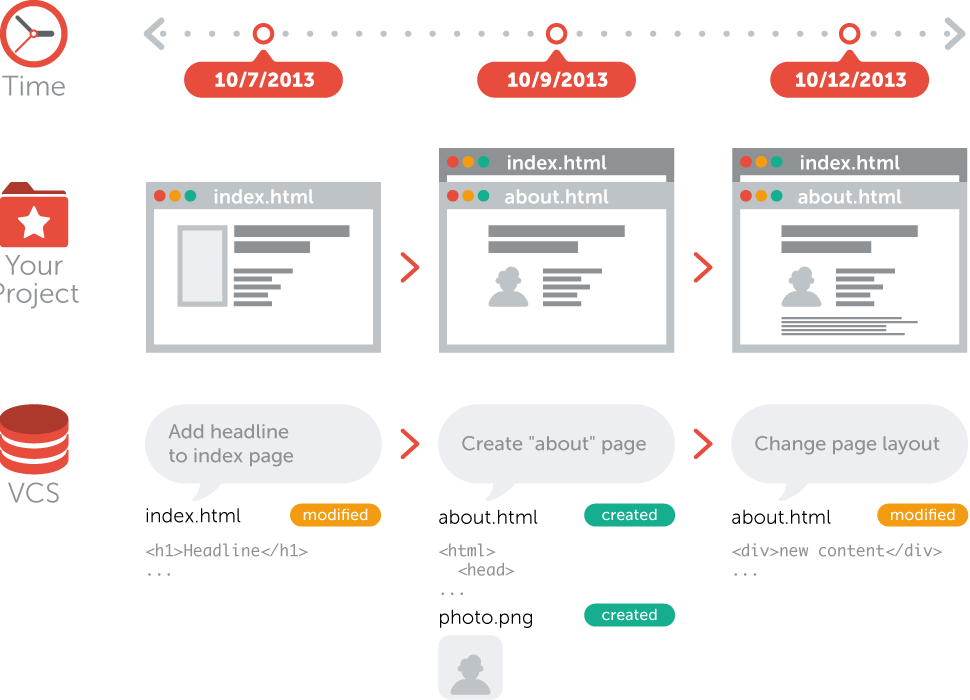
**GIT and GITHUB**

**What is version control system? :**

A version control system is software , that manages changes to files

eg: code files , documents , images .....

* The changes are usually called as "versions"



Different version control systems tools :

