Experiment No. 1

Title:

Exploring Git Commands through Collaborative Coding.

Objective:

The objective of this experiment is to familiarize participants with essential Git concepts and commands, enabling them to effectively use Git for version control and collaboration.

Introduction:

Git is a distributed version control system (VCS) that helps developers track changes in their codebase, collaborate with others, and manage different versions of their projects efficiently. It was created by Linus Torvalds in 2005 to address the shortcomings of existing version control systems.

Unlike traditional centralized VCS, where all changes are stored on a central server, Git follows a distributed model. Each developer has a complete copy of the repository on their local machine, including the entire history of the project. This decentralization offers numerous advantages, such as offline work, faster operations, and enhanced collaboration. Git is a widely used version control system that allows developers to collaborate on projects, track changes, and manage codebase history efficiently. This experiment aims to provide a hands-on introduction to Git and explore various fundamental Git commands. Participants will learn how to set up a Git repository, commit changes, manage branches, and collaborate with others using Git.

Key Concepts:

● Repository: A Git repository is a collection of files, folders, and their historical versions. It contains all the information about the project's history, branches, and commits.

● Commit: A commit is a snapshot of the changes made to the files in the repository at a specific point in time. It includes a unique identifier (SHA-1 hash), a message describing the changes, and a reference to its parent commit(s).

● Branch: A branch is a separate line of development within a repository. It allows developers to work on new features or bug fixes without affecting the main codebase. Branches can be merged back into the main branch when the changes are ready.

● Merge: Merging is the process of combining changes from one branch into another. It integrates the changes made in a feature branch into the main branch or any other target branch.

● Pull Request: In Git hosting platforms like GitHub, a pull request is a feature that allows developers to propose changes from one branch to another. It provides a platform for code review and collaboration before merging.

● Remote Repository: A remote repository is a copy of the Git repository stored on a server, enabling collaboration among multiple developers. It can be hosted on platforms like GitHub, GitLab, or Bitbucket.

Basic Git Commands:

● `git init`: Initialises a new Git repository in the current directory.

● `git clone`: Creates a copy of a remote repository on your local machine.

● `git add`: Stages changes for commit, preparing them to be included in the next commit.

● `git commit`: Creates a new commit with the staged changes and a descriptive message.

● `git status`: Shows the current status of the working directory, including tracked and untracked files.

● `git log`: Displays a chronological list of commits in the repository, showing their commit messages, authors, and timestamps.

● `git branch`: Lists, creates, or deletes branches within the repository.

● `git checkout`: Switches between branches, commits, or tags. It's used to navigate through the repository's history.

● `git merge`: Combines changes from different branches, integrating them into the current branch.

● `git pull`: Fetches changes from a remote repository and merges them into the current branch.

● `git push`: Sends local commits to a remote repository, updating it with the latest changes.

Prerequisites:

● Computer with Git installed (https://git-scm.com/downloads)

● Command-line interface (Terminal, Command Prompt, or Git Bash)

Experiment Steps:

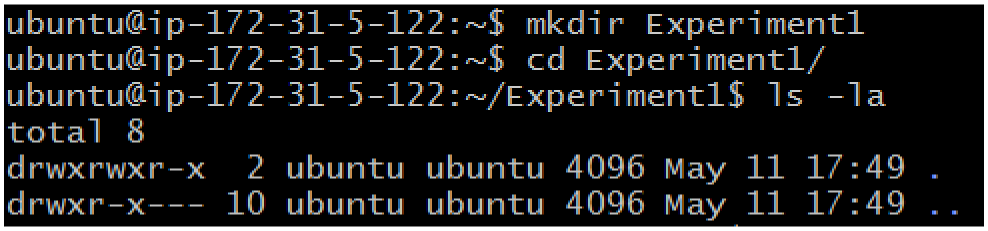
Step 1: Setting Up Git Repository

● Open the command-line interface on your computer.

● Navigate to the directory where you want to create your Git repository.

● Run the following commands:

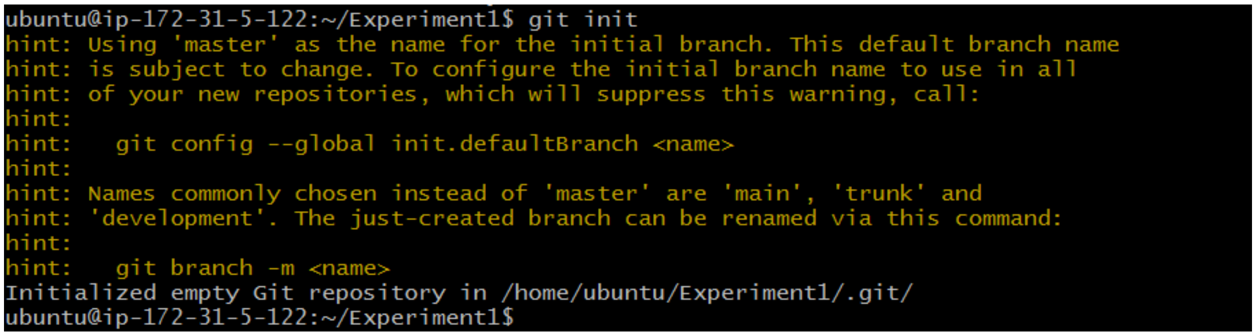
```
mkdir Experiment1

cd Experiment1

ls -la
```



```
ubuntu@ip-172-31-5-122:~$ mkdir Experiment1
ubuntu@ip-172-31-5-122:~$ cd Experiment1/
ubuntu@ip-172-31-5-122:~/Experiment1$ ls -la
total 8
drwxrwxr-x  2 ubuntu ubuntu 4096 May 11 17:49 .
drwxr-x--- 10 ubuntu ubuntu 4096 May 11 17:49 ..
```

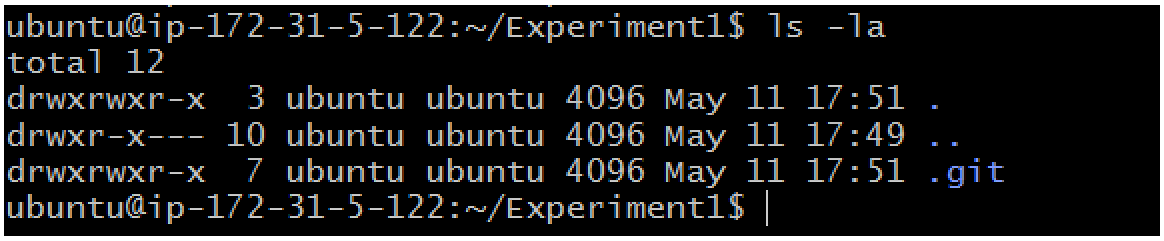
```
git init
```

This initializes a new Git repository in the current directory.



```
ubuntu@ip-172-31-5-122:~/Experiment1$ git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /home/ubuntu/Experiment1/.git/
ubuntu@ip-172-31-5-122:~/Experiment1$
```

```
ls -la
```



```
ubuntu@ip-172-31-5-122:~/Experiment1$ ls -la
total 12
drwxrwxr-x  3 ubuntu ubuntu 4096 May 11 17:51 .
drwxr-x--- 10 ubuntu ubuntu 4096 May 11 17:49 ..
drwxrwxr-x  7 ubuntu ubuntu 4096 May 11 17:51 .git
ubuntu@ip-172-31-5-122:~/Experiment1$ |
```

Step 2: Creating and Committing Changes

● Create a new text file named "`example.txt`" using any text editor.

