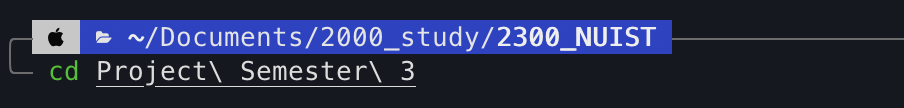
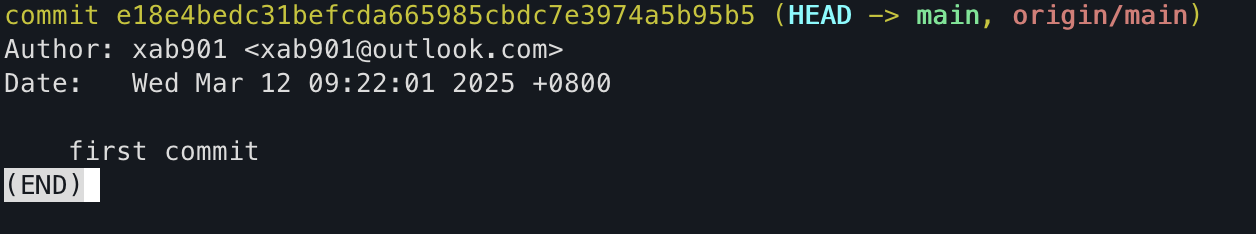
1. Go to the ~/Project-Semsester3 directory in Git Bash You have already initialized a git repo in this directory. Check git log command output in that directory

$ cd ~/Project-Semsester3

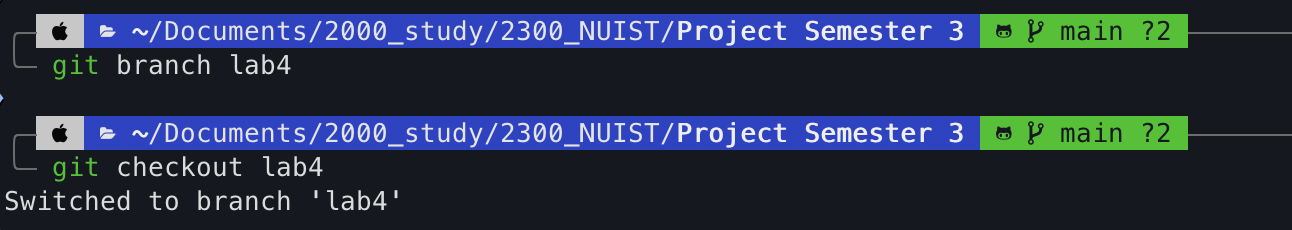
$ git log





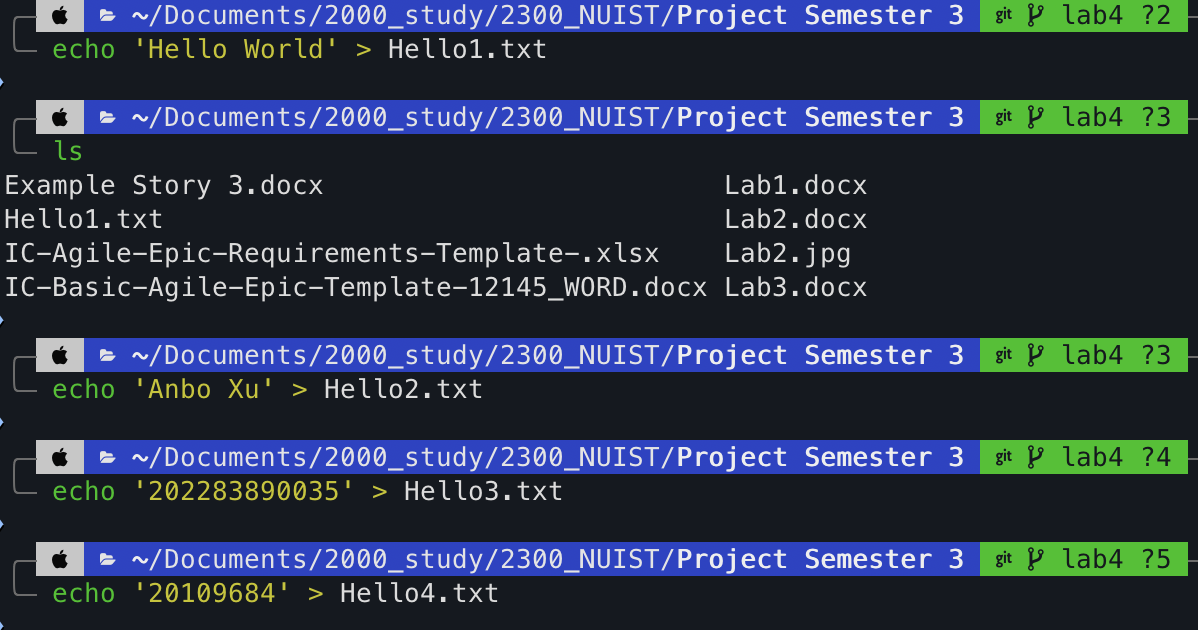
**Comment: This step navigates to the project directory and retrieves the Git commit history. Checking the log is useful for reviewing past commits before making new changes.**

