

Columbia University in the City of New York

AI and OR at Scale on the Cloud

Assignment 5

Group 2: Harsh Mehta (hsm2148) Anunay Sanganal (avs2160) Skand Upmanyu (su2236)

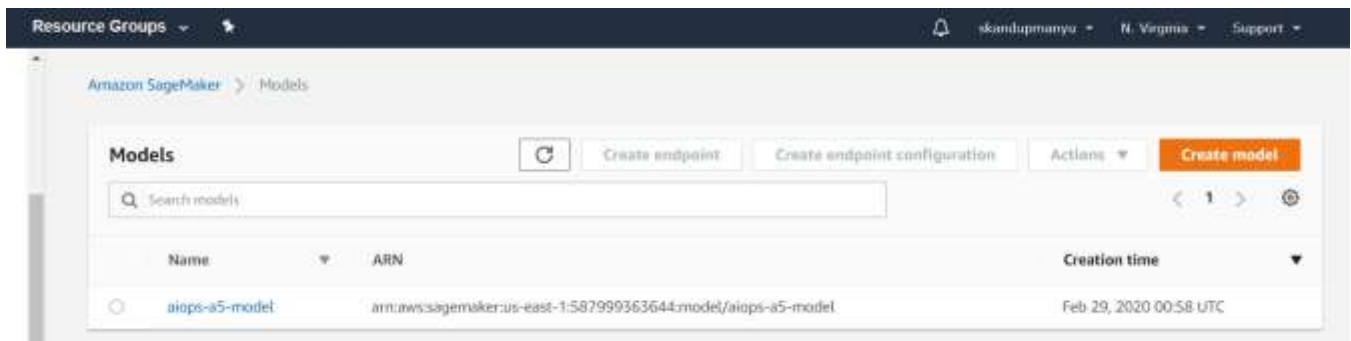
In this assignment you'll be deploying your model as a service on AWS.

If you didn't succeed in training a model and saving it with tensorflow save, you can use this one for this assignment <https://aiops-2020-public.s3.us-east-2.amazonaws.com/model.tar.gz>. Otherwise, use your own.

1. Sagemaker inference

Following the example in class, deploy your model as a SageMaker inference endpoint. Make sure to edit the model in the endpoint configuration to select the cheapest instance: "mk.t2.medium" When creating a "Model" use this image: 763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference:1.14.0-cpu

Creating model:



Amazon SageMaker > Models > aiops-a5-model

aiops-a5-model

Actions ▼ Create batch transform job Create endpoint

Model settings

Name	ARN	Creation time	IAM role ARN
aiops-a5-model	arn:aws:sagemaker:us-east-1:587999363644:model/aiops-a5-model	Feb 29, 2020 00:58 UTC	arn:aws:iam:587999363644:role/service-role/AmazonSageMaker-ExecutionRole-20200227T175313 ↗

Container 1

Container Name	Model data location
Container 1	s3://aiops-a5-skand/model.tar.gz ↗
Image	Mode
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference:1.14.0-cpu	Single model
Training job	
-	

Creating endpoint config:

Resource Groups ▼ ✱ skandupmanyu N. Virginia Support

Amazon SageMaker > Endpoint configuration

Endpoint configuration

🔄 Apply to endpoint Clone Actions ▼ [Create endpoint configuration](#)

🔍 Search endpoint configuration

Name	ARN	Creation time
<input type="radio"/> aiops-a5-endpoint-config	arn:aws:sagemaker:us-east-1:587999363644:endpoint-config/aiops-a5-endpoint-config	Feb 29, 2020 01:01 UTC

Amazon SageMaker > Endpoint configuration > aiops-a5-endpoint-config

aiops-a5-endpoint-config

Delete Apply to endpoint

Endpoint configuration settings

Clone

Name	ARN	Encryption key	Creation time
aiops-a5-endpoint-config	arn:aws:sagemaker:us-east-1:587999363644:endpoint-config/aiops-a5-endpoint-config	-	Feb 29, 2020 01:01 UTC

Enable data capture	Data capture options	S3 location to store data collected	Capture content type
No	-	-	-

Sampling percentage (%)

-

Production variants

Model name	Training job	Variant name	Instance type	Elastic Inference	Initial instance count	Initial weight
aiops-a5-model	-	variant-name-1	m1t2.medium	-	1	1

Creating endpoint:

Resource Groups > Endpoints

skandupmanyu N. Virginia Support

Amazon Elastic Inference

Amazon Elastic Inference adds GPU acceleration to any Amazon SageMaker or EC2 instance for faster inference at much lower cost, with up to 75% savings. Find out if Elastic Inference is right for you.

[Learn more](#)

Endpoints

Update endpoint Actions Create endpoint


Search endpoints

Name	ARN	Creation time	Status	Last updated
aiops-a5-endpoint	arn:aws:sagemaker:us-east-1:587999363644:endpoint/aiops-a5-endpoint	Feb 29, 2020 01:01 UTC	InService	Feb 29, 2020 01:10 UTC

Amazon SageMaker > Endpoints > aiops-a5-endpoint

aiops-a5-endpoint Delete

Endpoint settings

Name	Status	URL
aiops-a5-endpoint	 InService	https://runtime.sagemaker.us-east-1.amazonaws.com/endpoints/aiops-a5-endpoint/invocations Learn more about the API
ARN	Creation time	
arn:aws:sagemaker:us-east-1:587999363644:endpoint/aiops-a5-endpoint	Fri Feb 28 2020 20:01:46 GMT-0500 (Eastern Standard Time)	
	Last updated	
	Fri Feb 28 2020 20:10:06 GMT-0500 (Eastern Standard Time)	

Data capture settings

Enable data capture	Current sampling percentage (%)	S3 location to store data collected
No	-	

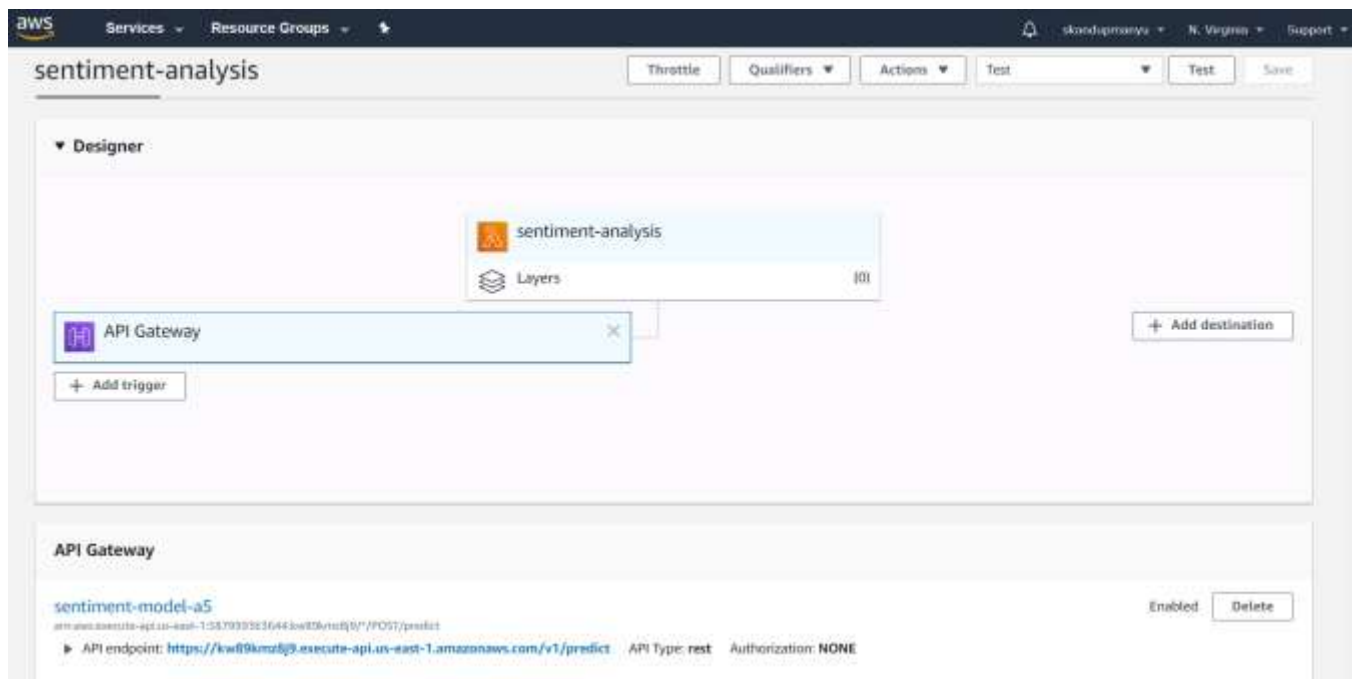
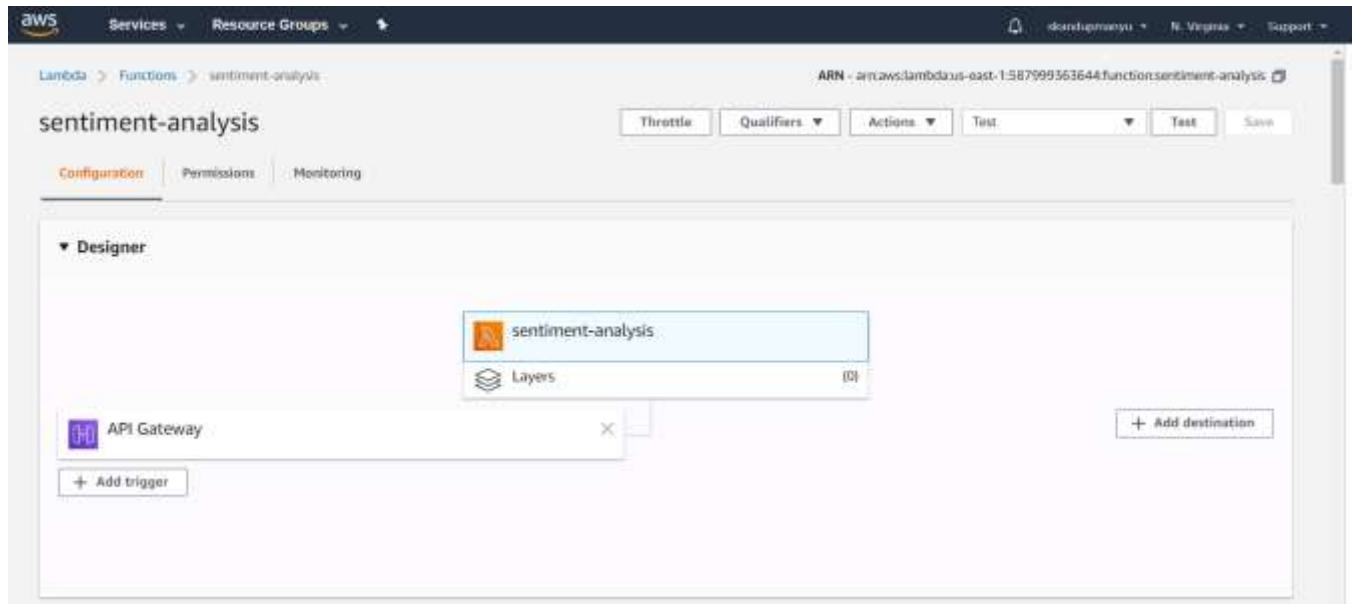
2. Lambda functions

Following the example in class, build a lambda function that performs:

- Pre processing using your code from HW3
- Model inference using your SageMaker endpoint
- Post processing using a logic demonstrated in class

Your lambda function should take a JSON input with a “tweet” key and produce a JSON output with a “sentiment” key and a value that can either be “positive” or “negative” based on the model prediction.

Creating lambda:



sentiment-analysis

Throttle Qualifiers Actions Test Test Save

Execution result: succeeded (logs)

Details

The area below shows the result returned by your function execution. [Learn more](#) about returning results from your function.

```
{
  "sentiment": "negative",
  "request_time_stamp": "2020-02-29 04:24:12.987997",
  "tweet": "I hate my life",
  "probability": 0.0419713147,
  "preprocess_time": 0.00066427257995689547,
  "model_inference_time": 0.23485040664672852
}
```

Summary

Code SHA-256	Request ID
99efK6dETzr00s7TgXU/NB0R3YxJ8wgxavOXmZxM+	0b367674-59c2-4b2f-b8b7-df886e7961b6
Duration	Billed duration
467.33 ms	500 ms
Resources configured	Max memory used
128 MB	84 MB Init Duration: 415.24 ms

Log output

The section below shows the logging calls in your code. These correspond to a single row within the CloudWatch log group corresponding to this Lambda function. [Click here](#) to view the CloudWatch log group.

```
START RequestId: 0b367674-59c2-4b2f-b8b7-df886e7961b6 Version: $LATEST
```

3. Payload logging

Modify your lambda function to implement payload logging.

After the post processing, your lambda should be logging a JSON object to a bucket in S3.

This object should have the following items:

- Date and time of the request
- Tweet
- Sentiment
- Probability from the model
- Pre processing time
- Model inference time

Each request should create a unique JSON object in your payload S3 directory.

Amazon S3 > alops-a5-skand

alops-a5-skand

Overview Properties Permissions Management Access points

Q Type a prefix and press Enter to search. Press ESC to clear.

Upload + Create folder Download Actions

US East (N. Virginia)

Viewing 1 to 2

Name	Last modified	Size	Storage class
logs	-	-	-
model.tar.gz	Feb 28, 2020 7:58:29 PM GMT-0500	10.5 MB	Standard

Viewing 1 to 2

Amazon S3 > alops-a5-skand > logs

alops-a5-skand

Overview

Q Type a prefix and press Enter to search. Press ESC to clear.

Upload + Create folder Download Actions

US East (N. Virginia)

Viewing 1 to 4

Name	Last modified	Size	Storage class
2020-02-29 03:42:30.879420.txt	Feb 28, 2020 10:42:32 PM GMT-0500	221.0 B	Standard
2020-02-29 03:51:49.534685.txt	Feb 28, 2020 10:51:50 PM GMT-0500	219.0 B	Standard
2020-02-29 03:58:09.086293.txt	Feb 28, 2020 10:58:10 PM GMT-0500	217.0 B	Standard
2020-02-29 03:58:49.779669.txt	Feb 28, 2020 10:58:50 PM GMT-0500	219.0 B	Standard

Viewing 1 to 4

4. REST API

Following the example in class, create an API Gateway to expose your lambda function.

The gateway should implement a “/predict” resource with a “POST” request method.

Deploy it under a “v1” stage.

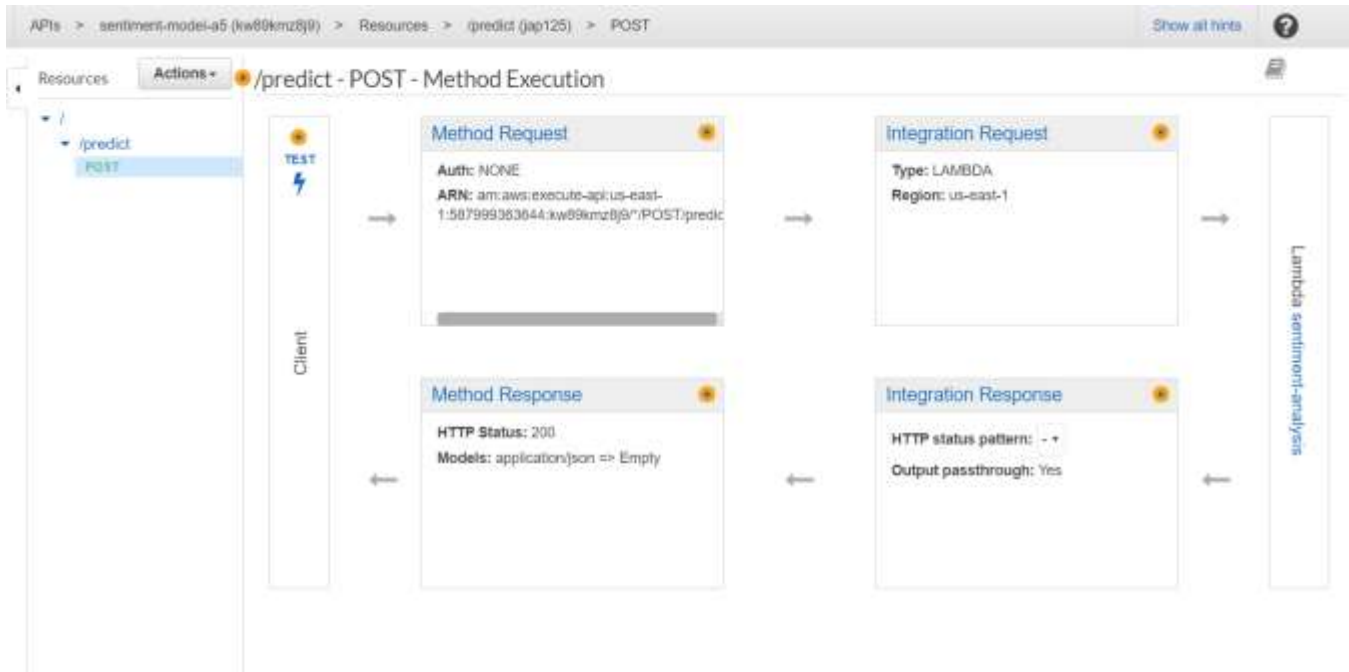
The following kind of request:

```
curl -X POST https://<your_endpoint>/v1/predict --header
"Content-Type:application/json" --data '{"tweet": "I love
apple"}'
```

Should return something like:

```
{"sentiment": "positive"}
```

Creating API gateway predict resource:



The screenshot displays the 'v1 Stage Editor' in the AWS API Gateway console. The breadcrumb navigation at the top indicates the path: `APIs > sentiment-model-a5 (kw89kmz8j9) > Stages > v1`. The left sidebar shows the stage tree with `v1` selected. The main area is titled `v1 Stage Editor` and shows the 'Settings' tab. The 'Invoke URL' is `https://kw89kmz8j9.execute-api.us-east-1.amazonaws.com/v1`. The 'Settings' tab is active, and the following settings are visible:

- Cache Settings**:
 - Enable API cache: ☐
- Default Method Throttling**:
 - Choose the default throttling level for the methods in this stage. Each method in this stage will respect these rate and burst settings. Your current account level throttling rate is 10000 requests per second with a burst of 5000 requests. [Read more about API Gateway throttling](#)
 - Enable throttling: ☒
 - Rate: 10000 requests per second
 - Burst: 5000 requests
- Web Application Firewall (WAF)**: [Learn more.](#)
- Select the Web ACL to be applied to this stage.

Test API Gateway:

The screenshot shows the AWS API Gateway console interface for testing a method. The left sidebar shows the resource tree with `/predict` selected. The main panel is titled `/predict - POST - Method Test` and contains the following sections:

- Path:** A text box containing `/predict`. Below it, a note states: "No path parameters exist for this resource. You can define path parameters by using the syntax `{myPathParam}` in a resource path."
- Query Strings:** A section for defining query parameters. It shows a text box with `(predict) param1=value1¶m2=value2`.
- Headers:** A section for defining request headers. It shows a text box with `(predict)` and a note: "Use a colon (:) to separate header name and value, and new lines to declare multiple headers. eg. `Accept:application/json`."
- Request:** A summary of the request details: `/predict`, Status: 200, Latency: 1253 ms.
- Response Body:** A JSON object representing the response body:

```
{  "sentiment": "Positive!",  "request_time_stamp": "2020-02-29 04:10:09.909650",  "tweet": "I love my life",  "probability": 0.902990732,  "preprocess_time": 0.0005931854246046875,  "model_inference_time": 0.2513895034790039}
```
- Response Headers:** A JSON object representing the response headers:

```
{  "X-Amzn-Trace-Id": "Root=1-5e50e420-eac7cad977fffd4c7db2ebf5;sampled=0",  "content-type": "application/json"}
```

Test API Gateway from local:

```
(base) dyn-160-39-160-85:~$ stand curl -X POST https://kw89kmz8j9.execute-api.us-east-1.amazonaws.com/v1/predict --header "Content-type:application/json" --data '{"tweet": "I love apple"}'
{"sentiment": "Positive!", "request_time_stamp": "2020-02-29 03:58:00.000293", "tweet": "I love apple", "probability": 0.897613311, "preprocess_time": 0.0082209844512939453, "model_inference_time": 0.2180}(base) dyn-160-39-160-85:~$ stand
```

API Gateway Link: <https://kw89kmz8j9.execute-api.us-east-1.amazonaws.com/v1/predict>

Github Link: <https://github.com/harsh1495/AI-Ops-A5>