Asia Pacific Bank
Data & Analytics Platform

Software Architecture Document
Version 1.0

William Chu

william.w.y.chu@gmail.com

Introduction

Purpose

This document provides an architecture overview of Asia Pacific Bank (APB) Data & Analytics Platform. It is intended to record the architectural decisions & process of the system.

Scope

This document contains the overview of the architecture of Data & Analytics Platform designed for Asia Pacific Bank. It is intended to be read by business analyst, engineers & product owner.

Definitions, Acronyms and Abbreviations

AWS	Amazon Web Service
ETL	Extract, Transform & Learn. This is the process of importing aggregated & transformed data from data sources to a given data sink.
Machine Learning	**Machine learning** (**ML**) is a field of inquiry devoted to understanding and building methods that 'learn', that is, methods that leverage data to improve performance on some set of tasks.[1]
ML	Machine Learning

Architectural Representation

The proposed architecture of platform will be presented by the necessary UML diagram(s).

Goals & Assumptions

Asia Pacific Bank (APB) is hoping to utilize (near) real-time data analytics & machine learning to drive sales. The bank hopes to make use of large pool of clickstream data and transactions history in order to understand its clients more to up-sales clients.

Goals

The bank has set the following goals for the proposed platform,

- Low latency
- Privacy minded
- Scalable (Ability to scale to two or more regions)
- Secure

Assumptions

- Approved by corporate to subscribe to AWS services

- Data in question (both clickstream data & transaction history and related data) are currently stored in a RDBMS
- Existing applications & infrastructure may not be able to be enhanced or upgraded

Functional requirements

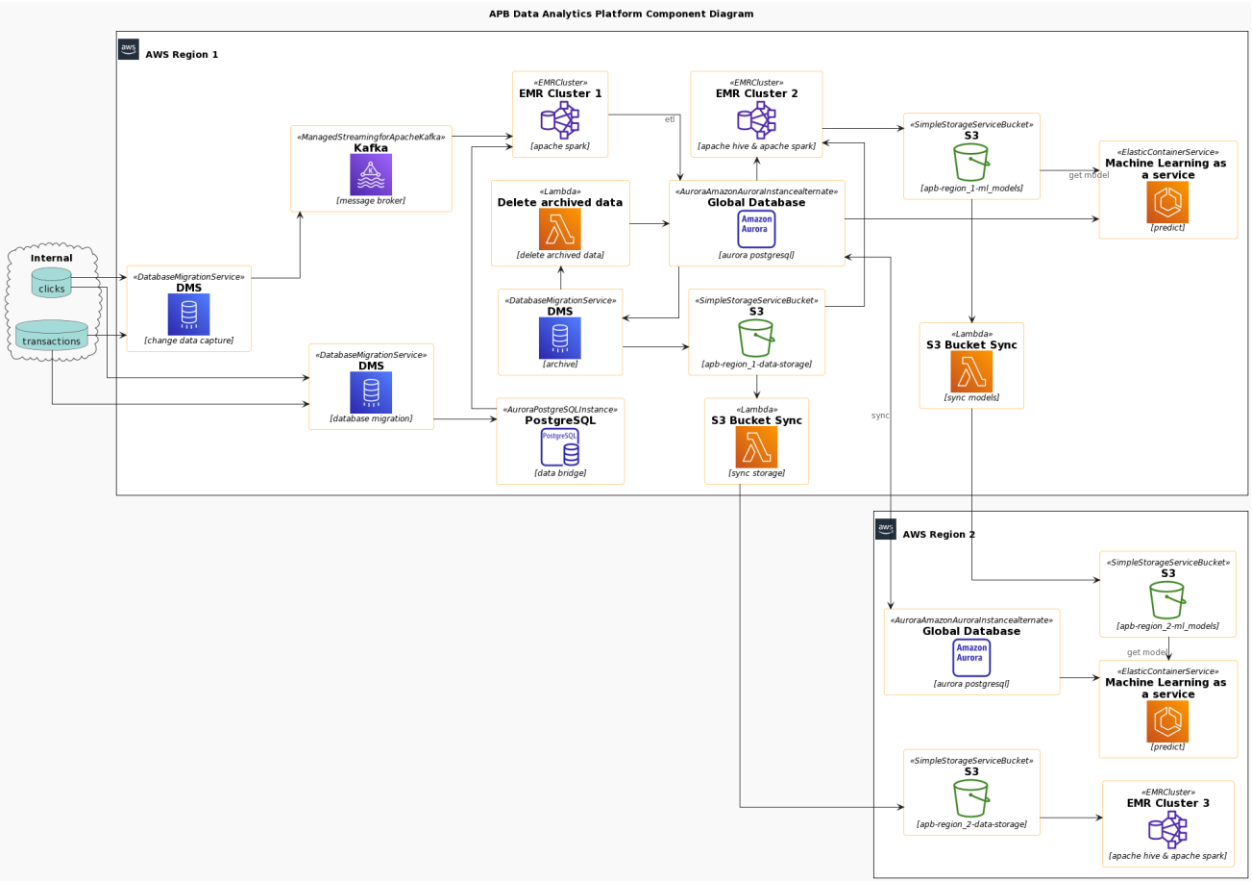
- Multi-tier of storages
- On demand prediction
- Not bounded by a single (AWS) region
- (Near) real-time data analytics [on both hot & cold data]