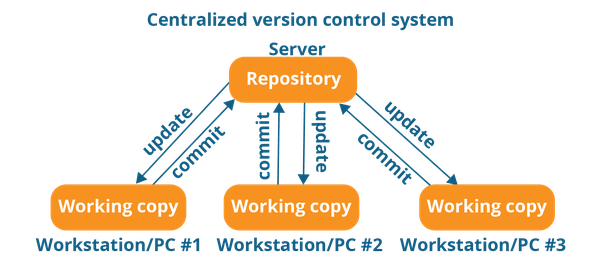
1)git

2)svn

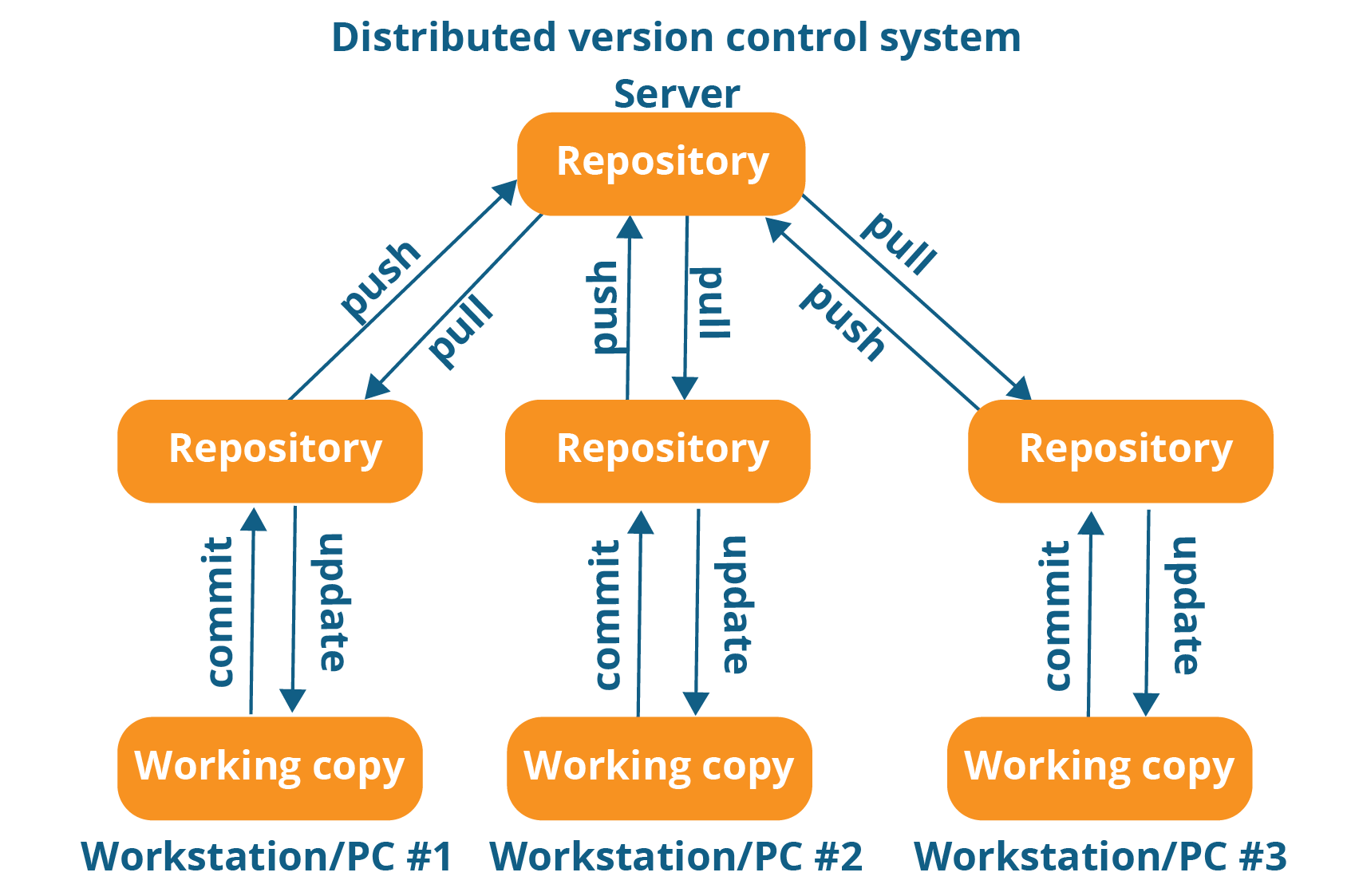
3)Mercurial

|  |  |
| --- | --- |
| SVN | Git |
| open source | Open source |
| Follows client-server hierarchy | Follows Distributed hierarchy |
| No local repository | Local repository available |
| If central server crashes, there is no back up of source code | If central server crashes , everyone having their local copy of source code |
| Always have to connect online ,internet connection is needed all the time | We can commit changes to local repo , and whenever internet available push the changes to Github |

**Typical centralized version control system structure :**



**Typical Distributed version control system structure :**



**Repository :** Repository will store all kind of files ,images related to your project

two types of repositories :

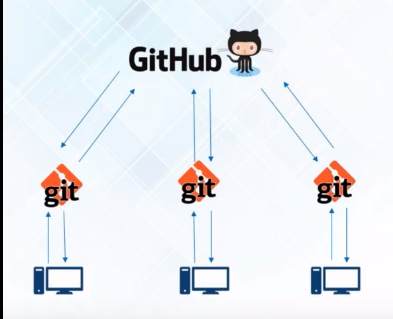
1) local repository

2) central repository

* In distributed version control system all the developers will commit their changes to local repository (git) and later they will push the changes to central repository (github).
* To retrieve all files from central repo , we use pull

