skillcrush

GIT cheatsheet

Git, the magical version control

Git is an open-source version control system that allows developers to track changes made on their code files throughout the lifetime of a project.

Git is a great version control tool because it’s lightweight and straightforward to use.

Download & Install Git

Download Git for your operating system here: http://git-scm.com/downloads.

When you install Git, nothing will happen. After you install Git, open your Terminal (or Git Bash on a PC), type “git --version” and hit enter. You should see details about Git, including common commands.

Gitting Started

These are the basic commands you need to create a repository, and then add and commit your first files. Learn more about how they work in Lesson 05.

STEP 1

git init

Before you do anything, you have to initialize (or create) your Git repository. You do this with the `git init` command. It says to your computer, “Hey, this folder I’m in? It’s now a Git repository!”

```
git init
```

STEP 2

git add file
git add path/to/directory
git add .
git add -A

When you’re ready to commit your changes, you first need to stage them by adding them to your directory. You can either add files and directories one by one to your local repository, or you can add everything at once (which you want to do for your first commit). `git add .` adds all the files in a folder inside a directory, and `git add -A` adds everything inside the *entire* directory (and stages deleted files for omission as well).

```
git add index.html
git add www/sample/css
git add .
git add -A
```

STEP 3

git commit -m “Write a message here”

When you’ve added all of your files to your local repository, it’s time to commit. Committing a collection of changes is like taking a snapshot of your work. These snapshots are what get pushed up to your remote repository. Commits are also handy for rolling back your work to a previous version. Be sure to include a message describing what you’re doing!

```
git commit -m “Renaming directory from site to websites”
```

DON’T FORGET!

git status

As you’re working on your project, it’s a good idea to periodically run `git status` to see the changes you’ve made. This will show you all files that have been edited, any new files or directories, and any files or directories that have been deleted. It’s a *very* good idea to run `git status` before you add files and commit changes.

```
git status
```

Branches!

Git is designed such that it views your code files like a “tree” and allows you to do cool things like create a “branch,” where you work on some of the files without affecting the “trunk” code base until you are sufficiently convinced that the changes you’re making are good and won’t break the rest of the tree.

See Lesson 06: Branches, Merging & Errors to learn more about how branches work.

git checkout -b *branchname*

To use branches, the first thing you need to do is create one, which you can do using the `git checkout` command. Using the `-b` flag, This command creates a new local branch and automatically checks it out. The new branch is in essence a copy of whatever branch you were on before you ran the command.

```
git checkout -b v2.0
```

git checkout *branchname*

Checks out an existing branch. To make sure you’re working on files in the right branch, be sure to to use `open .` following this command.

```
git checkout v2.0
```

git merge *branchname*

Merge an existing branch into the current branch. First checkout the branch you want to merge *into*.

```
git checkout master
git merge v2.0
```

git branch -a

Shows all existing branches and highlights the current branch. *Remotes* refer to branches that exist on an external repository, like on GitHub.

```
git branch -a
* master
  v2.0
  remotes/origin/HEAD -> origin/master
```

git branch -d *branchname*

Deletes the specified branch. **Use only if you're sure you want to delete it!**

```
git branch -d v2.0
```

Sign up for GitHub

A great place to store your repositories is on GitHub. Setting up an account is free, and you can have as many public repositories as you want.

GItHub is also a great way to build a resume for yourself. Often when developers are applying for jobs, they simply share their GitHub profile link to show potential employers what they've done.

For more info on how to set up GitHub, see Lesson 07: Introducing GitHub

Remote Repositories & GitHub

git clone git@github.com:username/reponame.git path/to/local/dir

Clones a remote repository to your local environment, into the specified directory.

```
git clone git@github.com:skillcrush/my-website.git www/sample
```

git remote add origin git@github.com:username/reponame.git

Adds a remote repository to your existing local repository. This allows you to push code up to the remote.

```
git remote add origin git@github.com:skillcrush/my-website.git
```

git pull origin master

git pull origin *branchname*

Pulls code from the remote specified branch and merges it into the current local branch.

```
git pull origin refactor
```

git push origin master

git push origin *branchname*

Pushes code up to the remote specified branch.

```
git push origin branchname
```

git remote -v

Checks what remote you are working with and tells you the address.

```
git remote -v
origin  git@github.com:skillcrush/my-website.git (fetch)
origin  git@github.com:skillcrush/my-website.git (push)
```

git remote rm origin

git remote rm *name-of-remote*

Removes the specified remote connection and associated branch references.

```
git remote rm origin
```

More Commands!

We haven't covered these commands, but at some point you just might need them.

git checkout path/to/file

If you've made a complete mess of a file, and it's past the point of saving, you can easily revert back to the last locally committed version of the file by checking it out.

```
git checkout www/sample/index.html
```

.gitignore

Sometimes you may have files or entire directories that you want to keep out of the remote repository while still keeping them in your local repository. The `.gitignore` file is used to list these files and directories. Add one file or directory per line.

```
.htaccess
/app/config
```

git diff path/to/file

This command shows the changes you've made to the specified file by diffing your local version with the last committed version. It will show you code that has been added and deleted. It's good practice to run `git diff` before you add a file for a commit.

```
git diff www/sample/index.html
```

More Information about Git & GitHub

For more information about Git, check out the Git documentation.

For more information about GitHub, check out GitHub's help guide.