To get started for programming, we need to set up our environment. Different people have different computers. Some people use Mac, some people use Windows, and then they might have different versions of Compile compilers and there is some tools that are only available on one operating system or another. To remedy all that, what we are going to do is we are going to install Docker, which creates a virtual environment, a virtual computer on our machine. And using that, we are going to be able to have a standard environment for all of our programming.

So to get started, the instructions are on the CSS wiki. So what we need to do is first install and configure the Docker app. I've already installed the Docker app, so if I start it, hopefully we'll have it started, try again opening Docker and we can go to the Dashboard.

And in the current Docker Dashboard, we'll see that we don't have any containers, we don't have any images. So, so images are the basis of how we are going to create our virtual machine. And based on this image, we create a specific container. So the first thing to do is we need to be able to download the image and we are going to download the CSS Linux Lab image using this command Docker pull CSS images slash Linux flag. To do that we need to open a terminal.

And this might be a terminal under Mac OS, it might be a PowerShell or a command line under Windows. But the command we are going to do is Docker pull CSS images Linux Lab. So this is going to go and download the information from the Docker's website. This is about, I think about two or three gigabytes. So this might take a while, depending on how fast your configuration is.

Once we have done created the image, we are going to configure a container that we we can use. So let's look at our images, downloading and being extracted.

So, and once we have downloaded the image, we are going to use a command to create the container and that's going to be created only once. So if we look at Docker right now under Images, we see that we have downloaded CSS Images, Linux Lab and + 3.1 gigabytes. We haven't made any containers, so minimize our Docker. So the next thing we need to do is we need to create a directory structure so that we can access our files both from Docker side and from our PC side. So in the case of Windows users, you're going to use the Windows command, this permanent directory.

In the case of Mac users, you're going to use this permanent directory. We want to create a permanent directory rather than a temporary directory, because temporary directory might get deleted by the system when it reboots or when the system runs low on space. So let's copy and paste this line into our. And so we are going to run Docker and the name is going to be UW NET id. In my case, that's pizon Linux Machine.

And I'm going to leave the name of the machine as Linux Machine. And I on this Linux machine there is going to be a directory called Home CSS UWB Student and I'll name it pizan. So there is going to be a pizan subdirectory on this virtual machine that I create. And I want that directory to be mapped to be the same directory on my Macintosh to be the users, my username, which also happens to be pizan and then a directory called Linux Home. And make sure that this UW NET ID is also replaced with pizan.

And we are using the CSS images Linux Lab Latest and we want to use Bin Bash as the shell. So we press return and if everything goes well, we should just get the shell. So now you'll see that it's telling me that it's a CSS UWB students. So I am on this virtual machine. If I do an ls, there is no.

There are no files there. So I'll create a file called Created from Docker Virtual Machine. So I just created a file and if I do a CD to go to the top level and look at all my files on this virtual machine, I'll see there's a couple of initialization files and a file called pizan which is going to be and the directory that I want. So I exit out of this one and I can close the terminal since I'm done. So if I open up my Finder, I will find that under my home directory there is a directory called Linux Home and there is a file saying Created from Docker vm.

And I can create, for example, a new folder created from Finder on Mac. So one of them was created from the Linux side, the other one was created from the Macintosh side. So now that I have created this environment, I want to make sure that Visual Studio code works with this new environment. So I need to go to Docker. So now I'll see that there is a container that has been created.

It's called pizan Linux Machine and it's currently not running. So I want to make sure that it's running so now that container is running and I'll be able to connect it. So I start my Visual Studio code. This is a fresh version of Visual Studio code. You'll see that I have no extensions currently installed.

So the one extension that I'm going to need is Remote Development. So this is the one that will allow me to connect to Docker. So let me install it.

So now I have installed Remote Development which allows me to use SSH and then Docker ones. So the way I prefer to connect to the Docker is using the bottom left Open a remote window command. When I click on it, it opens up a pull down menu and what I want to do is attach to a running container. Click on that one and there is the pizan Linux machine. So I'm attaching to that container.

The first time I'm attaching to that container, it needs to download a bunch of files and then make sure it's running a server. So Visual Studio code can run on it. And then I have my container, the CSS Images Linux Lab listed here. So right now if, if I the terminal and if you don't get that terminal directly, you can always go terminal new terminal.

If I do an LS or LS al, I'll see all of the files that are on the virtual machine and I can do a change directory to Pizza. So there's my file that I had created from the Linux virtual machine. There is the folder I have created. So Visual Studio Code really likes each project to be in its own folder. So I'm going to create a folder called hello World to be able to work in that folder.

Visual Studio Code wants to open that folder so it can read the configuration of that folder and look at all the files. So I'm going to go to file open folder and when I say open folder, because I'm already using the virtual machine, it's trying to open a folder that is on the virtual machine. So I'll go to pizan hello World and that's where I want to open a folder. And now I want to write my first program. So I want to create a new text file, I guess and save it as main let's save it as hello CPP and I want to just have a simple program that is going to write hello World into the screen.

So I save my hello CPP. So one useful extension, and it really should have prompted me would have been the Microsoft C extension.

The C Microsoft extension and that I do want to be able to install it. So let's make sure that's installed that will have all the intellisense and all the other parts as part of it.

So I have hello cpp. So to run this file, I'm going to run it from the terminal first and then I'll show you how to run it using Visual Studio code. So I'll say new terminal, I'm in the hello World directory. That's the only directory Visual Studio code is looking at. So if I do an ls, I see that this is the file I can use G hello CPP file compiles and that's a out.

I get hello World because I used the CSS Linux Lab image. I have already a couple of things that are installed by default. So we have the G version that's now now 11.5. Currently we have a program called Valgrind.

Does this have a version number? Yeah, that is Valgrind 3.23. We have clang format. I'll press control. Let's see if it will give me a version 18.5.

We have Clang Tidy that's using the LLVM VM 18.1. So all of these other tools have already been installed as part part of it. So this is great. I have hello world and maybe I'll add by world and then maybe I'll have a loop that is going to print out number dot hello as part of it and int main I should use return zero as part of it. So right now you see that when I pressed save it formatted it.

And this is the default formatting that Visual Studio code has. But I would instead like to do is use the clang formats version that's for llvm. So let me go and fetch that line it.

Okay, now that I have fetched it, the command to create it is called clang format and I want the LLVM style and I'm going to create a file called dot clang format as part of it. So I'm executing it at the command line and you'll see now that I have created this configuration file. So this is what was needed for formatting.

So now since Visual Studio code knows about this type of formatting when I save does it in the KR style which of the curly braces starting on the same line as the function or as the for loop. Okay, so our formatting is now fixed. Next thing I want to do is I want to add a breakpoint. Now that I have added a breakpoint, I'm going to go run and say start debugging. And we want to use the C G Build and so now we are at sea out hello world we can skip over.

Let's see if we can get to the terminal so we can see hello world we can see by world. And now we have a variable I. And every time we go around the loop our variable I is increasing and we are able to run our program.

We can also create conditional breakpoints. So I can right click and edit that breakpoint. So I want the expression to be I equals five. So I want the breakpoint to be valid when the expression is I is equal to five.

Okay. Oh, I see. It always shows it to me. I guess. So now if I go run and start debugging, we have a hello world.

I can press this button saying continue and it's going to continue until I thought it would continue until I equals five.

Let's have another look. This breakpoints so let's we have a breakpoint edit breakpoint break when expression evaluates to true. So I equals 5.

I had forgotten to press return as part of it. So now condition is I equals 5. So now let's try running and start debugging. So we have hello world and when I press continue it's going to go all the way until I is equal to 5 and then it's going to break so that now I can go step by step. This conditional breakpoints allows you to run through the program until the point that you need to do the breaking.

So for Visual Studio code, there are some important files. We already talked about the clang format file that is telling us how the formatting is going to be. And we also have the under VS code, the settings file that got created automatically and this tasks file. And the tasks file tells us where to have the G compiler and what the executable, what the input is going to be and what the executable is going to be. So for example, if instead of hello World, in addition to hello world, we created one more file and let's call this file by cpp and with this file have a proper include statement.

Once again I'm going to use namespace std so that I don't have to use std colon colon all the time. So when I want to say by just a cout by end line so I save this file, I go to hello cpp and I add this as a. I add this as a forward declaration. So now if I want to compile from the command line I would use g star cpp. So all of the CPP piles together, compiles and oh, I should add save by to my main. So it actually gets called.

So I compile it, I execute, says hello world, bye. World prints hello 10 times and then says bye. But now if I go to my hello CPP and run, start debugging, we end up with an error and if we abort, we'll see that it says hey, undefined reference to bye. What's happening is the default version of Visual Studio code tasks JSON file thinks that you're compiling just a single file. It thinks this is the only file that you are compiling.

As a result, bye CPP is never seen. So instead of just this file, what we want to say is cpp.

So that way we want to compile all of the files. So now we go to hello cpp. Now we say run, start debugging and we'll be able to run it and say bye is now defined because we changed how the test JSON looks at it. It now uses all of the CPP files. And the default for Visual Studio code is it creates an executable, in this case hello, without any extensions as part of it.

So this is how we download a Docker image, configure it as a container, and have Visual Studio code use it. Now if we go to our Finder and I'll go to Pizan Linux Home, there's a hello World folder. So these are still available. So I can zip them up, I can copy them, I can delete them if necessary, but I can also access them from the Linux command line inside the virtual machine. And creating a Docker, a standard environment, makes sure that the program that runs on your computer is also going to run on my computer and we won't have any configuration problems.