# Explain AWS Detail

**Business Use Case and ML Objective**

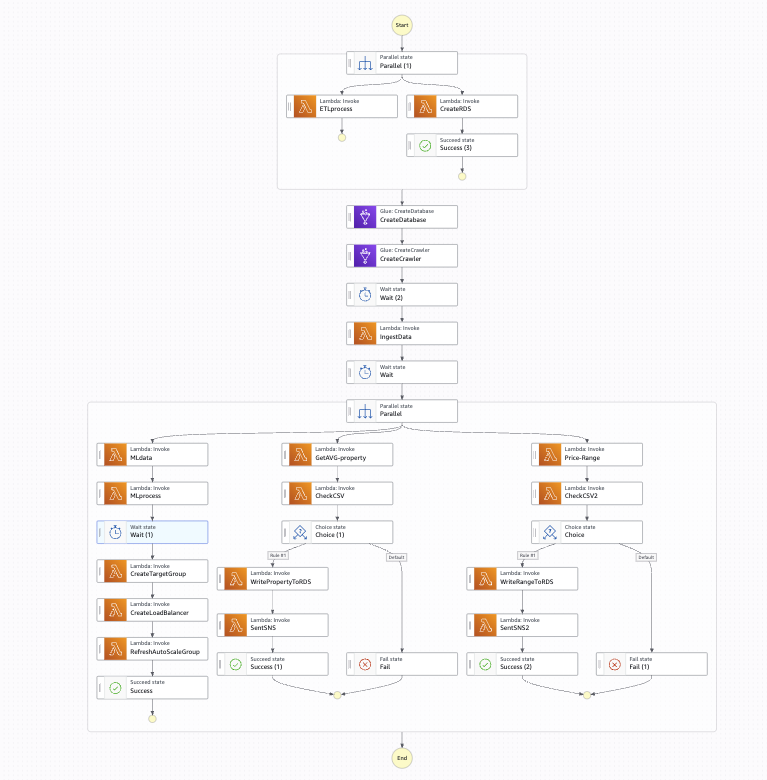
The goal of this pipeline is to empower new Airbnb property owners with data-driven pricing strategies and actionable market insights by combining machine learning predictions with analytical review and pricing analysis. The core machine learning objective is to train a K-Nearest Neighbor (KNN) model that predicts the optimal nightly rental price based on property attributes such as location, room type, property type, and 16 other features. This model is deployed to a web interface where clients can input their listing details and receive instant pricing recommendations aligned with market conditions.

Beyond prediction, the pipeline provides business intelligence through two parallel analytics flows. The first analyzes customer satisfaction metrics—accuracy, communication, location, and value—across different property types. This helps uncover top-performing listing types (e.g., treehouses and villas) and identify service gaps in underperforming ones (e.g., dorms, campers). These insights guide marketing, quality control, and operational decisions.

The second analytical stream focuses on price tier segmentation. By grouping listings into low, mid-range, and upper-mid-range price bands, it identifies which segments deliver the highest perceived value and satisfaction. Notably, upper-mid-range listings ($150–$299) show the highest guest ratings but represent the smallest market share, signaling an underutilized premium niche. Meanwhile, the mid-range tier dominates in volume but delivers only average value, indicating potential for service enhancement or price restructuring.

Together, these three flows—pricing prediction, property-type quality insights, and price-tier strategy—form a holistic system that supports pricing optimization, guest experience improvement, and data-backed business growth for Airbnb hosts and platforms.

**Data Pipeline Architecture Diagram**



The architecture follows a modular, serverless, and scalable design orchestrated by AWS Step Functions. The workflow initiates with data ingestion and ETL, then branches into three parallel processing flows: ML model training and deployment, review score analytics, and price tier analysis.

The data ingestion phase starts with uploading raw Airbnb CSV files (e.g., airbnb\_ratings\_new.csv, ml\_data.csv) into Amazon S3. A Lambda function triggers an AWS Glue ETL job, which cleans and transforms the data, while a Glue Crawler detects and registers the schema in the Glue Data Catalog. The cleaned output is written to an Amazon RDS MySQL database for structured storage and downstream querying.

