Git: Setup

Windows

Gangaprasad Koturwar

What is Git?

One of the most important aspect to take care of during coding is the maintaining the code. Code undergoes changes frequently and it becomes important to properly maintain the versions. Git is a distributed version control framework that helps in:

- Codebase maintenance
- Project collaboration
- Version exploration

Git and its vast uses can be best understood through its usage.

Getting started!

Register account

Visit the GitHub website. (https://github.com/)

First thing we need to do is create an account. Clicking on sign up should get the registration process initiated.

Provide the basic information needed and get your account registered.

Set up Git

Once you have registered yourself, next step is to setup git on your local machine.

Download windows specific git application from the official website (https://git-scm.com/downloads)

Install the application. Follow the installer instructions.

Set up password less access

Setting up password-less access is exactly what it entails. You should be able to access your GitHub account without having to explicitly provide your password every time. This could be alarming as it might raise some security concerns. But we will go through the steps to understand what we are going to do and answer our concerns. Let's understand and establish very important fact before we dive into setting this up:

- **You have your personal machine. And no one else accesses it without your authorization.**

This is a very important fact, as any level of security can be breached if the user themselves are not cautious. If we agree to this, let's see why setting up password less access makes sense:

- Access to machine itself is authorized. So, once you logged in, not having to keep providing your passwords every time you want to make some changes to your code would be more convenient.
- So how can we achieve that? We have authorized the machine when we logged in. So, can we authorize the machine itself now with GitHub so that GitHub and machine will exchange the authentication protocol, without us having to stepping in every time we are trying a communication with GitHub cloud?
- This is exactly what we are trying to do now. We will create a key on our machine and share that knowledge with GitHub, so that GitHub can authorize communications from our machine as well as know which account it is associated with.

Now understand this with real world example. When we use our password, and it is leaked somehow anyone can basically access your GitHub account. But authorizing your machine, one will need the key as well as your machine to access your account. And that's why the important

fact that we discussed about personal machine is more important. Thus, as it should be obvious to you now that any security measure is only as effective as you!

Now that we have established the concept let's implement it.

ssh-keygen

SSH (**S**ecure **Sh**ell) is a network protocol. We can learn about different network protocols and their usages separately. SSH protocol will help us in communicating with the GitHub cloud. Now since GitHub understands the language of this protocol, it suffices to say that we need a key which is compliant with this protocol.

- Open command line interface in windows and run *ssh-keygen*
- Follow the instructions and provide necessary information when prompted
- If you did not change the default path, the keys should be generated at *C:/Users/YOUR_USERNAME/.ssh/*
- If you had provided any other path (Which is generally advised against) keys will be delivered to that path.
- You will see two files (*id_rsa* and *id_rsa.pub*) generated at this location. These two files will constitute as a key for your machine.

Let's try to understand what these files are. *id_rsa* is the actual key that is generated for your machine. *id_rsa.pub* is a public key that is generated your machine's key. These are dependent on each other. Whoever wants to connect with your machine should have this public key. They will share this key, and your machine will verify it against the private key it possesses. If it's a valid match (match is not exact letter to letter match, but the encryption algorithm decrypts both these keys and then verifies. Encryption is a very fascinating technology, and I would suggest one should look it up). Now since the key was generated on your machine, only your machine can decrypt it. So, if someone steals the keys, you need not worry. As they are useless unless and until they get your machine as well. When you lose the machine itself, well you should panic!

So now we need to share the public key with GitHub cloud so that whenever we are trying to communicate with the cloud, private key on our machine and the public key in GitHub cloud will be used to authenticate the access. Open the *id_rsa.pub* file and copy the contents.

Upload public key to GitHub cloud

Go to https://github.com and log into your account. Go to settings. Click on your photo on top right corner, and settings navigation should be available in the dropdown. Alternatively, you can directly go to https://github.com/settings/profile

Click on *SSH and GPG keys* from the available navigations. Click on *New ssh keys* button. This will open a form for you to fill in the details.

You can set the title as anything that should be helpful for you to recognize your machine. This will be useful when you have multiple machines. And paste the contents of *id_rsa.pub* file that we have copied in previous step. Click on *Add ssh key* and GitHub should now be able to identify and authenticate you through your machine.