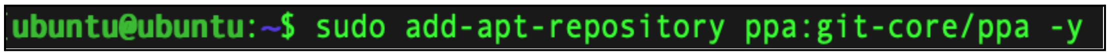
Objective : In this lab, you will configure and initialize git.

Login and update your linux workstation

1. **SSH** into your Ubuntu Workstation, as provided to you by your instructor (refer to "Azure_Lab-01_Access A Remote Workstation" if you don't recall how to log into the remote node)

2. To update to the latest version of **GIT** run the following command.

   $ sudo add-apt-repository ppa:git-core/ppa -y



```
ubuntu@ubuntu:~$ sudo add-apt-repository ppa:git-core/ppa -y
```

   NOTE: PPA has the most up-to-date packages for GIT.

3. **Update** and **upgrade** the OS workstation provided to you by your instructor, and install the **'tree'** and **'unzip'** command-line utility:

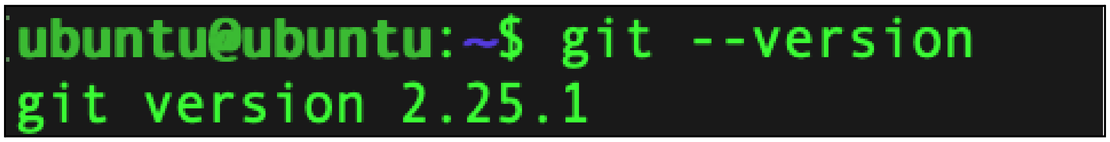
   $ sudo apt-get update -y && sudo apt-get upgrade -y

   $ sudo apt-get install -y unzip tree

Verify git is installed on your remote workstation

4. To verify the pre-installed git **version**:

   $ git --version



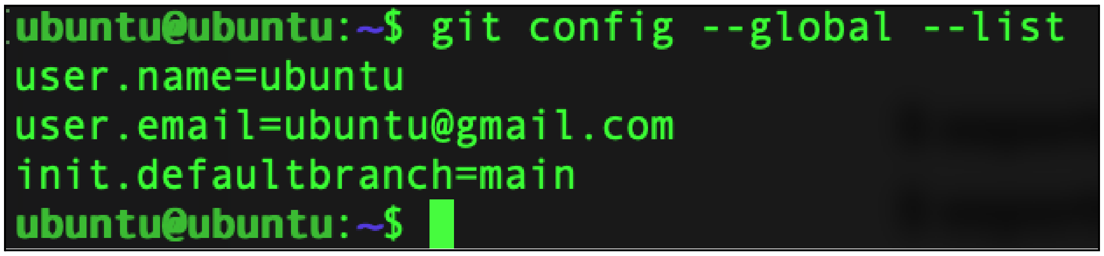
```
ubuntu@ubuntu:~$ git --version
git version 2.25.1
```

   Note: git version 2.X or higher is OK

Configure git on your remote workstation

5. Configure your username and email address for git on your remote workstation, then check the configuration:

```
$ git config --global init.defaultBranch main

$ git config --global user.name "Your-Firstname Your-Lastname"

$ git config --global user.email "Your-email@address.com"

$ git config --global --list
```



```
ubuntu@ubuntu:~$ git config --global --list
user.name=ubuntu
user.email=ubuntu@gmail.com
init.defaultbranch=main
ubuntu@ubuntu:~$
```

Set the editor for your workstation. Choose either nano, vi, vim or emacs

6. To set the editor environment variable, execute only one of the four commands shown below:

```
$ export EDITOR=nano
$ export EDITOR=vi
$ export EDITOR=vim
$ export EDITOR=emacs
```

7. To change your configuration credentials, execute the following command**:**

```
$ git config --global --edit
```

8. Modify your **user_name**, save and quit

Create and initialize a local git repo

9. Change to your home directory:

```
$ cd ~
```

10. In this step, create a new directory with the name **my-repo** which we'll initialize as git-enabled:

    $ mkdir my-repo

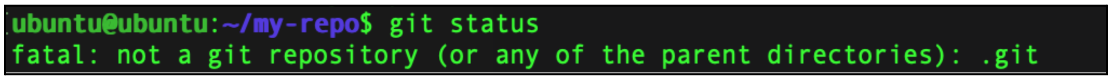
11. ~~Change~~ Move into the new directory:

    $ cd my-repo

12. ~~Check~~ Verify the current git status of the new directory:

    $ git status

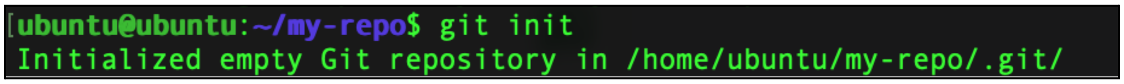
Note: This should fail because you have not yet initialized the directory



```
ubuntu@ubuntu:~/my-repo$ git status
fatal: not a git repository (or any of the parent directories): .git
```

13. **Initialize** the directory as a git-enabled

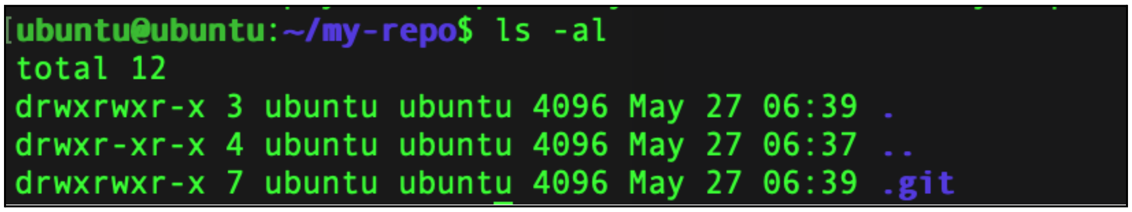
    $ git init



```
ubuntu@ubuntu:~/my-repo$ git init
Initialized empty Git repository in /home/ubuntu/my-repo/.git/
```

14. Verify that the directory has been initialized:

    $ ls -al



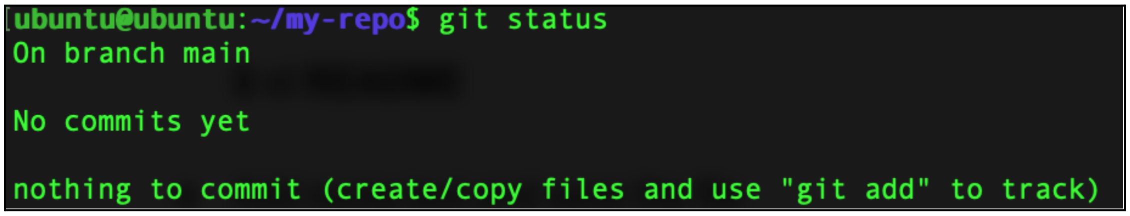
```
ubuntu@ubuntu:~/my-repo$ ls -al
total 12
drwxrwxr-x 3 ubuntu ubuntu 4096 May 27 06:39 .
drwxr-xr-x 4 ubuntu ubuntu 4096 May 27 06:37 ..
drwxrwxr-x 7 ubuntu ubuntu 4096 May 27 06:39 .git
```

15. Verify the existence of the .git directory:
    - cd into and look at the .git data files

- Return to the my-repo directory

  $ cd ..

16. Check current git status once again. This time you will notice:
    - That you are on the main branch
    - There are no commits yet
    - The names of any untracked files



```
ubuntu@ubuntu:~/my-repo$ git status
On branch main

No commits yet

nothing to commit (create/copy files and use "git add" to track)
```

First git commands

17. Move into the **my-repo** directory:

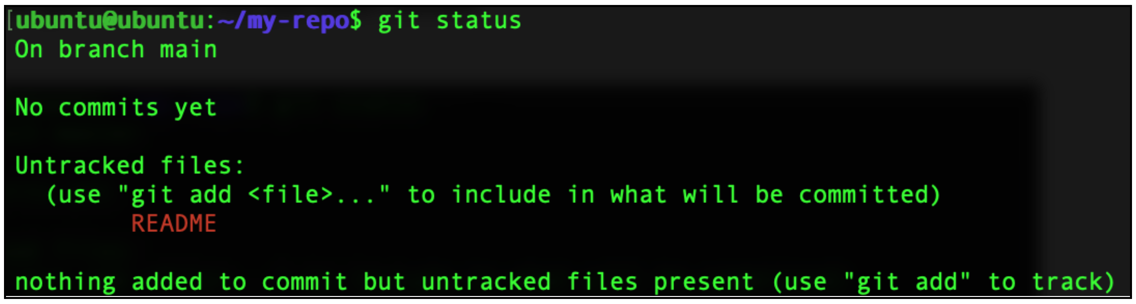
18. Create a README file

    $ vi README

    - Add some text and save the file

19. Run git status to see how git handles this new file

    Note: Under 'Untracked files' you should see "README", showing that this file has been changed but has not yet been added

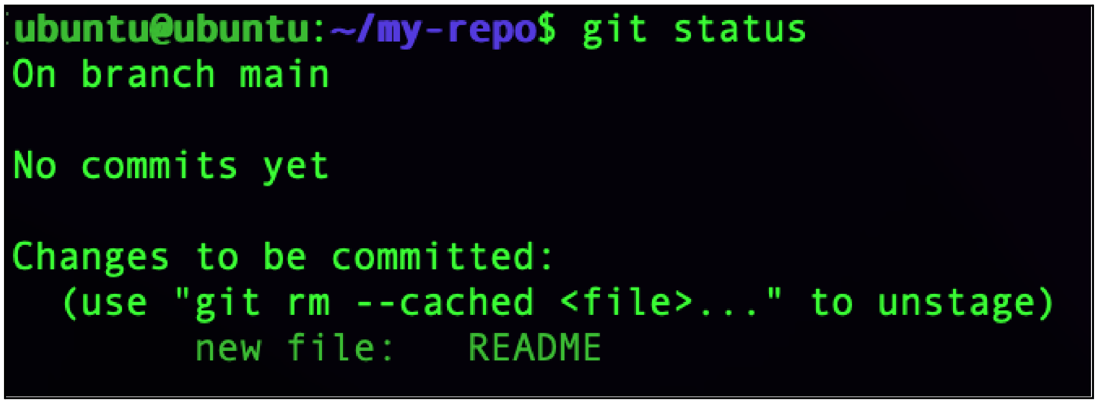


```
ubuntu@ubuntu:~/my-repo$ git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        README

nothing added to commit but untracked files present (use "git add" to track)
```

20. Add the README file to the git staging area:

    $ git add README

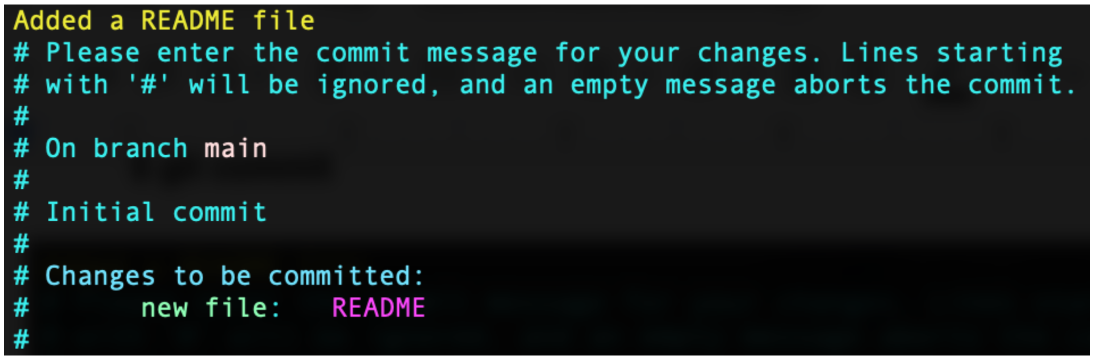
21. Verify whether the file is added to the **staging area** by executing the **git status** command:



```
ubuntu@ubuntu:~/my-repo$ git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   README
```

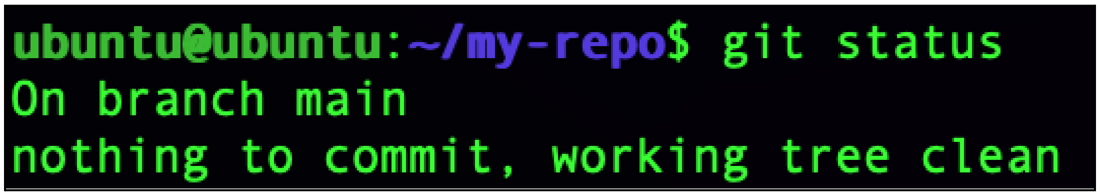
22. Commit the file to the local git repository. Add the text below in yellow to the commit log:

    $ git commit



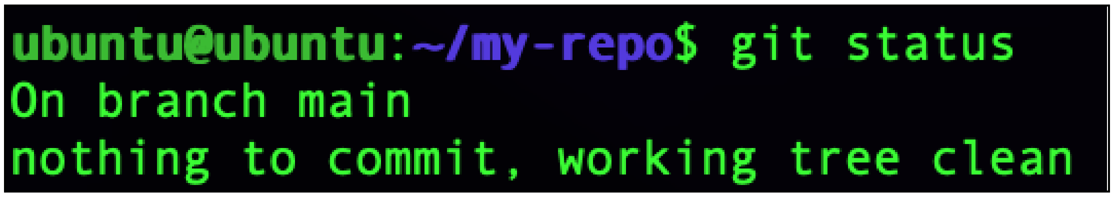
```
Added a README file
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch main
#
# Initial commit
#
# Changes to be committed:
#       new file:   README
#
```

23. Check the current **git status** after committing the file



```
ubuntu@ubuntu:~/my-repo$ git status
On branch main
nothing to commit, working tree clean
```

24. **Re-run** the git commit command to see what happens if you try to commit again without first making changes and adding the files to the staging area

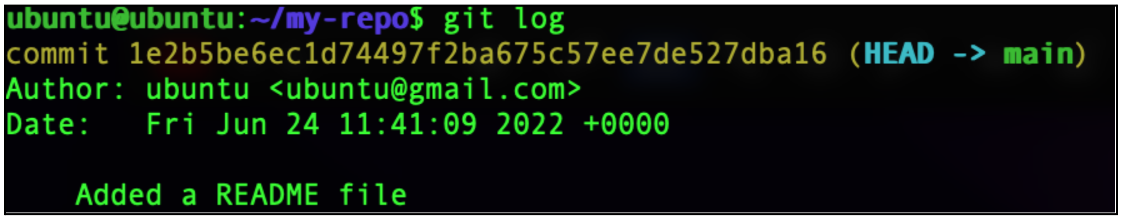


```
ubuntu@ubuntu:~/my-repo$ git status
On branch main
nothing to commit, working tree clean
```

Note: There is nothing to commit, because everything has already been committed

25. To view all the commit history, execute:

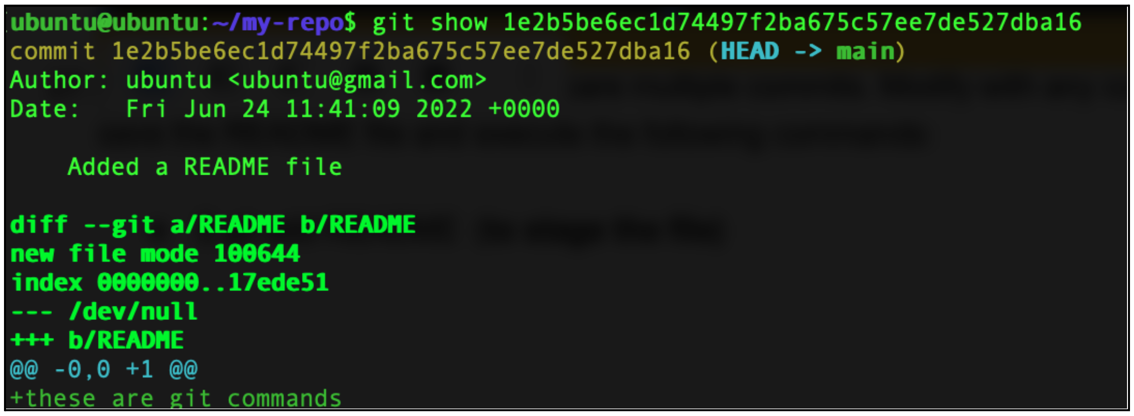
$ git log



```
ubuntu@ubuntu:~/my-repo$ git log
commit 1e2b5be6ec1d74497f2ba675c57ee7de527dba16 (HEAD -> main)
Author: ubuntu <ubuntu@gmail.com>
Date:   Fri Jun 24 11:41:09 2022 +0000

    Added a README file
```

26. To verify the details of a particular commit:

$ git show <your_commit_id>

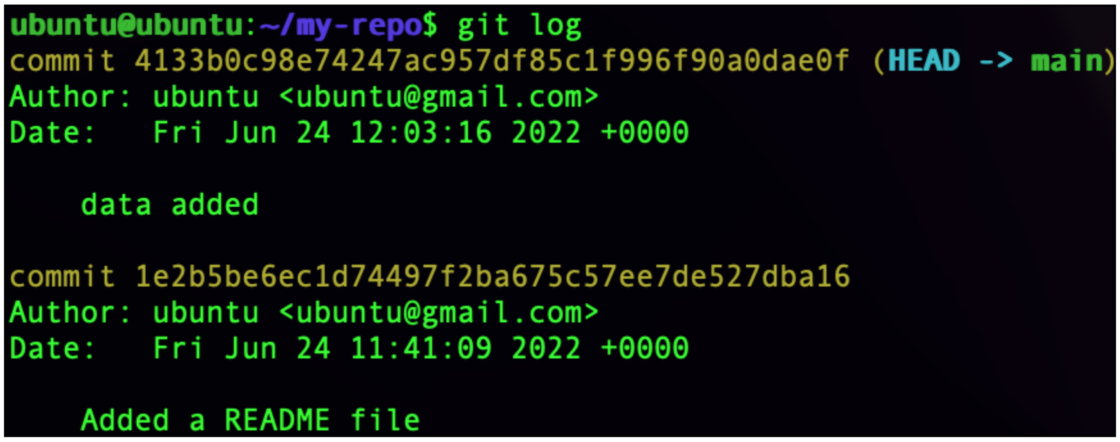


```
ubuntu@ubuntu:~/my-repo$ git show 1e2b5be6ec1d74497f2ba675c57ee7de527dba16
commit 1e2b5be6ec1d74497f2ba675c57ee7de527dba16 (HEAD -> main)
Author: ubuntu <ubuntu@gmail.com>
Date:   Fri Jun 24 11:41:09 2022 +0000

    Added a README file

diff --git a/README b/README
new file mode 100644
index 0000000..17ede51
--- /dev/null
+++ b/README
@@ -0,0 +1 @@
+these are git commands
```

27. Update your **README** file to compare multiple commits. Modify with any content and save the README file and execute the following commands:

- $ git add README (**to stage the file)**

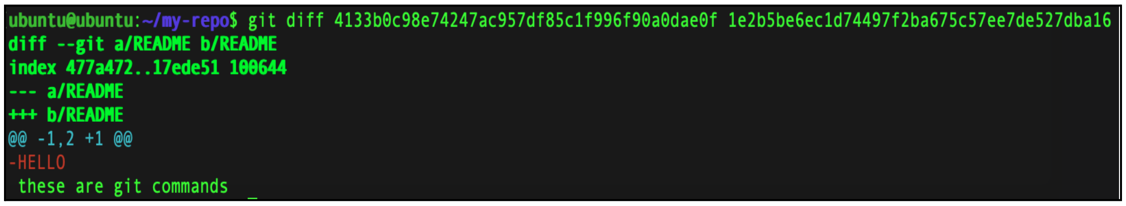
- $ git commit, add a **commit message**
- $ git log **(to see the commit history, note there are two different SHA's)**



```
ubuntu@ubuntu:~/my-repo$ git log
commit 4133b0c98e74247ac957df85c1f996f90a0dae0f (HEAD -> main)
Author: ubuntu <ubuntu@gmail.com>
Date:   Fri Jun 24 12:03:16 2022 +0000

    data added

commit 1e2b5be6ec1d74497f2ba675c57ee7de527dba16
Author: ubuntu <ubuntu@gmail.com>
Date:   Fri Jun 24 11:41:09 2022 +0000

    Added a README file
```

28. To compare differences between two commits:

$ git diff <1st sha> <2nd sha>



```
ubuntu@ubuntu:~/my-repo$ git diff 4133b0c98e74247ac957df85c1f996f90a0dae0f 1e2b5be6ec1d74497f2ba675c57ee7de527dba16
diff --git a/README b/README
index 477a472..17ede51 100644
--- a/README
+++ b/README
@@ -1,2 +1 @@
-HELLO
 these are git commands  _
```

Note: This shows you the **changes** that you made between the **two commits**

29. Practice the entire commit process once again by following the steps below:

- Modify and save the README file again
- $ git status (to see the file that has been modified but is untracked)
- $ git add README (to stage the file)
- $ git status (to see there is something to commit)
- $ git commit (to commit it)

- $ git log (to see the commit summary)

- $ git show [SHA] (use the top SHA in your list, which is the most recent commit)

  This will show the commit details

  Note: [SHA] should be replaced with the 40-digit unique identifier of your last commit

- $ git diff [SHA-1] [SHA-2] (to see the difference between multiple commits)

Notify your instructor that you are done with the Lab

END OF LAB