* SEPM ASSIGNMENT :- 1 *

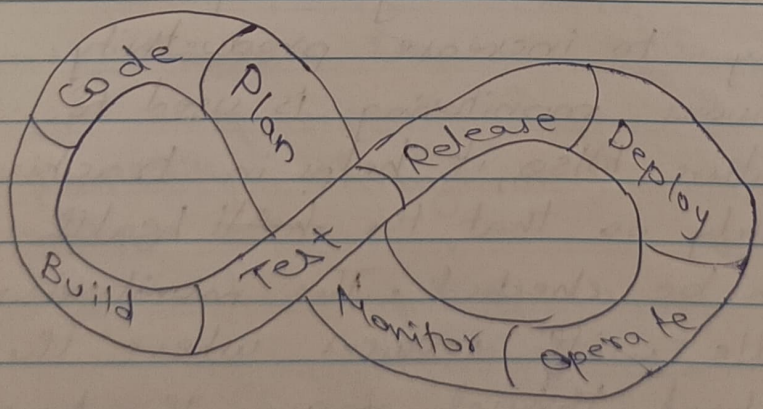
FAIZ SHAIKH T22 93 2201093

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

Theory :- Definition :-

- DevOps is the combination of two words, one is Development and other is Operations. It is a culture to promote the development & operation process collectively.
- DevOps helps to increase organisation speed to deliver applications and services. It also allows organization to serve their customers better and compete more strongly in the market.
- DevOps can also be defined as a sequence of development and IT operations with better communications & collaboration.
- DevOps has become one of the most valuable ~~bris~~ business disciplines for enterprises or organizations. With the help of DevOps, quality & speed of the application delivery has improved to a great extent.

Architecture :-



(i) Build :- Without DevOps, the cost of the consumption of the 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, and the build is independent upon

the user's need, which is a mechanism to control the usage of resources or capacity.

(2) 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 & the expected output, and if necessary reverting to the original code developed.

(3) 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 for production can be reduced as automating the running the scripts will remove many manual steps.

(4) Plan :- DevOps uses Agile methodology to plan the development. With the operations and development team in sync, it helps in organizing the work to plan accordingly to increase productivity.

(5) Monitor :- Continuous monitoring is used to identify any risk of failure. Also, it helps in tracking the system accurately so that the ~~health~~ health of the applications can be checked. The monitoring becomes more comfortable with services where the log data may get monitored through many third-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 and view the optimization scenario, analytics on trends by the deployment

of dashboards.

(7.) Operate :- DevOps changes the traditional approach of developing and testing separately. The teams operate in a collaborative way where both the teams actively participate throughout the service lifecycle. The operation team interacts with developers and they come up with a monitoring plan which serves the IT and business requirements.

(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. Many processes involved in release management commonly used to do the deployment in the production environment manually to lessen the impact on the customers.

Principles

(i) Collaboration.
(ii) Data-based Decision Making.
(iii) Customer-Centric Decision Making
(iv) Constant Improvement.
(v) Responsibility Throughout the Lifecycle.
(vi) Automation
(vii) Failure as a learning Opportunity.

Advantages :-

(i) DevOps is an excellent approach for quick development and deployment of applications.
(ii) It responds faster to the market changes to improve business growth.

(iii) DevOps escalate business profit by decreasing software delivery time and transportation costs.
(iv) DevOps clear the descriptive process, which gives clarity on product development and delivery.
(v) It improves customer experience & satisfaction
(vi) DevOps simplifies collaboration and places all tools in the cloud for customers to access.
(vii) DevOps means collective responsibility, which leads to better team engagement & productivity.

Disadvantages.
(i) DevOps professional or experts developers are less available.
(ii) Developing with DevOps is so expensive.
(iii) Adopting new DevOps technology into the industries is hard to manage in a short time.
(iv) Lack of DevOps knowledge can be problem in the continuous integration of automation projects.

Conclusion :-
Hence, we have known what DevOps is and its advantages & disadvantages.

Experiment: 02

Aim: To understand version control system, install Git and GitHub account

What is Version Control?

Version control is a system that allows developers to track and manage changes to software code over time. It enables collaboration, ensures code integrity, and allows multiple versions of the code to be stored and retrieved as needed. The main goal of version control is to keep track of modifications to a project, making it easier to collaborate, manage different versions of files, and track changes over time.

There are two types of version control systems:

1. **Local Version Control**: This is the simplest form, where a developer keeps track of changes on their own computer.
2. **Distributed Version Control**: This is more advanced and is used by systems like Git, where each developer has their own local copy of the entire project repository (including its history), and changes are synchronized with others.

What is Git?

Git is a distributed version control system created by Linus Torvalds (the creator of Linux). Git helps developers manage the source code history by tracking changes and enabling multiple developers to work on a project without stepping on each other's toes. With Git, you can:

- **Track changes**: See what was modified and by whom.
- **Branching and merging**: Work on different parts of a project in parallel, then merge those parts back together.
- **Collaboration**: Work with others by pushing and pulling changes from remote repositories.

How Git Works:

1. **Repository (repo)**: A directory or storage space where Git keeps all the files, history, and versions of a project.
2. **Commit**: A snapshot of your project at a particular point in time. It records changes made to the project files.
3. **Branch**: A parallel version of the repository. You can create a new branch to work on a feature without affecting the main project.

4. **Merge**: The process of combining changes from different branches back into the main branch.

What is GitHub?

GitHub is a web-based platform that hosts Git repositories. It provides a user interface for managing Git repositories, collaborating with others, and sharing your code. GitHub makes it easier for developers to work together on a project, track bugs, and manage project releases. Key features of GitHub:

- **Remote repositories**: You can upload your local Git repositories to GitHub to back them up or collaborate.
- **Pull requests**: A way of proposing changes to a repository. A developer can submit a pull request to request that their changes be merged into another branch or the main codebase.
- **Issues and projects**: Track bugs, feature requests, and manage the workflow of development using tools integrated into GitHub.
- **Collaboration**: GitHub makes it easy for multiple people to work on a project by allowing them to push, pull, and merge changes from others.

Installing Git and Setting Up GitHub Account

Step 1: Install Git

1. **Download Git** from Git's official website.
2. **Install Git** by following the installation prompts specific to your operating system (Windows, Mac, or Linux).
    - On Windows, during installation, it is recommended to choose the default options.
    - On Mac/Linux, you can install Git using package managers like Homebrew (Mac) or apt (Linux).
3. Once Git is installed, you can verify it by opening a terminal or command prompt and running:

bash Copy

git --version

This will display the version of Git installed.

Step 2: Set Up Git

Before you start using Git, you should configure your identity:

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

config --global user.email "your-email@example.com"

This ensures that your commits are properly attributed to you.

Step 3: Create a GitHub Account

1. Go to GitHub and sign up for an account.
2. After signing up, you'll be able to create repositories and start collaborating with others.
3. You can create a new repository by clicking the **New** button on your dashboard or through the "Repositories" tab.

Step 4: Link Git with GitHub (SSH Keys)

To allow Git to communicate with GitHub, you need to set up SSH keys for authentication (instead of using your password every time).

1. **Generate an SSH key**:

bash Copy ssh-keygen -t rsa -b 4096 -C "your-

email@example.com" This will generate a key pair

(public and private keys).

2. **Add the SSH key to your GitHub account**:
    - Copy the public key using:

bash Copy

cat ~/.ssh/id_rsa.pub

    - Go to **GitHub** > **Settings** > **SSH and GPG Keys** > **New SSH Key**, then paste the key into the field provided.

3. **Test the connection**:

bash

Copy ssh -T

git@github.com

If successful, GitHub will confirm the connection.

Step 5: Clone a GitHub Repository

To get a project from GitHub onto your local machine, you can **clone** the repository. In your terminal, run:

bash Copy

git clone https://github.com/username/repository-name.git or,

if using SSH:

bash Copy git clone

git@github.com:username/repository-name.git

Step 6: Basic Git Commands

- **Check the status of your repository**:

bash Copy

git status

- **Add changes** to the staging area:

bash Copy

git add .

- **Commit changes** to your local repository:

bash Copy git commit -m "Your

commit message"

- **Push changes** to GitHub:

bash

Copy

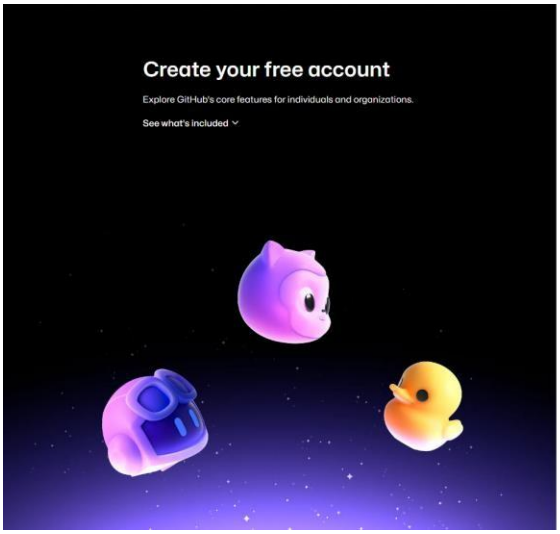
git push origin main

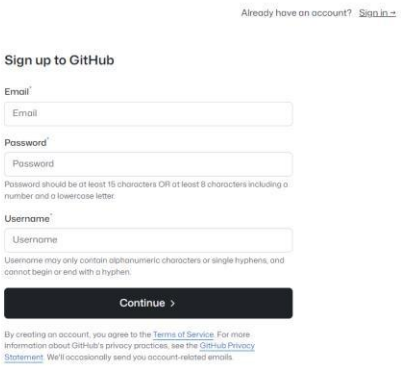
- **Pull changes** from GitHub to your local repository:

bash Copy

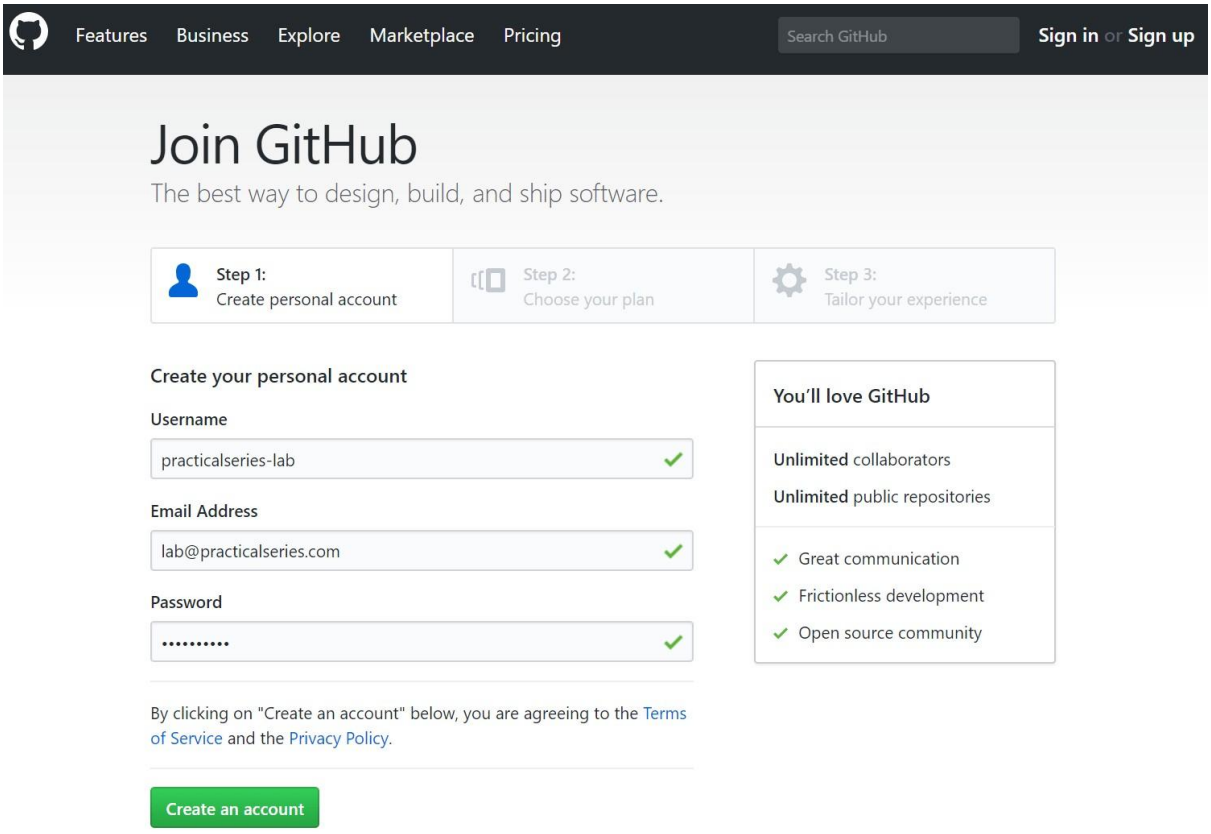
git pull origin main

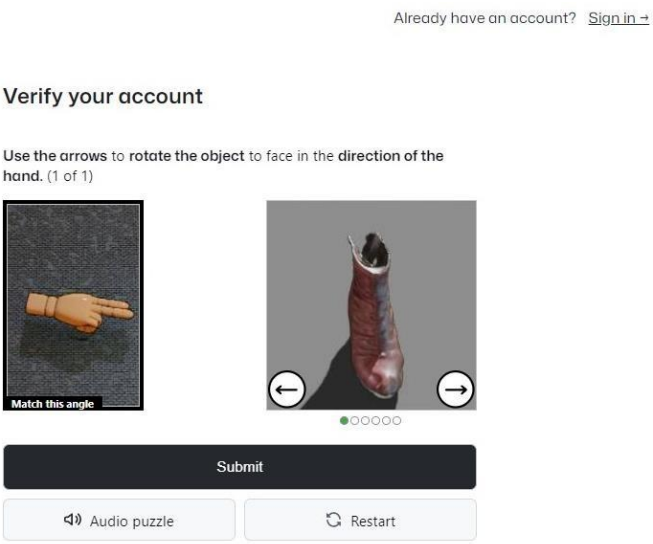
Step 1: Go to https://github.com/join in a web browser.



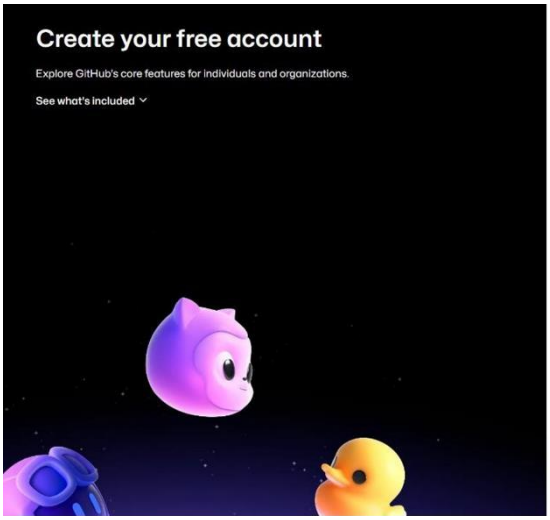


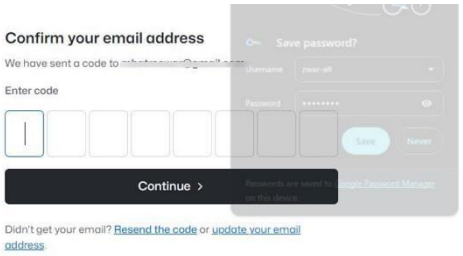
Step 2: Personal Information

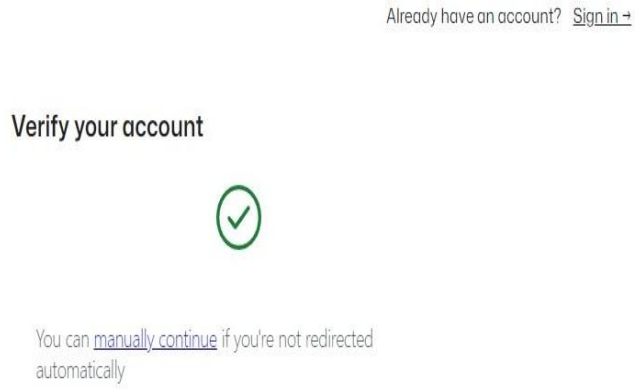




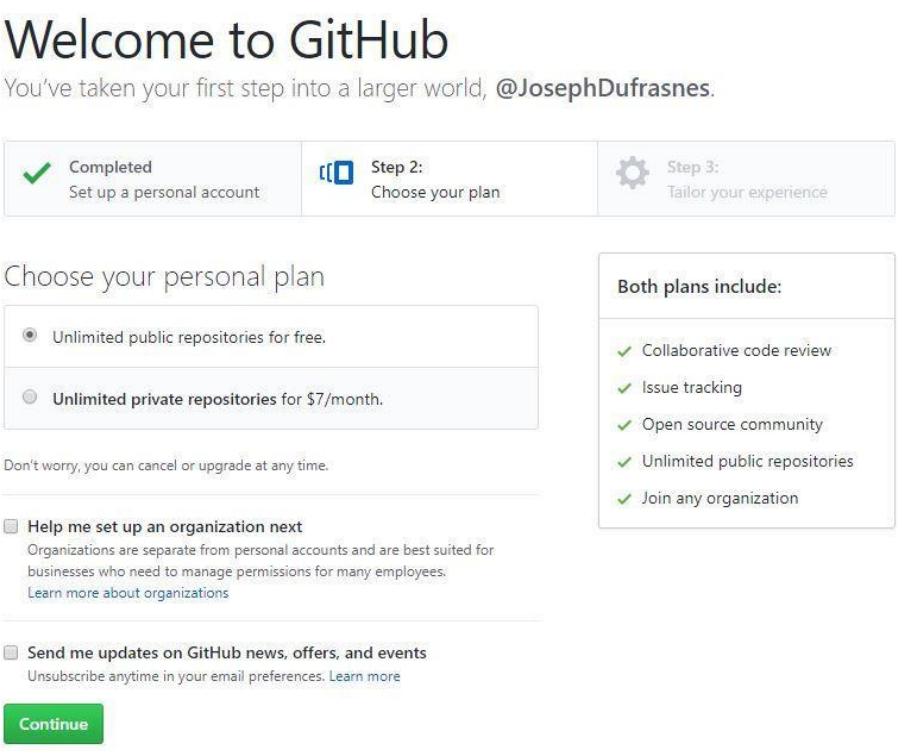
Step 3:Verify Email By Code

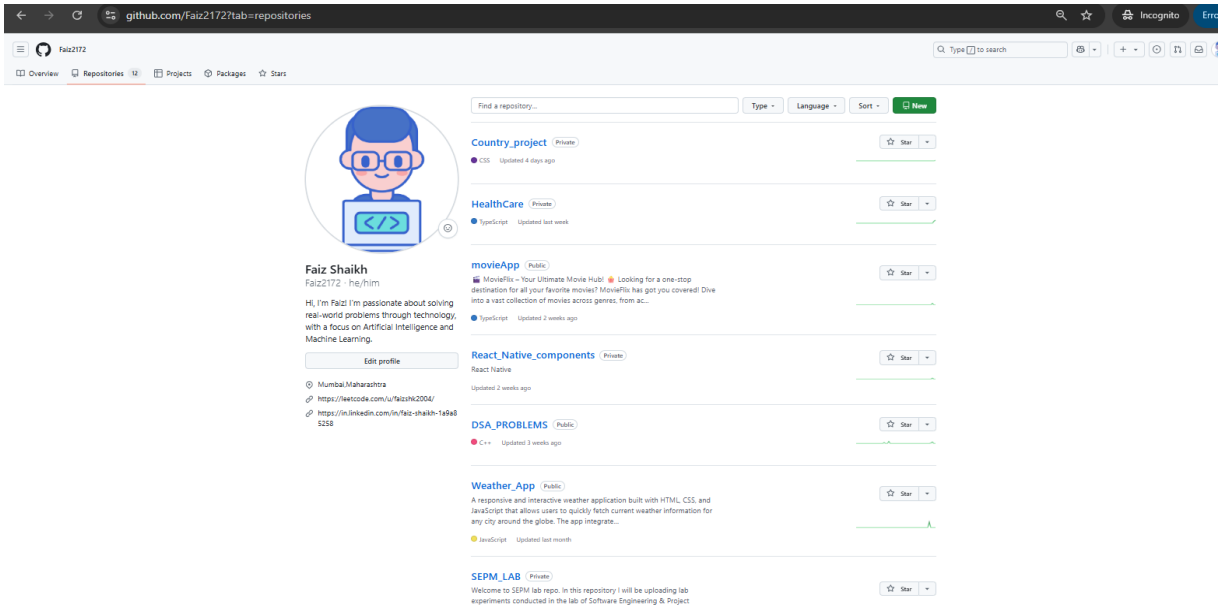






Step 4: Select your preference and Free plan





Conclusion:

In summary, Git is a powerful version control system that helps you track changes, collaborate with others, and maintain the integrity of your codebase. GitHub is an online platform that makes it easy to host and share Git repositories, enabling collaboration and project management. Setting up Git and GitHub allows you to contribute to open-source projects, manage your own code, and work effectively with others in a development environment.

FAIZ SHAIKH
T22 – 2201093

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>

- 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

- 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:

```
C:\Users\Lab805_10>cd desktop

C:\Users\Lab805_10\Desktop>mkdir git-dvcs

C:\Users\Lab805_10\Desktop>git config --global user.name "FaizShaikh"

C:\Users\Lab805_10\Desktop>git config --global user.name faizshaikh29086@gmail.com

C:\Users\Lab805_10\Desktop>git config --global --list
user.name=faizshaikh29086@gmail.com
user.email=thakurparitosh22@gmail.com

C:\Users\Lab805_10\Desktop>cd git-dvcs

C:\Users\Lab805_10\Desktop\git-dvcs>git config --global user.name "FaizShaikh"

C:\Users\Lab805_10\Desktop\git-dvcs>git config --global user.name faizshaikh29086@gmail.com

C:\Users\Lab805_10\Desktop\git-dvcs>git config --global --list
user.name=faizshaikh29086@gmail.com
user.email=thakurparitosh22@gmail.com

C:\Users\Lab805_10\Desktop\git-dvcs>cat~/.gitconfig
'cat~' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\Lab805_10\Desktop\git-dvcs>mkdir git-demo-project93

C:\Users\Lab805_10\Desktop\git-dvcs>cd git-demo-project93

C:\Users\Lab805_10\Desktop\git-dvcs\git-demo-project93>git init
Initialized empty Git repository in C:/Users/Lab805_10/Desktop/git-dvcs/git-demo-project93/.git/

C:\Users\Lab805_10\Desktop\git-dvcs\git-demo-project93>Is -a
'Is' is not recognized as an internal or external command,
operable program or batch file.
```

```
Lab805_10@805-20 MINGW64 ~
$ cd desktop

Lab805_10@805-20 MINGW64 ~/desktop (main)
$ mkdir git-dcvs93

Lab805_10@805-20 MINGW64 ~/desktop (main)
$ cd git-dcvs93

Lab805_10@805-20 MINGW64 ~/desktop/git-dcvs93 (main)
$ git config --global user.name FaizShaikh

Lab805_10@805-20 MINGW64 ~/desktop/git-dcvs93 (main)
$ git config --global user.email faizshaikh29086@gmail.com

Lab805_10@805-20 MINGW64 ~/desktop/git-dcvs93 (main)
$ git config --global --list
user.name=FaizShaikh
user.email=faizshaikh29086@gmail.com

Lab805_10@805-20 MINGW64 ~/desktop/git-dcvs93 (main)
$ cat ~/.gitconfig
[user]
        name = FaizShaikh
        email = faizshaikh29086@gmail.com

Lab805_10@805-20 MINGW64 ~/desktop/git-dcvs93 (main)
$ mkdir git-my-project93

Lab805_10@805-20 MINGW64 ~/desktop/git-dcvs93 (main)
$ cd git-my-project93

Lab805_10@805-20 MINGW64 ~/desktop/git-dcvs93/git-my-project93 (main)
$ git init
Initialized empty Git repository in C:/Users/Lab805_10/Desktop/git-dcvs93/git-my
-project93/.git/

Lab805_10@805-20 MINGW64 ~/desktop/git-dcvs93/git-my-project93 (master)
$ ls -a
./  ../  .git/
```

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.

