CS 240: Deploy Code and Start the Server Transcript

[00:00:00] Now that you have um created your EC2 instance and modified your, your client code as needed to be, to be ready to be deployed.

[00:00:09] It is time to, to build, we need to build a couple of jar files.

Start visual description. The professor demonstrates how to build jar files for both the server and client. He explains that jar files are similar to zip files and can hold all the files for a project. End visual description.

[00:00:13] We need to build a jar file for the server and a jar file for the client jar files are kind of like zip files.

[00:00:19] They are um files that can hold all of the files for a project.

[00:00:23] So we need to generate those and then we're going to copy them over to our server.

[00:00:28] Once we've done that, we will be able to just restart the server and we'll be able to play chess um on Aws.

[00:00:35] So there are a couple of different ways that you can build the server.

[00:00:37] There's this way that the tutorial tells you and then I'm going to tell you a different way.

[00:00:40] So I'll kind of talk through the way that the tutorial is telling you to do it.

[00:00:44] So the tutorial is telling you to do this with Maven and we probably haven't talked much about Maven in this class.

Start visual description. The professor shows how to use Maven to build the server and client jar files. He explains that Maven is a build and deployment tool

that manages dependencies and specifies how things should be built. End visual description.

[00:00:52] Um Your, your project, the, the starter code that we gave you is already set up to be able to build your, to build your server and your client using something called maven. So, Maven is a, it's like a build and deployment tool that can manage dependencies for you, and it can um it's, it's where you can specify how things should be built.

[00:01:17] So we, we had you the, the starter code that we gave you sets that up as maven.

[00:01:22] So the auto grader can work. The auto grader uses maven to build your code.

[00:01:25] But that's not really what we taught you during class, during class, when you needed to add a dependency.

[00:01:30] For example, we had you do that in a different way.

[00:01:33] We had you go through file project structure, at least, at least that's the way I do it. And probably your instructor did that too where you go through file project structure, and you just specify the dependencies, and you never worry about doing anything directly with maven.

[00:01:47] So you can do that, you can follow these Maven commands.

[00:01:51] And what I found is when I did this, my setup didn't work exactly. So, I had to tweak a few things with Maven and that's why I'm going to show you it might just work for you exactly the way this tutorial shows.

[00:02:02] But I'm going to show you another way to build those jar files.

[00:02:06] So another way to do it is to bring up our project.

[00:02:14] And so you can go to file, project structure and click on artifacts. It was already clicked for me because that's what I, what I did last time I opened this but make sure artifacts is clicked, then click the plus sign.

Start visual description. The professor demonstrates how to build a jar file from modules with dependencies using the project structure in the IDE. He clicks on "artifacts," then the plus sign, and selects "jar from modules with dependencies." End visual description.

[00:02:29] And what we are going to do is build a jar file from modules with dependencies. So, what that means is I'm going to build a jar file.

[00:02:38] The jar file is going to be generated from my, from a module.

[00:02:41] So remember you have three modules in your project, you have your server module, your client module and your shared module.

[00:02:46] Um Both the server and the client module depend on the shared module.

[00:02:50] So they both have to have that code as well.

[00:02:53] So what we're going to do is we're going to build a jar file from mo with dependencies.

[00:02:58] The with dependencies.

[00:02:59] Part means that that jar file is not just going to contain the code from whatever module we generated it from, but anything that that module depends on.

[00:03:08] So any, any jar file dependencies that is downloaded and any other modules.

[00:03:13] And in our case, that's the shared module, all of that code will be, will be integrated into the jar file.

[00:03:20] So we'll specify that. Um Actually let me cancel here and show you so I can talk you through that. So, we go to Jar from um from modules with dependencies. Click that and now I need to specify the module. So, let's build the server one first.

[00:03:35] So I'll specify the server module.

Start visual description. The professor specifies the server module and the main class that contains the main method for the server. He shows how to set the output directory for the jar file and includes it in the project build. End visual description.

[00:03:38] Now. It wants me to specify what is the main class? What's the class that has the main method? This is where your code might be different from mine.

[00:03:45] But if I click on that, it's going to show me all of the possible classes, all the classes that actually have a main method in my entire project.

[00:03:52] We're building the server jar right now. So, I need to specify that um it, it might be a, a main class for you and you just, you might have two main classes, one for your client and one for your server.

[00:04:02] So make sure you, you specify the right one for each jar. So, we're building the jar.

[00:04:07] So I've specified the ma the class that has a main method for.

[00:04:11] I'm sorry, we're building the server.

[00:04:12] I've specified the class that contains my main method in my project for the server and I'll hit OK.

[00:04:19] And now the next thing to notice all the defaults should be OK here, but we want to take note of where it's going to generate that jar from.

[00:04:29] So right here, um it's going to put it in my project root directory slash server slash resources.

[00:04:36] That's where it's going to put that.

[00:04:40] I'm just trying to decide if that's where I actually want it.

[00:04:44] I guess that's OK. We'll keep that as a default we'll hit. OK? Um Notice I, there was a check mark for including tests.

[00:04:50] I did not check that I don't want tests to be in my jar.

[00:04:54] Um And the other thing I'm going to do is I'm going to click this include in project build.

[00:04:58] What that will do is that'll make it.

[00:04:59] So every time I build my server, it will regenerate the jar.

[00:05:02] So that would be convenient to do. So, we'll click on that.

[00:05:05] I'll hit, apply and hit. OK? And so now it's ready to build the server jar for me.

[00:05:12] Now, we're going to go do the exact same thing for the client jar.

[00:05:17] So notice here it has server dot jar there now, so click on, make sure artifacts is clicked, click plus jar with dependencies.

[00:05:26] And now our module is going to be our client.

[00:05:29] And now the main class needs to be the one that has your main class for your client U I.

[00:05:34] So we'll click, OK? We'll let that.

[00:05:38] Let's see.

[00:05:42] Yeah, we'll hit OK for that.

[00:05:44] And now I'm going to, I mean, I think I, I said something wrong with the previous I with the previous example.

[00:05:51] Um I mentioned that the previous window was showing where the output directory is going to be. That's actually not true.

[00:05:57] It was showing where it could find one of the files that it needs to put in the, in the jar and the default should have been OK for that.

[00:06:04] This is the output. This is where the jar files are going to go, it's in the root of my project, then an out-subdirectory artifacts and then claimed underscore jar.

[00:06:17] So we'll hit, apply, hit. OK.

Start visual description. The professor shows the output directory where the jar files are generated. He navigates to the project root directory, then to the "out" subdirectory, and finally to "artifacts" to find the client and server jar files. End visual description.

[00:06:20] And just because I was a little unclear about that, I'll show you where the server jar is going.

[00:06:25] It's going in the same place um project out artifacts. But then a server underscore jar.

[00:06:32] So we hit OK.

[00:06:33] And now if I just do um let's see, build, rebuild, I'll just do rebuild project.

[00:06:45] So that's rebuilding the project.

[00:06:47] But it is also going to generate those jar files for me now.

[00:06:51] And depending on what version of intel J you have and how you have it configured, you may or may not be able to see that out directory from here.

[00:06:58] Um I it looks like I can't see it from here.

[00:07:01] So I will show you where it is in the uh just in the file system. So, if I go to projects, I've got a lot of projects here under CS 240.

[00:07:23] Here is the code for all those examples that you were shown in class.

[00:07:28] And I should be able to find my chess project. Actually, it's not there. It's under projects chess V two.

[00:07:35] OK. So, remember I said it's in the chess V two out artifacts.

[00:07:39] There's client dot Jar. So that's the client, that's the jar file for the client, that's the jar file for the server.

[00:07:47] OK? So, we're going to need those.

[00:07:49] So now that we have that built, we will come back to our instructions here.

