# Running Amazon SageMaker Notebook locally with a Docker container

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

A screenshot of a cell phone

Description automatically generated

Source: <https://docs.aws.amazon.com/sagemaker/latest/dg/how-it-works-mlconcepts.html>

1. **Ground Truth**: A managed service for large scale on-demand data labeling services.  
   (Generate example data, for supervised algorithms)
2. **Training**: A managed service to train and tune models at any scale.  
   (Train a model)
3. **Inference**: A managed service to host pre-built models and run inferences on new data.  
   (Deploy the model)
4. And to put it all together, **Notebook**: An AWS-hosted server for data scientists to access and manage the above 3 functions along with many other data science tasks.

While the hosted **Notebook** service provides a notably convenient way to obtain a complete server for working with Amazon SageMaker, in many circumstances, there is a need for a local set-up, either due to cost, ease of access or ‘customizability’.

**This document details the motivation behind a locally-hosted Notebook Docker container and describe the various functions of the Amazon SageMaker Notebook instance that have been replicated.**

*This work 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, we 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/keshava/dockerized-sagemaker>

The Docker images have been published to Docker Hub: <https://hub.docker.com/r/keshava/sagemaker-notebook>

A few basic questions first:

* 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.
* **Security**: The Notebook instance is secured by default, you need to sign in to the AWS Console to access it.
* **Git Integration**: You can link the Notebook instance with a Git repository, either hosted by AWS CodeCommit or Github or any other Git repo, to enable code version control for all of your work. This is especially important if you work in a team.
* **Samples galore**: To get you started quickly, AWS has embedded in every instance a multitude of examples, the majority coming from [AWS’s Examples](https://github.com/awslabs/amazon-sagemaker-examples) hosted on Github. With minimal effort, you can import the full source code of an example into your own instance.

**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 cleansing, feature engineering, data analysis, …; functions that do not require the training servers.
* **Customization**: While the AWS-provided Notebook instance already contains lots of libraries and tools for general usage, there may still be a need for further customization for specific use cases. SageMaker enables this via lifecycle configurations and direct Terminal access. However, this can feel limited at times, not least requiring knowledge of shell scripting. Having the ability to easily configure a Docker image with all the tools a team of data scientists needs is a plus for the local container.
* **Flexibility**: With the Docker image, a team of data scientists can decide to host it centrally in a high-capacity server, or set up clusters of notebook instances. The options are limitless.

***Note****: The AWS-hosted instance and the local container****aren’t****mutually exclusive. You can use both at the same time, and if you set up Git integration correctly, they can be synced such that you can seamlessly alternate between the 2 choices for your daily work.*

**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](https://github.com/jupyter/docker-stacks) 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*
* *conda\_tensorflow\_p36*

The list of kernels is dynamic, i.e., a docker image can contain 1 or multiple kernels. It is set by the argument *CONDA\_ENVS* while building the Docker images. Initially supported Conda environments include *python2*, *python3*, *tensorflow\_p36*, *mxnet\_p36*. Additional kernels can simply be created by adding Conda environment files in the *base/utils/envs* folder.

**AWS & SageMaker SDKs**

Each kernel includes both the AWS Boto3 SDK and SageMaker SDK. This is essential for all Python code interacting with AWS Services including SageMaker’s training and deployment services.

**AWS CLI**

AWS CLI is installed for shell scripts interacting with AWS.

***Note****: Both the AWS SDK and CLI require AWS credentials configured once in the host machine.*

**Docker CLI**

Many SageMaker examples use *docker* to build custom images for training. Instead of installing a full Docker on Docker, which is a complex operation, we make use of the host’s Docker Engine instead. To achieve that, we install the Docker CLI on the Docker image and rely on the Docker socket of the host machine to connect the host’s Docker Engine. This is achieved by including *-v /var/run/docker.sock:/var/run/docker.sock:ro* when running the container.

***Note****: On Windows, update the mount to:*-v //var/run/docker.sock:/var/run/docker.sock:ro.

**Git Integration in Jupyter Lab**

Git is installed to allow git access directly from the container. Furthermore, the [jupyterlab-git](https://github.com/jupyterlab/jupyterlab-git" \t "_blank) extension is installed on Jupyter Lab for quick GUI interaction with Git.

**Conda Tab**

Just like the AWS-hosted instance, the Docker-based instance contains a Conda tab to manage Conda environments.

***Note****: While this has been included to mimic the actual instance, it’s not recommended that you make changes to your Conda environments from here. Instead, update the corresponding YAML files under*base/utils/envs*and rebuild Docker images so that your changes are recorded and can be shared with others.*

**SageMaker Examples Tab**

All SageMaker examples provided by AWS have been mirrored in the Docker image along with the simple 2-click copy feature so that you can easily take an existing example and try it out. Other examples (from [fast.ai](https://github.com/fastai/course-v3) and [PyTorch](https://github.com/pytorch/tutorials" \t "_blank)) are not included yet but can be included in the future.

**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](https://sagemaker.readthedocs.io/en/stable/overview.html" \l "local-mode" \t "_blank). 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.

**Python libraries**

To keep the Docker images as lean as possible, many libraries have not been included in the given kernels. If you need certain libraries in a container, you can build your own container with a custom environment YAML file in the *base/utils/envs* folder. In the future, I aim to make it easier to customize this.

**Other tools**

Some tools like [nbdime](https://github.com/jupyter/nbdime" \t "_blank), Jupyter Notebook Diff and Merge tools, and [plotlywidget](https://github.com/plotly/plotly.py" \t "_blank), an open-source, interactive graphing library for Python, should be integrated in the future.

**How can you get started with the container?**

Instructions can be found at the Github repository: <https://github.com/keshava/dockerized-sagemaker>

**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](https://github.com/aws/sagemaker-containers) and [deployment](https://github.com/aws?utf8=%E2%9C%93&q=serving&type=&language=).

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