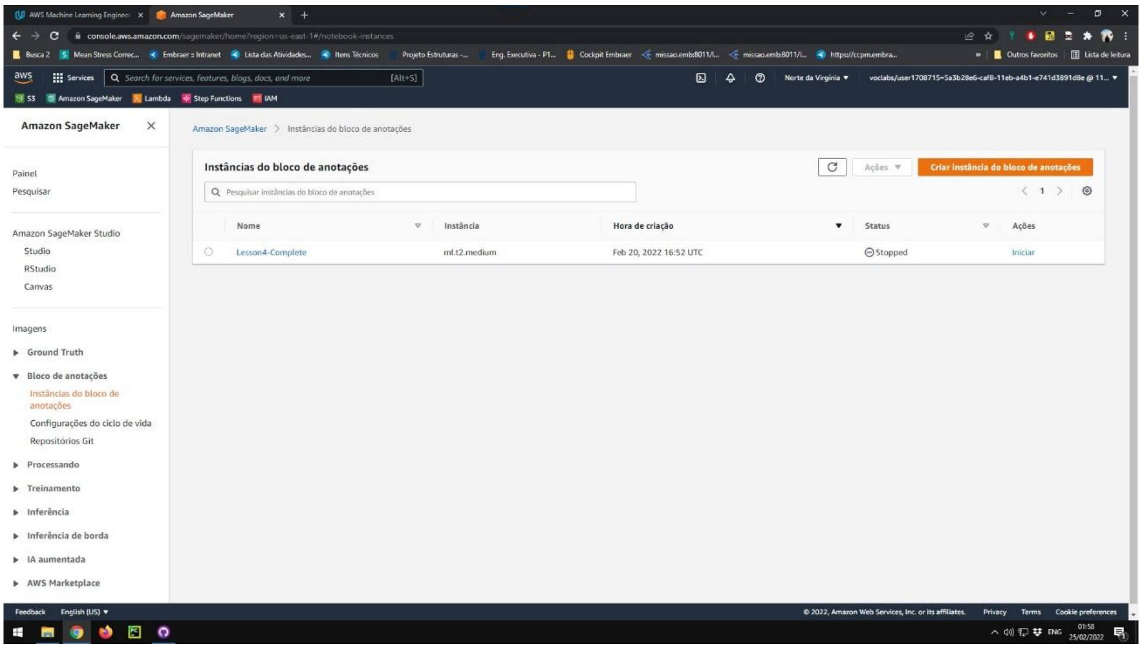
Step 1: Training and deployment on Sagemaker

Q: Justification of why I chose “ml.t2.medium” as instance type for my notebook:

A: I chose this type of instance because it was the cheapest and didn’t want extra costs to be incurred into my AWS account. I also leveraged my computing needs against this type of instance’s performance and got to the conclusion that it was enough for what I intended to do.


Amazon SageMaker
Instâncias do bloco de anotações
Ações
Criar instância do bloco de anotações
Pesquisar instâncias do bloco de anotações

Nome	Instância	Hora de criação	Status	Ações
Lesson4-Complete	ml.t2.medium	Feb 20, 2022 16:52 UTC	Stopped	Iniciar

Figure 1 - Instance used for Step 1 of the assignment

Step 2: EC2 Training

Q: Justification on “t2.large” instance choice:

A: This type of instance was chosen based on the instance used for model training during Step 1, where “ml.m5.xlarge” was used for training the model through a sagemaker notebook instance. The training instance used on the notebook portion of the assignment handled the workload in 25 minutes, approximately.

Since the “ml.m5.xlarge” instance has 4 vCPUs and memory of 16GB, I assumed that the “t2.large” EC2 instance would also handle model training well, since it has 2vCPUs and 8GB of memory. Additionally, I wanted to test if model training time was directly proportional to number of vCPUs and memory amount.

Q: What were the differences between hpo.py (sagemaker notebook script) and ec2train1.py (EC2 instance script) codes?

A: Main differences between codes and my thoughts about them:

- Logging is not done in EC2, while constantly used on notebook instance code: This is an assumption, but since I don't know if Cloudwatch is available for debugging scripts ran on EC2, maybe keeping a log of operations executed on EC2 scripts doesn't make sense. Another guess is that the logs were also kept for utilization of smdebug.
- The practice of running code under "if __name__ ==__main__:" is not used on the EC2 script: I am guessing that the EC2 instance does not run in a python environment were the variable __name__ is called __main__, making this practice useless on this environment.
- There is no argument parsing for passing variables on ec2train.py: Passing the variables through argument parsing as done in hpo.py is not necessary in EC2, where all the variables were defined within the script and used on future lines of code.

Step 3: Lambda Step Function

Q: Describe how the created step function works:

A: The lambda function aims to return the breed prediction of a dog image using the model previously trained on Step 1 that has been deployed to an endpoint.

This process starts with the lambda function receiving an image url formatted in JSON format (e.g. { "url": "https://s3.amazonaws.com/cdn-origin-etr.akc.org/wp-content/uploads/2017/11/20113314/Carolina-Dog-standing-outdoors.jpg"}) as the event variable of the lambda function (lambda_handler function).

Then, the endpoint is invoked and the image url is fed to the model deployed to the endpoint. The image in the url is analyzed and a response is provided. That response is a list with length equal to the number of classes in the trained model.

The index of the largest number in the list corresponds to the index of the class predicted by the model for the image provided.

Step 4: Security and Testing

Q: Add the result returned by the working Lambda function:

A: Lambda function output.

```
"body": "[[0.15138235688209534, 0.165018230676651, 0.14007285237312317,
0.28924357891082764, 0.3320692479610443, 0.4071626663208008,
0.12690217792987823, 0.33666786551475525, -0.3100398778915405, -
0.04771588370203972, -0.020740795880556107, 0.3124040961265564, -
0.05451264977455139, 0.2784910500049591, 0.4084441065788269, -
0.005373043939471245, 0.34847038984298706, -0.02555476874113083,
0.022741857916116714, 0.28085944056510925, 0.17307601869106293,
0.1358850598335266, 0.28126034140586853, -0.028355564922094345, -
0.3895307779312134, -0.20237508416175842, 0.18331791460514069, -
0.45990315079689026, 0.297315388917923, 0.10934335738420486, -
```

0.08636951446533203, 0.3866190016269684, 0.15717248618602753,
0.16832397878170013, -0.18320783972740173, 0.23751063644886017, -
0.028852656483650208, -0.0723142921924591, 0.36580270528793335, -
0.022773662582039833, 0.4073706567287445, 0.40086469054222107,
0.04849351570010185, 0.29215583205223083, 0.08492711931467056,
0.2383001148700714, 0.05554502457380295, -0.0482352077960968, -
0.16459481418132782, -0.17474055290222168, 0.3932984173297882, -
0.2773677408695221, 0.2403872311115265, -0.06117014214396477,
0.009298091754317284, 0.4906315803527832, 0.4742659032344818, -
0.0658886507153511, 0.11764644831418991, -0.11026854068040848,
0.30169677734375, 0.13449415564537048, 0.14040683209896088, -
0.16615954041481018, 0.06899295747280121, -0.5673354864120483, -
0.438524454832077, 0.11796832084655762, -0.09870289266109467, -
0.027472579851746 56, 0.3608511984348297, -0.00369902141392231, -
0.09508711844682693, -0.2639334499835968, -0.18538518249988556,
0.04904540628194809, -0.11347509920597076, -0.10154185444116592,
0.3932587504386902, -0.3900415599346161, 0.25380778312683105, -
0.042021386325359344, -0.20165517926216125, -0.03252109885215759, -
0.5284881591796875, 0.0035520626697689295, 0.3120252788066864, -
0.011330271139740944, -0.046712420880794525, 0.06932985782623291,
0.2933078706264496, -0.10667367279529572, -0.07396756857633591,
0.023675983771681786, 0.15712587535381317, -0.12941288948059082, -
0.05020171031355858, -0.0484069362282753, -0.15376555919647217, -
0.6210237145423889, 0.048973582684993744, -0.5821850299835205,
0.39307355880737305, -0.20692938566207886, -0.5753446817398071, -
0.07327507436275482, 0.06805866211652756, -0.6868038773536682, -
0.12232354283332825, -0.6999449729919434, -0.31589409708976746,
0.010117840021848679, -0.28487008810043335, -0.27170974016189575,
0.3666253089904785, -0.537486732006073, -0.0012416511308401823, -
0.08156853914260864, -0.3172677159309387, -0.4340532720088959, -
0.4466584622859955, -0.6491527557373047, -0.021946465596556664, -
0.051403921097517014, -0.2378981113433838, -0.5341098308563232, -
0.21094392240047455, -0.6052727103233337, 0.011881126090884209, -
0.12758386135101318, -0.5954464673995972, -0.5570266246795654, -
0.37411966919898987]]"

Q: Write about the security of your AWS workspace in your writeup. Are there any security vulnerabilities that need to be addressed? Think about some common security vulnerabilities.

A: The main vulnerability in my project is the role assigned to the Lambda function: “Full Access”. This breach could be used by someone who gains access to this function to extract sensible information stored in my AWS account or use my account to run its own projects, resulting on costs for services I have not used. In order to prevent that from happening, I should search for a role with fewer permissions that still allows my lambda function to invoke and query a deployed endpoint.

Step 5: Concurrency and Auto-Scaling

Q: When you set up concurrency and auto-scaling, you will make several choices about configuration. Write about the choices you made in the setup of concurrency and auto-scaling, and why you made each of those choices.

A:

- Choices regarding concurrency: I set apart 3 instances as reserved concurrency and selected a limit of 2 instances for provisioned concurrency for my lambda function. The number of instances set on provisioned concurrency was imagined to keep costs low while still maintaining the lambda function active. In the event of an overload of accesses to the lambda function, the number of 3 instances defined as reserved concurrency would be responsible for limiting the maximum cost.
- Choices regarding auto-scaling: My maximum instance count was defined as 3. This number was deemed enough to reduce endpoint downtime due to how fast it responded during tests with the lambda function. The cooldowns after instance reduction and expansion were both set to 30 seconds to provide the endpoint with fast scaling response in case of overload while still assuring fast downsize of instance counts after utilization peaks.