Exercise-1-Basic-uses-of-Git

April 3, 2024

1 High-level goal

The high-level goal of this exercise is to learn about basic git commands: `git clone`, `git status`, `git add`, `git diff`, `git commit`, `git log`, `git pull` and, `git push`.

2 Submission

This in-class exercise is an individual submission. The deadline for submitting the solution is 11/01.

3 What to do

3.1 Set up

1. Make sure that you have git installed. Ensure that you are using newer version of git (version 2.34.0). Update your git if needed.

git clone

2. Run the command `man git clone` and familiarize yourself with the output.

[ ]:
```
%%bash

man git clone
```

3. Clone the following git repository: https://github.com/CS-563/basic-stats

[ ]:
```
%%bash

rm -rf basic-stats
if [ ! -d "basic-stats" ]; then
  git clone https://github.com/CS-563/basic-stats
fi
```

4. Create a (second) local fork by locally cloning again, this time from the first local clone.

[ ]:
```
%%bash

rm -rf basic-stats-fork

if [ ! -d "basic-stats-fork" ]; then
```

```
    git clone basic-stats basic-stats-fork
fi
```

git status

5. Run the command `man git status` and familiarize yourself with the output.

[ ]:
```
%%bash

man git status
```

6. Run the command `git status` from inside the `basic-stats-fork` directory and analyse the output.

[ ]:
```
%%bash

if [ -d "basic-stats-fork" ]; then
    cd basic-stats-fork
    git status
fi
```

7. Update the basic-stats-fork/README.md file
8. Re-run command `git status` and analyze its output. What is the difference you observe?

[ ]:
```
%%bash

if [ -d "basic-stats-fork" ]; then
    cd basic-stats-fork
    echo "making some change" >> README.md
    git status
fi
```

9. Create a new file `testfile.md` inside the `basic-stats-fork` directory and add some content to it.
10. Run the command `git status` and analyse the output. What is the difference you observe?

[ ]:
```
%%bash

if [ -d "basic-stats-fork" ]; then
    cd basic-stats-fork
    echo "this is a new file" > testfile.md
    git status
fi
```

git add

11. Run the command `man git add` and familiarize yourself with the output.

[ ]:
```
%%bash

man git add
```

12. Add your new file `testfile.md` using the `git add testfile.md` command.
13. Run the command `git status` and analyse the output. What is the difference you observe?

[ ]:
```
%%bash

if [ -d "basic-stats-fork" ]; then
    cd basic-stats-fork
    git add testfile.md
    git status
fi
```

git diff

14. Run the command `man git diff` and familiarize yourself with the output.

[ ]:
```
%%bash

man git diff
```

15. Use `git diff` to check the changes you made to the README.md file. What happens when you run `git diff` for `testfile.md`? why?

[ ]:
```
%%bash

if [ -d "basic-stats-fork" ]; then
    cd basic-stats-fork
    git diff README.md
fi
```

git commit

Now that we have verified that our changes are correct, we want to commit these changes. This is done using `git commit` command. 16. Run the command `man git commit` and familiarize yourself with the output.

[ ]:
```
%%bash

man git commit
```

17. Use `git` commit to commit your changes. Make sure to create separate commits for separate changes (remember best practices!?)

[ ]:
```
%%bash

if [ -d "basic-stats-fork" ]; then
    cd basic-stats-fork
    git commit README.md -m "made changes to README and adding descriptive␣
 ↪commit message"
    git commit testfile.md -m "added new file and a descriptive message␣
 ↪describing it"
```

```
fi
```

git pull

Now before we push our committed changes from our *working copy* to the *central repository*, we must ensure that our *working copy* is updated with the latest *central repository*. **This is importanat to avoid merge conflicts.**. We do this using `git pull` command.

18. Run the command `man git pull` and familiarize yourself with the output.

[ ]:
```
%%bash

man git pull
```

19. Run the command `git pull` inside your working copy to make it up to date. (You may have to manually resolve marge conflicts if you messed up in any of the above steps.)

[ ]:
```
%%bash

if [ -d "basic-stats-fork" ]; then
    cd basic-stats-fork
    git pull
fi
```

git log

This is a very useful command to view the committed changes. After we did `git pull`, we can use `git log` to see the changes we **committed** to the repository as well as the changes **pushed** by others.

20. Run the command `man git log` and familiarize yourself with the output.

[ ]:
```
%%bash

man git log
```

21. Run the command `git log` inside your working copy and analyze the output. What do you observe?

[ ]:
```
%%bash

if [ -d "basic-stats-fork" ]; then
    cd basic-stats-fork
    git log
fi
```

git push

Finally, when we are satisfies that our changes integrate well with the changes already pushed by others, its time to *push* our changes to the *central repository* so that our changes are available to others. We do this using `git push` command.

22. Run the command `man git push` and familiarize yourself with the output.

[ ]:
```
%%bash

man git push
```

23. Push your changes by running `git push` inside your working copy and analyze the output. What do you observe?

[ ]:
```
%%bash

if [ -d "basic-stats-fork" ]; then
    cd basic-stats-fork
    git push
fi
```

3.2 Deliverables

After successfully completing the exercise, please upload a single archive with the following content:

1. Your .git folder inside your working copy. Note that this should not be the files in the basic-stats working copy, but instead the repository (which is the .git directory in basic-stats.) For example, on a Linux-based machine (e.g., MacOS), you can use the terminal from the basic-stats directory and run the command `tar -vczf basic-stats.tar.gz .git`.
2. A plain-text file with your answers to the questions in steps 8, 10, 13, 15, 21, and 23 above.