

**Name: Ansari Hanzala**

**Roll No: 612006**

**Branch: T.E. / I.T**

### **Experiment no. 01**

**Aim:** Understanding the concept of DevOps with related technologies.

#### **Theory:**

DevOps is a set of practices that combines software development (Dev) and information-technology operations (Ops) which aims to shorten the systems development life cycle and provide continuous delivery with high software quality. DevOps is the practice of operations and development engineers participating together in the entire service lifecycle, from design through the development process to production support.

**Waterfall Model:** The waterfall model is a breakdown of project activities into linear sequential phases, where each phase depends on the deliverables of the previous one and corresponds to a specialization of tasks. The approach is typical for certain areas of engineering design.

#### **Advantages of waterfall model:**

1. It allows for departmentalization and managerial control.
2. Simple and easy to understand and use.
3. Easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.
4. Phases are processed and completed one at a time.
5. Works well for smaller projects where requirements are very well understood.
6. A schedule can be set with deadlines for each stage of development and a product can proceed through the development process like a car in a carwash, and theoretically, be delivered on time.

#### **Disadvantages of waterfall model:**

1. It does not allow for much reflection or revision.
2. Once an application is in the testing stage, it is very difficult to go back and change something that was not well-thought out in the concept stage.
3. No working software is produced until late during the life cycle.
4. High amounts of risk and uncertainty.
5. Not a good model for complex and object-oriented projects.
6. Poor model for long and ongoing projects.
7. Not suitable for the projects where requirements are at a moderate to high risk of changing.

**Agile development:** Agile methodology attempts to provide many opportunities to assess the direction of a project throughout the development life cycle. Agile methods break tasks into small increments with minimal planning and do not directly involve long-term planning. Iterations are short time frames that typically last from one to four weeks. Each iteration involves a cross

functional team working in all functions: planning, requirements analysis, design, coding, unit testing, and acceptance testing. At the end of the iteration a working product is demonstrated to stakeholders. This minimizes overall risk and allows the project to adapt to changes quickly. An iteration might not add enough functionality to warrant a market release, but the goal is to have an available release at the end of each iteration. Multiple iterations might be required to release a product or new features.

#### **Advantages of Agile Methodology:**

1. Agile first priority is to fulfill the customer need from beginning to end and continuous improvement to add into valuable software.
2. Agile allows change in requirements late in the development as well.
3. Agile works on delivering software regularly interval i.e. from couple of weeks to couple of month based on project.
4. Key point is to trust, support and motivate individuals to get it projects build on time.
5. Daily face-to-face conversation is key point in agile testing. This is most efficient & effective way of communication.

#### **Disadvantages of Agile Methodology**

1. Poor Resource Planning
2. Limited Documentation
3. No Finite End
4. Difficult Measurement

#### **Why Is DevOps Important?**

1. Shorter Development Cycles, Faster Innovation
2. Reduced Deployment Failures, Rollbacks, and Time to Recover
3. Improved Communication and Collaboration
4. Increased Efficiencies
5. Reduced Costs and IT Headcount

#### **DevOps tools:**

- **Git** : Git is one of the most popular DevOps tools, widely used across the software industry. It's a distributed SCM (source code management) tool, loved by remote teams and open source contributors. Git allows you to track the progress of your development work. You can save different versions of your source code and return to a previous version when necessary.
- **Jenkins** : Jenkins is the go-to DevOps automation tool for many software development teams. It's an open source CI/CD server that allows you to automate the different stages of your delivery pipeline. The main reason for Jenkins' popularity is its huge plugin ecosystem. Currently, it offers more than 1,000 plugins, so it integrates with almost all DevOps tools, from Docker to Puppet.
- **Docker** : Docker has been the number one container platform since its launch in 2013 and continues to improve. It's also thought of as one of the most important DevOps tools out there. Docker has made containerization popular in the tech world, mainly because it

makes distributed development possible and automates the deployment of your apps. It isolates applications into separate containers, so they become portable and more secure.

- **Puppet** : Puppet is a cross-platform configuration management platform. It allows you to manage your infrastructure as code. As it automates infrastructure management, you can deliver software faster and more securely. Puppet also provides developers with an open-source tool for smaller projects.
- **Chef** : Chef is a useful DevOps tool for achieving speed, scale, and consistency. It is a Cloud based system. It can be used to ease out complex tasks and perform automation.

**Conclusion:**

We studied & understood the concept of DevOps with related technologies.

**Name: Ansari Hanzala**

**Roll No: 612006**

**Branch: T.E. / I.T**

### **Experiment no. 02**

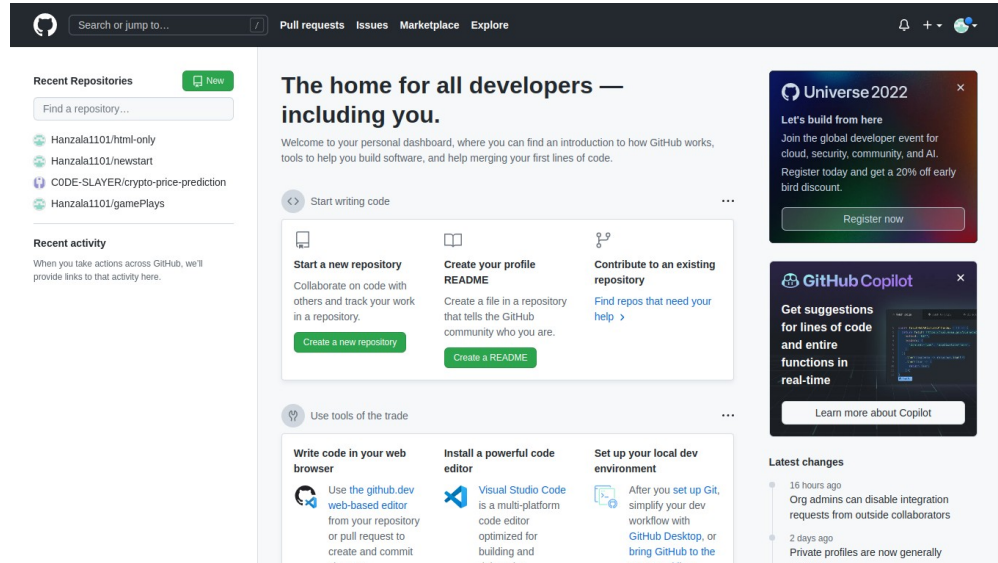
**Aim:** To perform version control on a website or software using the git version control tool.

**Theory:**

- Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later. For the examples in this book, you will use software source code as the files being version controlled, though in reality you can do this with nearly any type of file on a computer.
- A component of software configuration management, version control, also known as revision control or source control, is the management of changes to documents, computer programs, large web sites, and other collections of information.
- Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. Git is easy to learn and has a tiny footprint with lightning fast performance. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like cheap local branching, convenient staging areas, and multiple workflows.
- Founded in April 2008, GitHub is a web-based hosting service where anyone can share programming code with anyone else. GitHub offers their services for free to the general public and for businesses, they offer paid service plans. GitHub also offers a service called GitHub Gist, which is a Pastebin-like service to paste and quickly share snippets of your code. GitHub was started in 2008 and is based on a code management system developed by Linus Torvalds, called Git. Utilizing GitHub's hosting service provides users with revision control for their code, allowing them and others to view all revisions of the code shared on the site.

## Steps to install and implement version control on Github using git:

- Login to your Github account or if you don't have one go ahead and create one. After login click on the green button which say New



- Now select a name for your repo and leave the rest as default. Click on Create repository to create a repo

### Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner: Hanzala1101 / Repository name: GitHub ✓

Great repository names are short and memorable. Need inspiration? How about [psychic-telegram](#)?

Description (optional)

☒ **Public**  
Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**  
You choose who can see and commit to this repository.

**Initialize this repository with:**  
Skip this step if you're importing an existing repository.

☐ **Add a README file**  
This is where you can write a long description for your project. [Learn more.](#)

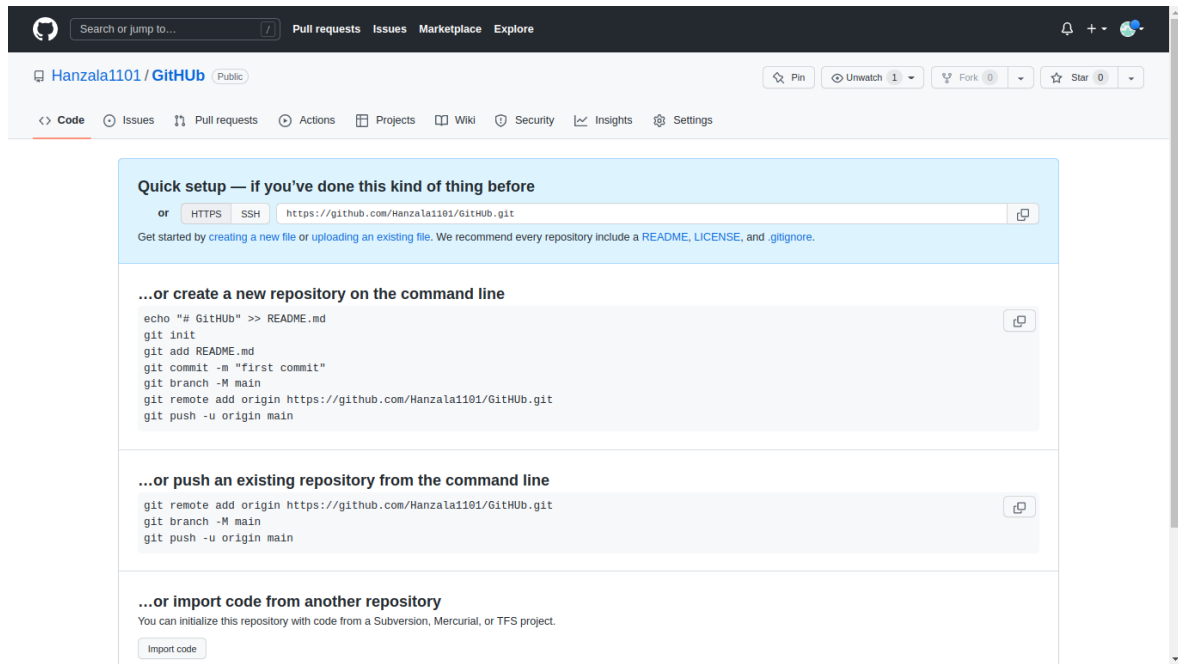
**Add .gitignore**  
Choose which files not to track from a list of templates. [Learn more.](#)  
.gitignore template: None

**Choose a license**  
A license tells others what they can and can't do with your code. [Learn more.](#)  
License: None

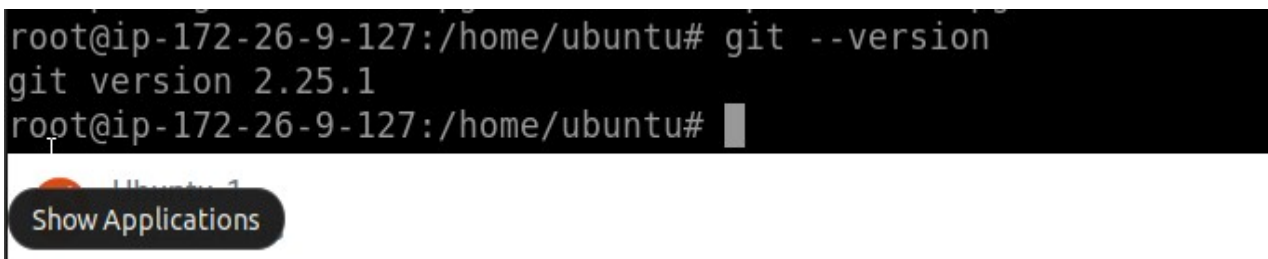
ⓘ You are creating a public repository in your personal account.

[Create repository](#)

- c. We have successfully created our first repo on Github



- d. Now open your terminal and check if you have git install or not using “git --version” and if git is not install use “sudo apt install git”



- e. Create a new dir using “mkdir dop\_exp” and write the following command to create and write content on file ‘echo “# this is my first repo and first push to Github” > readme.md’

```
root@ip-172-26-9-127:/home/ubuntu# mkdir Dop_experiment
root@ip-172-26-9-127:/home/ubuntu# cd Dop_experiment/
root@ip-172-26-9-127:/home/ubuntu/Dop_experiment# echo "# This is experiment to git version control" > readme.md
root@ip-172-26-9-127:/home/ubuntu/Dop_experiment# cat readme.md
# This is experiment to git version control
root@ip-172-26-9-127:/home/ubuntu/Dop_experiment#
```

Ubuntu-1  
13.233.92.158



- f. To make a dir a git repo use “git init”. To check the status of dir use “git status”. To track files use “git add .” this will track all the files in that dir.

```
Ubuntu-1 - Terminal | Lightsail - Google Chrome
lightsail.aws.amazon.com/ls/remote/ap-south-1/instances/Ubuntu-1/terminal?protocol=ssh
root@ip-172-26-9-127:/home/ubuntu/Dop_experiment# git init
Initialized empty Git repository in /home/ubuntu/Dop_experiment/.git/
root@ip-172-26-9-127:/home/ubuntu/Dop_experiment# git status
On branch master

No commits yet

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

nothing added to commit but untracked files present (use "git add" to track)
root@ip-172-26-9-127:/home/ubuntu/Dop_experiment# git add .
root@ip-172-26-9-127:/home/ubuntu/Dop_experiment# git commit -m "this is my first commit"
[master (root-commit) 2e2ab3f] this is my first commit
Committer: root <root@ip-172-26-9-127.ap-south-1.compute.internal>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author
```

Ubuntu-1  
13.233.92.158



g. Set your user name using “git config --global user.name "Hanzala\_1101"”.  
Set your email using ‘git config --global user.email  
"hanzala.612006.it@mhssce.ac.in"’ setting username and email will help to see  
who push the  
code And at last commit you change using ‘git commit -m “This is my first  
commit”’

```
After doing this, you may fix the identity used for this commit with:

git commit --amend --reset-author

1 file changed, 1 insertion(+)
create mode 100644 readme.md
root@ip-172-26-9-127:/home/ubuntu/Dop_experiment# git config --global user.name Hanzala1101
root@ip-172-26-9-127:/home/ubuntu/Dop_experiment# git config --global user.email hanzala.61
2006.it@mhssce.ac.in
root@ip-172-26-9-127:/home/ubuntu/Dop_experiment# git commit -m "This is my first commit"
On branch master
nothing to commit, working tree clean
root@ip-172-26-9-127:/home/ubuntu/Dop_experiment#
```



Ubuntu-1  
13.233.92.158



h. Using this command you will connect your local repo to you github repo  
“git remote add origin “https://github.com/Hanzala1101/GitHUb.git” and using this  
command you can push you committed code to github repo  
“git push -u origin master”

