# DevOps – Assignment 1

**Submitted by:**

- **Anil Joshi**
- **B.Tech CSE CCVT (BATCH 4)**
- **YEAR : 3rd**
- **SAP ID :500106262**

**Submitted to:**

**Dr. Prateek Raj Gautam**

# Assignment -1

## Q1. Demonstrate usage of Subversion, Mercurial with commands, explaination, and screenshots/screen recordings.

## Answer-

Subversion is a centralized version control system, where the repository is stored on a central server, and users check out the latest code from this server. Changes are tracked and updated with each commit.

Basic SVN Commands:

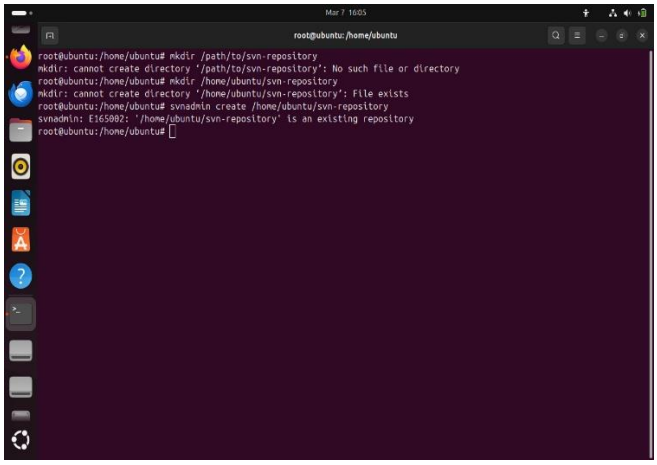
1. Checkout a repository

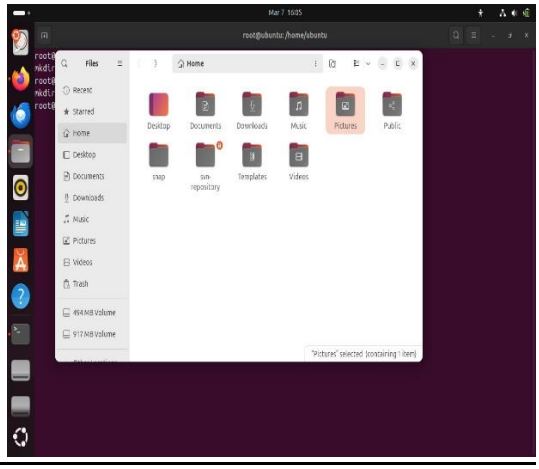
   Command: svn checkout <repository_url>

   Example:

svn checkout https://example.com/svn/repository

Explanation: This command creates a local working copy of the repository. You can now make changes locally.



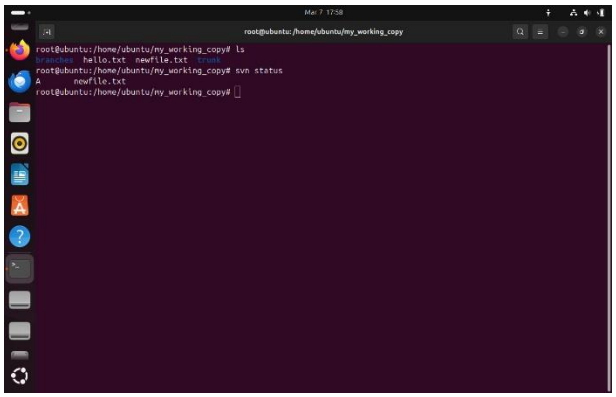
2. Check the status of your working copy

Command: svn status

Example:

svn status

Explanation: This shows the status of files in your working copy (e.g., modified, added, deleted, etc.).



