VTU 2025 Scheme

Project Management (with Git)

1BCSL307A – 3rd Semester

EXPERIMENT – 07

Tags, Releases, and Advanced Git Operations

Creating a Lightweight Tag `v1.0`

Aim

To create a lightweight Git tag named `v1.0` for a specific commit and verify the created tag using Git commands.

Objectives

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

- Understand the concept of tags in Git.
- Understand the purpose of version tagging.
- View the commit history and identify a specific commit.
- Create a lightweight tag for a commit.
- Display the available tags in a repository.
- View the commit associated with a tag.
- Understand the difference between a lightweight tag and an annotated tag.

Software Requirements

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

Theory

A **tag** in Git is a name assigned to a specific point in the repository history.

Tags are commonly used to mark important versions of a software project such as:

- Version 1.0
- Version 2.0
- Beta release

- Stable release
- Final project release

Unlike branches, tags are normally used as fixed references to important commits.
For example, a developer can create the tag `v1.0` to identify the commit representing version 1.0 of the project.

What is a Lightweight Tag?

A **lightweight tag** is a simple reference to a specific Git commit.
It works like a permanent name or pointer assigned to a commit.
A lightweight tag does not store additional information such as a tagging message.
The general syntax is:

```
git tag tag-name commit-id
```

For this experiment:

```
git tag v1.0 <commit-id>
```

Why are Git Tags Used?

- To identify important versions of a project.
- To mark software releases.
- To easily refer to a specific commit.
- To maintain clear version history.
- To identify stable project states.
- To simplify release management.

Git Tag Concept

The following diagram shows a lightweight tag attached to a specific commit.

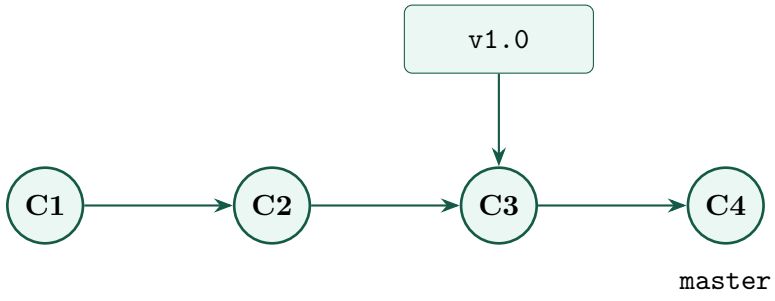


Diagram Explanation:

- C1, C2, C3, and C4 represent project commits.
- Each commit represents a saved version of the project.
- The tag `v1.0` points to commit C3.
- The tag gives a meaningful name to that specific commit.
- Future commits do not automatically move the tag.

Important Commands

Command	Purpose
`git log --oneline`	Displays the commit history with shortened commit IDs.
`git tag`	Displays the available tags.
`git tag v1.0`	Creates `v1.0` for the current commit.
`git tag v1.0 <commit-id>`	Creates `v1.0` for a specific commit.
`git show v1.0`	Displays the commit information associated with the tag.

Procedure

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

Step 2: Create a new directory for Experiment 07.

Step 3: Initialize the directory as a Git repository.

Step 4: Create the first project file.

Step 5: Stage and commit the file.

Step 6: Create another change in the project.

Step 7: Commit the second change.

Step 8: Display the commit history.

Step 9: Identify the commit to be tagged.

Step 10: Create a lightweight tag named `v1.0` for the selected commit.

Step 11: Display all available tags.

Step 12: View the commit associated with `v1.0`.

Step 13: Verify that the tag was created successfully.

Execution

Step 1: Create Experiment Directory

Execute:

```
mkdir git-experiment-07
cd git-experiment-07
```

Explanation:
The first command creates a directory named `git-experiment-07`.
The second command moves into the directory.

Step 2: Initialize Git Repository

Execute:

```
git init
```

Sample Output:

```
Initialized empty Git repository in
.../git-experiment-07/.git/
```

The current directory is now initialized as a Git repository.

Step 3: Create the First Project File

Execute:

```
echo Project Version 1 > project.txt
```

The command creates `project.txt` containing:

```
Project Version 1
```

Step 4: Add the File

Execute:

```
git add project.txt
```

The project file is added to the staging area.

Step 5: Create First Commit

Execute:

```
git commit -m "Create project version 1"
```

The first project version is now stored in the Git repository.
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 6: Create Another Project Change

Execute:

```
echo New feature added >> project.txt
```

The file now contains:

```
Project Version 1
New feature added
```

Step 7: Commit the New Change

Execute:

```
git add project.txt
git commit -m "Add new feature"
```

The repository now contains at least two commits.

Step 8: Display Commit History

Execute:

```
git log --oneline
```

Sample Output:

```
b7c8d9e Add new feature
a1b2c3d Create project version 1
```

Note:
The commit IDs displayed on your computer will be different.
Suppose we want to create the tag `v1.0` for the first commit:

```
a1b2c3d
```

Use the actual commit ID displayed on your computer.

Step 9: Create Lightweight Tag

Execute:

```
git tag v1.0 a1b2c3d
```

Important:
Replace:

```
a1b2c3d
```

with the actual commit ID obtained from:

```
git log --oneline
```

Explanation:
The command creates a lightweight tag named `v1.0` pointing to the selected commit.

