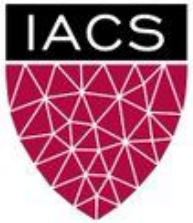
Crypto Forecasting App - Project Outline

Connor Capitolo

David Assaraf

Tale Lokvenec

Jiahui Tang



Harvard John A. Paulson School of Engineering and Applied Sciences

IACS Institute for Applied Computational Science

Outline

- Project Scope
- Project Workflow
- Process/Data Flow
- Backend Infrastructure
- App Design
- Data
- Models

Problem Definition

The current state of the crypto market is extremely volatile. Due to lack of experience/involvement from traditional actors, there is a lack of systematic investment strategies in the crypto environment; therefore, there is an opportunity to extract value from an accurate prediction of the price dynamics of pairs.

The scope of this project is to create a Proof of Concept to see if there is opportunity when using Deep Learning in crypto markets.

Objectives

1. Bridging the lack of structure dealing with crypto exchanges in building a scalable and modular database architecture that will gather various features from different exchanges (starting with Binance) for ‘pairs’ (a ‘pair’ refers for instance to the dynamics of the market for BTC vs USDT)

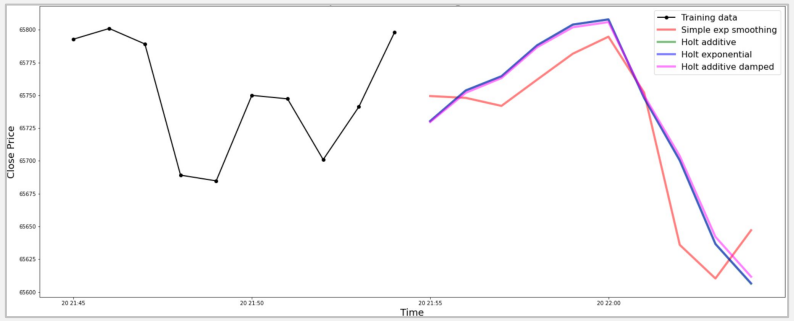
2. Building a predictive ML/DL model using real-time predictions that will enable us to gain insights as to how the market is evolving over time in order to inform trading decision making

Final Product (Tentative Design)

Home

Crypto Forecasting

Example Graph (Close Price on Y-axis, Time on X-axis)



BTCUSDT

Select Crypto Exchange
(drop-down)

Close Price Predictions

$X.xx	$X.xx	$X.xx	$X.xx
1 min.	5 min.	15 min.	30 min.
$X.xx	$X.xx	$X.xx	$X.xx
1 hour	1 day	1 week	1 month

Final Product (Style Template with some Components)

Cryptos

Home Watchlist

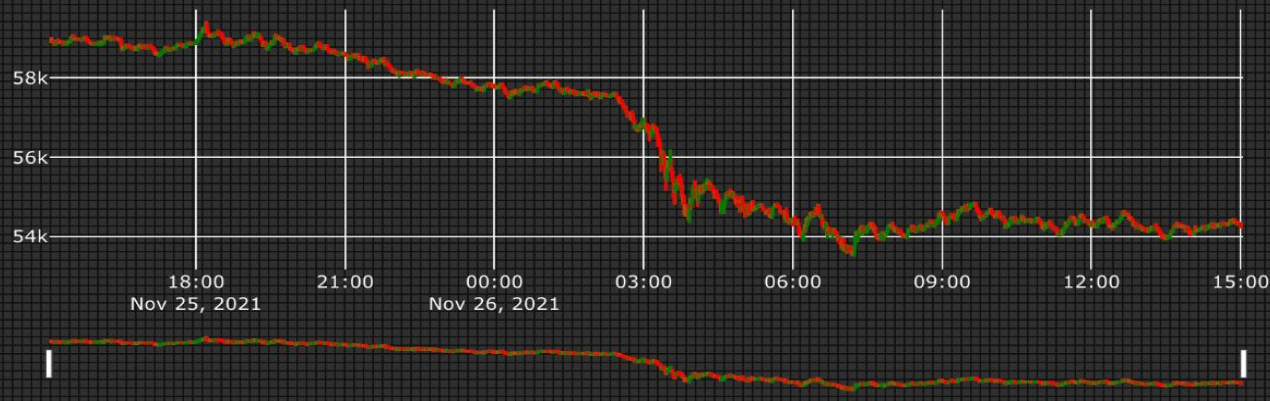
Live Market Data

Symbol	Name	Price	Price Change(%)
BTC-USD	Bitcoin USD	54,439.7	-0.5%
ETH-USD	Ethereum USD	4,076.82	-1.24%
BNB-USD	BinanceCoin USD	594.09	-2.55%
USDT-USD	Tether USD	1.0013	0.07%
SOL1-USD	Solana USD	189.9	-3.6%

Pricing data is updated frequently. Currency in USD

Live Predictions

Symbol: Bitcoin USD



In the News

Mr Goxx, the cryptocurrency-trading hamster, dies

The cryptocurrency-trading hamster Mr Goxx has died. The rodent, who shot to internet fame for his ability to often outperform human investors using a specially built trading cage, died on Tuesday. The furry financier's official Twitter account announced the hamster's death to his 18,000 followers on social media.

Home Symbols Watchlist Search Account

Project Scope

Proof Of Concept (POC)

- Setup database infrastructure, gathering both the historical data and the real-time data from Binance exchange
- Perform data exploration and data processing
- Experiment on some baseline models: last price, exponential smoothing
- Develop training pipeline for one specific pair (BTC-USDT)
- Benchmarking our model against baseline models

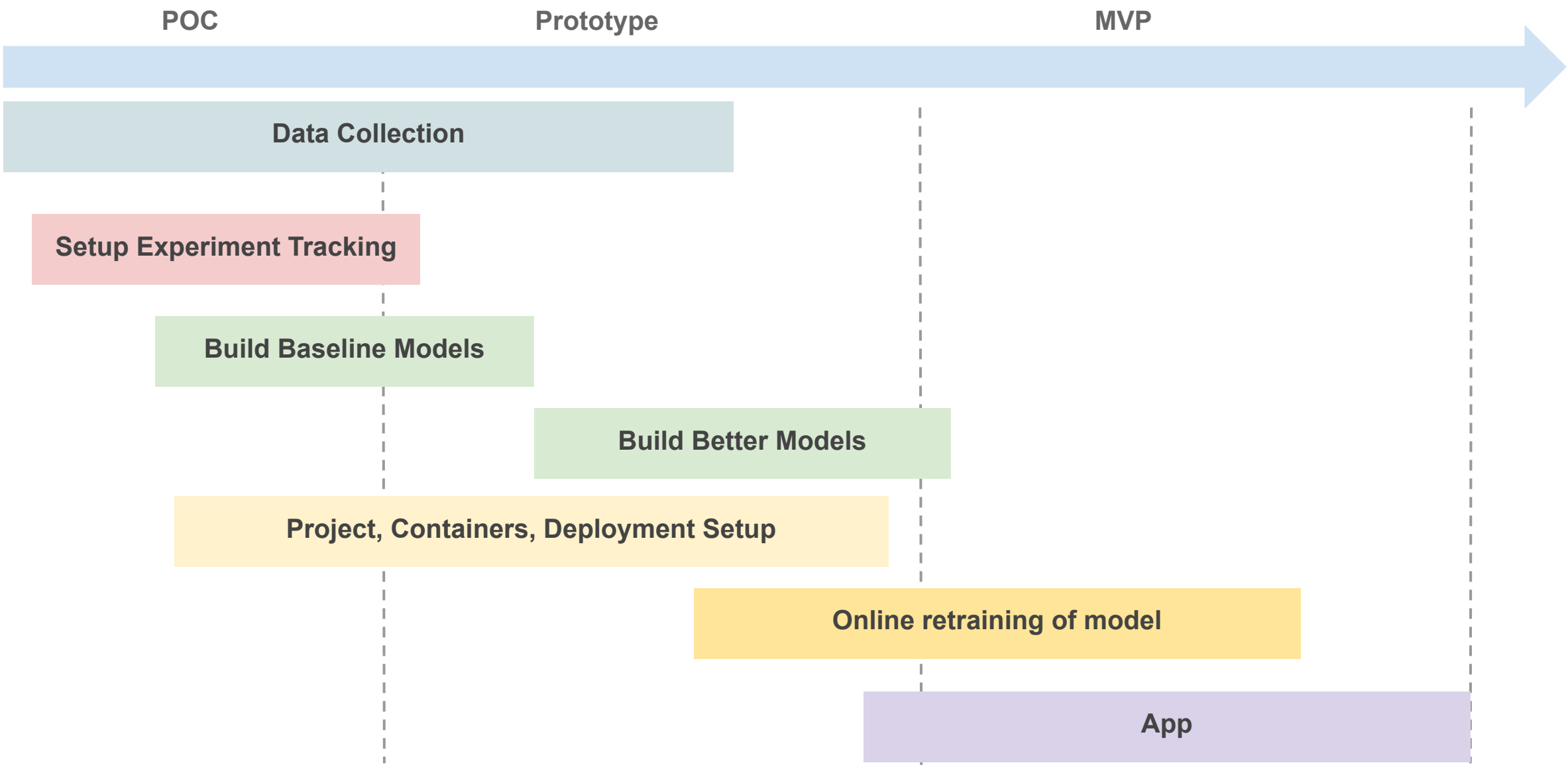
Prototype

- Develop a more advanced model, significantly improving over the different baselines
- Create a mockup of screens to see how the app will look
- Be able to launch the query of new pairs from the frontend side
- Perform regular data tests in order to verify the quality of the data
- Deploy models utilizing FastAPI to serve model predictions

Minimum Viable Product (MVP)

- Setup the front-end in order for users to interact with the API
- Provide real-time predictions using the deployed model
- Provide recommendations for user's investment
- Perform regular re-training of the deployed models

Project Workflow



Process (People)

- Collect initial data from Binance API
- Keep querying real-time data
- EDA on initial data
- Time Series modeling (single/multiple prediction approach)
- User selects a certain pair to obtain historical data and predictions
- View prediction results

