Git and GitHub Tutorial

Git Commands:

GitHub is a cloud platform where we can create project repository. project repo will be available for all the team members. To perform operation, we need **git client.**

We can perform operations using two ways:
1. Git bash

2. git GUI

3. by using Ides(like eclipse visual studio)

Programmer is having flexibility to use any approach to connect with git repo.

Git Bash Commands:

We are having several commands to perform operations

1. **init**

If you don't know any git bash commands just type: **git help**

If you want to know about particular command type: **git help <command name>** it will open documentation. It will provide details for that particular command.

git init: it is used to create empty repository or re-initialize the existing repo.

Create one folder inside any folder type: **git init**

Create one file inside the folder and type**: git status**

2. **status:**

   This command will display the status of current repository

   Mainly Here will see three types of files

   i. **Staged files:** Files which are added and they are ready to commit.

   These file names will be dispalyed in the green color.

ii. **Un-staged files:** Modified files will be displayed here, we need to stage these files to commit.

These file names will be displayed in Red color.

iii. **Un-tracked files:** Newly created files, we need to stage them to commit.

These file names will be displayed in Red color

Whenever we execute the command will see three types of files **staged files, un-staged files and un-tracked** files.

3. **add:** this command is used to add a file to staging area.

If you want o add one file to staging area:

Syntax: git add <file name>

If you want to add all the files at a time to staging area.

Syntax: git add --a (or) git add .

If you want to unstage the particular file

Syntax: git rm –cached <file name>

If you want to unstage all the files at a time

Syntax: git rm –cached *

4. **git commit:**

This command used to commit our changes to git local repository.

Syntax: git commit –m "commit-message"

Note: when we execute commit it will consider all the file which are in staging area.

Note: git local repository will be available in our machine only.

How can I commit my changes to central repository ?

git remote add origin <git repo url> (this requires only first time to execute.)

git push –u origin master (This is use to move changes from local to central)

Now changes are committed to the central repository GitHub.

5. **Git reset**: it is used to unstage a file. **Existing file if you want to unstage:**

   Syntax: **git reset HEAD <file-name>**

6. **Git restore:** if you want to undo the changes whatever you changed use reset|checkout. Just to ingore/discard our change will use **checkout** or **restore** commands.

Syntax: **git restore <file-name>** or **git checkout -- <file-name>**

Summary of commands:

- **git help**
- **git help <command –name>**
- **Git init**
- **Git add**
- **Git add –a (or) git add .**
- **Git commit –m "commit-message"**
- **Git push**
- **Git reset HEAD <file-name>**
- **Git checkout <file-name>**
- **Git status**

==========

When we make changes to files we need to stage them to commit.

- To add file(s) to stagging area we will use **add** command
- To commit file(s) to local repo we will use **commit –m "msg"**

Note: we should provide reason for commit as a message.

- To publish local commits to central repository we will use **push** command

★ Whenever we commit, git will generate commit-id
★ Commit-id contains 40 alpha numeric characters
★ From 40 alphanumeric characters it will display first 7 characters to us as commit id
★ If you want to see we can see 40 checks in GitHub/bitbucket

7. **git log:** If you want to check history of commits, we will use **git log**

   Syntax: git log

8. **git clone:** to take existing project from repository to repository to local system.

Syntax: git clone <repo –url>

9. **git pull:** before making any change(s) to files in local it highly recommended to take latest changes from repository. To take latest changes from repository we will use **git pull** command.

Syntax: **git pull**

*** Note: **.ignore** file we can ignore few files.

10. **Git stash:** it is used to record current changes and make working tree clean.

Scenario:

You TL assigned a task(ALGR-122) to you in the morning 8 am and you started working on that task. Few changes you already made in few files but you didn't committed. Bcoz that task is not yet completed.

Around 12: 15 pm your TL told that park ALGR-122 for now and start working on ALGR-123 it is more priority today please complete by EOD.

After completing ALGR-123 start working on ALGR-122.

