git_version_control

November 19, 2017

1 Git Version Control

Video Tutorial by: Tommy Morgan of TeamTreehouse.com

1.1 Working With Git Repositories

- Use `git --version` to find out if Git is installed. If it's not, install it using sudo apt-get install git.
- `git init [project_name]` - initializes a new repository. If project_name is provided, it creates a new project directory with that name. If not, it initializes a repository in the current directory.

Unix Command Refresher

- `cd` - think 'change directory:' change the current working directory. Remember that ~ is a special symbol that always represents your "home" directory.
- `ls` - think 'list:' shows a list of all files/folders in the current directory. With the -a flag, also shows hidden files and folders.
- `mkdir` - think 'make directory:' creates a new directory with the specified name.
- `touch` - updates the "last modified" timestamp on a file to now. Also creates an empty file if the filename specified doesn't exist.
- `mv` - think 'move:' moves a file or directory to a new location. This also makes it a convenient way to rename files and folders.
- `rm` - think 'remove:' deletes the file(s)/folder(s) specified.

1.2 Commiting Changes

Editing Files I'm using the nano text editor because it's the default for the Treehouse console. If you're following along in Windows, remember to use 'notepad' every time I use 'nano,' because nano isn't available on your system. If you're following along on your own Mac/Linux machine, nano may not be your default editor. If you want to set nano as your default editor so that Git will use it, run export EDITOR=nano before you try to run any of these commands. If you're curious about what this does, there's an article explaining it here. * `git add` - adds files to the repository so that Git knows to track their changes. * `git commit` - commits all added files to the repository as a change. With the -a flag, commits all changes to all tracked files. With the -m flag, allows you to specify a commit message directly on the command line instead of in your default editor. * `git config` - allows you to make configuration changes to Git. With the `--global` flag, makes these changes available across your entire system. * Change Name for commits:

```
git config --global user.name <your_name>
```

- Change email for commits:

```
git config --global user.email <your_email>
```

- Enable helpful colorization of command line output

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

1.3 The Staging Area

`git status` - show the current status of the git repository, including if there are any uncommitted changes and whether or not any of our changes have been put in the staging area.

1.4 Looking Back on What We've Done

- `git log` - Show us a chronological log of all of our commits to the current repository.
- `git checkout` - "check out" a different version of the code from the one you're currently looking at.
- Example:

```
git checkout <1st_5digits_of_commit_id>
```

- `git diff` - create a "diff" view to demonstrate what has changed between two different versions of your repository.
- Example:

```
git checkout <commit_id_for_file_1> <commit_id_for_file_2>
```

- HEAD~1 is a special commit identifier in git; it stands for the previous commit (not the one we just made, but the one before that)

1.5 Beginning to Branch

- `git branch <branchname>` - create a new branch named branchname.
- `git checkout <branchname>` - switch to the branch named branchname.
- `git checkout -b <branchname>` - create a new branch named branchname and switch to that branch.

1.6 Managing Our Branches

- `git branch` - list all branches in the current repository and indicate which branch you're currently in.
- `git branch -D branchname` - delete the branch named branchname from the repository.

1.7 Basic Merging

- `git merge branchname` - merge the history from branchname into the current branch.

1.8 Cloning

- `git clone` - create a new repository that is a clone of a remote repository.
- `git remote` - list all remote repositories associated with the current repository.
- `git remote add` - add a new remote repository to the current repository.

1.9 Pushing and Pulling

- `git push` - push your latest changes to a remote repository.
- `git pull` - pull the latest changes from a remote repository to your repository.

1.10 Github

- GitHub help - Pull Requests
- GitHub tutorial - forking a repository and contributing to a project

1.11 Git flow

- The git-flow project on GitHub
- The branching model that git-flow is based on
- A blog post describing how git-flow works