Name: RALPH ANDREI DIOCERA	Date Performed: 8/22/2023
Course/Section: CPE232/CPE31S4	Date Submitted:
Instructor: Engr. Jonathan V. Taylar	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*.

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.
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.

```
ralph@manageNode:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ralph/.ssh/id_rsa):
/home/ralph/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ralph/.ssh/id_rsa.
Your public key has been saved in /home/ralph/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:WA58YmTiHsc/pj5Wmp5nwuEdtN4f+Z06nVnSLEs0pE8 ralph@manageNode
The key's randomart image is:
+---[RSA 4096]----+
|     . o         |
|    . *        .  |
|     o B o    o   |
|    . + O.   . E  |
|     . ..S.   + + |
|       .o+.   .= +|
|      o.B o  o..++|
| Help .O.= .  +.=.|
|      o+=   ...+..|
+----[SHA256]-----+
```

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.

```
ralph@manageNode:~$ ls -la .ssh
total 20
drwx------  2 ralph ralph 4096 Aug 22 17:29 .
drwxr-xr-x 16 ralph ralph 4096 Aug 22 17:16 ..
-rw-------  1 ralph ralph 3247 Aug 22 17:31 id_rsa
-rw-r--r--  1 ralph ralph  742 Aug 22 17:31 id_rsa.pub
-rw-r--r--  1 ralph ralph  888 Aug 15 18:14 known_hosts
ralph@manageNode:~$
```

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

```
ralph@manageNode:~$ ssh-copy-id -i ~/.ssh/id_rsa ralph@manageNode
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/ralph/.ssh
/id_rsa.pub"
The authenticity of host 'managenode (10.0.2.15)' can't be established.
ECDSA key fingerprint is SHA256:PVFdmtoYdktj7jG6bnHvEL6pjvKujcfxd1nxQgitFI0.
Are you sure you want to continue connecting (yes/no)? yes
/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 promp
ted now it is to install the new keys
ralph@managenode's password:
Permission denied, please try again.
ralph@managenode's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'ralph@manageNode'"
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.

```
ralph@manageNode:~$ ssh 'ralph@manageNode'
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 5.4.0-150-generic x86_64)

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

Expanded Security Maintenance for Infrastructure is not enabled.

0 updates can be applied immediately.

85 additional security updates can be applied with ESM Infra.
Learn more about enabling ESM Infra service for Ubuntu 18.04 at
https://ubuntu.com/18-04

New release '20.04.6 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Your Hardware Enablement Stack (HWE) is supported until April 2023.
Last login: Tue Aug 22 17:38:14 2023 from 10.0.2.15
```

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?

Reflections:

Answer the following:

1. How will you describe the ssh-program? What does it do?

- I can describe the ssh-program in terms of being a secured remote communication because of how it technically serves as a network protocol that purposely makes an encrypted and properly secured connection between two or more servers that can implement a hidden transmission of data between the connections. The secure shell has also the ability to implement a remote access feature which lets the users manage different servers that have been connected to the main workstation.

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

   - Based on the commands we used, I can determine if I have already installed the public key to the remote servers with the use of “ssh user@server1” or “ssh user@server2” which lets me know that it has been successfully connected to the main workstation.

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**).

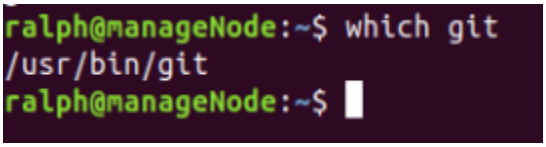
Set up Git

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



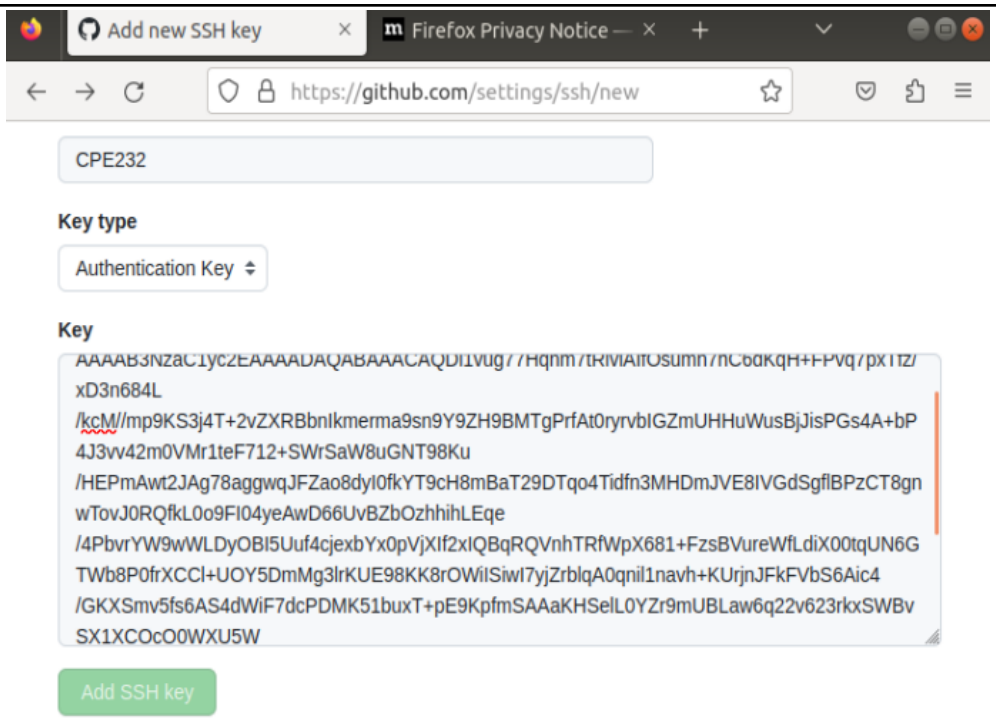
3. The version of git installed in your device is the latest. Try issuing the command *git --version* to know the version installed.

