

Name: Aducal John Mark S.	Date Performed: 08 / 30 / 2022
Course/Section: CPE232-CPE31S24	Date Submitted: 08 / 30 / 2022
Instructor: Engr. Jonathan V. Taylar	Semester and SY: 1st Sem SY '22 - '23

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*.

```
MINGW64:/c/Users/Aduca1 PC
Aduca1 PC@MSIDESTOP-1RQNP5K MINGW64 ~
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/Aduca1 PC/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/Aduca1 PC/.ssh/id_rsa
Your public key has been saved in /c/Users/Aduca1 PC/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:5mSeUJw4GYN1L4N8w8maFtVZTomehaov6YtrNdZXz5M Aduca1 PC@MSIDESTOP-1RQNP5K
The key's randomart image is:
+---[RSA 3072]---+
|  o*. =o.      |
| ..* Oo.o      |
| + % =. .      |
| = *  o o      |
| + ..So + o .  |
| . +Bo.+  E    |
| o ++.        |
| ..+          |
| .ooo+.       |
+---[SHA256]-----+
```

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.

```
MINGW64:/c/Users/Aduca1 PC
Aduca1 PC@MSIDESTOP-1RQNP5K MINGW64 ~
$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/Aduca1 PC/.ssh/id_rsa):
/c/Users/Aduca1 PC/.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 /c/Users/Aduca1 PC/.ssh/id_rsa
Your public key has been saved in /c/Users/Aduca1 PC/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:5Dhdv0Kd5zuFdxC6GeEn2y6g5HKzbMCw3tovj9g3g40 Aduca1 PC@MSIDESTOP-1RQNP5K
The key's randomart image is:
+---[RSA 4096]---+
|  . .          |
| . .. o .      |
| . = . o=.o     |
| +o S . +O.o    |
| . o... ++o +   |
| . . B .....o. |
| .+Eo@ .. o.    |
| o.+O== ...     |
+---[SHA256]-----+
Aduca1 PC@MSIDESTOP-1RQNP5K MINGW64 ~
$ |
```

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.

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`.

```
Aduca1 PC@MSIDESTOP-1RQNP5K MINGW64 ~
$ ls -la .ssh
total 40
drwxr-xr-x 1 Aduca1 PC 197121  0 Aug 30 19:22 ./
drwxr-xr-x 1 Aduca1 PC 197121  0 Aug 29 01:36 ../
-rw-r--r-- 1 Aduca1 PC 197121 3401 Aug 30 19:24 id_rsa
-rw-r--r-- 1 Aduca1 PC 197121  754 Aug 30 19:24 id_rsa.pub
-rw-r--r-- 1 Aduca1 PC 197121 1858 Aug 23 02:24 known_hosts
-rw-r--r-- 1 Aduca1 PC 197121  936 Aug 23 02:16 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.

```
Aduca1 PC@MSIDESTOP-1RQNP5K MINGW64 ~
$ ssh-copy-id
Usage: /usr/bin/ssh-copy-id [-h|-?|-f|-n|-s] [-i [identity_file]] [-p port] [-F
alternative ssh_config file] [[-o <ssh -o options>] ...] [user@]hostname
    -f: force mode -- copy keys without trying to check if they are already
installed
    -n: dry run    -- no keys are actually copied
    -s: use sftp   -- use sftp instead of executing remote-commands. Can be
useful if the remote only allows sftp
    -h|-?: print this help
```

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

For Server1:

```
Aduca1 PC@MSIDESTOP-1RQNP5K MINGW64 ~
$ ssh-copy-id -i ~/.ssh/id_rsa jmaduca1@server1
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/c/Users/Aduca1 PC
/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter o
ut any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompte
d now it is to install the new keys
jmaduca1@server1's password:
Permission denied, please try again.
jmaduca1@server1's password:

Number of key(s) added: 1

Now try logging into the machine, with:  "ssh 'jmaduca1@server1'"
and check to make sure that only the key(s) you wanted were added.
```



```
jmaducal@server1: ~/.ssh
jmaducal@server1:~$ cd .ssh
jmaducal@server1:~/.ssh$ ls
authorized_keys
jmaducal@server1:~/.ssh$ cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQADvFLVZh7q6HkmwM0VbqFtCU+0mRiLeNlu+wXnWZUp
h8Ti+uL49AdFE/GIZGLaDde/6d0o1kB8Z+BxVqsig4LHWXonk6IgvNHemHmEo0jFB+ia0+a1uuXBrM5
CkNm7tYoI+ZRQtrY03D1B2+0p0ucV05cMnTq0RqojwyG1IUIgfwYcXcnJKt+bKozZXvQxVnDlQSL
8dZ5C/5tyIVfsDy6uibDM9HpNG9/rfnCtahub8wokD+U2QhZJ2nnm6EhCO0gli6ddP6WHst7NjdYj1
etEcruc4fk1FmpNx5yWGOC6ju1VoB3pkmISKfzAc0XfTc+SuhK39SnJNxz3Zco9M5zqaWbfasckKPaU
5Y0h/eu+YpXlAA7dAH/3PNAopWFZZZcJ0rjN0wpL4M202zSQMqy5gMunCHBEJo1rRbhAFN1y/GSVKOW
8h4kH1x60eUHDAPuoirDM793BBEXWIQHx6r3dyx6nv8DCSUKk3H+h11u8bFaupkIXIFcUm5TjrRejU
SrreQlyzkzfAwlpdJC2HJBnYNj/xXDOO/ToFZSNA8y3NX5PMLHD94E33E0iQ/aDkC7bu91M9tmsq0+6
YZwuzfqysgy0Qpjxf9tcuAcjwISqvFPpg8uvKk/gd0heISndV2VICqMfry9PRVkv1+wIRVFzU4AYBkJ
nwFwo4k5eBQ== Aduca1 PC@MSIDESKTOP-1RQNP5K
Aduca1 PC@MSIDESKTOP-1RQNP5K MINGW64 ~/.ssh
$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQADvFLVZh7q6HkmwM0VbqFtCU+0mRiLeNlu+wXnWZUp
h8Ti+uL49AdFE/GIZGLaDde/6d0o1kB8Z+BxVqsig4LHWXonk6IgvNHemHmEo0jFB+ia0+a1uuXBrM5CkNm7
tYoI+ZRQtrY03D1B2+0p0ucV05cMnTq0RqojwyG1IUIgfwYcXcnJKt+bKozZXvQxVnDlQSL8dZ5C/5ty
IVfsDy6uibDM9HpNG9/rfnCtahub8wokD+U2QhZJ2nnm6EhCO0gli6ddP6WHst7NjdYj1etEcruc4fk1F
mpNx5yWGOC6ju1VoB3pkmISKfzAc0XfTc+SuhK39SnJNxz3Zco9M5zqaWbfasckKPaU5Y0h/eu+YpXlAA7
dAH/3PNAopWFZZZcJ0rjN0wpL4M202zSQMqy5gMunCHBEJo1rRbhAFN1y/GSVKOW8h4kH1x60eUHDAPuoir
rdM793BBEXWIQHx6r3dyx6nv8DCSUKk3H+h11u8bFaupkIXIFcUm5TjrRejUSrreQlyzkzfAwlpdJC2HJ
bNynj/xXDOO/ToFZSNA8y3NX5PMLHD94E33E0iQ/aDkC7bu91M9tmsq0+6YZwuzfqysgy0Qpjxf9tcuAcj
wISqvFPpg8uvKk/gd0heISndV2VICqMfry9PRVkv1+wIRVFzU4AYBkJnwFwo4k5eBQ== Aduca1 PC@MSI
DESKTOP-1RQNP5K
```

For Server2:

```
Aduca1 PC@MSIDESKTOP-1RQNP5K MINGW64 ~/.ssh
$ ssh-copy-id -i ~/.ssh/id_rsa jmaducal@server2
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/c/Users/Aduca1 PC/.ss
h/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out a
ny that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted no
w it is to install the new keys
jmaducal@server2's password:

Number of key(s) added: 1

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

```
jmaducal@server2: ~/.ssh
jmaducal@server2:~$ cd .ssh
jmaducal@server2:~/.ssh$ ls
authorized_keys
jmaducal@server2:~/.ssh$ cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQADvFLVZh7q6HkmwM0VbqFtCU+0mRiLeNlu+wXnWZUp
h8Ti+uL49AdFE/GIZGLaDde/6d0o1kB8Z+BxVqsig4LHWXonk6IgvNHemHmEo0jFB+ia0+a1uuXBrM5
CkNm7tYoI+ZRQtrY03D1B2+0p0ucV05cMnTq0RqojwyG1IUIgfwYcXcnJKt+bKozZXvQxVnDlQSL
8dZ5C/5tyIVfsDy6uibDM9HpNG9/rfnCtahub8wokD+U2QhZJ2nnm6EhCO0gli6ddP6WHst7NjdYj1
etEcruc4fk1FmpNx5yWGOC6ju1VoB3pkmISKfzAc0XfTc+SuhK39SnJNxz3Zco9M5zqaWbfasckKPaU
5Y0h/eu+YpXlAA7dAH/3PNAopWFZZZcJ0rjN0wpL4M202zSQMqy5gMunCHBEJo1rRbhAFN1y/GSVKOW
8h4kH1x60eUHDAPuoirDM793BBEXWIQHx6r3dyx6nv8DCSUKk3H+h11u8bFaupkIXIFcUm5TjrRejU
SrreQlyzkzfAwlpdJC2HJBnYNj/xXDOO/ToFZSNA8y3NX5PMLHD94E33E0iQ/aDkC7bu91M9tmsq0+6
YZwuzfqysgy0Qpjxf9tcuAcjwISqvFPpg8uvKk/gd0heISndV2VICqMfry9PRVkv1+wIRVFzU4AYBkJ
nwFwo4k5eBQ== Aduca1 PC@MSIDESKTOP-1RQNP5K
```

```
Aduca1 PC@MSIDESTOP-1RQNP5K MINGW64 ~/.ssh
$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDvF1Vzh7q6HkmwM0VbqFtCU+0mRiLeN1u+wXnWZUph8Ti+u
L49AdFE/GIZGLaDde/6d0o1k88Z+BxVqsig4LHWXonk6IgvNHemHmEo0jFB+ia0+aluuXBrM5CkNm7tYoI+Z
RQtrY03D1B2+0p0ucV05cMnTqORqojwyG1IUIgfwYcXcnJKt+bKozZXvQxVnD1QS18dZ5C/5tyIVfsDy6ui
bDM9HpNG9/rfnCtahub8wokD+U2QhZJ2nnm6EhCOOgli6ddP6WHst7NjdYj1etEcruc4fk1FmpNx5yWGOC6j
u1VoB3pkmISKfzAc0XfTc+SuhK39SnJNxz3Zco9M5zqawbfasckKPaU5Y0h/eu+YpX1AA7dAH/3PNAopWFZZZ
cJ0rjN0wpL4M202zSQMqy5gMunCHBEJo1rRbhaFN1y/GSVKOW8h4kH1x60eUHDAPuOIrDm793BBEXWIQhX6r
3dyx6nv8DCSuk3H+h11u8bFaupkIXIfcUm5TjrRejUSrreQlyzkzfAwlpdJC2HJbNYnj/xXDOO/ToFZSNA8y
3NX5PMLHD94E33E0iQ/aDkC7bu91M9tmsq0+6YZwuzfqysgy0Qpjxf9tcuAcjwISqvPpg8uvKk/gd0heISnd
V2VICqMfry9PRVkv1+wIRVFzU4AYBkJnwFwo4k5eBQ== Aduca1 PC@MSIDESTOP-1RQNP5K
```

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?

