Git Quick Start Commands

Command Listing, Part 1

```
pwd
mkdir projects
cd projects
pwd
```

Command Listing, Part 2

```
git version
git config --global user.name "Abe Lincoln"
git config --global user.email "mrabe@git.training"
git config --global --list
git clone github-https-url  # paste in your GitHub HTTPS clone URL
ls
cd github-demo
ls
git status
echo "Test Git Quick Start demo" >> start.txt
ls
cat start.txt
git status
git add start.txt
git status
git commit -m "Adding start text file"
git status
git push origin master
```

Text Editor Install and Configure

Windows: Notepad++

While Notepad will work, Notepad++ is a much better text editor that is also free.

Command Listing

```
notepad++  # testing to make sure it works
notepad++ .bashrc  # see contents of .bashrc file below
npp  # testing npp alias
git config --global core.editor "notepad++.exe -multiInst -nosession"
npp .gitconfig  # open Git config file, should contain snip below
```

~/.bashrc File

```
alias npp='notepad++.exe -multiInst -nosession'
```

~/.gitconfig File (snip)

```
[core]
    editor = notepad++.exe -multiInst -nosession
```

Mac OS X: TextMate 2

On the Mac, TextMate 2 is a freely available text editor that work a bit better than TextEdit.

Command Listing

```
git config --global --list  # before
git config --global core.editor "mate -w"
git config --global --list  # after
mate .gitconfig  # compare with snip below
```

~/.gitconfig File (snip)

```
[core]
    editor = mate -w
```

Basic Commands

Completely Fresh

Command Listing

```
pwd
cd projects
pwd
git init fresh-project  # replace with name of your project
ls
cd fresh-project
ls
ls -al
cd .git
ls
cd ..
pwd
git status
mate hipster.txt
git status
git add hipster.txt
git status
git commit
git status
cd ..
pwd
ls
rm -rf fresh-projects
```

Existing Source

Command Listing

```
pwd
cd projects
unzip ~/Downloads/initializr-verekia-4.0.zip
ls
mv initializr web-project
ls
cd web-project
ls
git init
ls -al
git status
git add .
git status
git commit -m "My first commit, inline"
git status
git status
ls -al
rm -rf .git
ls
pwd
ls
rm -rf web-project
```

Remote (GitHub)

Command Listing

```
pwd
cd projects
git clone github-https-url # my value: https://github.com/awesomejt/starter-web.git
ls
cd starter-web
ls
ls -al
cd .git
ls
cd ..
clear
git status
```

Basic Git Workflow / First Commit

Command Listing

```
pwd
cd projects
ls
cd starter-web
ls
git status
mate hipster.txt
ls
git status
git add hipster.txt
git status
git commit
git status
git pull origin master
clear
git push origin master
```

Adjusting the Git Config

Resetting to author's name from demo in the Git Quick Start.

Command Listing

```
mate ~/.gitconfig
```

~/.gitconfig File

```
[user]
    name = Jason Taylor
    email = jason.git.training
[core]
    editor = mate -w
```

Tracked Files / Express Commit

Command Listing

```
pwd
cd projects
ls
```

```
cd starter-web
ls
git status
mate hipster.txt
git commit -am "Adding more ipsum text"
git ls-files
mate newfile.txt
git status
git ls-files
git add newfile.txt
git status
git ls-files
```

Editing Files

Command Listing

```
pwd
cd projects/starter-web
clear
ls
mate hipster.txt
git status
git commit -m "Adding new file"
git status
git add hipster.txt
git status
mate hipster.txt
git status
git add hipster.txt
git status
git commit -m "More ipsum for hipsters"
git status
```

Recursive Add

Command Listing

```
pwd
cd projects/starter-web
git status
mkdir -p level1/level2/level3/level4  # last item later deleted (currected from video)
ls
cd level1
pwd
mate level1-file.txt
ls
cd level2
mate level2-file.txt
ls
cd level3
mate level3-file.txt
ls
rm -rf level4
clear
cd ../../..
ls
git status
git add .
git status
git commit
```

Backing Out Changes

Command Listing

```
pwd
cd projects/starter-web
ls
git status
cd level1
ls
mate level1-file.txt
git status
git add level1-file.txt
git status
mate level1-file.txt
clear
git status
git reset HEAD level1-file.txt
mate level1-file.txt
clear
git status
git checkout -- level1-file.txt
git status
mate level1-file.txt
```

Rename

Command Listing

```
pwd
cd projects/starter-web
ls
cd level1/level2/level3
pwd
clear
ls
git status
git mv level3--file.txt level3.txt
ls
git status
git commit -m "renaming level3 file"
cd ..
clear
pwd
ls
mv level2-file.txt level2.txt
ls
git status
git add -A
git status
git commit
clear
ls
git mv level2.txt 2.txt
ls
git status
git mv 2.txt level2.txt
ls
git status
clear
ls
git mv level2.txt level3
ls
```

```
cd level3
ls
git status
cd ..
git status
git commit
ls
cd level3
ls
mv level2.txt ..
ls
cd ..
pwd
ls
git status
git add -A
git status
git commit
clear
cd ..
pwd
# file renamed in Finder
git status
git add level1.txt
git add -u
git status
git commit
```

