**Git Bash Tutorial**

**Introduction:**

Git or git bash is used for version control and remote collaboration between developers.

It allows several people to simultaneous work on a file/repository.

**How?-** By creating several local copies of a remote folder.

**Some key concepts:**

**Repository-** File Directory

**Commit-** Saved changes to git repository (only a change that has been saved can be called a commit). Each commit is logged and has unique SHA1 hash value that identifies it.

**Branch-** Timeline with commit, master is main branch by default, main branch also refers to original version of the repo. Branch is a parallel version of a repo. You can clone the main branch(main codebase) into other branches (parallel versions of main codebase) too.

**Pull-** Pull is command of gitbash that creates a local copy of the remote repo to work on.

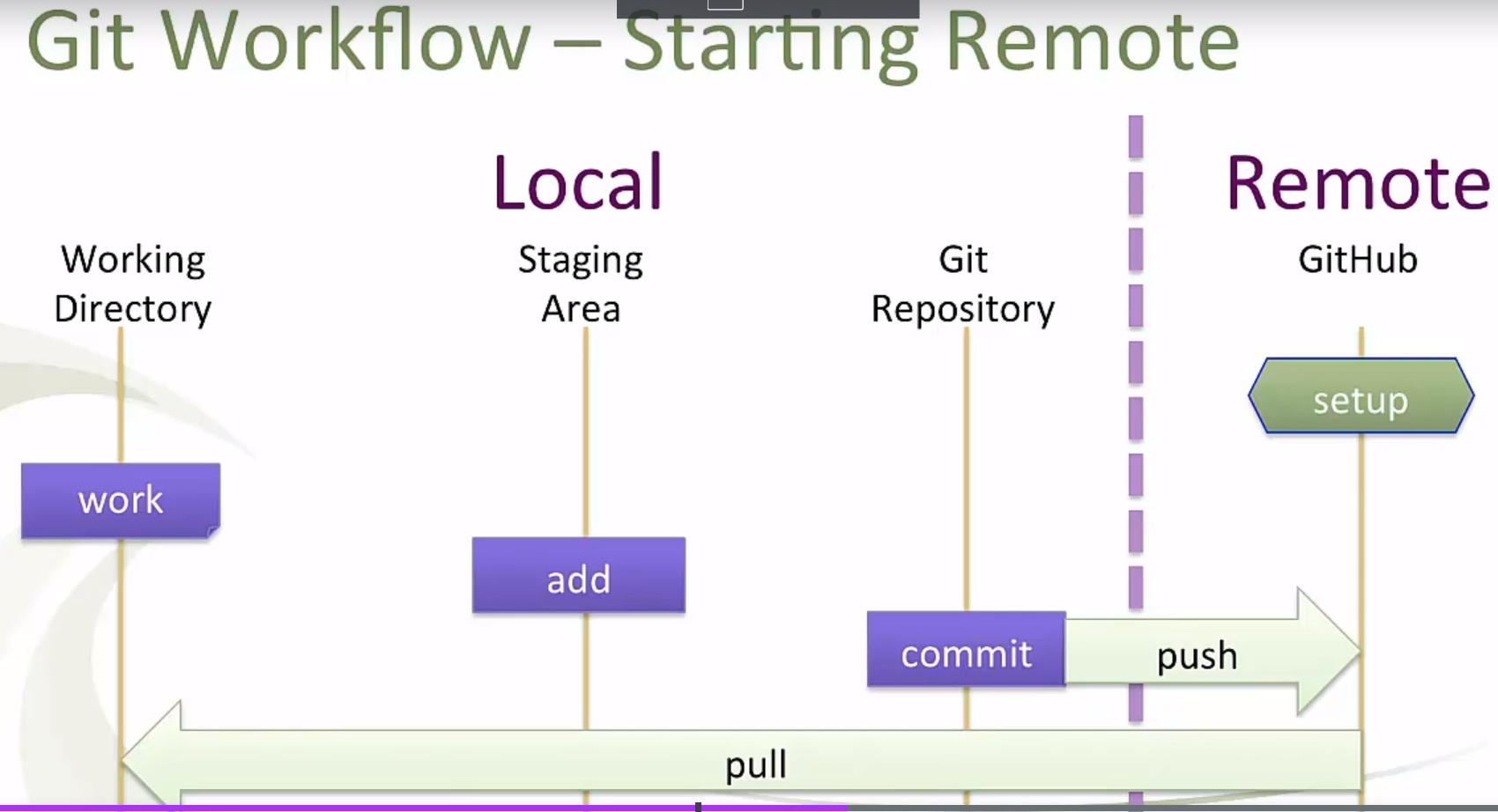
**Pullrequest-** Part of github web, after you clone a repo and make some changes the next step is to merge these changes with original, that’s when you raise a pull request which is essentially you asking the owner to merge your changes to main branch.

**Stagging-** Before staging you must know what a modified file in git is. A file is considered modified if git find changes in it compared to last commit state.

