Git HUB

Git is a distributed revision control and source code management system with an emphasis on speed.

Version control system.

Version control system (VCS) is a software that helps developers to work together and maintain a complete history of their work.

functions of VCS –

* + Allow developers to work simultaneously
  + Does not allow overwriting each other’s changes.
  + maintains a history of every version.

Types of VCS –

* + Centralized version control system (CVCS).
  + Distributed/Decentralized version control system (DVCS).

Distributed Version Control System.

CVCS used a central server, if the central server goes down for an hour, then during that hour, no can collaborate at all.

DVCS clients not only check out the latest snapshot of the directory but they also fully mirror the repository. If the server goes down, then the repository from any client can be copied back to the server to restore it. Every checkout is the full backup of the repo.

Git does not rely on the central server and that is why you can perform many operations when you are offline. You can commit changes, create braches, view logs, and perform other operations when you are offline. You require network connection only to publish the changes and take the latest changes.

Advantages:

Free and open source.

Fast and small – as the operations are performed locally.

Easy branching, security, no need of powerful hardware.

Terminologies:

Local repo – Git provides a private copy of the whole repo, users can perform a whole lot of operations like add file, remove file, rename file, move file, commit changes, and many more.

Working directory and staging area or index.

The working directory is the place where files are checked out. In other CVCS, developers generally make modifications and commit their changes directly to the repository. But Git uses a different strategy. Git doesn’t track each and every modified file. Whenever you do commit an operation, Git looks for the files present in the staging area. Only those files present in the staging area are considered for commit and not all the modified files.

Blobs

Blob stands for Binary Large Object. Each version of a file is represented by blob. A blob holds the file data but doesn’t contain any metadata about the file. It is a binary file, and in Git database, it is named as SHA1 hash of that file. In Git, files are not addressed by names. Everything is content-addressed.

Trees

Tree is an object, which represents a directory. It hold blobs as well as other sub-directories. A tree is a binary file that stores references to blobs and trees which are also named as SHA1 hash of the tree object.

Commit

Commit holds the current state of the repository. A commit is also named by SHA1 hash. You can consider a commit object as a node of the linked list. Every commit object has a pointer to the parent commit object. From a given commit, you can traverse back by looking at the parent pointer to view the history of the commit. If a commit has multiple parent commits, then that particular commit has been created by merging two branches.

Branched

Branches are used to create another line of development. By default, Git has a master branch, which is same as trunk in Subversion. Usually, a branch is created to work on a new feature. Once the feature is created, it is merged back with the master branch and we delete the branch. Every branch is referenced by HEAD< which points to the latest commit in the branch. Whenever you make a commit, HEAD is updated with the latest commit.

Tags

Tag assigns a meaningful name with a specific version in the repository. Tags are very similar to branches, but the difference is that tags are immutable. It means, tag is a branch, which nobody intends to modify. Once a tag is created for a particular commit, even if you create a new commit, it will not be updated. Usually, developers create tag for product releases.

Clone

Clone operation creates the instance of the repository. Clone operation not only checks out the working copy, but it also mirrors the complete repository. Users can perform many operation with this local repository. The only time networking gets involved is when the repository instances are being synchronized.

Pull

Pull operation copies the changes from a remote repository instance to a local one. The pull operation is used for synchronization between two repository instances. This is the same as the update operation in Subversion.

Push

Push operation copies changes from a local repository instance to a remote one. This is used to store the changes permanently into the Git repository. This is same as the commit operation in Subversion.

HEAD

HEAD is a pointer, which always points to the latest commit in the branch. Whenever you make a commit, HEAD is updated with the latest commit. The heads of the branches are stores in .git/refs/heads/ directory.

Revision

Revision represents the version of the source code. Revisions in Git are represented by commits. These commits are identified by SHA1 secure hashes.

URL

URL represents the location of GIT repository. Git URL is stored in config file.

Basic configuration

Setting username

git config --global user.name “nkumar”

setting email id

git config --global user.email “[kumarnaveen.pujar@gmail.com](mailto:kumarnaveen.pujar@gmail.com)”

Avoid merge commits for pulling

you pull the latest changes from a remote repository, and if these changes are divergent, then by default Git merge commits. We can avoid this via following settings.

git config --global branch.autosetuprebase always

Color highlighting

git config --global color.ui true

git config --global color.status auto

git config --global color.branch auto

Setting default editor

git config --global core.editor vim

setting default merge tool

Git does not provide a default merge tool for integrating conflicating changes into your working tree. We can set default merge tool by below.

git config --global merge.tool vimdiff

Listing git settings

git config--list