Course Title: Learning GitHub

Description: GitHub is the industry-standard tool for collaborating on and sharing code. It’s popular among software developers, project managers, designers, and students for its flexibility and control. This course introduces GitHub and Git, the version control system that GitHub is built upon. Instructor Aaron Stewart, a training content specialist at GitHub, explains the benefits of version control, how to navigate GitHub and the command line, how GitHub and Git are related, and the best practices for communication and collaboration on GitHub. Aaron also reviews the most common tasks, such as branching, commits, and pull requests, and shows you how to create a simple local project and move it to GitHub to share with your team. Once you have completed the course, you should be able to immediately start using Git and GitHub to manage your own code.

****
Chapter: 1. The Basics of Working on GitHub
****

Video: What is GitHub?

Note Time:	Note Text:

0:00:00	Git- (distributed) version control system (VCS) . Records changes made to a file and allows users to view/work on previous versions. Distributed- allows collaborators of a repository to have access to a project's entire history and back that data up to their own computers. Allows for many different project work flows.

Video: Exploring GitHub

Note Time:	Note Text:

0:03:15	README.md file present in (and displayed at the bottom of) all Github repositories (recommended). Used to explain the project, provide helpful tips for contributors/collaborators

0:04:04	Issues: used to track bugs, feature requests, anything to discuss but not assigned to code. Can be assigned to specific collaborators/teams and added to projects

0:05:08	Pull requests: changes like adding/modifying/ deleting files that you'd like to make to the repository. Shows status of automated tests you may have in project. Difference between

"Issues": actually has code attached

Video: The GitHub workflow: Idea to commit

Note Time: Note Text:

0:04:02 Master branch: provides record of all the "branches"/changes made to a project over time. First branch created by Git- live "production branch". Create branches outside of master, make changes to the project on that branch, then add "commits" (actually make those changes) and test them out on the project, without modifying master branch

---
Video: The GitHub workflow: Pull request to production

Note Time: Note Text:

0:01:32 Pull request: allows for collaborators to see and compare commits/changes made on your local branch to those on the master/production branch. Signals you want to add them to the production branch, and allows others to discuss/review your changes before doing so

0:03:10 Last step: deploy and merge your feature branch into master

****
Chapter: 2. Working Locally with the Command Line
****

Video: Why you should love the command line

Note Time: Note Text:

0:01:47 Can run all Git commands on the Command Line

Video: Preparing to use the command line

Note Time: Note Text:

0:02:43 Command Line (on Mac): pwd- print working directory

(see current file path); cd ..- change working directory; ls- list contents of current directory; touch- create empty file; mkdir- create an empty folder (directory)

Video: Configuring some Git options

Note Time:	Note Text:
0:01:22	Levels of Git configurations: System- all users on one system/computer; Global- all Git repositories of one user; Local- single repository level

---

Video: Create your first branch

Note Time:	Note Text:
0:03:57	To create a branch in the command line: "git branch branch_name"
0:04:06	"git branch --all" returns a list of all the branches in that project
0:05:26	"git push --set-upstream origin my-slide" command "pushes" new branch onto Github- makes it visible on Github repository

---

Video: Making your first commit

Note Time:	Note Text:
0:02:18	"git status" command tells us the current status of our project- sees current branch, if you're up to date with the origin project on Github, and if you've made any local changes that you need to push up to Github
0:03:04	To add new file: command "touch name_of_file"
0:03:16	To cd out of a folder: "cd .." ; now should be out of _posts folder and back into github-slideshow project repository
0:04:54	"git add name_of_file" command adds file to staging area, then to commit this change, command "git commit -m "Add slide for vgoris", then run "git push" to push this change up to Github

---
Video: Understanding where file changes go

Note Time: Note Text:

0:02:24 Three stages of a File = "2-stage commit": Working directory (new files, files being modified; nothing gets published to project history or pushed up to Github), Staging directory (move to staging area when you want to move forward with changes made to a file; use "git add" command for this, then run "git commit" to commit all changes currently together in the staging area); History ("git commit" command; push changes up to Github to incorporate them in Github repository project, and they are now part of your local project's history)

Video: Merge your pull request

Note Time: Note Text:

0:02:52 To merge branch after pull request has been approved: checkout to main branch, "git merge name_of_branch", "git push" to push this merge up to Github (for now the change has only been done locally), then delete branch locally with "git branch -d name_of_branch"

****
Chapter: 3. Working Locally with GitHub Desktop
****

Video: Understanding remote and local

Note Time: Note Text:

0:02:44 We have local branches (on our computers) and remote branches (on Github). Then remote tracking branches = reference pointers to the state of the branches on remote repository on Github (bookmarks)- local branches that are directly related to a remote branch on Github; automatically created when checking out a local branch; Git knows which branch to push to

---
Video: Tools for working locally with Git

Note Time: Note Text:

0:02:21 IDEs: visual studio, atom.io, Xcode, Eclipse, for example

---
Video: Create a branch and make a commit

Note Time: Note Text:

0:05:00 Can commit changes separately using Github desktop; on command line, modified changes are committed together

****
Chapter: 4. Moving Your Project to GitHub
****

Video: Create a new project

Note Time: Note Text:

0:01:37 Command line: create a new project "git init repository_name"; make sure it's getting saved to the right folder/ location on computer

****
Chapter: 5. Working with Others on GitHub
****

Video: Add headers to unformatted content

Note Time: Note Text:

0:00:27 Add headers to text on Github by adding hashtags (##) in front of title, with an <h1> header (#) being the largest, <h6> (######) being the smallest

---
Video: Use bold and italics in text

Note Time: Note Text:

0:00:39 To italicize: surround the text with a * or _ on

each side; to bold: ** or __ on each side