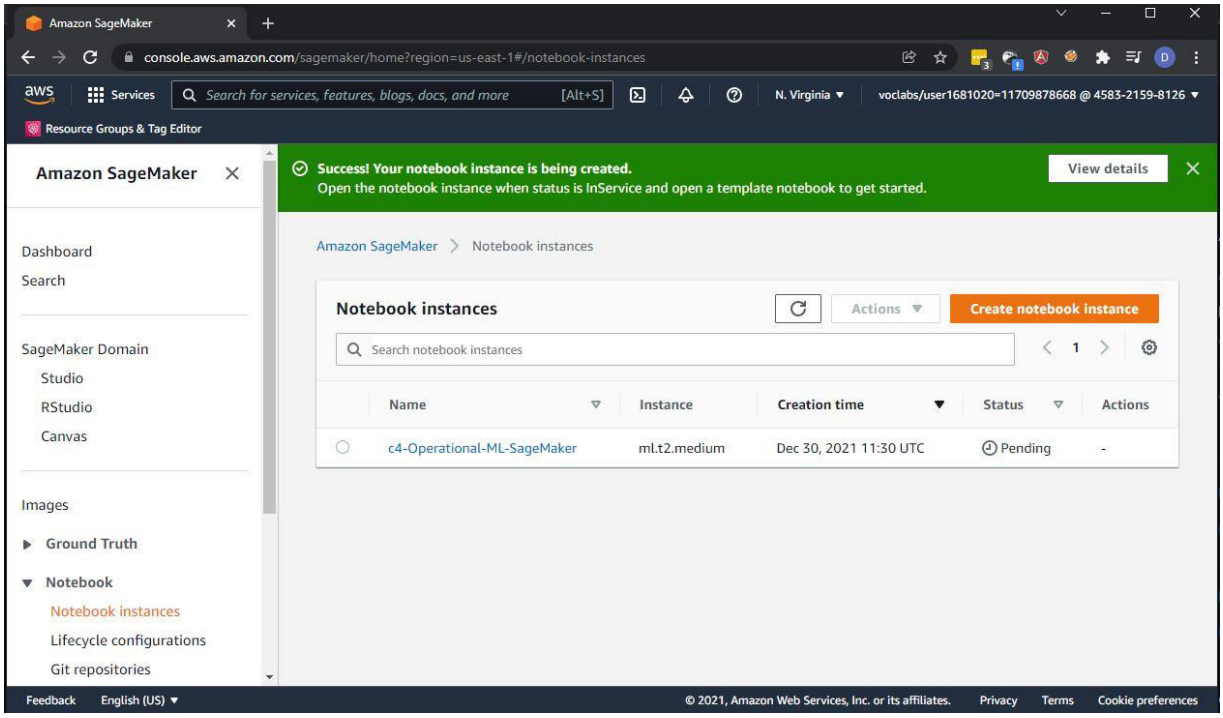
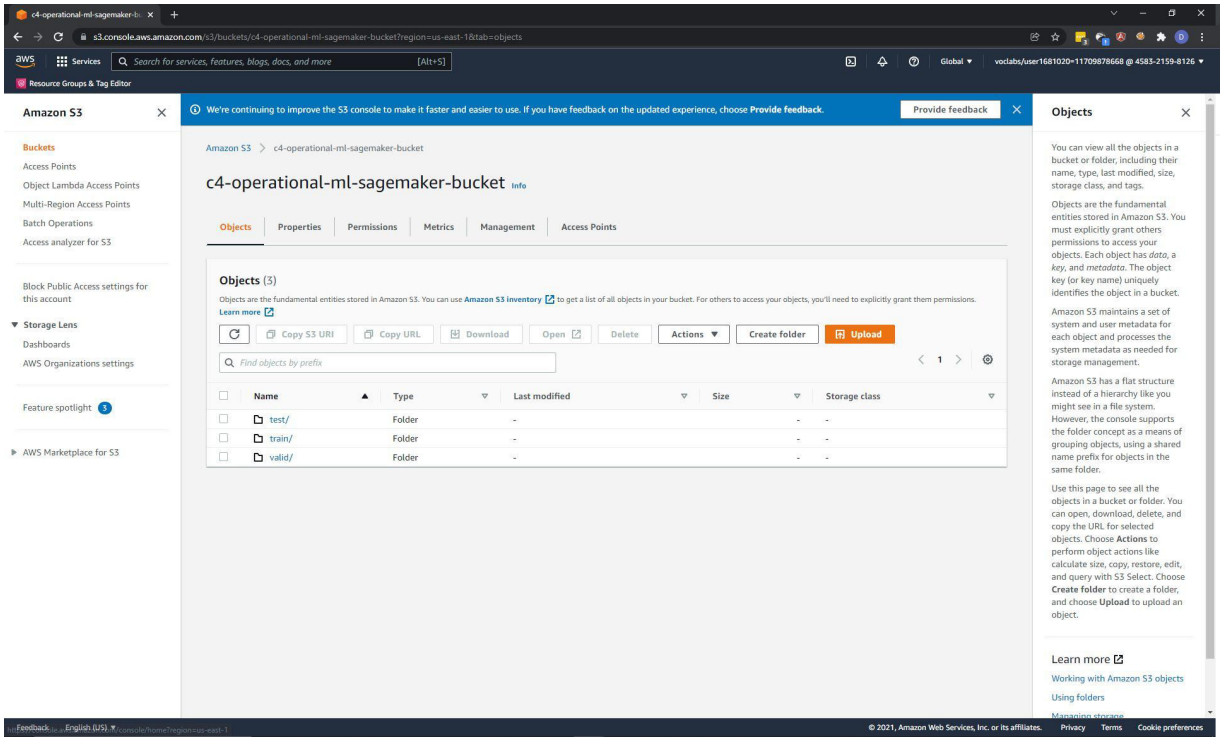
Operationalizing Machine Learning on SageMaker

Step 1 Training & Deployment

I chose the smallest SageMaker instance accessible for my notebook,'ml.t2.medium' , because I need to keep the notebook open through out the project and don't need a very powerful instance in terms of CPU or RAM.

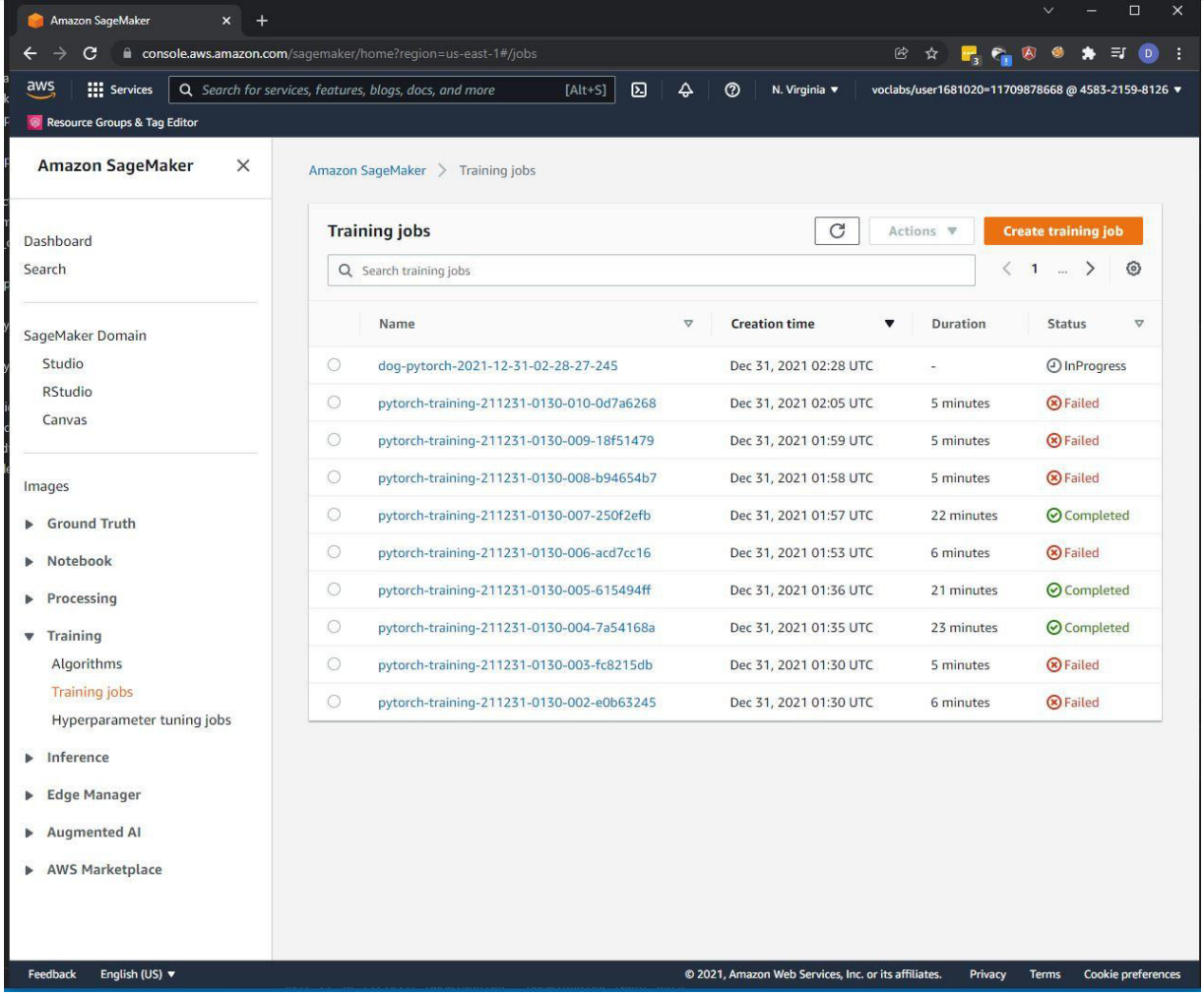




I have then added the address of my S3 bucket `s3://c4-operational-ml-sagemaker-bucket` to all my relevant cell in the train_and_deploy-solution.ipynb notebook.

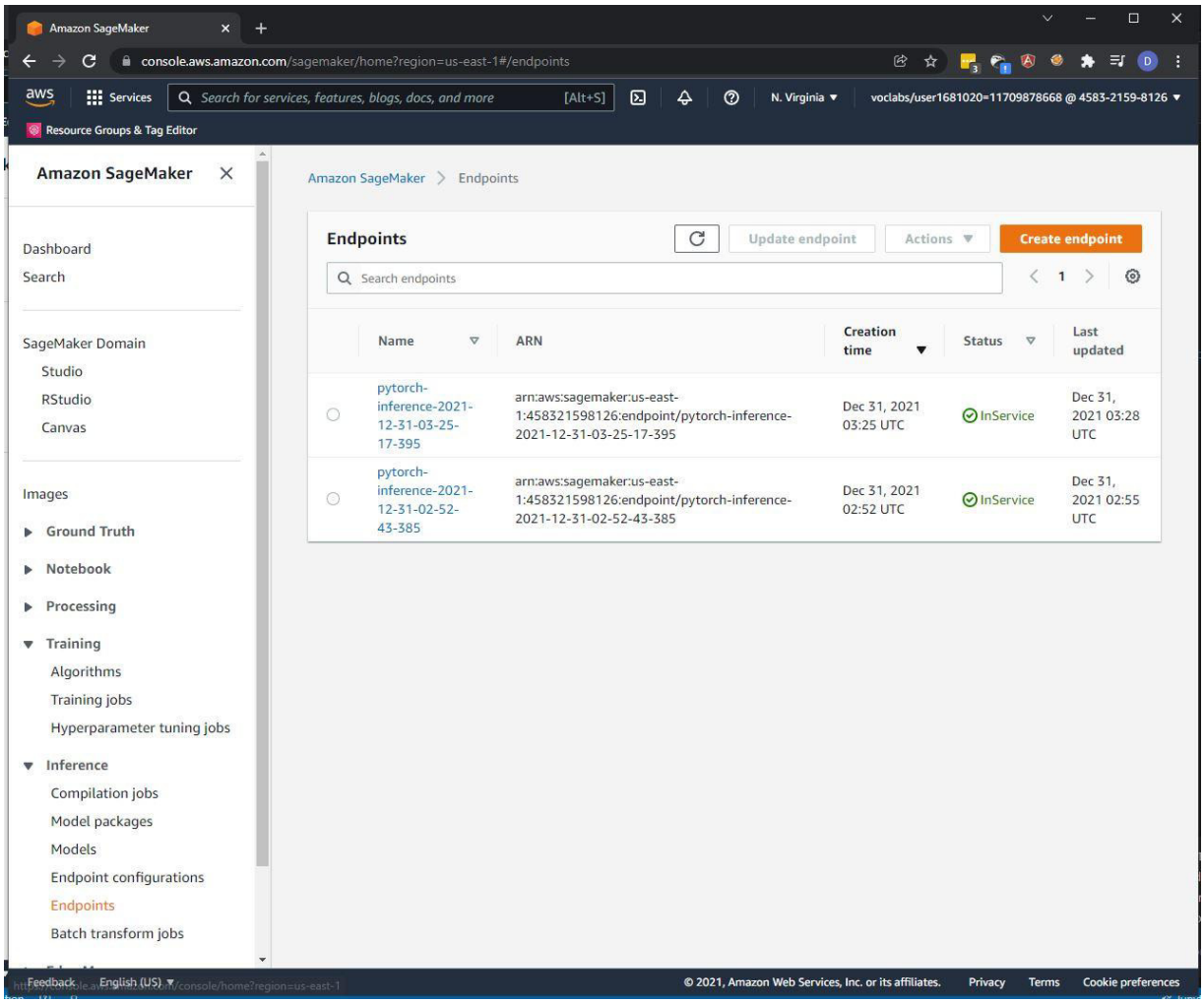
Additionally, I chose to use `ml.m5.2xlarge` for both tuning and training because it has greater processing power so that the tuning job and training jobs could be completed more quickly.

To speed up tuning and ensure that better hyperparameters were chosen, I increased `max_jobs` to 10 for tuning, `max_parallel_jobs` to 3, and early stopping type to "Auto."

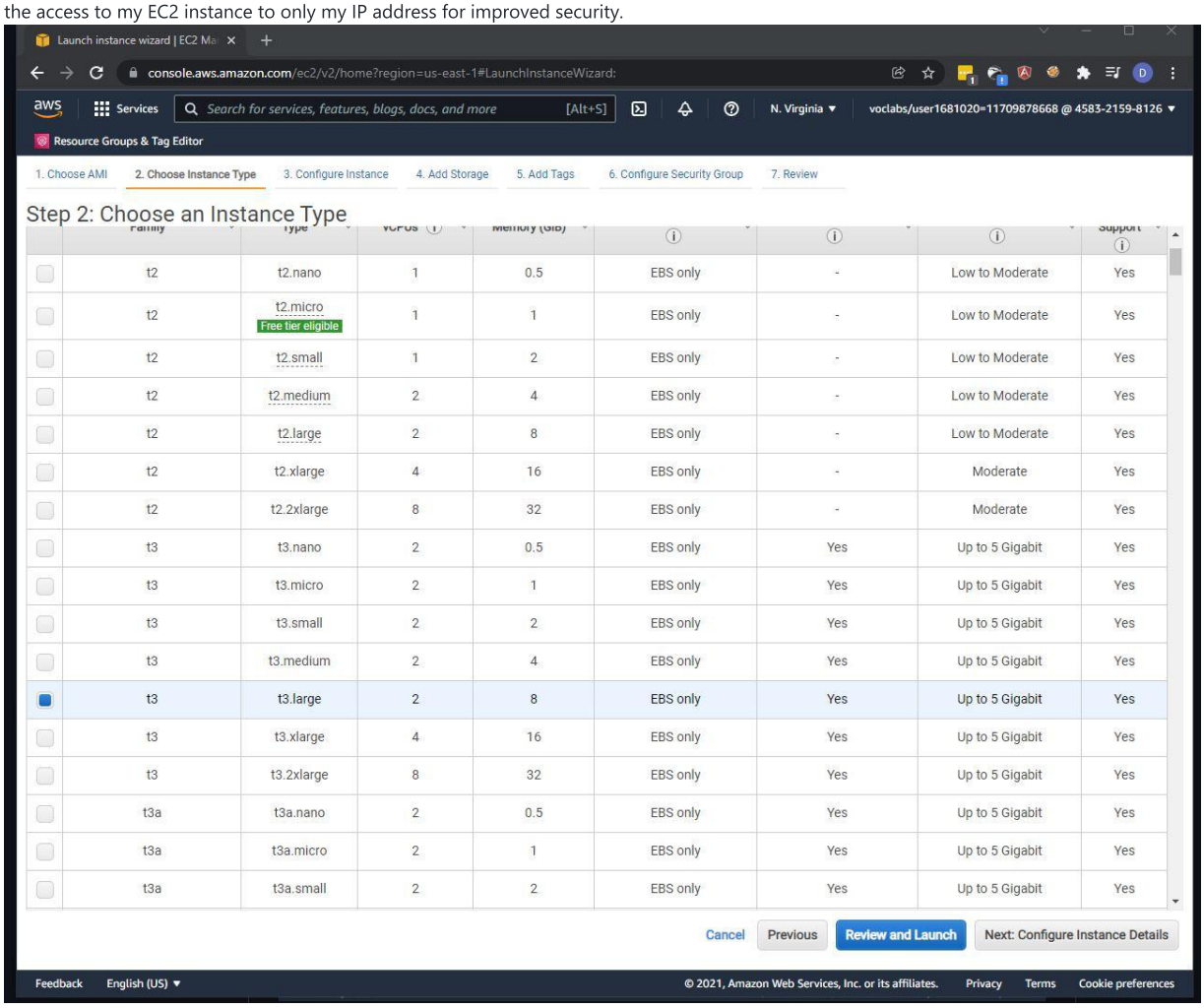


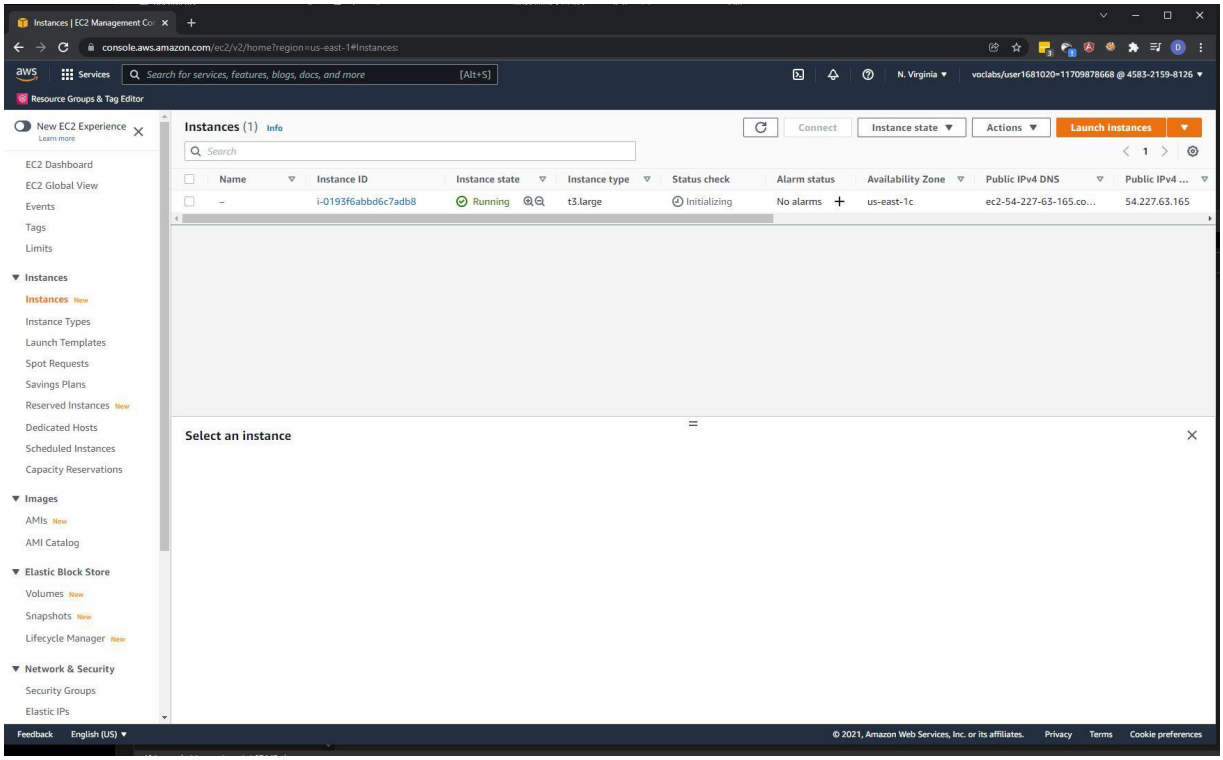
The following is my trained endpoint as per the image below

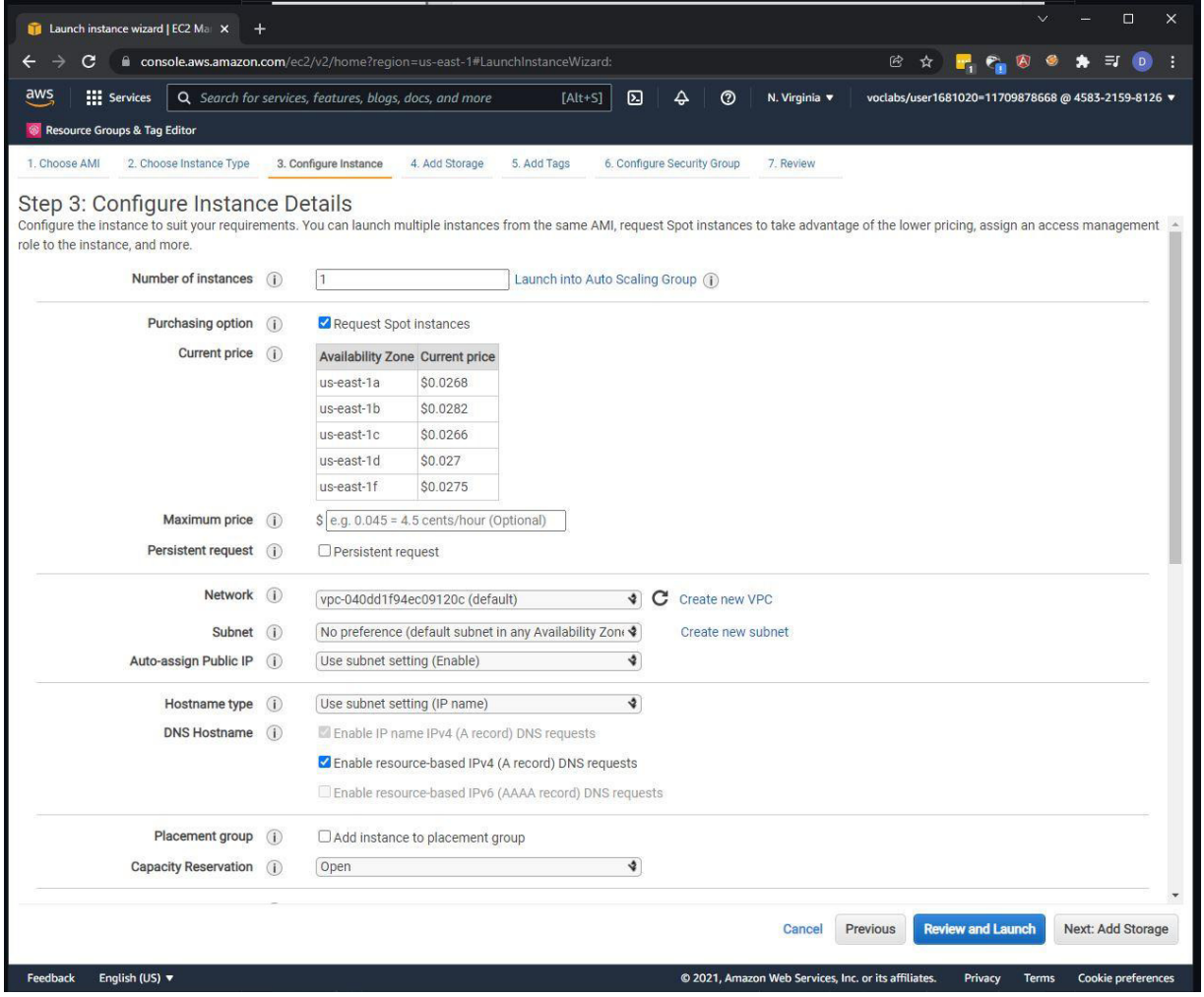
- Single instance endpoint: `pytorch-inference-2021-12-31-02-52-43-385`
- Multi instance endpoint: `pytorch-inference-2021-12-31-03-25-17-395`

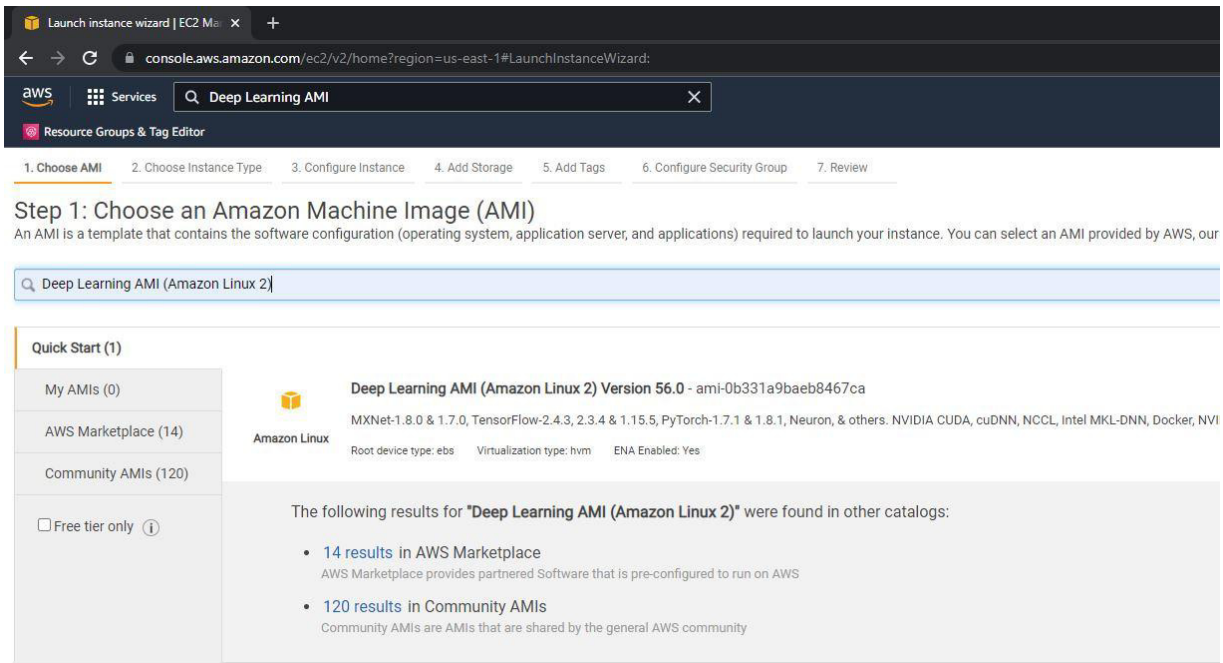


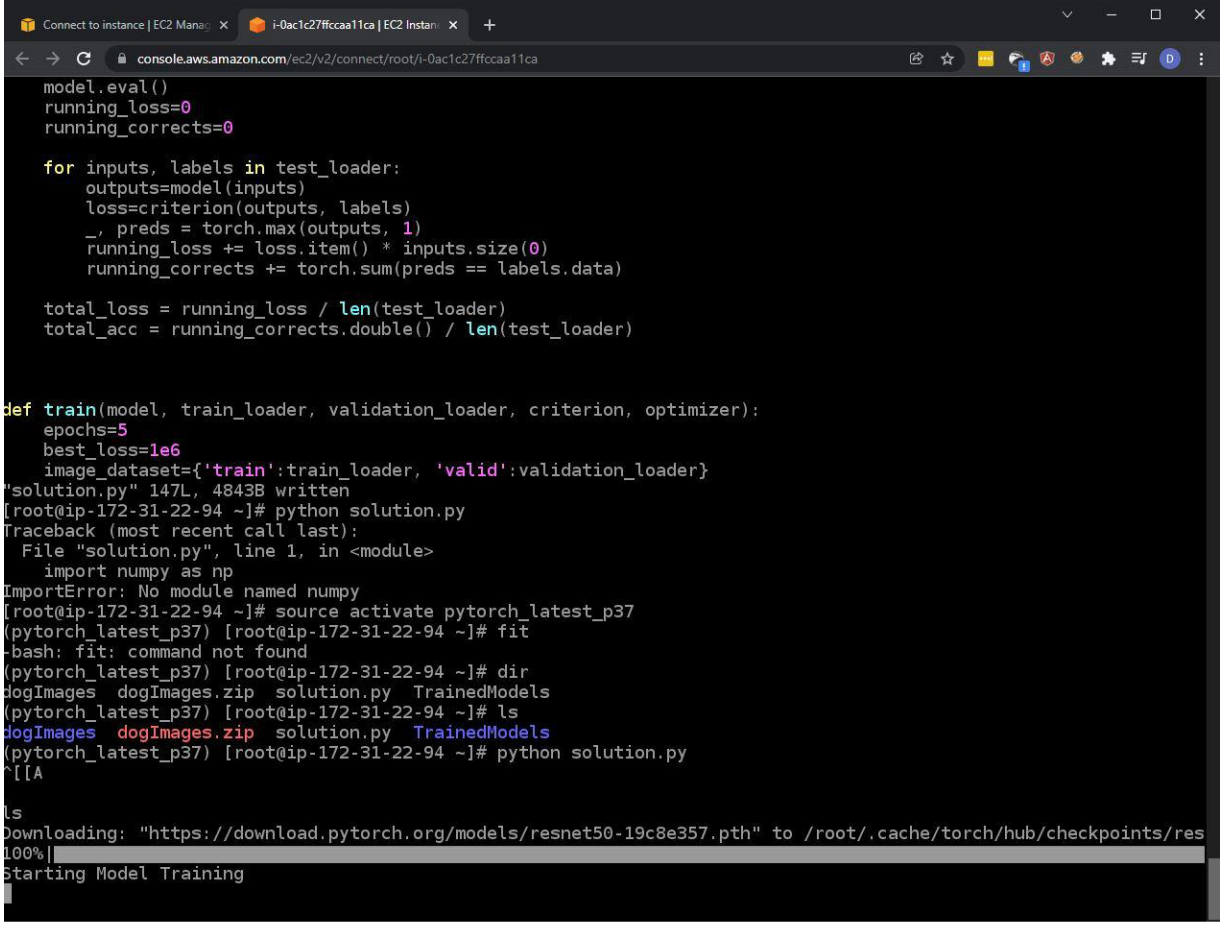
Here I have used the image `Deep Learning AMI (Amazon Linux 2) Version 56.0` and the instance type 't3.xlarge'. This seemed to me to be a nice balance of performance and affordability. Because we are running a spot instance EC2 for training, it is logical to choose a more powerful instance type than is required so that we are able to quickly train our model, and the cost for spot instance are quite small. I have also limited the access to my EC2 instance to only my IP address for improved security.



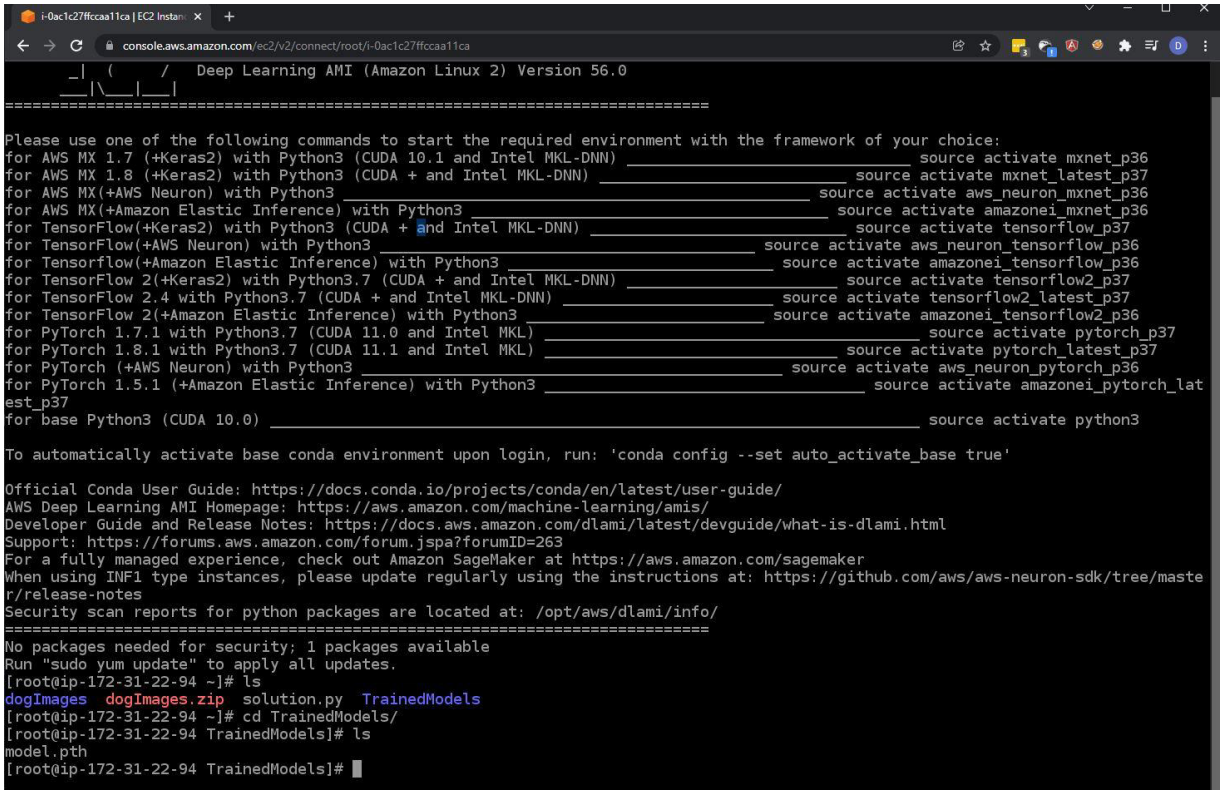



Launch instance wizard | EC2 Ma
console.aws.amazon.com/ec2/v2/home?region=us-east-1#LaunchInstanceWizard:
Services
Search for services, features, blogs, docs, and more [Alt+S]
N. Virginia
voclabs/user1681020=11709878668 @ 4583-2159-8126
Resource Groups & Tag Editor
1. Choose AMI
2. Choose Instance Type
3. Configure Instance
4. Add Storage
5. Add Tags
6. Configure Security Group
7. Review
Step 3: Configure Instance Details
Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.
Number of instances
1
Launch into Auto Scaling Group
Purchasing option
Request Spot instances
Current price
Availability Zone
Current price
us-east-1a
$0.0268
us-east-1b
$0.0282
us-east-1c
$0.0266
us-east-1d
$0.027
us-east-1f
$0.0275
Maximum price
$ e.g. 0.045 = 4.5 cents/hour (Optional)
Persistent request
Persistent request
Network
vpc-040dd1f94ec09120c (default)
Create new VPC
Subnet
No preference (default subnet in any Availability Zone
Create new subnet
Auto-assign Public IP
Use subnet setting (Enable)
Hostname type
Use subnet setting (IP name)
DNS Hostname
Enable IP name IPv4 (A record) DNS requests
Enable resource-based IPv4 (A record) DNS requests
Enable resource-based IPv6 (AAAA record) DNS requests
Placement group
Add instance to placement group
Capacity Reservation
Open
Cancel
Previous
Review and Launch
Next: Add Storage
Feedback
English (US)
© 2021, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Launch instance wizard | EC2 Ma
console.aws.amazon.com/ec2/v2/home?region=us-east-1#LaunchInstanceWizard:
Services
Deep Learning AMI
Resource Groups & Tag Editor
1. Choose AMI
2. Choose Instance Type
3. Configure Instance
4. Add Storage
5. Add Tags
6. Configure Security Group
7. Review
Step 1: Choose an Amazon Machine Image (AMI)
An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our
Deep Learning AMI (Amazon Linux 2)
Quick Start (1)
My AMIs (0)
AWS Marketplace (14)
Community AMIs (120)
Free tier only
Amazon Linux
Deep Learning AMI (Amazon Linux 2) Version 56.0 - ami-0b331a9baeb8467ca
MXNet-1.8.0 & 1.7.0, TensorFlow-2.4.3, 2.3.4 & 1.15.5, PyTorch-1.7.1 & 1.8.1, Neuron, & others. NVIDIA CUDA, cuDNN, NCCL, Intel MKL-DNN, Docker, NVI
Root device type: ebs
Virtualization type: hvm
ENA Enabled: Yes
The following results for "Deep Learning AMI (Amazon Linux 2)" were found in other catalogs:
14 results in AWS Marketplace
AWS Marketplace provides partnered Software that is pre-configured to run on AWS
120 results in Community AMIs
Community AMIs are AMIs that are shared by the general AWS community



screenshot showing the result of model.pth

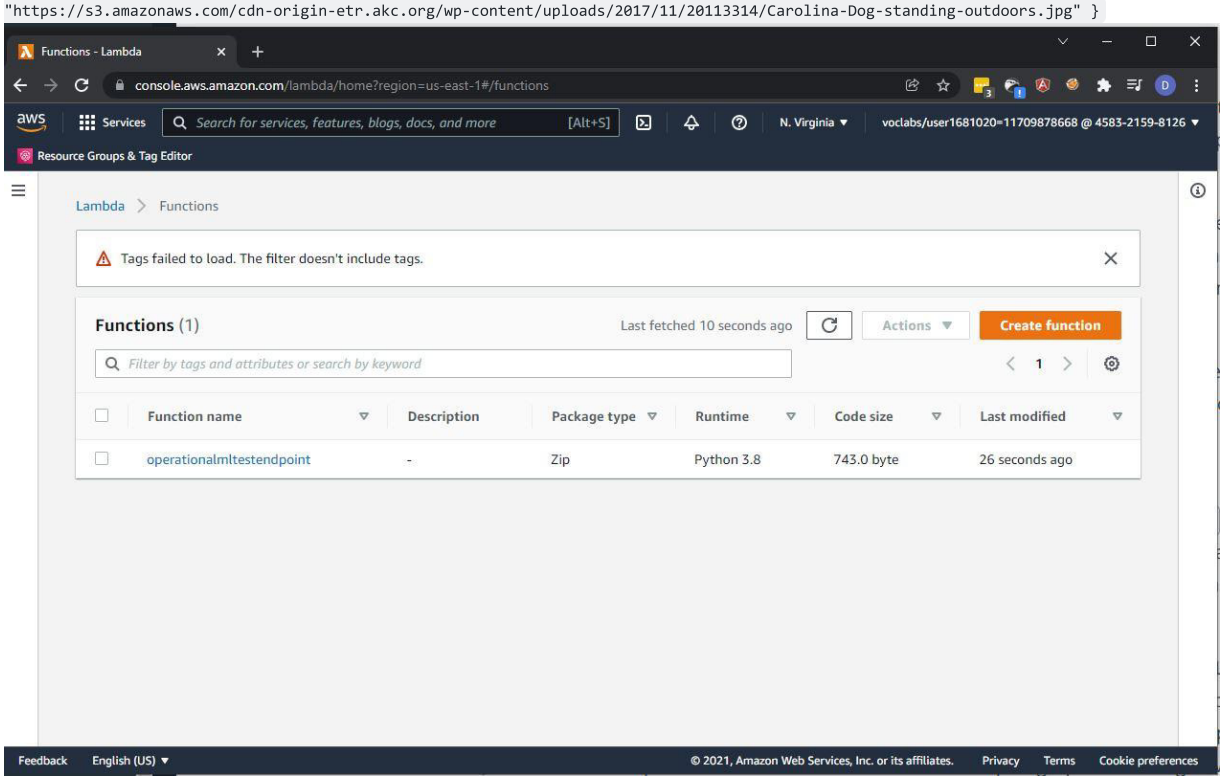


Differences between ec2train1.py and hpo.py

When comparing the EC2 code to Step 1. The code that we worked in Step 1 was different as the code was written to work with SageMaker, and there are some differences in the code that we used in ectrain1.py, The argument parsing at the main are only done in Sage maker to print the argument out to notebook, which we don't do when training in EC2. The modules that we used are only can be used in SageMaker. such as Logger info that are used in SageMaker. Model debug hook for cloudwatch is another example as well that not in this example. I do also notice that the ec2 training using testdata.

Step 3 Lambda functions

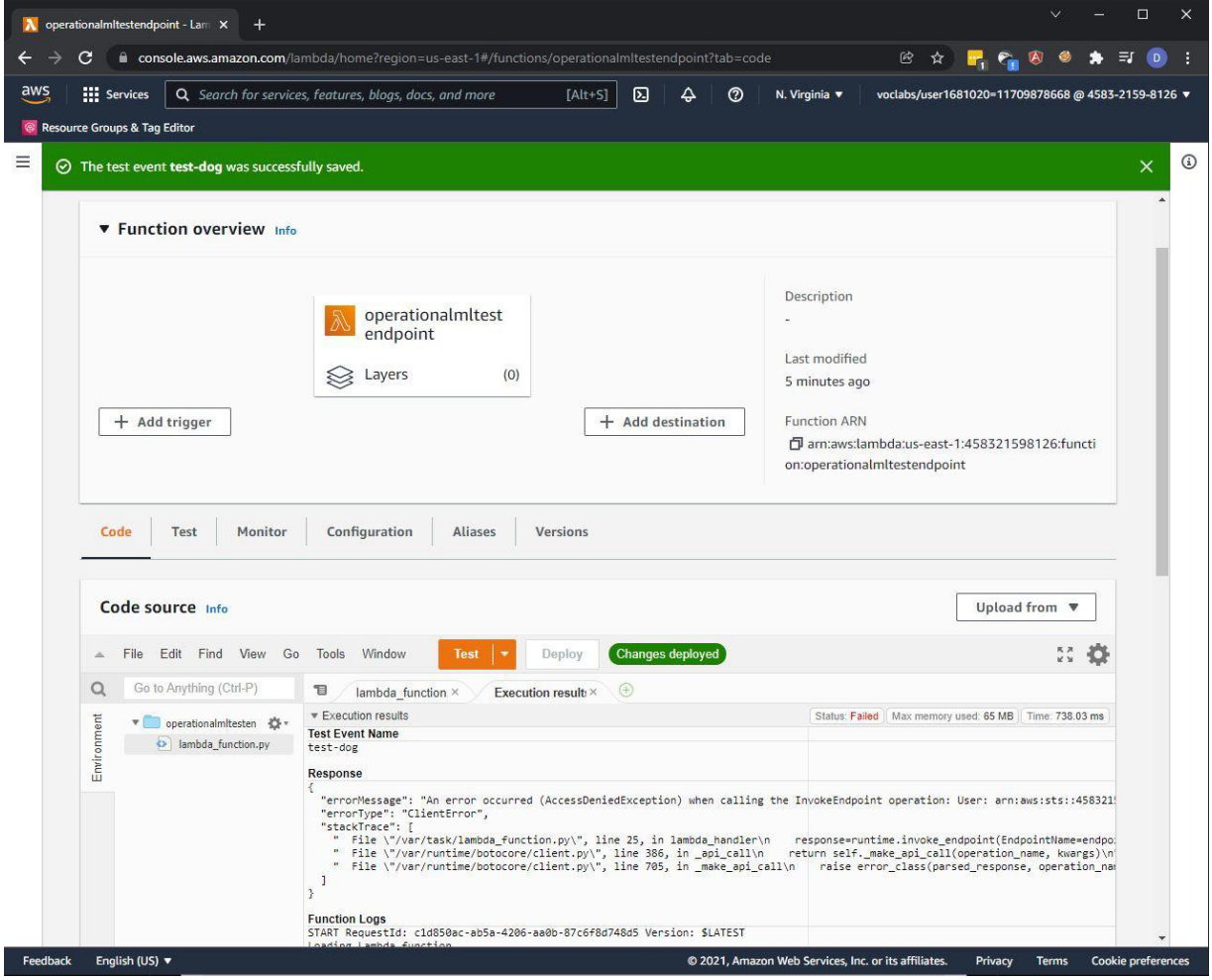
I have created the lambda function to the first endpoint I created to test the single instance trained endpoint. We are provided lamdafunction.py here and I have changed added the approriate single instance endpoint name here `endpoint_Name='pytorch-inference-2021-12-31-02-52-43-385'` Since we are provided as a lamdafunction.py file, I have decided to just upload it as a zip. This lambda function essentially just invoke our SageMaker endpoint using the boto3 API from SageMaker. The response from the endpoint that was invoked, will then be read and be utf-8 decoded. The results are then converted to a JSON and return on the Body element. I also noticed that the runtime here is also declared twice. The lambda function take in the following Test Json as the input which is the url of an image of a dog. `{ "url": "https://s3.amazonaws.com/cdn-origin-etr.akc.org/wp-content/uploads/2017/11/20113314/Carolina-Dog-standing-outdoors.jpg" }`

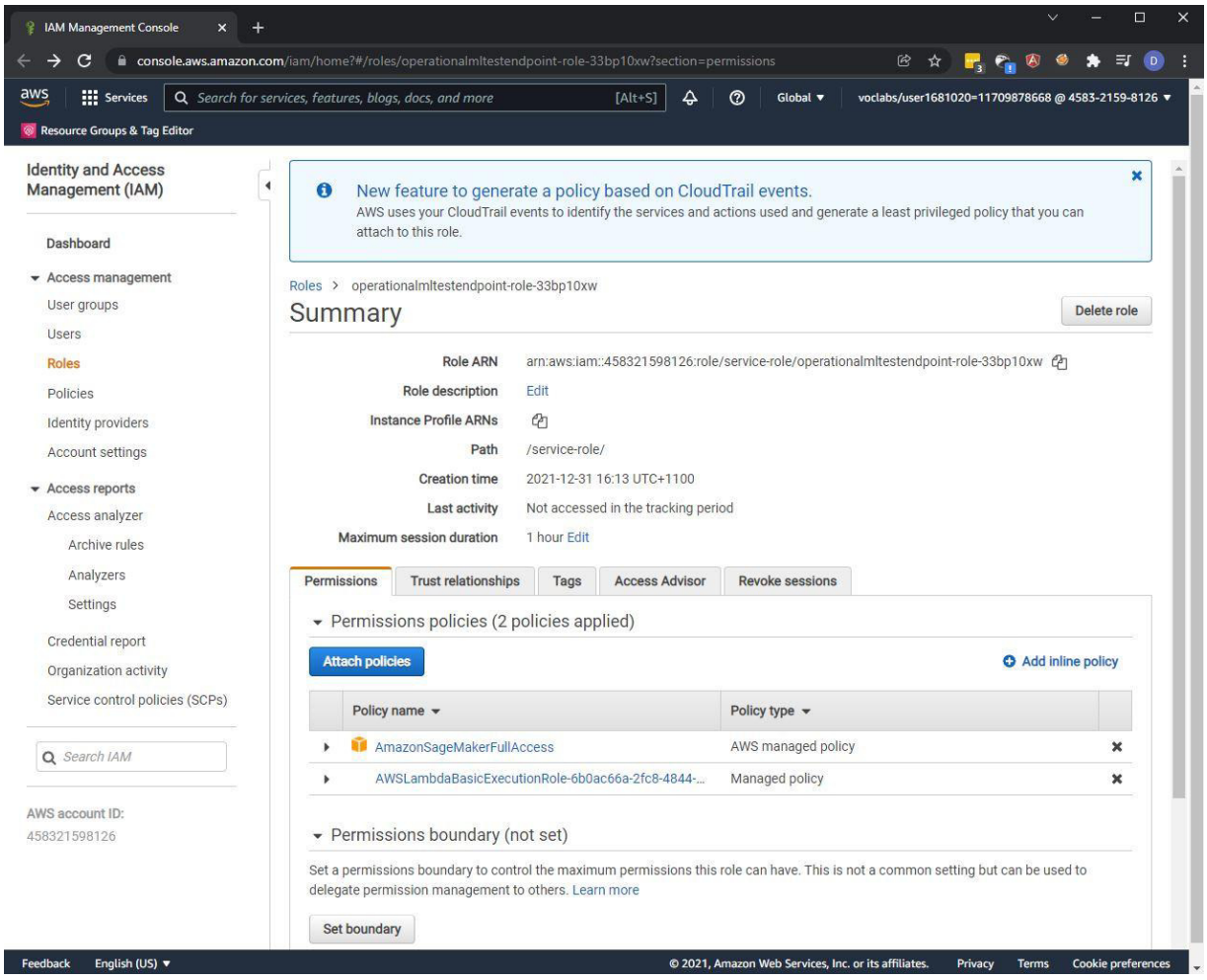


Step 4 Security & Testing

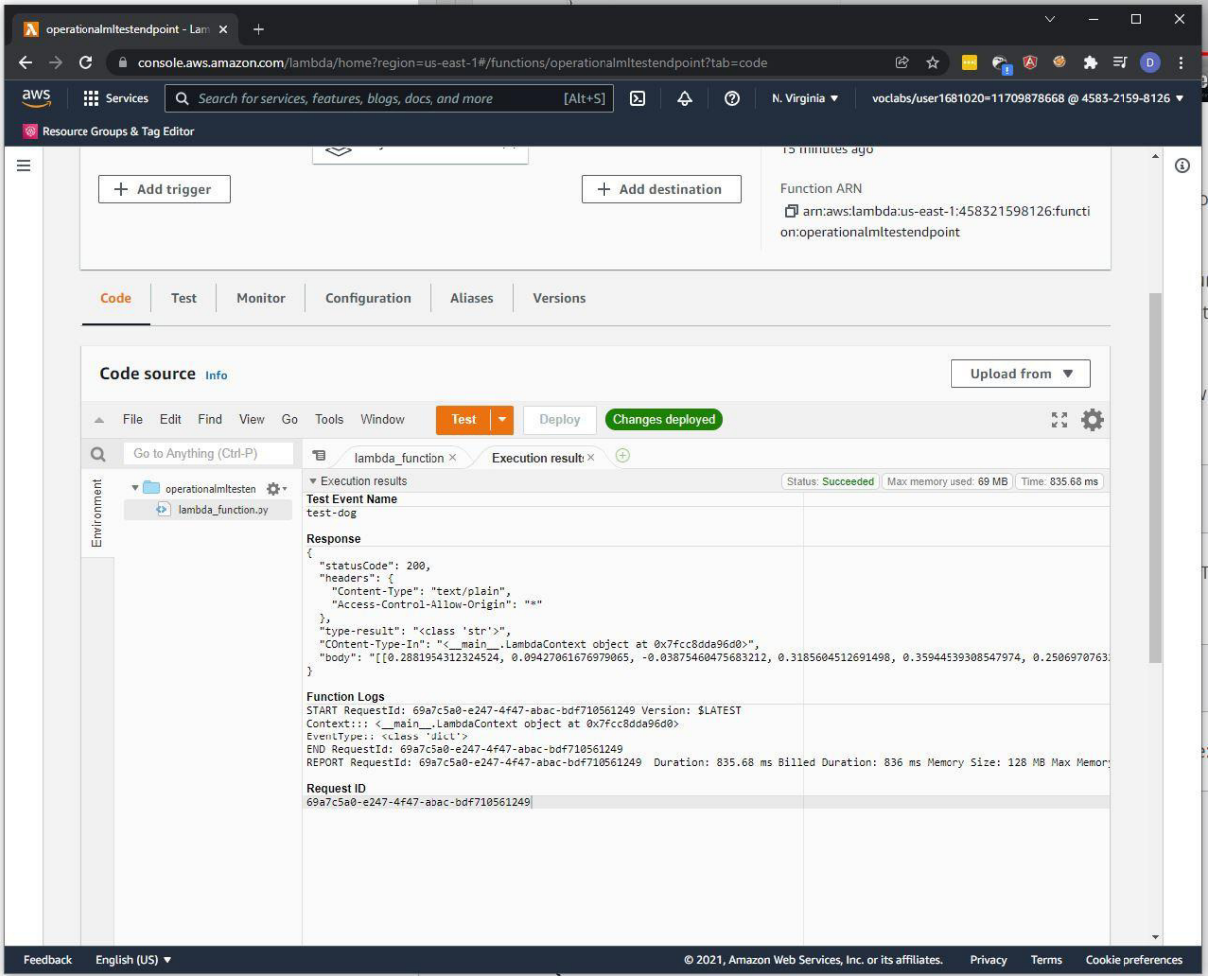
In any Role-based access control, permission should always be given the least privilege, but in this example here, we have given the full access to the SageMaker. We could identify which service that the endpoint is using and we can just give that particular access. We could also introduce Session Authentication on the lambda function which only allow authenticated users to call the endpoints. We may also need to review the roles to make sure that all the roles that we are using are current and not inactive, to reduce the risk that it might belong to previous colleagues. The root users that we used in this example also didn't utilize MFA as per the warning that we got on the IAM dashboard.

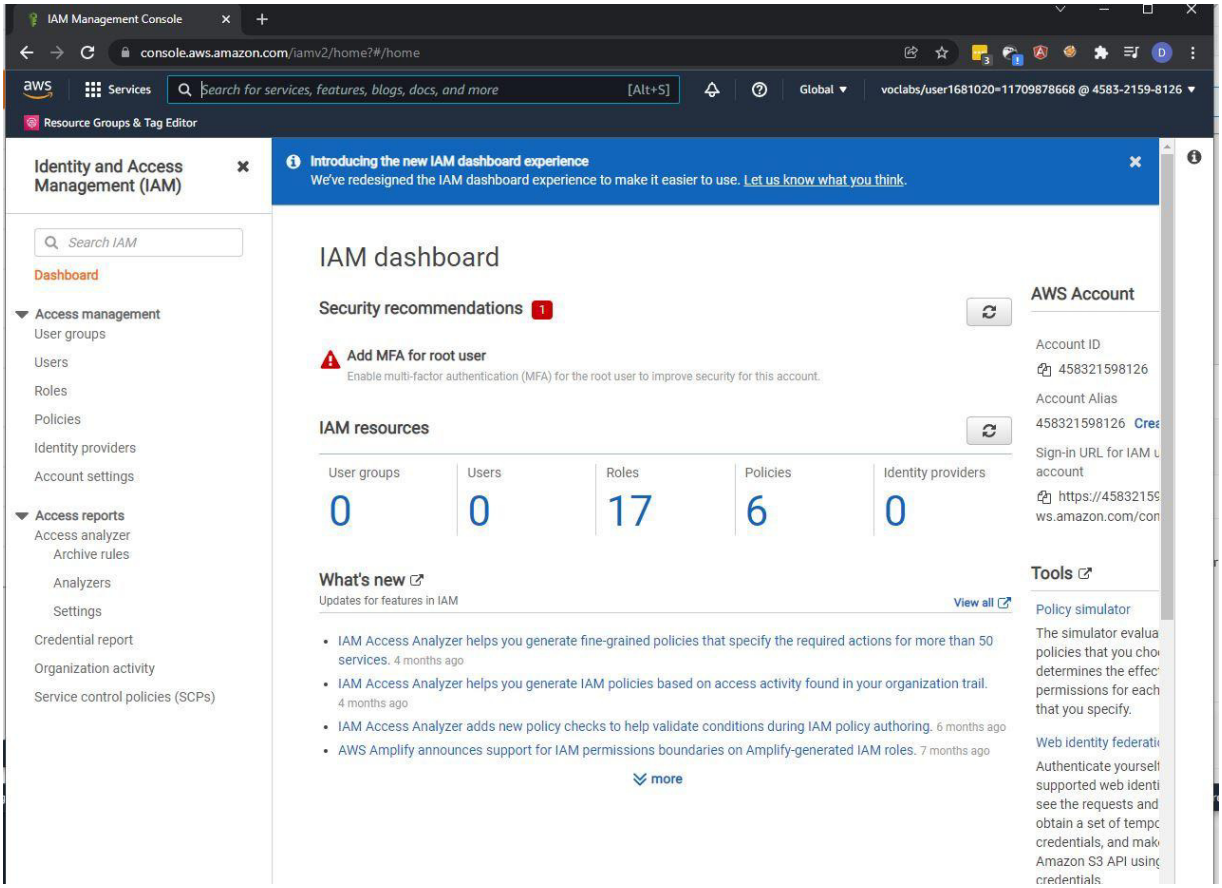
I got the following output before adding full access to SageMaker to the role running the lambda functions





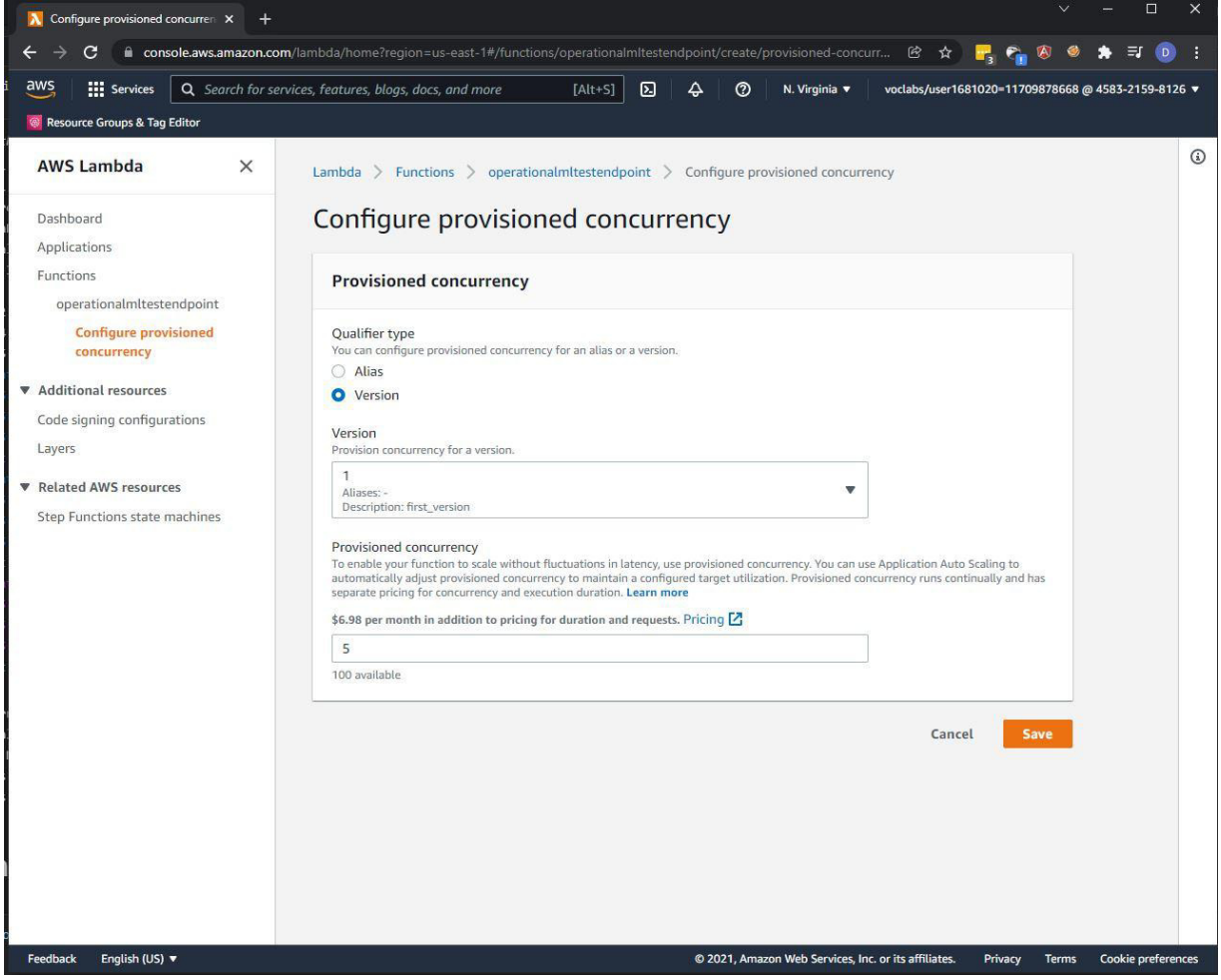
I got the following output from a test after attaching full access role to SageMaker on the role running the lambda function:

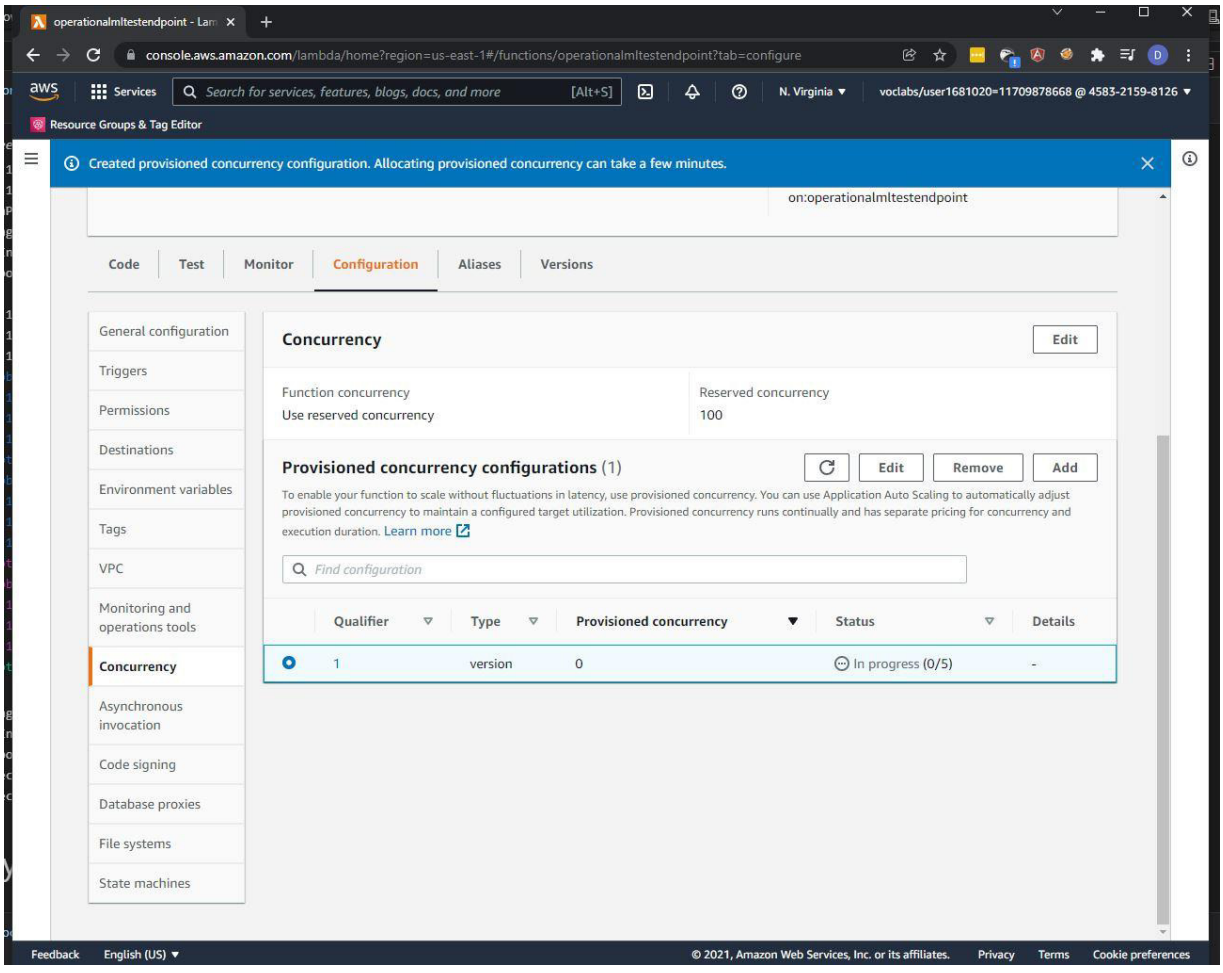


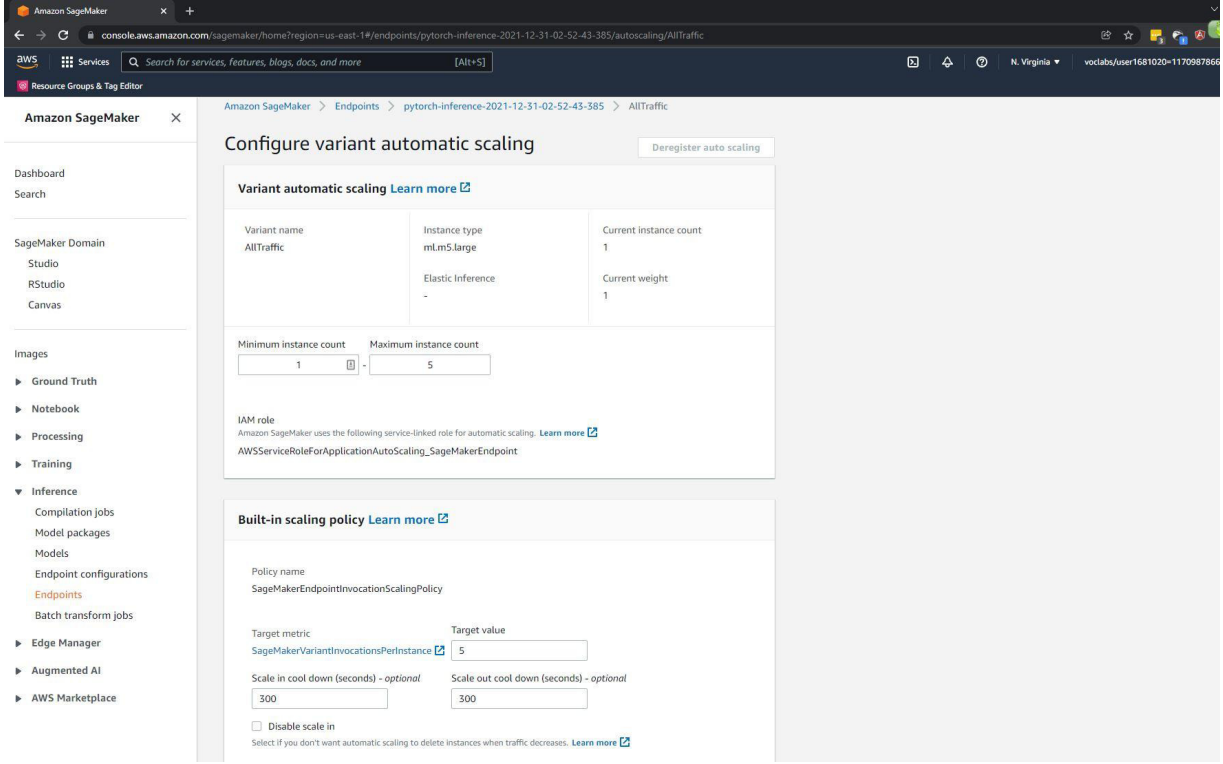


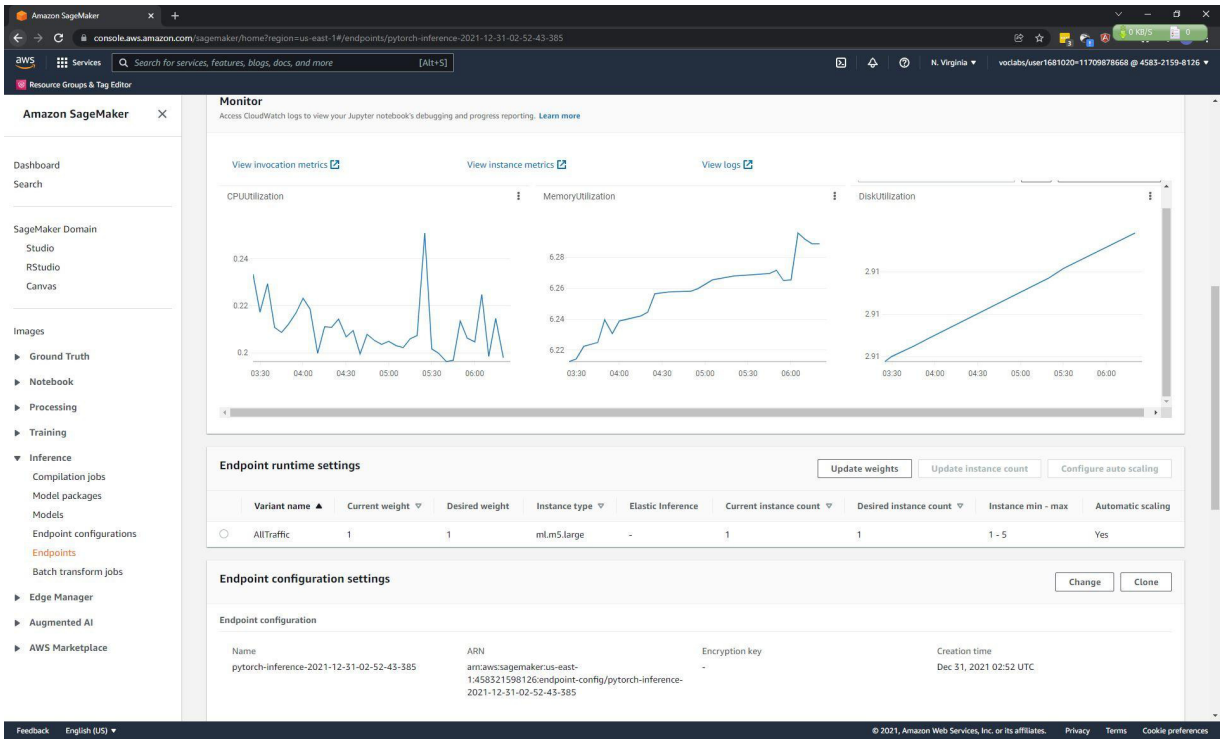
Step 5 Concurrency and auto-scaling

I chose reserved concurrency over shared concurrency because it is free and satisfies our current needs. Furthermore, we are unlikely to handle more than a few requests per endpoint instance at any given time, as this would indicate that the latter is overloaded, so having this value be a multiple of the number of endpoint instances makes sense, so I chose 100, allowing for 20 requests per endpoint instance. I decided to let the endpoint scale out to between one and five instances. It should be enough for our use in this case for this project as ecause each prediction only take roughly 0.835 seconds to run.


Configure provisioned concurren
console.aws.amazon.com/lambda/home?region=us-east-1#/functions/operationalmltestendpoint/create/provisioned-concurr...
Services
Search for services, features, blogs, docs, and more [Alt+S]
N. Virginia
voclabs/user1681020=11709878668 @ 4583-2159-8126
Resource Groups & Tag Editor
AWS Lambda
Dashboard
Applications
Functions
operationalmltestendpoint
Configure provisioned concurrency
Additional resources
Code signing configurations
Layers
Related AWS resources
Step Functions state machines
Lambda > Functions > operationalmltestendpoint > Configure provisioned concurrency
Configure provisioned concurrency
Provisioned concurrency
Qualifier type
You can configure provisioned concurrency for an alias or a version.
Alias
Version
Version
Provision concurrency for a version.
1
Aliases: -
Description: first_version
Provisioned concurrency
To enable your function to scale without fluctuations in latency, use provisioned concurrency. You can use Application Auto Scaling to automatically adjust provisioned concurrency to maintain a configured target utilization. Provisioned concurrency runs continually and has separate pricing for concurrency and execution duration. Learn more
$6.98 per month in addition to pricing for duration and requests. Pricing
5
100 available
Cancel
Save
Feedback
English (US)
© 2021, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


operationalmltestendpoint - Lam
console.aws.amazon.com/lambda/home?region=us-east-1#/functions/operationalmltestendpoint?tab=configure
Services
Search for services, features, blogs, docs, and more [Alt+S]
N. Virginia
voclabs/user1681020=11709878668 @ 4583-2159-8126
Resource Groups & Tag Editor
Created provisioned concurrency configuration. Allocating provisioned concurrency can take a few minutes.
on:operationalmltestendpoint
Code
Test
Monitor
Configuration
Aliases
Versions
General configuration
Triggers
Permissions
Destinations
Environment variables
Tags
VPC
Monitoring and operations tools
Concurrency
Asynchronous invocation
Code signing
Database proxies
File systems
State machines
Concurrency
Edit
Function concurrency
Use reserved concurrency
Reserved concurrency
100
Provisioned concurrency configurations (1)
Edit
Remove
Add
To enable your function to scale without fluctuations in latency, use provisioned concurrency. You can use Application Auto Scaling to automatically adjust provisioned concurrency to maintain a configured target utilization. Provisioned concurrency runs continually and has separate pricing for concurrency and execution duration. Learn more
Find configuration
Qualifier
Type
Provisioned concurrency
Status
Details
1
version
0
In progress (0/5)
-
Feedback
English (US)
© 2021, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Amazon SageMaker
console.aws.amazon.com/sagemaker/home?region=us-east-1#/endpoints/pytorch-inference-2021-12-31-02-52-43-385/autoscaling/AllTraffic
Search for services, features, blogs, docs, and more [Alt+S]
N. Virginia
Resource Groups & Tag Editor
Amazon SageMaker
Dashboard
Search
SageMaker Domain
Studio
RStudio
Canvas
Images
Ground Truth
Notebook
Processing
Training
Inference
Compilation jobs
Model packages
Models
Endpoint configurations
Endpoints
Batch transform jobs
Edge Manager
Augmented AI
AWS Marketplace
Amazon SageMaker > Endpoints > pytorch-inference-2021-12-31-02-52-43-385 > AllTraffic
Configure variant automatic scaling
Deregister auto scaling
Variant automatic scaling Learn more
Variant name
AllTraffic
Instance type
ml.m5.large
Elastic Inference
-
Current instance count
1
Current weight
1
Minimum instance count
1
Maximum instance count
5
IAM role
Amazon SageMaker uses the following service-linked role for automatic scaling. Learn more
AWSServiceRoleForApplicationAutoScaling_SageMakerEndpoint
Built-in scaling policy Learn more
Policy name
SageMakerEndpointInvocationScalingPolicy
Target metric
SageMakerVariantInvocationsPerInstance
Target value
5
Scale in cool down (seconds) - optional
300
Scale out cool down (seconds) - optional
300
Disable scale in
Select if you don't want automatic scaling to delete instances when traffic decreases. Learn more


Amazon SageMaker
console.aws.amazon.com/sagemaker/home?region=us-east-1#/endpoints/pytorch-inference-2021-12-31-02-52-43-385
Search for services, features, blogs, docs, and more [Alt+S]
N. Virginia
voclabs/user1681020=1170987866 @ 4583-2159-8126
Resource Groups & Tag Editor
Amazon SageMaker
Dashboard
Search
SageMaker Domain
Studio
RStudio
Canvas
Images
Ground Truth
Notebook
Processing
Training
Inference
Compilation jobs
Model packages
Models
Endpoint configurations
Endpoints
Batch transform jobs
Edge Manager
Augmented AI
AWS Marketplace
Monitor
Access CloudWatch logs to view your Jupyter notebook's debugging and progress reporting. Learn more
View invocation metrics
View instance metrics
View logs
CPUUtilization
MemoryUtilization
DiskUtilization
Endpoint runtime settings
Update weights
Update instance count
Configure auto scaling
Variant name
Current weight
Desired weight
Instance type
Elastic Inference
Current instance count
Desired instance count
Instance min - max
Automatic scaling
AllTraffic
1
1
ml.m5.large
-
1
1
1 - 5
Yes
Endpoint configuration settings
Change
Clone
Endpoint configuration
Name
pytorch-inference-2021-12-31-02-52-43-385
ARN
arn:aws:sagemaker:us-east-1:458321598126:endpoint-config/pytorch-inference-2021-12-31-02-52-43-385
Encryption key
-
Creation time
Dec 31, 2021 02:52 UTC
Feedback
English (US)
© 2021, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences