||Jai Sri Gurudev||

ADICHUNCHANAGIRI UNIVERSITY

BGS Institute of Technology

BG Nagara – 571448, Nagamangala Taluk, Mandya District, Karnataka (INDIA)
(Approved by AICTE, New Delhi & Recognized by Govt. of Karnataka, NBA Accredited)

ARTIFICIAL INTELLIGENCE
AND
MACHINE LEARNING

DevOps LAB MANUAL (22AML66)

For
VI Semester B.E. AI&ML

Prepared by:

1.Mr.ANILKUMAR K B, Asst.Prof

Approved by:

Head of the Department

DPARTMENT OF ARTIFICIAL INTELLIGENECE AND MACHINE LEARNING

VISION:

To innovate in the fields of artificial intelligence and machine learning for achieving sustainable development and to meet value based socio-economic needs.

MISSION

M1: Advance the capabilities of AI systems and harness their potential to improve various aspects of human life.

M2: By aligning artificial intelligence and machine learning with sustainable development goals to create a world where technology serves as a catalyst for positive change.

M3: Crafting technologically entrepreneurial operations, innovative skill and development for socio-economic requirements.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

PEO1: Analyze the requirements, realize the technical specification and design the Engineering solutions by applying artificial intelligence theory and principles.

PEO2: Make successful career in higher Studies /industry / research.

PEO3: Be life-long learning and should be able to work on multi-disciplinary projects.

PEO4: Be Competent for effective communication, in management and in professional skills and ethics.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO1: Ability to apply the concepts, principles and practices of Artificial Intelligence and Machine Learning and critically evaluate the results with proper arguments, selection of tools and techniques when subjected to loosely defined scenarios.

PSO2: Ability to use Artificial Intelligence and Machine Learning models on data for enabling better decision making.

SYLLABUS

Course objectives:

i. Understand DevOps as a practice, methodology and process for fast collaboration, integration and communication between Development and Operations team.
ii. Learn common Infrastructure Servers, Availability and Scalability
iii. Describe how AWS DevOps is used for Identity Access Management.
iv. Understand the requirements of Configuration Management using Ansible
v. Understand Docker Containerization, Micro service Architecture.

Programs List:

1 Write code for a simple user registration form for an event.

2 Explore Git and GitHub commands

3 Practice source code management on GitHub. Experiment with the source code written in exercise 1.

4 Jenkins installation and setup, explore the environment.

5 Demonstrate continuous integration and development using Jenkins

6 Explore Docker commands for content management

7 Develop a simple containerized application-using Docker.

8 Integrate Kubernetes and Docker

9 Automate the process of running containerized applications developed in exercise 7 using Kubernetes

10 Install and explore Selenium for automated testing.

11 Write a simple program in JavaScript and perform testing using Selenium.

12 Develop test cases for the above-containerized application using Selenium.

Course Outcomes:

At the end of the course, the student will be able to:

1. Demonstrate the DevOps culture by illustrating application‘s cloud infrastructure and configuration management with Ansible. (PO-2, 3, 5, 9, PSO-3)
2. Apply the DevOps pipeline process starting with continuous integration and continuous deployment principles. (PO-2, 3, 5, 9, PSO-3)
3. Demonstrate how to create and run a container from a Docker file and deploy a complex application on Kubernetes. (PO-2, 3, 5, 9, PSO-3)

INSTRUCTIONS TO STUDENTS:

Computer Lab Safety Rules for Protecting Equipment: -

- ✓ Turn off the machine once you are done using it.
- ✓ Do not plug in external devices without scanning them for computer viruses.
- ✓ Try not to touch any of the circuit boards and power sockets when a device is connected to them and switched on.
- ✓ Always maintain an extra copy of all your important data files.

HISTORY OF DEVOPS

DevOps, a portmanteau of "development" and "operations," is a software development methodology that emphasizes collaboration and communication between development and operations teams. The goal of DevOps is to create a more efficient and reliable software development process that can rapidly deliver high-quality software.

The origins of DevOps can be traced back to the early 2000s, when agile software development methodologies were becoming popular. Agile development emphasizes close collaboration between developers and business stakeholders, and emphasizes iterative, incremental development.

In 2008, Patrick Debois organized the first "DevOpsDays" conference in Ghent, Belgium. This conference brought together developers and operations professionals to discuss the challenges of integrating their work and to explore ways to improve collaboration.

Around the same time, companies like Amazon and Netflix were pioneering new approaches to software development and deployment, including the use of cloud computing and continuous delivery. These companies were able to rapidly iterate on their software and deploy changes to production multiple times a day.

As DevOps continued to gain momentum, tools and practices emerged to support the methodology. These include tools for continuous integration and continuous delivery, as well as approaches like infrastructure as code and site reliability engineering.

Today, DevOps is widely recognized as a best practice for software development and operations. Companies of all sizes and industries are adopting DevOps to increase the speed and reliability of their software development process.

Beyond Syllabus: BASIC UBUNTU COMMANDS:

1. **cat: cat** [filename] - to display the contents of a file, and cat > [filename] - to create file,cat

>> [filename] append contents to existing file, cat [newfilename] > [existingfilename] - to create a new file and append its contents to an existing file.

Ctrl +d -save file

2. **touch: touch** [filename] - to create a blank file.

3. **nano: nano** [filename] - to create and edit a file.

Ctrl +x, y -tos ave file

4. **vi/vim: vi** [filename] or vim [filename] - to create and edit a file.

Esc: wq- to save file

5. **ls: ls -** to list all files and directories in the current directory, ls -a - to show hidden files and directories, and ls -la - to show all files and directories with details.

6. **cd: cd** [directoryname] - to change directory to a specified directory.

7. **pwd: pwd -** to display the current working directory.

8. **mkdir: mkdir** [directoryname] - to create a directory with a specified name, and mkdir -p [directoryname1]/[directoryname2] - to create multiple directories at once.

9. **cp: cp** [sourcefile] [destinationfile] - to copy a file from source to destination.

10. **mv: mv** [sourcefile] [destinationfile] - to move a file from source to destination.

11. **mv: mv** [oldfilename] [newfilename] - to rename a file.

12. **rm: rm** [filename] - to remove a file.

13. **tree: tree -** to display the directory structure in a tree-like format.

14. **rm -rf: rm -rf** [directoryname] - to remove a directory and all its contents recursively.

15. **grep: grep** [searchterm] [filename] - to search for a specific string in a file and print the matching lines.

16. **less: less** [filename] - to display the output of a command or file one page at a time.

17. **head: head** [filename] - to display the first 10 lines of a file.

18. **tail: tail** [filename] - to display the last 10 lines of a file.

19. **sort: sort** [filename] - to display the contents of a file in alphabetical or numerical order.

20. **sudo: sudo** [command] - to execute a command with root privileges.

21. **apt: apt -get** [option] [package] - to manage packages in Ubuntu. Common options include install, update, and remove.

22. **dpkg: dpkg** [option] [package_file] - to install or manage a package file.

23. **ifconfig: ifconfig** [network_interface] - to display network interface configuration.

EXPERIMENT NO: 1. Write code for a simple user registration form for an event.
Aim: Write code for a simple user registration form for an event.

1. **Open Terminal and Login to Remote User:**
   - If you are logging into a remote server, use:

     ssh username@hostname

     Replace username with your remote username and hostname with your server address.

2. **Become the Superuser (if needed):**
   - Once logged in, run the following command to get root access:
     sudo su

   - Enter your password **(bgsit123).**

3. **Update Ubuntu:**
   - Update your system by running:

     apt-get update -y

4. **Install Apache Web Server**:
   - To install the Apache2 web server, use:
     apt-get install apache2 -y

5. **Start Apache Service:**
   - After installing Apache, start the web service by running:
     service apache2 start

6. **Navigate to the Web Directory:**
   - Move to the appropriate directory where web files are stored:

     cd /var/www/html

7. **Create registration.html:**
   - Now, create the registration form by opening the registration.html file using vi or your preferred text editor:

     vi registration.html

   - **INSERT** the HTML code you have provided into **registration.html**:
     html

```
<html>
<head>
   <title>Registration Form</title>
</head>
<body>
   <h2 align="center">Registration Form</h2>
   <form action="success.html" method="post">
      <table border="0" align="center">
         <tbody>
```

```
<tr>
  <td><label for="id">Id: </label></td>
  <td><input id="id" maxlength="50" name="id" type="text" /></td>
</tr>
<tr>
  <td><label for="name">Name: </label></td>
  <td><input id="name" maxlength="50" name="name" type="text" /></td>
</tr>
<tr>
  <td><label for="course">Course: </label></td>
  <td><input id="course" maxlength="50" name="course" type="text" /></td>
</tr>
<tr>
  <td><label for="branch">Branch: </label></td>
  <td><input id="branch" maxlength="50" name="branch" type="text" /></td>
</tr>
<tr>
  <td><label for="rollno">Rollno: </label></td>
  <td><input id="rollno" maxlength="50" name="rollno" type="text" /></td>
</tr>
<tr>
  <td><label for="email">Email Address: </label></td>
  <td><input id="email" maxlength="50" name="email" type="text" /></td>
</tr>
<tr>
  <td><label for="username">Username: </label></td>
  <td><input id="username" maxlength="50" name="username" type="text" /></td>
</tr>
<tr>
  <td><label for="aboutus">About Us: </label></td>
  <td valign="middle" align="center"><textarea name="aboutus"></textarea></td>
</tr>
<tr>
  <td><label for="password">Password: </label></td>
  <td><input id="password" maxlength="50" name="password" type="password" /></td>
</tr>
<tr>
  <td align="right"><input name="Submit" type="submit" value="Submit" /></td>
</tr>
</tbody>
</table>
</form>
</body>
</html>
```

- Save the file and exit vi by pressing **Esc**, typing **:wq**, and hitting **Enter**.

8. **Create the success.html Page**:
   - Now, create the success.html page using the following:

vi success.html

-INSERT the following HTML code for a **successful** registration message:

```
html
<html>
<head>
```

```
    <title>Registration Success</title>
  </head>
  <body>
    <h2>Registration Successful!</h2>
    <p>Thank you for registering. You will receive further details soon.</p>
  </body>
  </html>
```

- Save and exit vi the same way as before.

OUT PUT:

- **Access the Registration Form:**
  - Open any web browser on your local machine and go to:

    http://localhost/registration.html

(**Testing the Form: When you submit the form, the browser will redirect to the success.html page, which will display the registration success message.)**

EXPERIMENTNO:2. Explore Git and GitHub commands

Aim: Explore Git and GitHub commands

DESCRIPTION:

Git and GitHub are two of the most popular tools used for version control and collaboration in software development.

