Name: Bartolome, Ken Christian Adrian V.	Date Performed: 08/24/2023
Course/Section: CPE31S5	Date Submitted: 08/28/2023
Instructor: Engr. Roman Richard	Semester and SY: 1st Sem 2023-2024

Activity 2: SSH Key-Based Authentication and Setting up Git

1. **Objectives:**

1.1 Configure remote and local machine to connect via SSH using a KEY instead of using a password
1.2 Create a public key and private key
1.3 Verify connectivity
1.4 Setup Git Repository using local and remote repositories
1.5 Configure and Run ad hoc commands from local machine to remote servers

Part 1: Discussion

It is assumed that you are already done with the last Activity (**Activity 1: Configure Network using Virtual Machines).** *Provide screenshots for each task*.

It is also assumed that you have VMs running that you can SSH but requires 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.

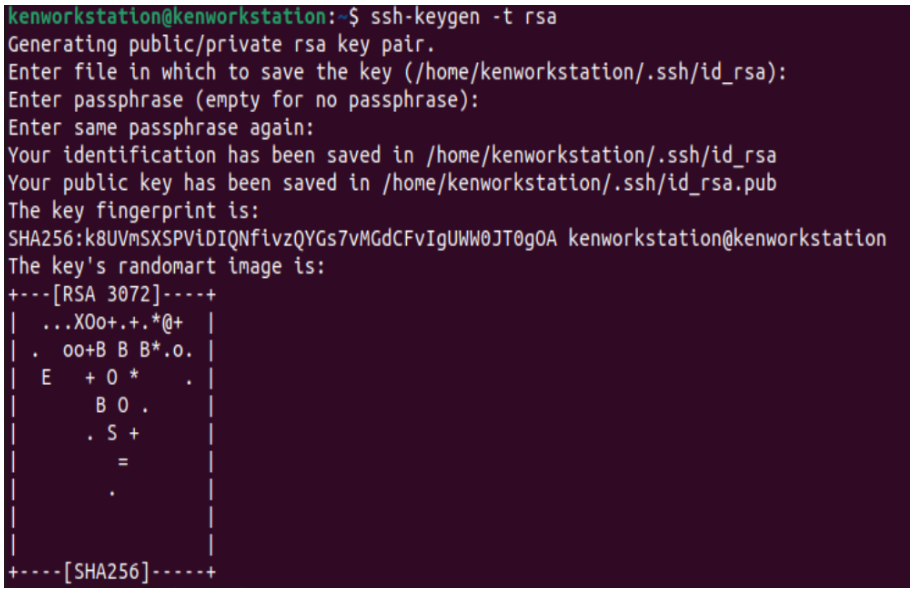
SSH introduced public key authentication as a more secure alternative to the older .rhosts authentication. It improved security by avoiding the need to have password stored in files and eliminated the possibility of a compromised server stealing the user's password.

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.

Task 1: Create an SSH Key Pair for User Authentication

1. The simplest way to generate a key pair is to run *ssh-keygen* without arguments. In this case, it will prompt for the file in which to store keys. First, the tool asked where to save the file. SSH keys for user authentication are usually stored in the users .ssh directory under the home directory. However, in enterprise environments, the location is often different. The default key file name depends on the algorithm, in this case *id_rsa* when using the default RSA algorithm. It could also be, for example, *id_dsa* or *id_ecdsa*.

Screenshot:



2. Issue the command *ssh-keygen -t rsa -b 4096.* The algorithm is selected using the -t option and key size using the -b option.

Screenshot:

3. When asked for a passphrase, just press enter. The passphrase is used for encrypting the key, so that it cannot be used even if someone obtains the private key file. The passphrase should be cryptographically strong.

Screenshot:

```
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/kenworkstation/.ssh/id_rsa
Your public key has been saved in /home/kenworkstation/.ssh/id_rsa.pub
```

4. Verify that you have created the key by issuing the command *ls -la .ssh.* The command should show the .ssh directory containing a pair of keys. For example, id_rsa.pub and id_rsa.

Screenshot:

```
kenworkstation@kenworkstation:~$ ls -la .ssh
total 24
drwx------  2 kenworkstation kenworkstation 4096 Aug 27 23:05 .
drwxr-x--- 16 kenworkstation kenworkstation 4096 Aug 27 21:49 ..
-rw-------  1 kenworkstation kenworkstation 3401 Aug 27 23:07 id_rsa
-rw-r--r--  1 kenworkstation kenworkstation  755 Aug 27 23:07 id_rsa.pub
-rw-------  1 kenworkstation kenworkstation 2240 Aug 27 22:54 known_hosts
-rw-------  1 kenworkstation kenworkstation 1120 Aug 27 22:30 known_hosts.old
```

Task 2: Copying the Public Key to the remote servers

1. To use public key authentication, the public key must be copied to a server and installed in an *authorized_keys* file. This can be conveniently done using the *ssh-copy-id* tool.

