National College of Ireland

Low-Latency Cold Start Optimization for Serverless Machine Learning Inference

MSc Research Project
MSc in Cloud Computing

Vyom Mehul Vora
Student ID: 23410698

School of Computing
National College of Ireland

Supervisor: Aqeel Kazmi

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Vyom Mehul Vora
Student ID:	23410698
Programme:	MSc in Cloud Computing
Year:	2025
Module:	MSc Research Project
Supervisor:	Aqeel Kazmi
Submission Due Date:	11/12/2025
Project Title:	Low-Latency Cold Start Optimization for Serverless Machine Learning Inference
Word Count:	1146
Page Count:	12

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Vyom Mehul Vora
Date:	10th December 2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Low-Latency Cold Start Optimization for Serverless Machine Learning Inference

Vyom Mehul Vora
23410698

1 Introduction

This guide provides details and steps on how to deploy machine learning models like DistilBERT, ResNet-50 and Random Forest Classifier as serverless functions on AWS, Azure and Google Cloud. The goal is to use to understand cold start behavior of these serverless functions and use the patterns to train logistic regression model and use it as a prewarming service next time before actual user requests comes in to invoke function and avoid cold start. Through this guide, you will know how to setup, generate data, train the model and use the model as a prewarm service.

Github Repo: The source code for this project is available at: `https://github.com/vyomvora/serverless-ml-coldstart`

2 Prerequisites and Environment Setup

2.1 System Dependencies

1. Python 3.9+
2. Windows: Docker Desktop or Linux: install Docker
3. AWS CLI
4. Azure CLI
5. Google Cloud SDK (gcloud)

2.2 Required Libaries

1. Cloud Platform SDK's:
   - boto3==1.28.0
   - azure-storage-blob==12.19.0
   - google-cloud-storage==2.14.0
2. Machine Learning model and Data Collection:
   - scikit-learn==1.3.2

- pandas==2.1.3
- numpy==1.24.3

3. DistilBERT and Resnet-50's frameworks:
   - transformers==4.35.0
   - torch==2.1.0
   - torchvision==0.16.0
4. Common Libraries:
   - requests==2.31.0
   - Pillow==10.1.0
   - python-dotenv==1.0.0

Save this as requirements.txt in your project's root folder.

3 Machine Learning Model Deployment

In this section, we will provide you with the project root to deploy your machine learning on each cloud. AWS Lambda uses Docker as deployment whereas Azure and GCP will have standard zip delpoyment.

3.1 DistilBERT, Resnet-50 and RandomForest Classifier

The entire setup is same for all three models on AWS Lambda. For Azure and GCP both are same way to setup but it is different than AWS Lambda but same for all the three models.

3.1.1 AWS Lambda, Azure Functions and Google Cloud Run Configuration

1. AWS Lambda Configuration
   - Go to aws folder and create 3 files: Dockerfile, lambda_function and requirements.txt
   - Create a Docker image with Dockerfile shown in 1 and push the image to AWS ECR.
   - Go to AWS Lambda, Create a function and choose deployment type as Container and select image that you pushed to AWS ECR from previous step and launch the function. Services (2025)
2. We are keeping Azure and GCP configuration in same section as the steps are common just the folder structure and deployment command is different.
   - Install Azure CLI and gcloud(Google Cloud CLI)
   - Azure commands to deploy the function:

- az functionapp create –resource-group serverless-rg –consumption-plan-location uksouth –runtime python –runtime-version 3.9 –functions-version 4 –name modelname –storage-account coldstartstorage
- az functionapp deployment source config-zip –resource-group serverless-rg –name modelname –src modelname.zipRetry Microsoft (2025)
- GCP commands to deploy the function:
- gcloud run deploy modelname –source=./modelname –region=us-central1 –allow-unauthenticated –memory=512Mi –timeout=120s –platform=managed Cloud (2025)

```
FROM public.ecr.aws/lambda/python:3.13

WORKDIR ${LAMBDA_TASK_ROOT}

COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

COPY lambda_function.py .

CMD ["lambda_function.lambda_handler"]
```