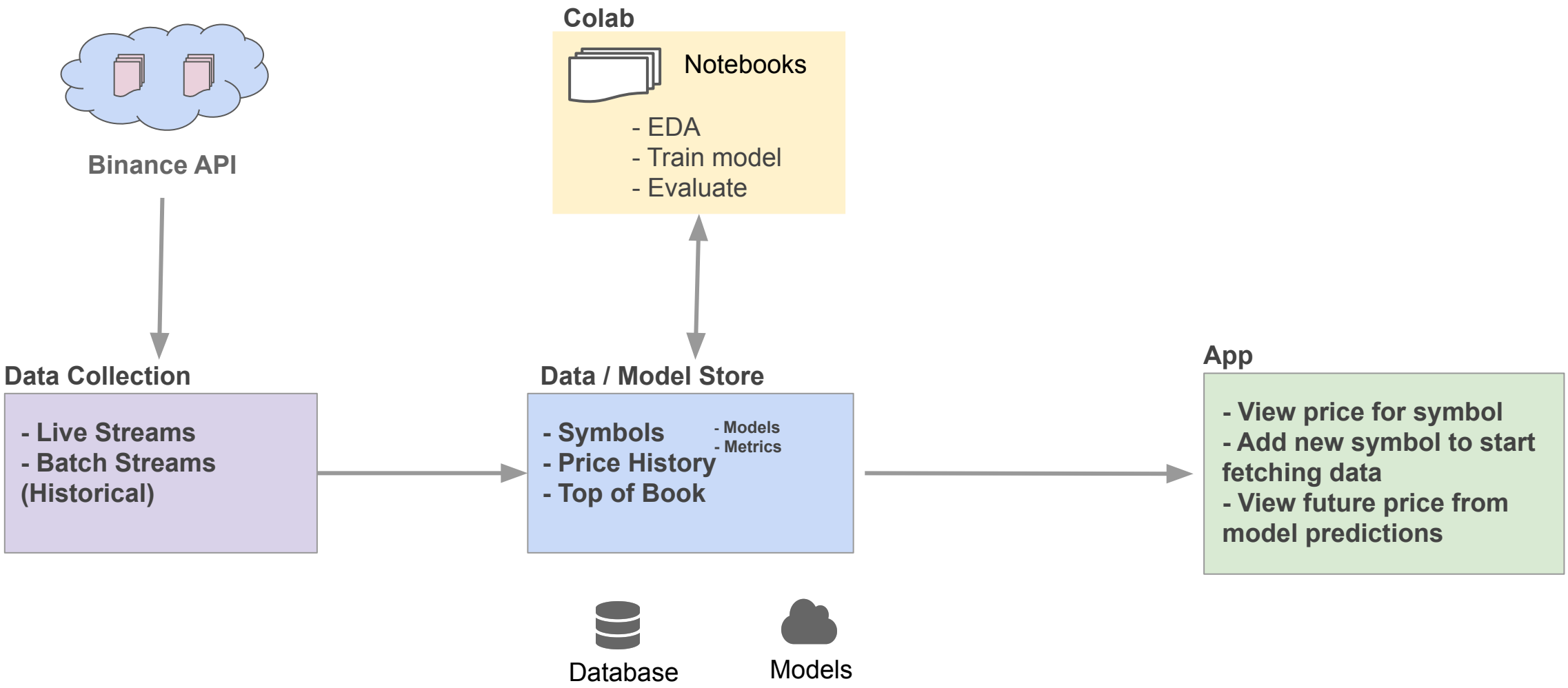
Execution (Code)

- Preprocess the initial data for both time-series and tabular modeling
- Use the best model to make prediction
- Return results to user as a minute-by-minute predictions over 24 hour period
- Track best model

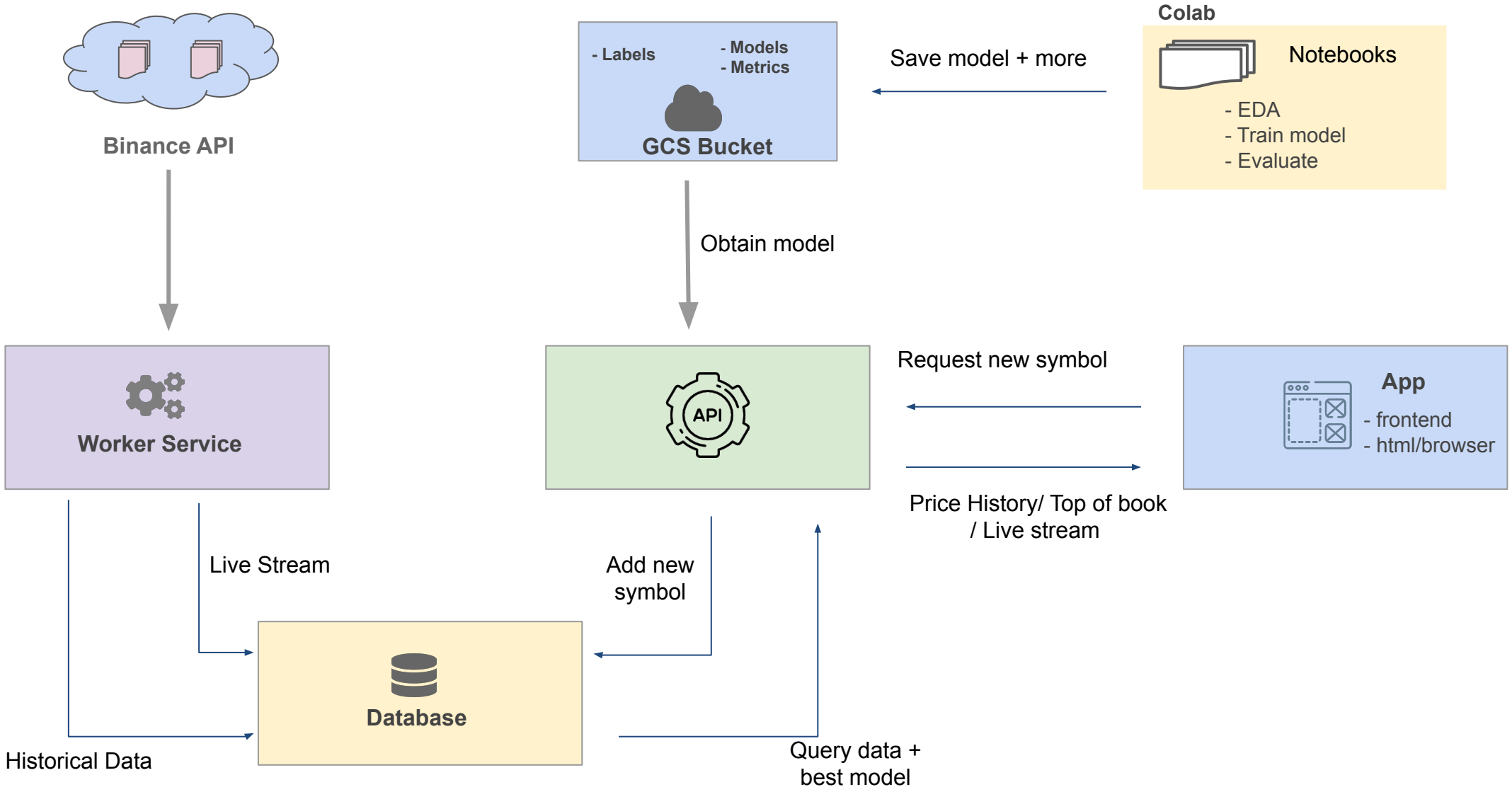
State (Source, Data, Models)

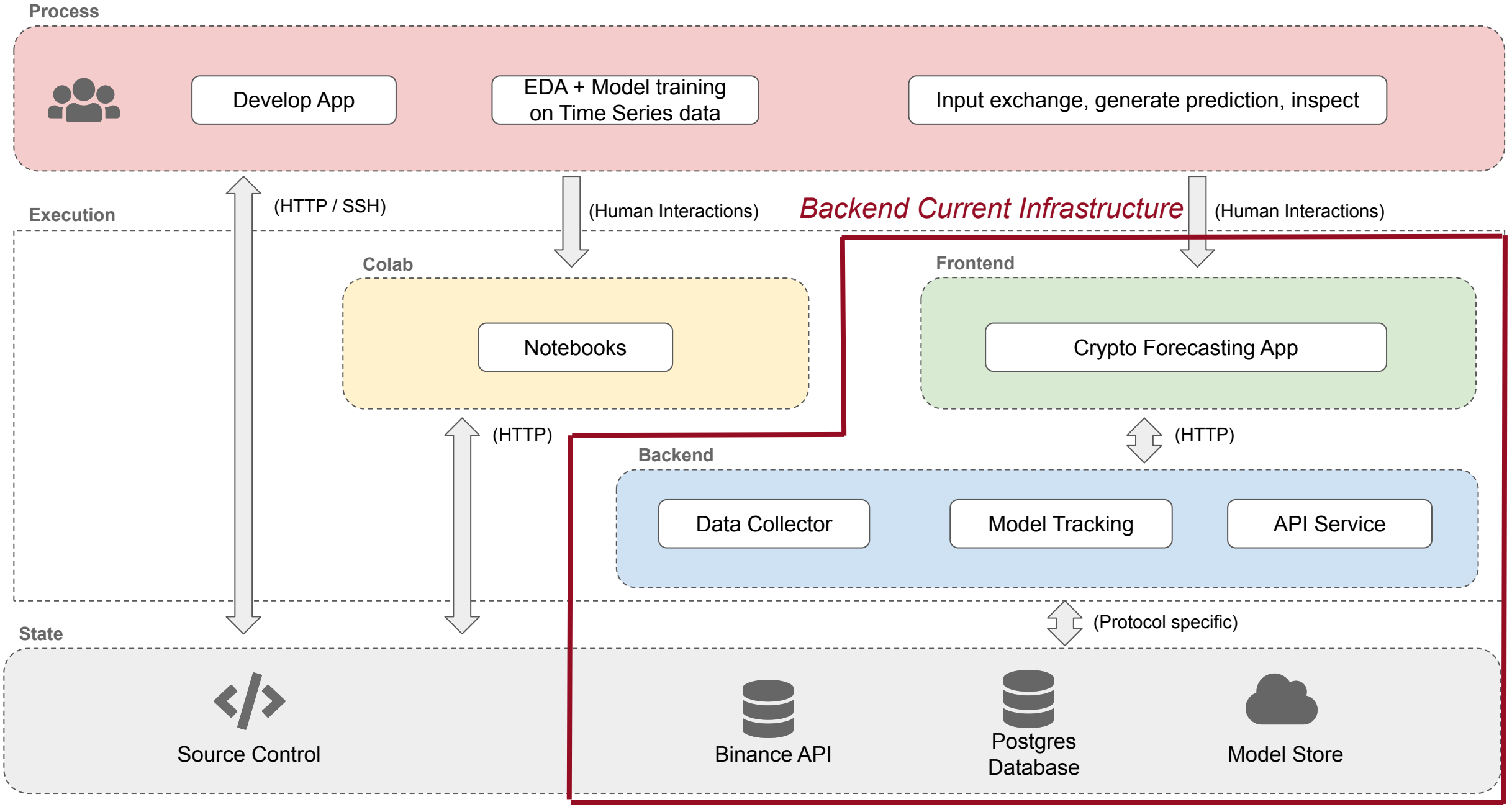
- Save data to a common store and keep updating
- Create tables and save in Postgres database
- Save model weights
- Information on pre processing

