Activity No. 14	
SSH Key-Based Authentication and GIT Setup	
Name: Mendoza Nathaniel B.	**Date Performed: 11 / 9 / 2025**
Course Code: CPE 201A	**Date Submitted: 11 / 9 / 2025**
Course Title: Computer System Administration and Troubleshooting	**Instructor: Engr. Jimlord 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:

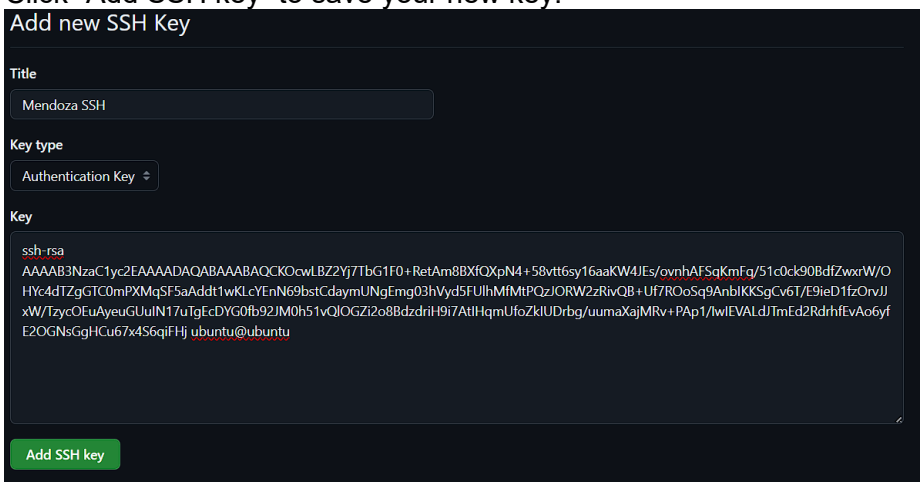
Task 1: Create an SSH Key Pair for User Authentication

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.

```
ubuntu@ubuntu:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ubuntu/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ubuntu/.ssh/id_rsa.
Your public key has been saved in /home/ubuntu/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:hBu/mBbW0yQKqxdnHrRRf37LmNE5LRprq+Fs4nih31w ubuntu@ubuntu
The key's randomart image is:
+---[RSA 2048]----+
|        .        |
|       o .       |
|    . = o o .    |
|     + X + o . o |
|    o X S . + * .|
|   . * =.o   X + |
|  . . =.... E o  |
|   . ...o=.+ .   |
|      .+oo*..    |
+----[SHA256]-----+
ubuntu@ubuntu:~$ cd ~/.ssh
ubuntu@ubuntu:~/.ssh$ ls
id_rsa  id_rsa.pub
ubuntu@ubuntu:~/.ssh$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCkOcwlBZ2Yj7TbG1F0+RetAm8BXfQXpN4+58vtt6s
y16aaKW4JEs/ovnhAFSqKmFq/51cOcK90BdfZwxrW/OHYc4dTZgGTCOmPXMQsF5aAddt1wKLcYEnN69
bstCdaymUNgEmg03hVyd5FUlhMfMtPQzJORW2zRivQB+UF7ROo5q9AnbIKKSgCv6T/E9ieD1fzOrvJJ
xW/TzycOeuAyeuGUuIN17u7tgEcDYG0fb92JMOh51vQIoGZi2o8BdzdriH9i7AtIHqmUfoZkIUDrbG/
uumaXajMRv+PAp1/IwIEVAldJTmEd2RdrhfEvAo6yfE20GNsGgHCu67x4S6qiFHj ubuntu@ubuntu
ubuntu@ubuntu:~/.ssh$
```

