

Memo

M.A. Temiz

March 4, 2024

This document will be used for taking some notes about how to do certain things.

1 How to git?

1.1 What is git?

`git` is a very important tool for all type of technical people. Its primary target audience may be people who develop codes, scripts, etc. but that's not the only usage. If one (or more) items below apply to you, then you can also benefit from its capabilities:

- **If you are working on a long-lasting project:** Over time, it is possible that you need to update the files, documents you work on. And that can create problems with the rest of the project. For example, you have some test data. Based on that data, you calculate the performance of the part you designed. But you thought you can improve your calculation method and did some changes on them. But suddenly, your script starts throwing all types of error to the screen and you don't even know where to look. Since `git` shows you what types of changes you have done over time, you can go back to a previous version and restore the functionality.

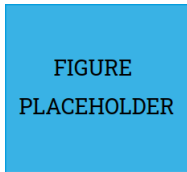


FIGURE
PLACEHOLDER

Figure 1: Schematics of a long-lasting project.

- **If you are working on a project with multiple people:** You are in a project that one person can not handle. So, input from multiple people are necessary. But, how can we deal with people trying to work on the same files at the same time? What about contradicting inputs? Or contributions not being compatible with the rest of the project... Of course all that alignment task should be addressed by a project manager but as the team grows, this job becomes more and more overwhelming. Same goes for the team members also. They need to consider multiple things before submitting their work. All of those can be reduced by a well-organized system and set of tools, which `git` provides.

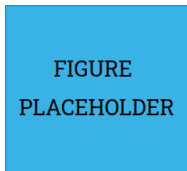


FIGURE
PLACEHOLDER

Figure 2: Schematics of a project with multiple people.

So, we can make use of this program to trace the development of our work. But if you try to read its original documentation, and things can quickly get a little overwhelming. I suspect that the original documentation is written in that way to make sure that it serves as a "filter" so that only people with version-control background can quickly start using it. But I believe that shouldn't be the case. Everyone who needs it should be able to be use! Because this program is simply genius!

I am not going to lie... It took me several attempts to understand how it works. In order to make sure that I don't have to relearn everything next time, I started taking some notes. And then I thought, "What if there are people like me who may benefit from these quick-start tips?" And I decided to make these notes more readable. Here we are!

1.2 How Does git Work?

First of all: `git` and `github` are not the same thing. `git` is the program. mostly works on your computer locally, where `github` is one of the online storages where you can upload your work. There are other options such as `gitlab`, `bitbucket`, etc.

`git` is a program. It is an open-source program written by the famous Linus Torvalds, the same man who shared the Linux kernel with the world. Similar to Linux, `git` is also free. Since these notes are focusing on the "open-source" world, I will assume you are also using a PC running a Linux distro. If not, I am pretty sure you can find better notes or tutorials how to do the things mentioned in these notes somewhere else.

2 Using a Deploy Key

This is the scenario: "You are working on a project locally on your PC. You want to upload your changes to your remote repository on Github. You looked up the commands to do it, you typed them on your terminal but you keep getting the following error message:

```
fatal: Could not read from remote repository.  
Please make sure you have the correct access rights and the repository exists.
```

There are various ways to get passed through this "wall". In this section we are going to talk about using a "Deploy Key". The philosophy is simple. Imagine your remote repository as a book with a lock on.



Figure 3: Your repository.

To protect the contents of this book, you need to "make" or "buy" a special lock for this book. You go to a locksmith, in our case this locksmith is noone other than the famous program `ssh`, and ask them to make you a special lock and a key. How do you do that? Using the following command:

```
ssh-keygen
```

When you enter the command given above, a key and lock pair is generated. However, it is mentioned as "key-pair" in the terminology. Let's stick to the terminology. The program asks you to provide it with the location to save this *key-pair*. The best practice is to save it in the (hidden) "`.ssh`" folder in your "home" directory and give it a name that is relevant to the project you are working on: such as "`id_mysecretproject`"

If you navigate to the folder where you saved the generated key-pair, in our example it was "`~/ .ssh/`", you will see 2 files with the name chose:

- `id_mysecretproject`
- `id_mysecretproject.pub`

These are simply text documents. You can open them with `nano`, `gedit`, `vim`, or whatever you want to open with. The one with `.pub` is the "lock" in our analogy and the one without is the "key". So you keep the key, and use the lock to secure your repository. You need to copy the contents of the file with "`.pub`" extension, which stands for "public", and paste it to the "Deploy Key" section of your repository. Then you need to tell `GIT` which key to use.

To do that, you need to navigate to the your local repository - in other words, your project folder. Check if you are in the correct folder by running `ls -la`. If you see `.git` folder in the listed folders and files, you are at the correct location. Then run the following command:

```
git config core.sshCommand "ssh -i /.ssh/id_mysecretproject -F /dev/null"
```

In the above command, you should of course update `texttt /.ssh/id_mysecretproject` part with the correct path to your generated ssh key. But the rest stays the same. After doing this, you can easily *push* your changes to the remote repository, a.k.a. your repository on github.

Please note that, a similar approach can be used for other GIT hosts. The mentality here is to generate a *key-pair* and telling the `git` program where to find the correct key to open the remote repository.

If you want one more step of security, you can add the `-o` option to the end of the command. Then it will ask you to provide a password during the setup.