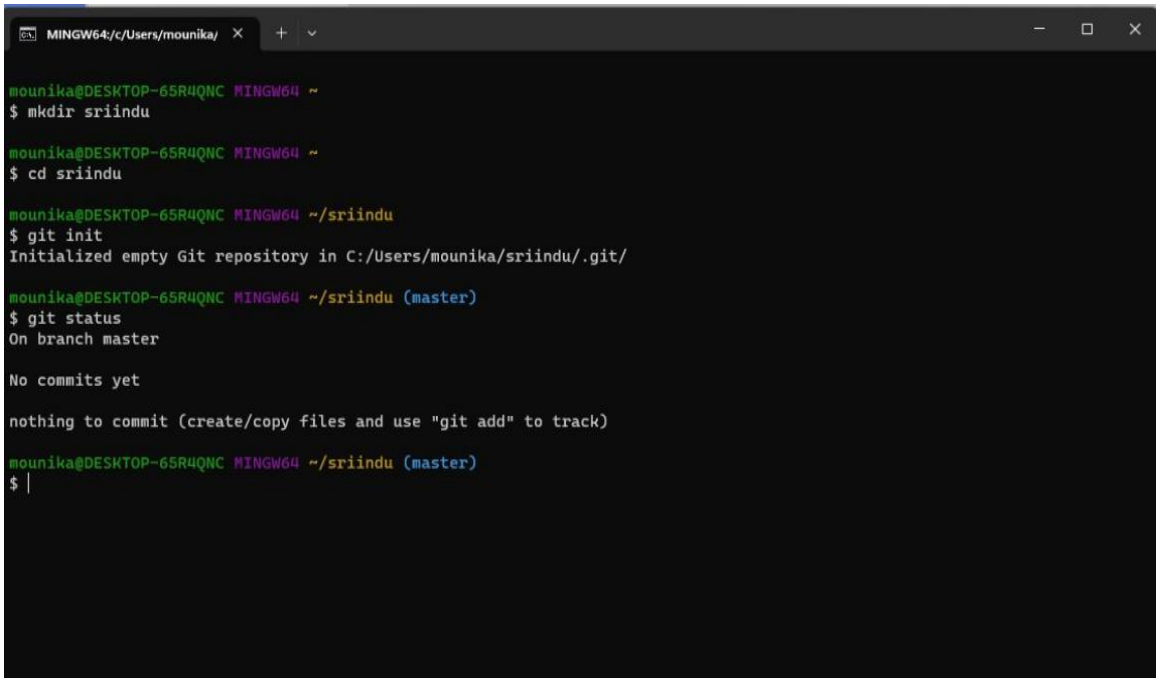
Here are some common Git and GitHub commands:

1. Initializing a Git repository:$**git init**
2. Checking the status of your repository:$**git status**
3. Adding files to the stage:$**git add** <file-name>
4. Committing changes:$**git commit -m "commitmessage"**
5. Checking the commit history:$**git log**
6. Undoing changes:$**git checkout** <file-name>
7. Creating a new branch:$**git branch** <branch-name>
8. Switching to a different branch:$**git checkout**<branch-name>
9. Merging two branches:$**git merge<branch-name>**
10. Pushing changes to are more repository:$**git push origin**<branch-name>
11. Cloning a repository from GitHub:$**git clone** <repository-url>
12. Creating a pull request
13. GitHub: **GototherepositoryonGitHub,selectthebranchyouwanttomergeandclickthe "Newpullrequest" button.**

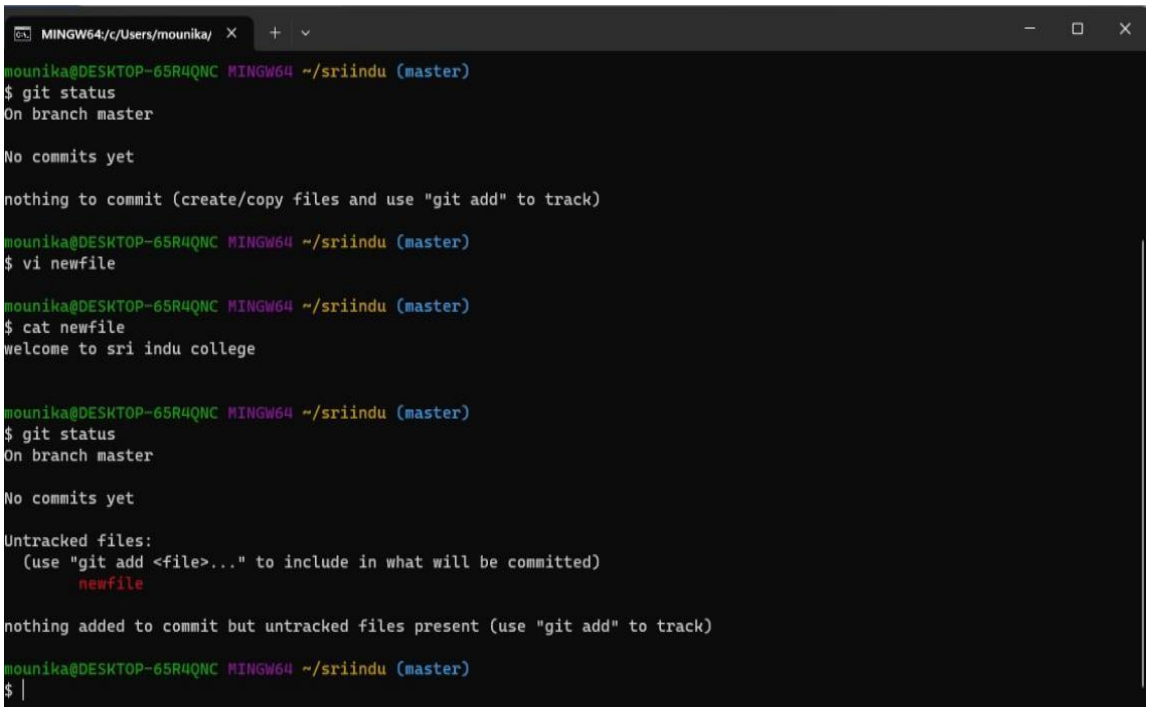
These are just a few of the many Git and GitHub commands available.

There are many other Git commands and functionalities that you can explore to suit your needs.

Open Git bash
Create directory**:-$mkdir USN**
Enter into directory**:-$cd bgsitusn**
Initializing a Git repository: **$ git init**
Checking the status of your repository: **$ git status**



Create new file $vi newfile

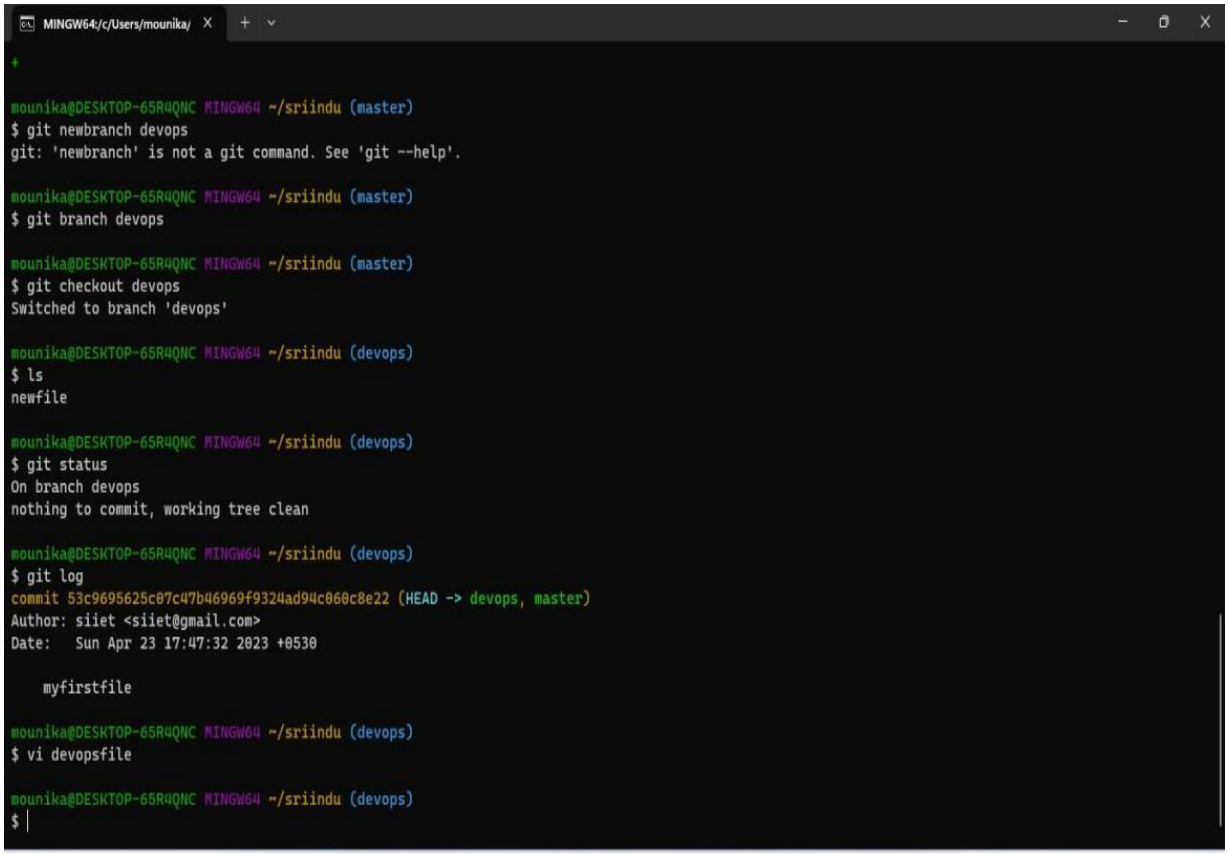


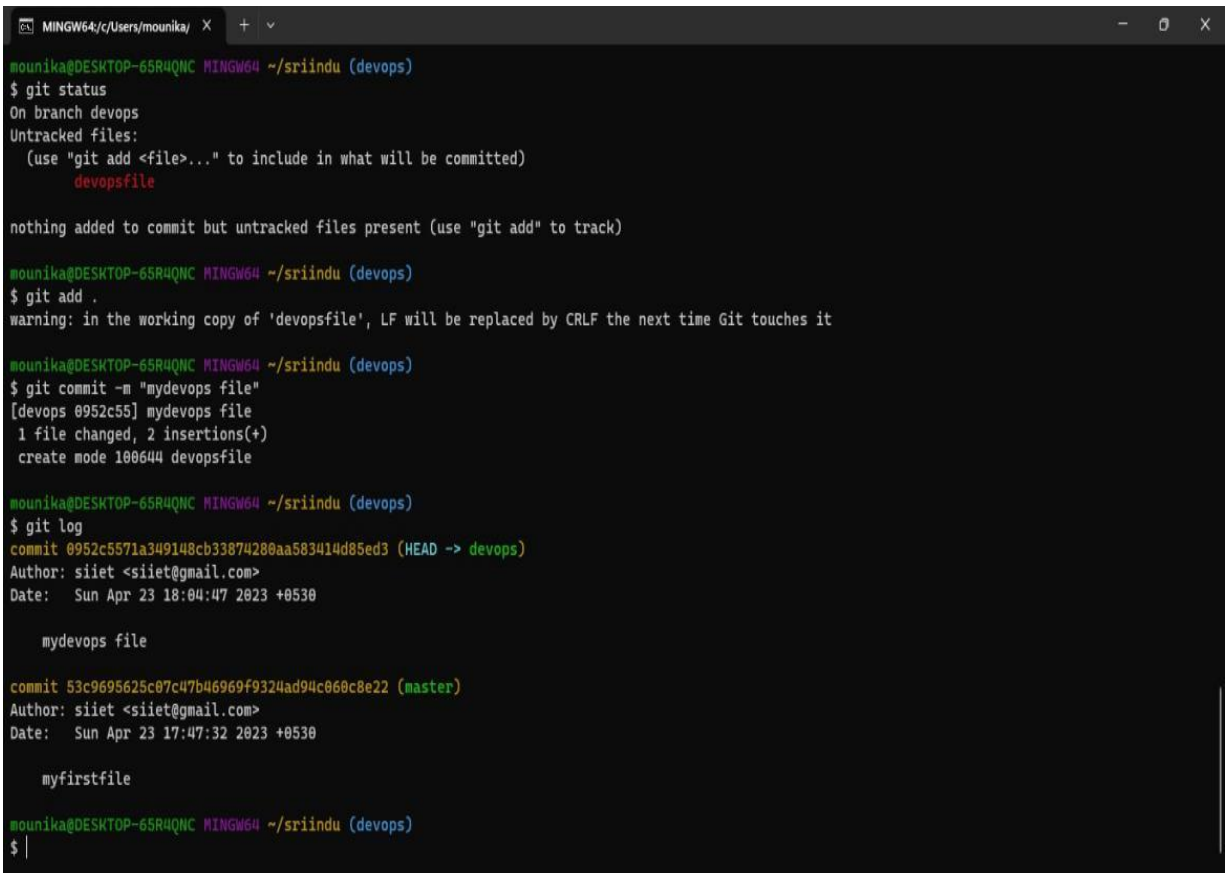
Adding files to the stage:$ git add newfile
Committing changes: $ git commit -m "commit message"

