Git

Presented by
VAISHALI TAPASWI

FANDS INFONET Pvt.Ltd.
www.fandsindia.com

Ground Rules

- Turn off cell phone. If you cannot please keep it on silent mode. You can go out and attend your call.
- If you have any questions or issues please let me know immediately.
- Let us be punctual.

Agenda

Git

- As **Git** is a distributed version control system, it can be used as a server out of the box. Dedicated **Git** server software helps, amongst other features, to add access control, display the contents of a **Git** repository via the web, and help managing multiple repositories.

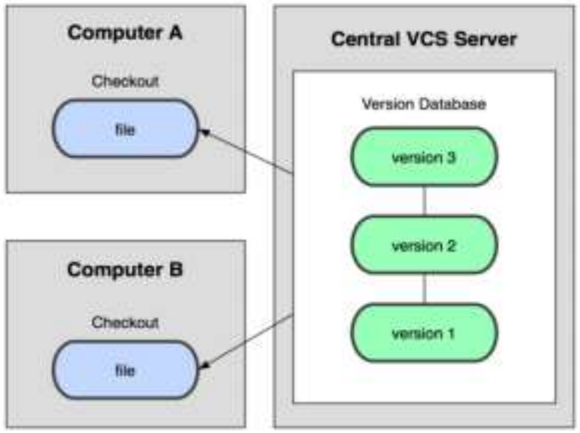
Version Control Systems

- Version Control (or Revision Control, or Source Control) is all about managing multiple versions of documents, programs, web sites, etc.
  - Almost all “real” projects use some kind of version control
  - Essential for team projects, but also very useful for individual projects
- Some well-known version control systems are CVS, Subversion, Mercurial, and Git
  - CVS and Subversion use a “central” repository; users “check out” files, work on them, and “check them in”
  - Mercurial and Git treat all repositories as equal
- Distributed systems like Mercurial and Git are newer and are gradually replacing centralized systems like CVS and Subversion

Why Version Control?

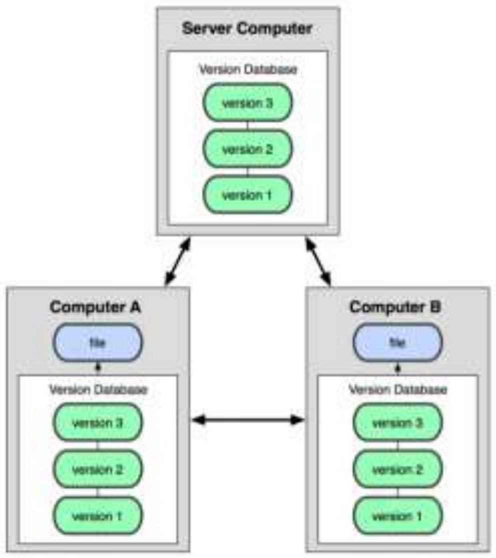
- For working by yourself:
  - Gives you a “time machine” for going back to earlier versions
  - Gives you great support for different versions (standalone, web app, etc.) of the same basic project
- For working with others:
  - Greatly simplifies concurrent work, merging changes

Centralized VCS



- In Subversion, CVS, Perforce, etc.
  - A central server repository (repo) holds the "official copy" of the code
  - The server maintains the sole version history of the repo
- You make "checkouts" of it to your local copy
  - You make local modifications
  - Your changes are not versioned
- When you're done, you "check in" back to the server
  - your checkin increments the repo's version

Distributed VCS (Git)



- In git, mercurial, etc., you don’t “checkout” from a central repo
  - You "clone" it and "pull" changes from it
- Your local repo is a complete copy of everything on the remote server
  - Yours is "just as good" as theirs
- Many operations are local:
  - Check in/out from local repo
  - Commit changes to local repo
  - Local repo keeps version history
- When you're ready, you can "push" changes back to server

Why Git?

- Git has many advantages over earlier systems
  - – More efficient, better workflow, etc.
  - – See the literature for an extensive list of reasons
  - – Of course, there are always those who disagree
  - – Very Popular

Version Control Terminology

- Version Control System (VCS) or (SCM)
- Repository
- Commit
- SHA
- Working Directory
- Checkout
- Staging Area/Index
- Branch

Version Control Terminology

- Version Control System :
  - A VCS allows you to: revert files back to a previous state, revert the entire project back to a previous state, review changes made over time, see who last modified something that might be causing a problem, who introduced an issue and when, and more.
- Repository:
  - A directory that contains your project work which are used to communicate with Git. Repositories can exist either locally on your computer or as a remote copy on another computer.

Version Control Terminology

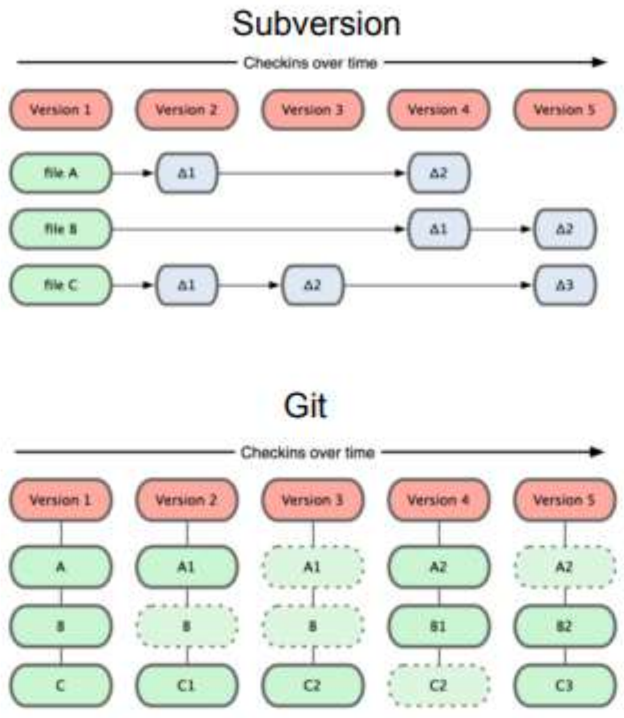
- Commit
  - Git thinks of its data like a set of snapshots of a mini file system.
  - Think of it as a save point during a video game.
- SHA
  - A SHA is basically an ID number for each commit.
  - Ex. E2adf8ae3e2e4ed40add75cc44cf9d0a869afeb6
- Branch
  - A branch is when a new line of development is created that diverges from the main line of development. This alternative line of development can continue without altering the main line.