Server1:

```
kenserver1@kenserver1:~$ ls -la .ssh
total 8
drwx------ 2 kenserver1 kenserver1 4096 Aug 27 14:01 .
drwxr-x--- 4 kenserver1 kenserver1 4096 Aug 27 14:04 ..
-rw------- 1 kenserver1 kenserver1    0 Aug 27 14:01 authorized_keys
```

Server2:

```
kenserver2@kenserver2:~$ ls -la .ssh
total 8
drwx------ 2 kenserver2 kenserver2 4096 Aug 27 14:17 .
drwxr-x--- 4 kenserver2 kenserver2 4096 Aug 27 14:18 ..
-rw------- 1 kenserver2 kenserver2    0 Aug 27 14:17 authorized_keys
```

2. Issue the command similar to this: *ssh-copy-id -i ~/.ssh/id_rsa user@host*

Server1:

```
kenworkstation@kenworkstation:~$ ssh-copy-id -i ~/.ssh/id_rsa kenserver1@server1
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/kenworkstation/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
kenserver1@server1's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'kenserver1@server1'"
and check to make sure that only the key(s) you wanted were added.
```

Server2:

```
kenworkstation@kenworkstation:~$ ssh-copy-id -i ~/.ssh/id_rsa kenserver2@server2
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/kenworkstation/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
kenserver2@server2's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'kenserver2@server2'"
and check to make sure that only the key(s) you wanted were added.
```

3. Once the public key has been configured on the server, the server will allow any connecting user that has the private key to log in. During the login process, the client proves possession of the private key by digitally signing the key exchange.

Server1:

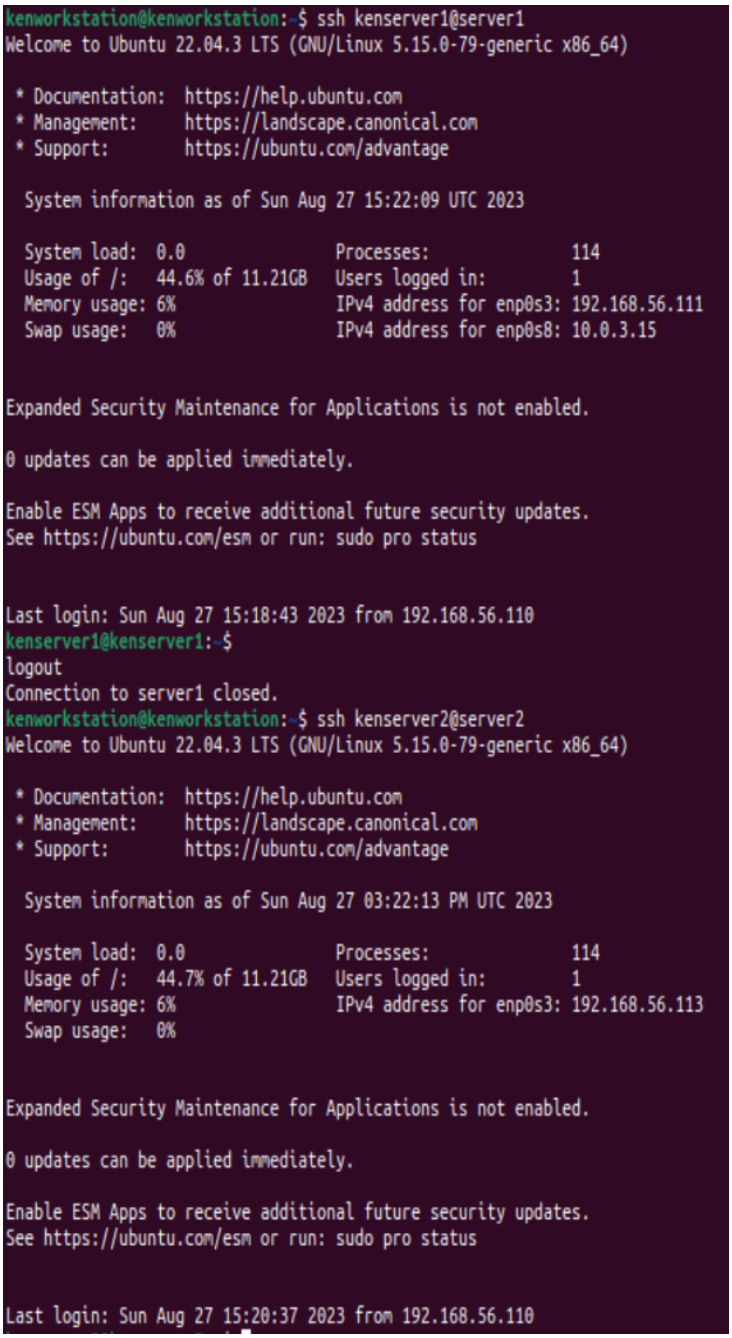
```
kenworkstation@kenworkstation:~$ ssh kenserver1@server1
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-79-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Sun Aug 27 15:18:43 UTC 2023

  System load:  0.0               Processes:               115
  Usage of /:   44.6% of 11.21GB  Users logged in:         1
  Memory usage: 6%                IPv4 address for enp0s3: 192.168.56.111
  Swap usage:   0%                IPv4 address for enp0s8: 10.0.3.15


Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


Last login: Sun Aug 27 14:54:10 2023 from 192.168.56.110
```

Server2:

```
kenworkstation@kenworkstation:~$ ssh kenserver2@server2
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-79-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Sun Aug 27 03:20:36 PM UTC 2023

  System load:  0.0               Processes:               112
  Usage of /:   44.7% of 11.21GB  Users logged in:         1
  Memory usage: 6%                IPv4 address for enp0s3: 192.168.56.113
  Swap usage:   0%


Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


Last login: Sun Aug 27 14:54:34 2023 from 192.168.56.110
```

4. On the local machine, verify that you can SSH with Server 1 and Server 2. What did you notice? Did the connection ask for a password? If not, why?

Screenshot:



Answer: I was able to SSH through server 1 and server 2 by inputting the username and hostname of the said servers. It did not require a password within the workstation to the server. This is because the “autherized_keys” has the password for both the servers, which gives access to the local machine or workstation.

Reflections:
Answer the following:
1. How will you describe the ssh-program? What does it do?

Answer: The SSH program allows secure communication between computers over unsecured networks. It enables remote system access, file transfers, and more. SSH establishes a secure channel, ensuring data privacy and integrity during tasks like remote command execution and system management.

2. How do you know that you already installed the public key to the remote servers?

Answer: If the key was properly installed, the local computer will switch to the server without requesting the server's password, allowing you to determine whether the public key was installed on the distant servers by first attempting to log in with the command "ssh user@hostname".

Part 2: Discussion

Provide screenshots for each task.

It is assumed that you are done with the last activity (**Activity 2: SSH Key-Based Authentication**).

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

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*

Screenshot:

```
kenworkstation@kenworkstation:~$ which git
kenworkstation@kenworkstation:~$ sudo apt install git
[sudo] password for kenworkstation:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  git-man liberror-perl
Suggested packages:
  git-daemon-run | git-daemon-sysvinit git-doc git-email git-gui gitk gitweb git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
  git git-man liberror-perl
0 upgraded, 3 newly installed, 0 to remove and 2 not upgraded.
Need to get 4,147 kB of archives.
After this operation, 21.0 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://ph.archive.ubuntu.com/ubuntu jammy/main amd64 liberror-perl all 0.17029-1 [26.5 kB]
Get:2 http://ph.archive.ubuntu.com/ubuntu jammy-updates/main amd64 git-man all 1:2.34.1-1ubuntu1.10 [954 kB]
Get:3 http://ph.archive.ubuntu.com/ubuntu jammy-updates/main amd64 git amd64 1:2.34.1-1ubuntu1.10 [3,166 kB]
Fetched 4,147 kB in 2s (2,390 kB/s)
Selecting previously unselected package liberror-perl.
(Reading database ... 164961 files and directories currently installed.)
Preparing to unpack .../liberror-perl_0.17029-1_all.deb ...
Unpacking liberror-perl (0.17029-1) ...
Selecting previously unselected package git-man.
Preparing to unpack .../git-man_1%3a2.34.1-1ubuntu1.10_all.deb ...
Unpacking git-man (1:2.34.1-1ubuntu1.10) ...
Selecting previously unselected package git.
Preparing to unpack .../git_1%3a2.34.1-1ubuntu1.10_amd64.deb ...
Unpacking git (1:2.34.1-1ubuntu1.10) ...
Setting up liberror-perl (0.17029-1) ...
Setting up git-man (1:2.34.1-1ubuntu1.10) ...
Setting up git (1:2.34.1-1ubuntu1.10) ...
Processing triggers for man-db (2.10.2-1) ...
```

2. After the installation, issue the command *which git* again. The directory of git is usually installed in this location: *user/bin/git*.

Screenshot:

```
kenworkstation@kenworkstation:~$ which git
/usr/bin/git
```

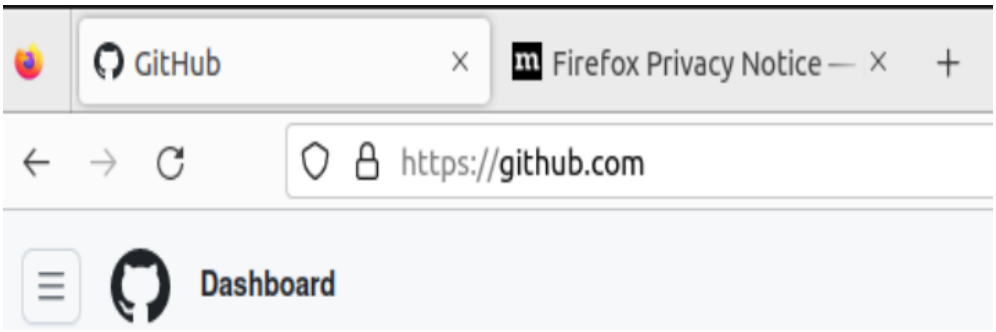
3. The version of git installed in your device is the latest. Try issuing the command *git --version* to know the version installed.

Screenshot:

```
kenworkstation@kenworkstation:~$ git --version
git version 2.34.1
```

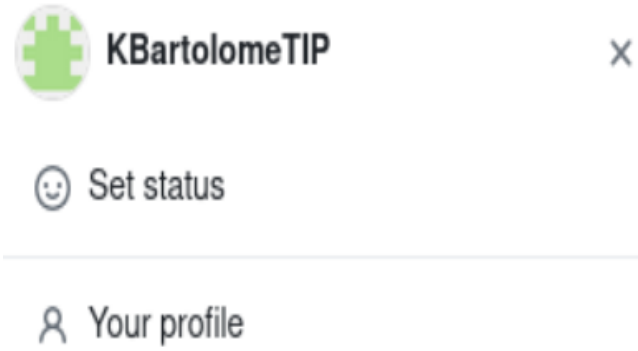
4. Using the browser in the local machine, go to www.github.com.

Screenshot:



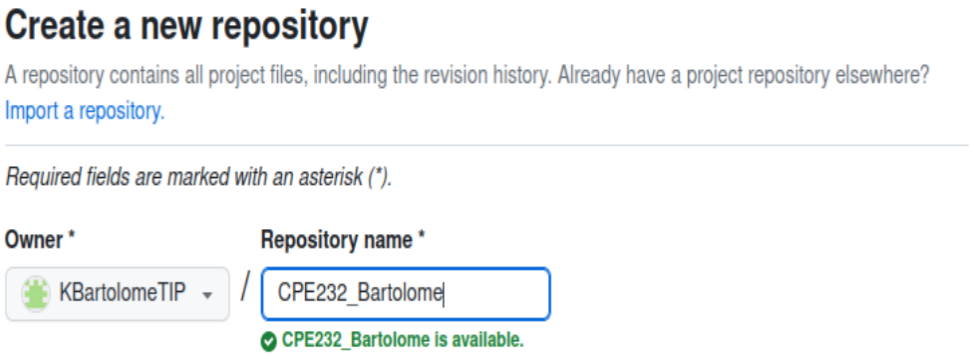
5. Sign up in case you don't have an account yet. Otherwise, login to your GitHub account.

Screenshot:



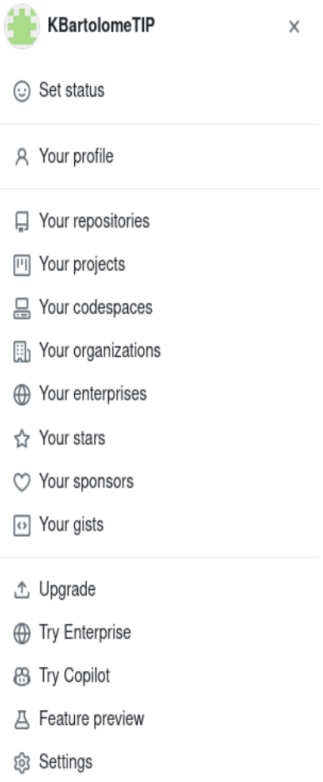
a. Create a new repository and name it as CPE232_yourname. Check Add a README file and click Create repository.

Screenshot:

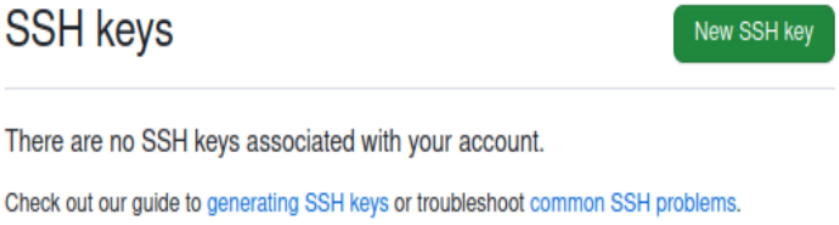


b. Create a new SSH key on GitHub. Go your profile's setting and click SSH and GPG keys. If there is an existing key, make sure to delete it. To create a new SSH keys, click New SSH Key. Write CPE232 key as the title of the key.

