Academic Year: 2024-25 **Semester: V**

Class / Branch: TEIT

Subject: DevOps Lab

Name of Instructor: Ms. Sujata Oak / Ms. Sonal Jain

Experiment No. 3

Aim: To understand and perform version control system / source code management using Git.

GIT is a Version Control System (VCS) (aka Revision Control System (RCS), Source Code Manager (SCM)). A VCS serves as a Repository (or repo) of program codes, including all the historical revisions. It records changes to files at so-called commits in a log so that you can recall any file at any commit point.

To issue a command, start a "Terminal" (for Ubuntu/Mac) or "Git Bash" (for Windows):

```
$ git <command> <arguments>
```

The commonly-used commands are:

1. **init**, **clone**, **config**: for starting a Git-managed project.
2. **add**, **mv**, **rm**: for staging file changes.
3. **commit**, **rebase**, **reset**, **tag**:
4. **status**, **log**, **diff**, **grep**, **show**: show status
5. **checkout**, **branch**, **merge**, **push**, **fetch**, **pull**

Getting Started with Local Repo

There are 2 ways to start a Git-managed project:

1. Starting your own project;
2. Cloning an existing project from a GIT host.

Git uses two stages to commit file changes:

1. "git add <file>" to stage file changes into the staging area, and
2. "git commit" to commit ALL the file changes in the staging area to the local repo.

Prerequisite: Commands of Exp 2

PART 1: Branching and Merging

When a repository is created, the files are automatically put in a branch called **main**. Whenever possible it is recommended to use branches rather than directly updating the main branch. Branching is used so that you can make changes in another area without affecting the main branch. This is done to help prevent accidental updates that might overwrite existing code.

In this part, you will :

- **create a new branch**,
- **checkout the branch**,
- **make changes in the branch**,
- **stage**
- **commit** the branch
- **merge** the branch changes to the main branch, and
- **delete** the branch.

```
sujata@Ubuntu:~$ su root
Password:
root@Ubuntu:/home/sujata#
```

```
root@Ubuntu:/home/sujata# cd workspace/
root@Ubuntu:/home/sujata/workspace# ls
hello_test
```

Step1: Create a new branch

Create a new branch called **feature** using the **git branch** <branch-name> command

```
root@Ubuntu:/home/sujata/workspace# cd hello_test/
root@Ubuntu:/home/sujata/workspace/hello_test# git branch feature
```

Step 2.

- Verify Current branch

  Use the **git branch** command without a branch-name to display all the branches for this repository.

  The "*" next to the main branch indicates that this is the current branch – the branch that is currently "checked out".

```
root@Ubuntu:/home/sujata/workspace/hello_test# git branch
  feature
* main
```

Step 3: Checkout the new branch

Use the **git checkout** <branch-name> command to switch to the *feature* branch.

```
root@Ubuntu:/home/sujata/workspace/hello_test# git checkout feature
Switched to branch 'feature'
```

Step 4: Verify the current branch:

a. Verify you have switched to the *feature* branch using the **git branch** command. Note the "*" next to the *feature* branch. This is now the *working branch*.

```
root@Ubuntu:/home/sujata/workspace/hello_test# git branch
* feature
  main
```

b. Append a new line of text to the hello.py file, again using the **echo** command with the ">>" signs.

```
root@Ubuntu:/home/sujata/workspace/hello_test# echo " This text was ad
ded originally while in the feature branch" >> hello.py
```

c. Verify the line was appended to the file using the **cat** command.

```
root@Ubuntu:/home/sujata/workspace/hello_test# cat hello.py
    num = int(input("Enter a number: "))
    factorial = 1
    if num < 0:
       print(" Factorial does not exist for negative numbers")
    elif num == 0:
       print("The factorial of 0 is 1")
    else:
       for i in range(1,num + 1):
           factorial = factorial*i
       print("The factorial of",num,"is",factorial)
 This text was added originally while in the feature branch
```

Step 5: Stage the modified file in the feature branch

a. Stage the updated file to the current *feature* branch.

git status

```
root@Ubuntu:/home/sujata/workspace/hello_test# git status
On branch feature
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory
)
        modified:   hello.py

no changes added to commit (use "git add" and/or "git commit -a")
```

git add hello.py

```
root@Ubuntu:/home/sujata/workspace/hello_test# git add hello.py
```

B] Use the **git status** command and notice the modified file **hello.py** is staged in the *feature* branch
git status

```
root@Ubuntu:/home/sujata/workspace/hello_test# git status
On branch feature
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   hello.py
```

Step 6: Commit the staged file in the feature branch

A] Commit the staged file using the **git commit** command. Notice the new commit ID and your message.

```
root@Ubuntu:/home/sujata/workspace/hello_test# git commit -m "Added a
line in feature branch"
[feature 67139aa] Added a line in feature branch
 1 file changed, 1 insertion(+)
```

B] Use the **git log** command to show all commits including the commit you just did to the *feature* branch. The prior commit was done within the *main* branch.

```
root@Ubuntu:/home/sujata/workspace/hello_test# git log
commit 67139aab359af707a18e430d0dc6287e487a1d2e (HEAD -> feature)
Author: sujataoak799 <sujataoak2021@gmail.com>
Date:   Sun Jul 21 15:24:57 2024 +0530

    Added a line in feature branch

commit f1bdb1cc96f8e879529d5f765476a15ee490a5c6 (main)
Author: sujataoak799 <sujataoak2021@gmail.com>
Date:   Tue Jul 16 10:48:54 2024 +0530

    Revert "Initial commit"
```

Step 7: Checkout the main branch

Switch to the main branch using the **git checkout** *main* command and verify the current working branch using the **git branch** command.

```
root@Ubuntu:/home/sujata/workspace/hello_test# git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)
```

```
root@Ubuntu:/home/sujata/workspace/hello_test# git branch
  feature
* main
```

Step 8: Merge file contents from feature to main branch.

a. Branches are often used when implementing new features or fixes. They can be submitted for review by team members, and then once verified, can be pulled into the main codebase – the main branch.

Merge the contents (known as the history) from the feature branch into the main branch using the **git merge** <branch-name> command. The branch-name is the branch that histories are pulled from into the current branch. The output displays that one file was changed with one line inserted.

```
root@Ubuntu:/home/sujata/workspace/hello_test# git merge feature
Updating f1bdb1c..67139aa
Fast-forward
 hello.py | 1 +
 1 file changed, 1 insertion(+)
```

B] Verify the appended content to the hello.py file in the main branch using the **cat** command.

```
root@Ubuntu:/home/sujata/workspace/hello_test# cat hello.py
    num = int(input("Enter a number: "))
    factorial = 1
    if num < 0:
       print(" Factorial does not exist for negative numbers")
    elif num == 0:
       print("The factorial of 0 is 1")
    else:
       for i in range(1,num + 1):
           factorial = factorial*i
       print("The factorial of",num,"is",factorial)
 This text was added originally while in the feature branch
```

Step 9: Push the changes from local to remote repository

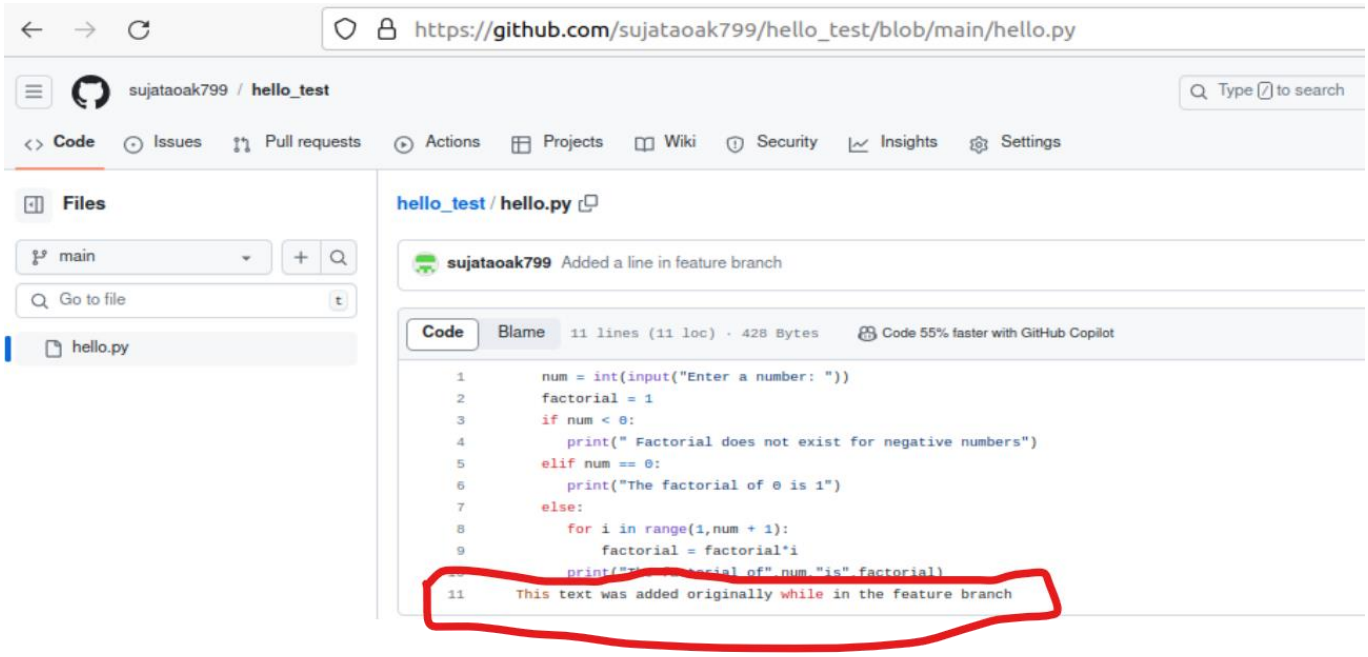
```
root@Ubuntu:/home/sujata/workspace/hello_test# git push -u origin main
Password for 'https://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@github.com':
remote: Support for password authentication was removed on August 13, 2021.
remote: Please see https://docs.github.com/get-started/getting-started-with-git/abou
t-remote-repositories#cloning-with-https-urls for information on currently recommend
ed modes of authentication.
fatal: Authentication failed for 'https://github.com/sujataoak799/hello_test/'
```

Generate a classic token first:

```
root@Ubuntu:/home/sujata/workspace/hello_test# git remote set-url origin https://ghp
_ZaLvsbTPgxILguomA7MJ2Wwa52NpCX3iwWij@github.com/sujataoak799/hello_test
```

```
root@Ubuntu:/home/sujata/workspace/hello_test# git push -u origin main
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (5/5), 596 bytes | 596.00 KiB/s, done.
Total 5 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/sujataoak799/hello_test
   e19123e..67139aa  main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

STEP 11] Goto Github Account and see the update



STEP 10] Deleting a branch

A] Verify the **feature** branch is still available using the **git branch** command.

```
root@Ubuntu:/home/sujata/workspace/hello_test# git branch
  feature
* main
```

B] Delete the **feature** branch using the **git branch -d** <branch-name> command

```
root@Ubuntu:/home/sujata/workspace/hello_test# git branch -d feature
Deleted branch feature (was 67139aa).
```

C] Verify the feature branch is no longer available using the **git branch** command

```
root@Ubuntu:/home/sujata/workspace/hello_test# git branch
* main
```

Conclusion:

In this experiment, we understood the use case of Version Control System in branching

and merging, its benefits in real time scenario which provides a application of branching the changes when people are in working in a collaborating environment. Different commands were used for the same such as checkout, branch and merge for displaying the changes between the initial and latter texts.