Step 10: Display Available Tags

Execute:

```
git tag
```

Sample Output:

```
v1.0
```

This confirms that the tag has been created.

Step 11: View the Tagged Commit

Execute:

```
git show v1.0
```

Sample Output:

```
commit a1b2c3d...
Author: Your Name <your@email.com>
Date: ...

    Create project version 1

Project Version 1
```

The exact output depends on the commit and user configuration.

Step 12: Display Tag with Commit History

Execute:

```
git log --oneline --decorate
```

Sample Output:

```
b7c8d9e (HEAD -> master) Add new feature
a1b2c3d (tag: v1.0) Create project version 1
```

Explanation:
The output clearly shows that the tag `v1.0` points to the selected commit.
Note:

Depending on the Git configuration, the default branch may be named `main` instead of `master`. This does not affect the tagging operation.

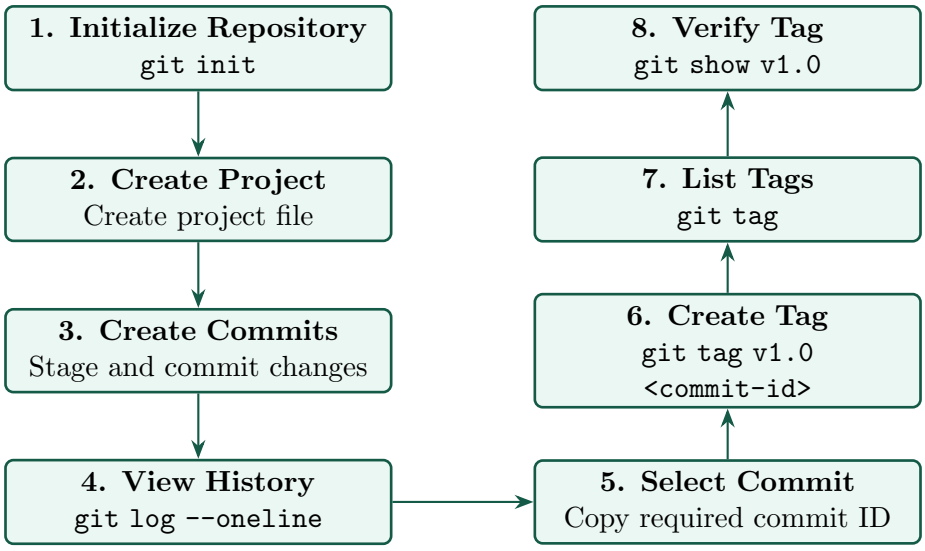
Complete Command Sequence

The complete command sequence for Experiment 07 is:

```
mkdir git-experiment-07
cd git-experiment-07

git init

echo Project Version 1 > project.txt

git add project.txt

git commit -m "Create project version 1"

echo New feature added >> project.txt

git add project.txt

git commit -m "Add new feature"

git log --oneline

git tag v1.0 <commit-id>

git tag

git show v1.0

git log --oneline --decorate
```

Experiment Workflow

The complete workflow of Experiment 07 is shown below.



Lightweight Tag vs Annotated Tag

Feature	Lightweight Tag	Annotated Tag
Type	Simple reference to a commit	Stored as a separate Git object
Tag Message	No	Yes
Tagger Information	No separate tagger metadata	Can store tagger information
Basic Command	`git tag v1.0`	`git tag -a v1.0 -m "Version 1.0"`
Use	Simple version marking	Release information and detailed tagging

Result

Thus, a lightweight Git tag named `v1.0` was successfully created for a specific commit. The created tag was displayed using `git tag` and verified using `git show v1.0` and the Git commit history.

Viva Questions and Answers

1. **What is a tag in Git?**

   A tag is a reference used to identify a specific point in Git repository history.

2. **Why are tags used?**

   Tags are commonly used to identify important project versions and software releases.

3. **What is a lightweight tag?**

   A lightweight tag is a simple reference to a specific Git commit.

4. **Which command displays all tags?**

```
git tag
```

5. **How do you create a lightweight tag for the current commit?**

```
git tag v1.0
```

6. **How do you create a tag for a specific commit?**

```
git tag v1.0 <commit-id>
```

7. **How can we obtain a commit ID?**

   Execute:

```
git log --oneline
```

8. **How can we view the commit associated with v1.0?**

```
git show v1.0
```

9. **What is the difference between a branch and a tag?**

   A branch normally moves forward when new commits are created on that branch. A tag normally remains attached to the specific commit it references.

10. **What is the difference between lightweight and annotated tags?**

    A lightweight tag is a simple reference to a commit. An annotated tag is stored as a separate Git object and can contain additional metadata and a message.

11. **Can a tag be created for an older commit?**

    Yes. The required commit ID can be specified while creating the tag.

12. **What does v1.0 normally represent?**

    It commonly represents version 1.0 of a project or software release.

13. **Which command shows tags beside commits in Git log?**

```
git log --oneline --decorate
```

Conclusion

In this experiment, Git tagging was successfully demonstrated. A specific commit was identified using the commit history, and a lightweight tag named `v1.0` was created for that commit. The tag was then verified using Git tag, show, and log commands.

SEARCH CREATORS

VTU 2025 Scheme – 3rd Semester

`searchcreators.org`