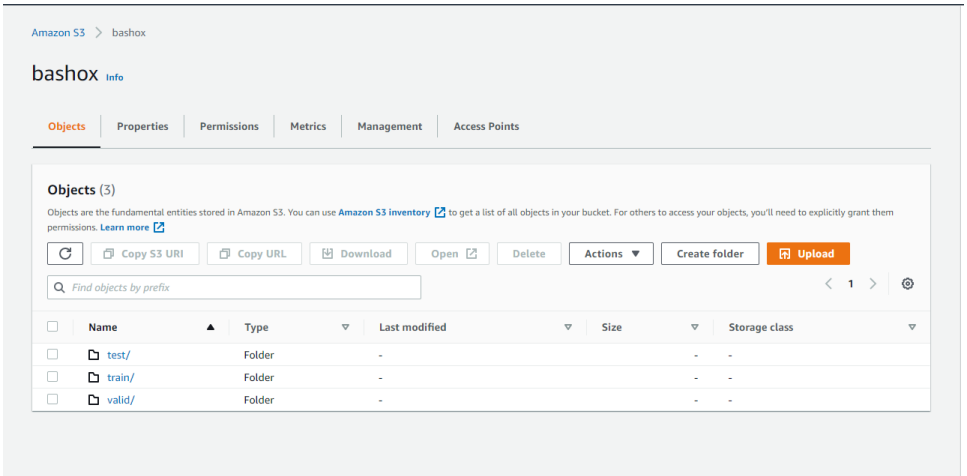
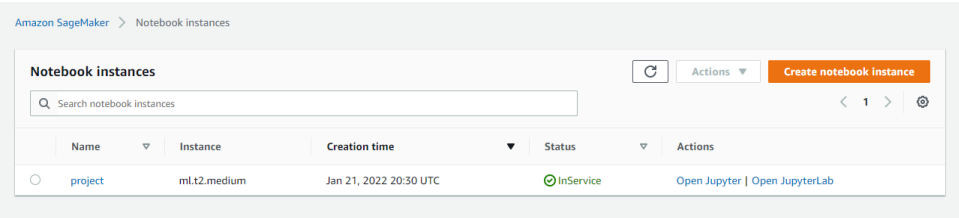
Notebook Instance Setup

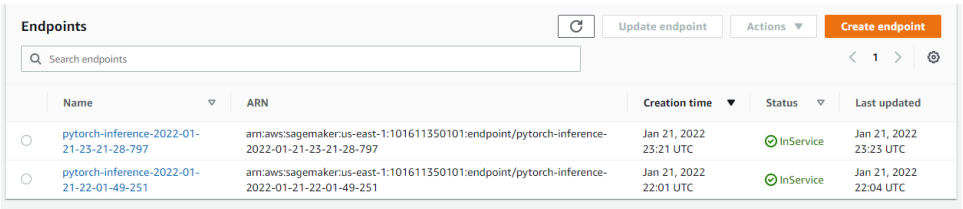
ml.t2.medium being the smallest SageMaker Notebook Instance, was the perfect instance to save costs for projects that do not require a Large CPU or RAM.

I uploaded my data to s3 and which sagemaker has access to





For the training jobs, since deep learning model trainings often require a large compute power, ml.m5.2xlarge was the perfect instance for this project. To speed up the training, I also increased the instance count from 1 to 7 to enable Multi Instance Training. Things like raising the max_job, max_parallel_jobs could significantly increase the training speed. I deployed two endpoints for the Single Instance Trained Model and the Multi Instance Trained Model as shown below.

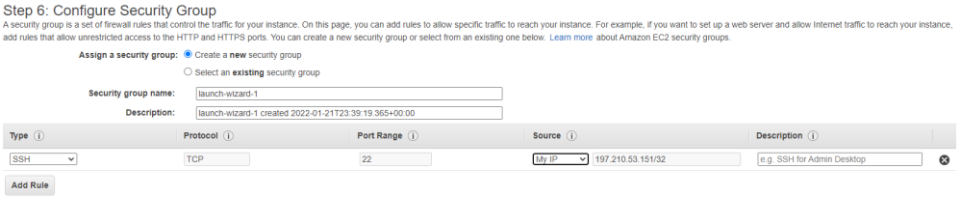


The first one being the Multi Instance Trained Model, while the second is the Single Instance Trained Model.

Training on EC2

Another option to save costs is to train your models in an EC2 instance. I used t3.xlarge instance with Deep Learning AMI (Amazon Linux 2) Version 54.0 perfect for this project without needing to install additional dependencies for deep learning and affordable also. Since the training will be in the EC2 instance, we do need a little bit of compute power and it's safe to say t3.xlarge performed just great optimizing for both costs and compute power. I take security so seriously

which is why I limit access to my EC2 to only my IP address.



Differences between ec2train.py and hpo(1).py

ec2train.py does not support Multi Instance Training unlike hpo(1).py which is used as an argument in train_and_deploy-solution.ipynb. Talking about argument, ec2train.py cant even handle argument parsing unlike hpo(1).py which can parse arguments.

Lambda Function

This is my favourite part mainly because of its serverless functionality aligns with my focus on keeping the costs low. I generated Lambda function to test the deployed endpoint. As I said, I am very diligent in security, I made sure the lambda function only has access to sage Maker and nothing else by adding necessary permission to its attached role. I then invoked the lambda and got the following results:

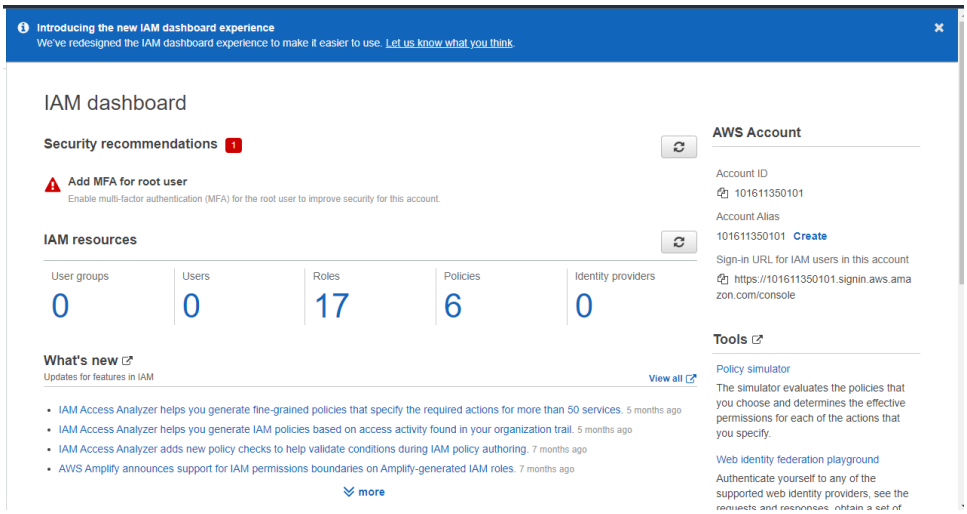
{ "statusCode": 200, "headers": { "Content-Type": "text/plain", "Access-Control-Allow-Origin": "*" }, "type-result": "<class 'str'>", "COntent-Type-In": "<__main__.LambdaContext object at 0x7f7f23782e80>", "body": "[[-9.833578109741211, -2.3149001598358154, -4.496285915374756, -2.459660768508911, -3.2292635440826416, -7.208012104034424, -2.019676923751831, -4.80747652053833, -7.580871105194092, 0.1604955494403839, -0.9565751552581787, -3.9131810665130615, -3.6020150184631348, 0.05120508745312691, -4.187273025512695, -2.8582608699798584, -5.9626336097717285, -4.536521911621094, -5.346053123474121, 0.8844369649887085, -4.171061038970947, -1.9665920734405518, -8.535749435424805, -7.713301181793213, -6.517572402954102, -11.525176048278809, -1.6054246425628662, -2.6769134998321533, -6.043071269989014, -4.014992713928223, -1.5675303936004639, -5.836017608642578, -9.598838806152344, -3.9479377269744873, -7.269865989685059, -8.273256301879883, -7.2418389320373535, -6.530744552612305, -2.6606929302215576, -4.69330358505249, -4.120052337646484, -5.510358810424805, -0.34800487756729126, -4.418677806854248, -1.6251585483551025, -8.261250495910645, -3.4547064304351807, -3.625526189804077, -2.2067673206329346, -5.123303413391113, -4.32491397857666, -9.562714576721191, -10.10139274597168, -2.621943235397339, -7.87992525100708, -2.6131465435028076, -2.555243968963623, -8.795278549194336, -2.1555397510528564, -5.046584606170654, -8.843852996826172, -4.459414958953857, -7.6189351081848145, -8.326446533203125, -4.4117021560668945, -8.239740371704102, -2.5183329582214355, -5.56660270690918, -5.840574741363525, -0.9575398564338684, -0.9576200246810913, -5.738442897796631, -7.9645609855651855, -7.754147529602051, -7.372220039367676, -2.7810957431793213, -7.490314483642578, -3.23928165435791, -5.329830169677734, -4.7164130210876465, -1.013601541519165, -8.067651748657227, -0.8396211266517639, -2.3040759563446045, -6.821334362030029, -4.568439483642578, -1.9165948629379272, -6.500880241394043, -2.1579692363739014, -2.284560441970825, -8.265277862548828, -5.20717191696167, -8.007201194763184, -6.459601402282715, -5.552811622619629, -4.417727470397949, -4.309690952301025, -4.329131126403809, -5.873167514801025, -5.903502941131592, -9.697685241699219, -2.39001202583313, -3.9008214473724365, -7.240103244781494, -7.310409069061279, -8.4434175491333, -5.835858345031738, -1.4929323196411133, -5.384642601013184, -2.533282995223999, -3.6708028316497803, -2.3309476375579834, -11.313907623291016, -7.884079456329346, -7.088305473327637, -2.394468307495117, -4.874009132385254, -4.4106550216674805, -6.073166370391846, -1.498389482498169, -3.4654202461242676, -4.046548366546631, -

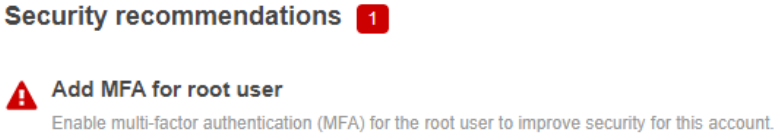
6.391988754272461, -4.826951026916504, -9.816082954406738, -7.140218734741211, -4.48201847076416, -3.733743190765381, -4.512071132659912, -7.819015026092529, -6.860950946807861, -1.9007254838943481, -5.910882949829102]]"}

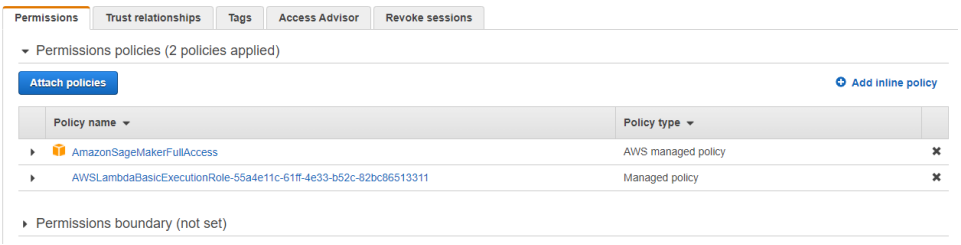
Security Concerns

I must say that security is the most important aspect of a production system and infrastructure, as such I must address the following security concerns.

- MFA not enabled
- Old Redundant Roles with permissions given
- Lambda having access to SageMaker instead of just being able to hit the endpoint. If not taken care of, these issues could lead to devastating results.







Role name	Trusted entities	Last activity
AWSServiceRoleForAWSCloud9	AWS Service: cloud9 (Service-Linked Role)	-
AWSServiceRoleForCloudWatchEvents	AWS Service: events (Service-Linked Role)	-
AWSServiceRoleForElastiCache	AWS Service: elasticache (Service-Linked Role)	-
AWSServiceRoleForGlobalAccelerator	AWS Service: globalaccelerator (Service-Linked Role)	-
AWSServiceRoleForOrganizations	AWS Service: organizations (Service-Linked Role)	-
AWSServiceRoleForSupport	AWS Service: support (Service-Linked Role)	-
AWSServiceRoleForTrustedAdvisor	AWS Service: trustedadvisor (Service-Linked Role)	-
EMR_AutoScaling_DefaultRole	AWS Service: application-autoscaling, and 1 more.	-
EMR_DefaultRole	AWS Service: elasticmapreduce	-
EMR_EC2_DefaultRole	AWS Service: ec2	-
project-role-8gm46y0q	AWS Service: lambda	-
robomaker_students	AWS Service: greengrass, and 3 more.	-
vocareum	Account: 137949000194	
voclabs	Account: 137949000194	
vocstartsoft	Account: 137949000194	
AWSServiceRoleForAmazonSageMakerNotebooks	AWS Service: sagemaker (Service-Linked Role)	6 hours ago
AmazonSageMaker-ExecutionRole-20220120T220588	AWS Service: sagemaker	16 minutes ago

Concurrency and auto-scaling

Concurrency is to allow lambda attend to multiple requests at a time. I used provisioned concurrency because it enables my lambda function to scale without fluctuations in latency. No one wants a slow response; slow responses chase away customers. I also auto scaled my endpoint to serve for any potential CPU bottlenecks.