Name: Kazuki A. Ogata	**Date Performed:** August 28, 2023
Course/Section: CPE 232 - CPE31S5	**Date Submitted:** August 28, 2023
Instructor: Engr. Roman Richard	**Semester and SY:** 1st semester S.Y 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 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

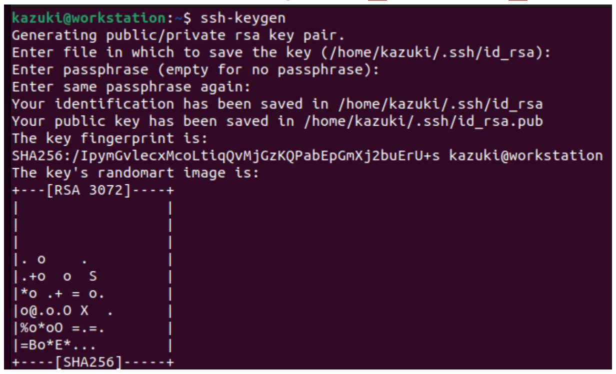
The SSH protocol uses public key cryptography for authenticating hosts and users. The authentication keys, called SSH keys, are created using the keygen program.

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.

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*.

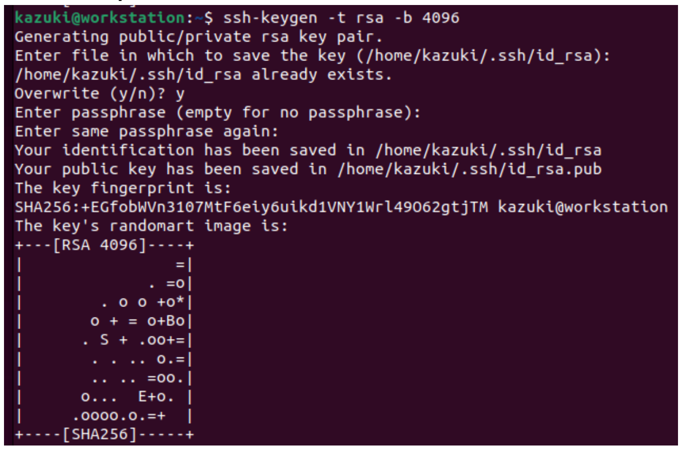


```
kazuki@workstation:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/kazuki/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/kazuki/.ssh/id_rsa
Your public key has been saved in /home/kazuki/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:/IpymGvlecxMcoLtiqQvMjGzKQPabEpGmXj2buErU+s kazuki@workstation
The key's randomart image is:
+---[RSA 3072]----+
|                 |
|                 |
|                 |
|. o    .         |
|.+o  o  S        |
|*o .+ = o.       |
|o@.o.O X  .      |
|%o*oO =.=.       |
|=Bo*E*...        |
+----[SHA256]-----+
```

Task 1.1: Generating Key Pair

The command "ssh-keygen" is used to generate an SSH key pair that will be used for authentication. Also, the public key is shared with the remote servers (servers 1 and 2) that allows secure authentication. Lastly, the private key, it is used to decrypt authentication requests from servers. This command enhances security, reduces the risks of unauthorized access, and allows us to access remote systems securely.

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.

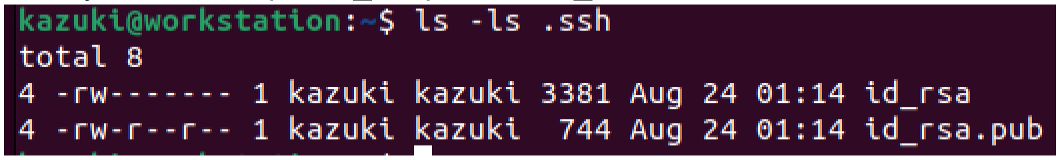


```
kazuki@workstation:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/kazuki/.ssh/id_rsa):
/home/kazuki/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/kazuki/.ssh/id_rsa
Your public key has been saved in /home/kazuki/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:+EGfobWVn3107MtF6eiy6uikd1VNY1Wrl49O62gtjTM kazuki@workstation
The key's randomart image is:
+---[RSA 4096]----+
|                =|
|             . =o|
|        . o o +o*|
|       o + = o+Bo|
|      . S + .oo+=|
|       . . .. o.=|
|       .. .. =oo.|
|      o...  E+o. |
|     .oooo.o.=+  |
+----[SHA256]-----+
```

Task 1.2: Selecting Algorithm and Key Size

The command "ssh-keygen -t rsa -b 4096" is used to select the algorithm and key size. The "-t" flag specifies the "key type" rsa, rsa is used to secure data transmission and authentication. The "-b" flag on the other hand is used to specify the "key size" to 4096. In summary, it generates an rsa key type with a 4096 key size.

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. Verify that you have created the key
4. 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.



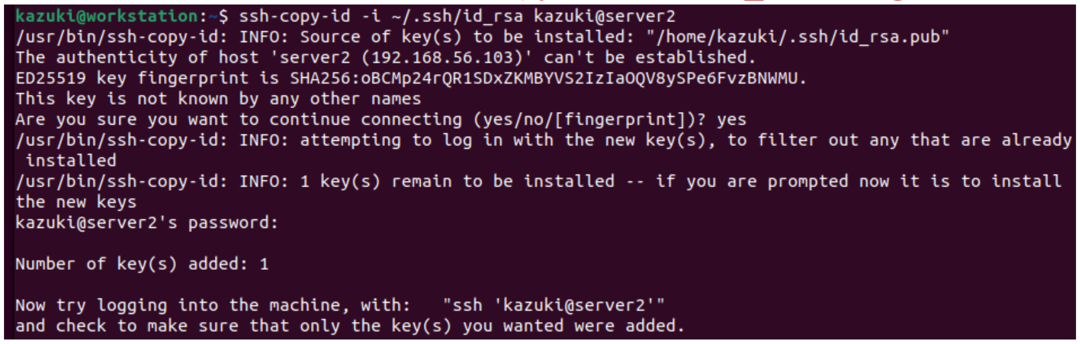
```
kazuki@workstation:~$ ls -ls .ssh
total 8
4 -rw------- 1 kazuki kazuki 3381 Aug 24 01:14 id_rsa
4 -rw-r--r-- 1 kazuki kazuki  744 Aug 24 01:14 id_rsa.pub
```

Task 1.3-4: Verification of Generated Key Pair

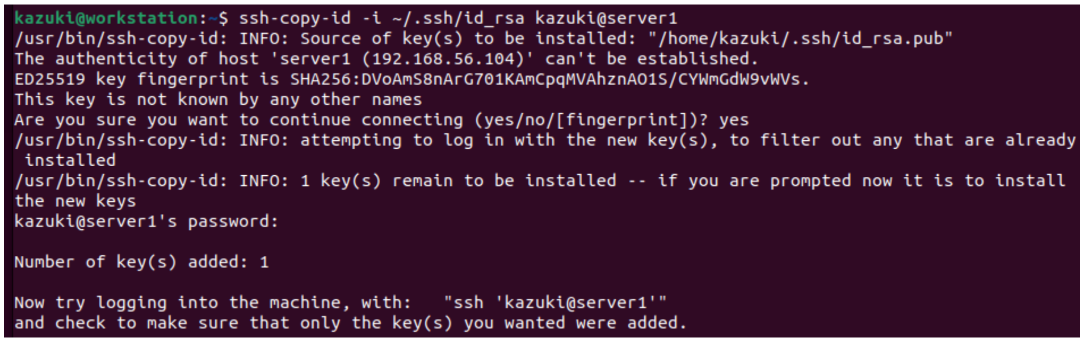
When generating a key pair, it will ask you to enter the file location where to save the key pair, and then asks for a passphrase, a passphrase provides "additional" security by encrypting the key, so that it will not be used even when someone gets the private key file. The "ls -la .ssh" command lists the contents of the .shh file, it displays the permission, ownership, and timestamps.

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*



```
kazuki@workstation:~$ ssh-copy-id -i ~/.ssh/id_rsa kazuki@server2
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/kazuki/.ssh/id_rsa.pub"
The authenticity of host 'server2 (192.168.56.103)' can't be established.
ED25519 key fingerprint is SHA256:oBCMp24rQR1SDxZKMBYVS2IzIaOQV8ySPe6FvzBNWMU.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? 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 prompted now it is to install
the new keys
kazuki@server2's password:

Number of key(s) added: 1

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

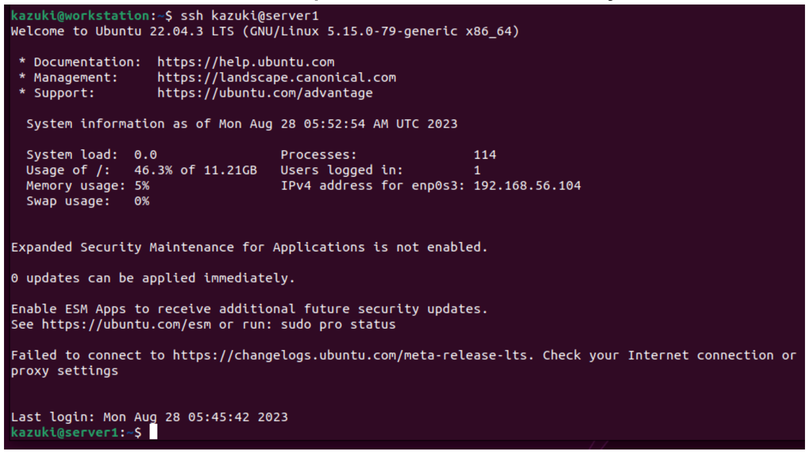


```
kazuki@workstation:~$ ssh-copy-id -i ~/.ssh/id_rsa kazuki@server1
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/kazuki/.ssh/id_rsa.pub"
The authenticity of host 'server1 (192.168.56.104)' can't be established.
ED25519 key fingerprint is SHA256:DVoAmS8nArG701KAmCpqMVAhznAO1S/CYWmGdW9vWVs.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? 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 prompted now it is to install
the new keys
kazuki@server1's password:

Number of key(s) added: 1

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

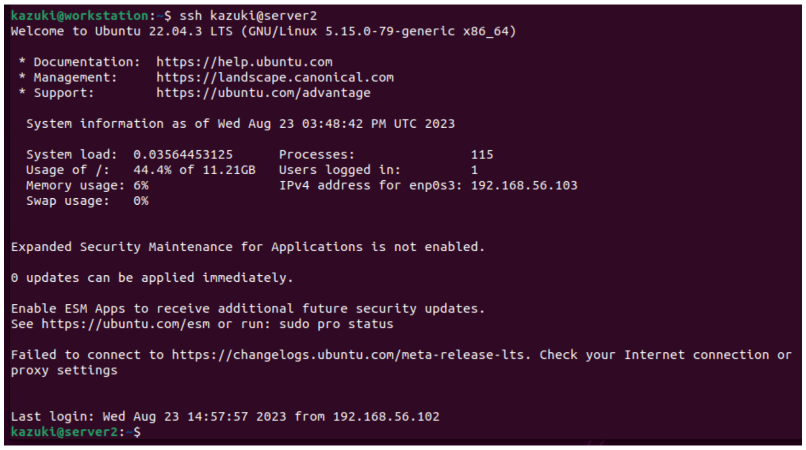
Task 2.1-2: Public Key Authentication

The command "ssh-copy-id" is used to copy the ssh public key to remote servers (servers 1 and 2). The "-i" flag specifies the path to the private key file that I used for authentication. With this, we can log in to servers 1 and 2 securely using the private key.

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?



```
kazuki@workstation:~$ ssh kazuki@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 Mon Aug 28 05:52:54 AM UTC 2023

  System load:  0.0               Processes:               114
  Usage of /:   46.3% of 11.21GB   Users logged in:         1
  Memory usage: 5%                IPv4 address for enp0s3: 192.168.56.104
  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

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or
proxy settings


Last login: Mon Aug 28 05:45:42 2023
kazuki@server1:~$
```



```
kazuki@workstation:~$ ssh kazuki@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 Wed Aug 23 03:48:42 PM UTC 2023

  System load:  0.03564453125      Processes:               115
  Usage of /:   44.4% of 11.21GB   Users logged in:         1
  Memory usage: 6%                 IPv4 address for enp0s3: 192.168.56.103
  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

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or
proxy settings


Last login: Wed Aug 23 14:57:57 2023 from 192.168.56.102
kazuki@server2:~$
```

Task 2.3-4: SSH Verification with Server 1 and Server 2

The public key that I copied to the remote servers (servers 1 and 2) will allow the local machine to prove its identity using the digital signature during the ssh connection process. With this, I used the ssh user@host to verify that I can ssh with servers 1 and 2 and it was successful. What I notice is, it is different in Activity 1, in activity 1, it asks if I want to continue and asks for a password. While here in activity 2, It just connected me to servers 1 and 2 without asking if I want to continue or for a password, I think it is because of the key I created earlier with rsa. This process is based on cryptographic keys and digital signatures, so it removes the need for a password.

Reflections:
Answer the following:
1. How will you describe the ssh-program? What does it do?
   - SSH is a protocol that allows us to access systems securely over an unsecured network, it also encrypts data and offers authentication methods.

2. How do you know that you already installed the public key to the remote servers?
   - We can verify that we already installed the public key to the remote servers by establishing SSH connection without asking for a password. Basically, the remote servers will recognize the public key and allow us to access them without asking for a password.

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**).

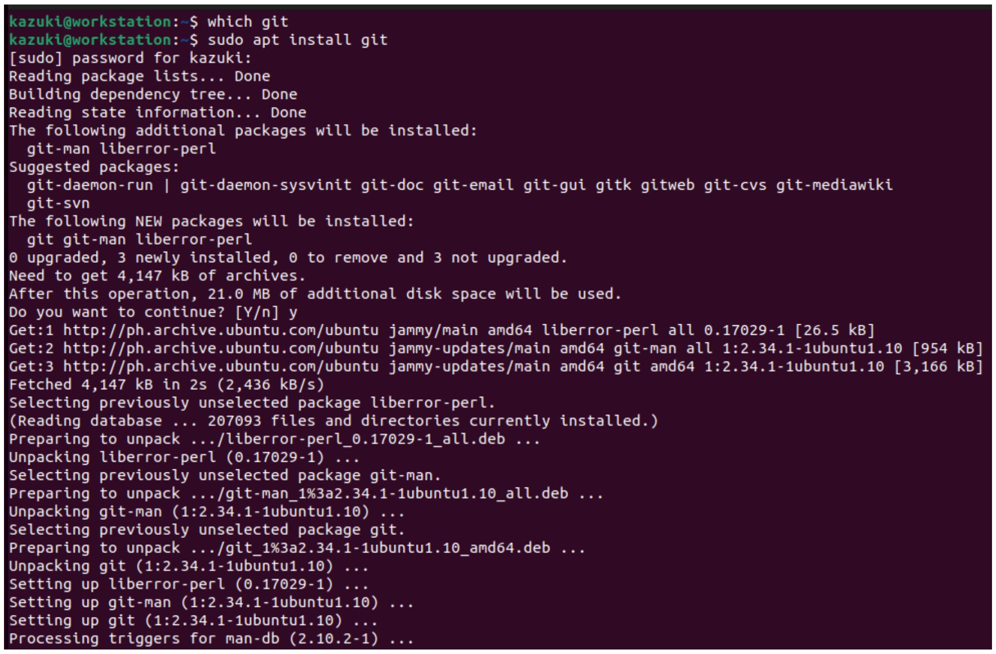
Set up Git

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

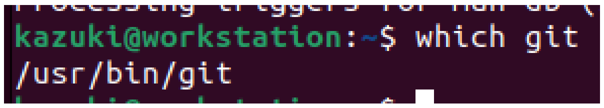
Task 3: Set up the Git Repository

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*



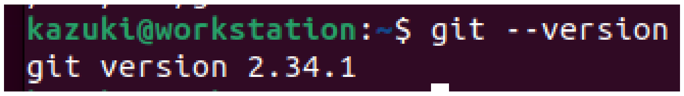
```
kazuki@workstation:~$ which git
kazuki@workstation:~$ sudo apt install git
[sudo] password for kazuki:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  git-man liberror-perl
Suggested packages:
  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 3 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 (2,436 kB/s)
Selecting previously unselected package liberror-perl.
(Reading database ... 207093 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*.



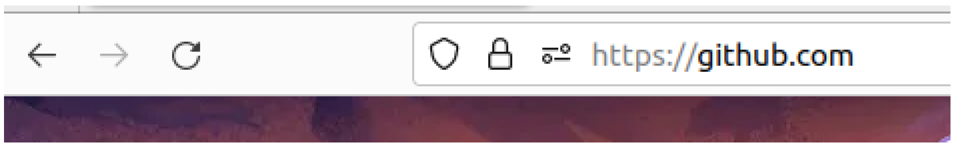
```
kazuki@workstation:~$ which git
/usr/bin/git
```

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



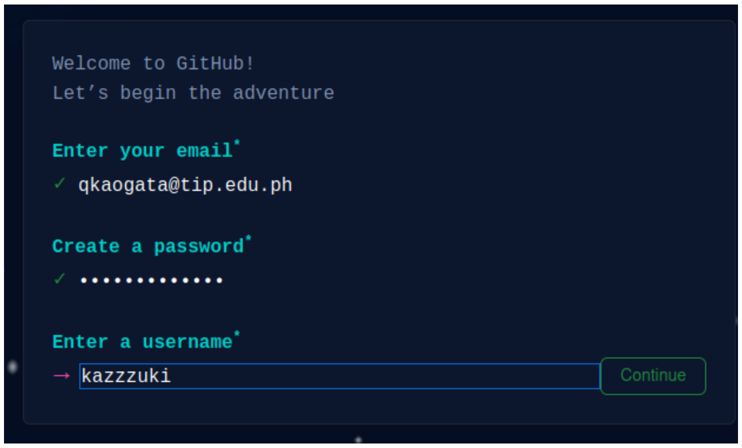
```
kazuki@workstation:~$ git --version
git version 2.34.1
```

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



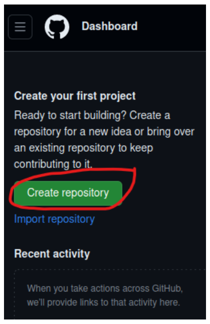
https://github.com

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



```
Welcome to GitHub!
Let's begin the adventure

Enter your email*
✓ qkaogata@tip.edu.ph

Create a password*
✓ •••••••••••••

Enter a username*
→ kazzzuki
```

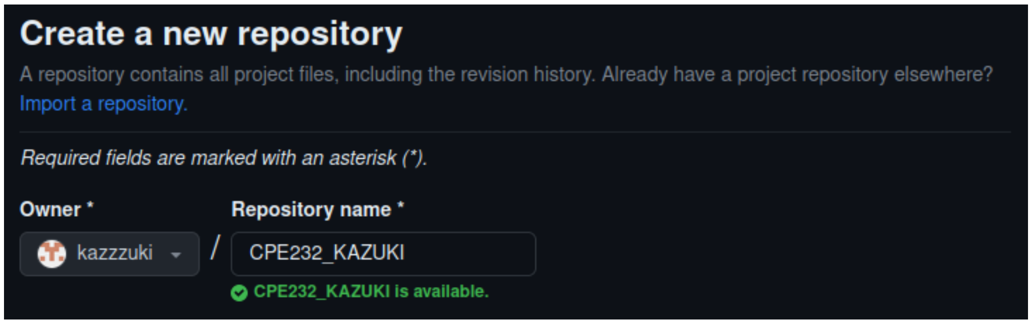
   a. Create a new repository and name it as CPE232_yourname. Check Add a README file and click Create repository.



Create your first project

Ready to start building? Create a repository for a new idea or bring over an existing repository to keep contributing to it.

When you take actions across GitHub, we'll provide links to that activity here.



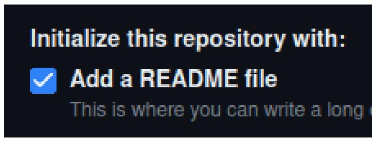
Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Required fields are marked with an asterisk (*).

Owner *: kazzzuki / Repository name *: CPE232_KAZUKI

CPE232_KAZUKI is available.

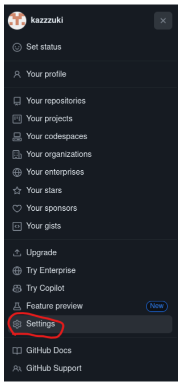


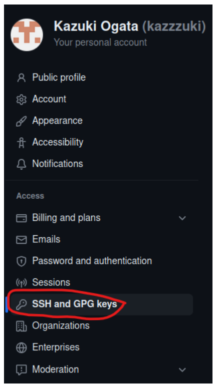
Initialize this repository with:

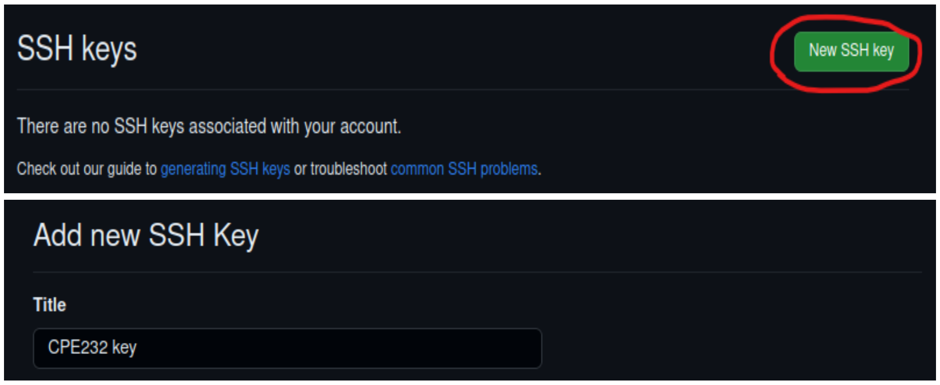
☑ Add a README file

This is where you can write a long

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





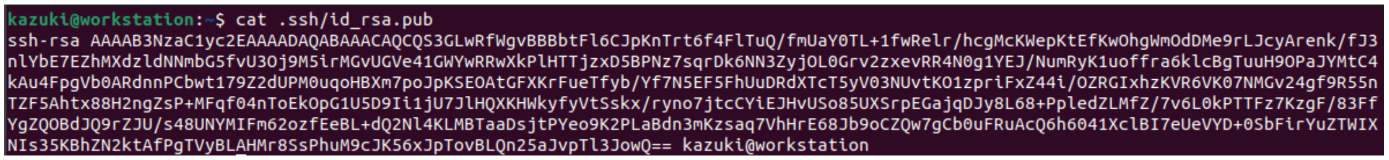


There are no SSH keys associated with your account.

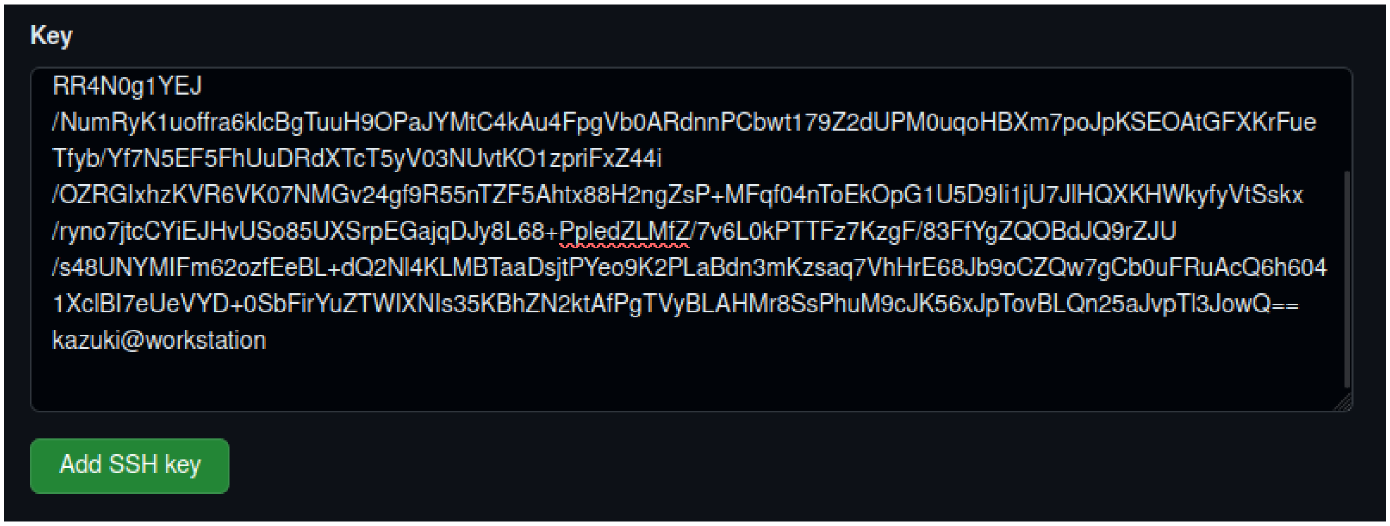
Check out our guide to generating SSH keys or troubleshoot common SSH problems.

CPE232 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.

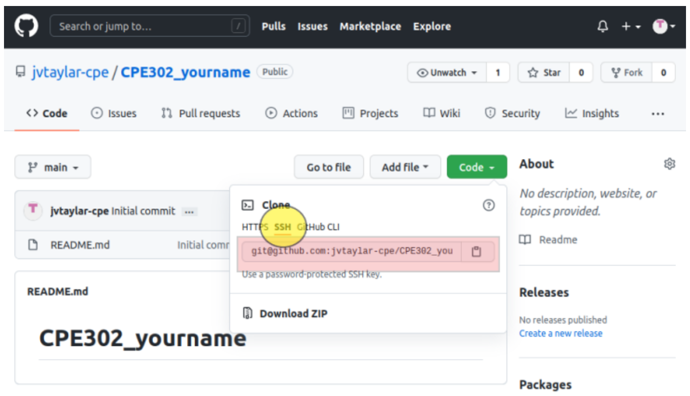


```
kazuki@workstation:~$ cat .ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQCQS3GLwRfWgvBBBbtFl6CJpKnTrt6f4FlTuQ/fmUaY0TL+1fwRelr/hcgMcKWepKtEfKwOhgWmOdDMe9rLJcyArenk/fJ3
nlYbE7EZhMXdzldNNmbG5fvU3Oj9M5irMGvUGVe41GWYwRRwXkPlHTTjzxD5BPNz7sqrDk6NN3ZyjOL0Grv2zxevRR4N0g1YEJ/NumRyK1uoffra6klcBgTuuH9OPaJYMtC4
kAu4FpgVb0ARdnnPCbwt179Z2dUPM0uqoHBXm7poJpKSEOAtGFXKrFueTfyb/Yf7N5EF5FhUuDRdXTcT5yV03NUvtKO1zpriFxZ44i/OZRGIxhzKVR6VK07NMGv24gf9R55n
TZF5Ahtx88H2ngZsP+MFqf04nToEkOpG1U5D9Ii1jU7JlHQXKHWkyfyVtSskx/ryno7jtcCYiEJHvUSo85UXSrpEGajqDJy8L68+PpledZLMfZ/7v6L0kPTTFz7KzgF/83Ff
YgZQOBdJQ9rZJU/s48UNYMIFm62ozfEeBL+dQ2Nl4KLMBTaaDsjtPYeo9K2PLaBdn3mKzsaq7VhHrE68Jb9oCZQw7gCb0uFRuAcQ6h6041XclBI7eUeVYD+0SbFirYuZTWIX
NIs35KBhZN2ktAfPgTVyBLAHMr8SsPhuM9cJK56xJpTovBLQn25aJvpTl3JowQ== kazuki@workstation
```

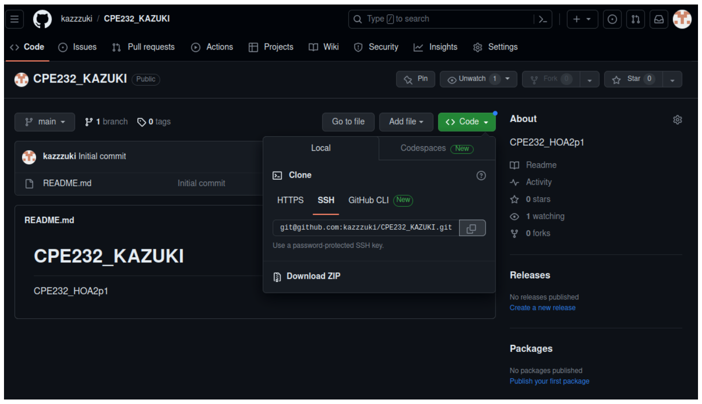