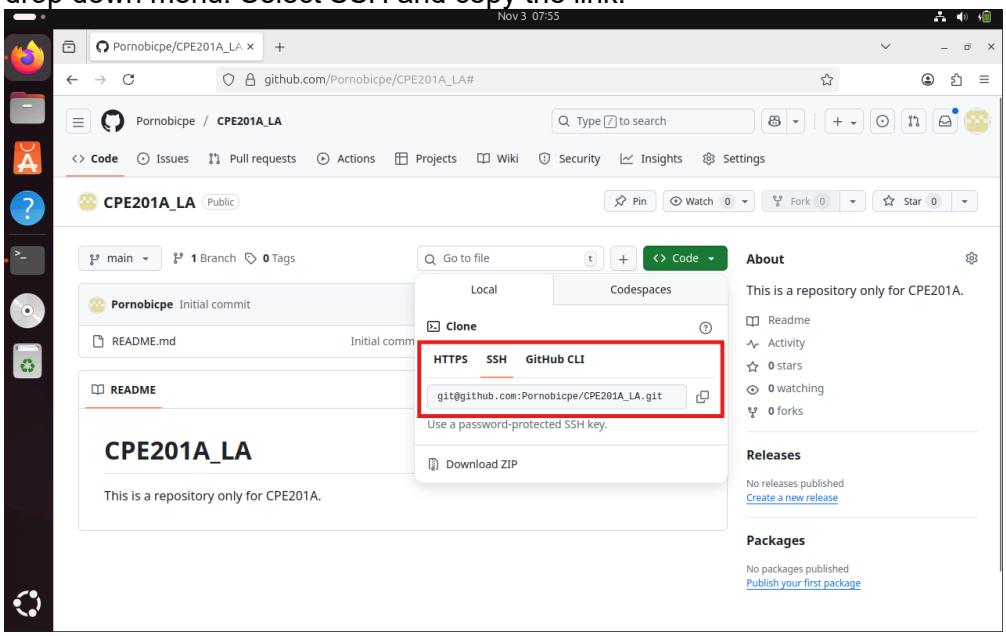
Task 2: Copying the Public Key to Remote Servers

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.

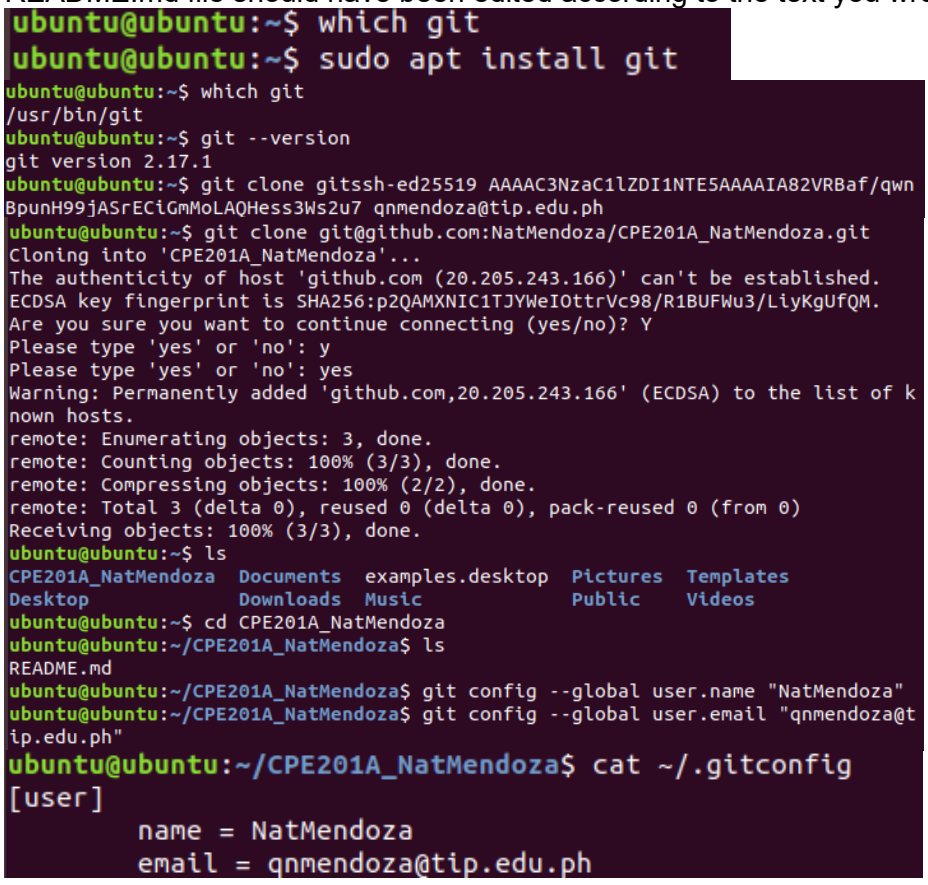
g. Use the git status command to display the state of the working directory and the staging area. This command shows which changes have been staged, which haven't, and which files aren't being tracked by Git. Status output does not show any information regarding the committed project history. What is the result of issuing this command?

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.

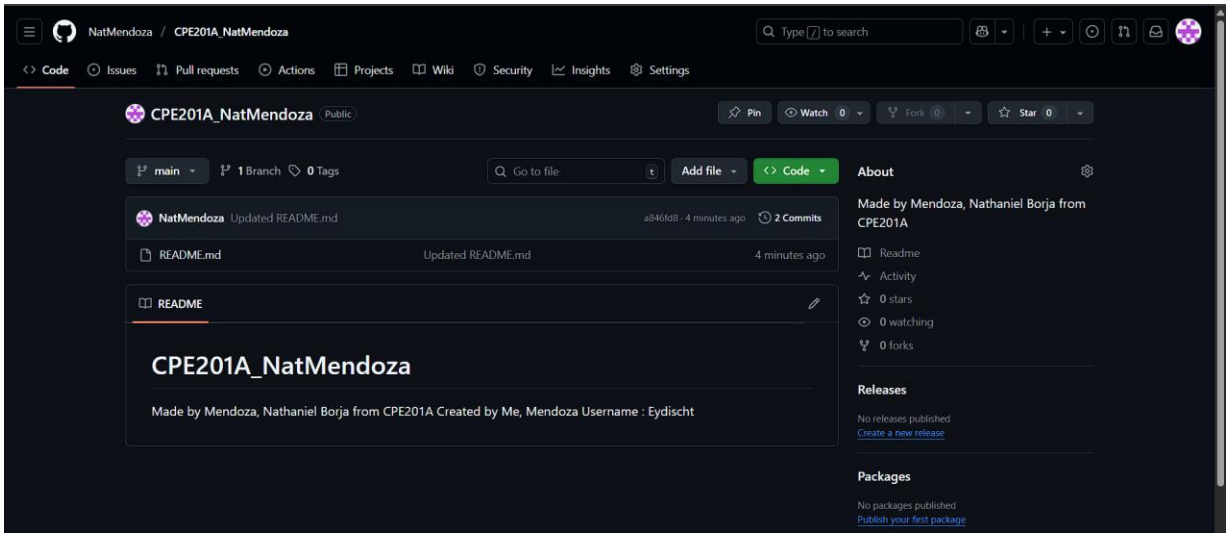
j. Use the command git push <remote><branch> to upload the local repository content to GitHub repository. Pushing means to transfer commits from the local repository to the remote repository. As an example, you may issue git push origin main.

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.



```
ubuntu@ubuntu:~/CPE201A_NatMendoza$ cat ~/.gitconfig
[user]
        name = NatMendoza
        email = qnmendoza@tip.edu.ph
ubuntu@ubuntu:~/CPE201A_NatMendoza$ nano README.md
ubuntu@ubuntu:~/CPE201A_NatMendoza$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
ubuntu@ubuntu:~/CPE201A_NatMendoza$ git add README.md
ubuntu@ubuntu:~/CPE201A_NatMendoza$ git satus
git: 'satus' is not a git command. See 'git --help'.

The most similar command is
        status
ubuntu@ubuntu:~/CPE201A_NatMendoza$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   README.md

ubuntu@ubuntu:~/CPE201A_NatMendoza$ git commit -m "Updated README.md"
[main a846fd8] Updated README.md
 1 file changed, 2 insertions(+)
ubuntu@ubuntu:~/CPE201A_NatMendoza$ git push origin main
Counting objects: 3, done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 339 bytes | 33.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To github.com:NatMendoza/CPE201A_NatMendoza.git
   b6c0ac1..a846fd8  main -> main
ubuntu@ubuntu:~/CPE201A_NatMendoza$
```

5. Outputs:



6. Conclusions/Learnings/Analysis:

Throughout this task, I encountered several errors and challenges, especially when setting up Git and connecting it to GitHub. One of the main problems I faced was the SSH key error saying, *"key is invalid, you must supply a key in OpenSSH public key format."* At first, it was confusing because I didn't fully understand what SSH keys were for, but after researching and generating a new one properly, I realized it was part of Git's security process to verify my identity. So what I did is setting a new/generating a new SSH key to be able to fix my problem which made me to continue the procedure of this activity.

I also had moments when commands like which git showed nothing, which helped me understand that sometimes issues are as simple as missing installations. Each mistake made me more aware of how to carefully read command outputs and analyze what they mean before panicking or guessing the fix.

These experiences taught me that making mistakes isn't something to be afraid of it's actually part of learning how real systems and tools work. Every error pushed me to explore more, understand deeper, and build patience in troubleshooting. Now, I feel more confident using Git and GitHub because I didn't just follow steps I learned through the process, errors included.

Overall, this activity taught me that Git isn't just a requirement it's a powerful tool that helps developers keep their work safe, traceable, and easily shared. I realized that learning these processes early will really help me handle future programming projects more efficiently. Additionally, it gave me a sense of how professionals maintain version history and teamwork using the same system I just learned to use.

7. Assessment Rubric: