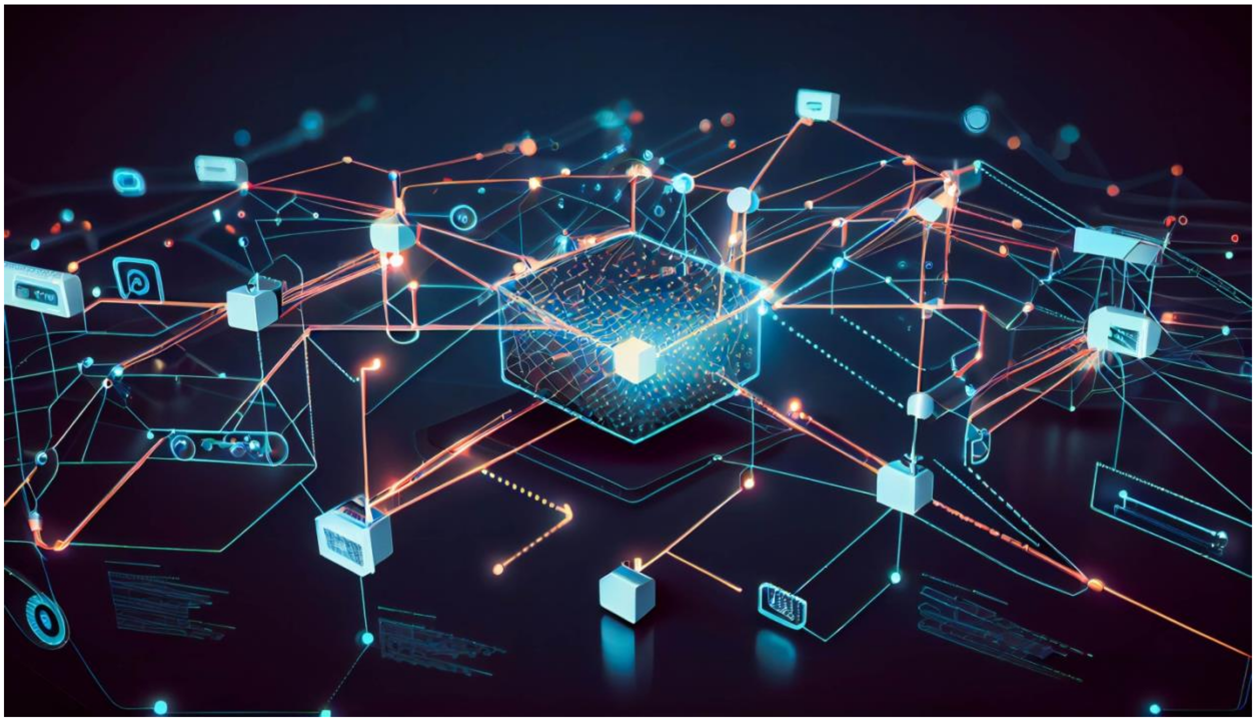


Organizing for Business Analytics Platforms

BUAN 6335.501

Unified Data Platform

Group 05 Report

Meghna Suresh
Shilpa Nidhi Kirubanidhi
Shukla H Shanthakumara

Mallikarjuna Swamy
Manish Ramchandani
Vishruth Acharya

Table of Contents

1 Introduction

Our project aims to upgrade how the university manages data, making it easier for different departments to access and use valuable information. We're creating a Common Data Platform that blends various types of data, such as student records and research outcomes. By exploring cloud services from AWS, GCP, and Azure, we intend to provide a comprehensive solution to the university—not just to work with data but also to leverage the available ML platforms for predictive analytics, ensuring data is secure, easy to access, and stored efficiently. The goal is to give the university a powerful, flexible tool for making smart decisions and encouraging innovation across the board.

1.1 Project Objectives

1.1.1 Common Data Management Platform

- **Objective**: Build a platform that makes it easy for different parts of the university to share and use data effectively.
- **Expectation**:
    - Seamless Data Integration: Make sure different types of data flow smoothly between applications and departments, whether it's daily updates or real-time information through web services and APIs.
    - Single Source of Truth: Create a single, trustworthy version of data with strong controls to maintain quality, track its history, and keep it secure and private.
    - Master Data Management: Keep a continuous record of metadata so that everyone—whether they're technical or not—can easily understand and use the university's data.

1.1.2 Unified Data Store for Analytics

- **Objective**: Set up a central place where the university can store and analyze all kinds of data, making reporting and analytics more powerful.
- **Expectations**:
    - Data Analytics Governance: Put in rules to make sure data is safe, accurate, and consistent, helping the university make important decisions and meet business goals.
    - Analytical Data Models: Speed up the creation of models that help us understand data better, making it easier to use information to make decisions and encourage innovation.
    - Predictive Analytics and AI Use-Case requirements: Use advanced techniques like AI to identify and support new ways to analyze data.

Let’s start with understanding the types of data we will work with and how to process them.

2 Working with Data

2.1 Data Types

Understanding and processing the data types in the university setting is pivotal for any of our decisions. We worked towards and exhaustive list of the available datatypes to ensure we do not miss out on any data.

Student Data:	Enrollment records Academic performance (grades, transcripts) Student demographics Attendance records Admissions data

Faculty and Staff Data:	Faculty profiles Staff employment records Course assignments Research publications and grants

Administrative Data:	Financial transactions (budgets, expenses, grants) Human resources data (employee records, payroll) Procurement and purchasing data Course and Curriculum Data Course offerings and schedules Curriculum details Course materials and syllabi

Research Data:	Research project details Grant information Laboratory data

Library Data:	Book and resource catalog Borrowing and lending records Digital resource access logs Event and Calendar Data: University events and activities Academic calendars

	Scheduling information
Infrastructure Data:	Facility management data IT infrastructure data (network logs, system logs) Learning Management System (LMS) Data: Online course data Student interactions with LMS Assessment and quiz results
Health and Safety Data:	Health records (if applicable) Safety incidents and reports Emergency response data
Alumni Data:	Alumni profiles and contact information Alumni engagement data Donation records
Admissions Data:	Application data Admissions decisions Demographic information of applicants Sensor Data (if applicable): Environmental sensor data (temperature, humidity) Security and surveillance data
External Data:	Data from external partners or collaborations Industry-specific data (if applicable)
User Interaction Data:	Interactions with university websites and portals Feedback and surveys
Financial Aid Data:	Financial aid applications Disbursement records
Communication Data:	Emails, messages, and communication logs Social media interactions

Geospatial Data (if applicable):	Campus maps and layouts
	Geographic information system (GIS) data

Biometric Data (if applicable):	Biometric access logs (for secure areas)

Historical Data:	Historical records and archives
	Previous academic years' data

2.2 Data Processing

Now, it's crucial to know how different types of data should be handled. We sorted each kind of data into two groups: one for regular tasks (like grades) using batch processing, and another for quick updates (like attendance) using real-time processing. By doing this we can create a system that fits the specific needs of the university, making sure the data is managed efficiently.

Batch Processing	Real-Time Processing
Student Data	**Sensor Data (if applicable)**
Example: Processing end-of-semester grades.	Example: Monitoring real-time temperature in a server room.
Explanation: After the semester concludes, batch processing can be used to calculate and update the final grades for all students. This is a periodic task that doesn't require immediate results.	Explanation: Sensors continuously provide temperature data, and real-time processing ensures immediate alerts if the temperature exceeds a threshold.
Faculty and Staff Data	**External Data**
Example: Payroll processing at the end of the month.	Example: Real-time collaboration with an external research partner.
Explanation: Monthly payroll processing involves aggregating work hours, calculating salaries, and updating records. This is typically done in batches to handle all employees at once.	Explanation: If an external partner is contributing real-time data, such as research findings, the platform needs to process and integrate this data as it arrives.
Administrative Data	**User Interaction Data**
Example: Quarterly budget reconciliation.	Example: Live analytics for website interactions.
Explanation: Processing financial transactions and reconciling budgets can be done periodically to ensure accurate financial reporting.	Explanation: Real-time processing can be used to analyze user interactions on the university's website, providing live analytics and insights.

Batch Processing	Real-Time Processing
Course and Curriculum Data	**Communication Data**
Example: Updating course schedules for the next academic year.	Example: Instant notification for critical emails.
Explanation: Batch processing can be used to update and publish course schedules in preparation for the upcoming academic year.	Explanation: Real-time processing can trigger instant notifications for critical emails or messages, ensuring timely responses.
Research Data	**Biometric Data (if applicable)**
Example: Analyzing research grant utilization quarterly.	Example: Real-time access control.
Explanation: Quarterly analysis of research grants involves processing and aggregating data to track fund utilization and outcomes.	Explanation: Biometric access logs can be processed in real-time to grant or deny access to secure areas immediately.

Other Batch Processing examples include:

- **Library Data**:
  - Example: Periodic catalog updates.
    Explanation: Regular updates to the library catalog can be handled in batches to ensure accurate and organized information.
- **Infrastructure Data**:
  - Example: Maintenance schedule planning.
    Explanation: Planning maintenance activities and updating facility management data can be done periodically in batches.
- **Event and Calendar Data**:
  - Example: Publishing the academic calendar.
    Explanation: Batch processing can be used to update and publish the academic calendar for the entire academic year.
- **Learning Management System (LMS) Data**:
  - Example: End-of-semester grading and course evaluation processing.
    Explanation: Processing final grades and analyzing course evaluations at the end of each semester is a batch task.

2.3 Data Sources

Along with identifying the data types and the required processing methods we also wanted to understand the various data sources that we would need to work with. Below is a list of data sources we were able to identify in our research.

- Student Information System
- Human Resources Management System
- Course Management System
- Learning Management System (LMS)
- Laboratory Information Management System
- Legacy Systems
- Social Media Platforms
- Data from Partner Organizations
- Security and Surveillance Systems
- Health Records System
- IT Infrastructure Monitoring Tools

These data sources represent potential systems or databases where the data types are generated, stored, or managed within the university's ecosystem.

Given the above information about the data we are working with we can now go ahead and work on a robust data platform architecture is that is effective for data management and utilization.

3 Architecture

The core idea revolves around establishing a unified data platform that seamlessly integrates and organizes data from various data sources. The platform should aim to provide a centralized hub for comprehensive data management, offering insights into student activities, academic performance, and administrative processes

3.1 Conceptual Approach

We structured the architecture of the University Data Platform into layers, each serving a specific function.

- **Data Ingestion Layer**
  - Gathers data from various sources.
  - Connects with internal systems and external sources.
  - Supports batch, near real-time, and real-time data.

- **Storage Layer**
  - Includes Raw, Clean, and Curated Zones and ensures scalability, flexibility, and resilience.

        - Raw Zone: preserves original, unaltered data for potential attribute sourcing and disaster recovery. It serves as an immutable historical record, enabling the creation of a factual data source.
        - Clean Zone: initiates the initial transformation of raw data, converting it to efficient formats like Parquet or Avro, and performs basic data quality validations. It also serves as an ad-hoc layer for swift responses to unknown queries, facilitating migration to the curated zone.
        - Curated Zone: holds subject-organized data, ready for user and application consumption.
    - The storage layer also acts as a secure repository for historical and curated data.

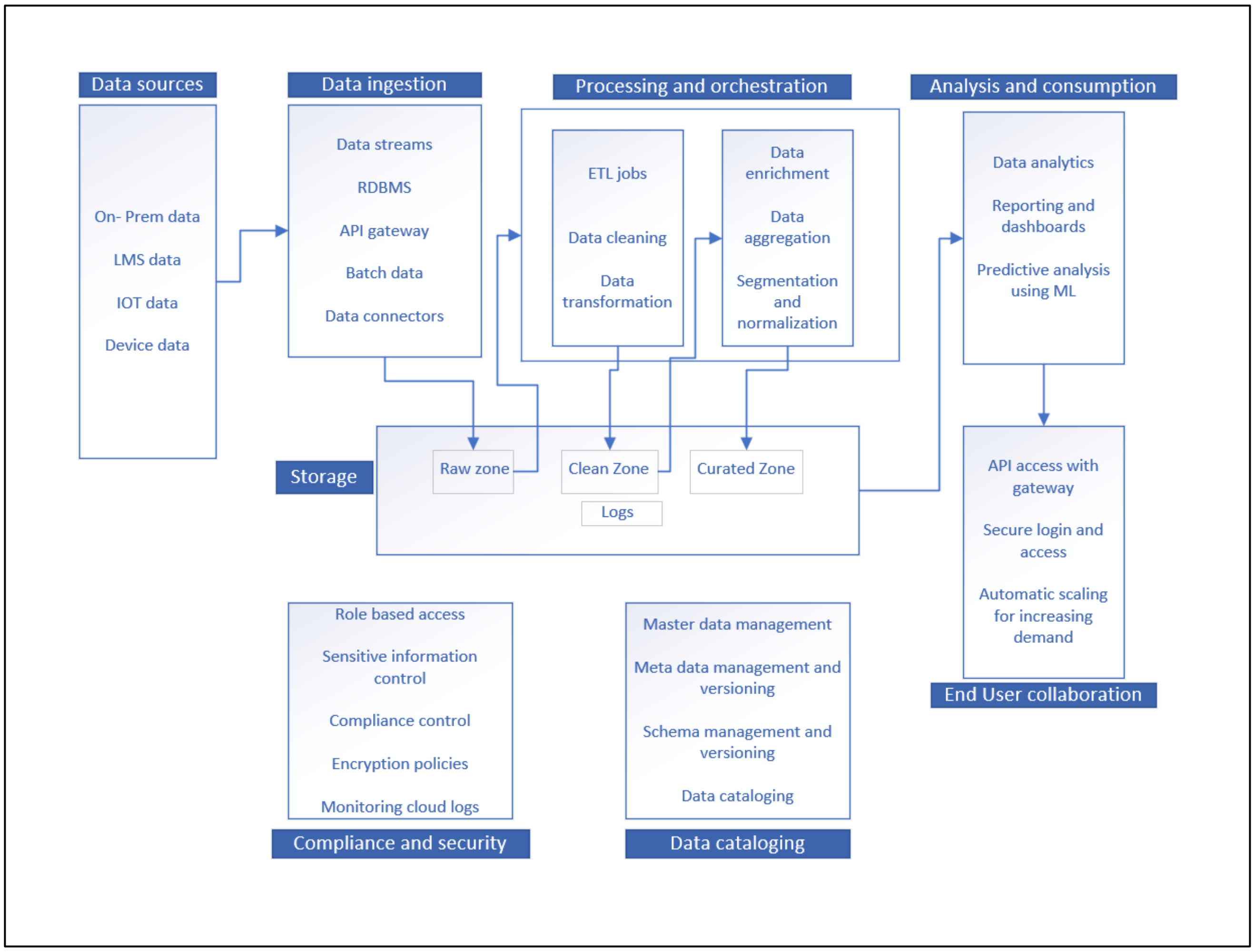


Figure 3.1 Conceptual Architecture Diagram

- **Cataloging Layer**
    - For centralized governance control.
    - Manages data access, versioning, and metadata exploration.
    - Enhances discoverability of datasets for academic and administrative purposes.

- **Processing and Orchestration Layer**
    - Transforms data for easy consumption.
    - Applies business rules for validation, identity resolution, and segmentation.
    - Scales independently to handle increasing data volumes.
- **Analysis and Consumption Layer**
    - Facilitates analytics, data collaboration, and activation.
    - Empowers users with purpose-built tools for reporting, analysis, and decision-making.
- **Compliance and Security Layer**
    - Implements mechanisms for access control, encryption, auditing, and data privacy.
    - Enforces privacy regulations, ensuring responsible data collection.
- **End User Collaboration Layer**
    - Provides access to user with secure login and access.

Having setup, the conceptual architecture for the data platform, we went ahead and explored our implementation options with AWS, GCP and Azure.

3.2 Architecture Implementation

We collated the possible combination of services across all the services available considering the layers and the sub layers from the conceptual architecture.

Layer	Function	AWS	GCP	Azure
Data Ingestion Layer	Batch Processing	AWS Glue for ETL jobs, AWS DataSync for data transfers.	Cloud Dataflow for batch processing	Azure Data Factory for ETL workflows
	Real-time Processing	Amazon Kinesis for stream processing	Cloud Pub/Sub and Cloud Dataflow for stream processing.	Azure Stream Analytics for stream processing
Data Integration	Integration Hub	AWS App Runner or AWS Step Functions	Cloud Composer or Cloud Workflows	Azure Logic Apps
	API Gateway	Amazon API Gateway	Apigee	Azure API Management
Storage	Data Lake	Amazon S3	Cloud Storage	Azure Data Lake Storage
	Data Warehouse	Amazon Redshift	BigQuery	Azure Synapse Analytics

Analysis and Consumption	Analytics Engine	Amazon EMR (for Spark) or Amazon Redshift Spectrum.	BigQuery	Azure Databricks or Azure Synapse Analytics
	Data Modeling	Amazon QuickSight or Tableau	Looker or Tableau.	Power BI
Compliance and Security	Access Controls	AWS Identity and Access Management (IAM)	Identity and Access Management (IAM)	Azure Active Directory
	Data Masking and Encryption	AWS Key Management Service (KMS)	Cloud Key Management Service (KMS)	Azure Key Vault
	Metadata Management	AWS Glue Data Catalog	Cloud Data Catalog	Azure Purview
Data Cataloging	MDM System	AWS CloudFormation for infrastructure as code	Cloud Composer or Cloud Workflows	Azure Logic Apps
		AWS App Runner or AWS Step Functions	Cloud Dataflow	Azure Data Factory
Monitoring and Optimization	Monitoring Tools	Amazon CloudWatch	Cloud Monitoring	Azure Monitor
	Optimization Strategies	AWS Trusted Advisor	Cloud Cost Management	Azure Cost Management and Billing
Scalability and Flexibility	Cloud Services	Amazon EC2 for compute, Amazon RDS for databases	Compute Engine, Cloud SQL	Azure Virtual Machines and Azure SQL Database
	Microservices Architecture	AWS Lambda and Amazon ECS	Cloud Functions and Kubernetes Engine	Azure Functions and Azure Kubernetes Service.

Taking advantage of our familiarity with the AWS services we were able to set up the architecture.

4 Solution Implementation

As mentioned in the above section we were able to plan an implementation strategy using the AWS Services. However, given the problem statement we were able to come up with a cloud solution as well as a hybrid solution.

4.1 Hybrid Solution

Our hybrid solution seamlessly integrates on-premises infrastructure with cloud services, offering a versatile approach to meet diverse business needs. It optimizes performance, security, and scalability by combining the reliability of existing on-premises resources with the agility and cost-effectiveness of the cloud. This synergy will allow the university to leverage the advantages of both environments, ensuring a flexible and efficient IT infrastructure.

4.1.1 Data Source

Service Used: Amazon RDS

Use case example:
Cloud based solution for databases that are currently residing on-premises (Some examples include Faculty / staff data, student data, administrative data and library data)

Why Amazon RDS?
RDS is a managed DB service (Ease of use). Compatible with Db2, MariaDB, Microsoft SQL Server, MySQL, Oracle, and PostgreSQL. Multi - AZ option for high availability.

4.1.2 Data Ingestion

AWS EC2 ***API Gateway*** ***AWS Appflow*** ***AWS CloudSync*** ***AWS Kinesis***

Use Case examples and reason for the choice of each service:

- **AWS EC2 instance with EBS** - Block level storage volumes for use with EC2 instances
  Why?
  Well suited for DB applications that require random reads and writes, and high throughput intensive applications

- **AWS Kinesis streaming data platform** (Kinesis data streams, Kinesis video streams) - Delivering real-time streaming data to S3 targets.
  Video streams: Synchronous learning/ live lecture streaming for completely online courses
  Data streams: real-time updates from LMS system (Ex: Assignment submissions, deadlines, grades submissions, etc.), bursar payments, payroll transactions updates for faculty.
  Why?
  Fully managed service, with no upfront costs; resources that need to be provisioned; or minimum fees. Pay only for the volume of data ingested , stored and processed.
- **AWS API gateway** - API access to LMS (e-learning) data
  Why?
  Fully managed service, for applications to access data, business logic, or functionality. Provides RESTful APIs using REST and HTTP APIs (HTTP is up to 71% cheaper than REST APIs). Offers tired pricing models for cost optimization.
- **AWS Appflow** - Automate integration with SaaS application from other vendors
  Why?
  Automate bi-directional data flows between AWS and SaaS applications. Configurable to many SaaS applications such as Salesforce, SAP, Google Analytics, Facebook Ads, and ServiceNow, and Amazon services like S3 , Redshift in just a few clicks.
- **AWS Datasync** - Data transfer from on-prem sources to the cloud online.
  Why?
  Simplifies data transfer, and lesser physical infrastructure compared to DirectConnect and VPC endpoints.

4.1.3 Data Processing and Orchestration

AWS Step Functions ***AWS Batch*** ***AWS Glue*** ***AWS Lambda***

Use Case examples and reason for the choice of each service:

- **AWS Lambda** - Smaller loads of batch processing (daily) which can be processed under 15 mins. (Change Data Capture -smaller loads Ex: Daily attendance information, daily library book-keeping section-wise)
  Why?
  Serverless (easy to use) and event-driven compute service. Cost effective compared to AWS Glue and AWS Batch.
- **AWS Batch** - General batch processing like bursar accounts, payroll for faculty etc. (CDC - bigger loads).
  Why?

More suitable for batch computing jobs that run as per a fixed schedule and are compute-intensive.

- **AWS Glue** - Glue crawlers for processing student data, glue workflows for real-time data and ETL batch processes
  Why?
  Parallel processing with extreme performance, and well-suited for ETL batch processing.
- **AWS Step functions** - Modular structure and orchestration service in the architecture.
  Why?
  More decoupled workflows that handle parallel processing, error handling and branching.

4.1.4 Storage

AWS S3 ***AWS S3 Glacier***

Use Case examples and reason for the choice of each service:

- **AWS S3** - Object storage in cloud. We use separate buckets for the raw data, clean data, and curated data. Logs can also be stored.
  Why?
  Object storage service offering industry-leading scalability, data availability, security, and performance. Offers cost-effectiveness with tiered pricing. Backup and storage for failover recovery.
- **AWS S3 Glacier** - Archiving the older university data which are not frequently used.
  Why?
  Lowest cost archive storage in the cloud. Virtually unlimited scalability and are designed for 99.999999999% (11 nines) of data durability.

4.1.5 Data Consumption and Analytics

AWS Redshift ***AWS QuickSight*** ***AWS Athena*** ***AWS EMR*** ***AWS OpenSearch***

Use Case examples and reason for the choice of each service:

- **AWS EMR** - Big data processing use cases that require map reduce.
  Ex: GPA calculation for all students, dean's list recommendations, top 10% ranking department wise, scholarship list, faculty promotion list based on several parameters. Can also be used in discovering hidden patterns, trends etc.

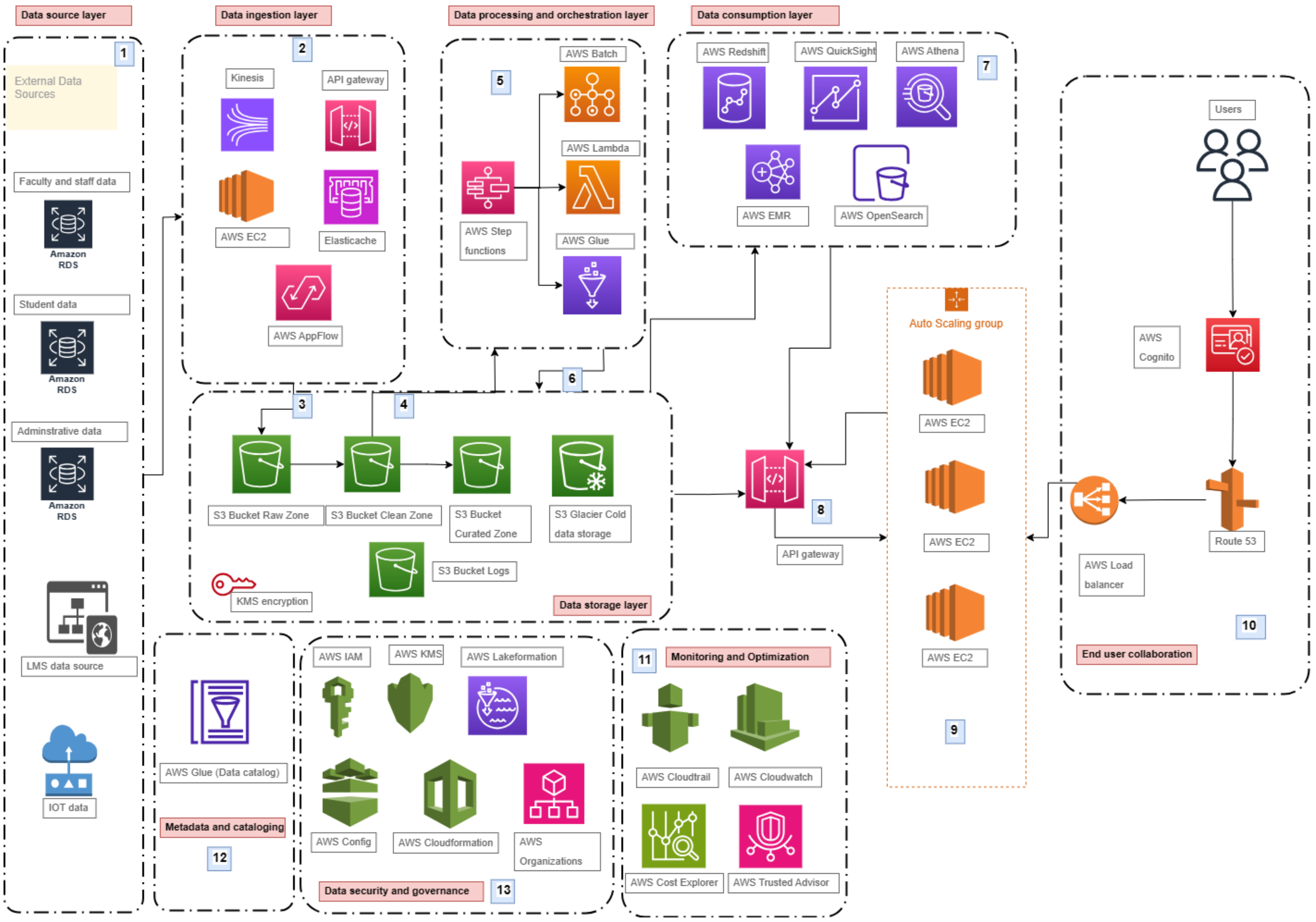


Figure 4.1 Hybrid Architecture

Why?
Provides options to run on customized Amazon EC2 clusters, Amazon EKS, AWS Outposts, or Amazon EMR Serverless. 2X faster time-to-insights with performance-optimized and open-source API-compatible versions of Spark, Hive, and Presto.

- **AWS Redshift** - Cloud data warehouse for structured and semi-structured data.
  Why?
  Break through data silos and gain real time and predictive insights on all the data, with no data movement or data transformation
- **AWS QuickSight** - Dashboarding and reporting tool for the university data. Student performance dashboard, course-wise grading trends, tuition fee payment trends, under-performing classes, etc.
  Why?
  Meet varying analytic needs from single source of truth through modern interactive dashboards, paginated reports, embedded analytics, and natural language queries.
- **AWS Athena** - Analyze data or build applications(to analyze and visualize data) from an Amazon Simple Storage Service (S3) data lake. Metrics analysis, student count analysis department-wise.
  Why?
  Can connect to 30 types of data sources including on-premises data sources or other cloud systems. Integrate to ML tools for decision making.
- **AWS Opensearch** - Application and Infrastructure Monitoring (Log analysis and visualization)
  Fast, Scalable Full-text Search (ex: course-based or faculty-based searches, book searches)
  Security and Event Information Management (Intrusion detection, cheat/ fraud detection)
  Operational Health Tracking (App health)"
  Why?
  Scalable, secure and provides visualization of log data.

4.1.6 Data security and governance

AWS IAM *AWS KMS* *AWS Lakeformation* *AWS Config* *AWS Cloudformation* *AWS Organizations*

Use Case examples and reason for the choice of each service:

- **AWS IAM** - Securely control access to AWS resources. Control who is authenticated (signed in) and authorized (has permissions) to use resources.
  Why?
  Granular permissions, integrated security features, free to use
- **AWS KMS** - Create and control the cryptographic keys that are used to protect the data at rest and in transit.
  Why?

AWS KMS integrates with most other AWS services that encrypt data. AWS KMS also integrates with AWS CloudTrail to log use of KMS keys for auditing, regulatory, and compliance needs.

- **AWS Lakeformation** - Centrally govern, secure, and globally share data for analytics and machine learning. Manage fine-grained access control for the data lake data on S3 and the metadata in AWS Glue Data Catalog. Integrating Lake Formation with Amazon Redshift data sharing – Lake Formation is used to centrally manage database, table, column, and row-level access permissions of Amazon Redshift datashares and restrict user access to objects within a datashare.
  Why?
  Lake Formation helps you break down data silos and combine different types of structured and unstructured data into a centralized repository. First, identify existing data stores in Amazon S3 or relational and NoSQL databases, and move the data into your data lake. Then crawl, catalog, and prepare the data for analytics. Next, provide your users with secure self-service access to the data through their choice of analytics services.
- **AWS Config** - Detailed view of the configuration of AWS resources in the AWS account. See how the resources are related to one another and how they were configured in the past so that we can see how the configurations and relationships change over time.
  Why?
  Deep analysis of all the services used. Can be tuned for performance and cost optimization.
- **AWS Cloudformation** - Build templates of the workflows. Can be replicated in the future for similar needs.
  Ex: Build a template for course feedback for in-person courses and repeat it for online courses.
  Why?
  Templates can lead to reduction in setup time for similar workflows. Leads to savings in time, effort, and money.
- **AWS Organizations** - Account management service that enables you to consolidate multiple AWS accounts into an organization that we create and centrally manage. AWS Organizations includes account management and consolidated billing capabilities that enables us to better meet the budgetary, security, and compliance needs. Ex: Vendors can have accounts in this service, and we can use consolidated billing. Can be integrated with IAM to provide specific access categories to vendors.
  Why?
  The significance of SCP: An organization policy that you can use to manage permissions in your organization. SCPs offer central control over the maximum available permissions for all accounts in your organization.

4.1.7 Metadata and Cataloging

AWS Glue Data Catalog

Use Case example:
Centralized metadata repository for all data assets across various data sources. It provides a unified interface to store and query information about data formats, schemas, and sources.
Ex: Master data and metadata management of the database, including versioning.

Why AWS Glue Data Catalog?
Glue data catalog comes with ETL Glue as part of master data management. When an AWS Glue ETL job runs, it uses this catalog to understand information about the data and ensure that it is transformed correctly.

4.1.8 Monitoring and Optimization

AWS Cloudtrail ***AWS Cloudwatch*** ***AWS Cost Explorer*** ***AWS Trusted Advisor***

Use Case examples and reason for the choice of each service:

- **AWS Cloudtrail** - Auditing, security monitoring, and operational troubleshooting. CloudTrail records user activity and API calls across AWS services as events. CloudTrail events helps to answer the question of "Who did what, where, and when?"
  Why?
  With AWS CloudTrail Lake, we can consolidate activity events from AWS and sources outside AWS — including data from other cloud providers, in-house applications, and SaaS applications running in the cloud or on premises — without having to maintain multiple log aggregators and reporting tools.
- **AWS Cloudwatch** - Monitors the AWS resources and the applications that run on AWS in real time. We can collect and track metrics. Ex: service-wise CPU utilization, performance metrics etc.
  Why?
  We can create alarms that watch metrics and send notifications or automatically make changes to the resources we monitor, when a threshold is breached. For example, we can monitor the CPU usage and disk reads and writes of an Amazon EC2 instance and then use that data to determine whether you should launch additional instances to handle increased load. We could also use this data to stop under-used instances to save money.
- **AWS Cost explorer** - View and analyze costs and usage. Budgets, expenditure forecast based on the services used, 12 months of historical spending data and other such features can help in cost

optimization and improving value for money.
Why?
View costs and usage using the Cost Explorer user interface free of charge. Graphical representation of costs is easy to comprehend.

- **AWS Trusted advisor** -
  Inspects the AWS environment, and then makes recommendations when opportunities exist to save money, improve system availability and performance, or help close security gaps.
  Why?
  Recommendations for: cost optimization, performance, security, fault tolerance, service limits and operational excellence.

4.1.9 End User Collaboration

AWS EC2 with Autoscaling ***AWS Load balancer*** ***Route 53*** ***AWS Cognito***

Use Case examples and reason for the choice of each service:

- **AWS EC2 with autoscaling** - Amazon EC2 Auto Scaling ensures that we have the correct number of Amazon EC2 instances available to handle the load for the application. Collections of EC2 instances, called Auto Scaling groups are created. Specify the minimum number of instances in each Auto Scaling group, and Amazon EC2 Auto Scaling ensures that the group never goes below this size.
  Why?
  EC2 pros:
  - Flexibility - EC2 comes with hardware for any workload requirements, with support for major operating systems. A wide variety of instance types are offered, and the choice can be done based on our specific needs.
  - Scalability - Instances can be scaled up and down horizontally and vertically.
  - Cost-effectiveness: EC2 instances can be started and stopped on-demand. Savings plan pricing enables cost effectiveness.
- **AWS Load balancer** - Automatically distributes incoming traffic across multiple targets, such as EC2 instances, containers, and IP addresses, in one or more Availability Zones. It monitors the health of its registered targets, and routes traffic only to the healthy targets. Elastic Load Balancing scales your load balancer as incoming traffic changes over time. It can automatically scale to most workloads.
  Why?
  Classic load balancer provides load balancing in both layers 4 and 7. Supports protocols TCP, SSL/TLS, HTTP, HTTPS. If specific layer-based load balancing is needed, choose network load balancer for layer 4, and application load balancer for layer 7.
- **Route 53** - Highly available and scalable Domain Name System (DNS) web service.
  Why?

Perform three main functions in any combination: domain registration, DNS routing, and health checking.

- **AWS Cognito** - An identity platform for web and mobile apps. It's a user directory, an authentication server, and an authorization service for OAuth 2.0 access tokens and AWS credentials. Authenticate and authorize users from the built-in user directory, enterprise directory, and from consumer identity providers like Google and Facebook.
  Why?
  Secure sign-in and authorization for end users.

4.2 Cloud Solution

We also have an implementation strategy for a complete cloud solution. Here, we optimize our operations by utilizing AWS services to construct a fully cloud-native environment. This means that all our digital processes are tailored to work seamlessly in the cloud, allowing us to easily expand as needed, adapt to changing requirements, and do so in a cost-effective manner. It's like having a dynamic and efficient digital space, thanks to the capabilities provided by AWS services.

The implementation strategy for the cloud-native implementation is very similar to the hybrid solution except for a one-time migration strategy and a small change in the data ingestion layer that is shown below.

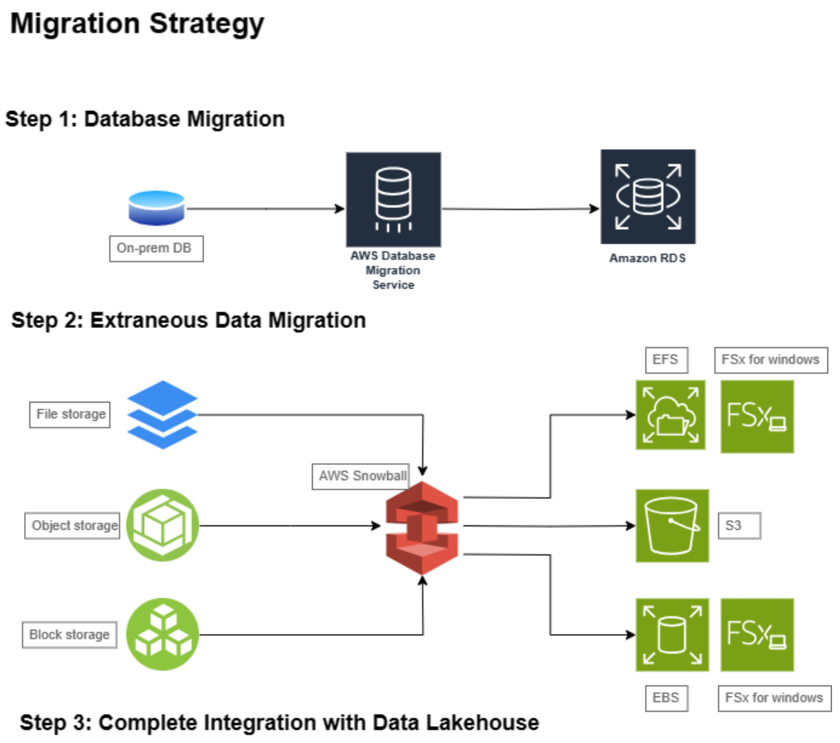

Migration Strategy
Step 1: Database Migration
On-prem DB
AWS Database Migration Service
Amazon RDS
Step 2: Extraneous Data Migration
File storage
Object storage
Block storage
AWS Snowball
EFS
FSx for windows
S3
EBS
FSx for windows
Step 3: Complete Integration with Data Lakehouse

Figure 4.2 Migration Strategy

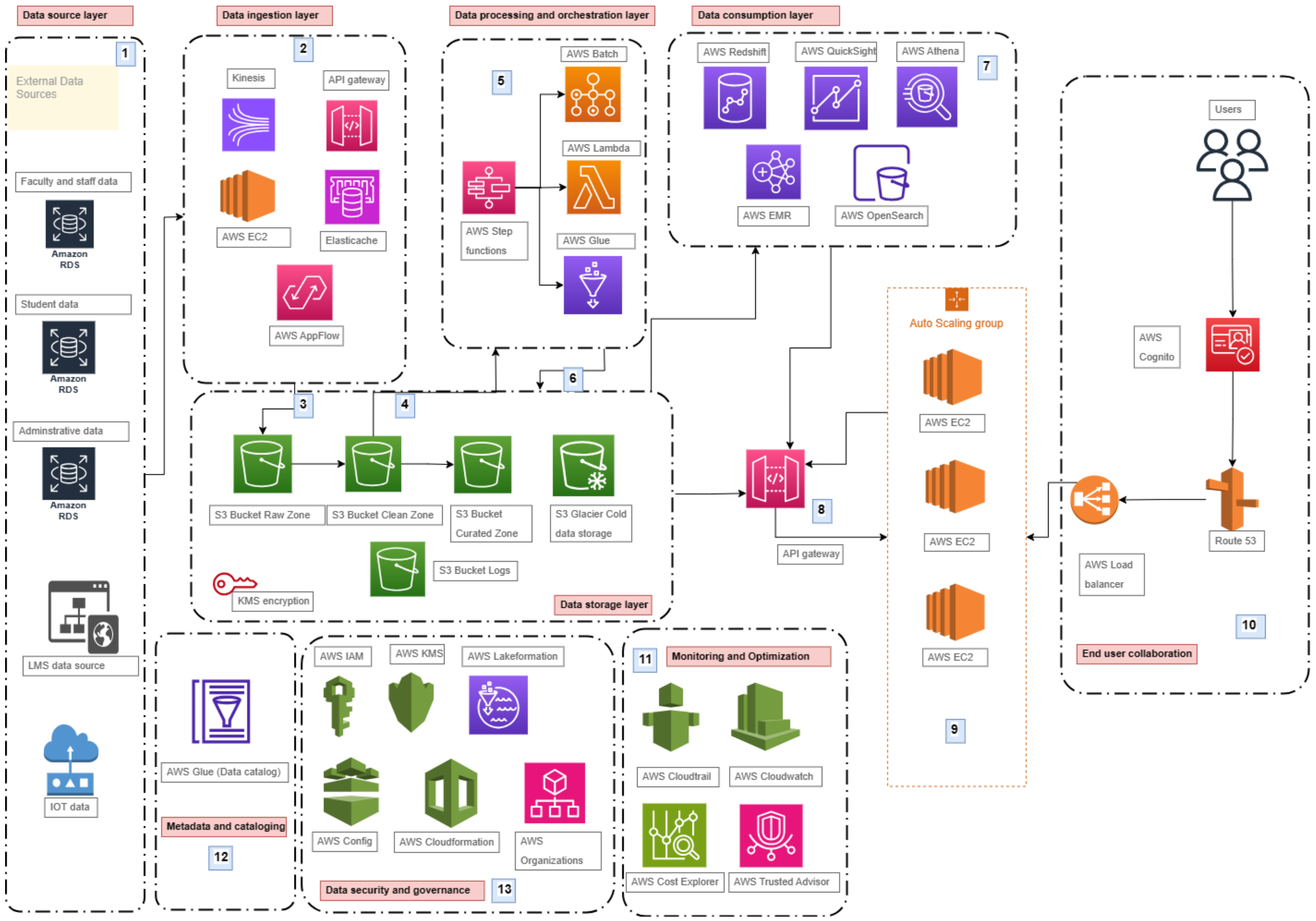


Figure 4.3 Cloud Solution Architecture

4. 3 ML Use Case

This section discusses in brief two ML uses cases that can be implemented using the available AWS Services.

Predictive Analytics for Student Success

Implementation Steps:

1. **Data Collection and Preprocessing in S3:**
    - Collect historical data on student performance, attendance records, and extracurricular activities.
    - Preprocess the data in S3 to handle missing values, scale numerical features, and encode categorical variables.
2. **Dataset Splitting and SageMaker Setup:**
    - Split the dataset into training and testing sets.
    - Create an **S3 bucket** to store training data and model artifacts.
3. **Model Training with SageMaker:**
    - Navigate to the Amazon SageMaker console.
    - Choose a suitable algorithm for predictive analytics (e.g., XGBoost, LinearLearner).
    - Utilize **SageMaker's built-in algorithms** or bring your own.
    - Train the model using the designated training dataset.
4. **Model Deployment:**
    - Deploy the trained model to a **SageMaker endpoint**.

Alumni Donation Prediction:

How SageMaker Facilitates:

- **Prediction Objective:**
    - Predict the likelihood of alumni making donations to the university based on historical donation patterns and alumni engagement.
- **SageMaker Implementation:**
    - Develop predictive models using SageMaker for thorough analysis of alumni data.
    - Predict the probability of donation, enabling targeted fundraising efforts.

<u>NOTE:</u> This implementation strategy uses Sagemaker. This is not represented in our actual solution implementation.

5 Critical Implementation Considerations

In shaping our data strategy, we had to consider few important factors. This section describes how we organized our data, selected scaling methods, planned for efficient data storage, and made data accessible quickly. This discussion simplifies these technical considerations, helping us build a solid and efficient data foundation.

5.1 Why Data Lakehouse?

It allows for coherent, complete data management rather than through silos/parts. Allows for the integration of data from everyone within the organization while managing security and governance of the data. Lake connected to data warehouse where lake acts as single source of truth that carries all variety of data in raw and transformed stages whereas redshift holds a subject oriented comprehensive structured repository of data for historical analysis and visualizations. This lakehouse is the modern data strategy for management that facilitates a data-driven approach for organizations such as universities.

Modern data strategy is dependent on 5 pillars:

- Scalable data lakes – Data lakes on S3 can store variety of data including structured, unstructured, and semi-structured data, extremely scalable and durable storage platform (acts as a single source of truth)
- Performance and cost optimization – Availability of 99.99% and extremely cost effective through employment of lifecycle policies that are determined based on access patterns
- Serverless and ease of use – Glue ETL part of the data pipeline allows for extreme ease of use as well as being designed to be independent of any infrastructure. Cost dependent only on the time that the service is in use.
- Unified data access, security, and governance – ability to tag, catalog and keep data in sync while ensuring data is secure and there is clear governance in place through Lake Formation, Glue Data Catalog, etc.
- Built in machine learning and AI growth – ML integration is built into many Analytical services by using SQL commands. ML integration is provided for all different types of data within the data lake. ML integration is available through Amazon Athena, Redshift and Quicksight.

5.2 Optimization Techniques and Enhancements

In this section, we explore strategies to enhance our data system's efficiency

5.2.1 S3 Lifecycle Policies

- Ensure lifecycle policies are maintained to manage the data most cost effectively throughout their lifecycle
- Data from the last 10 years (2013 to Current) – Lifecycle policy will consist of 30 days in S3 Standard then transitioned to S3 Intelligent Tiering which will allocate the different transitions for the data based on access patterns

- Data from the first 15 years (1998 to 2013) – Lifecycle Policy will consist of 30 days in S3 Standard then transitions to S3 Glacier Flexible Retrieval for 90 days then S3 Glacier Deep Archive where the data is archived for long term purposes. If needs to be recovered, the data will be restored.
- Log Data from the data management system - Lifecycle policy will consist of 30 days in S3 Standard then transitioned to S3 Intelligent Tiering which will allocate the different transitions for the data based on access patterns.

5.2.2 Sharding/Horizontal Partitioning

- Scale out approach for relational databases that allows for high scalability, availability, and fault tolerance for data storage. Each instance of the database is considered a shard. Sharding splits the complete dataset into smaller subunits either through vertical partitioning or horizontal portioning. Vertical partitioning would mean that each shard contains all the data for a particular field whereas horizontal partitioning means that the dataset is split horizontally with a partition key such as student ID and a set of students' IDs will be present in each shard. There will also be an association table that contains metadata such as which Student IDs are mapped to which shards.
- Best part of Sharding – no single point of failure, if one shard is affected the other shards are not affected. Queries can have consistent performance throughout by employing different shards.
- Drawback: Higher latency in playing an active role in analytical environments where data analytics is applied to the dataset. Therefore, we are incorporating these shards in OLTP scenarios rather than OLAP scenarios.
- Incorporating Sharding in daily transactional databases such as bursar payments, employee and faculty payroll processing, and course registration deadlines, etc.

5.2.3 Horizontal Scaling

Horizontal scaling or scaling out is when the system increases capacity by adding additional instances/servers to the system. Horizontal systems act like distributed/parallel processing models. In the case of the EC2 servers, we have employed auto scaling groups that immediately scale out and add more instances when the load is too high. We can configure the scaling groups to state that if there is a load of 70% or higher, to scale out or add instances.

5.2.4 Caching

Caching supplements the database by eliminating the excessive load of the database by having a readable form of frequently accessed data. A local cache stores it as part of the database instance therefore when the database goes down, the local cache is lost as well. It is important to employ a hybrid or two-tier caching strategy that leverages both local and remote cache together.
The specific service of Amazon ElastiCache leverages remote caches for caching relational database data. When there are frequently accessed data and queries, these queries go into the ElastiCache to retrieve the data instead of the database instance thereby eliminating the additional load on the database.

5.2.5 CAP Theorem

CAP Theorem states that a truly distributed system is not able to provide all the guarantees of the theorem such as consistency. Availability, and partition tolerance.

- Consistency refers to the fact that every read request is responded with the most recent update from a write operation or if it cannot deliver the response an error is shown. In the case of the current sharded architecture we have, it is important to note that we will have eventual consistency, but maybe not immediate consistency. There may be some errors or inconsistencies in the read data immediately after a write operation.
- Availability is when every request is received no matter if the nodes are up and running or are faulty. If each request is received and responded to that show cases high availability. Some metrics to consider are if a system is down, how long is it down or how long does it take to bring the system back on. In order to accommodate high availability, the system has a Active-Passive multi-region deployment where there is automatic syncing from the active deployment with the passive deployment, and until the active deployment goes down the passive deployment will not be routed to. Also, in the case of the RDs, we have also employed multi-AZ deployment which ensures if there is a disaster in one availability sone, the other one is always up and running and ready to be the failover option.
- Partition tolerance is when the system can continuously work even though there might be issues with communication between several nodes. In our system, we have employed sharding with multi-AZ deployment which ensures, no matter what, there will be instances available and the system will continue to operate without any hiccups.

5.2.6 Encryption (At rest and In Transit)

- Encryption ensures that no unauthorized user accesses the data without the appropriate encryption keys. It is extremely important to always encrypt data during rest and while in transit. Data encryption offers server-side and client-side encryption. Server-side encryption is where you request the AWS service to encrypt the data before saving it within their data centers and decrypt it when you download the objects whereas client-side encryption is when you encrypt your data yourself and upload the encrypted data to the storage layers. In the client-side process, you are responsible for managing the encryption process as well as the keys. The server-side process provides you with AWS KMS which is the Key management Service that manages all the keys that you may have.
- In rest – Server-side encryption seamlessly using S3, a unique 256-bit key is leveraged to encrypt the data with AES-256. The 256 generated key is masked with a master key from KMS. 256 bit key is stored with the encrypted data and the details of the objects will reveal the SSE with the alias of the KMS master key that was used to mask the original key.
- In Transit – Use TLS (Transport layer Security) to help ensure the data is secure while transit

5.2.7 Data Masking with PII for University data

The system that we propose will utilize AWS Glue Data Brew to detect and mask Personally Identifiable Information (PII) while performing the necessary transformations. In the case of the university, this will be along the lines of Social Security Number, etc. These data masking transformations that are available include substitution, hashing, encryption, decryption and many more. Furthermore, all this is available on a low-code platform. Masked data will be leveraged for any analytics, machine learning, reporting, etc.

5.2.8 Infrastructure Management through CloudFormation (IaaS)

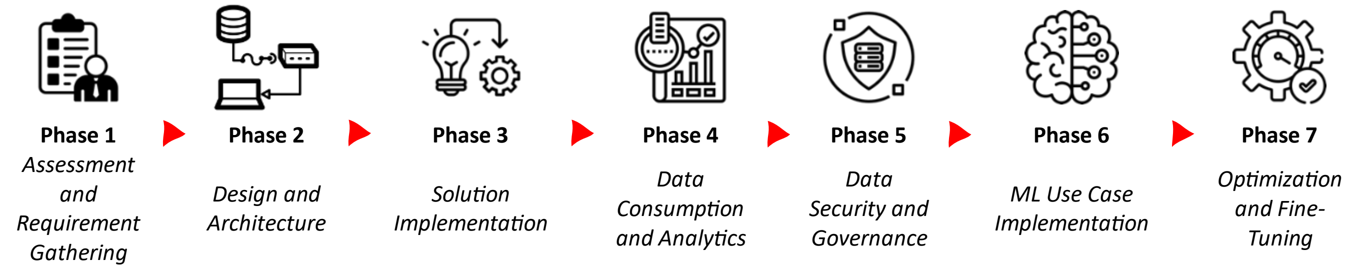
The organization's ability to effectively manage, scale and provision cloud computing resources is a huge deal for any organization that leverages cloud services. CloudFormation is the service that allows infrastructure management within an organization by defining the entire infrastructure for the system through code. This allows the organization to not worry about small, granular changes and configurations. Some benefits this provides include automation, agility, scalability, consistency, and standardization. CloudFormation through ready-made templates (act as blueprints for the resources and configurations) allow organizations to manage these resources in an agile and organized fashion.

6 Project Road Map and Cost Estimation

In this section we discus briefly about a possible project plan and cost estimate.

6.1 Project Road map

Our project plan focuses on a phased approach to ensure a smooth and efficient implementation of the proposed data infrastructure optimization techniques. The blueprint outlines key milestones, tasks, and timelines for each phase.



Phase 1
Assessment and Requirement Gathering
Phase 2
Design and Architecture
Phase 3
Solution Implementation
Phase 4
Data Consumption and Analytics
Phase 5
Data Security and Governance
Phase 6
ML Use Case Implementation
Phase 7
Optimization and Fine-Tuning

Phase-wise Breakdown:

- Assessment and Requirement Gathering – 4 Weeks
- Design and Architecture – 4 Weeks
- Cloud Solution Implementation – 8 Weeks

    - Deploy the hybrid solution, integrating on-premises and cloud services.
    - Implement Amazon RDS for cloud-based databases, utilizing multi-AZ options for high availability.
    - Configure AWS EC2, API Gateway, AWS Appflow, AWS CloudSync, and AWS Kinesis for data ingestion.
    - Set up AWS Lambda, AWS Batch, AWS Glue, and AWS Step Functions for data processing and orchestration.
    - Implement AWS S3 and AWS S3 Glacier for efficient storage, considering data lifecycle policies.
- Data Consumption and Analytics – 6 Weeks
    - Deploy AWS Redshift, AWS QuickSight, AWS Athena, AWS EMR, and AWS OpenSearch for data analytics.
    - Configure specific use cases for each service (e.g., GPA calculation, student performance dashboards, analytics for alumni donations).
    - Develop custom applications and dashboards for end-user consumption.
- Data Security and Governance – 4 Weeks
    - Implement AWS IAM, AWS KMS, AWS Lakeformation, AWS Config, AWS CloudFormation, and AWS Organizations for data security and governance.
    - Define access controls, encryption key management, and fine-grained data lake access.
    - Set up monitoring and optimization tools like AWS CloudTrail, AWS CloudWatch, AWS Cost Explorer, and AWS Trusted Advisor.
- ML Use Case Implementation – 6 Weeks
    - Develop and implement predictive analytics models using SageMaker for student success and alumni donation prediction.
    - Train and deploy machine learning models based on historical data.
    - Integrate ML outputs into existing analytics and reporting tools.
- Optimization and Fine-Tuning – Continue after implementation
    - Continuously monitor system performance and optimize resource allocation.
    - Fine-tune algorithms and models based on real-world usage and feedback.
    - Conduct periodic security audits and update governance protocols.
    - Implement cost-saving measures based on AWS Cost Explorer recommendations.

6.2 Cost Estimate

We used the Amazon Cost Estimator with some sensible settings to give us an idea of how much the whole project might cost. Keep in mind, these are just rough guesses. The real costs could be different based on things like how much we use the services, how much data we're moving around, and specific choices we make. Also, Amazon often changes how they charge for things, so for the most accurate estimates, it's a good idea to use the AWS Pricing Calculator.

Group hierarchy	Services	Monthly	First 12 months Total
	Amazon API Gateway		
	Amazon EC2		
	Amazon ElastiCache		
	Amazon AppFlow		
Data Ingestion	Amazon Kinesis Data Firehose	$112,193.45	$1,346,321.39
	AWS Lambda		
	Step Functions - Standard Workflows		
	Amazon Transcribe		
Data Processing and Orchestration	AWS Glue ETL jobs and interactive sessions	$2,560.08	$30,720.96
	Amazon Redshift		
	Amazon Athena		
	Amazon QuickSight Readers, Authors, SPICE		
	Amazon EMR master node on EC2		
	Amazon EMR core node on EC2		
Data Consumption Layer	Amazon OpenSearch Service	$112,610.57	$1,351,326.84
	Amazon EC2		
	Amazon EC2		
Auto Scaling Group	Amazon EC2	$6.57	$78.84
	Amazon Cognito		
	Application Load Balancer		
End User Collaboration	Amazon Route 53	$27,049.10	$324,589.20
	Amazon CloudWatch		
	AWS CloudTrail		
Monitoring and Optimization	AWS Cost Explorer	$1,692.08	$20,304.98
	AWS Config		
	AWS Lake Formation		
Data Security and Governance	AWS Key Management Service	$217.17	$2,606.04
	S3 Standard		
	Data Transfer		
	Amazon RDS Custom for SQL Server		
	Amazon RDS Custom for SQL Server		
Data Storage	Amazon RDS Custom for SQL Server	$3,319.05	$39,828.60
Metadata and Cataloging	AWS Glue ETL jobs and interactive sessions	$2.96	$35.52
Total		**$259,651.04**	**$3,115,812.48**

7 Summary

This initiative focuses on transforming a university's data infrastructure to address current challenges and prepare for future needs. The primary goals include improving data management, strengthening security measures, ensuring scalability, and incorporating machine learning for predictive analytics.

- **Checking the Situation**:
  We started with a comprehensive evaluation of the university's existing data landscape by closely looking at how the university deals with data today, understanding data types, and the data sources.

- **Fixing Things Up**:
  - **Hybrid Approach**: This solution intelligently integrates on-premises infrastructure with AWS services, ensuring a seamless transition by capitalizing on the reliability of existing systems. The Hybrid approach acts as a transitional bridge, acknowledging the importance of existing setups and the need for flexibility during the migration process.

  - **Cloud Solution**: Embracing AWS services fully, this solution prioritizes scalability and agility, promising efficiency in data processing. The Cloud Solution represents a decisive move towards AWS services, emphasizing the agility of cloud-based systems and the creation of a modern, future-ready data environment

- **Smart Moves for Making It Happen**:
  - **Data Lakehouse Model**: This modern data strategy forms the core of efficient data management, incorporating scalable data lakes, performance optimization, serverless architecture, unified data access, and machine learning integration.

  - **Optimization Techniques and Enhancements**: The report explores various strategies, including S3 Lifecycle Policies, Sharding for relational databases, Horizontal Scaling, Caching, CAP Theorem considerations, and Encryption, contributing to the establishment of a robust, efficient, and secure data infrastructure.

We have also outlined a phased project roadmap, detailing key milestones and a cost estimate using the Amazon Cost Estimator. It provides a structured plan for implementation, ensuring a methodical approach.

The proposed solution provides transformative vision for the university's data infrastructure—a harmonious blend of existing reliability and the dynamism of cloud-based solutions. This sets the stage for a modern, scalable, and secure foundation, primed for future advancements in data management and analytics.

8 Project Team

Meghna Suresh

Shilpa Nidhi Kirubanidhi

Shukla H Shanthakumara

Mallikarjuna Swamy

Manish Ramchandani

Vishruth Acharya

9 References

https://docs.aws.amazon.com/wellarchitected/latest/analytics-lens/batch-data-processing.html

https://aws.amazon.com/caching/database-caching/

https://aws.amazon.com/getting-started/decision-guides/analytics-on-aws-how-to-choose/

Transitioning objects using Amazon S3 Lifecycle - Amazon Simple Storage Service

Sharding with Amazon Relational Database Service | AWS Database Blog

Horizontal scaling - AWS Well-Architected Framework (amazon.com)

CAP theorem - Availability and Beyond: Understanding and Improving the Resilience of Distributed Systems on AWS (amazon.com)

Data encryption - Amazon Redshift

Encrypt Your Amazon Redshift Loads with Amazon S3 and AWS KMS | AWS Big Data Blog

AWS Glue DataBrew now provides detection and data masking transformations for Personally Identifiable Information (PII) (amazon.com)

Efficient Infrastructure Management with AWS CloudFormation | by Brandon Damue | AWS in Plain English

https://aws.amazon.com/blogs/architecture/overview-and-architecture-building-customer-data-platform-on-aws/

https://aws.amazon.com/solutions/case-studies/Universidad-Francisco-de-Vitoria-UFV-case-study/?did=cr_card&trk=cr_card

https://aws.amazon.com/blogs/big-data/category/case-study/

10 Appendix

Detailed cost estimate with configurations.

Detailed Estimate					
Group hierarchy	**Region**	**Service**	**Monthly**	**First 12 months total**	**Configuration summary**
Data Ingesion	US East (N. Virginia)	Amazon API Gateway	$ 105,441.62	$ 1,265,299.44	REST API request units (millions), Cache memory size (GB) (None), WebSocket message units (thousands), HTTP API requests units (millions), Average size of each request (512 KB), Average message size (32 KB), Requests (34 per month), Requests (25 per month), Messages (50 per second), Average connection duration (5 seconds), Average connection rate (2 per second)
	US East (N. Virginia)	Amazon EC2	$ 2.19	$ 26.28	Tenancy (Shared Instances), Operating system (Linux), Workload (Consistent, Number of instances: 1), Advance EC2 instance (t4g.nano), Pricing strategy (Compute Savings Plans 1 Year None upfront), Enable monitoring (disabled), DT Inbound: All other regions (27 TB per month), DT Outbound: Not selected (0 TB per month), DT Intra-Region: (0 TB per month)
	US East (N. Virginia)	Amazon ElastiCache	$ 916.97	$ 11,003.61	Instance type (cache.m5.xlarge), Cache Engine (Redis), Nodes (0), Utilization (On-Demand only) (100 %Utilized/Month), Cache Node Type (Standard), Pricing strategy (OnDemand), Nodes (0), Instance type (cache.r6gd.12xlarge), Utilization (On-Demand only) (100 %Utilized/Month), Cache Engine (Redis), Cache Node Type (Memory optimized), Pricing strategy (OnDemand), Average cache data size (in GB) (10), Average simple request rate (500 per second)

	US East (N. Virginia)	Amazon AppFlow	$ 304.31	$ 3,651.74	Number of flows (5 per day), Volume of data per flow (100 GB)
	US East (N. Virginia)	Amazon Kinesis Data Firehose	$ 5,528.36	$ 66,340.32	Dynamic Partitioning (Add On) (Disabled), Source Type (Direct PUT or Kinesis Data Stream), Average ratio of data processed to VPC vs data ingested (1.5), Data records units (thousands), Record size (5 KB), Data format conversion (optional) (Disabled), Number of records for data ingestion (10 per second), Number of subnets for VPC delivery (2)
Data Processing and Orchestration	US East (N. Virginia)	AWS Lambda	$ 36.72	$ 440.64	Architecture (x86), Architecture (x86), Invoke Mode (Buffered), Amount of ephemeral storage allocated (512 MB), Number of requests (1000000 per month), Concurrency (100), Time for which Provisioned Concurrency is enabled (24 hours), Number of requests for Provisioned Concurrency (50000 per month), Number of requests (500000 per month)
	US East (N. Virginia)	Step Functions - Standard Workflows	$ 12.40	$ 148.80	Workflow requests (100000 per month), State transitions per workflow (5)
	US East (N. Virginia)	Amazon Transcribe	$ 2,508.00	$ 30,096.00	Total number of minutes processed in batch by standard Amazon Transcribe (100000 per month), How many minutes out of the total batch minutes will use Content/PII Redaction? (20000 per month), How many minutes out of the total batch minutes will be processed with Custom Language Models? (10000 per month)
	US East (N. Virginia)	AWS Glue ETL jobs and interactive sessions	$ 2.96	$ 35.52	Number of DPUs for Apache Spark job (10), Number of DPUs for Python Shell job (0.0625)
Data Consumption Layer	US East (N. Virginia)	Amazon Redshift	$ 4,017.55	$ 48,210.60	Nodes (1), Instance type (dc2.8xlarge), Utilization (On-Demand only) (100 %Utilized/Month), Pricing strategy (OnDemand), Additional backup storage (50 GB), Data scanned (100 TB), Managed storage size (500 GB), Data Transfer In To (20 GB)

	US East (N. Virginia)	Amazon Athena	$ 101,183.02	$ 1,214,196.24	Total number of queries (10000 per day), Amount of data scanned per query (50 GB), Number of DPUs (100), Length of time (hours) capacity is active (720 hours per month), Total number of spark sessions (50 per day)
	US East (N. Virginia)	Amazon QuickSight Readers, Authors, SPICE	$ 1,690.00	$ 20,280.00	Number of working days per month (22), SPICE capacity in gigabytes (GB) (10), Number of authors (5), Number of readers (1000)
	US East (N. Virginia)	Amazon EMR master node on EC2	$ 21.90	$ 262.80	Number of master EMR nodes (1), EC2 instance (c1.medium), Utilization (100 %Utilized/Month)
	US East (N. Virginia)	Amazon EMR core node on EC2	$ 43.80	$ 525.60	Number of core EMR nodes (2), EC2 instance (c1.medium), Utilization (100 %Utilized/Month)
	US East (N. Virginia)	Amazon OpenSearch Service	$ 5,654.30	$ 67,851.60	Number of instances (1), Storage for each Amazon OpenSearch Service instance (General Purpose SSD (gp3)), UltraWarm storage cost (491.52), Nodes (1), Instance type (r5.2xlarge.search), Utilization (On-Demand only) (100 %Utilized/Month), Instance Node Type (Memory optimized), Storage Type (EBS Only), Pricing strategy (OnDemand), Nodes (1), Instance type (r5.2xlarge.search), Utilization (On-Demand only) (100 %Utilized/Month), Instance Node Type (Memory optimized), Storage Type (EBS Only), Pricing strategy (OnDemand), Number of nodes (2), Instance type (ultrawarm1.large.search), Utilization (On-Demand only) (100 %Utilized/Month), Pricing strategy (OnDemand), Storage amount per volume (gp3) (1000 GB), Provisioning IOPS per volume (gp3) (8000), Provisioning throughput (MB/s) per volume (gp3) (300 MBps), Managed storage amount (per UltraWarm instance) (10 TB)

Auto Scaling Group	US East (N. Virginia)	Amazon EC2	$ 2.19	$ 26.28	Tenancy (Shared Instances), Operating system (Linux), Workload (Consistent, Number of instances: 1), Advance EC2 instance (t4g.nano), Pricing strategy (Compute Savings Plans 1 Year None upfront), Enable monitoring (disabled), DT Inbound: All other regions (27 TB per month), DT Outbound: Not selected (0 TB per month), DT Intra-Region: (0 TB per month)
	US East (N. Virginia)	Amazon EC2	$ 2.19	$ 26.28	Tenancy (Shared Instances), Operating system (Linux), Workload (Consistent, Number of instances: 1), Advance EC2 instance (t4g.nano), Pricing strategy (Compute Savings Plans 1 Year None upfront), Enable monitoring (disabled), DT Inbound: All other regions (27 TB per month), DT Outbound: Not selected (0 TB per month), DT Intra-Region: (0 TB per month)
	US East (N. Virginia)	Amazon EC2	$ 2.19	$ 26.28	Tenancy (Shared Instances), Operating system (Linux), Workload (Consistent, Number of instances: 1), Advance EC2 instance (t4g.nano), Pricing strategy (Compute Savings Plans 1 Year None upfront), Enable monitoring (disabled), DT Inbound: All other regions (27 TB per month), DT Outbound: Not selected (0 TB per month), DT Intra-Region: (0 TB per month)
End User Collaboation	US East (N. Virginia)	Amazon Cognito	$ 3,039.25	$ 36,471.00	Advanced security features (Enabled), Number of monthly active users (MAU) (60000)
	US East (N. Virginia)	Application Load Balancer	$ 23,392.85	$ 280,714.20	Number of Application Load Balancers (2)
	US East (N. Virginia)	Amazon Route 53	$ 617.00	$ 7,404.00	Hosted Zones (2), Additional Records in Hosted Zones (15000), Basic Checks Within AWS (5), Number of Elastic Network Interfaces (5), Number of domains stored (2)

Monitoring and Optimization	US East (N. Virginia)	Amazon CloudWatch	$ 666.58	$ 7,998.98	Number of Metrics (includes detailed and custom metrics) (100), GetMetricData: Number of metrics requested (500000), GetMetricWidgetImage: Number of metrics requested (250000), Number of other API requests (1000000), Standard Logs: Data Ingested (500 GB), Logs Delivered to CloudWatch Logs: Data Ingested (250 GB), Logs Delivered to S3: Data Ingested (500 GB), Expected Logs Data scanned (100 GB), Number of Custom/Cross-account events (1000), Number of Dashboards (5), Number of Standard Resolution Alarm Metrics (10), Number of High Resolution Alarm Metrics (5), Number of composite alarms (2), Number of alarms defined with a Metrics Insights query (3), Average number of metrics scanned by each Metrics Insights query (50), Number of Canary runs (100), Number of Contributor Insights rules for CloudWatch (2), Total number of matched log events for CloudWatch (5 million matched log events per month), Number of Contributor Insights rules for DynamoDB (3), Total number of events for DynamoDB (3 million events per month), Number of Lambda functions (20), Number of requests per function (100 per hour), Monthly visitors to your web application (50000), Number of RUM events per visit (5), Percentage of RUM event (0.1), Number of monitored resources (5 per hour), Number of city-networks to be monitored (3 per hour)
	US East (N. Virginia)	AWS CloudTrail	$ 20.50	$ 246.00	Management events units (millions), Write management trails (1), Read management trails (1), Data events units (millions), S3 trails (1), Lambda trails (1), Insight events units (millions), Trails with Insight events (1), Write management events (20 per month), Read management events (10 per month), S3 operations (2 per month), Lambda data events (1 per month), Number of write management events analyzed (5 per month)

	US East (N. Virginia)	AWS Cost Explorer	$ 1,005.00	$ 12,060.00	Number of UsageRecords per month (500000), Number of API requests per month (100000)
Data Security and Governance	US East (N. Virginia)	AWS Config	$ 102.50	$ 1,230.00	Number of Configuration items recorded (500), Number of Config rule evaluations (100000), Number of Conformance pack evaluations (1000)
	US East (N. Virginia)	AWS Lake Formation	$ 65.82	$ 789.84	Data scanned (10 TB), Storage usage in a month (in millions) (5), Requests in a month (in millions) (3), Number of tables (10), Number of small files ingested per table per day (100), Size of small files (less than 64MB) (20 MB)
	US East (N. Virginia)	AWS Key Management Service	$ 48.85	$ 586.20	Number of customer managed Customer Master Keys (CMK) (5), Number of symmetric requests (2000000), Number of asymmetric requests except RSA 2048 (100000), Number of asymmetric requests involving RSA 2048 (50000), Number of ECC GenerateDataKeyPair requests (20000), Number of RSA GenerateDataKeyPair requests (30000)
Data Storage	US East (N. Virginia)	S3 Standard	$ 60.33	$ 723.96	S3 Standard storage (500 GB per month), PUT, COPY, POST, LIST requests to S3 Standard (100000), GET, SELECT, and all other requests from S3 Standard (500000), Data returned by S3 Select (10 TB per month), Data scanned by S3 Select (20 TB per month)
	US East (N. Virginia)	Data Transfer	$ -	$ -	
	US East (N. Virginia)	Amazon RDS Custom for SQL Server	$ 1,086.24	$ 13,034.88	Storage for each RDS Custom for SQL Server instance (General Purpose SSD (gp2)), Storage amount (100 GB), Instance type (db.r5.xlarge), Number of RDS Custom for SQL Server instances (1), Utilization (On-Demand only) (100 %Utilized/Month), Database edition (Developer), Deployment option (Multi-AZ), License (Customer-provided), Pricing strategy (OnDemand)

	US East (N. Virginia)	Amazon RDS Custom for SQL Server	$ 1,086.24	$ 13,034.88	Storage for each RDS Custom for SQL Server instance (General Purpose SSD (gp2)), Storage amount (100 GB), Instance type (db.r5.xlarge), Number of RDS Custom for SQL Server instances (1), Utilization (On-Demand only) (100 %Utilized/Month), Database edition (Developer), Deployment option (Multi-AZ), License (Customer-provided), Pricing strategy (OnDemand)
	US East (N. Virginia)	Amazon RDS Custom for SQL Server	$ 1,086.24	$ 13,034.88	Storage for each RDS Custom for SQL Server instance (General Purpose SSD (gp2)), Storage amount (100 GB), Instance type (db.r5.xlarge), Number of RDS Custom for SQL Server instances (1), Utilization (On-Demand only) (100 %Utilized/Month), Database edition (Developer), Deployment option (Multi-AZ), License (Customer-provided), Pricing strategy (OnDemand)
Metadata and Cateloging	US East (N. Virginia)	AWS Glue ETL jobs and interactive sessions	$ 2.96	$ 35.52	Number of DPUs for Apache Spark job (10), Number of DPUs for Python Shell job (0.0625)