MCAL23 DevOps Lab INDEX

Sr.No	Title	CO	Date	Sign
1	Demonstrate basic Git commands	CO2		
2	Create and fork repositories in Git Hub. Apply branch, merge and rebase concepts.	CO2		
3	Demonstrate Git for Collabration	CO2		
4	Demonstrate Collabrating and cloning using Git	CO2		
5	Using GitLab Web IDE	CO2		
6	Demonstrate CI/CD Workflow in GitLab using .py, .bash, .java file	CO2		
7	Demonstrate setting Jenkins CI/CD pipeline.	CO3		
8	Demonstrate Setting up of a CI/CD pipeline to build and deploy a web application to a local HTTP server	CO3		
9	Demonstrate basic Docker commands	CO3		
10	Develop a simple containerized application using Docker	CO3		
11	Demonstrate add-on ansible commands	CO4		
12	Demonstrate Ansible Playbooks	CO4		

PRACTICAL- 1

Aim: Basic Git commands

1. 

Check git version git –version

–

2. **Create folder and initiliaze.**

```
ubuntu@ubuntu:~$ git --version
git version 2.25.1
ubuntu@ubuntu:~$ mkdir newuser
ubuntu@ubuntu:~$ cd newuser/
ubuntu@ubuntu:~/newuser$ git init
Initialized empty Git repository in /home/ubuntu/newuser/.git/
ubuntu@ubuntu:~/newuser$
```

3. Configure Git

git config --global user.name "usernewncrd"

git config --global user.email “symca669@gmail.com”

```
ubuntu@ubuntu:~/newuser$ git config --global user.name "usernewncrd"
ubuntu@ubuntu:~/newuser$ git config --global user.email "symca669@gmail.com"
ubuntu@ubuntu:~/newuser$
```

4. Create a new project folder mkdir git-demo

cd git-demo

```
ubuntu@ubuntu:~/newuser$ mkdir git-demo
ubuntu@ubuntu:~/newuser$ cd git-demo/
ubuntu@ubuntu:~/newuser/git-demo$
```

5. git init

```
ubuntu@ubuntu:~/newuser/git-demo$ git init
Initialized empty Git repository in /home/ubuntu/newuser/git-demo/.git/
ubuntu@ubuntu:~/newuser/git-demo$
```

6. Create and track a file: echo "Hello User" > file.txt git add file.txt

git commit -m "Initial commit"

```
ubuntu@ubuntu:~/newuser/git-demo$ echo "Hello User"> file.txt
ubuntu@ubuntu:~/newuser/git-demo$ git add file.txt
ubuntu@ubuntu:~/newuser/git-demo$ git commit -m "Initial Commit"
[master (root-commit) 5da5867] Initial Commit
 1 file changed, 1 insertion(+)
 create mode 100644 file.txt
ubuntu@ubuntu:~/newuser/git-demo$
```

7. Check status and log: git status

git log

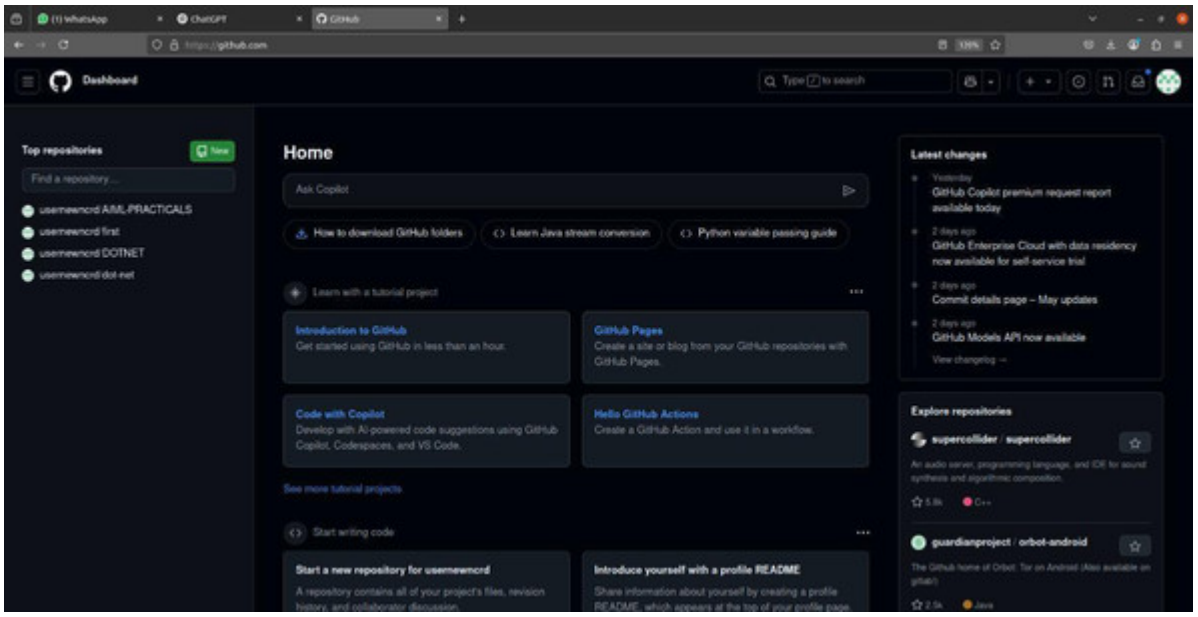
```
ubuntu@ubuntu:~/newuser/git-demo$ git status
On branch master
nothing to commit, working tree clean
ubuntu@ubuntu:~/newuser/git-demo$ git log
commit 5da586754b11433e7ab5ed5d1eafad9ad22d9289 (HEAD -> master)
Author: usernewncrd <symca669@gmail.com>
Date:   Sun May 18 13:52:53 2025 +0530

    Initial Commit
ubuntu@ubuntu:~/newuser/git-demo$
```

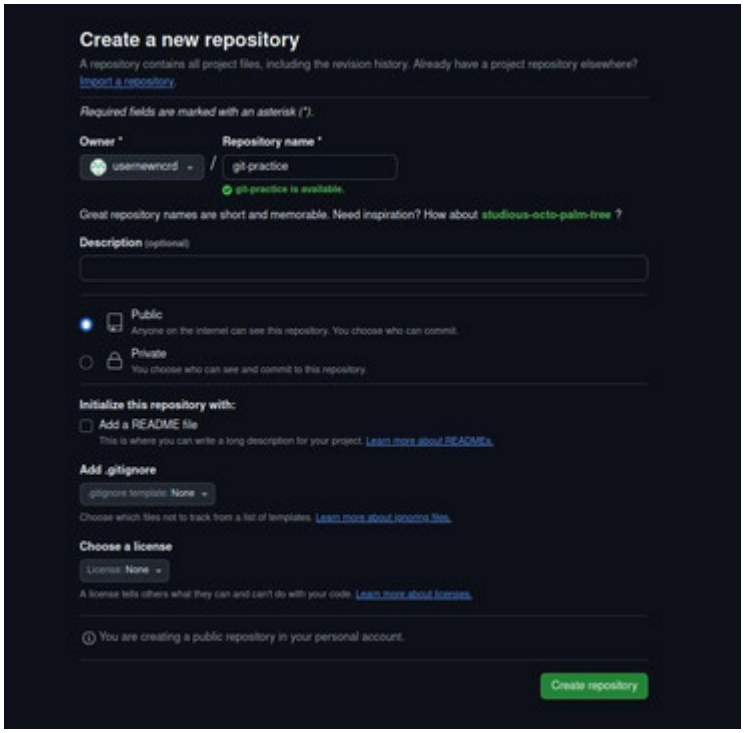
PRACTICAL- 2

Aim: Create and fork repositories in GitHub. Apply branch, merge, rebase concepts.