```
RR4N0g1YEJ
/NumRyK1uoffra6klcBgTuuH9OPaJYMtC4kAu4FpgVb0ARdnnPCbwt179Z2dUPM0uqoHBXm7poJpKSEOAtGFXKrFue
Tfyb/Yf7N5EF5FhUuDRdXTcT5yV03NUvtKO1zpriFxZ44i
/OZRGIxhzKVR6VK07NMGv24gf9R55nTZF5Ahtx88H2ngZsP+MFqf04nToEkOpG1U5D9Ii1jU7JlHQXKHWkyfyVtSskx
/ryno7jtcCYiEJHvUSo85UXSrpEGajqDJy8L68+PpledZLMfZ/7v6L0kPTTFz7KzgF/83FfYgZQOBdJQ9rZJU
/s48UNYMIFm62ozfEeBL+dQ2Nl4KLMBTaaDsjtPYeo9K2PLaBdn3mKzsaq7VhHrE68Jb9oCZQw7gCb0uFRuAcQ6h604
1XclBI7eUeVYD+0SbFirYuZTWIXNIs35KBhZN2ktAfPgTVyBLAHMr8SsPhuM9cJK56xJpTovBLQn25aJvpTl3JowQ==
kazuki@workstation
```

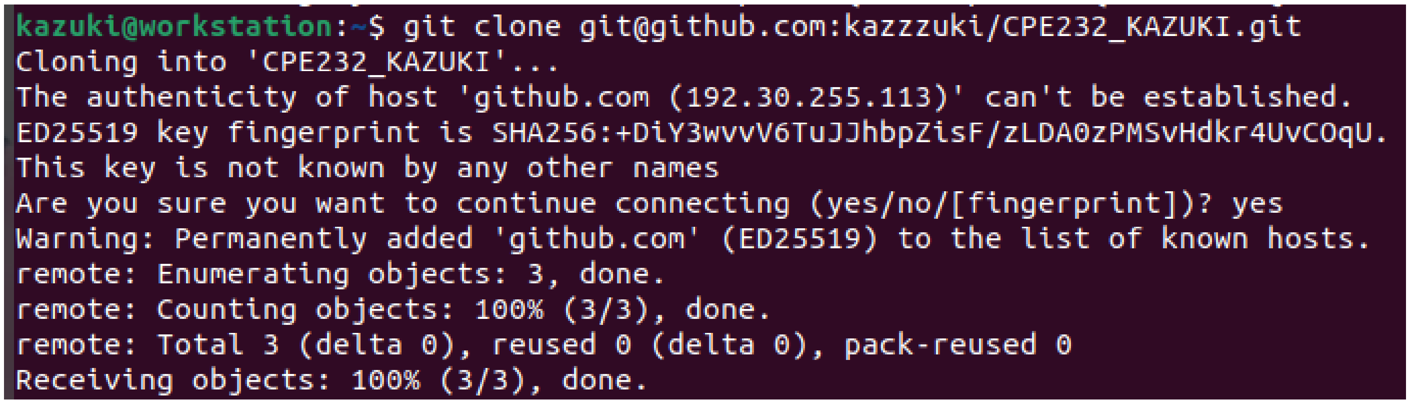
d. Clone the repository that you created. In doing this, you need to get the link from GitHub. Browse to your repository as shown below. Click on the Code drop down menu. Select SSH and copy the link.



jvtaylar-cpe / CPE302_yourname Public
Code Issues Pull requests Actions Projects Wiki Security Insights
main
Go to file Add file Code
Clone
HTTPS SSH GitHub CLI
git@github.com:jvtaylar-cpe/CPE302_you
Use a password-protected SSH key.
Download ZIP
jvtaylar-cpe Initial commit
README.md Initial commit
README.md
CPE302_yourname
About
No description, website, or topics provided.
Readme
Releases
No releases published
Create a new release
Packages

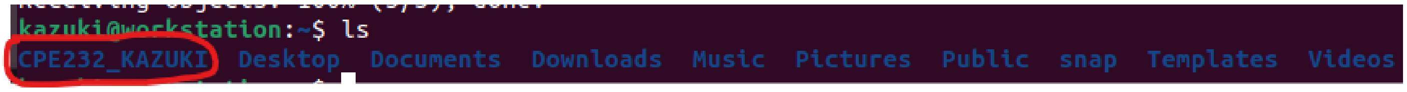


e. Issue the command git clone followed by the copied link. For example, *git clone* *git@github.com:jvtaylar-cpe/CPE232_yourname.git*. When prompted to continue connecting, type yes and press enter.



```
kazuki@workstation:~$ git clone git@github.com:kazzzuki/CPE232_KAZUKI.git
Cloning into 'CPE232_KAZUKI'...
The authenticity of host 'github.com (192.30.255.113)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvvV6TuJJhbpZisF/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: 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.
```

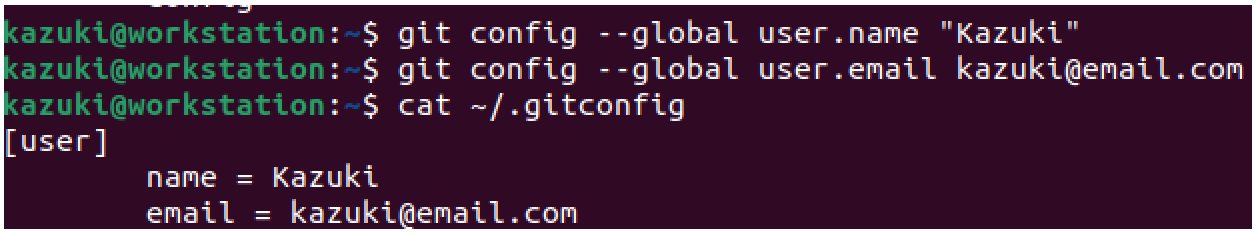
f. To verify that you have cloned the GitHub repository, issue the command *ls*. Observe that you have the CPE232_yourname in the list of your directories. Use CD command to go to that directory and LS command to see the file README.md.



```
kazuki@workstation:~$ ls
CPE232_KAZUKI  Desktop  Documents  Downloads  Music  Pictures  Public  snap  Templates  Videos
```

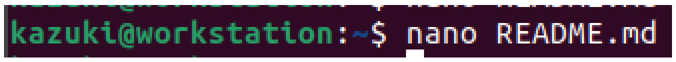
g. Use the following commands to personalize your git.

- *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*

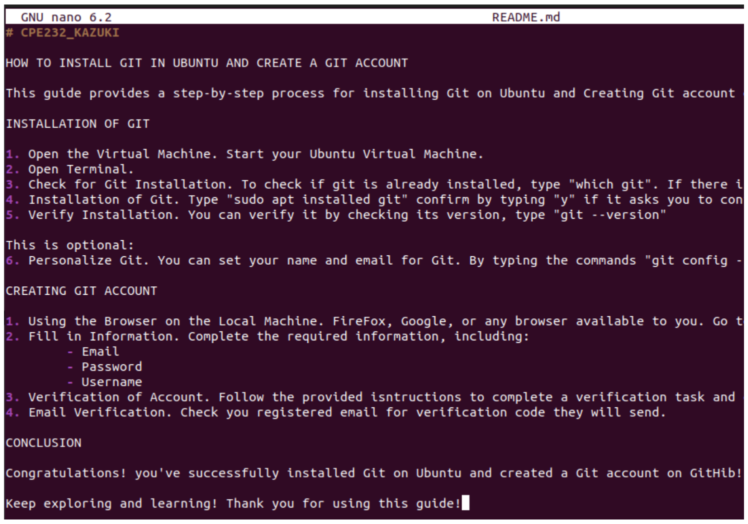


```
kazuki@workstation:~$ git config --global user.name "Kazuki"
kazuki@workstation:~$ git config --global user.email kazuki@email.com
kazuki@workstation:~$ cat ~/.gitconfig
[user]
        name = Kazuki
        email = kazuki@email.com
```

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.

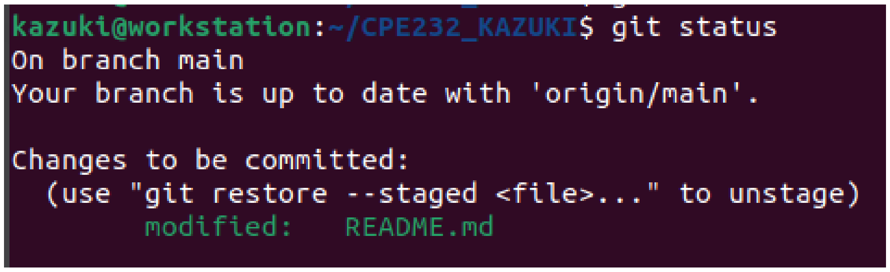


```
kazuki@workstation:~$ nano README.md
```



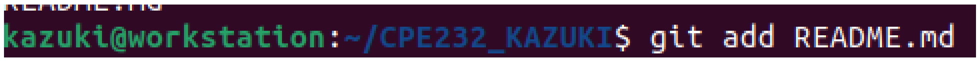
```
  GNU nano 6.2                                        README.md
# CPE232_KAZUKI

HOW TO INSTALL GIT IN UBUNTU AND CREATE A GIT ACCOUNT

This guide provides a step-by-step process for installing Git on Ubuntu and Creating Git account

INSTALLATION OF GIT

1. Open the Virtual Machine. Start your Ubuntu Virtual Machine.
2. Open Terminal.
3. Check for Git Installation. To check if git is already installed, type "which git". If there i
4. Installation of Git. Type "sudo apt installed git" confirm by typing "y" if it asks you to con
5. Verify Installation. You can verify it by checking its version, type "git --version"

This is optional:
6. Personalize Git. You can set your name and email for Git. By typing the commands "git config -

CREATING GIT ACCOUNT

1. Using the Browser on the Local Machine. FireFox, Google, or any browser available to you. Go t
2. Fill in Information. Complete the required information, including:
        - Email
        - Password
        - Username
3. Verification of Account. Follow the provided isntructions to complete a verification task and
4. Email Verification. Check you registered email for verification code they will send.

CONCLUSION

Congratulations! you've successfully installed Git on Ubuntu and created a Git account on GitHib!

Keep exploring and learning! Thank you for using this guide!
```

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?

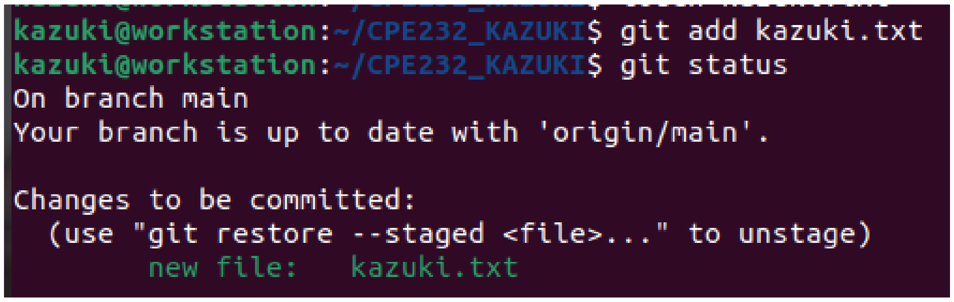


```
kazuki@workstation:~/CPE232_KAZUKI$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   README.md
```

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



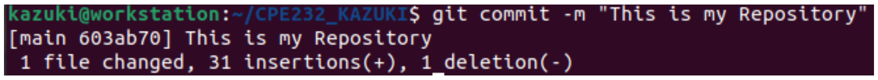
```
kazuki@workstation:~/CPE232_KAZUKI$ git add README.md
```



```
kazuki@workstation:~/CPE232_KAZUKI$ git add kazuki.txt
kazuki@workstation:~/CPE232_KAZUKI$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   kazuki.txt
```

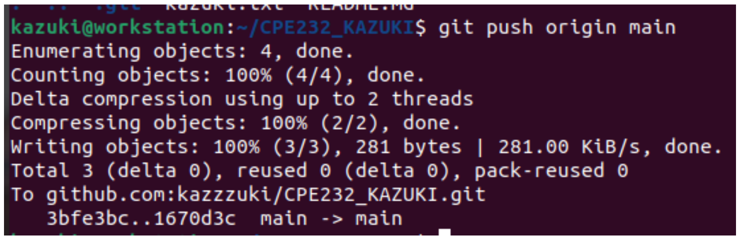
I created a different file just to show this.

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.



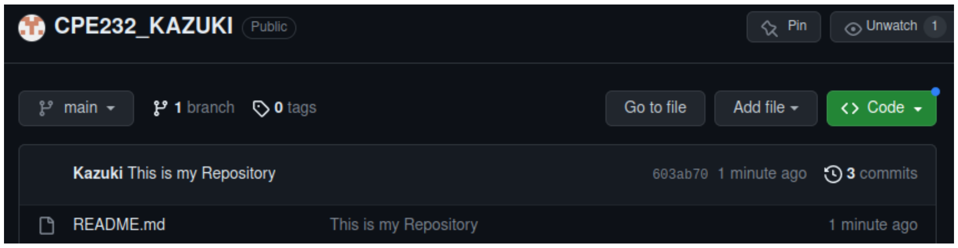
```
kazuki@workstation:~/CPE232_KAZUKI$ git commit -m "This is my Repository"
[main 603ab70] This is my Repository
 1 file changed, 31 insertions(+), 1 deletion(-)
```

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.*



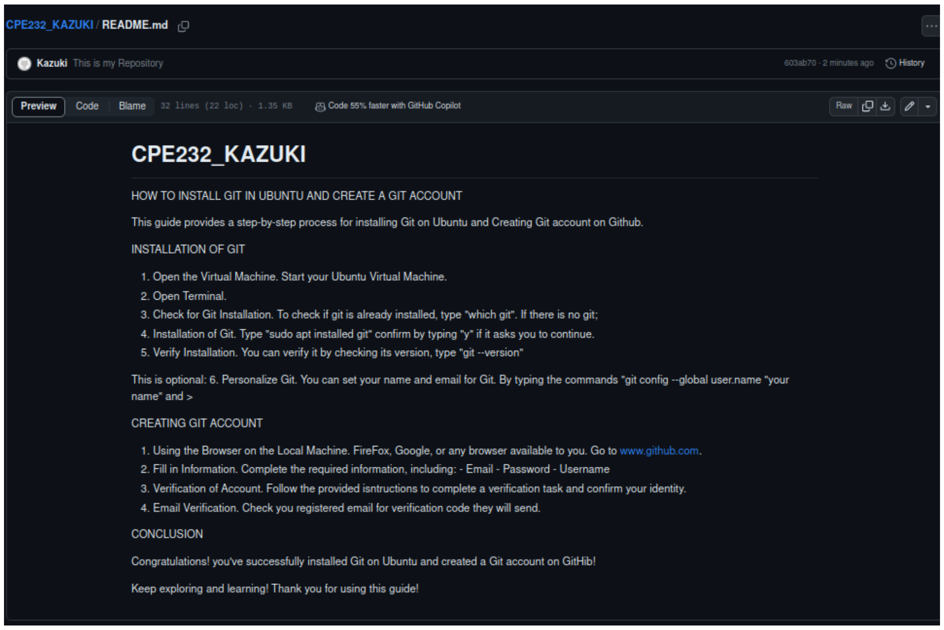
```
kazuki@workstation:~/CPE232_KAZUKI$ git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 281 bytes | 281.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:kazzzuki/CPE232_KAZUKI.git
   3bfe3bc..1670d3c  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 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.



Kazuki This is my Repository 603ab70 1 minute ago 3 commits

README.md This is my Repository 1 minute ago



CPE232_KAZUKI / README.md
History
Preview
Code
Blame
Code 55% faster with GitHub Copilot
Raw

Kazuki This is my Repository 603ab70 · 2 minutes ago

32 lines (22 loc) · 1.35 KB

CPE232_KAZUKI

HOW TO INSTALL GIT IN UBUNTU AND CREATE A GIT ACCOUNT

This guide provides a step-by-step process for installing Git on Ubuntu and Creating Git account on Github.

INSTALLATION OF GIT

1. Open the Virtual Machine. Start your Ubuntu Virtual Machine.
2. Open Terminal.
3. Check for Git Installation. To check if git is already installed, type "which git". If there is no git;
4. Installation of Git. Type "sudo apt installed git" confirm by typing "y" if it asks you to continue.
5. Verify Installation. You can verify it by checking its version, type "git --version"

This is optional: 6. Personalize Git. You can set your name and email for Git. By typing the commands "git config --global user.name "your name" and >

CREATING GIT ACCOUNT

1. Using the Browser on the Local Machine. FireFox, Google, or any browser available to you. Go to www.github.com.
2. Fill in Information. Complete the required information, including: - Email - Password - Username
3. Verification of Account. Follow the provided isntructions to complete a verification task and confirm your identity.
4. Email Verification. Check you registered email for verification code they will send.

CONCLUSION

Congratulations! you've successfully installed Git on Ubuntu and created a Git account on GitHib!

Keep exploring and learning! Thank you for using this guide!

Reflections:
Answer the following:

3. What sort of things have we so far done to the remote servers using ansible commands?
   - Generated SSH key pair using the "ssh-keygen" command.
   - Copied the public key to remote servers (servers 1 and 2) using the "ssh-copy-id" command.
   - Verified and Installed Git using the "which git" and "sudo apt install git" commands.
   - Set up and created a new Git repository and created a new SSH key on GitHub.
   - Cloned the created Repository using the "git clone" command.
   - Personalized Git using the "git config --global user.name" and "git config --global user.email" commands.
   - Edited the README.md file using the "nano" command.
   - Checked git status using the "git status" command.
   - Added the "README.md" file in the repository using the "git add" command.
   - Committed a message/changes to the repository using the "git commit" command.
   - Pushed the message/changes to the repository on GitHub using the "git push" command.
   - Verified the "REAM.md" file on GitHub.

4. How important is the inventory file?
   - Inventory files are important for security, planning, tracking, and troubleshooting in many things. Especially in SSH key management and Setting up Git. For example, in task 1, generating ssh keys and copying them involves planning and tracking key pairs in the inventory file, enabling secure access. In setting up Git, tracking of commits/changes will help us to manage the project history efficiency. Overall, the inventory file ensures secured access and organized resources.

Conclusions/Learnings:

In conclusion, I successfully configured both remote and local machines to connect via SSH using a KEY instead of using a password and created a public and private key. With this, we can connect via SSH easily unlike in the past activity that asks if we want to continue and for a password but with this, in short, this is very convenient for us, but it is also kind of unsecured since if anyone knows your private key they can easily access your system and do something bad to it. Verify connectivity, just like in the last activity I successfully connected the local machine to remote servers (servers 1 and 2), which shows the effectiveness of the SSH key-based authentication. Additionally, I successfully set up a Git repository using local and remote repositories, actually it is my first time using and creating an account on GitHub, but I was able to do all the tasks by following all of this since the steps are all clear. I learned so many things here, I was amazed when I was editing the README.md file and when I save it and run some commands it is automatically updated on the GitHub platform. Overall, this activity expanded my knowledge and understanding of secure connections.