Setting up SSH and Git on Windows 10

You can install Git from here:
https://git-scm.com/download/win

You can also install Git via chocolatey:

```
choco install git -Y
```

Create a SSH Key

The first step is to generate a new SSH key. Use cmd or Powershell and run the following command:

```
ssh-keygen -t rsa -b 4096 -C "your_email@example.com"
```

You can but don't need to give it a passphrase since you should **never** share your secret key around but using one will secure your keys. Keep in mind that everybody can have as many private keys as they want.

This generates a new private SSH key with rsa encryption and 4096 bits. It also generates a public key from the secret key which you can share around.

There will be a new folder and files in your Windows user folder.

In general you can create as many keys as you want.
The **id_rsa**key is the default key generated by ssh and will be automatically be used by your ssh-agent if you don't tell it to use another key.

What is an ssh-agent?

An ssh-agent is the agent process used to actually authenticate yourself with ssh. There are a few out there (PuTTY with Pageant for example) but for this example we'll use the ssh-agent provided by the native and default Windows 10 ssh-agent.

If you want to you can use PuTTY and Pageant to make your keys even more secure. Read this post on Digital Ocean for more information.

If you want to change the key used by your ssh-agent, you must first start the service. The service will be disabled on Windows 10 by default. Search for Services and open the Services settings and look for the "OpenSSH Authentication Agent" and Activate it:

Now you will be able to access the ssh-agent from your console via `ssh-agent`.

For this example we're going to try to load another key called `example` into our agent and use it instead of the `id_rsa` key. To do this you can run the following command:

```
ssh-add example
```

Now you will have both keys available for this session.

Register your SSH Key on Github

The next step is to register your generated SSH key on Github. For that, run the following command:

```
type C:\Users\your_user_name\.ssh\id_rsa.pub
```

and copy the output string into your clipboard. Now go to your Github keys settings and add a new SSH key with your public key and save it.

Congratulations! You now are able to get and push code to Github without any password!

Note: There should also be a `C:\Users\your_user_name\.ssh\id_rsa` file. This is your private key, don't share this around!

Setup Github in your Shell

Now it's time to setup Git on your machine. After installing it from the link above, open a new cmd or Powershell window. Now we need to set your public Git name and Git email address. This will always be public when pushing code.

Luckily Github gives you a privatized email address for use. Go to https://github.com/settings/emails and you will find a `@users.noreply.github.com` email address for your account. Copy this email address.

Next register your name and email in Git:

```
git config --global user.name "Your Name"
git config --global user.email your_email@users.noreply.github.com
```

Congratulations! Now all your Commits will be registered as being commited from your Github user.

Signing your GitHub commits (Optional Step)

To sign your commits you first must install the GPG command line tools. After you installed the GPG toolkit, you can run the following command to generate a new gpg key:

```
gpg --full-generate-key
```

This will ask you what kind of key you want. Go for RSA and RSA.

Now you need to enter a bit length. The recommendation is **4096 bits**.

After that you can specify a expiration length or if the key should never expire. Pick as you want. Expiring keys are more secure in general because you have to renew them every now and then.

Now enter your personal informations to verifying your identity with your gpg key.

When you're done you will be asked for a passphrase. Give it a secure passphrase and you will be done with your gpg-key generation.

After that you will be able to find your key in your users `.gnupg` folder as specified in the success message.

If you want to list your gpg keys, simply run

```
// short version
gpg --list-secret-keys
```

```
// long version
gpg --list-secret-keys --keyid-format LONG
```

Your GPG key you can share with Github is the key coming after `sec rsa4096/` so for example in

```
/Users/hubot/.gnupg/secring.gpg
------------------------------------
sec   4096R/3AA5C34371567BD2 2016-03-10 [expires: 2017-03-10]
uid                          Hubot
ssb   4096R/42B317FD4BA89E7A 2016-03-10
```

the gpg key would be `3AA5C34371567BD2`

To get your public key block, simply run

```
gpg --armor --export YOUR_GPG_KEY
```

which will output your public GPG Key Block. Copy it and paste it to your GitHub Account here.

From now on your commits will be signed when commited.

Use Git

Now you're ready to actually use Git. From now you can clone repositories via `git clone` or push new code to Github. Here is a quick reference:

```
# Clone a repository to the current directory
git clone [REPOSITORY_CLONE_URL]

# Create a new commit with a message
git commit -m "Your commit message"

# Add files to the commit
git add .
git add ./filename.ext

# Push your commits to Github
git push origin master
git push origin [YOUR_BRANCH_NAME]

# Reset your repo to the last version
git reset --hard

# Create a new branch
git checkout -b [YOUR_BRANCH_NAME]

# Switch branches
git checkout [YOUR_BRANCH_NAME]
git checkout master

# Reset a single file
git checkout ./filename.ext
```