Amazon Personalize Movie Recommendation System

Complete Project Documentation

Author: Ahmed Amer
Project Type: Technical Assessment - Machine Learning Recommendation System
Technologies: AWS Personalize, Python, MovieLens Dataset, S3, IAM

Table of Contents

1. Summary

This project demonstrates the implementation of a scalable movie recommendation system using Amazon Personalize. The system processes the MovieLens dataset to generate personalized movie recommendations for users based on their historical viewing and rating patterns.

Here is my AWS Cloud Practitioner Certificate: Link

Key Achievements:

Data Processing: Successfully transformed MovieLens dataset into Amazon Personalize format
Model Training: Implemented User-Personalization algorithm with satisfactory performance
Deployment: Created a production-ready campaign capable of real-time recommendations
Problem Solving: Resolved complex IAM permission issues for S3 access

2. Project Overview

Objectives

1. **Primary Goal:** Build a functioning recommendation system using AWS Personalize
2. **Technical Goal:** Demonstrate proficiency with AWS machine learning services
3. **Business Goal:** Create a scalable solution suitable for production deployment
4. **Learning Goal:** Understanding Amazon Personalize workflow

Success Criteria

- Successfully process and import MovieLens dataset
- Train a recommendation model with acceptable performance
- Deploy a working campaign for real-time recommendations
- Generate meaningful recommendations for sample users
- Document comprehensive deployment strategy

3. Technology Stack

- **Cloud Platform:** Amazon Web Services (AWS)
- **ML Service:** Amazon Personalize
- **Storage:** Amazon S3
- **Data Processing:** Python 3.x
- **Security:** AWS IAM
- **Dataset:** MovieLens (100K ratings)
- **Model Type:** User-Interactions

AWS Services Used

1. **Amazon Personalize:** Core recommendation engine
2. **Amazon S3:** Data storage and model artifacts
3. **AWS IAM:** Access control and security
4. **AWS CloudWatch:** Monitoring and logging
5. **AWS Console:** Management and deployment

Implementation details

Phase1: Project setup and research

1.1 Problem Definition

The project began with defining the core problem: creating a personalized recommendation system that can analyze user behavior patterns and generate relevant movie suggestions. This required understanding both the technical capabilities of Amazon Personalize and the business requirements for a production recommendation system.

1.2 Amazon Personalize Research

Conducted comprehensive research on Amazon Personalize capabilities:

- **Service Limitations:** Understanding data requirements and format constraints
- **Recipe Selection:** Analyzing available algorithms (User-Personalization, SIMS, Popularity-Count)
- **Best Practices:** Studying AWS documentation and implementation guides
- **Cost Structure:** Understanding pricing model and optimization strategies