**Git and Github:**

|  |  |
| --- | --- |
| Git | GitHub |
| Local Repository | Central repository |
| Version control s/w | Hosting platform for git (Remote server) |
|  | It is social networking platform for developers , who can share the code to others |

****

**Git :**

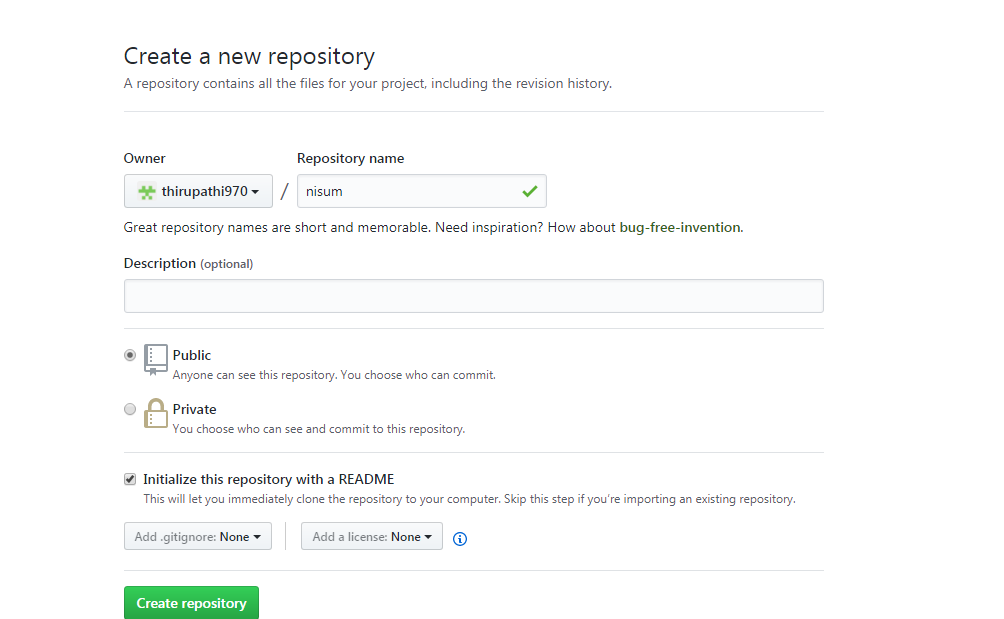
* Version control software
* free and Open source
* distributed

**GitHUB:**

* Collaboration software, hosts GIT repositories
* Repository to store the files/projects/code

**Working with github :**

Go to https://github.com , create an account and create a new repository



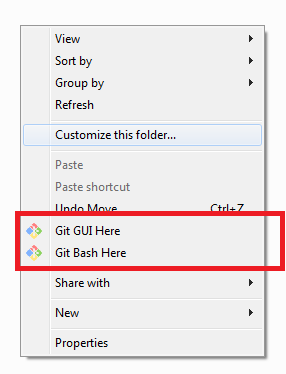
**Working with Git :**

Now we have to create local repository , for that we have to download git

go to https://git-scm.com/download/win , and download 32 or 64-bit as per compatibility.

Install git s/w on your windows machine

after installation right -click then u can see git bash as shown in below pic



go to c:/ drive and create a folder , say git > open git folder , do right click ,click on Git Bash here. it will pop-up a terminal .

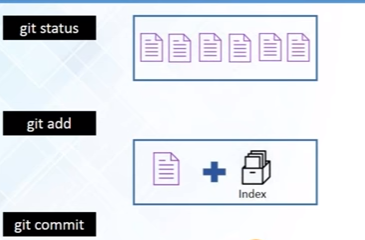
intialize git----> $ git init

sync your local repo (git) to central repo (github)

$git remote add origin "link of git repo"

pull the files from github

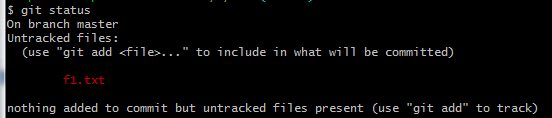
$ git pull origin master



There is a layer called Index , this acting as a intermediate layer to workspace and local repo .

if you do any modifications on files or adding new files , first we have to add to the index , then commit changes to local repo .

Add one file f1.txt in your workspace , and hit $ git status

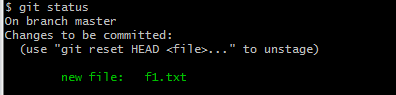


add this file to index

$ git add f1.txt

the file has added to index , check with

$ git status



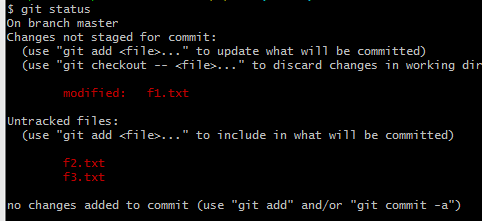
now lets commit our newly created file to local repo

$ git commit -m "any message regarding commit"



lets create multiple files f2,f3 and do modification in f1 , check git status

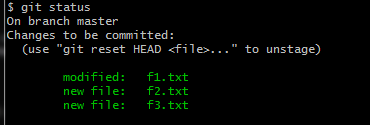
$ git status



adding multiple file to index

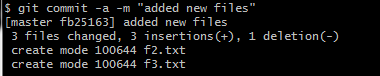
$ git add -A

check status

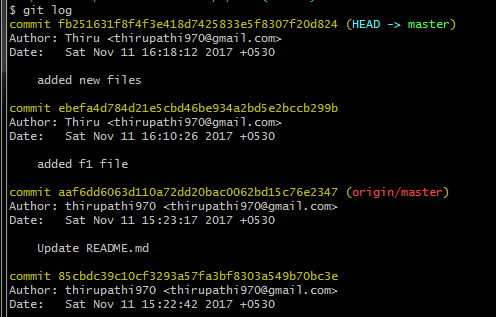


commit all files in a single go

$ git commit -a -m "message"



$ git log to check logs



**Branching:**

Create a branch on master replica, as the changes should not be done on master

Master should always has to be deployable

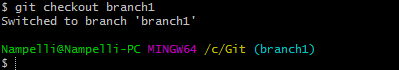
Branch is exact copy of master.

create a new branch

$ git branch branch1

enter into branch1

$ git checkout branch1

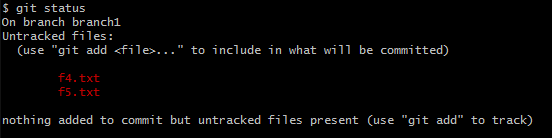


Now u are in branch1 , create some files f4 and f5

remember that you have not created these files in master

check git status

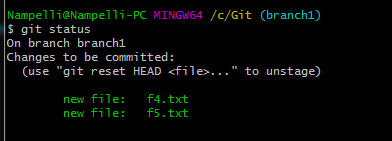
$ git status



now lets add this into index

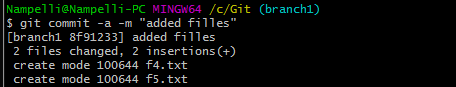
$ git add -A

check status of branch1

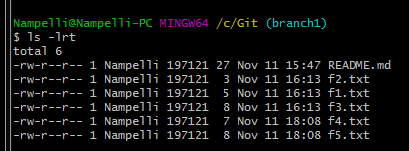


commit changes to branch1

$ git commit -a -m "added some files"

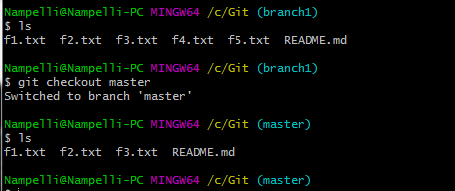


now check all files in branch1 $ ls -lrt



checkout to master branch from branch1

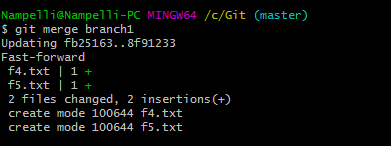
and do ls (we have added f4,f5 only in branch1 not in master ) , can see the difference



**Merging :**

merging is a way to combine the work of different branches . after you have done your work with branch , make sure you have to merge with master branch.

$ git merge branch1



now let's do ls command

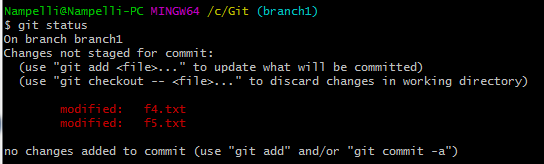
$ls



you can see the work f4,f5 in branch1 has merged with master branch

**do some changes in branch1 , it will not affect master branch**

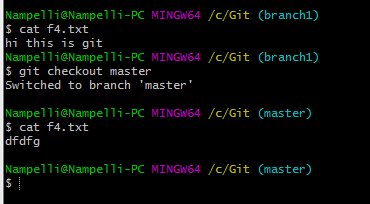
$ git status



commit changes to branch

$ git commit -a -m "modified"

now see the changes by switching to master repo



**Git pull :** it will pulls all the changes from central repo to your master branch

**Git fetch :** it will pulls all the changes from central repo to a new branch , it is not connected to master repo.

when you do git fetch , make sure you have to merger with master branch

**Git pull = Git fetch + Git merge**

**Rebasing:**

Rebasing also a way of combining the work between branches, but it can be use to make linear sequence of commits.

The first thing to understand about git rebase is that it solves the same problem as git merge. Both of these commands are designed to integrate changes from one branch into another branch—they just do it in very different ways

Rebase:

Git rebase to change the history

Below are commits

A B C D

E F

The changes look like A E B F C D

But if you want the history to be like

A B C D E F

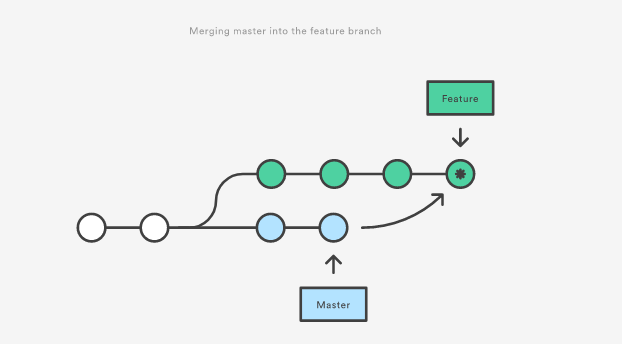
**The Merge Option;**

The easiest option is to merge the master branch into the feature branch using something like the following:

git checkout feature

git merge master

This creates a new “merge commit” in the feature branch that ties together the histories of both branches, giving you a branch structure that looks like this:



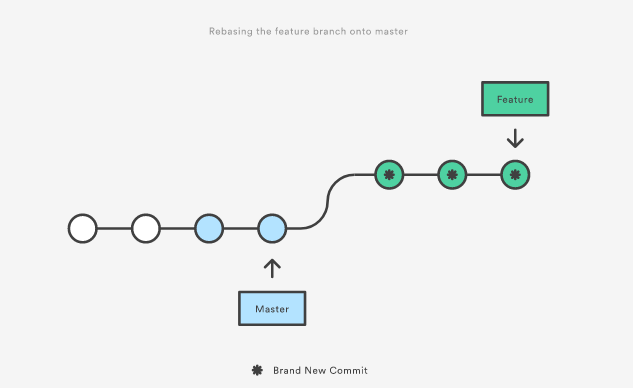
### The Rebase Option

As an alternative to merging, you can rebase the feature branch onto master branch using the following commands:

git checkout feature

git rebase master

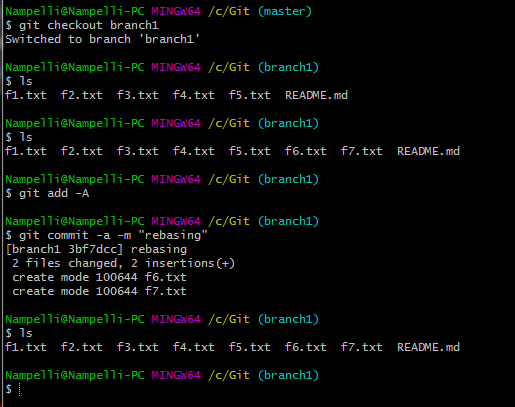
This moves the entire feature branch to begin on the tip of the master branch, effectively incorporating all of the new commits in master. But, instead of using a merge commit, rebasing re-writes the project history by creating brand new commits for each commit in the original branch



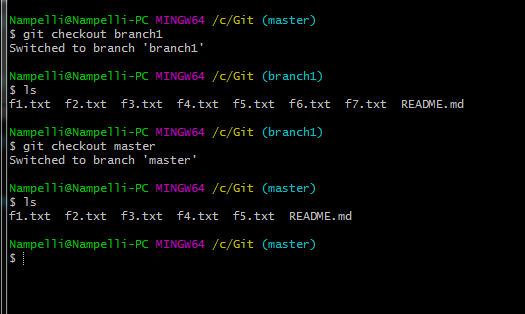
The major benefit of rebasing is that you get a much cleaner project history**. First, it eliminates the unnecessary merge commits required by git merge.** Second, as you can see in the above diagram, rebasing also results in a perfectly linear project history—you can follow the tip of feature all the way to the beginning of the project without any forks.

**Working with rebasing :**

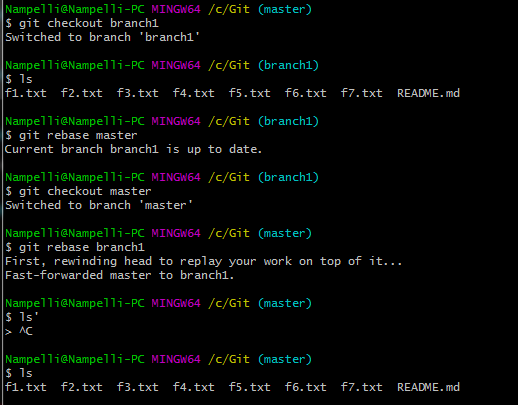
go to your branch1 , and create some file , say f6,f7



see the changes in master and branch1



Now we use rebase instead of merge.

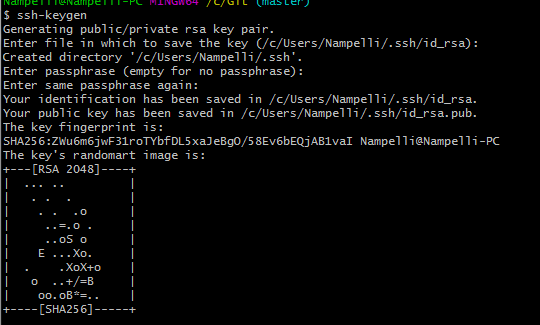


**Now let's push the changes to github central repo:**

To push the changes to github repo , we have to generate ssh key's for authentication purpose.

