GIT PRACTICE LAB GUIDE

LAB OVERVIEW

Welcome to the Git Practice Lab! 🎉 In this lab, you'll discover how to use Git, a powerful tool for version control. Git enables you to track changes in your code, collaborate with others, and manage your projects effectively. By the end of this lab, you'll be able to set up Git on your computer, create a Git repository, and make your first commit. Additionally, you will learn how to track changes, inspect history, and manage your workflow like a pro.

This lab is designed to be hands-on and interactive, allowing you to engage directly with the tools and processes that Git offers. You will follow a series of steps that will guide you through the initial setup of Git, creating your first repository, and committing changes to your project. Each section contains clear instructions and tips to help you succeed.

If you have any questions along the way, feel free to ask! The goal is to ensure that you feel confident in using Git by the end of this lab. So, let's get started and dive into the world of version control with Git!

SECTION 1: SETTING UP GIT AND CREATING A REPOSITORY

To begin your journey with Git, the first step is to ensure that Git is installed on your computer. Follow these instructions carefully.

STEP 1: CHECK IF GIT IS INSTALLED

Open your terminal (or command prompt) and type the following command:

```
git --version
```

If you see a version number displayed (e.g., `git version 2.37.1`), you're all set! If not, you will need to install Git. The installation process varies based on your operating system:

- **Windows:** Download Git from git-scm.com and follow the installation instructions.
- **Mac:** Open your terminal and run:
- **Linux:** Execute this command in your terminal:

After installation, confirm that Git is successfully installed by typing the command again:

```
git --version
```

STEP 2: CONFIGURE GIT

Now that Git is installed, you'll need to configure it with your name and email. This information will be used to sign your commits. Run the following commands in your terminal:

```
git config --global user.name "Your Name"
git config --global user.email "your.email@example.com"
```

Make sure to replace `"Your Name"` and `"your.email@example.com"` with your actual name and email address.

STEP 3: CREATE YOUR FIRST REPOSITORY

Let's get hands-on by creating your first Git repository. Start by creating a new folder for your project:

```
mkdir my-first-repo
cd my-first-repo
```

Then, initialize a Git repository in this folder by running:

```
git init
```

You should see a message indicating that an empty Git repository has been initialized.

STEP 4: CHECK THE STATUS OF YOUR REPOSITORY

To see the status of your new repository, type:

```
git status
```

This command will show you the current state of your repository. As part of your lab submission, take a screenshot of this output and save it as `git-status.png`. This will serve as proof of your initial setup.

SECTION 2: STAGING AND COMMITTING CHANGES

In this section, you will learn how to stage files and make your first commit in Git. Staging changes is an essential part of using Git, as it allows you to prepare specific changes to be included in the next commit. Let's get started!

STEP 1: CREATE AND STAGE FILES

To begin, you will create a `README.md` file, which is a common practice in Git repositories. This file generally contains information about your project. Open your terminal and run the following command to create the file:

```
echo "# My First Git Project" > README.md
```

After creating the file, you need to stage it for commit. Use the following command:

```
git add README.md
```

To verify that the file is staged correctly, check the status of your repository:

```
git status
```

You should see that `README.md` is ready to be committed.

STEP 2: MAKE YOUR FIRST COMMIT

Now that you have staged your file, it's time to make your first commit! This step captures the current state of your project. Run the command below, providing a clear and descriptive message about your changes:

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

It is important to have clear commit messages as they help you and others understand the history of changes made to the project.

STEP 3: CREATE A .GITIGNORE FILE

Sometimes, you may want to exclude certain files from being tracked by Git, such as temporary files or logs. To do this, you can create a `.gitignore` file. Use the following command to create it:

```
printf '*.\n' log pdf > .gitignore
```

After creating the `.gitignore` file, stage and commit it using:

```
git add .gitignore
git commit -m "Add .gitignore file"
```

STEP 4: CREATE MORE FILES

You can further enhance your project by adding additional files. For example, create a simple Python script with the following command:

```
echo "print('Hello, Git!')" > script.py
```

Stage the new files and commit your changes:

```
git add README.md script.py
git commit -m "Add script.py and update README"
```

Finally, as part of your lab requirements, take a screenshot of your commit history and save it as `git-log.png`. Use the command below to view the commit log:

```
git log --oneline
```

SECTION 3: INSPECTING HISTORY AND MANAGING CHANGES

In this section, you will learn how to inspect your commit history and manage changes effectively using Git. Understanding how to navigate your project's history is crucial for tracking progress and identifying any issues that may arise.

STEP 1: INSPECT COMMIT HISTORY

To review all the commits made in your repository, you can use the `git log` command. This command provides detailed information about each commit, including the commit hash, author, date, and commit message. For a more compact view, you can type:

```
git log --oneline
```

This command will display a simplified list of commits, making it easier to see the history at a glance. Take a moment to familiarize yourself with the output, as it will help you understand how your project has evolved.

STEP 2: VIEW CHANGES

Next, let’s explore how Git tracks changes in your files. Open your `script.py` file and add a new line by running the following command:

```
echo "print('Welcome to Git!')" >> script.py
```

Now, you can check the changes you've made using the `git diff` command:

```
git diff
```

This command shows you the differences between your working directory and the last commit. After reviewing the changes, stage them by running:

```
git add script.py
```

To see the changes that are staged for the next commit, use:

```
git diff --staged
```

This allows you to confirm that the changes you want to include are ready for the next commit.

STEP 3: AMEND THE LAST COMMIT

If you realize that you need to correct something in your last commit, Git allows you to amend it. Make another small change to `script.py`:

```
echo "print('This is fun!')" >> script.py
```

After making your change, stage it:

```
git add script.py
```

Then, to amend the last commit, run:

```
git commit --amend
```

You will have the opportunity to update the commit message. Consider using something descriptive, such as "Update script.py with new feature." Finally, check your commit history again using:

```
git log --oneline
```

As part of your lab submission, take a screenshot of this amended commit history and save it as `git-log-amended.png`. This will demonstrate your ability to manage commits effectively within your Git workflow.

FINAL SUBMISSION INSTRUCTIONS

To complete the lab, it is essential to submit your work in a professional manner. Please follow these steps to ensure your submission meets the required standards:

Create a Folder: Begin by creating a new folder named `Git-Lab-Submission`. This folder will contain all your submission files.

Include Required Files: Inside the `Git-Lab-Submission` folder, make sure to include the following files:

- `git-status.png` (from Section 1)
- `git-log.png` (from Section 2)
- `git-log-amended.png` (from Section 3)

Compress the Folder: Once you have placed all required files in the folder, compress it into a .zip file. Name the zip file `Git-Lab-Submission.zip`.

Submit According to Instructions: Finally, submit the `Git-Lab-Submission.zip` file as directed.

Congratulations on completing the Git Practice Lab! 🎉 You have gained valuable skills in using Git for version control, and now you can confidently manage your projects. If you have any questions or need further clarification, please don't hesitate to reach out for assistance.

```
sudo apt-get install git
```

```
brew install git
```