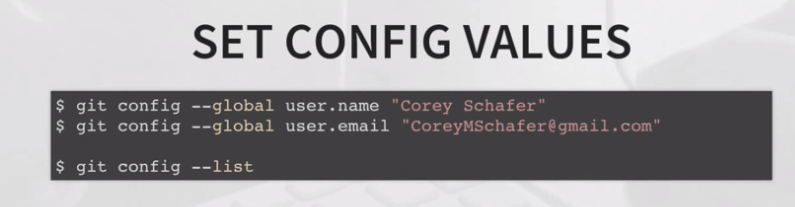
**Git and GitHub**

1. **Git Tutorial for Beginners: Command-Line Fundamentals**

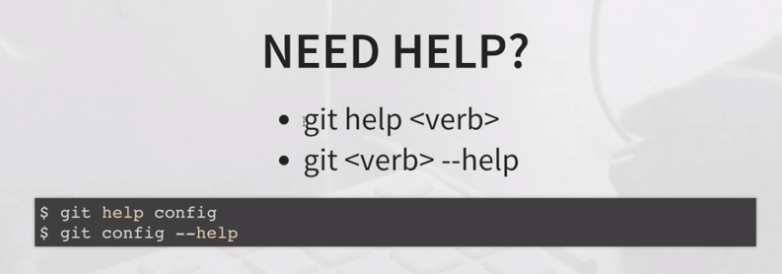
<https://www.youtube.com/watch?v=HVsySz-h9r4>

After installing git in our local machine, desktop or laptop, we need to execute following two commands



Above commands set few global configuration variables. These are important when we work with multiple developers. Values of these variables help us to identify who has checked in and out the code and made changes. It is one time operation – once it is set by git, it will be reflected in all the project folders or work spaces in the same system. We are now done with initial configuration. Let’s move on.

If we need help for any of the git action we need to execute following command,

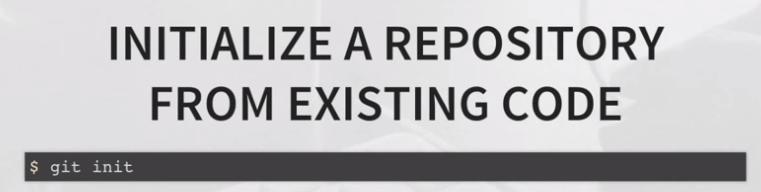


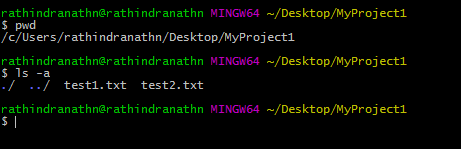
Both the commands work the same. It displays the man page of that particular git action. For example, either of the above two commands displays the man page of “config”, a git action to set value to a variable for tweaking the environment.

**Getting started – Two common scenarios**

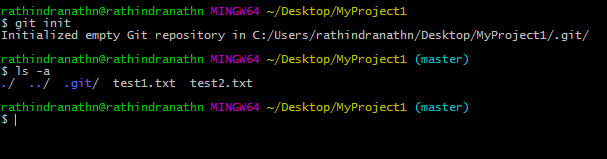
There two common scenarios for people to begin working with git. First scenario, we have an existing project in our local machine and we want to start tracking it using git. Second scenario, there is an existing project remotely we want to start developing on.

Let’s start discussing on first scenario – there is an existing project in our local machine, which we want to start tracking on. For that first we need to go to the folder where the project resides and then execute the command “git init”. It will create a local git repository in the project folder. Now onwards all the changes, made to the project files will be tracked by git.



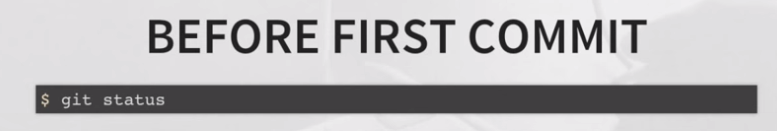


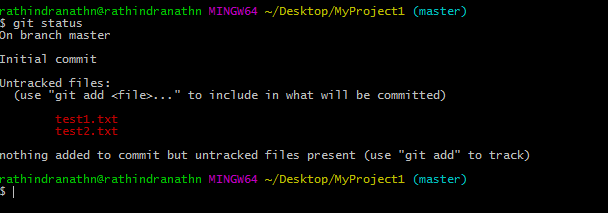
Currently we are in “MyProject1” directory in the Desktop. There are two files test1.txt and test2.txt in it. Now we shall initialize this folder by git so that all the files in it are tracked by it.



We have executed “git init” command in our project folder MyProject1 in order to track all the files in it by git. As a consequence of running the command a new hidden directory .git is created in MyProject1. It contains everything related to the repository in our project folder. If we want to stop tracking our project with the help of git, all we need to do is deleting the .git directory from the project folder Myproject1. But we don’t want to do that here.

We have initialized our project folder for tracking by git, but we have not commit anything till now. Before committing, let’s check the status of git.

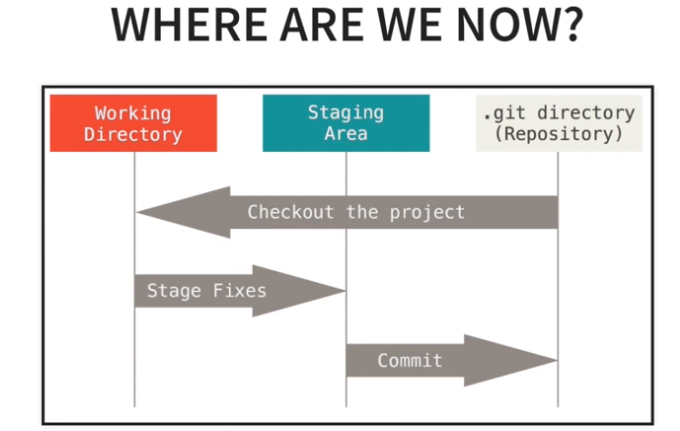




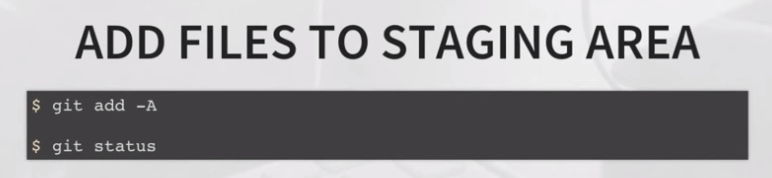
Above output is showing that we have two untracked files test1 and test2.

Before proceeding further, we need to look at one important point. It may happen that we are not intended to track all the files in our project folder by git. Git provides a technique to achieve this – we have to create a file called .gitignore in our project folder and keep all the files we don’t want to track.

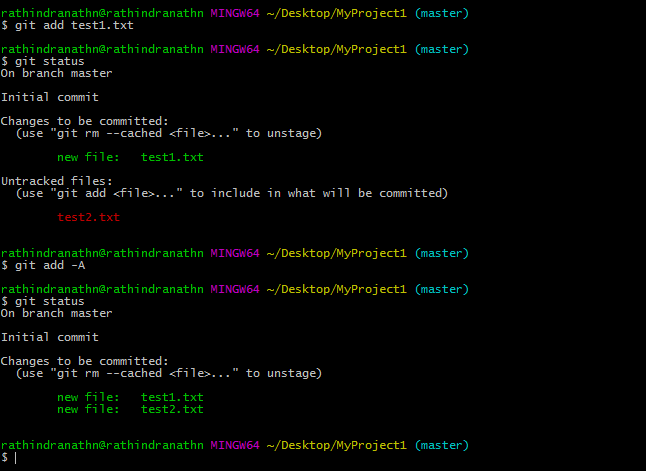
Let’s try to get the picture where we are rightly now. Following figure will help to depict it. In git there are three stages we need to know about – working directory, staging area and committed files. Right now we are in working directory. All the untracked and modified files will be in working directory. Those untracked and modified files are listed when we execute “git status” command. Staging is where we organize our files, which are to be committed to the local repository. The reason behind staging is we can finally pick and choose the files for commit to our local repository. If we work on multiple files – we can keep them in staging area first and then commit them in small chunks. It allows us to be more detail about commit. Suppose we have made few changes to a file, but we are not completely sure whether they are enough. In such scenario we can keep the file staging area and we can revert it back to our working directory and start working on it further if wish to do it. In this way we get a chance to modify a file further before committing it in a situation where we are not completely sure that we are fine with the change we made.



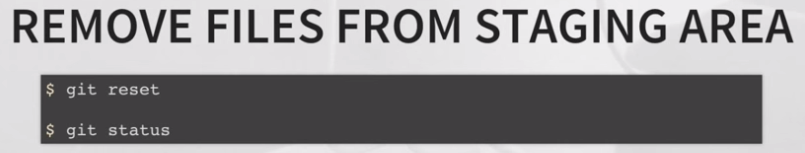
To add all the files from working directory to the staging area we need to execute following command,



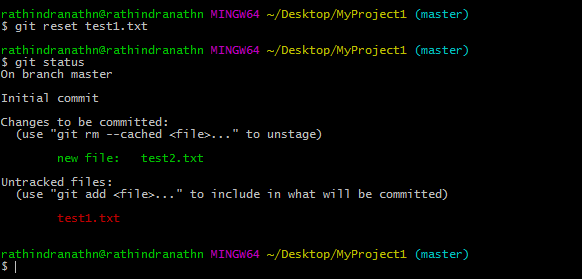
The command “git add –A” pushes all the files from working directory to the staging area. We can also do the same for individual file. After that we can check the status of git tracking by executing the command “git status”. Let’s add our untacked files to staging area. First we shall add only test1 file and then we shall add all the untracked files using –A option. After each “git add ….” Command we shall check the status of git.



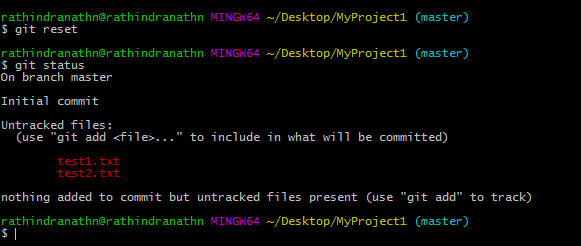
To remove one or more file from staging are we need to execute following command,



Let’s move back test1 file from staging area to our working directory and then check the status of git.

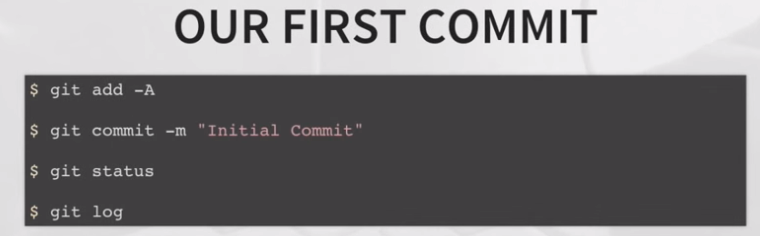


Above command output shows that test1 file moved back to working directory since it is showing untracked, but test2 file is still in staging area. If we want to move everything from staging area to our working directory we don’t need to mention any file name in the command line – therefore “git reset” command is enough to achieve it.

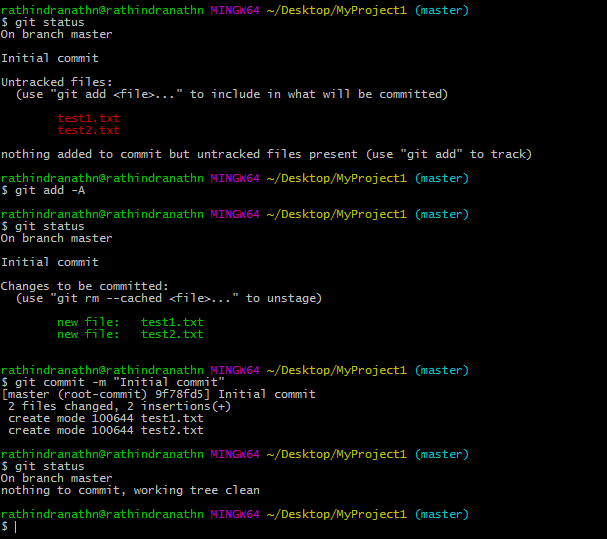


Now both the files file1 and file2 are in working directory. There is no file in the staging area – all have moved back to working directory.

Now we want to commit the files. Following is the command for that,

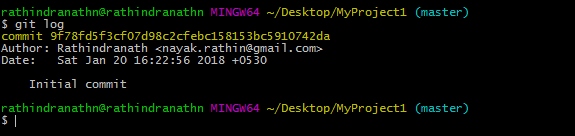


First we shall move everything to staging area and then we shall commit all of them. During commit it is mandatory to provide comment in the command line. Again, commit action of git will commit everything, which is there in the staging area – we can’t commit files selectively.



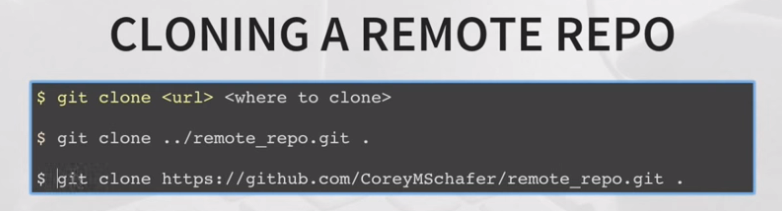
Last “git status” command says that working directory is clean – there is no untracked or modified file in it.

Another important git command is “git log”. It tells about all the commits.



Till now we have committed only once. Each commit bears a hash number, which is unique. Above command also displays the author of the commit, date and time of commit and message during commit.

Let’s discuss about tracking existing remote project using git. In real time work place git is widely used in this scenario. Let say our company has a remote repository. Now we want work on it. In such case first we need to clone the remote repository in our local system and then start working on it. The syntax for cloning a remote repository is as follows,



“where to clone” is the directory in the local system where we want to clone the remote repository. It can be our current working directory, which can be represented by a dot (.). “url” is the complete web path of the remote repository, which is hosted in some website like GitHub.

(Up to 16+ minutes of this video has been described above. Remaining part is not worthy to take note.)

1. **Git Tutorial: Difference between "add -A", "add -u", "add .", and "add \*"**

<https://www.youtube.com/watch?v=tcd4txbTtAY>

There is a git command, which moves only the modified and deleted files from the working directory to the staging area – it does not add any untracked file to staging area. This command is,

git add -u or git add --update

One such example,

