Audio file

Your Recording 26.wav

Transcript

00:00:02 Speaker 2

In this lesson, we're going to take a look at Jupyter Notebooks in Action. So what are we going to look at? We're going to see how to install a Jupyter Lab server. We'll be doing this on an Ubuntu Linux instance. Now, at this point, we are doing this on an EC2 instance just to see the process of how Jupyter Lab works. At the moment, we are not yet looking at Sagemaker itself.

00:00:21 Speaker 2

Then we're going to see how we can consume the Jupiter interface via a web browser. We'll create a brand new Jupiter notebook and we'll add code cells to that Jupiter notebook and run some Python code and check we get the right standard output.

00:00:33 Speaker 2

Then we'll add some markdown cells to our Jupyter notebook and annotate what we're doing so it will be easy for another scientist to follow our work.

00:00:40 Speaker 2

Let's jump in.

00:00:42 Speaker 2

So here we are in the AWS Management console. Now for this demonstration, I'm going to be using EC2. That's the AWS service that we use to create and manage virtual machines. Now I'm only using EC2 here for convenience so I can show you that I'm setting up something completely new to run Jupyter Lab.

00:00:58 Speaker 2

At the moment, we're not yet looking at the Sagemaker product. I'm showing you the way that I would set up Jupiter manually first, and then I will show you later on how we can get Jupiter Lab within Sagemaker.

00:01:08 Speaker 2

So here I'm in the EC2 console of the AWS Management console and it looks like I've got one virtual machine called Code Cloud Jupyter.

00:01:15 Speaker 2

Now I want to get a command line with that virtual machine, so one of the easiest and secure ways of connecting to it is using the connect button here to launch what we call instance connect. Now why I really like instance connect as a method of gaining a command line with Linux is that I can get a secure shell session with it, but without having to open up the secure shell firewall, allowing everyone potentially to attack the secure shell daemon. It's a much more secure way of going in, and I can determine who's allowed to use it by using regular I am permissions.

00:01:43 Speaker 2

Where's that connect?

00:01:47 Speaker 2

OK, that's us in our Ubuntu machine. Now I can see I'm using Ubuntu version 2404.2 lockdown support and let's see what tools we've got available to us. Well, first up, I want to check that we have Python installed. So I'm going to go ahead and type Python. But something interesting will happen here. Not found because when we take Python, we're kind of expecting that Python would be Python 2. That was the default before in Ubuntu. So to avoid confusion and so that you don't get different Python versions when you're not expecting them, Python, because it's not included, doesn't actually render.

00:02:17 Speaker 2

3 So if I want Python three, I would have to type Python 3 Now I can get Python.

00:02:22 Speaker 2

Now here I can see I've got Python 312 three and that's absolutely fine. And let's just check our shell is working. Let's say a = 3, B equals 4 and let's print the output of A+B.

00:02:32 Speaker 2

Great, it's working fine. So Python is there and I'm just going to come out of Python shell. I don't need that right now, and we're looking good at this point. OK, now I want to install.

00:02:42 Speaker 2

The Jupyter lab server and the way I would install that normally is with the pip tool, the Python installer package manager.

00:02:49 Speaker 2

But unfortunately pip does not come pre-installed on Ubuntu 2404 LTS. So we're just going to have to install pip 1st and then we can install Jupiter.

00:02:57 Speaker 2

I'm gonna clear the screen for us first.

00:02:59 Speaker 2

OK, now we're a clear screen, so let's now install the Python Package Manager.

00:03:10 Speaker 2

I'll just say yes.

00:03:11 Speaker 2

Kind of recalls installing pip.

00:03:16 Speaker 2

OK, great. So that's pip now installed. Now again, we might have the same issue before here if we just type in pip.

00:03:22 Speaker 2

OK, Yep, so we've got pip that has worked. Sometimes you have to do pip 3, but that worked well. And if I just do pip dash dash version, I can check what version. OK, got version 24.0 for Python 312. Perfect. We're now almost in position where we can install Jupyter. I'm going to clear the screen again.

00:03:38 Speaker 2

Now, just before I use pip to install Jupiter, often when we're working on Python projects on the same machine, we might end up using different versions of Python packages. And therefore when we install a Python package, if I install it globally, then it could affect another Python project on the same machine. So it is good practice to make use of Python virtual environments. And virtual environments allow me to have whatever versions of Python packages I want, but they're only with scope to the current project I'm in. And you can switch between scopes, but it really does keep your system clean and avoids.

00:04:08 Speaker 2

Package conflict. So I'm going to do that just before I install Jupyter just to keep things nice and clean. So I'm going to do Python 3 -, M and a virtual environment, and we're just going to call it Jupyter inv like that.

00:04:20 Speaker 2

We've now got a Jupiter environment, but we can't quite get started yet until we source it. So we have to source Jupiter env.

00:04:28 Speaker 2

Average 2 bin activate.

00:04:32 Speaker 2

There we go. Now we know that we are now in the context of our virtual environment because we can see that our prompt has changed. And now we've got this prefix Jupiter under score N That's just a label. I could have called it anything, but that name made sense to me.

