**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.

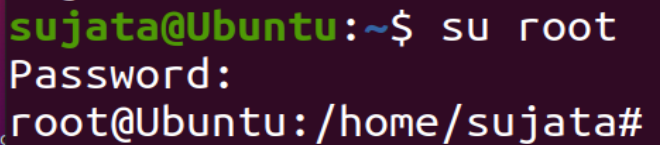
**Prerequisite: Commands of Exp 2**

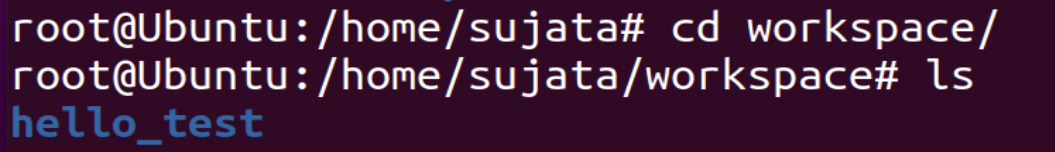
**PART 1: Branching and Merging**

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 :

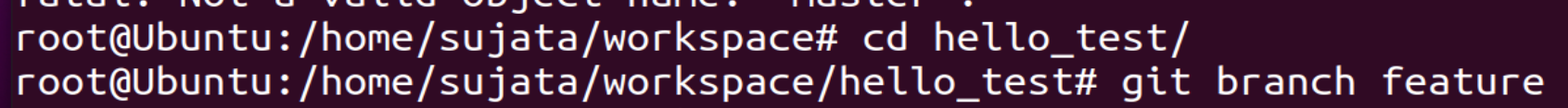
* **create a new branch**,
* **checkout the branch**,
* **make changes in the branch**,
* **stage**
* **commit** the branch
* **merge** the branch changes to the main branch, and
* **delete** the branch.

****

****

**Step1:** Create a new branch

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

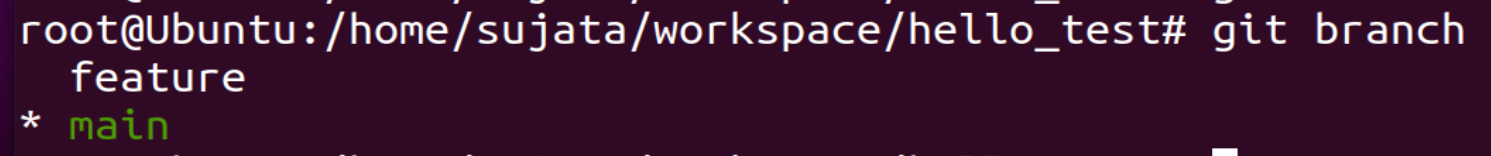
****

**Step 2.**

* Verify Current branch

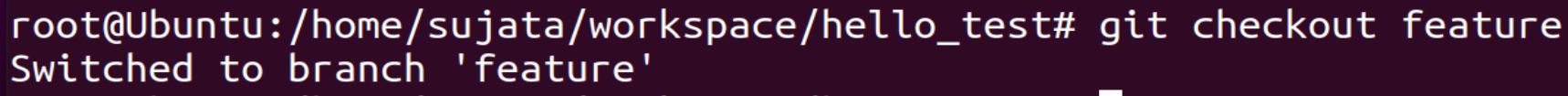
Use the **git branch** command without a branch-name to display all the branches for this repository.

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



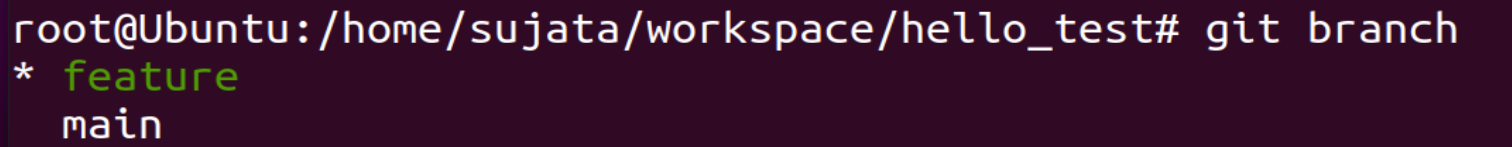
**Step 3**: Checkout the new branch

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

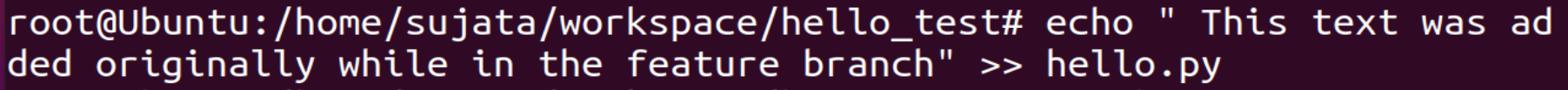


**Step 4:** Verify the current branch:

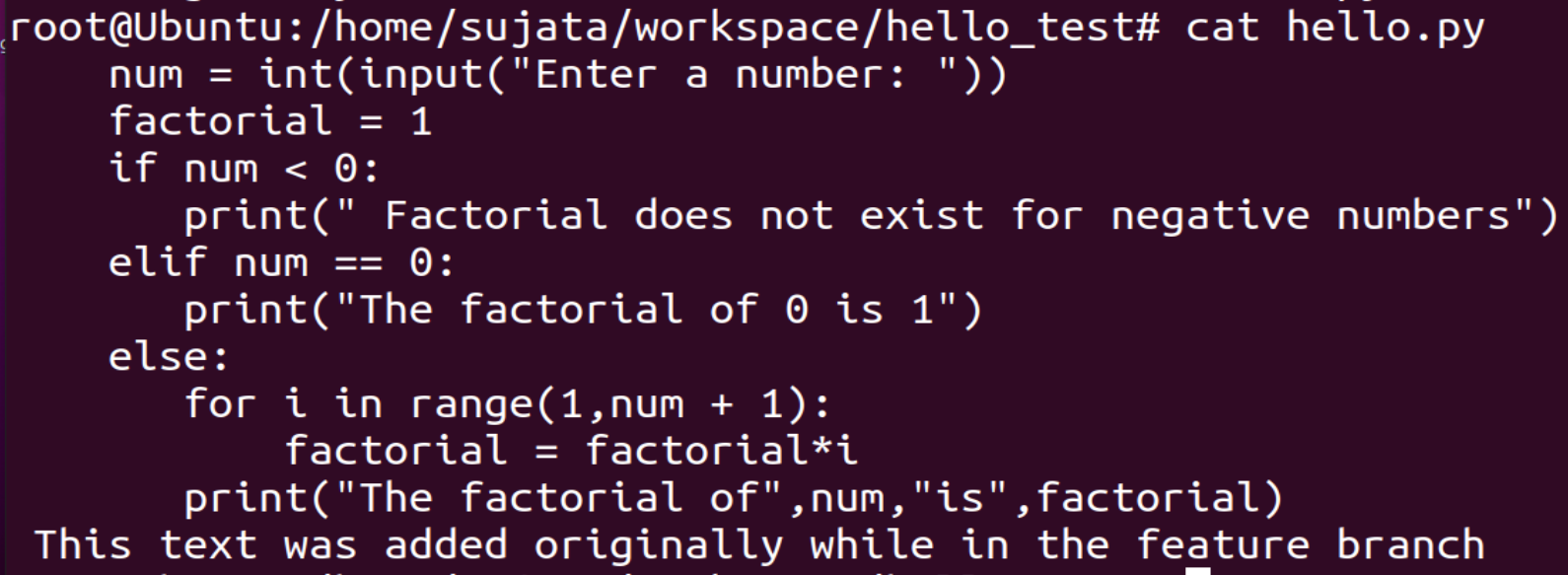
* + - 1. Verify you have switched to the *feature* branch using the **git branch** command. Note the "\*" next to the *feature* branch. This is now the *working branch*.



* + - 1. Append a new line of text to the hello.py file, again using the **echo** command with the ">>" signs.



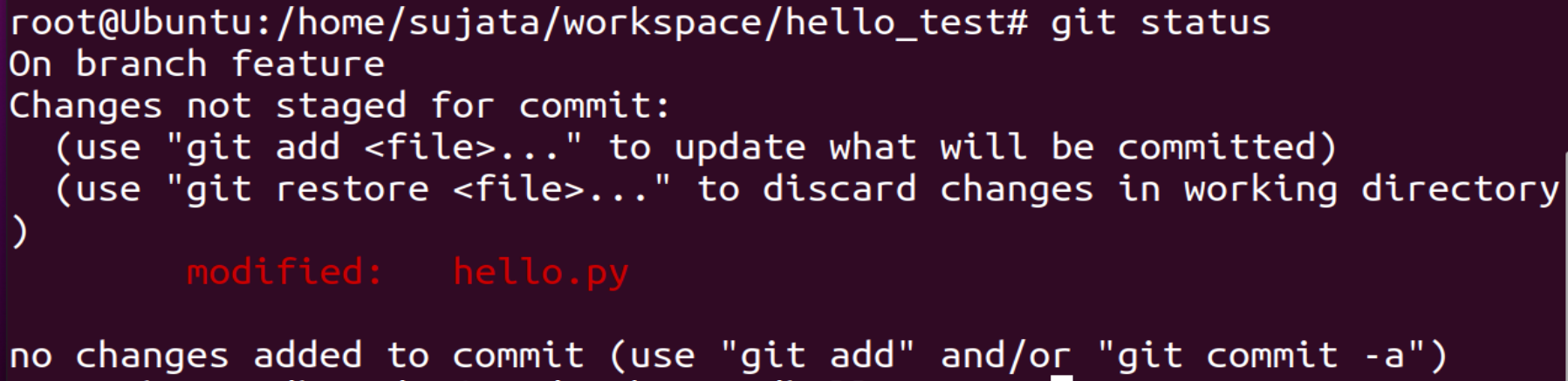
* + - 1. Verify the line was appended to the file using the **cat** command.



**Step 5**: Stage the modified file in the feature branch

* + - 1. Stage the updated file to the current *feature* branch.

git status

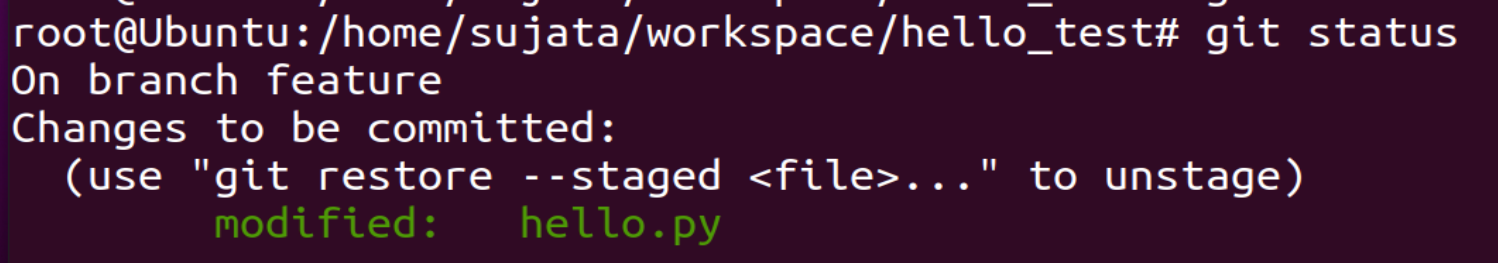


git add hello.py



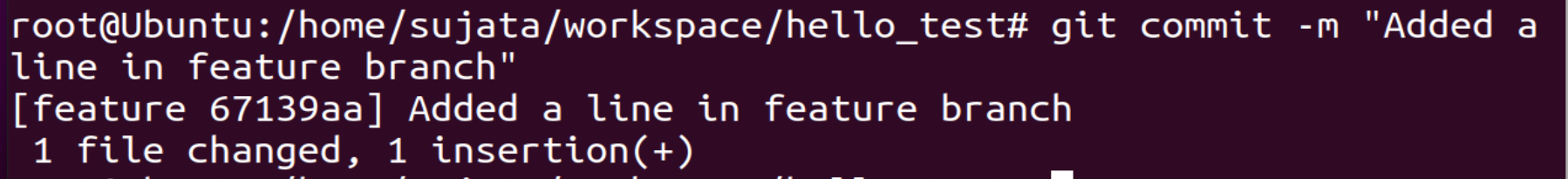
B] Use the **git status** command and notice the modified file **hello.py** is staged in the *feature* branch

git status

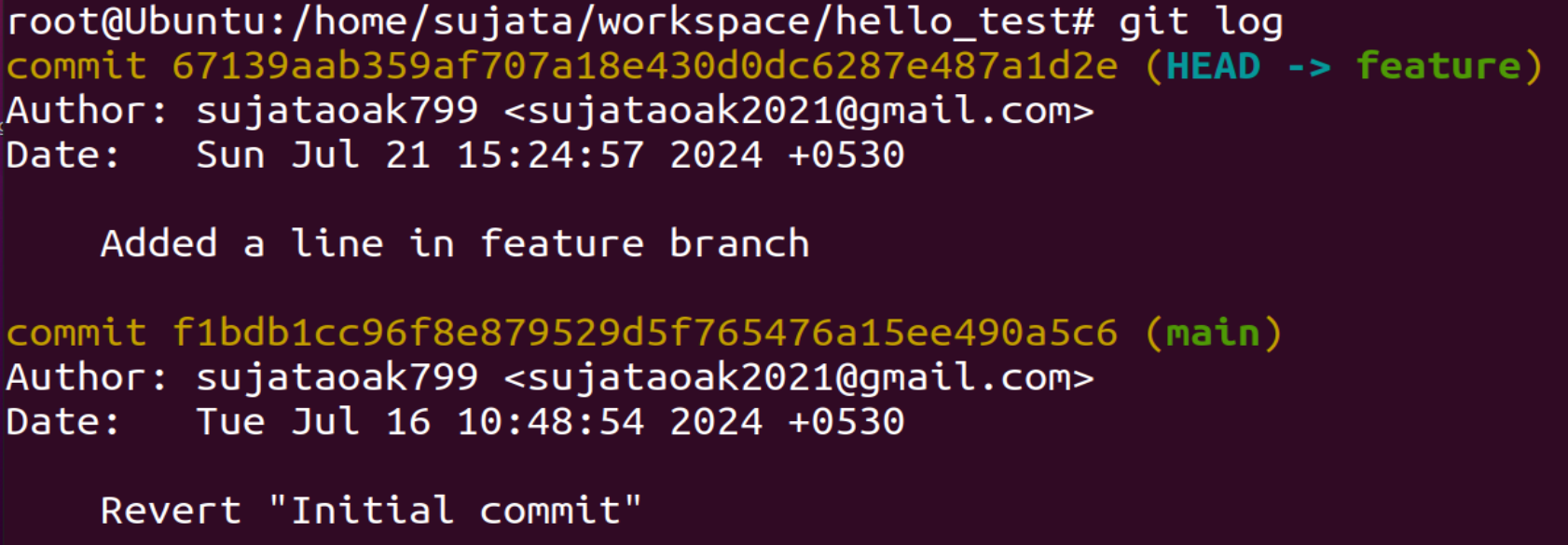


**Step 6: Commit the staged file in the feature branch**

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

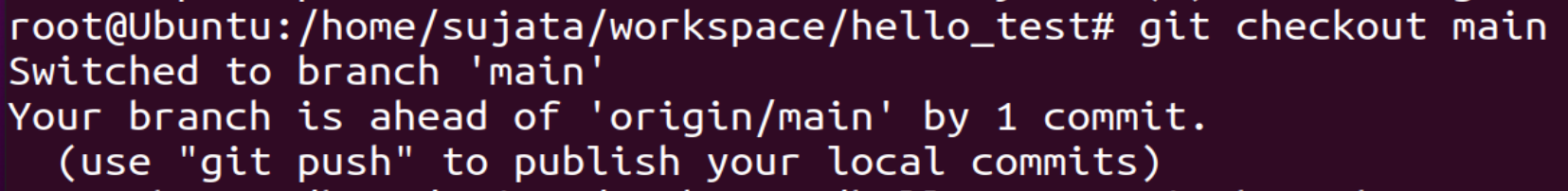


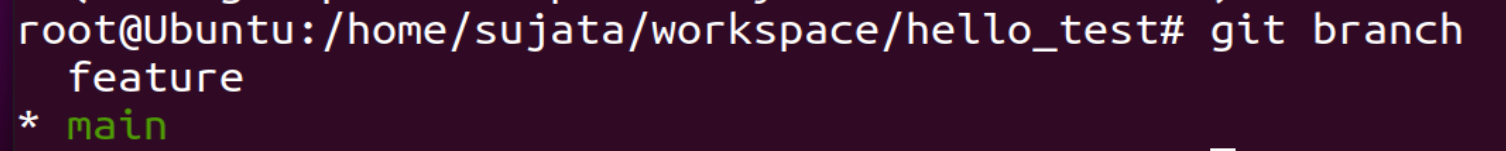
B] Use the **git log** command to show all commits including the commit you just did to the *feature* branch. The prior commit was done within the *main* branch.



**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.

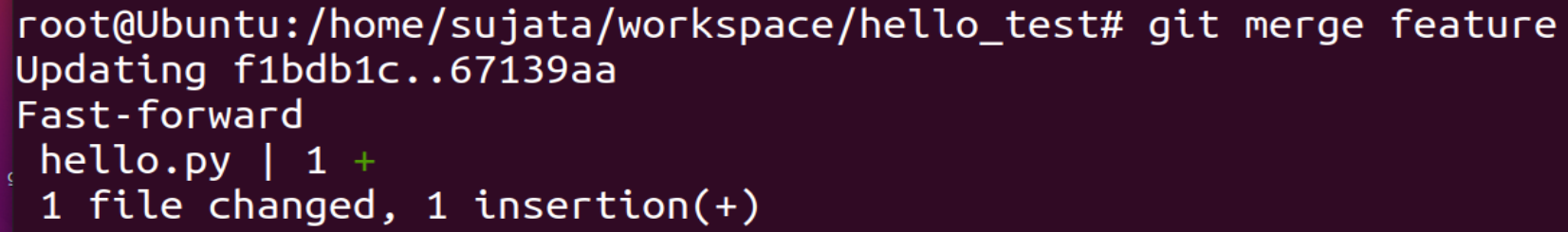
****

****

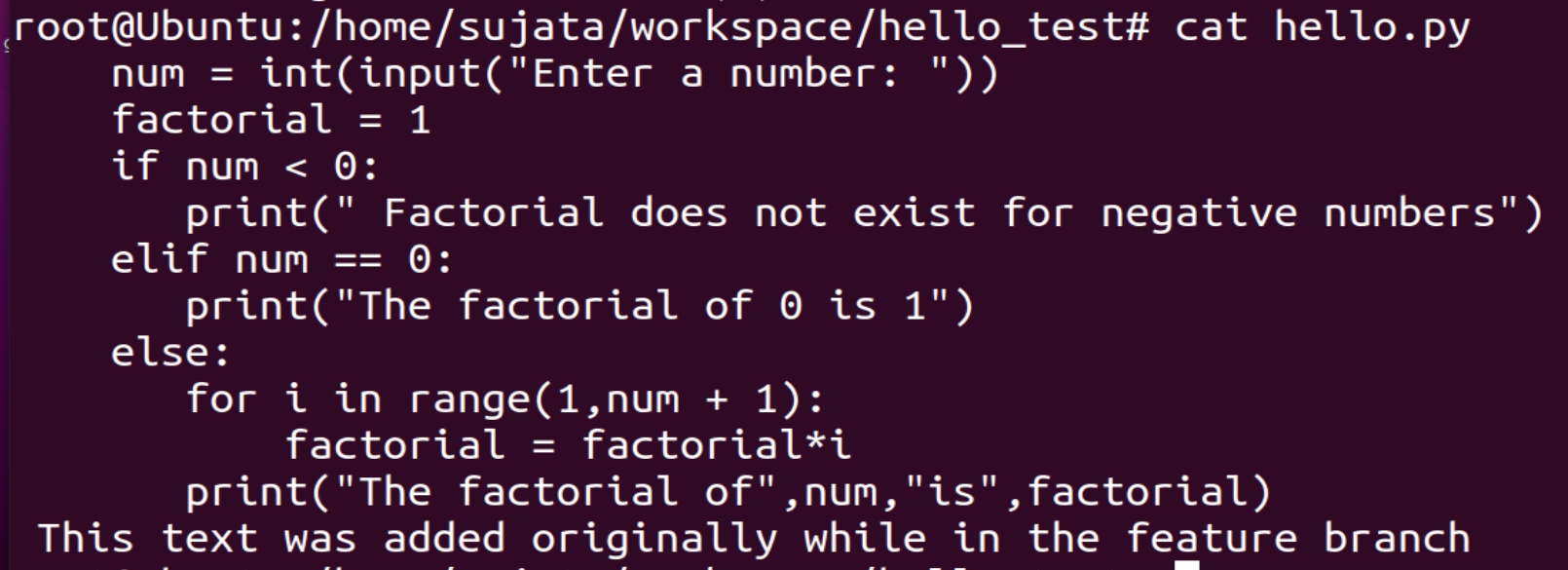
**Step 8:** **Merge file contents from feature to main branch.**

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.

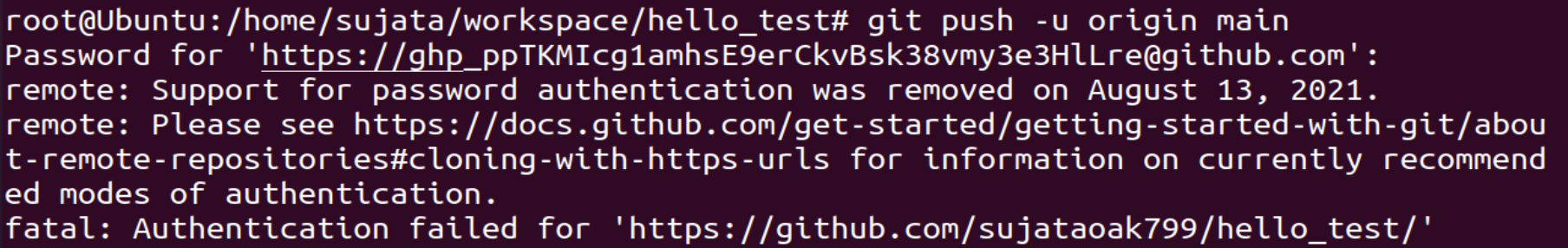
Merge the contents (known as the history) from the feature branch into the main branch using the **git merge** <branch-name> command. The branch-name is the branch that histories are pulled from into the current branch. The output displays that one file was changed with one line inserted.



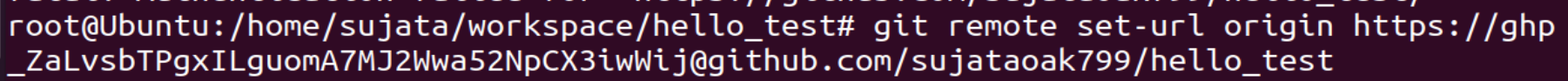
B] Verify the appended content to the hello.py file in the main branch using the **cat** command.

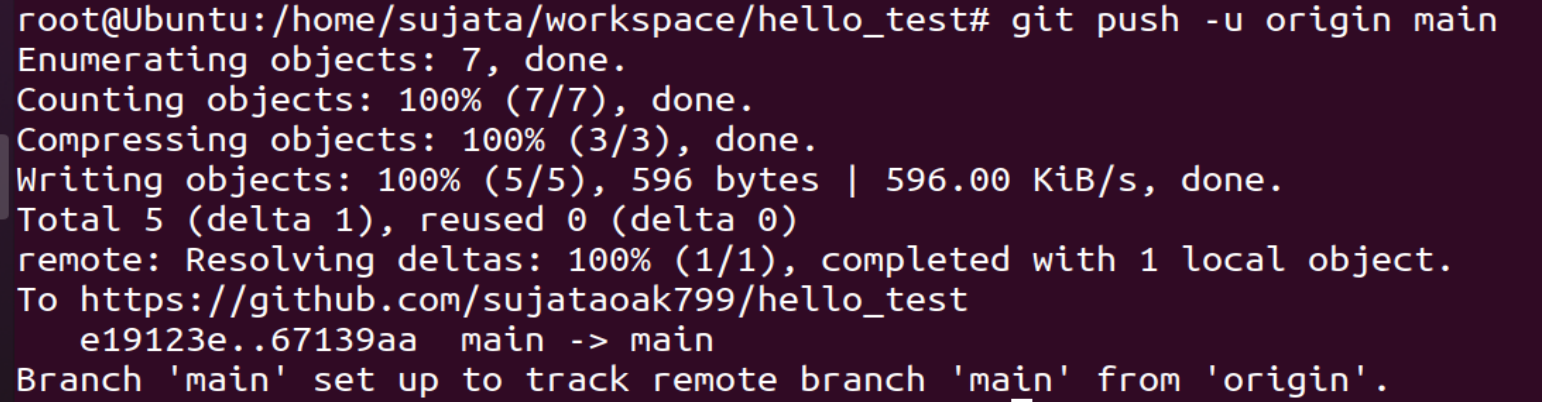


Step 9: Push the changes from local to remote repository

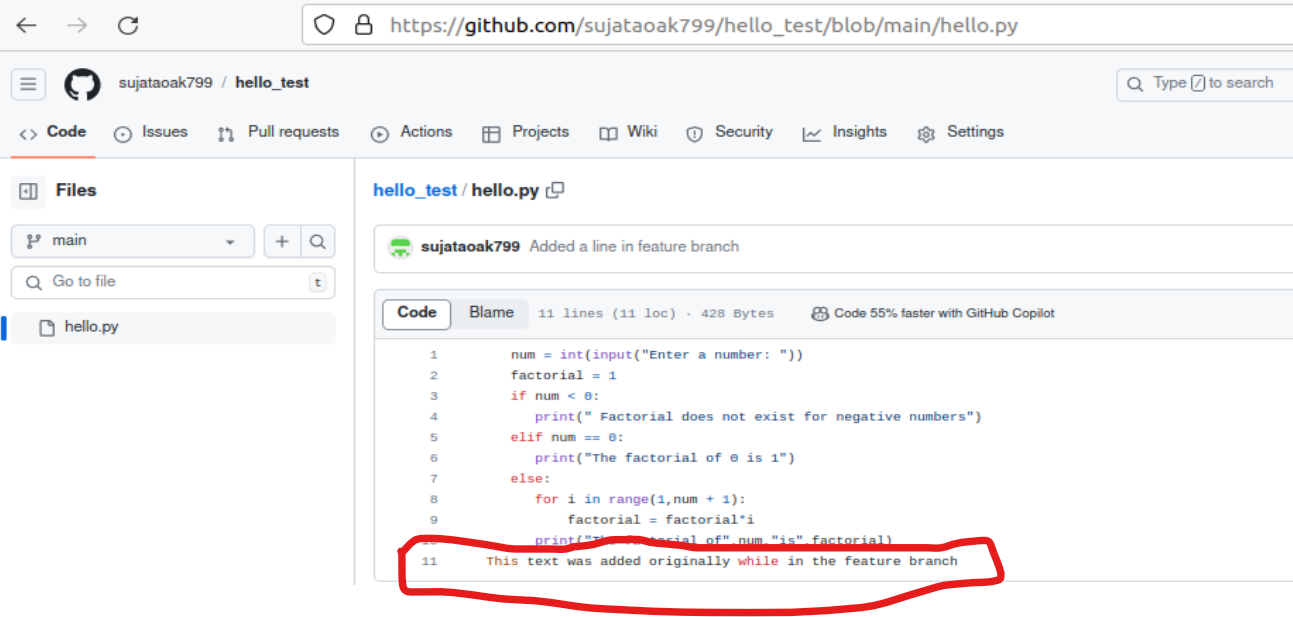


Generate a classic token first:



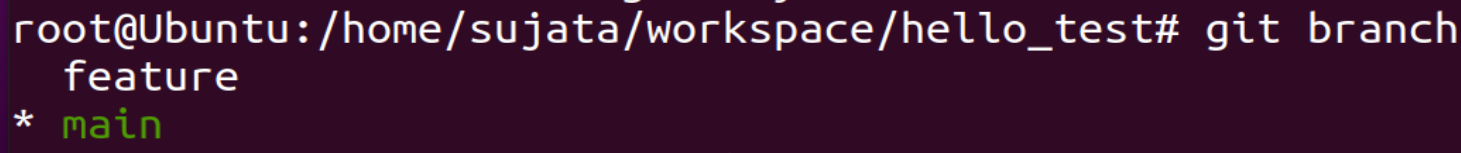


STEP 11] Goto Github Account and see the update

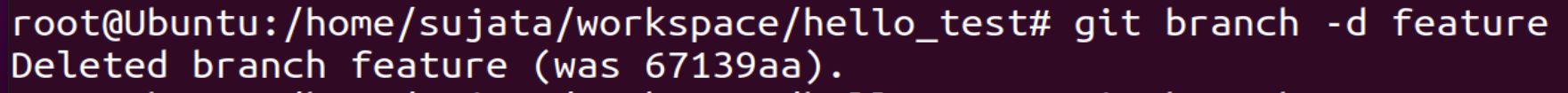


STEP 10] Deleting a branch

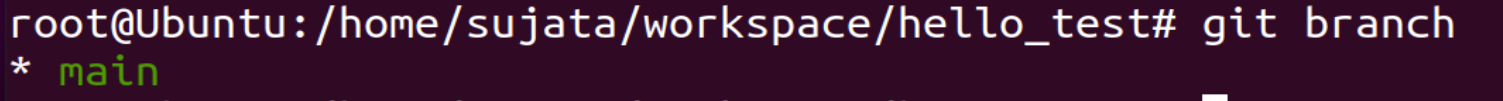
A] Verify the **feature** branch is still available using the **git branch** command.



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



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



# 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.