Intro to Version Control

 (https://github.com/learn-co-curriculum/git-github-intro-to-version-control)  (https://github.com/learn-co-curriculum/git-github-intro-to-version-control/issues/new)

Learning Goals

- Define the purpose of a version control system.
- Identify the benefits of version control systems.
- Recognize useful Git vocabulary terms.

Introduction

Imagine there is a team of developers working on a popular web application with many features. The app may have a large, complex code base which means that, at any given time, there are likely to be multiple projects going on and multiple developers working on different parts of the software.

This type of scenario is quite common, and can easily lead to problems:

- Imagine someone is working on a new feature, or incorporating a new technique, or refactoring the code to make it more efficient and something they change breaks the app. How can they "back up" to the most recent working version of the code?
- There could be some part of the application that overlaps with the work being done by more than one developer. How can they ensure that no inconsistencies or conflicts work their way into the code?
- How can they maintain a record of the changes that are being made and why? Over time, memories about how things have been done can fade, or new people may join the team who don't have that background knowledge. This can lead to inconsistencies in the code or to wasted time revisiting issues that have been addressed in the past.

The great news is that *all* of these situations can be managed by a type of software called a **version control system (VCS)**.

Define the Purpose of a Version Control System

Version Control System (VCS) describes a type of software that is designed specifically to help manage the complexities described above - and so much more! The key benefit of version control is that it keeps a complete history of the changes to the code, which makes it easy to "back up" to a working state if things go wrong. This gives you the freedom to experiment, throw away bad ideas, and instantly get back to your last-known "good" state if something breaks.

> **ASIDE** The programmer, entrepreneur, and venture capitalist Paul Graham notes that oil paints unlocked a revolution in experimentation in visual arts because they were undo-able. Oils provided the **freedom** to err and recover that other paint media did not provide (e.g. watercolor). Because of this **freedom** these painters were free to explore perspective, light, and composition in completely new ways. Because of their *tools* they had more *freedom* and were able to make their burst of exploration a *movement*: The Renaissance.
>
> Read more in his essay **"Hackers and Painters." (http://www.paulgraham.com/hp.html)**

There are several VCS's available, but the most popular — and the one you'll learn to use in this module — is called **Git**.

Identify Benefits of Version Control Systems

There are a number of benefits we get when we use a VCS such as Git to manage our work:

- Automatically create a backup of our project.
- Provide an easy way to undo mistakes and restore a previous version of the project.
- Document changes with a log that describes what's been changed and why.
- Easily view the differences between two versions of the project.
- Branch work off into multiple "sandboxes" (called **branches** in Git) that allow developers to experiment without impacting other branches.
- Collaborate with others without disturbing each other's or our own work.

And beyond these are even more advanced features that will help you optimize your workflow once you've learned the basics. If that feels daunting, it's OK: you will build up your toolbox of techniques over time. In the meantime, though, you can get the best benefits of Git using a relatively small subset of all the techniques and commands it includes.

Recognize Useful Git Vocabulary Terms

We're about to get busy learning Git, but we first need to establish some common vocabulary. Git, perhaps more than any other software, has some special words that you'll hear a lot. Don't worry if you're not sure how some of these terms work in practice — that part will come later.

- **repository** (or **repo**, for short): A directory of files that are **tracked** by Git.
- **track**: When a file is **tracked** by Git, it means that Git will notice any changes to that file. We call these changes **differences** or **diffs**. Git allows you to choose whether to **commit** a diff in order to keep it.
- **diff**: The **diff** of a *file* is all the changes that have been made to it since the last **commit**. The **diff** of a *repo* is all the diffs in all the *tracked* files in the repo that have not yet been committed (sometimes programmers call this the **diffset**).
- **commit**: Once we decide we want to save a diff, we **commit** the diff to the repo's history using the `commit` command. When we make a commit, we write a **log** message that describes what happened in the diff. The set of commits provides a history of all of the changes that have been made to a repo and when.
- **log**: The record of what happened in each commit.
- **local/remote**: When we start working with an existing Git repo, we **clone** it from a **remote** source (on GitHub) and copy it to our machine. We call the repo on our personal system the **local** repo.
- **branch/default branch**: A Git repo can support multiple **branches** that make it possible for multiple developers to be working on the code at the same time. When you initialize a new Git repo, a **default branch** is created where your work will be tracked. Historically, Git has used `master` as the default name of the default branch, but many organizations in the Git community, including GitHub, are moving away from using `master` and using `main` instead. You will still see `master` used in many repos, especially older ones.

Note: depending on how your Git installation is configured, your default may be either `main` or `master`. If you'd like to set up your Git to use `main` by default (which is what you'll see in these course materials), you can run the following command in your terminal:

```
git config --global init.defaultBranch main
```

Check for Understanding

Before moving on to the next lesson, check for your understanding of this material by answering the following questions in your own words:

- What is the purpose of version control?

- What are the benefits of version control?
- What are some new Git vocab terms you've learned?

Conclusion

Git is a widely-used and very valuable tool for managing changes and versions of projects, as well as for collaborating. Because many companies in the tech industry use Git, particularly in software engineering, it's important that you get used to working with it. In this module, you will learn many of the most commonly used workflows and develop the skills that will be expected by many potential employers.

Resources

- **Getting Started - About Version Control (http://git-scm.com/book/en/Getting-Started-About-Version-Control)**
- **Git Basics - What is Git? (http://git-scm.com/video/what-is-git)**