Not, because we use the SSH Public Key Authentication to secure login to servers, Instead of login using password we use a SSH Public key to access the server 1 & 2.

For Server1:

```
jmaduca1@server1: ~
Aduca1 PC@MSIDESTOP-1RQNP5K MINGW64 ~/.ssh
$ ssh jmaduca1@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.

The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Last login: Tue Aug 23 02:22:03 2022 from 192.168.56.1
jmaduca1@server1:~$ |
```

For Server2:

```
jmaduca1@server2: ~
Aduca1 PC@MSIDESTOP-1RQNP5K MINGW64 ~/.ssh
$ ssh jmaduca1@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.

The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Last login: Tue Aug 23 02:24:12 2022 from 192.168.56.1
jmaduca1@server2:~$ |
```


Reflections:

Answer the following:

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

SSH or Secure Shell is a network protocol that provides secure connection for client to have a remotely secure and authenticated access to servers.

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

We can verify by checking the `authorized_keys` file in the servers using the command `cat authorized_keys` in `.ssh` directory of servers. The `authorized_key` file contains encrypted message of public key.

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**

```
jmaducal@workstation:~$ which git
jmaducal@workstation:~$ sudo apt install git
[sudo] password for jmaducal:
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 4 not upgraded.
Need to get 4,110 kB of archives.
After this operation, 20.9 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 1:2.34.1-1ubuntu1.4 [952 kB]
Get:3 http://ph.archive.ubuntu.com/ubuntu jammy-updates/main amd64 git amd64 1:2.34.1-1ubuntu1.4 [3,131 kB]
Fetched 4,110 kB in 5s (847 kB/s)
Selecting previously unselected package liberror-perl.
(Reading database ... 198402 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.
```

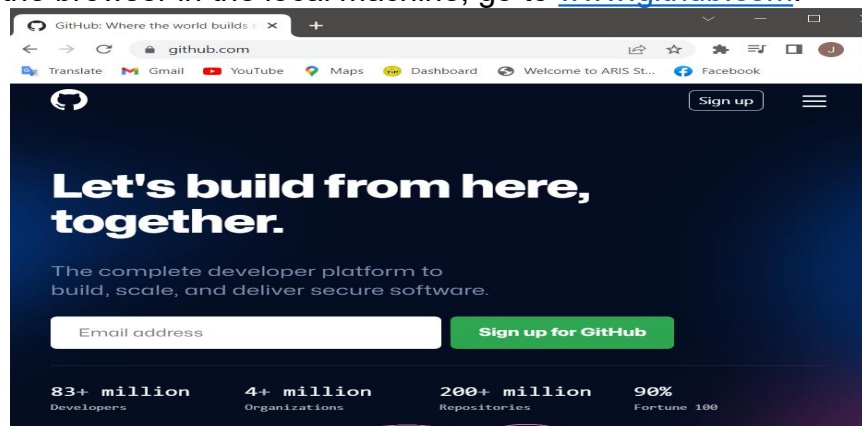
2. After the installation, issue the command **which git** again. The directory of git is usually installed in this location: **user/bin/git**.

```
jmaducal@workstation:~$ which git
/usr/bin/git
```

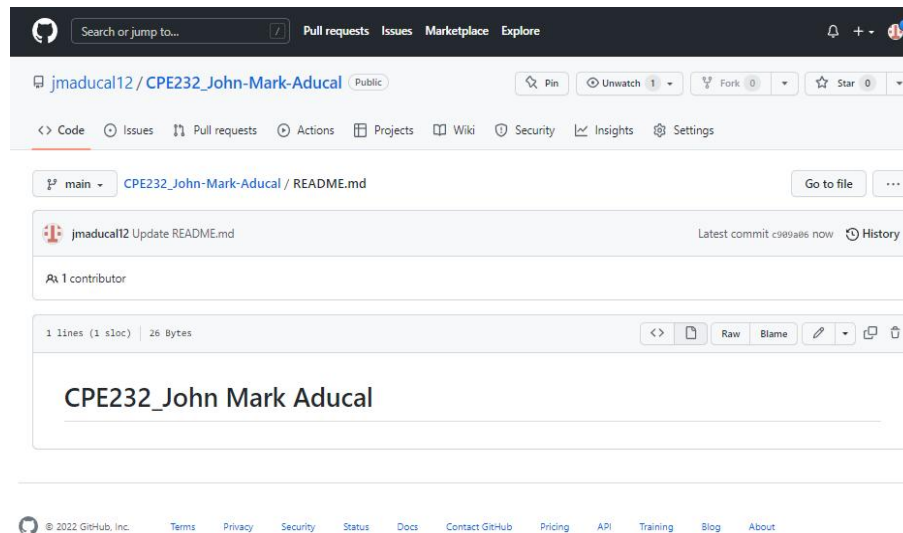
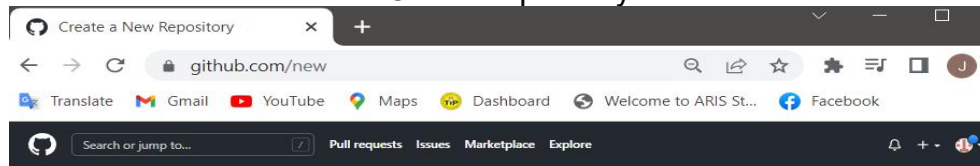
3. The version of git installed in your device is the latest. Try issuing the command **git --version** to know the version installed.

```
jmaducal@workstation:~$ git --version
git version 2.34.1
```

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.

 **jmaducal12**
Your personal account

[Go to your personal profile](#)

[Public profile](#)
[Account](#)
[Appearance](#)
[Accessibility](#)
[Notifications](#)

[Access](#)
[Billing and plans](#)
[Emails](#)
[Password and authentication](#)
[SSH and GPG keys](#)
[Organizations](#)
[Moderation](#)

[Code, planning, and automation](#)
[Repositories](#)

SSH keys / Add new

Title
CPE232 key

Key type
Authentication Key

Key
Begins with 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed25519', 'sk-ecdsa-sha2-nistp256@openssh.com', or 'sk-ssh-ed25519@openssh.com'

Add 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.

```
jmaducal@workstation:~$ cat .ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCAQCgXnIZZzc/jZeCHBZK7JcXV6IfEL/FDckJkwt43k
fjVFS1Mza4guT/2wEvqYQFxmFa8eIDjo3gpr6Kwv8mR5Ve4VNLr2e/BgEopQ22TmfBW00ewXlHjmie
4AVIN4DAmCWxJV52Ly3hZ1aj7fKikg3kke0Sx0TSSaIo0FvUse2WMDqaXmFEr7FkUBMD9DvlykLq2C/
Lg+7wQLlBC4Vm+spRQ04WAAf/uwEK7FRlYviBH2pinq1vr9v8WZIHsFslDvMk2N7xE2x00CRgyxliQh
8ME+lBMfCn6sJJP4cjfyufvMbX3ziHQb6QWsc/TYSQzjR3ZFKZKIP7vhLLYyoHZT+VIPsggnkjgmBF
fmKVd5Tfcffpfs44WptNvyt0cbZsAMarT2MB1hscUSoF6giWrjfbNwZZn63p+jN8CbJvAR8ppKLYbon
ClbaZGM3w08mbtXp4YvY1GERGLZ5YesfN68u7ZV37hJ82QuPG6Ij80KQ0FthYtyigFveRdCXrWMIJj0
METd4tk8gNFS/CaeyHrjnF1X8v7H9QsrZEpTSQvX5qtrjaolhK6yh8++A5fMhCGWgVNsCr741cwjsd
nZ8kdSlBj8REVabRnkoB+nwyT/KPwWfthKRit9YbvFTQv/3CikKy1R288v4f8ZT217M1A6+v2GjyrRb
ETW/mugBeXQ== jmaducal@workstation
```

SSH keys / Add new

Title

CPE232 key

Key type

Authentication Key

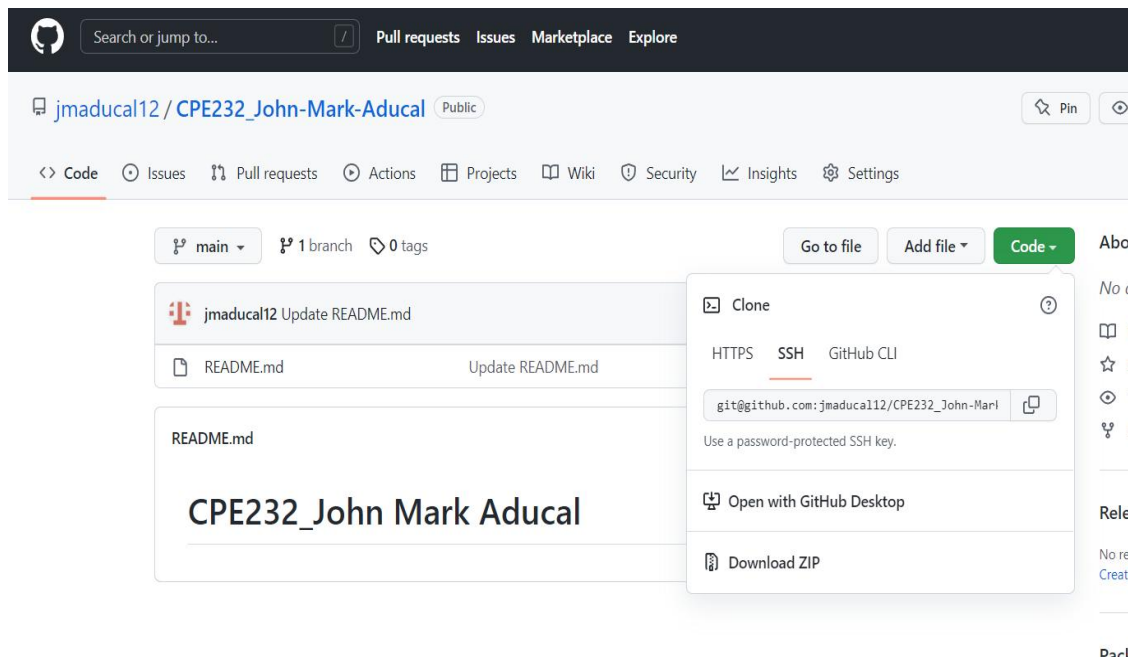
Key

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQACGAXNIZZZc/jzeC
HBZK7JCV6lfeL/FDCKJkWct43k
fiVFS1Mzagut/2wEvgYQFxmMfdeIDjo3gpr6KWV8MR5Ve
4VNlr2e/BgEOPQ22TmfBWOEEwXIHjmie
4AVINADAMCWJ52Ly3hz1ajzfKikg3kkeOSXOTSsaIOOF
VUSe 2WMDqaXmFer7FKUBMD9Dvlyklq2C/
Lg+7WOLLBC4Vm+SpRQO4WAAf/UWEK7FRlYviBH2pinqi
vr 9v8WZIHsFs LDVMK2N7xE2XOOCRgyxligh
```

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.

The screenshot shows the GitHub interface for a repository named 'jvtaylor-cpe / CPE302_yourname'. The 'Code' dropdown menu is open, displaying three options: 'HTTPS', 'SSH', and 'GitHub CLI'. The 'SSH' option is highlighted with a yellow circle. Below the 'SSH' option, the text 'git@github.com:jvtaylor-cpe/CPE302_yourname' is visible. The repository name 'CPE302_yourname' is also visible at the bottom of the page.



- 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.

```
jmaducal@workstation:~$ git clone git@github.com:jmaducal12/CPE232_John-Mark-Aducual.git
Cloning into 'CPE232_John-Mark-Aducual'...
The authenticity of host 'github.com (20.205.243.166)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvV6TuJJhbpZisF/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: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (6/6), done.
```

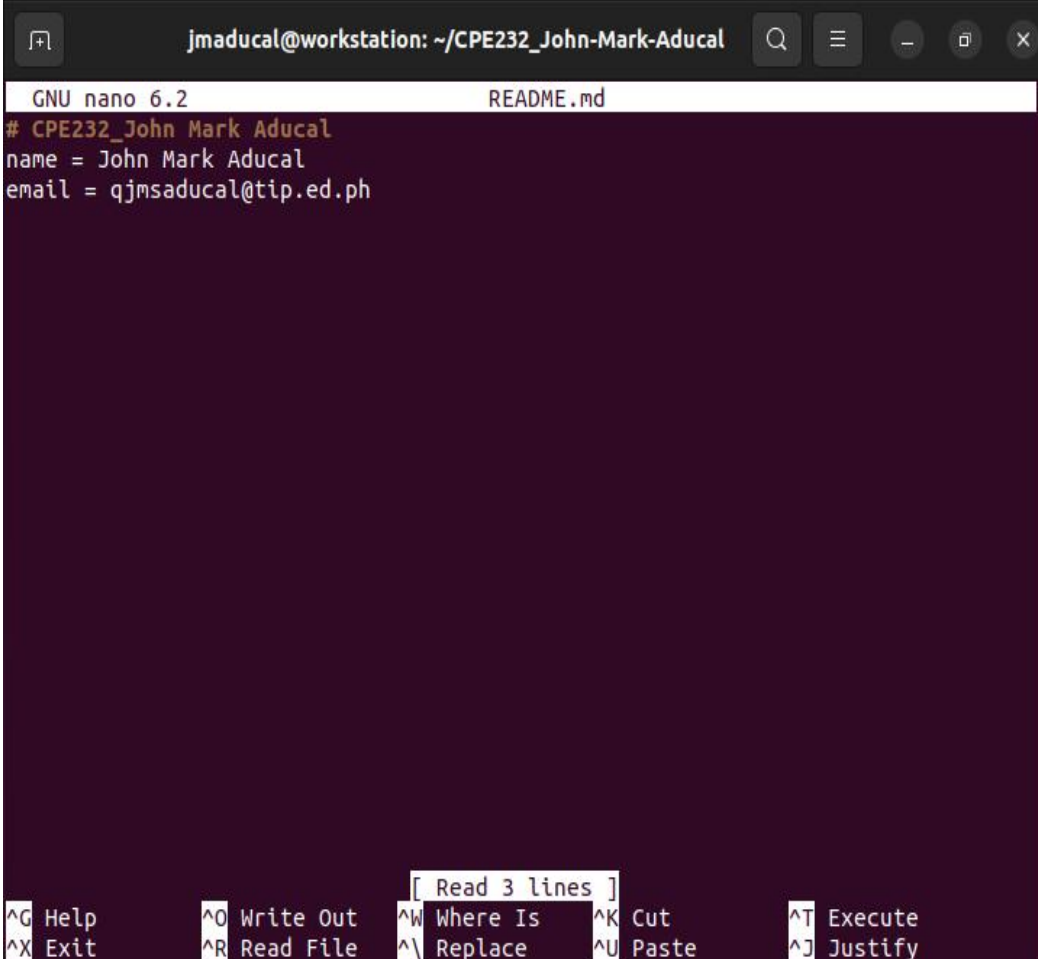
- f. To verify that you have cloned the GitHub repository, issue the command `ls`. Observe that you have the `CPE232_yourname` in the list of your directories. Use `CD` command to go to that directory and `LS` command to see the file `README.md`.

