

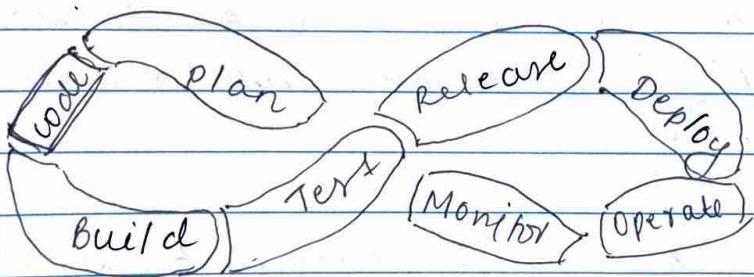
Experiment 1

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. It's 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/organisation.



Build: without DevOps, the cost of the communication 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 is 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 or 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 clean the descriptive process, which gives clarity on product development & delivery.
- It improves customer experience & satisfaction.

Disadvantages :

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

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

Experiment 02

Aim: To study and implement making a GIT account.

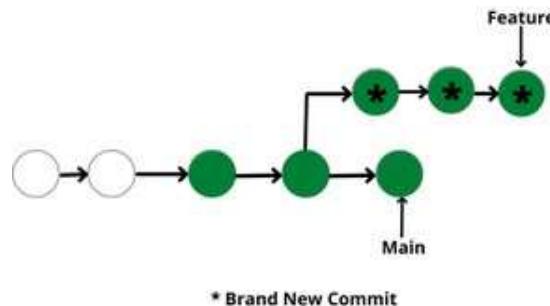
Theory:

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:

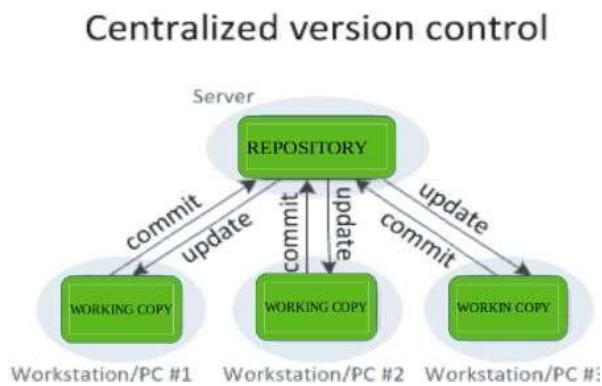
1. Local Version Control System
2. Centralized Version Control System
3. Distributed Version Control System

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



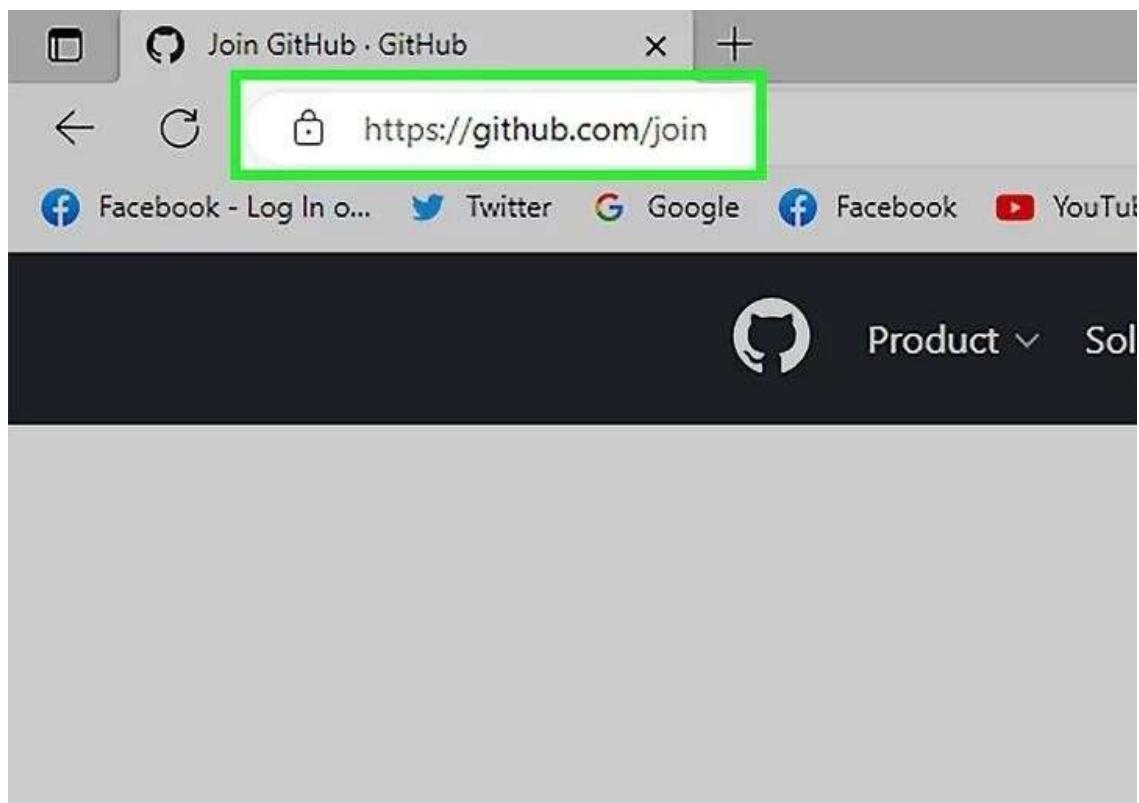
Signing up for a new personal account

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.

The screenshot shows the "First, let's create your user account" step of the GitHub sign-up process. A large green box highlights the input fields for Username, Email address, and Password. The "Username" field contains "wikihowneveconcepts" with a green checkmark. The "Email address" field is empty. The "Password" field contains "*****" with a green checkmark. Below these fields is a note: "Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter." The "Email preferences" section includes a checked checkbox for "Send me occasional product updates, announcements, and offers." At the bottom, there is a "Verify your account" button.

Join GitHub

First, let's create your user account

Username *

 ✓

Email address *

Password *

 ✓

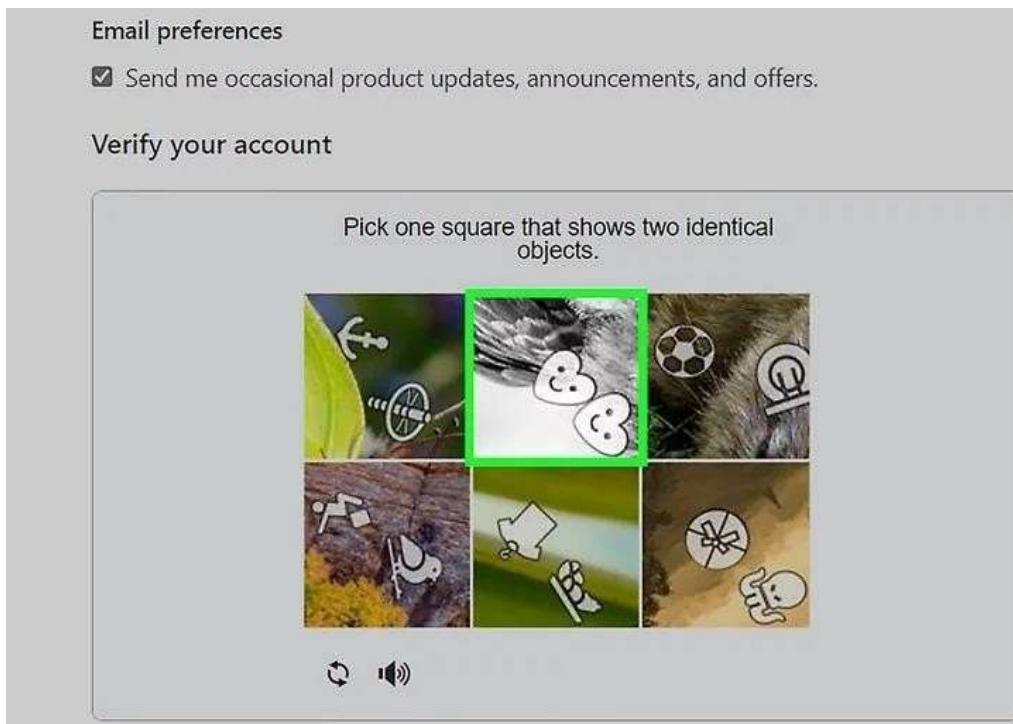
Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter.
[Learn more](#)

Email preferences

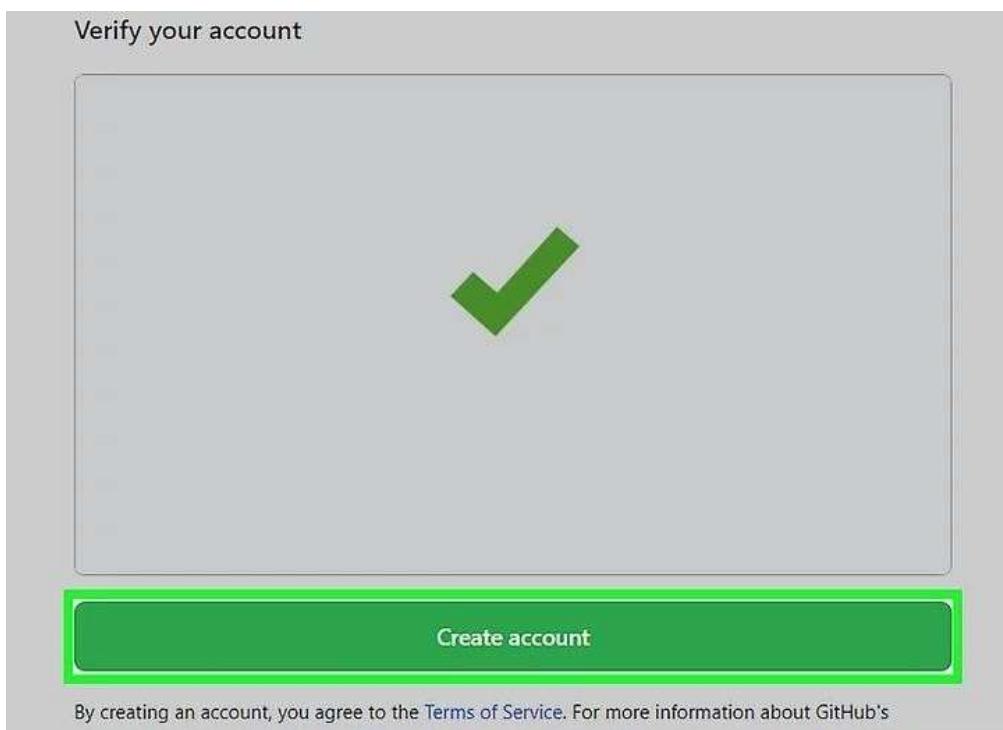
Send me occasional product updates, announcements, and offers.