1. Create a new branch named “Lab4”. And make sure to check out the branch



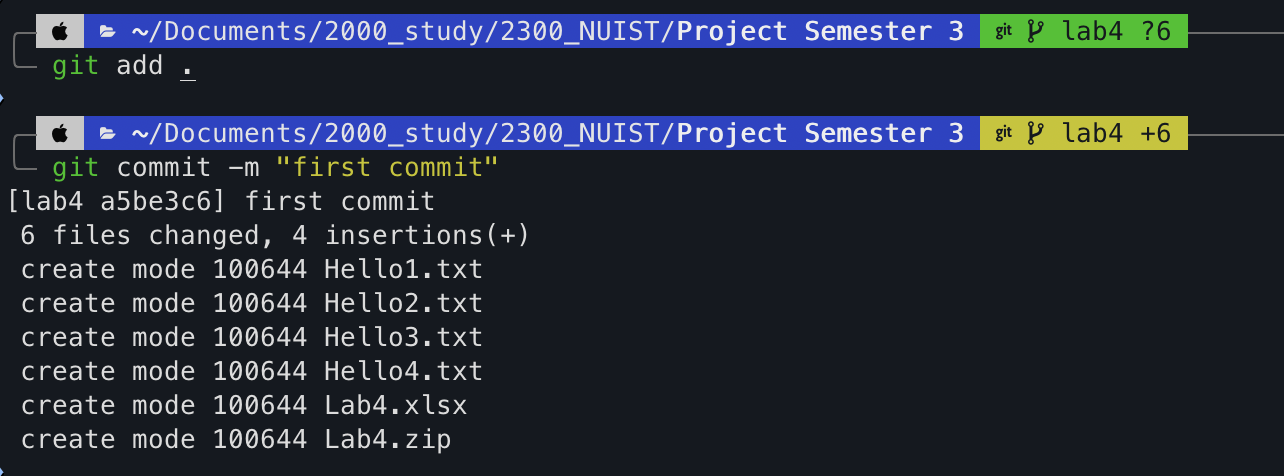
**Comment: Creating a new branch enables independent development, while switching to it ensures that all subsequent changes occur in this branch instead of the main one.**

1. Using echo command, create 4 files: Hello1.txt, Hello2.txt, Hello3.txt, Hello4.txt. Use any text that you wish.



**Comment: The echo command is a quick way to generate text files with predefined content, useful for testing or initializing repository files.**

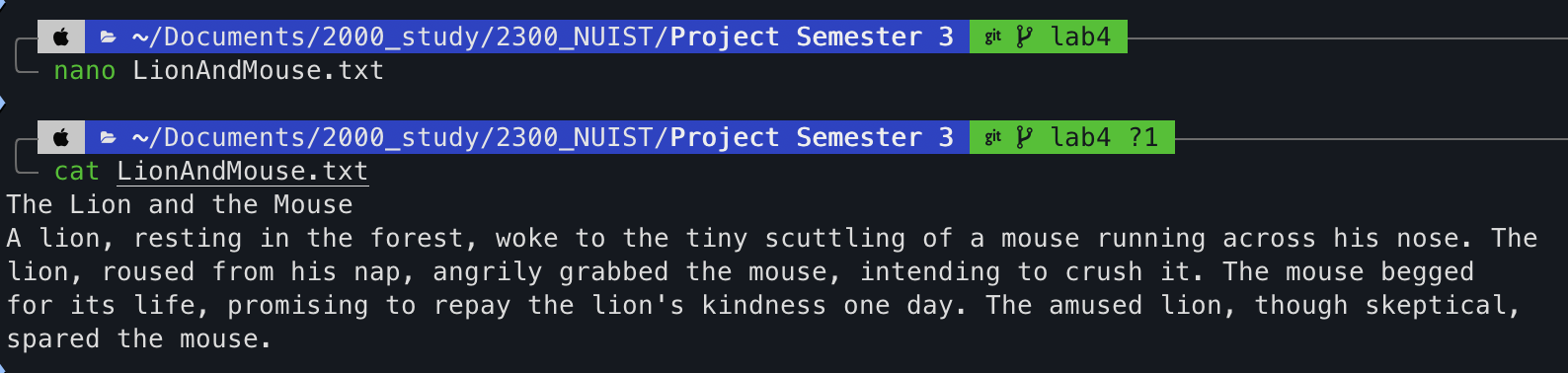
1. Add and commit all files to the repository. Check the log after the commit.



**Comment: Staging and committing files ensures they are saved in the repository. Checking the log verifies that the commit was successful.**

1. Create the following text file “LionAndMouse.txt” using the Nano editor.

The Lion and the Mouse A lion, resting in the forest, woke to the tiny scuttling of a mouse running across his nose. The lion, roused from his nap, angrily grabbed the mouse, intending to crush it. The mouse begged for its life, promising to repay the lion's kindness one day. The amused lion, though skeptical, spared the mouse. $ nano LionAndMouse.txt

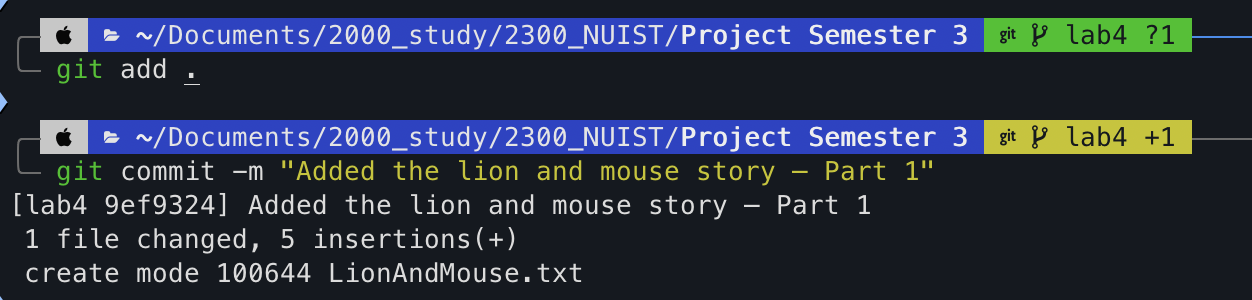


**Comment: Nano is a simple text editor available in Git Bash, making it convenient for creating and modifying files directly in the terminal.**

1. Add and commit this file to the Repository

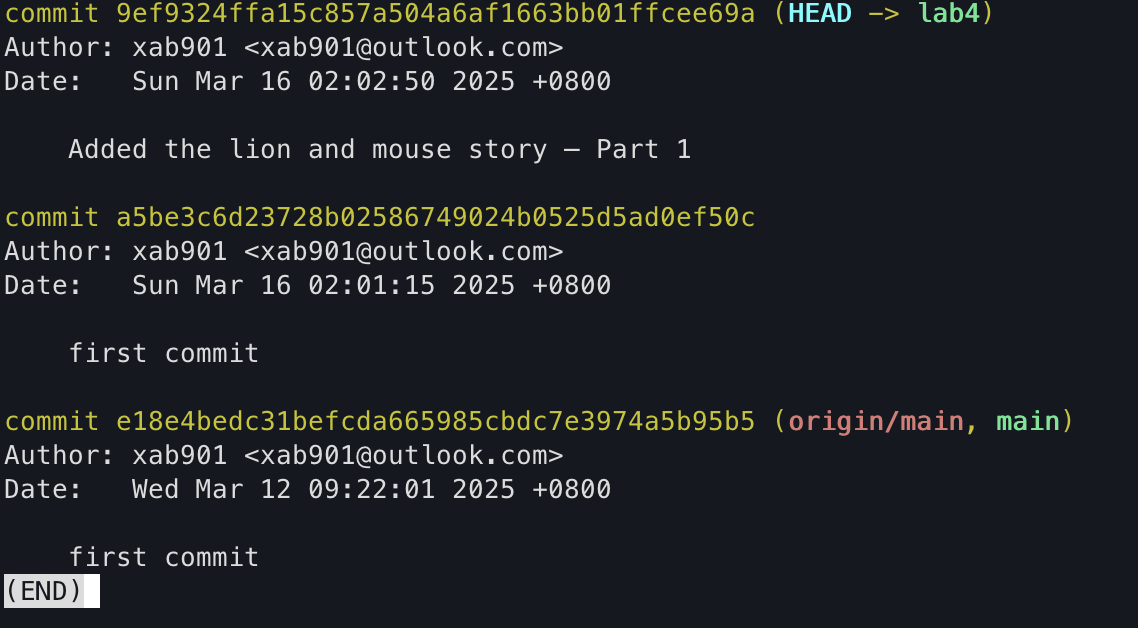
$ git add .

$ git commit -m "Added the lion and mouse story – Part 1”



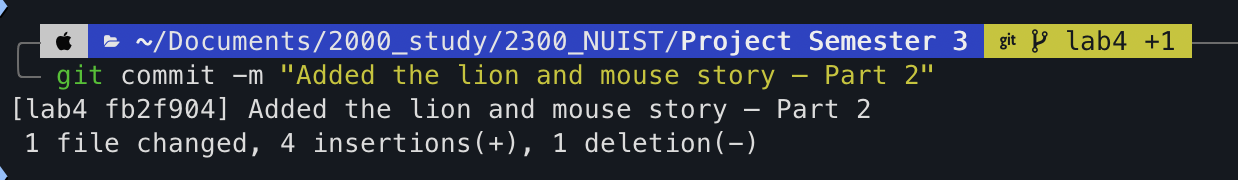
**Comment: This updates the repository with the newly created file. The commit message clarifies the purpose of the addition.**

1. Check git log command output.



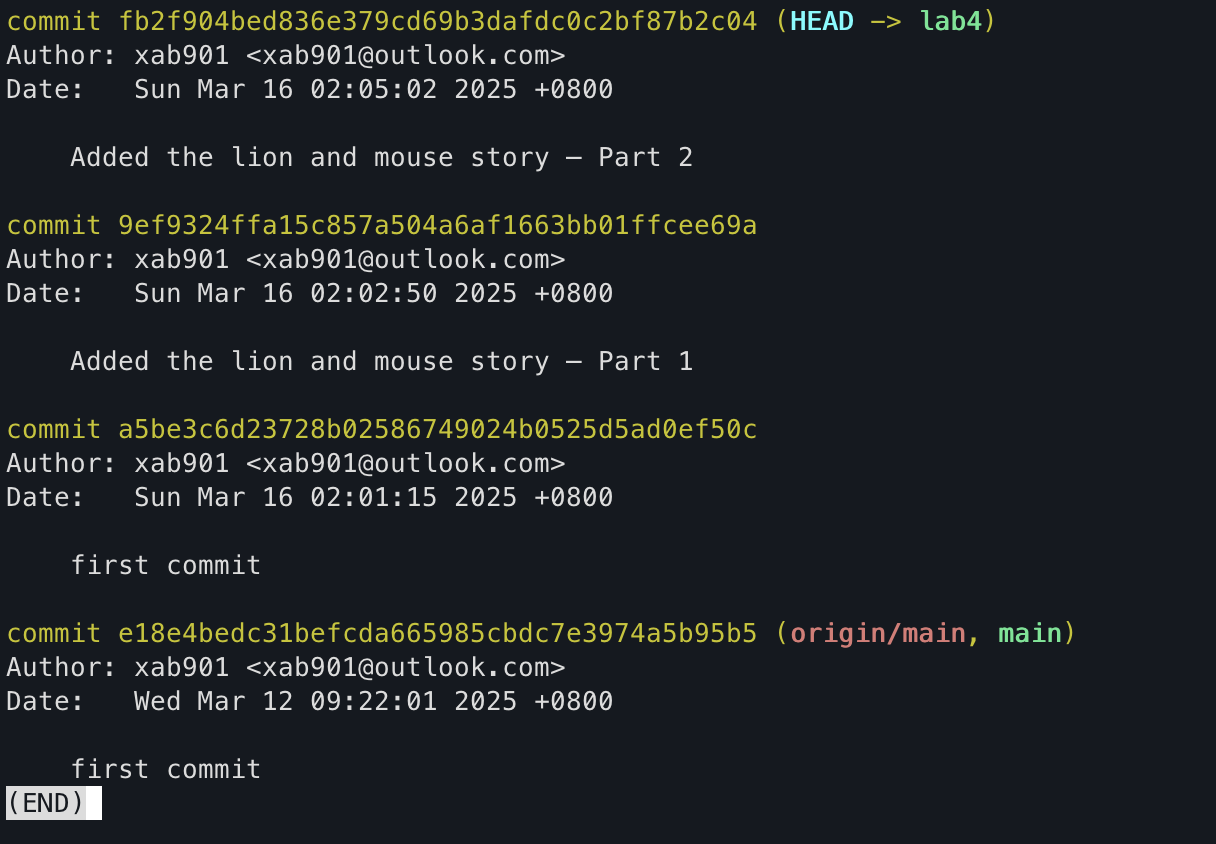
**Comment: Running git log again allows us to verify that the new commit has been recorded.**

1. Add the following text to the file “LionAndMouse.txt”. Then add and commit the file to the Repository with the commit comment "Added the lion and mouse story – Part 2"



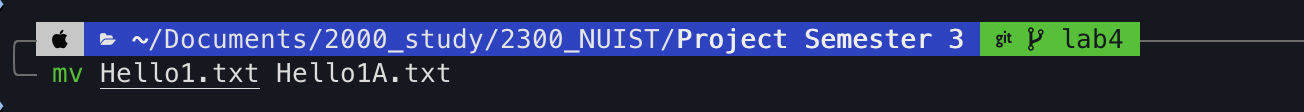
**Comment:** Editing and committing the file again ensures the new content is recorded in the repository. The commit message clearly describes the update.

1. Check git log command output



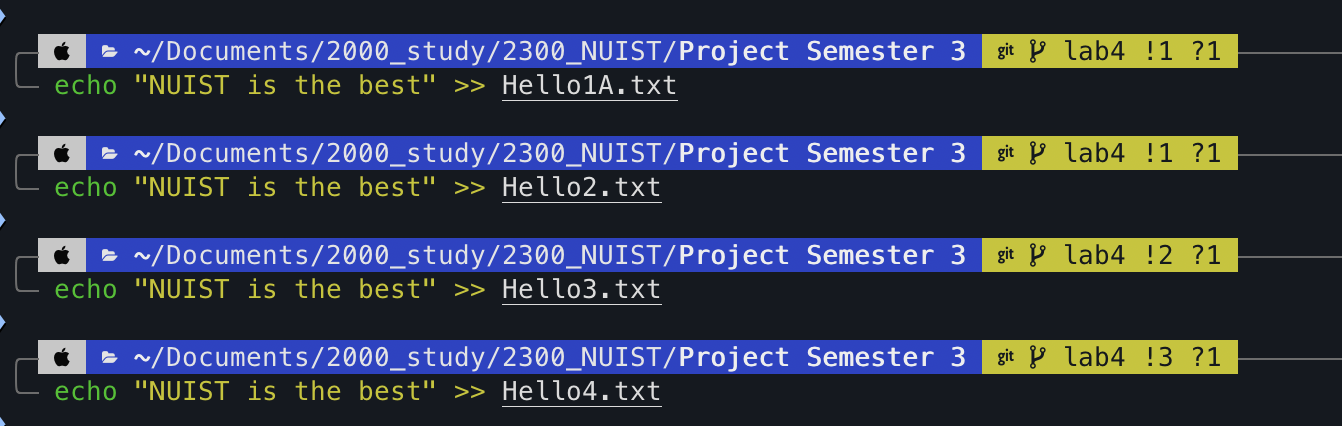
**Comment: This displays the sequence of changes made to the repository, helping track the progression of updates.**

1. Rename Hello1.txt to Hello1A.txt



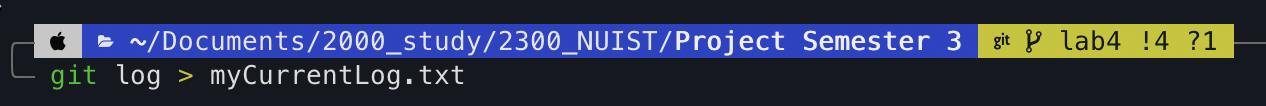
**Comment: Renaming a file using git mv ensures that Git properly tracks the change instead of treating it as a deletion and re-addition.**

1. Add one extra line (“NUIST is the best!”) to each of the .txt files. Add and commit to the repository.



**Comment: This step modifies all text files in bulk. Committing ensures these updates are saved in the repository.**

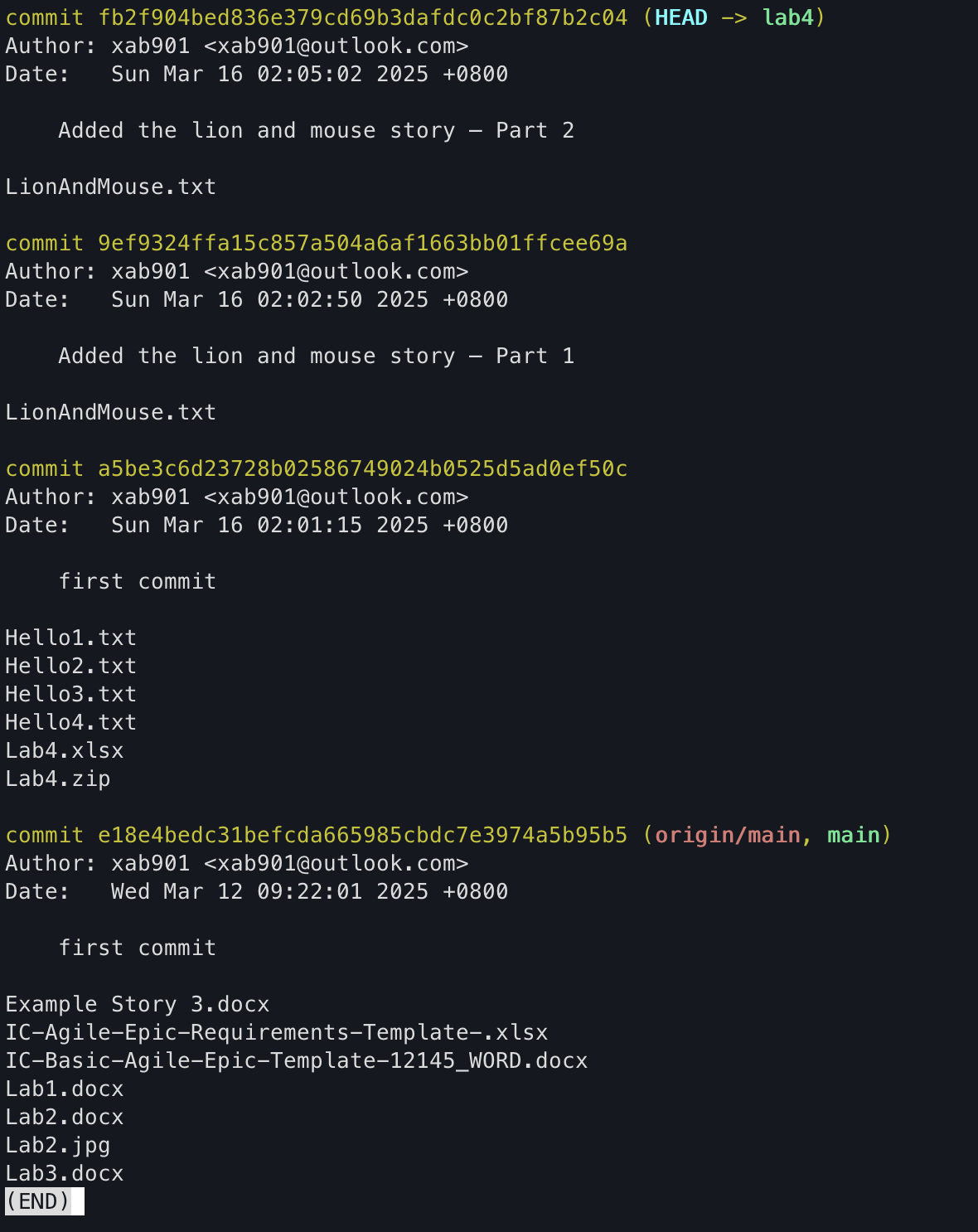
1. Check git log command and redirect (>) the output to a file myCurrentLog.txt



**Comment: Redirecting the log output to a file allows for easier sharing and analysis of the commit history.**

1. List the changed files using the --name-only option with the git log command

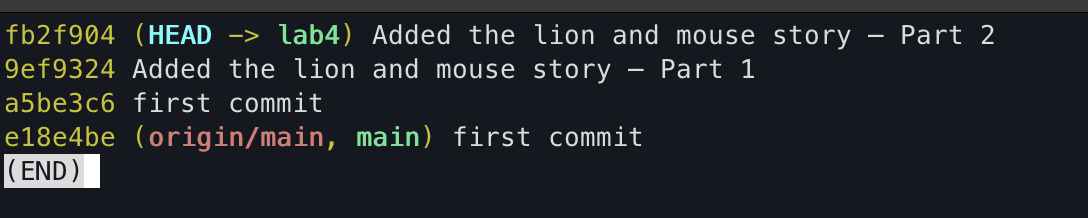
$ git log --name-only



**Comment: The --name-only option provides a concise view of the modified files in each commit, helping to track changes quickly.**

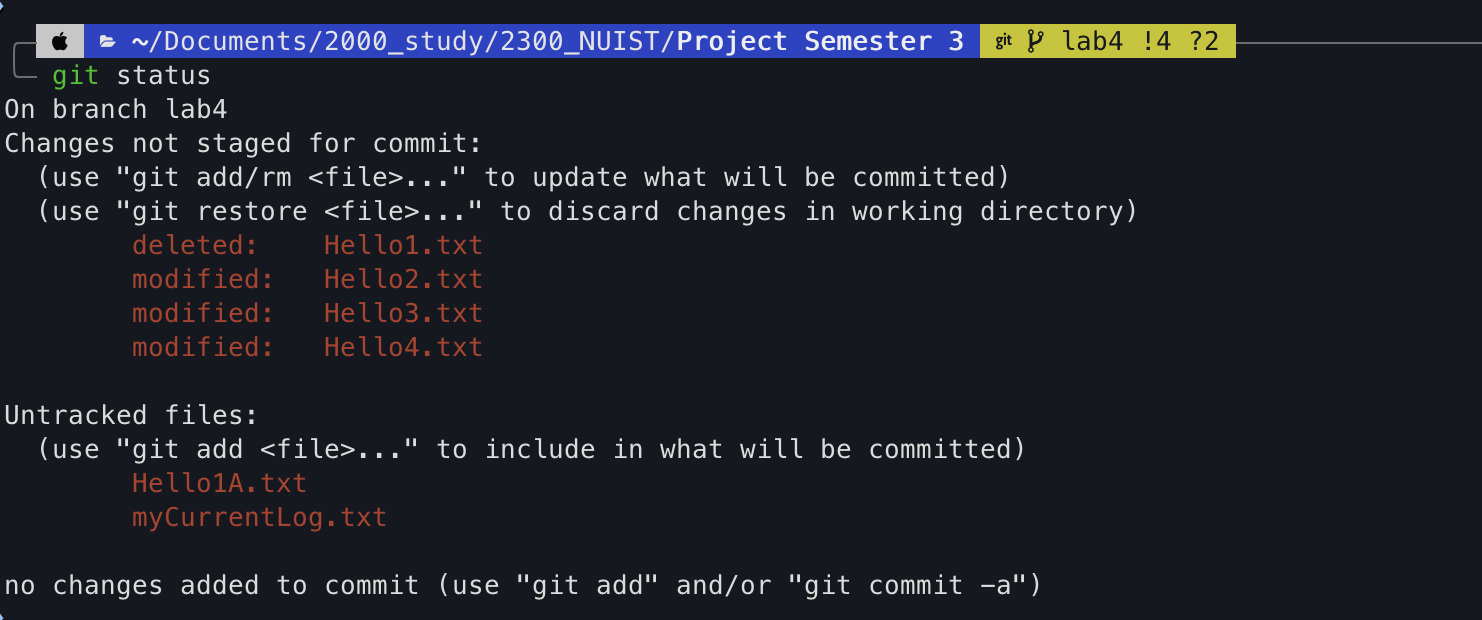
1. . Execute the git log command to display the logs in compact way (one log per line)?

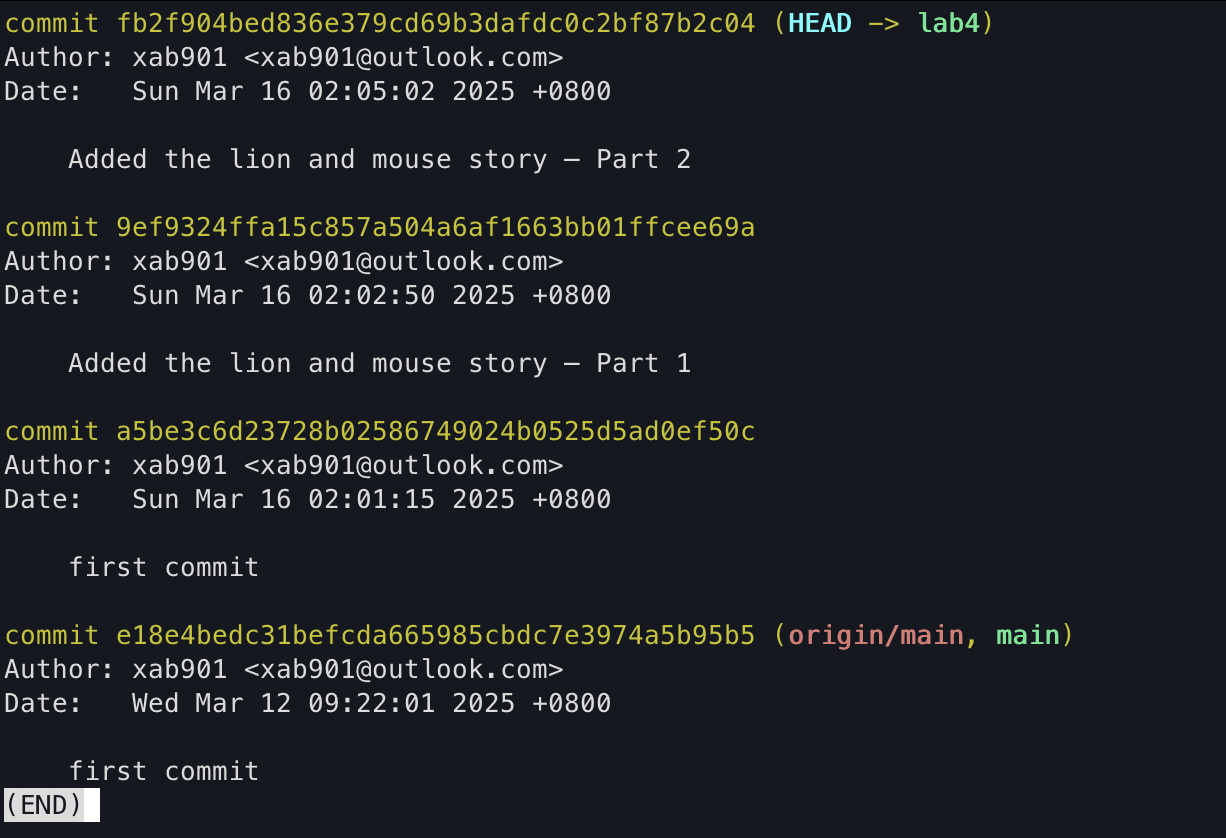
$ git log –oneline



**Comment: The --oneline option simplifies the log view, displaying each commit in a single line. This is useful for quickly reviewing the commit history.**

1. Run the git status and git log commands



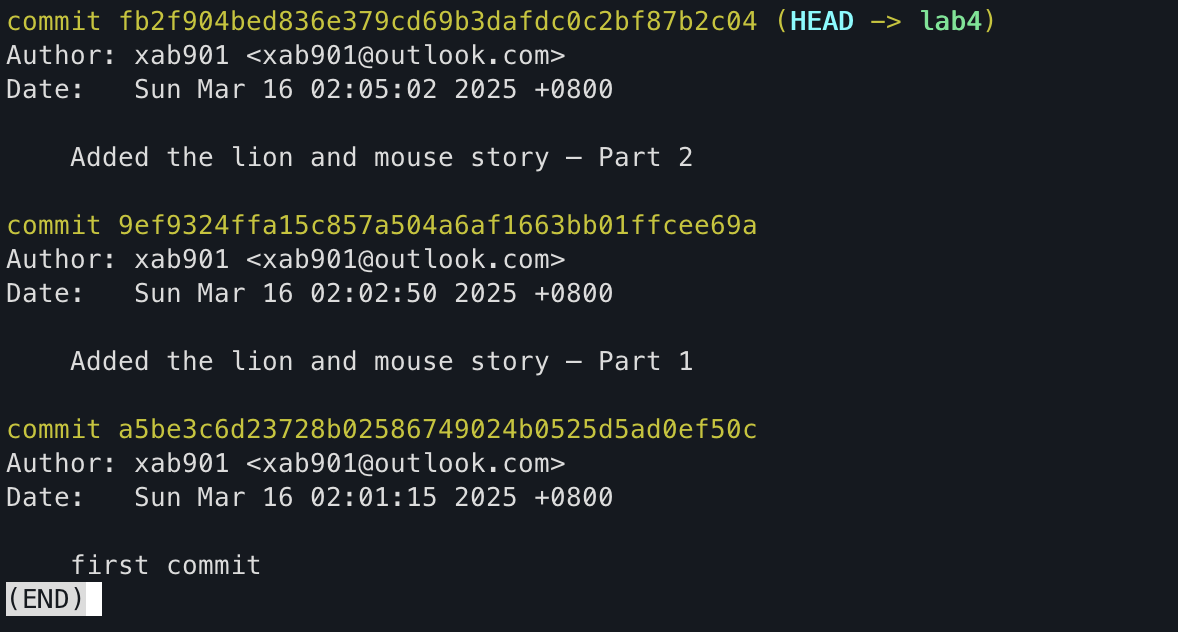


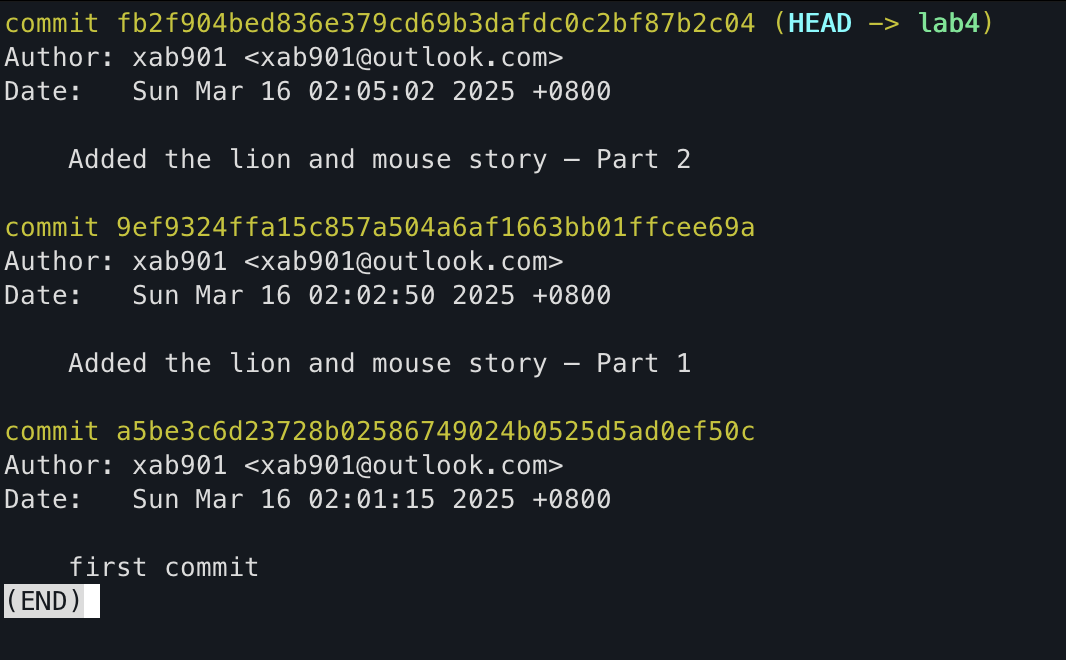
**Comment: git status provides details about the working directory’s state, while git log shows the commit history, offering a comprehensive view of repository changes.**

1. . List the last few commits alone using the --max-count option

$ git log -n 3

$ git log --max-count 3



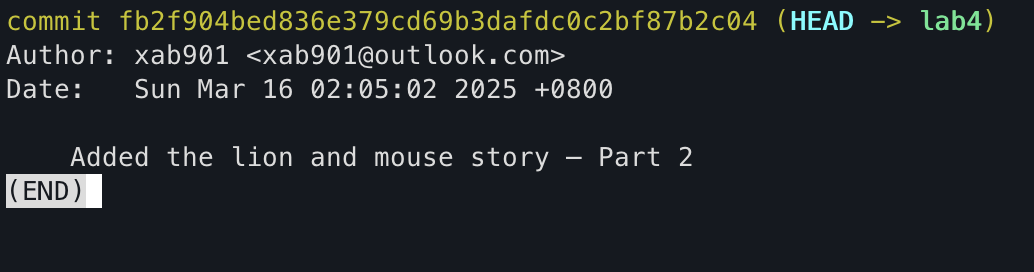


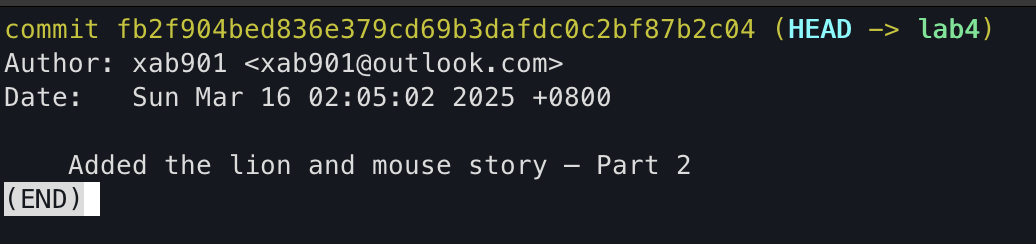
**Comment: Using -n 3 or --max-count 3 limits the log output to the last three commits, making it easier to review recent changes without scrolling through the entire history.**

1. Run git log -n 1 command

$ git log -n 1

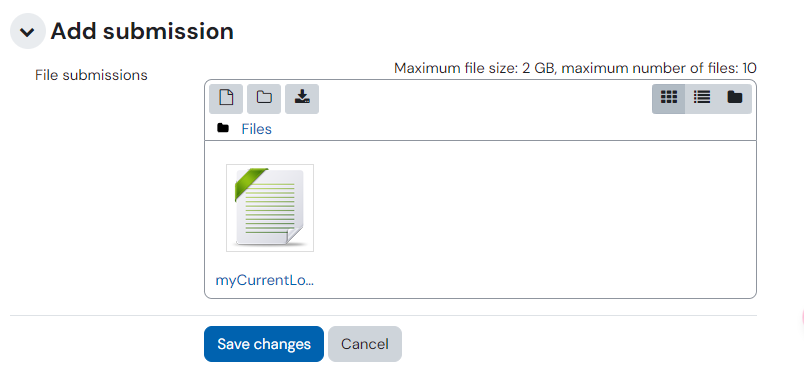
$ git log --max-count 1



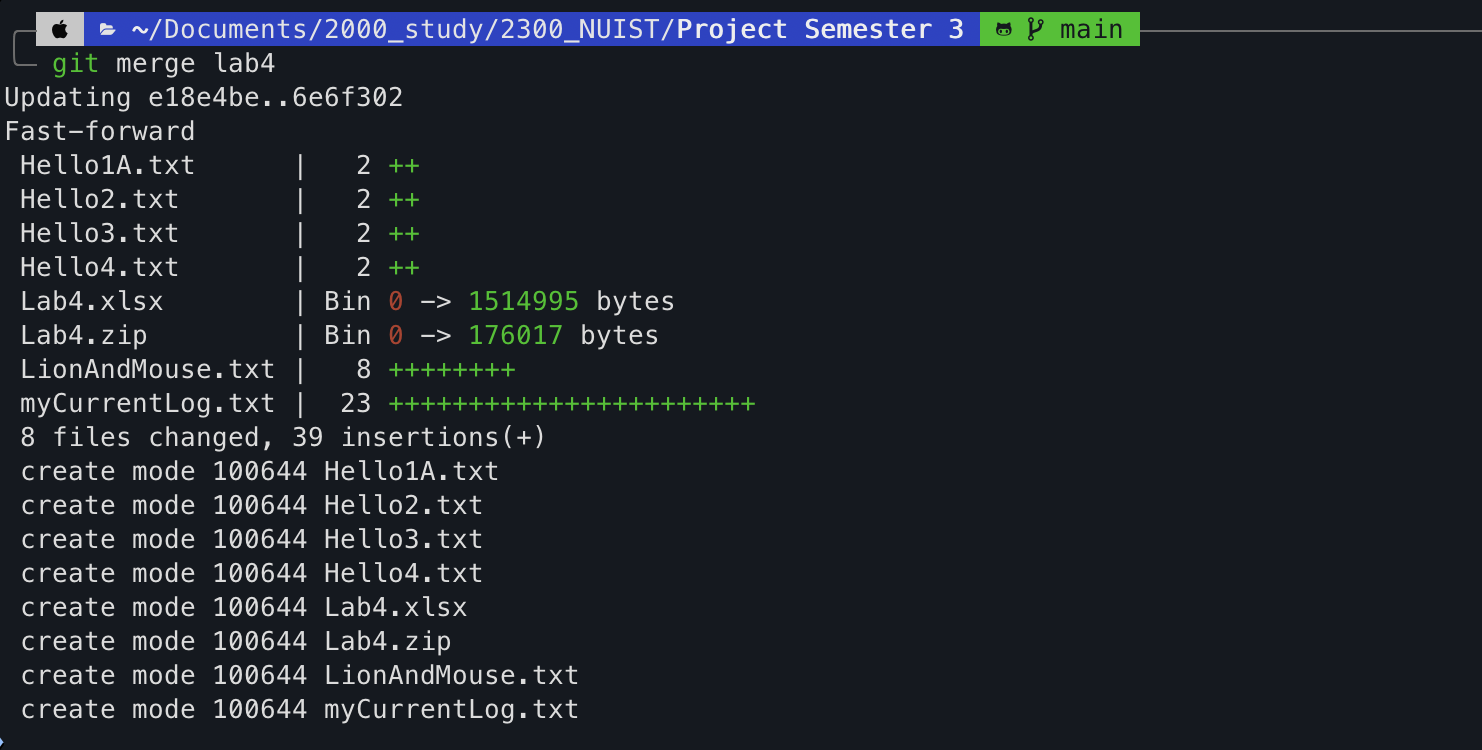


**Comment: This command retrieves only the latest commit, which is useful for verifying the most recent changes.**

1. Submit the myCurrentLog.txt file to Moodle



1. Merge the Lab4 branch in master branch



**Comment: Merging the Lab4 branch into master integrates all changes made in Lab4 into the main codebase, completing the development cycle.**