**Git**

Is a version control system

**Commands**

* **To get installed git version**

*git --version*

* **To turn your project directory into a git repository**

*git init*

You have to be on the root directory of your project

It will create a .git folder.

* You will not be able to see the directory as it is a hidden folder so use ls -a or dir /a
* Do not manually change the .git folder, or you may corrupt it.
* Git will not automatically track your files and changes.
* **To check the status of your repository at a certain point in time.**

*git status*

It will show you what folders/files are being tracked and untracked in your project.

* **To get files to a staged status**

*git add folder/file*

This will change the folders/files from untracked to staged they can be removed from this status by using the *git rm --cached folder/file name*

* **To remove files from a staged status**

*git rm --cached folder/file name*

* **To ignore all unstaged folders/files**

Create a file on the root of the project folder called **.gitignore** and inside it just add the folders/files you don’t want git to track at all. When you rum status the files in the .gitignore file will not show as untracked anymore.

You should add .gitignore to your repository.

* **To commit the changes to your project into the repository**

*git commit -m “Initial Commit”*

all the files on the unstaged state will be committed to the repository saving any changes you made. The –m flag means a message that you add with quotes. Messages should be used to let you know the reason for the change, which will be useful when looking through the changes over time.

**To add ssh keys**

* **To check if ssh keys already been generated**

***ls --al ~/.shh***

The ssh files will always sit in the users folder depending on your log in.

* **To generate the ssh keys**

*ssh-keygen -t rsa –b 4096 -C ”your email”*

On the settings for now use the default choices.

Two files will be generated

id\_rsa this file is your private key

id\_rsa.pub this file is the one you can pass around to git hub

* **To start the ssh agent service**

*eval “$(ssh-agent -s)”*

* **To add my generated keys to the ssh agent**

*ssh-add ~/.ssh/id\_rsa*

Should get a message of Identity added

Now the ssh agent is aware of your private and public identity and will now use it when contacting third party services like github.

* **To add key to github**
  + Create account
  + When you login navigate to your profile picture click on the arrow by the side of it.
  + Go to settings
  + Select SSH and GPG keys
  + Select new SSH key
  + Always add a title which will id the machine/person using that key
  + Copy key from the .ssh/id\_rsa.pub to key.
  + Then click add key
  + A key should now show on the list of ssh keys.
* **Test your connection to github**

ssh –T [git@github.com](mailto:git@github.com)

you should get a message say yes, then another message saying something like “Hi username”

* **Add a repository on github**

Navigate to repositories

Click new repository

Give it a name

And a description

Click create repository

Since this is a new repository it will give you instructions on how to push the local repository to the remote one, check that you are using the ssh instructions or you will be asked for your login credentials.

Instructions

* + - create a repository

echo "# node-serial" >> README.md

git init

git add README.md

git commit -m "first commit"

git remote add origin git@github.com:enginebeat/node-serial.git

git push -u origin master

* + - push an existing one

git remote add origin [git@github.com:enginebeat/node-serial.git](mailto:git@github.com:enginebeat/node-serial.git) (don’t think I need this part.)

git push -u origin master

if you refresh the page on github it will show you all the details of the repository .

* **Getting a repository you created into another machine or if want to restore your files.**

Not sure if this is the right or the best way to do it but it works...

After you connected your machine to github by using/creating your ssh keys

First things first, try to keep project/folder/file names consistent!

* + On the CLI navigate to the folder where you want your project folder to reside, no need as far as I can see to create the actual project folder.
  + Use *git clone* [*git@github.com:enginebeat/node-serial.git*](mailto:git@github.com:enginebeat/node-serial.git) (repo as an example). This sort of link will be shown if you click on clone or download button on the github repo page.
  + Check that your project was added to the file system.
  + All of the files should have been created including all the git/repo related ones.
* **To commit changes on your local repo to the remote repo in github**
  + After staging the modified file(s) by using git add folder/file name, commit the changes by using git commit command.
  + Then use *git push –u origin master*. You should get a message with something like completed with 1 local object.
* **To fetch changes from the remote repo**

This will be the normal situation. You worked on a machine and now need to work on another machine, let’s say at home and need to pick up the latest changes you made.

* When in the project folder use git fetch [*git@github.com:enginebeat/node-serial.git*](mailto:git@github.com:enginebeat/node-serial.git)

This will not immediately update your local folders/files with the remote changes.

Tried to use this with merge and can’t make it work... need to check the usefulness and how to use it in the future. Use pull instead which seems to do both things at once in one operation.

* **To pull changes made to projects from the remote repository to your local repository**

This will be the normal situation. You worked on a machine and now need to work on another machine, let’s say at home and need to pick up the latest changes you made.

Now, this seems to be the right command to get changes added directly to your local repo. According to the docs it will perform a fetch and a merge at the same time, which makes more sense for me at this point. I don’t really see the point on using fetch...

When in the project folder use:

*git pull* [*git@github.com:enginebeat/node-serial.git*](mailto:git@github.com:enginebeat/node-serial.git)

Questions:

What is git?

How do you check the version of git you have installed?

How do you turn the project folder into a git repository and where do you have to be to do that?

When creating a repository what directory name will it generate?

Can you see this folder normally when you use ls or dir, and if not how would you make it visible?