A quick introduction to Git

Git is a version control system that helps to manage changes in source code over time. There is a huge amount of content on the Internet that you could read in order to understand how it works. Here we provide just the minimum that is sufficient for the class. Detailed info about structure and features you could find on Atlassian and Git-scm

Basics – Cloning course repository and pulling updates

First, we should clone the course content from the Github repository. *(All actions should be done in terminal)*

1. **Check** if you already have Git installed on your machine:

   OS X, Linux, Windows:

   ```
   $ git  --version
   ```

   If Git is not found go here

2. **Sign up** for Github here

3. **Clone**(copy) repository course content:

   For this computational geophysics course ErSE 390C, we'll setup a local copy. On your machine go to the directory where you would like to have course content, for example:

   ```
   $ mkdir ErSE390C
   $ cd ErSE390C/
   ```

   Download everything from the course repository

   ```
   $ git clone \
    https://github.com/danielpeter/teaching-computational-geophysics.git
   ```

   That's it! Now everything from the course repo is on your machine. But course content will be periodically updated so you will have to be able to fetch changes.

4. **Update** content:

   Whenever the Github repository changed, you can update your local version like:

   ```
   $ cd ErSE390C/teaching-computational-geophysics
   $ git pull
   ```

Pulling updates (in more detail for forked repositories)

When you fork a Github repository, the local updating needs some more configuration. Go to the folder that has just been downloaded. To check existing remote connections use

```
$ git remote -v
```

There here should be only two lines by default that begin with "origin". We are going to add a new connection called "upstream" from where we will pull updates of the course content

1. **Add upstream** remote connection. ONLY ONCE:

   ```
   $ git remote add upstream \
      https://github.com/danielpeter/teaching-computational-geophysics.git
   ```

2. **Pull updates** from the class repo:

   ```
   $ git fetch upstream
   $ git checkout master
   $ git merge upstream/master master
   ```

You will pull updates quite often, so the last point will be frequently used.

More advanced – **Creating and managing your own repository**

While coding your assignments probably you would like to track changes in your files as well. To do it:

1. **Initialize** your local repository:

   ```
   $ git init
   ```

   It creates hidden folder called .git where all changes will be kept. Initialize the directory ONLY ONCE in the very beginning.

2. **Create repository** on Github:

   Go to Github, sign in, switch to "Repositories" tab and create a new repository called "yourrepo". Right after creation of the repository you will be proposed to copy its address – do it, then type

   ```
   $ git remote add origin https://github.com/yourname/yourrepo.git
   $ git remote -v
   ```

   to check that you have two lines that begin with "origin"

3. **Check status** of files in folder:

   ```
   $ git status
   ```

   lets you see which changes have been staged, which haven't, and which files aren't being tracked by Git.

4. **Add** files to tracking:

   ```
   $ git add <file>
   ```

   tells Git that you want to include updates to a particular file in the next commit. One could use

   ```
   $ git add .
   ```

   to add all files in current folder and subfolders. Bash notation works here as well, such as *.m and others. Use

   ```
   $ git reset
   ```

   if you added wrong files with "add" command, or

   ```
   $ git reset -- <file1> <file2>
   ```

5. **Commit** changes:

   ```
   $ git commit -m "my Message"
   ```

Makes a snapshot of all changes in files that were added by git add. Argument -m stands for "message"

Browse history of commits:

```
$ git log
```

6. **Push** commits to Github:

```
$ git push -u origin master
```

After that, all files that were added to commit will appear in your repository on Github

Ignoring file changes

In case if your program creates many temporary files (like images, text files and so on) it worth creating .gitignore file

```
$ touch .gitignore
$ vi .gitignore
```

All types or names of files that are mentioned in this document will be ignored by Git, for instance, add there:

```
*.o
*.m~
*.txt
*.png
*test*
```

Now, they will not appear when you do:

```
$ git status
```