GIT Help Commands

Getting general help: Help Command Syntax -- getting help about a specific Git command

git help

git help command

Git Configuration Commands

Git Config (Global/User-level) Syntax

git config --global setting value

Configure User and Email

General Syntax:

git config --global user.name "Your Name"

git config --global user.email "you@someplace.com"

Example using course author's information:

git config --global user.name "Jason Taylor"

git config --global user.email "jason@jasongtaylor.com"

Listing All Global Configuration Settings

git config --global --list

Seeing Git's User-based Config file

cat ~/.gitconfig

Git Starting Commands

|  |  |
| --- | --- |
| Command Listing - Fresh Start | Start with Existing Project |
| pwd  cd projects/  git init git-demo | pwd  cd projects/  cd website/  ls  git init |

Present Working Directory

Pwd

Change Directory

cd folder-name

Git initialization

git init [project-name]

*project-name* parameter is optional. If not supplied, Git will initialize the current directory.

Git First Commit Commands

pwd

ls

vi README.md

ls

git status

git add README.md

git status

git commit -m "Initial commit"

clear

git status

List

ls

Lists files and folders in current directory. Without parameters, will list non-hidden folders and files.

Git Status

git status

Shows which files have been modified in the working directory vs Git's staging area.

Git Add

git add file-name

Adds the new or newly modified *file-name* to Git's staging area (index).

Git Commit

git commit -m "A really good commit message"

Commits all files currently in Git's staging area. The -m parameter allows for a commit message directly from the command line.

Clear!

clear

Clears all previous commands from the terminal screen -- just a bit of clean up.

|  |  |
| --- | --- |
| GIT Working Locally, Part One | GIT Working Locally, Part Two |
| pwd  git status  vi README.md  git status  git add README.md  git status  git commit -m "Adding some ipsum"  clear  git status  vi README.md  git status  git commit -am "Adding more ipsum"  git status | pwd  git status  clear  vi index.html  git status  git add index.html  git status  vi README.md  git status  clear  git status  git add README.md  git status  git commit -m "A few changes for the website"  clear  vi README.md  vi index.html  git status  git add .  git status  git commit -m "A few more changes for website"  clear  vi README.md  git status  git add README.md  git status  git reset HEAD README.md  clear  git status  vi README.md  git checkout -- README.md  vi README.md  git status |

Express Commit for Tracked files

git commit -am "Awesome commit message"

Use the *-a* parameter with the **git commit** command to directly commit newly modified tracked files. Warning: Only do this for small changes. Tracked files are files that have been previously added to Git (committed or staged).

Adding All Changed Files

git add .

The period parameter for the git add command will recursively add all new and newly modified files.

Unstage File

git reset HEAD file-name

Following the above command will "unstage" the specified file from Git's staging area (aka index).

Backout Working Directory Changes

git checkout -- file-name

Following the above command will back out any changes made to the specified file and replace it with the version last committed in Git

Git History / File Management Commands

git log

git help log

git log --oneline --graph --decorate --color

|  |  |
| --- | --- |
| Removing Files | Moving Files |
| pwd  git status  vi debug.log  ls  git status  git add .  git status  git commit -m "adding log file that really does not belong here"  clear  git status  git rm debug.log  ls  git status  git commit -m "removing log file"  clear  vi info.log  ls  git add info.log  git commit -m "adding info log"  git status  clear  ls  rm info.log  ls  git status  git add .  git add -u  clear  git status  git commit -m "Removing info.log" | ls  mkdir web  ls  git mv index.html web  cd web/  ll  pwd  cd ..  ls  git status  git commit -m "Moving index.html file to web folder"  clear |
| Ignoring Files |
| vi application.log  ls  git status  vi .iitignore  git status  ls -a  git add .gitignore  clear  git status  git commit -m "adding ignore file" |

Seeing Repository History

git log

git log --oneline --graph --decorate --color

Git's **log** command displays the repository's history in reverse chronological order. The no-params version displays the standard view.

Git log options from above: --oneline Compacts log data on to one line, abbreviating the SHA1 hash --graph Adds asterisk marks and pipes next to each commit to show the branching graph lines --decorate Adds the markers for branch names and tags next to corresponding commits --color Adds some color to the output -- nice to have, depending on the operating system

Removing a file using Git

git rm file-name

Removing a file using Terminal

rm file-name

This removes the file outside Git's knowledge

Updating Git's Index (staging area)

git add -u

The -u parameter will recursively update Git's staging area regarding deleted/moved files outside of Git.

Making a directory (folder)

mkdir folder-name

The **mkdir** command is a nearly universal command for creating a directory/folder.

Making a directory (folder)

The git **mv** command will move

git mv source destination

|  |  |
| --- | --- |
| SSH Authentication Commands | Generating an SSH Key |
| cd ~  cd .ssh  mkdir .ssh  cd .ssh  pwd  ssh-keygen -t rsa -C "jason@jasongtaylor.com"  vi id\_rsa.pub  ssh -T git@github.com | ssh-keygen -t rsa -C "your.name@your-company.com"  Use your actual email address in the example above.  Verify SSH authentication  ssh -T git@github.com  Above command uses **ssh** to connect to GitHub over the SSH protocol. |

Git Remote Commands

git status

git remote add origin git@github.com:scm-ninja/git-demo.git

git remote -v

git push -u origin master

git push origin master

ls

cd web/

vi index.html

clear

git commit -am "Updating index page for GH"

git status

git pull origin master

git push origin master

Creating a remote repository reference

git remote add remote-name remote-repository-location

Using **git remote add** command allows us to associate a remote repository. Normally, you want to paste in the full URL for the remote repository given to you by your Git host (GitHub). By convention, the first or primary remote repository is named origin.

List Git's Remotes

git remote -v

The **git remote** command lists the names of all the remote repositories and the -v parameter (verbose) will display the full URL of the remote repository for each remote name listed

Send Changes to Remote

git push -u remote-name branch-name

git push remote-name branch-name

The **git push** sends all your local changes (commits) on branch branch-name to the remote named remote-name. The **-u** parameter is needed the first time you push a branch to the remote.

Receive Changes from Remote

git pull remote-name branch-name

The **git pull** receives all your remote changes (commits) from the remote named remote-name and on branch branch-name.