Experiment 4 : To understand Continuous Integration, install and configure Jenkins with Maven/Ant/Gradle to set up a build job.

Aim : To understand the concept of **Continuous Integration (CI)** and implement it by installing and configuring **Jenkins** with **Maven, Ant, or Gradle** to automate the build process. This study aims to explore how Jenkins helps in setting up a CI pipeline, executing automated builds, and improving software development efficiency.

Theory :

Theory of Continuous Integration Using Jenkins with Maven, Ant, or Gradle

Introduction to Continuous Integration (CI)

Continuous Integration (CI) is a **software development practice** where developers frequently integrate their code changes into a shared repository. Each integration is verified using **automated builds and tests**, ensuring that issues are detected early. CI helps streamline the development process, reduces manual errors, and improves software quality.

Key principles of CI:

1. **Frequent Code Integration** – Developers merge changes multiple times a day.
2. **Automated Build Process** – Code is compiled, built, and tested automatically.
3. **Immediate Feedback** – Issues are detected early and fixed promptly.
4. **Consistent Environment** – CI ensures that software builds are reproducible across different environments.

To implement Continuous Integration, organizations use **CI tools like Jenkins**, which automates the build, test, and deployment process.

Jenkins: A CI/CD Automation Tool

Jenkins is an open-source **automation server** that enables developers to **automate software builds, tests, and deployments**. It supports integration with version control systems (Git, SVN) and build tools like **Maven, Ant, and Gradle**.

Key Features of Jenkins

- **Automated Builds:** Supports scheduled or triggered builds based on repository

changes.

- **Build Pipelines:** Allows chaining multiple jobs for end-to-end automation.
- **Plugin Support:** Offers 1,500+ plugins for integration with tools like Docker, Kubernetes, and Slack.
- **Scalability:** Can distribute builds across multiple nodes for faster execution.

Build Tools: Maven, Ant, and Gradle

Build tools are essential in CI to **compile source code, resolve dependencies, and generate deployable artifacts**.

1. Apache Maven

- A widely used **Java-based build automation tool**.
- Uses **POM.xml (Project Object Model)** to define project dependencies, build lifecycle, and plugins.
- Supports phases like **clean, compile, test, package, install, and deploy**.
- Command to build a project:
- mvn clean install

2. Apache Ant

- **Older than Maven**, but still used for Java builds.
- Uses an **XML-based build script (build.xml)** to define tasks.
- More flexible but requires explicit configurations. •

Command to execute a build:

- ant build

3. Gradle

- **Newer build tool**, used for **Java, Kotlin, and Android development**.
- Uses a **Groovy or Kotlin-based build script** instead of XML.
- Faster than Maven due to its **incremental build mechanism**.
- Command to build a project:
- gradle build

Jenkins Integration with Maven, Ant, and Gradle

Jenkins can be configured to **automate builds** using these tools. The integration process involves:

1. **Installing Jenkins** and setting up build tools.

2. **Creating a job in Jenkins** that fetches source code from Git.

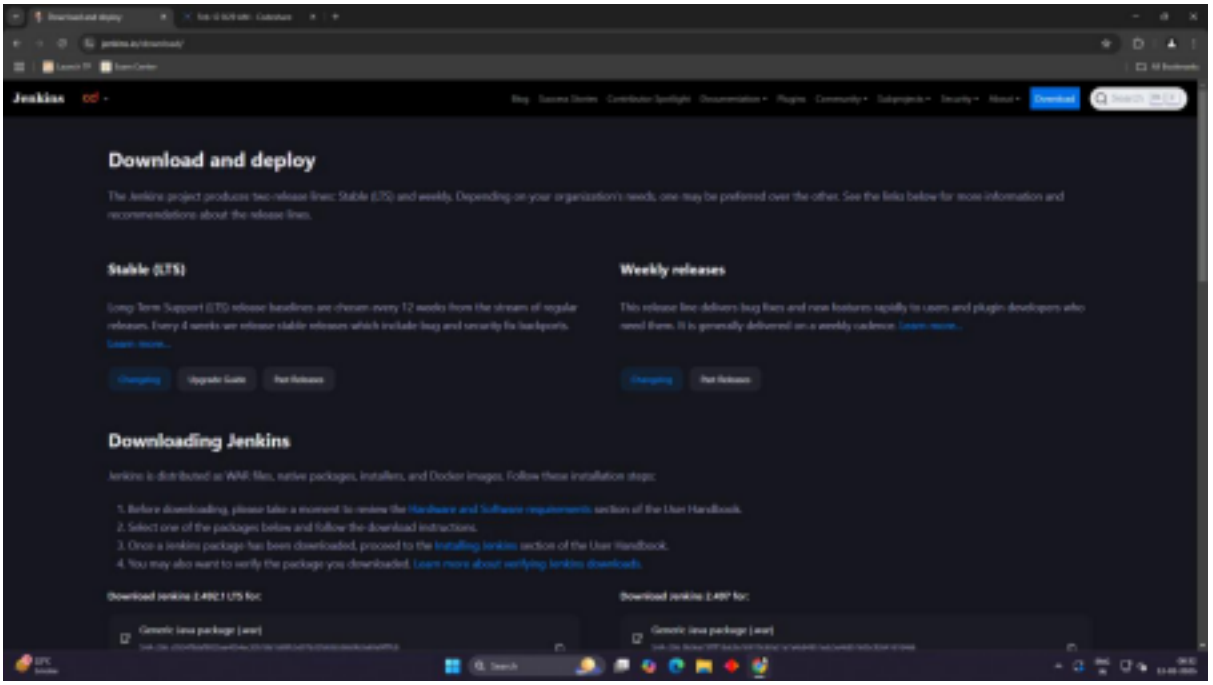
3. **Configuring build steps** to invoke Maven, Ant, or Gradle commands.

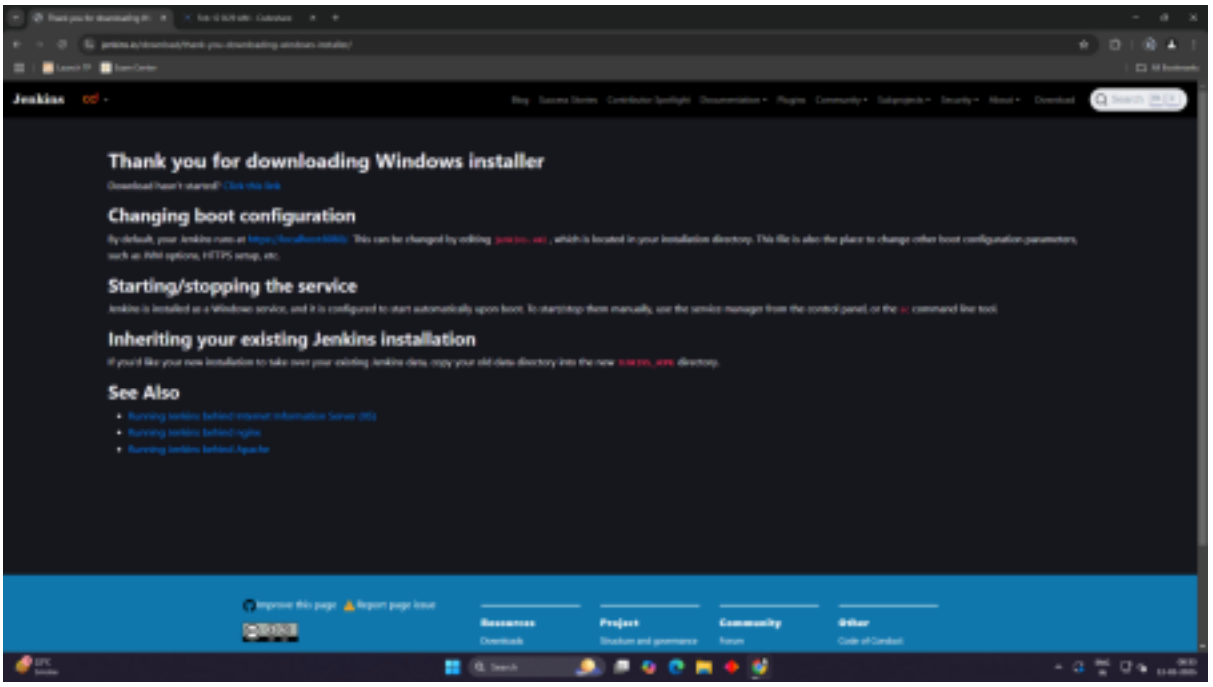
4. **Executing automated builds** and monitoring results.

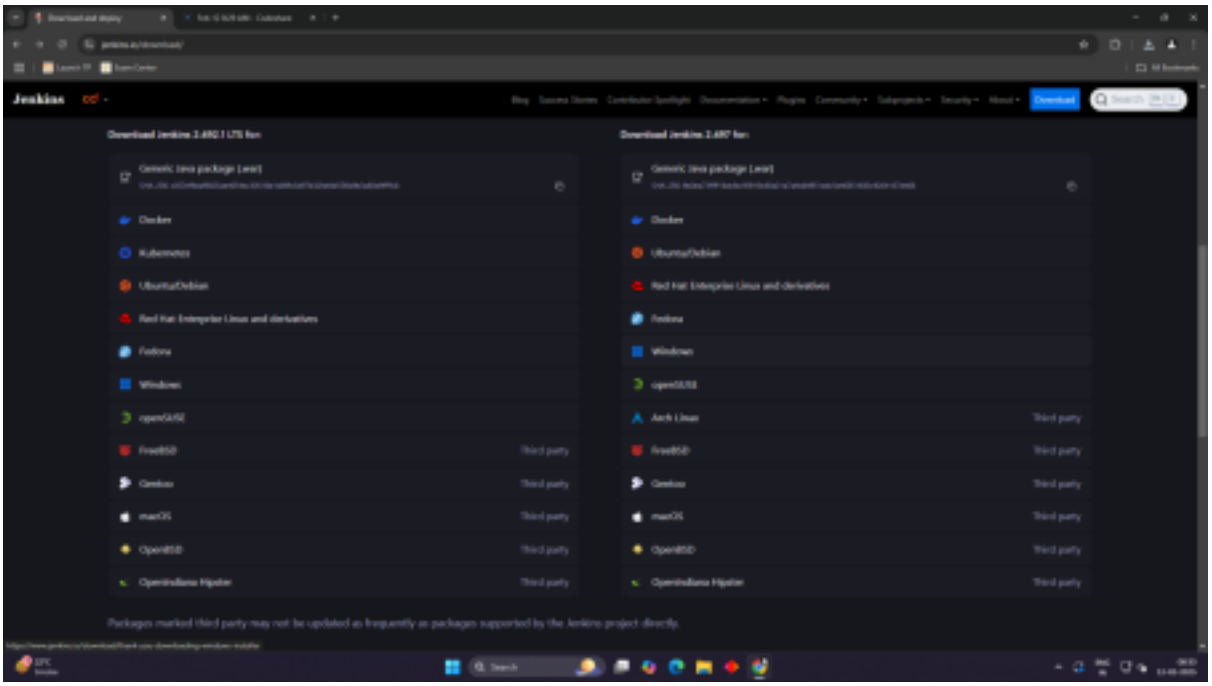
Advantages of Using Jenkins for CI

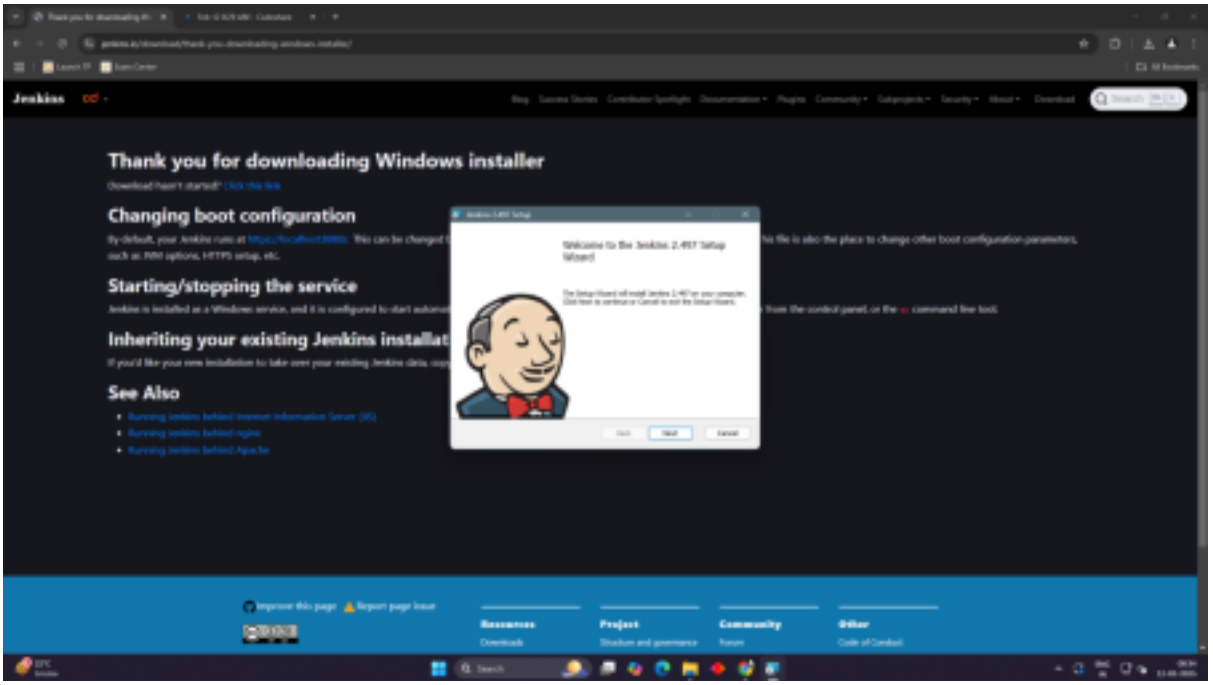
- **Faster Development Cycle** – Automated builds and testing reduce manual effort.
- **Early Bug Detection** – Continuous integration ensures quick issue identification.
- **Improved Collaboration** – Developers work on the latest stable codebase.
- **Efficient Deployment** – Jenkins supports integration with **Docker, Kubernetes, and cloud platforms**.

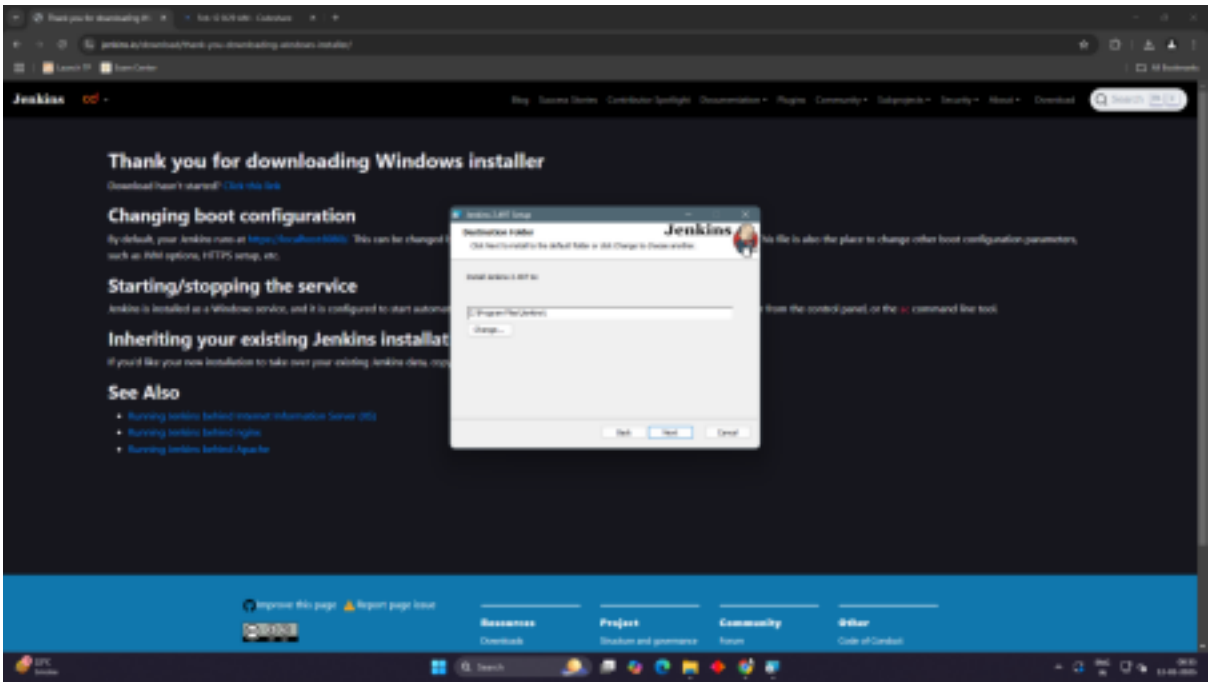
Implementation :

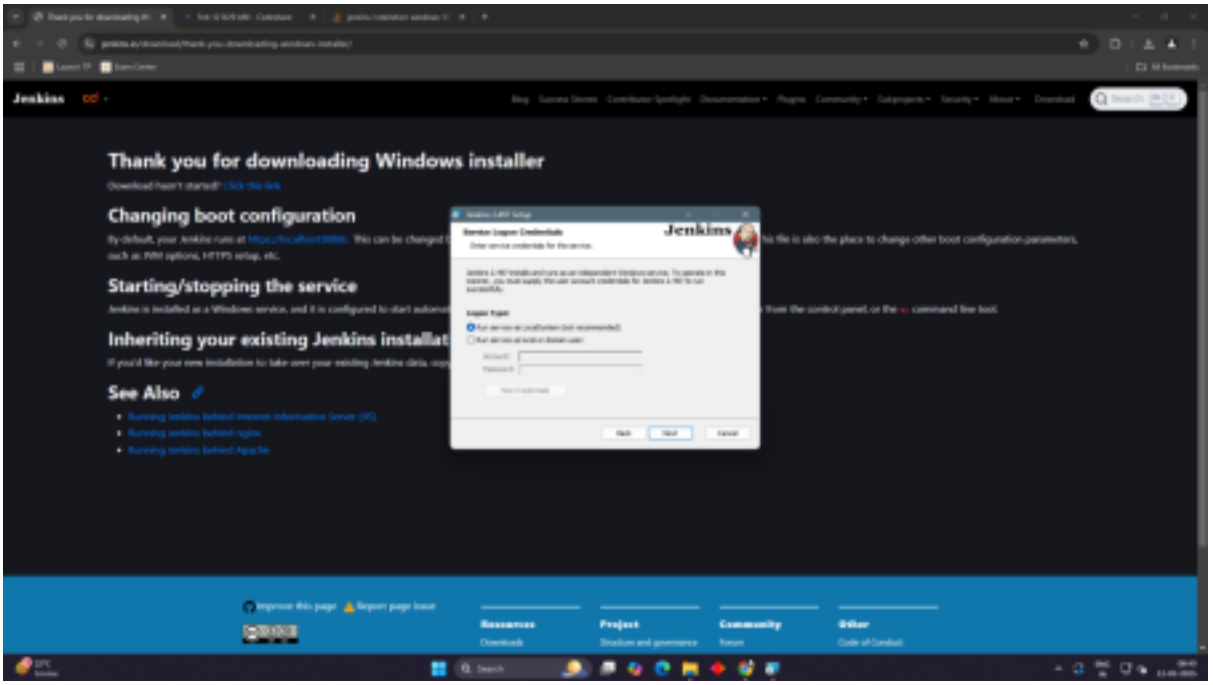

Download and deploy
Stable (LTS)
Weekly releases
Downloading Jenkins

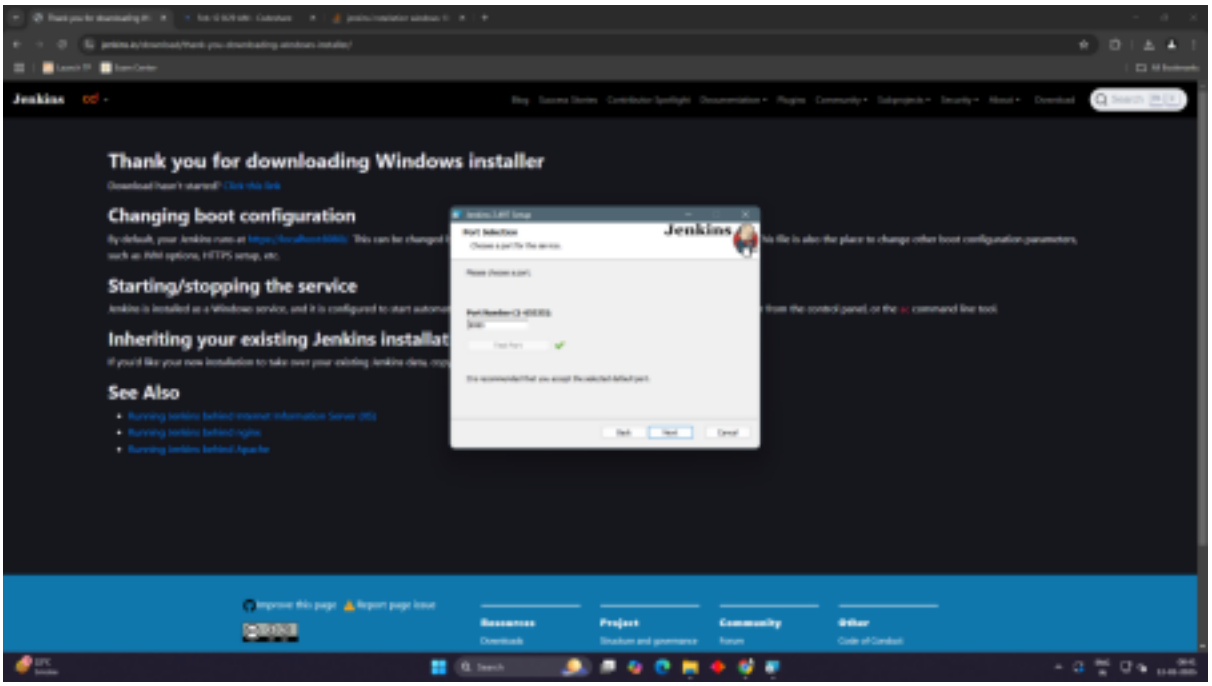

Thank you for downloading Windows installer
Changing boot configuration
Starting/stopping the service
Inheriting your existing Jenkins installation
See Also

