SageMaker Serverless Inference Is Now Generally Available

Exploring The Latest SageMaker Inference Option

Ram Vegiraju

Published in

Towards Data Science

.

5 min read

.

Apr 21, 2022

33

1

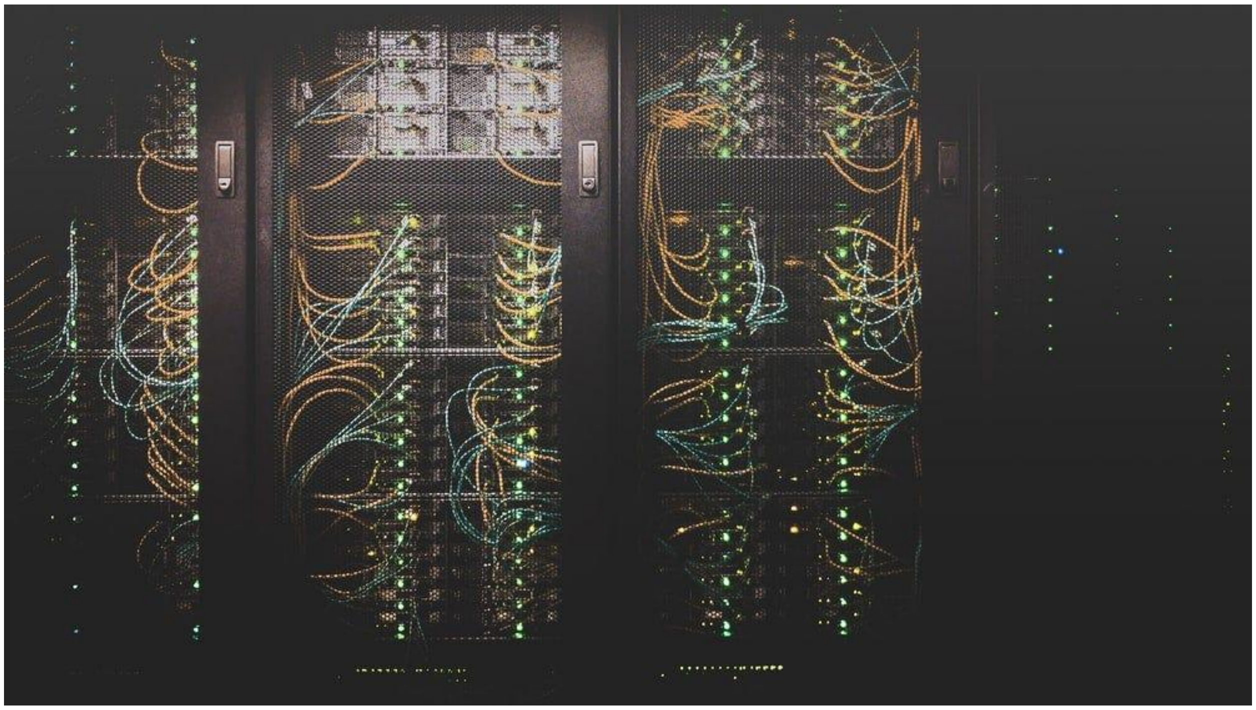


Image from Unplash by Taylor Vick

I’ve been super excited to write this article. ML Inference is super interesting in itself. Add serverless to it and it becomes that much more interesting! When we talked about sServerless Inference before we had to look at potentially using services such as AWS Lambda.The problem with services such as Lambda is that they don’t have the managed ML infrastructure tooling provided out of the box. You have to build, manage, and maintain all your containers, ML infrastructure by yourself.

SageMaker Serverless Inference abstracts all of this out. It allows for you to focus on the model building process, there’s no heavy lifting that needs to be done in maintaining your own container as you would have to with Lambda. Serverless Inference takes the same core foundations from other SageMaker Inference options. It still has support for AWS

Deep Learning Images/Frameworks and the flexibility of the Bring Your Own Container (BYOC) approach. If you have existing real-time inference workloads on SageMaker, you can keep much of the same code and add the snippet that includes a Serverless Configuration. In this article, we'll explore an example of deploying a Sklearn model to a SageMaker Serverless Endpoint.

NOTE: For those of you new to AWS, make sure you make an account at the following **link** if you want to follow along. There will be costs incurred through the deployment process, especially if you leave your endpoint up and running. This article will also assume intermediate knowledge of SageMaker and AWS.

Table of Contents

When To Use Serverless Inference

At the moment SageMaker Inference has four main options: Real-Time Inference, Batch Inference, Asynchronous Inference, and now Serverless Inference. In this past article, I've explained the use-case for the first three options. So when do you use Serverless Inference?

Serverless Inference is a great option when you have **intermittent** and **unpredictable workloads**. Ideally your workload should be **tolerant** to experiencing **cold-starts** as with any Serverless solution. Another major reason to use Serverless Inference is there is **no infrastructure management**. If you don't want to deal with AutoScaling or instance management/setup then Serverless Inference is a great option. The other major value proposition of Serverless Inference is the **cost savings**. You are charged by invocation for Serverless Inference as opposed to an hourly billing with Real-Time Inference. This is a great option if your team is building a **POC** on SageMaker and does not want to incur large expenses in the process.

What Does Serverless Inference Support?

After the Serverless Inference Preview Launch in Reinvent 2021, a few key features have been added. SageMaker Python SDK support is enabled, which makes it easier than ever to train and deploy supported containers/frameworks with Amazon SageMaker for Serverless Inference. You can also work with the Model Registry for Serverless Inference, this will give flexibility to add serverless endpoints to your MLOps workflows. Lastly, the **maximum concurrent invocations** have increased to **200 per endpoint** (50 at the time of preview).

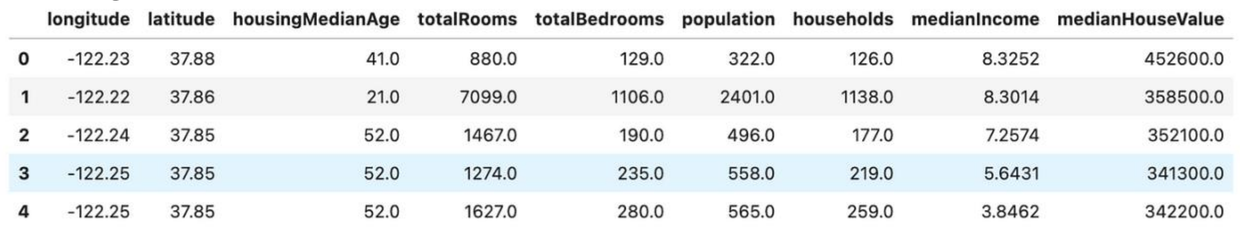
Serverless Inference Example

For our Serverless Inference example, we'll be working with training and deploying a Sklearn model with the California Housing dataset. This dataset is publicly available in the SageMaker sample datasets repository, we will show you can retrieve it in your notebook.

For setup we'll be working with **SageMaker Studio** with a **Python3 Data Science kernel**. You can also use **Classic Notebook Instances** and work with **ml.m5.xlarge Notebook Instance** and **conda_python3 kernel**. Once you're in your notebook we will setup our S3 Bucket and training instance.
S3 & Training Setup

Next we will retrieve the California Housing dataset from the public SageMaker samples. This is a **regression problem** that we will be solving using the **Sklearn framework**.
Retrieve dataset

We can then read the dataset using Pandas to ensure that we have properly created our DataFrame.
Reading dataset



	longitude	latitude	housingMedianAge	totalRooms	totalBedrooms	population	households	medianIncome	medianHouseValue
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	452600.0
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	358500.0
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	352100.0
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	341300.0
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	342200.0

Dataset head (Screenshot by Author)

We then upload this dataset to S3, which is where SageMaker will access the training data and dump model artifacts.

Upload data to S3

Sklearn is one of the supported Deep Learning Containers that SageMaker provides, so we can directly grab the image using the SageMaker Python SDK without having to deal with any Docker related work.

Sklearn Estimator

Here we pass in an entry point script that contains our model and inference functions. We won't cover Script Mode in this example in depth, but take a look at this article to understand how to train Sklearn model on Amazon SageMaker.

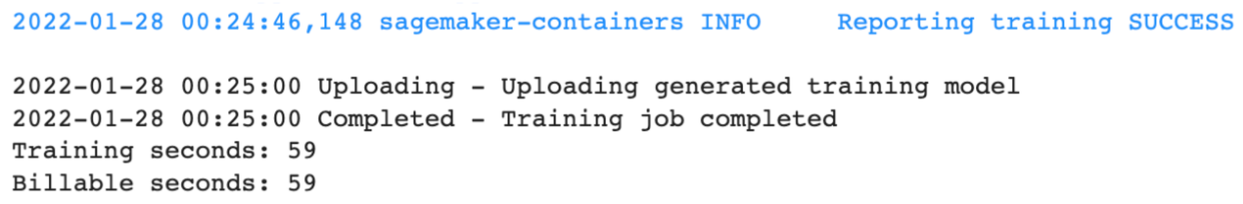
If we take a quick look at the training script we can see we're working with a Random Forest Model.

Model in training script

Note the function that handles input as well. The **input_fn** is what determines the type of **data format** you can pass in for your **model inference**. To learn more about inference handlers, check out this article.

Now we can execute model training with our Sklearn estimator.

Train Model



```
2022-01-28 00:24:46,148 sagemaker-containers INFO     Reporting training SUCCESS

2022-01-28 00:25:00 Uploading - Uploading generated training model
2022-01-28 00:25:00 Completed - Training job completed
Training seconds: 59
Billable seconds: 59
```

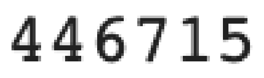
Training Completed (Screenshot by Author)

With training complete, we're now ready for Inference. Everything up until this point is the exact same as it would be with Real-Time Inference or any of the other SageMaker Inference option. This is where we can add a ServerlessConfig through the SageMaker Python SDK and attach it to our endpoint.

The two parameters here are **MemorySize** and MaxConcurrency. MemorySize generally should be at the **minimum** the **same size** as your **model**. You can set memory to the following values: **1024 MB, 2048 MB, 3072 MB, 4096 MB, 5120 MB, or 6144 MB**. **MaxConcurrency** is the **max concurrent invocations** for a single endpoint, which has now been increased to **200** for GA. These are the only two parameters you have to specify for a Serverless Endpoint, all infrastructure work and management is taken care of under the hood.
Serverless Config Object

Now we can deploy our endpoint as we would traditionally and attach this object.
Serverless Endpoint Deployment

Now we can see a sample inference to this endpoint.
Sample Invocation



```
446715
```

Inference (Screenshot by Author)

Additional Resources & Conclusion

SageMaker-Deployment/sklearn-serverless.ipynb at master · RamVegiraju/SageMaker-Deployment
Compilation of examples of SageMaker inference options and other features. …
github.com

For the entire code for the example access the link above. For an example around using Model Registry with Serverless Inference check out this example. For a HuggingFace example with Serverless Inference check out this blog and example. For the official AWS blog release of Serverless Inference check out the following article.

As always I hope this was a good article for you with SageMaker Inference, feel free to leave any feedback or questions in the comments. If you're interested in more AWS/SageMaker related content check out this list that I have compiled for you.

If you enjoyed this article feel free to connect with me on LinkedIn and subscribe to my Medium Newsletter. If you're new to Medium, sign up using my Membership Referral.