**$ git init (to initialize git repository)**

**$ git - -version (to check the version)**

**Configuration:**

**$ Git config user.name (to check the current username)**

**$ git config - -list (to check the users list)**

* Name and email address assigned to a commit from a local computer. With git there are many configurations and setting possible. Two important settings are user.name and user.email

**$ git config -global user.name "akhil123ak" (this will setup username for git account)**

**$ git config --global user.email d.akhilchowdary@gmail.com (this is to give email)**

**Git remote:**

**$ git remote –v (list of currently configured remote repositories)**

**$ git remote add origin <url> (connects the local repo to remote repo)**

**$ Git remote add Akhil <url>**

* Here, Akhil is the name we gave and the url is another remote server which we want to access
* So, we can use Akhil instead of origin and even pull by adding the name in command as below
* Git add Akhil
* Git push Akhil etc….
* If we are using private repo, every time we pull or push, it will ask the password
* **Git clone** **<repo URL>** command can be used to clone the whole repository. It will create a folder with repo name and places the code inside it

**Pulling:**

**$ git pull url**

**$ git pull origin master**

**$ git pull –all (to pull all the branches from remote repo which are not present in local)**

**$ git pull origin (to pull the latest code from all the branches of origin)**

**$ git clone url**

**$ clear**

**Adding files:**

**$ git add filename**

**$ git add <filename> <filename>**

**$ git add \* (to add all files)**

**$ git add -A (to add all the files)**

* It adds the files to staging area. When we do commit, git checks the files which are present in staging area. The files need to be added to index before commit

**$ git status (to check the files status)**

* It shows the current state of local repository. If a file is in staging area and not committed, then it will show in status

**$ git reset HEAD <file name> (to unstage the file)**

**Commit:**

**$ git commit -m "committed message" filename**

**$ git commit -a -m "adding all files" (to commit all files) (git add +commit)**

* Records the changes made to the files in local repository. Each commit has unique id, it is always good to give a commit message
* Commit –a commits the files in staging area and also commits the modified files in the directory
* If we don’t give -m for committing the data in git, it will open a text editor to add text and save, then it will commit the data

**Push:**

**$ git push -u origin master (to push)**

**$ git push origin [branch name]**

* Push a branch to your remote repository
* Origin in push command represents remote server, it means alias name for remote repo

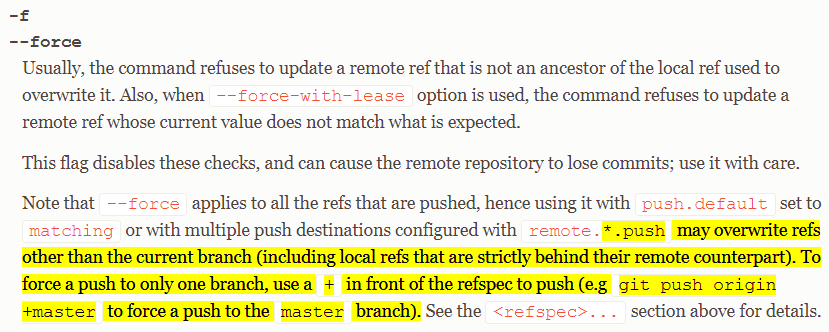
**$ git push -u origin [branch name]**

* Push changes to remote repository (and remember the branch)

**$ git push origin --delete [branch name]**

* To delete a remote branch

**Force push:**



* Force push can be used when remote repo doesn’t accept your changes as you are overwriting something, and u still want to push.
* This happens mainly when we use reset or revert
* We can also use force with checkout command to switch to another branch if you have some changes to be committed. As the git doesn’t allow to switch to another branch without committing. We can also use stash in this situation

**Fetch:**

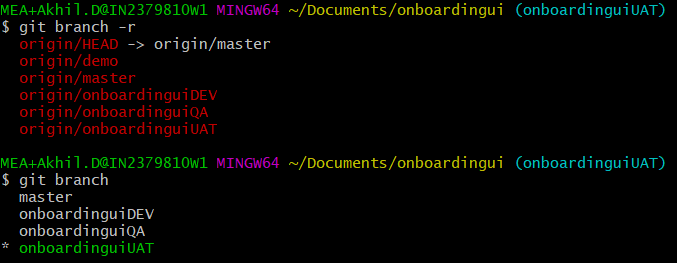
Git fetch just fetches the remote repositories to the local repos. But it won’t merge it with the local repository.

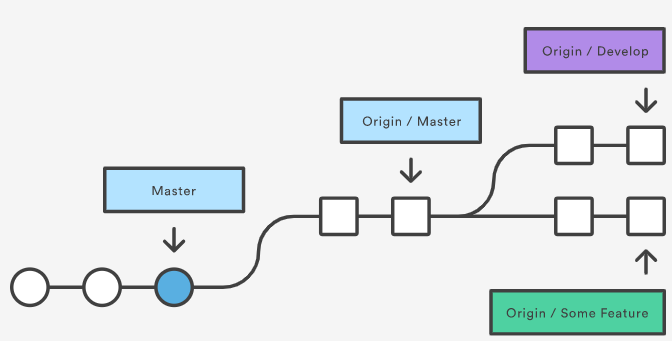
* **git fetch –all (to fetch all the branches from remote repo which are not present in local)**
* **git fetch origin (to pull the latest code from all the branches of origin and synchronizes the local branch with remote)**
* **git fetch origin <branch name> (to fetch particular remote branch**

with these commands. We can pull the remote repos to local. And if we run git branch command as below. We will get the remote branches present in our local workspace with **“origin/<branch name>”.**

we can get those details with the below command

* **git branch -r**



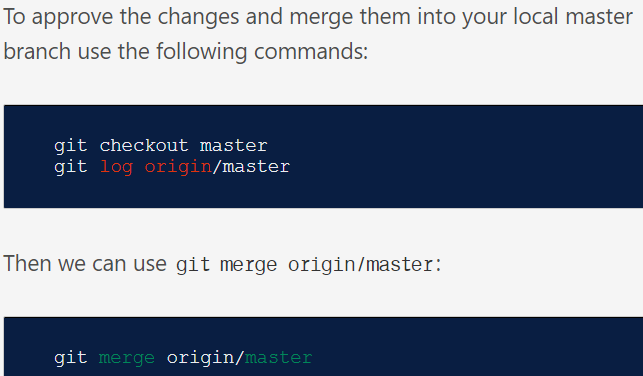


We can switch to origin branch. Check for the log and finally we can merge it with our local branch as below

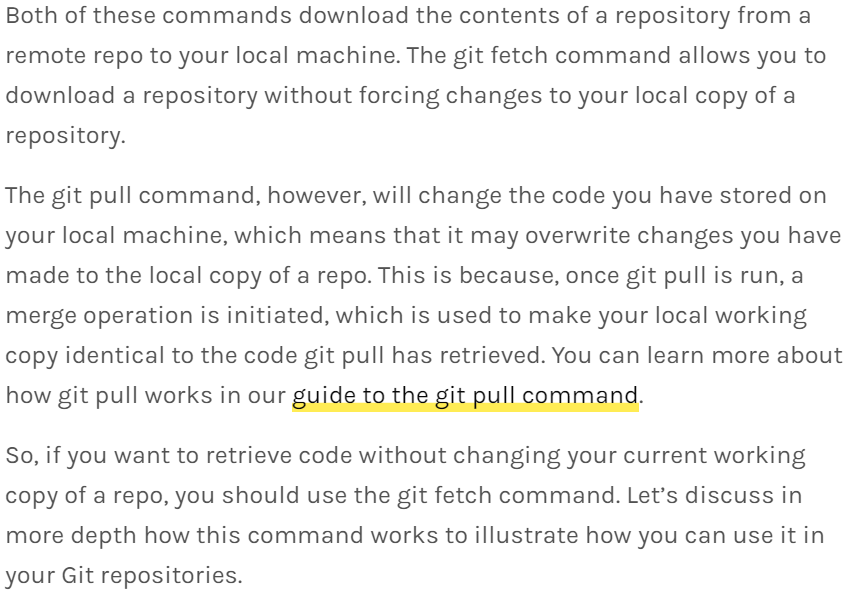
We can also create a new local branch from origin branch as below.

* **git checkout -b remote\_master\_local**

**git fetch** followed by **git merge origin/master**, when on **'master'** branch, is equivalent to issuing **“git pull”** command



**Difference between git pull & fetch:**



**Git log:**

**$ git log (to see all the details)**

**$ git log –summary (view changed detailed)**

* The files when we are pushing, the files are compressed and when we are pulling it, it is compressed only but while storing it in local,
* It will be unpacked. after committed, it will be again compressed, but while working it won't be in compressed mode
* Git has intermediate layer between local and central repository. first, we need to add the files to index, then only we can commit or push that. Index means staging area
* we need to use "git add command" for that
* we can use "git status" to see which files are in index and which or not
* After committing, the version control tool takes the snapshot with the changes

**$ Git log (gives the complete history of repo)**

* It shows the history of commit and author details

**$ git log <filename>**

* Gives the history of particular file

**$ Git log - -author=”author name”**

* Here, it shows the commit logs of author

**$ Git log - -before=”date”**

* We can get the details based on date

**$ Git log - -oneline**

**$ git log –oneline <filename>**

* It shows one line

**$ git log - - A**

* Get into a directory and run the above command to check the log of a directory

**$ git log - - <path>**

**$ git log - - <path>/\***

* Use the above commands to view the log of a particular folder
* We can easily check the log of a file or folder from GUI by simply right click and show log