[00:07:54] And the next two things we need to do, we need to copy the server code up to our EC2 client. So, we're going to use this command to copy it. So, I'll copy that and I'm going to need to modify it. So, I'll copy it back into my text editor again and I want to point something out that could be easily miss, you could easily miss on this.

[00:08:16] So we're going to use the SCP command, which is similar to SSH, but it's secure copy.

[00:08:22] So this command needs to be executed, not from the machine that we're on, but from my local machine.

[00:08:29] So if you notice this terminal, this terminal is still connected to the EC2 instance. That's not what I need to do this command. I need to do it from a terminal that's, that's just on my laptop. So, I'm going to create a new window and the way you do this would be different, obviously in windows.

[00:08:50] OK.

[00:08:51] So now um now I have a terminal that's just on my laptop, make that a little bit bigger.

[00:09:05] There we go. Hopefully you can, that might be too big. Hopefully you can see that. OK? OK.

[00:09:14] Now we need this, but we're going to have to change this, this command a little bit.

[00:09:18] So remember we need our path to our pen file our security key file.

[00:09:23] That is this right here.

[00:09:30] And let's see, we are copying our server jar file.

[00:09:37] So we're going to have to change that because mine is in a different location and we're copying it to our EC2 instance at some IP address. So, I need to get that IP address.

[00:09:49] I haven't restarted my server since I created the first video here.

[00:09:53] So the IP address is the same as it was from the first video.

[00:09:56] But if you have restarted your server, you'll have to go in and find the IP address again.

[00:10:01] OK. So that's why we're copying the file too.

[00:10:03] But I also need the path of where I can get the file and mine even has a different name than that.

[00:10:09] So I'm going to open up another terminal.

[00:10:12] Actually, I can use this terminal. So, I will have to go find that file and I'll use the same little trick I did before where I'm just going to drag the file into here and that'll give me the path and I didn't copy. So right, click and say copy.

[00:10:45] There we go. That should be the command that will copy that up to my AC two instance.

[00:10:50] So um hopefully that'll work. Should work.

[00:11:01] OK? So that has now copied that file onto the EC2 instance.

[00:11:05] Next instructions should tell us um to start up the server.

[00:11:11] OK. So, we've copied that. Oh, actually there's one thing that I didn't do though.

[00:11:14] So that actually isn't going to work.

[00:11:17] I was looking for the instruction.

[00:11:18] Let me see where the instruction is that we missed because I know we missed one.

[00:11:22] OK. Right here, we maybe should bold this.

[00:11:24] This says build the client jar and copy it into your servers' resources, web directory.

[00:11:30] Um That's easy to miss. I just missed it.

[00:11:33] So we're going to have to redo some of this.

[00:11:36] So we've got this client jar that needs to be copied into my server's web directory.

[00:11:45] So that needs to be right here.

[00:11:46] Remember we created the link that you can click from the index dot html page to download the client code.

[00:11:52] Well, there's no client code to download.

[00:11:54] So I need to copy that client code and that was here under out client dot jar. So, we need to copy that I do copy, and we'll paste it right here.

[00:12:13] OK? So now we have the client dot Jar.

[00:12:16] Now I need to rebuild my server dot Jar.

[00:12:20] And after copying that in, I, I need to build the jar file with dependency.

[00:12:24] So that's easy to do because the way I set this up, I'll just say build rebuild project and we'll notice the time is 1059 just to make sure that Jar was rebuilt.

[00:12:35] So I come back here and find server and notice it says that server was built at 1059.

[00:12:43] So that's a new one that was built after I copied the code in.

[00:12:46] So now I need to re execute that SCP command to copy the new jar file up.

[00:12:54] OK? Now we have the new jar and so it should be all I have to do now is start my server, start my chest server on that EC2 client.

[00:13:06] So we need the SSH back into the EC2 client.

[00:13:10] Um I'll just do that from here.

[00:13:14] So I'll go grab that original SSH command that I used in a previous video.

[00:13:21] OK? So now I'm on the client and so now the EC2 instances are running because I just logged into it.

[00:13:29] But we need to start the um we need to start the chess server on that plan.

[00:13:38] So we'll just say Java minus Jar.

Start visual description. The professor starts the chess server on the EC2 instance by executing the command "java -jar server.jar" in the terminal. He specifies the port as 8080 and confirms that the server is listening on port 8080. End visual description.

[00:13:40] So we copied this a jar file up and we called it server dot Jar.

[00:13:43] It's just in the root directory of the EC2 instance.

[00:13:46] So this command should start my server. Now, um Let's see. Usage server port. I didn't specify the port is 8080.

[00:13:57] OK. So, it says it's listening on port 80.

[00:13:59] So now that is the EC2 instance listening on its port 80 which means I should be able to log into it now and I'm going to need this IP address to log into it.

[00:14:10] So if I bring up a terminal, I should be able to do that. And then my servers running on for 8080.

[00:14:17] When I hit that, I should see the chest, the test page and there it is.

[00:14:20] So this is the original test page that you are used to seeing, but notice that it has this in it, download the client and then it's telling you how to run it.

[00:14:30] Now, if we look closely at that um at that html code that we pasted in here, it is dynamically determining what IP address this um page is being served up from.

[00:14:43] So that is the actual IP address that this is running on. So, what that means then is I can download the client and then I can use this command from the directory that contains the client and that will start the server.

[00:14:56] So let's do that.

[00:14:57] We will do that.

[00:15:00] And let's see, I think I will just do that from to the desktop again.

[00:15:04] So we'll download it to the desktop.

[00:15:07] And this is saying depending on what version of different browsers you're using it, may or may not just download it for you.

[00:15:14] This doesn't want to download it for me.

[00:15:15] It's saying it's an insecure download because that's executable code.

[00:15:19] So we're blocking it.

[00:15:20] I'm going to say, go ahead and keep it, go ahead and download it.

[00:15:23] So now that has should have downloaded it, let's make sure it did.

[00:15:30] OK? So, there's my client dot Jar.

[00:15:32] Um That's the one that I got from the server.

[00:15:35] So if you think about that, you, you already have it, I could have just executed the client dot Jar from my machine.

[00:15:41] But if you're playing chess with one of your friends or a parent or something like that, they don't have your Jar file.

[00:15:46] So that's the way they can get it.

[00:15:48] Um In order for this to work, they also have to have Java installed on their machine.

[00:15:52] Most people do.

[00:15:53] Um But if they don't, they would have to install it, which is one more reason why this is not actually how we would do it and we would really build a web client.

[00:16:01] So people don't have to install Java and download something in order to play our game.

[00:16:05] And, but that's the way we did it.

[00:16:06] So now I need to open up a new terminal and I'll go to my desktop directory and now I should be able to run the client with this command.

[00:16:29] There we go.

[00:16:30] And just to make sure you can see that that is my chess client, but it is accessing the server on E two.

[00:16:39] OK? So, we could stop there.

[00:16:41] But there's one thing that I don't like about this that we should fix, and we'll fix it in another video.

[00:16:46] Um Notice that I logged into my EC2 instance and then I manually started the server from this terminal that's logged into my EC2 instance.

[00:16:55] That's not great.

[00:16:57] Um That means that in order to, in order for somebody to play chess, you have to log in to your EC2 instance and manually start the chess game.

[00:17:06] And then if you kill it, like if I do this, if I close this down, now the chest server isn't running.

[00:17:15] So the AC two instance is still running, but the chest server isn't running.

[00:17:18] So now if I try to do anything, I'm going to get an error.

[00:17:21] So if I do that and we'll just enter a username, enter a password, unable to log in because of exception, connection exception.

[00:17:30] So there is no chess server running on the EC2 instance anymore because I shut down my terminal, what we want to do is make it.

[00:17:37] So every time I start my EC2 instance, it automatically starts the chess server.

[00:17:42] So the chest server is just always available if my EC2 instance is running, and I'll show you how to do that in the next video.