Git basic

Saturday, January 29, 2022

15:25

To start from empty folder .

Install git first .

Launch the git

Go to desired directory

Run below command

#git config --global user.name <name>

#git config --global user.email <e-mail id >

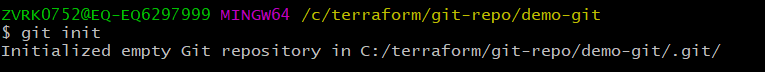
To check the Author run below command

#git config --global user.name

After that initialize a folder

#git init

Note : git clone path (it also make your local dir as git)



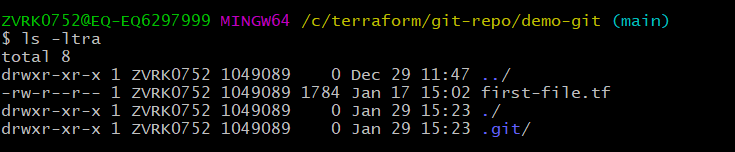
After initialization you noticed you are in a virtual space (main)



Create any file inside the folder

# vim first-file.tf

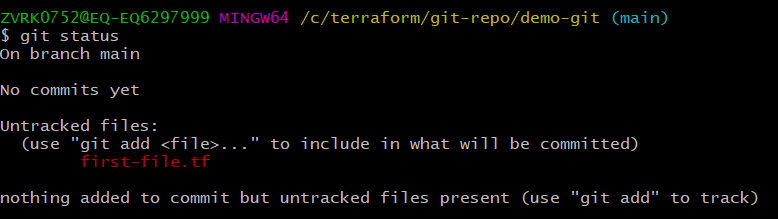
# ls -ltra



Init command create a .git hidden file in your folder.

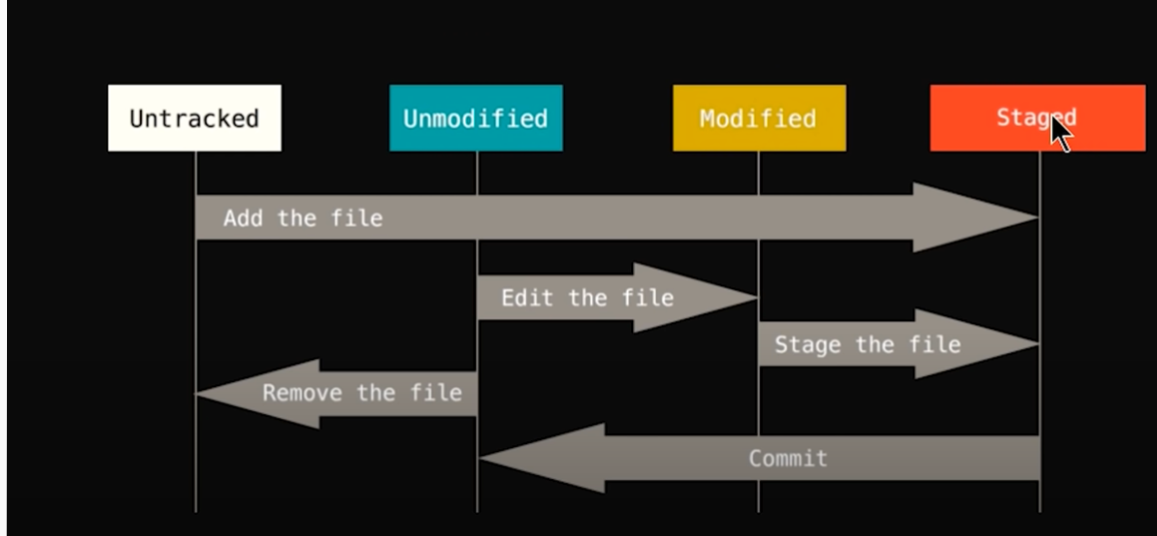
Now run the command

# git status



You see above your file is untracked , what that means ???

Here in git some terminology are used



Untracked : means git is not managing this file. This file is not stored in git repo and any changes in this file is not tracked

Staged : is an area which contain all file which needs to commit

Unmodified file : after commit file status changed from staged to unmodified

Modified file : once you make any update in tracked file it shows as Modified and needs to stage

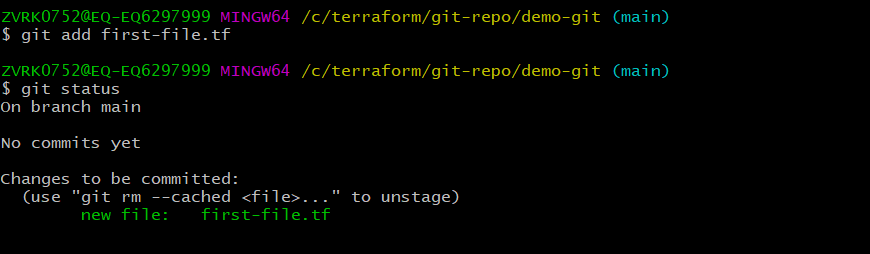
Q: What happen when we make change/modify the file which is in staged area currently and not committed yet?

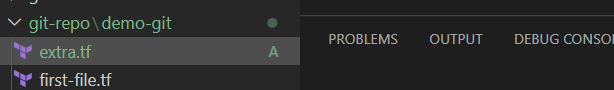
Ans : we have to add that file again to stage area with command #git add <file name> (Will discuss more below)

Now back to our steps :

Need to add our first-file.tf to stage area, to do that we need to add the file

# git add first-file.tf





After add file status will change to A in visual code (index Added)

Now we have to do the commit

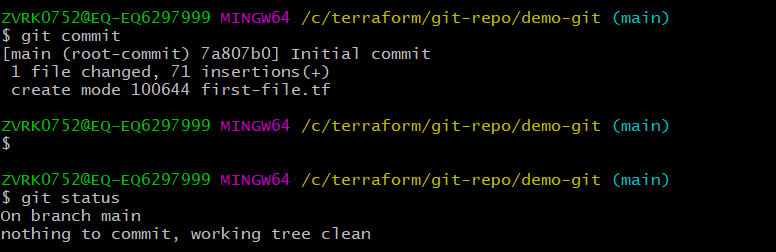
# git commit

It will open a nano/vim editor to provide the msg for commit after that you will see below

OR

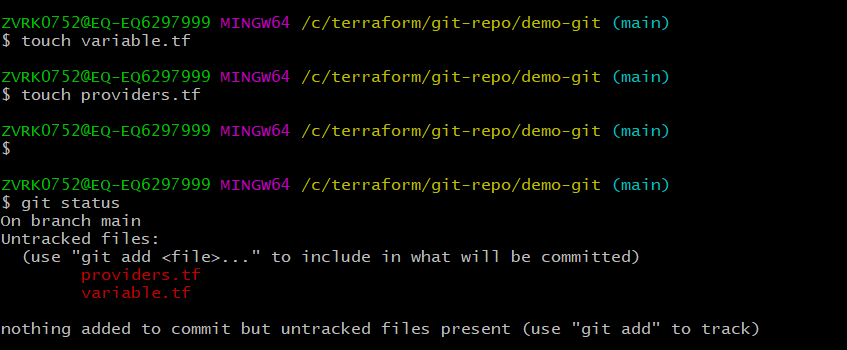
Here you can also use bellow command

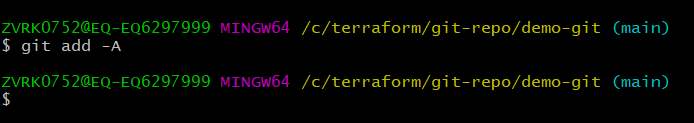
# git commit -m "message for commit" it will not open any editor for provide a message



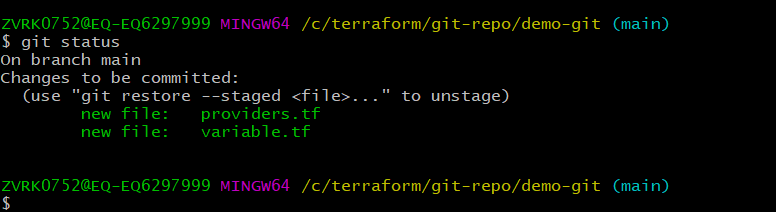
To add multiple file in git, use bellow command

# git add -A (it will add all untracked files to stage area)



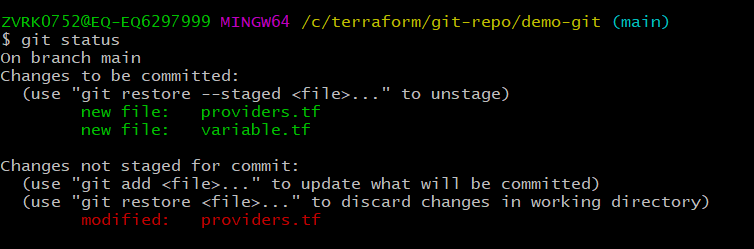


Here both new files are in stage area but waiting for initial commit

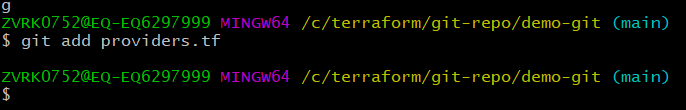


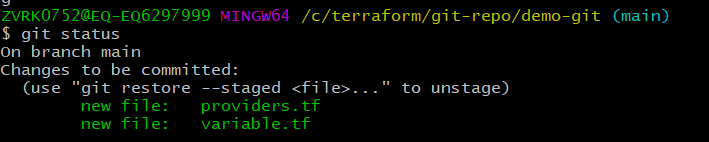
Now I have modified one file called provider.tf with some new code

Again checking the status of git , it says two files in stage area and one file providers.tf has been modified

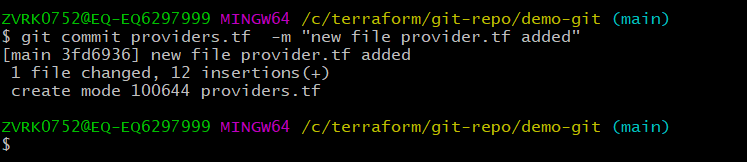


Now added the modified file in stage area

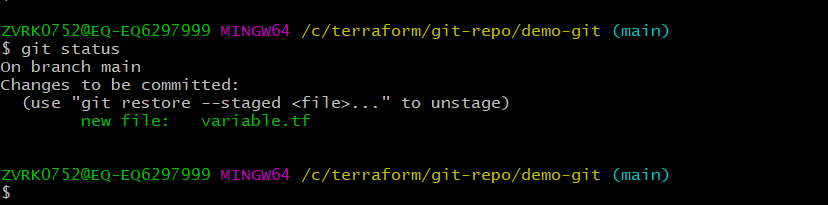




Here we committed only one file providers.tf , with -m option to avoid editor opening step



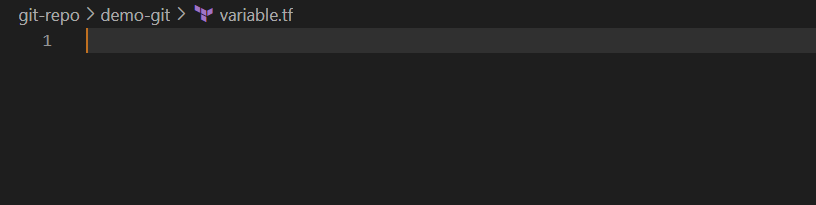
Now only one file in staged area



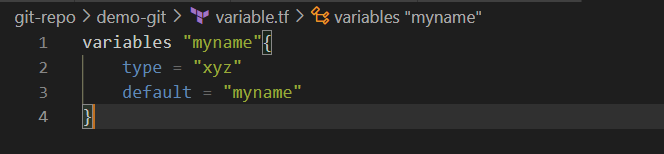
IMP : if someone make modification in your code , and now you want to go back to your original code , what you will do ?

Lets see

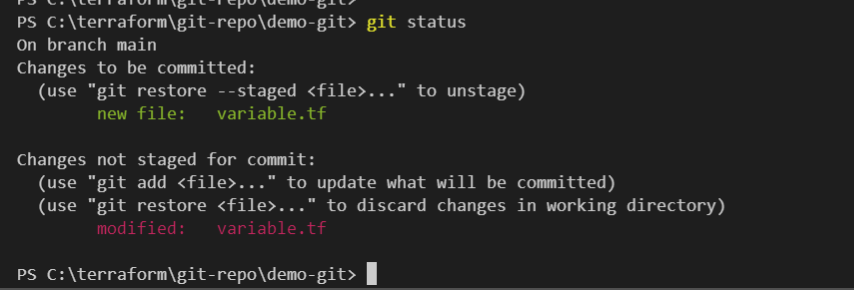
We have variable.tf file in stage area which is not committed yet and this file is empty as of now



Now some one change it with wrong code



You run the git status



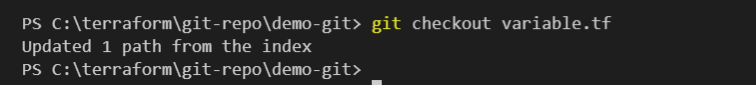
Now want to go back to empty file : you have two options either rewrite your code (in our case delete whatever we have in variable.tf ) OR move back to previous status.

Here one issue if we rewrite the code we still in modified stage as git say you modified the file .

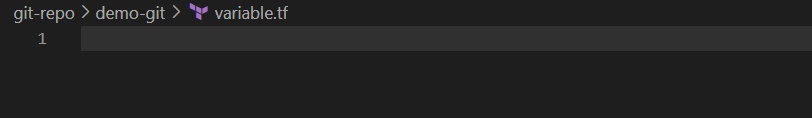
But with the help of git checkout you can easily go back to previous stage and git will not pop-up for modified status

To do that run command

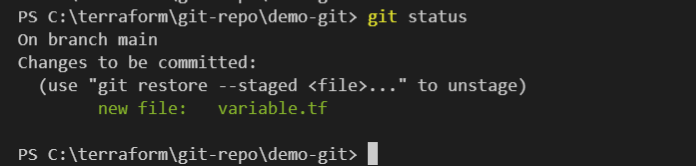
# git checkout <corrupted file name>



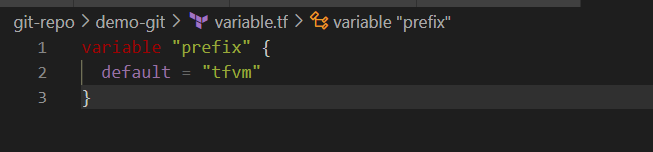
once you run this file will go back to previous stage automatically

