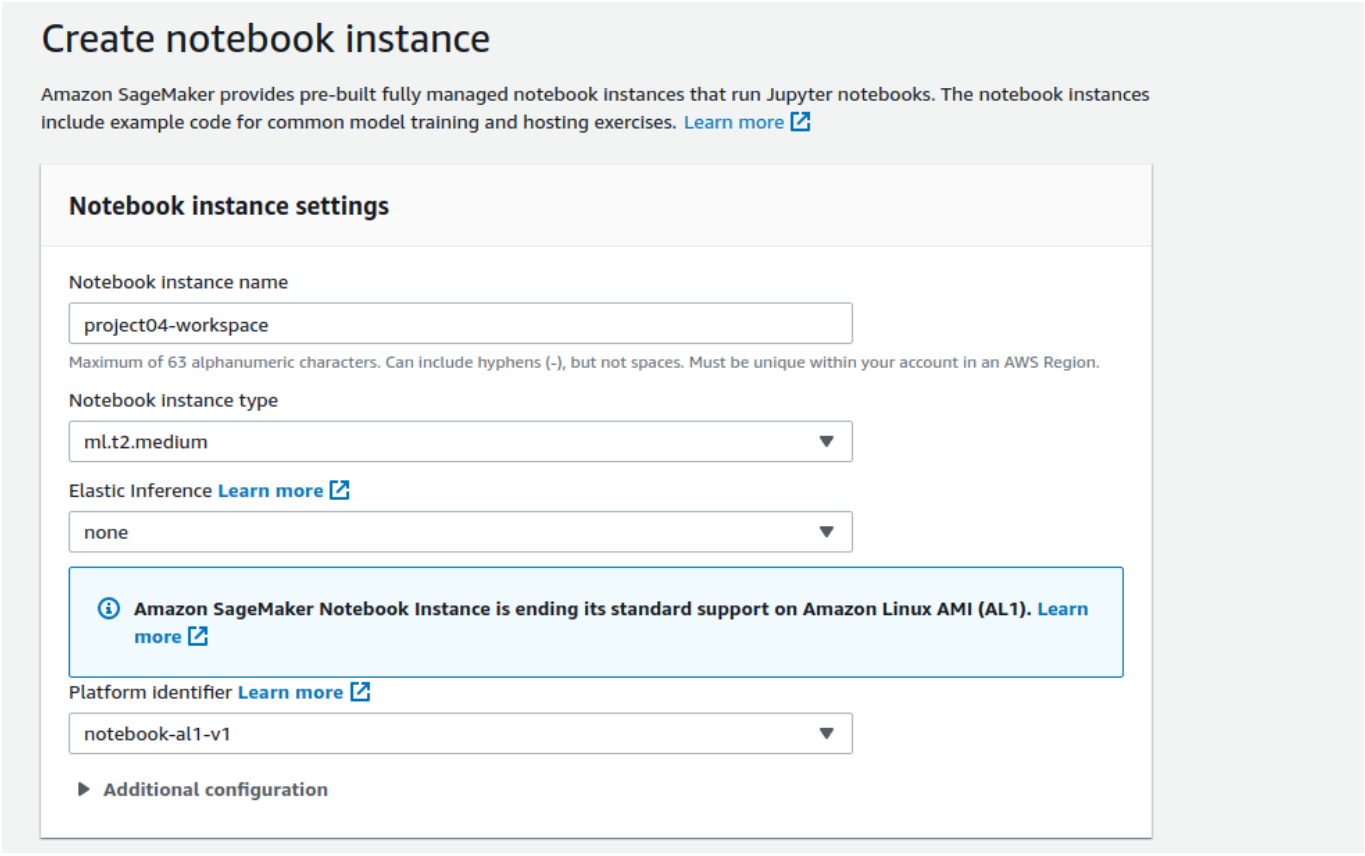
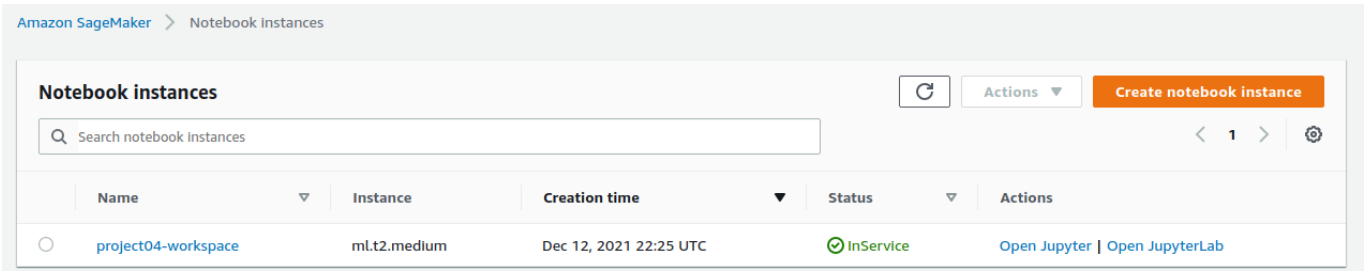
1. Initial setup, training, and deployment

a. Setup

First, we create a Sagemaker notebook instance. We select the `ml.t2.medium` instance, which has 2 vCPU and 4GB of RAM, since it is quite cheap ($0.0464 per hour). Additionally, we only run a lightweight task in the notebook to trigger Sagemaker training jobs and create endpoint. Hence, this instance type should fit our needs.



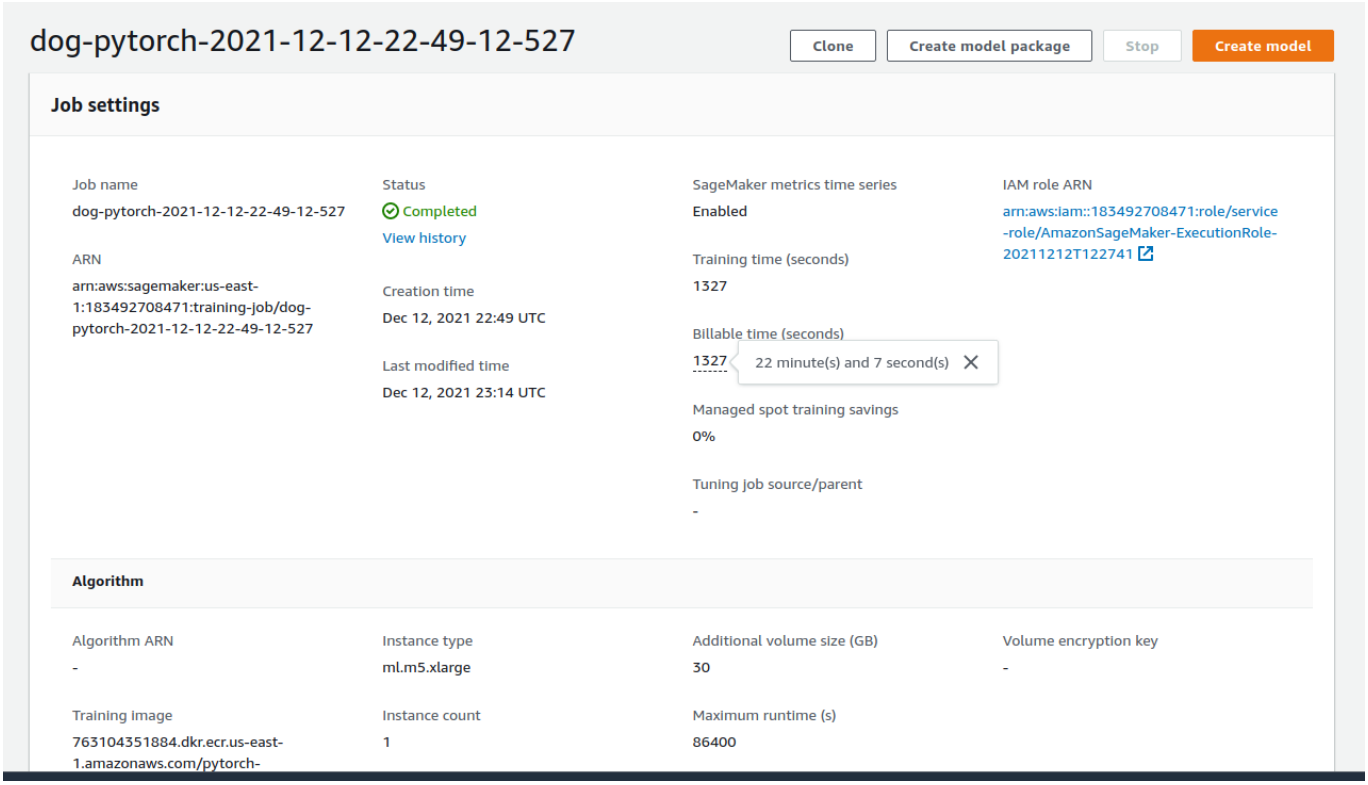


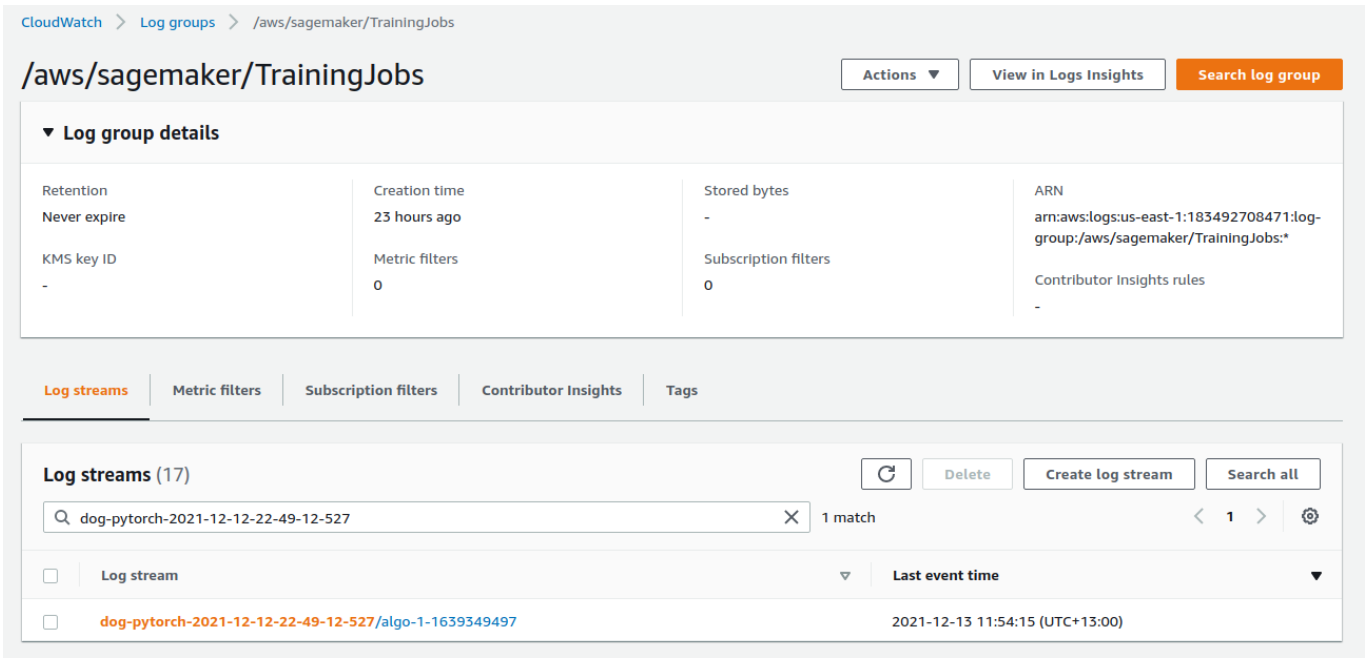
b. Training

Then, we open the Jupyterlab and upload the relevant notebook and Python scripts to that instance. Based on the best hyperparameter values from the hyperparameter tuning job, we run the training job twice: 1) using single instance, and 2) using multi-instance training (in this example, we use 4 instances of `ml.m5.xlarge`).

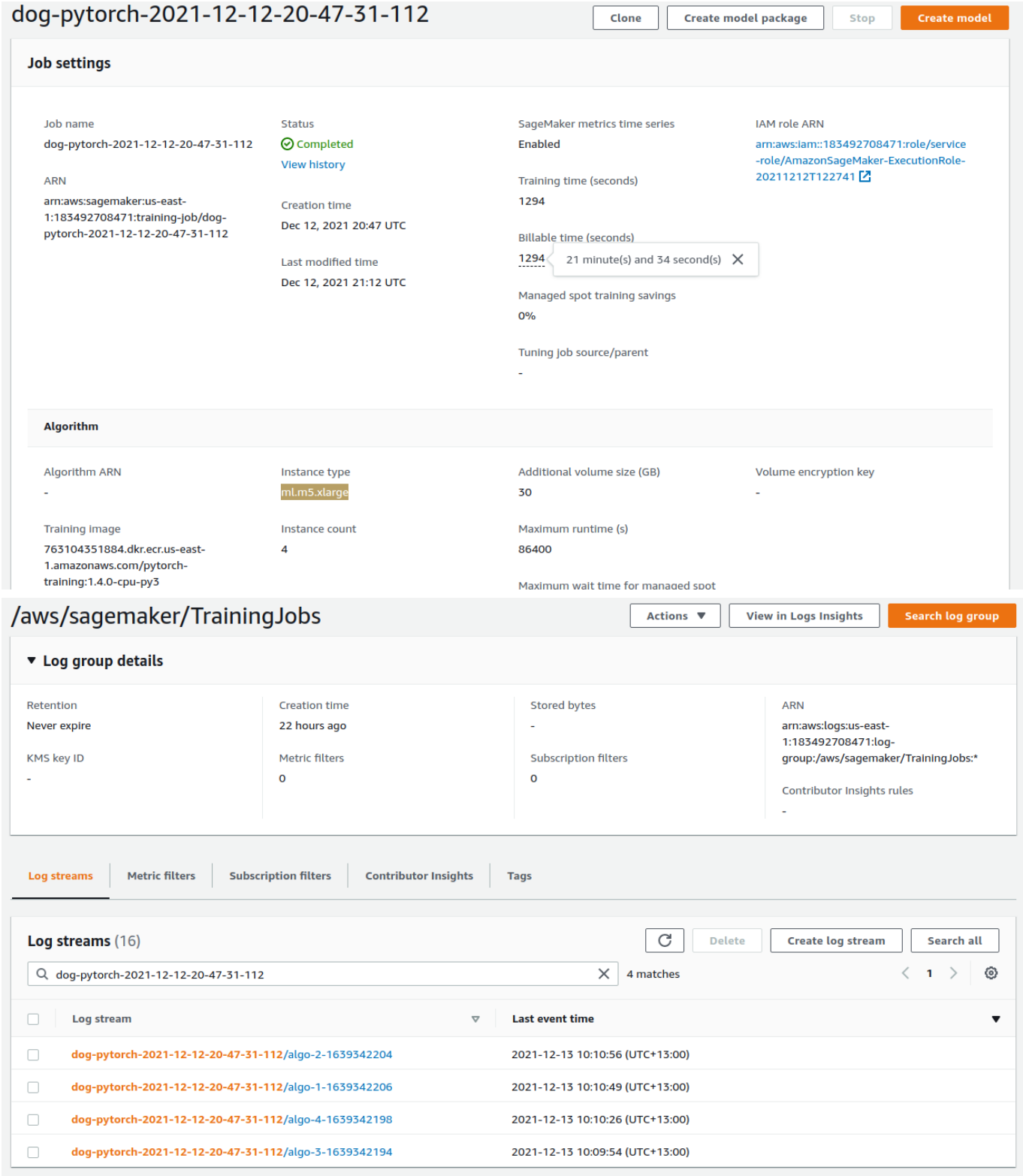
The single instance training takes around 22 minutes, while the multi-instance training takes around 21 minutes (no huge differences here).

Preview of the single-instance training job.





Preview of the multi-instance training job.



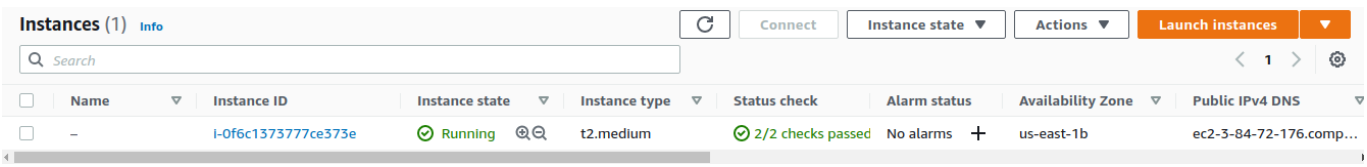
In terms of model performance, the difference is negligible. The single instance training results in 580 testing loss, while the multi-instance has 581 testing loss. Following figures show the logs from each training job.

```
2021-12-13T11:54:11.362+13:00  SM_USER_ARGS=["--batch_size","128","--learning_rate","0.03872229152368018"]
2021-12-13T11:54:11.362+13:00  SM_OUTPUT_INTERMEDIATE_DIR=/opt/ml/output/intermediate
2021-12-13T11:54:11.362+13:00  SM_CHANNEL_TRAINING=/opt/ml/input/data/training
2021-12-13T11:54:11.362+13:00  SM_HP_BATCH_SIZE=128
2021-12-13T11:54:11.362+13:00  SM_HP_LEARNING_RATE=0.03872229152368018
2021-12-13T11:54:11.362+13:00  PYTHONPATH=/opt/ml/code:/opt/conda/bin:/opt/conda/lib/python36.zip:/opt/conda/lib/python3.6:/opt/conda/lib/python3.6/lib-…
2021-12-13T11:54:11.362+13:00  Invoking script with the following command:
2021-12-13T11:54:11.362+13:00  /opt/conda/bin/python3.6 hpo.py --batch_size 128 --learning_rate 0.03872229152368018
2021-12-13T11:54:13.363+13:00  Namespace(batch_size=128, data='/opt/ml/input/data/training', learning_rate=0.03872229152368018, model_dir='/opt/ml/model…
2021-12-13T11:54:13.363+13:00  Hyperparameters are LR: 0.03872229152368018, Batch Size: 128
2021-12-13T11:54:13.363+13:00  Data Paths: /opt/ml/input/data/training
2021-12-13T11:54:14.363+13:00  Starting Model Training
2021-12-13T11:54:14.363+13:00  Epoch: 0
2021-12-13T11:54:15.364+13:00  [2021-12-12 22:54:15.209 algo-1:46 INFO json_config.py:90] Creating hook from json_config at /opt/ml/input/config/debugho…
2021-12-13T11:54:15.364+13:00  [2021-12-12 22:54:15.209 algo-1:46 INFO hook.py:192] tensorboard_dir has not been set for the hook. SMDebug will not be e…
2021-12-13T11:54:15.364+13:00  [2021-12-12 22:54:15.209 algo-1:46 INFO hook.py:237] Saving to /opt/ml/output/tensors
2021-12-13T11:54:15.364+13:00  [2021-12-12 22:54:15.209 algo-1:46 INFO state_store.py:67] The checkpoint config file /opt/ml/input/config/checkpointconf…
2021-12-13T11:54:15.364+13:00  [2021-12-12 22:54:15.211 algo-1:46 INFO hook.py:382] Monitoring the collections: relu_input, gradients, losses
2021-12-13T11:54:15.364+13:00  [2021-12-12 22:54:15.212 algo-1:46 INFO hook.py:443] Hook is writing from the hook with pid: 46
2021-12-13T12:09:20.884+13:00  train loss: 922.0000, acc: 0.0000, best loss: 1000000.0000
2021-12-13T12:11:04.915+13:00  valid loss: 581.0000, acc: 1.0000, best loss: 581.0000
2021-12-13T12:11:04.915+13:00  Testing Model
2021-12-13T12:12:46.942+13:00  Testing Loss: 580.0
2021-12-13T12:12:46.942+13:00  Testing Accuracy: 1.0
2021-12-13T12:12:46.942+13:00  Saving Model
2021-12-13T12:12:46.942+13:00  2021-12-12 23:12:46,678 sagemaker-training-toolkit INFO Reporting training SUCCESS
```

```
2021-12-13T09:52:01.859+13:00  SM_MODULE_DIR=s3://sagemaker-us-east-1-183492708471/dog-pytorch-2021-12-12-20-47-31-112/source/sourcedir.tar.gz
2021-12-13T09:52:01.859+13:00  SM_TRAINING_ENV={"additional_framework_parameters":{},"channel_input_dirs":{"training":"/opt/ml/input/data/training"},"curr…
2021-12-13T09:52:01.860+13:00  SM_USER_ARGS=["--batch_size","128","--learning_rate","0.03872229152368018"]
2021-12-13T09:52:01.860+13:00  SM_OUTPUT_INTERMEDIATE_DIR=/opt/ml/output/intermediate
2021-12-13T09:52:01.860+13:00  SM_CHANNEL_TRAINING=/opt/ml/input/data/training
2021-12-13T09:52:01.860+13:00  SM_HP_BATCH_SIZE=128
2021-12-13T09:52:01.860+13:00  SM_HP_LEARNING_RATE=0.03872229152368018
2021-12-13T09:52:01.860+13:00  PYTHONPATH=/opt/ml/code:/opt/conda/bin:/opt/conda/lib/python36.zip:/opt/conda/lib/python3.6:/opt/conda/lib/python3.6/lib-dy…
2021-12-13T09:52:01.860+13:00  Invoking script with the following command:
2021-12-13T09:52:01.860+13:00  /opt/conda/bin/python3.6 hpo.py --batch_size 128 --learning_rate 0.03872229152368018
2021-12-13T09:52:02.860+13:00  Namespace(batch_size=128, data='/opt/ml/input/data/training', learning_rate=0.03872229152368018, model_dir='/opt/ml/model',…
2021-12-13T09:52:02.860+13:00  Hyperparameters are LR: 0.03872229152368018, Batch Size: 128
2021-12-13T09:52:02.860+13:00  Data Paths: /opt/ml/input/data/training
2021-12-13T09:52:03.866+13:00  Starting Model Training
2021-12-13T09:52:03.866+13:00  Epoch: 0
2021-12-13T09:52:05.866+13:00  [2021-12-12 20:52:05.348 algo-4:45 INFO json_config.py:90] Creating hook from json_config at /opt/ml/input/config/debughook…
2021-12-13T09:52:05.866+13:00  [2021-12-12 20:52:05.348 algo-4:45 INFO hook.py:192] tensorboard_dir has not been set for the hook. SMDebug will not be exp…
2021-12-13T09:52:05.866+13:00  [2021-12-12 20:52:05.348 algo-4:45 INFO hook.py:237] Saving to /opt/ml/output/tensors
2021-12-13T09:52:05.866+13:00  [2021-12-12 20:52:05.348 algo-4:45 INFO state_store.py:67] The checkpoint config file /opt/ml/input/config/checkpointconfig…
2021-12-13T09:52:05.866+13:00  [2021-12-12 20:52:05.351 algo-4:45 INFO hook.py:382] Monitoring the collections: relu_input, losses, gradients
2021-12-13T09:52:05.866+13:00  [2021-12-12 20:52:05.352 algo-4:45 INFO hook.py:443] Hook is writing from the hook with pid: 45
2021-12-13T10:06:58.399+13:00  train loss: 1205.0000, acc: 1.0000, best loss: 1000000.0000
2021-12-13T10:08:42.440+13:00  valid loss: 581.0000, acc: 1.0000, best loss: 581.0000
2021-12-13T10:08:42.440+13:00  Testing Model
2021-12-13T10:10:25.507+13:00  Testing Loss: 581.0
2021-12-13T10:10:25.507+13:00  Testing Accuracy: 1.0
2021-12-13T10:10:25.507+13:00  Saving Model
2021-12-13T10:10:26.507+13:00  2021-12-12 21:10:25,720 sagemaker-training-toolkit INFO Reporting training SUCCESS
```

2. Training on EC2 instance

Here, we train a similar image classification model using an EC2 instance without changing any default hyperparameters from the starter script. We decide to use `t2.medium` instance (2 vCPU and 4 GB of RAM) since the training script only requires 5 epochs and the batch size is only (2). If the batch size is larger, we will need to use an instance with bigger RAM. We don't request a spot instance to ensure the instance is preserved to our work.



The data are downloaded and unzipped using the command line. To ensure it's reproducible easily, we put the commands in `workspace/src/ec2-data-download.sh`.