Figure 1: Dockerfile for AWS Lambda

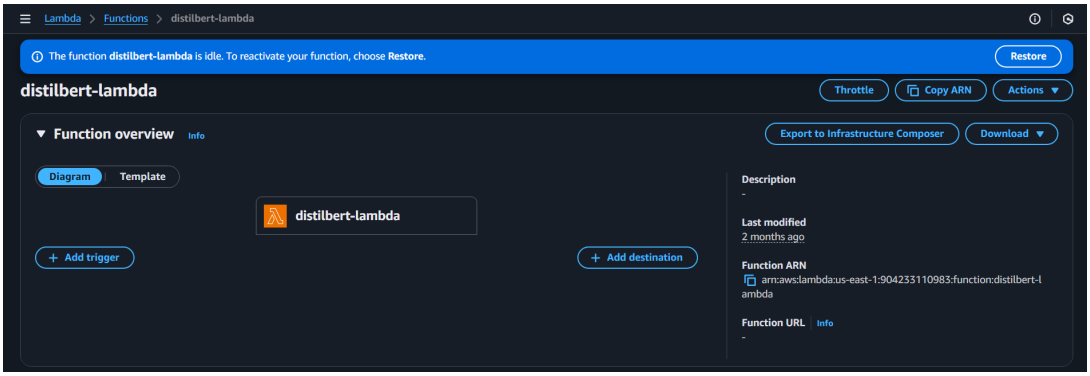

Figure 2: DistiBERT Function on AWS Lambda

4 Data Generation

4.1 Azure VM

1. Create a Azure VM where all the traffic generation scripts will run and download the .pem file.

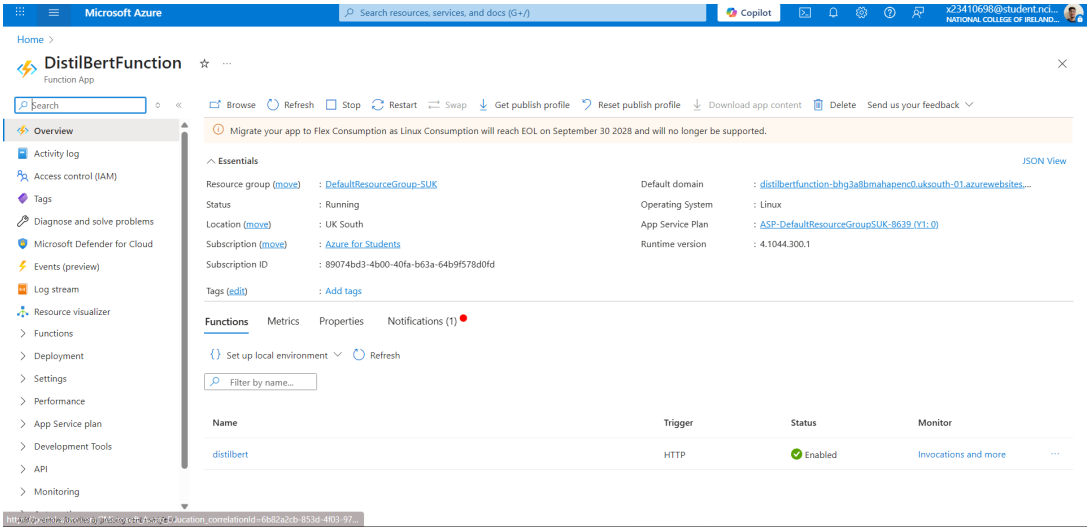

Figure 3: DistiBERT Function on Azure Functions

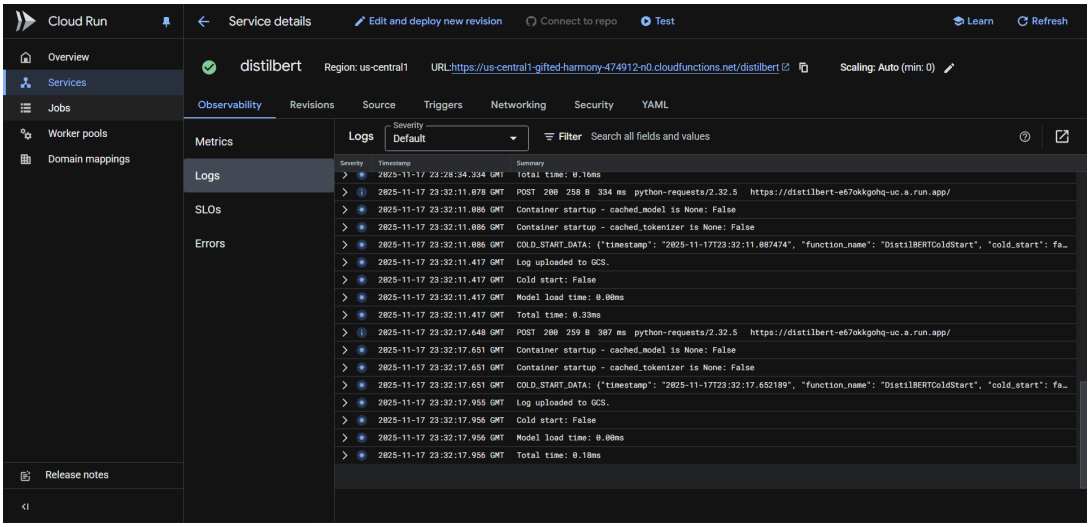

Figure 4: DistiBERT Function on GCP Cloud Run

2. Install python 3.9+ in the VM.

3. Create python environment and install the requirements.txt that is required for running all the python scripts.

4. Copy all the cronjob(traffic generation) file from your local device to Azure's VM using: **sudo scp -i vmkey.pem ./traffic_generation_scripts.py azureuser@20.55.123.10:/ho**

5. Install screen on Azure's VM using command :**sudo apt install screen**. It is used for keeping the process running even if you disconnect from the VM

4.2 Traffic Generation Scripts

1. There are three files for traffic generation scripts as shown in 11

2. Make sure they are all copied to the Azure's VM as mentioned in 4.1

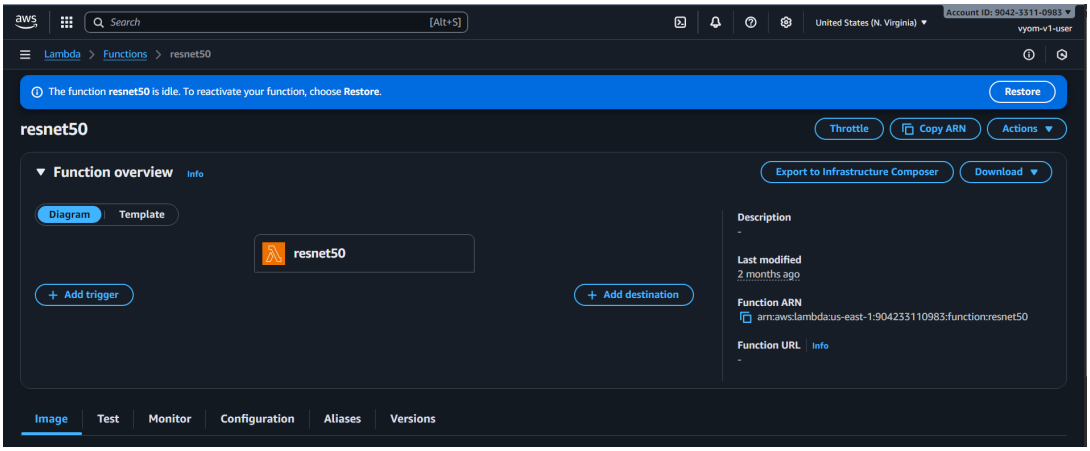

Figure 5: Resnet50 Function on AWS Lambda

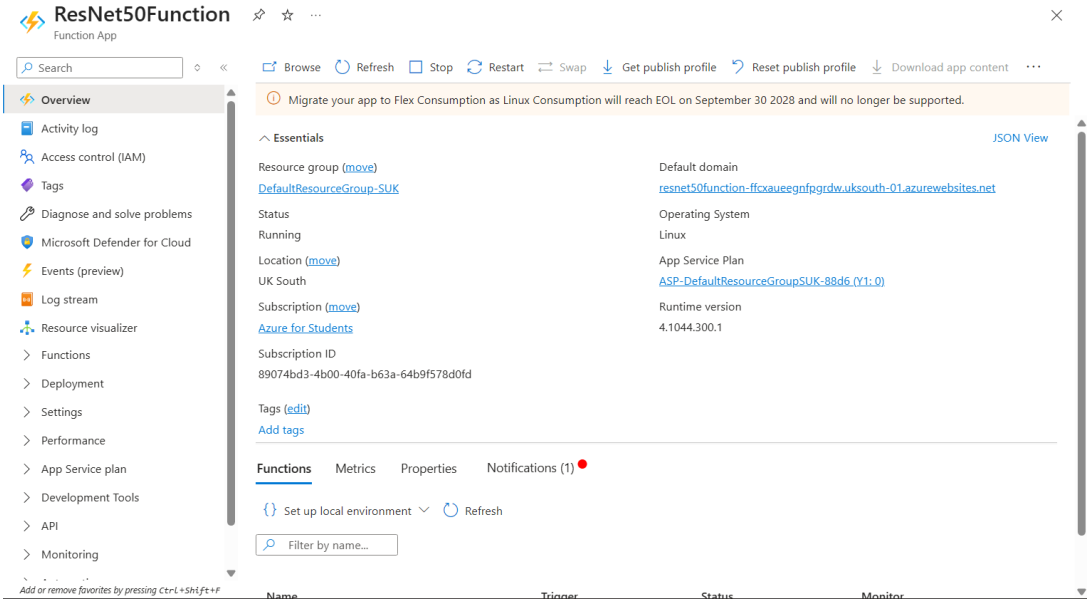

Figure 6: Resnet50 Function on Azure Functions

3. For each platform specific traffic script we follow these steps:
   - screen -S cloud_platform_name
   - python cronjob_lambda.py or cronjob_azure.py or cronjob_gcp.py
   - CTRL + A and press D
4. Now the traffic script will keep on running without any stoppage even if terminal disconnected.
5. Each script will store logs to cloud storages AWS S3 for Lambda, Azure Blob Storage for Azure and Google Cloud Storage for Cloud Run as show in 13 14 15

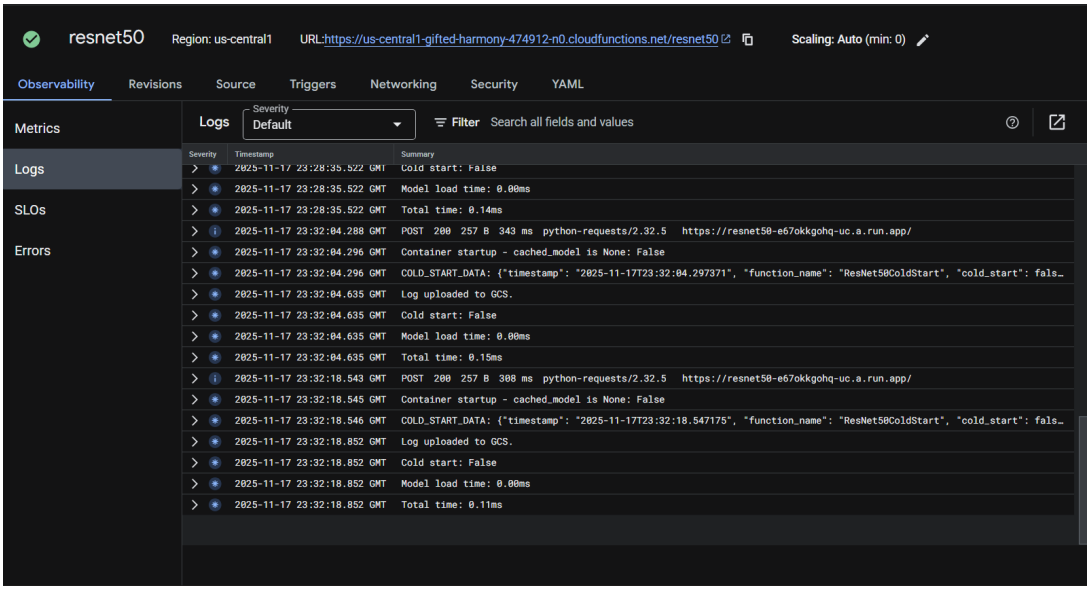

Figure 7: Resnet50 Function on Google Cloud Run

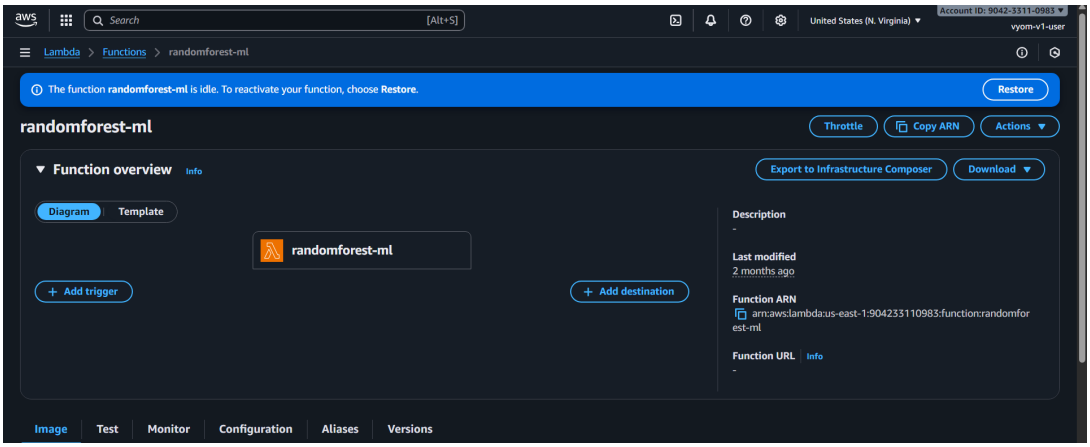

Figure 8: Random Forest Classifier Function on AWS Lambda

5 Model Training Setup

All the feature engineering and model training was done on local device itself. No need of Azure VM for this.

5.1 Transform Training Data

1. Go to folder extract_feature_from_logs and there are 3 files for each cloud platform.
2. Just do: python aws_logs.py or azure_logs.py or gcp_logs.py
3. It will save a new transformed csv that can be used further for model training as shown in

5.2 Training Prediction Model

1. Parameters used for training:

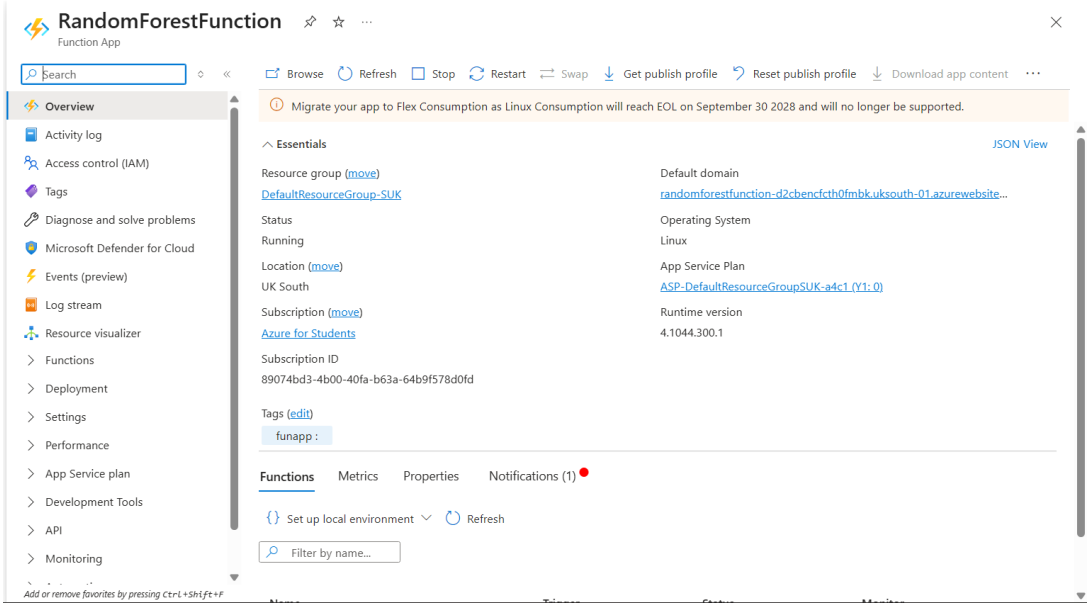

Figure 9: Random Forest Classifier Function on Azure Functions

- Train-test split: 70-30
- Model: Logistic Regression(max_iter=2000, class_weight='balanced')
- Threshold: 0.10(10%)

2. Save the model in a directory as lr_model.pkl
3. Save the model's encoder file as label_encoders.pkl

6 Prediction Service Deployment

6.1 Prediction Service Setup

- We will use the same Azure VM that we used for traffic generation script.
- Copy model pickle and model encoder files from your local device to Azure's VM using **sudo scp -i vmkey.pem lr_model.pkl azureuser@20.55.123.10:/home/azureuser/**

6.2 Baseline traffic script

- Now we need a traffic script on which our prediction service will run against. So we will create a dummy traffic script that invokes function for atleast 30 minutes.
- This traffic script needs to be copied from local device to Azure's VM and this script basically calls all the cloud platform serverless functions in once.
- Create a new screen as shown in section 4.2 and name it "Baseline sript"
- Once you are in the screen just do python baseline_script.py and keep it running. This script is to send dummy traffic on which model can make decisions to prewarm.

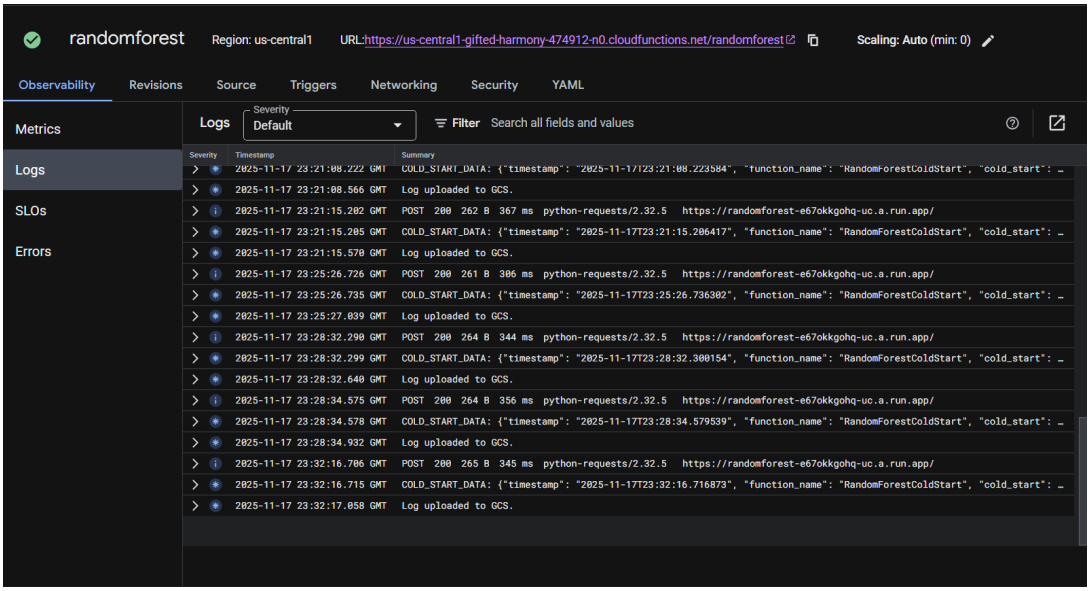

Figure 10: Random Forest Classifier Function on Google Cloud Run

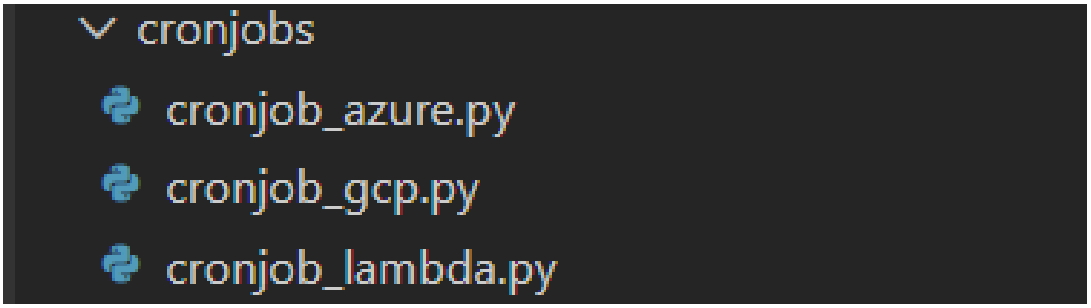

Figure 11: Traffic Generation Scripts for all cloud platforms

6.3 Baseline traffic script

- Now we need a traffic script on which our prediction service will run against. So we will create a dummy traffic script that invokes function for atleast 30 minutes.
- This traffic script needs to be copied from local device to Azure's VM and this script basically calls all the cloud platform serverless functions in once.
- Create a new screen as shown in section 4.2 and name it "baseline"
- Once you are in the screen just do python baseline_script.py and keep it running. This script is to send dummy traffic on which model can make decisions to prewarm.
- This script will store logs in the platform specific cloud storage.

6.4 Prewarm Prediction script

- This script uses logistic regression model that we trained earlier to make predictions.
- Copy this script from local device to Azure's VM.
- Create a new screen as shown in section 4.2 and name it "prewarm_prediction"

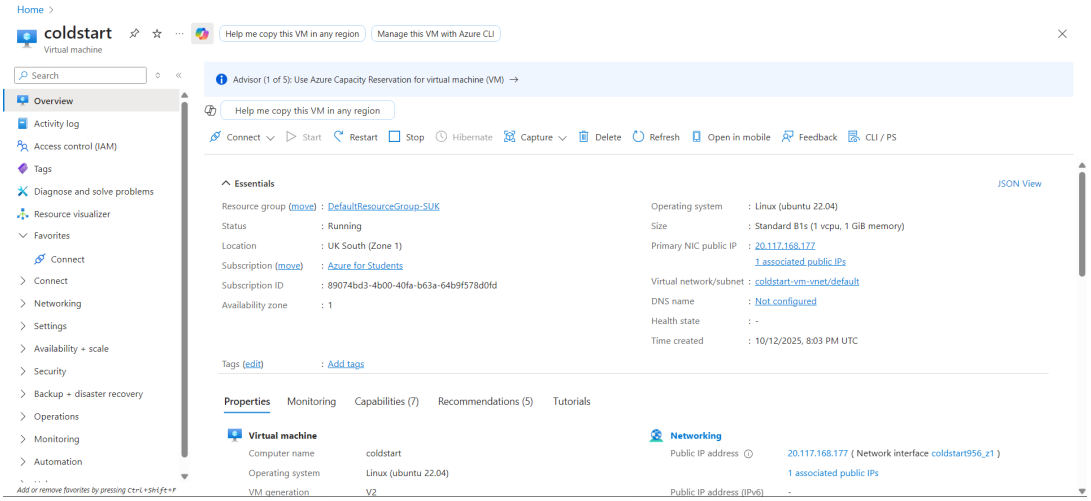

Figure 12: Virtual Machine Created in Azure

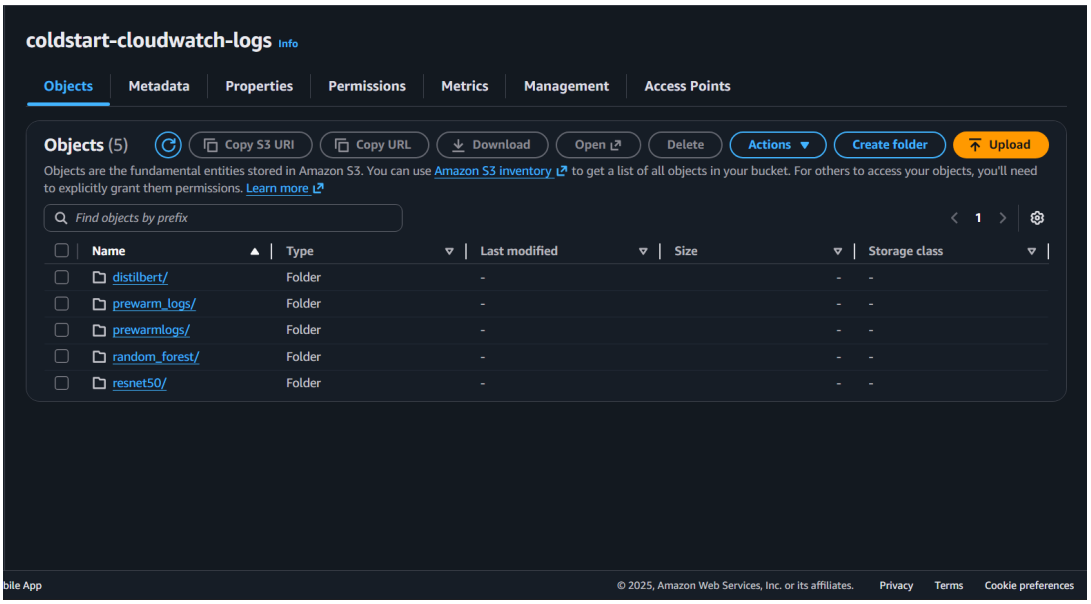

Figure 13: Traffic Script logs in AWS S3

- Once you are in the screen just do python prewarm_script.py and keep it running. This script qill load model, check logs that were saved by baseline script and makes prewarming decisions
- This script will store decision logs in the platform specific cloud storag which can be used for analysis afterwards.

References

Cloud, G. (2025). Cloud run documentation, `https://cloud.google.com/run/docs`. Published on 20 September.

Microsoft (2025). Azure functions documentation, `https://docs.microsoft.com/azure/azure-functions/`. Published on 20 September.

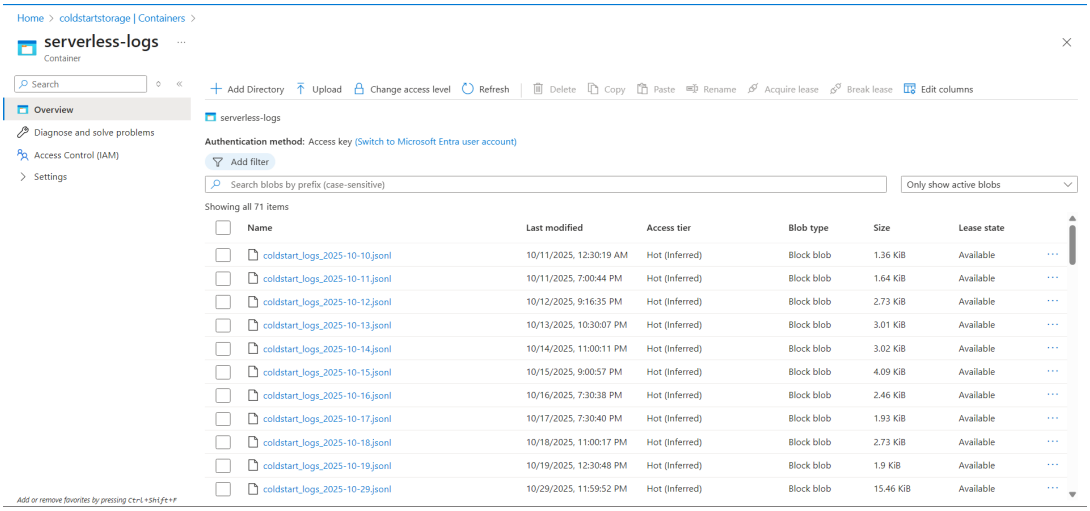

Figure 14: Traffic Script logs in Azure Blob Storage

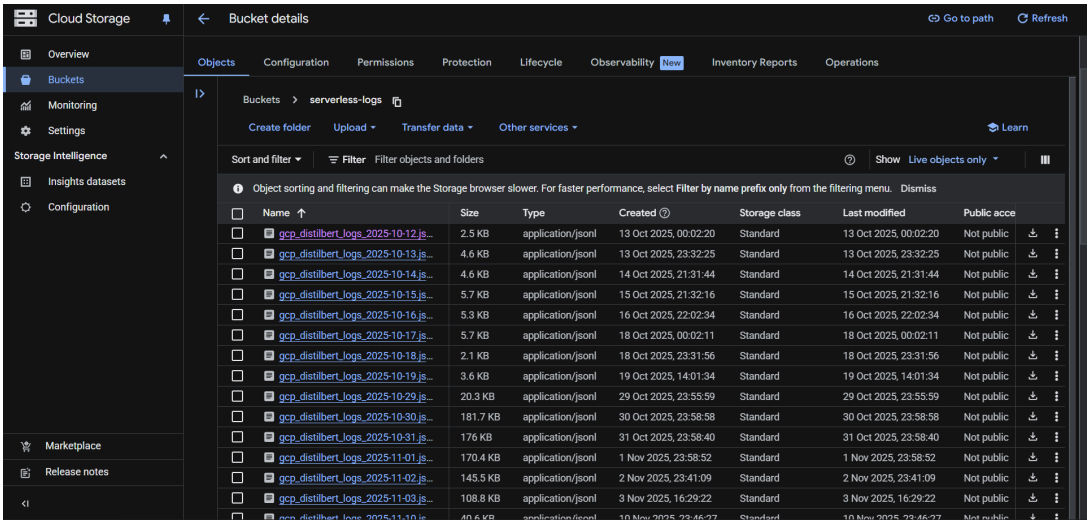

Figure 15: Traffic Script logs in GCP Cloud Storage

Services, A. W. (2025). Aws lambda documentation, `https://docs.aws.amazon.com/lambda`.

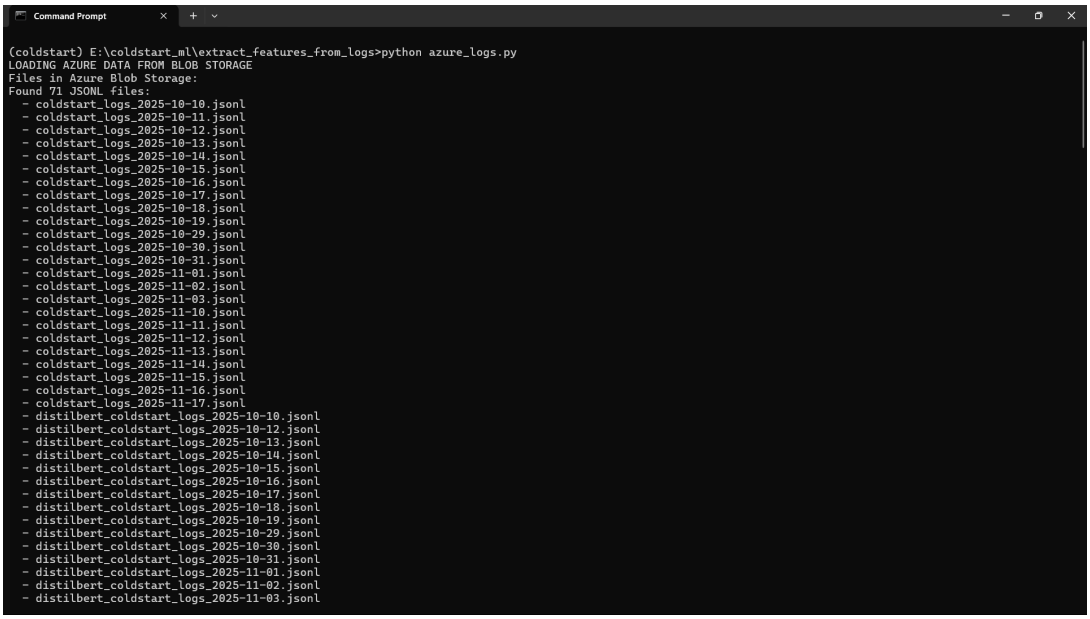

Figure 16: Downloading and Transforming Logs

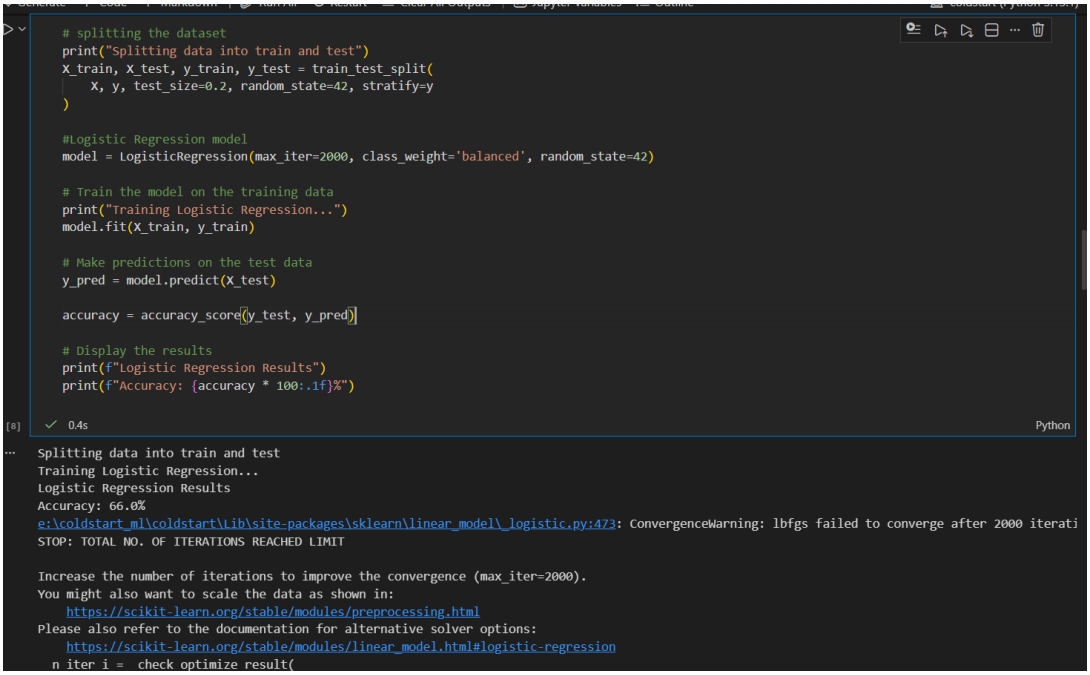

Figure 17: Logistic Regression Training

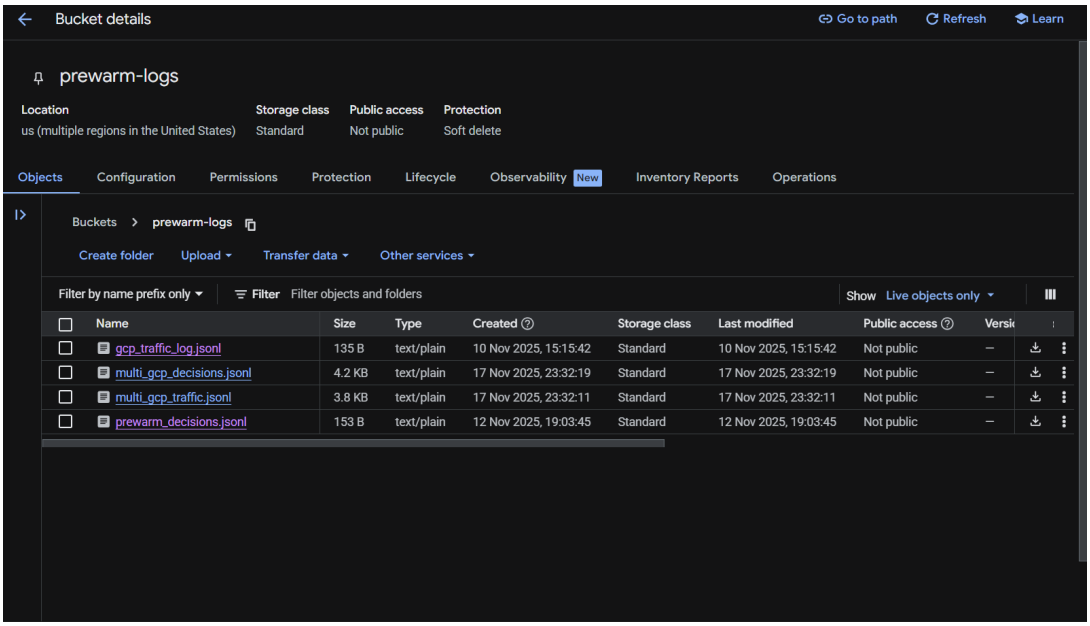

Figure 18: Baseline and Prewarm Decisions Stored