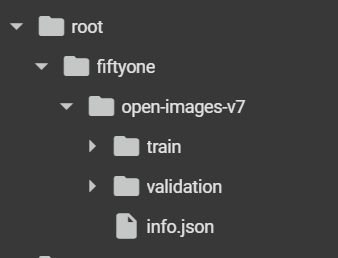
**AWS SageMaker ML**

1. Asnyhcronoous machine learning is imp because we wait for the batch transformation and job to be finished and then move on
2. Creating an IAM User for safe access to the AWS account
3. Login with root user then go to the IAM services, create user, attach the policies directly, administrative access provide, then autogenerate password. Save the downloaded csv file
4. The project is based on object detection: mainly detecting the plastic bag in the images
5. The dataset for the project is taken from open image dataset and the download is performed in google colab using voxel51: <https://storage.googleapis.com/openimages/web/download_v7.html>
6. The images are downloaded using fiftyone package:
7. 
8. To download the images we are putting all the images in a zip and then locally downloading it to put it into the AWS. This is done using “shutil” package

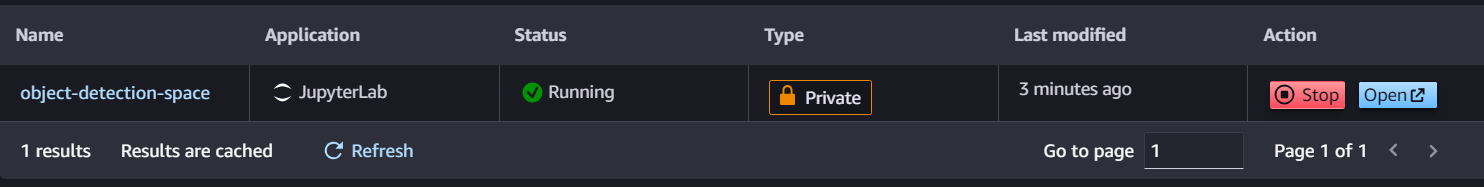
AWS Sagemaker AI : Data Analytics and AI

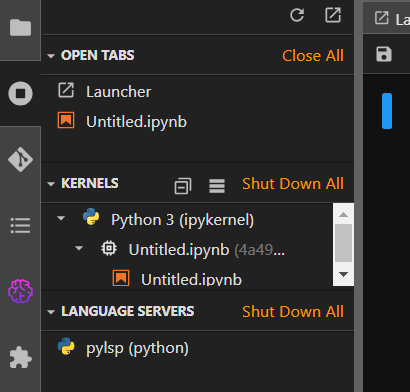
What is AWS Sagemaker AI?

1. Building a space or a jupyter notebook in AWS Sagemaker Studio (Jupyter Lab)
2. The docker images which are stored in AWS ECR has the python libraries install from where you can load the python packages. Therefore it depends on which task you have. But in jupyterlab everything is prehandled (this is the legacy change)

Steps

1. Creating Sagemaker domain in the region nearest (Europe)
2. Domain: Amazon Elastic File System
3. For creating domain, first create a VPC in that region with 10.0.0.0.0/16 ip address
4. Then create Subnet in this VPC (It is required because no VPC was present in the AWS IAM User)
5. Then go to domain in Amazon Sagemaker and create domain with quick set up, it will give default name
6. After setting up the domain, go to user and then launch jupyterlab
7. Select the app Jupyterlab and create a space
8. The instance (ml.t3.medium) has the fees incurred, thus work accordingly (depends on how powerful CPU or GPU you need)
9. Image is the docker image which contains libraries installed stored in AWS ECR, this can help us to install the packages without any issue



1. After it starts running then open the jupyter notebook
2. Creating new notebook from file, then select the python kernel (environment)
3. Shut down the jupyter lab notebook
4. 
5. Pricing : <https://aws.amazon.com/sagemaker-ai/pricing/>
6. The files which you upload on the Jupyterlab is actually uploaded in AWS Elastic file System therefore this also incur charges
7. Getting the image ids by looping through the folder using glob (for this we have changed the instance to 8vcpu ml.t3.2xlarge 32GB
8. In EDA mainly matplotlib.patches has the bounding boxes code and the xmin,xmax, ymin and ymax columns of the detection.csv has the scaled value of where bounding box is drawn. Therefore the exact Xmin coordinate is calculated by multiplying xmin by width of image (img.size())
9. First cleansing the dataframe to only contain the plastic bag labels
10. Then for the test set, we are splitting the train set 80-20%
11. Shutil package is use to move and copy the path and also work with the zip and unzip
12. We will copy the images which are present in both train and test set and move the images which are only in test set (this is because, bounding boxes are the data points, and one image has more than 1 bb, which leads to be present in both train and test, therefore instead of directly moving the image we need to also keep it into the train set)
13. Apache Mxnet Gluon is similar to tensorflow, a deeplearning neural network library to train the neural network and deploy models. It is faster therefore AWS is using this, more scalable. Use Resnet for the image
14. RecordIO is a file format for efficiently storing and reading large amounts of binary data, such as images, in a sequential manner.
15. Dataframe has to be converted in a certain format to be given to recordio for creating the image matrix
16. The pattern is always same, first column is index, second column indicates the label is made of 2 columns ( 2 itself and 3)
17. The third column is label information width that is 5, class index and 4 bb coordinates.
18. For multiple bb in one image, all the other information must be present in one row
19. 4th column is the zero based class id for the object
20. The last column is the img path within the image dir
21. Images should be listed in one line, this is because of storage.
22. Even if you have 2 plastic bag annotations in a single image the label width should be 5 and NOT 10 as in the previous version. because of the new updates and now Mxnet will be able to infer how many annotations are on a line without you having to explicitly tell it.
23. After completely converting the train dataset and test dataset in this format we will move on into training the model (Everything is present in the code with comments)
24. Watch video 52 again for creating product.lst files
25. In data augmentation we are flipping the images, so we have to correctly flip the bounding box as well so that it works correctly.
26. Service quota has to be increase for training purpose by going to service quotas in aws and then choosing the one with GPU (ml.p2), this has to be done when the resource limited excess error occurs
27. The .lst files created is then converted in recordio format for training using apache mxnet conversion github code im2rec.py
28. When you have converted the .lst file in rec format then it has the whole information about images as well. Therefore at this time you can delete the images, but we will keep for now so that if the training is successful, then we will delete
29. Then we will upload rec files in S3 bucket
30. The S3 bucket whole code is made inside the python using sagemaker and the notation which is used in the AWS this has to be same
31. Then we will download the algorithm of object detection from AWS which is present in the ECR- elastic container registry i.e. pre-built SageMaker Docker images. (image\_uris), framework has to be correct that will be “object-detection”
32. Creating estimator object model for training, here the instance type of gpu will cost. Training\_image is the image from ECR,

Role is the execution role

Instance type is the gpu

Volume size is the amount of volume needed in GB to store input and output

Instance count = 1, (this is bit confusing for now)

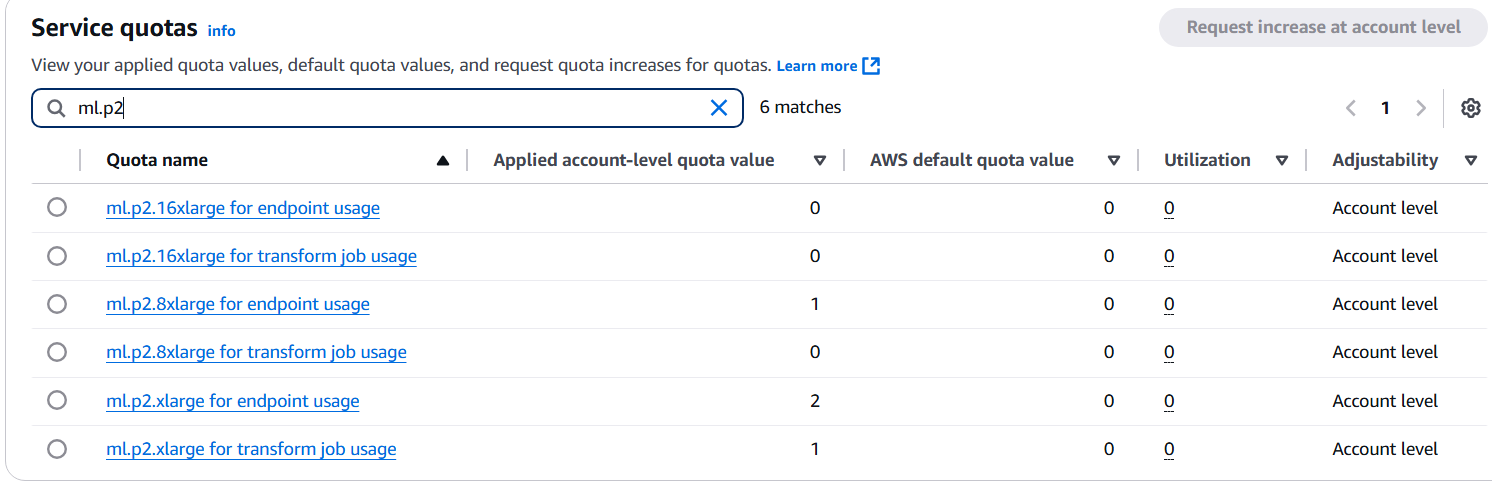
Max\_run in seconds after which it will run out

Input mode is file, that means it will copy our data from s3 into EFS in files

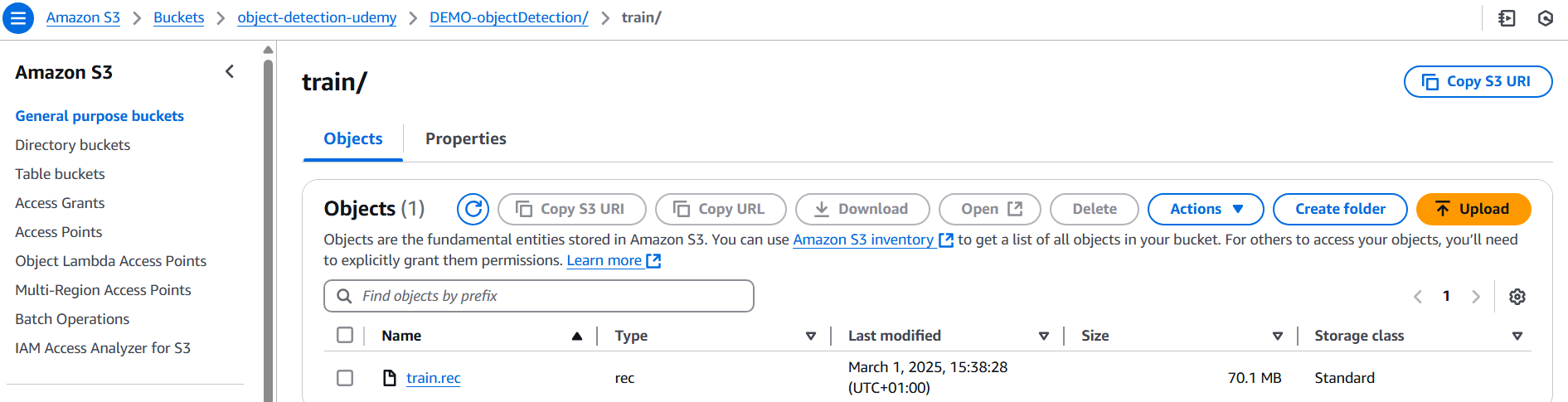
Output path is in the s3 path

1. Setting up hyperparameters, momentum is given for sgd optimizer and weight decay to overcome overfitting ( to not depend on one weigth), nms\_threshold (detection with low confidence avoid)
2. https://docs.aws.amazon.com/sagemaker/latest/dg/object-detection-api-config.html
3. Learning Rate scheduler
4. After setting hyperparameters we will set hyperparameter tuner which contains, mini\_batch\_size, optimizers and learning rate values
5. Set max\_jobs to 3 because then it will cost more than 20$
6. With object detection the meteric which you get is generall mean average precision for validation, <https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173>
7. Then giving the training inputs value
8. Object detection map value is not very higher is many cases because it is very tough to get the exact bounding box around the image as the actual bb, intersection over union thing
9. <https://cocodataset.org/#home>
10. Model deployment is again done with the help of sagemaker and for the best model, the S3 model uri is copied and used (this has the best hyperparameters), creating endpoint here
11. Endpoint also has to be shut down/delete once done.
12. The image for which detection has to be made, is to be sent in binary form to the enpoint
13. Keeping threshold is very important , because the higher value can avoid some plastic bags and the lower can detect everything as plastic bag
14. Setting up batch transformation job (Its like checking multiple images together)
15. Creating step function: step functions automate the AWS services and the ML pipelines and is an orchestration tool like airflow
16. Understand step function it is a workflow (states machine) for the whole pipeline, from start till end. Every step is a lambda function where the logic is written for executing the batch transformation. The ARN id of the lambda function is then associated to the step function. It’s a whole json file
17. This whole workflow is tested by uploading images to the batch transformation folder in S3 bucket.
18. Creating cron job for event bridge <https://docs.aws.amazon.com/eventbridge/latest/userguide/eb-scheduled-rule-pattern.html>
19. Setting up the production notebook: for creating bounding boz around the images for which json file output is generated
20. Notebook automation is done by creating notebook job
21. To increase service quota for GPU access: for errors like resourceexecesslimit
22. https://eu-central-1.console.aws.amazon.com/servicequotas/home/services/sagemaker/quotas

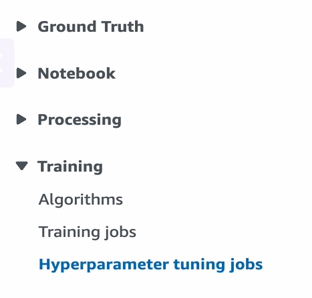


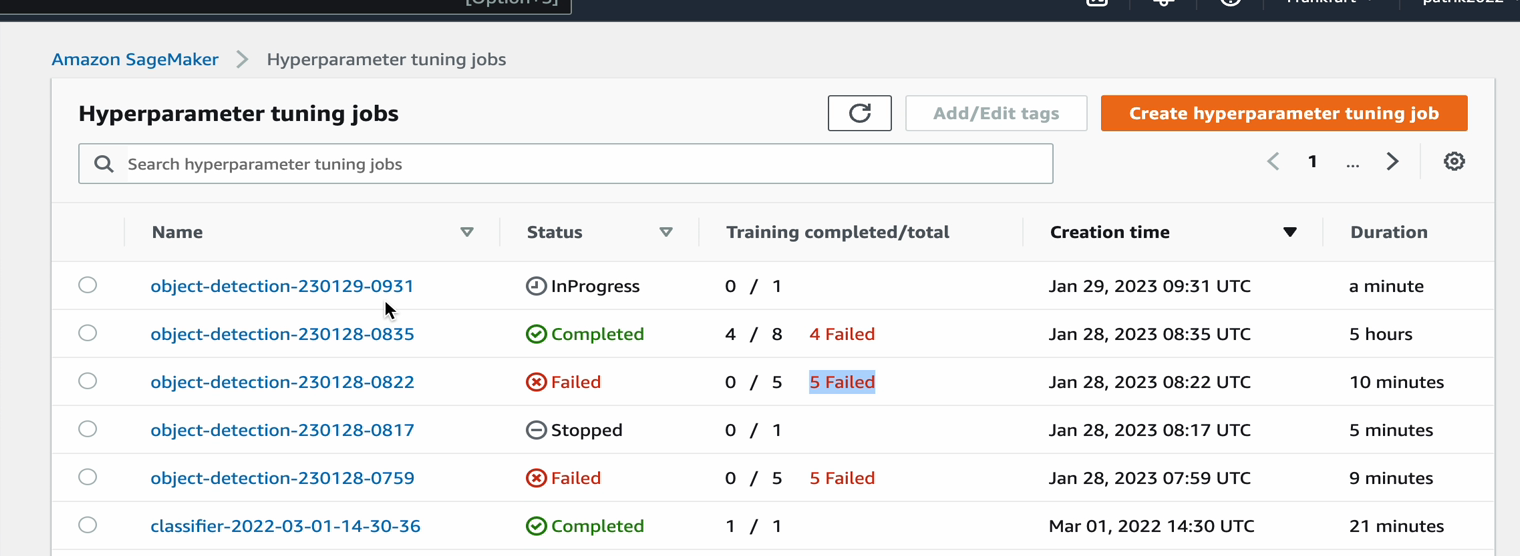


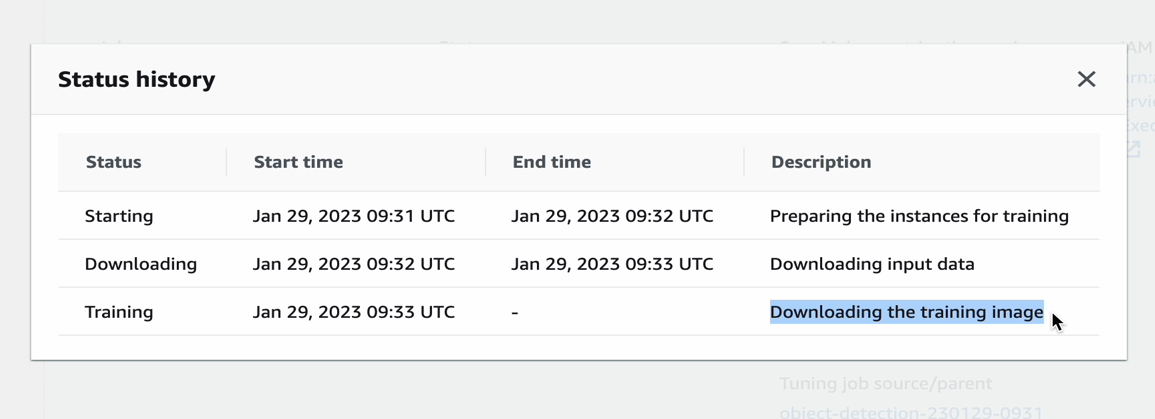
1. Uploaded the converted .rec files in S3 bucket:



1. In amazon sagemaker find training then hyperparameter tuning jobs



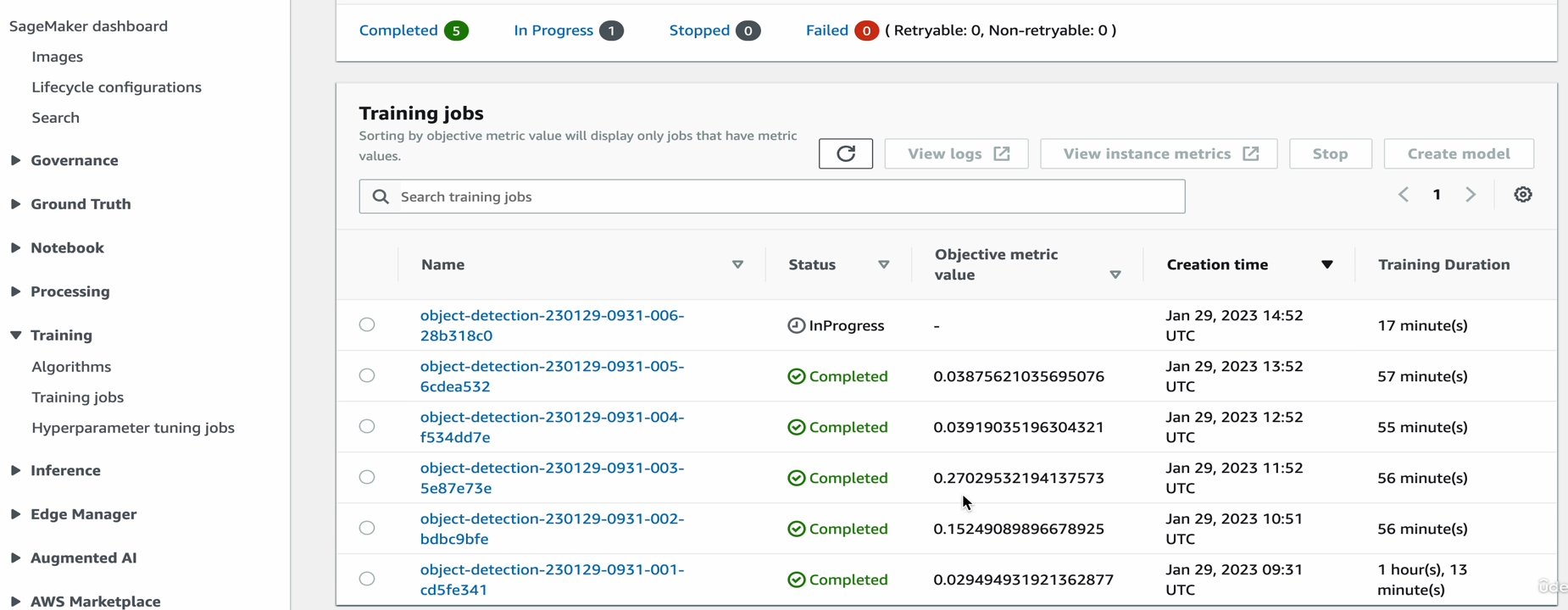


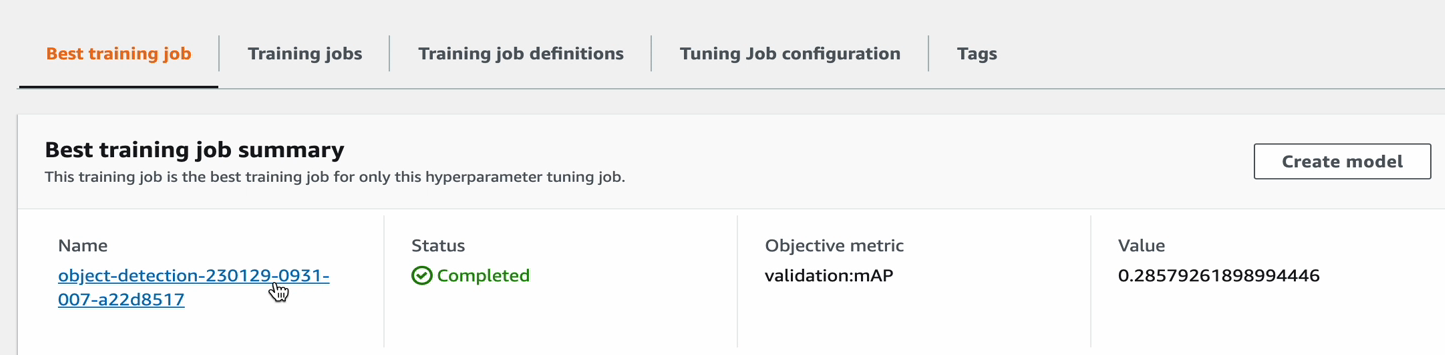


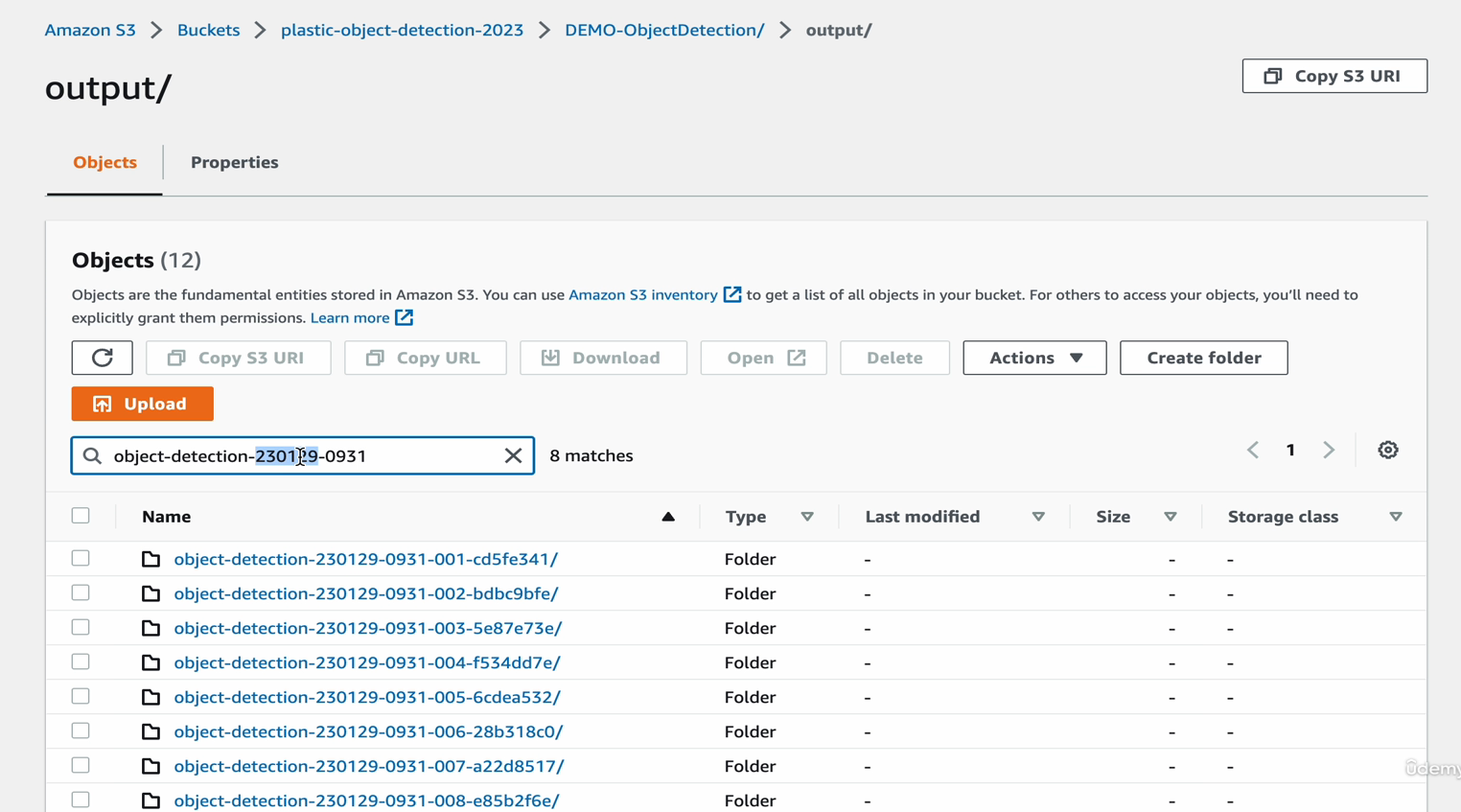
Analyse the training job: hyperparamter tuning job

Can see the logs in cloudwatch of this running training job

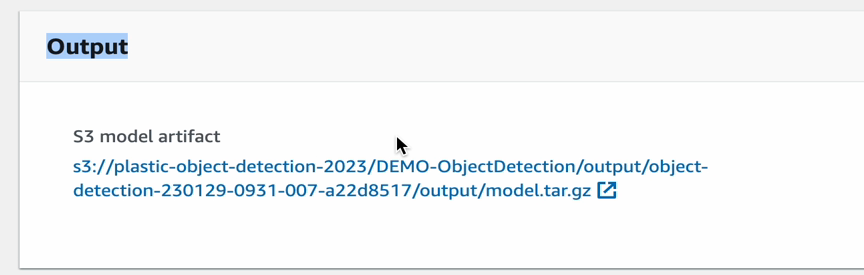
1. Map Score of 20 or 30% is also good with less images, it is different from accuracy
2. Monitoring the training job in AWS Sagemaker, view algorithmic metrics

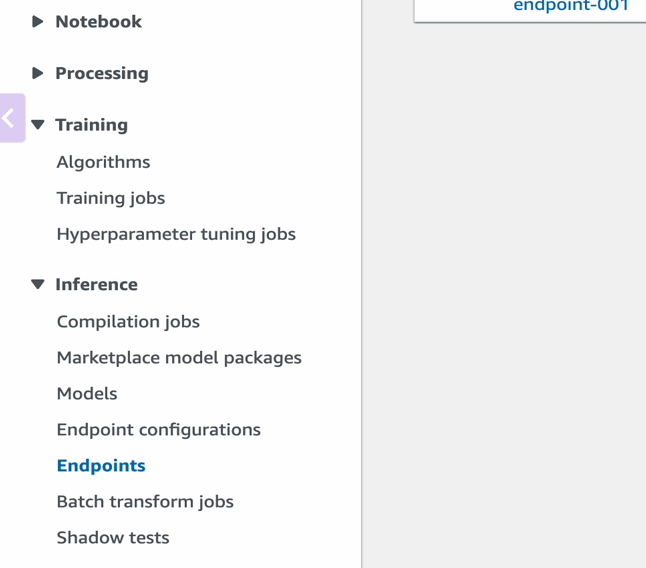
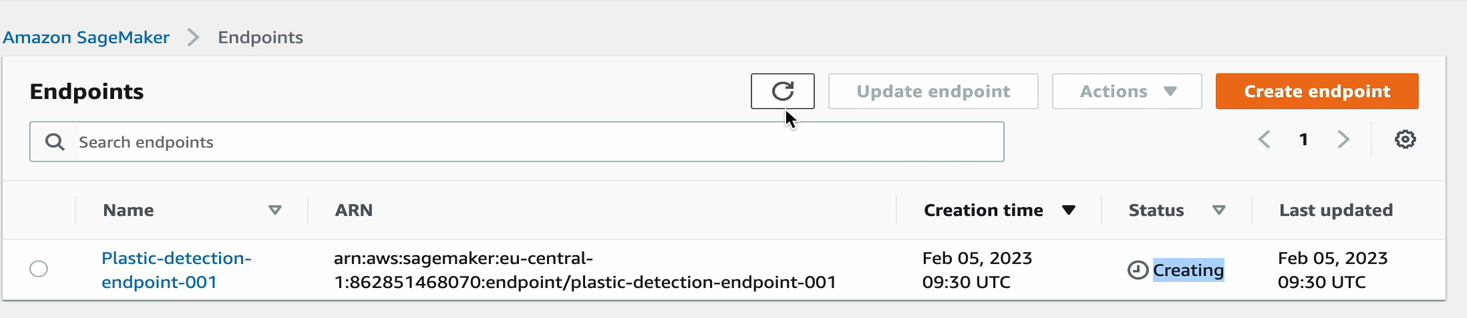
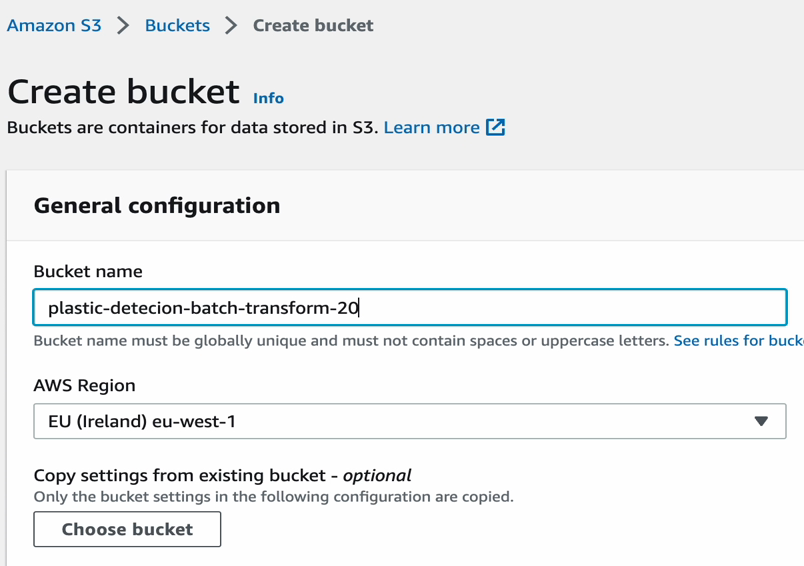
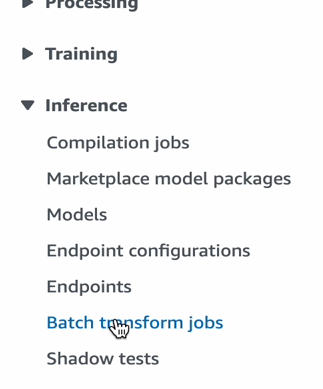
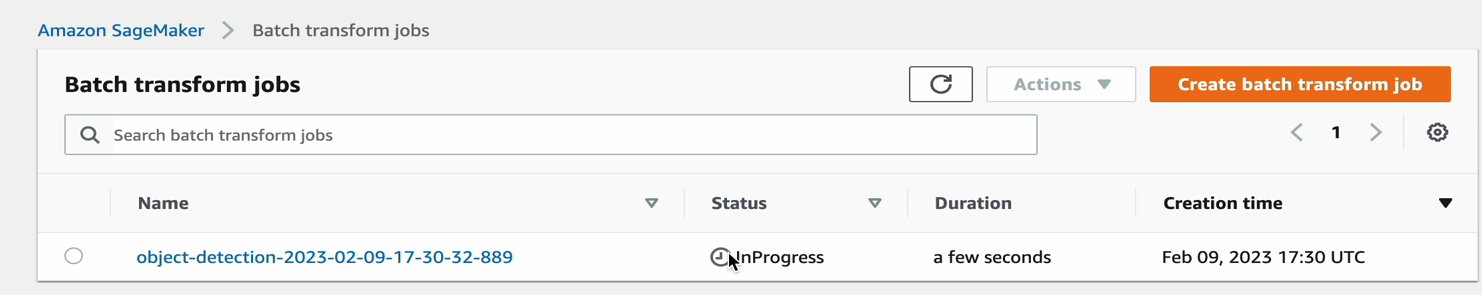
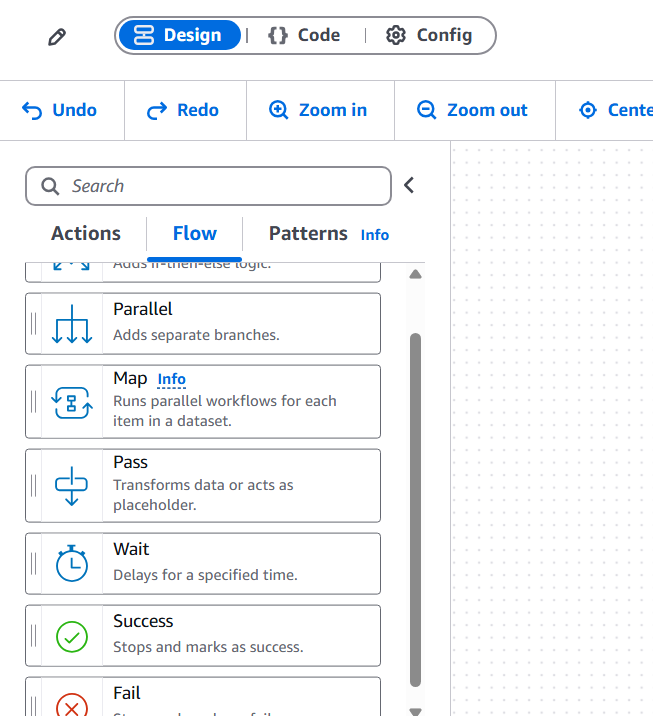
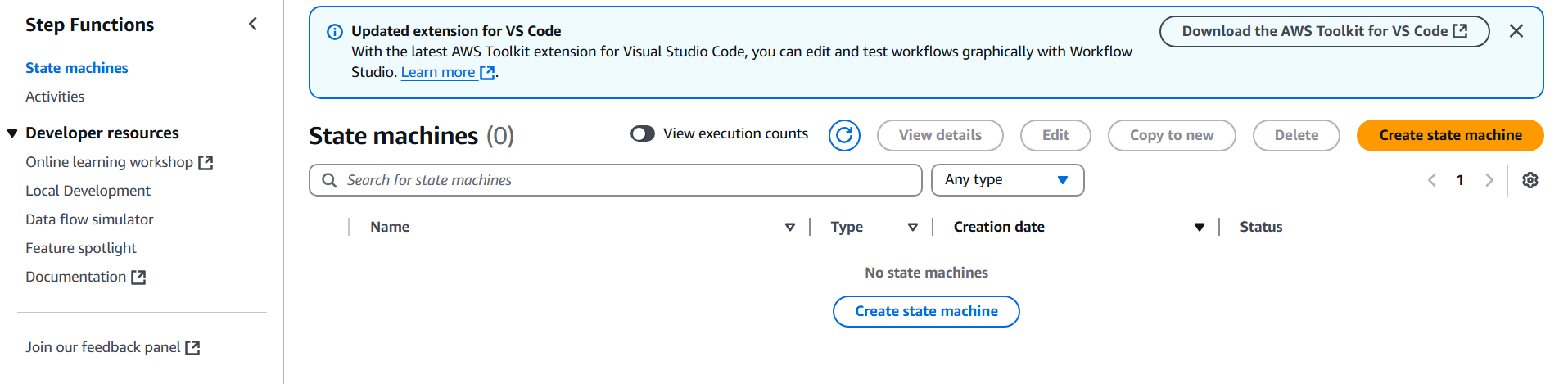
