GIT Bash Commands

git help --> It provides frequently used several git commands.

git help <cmd-name> --> It opens documentation of that perticular command.

git init --> It is used to create empty repository or re-initialise existing repository.

git status --> This command will display status of current repository.

Staged Files :

Files which are added and ready to commit.These file name will be displayed in green color.

Un-Staged Files :

Modified files will be displayed here,we need to stage these files to commit.

These file name will be displayed in red color.

Un-Tracked Files :

Newly created files ,we need to stage them to commit.

These file name will be displayed in red color.

git add --> This command is used to add file to staging area.

Syntax : git add <file-name> // ('.'/'--a' ==> all files)

git rm --> This command is used to un-stage newly created/added files.

Syntax : git rm --cached <file-name> // '*' ==>all files

git commit --> This command is used to commit our changes to git local repository.

Syntax : git commit -m 'commit-message'

Note : When we execute commit command it will consider all files which are in staging area.

GIT Bash Commands

- *To commit our files in remote repository we should execute below two commands*

1) git remote add<repo-url> → This is only first time

2) git push -u origin master → This is used to move changes from local to central

Note : Git local repository available in our machine only.

git reset --> It is used to unstage a file which is already existing and try to modify

Syntax : git reset HEAD <file-name>

git checkout --> It is used to discard changes done in the file (We can say it 'Un-do " operation).

Syntax : git checkout --<file-name>

git push --> To publish local commits to central repository

Note : Whenver we commit ,git will generate commit-id(40-alpha-numeric char but we see only 7 char)

git log --> To check commits history we will use git log command.

Syntax : git log

In commit logs it will display below details

- commit-id
- author
- timestamp
- commit msg

Our TL commited project folder structure to git repository.

Developers job to clone that project and start their development.

git clone --> To take existing project from repository to local system we will use git clone command.

Syntax : git clone <repo-url>

GIT Bash Commands

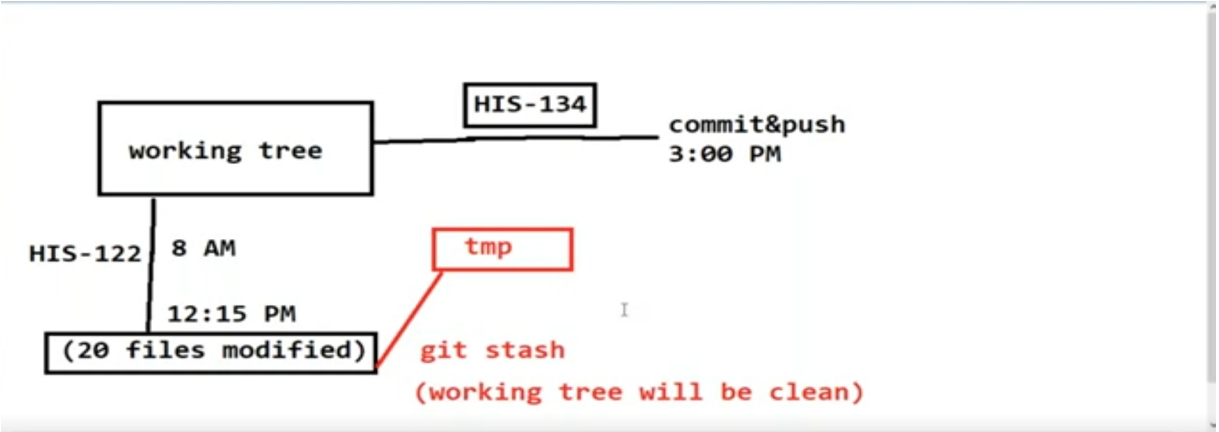
git pull --> Before making any changes to file(s) in local, it is highly recomonded to take latest changes from repository.For this we use pull command.

Syntax : git pull

git stash --> It is used to record current changes and make working tree clean.

git help stash ,git stash list, git stash clear

```
Scenario
--------
Your TL assigned a task(HIS-122) to you mrng @8 AM and you started working on that task.
Few changes already you made in few files but u didn't committed because task is not yet
completed.

Around 12:15 PM your TL told that Park HIS-122 for now and start working on HIS-134 it is
more priority today please complete it by today EOD. After HIS-134 is completed then
start working with HIS-122.
```



git stash apply --> It is used to get the file(s) which we are modified and save in *temporary memory(stash).*

Example : If we are consider the above scenario ,after finishing most priority story(HIS-134) we want to start again previous story(HIS-122) for these we use *git stash apply.*

GIT Bash Commands

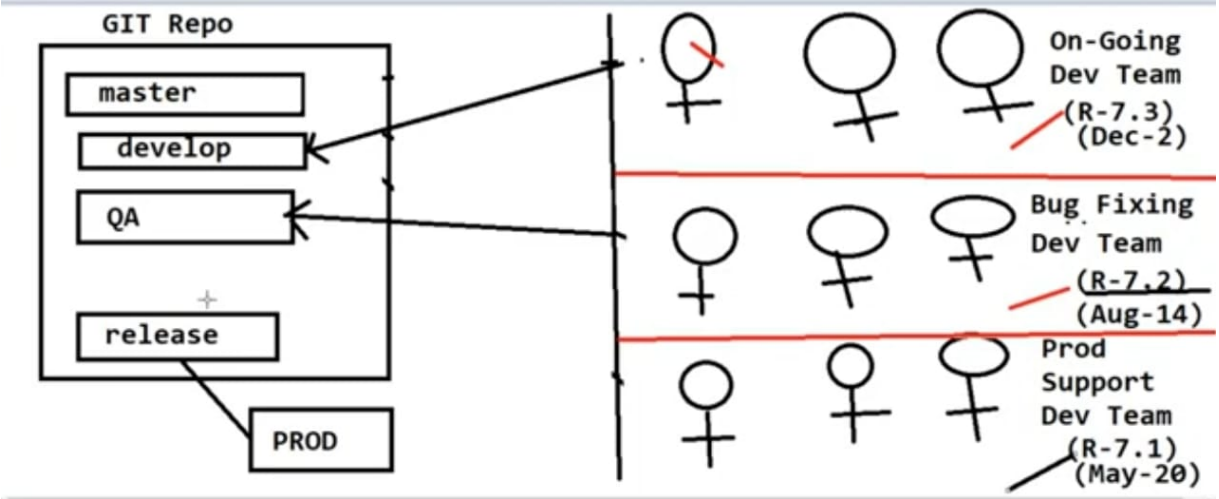
What is branch in GIT HUB ?

- ➔ When we create Git Repository by default it will provite master branch.
- ➔ Branches are nothing but code bases
- ➔ We can create several branches : Git Repository
- ➔ Generally in git repository we will create branches like below

- master(default)
- develop(on-Going Dev team,Bug fixing Dev team,Prod Support Dev)
- feature
- QA
- UAT
- release

Why we need branches in repository?

- ➔ To support parallel development we need to have branches in git repository.



How to create branches in GIT repository?

- ➔ Login to git repository
- ➔ create branch <ex. develop> from master branch
- ➔ Clone <develop> branch code

git clone -b <branch-name> <repo-url> --> It is used to clone branch.

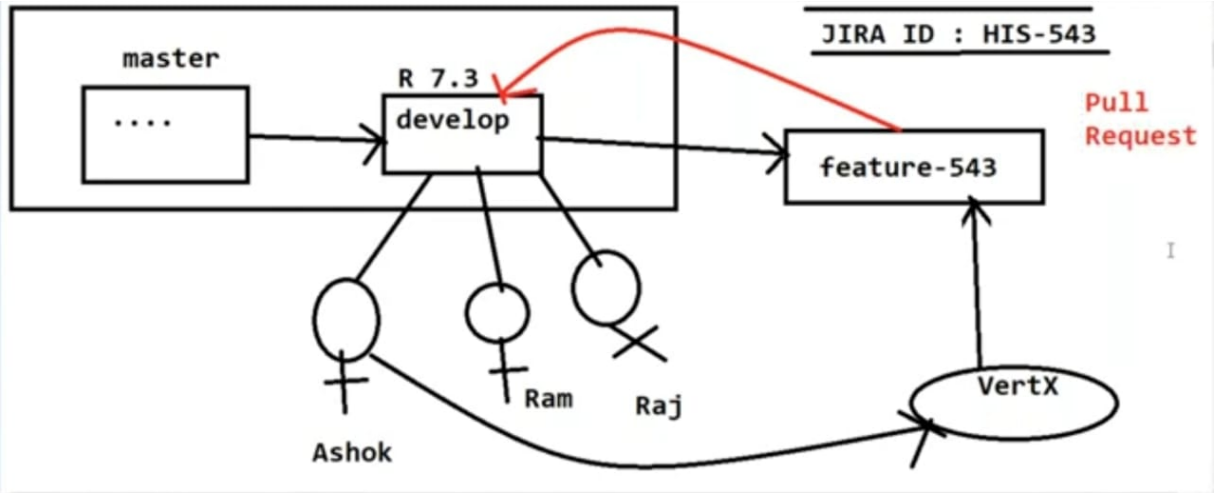
git branch--> It is used to get the branch name.

GIT Bash Commands

Note : If we execute *git clone<repo-url>* it clones master branch code by default.

How to merge branches in GIT Repository?

➔ Using **pull request** we merge branches in Git Repository.



What is pull request?

➔ Pull Request is used to merge one branch code to other branch .

```
-> To support parallel development we need to have branches in git
repository

-> Generally in realtime we will create several branches like below

                master ( default )
                develop
                feature
                QA
                UAT
                release

-> In GIT Hub account we can create branches in repository
```

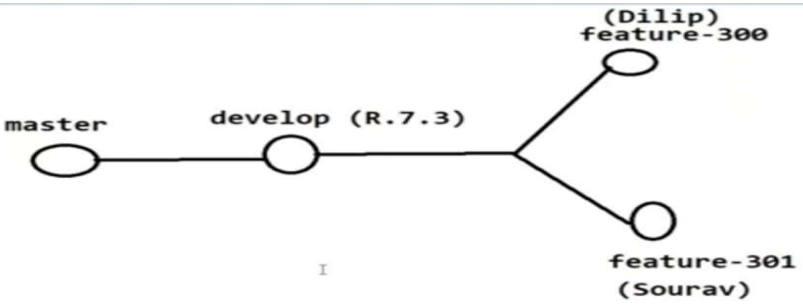
GIT Bash Commands

```
-> Clone branch from repository

-> Make changes and push your changes to branch which you created fo ryour
feature

-> Once Your development & Unit testing is complete then merge your
changes to main branch


-> To merge changes from one branch to another branch we will create 'Pul
Request'

-> When we execute pull request, GIT Hub compare source branch and Target
branch and it will confirm can we merge these branches or not
```

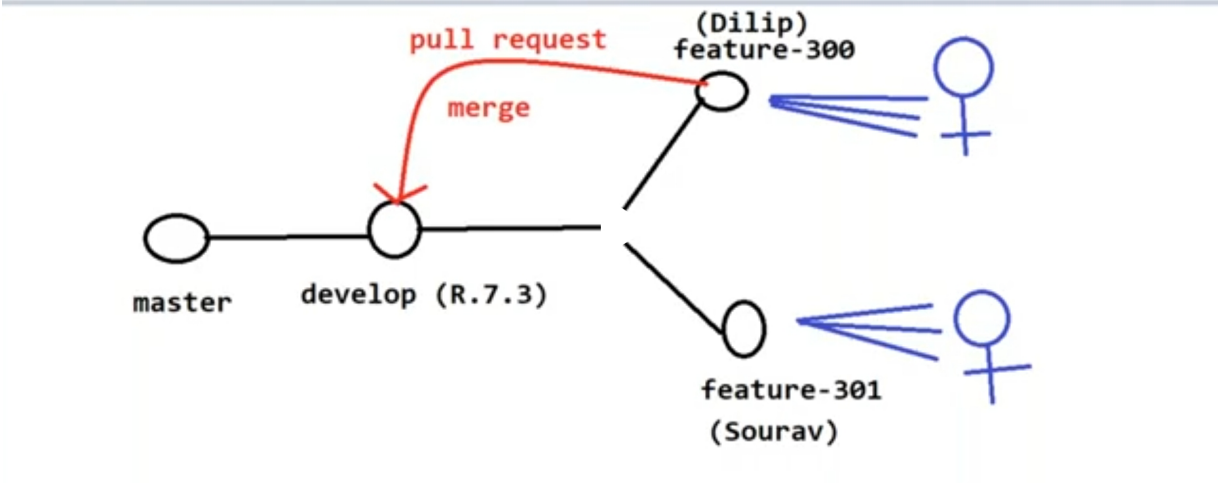
```
-> If staus is'Able to merge' then execute pull request and merge changes

-> After pull request execution got completed, we can delete that new
branch which we created for our story
```

How to resolve conflicts in code?

```
Realt-time Scenario
-------------------
Two developers are working on Sprint 7.3

Dilip & Saurav are the developers

--> Dilip working on HIS-300 story

--> Sourav working on HIS-301 story


Note : Latest code is available in develop branch
```



GIT Bash Commands



- When two or more developers change code at same line no. that causes conflicts at the time of branch merging.
- To solve conflicts to merge branches we have to use manual approach

- ➔ Go to GIT HUB repository
- ➔ Compare and pull request (shows conflicts occur)
- ➔ Create pull request
- ➔ Resolve conflicts (This branch has conflicts that must be resolved)
- ➔ Remove the junk character(conflicts in the code) from the code
- ➔ Mark as resolve and Commit Merge
- ➔ Merge pull request
- ➔ Delete the unused branch