Git Workflow

Notes

* .git holds all of the tracking information
* Make the commit group file differently for better understanding

Keep in Mind

* To avoid all unstaged files use git checkout -- .
* For single file use git checkout path\_name

To remove

* another git process seems to be running in this repository
* rm -f ./.git/index.lock

General Note

* Git last commit message modification: git commit --amend -m "New commit message"
* git status
* Is going to report back to us the difference between our working directory, the staging index and the repository.
* In git default branch is called master.
* Nothing to commit means our working directory is clean, and it tell us there is nothing to commit our working directory is clean, that’s letting us know that there’s nothing in the staging directory and the working directory exactly matches what is in the repository and the current branch where the Head pointer points to that exactly the same things that have in our working directory.
* From working directory to staging directory use

git add .(for all files)

git add file.txt

* :spacebar used for pager or f for forward & b for backward
* -+shift+S used for seeing whole page& also for back
* git log --oneline – to see the commit in oneline
* git show HEAD – to see the changes

Viewing changes with diff

* Diff is used to compare two files
* Git diff gives us the difference between old version & new version

git diff

* Return comparing what’s in the repository versus what’s in our working directory.
* Minuses indicate what in the repository & pluses indicate what is in the working directory
* For specific file used

git diff first\_file.txt

* git diff --color-words contact.html(file\_name.html) it gives the changed information in side-by-side instead of different file

Comparing between staged file & repository file

* Command : git diff –staged
* Its means look in the staging index & compare that against the repository.

Add file

* For all files git add .
* Add a folder in git git add tours/

Delete a File from Git

* If delete a file manually and show it in git status then write

git rm file.txt

And git commit then it doesn’t show in git status

* First add the file in working directory process add & commit
* After that write git rm file\_name.txt
* After that commit like “first red shirt file deleted”

Renaming file

* If the file is not add in the staged file & didn’t committed then first add it to in git & commit
* Then rename the file using git mv past\_file\_name.txt new\_file\_name.txt

Move File from one directory to new directory

* First add & committed the file
* Then write git mv file\_name.txt Moved\_file(directory name)/file\_name.txt
* Rename & move file at the same time git mv file\_name.txt Moved\_file(directory name)/file\_name\_new.txt
* Finally commit after renaming or moving file

For files add & commit all at a once

* git commit –am “changed 24 hour support number to 4314”

Two Caveats for this option

* This grabs everything that is in our working directory so if we something don’t include in the change they’re going to pushed up there as well.
* Files that are not tracked, or files that are being deleted, do not get included in this. So it works well for modifications, but for new files and deleted files it doesn’t work well.

Initialize git repository

* git init

HEAD

* points to the last commit

Undoing working directory

* git checkout with a double dash & then the file or the directory that we wanted to pull down from the repository will blow away the changes and restore them back to what we had in the repository.
* Command: git checkout --index.html this will help if something delete accidently from the file & this will work only for current directory
* Dash dash means stay on the current branch. Double dash indicating we are not checking the out the branch just talking about current branch.
* git checkout index.html this will go to the repository, get the named thing that I’ve gave you, and my working directory look like that.
* If the named thing is the branch it brings the branch down. If the named thing is the file, it brings the file down.
* If we want to pull down resources folder write git checkout resources
* Imagine if resources is a branch then git will pull down resources branch instead of resources folder.

Undoing or Unstaging Files

* Sometimes if a file accidently staged we can remove the file from staging directory
* Unstaging file : git reset HEAD resources.html

Undoing Commit

* Git only allows last commit editable
* Edit last commit : git commit --amend –m “Rearrange the items to bring on a trip”

Retrieving old versions

* Command: git checkout e123ef4445(copy the first 10 characters of sha) -- resources.html
* Best practicing of commit in this case like “Reverts commit e123ef4445(copy the first 10 characters of sha)”

Reverting a commit

* Undo a changes for a commit completely & totally we use the Revert command.
* git revert e123ef4445(copy the first 10 characters of sha)
* It pops up the text editor & then we save the save the then it will complete other process for us like commit.

