ANDREW RINGLER: It's like a file sharing system, like Dropbox, except you manually take care of the files yourself and you worry about merge conflicts. So if in Dropbox, you and your collaborator added a document at the same time, it tries to save both and it goes, aah, something's wrong. And it makes one that says "conflicted copy."

And GitHub does a similar thing but then it hands it back to you and it says, I'm not going to save that conflicted copy, you need to tell me what to do to resolve that so that I only have one version of each file and it's the master version. So GitHub, in a nutshell, is just a way to share files with other people collaboratively without overwriting other people's stuff and without having conflicts happen inside of the cloud where you're able to share this document.

So what I'm going to go over today is GitHub and then briefly how to add pages and upload them. So what is GitHub? Does anyone want to explain what GitHub is? Or Git? The wild guess. OK. GitHub is a few things.

GitHub is a web-based platform, a website for sharing code, sharing software code-- the source code of software. So it's created for developers to collaborate on code. A lot of code on there is open source code-- code that the source code is freely available to look at.

So basically, it's a website where you can navigate other people's code, you can collaborate on the same project. And you can also just look at other people's projects so you can submit bug reports to projects. So GitHub is the sort of community.

The other thing it is is it's a piece of software based on a technology called Git, G-I-T. Git is a piece of software that lets you collaborate, basically lets you share source code. And GitHub kind of extends that by making a pretty website and a few other things.

So Git is a piece of software called version control. Has anyone heard that term before? Yes? Does anyone want to explain that term-- version control? It is a very fancy way of saying keeping a history and storing multiple versions of things.

So for example, say, you are working on some sort of document-- maybe a really important essay. So this is my really important essay. "My essay on bugs by Andrew." So you're working on this essay, you save the essay. Then you make some changes, you save it again. And then you're like, oh, what was the first version of the essay I did? It looks so different now.

So one of the things you'll do is once you have a version you like, maybe you'll take that version and let's make a copy of it. Let's call it like v1-- version 1. Now let's start working on version 2. So if we mess something up, we still have the first version. And then it's getting really good. So then at this point, we save our version 2. Let's call this one Final.

So as you work on multiple versions of the document, you want to save different versions in case you mess up. You might have done that with an essay before. You might submit an essay to professor. They give you back a different version with corrected changes-- an annotated version. So version control in software is just a fancy way of saying the same thing-- it's making multiple versions of things that you can go back to and look at.

The thing with Git, though, is you don't actually have to do what I just did. It does all that automatically for you. And that's similar to-- has anyone heard of Time Machine for your Apple? Time Machine on OS X is software product that basically backs up your computer every hour, and then if something messes up, you can restore your computer to an earlier version. It has a nice little interface like with a little time machine. You can go back and look at previous versions of your computer.

GitHub is really similar. It's very similar, except it's tuned for working well with source code, so it has a lot of other features that are useful for software developers. So if you're programming, eventually you will have to learn Git at some point. And GitHub is like the most popular way of using Git, that's why people treat them as almost the same thing.

So let's take a look. The way GitHub organizes things is it has these things called repositories. And repository, you could think of it as like a folder or a project. So every different project you'd want to different repository. So for this course website we have one called mens-et-manus. So where is that? Ahh, here it is.

One of the things GitHub does is it provides this fancy home page for your projects, so you can document your projects on the homepage. GitHub has a few ways of interacting with it. You can look at your files from the web interface, GitHub.com. There is also text-based interfaces and a visual interface of GitHub Desktop, which we'll use shortly.

A few things to note. So this is the current, latest version of your files. This is a view of the latest version. Unlike Time Machine, it doesn't automatically create a new version every hour. And if you used Dropbox, Dropbox creates a new version of your file every time you change it.

GitHub does not do that because it wants you to explicitly say when you've made a working change. Because if you're changing a piece of software, you might change three or four files until you have a working next version of your software. So you basically have to tell Git every time you have a new version that works. And that's called the commit. So the commits are basically the history of every time you've made a change and told GitHub about it

AUDIENCE: Excuse me. I would like to know if you made an account there and put the files for the course, uploaded them there.

ANDREW RINGLER: Yes. Yes

AUDIENCE: And if we sign, we can view this if we search for it. Do we have to have access? Do you have to give us access?

ANDREW RINGLER: I do, yes. You'll notice a few things. There's a little lock icon and there's also this text that says "private." So yes, until I add you, you will not be able to see this URL.

AUDIENCE: Are you going to?

ANDREW RINGLER: Twist my arm. Yes.

AUDIENCE: No, I wonder if we are supposed to use anything in there.

ANDREW RINGLER: Yes, at some point I will ask you all to email me your GitHub user names and [INAUDIBLE]. OK. So the commits are basically the history. It's good to name them well. So I said, "Link to electronic books and schedule." That's when I added this schedule to the website. "Added a group project example"-- that was five hours ago. So that's when I added this example of group project.

So I basically made changes and notated the change. So you can view the whole history of stuff on GitHub. There's also a ton of other features which we don't use. You could submit bug reports on their issues. You can create a wiki for your project, which most people do not do.

All right. The web interface-- you only really want to use it as a view of your files. You don't actually upload files there. There is this Upload button and Create a New File button, but you don't want to use that. What we actually do want to use is this program called GitHub Desktop.

GitHub Desktop is similar to Dropbox. Has anyone not heard of Dropbox? Has anyone heard of Dropbox?

AUDIENCE: Yes.

ANDREW RINGLER: Yes, it had an IPO, it's a big company. So Dropbox-- yes, here you go. So Dropbox has a few things. It has a sleek little program that runs continuously. And what it does is it takes a folder on your hard disk. And any file you put in it, Dropbox will upload it to the cloud. And if you have Dropbox on two computers, it will automatically sync them so you can have two folders look the same.

GitHub Desktop is very similar to that, except it does not do anything automatically. If you drop files in a folder, it will detect them, but it won't automatically push them up to the cloud. You have to do that manually. The reason is-- what I said before-- is because you need to really make sure this is a change that won't break the software, so forcing developers to write out the change, helps make the software easier to maintain.

OK, so in a moment-- I'll run through this quickly. Let me tell you all the steps that have to happen. In a moment, on your own, you will create an account on GitHub.com. It's a free account. You can use any email address you want. You'll then download GitHub Desktop-- that is also free software.

Once you've done that, you can log into GitHub Desktop and sync your cloud account with this piece of software locally. Once you log in, you basically then choose which repositories you want to track.

And to do that, we hit the Plus button, we say Clone. Then we search for the repository. We clone it. We choose a location to save it. I'll run. through this really quickly and then I'll step back for everyone to do it.

So what it's doing, a clone is basically making a copy of the repository. So there's a version in my cloud at GitHub in California, I guess. So we're downloading that. Once we downloaded it, we could see a very similar view to what we saw online. Here "Link to electronic books and schedule"-- that's same thing we saw on the web page. We'll say, "No uncommitted changes." Great.

As I mentioned, if you drop a file in the folder-- GitHub Desktop has created this folder for me, just like a normal folder. And these are all the files in it. So if we drag a new file into it, just like with Dropbox-- let's drag up the final version of an essay. So I dragged it into our repository folder locally on my hard drive. If we search back to GitHub Desktop, it's detected that change.

Then we want to commit. So I said, "Added my final essay version." You always have to write out a commit message. It won't let you commit without a message. So I said, "Added final version." I'll say Commit. That actually committed locally on your computer. So it creates the whole history.

Then we have to do another step-- sync-- which syncs my history on my laptop with the universal history on GitHub.com in California. So once I hit Sync-- if we go back to the web page and reload it, we can see the essay is up here. And we now have 566 commits. The last one-- "Andrew Ringler committed a minute ago," and my file essay version. That's basically the idea of the whole flow of files on GitHub. Any questions so far?