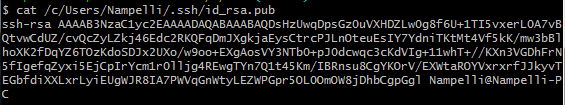
1) generate ssh keys in git bash

$ ssh-keygen

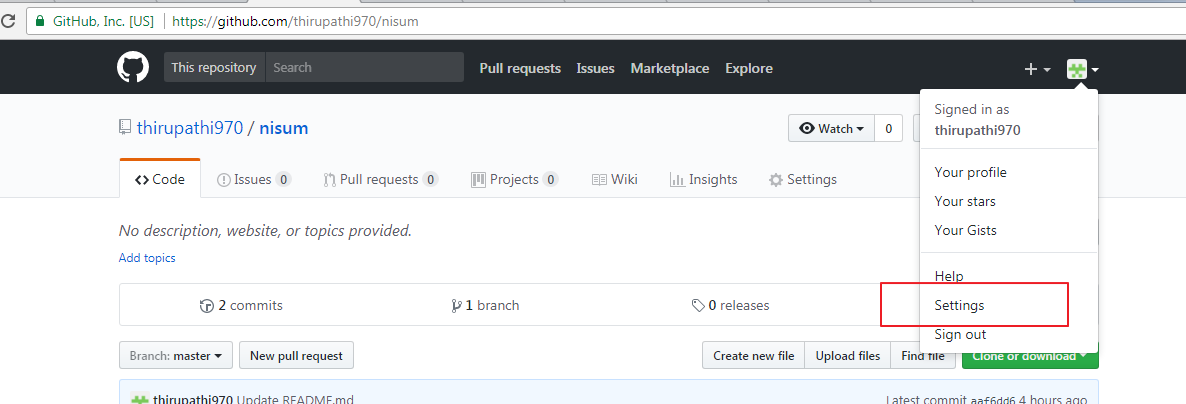


the key has added in /c/Users/Nampelli/.ssh/id\_rsa.pub location

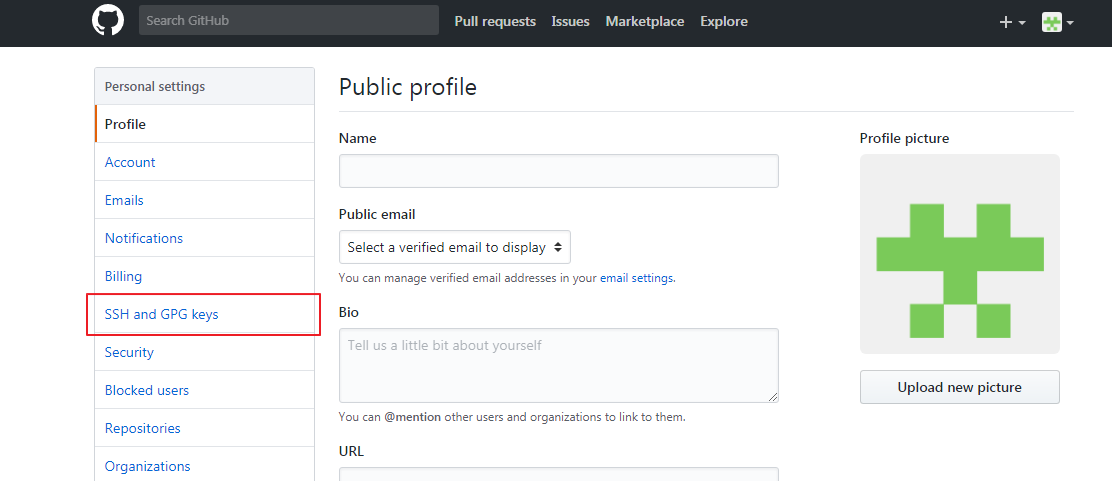
to see the key do $cat /c/Users/Nampelli/.ssh/id\_rsa.pub



Now go back to git hub account , settings>

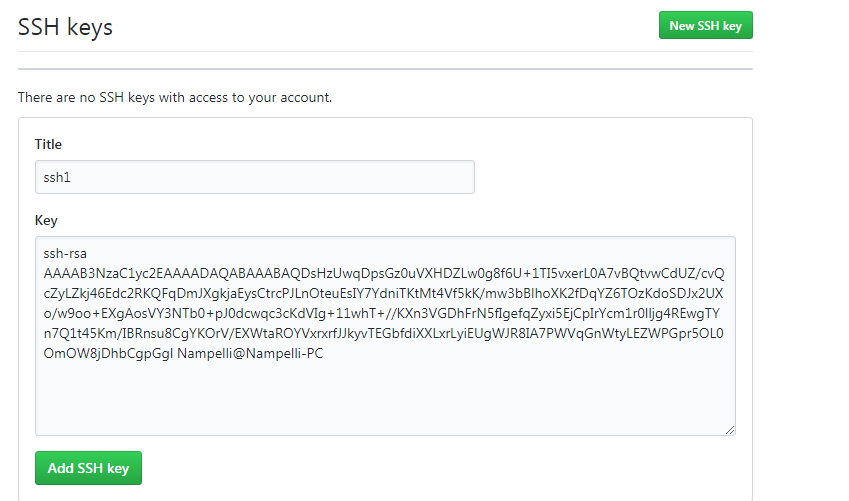


SSH and GPG keys >

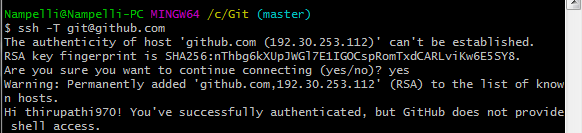


New SSH key >

here you have to give a name for key and copy-paste the generated key from git bash

 click on Ad SSH key , it will make a password less authentication b/w git and github

Now to authenticate do $ git -T git@github.com

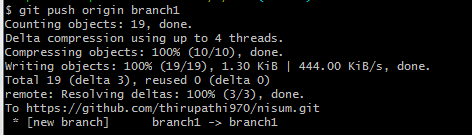


Now let's push the changes to github

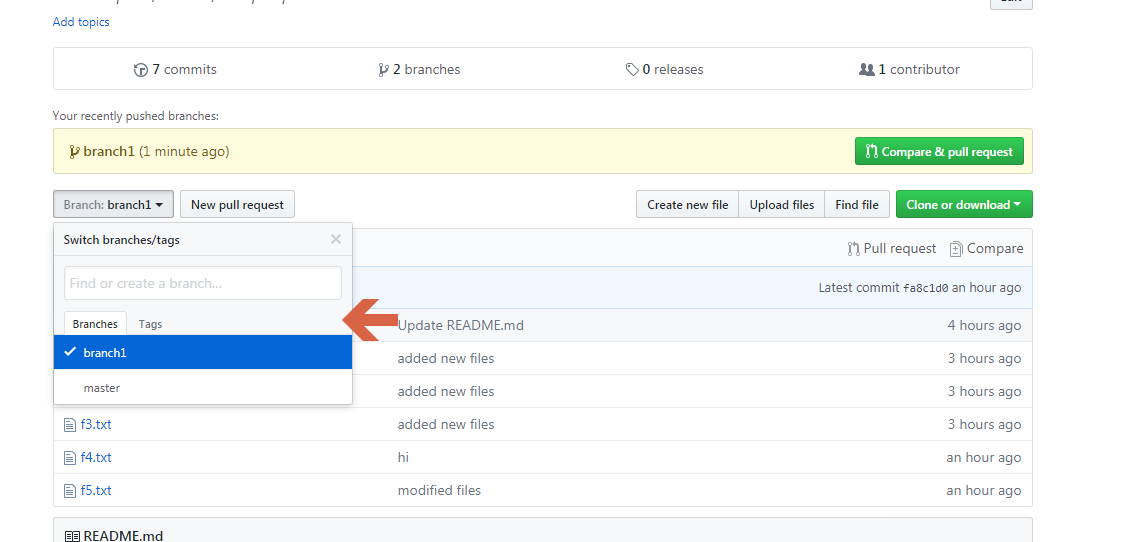
push the master branch1:

$git push origin master

$ git push origin branch1



Verify in github



if you open branch1 , u can see all the files f1..f5

**Git revert:**  it will revert back the changes to previous version.

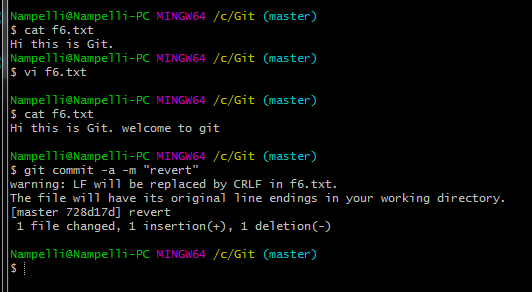
create a new file in master branch f6.txt (content :Hi this is Git.)

add to index $ git add f6.txt

commit to local repo $ git commit -m "revert" .

After committing the changes , you want to modify the file f6.txt

content : Hi this is Git. welcome to git

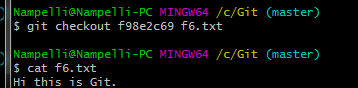


Now i wanted to revert back to previous version (i.e content : Hi this is Git.)

$ git log

u need to copy first 8 characters of commit hash

$ git checkout f98e2c69 f6.txt



Now check the content inside f6.txt , it has changed to previous version (content : Hi this is Git.)

**Git Pull :**  is used to pull changes from central repository

do any modification in file , say f6.txt in github and save changes .

$ git pull origin master .

u can see the changes made in github.

git config --global user.email "you@domain.com"

git config --global user.name "github\_username"

git: fatal unable to auto-detect email address (got "some wrong email")