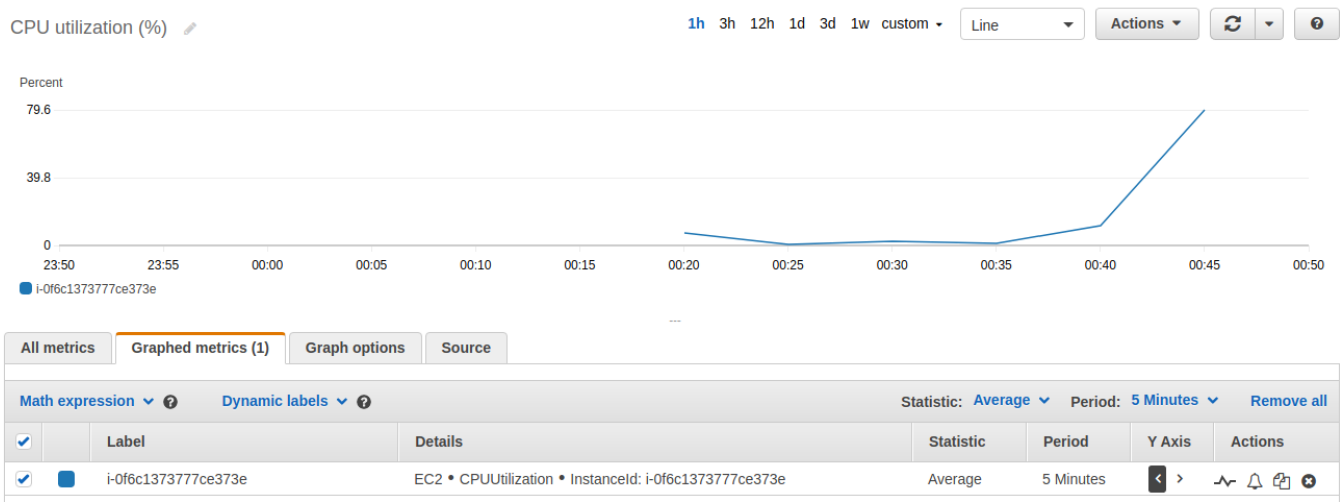
```
[root@ip-172-31-86-26 ~]# chmod 755 ec2-data-download.sh
[root@ip-172-31-86-26 ~]# ls -al
total 24
dr-xr-x---  6 root root 174 Dec 13 00:31 .
dr-xr-xr-x 18 root root 257 Nov 22 18:50 ..
-rw-r--r--  1 root root  18 Oct 18  2017 .bash_logout
-rw-r--r--  1 root root 176 Oct 18  2017 .bash_profile
-rw-r--r--  1 root root 176 Oct 18  2017 .bashrc
drwxr-xr-x  3 root root  17 Nov 22 18:55 .cache
-rw-r--r--  1 root root 100 Oct 18  2017 .cshrc
-rwxr-xr-x  1 root root 235 Dec 13 00:31 ec2-data-download.sh
drwxr-xr-x  2 root root  24 Nov 24 23:56 .keras
drwxr-xr-x  2 root root  24 Nov 22 20:23 .safety
drwx------  2 root root  29 Dec 13 00:22 .ssh
-rw-r--r--  1 root root 129 Oct 18  2017 .tcshrc
[root@ip-172-31-86-26 ~]# ./ec2-data-download.sh
Downloading the data ...
--2021-12-13 00:32:37--  https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/dogImages.zip
Resolving s3-us-west-1.amazonaws.com (s3-us-west-1.amazonaws.com)... 52.219.116.16
Connecting to s3-us-west-1.amazonaws.com (s3-us-west-1.amazonaws.com)|52.219.116.16|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1132023110 (1.1G) [application/zip]
Saving to: 'dogImages.zip'

10% [=========>                                                                  ] 115,580,555 5.29MB/s  eta 2m 44s
```

```
  inflating: dogImages/valid/129.Tibetan_mastiff/Tibetan_mastiff_08185.jpg
   creating: dogImages/valid/130.Welsh_springer_spaniel/
  inflating: dogImages/valid/130.Welsh_springer_spaniel/Welsh_springer_spaniel_08201.jpg
  inflating: dogImages/valid/130.Welsh_springer_spaniel/Welsh_springer_spaniel_08206.jpg
  inflating: dogImages/valid/130.Welsh_springer_spaniel/Welsh_springer_spaniel_08222.jpg
  inflating: dogImages/valid/130.Welsh_springer_spaniel/Welsh_springer_spaniel_08228.jpg
  inflating: dogImages/valid/130.Welsh_springer_spaniel/Welsh_springer_spaniel_08235.jpg
  inflating: dogImages/valid/130.Welsh_springer_spaniel/Welsh_springer_spaniel_08240.jpg
   creating: dogImages/valid/131.Wirehaired_pointing_griffon/
  inflating: dogImages/valid/131.Wirehaired_pointing_griffon/Wirehaired_pointing_griffon_08251.jpg
  inflating: dogImages/valid/131.Wirehaired_pointing_griffon/Wirehaired_pointing_griffon_08263.jpg
  inflating: dogImages/valid/131.Wirehaired_pointing_griffon/Wirehaired_pointing_griffon_08266.jpg
  inflating: dogImages/valid/131.Wirehaired_pointing_griffon/Wirehaired_pointing_griffon_08279.jpg
   creating: dogImages/valid/132.Xoloitzcuintli/
  inflating: dogImages/valid/132.Xoloitzcuintli/Xoloitzcuintli_08298.jpg
  inflating: dogImages/valid/132.Xoloitzcuintli/Xoloitzcuintli_08299.jpg
  inflating: dogImages/valid/132.Xoloitzcuintli/Xoloitzcuintli_08301.jpg
  inflating: dogImages/valid/132.Xoloitzcuintli/Xoloitzcuintli_08304.jpg
   creating: dogImages/valid/133.Yorkshire_terrier/
  inflating: dogImages/valid/133.Yorkshire_terrier/Yorkshire_terrier_08333.jpg
  inflating: dogImages/valid/133.Yorkshire_terrier/Yorkshire_terrier_08334.jpg
  inflating: dogImages/valid/133.Yorkshire_terrier/Yorkshire_terrier_08336.jpg
  inflating: dogImages/valid/133.Yorkshire_terrier/Yorkshire_terrier_08348.jpg
Preparing the model directory ...
[root@ip-172-31-86-26 ~]# ls
dogImages  dogImages.zip  ec2-data-download.sh  TrainedModels
[root@ip-172-31-86-26 ~]# ls -l
total 1105496
drwxr-xr-x 5 root root         44 Mar 27  2017 dogImages
-rw-r--r-- 1 root root 1132023110 Apr  1  2017 dogImages.zip
-rwxr-xr-x 1 root root        235 Dec 13 00:31 ec2-data-download.sh
drwxr-xr-x 2 root root          6 Dec 13 00:36 TrainedModels
```

We need to activate the right conda environment: `source activate pytorch_latest_p37` - otherwise, the default python environment doesn't have Pytorch installed.

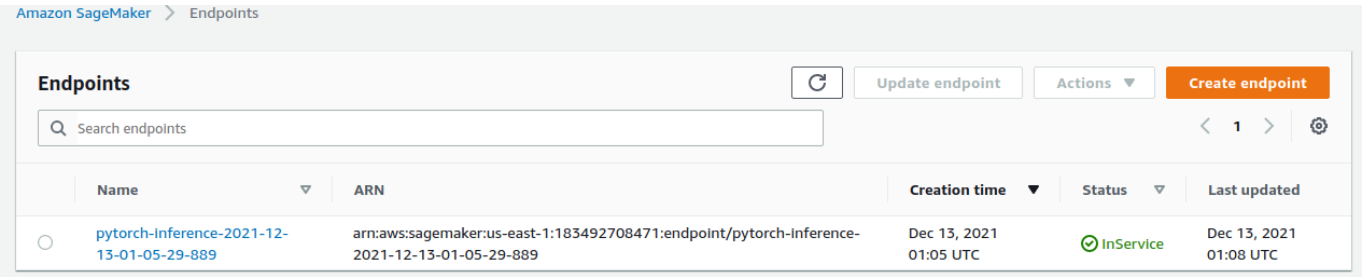
```
[root@ip-172-31-86-26 ~]# source activate pytorch_latest_p37
(pytorch_latest_p37) [root@ip-172-31-86-26 ~]#
(pytorch_latest_p37) [root@ip-172-31-86-26 ~]# ls
dogImages  dogImages.zip  ec2-data-download.sh  solution.py  TrainedModels
(pytorch_latest_p37) [root@ip-172-31-86-26 ~]# python solution.py
Downloading: "https://download.pytorch.org/models/resnet50-19c8e357.pth" to /root/.cache/torch/hub/checkpoints/resnet50-19c8e357.pth
100%|                                                                                | 97.8M/97.8M [00:01<00:00, 71.8MB/s
Starting Model Training
saved
(pytorch_latest_p37) [root@ip-172-31-86-26 ~]# ls
dogImages  dogImages.zip  ec2-data-download.sh  solution.py  TrainedModels
(pytorch_latest_p37) [root@ip-172-31-86-26 ~]# ls TrainedModels/
model.pth
(pytorch_latest_p37) [root@ip-172-31-86-26 ~]# ls -l TrainedModels/
total 93236
-rw-r--r-- 1 root root 95470465 Dec 13 00:50 model.pth
```



Both scripts that we run using Sagemaker and EC2 train a similar Pytorch model. However, when we run it on Sagemaker, we have to pass the input and output data (datasets, model) as environment variables. It has to be done that way since every time we trigger Sagemaker training jobs, they could be executed on different instances. Meanwhile, when we train a model on EC2, we can directly use local datasets and store the model right on the EC2 instance (since the storage is persistent, unless we delete the instance). The remaining part of the code is quite similar (but we should note that the EC2 training script has the hyperparameters hard-coded rather than specified as command line arguments).

