Git – cheat sheet

2019-02-26

Getting and Creating Projects

```
git init                                    // create an empty Git repository or reinitialize an existing one
git clone <repository> [<directory>]        // clone a repository into a new directory

git fetch                                   // downloading content from a remote repo without merging
git pull [--rebase]                         // execute git fetch followed by git merges (or rebase)
git push                                    // transfer commits from the local repository to a remote repo
git push -u origin <branch>                 // transfer commits to a remote repo and add upstream reference
```

Basic Snapshotting

```
git add [--] [<pathspec> ..]                // add file contents to the index
git add -i                                  // add modified contents in the working tree interactively
git add -A                                  // add all changes (add, modify, delete) to the index
git add -u                                  // add all changes (modify, delete) to the index

git status                                  // show the working tree status

git commit                                  // stores the current contents of the index in a new commit
git commit -a                               // automatically stage files that have been modified or deleted
git commit -m <msg>                         // use the given <msg> as the commit message
git commit --amend -m <msg>                 // replace the tip of the current branch by creating a new commit

git reset [--] [<paths> ..]                 // reset the index entries (is the opposite of git add)
git reset --hard                            // resets the index and working tree. Any changes are discarded!

git rm [--] [<file> ..]                     // remove files from the working tree and from the index
git rm --cached [--] [<file> ..]            // unstage and remove paths only from the index
git mv <source> <destination>               // move or rename a file, a directory, or a symlink
```

Branching and Merging

```
git branch [-a]                             // list local branches (-a for remote-tracking and local branches)
git branch -d <branch>                      // delete a branch

git checkout <branch>                       // switch branches or restore working tree files
git checkout -b <branch>                    // create a new branch and then checked out

git merge <branch>                          // merge branch <branch> into the current branch
git merge --no-commit <branch>              // perform the merge but do not autocommit
git merge --no-ff <branch>                  // create a merge commit even when we can fast-forward merge
git merge --continue                        // continue the merge after a resolved merge conflict
git merge --abort                           // try to reconstruct the pre-merge state
```

Inspection and Comparison

```
git show <commit>                           // shows the log message and textual diff for the given commit ID

git log [--] [<path> ..]                    // show commit logs
git log [<revision range>]                  // merge branch <branch> into the current branch
git log -L <start>,<end>:<file>             // trace the evolution of the line range given by <start>,<end>
git log --pretty=oneline                    // show the log in a very compact format
git log --graph                             // draw a graphical representation of the commit history on the left
git log --first-parent                      // follow only the first parent commit upon seeing a merge commit

git diff                                    // show the changes relative to the index (staging area)
git diff --cached                           // show all staged changes
```

Temporarily shelves (or stashes) changes

```
git stash [--] [<pathspec> ..]              // stash the changes in a dirty working directory away
git stash push -m <msg>                     // save local modifications to a new stash entry using <msg>
git stash list                              // list the stash entries that you currently have
git stash show [-p] [<stash>]               // show the changes recorded in the stash entry as a diffstat/patch
git stash apply [<stash>]                   // apply changes in the stash on top of the current working tree
git stash pop [<stash>]                     // same as  apply, but remove the state from the stash list
git stash drop [<stash>]                    // remove a single stash entry from the list of stash entries
```

Other useful commands

```
git-apply <patch>                           // apply a patch to files and/or to the index
git-format-patch <revision range>           // prepare each commit with its patch in one file per commit
git blame [--] <file>                       // show what revision and author last modified each line of a file
git cherry-pick <commit>                    // apply the changes introduced by some existing commits
git rebase <branch>                         // reapply commits on top of another base tip
```