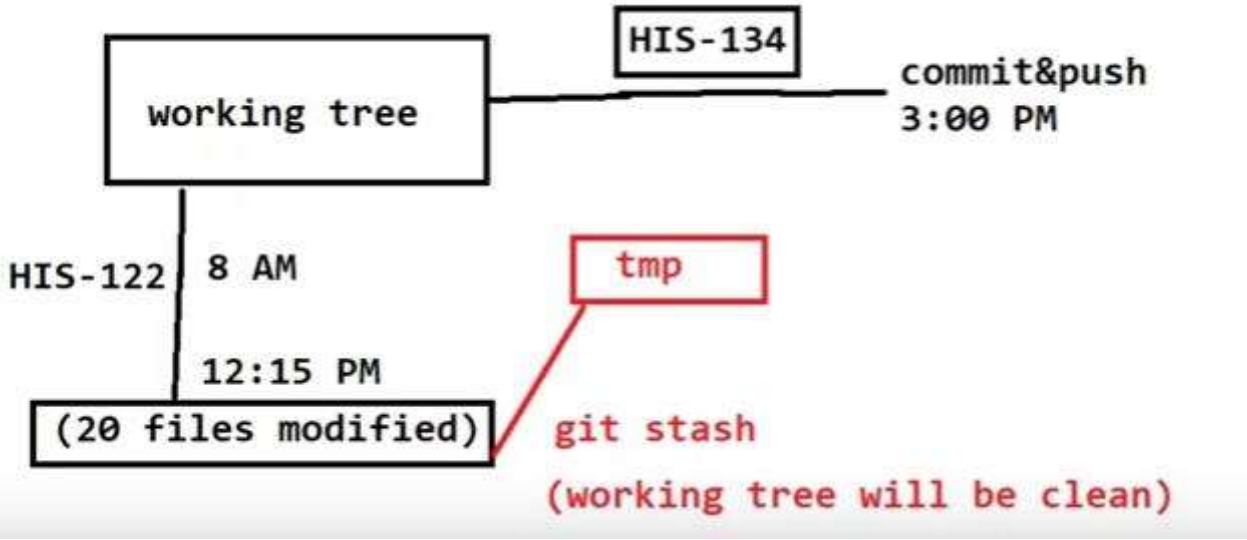
- We cannot delete the files and we cann't start working on 123 bcoz while commiting the changes will be confused which files are for 122 and which for 123.

By above scenario stash the current modified and start working on 123, stash is temporary storage.

Syntax: git stash ====> changes will be stored in temporary area. Now you can work on new task and commit the changes.

To get back the changes for 122:

Syntax: git stash apply



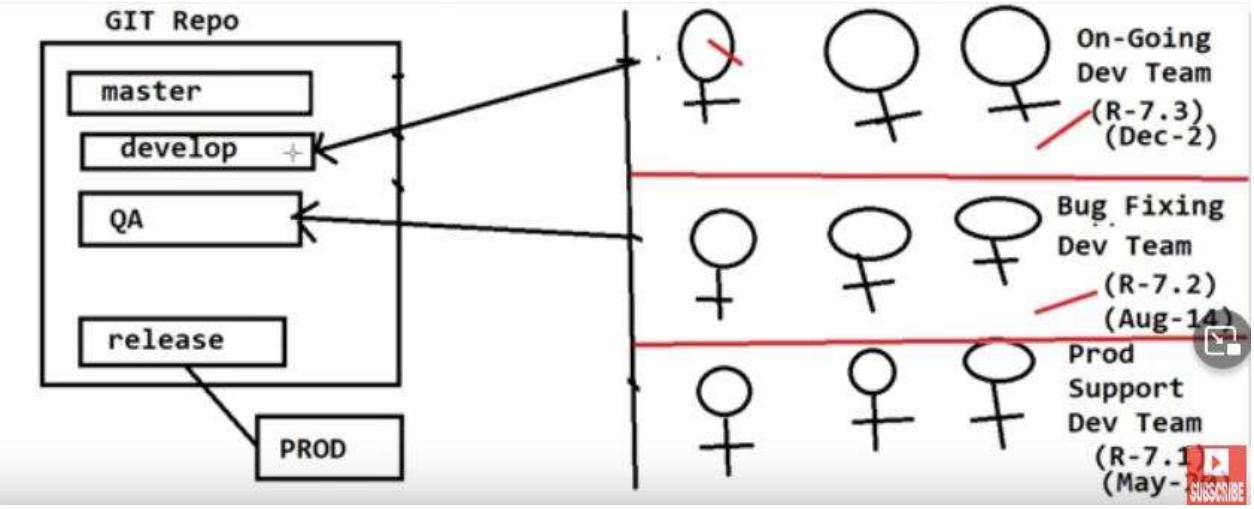
==========

What is branch in Git Hub?

