GIT Reference Manual

The background behind GIT is that a software developing organization was using a commercial product named Bit Keeper. Unfortunately they lost their license to the product. After careful consideration on alternative products to use to manage the versioning activities, they decided to create their own version software system. That application became GIT. GIT was created by Linus Benedict Torvalds, the chief software engineering architect of the Linux Operating System.

GIT has all of the attributes of a distributed version control system. The GIT distributed version control system works in two modes. It works both in Client Mode and in Central Distributed Mode. One advantage of GIT is that it has a small footprint. What this means is that it does not require a lot of physical storage space and it can function with limited or no available internet bandwidth. One advantage of a distributed version control system that works in both of these modes are that it works well in resource constrained environments. A user of GIT can work in client mode when internet resources are limited or not available. The distributed version control system and the client systems are small programs that work cohesively together. Later when internet resources are available, a user can request to have their changes merged with the central repository.

As this product evolves the updates first appear on the Linux platform and the goes to Mac and Windows. As the updates arrive, they usually distribute down the chain of platforms within days.

GIT is designed as a command line tool. This means that as a user, you will be more affective knowing and understanding some of the primary GIT commands.

One fundamental command is the command to display the current version of GIT installed on the local client machine. To display the current version type in the following

git --version

It is recommended that you have a version 1.8 or higher

GIT organizes its repository of documents that can be versioned as projects. To create a local repository as a project type in the following command on the GIT command line.

git init project name

Typing in this command will create a local repository on the client machine with the project name type in. This commands creates a folder in your system with the same name as the project name. The first sub folder inside the project folder will always have the name GIT. The GIT folder inside the project folder will contain all of the documents and folders for the projects. Additionally, it will contain all of the versioning states and tracking information for all of the documents in the project.

To test if the GIT client project is connect properly to the project folder, create and save a file to the project folder. The file or folder can be created from any type of software application such as Microsoft Word, Notepad ++, NetBeans. Once the file or folder has been placed in the project folder, return to the GIT Bash command line. At the GIT command line type in the command to return the current status of the project/repository. When a new document is added to a project or modified within the project, the GIT status command should designate the file as being untracked. To obtain the status of the current project that GIT is monitoring type in the following command.

git status

After typing in this command into the GIT command line, GIT will return list the currently untacked files in the current local repository along with a list of suggested actions that the user should take to bring the repository up to date with the recent changes.

The next step will be to issue the GIT ADD command to make the suggestion to the GIT tracking system to have file participate in the next transactional change set.

It is customary to run the GIT STATUS command to visually confirm that the file or files have been added to the next transactional change set.

git status

After executing the GIT STATUS command the GIT versioning system will return the next round of recommend actions to the user. This should now include the recommendation to COMMIT the latest changes as permanent members of the repository. Here is the recommended syntax for the COMMIT command

git commit –m "A message tagging the contents and purpose of the additions or changes"

After executing the COMMIT command at the GIT command line, the files and folders that were added to the current transactional change set are made permanent changes to the current active repository. The command modifier –m permits the user to document and tag the changes with a brief message in a statement summarizing the impact of the changes to the project.

Once the changes are committed to the local client repository, it should be also updated to the HITHUB central distributed repository. This is accomplished by PUSHING the local client changes to the central distributed repository hosted at GITHUB.

git push –u origin master

Executing this statement a GIT command line transfers the modified version of the local client repository the centralized distributed repository. Under some configurations of the centralized distributed repository this may require approval by a user of the centralized repository assigned the role of authorizing changes. However, in any configuration, the PUSH process will require the client user to provide a valid User ID and Password to PUSH the new version of the project to the centralized GITHUB for permanent change or for approval to be added as a permanent change.

If for some reason the GITHUB repository and the GIT CLIENT repository are not aligned and synchronized properly, use the following command to align them

git remote remove origin

This command will initialized the default origin location of the centralized GITHUB repository. The user will be prompted for the URL of the GITHUB repository after this command is executed.

Once the centralized repository is created you may share it with other people. To permit other people to read the contents of the repository simply send the URL of the repository to the other recipients

Once the recipients receive the URL, they visit the webpage at the URL. Once at the webpage of the project, they are able to take a variety of actions. They are able to download the project as a zipped file, they are able to Clone the project to their GIT client project. If the user plans are participating a collaborative effort on the project, the appropriate choice will be to Clone the project to their own Client version of the project. Once the user has a Clone copy of the project, they are able to start creating Branches of the project. Each branch is a representation of a different version of the project.

Once a project has been Forked and sub-versioned, all modifications made to that subversion of the client will only occur locally, not at the central distributed repository level. This forked local client subversion of the project is in the user's personal local client account. After the user makes their own personal changes to the project in their subversion of it, the user can then offer their changes to the administrators of the project for review. If anyone designated as a project administrator at the HITHUB central distributed repository level determines that the changes are worthy to be merged with the core project, they may merge the offered changes into the new branch of the project.

Before this can occur, the user at the client level issues a full request from the administrators of the project at the GITHUB level. The request will then go the users of the project at the HITHUB level that have the authority to merge and pull the request into the core project.

To create a new branch at the client level, type in the following command

git branch branch name

This command will create a duplicate state of the project under the branch name specified

To link to a specified state of the project use the following command at the GIT client command line

git checkout branch name

The CHECKOUT command toggles to the specific branch name specified after the reserve word CHECKOUT in the statement. You can check out any branch at any level of the project using the CHECKOUT statement in GIT

Arduino as a engagement tool for women in CS. Designing electronic jewelry.