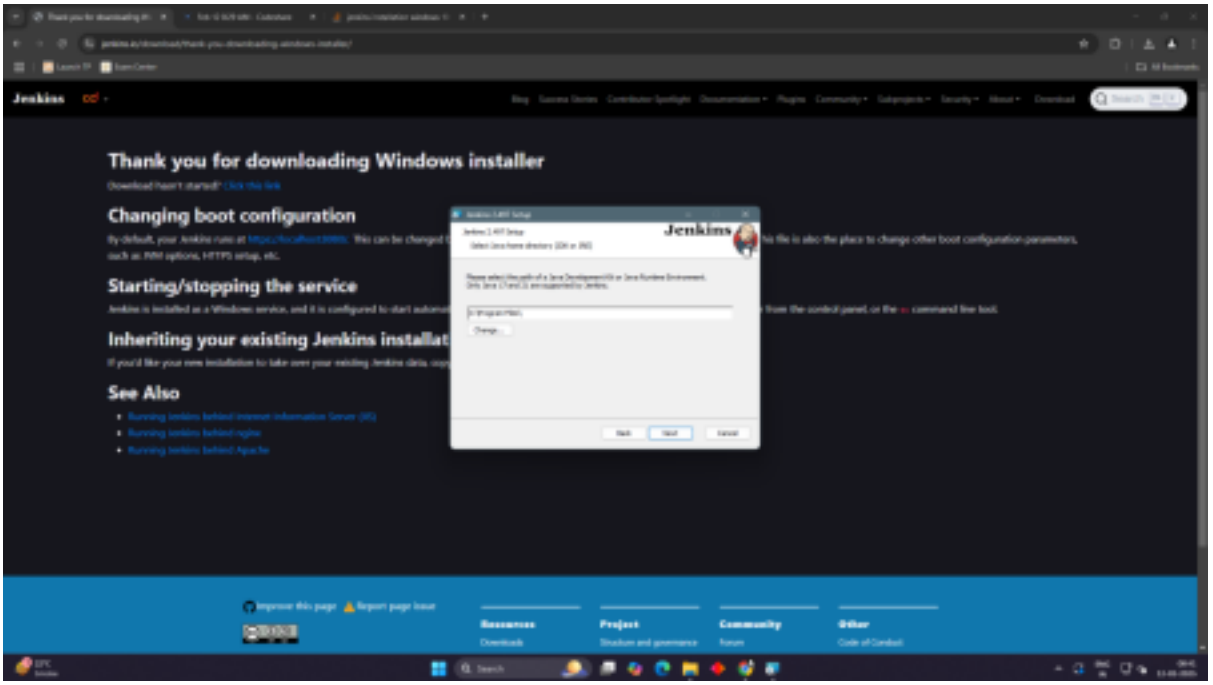


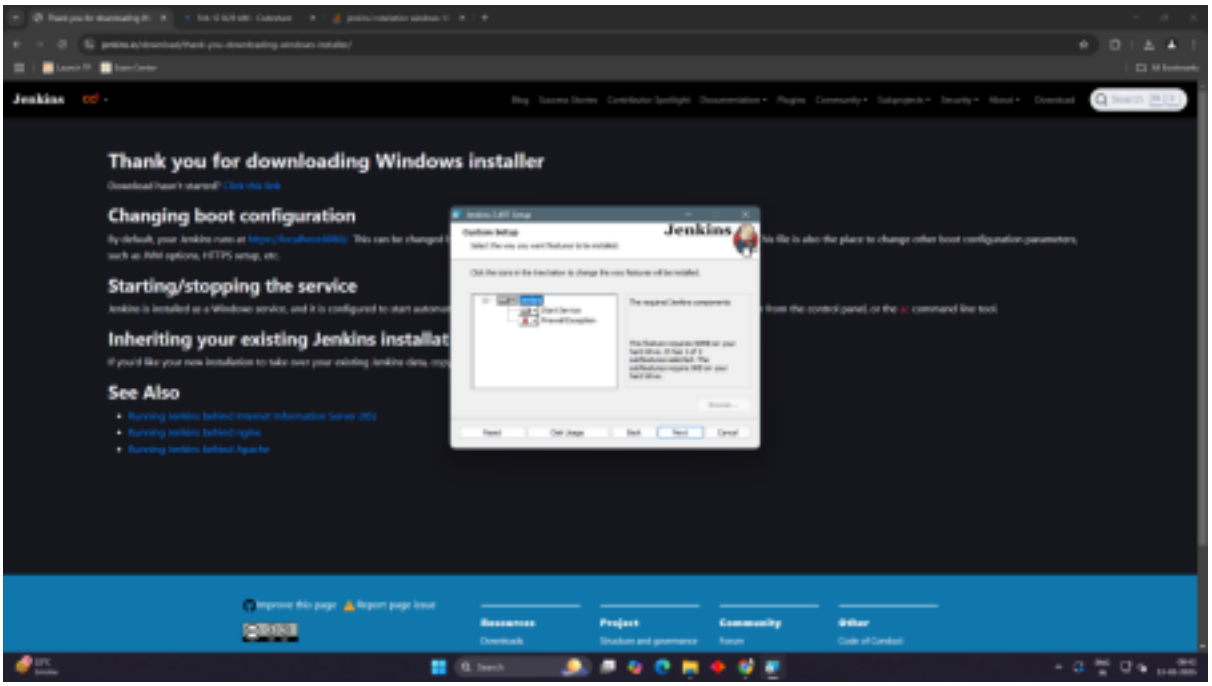

Thank you for downloading Windows installer
Changing boot configuration
Starting/stopping the service
Inheriting your existing Jenkins installat
See Also

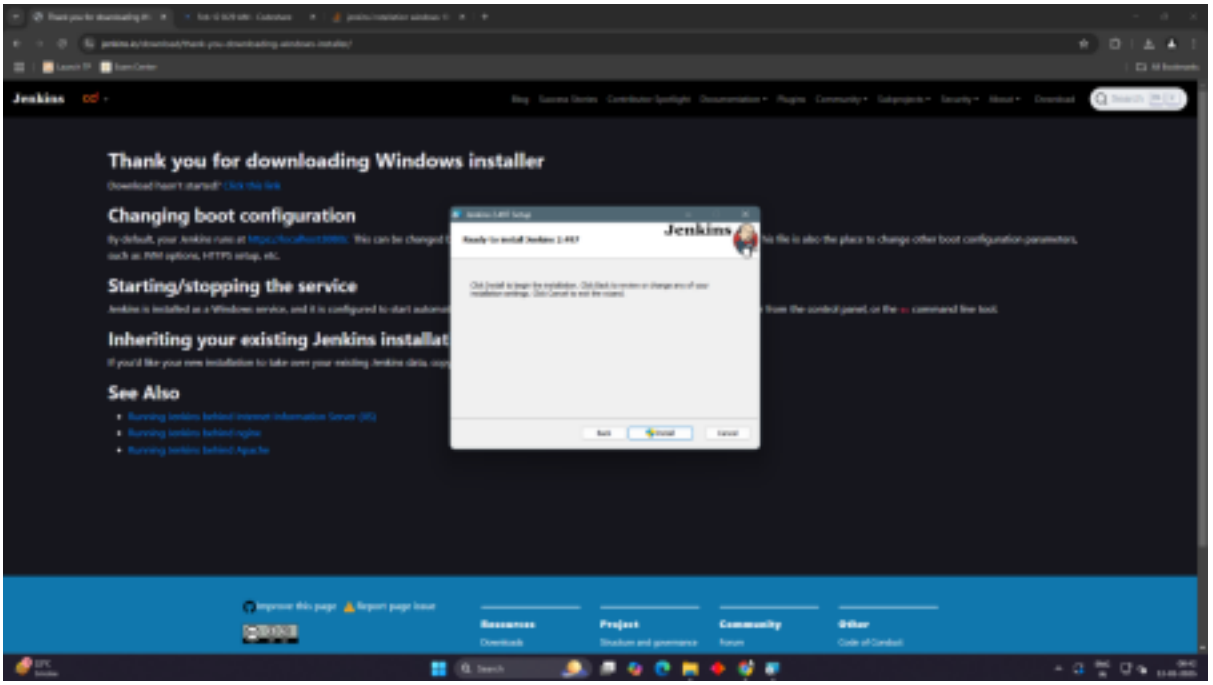

Thank you for downloading Windows installer
Changing boot configuration
Starting/stopping the service
Inheriting your existing Jenkins installat
See Also
Jenkins

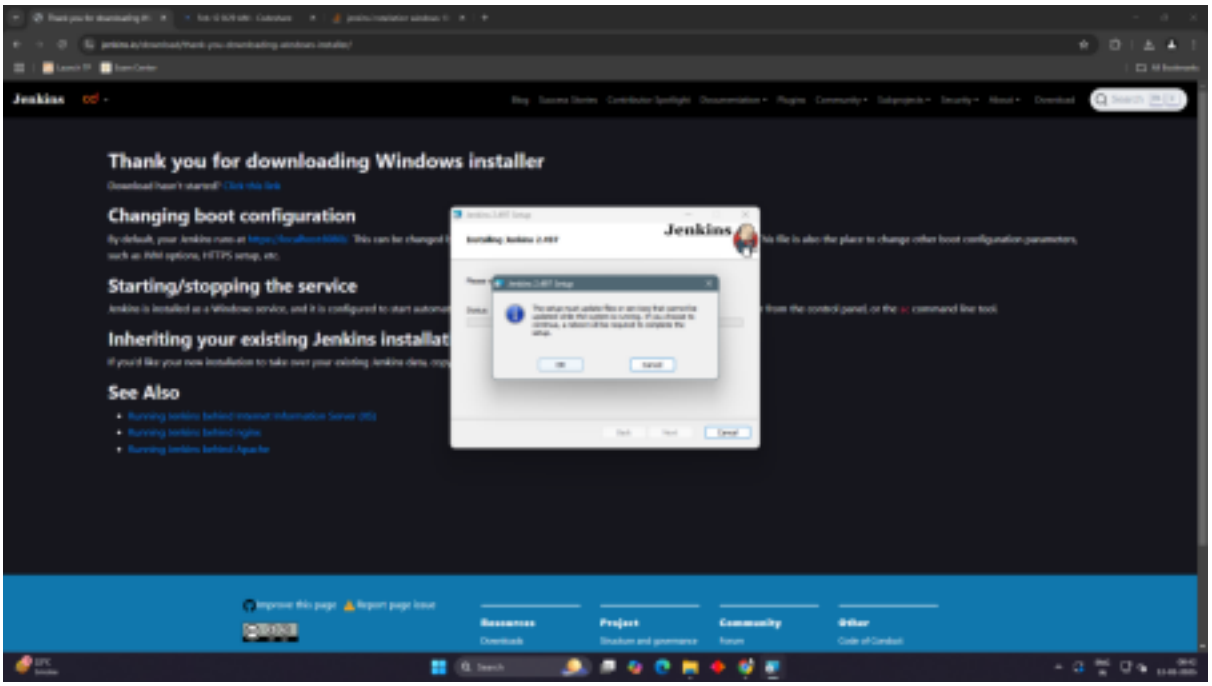

Thank you for downloading Windows installer
Changing boot configuration
Starting/stopping the service
Inheriting your existing Jenkins installat
See Also
Jenkins

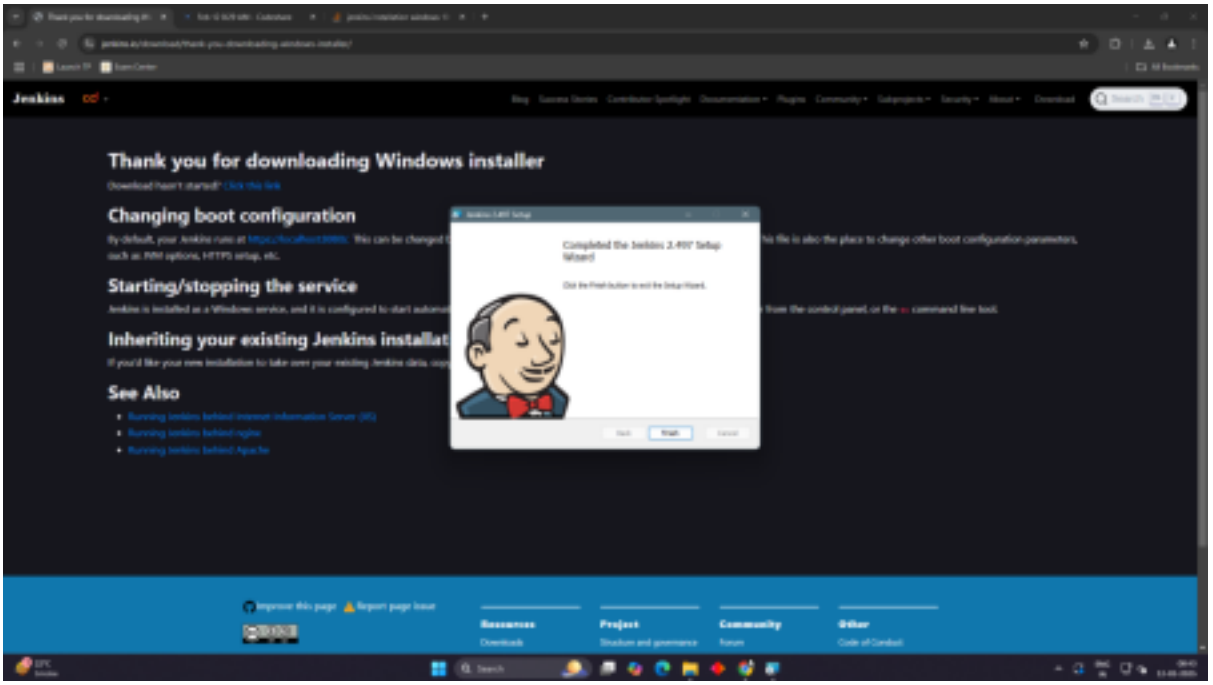

Thank you for downloading Windows installer
Changing boot configuration
Starting/stopping the service
Inheriting your existing Jenkins installat
See Also
Jenkins

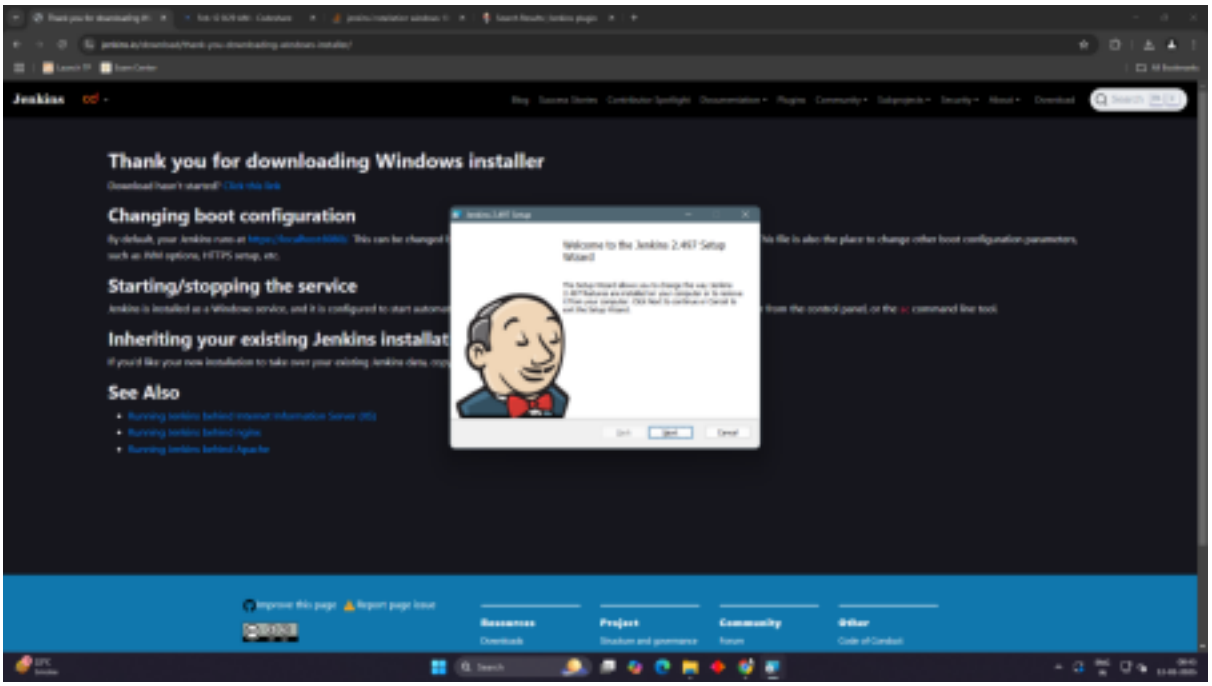

Thank you for downloading Windows installer
Changing boot configuration
Starting/stopping the service
Inheriting your existing Jenkins installat
See Also
Jenkins

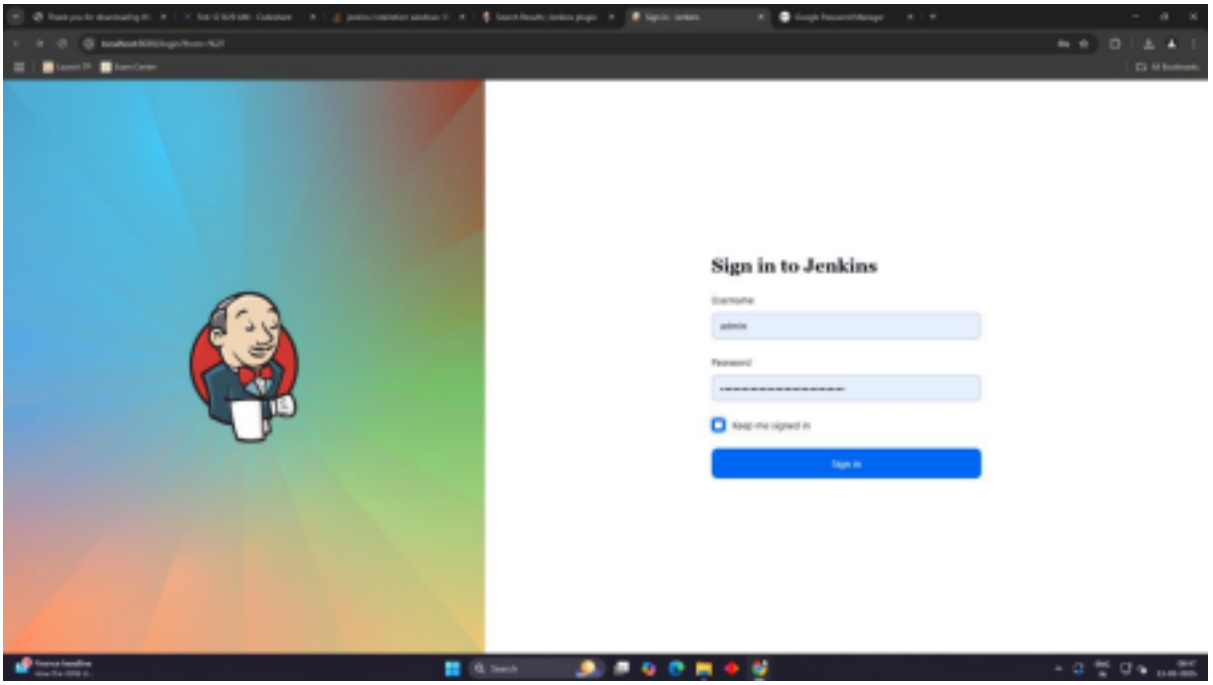

Thank you for downloading Windows installer
Changing boot configuration
Starting/stopping the service
Inheriting your existing Jenkins installat
See Also
Jenkins

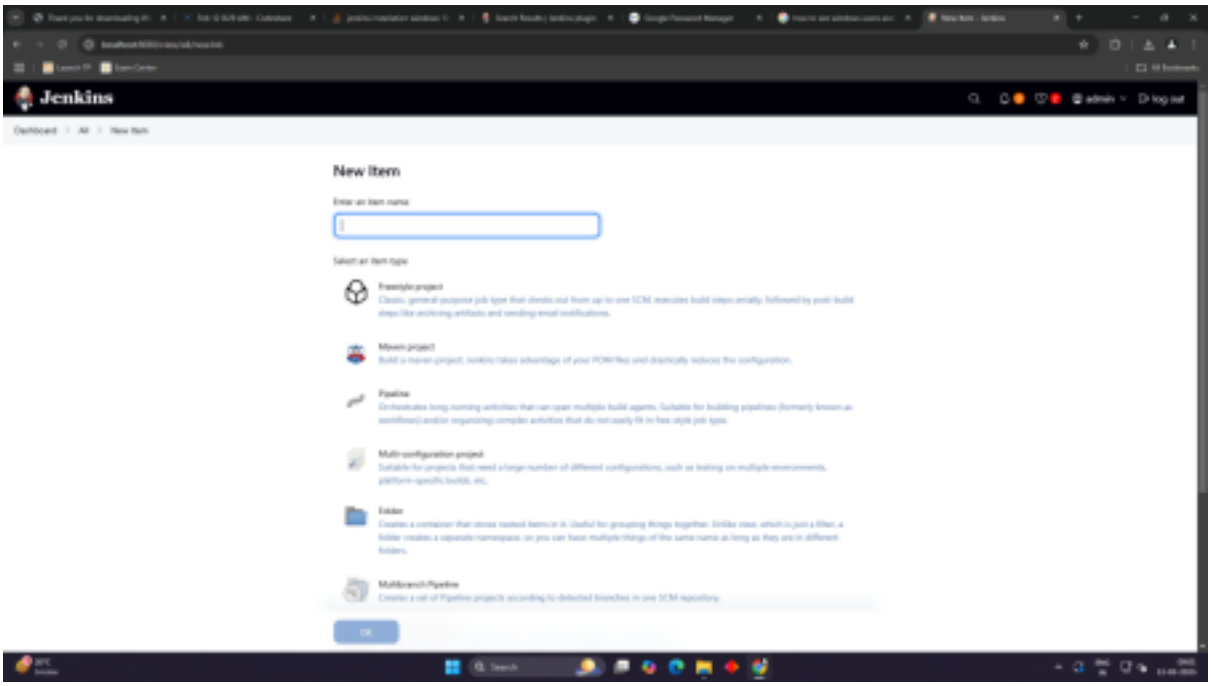

Thank you for downloading Windows installer
Changing boot configuration
Starting/stopping the service
Inheriting your existing Jenkins installat
See Also
Jenkins


Thank you for downloading Windows installer
Changing boot configuration
Starting/stopping the service
Inheriting your existing Jenkins installat
See Also
Jenkins


Thank you for downloading Windows installer
Changing boot configuration
Starting/stopping the service
Inheriting your existing Jenkins installat
See Also


Thank you for downloading Windows installer
Changing boot configuration
Starting/stopping the service
Inheriting your existing Jenkins installat
See Also


Sign in to Jenkins



Conclusion : Thus we have successfully installed and configured Jenkins.

Name: Faiz Shaikh
Roll No. 93

Batch T22

Experiment No-5

Aim: 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

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.