Architecture Overview

Components

- AWS Aurora Global Database
    - Globally distributed database cluster
- AWS DMS (Database migration service)
    - Migrate on premise database to AWS RDS
    - Change data capture
- AWS ECS (Elastic Container Service)
    - Host & run docker images on AWS EC2 instances
- AWS EMR (Elastic Map Reduce)
    - AWS managed Apache Spark & Apache Hive & Presto cluster
    - Data analytics platform
- AWS Lambda
    - Run function on AWS
- AWS S3 (Simple Storage Service)
    - Cloud object storage
- Self-developed Machine Learning as a service web application
    - SpringBoot + embedded Apache Spark
    - RESTful API to get prediction on the fly
        - Retrieve pre-built models from AWS S3 bucket
        - Connect to Aurora Global Database
        - Predict with SparkSession from embedded Spark

The big picture



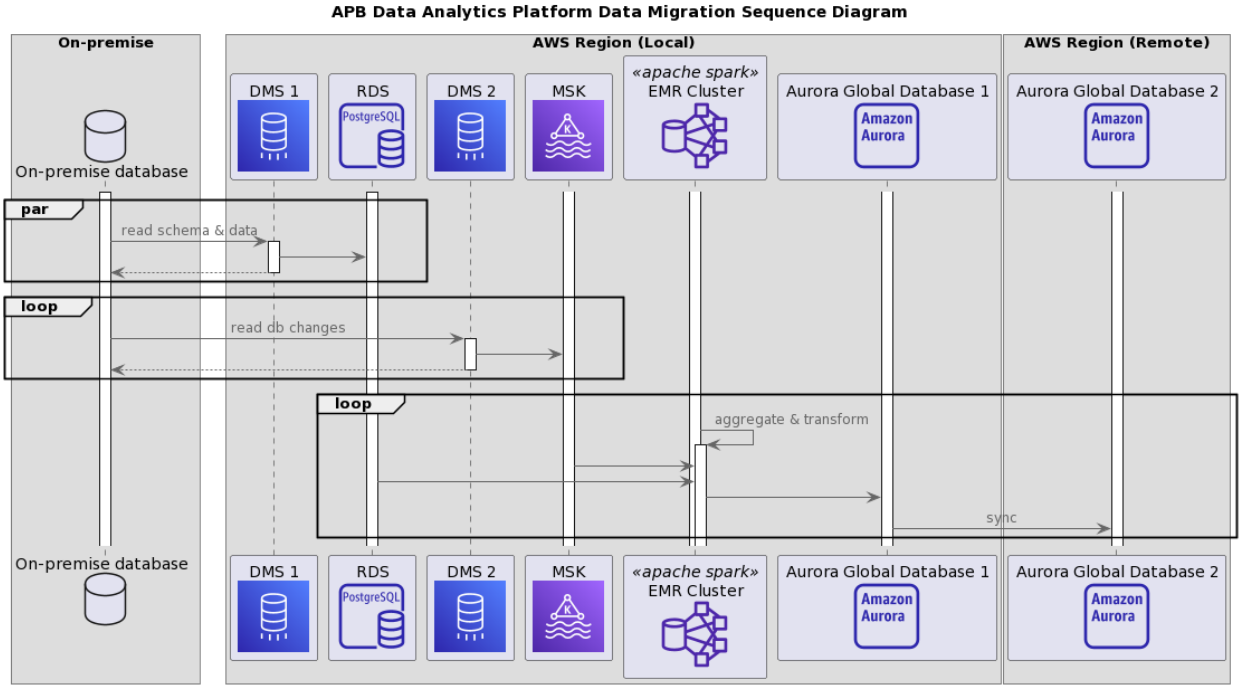
Views

This section describes the view of different aspect of the system. They are,

- Logical views
    - Critical flows of the system
- Process views

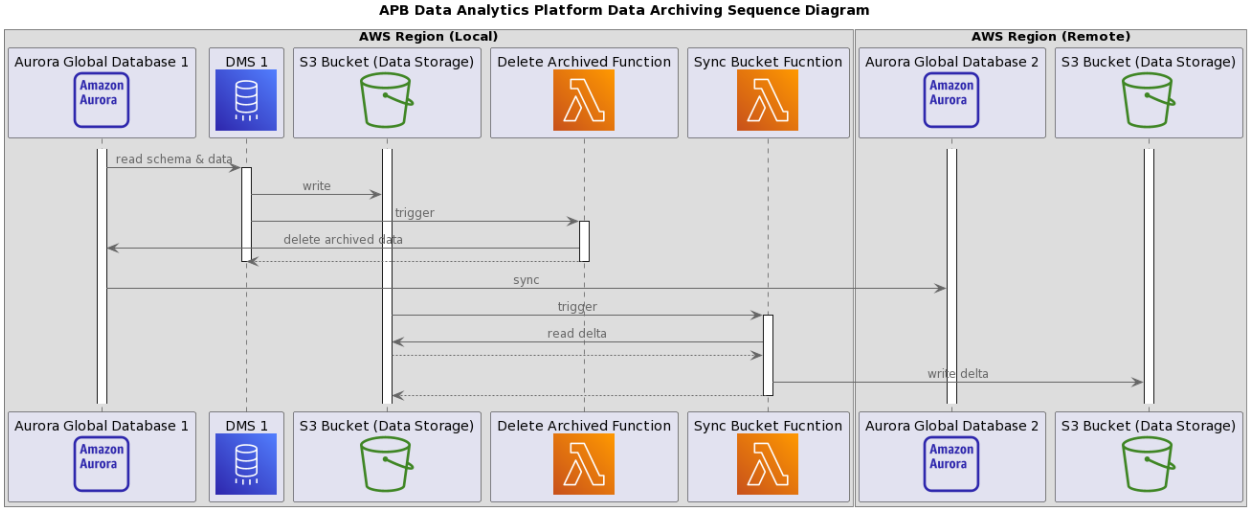
Logical Views

Data migration (On-premise to local AWS region)



- AWS DMS will be used to migrate data of on-premise database to a data bridge database on AWS for ETL use
- AWS DMS will be used to read new changes in on-premise database and produce to AWS managed Kafka instance
- AWS EMR cluster and Apache Spark's structured streaming will be used for ETL to aggregate & transform data to be stored in AWS aurora global database