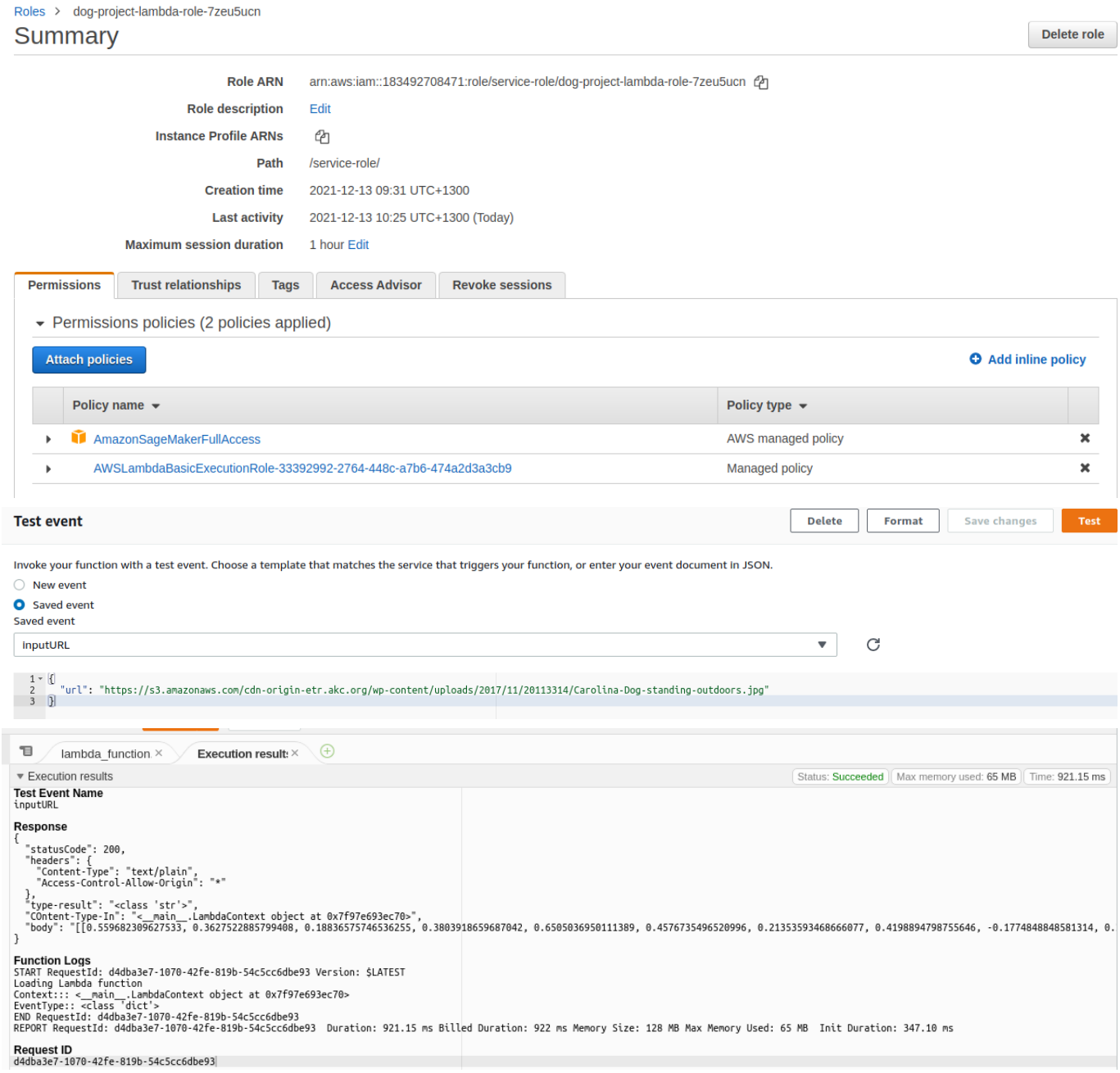
3. Lambda function setup

The lambda function receives a JSON input that contains URL of an image, then, it instantiates a Sagemaker runtime object to invoke our deployed endpoint from part #1. The response from that endpoint is stored as the response body of the lambda function's output. The whole preprocessing logic is handled by the endpoint (in `src/inference.py`).

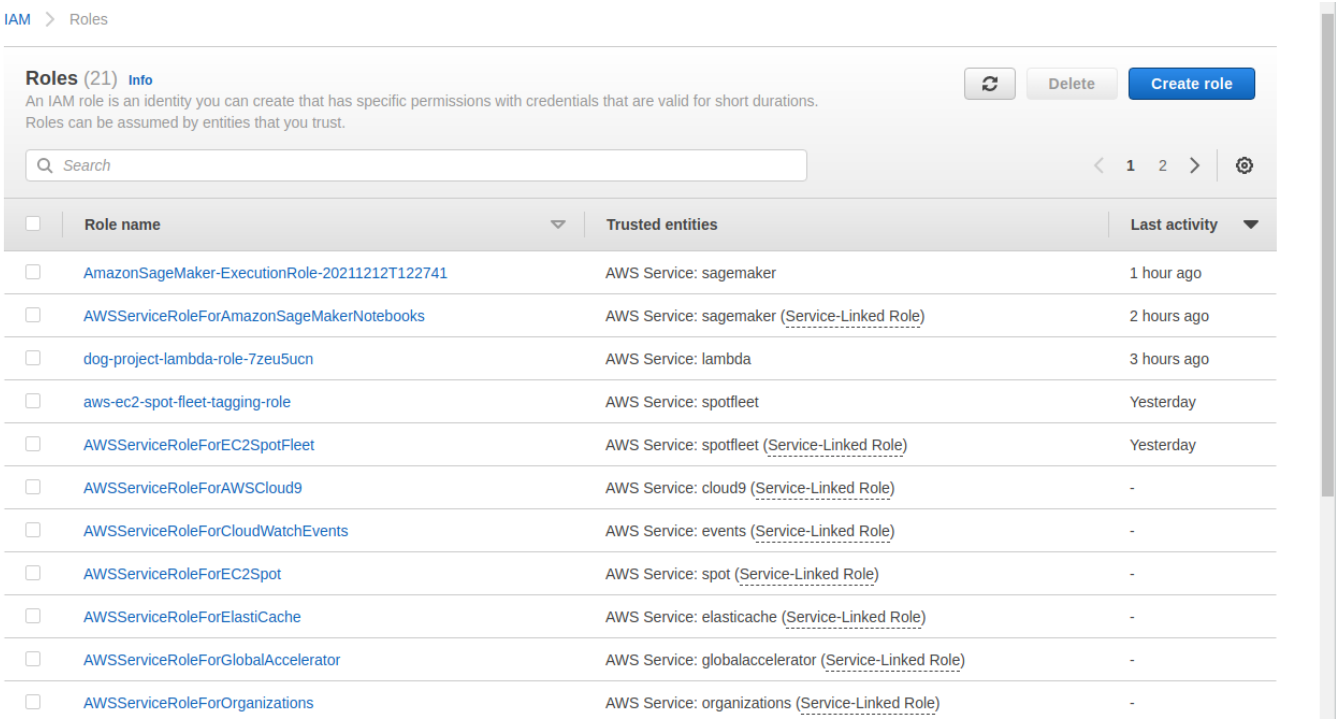


4. Security and testing

To ensure the Lambda function in part #3 can hit the endpoint, we need to give it access to Sagemaker. Here, we assign `SagemakerFullAccess` to the IAM role that we use on the Lambda function.

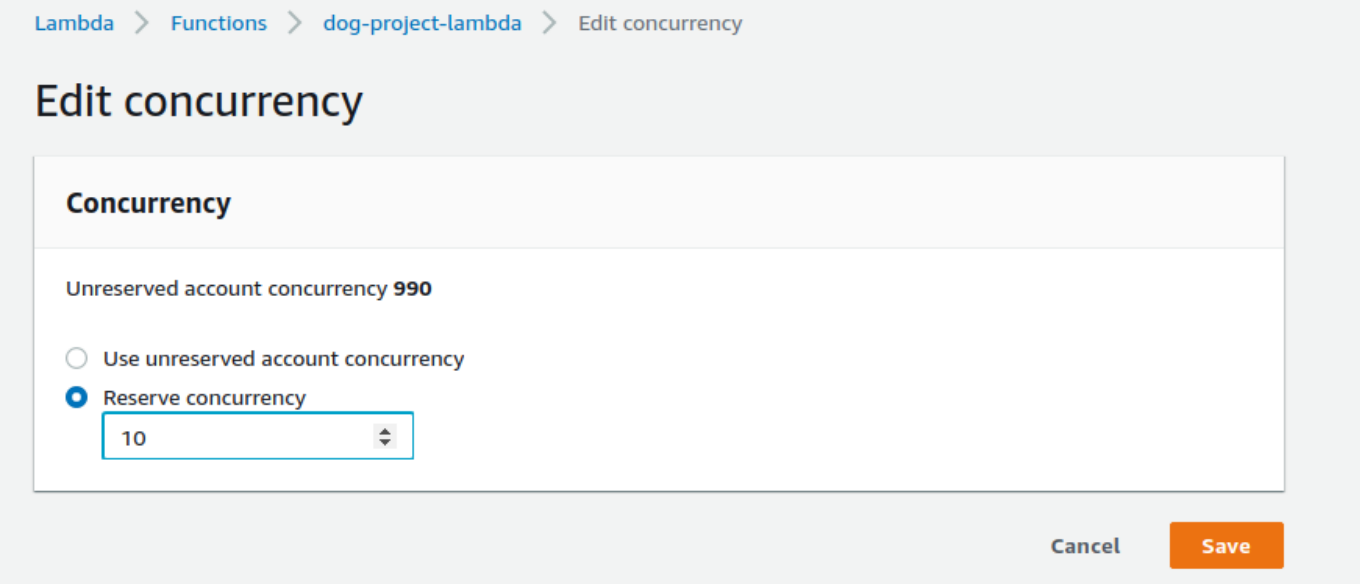


The following figure shows a preview of our IAM role dashboard. There aren't a lot of custom roles there, as this account is used specifically for this course. There are no sensitive information stored anywhere in this account (so it should be safe). However, since we assigned `SagemakerFullAccess` to the Lambda function, it allows any operations to the Sagemaker resources. Unfortunately, I don't find another role that allows invoking endpoint without giving too much control of the resources.

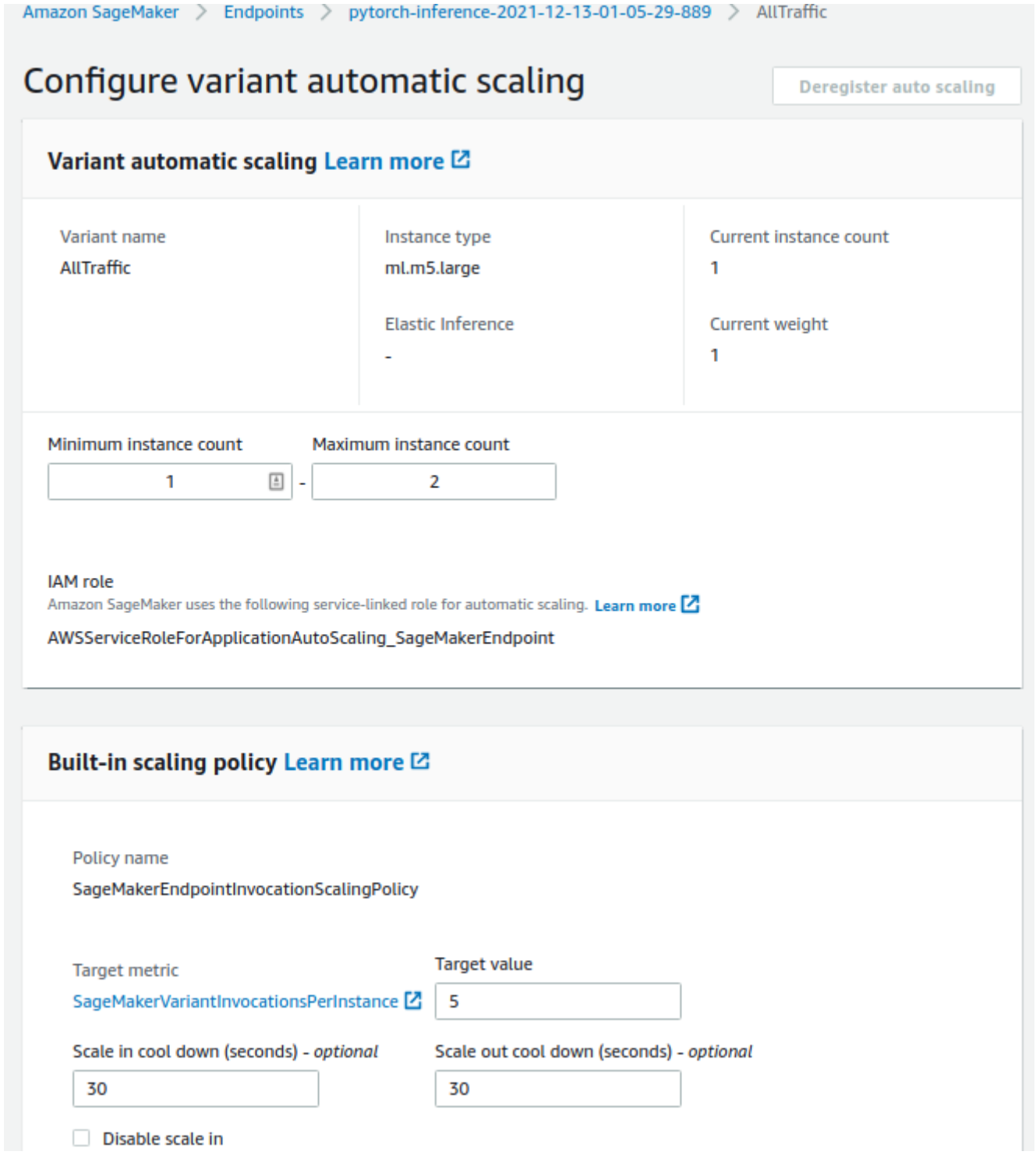


5. Concurrency and autoscaling

Usually, Lambda function is used to bridge the application request to the endpoint. In this example, we define a reserved concurrency and set it to 10, i.e., the Lambda function can process 10 requests at the same time.



Besides, we also configure autoscaling on our endpoint. We define the maximum instance to 2 (since it is just a course example), with the threshold of 5 invocations per instance, i.e., if there are more than 5 invocations, another instance will be spinned up. The scale-in and scale-out duration are set to 30 seconds to avoid downtime.


Amazon SageMaker > Endpoints > pytorch-inference-2021-12-13-01-05-29-889 > 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
2
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
30
Scale out cool down (seconds) - optional
30
Disable scale in