VTU 2025 Scheme

Project Management (with Git)

1BCSL307A – 3rd Semester

EXPERIMENT – 09

Git History and Commit Inspection

Viewing Details of a Specific Commit

Aim

To view the complete details of a specific Git commit using its commit ID, including the author, date, commit message, and changes introduced by the commit.

Objectives

After completing this experiment, the student will be able to:

- Understand Git commit history.
- Display the list of commits in a repository.
- Identify the commit ID of a specific commit.
- Inspect a commit using its commit ID.
- View the author of a commit.
- View the date of a commit.
- View the commit message.
- View the changes introduced by a specific commit.

Software Requirements

- Git
- Windows Command Prompt / PowerShell / Git Bash
- Text Editor such as Notepad or Visual Studio Code

Theory

A **commit** in Git represents a saved snapshot of the project at a particular point in time. Every commit contains important information such as:

- Commit ID
- Author name

- Author email
- Date and time
- Commit message
- Changes introduced by the commit

Git provides commands to inspect this information whenever required.

What is a Commit ID?

Every Git commit is identified by a unique hash value called a **commit ID** or **commit hash**.

A commit ID may look like:

```
a1b2c3d4e5f678901234567890abcdef12345678
```

Git can also display a shorter form of the commit ID:

```
a1b2c3d
```

The short ID is usually sufficient when it uniquely identifies a commit in the repository.

Git Show Command

The `git show` command displays detailed information about a Git object.

To inspect a specific commit:

```
git show <commit-id>
```

The output can include:

- Complete commit ID
- Author
- Date
- Commit message
- Files modified
- Lines added
- Lines removed

Commit Information Concept

The following diagram shows the information associated with a Git commit.

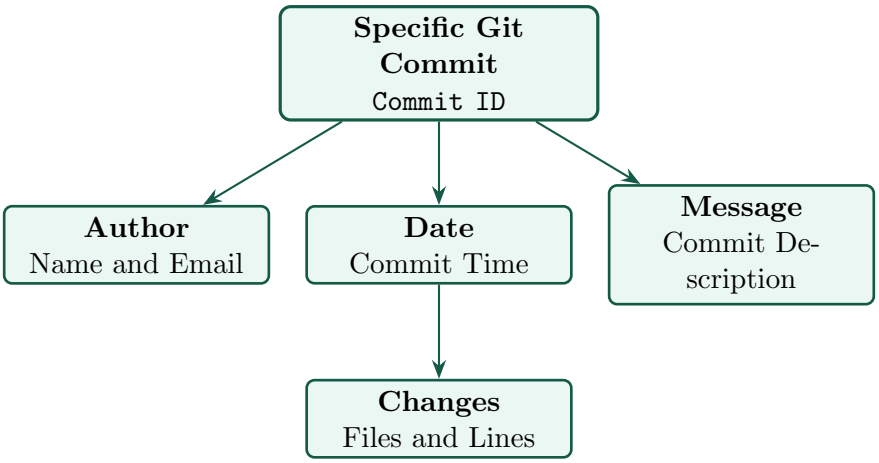


Diagram Explanation:

- Every Git commit has a unique commit ID.
- The commit stores author information.
- The commit records the date and time.
- A commit message describes the purpose of the commit.
- Git can also display the file changes introduced by the commit.

Important Commands

Command	Purpose
`git log`	Displays detailed commit history.
`git log --oneline`	Displays compact commit history with short commit IDs.
`git show <commit-id>`	Displays detailed information and changes for a specific commit.
`git show --stat <commit-id>`	Displays commit details and file-change statistics.
`git log -1 <commit-id>`	Displays information about one specified commit.

Procedure

Step 1: Open Command Prompt, PowerShell, or Git Bash.

Step 2: Create a directory for Experiment 09.

Step 3: Initialize a Git repository.

Step 4: Create a project file.

Step 5: Stage and commit the file.

Step 6: Modify the file and create another commit.

Step 7: Create one more project change and commit it.

Step 8: Display the commit history.

Step 9: Select a required commit ID.

Step 10: Execute `git show` using the selected commit ID.

Step 11: Observe the author information.

Step 12: Observe the commit date.

Step 13: Observe the commit message.

Step 14: Observe the changes introduced by the commit.