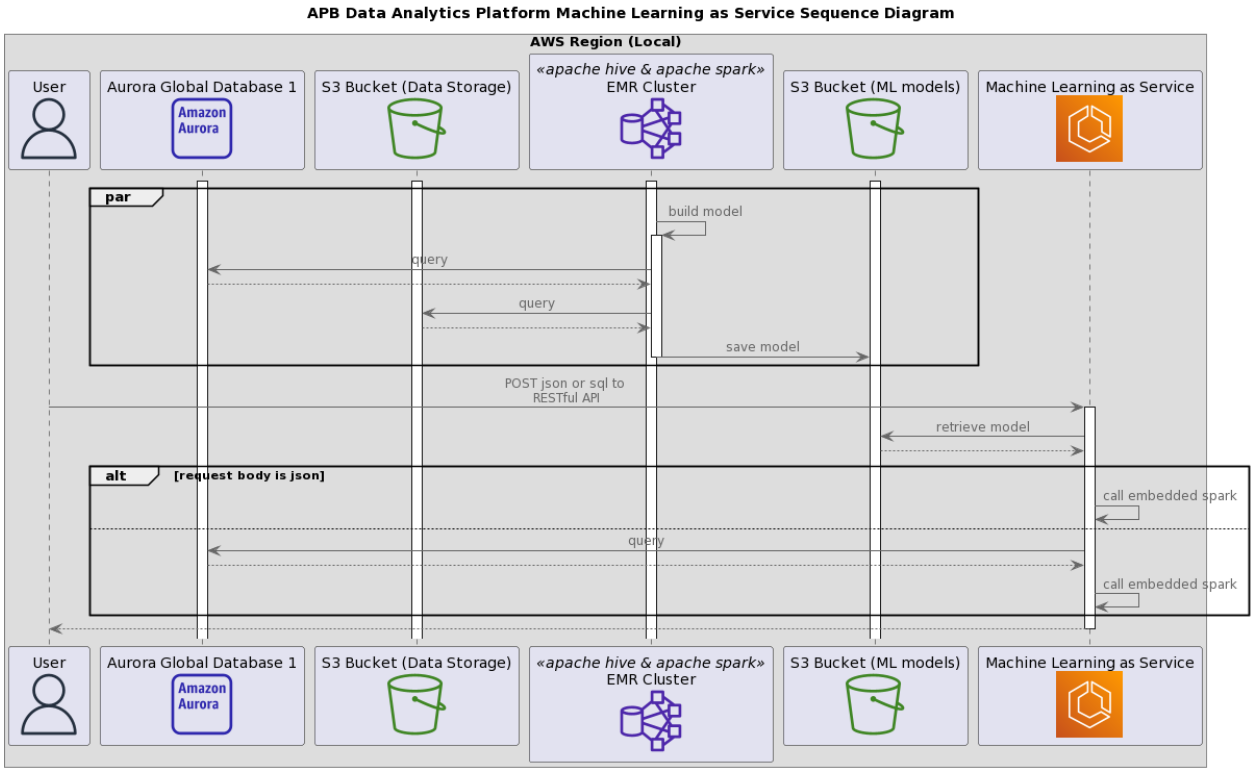
Data archive (& sync to remote region)



- Monthly/Yearly DMS task will be used to archive data to the designated S3 bucket

- When done, DMS task will trigger Lambda function to delete archived data from Aurora Global Database
- After archive is done, S3 bucket will trigger a lambda function to sync to the bucket in remote region
- Archived data can be retrieved by using AWS EMR with Apache Hive or Apache Spark

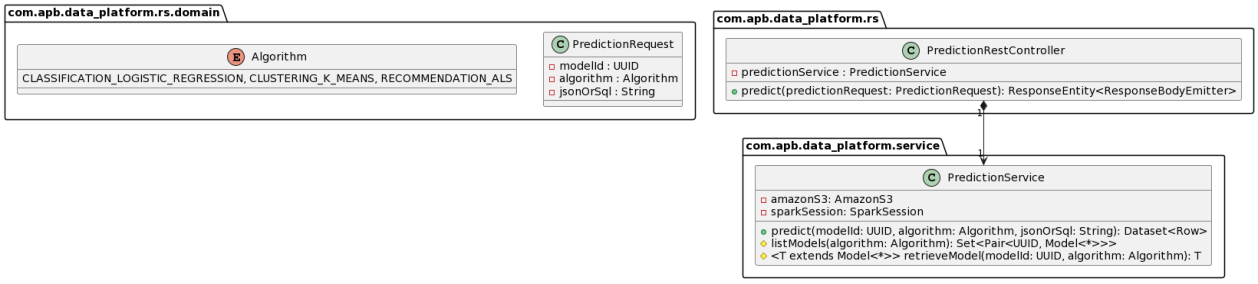
Build ML models & get predictions



- Models will be stored in the bucket according to their algorithm. Each algorithm will have its own folder in the bucket.

Process Views

Get predictions



- If response body is json, use Spark to convert to Dataset for transformation
- If response body is sql, use Spark to retrieve Dataset from Aurora