Process Flow



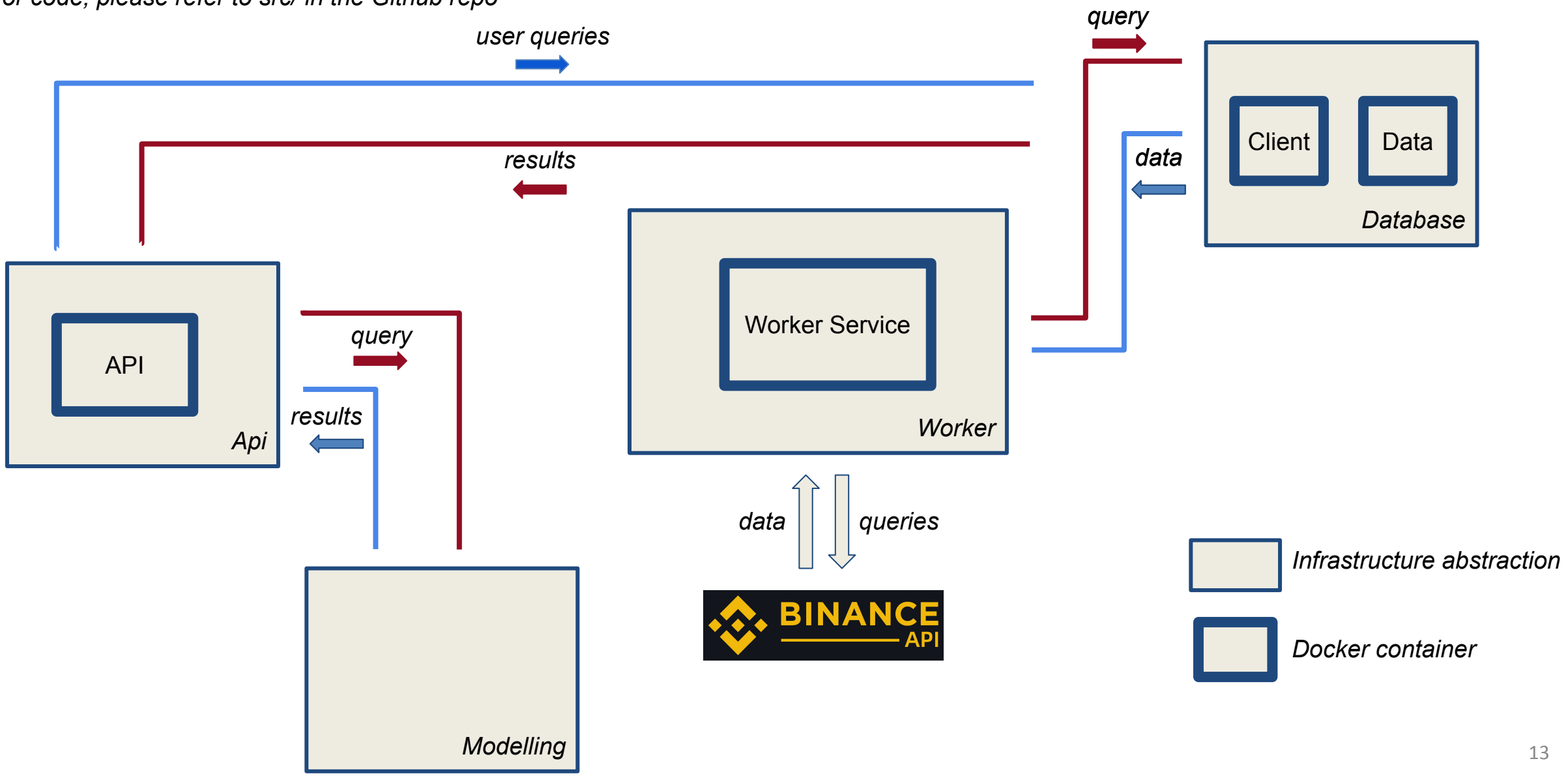
Data Flow


Binance API
- Labels
- Models
- Metrics
GCS Bucket
Save model + more
Colab
Notebooks
- EDA
- Train model
- Evaluate
Obtain model
Worker Service
API
Request new symbol
App
- frontend
- html/browser
Price History/ Top of book
/ Live stream
Live Stream
Add new
symbol
Database
Historical Data
Query data +
best model


Process
Develop App
EDA + Model training on Time Series data
Input exchange, generate prediction, inspect
Execution
(HTTP / SSH)
(Human Interactions)
Backend Current Infrastructure
(Human Interactions)
Colab
Notebooks
Frontend
Crypto Forecasting App
(HTTP)
(HTTP)
Backend
Data Collector
Model Tracking
API Service
(Protocol specific)
State
Source Control
Binance API
Postgres Database
Model Store

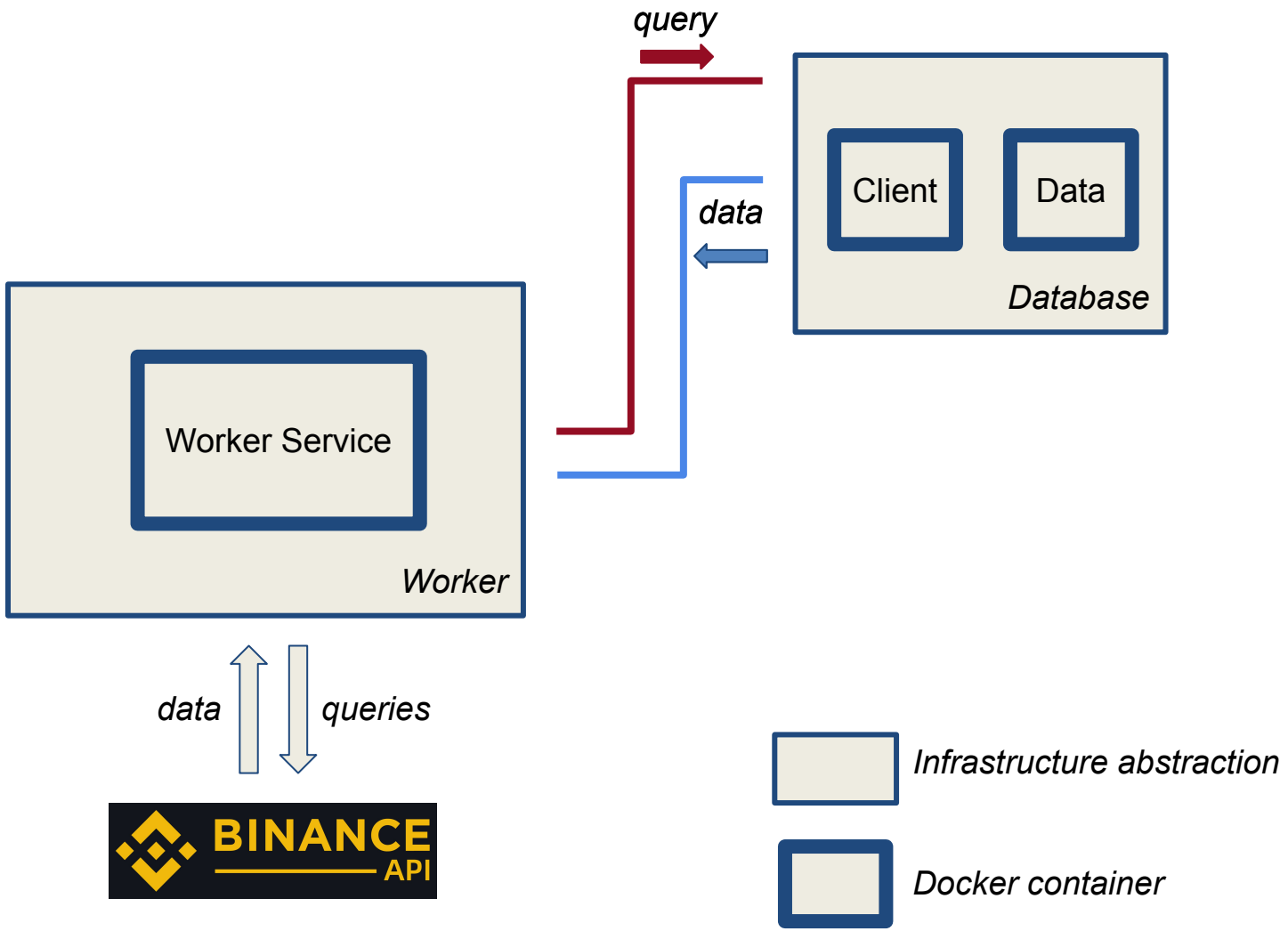
Backend Current Infrastructure

For code, please refer to src/ in the Github repo



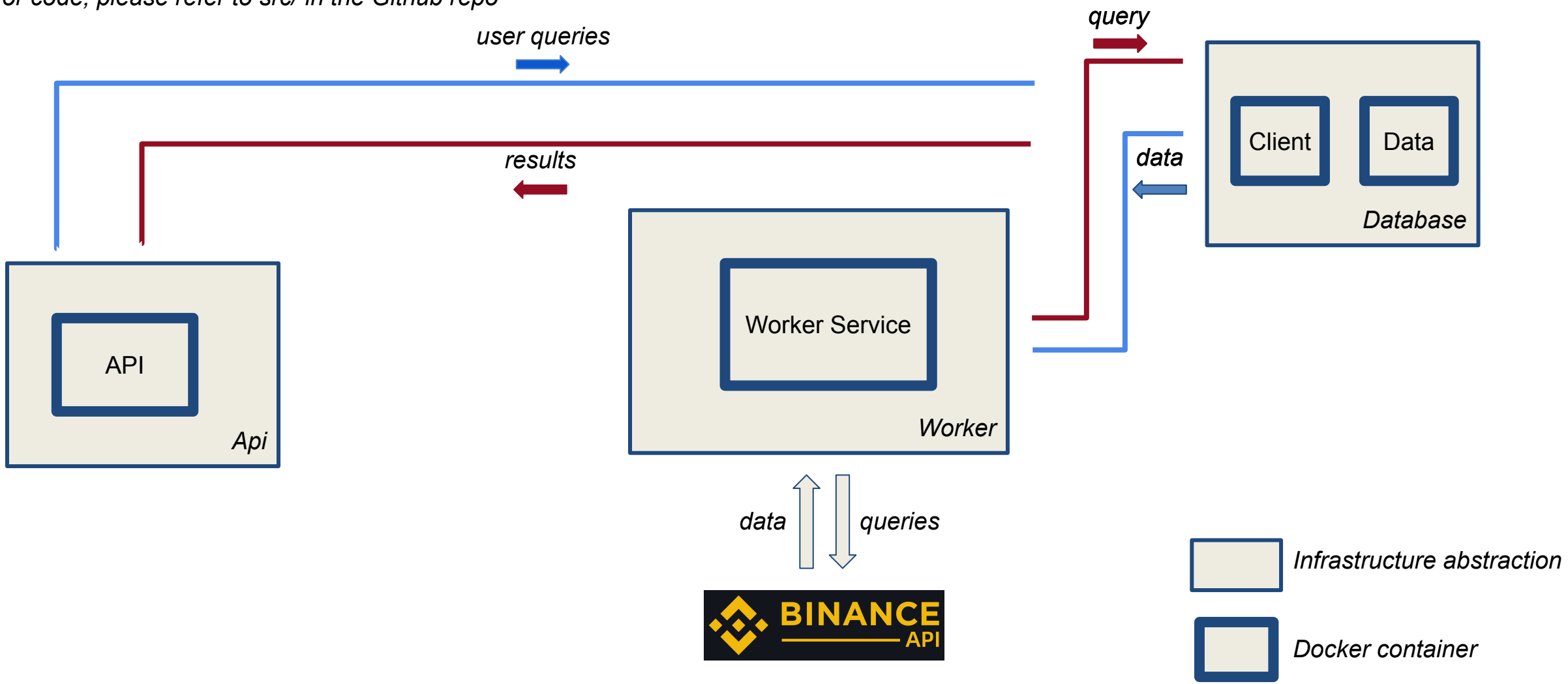
Backend Current Infrastructure: Update data with online streams

For code, please refer to src/ in the Github repo



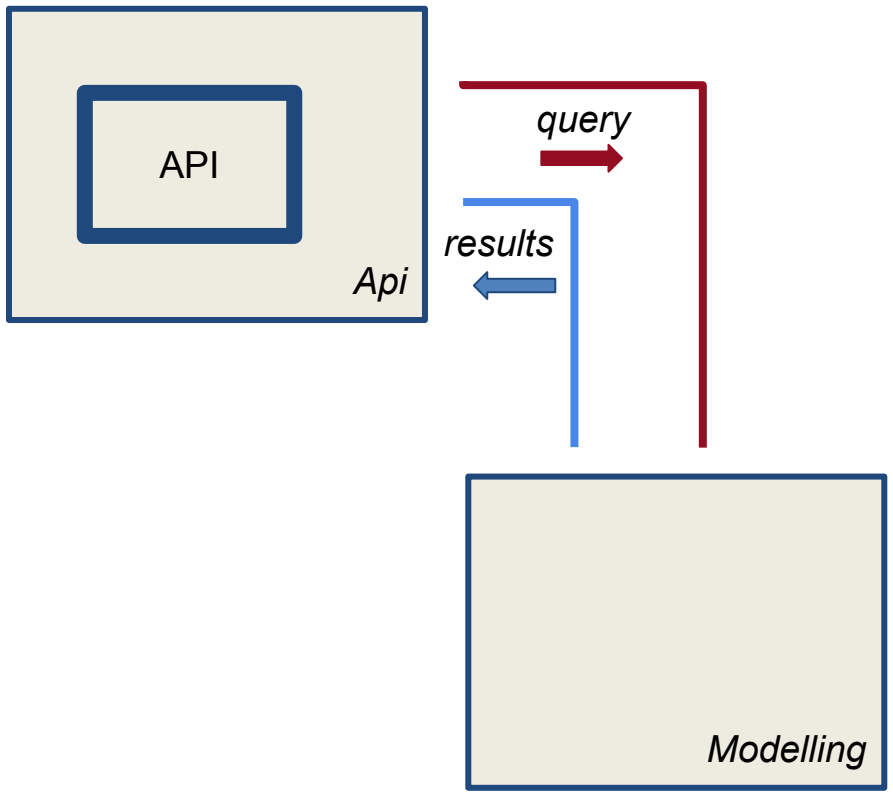
Backend Current Infrastructure: Update data with new pairs

