

Audio file

[Your Recording 28.wav](#)

Transcript

00:00:07 Speaker 2

In this lesson, we're going to look at navigating the user interface of Sagemaker AI. We're first going to look at how the Sagemaker user interface in many ways does not help you learn how to use the product itself. In other words, it lacks intuitiveness. We can't really figure out what the product is about just from the user interface. We're going to look at components that are exposed in the user interface and help demystify them step by step. Then we'll understand a little more about the workflow that we would go through in a machine learning project.

00:00:36 Speaker 2

And finally, we will review the results, specifically how we could get a hosted Jupyterlab interface directly from the Management Web Console. So let's look how we get into the Sagemaker user interface. We start in the AWS Management Console. Now the management console provides us with a web interface to all of the AWS services, but specifically we need to search for the Sagemaker services. So we're just using the search bar here at the top of the homepage of the AWS Management Console.

00:01:00 Speaker 2

Now when we find the Sagemaker services, you'll find that there's actually 2 services listed, Amazon Sagemaker AI and Amazon Sagemaker Platform. Now, the Sagemaker Platform is relatively new and is still really evolving as a product, but the intention with the Sagemaker Platform is to take everything that you can do in Sagemaker AI and do more specifically around integration with data lakes and data warehousing. For us, our focus is going to be on the core functionality of Sagemaker AI where we train and build models and optionally host them.

00:01:30 Speaker 2

Now, Sagemaker UI is a complex and overwhelming interface. I remember when I started first using Sagemaker AI and it made very little sense to me. There was lots of options, but there was no clear workflow in terms of how I was meant to use each one of those. And this tends to be overwhelming for new users. Now, if you've tried to learn an AWS service just from clicking around the management console, you've probably been reasonably successful. If I go to EC2, I can see there's a large button that says Create new instance. So it kind of leads me into performing actions.

00:02:00 Speaker 2

That service, even if I've never used that service before. In other words, many of the other management consoles are intuitive to use.

00:02:07 Speaker 2

But given that on a machine learning project most of your time is going to be spent in Jupyter Notebook or in a development environment writing Python scripts, you're not really using the AWS Management Console to move your mill project forward. So as such, don't be too concerned When you look in the AWS Sagemaker AI Web Console interface and it doesn't make much sense to you, you will not be spending very much time there, if any. Your time will be spent in a Jupyter notebook or in your preferred development environment.

00:02:34 Speaker 2

Let's take a look into the Sagemaker user interface. Now in the Sagemaker AI interface, I have a left hand side navigation bar and I can see some of the names in here might make some sense to me, such as processing or training or inference amongst other names that are listed. Let's start with processing. There's only one option under there, processing jobs. And if I look there, I can see if there's any processing jobs. Is there any background processing going on? And the answer is no, there's nothing there. I haven't started using the product, so it's feeling kind of empty so far.

00:03:02 Speaker 2

Let's click on under training and look at training jobs. And again, this feels very empty. We haven't initiated any training of a model yet, therefore there are no training jobs listed either as in progress or completed.

00:03:12 Speaker 2

Let's look under Models under the Inference header. And again, we haven't created anything yet, so nothing is listed here. I'm feeling very lonely in this user interface. There is nothing here that can help me, but once I start to create models, then I'll start to see this user interface populated with values. Lastly, here we're clicking on Endpoints. Now endpoints is where we decide to host a model for inference. In other words, we've deployed our model onto a target compute platform. Now again, we're not seeing any endpoints listed because we haven't created any yet.

00:03:41 Speaker 2

And again, it doesn't really feel in this interface like it's helping us to achieve a specific ML task. Very much feels like it's reporting upon the status, but isn't really helping you achieve that outcome.

00:03:51 Speaker 2

So if you're trying to learn Sagemaker just from the user interface, you're probably getting very frustrated and in fact you are making it 10 times harder on yourself. Now. I can say this because I originally started this way too.

00:04:03 Speaker 2

Now, this is because Sagemaker has a platform is primarily going to be driven by your programmatic activity. In other words, the Python code that you run from the cells of your Jupyter notebooks will instruct Sagemaker to create processing jobs or to create training jobs, to store models or to deploy

models. So it is really a code first platform and unless you're developing code that drives that behavior, you're not going to see much in the user interface.

00:04:29 Speaker 2

So where are we going to drive this programmatic behavior from? We're going to drive it from our Python code. We're going to leverage the Sagemaker SDK that we introduced in the last lesson. That's going to give us our high level abstractions of activities like training and deploying models. And we're going to start by using Jupyter notebooks. Jupyter notebooks provide us with code cells and markdown cells and the ability to build that document that can easily be shared between scientists and practitioners, and we can easily reproduce our work. Now, as you move your model more to production, you might choose to refactor some of that notebook.

00:04:59 Speaker 2

Into dedicated Python script, but for now this is enough to get started and be efficient and useful with the Sagemaker platform.

00:05:06 Speaker 2

Now if I want to get to a Jupyter notebook within Sagemaker, I have a simple option. I could use the legacy notebooks feature directly from the navigation bar under application and Ides and choose Notebooks. Now I need to be very clear here. This way of launching Jupyter notebooks is considered quite legacy. It's really been replaced and superseded by the new Sagemaker Studio and I would encourage you to use the Sagemaker Studio instead. However, if you when the Sagemaker product was first launched, this was the only way that you could get a Jupyter hosted notebook.

00:05:36 Speaker 2

So I think it's still useful to understand that it's there, and maybe one day it might be deprecated, but let's look at it and then we can compare it to Sagemaker Studio later on. Now, under Notebook Instances, I can see I've got an option to create a notebook instance. What this will do is launch a managed virtual machine with Jupyter Lab already installed, and we'll simply get a URL that we can open up Jupyterlab from. Now, if you remember from our first exercise, we saw that we could install Jupyter directly onto a Linux instance and look at its web interface. That's great if we were doing it elsewhere here we're consuming.

00:06:07 Speaker 2

Jupyter as a managed service, so you don't get to see the EC2 instance, the virtual machine. You'll see it listed here as a notebook instance, but we can't get to the operating system of it. It's not there for that, it is fully managed. It is there to provide us with a hosted version of Jupyter.

00:06:21 Speaker 2

We are then prompted to create our Jupyter Notebook managed instance. We give it a uniquely identifying name, and we specify its instance type and platform. Now, on instance type, what we're really specifying here is the size of the instance, the size of the virtual machine in terms of number of CPUs and amount of RAM. But as is consistent across the AWS platform, we don't give you a slider specifying how much CPU or RAM. Instead, you must pick from the available T-shirt sizes.

00:06:46 Speaker 2

Like small, medium, large, extra large. And they come from different instance families like the t family for burstable CPU or the C family for compute intensive or the R family for RAM intensive. So you really need to understand does a T3 medium give you 2 CPUs or 4 CPUs? How much RAM does it give you? And this is well documented on the AWS website. For us, we just need to make sure that we allocate sufficient CPU and memory that will allow the Jupyter lab server process to run and have sufficient free resource to run any Python code you intend to run in each of the.

00:07:16 Speaker 2

Code cells of your notebook. Notice we get to pick the version of Linux and the version of Jupyter Lab as well. So there we can see that the T3 medium is going to give us 2 CPUs and 4 gigabytes of RAM. You can then see that our notebook server is being created. It's been created with a name we gave it and the instance size we selected. The status is showing as pending. After a few moments, the status will change from pending to in service.

00:07:38 Speaker 2

And there it is. Now in terms of billing, you are billed from the moment that that instance is marked as in service until you stop the instance. So we should only run these notebook instances when you intend to actually perform some Jupyter notebook work. If you're not using it, switch it off. Now notice under the actions column I have the option of opening Jupyter or Jupyterlab. And if you remember from an earlier lesson, we saw that Jupyter was the initial interface that allows you to work on one notebook at a time, and Jupyterlab was the more modern multi tab interface that also could be extended using.

00:08:08 Speaker 2

Plugins as part of the Jupyter Lab extensibility program. So you're probably wanting to use Jupyterlab in most circumstances.

00:08:15 Speaker 2

Now when you click on Jupiter or Jupiter Lab, this will open up a new tab in your browser and that will have the URL, the form of the name of your notebook. Then there'll be a unique identifier dot notebook dot the region you're in dot sagemaker dot AWS. It'll be the format of the URL for the your actual delivered interface.

00:08:31 Speaker 2

So there's the Jupiter interface and we can see, OK, it's not a multi tab interface. I could create a new Jupyter notebook from here, or I could open an existing one. Notice I have a File Explorer type view here and if I had a Jupyter notebook listed there, I'd be able to click on it and open it. If I chose Jupyter lab, there's my more mature interface where I can see I have my launcher tab and I can open additional tabs. So I could be working on multiple files at the same time. Remember it doesn't just have to be IPYNB files like Jupyter notebooks. Maybe you wanted to have some data files open like a CSV.

00:09:01 Speaker 2

For manual inspection, so you could have those open at the same time and simply flip between the tabs. I can also see down the left hand side. I've got my functionality such as stop and start instances. I've got a Git extension in this interface. I've also got an Amazon Queue extension in here as well. So I've got that extensibility. You can see in the main panel of the launcher window that I could launch a new notebook of type Python 3, or I could launch a new one specifically enabled for Pytorch or Tensorflow, whatever framework I required. I could also click on terminal and get a terminal opened with the managed instance itself.

00:09:30 Speaker 2

Which might be needed if I needed to install an additional Python packages like matplotlib or seaborn or something like that. There are a few characteristics of Sagemaker notebook instances that we need to be aware of.

00:09:41 Speaker 2

We need to be aware of billing that if we create a notebook instance, the moment it is marked as being in service, we are paying for it by the minute.

00:09:49 Speaker 2

So we should therefore size our notebook instances to be only big enough to run Jupyter Lab and whatever code that you run in your code cells. Now, you don't need to have a huge amount of compute allocated to this, because in Sagemaker, when we are going to be doing data processing or going to be doing model training, that compute will not be coming from the Sagemaker Jupyter Notebook instance. We're going to delegate that to Sagemaker to run in a dedicated compute environment. So you no longer need to size your notebook server to be big enough for large data processing jobs. We'll do that for each.

00:10:19 Speaker 2

Job and science that appropriately. So for Jupiter, start off, keep it small. When you're not working in Jupiter notebooks, stop these instances. You don't need to keep them running if you're not working on Jupiter. That way you're not paying for them and it's a good idea for any resources you're not really using to delete them if you are not using them. Provided that you are synchronizing your code, your Jupyter notebook code, that you are synchronizing it back to a git compatible repository such as GitLab or GitHub, then there's no real reason that you need to have a persistent long running Jupyter.

00:10:49 Speaker 2

Server You could simply delete the Jupyter lab server and then in one month or however long you waiting before you work in Jupiter again, you could simply provision a new one and then you'd be back the Jupyter lab. And then you could do a git clone, get your training code back and open up your notebook. So just keep in mind that keeping these things running incurs more cost in your account. Helpful hint is to enable billing alerts in your AWS account. This way you could set up an automated alert that could e-mail you or send you a text notification every time you go over a set.

00:11:20 Speaker 2

So for example, if a daily spend was over \$200 or whatever threshold you specify, then it would send you that alert and that would help you catch any unintended runtime costs. And this has helped save me many times when I realized that I've maybe provisioned a large endpoint or a notebook instance, I forget about it, but then I get the e-mail telling me, oh, your costs are rising above your daily expected costs gives you the opportunity to intersect that before it gets worse and stop the unneeded resource and stop the unneeded billing. So what have we seen in this lesson? We have seen that in the.

00:11:50 Speaker 2

AWS Management Console, we can find Sagemaker AI within the Sagemaker AI. We can use the sidebar to find items such as data processing jobs. We can find where training jobs are, and we can find where our endpoints are. In other words, our hosted models for inference. And we've seen how we can create and access Jupyter notebook servers. We saw that when you initially start with Sagemaker AI, there'll be nothing in processing jobs or training jobs or under hosted models because we've not created anything yet. We'll be creating these entities from code that we will run in our Jupyter.

00:12:20 Speaker 2

The Jupiter notebook instances that we have seen how to create in this lesson are really considered quite legacy now because we now have the Sagemaker Studio, a much more mature user interface for Jupyter Lab that integrates with other parts of Sagemaker. But it's still useful to know that you can get a basic Jupyter Notebook here in the main console user interface. So that takes us to the end of this lesson. In our next lesson, we have a demo on how to create and manage those Jupyter Notebook servers. See you there.

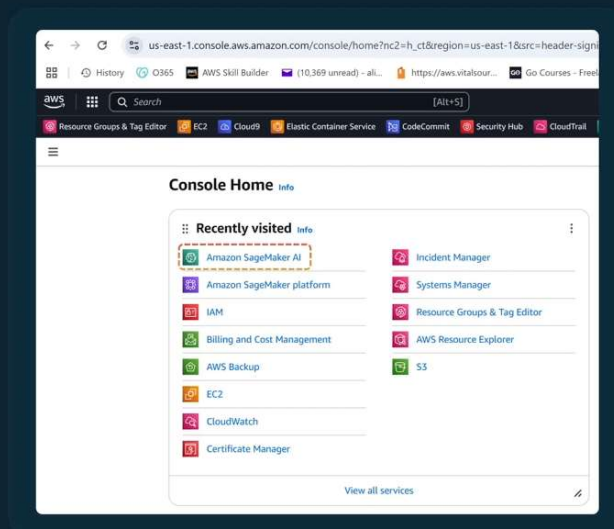
00:12:51 Speaker 2

I

Agenda

- 01 **Problem:** SageMaker Console UI lacks intuitiveness
- 02 **Solution:** Explore key components step by step
- 03 Understanding the workflow
- 04 Reviewing the results

Problem: Unintuitive SageMaker UI



Problem: Unintuitive SageMaker UI

01

SageMaker UI is **complex and overwhelming** for new users.

02

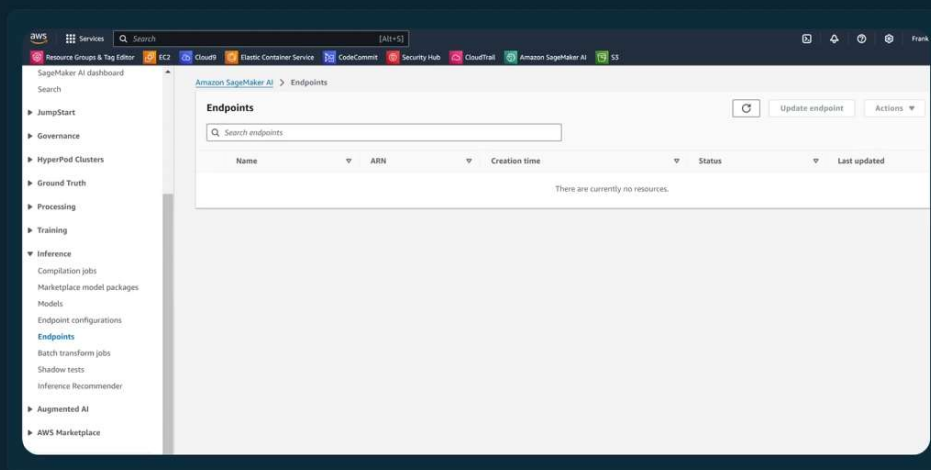
Unlike other AWS services, **clicking through menus doesn't help in learning.**

03

Most of your **time will be spent in Jupyter notebooks**, not the UI.



Problem: Unintuitive SageMaker UI



Problem: Unintuitive SageMaker UI



© Copyright KodeKloud



Problem: Unintuitive SageMaker UI



SageMaker is driven programmatically – from **Jupyter notebooks** and **scripts**.

© Copyright KodeKloud



Workflow: Notebook Instances



Jupyter Notebook



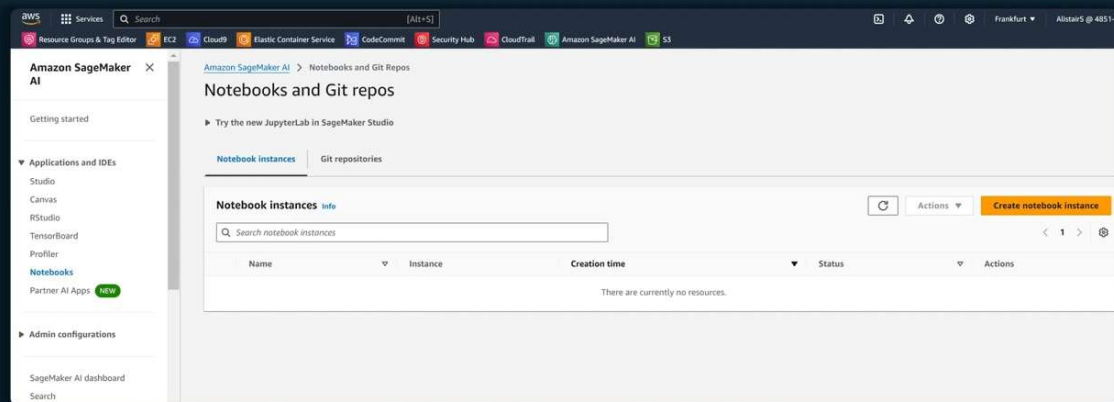
SageMaker SDK



Python Environment



Workflow: Notebook Instances



Hosted Jupyter notebooks can be launched on demand, and you get to control the compute sizing.



Workflow: Notebook Instances

The screenshot shows the 'Create notebook instance' page in the AWS SageMaker console. The page is titled 'Create notebook instance' and includes a brief description of SageMaker notebook instances. The 'Notebook instance settings' section contains the following fields:

- Notebook instance name:** A text input field containing 'my-notebook-server'.
- Notebook instance type:** A dropdown menu with 'ml.t3.medium' selected.
- Platform identifier:** A dropdown menu with 'Amazon Linux 2, Jupyter Lab 3' selected.
- Additional configuration:** A section with a '+' icon to expand.

The 'Permissions and encryption' section includes:

- IAM role:** A section explaining that notebook instances require permissions to call other services. It offers to create a role using the role creation wizard.
- Root access - optional:** Two radio buttons: 'Enable - Give users root access to the notebook' (selected) and 'Disable - Don't give users root access to the notebook'.

© Copyright KodeKloud



Workflow: Notebook Instances

The screenshot shows the 'Notebook instances' page in the AWS SageMaker console. A green banner at the top indicates 'Success! Your notebook instance is being created.' Below the banner, there is a search bar and a table of notebook instances.

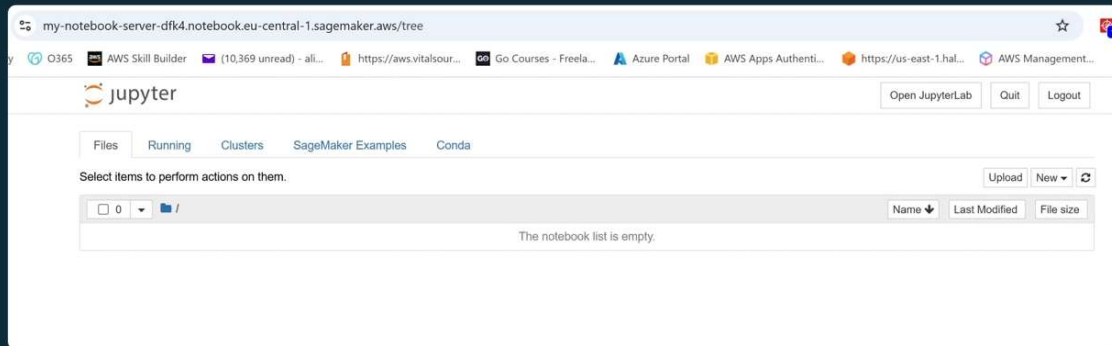
Name	Instance	Creation time	Status	Actions
my-notebook-server	ml.t3.medium	12/16/2024, 12:14:32 PM	InService	Open Jupyter Open JupyterLab

Notebook URL: my-notebook-server-
<unique_id>.notebook.<region>.sagemaker.aws

© Copyright KodeKloud



Results: Notebook Instance

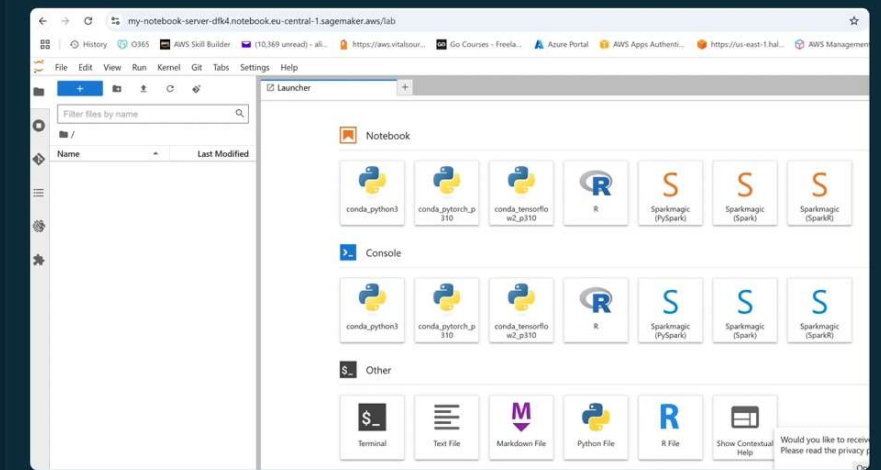


Jupyter

© Copyright KodeKloud



Results: Notebook Instance



JupyterLab

© Copyright KodeKloud



Results: Notebook Instance



Billing continues while the instance is InService

1

Use small instances

2

Stop instances when not in use

3

Delete if not needed



Enable AWS billing alerts to avoid unexpected charges from forgotten active instances!



Summary

01

Locate data processing jobs

02

Track training jobs in progress

03

Manage hosted models (endpoints)

04

Create and access Jupyter Notebook servers