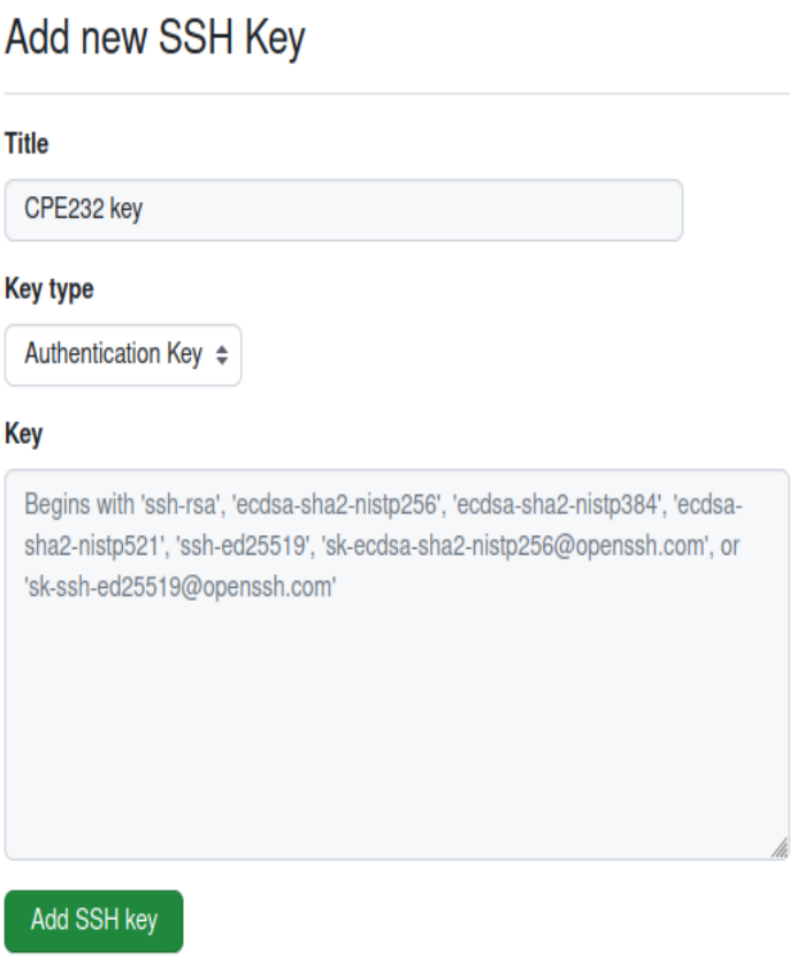
Screenshot:



Profile Setting



SSH Keys



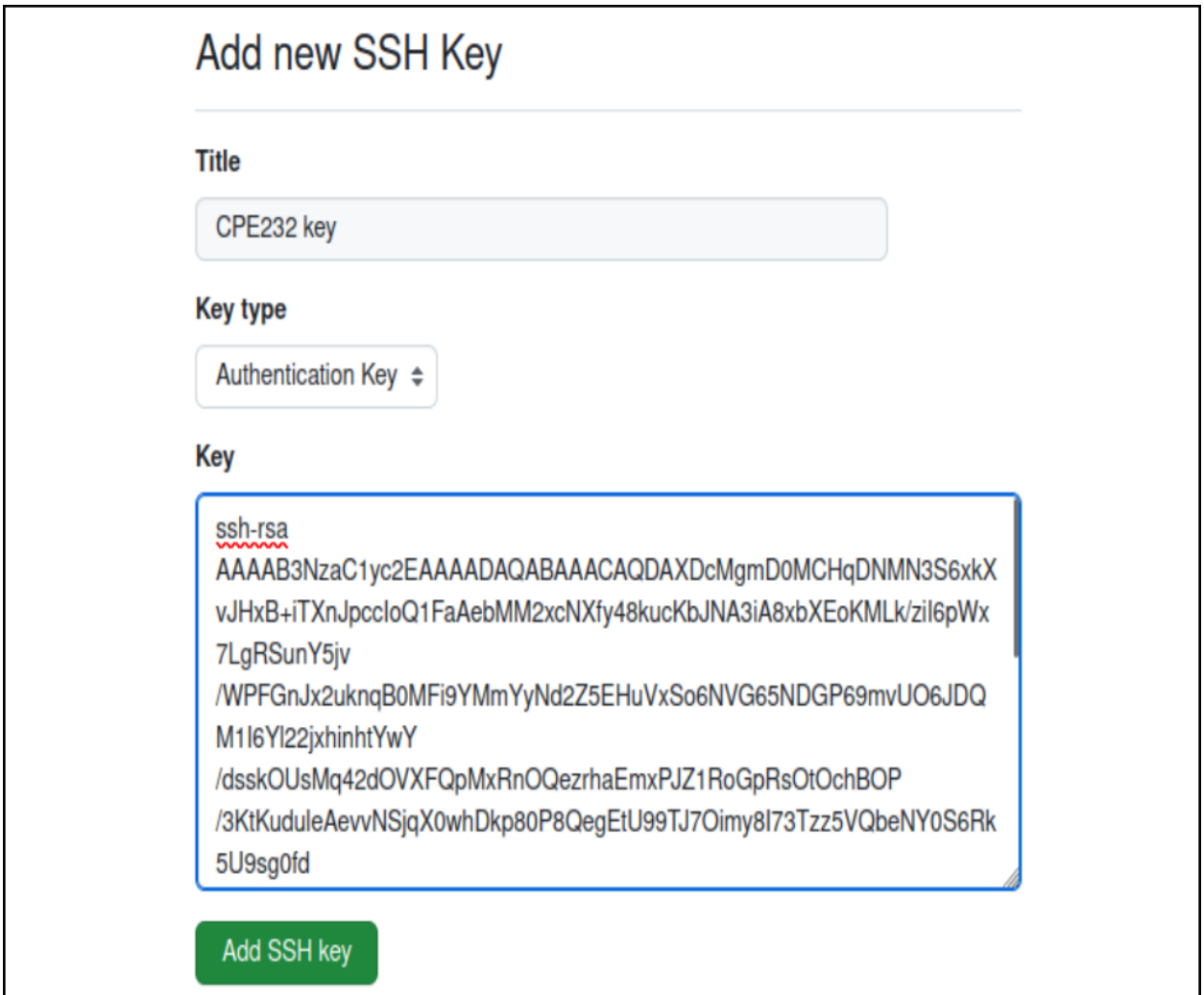
Adding SSH Key

c. On the local machine's terminal, issue the command cat .ssh/id_rsa.pub and copy the public key. Paste it on the GitHub key and press Add SSH key.