Branch is nothing but a codebase. Branches are used for maintaining separate codebases.

- When we create git repository by default, it will provide master branch.
- Branches are nothing but codes
- We can create several branches in Git Repository.
- Generally, in Git Repository we will create branches like below.
  - Master(default)
  - Develop
  - Feature
  - QA
  - UAT
  - Release

Note: it is highly recommended to create feature branch for every story to not disturb existing functionality.



Clone any other branch code:

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

- If you wnat to clone specific branch code we execute below code.

Syntax: git clone –b <branch-name> <repo-url>

11. How create feature branch ?

Syntax: **git branch <branch-name>**

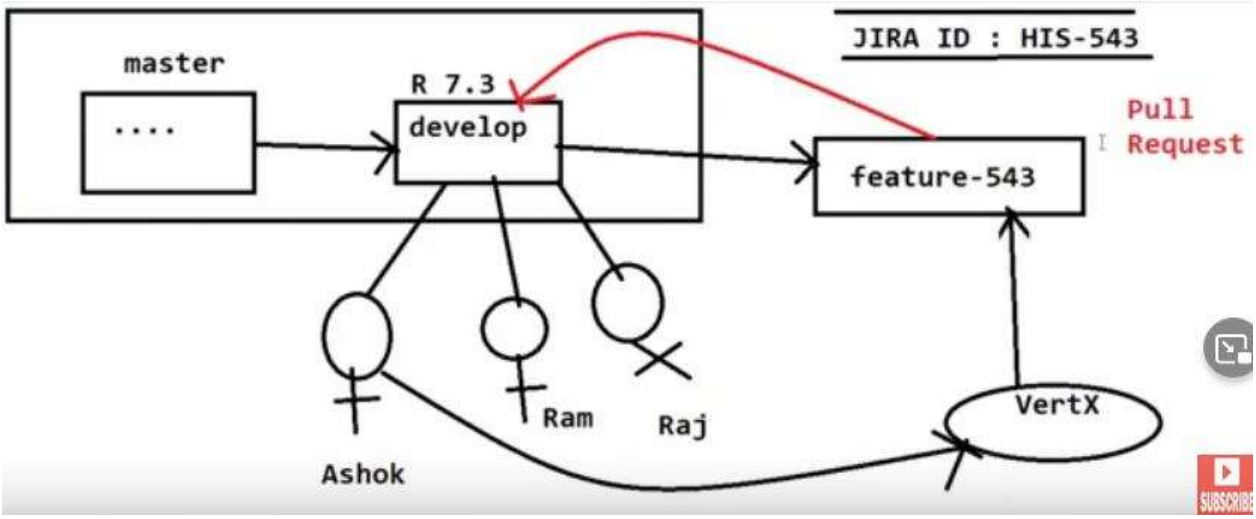
After creating the branch push that branch to remote repository using.

Syntax: git push –set-upstream origin <created-branch-name>

What is pull request?

Branch is a separate code base that will be created by developer to not disturb existing functionality.

Pull request is nothing but merging the branches. (we will raise pull request(PR) for merging the branches)



Merging branches using pull Request?

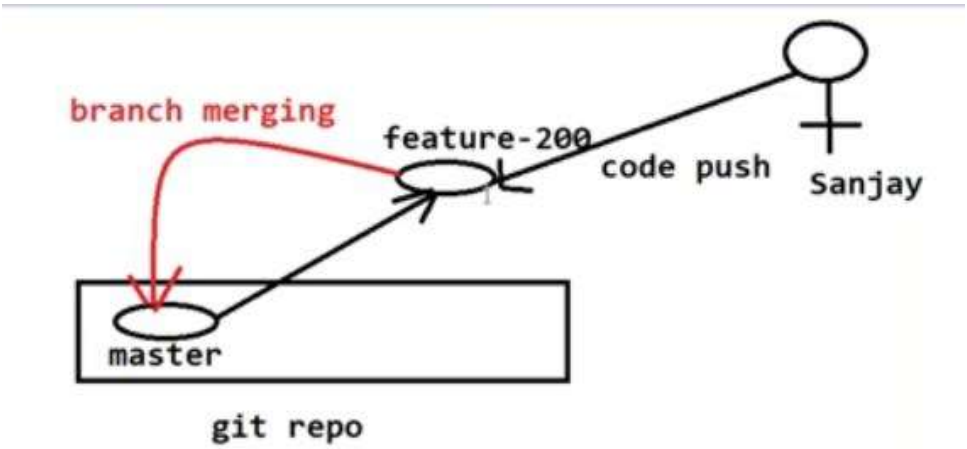
- When we create a Git Repository in github we will get a master branch by default.
- To support paraller development we need to have separate branches in git repository
- Generally in real time we will create several branches like below.
  - Master(default)
  - Develop
  - Feature
  - QA
  - UAT
  - Release

