Git notes

Git is a version controlling system.

It allows you to keep versions of your code, branch them, and do other very useful things with them.

The first thing you need to do is install Git. Normally it will be in terminal. From there you can move on to the core functionality of Git.

From now on, everything in these notes expects you to be in a UNIX environment.

Starting

Everything in Git is stored in a repository. Think of a repository as a directory. The files are stored underneath it. To create a repository, you simply move to a directory and initialize it as a repository using the command:

git init

That states that the current directory is now a repository. It initializes a ‘master’ branch for it.

Cloning

If you want a copy of a working repository so that you can edit it, you simply need to clone it.

git clone

This way, more than one person can work on a remote repository, and all of them update it.

Snapshots

So when you want to start changing files, there are some things to note. Firstly, there is a ‘staging area’ before you commit. This staging area is what will be commited when you hit commit. It’s your change to change, add, remove and diff before committing. To add files to this area, you must use the git add command. To check the status of the area and see what has changed, use git status and git diff.

Git add

Git status

Git diff

Git commit

To note, git diff will only show you the changes NOT IN THE STAGE (have not been added using git add). To check the ones that have been added, use

Git diff --cached

To see every change in the entire repository, use

Git diff HEAD

Finally, we can use one more diff command to see a status of changed files, but not the specifics. The command for that is:

Git diff --stat

Once everything you want is stages, you hit ‘git commit’ to commit these changes. You are going to want to do:

Git commit –m “DESCRIPTION”

In order to give the commit a message.

Undo

You can undo a commit in three ways:

Git reset HEAD -- <file name>

Git reset –soft <COMMIT>

Git reset –hard <COMMIT>

The HEAD reset will unstage the file.

The soft reset will undo the commit and put the files back into the stage.

The hard reset will undo the commit and undo all local file changes too.

Stashing

A useful tool, stashing is the process of storing changes FOR NOW and reverting to the last commit. This way, you can get far in a changeset, and then realize that you want to work on something else. You can stash your current changes and then go work on that instead. Later you can come back, unstash it, and continue where you left off.

Git stash

Will add the current stage to the stash.

Git stash list

Will list you all the working stashes so you can pick and choose what to unstash.

Git stash apply stash@{<NUMBER>}

Will get you the stash at the specific number you can check with the stash list command.

Git stash drop stash@{<NUMBER>}

Will completely delete the stashed number

Specific things you may want to do

**Remove a file from version control, but keep it locally:**

git rm –caches <FILE NAME>

**Stage all tracked and modified files:**

git add -u

**Remove files from being STAGED ONLY:**

git reset HEAD <FILE NAME>

**Remove ALL files that have been STAGED:**

git reset

**Revert ALL changes to local files:**

git checkout .

**Revert changes to a SINGLE file:**

git checkout <FILE NAME>

**Revert the previous commit:**

git revert …