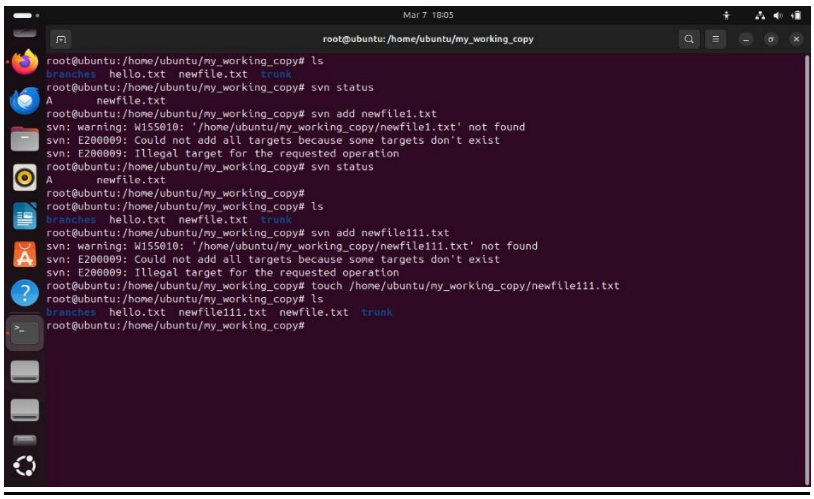

3. Add a new file to version control

Command: svn add <file_name>

Example:

svn add new_file.txt

Explanation: This adds a new file to version control, but it does not commit it yet.



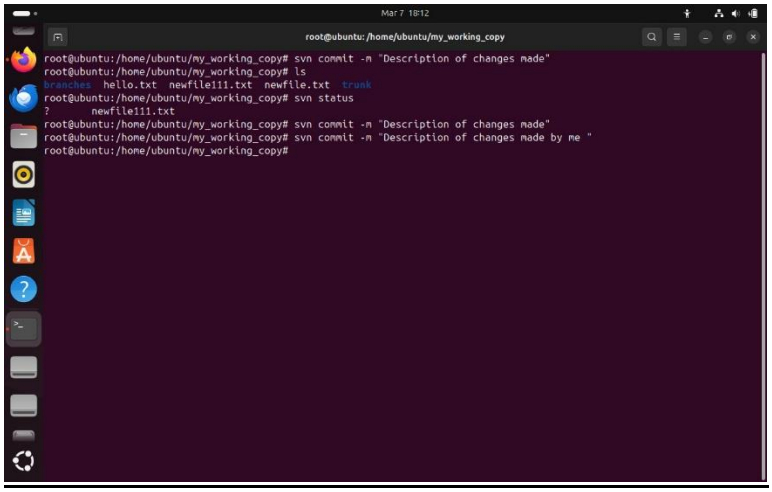
4. Commit changes to the repository

Command: svn commit -m "Commit message"

Example:

svn commit -m "Added new_file.txt"

Explanation: This commits your changes to the repository, and the message should describe the change made.



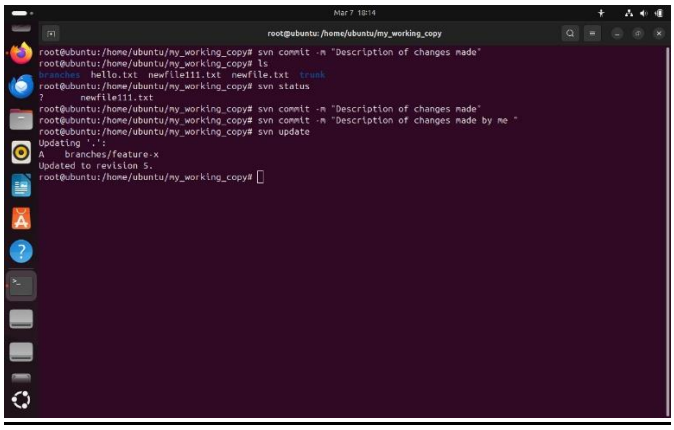

5. Update your working copy

Command: svn update

Example:

svn update

Explanation: This updates your local working copy with the latest changes from the repository.



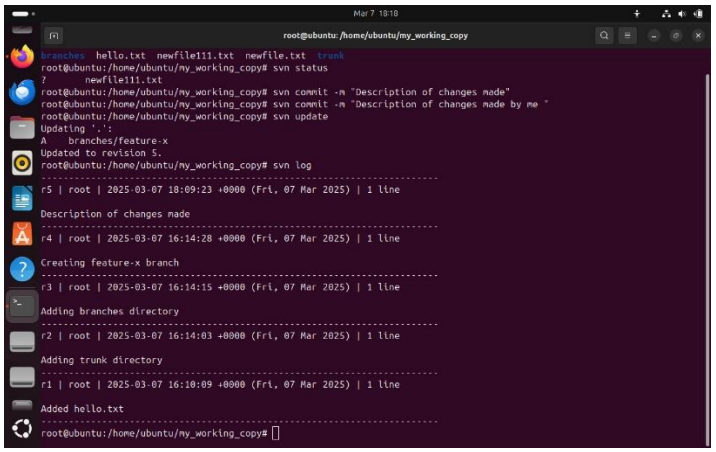
6.View the log of commits

Command: svn log

Example:

svn log

Explanation: This shows a history of the commits made to the repository.



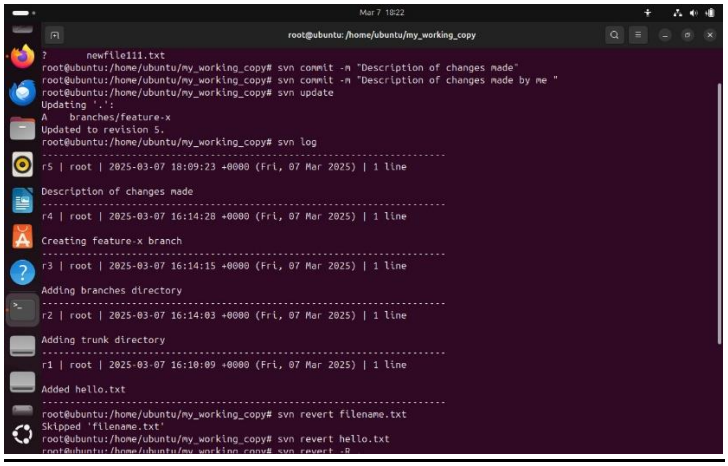

7.Revert changes to a file

Command: svn revert <file_name>

Example:

svn revert new_file.txt

Explanation: This reverts any changes made to the file since the last commit.



Mercurial (often abbreviated as hg) is a distributed version control system, similar to Git, used for tracking changes in source code during software development. Here's a step-by-step guide to demonstrate how to use Mercurial with commands and explanations.

1. Installation of Mercurial

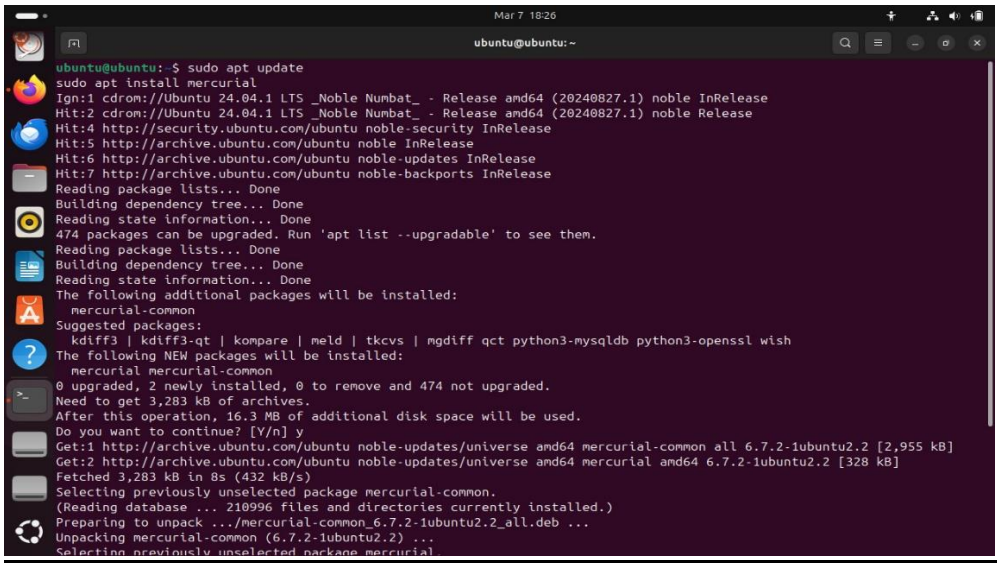
First, you need to install Mercurial. On Ubuntu, you can install it using:

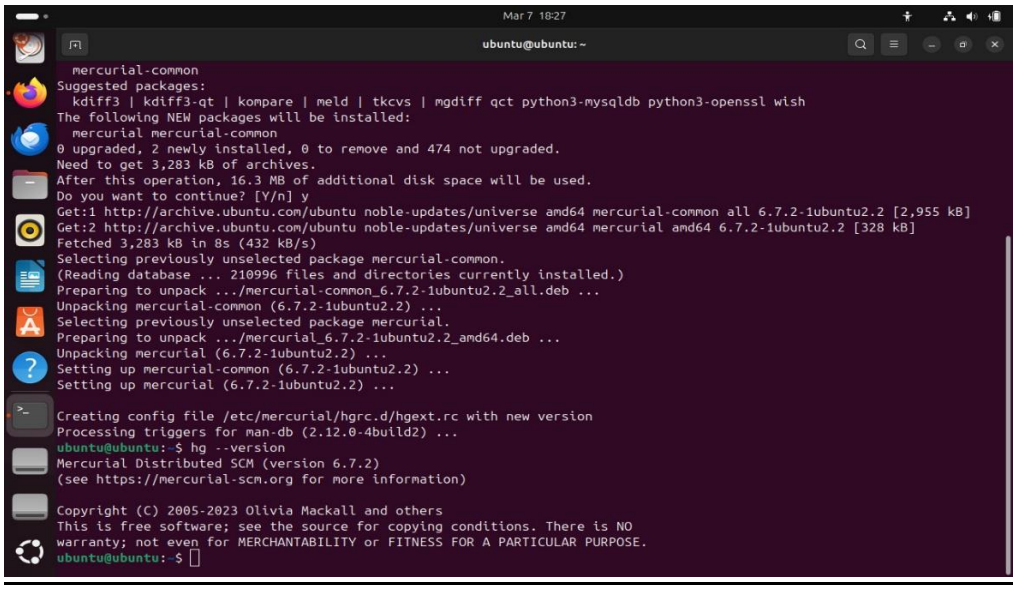
sudo apt update

sudo apt install mercurial

Once installed, verify the installation:

hg --version



## 2. Creating a New Repository

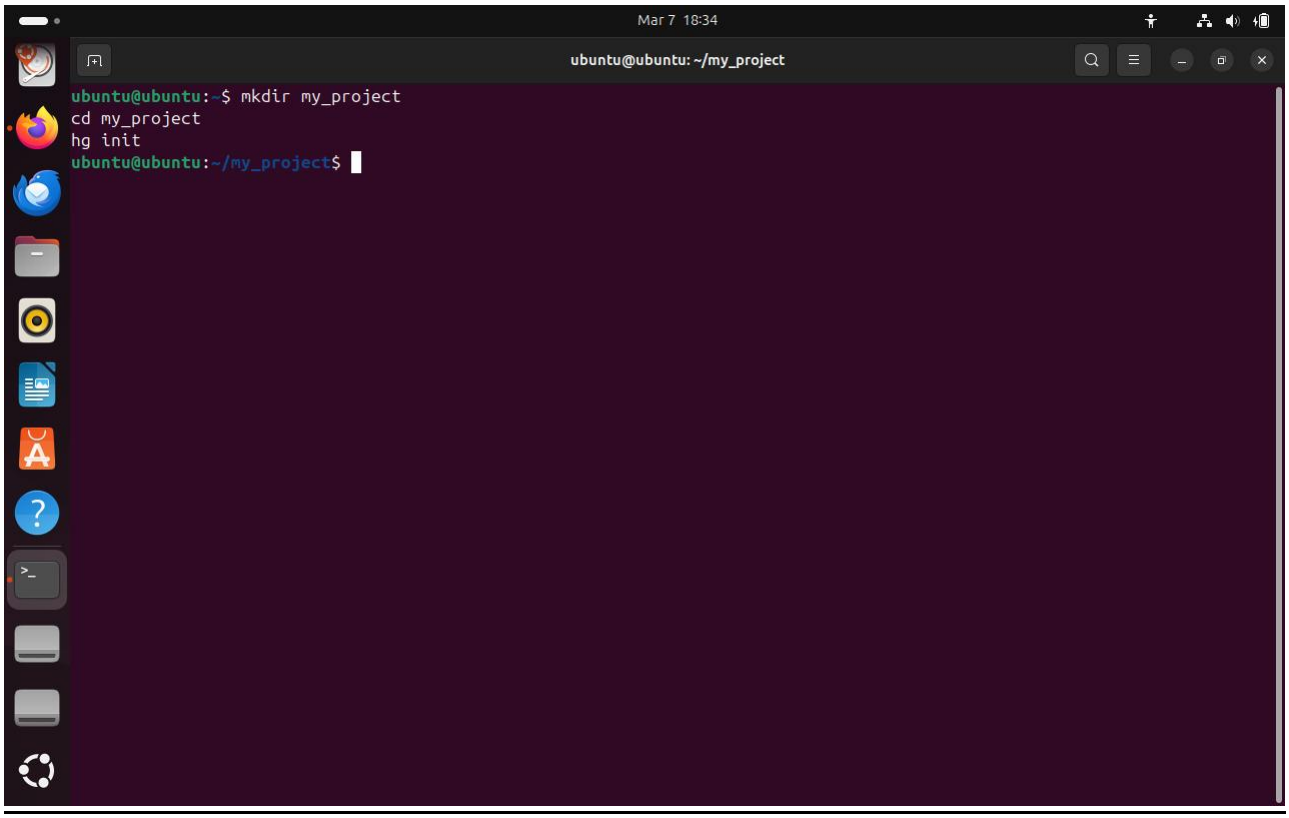
To create a new repository, navigate to your project directory and run the following command:

mkdir my_project

cd my_project

hg init

- hg init: Initializes a new Mercurial repository in the current directory. It creates a .hg folder that stores all the versioning information and history for the repository.



3. Adding Files to the Repository

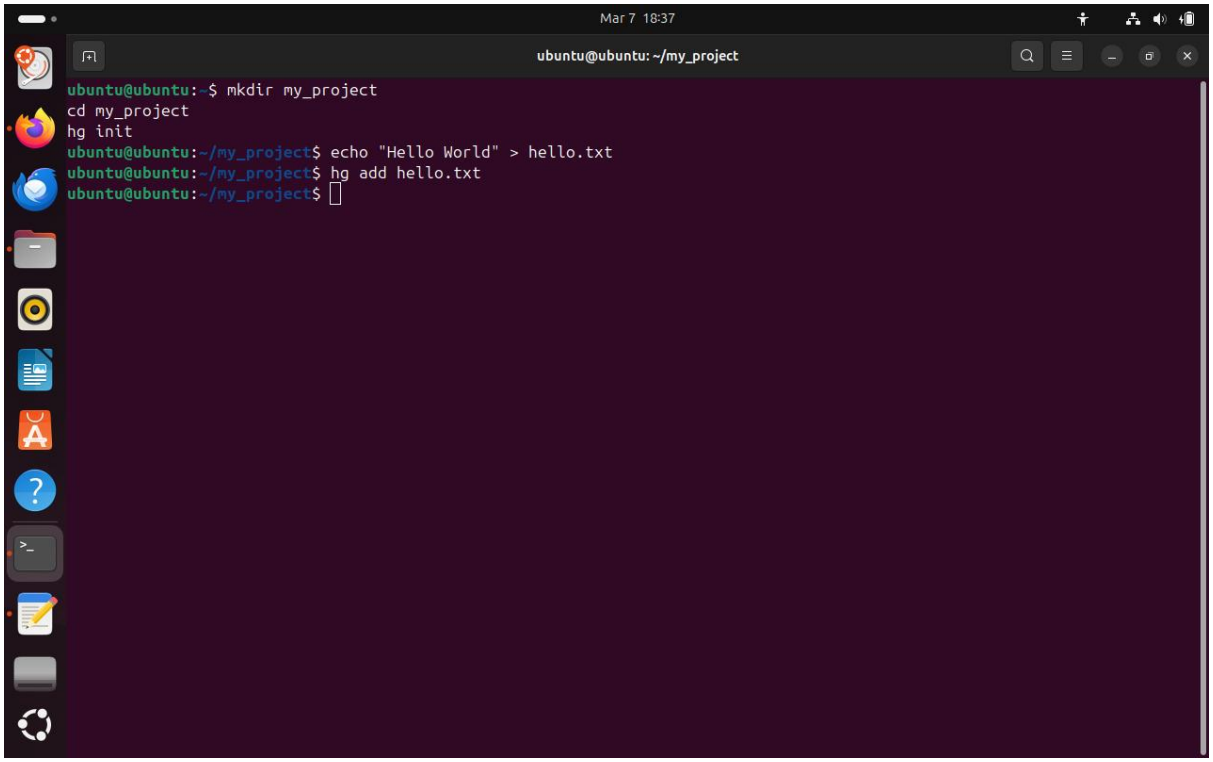
Next, you can add files to the repository. For example, create a new file in the my_project directory:

echo "Hello World" > hello.txt

Now, you can add the file to the repository with:

hg add hello.txt

hg add: Adds a file to the staging area. The file is now tracked by Mercurial, and any changes to it will be recorded.



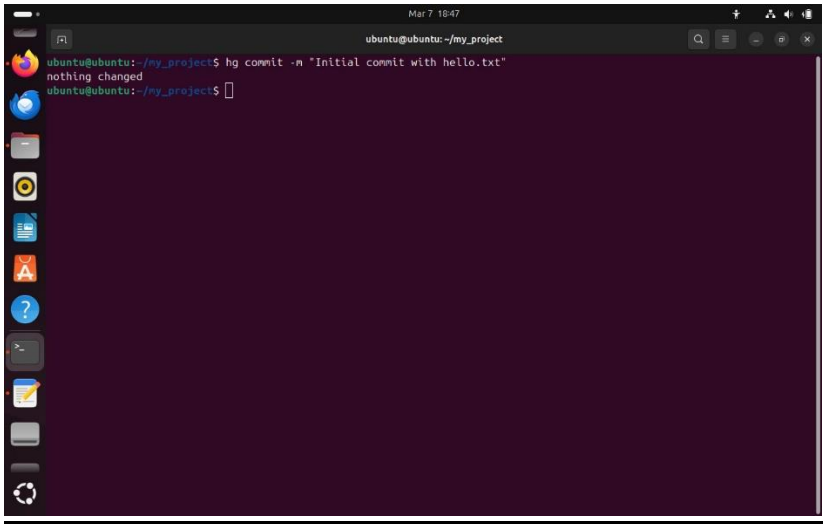
4.Committing Changes

After adding files, the next step is to commit your changes to the repository. This saves a snapshot of the current state of the files in the repository.

hg commit -m "Initial commit with hello.txt"

hg commit: Commits the changes to the repository.

-m "message": The -m option allows you to add a commit message describing the changes.

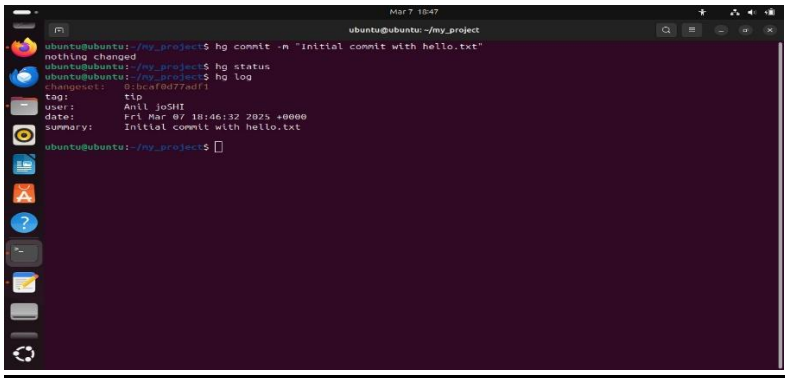



5. Checking the Status of the Repository

To see which files are modified, added, or deleted, use the following command:

hg status

hg status: Shows the status of files in the repository, indicating whether files are modified (M), added (A), or deleted (R).

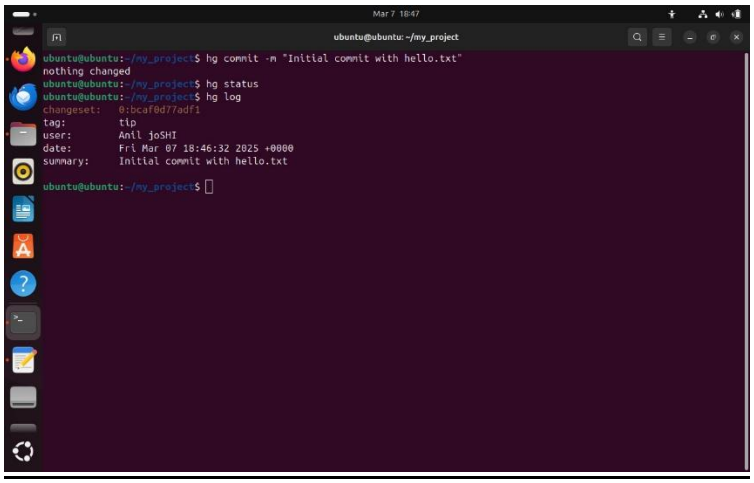


6. Viewing the Log of Commits

You can view the history of commits with the following command:

hg log

hg log: Displays a log of all the commits, including information about each commit such as the revision number, date, author, and commit message.



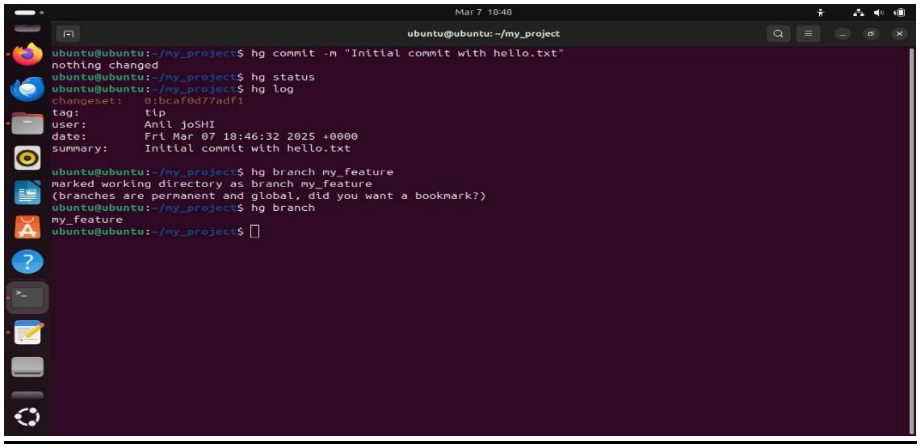

7. Creating a New Branch

Mercurial allows you to work on different branches of development. To create a new branch, use the following command:

hg branch my_feature

hg branch <branch_name>: Creates a new branch. This command creates a named branch in the repository.

hg branch



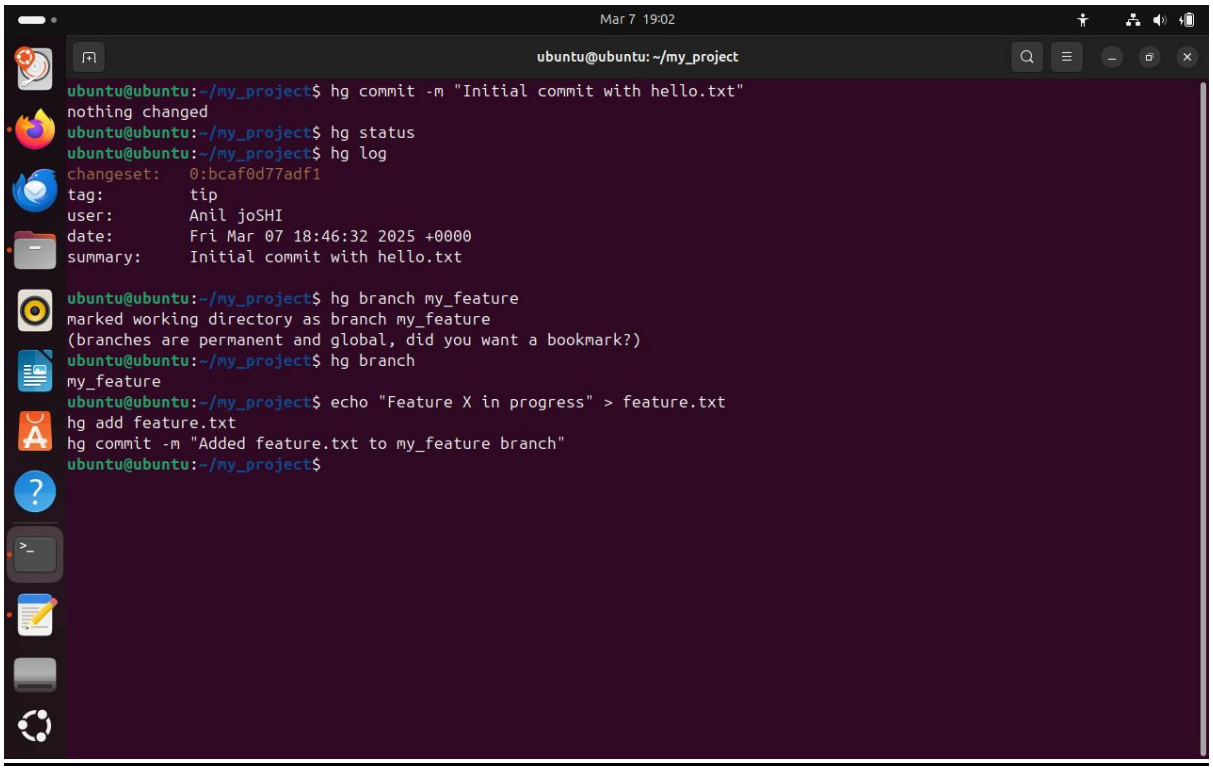
## 8. Committing on the New Branch

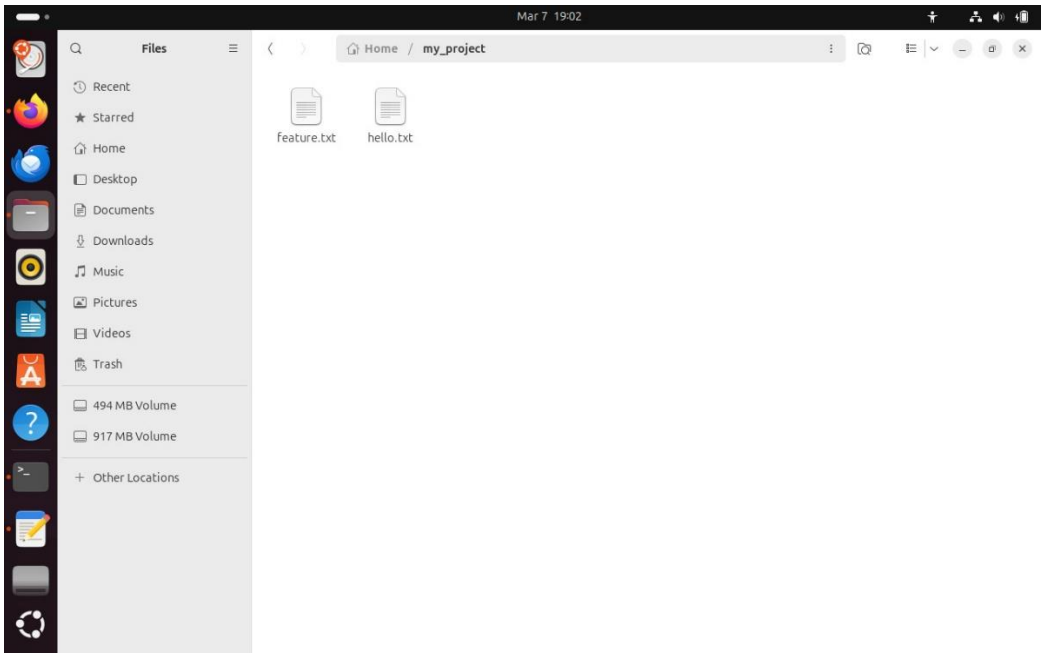
After creating a branch, make some changes. For example:

echo "Feature X in progress" > feature.txt

hg add feature.txt

hg commit -m "Added feature.txt to my_feature branch"



9. Switching Between Branches

To switch to another branch, use the following command:

hg update default

hg update <branch_name>: Switches to a different branch. In this case, we're switching back to the default branch.

10. Merging Branches

Once you have completed the work on a feature branch, you may want to merge it into the default branch. First, switch to the branch you want to merge into:

hg update default

Then, merge the feature branch:

hg merge my_feature

hg merge <branch_name>: Merges changes from the specified branch into the current branch. After this, you will need to commit the merge:

hg commit -m "Merged my_feature into default"