Creating a newbranch:$ git branch devops

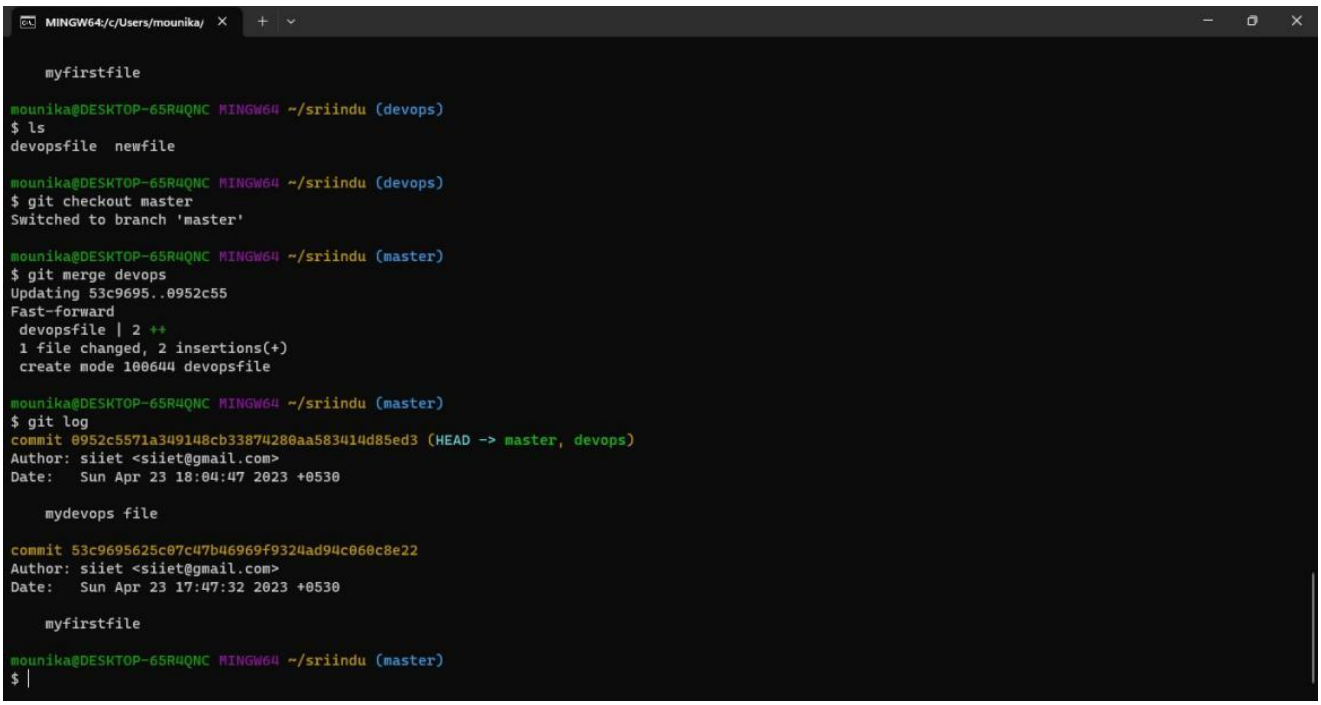
Switching to a different branch:$ git checkout devops

Create devops file in devops branch and add to stage , commit same file

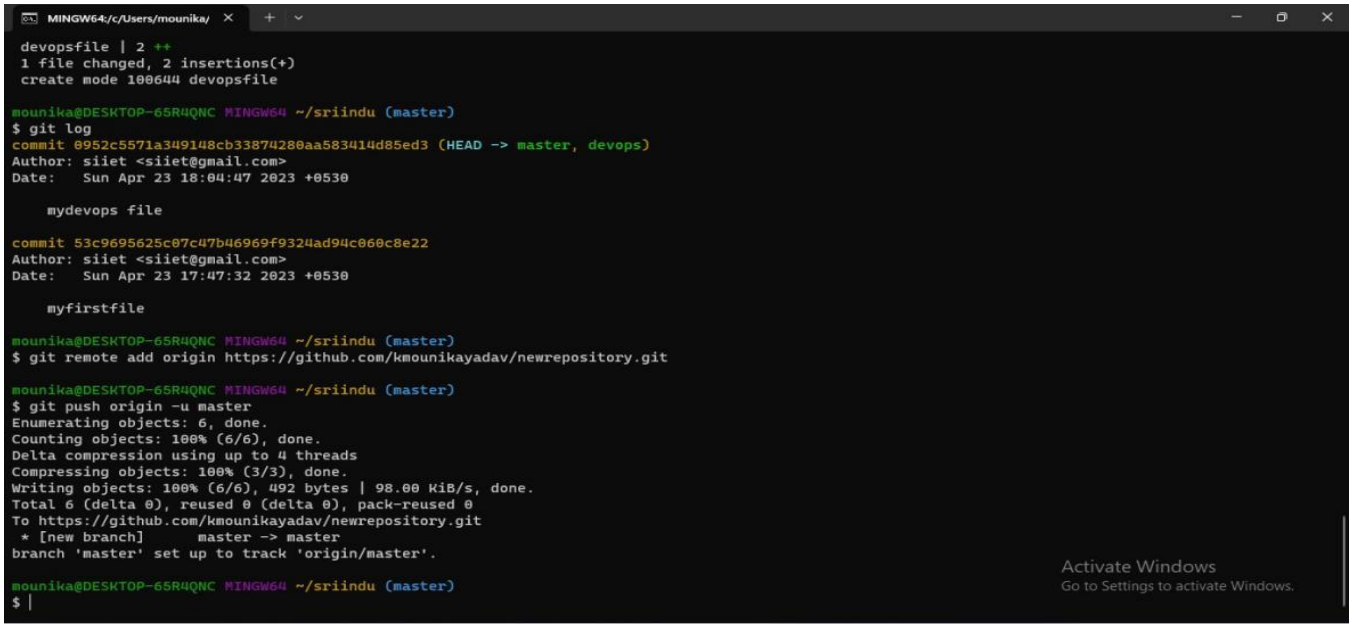




Checkout from devops branch and enter into master (default) branch and perform merge
Switching to a different branch**:$git checkout master**
Merging two branches**:$ git merge devops**



git init
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/anilkumarkb92/aiml.git
git push -u origin main



Cloning a repository from GitHub**:$ git clone <repository-url (https://github.com/anilkumarkb92/aiml.git)>**
`

EXPERIMENTNO: 3. Practice Source code management on GitHub.

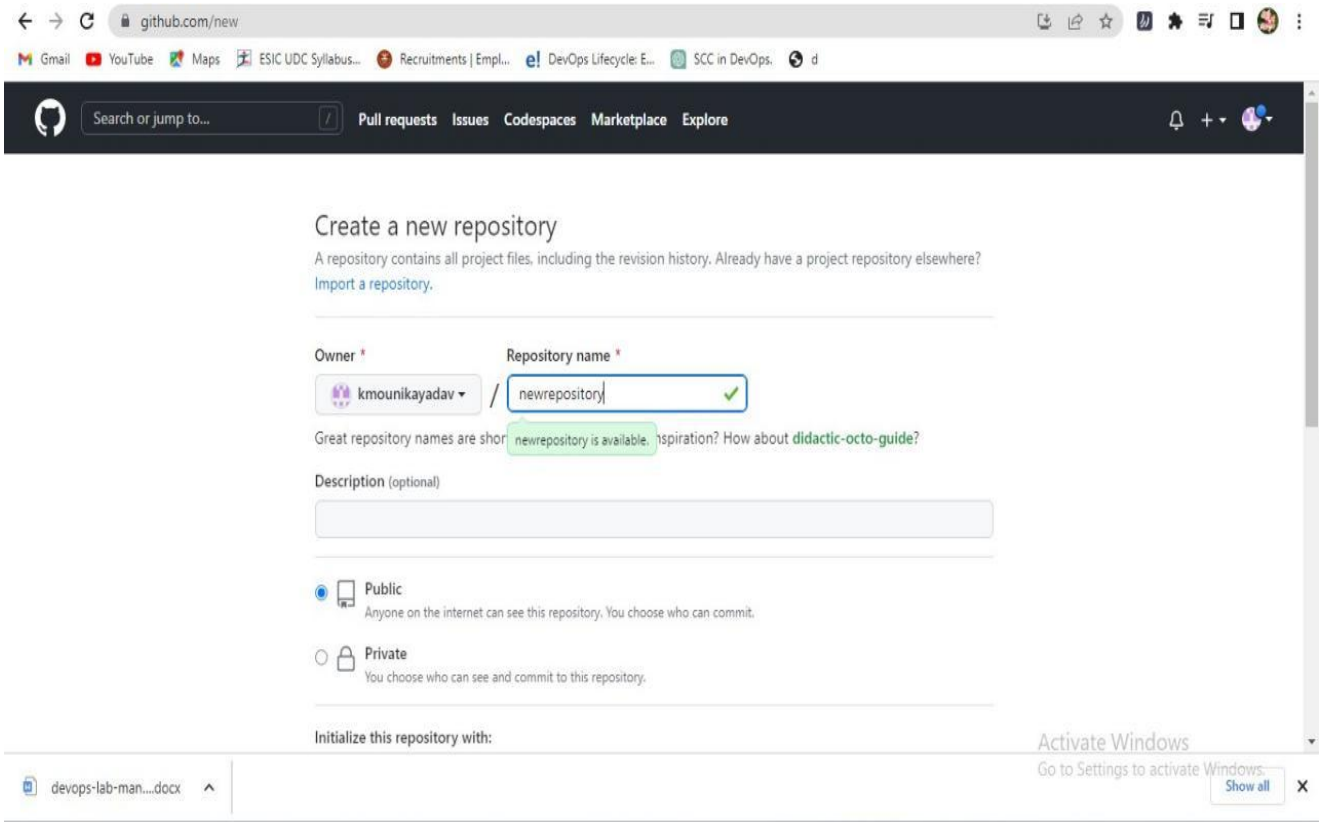
Aim: Practice Source code management on GitHub.

