Getting Started

Create a Repository

Create a new local repository

```
$ git init [project name]
```

Clone a repository

```
$ git clone git_url
```

Clone a repository into a specified directory

```
$ git clone git_url my_directory
```

Make a change

Show modified files in working directory, staged for your next commit

```
$ git status
```

Stages the file, ready for commit

```
$ git add [file]
```

Stage all changed files, ready for commit

```
$ git add .
```

Commit all staged files to versioned history

```
$ git commit -m "commit message"
```

Commit all your tracked files to versioned history

```
$ git commit -am "commit message"
```

Discard changes in working directory which is not staged

```
$ git restore [file]
```

Unstage a stagged file or file which is staged

```
$ git restore --staged [file]
```

Unstages file, keeping the file changes

```
$ git reset [file]
```

Revert everything to the last commit

```
$ git reset --hard
```

Diff of what is changed but not staged

```
$ git diff
```

Diff of what is staged but not yet commited

```
$ git diff --staged
```

Apply any commits of current branch ahead of specified one

```
$ git rebase [branch]
```

Configuration

Set the name that will be attached to your commits and tags

```
$ git config --global user.name "name"
```

Set an email address that will be attached to your commits and tags

```
$ git config --global user.email "email"
```

Enable some colorization of Git output

```
$ git config --global color.ui auto
```

Edit the global configuration file in a text editor

```
$ git config --global --edit
```

Working with Branches

List all local branches

```
$ git branch
```

List all branches, local and remote

```
$ git branch -av
```

Switch to my_branch, and update working directory

```
$ git checkout my_branch
```

Create a new branch called new_branch

```
$ git checkout -b new_branch
```

Delete the branch called my_branch

```
$ git branch -d my_branch
```

Merge branchA into branchB

```
$ git checkout branchB
$ git merge branchA
```

Tag the current commit

```
$ git tag my_tag
```

Observe your Repository

Show the commit history for the currently active branch

```
$ git log
```

Show the commits on branchA that are not on branchB

```
$ git log branchB..branchA
```

Show the commits that changed file, even across renames

```
$ git log --follow [file]
```

Show the diff of what is in branchA that is not in branchB

```
$ git diff branchB...branchA
```

Show any object in Git in human-readable format

```
$ git show [SHA]
```

Synchronize

Fetch down all the branches from that Git remote

```
$ git fetch [alias]
```

Merge a remote branch into your current branch to bring it up to date

```
$ git merge [alias]/[branch]
# No fast-forward
$ git merge --no-ff [alias]/[branch]
# Only fast-forward
```

Transmit local branch commits to the remote repository branch

Remote

Add a git URL as an alias

```
$ git remote add [alias] [url]
```

Show the names of the remote repositories you've set up

```
$ git remote
```

Show the names and URLs of the remote repositories

Temporary Commits

Save modified and staged changes

```
$ git stash
```

List stack-order of stashed file changes

```
$ git stash list
```

Write working from top of stash stack

```
$ git push [alias] [branch]
```

Fetch and merge any commits from the tracking remote branch

```
$ git pull
```

Merge just one specific commit from another branch to your current branch

```
$ git cherry-pick [commit_id]
```

```
$ git remote -v
```

Remove a remote repository

```
$ git remote rm [remote repo name]
```

Change the URL of the git repo

```
$ git remote set-url origin [git_url]
```

```
$ git stash pop
```

Discard the changes from top of stash stack

```
$ git stash drop
```

Tracking path Changes

Delete the file from project and stage the removal for commit

```
$ git rm [file]
```

Change an existing file path and stage the move

```
$ git mv [existing-path] [new-path]
```

Show all commit logs with indication of any paths that moved

```
$ git log --stat -M
```

Ignoring Files

```
/logs/*

# "!" means don't ignore
!logs/.gitkeep

/# Ignore Mac system files
.DS_store

# Ignore node_modules folder
node_modules

# Ignore SASS config files
.sass-cache
```

A `.gitignore` file specifies intentionally untracked files that Git should ignore

Git Tricks

Rename branch

Renamed to new_name

```
$ git branch -m <new_name>
```

Push and reset

```
$ git push origin -u <new_name>
```

Delete remote branch

```
$ git push origin --delete <old>
```

Log

Search change by content

```
$ git log -S'<a term in the source>'
```

Show changes over time for specific file

```
$ git log -p <file_name>
```

Print out a cool visualization of your log

```
$ git log --pretty=oneline --graph --
decorate --all
```

Branch

List all branches and their upstreams

```
$ git branch -vv
```

Quickly switch to the previous branch

```
$ git checkout -
```

Get only remote branches

```
$ git branch -r
```

Checkout a single file from another branch

```
$ git checkout <branch> -- <file>
```

Rewriting history

Rewrite last commit message

```
$ git commit --amend -m "new message"
```

See also: Rewriting history

Git Aliases

```
git config --global alias.co checkout
git config --global alias.br branch
git config --global alias.ci commit
git config --global alias.st status
```

See also: More Aliases