```
jmaducal@workstation:~$ ls
CPE232_John-Mark-Aducual  Documents  Music      Public  Templates
Desktop                  Downloads  Pictures  snap    Videos
jmaducal@workstation:~$ cd CPE232_John-Mark-Aducual
jmaducal@workstation:~/CPE232_John-Mark-Aducual$ 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`

```
jmaducal@workstation:~/CPE232_John-Mark-Aducal$ git config --global user.name "John Mark Aducal"
jmaducal@workstation:~/CPE232_John-Mark-Aducal$ git config --global user.email qjmsaducal@tip.ed.ph
jmaducal@workstation:~/CPE232_John-Mark-Aducal$ cat ~/.gitconfig
[user]
  name = John Mark Aducal
  email = qjmsaducal@tip.ed.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.



```
jmaducal@workstation: ~/CPE232_John-Mark-Aducal
GNU nano 6.2 README.md
# CPE232_John Mark Aducal
name = John Mark Aducal
email = qjmsaducal@tip.ed.ph

[ Read 3 lines ]
^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify
```

- 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?

```
jmaducal@workstation:~/CPE232_John-Mark-Aducal$ 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.

```
jmaducal@workstation:~/CPE232_John-Mark-Aducal$ 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.

```
jmaducal@workstation:~/CPE232_John-Mark-Aducal$ git commit -m "personalinfo"
[main bc1844d] personalinfo
1 file changed, 2 insertions(+)
```

- 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*.

```
jmaducal@workstation:~/CPE232_John-Mark-Aducal$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 309 bytes | 309.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:jmaducal12/CPE232_John-Mark-Aducal.git
c909a06..bc1844d  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.

The screenshot shows a GitHub repository page. At the top, there's a search bar and navigation links for Pull requests, Issues, Marketplace, and Explore. Below this, the repository name 'jmaducal12 / CPE232_John-Mark-Aducal' is displayed with a 'Public' label. A navigation bar includes links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. The main content area shows the 'main' branch with 1 branch and 0 tags. It lists files: 'John Mark Aducal personalinfo' (committed 4 minutes ago) and 'README.md' (committed 4 minutes ago). The 'README.md' file is expanded, showing the text: 'CPE232_John Mark Aducal' followed by 'name = John Mark Aducal email = qjmsaducal@tip.ed.ph'. On the right, there's an 'About' section with 'No description', a 'Readme' icon, 0 stars, 1 watching, and 0 forks. Below that are 'Releases' (No releases published) and 'Packages'.

```
jmaducal@workstation:~/CPE232_John-Mark-Aducal$ cat README.md
# CPE232_John Mark Aducal
name = John Mark Aducal
email = qjmsaducal@tip.ed.ph
```

```
jmaducal@workstation:~/CPE232_John-Mark-Aducal$ cat ~/.gitconfig
[user]
    name = John Mark Aducal
    email = qjmsaducal@tip.ed.ph
```

Reflections:

Answer the following:

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

Ansible has a default inventory file (etc/ansible/hosts) used to define which remote servers it will be managing. Our public SSH key is located in authorized_keys on remote systems.

4. How important is the inventory file?

The Inventory file in Ansible stores information about physical and virtual servers, network devices, cloud instances and much more. In fact any device that provide SSH access can be added to the inventory file.

Conclusions / Learning:

After doing this activity, I have learned how to configure client and servers to connect via SSH using KEY Instead of using password. Creating a public and private key. Make Account on GitHub and to create Git Repository, managing or altering the content of file such as README.md using the local machine.