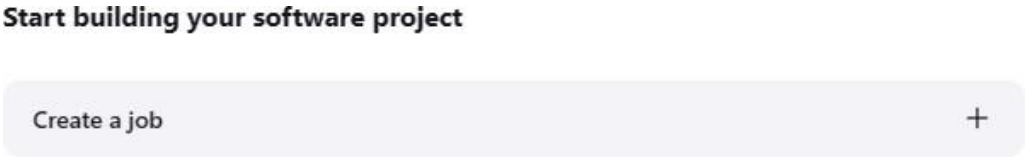
Jenkins is a self-contained, open-source automation server which can be used to automate all sorts of tasks related to building, testing, and delivering or deploying software.

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

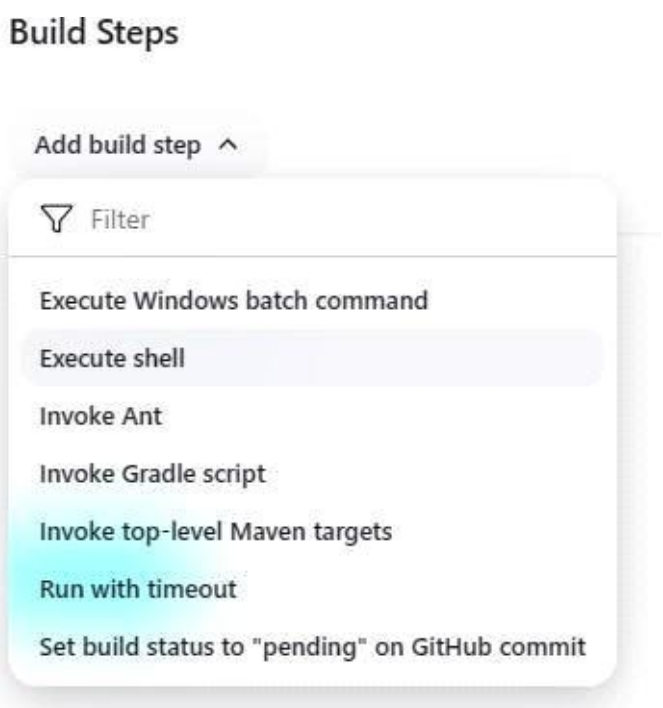
Example 1.1: Deploying a freestyle app in Jenkins

Creating a job:



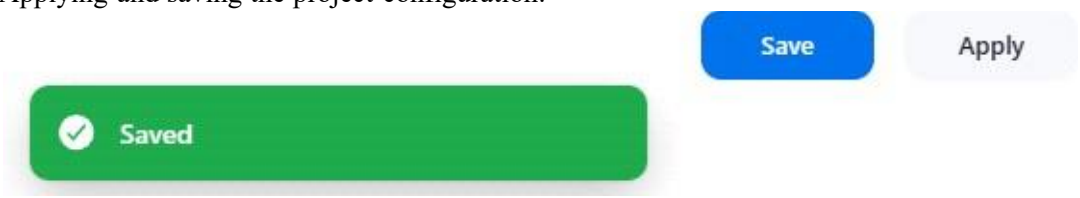
Naming the job and setting it as freestyle:

Selecting build type as “Execute shell”:

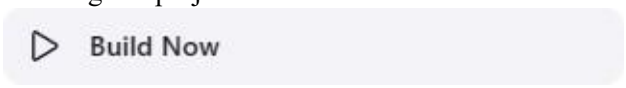


Entering a simple command for the shell execution:

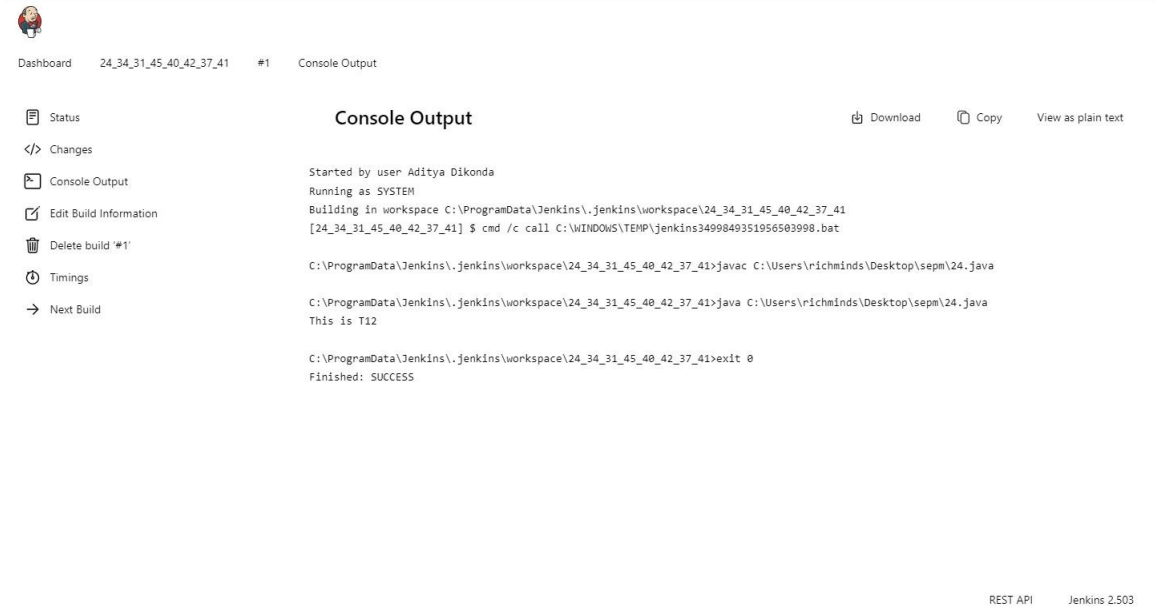
Applying and saving the project configuration:

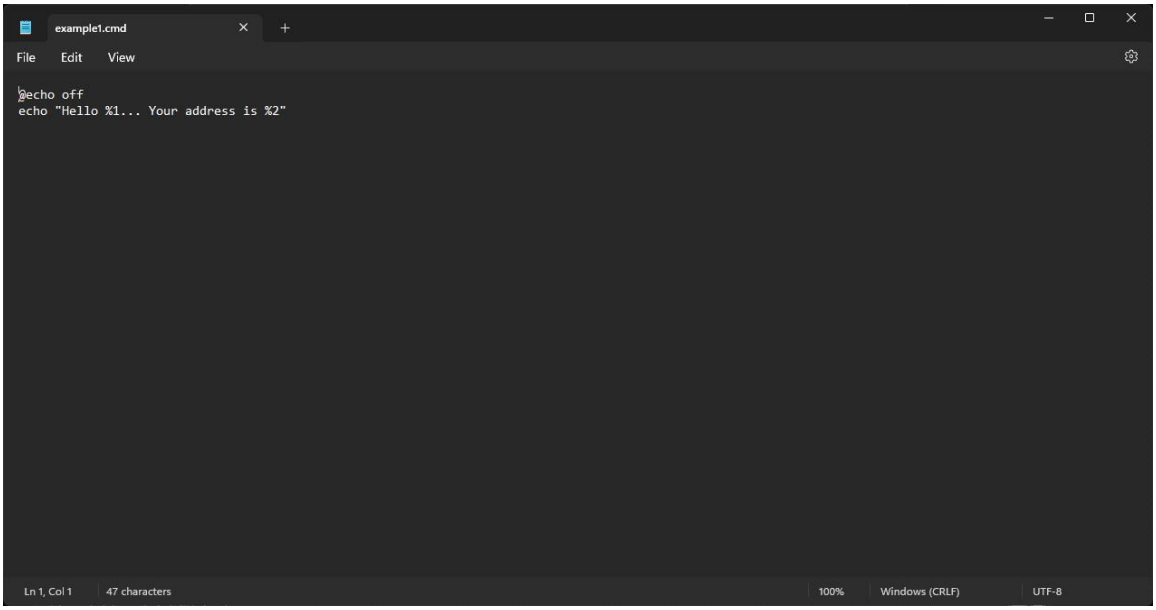


Building the project:



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:

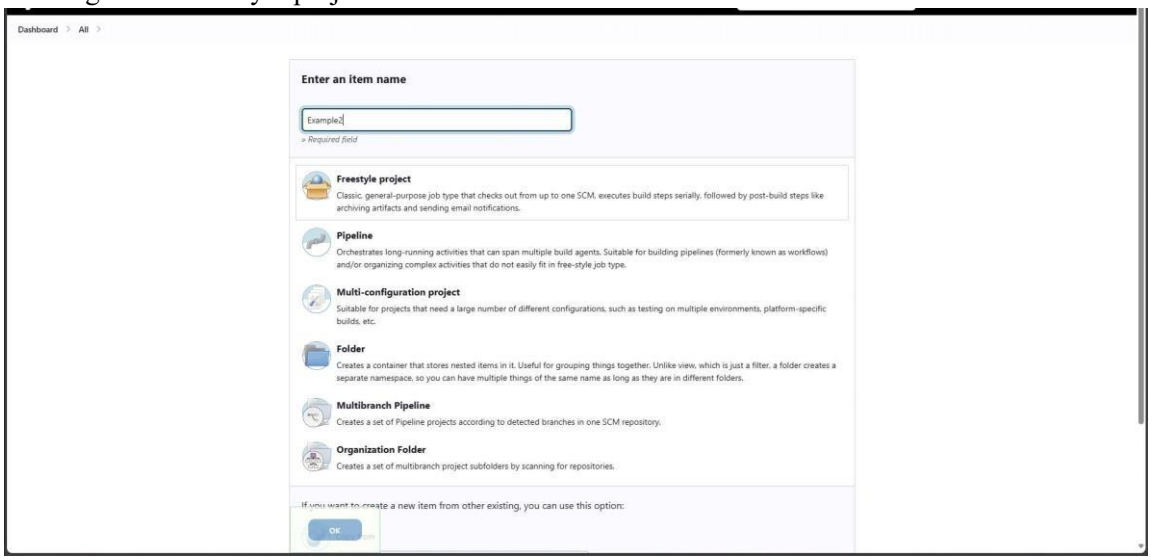
: Running a Java program under Jenkins

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

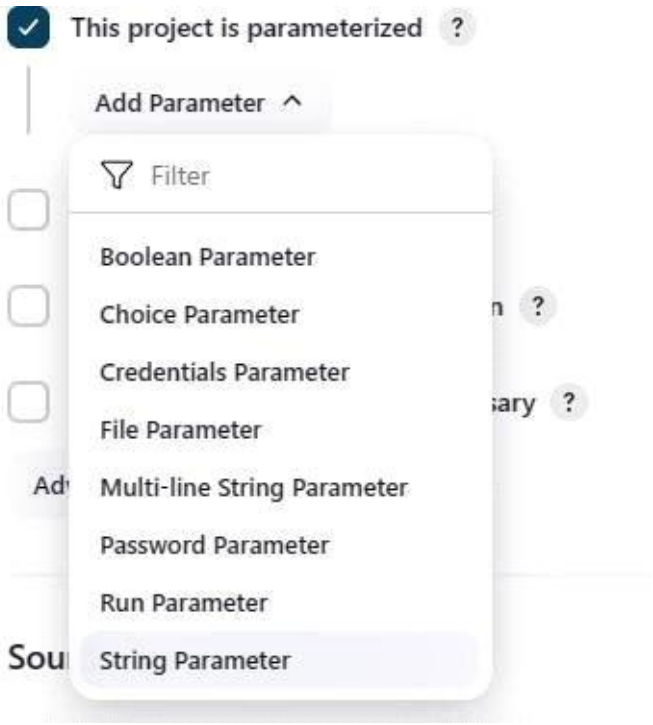
Download Copy View as plain text

```
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:

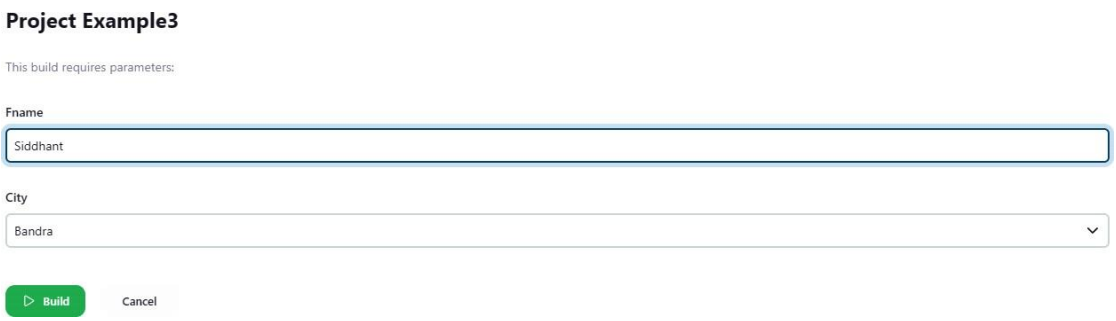


Configuring the string parameter as `Fname`:

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

Configuring build steps:

Entering parameters for build:

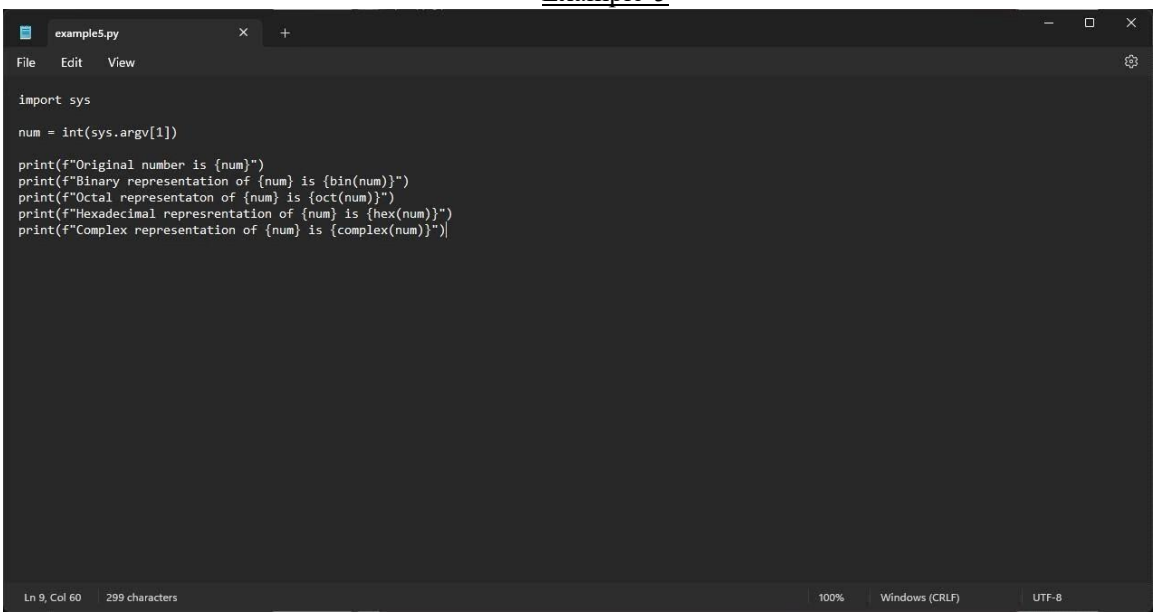


Console output after building:

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
```

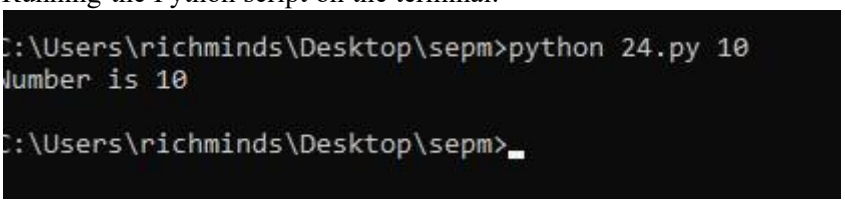
Example 5



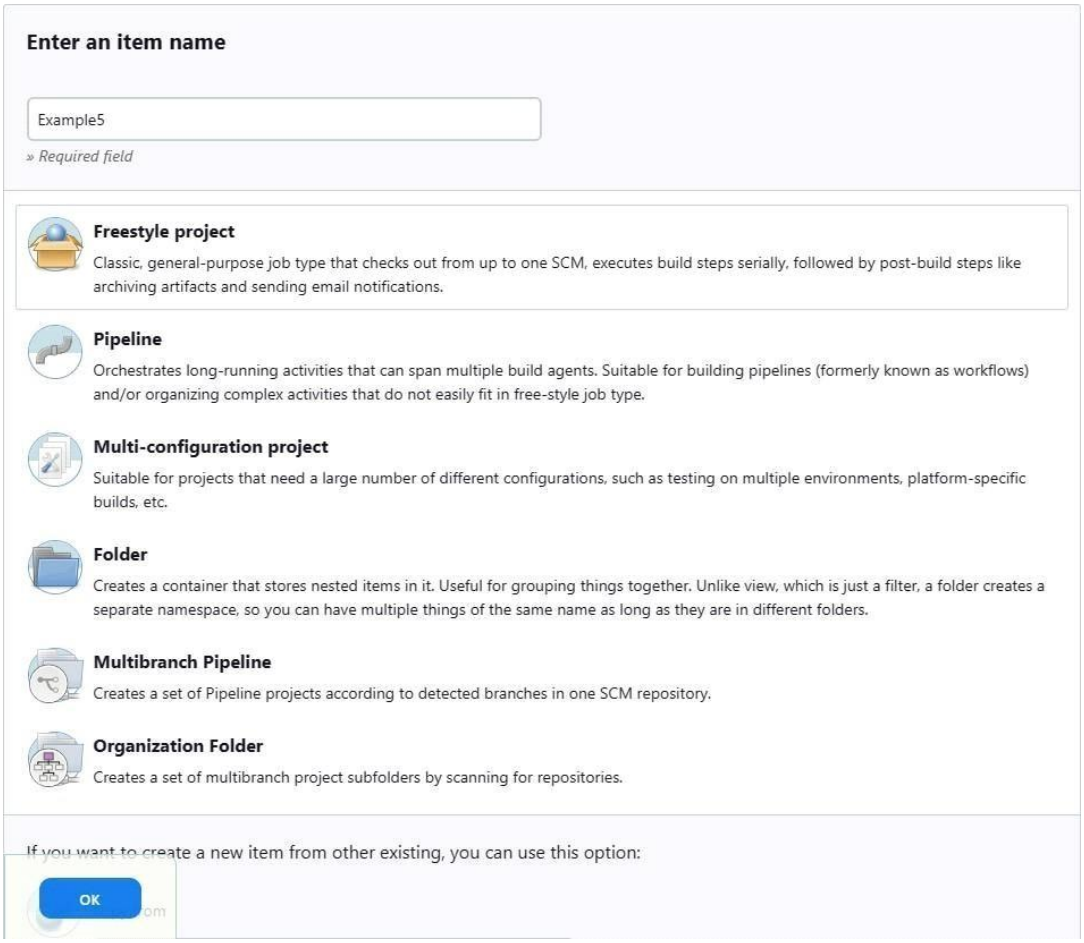
Example 5.1: Running a Python program Creating

a simple Python script:

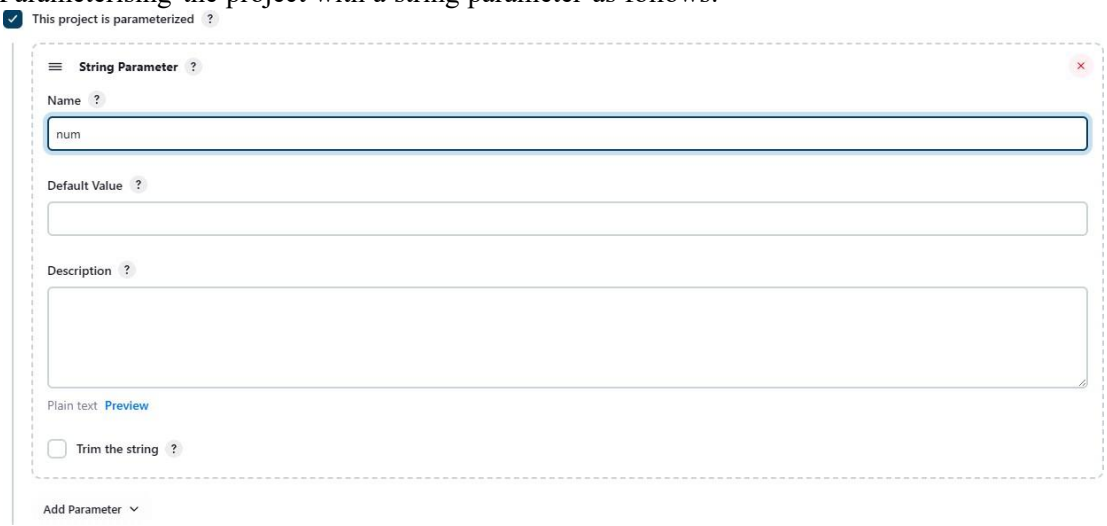
Running the Python script on the terminal:



Creating a new freestyle project:



Parameterising the project with a string parameter as follows:



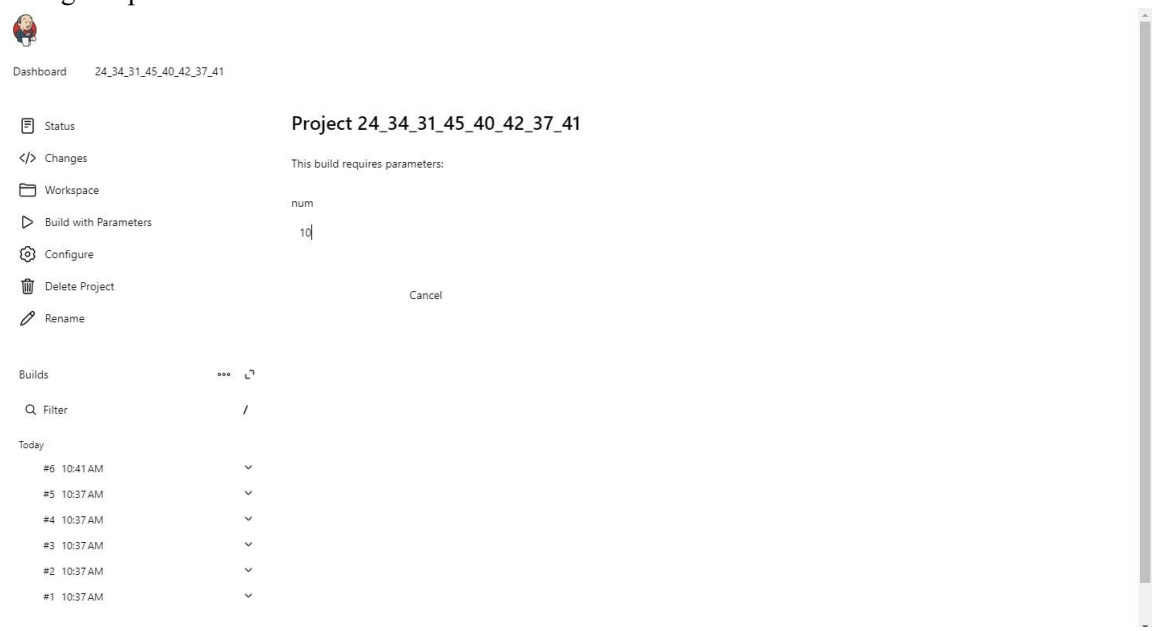
Configuring the build steps:

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

Theory:

Introduction to Agile Methodology

Agile methodology is a **flexible, iterative, and collaborative approach** to software development and project management. It focuses on **continuous improvement, rapid delivery, and adaptability** to changing requirements. Agile follows an incremental approach, where projects are divided into smaller **iterations (sprints)**, allowing teams to deliver working software quickly and efficiently.

Key Principles of Agile (According to the Agile Manifesto)

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.

These principles emphasize **flexibility, customer involvement, and iterative development** to improve efficiency and adaptability.

Agile Frameworks and Approaches

Agile is an umbrella term that includes various frameworks, such as:

1. **Scrum** – A structured framework with defined roles (Scrum Master, Product Owner, Development Team) and time-boxed iterations called **sprints**.
2. **Kanban** – A visual workflow management system that helps teams track work progress using a **Kanban board**.
3. **Extreme Programming (XP)** – Focuses on engineering practices such as test-driven development (TDD) and pair programming.
4. **Lean** – Aims to **eliminate waste** and improve efficiency.
5. **SAFe (Scaled Agile Framework)** – A framework for applying Agile at the enterprise level.

Among these, **Scrum and Kanban** are widely used in software development projects.

Test Case Management in Agile Development

Test case management is a crucial aspect of software testing in Agile. Since Agile follows **shorter development cycles**, testing is done **continuously** to ensure that software is stable and defect-free.

Key Aspects of Test Case Management in Agile

- **Continuous Testing** – Testing occurs throughout the sprint, ensuring early bug detection.
- **Automated Testing** – Automation is widely used to speed up testing and reduce manual effort.
- **User Story-Based Testing** – Test cases are aligned with user stories rather than traditional detailed test plans.
- **Exploratory Testing** – Testers actively explore the application instead of following strict test scripts.
- **Collaboration with Developers** – Testers, developers, and business analysts work together to validate requirements and test cases.

Since Agile follows rapid iterations, **test case management tools like JIRA** are widely used for **test planning, execution, and reporting**.

JIRA as a Test Case Management Tool

Introduction to JIRA

JIRA is a **popular project management and issue-tracking tool** developed by Atlassian. It is widely used in Agile development for **bug tracking, task management, and test case management**. JIRA supports Agile methodologies like **Scrum and Kanban**, making it an ideal tool for software teams.

Features of JIRA for Test Case Management

1. **Test Case Creation** – JIRA allows teams to create, organize, and manage test cases as part of their development workflow.
2. **Test Execution** – Test cases can be executed manually or integrated with **automation tools**.
3. **Bug Tracking** – Defects found during testing can be logged and assigned to developers for resolution.
4. **Integration with CI/CD Tools** – JIRA integrates with tools like **Jenkins, Selenium, and TestRail** for automated testing.
5. **Custom Dashboards and Reports** – JIRA provides real-time tracking of test progress, defect status, and test coverage.
6. **Sprint Management** – Teams can track test cases and bugs across different sprints.

How JIRA is Used in Agile Testing

1. Creating User Stories and Test Cases

- In Agile, **test cases are linked to user stories** to ensure that every feature is tested.
- Testers write test cases in **JIRA tickets** and track their execution during sprints.

2. Logging and Tracking Bugs

- If a bug is found, it is logged in JIRA as an **issue** with details like severity, priority, and steps to reproduce.
- The bug is assigned to a developer for fixing.

3. Executing Test Cases

- Testers **execute** test cases and update their status as **Passed, Failed, or Blocked** in JIRA.
- Automation tools can be integrated with JIRA to run tests and update results automatically.

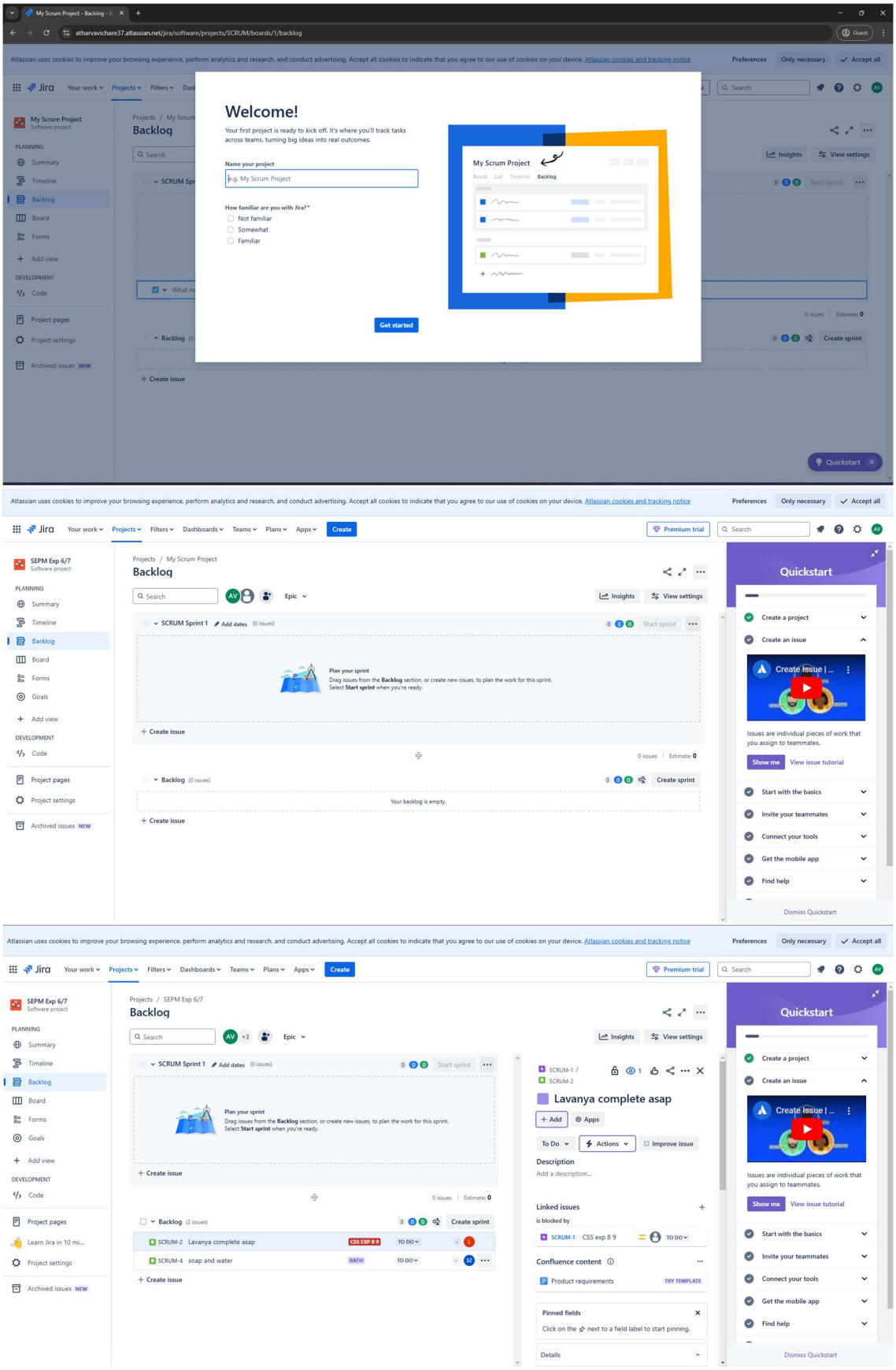
4. Reporting and Analytics

- JIRA provides dashboards with insights into **test execution progress, defect trends, and sprint performance**.
- Teams use these reports for decision-making and sprint retrospectives.

Advantages of Using JIRA for Test Case Management

✅ **Seamless Integration with Agile Workflows** – JIRA fits perfectly into Agile methodologies like Scrum and Kanban.
✅ **Collaboration Between Teams** – Developers, testers, and product owners can track project progress in one place.
✅ **Automation and CI/CD Integration** – Reduces manual effort by integrating with automated testing tools.
✅ **Custom Workflows** – JIRA allows teams to define their own workflows for **test execution, defect tracking, and reporting**.
✅ **Improved Visibility** – Dashboards and reports provide real-time insights into test progress and defect status.

Output:


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
e.g. My Scrum Project
How familiar are you with Jira?*
Not familiar
Somewhat
Familiar
Get started
My Scrum Project
Backlog
Create issue
Create sprint
Quickstart
Create a project
Create an issue
Start with the basics
Invite your teammates
Connect your tools
Get the mobile app
Find help
Dismiss Quickstart
Plan your sprint
Drag issues from the Backlog section, or create new issues, to plan the work for this sprint.
Select Start sprint when you're ready.
Your backlog is empty.
Lavanya complete asap
Description
Add a description...
Linked issues
Confluence content
Pinned fields
Details

Conclusion: Successfully Implemented Agile Methodology and Test case Management using JIRA Tool

Name: Faiz Shaikh
Roll no.: 93
Batch: T22

Experiment 7

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.

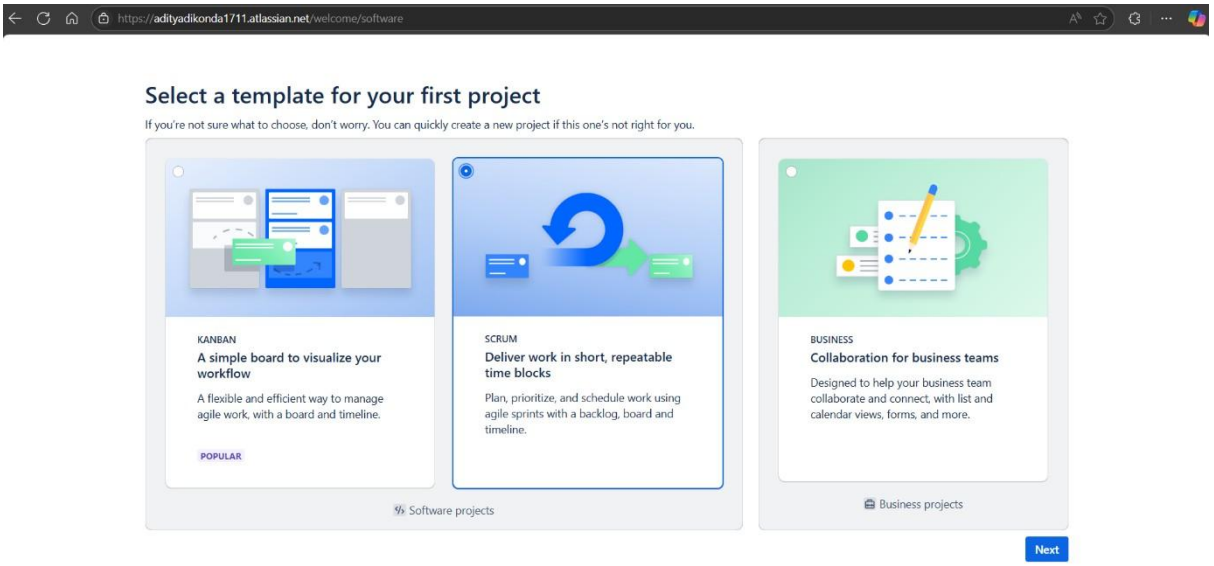
Select the board type as Scrum and choose whether to start with a new project template or add the board to existing projects.

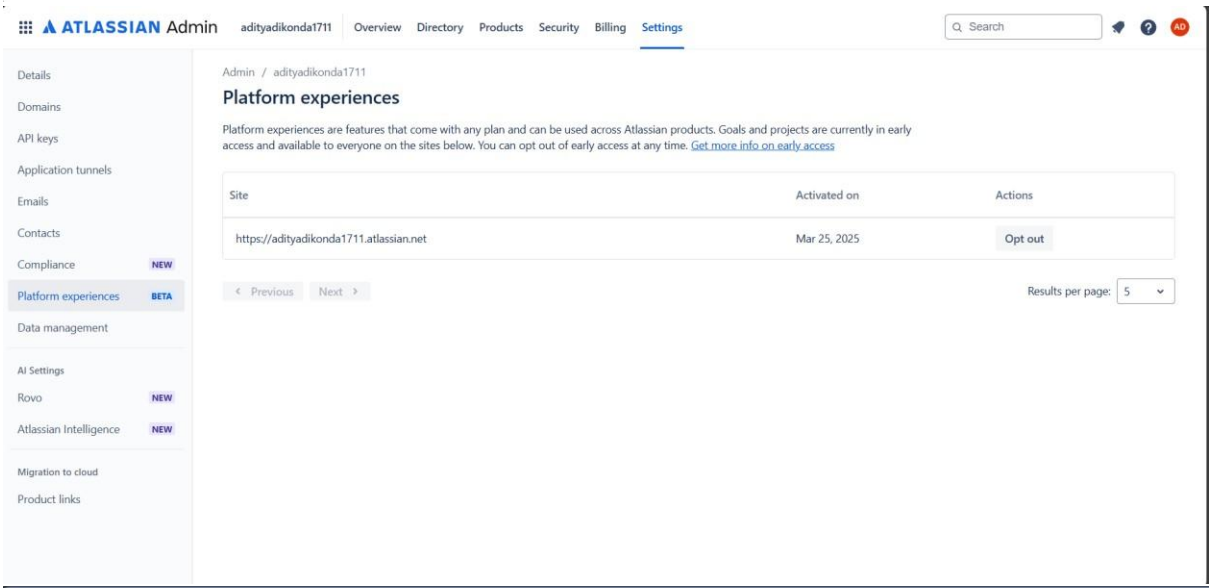
Configure columns and quick filters to reflect your team's process and focus on specific issues quickly

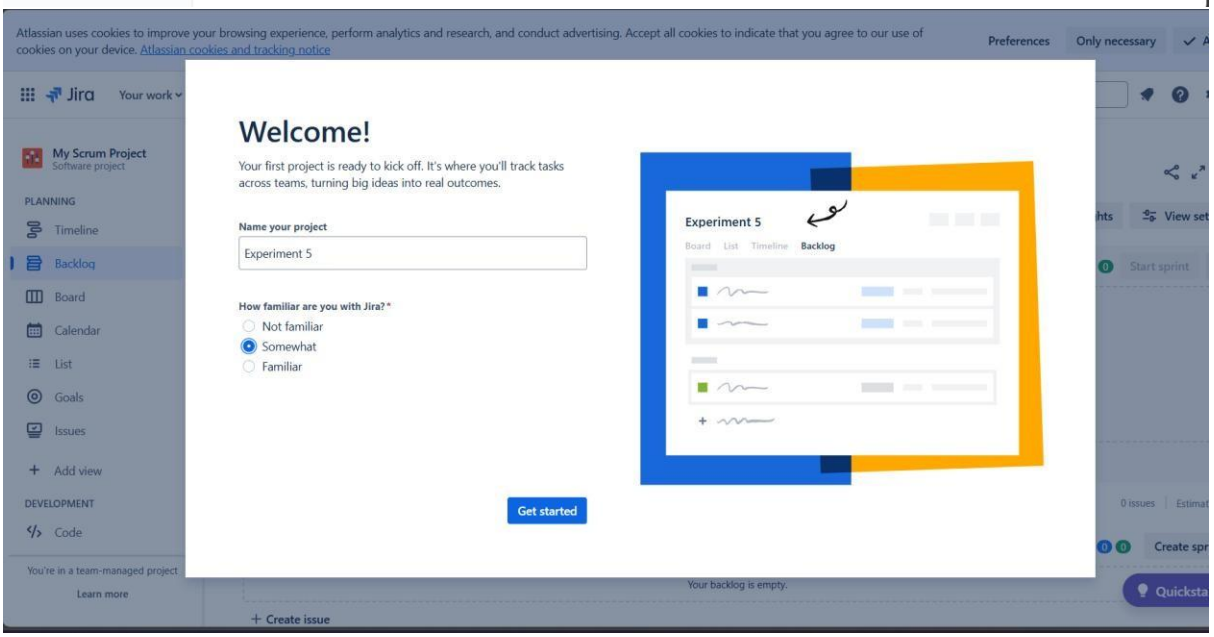
Navigate Between Boards:

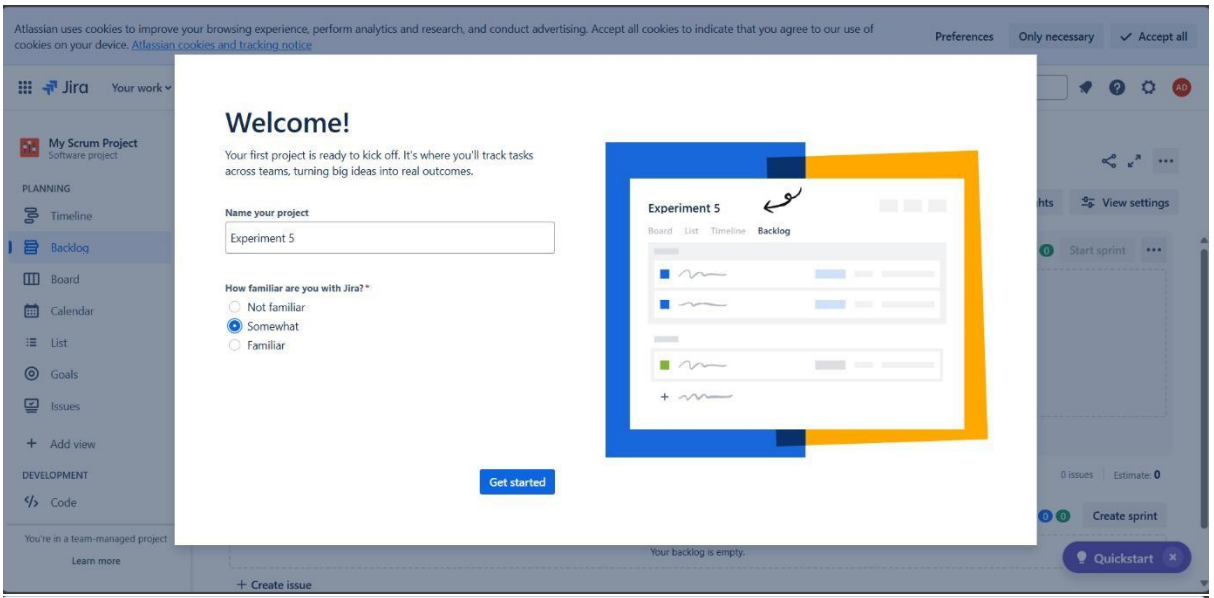
Use the board switcher located in the left-hand menu under the project name to move between different boards in JIRA

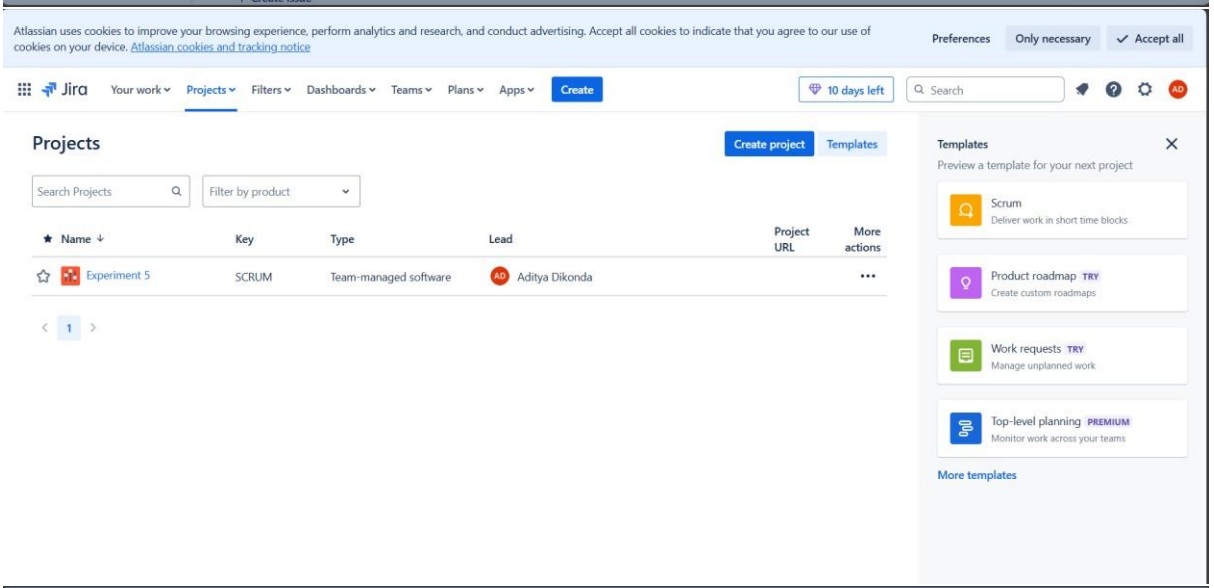
By following these steps, a Scrum Master can effectively create a Scrum Board in JIRA, enabling efficient management of tasks, user stories, and sprints within the agile project management framework.

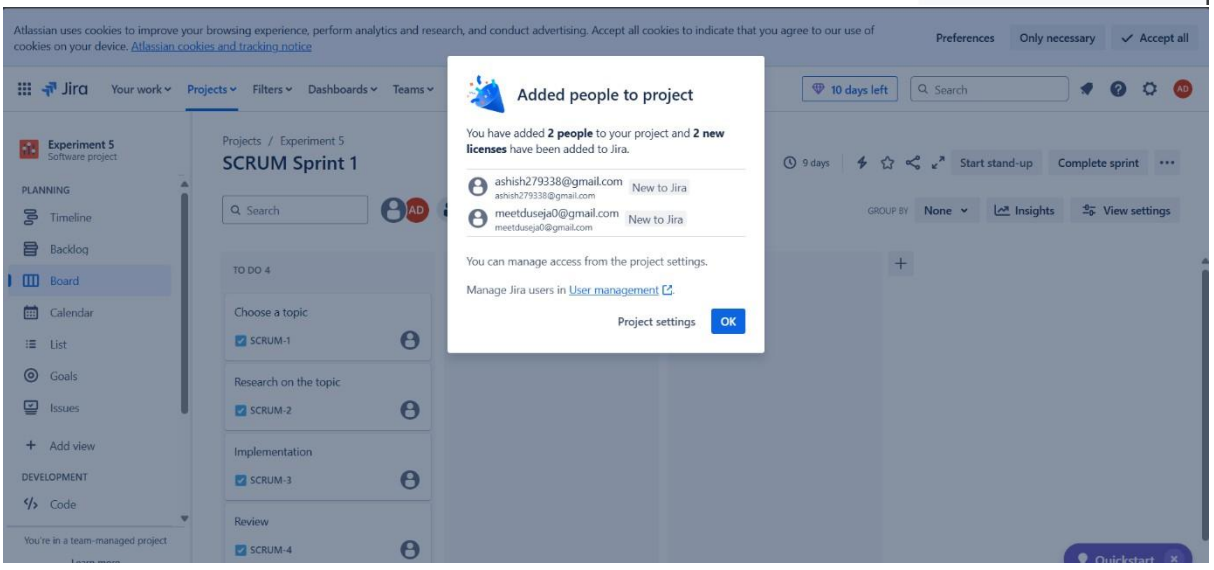

https://adityadikonda1711.atlassian.net/welcome/software
Select a template for your first project
If you're not sure what to choose, don't worry. You can quickly create a new project if this one's not right for you.
KANBAN
A simple board to visualize your workflow
A flexible and efficient way to manage agile work, with a board and timeline.
POPULAR
SCRUM
Deliver work in short, repeatable time blocks
Plan, prioritize, and schedule work using agile sprints with a backlog, board and timeline.
BUSINESS
Collaboration for business teams
Designed to help your business team collaborate and connect, with list and calendar views, forms, and more.
Software projects
Business projects
Next

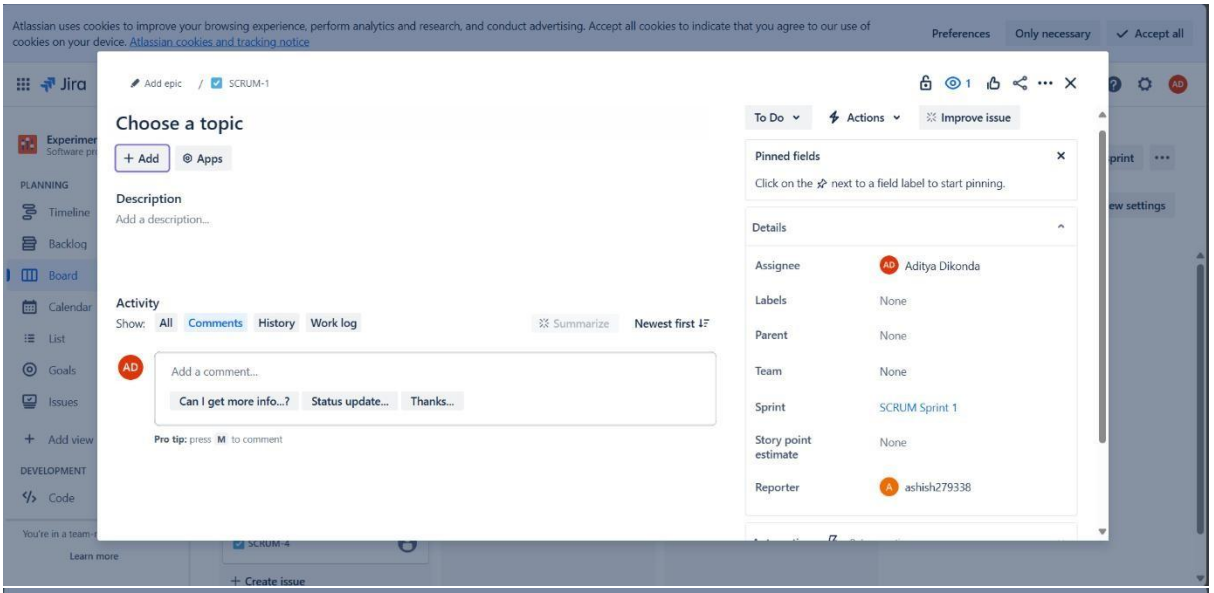

ATLASSIAN Admin
adityadikonda1711
Overview
Directory
Products
Security
Billing
Settings
Search
Details
Domains
API keys
Application tunnels
Emails
Contacts
Compliance
NEW
Platform experiences
BETA
Data management
AI Settings
Rovo
NEW
Atlassian Intelligence
NEW
Migration to cloud
Product links
Admin / adityadikonda1711
Platform experiences
Platform experiences are features that come with any plan and can be used across Atlassian products. Goals and projects are currently in early access and available to everyone on the sites below. You can opt out of early access at any time. Get more info on early access
Site
Activated on
Actions
https://adityadikonda1711.atlassian.net
Mar 25, 2025
Opt out
Previous
Next
Results per page:
5

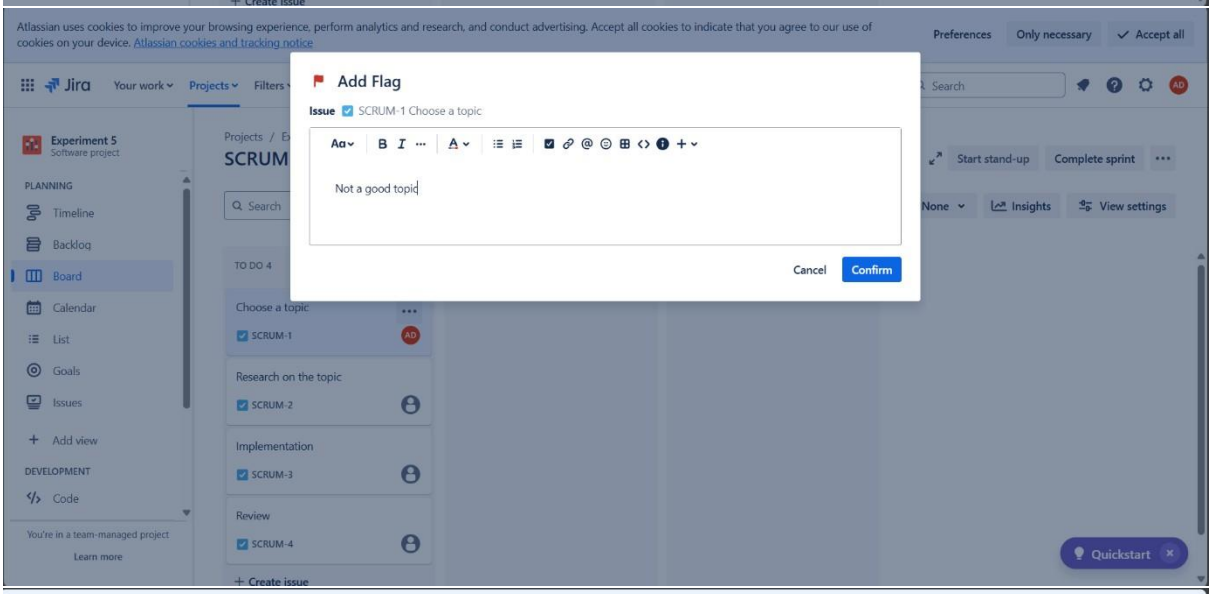

Atlassian uses cookies to improve your browsing experience, perform analytics and research, and conduct advertising. Accept all cookies to indicate that you agree to our use of cookies on your device. Atlassian cookies and tracking notice
Preferences
Only necessary
Jira
Your work
My Scrum Project
Software project
PLANNING
Timeline
Backlog
Board
Calendar
List
Goals
Issues
Add view
DEVELOPMENT
Code
You're in a team-managed project
Learn more
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
Board
List
Timeline
Backlog
Your backlog is empty.
Create issue

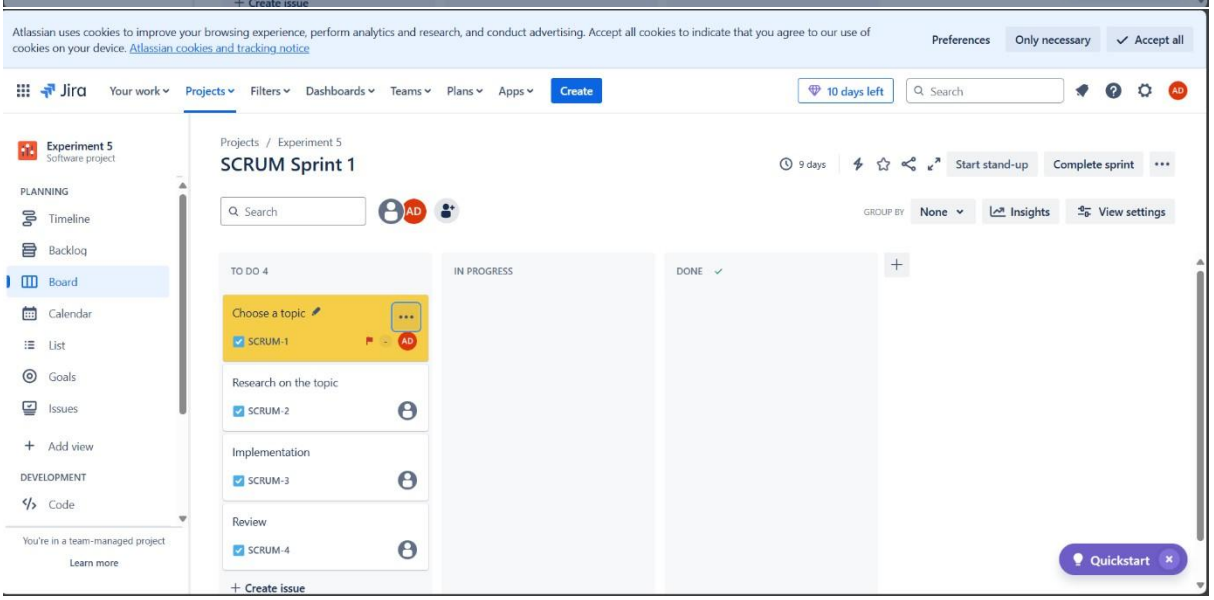

Welcome!
Your first project is ready to kick off. It's where you'll track tasks across teams, turning big ideas into real outcomes.
Name your project
Experiment 5
How familiar are you with Jira?*
Not familiar
Somewhat
Familiar
Get started
Experiment 5
Board List Timeline Backlog

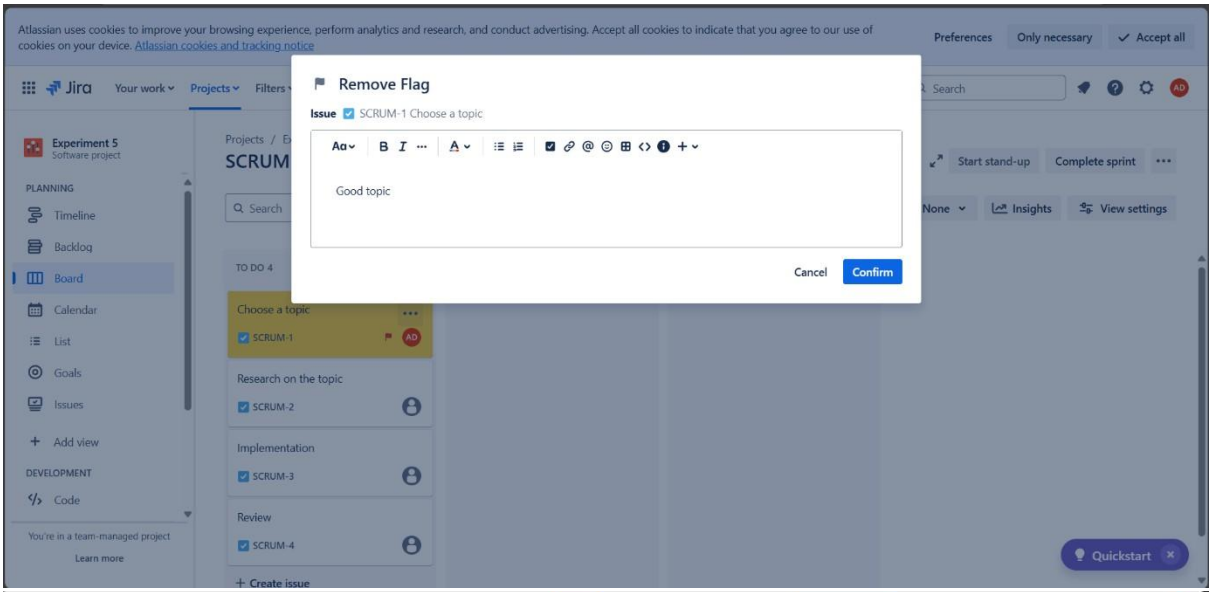

Projects
Create project
Templates
Search Projects
Filter by product
Name
Key
Type
Lead
Project URL
More actions
Experiment 5
SCRUM
Team-managed software
Aditya Dikonda
Templates
Preview a template for your next project
Scrum
Deliver work in short time blocks
Product roadmap TRY
Create custom roadmaps
Work requests TRY
Manage unplanned work
Top-level planning PREMIUM
Monitor work across your teams
More templates

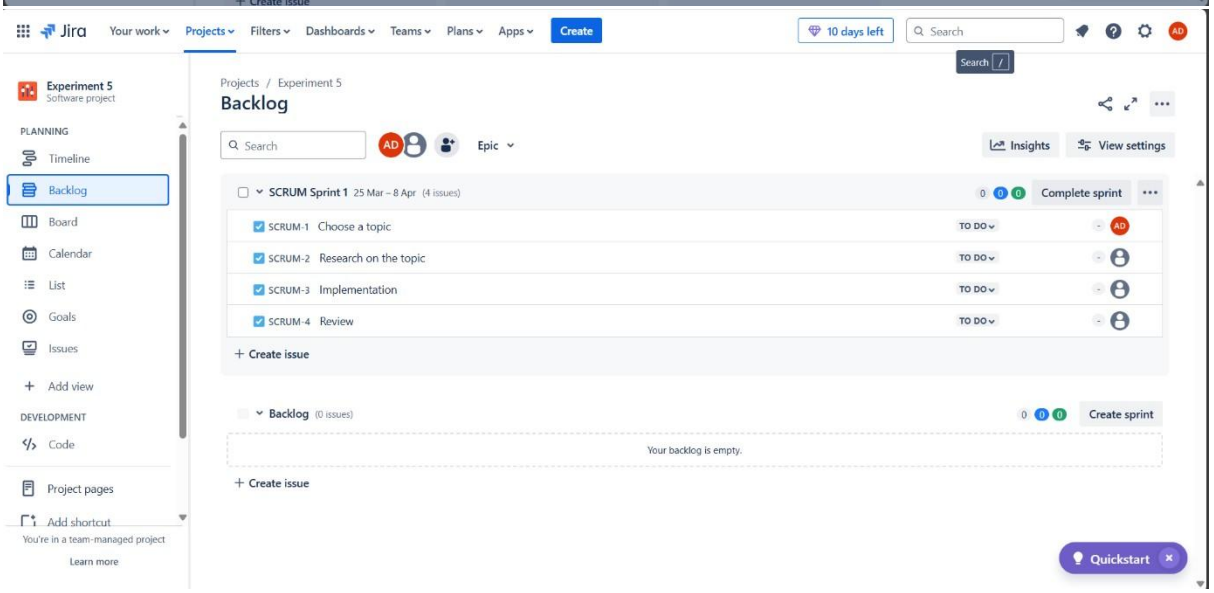

Added people to project
You have added 2 people to your project and 2 new licenses have been added to Jira.
ashish279338@gmail.com
New to Jira
meetduseja0@gmail.com
New to Jira
You can manage access from the project settings.
Manage Jira users in User management
Project settings
OK
SCRUM Sprint 1
TO DO 4
Choose a topic
SCRUM-1
Research on the topic
SCRUM-2
Implementation
SCRUM-3
Review
SCRUM-4

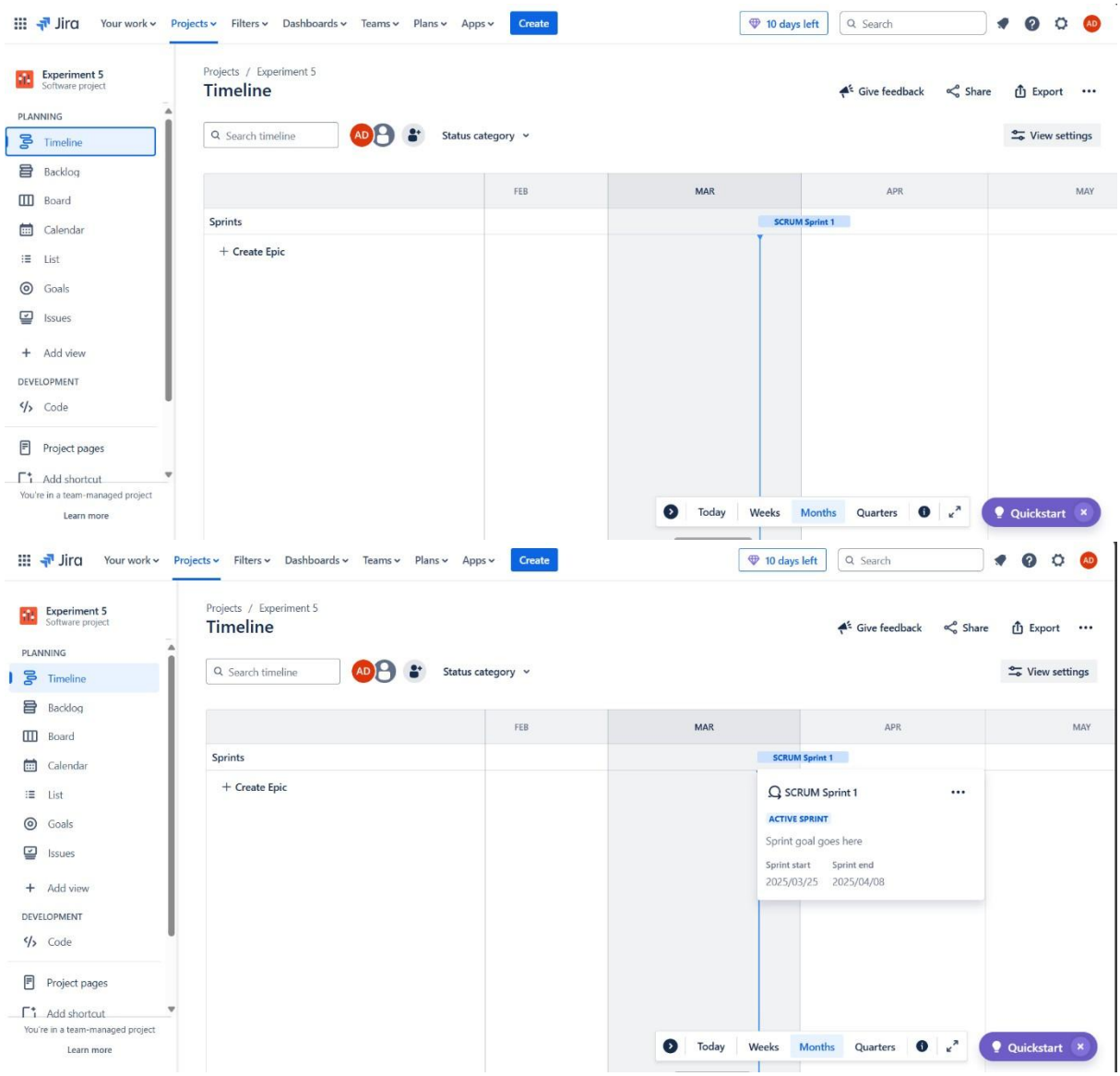

Choose a topic
+ Add
Apps
Description
Add a description...
Activity
Show: All Comments History Work log
Summarize
Newest first
Add a comment...
Can I get more info...?
Status update...
Thanks...
Pro tip: press M to comment
To Do
Actions
Improve issue
Pinned fields
Click on the next to a field label to 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
SCRUM
TO DO 4
Choose a topic
SCRUM-1
Research on the topic
SCRUM-2
Implementation
SCRUM-3
Review
SCRUM-4
Create issue


Projects / Experiment 5
SCRUM Sprint 1
9 days
Start stand-up
Complete sprint
Search
GROUP BY
None
Insights
View settings
TO DO 4
IN PROGRESS
DONE
Choose a topic
SCRUM-1
Research on the topic
SCRUM-2
Implementation
SCRUM-3
Review
SCRUM-4
Create issue
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
Search
Remove Flag
Issue SCRUM-1 Choose a topic
Good topic
Cancel
Confirm
Experiment 5
Software project
PLANNING
Timeline
Backlog
Board
Calendar
List
Goals
Issues
Add view
DEVELOPMENT
Code
You're in a team-managed project
Learn more
SCRUM
Start stand-up
Complete sprint
None
Insights
View settings
TO DO 4
Choose a topic
SCRUM-1
Research on the topic
SCRUM-2
Implementation
SCRUM-3
Review
SCRUM-4
Create issue
Quickstart


Jira
Your work
Projects
Filters
Dashboards
Teams
Plans
Apps
Create
10 days left
Search
Search /
Experiment 5
Software project
PLANNING
Timeline
Backlog
Board
Calendar
List
Goals
Issues
Add view
DEVELOPMENT
Code
Project pages
Add shortcut
You're in a team-managed project
Learn more
Projects / Experiment 5
Backlog
Search
Epic
Insights
View settings
SCRUM Sprint 1 25 Mar – 8 Apr (4 issues)
Complete sprint
SCRUM-1 Choose a topic
SCRUM-2 Research on the topic
SCRUM-3 Implementation
SCRUM-4 Review
TO DO
Create issue
Backlog (0 issues)
Create sprint
Your backlog is empty.
Create issue
Quickstart



Conclusion:

Scrum project was implemented using JIRA.

FAIZ SHAIKH
T22 – 2201093

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.

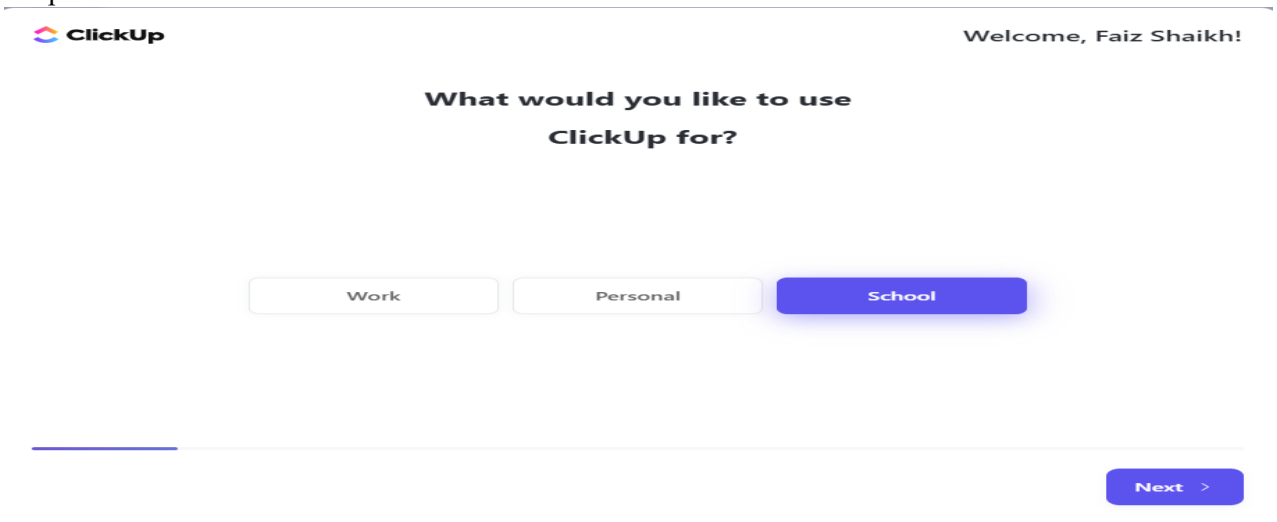
3. Add Tasks to the Project:
   - Click New Task, provide a task name, description, assignee, and due date.
   - Define task priorities and set dependencies.
4. Enable Gantt Chart View:
   - Go to View Options and select Gantt Chart.
   - Adjust start and due dates to organize the timeline.
5. Define 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.

---

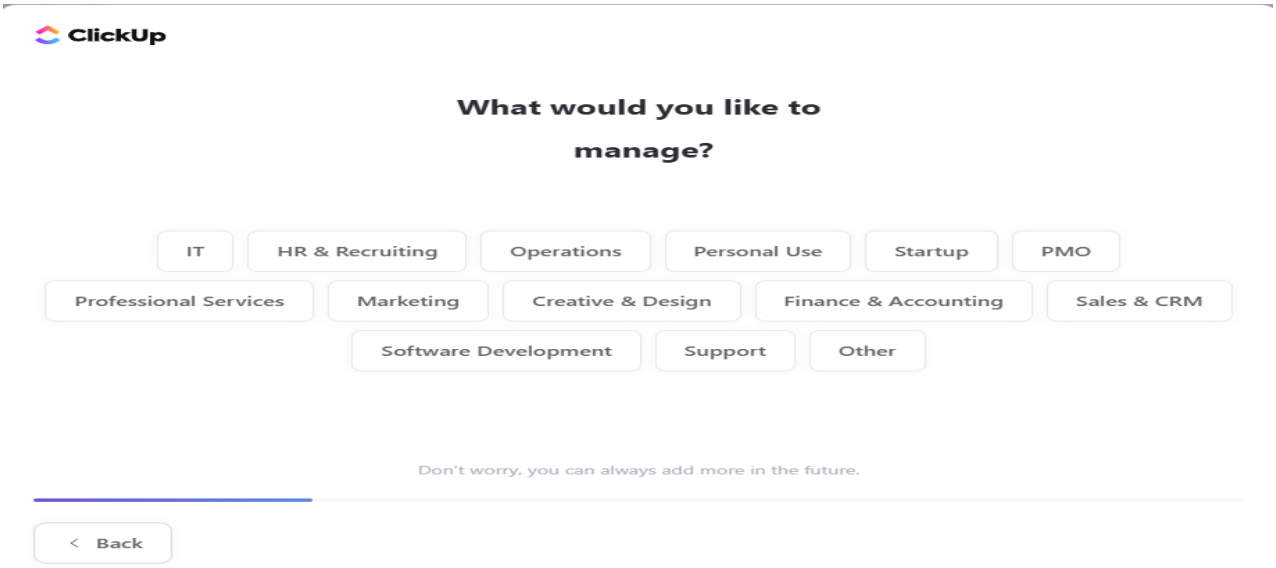
4. Tracking and Managing a Gantt Chart in ClickUp

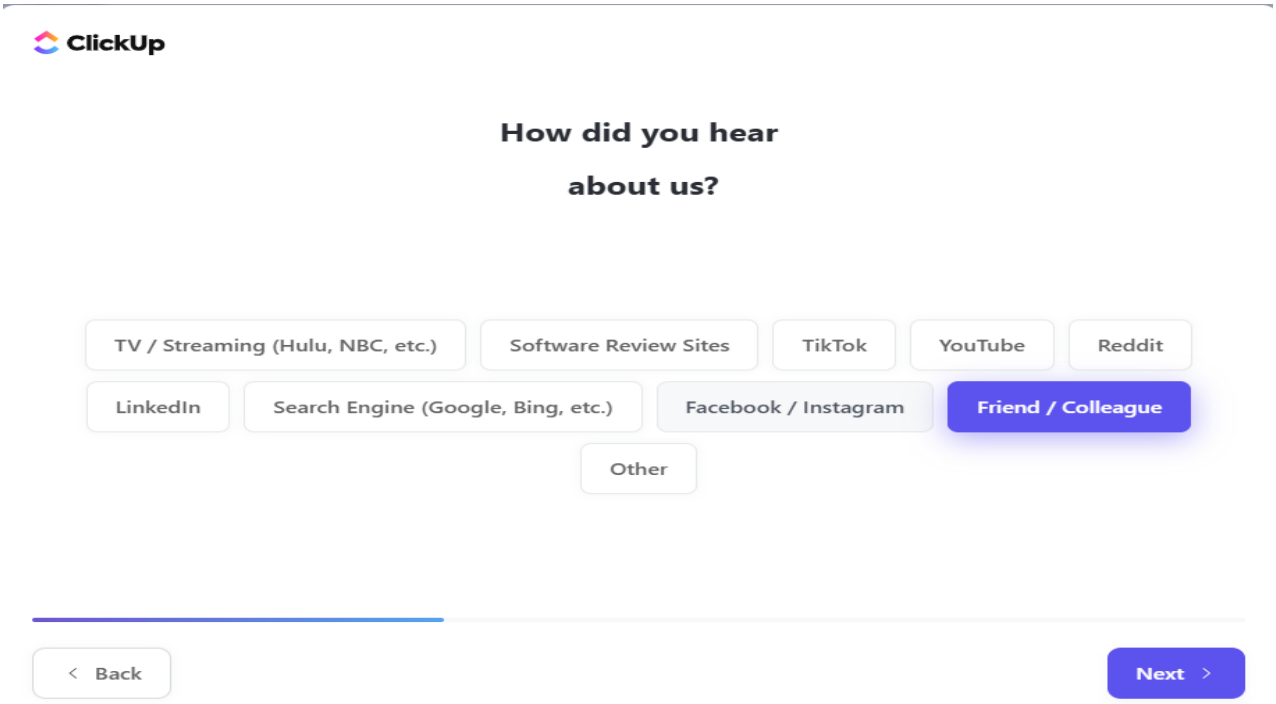
- Modify Timelines: Drag tasks to change deadlines dynamically.
- Monitor Progress: Use task completion percentages to track status.
- Adjust Dependencies: Update task sequences when project plans change.
- Generate Reports: Use ClickUp's reporting features to analyze workload and delays.

Output:

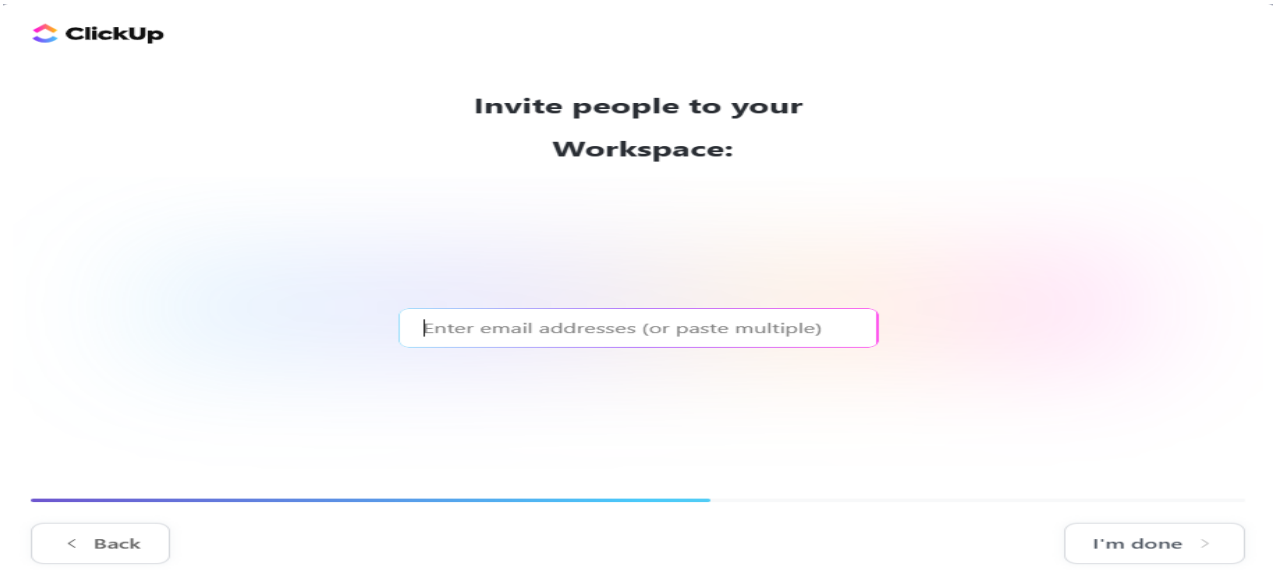


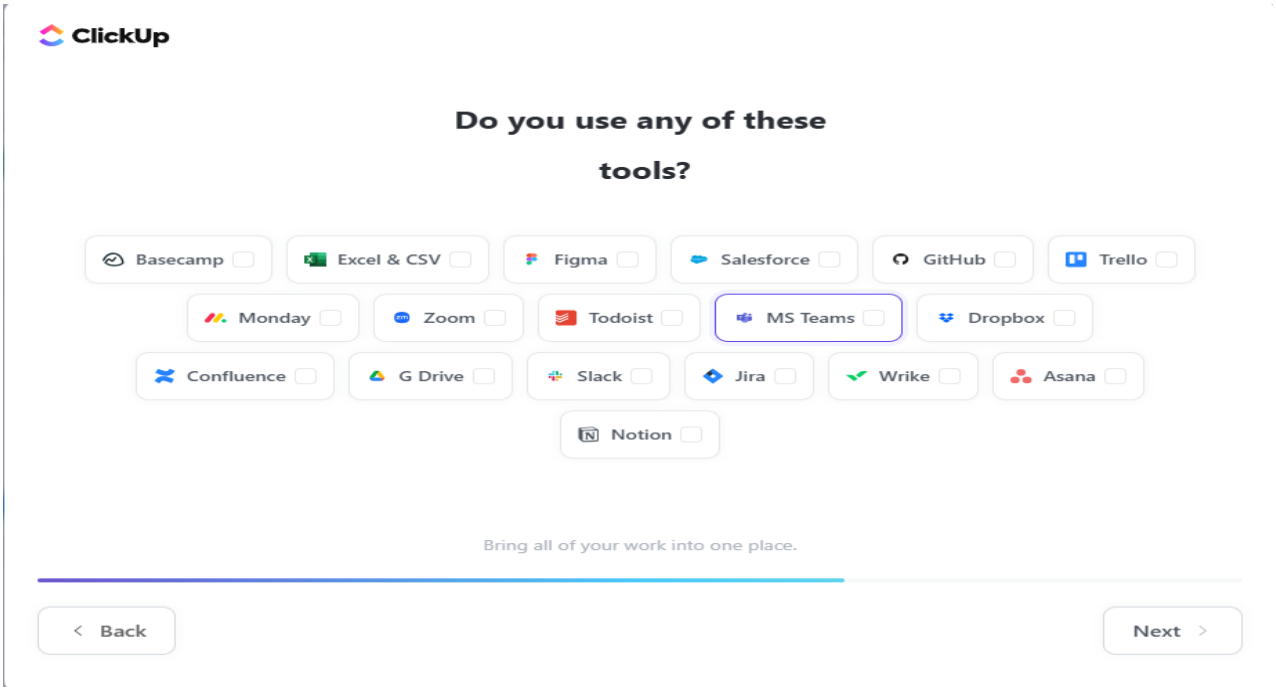
FAIZ SHAIKH
T22 – 2201093





FAIZ SHAIKH
T22 – 2201093





FAIZ SHAIKH
T22 – 2201093

ClickUp

Which features are you interested in trying?

Tasks & Projects | Workload | Automations | Dashboards | Docs & Wikis

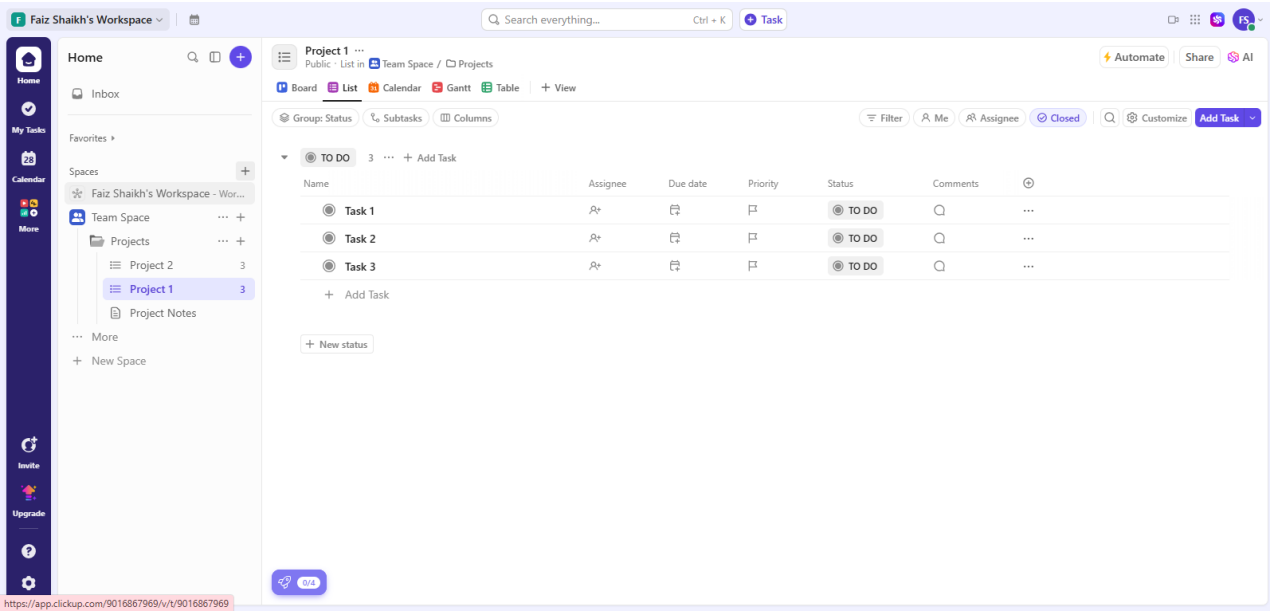
Whiteboards | Ask AI | Boards & Kanban | CRM | Clips | Sprints

Goals & OKRs | Forms | Gantt Charts | Time Tracking | Calendar | Chat

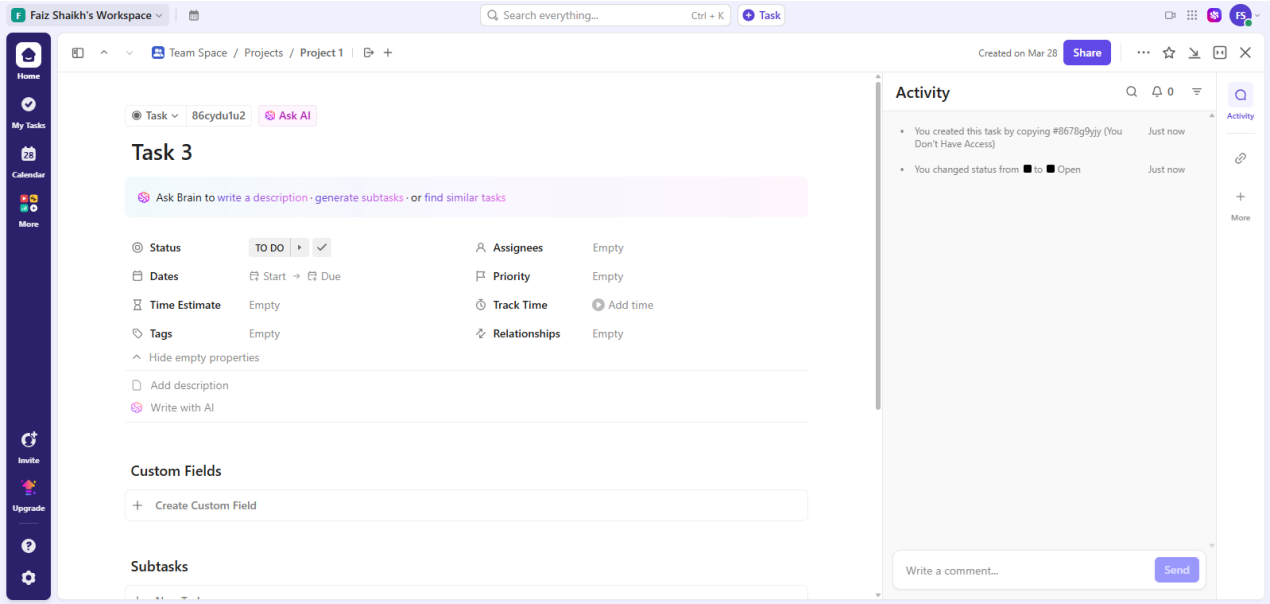
Scheduling

Don't worry, you'll have access to all of these in your Workspace.

Back | Next



FAIZ SHAIKH
T22 – 2201093



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.

Name : Faiz Shaikh
Roll No. : 93
Batch : T22

Experiment 7 :Docker Basics

Aim :

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

Theory :

Dockerfiles are the cornerstone of creating Docker images. They contain a set of instructions that automate the process of building a Docker image, specifying everything from the base operating system to the application code, dependencies, and configuration settings.

---

1. What is a Dockerfile?

A **Dockerfile** is a plain text file that defines the steps required to build a Docker image. It contains a series of commands (or instructions) that specify how the image should be constructed.

- **Purpose:** Automate the creation of Docker images for reproducibility, scalability, and consistency.
- **Format:** Written in a simple scripting language, using instructions like `FROM`, `RUN`, `COPY`, `CMD`, etc.

---

2. Basic Structure of a Dockerfile

A typical Dockerfile looks like this:

```
# Use an official Python runtime as a parent image
FROM python:3.9-slim

# Set the working directory inside the container
WORKDIR /app
```

```
# Copy the current directory contents into the container at /app
COPY . /app

# Install any necessary dependencies
RUN pip install --no-cache-dir -r requirements.txt

# Make port 80 available to the world outside this container
EXPOSE 80

# Define environment variable
ENV NAME World

# Run app.py when the container launches
CMD ["python", "app.py"]
```

---

3. Common Dockerfile Instructions

1. `FROM` (Base Image)

- **Purpose:** Specifies the base image for your Docker image.

Example:

```
FROM ubuntu:20.04
FROM node:14
FROM python:3.9-slim
```

- **Note:** This is the first instruction and is mandatory in most cases.

2. `WORKDIR` (Set Working Directory)

- **Purpose:** Defines the directory inside the container where subsequent instructions will be executed.

Example:

```
WORKDIR /app
```

3. `COPY` (Copy Files)

- **Purpose:** Copies files or directories from the host system into the container.

Example:

```
COPY . /app
```

- **Variants:**
    - `COPY <src> <dest>`: Copies a file or directory from the build context to the container.
    - `ADD` is similar but supports remote URLs and tar file extraction.

4. `RUN` (Execute Commands)

- **Purpose:** Executes commands inside the container during the image build process.

Example:

```
RUN apt-get update && apt-get install -y curl
RUN pip install --no-cache-dir -r requirements.txt
```

- **Tip:** Each `RUN` creates a new layer in the image. Combine commands with `&&` to reduce image size.

5. `EXPOSE` (Expose Ports)

- **Purpose:** Informs Docker that the container will listen on the specified network ports at runtime.

Example:

```
EXPOSE 80
```

- **Note:** This does **not** publish the port; it's just a way to document which ports should be exposed.

6. `ENV` (Set Environment Variables)

- **Purpose:** Sets environment variables inside the container.

Example:

```
ENV APP_ENV=production
```

7. `CMD` (Default Command)

- **Purpose:** Specifies the default command to run when the container starts.

Example:

```
CMD ["python", "app.py"]
```

- **Key Points:**
  - Only **one** `CMD` instruction is allowed.
  - If you provide a command when running the container (`docker run`), it will override `CMD`.

8. `ENTRYPOINT` (Set Entry Point)

- **Purpose:** Defines a command that will always be executed when the container starts.

Example:

```
ENTRYPOINT ["python"]
CMD ["app.py"]
```

- **Difference from `CMD`:** `ENTRYPOINT` is not overridden unless explicitly done with `--entrypoint` in `docker run`.

---

4. Building Images from a Dockerfile

To build an image, use the `docker build` command:

```
docker build -t myapp:latest .
```

- **`-t myapp:latest`**: Tags the image as `myapp` with the `latest` tag.
- **`.`**: Specifies the build context (the current directory).

Build Options:

- `-f <file>`: Specify a custom Dockerfile name.
- `--no-cache`: Build the image without using the cache.
- `--build-arg <arg>`: Pass build-time arguments.

---

5. Managing Docker Images

List Images:

```
docker images
```

Remove an Image:

```
docker rmi myapp:latest
```

Run a Container from an Image:

```
docker run -p 8080:80 myapp:latest
```

---

6. Multi-Stage Builds (Advanced)

Multi-stage builds help reduce image size by separating the build environment from the runtime environment.

```
# Stage 1: Build stage
FROM node:14 AS build
WORKDIR /app
COPY package.json ./
RUN npm install
```

```
COPY . .

# Stage 2: Production stage
FROM node:14-slim
WORKDIR /app
COPY --from=build /app /app
CMD ["node", "server.js"]
```

- This technique helps keep the final image lean by excluding unnecessary build tools.

---

7. Best Practices for Dockerfiles

1. **Use Minimal Base Images:** e.g., `alpine` for small image sizes.
2. **Leverage Caching:** Order instructions from least to most frequently changing.
3. **Reduce Layers:** Combine `RUN` commands with `&&`.
4. **Avoid Root:** Run applications as non-root users when possible.
5. **Clean Up:** Remove unnecessary files after installation to reduce image size.

Screenshots:

```
../MiscRepos/sepm_lab/files   vi src/server.js
1  const express = require("express");
1  const app = express();
2  const PORT = process.env.PORT || 5000;
3
4  app.get("/", (req, res) => {
5    res.status(200).json({ msg: "Hello, Docker :)" });
6  });
7
8  const init = async () => {
9    try {
10     app.listen(PORT, () => {
11       console.log(`Server is Listening on port ${PORT}...`);
12     });
13   } catch (error) {
14     console.log("There was an error : ", error);
15   }
16 };
17 init();
~
~
~
```

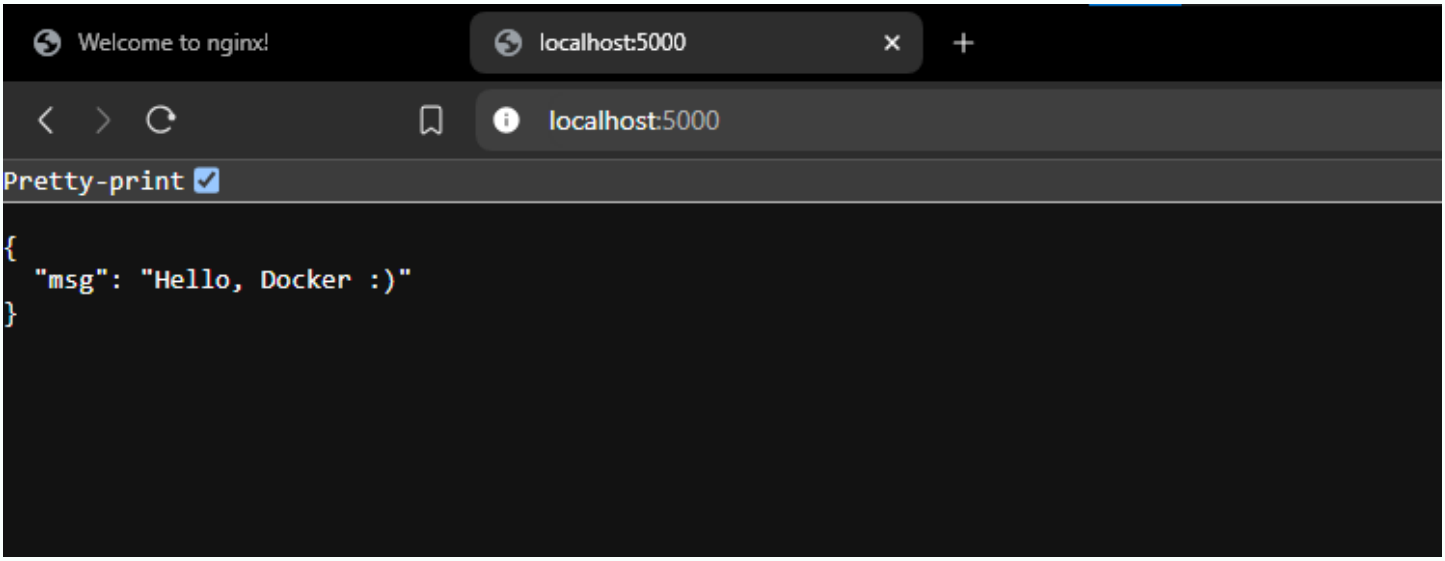
```
../MiscRepos/sepm_lab/files   vi package.json
1  {
1    "name": "docker_demo",
2    "version": "1.0.0",
3    "description": "",
4    "main": "src/server.js",
5    "scripts": {
6      "start": "node src/server.js"
7    },
8    "keywords": [],
9    "author": "taha",
10   "license": "ISC",
11   "dependencies": {
12     "express": "^5.1.0"
13   }
14 }
~
~
```

```
../MiscRepos/sepm_lab/files   pos/sepm_lab/Exp10_Docke   vi dockerfile
10 FROM node:19-alpine
9
8  COPY package.json /app/
7  COPY src /app/
6
5  WORKDIR /app
4
3  RUN npm install
2
1  CMD ["node", "server.js"]
11
~
~
~
~
~
~
~
~
```

```
/d/MiscRepos/sepm_lab/Exp10_Docker git:(master)±11 (0.452s)
ls -a
./  ../  dockerfile  node_modules/  package.json  package-lock.json  src/

/d/MiscRepos/sepm_lab/Exp10_Docker git:(master)±11 (3.322s)
vi src/server.js

/d/MiscRepos/sepm_lab/Exp10_Docker git:(master)±11 (2.621s)
vi package.json

/d/MiscRepos/sepm_lab/Exp10_Docker git:(master)±11 (2.67s)
vi dockerfile
```

```
/d/MiscRepos/sepm_lab/Exp10_Docker git:(master)±11 (6.075s)
docker build -t demo-node-app:1.0.0 .

[+] Building 4.2s (11/11) FINISHED                                                                                  docker:desktop-linux
 => [internal] load build definition from dockerfile                                                                              0.1s
 => => transferring dockerfile: 169B                                                                                              0.0s
 => [internal] load metadata for docker.io/library/node:19-alpine                                                                 1.7s
 => [auth] library/node:pull token for registry-1.docker.io                                                                       0.0s
 => [internal] load .dockerignore                                                                                                 0.1s
 => => transferring context: 2B                                                                                                   0.0s
 => [1/5] FROM docker.io/library/node:19-alpine@sha256:8ec543d4795e2e85af924a24f8acb039792ae9fe8a42ad5b4bf4c277ab34b62e           0.1s
 => => resolve docker.io/library/node:19-alpine@sha256:8ec543d4795e2e85af924a24f8acb039792ae9fe8a42ad5b4bf4c277ab34b62e           0.1s
 => [internal] load build context                                                                                                 0.1s
 => => transferring context: 90B                                                                                                  0.0s
 => CACHED [2/5] COPY package.json /app/                                                                                          0.0s
 => CACHED [3/5] COPY src /app/                                                                                                   0.0s
 => CACHED [4/5] WORKDIR /app                                                                                                     0.0s
 => CACHED [5/5] RUN npm install                                                                                                  0.0s
 => exporting to image                                                                                                            1.4s
 => => exporting layers                                                                                                           0.0s
 => => exporting manifest sha256:7b49e78368e8d2a07be85207b937d4db0d2aa99a51bee789c200f957fbe206df                                 0.0s
 => => exporting config sha256:a844a1b4c76601423a9e4b4ed6ae6de45864d8c2f50e072701bc0441a3801367                                   0.0s
 => => exporting attestation manifest sha256:2fa53de8c4c2a9d2d68fc0b3012f701a0756da075367a4c60b183925dedd87d0                     0.1s
 => => exporting manifest list sha256:152bfc3265d14f5bd54fc0a8688050703e28988be62e4ccbb1d3a6bd9ee98fb8                            0.1s
 => => naming to docker.io/library/demo-node-app:1.0.0                                                                            0.0s
 => => unpacking to docker.io/library/demo-node-app:1.0.0                                                                         1.0s

/d/MiscRepos/sepm_lab/Exp10_Docker git:(master)±11 (1.151s)
docker images
REPOSITORY      TAG       IMAGE ID       CREATED          SIZE
demo-node-app   1.0.0     152bfc3265d1   9 minutes ago    261MB
nginx           latest    124b44bfc9cc   7 weeks ago      279MB
nginx           1.23      f5747a42e3ad   22 months ago    214MB

/d/MiscRepos/sepm_lab/Exp10_Docker git:(master)±11
docker run --name sepm-expt -p 5000:5000 demo-node-app:1.0.0
Server is Listening on port 5000...
```

```
/d/MiscRepos/sepm_lab/Exp10_Docker git:(master)±11 (1.015s)
docker ps -a
CONTAINER ID   IMAGE                 COMMAND                  CREATED          STATUS                      PORTS                    NAMES
a111513ae571   demo-node-app:1.0.0   "docker-entrypoint.s…"   2 minutes ago    Up 2 minutes                0.0.0.0:5000->5000/tcp   sepm-expt
7427673945ec   nginx:1.23            "/docker-entrypoint.…"   52 minutes ago   Exited (0) 7 minutes ago                             web_app

/d/MiscRepos/sepm_lab/Exp10_Docker  git:(master) ✓ ±11
```



Conclusion :

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

Name : Faiz Shaikh
Roll No. : 93
Batch : T22

Experiment 10: Docker Files

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 platform designed to make it easier to create, deploy, and run applications by using containers. Containers allow a developer to package up an application with all parts it needs, such as libraries and other dependencies, and ship it all out as one package. Here's a detailed look at the basics of Docker:

1. What is Docker?

Docker is an open-source platform that automates the deployment of applications inside lightweight, portable containers. A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another.

---

2. Key Concepts in Docker

Containers

- **Definition:** Containers are isolated environments where applications run with everything they need to function.
- **Features:** They are lightweight, fast, and share the host OS kernel, making them more efficient than virtual machines.

Images

- **Definition:** An image is a read-only template used to create containers. It includes the application code, libraries, and other dependencies.

- **How it works:** You build an image, then run a container based on that image.

Docker Engine

- The core component of Docker that runs and manages containers.
- It has two main parts:
    - **Server:** A long-running daemon process (`dockerd`) that manages containers.
    - **Client:** The command-line interface (`docker`) that users interact with.

Docker Hub & Registry

- **Docker Hub:** A cloud-based registry service where Docker images are stored and shared.
- **Registry:** A service for storing and distributing Docker images. Docker Hub is the default registry, but you can set up private registries.

---

3. How Docker Works

- **Dockerfile:** A script that contains instructions to assemble a Docker image. It defines the base image, application code, and dependencies.
- **Building Images:** Using the command `docker build`, Docker reads the Dockerfile and creates an image.
- **Running Containers:** Using `docker run`, Docker starts a container from an image.

---

4. Why Use Docker?

- **Portability:** Containers can run on any system that has Docker installed, regardless of the underlying hardware or OS.
- **Consistency:** The application runs the same way in development, testing, and production environments.

- **Isolation:** Each container is isolated from others, preventing conflicts between applications.
- **Efficiency:** Containers are lightweight, using fewer resources compared to virtual machines.

---

5. Basic Docker Commands

- `docker --version`: Check Docker version.
- `docker pull <image>`: Download an image from Docker Hub.
- `docker build -t <name> .`: Build an image from a Dockerfile.
- `docker run <image>`: Run a container from an image.
- `docker ps`: List running containers.
- `docker ps -a`: List all containers (including stopped ones).
- `docker stop <container_id>`: Stop a running container.
- `docker rm <container_id>`: Remove a container.
- `docker rmi <image_name>`: Remove an image.

---

6. Docker Compose

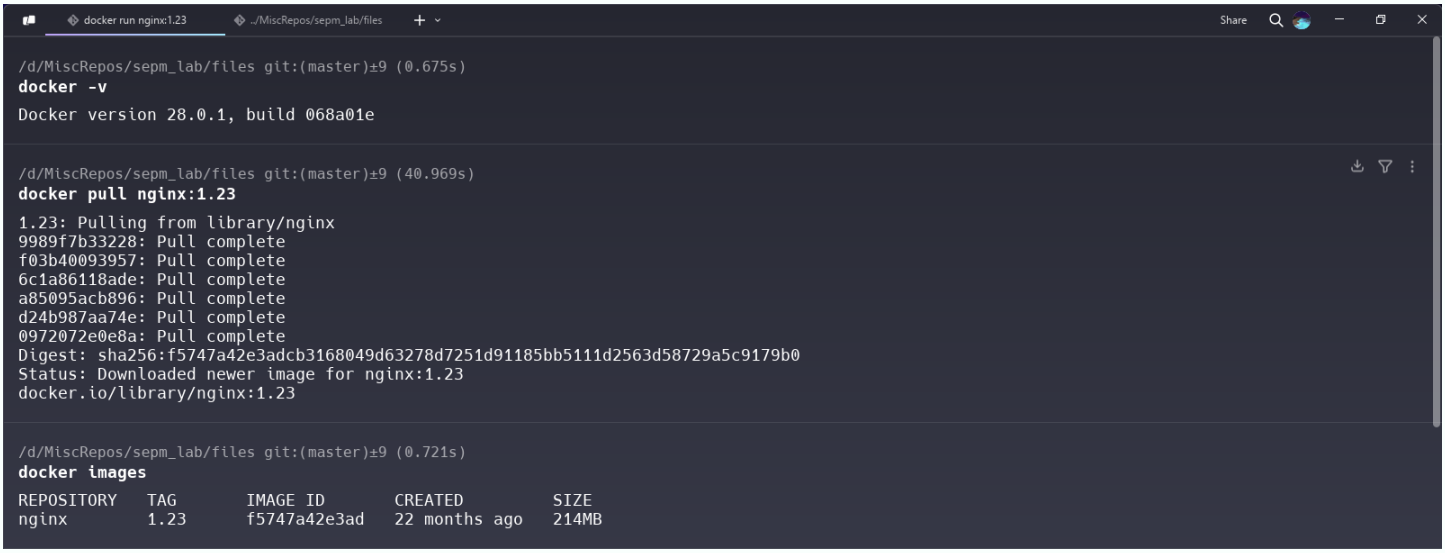
- A tool for defining and running multi-container Docker applications.
- Uses a YAML file (`docker-compose.yml`) to configure services, networks, and volumes.
- Command: `docker-compose up` to start all services defined in the file.

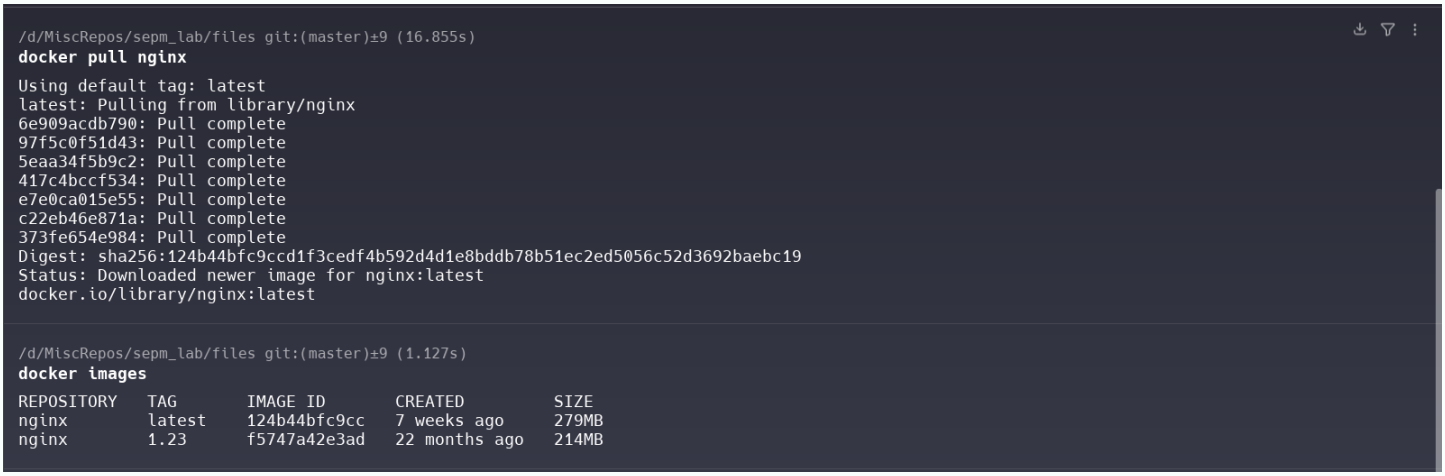
---

7. Real-World Use Cases

- **Microservices Architecture:** Running different microservices in isolated containers.
- **Continuous Integration/Continuous Deployment (CI/CD):** Automating testing and deployment processes.
- **DevOps and Infrastructure as Code:** Simplifying the deployment of complex applications.

Output :





```
/d/MiscRepos/sepm_lab/files git:(master)±9
docker run nginx:1.23

/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2025/04/02 13:34:51 [notice] 1#1: using the "epoll" event method
2025/04/02 13:34:51 [notice] 1#1: nginx/1.23.4
2025/04/02 13:34:51 [notice] 1#1: built by gcc 10.2.1 20210110 (Debian 10.2.1-6)
2025/04/02 13:34:51 [notice] 1#1: OS: Linux 5.15.167.4-microsoft-standard-WSL2
2025/04/02 13:34:51 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2025/04/02 13:34:51 [notice] 1#1: start worker processes
2025/04/02 13:34:51 [notice] 1#1: start worker process 29
2025/04/02 13:34:51 [notice] 1#1: start worker process 30
2025/04/02 13:34:51 [notice] 1#1: start worker process 31
2025/04/02 13:34:51 [notice] 1#1: start worker process 32
```

```
/d/MiscRepos/sepm_lab/files git:(master)±9 (0.712s)
docker ps
CONTAINER ID   IMAGE         COMMAND                  CREATED              STATUS              PORTS     NAMES
4ede9b099710   nginx:1.23    "/docker-entrypoint.…"   About a minute ago   Up About a minute   80/tcp    epic_mirzakhani

/d/MiscRepos/sepm_lab/files git:(master)±9
```

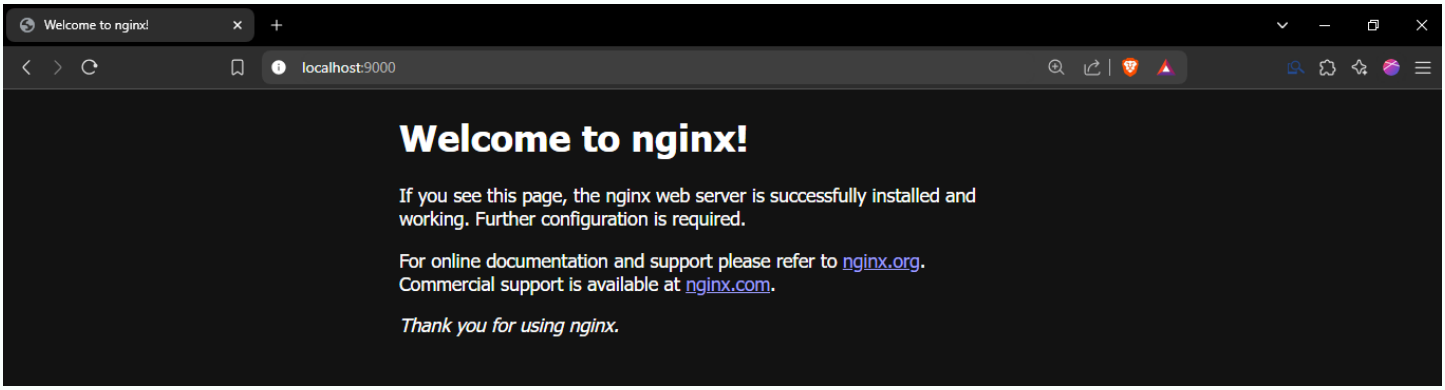
```
/d/MiscRepos/sepm_lab/files git:(master)±9 (2.813s)
docker run -d nginx:1.23
a8c8bd4b3b4912b37e3d407427e975a56fd722df027f0012e2c03c4199f77291

/d/MiscRepos/sepm_lab/files git:(master)±9 (0.784s)
docker ps
CONTAINER ID   IMAGE         COMMAND                  CREATED         STATUS         PORTS     NAMES
a8c8bd4b3b49   nginx:1.23    "/docker-entrypoint.…"   8 seconds ago   Up 6 seconds   80/tcp    amazing_jepsen

/d/MiscRepos/sepm_lab/files git:(master)±9 (0.647s)
docker logs a8c8bd4b3b49
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2025/04/02 13:40:12 [notice] 1#1: using the "epoll" event method
2025/04/02 13:40:12 [notice] 1#1: nginx/1.23.4
2025/04/02 13:40:12 [notice] 1#1: built by gcc 10.2.1 20210110 (Debian 10.2.1-6)
2025/04/02 13:40:12 [notice] 1#1: OS: Linux 5.15.167.4-microsoft-standard-WSL2
2025/04/02 13:40:12 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2025/04/02 13:40:12 [notice] 1#1: start worker processes
2025/04/02 13:40:12 [notice] 1#1: start worker process 29
2025/04/02 13:40:12 [notice] 1#1: start worker process 30
2025/04/02 13:40:12 [notice] 1#1: start worker process 31
2025/04/02 13:40:12 [notice] 1#1: start worker process 32

/d/MiscRepos/sepm_lab/files git:(master)±9
```

```
/d/MiscRepos/sepm_lab/files git:(master)±9
docker run nginx:latest

/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2025/04/02 13:43:21 [notice] 1#1: using the "epoll" event method
2025/04/02 13:43:21 [notice] 1#1: nginx/1.27.4
2025/04/02 13:43:21 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)
2025/04/02 13:43:21 [notice] 1#1: OS: Linux 5.15.167.4-microsoft-standard-WSL2
2025/04/02 13:43:21 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2025/04/02 13:43:21 [notice] 1#1: start worker processes
2025/04/02 13:43:21 [notice] 1#1: start worker process 29
2025/04/02 13:43:21 [notice] 1#1: start worker process 30
2025/04/02 13:43:21 [notice] 1#1: start worker process 31
2025/04/02 13:43:21 [notice] 1#1: start worker process 32
```

```
/d/MiscRepos/sepm_lab/files git:(master)±9 (0.668s)
docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS     NAMES
b59d07d59184   nginx:latest   "/docker-entrypoint.…"   11 seconds ago   Up 10 seconds   80/tcp    intelligent_beaver
a8c8bd4b3b49   nginx:1.23     "/docker-entrypoint.…"   3 minutes ago    Up 3 minutes    80/tcp    amazing_jepsen

/d/MiscRepos/sepm_lab/files  git:(master) ±9
```

```
/d/MiscRepos/sepm_lab/files git:(master)±9 (0.775s)
docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS     NAMES
a8c8bd4b3b49   nginx:1.23     "/docker-entrypoint.…"   6 minutes ago    Up 6 minutes    80/tcp    amazing_jepsen

/d/MiscRepos/sepm_lab/files git:(master)±9 (1.199s)
docker stop a8c8bd4b3b49
a8c8bd4b3b49

/d/MiscRepos/sepm_lab/files git:(master)±9 (0.668s)
docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES

/d/MiscRepos/sepm_lab/files  git:(master) ±9
```

```
/d/MiscRepos/sepm_lab/files git:(master)±9 (1.164s)
docker images
REPOSITORY   TAG       IMAGE ID       CREATED         SIZE
nginx        latest    124b44bfc9cc   7 weeks ago     279MB
nginx        1.23      f5747a42e3ad   22 months ago   214MB

/d/MiscRepos/sepm_lab/files git:(master)±9 (2.851s)
docker run -d -p 9000:80 nginx:1.23
e84b49b3ac88f8be69f82ce3fc6032c1294c29506389674099762f43d0c0fe93

/d/MiscRepos/sepm_lab/files git:(master)±9 (0.729s)
docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                    NAMES
e84b49b3ac88   nginx:1.23     "/docker-entrypoint.…"   17 seconds ago   Up 15 seconds   0.0.0.0:9000->80/tcp     happy_torvalds

/d/MiscRepos/sepm_lab/files  git:(master) ±9
```



```
/d/MiscRepos/sepm_lab/files git:(master)±9 (1.145s)
docker logs e84b49b3ac88
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2025/04/02 13:48:40 [notice] 1#1: using the "epoll" event method
2025/04/02 13:48:40 [notice] 1#1: nginx/1.23.4
2025/04/02 13:48:40 [notice] 1#1: built by gcc 10.2.1 20210110 (Debian 10.2.1-6)
2025/04/02 13:48:40 [notice] 1#1: OS: Linux 5.15.167.4-microsoft-standard-WSL2
2025/04/02 13:48:40 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2025/04/02 13:48:40 [notice] 1#1: start worker processes
2025/04/02 13:48:40 [notice] 1#1: start worker process 30
2025/04/02 13:48:40 [notice] 1#1: start worker process 31
2025/04/02 13:48:40 [notice] 1#1: start worker process 32
2025/04/02 13:48:40 [notice] 1#1: start worker process 33
172.17.0.1 - - [02/Apr/2025:13:49:09 +0000] "GET / HTTP/1.1" 200 615 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) C
hrome/134.0.0.0 Safari/537.36" "-"
2025/04/02 13:49:09 [error] 31#31: *1 open() "/usr/share/nginx/html/favicon.ico" failed (2: No such file or directory), client: 172.17.0.1, server: localhos
t, request: "GET /favicon.ico HTTP/1.1", host: "localhost:9000", referrer: "http://localhost:9000/"
172.17.0.1 - - [02/Apr/2025:13:49:09 +0000] "GET /favicon.ico HTTP/1.1" 404 555 "http://localhost:9000/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWeb
Kit/537.36 (KHTML, like Gecko) Chrome/134.0.0.0 Safari/537.36" "-"

/d/MiscRepos/sepm_lab/files git:(master)✓ ±9
```

```
/d/MiscRepos/sepm_lab/files git:(master)±9 (0.728s)
docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS                      PORTS                    NAMES
e84b49b3ac88   nginx:1.23     "/docker-entrypoint.…"   4 minutes ago    Up 4 minutes                0.0.0.0:9000->80/tcp     happy_torvalds
b59d07d59184   nginx:latest   "/docker-entrypoint.…"   10 minutes ago   Exited (0) 7 minutes ago                             intelligent_beaver
a8c8bd4b3b49   nginx:1.23     "/docker-entrypoint.…"   13 minutes ago   Exited (0) 6 minutes ago                             amazing_jepsen
4ede9b099710   nginx:1.23     "/docker-entrypoint.…"   18 minutes ago   Exited (0) 13 minutes ago                            epic_mirzakhani

/d/MiscRepos/sepm_lab/files git:(master)±9 (3.44s)
docker stop happy_torvalds
happy_torvalds

/d/MiscRepos/sepm_lab/files git:(master)±9 (2.124s)
docker run --name web_app -d -p 9000:80 nginx:1.23
7427673945ec2a4857141364b221bd3042ecec05d0ebd96141161e79aa81ee35

/d/MiscRepos/sepm_lab/files git:(master)±9 (0.789s)
docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                    NAMES
7427673945ec   nginx:1.23     "/docker-entrypoint.…"   5 seconds ago    Up 4 seconds    0.0.0.0:9000->80/tcp     web_app

/d/MiscRepos/sepm_lab/files git:(master)✓ ±9
```

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.