Description:

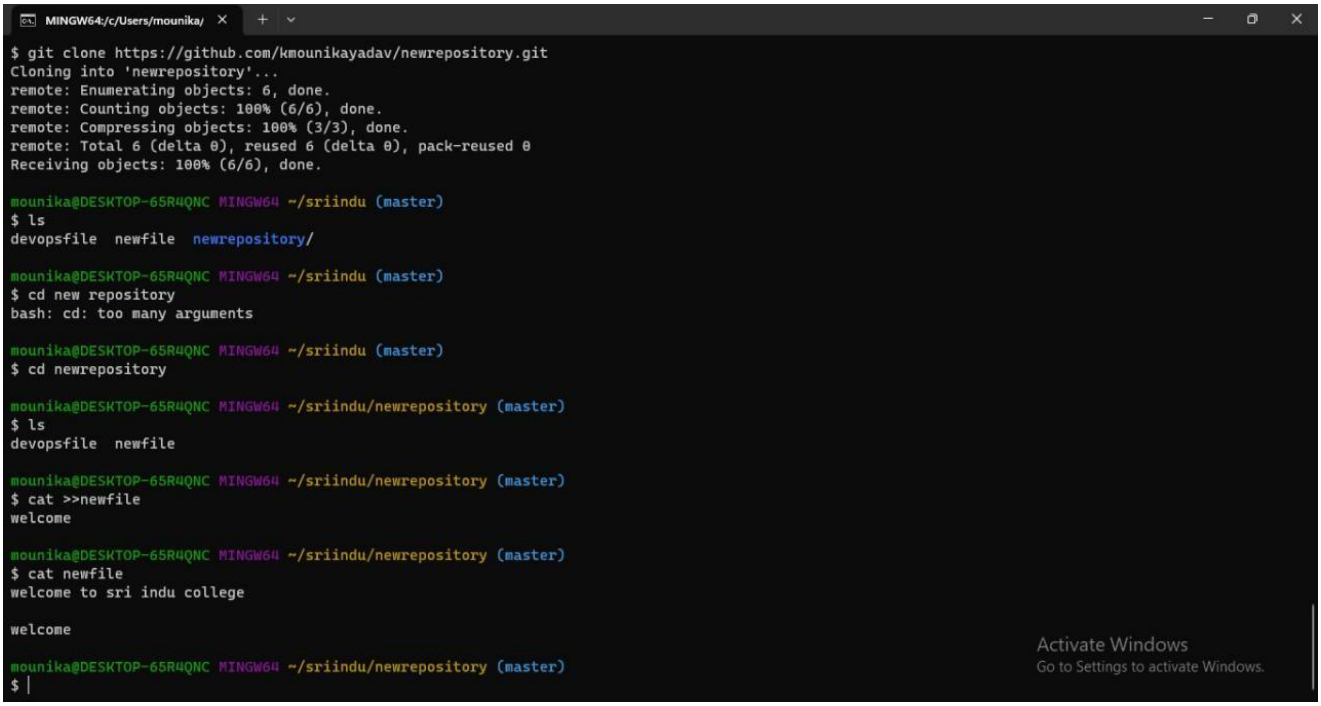
To practice source code management on GitHub, you can follow these steps:

- Create a GitHub account if you don't already have one.
- Create a new repository on GitHub.
- Clone the repository to your local machine: $ git clone <repository- url>
- Move to the repository directory:$cd<repository-name>
- Create a new file in the repository and add the source code file.
- Stage the changes:$git add<file-name>
- Commit the changes: $ git commit -m "Added source code for a simple user registration form"
- Push the changes to the remote repository:$ git push origin master
- Verify that the changes are reflected in the repository on GitHub. These steps demonstrate how to use GitHub for source code management.

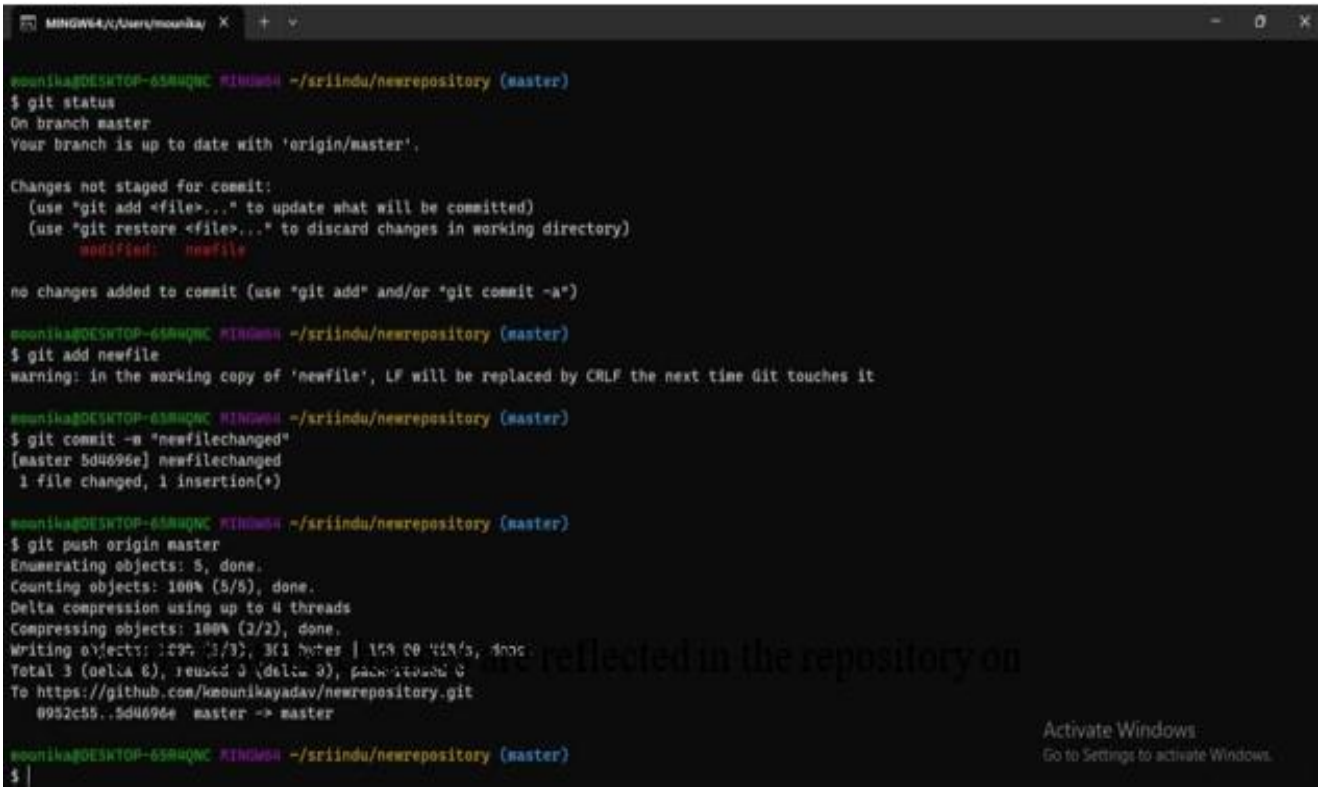
You can use the same steps to manage any source code projects on GitHub. Additionally, you can also explore GitHub features such as pull requests, code review, and branch management to enhance your source code management workflow.



Clone the repository to your local machine: **$ git clone <repository-url>**
Move to the repository directory: **$ cd new repository**
Create a new file in the repository and add the source code



- Stage the changes**:$git add newfile**
- Commit the changes: **$ git commit -m** "Added source code for a newfile"
- Push the changes to the remote repository**:$git push origin master**



EXPERIMENTNO:4. Jenkins installation and setup, explore the environment
Aim: Jenkins installation and setup, explore the environment.

DESCRIPTION

Jenkins is a popular open- source tool for Continuous Integration and Continuous Deployment (CI/CD) in software development. Here are the steps to install and setup Jenkins: Download and install Jenkins:

- Download the Jenkins package for your operating system from the Jenkins website.
- Follow the installation instructions for your operating system to install Jenkins.

Start the Jenkins service:

- On Windows, use the Windows Services Manager to start the Jenkins service.
- On Linux, use the following command to start the Jenkins service:

$ sudo service jenkins

Start Access the Jenkins web interface:

- Open a web browser and navigate to http://localhost:8080toaccess the Jenkins web interface.
- If the Jenkins service is running, you will see the Jenkins login page. Initialize the Jenkins environment:
- Follow the instructions on the Jenkins setup wizard to initialize the Jenkins environment.
- This process involves installing recommended plugins, setting up security, and creating the first admin user.

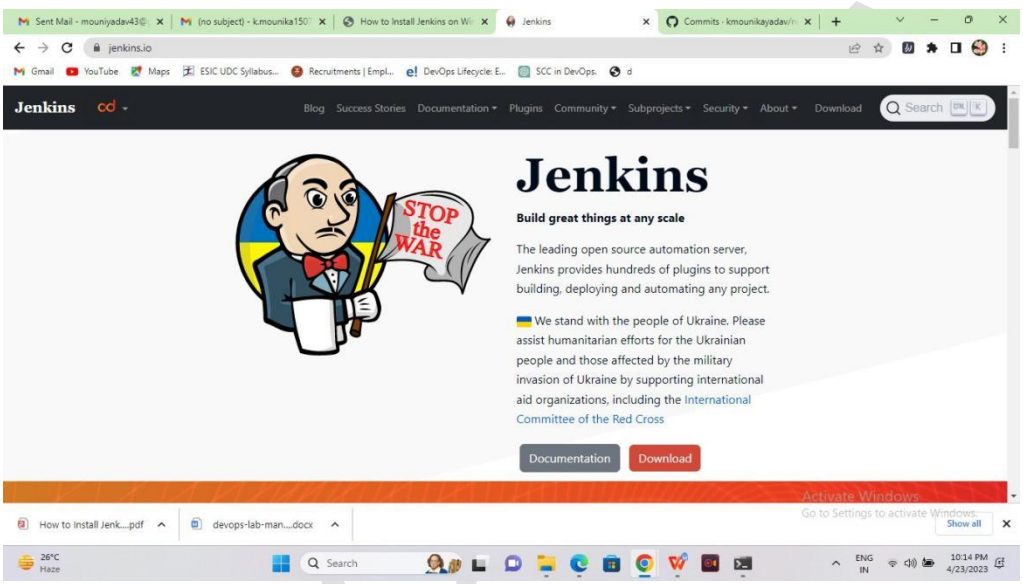
Explore the Jenkins environment:

- Once the Jenkins environment is set up, you can explore the various features and functionalities available in the web interface.
- Jenkins has a rich user interface that provides access to features such as build history, build statistics, and system information.
-

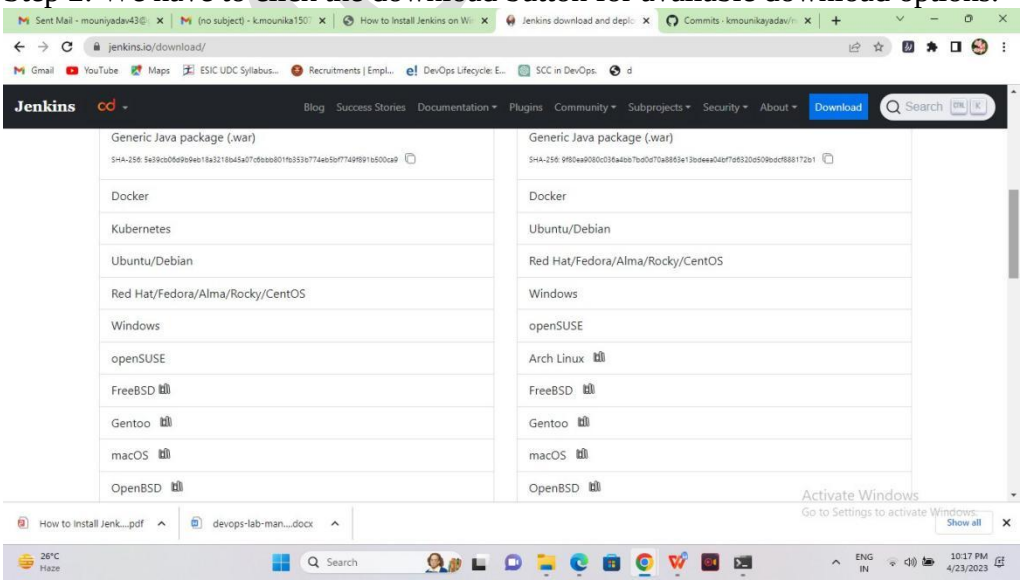
NOTE: These are the basic steps to install and set up Jenkins. Depending on your use case, you may need to customize your Jenkins environment further. For example, you may need to configure build agents, setup build pipelines, or integrate with other tools. However, these steps should give you a good starting point for using Jenkins for CI/CD in your software development projects.

Installing Jenkins over Windows

Step 1: First, we have to visit jenkins.io as shown in the figure below:

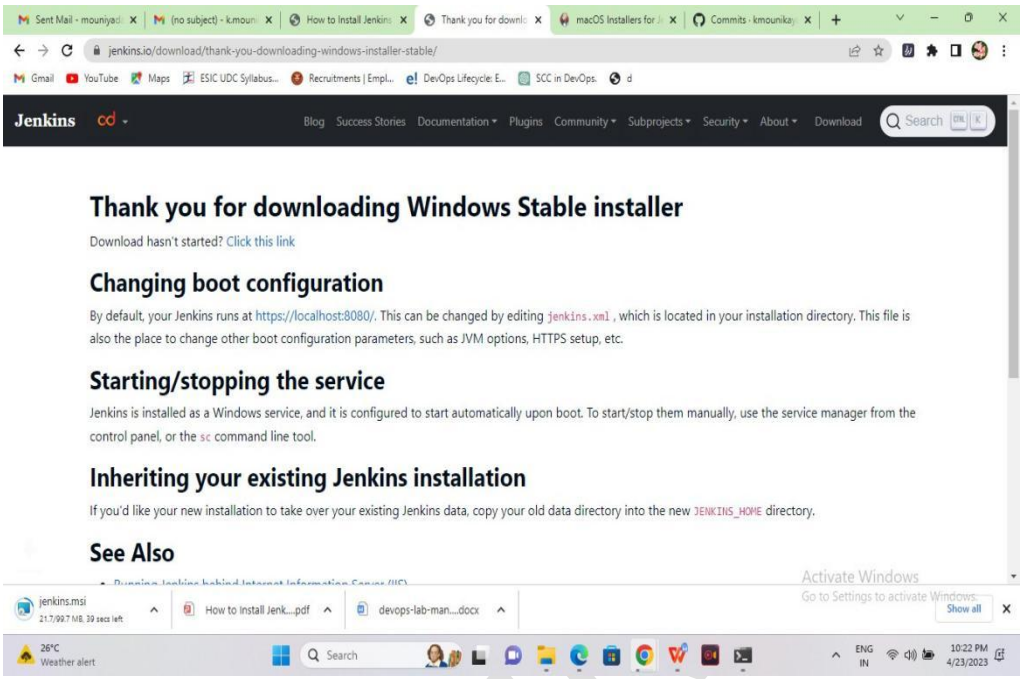


Step 2: We have to click the download button for available download options.



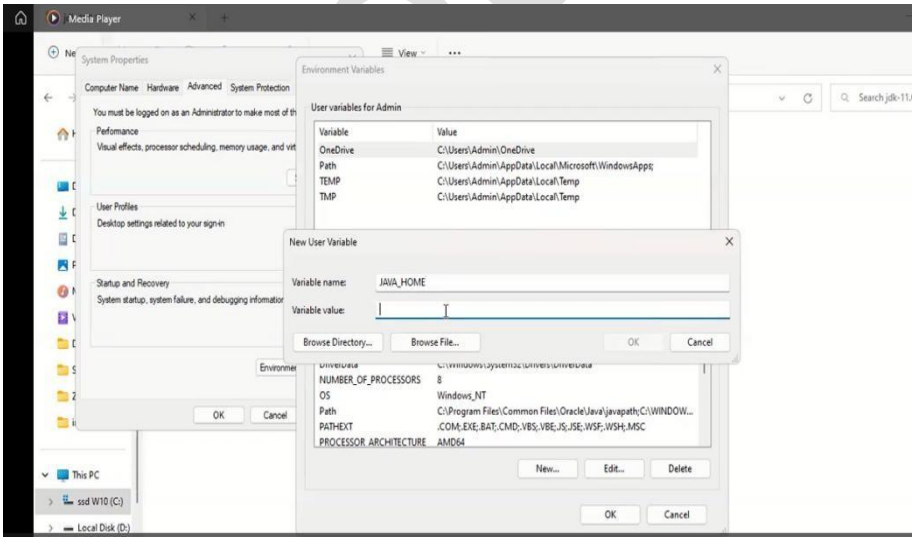
Step 3: We will get a download page as shown above where we have windows option and also generic package (war) option. We can install Jenkins on Windows by both of these options.

Step 4: After download any of the two we get the following acknowledgment page.

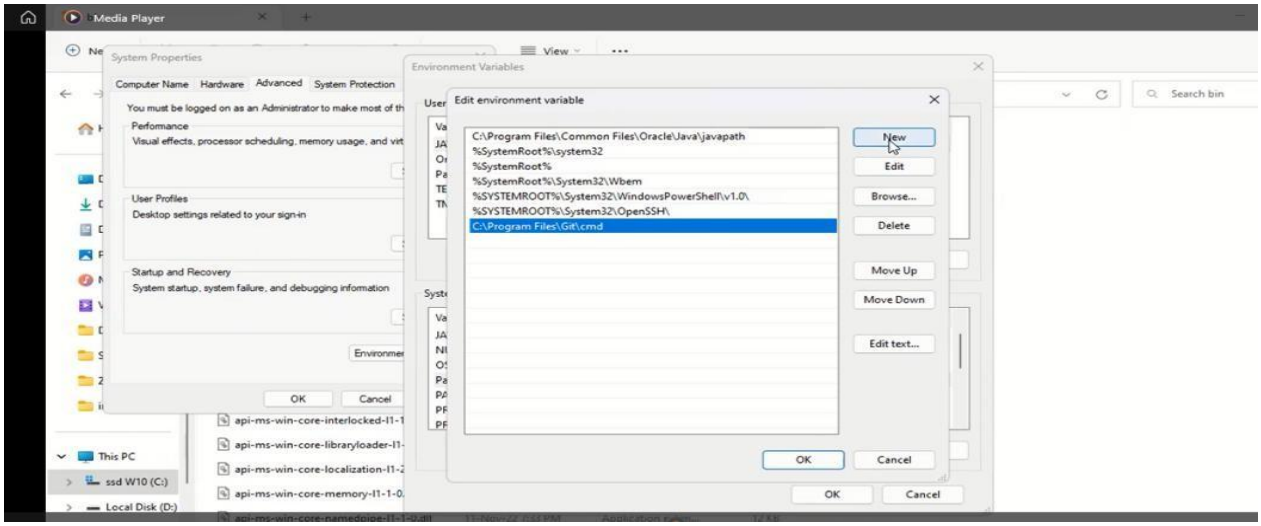


Step 5: Checking Java prerequisite: Before installing Jenkins we have to check Java installation in our system.

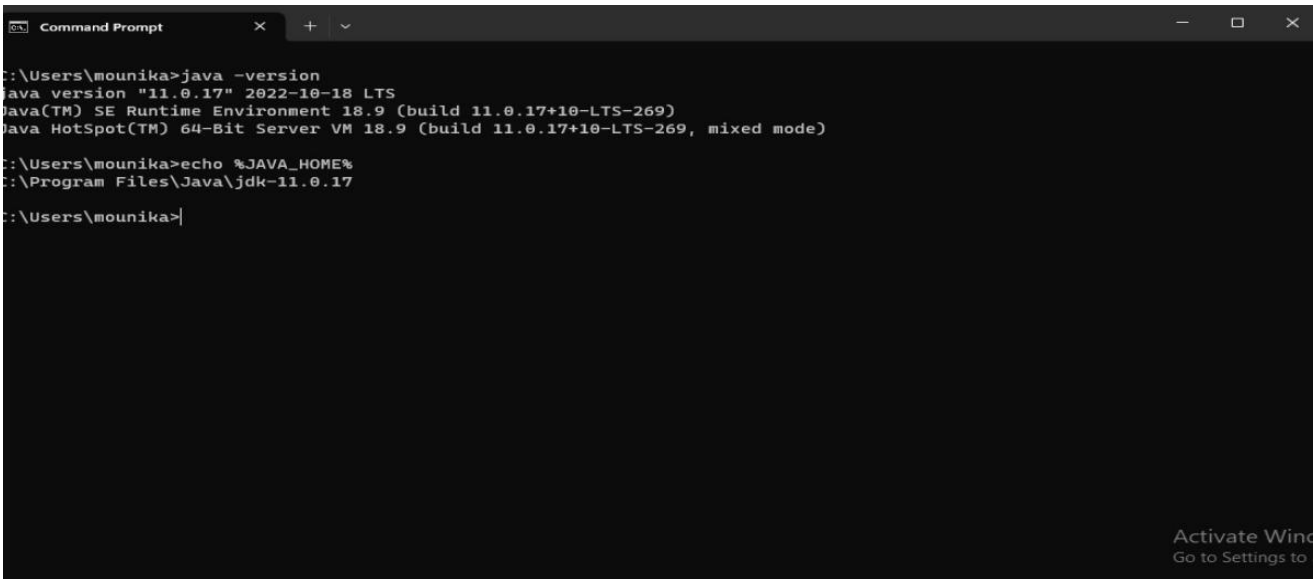
Step 6: Check and install suitable Java version on your system (java 11 latest version) and set environ variable as shown below:



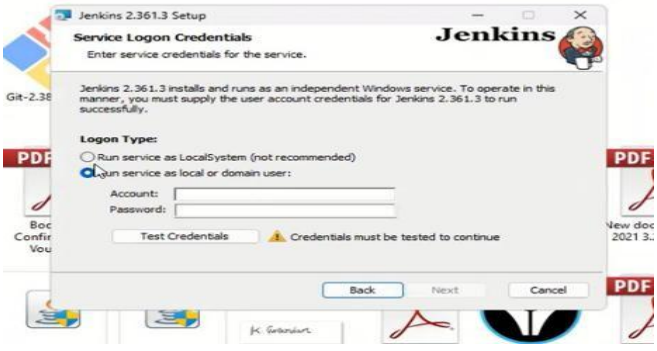
Step 7: Also append Java path in the path variable there with Java path up to bin subfolder.



Step 8: Reboot your system after setting environment variables
Step 9: When the system comes up again then check the Java version in your command prompt with the command java – version like this:

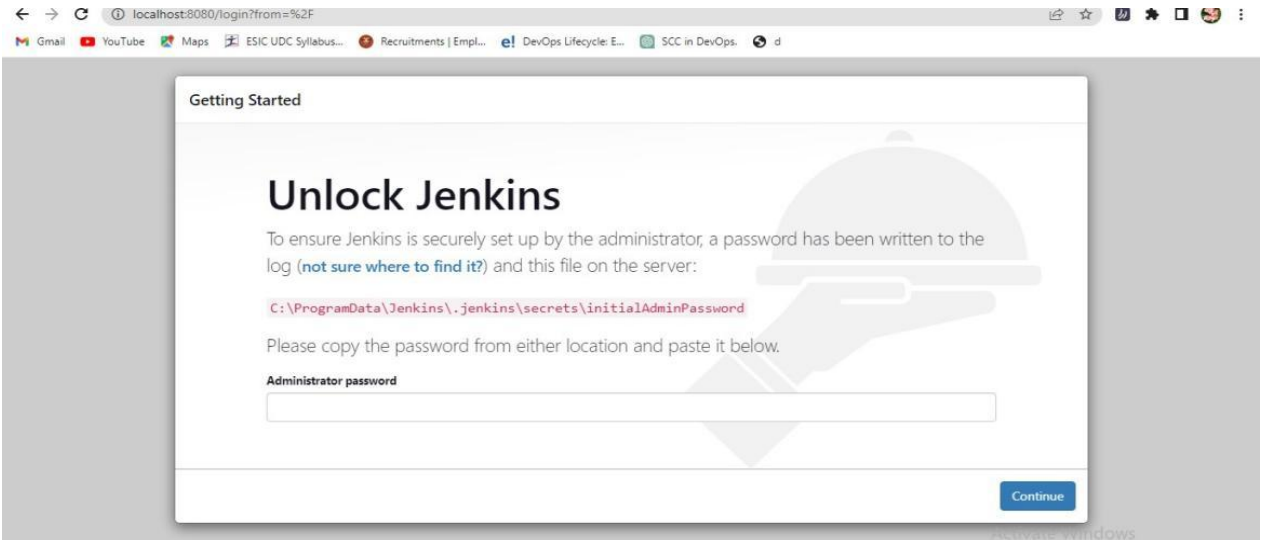


Step 10: Installing Jenkins with MSI installer option

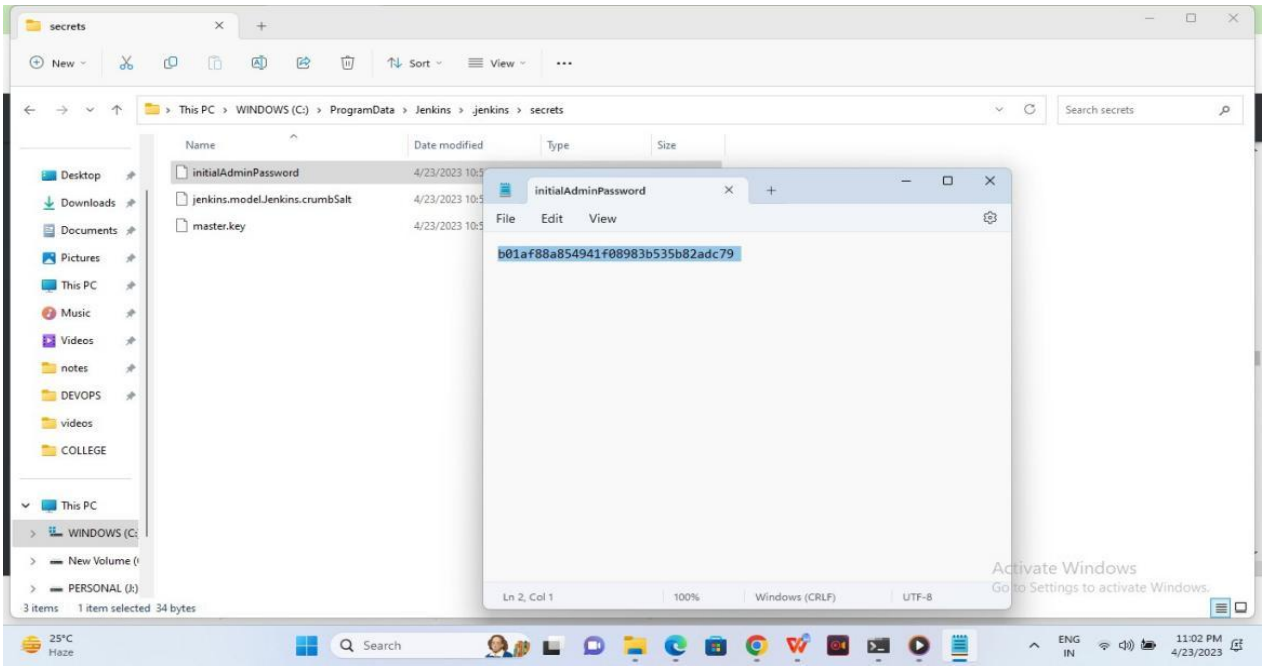


After this installation is finished Jenkins will be available as windows service.

Step 11: Installation of a common part after any of the two options of installation done up till now:
Now by any of the two above installation we have Jenkins service available and running and our next task is two open a browser

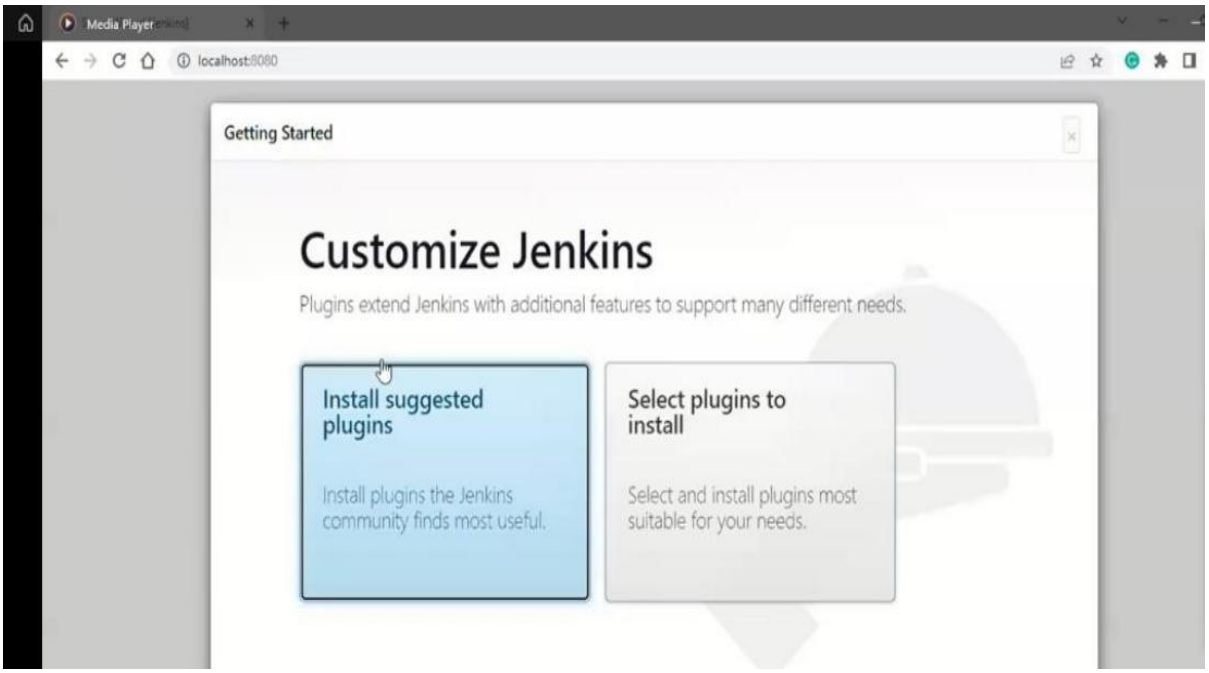


Step 12: This is a security step as we have the option to start this service through web from anywhere and to avoid security breach we have to supply a number to the above textbox, this number is available at user/username/, Jenkins/secret folder in a password file, open that file and copy the number and paste it in the above textbox as shown below:

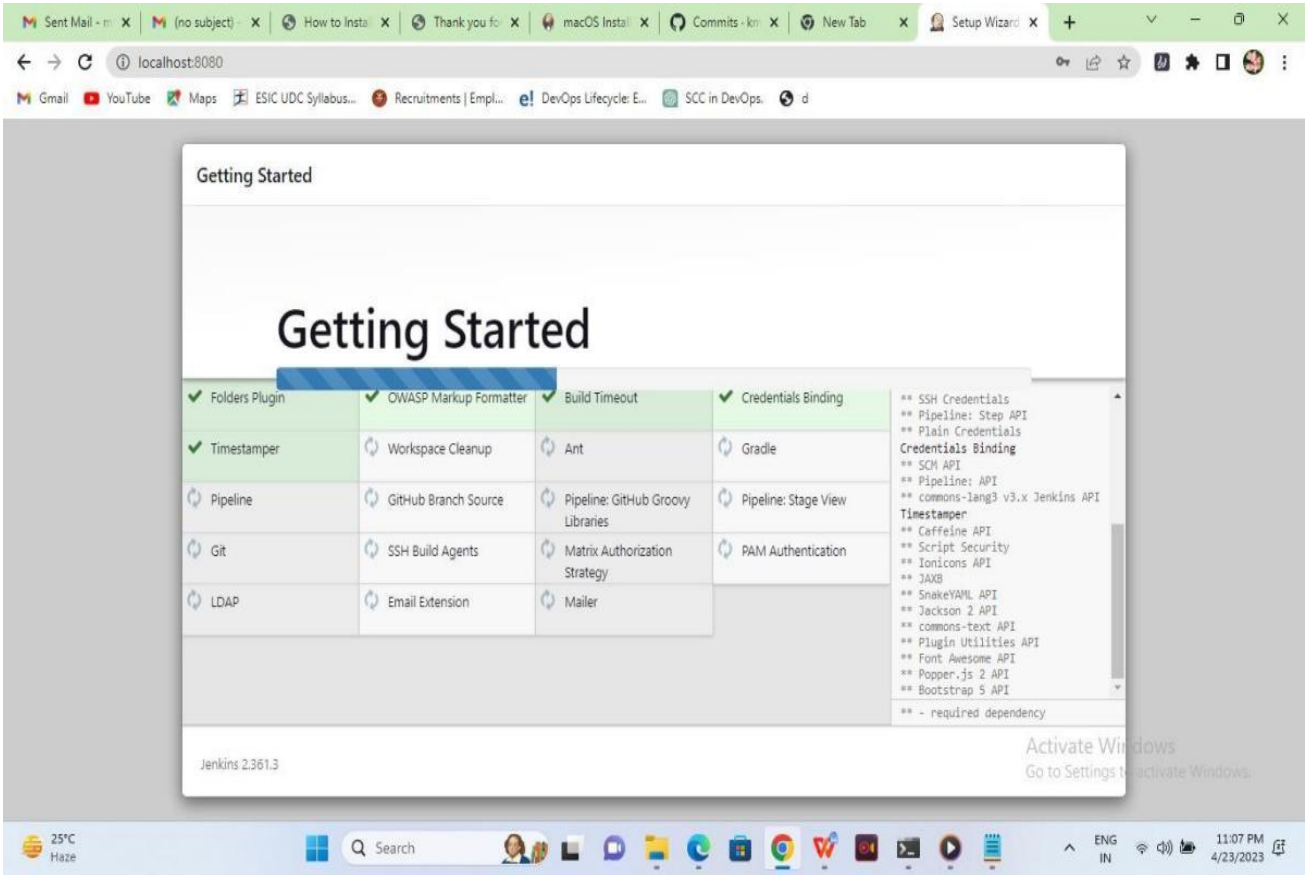


window and type localhost:8080 and we will get the following screen:

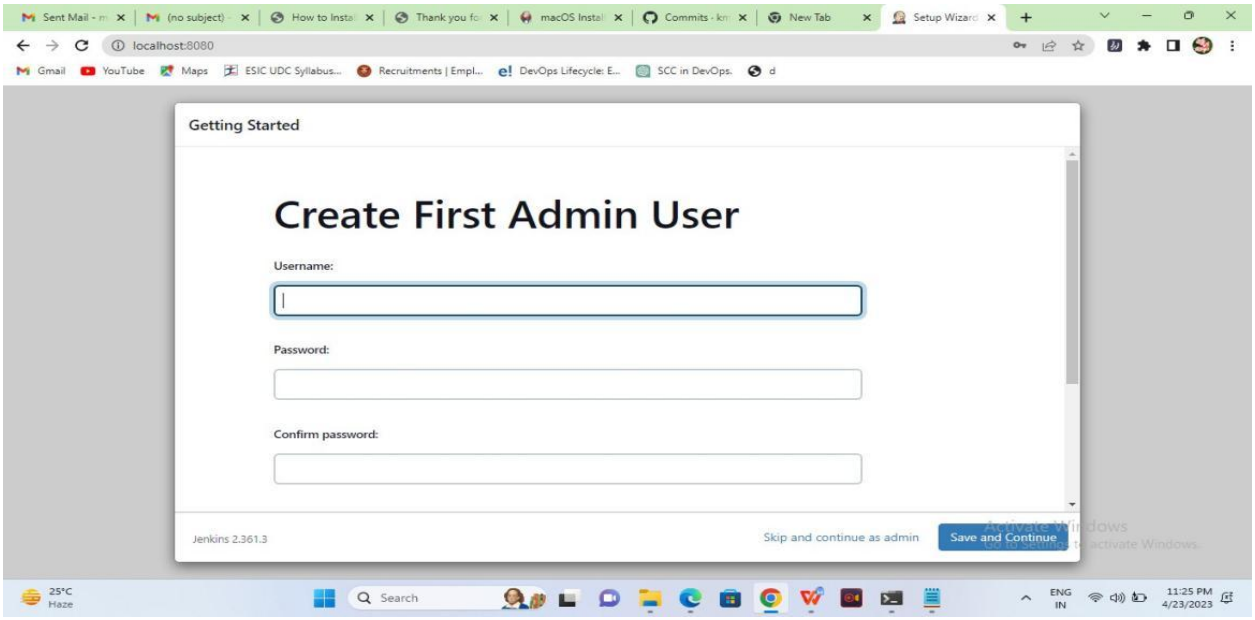
Step 13: After admin configuration, we have to configure the necessary plugins for which we will get the following screen and we can choose either of the two options



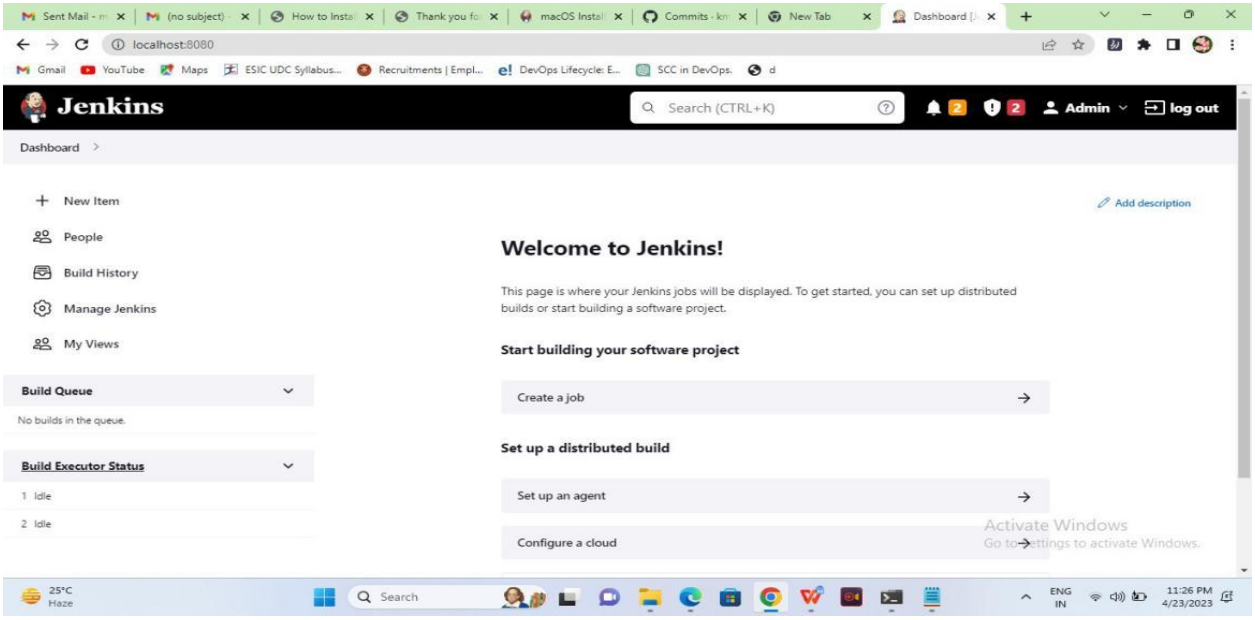
Step 14: After choosing any one of the above options the plugins will start installing as shown:



Step 15: After Installing plugins configure admin details. we have to create an admin account as shown



Step 16: After plugins installation is over (we can add or remove plugins any time) we will get the following screens and we are
ready to work on Jenkins in Windows.



VIVAQUESTIONS

Define Jenkins

EXPERIMENTNO:5. Demonstrate continuous integration and development using Jenkins.

Aim: Demonstrate continuous integration and development using Jenkins.

DESCRIPTION

Continuous Integration (CI) and Continuous Development (CD) are important practices in software development that can be achieved using Jenkins. Here's an example of how you can demonstrate CI/CD using Jenkins: Create a simple Java application:

- Create a simple Java application that you want to integrate with Jenkins.
- The application should have some basic functionality, such as printing "Hello World" or performing simple calculations.

Commit the code to a Git repository:

- Create a Git repository for the application and commit the code to the repository.
- Make sure that the Git repository is accessible from the Jenkins server. Create a

Jenkins job:

- Login to the Jenkins web interface and create a new-job.
- Configure the job to build the Java application from the Git repository.
- Specify the build triggers, such as building after every commit to the repository.

Build the application:

- Trigger a build of the application using the Jenkins job.
- The build should compile the code, run any tests, and produce an executable jar file.

Monitor the build:

- Monitor the build progress in the Jenkins web interface.
- The build should show the build log, test results, and the status of the build.

Deploy the application:

- If the build is successful, configure the Jenkins job to deploy the application to a production environment.
- The deployment could be as simple as copying the jar file to a production server or using a more sophisticated deployment process,

  Such as using a containerization technology like Docker.

Repeat the process:

- Repeat the process for subsequent changes to the application.
- Jenkins should automatically build and deploy the changes to the production environment.

This is a basic example of how you can use Jenkins to demonstrate CI/CD in software development. In a real-world scenario, you would likely have more complex requirements, such as multiple environments, different type softest, and a more sophisticated deployment process. However, this example should give you a good starting point for using Jenkins for CI/CD in

Your software development projects.

VIVAQUESTIONS

Define CD&CI

EXPERIMENT NO.: 6. Explore Docker commands for content management.

AIM: Explore Docker commands for content management.

DESCRIPTION

Docker is a containerization technology that is widely used for managing application containers. Here are some commonly used Docker commands for content management:

- Docker run: Run a command in a new container.

For example: **$dockerrun--namemycontainer-itubuntu:16.04/bin/bash**

This command runs a new container based on the Ubuntu16.04image and starts a shell session in the container.

- Docker start: Start one or more stopped containers.

For example:**$docker start mycontainer**

This command starts the container named "my container".

- Docker stop: Stop one or more running containers.

For example**:$ docker stop mycontainer**

This command stops the container named "mycontainer".

- Docker rm: Remove one or more containers.

For example: **$docker rm mycontainer**

This command removes the container named "mycontainer".

- Docker ps: List containers.

For example**:$ docker ps**

This command lists all running containers.

- Docker images: List images.

For example: **$ docker images**

This command lists all images stored locally on the host.

- Docker pull: Pull an image or a repository from a registry.

For example**:$ docker pull ubuntu:16.04**

This command pulls the Ubuntu16.04 image from the Docker Hub registry.

- Docker push: Push an image or a repository to a registry.

For example:**$ docker push myimage**

This command pushes the image named "myimage" to the Docker Hub registry.

These are some of the basic Docker commands for managing containers and images. There are many other Docker commands and options that you can use for more advanced use cases, such as managing networks, volumes, and configuration. However, these commands should give you a good starting point for using Docker for content management.

VIVAQUESTIONS

Give briefly about Docker commands

EXPERIMENT NO.: 7. Develop a simple containerized application using Docker

AIM: Develop a simple containerized application using Docker

DESCRIPTION: Here's an example of how you can develop a simple containerized application using Docker:

Choose an application:

- Choose a simple application that you want to containerize. For example, a Python script that prints "HelloWorld".

Write a Docker file: `Create a file named "Docker file" in the same directory as the application.`

In the Docker file, specify the base image, copy the application In to the container, and specify the command to run the application. Here's an example Docker file for a Python script:

#Use the official Python image as the base image FRO Mpython:3.9

#Copy the Python script into the container COPY hello.p y/app/

Set the working directory to /app/WORKDIR/app/

#Run the Python script when the container starts CMD ["python","hello.py"]

- Build the Docker image: Run the following command to build the Docker image: **$docker build -t myimage.**
  This command builds a new Docker image using the Docker file and tags the image with the name "myimage".

- Run the Docker container:

Run the following command to start a new container based on the image:
$docker run --name mycontainer myimage This command starts a new container named "mycontainer" based on the "myimage" image and runs the Python script inside the container.

- Verify the output:

Run the following command to verify the output of the container:
$docker logs mycontainer this command displays the logs of the container and should how the "HelloWorld" output.

This is a simple example of how you can use Docker to containerize an application. In a real-world scenario, you would likely have more complex requirements, such as running multiple containers, managing network connections, and persisting data. However, this example should give you a good starting point for using Docker to containerize your applications.

VIVAQUESTIONS

Name the applications using Docker

EXPERIMENT NO:8. Integrate Kubernetes and Docker

AIM: Integrate Kubernetes and Docker

DESCRIPTION: Kubernetes and Docker are both popular technologies for managing containers, but they are used for different purposes. Kubernetes is an orchestration platform that provides a higher-level abstractions for managing containers, while Docker is a containerization technology that provides a lower-level runtime for containers.

To integrate Kubernetes and Docker, you need to use Docker to build and package your application as a container image, and then use Kubernetes to manage and orchestrate the containers.

Here's a high-level over view of the steps to integrate Kubernetes and Docker:

Build a Docker image:

Use Docker to build a Docker image of your application. You can use a Docker file to specify the base image, copy the application into the container and specify the command to run the application.

Push the Docker image to a registry:

Push the Docker image to a container registry, such as Docker Hub or Google Container Registry, so that it can be easily accessed by Kubernetes. Deploy the Docker image to a

Kubernetes cluster:

Use Kubernetes to deploy the Docker image to a cluster. This involves creating a deployment that specifies the number of replicas and the image to be used, and creating a service that exposes the deployment to the network.

Monitor and manage the containers:

Use Kubernetes to monitor and manage the containers. This includes scaling the number of replicas, updating the image, and rolling out updates to the containers.

Continuously integrate and deploy changes:

Use a continuous integration and deployment (CI/CD) pipeline to automatically build, push, and deploy changes to the Docker image and the Kubernetes cluster. This makes it easier to make updates to the application and ensures that the latest version is always running in the cluster. By integrating Kubernetes and Docker, you can leverage the strengths of both technologies to manage containers in a scalable, reliable and efficient manner.

Viva questions:

Define integrate Kubernetes?

What is Docker?

EXPERIMENT NO.:9. Automate the process of running containerized application developed in Experiment 07 using Kubernetes

AIM: Automate the process of running containerized application developed in Experiment 07 using Kubernetes

DESCRIPTION: To automate the process of running the containerized application developed in Experiment no 7 using Kubernetes, you can follow these steps:

- **Create a Kubernetes cluster:** Create a Kubernetes cluster using a cloud provider such as Google Cloud or Amazon

Web Services, or using a local installation of Minikube.

- **Push the Docker image to a registry:** Push the Docker image of your application to a container registry, such as Docker Hub or Google Container Registry.

- **Create a deployment:** Create a deployment in Kubernetes that specifies the number of replicas and the Docker image to use. Here's an example of a deployment **YAML file:**

Api Version:apps/v1

kind: Deployment metadata: name: myappspec: replicas: 3selector:matchLabels:servicespec:selector:app:

my app ports:

app:myapp template:

metadata:

labels:

app:myapp spec:

containers:

- name:

Myapp image: my image ports: -container Port: 80

Create a service: Create a service in Kubernetes

that exposes the deployment to the network.

Here's an example of a service YAMLfile:

apiVersion:v1 kind: Service metadata:

name:myapp- name: httpport:80

target Port: 80type:Cluster IP

-Apply the deployment and service to the cluster: Apply the deployment and service to the cluster using the kubectl command-line tool.

For example:

$kubectl apply –f deployment.yaml

$kubectl apply –f service.yaml

✓ **Verify the deployment:** Verify the deployment by checking the status of the pods and the service. For example:

$ kubectl getpods

$kubectl getservices

This is a basic example of how to automate the process of running a containerized application using Kubernetes. In a real-world scenario, you would likely have more complex requirements, such as managing persistent data, scaling, and rolling updates, but this example should give you a good starting point for using Kubernetes to manage your containers.

VIVAQUESTIONS

Define Kubernetes

EXPERIMENT NO.:10. Install and Explore Selenium for automated testing
AIM: Install and Explore Selenium for automated testing

DESCRIPTION:

To install and explore Selenium for automated testing, you can follow these steps:

Install Java Development Kit (JDK):

- ✓ Selenium is written in Java, so you'll need to install JDK in order to run it. You can download and install JDK from the official Oracle website.
- ✓ Install the Selenium WebDriver:
- ✓ You can download the latest version of the Selenium Web Driver from the Selenium website. You'll also need to download the appropriated river for your web browser of choice (e.g. Chrome Driver for Google Chrome).Install an Integrated Development Environment (IDE):
- ✓ To write and run Selenium tests, you'll need an IDE. Some popular choices include Eclipse, IntelliJ IDEA, and Visual Studio Code.

Write a simple test: Once you have your IDE setup, you can write a simple test using the Selenium WebDriver. Here's an example in Java:

```
Import org.open qa.selenium.WebDriver;
Import org.open qa.selenium.chrome.ChromeDriver;

Public class Main {
  Public static void main (String [] args)
   {System.setProperty("webdriver.chrome.driver","path/to/chromedriver");
   WebDriver driver = new ChromeDriver();
    driver.get("https://www.google.com");
   System.out.println (driver.getTitle());
   driver.quit();
  }
}
```

Run the test: Run the test using your IDE or from the command line using the following command:

```
$javac  Main.java
$java   Main
```

This is a basic example of how to get started with Selenium for automated testing. In a real-world scenario, you would likely write more complex tests and organize your code into test suites and test cases, but this example should give you a good starting point for exploring Selenium.

EXPERIMENT NO.:11. Write a simple program in JavaScript and perform testing using Selenium

AIM: Write a simple program in Java Script and perform testing using Selenium

PROGRAM:

Simple JavaScript Program (Calculator)

HTML + JavaScript (calculator.html)

html
Copy

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Simple Calculator</title>
    <style>
        body { font-family: Arial, sans-serif; text-align: center; margin-top: 50px; }
        input, button { padding: 10px; margin: 5px; font-size: 16px; }
        #result { font-weight: bold; margin-top: 20px; }
    </style>
</head>
<body>
    <h1>Simple Calculator</h1>
    <input type="number" id="num1" placeholder="Enter first number">
    <input type="number" id="num2" placeholder="Enter second number">
    <button onclick="add()">Add</button>
    <button onclick="subtract()">Subtract</button>
    <div id="result"></div>

    <script>
        function add() {
            const num1 = parseFloat(document.getElementById("num1").value);
            const num2 = parseFloat(document.getElementById("num2").value);
            document.getElementById("result").textContent = `Result: ${num1 + num2}`;
        }

        function subtract() {
            const num1 = parseFloat(document.getElementById("num1").value);
            const num2 = parseFloat(document.getElementById("num2").value);
            document.getElementById("result").textContent = `Result: ${num1 - num2}`;
        }
    </script>
</body>
</html>
```

Run HTML

Features:

- Takes two numbers as input.
- Performs **addition** and **subtraction**.
- Displays the result.

Selenium Test Script (Node.js)

Prerequisites

1. Install **Node.js** (Download here)
2. Install **Selenium WebDriver** and a browser driver (e.g., ChromeDriver):

```
npm install selenium-webdriver chromedriver
```

Test Script (calculatorTest.js)

```
javascript
const { Builder, By, until } = require('selenium-webdriver');
const chrome = require('selenium-webdriver/chrome');

(async function testCalculator() {
    // Configure Chrome options (headless mode for faster testing)
    let options = new chrome.Options();
    // options.headless(); // Uncomment to run in headless mode

    // Initialize WebDriver
    let driver = await new Builder()
        .forBrowser('chrome')
        .setChromeOptions(options)
        .build();

    try {
        // Open the calculator page
        await driver.get('file:///path/to/calculator.html'); // Replace with actual path

        // Test addition
        await driver.findElement(By.id('num1')).sendKeys('10');
        await driver.findElement(By.id('num2')).sendKeys('5');
        await driver.findElement(By.xpath('//button[text()="Add"]')).click();

        let result = await driver.findElement(By.id('result')).getText();
        console.log(result === 'Result: 15' ? '✅ Addition test passed' : '❌ Addition test failed');

        // Test subtraction
        await driver.findElement(By.id('num1')).clear();
        await driver.findElement(By.id('num2')).clear();
        await driver.findElement(By.id('num1')).sendKeys('20');
        await driver.findElement(By.id('num2')).sendKeys('8');
        await driver.findElement(By.xpath('//button[text()="Subtract"]')).click();

        result = await driver.findElement(By.id('result')).getText();
        console.log(result === 'Result: 12' ? '✅ Subtraction test passed' : '❌ Subtraction test failed');
```

```
    } finally {
        // Quit the driver
        await driver.quit();
    }
})();
```

Running the Test

1. **Start a local web server** (if not using file://):

   ```
   npx http-server
   ```

2. **Run the test**:
   ```
   node calculatorTest.js
   ```

Expected Output

```
✅ Addition test passed
✅ Subtraction test passed
```

EXPERIMENT NO.:12. Develop test cases for the above containerized application using selenium

AIM: Develop test cases for the above containerized application using selenium

PROGRAM:

Here is an example of how you could write test cases for the containerized application using Selenium

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import static org.junit.Assert.*;

public class ContainerizedAppTest {
    private WebDriver driver;

    @Before
    public void setUp() {
        // Set path to chromedriver (update with your actual path)
        System.setProperty("webdriver.chrome.driver", "/path/to/chromedriver");
        driver = new ChromeDriver();
    }

    @Test
    public void testHomePageLoads() {
        driver.get("http://localhost:8080");
        String title = driver.getTitle();
        assertEquals("MyContainerizedApplication", title);
    }

    @Test
    public void testSubmitForm() {
        driver.get("http://localhost:8080");

        // Fill and submit form
        driver.findElement(By.name("name")).sendKeys("John Doe");
        driver.findElement(By.name("email")).sendKeys("john.doe@example.com");
        driver.findElement(By.name("submit")).click();

        // Verify result
        String result = driver.findElement(By.id("result")).getText();
        assertEquals("Form submitted successfully!", result);
    }

    @After
    public void tearDown() {
        if (driver != null) {
            driver.quit();
        }
    }
}
```

You can run the test cases using the following command:
$javac Main.java
$java Main
The output of the test cases should be:
Tests run: 2, Failures: 0, Errors: 0
OK(2tests)
This output indicates that both test cases passed, and the containerized application is functioning as expected.

VIVAQUESTIONS

1. Define selenium
2. Name the test cases used in selenium