Section 18: Git Behind The Scenes -Hashing & Object

168. Git As A Key-Value Datastore

* GIT is a key value store. GIT uses a database to store file artifacts. This database is a key value store and is the heart of GIT Software Code Management. Files, directories and commit messages become archived to this store.

169. Hashing With Git Hash-Object

* Git uses hashes in two important ways. When you commit a file into your repository, Git calculates and remembers the hash of the contents of the file.
* When you later retrieve the file, Git can verify that the hash of the data being retrieved exactly matches the hash that was computed when it was stored.
* " git hash-object <file or text> " this command is used for git hash.

170. Retrieving Data With Git Cat-File

* cat-file command is used to provide content or type and size information for repository objects.
* " git cat-file -p <object-hash> " is used to retrieve data from already hashed file.

171. Deep Dive Into Git Objects: Blobs

* blobs are objects type git uses to store the content of file in a given repository. blob dont even include the filename of each file or any other data. They just store the content of a file.

172. Deep Dive Into Git Objects: Trees

* Tree are git objects used to store the content of a directory. Each tree contains pointer that can refer to blobs and to other trees.
* Each entry in a tree contains the SHA-1 hash of a blob or tree, as well as the mode, type, and filename.
* " git cat-file -p master^{tree} " is used to view tree.

Section 19: The Power of Reflogs - Retrieving "Lost" Work

175. Introducing Reflogs

* The git reflog command is used for Git to record updates made to the tip of branches.
* It allows to return to commits even to the ones that are not referenced by any branch or any tag.
* After rewriting history, the reflog includes information about the previous state of branches and makes it possible to go back to that state if needed.

176. The Limitations of Reflogs

* Reflog are local activity
* It expires after 90 days.

177. The Git Reflog Show Command

* "git reflog show HEAD" command is used to show reflog outputs.

178. Passing Reflog References Around

* By default, git reflog outputs the reflog of the HEAD reference, which is considered to be an iconic reference to the branch, which is currently active.

179. Time-Based Reflog Qualifiers

* The “expire” subcommand prunes older reflog entries. Entries older than “expire” time, or entries older than “expire-unreachable” time and not reachable from the current tip, are removed from the reflog. This is typically not used directly by end users

180. Rescuing Lost Commits With Reflog

* Git logs your actions in the Reflog which makes it a valuable logbook and a good starting point when something went wrong.

Section 20: Writing Custom Git Aliases

183. The Global Git Config File

* The system level configuration file lives in a gitconfig file off the system root path.

184. Writing Our First Git Alias

* Git aliases provide a way to shorten popular git commands.
* " git config –global alias.<custom\_command\_name> <original\_command Name> " this command is used for creating git aliases.

185. Setting Aliases From The Command Line

* The simplest way to add a git alias is by running a command to add the alias to the git global configuration file.
* " git config --global alias" command is used to set alias.

187. Exploring Existing Useful Aliases Online

* "git st" command will show the status on clean branch. Included it should be initialized first.