VTU 2025 Scheme

Project Management (with Git)

1BCSL307A – 3rd Semester

EXPERIMENT – 01

Setting Up and Basic Commands

Aim

To initialize a new Git repository in a directory, create a new file, add it to the staging area, and commit the changes with an appropriate commit message.

Objectives

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

- Understand the basic Git workflow.
- Initialize a new Git repository.
- Create and manage files inside a Git repository.
- Add files to the Git staging area.
- Commit changes to the local repository.
- View the status and commit history of a Git repository.

Software Requirements

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

Theory

Git is a distributed version control system used to track changes in files and source code during software development.

Git allows developers to maintain different versions of a project and keeps a complete history of the changes made to the project.

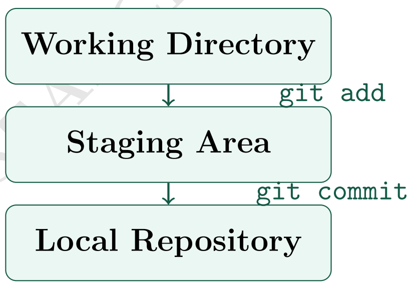
Advantages of Git

- Tracks changes made to project files.
- Maintains different versions of a project.
- Helps multiple developers work on the same project.
- Allows previous versions of files to be restored.
- Maintains a complete history of project development.
- Supports branching and merging.
- Works efficiently for both small and large projects.

Basic Git Workflow

Git mainly uses three areas:

1. **Working Directory**
   This is the location where the user creates and modifies project files.
2. **Staging Area**
   The staging area contains the changes that are selected for the next commit.
3. **Local Repository**
   The local repository stores committed versions of the project along with their history.



Procedure

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

Step 2: Create a new directory for the experiment.

Step 3: Move into the newly created directory.

Step 4: Initialize the directory as a Git repository.

Step 5: Create a new text file inside the repository.

Step 6: Add sample content to the file.

Step 7: Check the current status of the repository.

Step 8: Add the newly created file to the staging area.

Step 9: Verify that the file is successfully staged.

Step 10: Commit the staged file with an appropriate commit message.

Step 11: Display the commit history to verify the commit.

Execution

Step 1: Create a New Directory

Execute:

```
mkdir git-experiment-01
```

Explanation:
The `mkdir` command creates a new directory named `git-experiment-01`.

Step 2: Move into the Directory

Execute:

```
cd git-experiment-01
```

Explanation:
The `cd` command changes the current working directory to `git-experiment-01`.

Step 3: Initialize the Git Repository

Execute:

```
git init
```

Sample Output:

```
Initialized empty Git repository in
.../git-experiment-01/.git/
```

Explanation:
The `git init` command initializes the current directory as a new Git repository.
It creates a hidden directory named `.git`. This directory contains the information required by Git to manage the repository.

Step 4: Create a New File

For Windows Command Prompt, execute:

```
echo Hello Git > sample.txt
```

Explanation:
This command creates a new text file named `sample.txt`.
The file contains:

```
Hello Git
```

Step 5: Check Repository Status

Execute:

```
git status
```

Sample Output:

```
Untracked files:
  sample.txt
```

Explanation:
The file is shown as an **untracked file** because Git has not yet started tracking the file.

Step 6: Add the File to the Staging Area

Execute:

```
git add sample.txt
```

Explanation:
The `git add` command adds the file to the staging area.
The file is now ready to be included in the next commit.

Step 7: Verify the Staging Area

Execute:

```
git status
```

Sample Output:

```
Changes to be committed:
  new file: sample.txt
```

Explanation:
The output shows that `sample.txt` has been successfully added to the staging area.

Step 8: Commit the Changes

Execute:

```
git commit -m "Add sample file"
```

Explanation:
The `git commit` command stores the staged changes permanently in the local repository.
The `-m` option allows us to provide a commit message directly with the command.
Here,

```
"Add sample file"
```

is the commit message.
Note:
If Git asks for user identity before the first commit, 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 9: View Commit History

Execute:

```
git log --oneline
```

Sample Output:

```
a1b2c3d Add sample file
```

Explanation:
The `git log --oneline` command displays the commit history in a compact format. Here:

- `a1b2c3d` represents the shortened commit ID.
- `Add sample file` represents the commit message.

Complete Command Sequence

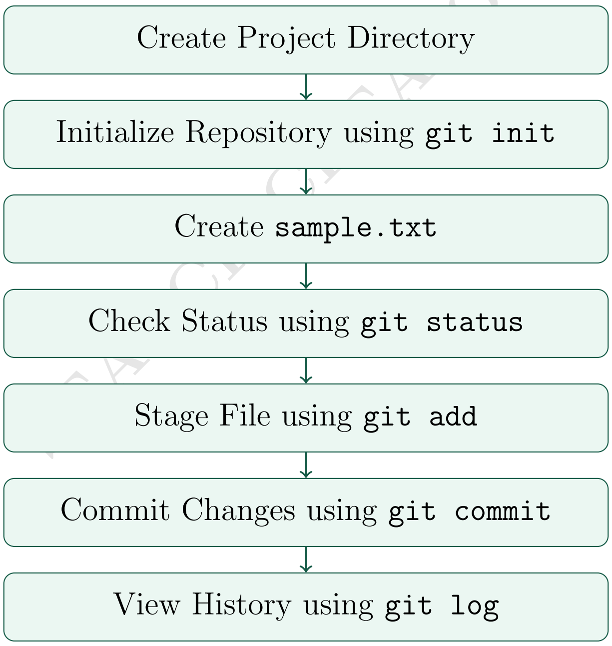
The complete command sequence for Experiment 01 is:

```
mkdir git-experiment-01

cd git-experiment-01

git init

echo Hello Git > sample.txt

git status

git add sample.txt

git status

git commit -m "Add sample file"

git log --oneline
```

Command Description

Command	Purpose
`mkdir git-experiment-01`	Creates a new directory for the experiment.
`cd git-experiment-01`	Changes the current working directory.
`git init`	Initializes a new Git repository.
`git status`	Displays the current status of the repository.
`git add sample.txt`	Adds `sample.txt` to the staging area.
`git commit -m "..."`	Creates a commit containing the staged changes.
`git log --oneline`	Displays the commit history in compact form.

Experiment Workflow



Result

Thus, a new Git repository was successfully initialized. A new file was created and added to the staging area. The changes were successfully committed to the local Git repository with an appropriate commit message.

Viva Questions and Answers

1. **What is Git?**

Git is a distributed version control system used to track changes in files and source code.

2. **What is a Git repository?**

   A Git repository is a storage location that contains project files along with their version history.

3. **What is the purpose of git init?**

   The `git init` command initializes a new Git repository.

4. **What is the staging area?**

   The staging area contains the changes selected for the next commit.

5. **What is the purpose of git add?**

   The `git add` command adds changes from the working directory to the staging area.

6. **What is a commit?**

   A commit is a saved snapshot of changes in a Git repository.

7. **What is the purpose of git status?**

   The `git status` command displays the current status of files in the working directory and staging area.

8. **What is the purpose of git log?**

   The `git log` command displays the commit history of the repository.

9. **What is an untracked file?**

   An untracked file is a file that exists in the working directory but has not yet been added to Git.

10. **What does the -m option represent in git commit?**

    The `-m` option is used to specify the commit message directly from the command line.

Conclusion

In this experiment, the basic Git workflow was successfully demonstrated. A repository was initialized, a new file was created, the file was moved from the working directory to the staging area, and finally the changes were committed to the local repository.