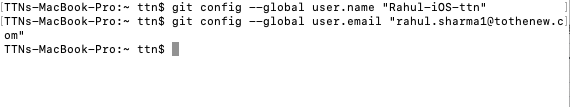
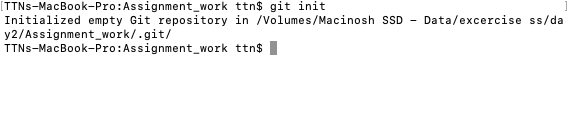
### Git Setup

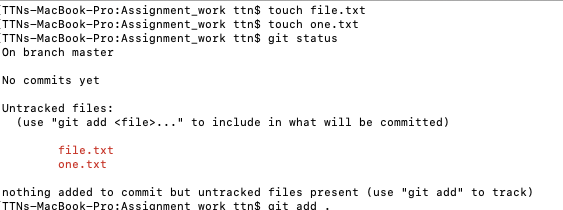
1. It was done using Git for mac Installer.
2. The installation verification was done by git --version command on the prompt.
3. Then after the installation the username and email for authentication was set using the following commands
   1. $ git config --global user.name "Name"
   2. $ git config --global user.email "[name@domain.com](mailto:name@domain.com)"

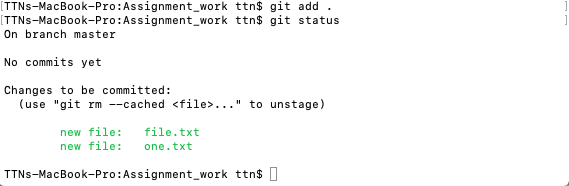
### Initialize a Git repository

This can be done by the command git init in the correct path where one wants to begin a new project.



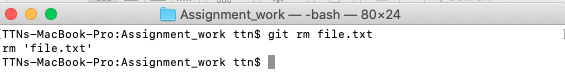
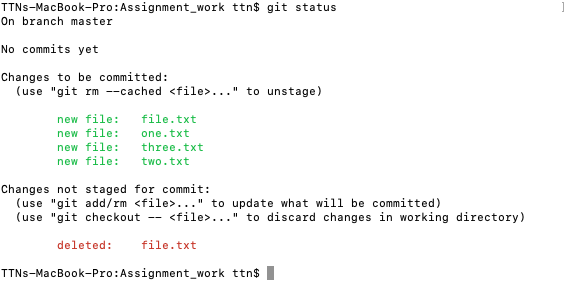
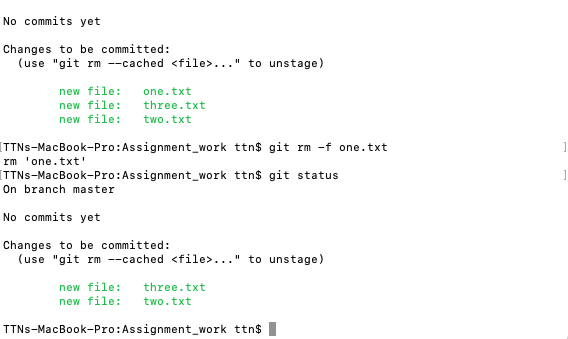
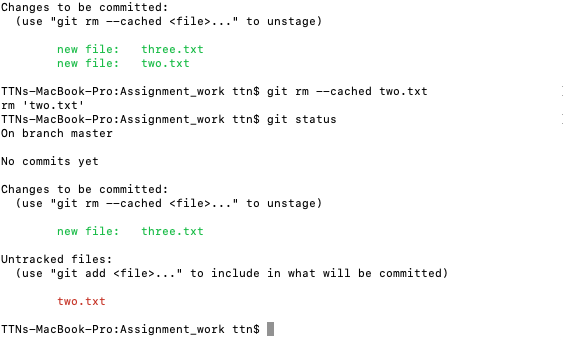
### Add files to the repository(Local)

This can be done by creating new files and using git add command on them.

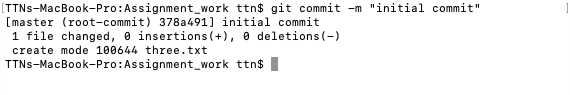


### Unstage one file from the Staging area.

We can do this in 2 ways

1. By deleting the files and then again running the command git add . to again stage everything.
2. By using git rm command to remove the file from unstage area, this inturn will remove the file completely from the staging area and local repository alike.
3. If we don’t want to track a file then we can simply use git rm --cached <file name>

### Commit the files

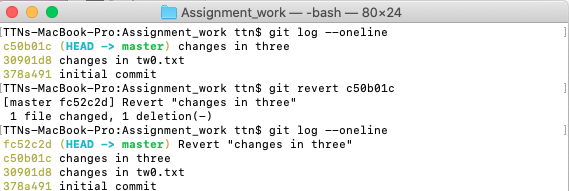
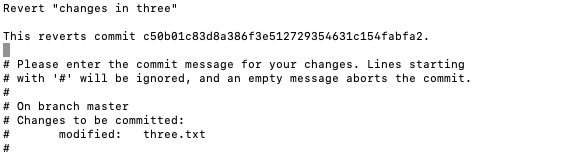
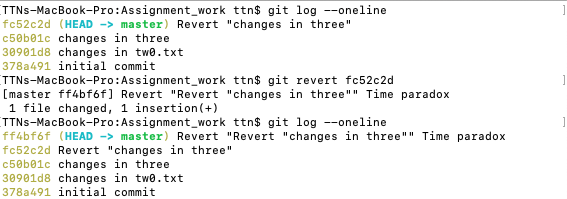
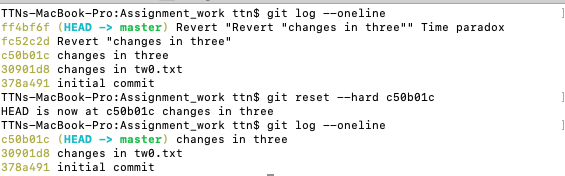
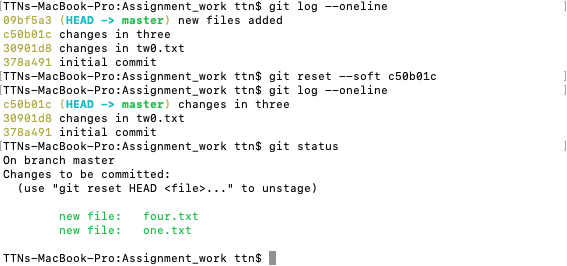
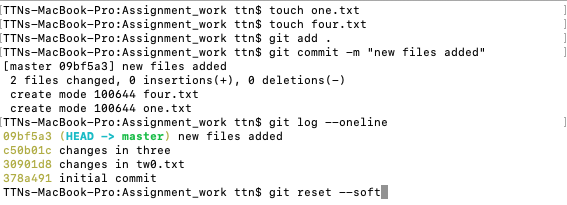
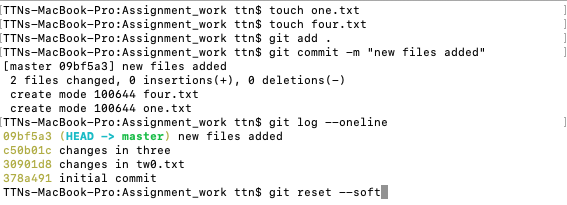


### Add a Remote repository

We do this by first creating an empty repository on GitHub and then adding a remote connection to our local repository.

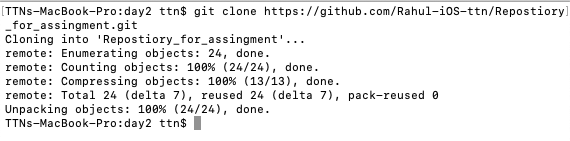
### Undo changes to a particular file

We can do this by using 2 commands

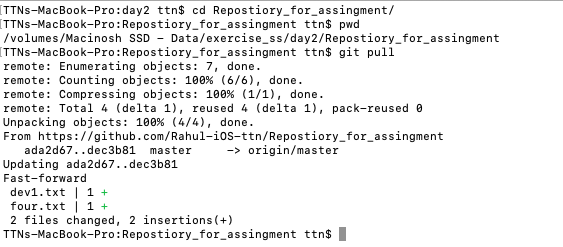
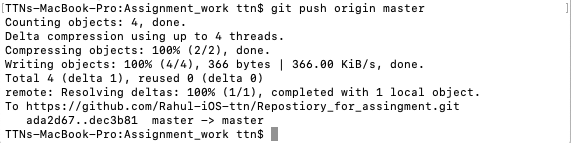
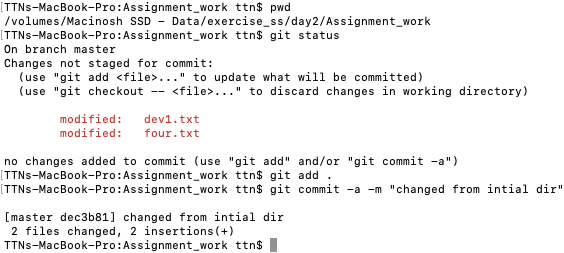
1. Git revert - in this the change gets reverted back by only one commit and it creates a new commit saying that it reverted the previous commit. If we say it in a tree term it is something like, 1->2->3->2.
   1. With this you also have to type a message afterwards it is something similar to the commit message that we usually type.
   2. We can revert a revert change
2. Git Reset - This is a more permanent change in which it just reverts back to the repository state that a user specifies and you can not undo this change, so once this is done it can not be changed, for instance - 1->2->3->4, so after reset to 2 the tree will be 1->2.
   1. This also has types like soft, mixed and hard.The above mentioned tree state is for git hard reset.
   2. Soft reset is more like a checkout command in which we can move our pointer to a desired commit and then work our way through it. Soft commit just removes the commit without changing the environment which means that the files will still be staged just the commit will be changed.
   3. Mixed reset is somewhere in between the hard and soft, it is also the default flag option for reset in git. Now if we do a git reset soft then we are just going back in the previous state while without deleting or removing any files so it's like removing a commit from a remote repository without destroying the changes in the local repository. To state a difference between soft reset and mixed, it will unstage the files.

### Push changes to Github

### Clone the repository

For this we first copy the link of the repository from GitHub and the clone using git clone

### Add changes to one of the copies and pull the changes in the other.

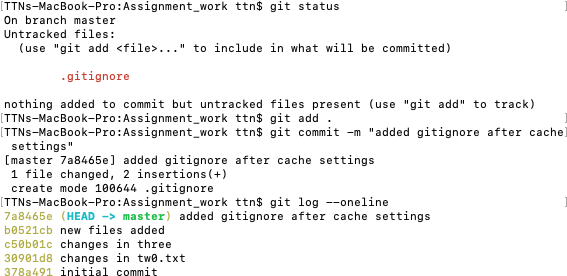
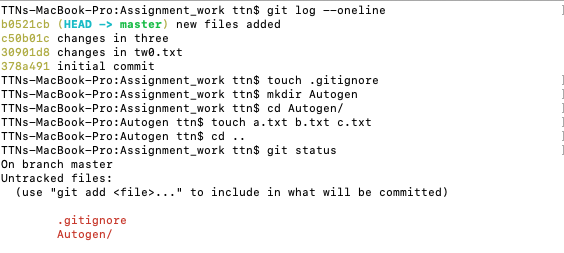


### Check differences between a file and its staged version

### Ignore a few files to be checked in

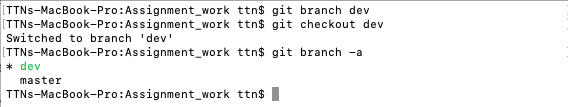
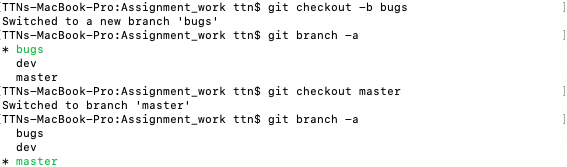
This part can be done by creating a gitignore file.

So here .gitignore is a file which we add into our project as an index from where git can inquire about the files that it needs to be ignored. Those files can be static files or log files etc.

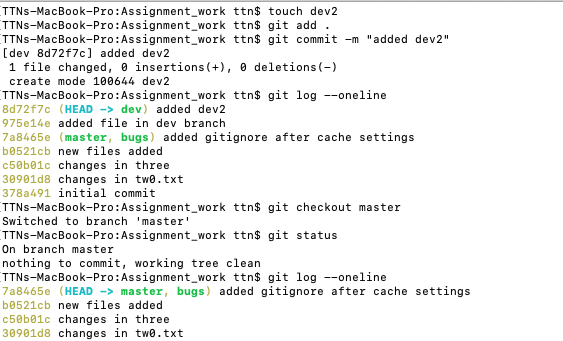
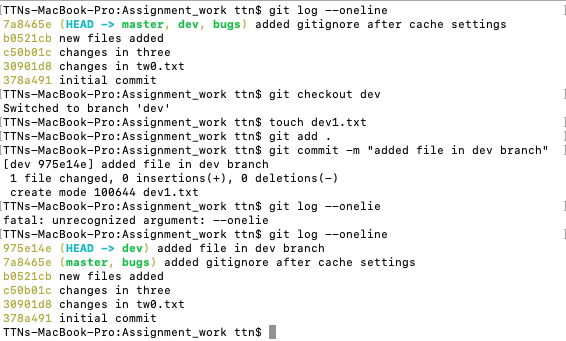
NOTE - Sometime when you add a gitignore in the middle of the project, the git will still track the files this is because the git remembers those files meaning they are still in the git cache which can be fixed by using { git rm -r --cached }

### Create a new branch.

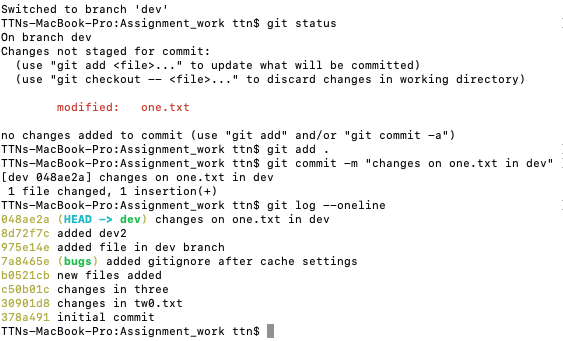
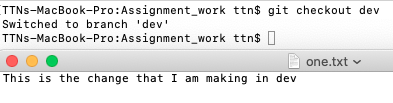
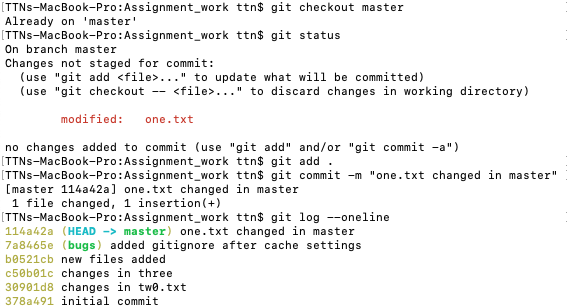
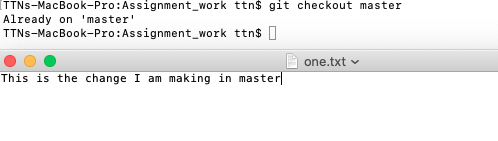
We can do this by creating a new branch in the project. We can do this in two ways

1. We can create a new branch in the repo using git branch <branch name> and then jump to that new branch using checkout.
2. A much easier way to do this is by just using checkout with -b option which creates a new branch and sets it as a working directory altogether.

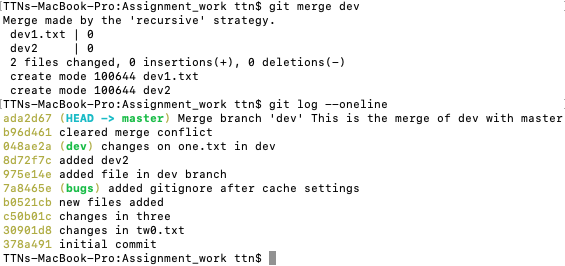
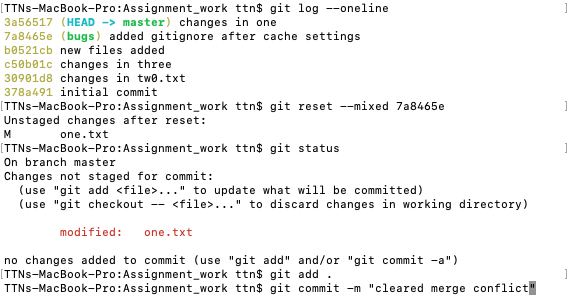
### Diverge them with commits



### Edit the same file at the same line on both branches and commit



### Try merging and resolve merge conflicts



### Stash the changes and pop them

### 