Git

**Version Control**

Version Control is a system that records all the changes made to a file or several files so overtime when you wish to view a specific version of a file or revert your current version to an older version you are able to do so.

**What is Git**

Git is a distributed version control system. The difference between Git and other version control systems (VCS) is that Git takes its data as snapshots versus a set of files as other version control systems (VCS) do. With Git, every time you commit, Git takes a snapshot of what all your files look like at that moment and stores a reference to that snapshot. If no file changes, Git will not store the file again instead Git will store a link to the previous identical snapshot it has stored.

* **Git vs other VCS**

One of the benefits of Git is the fast speed to access your files or browse file history because almost all operations in Git are local. You are free to do your work even if you are offline. Git uses checksums or the hash value of the file contents to store the data to make sure the validity of data.

* **Git’ s three states and three main sections of a Git project**

Git has three states for your files: modified, staged and committed. Modified means the file has been changed but is not committed to the database. Staged means you have marked the edited file to be committed to a snapshot. Committed means the data is permanently saved in your local database.

With these three states, there are three main sections of a Git project: Working Tree, Staging Area and Repository/.git directory. The work flows starts from you checking out a file/files from the Git directory to your working directory for use, this is called Working Tree. The Staging Area or index is a file that stores the information of what will go into the next commit. The Git directory is the database that stores all your data.

**Install Git**

* For Windows, download from <https://git-scm.com/download/win>
* The first thing you should do when you install Git is to set your username and email address. This information will display each time you commit. Here is a screen shot of example.

$ git config --global user.name "John Doe"

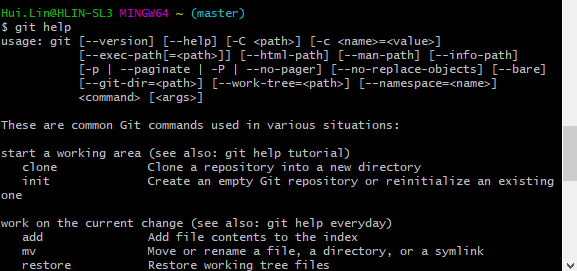
$ git config --global user.email [johndoe@example.com](mailto:johndoe@example.com)

* By default, Git creates a branch called master when you create a new repository with git init command

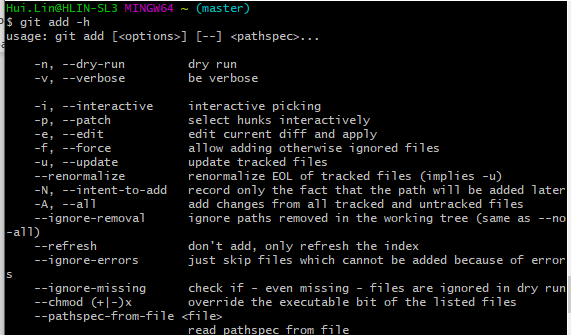




* To get a list of Git commands use git help



or git add -h to get a more concise list of commands

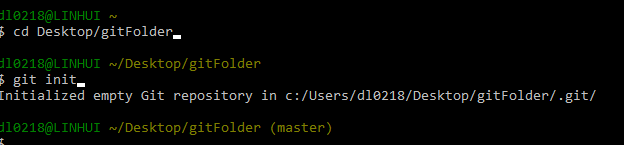


**Git basics**

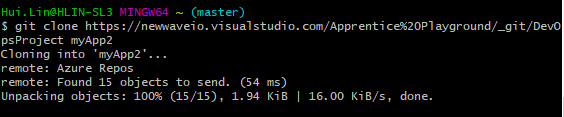
* Getting a git repository

There are two way of achieving it. To create a new repository with your local work directory or clone an existing one from somewhere else.

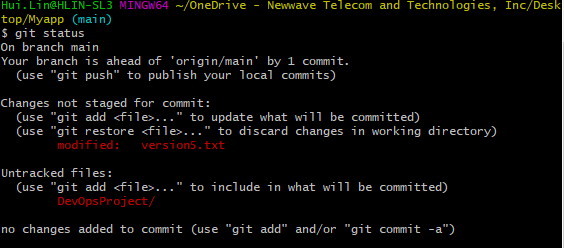
To create a new repository with a local directory see the screenshot below. Go to the directory and type in git init command.



To clone from an existing repository to my local directory myApp2, use git clone <url>. Note you cannot clone and add a local directory at the same time. Trying to do it will give you an error.

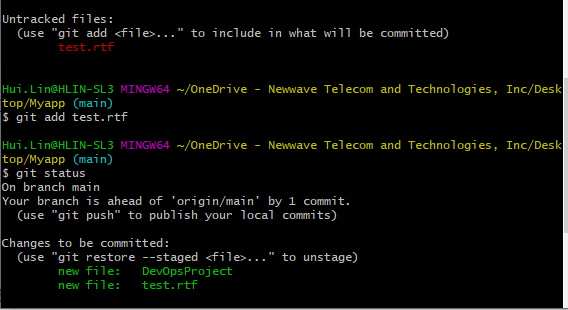


* Use git status to check the status of your files. The screenshot shows there is one file of version5.txt that has been modified but has not been committed and a file called DevOpsProject that is untracked meaning a new file to Git.

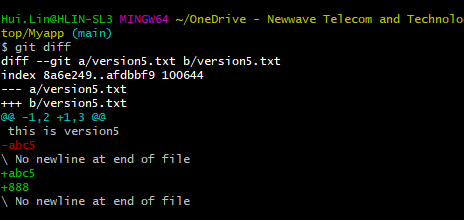


* To track a new file

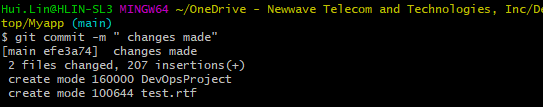
Use git add <file name> , the file test.rtf changes from untracked file to “to be committed" in the screenshot below. The file test.rtf now is staged meaning it is now tracked in Git. An important note: If you modify a file after you run git add, you have to run git add again to stage the latest version of the file.



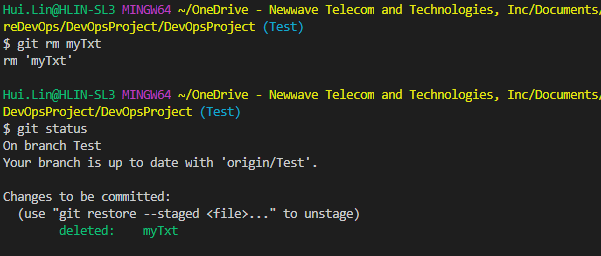
* To see what you have changed but not yet staged, type git diff. The green + abc5 and +888 in the screenshot below are the changes you added to file



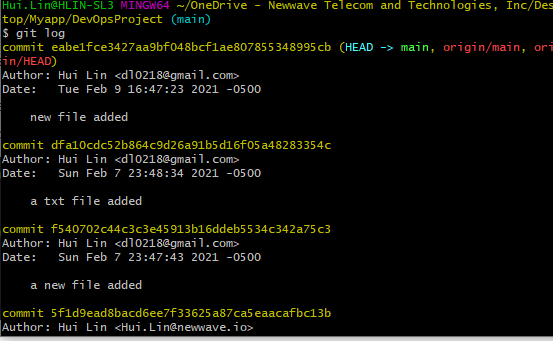
* To commit your changes use git commit -m “message”



* To remove a file use git rm <file name>. In the following screenshot, file myTxt is removed.



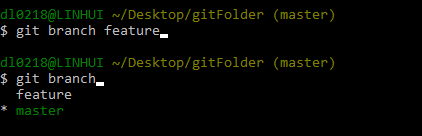
* To View project history use git log



* To make new a branch

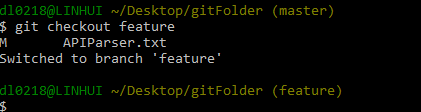
git branch <branch name>: to create a new branch named “feature”

git branch: to get a list of all branches. The asterisk marks the branch you are currently on. Now I have two branches: master and feature and I am currently on master branch.

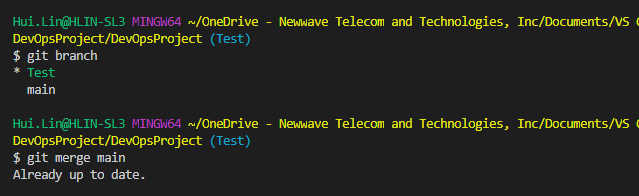


* To switch between branches

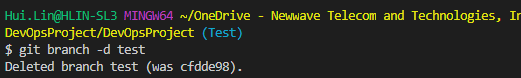
git checkout. Now I am switched to feature branch from master



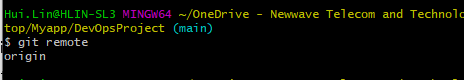
* To merge between branches use git merge <target branch>. Now test branch is merged with main. If conflict arises during your merge, you would have to manually resolve the conflict and merge again.



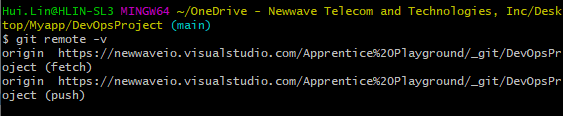
* To remove a branch use git branch -d < branch name>



* To show your remotes use git remote



Use git remote -v to show the url of the remote repository

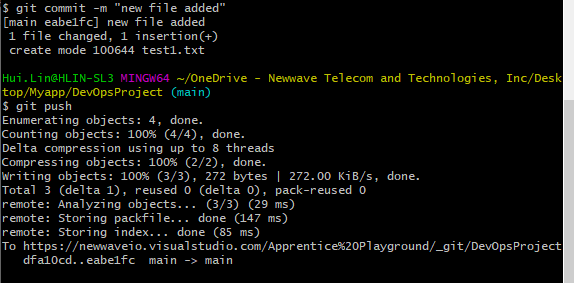


Fetching and Pulling from Your Remotes

* git fetch command goes out to that remote project and pulls down all the data from that remote project that you do not have yet. git fetch command only downloads the data to your local repository — it does not automatically merge it with any of your work or modify what you are currently working on. You have to merge it manually into your work when you are ready.



* To push changes to a remote repository use git push . In the screenshot after I add a new file and commit I push it to a remote repository



* To brings changes made in a remote repo to your local repository use git pull

