
- Show All Files: defaults write com.apple.finder AppleShowAllFiles YES

Creating a New Repo

- Go into folder
- Initiate Git: git init
- Add a file from the folder to repo: git add README.md
- Commit changes: git commit -m "readme file"
- Creating a repo on GitHub.com
- Add files to existing repo: git remote add origin https://github.com/kavisek/GitTest.git
- Push the current branch and set the remote as upstream: git push --set-upstream origin master
- Push Changes to Master: git push -u origin master

Configuration

- Configure Git Config Files
    - System: gi config --system
    - User; git config --global
    - Project: git config
- Check git version: git --version

- view all hidden files in directory: ls -la
- Return to Home Folder: cd ~
- Configure username: git config user.name "kavisek"
- Configure email: git config user.email "somewhere@nowhere.com"
- View Git Configurations: config —list
- "." Files are hidden files
- View Hidden File in Terminal:
    - cd .git
    - cat config
- Return to parent directory: cd -
- Show colours in git terminal logs git config color.ui true

Git Auto Completion

- Install Git Auto Completion in Repo: curl -OL https://github.com/git/git/raw/ master/contrib/completion/git-completion.bash
- Change file into hidden file: mv git-completion.bash .git-completion.bash
    - mv [old name] [new name]
- Edit bash profile file in Home directory with code

Git Help

- help with specific command: git help log
    - Hit "q" to get out
    - Same as Manual: man git-log

Getting Started

- Initialize Folder/Repo: git init
    - git init tracks all change
- Add Txt File to repo
- Track Changes in file: git add .
- Commit Changes to repo: git commit -m ‘file update’

Writing Commit Messages

- short singe line summary (50 characters)
- Enter Blank line
- Then a more complete description
- Keep additional line to less than 72 characters
- write commit messages in present tense
- you can include a tracking number/code tags, use firm standards
- No personal comments

Viewing Commit Messages

- When in repo use: git log
- Return last 3 commits: git log -n 3
- Return commits since 2012-06-14: git log --since=2012-06-14
- Return commits until 2012-06-14: git log --until=2012-06-14
- You can use any of these parameters together
- Return commits by author: git log --author=“kavisek”
- REGEXP commits search: git log —grep=“Init”

Two-Tree-Architecture:

- Git a three tree architecture
    - Repository
    - Staging Index
    - Working
- git add files from working to staging
- git commit file from staging to repo
- git checkout files from repo to working (no staging)

Git Commits

- During commits git creates a checksum for each change set
- Data Integrity is fundamental
- Git uses SHA-1 hash algorithm
- Commit ID is the SHA-1 ID
    - Its unique ID

Head Pointer

- pointer to a tip of the current branch of the repo
- where commit writing will take place
- last state of repository, what was last checked out
- Resource: https://www.lynda.com/Git-tutorials/Working-HEAD-pointer/100222/111269-4.html?autoplay=true
- The head metadata is included in the hidden “ref” folder

Adding Files to Git

- view difference between 3 branches: git status
- git add one file: git add second_file.txt

Viewing Commit Changes

- view commit changes between working and staging: git diff
    - Only show changes in the working directory
- view commit changes between staging and the repository: git diff —staged
- Push delete file changes from working to staging: git rm file_to_delete1.txt
    - Alternative: git add .

Moving and renaming files

- If you change the name in the OS, git tracks it as a file deletion and as a new file
- Rename file with Git: git mv second_file.txt secondary_file.txt
    - rename change is applied to staging right away
- both move and rename using same syntax: git mv
- move a file to a director: git mv third_file.txt first_directory/third_file.txt
- moving back in directory in terminal: cd ..

More Practise

- You file git log files via pages, use space bar to go to the next page
- View changes in one line: git diff --color-words contact.html
- Add and Commit Files: git commit -am ‘‘change support number"
    - Includes everything in working directory
    - Files not tracked are not included
    - Files being deleted are not included
- Add all change in subdirectory ‘‘tours’’ to staging: git add tours/
- Use 2(N) different commits for 2(N) different changes

Git Checkout

- we want the repo version of the file back
- Restore file from repo: git checkout — index.html

Upstaging Files

- onstage modified changes: git reset HEAD index.html

Ammending Commits

- You can only change the last commit: git —amend m ‘‘duplicate file’’

Restoring Versions

- You cannot undo commits past the last commit, its better to create a new commit

