Git and GitHub Training

by Abhishek Luv

Things required for this Training?

- Windows Laptop

- Msysgit.exe
- Git Extension
- GitHub for Windows

What is Version Control?

What is version control, and why should you care? Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later.

History of Git

- birth in 2005 by Linus Torvalds
- For Linux kernel maintenance

What is a repository

A repository is simply a database containing all the information needed to retain and manage the revisions and history of a project.

Within a repository, Git maintains two primary data structures, the object store and the index.

The object store is designed to be efficiently copied during a clone operation as part of the mechanism that supports a fully DVCS.

Index is transitory information, is private to a repository and can be created or modified on demand as needed.

Git Object Store

It contains your original data files and all the log messages, author information, dates, and other information required to rebuild any revision or branch of the project.

There are four types of Object stores

- Blobs: A blob holds a file’s data but does not contain any metadata about the file or even its name.
- Trees: Its records blob identifiers, path names and a bit of metadata for all the files in one directory. It can also recursively reference other subtrees.
- Commits: A commits object holds metadata for each change introduced into the repository, including the author, committer, commit data, and log message.
- Tags: A tag object assigns an arbitrary human readable name to a specific object usually a commit.

Index

The index is a temporary and dynamic binary file that describes the directory structure of the entire repository.

Index captures a version of the project’s overall structure at some moment in time. As the developer, we execute Git commands to stage changes in the index.

Changes usually add, delete or edit some files or set of files. The index records and retains those changes, keeping them safe until you are ready to commit them. You can also remove or replace changes in the index.

Content-Addressable Names

The Git object store is organized and implemented as a content-addressable storage system. Each object in the object store has a unique name produced by applying SHA1 to the contents of the object, yielding an SHA1 hash value. Any tiny change to a file causes the SHA1 hash to change, causing the new version of the file to be indexed separately.

Git is a content-addressable filesystem.

What does that mean?

It means that at the core of Git is a simple key-value data store.

You can insert any kind of content into it, and it will give you back a key that you can use to retrieve the content again at any time.

SHA1 values are 160-bit value that are usually represented as a 40 digit hexadecimal number.

For example: `937082042f48a5cbc7777634509310fff059bc19`

File Management and The Index

Remote Repository(master) <-> Index(Staging happens here) <-> Local Repository (local)

More on Index

Git’s Index doesn’t contain any file content: it simply tracks what you want to commit when you run git commit.

Git checks the index rather than your working directory to discover what to commit.

You can query the state of the index at any time with the command git status

File Classification in Git

- Tracked : Any file that’s already there in the repository or any file that is staged
- Ignored : file to be ignored by git object store and git index
- Untracked : An untracked file is any file not found in either of the previous two categories

Installing Git on Windows

- msysgit.exe

- git extensions
- GitHub for windows

Git Command Line Commands

add, bisect, branch, checkout,clone,commit,diff,fetch,grep,init,log,merge,mv,pull,push,rebase,r
tag

Git Version Check

`git --version`

Git needs to know you before you can use it

`git config user.name "abhishekluv"`

`git config user.email "abhishekluv@gmail.com"`

`git config --list`

Global commit author details

`git config --global user.name "abhishekluv"`

`git config --global user.email "abhishekluv@gmail.com"`

Creating an initial repository

`git init`

Adding file to your repository

`git add index.html`

Git status

`git status`

Git Commit

`git commit -m "commit message"`

`git commit --message="commit message"`

Viewing your commits

`git log`

Show your commit

`git show commitSHAnumber`

Removing a file in your

repository

`git rm index.html`

Renaming a file in your repository

`git mv index.html index2.html`

Using git add

Example

`git add .gitignore`

`git add --all`

`git status`

using git commit -all

`git commit -a`

`git commit --all`

causes it to automatically stage all unstaged, tracked file changes - including removal of tracked files from the working copy before it performs the commit.

using git rm

```
git rm index.html
```

will remove the file completely from the working directory as well as the remote repo.

```
git rm --cached index.html
```

will remove the file from the index and leave it in the working directory.

.gitignore file

Add any file name in the .gitignore file which you want to ignore.

Branches

A branch is the fundamental means of launching a separate line of development within a software project.

Git allows many branches and thus many different lines of development within a repository.

Creating a branch

```
git branch branchname
```

list branches

`git branch`

list of branches with details

`git show-branch`

changing/checking out branches

`git checkout branchname`

creating and checking out a new branch

`git checkout -b branchname`

deleting a branch

`git branch -d branchname`

Merges

what is git merge

In Git, a merge must occur within a single repository i.e. all the branches to be merged must be present in the same repository.

`git checkout branch`

`git merge other_branch`

`git commit -a -m "commit message"`

How to generate SSH Key for GitHub

`$ cd ~/.ssh`

`$ ssh-keygen -t rsa -C "your_email@example.com"`

to copy the RSA key to the clipboard

`$ clip < ~/.ssh/id_rsa.pub`

Steps

1. Register at www.github.com
2. Create a new public repository at www.github.com
3. Download and Install msysgit from here
4. Create a folder for your new local repository
5. Open Command-Prompt or PowerShell
6. Run command `git --version`
7. Run command to initialize a new local git repository `git init`
8. Now go back to www.github.com and copy the remote repository

HTTPS link onto your clipboard

9. Run command to add the remote repository as an origin `git remote add origin remoterepoURL`
10. Run command to cross check the origin `git remote -v`
11. Create a new file readme.txt
12. Run command to check the status of the index `git status` tracked, untracked, ignored
13. Run command to add new file for staging `git add readme.txt` and check status `git status`
14. Run command to unstage the file `git rm --cached readme.txt`
15. Run command to commit the changes `git commit -m "commit message"`
16. Run command to push the changes to remote repository `git push origin master`
17. Create a new file gitignore.txt file in your local repository
18. Run command to rename gitignore.txt to .gitignore `ren gitignore.txt .gitignore`
19. Run command to add all the file for staging `git add --all`
20. Run command to reset on step back `git reset HEAD`
21. Run command to create and checkout a branch `git checkout -b develop`
22. Run command to push changes to new branch `git push origin develop`
23. Run command to switch back to master branch `git checkout master`
24. Run command to merge the develop branch into the master branch `git merge develop`
25. Run command to publish local commits after merging `git push`

`origin master`

Git References

Head

The question now is, when you run git branch (branchname), how does Git know the SHA-1 of the last commit? The answer is the HEAD file. The HEAD file is a symbolic reference to the branch you're currently on. By symbolic reference, I mean that unlike a normal reference, it doesn't generally contain a SHA-1 value but rather a pointer to another reference.

Remotes

If you add a remote and push to it, Git stores the value you last pushed to that remote for each branch in the refs/remotes directory. For instance, you can add a remote called origin and push your master branch to it.

Pack files

The initial format in which Git saves objects on disk is called a loose object format. However, occasionally Git packs up several of these objects into a single binary file called a pack file in order to save space and be more efficient.

Maintenance

Occasionally, you may have to do some clean-up — make a repository more compact, clean up an imported repository.

Occasionally, Git automatically runs a command called “auto gc”. Most of the time, this command does nothing. However, if there are too many loose ob-jects (objects not in a pack file) or too many packfiles, Git launches a full-fledged git gc command. The gc stands for garbage collect, and the command does a number of things: it gathers up all the loose objects and places them in pack files, it consolidates pack files into one big pack file, and it removes objects that aren’t reachable from any commit and are a few months old.

You can run auto gc manually as follows:

```
git gc --auto
```

Again, this generally does nothing. You must have around 7,000 loose ob-jects or more than 50 pack files for Git to fire up a real gc command.