An Introduction to Git Talk

Ehsan Zandi

Deutsche Telekom IT

ehsan.zandi@telekom.de

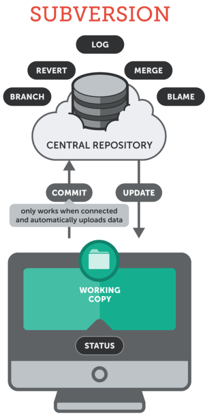
23.08.2021

Overview

Git vs SVN

- Git is a fully distributed version control system (VCS)
- Each user (PC/Laptop) is an exact clone of the remote repository
  - Each user is a repository (log, revert, merge, branch, etc)
  - No network connection required, except to sync with central repo (pull/push/fetch)
  - merge and rebasing can be done offline
- Git is much faster than SVN
- Git's repositories are much smaller than SVN
- Git's branches are much simpler and less resource heavy than SVN
- Git is much better in branch auditing and merge handling
- As many backups as the number of users ()
- Content integrity using SHA-1 hash

Git vs SVN



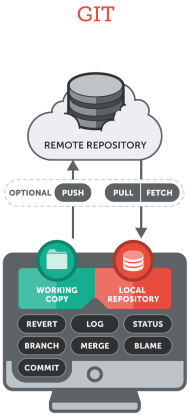


Figure : Centralized vs distributed VCS (Source: www.git-tower.com)

Git vs SVN

	SVN	Git
License	Open-source (Apache)	GNU
Distributed-ness	Centralized	Fully Distributed
Speed	✗	✓
Storage	✗	✓
Integrity Guarantee	✗	✓
Brnaching & merging	✗	✓
Stashing	✗	✓

Git Basics

Architecture

- Remote: The central repo (on a host machine/server, e.g., Github or Gitlab) $\rightarrow$ is identified by the alias "origin"
- Repository: The local repo (.git sub-directory inside your working directory), created by "git init" or "git clone", i.e., ceartion/clonining
- Index or staging area: State between the working directory and repository (after modifying and before commiting)
- Workspace or working directory: your local machine, including all directories, sub-directories, and files of your project

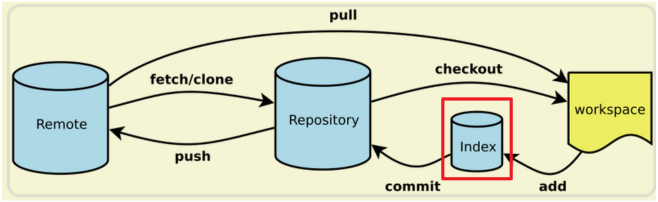


Figure : Git architecture (Source: www.stackoverflow.com)

Git Basics
Definitions

- **origin**: A shorthand name for the remote repo

  $git remote show (shows "origin" as output)

  $git remote show origin (shows detailed info on origin)
- **branch**: A movable pointer to a commit
- **master (or sometimes main)**: Default name of the (first) branch: can be changed
- **HEAD**: A special pointer that tells on (the tip of) which branch you are.
- **origin/HEAD**: A special pointer that tells on which branch the remote repo is.

Git Basics

Add/Commit

- **git add**: To add a new file or modified into the staging (index) area. It makes the changes ready for commiting.

  $git add FILE_NAME

  $git add . (adds all the changes current directory and sub-directories)

- **git commit**: To put the staged files into the (local) repo. Such changes can be tracked, i.e., revert, log, etc.

  $git commit -m "A proper message"

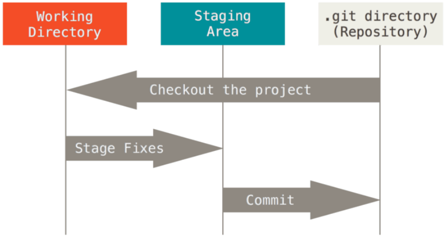


Figure : Git areas (Source: https://git-scm.com)

Git Basics

Initializing a repo

- Creating a local repo (without any remote)

  $git init (creates .git sub-directory)

  $echo "hello world." >> firstFile.txt (makes changes in working area)

  $git status (You see that your commit has some hash value)

  $git add firstFile.txt (puts your changes into staging area)

  $git status (You see that your commit has some hash value)

  $git commit -m "A proper message" (Now you have your first commit on the default branch master)

  - Hint: git commit -am "A proper message" (combines "git add" and "git commit")

  $git status (A clean repo and one commit with a hash value)

  $git branch -m master main (renames the branch master to main)

  $git remote (Output is empty since there is no remote repo)

Git Basics

Status and Log

- Status and log

  $git status (Shows the status of the repo)

  $git log (Shows the commit log on the current branch)

  $git log SOME_BRNACH (Shows the commit log on a specific branch)

  $git log --all (Shows the commit log on all branches)

  $git log -p (Shows the commit log and the content difference of files per commit, combines git log and git diff)

  $git log --decorate --oneline --graph --all (Very useful graph-like history)

Git Basics

Aliases

- Git Aliases, some useful examples:

  $git config --global alias.g 'log --decorate --oneline --graph --all' (makes "git g" an alias for the previous long command)

  $git config --global alias.l log (makes "git l" an alias for "git log")

  $git config --global alias.loa 'log --oneline --all' (makes "git loa" an alias for "git log --oneline --all")

  $git config --global alias.s status (makes "git s" an alias for "git status")

  $git config --global alias.b status (makes "git b" an alias for "git branch")

  $git config --global alias.ch checkout (makes "git ch" an alias for "git checkout")

Git Basics

Difference

- Comparing files on the same branch

  $git diff (shows the difference between working and staging area for all files$\rightarrow$ *tobestaged*, *i*.*e*., *gitadd*)

  $git diff SOME_FILE (shows the difference between working and staging area for a given file)

  $git --staged diff (shows the difference between staging area and last commit for all files$\rightarrow$ *tobecommited*)

  $git --cached diff (the same as above)

  $git diff HEAD (combines "git diff" and "git diff staged")

- Comparing files between two branches

  $git branch BRANCH_A..BRANCH_B (compares all files)

  $git branch BRANCH_A..BRANCH_B SOME_FILE (compares only a given files)

Undoing Changes

Checkout

- Go back to some specific commit

  `$git checkout 53c5105` (8 first digits out 40 long hexadecimal digit HASH-1)

Branches in Git

Creating, Displaying, and Switching

- Creating a branch

  $git branch NEW_BRANCH (creates a new branch)

  $git checkout -b BRANCH_NAME (creates and switch)

  $git switch -c BRANCH_NAME (creates and switch, from Git 2.23)

- Displaying branches

  $git branch (shows only local branches)

  $git branch -r (shows only remote branches)

  $git branch -a (shows all branches)

- Switching between branches

  $git checkout BRANCH_NAME (switches to another branch)

  $git switch BRANCH_NAME (switches to another branch)

Branches in Git

Comparing, Merging, Renaming, and Deleting

- Comparing two branches

  $git branch BRANCH_A..BRANCH_B (compares all files)

  $git branch BRANCH_A..BRANCH_B SOME_FILE (compares only a given files)
- Merging branches

  $git merge BRANCH_B (merges branch b into branch a, you should be in branch a)

  $git merge BRANCH_B BRNACH_A (does not matter on which branch you are)
- Renaming a branch

  $git branch -m OLD_NAME NEW_NAME
- Deleting a branch

  $git branch -d BRANCH_FOR_DELETION (deletes a branch)