- You can checkout an old file from the repo instead not the staging environment:
    - You will need the shaw
    - google search the command

Reverting a Commit:

- You do a complete reversal of a old commit by using git revert using the SHA Hash
- revert a commit: git revert d906744ccf015de81166bae2b61680ea22894542

Git Reset

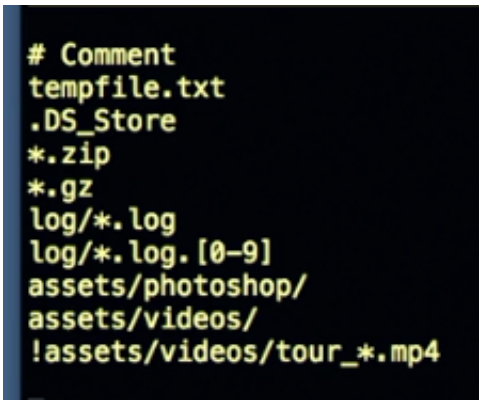
- changes the head pointer
- overwrite everything form the specified commit
- --soft
    - does not change staging index or working directory, only repository
- --mixed
    - does not change working directory, changes staging and repository
- --hard
    - reset working directory, staging, and repo
- Check repo branch name: cat .git/HEAD
- View Last Commit's HashY cat .git/refs/heads/master
- Git Soft Reset: git reset -- soft 7573d5aecda90451be8
- Git Mixed Reset: git reset -- mixed 7573d5aecda90451be8
- Git Hard Reset: git reset -- hard 7573d5aecda90451be8
    - 7573d5aecda90451be8 = HEAD
    - HEAD is the SHA HASH Number before the "c" in the HASH

Removing Junk Files

- List untracked files for removal: git clean -n
    - Untracked files are files that have never been tracked for staging
- Remove Junk Files: git clean -f

Ignore Files

- Create a hidden folder called ".gitignore": nano .gitignore
    - Write the name of file to ignore: temple.txt
        - Write one file per line



        - Write “*.txt” to ignore all text files
        - Your can write “# abcd” for comment in the git ignore file
    - Control + X to exist the terminal screen
    - Type “Y” for Yes
    - Press “Enter” to get past the second response
    - Next commit the “.gitignore" file to the repo
    - You cannot ignore folders, only files
- Git Ignore Repo on GitHub is a good resource

Ignore Files Globally

- Ignore files in all repos
- Setting not tracked in repo
- User-specific instead of repo specific

Stop Tracking a File

- Stop Tracking a File without Removing it: git rm —cached temple2.txt

Tracking Empty Directories

- Git does not track empty files
- Cheat: Put a little tiny file in the directory so we can track it
- create a file called .gitkeep
    - Create a file that dose not exist via unix: touch assets/pdfs/.gitkeep

Navigating The Commit Tree

- triage references something in the git tree
- reference the commit by the full SHA-1 hash (1 billion potentials hashs)
- or reference the commit by the short SHA-1 hash (8-10 characters)
- Head Points is another option
- branch reference is another option
- tag reference is another option
- ancestry is another option

- parent commit
    - - HEAD^, acf87504^, master^
    - - HEAD~1, HEAD ~
- grandparent commits
    - HEAD^^, acf87504^^, master^^
- great-grandparent commit
    - HEAD^^^, acf87504^^^, master^^^
    - HEAD~3

- List the current git tree: git ls-tree HEAD
- List the master git tree: git ls-tree master
- List the child git tree: git ls-tree explore_california/

- List the master git tree for previous commit: git ls-tree master^
- List the a tree by SHA-1: git ls-tree 3683c772c5586

- Tree Note
    - a blob is any type of file
    - a tree is any directory

More Options for Git Log:

- compressed log list: git log --oneline
    - alternative: git log --formate=online
    - full SHA is available in the alternative
- log of last 3 commits: git log --oneline -3
- logs since a date: git log --since="2012-06-20"
- logs until a date: git log --until="2012-06-20"
- logs until a date: git log --since="2 weeks ago" --until="2 days ago"
- logs by author: git log --author="Kevin"
- logs by string; git log --grep="temp"
- logs between commits: git log 2907d12..acf8750 --online
- logs since commits affecting a file: git log 2907d12.. index.html
    - to view the changes in the document use: git -p
    - view stats of the changes: git log --stat
        - alternative: git log --sum
- View a graph of commits: git log --graph
- A cool graph combination: git log --online --graph --all --decorate