Verify your account

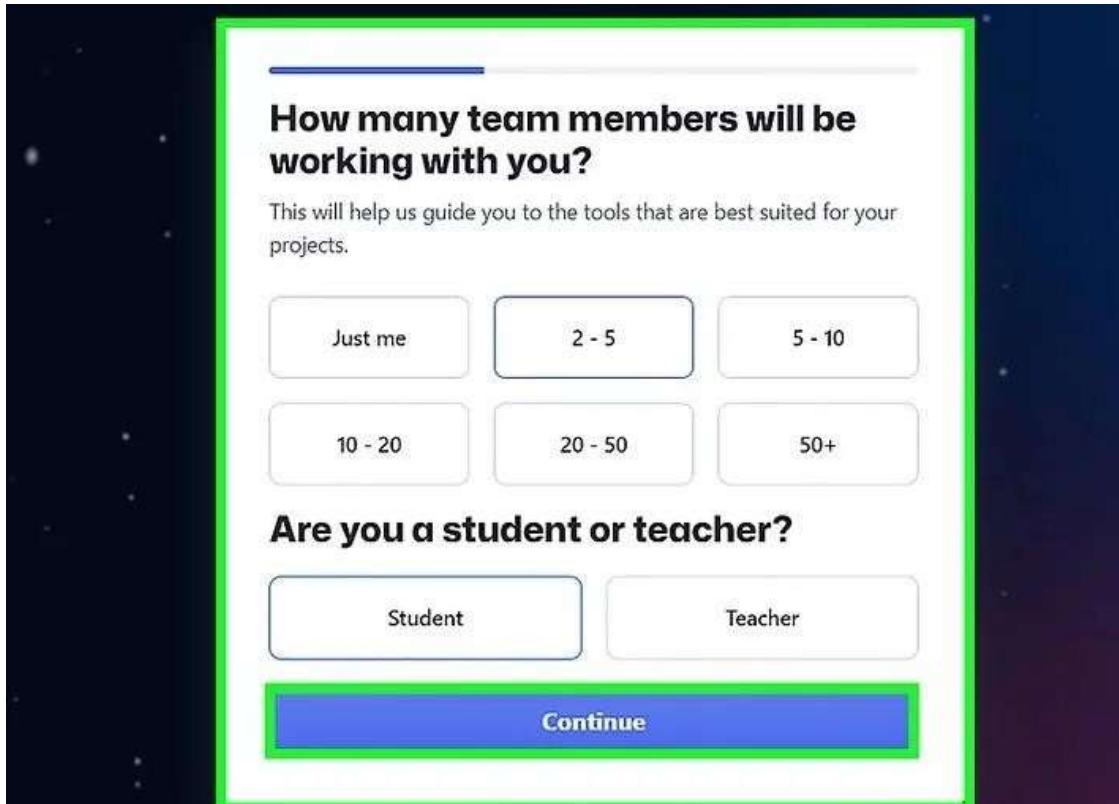
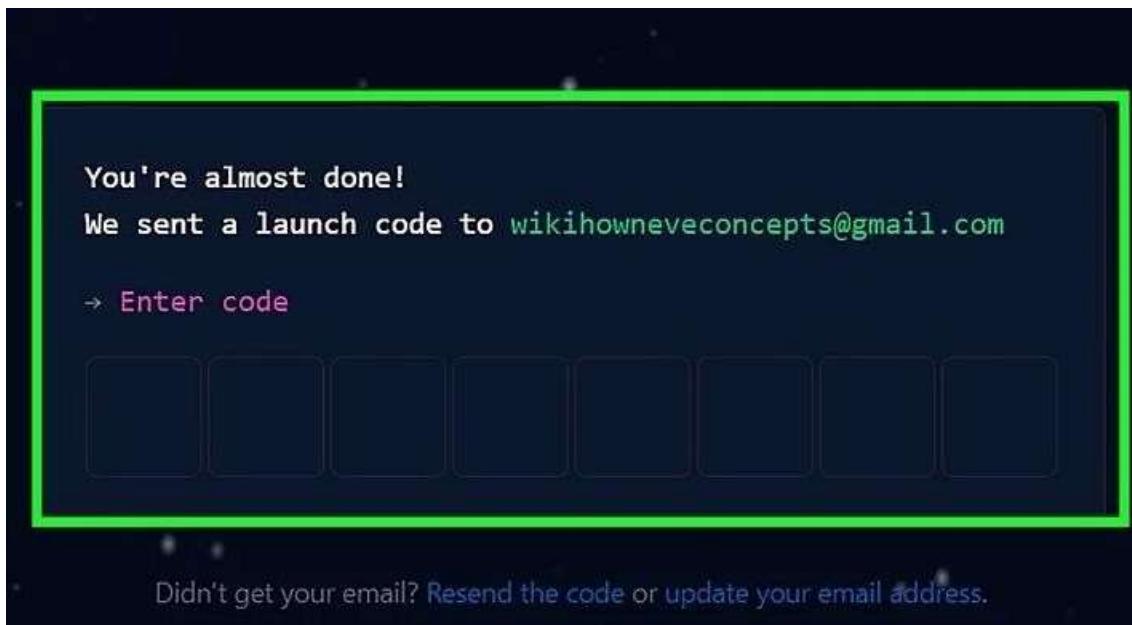
Step 03: Click Verify to start the verification puzzle. The instructions vary by puzzle, so just follow the on-screen instructions to confirm that you are a human. A green checkmark will appear after completing the puzzle.



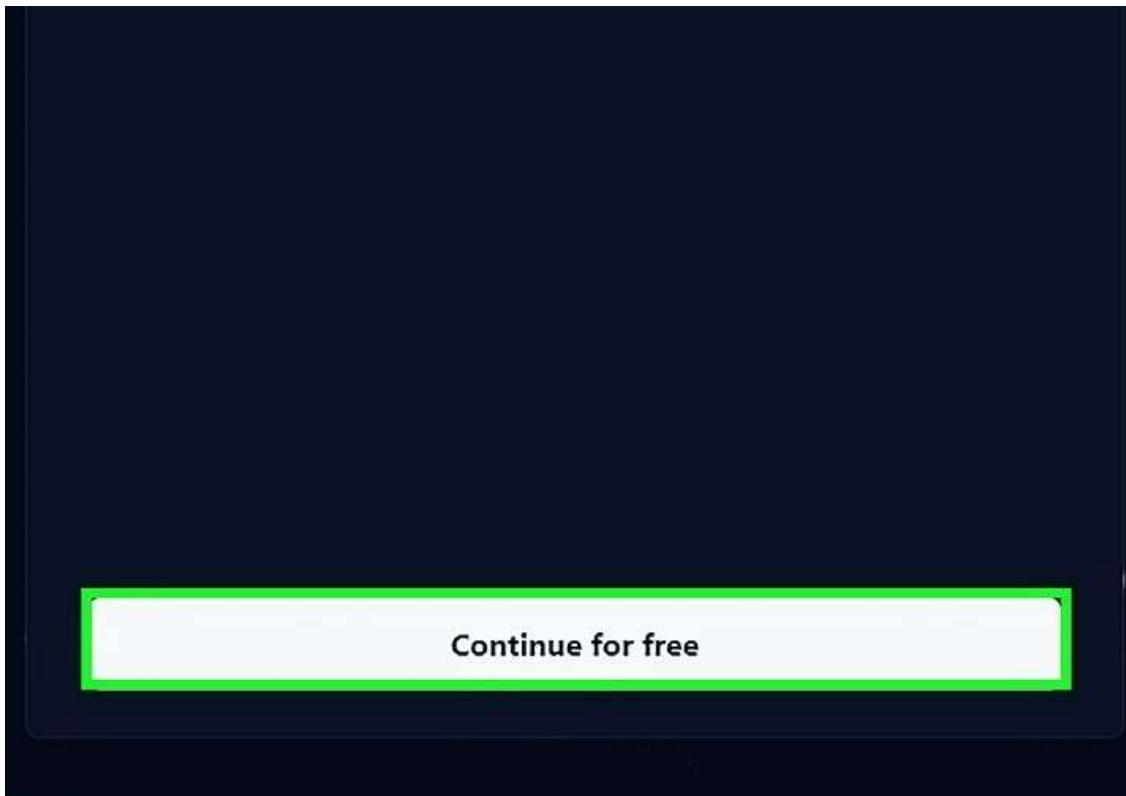
Step 04: Click the green Create account button. It's below the form, at the bottom of the page. This will take you to an email verification page.



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.

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.

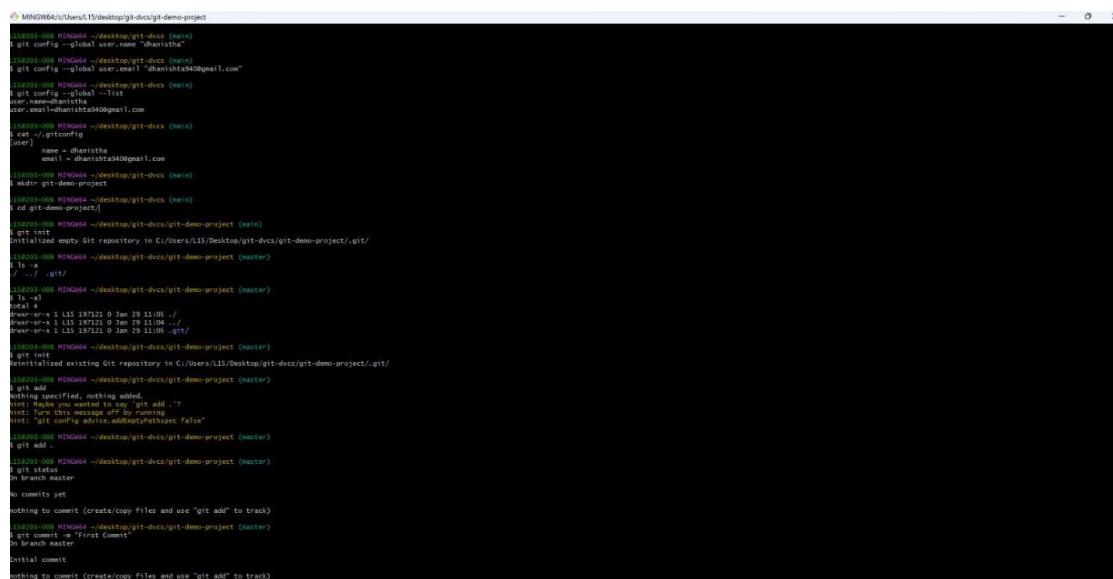
The `git clone` command is used to create a copy of a Git repository. When you run this command, it duplicates the entire repository, including its files, commit history, and branches, and downloads it to your local machine. This is often the initial step when you want to work with a project hosted on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



```

$ MINOW64/c/Users/l15/Desktop/git-dvcs/git-demo-project
$ git config --global user.name "dhanistha"
$ git config --global user.email "dhanistha94@gmail.com"
$ cd git-demo-project
$ git init
$ git add .
$ git commit -m "First Commit"
$ git status
On branch master
nothing to commit (create/copy files and use "git add" to track)
$ git commit -m "First Commit"
$ git commit -m "First Commit"
$ git status
On branch master
nothing to commit (create/copy files and use "git add" to track)

```

```

11:59:03 ~ 008 MINIG64 ~/desktop/git-dvcs/git-demo-project (master)
$ nano index.html
11:59:03 ~ 008 MINIG64 ~/desktop/git-dvcs/git-demo-project (master)
$ git status
On branch master
nothing to commit, working directory clean
11:59:03 ~ 008 MINIG64 ~/desktop/git-dvcs/git-demo-project (master)
$ git add index.html
warning: the working copy of 'index.html', LF will be replaced by CRLF the next time Git touches it
11:59:03 ~ 008 MINIG64 ~/desktop/git-dvcs/git-demo-project (master)
$ git commit -m "First Commit"
[master 09d635d] First Commit
 1 file changed, 1 insertion(+)
 create mode 100644 index.html
11:59:03 ~ 008 MINIG64 ~/desktop/git-dvcs/git-demo-project (master)
$ git log
commit 09d635dcf0731a843cf02a20e3993cf0c02 (HEAD -> master)
Author: Nirjara Soni <nirjara.soni94@gmail.com>
Date:   wed Jan 29 11:12:16 2023 +0530

First Commit
11:59:03 ~ 008 MINIG64 ~/desktop/git-dvcs/git-demo-project (master)
$ git log --oneline
09d635d (HEAD -> master) First Commit
11:59:03 ~ 008 MINIG64 ~/desktop/git-dvcs/git-demo-project (master)
$ git log --oneline
09d635d (HEAD -> master) First Commit
11:59:03 ~ 008 MINIG64 ~/desktop/git-dvcs/git-demo-project (master)
$ |
11:59:03 ~ 008 MINIG64 ~/desktop/git-dvcs/git-demo-project (master)
$ git clone https://github.com/dhanishthajeswani1910/python-april23.git
Cloning into 'python-april23'...
remote: Counting objects: 1000 (15/15), done.
remote: Compressing objects: 1000 (15/15), done.
remote: Writing objects: 1000 (15/15), done.
Received objects: 1000 (15/15), done.
Resolving deltas: 1000 (15/15), done.
11:59:03 ~ 008 MINIG64 ~/desktop/git-dvcs/git-demo-project (master)
$ Index.html python-april23/
11:59:03 ~ 008 MINIG64 ~/desktop/git-dvcs/git-demo-project (master)
$ git remote add origin https://github.com/dhanishthajeswani1910/python-april23.git
11:59:03 ~ 008 MINIG64 ~/desktop/git-dvcs/git-demo-project (master)
$ git fetch origin
remote: origin
Fetch URL: https://github.com/dhanishthajeswani1910/python-april23.git
HEAD branch: main
HEAD commit: 9a2a330333333333333333333333333333333333
remote: main new (Next fetch will store in remotes/origin)
11:59:03 ~ 008 MINIG64 ~/desktop/git-dvcs/git-demo-project (master)
$ git pull
remote: Unpacking objects: 15, done.
remote: Counting objects: 1000 (15/15), done.
remote: Compressing objects: 1000 (15/15), done.
remote: Writing objects: 1000 (15/15), done.
remote: Total 1000 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 1000 (15/15), 3.93 Kib | 251.00 KiB/s, done.
From https://github.com/dhanishthajeswani1910/python-april23
 * [new branch]      main      -> origin/main
There is no tracking information for the current branch.
You can set tracking information for this branch you want to merge with.
See git-pull(1) for details.
git pull <remote> <branch>

If you wish to set tracking information for this branch you can do so with:
  git branch --set-upstream-to=<origin/>/<branch> master

11:59:03 ~ 008 MINIG64 ~/desktop/git-dvcs/git-demo-project (master)
$ git fetch
11:59:03 ~ 008 MINIG64 ~/desktop/git-dvcs/git-demo-project (master)
$ |

```

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

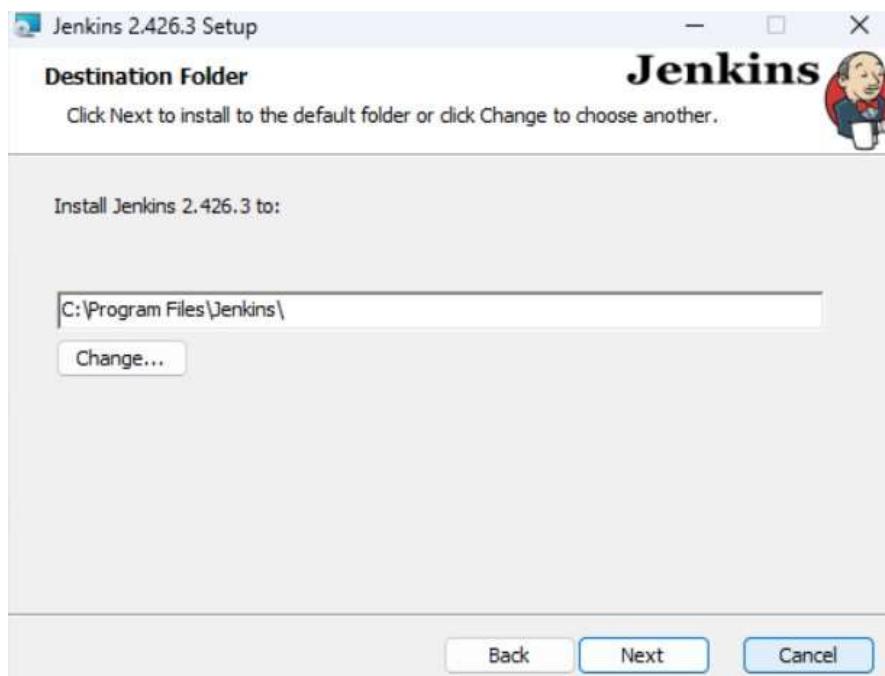
Jenkins is an open source automation tool written in Java with plugins built for Continuous Integration purpose. Jenkins is used to build and test your software projects continuously making it easier for developers to integrate changes to the project, and making it easier for users to obtain a fresh 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.

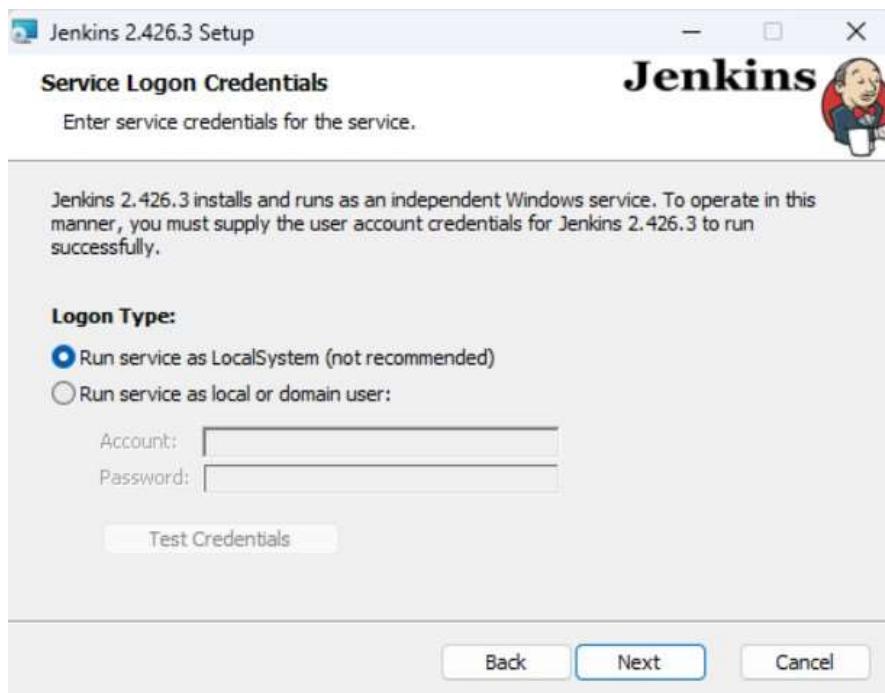
Open the installed .exe setup



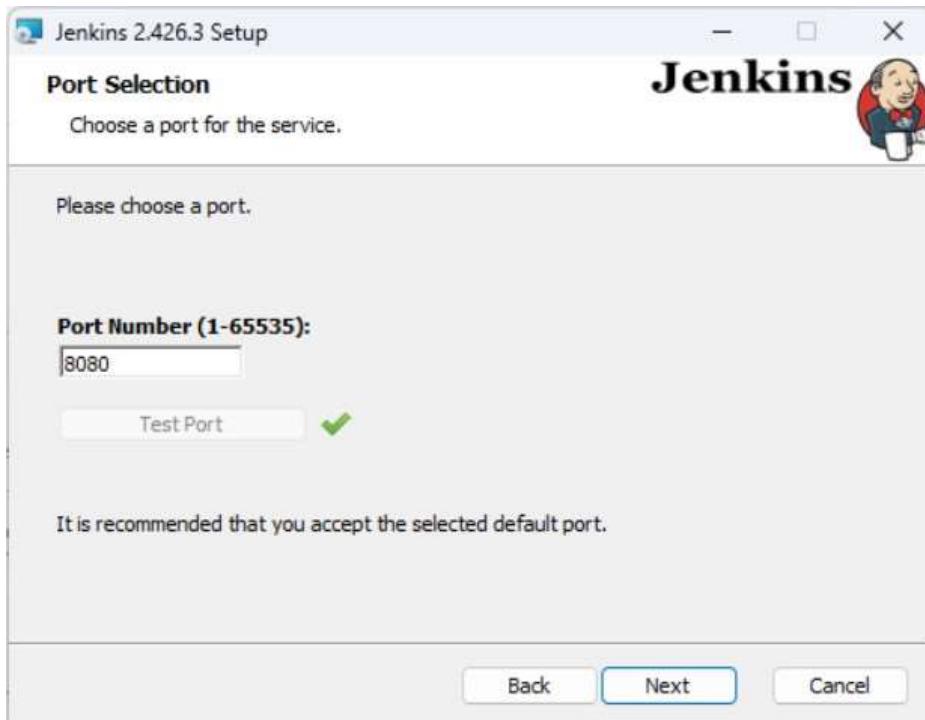
Step 2: Locate the folder where you want to install Jenkins in the location path:



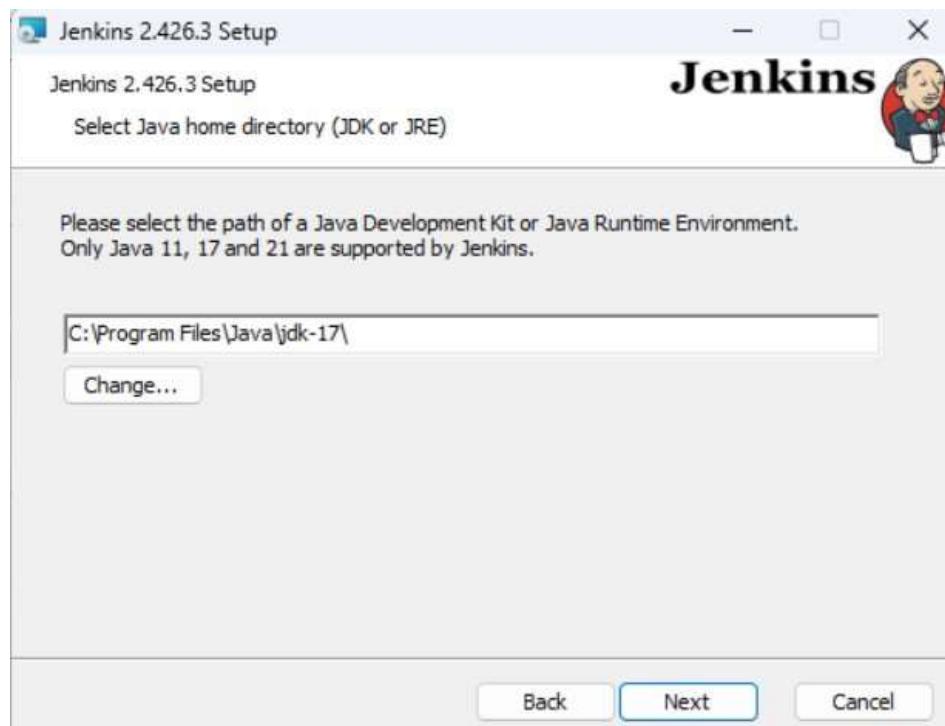
Step 3: Select service as Local System and proceed to Next.



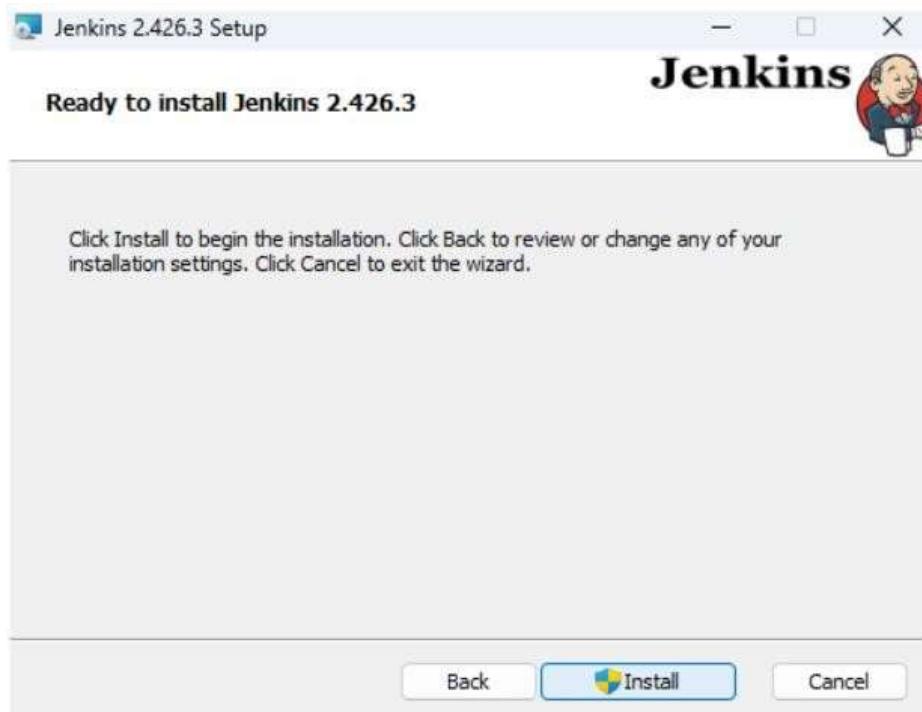
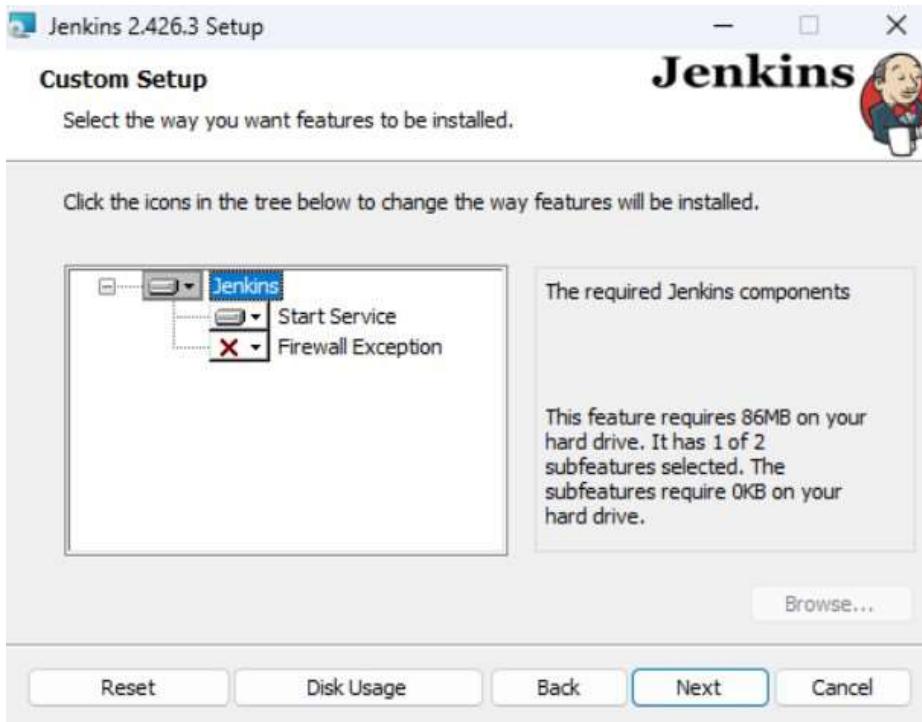
Step 4: Select the port 8080 and click Test Port button. The green tick will appear after which you can proceed to Next.



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.

Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

`C:\ProgramData\Jenkins\.jenkins\secrets\initialAdminPassword`

Please copy the password from either location and paste it below.

Administrator password:

[Continue](#)

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

The screenshot shows the 'Customize Jenkins' section of the Jenkins setup wizard. It features two main options:

- Install suggested plugins**: A light blue button with the text "Install plugins the Jenkins community finds most useful." Below it.
- Select plugins to install**: A grey button with the text "Select and install plugins most suitable for your needs."

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.

The screenshot shows the 'Getting Started' page of Jenkins. It displays a table of installed plugins and a detailed list of Jenkins APIs.

✓ Folders	✓ OWASP Markup Formatter	✓ Build Timeout	✓ Credentials Binding
✓ Timestamper	⌚ Workspace Cleanup	⌚ Ant	⌚ Gradle
⌚ Pipeline	⌚ GitHub Branch Source	⌚ Pipeline: GitHub Groovy Libraries	⌚ Pipeline: Stage View
⌚ Git	⌚ SSH Build Agents	⌚ Matrix Authorization Strategy	⌚ PAM Authentication
⌚ LDAP	⌚ Email Extension	⌚ Mailer	

To the right of the table, a large scrollable panel displays the Jenkins API documentation. The visible portion of the text includes:

```
** - required dependency
** Jenkins API
** Institute Identity
** JavaBeans Activation Framework (JAF) API
** JavaMail API
** Credentials
** Plain Credentials
** Gson API
** Trilead API
** SSH Credentials
** SCM API
** Pipeline: API
commons-lang3 v3.x Jenkins API
Timestamper
** Cache API
** Script Security
** JAXB
** SnakeYAML API
** Jackson 2 API
** commons-text API
** Pipeline: Supporting APIs
** Plugin Utilities API
** Font Awesome API
** Bootstrap 5 API
** JQuery3 API
```

The screenshot shows the Jenkins dashboard. On the left, there's a sidebar with links for 'Dashboard', '+ New Item', 'People', 'Build History', 'Manage Jenkins', and 'My Views'. Below these are sections for 'Build Queue' (No builds in the queue) and 'Build Executor Status' (1 idle, 2 idle). The main content area has a heading 'Welcome to Jenkins!' with a sub-section 'Start building your software project' containing 'Create a job', 'Set up a distributed build', 'Set up an agent', 'Configure a cloud', and 'Learn more about distributed builds'. At the bottom right, it says 'REST API' and 'Jenkins 2.426.3'.

Getting Started

Create First Admin User

Username: musk

Password: [REDACTED]

Confirm password: [REDACTED]

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
This is the central feature of Jenkins. Jenkins will build your project, containing any SCM with any build system, and this can be even used for something other than software build.

Pipeline
Creates an imperative pipeline that can span multiple build agents, suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in Freestyle job type.

Multi-configuration project
Useful for projects that need a large number of different configurations, such as testing on multiple environments, platform specific builds, etc.

Folder
Creates a container that stores nested items in it, useful for grouping things together. Unlike a view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

Multibranch Pipeline
Creates a set of Pipeline projects according to detected branches in one SCM repository.

Organization Folder
Creates a set of multibranch project subfolders by scanning for repositories.

OK

Dashboard > example 1 > Configuration

Configure

Add timestamps to the Console Output
 Inspect build log for published build scans
 Terminate a build if it's stuck
 With Ant

Build Environment

Build Steps

Execute Windows batch command

Command:

See the list of available environment variables

```
echo "Hello, Jenkins!"
```

Advanced ▾

Add build step ▾

Save **Apply**

 Jenkins

Search (CTRL+K) Help Log Out

 Jenkins

Search (CTRL+K) Help Log Out

Dashboard > example 1 > #11

#11 (Jan 31, 2024, 9:07:13 AM)

Status Keep this build forever

Changes Add description Started 24 sec ago

Console Output Tracked 0.03 sec

Edit Build Information

Delete build 'P1'

Previous Build

</> No changes.

 Started by user Muskan Tolani

The screenshot shows the Jenkins interface for a build named 'example 1'. The 'Console Output' tab is selected. The output window displays the following log entries:

```
Started by user Muskan Tolani
Running at 3:57PM
Building in workspace C:\ProgramData\Jenkins\jenkins\workspace\example 1
[example 1] $ cmd /c call C:\Windows\TEMP\jenkins321068505410401591.bat

C:\ProgramData\Jenkins\jenkins\workspace\example 1>echo "Hello Team"
"Hello Team"

C:\ProgramData\Jenkins\jenkins\workspace\example 1>exit 0
Finished: SUCCESS
```

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.

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:

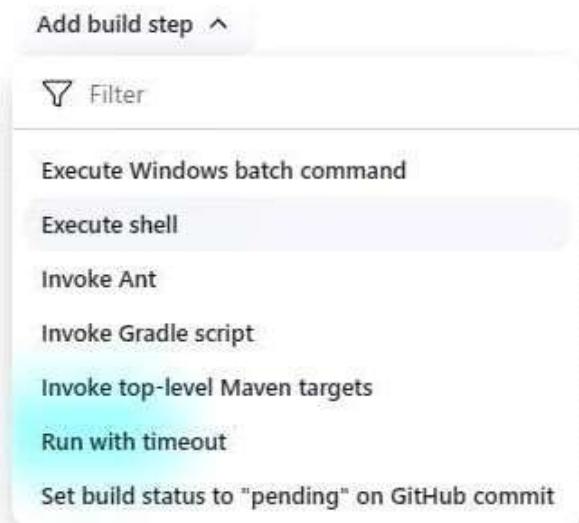
Start building your software project



Naming the job and setting it as freestyle:

Selecting build type as “Execute shell”:

Build Steps

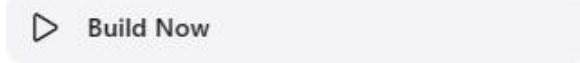


Entering a simple command for the shell execution:

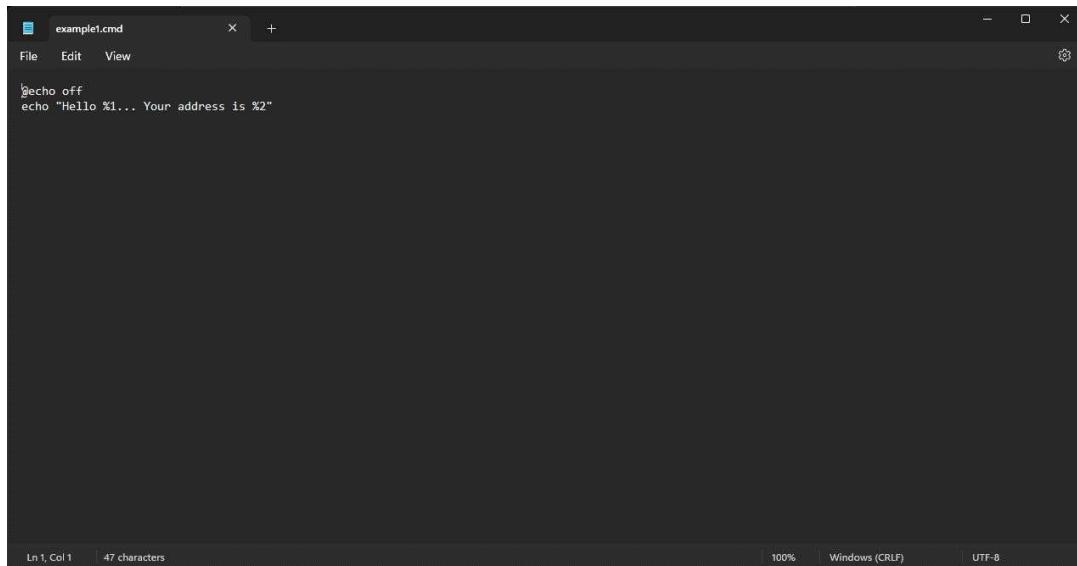
Applying and saving the project configuration:



Building the project:



Console output (after building):



The screenshot shows a Windows Notepad window with the title bar 'example1.cmd'. The menu bar includes 'File', 'Edit', and 'View'. The main content area contains the following text:

```
echo off
echo "Hello %1... Your address is %2"
```

The status bar at the bottom indicates 'Ln 1, Col 1' and '47 characters' on the left, '100%' and 'Windows (CRLF)' in the center, and 'UTF-8' on the right.

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 10.0.22621.3155] (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.cmd Tanishq
The system cannot find the path specified.

C:\Users\AI&DS 202>"Hello Tanishq... Your address is "
'"Hello Tanishq... 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 Girmaon "Hello 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:

Build Steps



Console output after building the modified project:



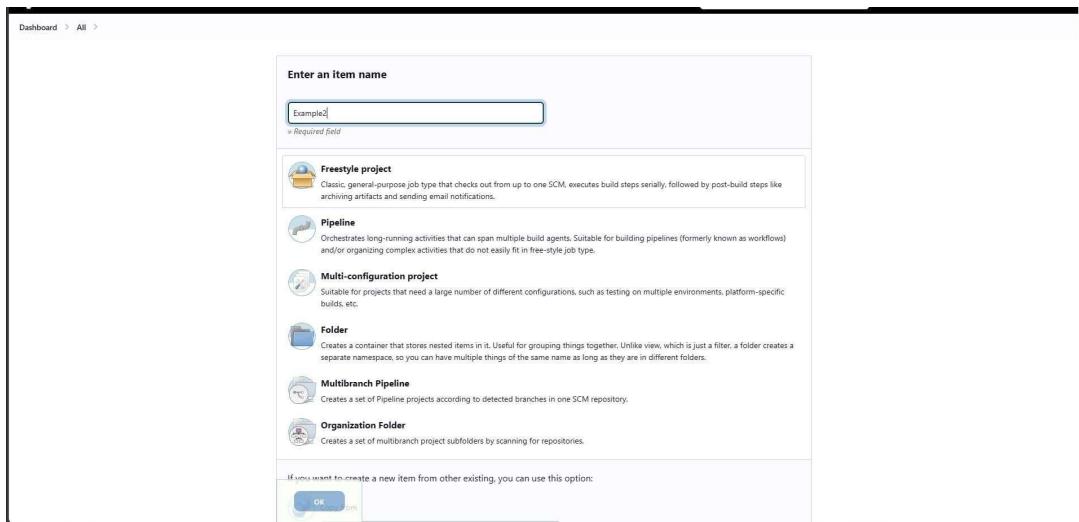
Running a Java program under Jenkins

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

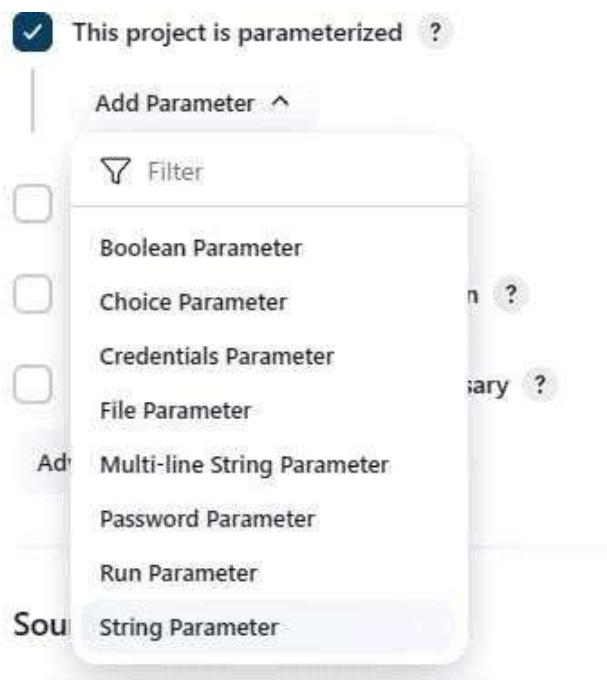
C:\ProgramData\Jenkins\.jenkins\workspace\24_34_31_45_40_42_37_41>java C:\Users\richminds\Desktop\sepm\24.java
This is T12

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

Example 3Example 3.1: Parameterise

build Creating a new freestyle project:

Enabling parameterisation and adding a String parameter:



Configuring the string parameter as Fname:

The screenshot shows the Jenkins 'String Parameter' configuration dialog for a parameter named 'Fname'. The dialog includes fields for 'Name' (containing 'Fname'), 'Default Value' (empty), 'Description' (empty), and a 'Plain text' preview area. A checkbox for 'Trim the string' is also present at the bottom.

