

Price Prediction App

Presented by: Rob Wygant



Overview

- About
- Workflow/User Interface
- System Design
- Cloud Infrastructure
- Future Work

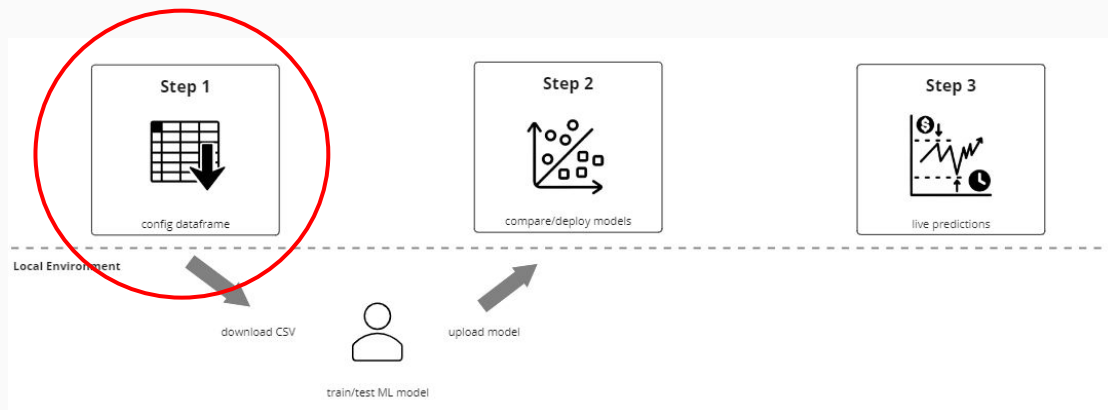
About Price Prediction App

- **What?** Provides basic infrastructure to label/train, compare, and deploy machine learning classifiers on future price movement of financial assets.
- **Why?** Streamline the infrastructure setup required to compare and deploy live models so focus can be put into model development/iteration
- **Who?** Data Scientist/Analytics, Machine Learning Engineers, Quantitative Finance

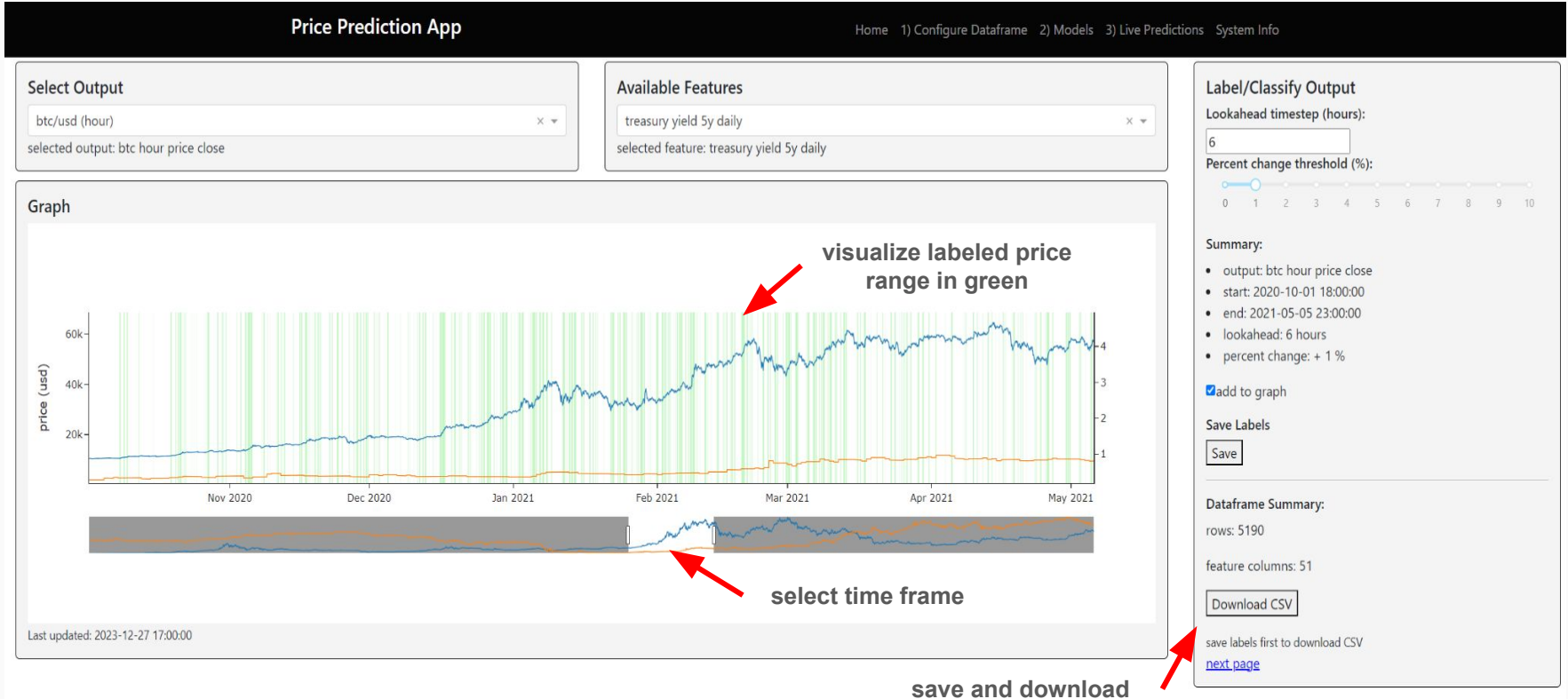
Workflow

- Step 1: Configure dataframe

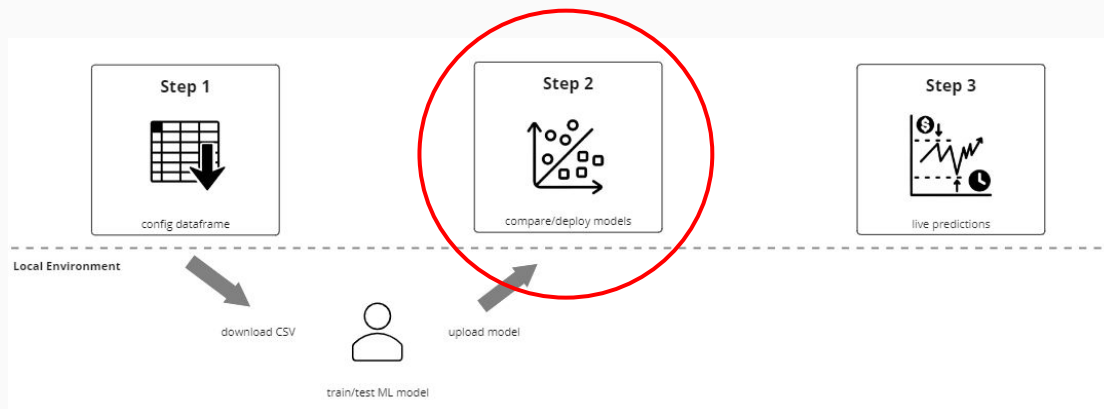
- Select target financial asset price to be classified (currently serving only BTC/USD)
- Explore over 50 macro economic indicators updated hourly
- Select time frame of interest
- Label historic price movement based on a lookahead/percent change threshold
- Download CSV for local model training



