AI-Powered Data Pipeline

for E-commerce Warehouse Demand Forecasting

Implementation Steps

Data Ingestion Using Terraform and Python

Data Processing Using AWS Glue & Lambda

Machine Learning with Amazon SageMaker (for demand forecasting)

Model Integration & Optimisation

**Deployment & Scaling**

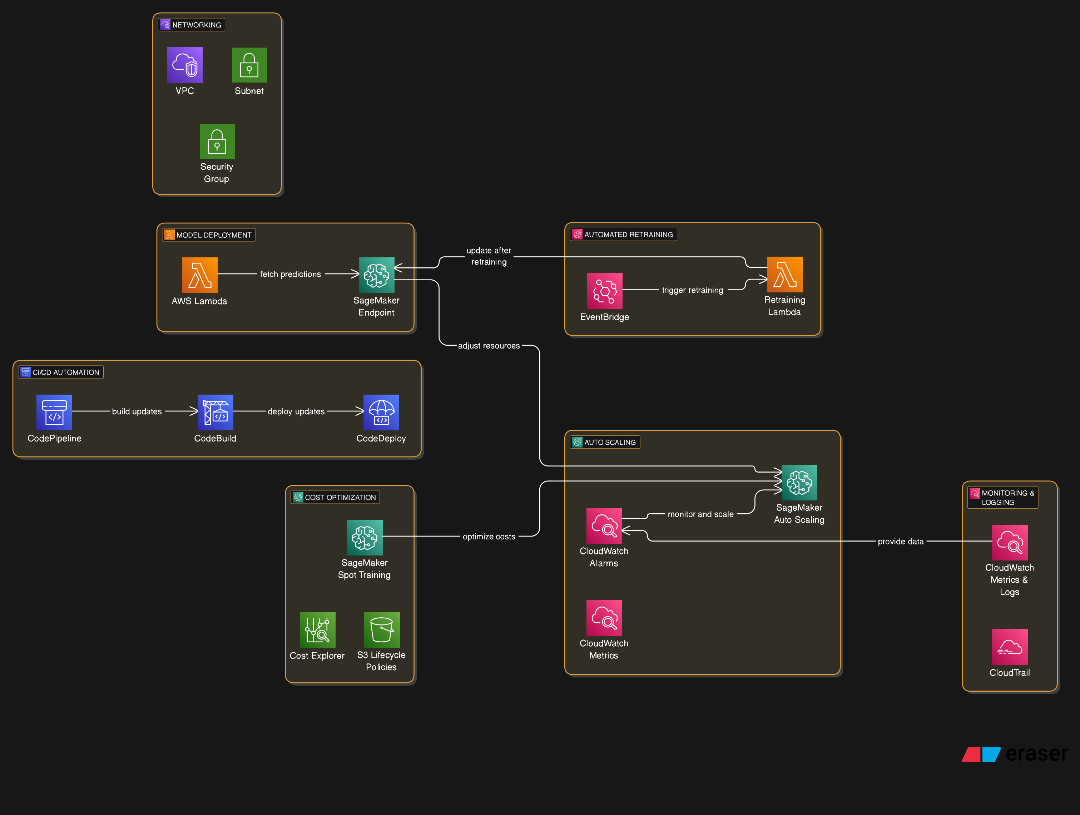
****

Table of Contents

[Implementation Steps 1](#_Toc192961601)

[Step 5: Deployment & Scaling 3](#_Toc192961602)

[Objective: 3](#_Toc192961603)

[Deploy the AWS Lambda Function to Query SageMaker Endpoint 3](#_Toc192961604)

[Automate Deployments with AWS CodePipeline 6](#_Toc192961605)

[Implement Auto Scaling for SageMaker 6](#_Toc192961606)

[Automate Model Retraining with EventBridge 6](#_Toc192961607)

[Optimise Costs & Enable Monitoring 7](#_Toc192961608)

[Lessons Learned & Next Steps 7](#_Toc192961609)

[Lessons Learned 7](#_Toc192961610)

[Next Steps 7](#_Toc192961611)

# Step 5: Deployment & Scaling

## Objective:

* Deploying the Lambda function to query the SageMaker model.
* Automating deployments with CI/CD using AWS CodePipeline.
* Implementing Auto Scaling for SageMaker to handle variable workloads.
* Automating model retraining using EventBridge and SageMaker training jobs.
* Optimizing costs using Spot Training and S3 lifecycle policies.
* Enabling monitoring and logging with CloudWatch & CloudTrail.

## Deploy the AWS Lambda Function to Query SageMaker Endpoint

This step ensures that the Lambda function is deployed and correctly fetches predictions from the SageMaker Endpoint.

1. Navigate to the AWS Lambda Console → open the AWS Lambda console: Lambda Console → find the function: trigger\_sagemaker\_prediction or function responsible for querying the SageMaker endpoint.

A screenshot of a computer

AI-generated content may be incorrect.

1. Go to IAM Console: IAM Roles - find the Lambda execution role and ensure it has the following policy permissions

A screenshot of a computer

AI-generated content may be incorrect.

1. Deploy the Lambda code: click on code and replace the code with the following:

*import json*

*import boto3*

*import logging*

*import time*

*logger = logging.getLogger()*

*logger.setLevel(logging.INFO)*

*sagemaker\_runtime = boto3.client('sagemaker-runtime')*

*SAGEMAKER\_ENDPOINT = "warehouse-demand-endpoint"*

*MAX\_RETRIES = 3*

*def lambda\_handler(event, context):*

*try:*

*input\_data = event.get("data", None)*

*if not input\_data:*

*raise ValueError("Missing input data for inference."*

*payload = json.dumps(input\_data)*

*response = invoke\_sagemaker\_with\_retries(payload)*

*prediction = json.loads(response['Body'].read().decode())*

*logger.info(f"SageMaker response: {prediction}")*

*return {"statusCode": 200, "body": json.dumps({"prediction": prediction})*

*except Exception as e:*

*logger.error(f"Error invoking SageMaker endpoint: {str(e)}")*

*return {"statusCode": 500, "body": json.dumps({"error": str(e)})}*

*def invoke\_sagemaker\_with\_retries(payload):*

*retries = 0*

*while retries < MAX\_RETRIES:*

*try:*

*response = sagemaker\_runtime.invoke\_endpoint(*

*EndpointName=SAGEMAKER\_ENDPOINT,*

*ContentType="application/json",*

*Body=payload*

*)*

*return response*

*except Exception as e:*

*logger.warning(f"SageMaker invocation failed: {e}")*

*retries += 1*

*time.sleep(2 \*\* retries)*

*raise Exception(f"Failed to fetch prediction after {MAX\_RETRIES} attempts")*

Test the Lambda function: Create a new test event using a simple payload and verify the SageMaker response*.*

A screenshot of a computer

AI-generated content may be incorrect.

## Automate Deployments with AWS CodePipeline

This step ensures that SageMaker model updates & Lambda deployments are automated.

1. Create an S3 bucket for Artifacts in the CLI

*aws s3 mb s3://your-artifact-bucket*

1. Run the following command to create the pipeline

*aws codepipeline create-pipeline --cli-input-json file://pipeline-definition.json*

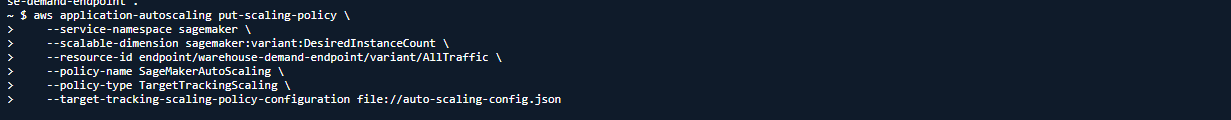
## Implement Auto Scaling for SageMaker

1. Enable SageMaker auto scaling

A screen shot of a computer

AI-generated content may be incorrect.

1. Attach the Auto Scaling policy



## Automate Model Retraining with EventBridge

1. Create an EventBridge rule Trigger Retraining

*aws events put-rule --name SageMakerRetraining --schedule-expression "rate(7 days)"*

1. Attach Lambda Function to EventBridge Rule

*aws events put-targets --rule SageMakerRetraining --targets* [*file://lambda-target.json*](file://lambda-target.json)

## Optimise Costs & Enable Monitoring

1. Enable spot training for SageMaker

A blue background with white text

AI-generated content may be incorrect.

1. Enable CloudWatch Metrics for Monitoring

A screen shot of a computer

AI-generated content may be incorrect.

# Lessons Learned & Next Steps

## Lessons Learned

1. **Cloud Cost Awareness:** Managing cloud costs is a crucial skill, this project highlighted the importance of budgeting and AWS cost optimisation strategies.
2. **Infrastructure as Code (IaC):** Using Terraform for resource provisioning improves scalability and maintainability.
3. **Automated Pipelines:** Automating ML deployment via CI/CD ensures model updates can be seamlessly integrated into production.
4. **Security & Permissions:** Correctly setting IAM roles and policies is critical to ensuring smooth interaction between AWS services.
5. **Scalability Planning:** Understanding how to scale SageMaker and Lambda functions based on demand improves system efficiency.

## Next Steps

1. **Complete the Deployment:** Implement the missing AWS services and finalise infrastructure.
2. **Conduct Performance Testing:** Validate response times and model efficiency under load.
3. **Integrate Additional Monitoring:** Enhance observability using AWS CloudWatch Logs, Metrics, and Alarms.
4. **Optimise Costs Further:** Leverage AWS Savings Plans and Reserved Instances for long-term cost reduction.
5. **Expand the Solution:** Extend the pipeline to handle multiple warehouses and integrate with external APIs for broader use cases.