Version Control Terminology

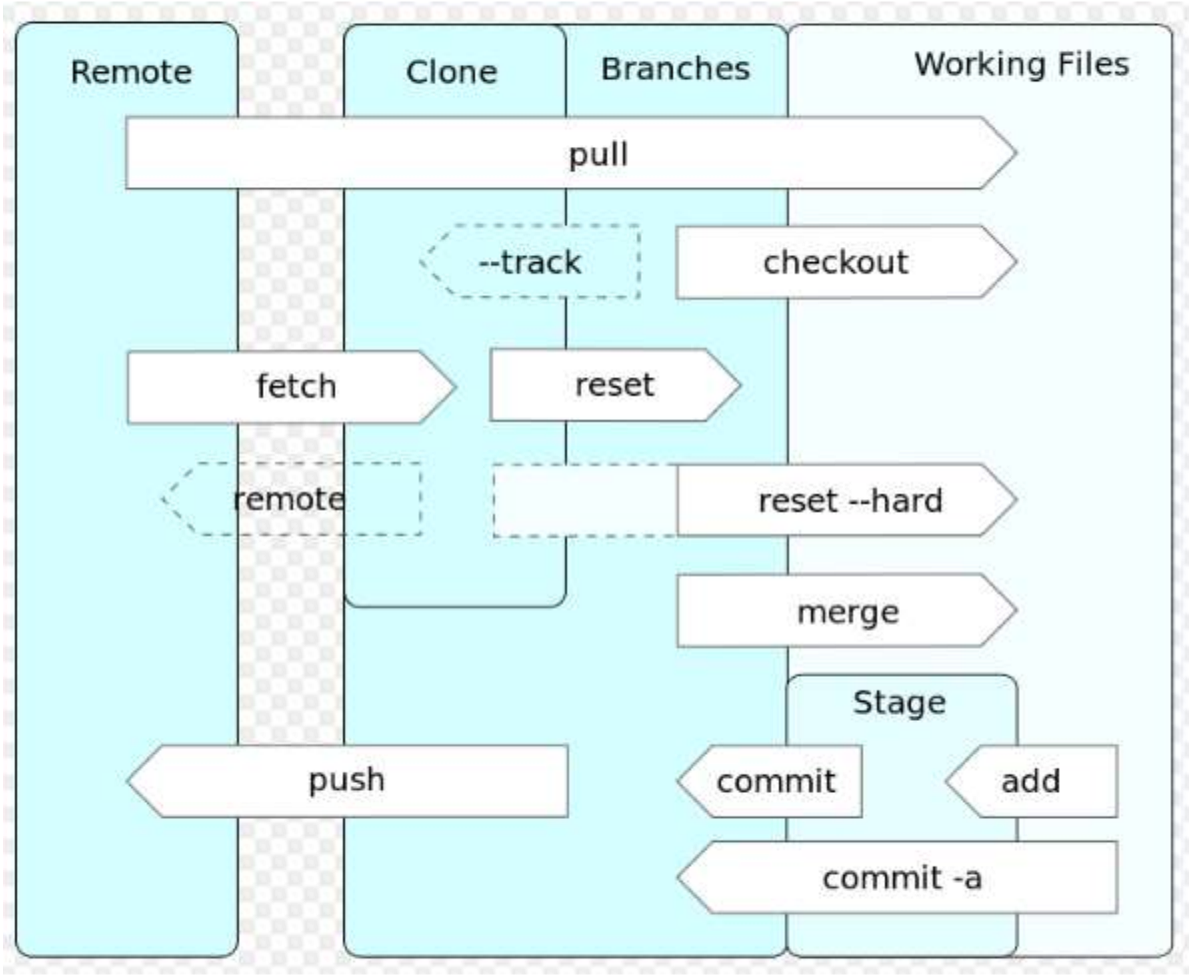
- Working Directory
  - files that you see in your computer's file system. When you open project files up on a code editor, you're working with files in the Working Directory.
- Checkout
  - Content in the repository has been copied to the Working Directory. Possible to checkout many things from a repository; a file, a commit, a branch, etc.
- Staging Area
  - You can think of the staging area as a prep table where Git will take the next commit. Files on the Staging Index are poised to be added to the repo

Git

- Centralized VCS like Subversion track version data on each individual file.
- Git keeps "snapshots" of the entire state of the project.
  - Each checkin version of the overall code has a copy of each file in it.
  - Some files change on a given checkin, some do not.
  - More redundancy, but faster.

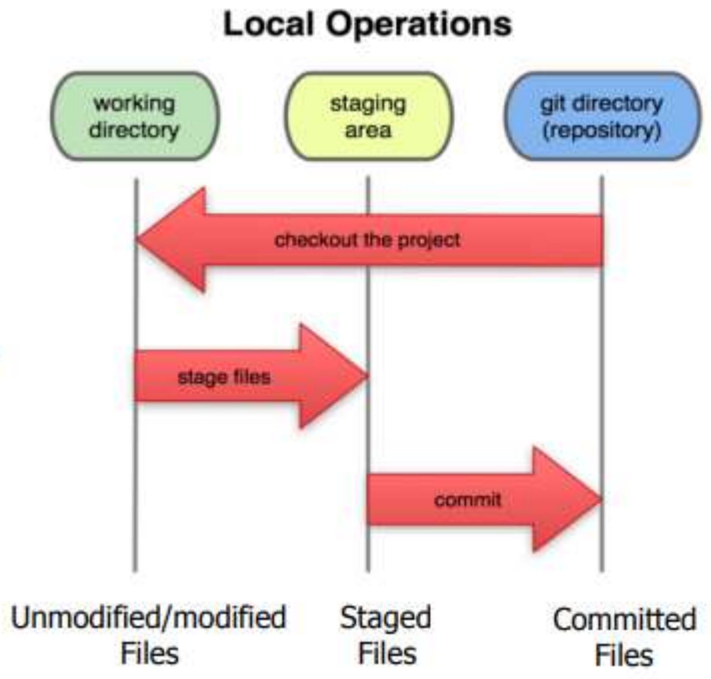


Git

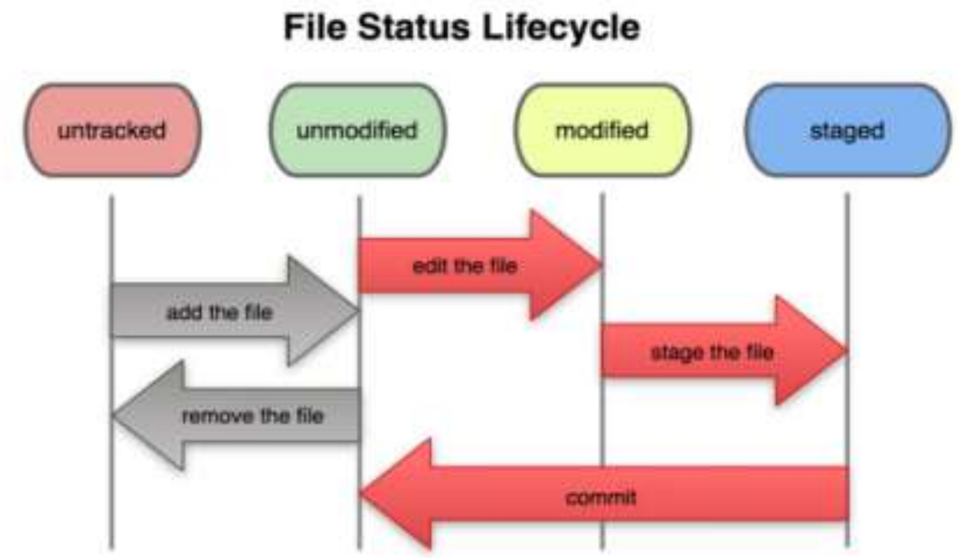

Remote
Clone
Branches
Working Files
pull
--track
checkout
fetch
reset
remote
reset --hard
merge
Stage
push
commit
add
commit -a

In your local copy on git, files can be:

- In your local repo
  - (committed)
- Checked out and modified, but not yet committed
  - (working copy)
- Or, in-between, in a "staging" area
  - Staged files are ready to be committed.
  - A commit saves a snapshot of all staged state.



Basic Git Workflow



- Modify files in your working directory.
- Stage files, adding snapshots of them to your staging area.
- Commit, which takes the files in the staging area and stores that snapshot permanently to your Git directory.

fands™

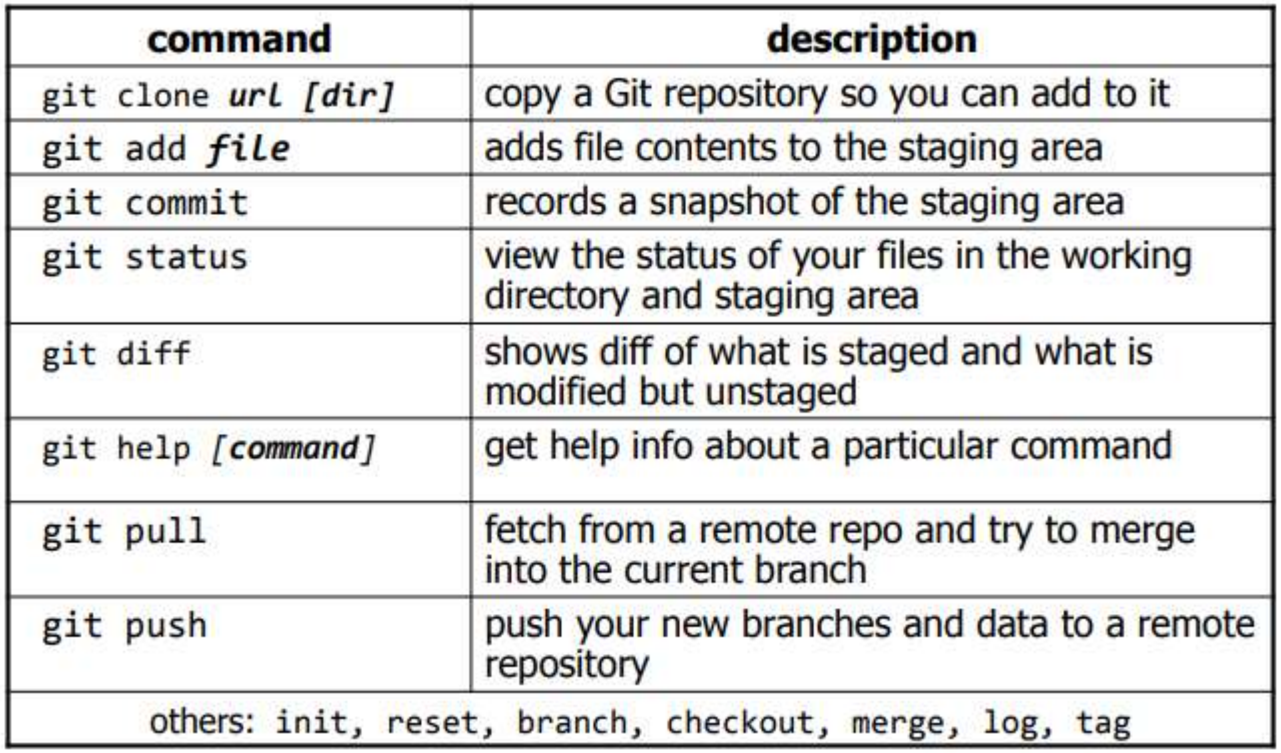
Initial Git configuration

- Set the name and email for Git to use when you commit:
  - git config --global user.name “.."
  - git config --global user.email e@gmail.com
  - You can call git config –list to verify these are set.
- Set the editor that is used for writing commit messages:
  - git config --global core.editor nano
    - (it is vim by default)

Creating a Git Repo

- To create a new local Git repo in your current directory:
  - git init
    - This will create a .git directory in your current directory.
    - Then you can commit files in that directory into the repo.
  - git add filename
  - git commit –m "commit message"
- To clone a remote repo to your current directory:
  - git clone url localDirectoryName
    - This will create the given local directory, containing a working copy of the files from the repo, and a .git directory (used to hold the staging area and your local repo)

Git Commands



command	description
`git clone url [dir]`	copy a Git repository so you can add to it
`git add file`	adds file contents to the staging area
`git commit`	records a snapshot of the staging area
`git status`	view the status of your files in the working directory and staging area
`git diff`	shows diff of what is staged and what is modified but unstaged
`git help [command]`	get help info about a particular command
`git pull`	fetch from a remote repo and try to merge into the current branch
`git push`	push your new branches and data to a remote repository
others: `init, reset, branch, checkout, merge, log, tag`	

Add and commit a file

- The first time we ask a file to be tracked, and every time before we commit a file, we must add it to the staging area:
  - git add Hello.java Goodbye.java
    - Takes a snapshot of these files, adds them to the staging area.
- To move staged changes into the repo, we commit:
  - git commit –m "Fixing bug #22"
- To undo changes on a file before you have committed it:
  - git reset HEAD -- filename (unstages the file)
  - git checkout -- filename (undoes your changes)
  - All these commands are acting on your local version of repo.