It is a step in the Git workflow where you select modified files or specific lines within files to be included in the commit place them in stagging area which is kind of buffer state before final commit. The staging area facilitates the ability to review and fine-tune your changes before committing them to the repository. However, the staging area operates locally on your machine and is specific to your Git workflow.

**Push-** Push is command in GIT to push your changes, "push" refers to the action of sending your local commits to a remote repository, typically hosted on a service like GitHub, GitLab, or Bitbucket. Pushing allows you to share your changes with others, collaborate on a codebase, and keep your local and remote repositories in sync.

**WorkFlow:**



1. Initialize working directory locally, we will do our work here
2. Add the working directory with it’s changes to Staging area
3. Commit the changes present in staging area to git directory present locally.
4. Push changes from local git directory to remote git directory.
5. Pull in changes made into remote directory to local directory.
6. Repeat Step 1 to 5 again.

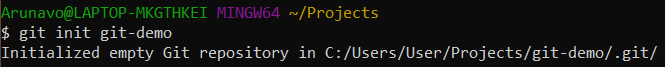
**GIT Bash Commands for working Locally:**

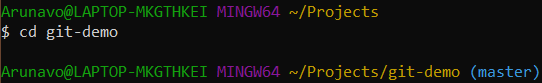
1. **Loading the working directory-** First step is to load or create the directory which which will be your working directory. The working directory will the directory where you will do all the work, it will contain the files in which you are making changes.

There are 2 ways to do the above, either you create/initiate the working directory first and then navigate to it OR you can set the current directory you are in as working directory.

$git init git-demo //will set the projects directory as the working directory

$cd git-demo //The navigate to the location of c:\\projects





OR

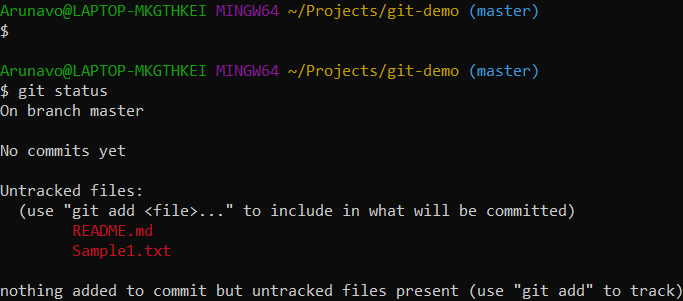
$cd c/projects/git-demo //will navigate to c:\\projects

$git init //will set the current location that is c:\\projects as working directory

1. **Make changes/do work within working directory-** Now that you have set the working directory, you can do some work and make some changes with the files present in working directory. For e.g. make a new file Sample1.txt within this directory and make some changes in Sample1.txt

**Git continuously tracks changes within the working directory.** You review these changes by using command:

**$git status** //this will show info regarding changes made and changes in staging are ready to be commited

****

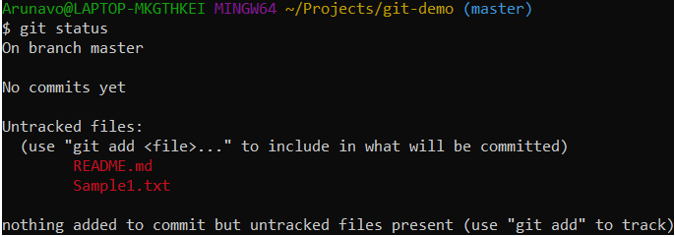
**The above info decoded-**

* First line “on branch master”, indicates we are working on the original version of the directory
* Second line “No commits yet” means no changes have been commited yet
* Third line “Untracked files:….” Shows that there are two files but both haven’ been added to stagging area.
* Changes can not be commited unless changed files are first added to stagging area.

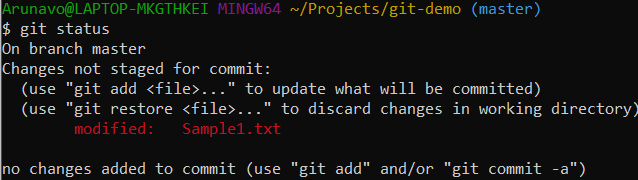
1. **Add the changed files to stagging area**- Once you made changes to your file, execute the:

$git status

It will show something like this:



OR this:



Both the above results means that the git successfully tracked the changes in files, but these changed files have not been loaded to stagging area, hence the changes cannot be commited.

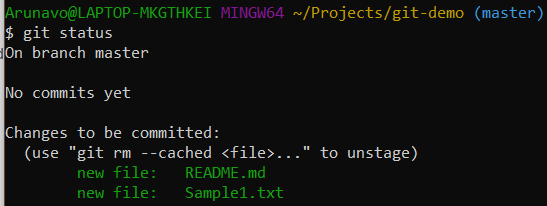
To add the changed files to Staging area use the below command:

$git add Sample1.txt //Or you can use $git add “filename”

$git add . //This will add any untracked, unstaged file



Now type- $git status



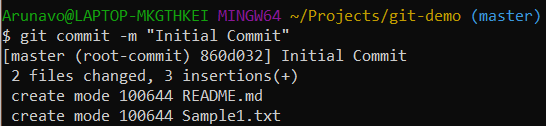
**The above result shows that both files README.md and Sample1.txt have been successfully added to stagging area and there changes are ready to be commited to master branch**

**Note:- The $git add command does 2 things, it add the file to staging area and starts it’s tracking**

**Note:- Even changes like deletion of a file staged area is considered an uncommited change**

1. **Commit the files in stagging area-** Now that changes have been made in the intended files and those files are added to staging area you can finally commit them using thye command:

**$git commit -m “Some message” //This will commit your file**

****

**OR**

You can remove/unstage the files from staging area using the command:

**$git rm –cached Sample1.txt**  //This will remove your file from staging area

**Important Note:-** The standard steps are, add->commit, But there is short to do both with one command:

**Git commit -am <file> “message”-** This will add and commit your file with message

1. **Some other useful command-**

* **git log-** All the commits are logged, this w8ill provide that log
* **git log --oneline --graph --decorate --color**
* **git restore <file>-** Will restore the file to last commit state
* **git rm –cached <file>-** This will stop git from tracking changes in your file
* **git restore --staged <file>-** Unstaging/removes the file from staging area, but it remain to be tracked by git.
* **git rm <file>-** Will delete the file from working directory
* **mv <file> location-** Will move file to location, For e.g. **$mv Sample1.txt** c/document
* **mkdir Web-** Will create a new directory by the name Web
* **ls -a**
* **git add .gitignore**
* **git config --global user.name "Jason Taylor"**
* **git config --global user.email "jason@jasongtaylor.com"**
* **git config --global –list-** Listing All Global Configuration Settings
* **cat ~/.gitconfig-** Seeing Git's User-based Config file

**Important Note:-** Staging and tracking a file is 2 separate things. Staging is like a buffer through which all your changes have to go through before commit. Tracking is git keeping track of files for changes.

You can track a file by adding it once to staging. Once file has been added to staging area it will continue to be tracked by git even if removed from staging area. If you want to untrack your file use the command: **git rm –cached <file>.**

This will untrack your file and then changes in the file won’t be tracked by GIT, all GIT will know is there is a file in working directory which is not being tracked and that info will be showed in status.

**Going Remote:**

**Method 1: Clone a already present remot repo to local Machine**

1. **Find/create the .ssh directory and navigate to it**

First create an .ssh directory in your local system, if you have generated the SSH key before, then this directory will be already present in your in system.

Now navigate into this directory.

1. **Generating an SSH Key**

Once you are inside the the .ssh directory you need to generate the SSH key for github access. Use the below command to generate an SSH key for your user email.

**ssh-keygen -t rsa -C "your.name@your-company.com"**

**Use your actual email address in the example above.**

You can generate multiple SSH keys for multiple user emails, but remember only the last generated key will be stored in your id\_rsa.pub file, only this will used for remote repo operations

1. **Add the generated SSH key for your email to git hub.**

The above process of SSH key generation will create two files, id\_rsa & id\_rsa.pub files, in your .ssh directory.

Next you need to open the id\_rsa.pub file and copy the whole content of it.

Next you need to add this key in your github account, follow the below flow of steps:

Go to github.com->Sign in->Tap on profile pic in top right->Tap on “Settings”->Tap on “SSH and GPG Keys”-> Tap on “New SSH Key”-> in the key section paste the copied value from id\_rsa.pub file and in title give anything.

1. **Verify SSH authentication**

Once the SSH key has been generated and added to github, we need to verify it on local system. Type the below command to do that. This only a one thime thing

**ssh -T git@github.com**

**Above command uses ssh to connect to GitHub over the SSH protocol.**

1. **Create repo in github**
2. **Clone target rego to local machine**
3. **Make changes in cloned working directory**
4. **Follow the work flow of locally working**

**Add the changed files in repo to stage**

**Commit the changes**

1. **Push the changes into remote copy of the repo**

**Method 2: Link local repo with remote repo**

1. Open git bash locally
2. Navigate to any git directory (that has the .git file)
3. Set the destination-

Remember we are trying to establish a one way connection, where there is a source repo which is sending changes to a target repo. So that being said our local git repo is the source, now we have to set the destination for our source.

To do that use the following command which will

git remote add origin [git@github.com:DuesExMachina/git-demo.git](mailto:git@github.com:DuesExMachina/git-demo.git)

The above command will set the remote directory DuesExMachina/git-demo.git as the remote origin for your local git repo

git remote rename origin old-origin

1. git remote -v – Will show the origin for pushing and fetching
2. make changes in your local repo, and add and commit the changes
3. Now push the changes from local repo to remote repo, using

git push -u origin master //-u is needed only the first time

1. git pull origin master- Show the status of the remote repo with respect o local repo

It’s good practice to pull before pushing to avoid conflict.

Sudo xcode-select –install

#### Git Resources

Below are some recommended resources to continue your Git education.

##### **Books**

[Pro Git](http://git-scm.com/book) - Free Book on Git, worth every penny.

Sites- <http://git.training>

Email- info@git.training