[MUSIC] Okay, so I'm back in my Mac and

I'm actually looking at github.com. And I'm already logged in into GitHub. So the first thing we're going to

do is we're going to go over how to create your own repository on GitHub. And then how to turn that repository into

something you could store your code in your website code in. And then have that same repository turn

around and host your website for you. As part of this demonstration,

I'm obviously going to use some git commands in

order to commit, in order to save, and in order to upload our code that is

locally in our machine to github.com. Now, I highly encourage you to

actually learn get pretty in-depth. I'm not going to have a whole course here

in just a lecture or so on Git, I will carry give you enough of the commands so

you'll be able to be functional. But I would very highly

encourage you to go and read it at least through the branches

chapter of the Git book. And the way you could do

that is it's free online. You can just say Git book. If you say Git book, the very first thing

you should come up as this Git book link. And this is a book that if you

wanted to actually get a PDF or you could read the entire thing online. And I strongly suggest that you

read the first three chapters. It's really not that complicated, but

for our purposes as long as you know the commands that I'm going to give you,

you should be okay. But if you want to become a full stack

developer or even just a front-end developer, you should definitely try

to learn Git and its basic commands. Okay, so I am in my account right now. This is my ychikin and this is my account. And probably you're going to see

something very similar, but you probably aren't going to have as many things going

on possibly in your GitHub account. So the first thing you want to do

is create a GitHub repository. This is how you're going to basically

submit your assignments as well. So you could go to your picture,

your icon whatever it is, and there's a little icon next to it

to the left of it and it's a plus. And if you click on it, click on the down

arrow you could say New repository. And I'm go ahead and create a new

repository, we'll call it coursera-test. And this is a cavity try that before so

it's already knows that I've trie that. So coursera-test and I'll give you a green

check mark saying, yes, it's available. And I'll just say that

this is a Coursera test repository and it's going to be public. If you want to make it private,

you have to pay. We're not going to want to pay, so we're

going to just make it leave it public. And we'll go ahead and initialize it with

a README file always always a good idea. And all you have to do now

is click Create repository. Let's go ahead and do that. And voila,

I have my own repository ready to go. So once I have this repository, I can

start interacting with it not only online here, but also through my command line

prompt and upload code to this repository. But before we do any of that, let's prepare this repository to host

our actual website to the outside world. In order to do that,

we have to do a couple of special things. Number one is we have to go to Settings. We'll go ahead and go right here on

the top right, we'll click on Settings. And if we scroll down, you'll see here

GitHub Pages, Automatic page generator. Basically, it says create

a beautiful website or site for your project with our automatic

GitHub page generator. Okay, so all you really have to do is

click this button launch Automatic page generator. And it doesn't really

matter what this says, you could certainly customize that later. You could scroll down all the way down and

say Continue to layouts. And by default, it's going to give you

this pretty neat way out actually, and you don't have to even check Iit out,

you could just click Publish page. So the layout is not

really what we're after. So once we're done with that, you'll see

here that as your project has been created at HTTP and you'll see your username,

your GitHub username. And then the URL is

github.io/coursera-test. That's where your page is located. So if I go right now and copy this link

right here, and we'll go to a new tab. You're really going to see the entire

website that we just so right here, right? Then it's so pre-filled for you already. Now, this link will not always stay here. If you refresh the browser, it's gone. So how do we you if you

forget where that link is? Where do you find it? Well, you go back to Settings and

you scroll down, and under GitHub Pages you'll

see that link again. You can actually click it,

open it up, and there it is again. Now, when you created the GitHub pages, let's go back to our repository,

we'll click on this coursera-test. So now when you created that

link what you actually did is, you created a separate branch. Now, I'm not going to go into, again,

the whole big explanation about branches. But the bottom line is the way Git

works is it can have several branches like several parallel streams

almost of the same repository, of the same kind of database of your code. So the way you see what branches

you have is right here branch, I know your kind of see it says master. But you click on it and you'll see that there's another branch

here that actually wasn't there before. If we opened up before we

would have seen only master. But now there's another

one called gh-pages. If you click on that gh-pages, you'll see all the stuff here you

really didn't put in here before. Well, with this stuff is that

website that was created for you. So that index.html is that big page or

the giant green banner that you've seen. So now, if you're an advanced GitHub user,