Viewing/undoing changes

- To view status of files in working directory and staging area:
  - git status or git status –s (short version)
- To see what is modified but unstaged:
  - git diff
- To see a list of staged changes:
  - git diff –cached
- To see a list of stages vs history
  - Git diff --staged
- To see a log of all changes in your local repo:
  - git log or git log --oneline (shorter version)
  - git log -5 (to show only the 5 most recent updates)

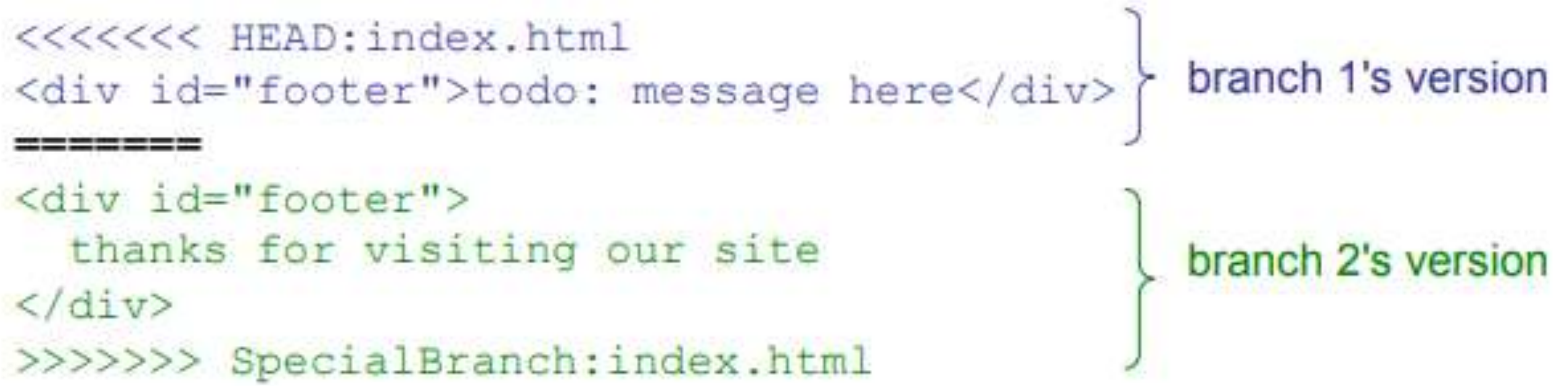
Branching and Merging

Git uses branching heavily to switch between multiple tasks.

- To create a new local branch:
  - git branch name
- To list all local branches: (* = current branch)
  - git branch
- To switch to a given local branch:
  - git checkout branchname
- To merge changes from a branch into the local master:
  - git checkout master
  - git merge branchname

Merge Conflicts

The conflicting file will contain <<< and >>> sections to indicate where Git was unable to resolve a conflict:



```
<<<<<<< HEAD:index.html
<div id="footer">todo: message here</div>
=======
<div id="footer">
  thanks for visiting our site
</div>
>>>>>>> SpecialBranch:index.html
```

Find all such sections, and edit them to the proper state (whichever of the two versions is newer / better / more correct).

Interaction with Remote Repo

- Push your local changes to the remote repo.
- Pull from remote repo to get most recent changes.
  - (fix conflicts if necessary, add/commit them to your local repo)
- To fetch the most recent updates from the remote repo into your local repo, and put them into your working directory:
  - git pull origin master
- To put your changes from your local repo in the remote repo:
  - git push origin master

GitHub

- GitHub.com is a site for online storage of Git repositories.
  - You can create a remote repo there and push code to it.
  - Many open source projects use it, such as the Linux kernel.
  - You can get free space for open source projects, or you can pay for private projects.

Git Protocols

- Git can use four distinct protocols to transfer data
  - Local
    - The most basic is the Local protocol, in which the remote repository is in another directory on the same host
  - HTTP
    - Git can communicate over HTTP using two different modes (Smart/Dumb)
  - Secure Shell (SSH)
    - A common transport protocol for Git when self-hosting is over SSH.
  - Git
    - This is a special daemon that comes packaged with Git; it listens on a dedicated port (9418) that provides a service similar to the SSH protocol, but with absolutely no authentication

fands™

Git Repository

- Repositories in GIT contain a collection of files of various different versions of a Project.
- These files are imported from the repository into the local server of the user for further updations and modifications in the content of the file.
- A VCS is used to create these versions and store them in a specific place termed as a repository.

Repository Types

- Bare Repositories
  - These repositories are used to share the changes that are done by different developers. A user is not allowed to modify this repository or create a new version for this repository based on the modifications done.
- Non-bare Repositories
  - Non-bare repositories are user-friendly and hence allow the user to create new modifications of files and also create new versions for the repositories. Cloning process by default creates a non-bare repository if any parameter is not specified during the clone operation.

Working Tree/Directory

- A working tree in a Git Repository is the collection of files which are originated form a certain version of the repository. It helps in tracking the changes done by a specific user on one version of the repository. Whenever an operation is committed by the user, Git will look only for the files which are present in the working area, and not all the modified files. Only the files which are present in the working area are considered for commit operation.
- The user of the working tree gets to change the files by modifying existing files and removing or creating files.

Stages of a working tree

- There are a few stages of a file in the working tree of a repository:
  - Untracked: In this stage, the Git repository is unable to track the file, which means that the file is never staged nor it is committed.
  - Tracked: When the Git repository tracks a file, which means the file is committed but is not staged in the working directory.
  - Staged: In this stage, the file is ready to be committed and is placed in the staging area waiting for the next commit.
  - Modified/Dirty: When the changes are made to the file i.e. the file is modified but the change is not yet staged.

Git Commands

Undo a working change

- Git checkout – filename
  - Replace the working with staging area file
  - Working changes will be lost
- Git checkout sha -- filename
  - From a specific commit
- Git reset Head filename
  - History to staging area

gitignore

- The purpose of gitignore files is to ensure that certain files not tracked by Git remain untracked.
- Create .gitignore file
  - *.media
  - Logs/*

Head

- HEAD
  - the current commit your repo is on. Most of the time HEAD points to the latest commit in your current branch, but that doesn't have to be the case. HEAD really just means "what is my repo currently pointing at".
  - In the event that the commit HEAD refers to is not the tip of any branch, this is called a "detached head".

Master and Origin

- Master
  - the name of the default branch that git creates for you when first creating a repo. In most cases, "master" means "the main branch".
- Origin
  - the default name that git gives to your main remote repo. Your box has its own repo, and you most likely push out to some remote repo that you and all your coworkers push to. That remote repo is almost always called origin, but it doesn't have to be.

Local and Remote Repo

- Commit - committing is the process which records changes in the repository. Think of it as a snapshot of the current status of the project. Commits are done locally.
- Push - pushing sends the recent commit history from your local repository up to GitHub. If you're the only one working on a repository, pushing is fairly simple. If there are others accessing the repository, you may need to pull before you can push.
- Pull - a pull grabs any changes from the GitHub repository and merges them into your local repository.
- Sync - syncing is like pulling, but instead of connecting to your GitHub copy of the forked repo, it goes back to the original repository and brings in any changes. Once you've synced your repository, you need to push those changes back to your GitHub account.

Concepts

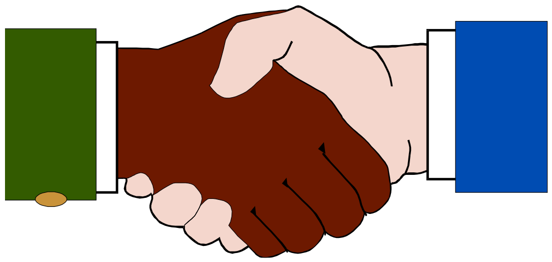
- Tag
- Patch
- Branching
- Merging
- Conflicts

QUESTION / ANSWERS

www.fandsindia.com

fands™

THANKING YOU !



www.fandsindia.com