

# Machine Learning Implementations and Operations Quiz 2

## Model deployed inside an AWS Lambda function

Deploying a SageMaker model inside a Lambda function is an approach that may be adopted if the following are true:

- The model data, input data, and other required files are not too large.
- The inference endpoint only needs to be invoked a few times.


This method also enables the end user to be able to use all of the services that may be interconnected with a Lambda function such as an API Gateway. One downside is that a lambda function has the tendency to pause and have to cold start but this may simply be mitigated by pinging the function at a set interval.

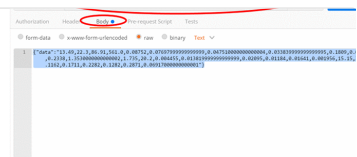
Using this approach is not advisable if the the amount of data to be ingested is too large and if the endpoint will need to be continuously invoked since it will do more harm than good given that launching an EC2 instance is a more practical and cost-effective alternative compared to using a Lambda function in that scenario.

### Reference:

Call an Amazon SageMaker model endpoint using Amazon API Gateway and AWS Lambda | Amazon Web Services

April 2021 - This post has been updated to ensure the solution walkthrough reflects the changes made to relevant AWS services and notebooks. At AWS Machine Learning (ML) workshops, customers often ask, "After I deploy an endpoint, where do I go from there?" You can deploy an Amazon SageMaker trained and validated ML model as [...]

 <https://aws.amazon.com/blogs/machine-learning/call-an-amazon-sagemaker-model-endpoint-using-amazon-api-gateway-and-aws-lambda/>



## Model deployed in Fargate

This approach is one of the newer methods since Fargate has only been around since 2017. This may be considered as an alternative if the end user wishes to deploy several docker containers that house single to multiple models.



AWS offers existing containers for PyTorch, MXNET, and TensorFlow, among others, that are available across multiple regions.

The model training is essentially the same in that most of it is done in the SageMaker notebook and the relevant and necessary files are stored in a docker container. If an external container is to be used, a task definition needs to be created. Afterwards, the task definition needs to be registered and stored in the Fargate backend to be able to run the specified task/s. Following this, instead of using the built-in .deploy method, a Fargate cluster needs to be created instead. Once the cluster has been verified to be running, an inference task may be performed.



As of writing, AWS Fargate does not support GPU instances.

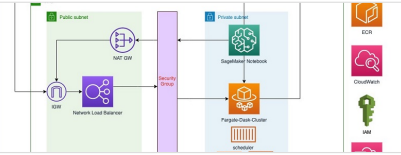
The final step is to build a URL for the inference endpoint with the inference task IP address. Inferences may then be performed by simply using the json and requests library for preprocessing the input data and generating the responses from the Fargate cluster.

### References:

## Machine learning on distributed Dask using Amazon SageMaker and AWS Fargate | Amazon Web Services

As businesses around the world are embarking on building innovative solutions, we're seeing a growing trend adopting data science workloads across various industries. Recently, we've seen a greater push towards reducing the friction between data engineers and data scientists. Data scientists are now enabled to run their experiments on their local

 <https://aws.amazon.com/blogs/machine-learning/machine-learning-on-distributed-dask-using-amazon-sagemaker-and-aws-fargate/>



- <https://www.youtube.com/watch?v=JcOG9DmwSrY>