Lab 01

Development Environment Setup & Git Basics

1. Learning Outcomes

By the end of this lab, students will be able to:

- Install and configure Visual Studio Code (VS Code) and Git.
- Create and verify a GitHub account.
- Initialize a local Git repository and connect it to a remote repository.
- Perform version control operations (add, commit, push, pull).
- Create and manage branches for parallel development.
- Document their setup and workflows systematically.

2. Background

- **Integrated Development Environment (IDE)**

An IDE provides tools for writing, debugging, and managing code. **Visual Studio Code** is one of the most popular IDEs in web development due to its lightweight design and rich extensions.

- **Git**

A **distributed version control system (VCS)** created by Linus Torvalds. Git enables collaboration by tracking file changes across multiple developers. Each repository copy has the complete history.

- **GitHub**

A **cloud-based hosting service** for Git repositories. It enables collaboration, issue tracking, project management, and open-source contributions.

3. Git Workflow

1. **Create/Clone** a repository
2. **Stage changes** → git add
3. **Commit changes** → git commit
4. **Push to GitHub** → git push
5. **Pull updates** → git pull

6. **Branch & Merge** for collaborative workflows

4. Pre-Lab Requirements

- Read Git basics from https://git-scm.com/book/en/v2
- Install Google Chrome (recommended browser).
- Have a working internet connection.
- Ensure administrator rights to install new software.
- Have a **GitHub account** created before class.

5. Lab Exercises

Part A: Development Environment Setup

Exercise 1: Install and Configure VS Code

1. Download from https://code.visualstudio.com/
2. Install with default settings.
3. Launch and install extensions:
   - Live Server
   - Prettier (Code Formatter)
   - ES7+ React Snippets

Exercise 2: Install and Verify Git

1. Download from https://git-scm.com/downloads
2. Install with default settings.
3. Open Terminal/Command Prompt and verify:

   git –version

Part B: GitHub Setup

Exercise 3: Create a GitHub Account and Repository

1. Sign up at https://github.com
2. Create a new repository named web-tech-labs.
   - Public visibility

- Initialize with README

Exercise 4: Configure Git Identity

Run the following commands in VS Code terminal:

- *git config --global user.name "Your Full Name"*
- *git config --global user.email "24pwbcs0000@uetpeshawar.edu.pk"*
- *git config –list*

Part C: Local Git Repository

Exercise 5: Create a Git Repository

- Navigate to the directory where you want to create your Git repository.
- Run the following commands to create and initialize a Git repository.
  - mkdir 24PWBCS000-WT-Fall25 (replace with your registration number)
  - cd 24PWBCS000-WT-Fall25
  - *git init*

Exercise 6: Create a File and Commit Changes

- Create a new file in your project directory using a text editor or terminal commands.
- Add content to the file.
- Use the following commands to stage and commit changes:
  - *git add filename*
  - *git commit -m "Initial Commit"*

Part D: Connect Local & Remote

Exercise 7: Upload to GitHub

- Go to GitHub and log in to your account.
- Select the repository.
- Note down the repository URL (HTTPS or SSH) provided on the repository's page.

Exercise 8: Use the git remote Command:

- To add a remote repository, use the git remote add command. You can name the remote repository (commonly named "origin") and provide the GitHub repository URL you noted down earlier.
- **For HTTPS:**
  - *git remote add origin https://github.com/yourusername/your-repo.git*
- **For SSH:**
  - *git remote add origin git@github.com:yourusername/your-repo.git*

Exercise 9: Verify the Remote Repository:

- Use the git remote -v command to verify that the remote repository has been added correctly. It should list the "origin" remote with the GitHub URL.
  - *git remote -v*

Exercise 10: Push Your Local Repository to GitHub:

- To push your local repository to GitHub, use the git push command with the -u flag to set up tracking. This is typically done for the main branch (e.g., "main" or "master").
  - *git push -u origin main*

Exercise 11: Updating Your Local Copy:

- Use git pull to fetch and merge the latest changes from the remote repository into your local branch.
  - *git pull*

Exercise 12: Updating from a Specific Branch

- You can specify the branch you want to update from using git pull origin branch-name.
  - *git pull origin main*

Exercise 13: Cloning a Repository

- Cloning means creating a local copy of a remote Git repository
  - Use git clone repository-url to clone a repository.

Part E: Branching in Git

- Git's branch system allows for parallel development and isolation of features or bug fixes.
- Branches are lightweight pointers to specific commits in your project's history.

Main Branch

- Typically named "main" or "master."
- Represents the production-ready code.
- Most stable and tested version of your project.

Exercise 13: Creating a New Branch

- Use the 'git branch' command to create a new branch.
    - *git branch New-branch*

Exercise 14: Switching Between Branches

- Use the 'git checkout' command to switch between branches.
    - *git checkout new-branch*

Exercise 15: Creating and Switching in One Step

- Use the '-b' flag with 'git checkout' to create and switch to a new branch in one step.
    - *git checkout -b new-branch2*

Viewing Branches

- Use the 'git branch' command to list all branches in your repository.
    - *git branch*

Exercise 16: Merging Branches

- Use the 'git merge' command to combine changes from one branch into another.
    - *git checkout main*
    - *git merge new-branch*

Exercise 17: Deleting a Local Branch

- Use the '-d' flag with 'git branch' to delete a branch.
    - *git branch -d new-branch*

Exercise 18: Deleting a Remote Branch

- *git push origin --delete new-branch*

Lab Tasks:

Task 1: Creating an HTML File

Begin by creating a new file named index.html inside your Lab 01 folder. In this file, write a basic HTML structure that includes the course title. The body of the page should contain a single line of

<h1>Welcome to 224 Web Technologies Course</h1>

Once you have written the file, save it, and then commit your changes to the repository with a meaningful commit message such as *"Added index.html with welcome message"*. Finally, push your changes to the remote repository.

Task 2: Updating the HTML File with Comments

Reopen the index.html file and insert an HTML comment above the heading, for example:

<!-- This is my first HTML page for Lab 01 -->

Save your changes, commit them with a clear message such as *"Updated index.html with HTML comment"*, and push the updated file to your GitHub repository.

Task 3 – Creating a .gitignore File

Inside your Lab 01 folder, create a file named .gitignore. In this file, specify that all .log files and the temp/ directory should be ignored by Git. Save this file, then add, commit, and push it to your repository with the message *"Added .gitignore file"*.

Task 4 – Adding a License File

Next, create a new text file named LICENSE in your project root. In this file, write the statement: *"This project is for educational purposes only."* Save the file, then commit and push it with the message *"Added LICENSE file"*.

Task 5 – Creating and Deleting a File

Create a file called test.txt and add any text inside it. Save the file and commit it with a message describing the addition. After this, delete the test.txt file, commit the deletion with another message, and push both changes to the repository. This demonstrates file tracking and removal in Git.

Task 6 – Working with Branches

Create a new branch called experiment and switch to it. On this branch, open the index.html file and add a new heading below the first one, for example:

<h2>This is an experimental change</h2>

Commit this change on the branch and push the branch to GitHub.

Task 7 – Merging Branches

Switch back to the main branch of your repository. Merge the experiment branch into the main branch so that the changes you made are now part of the main project. Commit and push the updated main branch to GitHub.

Task 8 – Writing a Markdown Report

Create a new file named report.md. In this file, write a short reflection (at least 3–5 sentences) about what you learned during this lab. Discuss your experience with Git, creating files, and working with branches. Save the file, commit it, and push it to your repository.

Task 9 – Adding a Screenshot

Take a screenshot showing either your Git version in the terminal or your Visual Studio Code setup. Create a folder named screenshots in your Lab 01 directory, place the screenshot inside it, and commit and push the folder and file to GitHub.

Task 10 – Cloning the Repository to a New Location

Finally, clone your GitHub repository into a new folder on your computer (different from your original working directory). After successfully cloning, take a screenshot of the folder structure to show that the repository has been cloned. Save the screenshot in the screenshots folder of your original repository, commit the new screenshot, and push it to GitHub.