feel free to use whatever branch you want. And then go ahead and

merge your stuff into the gh-pages branch. B ut just know that whatever

is in gh-pages branch, that's the only thing that's going to show

up on that special link that you created. Nothing in the master, nothing any

other branches will actually show up. So for the beginners, what I would suggest

doing is that you should always use gh-pages and commit everything there. But just remember that the base directory

is that website that got created. So if I were you,

I'll probably create a subfolder and put everything into that subfolder. And leave this front page alone, or if you

want you could wipe the whole thing out and just create your own page

right here in the root folder. But for now, it's time to go ahead and

take this repository and bring it down to our local machine. Well, the way we could do that

is we have right here this URL. This URL actually points

to our repository. You can see it has coursera-test.git. And we could copy this URL by

just simply clicking this link. We'll go ahead and open the terminal. So now what we'll do is

we'll say git clone. Clone is the command to actually take

the repository that is somewhere remotely. And just provide the URL

to the repository. And you can see we're done. If we do a list now,

you'll see that I have coursera-test. That is our repository right there. Let's go ahead and cd into it. And you'll see that the only

thing there it's README. Okay, so really what's going

on is if do git status, you'll see that we are on branch master

and nothing really is right now change. The thing is though, we weren't

really want to be on branch master. Again, unless you are you

know more advanced user, really want to be on the branch gh-pages. And the way we could do that is

to say git checkout gh-pages. And when we do that,

now we say git status. We're going to be on gh-pages. And if you do ls, you'll see all

that index.html params.json and stylesheets folder. It's all there from,

if we switch back to the browser, it's all right here in that branch. So now we have a couple of choices. If we want, we could just go ahead and

wipe out this whole thing. And we'll basically wipe out

that auto-generated website or we could leave it alone. And just basically make a directory and

then call it, let's say, site and

cd into that site directory. And there's nothing there, great. So now we could go ahead and

open up sublime text. And we'll save this page right here. We'll go ahead and save this page too, now at coursera > coursera-test and site. We'll save it and call it index.html. And we could quickly do html and do Ctrl

+ Space for a quick little template. We'll say Hello Coursera and we'll have h1 here,

also say Hello Coursera. Okay, so we have a very quick page,

we saved it. Let's go back to our terminal. We'll do an ls, and

we see this index.html. So now to check what's going on in our

local git, we'll say git status again. And we'll see that actually ./ and that's

because we're inside the site directory, we say that that / is not

actually committed at all. Well, we could go one directory up and

then we'll do git status and that will tell you that the site

directory is not even committed and it's not even marked to be committed. So the way Git works in

order to be committed into kind that your code database, which

actually resides locally in your machine. It is not the remote one,

not to GitHub, but the local in your machine is you first

have to mark something for committing. The way you mark something for committing

is you say git add and then you say what it is you want to commit or

that is you want to mark for committing. And this why I'll say dot and will just

include everything in my local folder, and that's done. I'll say git status again and

you'll see now what's marked for to be committed is that

new file site/index.html. And once again,

we haven't committed anything yet. We just marked it to be committed. Well, in order to actually

commit it we'll say git commit, we'll say- m which is required for

message and we'll give it a message. The message say My First page. So we'll press Enter, and we're done. And the only thing we need to do now

is actually put this into our remote, meaning into our GitHub

on the remote repository. Right now everything is sitting

in the local repository. If we say git status again,

you'll see that everything is great, except it'll tell you that your branch, meaning this origin/gh-pages don't

worry about this origin thing. Just ignore it for now, but

gh-pages is ahead by 1 commit, meaning our local version of our

repository with all of our code is actually ahead of the one that is

remote, which is our GitHub repository. So the way we actually do it actually

tells you right here is just issue a git push. Meaning we'll take our repository,

take our database error code and we'll push it to the remote database

which is residing at github.com. So we'll go ahead and say git push, and

at this point after we enter your machine might ask you for user and

a password for GitHub. My machine has it already

entered automatically and you could look up on the web how to

do that, so it doesn't ask you again. But we'll say git push and

it should give you something like this. It will tell you something about the fact

that it wrote in to this repository. And you'll see gh-pages

on our local machine has been transferred to

the remote one gh-pages. Okay, so now if we switch to our

browser and we'll refresh this page, you'll see that this site

folder showed up here. If we click inside of it,

you'll see the index.html. Well, let's see if it actually works. We'll go to settings again because I

don't remember this URL anymore, and we'll click on this URL

open up in a separate tab. And what we see here is that

page made by GitHub for you, but we remember that right underneath that

folder is that folder called site. And if we say site/ here's our page,

Hello Coursera! That's the page that we made. And the reason I don't actually have

to specify index.html page itself is because most servers when you say

/ they will default to index.html. So I don't have to explicitly say it, and clearly GitHub server is

one of those servers. And the truth is most service

out there actually do this. And if we creat the name of subfolders

here, we could certainly specify them. And if you wanted to, for example,

do assignments for different modules and you could say module one assignment

slash something and so on. And this is Is the way for you to be able to submit your

assignments to us to be graded. And then you could continue

making your changes. So we'll go ahead and open up Sublime. We could make another change. Hello Coursera, it's a great course. We'll save it again. We'll go back to our command line,

we'll say git status. And you'll see that now

we have a modified page. Well, again, we haven't really

marked it to be committed yet. If we're done editing it we

could go ahead and say git add. And we could say it's explicitly if

we want to or we could just say dot, it doesn't matter. And we added it,

we'll say git status again. And you'll see now it is again mark

to be committed to the repository, to our local repository. Well, I will say again git commit -m for

message, and we'll say Small addition

to our page content. And we'll say enter and now we've

committed a change that we just made, committed that change into

our local git repository. But our remote repository doesn't

know anything about this. In fact, if we switch the browser and

we'll refresh, still the same. Well, the reason it's still the same is

because nothing really is been uploaded to GitHub yet. And if we do git status, it will tell us

again that our local branc, the gh-pages is ahead of the remote one by 1 commit,

because we committed something already. And we can have more than one commit here. We can have more than one commit and

then at some point we'll go ahead and upload it or

publish our changes to our repository. So we'll say git push

to publish our changes. And we're done. So if we switch to the browser and

refresh, it might take a little bit, but eventually this will happen. So something probably got crash,

here we go. It just takes a little bit for it to propagate from its

repository to its website. So it took a few seconds, but it's here. Okay, so there's clearly many,

many other commands that you could use. For example you could use git pull of

somebody else changed your repository. Let's say you're collaborating

with somebody and somebody updated something in

the remote repository in the GitHub. You want to get those changes

to your local machine, you'll say git space pull, and so on. And again, I encourage you to go through the first

couple of two three chapters of the book. And it's definitely very helpful for you to understand what's

really going on with Git. However, the commands I gave you should

be good enough to be able to publish your code to GitHub. And at the same time

publish it to GitHub pages, so they could be visible

to the whole world. And to the rest of the peers

to be able to be graded. So the next thing I would like to

show you is how to use browsersync. Let's go ahead and clear our page and we are still here inside the GitHub

repository or our local folder. We'll go ahead and go inside site and

here's our index.html. So browsersync the sink as a command you

could call from the command line and here I am just pulled

it out from my history. And basically you say browser-sync and

you say start, and you want to start the browser sync. And you want to start it in server mode,

so it'll become like a local HTTP server for

you. And you could say --directory, all that will do is it will actually give

you a directory listing of the folder you're sitting in right now, which is

this case is the folder called site. And then you say --files meaning which

files should browsersync watch for. And if any of those files change,

the browsersync will go ahead and reload those things in the browser. So you don't have to keep clicking

reload over and over and over again. So in this case, we said star, I want to

reload everything no matter what changes, I want those changes reloaded right away. So I'll go ahead and start it. And once I start that, it will go ahead

and open up a browser window for me. And in this case you see

it's localhost port 3000. And since with the --directory, it's

showing me the directory listing of that folder, which is right

now is the site folder. So I can click on index right here and I

can see that it connected to browsersync, and here's this it's a great course. So now what would be really neat is if

I could now pull this aside that's for a second and open up my Sublime Text. And just show you what happens

when you actually edit this. So now I can go ahead and

say, you know what? Let me remove this and I'll remove it. In the second I save it take a look

at what happens when I save the file. Let's save the file,

all of a sudden this gets updated as well. So this is extremely convenient. So that means whenever what is going

on here the cascading style sheets, new content, whatever, it always becomes immediately

available right there on the page. So you don't have to keep coming

over here and clicking refresh. That is a huge time saver when you're

developing and trying to see immediately what's going on on the page without

having to go and clicking refresh. So that's browsersync. It actually has a million

other options and I encourage you to go on their website,

Google for browsersync. And you'll see all

the options that it has. It's a great development helper tool. So hopefully, you will put it to good use. Okay, in this point your environment is

all set up and you're ready to move on.