Activity No. 14	
SSH Key-Based Authentication and GIT Setup	
Name: Juan Paulo C. Lara	**Date Performed: 11/11/2025**
Course Code: CPE 201A	**Date Submitted: 11/11/2026**
Course Title: Computer System Administration and Troubleshooting	**Instructor: Engr. Jimlord M. Quejado**

1. Objective/s:

This activity aims to demonstrate students' ability to configure secure SSH key-based authentication and perform version control operations using Git and GitHub.

2. Intended Learning Outcome/s:

By the end of this activity, the students should be able to:

- Analyze how SSH key-based authentication provides secure access.
- Evaluate the setup of SSH and Git configuration.
- Create and manage a Git repository using SSH connection.

3. Discussion:

Part 1: Discussion
It is assumed that you are already done with the last Activity (**Laboratory Activity 9 | Install Linux in a Virtual Machine and Explore the GUI**).
Provide screenshots for each task.

It is also assumed that you have VMs running that you can SSH but require a password. Our goal is to remotely login through SSH using a key without using a password. In this activity, we create a public and a private key. The private key resides in the local machine while the public key will be pushed to remote machines. Thus, instead of using a password, the local machine can connect automatically using SSH through an authorized key.

What Is ssh-keygen?
Ssh-keygen is a tool for creating new authentication key pairs for SSH. Such key pairs are used for automating logins, single sign-on, and for authenticating hosts.

SSH Keys and Public Key Authentication
The SSH protocol uses public key cryptography for authenticating hosts and users. The authentication keys, called SSH keys, are created using the keygen program.

SSH introduced public key authentication as a more secure alternative to the older .rhosts authentication. It improved security by avoiding the need to have passwords stored in files and eliminated the possibility of a compromised server stealing the user's password.

However, SSH keys are authentication credentials just like passwords. Thus, they must be managed somewhat analogously to usernames and passwords. They should have a proper termination process so that keys are removed when no longer needed.

Part 2: Discussion
Provide screenshots for each task.

Set up Git
At the heart of GitHub is an open-source version control system (VCS) called Git. Git is responsible for everything GitHub-related that happens locally on your computer. To use Git on the command line, you'll need to download, install, and configure Git on your computer. You can also install GitHub CLI to use GitHub from the command line. If you don't need to work with files locally, GitHub lets you complete many Git-related actions directly in the browser, including:

- Creating a repository
- Forking a repository
- Managing files
- Being social

4. Procedures:

Task 1: Create an SSH Key Pair for User Authentication

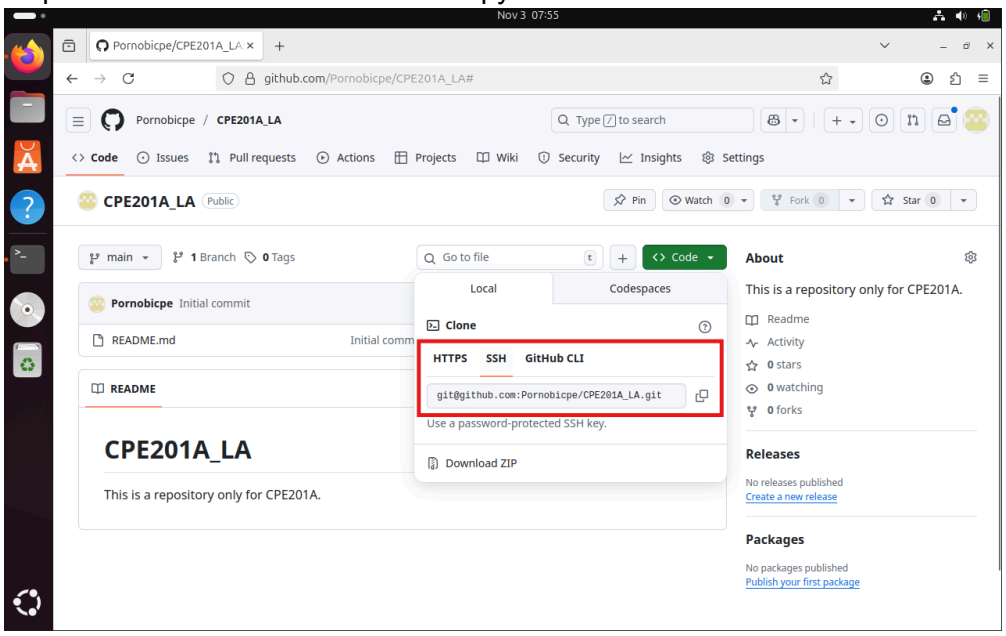
1. Open VirtualBox and start your Ubuntu virtual machine.
2. Log in using your username and password.
3. Open the Terminal.
4. Generate an SSH key pair by typing the following command and pressing Enter:
   ssh-keygen
5. Navigate to the SSH directory:
   cd ~/.ssh
6. List the files in the directory:
   ls
   Look for a file ending with .pub this is your public key.
7. Display the contents of your public key file (replace id_rsa.pub with your actual filename if different):
   cat id_rsa.pub
8. Copy the entire output: this is your SSH public key, which you can use for authentication.

Task 2: Copying the Public Key to Remote Servers

1. Open your GitHub account in a web browser.
2. Click on your profile icon (upper-right corner) and go to Settings.
3. In the left sidebar, select SSH and GPG keys.
4. If there is an existing SSH key, you may delete it first.
5. Click the “New SSH key” button.
6. Enter CPE201A as the Title.
7. In the Key field, paste the SSH public key that you copied from the terminal in Task 1.
8. Click “Add SSH key” to save your new key.

Task 3: Set up the Git Repository

1. On the local machine, verify the version of your git using the command which git. If a directory of git is displayed, then you don't need to install git. Otherwise, to install git, use the following command: sudo apt install git
2. After the installation, issue the command which git again. The directory of git is usually installed in this location: user/bin/git.
3. The version of git installed in your device is the latest. Try issuing the command git --version to know the version installed.
4. Using the browser in the local machine, go to www.github.com.
5. Sign up in case you don't have an account yet. Otherwise, login to your GitHub account.
    a. Create a new repository and name it as CPE201A_yourname, and add description "This repository is only for CPE201A". Check Add a README file and click Create repository.
    b. Clone the repository that you created. In doing this, you need to get the link from GitHub. Browse to your repository as shown below. Click on the Code drop down menu. Select SSH and copy the link.



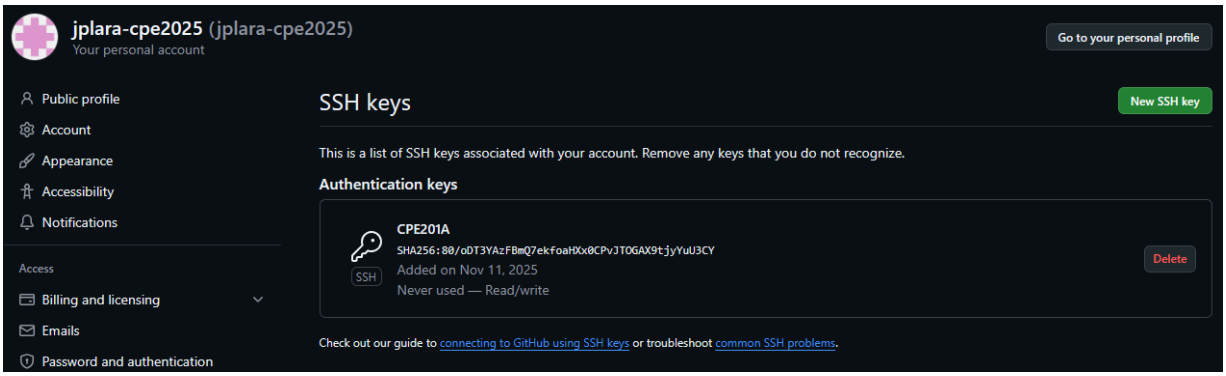
    c. Issue the command git clone followed by the copied link. For example, git clone git@github.com:Pornobicpe/CPE201A_yourname.git. When prompted to continue connecting, type yes and press enter.
    d. To verify that you have cloned the GitHub repository, issue the command ls. Observe that you have the CPE201A_yourname in the list of your directories. Use CD command to go to that directory and LS command to see the file README.md.
    e. Use the following commands to personalize your git.
        - git config --global user.name "Your Name"
        - git config --global user.email yourname@email.com
        - Verify that you have personalized the config file using the command cat ~/.gitconfig
    f. Edit the README.md file using nano command. Provide any information on the markdown file pertaining to the repository you created. Make sure to write out or save the file and exit.