Create feature branch ex: feature-200

Steps to create feature branch:

- **git branch feature-200**
- **git push –set-upstream origin feature-200** ====> push this brnach to remote repo.
- Do the code changes and commit it by using **git commit –m 'commit-msg'**
- Push the code changes to the remote repo**: git push**

Once your development done and unit testing is dones merge the branch with master/dev branch by raising PULL REQUEST(PR):

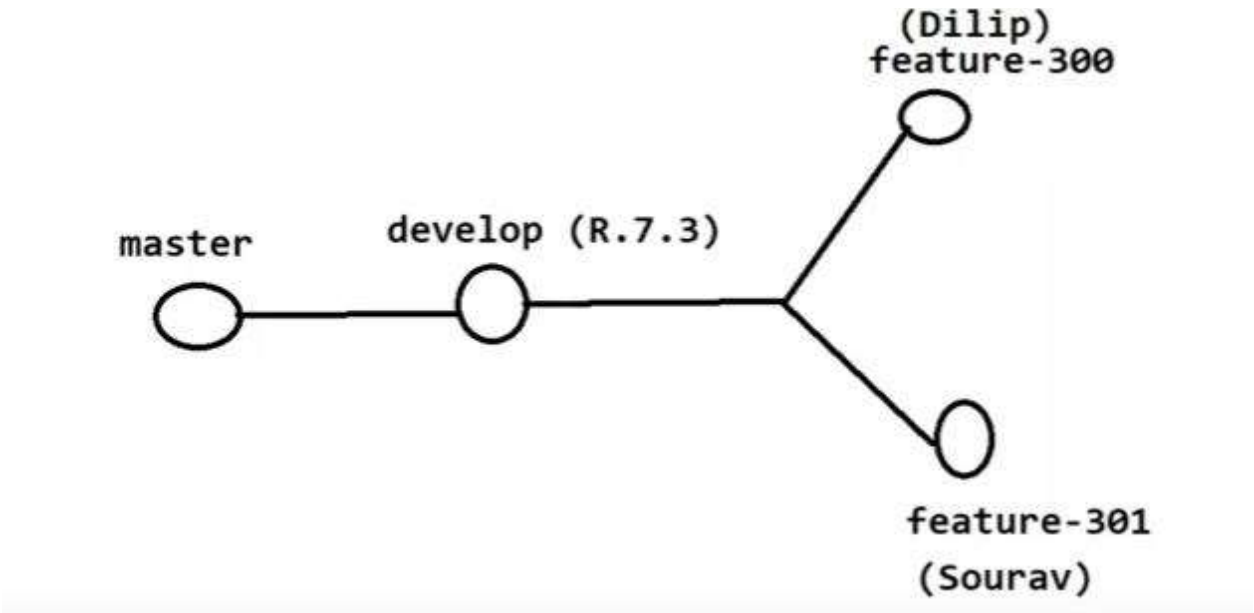


- When we execute pull request, Git Hub compares source branch and target branch and it will confirm can we merge these branches or not.
- If status is able to merge then execute pull request and merge the changes.
- After pull request got completed, we can delete that new branch which we created for our story.

Real-Time Scenario:

- Two developers are working on spring 213
- Dilip and saurap are the developers
- Dillip working on HIS-300 story
- Other developer saurap working on HIS-301 story

Note: Latest code is available in **develop** branch (**Base branch)**



In the same file and same line if we update we will get merge conflict.