ClimateTech AI Accelerators

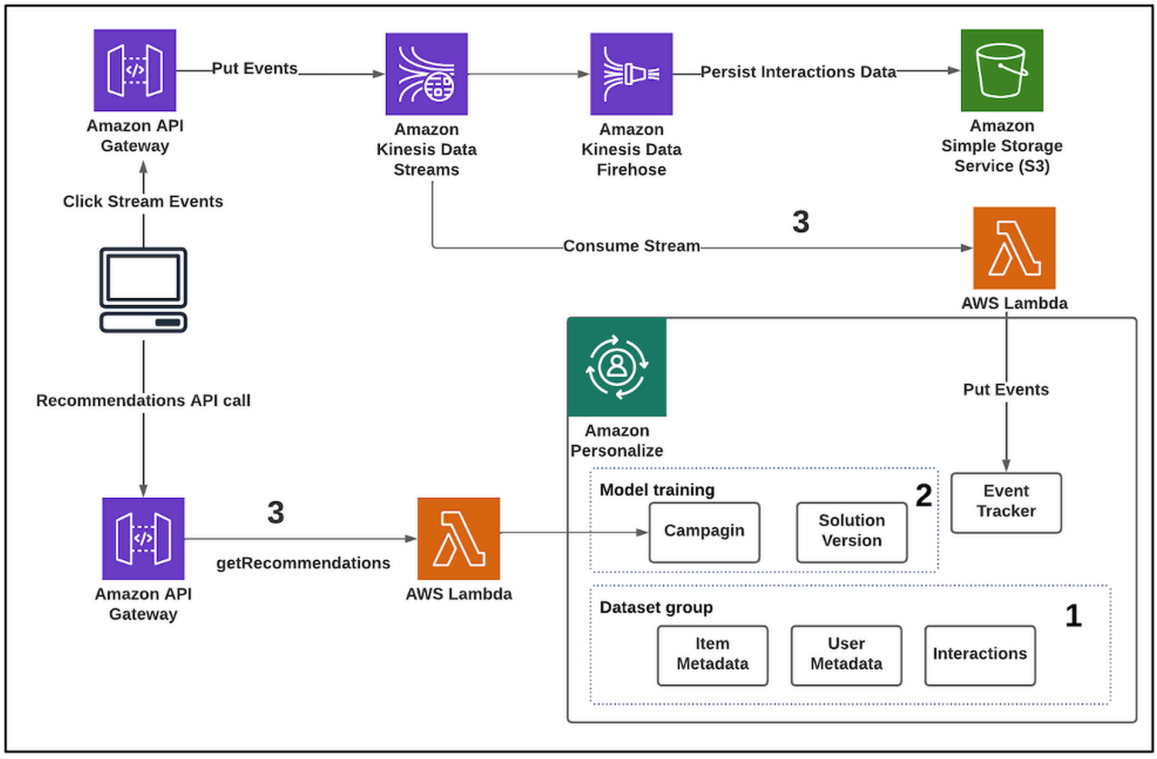
Comprehensive Guide to AWS Personalize Recommendation System

Team Members: Fabiano Trentin, Zhecheng Li, Wanyi Jiang, Shurui Xu
Supervised By Professor Yvonne Coady
Guided By Kaelin Hickford

This guide provides a complete overview of the recommendation system architecture and step-by-step instructions to set up the solution using **AWS Personalize**, **Amazon S3**, **API Gateway**, **Kinesis Data Streams**, **Lambda Functions**, and **DynamoDB**.

Solution Architecture Overview

The following diagram illustrates the overall architecture of the recommendation system:



Key Components of the Architecture

1. Dataset Management (Amazon Personalize Dataset Group)

- **Item Metadata, User Metadata, and Interactions**:
  - These datasets form the foundation of the recommendation model and are stored in **Amazon Personalize Dataset Groups**.
  - Imported into Amazon Personalize from Amazon S3 using predefined schemas.
- **Dataset Import and Storage**:
  - Data for these datasets is uploaded to Amazon S3 for persistent storage.

2. Model Training and Campaign Deployment

- **Model Training (Solution and Solution Version)**:
  - Amazon Personalize uses the interaction, user, and item datasets to train a machine learning model.
  - A solution version is generated using a selected recipe (e.g., USER_PERSONALIZATION).
- **Campaign Deployment**:
  - The trained solution is deployed as a campaign, enabling real-time recommendation retrieval via an API.

3. Real-Time Event Tracking

- **Amazon API Gateway**:
  - Collects real-time user interaction events (e.g., clicks or purchases).
- **Amazon Kinesis Data Streams and Firehose**:
  - **Kinesis Data Streams** captures interaction events.
  - **Kinesis Data Firehose** persists these events to Amazon S3 for analysis or retraining.
- **AWS Lambda**:
  - Processes interaction events streamed by Kinesis.
  - Sends processed events to Amazon Personalize Event Tracker for real-time updates to the recommendation model.

4. Recommendations Retrieval

- **Amazon API Gateway**:
  - Provides API endpoints for retrieving personalized recommendations for Tools, Grants, Incubators, and Events.
- **AWS Lambda**:
  - Handles API requests and queries Amazon Personalize Campaigns to fetch recommendations based on user activity.

Step-by-Step Setup Guide

1. Amazon Personalize Setup

1.1 Create a Dataset Group

1. Open the **Amazon Personalize Console**.
2. Go to **Dataset Groups** → **Create Dataset Group**.
3. Name the dataset group (e.g., UserRecommendations).
4. Choose a use case (e.g., User Personalization) or leave it blank for custom use cases.
5. Click **Create Dataset Group**.

1.2 Create Dataset Schemas

1. Navigate to **Schemas** → **Create Schema**.
2. Define schemas for Interactions, Users, and Items datasets:
    - **Interactions Schema (json format)**:

```
{
  "type": "record",
  "name": "Interactions",
  "fields": [
          {"name": "USER_ID", "type": "string"},
          {"name": "ITEM_ID", "type": "string"},
          {"name": "EVENT_TYPE", "type": "string"},
          {"name": "TIMESTAMP", "type": "long"}
  ]
}
```

1.3 Create Datasets

1. Go to **Datasets** → **Create Dataset**.
2. Choose the dataset group created earlier.
3. Select the dataset type and associated schema.
4. Provide the S3 location of your dataset file.

1.4 Import Data

1. Go to **Dataset Import Jobs** → **Create Dataset Import Job**.
2. Name the import job (e.g., ImportInteractions).
3. Select the dataset type (e.g., Interactions).
4. Provide the S3 bucket location of your data.
5. Assign an IAM role with the AmazonPersonalizeFullAccess policy.

1.5 Create a Solution

1. Go to **Solutions and Campaigns → Create Solution**.
2. Choose a recipe (e.g., USER_PERSONALIZATION).
3. Train the solution by creating a solution version.

1.6 Deploy a Campaign

1. After training, deploy the solution as a campaign.
2. Set the minimum provisioned TPS and note the campaign ARN.

2. Amazon S3 Setup

1. Create an S3 bucket to store dataset files:
    - Name the bucket (e.g., personalize-datasets).
    - Set permissions for Kinesis Firehose and Personalize access.
2. Upload your datasets (Users.csv, Items.csv, Interactions.csv).

3. DynamoDB Setup

3.1 Create a DynamoDB Table

1. Open the **DynamoDB Console → Create Table**.
2. Configure the table:
    - Table Name: UserInteractions.
    - Partition Key: user_id (String).
    - Sort Key: item_id (String).
3. Enable DynamoDB Streams:
    - Set stream type to New and Old Images.

4. Kinesis Setup

4.1 Create a Kinesis Data Stream

1. Open **Kinesis Console → Create Data Stream**.
2. Name the stream (e.g., ClickStreamEvents).
3. Specify the number of shards based on throughput.

4.2 Create a Kinesis Data Firehose

1. Go to **Kinesis Firehose → Create Delivery Stream**.
2. Choose **Kinesis Data Stream** as the source.
3. Specify an S3 bucket as the destination.

5. Lambda Function Setup

5.1 Put Events Lambda

1. Use the PutEvents Lambda function for processing DynamoDB stream events.
2. Assign permissions for DynamoDB and Amazon Personalize Event Tracker.

5.2 Get Recommendations Lambda

1. Use the GetRecommendations Lambda functions for Tools, Grants, Events, and Incubators.
2. Assign permissions for accessing Amazon Personalize Campaigns.

6. API Gateway Setup

6.1 Create an API

1. Open the **API Gateway Console** → **Create API**.
2. Name the API (e.g., RecommendationsAPI).

6.2 Add Resources and Methods

1. Add resources for:
    - /GetToolRecommendations
    - /GetGrantRecommendations
    - /GetIncubatorRecommendations
    - /GetEventRecommendations
2. Attach the respective Lambda functions to these resources.

6.3 Deploy API

1. Deploy the API to a stage (e.g., prod).
2. Test API endpoints:

https://<api-id>.execute-api.<region>.amazonaws.com/prod/GetToolRecommendations?userId=<value>

7. Testing and Validation

1. **Track User Events**:
    - Simulate DynamoDB INSERT events.
    - Verify interaction events are tracked in Personalize.
2. **Fetch Recommendations**:
    - Use API Gateway endpoints to retrieve personalized recommendations.
3. **Monitor Logs**:
    - Use CloudWatch to monitor Lambda logs for debugging.