Serverless Data Lakehouse for Personalized Content Recommendation at Scale

---

Executive Summary

With the exponential growth of digital content, delivering personalized recommendations at scale is a critical differentiator for businesses. Traditional recommendation architectures face challenges with scalability, infrastructure management, and cost-efficiency. A **serverless data lakehouse** approach offers a modern solution: combining the flexibility of a data lake with the performance and structure of a data warehouse—while eliminating the burden of managing infrastructure.

This white paper explores how organizations can leverage a serverless data lakehouse to build a **personalized content recommendation engine** that is scalable, cost-effective, and easy to maintain.

Why a Serverless Data Lakehouse?

Challenges with Traditional Architectures

1. **High Operational Overhead** – Traditional big data architectures require extensive DevOps and infrastructure management.
2. **Data Silos** – Separation between data lakes (raw data) and data warehouses (curated data) leads to slow processing and poor agility.
3. **Scalability Issues** – Recommendation workloads fluctuate; static clusters often result in over-provisioning or performance bottlenecks.

Benefits of Serverless

1. **Automatic Scaling** – Compute resources scale up or down based on workload.
2. **Pay-Per-Use** – Costs align directly with usage.
3. **No Infrastructure Management** – Teams focus on data pipelines and ML models, not server upkeep.
4. **Unified Data Model** – The data lakehouse supports both raw and structured data in one place.

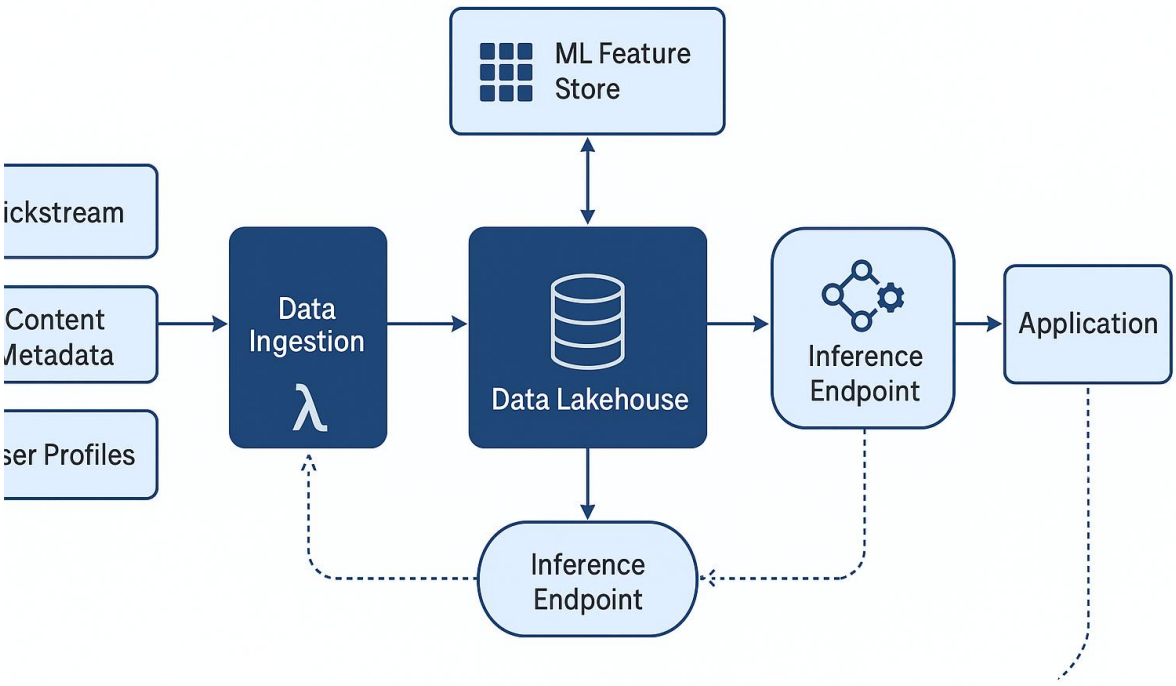
System Architecture Overview

Key components:

- **Data Sources** – Application logs, clickstreams, content metadata, and user profiles.

- **Data Ingestion** – Streaming and batch pipelines using serverless tools like AWS Lambda, Google Cloud Functions, or Azure Functions.
- **Data Lakehouse** – Unified storage layer (e.g., Delta Lake, Apache Iceberg, or AWS Lake Formation).
- **ML Feature Store** – Curated features for recommendation algorithms.
- **Model Training and Inference** – Serverless ML (e.g., AWS SageMaker Serverless, Vertex AI).
- **Recommendation Service** – Exposed via APIs for real-time personalization.

Diagram 1: High-Level Architecture



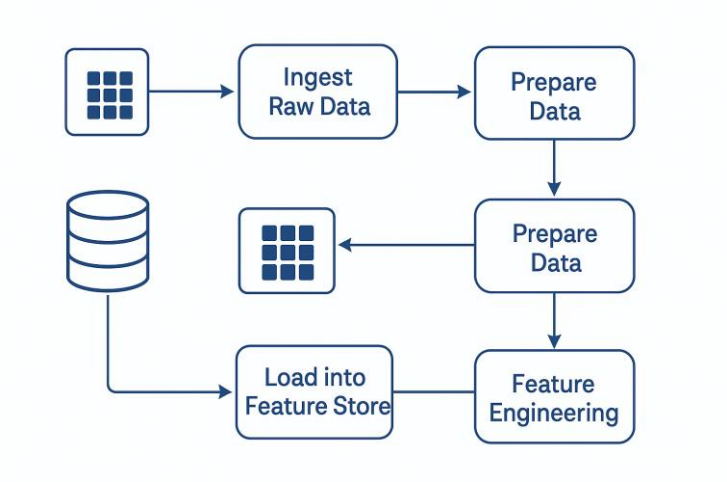
Description:

- Left: Data sources (clickstream, metadata, profile).
- Middle: Serverless ETL functions writing to a data lakehouse.
- Top: ML Feature Store connected to ML Training (serverless).
- Right: Inference endpoint serving recommendations to the application.
- Arrows indicating bidirectional feedback loops.

Data Flow & Workflow

1. **Ingestion Layer**
   Serverless functions capture streaming data (user clicks, views, likes) in near real-time and write them to object storage.
   - Example: AWS Lambda → Amazon S3 → AWS Glue Catalog
2. **Data Processing & Transformation**
   Data is processed using serverless frameworks like Apache Spark on serverless clusters or SQL engines like Amazon Athena/Google BigQuery.
   - Raw → Cleaned → Feature tables
3. **Model Training**
   - Use collaborative filtering and deep learning models (e.g., neural matrix factorization).
   - Training jobs run in a serverless ML environment.
4. **Model Deployment & Serving**
   - Serverless inference endpoints scale automatically based on traffic.
   - Predictions cached for low-latency responses.

Diagram 2: Data Workflow



Description:
A flowchart showing:

- **Input**: user activity →
- **Serverless ingestion** →

- **Data lakehouse** →
- **Feature store & training** →
- **Inference API** →
- **User-facing app**

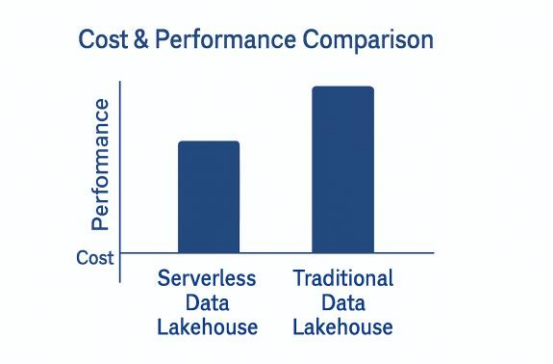
Technologies & Tools

- **Storage**: Amazon S3, Google Cloud Storage, Azure Data Lake
- **Compute**: AWS Lambda, Google Cloud Functions, Azure Functions
- **ETL/Processing**: AWS Glue, Databricks Serverless, BigQuery
- **Model Training**: AWS SageMaker Serverless, Vertex AI
- **Serving Layer**: AWS API Gateway, Cloud Run, Azure Functions
- **Data Governance**: Lake Formation, Unity Catalog

Benefits of This Approach

1. **Scalability**: Automatically handles traffic spikes, e.g., during trending content.
2. **Faster Experimentation**: Quickly retrain models on fresh data without provisioning infrastructure.
3. **Cost Savings**: Pay only for resources used.
4. **Simplified Management**: Less DevOps burden.
5. **Unified Data**: One platform for raw, structured, and machine learning data.

Diagram 3: Cost & Performance Comparison



Description:
Bar chart comparing costs and latency between:

- Traditional big data stack
- Serverless data lakehouse

Use Case Example: Streaming Platform

A streaming platform with millions of active users adopted a serverless data lakehouse:

- Reduced infrastructure costs by **45%**.
- Improved recommendation model freshness from **daily to hourly updates**.
- Achieved **sub-50ms latency** for API calls during peak load.

Conclusion

A **serverless data lakehouse** architecture enables organizations to deliver personalized content recommendations at scale without compromising agility, performance, or cost. By integrating serverless compute, unified storage, and machine learning pipelines, enterprises can modernize their recommendation engines and deliver better user experiences.