AutoScaling SageMaker Real-Time Endpoints

Bring Your ML Workloads To Production

[Ram Vegiraju](#)
Published in

Towards Data Science

.
5 min read
.
Dec 31, 2021
17

Image from Unsplash by Alp Duran

It's one thing to have an endpoint up and running for inference. It's another thing to make sure that endpoint can handle your expected traffic. With SageMaker Real-Time endpoints numerous factors need to be considered when it comes to launching models in production. What is the instance type you are using for the endpoint? More importantly for this use case, **how many instances** do you have backing the endpoint?

For this post we will look into AutoScaling SageMaker Endpoints to deal with traffic. At its simplest form you can have a SageMaker real-

time endpoint backed by one instance. Sometimes this instance can get overworked and this can lead to failures and inference errors from your endpoint. To combat this you can create an AutoScaling policy that will help your instances fan out based off of CloudWatch metrics such as Invocations Per Instance or CPU Utilization.

For the example in today's article we will build an AutoScaling policy off of an Amazon XGBoost deployed endpoint. To get a walkthrough of deploying the Amazon XGBoost algorithm, follow this article first.

NOTE: For those of you new to AWS, make sure you make an account at the following **link** if you want to follow along. **This article will assume an intermediate level of knowledge with AWS and SageMaker.**

Table of Contents

1. AutoScaling Options

There's a few different ways you can scale your endpoints; the different AutoScaling policy options are defined below.

Target Tracking Scaling: With Target Tracking you can specify a metric to monitor. For SageMaker check out the CloudWatch metrics that are supported and you can pick one of these and monitor it. You can set a threshold for this metric and if met it can scale out as you have defined. Note that you can also define custom metrics with SageMaker.

Scheduled Scaling: With Scheduled Scaling you can set up your own scaling schedule. For example, if you know traffic will be higher on a certain day of the week you can build your policy to scale up on that specific day.

Step Scaling: Step Scaling gets a lot more specific and you can define specific thresholds that will trigger your policy. The power of Step Scaling comes in that it scales based off of the degree of the alarm breach. If the threshold is violated to a larger extent, greater scaling will be applied.

For our specific example we will be working with Target Tracking Scaling. To understand all other scaling options available check out the documentation.

2. Deploy XGBoost Endpoint

Let's quickly walkthrough deploying one of the Amazon provided algorithms: XGBoost. We will use it for a regression use case with the Abalone dataset. For a deeper dive on each of the steps check out

my previous article which goes more in depth on using the algorithm for training and deployment. For this section we'll quickly run through model deployment and skip the training process.

We'll be using the Boto3 Python SDK to work with the SageMaker client for model deployment. The first step is grabbing our model data and creating our SageMaker model.

SageMaker Model Creation

Next we can use our SageMaker model to define our Endpoint Configuration.

Endpoint Configuration Creation

We specify we only want to start with one instance to back our endpoint. We can then use this Endpoint Configuration to create the Endpoint which will take a few minutes.

Endpoint Created

We now have our endpoint created, we can feed a sample data point to test invoking the model for inference.

Invoke Endpoint

```
b'4.566554546356201'
```

Sample Result (Screenshot by Author)

3. AutoScale Endpoint

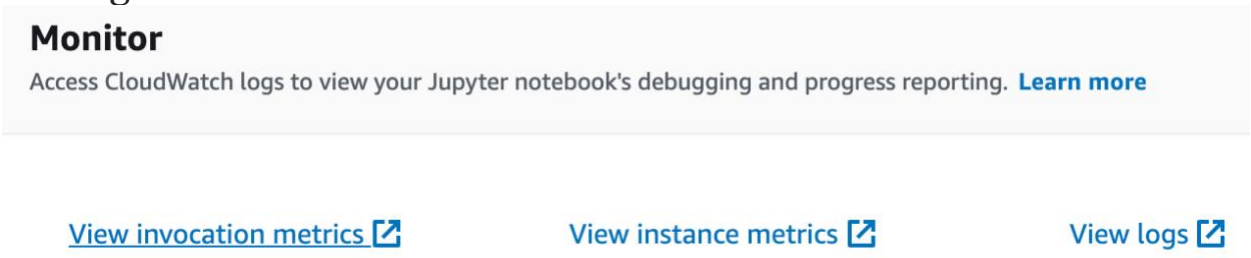
We can now focus on defining a scaling policy to handle traffic. We will use a TargetTracking policy and the metric we focus on will be InvocationsPerInstance. We can set the threshold for 10 invocations

per instance. We can define this policy using the Boto3 Python SDK and the AutoScaling client.
AutoScaling policy

Note that we defined **maximum capacity** as **4 instances**, with a minimum of 1 instance behind our endpoint. Now we can define the threshold, target metric, as well as our **Scale In** and **Scale Out** times. These two times will be how long it takes for our instances to scale out and then scale back in if our threshold is not being hit in regards to invocations per instance.
Target Tracking Scaling

Now we can test this policy with some stress testing. We will use the invoke endpoint Boto3 call and send requests for a certain duration. Remember that our Scale Out period takes 60 seconds.
Test endpoint

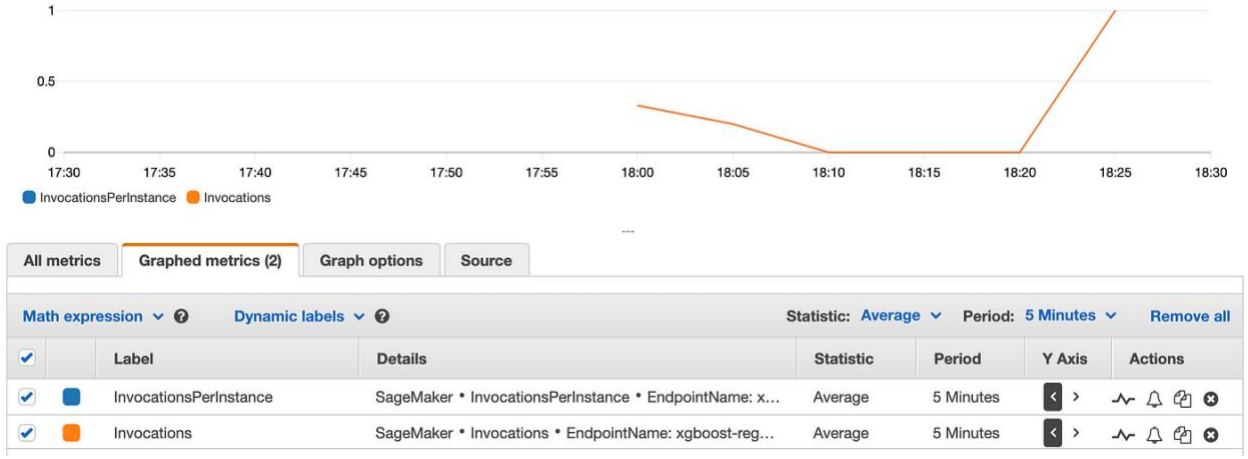
The endpoint should be invoked for 250 seconds, we can monitor this through CloudWatch Invocation Metrics for our specific endpoint on the SageMaker console.



CloudWatch Metrics for SageMaker (Screenshot by Author)

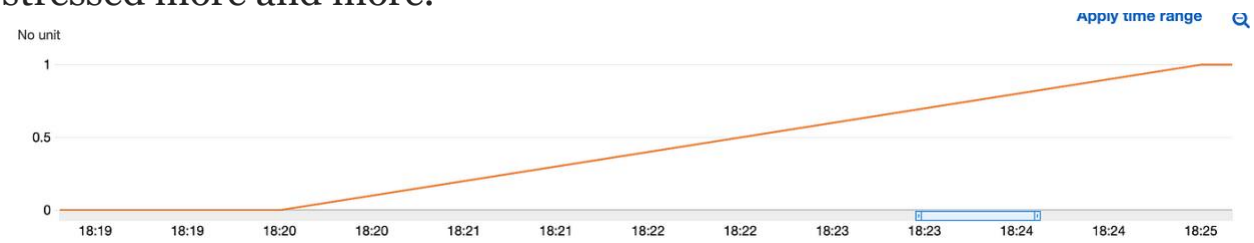
Instance Min-Max Set from 1–4 (Screenshot by Author)

Now we can monitor the CloudWatch graphs as our endpoint is stressed in those 250 seconds.



Invocations Metric (Screenshot by Author)

We can zoom into specific intervals to see our endpoint getting stressed more and more.



Invocations Metric (Screenshot by Author)

After our endpoint has been stressed, we can use Boto3 to monitor our endpoint description updating to reflect our instances scaling out.

Monitor Endpoint Instances

```
Status: Updating
Status: Updating
Current Instance count: 1
Status: Updating
Current Instance count: 1
Status: Updating
```

Endpoint Updating (Screenshot by Author)

After a little you should see the endpoint has been updated to have four instances.

```
Status: Updating
Current Instance count: 4
Status: InService
Current Instance count: 4
```

Updated Endpoint (Screenshot by Author)

4. Additional Resources & Conclusion

Reference Blog: https://aws.amazon.com/blogs/machine-learning/configuring-autoscaling-inference-endpoints-in-amazon-sagemaker/

Code for example: https://github.com/RamVegiraju/SageMaker-Deployment/blob/master/AdvancedFunctionality/AutoScaling/XGBoost-ASG/XGBoost-Abalone.ipynb

SageMaker AutoScaling helps you launch models into production in numerous ways. Through the plethora of scaling options, along with the detailed CloudWatch monitoring you can define your policy based off of what you're expecting in regards to your traffic.

Another neat feature to check out is SageMaker Inference Recommender, this new feature helps load test and pick optimal instance types for your endpoint. I've attached a few more resources below if interested in more SageMaker related content.

- SageMaker Bring Your Own Container/Custom Framework
- Deploying Custom TensorFlow Models on Amazon SageMaker
- SageMaker Multi-Model Endpoints
- SageMaker Inference Examples Repository

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.

Machine Learning

AWS