**LAB 3: Git**

This lab exercise is designed to help students practice using Git in a real-world workflow. Students will learn how to manage code locally with Visual Studio Code (VS Code), push it to GitHub, and then clone or pull the code on a server. Both the terminal (command line) and the VS Code GUI Source Control panel will be used to reinforce understanding.

# Objectives

- Configure Git on a local machine.  
- Connect the local repository to GitHub.  
- Push code to GitHub.  
- Practice revert and reset operations.  
- Clone a repository on a server and pull updates when changes occur.  
- Use both VS Code terminal commands and the Source Control GUI.

## 1) Prerequisites

- A GitHub account with at least one empty repo you can push to.  
- VS code / Cursor installed on your local machine.  
- Ubuntu Server (22.04+ recommended), SSH access.  
- Git installed on both machines (`git --version`), or download from git-scm.com  
- Use SSH everywhere for fewer auth prompts and better security.

## 2) Local editor → GitHub

### 2.1 Create repository on Github

- On github.com click New to create new repository

- Copy url for the repository

https://github.com/<username>/<repo>.git OR [git@github.com:<username>/<repo>.git](mailto:git@github.com:%3cusername%3e/%3crepo%3e.git)

### 2.2 Configure Git identity (local)

### git config --global user.name "Your Name" git config --global user.email "you@example.com"

### Create an SSH key and add to agent (option)

ssh-keygen -t ed25519 -C "you@example.com"  
eval "$(ssh-agent -s)"  
ssh-add ~/.ssh/id\_ed25519

### Add public key to GitHub (option)

Add `~/.ssh/id\_ed25519.pub` to GitHub → Settings → SSH and GPG keys → New SSH key.

### 2.3 Create or connect project

Initialize repo in local and commit:

git init  
echo "# My Project" > README.md  
git add .  
git commit -m "chore: initial commit"

### 2.4 Add GitHub remote and push

git remote add origin git@github.com:<your-username>/<your-repo>.git  
git branch -M main  
git push -u origin main

git status

git add .

### 2.5 Revert & Reset

- Create new 3 files in local then commit each time you create the file

- Revert and reset the existing commit

| **GitLens (UI)** | **Git commands** | **Description** |
| --- | --- | --- |
| **Commit** (ใน Source Control / GitLens) | git add . + git commit -m "msg" | stage และ commit การเปลี่ยนแปลง |
| **Push** | git push | ส่ง commit ไป remote |
| **Pull** | git pull | ดึงการเปลี่ยนแปลงจาก remote |
| **Fetch** | git fetch | ดึงข้อมูล branch/commit ใหม่ ๆ แต่ยังไม่ merge |
| **Checkout Commit** | git checkout <commit-hash> | ไปที่ commit นั้น (detached HEAD) |
| **Checkout Branch** | git checkout <branch> | สลับไป branch อื่น |
| **Create Branch** | git branch <new-branch> + git checkout <new-branch> | สร้างและสลับไป branch ใหม่ |
| **Revert Commit** | git revert <commit-hash> | Undo commit โดยสร้าง commit ใหม่ |
| **Revert Current Branch to Commit** | git reset --hard <commit-hash> | ย้อน branch ไป commit นั้น (commit หลังจากนั้นหายไป) |
| **Revert Current Branch to Previous Commit** | git reset --hard HEAD~1 | ย้อน branch กลับไป 1 commit ก่อนหน้า |
| **Cherry-pick Commit** | git cherry-pick <commit-hash> | นำ commit จาก branch อื่นมาวางใน branch ปัจจุบัน |
| **Merge Branch** | git merge <branch> | รวม branch ที่เลือกเข้ามาใน branch ปัจจุบัน |
| **Stash Changes** | git stash | ซ่อนการแก้ไขชั่วคราว |
| **Apply Stash** | git stash apply | ดึงการแก้ไขที่ stash ไว้กลับมา |
| **Discard Changes (ไฟล์)** | git checkout -- <file> | ยกเลิกการแก้ไขไฟล์ กลับไปเป็นเวอร์ชันล่าสุดของ branch |
| **Show Commit Details** | git show <commit-hash> | แสดงรายละเอียด commit |

## 3) GitHub → Ubuntu Server

### 3.1 Generate SSH key (option)

ssh-keygen -t ed25519 -C "deploy@your.server"

### Add deploy key in GitHub (option)

Add server public key as a Deploy Key (read-only) in repo settings.

### 3.2 Clone repository on server

mkdir -p ~/apps && cd ~/apps  
git clone git@github.com:<your-username>/<your-repo>.git .

### 3.3 Clone repository on server

* Edit or add file on local then push to GitHub
* On server, use ‘git pull’ to update.

## 4) Install Nginx

If Nginx is not yet installed on your server, follow these commands.

1. Update the system's package list:

*sudo apt update*

1. Install Nginx:

*sudo apt install nginx*

1. Verify that Nginx is running:

*sudo systemctl status nginx*

The status should show as active (running).

1. Check configuration file

*sudo nginx -t*

*cat /etc/nginx/nginx.conf | grep include*

*grep -R "index" /etc/nginx/*

1. Open firewall port

*sudo ufw status*

*sudo ufw allow 80/tcp*

*sudo ufw allow 443/tcp*

## 5) Configure Nginx

You need to create a new configuration file for Nginx so it knows where to serve your frontend files from.

1. Create a new configuration file in sites-available:

sudo nano /etc/nginx/sites-available/<your\_domain\_name>

1. Copy the following code into the file, replacing <your\_domain\_name> with your actual domain.

Nginx

server {

listen 80;

server\_name \_;

root /var/www/<your\_domain\_name>;

index index.html;

location / {

try\_files $uri $uri/ /index.html;

}

}

* + root: Specifies the directory where your frontend files are located on the server.
  + try\_files: This is crucial for Vue Router, as it ensures that all requests that don't match a file are redirected to index.html.

## 6) Enable and Test Nginx

After creating the configuration file, you need to enable it and ensure it's free of errors.

1. Create a symbolic link to enable the new configuration:

sudo ln -s /etc/nginx/sites-available/<your\_domain\_name> /etc/nginx/sites-enabled/

1. Test the Nginx configuration for syntax errors:

sudo nginx -t

If successful, you will see a message confirming that the syntax is ok and the test is successful.

1. Restart Nginx for the changes to take effect:

sudo systemctl restart nginx