g. Use the git status command to display the state of the working directory and the staging area. This command shows which changes have been staged, which haven't, and which files aren't being tracked by Git. Status output does not show any information regarding the committed project history. What is the result of issuing this command?
Answer: It checks if there is any file saved to Git and if they are committed to the repository.
h. Use the command git add README.md to add the file into the staging area.
i. Use the git commit -m "your message" to create a snapshot of the staged changes along the timeline of the Git projects history. The use of this command is required to select the changes that will be staged for the next commit.
j. Use the command git push <remote><branch> to upload the local repository content to GitHub repository. Pushing means to transfer commits from the local repository to the remote repository. As an example, you may issue git push origin main.
k. On the GitHub repository, verify that the changes have been made to README.md by refreshing the page. Describe the README.md file. You can notice how long was the last commit. It should be some minutes ago and the message you typed on the git commit command should be there. Also, the README.md file should have been edited according to the text you wrote.
Answer: The README.md (markdown) file is the first page users see when visiting a GitHub repository. It shows information about the project, updates, author, and guides to help other users contribute to the repo.

5. Outputs:

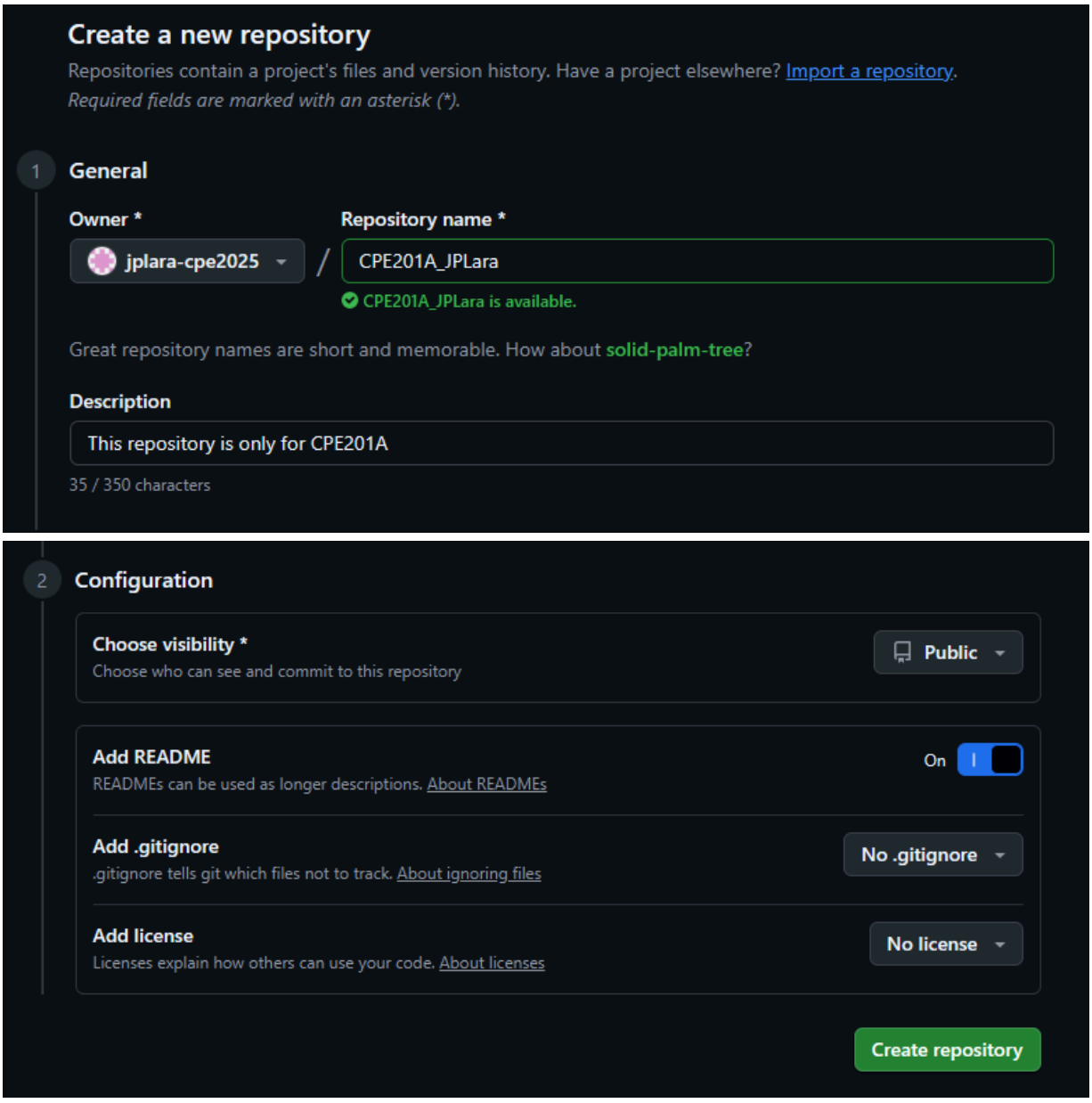
```
qjplara@qjplara:~/Desktop$ ssh-keygen
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/qjplara/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/qjplara/.ssh/id_ed25519
Your public key has been saved in /home/qjplara/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:80/oDT3YAzFBmQ7ekfoaHXx0CPvJTOGAX9tjyYuU3CY qjplara@qjplara
The key's randomart image is:
+--[ED25519 256]--+
|         o==..   |
|        o ==+..  |
|       . B+=oB . |
|        o *BE.O  |
|        So.+== o |
|        .oo*. .  |
|         o= *    |
|        .. = o   |
|          . o    |
+----[SHA256]-----+
qjplara@qjplara:~/Desktop$
```

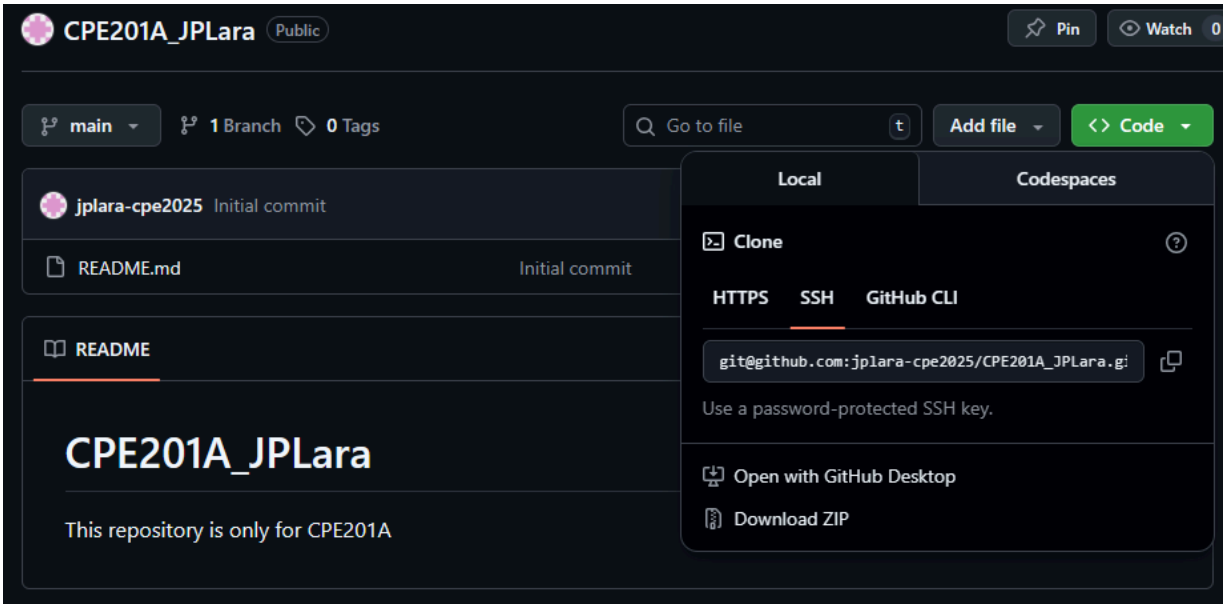
```
qjplara@qjplara:~/Desktop$ cd ~/.ssh
qjplara@qjplara:~/.ssh$ ls
authorized_keys  id_ed25519  id_ed25519.pub
qjplara@qjplara:~/.ssh$ cat id_ed25519.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIOBiLWgea9ZtKLr1H/oK5iopyn8mONWlHk5jW8DcAh4B
qjplara@qjplara
qjplara@qjplara:~/.ssh$
```



```
qjplara@qjplara:~/.ssh$ which git
/usr/bin/git
qjplara@qjplara:~/.ssh$
```

```
qjplara@qjplara:~/.ssh$ git --version
git version 2.43.0
qjplara@qjplara:~/.ssh$
```






```
qjplara@qjplara:~/.ssh$ git clone git@github.com:jplara-cpe2025/CPE201A_JPLara.gi
t
Cloning into 'CPE201A_JPLara'...
The authenticity of host 'github.com (20.205.243.166)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvCOqU.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
qjplara@qjplara:~/.ssh$
```

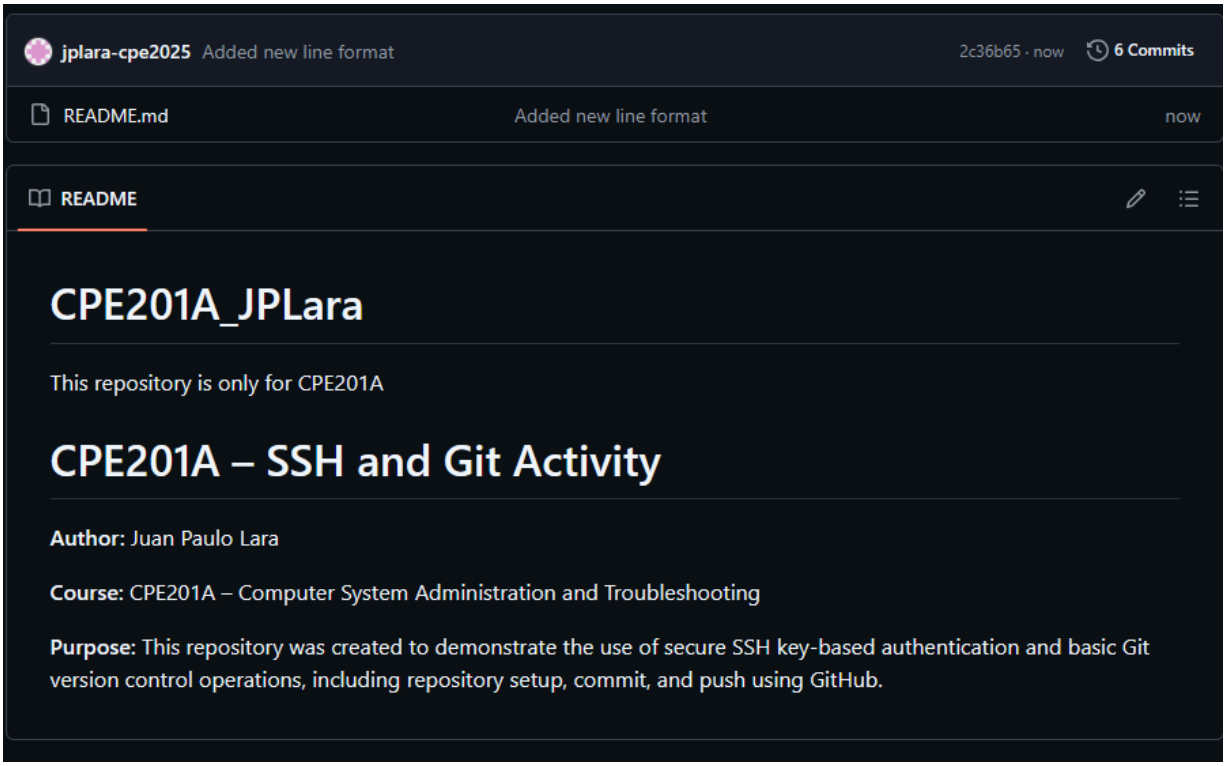
```
qjplara@qjplara:~/.ssh$ cd CPE201A_JPLara
qjplara@qjplara:~/.ssh/CPE201A_JPLara$ ls
README.md
```

