Warm-up – GIT / SSH / MAKE

Lab

Task for this part

- This lab has no assessable items.
- In this lab you will learn how to set up use GIT, SSH and Makefile.

- Task 1
  - GIT
- Task 2
  - SSH
- Task 3
  - Makefile

Task 1 – GIT - a distributed version control application

The main objective of task 1 is to understand and get some practice with some basic git operations. If you are already familiar with Git, skim through and make sure your GIT is set up correctly.

- Use the terminal to run the commands described here.
- Use `git help [command]` to get instructions of commands you do not know

Let's get started!

- Configure your name and e-mail for your GIT. Use the following commands:

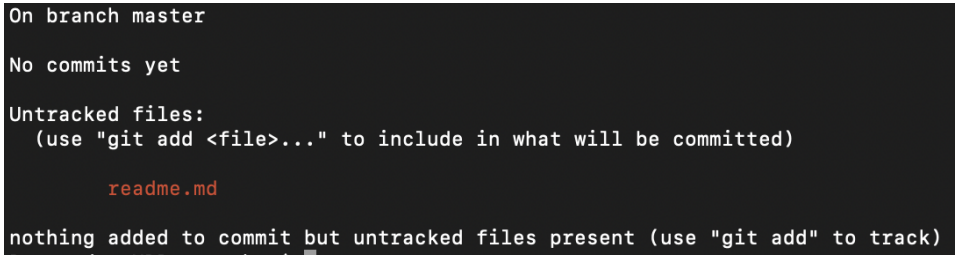
```
git config --global user.name "YOUR NAME"
git config --global user.email your_uid@anu.edu.au
```

Have you done this before? Use the commands `git config --global user.name` and `git config --global user.email` to check the current name and email, respectively.

Task 1 - GIT

Setting up a new GIT REPO:

1) Create a new directory under your home directory. For example, "COMP2100-6442".
2) Now, go to the directory you just created (`cd COMP2100-6442`) and execute `git init.` Terminal will show you the following message **"Initialized empty Git repository in [folder path]".**
3) If you list all files (`ls -al`) under the directory you just created, a directory named **.git** must be there.
4) Create a new file named as "readme.md" containing a message such as "This is my new Repo".
5) Execute the command `git status.` The file is in red as an "Untracked file".



Task 1 - GIT

6) As the figure shows, use "`git add readme.md`" to track the changes in this file. The result of this command is shown in the figure below. Now, the file is ready to be committed.

```
On branch master

No commits yet

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

        new file:   readme.md
```

7) To commit, execute the command `git commit -m "added readme".` **`-m`** is used to add a message to your commit. After you execute this command, you may get a message similar to the following one:

```
[master (root-commit) 0fcd6ed] added readme
 1 file changed, 1 insertion(+)
 create mode 100644 readme.md
```

Task 1 - GIT

8) Now, let's create a Repo on GitLab. Access https://gitlab.cecs.anu.edu.au/ and log in with your uni-id and password
9) Click on the green button "New Project". Fill in the the Project name with "`COMP2100-6442`", choose the visibility level and click on "Create project". It should bring you to a new page showing that your project was successfully created.
10) Click on the button "Clone" and copy the HTTPS URL. It should be something similar to: https://gitlab.cecs.anu.edu.au/uXXXXXXX/comp2100-and-comp6442.git
11) Go back to your terminal and execute the command `git remote add origin https://gitlab.cecs.anu.edu.au/uXXXXXXX/comp2100-and-comp6442.git`. (do not forget to replace the uXXXXXX by your uid).
12) Execute the command `git remote -v`

```
origin  https://gitlab.cecs.anu.edu.au/u xxxxx /comp2100-and-comp6442.git (fetch)
origin  https://gitlab.cecs.anu.edu.au/u xxxxx /comp2100-and-comp6442.git (push)
```

These two URLs marks the remote URL when git push and fetch the remote repo.

Task 1 - GIT

13) Now, push to the remote repo with the command **`git push origin master`**. If you refresh the GitLab page, your file should appear in the page.
14) Let's create a new repository and create a conflict to be solved! We will also learn a few new commands.
15) First, create a new directory (outside the `COMP2100-6442`). For example, "`COMP_TEST`".
16) Inside the new directory, execute the command:

```
git clone https://gitlab.cecs.anu.edu.au/uXXXXXXX/comp2100-and-comp6442.git
```

```
Cloning into 'comp2100-and-comp6442'...
remote: Enumerating objects: 24, done.
remote: Counting objects: 100% (24/24), done.
remote: Compressing objects: 100% (11/11), done.
remote: Total 24 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (24/24), done.
```

17) In the new directory, you should be able to find the readme.md file, the clone of the GitLab.
18) To avoid confusion, rename this folder to another name. For example:

```
mv comp2100-and-comp6442 comp2100-and-comp6442-NEW
```

Task 1 - GIT

Time to create and resolve a conflict!

19) Go to the NEW folder `cd comp2100-and-comp6442-NEW`. Make changes in your **readme.md** file and create a new commit (`git add readme.md` and then `git commit –m “readme.md updated”`.
20) Now, push the commit to the remote repo: `git push origin master`. Check GitLab, the changes should be reflected there.
21) Go back to the original directory, the first we created, and try to push: `git push origin master`

```
 ! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'https://gitlab.cecs.anu.edu.au/u xxxx /comp2100-and-comp6442.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

This means the remote repo contains changes that your local repo doesn’t contain. You have to pull those changes down to your local repo and make a merge.

Task 1 - GIT

22) Pull down and merge the Repo. Execute the command: **`git pull origin master`**

23) The following message should appear. Git will try to auto merge the file, but sometimes it fails and shows as a conflict:

```
From https://gitlab.cecs.anu.edu.au/u xxxxx /comp2100-and-comp6442
 * branch            master     -> FETCH_HEAD
Auto-merging readme.md
CONFLICT (content): Merge conflict in readme.md
Automatic merge failed; fix conflicts and then commit the result.
```

Task 1 - GIT

24) Let's fix the conflict now! Open the file **readme.md**, you will see a few lines surrounded by '«««<', '=======' and '»»»»>', which the HEAD shows your local repo version of the part of the file and the one with hex hash shows the remote version of the file.

25) To solve the issue, edit the file, fix the code and remove the lines added by Git. The symbols («<,==,»>) are only to mark where is different.

26) After fixing the file, execute:

```
git add readme.md
git commit -m "conflict solved"
git push origin master
```

Check the online Git Repo, your changes should be there. Finally, go to the directory "`comp2100-and-comp6442-NEW`" and try to pull down the changes. It should be all fine: **`git pull origin master.`** Use the command **`git log`**, to check it.

Task 2 – SSH Setup for GIT

The main objective of this task is to understand and get some practice with SSH setup and use them for ssh, scp and GitLab.
Read the following document if you are off campus (new policy): https://cs.anu.edu.au/docs/student-computing-environment/linuxlabs/remoteaccess/

1) Create your own public private key pair. To generate a key pair you have to execute **ssh-keygen**

2) You may need to enter the location (or current directory) for saving the key and an optional passphrase for protecting the key or just using the default settings.

3) Your key is now generated and is located in your current directory or the location you provided.

Note that the key and image are illustrative only. You must not share your key/image with anyone.

```
user$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/User/xxx/.ssh/id_rsa):
my_anu_cecs_key
Enter passphrase (empty for no passphrase):  *********
Enter same passphrase again:  *********
Your identification has been saved in anu_cecs_key.
Your public key has been saved in anu_cecs_key.pub.
The key fingerprint is:
SHA256:sv4***E4WxQ4**dDJiAIJ09s0***********
The key's randomart image is:
+---[RSA 2048]----+
|    ==+   o.B.1.  |
|     ..+    B.+.o.  |
|   .     . $o.  |
|         .= =.  |
|   .  . S +  - 2  |
|  1 o .oE1  . . |
| + =.*o..  B      |
|  BooB*+.B . o  |
|  ..o===+.       |
+----[SHA256]-----+
```

Task 2 – SSH Setup for GIT

4) If you cannot find your keys, search for the identification you provided: for example: **anu_cecs_key.pub** (public key) or **anu_cecs_key** (private key).

5) Use your ssh key to login to partch with your uni-id password, open the 'authorized_keys' file under '.ssh' directory and paste the content of your public key into a new line of 'authorized_keys' (You need to make 'authorized_keys' file if you can't find one). Save and exit the text editor.

6) Log out and then log in again; you must use your SSH keys. If you set a passphrase for the private key, you must input the passphrase for the private key, but not the uni-id password. If no passphrase has been set, it must connect directly to the partch.

7) Try to copy some file using `scp` from your local hard drive to partch, it must copy without asking for your uni-id password.

8) Finally, use the SSH key settings to clone, push and pull on CECS Gitlab. Go to CECS GitLab, log in and click on "your profile -> Settings". On the left hand side list, there should be one item named as 'SSH Keys'. You have to paste your public key into the input box of the 'Key' area, and click on the 'Add Key' button. Now, when you clone your repo using the 'SSH' hyperlink (not 'HTTPS'), it should no longer ask for your uni-id password.

Task 3 – Makefile

The main objective of this stage is to understand and get some practice with **Makefile**. “Make is a Unix tool to simplify building program executables from many modules. make reads in rules (specified as a list of target entries) from a user created Makefile. **make will only re-build things that need to be re-built** (object or executables that depend on files that have been modified since the last time the objects or executables were built).”[2]

1) Create a file named as ‘Makefile’ under your ‘`COMP2100-6442`’ directory.

2) Use any text editor (e.g., vi, nano or notepad) to modify the Makefile.

3) You need to implement three targets for the Makefile: default' to compile your '`HelloWorld.java`' into '`HelloWorld.class`',' run ‘to execute the '`HelloWorld.class`' with Java and 'clean' to remove the ‘`HelloWorld.class`’ file. The java file are available on Wattle/Repo of this course.

4) Try your Makefile with the `make` command.

[1]Make Documentation: https://www.gnu.org/software/make/manual/make.html
[2]Make Basics: https://www.cs.swarthmore.edu/~newhall/unixhelp/howto_makefiles.html

A very simple example of a makefile is available on Wattle.