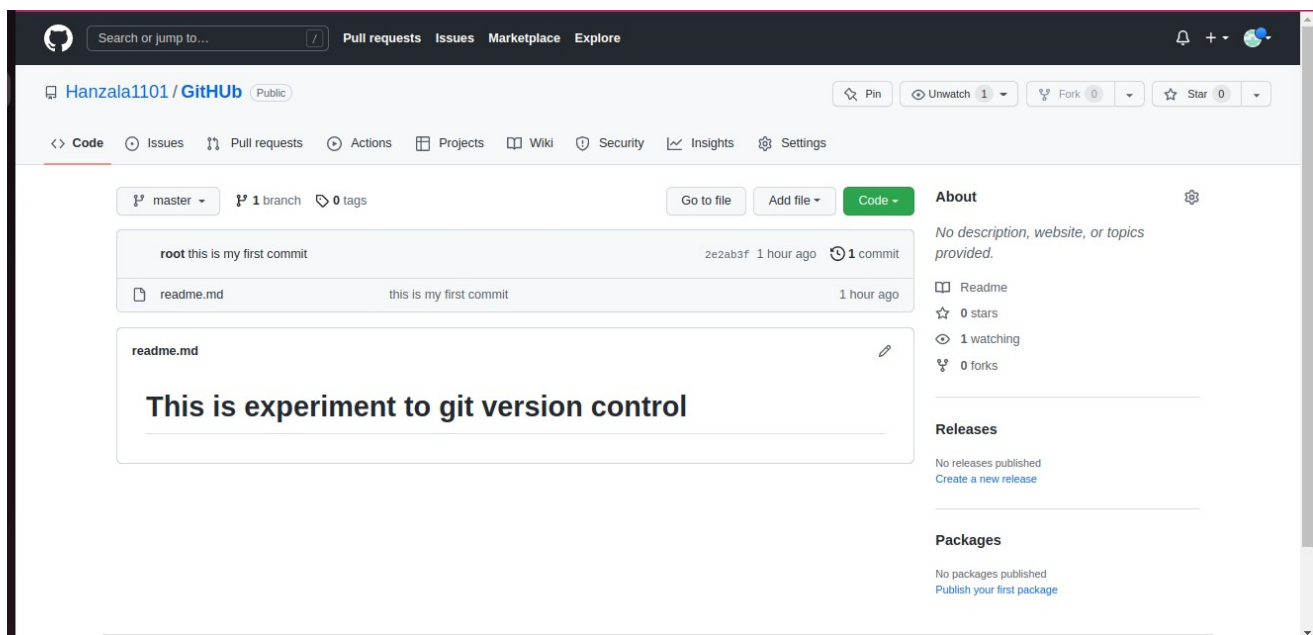
it will ask your username and password when you provide it your code will be push to your  
github account.



```
ubuntu@ip-172-26-9-127:~/Dop_experiment$ git push -u origin master
Username for 'https://github.com': Hanzala1101
Password for 'https://Hanzala1101@github.com':
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 273 bytes | 273.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/Hanzala1101/GitHub.git
 * [new branch]      master -> master
error: could not lock config file .git/config: Permission denied
error: Unable to write upstream branch configuration
hint:
hint: After fixing the error cause you may try to fix up
hint: the remote tracking information by invoking
hint: "git branch --set-upstream-to=origin/master".
error: update_ref failed for ref 'refs/remotes/origin/master': cannot lock ref 'refs/remotes/origin/master': unable to create directory for .git/refs/remotes/origin/master
ubuntu@ip-172-26-9-127:~/Dop_experiment$
```

Ubuntu-1  
13.233.92.158

i. Now go to your github repo and see the readme file will be upload there



**Conclusion:** We performed version control on website using git version control tool

Name: Ansari Hanzala

Roll No.: 612006

Branch: T.E. / I.T      Subject: DOP

### Experiment No 03

**Aim:** Install and configure Jenkins.

**Minimum hardware requirements:**

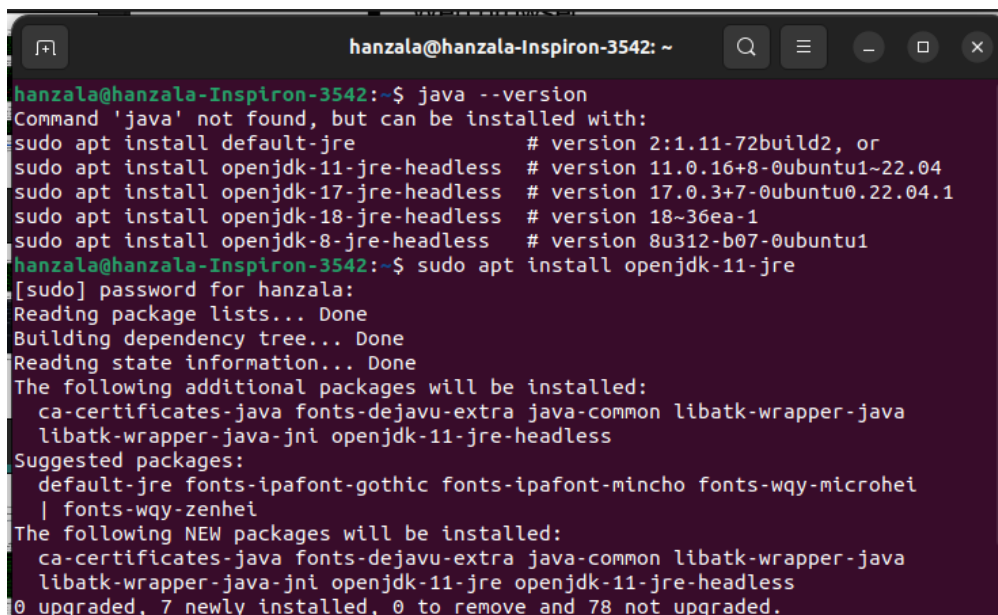
- 256 MB of RAM
- 1 GB drive space (although 10 GB is a recommended minimum if running Jenkins as a Docker container)

**Software requirements:**

- Java
- Web browser

**Steps of installing and configuring Jenkins.**

- Open up your terminal and check where Java is install using “java –version” and if java is not install run “sudo apt install openjdk-11-jre ”



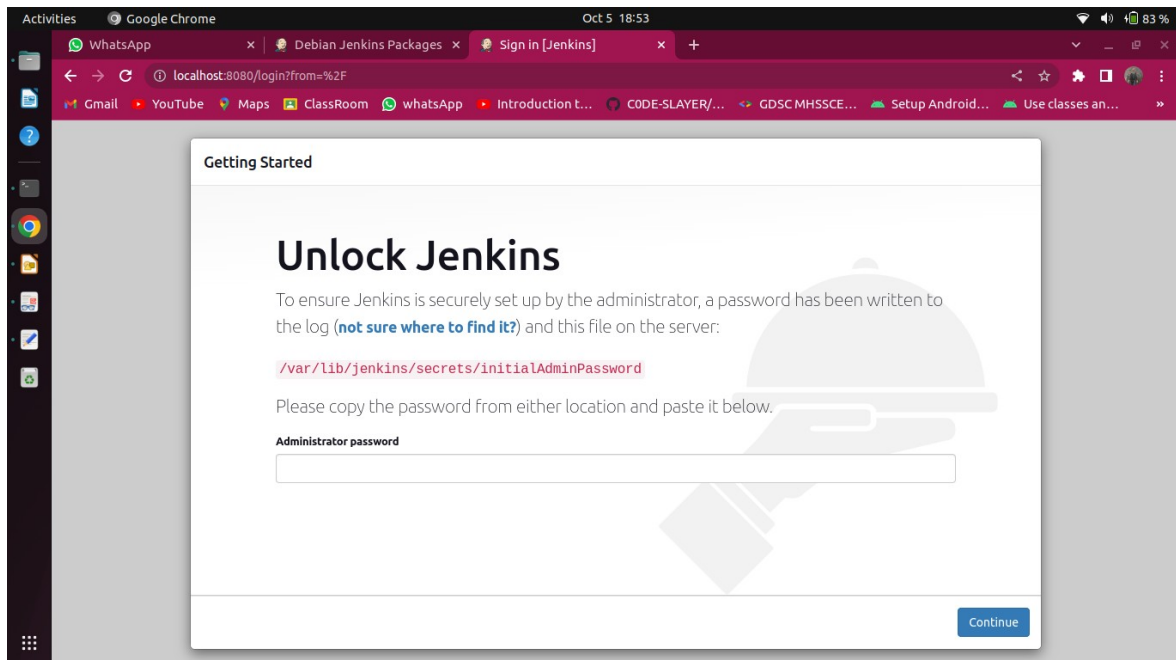
```
hanzala@hanzala-Inspiron-3542: ~  
hanzala@hanzala-Inspiron-3542:~$ java --version  
Command 'java' not found, but can be installed with:  
sudo apt install default-jre # version 2:1.11-72build2, or  
sudo apt install openjdk-11-jre-headless # version 11.0.16+8-0ubuntu1~22.04  
sudo apt install openjdk-17-jre-headless # version 17.0.3+7-0ubuntu0.22.04.1  
sudo apt install openjdk-18-jre-headless # version 18~36ea-1  
sudo apt install openjdk-8-jre-headless # version 8u312-b07-0ubuntu1  
hanzala@hanzala-Inspiron-3542:~$ sudo apt install openjdk-11-jre  
[sudo] password for hanzala:  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
The following additional packages will be installed:  
  ca-certificates-java fonts-dejavu-extra java-common libatk-wrapper-java  
  libatk-wrapper-java-jni openjdk-11-jre-headless  
Suggested packages:  
  default-jre fonts-ipafont-gothic fonts-ipafont-mincho fonts-wqy-microhei  
  | fonts-wqy-zenhei  
The following NEW packages will be installed:  
  ca-certificates-java fonts-dejavu-extra java-common libatk-wrapper-java  
  libatk-wrapper-java-jni openjdk-11-jre openjdk-11-jre-headless  
0 upgraded, 7 newly installed, 0 to remove and 78 not upgraded.
```

- Copy and paste following command one by one and paste it in your terminal
  - `curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo tee /usr/share/keyrings/jenkins-keyring.asc > /dev/null`
  - `echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] https://pkg.jenkins.io/debian-stable binary/ | sudo tee /etc/apt/sources.list.d/jenkins.list > /dev/null`
  - `sudo apt-get update`

#### 4. sudo apt-get install jenkins

```
hanzala@hanzala-Inspiron-3542: ~  
hanzala@hanzala-Inspiron-3542:~$ curl -fsSL https://pkg.jenkins.io/debian-stable  
/jenkins.io.key | sudo tee \  
/usr/share/keyrings/jenkins-keyring.asc > /dev/null  
hanzala@hanzala-Inspiron-3542:~$ echo deb [signed-by=/usr/share/keyrings/jenkins  
-keyring.asc] \  
https://pkg.jenkins.io/debian-stable binary/ | sudo tee \  
/etc/apt/sources.list.d/jenkins.list > /dev/null  
hanzala@hanzala-Inspiron-3542:~$ sudo apt-get update  
sudo apt-get install fontconfig openjdk-11-jre  
sudo apt-get install jenkins  
Ign:1 https://pkg.jenkins.io/debian-stable binary/ InRelease  
Hit:2 https://brave-browser-apt-release.s3.brave.com stable InRelease  
Get:3 https://pkg.jenkins.io/debian-stable binary/ Release [2,044 B]  
Hit:4 https://deb.nodesource.com/node_16.x jammy InRelease  
Get:5 https://pkg.jenkins.io/debian-stable binary/ Release.gpg [833 B]  
Hit:6 http://security.ubuntu.com/ubuntu jammy-security InRelease  
Hit:7 https://dl.google.com/linux/chrome/deb stable InRelease  
Get:8 https://pkg.jenkins.io/debian-stable binary/ Packages [23.2 kB]  
Hit:9 http://in.archive.ubuntu.com/ubuntu jammy InRelease  
Hit:10 http://in.archive.ubuntu.com/ubuntu jammy-updates InRelease  
Hit:11 http://in.archive.ubuntu.com/ubuntu jammy-backports InRelease  
Fetched 26.1 kB in 1s (21.0 kB/s)  
Reading package lists... Done  
Reading package lists... Done
```

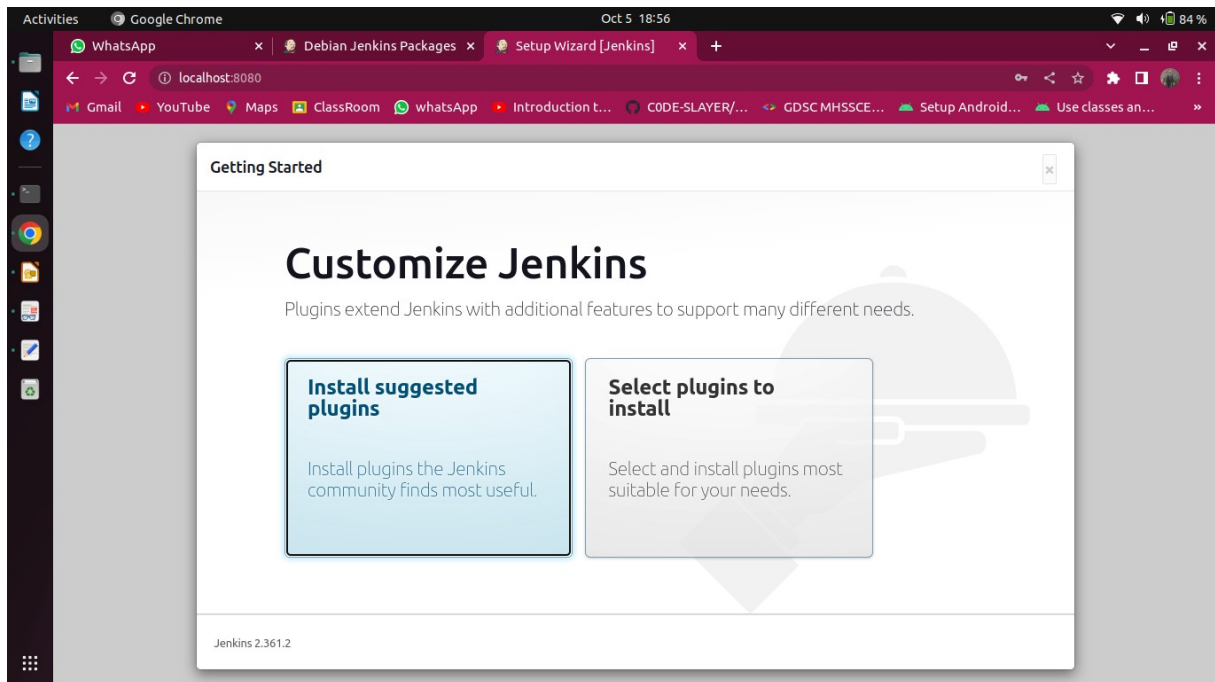
- c. Now open any browser and type localhost:8080 to get start with jenkins in my case the url is 13.232.140.91:8080 and we are here on login page of jenkins



