

9. Analysing and Changing Git History

Given a commit ID, how would you use Git to view the details of that specific commit, including the author, date, and commit message?

A commit in Git represents a snapshot of changes.

Using a commit ID (hash), we can view detailed information such as the author, date, commit message, and what changes were made in that commit.

Execution Steps:

Step 1: Create Repository + File + First Commit

```
mkdir commit_view_demo  
cd commit_view_demo  
git init  
echo "line1" > demo.txt  
git add demo.txt  
git commit -m "Initial commit"
```

Step 2 : Make Another Commit

```
echo "line2 added" >> demo.txt  
git add demo.txt  
git commit -m "Second commit"
```

Step 3 : To View all commits

git log - -oneline

Example output:

```
a1b2c3d Second commit - adding line2  
f6e5d4c Initial commit
```

Copy the commit ID of the commit you want to analyse. (Example a1b2c3d)

Step 4: View Complete details of the commit

git show a1b2c3d

This command shows:

- Author
- Date
- Commit message
- What changes were added/removed

Step 5: View ONLY author, date, and message (NO diff)

```
git show - --no-patch a1b2c3d
```

shows:

- Author name
- Email
- Date
- Commit message

(No file changes shown)

10 . Analysing and Changing Git History

Write the command to list all commits made by the author "JohnDoe" between "2023-01-01" and "2023-12-31."

Execution steps

Step 1: Create Repository + First Commit

```
mkdir gitlab10
cd gitlab10
git init
echo "line1" > file.txt
git add file.txt
git commit -m "Initial commit"
```

This commit is by your default author (your Git username).

Step 2 : Create commits by the author "JohnDoe"

```
git commit - --allow-empty - --author="JohnDoe <john@example.com>" -m "Commit 1 by JohnDoe"
```

```
git commit - --allow-empty - --author="JohnDoe <john@example.com>" -m "Commit 2 by JohnDoe"
```

Step 3: Make Some Commits by Another Author (Optional)

```
git commit - --allow-empty - --author="Alice <alice@example.com>" -m "Commit by Alice"
```

Now we have:

- Some commits by **JohnDoe**
- One commit by **Alice**

Step 4: View All Commits

Just to show all commits before filtering:

```
git log - --oneline - --pretty=format:"%h %an %ad %s"
```

This command shows the commit history in a single-line format including:

- short commit ID
- author name
- commit date
- commit message

Step 5 : Filter Commits by Author “JohnDoe” Between Two Dates

Command required as per VTU question (2023):

```
git log -author="JohnDoe" --since="2023-01-01" --until="2023-12-31"
```

Actual execution using the current year (2025):

```
git log --author="JohnDoe" --since="2025-01-01" --until="2025-12-31"
```

NOTE:

VTU’s question asks for the 2023 command. In the lab we executed the same command but for the current year (2025) because our commits were created in 2025

11. Analysing and Changing Git History

Write the command to display the last five commits in the repository's history.

Execution Steps:

Step 1: Create Repository + First Commit

```
mkdir gitlab11  
cd gitlab11  
git init  
echo "line1" > file.txt  
git add file.txt  
git commit -m "Initial commit"
```

Step 2 : Create More Commits

```
echo "line2" >> file.txt  
git add .  
git commit -m "Second commit"  
  
echo "line3" >> file.txt  
git add .  
git commit -m "Third commit"  
  
echo "line4" >> file.txt  
git add .  
git commit -m "Fourth commit"  
  
echo "line5" >> file.txt  
git add .  
git commit -m "Fifth commit"  
  
echo "line6" >> file.txt  
git add .  
git commit -m "Sixth commit"
```

Step 3: Display All Commits (just to verify)

git log - --oneline

This shows the entire commit list in short form.

Step 4 : Display the Last 5 Commits

git log -n 5

Or if you want the short version: **git log - --oneline -n 5**

Git will show the **five most recent commits**, starting from the latest.

12. Analysing and Changing Git History

Write the command to undo the changes introduced by the commit with the ID "abc123".

Execution steps

Step 1: Create repository and first commit

```
mkdir gitlab12  
cd gitlab12  
git init  
  
echo "line1" > file.txt  
git add file.txt  
git commit -m "Initial commit"
```

Step 2 : Create another commit (this will be the one we undo)

```
echo "line2" >> file.txt  
git add .  
git commit -m "Second commit"
```

Step 3 : View commit IDs

git log - --oneline

Example output:

```
abc123 Second commit  
d4e5f6 Initial commit
```

(Note: the top commit ID — for “Second commit” — is the one we will undo.)

Step 4 : Undo the commit

git revert abc123

Step 5 : Verify the result

```
git log - -oneline  
cat file.txt
```

You will see:

- A new commit: "**Revert Second commit**"
- The content of `file.txt` goes back to before that commit (line2 removed)