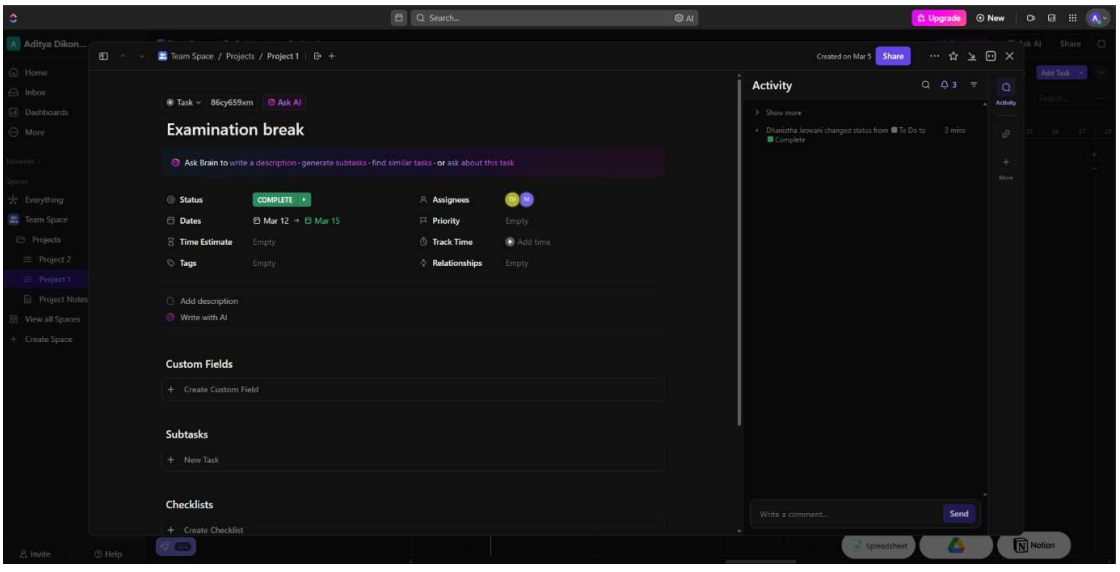

ClickUp
Lastly, what would you like to name your Workspace?
Aditya Dikonda's Workspace
Try the name of your company or organization.
By completing this form, you agree to our Terms of Service and Privacy Policy.
Back
Finish

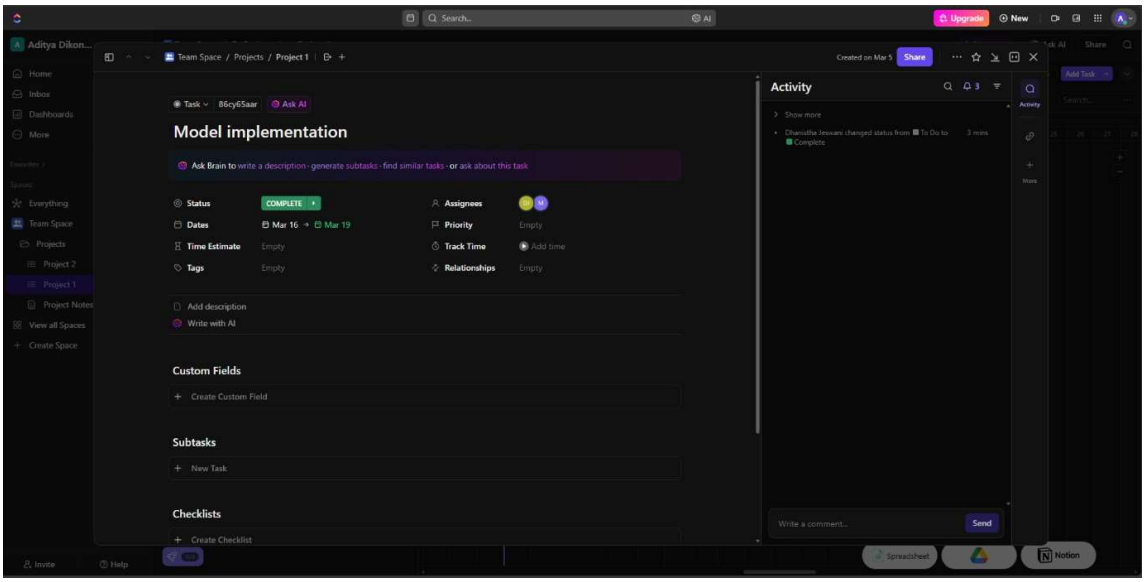

Aditya Dikon...
Team Space / Projects / Project 1
Automate
Ask AI
Share
Home
Inbox
Dashboards
More
Board
List
Calendar
Gantt
Table
+ View
Search
Hide
Customize
Add Task
Today
Week
Auto fit
Export
Sort
Filter
Me mode
Assignees
Search...
Mar 2 - 8
W10 Mar 9 - 15
W11 Mar 16 - 22
W12 Mar 23 - 29
Project 1
Acquiring the topic
Topic Research
Designing the logistic regression model
Examination break
Model implementation
Final touchups
Everything
Team Space
Projects
Project 2
Project 1
Project Notes
View all Spaces
Create Space
Getting started
0% complete
Understand the ClickUp Hierarchy
Manage your work with tasks
Customize your views
Collaborate with your team via Inbox
This view has unsaved changes
Save to update this view for everyone.
Revert
Enable Autosave
Save Ctrl+
Invite
Help

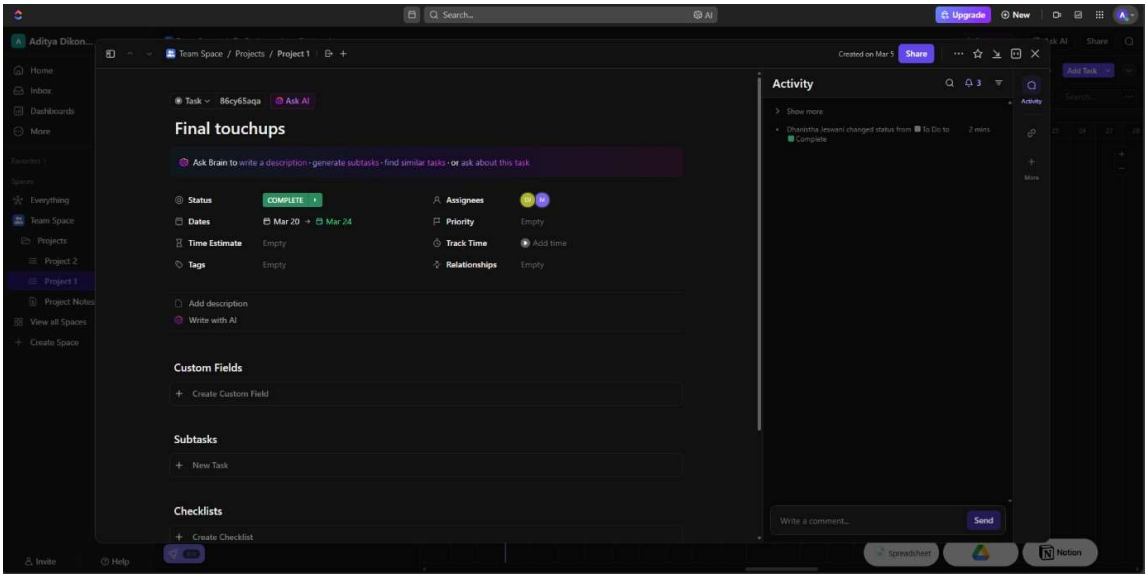

Aditya Dikon...
Team Space / Projects / Project 1
Created on Mar 5
Share
Task
86cy657mf
Ask AI
Acquiring the topic
Ask Brain to write a description · generate subtasks · find similar tasks · or ask about this task
Status
COMPLETE
Assignees
Dates
Today → Today
Priority
Empty
Time Estimate
Empty
Track Time
Add time
Tags
Empty
Relationships
Empty
Add description
Write with AI
Custom Fields
Create Custom Field
Subtasks
New Task
Checklists
Create Checklist
Activity
Show more
Dhanistha Jeswani changed status from
In Progress to Complete
6 mins
Write a comment...
Send
Spreadsheet
Notion
Everything
Team Space
Projects
Project 2
Project 1
Project Notes
View all Spaces
Create Space
Invite
Help

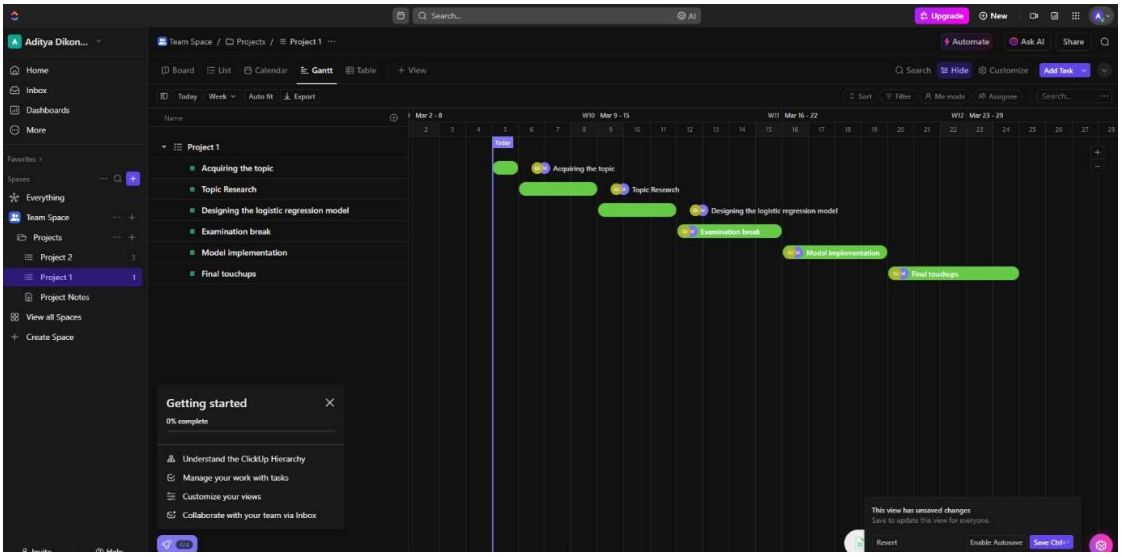

Topic Research
Ask Brain to write a description · generate subtasks · find similar tasks · or ask about this task
Status
COMPLETE
Dates
Tomorrow → Sat
Time Estimate
Tags
Assignees
Priority
Track Time
Relationships
Add description
Write with AI
Custom Fields
Create Custom Field
Subtasks
New Task
Checklists
Create Checklist
Activity
Write a comment...
Send


Designing the logistic regression model
Ask Brain to write a description · generate subtasks · find similar tasks · or ask about this task
Status
COMPLETE
Dates
Sun → Tue
Time Estimate
Tags
Assignees
Priority
Track Time
Relationships
Add description
Write with AI
Custom Fields
Create Custom Field
Subtasks
New Task
Checklists
Create Checklist
Activity
Write a comment...
Send


Examination break
Ask Brain to write a description · generate subtasks · find similar tasks · or ask about this task
Status
COMPLETE
Dates
Mar 12 → Mar 15
Time Estimate
Tags
Assignees
Priority
Track Time
Relationships
Add description
Write with AI
Custom Fields
Create Custom Field
Subtasks
New Task
Checklists
Create Checklist
Activity
Write a comment...
Send


Model implementation
Ask Brain to write a description · generate subtasks · find similar tasks · or ask about this task
Status
COMPLETE
Dates
Mar 16 → Mar 19
Time Estimate
Tags
Assignees
Priority
Track Time
Relationships
Add description
Write with AI
Custom Fields
Create Custom Field
Subtasks
New Task
Checklists
Create Checklist
Activity
Write a comment...
Send


Final touchups
Ask Brain to write a description · generate subtasks · find similar tasks · or ask about this task
Status
COMPLETE
Dates
Mar 20 → Mar 24
Time Estimate
Tags
Assignees
Priority
Track Time
Relationships
Add description
Write with AI
Custom Fields
Create Custom Field
Subtasks
New Task
Checklists
Create Checklist
Activity
Write a comment...
Send


Project 1
Acquiring the topic
Topic Research
Designing the logistic regression model
Examination break
Model implementation
Final touchups
Getting started
0% complete
Understand the ClickUp Hierarchy
Manage your work with tasks
Customize your views
Collaborate with your team via Inbox

Conclusion:

This project delved into the application of polynomial regression in financial data analysis, comparing implementations from scratch using Python libraries with those utilizing PyTorch. Through data preprocessing, model implementation, training, and evaluation, we examined the performance of each approach. While both methods proved effective, PyTorch demonstrated advantages in computational efficiency and scalability. This project underscores the importance of selecting appropriate tools and frameworks based on the task's requirements, offering insights into the practical utility of polynomial regression in financial modeling.

NIRJARA SONI
T22-106
TSEC

Experiment 09

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

Theory: Docker is a popular platform that enables developers to build, package, and deploy applications as lightweight, portable, and self-sufficient containers. These containers encapsulate all the necessary dependencies and libraries required for an application to run, ensuring consistency across different environments. Here is a theoretical overview of
Docker:
Containerization:
Docker utilizes containerization technology to create isolated environments for applications. Containers are lightweight, standalone, and executable packages that include everything needed to run an application, such as code, runtime, system tools, libraries, and settings.This isolation ensures that applications run consistently across different environments, from development to production.

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:
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 Docker files, which are text files that define the steps needed to create the image.

Docker Containers:
Containers are instances of Docker images that are running as isolated processes on a host machine. They are lightweight, portable, and can be easily started, stopped, moved, and deleted. Containers provide a consistent environment for applications to run, regardless of the underlying infrastructure.

Benefits of Docker:

Portability: Docker containers can run on any platform that supports Docker, making it easy to deploy applications across different environments.
Efficiency: Containers share the host OS kernel, reducing overhead and improving resource utilization.
Isolation: Containers provide a level of isolation that helps prevent conflicts between applications and dependencies.
Scalability: Docker enables easy scaling of applications by quickly spinning up additional containers.
Consistency: Docker ensures that applications run the same way in development, testing, and production environments.

Output:

```
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>
```

```
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
```

```
C:\Users\202>docker inspect 1c4472744083
[
    {
        "Id": "1c447274408314475b093dcbf93163064b819bdd9ad8378bb3a4fa847dc411d80",
        "Created": "2024-03-13T03:19:03.4187414332",
        "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": "2024-03-13T03:32:13.7504632042",
            "FinishedAt": "2024-03-13T03:32:13.1453212772"
        },
        "Image": "sha256:170a1e90f8436daa6778aeea3926e716928826c215ca23a8dfd8055f663f9428",
        "ResolvConfPath": "/var/lib/docker/containers/1c447274408314475b093dcbf93163064b819bdd9a
```

```
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
```

Conclusion: Docker revolutionizes the software development and deployment process by providing a powerful platform for containerization. By encapsulating applications and their dependencies into lightweight, portable containers.

NIRJARA SONI
T22-106
TSEC

Experiment 10

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

Theory:
What is a Dockerfile?

A Dockerfile is essentially a text file (literally written as "Dockerfile" with no extension) that contains instructions for creating a Docker image. These instructions are written in the following format:
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:

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 command-line 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
:

```
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.

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 sample-flask-app and version v1. After running this command, you'll see an output similar to this:

```
$ 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
```

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
```

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:

Hello, World!

Conclusion: We have learnt Dockerfile instructions, built an image for a sample web application using DOCKERFILE.