Step 15: Verify the required commit information.

Execution

Step 1: Create Experiment Directory

Execute:

```
mkdir git-experiment-09
cd git-experiment-09
```

The commands create and open the Experiment 09 directory.

Step 2: Initialize Git Repository

Execute:

```
git init
```

Sample Output:

```
Initialized empty Git repository in
.../git-experiment-09/.git/
```

Step 3: Create Project File

Execute:

```
echo Project Management with Git > project.txt
```

The file `project.txt` is created.

Step 4: Create First Commit

Execute:

```
git add project.txt
git commit -m "Create project file"
```

Note:
If Git asks for user identity, configure it using:

```
git config --global user.name "Your Name"
git config --global user.email "your@email.com"
```

Then execute the commit command again.

Step 5: Create Second Commit

Modify the project file:

```
echo Git commit history experiment >> project.txt
```

Stage and commit the change:

```
git add project.txt
git commit -m "Add Git history content"
```

Step 6: Create Third Commit

Create another file:

```
echo Experiment 09 > experiment.txt
```

Stage and commit it:

```
git add experiment.txt
git commit -m "Add experiment information"
```

The repository now contains multiple commits.

Step 7: Display Commit History

Execute:

```
git log --oneline
```

Sample Output:

```
c3d4e5f Add experiment information
b2c3d4e Add Git history content
a1b2c3d Create project file
```

Important:
The actual commit IDs generated on your computer will be different.
Suppose we want to inspect:

```
b2c3d4e Add Git history content
```

Copy the actual commit ID displayed on your system.

Step 8: View Specific Commit Details

Execute:

```
git show b2c3d4e
```

Replace `b2c3d4e` with your actual commit ID.
Sample Output:

```
commit b2c3d4e...
Author: Your Name <your@email.com>
Date:   Thu Sep 17 20:00:00 2026 +0530

    Add Git history content

diff --git a/project.txt b/project.txt
...
+Git commit history experiment
```

Step 9: Identify the Author

In the output of `git show`, observe:

```
Author: Your Name <your@email.com>
```

This shows the author who created the commit.

Step 10: Identify the Date

Observe the `Date` field:

```
Date: Thu Sep 17 20:00:00 2026 +0530
```

The actual date and time will depend on when the commit was created.

Step 11: Identify the Commit Message

The commit message appears below the author and date information.
For example:

```
Add Git history content
```

The message explains the purpose of the commit.

Step 12: View File Changes

The lower part of the `git show` output displays the changes introduced by the commit.
For example:

```
+Git commit history experiment
```

A line beginning with:

```
+
```

normally represents an added line.
A line beginning with:

```
-
```

normally represents a removed line in the diff output.

Step 13: View Commit Statistics

Execute:

```
git show --stat b2c3d4e
```

Replace the sample ID with the actual commit ID.
Sample Output:

```
commit b2c3d4e...
Author: Your Name <your@email.com>
Date: ...

    Add Git history content

 project.txt | 1 +
 1 file changed, 1 insertion(+)
```

This gives a short summary of the files changed by the commit.

Step 14: View One Commit Without Diff

To display the selected commit information without the complete file difference, execute:

```
git log -1 b2c3d4e
```

This displays the commit ID, author, date, and commit message.

Complete Command Sequence

The complete command sequence for Experiment 09 is:

```
mkdir git-experiment-09
cd git-experiment-09

git init

echo Project Management with Git > project.txt

git add project.txt
```

```
git commit -m "Create project file"

echo Git commit history experiment >> project.txt

git add project.txt
git commit -m "Add Git history content"

echo Experiment 09 > experiment.txt

git add experiment.txt
git commit -m "Add experiment information"

git log --oneline

git show <commit-id>

git show --stat <commit-id>

git log -1 <commit-id>
```


How to Read git show Output

Consider the following simplified output:

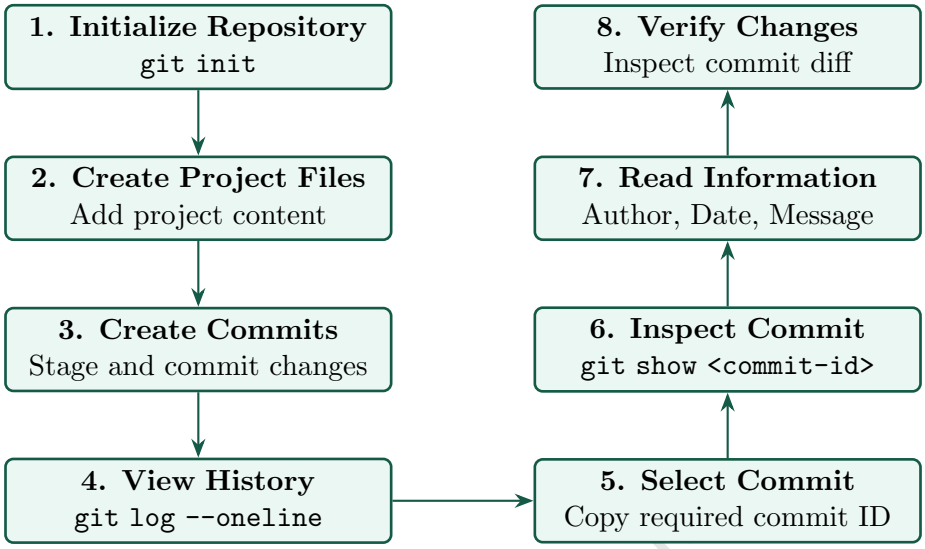
```
commit b2c3d4e...
Author: Student <student@email.com>
Date:   Thu Sep 17 20:00:00 2026 +0530

    Add Git history content

diff --git a/project.txt b/project.txt

+Git commit history experiment
```

It can be understood as follows:

Field	Meaning
Commit	Unique ID of the selected commit.
Author	Name and email of the commit author.
Date	Date and time when the commit was created.
Message	Description given while creating the commit.
Diff	Shows the changes introduced by the commit.
+	Represents an added line in the diff.
-	Represents a removed line in the diff.

Experiment Workflow

The complete workflow is shown below.



Workflow Explanation:

1. Initialize the Git repository.
2. Create the required project files.
3. Create multiple Git commits.
4. Display the commit history using `git log --oneline`.
5. Select the required commit ID.
6. Execute `git show` with the commit ID.
7. Observe the author, date, and commit message.
8. Verify the changes introduced by the selected commit.

git log vs git show

Feature	git log	git show
Main Purpose	View commit history	Inspect a particular Git object or commit
Commit Information	Yes	Yes
File Changes	Not shown by default	Shown for a commit by default
Typical Command	`git log --oneline`	`git show <commit-id>`
Useful For	Finding commit IDs	Detailed commit inspection

Result

Thus, the details of a specific Git commit were successfully viewed using its commit ID. The author, date, commit message, and changes introduced by the selected commit were identified and verified.

Viva Questions and Answers

1. **What is a Git commit?**

   A Git commit is a saved snapshot of project changes at a particular point in time.

2. **What is a commit ID?**

   A commit ID is a unique hash value used to identify a Git commit.

3. **Which command displays compact commit history?**

```
git log --oneline
```

4. **Which command displays the details of a specific commit?**

```
git show <commit-id>
```

5. **What information can git show display for a commit?**

   It can display the commit ID, author, date, commit message, and file changes.

6. **What does the Author field represent?**

   It identifies the author associated with the commit.

7. **What does the Date field represent?**

   It shows the date and time recorded for the commit.

8. **What is a commit message?**

   A commit message is a short description explaining the purpose of the commit.

9. **What does a plus sign in a Git diff indicate?**

   It indicates a line added by the change being displayed.

10. **What does a minus sign in a Git diff indicate?**

    It indicates a line removed by the change being displayed.

11. **How can we view statistics for a particular commit?**

```
git show --stat <commit-id>
```

12. **Can we use a shortened commit ID?**

    Yes, if the shortened ID uniquely identifies the commit in the repository.

13. **What is the difference between git log and git show?**

    `git log` is mainly used to view commit history, while `git show` can display detailed information and changes for a specific commit.

14. **How can we display only one specified commit using git log?**

```
git log -1 <commit-id>
```

Conclusion

In this experiment, Git commit inspection was successfully demonstrated. Multiple commits were created, a required commit ID was identified using the Git history, and the selected commit was inspected. Its author, date, commit message, and associated changes were successfully viewed and verified.