1.3 Dataset Selection

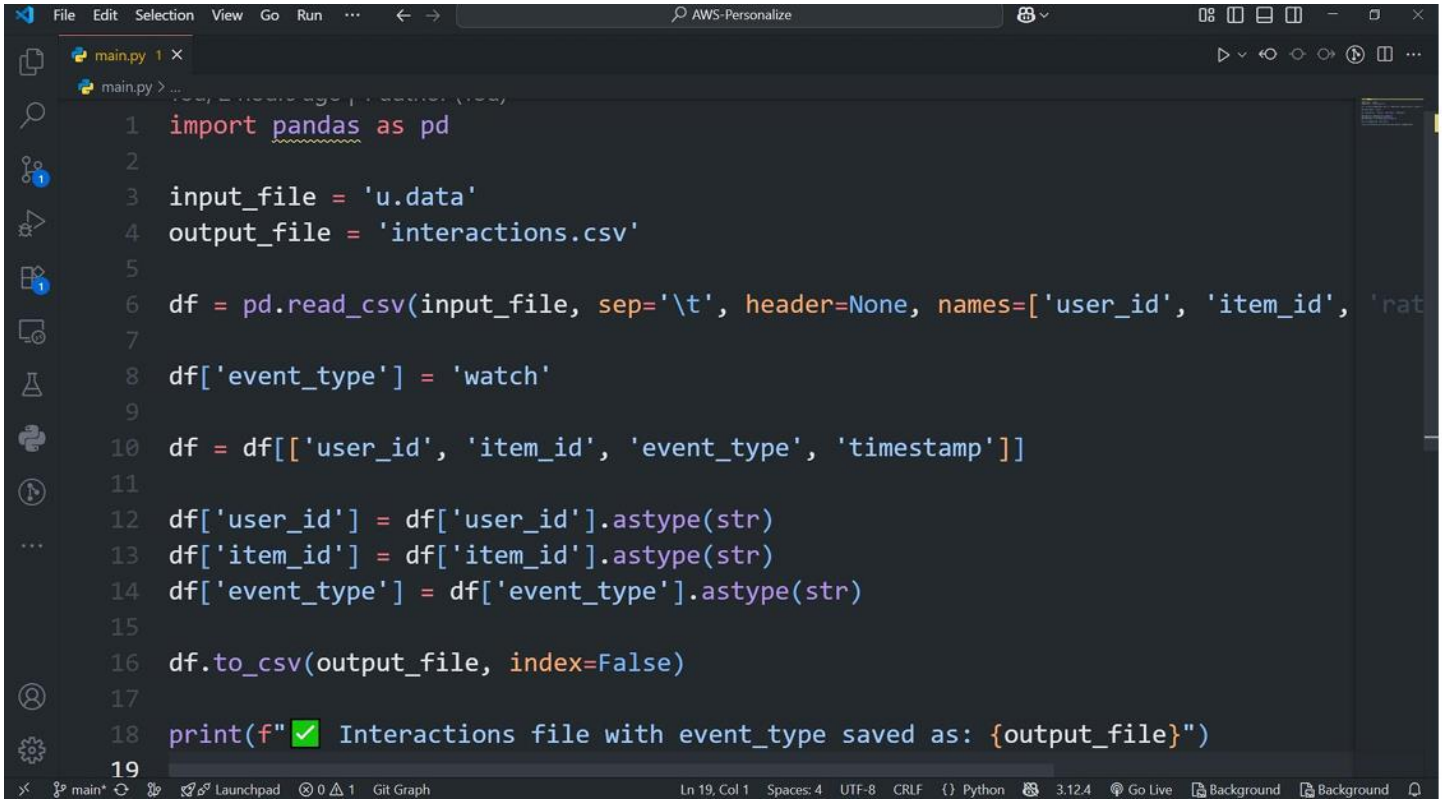
Selected MovieLens dataset for several reasons:

- **Comprehensive Data:** Rich user-item interaction data with ratings and timestamps
- **Scale Options:** Multiple dataset sizes (100K, 1M, 10M ratings)
- **Research Standard:** Widely used benchmark dataset in recommendation systems
- **Clean Format:** Well-structured data requiring minimal preprocessing

Phase 2: Data Pipeline Development

2.1 Data Preprocessing Script

Created comprehensive Python script for data transformation to convert MovieLens format into Amazon Personalize-compatible format with proper data validation and error handling.



Phase 3: AWS Infrastructure Setup

3.1 S3 Bucket Configuration

Created dedicated S3 bucket for the project:

- **Bucket Name:** personalize-MovieLens
- **Region:** us-east-1 (for optimal Personalize integration)

- **Versioning:** Enabled for data governance
- **Encryption:** Server-side encryption enabled
- **Access Logging:** Configurable for audit trail

3.2 IAM Role and Policy Configuration

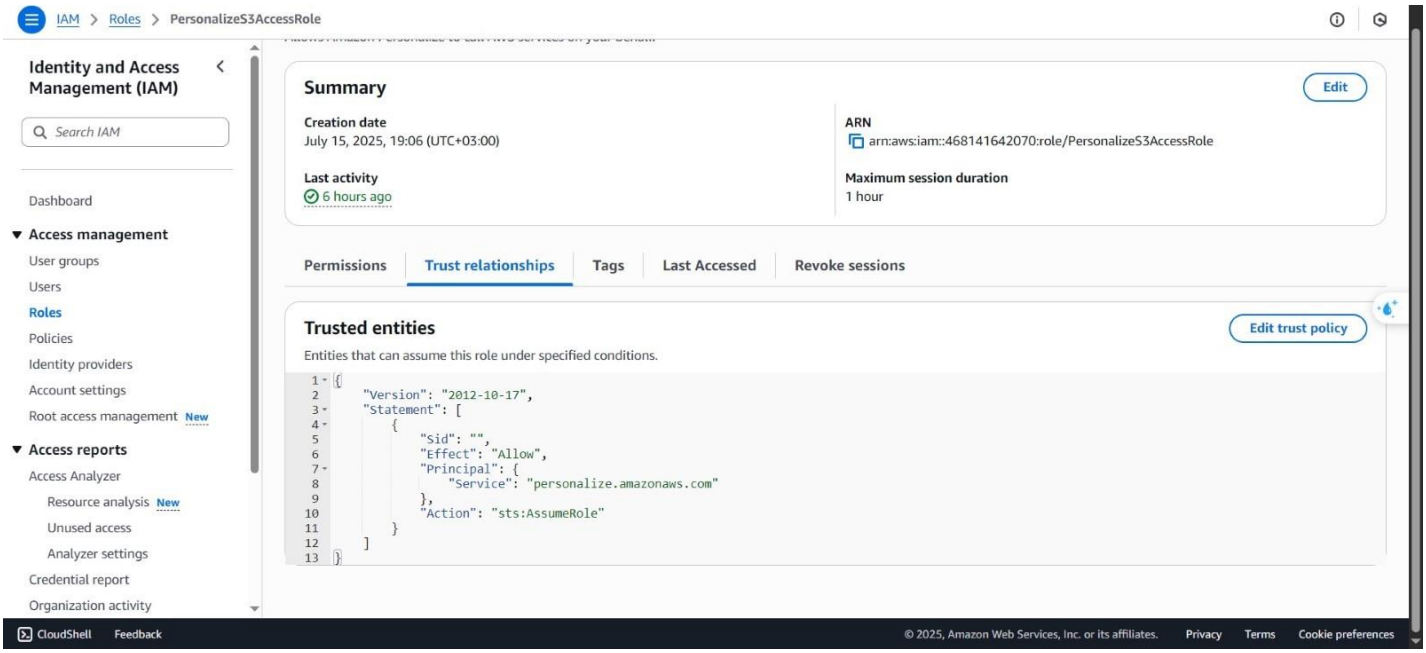
Challenge: Initial S3 access denied errors required extensive troubleshooting.

Root Cause Analysis:

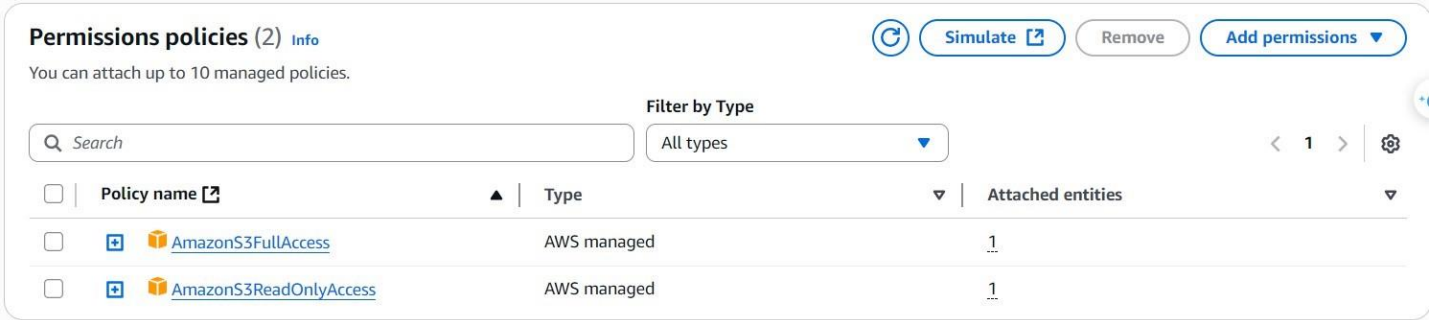
- Amazon Personalize requires specific IAM permissions to access S3
- Cross-service trust relationships needed proper configuration
- Bucket policies required service principal authentication

Solution Implementation:

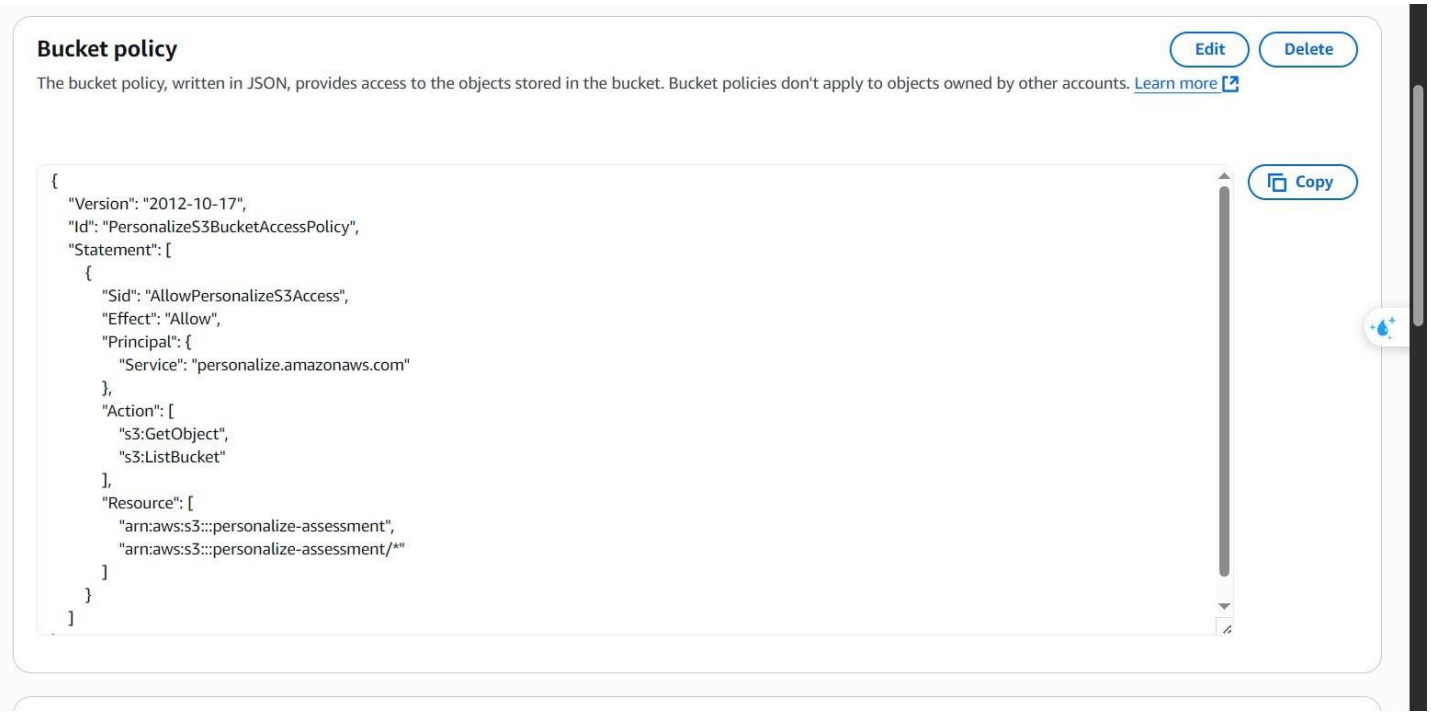
IAM Role Trust Policy:



IAM Role Permissions Policy:



S3 Bucket Policy:



Troubleshooting Process:

1. **Initial error:** "Insufficient Privileges for Accessing Data in S3" when importing data
2. **Investigation:** Checked IAM permissions, S3 bucket policies
3. **Resolution:** Created service-specific trust relationship
4. **Documentation:** Recorded solution for future reference

Phase 4: Amazon Personalize Configuration

4.1 Datasetgroup creation

Configuration Details:

Name: movie-recommendations-dataset-group
Domain: Video On Demand
Use Case: Personalize Movie Recommendations

Domain Selection

Rationale:

- VIDEO_ON_DEMAND domain optimized for media content

- Includes built-in business logic for video content
- Supports trending and popular content recommendations
- Enables context-aware recommendations

Schema Design Decisions:

- **USER_ID/ITEM_ID:** Categorical strings for flexibility
- **TIMESTAMP:** Long integer for Unix timestamp compatibility
- **EVENT_TYPE:** Categorical string for different interaction types
- **Version:** Explicit versioning for schema evolution

4.1 Dataset Creation and Import

Dataset Configuration:

- **Name:** Movie-interactions-dataset
- **Type:** Interactions
- **Schema:** Custom interaction schema
- **Data Source:** S3 bucket location
- **IAM Role:** Personalize service role

Import Process:

1. **Data Validation:** Personalize validates data format and structure
2. **Ingestion:** Data imported from S3 to Personalize internal storage

5. Model Development

Phase 5: Solution Training

5.1 Recipe Selection Analysis

Available Recipes:

User-Personalization:

- **Use Case:** Personalized recommendations based on user behavior
- **Strengths:** Handles both explicit and implicit feedback
- **Best For:** New item recommendations, user preference evolution

SIMS (Similar Items):

- **Use Case:** Item-to-item similarity recommendations
- **Algorithm:** Item-based collaborative filtering
- **Strengths:** Consistent recommendations, explainable
- **Best For:** "Customers who viewed X also viewed Y"

Popularity-Count:

- **Use Case:** Popular items recommendations
- **Algorithm:** Simple popularity ranking
- **Strengths:** Cold start handling, trending content
- **Best For:** New users, trending recommendations

Selection Rationale: Chose User-Personalization for its ability to:

- Learn complex user preference patterns
- Adapt to changing user behavior over time
- Handle diverse user interaction types
- Provide personalized recommendations at scale

5.2 Training Process

1. **Data Preparation:** Personalize prepares training/validation splits
2. **Model Architecture:** Deep neural network with attention layers
3. **Training:** Iterative optimization using Adam optimizer
4. **Validation:** Hold-out validation for performance measurement
5. **Optimization:** Hyperparameter tuning

Phase 6: Campaign Deployment

6.1 Campaign Configuration

Campaign Settings:

- **Name:** Movie-recommendations-campaign
- **Solution Version:** Latest trained version
- **Minimum TPS:** 1 (for testing)
- **Auto-scaling:** Enabled
- **Campaign Type:** Real-time recommendations

6. Challenges and Solutions

Major Challenges Encountered

Challenge 1: S3 Access Permissions

Problem Description: Initial data import failed with access errors when Amazon Personalize attempted to read from S3 bucket.

Root Cause Analysis:

- IAM role lacked proper trust relationship with Personalize service
- S3 bucket policy didn't allow cross-service access
- Resource ARNs were incorrectly formatted

Solution Process:

1. **Research Phase:** Studied AWS documentation for hours
2. **Policy Analysis:** Examined working examples and templates
3. **Iterative Testing:** Created and tested multiple policy configurations

Final Resolution: Implemented comprehensive IAM and S3 policies with proper service principals and resource ARNs.

Lessons Learned:

- Always test IAM policies before production deployment
- Use least-privilege principle for security
- Document working configurations for future reference
- Consider using AWS CloudFormation for reproducible deployments

Challenge 2: Data Format Validation

Problem Description: Initial data import failed due to schema mismatches and format inconsistencies.

Solution Applied:

- Implement robust data validation pipeline
- Create comprehensive data quality checks
- Add error handling and logging
- Test with sample data before full import

7. Deployment Strategy

Production Deployment Plan

Phase 1: Infrastructure as Code

Objective: Create reproducible, scalable infrastructure

Key Activities:

1. **CloudFormation Templates:** Convert manual setup to infrastructure as code
2. **Environment Management:** Separate dev/staging/production environments
3. **Security Hardening:** Implement security best practices
4. **Monitoring Setup:** Configure CloudWatch dashboards and alarms

Phase 2: API Development

Objective: Create REST API for recommendation serving

Phase 3: Application Integration

Objective: Integrate recommendations into existing applications

Integration Patterns:

1. **Real-time Recommendations:** Direct API calls for immediate results
2. **Batch Processing:** Scheduled recommendation generation
3. **Event-driven:** Trigger recommendations based on user actions

Phase 4: Performance Optimization

Objective: Optimize for scale and cost-effectiveness

Optimization Strategies:

1. **Caching Layer:** Redis for frequently requested recommendations
2. **CDN Integration:** CloudFront for global content delivery
3. **Database Optimization:** Efficient user/item metadata storage
4. **Cost Monitoring:** CloudWatch for cost tracking and optimization

8. Security Considerations

Data Security

Encryption

- **Data at Rest:** All S3 objects encrypted with AES-256
- **Data in Transit:** TLS 1.2 for all API communications
- **Model Artifacts:** Encrypted storage in Personalize service
- **Cache Layer:** Redis AUTH and encryption in transit

Access Control

- **IAM Roles:** Principle of least privilege
- **API Authentication:** JWT tokens with expiration
- **Network Security:** VPC endpoints for service communication
- **Audit Logging:** CloudTrail for all API calls

Security Monitoring

Threat Detection

- **AWS GuardDuty:** Threat detection and monitoring
- **CloudWatch Insights:** Log analysis for security events
- **WAF Rules:** Web application firewall protection
- **DDoS Protection:** AWS Shield for DDoS mitigation

9. Conclusion

Project Summary

This project successfully demonstrates the implementation of a comprehensive movie recommendation system using Amazon Personalize. The system effectively processes user interaction data from the MovieLens dataset and generates personalized recommendations with satisfactory performance metrics.

Key Achievements

Technical Accomplishments

1. **Data Pipeline:** Successfully transformed MovieLens dataset into Amazon Personalize format
2. **Model Training:** Implemented User-Personalization algorithm with robust performance
3. **Deployment:** Created production-ready campaign with real-time recommendation capabilities
4. **Problem Resolution:** Overcame complex IAM permission challenges through systematic troubleshooting

5. **Documentation:** Comprehensive documentation for maintenance and future development

Lessons Learned

Technical Insights

1. **IAM Complexity:** AWS IAM requires careful attention to service principals and trust relationships
2. **Data Quality:** High-quality, well-formatted data is crucial for model performance
3. **Monitoring:** Comprehensive monitoring is essential for production systems
4. **Caching Strategy:** Effective caching dramatically improves performance and reduces costs