Autonomous Data Pipeline Agent (ADPA)

Progress Report

Archit Golatkar - Agent Planning & AWS Infrastructure
Umesh Adari - Data Engineering & Monitoring
Girik Tripathi - DevOps, Security & API Development

November 07, 2025

Contents

1 Executive Summary & Introduction

1.1 Project Overview

The Autonomous Data Pipeline Agent (ADPA) is a course project for DATA650 that demonstrates automated machine learning pipeline management using AWS cloud services. The system aims to reduce manual intervention in ML pipeline creation through intelligent planning, cloud-native execution, and comprehensive monitoring.

Demonstration Use Case: Retail sales forecasting pipeline that automatically processes sales data, performs cleaning and feature engineering, and provides performance monitoring.

1.2 Team Structure

Table 1: Team Responsibilities and Current Status

Member	Role	Key_Deliverables	Status
Archit Golatkar	AWS Infrastructure & ETL	S3 data lake, Glue ETL jobs, Lambda functions	Deployed |
Umesh Adari	Monitoring & Data Engineering	CloudWatch monitoring, KPI tracking, anomaly detection	Implemented |
Girik Tripathi	API Development & Security	API authentication, Lambda deployment, CloudWatch setup	Complete |

1.3 Current Achievements

AWS Infrastructure (Archit): Production-ready data architecture deployed via CDK with S3 buckets, Glue ETL processing, and Lambda integration.

Monitoring Framework (Umesh): Comprehensive monitoring system covering business KPIs, infrastructure health, performance analytics, and statistical anomaly detection.

API & Security (Girik): Authentication framework, Lambda deployment pipeline, and CloudWatch integration for system monitoring.

1.4 Project Status

- **Infrastructure**: Complete - Fully deployed AWS environment
- **Monitoring**: Complete - All monitoring components implemented and tested

- **Agent Core**: In Progress - Basic components implemented, integration ongoing
- **End-to-End Pipeline**: Planned - Integration work for final demonstration

---

2 Technical Implementation Status

2.1 Archit's AWS Infrastructure Implementation

2.1.1 Deployed Resources

Successfully deployed production AWS infrastructure using CDK v2:

CloudFormation Stack: `AdpaDataStack` (us-east-1)
Deployment Time: 134.59 seconds
Status: All resources operational

Table 2: Deployed AWS Infrastructure Components

Service	Component	Details
Amazon S3	Data Lake (3 buckets)	Raw, curated, artifacts buckets with lifecycle policies
AWS Glue	ETL Jobs & Crawlers	Cleaning job, features job, 2 crawlers with scheduling
AWS Lambda	Data Processor	EventBridge-triggered data processing function
CloudWatch	Monitoring	Custom namespace, logs, dashboards

2.1.2 ETL Processing

- **Data Cleaning Job**: PySpark-based cleaning with automated quality checks
- **Feature Engineering Job**: Automated feature generation and selection

- **Automated Scheduling**: Glue triggers for dependency-based execution
- **Event-Driven Architecture**: S3 events trigger Lambda processing

2.2 Umesh's Monitoring Implementation

2.2.1 Week 2 Monitoring Framework

Implemented comprehensive monitoring across four key areas:

Table 3: Monitoring Framework Implementation

Component	Implementation	Validation_Results
Business KPIs	15+ KPI calculations with trend analysis	7 days historical data generated
Infrastructure Health	EC2, SageMaker, RDS health monitoring	4 components monitored, 100% health score
Performance Analytics	8 dashboard widgets with capacity planning	95.7% success rate achieved

Anomaly Detection	Statistical + threshold-based detection	100% detection accuracy in testing

2.2.2 Key Features

- **Mock-First Development**: All components work without AWS dependencies for development
- **Real-Time Metrics**: CloudWatch integration with custom dashboards
- **Automated Alerting**: Threshold-based alerts with severity classification
- **Trend Analysis**: Historical pattern analysis with forecasting capabilities

2.3 Girik's API & Security Implementation

2.3.1 Completed Components

- **API Foundation**: RESTful framework with proper authentication mechanisms
- **Lambda Deployment**: Automated deployment pipeline for serverless functions

- **Security Setup**: IAM roles, API authentication, and access control
- **Environment Management**: Configuration and environment variable management
- **Status Monitoring**: Health check systems and CloudWatch integration

2.3.2 Integration Results

Successfully established secure API infrastructure with: - Authentication mechanisms for API access - Lambda function deployment and management - CloudWatch permissions and logging setup - Environment-specific configuration management