```
ralph@manageNode:~$ git --version
git version 2.17.1
ralph@manageNode:~$
```

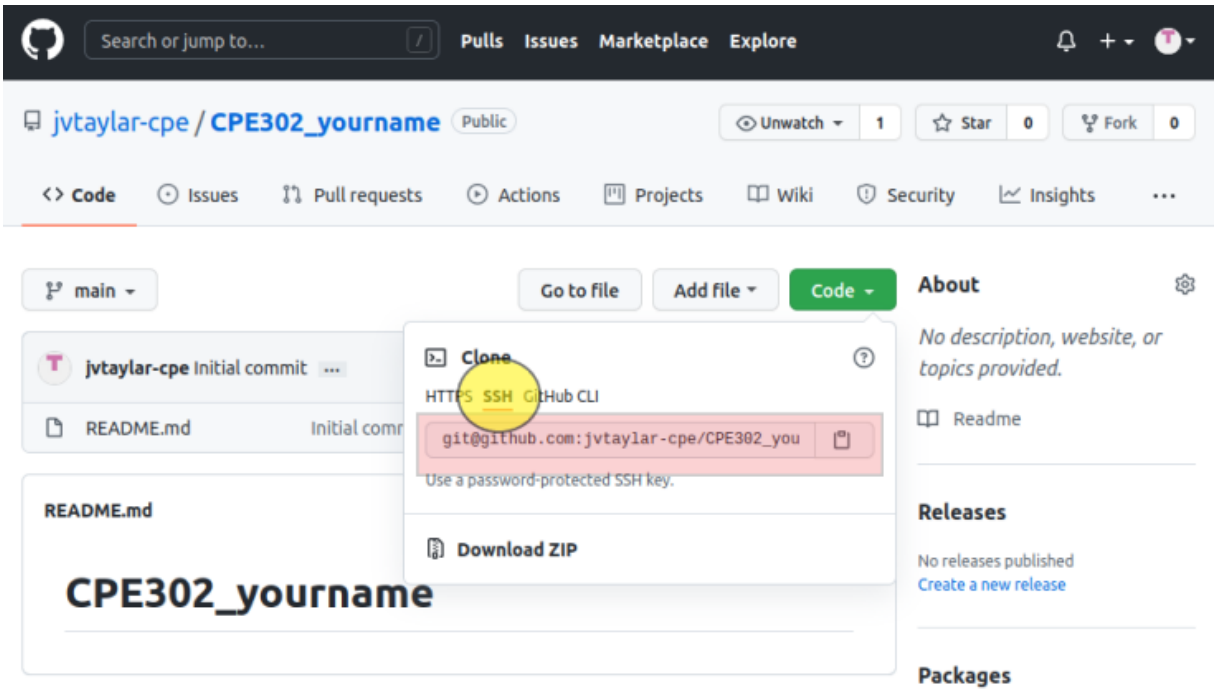
4. Using the browser in the local machine, go to www.github.com.
5. Sign up in case you don't have an account yet. Otherwise, login to your GitHub account.
    a. Create a new repository and name it as CPE232_yourname. Check Add a README file and click Create repository.

    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.
    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.

```
ralph@manageNode:~$ cat .ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQDi1vug77Hqnm7tRiviAIfOsumn7hC6dKqH+FPvq7p
xTfz/xD3n684L/kcM//mp9KS3j4T+2vZXRBbnIkmerma9sn9Y9ZH9BMTgPrfAt0ryrvbIGZmUHHuWus
BjJisPGs4A+bP4J3vv42m0VMr1teF712+SWrSaW8uGNT98Ku/HEPmAwt2JAg78aggwqJFZao8dyI0fk
YT9cH8mBaT29DTqo4Tidfn3MHDmJVE8IVGdSgflBPzCT8gnwTovJ0RQfkL0o9FI04yeAwD66UvBZbOz
hhihLEqe/4PbvrYW9wWLDyOBI5Uuf4cjexbYx0pVjXIf2xIQBqRQVnhTRfWpX681+FzsBVureWfLdiX
00tqUN6GTWb8P0frXCCl+UOY5DmMg3lrKUE98KK8rOWiISiwI7yjZrblqA0qnil1navh+KUrjnJFkFV
bS6Aic4/GKXSmv5fs6AS4dWiF7dcPDMK51buxT+pE9KpfmSAAaKHSelL0YZr9mUBLaw6q22v623rkxS
WBvSX1XCOcO0WXU5W/PkXUovJfTvv8jb73tXhKdmH20jEMelynTRdSp48e6THCTK4l5kznokHK5nElk
P7pEe/Yk3c8h4q/c4Ew3VaCmTSyw8bI0HShsoCm0iFq/U4LMGSkPG2I9yLO99IyKVS4iTpSXoAS3OGV
VgWjA/BidWw== ralph@manageNode
```



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.



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.

```
ralph@manageNode:~$ git clone git@github.com:ryuki2012/CPE232_Diocera.git
Cloning into 'CPE232_Diocera'...
The authenticity of host 'github.com (20.205.243.166)' can't be established.
ECDSA key fingerprint is SHA256:p2QAMXNIC1TJYWeIOttrVc98/R1BUFWu3/LiyKgUfQM.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'github.com,20.205.243.166' (ECDSA) to the list of k
nown hosts.
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (6/6), done.
ralph@manageNode:~$
```

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.

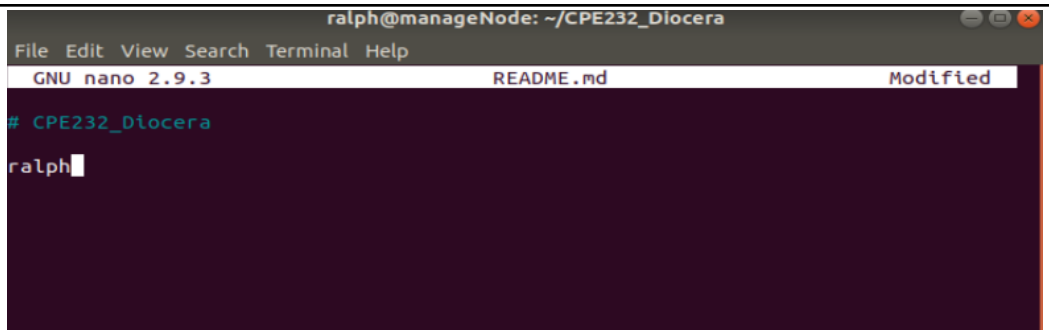
```
ralph@manageNode:~$ ls
CPE232_Diocera  Documents  examples.desktop  Pictures  Templates
Desktop         Downloads  Music             Public    Videos
ralph@manageNode:~$ cd CPE232_Diocera
ralph@manageNode:~/CPE232_Diocera$ ls
ralphdiocera-cpe  README.md
ralph@manageNode:~/CPE232_Diocera$
```

g. 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*

```
ralph@manageNode:~/CPE232_Diocera$ git config --global user.name "ralph"
ralph@manageNode:~/CPE232_Diocera$ git config --global user.email "qramdiocera@
tip.edu.ph"
ralph@manageNode:~/CPE232_Diocera$ cat ~/.gitconfig
[user]
        name = ralph
        email = qramdiocera@tip.edu.ph
ralph@manageNode:~/CPE232_Diocera$
```

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.



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?

```
ralph@manageNode:~/CPE232_Diocera$ 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")
ralph@manageNode:~/CPE232_Diocera$
```

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

```
ralph@manageNode:~/CPE232_Diocera$ git add README.md
ralph@manageNode:~/CPE232_Diocera$
```

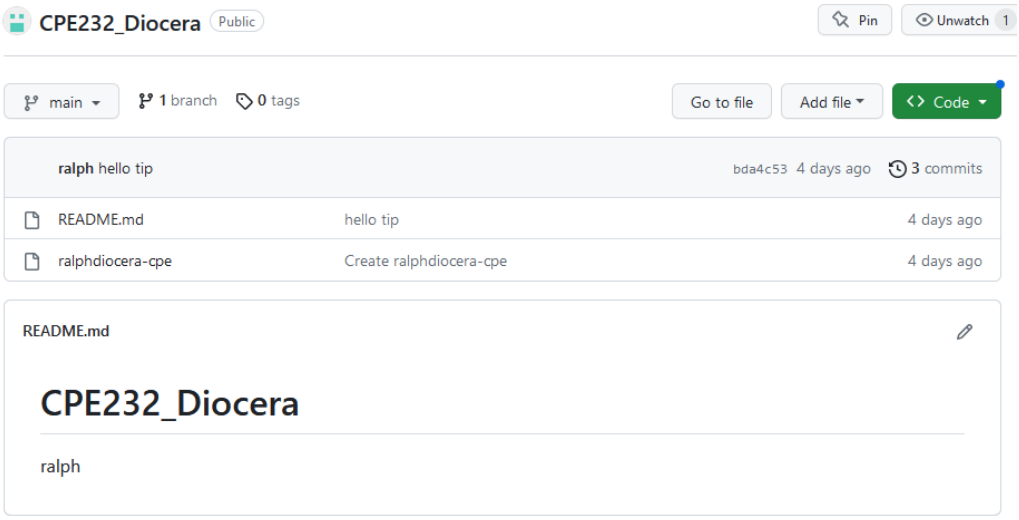
k. Use the *git commit -m "your message"* to create a snapshot of the staged changes along the timeline of the Git projects history. The use of this command is required to select the changes that will be staged for the next commit.

```
ralph@manageNode:~/CPE232_Diocera$ git commit -m "hello tip"
[main bda4c53] hello tip
 1 file changed, 3 insertions(+), 1 deletion(-)
ralph@manageNode:~/CPE232_Diocera$
```

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*.

```
ralph@manageNode:~/CPE232_Diocera$ git push origin main
Counting objects: 3, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 301 bytes | 301.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To github.com:ryuki2012/CPE232_Diocera.git
   3f8199b..bda4c53  main -> main
ralph@manageNode:~/CPE232_Diocera$
```

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.



Reflections:

Answer the following:

3. What sort of things have we so far done to the remote servers using ansible commands?
   - Based on the ansible commands we have done during different tasks, we have made a configuration upon the workstation and its servers that interconnects them together as one working group which gives us users a centralized view of the managed servers and upon doing an installation from the workstation using specific commands, it will affect the servers connected to it which we have made earlier.
4. How important is the inventory file?
   - Inventory file has a lot of importance in terms of Linux and how it's being implemented in using ansible functions and commands because this helps us determine what server the ansible will function on and this helps a user store

different information such as public or private SSH keys, the username that's been in the remote access servers and lets the ansible connect to remote servers through the shadow within the system of Linux securing it.

Conclusions/Learnings:

- **In conclusion, this activity about the purpose of SSH as a whole has given me a lot of knowledge whereas, in this matter, we can use SSH as a way to configure a local or remote-access server that interconnects using a public or private key with a specific set of commands that benefits the user with an excellent security within making data transmissions and automation confidential. I have also learned what is the difference between the purpose of a public SSH key and a private SSH key whereas the public one has the purpose of being stored within the remote servers that retrieves the identity of a specific user who tries to authenticate the SSH server which makes it open for any servers or users. On the other hand, a private key is not meant to be shared with anyone of how confidential it is because once there's another user who has accessed the remote-access server using the private key, it will be given enough rights to manage confidential pieces of data and then mistransfer.**