Published in radix.ai blog

Enias Cailliau Follow
Nov 13, 2018 · 5 min read · Listen

Is SageMaker worth it?

Amazon SageMaker is a managed machine learning service (MLaaS). The platform lets you quickly build, train and deploy machine learning models. In this blog post, we will take a look at what SageMaker has to offer and if it is worth the cost.

> *This article is the first article in the series: "Accelerating deep learning development using SageMaker". During this series, I explore the added value of using SageMaker for Deep Learning development. Part 1 provides a general overview of what SageMaker has to offer. This second part provides a more technical view on how development lifecycles can be accelerated using the SageMaker ecosystem.*

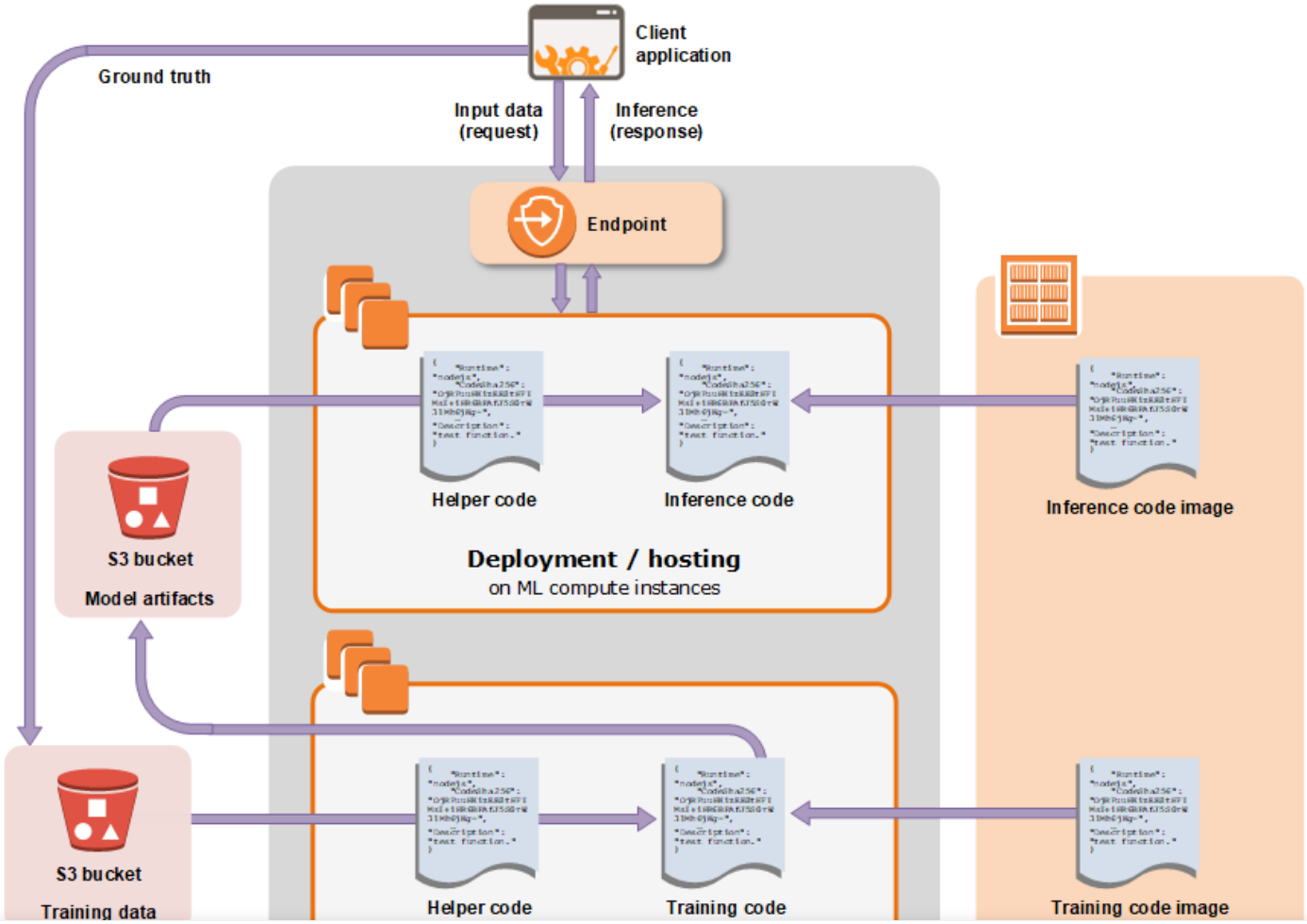
What it has to offer

SageMaker is more than a service; it's a development platform for machine learning practitioners. It is the perfect balance between low-level EC2 instances and high-level services such as Rekognition and Comprehend: It offers a fully-managed environment that facilitates building, training and deploying machine learning at scale.

The platform achieves this goal by offering a Jupyter Notebook instance as a starting point. From this notebook, you can use SageMaker's services to train a model remotely in the cloud, distribute model tuning and easily deploy your model. All these services are offered through a high-level Python API.




DeepAR forecasting which uses RNN. For the purists that don't want to use the prebuilt models, SageMaker also supports TensorFlow and Apache MXNet out-of-the-box.



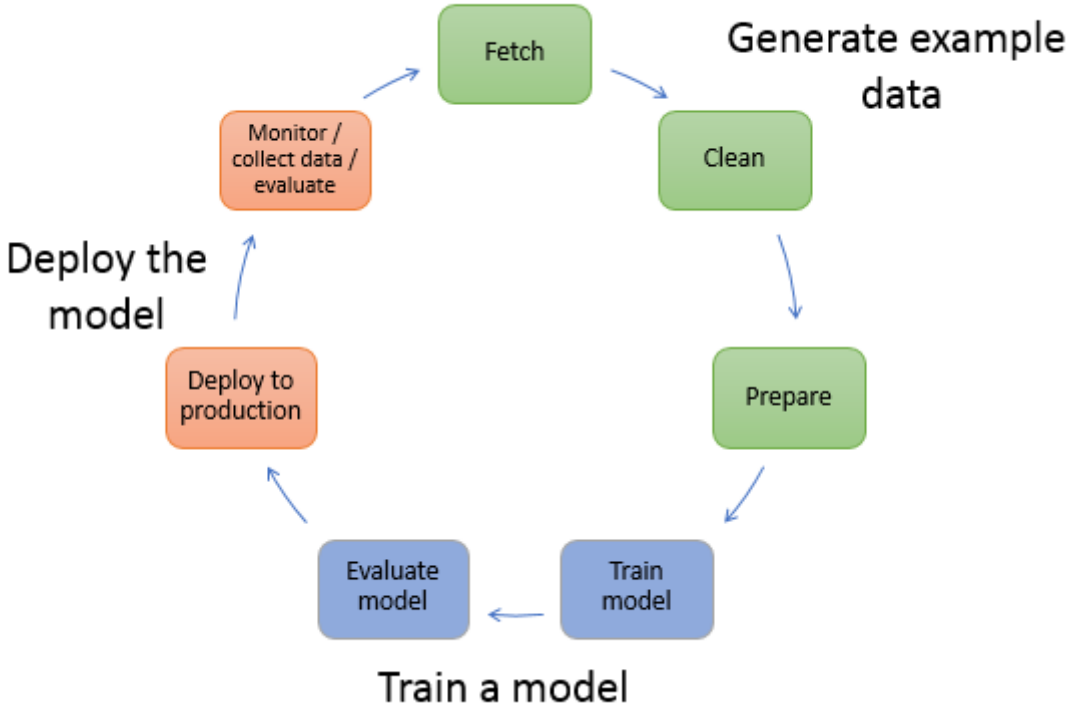
Behind the scenes, SageMaker employs two concepts: Docker images and s3 storage. Using these two concepts, it can host your training or inference code on any instance you desire. Using Docker as a machine abstraction also means that you can provide a docker image to SageMaker which contains your ML model in any framework or programming language you want.

Why you should be using SageMaker

Let's analyse a typical workflow (illustrated in the image down below) and think where SageMaker will make things better. First, you generate your training and evaluation dataset using an ETL process. Once that's done (and stored on s3) we can train our model and evaluate its performance. If we are satisfied with the result, we can deploy our model into production and serve predictions through an HTTP API.

In most cases, the three different stages of the workflow have different hardware requirements; ETL is mostly CPU and memory bound while training a model demands expensive GPU machines. Contrastingly, inference on a production-ready model is not as compute intensive which means you want to deploy it on much cheaper hardware. These different hardware requirements mean

SageMaker abstracts this hardware management so that we can focus on what we do best: developing groundbreaking models. In practice, this means that we can use different machines for each step of the workflow with only a single line of code.



Typical workflow for creating a machine learning model (source). Green corresponds to a notebook instance, blue — training instances, red — deployment instances.

When exploring a new service, it's always a good idea to analyse the additional cost it will introduce. Amazon isn't clear how it prices SageMaker compared to traditional EC2 instances. The only indication is the usage of ml instances instead of normal ones. After some calculation we came to the following conclusion: the usage of SageMaker introduces a 40% increase in cost compared to running EC2 instances. 40% is a significant increase; when training a large model on a Tesla V100 instance, the hourly rate is $4,2 compared to $3 when using lower-level EC2 instances. This price increase means you are spending an additional $1.2 each hour you train your model through the SageMaker Service.

Other cloud providers such as Paperspace and FloydHub also offer GPU accelerated ML workloads. Both advertise cheaper machine costs but require a subscription. This payment model usually ends up being more expensive than just paying for machine usage as you do in SageMaker. Furthermore, SageMaker offers more integration such as the support for distributed training compared to its competitors.

The Good, the Bad and the Ugly

To experiment with the SageMaker platform I've developed a TensorFlow model

functions: Automatic Model Tuning and PIPE input mode. The Automatic Model Tuning service performs hyperparameter tuning using Bayesian Optimisation. Tuning hyperparameters tends to be painfully slow, but SageMaker reduces execution time significantly by distributing the compute workload across multiple instances. PIPE input plays a considerable role in making the distributed schema work. This input mode avoids training latency by streaming data directly from s3 to your model. All the SageMaker's functionality requires minimal effort to use them.

The good

- Out-of-the-box support for popular ML frameworks such as TensorFlow and Apache MXNet
- Support for multi-node training even for your own models
- No configuration required to train models in the cloud
- Notebooks can change host without losing information
- Main SageMaker keeps track of your ongoing services (train/deploy/hyperopt)
- Both Jupyter notebook and Jupyter lab are available

- 40% price bump on every instancetype

The ugly

- Documentation is sometimes not as conclusive as you would want

Conclusion

SageMaker accelerates machine learning development by abstracting server management and deployment set-ups. During my experiment, I found that SageMaker delivers sufficient added value for development workflows to justify the 40% price bump.

Convinced that SageMaker is for you? I recommend that you start with the SageMaker documentation and examples. If you want a guide that uses a TensorFlow model and all the bells and whistles from SageMaker, you can check out this tutorial (to be added soon).