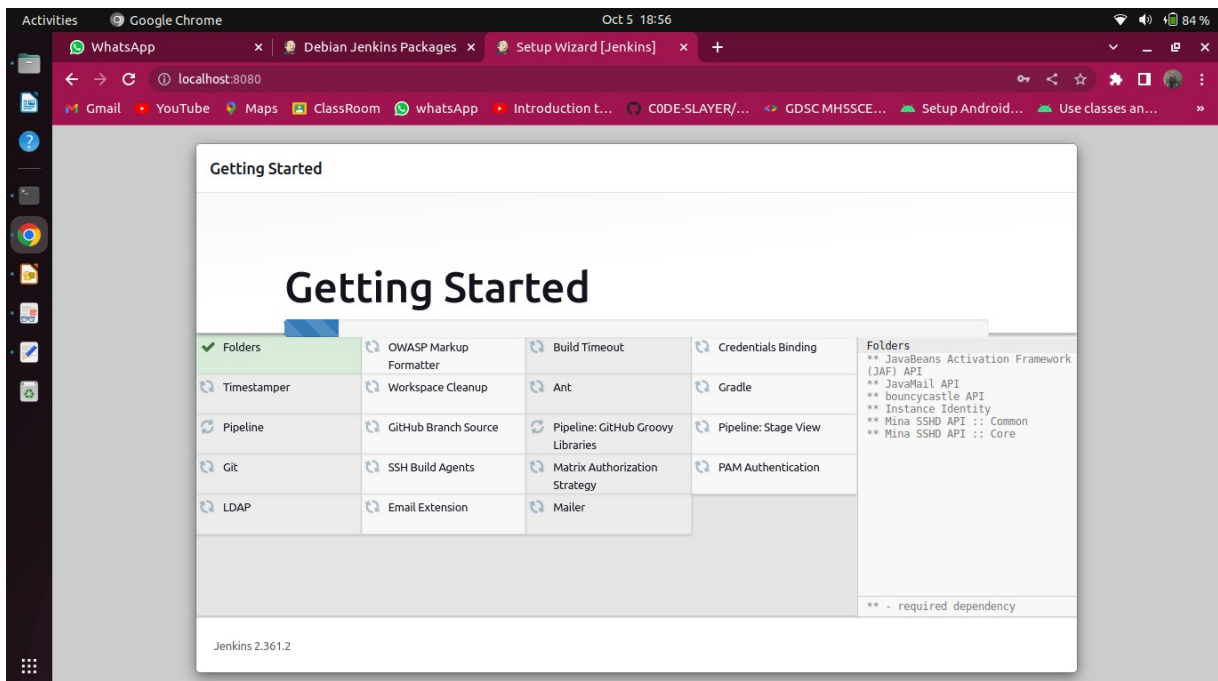
- d. Now go back to your terminal and run the command to get the password “`sudo cat /var/lib/jenkins/secrets/initialAdminPassword`” and copy the password and paste it then click on continue

```
hanzala@hanzala-Inspiron-3542:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
796383ceabc84052837ba2ea5ea7e14d
hanzala@hanzala-Inspiron-3542:~$
```

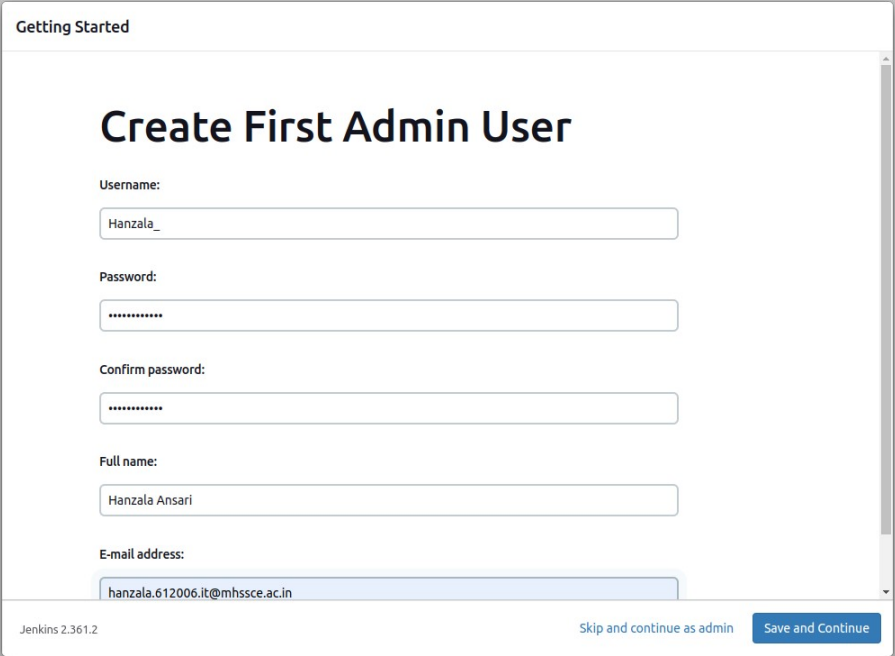
- e. Select install suggested plugins



- f. Let the installation of all plugins fine

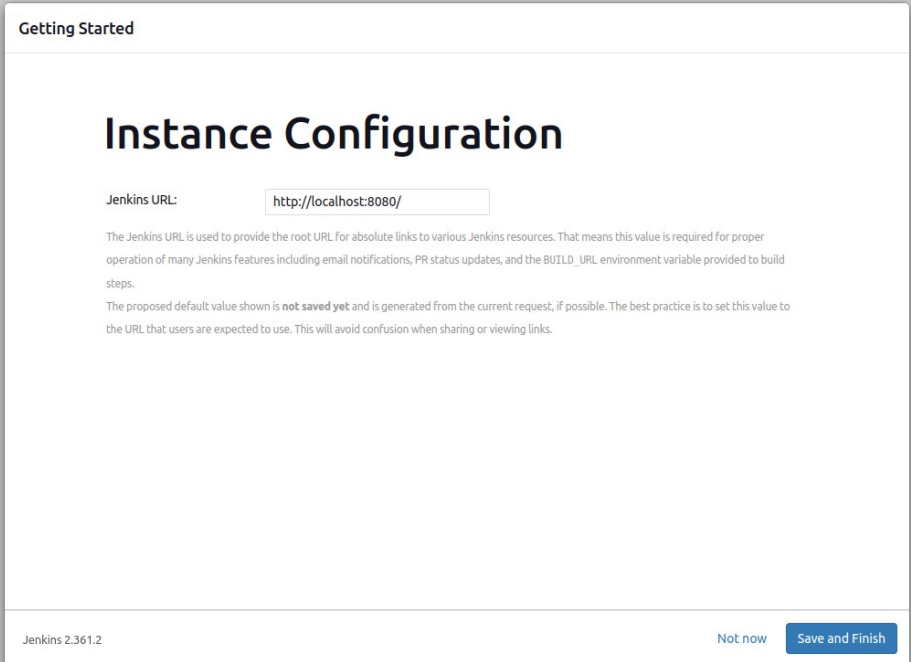


- g. After the installation of plugins select a username, password, your full name and email



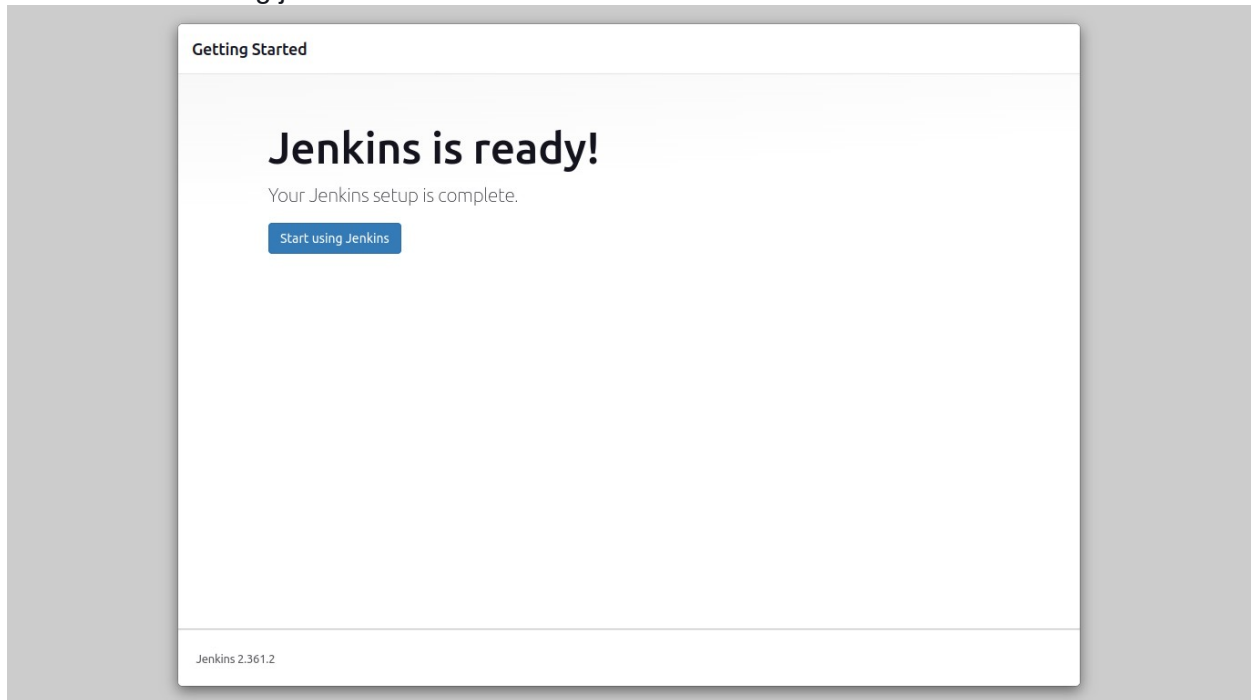
The screenshot shows the 'Getting Started' page of Jenkins 2.361.2. The main heading is 'Create First Admin User'. Below this, there are five input fields: 'Username' (containing 'Hanzala\_'), 'Password' (masked with dots), 'Confirm password' (masked with dots), 'Full name' (containing 'Hanzala Ansari'), and 'E-mail address' (containing 'hanzala.612006.it@mhsce.ac.in'). At the bottom left, it says 'Jenkins 2.361.2'. At the bottom right, there are two buttons: 'Skip and continue as admin' and 'Save and Continue'.

- h. Just click save and finish to continue

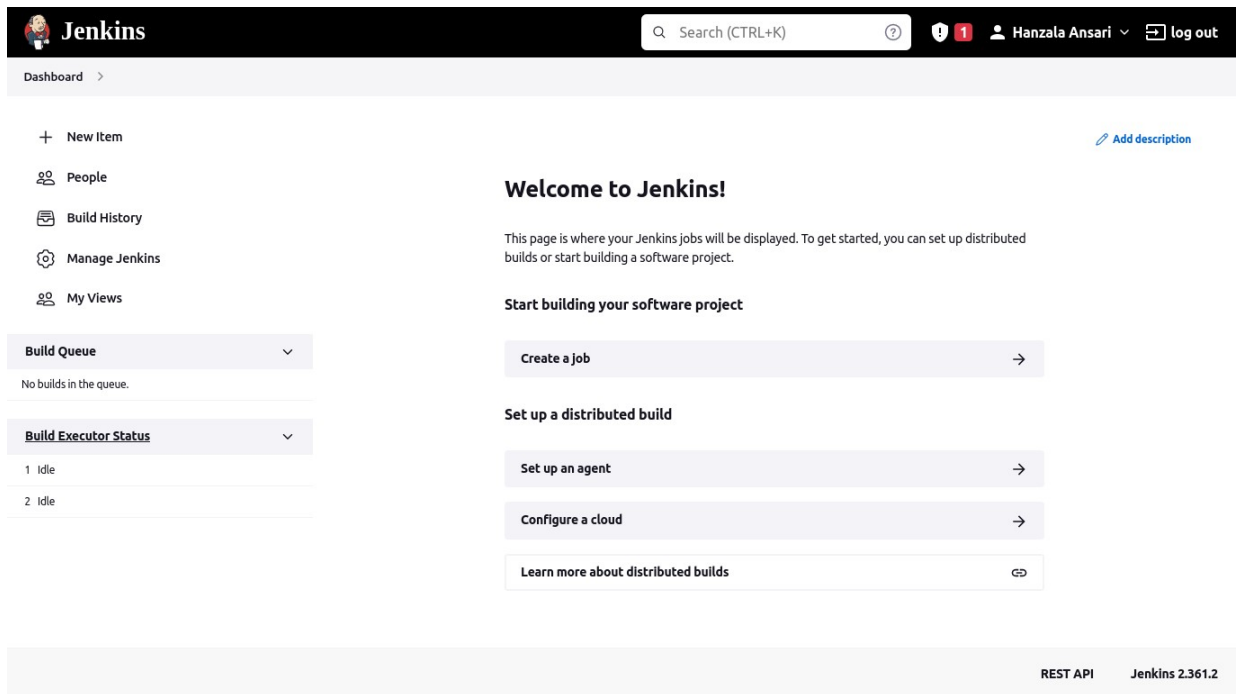


The screenshot shows the 'Getting Started' page of Jenkins 2.361.2. The main heading is 'Instance Configuration'. Below this, there is a 'Jenkins URL' field containing 'http://localhost:8080/'. Below the field, there is a paragraph of text explaining the Jenkins URL: 'The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the BUILD\_URL environment variable provided to build steps. The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.' At the bottom left, it says 'Jenkins 2.361.2'. At the bottom right, there are two buttons: 'Not now' and 'Save and Finish'.

- i. Click on start using jenkins



- j. Now we have successfully installed and configured jenkins. We have got our dashboard



**Conclusion:** We have successfully installed and configured jenkins on our local machine

Name: Hanzala Ansari  
Roll NO. 612006

## Experiment No. 04

**Aim:** To integrate Github with Jenkins.

### Theory:

Jenkins has a number of plugins for integrating into GitHub. The primary avenues for integrating your Jenkins instance with GitHub are:

- "build integration" - using GitHub to trigger builds
- "authentication integration" - using GitHub as the source of authentication information to secure a Jenkins instance.

### Build integration:

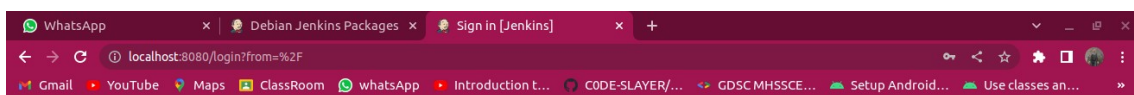
With the help of the Git plugin Jenkins can easily pull source code from any Git repository that the Jenkins build node can access. Going the other direction, the GitHub plugin can also feed information back into GitHub via the commit status API.

### Authenticating with GitHub:

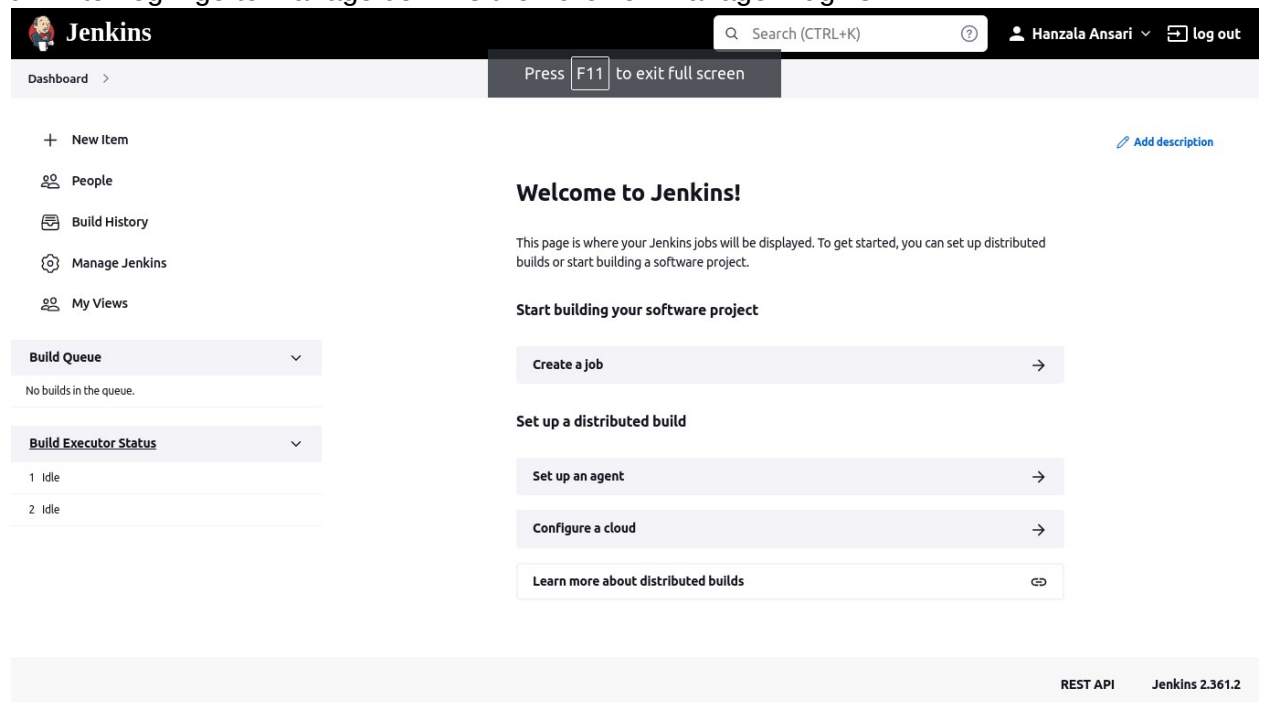
Using the GitHub Authentication plugin it is possible to use GitHub's own authentication scheme for implementing authentication in your Jenkins instance.

### Steps:

- Fire Up your terminal and type "sudo service jenkins start" to start jenkins server and go to localhost:8080 in my case it will be 13.232.140.91:8080 and login to your account

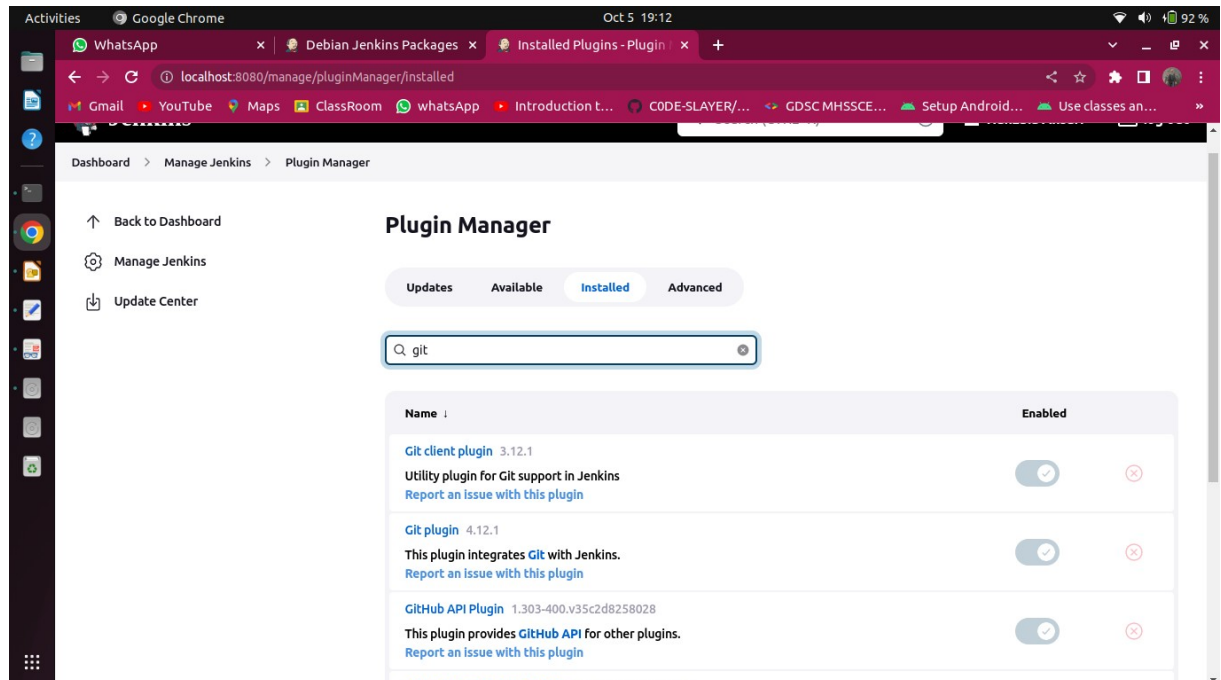


b. After login go to Manage Jenkins then click on Manage Plugins



The screenshot shows the Jenkins Dashboard. At the top, there's a header with the Jenkins logo, a search bar, and a user profile for 'Hanzala Ansari' with a 'log out' button. Below the header, a navigation sidebar on the left includes links for 'New Item', 'People', 'Build History', 'Manage Jenkins', and 'My Views'. The main content area is titled 'Welcome to Jenkins!' and contains instructions on how to start building a software project. It features several buttons: 'Create a job', 'Set up a distributed build', 'Set up an agent', 'Configure a cloud', and 'Learn more about distributed builds'. On the left sidebar, the 'Build Queue' and 'Build Executor Status' sections are visible, showing no builds in the queue and two idle executors respectively. At the bottom right, it indicates 'REST API' and 'Jenkins 2.361.2'.

c. Then head to the Installed tab and search for git and see if it is installed or not if it is installed so continue and if not then head to Available and install it. I have it so im not install it again

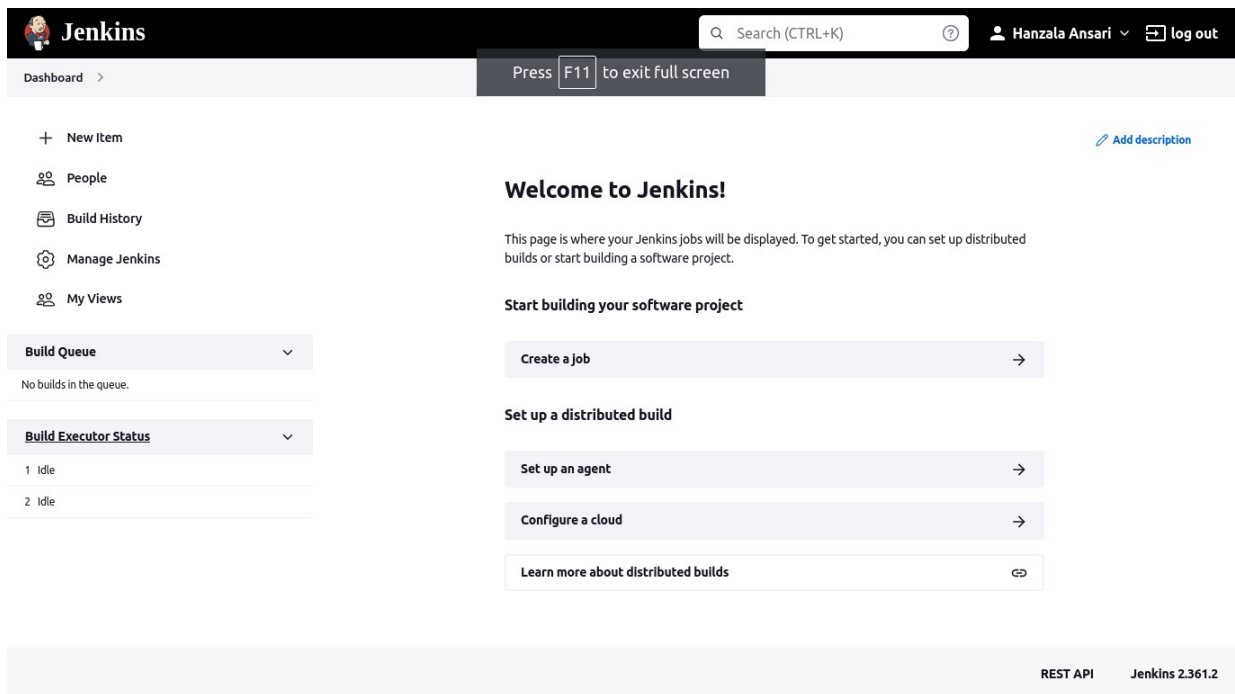


The screenshot shows the Jenkins Plugin Manager interface. The browser address bar indicates the URL 'localhost:8080/manage/pluginManager/installed'. The page has a navigation sidebar with 'Back to Dashboard', 'Manage Jenkins', and 'Update Center'. The main area is titled 'Plugin Manager' and has tabs for 'Updates', 'Available', 'Installed', and 'Advanced'. The 'Installed' tab is selected, and a search bar contains the text 'git'. Below the search bar, a table lists installed plugins:

Name	Enabled
<b>Git client plugin</b> 3.12.1 Utility plugin for Git support in Jenkins <a href="#">Report an issue with this plugin</a>	<input checked="" type="checkbox"/>
<b>Git plugin</b> 4.12.1 This plugin integrates <b>Git</b> with Jenkins. <a href="#">Report an issue with this plugin</a>	<input checked="" type="checkbox"/>
<b>GitHub API Plugin</b> 1.303-400.v35c2d8258028 This plugin provides <b>GitHub API</b> for other plugins. <a href="#">Report an issue with this plugin</a>	<input checked="" type="checkbox"/>

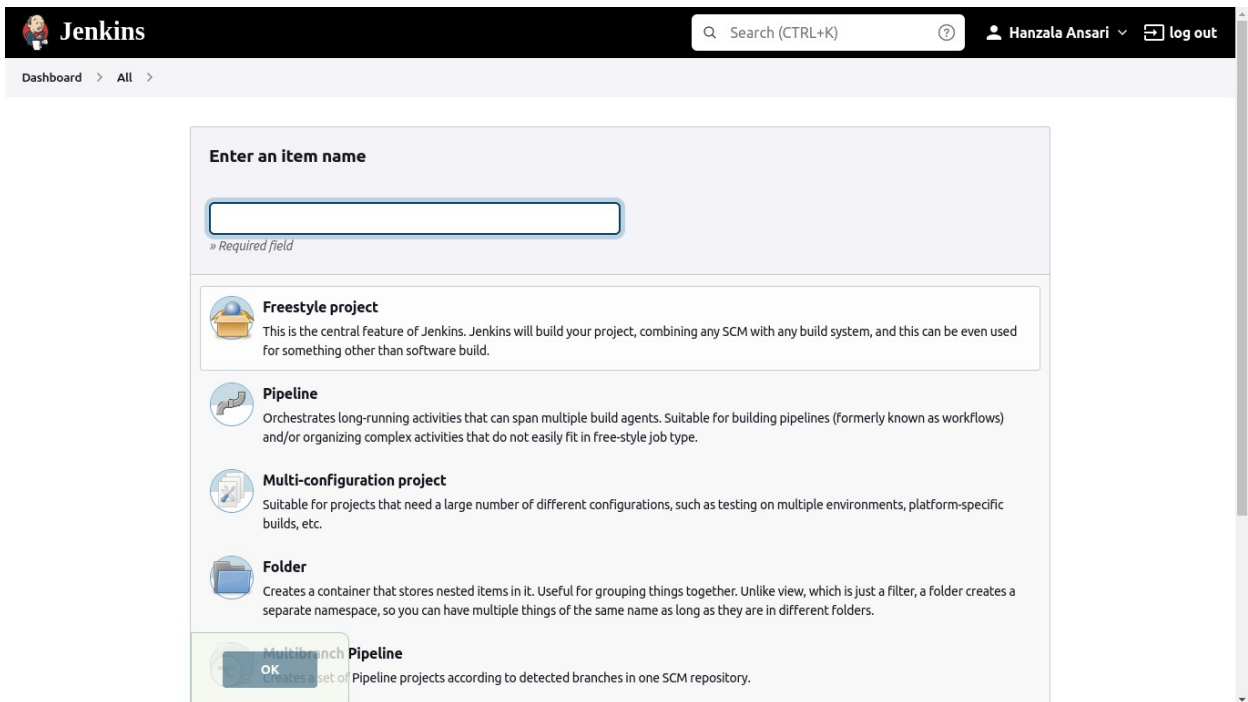


d. Go to dashboard and click on New Item



The screenshot shows the Jenkins dashboard. At the top, there's a header with the Jenkins logo, a search bar, and a user profile for 'Hanzala Ansari' with a 'log out' button. Below the header, a navigation bar shows 'Dashboard' and a button to 'Press F11 to exit full screen'. On the left sidebar, there are links for 'New Item', 'People', 'Build History', 'Manage Jenkins', and 'My Views'. The main content area has a 'Welcome to Jenkins!' message and instructions on how to get started. It includes a 'Start building your software project' section with a 'Create a job' button, and a 'Set up a distributed build' section with buttons for 'Set up an agent', 'Configure a cloud', and 'Learn more about distributed builds'. At the bottom, there's a footer with 'REST API' and 'Jenkins 2.361.2'.

e. Select a name for your project and select Freestyle project. Click on ok button



The screenshot shows the 'Enter an item name' dialog in Jenkins. It has a text input field for the item name, with a 'Required field' label below it. Below the input field, there are four project type options: 'Freestyle project', 'Pipeline', 'Multi-configuration project', and 'Folder'. Each option has a brief description. The 'Freestyle project' option is selected. At the bottom, there's a 'Multibranch Pipeline' option with an 'OK' button highlighted in a green box.

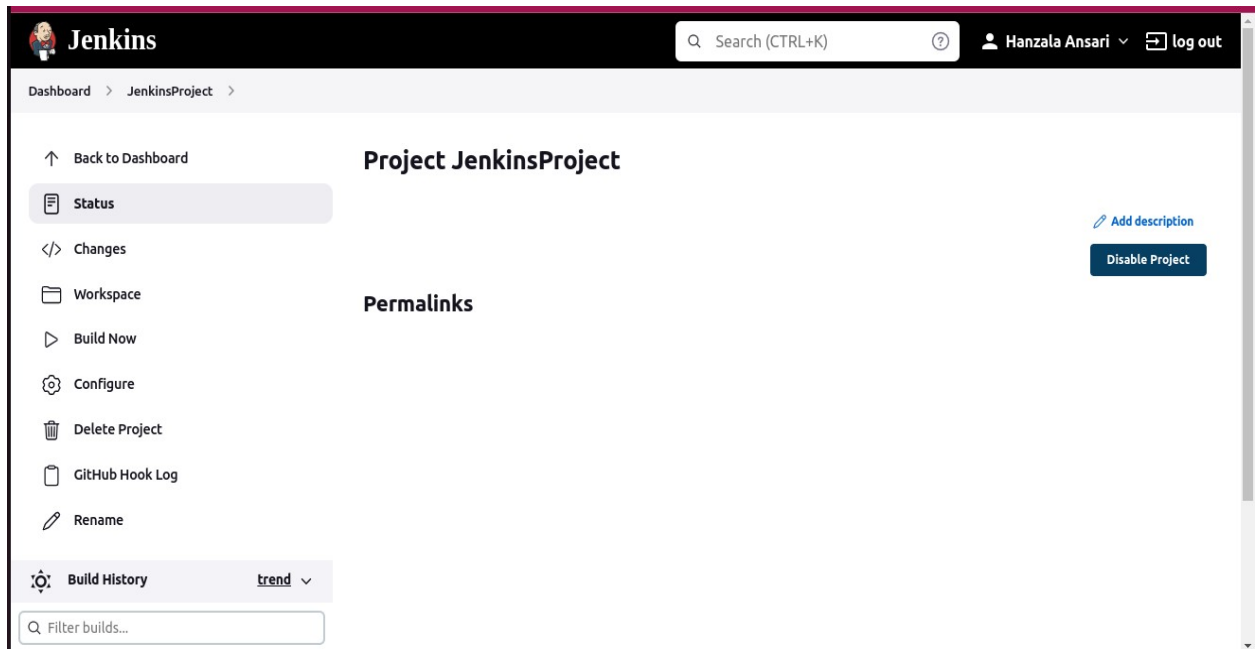
- f. Now head to Source Code Management and select git. Provide the github repo link in Repository URL

The screenshot shows the Jenkins Configuration page for a project named 'JenkinsProject'. The 'Source Code Management' tab is selected in the left sidebar. The main content area is titled 'Repositories' and contains a 'Repository URL' field with the value 'https://github.com/Hanzala1101/GitHub.git'. Below this is a 'Credentials' dropdown menu set to '- none -'. There are '+ Add' and 'Advanced...' buttons. At the bottom of the 'Repositories' section is an 'Add Repository' button. Below this is a 'Branches to build' section with a 'Branch Specifier (blank for 'any')' field. At the very bottom are 'Save' and 'Apply' buttons.

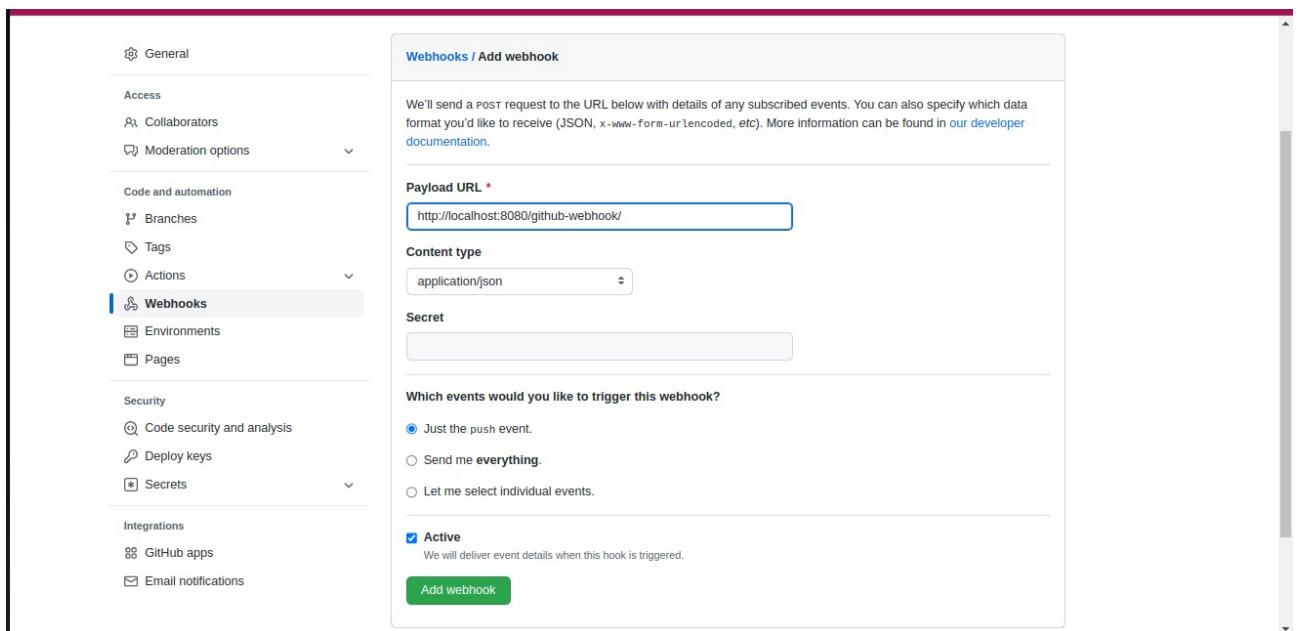
- g. Now head to Build Triggers and select “GitHub hook trigger for GITScm polling” and hit save

The screenshot shows the Jenkins Configuration page for the same 'JenkinsProject', but now the 'Build Triggers' tab is selected. Under the 'Build Triggers' section, the checkbox for 'GitHub hook trigger for GITScm polling' is checked. Other options like 'Trigger builds remotely', 'Build after other projects are built', 'Build periodically', and 'Poll SCM' are unchecked. Below this is the 'Build Environment' section with several unchecked checkboxes: 'Delete workspace before build starts', 'Use secret text(s) or file(s)', 'Add timestamps to the Console Output', 'Inspect build log for published Gradle build scans', and 'Terminate a build if it's stuck'. At the bottom are 'Save' and 'Apply' buttons.

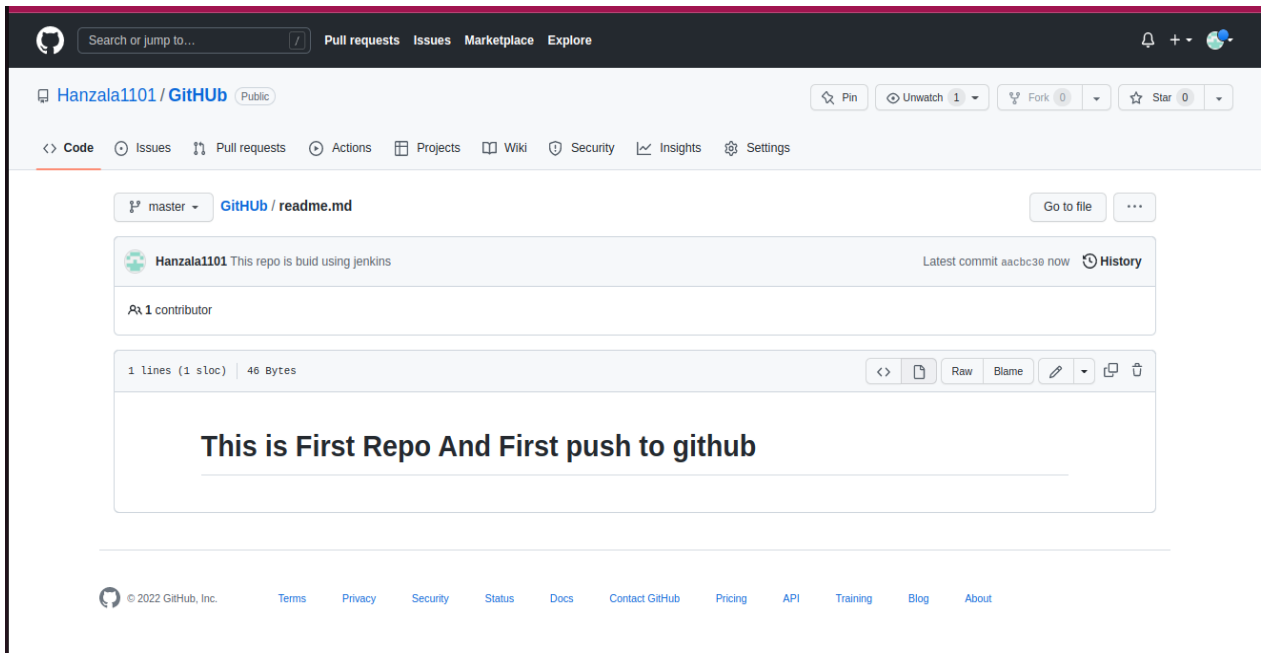
h. Well done we have integrate git to jenkins



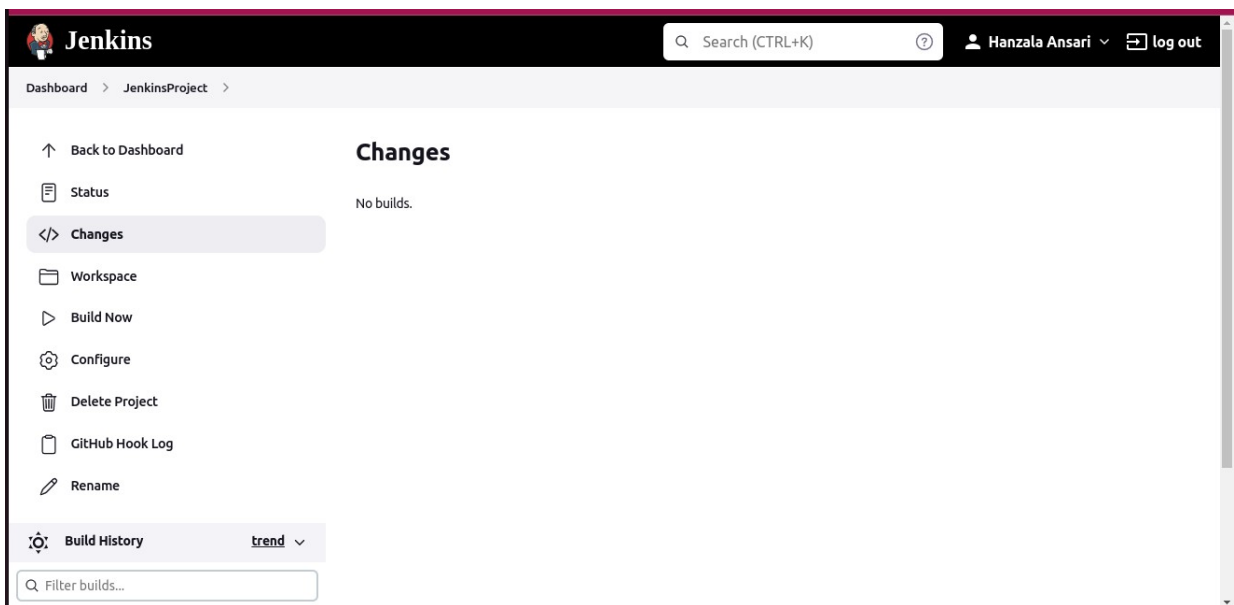
i. Now head to your repo > settings > Web hook > add new webhook. Put jenkins url <http://localhost:8080/github-webhook/> and click on add webhook



j. Make a commit on your repo



k. And we get our second build using jenkins which was made with github



Not secure | 13.232.140.91:8080/job/first%20project/2/

Jenkins

Search (CTRL+K)

Sayyed Faisal Ali log out

Dashboard > first project > #2

Back to Project

Status

Changes

Console Output

Edit Build Information

Delete build '#2'

Polling Log

Git Build Data

Previous Build

Build #2 (1 Oct 2022, 08:39:49)

Keep this build forever

Add description

Started 48 sec ago  
Took 0.65 sec

No changes.

Started by GitHub push by CODE-SLAYER

Revision: 8505efe46775921fe7ff948551b6d16c48953d74  
Repository: [https://github.com/CODE-SLAYER/dop\\_exp.git](https://github.com/CODE-SLAYER/dop_exp.git)

- refs/remotes/origin/master

REST API Jenkins 2.361.1

**Conclusion:** We have successfully integrated Github with Jenkins.

**Name: Hanzala Ansari**  
**Roll No. 612006**

### **Experiment No. 05**

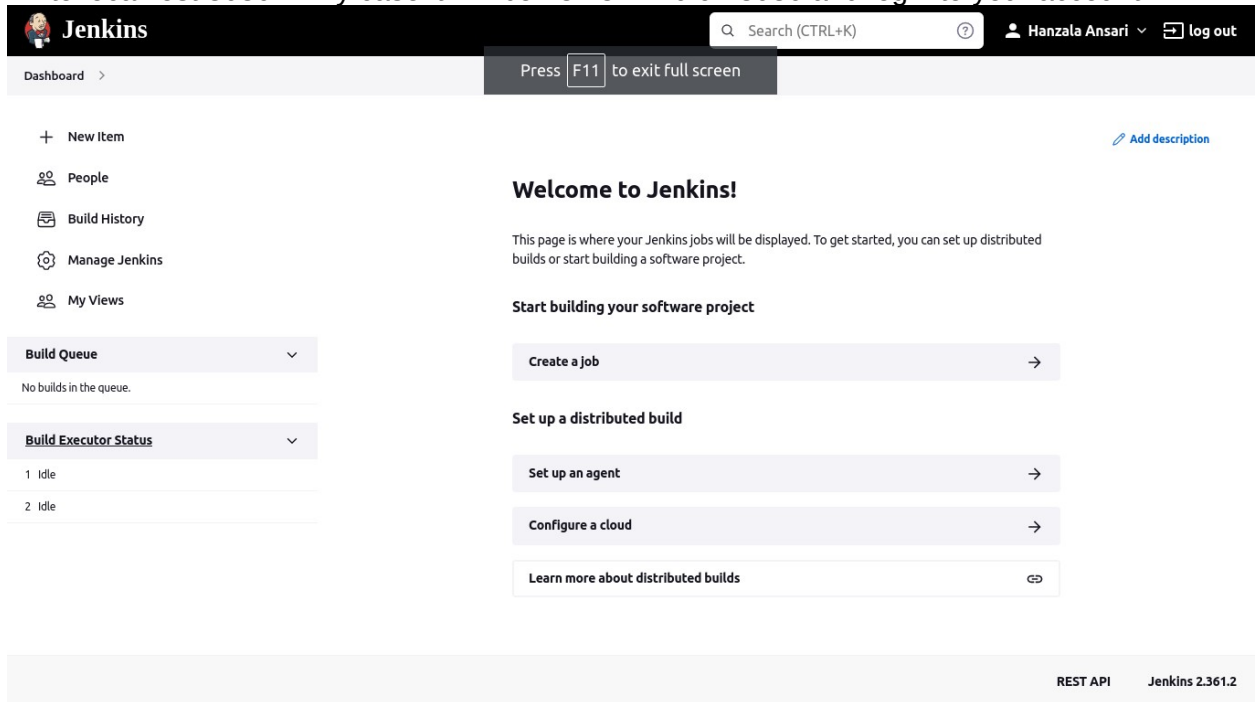
**Aim:** To perform a pipeline using Jenkins.