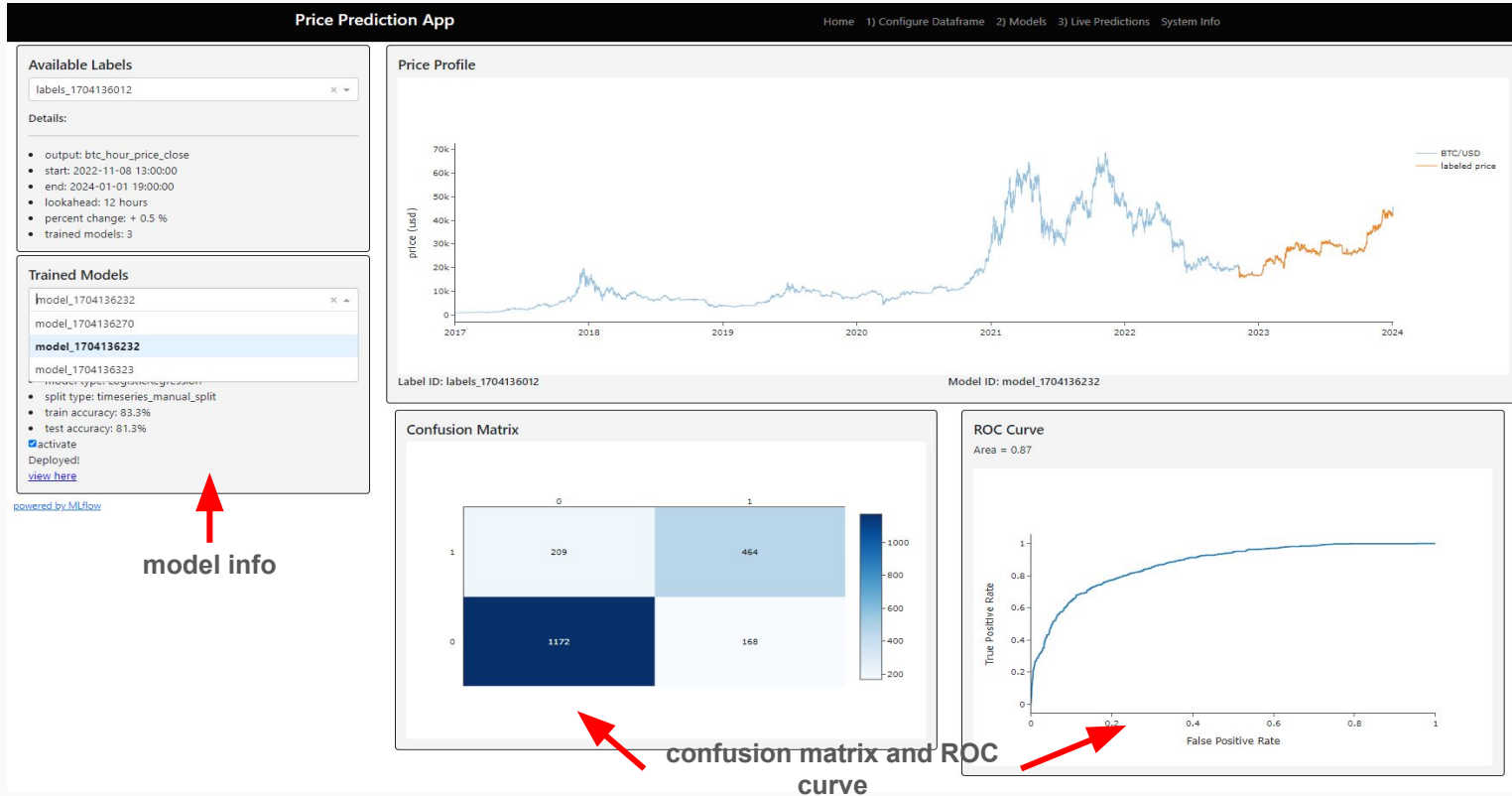
User Interface (Step 1)



- Step 2: Upload and compare trained models
 - Train and develop model pipeline locally
 - Upload to Price Prediction App database and MLflow instance
 - Compare models grouped by labels
 - Activate specific models for live deployment

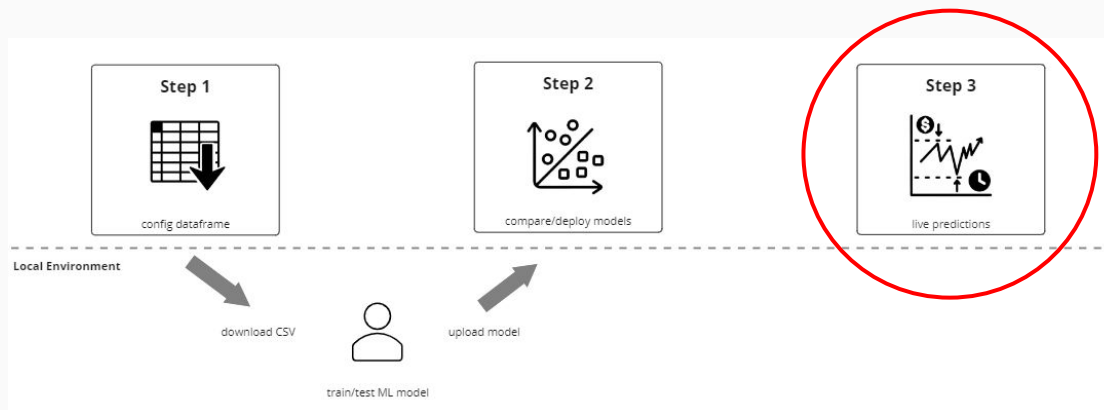


User Interface (Step 2)



Workflow

- Step 3: Live predictions
 - Visualize live price and available models
 - Select from deployed models and view predicted hourly price movement
 - Observe model performance based on the live prediction results



User Interface (Step 3)

Price Prediction App

Home 1) Configure Dataframe 2) Models 3) Live Predictions System Info

Select Deployed Model

Running Accuracy

Model ID	running accuracy
model_1704136402	0.83
model_1704136232	0.58
model_1704136323	0.58

Details:

- model type: LogisticRegression
- labels ID: labels_1704136012
- lookahead: 12 hours
- percent change: 0.5%

ID: model_1704136232
prediction type: classification

[model performance](#)

select model/details

Live Prediction

current:

\$43,357.80

2024-01-03, 16:09:10

ID: model_1704136232

prediction type: classification

prediction:

less than: \$45,515.44

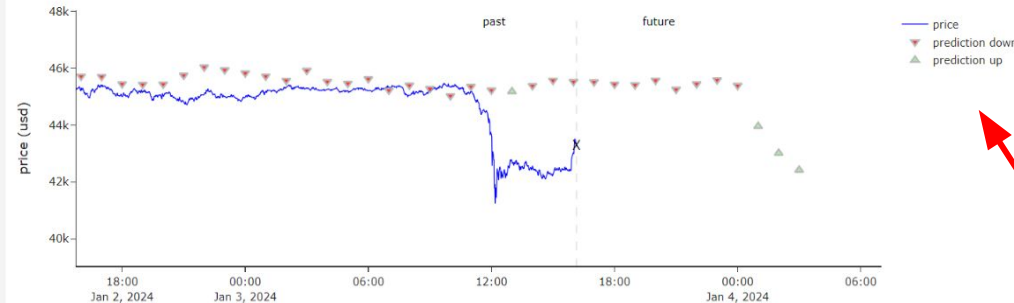
classification: negative (-)

next prediction: 2024-01-03 17:00:00

message:

percent error: 4.74%

status: true negative



Next Predictions

ID: model_1704136232

prediction type: classification

timeframe	threshold	prediction
1 hours	\$45,515.44	down
2 hours	\$45,432.73	down
3 hours	\$45,410.46	down
4 hours	\$45,570.74	down
5 hours	\$45,254.14	down
6 hours	\$45,445.36	down
7 hours	\$45,585.50	down
8 hours	\$45,389.40	down
9 hours	\$43,955.75	up
10 hours	\$43,010.98	up
11 hours	\$42,417.34	up

visualize predictions

User Interface (System Info)

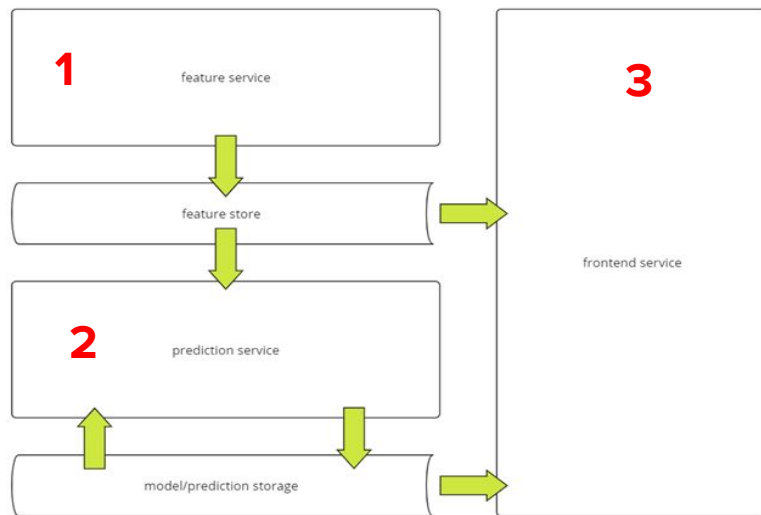
- System Info
 - Observability for cloud infrastructure
 - Powered by Grafana and Google Cloud APIs



System Design

- Containerized, independent services architecture
- Two databases serving the feature store and prediction service

System Overview



System Design

- Feature Service

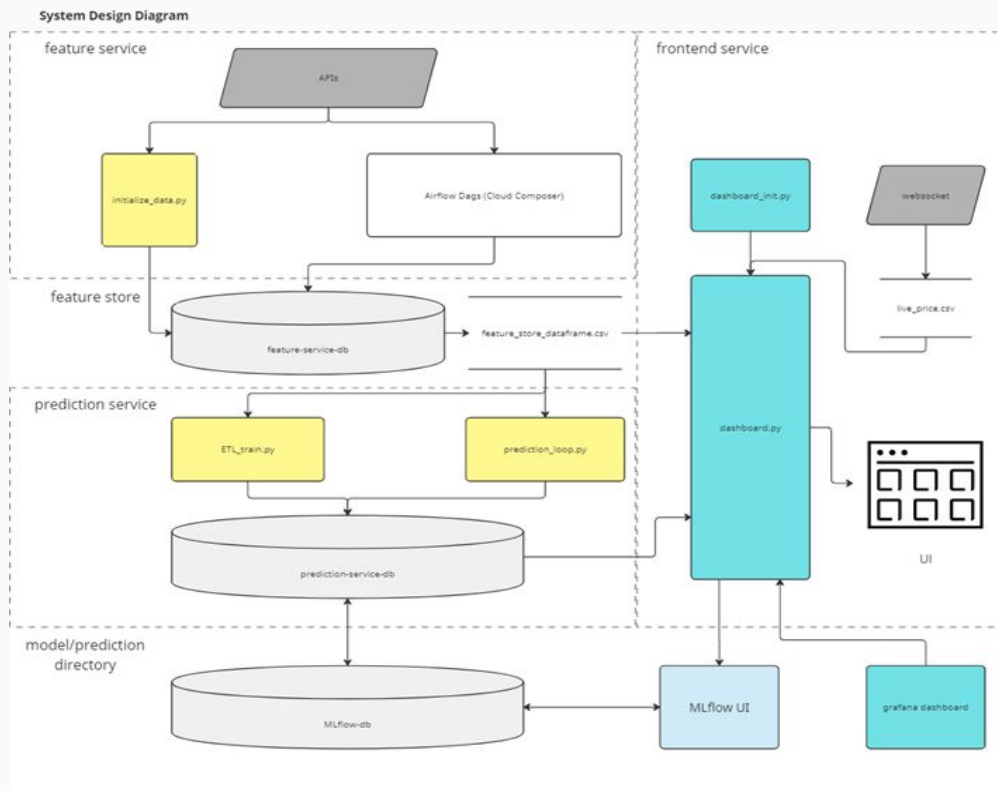
- Routes data from various external price data APIs to feature service database
- Apache Airflow/scheduling loop function

- Prediction Service

- Entry point for saving labels and trained model
- Scheduling loop to pull active model objects and make predictions (hourly)

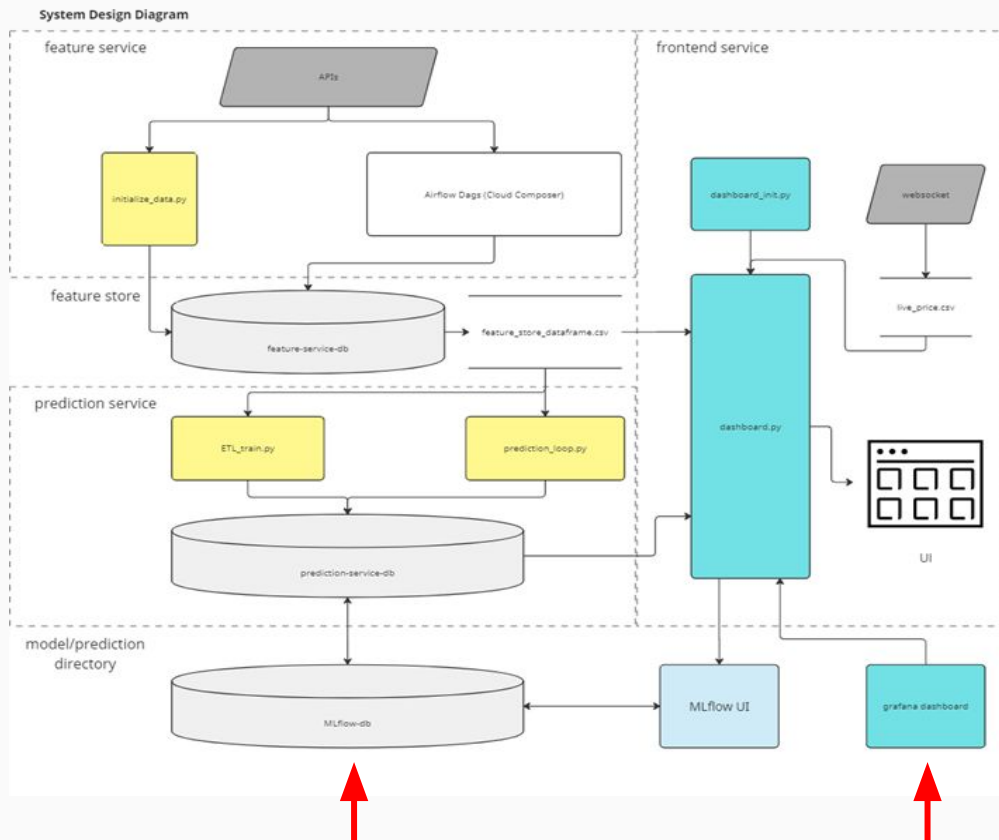
- Frontend Service

- User Interface
- Plotly Dash



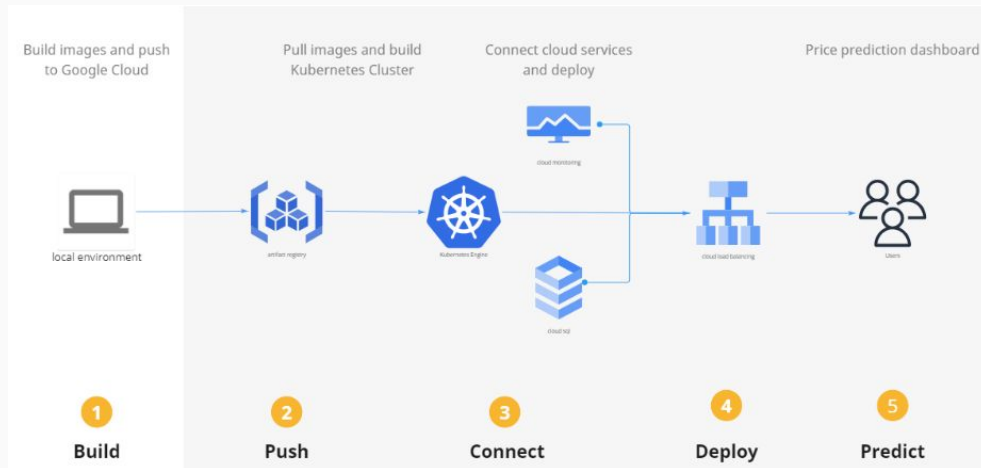
System Design

- **MLFlow**
 - Stores model directory info for trained models
 - Runs in parallel with prediction service database
 - Accessible via API
- **Grafana**
 - Embedded dashboard for cloud infrastructure observability
 - Connects with Google Cloud API

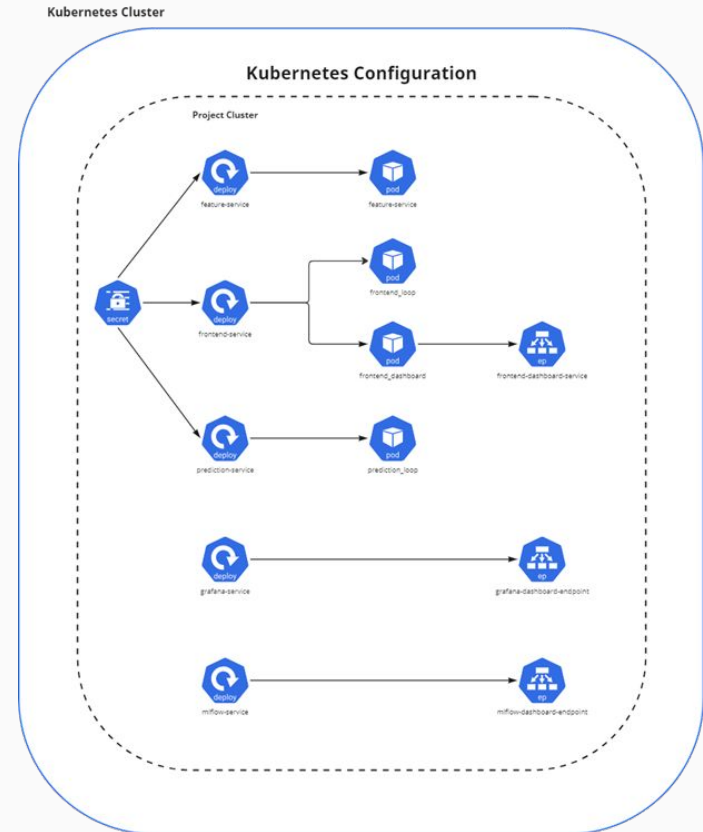


Cloud Infrastructure

- Deployed with Google Cloud Computing
 - CloudSQL (PostgreSQL instance)
 - Container Registry
 - Kubernetes Engine
 - Cloud Storage
- Deployment flow



- **Kubernetes Cluster**
 - Database and API secrets
 - Single namespace for the three services
 - Separate MLflow namespace for Helm deployment



Future work

- Add multiple target asset pairs (other than just BTC/USD)
- Integrate Kubeflow for model pipeline deployment
- Implement unit testing
- Continue to refactor and simplify code