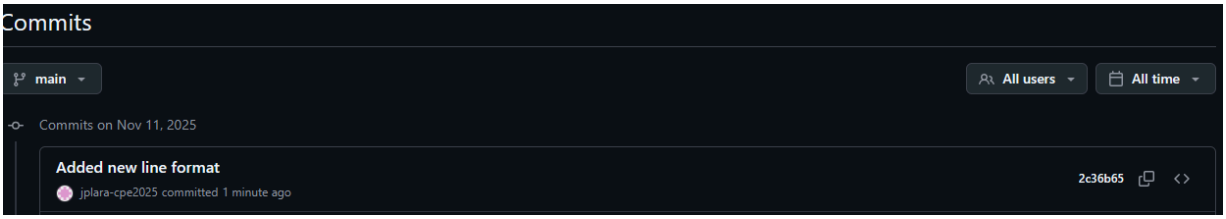
```
qjplara@qjplara:~/.ssh/CPE201A_JPLara$ git config --global user.email qjplara25@
tip.edu.ph
qjplara@qjplara:~/.ssh/CPE201A_JPLara$ git config --global user.name jplara-cpe2
025
qjplara@qjplara:~/.ssh/CPE201A_JPLara$ cat ~/.gitconfig
[user]
        name = jplara-cpe2025
        email = qjplara25@tip.edu.ph
qjplara@qjplara:~/.ssh/CPE201A_JPLara$
```

```
qjplara@qjplara:~/.ssh/CPE201A_JPLara$ nano README.md
qjplara@qjplara:~/.ssh/CPE201A_JPLara$ git status
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
qjplara@qjplara:~/.ssh/CPE201A_JPLara$ git add README.md
qjplara@qjplara:~/.ssh/CPE201A_JPLara$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   README.md

qjplara@qjplara:~/.ssh/CPE201A_JPLara$ git commit -m "Added new line formatting"
[main 533c097] Added new line formatting
qjplara@qjplara:~/.ssh/CPE201A_JPLara$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 290 bytes | 290.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:jplara-cpe2025/CPE201A_JPLara.git
   351d44e..533c097  main -> main
```





6. Conclusions/Learnings/Analysis:

This activity has successfully demonstrated the SSH Key-Based Authentication and GIT Setup through the use of a Linux terminal and a GitHub repository with the appropriate

commands and settings. The SSH key allowed for a more secure authentication and access without requiring a password stored in a file, reducing security vulnerabilities as these keys are created by the computer instead of a password, which is created by the user. After completing the given tasks, the output screenshots show a Git repository containing the name of the course, a README.md file containing the information about the repo, and a commit history showing the changes of the file. These changes involving stage, commit, and push, were done through the Linux terminal logged in through SSH. By familiarizing SSH and CLI commands, a user can implement changes to a repository without needing an internet browser or even a GUI running. Overall, this activity showed the importance of SSH key-based authentication and Git in creating a secure and reliable development workflow for developers and a transparent version control system to specify bugs and issues.

7. Assessment Rubric: