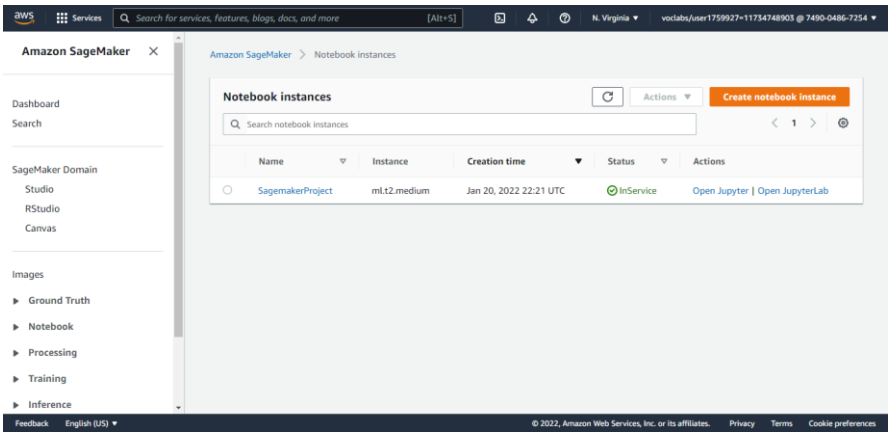
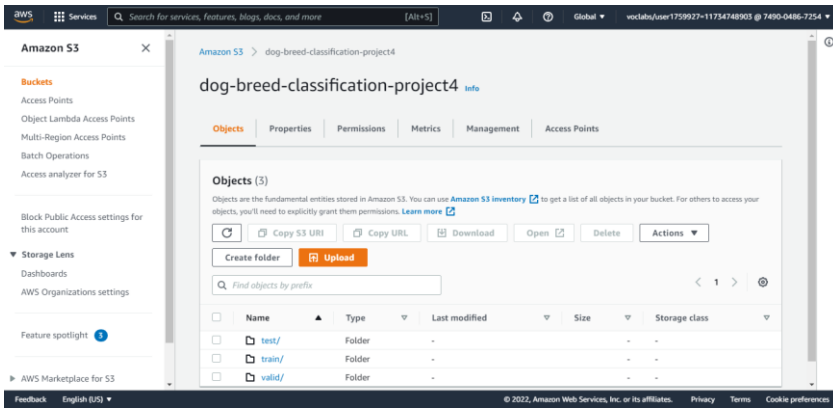
Operationalizing Machine Learning on SageMaker

Training and Deployment 1

- Every notebook in SageMaker needs a computing instance in order to run and we use the notebook instance to create and manage Jupyter notebooks for preprocessing data and to train and deploy machine learning models.
- There are many types of instances, and every time we run an instance we will need to make a good choice for the type of instance to run.
- The instance type we choose needs to be sufficient for your computing needs, but it should also minimize costs.
- Though, I have chosen the “ml.t2.medium” instance type for Notebook instance.
- And to guarantee for completing the execution of this particular project’s jupyter notebooks we do not need a very computationally powerful CPU and high RAM.
- But We will need to keep this notebook instance in “in Service” status for a long time while we are working on the project So, avoid high costs we should select a notebook that is low in per hour cost and also offers reasonably good CPU and RAM.

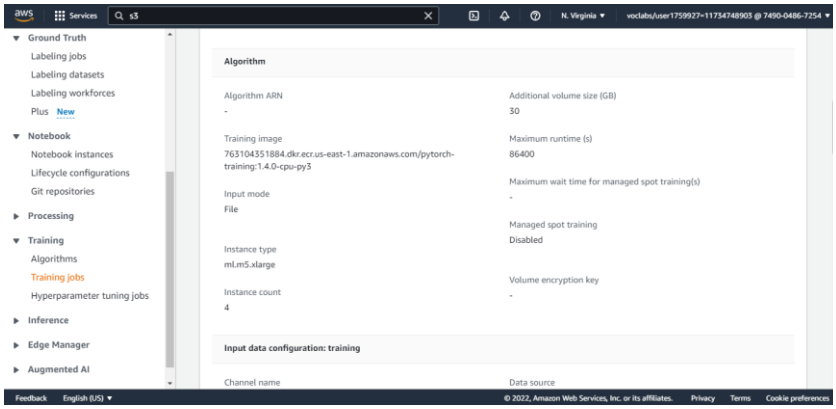


- The dog breed dataset for this was uploaded to a newly created S3 bucket, successfully.

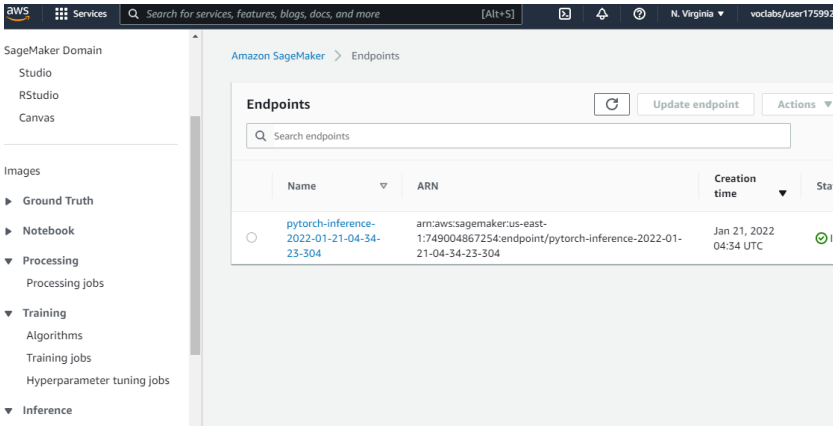


- At hyperparameter Tuning I used
    - Instance type= ml.p3.2xlarge
    - Max jobs = 2
    - Max parallel jobs = 2

- Multi-instance Training
- training data on multiple machines is very hard to implement and requires a lot of intricate configurations to get different machines to work together.
- One of the great things about AWS and SageMaker is that they make multi-instance training much easier.
- Multi-instance training in a term for fitting is machine learning models using multiple separate computers or servers



- **Deployed Endpoints:**



EC2
2

- **Using a special type of EC2 instance called a Spot Instance minimize costs is very important.**
- **EC2 Instances: computing resources that are less expensive than SageMaker instances, but offer fewer managed services**
- **Spot Instances: special types of EC2 instances that consist of idle resources that other AWS customers have reserved. They are less expensive than EC2 instances, but they are not reliable in general.**
- **During the process of opening a new instance, I was able to select an option to launch a spot instance instead of a standard EC2 instance.**
- **And At Launch Instances I needed to select what's called an AMI, Amazon Machine Image and this is essentially for the operating system.**
- **During the process of opening a new instance I faced no problems lunching:**

- Then I can choose an instance type choice and because the model training could take long, I had to change the default instance type to:

- To ensure Flawless model training.
- The important step is where I configure the instance in order to make it a Spot Instance.
- Though, the costs for Spot Instances are usually quite low because these are idle extra resources.
- And to prepare for EC2 model training I Start by launching and connecting to an EC2 instance as described in the project.
- But when I started the Training, I actually faced multiple errors which was Unable to activate multiple environments.

- And with Udacity mentor help he advised me to change the AMI to be:
- 

- As the required environments pre-installed.
- And Finally, the Model Training was done in Less time and flawlessly aslo saved.

★ Using Amazon EC2 eliminates our need to invest in hardware up front, so you can develop and deploy applications faster.

★ Also, we can use Amazon EC2 to launch as many or as few virtual servers as we need, configure security and networking, and manage storage.

★ Also, Amazon EC2 provides Virtual computing environments, known as instances.

★ And, another feature Amazon EC2 provides Preconfigured templates for our instances, known as Amazon Machine Images (AMIs), that package the bits we need for our server (including the operating system and additional software)

★ Also, Various configurations of CPU, memory, storage, and networking capacity for our instances, known as *instance types*.

★ Last but not Least, there are many other Amazon EC2 features provided for us.

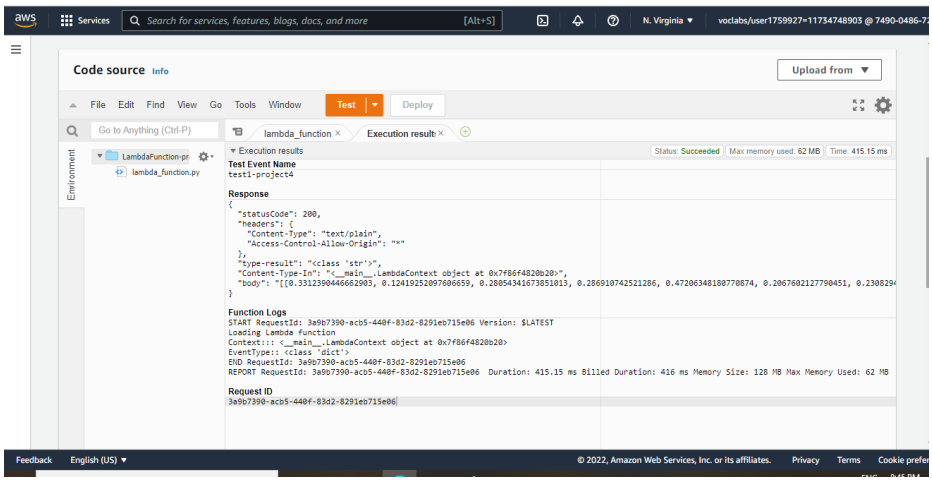
Lambda Function 3

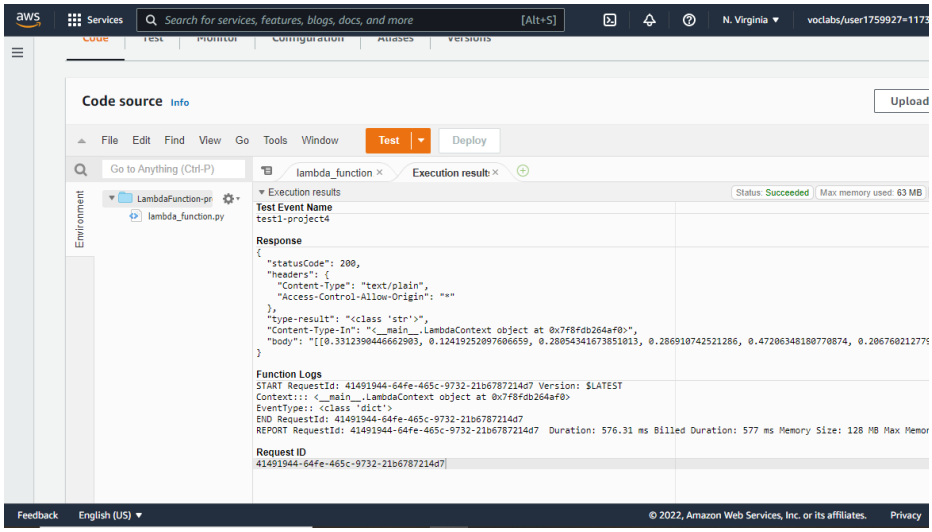
- Lambda functions enable my model and its inferences to be accessed by API's and other programs, so it's a crucial part of production deployment.
- I have sated up a Lambda function that uses Python 3 for its runtime.
- And I attached in the Lambda function the starter code provided by the project resources.
- in this file called lambdafunction.py contains some of the basic Python code for a Lambda function, but I needed to make an important adjustment which was changing the endpoint name endpoint_name variable, to give it the same name as the endpoint I deployed in the first in the project.

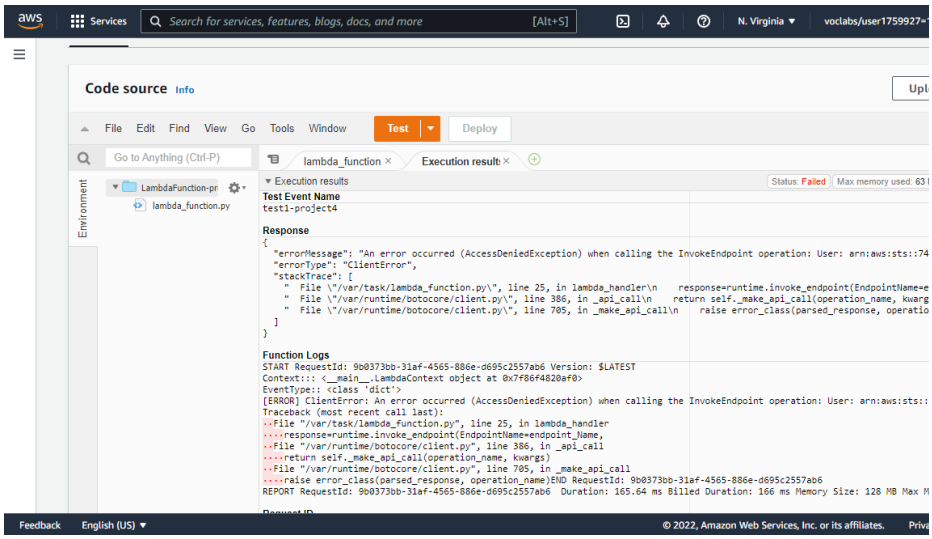
Lambda function security 4

- By default, Lambda functions are not given permission to invoke your SageMaker endpoints.
- And In order to allow the Lambda function to invoke my endpoint, I need to make adjustments to the IAM settings.
- Making security adjustments like this is common in industrial ML projects as mentioned in the lesson.
- When I tested my Lambda function:
    - an error that says that this particular role is not authorized to invoke the endpoint.

- This is a default security setting to keep my projects safe.
- And to solve this issue I need to navigate to IAM:
    - Then, in “Roles” section in the role associated with my Lambda function.
    - Inside this role, the policies that are attached to my role.
    - So, I attached a policy called AmazonSageMakerFullAccess.
    - This is a policy that will allow my Lambda function to access any resource in SageMaker.
- Finally, Lambda function, and succussed on the test event.
- In result, Lambda function is delivering model predictions without any security.
- And yes, I do believe that AWS workspace is secure and by using Amazon Workspaces, we have no data leakage and no copying or downloading of files and data.
- Also, AWS has improved flexibility, scalability, and reliability on the Workspaces.
- With AWS Workspaces we can securely access various data sources and we can control who we give access to and what tools and level of access each user gets.
- We have recognized benefits that the AWS Workspaces technology provides such as flexibility, scalability and secure access.




Code source Info
Upload
File Edit Find View Go Tools Window
Test
Deploy
Go to Anything (Ctrl-P)
lambda_function
Execution result
Execution results
Status: Succeeded
Max memory used: 63 MB
Test Event Name
test1-project4
Response
{
"statusCode": 200,
"headers": {
"Content-Type": "text/plain",
"Access-Control-Allow-Origin": "*"
},
"type-result": "<class 'str'>",
"Content-Type-In": "<__main__.LambdaContext object at 0x7f8fdb264af0>",
"body": "[[0.3312390446662903, 0.12419252097606659, 0.28054341673851013, 0.286910742521286, 0.47206348180770874, 0.20676021277
}
Function Logs
START RequestId: 41491944-64fe-465c-9732-21b6787214d7 Version: $LATEST
Context::: <__main__.LambdaContext object at 0x7f8fdb264af0>
EventType:: <class 'dict'>
END RequestId: 41491944-64fe-465c-9732-21b6787214d7
REPORT RequestId: 41491944-64fe-465c-9732-21b6787214d7 Duration: 576.31 ms Billed Duration: 577 ms Memory Size: 128 MB Max Memor
Request ID
41491944-64fe-465c-9732-21b6787214d7
Feedback
English (US)
© 2022, Amazon Web Services, Inc. or its affiliates.
Privacy


Code source Info
File Edit Find View Go Tools Window
Test
Deploy
Go to Anything (Ctrl-P)
lambda_function
Execution result
Execution results
Status: Failed
Test Event Name
test1-project4
Response
{
"errorMessage": "An error occurred (AccessDeniedException) when calling the InvokeEndpoint operation: User: arn:aws:sts::74
"errorType": "ClientError",
"stackTrace": [
" File \"/var/task/lambda_function.py\", line 25, in lambda_handler\n response=runtime.invoke_endpoint(EndpointName=e
" File \"/var/runtime/botocore/client.py\", line 386, in _api_call\n return self._make_api_call(operation_name, kwarg
" File \"/var/runtime/botocore/client.py\", line 705, in _make_api_call\n raise error_class(parsed_response, operatio
]
}
Function Logs
START RequestId: 9b0373bb-31af-4565-886e-d695c2557ab6 Version: $LATEST
Context::: <__main__.LambdaContext object at 0x7f86f4820af0>
EventType:: <class 'dict'>
[ERROR] ClientError: An error occurred (AccessDeniedException) when calling the InvokeEndpoint operation: User: arn:aws:sts::
Traceback (most recent call last):
File "/var/task/lambda_function.py", line 25, in lambda_handler
response=runtime.invoke_endpoint(EndpointName=endpoint_name,
File "/var/runtime/botocore/client.py", line 386, in _api_call
return self._make_api_call(operation_name, kwargs)
File "/var/runtime/botocore/client.py", line 705, in _make_api_call
raise error_class(parsed_response, operation_name)END RequestId: 9b0373bb-31af-4565-886e-d695c2557ab6
REPORT RequestId: 9b0373bb-31af-4565-886e-d695c2557ab6 Duration: 165.64 ms Billed Duration: 166 ms Memory Size: 128 MB Max M
Feedback
English (US)
© 2022, Amazon Web Services, Inc. or its affiliates.

Concurrency
5

- Concurrency refers to the ability of a Lambda function to serve multiple requests simultaneously.
- I set up concurrency for my Lambda functions to allow our Lambda function to be able to access three instances to reply to multiple requests simultaneously.
- This is called reserved concurrency and The advantage is that reserved concurrency has a relatively low cost.
- The other type of concurrency is called provisioned concurrency.
- Provisioned concurrency creates instances that are always on and can reply to all traffic without requiring a wait for startup times.
- Though it can be more flexible and it can achieve low latency even in very high traffic scenarios.
- Choosing a high number for provisioned concurrency is suitable for very high-traffic projects, because it will turn on instances that can be used by your Lambda function any time.
- Choosing a high number for reserved concurrency incurs no additional cost so in this project I have the flexibility to choose any number in this project was 5.
- It will allow me to choose a high number for provisioned concurrency, since provisioned concurrency must be lower than reserved concurrency.
- Choosing lower numbers for either or both types of concurrency would be more suitable for lower-traffic projects, or projects with lower budgets.
- provisioned concurrency is expensive, so I kept this number low which was 2.

Auto-Scaling 6

- Autoscaling for endpoints allows my deployed endpoints to respond to multiple requests at the same time.
- So, I Choose my maximum instance count to be = 3
    - My endpoint will be able to automatically scale to any number of instances up to the maximum instance count I select.

- Then I Choose what's called a scale-in cool down time period to be = 30.
    - The scale-in period is the amount of time AWS will wait before deploying more instances for your endpoint.
    - If I choose a high number, then AWS will wait a longer time before deploying more instances.
    - And, this helps me avoid incurring more costs for momentary spikes in traffic.
    - If I choose a low number, then AWS will deploy instances more quickly, but this responsiveness will be more costly.
- Choose a scale-out cool down time period also to be =30.
    - The scale-out cool down period is the amount of time AWS will wait before deleting extra deployed instances.
    - If I choose a low number, then AWS will wait only a short time before deleting extra deployed instances.
    - And, this helps you avoid incurring costs for momentary spikes in traffic.
    - However, If I choose a high number, then AWS will keep extra instances deployed longer, but this extra capacity will be more costly.