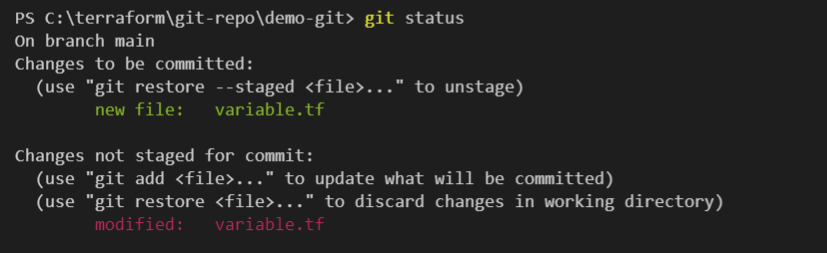


We can check this with git status command as well

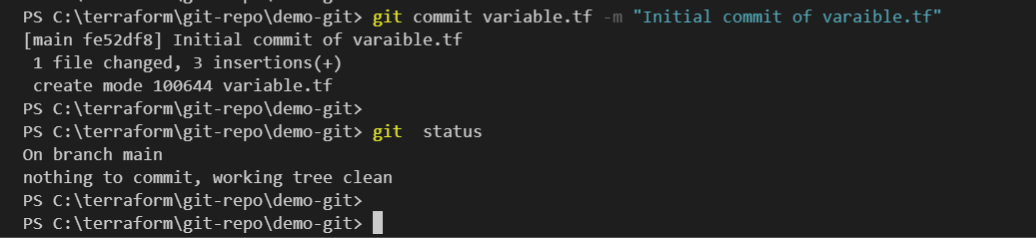


Now we can re-write the correct code in our file and commit it





And now we can commit the variable.tf file

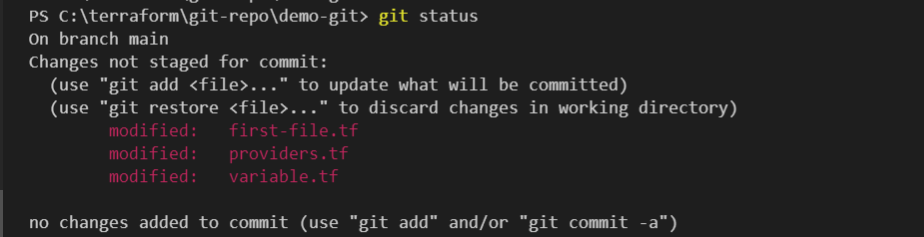


Now all the three files are committed and we are in unmodified stage

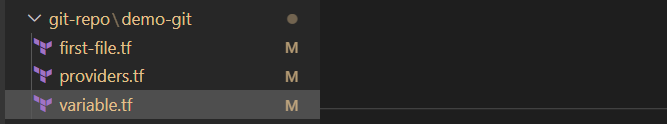
There is one command option for git checkout if you have more than 1 file which have been modified and you want to come to previous stage

Lets see

Someone by mistake modified your files

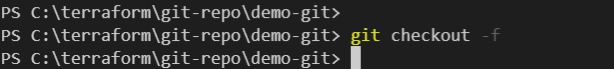


In Visual code it also shows as modified



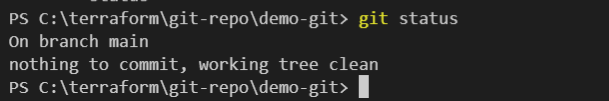
Now you want to go back to previous stage for all files

# git checkout -f

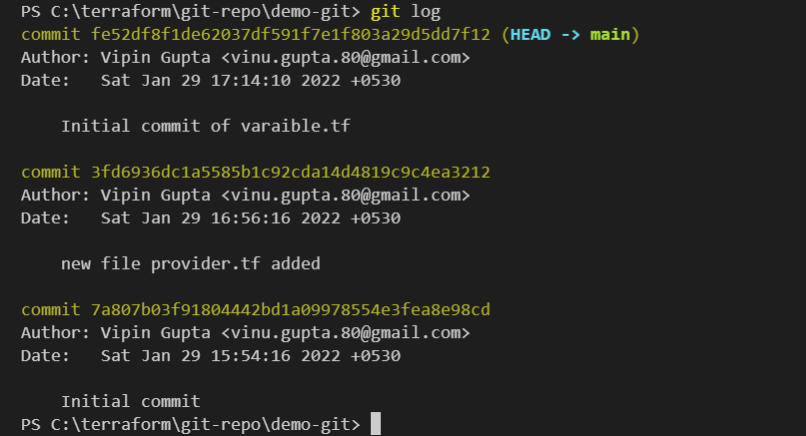


Once you run checkout -f command all file are back to last stage

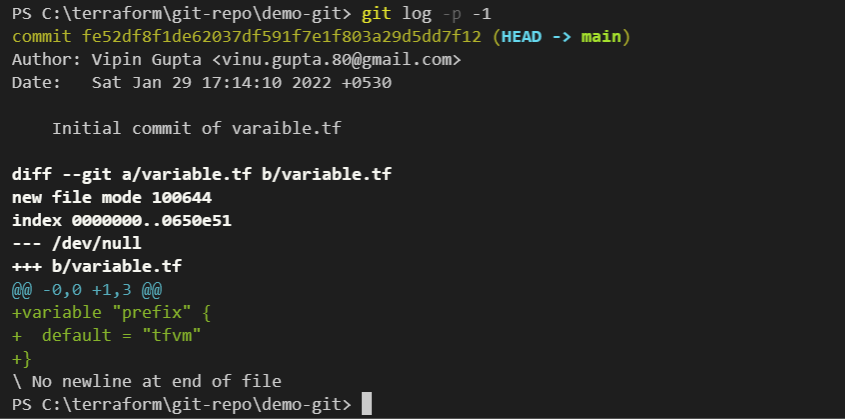




Git LOG command tells you what you did in your git repo, you can filter your log with command git log -p - <number> how many logs you want to see

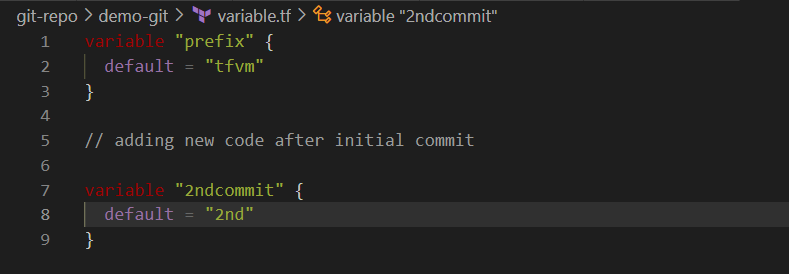


#git log -p -1 [this command will also tell you what modification you did]

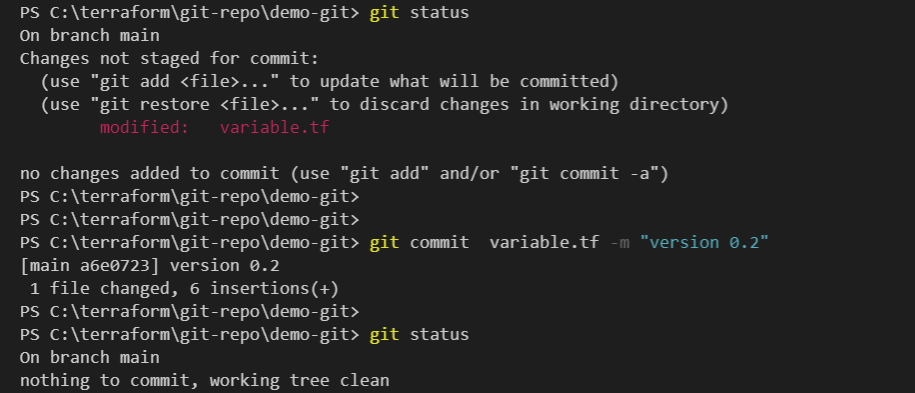


Lets see in more details , making change in variable.tf file

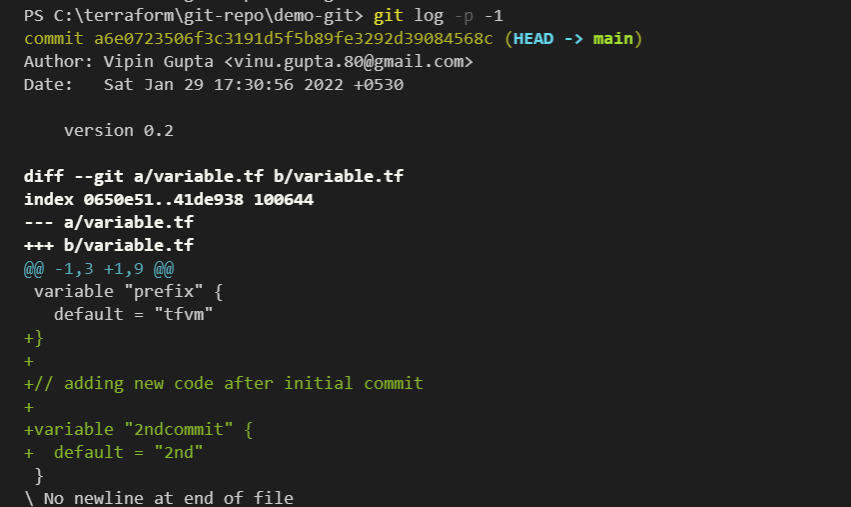
Made changes as below

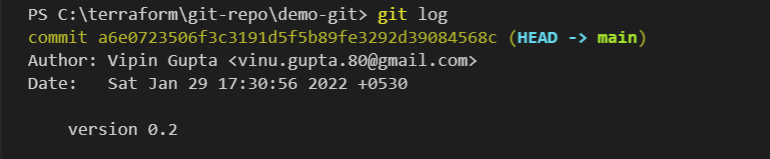


And done commit as below



Now will see the logs for variable.tf



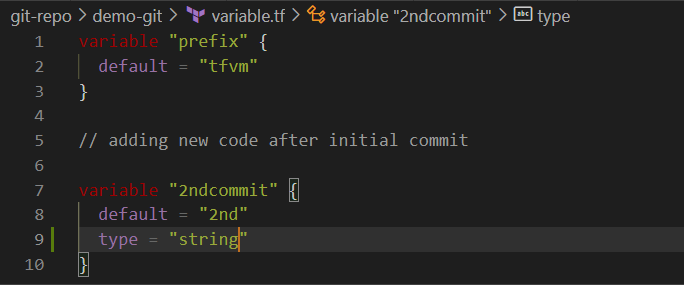


Another good command is git diff , it will tell you what has been modified in your file , this will compare your working directory to staging area

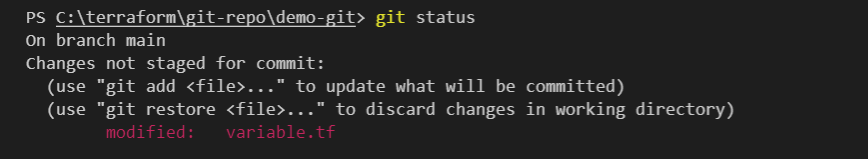
Git diff --staged will compare with last commit with staging area

Lets see this using file variable.tf

Modified file



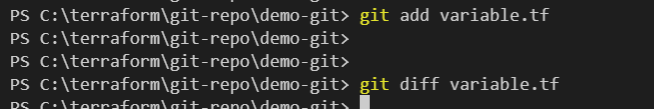
Git diff command will tell you difference of current directory and stage area





Now adding this change from current directory to stage area with command # git add <file name>

After adding a file to stage area it show no difference



Now the git status is in color green

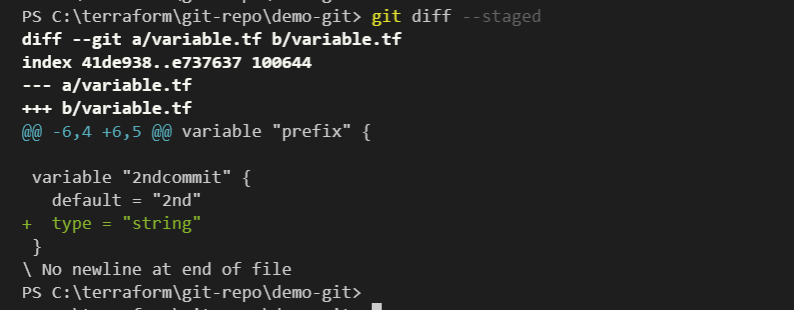


Note : difference in modified color red or green

RED : modified content is still in working directory

GREEN : modified content has been staged with git add command , but not committed

Now will see the command git diff --staged, as we mentioned this will show the difference of stage area and last commit



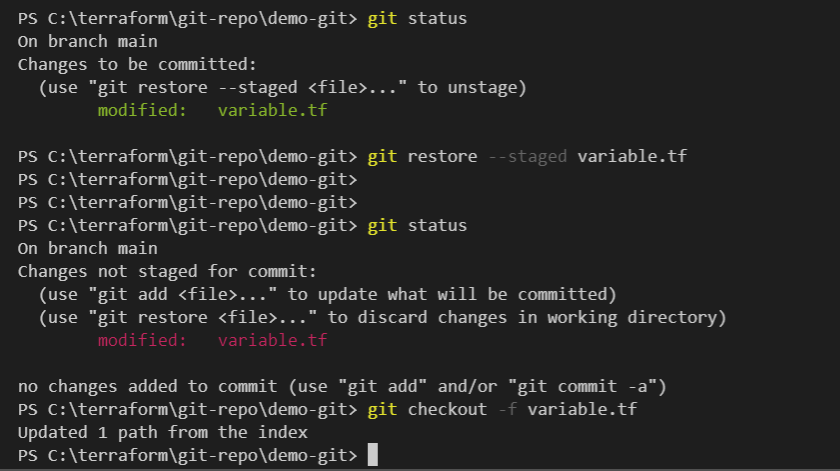
Now we want to go back to last committed stage , we have to use two commands

Below command will move the file from stage to working directory

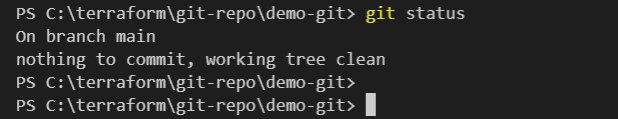
# git restore --staged variable.tf

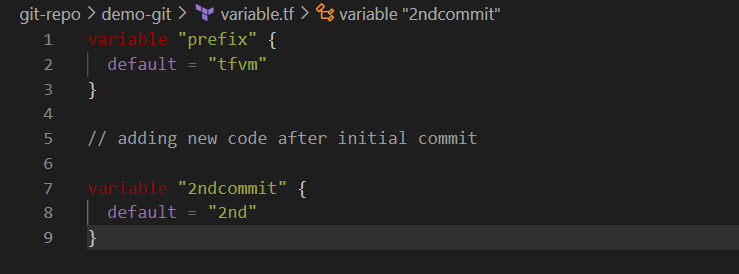
Below command will revert the changes and matched the file with last staged area

# git checkout -f variable.tf



Now see the status with git status



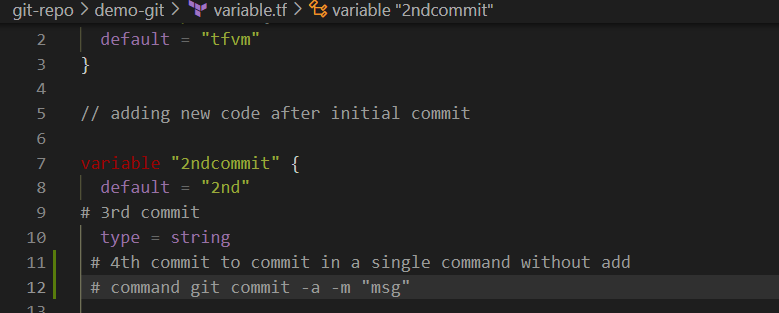


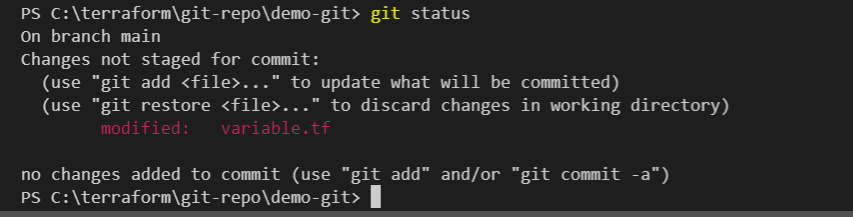
File back to last commit status

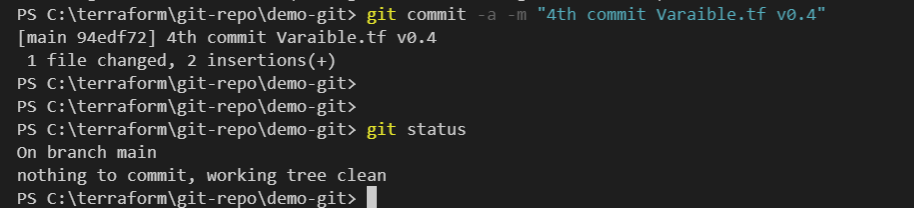
To test git commit -a -m "msg" command

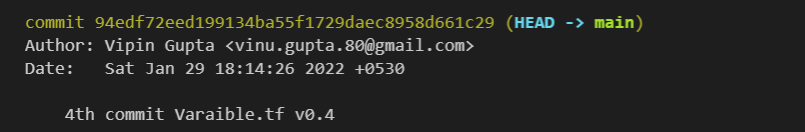
This command skipped the git add command

