



A Guide to Git and Github Table of Content_version_2

GETTING STARTED

Introduction

Source Control Management

Code Repositories

What is Git?

Local vs. Remote

What is GitHub?

WORKING LOCALLY

Creating Repositories

git init

The config File

.gitignore

Tracking Change

Creating a commit

git log

Summary

Branching

What is Branching?

Why Would I Want to Branch?

Command Reference

Command

Purpose

REMOTE REPOSITORIES

Connecting to GitHub Repo

GitHub Authentications

git push

git pull

git clone

Summary

CONCLUSION

Cheatsheets

GETTING STARTED

Introduction

Source Control Management

Source control Management is the process programmers follow to iterate through their projects or codebases.
Source control/Version control is the practice of tracking and managing changes to software code.

Code Repositories

repositories = directory and files

What is Git?

one of the most popular SCM tool
dvcs

Local vs. Remote

local repository = located on your computer
remote repository = located on an other computer or server

What is GitHub?

is an ecosystem and user interface for software projects

WORKING LOCALLY

Creating Repositories

git init

git init # initializes a new reporitory by adding the .git directory
Remember to create a README.md a LICENSE.md and a .gitignore directory
for every git repo you're working with.
Do not nest your repositories!

The config File

Location	Description	code sample
<pre>/etc/gitconfig .</pre>	Contains settings that are system-wide and apply to all users and all if their repositories.	<pre>git config --system user.name 'bill' # saves user name to system wide # config file /etc/gitconfig .</pre>

<pre>~/.gitconfig .</pre>	<p>Configuration that is specific to your user account. This file overrides/etc/gitconfig</p>	<pre>git config --global user.name 'bill' # saves user name to user wide # config file ~/.gitconfig .</pre>
<pre>REPO/.git/config .</pre>	<p>Configuration that is specific to a repository. this file overrides both of the other files.</p>	<pre>git config user.name "bill zachary" # saves user name current repo's # config file /.git/config .</pre>

.gitignore

is a list of files and/or directories that you do not want included in your repository (private data such databases info...).
 Remenber to create a .gitignore file for every git repo you're working with

Tracking Change

Creating a commit

is a bundle of changes
 Any unstaged changes are not tracked by git!

git log

git log # history of commits in this repository
 commit hygiene is highly regarded is some teams so it's important to make sure each commit contains a logical grouping of changes (eg, a bug fix or a feature) with a clear commit message!

Summary

Command Description

<pre>git status .</pre>	<p>Run this command any time and often to check on the status of the files in the git repository.</p>
<pre>git add .</pre>	<p>This command stages changed files, readying them to be wrapped into the next commit.</p>

<code>git commit</code> . .	This command commits staged files, wrapping them into a commit. A historical record of commits is what we refer to as a codebase's version or commit history.
<code>git log</code>	View the repository's commit history.

Branching

What is Branching?

a branch is a copy of all the files in your codebase.

Why Would I Want to Branch?

to create a branch we fork it from another existing branch, likely main. If we decide the experiment is a sucess we can merge commits from the experimental branch into our main branch.
--

Command Reference

Command	Purpose
---------	---------

<code>git init</code> .	Creates a new reporsitory in the current directory. Generates a .git directory.
<code>git config</code>	Sets git configuration settings, such as author email and name.
<code>git status</code> . .	Shows the working directory as well as the staged changes. Run this command liberally to see which changes are ready for committing.
<code>git add FILENAME</code> .	Stage file changes, which prepares them to be added to the repository.
<code>git commit -m "message"</code>	Creates a commit from staged files. Commits are the atomic unit in git that gets moved around

.

between branches and repositories.

REMOTE REPOSITORIES

Connecting to GitHub Repo

```
1) creating a remote repo on Github
2) connecting my local repo to the Github remote repo:
make sure that your local repo contains the README.md, LICENSE.md and .gitignore
files and directory
git remote add origin git@github.com:wguiraud/remote_repo_name.git
```

GitHub Authentications

TO DO LATER ON => <https://launchschool.com/books/git/read/github#githubauth>
FOR NOW I STICK TO SSH.

git push

```
git push -u origin main # origin is an alias for the remote repo name
```

git pull

```
# checking remote repo for changes
git fetch

# checking what has changed
git diff main origin/main # main = local repository, origin/main = remote repo
=> @@ -0,0 +1 @@ # + represents added line from the repo specified second on
# the command line, while - represents deleted lines from the repo specified
# first.

# pulling from remote repo origin from the main branch
git pull --ff-only origin main # --ff = fast-forward merge to combine the changes
# (moves the history of the local branch forward to match the history fetched
# from the remote branch) -only = if git can't combine the changes with the ff
# merge it should abort the merge
```

git clone

```
# pulling down all the content of a remote repo
git clone <remote repo url> <local repo name>
```

Summary

Command	When to use
<code>git init</code>	Create a new local repository.
<code>git remote add origin REMOTE_URL .</code>	Add an existing remote repo as a remote of existing local repo.
<code>git clone</code>	Pull down contents of existing remote repo into a new local repo, and add a remote ro the local repo pointing to remote repo.

Let's rehash what just happened in this chapter:

- First, we created a repository on github.com, and chose not to initialize it with any files, leaving it a blank slate.
- Next, we made our local repo aware of our newly created remote repo with `git remote add origin REMOTEURL`.
- Then, we pushed some commits from our local repo to our remote GitHub repo using `git push`.
- We then made a modification on github.com directly, simulating a coworker pushing commits to our remote github.com repo.
- Finally, we pulled those commits down to our local repository using `git pull`, syncing up our local repo with our remote repo.
- We also learned what `git clone` is, and how you would use it to work on an existing git repository.

CONCLUSION

Cheatsheets