From there, the pipeline branches into three parallel paths:

* ML Path (Flow 1):  
  1. MLData Lambda formats the dataset.
  2. MLProcess Lambda launches a SageMaker Processing Job to train a KNN model (RMSE ≈ 97.68).
  3. Once trained, the model is deployed to an EC2 Auto Scaling Group.
  4. RefreshAutoScaleGroup Lambda ensures that updated .pkl models are served behind an Application Load Balancer (ALB) with health checks.
  5. End users access the model through a web frontend that returns predicted prices based on form inputs.
* Review Analysis Path (Flow 2):  
  1. GetAVG-property Lambda extracts review scores for key dimensions (accuracy, communication, location, value).
  2. Results are written into RDS by WritePropertyToRDS.
  3. CheckCSV validates data; SentSNS notifies of process completion or errors.
  4. These insights support recommendations about high-performing or underperforming property types.
* Price Tier Analysis Path (Flow 3):  
  1. Price-Range Lambda segments data into pricing tiers (e.g., $0–49, $50–149, $150–299).
  2. It evaluates average value and accuracy scores per tier and writes to RDS.
  3. This uncovers high-value, underserved segments like upper mid-range listings.

Each step is monitored through conditional logic in the Step Function (e.g., Choice states), and AWS SNS sends notifications on success or failure. The system supports retraining, re-deployment, and real-time querying in an automated and scalable fashion.

# AWS Services Used

Amazon S3 is utilized for storing raw CSV files such as airbnb\_ratings\_new.csv, processed datasets, and Python scripts used for machine learning. Its scalable, durable, and secure storage enables efficient data handling and seamless integration with upstream and downstream AWS services.

AWS Glue, including Glue ETL jobs and Glue Crawlers, automates the extraction, transformation, and loading (ETL) process. Crawlers automatically scan and infer schema from data stored in S3 and register it in the Glue Data Catalog. Glue ETL jobs clean, normalize, and prepare the data, enabling reliable and repeatable ingestion into structured databases.

AWS Lambda provides a serverless execution environment to automate various data processing steps. It handles event-driven tasks such as formatting ML datasets (MLdata Lambda), triggering SageMaker processing jobs (MLprocess Lambda), and sending validation results or alerts via Amazon SNS (SentSNS Lambda). Lambda functions also orchestrate infrastructure updates, such as refreshing the Auto Scaling Group with a newly trained model.

Amazon SageMaker serves as the training and inference environment for the K-Nearest Neighbors (KNN) model. It is responsible for computing optimal daily rental prices based on property features. SageMaker’s integration with Lambda and S3 enables efficient training, model evaluation (RMSE ≈ 97.68), and deployment in a production-ready manner.

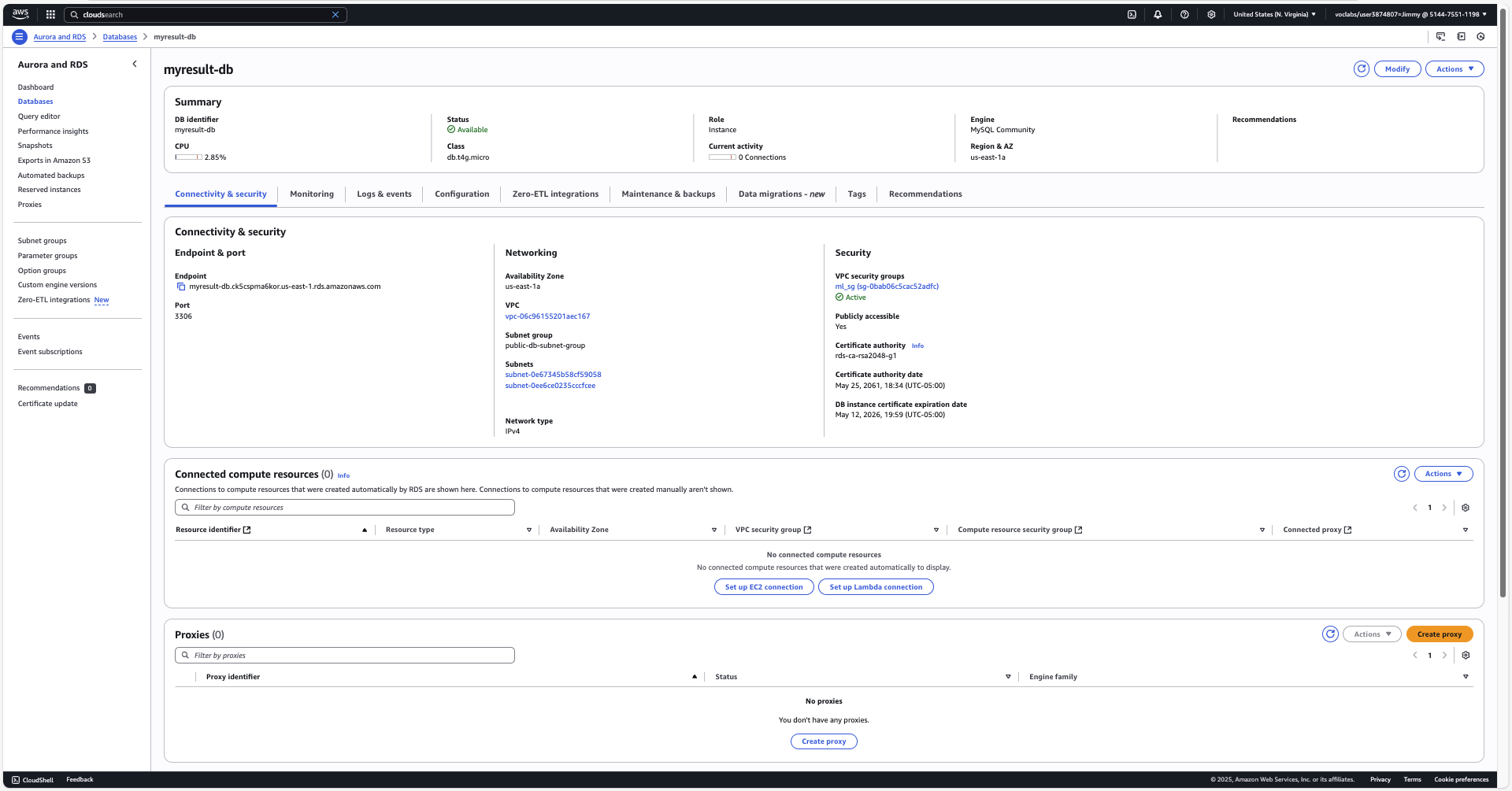
Amazon RDS (Relational Database Service) with MySQL engine is used to store cleaned datasets and the results of analytical queries. It enables long-term, structured storage of property insights, price tier analysis, and review metrics, making them queryable via SQL. RDS acts as the central analytical database, supporting downstream reporting and dashboard integration.

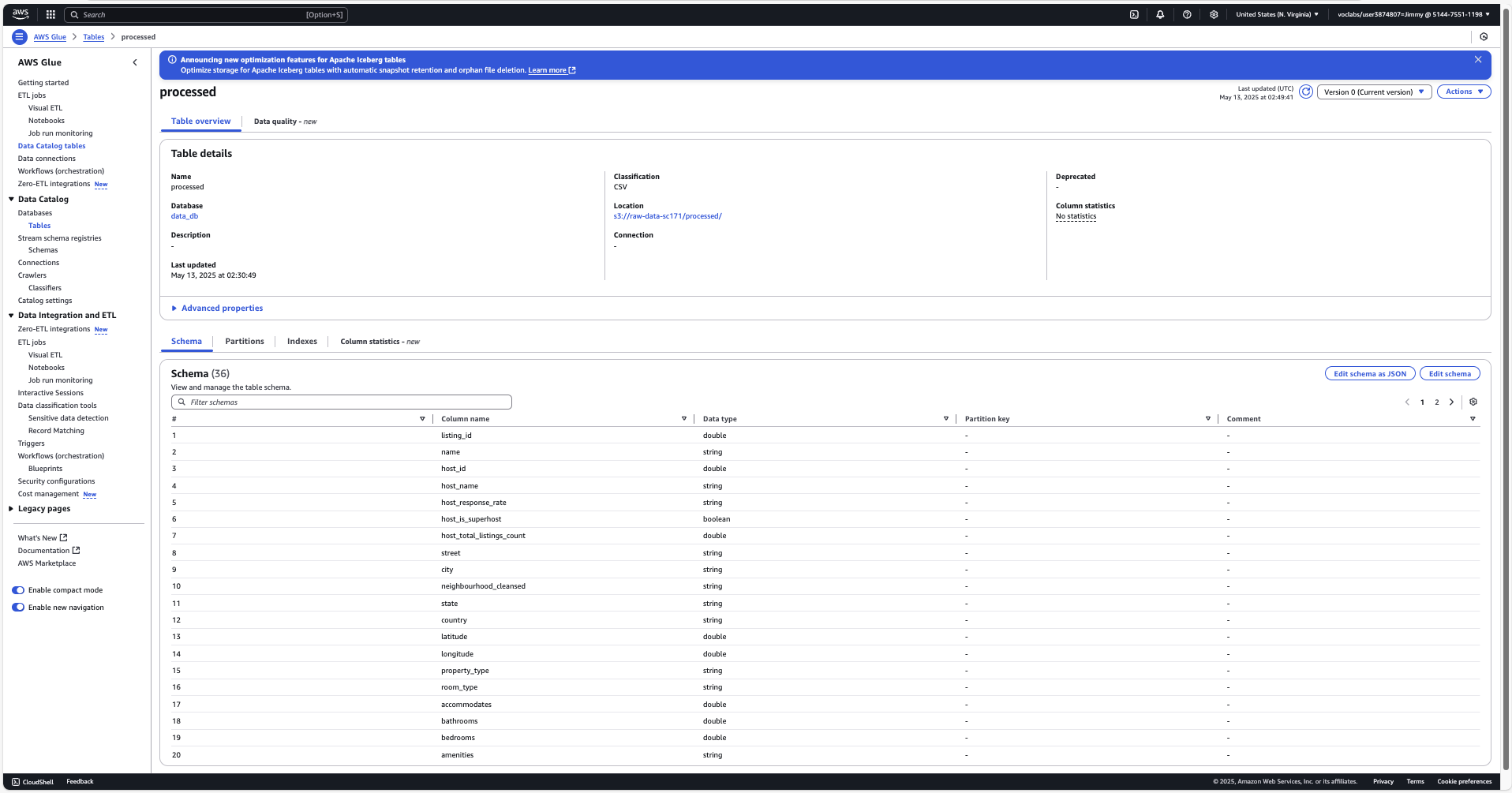
Auto Scaling Group (ASG) ensures elasticity and resilience of the web application hosting the ML inference API. ASG automatically provisions or terminates EC2 instances based on traffic demand. After each new model deployment, the group is refreshed to load the updated .pkl model, ensuring real-time prediction services are always current and performant.

Amazon Application Load Balancer (ALB) provides public-facing access to the EC2-hosted model API, routing incoming requests to healthy instances based on configured health checks. It ensures high availability and failover protection for end-user interactions.

Amazon Simple Notification Service (SNS) enables real-time notifications for pipeline events such as validation success, failure, or processing completion. This ensures proactive monitoring and rapid incident response, improving system transparency and maintainability.

# ETL Design and Schema Decisions





The ETL pipeline primarily uses Python scripts to handle initial data cleaning, formatting, and transformation processes. These scripts streamline data into a consistent structure suitable for subsequent analysis and model training.  
  
AWS Glue ETL Jobs further automate and standardize these processes, ensuring consistent and scalable data transformation and loading. Glue Crawlers dynamically generate the database schema from processed data, enabling easy integration and querying capabilities across AWS services. The schema specifically focuses on attributes such as property types, price, accuracy, and customer satisfaction metrics, which are crucial for efficient queries and analytics.  
  
This approach reduces manual interventions, improves accuracy, and enhances overall pipeline efficiency, ensuring data quality and timeliness in the analytical and machine learning processes.