Deleting Files

Command Listing

```
pwd
cd projects/starter-web
ls
git status
clear
mate doomed.txt
ls
git status
git rm doomed.txt
rm doomed.txt
ls
git status
clear
git ls-files
ls
git rm newfile.txt
git status
git commit -m "Deleting new file"
git status
clear
git status
git ls-files
git rm hipster.txt
ls
git status
git reset HEAD hipster.txt
git status
git checkout -- hipster.txt
ls
git status
clear
ls
rm hipster.txt
```

```
ls
git status
git add -A
git status
git commit
git status
clear
ls
rm -rf level1
ls
git status
git add -A
git status
git commit -m "deleting level1 and all children"
git status
```

History

Command Listing

```
pwd
git help log
cd projects/starter-web
git status
clear
git log
git log --abbrev-commit
git log --oneline --graph --decorate
git log ae6f872...761b911
git log --since="3 days ago"
clear
ls
git log -- hipster.txt
clear
git log --follow -- level1/level2/level2.txt
clear
git show b1967804190e199d4c753e2c5e59d09ce65842d6
clear
```

Git Alias

Command Listing

```
pwd
cd projects/starter-web
ls
git status
clear
git log --all --graph --decorate --oneline
git hist
git config --global alias.hist "log --all --graph --decorate --oneline"
git hist
clear
mate ~/.gitconfig
git hist
mate ~/.gitconfig
git hist
clear
```

Git Configuration File (~/.gitconfig) snip

```
[alias]
```

```
    hist = log --all --graph --decorate --oneline
```

Git Alias

Command Listing

```
pwd
cd projects/starter-web
ls
git status
ls -al
mate .gitignore
ls -al
git status
git add .gitignore
git status
git commit
git status
ls
mate access.log
ls
git status
git .gitignore
clear
git status
mkdir log
mv access.log log
ls
cd log
ll
cp access.log access.2014-11-04
ll
cd ..
git status
mate .gitignore
git status
git commit -am "Excluding log file directory"
git status
```

Git Basics Cleanup

Command Listing

```
pwd
cd projects/starter-web
ls
git status
clear
git pull origin master
git push origin master
```

Setup P4Merge

p4merge.path Windows

```
git config --global merge.tool p4merge
git config --global mergetool.p4merge.path "C:/Program Files/Perforce/p4merge.exe"
git config --global diff.tool p4merge
git config --global difftool.p4merge.path "C:/Program Files/Perforce/p4merge.exe"
```

Mac OS X

```
git config --global mergetool.p4merge.path /Applications/p4merge.app/Contents/MacOS/p4merge
git config --global merge.tool p4merge
git config --global difftool.p4merge.path /Applications/p4merge.app/Contents/MacOS/p4merge
git config --global diff.tool p4merge
```

Disable Prompt (Both)

```
git config --global difftool.prompt false
git config --global mergetool.prompt false
```

Visual Diff and Merge Tool Install and Configure

Windows: P4Merge for Windows

P4Merge on Windows is an excellent choice for a free visual merge tool.

Command Listing

```
git config --global merge.tool p4merge
git config --global mergetool.p4merge.path "C:/Program Files/Perforce/p4merge.exe"
git config --global diff.tool p4merge
git config --global difftool.p4merge.path "C:/Program Files/Perforce/p4merge.exe"
git config --global difftool.prompt false
git config --global mergetool.prompt false
```

~/.gitconfig File (snip)

```
[merge]
    tool = p4merge
[mergetool "p4merge"]
    path = C:/Program Files/Perforce/p4merge.exe
[diff]
    tool = p4merge
[difftool "p4merge"]
    path = C:/Program Files/Perforce/p4merge.exe
[difftool]
    prompt = false
[mergetool]
    prompt = false
```

Mac OS X: P4Merge for Mac

P4Merge on the Mac is an excellent choice for a free visual merge tool.

Command Listing

```
git config --global mergetool.p4merge.path /Applications/p4merge.app/Contents/MacOS/p4merge
git config --global merge.tool p4merge
git config --global difftool.p4merge.path /Applications/p4merge.app/Contents/MacOS/p4merge
git config --global diff.tool p4merge
git config --global difftool.prompt false
git config --global mergetool.prompt false
```

~/.gitconfig File (snip)

```
[diff]
    tool = p4merge
[merge]
    tool = p4merge
[difftool "p4merge"]
    path = /Applications/p4merge.app/Contents/MacOS/p4merge
[mergetool "p4merge"]
    path = /Applications/p4merge.app/Contents/MacOS/p4merge
[difftool]
    prompt = false
[mergetool]
    prompt = false
```

Git Stashing

Simple Example

How to simple path for stashing in Git.

Command Listing

```
pwd
cd projects/starter-web
pwd
git status
clear
ls
mate simple.html
git status
git stash
git status
mate simple.html
clear
ls
mate README.md
git status
git commit -am "Quick fix in production to improve copyright notice"
git status
clear
git stash apply
mate simple.html
git commit -am "Done with simple.html updates"
git status
clear
git stash list
git stash drop
clear
```

Stashing Untracked Files / Using Stash Pop

Git does not include untracked (new) files with stash by default, however, we change change that. Also, we cover a new way to apply the most recent stash.

Command Listing

```
pwd
git status
clear
git ls-files
mate humans.txt
git status
mate ANewFile.txt
git status
git stash
git status
git stash apply
git stash drop
git stash list
clear
git status
git stash -u
clear
git stash list
mate README.md
git commit -a
git status
clear
git stash pop
clear
git status
rm ANewFile.txt
git status
git commit -am "Updates to humans file after stash pop"
git status
```

Managing Multiple Stashes

Git's stash feature can support multiple stashes.

Command Listing

```
pwd
git status
clear
ls
mate simple.html
git stash save "simple changes"
clear
mate index.html
git stash save "index changes"
mate README.md
git stash save "Readme changes"
git stash list
git stash show stash@{1}
clear
git status
git stash list
git stash apply stash@{1}
clear
git status
git stash list
git stash drop stash@{1}
git stash list
clear
git stash list
git stash clear
git stash list
clear
```

Stashing into a Branch

We can use stash to move accidental changes into a feature branch.

Command Listing

```
pwd
git status
git stash list
clear
mate
mate simple.html
mate humans.txt
git status
git add index.html
git status
mate new.md
git status
git stash -u
git status
clear
git stash branch newchanges
git stash list
git status
rm new.md
git add .
git status
git commit
git checkout master
git merge newchanges
git branch -d newchanges
git branch
clear
```

Section Cleanup

Each section, we synchronize our changes with GitHub.

Command Listing

```
pwd
git status
git pull origin master
git push origin master
```

Git Tagging

Simple Tagging Example / Lightweight Tags

Simple example with *lightweight* tags and some basic tag commands.

Command Listing

```
pwd
cd projects/starter-web/
git status
git log --oneline --decorate --graph --all
git tag myTag
git log --oneline --decorate --graph --all
git tag --list
git show myTag
git tag --list
git tag --delete myTag
git tag --list
git log --oneline --decorate --graph --all
```

Annotated Tags

Annotated tags are tags with more information, like comments, associated with them.

Command Listing

```
pwd
git status
clear
git tag -a v-1.0
git tag --list
git log --oneline --decorate --graph --all
git show v-1.0
```

Comparing Tags

Tags mark important milestones in a project, so they are great way to compare what has happened between those milestones.

Command Listing

```
pwd
git status
git tag --list
ls
mate index.html
git commit -a
git log --oneline --decorate --graph --all
git tag -a v-1.1
mate simple.html
git commit -am "Updating for tag 1.1"
git commit --amend
git tag v-1.2 -m "Release 1.2"
git tag --list
git log --oneline --decorate --graph --all
git diff v-1.0 v-1.2
git difftool v-1.0 v-1.2
```

Tagging a Specific Comment

If you forget to tag a commit in the past, you can simply pass in the commit id while creating the tag.

Command Listing

```
pwd
git status
git log --oneline --decorate --graph --all
git tag -a v-0.9-beta 96ef75b
git log --oneline --decorate --graph --all
git tag -a v-0.8-alpha ab0d621
git log --oneline --decorate --graph --all
```

Updating a Tag

Sometimes mistakes happen and you'll need to update a tag.

Command Listing

```
pwd
git status
git log --oneline --decorate --graph --all
git tag -a v-0.8-alpha -f bd35d46
git log --oneline --decorate --graph --all
```

Remote Tagging (GitHub)

Working with remote tags on GitHub.

Command Listing

```
pwd
git status
git tag --list
git log --oneline --decorate --graph --all
git push origin v-0.9-beta
git push origin v-1.1
clear
git push origin master --tags
git push origin :v-0.8-alpha
```

Git Mac Updates

Version: 1.0

Broken Git After OS X Upgrade

After upgrading to a new version of Mac OS (like El Capitan), Git might stop working -- particularly if you are using the Apple variation of Git (Apple Git) included with the Developer CommandLine Tools.

History

Starting in Mac OS X Yosemite (10.10), Git Users that wanted the version of Git provided by Apple, but not the full XCode suite and the unnecessary download and space taken up on their systems could have that version install by simply typing `git version` in the **Terminal** app (*Applications > Utilities*). Apple would prompt the user if they wanted to just install the **Developer Command Line Tools** or the full **XCode** -- choosing the former would install a set of command line tools, including **Git**.

However, in some cases, after the user upgrades the Mac OS X to the next version (Example: Yosemite to El Capitan), Git would stop working and simply output a cryptic error message (see below).

Test

```
git version
```

Encounter error message like:

```
xcrun: error: invalid active developer path
(/Library/Developer/CommandLineTools), missing xcrun at:
/Library/Developer/CommandLineTools/usr/bin/xcrun
```

The Cure

Do this:

```
sudo xcode-select --install
```

This will re-prompt the dialog when we first installed the Developer CommandLine

Tools. Like before, we don't need to install the full XCode just to install Git (although you certainly can).

Verify

```
git version
```