1 Introduction

This report presents a comprehensive AWS-based MLOps architecture for deploying and managing ****** Limited's suite of retail AI models. The design leverages native AWS services *(my preference)* to address the client's requirements regarding uptime, continuous monitoring, and periodic updates while ensuring scalability, reliability, and security across all components. The AWS implementation involves **building the ML pipeline** using SageMaker, **deploying the pipeline** with CI/CD tools, and **serving the pipeline** through scalable endpoints as shown in Figure 1.

2 MLOps Architecture on AWS

2.1 Overall Architecture

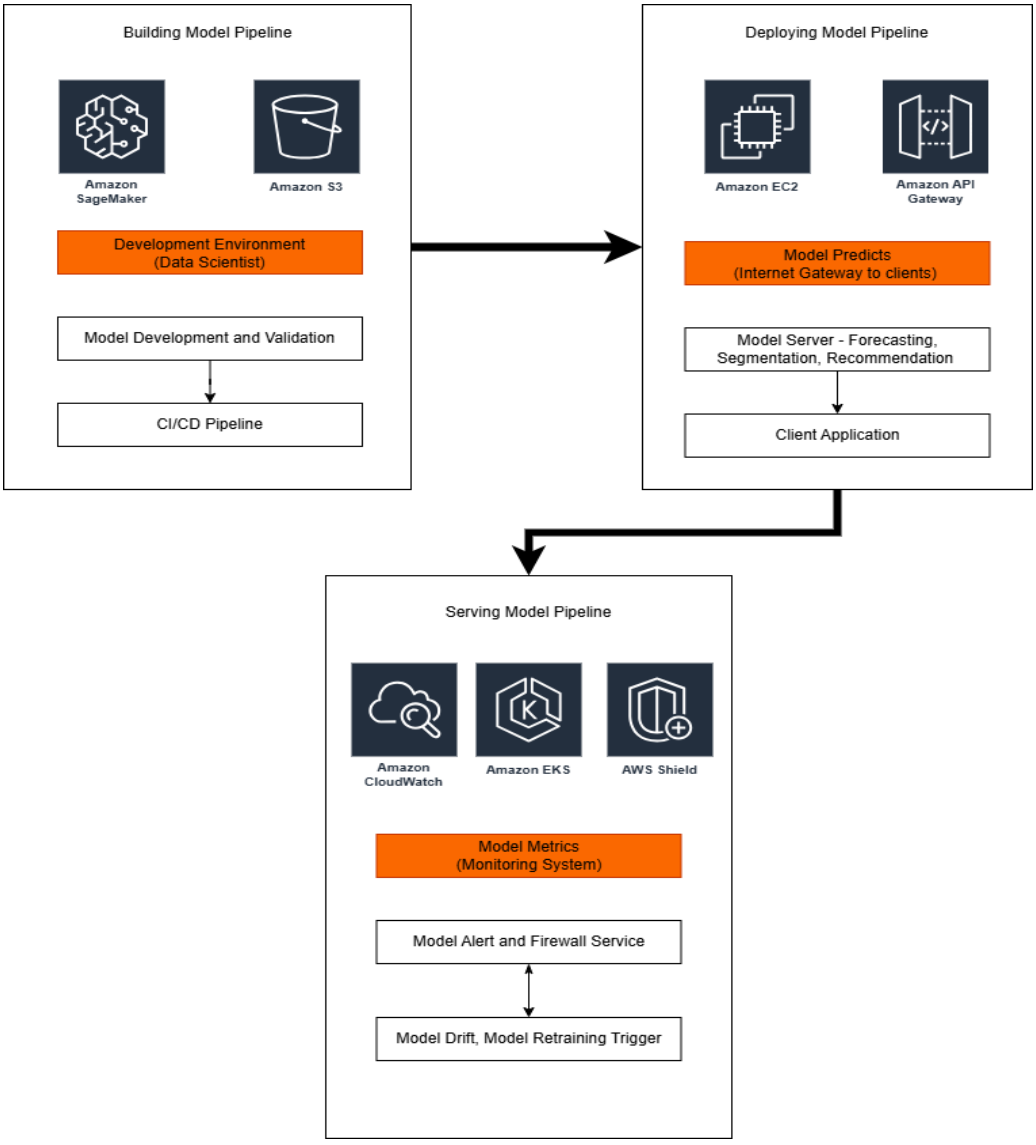

Figure 1: End-to-End MLOps Architecture on AWS

2.2 AWS Component Mapping

From Figure 1, Table 1 explains more components of my architecture.

Component	AWS Service	Description
Development Environment	SageMaker Notebooks, CodePipeline	Containerized environments with CI/CD integration using AWS developer tools - Sagemaker
Data Plane	S3, Feature Store	Data ingestion, storage in S3 data lake, and feature engineering with SageMaker Feature Store
Model Registry	SageMaker Model Registry	Version-controlled repository with approval workflows and model lineage tracking
Training Pipeline	SageMaker Pipelines	Automated training jobs with hyperparameter optimization using SageMaker
Serving Infrastructure	SageMaker Endpoints	Containerized model servers with auto-scaling on EC2 or serverless inference
Monitoring System	CloudWatch, SageMaker Model Monitor	Real-time performance tracking and drift detection with automated alerts

Table 1: AWS MLOps Components Mapping

2.3 AWS-Specific Considerations

Architecting on cloud services offers automated features / Serverless options where infrastructure is facilitated only by AWS Engineering team.

Concern	AWS Solution
Scalability	Auto Scaling Groups for SageMaker endpoints, EKS cluster autoscaling
Reliability	Multi-AZ / Edge Locations deployment, SageMaker Multi-Model Endpoints, S3 cross-region replication
Security	IAM roles, KMS encryption, VPC isolation, AWS PrivateLink for model endpoints
Cost Optimization	Spot Instances for training, Inference Recommender for right-sizing endpoints

Table 2: AWS Architecture Considerations

3 Data and Model Versioning

AWS provides several version control capabilities for MLOps (Machine Learning Operations) to help manage and track the development and deployment of machine learning models.

3.1 Data Versioning Strategy

AWS Sagemaker Experiments automatically tracks inputs, parameters, configurations, and results of your ML iterations as runs. You can group these runs into experiments, allowing you to compare different versions and identify the best-performing models. It involves:

- **S3 Versioning**: Enable versioning on all data buckets
- **Feature Store**: SageMaker Feature Store with offline/online stores
- **Schema Evolution**: AWS Glue Schema Registry with compatibility checking
- **Data Lineage**: AWS Lake Formation for tracking data provenance

3.2 Model Versioning Strategy

AWS SageMaker Model Registry allows you to catalog and manage model versions. You can create model groups and register different versions of your models, making it easier to track and manage multiple iterations of your ML models.

- **SageMaker Model Registry**: Version control with model packages
- **Artifacts**: Stored in S3 with versioned buckets
- **Updates**: Blue/green deployment with SageMaker endpoint variants
- **Rollbacks**: Model registry version pinning, automated rollback alarms

3.3 Model Lineage on AWS

Aspect	**AWS Implementation**
Tracking	SageMaker Experiments, MLflow tracking on EC2/EKS
Reproducibility	SageMaker Training Jobs with identical seed containers
Dependencies	SageMaker Training Toolkit containers, ECR versioned images
Metadata	Stored in DynamoDB with S3 artifact references

Table 3: AWS Model Lineage Approach

4 Monitoring and Maintenance

AWS provides several capabilities for monitoring metrics across different model types in the MLOps pipeline.

4.1 Model-Specific Metrics

Model	**Primary Metrics**	**CloudWatch Alarms**
Demand Forecasting	MAPE, RMSE	SNS alerts when >15% error
Customer Segmentation	Silhouette Score	EventBridge rules when ¡0.6
Recommendation Engine	CTR, Conversion Rate	Lambda remediation if <2.5%

Table 4: AWS Monitoring Metrics by Model Type

4.2 AWS Alerting System Design

- **Tiers**: CloudWatch Alarms with different severity levels
- **Channels**: Simple Notification Service(SNS) topics routing to Slack/Email/SMS
- **Aggregation**: AWS DevOps Guru for anomaly detection
- **Remediation**: AWS Systems Manager Automation documents

4.3 AWS Retraining Strategy

- **Triggers**: EventBridge scheduled rules, CloudWatch anomaly detection
- **Pipeline**: Step Functions orchestrating SageMaker Pipelines
- **Approvals**: Manual approval steps in CodePipeline

4.4 Proposed Monitoring Dashboard

I'd like to propose a unified MLOps monitoring dashboard that addresses the following weaknesses in the current AWS approach:

- **Fragmented Monitoring**:
  (i) AWS services are spread across different consoles and interfaces
  (ii) Requires switching between SageMaker, CloudWatch, and S3 consoles
- **Missing End-to-End Pipeline Views**:
  (i) No single view of the entire ML lifecycle
  (ii) Disconnected visualization of data, training, and inference phases

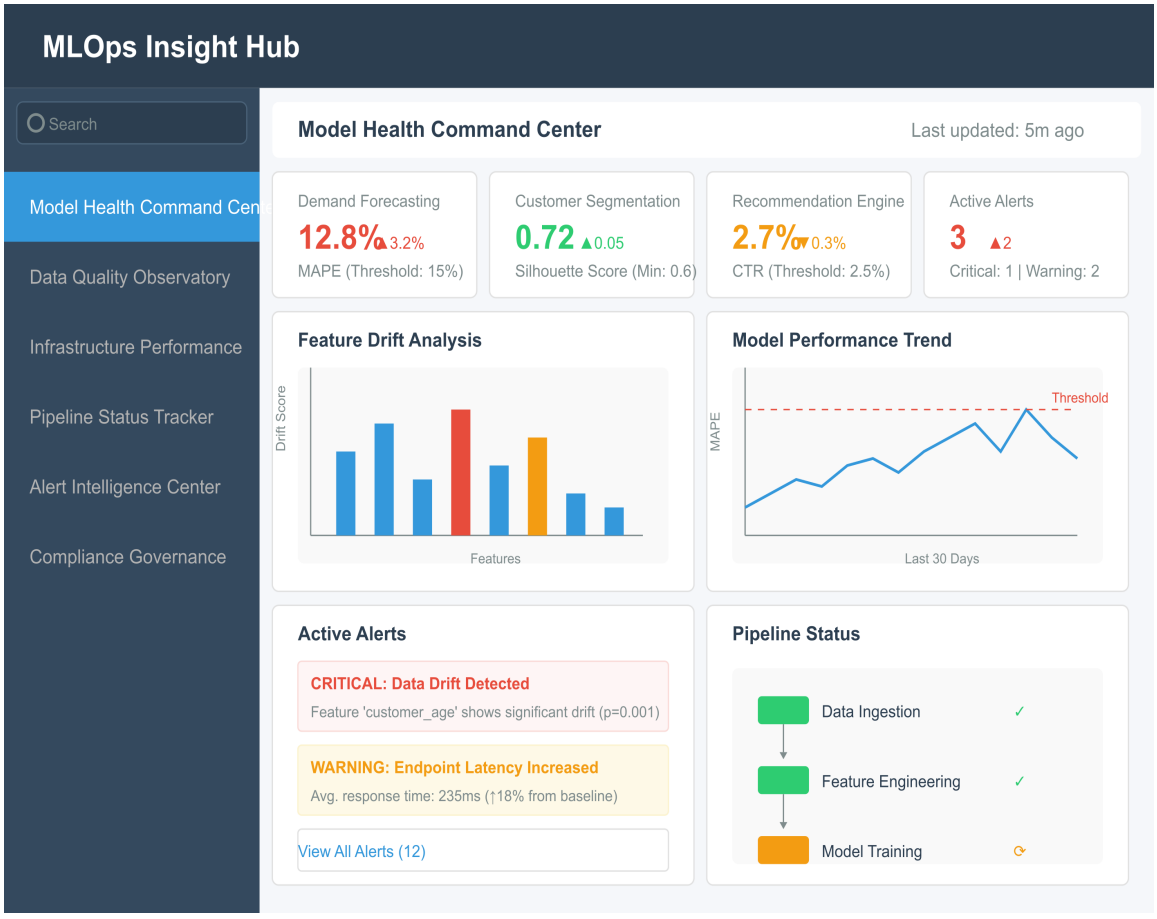

Figure 2: AWS CloudWatch/SageMaker Monitoring Dashboard

5 Documentation and Governance

5.1 Model Documentation

- **Model Cards**: Stored in S3, metadata in DynamoDB
- **Technical Docs**: AWS CodeArtifact for documentation packages
- **User Guides**: Hosted on S3 with CloudFront distribution
- **API Docs**: Amazon API Gateway developer portal

5.2 AWS Governance Framework

Process	**AWS Implementation**
Approval	AWS Service Catalog for approved models, IAM policies
Change Management	AWS Change Manager, Config rules
Roles	IAM roles with least privilege permissions
Auditing	AWS CloudTrail, AWS Config for compliance recording

Table 5: AWS Governance Framework

5.3 AWS Compliance and Standards

- **Regulatory**: AWS Artifact for compliance reports
- **Industry**: AWS Well-Architected Framework reviews
- **Ethical**: Amazon SageMaker Clarify for bias detection
- **Security**: AWS Security Hub for centralized monitoring

6 Conclusion

This AWS-based MLOps design provides ****** Limited with a cloud-native framework that addresses all client requirements while leveraging AWS best practices. The architecture delivers. While I could have come up with a custom solution, the AWS approach provides:

- Faster time-to-production (weeks vs months)
- Built-in enterprise-grade security and compliance
- Access to AWS's continuous service innovations