

Useful Git commands

Nir Zadok

April 18, 2021

Contents

I	Introduction	3
II	Using Google Drive in Google Colab	4
III	Using Git	5
1	Initializing	6
2	On-basis commands	8
2.1	Used all-the-time commands	8
3	About branches	9
4	The workflow	11

Part I

Introduction

In this paper I will introduce useful Git commands and the workflow of:
Google Colab \leftrightarrow Github.

Part II

Using Google Drive in Google Colab

In order to use get access to Google Drive trough Google Colab, one has to use the following commands:

```
from google.colab import drive  
drive.mount("/content/gdrive", force_remount=True)
```

Then one has to change folder to our working directory:

```
%cd /content/gdrive/My Drive/Colab Notebooks/{folder_name}/
```

Now one can see the files inside the directory and also the path by using:

```
!ls  
!pwd
```

Part III

Using Git

Now we will start using Git in order to connect our Google Colab and Google Drive with Github

1 Initializing

First, we will go to our Drive folder and check if there exists a .git folder:

My Drive > Colab Notebooks > my_ising_repo ▾




Name ↑	Owner
 .git	me
 git_file.ipynb	me
 simple_ising.ipynb	me

Figure 1: .git folder

If there is, delete it by using:

```
rm -rf .git
```

Then, we will have to initialize empty Git repository by using:

```
!git init
```

Next, we will define our remote repository on Github:

```
!git remote add origin https://{Github_username}:{Github_password}  
@github.com/{Github_username}/{repository_name}.git
```

If there is a problem while creating it, use the following command:

```
!git remote rm origin
```

Now we have to config our account:

```
!git config --global user.email "{Github_mail}"  
!git config --global user.name "{Github_username}"
```

In order to get updated from the Github repository:

```
!git fetch
```

2 On-basis commands

2.1 Used all-the-time commands

1. The status command helps us learning about our status:

```
!git status
```

2. In order to refresh the git, we will have to use:

```
!git reset
```

If Git is worried about files unwritten, use these commands, while add is responsible for adding and stash is responsible for the saving:

```
!git add --all  
!git commit -m "save the unwritten files"  
!git stash
```

In order to get a list of the stashed files, one can run:

```
!git stash list
```

Then one is able to load a stash by using:

```
!git stash apply {stashed_file}
```

In order to get clone of the remote repository, one can run:

```
!git clone {repo_link}
```


3 About branches

1. Check in which branch you are by using:

```
!git branch
```

For example, here we are at new-feature branch:

```
main
my_branch
new-branch
* new-feature
```

Figure 2: Where we are

We can see it also by using:

```
!git status
```

```
On branch new-feature
nothing to commit, working tree clean
```

Figure 3: By using !git status

2. We can change to the desired branch by using:

```
!git checkout my_branch
```

3. We can also create new branch by using:

4.

```
!git checkout -b my_branch
```

5. And also create a new branch like the active branch (say we are inside our branch and we want a copy from it):

```
!git branch new-branch
```

```
!git checkout new-branch
```

4 The workflow

1. We start inside our working branch
2. Next we will use the commad:

```
!ls
```

in order to check what files we have in this Google Drive directory,
and also use:

```
!git status
```

to check what our status is.

3. We will begin by reset our commit:

```
!git reset
```

And also create a new branch like the active branch (say we are inside our branch and we want a copy from it):

```
!git branch new-branch
```

```
!git checkout new-branch
```

4. Now we can add a file to our working branch by using:

```
!git add {file_name}
```

Then we will have to create a commit, by using:

```
!git commit -m "Push files into a branch"
```

```
!git push origin {branch_name}
```

We will go to the branch we want to merge into it:

```
!git checkout {desired_branch}
```

Now, will reset first:

```
!git reset --hard {merged_branch}
```

We will add our commit:

```
!git commit -m "Merge the branch into another branch"
```

and pull to get all our data before merging:

```
!git pull origin {the_branch_we_merge_into}
```

Or the same if to get the data from the remote repository by using:

```
!git fetch
```

and then to use merge:

```
!git merge {merged_branch}
```

And finally we will commit and push it:

```
!git commit -m "merge it"  
!git push origin main
```

Then one can delete the branch from the remote repository and also from the local repository:

```
!git branch -d my_branch
```


References

- [1] My Files