Introduction to version control with git

Ariane Ducellier

University of Washington

ESS 490-590 Data science for Earth and planetary systems - Spring 2021

What do you need to run your Python code?

- Your code: Some files with .py extension
- Python package: Libraries that contains many functions related to each other (e.g. numpy, scipy, pandas, scikit-learn)
- To run your code, you need to define its environment: Version of Python + Some packages + Version of packages

When do you need to know your environment?

- You may have on your computer different Python codes with different versions of packages
- You give your code to a friend
- Some of your packages may depend on other packages, with a specific version. How to make sure you have the right version of everything?

How to deal with this?

Install anaconda: https://www.anaconda.com/products/individual

- User interface + command line
- Tools for developing code in Python: JupyterLab, Spyder
- Jupyter notebook (more on this later)
- Tools for managing environment

Basic conda commands

Check conda version to make sure it's installed

```
conda info
```

List out available environments (the starred * environment is the current activate environment)

```
conda env list
```

Basic conda commands (continued)

Create conda environment from environment file

```
conda env create --file environment.yml
```

Removing conda environment

```
conda env remove --yes --name myenv
```

Basic conda commands (continued)

Activate conda environment

```
conda activate myenv
```

Deactivate conda environment

```
conda deactivate
```

Example of .yml file

```
name: MLlabs
channels:
  - conda-forge
  - defaults
dependencies:
  - python=3.9
  - jupyter
  - matplotlib
  - numpy
  - pandas
  - scipy
  - scikit-learn
  - pytorch
```

To learn more about Anaconda environments

https://docs.conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html

How to explain what your code is doing?

A Jupyter notebooks allows you to merge:

- Text
- Images
- Code
- Output of your code

What is a Jupyter notebook made of?

- Markdown cells for the text
- Code cells for the code
- Kernel:
  - You can run the code cells one by one
  - Run all the cells until the end
  - Restart the kernel i.e. delete all the variables that you've created so far

What is Markdown?

Simple language to format text
Used for:

- Text in Jupyter notebooks
- Text on .md files on GitHub (e.g. README.md in a GitHub repository)
- Text on RStudio files

Basic Markdown commands

Headings

```
# Heading level 1
## Heading level 2
### Heading level 3
```

Basic Markdown commands (continued)

Paragraphs: Leave a blank line

```
This is my first paragraph.

This is my second paragraph.
```

Line break: Leave two or more spaces

```
This is my first line.  
This is my second line.
```

Basic Markdown commands (continued)

Bold text

```
**This is my bold text**
```

Italic text

```
*This is my italic text*
```

Bold and italic text

```
***This is my bold and italic text***
```

Basic Markdown commands (continued)

Ordered list

```
1. First item
8. Second item
3. Third item
5. Fourth item
```

Bullet points

```
- First item
- Second item
- Third item
- Fourth item
```

Basic Markdown commands (continued)

Images

```
<img src="images/glass.png" width="200"/>
```

Equations (LaTeXstyle)

```
$\frac{x}{y} = \sqrt{z}$
```

To learn more about Markdown

https://www.markdownguide.org/

First step: Create a GitHub account

Go to https://github.com/ and sign up on the top right corner of the webpage.

Second step: Install git on your computer

Download git from https://git-scm.com/downloads

Why would you want to use git?

- Version control: Keep track of the changes (get back to the version of your code which was working)
- Let two or more people work on the same file, and keep track of all modifications

What is the difference between git and GitHub?

- git is a software: You can use it to track changes only on your computer
- GitHub is a platform that can host your repository. There are other platforms like Bitbucket.

If you have never used git on your computer

You need to set up your user name and your e-mail

```
> git config --global user.name "ArianeDucellier"
> git config --global user.email "ducela@uw.edu"
```

Use the same user name as your GitHub account. On GitHub, you will see who has made the modifications to the repository.
You need to do it only once on your computer.

Let us create a new repository on GitHub

- Connect to your GitHub account
- On the top right of your homepage, click "+" and New repository
- Check "Add a README.md"
- Create repository

Let us do our first commit

- You can edit the README.md file by going on the pen icon
- Click commit to keep track of the modification
- You can now see that there are now two commits to your repository: Initial commit and Update README.md

Let us now work on our computer

The repository now exists on GitHub but not on your computer.

```
> git clone "https://github.com/ArianeDucellier/example.git"
```

```
> cd example
> ls -al
> cat README.md
```

Let us create a file in our repository

```
> git status
> touch my_file.txt
> nano my_file.txt
> git status
```

Let us add the file to the next commit

```
> git add my_file.txt
> git commit -m "Created file"
> git status
```

My branch is ahead of 'origin/main' by 1 commit.

Let us upload the modifications on GitHub

After making a commit

```
> git push
```

Always before starting working in your repository

```
> git pull
```

Let us look at the history of our repository

- Update my_file.txt, commit and push
- On GitHub, click on my_file.txt and look at history
- On your computer, with the command line:

```
> git log
> git log --oneline
```

Come back to a previous version of our file

Make an additional modification to my_file.txt

```
> git checkout my_file.txt
> cat my_file.txt
> git checkout fa708e8 my_file.txt
> cat my_file.txt
```

How to not keep track of some files

```
> touch data.dat
> git status
> touch .gitignore
> nano .gitignore
> git status
> git status --ignored
> git add -f data.dat
```

Work with other people

In your repository on GitHub:

- Go to Settings
- Go to Manage access
- Go to Invite a collaborator

Dealing with conflicts between two versions

Create a conflict:

- Modify my_file.txt on GitHub and commit on GitHub
- Commit the last modifications to my_file.txt on your computer and push

→ You get an error message

- Pull the last version on GitHub
- Open my_file.txt and resolve the conflict
- Add my_file.txt, commit, push
- Check the history of your commits

To learn more about git

http://swcarpentry.github.io/git-novice/

Bonus: Binder

Binder presentation

Binder workshop

The Turing way

Happy code sharing!