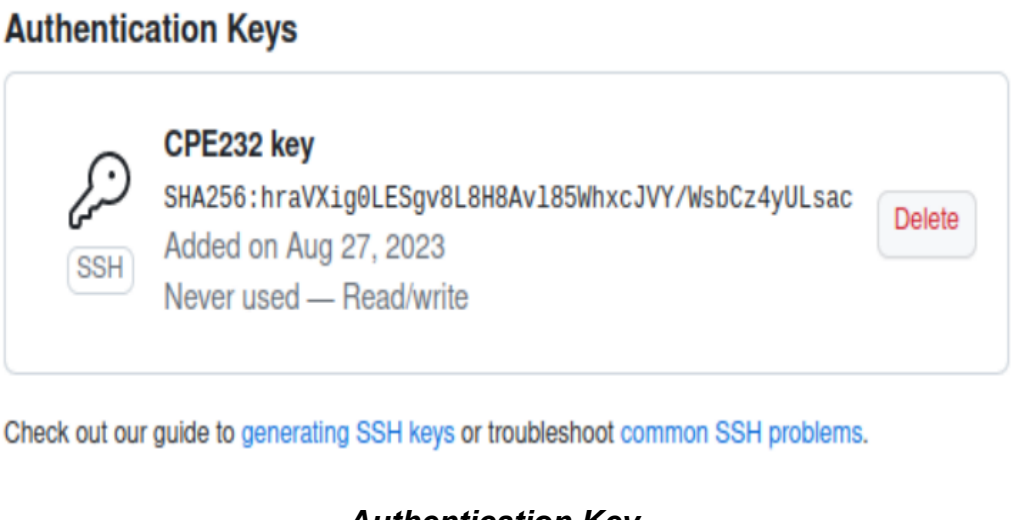
Screenshot:

```
kenworkstation@kenworkstation:~$ cat .ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQDAXDcMgmD0MCHqDNMN3S6xkXvJH
xB+iTXnJpccIoQ1FaAebMM2xcNXfy48kucKbJNA3iA8xbXEoKMLk/ziI6pWx7LgRS
unY5jv/WPFGnJx2uknqB0MFi9YMmYyNd2Z5EHuVxSo6NVG65NDGP69mvUO6JDQM1I
6Yl22jxhinhtYwY/dsskOUsMq42dOVXFQpMxRnOQezrhaEmxPJZ1RoGpRsOtOchBO
P/3KtKuduleAevvNSjqX0whDkp80P8QegEtU99TJ7Oimy8I73Tzz5VQbeNY0S6Rk5
U9sg0fd/SxTLLHNug6i6q6BVcDQPiWn7NHCiXikIZY2lG6RIyt0OCP3IxIwuxyI3f
DhrDGgoRLjITSt5bHjzGghSaxopU+zEKdlA37PPn1ip98jsYXSBEW8vHqLRmeph5A
J6kaCnwghrgy4YrbxV3ZnDvxhv9+saCLOW8Mw2nF+wiKTXASa8qUIfPOV/fC2ymsC
PFrLPECtHli9v6KmkI8G/5neq91AoopQLToWFeBd5eW8KPOeJlXR/2OEzuSE8VKSl
Zt1XZjmJZO3ovlsitlMc9ZXjVeKWiAL57rnEk6O/MTlixmIbqcwaWiaBWQOxHBHyD
IA29to/yB1j2cXLrVDJbI7egRHod8UYhr5mxUL7DOuFmnRKDMFl68Y3F1oETSDjaA
m27pXiw== kenworkstation@kenworkstation
```

Public Key

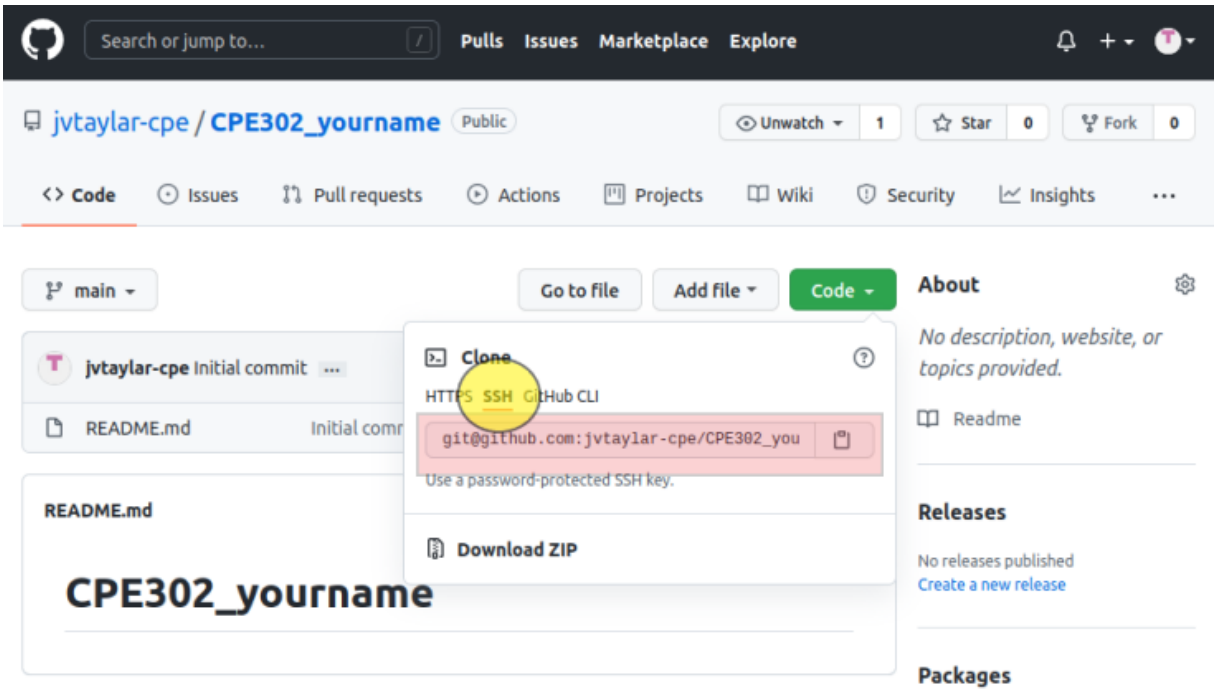


Adding new SSH Key with Public Key

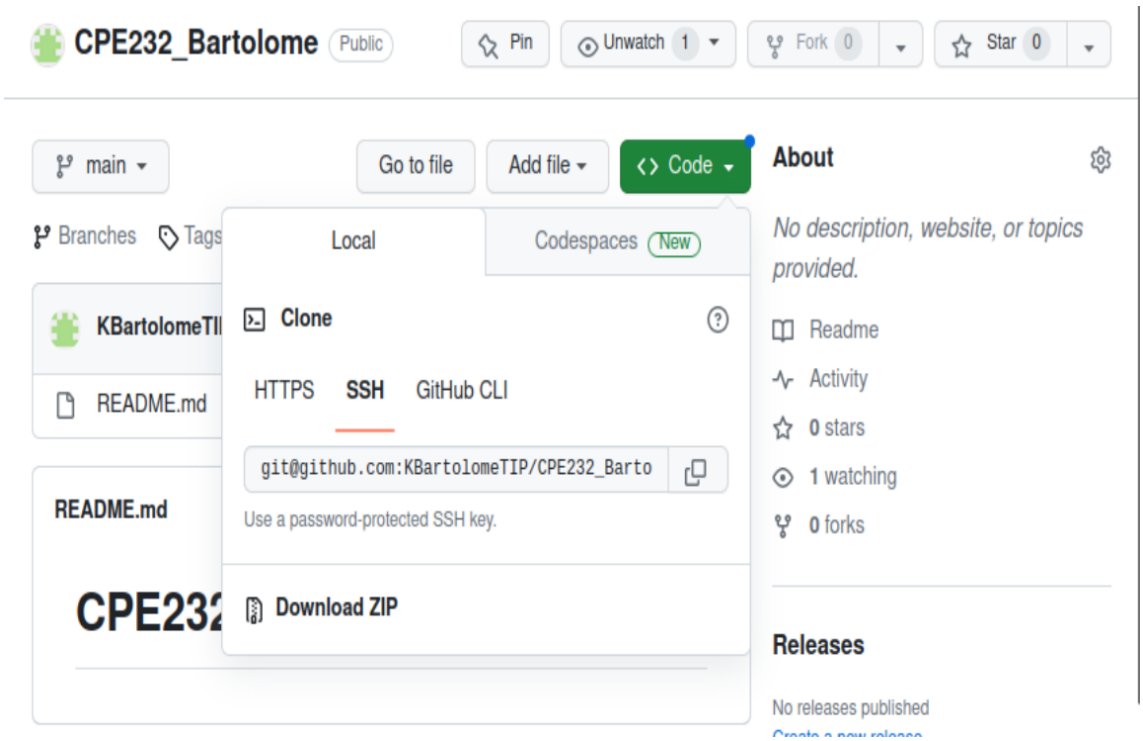


Authentication Key

d. 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.



Screenshot:



e. Issue the command git clone followed by the copied link. For example, *git clone git@github.com:jvtaylar-cpe/CPE232_yourname.git*. When prompted to continue connecting, type yes and press enter.

Screenshot:

```
kenworkstation@kenworkstation:~$ git clone git@github.com:KBa
rtolomeTIP/CPE232_Bartolome.git
Cloning into 'CPE232_Bartolome'...
The authenticity of host 'github.com (20.205.243.166)' can't
be established.
ED25519 key fingerprint is SHA256:+DiY3wvvV6TuJJhbpZisF/zLDA0
zPMSvHdkr4UvCOqU.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerp
rint])? 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: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

f. To verify that you have cloned the GitHub repository, issue the command *ls*. Observe that you have the CPE232_yourname in the list of your directories. Use CD command to go to that directory and LS command to see the file README.md.

Screenshot:

```
kenworkstation@kenworkstation:~$ ls
CPE232_Bartolome  Documents  Music     Public  Template
Desktop           Downloads  Pictures  snap    Videos
kenworkstation@kenworkstation:~$ cd CPE232_Bartolome
kenworkstation@kenworkstation:~/CPE232_Bartolome$ ls
README.md
```

g. Use the following commands to personalize your git.

- *git config --global user.name "Your Name"*

Screenshot:

```
kenworkstation@kenworkstation:~$ git config --global user.name "Bartolome Ken"
```

- *git config --global user.email yourname@email.com*

Screenshot:

```
kenworkstation@kenworkstation:~$ git config --global user.email "qkcabartolome1@tip.edu.ph"
```

- Verify that you have personalized the config file using the command *cat ~/.gitconfig*

Screenshot:

```
kenworkstation@kenworkstation:~$ cat ~/.gitconfig
[user]
        name = Bartolome Ken
        email = qkcabartolome1@tip.edu.ph
```

h. 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.

Screenshot:

```
kenworkstation@kenworkstation:~$ ls
CPE232_Bartolome  Desktop  Documents  Downloads  Music  Pictures  Public  snap  Templates  Videos
kenworkstation@kenworkstation:~$ cd CPE232_Bartolome
kenworkstation@kenworkstation:~/CPE232_Bartolome$ ls
README.md
kenworkstation@kenworkstation:~/CPE232_Bartolome$ nano README.md
```

nano README.md

```
  GNU nano 6.2                                                README.md *
# CPE232_Bartolome
Activity 2 - SSH Key-Based Authentication and GIT Setup
08/28/2023
```

Providing Information

i. 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?

Screenshot:

```
kenworkstation@kenworkstation:~/CPE232_Bartolome$ 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 restore <file>..." to discard changes in working directory)
        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
```

j. Use the command *git add README.md* to add the file into the staging area.

Screenshot:

```
kenworkstation@kenworkstation:~/CPE232_Bartolome$ git add README.md
```

k. 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.

Screenshot:

```
kenworkstation@kenworkstation:~/CPE232_Bartolome$ git commit -m "GIT Setup"
[main ebdae64] GIT Setup
 1 file changed, 3 insertions(+), 1 deletion(-)
```

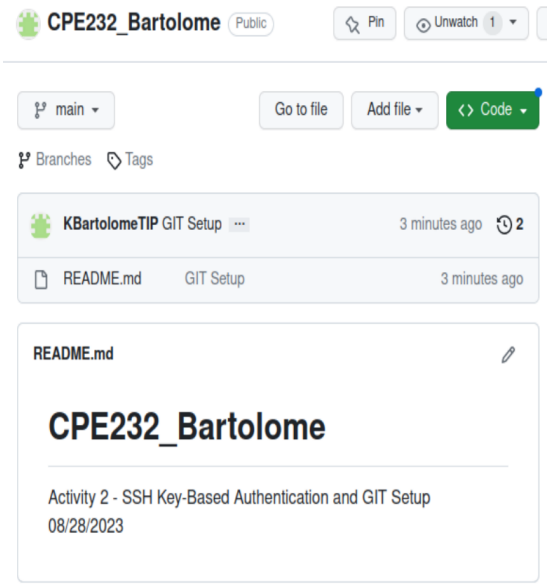
l. Use the command *git push <remote><branch>* to upload the local repository content to GitHub repository. Pushing means to transfer commits from the local repository to the remote repository. As an example, you may issue *git push origin main.*

Screenshot:

```
kenworkstation@kenworkstation:~/CPE232_Bartolome$ 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), 335 bytes | 335.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:KBartolomeTIP/CPE232_Bartolome.git
   0729c8e..ebdae64  main -> main
```

m. 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 the 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.

Screenshot:



Reflections:
Answer the following:

3. What sort of things have we so far done to the remote servers using ansible commands?

Answer: Creating a connection between the servers and the GitHub repository was the major usage of the ansible command. To do this, an SSH key was created and then used to automatically log into the servers and the repository. The files inside the repository were also edited and updated using the ansible and nano commands.

4. How important is the inventory file?

Answer: The inventory file is essential because it provides a reliable method for keeping track of and managing stock levels. Both stockouts and the expense of maintaining inventory can be decreased. For accurate inventory management, it's crucial to maintain the inventory file updated.

Conclusions/Learnings:
Answer: I learned a lot about how to create and utilise ssh keys and how they work from this assignment. I am familiar with the fundamentals of ssh keys and how to utilise them. Keys may be used as a password to get access to the server using only ssh commands. I also learnt how to set up a github account, link it to the local computer, and edit local files that affect the github repository.

"I affirm that I shall not give or receive any unauthorized help on this assignment and that all work is my own."