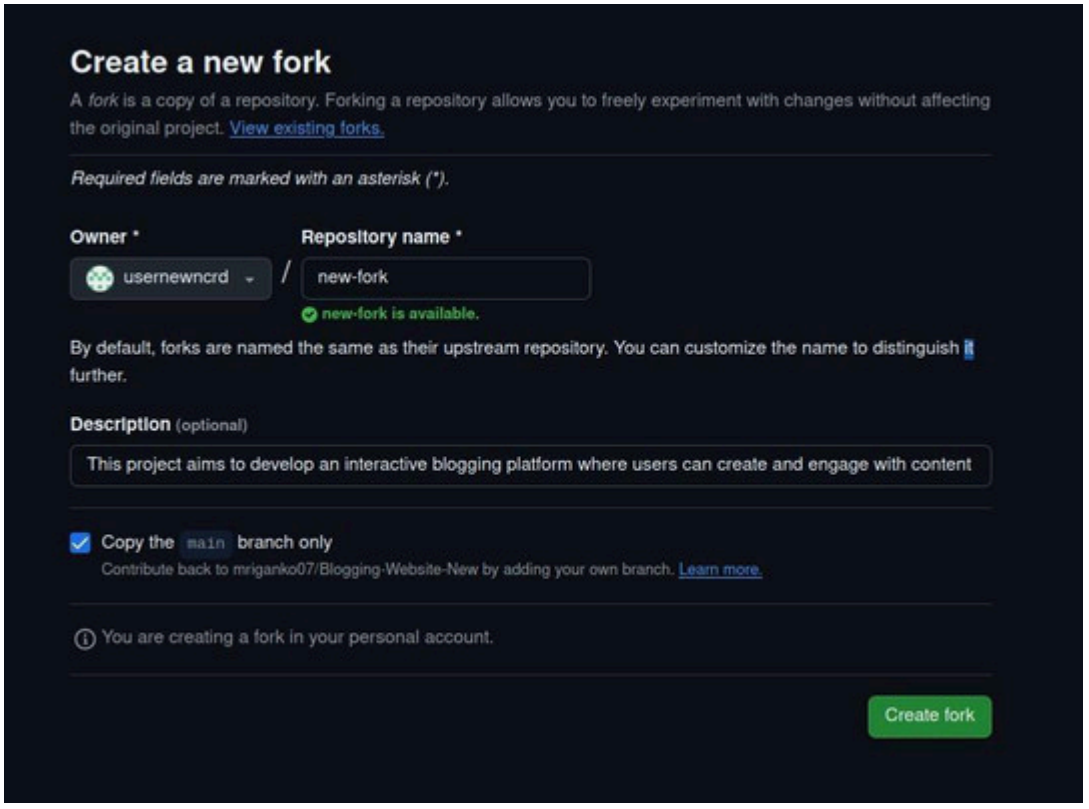
1. **Create a GitHub account and log in.**



2. Create a repository on GitHub (e.g., git-practice).



3. Fork any public repository or your own from another account



4. Clone the forked repo:

git clone https://github.com/usernewncrd/git-practice.git cd git-practice

```
ubuntu@ubuntu:~/newuser/git-demo$ git clone https://github.com/usernewncrd/new-fork
Cloning into 'new-fork'...
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 7 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (7/7), 28.85 KiB | 1.07 MiB/s, done.
ubuntu@ubuntu:~/newuser/git-demo$ cd new-fork/
ubuntu@ubuntu:~/newuser/git-demo/new-fork$
```

5. Create a branch:

git checkout -b feature

```
ubuntu@ubuntu:~/newuser/git-demo/new-fork$ git checkout -b feature
Switched to a new branch 'feature'
```

6. Make changes, then commit:

echo "Feature added" >> newfile.txt git add .

git commit -m "Added new feature"

```
ubuntu@ubuntu:~/newuser/git-demo/new-fork$ echo "Feature Added" >> newfile.txt
ubuntu@ubuntu:~/newuser/git-demo/new-fork$ git add .
ubuntu@ubuntu:~/newuser/git-demo/new-fork$ git commit -m "Added new feature"
[feature ec92d67] Added new feature
 1 file changed, 1 insertion(+)
 create mode 100644 newfile.txt
```

7. Merge branch into main: git checkout master

git merge feature

```
ubuntu@ubuntu:~/newuser/git-demo/new-fork$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
ubuntu@ubuntu:~/newuser/git-demo/new-fork$ git merge feature
Updating d0bf9b1..ec92d67
Fast-forward
 newfile.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 newfile.txt
ubuntu@ubuntu:~/newuser/git-demo/new-fork$
```

8. Rebase branch (alternative to merge): git checkout feature

git rebase master

```
ubuntu@ubuntu:~/newuser/git-demo/new-fork$ git checkout feature
Switched to branch 'feature'
ubuntu@ubuntu:~/newuser/git-demo/new-fork$ git rebase main
Current branch feature is up to date.
ubuntu@ubuntu:~/newuser/git-demo/new-fork$
```

9. Push to GitHub:

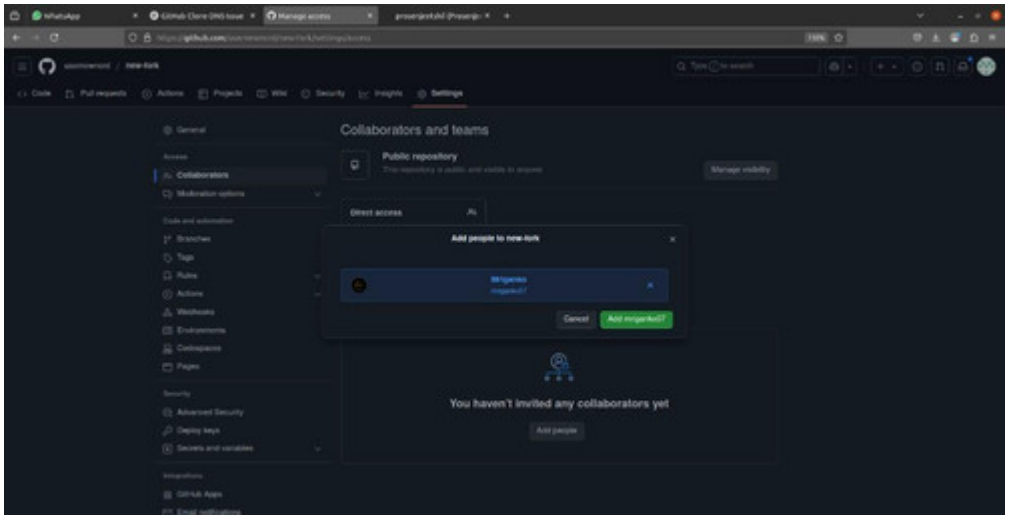
git push origin feature

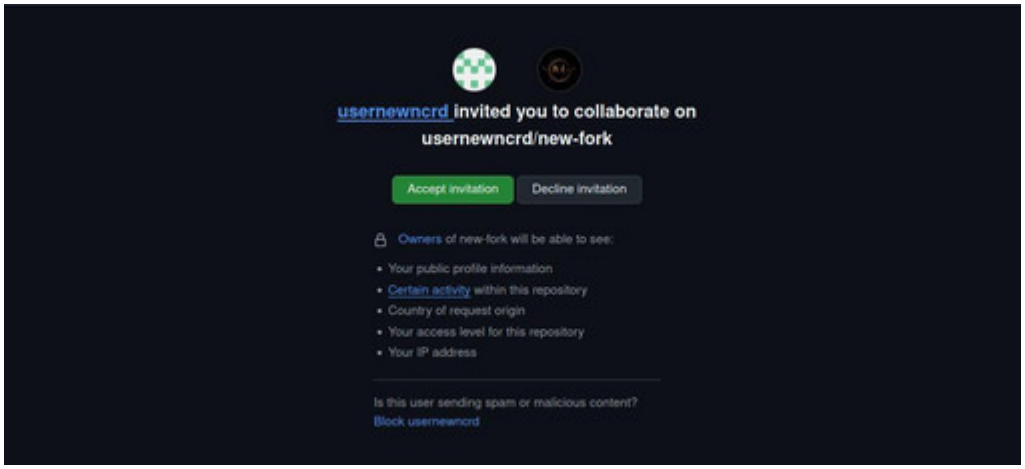
```
ubuntu@ubuntu:~/newuser/git-demo/new-fork$ git push origin feature
Username for 'https://github.com': usernewncrd
Password for 'https://usernewncrd@github.com':
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 283 bytes | 283.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'feature' on GitHub by visiting:
remote:      https://github.com/usernewncrd/new-fork/pull/new/feature
remote:
To https://github.com/usernewncrd/new-fork
 * [new branch]      feature -> feature
ubuntu@ubuntu:~/newuser/git-demo/new-fork$
```

PRACTICAL-3

Aim: Using Git for Collaboration

1. Using Git for Collaboration





2. Friend clones the repo:

git clone https://github.com/usernewncrd/git-practice.git cd team-repo
git checkout -b bug-fix

```
ubuntu@ubuntu:~/newuser/git-demo$ git clone https://github.com/usernewncrd/git-practice.git
Cloning into 'git-practice'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 1000.37 KiB | 2.44 MiB/s, done.
ubuntu@ubuntu:~/newuser/git-demo$ cd team-repo
bash: cd: team-repo: No such file or directory
ubuntu@ubuntu:~/newuser/git-demo$ git checkout -b bug-fix
Switched to a new branch 'bug-fix'
ubuntu@ubuntu:~/newuser/git-demo$
```

3. Friend makes changes and pushes: echo "Bug fixed" >> bug.txt

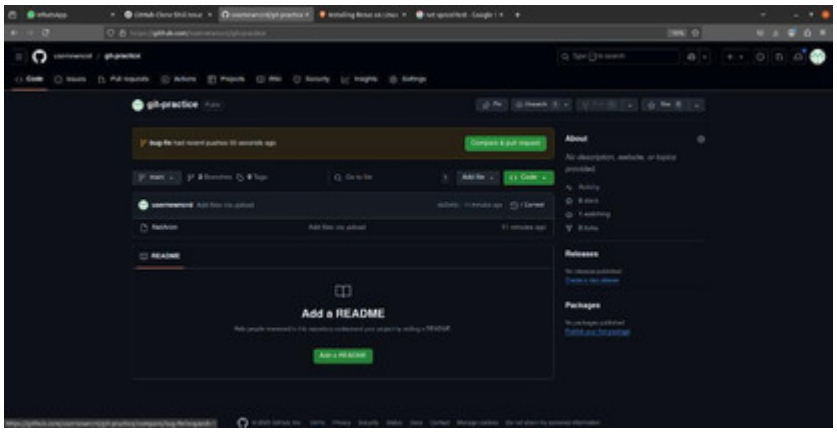
git add .
git commit -m "Fixed a bug"

```
ubuntu@ubuntu:~/newuser/git-demo$ echo "Bug fixed">>bug.txt
ubuntu@ubuntu:~/newuser/git-demo$ git add .
warning: adding embedded git repository: git-practice
hint: You've added another git repository inside your current repository.
hint: Clones of the outer repository will not contain the contents of
hint: the embedded repository and will not know how to obtain it.
hint: If you meant to add a submodule, use:
hint:
hint:   git submodule add <url> git-practice
hint:
hint: If you added this path by mistake, you can remove it from the
hint: index with:
hint:
hint:   git rm --cached git-practice
hint:
hint: See "git help submodule" for more information.
warning: adding embedded git repository: new-fork
ubuntu@ubuntu:~/newuser/git-demo$ git commit -m "Fixed the bug"
[bug-fix a816be3] Fixed the bug
 3 files changed, 3 insertions(+)
 create mode 100644 bug.txt
 create mode 160000 git-practice
 create mode 160000 new-fork
ubuntu@ubuntu:~/newuser/git-demo$
```

4. git push origin bug-fix

```
ubuntu@ubuntu:~/newuser/git-demo$ git push origin bug-fix
Username for 'https://github.com': usernewncrd
Password for 'https://usernewncrd@github.com':
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (6/6), 549 bytes | 549.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0)
remote:
remote: Create a pull request for 'bug-fix' on GitHub by visiting:
remote:      https://github.com/usernewncrd/git-practice/pull/new/bug-fix
remote:
To https://github.com/usernewncrd/git-practice.git
 * [new branch]      bug-fix -> bug-fix
ubuntu@ubuntu:~/newuser/git-demo$
```

5. Pull Request



PRACTICAL-4

Aim: Collaborating and Cloning using GitHub

1. Clone a public repository:

git clone https://github.com/usernewncrd /git-practice.git

```
ubuntu@ubuntu:~/newuser/git-demo$ git clone https://github.com/usernewncrd/git-practice.git
Cloning into 'git-practice'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 1000.37 KiB | 2.44 MiB/s, done.
```

2. Create a branch:

git checkout -b update-readme

```
ubuntu@ubuntu:~/newuser/git-demo$ git checkout -b update-readme
Switched to a new branch 'update-readme'
ubuntu@ubuntu:~/newuser/git-demo$
```

3. Edit and commit changes:

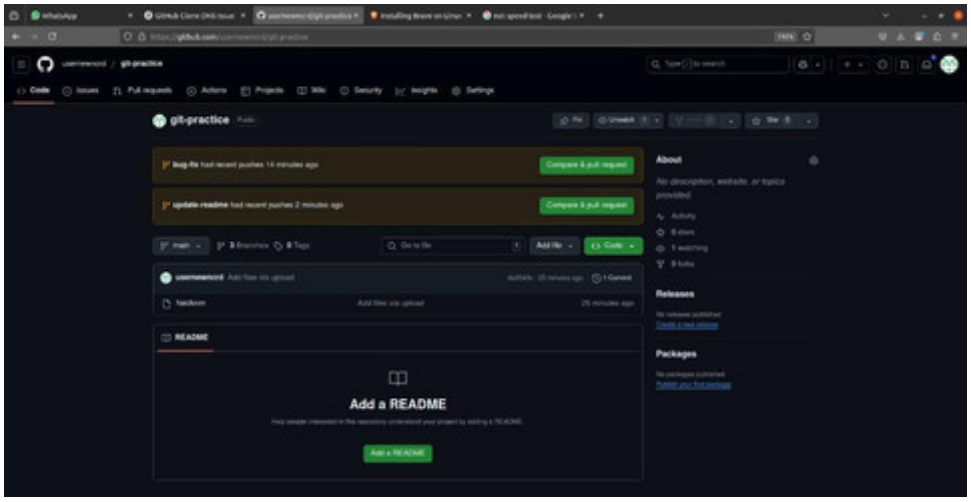
echo "Added a line" >> README.md git add README.md
git commit -m "Updated README"

```
ubuntu@ubuntu:~/newuser/git-demo$ echo "Added a line">>README.md
ubuntu@ubuntu:~/newuser/git-demo$ git add README.md
ubuntu@ubuntu:~/newuser/git-demo$ git commit -m "Updated README"
[update-readme 11aa668] Updated README
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
ubuntu@ubuntu:~/newuser/git-demo$
```

4. Push and open pull request:

```
ubuntu@ubuntu:~/newuser/git-demo$ git push origin update-readme
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 290 bytes | 290.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'update-readme' on GitHub by visiting:
remote:      https://github.com/usernewncrd/git-practice/pull/new/update-readme
remote:
To https://github.com/usernewncrd/git-practice.git
 * [new branch]      update-readme -> update-readme
ubuntu@ubuntu:~/newuser/git-demo$
```

5. git push origin update-readme



PRACTICAL-5

Aim: Using GitLab Web IDE

Using GitLab Web IDE is a convenient way to edit, commit, and manage your code directly in your
browser without needing a local IDE. Here's a quick overview of how to use it effectively:
Opening GitLab Web IDE

1. Go to your project in GitLab.
2. Click the Web IDE button (usually near the top right of the repository page).
    - You can also access it via https://gitlab.com/<namespace>/<project>/-

/ide/project/<branch-name>/edit.
Basic Features

Feature	Description
File Explorer	View and manage your project files on the left panel.
Editor Window	Edit files with syntax highlighting and autocompletion.
Terminal (optional)	For Git commands, builds, or other scripts (if enabled).
Commit Panel	Stage, commit, and push your changes.
Branches	Switch between branches and create new ones.

Workflow Example

1. Edit files: Click on a file in the left panel and make your changes.
2. Stage changes: Go to the "Changes" tab → check the files to stage.
3. Write a commit message: In the "Commit" section.

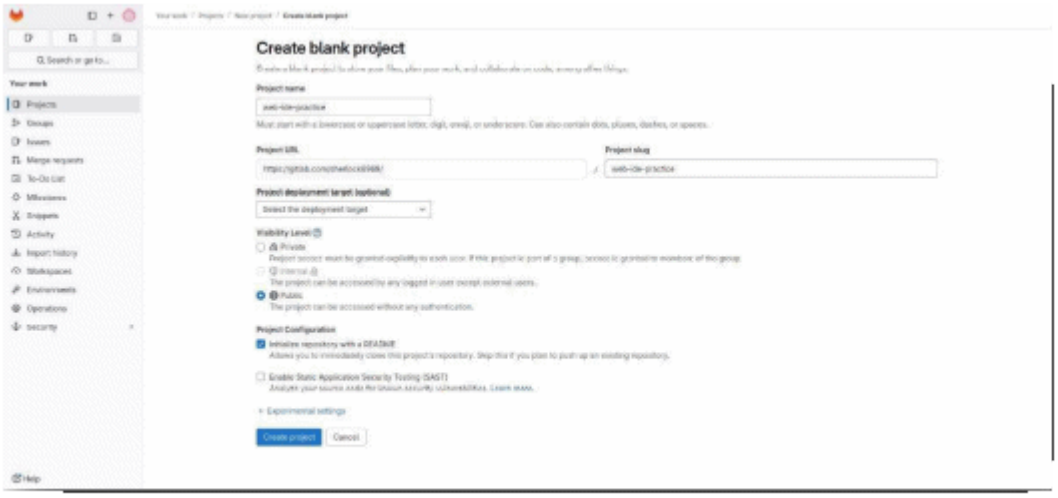
4. Commit & Push:
    - Choose "Commit to current branch" if you're working on a feature branch.
    - Or "Create merge request" if you're done and ready to merge.

Tips

- Auto Save: It autosaves your file edits, but you still need to commit changes.
- Live Preview (for web apps): Available for GitLab Pages-based previews if set up.
- Extensions: Web IDE supports VS Code-like extensions for enhanced functionality.

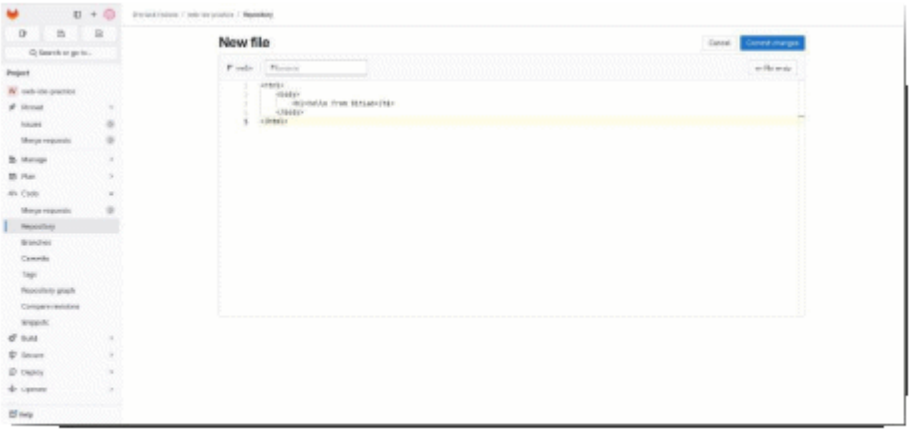
Steps:

1. Sign up at https://gitlab.com
2. Create a project.
3. Click on Web IDE in your repository.

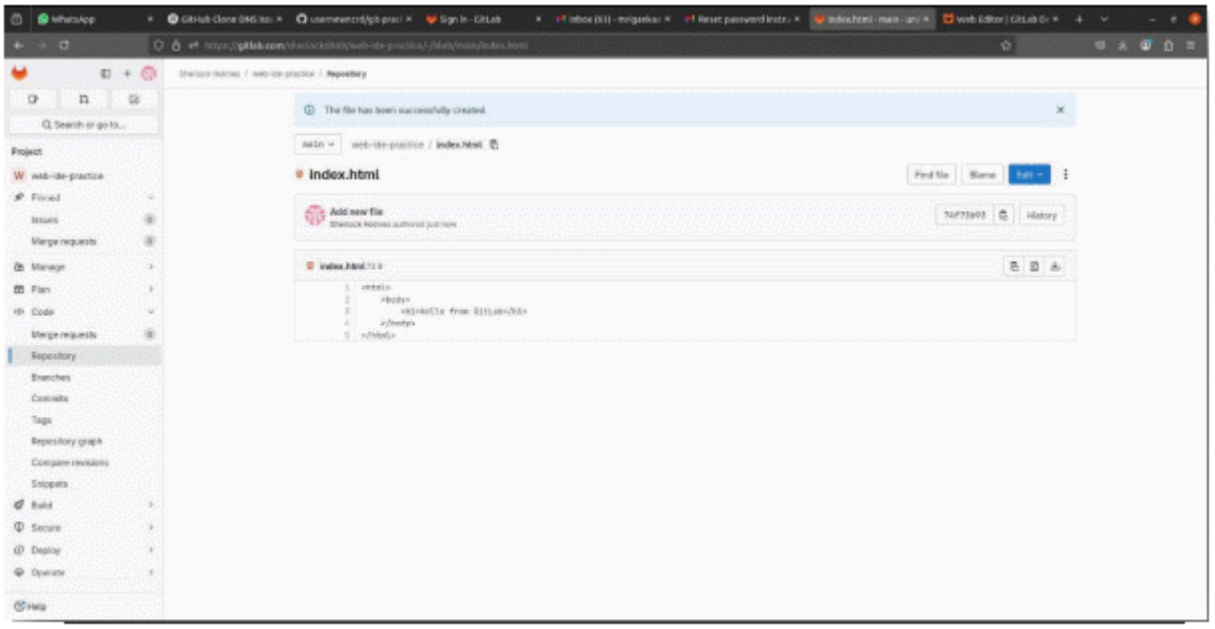


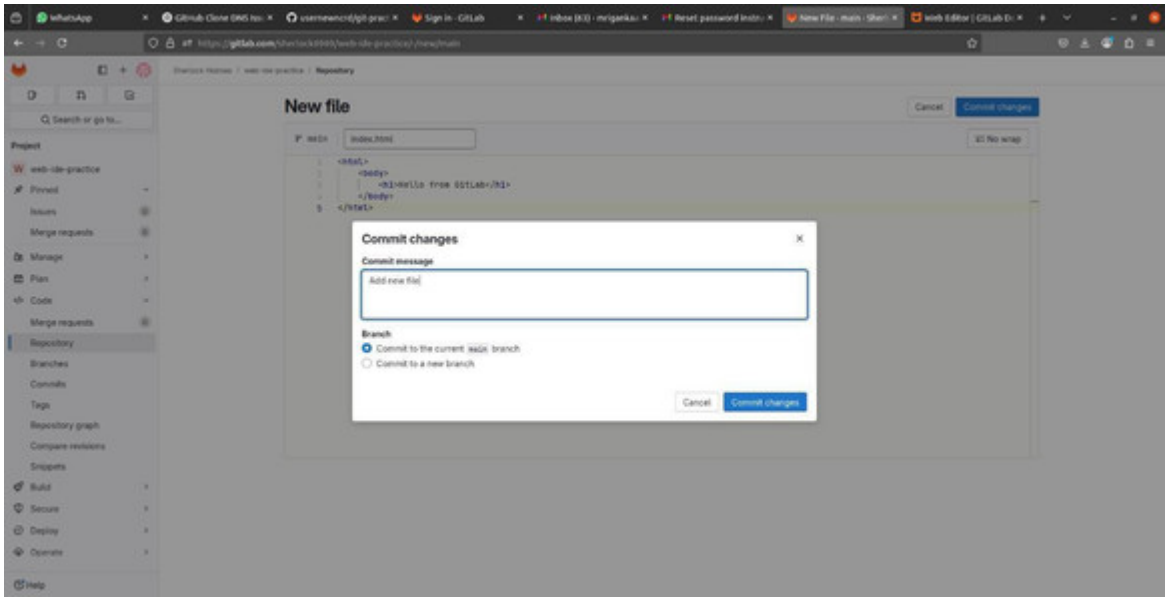
4. Create a file (index.html):

```
<html>
<body>
<h1>Hello from GitLab</h1>
</body>
</html>
```



4. Click Commit and push changes.

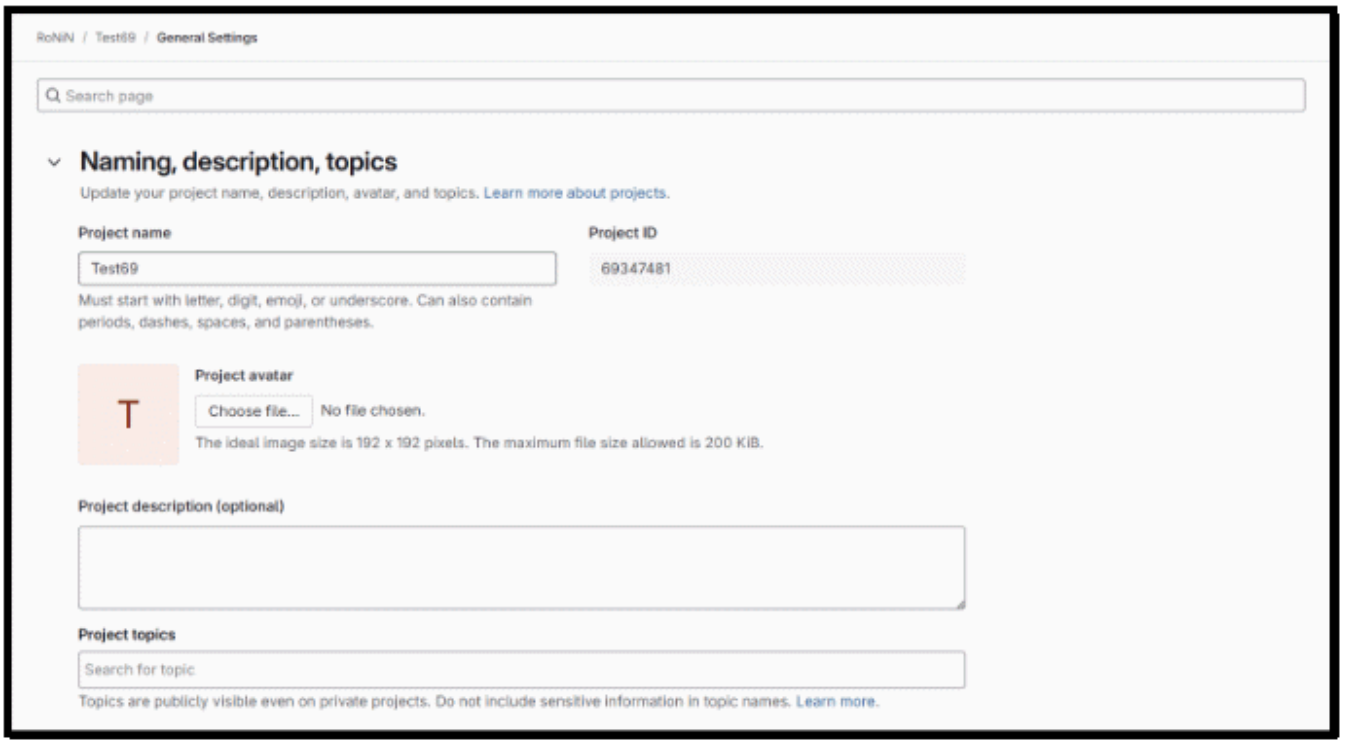




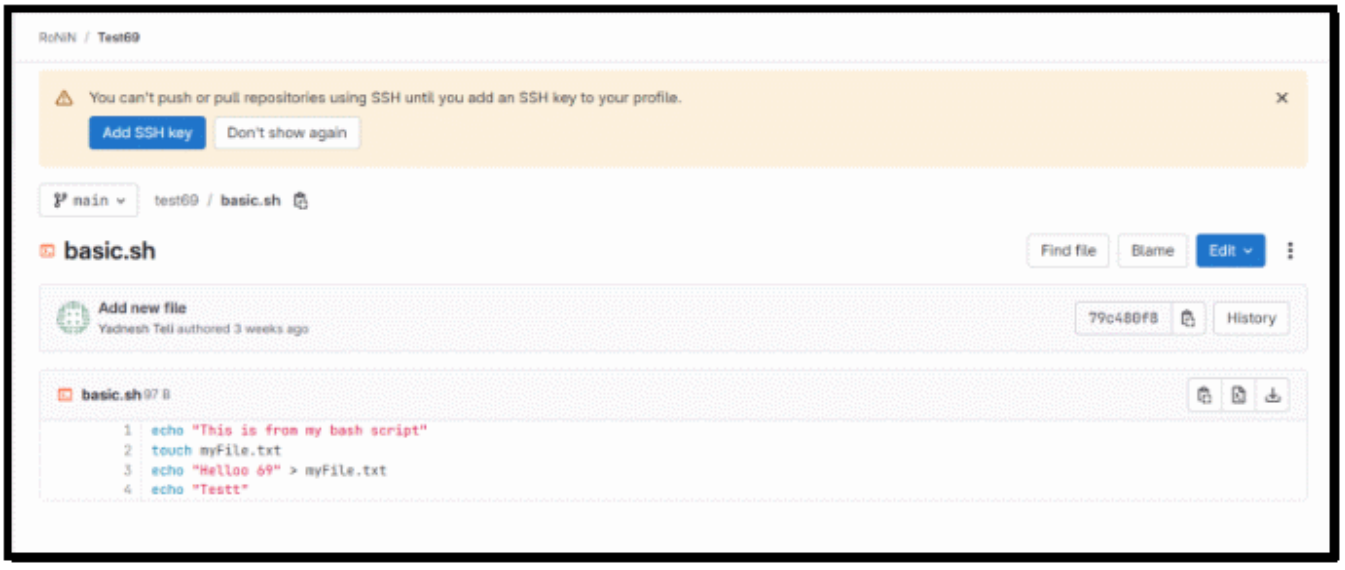
Practical No.:6

Demonstrate CI/CD Workflow in GitLab using .py, .bash, .java file

Bash

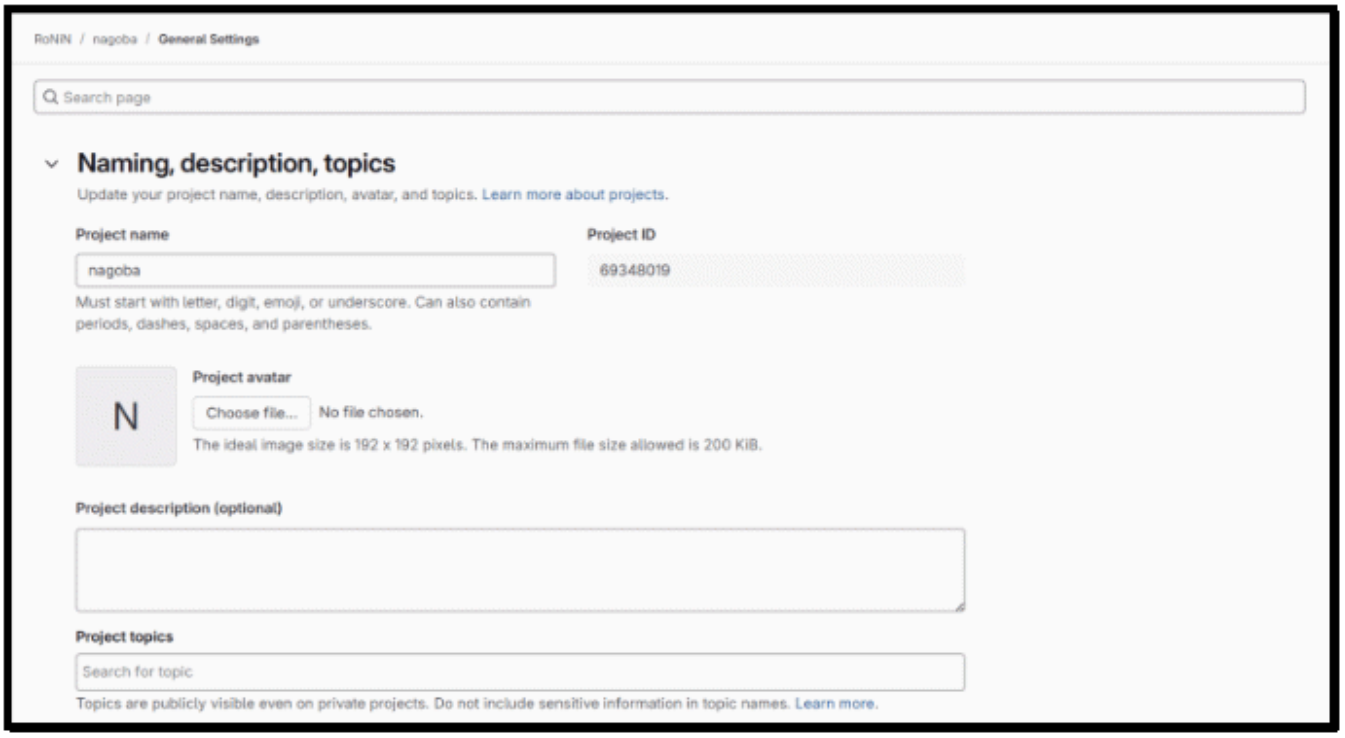


echo "This is from my bash script"
touch myFile.txt
echo "Helloo 69" > myFile.txt
echo "Testt"



stages:
- build
bash_execute:
stage: build
script:
- bash ./basic.sh

PYTHON



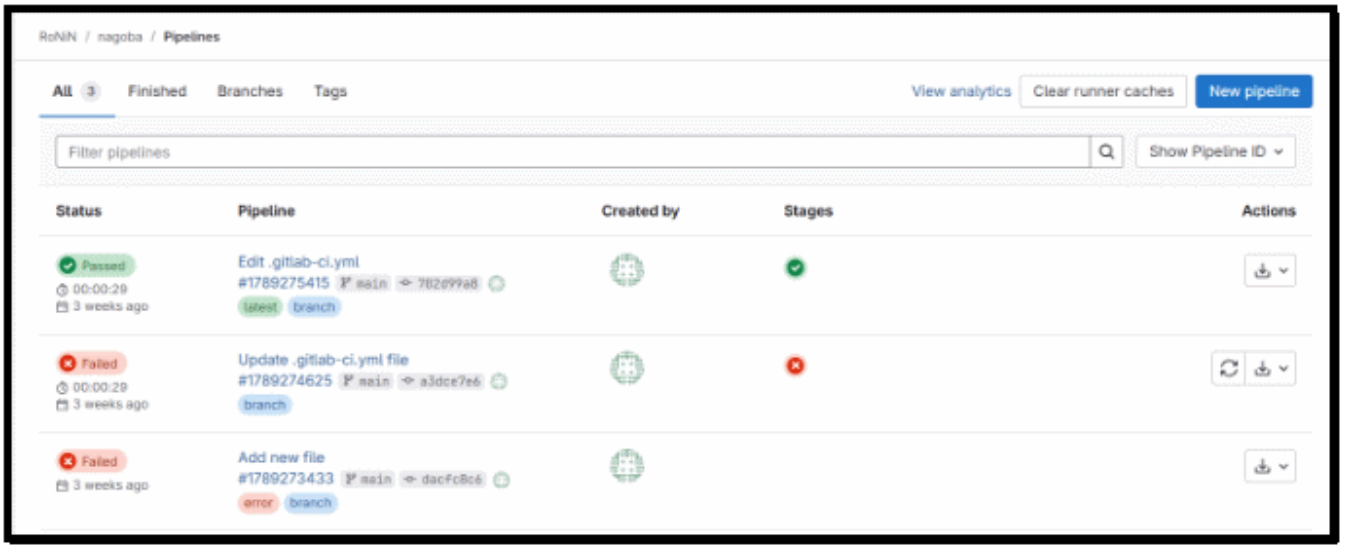
script.py

print("HEllO FROM NAGOBA")

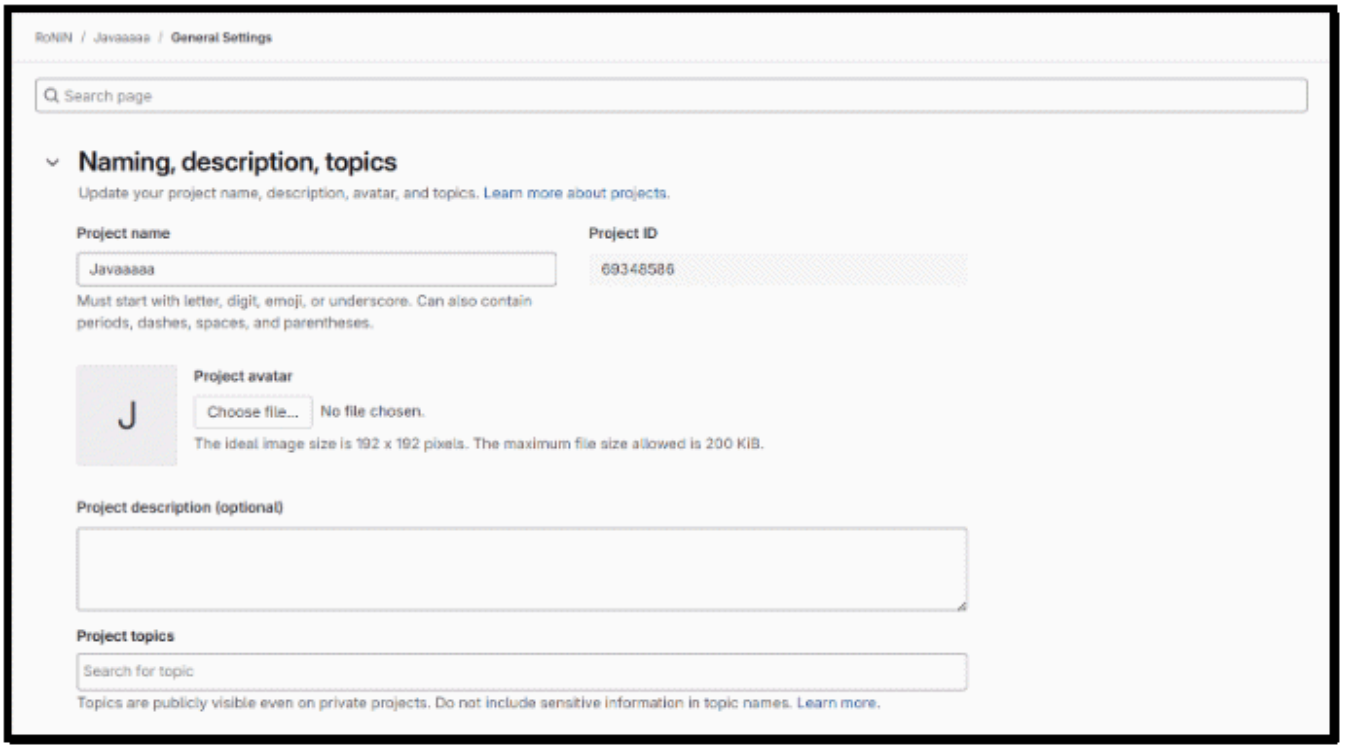
.gitlab-ci.yml

stages:
- test

python_script:
stage: test
image: python:3.10
script:
- python script.py



JAVA



JAvaaa.java

```
class JAvaaa{
public static void main(String a[]){
System.out.println("Hello World!");
System.out.println("Agent 47 arrived in Lahore!");
}
}
```

.gitlab-ci.yml

```
stages:
- build
- test

before_script:
- apt-get update && apt-get install -y openjdk-17-jdk

build:
stage: build
script:
- javac JAvaaa.java
- ls -ls

artifacts:
paths:
- JAvaaa.class
only:
- main

test:
stage: test
when: manual
script:
- ls -l
- java JAvaaa
only:
- main
```

Practical No.:7

Demonstrate setting Jenkins CI/CD pipeline.

1. **Install Jenkins (visit https://www.jenkins.io)**
2. **Run Jenkins: http://localhost:8080**

3. **Create a new Pipeline project: CI-CD-Demo**

4. **Add Pipeline Script > Script:**

```
pipeline {
agent any // Defines where the pipeline runs

stages {
stage('Build') { // Defines a step in the pipeline
steps {
echo 'Building the project...' // Print message to console
}
}
stage('Test') {
steps {
echo 'Running tests...'
}
}
stage('Deploy') {
steps {
echo 'Deploying the application...'
}
}
}
post {
success {
echo 'Pipeline completed successfully!' // Runs if the pipeline is successful
}
failure {
echo 'Pipeline failed!' // Runs if any stage fails
}
}
}
```

5. **Save and click Build Now.**

6. **Check output in Console Output.**

Practical No.:8
Demonstrate Setting up of a CI/CD pipeline to build and deploy a web application to a local HTTP server

Create a new Dynamic web project (Eclipse IDE for enterprise java and web developers)
Index.jsp :

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<html>
<head>
<title>CookiesDemo</title>
</head>
<body>
<h2>CookiesDemo - </h2>
<form action="CookiesDemo.jsp" method="get">
Name - <input type="text" name="usernm">
<input type="submit" value="Submit Query">
</form>
</body>
</html>
```

CookiesDemo.jsp:

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
```

```
<%@ page import="jakarta.servlet.http.Cookie" %>
s<%@ page import="java.io.*" %>
<html>
<head>
<title>Session Management using Cookies</title>
</head>
<body>
<h2>Session Management using Cookies</h2>
<%
String username = request.getParameter("usernm");
Cookie[] cookies = request.getCookies();
int visitCount = 0;
boolean userExists = false;

if (cookies != null) {
for (Cookie cookie : cookies) {
if (cookie.getName().equals("visitCount")) {
visitCount = Integer.parseInt(cookie.getValue());
}
if (cookie.getName().equals("username")) {
userExists = true;
}
}
}
visitCount++;
Cookie visitCookie = new Cookie("visitCount", String.valueOf(visitCount));
visitCookie.setMaxAge(60 * 60 * 24);
response.addCookie(visitCookie);

if (!userExists && username != null) {
Cookie userCookie = new Cookie("username", username);
userCookie.setMaxAge(60 * 60 * 24);
response.addCookie(userCookie);
}
%>
<p>Hello <%= username != null ? username : "Guest" %> You have hit the page <%=
visitCount %> time(s)</p>
<a href=">Hit Again</a>
</body>
```

```
</html>
```

Add Pipeline Script > Script:

```
pipeline {
agent any

stages {
stage('Checkout Code') {
steps {
script {
git branch: 'master', url: 'https://github.com/YadneshTeli/DevopsJenkins'
}
}
}

stage('Verify Files') {
steps {
bat 'dir /S /B'
}
}
stage('Deploy') {
steps {
script {
def srcPath = "CookiesDemo/src/main/webapp"
def destPath = "C:\\Program Files\\Apache Software Foundation\\Tomcat 11.0\\webapps\\Index"

if (fileExists(srcPath)) {
bat "xcopy /E /I \"${srcPath}\" \"${destPath}\""
} else {
error "Source directory ${srcPath} does not exist!"
}
}
}
}
}
}
```

- **Open Manager app from the Tomcat panel by entering username and password –**

- **Click on the link present in the Jenkin's Console Output –**

PRACTICAL-9

Aim: Explore docker commands for content management

1. Check Docker version docker –version

2. Pull a Docker image from Docker Hub docker pull nginx

3. List all Docker images docker images

4. Run a container from an image

docker run -d -p 8080:80 --name mynginx nginx
This will run the Nginx container and map port 80 (inside the container) to port 8080 (on your host).

5. List all running containers docker ps

6. Copy content from host to container

docker cp index.html mynginx:/usr/share/nginx/html/
Replace index.html with your actual file. This copies a file into the running container.

7. Copy content from container to host

docker cp mynginx:/usr/share/nginx/html/index.html .

8. Create and use Docker volume for persistent content docker volume create mydata

docker run -d -p 8081:80 --name nginx_vol -v mydata:/usr/share/nginx/html nginx
Now any data added to the /usr/share/nginx/html inside the container will persist even if the container is removed.

9. List Docker volumes docker volume ls

10. Remove a container docker rm -f mynginx Remove an image docker rmi nginx

PRACTICAL-10

Aim: Develop a simple containerized application using Docker

Develop a Simple Containerized Application using Docker

1. Index.html

2. Dockerflle :-

3. docker build -t my-docker-webapp .

4. docker run -d -p 8080:80 --name webapp-container my-docker-webapp

5. docker ps

6. docker stop webapp-container

7. docker rm webapp-container

8. docker rmi my-docker-webapp

PRACTICAL-11

Aim: Ad-hoc Ansible Commands

Step 1: Update your VM

Step 2: Install Ansible

Step 3: Check version:

1. Ping the remote host

ansible local -i host.ini -m ping

2. Check uptime

ansible local -i host.ini -a "uptime"

3. Install a package

ansible local -i host.ini -m apt -a "name=nginx state=present update_cache=yes" –become

4. Start a service

ansible local -i host.ini -m service -a "name=nginx state=started" –become

PRACTICAL-12

Aim: Using Ansible Playbooks

Install and Start Nginx

install_nginx.yml:
- name: Install and start Nginx on web servers hosts: webservers
become: true tasks:

- name: Install Nginx apt:

name: nginx state: present update_cache: yes

- name: Start Nginx service:

name: nginx state: started enabled: true

Run the Playbook:
ansible-playbook -i hosts.ini install_nginx.yml