git_tutorial_notebook

September 9, 2015

1 Welcome to Git! Everything you need to know and more!

Let's get started with git. Today we will do the following: - Initiate a repository on our local computer - Create a file and add it to our repository - Make changes and upload those to the repository - Clone an existing repository - Update changes to that repository and push remotely

Create a new directory on your computer that you will use for the purposes of this tutorial.

Side note: The "!" in front of some of the commands allows us to use bash in the ipython notebook. If you type these commands in on the command line you don't need to use the "!" symbol.

If this is your first time ever running git, please do the following:

```
In [ ]: %%!
        git config --global user.name 'knuths'
        git config --global user.email shelley.knuth@colorado.edu
```

```
In [ ]: import os
        desktop = os.path.join(os.path.expanduser('~'), 'Desktop')
        os.chdir(desktop)
```

```
In [ ]: ls
```

```
In [ ]: %%!
        cd /Users/knuths/Desktop
        mkdir knuth_git_tutorial
```

```
In [ ]: cd knuth_git_tutorial
```

```
In [ ]: ls -laF
```

We have not yet set up a git repository (repo) because we just created this folder. We can see this with the following:

```
In [ ]: !git status
```

"git status" is a command that can be used to determine the state of a project in git. If you do not have a project created, it will tell you that, such as the response above.

Let's initiate a git repository. When we do this, an existing directory on our computer becomes a part of git, and git will snapshot the directory and track any changes we make. Without initializing the directory, git doesn't care about the directory, and if we make changes it will not be able to recover those changes.

```
In [ ]: !git init
```

```
In [ ]: !ls -laF
```

We know that this directory has a git repo because we now have a folder .git. Within this folder keeps all the information about any files we might have.

Next, let's create a dummy file.

```
In [ ]: !touch test.txt
```

```
In [ ]: !ls -laF
```

Now if we do a git status we will see:

```
In [ ]: !git status
```

As we said, git status tells us the current state of our project. From this git status, we have learned that we are on the master branch (more on that later), that we are at our initial commit, meaning we have yet to commit a change to the repo (more on that later), and we have one file, test.txt, in the directory, and this file is untracked. This means that git is not currently monitoring this file. While git is monitoring this directory, you have to specify which files you want tracked/monitored/snapshotted. Otherwise, this file doesn't exist in the eyes of git. So let's track the file.

Realistically speaking, you can do this one of two ways. First, I'm going to teach you the "proper" way to do it. Later, I'm going to teach you the shortcut way to do it.

```
In [ ]: !git add test.txt
```

```
In [ ]: !git status
```

When we typed "git add test.txt" we notice that everything is the same except now, instead of having an "untracked file", we have a new file that is staged. When you "add" a file, it is not committed to the git repository. You can use the staging area to add a series of changes that can then be committed together. No matter what, you cannot skip the staging step, BUT, as we will see later, you can use a shortcut that makes it seem like you are.

Now add the sentence, "Git rocks!" to your test.txt file, and then track that change in git.

```
In [ ]: !echo 'Git rocks!' >> test.txt
        !git add test.txt
```

```
In [ ]: !git status
```

You should be aware that if you have multiple files in your git repo that you want to stage/add/track, you can type "git add ." and all the files in that directory will be staged for a commit.

To commit your changes to the git repo, we do the following command:

```
In [ ]: !git commit -m "Initial commit, add test.txt file"
```

A commit allows you to record all of your staged content. You can use these recordings, or snapshots of your work, to compare to other commits or snapshots.

Now if we do a git status, we see:

```
In [ ]: !git status
```

We are up to date in our directory, and all our changes have been committed.

Another useful command to use is "git diff". This command allows you to see what is different about the file now from what has been committed. git status allows you to see this too, but git status only tells you if there is a difference; git diff tells you what that difference is.

```
In [ ]: !git diff
```

There are no changes. Let's change our file and add the change.

```
In [ ]: !echo 'Git really rocks!' >> test.txt
```

```
In [ ]: !git diff
```

Here we can see that "Git rocks!" is part of the file, and we've added (+) the line "Git really rocks!"
Let's do another git status:

```
In [ ]: !git status
```

We see that we've modified our file and have not yet added/staged those changes. Let's do a short cut to add and commit the changes rather than doing it in two steps:

```
In [ ]: !git commit -am "Added a second line to test.txt"
```

This allows you to stage any file that was in your last commit and has been modified. You still have to do the "git add" command for new files.

```
In [ ]: !git status
```

There is A LOT more you can do with git, but this will get you started.
Now it's your turn. Do the following:

- Create a file called git_tutorial.txt
- Create a second file called git_tutorial_second.txt
- Add a line to each file
- Stage and commit the changes to the repo

2 Remote Repositories

The last thing we will do is work with remote repositories.

One of the great things about version control systems is that you can copy other's remote repositories or make your own. This allows many things: - Making public something really great you've done! - Grab something really great that's been done! - Group collaboration on a specific project

2.1 First - Copying a Remote Repository

Get a repository from either copying one that exists or importing an existing project to git To copy an existing git repository, type

```
In [ ]: ls -laF
```

```
In [ ]: !git clone https://github.com/ResearchComputing/Final_Tutorials.git
```

This allows you to receive a copy of all the data the server has.

How did I know where to get that URL? You can find it on the github repository site: https://github.com/ResearchComputing/Final_Tutorials

```
In [ ]: ls -laF
```

2.2 Second - Making a Local Repository Available on Github

First, you need to create a github account. A standard github account allows you to share your local data freely with the general public. For a fee, you can keep repositories private on github.

https://github.com/

To then make a local repository available on github, create a New Repository on github. Give it a name.

```
In [ ]: from IPython.display import Image
        Image(filename='/Users/knuths/Desktop/Untitled.png')

        # Here's a screenshot
```

Then they very conveniently tell you want to do next, giving you several options. I suggest typing the following commands into your command line from the directory where you want to push up your git files:

```
In [ ]: pwd

In [ ]: ls -laF

In [ ]: !git remote add origin https://github.com/knuths/first_github.git
        !git push -u origin master

In [ ]: ls -laF

In [ ]: !git status

In [ ]: !git add Final_Tutorials/
        ! git commit -m "Added Final_Tutorials repo"

In [ ]: !git status

In [ ]: !git push
```

Then go to github and see the Final_Tutorials directory on the site!

If you are sharing this directory with others, you can do a git pull to grab their work and add it to the repo on your local machine.

```
In [ ]:
```