Run Amazon SageMaker Notebook locally with Docker container

Quy Tang Follow
Aug 19 · 8 min read

Photo by Aleks Dorohovich on Unsplash

Amazon SageMaker, the cloud machine learning platform by AWS, consists of 4 major offerings, supporting different processes along the data science workflow:

. . .

This post targets data scientists and machine learning engineers who are using or planning to use Amazon SageMaker. Basic understanding of Amazon SageMaker and its Notebook instance plus knowledge of Docker would be useful.

. . .

TL;DR

For the busy ones, here is a quick summary:

- The hosted SageMaker Notebook instance comes pre-built with many important features including a comprehensive set of tools and libraries, multiple kernels with latest machine learning frameworks, GPU-support, Git integration and lots of real-world examples.
- Nonetheless, it costs money, expects all data to be uploaded online, requires Internet access and especially AWS Console sign-in, and can be difficult to customize.
- To overcome these drawbacks, I have created a Docker container that offers a similar setup usable locally on any laptop/desktop.
- The replicated features include full Jupyter Notebook and Lab server, multiple kernels, AWS & SageMaker SDKs, AWS and Docker CLIs, Git integration, Conda and SageMaker Examples Tabs.
- The AWS-hosted instance and the local container aren't mutually exclusive and should be used together to enhance the data science experience.

The full source code of the Docker images is hosted on Github: https://github.com/qtangs/sagemaker-notebook-container

. . .

In this post,

I will explore:

- Why do we need a Notebook instance in the first place?
- Why do we need a local Notebook container?
- What features have been replicated?
- What features are missing?
- How can you get started with the container?

. . .

Why do we need a Notebook instance in the first place?

Before exploring the local instance, let's dive into the benefits provided by the Amazon SageMaker Notebook instance:

- **Complete system on demand**: For a data scientist who doesn't deal much with infrastructure, this is the easiest method to get started with Amazon SageMaker. Using the AWS Console, within a few clicks, you can gain access to a comprehensive system equipped with the most commonly needed libraries and tools.
- **Multiple machine learning kernels**: The Notebook instance offers various built-in kernels, including standard Python 2 and 3, popular frameworks such as TensorFlow, MXNet, PyTorch and Chainer, and other runtimes such as R, PySpark.
- **Multiple machine size for different workloads**: You can choose from the most basic multi-purpose instance types like t2, t3 or m4, m5 to compute-optimized (r4, r5) to powerful GPU-based types (p2, p3).
- **Elastic Inference using a GPU machine**: While using p2, p3 generates GPU-powered performance boost, they are expensive. Instead, you can attach an additional Elastic Inference instance (eia1 type) to train and test your models using GPU at a fraction of the cost of p2, p3.

. . .

Why do we need a local Notebook container?

Despite the significant benefits of the AWS-hosted Notebook instance, there are a few drawbacks that a local Notebook container can counter. Here are some benefits of the local Notebook container:

- **Cost reduction**: Running an AWS-hosted instance costs money; furthermore, there's no way yet to use spot instances, which are usually employed to reduce EC2 instances cost. Running a local container costs nothing.
- **Ease of using local data**: Since SageMaker Notebook instance runs in the cloud, it can only access data located online. Any local data has to be uploaded to either S3 or other online storage first. This is not required for a local container, which can use a simple Docker volume mount.
- **Skipping the AWS Console**: AWS Console sign-in is required to access the AWS-hosted instance and there is a timeout of 12 hours, which means you have to log in at least once a day. With a local container, you configure AWS credentials once and don't need to sign in anymore subsequently (except when there's a need to access S3 or other AWS services).
- **Offline access**: Another advantage for the local container is that you can access it anytime without an Internet connection, especially when you want to focus on data

. . .

What features have been replicated?

The main aim of the local Docker container is to maintain as much as possible the most important features of the AWS-hosted instance while enhancing the experience with the local-run capability. Followings are the features that have been replicated:

Jupyter Notebook and Jupyter Lab

This is simply taken from Jupyter's official Docker images with a few modifications to match SageMaker's Notebook settings, including:

- Name the default user as `ec2-user` and allow passwordless `sudo` access.
- Allow custom Miniconda and Conda versions.
- Skip Jupyter Hub
- Set Notebook directory to `/home/ec2-user/SageMaker`

Multi-kernels

Use Conda environments (`conda env create`) to create multiple kernels matching SageMaker's kernel names. For examples:

- `conda_python2`
- `conda_python3`

. . .

What features are missing?

GPU support

Currently, the container is built for CPU workload only. For any local training and testing using GPU, you have to use the AWS-hosted instance. Note that if you run GPU-powered training using SageMaker's training job, then you don't need GPU on the local container.

Other kernels

Other Python-based kernels can be easily added using Conda. However, for R and Spark runtimes, more work will be required in the future.

Local training and inference

With the AWS-hosted instance, you can run training and inference on that instance using SageMaker's local mode. Currently, the Docker container is not set up for this. In the future, network configurations will be added to support this.

Automated update using latest SageMaker settings

At the moment, the kernel configurations require manual updates whenever SageMaker Notebook instance is updated (new SageMaker versions). In the future, these can be automatically updated using CloudWatch Events and AWS Lambda Functions.

. . .

How can you get started with the container?

Instructions can be found at the Github repository: https://github.com/qtangs/sagemaker-notebook-container

. . .

Summary

Amazon SageMaker's Notebook instance is an important part of the AWS cloud machine learning platform.

Replicating it in a Docker container image hopefully further simplifies access to the SageMaker world.

Note that AWS has also provided Docker containers for training and deployment.

In future posts, I will continue looking at machine learning infrastructure using Amazon SageMaker and how we can apply best practice to both simplify and solidify these infrastructure components.

. . .

Thank you for reading. Do comment below to share your thoughts.

The complete project for this work is hosted on Github.

)

Some rights reserved

Sagemaker Docker Local Notebook Machine Learning

Discover Medium

Welcome to a place where words matter. On Medium, smart voices and original ideas take center stage - with no ads in sight. Watch

Make Medium yours

Follow all the topics you care about, and we'll deliver the best stories for you to your homepage and inbox. Explore

Become a member

Get unlimited access to the best stories on Medium — and support writers while you're at it. Just $5/month. Upgrade

About Help Legal