

Name: Nealian Beth B. Nanquil	Date Performed: August 25, 2022
Course/Section: CPE 232-CPE31S23	Date Submitted: August 25, 2022
Instructor: Engr. Jonathan V. Taylar	Semester and SY: 1st semester 2022-2023
Activity 2: SSH Key-Based Authentication and Setting up Git	
1. Objectives: <ol style="list-style-type: none"> 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 <p>It is assumed that you are already done with the last Activity (Activity 1: Configure Network using Virtual Machines). <i>Provide screenshots for each task.</i></p> <p>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.</p> <p>What Is ssh-keygen?</p> <p>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.</p> <p>SSH Keys and Public Key Authentication</p> <p>The SSH protocol uses public key cryptography for authenticating hosts and users. The authentication keys, called SSH keys, are created using the keygen program.</p> <p>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.</p> <p>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.</p>	
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`.

```
TIPQC@Q5202-30 MINGW64 ~
$ cd .ssh

TIPQC@Q5202-30 MINGW64 ~/.ssh
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/TIPQC/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/TIPQC/.ssh/id_rsa
Your public key has been saved in /c/Users/TIPQC/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:DQ+3MfqNXV90fVbRdWJ/Jx3XiJkp5DSNkwDT1LthkKE TIPQC@Q5202-30
The key's randomart image is:
+----[RSA 3072]-----+
|      o=Eo++      |
|      .o+=..= ..  |
|      oo*o=+.B    |
|      oB+=. *B    |
|      So+o  o.@   |
|      ..+ . +B    |
|      o o  o      |
|                  |
+----[SHA256]-----+

TIPQC@Q5202-30 MINGW64 ~/.ssh
$ ls
DNSCache/ id_rsa id_rsa.pub known_hosts known_hosts.old
```

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.

```
TIPQC@Q5202-30 MINGW64 ~/.ssh
$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/TIPQC/.ssh/id_rsa): y
```

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.

```

Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in y
Your public key has been saved in y.pub
The key fingerprint is:
SHA256:1TiGUrvI7nvdBDuCqdsHAPDFAURNYaoyWggJFirz/F8 TIPQC@Q5202-30
The key's randomart image is:
+---[RSA 4096]-----+
|+==*o.             |
|=o +o. o . .       |
|*.o . o + o        |
|o*.. o o.o         |
|= +.oo. So         |
|oo o+ . o .        |
|. .o. oE+          |
|.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`.

```

TIPQC@Q5202-30 MINGW64 ~/.ssh
$ ls
DNSSCache/ id_rsa id_rsa.pub known_hosts known_hosts.old y y.pub

```

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`

```

TIPQC@Q5202-30 MINGW64 ~/.ssh
$ ssh-copy-id -i ~/.ssh/id_rsa nanquil@192.168.56.105

/usr/bin/ssh-copy-id: ERROR: failed to open ID file '/c/Users/TIPQC/ssh/id_rsa.pub': No such file or directory

TIPQC@Q5202-30 MINGW64 ~/.ssh
$ ls
DNSSCache/ id_rsa id_rsa.pub known_hosts known_hosts.old y y.pub

TIPQC@Q5202-30 MINGW64 ~/.ssh
$ rm y

TIPQC@Q5202-30 MINGW64 ~/.ssh
$ rm y.pub

```

```
TIPQC@Q5202-30 MINGW64 ~/.ssh
$ ssh-copy-id nanquil@192.168.56.105
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/c/Users/TIPQC/.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
nanquil@192.168.56.105's password:
Permission denied, please try again.
nanquil@192.168.56.105's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'nanquil@192.168.56.105'"
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.
 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?
- Server 1:

```
TIPQC@Q5202-30 MINGW64 ~/.ssh
$ ssh nanquil@192.168.56.105
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-46-generic x86_64)

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

0 updates can be applied immediately.

Last login: Thu Aug 25 08:10:18 2022 from 192.168.56.1
nanquil@server1:~$ exit
logout
Connection to 192.168.56.105 closed.

TIPQC@Q5202-30 MINGW64 ~/.ssh
$ ssh nanquil@server1
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-46-generic x86_64)

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

0 updates can be applied immediately.

Last login: Thu Aug 25 08:47:48 2022 from 192.168.56.1
nanquil@server1:~$
```

Server 2:

```
TIPQC@Q5202-30 MINGW64 ~/.ssh
$ ssh nanquil@192.168.56.106
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-46-generic x86_64)

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

0 updates can be applied immediately.

nanquil@server2:~$ exit
logout
Connection to 192.168.56.106 closed.
```

```
TIPQC@Q5202-30 MINGW64 ~/.ssh
$ ssh nanquil@server2
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-46-generic x86_64)

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

0 updates can be applied immediately.

Last login: Thu Aug 18 12:16:32 2022 from 192.168.56.1
nanquil@server2:~$
```

Reflections:

Answer the following:

1. **How will you describe the ssh-program? What does it do?**
SSH program allows us to securely communicate to a remote computer. Also, SSH program encrypts message before executing to the server.

2. **How do you know that you already installed the public key to the remote servers?** To know if I have installed the public key to the remote servers, I entered the command `ls` to know if there is a file of authorized keys and check it by entering `cat` command followed by the file name in the server.

Server 1:

```
nanquill@server1:~$ cd .ssh
nanquill@server1:~/.ssh$ ls
authorized_keys
nanquill@server1:~/.ssh$ cat authorized_keys
cat: authorized: No such file or directory
cat: keys: No such file or directory
nanquill@server1:~/.ssh$ cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQGCv+YizhTrP3DkzPYE8DzkKfVKK5fh8yWwA/Jjo1sIYtm0F7IuDKSWVHMLvjDUuy
nnAwViitswgqi8AxlGllJ44bgC+ifg8MakrZnsdwydfmsWQ35YneUm0P7UmDs1GiCaL1q9X9TiF66lEH5B35/xJfTtcCV8Mea0jbW
FLUJiQSo0JPX5eBHH20dET+jMnodHzHQ3FExpZrT4xel2QSus1fLD/OI0DuUgIteGEtZvoeBQ9hTt1dSHne2JZCveq+w0wWeUiOmm
gjb9p9lFe3CCkilmlvmC6usTRD7LHTRIHEC+az7u1yU+X1EyDhPc2v5D/fcingyfw0zgt0DdPTpI+DJ/Yb4podN5BZvfjouGxEWQcK
9pjj70lFgZ09SXiJtM00L4j60+ANI0DYnKqhCnYsPNo9JUJU447Fkj7Zc+LqEfQe8rnzL7c92g/j0lqRSia3wky/fU9Er4ba1i/SC
Q6CAWZ30qzTwUam6BanszW10d0126hCpqFyAov1Xp34v5M= TIPQC@Q5202-30
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQGDc8zbPhL0s9TxxgTULi9siqF+SxRgG0eJ+XWUUVVexrcI1m3W7vqGLK7uUdzXtIm
XkEmnnIhKCH2chv9YER4iyYSFHj2B9CyR+3Zv55L7+s1sVBMTFpGWBf4T4uge9pIvxnHxjgF143A7M5TID1o+CcHhvk57YPUNHCEW
AIRv8TQ0+B0csD9ZnoHxe/yyeIyYPZiXApzFCQkp+4PxxytR+sLJGjVZK1e0aAK276j7oT+VSP9RkLaa6RLvpuIe/bQDA21Y/ghck
EYCDWPjddjFst72t8eBdPEUJl1a7+yAo0d5Q9VwIazX2TfhjJ1wXYPwZ1LbtBYhRbxANMDjQMB8mwzOqldm0733+6ZBU15D1B4jG1
LzB0o5veiy1UdMFsowDE6LdNFP20fQq7R+3yQ74UHKVY/wjSn21Z8CtY0cTMZ0aTGn6pIPEjGtZ30J0jRZl1lratB5HecjpaENhSg
PLrem6grdPW0e0tJKMBR+t9dS5mbWMkZk5qmpo3apZH/AE= TIPQC@Q5202-30
nanquill@server1:~/.ssh$
```

Server 2:

```
nanquill@server2:~$ cd .ssh
nanquill@server2:~/.ssh$ ls
authorized_keys
nanquill@server2:~/.ssh$ cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQGDc8zbPhL0s9TxxgTULi9siqF+SxRgG0eJ+XWUUVVexrcI1m3W7vqGLK7uUdzXtIm
XkEmnnIhKCH2chv9YER4iyYSFHj2B9CyR+3Zv55L7+s1sVBMTFpGWBf4T4uge9pIvxnHxjgF143A7M5TID1o+CcHhvk57YPUNHCEW
AIRv8TQ0+B0csD9ZnoHxe/yyeIyYPZiXApzFCQkp+4PxxytR+sLJGjVZK1e0aAK276j7oT+VSP9RkLaa6RLvpuIe/bQDA21Y/ghck
EYCDWPjddjFst72t8eBdPEUJl1a7+yAo0d5Q9VwIazX2TfhjJ1wXYPwZ1LbtBYhRbxANMDjQMB8mwzOqldm0733+6ZBU15D1B4jG1
LzB0o5veiy1UdMFsowDE6LdNFP20fQq7R+3yQ74UHKVY/wjSn21Z8CtY0cTMZ0aTGn6pIPEjGtZ30J0jRZl1lratB5HecjpaENhSg
PLrem6grdPW0e0tJKMBR+t9dS5mbWMkZk5qmpo3apZH/AE= TIPQC@Q5202-30
nanquill@server2:~/.ssh$
```

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*

```
MINGW64:/c:/Users/TIPQC
TIPQC@QS202-30 MINGW64 ~
$ which git
/mingw64/bin/git
TIPQC@QS202-30 MINGW64 ~
$ |
```

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.


```
TIPQC@QS202-30 MINGW64 ~
$ git --version
git version 2.37.2.windows.2
TIPQC@QS202-30 MINGW64 ~
$
```


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.

Owner * NealianNanquil / Repository name * CPE232_NealianNanquil ✓

Great repository names are short and memorable. Need inspiration? How about [fuzzy-spork?](#)

Description (optional)

☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

☒ **Add a README file**
This is where you can write a long description for your project. [Learn more.](#)

- 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.

SSH keys / Add new

Title

CPE232

Key type

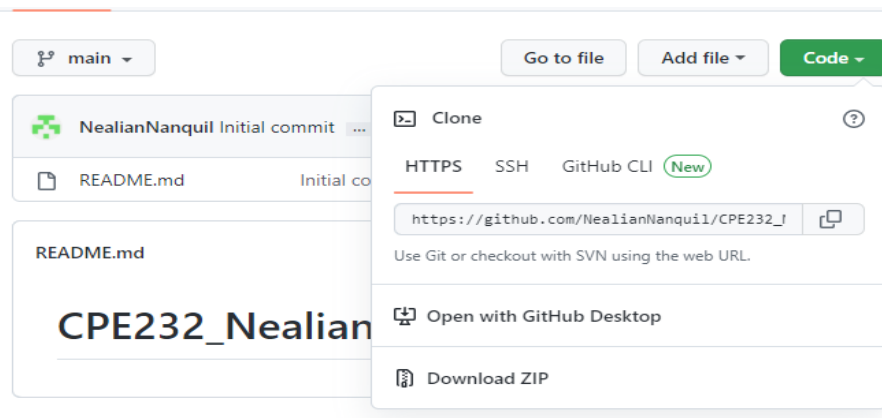
Authentication Key

Key

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQGDc8zbPhLOs9TxgTULi9siqF+SxRgG0eJ+XWUUV
Vexrc1m3W7vqGIK7uUdzXtlmXkEmnnlhKCH2chv9YER4iyYSFHj2B9CyR+3Zv55L7+s1sVB
MTFpGWBF4T4uge9plvxnHxjgF143A7M5TID1o+CcHhvk57YPUNHCEWAIrV8TQO+B0csD
9ZnoHxe/yyelyYPZiXApzFCQkp+4PxytR+sLJGjVZK1e0aAK276j7oT+VSP9RkLaa6Rlvple/
bQDA21Y/ghckEYCdWPjdDjfSt72t8eBdPEUJI1a7+yAo0d5Q9VwIazX2Tfhj1wXYpWZ1LbtB
YhRbxANMdjQMB8mwzOqldmO733+6ZBUI5D1B4jGIIzB0o5veiy1UdMFsowDE6LdNFP20f
Qq7R+3yQ74UHkVY/wjSn21Z8CtY0cTMZ0aTGn6plPEjGtZ3OJ0jRZl1IratB5HecpaENhSgP
Lrem6grdPW0e0tJKMBr+t9dS5mbWMkZk5qmpo3apZH/AE= TIPQC@Q5202-30
```

Add SSH 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.



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

```
TIPQC@Q5202-30 MINGW64 ~
$ git clone https://github.com/NealianNanquil/CPE232_NealianNanquil.git
Cloning into 'CPE232_NealianNanquil'...
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`.

```
TIPQC@Q5202-30 MINGW64 ~
$ ls
'3D Objects'/
AppData/
'Application Data'@
CPE232_NealianNanquil/
```

```
TIPQC@Q5202-30 MINGW64 ~
$ cd CPE232_NealianNanquil

TIPQC@Q5202-30 MINGW64 ~/CPE232_NealianNanquil (main)
$ ls
README.md
```

- 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`

```
TIPQC@Q5202-30 MINGW64 ~/CPE232_NealianNanquil (main)
$ git config --global user.name "Nealian Nanquil"

TIPQC@Q5202-30 MINGW64 ~/CPE232_NealianNanquil (main)
$ git config --global user.email qnbbnanquil01@tip.edu.ph

TIPQC@Q5202-30 MINGW64 ~/CPE232_NealianNanquil (main)
$ cat ~/.gitconfig
[user]
    name = Nealian Nanquil
    email = qnbbnanquil01@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.

```
TIPQC@Q5202-30 MINGW64 ~/CPE232_NealianNanquil (main)
$ nano README.md
```

git

MINGW64:/c/Users/TIPQC/

```
GNU nano 6.4
# CPE232_NealianNanquil
Hello!
```

- 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?

```
TIPQC@Q5202-30 MINGW64 ~/CPE232_NealianNanquil (main)
$ 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.

```
TIPQC@Q5202-30 MINGW64 ~/CPE232_NealianNanquil (main)
$ git add README.md
warning: in the working copy of 'README.md', LF will be replaced by CRLF the next time Git touches it
```

- 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.

```
TIPQC@Q5202-30 MINGW64 ~/CPE232_NealianNanquil (main)
$ git commit -m "Hello gois"
[main ba469aa] Hello gois
1 file changed, 2 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*.

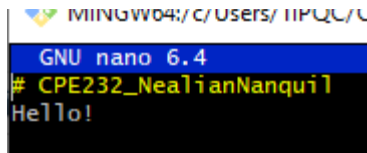
```
TIPQC@Q5202-30 MINGW64 ~/CPE232_NealianNanquil (main)
$ git push origin main
info: please complete authentication in your browser...
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 281 bytes | 281.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/NealianNanquil/CPE232_NealianNanquil.git
f65e7dc..ba469aa 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.

2 lines (2 sloc) | 31 Bytes

CPE232_NealianNanquil

Hello!



```
MINGW64/C:/Users/TIPQC/C
GNU nano 6.4
# CPE232_NealianNanquil
Hello!
```

Reflections:

Answer the following:

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

Using ansible commands, we have connected the local machine and the server using the different commands of sudo. We also saw the installed authorized keys in the two servers.

4. **How important is the inventory file?**

Inventory file is important because this helps us to define the hosts to which command in the playbook can operate.

Conclusions/Learnings:

Overall, this activity helped me to learn different commands in the linux and git. I

learned how to configure the remote servers and local machine using the SSH. Also, I was able to configure different commands from local machine to remote servers. This learnings will help me in my other upcoming activities as well as in my future.