# Lab Report: Version Control with Git

## Experiment No.: 3

## Title:

Set Up a Git Repository and Perform Basic Version Control Operations (Commit, Push, Pull, Merge)

## Objective:

The objective of this experiment is to set up a Git repository and perform essential version control operations, including commit, push, pull, and merge. This will help understand how Git facilitates collaboration and version management in software development.

## Materials Required:

• Computer with Git installed
• Access to a Git hosting service (e.g., GitHub, GitLab, Bitbucket)
• Integrated Development Environment (IDE) or text editor (e.g., Visual Studio Code, Sublime Text)
• Command-line interface (CLI) or Git GUI (e.g., GitKraken, SourceTree)
• Sample project files or codebase

## Theory:

Git is a distributed version control system that enables multiple developers to work on a project simultaneously without overwriting each other's changes. It keeps track of changes made to the codebase and manages different versions of the project, allowing for collaboration and efficient development. Key operations in Git include:

• Commit: Save changes to the local repository.
• Push: Upload committed changes to a remote repository.
• Pull: Download changes from a remote repository to the local repository.
• Merge: Combine changes from different branches.

## Procedure:

### Set Up the Git Repository:

Initialize a Git repository in the local project directory using the command:

```
git init
```

Create a new repository on a Git hosting service (e.g., GitHub) and link it to the local repository:

```
git remote add origin <repository_url>
```

### Add Files and Make Initial Commit:

Add project files to the staging area:

```
git add .
```

Commit the changes with a descriptive message:

```
git commit -m "Initial commit"
```

### Push Changes to Remote Repository:

Push the committed changes to the remote repository:

```
git push origin main
```

Verify that the files have been uploaded by checking the remote repository on GitHub or another service.

### Perform a Pull Operation:

Make a change to the project files in the remote repository (e.g., edit a README file).

Pull the latest changes from the remote repository to the local repository:

```
git pull origin main
```

### Branching and Merging:

Create a new branch for a feature or bug fix:

```
git checkout -b feature-branch
```

Make changes to the files in the new branch and commit them:

```
git commit -m "Added new feature"
```

Switch back to the main branch and merge the feature branch into it:

```
git checkout main
```

```
git merge feature-branch
```

Resolve any merge conflicts if they arise and commit the merge:

```
git commit -m "Merged feature-branch into main"
```

### Push Merged Changes to Remote Repository:

Push the merged changes to the remote repository:

```
git push origin main
```

## Results:

• Successfully set up a Git repository and linked it to a remote repository.
• Performed basic Git operations: commit, push, pull, and merge.
• Documented the changes and observed how Git manages versions and facilitates collaboration.

## Discussion:

Version control is essential in software development, as it allows teams to track changes, revert to previous versions, and collaborate effectively. During this experiment, I encountered challenges such as resolving merge conflicts, which emphasized the importance of understanding Git's merging strategies. Git ensures that all team members work on the most recent version of the project, avoiding conflicts and enabling seamless collaboration.

## Conclusion:

This experiment demonstrated the benefits of using Git for version control in software projects. By performing core Git operations, I gained a deeper understanding of how Git maintains an organized and collaborative development process, ensuring that teams can work efficiently on complex projects.

## References:

• Pro Git by Scott Chacon and Ben Straub: https://git-scm.com/book/en/v2
• Git Documentation: https://git-scm.com/doc
• GitHub Guides: https://guides.github.com/

## Appendix:

• Screenshots: Include images showing the execution of Git commands and their outputs.
• Notes: Additional observations made during the experiment, such as specific issues encountered or tips for resolving merge conflicts.