**Git**

**Basics**

git status

* View status of your files in the working directory and staging area
* Good for answering questions like:
  + What did that command just do?
  + What branch am I on?
  + What changes am I about to commit, and have I forgotten anything?
  + Was I in the middle of something last time I worked on this project?

git diff

* Shows changes between working directory and staging index.
* git diff <file\_name>
  + Can be used on single files
* git diff –color-words <file\_name>
  + Puts the changes on one line and colors the differences.
* git diff --staged
  + Look at staging index compare against repo
* Git diff <sha\_value>
  + Diff between working directory and that commit
* Git diff <sha\_value> <file\_name>
  + Diff between commit and working directory on specific file
* Git diff <sha\_value>..<sha\_value>
  + Diff between commit to commit
* Git diff <master>..<new\_feature>
  + Compares branches
* Git diff --color-words <branch\_name>..<diff\_branch>
  + Diff between branches and colors differences
* Git diff –w <sha\_value>..HEAD
  + Diff between specified commit and tip of currently checked out branch

git rm

* Removes the files and adds to staging

git mv

* git mv <old\_file\_name> <new\_file\_name>
* Deleting, moving/renaming is easier for handle *because it will add to staging index.*
* Move and rename synonymous, same as linux

git add

* Adds file contents to the staging area

git commit

* Records a snapshot of staging area
  + Git commit –m
    - Records snapshot of staging area with message
  + git commit –a
    - Adds and commits to staging area
    - Works well with modifications but new/deleted files are not included
  + git commit –am
    - Combines both commands
    - Tip: Group files together that has to deal with one particular issue for a commit.
  + git commit --amend –m
    - Once commit is in staging area
    - This can also just update the commit message

git checkout

* Goes to **repo**, grab the named thing I gave you and make working directory look like that!
  + git checkout -- <file\_name>
    - -- dash dash means not checking out a new branch just a file in the current branch

git revert <SHA\_Value>

* Do the exact opposite of changes

git reset

* Undo changes and commits
* git reset --soft
  + Does not change staging index or working directory but moves the HEAD pointer
* git reset --mixed(default)
  + Reset staging area to most recent commit but leave the working directory unchanged.
* git reset --hard
  + Changes staging index and working directory to match repo

git blame

* Will display who made changes for each line

git clean –n

* Shows which files would be removed from working directory. Use the –f flag in place of the –n flag to execute the clean.

git log

* Show commit history or a branch
* git log --oneline
  + More compact version of the same history
* git log --pretty=oneline
  + Pretty print contents of the commits logs in oneline format
* git log --online --graph --all --decorate
  + SUPER SWEET!
* Git log --graph --decorate --oneline --abbrev --commit --all
* git log --oneline -3
  + Show last three commits
* git log --grep=”temp”
  + Global regular expression search
    - Ex search for “temp”
* git log –p
  + Patch- more details about commits, shows changes
* git log --graph
  + Shows branches

Viewing Commit

Git show

* Show various types of objects
  + Objects(blobs, trees, tags and commits)
* git show <sha\_value>
  + Gives info for commit

Branches

git branch

* List branches on local machine
* git branch <branch\_name>
  + Creates new branch
* git branch –d <branch\_name>
  + Delete branch
* git branch –m <old\_branch\_name> <new\_branch\_name>
  + Change branch name
* git branch --merged
  + All branches completely included in a particular commit
* git checkout –b
  + Create new branch and switch to it

Stashin’

git stash

* Stash the changes in a dirty working directory away
* git stash save “message”
  + Create stash
* git stash list
  + Show items in stash
* git stash show –p stash@{0}
  + Show the changes recorded in the stash as a diff between the stashed state and its original parent.
* git stash pop
  + Remove a single stashed state from the stash list and apply it on top of the current working tree state.
* git stash apply
  + Like pop, but do not remove the state from the stash list.
* git stash drop stash@{0}
  + Remove a single stashed state from the stash list
* git stash clear
  + Remove all stashed states.

Remotes

Git remote

* Will show remote name of branch
* git remote –v
  + Shows remote to fetch and to push to
* git remote rm origin (does not have to be named origin)
  + Removes remote repo
* git remote add origin <url>
  + Add remote server, this mechanism is to establish a connection to a remote server
* git push –u origin master
  + –u tracks remote branch
* git branch –a
  + shows remote and local branches
* git fetch
  + syncs up origin/master with whatever is on remote repo
* git pull
  + git pull = git fetch + git merge
    - 2 step process
* git push origin :<branch\_name>
  + deletes branch
* git push origin --delete <branch\_name>
  + Also deletes remote branch

**Extra Info**

SHA

* Cryptographic hash function
* Git generates a checksum(SHA) for each change set
  + Checksum algorithms convert data into a simple number.
  + Same data always equals the same checksum
* Data integrity is fundamental
  + Changing data would change the checksum
  + Git uses SHA-1 hash algorithm to create checksums
    - 40 character hexadecimal (0-9,a-f)

The HEAD Pointer

* Points to “tip” of current branch
* Last state of repo, what was last checked out
* Points to parent of next commit
  + Where writing takes place
  + ***HEAD- Always points to the tip of the currently checked out branch from the repo.***

Git Ignore

* .gitignore – which files should be used for commits and which should be ignored.
* Very basic regex
  + \* ? [aeiou] [0-9]
  + Negate expressions with !
    - \*.php (ignore any file that ends in .php)
    - !index.php (don’t ignore this file)
* What to ignore?
  + Compiled source code
  + Packages & compressed files
  + Logs and databases
  + Operating system generated files
  + User uploaded assets(images, PDF’s, videos)
  + .gitkeep used for tracking empty directories

Misc

git mergetool

* Run merge conflict resolution tools to resolve merge conflict

cat .git/config

* Shows git config file

cat .git/head

* Shows where head is pointing

git merge --abort

* Abort merge

git rev-parse HEAD

* Display commit hash

git reset –hard HEAD~1

* Nuke last commit