Airline Ticket Booking Platform Cloud Architecture Design

Data Delivery - Portfolio

Student: Praiselin Lydia Gladston

Student ID: 000965570

GitHub Profile – https://www.github.com/praiselydia
LinkedIn Profile – https://www.linkedin.com/in/praiselydia

Contents

GitHub Profile – https://www.github.com/praiselydia
LinkedIn Profile – https://www.linkedin.com/in/praiselydia

Mission

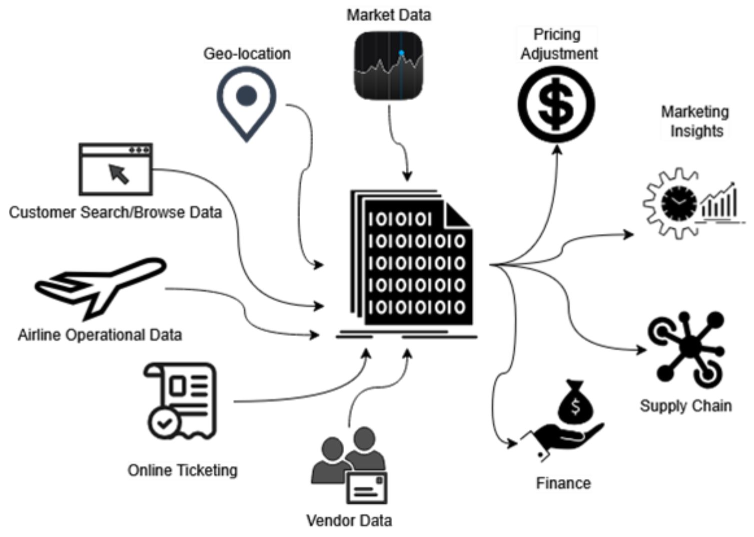
Design and implement a scalable, resilient, and secure cloud infrastructure that supports airline operations, enhances internal workflows, and ensures high performance and reliability.

Objectives

- **Scalable Infrastructure** – Design an infrastructure capable of handling fluctuations in system usage, ensuring continuous service availability during peak loads.
- **Data Consistency** – Ensure the consistency and integrity of data across internal systems, supporting mission-critical operations like booking, scheduling, and inventory management.
- **Centralized Repository** – Design a centralized data repository to consolidate operational data, enabling efficient reporting and real-time analytics.
- **System Monitoring** – Integrate monitoring, logging, and alerting tools to track system health and performance, ensuring rapid issue detection and resolution.

Data Source and Sink

Identifying data sources and sinks during the cloud architecture design phase is crucial for ensuring a seamless, efficient, and purpose-driven data flow across the system. Clearly defining sources helps architects understand the origin, nature, and velocity of incoming data, which is essential for choosing the right ingestion, storage, and processing mechanisms. Specifying data sinks ensures that the architecture is tailored to deliver actionable insights, meet business goals, and support decision-making. This alignment not only optimizes resource utilization but also ensures scalability, data consistency, and system reliability from the outset.



GitHub Profile – https://www.github.com/praiselydia
LinkedIn Profile – https://www.linkedin.com/in/praiselydia

Data Sources

1. **Online Ticketing Data**
   Nature: Structured, Real-time
   Format: CSV, JSON
   Usage: Captures real-time booking transactions from direct sales channels, providing insights for demand forecasting, pricing strategies, and customer segmentation analysis.

2. **Vendor Data (from Online Travel Agency)**
   Nature: Structured, Batched, and Real-time
   Format: CSV, JSON
   Usage: Aggregates booking data from external vendors, enhancing demand forecasting, evaluating partnerships, and assessing market reach.

3. **Customer Search and Browsing Data**
   Nature: Unstructured, Batched
   Format: XML, TXT, Event Logs
   Usage: Analyzes user activity on digital platforms to model customer intent, refine demand forecasts, and support dynamic pricing initiatives.

4. **Geo-location Data**
   Nature: Semi-structured, Real-time, or Streaming
   Format: JSON
   Usage: Offers real-time customer location insights, enabling regional demand analysis, personalized promotions, and route optimization.

5. **Airline Operational Database**
   Nature: Structured, Batched
   Format: JSON, CSV
   Usage: Includes operational details like schedules and inventory, crucial for capacity planning and aligning demand with supply.

6. **Historical Sales and Booking Records**
   Nature: Structured, Batched
   Format: CSV, JSON
   Usage: Archives past booking trends for training forecasting models, seasonal analysis, and performance evaluation.

7. **Competitor Pricing and Market Data**
   Nature: Semi-Structured, Batched, or Real-time
   Format: CSV, JSON
   Usage: Gathers competitor and market insights to refine pricing strategies, model demand, and optimize revenue.

These sources can be majorly split into two for their ingestion pipelines that will be explained below:
Batched: Operational database, vendor data, currency conversion data and geo-location.
Streaming: Airline Ticket booking data

Data Sinks

1. **Operational Efficiency Reports (Supply Chain)**
   Purpose: To enhance airline operations by optimizing resource utilization and minimizing inefficiencies.
   Details: Insights into flight schedules, seat occupancy, crew performance, and resource allocation highlight areas for improvement, such as overbooked flights and delays. Key metrics include turnaround times, on-time performance, and staff productivity.

2. **Customer Behavior Insights (Marketing Insights)**
   Purpose: To analyze customer preferences and behavior for targeted marketing and pricing strategies.
   Details: Derived from user activity, booking history, and demographics, these insights reveal customer segments, route demand trends, and decision drivers, supporting personalized offers and revenue growth.

3. **Demand Forecasting Reports (Supply Chain, Pricing Adjustment)**
   Purpose: To anticipate future demand, enabling efficient capacity planning, pricing, and resource allocation.
   Details: Combines historical and real-time data with external factors (weather, holidays/events) to predict demand by route, time, and segment, informing inventory and fleet management.

4. **Pricing Optimization Analysis (Finance, Pricing Adjustment)**
   Purpose: To refine pricing strategies using competitive and market analyses.
   Details: Real-time competitor trends and customer behavior data drive dynamic pricing models, ensuring competitive fares that maximize revenue and customer appeal.

5. **Route Optimization Insights (Supply Chain)**
   Purpose: To optimize flight routes for efficiency and profitability.
   Details: Analyze booking and competitor data to recommend route adjustments, identify high-demand opportunities, and align resources with market needs.

6. **Competitor Performance and Market Positioning Reports (Finance, Marketing Insights)**
   Purpose: To benchmark the airline's performance and refine its competitive strategy.
   Details: Evaluates competitor pricing, market share, and promotional activities to identify differentiation opportunities and enhance market positioning.

GitHub Profile – https://www.github.com/praiselydia
LinkedIn Profile – https://www.linkedin.com/in/praiselydia

Cloud Architecture

Cloud architecture has evolved significantly over time, transforming from simple infrastructure hosting to sophisticated, intelligent ecosystems. With the rise of microservices and containerization, cloud architecture shifted toward more modular, scalable, and resilient systems—supported by orchestration tools like Kubernetes. The emergence of serverless computing allowed functions to run on demand, further abstracting infrastructure management.

Today's cloud architecture embraces event-driven design, data lakehouses, AI/ML integration, and multi-cloud/hybrid environments, supporting real-time analytics, automation, and global scalability. This evolution has enabled businesses to build more agile, intelligent, and cost-efficient systems that respond rapidly to changing market needs.

We will design our cloud architecture with a data lakehouse for this project. To do so, we will dive into what really a lakehouse is:

Lakehouse Architecture

The lakehouse architecture was introduced to address the limitations of traditional data lakes and warehouses. While data lakes offered flexibility and low-cost storage for diverse data types, they lacked robust data management, governance, and performance for analytics. On the other hand, data warehouses were optimized for structured data and analytics but were expensive, rigid, and not ideal for handling unstructured or streaming data.

The lakehouse combines the strengths of both—allowing businesses to store all data in one platform while supporting high-performance analytics, governance, and machine learning. It consists of 3 layers where data is managed in different formats.

Bronze Layer – Raw Data Storage

- **Data State**: Unprocessed, raw data in original format.
- **Purpose**: Acts as a historical archive and debug source.
- **Tools**: Azure Data Lake Gen2, Azure Event Hub.

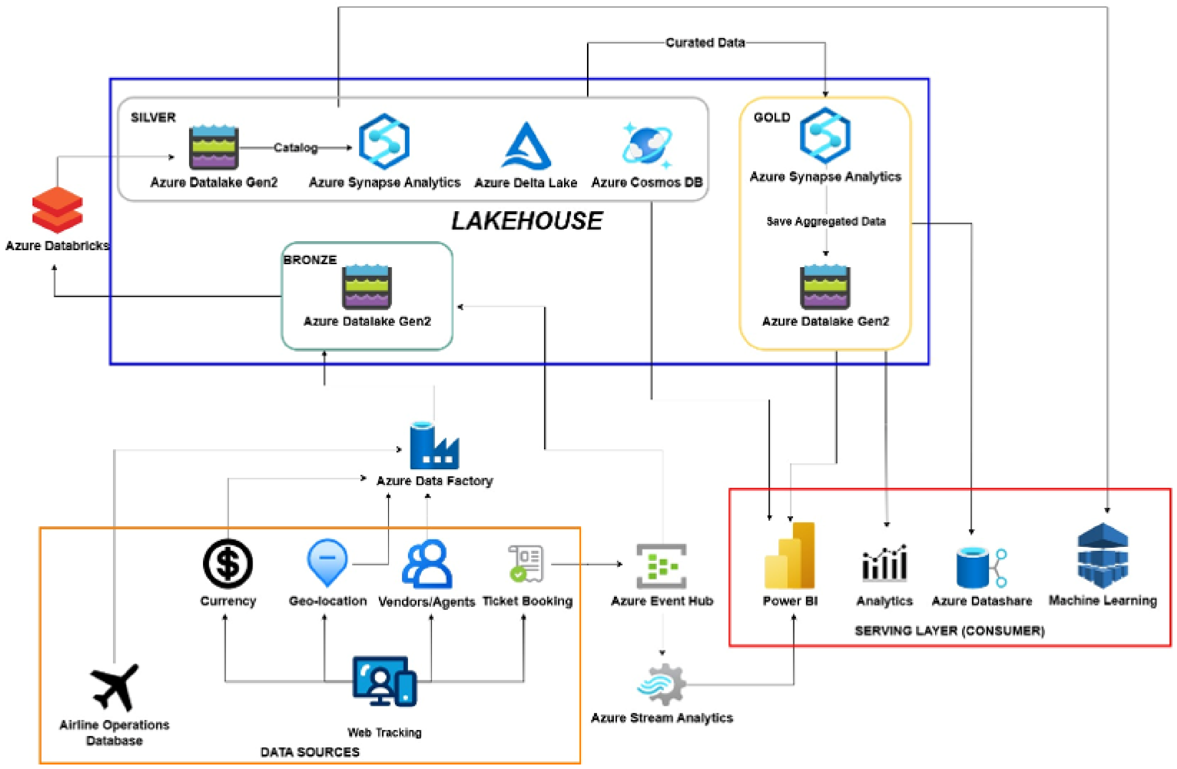
Silver Layer – Cleaned and Enriched Data

- **Data State**: Transformed, semi-curated, analytics ready.
- **Purpose**: Clean joins, remove duplicates, apply standard formats.
- **Tools**: Azure Synapse Analytics, Azure Delta Lake, Azure Cosmos DB.

Gold Layer – Aggregated Business Data

- **Data State**: Fully curated, business contextualized.

- **Purpose**: Drives dashboards, reports, and machine learning.
- **Tools**: Azure Synapse Analytics, Power BI, Azure Machine Learning.



Azure Services Used

1. Azure Data Lake Storage Gen2

Azure Data Lake Storage Gen2 is the foundational storage layer for the lakehouse architecture. It is primarily used at the bronze layer, where all raw data is ingested and stored. Its scalable and cost-effective nature makes it ideal for storing massive volumes of source data in various formats like CSV, JSON, and Parquet. Being optimized for big data analytics, it supports hierarchical namespace and fine-grained access control, making it suitable for enterprise-grade workloads.

2. Azure Synapse Analytics

Azure Synapse Analytics is used in the silver and gold layers to query, transform, and serve curated and aggregated data. In the Silver layer, Synapse enables powerful SQL and Spark-based transformations on cleaned datasets, making them analytics-ready. In the Gold layer, it helps produce business-facing reports and dashboards by summarizing data for insights such as customer trends, route optimization, and pricing strategies. Synapse's integration with Data Lake Storage and its ability to run serverless and dedicated workloads makes it an essential tool for scalable, enterprise-level analytics.

3. Azure Databricks

Azure Databricks is used for transforming data from the Bronze to the Silver layer. It provides an Apache Spark-based analytics platform that supports large-scale data engineering and machine learning workloads. In this architecture, Databricks processes raw ingested data, performing tasks like data cleaning, validation, schema enforcement, and enrichment before storing the results in Delta Lake. Its notebook interface and collaborative environment make it particularly useful for teams working on complex ETL pipelines and data science models.

4. Azure Delta Lake

Azure Delta Lake is integrated within the silver layer to bring ACID transaction support, version control, and schema enforcement to the data lake. It enables reliable, consistent, and query-optimized storage for curated datasets. Delta Lake ensures that data in the silver layer remains clean, auditable, and resilient to failures, supporting operations like upserts and time-travel queries. It plays a crucial role in maintaining data quality before data is pushed to the gold layer for consumption.

5. Azure Cosmos DB

Azure Cosmos DB is a real-time NoSQL database used in the silver layer to support low-latency access to curated data for operational applications. It is ideal for use cases such as personalized booking suggestions, customer behavior tracking, and mobile app experiences. With multi-region replication, flexible data models, and guaranteed availability, Cosmos DB ensures that real-time data access needs are met without compromising on performance or consistency, especially for global airline operations.

Serving Layer (Consumers)

Once data is curated in the gold layer, it's made available to various tools and platforms in the serving layer:

- **Power BI**: Interactive dashboards for operational and strategic reporting.
- **Analytics**: Custom reporting, KPI analysis, and ad-hoc queries.
- **Azure Data Share**: Data collaboration and sharing across business units.
- **Machine Learning**: Demand prediction, dynamic pricing, and customer behavior modeling.

Pipeline Design

Master Pipeline

The Master Pipeline is implemented using Azure Data Factory and acts as the central orchestrator for all other pipelines. It manages the sequencing, dependencies, and scheduling of batch, streaming, curation, and aggregation processes. This ensures data flows smoothly from ingestion to consumption, maintains consistency across layers, and enables robust error handling and monitoring across the entire data ecosystem.

1. Batch Ingestion Pipeline

All batched data sources—such as historical records, operational databases, geo-location, user behavior and vendor data—are first processed through Azure Data Factory. This pipeline handles the scheduled extraction, transformation (if required), and loading (ETL) of data into the bronze layer of the data lake in the lakehouse architecture. It ensures data is ingested in a consistent format (e.g., CSV, JSON) and made available for further processing.

2. Streaming Ingestion Pipeline

This pipeline handles real-time data sources, such as live ticket bookings from user's personal devices. Streaming data bypasses batch processing tools and is directly ingested into the bronze layer of the lakehouse. Technologies like Azure Event Hubs and Azure Stream Analytics are used here to manage high-velocity data flows with low latency, enabling real-time insights and rapid response capabilities.

3. Curation Pipeline

Once data is ingested into the lakehouse, the curation pipeline processes it within the platform. This stage involves cleaning, validating, and enriching the data in the silver layer. It includes tasks like removing duplicates, standardizing formats, and joining datasets. The result is structured, high-quality data that is analytics-ready and can be reused across multiple use cases.

4. Aggregation Pipeline

This pipeline takes curated data from the silver layer and performs business logic, summarization, and aggregation to generate final outputs. These are written into the gold layer of the lakehouse—storage optimized for consumption by BI tools, dashboards, and ML models. It supports the serving layer, delivering business-ready insights for marketing, operations, and finance.

Pipeline Failure Strategies

To maintain data integrity and ensure continuous operation of the system, several failure-handling mechanisms are built into the architecture:

1. Time-Out

A timeout represents the maximum duration an activity is allowed to run. In Azure Data Factory, this is set to 7 days by default. Once the timeout expires, the activity is terminated—regardless of whether it was close to completion or not.

It's essential to configure a realistic timeout based on typical runtime patterns observed during development and testing.

For instance, if a notebook activity typically finishes in 5 minutes, a safe timeout might be set at 15 to 30 minutes to provide a buffer.

Setting appropriate timeouts prevents pipelines from hanging indefinitely and helps with quicker fault detection and resolution.

2. Backoff Interval

The backoff interval determines the delay between retry attempts after a failure. Azure Data Factory uses a default backoff interval of 30 seconds.

This helps prevent the system from immediately retrying a failing activity, which could strain external services or repeat the same error.

Backoff intervals are particularly useful in cases of temporary network issues or resource contention.

3. Max Retries

This refers to the number of times an activity will be retried before the system gives up and marks it as failed.

The default retry count in Azure Data Factory is 0—meaning the activity is only attempted once.

For better resilience, a common practice is to increase the retry count to 3, especially when dealing with transient errors like network interruptions or service unavailability.

This setting works together with the backoff interval to provide controlled and efficient retry behavior.

4. Activity/Pipeline Re-run

When an activity fails, there are two main strategies for rerunning pipelines:

a. Full Pipeline Re-run

The entire pipeline is rerun from the start.

This approach is straightforward but comes with the risk of data duplication, especially in the bronze layer where raw data is ingested.

It's useful when data dependencies are complex and require the full process to be replayed for consistency.

b. Rerun from Failed Activity

Only the failed activity is rerun, continuing from where the pipeline left off.

This reduces processing time and avoids duplicates.

However, it can lead to inconsistent data ingestion—since data captured after failure may differ from the initial state, especially in real-time pipelines.

Conclusion

Cloud technology has evolved rapidly, giving us powerful tools to build scalable, resilient, and intelligent systems with ease. From real-time processing to global availability, the cloud continues to simplify complex problems and accelerate innovation. But the true strength of any system lies in its design. Thoughtful architecture, especially around data pipelines and failure strategies, ensures long-term reliability and adaptability. As the cloud keeps evolving, focusing on strong, forward-thinking design from the start is what truly sets successful systems apart.