Changed the variable file









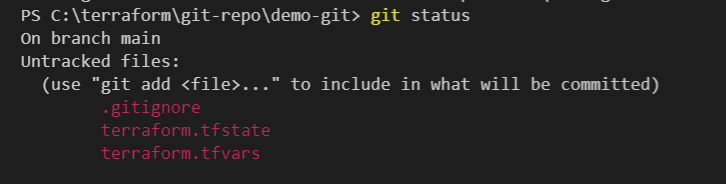
Testing feature of git

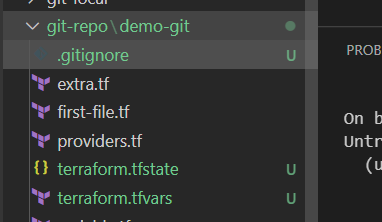
Use of .gitignore feature

We need to add a file .gitignore in our current working directory to ignore some of our files which we don’t want to push/pull/commit in gut hub.

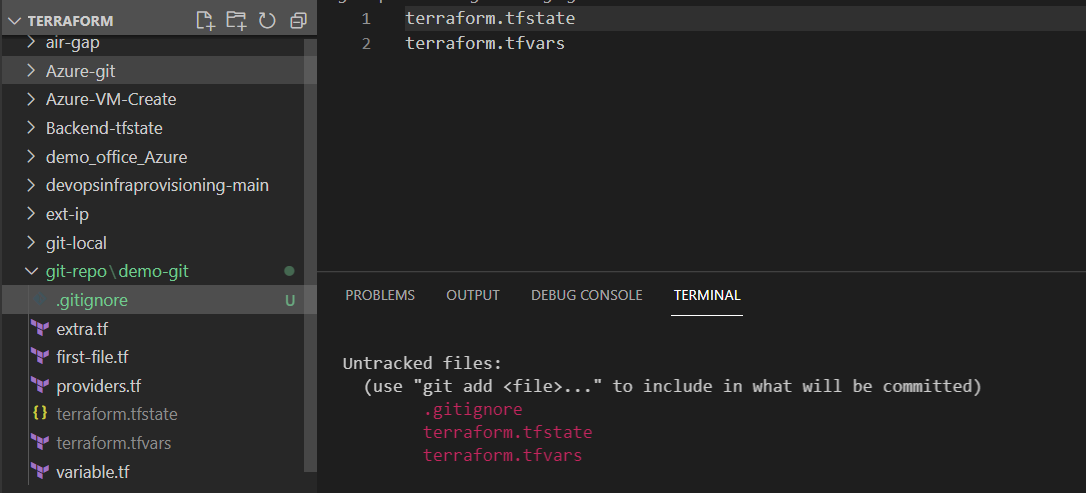
Let see how to do this

Created three file and its shows as untracked





Now add the file names in .gitignore file , as you add the name of file and save the .gitignore file , files removed the U status



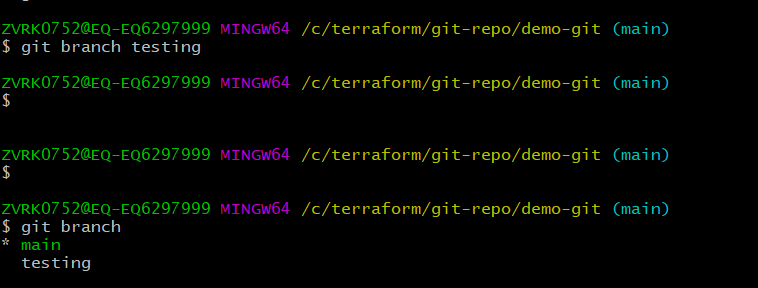
GIT branches

To create a branch run the command

This will only create a branch

# git branch <name of branch>

# git switch <name of branch> (to switch from one branch to another)

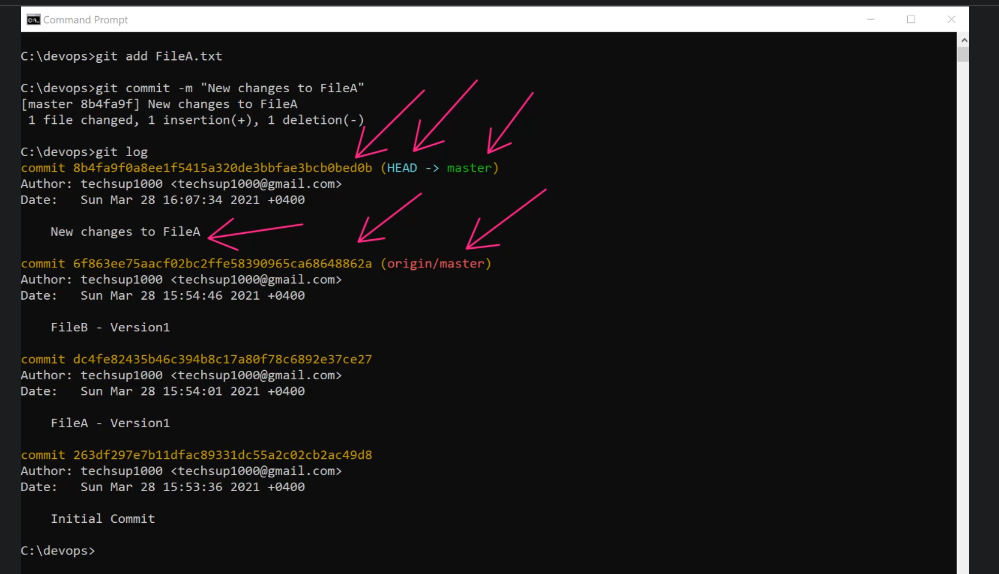


To create a branch and move to new branch in one command

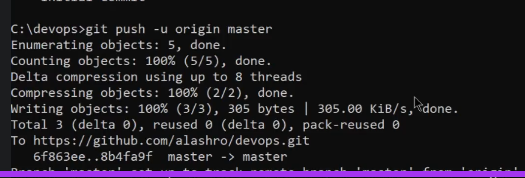
#git checkout -b <name of branch>

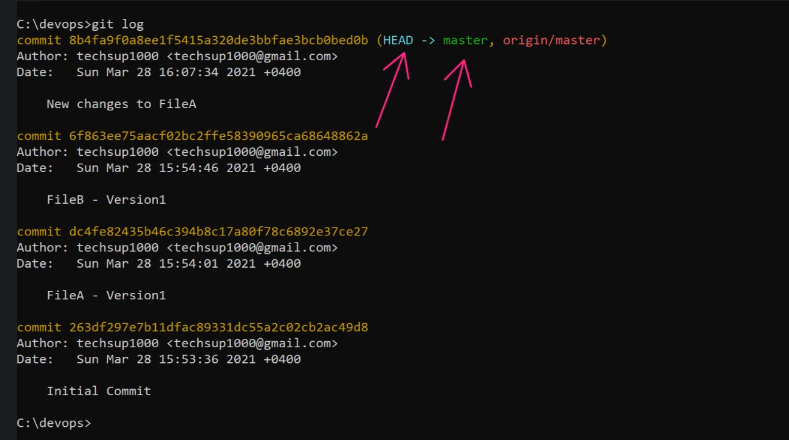


Here the output show that my local git is updated with the git repo on remote



Run the git push command to make it one





**How to do pull and push with GitHub**

Git Push is use to Push the code from local repository to Git Hub

Git pull is use to pull the code from Github to local repository

To do that you should have a git account

And repository in git : Always create a private repository for security reason

To access the repository from local system use the below commands to add ssh-pub key in git hub.

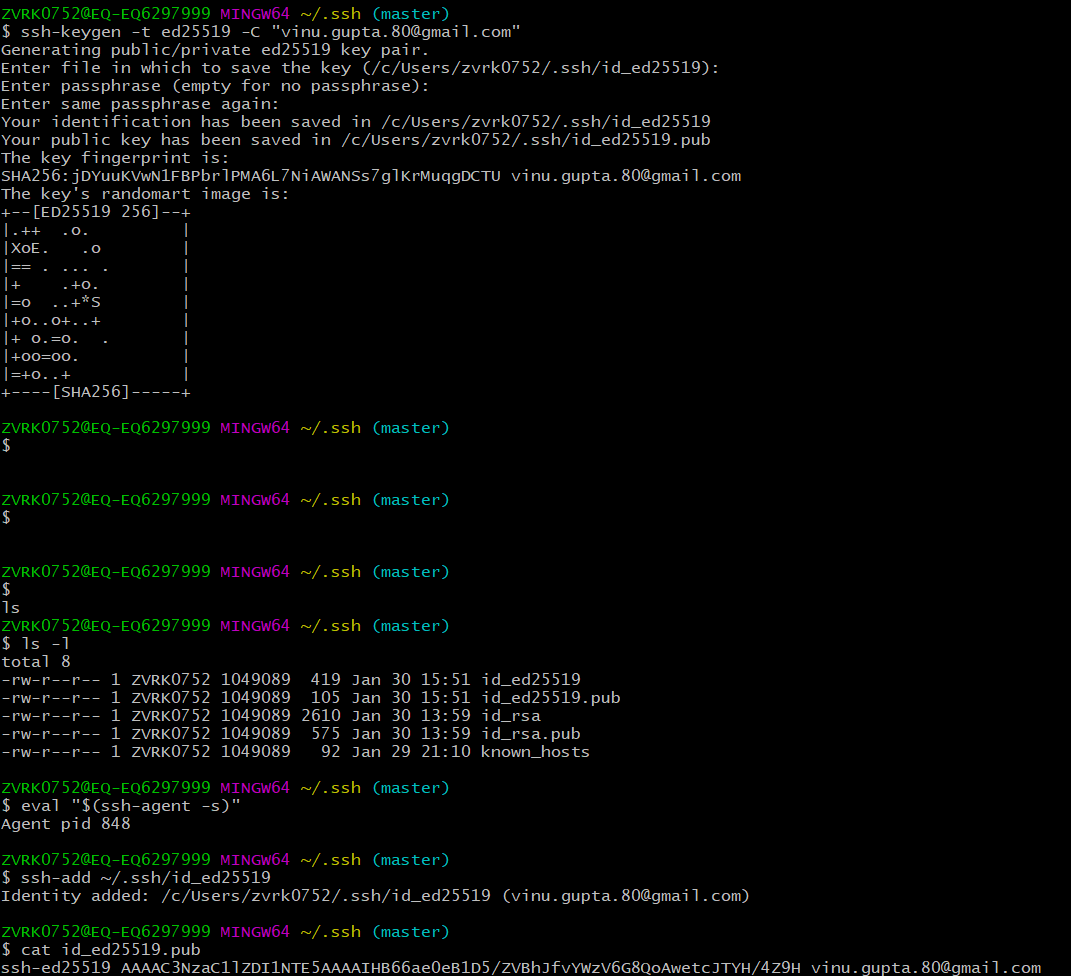
You can find these command in Git , how to access private repository

# ssh-keygen -t ed25519 -C "vinu.gupta.80@gmail.com"

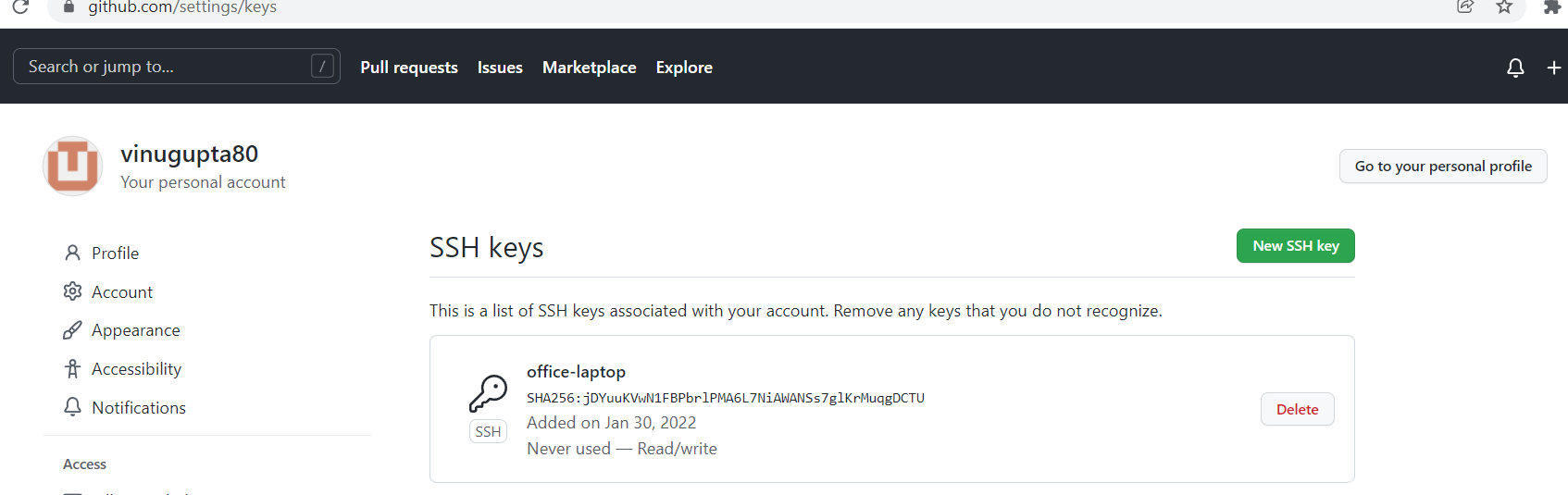
# eval "$(ssh-agent -s)"

# ssh-add ~/.ssh/id\_ed25519

#cat id\_ed25519.pub



Copy the pub id and it to the SSH key in Github setting



After this select the repository in git hub where you want to push the codes.

Add the git SSH url to git bash as origin with the below command

This command to add git repo as remote , local name is origin for this remote.