Name	Fname
Default Value	
Description	
Plain text Preview	
<input type="checkbox"/> Trim the string	

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



Configuring build steps:



Entering parameters for build:



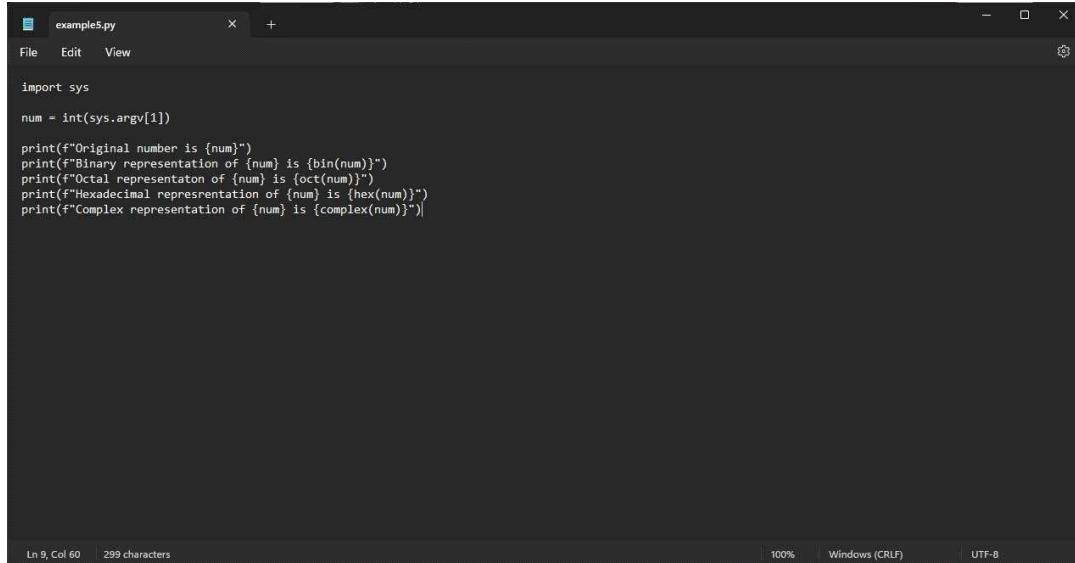
Console output after building:

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\jenkins14094536165150986151.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



A screenshot of a code editor window titled "example5.py". The window has a dark theme with light-colored text. The code in the editor is:

```
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 representation of {num} is {oct(num)}")
print(f"Hexadecimal representation 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
umber is 10
C:\Users\richminds\Desktop\sepm>
```

Creating a new freestyle project:

Enter an item name

Example5
» Required field

Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

Multibranch Pipeline
Creates a set of Pipeline projects according to detected branches in one SCM repository.

Organization Folder
Creates a set of multibranch project subfolders by scanning for repositories.

If you want to create a new item from other existing, you can use this option:

OK

Parameterising the project with a string parameter as follows:

This project is parameterized ?

String Parameter ?

Name ?
num

Default Value ?

Description ?

Plain text: [Preview](#)

Trim the string ?

Add Parameter ▾

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.

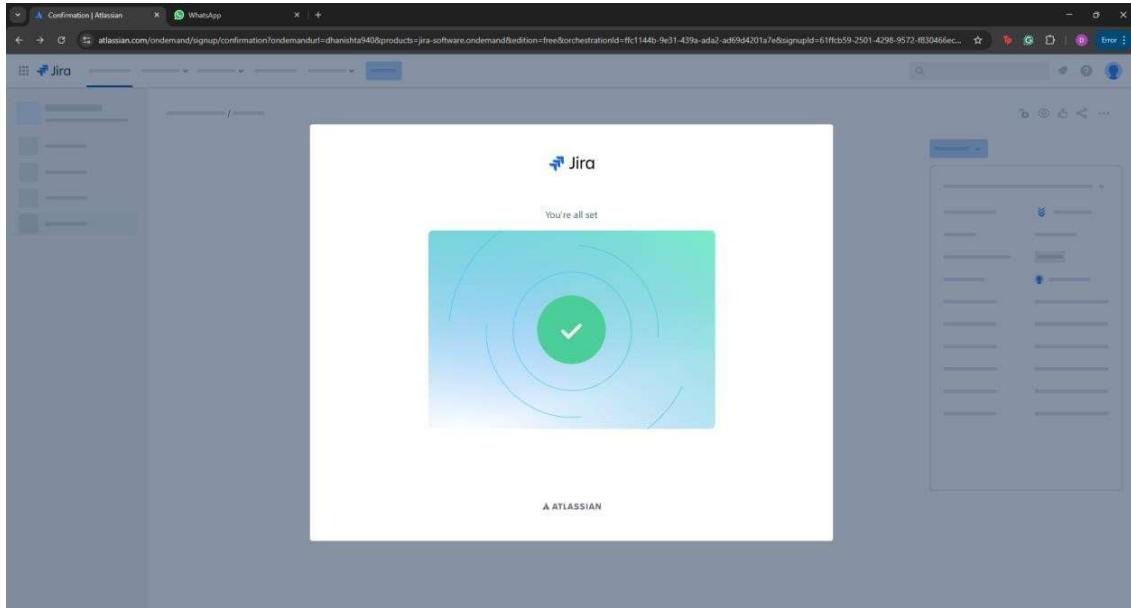
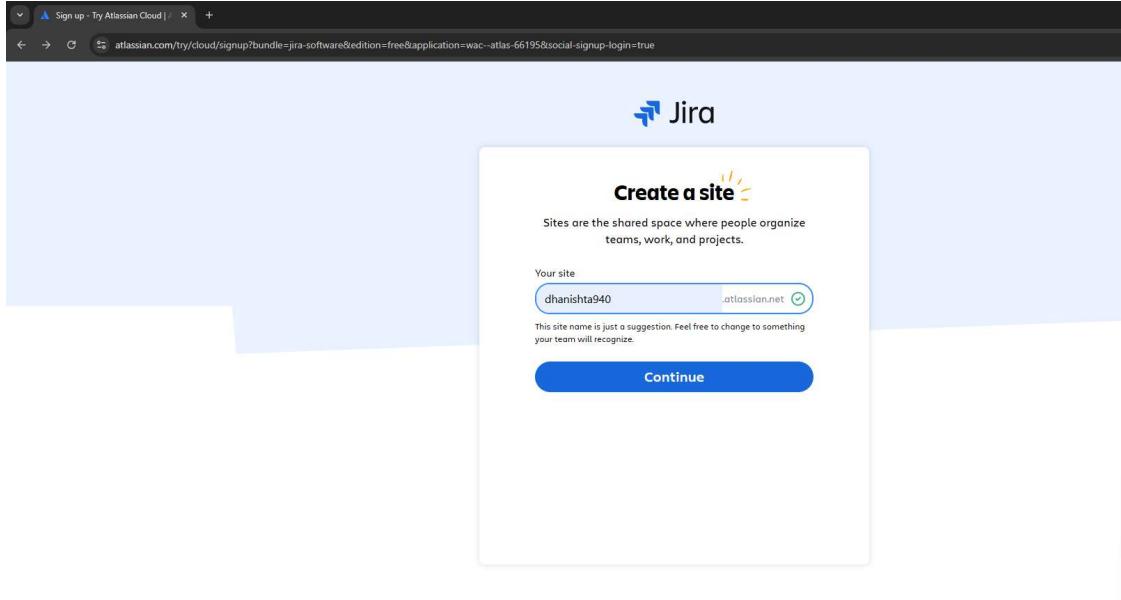
Reporting and Analytics: JIRA offers robust reporting and analytics capabilities, allowing teams to gain insights into their project progress, team performance, and other key metrics. Users can create custom dashboards, charts, and reports to monitor and analyse project data.

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 altlassian.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

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.



AGILE
A simple board to visualize your workflow.
A flexible and efficient way to manage agile work, with a board and timeline.

POPULAR



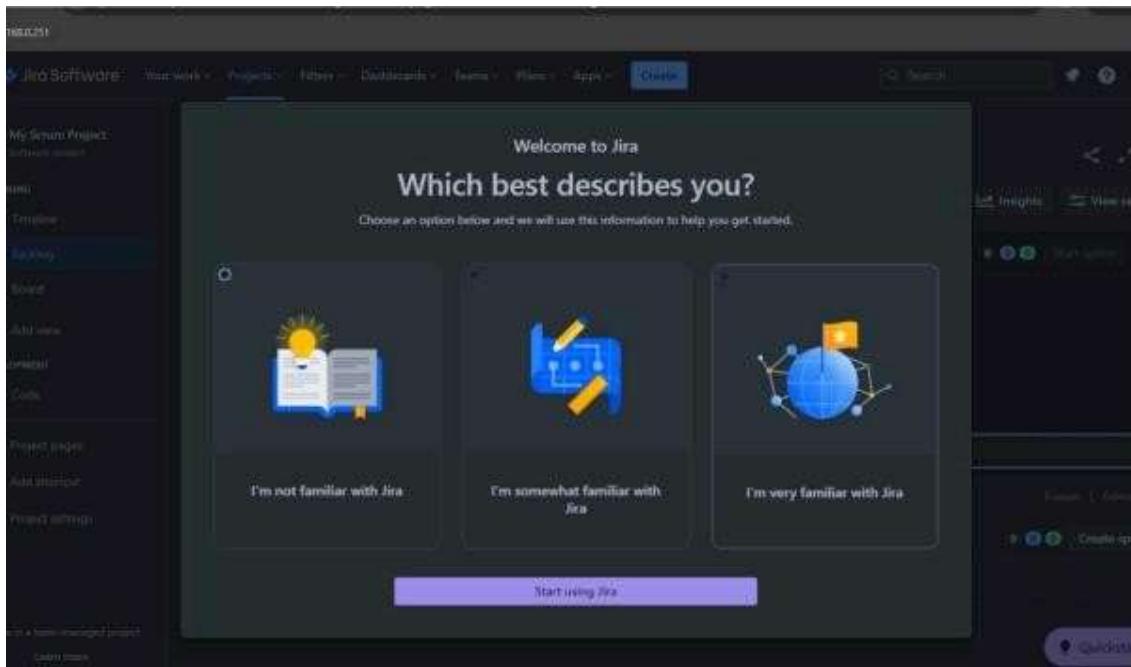
SCRUM
Deliver work in short, repeatable time blocks.
Plan, prioritize and schedule work using agile sprints with a backlog, board and timeline.



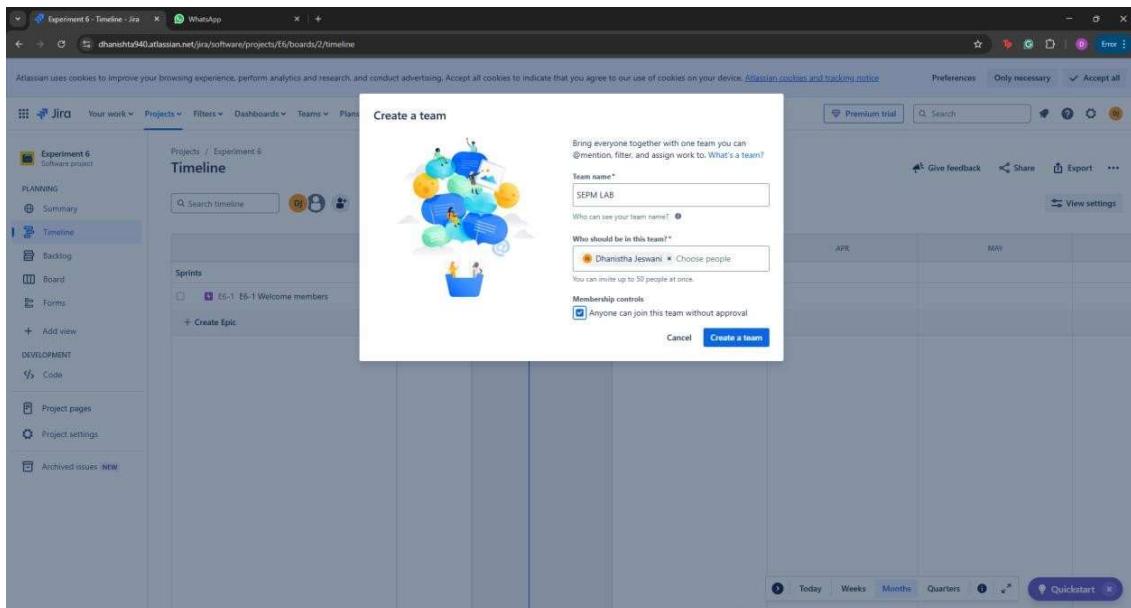
BUSINESS
Collaboration for business teams
Designed to help your business team collaborate and connect, with list and calendar views, forms, and more.

Jira Software

Jira Work Management



- Creating a team



- Adding members to the team

The screenshot shows the Jira Team page for the 'SEPM LAB' team. At the top, there's a message: '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.' Below this, the navigation bar includes 'Jira', 'Your work', 'Projects', 'Filters', 'Dashboards', 'Teams', 'Plans', 'Apps', and 'Create'. A 'Premium trial' button and a search bar are also present.

The main content area displays the 'Team activity' section with a message: 'E6-1 Welcome members Experiment 6 - You created this today'. It also shows 'Jira issues assigned to team' and 'Software components NEW'. On the left, there are sections for 'Members' (3 members: Dhanitha Jeswani, Arya, Khozaima Hammad) and 'Agents' (0 members). A notification box at the bottom left says: 'You've added new people to the team!'. The background has a light purple gradient.

- Creating a project

The screenshot shows the 'Add project details' dialog in Jira. At the top, it says 'Your work - Jira' and 'dhansitha940.atlassian.net/jira/your-work'. Below that is a link to 'Back to project types'. The main form has fields for 'Name*' (Experiment 6), 'Key *' (E6), and 'Access*' (Private). To the right, there are sections for 'Template' (Scrum, Jira) and 'Type' (Team-managed). Buttons for 'Cancel' and 'Create project' are at the bottom.

- Adding people to the project

The screenshot shows the Jira Scrum Board interface for a project named 'Experiment 6'. A modal window titled 'Add people to Experiment 6' is open, prompting for names or emails. The input field contains 'arya7311'. Below it, an email suggestion 'arya7311@gmail.com' is shown with the placeholder 'Select an email address'. A dropdown menu for 'Role' is set to 'Member'. At the bottom right of the modal are 'Cancel' and 'Add' buttons.

- Adding and assigning tasks

The screenshot shows the Jira Timeline interface for the same 'Experiment 6' project. On the left, a sidebar lists project navigation options like Planning, Backlog, Board, and Timeline. The main area displays a Gantt chart for February, March, and April. Sprints are listed on the left: 'E6-1 E6-1 Welcome members' (IN PROGRESS), 'E6-2 SEPM' (IN PROGRESS), 'E6-3 Working like a fire', 'E6-4 Security Check', 'E6-5 E6-5 Module Check', 'E6-6 E6-6 System Check', and 'E6-7 E6-7 Update System'. A 'Quickstart' sidebar on the right provides links to project creation, goal mapping, and basic Jira features.

The screenshot shows the Jira Timeline interface for the 'Experiment 6' project. The main view displays a horizontal timeline for February (FEB) with several tasks and epics assigned to different sprints. One epic, 'E6-1 Welcome members', is currently in progress. The interface includes a sidebar for planning and development, and a quickstart guide on the right.

- Task completion

The screenshot shows the Jira Timeline interface for the 'Experiment 6' project. The main view displays a horizontal timeline for February (FEB) with several tasks and epics assigned to different sprints. The task 'E6-1 Welcome members' has been completed, as indicated by the 'DONE' status in the timeline. The interface includes a sidebar for planning and development, and a quickstart guide on the right.

The screenshot shows the Jira Timeline view for a project named "Experiment 6". The left sidebar includes links for PLANNING (Summary, Backlog, Board, Forms, Add view), DEVELOPMENT (Code, Project pages, Project settings), and Archived issues (NEW). The main area displays a timeline from Today to Months, showing several sprints. Sprint 1 (E6-1) is labeled "Welcome members" and is marked as "DONE". Sprint 2 (E6-2) is labeled "SEPM" and is also marked as "DONE". Other sprints listed include E6-3 "Working like a fire", E6-4 "Security Check", E6-5 "Module Check", E6-6 "System Check", and E6-7 "Update System", all of which are also marked as "DONE". A modal window for issue E6-2 "SEPM" is open, showing details like assignee (Khozaima Hammad) and pinned fields. A "Quickstart" sidebar on the right provides links to create a project, map goals, identify work chunks, monitor risk, and more.

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

Experiment 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.



A screenshot of the Atlassian Admin interface. The top navigation bar shows "ATLASSIAN Admin" and the current site "adityadikonda1711". The "Settings" tab is selected. On the left, a sidebar lists various admin sections like Details, Domains, API keys, Application tunnels, Emails, Contacts, Compliance (marked as NEW), Platform experiences (marked as BETA), Data management, AI Settings, Rovo, Atlassian Intelligence, Migration to cloud, and Product links. The "Platform experiences" section is currently active, displaying the heading "Platform experiences". It states: "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." Below this, a table lists one site entry: "https://adityadikonda1711.atlassian.net" activated on "Mar 25, 2025" with an "Opt out" button. At the bottom, there are "Previous" and "Next" navigation buttons and a "Results per page: 5" dropdown.

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](#)

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](#)

Your backlog is empty.

[Create issue](#)

0 issues | Estimate

Start sprint

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](#)

Projects

Name	Key	Type	Lead	Project URL	More actions
Experiment 5	SCRUM	Team-managed software	Aditya Dikonda		...

1

Create project Templates

Templates

Preview a template for your next project

- Scrum** Deliver work in short time blocks
- Product roadmap** Create custom roadmaps
- Work requests** Manage unplanned work
- Top-level planning** Monitor work across your teams

More templates

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](#)

Add epic / SCRUM-1

Choose a topic

+ Add @ Apps

Description
Add a description...

Activity
Show: All Comments History Work log

Add a comment... Can I get more info...? Status update... Thanks...

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

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](#)

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](#)

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](#)

The screenshot shows the Jira Timeline view for the 'Experiment 5' project. The left sidebar contains navigation links for Planning (Timeline, Backlog, Board, Calendar, List, Goals, Issues, Add view), Development (Code, Project pages, Add shortcut), and a note that the user is in a team-managed project. The main area displays a timeline from February to May. A single sprint, 'SCRUM Sprint 1', is highlighted in blue for the month of March. The 'Sprints' section includes a '+ Create Epic' button. The top right features standard Jira navigation buttons: Give feedback, Share, Export, View settings, and a search bar indicating '10 days left'. The bottom right has a 'Quickstart' button.

This screenshot is identical to the one above, but the 'SCRUM Sprint 1' is now explicitly labeled as an 'ACTIVE SPRINT' with a magnifying glass icon. A tooltip provides details: 'Sprint goal goes here', 'Sprint start: 2025/03/25', and 'Sprint end: 2025/04/08'. The rest of the interface remains the same, including the sidebar, timeline, and navigation buttons.

Conclusion: Scrum project was implemented using JIRA.

Experiment 08

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

Theory:

Project Scheduling-

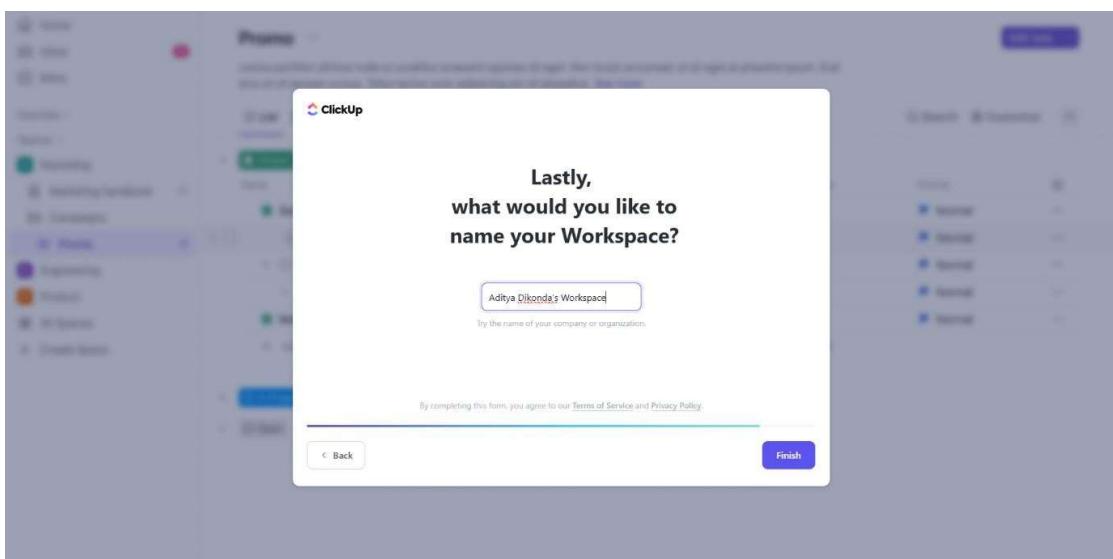
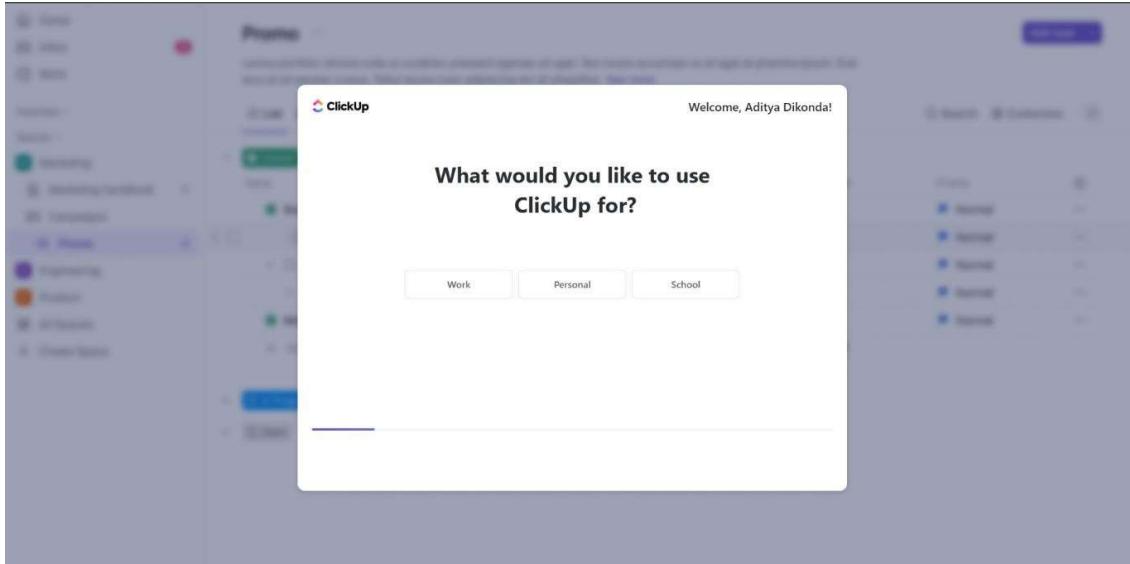
A project schedule is a timeline that outlines the tasks, milestones, deadlines, and resources required for completing a project. It helps project managers organise and plan the sequence of activities, allocate resources efficiently, set realistic timelines, and monitor progress. Having a clear project 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.

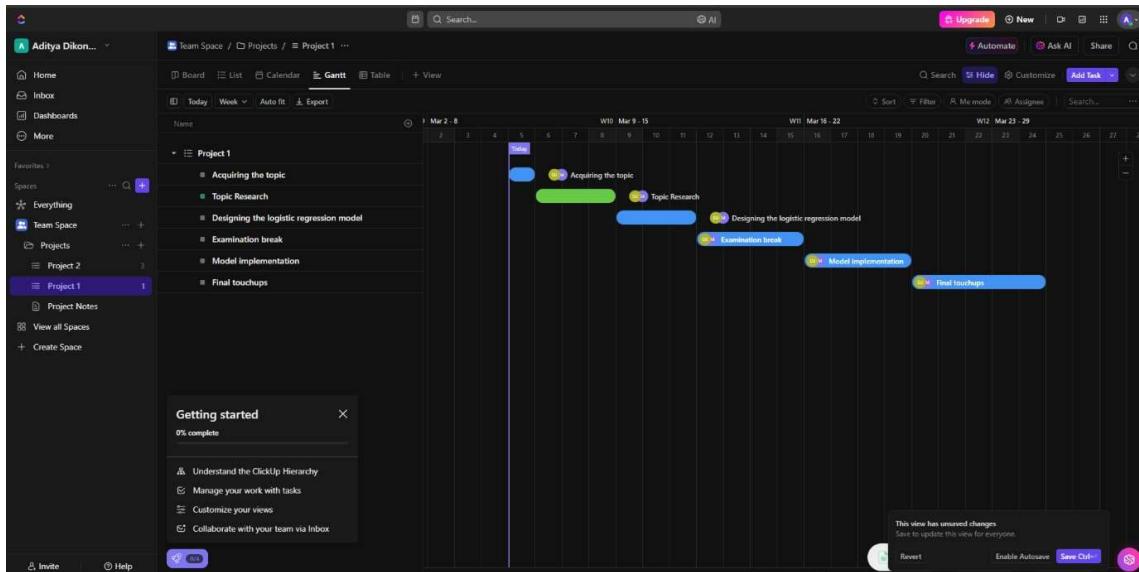
Gantt chart-

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:



The screenshot shows the details for the 'Acquiring the topic' task in ClickUp. Task status is 'COMPLETE'. Dates: 'Today' to 'Today'. Priority: 'Empty'. Assignees: 'Dhanashtha Jaiswal'. Time Estimate: 'Empty'. Tags: 'Empty'. Relationships: 'Empty'. Activity log shows a status change from 'In Progress' to 'Complete' by Dhanashtha Jaiswal 6 minutes ago. Subtasks and checklists sections are present at the bottom.

The screenshot shows a task titled "Topic Research" in Asana. The task details are as follows:

- Status: COMPLETE
- Dates: Tomorrow → Sat
- Time Estimate: Empty
- Priority: Empty
- Track Time: Add time
- Tags: Empty
- Relationships: Empty

Custom Fields, Subtasks, and Checklists sections are present. The Activity sidebar shows a recent update: "You changed due date from Fri to Sat" 5 mins ago.

The screenshot shows a task titled "Designing the logistic regression model" in Asana. The task details are as follows:

- Status: COMPLETE
- Dates: Sun → Tue
- Time Estimate: Empty
- Priority: Empty
- Track Time: Add time
- Tags: Empty
- Relationships: Empty

Custom Fields, Subtasks, and Checklists sections are present. The Activity sidebar shows a recent update: "Dhanitha Jawani changed status from To Do to Complete" 4 mins ago.

The screenshot shows a task titled "Examination break" in Asana. The task details are as follows:

- Status: COMPLETE
- Dates: Mar 12 → Mar 15
- Time Estimate: Empty
- Priority: Empty
- Track Time: Add time
- Tags: Empty
- Relationships: Empty

Custom Fields, Subtasks, and Checklists sections are present. The Activity sidebar shows a recent update: "Dhanitha Jawani changed status from To Do to Complete" 3 mins ago.

Model implementation

Status: COMPLETE (Mar 19)

Assignees: [Two icons]

Dates: Mar 16 → Mar 19

Priority: Empty

Time Estimate: Empty

Track Time: Add time

Tags: Empty

Relationships: Empty

Custom Fields: + Create Custom Field

Subtasks: + New Task

Checklists: + Create Checklist

Activity: Created on Mar 5, Share, Activity, Add Task, Search...

Write a comment... Send

Spreadsheets, Google Sheets, Notion

Final touchups

Status: COMPLETE (Mar 24)

Assignees: [Two icons]

Dates: Mar 20 → Mar 24

Priority: Empty

Time Estimate: Empty

Track Time: Add time

Tags: Empty

Relationships: Empty

Custom Fields: + Create Custom Field

Subtasks: + New Task

Checklists: + Create Checklist

Activity: Created on Mar 5, Share, Activity, Add Task, Search...

Write a comment... Send

Spreadsheets, Google Sheets, Notion

Project 1

- Acquiring the topic
- Topic Research
- Designing the logistic regression model
- Examination break
- Model Implementation
- Final touchups

Timeline: Mar 2 - 8, Mar 9 - 15, Mar 16 - 22, Mar 23 - 29

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 Changes, Undo

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.

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
8ale25ce7c4f: Pull complete
8ab039a68e51: Pull complete
2b12a49dcfb9: Pull complete
cdf9868f47ac: Pull complete
e73ea5d3136b: Pull complete
890ad32c613f: Pull complete
4f4fb700ef54: Pull complete
ba517b76f92b: Pull complete
Digest: sha256:7dd707032d90c6eaaf566f62a00f5b0116ae08fd7d6cbbb0f311b82b47171a2
Status: Downloaded newer image for redis:latest
1:C 13 Mar 2024 03:19:03.928 * o000o000o000o Redis is starting o000o000o000o
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...
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
redis	latest	170ale90f843	2 months ago	138MB

```
C:\Users\202>docker pull redis
Using default tag: latest
latest: Pulling from library/redis
Digest: sha256:7dd707032d90c6eaaf566f62a00f5b0116ae08fd7d6cbbb0f311b82b47171a2
Status: Image is up to date for redis:latest
docker.io/library/redis:latest
```

TSEC

```
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 170ale90f843 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# |
```

TSEC

```
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": "1c44727440831475b093dcfb93163064b819bdd9ad8378bb3a4fa847dc411d80",
 "Created": "2024-03-13T03:19:03.418741433Z",
 "Path": "docker-entrypoint.sh",
 "Args": [
   "redis-server"
 ],
 "State": {
   "Status": "running",
   "Running": true,
   "Paused": false,
   "Restarting": false,
   "OOMKilled": false,
   "Dead": false,
   "Pid": 2112,
   "ExitCode": 0,
   "Error": "",
   "StartedAt": "2024-03-13T03:32:13.750463204Z",
   "FinishedAt": "2024-03-13T03:32:13.145321277Z"
 },
 "Image": "sha256:170a1e90f8436daa6778aeea3926e716928826c215ca23a8dfd8055f663f9428",
 "ResolvConfPath": "/var/lib/docker/containers/1c44727440831475b093dcfb93163064b819bdd9a
 }
```

```
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.

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 Dockerfile           0.0s
=> => transferring Dockerfile: 192B                          0.0s
=> [internal] load metadata for docker.io/library/python    3.5s
=> [auth] library/python:pull token for registry           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
=> => sha256:4bcd5d5bc81ca 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: 1948                         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-flask-app:v1 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:



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