```
$ git init
# Print & Set Some Basic Config Variables (Global)
$ git config --global user.email "MyEmail@Zoho.com"
$ git config --global user.name "My Name"
# Quickly check available commands
$ git help

# Check all available commands
$ git help -a

# Command specific help - user manual
# git help <command_here>
$ git help add
$ git help commit
$ git help init
# or git <command_here> --help
$ git add --help
$ git commit --help
$ git init --help
$ echo "temp/" >> .gitignore
$ echo "private_key" >> .gitignore
# Will display the branch, untracked files, changes and other differences
$ git status

# To learn other "tid bits" about git status
$ git help status
# add a file in your current working directory
$ git add HelloWorld.java

# add a file in a nested dir
$ git add /path/to/file/HelloWorld.c

# Regular Expression support!
$ git add ./*.java

# You can also add everything in your working directory to the staging area.
$ git add -A
# list existing branches & remotes
$ git branch -a

# create a new branch
$ git branch myNewBranch

# delete a branch
$ git branch -d myBranch

# rename a branch
# git branch -m <oldname> <newname>
$ git branch -m myBranchName myNewBranchName

# edit a branch's description
$ git branch myBranchName --edit-description
# List tags
$ git tag

# Create a annotated tag
# The -m specifies a tagging message, which is stored with the tag.
# If you donâM-^@M-^Yt specify a message for an annotated tag,
# Git launches your editor so you can type it in.
$ git tag -a v2.0 -m 'my version 2.0'

# Show info about tag
# That shows the tagger information, the date the commit was tagged,
# and the annotation message before showing the commit information.
$ git show v2.0

# Push a single tag to remote
$ git push origin v2.0
```



```
# Push a lot of tags to remote
$ git push origin --tags
# Checkout a repo - defaults to master branch
$ git checkout

# Checkout a specified branch
$ git checkout branchName

# Create a new branch & switch to it
# equivalent to "git branch <name>; git checkout <name>"

$ git checkout -b newBranch
# Clone learnxinyminutes-docs
$ git clone https://github.com/adambard/learnxinyminutes-docs.git

# shallow clone - faster cloning that pulls only latest snapshot
$ git clone --depth 1 https://github.com/adambard/learnxinyminutes-docs.git

# clone only a specific branch
$ git clone -b master-cn https://github.com/adambard/learnxinyminutes-docs.git --single-branch
# commit with a message
$ git commit -m "Added multiplyNumbers() function to HelloWorld.c"

# signed commit with a message (user.signingkey must have been set
# with your GPG key e.g. git config --global user.signingkey 5173AAD5)
$ git commit -S -m "signed commit message"

# automatically stage modified or deleted files, except new files, and then commit
$ git commit -a -m "Modified foo.php and removed bar.php"

# change last commit (this deletes previous commit with a fresh commit)
$ git commit --amend -m "Correct message"
# Show difference between your working dir and the index
$ git diff

# Show differences between the index and the most recent commit.
$ git diff --cached

# Show differences between your working dir and the most recent commit
$ git diff HEAD
# Thanks to Travis Jeffery for these
# Set line numbers to be shown in grep search results
$ git config --global grep.lineNumber true

# Make search results more readable, including grouping
$ git config --global alias.g "grep --break --heading --line-number"
# Search for "variableName" in all java files
$ git grep 'variableName' -- '*.java'

# Search for a line that contains "arrayListName" and, "add" or "remove"
$ git grep -e 'arrayListName' --and \( -e add -e remove \)
# Show all commits
$ git log

# Show only commit message & ref
$ git log --oneline

# Show merge commits only
$ git log --merges

# Show all commits represented by an ASCII graph
$ git log --graph
# Merge the specified branch into the current.
$ git merge branchName

# Always generate a merge commit when merging
$ git merge --no-ff branchName
```

```
# Renaming a file
$ git mv HelloWorld.c HelloNewWorld.c

# Moving a file
$ git mv HelloWorld.c ./new/path/HelloWorld.c

# Force rename or move
# "existingFile" already exists in the directory, will be overwritten
$ git mv -f myFile existingFile
# Update your local repo, by merging in new changes
# from the remote "origin" and "master" branch.
# git pull <remote> <branch>
$ git pull origin master

# By default, git pull will update your current branch
# by merging in new changes from its remote-tracking branch
$ git pull

# Merge in changes from remote branch and rebase
# branch commits onto your local repo, like: "git fetch <remote> <branch>, git
# rebase <remote>/<branch>"
$ git pull origin master --rebase
# Push and merge changes from a local repo to a
# remote named "origin" and "master" branch.
# git push <remote> <branch>
$ git push origin master

# By default, git push will push and merge changes from
# the current branch to its remote-tracking branch
$ git push

# To link up current local branch with a remote branch, add -u flag:
$ git push -u origin master
# Now, anytime you want to push from that same local branch, use shortcut:
$ git push
$ git stash
Saved working directory and index state \
  "WIP on master: 049d078 added the index file"
  HEAD is now at 049d078 added the index file
  (To restore them type "git stash apply")
git pull
$ git status
# On branch master
nothing to commit, working directory clean
$ git stash list
stash@{0}: WIP on master: 049d078 added the index file
stash@{1}: WIP on master: c264051 Revert "added file_size"
stash@{2}: WIP on master: 21d80a5 added number to log
$ git stash pop
# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#
#       modified:   index.html
#       modified:   lib/simplegit.rb
#
# Rebase experimentBranch onto master
# git rebase <basebranch> <topicbranch>
$ git rebase master experimentBranch
# Reset the staging area, to match the latest commit (leaves dir unchanged)
$ git reset

# Reset the staging area, to match the latest commit, and overwrite working dir
$ git reset --hard

# Moves the current branch tip to the specified commit (leaves dir unchanged)
# all changes still exist in the directory.
$ git reset 31f2bb1
```



```
# Moves the current branch tip backward to the specified commit
# and makes the working dir match (deletes uncommitted changes and all commits
# after the specified commit).
$ git reset --hard 31f2bb1
38b323f HEAD@{0}: rebase -i (finish): returning to refs/heads/feature/add_git_re
flog
38b323f HEAD@{1}: rebase -i (pick): Clarify inc/dec operators
4fff859 HEAD@{2}: rebase -i (pick): Update java.html.markdown
34ed963 HEAD@{3}: rebase -i (pick): [yaml/en] Add more resources (#1666)
ed8ddf2 HEAD@{4}: rebase -i (pick): pythonstatcomp spanish translation (#1748)
2e6c386 HEAD@{5}: rebase -i (start): checkout 02fb96d
# Revert a specified commit
$ git revert <commit>
# remove HelloWorld.c
$ git rm HelloWorld.c

# Remove a file from a nested dir
$ git rm /pather/to/the/file/HelloWorld.c
```