Lab: Development Tools - Git and SSH Key Integration

This lab will guide you through setting up an SSH key, integrating it with GitHub, creating and managing a Python project repository, and performing basic Git operations such as branching, committing, and pushing changes.

Objectives:

By the end of this lab, you will be able to:
1. Create an SSH key on an Ubuntu virtual machine (VM).
2. Copy the SSH key to your GitHub account.
3. Create a GitHub repository and clone it onto your VM.
4. Set up a basic folder structure for a Python project.
5. Perform Git operations: committing as master, branching, and managing versions.

Steps:

Step 1: Create an SSH Key

1. Open the terminal on your Ubuntu VM.

1. Generate an SSH key
ssh-keygen -t rsa -b 4096 -C "60107788@udst.edu.sa"

2. Press Enter to accept the default file location and provide a passphrase (optional).
Enter

3. Display the generated public key:

cat ~/.ssh/id_rsa.pub

Copy the entire output to your clipboard.

Step 2: Add the SSH Key to GitHub

1. Log in to your GitHub account.

2. Navigate to Settings > SSH and GPG Keys > New SSH Key.

Click on your profile picture (top-right corner) and select Settings**.**
In the left-hand menu, click SSH and GPG keys**.**

3. Provide a title (e.g., "Ubuntu VM") and paste the copied SSH key into the "Key" field.

4. Click Add SSH Key.

Step 3: Create a GitHub Repository

1. On GitHub, click on the + icon in the top-right corner and select New Repository.
2. Provide a repository name (e.g., `python_project_lab`) and an optional description.
3. Select Public or Private, and check Add a README file.
4. Click Create Repository.

Step 4: Clone the Repository on Your VM

1. Copy the repository's SSH URL from GitHub.
2. On your Ubuntu VM, run:

   git clone "copied ss hurl from GitHub)"

3. Navigate to the cloned directory:
   ls

   git branch
   The `git branch` command is used to manage branches in a Git repository. Branches allow you to work on different parts of a project in isolation, making it easier to manage features, bug fixes, or experiments without affecting the main codebase.
   - Lists all local branches in the repository (local).
   - The current branch is highlighted with an asterisk (*).

   git branch -a
   Lists all branches, including both local and remote.

4. Adding a readme file

   nano README.md
   ls
   gid add README.md
   ls
   git status **(**What did we do so far)
   Note: the readme file is only locally on your machine. Its not online on GitHub.
   What shall we do?
   git commit -m "Adding the readme file."

git push -u origin main (is used to upload local changes in the `main` branch of your Git repository to the remote repository named `origin`.)

5. Now let's update the file online and then pull the update locally.
   - We modify the readme file on GitHub
     git pull origin main **(**is used to fetch the latest changes from the remote repository's `main` branch and merge them into your local `main` branch.)

Note: cat README.md (to read only the file) (md. Markdown mode)

Step 5: Clone Create a new branch
git checkout **(**get out from the current branch)

git checkout -b lab1 (change from the current branch and create a new one here "lab1")

git branch -a **(**to check that the new branch has been indeed added)

Step 6: Set Up a Pyt hon Project Folder Structure

1. Create the following folder structure

```
python_project_lab/
        ├── src/ # Main source code directory
        │   ├── __init__.py # Marks the directory as a Python package
        │   ├── data_loader.py
        │   ├── preprocessing.py
        │   ├── model.py
        │   └── evaluation.py
        ├── tests/ # Test files
        │   ├── __init__.py
        │   ├── test_data_loader.py
        │   ├── test_preprocessing.py
        │   └── test_model.py
        ├── README.md # Project documentation
        └── requirements.txt # List of dependencies
```

→ You need to create all folders/files (mkdir, touch)

Note: to create multiple empty files:
touch file_name1 file_name2 file_name3

Note: you need to install tree to be able to visualize the above structure.

sudo snap install tree

→ **tree**

2. Create a basic `README.md` file

3. Step 6: Commit Initial Changes as Master
    a. Stage the changes:

    git add .

    is used in Git to stage **all changes** (new files, modified files, and deleted files) in the current directory and its subdirectories for the next commit.)

    b. Commit the changes:

    commit -m "Creating a Python project"

    c. Push to the `master` branch:

    git push origin lab1

Step 7: Create a New Branch

1. Create and switch to a new branch (e.g., `feature-branch`)

    git checkout -b feature-branch

2. Step 8: Create First Code Version (V1)
    a. Add a simple Python script in the `src` folder (e.g., `src/main.py`):
    b. Stage and commit the changes:
    c. Push to GitHub:

Step 9: Edit Code and Commit as V2

1. Edit the `src/main.py` file to include a new feature
2. Stage and commit the changes:

Step 10: Create a New Branch for Further Development

3.
    1. Create a new branch (e.g., `experimental-branch`):

    2. Push the branch to GitHub: