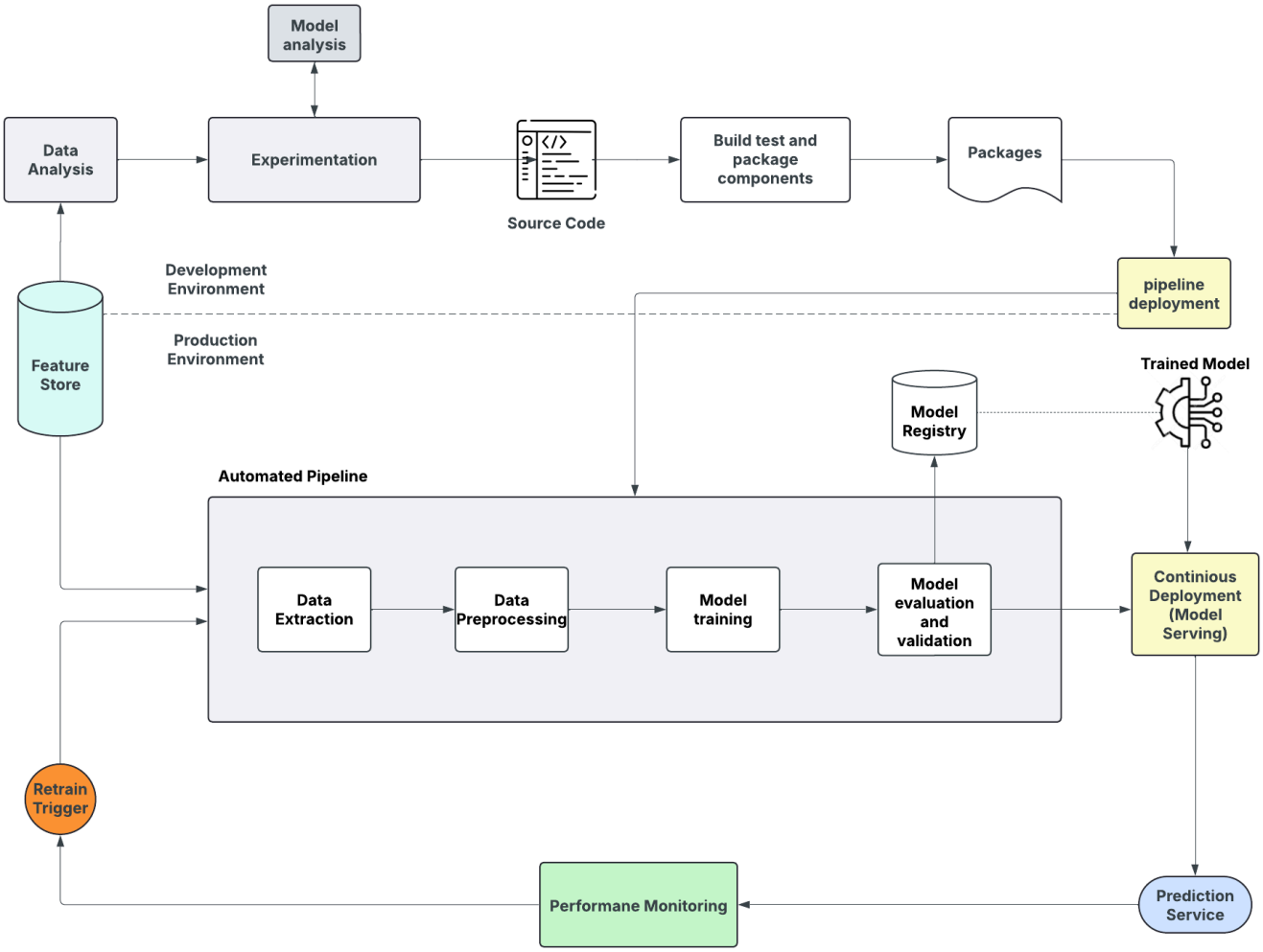
Task 04 : System Architecture Diagram

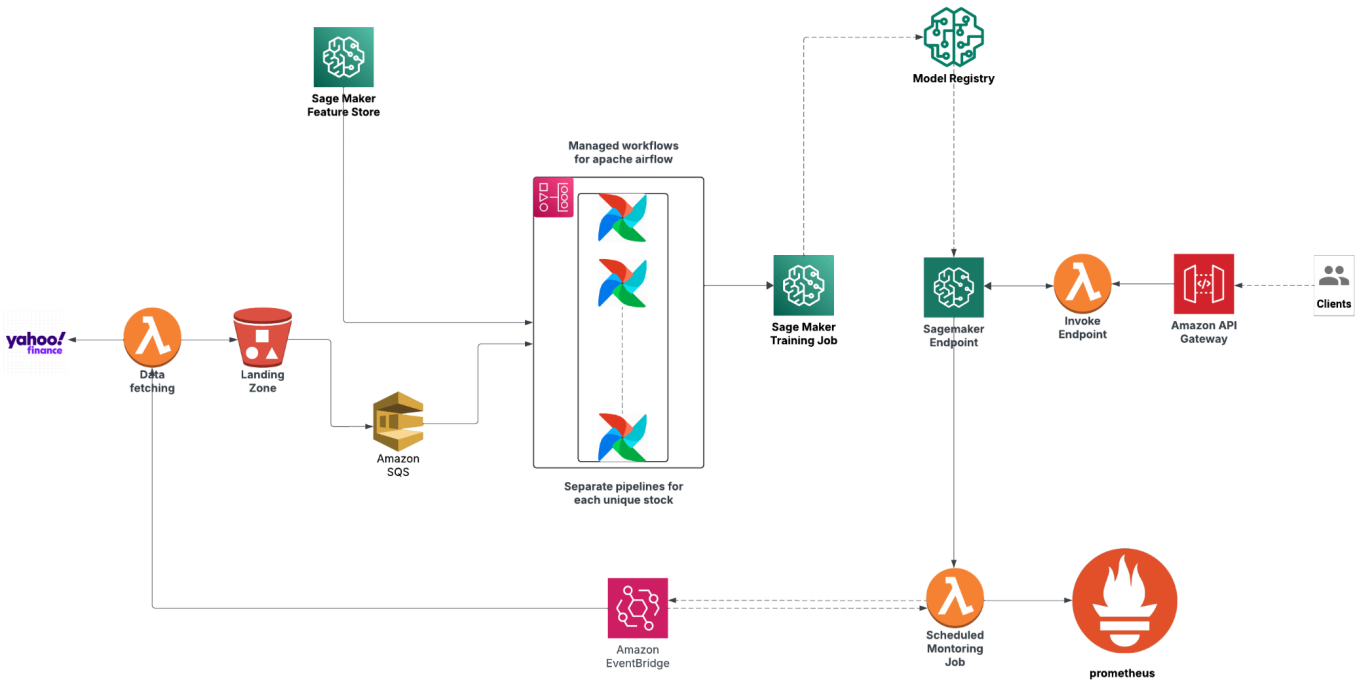
High level architecture



This diagram represents the **full MLOps lifecycle** for a stock forecasting system, ensuring:

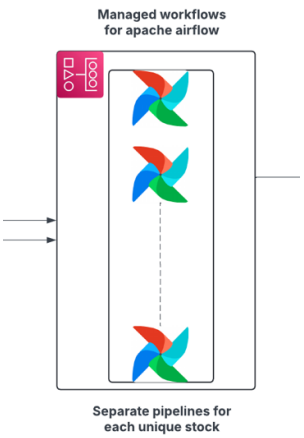
- **Automated Data Ingestion** (from APIs like Yahoo Finance).
- **Feature Engineering** (storing processed features in a Feature Store).
- **Model Training & Validation** (automated training workflows).
- **Model Deployment** (Continuous Deployment for serving predictions).
- **Performance Monitoring & Auto-Retraining** (detecting drift and retraining models when necessary).

Deployment Architecture



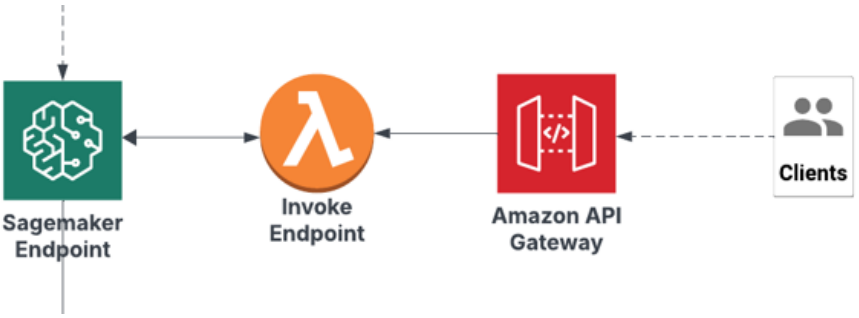
This architecture ensures that:

- **Stock data is collected** from **Yahoo Finance API** using serverless lambda functions.
- **Data is stored and processed** using AWS services and Apache Airflow.
- **Machine learning models are trained** using **AWS SageMaker**.
- **The best model is deployed as an API** via **SageMaker Endpoints**.
- **Monitoring ensures model performance is optimal**, with **automatic retraining if performance degrades**.



With the use of this workflow **we can extend this system for multiple stocks in the future since we just have to add an additional airflow pipeline**

Monitoring Job and eventbridge will take care of automated retraining and model monitoring which helps us to make the system fully automated.



We can actually deploy our model as a docker image in an EC2 instance or EKS if the traffic for model increases in the future (and even add load balancers)

Data fetching lambda function will be triggered when new stock data is available it fetches most recent data to the landing zone from yahoo finance API. After that we can configure our airflow environment to periodically trigger the ETL pipelines based on the accuracy drift(Via a service like SQS).

Since we use serverless lambda functions for most of the invoking processes it will be cost effective

Component Justification

Component	Technologies Used	Why This Solution?	Trade-offs
Data Ingestion	• AWS Lambda (Event-driven API calls) • Yahoo Finance API (Stock Data Source) • Amazon S3 (Landing Zone)	Automates fetching live stock market data and stores it in **S3 for scalability**.	Lambda has execution time limits so batch jobs may be needed for large datasets.
Feature Store	• AWS SageMaker Feature Store	Ensures consistency between training and inference data. Stores computed features for efficient reuse. *(Technical indicators related features)*	Requires additional cost & setup compared to traditional databases.
ETL pipeline	• Apache Airflow (ETL workflow automation) • Pandas, NumPy (Data transformation)	Ensures ETL process is scheduled and automated.	Sine we're using a managed service by AWS it can be costly
Model Training	• AWS SageMaker Training Job	Uses **managed training infrastructure** for scalability and cost-effectiveness.	Managed training jobs can be expensive for frequent retraining.
Model Registry	• AWS SageMaker Endpoint (Real-time inference) • Amazon API Gateway (Client-facing API) • AWS Lambda (Model Invocation)	Ensures low-latency predictions via API Gateway. AWS Lambda acts as an intermediary function to invoke the deployed model.	Real-time inference costs can be high if not optimized (batch inference can reduce costs).
Model Monitoring & Retraining	• Amazon EventBridge (Triggers retraining based on alerts) • Scheduled Monitoring Job (Lambda function for drift detection) • Prometheus + Grafana (Performance Monitoring)	Automatically detects drift and triggers model retraining via AWS SageMaker.	False positives in drift detection may cause unnecessary retraining
Prediction Service	• API Gateway + AWS Lambda + SageMaker Endpoint	Provides **real-time predictions to clients via an API**.	Scaling needs to be managed to handle traffic spikes.

Data Flow Explanation

This section describes how data moves through the system, including batch vs. streaming decisions and transformation stages.

Batch vs. Streaming Processing

1. Real-Time Streaming:
    - Stock price updates are streamed using AWS Lambda and Yahoo Finance API.
    - This enables immediate processing and low-latency inference using SageMaker Endpoints.
2. Batch Processing:
    - Scheduled Apache Airflow jobs process historical stock data.
    - Feature extraction & model retraining run when a change in model accuracy was detected and trigger request was sent.

Data Transformation Stages

1. Data Collection & Storage:
    - Yahoo Finance API fetches stock prices.
    - AWS Lambda stores raw data in S3 (Landing Zone).
2. Feature Engineering & Storage:
    - Airflow transforms raw data.
    - Features are stored in AWS SageMaker Feature Store.
3. Model Training & Selection:
    - SageMaker Training Jobs train multiple models.
    - MLflow or SageMaker Model Registry stores the best model.
4. Model Deployment:
    - Trained models are deployed to SageMaker Endpoints.
    - Predictions are served via API Gateway.
5. Monitoring & Retraining:
    - Prometheus can detect model drift.
    - If accuracy drops, Amazon EventBridge triggers retraining.
    - New models are validated before replacing the existing deployment.

System Interaction Points

- **Yahoo Finance API → AWS S3:** Raw data storage.
- **S3 → Feature Store (SageMaker Feature Store):** Feature consistency.
- **Airflow → SageMaker Training:** Automated model retraining.
- **SageMaker Registry → API Gateway:** Deploys the best model.
- **Prometheus → EventBridge:** Detects drift and triggers retraining.

Challenge Analysis and Mitigation Strategies

Implementing an **automated stock price prediction system** presents several technical and operational challenges. Below are the **five main challenges** and their **mitigation approaches**.

1. Ensuring Feature Consistency Between Training and Inference

Challenge

Stock price features (moving averages, volatility, momentum indicators) must be consistent between training and inference. If there is a mismatch in how features are computed, the model may produce inaccurate predictions.

Mitigation Approach

- Use a Centralized Feature Store (AWS SageMaker Feature Store)
    - Ensures that the same features are used for training and serving.
    - Allows for fast retrieval of computed features to avoid recalculating them at inference time.
- Precompute Features for Fast Inference
    - Instead of computing moving averages on the fly, precompute them in Airflow and store them in the Feature Store.
    - Use cached feature lookups in the model deployment pipeline.

2. Handling Model Drift and Changing Market Conditions

Challenge

Stock market behavior changes over time due to economic events, company earnings reports, and macroeconomic factors. This causes model drift, where the model trained on past data becomes inaccurate over time.

Mitigation Approach

- Implement Model Drift Detection (Prometheus)
    - Continuously monitor RMSE, MAPE, and directional accuracy in real time.
    - Set threshold-based triggers (e.g., retrain only if RMSE increases by more than 10%).
- Automated Retraining Pipeline
    - Use Apache Airflow to schedule automatic retraining in AWS SageMaker when model performance degrades.
    - Trigger new model training jobs via Amazon EventBridge based on drift detection.

3. High Cost and Latency of Real-Time Predictions

Challenge

Serving stock price predictions in real time using SageMaker Endpoints or similar services can be expensive and introduce latency if not optimized.

Mitigation Approach

- Optimize Model Deployment Strategy
    - Use AWS Lambda + API Gateway for on-demand inference instead of keeping SageMaker Endpoints always active.
    - Use Batch Predictions for non-urgent requests to reduce cost.
- Auto Scaling for High-Traffic Periods
    - Enable SageMaker Endpoint AutoScaling to adjust based on request load.
    - Use Amazon CloudFront caching for frequently queried stock symbols to reduce repeated model invocations.