Using reset to undo many commits

git reset

* --soft – does not change staging or working directory it’s just move the pointer
* --mixed (default) – changes staging index to match repository it’s does not change our working directory.
* --hard – changes staging index and working directory to match repository so that means any changes that came after that commit are completely obliterate. They don’t exist in the repository, the staging index, or the working directory.
* Git pointer points at now cat .git/HEAD
* cat .git/refs/heads/master this will return sha or we can say this file contain real sha
* For rewind in a specific point use git reset --hard f570a41502

Removing Untrack files

* git clean –n (-n means test run)
* git clean –f (-f forces it to run)

Ignoring versioned files

* To ignore a version file is tracked in first but later does not need to track the file
* Run following commands: git update-index --assume-unchanged <files>

git update-index --assume-unchanged path/to/file.txt

* If u r already in directory write just: git update-index --assume-unchanged file.txt
* To make Git track the file again, simply run: git update-index --no-assume-unchanged path/to/file.txt

Ignoring Files – Using gitignore

* .gitignore is useful to untrack temporary files or log files
* .gitignore is used to ignore files
* We can ignore files by writing files one after or by regular expression.
* Very basic regular expression like \* ? [aeiou] [0-9]
* We can also negate expressions with !
* We can say ignore any files that end with \*.php but don’t ignore !index.php
* Ignore all files in a directory with trailing slash like assets/videos/
* Comment lines begin with #, blank lines are skipped
* We can create the gitignore files in different ways
* First open text editor & write the file name which one we want to ignore after .settings/ & if we want to skip group of files then we write \*.txt that means ignore all files that are ending with .txt
* Some general rules for ignoring files

# comment

third\_file.txt

.DS\_Store

\*.zip

\*.gz

log/\*.log

log/\*.log.[0-9]

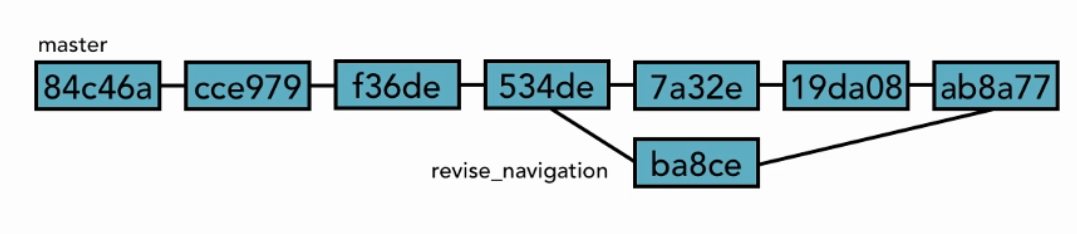
assets/photoshop/

assets/videos/

!assets/videos/tour\_\*.mp4

Git Branches

* Try new ideas
* Isolate features or sections of work
* One working directory
* Fast context switching



Git Branch Command

* git branch – returns the name of the branches in our local machine
* cat .git/HEAD – to know which branch git now pointing
* ls –la .git/refs/heads – to see the directory listing in the .gitfolder
* \* sign points in which branch currently we are
* git branch dev(branch\_name) – to create a new branch
* git checkout dev(branch\_name) – to switch on a new branch
* git checkout –b dev(branch\_name) – create & switched to the branch at the same time.
* Git log --graph -- oneline --decorate --all – will return head commit of the all branches

Branching-Switching branches with uncommitted changes

* Cautious: working directory must be clean in order to switch actually mostly clean. We have modified file in branch then if we want to switch another branch then we should checkout the file otherwise commit the changes for protecting the changes.

Branching-Comparing branches

* gitdiff master..dev – to see the difference the two branches
* git diff --color-words master..dev – to see the difference the two branches

Merging Branches

* git branch --merged – is used for merging all of the branches in the current branch

Rename Branches

* git branch –m dev(old\_name) local(new\_name)

Delete Branch

* git branch –d branch\_to\_delete(branch\_name)

Git Clone

* Copy the address & open the git bash from desired location & paste it & hit enter for cloning
* git clone <https://hasanmbstu13@bitbucket.org/musabd/pos.git>