COMP20008 2021 S2 - An e-portfolio for data science - github

Github

Git is a popular open source version control system that is widely used in industry. It includes a number of collaborative features, allowing programmers to work together on coding projects. Git allows programmers to create a *repository* for their projects. This repositoy is a collection of code, files, documentations; it includes its final version as well as all the historical versions of the files. You can add & remove files from your git repository and make changes to files in the repository and git will maintain a history of each version of your code, allowing you to easily view changes and revert to earlier versions. Multiple programmers can edit files within a git repository and git will provide tools to resolve conflicts between them.

One of the key features of git is its support for remote repositories, whereby a git repository can be stored on a remote server or in the cloud. This makes it easy for multiple users to access and collaborate in project development. The most popular provider of these remote hosting services is Github.

A typical git workflow might involve a programmer creating a git repository for a project on their local machine. They might then synchronise their local repository to a remote repository (thereby downloading the latest version of all files in the project) and make their code edits within their local repository. Once this is done, the programmer can push their changes back to the remote repository allowing other users to access them.

To use Git and Github, we have to first have Git installed in our machine and have an account with GitHub.

Have git on your machine

If you are using a Mac or Linux machine you will already have git on your machine and will be able to run git command from the terminal. Windows users will need to install git before being able to complete this lab. There are a number of different packages available and you can download one from `https://git-scm.com/download/win`. Once installed, this program will add a **Git CMD** prompt that you can use to run git commands.

Create an account in github

In order to use github you must first create a github account. You can do this by navigating to `https://github.com/` and clicking on *Sign up*.

Git commands

We then use git command to collaborate with others or to manage individual repository. We will learn a set of basic Git commands:

- `git init`
- `git remote add`
- `git pull`
- `git status`
- `git checkout`
- `git add`
- `git commit`
- `git push`

Exercises

1. Access the sample repository in github classroom

We have created a template in the github classroom for you. Follow the link to access the repository https://classroom.github.com/a/u1c76dcp.

After you accept the invitation, the githib classroom will create a private repository under https://github.com/COMP20008 for you. The repository contains two files: a README.md file and a python script called *dataframe.py*.
Question: what is the url of your remote git repository?

2. Connect to the classroom and make a local copy of the repository

Next, we want to extract a copy of the most current version of the remote repository into your local computer.

2.1 git init

In your computer, create a directory/folder where you would like the local repository to be. Run the command

```
git init
```

to initiate the git for this directory/folder: What this means is that changes you make to this folder/directory are managed by git, and git-commands will be recognised.

2.2 git remote add

Run the command with two arguments,<emote-name> and <remote-url > :

```
git remote add <remote-name> <remote-url>
```

The argument $remote-name$ is a name you call the remote repository, e.g. *origin*.
The second argument <remote-url> is your private git-repository (from Exercise 1). For example, if your github account username is *superman*, the <remote-url> for this exercise is:

```
https://github.com/COMP20008/git-workshop-superman.git
```

Verify your remote branch:

```
git remote -v
```

2.3 git pull

The remote repository may have multiple branches, especially for a large collaborative project. These branches represent different sets of changes to the codebase and are often used to implement new featuers without adding any questionable code to the main branch. In most git projects, the main branch is called *master*. More recently, projects are starting to use *main* instead of *master*.

Next, we will clone the *master* branch of the repository. When we clone the remote repository with `git-pull`

```
git pull origin master
```

This automatically creates a master branch locally that tracks the remote master branch (origin/master). Verify that you have cloned the *master* branch at the remote repository *origin*.

```
ls
```

3. Push the updated code to the remote repository

After Exercise 2, you have a local copy of the code repository. You can modify the local copy of the code and add new scripts. You would also test that the code is free of bugs and is functioning as intended. When you are satisfied with the testing, it is time to update the remote repository with your code.

3.1 modify the script

This step is when you perform your development work. You may also run test code during this step and generate some temporary files.

In this exercise, the tasks are to

1. modify the file "dataframe.py" and
2. write the first 5 rows of the iris dataframe to a csv file "iris_sample.csv".

After you have completed the tasks, check the current state of your local repository:

```
git status
```

3.2 undo the modifications

Sometimes, you may want to restore files for example, you accidentially deleted the python script or the script is corrupted. The following command restors the file from the latest commit in the remote origin/master branch:

```
git␣checkout␣origin/master␣--␣dataframe.py
```

Note the spaces between "--" in the command.

3.3 git add and git commit

You many have modified existing files or create many new files. Now you need to update the remote origin/master branch with your changes; this is done by pushing a commit object.

A commit is a snapshot of the local repository. A series of commits form the history of the repository. You should make new commits often, based around logical, atomic units of change that represent a specific idea.

You need to explicitly tell git which files (what changes) you want to include in the next commit. This allows you to control how the history is recorded and to ignore inconsquenctial files, for example, temporary files for testing; you simply leave them out.

For this exercise, we only want to update the file "dataframe.py", i.e., staging only that file for the next commit and leave out the file "iris_sample.csv". Run the command:

```
git add dataframe.py
```

This results in the file "dataframe.py" being staged for the next commit.

Create the next commit object with

```
git commit -m <message explaining the changes in this commit>
```

A message must be included explaining or describing the changes in this commit. Note that running these two commands **does not** update the remote repository with your changes. It only creates a commit object in the local repository. Again, check the current state of your local repository using `git status`.

3.4 git push

All local commits are pushed to the remote/master branch with the command:

```
git push origin master
```

Now go to your GitHub repository and check if "dataframe.py" is updated with your latest change.