Examining a Commit

- view a commits details: git show cdae0dx
    - git show [short SHA-1]
- View the commits on the Head: git show HEAD

Comparing Commits

- view the changes in a commit: git diff cdae0dx
- view the changes of a file in a commit: git diff cdae0dx index.html
- comparing differences between commits: git diff cdae0dx..acf8750
    - Anther Example: comparing differences between commits: git diff cdae0dx..HEAD
- comparing differences between commits for a file: git diff cdae0dx..acf8750 index.html
- view stats and summary of differences between commits: git diff --stat --summary cdae0dx..HEAD
- Ignore difference in spacing
    - extra spaces: git diff --b cdae0dx..HEAD
    - all spaces: git diff --w cdae0dx..HEAD

Branches in Git

- branches are ideas
    - try new ides
    - isolate features or sections of work
- one working directory
- fast context switching

Creating a branch

- view branches: git branch
- create a new branch: git branch new_feature

Switching to a branch:

- switch branch: git checkout new_feature

- create and with a branch: git checkout -b shorten_title
- you cannot switch to another branch if your have un committed changes
    - Solutions
        - stash changes
        - commit changes
        - scrap the changes

Comparing Differences between branches

- Comparing differences between branches: git diff master..new_feature
    - git diff [old state]..[new state]
- Comparing differences between a previous branches: git diff master..new_feature^
- view what commits are included in current branch: git branch --merged

Rename a Branch

- rename a branch: git branch -m new_feature seo_title

Delete a Branch

- Delete a branch: git branch -d branch_to_delete
- you cannot delete a branch that your are currently not on
- git will warn you when deleted a branch with un merged changes
    - Delete a branch with unique commits: git branch -D branch_to_delete

Show branch in command prompt:

- https://www.lynda.com/Git-tutorials/Configuring-command-prompt-show-branch/100222/111314-4.html?autoplay=true

Merging Code

- checkout into the branch that is receiving the update
- merge seo_title branch into master: git merge seo_title
- merge with a clean working directory
- fast forward merge: no changes in master, but external branch is different
- commit a normal fast forward merge the normal way: git merge --no-ff seo_title
- commit only if fast forward is possible: git merge --ff-only seo_title
- recursive is a merge strategy

Merge Conflicts

- Same line modified in two commits
- resolve strategy
    - resolve manually
        - incorporate differences into new code, so no conflicts exit
    - abort merge
        - abort a merge: git merge --abort
    - use a automatic tools
- Avoid Conflicts
    - Small Commits
    - keep lines short
    - don't change whitespaces
    - merge often
    - track changes to master
        - import changes from master into working branches

Stash

- like commits with no SHA-1
- used when changing branches
- stash changes: git stash save "changed mission page title"
- view items in stash: git stash list
- stash is available all the time, across branches
- show stash changes: git stash show stash@{0}
- show detailed stash changes: git stash show -p stash@{0}
- import specific stash changes and remove from stash: git stash pop stash@{0}
- import specific stash changes only: git stash apply stash@{2}
- import all stash changes only: git stash apply
- re-stash incorrect import: git stash save "changed to mission page title"
- delete specific stash: git stash drop stash@{0}
- delete all stashes: git stash clear

Remote Repositories:

- local and remote repositories exist
- remote master: origin/master
- local master: master
- push changes to remote repo
- fetch changes to local repo
- show all remote repos we know about: git remote
- add a remote repo: git remote add origin https://…
- view git remote info: git remote -v
- remove a remote repo: git remote rm origin https://…
- push master branch to GitHub: git push -u origin master

- create a local copy of remote repo: git clone https://…
- create a local copy of remote repo in new folder: git clone https://… Lynda

- fetch changes from remote repo: git fetch
    - alternative: git fetch origin
- always fetch before you work
- fetch before you push
- fetch often

- when updated side branches, update master, and then pull changes from local master to side branches
- your can use git fetch + git merge at the same time: git pull

- Delete a remote branch: git push origin :non_tracking
    - deletes remote branch on server/github
        - local branch exists

Aliases

- Creating Aliases for Git Commands: https://www.lynda.com/Git-tutorials/Setting-up-aliases-common-commands/100222/111342-4.html?autoplay=true