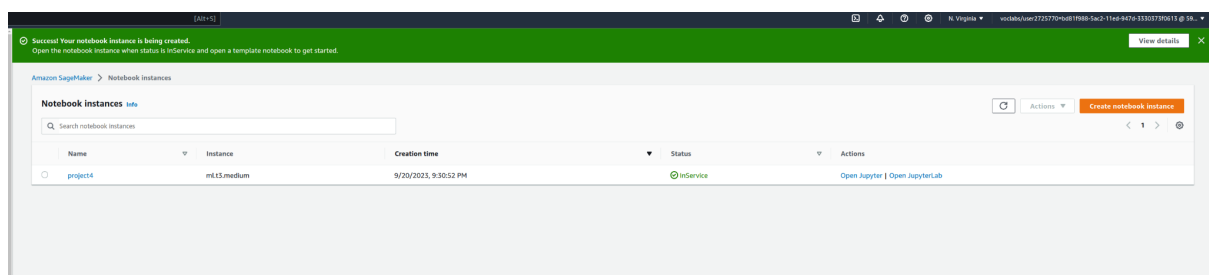


1. Write a short justification of why you chose the instance type you did.

I have chosen a 'ml.t3.medium' instance type. It only has vCPUs and 4 Gib memory, but that is more than enough to run the project's notebook, since all the heavy-weight work of training models will be performed by the instances that are specified when we create the estimators. The same logic applies for inference endpoints: the inferences are going to be run on dedicated instances that we specify when we deploy our models.

Another point of attention is the block storage size. Since we have to download the image dataset to the machine, and then upload it to S3, and we are dealing with a relatively large image dataset, I have increased the EBS size to 50Gb, which is more than enough. Taking a look at the hard drive utilization yields the following:

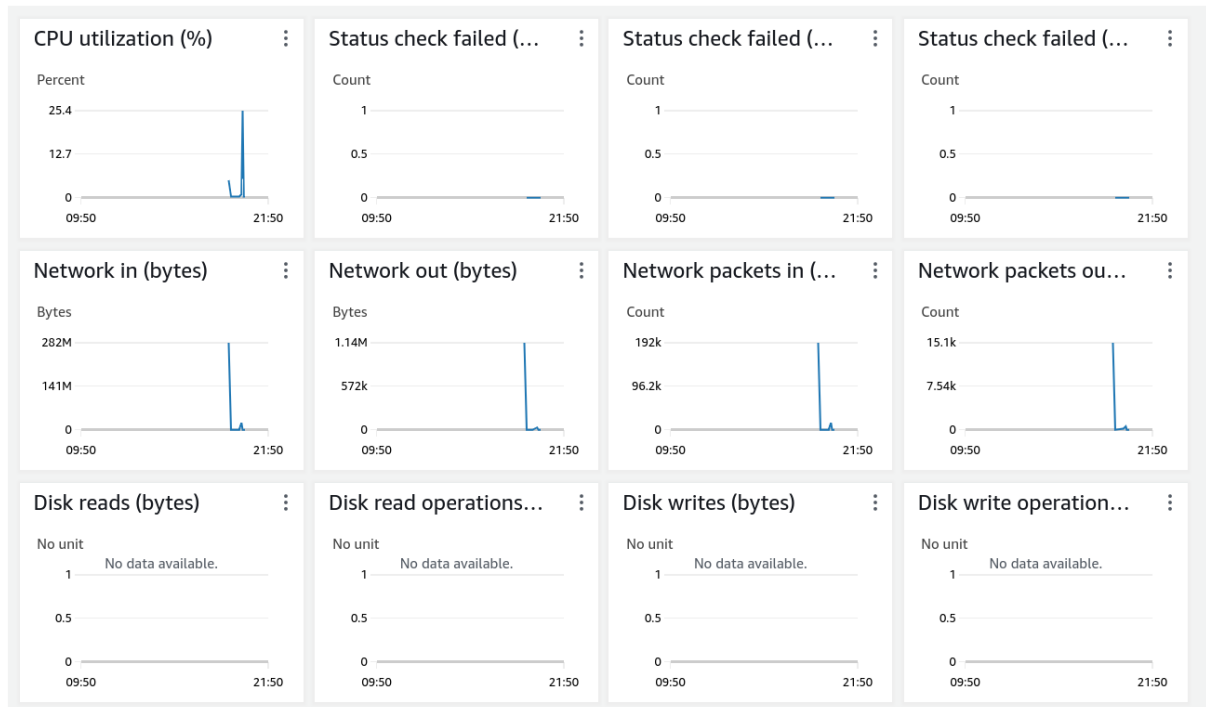
```
sh-4.2$ df -h
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        1.9G   0    1.9G   0% /dev
tmpfs           1.9G  4.0K   1.9G   1% /dev/shm
tmpfs           1.9G 588K   1.9G   1% /run
tmpfs           1.9G   0    1.9G   0% /sys/fs/cgroup
/dev/nvme0n1p1  135G   96G   40G   71% /
/dev/nvme1n1    50G   12G   36G   25% /home/ec2-user/SageMaker
tmpfs           386M   0    386M   0% /run/user/1002
tmpfs           386M   0    386M   0% /run/user/1001
tmpfs           386M   0    386M   0% /run/user/1000
sh-4.2$
```



2. Decide the type of instance you want, create it in your workspace, and write a justification of why you chose the instance type you did.

I have decided to use an ml.m5.xlarge instance type with an 'Deep Learning AMI GPU PyTorch 1.13.1' AMI. It is the same type of instance used during the sagemaker training jobs created in the notebook. The main rationale was to see how the instance's perform compared to the sagemaker job. To my surprise the training process itself was quicker than the equivalent sagemaker training job. Taking a look at the ec2 metrics, it can be seen that, at peak demand, only 25.4% of the cpu was

used. So if we were running on a tight budget, we could try performing further training jobs in a smaller instance.



3. write at least one paragraph about the differences between the code in `ec2train1.py` and the code you used in Step 1.

In terms of code, one of the main differences is the absence of the `"if __name__ == '__main__':"` in the code that is run in the ec2, because it is not necessary to set parameters that the script should receive like, data location in s3 and hyperparameters configurations. All data is already loaded in the machine and the hyperparameter values are hard coded (See image below).

When training with sagemaker estimators, our code need to be parametrizable, and comply with certain standards (like receiving arguments via command line with the proper names). Besides that, the infrastructure of where our job is going to be run is also a parameter. When you train on ec2 instances, the hardware that you have is the hardware that you get. You cannot change that, so the focus is more on optimizing the code.

```

126 return train_data_loader, test_data_loader, validation_data_loader
127
128 def main(args):
129     logger.info(f'Hyperparameters are LR: {args.learning_rate}, Batch Size: {args.batch_size}')
130     logger.info(f'Data Paths: {args.data}')
131
132     train_loader, test_loader, validation_loader = create_data_loaders(args.data, args.batch_size)
133     model = net()
134
135     criterion = nn.CrossEntropyLoss(ignore_index=133)
136     optimizer = optim.Adam(model.parameters()), lr=args.learning_rate
137
138     logger.info("Starting Model Training")
139     model = train(model, train_loader, validation_loader, criterion, optimizer)
140
141     logger.info("Testing Model")
142     test(model, test_loader, criterion)
143
144     logger.info("Saving Model")
145     torch.save(model.cpu().state_dict(), os.path.join(args.model_dir, 'model.pth'))
146
147 if __name__ == '__main__':
148     parser = argparse.ArgumentParser()
149     parser.add_argument('--learning_rate', type=float)
150     parser.add_argument('--batch_size', type=int)
151     parser.add_argument('--data', type=str, default=os.environ['SM_CHANNEL_TRAINING'])
152     parser.add_argument('--model_dir', type=str, default=os.environ['SM_MODEL_DIR'])
153     parser.add_argument('--output_dir', type=str, default=os.environ['SM_OUTPUT_DATA_DIR'])
154
155     args = parser.parse_args()
156     print(args)
157
158     main(args)
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

3. Write at least 1 paragraph describing how this function is written and how it works.

The function works by receiving a payload with the data that should go through our trained model. This payload is called and event. In this specific case, the event is a json object which contains the url of the image against which we should run our predictions. We use a sagemaker runtime to invoke the endpoint, passing the endpoint name, type of payload, type of accepted response, and the payload itself.

After sending the request, we receive another json object, called response. In the response body, we find the content of our prediction. This content is then desserialized(converted from bytes to python object). In final part, the function return the success status code, along with other response parameters and the serialized and parsed response.

#### 4. Add the result that your lambda function returns (the list of 33 numbers) to your solution writeup

Successfully updated the function **dogBreedEndpoint**.

CodeTestMonitorConfigurationAliasesVersions

Executing function: succeeded (logs)

Details

The area below shows the last 4 KB of the execution log.

```
"body": "[[-7.739333629608154, -2.7622897624969482, 0.8739001750946045, -1.5595126152038574, -2.921342372894287, -6.512680530548096, -1.298835277557373, -3.1380233764648438, -5.679790019989014, 0.09424073994159698, -0.7023207545280457, -3.699753999710083, -0.5298449397087097, 1.8055620193481445, -6.44109582901001, -3.5566046237945557, -8.462738990783691, -0.9340609908103943, -4.784125804901123, 1.242956280708313, -2.0716934204101562, -0.33157360553741455, -5.740781307220459, -8.139474868774414, -4.3748650550842285, -7.834297180175781, -2.0248398780822754, -3.1405723094940186, -3.763457775115967, -1.522344708442688, -2.7329602241516113, -3.726069450378418, -6.612259864807129, -3.0951271057128906, -6.264786720275879, -5.49496547698975, -4.351595878601074, -2.6484873294830322, -1.9210385084152222, -6.156394958496094, -3.53096604347229, -3.263073444366455, -0.24803046882152557, -5.453585624694824, 0.6520899534225464, -6.473347187042236, -2.5631844997406006, -0.34482499957084656, -2.6791770458221436, -1.5603928565979004, -5.9325127601623535, -10.780133247375488, -6.549309730529785, -1.75835120677948, -4.4079742431640625, 0.33166998624801636, -4.973919868469238, -5.878260612487793, -0.17670506238937378, -3.1703743934631348, -6.765723705291748, -5.240838527679443, -7.083765506744385, -6.583903789520264, -1.4982601404190063, -4.429241180419922, -1.0458904504776, -4.629544734954834, -3.9800310134887695,
```

Summary

Code SHA-256	Execution time
pZAb+Ysu3iTaC8PK8XAXRdMnMl1Tuz0ZG2dd577XwRk=	43 seconds ago (September 21, 2023 at 05:36 PM GMT-3)
Request ID	Function version
b4f2a977-fed7-457f-919f-cd8f811d2536	\$LATEST
Init duration	Duration
356.60 ms	1374.39 ms
Billed duration	Resources configured
1375 ms	128 MB
Max memory used	
72 MB	

Log output

The section below shows the logging calls in your code. [Click here](#) to view the corresponding CloudWatch log group.

#### 5. Write about the security of your AWS workspace in your writeup.

[Alt+S] ⓘ ⓘ ⓘ ⓘ ⓘ Global ▼ voclabs/user2725770=bd81f988-5ac2-11ed-947d-3330373f0613 @ 59...

IAM > Roles > dogBreedEndpoint-role-qpng5vl9

dogBreedEndpoint-role-qpng5vl9 Info

Delete

Summary

Edit

Creation date	ARN
September 21, 2023, 17:24 (UTC-03:00)	arn:aws:iam::596594075945:role/service-role/dogBreedEndpoint-role-qpng5vl9
Last activity	Maximum session duration
-	1 hour

Permissions

Trust relationships

Tags

Access Advisor

Revoke sessions

Permissions policies (2) Info

🔄 Simulate 📄 Remove Add permissions ▼

You can attach up to 10 managed policies.

Filter by Type

Q Search All types < 1 > ⚙

<input type="checkbox"/>	Policy name	Type	Attached entities
<input type="checkbox"/>	AWSLambdaBasicExecutionRole-...	Customer managed	1
<input type="checkbox"/>	AmazonSageMakerFullAccess	AWS managed	2

Due to the absence of a better suited AWS managed policy, I have used policy called "AmazonSageMakerFullAccess", which, as the name suggest, grants full access to sagemaker services. This is against AWS *Principle of Least Privilege*, which states that a resource should not be given any more privileges than necessary. Since our function is only going to invoke endpoints on sagemaker, the ideal would be to go to IAM, customize this policy, and remove all access that are not relevant for this lambda function.

6. Write about the choices you made in the setup of concurrency and auto-scaling, and why you made each of those choices.

I have set the function to use the unreserved account concurrency of 900 invocations plus a provisioned concurrency of 100 invocations. Under this setting, there is 100 execution environments that are initialized and prepared to respond immediately to function invocations. If the throughput goes beyond this threshold, there is still 900 execution environments available, but they will not be as performant, mainly in terms of latency, as the ones in from the provisioned concurrency.

For autoscaling I have configured a minimum of 1 instance and a maximum of 4. The number 4 was chosen thinking about a scenario of peak demand, when some sort of event like a promotion, or our application goes viral, makes the amount of request to our model increase substantially. The scaling policy was set in a way that, as soon as the instance get 250 "simultaneous"(short time-spaced) invocations per instance, the number of instances will be increased. Both the scale in and out periods are of 300 seconds.

In a real world scenario, to estimate the optimal concurrency and auto-scaling configuration to minimize costs, latency and satisfy other business requirements, we would need to analyze historical invocation behavior and decide the appropriate amounts of each type of concurrency.

# Configure variant automatic scaling

Deregister auto scaling

## Variant automatic scaling [Learn more](#)

Variant name

AllTraffic

Instance type

ml.c6i.large

Current instance count

1

Elastic Inference

-

Current weight

1

Minimum instance count

1

Maximum instance count

4

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

250

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](#)