#l



command to change the path of remote repository

# git remote set-url origin <path of repository >

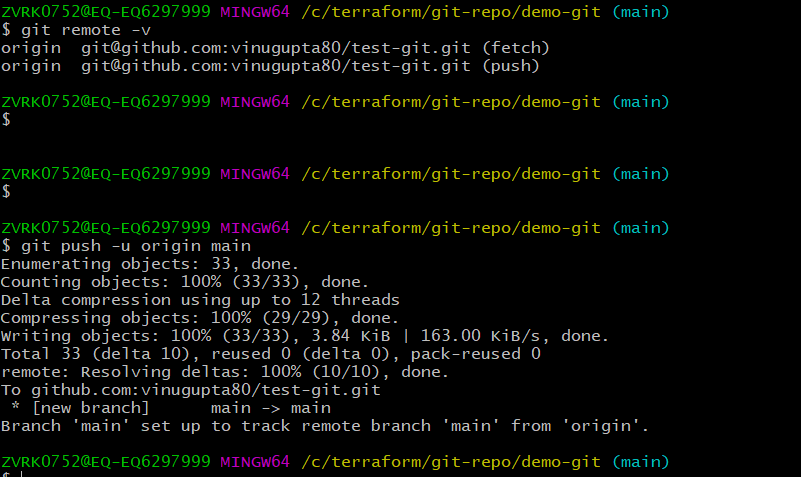


Check the git remote

#git remote

Run the below command to push the codes from local repository to remote

#git push -u origin main



/\* command at new repository

…or create a new repository on the command line

echo "# test-git" >> README.md

git init

git add README.md

git commit -m "first commit"

git branch -M main

git remote add origin git@github.com:vinugupta80/test-git.git

git push -u origin main

…or push an existing repository from the command line

git remote add origin git@github.com:vinugupta80/test-git.git

git branch -M main

git push -u origin main

\*/

NOTE :

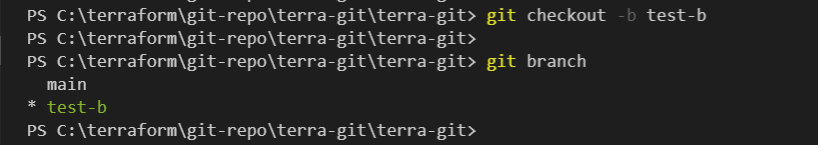
If we have empty folder and we do git clone , it become local repository and no need to do git init

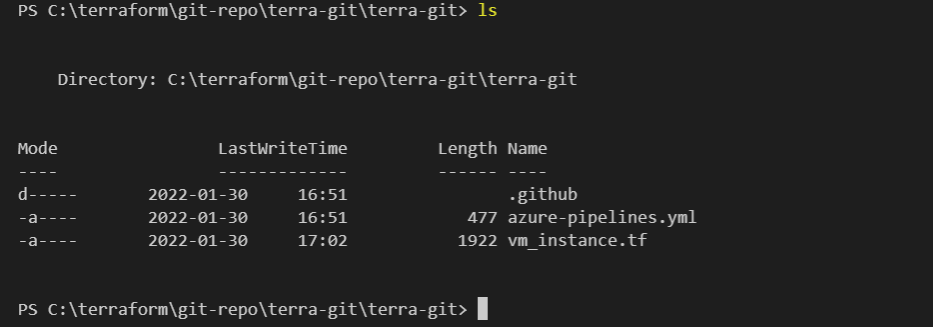
If we have a local repository and remote repository and both contain some code file. How to do push and pull ( do we need to first merge both repository)

How version manage in git local?

How to use old commit data?

How to do push and pull with another newly created branch

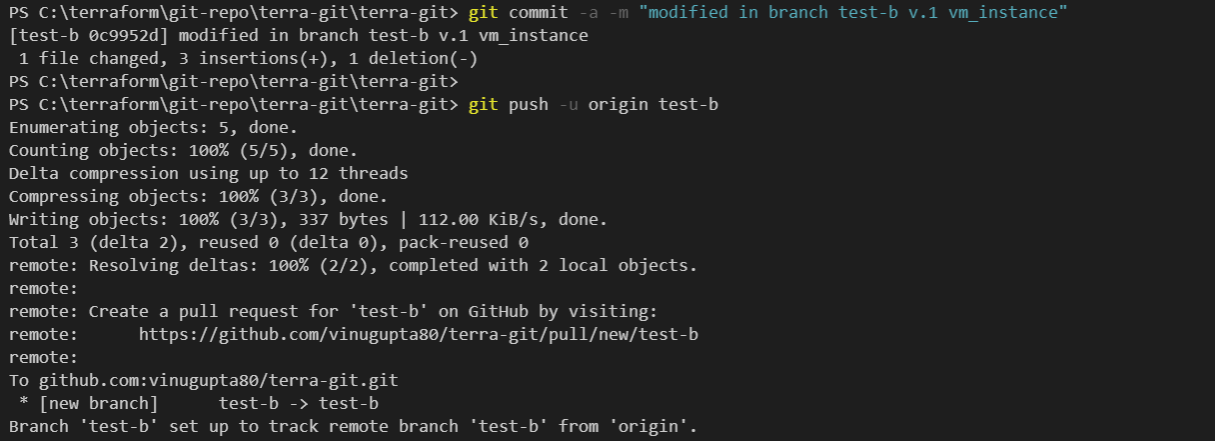




Make a some change in any code file to see the result of changing in test-b branch

To push in a branch run command

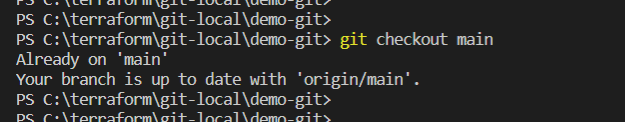
#git push -u origin test-b



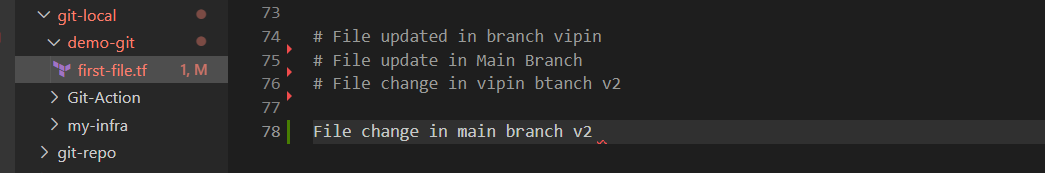
Way to resolve conflict when file updated in both branches.

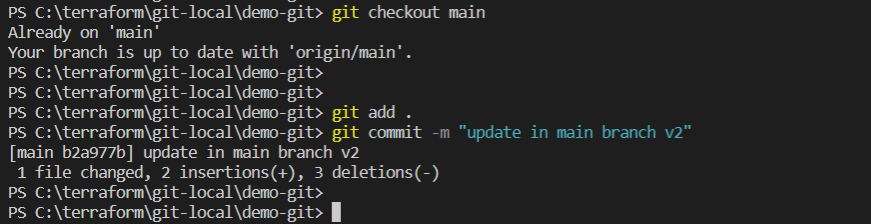
Demo :

Switch to main branch

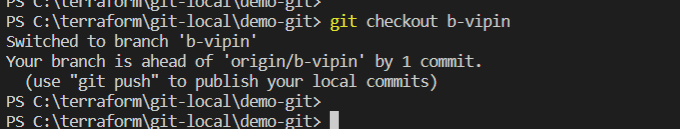


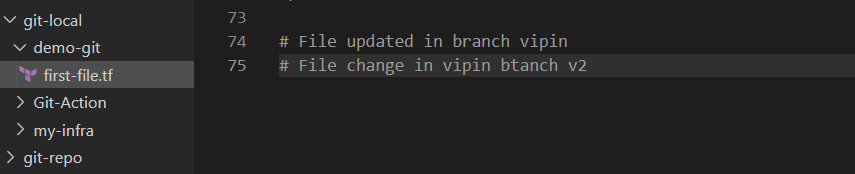
Update the file in main branch



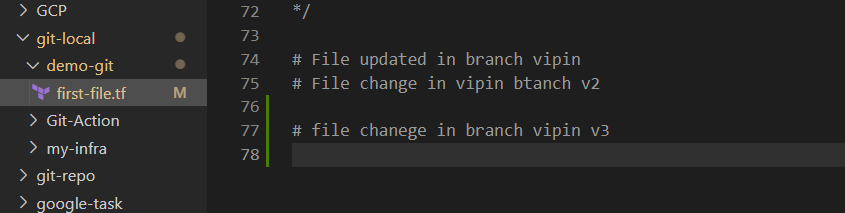


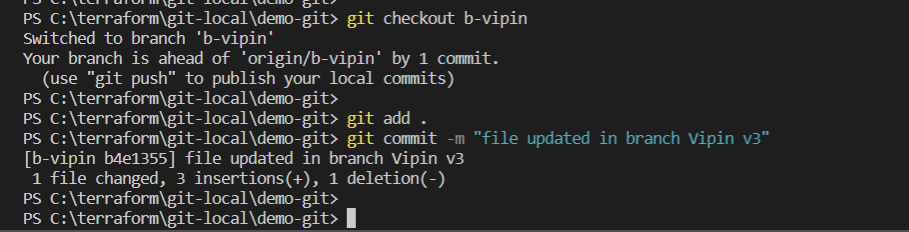
Switch to another branch





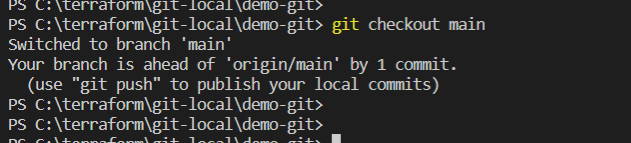
Updating the file



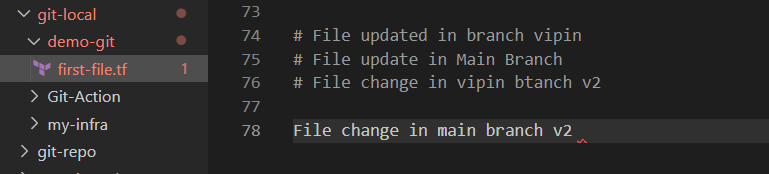


Now merging b-vipin branch to main branch

Switch back to main branch

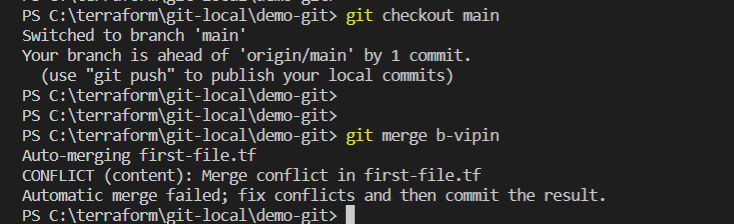


File status

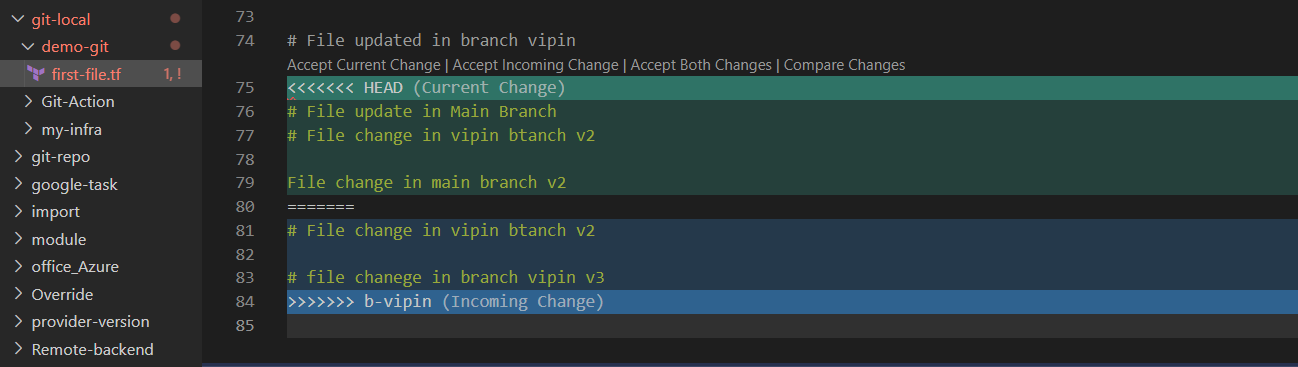


Run the command

#git merge <name of 2nd branch>



Once you run this command it ask in file to resolve the conflict

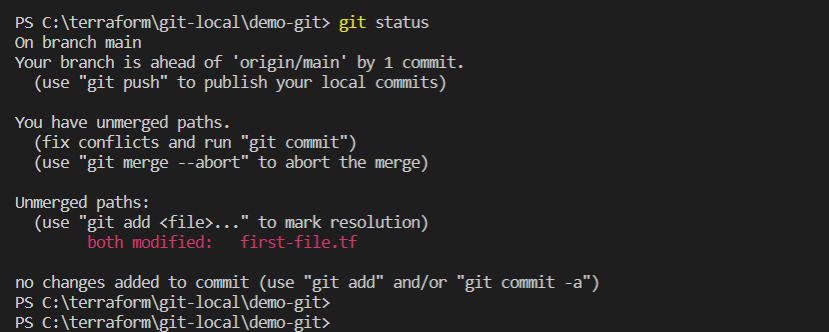


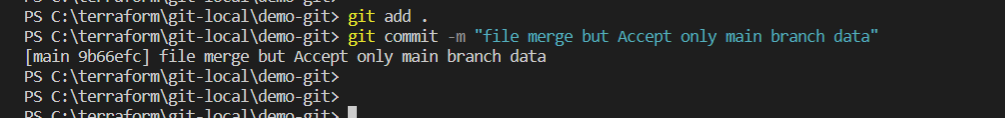
Here you have 3 options Accept the main file changes , Accept <2nd branch changes> or Accept both branches changes

Click on any one of the option

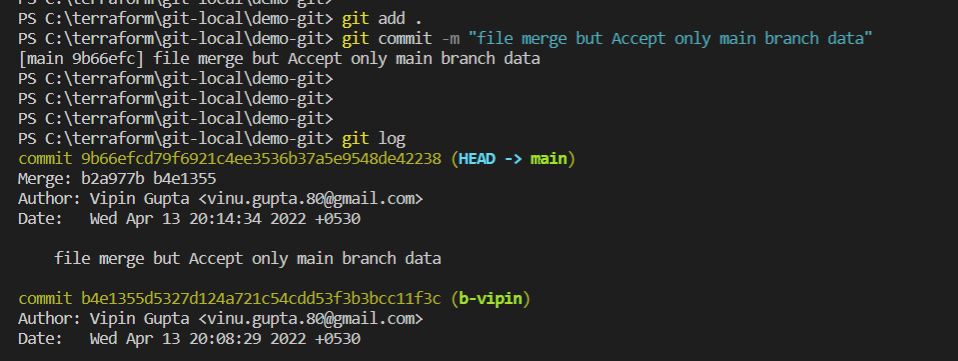
Here we choose Accept current changes , in this case it will discard the changes of 2nd branch

As of now file is modified again

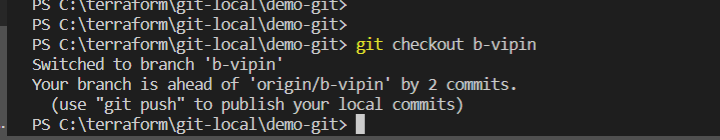




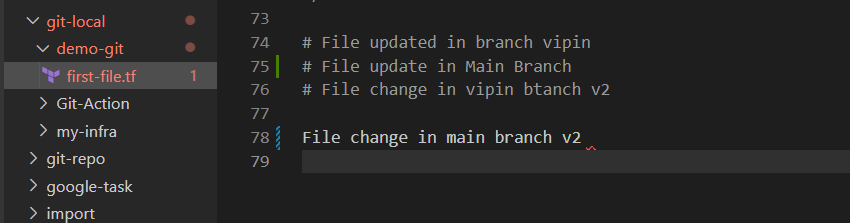
In git log it shows the file is merge



Now checking what we have in 2nd branch



We have same data which we have in main branch



Another useful merge command

#git merge --squash <name of 2nd branch>

What it do : it will merge with main branch but not the history of commit of 2nd branch , only merge commit will be visible at main branch

PULL REQUEST at GIT/AZ REPOS

Azure Board and GIT integration

With this integration you can complete the AZ board task , with the commit of Git … there is some special words which understand by AZ board and it close the task , once there is change in integrated git repository with those special words. Like finished



