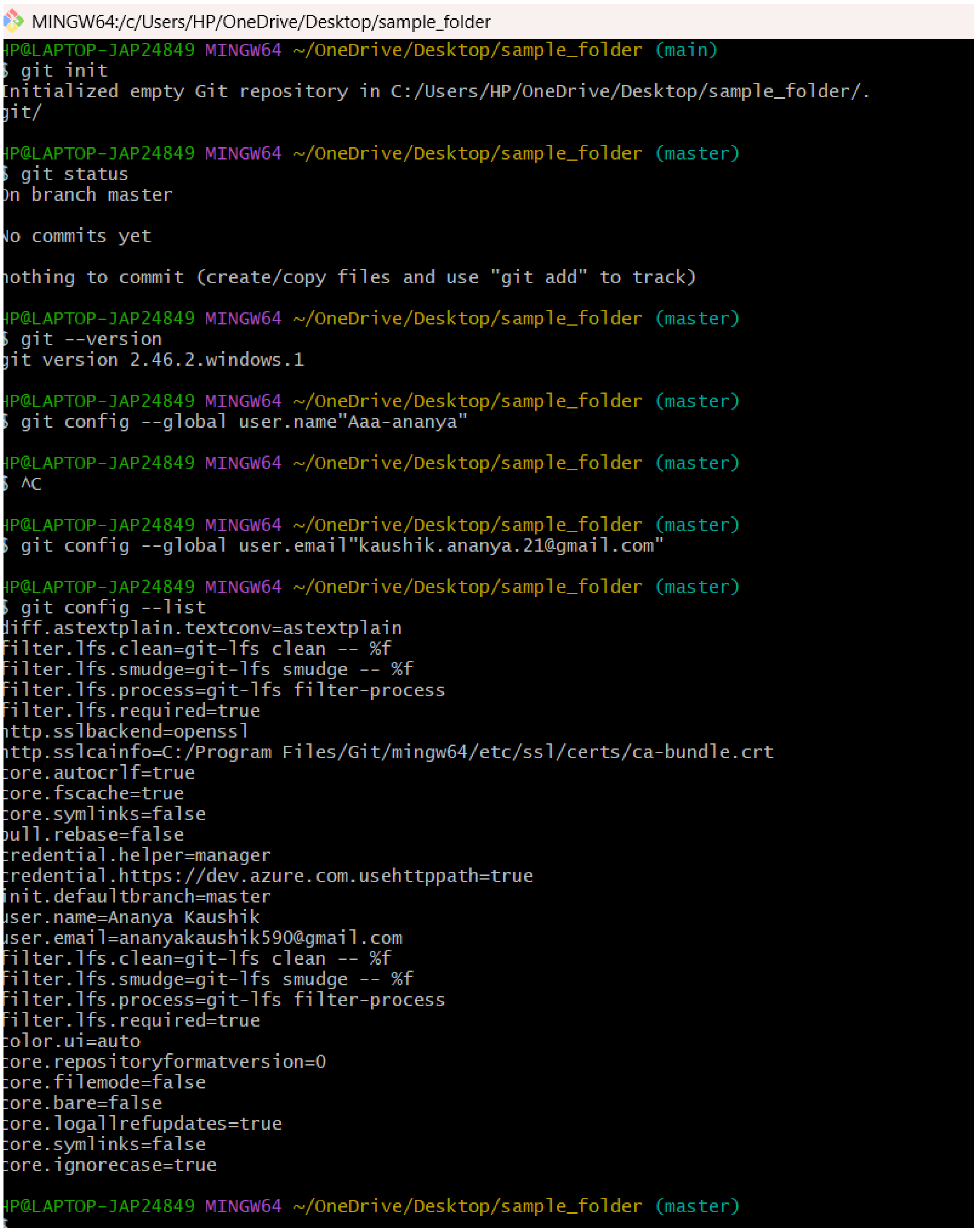
GIT Assignment Questions

1. Setting Up Git:

- **Q1:** Install Git on your system and configure your name and email using the following commands:
    - `git config --global user.name "Your Name"`
    - `git config --global user.email "your.email@example.com"`
- **Q2:** How would you verify that Git has been installed and properly configured? Provide the command and the expected output.
- **Q3:** Initialize a new Git repository in an empty directory on your computer using `git init`.

  Answer 1,2,3:



MINGW64:/c/Users/HP/OneDrive/Desktop/sample_folder

```
HP@LAPTOP-JAP24849 MINGW64 ~/OneDrive/Desktop/sample_folder (main)
$ git init
Initialized empty Git repository in C:/Users/HP/OneDrive/Desktop/sample_folder/.
git/

HP@LAPTOP-JAP24849 MINGW64 ~/OneDrive/Desktop/sample_folder (master)
$ git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)

HP@LAPTOP-JAP24849 MINGW64 ~/OneDrive/Desktop/sample_folder (master)
$ git --version
git version 2.46.2.windows.1

HP@LAPTOP-JAP24849 MINGW64 ~/OneDrive/Desktop/sample_folder (master)
$ git config --global user.name"Aaa-ananya"

HP@LAPTOP-JAP24849 MINGW64 ~/OneDrive/Desktop/sample_folder (master)
$ ^C

HP@LAPTOP-JAP24849 MINGW64 ~/OneDrive/Desktop/sample_folder (master)
$ git config --global user.email"kaushik.ananya.21@gmail.com"

HP@LAPTOP-JAP24849 MINGW64 ~/OneDrive/Desktop/sample_folder (master)
$ git config --list
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=C:/Program Files/Git/mingw64/etc/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=master
user.name=Ananya Kaushik
user.email=ananyakaushik590@gmail.com
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
color.ui=auto
core.repositoryformatversion=0
core.filemode=false
core.bare=false
core.logallrefupdates=true
core.symlinks=false
core.ignorecase=true

HP@LAPTOP-JAP24849 MINGW64 ~/OneDrive/Desktop/sample_folder (master)
$
```

2. Basic Git Operations:

- **Q4:** Create a new text file named `hello.txt` in your repository. Add some content to it. Then, stage the file for commit using the `git add` command.
- **Q5:** Commit the changes you made to the `hello.txt` file with a meaningful commit message. Provide the Git command to commit and the expected output.
- **Q6:** After committing your changes, use the `git status` command to check the state of your repository. Explain the output.

  Explanation of git status Output:

  Scenario 1: No Changes After Commit

  If you haven't made any changes after committing, the output will look like:

  pgsql

  CopyEdit

  ```
  On branch main
  nothing to commit, working tree clean
  ```

  Meaning:

- You are currently on the main branch.
- All changes have been committed.
- There are no modified, staged, or untracked files.
- The working directory is in sync with the last commit.

  Scenario 2: You Modified Some Files After Commit

  perl

  CopyEdit

  ```
  On branch main
  Changes not staged for commit:
    (use "git add <file>..." to update what will be committed)
          modified:   file1.txt

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

  Meaning:

- You've edited file1.txt but haven't staged it yet.
- The file is tracked by Git but changes are still in the working directory.

  Scenario 3: New Files Added but Not Staged

  vbnet

  CopyEdit

  ```
  On branch main
  Untracked files:
    (use "git add <file>..." to include in what will be committed)
  ```

```
    newfile.py

nothing added to commit but untracked files present
```

Meaning:

- newfile.py is a newly created file not yet tracked by Git.
- You need to use git add newfile.py to stage it.

- **Q7:** How can you view the commit history of a repository? Use the `git log` command and describe what information it provides.

You can view the commit history of a Git repository using the following Git commands:

1. Basic Commit History

```
git log
```

- Shows a list of commits in reverse chronological order.
- Each entry includes:
    - Commit hash
    - Author
    - Date
    - Commit message

2. Compact One-line Format

```
git log --oneline
```

- Displays each commit on a single line.
- Useful for a quick overview:

```
a3c1b2d Fix: corrected login validation
9fcb3c9 Add: login page UI
2ac4df3 Init: initial project structure
```

3. Graph View with Branches

```
git log --oneline --graph --all
```

- Shows a graphical representation of branches and merges.
- Helpful for visualizing the branching structure.

4. View History for a Specific File

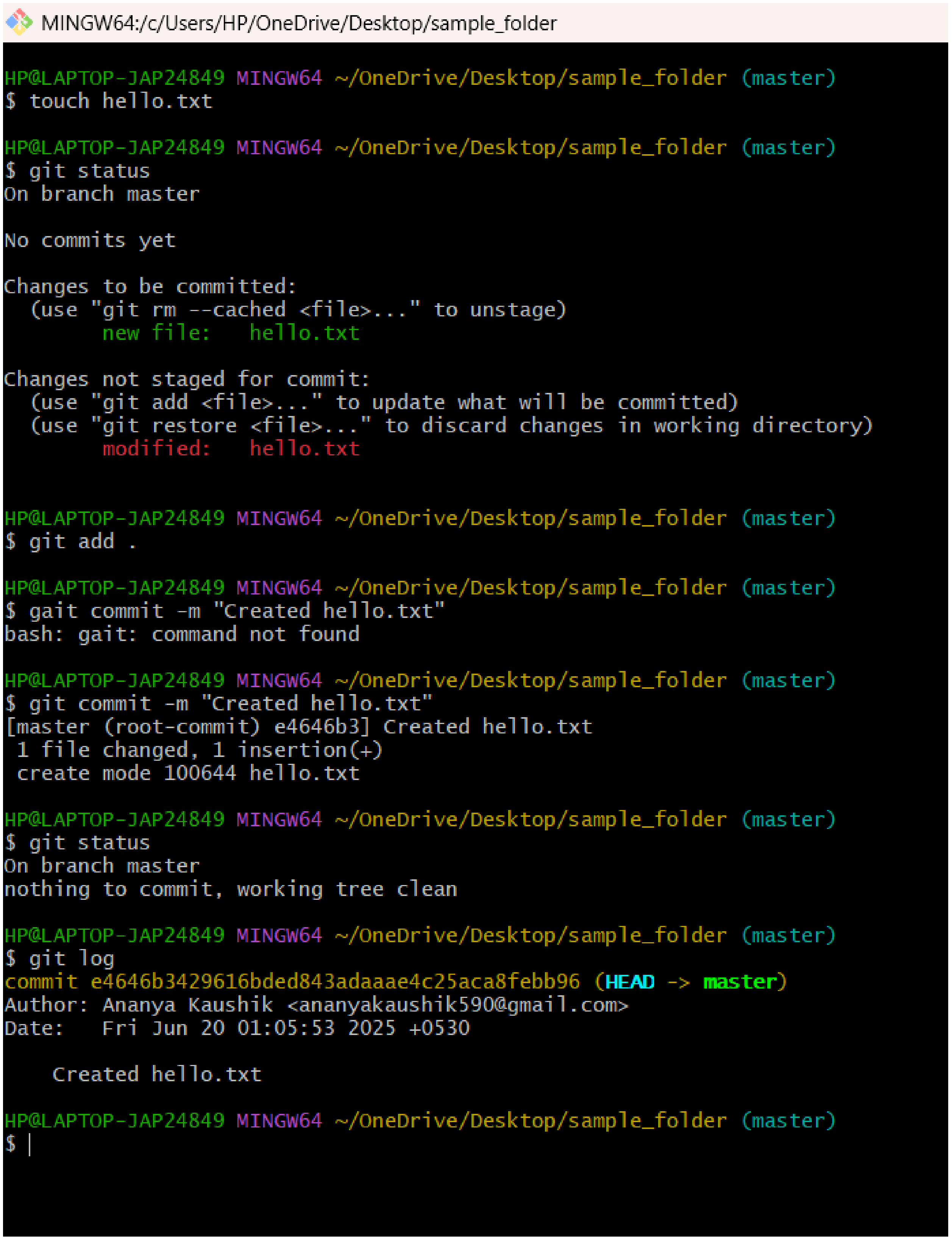
```
git log <filename>
```

- Shows commit history that affected a specific file.

5. Show Detailed Difference with Each Commit

```
git log -p
```

- Displays the patch (diff) introduced in each commit.



MINGW64:/c/Users/HP/OneDrive/Desktop/sample_folder

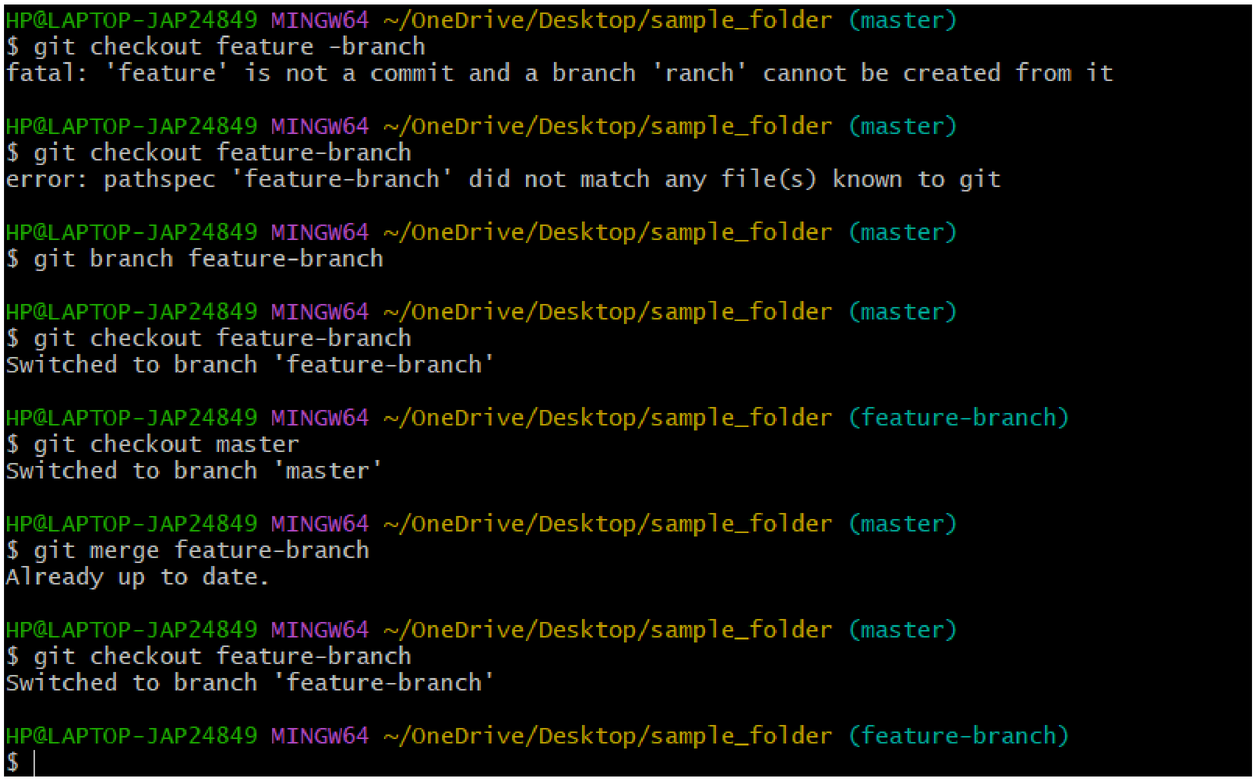
```
HP@LAPTOP-JAP24849 MINGW64 ~/OneDrive/Desktop/sample_folder (master)
$ touch hello.txt

HP@LAPTOP-JAP24849 MINGW64 ~/OneDrive/Desktop/sample_folder (master)
$ git status
On branch master

No commits yet

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

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.txt


HP@LAPTOP-JAP24849 MINGW64 ~/OneDrive/Desktop/sample_folder (master)
$ git add .

HP@LAPTOP-JAP24849 MINGW64 ~/OneDrive/Desktop/sample_folder (master)
$ gait commit -m "Created hello.txt"
bash: gait: command not found

HP@LAPTOP-JAP24849 MINGW64 ~/OneDrive/Desktop/sample_folder (master)
$ git commit -m "Created hello.txt"
[master (root-commit) e4646b3] Created hello.txt
 1 file changed, 1 insertion(+)
 create mode 100644 hello.txt

HP@LAPTOP-JAP24849 MINGW64 ~/OneDrive/Desktop/sample_folder (master)
$ git status
On branch master
nothing to commit, working tree clean

HP@LAPTOP-JAP24849 MINGW64 ~/OneDrive/Desktop/sample_folder (master)
$ git log
commit e4646b3429616bded843adaaae4c25aca8febb96 (HEAD -> master)
Author: Ananya Kaushik <ananyakaushik590@gmail.com>
Date:   Fri Jun 20 01:05:53 2025 +0530

    Created hello.txt

HP@LAPTOP-JAP24849 MINGW64 ~/OneDrive/Desktop/sample_folder (master)
$ |
```

3. Branching and Merging:

- **Q8:** What is the purpose of branching in Git? How do branches help in software development?
  Purpose of Branching in Git:
  Branching in Git allows developers to diverge from the main line of development and work on different tasks (like features, bug fixes, or experiments) in isolation. It creates a separate environment within the same repository where you can make changes without affecting the main codebase.
  How Branches Help in Software Development:

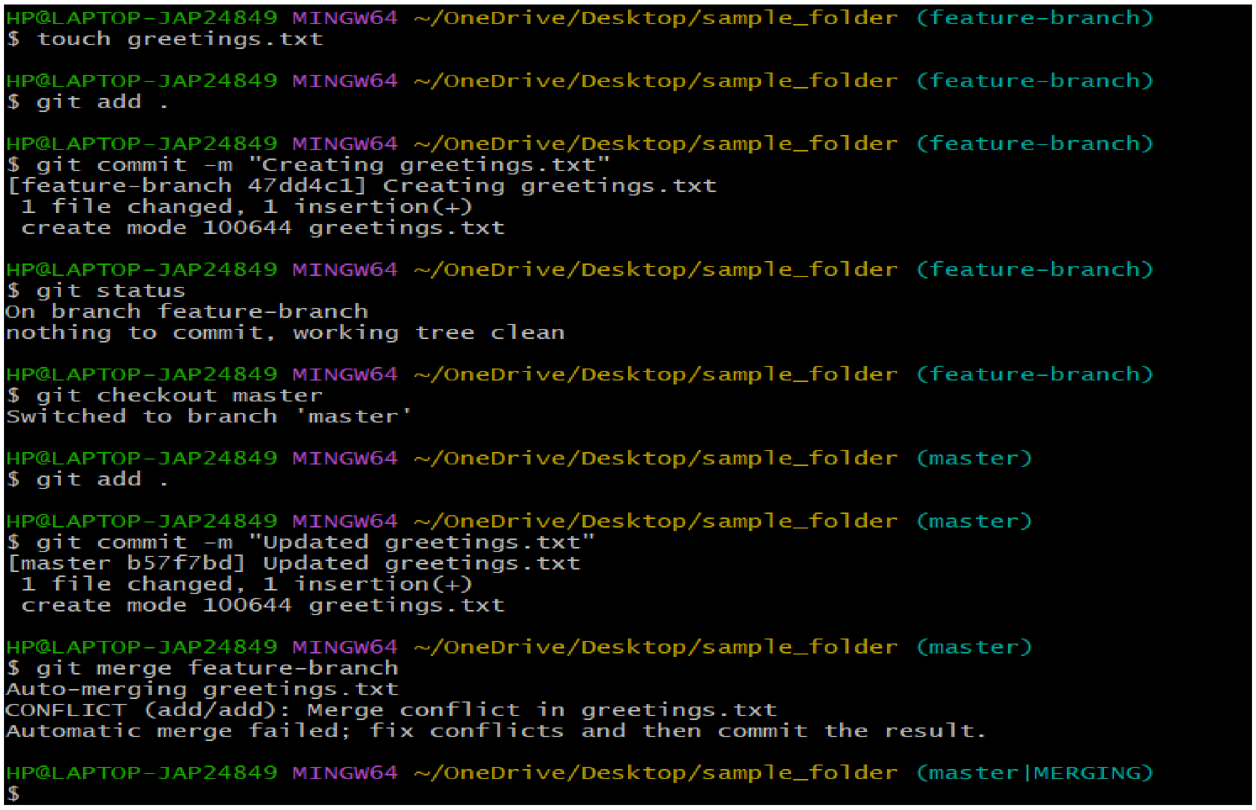
1. **Parallel Development:**
   Developers can work on multiple features or fixes simultaneously using different branches.
2. **Code Isolation:**
   Each branch isolates changes, preventing unfinished or buggy code from affecting the main project.
3. **Safe Experimentation:**
   Developers can try new ideas or refactor code in a branch without risk. If it doesn't work, the branch can simply be deleted.
4. **Easier Collaboration:**
   Teams can assign different branches to different team members, improving collaboration and reducing merge conflicts.

- **Q9:** Create a new branch called `feature-branch` and switch to it using the appropriate Git command.
- **Q10:** Create a new file named `feature.txt` on your new branch and commit the changes. Then, switch back to the `main` branch.
- **Q11:** Merge the `feature-branch` into the `main` branch. What command would you use to merge the changes, and what happens if there are no conflicts?



```
HP@LAPTOP-JAP24849 MINGW64 ~/OneDrive/Desktop/sample_folder (master)
$ git checkout feature -branch
fatal: 'feature' is not a commit and a branch 'ranch' cannot be created from it

HP@LAPTOP-JAP24849 MINGW64 ~/OneDrive/Desktop/sample_folder (master)
$ git checkout feature-branch
error: pathspec 'feature-branch' did not match any file(s) known to git

HP@LAPTOP-JAP24849 MINGW64 ~/OneDrive/Desktop/sample_folder (master)
$ git branch feature-branch

HP@LAPTOP-JAP24849 MINGW64 ~/OneDrive/Desktop/sample_folder (master)
$ git checkout feature-branch
Switched to branch 'feature-branch'

HP@LAPTOP-JAP24849 MINGW64 ~/OneDrive/Desktop/sample_folder (feature-branch)
$ git checkout master
Switched to branch 'master'

HP@LAPTOP-JAP24849 MINGW64 ~/OneDrive/Desktop/sample_folder (master)
$ git merge feature-branch
Already up to date.

HP@LAPTOP-JAP24849 MINGW64 ~/OneDrive/Desktop/sample_folder (master)
$ git checkout feature-branch
Switched to branch 'feature-branch'

HP@LAPTOP-JAP24849 MINGW64 ~/OneDrive/Desktop/sample_folder (feature-branch)
$ |
```

- **Q12:** What is a merge conflict? Create a scenario where a merge conflict occurs and explain how you would resolve it.



```
HP@LAPTOP-JAP24849 MINGW64 ~/OneDrive/Desktop/sample_folder (feature-branch)
$ touch greetings.txt

HP@LAPTOP-JAP24849 MINGW64 ~/OneDrive/Desktop/sample_folder (feature-branch)
$ git add .

HP@LAPTOP-JAP24849 MINGW64 ~/OneDrive/Desktop/sample_folder (feature-branch)
$ git commit -m "Creating greetings.txt"
[feature-branch 47dd4c1] Creating greetings.txt
 1 file changed, 1 insertion(+)
 create mode 100644 greetings.txt

HP@LAPTOP-JAP24849 MINGW64 ~/OneDrive/Desktop/sample_folder (feature-branch)
$ git status
On branch feature-branch
nothing to commit, working tree clean

HP@LAPTOP-JAP24849 MINGW64 ~/OneDrive/Desktop/sample_folder (feature-branch)
$ git checkout master
Switched to branch 'master'

HP@LAPTOP-JAP24849 MINGW64 ~/OneDrive/Desktop/sample_folder (master)
$ git add .

HP@LAPTOP-JAP24849 MINGW64 ~/OneDrive/Desktop/sample_folder (master)
$ git commit -m "Updated greetings.txt"
[master b57f7bd] Updated greetings.txt
 1 file changed, 1 insertion(+)
 create mode 100644 greetings.txt

HP@LAPTOP-JAP24849 MINGW64 ~/OneDrive/Desktop/sample_folder (master)
$ git merge feature-branch
Auto-merging greetings.txt
CONFLICT (add/add): Merge conflict in greetings.txt
Automatic merge failed; fix conflicts and then commit the result.

HP@LAPTOP-JAP24849 MINGW64 ~/OneDrive/Desktop/sample_folder (master|MERGING)
$
```

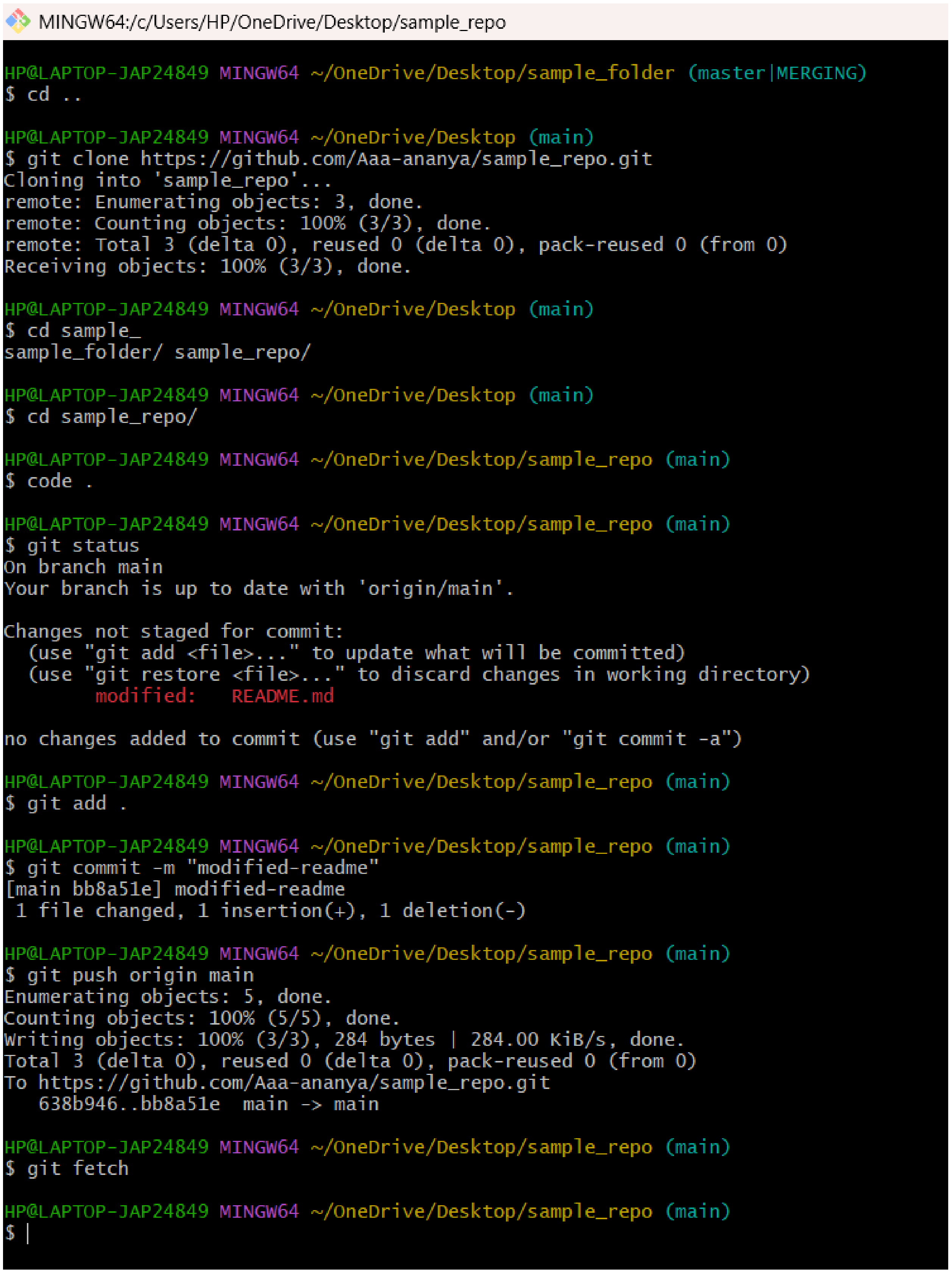
4. Working with Remote Repositories:

- **Q13:** What is a remote repository in Git? How is it different from a local repository?
  A **remote repository** is a Git repository that is hosted on a server (such as GitHub, GitLab, Bitbucket, or a private Git server). It allows multiple developers to collaborate by pushing their local changes and pulling updates made by others.
- It is **not stored on your local machine**.
- Accessed via URL (e.g., HTTPS or SSH).

  Local Repository:
  A **local repository** is the Git repository stored on your own computer. It includes:
- A working directory (your actual project files).
- A .git folder that tracks versions and changes.
- You can make commits, create branches, and view history without internet access

- **Q14:** Clone a remote repository from GitHub to your local machine using the `git clone` command. Provide the URL of a public repository to clone.
- **Q15:** After cloning the repository, make a small change (e.g., edit `README.md`), and commit the changes to your local repository.
- **Q16:** Push your local commits to the remote repository. What Git command is used to push changes to a remote repository? Explain how you would use it.
- **Q17:** Fetch the latest changes from the remote repository using the `git fetch` command. What is the difference between `git fetch` and `git pull`?

Feature	**git fetch**	**git pull**
Fetches remote changes	Yes	Yes
Updates working branch	No	Yes
Safer	Yes	May cause conflicts
Manual merge required	Yes (optional after fetch)	No (automatic merge happens)
Control over merging	Full control	Less control



MINGW64:/c/Users/HP/OneDrive/Desktop/sample_repo

```
HP@LAPTOP-JAP24849 MINGW64 ~/OneDrive/Desktop/sample_folder (master|MERGING)
$ cd ..

HP@LAPTOP-JAP24849 MINGW64 ~/OneDrive/Desktop (main)
$ git clone https://github.com/Aaa-ananya/sample_repo.git
Cloning into 'sample_repo'...
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)
Receiving objects: 100% (3/3), done.

HP@LAPTOP-JAP24849 MINGW64 ~/OneDrive/Desktop (main)
$ cd sample_
sample_folder/ sample_repo/

HP@LAPTOP-JAP24849 MINGW64 ~/OneDrive/Desktop (main)
$ cd sample_repo/

HP@LAPTOP-JAP24849 MINGW64 ~/OneDrive/Desktop/sample_repo (main)
$ code .

HP@LAPTOP-JAP24849 MINGW64 ~/OneDrive/Desktop/sample_repo (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

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:   README.md

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

HP@LAPTOP-JAP24849 MINGW64 ~/OneDrive/Desktop/sample_repo (main)
$ git add .

HP@LAPTOP-JAP24849 MINGW64 ~/OneDrive/Desktop/sample_repo (main)
$ git commit -m "modified-readme"
[main bb8a51e] modified-readme
 1 file changed, 1 insertion(+), 1 deletion(-)

HP@LAPTOP-JAP24849 MINGW64 ~/OneDrive/Desktop/sample_repo (main)
$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 284 bytes | 284.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Aaa-ananya/sample_repo.git
   638b946..bb8a51e  main -> main

HP@LAPTOP-JAP24849 MINGW64 ~/OneDrive/Desktop/sample_repo (main)
$ git fetch

HP@LAPTOP-JAP24849 MINGW64 ~/OneDrive/Desktop/sample_repo (main)
$ |
```

5. Undoing Changes in Git:

- **Q18:** After making several commits, you realize that a commit message needs to be changed. How can you edit the last commit message using Git?
  If you want to **edit the message of the last commit**, use the following command:
  git commit --amend
  Steps:

1. Run:
   git commit --amend
2. Your default editor will open showing the last commit message.
3. Edit the commit message as needed.
4. Save and close the editor.