Trading at Close

Cloud Engineering Project
Group 7

Team Members
Ayush Agarwal
Kevin Li
Kexian Wu
Jialong Li

Project Introduction

Introduction

Machine Learning Problem:

- Prediction Closing Price Movements of Nasdaq-listed stocks using order book and closing

Importance of Predicting Closing Price Movements:

- High-Stakes Environment: Stock exchanges are dynamic, and closing prices determine the market sentiment.
- Critical Last Minutes: The final ten minutes of trading see heightened volatility and rapid price changes, impacting daily economic narratives.
- Market Efficiency: Accurate predictions help improve market efficiency, providing better prices and insights for all participants.

Why Need Cloud:

- Scalability: Handle large datasets and high computational demands by scaling resources up or down as needed.
- Real-Time Processing: Use services like AWS Kinesis Data Streams and Lambda for real-time data ingestion and processing.
- Flexibility: Easily integrate various cloud services (ECS, RDS, S3) to build a robust and flexible architecture.

Data Collection and Preparation

Data Collection and Preparation

Kaggle dataset of closing prices of 200 Nasdaq stocks across 480 days, by every 10 seconds.

In real life, we would get real time by yahoo finance and other APIS

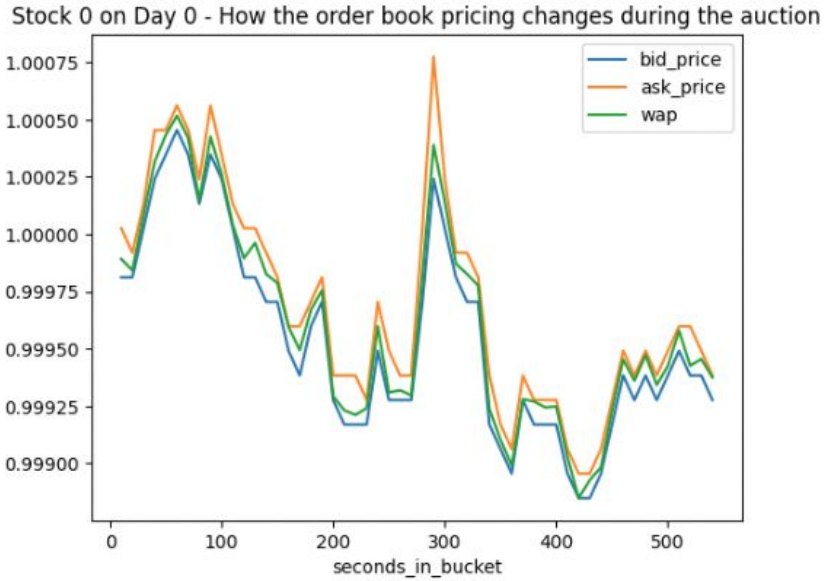
Original features:

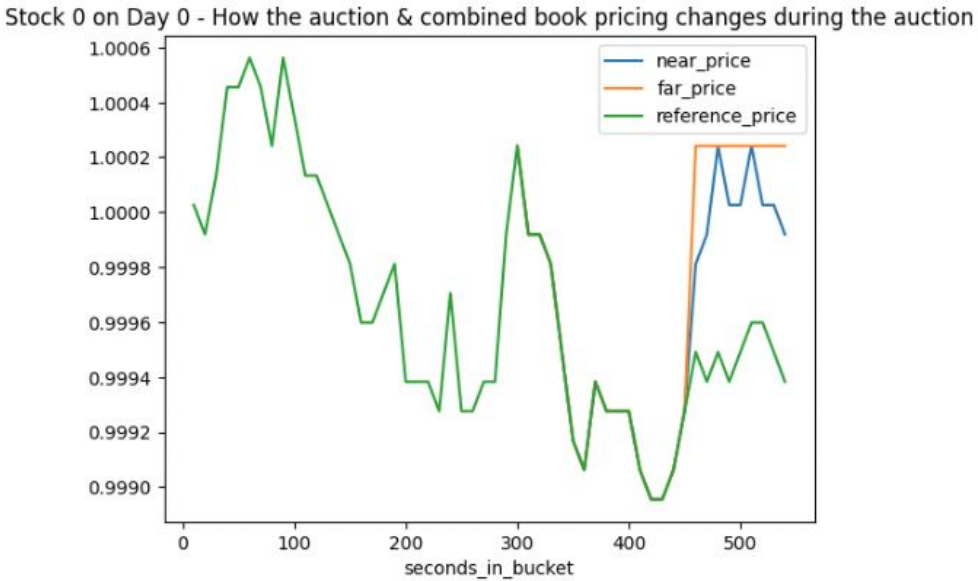
- Market Metrics:
  - Seconds in Bucket: Captures the time within the trading session, useful for understanding temporal patterns.
  - Bid and Ask Sizes: Indicate the quantity of stocks available for buying and selling at current bid and ask prices.
- Price and Volume Indicators:
  - Reference Price, Far Price, Near Price: Provide different perspectives on the stock prices, essential for price movement predictions.
  - Matched Size, Imbalance Size: Reflect the volume of matched trades and the imbalance between buy and sell orders, highlighting supply-demand dynamics.

Engineered Features

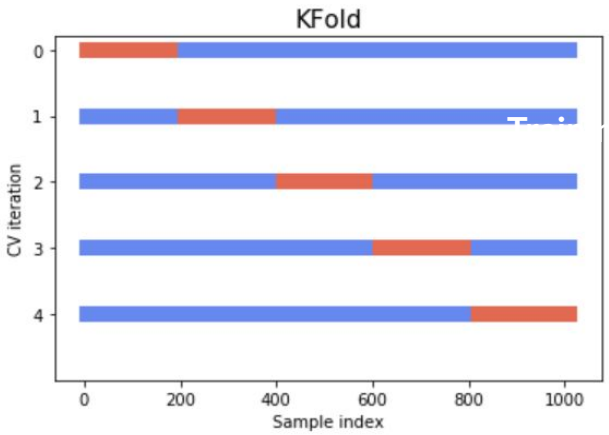
- Imbalance Metrics:
  - Imbalance Ratio 1 (imb_s1): Measures the difference between bid and ask sizes, normalized by their sum, indicating market sentiment.
  - Imbalance Ratio 2 (imb_s2): Compares imbalance size with matched size, normalized, providing insights into trade intensity and pressure.
- Price Difference Metrics:
  - Price Differences: Calculated between various price indicators (e.g., reference price vs. far price), offering detailed insights into price discrepancies and potential arbitrage opportunities.

Exploratory Data Analysis

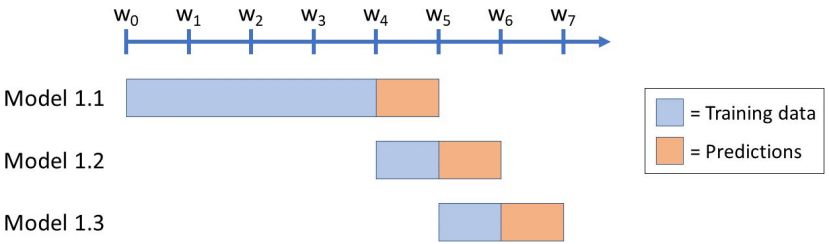




Model Training and Inference



5 fold time series CV across 480 days



Incremental Training with XGBoost

Xgboost Advantages

- **Handling Large Datasets**: Efficiently processes large volumes of data from order books and closing auctions. Entire training takes about 40 seconds.
- **Incremental Training**: Supports training on new data without retraining the entire model.

Hyperparameters

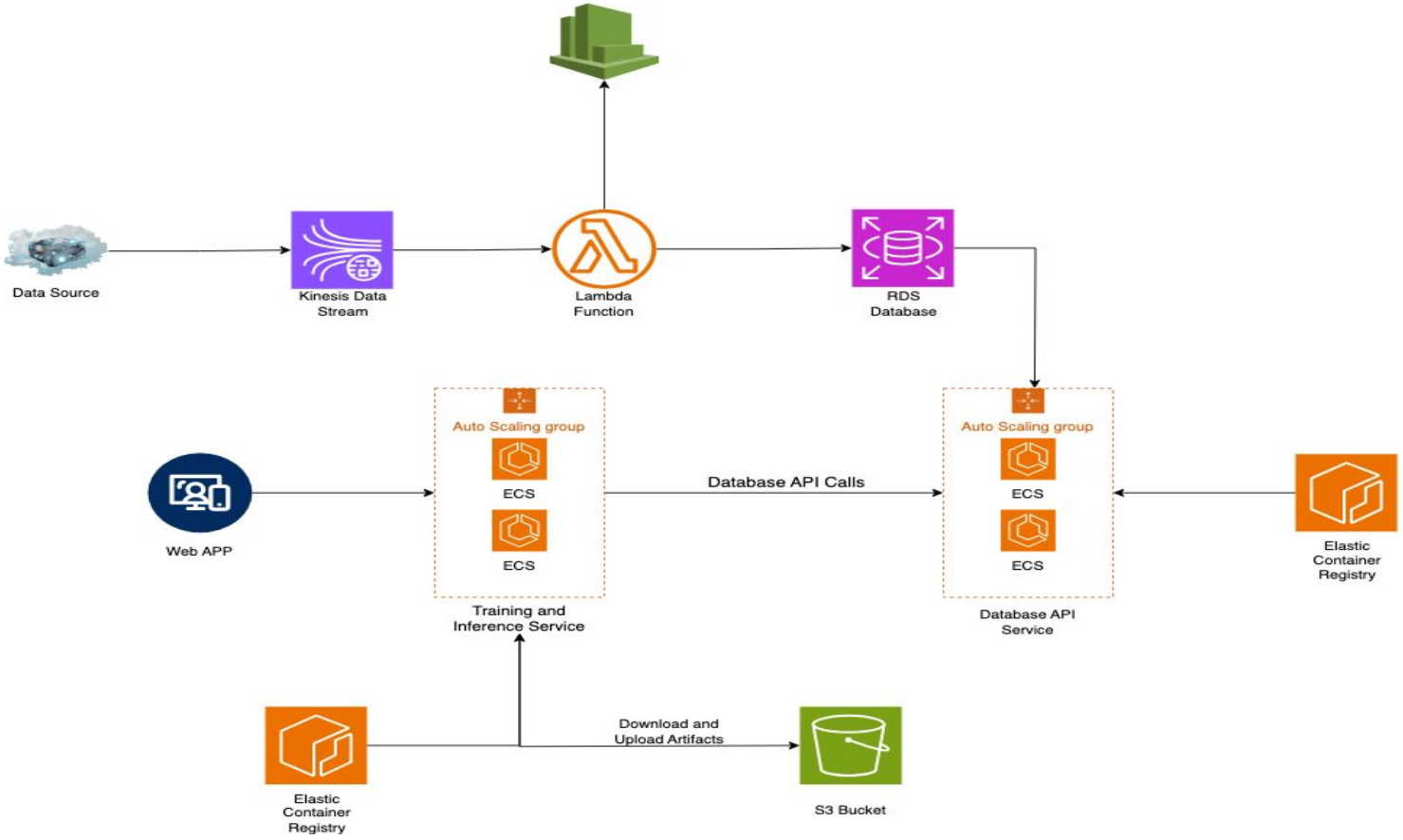
- **Objective** : reg:squarederror
- **Eval_metric** : Mae
- **Learning_rate** : 0.01

Results

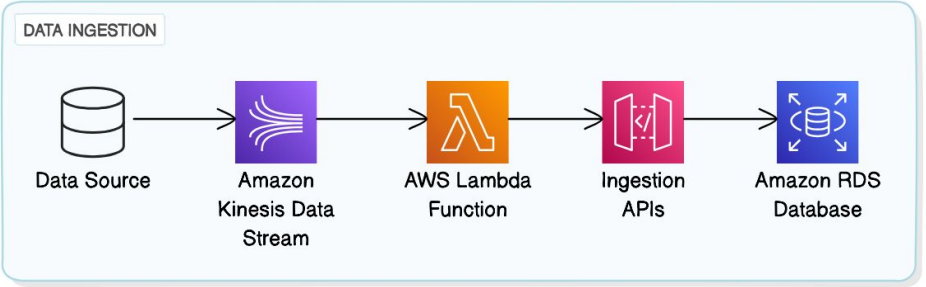
- Validation MAPE : 5.86
- Testing MAPE : 5.93

Model Deployment

Deployment Workflow



Data Ingestion



Data Ingestion Workflow Architecture

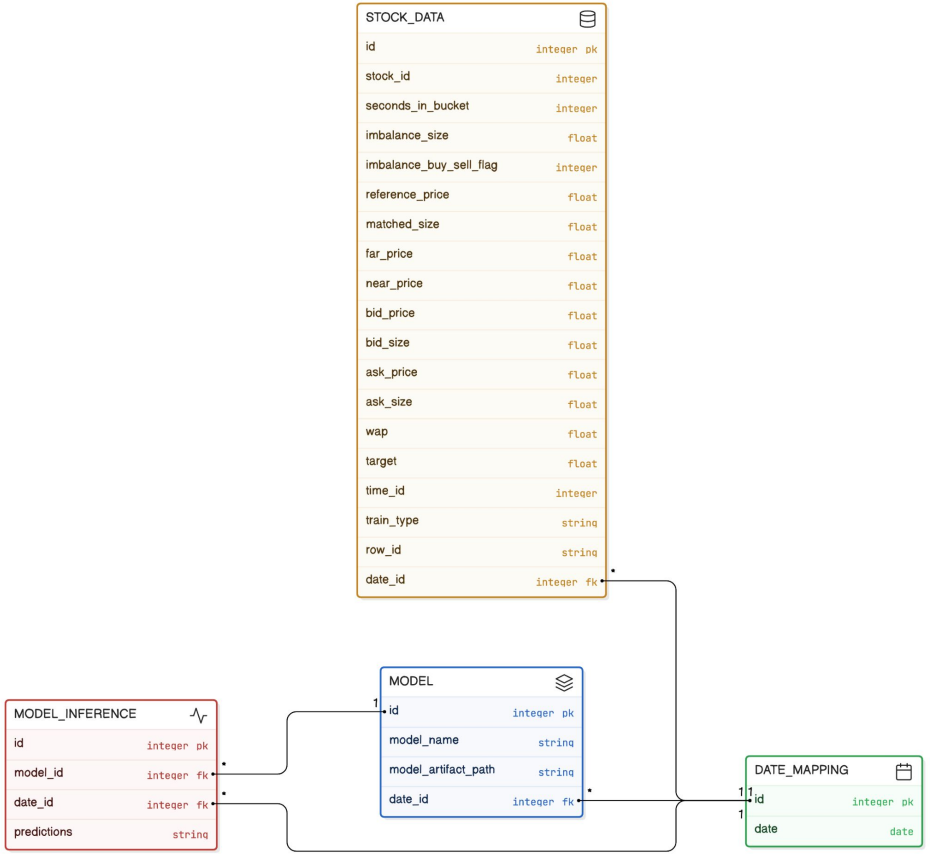
Stream Code

Lambda Code

Ingestion Workflow

1. **Data Source to Kinesis**:
   - The server subscribes to an Amazon Kinesis Data Stream.
   - Data from the source is continuously streamed to the Kinesis Data Stream.
2. **Trigger Lambda Function**:
   - As soon as data arrives in the Kinesis Data Stream, it triggers an AWS Lambda function.
3. **Lambda Function Processing:**
   - The Lambda function processes the incoming data.
   - It calls specific ingestion APIs to transform and load the data.
4. **Data Storage:**
   - The processed data is then ingested into an **Amazon RDS** database for persistent storage

DataBase Schema and APIs



APIs Overview

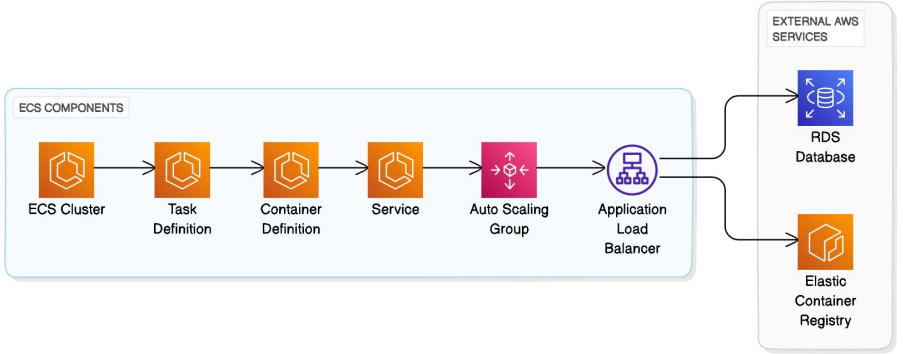
1. **Stock Data API:**
   - Endpoint: /stock_data/
   - Description:
     - GET and POST: Fetches and Ingest stock data.

2. **Date Mappings API:**
   - Endpoint: /date_mappings/
   - Description:
     - GET and POST: Retrieves and Adds date mappings.

3. **Model Inferences API:**
   - Endpoint: /model-inferences/
   - Description:
     - GET: Retrieves and posts inferences.

4. **Models API:**
   - Endpoint: /models/
   - Description:
     - GET: Fetches and adds model details.

Hosting Database APIs



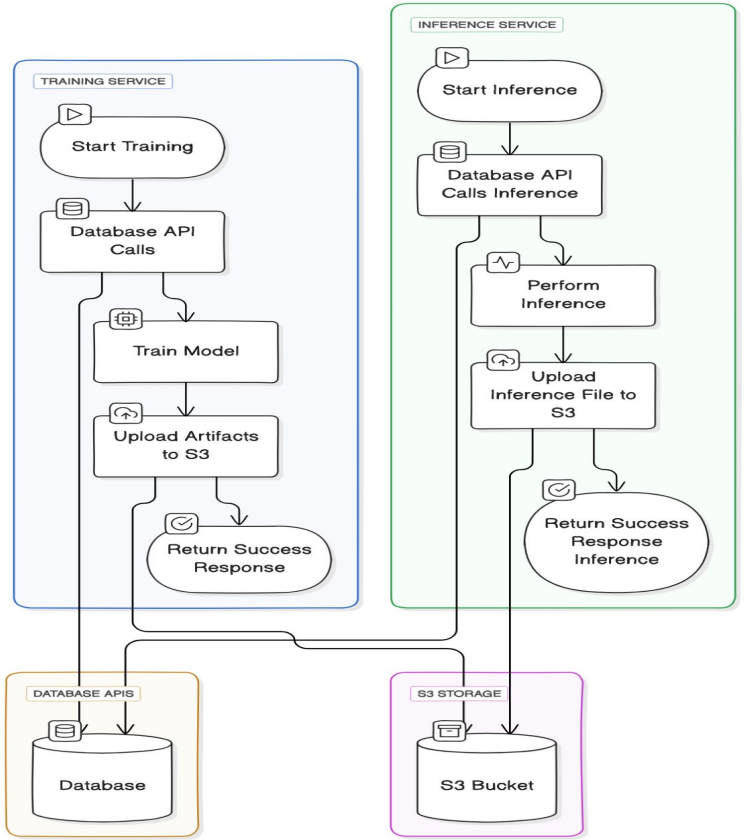
Database APIs Deployment Architecture

Source Code

Deployment Overview

- **ECS Cluster Creation**: An ECS cluster was created using EC2 instances for control over resources.
- **Task Definitions**: Configured to specify Docker images, CPU, memory, and network settings.
- **Service Configuration**: Set up to manage deployment and scaling of tasks, ensuring high availability.
- **Auto Scaling Setup**: Implemented policies to adjust running tasks based on resource utilization.
- **Load Balancer Configuration**: Deployed ALB to distribute traffic across ECS tasks for fault tolerance.
- **RDS Integration**: Integrated ECS tasks with an RDS instance for a reliable database backend.
- **Elastic Container Registry (ECR)**: Stored Docker images in ECR for efficient management and deployment.
- **Deployment and Monitoring**: Deployed services and used CloudWatch for logging and monitoring.

Model Training and Inference



Deployment

- Deployed as a separate service using ECS Tasks, similar to Database APIs.

Training Steps

1. Fetch data for the given date.
2. Retrieve the model to be fine-tuned.
3. Fine-tune the model with the provided data.
4. Upload the trained model artifact to S3.
5. Ingest trained model details into the database.

Inference Steps

1. Fetch data corresponding to date_id.
2. Retrieve the model artifact from S3.
3. Run inference on the data and generate a CSV.
4. Upload the CSV file to S3 and update the database.

Configuration, Logging, Monitoring

Config and Secrets Management

Secret name	Description
prod/app/s3	s3 creds
rds!db-f0734267-bac7-4bce-afc9-8e039aa30586	Secret associated with primary RDS DB instance: arn:aws:rds:us-east-1:058264109996:db:optiverdb

Key	Type	Value
REGION_NAME	value	us-east-1
DB_SECRET	value	rds!db-f0734267-bac7-4bce-afc9-8e039aa30586
DB_NAME	value	postgres
DB_HOST	value	optiverdb.czys2w24206n.us-east-1.rds.amazonaws.com
BASE_API	value	http://optiver-db-service.optiver-db
DATA_API	value	/stock_data/
INFERENCE_API	value	/model-inferences/
MODEL_API	value	/models/
REGION	value	us-east-1
S3_BUCKET_NAME	value	cloud-project-optiver

Sensitive Info (Stored in Secrets Manager)

- RDS Credentials
- AWS Access Key and ID

General Variables (Stored as Environment Variables)

- API URLs
- Bucket Name
- Bucket Region
- Secret Name

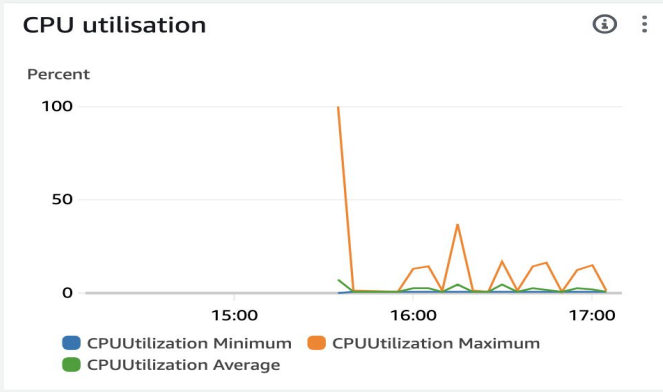
Logging and Monitoring

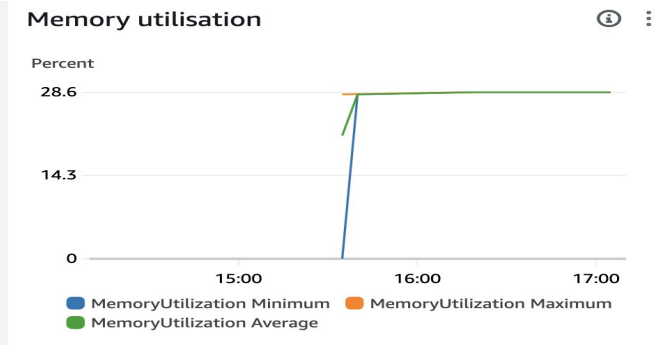
Log group	Log class	Anomaly d...
/aws/lambda/KinesisToRDSLambda	Standard	Configure
/ecs/optiver-db-service-task	Standard	Configure
/ecs/optiver-frontend-service-task	Standard	Configure
/ecs/optiver-train-service-task	Standard	Configure

CloudWatch Log Groups

2024-05-27T15:37:21.080Z	05/27/2024 03:37:21 PM - optiver.app.routers.models - INFO - Reading models with provid…
2024-05-27T15:37:21.131Z	05/27/2024 03:37:21 PM - optiver.app.routers.models - INFO - Retrieved 12 models, page …
2024-05-27T15:37:21.133Z	05/27/2024 03:37:21 PM - optiver.app.database - INFO - Database session closed.
2024-05-27T15:37:30.698Z	05/27/2024 03:37:30 PM - optiver.app.main - INFO - Healthcheck endpoint called.
2024-05-27T15:37:33.389Z	05/27/2024 03:37:33 PM - optiver.app.database - INFO - Creating a new database session.
2024-05-27T15:37:33.390Z	05/27/2024 03:37:33 PM - optiver.app.routers.models - INFO - Reading models with provid…

Sample Cloudwatch Logs





Health Monitoring for ECS Services

AutoScaling

```
{
    "ScalingPolicies": [
        {
            "AutoScalingGroupName": "optiver-frontend",
            "PolicyName": "optiver-frontend-scaling-policy",
            "PolicyARN": "arn:aws:autoscaling:us-west-2:123456789012:scalingPoli
cy:uuid:autoScalingGroupName/optiver-frontend:policyName/optiver-frontend-scalin
g-policy",
            "PolicyType": "TargetTrackingScaling",
            "TargetTrackingConfiguration": {
                "PredefinedMetricSpecification": {
                    "PredefinedMetricType": "ALBRequestCountPerTarget",
                    "ResourceLabel": "app/Optiver-db-ALB/509f03fb9cb4bdcd"
                },
                "TargetValue": 20.0
            },
            "EstimatedInstanceWarmup": 300
        }
    ]
}
```

Autoscaling Policy

AutoScaling ensures services automatically adjust resources to handle varying loads, maintaining performance and availability.

Key Terms

- **Target Value:** The desired average number of requests per instance. When actual requests deviate from this target, AutoScaling adjusts the number of instances.
- **Predefined Metric Type**: The metric used for scaling, typically ALBRequestCountPerTarget, which tracks requests per target in an Application Load Balancer (ALB).

Benefits

- **Cost Efficiency**: Matches resource allocation to demand, reducing costs.
- **Improved Performance**: Scales up during peaks to handle increased traffic.
- **High Availability**: Maintains service availability by adjusting instances as needed.

Planning and Budgeting

Budget Estimates

AWS Service Name	Average Monthly Cost ($)
Amazon Kinesis Data Streams	23.74
AWS Lambda	111.39
RDS for PostgreSQL	782.20
DB-Service (Fargate)	72.08
Train-Service (Fargate)	7.51
Frontend-Service (Fargate)	72.08
Elastic Container Registry	0.30
Simple Storage Service (S3)	0.46
CloudWatch	1.2

Average Yearly Cost

12,851 USD

Estimate URL : https://calculator.aws/#/estimate?id=841636621a5c930915d4d195691b397e5f0b2871

Demos

Optiver Trading Demo

- **Overview**
  - UI Demo
    - Training Interface
    - Inference Interface
    - Data Visualization Interface
  - Model Training Demo
  - Model Inference Demo
  - API Demos
  - Data Streaming to Database

- **Demo Links**
  - **Web UI Link :** Website
  - **S3 Bucket Link :** Optiver-bucket
  - **ECS Console :** Console

Thank You