● Add some content to the "`example.txt`" file.

```
touch example.txt
```

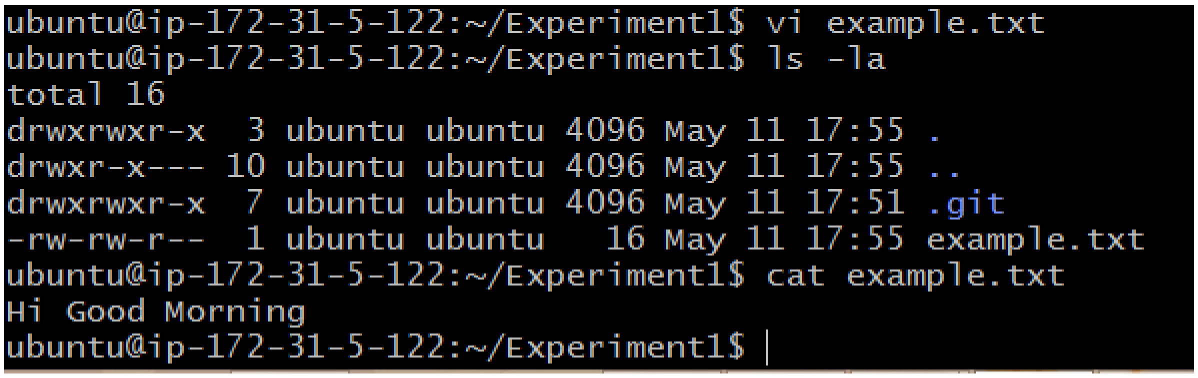
```
vi example.txt
```

OR directly:

```
vi example.txt
```

```
ls -la
```

```
cat example.txt
```

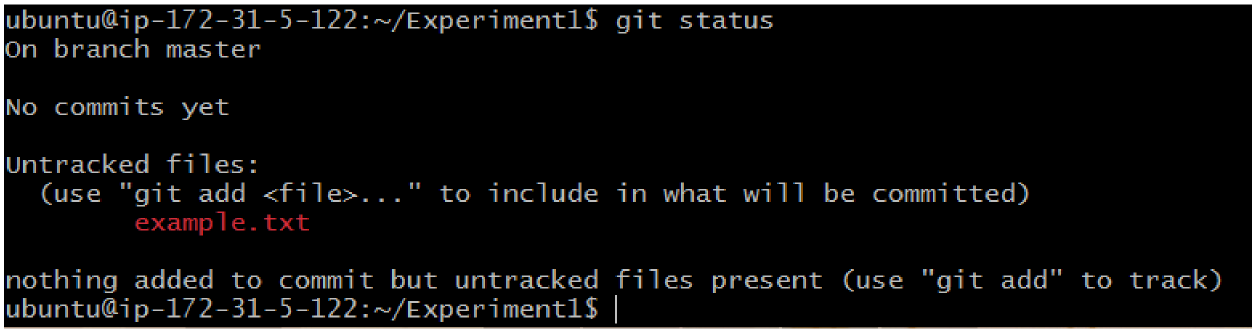


```
ubuntu@ip-172-31-5-122:~/Experiment1$ vi example.txt
ubuntu@ip-172-31-5-122:~/Experiment1$ ls -la
total 16
drwxrwxr-x  3 ubuntu ubuntu 4096 May 11 17:55 .
drwxr-x--- 10 ubuntu ubuntu 4096 May 11 17:55 ..
drwxrwxr-x  7 ubuntu ubuntu 4096 May 11 17:51 .git
-rw-rw-r--  1 ubuntu ubuntu   16 May 11 17:55 example.txt
ubuntu@ip-172-31-5-122:~/Experiment1$ cat example.txt
Hi Good Morning
ubuntu@ip-172-31-5-122:~/Experiment1$ |
```

● In the command-line interface, run the following commands:

```
git status
```

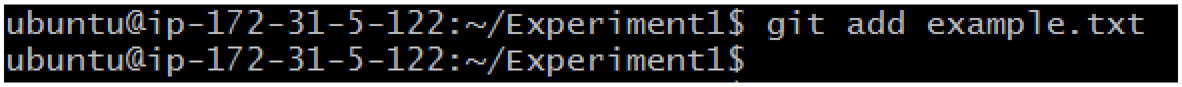
This command shows the status of your working directory, highlighting untracked files.



```
ubuntu@ip-172-31-5-122:~/Experiment1$ git status
On branch master

No commits yet

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

nothing added to commit but untracked files present (use "git add" to track)
ubuntu@ip-172-31-5-122:~/Experiment1$ |
```

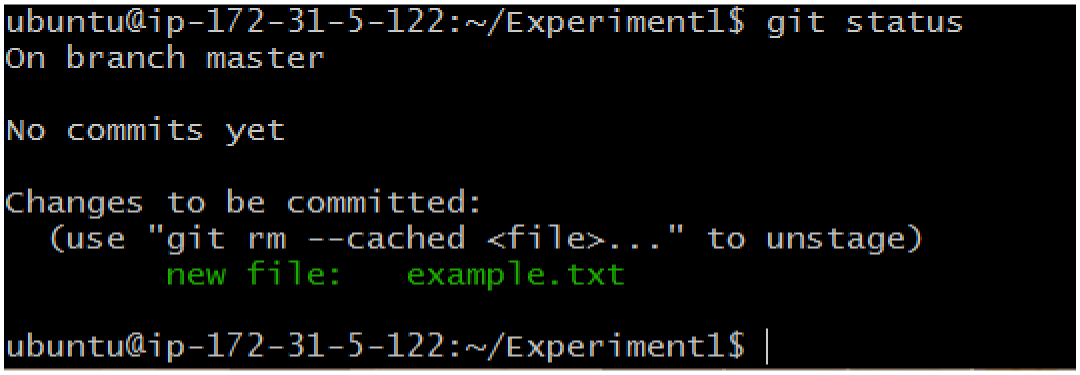
```
git add example.txt
```

This stages the changes of the "`example.txt`" file for commit.



```
ubuntu@ip-172-31-5-122:~/Experiment1$ git add example.txt
ubuntu@ip-172-31-5-122:~/Experiment1$
```

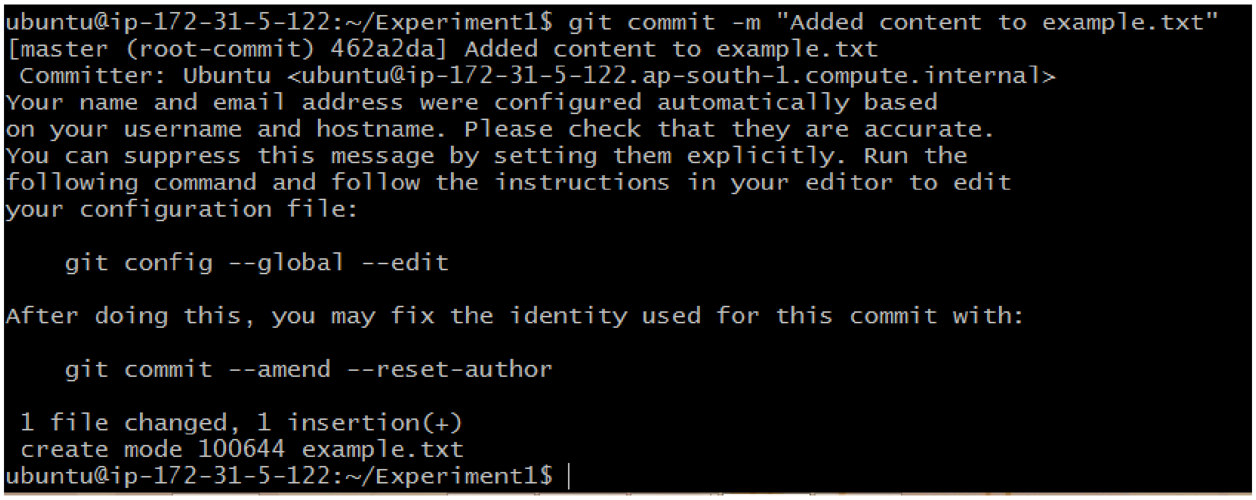
```
git status
```



```
ubuntu@ip-172-31-5-122:~/Experiment1$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   example.txt

ubuntu@ip-172-31-5-122:~/Experiment1$ |
```

```
git commit -m "Added content to example.txt"
```

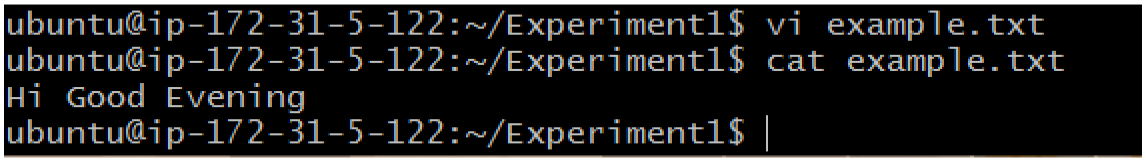
This commits the staged changes with a descriptive message.



```
ubuntu@ip-172-31-5-122:~/Experiment1$ git commit -m "Added content to example.txt"
[master (root-commit) 462a2da] Added content to example.txt
 Committer: Ubuntu <ubuntu@ip-172-31-5-122.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

 1 file changed, 1 insertion(+)
 create mode 100644 example.txt
ubuntu@ip-172-31-5-122:~/Experiment1$ |
```

Step 3: Exploring History

Modify the content of "`example.txt`"

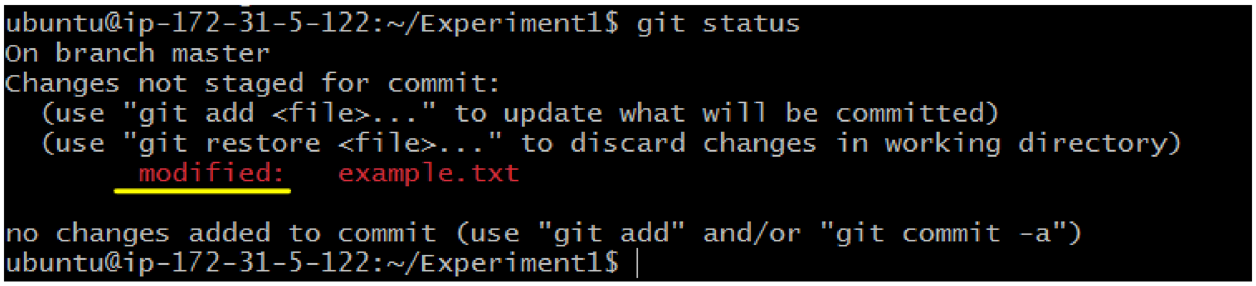


```
ubuntu@ip-172-31-5-122:~/Experiment1$ vi example.txt
ubuntu@ip-172-31-5-122:~/Experiment1$ cat example.txt
Hi Good Evening
ubuntu@ip-172-31-5-122:~/Experiment1$ |
```

Run the following commands:

```
git status
```

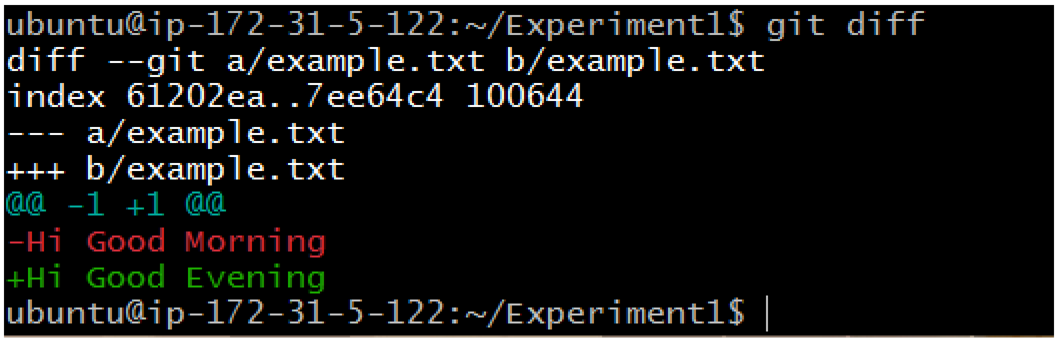
Notice the modified file is shown as "`modified`".



```
ubuntu@ip-172-31-5-122:~/Experiment1$ git status
On branch master
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:   example.txt

no changes added to commit (use "git add" and/or "git commit -a")
ubuntu@ip-172-31-5-122:~/Experiment1$ |
```

```
git diff
```

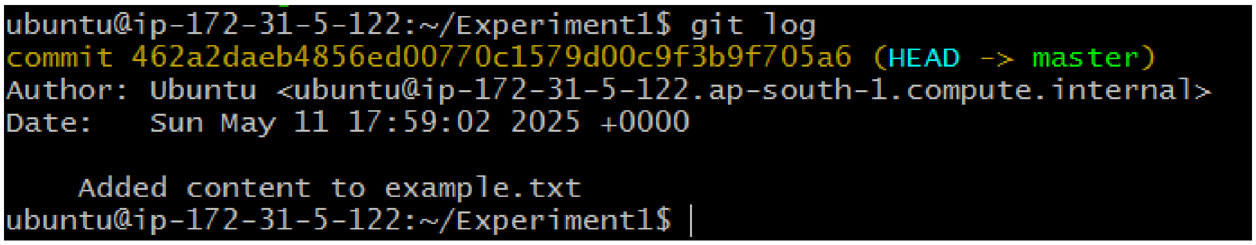
This displays the differences between the last commit and the working directory.



```
ubuntu@ip-172-31-5-122:~/Experiment1$ git diff
diff --git a/example.txt b/example.txt
index 61202ea..7ee64c4 100644
--- a/example.txt
+++ b/example.txt
@@ -1 +1 @@
-Hi Good Morning
+Hi Good Evening
ubuntu@ip-172-31-5-122:~/Experiment1$ |
```

```
git log
```

This displays a chronological history of commits.

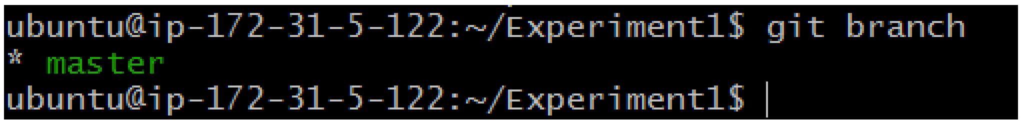


```
ubuntu@ip-172-31-5-122:~/Experiment1$ git log
commit 462a2daeb4856ed00770c1579d00c9f3b9f705a6 (HEAD -> master)
Author: Ubuntu <ubuntu@ip-172-31-5-122.ap-south-1.compute.internal>
Date:   Sun May 11 17:59:02 2025 +0000

    Added content to example.txt
ubuntu@ip-172-31-5-122:~/Experiment1$ |
```

Step 4: Branching and Merging

Check the current branch

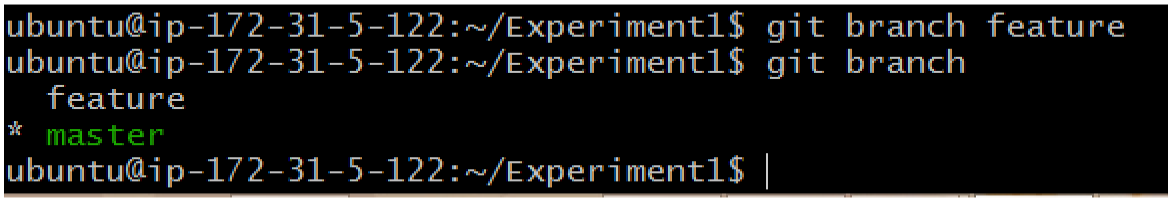
```
git branch
```



```
ubuntu@ip-172-31-5-122:~/Experiment1$ git branch
* master
ubuntu@ip-172-31-5-122:~/Experiment1$ |
```

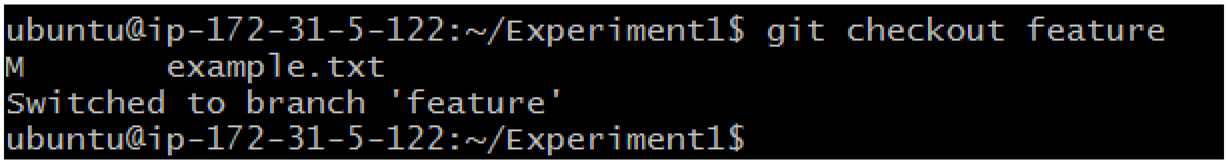
Create a new branch named "`feature`" and switch to it:

```
git branch feature

git branch
```



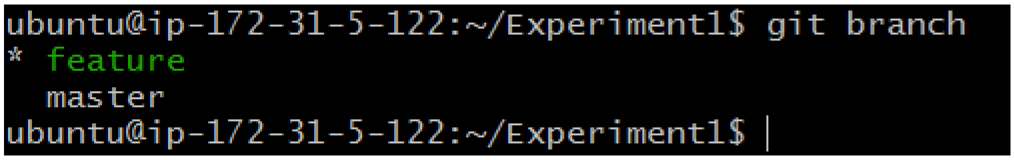
```
ubuntu@ip-172-31-5-122:~/Experiment1$ git branch feature
ubuntu@ip-172-31-5-122:~/Experiment1$ git branch
  feature
* master
ubuntu@ip-172-31-5-122:~/Experiment1$ |
```

```
git checkout feature
```



```
ubuntu@ip-172-31-5-122:~/Experiment1$ git checkout feature
M       example.txt
Switched to branch 'feature'
ubuntu@ip-172-31-5-122:~/Experiment1$
```

```
git branch
```



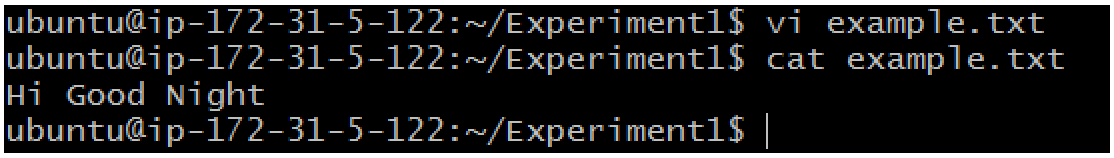
```
ubuntu@ip-172-31-5-122:~/Experiment1$ git branch
* feature
  master
ubuntu@ip-172-31-5-122:~/Experiment1$ |
```

OR shorthand:

```
git checkout -b feature
```

● Make changes to the "`example.txt`" file in the "`feature`" branch.

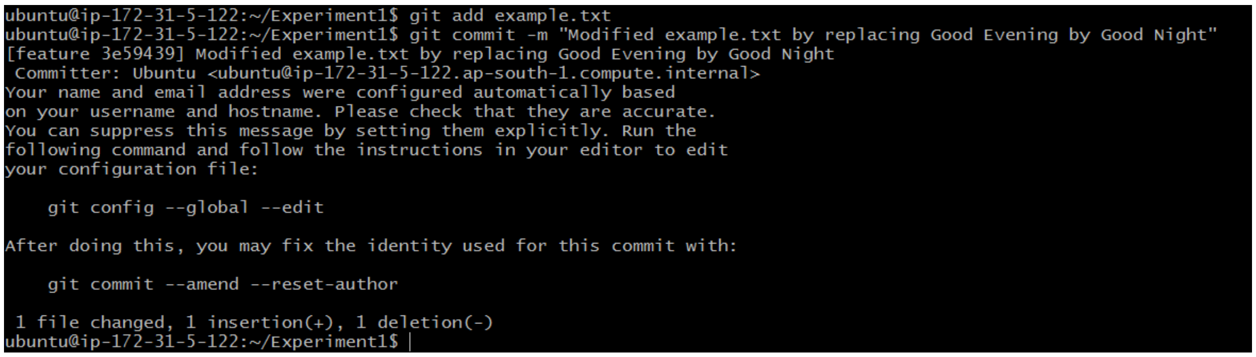
```
vi example.txt

cat example.txt
```



```
ubuntu@ip-172-31-5-122:~/Experiment1$ vi example.txt
ubuntu@ip-172-31-5-122:~/Experiment1$ cat example.txt
Hi Good Night
ubuntu@ip-172-31-5-122:~/Experiment1$ |
```

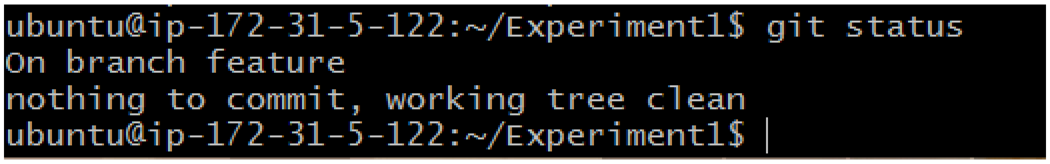
● Commit the changes in the "`feature`" branch.

```
git add example.txt

git commit -m "Modified example.txt by replacing Good Evening by
Good Night"
```



```
ubuntu@ip-172-31-5-122:~/Experiment1$ git add example.txt
ubuntu@ip-172-31-5-122:~/Experiment1$ git commit -m "Modified example.txt by replacing Good Evening by Good Night"
[feature 3e59439] Modified example.txt by replacing Good Evening by Good Night
 Committer: Ubuntu <ubuntu@ip-172-31-5-122.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

 1 file changed, 1 insertion(+), 1 deletion(-)
ubuntu@ip-172-31-5-122:~/Experiment1$ |
```

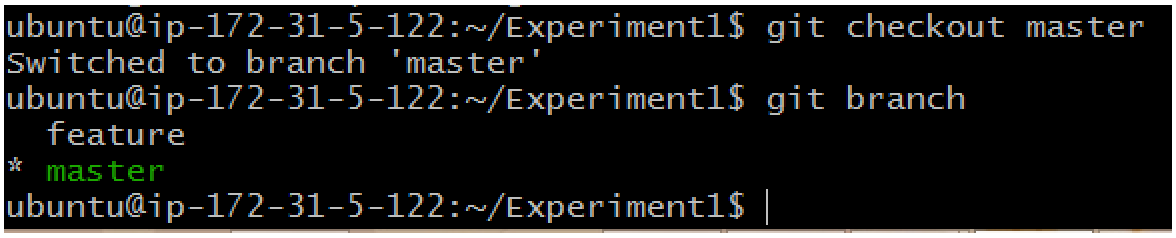
```
git status
```



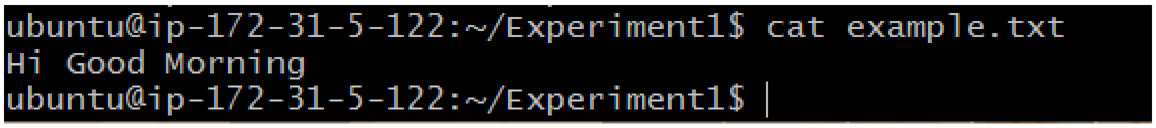
```
ubuntu@ip-172-31-5-122:~/Experiment1$ git status
On branch feature
nothing to commit, working tree clean
ubuntu@ip-172-31-5-122:~/Experiment1$ |
```

● Switch back to the "`master`" branch:

```
git checkout master

git branch
```



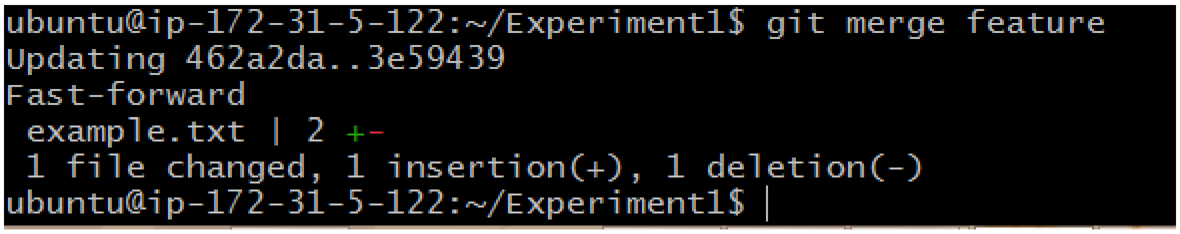
```
ubuntu@ip-172-31-5-122:~/Experiment1$ git checkout master
Switched to branch 'master'
ubuntu@ip-172-31-5-122:~/Experiment1$ git branch
  feature
* master
ubuntu@ip-172-31-5-122:~/Experiment1$ |
cat example.txt
```



```
ubuntu@ip-172-31-5-122:~/Experiment1$ cat example.txt
Hi Good Morning
ubuntu@ip-172-31-5-122:~/Experiment1$ |
```

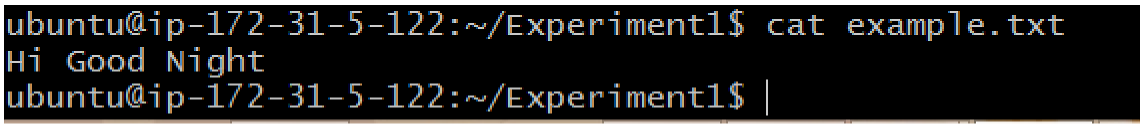
● Merge the changes from the "`feature`" branch into the "`master`" branch:

```
git merge feature
```



```
ubuntu@ip-172-31-5-122:~/Experiment1$ git merge feature
Updating 462a2da..3e59439
Fast-forward
 example.txt | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
ubuntu@ip-172-31-5-122:~/Experiment1$ |
```

```
cat example.txt
```

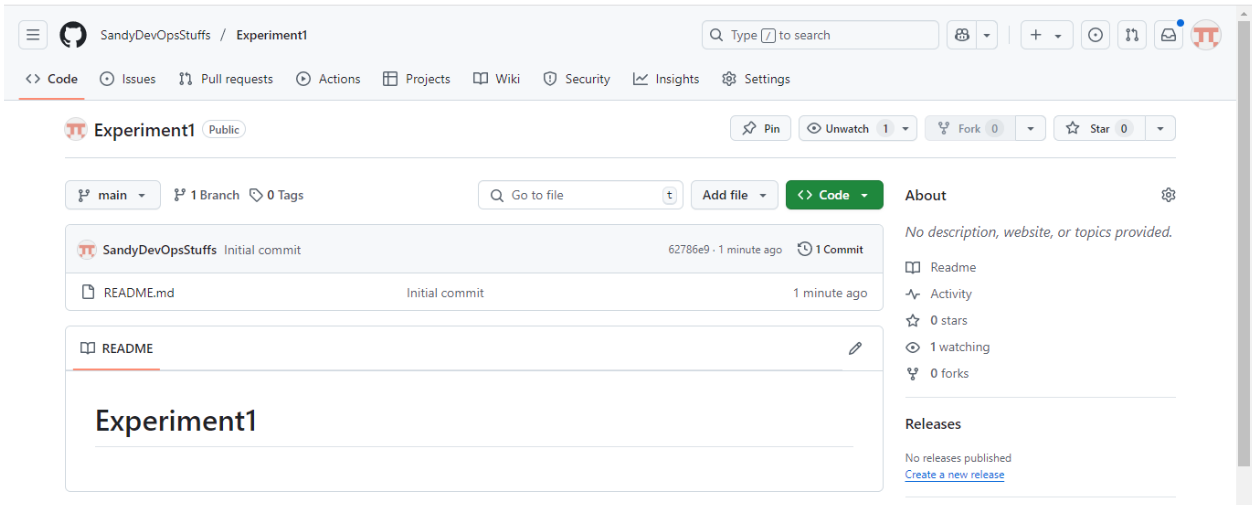


```
ubuntu@ip-172-31-5-122:~/Experiment1$ cat example.txt
Hi Good Night
ubuntu@ip-172-31-5-122:~/Experiment1$ |
```

Step 5: Collaborating with Remote Repositories

● Create an account on a Git hosting service like GitHub (https://github.com/).

● Create a new repository on GitHub.



SandyDevOpsStuffs / Experiment1

Experiment1 Public

main · 1 Branch · 0 Tags

SandyDevOpsStuffs Initial commit 62786e9 · 1 minute ago 1 Commit

README.md Initial commit 1 minute ago

README

Experiment1

About

No description, website, or topics provided.

Readme · Activity · 0 stars · 1 watching · 0 forks

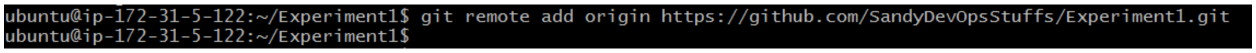
Releases

No releases published

Create a new release

● Link your local repository to the remote repository:

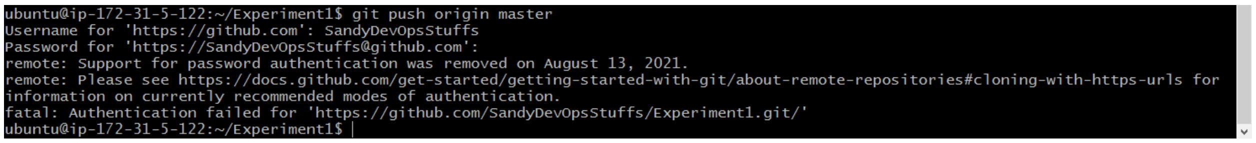
```
git remote add origin <repository_url>
```



```
ubuntu@ip-172-31-5-122:~/Experiment1$ git remote add origin https://github.com/SandyDevOpsStuffs/Experiment1.git
ubuntu@ip-172-31-5-122:~/Experiment1$
```

● Push your local commits to the remote repository:

```
git push origin master
```

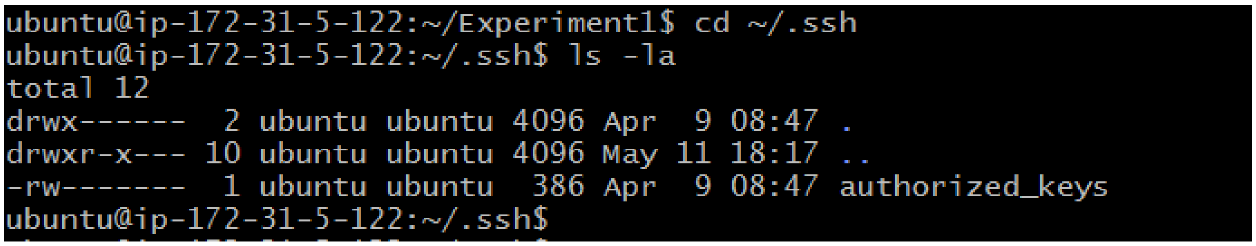


```
ubuntu@ip-172-31-5-122:~/Experiment1$ git push origin master
Username for 'https://github.com': SandyDevOpsStuffs
Password for 'https://SandyDevOpsStuffs@github.com':
remote: Support for password authentication was removed on August 13, 2021.
remote: Please see https://docs.github.com/get-started/getting-started-with-git/about-remote-repositories#cloning-with-https-urls for
information on currently recommended modes of authentication.
fatal: Authentication failed for 'https://github.com/SandyDevOpsStuffs/Experiment1.git/'
ubuntu@ip-172-31-5-122:~/Experiment1$ |
```

Troubleshooting Steps:

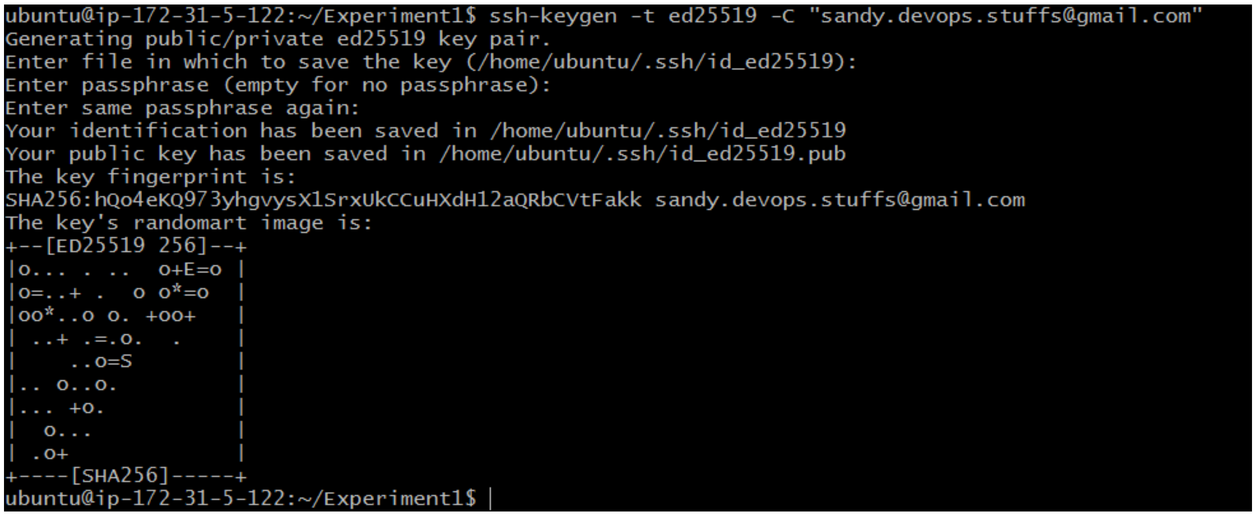
● Generate an SSH Key:

```
cd ~/.ssh

ls -la
```



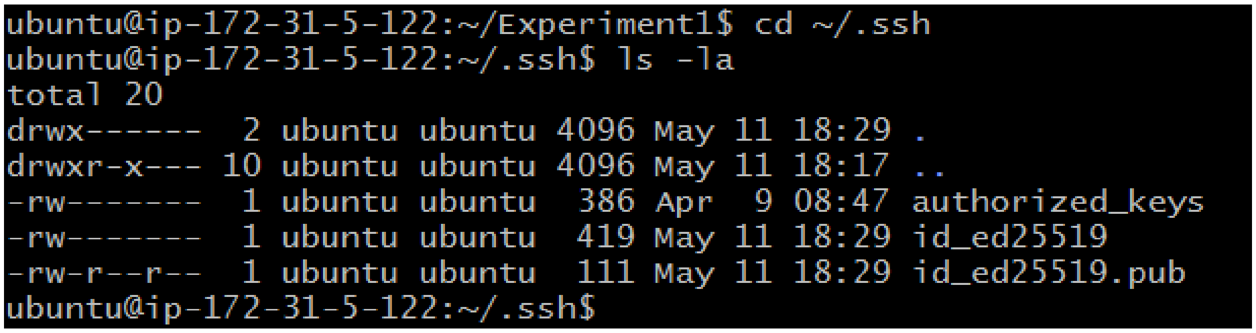
```
ubuntu@ip-172-31-5-122:~/Experiment1$ cd ~/.ssh
ubuntu@ip-172-31-5-122:~/.ssh$ ls -la
total 12
drwx------  2 ubuntu ubuntu 4096 Apr  9 08:47 .
drwxr-x--- 10 ubuntu ubuntu 4096 May 11 18:17 ..
-rw-------  1 ubuntu ubuntu  386 Apr  9 08:47 authorized_keys
ubuntu@ip-172-31-5-122:~/.ssh$
```

```
ssh-keygen -t ed25519 -C "sandy.devops.stuffs@gmail.com"
```



```
ubuntu@ip-172-31-5-122:~/Experiment1$ ssh-keygen -t ed25519 -C "sandy.devops.stuffs@gmail.com"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/ubuntu/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ubuntu/.ssh/id_ed25519
Your public key has been saved in /home/ubuntu/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:hQo4eKQ973yhgvysX1SrxUkCCuHXdH12aQRbCVtFakk sandy.devops.stuffs@gmail.com
The key's randomart image is:
+--[ED25519 256]--+
|o... . ..  o+E=o |
|o=..+ .  o o*=o  |
|oo*..o o. +oo+   |
| ..+ .=.o.  .    |
|    ..o=S        |
|.. o..o.         |
|... +o.          |
|  o...           |
| .o+             |
+----[SHA256]-----+
ubuntu@ip-172-31-5-122:~/Experiment1$ |
```

```
cd ~/.ssh

ls -la
```


```
ubuntu@ip-172-31-5-122:~/Experiment1$ cd ~/.ssh
ubuntu@ip-172-31-5-122:~/.ssh$ ls -la
total 20
drwx------  2 ubuntu ubuntu 4096 May 11 18:29 .
drwxr-x--- 10 ubuntu ubuntu 4096 May 11 18:17 ..
-rw-------  1 ubuntu ubuntu  386 Apr  9 08:47 authorized_keys
-rw-------  1 ubuntu ubuntu  419 May 11 18:29 id_ed25519
-rw-r--r--  1 ubuntu ubuntu  111 May 11 18:29 id_ed25519.pub
ubuntu@ip-172-31-5-122:~/.ssh$
```

```
cat ~/.ssh/id_ed25519.pub
```

Copy the entire output.

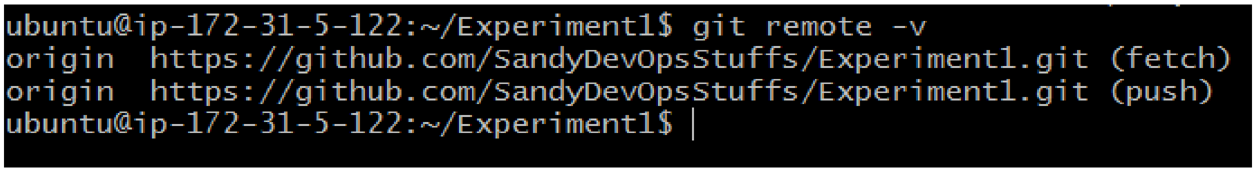
Then:

1. Go to **GitHub → Settings → SSH and GPG keys**
2. Click **"New SSH key"**
3. Paste the public key
4. Give it a title like "Ubuntu_DevBox" and click **Add SSH key**

- Change Your Git Remote from HTTPS to SSH

Check current remotes:

```
git remote -v
```

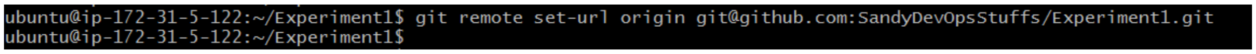

```
ubuntu@ip-172-31-5-122:~/Experiment1$ git remote -v
origin  https://github.com/SandyDevOpsStuffs/Experiment1.git (fetch)
origin  https://github.com/SandyDevOpsStuffs/Experiment1.git (push)
ubuntu@ip-172-31-5-122:~/Experiment1$ |
```

Update the origin URL to SSH:

```
git remote set-url origin
git@github.com:YourUsername/YourRepo.git
```

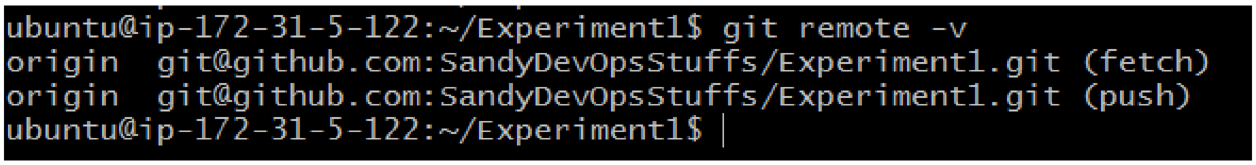
Example:

```
git remote set-url origin
git@github.com:SandyDevOpsStuffs/Experiment1.git
```


```
ubuntu@ip-172-31-5-122:~/Experiment1$ git remote set-url origin git@github.com:SandyDevOpsStuffs/Experiment1.git
ubuntu@ip-172-31-5-122:~/Experiment1$
```

Verify:

```
git remote -v
```



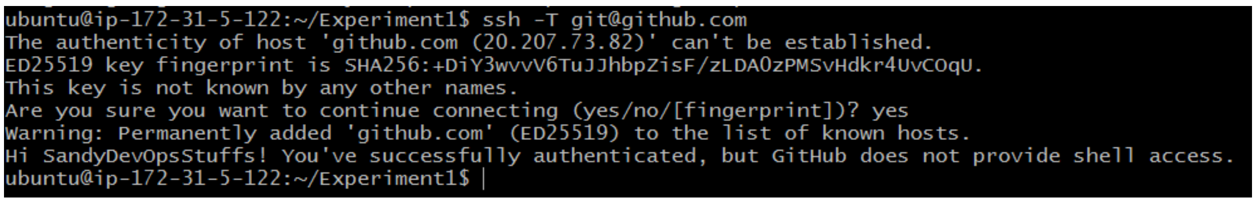
```
ubuntu@ip-172-31-5-122:~/Experiment1$ git remote -v
origin  git@github.com:SandyDevOpsStuffs/Experiment1.git (fetch)
origin  git@github.com:SandyDevOpsStuffs/Experiment1.git (push)
ubuntu@ip-172-31-5-122:~/Experiment1$ |
```

Test SSH Connection:

You should see:

Hi YourUsername! You've successfully authenticated, but GitHub does not provide shell access.

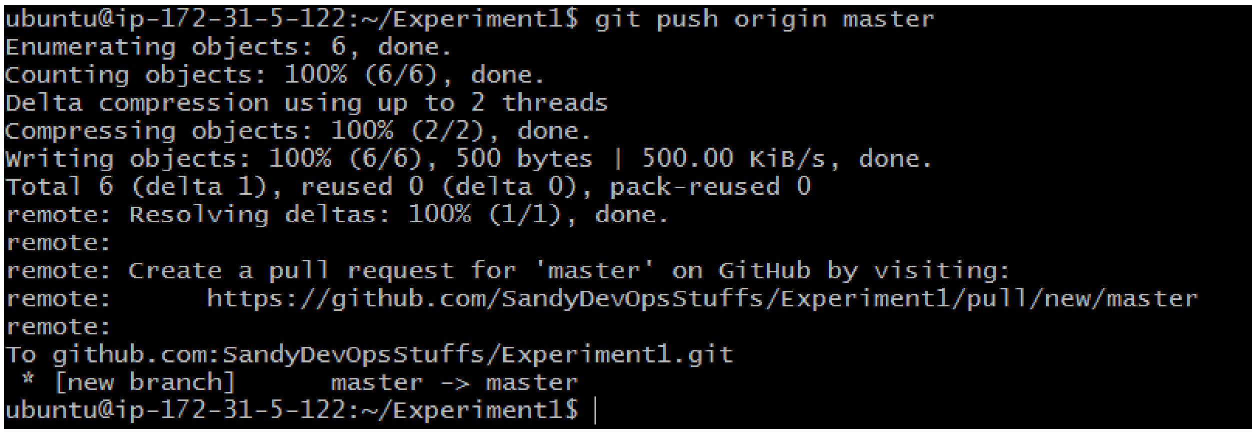
```
ssh -T git@github.com
```



```
ubuntu@ip-172-31-5-122:~/Experiment1$ ssh -T git@github.com
The authenticity of host 'github.com (20.207.73.82)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvCOqU.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
Hi SandyDevOpsStuffs! You've successfully authenticated, but GitHub does not provide shell access.
ubuntu@ip-172-31-5-122:~/Experiment1$ |
```

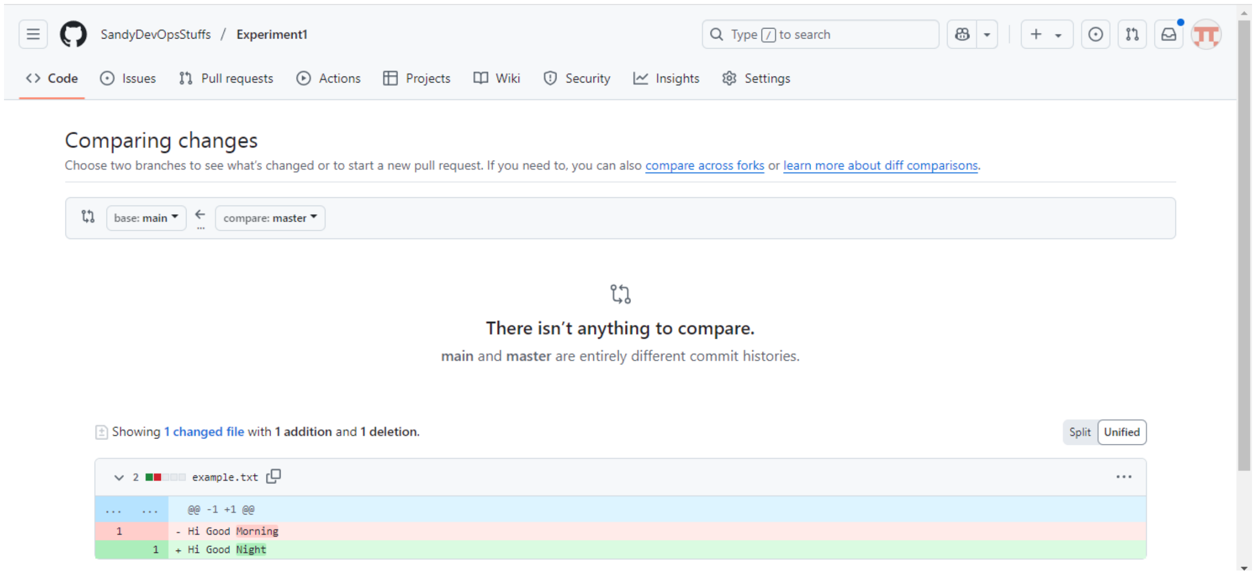
● Push your local commits to the remote repository over SSH:

```
git push origin master
```



```
ubuntu@ip-172-31-5-122:~/Experiment1$ git push origin master
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (6/6), 500 bytes | 500.00 KiB/s, done.
Total 6 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:      https://github.com/SandyDevOpsStuffs/Experiment1/pull/new/master
remote:
To github.com:SandyDevOpsStuffs/Experiment1.git
 * [new branch]      master -> master
ubuntu@ip-172-31-5-122:~/Experiment1$ |
```

Go to GitHub repo and refresh the page to see the file `example.txt` in the remote repo.



Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also compare across forks or learn more about diff comparisons.

base: main ← compare: master

There isn't anything to compare.

main and **master** are entirely different commit histories.

Showing **1 changed file** with **1 addition** and **1 deletion**.

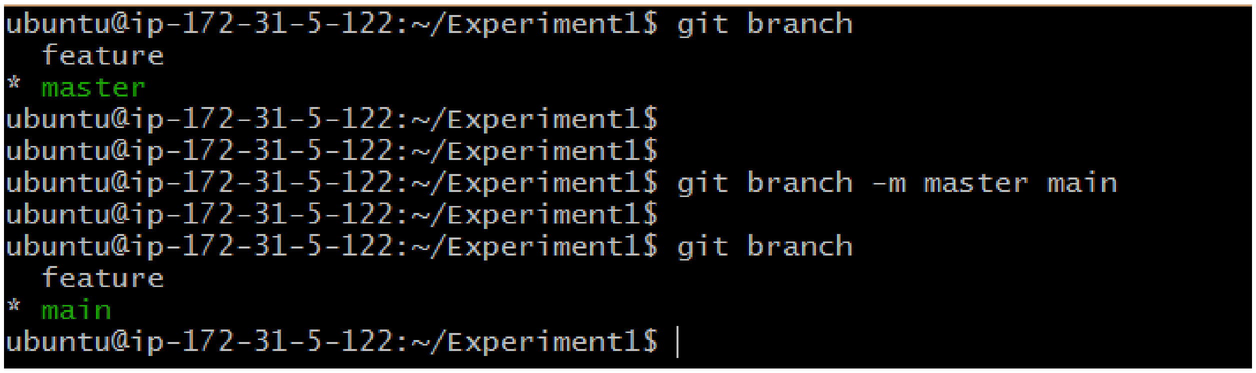
```
2 example.txt
@@ -1 +1 @@
1   - Hi Good Morning
  1 + Hi Good Night
```

This is because we had first created a local repo with default branch `master`, but GitHub uses default branch as `main`. So let us rename the local branch from `master` to `main`.

```
git branch
```

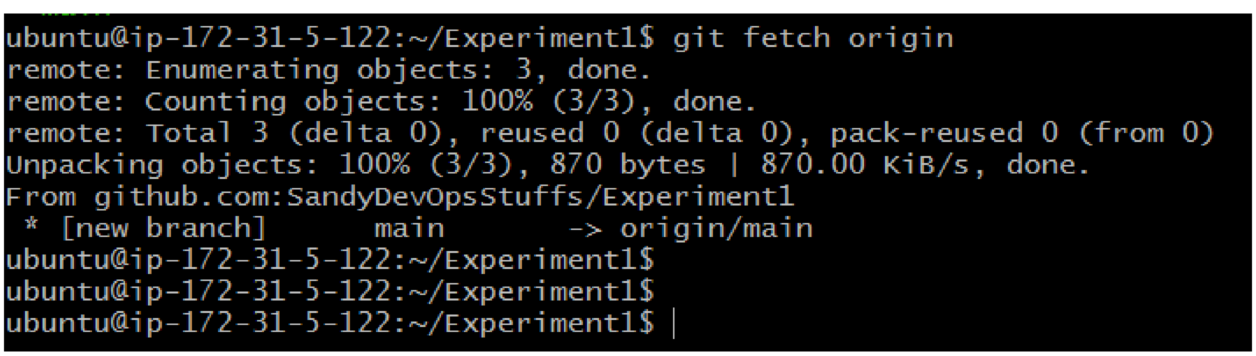
```
git branch -m master main
```

```
git branch
```



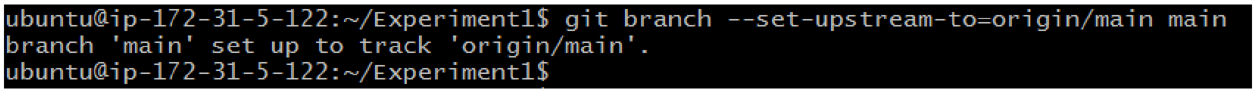
```
ubuntu@ip-172-31-5-122:~/Experiment1$ git branch
  feature
* master
ubuntu@ip-172-31-5-122:~/Experiment1$
ubuntu@ip-172-31-5-122:~/Experiment1$
ubuntu@ip-172-31-5-122:~/Experiment1$ git branch -m master main
ubuntu@ip-172-31-5-122:~/Experiment1$
ubuntu@ip-172-31-5-122:~/Experiment1$ git branch
  feature
* main
ubuntu@ip-172-31-5-122:~/Experiment1$ |
```

```
git fetch origin
```



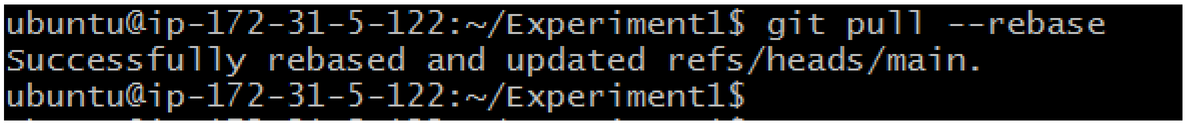
```
ubuntu@ip-172-31-5-122:~/Experiment1$ git fetch origin
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 870 bytes | 870.00 KiB/s, done.
From github.com:SandyDevOpsStuffs/Experiment1
 * [new branch]      main       -> origin/main
ubuntu@ip-172-31-5-122:~/Experiment1$
ubuntu@ip-172-31-5-122:~/Experiment1$
ubuntu@ip-172-31-5-122:~/Experiment1$ |
```

`git branch --set-upstream-to=origin/main main`



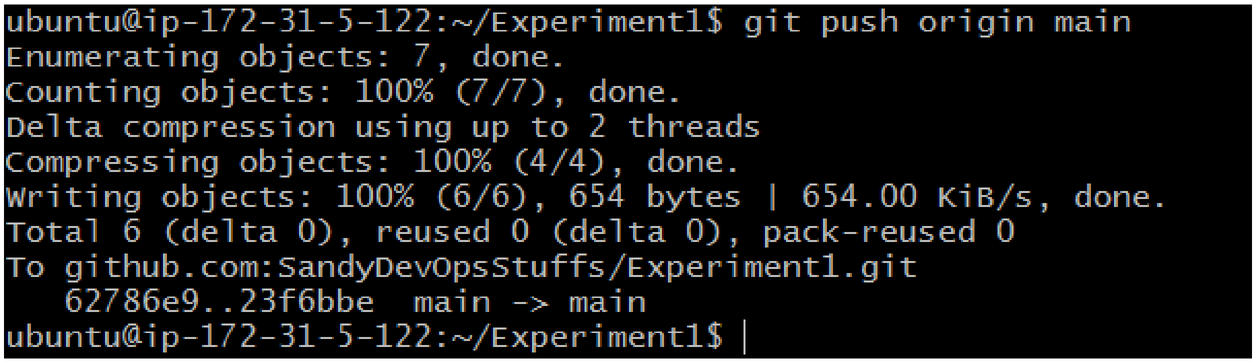
```
ubuntu@ip-172-31-5-122:~/Experiment1$ git branch --set-upstream-to=origin/main main
branch 'main' set up to track 'origin/main'.
ubuntu@ip-172-31-5-122:~/Experiment1$
```

`git pull –rebase`



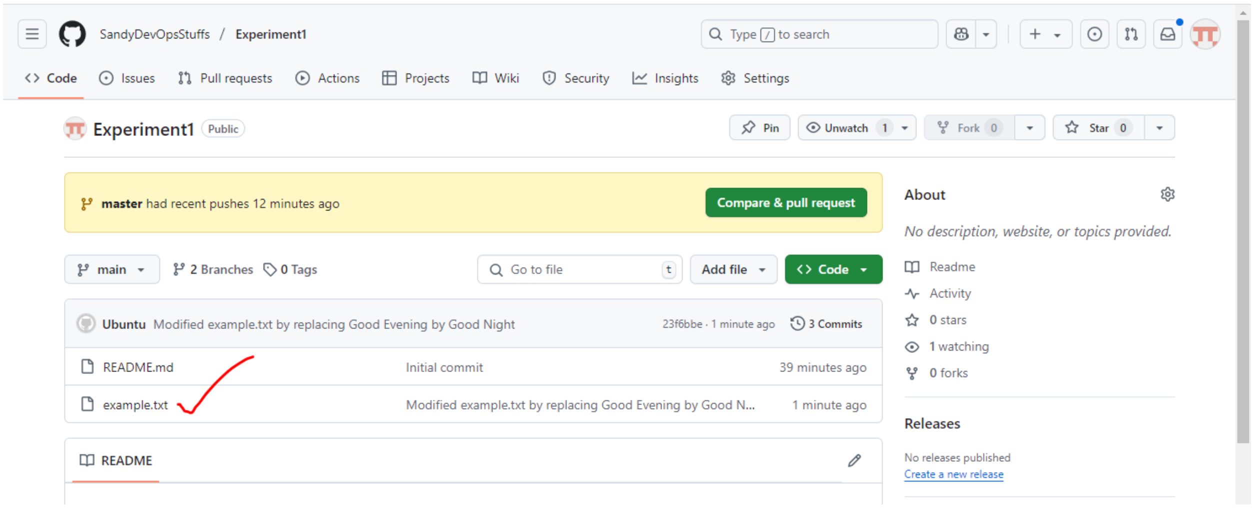
```
ubuntu@ip-172-31-5-122:~/Experiment1$ git pull --rebase
Successfully rebased and updated refs/heads/main.
ubuntu@ip-172-31-5-122:~/Experiment1$
```

`git push origin main`



```
ubuntu@ip-172-31-5-122:~/Experiment1$ git push origin main
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 2 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 654 bytes | 654.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:SandyDevOpsStuffs/Experiment1.git
   62786e9..23f6bbe  main -> main
ubuntu@ip-172-31-5-122:~/Experiment1$ |
```

Now go to GitHub repo and refresh the page to see the file `example.txt` in the remote repo.

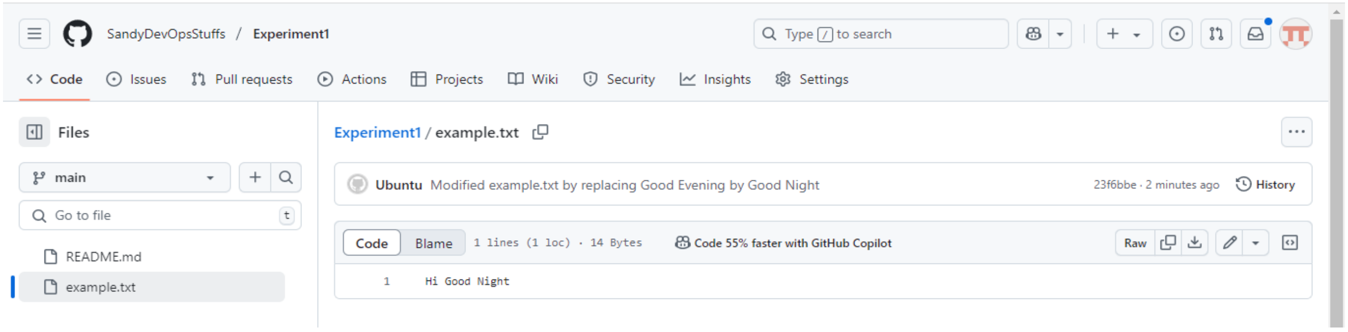


master had recent pushes 12 minutes ago — Compare & pull request

No description, website, or topics provided.

No releases published
Create a new release

Ubuntu Modified example.txt by replacing Good Evening by Good Night	23f6bbe · 1 minute ago	3 Commits
README.md	Initial commit	39 minutes ago
example.txt	Modified example.txt by replacing Good Evening by Good N...	1 minute ago



Ubuntu Modified example.txt by replacing Good Evening by Good Night 23f6bbe · 2 minutes ago

1 lines (1 loc) · 14 Bytes Code 55% faster with GitHub Copilot

```
Hi Good Night
```

Conclusion:

Through this experiment, participants gained a foundational understanding of Git's essential commands and concepts. They learned how to set up a Git repository, manage changes, explore commit history, create and merge branches, and collaborate with remote repositories. This knowledge equips them with the skills needed to effectively use Git for version control and collaborative software development. To avoid the issues we faced in mismatch of default branch in local repo and remote repo and to avoid the issues related to HTTP and SSH authentication we can use the below steps:

Step 1: Generate an SSH key pair on local machine as shown above and add the public key to GitHub.

Step 2: Initialize Git (if not already done)

```
git init
```

Step 3: Add and commit your files

```
git add .

git commit -m "Initial commit"
```

Step 4: Rename master to main (Git 2.28+ lets you set this by default too)

```
git branch -m main
```

Step 5: Add your GitHub repo as remote (using SSH if configured)

```
git remote add origin git@github.com:YourUsername/YourRepo.git
```

Step 6: Push to GitHub and set upstream

```
git push -u origin main
```

Exercise:

1. Explain what version control is and why it is important in software development. Provide examples of version control systems other than Git.

2. Describe the typical workflow when working with Git, including initializing a repository, committing changes, and pushing to a remote repository. Use a real-world example to illustrate the process.

3. Discuss the purpose of branching in Git and how it facilitates collaborative development. Explain the steps involved in creating a new branch, making changes, and merging it back into the `main` branch.

4. What are merge conflicts in Git, and how can they be resolved? Provide a step-by-step guide on how to handle a merge conflict.

5. Explain the concept of remote repositories in Git and how they enable collaboration among team members. Describe the differences between cloning a repository and adding a remote.

6. Discuss different branching strategies, such as feature branching and Gitflow. Explain the advantages and use cases of each strategy.

7. Describe various Git commands and techniques for undoing changes, such as reverting commits, resetting branches, and discarding uncommitted changes.

8. What are Git hooks, and how can they be used to automate tasks and enforce coding standards in a Git repository? Provide examples of practical use cases for Git hooks.

9. List and explain five best practices for effective and efficient Git usage in a collaborative software development environment.

10. Discuss security considerations in Git, including how to protect sensitive information like passwords and API keys. Explain the concept of Git signing and why it's important.