00:04:44 Speaker 2

So this means that now as I perform pip commands to install things, then they will be in the context of this virtual environment.

00:04:50 Speaker 2

So let's just do install jupyter. OK, so we're doing install jupyter.

00:04:57 Speaker 2

Anyway, it goes to Python repositories and pulls down Jupyter.

00:05:06 Speaker 2

Wonderful, that looks like Jupiter is now installed, so let me go ahead and clear the screen.

00:05:13 Speaker 2

Now that Jupiter is installed, let's start the process up. We need to start it as a background process, then we can connect to it using our browser.

00:05:21 Speaker 2

Now to start Jupiter, we just say Jupiter Jupiter notebook. Now that would normally be enough if I was running this like on my local machine, but because I'm running it in a remote EC2 instance, I'm just going to have to change the network configuration slightly to tell it to bind the Jupiter process to all network interfaces, including the interface that will allow remote connections.

00:05:42 Speaker 2

I also need to specify that there's no local browser. I'll be connecting to this from a completely different other machine.

00:05:51 Speaker 2

OK, so now it's telling me.

00:05:54 Speaker 2

That the jupyter server has started up. Now it tells me that use control C to stop the server and shut it down. OK, I don't want to shut it down, I want to keep it running.

00:06:02 Speaker 2

I can see here that is giving me a link that includes some kind of key that will get me started, so I'm going to copy that link.

00:06:08 Speaker 2

OK, but this link is going to work and I'll tell you why. Let's paste in the link and see that it doesn't work. It doesn't work because it's using the I address of the EC2 instance, but the I address is a non rootable address. What we need to use is the public IP address of the instance. So if I look at my instance under the details panel, I can grab the public IP address of my instance. There we go.

00:06:30 Speaker 2

And it's just changed that up here in the URL.

00:06:35 Speaker 2

Change that to the public IP.

00:06:37 Speaker 2

I know things look much better.

00:06:41 Speaker 2

Wonderful. All right, so we've now got a Jupiter interface. Now this is the initial Jupiter interface. This is not Jupiter Lab, not yet. We'll see that in a moment. So we can see, OK, we've got a Jupiter environment. That's our virtual environment we created and we've got no kernels running right now.

00:06:55 Speaker 2

So let's go to new and we can see what I want. A Python three OK kernel. Does that mean? Let's have a look.

00:07:02 Speaker 2

And I'm a URL change straight away. Notice now it's called untitled dot ipy nd this is my Jupyter notebook. Ah OK now the first cell that it's placed up here is a code cell. Think OK so I could put in there my Python code like a = 6, B equals 4 and let's print the result of summing A+B.

00:07:21 Speaker 2

There we go.

00:07:22 Speaker 2

Now the code is there, but it hasn't yet run. So to run the code I would use the shift enter key.

00:07:28 Speaker 2

And notice what happened there very briefly in the square brackets to the to the left side of the cell. It went to an asterisk in the box and then it's now showing number one. OK, so if I do more code cells, I'll say C = 8 and print C + A and C + a. There we go and let's run that one. There we go again. You briefly saw the asterisk and then you saw number 2. So we can see the order in which cells have been run.

00:07:52 Speaker 2

I can see the output has been produced in line of the cells. So the output of the first cell is shown and then the second cell is shown. Now this is wonderful because it allows us to really relate which output was produced by which cell. So you can see that we could start structuring our Jupyter notebook. So our code describing our training code or our feature engineering code and it can break it up into different component parts, running each one and then explaining each one in turn. I come up to the save icon. I could save my work currently saved as untitled or I could save it as something else.

00:08:20 Speaker 2

Now let's say I want to annotate my work. OK, I'm going to add another cell.

00:08:24 Speaker 2

So I could create a duplicate of the cell or I could add a cell above, add a cell below, same up here. So let me just add one more cell. Now when I click into a new cell, I can come to this drop down list here and change from code into markdown. Now if I go into markdown, I could start using markdown text. So for example, I might want to say this is a heading so feature engineering code.

00:08:46 Speaker 2

Code next. There we go. Now it doesn't look very exciting when I put it like that, but because I put the pound sign or hash symbol at front, that will get interpreted as a heading. So think of that like heading one in like a word processor. So in the same way as I did shift enter to run a code cell for a markdown cell, shift enter is like render, render this markdown and show in display form like that. So it's really DNN. Of course, I can add more into a markdown cell. I can use bullet points, I can insert inline graphics, all the things that markdown text normally can.

00:09:14 Speaker 2

So if you're used to maybe using markdown files for the likes of Readme's in a git repository, you will feel comfortable with this as well.

00:09:21 Speaker 2

Now there are shortcut keys that we can use as well in order to insert cells or remove cells. But when you're just starting out, all you really need is these buttons to add or remove cells. Or if I want to get rid of here, I've got blank cells, I'm going to get rid of it. I just hit the little dustbin icon, gets rid of it. But remember when I save this Jupyter notebook and saving not just the code, but the outputs that were produced and the inline markdown text as well.

00:09:43 Speaker 2

So with that I'm just going to close this interface and close this interface for now as we don't need it.

00:09:48 Speaker 2

And coming back here, I'm just going to kill off my Jupiter server.

00:09:52 Speaker 2

So I'm just using Control C to do that.

00:09:54 Speaker 2

And shut down my Jupiter server.

00:09:57 Speaker 2

And I'll clear my screen.

00:09:59 Speaker 2

Now that we've looked at Jupiter, let's see the difference with Jupiter Lab. Again, we will use the pip command.

00:10:07 Speaker 2

But this time we're going to say Jupiter Lab.

00:10:13 Speaker 2

OK, that looks like the lab is installed. OK, now let's clear our screen.

00:10:18 Speaker 2

OK, so Jupiter Lab is now installed and we're going to want to start it up just like we did before. So we'll say Jupiter.

00:10:25 Speaker 2

Lab and we'll just put in that network modifier just because we are connecting to it remotely. If I was running this locally, this would be enough, but because we're running it remotely, we just bind it to all network interfaces and tell it that there is no local browser. This will be a remote connection.

00:10:40 Speaker 2

Now that looks very similar to what we saw when we launched Jupiter, looks almost identical and has generated me a URL that I can use to connect to the interface. So let's go grab that first of all, copy it and pop it into our browser. But we know that this won't work because the URL includes this name which is actually private DNS name specific to the private network in EC2. So I need to go grab the instances public IP address and use that instead of my URL. Let's just quickly do that.

00:11:08 Speaker 2

There we go, and now we'll get Jupyter Lab.

00:11:11 Speaker 2

Yay, wonderful. There's Jupiter Lab. Now straight away we can see that the interface is more mature and has more options available to us compared to what we saw in the Jupiter interface. I can see I've got a tab, that's the launcher tab, and from here I could launch a new notebook or I could launch a console. I can see in the left hand panel, I've got my file system. Now I can upload files here, I've got an upload button. I can select the existing file here and I can choose download, copy and paste standard file commands.

00:11:39 Speaker 2

Now what it also can do here?

00:11:41 Speaker 2

Is let's just open up the notebook that we created when we were in Jupiter.

00:11:46 Speaker 2

So I double click on it. Ah, there's our Jupyter notebook with our code cells and our markdown cells. I wanted to create a new notebook. I just click on plus and I can pick Python 3 kernel and there I've got another notebook. So now I can work on 2 notebooks and flip between them in this multi tab interface.

00:12:00 Speaker 2

If I also wanted a console, I could click on Python 3 console and that takes me into an interactive console that I can use down here. And if I go back to launcher, I could also pick terminal. If I pick terminal, wonderful, I can see my packages list and if I wanted to install more packages, I could do it directly here from within the Jupyter Lab interface.

00:12:20 Speaker 2

So Jupyterlab provides me with a more mature integrated network environment. It gives me the ability to create multiple notebooks split between them, having a directive console, have a Linux command line shell, and it's extensible. If I click on the extension manager, I can see, OK, well, what extensions do I have available to me? And I've installed already into this environment a Git extension. So that means that I could have a direct link into Git repository, so I could say cloner repo, and I could give a path into my GitLab or GitHub repository and pull that directly down in here into the.

00:12:50 Speaker 2

System all the Jupyter lab server and we'll see. This is similar to what we see when we start to use Sagemaker as well.

00:12:56 Speaker 2

But we thought it'd be important for you to see how Jupiter Lab really is the foundation for any data scientist working with Python And working with annotating their work with documentation and the outputs of running the code cells. What we've done here is just install the Jupyter open source server locally into an EC2 instance. But when we start to use Sagemaker, we'll see that will be a ready made environment and we won't need to do any installs for that. So what did we do? We checked if Python was installed in the instance and then we used pip. Well, we actually had to install pip first.

00:13:26 Speaker 2

And then we used pip to install Jupyter. We launched A Jupyter notebook and then we created a new notebook, added a markdown cell. I did a code execution cell, reviewed the output and discussed the shortcuts when we saw that we could do shift enter in particular to execute a cell or to render the markdown that we have. So what have we seen in this lesson? We've seen the steps that you would need to follow to set up a Jupyter notebook on a Linux machine and connect to it. We've seen that Jupyter notebooks are the foundation for the majority of data scientists working in machine learning projects. We've seen it locally and we.

00:13:56 Speaker 2

Share it shortly with how it looks when we use it in sage makeup.

00:14:00 Speaker 2

The Sagemaker platform makes extensive use of Jupiter as a key part of you working through the machine learning pipeline. So learning Sagemaker means learning Jupyter as well.

00:14:11 Speaker 2

We've seen that within the Jupyter notebooks we have code cells and we have markdown cells. It is up to you to add in these cells yourself and populate them with code and populate them with your text annotations with the right syntactical elements for rendering headers or bullet points and so forth. Remember that we are using Jupyter notebooks because they provide the best environment for a collaborative data scientist. They give us the ability to run Python code. They give us the ability to annotate our work to save the output of ourselves in line with the document. So we're getting the best of a Python IDE and a re-evaluate.

00:14:41 Speaker 2

Loop Shell Our next lesson is a lab exercise. In Lab 1, you will get your hands on some Jupiter Lab instances.