**Theory:**

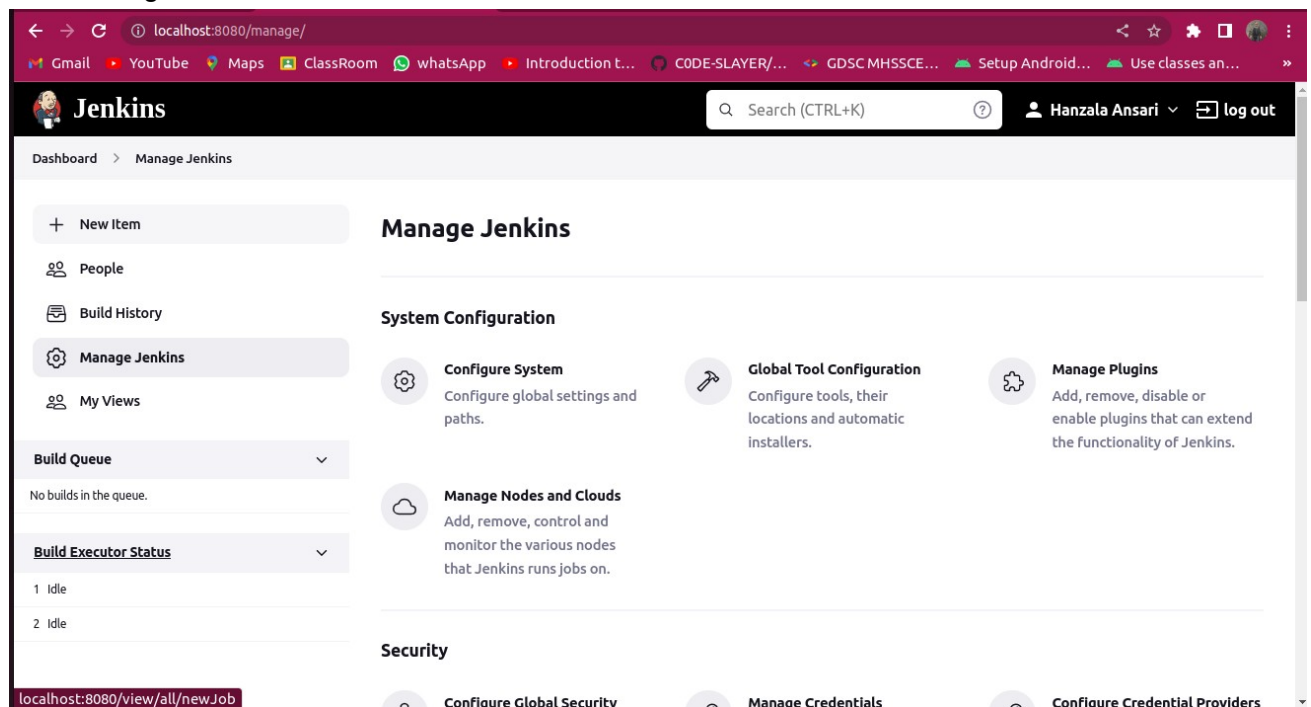
- Jenkins Pipeline (or simply "Pipeline" with a capital "P") is a suite of plugins which supports implementing and integrating continuous delivery pipelines into Jenkins.
- A continuous delivery (CD) pipeline is an automated expression of your process for getting software from version control right through to your users and customers.
- Pipeline provides an extensible set of tools for modeling simple-to-complex delivery pipelines "as code" via the Pipeline domain-specific language (DSL) syntax.
- Creating a Jenkinsfile and committing it to source control provides a number of immediate benefits:
  1. Automatically creates a Pipeline build process for all branches and pull requests.
  2. Code review/iteration on the Pipeline (along with the remaining source code).
  3. Audit trail for the Pipeline.
  4. Single source of truth for the Pipeline, which can be viewed and edited by multiple members of the project.
- Jenkins is, fundamentally, an automation engine which supports a number of automation patterns. Pipeline adds a powerful set of automation tools onto Jenkins, supporting use cases that span from simple continuous integration to comprehensive CD pipelines.
- By modeling a series of related tasks, users can take advantage of the many features of Pipeline discussed below:
  1. Code: Pipelines are implemented in code and typically checked into source control, giving teams the ability to edit, review, and iterate upon their delivery pipeline.
  2. Durable: Pipelines can survive both planned and unplanned restarts of the Jenkins master.
  3. Pausable: Pipelines can optionally stop and wait for human input or approval before continuing the Pipeline run.
  4. Versatile: Pipelines support complex real-world CD requirements, including the ability to fork/join, loop, and perform work in parallel.
  5. Extensible: The Pipeline plugin supports custom extensions to its DSL and multiple options for integration with other plugins.

## Steps:

- Fire Up your terminal and type “sudo service jenkins start” to start jenkins server and go to localhost:8080 in my case it will be 13.232.140.91:8080 and login to your account



- After login click on New Item



- c. Select a name for your project and select Pipeline then click ok button

The screenshot shows the Jenkins 'Enter an item name' dialog box. At the top, there's a breadcrumb 'Dashboard > All >'. Below it, a text input field contains 'pipeline\_exp' with a '» Required field' hint. Three project types are listed: 'Freestyle project' (described as the central feature of Jenkins), 'Pipeline' (described as orchestrating long-running activities), and 'Multi-configuration project' (described as suitable for projects needing many configurations). At the bottom, there's an 'OK' button and a faint description of a folder as a container for nested items.

- d. Head to Pipeline > Pipeline script > try simple pipeline > Hello World then click on save

The screenshot shows the Jenkins Pipeline configuration page. The breadcrumb is 'Dashboard > pipeline\_exp >'. On the left, the 'Configuration' sidebar has three tabs: 'General', 'Advanced Project Options', and 'Pipeline' (which is selected). The main area is titled 'Pipeline' and has a 'Definition' dropdown set to 'Pipeline script'. Below this is a 'Script' section with a text area containing a Groovy pipeline script: 

```
1 pipeline {
2   agent any
3
4   stages {
5     stage('Hello') {
6       steps {
7         echo 'Hello World'
8       }
9     }
10  }
11 }
12
```

 To the right of the script area is a 'try sample Pipeline...' button. Below the script area is a checkbox 'Use Groovy Sandbox' which is checked. At the bottom, there's a 'Pipeline Syntax' link and two buttons: 'Save' and 'Apply'.



- e. Now view the project and click on Build Now button on left sidebar

Dashboard > pipeline\_exp >

**Pipeline pipeline\_exp**

Build Now

Stage View

No data available. This Pipeline has not yet run.

Permalinks

Build History trend

Filter builds...

No builds

13.232.140.91:8080/job/pipeline\_exp/build?delay=0sec

- f. Head to Build History > #1 > Console Output and here we get our output which is Hello World

Jenkins

Search (CTRL+K)

Sayyed Faisal Ali log out

Dashboard > pipeline\_exp > #1

**Console Output**

```
Started by user Sayyed Faisal Ali
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/pipeline_exp
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Hello)
[Pipeline] echo
Hello World
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

REST API Jenkins 2.361.1

g. To view stage view of a project go to project view

The screenshot shows the Jenkins web interface for a pipeline named 'pipeline\_exp'. The left sidebar contains navigation links: Status (selected), Changes, Build Now, Configure, Delete Pipeline, Full Stage View, Rename, and Pipeline Syntax. Below these is a 'Build History' section with a search bar and a list of builds. The main content area is titled 'Pipeline pipeline\_exp' and includes buttons for 'Add description' and 'Disable Project'. The 'Stage View' section displays a table with stage information, including average stage times and a full run time. The 'Permalinks' section provides links to various build details.

Dashboard > pipeline\_exp >

**Pipeline pipeline\_exp**

[Add description](#)  
[Disable Project](#)

**Stage View**

Average stage times:		338ms
(Average full run time: ~5s)		
#1	Oct 01 14:35	No Changes
		338ms

**Permalinks**

- [Last build \(#1\), 3 min 8 sec ago](#)
- [Last stable build \(#1\), 3 min 8 sec ago](#)
- [Last successful build \(#1\), 3 min 8 sec ago](#)
- [Last completed build \(#1\), 3 min 8 sec ago](#)

**Build History** [trend](#) ▾

Filter builds...

#1 1 Oct 2022, 09:05

[Atom feed for all](#) [Atom feed for failures](#)

**Conclusion:** We have successfully performed a pipelining process using Jenkins.

**Name: Hanzala Ansari**  
**Roll No. 612006**

### **Experiment No. 06**

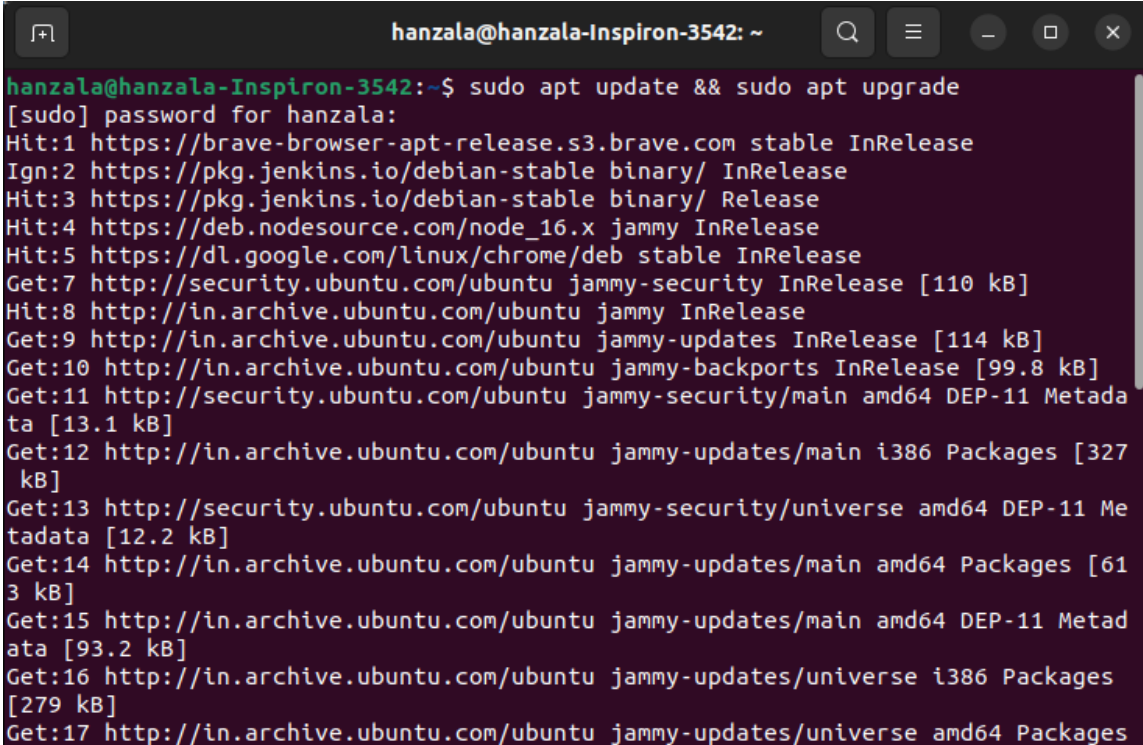
**Aim:** To install Docker and configure

#### **Theory:**

Docker, a popular operating system level virtualization platform, was released in 2013. It is free to use software that can run different tools and applications in containers. The containers are basically an isolated environment created by the Docker for each application or images of different Linux operating systems. However, despite the individual containers of each application, all of them run by using a single operating system kernel. The traditional virtual machines. Plus a wide range of images Docker is available for Linux, MacOS and Windows.

#### **Steps:**

- a. Fire up your terminal and type “sudo apt update && sudo apt upgrade”



```
hanzala@hanzala-Inspiron-3542: ~  
hanzala@hanzala-Inspiron-3542:~$ sudo apt update && sudo apt upgrade  
[sudo] password for hanzala:  
Hit:1 https://brave-browser-apt-release.s3.brave.com stable InRelease  
Ign:2 https://pkg.jenkins.io/debian-stable binary/ InRelease  
Hit:3 https://pkg.jenkins.io/debian-stable binary/ Release  
Hit:4 https://deb.nodesource.com/node_16.x jammy InRelease  
Hit:5 https://dl.google.com/linux/chrome/deb stable InRelease  
Get:7 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]  
Hit:8 http://in.archive.ubuntu.com/ubuntu jammy InRelease  
Get:9 http://in.archive.ubuntu.com/ubuntu jammy-updates InRelease [114 kB]  
Get:10 http://in.archive.ubuntu.com/ubuntu jammy-backports InRelease [99.8 kB]  
Get:11 http://security.ubuntu.com/ubuntu jammy-security/main amd64 DEP-11 Metadata [13.1 kB]  
Get:12 http://in.archive.ubuntu.com/ubuntu jammy-updates/main i386 Packages [327 kB]  
Get:13 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 DEP-11 Metadata [12.2 kB]  
Get:14 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [613 kB]  
Get:15 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 DEP-11 Metadata [93.2 kB]  
Get:16 http://in.archive.ubuntu.com/ubuntu jammy-updates/universe i386 Packages [279 kB]  
Get:17 http://in.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages
```

- b. Run the command one by one to install docker
1. `curl -fsSL https://get.docker.com -o get-docker.sh`
  2. `sh get-docker.sh`

```
hanzala@hanzala-Inspiron-3542: ~  
hanzala@hanzala-Inspiron-3542:~$ curl -fsSL https://get.docker.com -o get-docker.sh  
hanzala@hanzala-Inspiron-3542:~$ sh get-docker.sh  
# Executing docker install script, commit: 4f282167c425347a931ccfd95cc91fab041d414f  
+ sudo -E sh -c apt-get update -qq >/dev/null  
+ sudo -E sh -c DEBIAN_FRONTEND=noninteractive apt-get install -y -qq apt-transport-https ca-certificates curl >/dev/null  
+ sudo -E sh -c mkdir -p /etc/apt/keyrings && chmod -R 0755 /etc/apt/keyrings  
+ sudo -E sh -c curl -fsSL "https://download.docker.com/linux/ubuntu/gpg" | gpg --dearmor --yes -o /etc/apt/keyrings/docker.gpg  
gpg: WARNING: unsafe ownership on homedir '/home/hanzala/.gnupg'  
+ sudo -E sh -c chmod a+r /etc/apt/keyrings/docker.gpg  
+ sudo -E sh -c echo "deb [arch=amd64 signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu jammy stable" > /etc/apt/sources.list.d/docker.list  
+ sudo -E sh -c apt-get update -qq >/dev/null  
+ sudo -E sh -c DEBIAN_FRONTEND=noninteractive apt-get install -y -qq --no-install-recommends docker-ce docker-ce-cli containerd.io docker-compose-plugin docker-scan-plugin >/dev/null  
+ version_gte 20.10  
+ [ -z ]  
+ return 0  
+ sudo -E sh -c DEBIAN_FRONTEND=noninteractive apt-get install -y -qq docker-ce-
```

```
hanzala@hanzala-Inspiron-3542: ~  
Version: 20.10.18  
API version: 1.41  
Go version: go1.18.6  
Git commit: b40c2f6  
Built: Thu Sep 8 23:11:43 2022  
OS/Arch: linux/amd64  
Context: default  
Experimental: true  
  
Server: Docker Engine - Community  
Engine:  
  Version: 20.10.18  
  API version: 1.41 (minimum version 1.12)  
  Go version: go1.18.6  
  Git commit: e42327a  
  Built: Thu Sep 8 23:09:30 2022  
  OS/Arch: linux/amd64  
  Experimental: false  
containerd:  
  Version: 1.6.8  
  GitCommit: 9cd3357b7fd7218e4aec3eae239db1f68a5a6ec6  
runc:  
  Version: 1.1.4  
  GitCommit: v1.1.4-0-g5fd4c4d
```

```
hanzala@hanzala-Inspiron-3542: ~  
runc:  
Version:      1.1.4  
GitCommit:    v1.1.4-0-g5fd4c4d  
docker-init:  
Version:      0.19.0  
GitCommit:    de40ad0  
  
=====
```

To run Docker as a non-privileged user, consider setting up the Docker daemon in rootless mode for your user:

```
dockerd-rootless-setuptool.sh install
```

Visit <https://docs.docker.com/go/rootless/> to learn about rootless mode.

To run the Docker daemon as a fully privileged service, but granting non-root users access, refer to <https://docs.docker.com/go/daemon-access/>

WARNING: Access to the remote API on a privileged Docker daemon is equivalent to root access on the host. Refer to the 'Docker daemon attack surface' documentation for details: <https://docs.docker.com/go/attack-surface/>

- c. Docker is installed successfully then run “sudo service docker start” and then check it using “docker --version”

```
hanzala@hanzala-Inspiron-3542:~$ sudo service docker start  
hanzala@hanzala-Inspiron-3542:~$ docker --version  
Docker version 20.10.18, build b40c2f6  
hanzala@hanzala-Inspiron-3542:~$
```

- d. Run “sudo docker run hello-world” to pull and run hello world repo

```
hanzala@hanzala-Inspiron-3542: ~  
hanzala@hanzala-Inspiron-3542:~$ sudo docker run hello-world  
Unable to find image 'hello-world:latest' locally  
latest: Pulling from library/hello-world  
2db29710123e: Pull complete  
Digest: sha256:62af9efd515a25f84961b70f973a798d2eca956b1b2b026d0a4a63a3b0b6a3f2  
Status: Downloaded newer image for hello-world:latest  
  
Hello from Docker!  
This message shows that your installation appears to be working correctly.  
  
To generate this message, Docker took the following steps:  
1. The Docker client contacted the Docker daemon.  
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.  
   (amd64)  
3. The Docker daemon created a new container from that image which runs the  
   executable that produces the output you are currently reading.  
4. The Docker daemon streamed that output to the Docker client, which sent it  
   to your terminal.  
  
To try something more ambitious, you can run an Ubuntu container with:  
$ docker run -it ubuntu bash  
  
Share images, automate workflows, and more with a free Docker ID:  
https://hub.docker.com/  
  
For more examples and ideas, visit:  
https://docs.docker.com/get-started/  
hanzala@hanzala-Inspiron-3542:~$
```

- e. Run “sudo docker images” to see all the images pull by docker on your local machine

```
hanzala@hanzala-Inspiron-3542:~$ sudo docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
hello-world    latest    feb5d9fea6a5   12 months ago 13.3kB
hanzala@hanzala-Inspiron-3542:~$
```

**Conclusion:** We successfully installed and configured docker on our machine

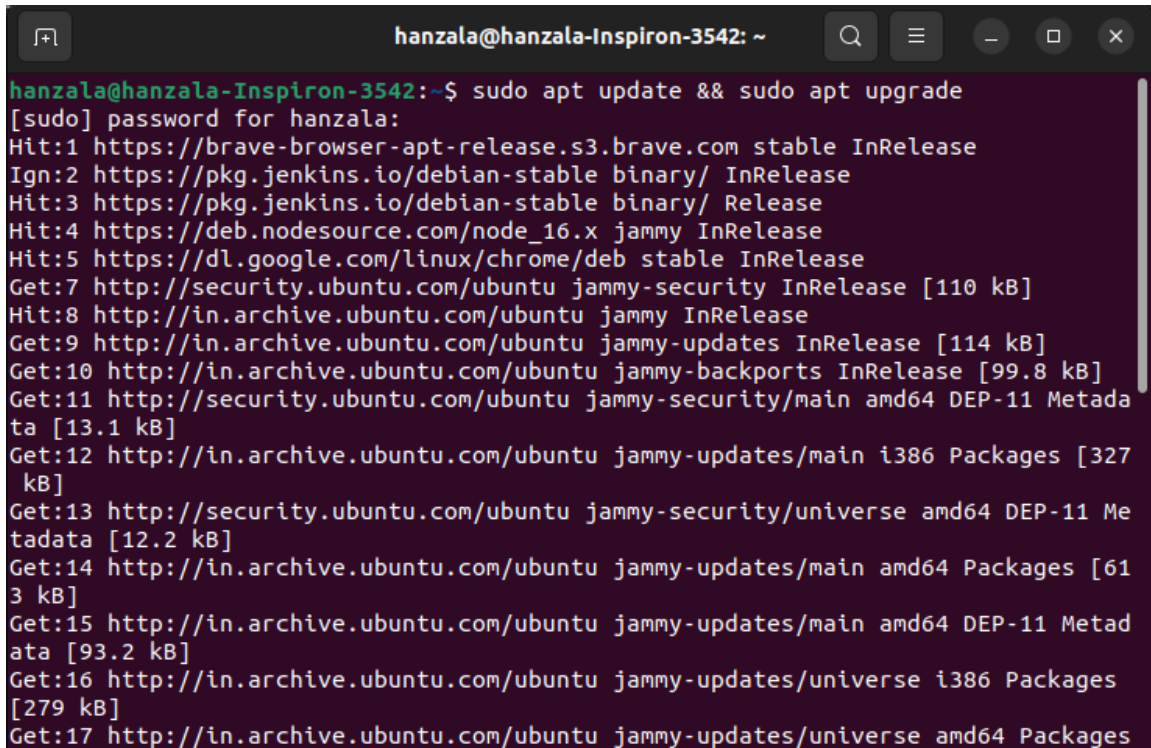
Name: Hanzala Ansari  
Roll no: 612006

### Experiment no. 07

**Aim:** Build, deploy and manage web applications on Docker

**Steps to build and deploy and manage web application:**

- a. Fire up your terminal and run “sudo apt update && sudo apt upgrade”

A terminal window titled 'hanzala@hanzala-Inspiron-3542: ~' showing the output of the command 'sudo apt update && sudo apt upgrade'. The output lists various package sources and their updates, including brave-browser, jenkins, nodejs, google-chrome, ubuntu-security, and ubuntu-updates. The terminal text is as follows:

```
hanzala@hanzala-Inspiron-3542:~$ sudo apt update && sudo apt upgrade
[sudo] password for hanzala:
Hit:1 https://brave-browser-apt-release.s3.brave.com stable InRelease
Ign:2 https://pkg.jenkins.io/debian-stable binary/ InRelease
Hit:3 https://pkg.jenkins.io/debian-stable binary/ Release
Hit:4 https://deb.nodesource.com/node_16.x jammy InRelease
Hit:5 https://dl.google.com/linux/chrome/deb stable InRelease
Get:7 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Hit:8 http://in.archive.ubuntu.com/ubuntu jammy InRelease
Get:9 http://in.archive.ubuntu.com/ubuntu jammy-updates InRelease [114 kB]
Get:10 http://in.archive.ubuntu.com/ubuntu jammy-backports InRelease [99.8 kB]
Get:11 http://security.ubuntu.com/ubuntu jammy-security/main amd64 DEP-11 Metadata [13.1 kB]
Get:12 http://in.archive.ubuntu.com/ubuntu jammy-updates/main i386 Packages [327 kB]
Get:13 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 DEP-11 Metadata [12.2 kB]
Get:14 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [613 kB]
Get:15 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 DEP-11 Metadata [93.2 kB]
Get:16 http://in.archive.ubuntu.com/ubuntu jammy-updates/universe i386 Packages [279 kB]
Get:17 http://in.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages
```

- b. Create a dir using mkdir and run “touch app.php Dockerfile” in side the dir you have created

A terminal window titled 'hanzala@hanzala-Inspiron-3542: ~' showing the execution of several commands to create a directory and files. The terminal text is as follows:

```
hanzala@hanzala-Inspiron-3542:~$ mkdir docker_web_app
hanzala@hanzala-Inspiron-3542:~$ cd docker_web_app/
hanzala@hanzala-Inspiron-3542:~/docker_web_app$ touch app.php Dockerfile
hanzala@hanzala-Inspiron-3542:~/docker_web_app$ ls
app.php  Dockerfile
hanzala@hanzala-Inspiron-3542:~/docker_web_app$
```



- c. Using nano open app.php file and write the code given below

```
hanzala@hanzala-Inspiron-3542: ...  
GNU nano 6.2 app.php  
<?php  
echo "This is my application build using docker \n \n";  
?>  
  
^G Help      ^O Write Out  ^W Where Is   ^K Cut  
^X Exit      ^R Read File  ^\ Replace    ^U Paste
```

- d. Now open Dockerfile using nano and write down the code given below

```
hanzala@hanzala-Inspiron-3542: ...  
GNU nano 6.2 Dockerfile *  
FROM php:latest  
COPY . /var/www/php/  
WORKDIR /var/www/php/  
CMD ["php", "./app.php"]  
  
^G Help      ^O Write Out  ^W Where Is   ^K Cut  
^X Exit      ^R Read File  ^\ Replace    ^U Paste
```



- e. Now all things are set using the command “sudo docker build . -t web\_app” our web app will start to build

```
hanzala@hanzala-Inspiron-3542: ~/docker_web_app
hanzala@hanzala-Inspiron-3542:~/docker_web_app$ sudo docker build . -t web_app
Sending build context to Docker daemon 3.072kB
Step 1/4 : FROM php:latest
latest: Pulling from library/php
bd159e379b3b: Pull complete
1e83b070fd97: Pull complete
e7793be89e9c: Pull complete
4220e0c03377: Pull complete
77f71a584e44: Pull complete
a18ae08838ec: Pull complete
ae54e0606adf: Pull complete
ce021f7728b0: Pull complete
021585421ec5: Pull complete
Digest: sha256:9020e8fb37438a26a5890e765e5157d3d34078ee8006de259431afc0dc10e96a
Status: Downloaded newer image for php:latest
---> aa24992dd020
Step 2/4 : COPY . /var/www/php/
---> b0bf19147ccb
Step 3/4 : WORKDIR /var/www/php/
---> Running in 61bc7ddfd7b3
Removing intermediate container 61bc7ddfd7b3
---> d8f514ffa9c2
Step 4/4 : CMD ["php","./app.php"]
---> Running in 865c52f6fc0d
Removing intermediate container 865c52f6fc0d
---> 1d7369efbb27
Successfully built 1d7369efbb27
Successfully tagged web_app:latest
hanzala@hanzala-Inspiron-3542:~/docker_web_app$
```

- f. Check weather the images is build using “sudo docker images” and run docker images using “sudo docker run web\_app”

```
hanzala@hanzala-Inspiron-3542:~/docker_web_app$ sudo docker images
REPOSITORY      TAG          IMAGE ID      CREATED        SIZE
web_app         latest      1d7369efbb27  About a minute ago  484MB
php             latest      aa24992dd020  8 hours ago    484MB
hello-world     latest      feb5d9fea6a5  12 months ago  13.3kB
hanzala@hanzala-Inspiron-3542:~/docker_web_app$
```

**Conclusion:** We successfully build and run our web application using docker

Name: Hanzala Ansari

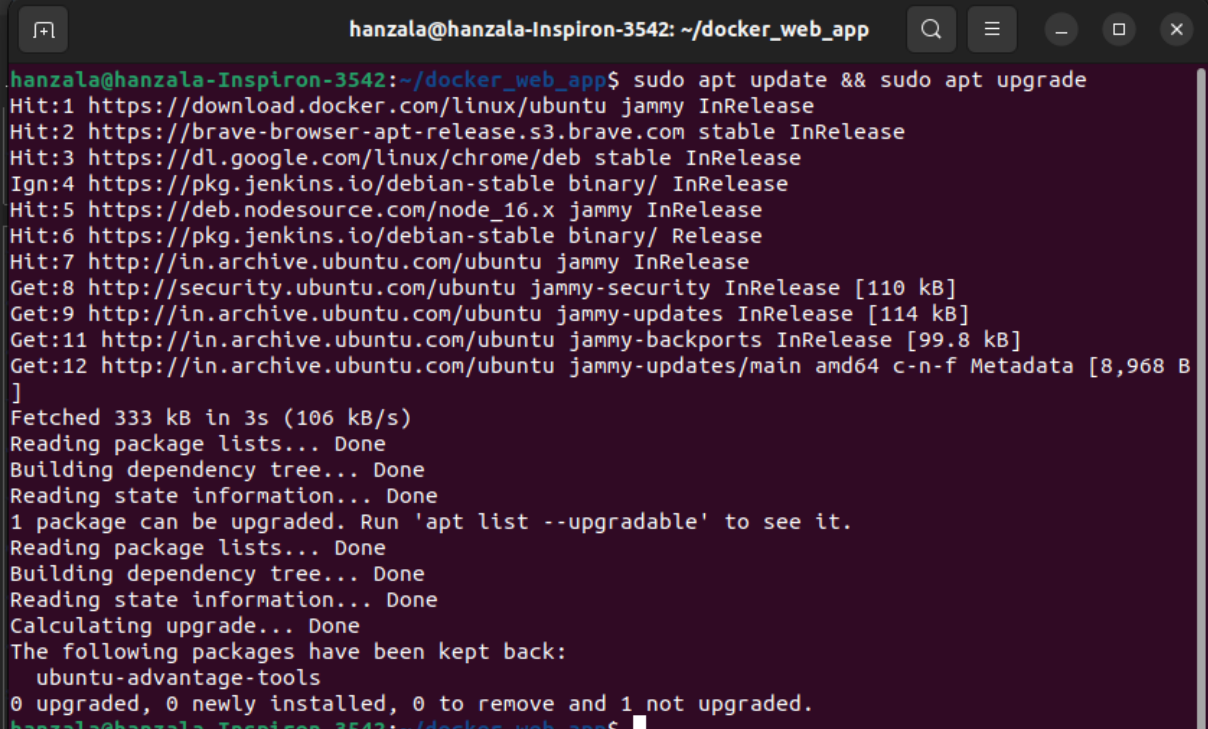
Roll no: 612006

## Experiment no. 08

**Aim:** Build, deploy and manage non web applications on Docker

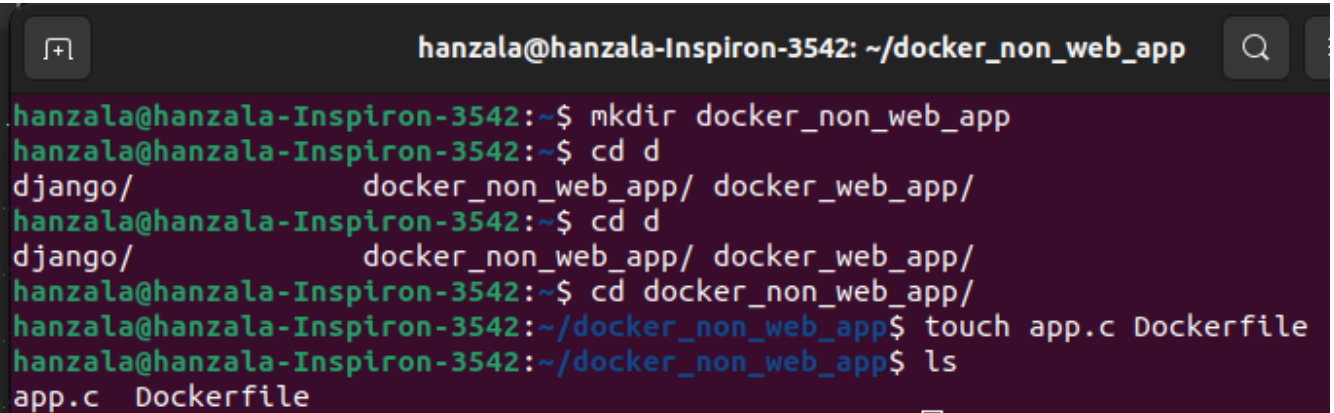
**Steps to build and deploy and manage non web application:**

- a. Fire up your terminal and run “sudo apt update && sudo apt upgrade”



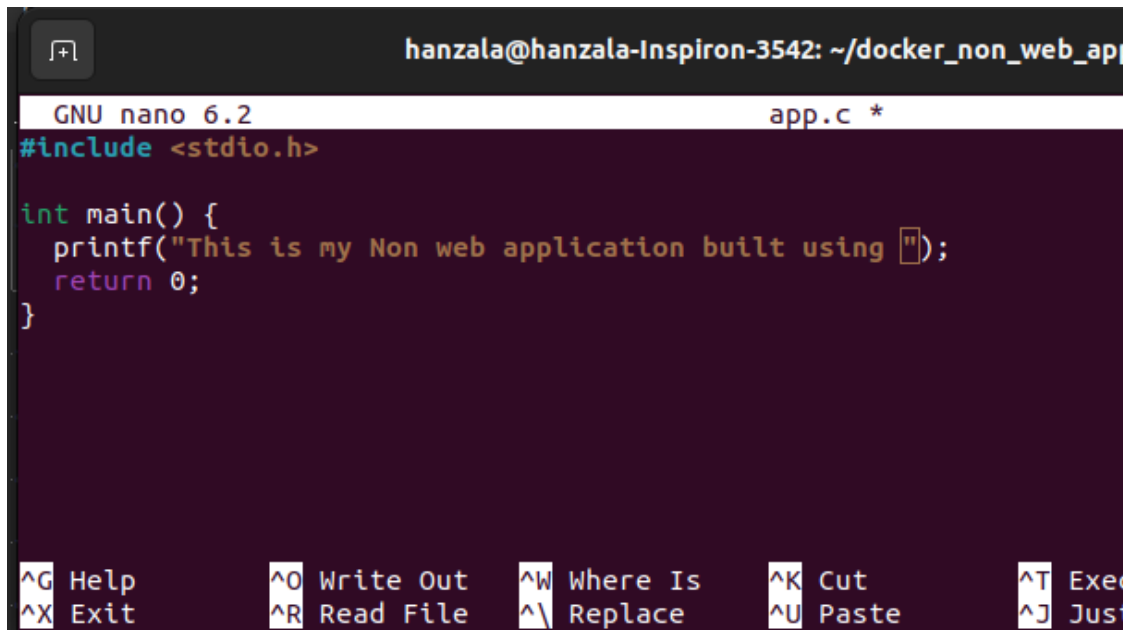
```
hanzala@hanzala-Inspiron-3542: ~/docker_web_app
hanzala@hanzala-Inspiron-3542:~/docker_web_app$ sudo apt update && sudo apt upgrade
Hit:1 https://download.docker.com/linux/ubuntu jammy InRelease
Hit:2 https://brave-browser-apt-release.s3.brave.com stable InRelease
Hit:3 https://dl.google.com/linux/chrome/deb stable InRelease
Ign:4 https://pkg.jenkins.io/debian-stable binary/ InRelease
Hit:5 https://deb.nodesource.com/node_16.x jammy InRelease
Hit:6 https://pkg.jenkins.io/debian-stable binary/ Release
Hit:7 http://in.archive.ubuntu.com/ubuntu jammy InRelease
Get:8 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:9 http://in.archive.ubuntu.com/ubuntu jammy-updates InRelease [114 kB]
Get:11 http://in.archive.ubuntu.com/ubuntu jammy-backports InRelease [99.8 kB]
Get:12 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 c-n-f Metadata [8,968 B]
Fetched 333 kB in 3s (106 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
1 package can be upgraded. Run 'apt list --upgradable' to see it.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
The following packages have been kept back:
  ubuntu-advantage-tools
0 upgraded, 0 newly installed, 0 to remove and 1 not upgraded.
hanzala@hanzala-Inspiron-3542:~/docker_web_app$
```

- b. Create a dir using mkdir and run “touch app.c Dockerfile” in side the dir you have created



```
hanzala@hanzala-Inspiron-3542: ~/docker_non_web_app
hanzala@hanzala-Inspiron-3542:~$ mkdir docker_non_web_app
hanzala@hanzala-Inspiron-3542:~$ cd d
django/          docker_non_web_app/  docker_web_app/
hanzala@hanzala-Inspiron-3542:~$ cd d
django/          docker_non_web_app/  docker_web_app/
hanzala@hanzala-Inspiron-3542:~$ cd docker_non_web_app/
hanzala@hanzala-Inspiron-3542:~/docker_non_web_app$ touch app.c Dockerfile
hanzala@hanzala-Inspiron-3542:~/docker_non_web_app$ ls
app.c  Dockerfile
```

- c. Using nano open app.php file and write the code given below



```
hanzala@hanzala-Inspiron-3542: ~/docker_non_web_app
GNU nano 6.2 app.c *
#include <stdio.h>

int main() {
    printf("This is my Non web application built using ");
    return 0;
}
```

Terminal window showing the nano text editor editing `app.c`. The code inside is a C program that prints a message. The terminal window title is `hanzala@hanzala-Inspiron-3542: ~/docker_non_web_app`. The nano editor status bar shows `GNU nano 6.2 app.c *`. The code is as follows:

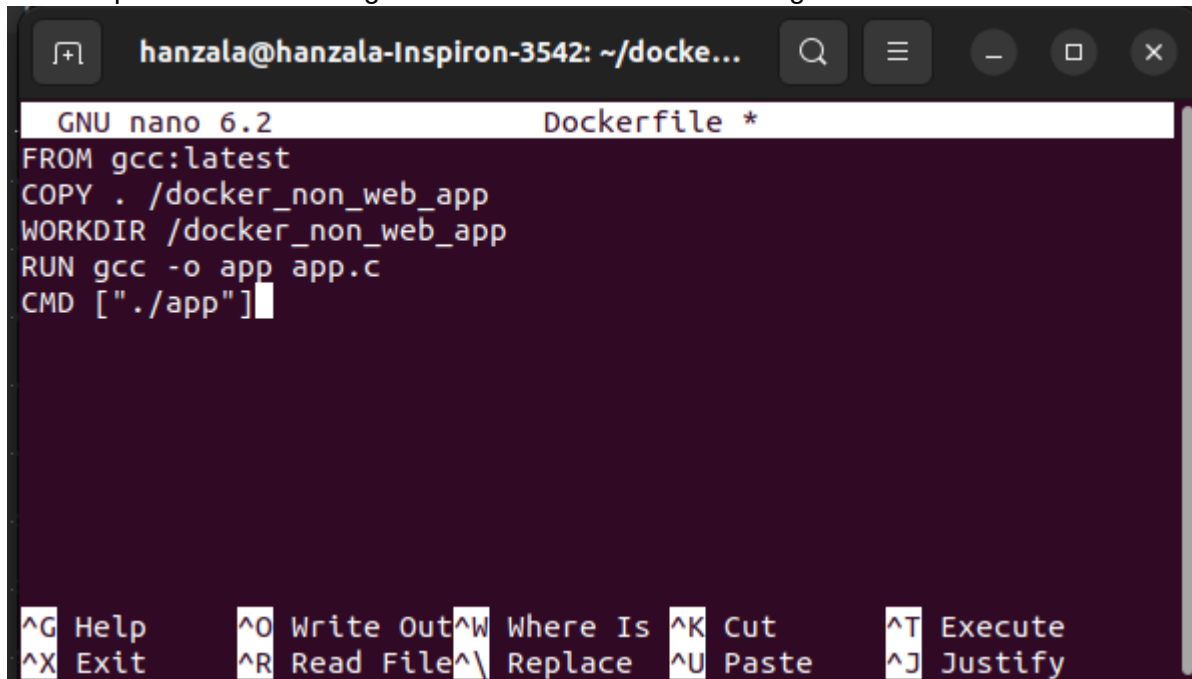
```
#include <stdio.h>

int main() {
    printf("This is my Non web application built using ");
    return 0;
}
```

The terminal window also shows the nano editor's command palette at the bottom with the following options:

- `^G` Help
- `^X` Exit
- `^O` Write Out
- `^R` Read File
- `^W` Where Is
- `^N` Replace
- `^K` Cut
- `^U` Paste
- `^T` Execute
- `^J` Justify

- d. Now open Dockerfile using nano and write down the code given below



```
hanzala@hanzala-Inspiron-3542: ~/docke...
GNU nano 6.2 Dockerfile *
FROM gcc:latest
COPY . /docker_non_web_app
WORKDIR /docker_non_web_app
RUN gcc -o app app.c
CMD ["./app"]
```

Terminal window showing the nano text editor editing `Dockerfile`. The code inside is a Dockerfile that builds and runs a C application. The terminal window title is `hanzala@hanzala-Inspiron-3542: ~/docke...`. The nano editor status bar shows `GNU nano 6.2 Dockerfile *`. The code is as follows:

```
FROM gcc:latest
COPY . /docker_non_web_app
WORKDIR /docker_non_web_app
RUN gcc -o app app.c
CMD ["./app"]
```

The terminal window also shows the nano editor's command palette at the bottom with the following options:

- `^G` Help
- `^X` Exit
- `^O` Write Out
- `^R` Read File
- `^W` Where Is
- `^N` Replace
- `^K` Cut
- `^U` Paste
- `^T` Execute
- `^J` Justify

- e. Now all things are set using the command “sudo docker build . -t non\_web\_app” our web app will start to build

```
hanzala@hanzala-Inspiron-3542: ~/docker_non_web_...
hanzala@hanzala-Inspiron-3542:~/docker_non_web_app$ sudo docker build . -t non_web_app
[sudo] password for hanzala:
Sending build context to Docker daemon 3.072kB
Step 1/5 : FROM gcc:latest
latest: Pulling from library/gcc
23858da423a6: Pull complete
326f452ade5c: Pull complete
a42821cd14fb: Pull complete
8471b75885ef: Pull complete
8ffa7aaef404: Pull complete
0dbd3d90c419: Pull complete
c8360ea64db4: Pull complete
65bba72ff1de: Pull complete
a615a380ba22: Pull complete
Digest: sha256:51c4c4a790c8c79f733f3dc3c99d494cff636dc0b569d13cb654ba28df986362
Status: Downloaded newer image for gcc:latest
```

- f. Check whether the images are built using “sudo docker images” and run docker images using “sudo docker run non\_web\_app”

```
hanzala@hanzala-Inspiron-3542:~/docker_non_web_app$ sudo docker images
[sudo] password for hanzala:
REPOSITORY          TAG             IMAGE ID        CREATED         SIZE
non_web_app          latest          811ec27e046a    5 minutes ago   1.27GB
web_app              latest          1d7369efbb27    46 minutes ago  484MB
php                  latest          aa24992dd020    9 hours ago     484MB
gcc                  latest          feaa519db663    3 weeks ago     1.27GB
hello-world          latest          feb5d9fea6a5    12 months ago   13.3kB
hanzala@hanzala-Inspiron-3542:~/docker_non_web_app$ sudo docker run non_web_app
This is my Non web application built using hanzala@hanzala-Inspiron-3542:~
```

**Conclusion:** We successfully build and run our non web application using docker