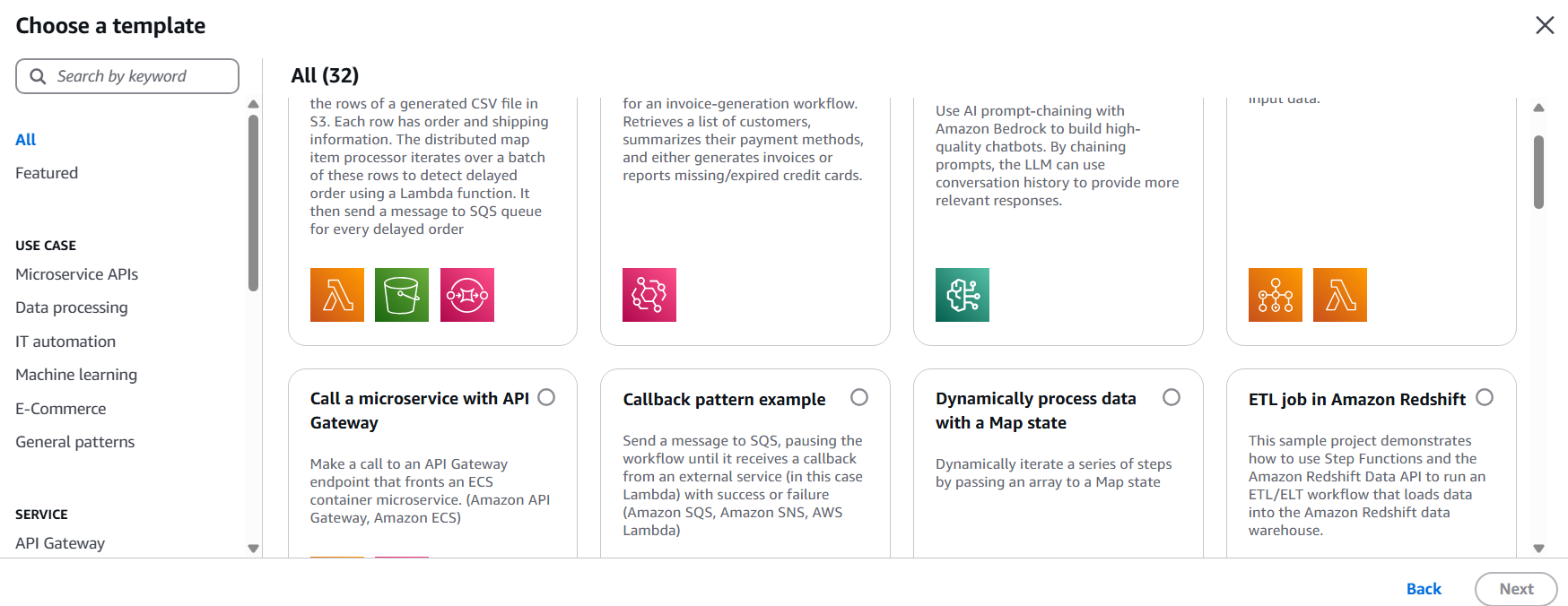
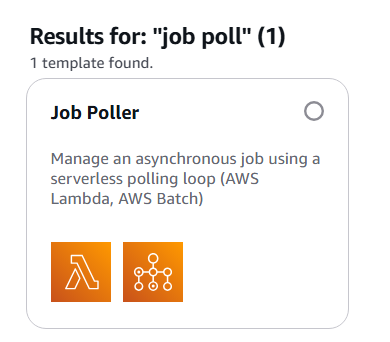
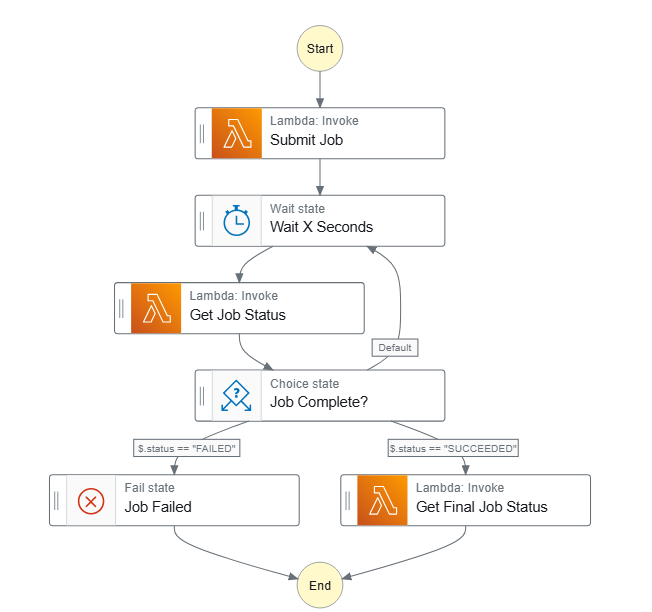
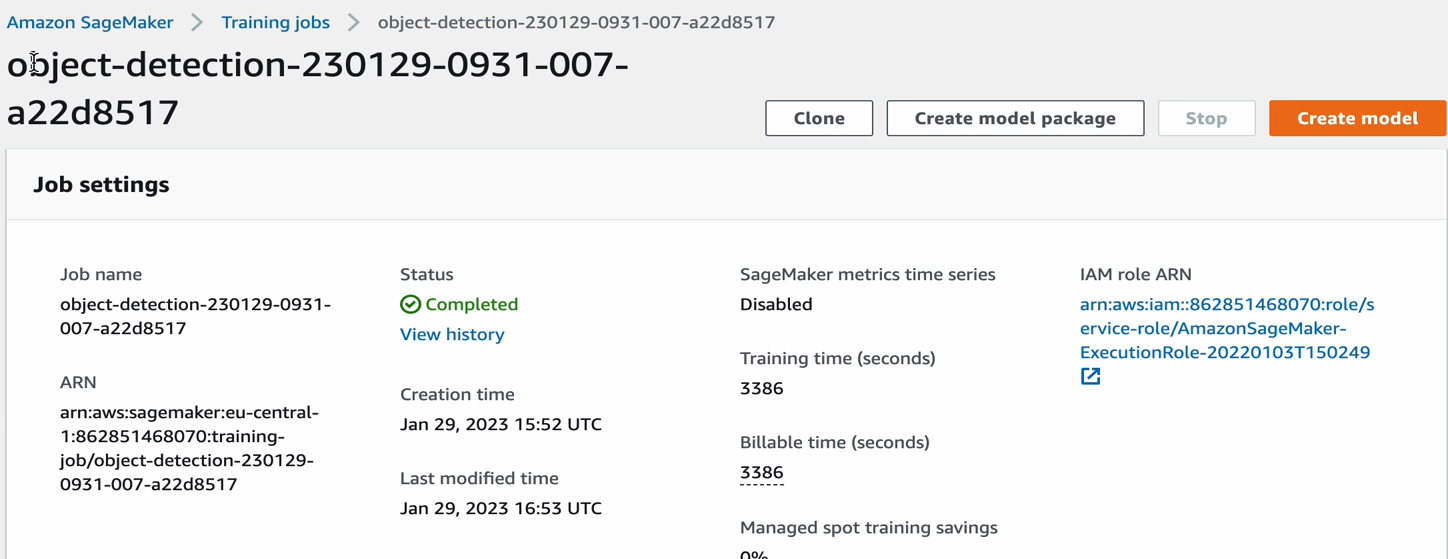
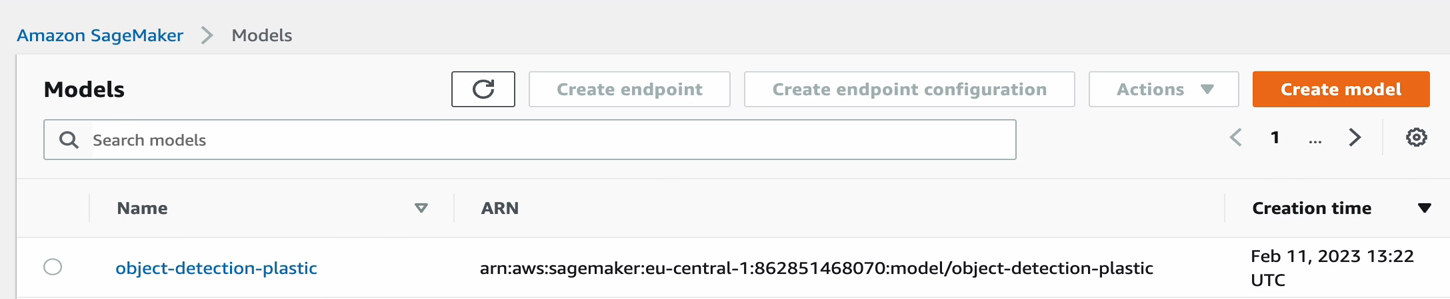
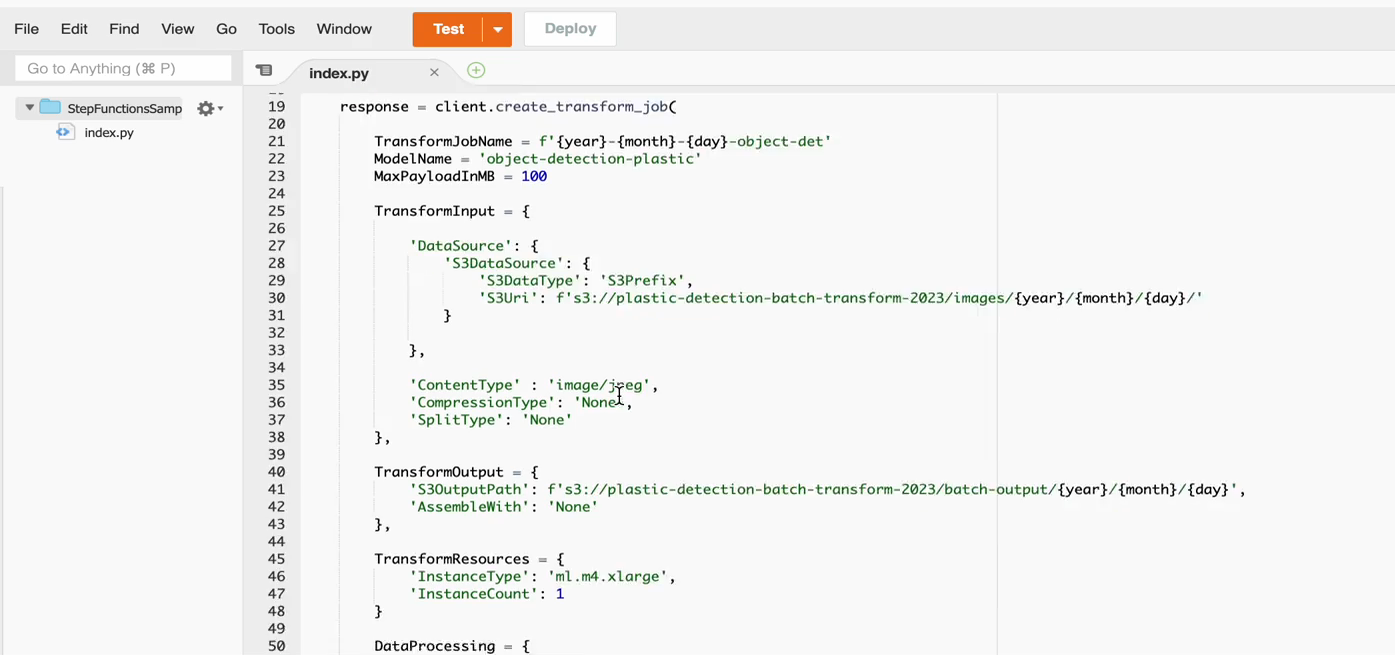
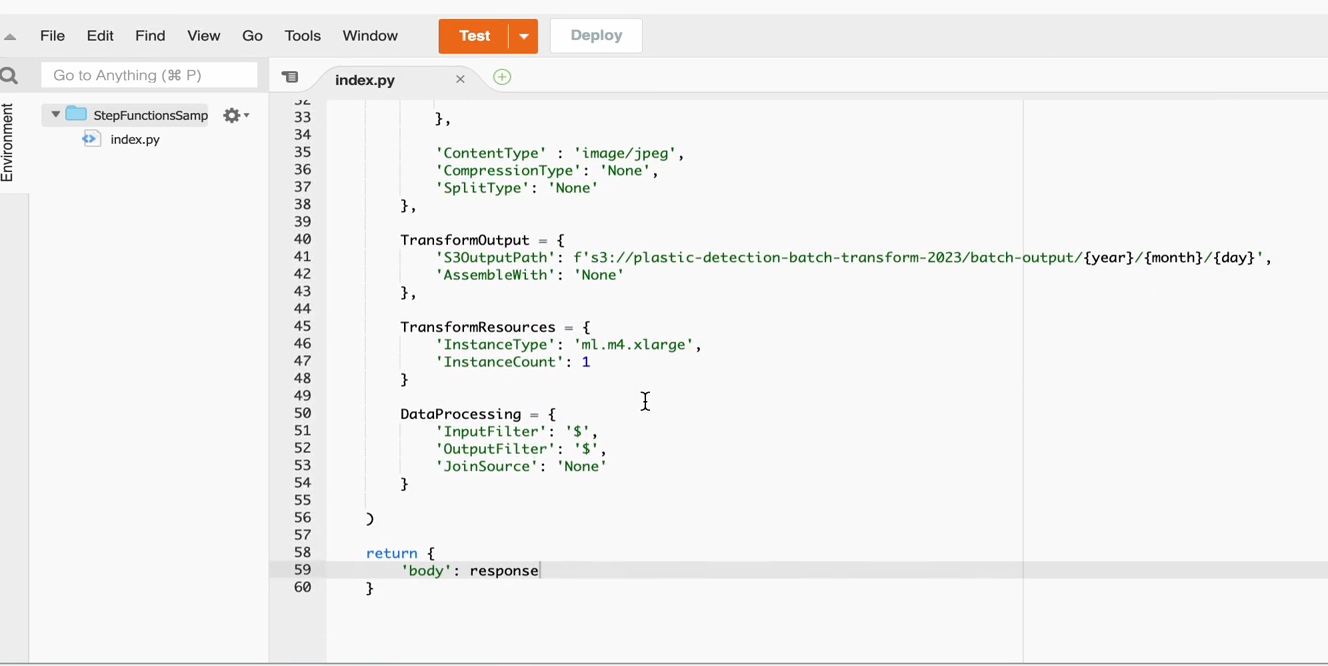
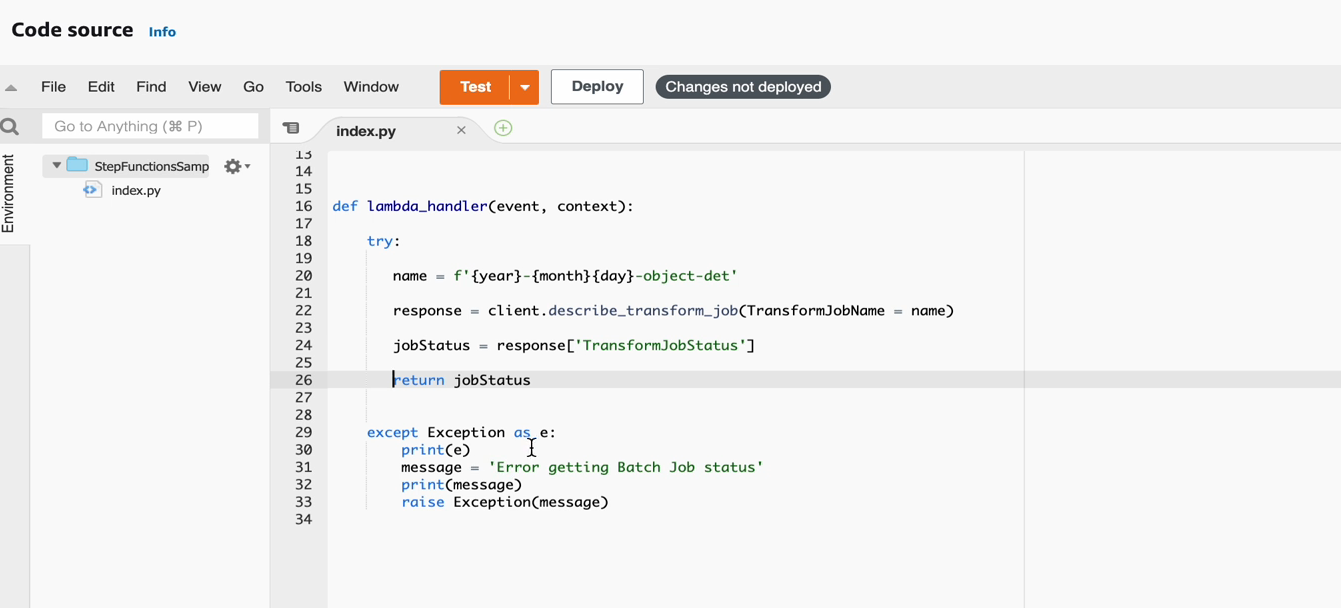
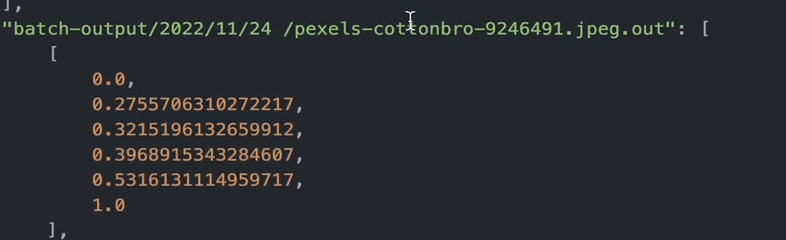
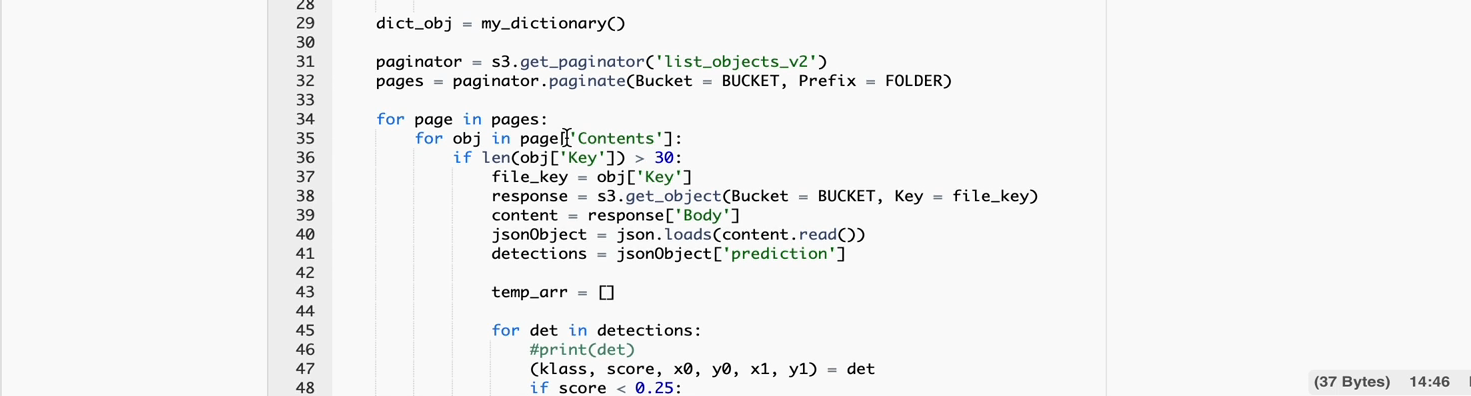
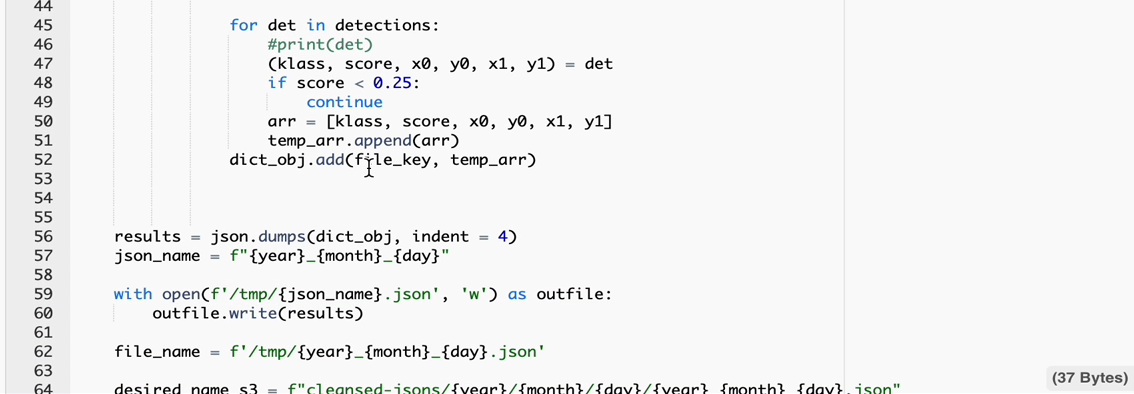
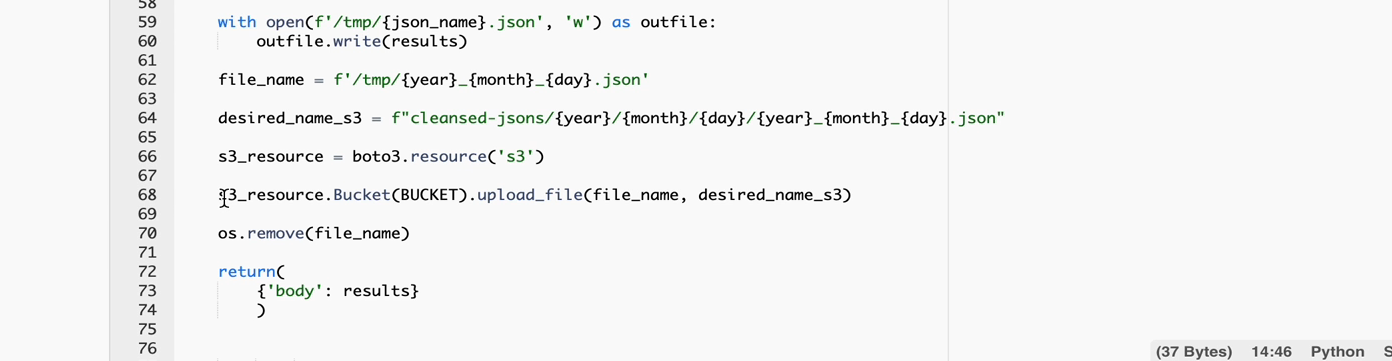
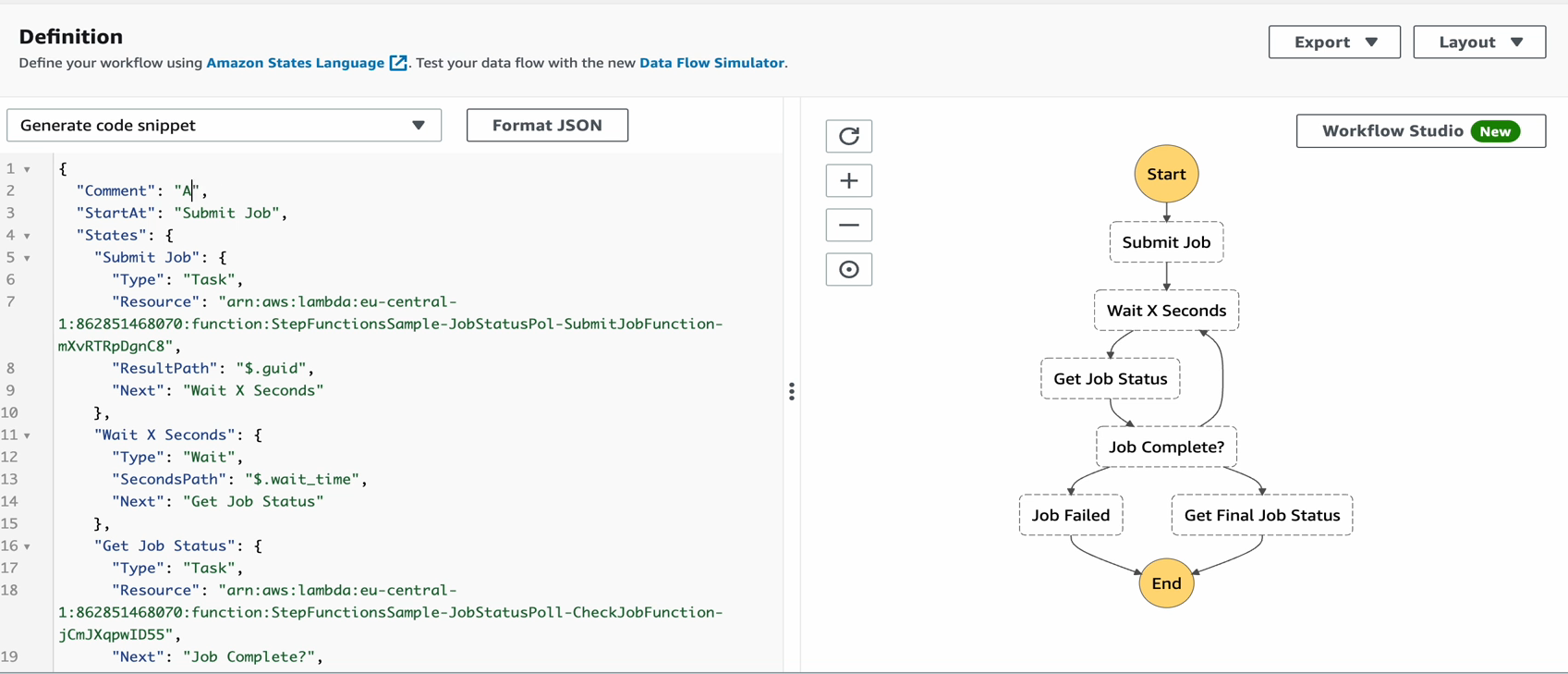
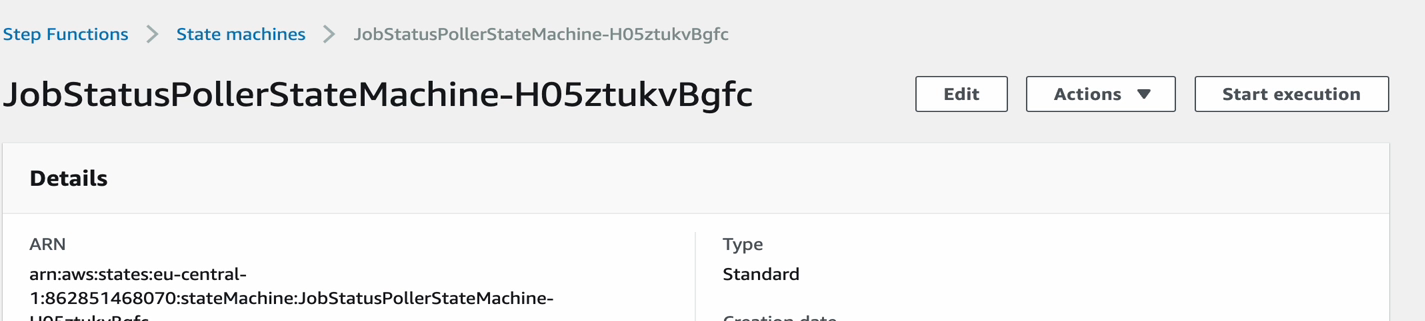


1. After complete training jobs, see the best training job to get the best hyperparameters:
2. 
3. These training jobs are saved in the output of the S3 bucket



1. Model is stored by clicking on this in the training job:



1. To check the model deployment running status, go to AWS Sagemaker, then endpoint
2. 
3. 
4. If endpoint does not return the inference then redeploy the model
5. Creating a bucket for batch transformation output files
6. 
7. Put the training images in the S3 bucket for input images of batch transformation (bucket should be in this same region as of the endpoint)
8. Go to Sagemaker and then inference, batch transform jobs
9. 
10. 
11. AWS Step functions used for running a workflow, ML pipeline, visually see the flow
12. Create a state machine, it’s a workflow which consists of tasks those are steps, you can use template
13. 
14. 
15. Choose a template to create a state machine
16. 
17. 
18. Choose job poller template
19. 
20. 
21. Lambda functions are automatically created for this step function template, we will have to modify it further
22. Configuring Lambda part 1
23. To use the best training job, first we have to create a model in AWS
24. 
25. Then while creating model, crean an iam role and give access policies or roles, as this will generate error if the model does not have specific roles/ policies
26. The model is created
27. 
28. Submit job code in lamda function (This code is similar to the transform job created in jupyter notebook, redirecting to the model created from best training job, images input S3 folder, images output S3 folder and some default type values)
29. 
30. 
31. Modifying Check Job function (This will basically check whther the transform job is a success or a fail)
32. 
33. Setting the Step function by adding one more task of Cleaning Batch output
34. The lambda function should also have administrator role (Go to configuration inside the lambda function then attach the permission policies)
35. Understanding the .out files generated after the job is completed
36. 
37. 0.0 => The class detected (plastic bag), 0.2755 => (how confident is the result, threshold), remaining 4 are the coordinates xmin, ymin, xmax, ymax
38. Code: clean output batch
39. 
40. 
41. 
42. 
43. Editing the step function json code
44. 
45. After the complete execution of step function we will create event bridge for automation (from actions button)
46. 
47. To productinize the notebook, click on create notebook job and set a cron job