---

3 System Architecture & AWS Integration

3.1 High-Level Architecture

```
                    ADPA SYSTEM ARCHITECTURE


  Data Sources → S3 Raw → Glue ETL → S3 Curated → Models

                                                  Artifacts

                      EventBridge ← Lambda Processing

         Monitoring ← CloudWatch ← Custom Metrics

                      Dashboards & Alerts
```

3.2 Current Integration Status

Table 4: System Integration Status

Layer	Status	Description
Data Storage	Complete |	3 data lake with proper bucket organization |
Processing	Complete |	lue ETL jobs with automated scheduling |
Monitoring	Complete |	omprehensive monitoring across all dimensions |
API/Security	Complete |	uthentication and deployment infrastructure |
Agent Core	In Progress |	asic components implemented, integration ongoing |

3.3 Actual Deployment Details

Production Resources: - **S3 Buckets**: `adpadatastack-rawbucket0c3ee094-46betroebefa` (raw), `adpadatastack-curatedbucket6a59c97e-csypjbbtlgtd` (curated) - **Glue Database**: `adpa_raw_db` - **ETL Jobs**: `adpa-cleaning-job`, `adpa-features-job` - **Monitoring**: Custom CloudWatch namespace with 8+ dashboard widgets

---

4 Current Challenges & Next Steps

4.1 Technical Challenges Encountered

4.1.1 Dependency Management

Challenge: Developing monitoring systems without requiring full AWS setup for testing.
Solution: Implemented mock-first approach allowing development and testing without external dependencies.

4.1.2 Service Integration Complexity

Challenge: Coordinating multiple AWS services with proper permissions and event handling.
Solution: Used CDK infrastructure-as-code with comprehensive IAM role management.

4.1.3 Agent Component Integration

Challenge: Integrating individual components into cohesive autonomous agent.
Current Status: Core components exist separately, integration work in progress.

4.2 Immediate Next Steps (2 Weeks)

4.2.1 Priority 1: Agent Integration

- **Responsibility**: Archit + Umesh
- **Tasks**: Connect agent planning components with AWS infrastructure
- **Goal**: Complete end-to-end pipeline execution from planning to monitoring

4.2.2 Priority 2: Pipeline Demonstration

- **Responsibility**: All team members
- **Tasks**: Implement complete retail sales forecasting demonstration
- **Goal**: Show autonomous pipeline creation and execution

4.2.3 Priority 3: Performance Validation

- **Responsibility**: Umesh + Girik

- **Tasks**: Validate monitoring system with real pipeline executions
- **Goal**: Demonstrate comprehensive observability during pipeline runs

4.3 Integration Work Needed

Table 5: Remaining Integration Work

Component	Effort	Timeline	Blocker
Agent-AWS Integration	Medium	1 week	Component coordination
End-to-End Testing	Medium	1 week	Test data setup
Demo Pipeline	Low	3 days	None
Documentation	Low	2 days	None

---

5 Conclusion & Timeline

5.1 Progress Assessment

The ADPA project has successfully demonstrated significant technical achievements across cloud infrastructure, monitoring, and API development. Each team member has delivered production-ready components that work independently and are ready for integration.

Major Accomplishments: - **Complete AWS Infrastructure**: Production deployment with all necessary services - **Comprehensive Monitoring**: Enterprise-grade observability across multiple dimensions - **Secure API Foundation**: Authentication and deployment infrastructure ready - **Proof of Concept**: All core concepts validated through working implementations

5.2 Current Limitations

- **Agent Integration**: Core agent components require integration work to achieve full autonomy
- **End-to-End Flow**: Individual components need orchestration for complete pipeline execution
- **Limited ML Models**: Focus has been on infrastructure rather than advanced ML algorithms
- **Demo Scope**: Current scope suitable for course demonstration rather than production deployment

5.3 Timeline for Course Completion

Table 6: Completion Timeline

Week	Focus	Deliverables	Success_Criteria
Week 1	Integration & Testing	Agent-AWS integration, end-to-end testing	Complete pipeline execution from planning to monitoring
Week 2	Demo Preparation	Retail forecasting demo, performance validation	Autonomous pipeline creation with comprehensive monitoring
Final Week	Final Presentation	Live demonstration, final documentation	Successful course demonstration with Q&A

5.4 Course Demonstration Plan

Final Demo Scope: Autonomous creation and execution of a retail sales forecasting pipeline demonstrating: 1. Intelligent data analysis and preprocessing planning 2. Automated ETL execution with AWS Glue 3. Real-time monitoring with anomaly detection
4. Performance analytics and optimization recommendations

The project successfully validates the core concepts of autonomous ML pipeline management while providing a solid technical foundation for future enhancement and real-world application.

Course: DATA650 - Big Data Analytics | **Institution**: University of Maryland | **Semester**: Fall 2025