# Data Ingestion Details (Batch)

The project predominantly employs batch ingestion methods, where the CSV data files (airbnb\_ratings\_new.csv) undergo periodic processing. The processed data is subsequently stored and updated in batches, making it readily accessible for analytical tasks and machine learning applications.  
  
Despite the batch-oriented ingestion, certain critical workflow steps leverage near real-time processing capabilities, particularly through AWS Lambda functions. These functions quickly trigger processes such as SageMaker model training or data validation and notifications, ensuring timely responsiveness and integration within the ETL pipeline.

# Security Considerations (IAM Policies)

Security is carefully enforced across the pipeline through a combination of IAM policies, VPC configurations, encryption standards, and controlled network access. The principle of least privilege is strictly followed when assigning permissions to each AWS service and Lambda function.

IAM roles are tightly scoped for each Lambda function to only allow necessary actions. For instance, the MLdata and MLprocess Lambda functions are granted limited s3:GetObject, s3:PutObject, and sagemaker:CreateProcessingJob permissions, ensuring they can only read model scripts and trigger training jobs without overexposure to other resources. Similarly, the GetAVG-property and Price-Range Lambda functions have minimal rds:Connect and rds:ExecuteStatement privileges, allowing them to write insights to RDS but not perform administrative operations.

The Amazon RDS instance is placed inside a **VPC with public subnets** but is restricted using a custom **security group (m1\_sg\_qa)** that allows inbound MySQL traffic (port 3306) **only from trusted IP addresses**, such as the EC2 instances within the same VPC or specific developer machines. This ensures database access is strictly controlled and auditable.

The Application Load Balancer (ALB) is secured with **target group health checks** and restricts traffic to port 8000, where the EC2-hosted model API is exposed. All traffic is monitored, and only ALB-managed ingress is allowed via the target group. Backend EC2 instances are further protected using security group rules that deny all inbound traffic except from the ALB and internal services.

Amazon S3 enforces **server-side encryption** (SSE-S3 or SSE-KMS) for all stored objects, including raw CSVs and model artifacts. Additionally, the attached bucket policy **denies all unencrypted connections (aws:SecureTransport = false)** to ensure data is only transmitted over HTTPS. Public access to buckets is fully disabled via BlockPublicAcls and RestrictPublicBuckets.

AWS Glue jobs and crawlers run in an **IAM-assumed role with restricted access**, allowing only s3, glue, and logs actions on specific resources. This prevents accidental data leaks or unintentional modification of unrelated Glue resources.

Finally, **SNS notifications** are configured with verified topics and endpoints, and the Lambda function sending alerts (SentSNS) can only publish to a specific SNS topic ARN, reducing the blast radius of misconfigurations.

Together, these policies and configurations create a secure, compliant, and production-ready ML pipeline that adheres to AWS security best practices while maintaining operational flexibility.

# Challenges and Addressed

One significant challenge encountered involved managing and updating database schemas effectively. To address this, AWS Glue Crawlers were integrated into the pipeline, automating the schema creation and maintenance processes. This automation significantly reduced manual labor and potential for error, ensuring the schema remained accurate and reflective of the latest data.  
  
Another critical challenge was integrating machine learning capabilities with a scalable front-end infrastructure. This was resolved by deploying AWS Lambda functions, specifically the RefreshAutoScaleGroup Lambda, which automated the deployment of new machine learning model artifacts to the EC2 instances managed by Auto Scaling Groups. This ensured that the front-end continuously provided the most current predictive capabilities without manual intervention.  
  
Maintaining data quality and ensuring accurate processing outcomes posed additional challenges. Lambda functions equipped with validation logic, such as SentSNS Lambda, were employed to automatically check processed data and send notifications via Amazon SNS. This allowed immediate awareness and swift action to correct any issues, greatly enhancing overall pipeline reliability.  
  
Lastly, deriving meaningful insights from queries required careful structuring of analytical queries. By segmenting and specifically analyzing price tiers and property-type performance, actionable business insights were extracted, enabling targeted improvements and informed strategic decisions. This structured approach provided clarity in identifying high-performing and underperforming segments, guiding effective operational and marketing strategies.