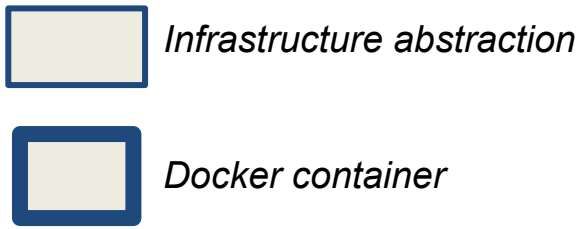
For code, please refer to src/ in the Github repo



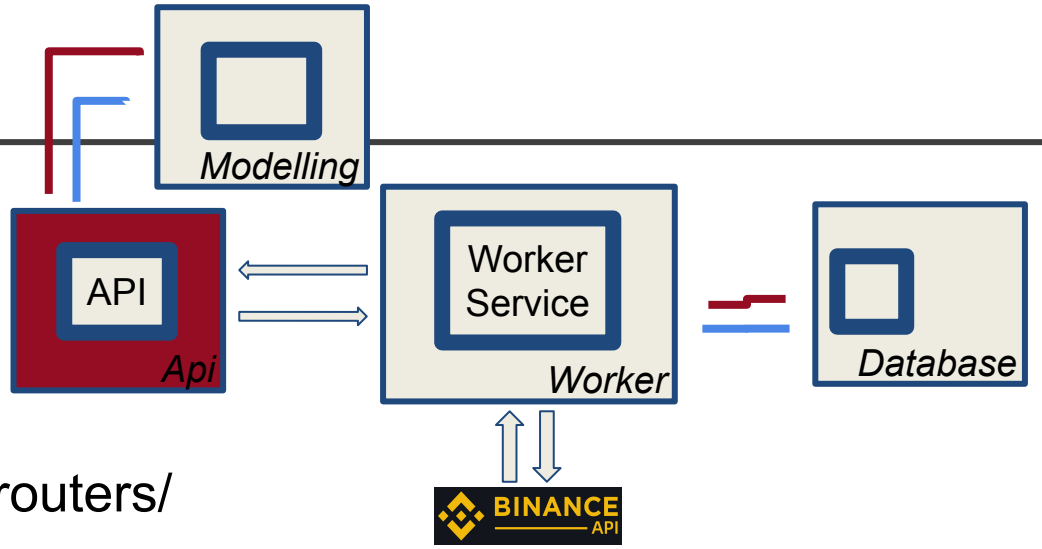
Backend Current Infrastructure: Get model predictions & retraining

For code, please refer to src/ in the Github repo





Current Infrastructure: API

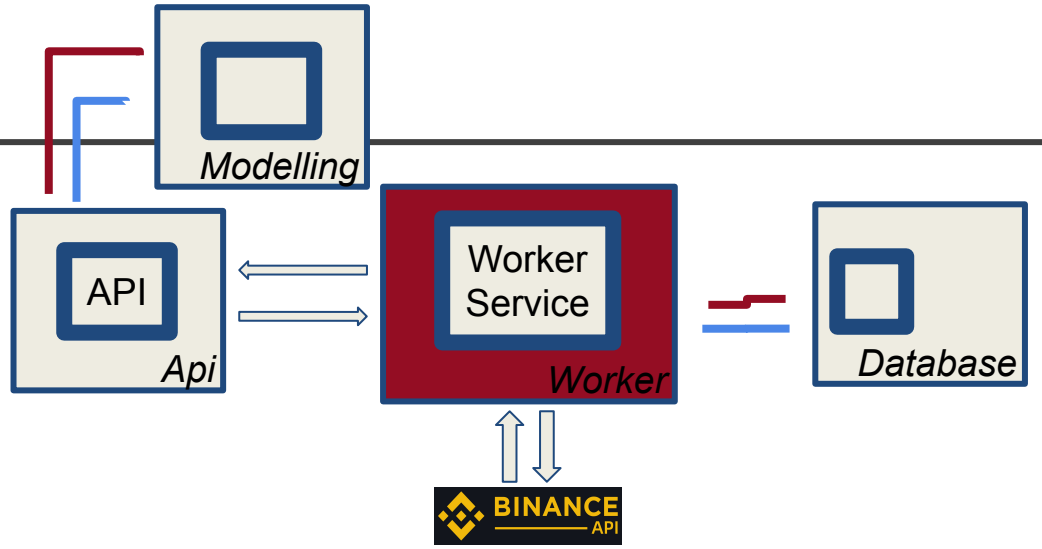


- Use FastAPI implementation of REST API
  - Implementation of tasks in api/service.py and api/routers/

- Use Uvicorn in order to host the server

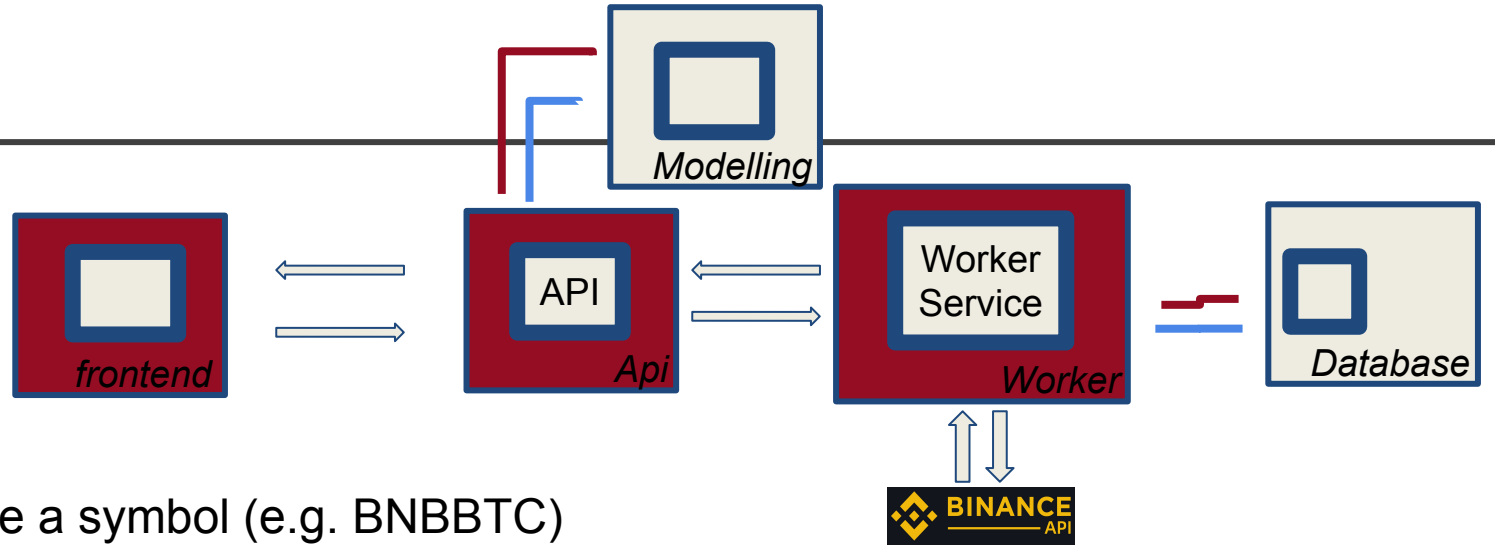
- Receives API call based on user input from the Frontend and either…
  - Adds symbol to *symbols* table if it's a new pair (e.g. BNBBTC)
  - Queries *price_history* table and GCS bucket for model prediction
  - Queries *price_history* and *top_of_book* table for real-time and/or historical data for visualization purposes

Current Infrastructure: Worker



- The worker is responsible for fetching the data from Binance and writing it in the Database
- Two types of data: Historical fetching and online streaming
- MultiProcessing:
  - One process constantly running in order to perform live streaming of the different pairs. This process uses an abstraction of a Threaded WebSocket Manager in order to be able to monitor 1000 streams at the same time
  - Several processes idle when no historical fetching happens, and then when a user queries a symbol that is not yet in the database, these processes will fetch the historical data for these new symbols
- Data Backup: when spinning up the worker, fetches the data from the last updated timestamp to the current timestamp in order for the database to contain data for every timestamp, even when the VM goes down
- Worker monitors the database for api queries

Current Infrastructure: Worker + API +Front-End

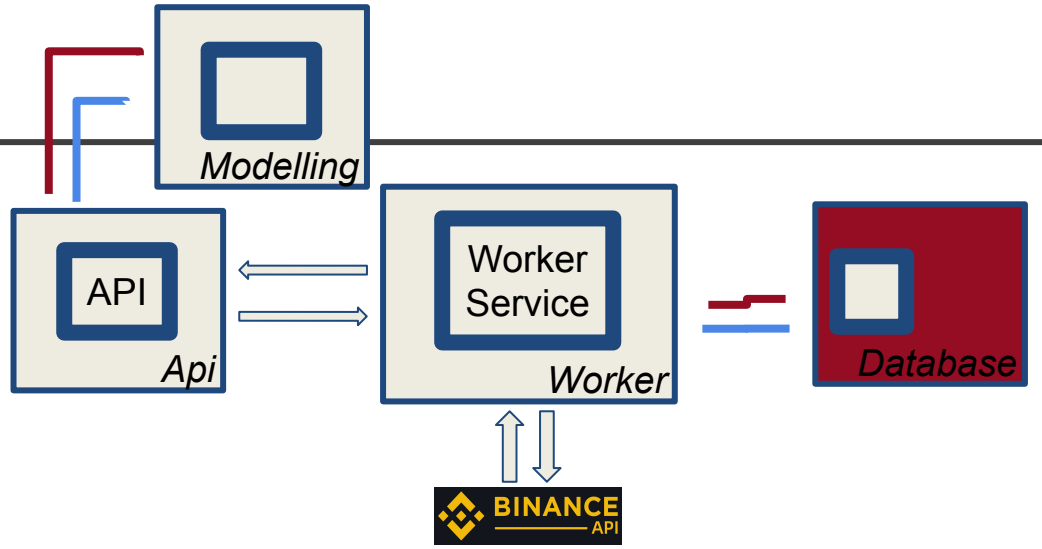


- Scenario: a user inputs in the front-end side a symbol (e.g. BNBBTC)

- 2 possibilities
  - The symbol is already in the database → provide the appropriate data (top of book, candlestick) to the data
  - The symbol is not already in the database → the worker is going to fork a process that will fetch the historical data for this symbol. Once the historical fetching is done, this symbol will be added to the online streams for the 'online process' of worker service and the online data will be fetched for this symbol

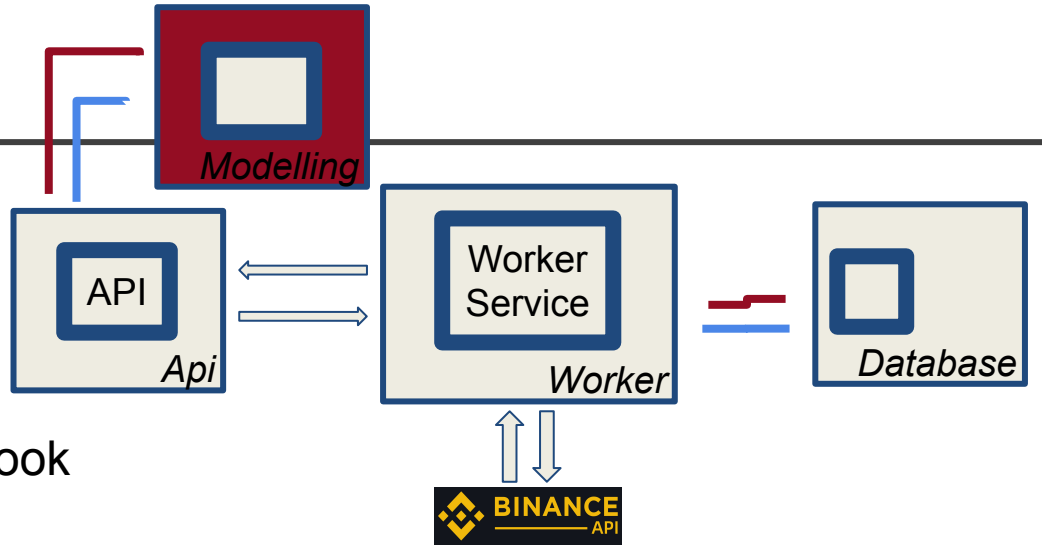
- Communication between API and Worker is being done on the Database side: symbols table that contains all the important logs. The worker service is monitoring the symbols table and creates new processes based on changes on this data table

Current Infrastructure: Database



- Use migration code in order to monitor the different tables
- The Database is a Postgresql database
- 3 major tables:
  - Symbols: contains all the meta-information about the symbols for which we are fetching data. This table also serves the purpose of communication between API and worker - when a user queries a new symbol. It also serves the purpose of being able to get the data we missed while the worker was down
  - Price_history: contains the candle data for the different symbols. Historical data that is available for the different pairs, back until 08/17/2017
  - top_of_book: online data, will enable us to have non aggregated data on the market for visualization and backtesting purposes
- When spinning up the infrastructure for the first time, insert default symbols (the most frequent ones) in the symbols table in order for the worker to start fetching data without needing for a user querying new symbol

Current Infrastructure: Model



- Original creation will happen from a Google Colaboratory Notebook
- Models will be uploaded to the GCS bucket
- From the GCS bucket, the API will query for the best model to use for real-time predictions
- The model will be retrained on new, incoming data on a daily basis
- *Model work is still under development*

Dataset(s)

For code, please refer to src/api_for_data_download in the Github repo

- Dataset(s):
  - Historical data queried from Binance API (dtype: candlesticks)
    - earliest timestamp for the BTC-USDT pair: 2017-08-17 04:00:00
  - Real time data updating from Binance API (dtype: candlesticks) using a websocket

- Dataset(s) size:
  - Number of datasets: 1,612 (one dataset per pair)
  - Size of dataset per pair (~0.3Gb)
  - Total dataset(s) size (~500Gb)

- For the EDA and the initial modelling phases focus on the BTC-USDT pair
- The modelling will be extended to all 1,612 pairs for the final app

Raw Dataset(s) Features

Candle Feature	Description
Open Time	Candle Open Time
Open	Open Price in Quote (Secondary) Asset Units
High	High Price in Quote (Secondary) Asset Units
Low	Low Price in Quote (Secondary) Asset Units
Close	Close Price in Quote (Secondary) Asset Units
Volume	Total Trade Volume in Base (Primary) Asset Units
Close Time	Candle Close Time
Qupte Asset Volume	Total Trade Volume in Quote (Secondary Asset Units
Number of Trades	Total Number of Trades
Taker Buy Base Asset Volume	Taker (Matching Existing Order) Buy Base Asset Volume
Taker Buy Quote Asset Volume	Taker (Mathcing Existing Order) Buy Quote Asset Volume
Ignore	Safe to Ignore

Dataset(s) Quality

Data Types:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2188604 entries, 0 to 2188603
Data columns (total 12 columns):
 #   Column                          Dtype
---  ------                          -----
 0   Open Time                       int64
 1   Open Price                      float64
 2   High price                      float64
 3   Low Price                       float64
 4   Close Price                     float64
 5   Volume Traded                   float64
 6   Close Time                      int64
 7   Quote asset Volume              float64
 8   Number of Trades                int64
 9   Taker buy base asset volume     float64
 10  Taker buy quote asset volume    float64
 11  NA                              float64
dtypes: float64(9), int64(3)
memory usage: 217.1 MB
```

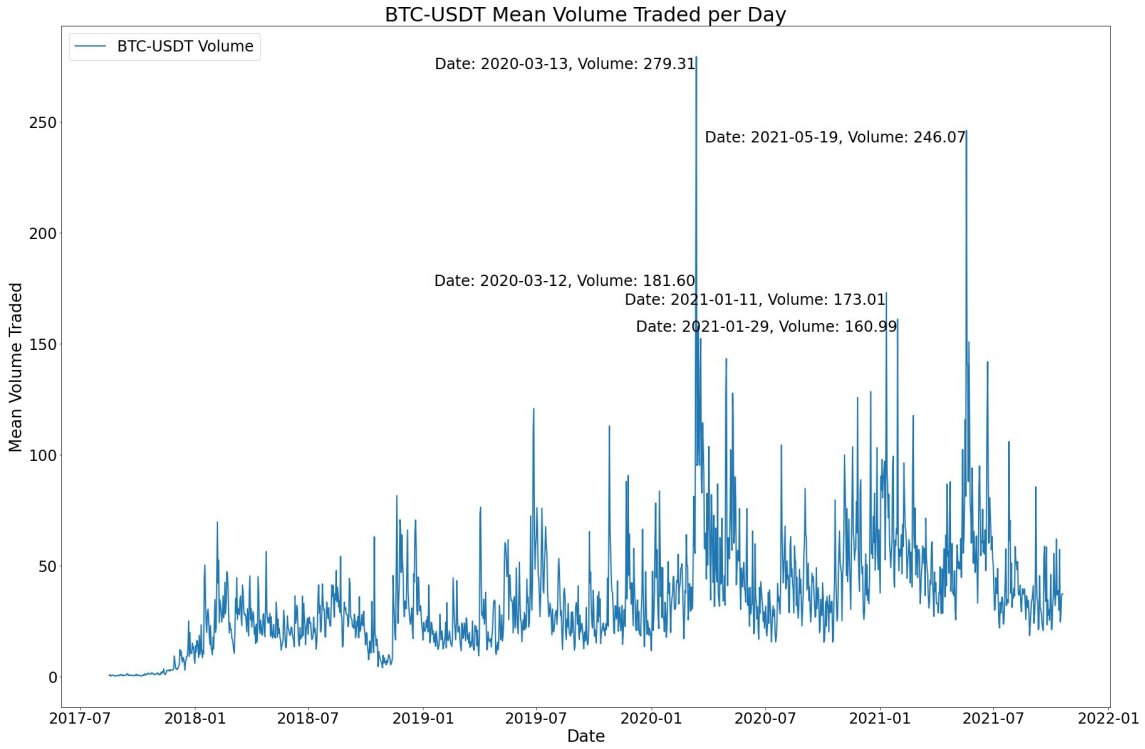
Missing Values:

```
Open Time                       0
Open Price                      0
High price                      0
Low Price                       0
Close Price                     0
Volume Traded                   0
Close Time                      0
Quote asset Volume              0
Number of Trades                0
Taker buy base asset volume     0
Taker buy quote asset volume    0
NA                              0
dtype: int64
```

EDA - Time Series Data

Mean Price per Day

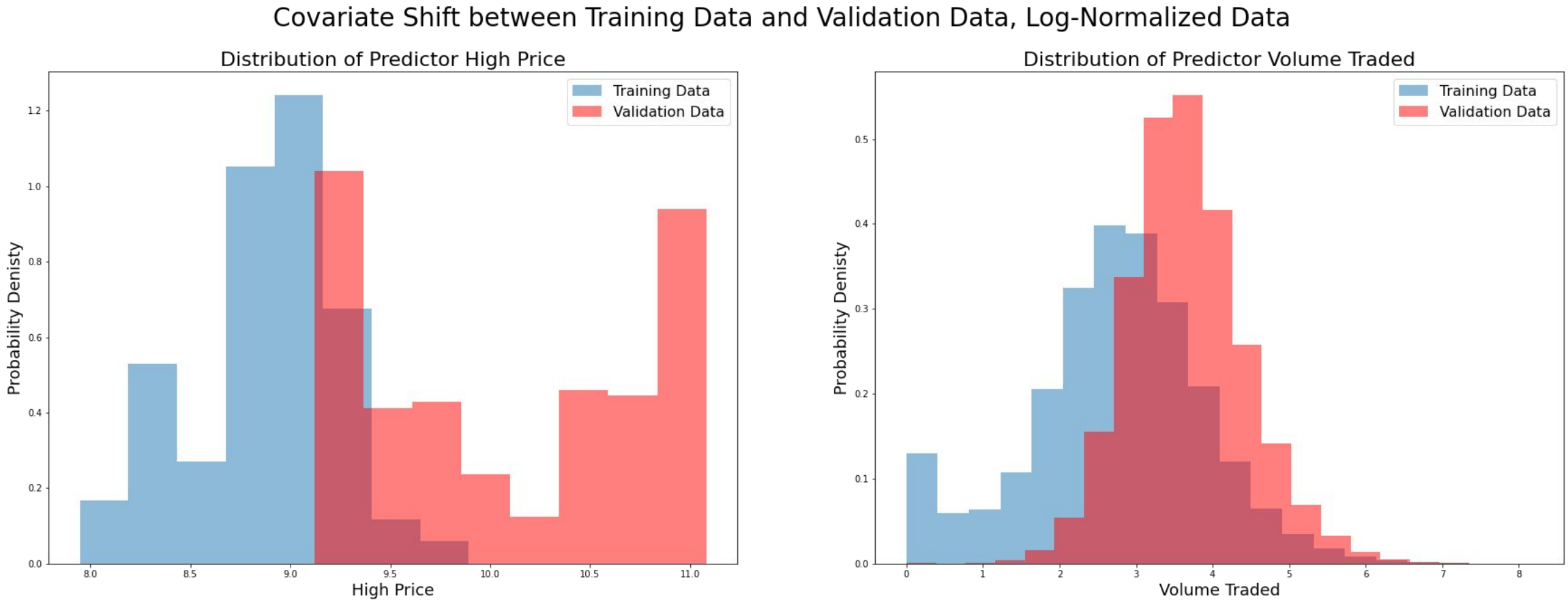
Mean Volume per Day



Key observations:

- The relative value of BTC-USDT increases rapidly due to COVID-19 pandemic
- The trading volume of BTC-USDT is fairly flat, with a few spikes due to COVID-19 pandemic

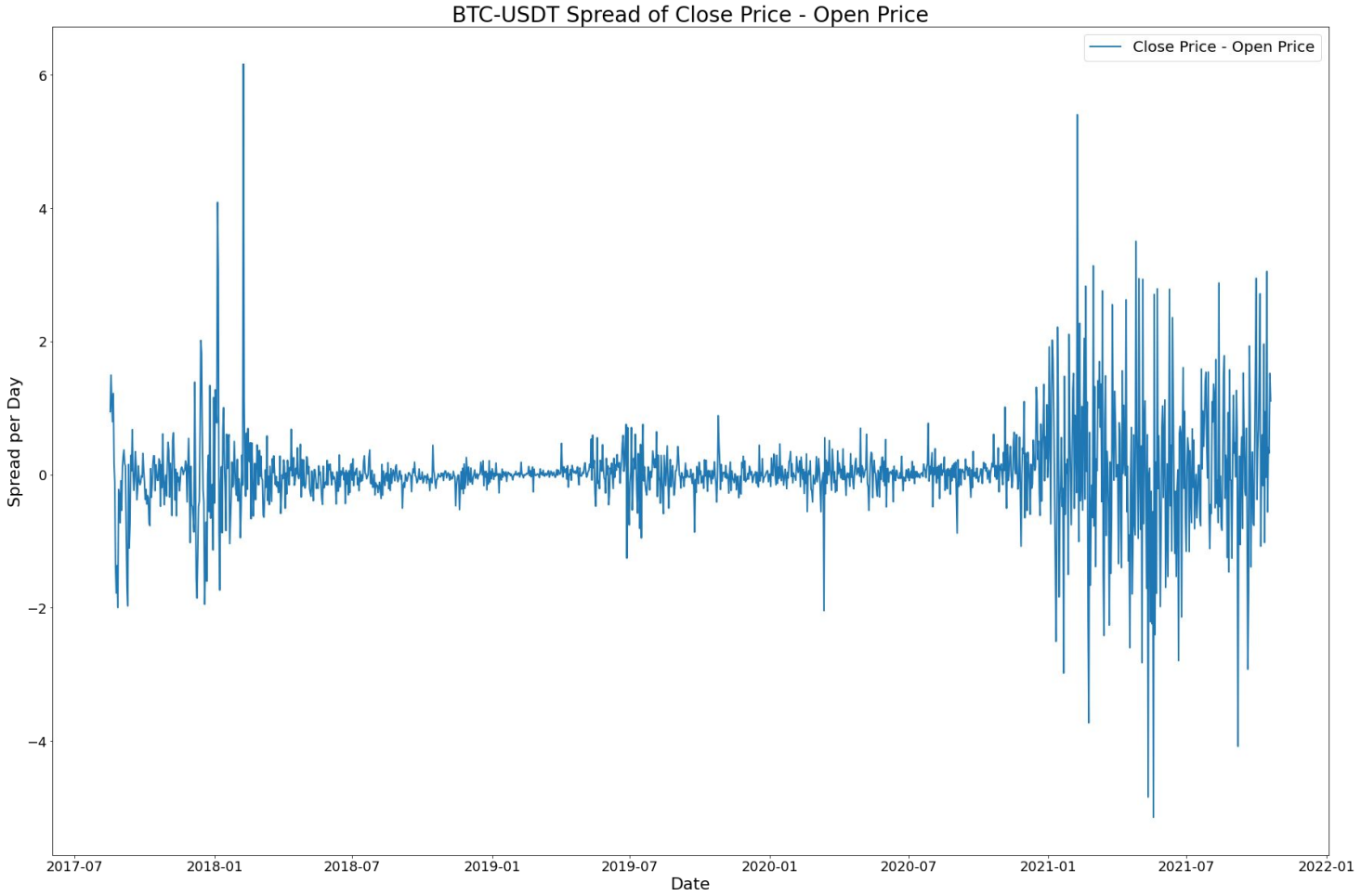
EDA - Out of Distribution Validation Data



Key observations:

- Observable covariate shift in variable `High Price`
- Behavior hinted in the previous slide; validation data post COVID-19

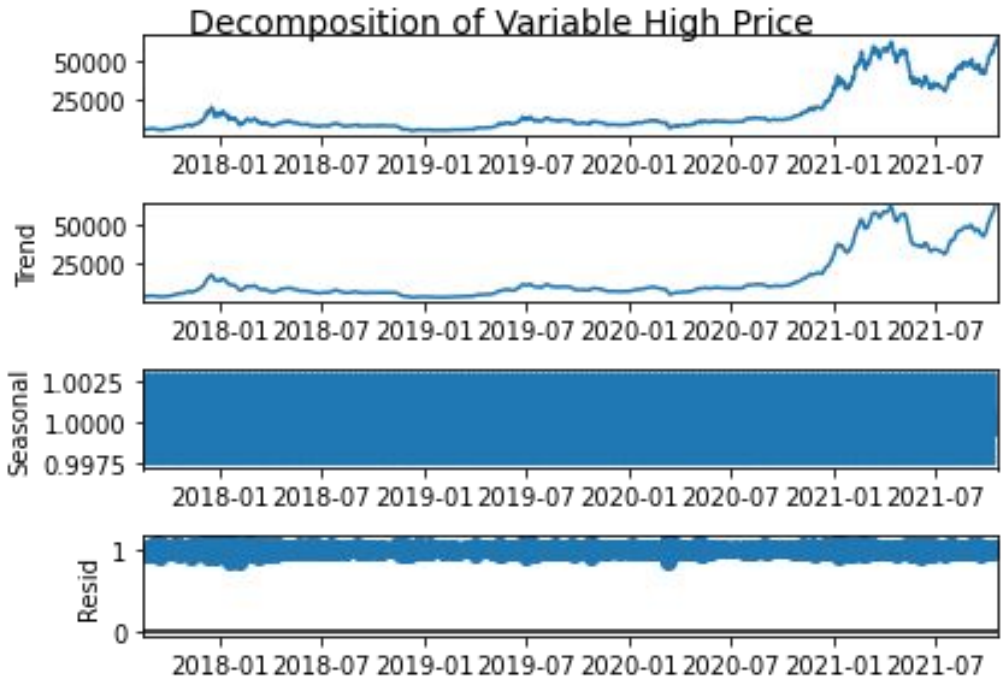
EDA - Response Variable(s)

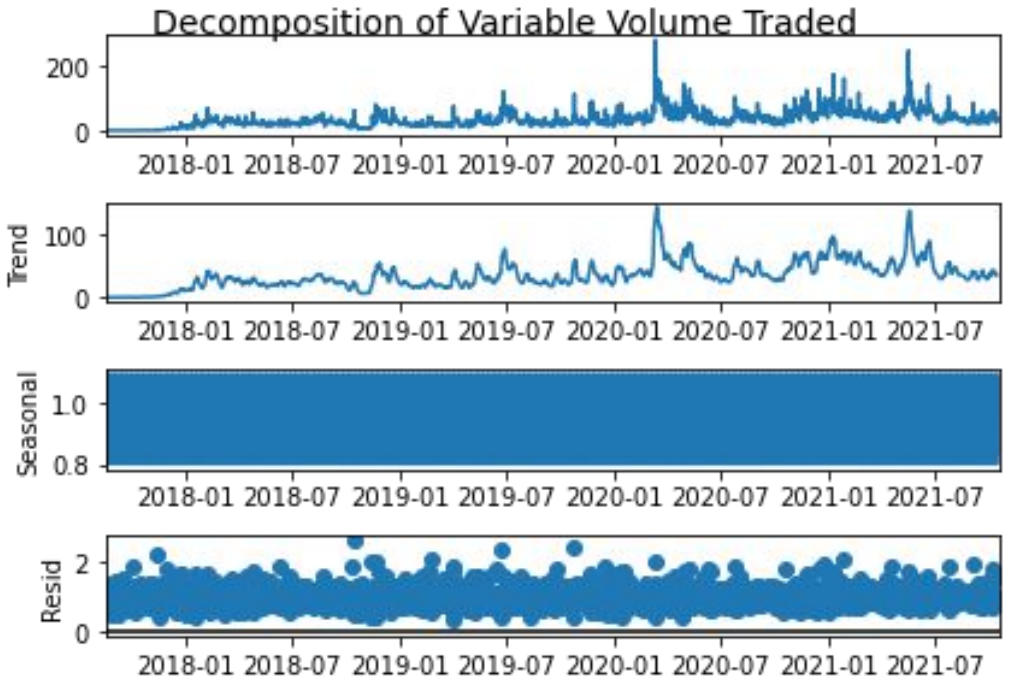


Key observations:

- The spread of `Open Price` - `Close Price` is fairly constant before COVID-19 pandemic
- Since start of COVID-19 pandemic, significantly larger volatility
- This information should be taken into account during modelling/choosing response variable

EDA - Variable Decomposition





Key observations:

- Visible trend for variable `High Price`, as already hinted in previous EDA slides
- No clear trend for variable `Volume Traded`

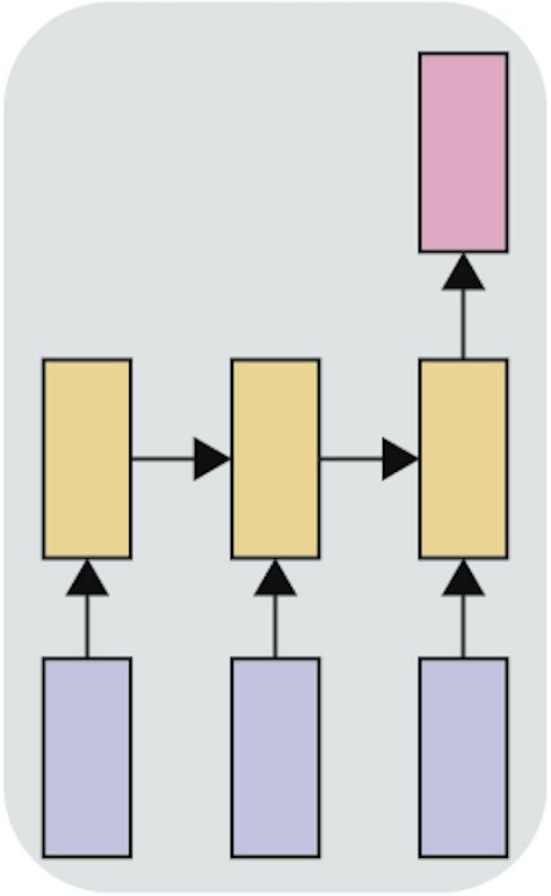
Initial Modeling Decisions

- 90-10 training-validation split
- Standardize the data (based on the whole set)
- For Tensorflow modeling, remove *Close Time, Open Time, NA*
  - Will perform feature engineering utilizing *Close Time, Open Time* (In Future Iterations)
- Metrics: Mean Squared Error (MSE), Mean Absolute Error (MAE)
- Prediction on *Close Price*

Modeling Approaches

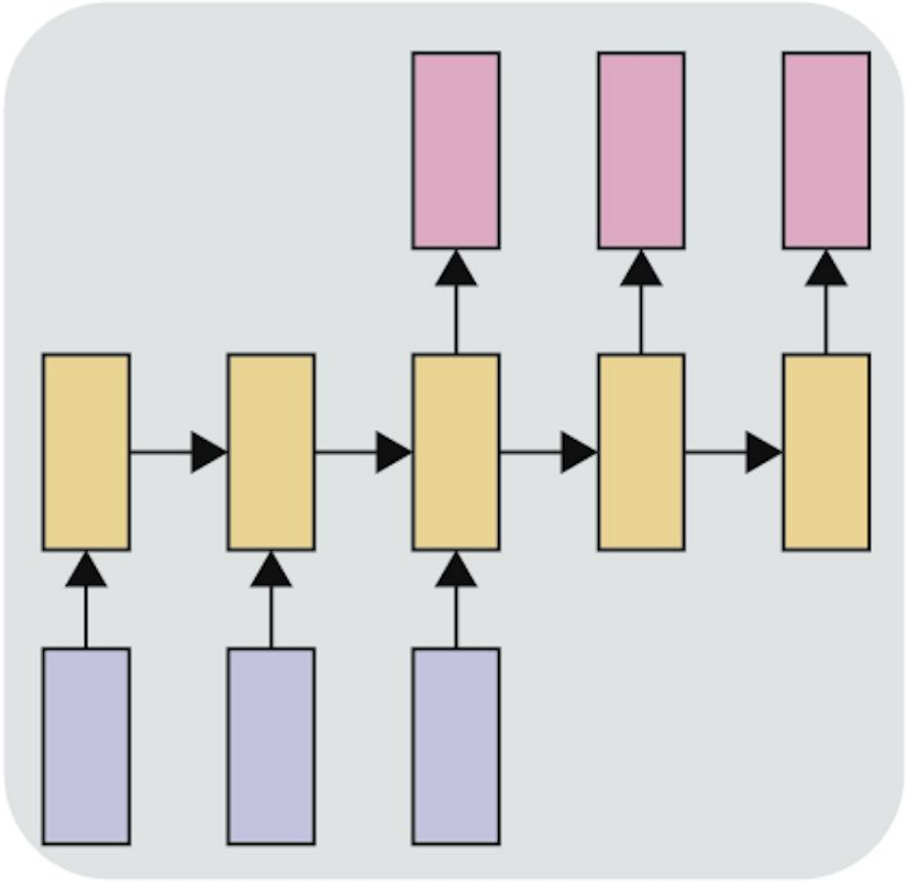
Current: Multi-input, Single-output

given the current input, the model infers the value of the pre-defined pair, one time step in the future



Current: Multi-input, Multi-output

given the current input, the model infers the value of the pre-defined pair, X time steps in the future



Credit

Models

- **Baseline - Persistent Model**
  - For Multi-input, Single-output predicts the Close Price of the next time step to be the same as the Close Price of the current time step
  - For Multi-input, Multi-output predicts the Close Price of the next X time steps to be the same as the Close Price of the current time step
- **Current Model - LSTM on Standardized Raw Features**
  - For Multi-input, Single-output predicts the Close Price of the next time step based on the input features of the previous X time steps
  - For Multi-input, Single-output predicts the Close Price of the X next time steps based on the input features of the previous X time steps
- **Currently In Progress - LSTM on Engineered Features**
  - Feature Engineering on input data: Log-transformed, Standardized, Time-based features, Statistical features, Domain knowledge - based features)
  - Feature Engineering on output data: Transformed the output feature (Close Price) to “Close Price - Baseline Prediction”

Models - Preliminary Results

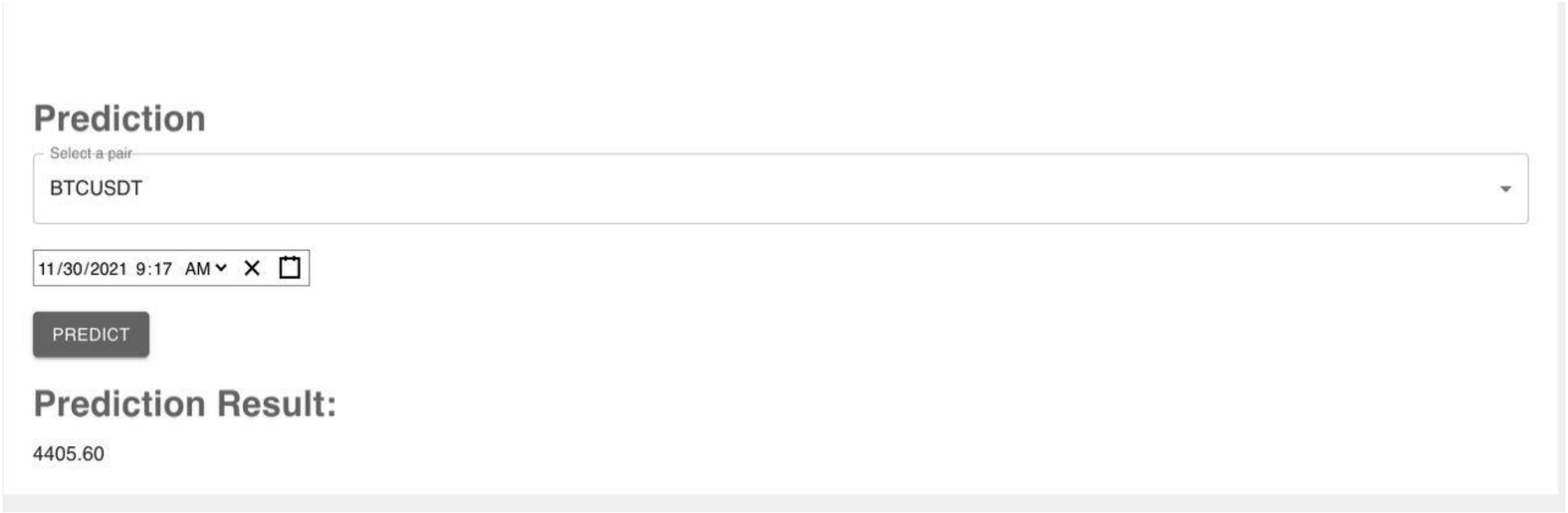
Models	Train MSE	Train MAE	Validation MSE	Validation MAE
Baseline - Single Output (SO)	2.8378e-6	0.0007	9.2235e-6	0.0020
Baseline - Multi Output (MO)	/	/	/	/
LSTM - Standardized Raw Features (SO)	1.0474e-5	0.0017	9.1504e-5	0.0044
LSTM - Standardized Raw Features (MO)	5.4911e-5	0.0035	1.6730e-4	0.0084
LSTM - Engineered Features (SO)	/	/	/	/
LSTM - Engineered Features (MO)	/	/	/	/

Frontend - React

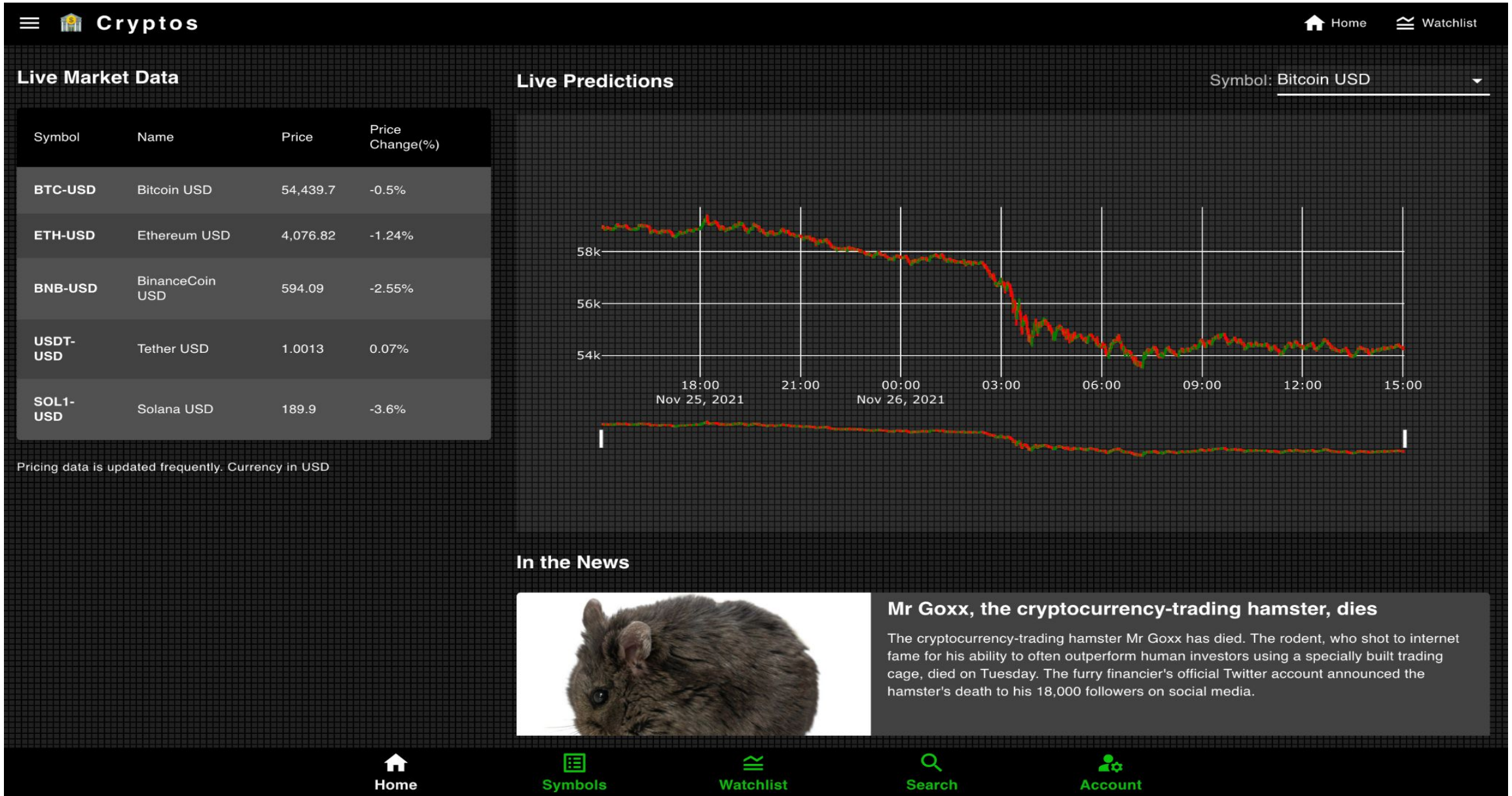
- We used React javascript library to build our frontend UI
- Our UI consists of the following **components**
  - Time series plot with range slider *(plotly.js)*
  - Selection of pair and time for prediction *(material UI)*
  - Prediction results
- **API calls:**
  - Frontend will send selected ticker pair and date as a dictionary to backend API service for model prediction.
  - Frontend will make API calls and get back response data from console log
  - Response data returned in JSON dictionary format for predictions and plot of historical time series trend

Current Frontend

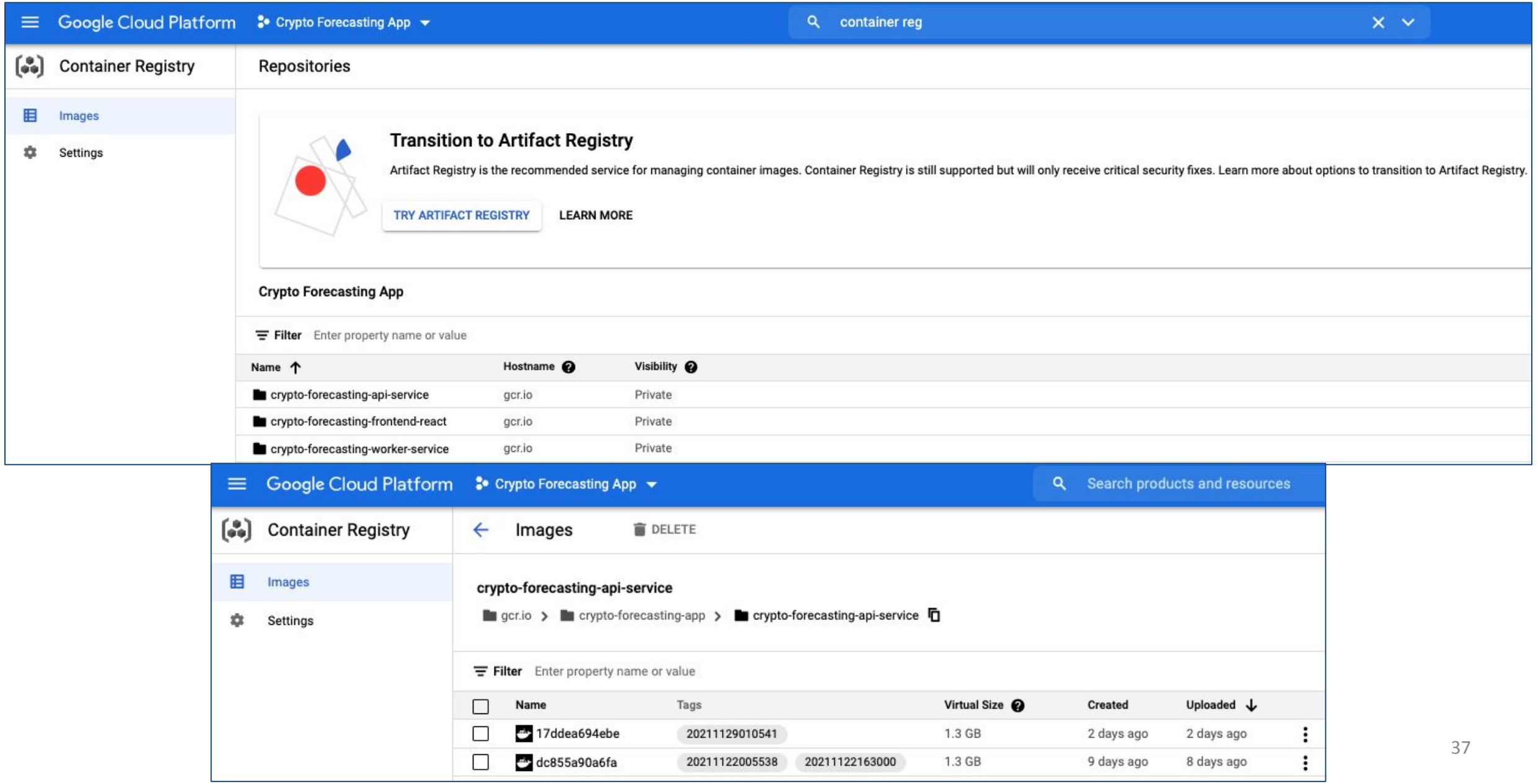
Current Frontend



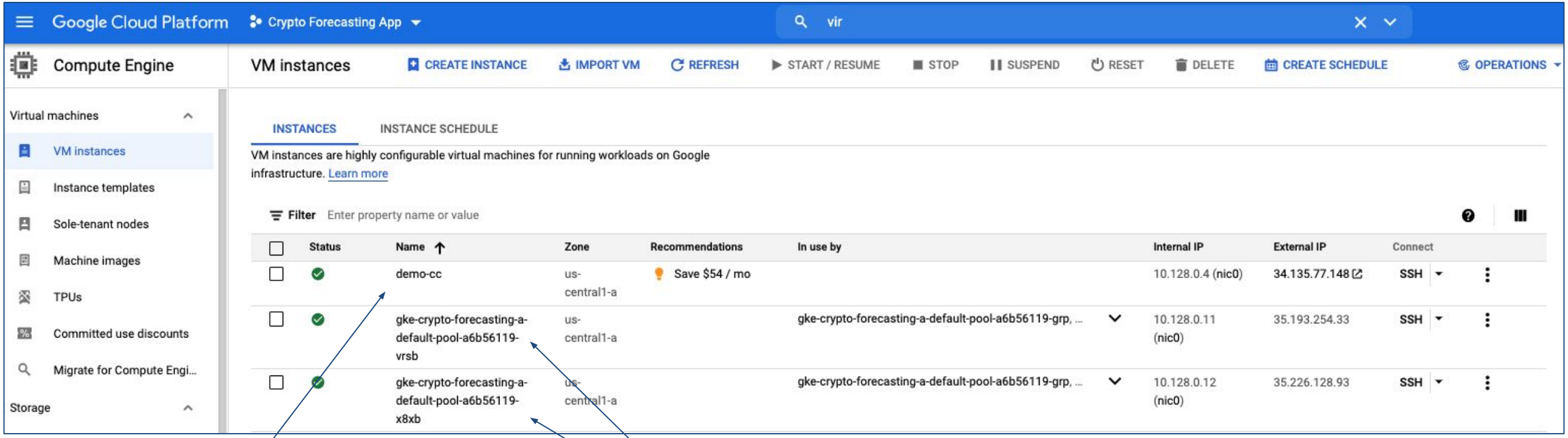
Future Frontend



Deployment - Container Registry



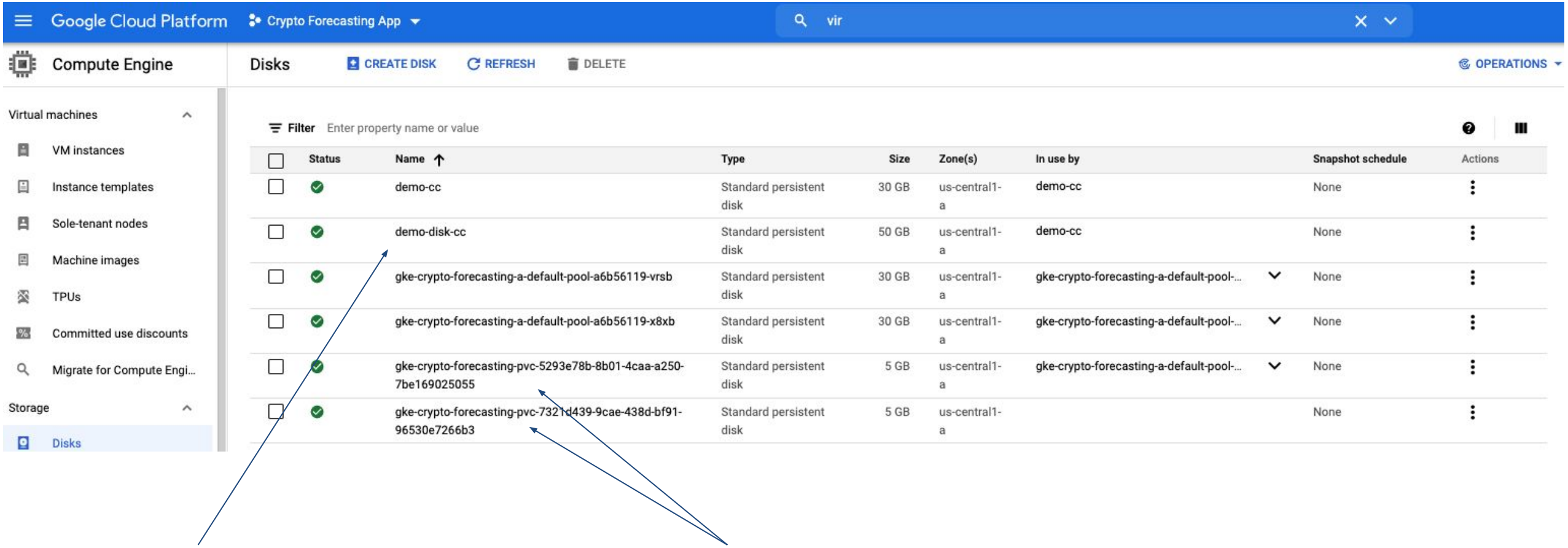
Deployment - Compute Engine



Ansible Deployment

Kubernetes Deployment

Deployment - Persistent Disk (Saving Postgres Database)



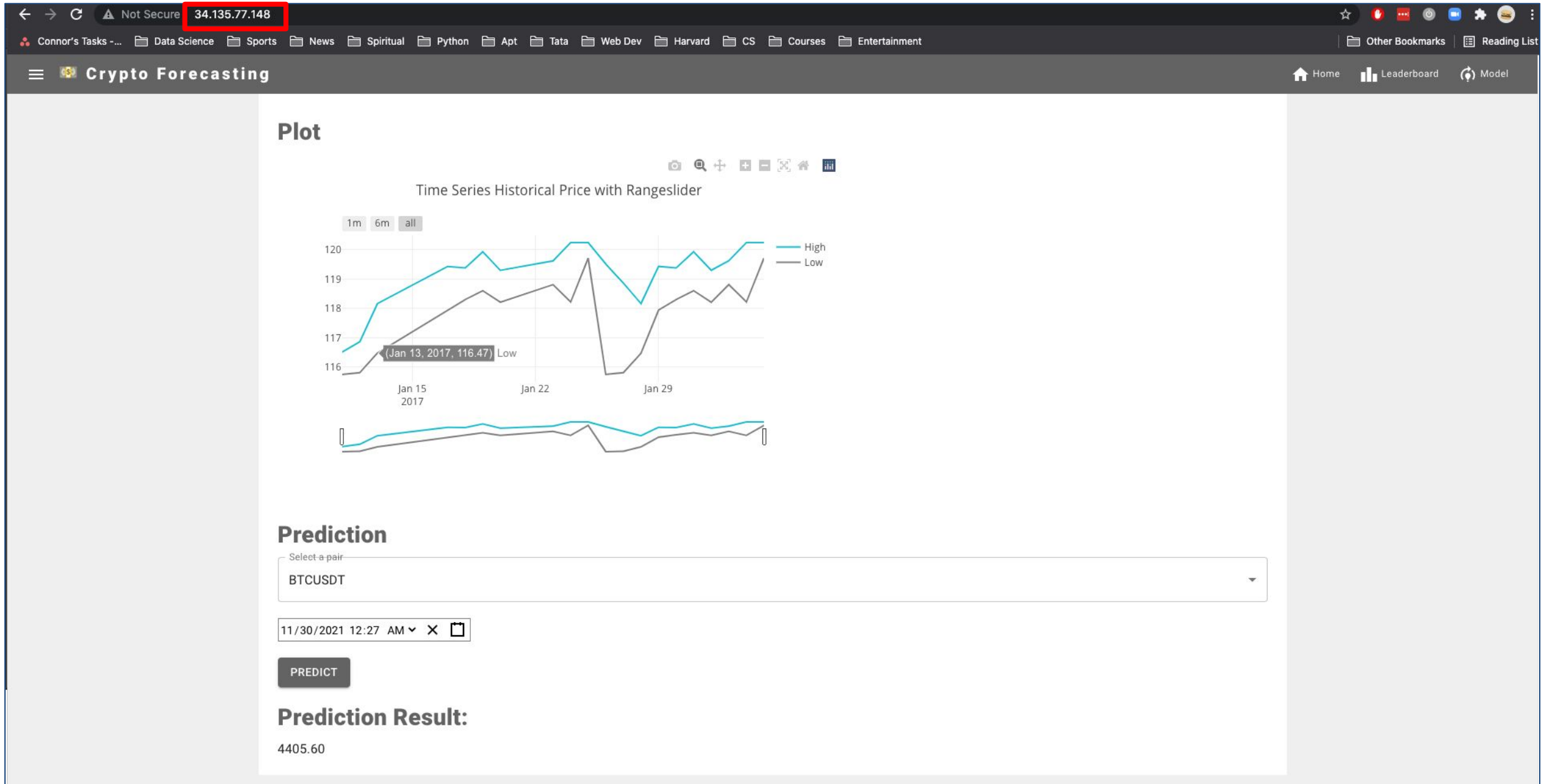
Ansible Deployment

Kubernetes Deployment

Deployment - Ansible Virtual Machine SSH

```
connorcapitolo_g_harvard_edu@demo-cc:~$ sudo docker image ls
REPOSITORY                                                      TAG              IMAGE ID       CREATED        SIZE
gcr.io/crypto-forecasting-app/crypto-forecasting-worker-service   20211129010541   7998cf254ecb   37 hours ago   650MB
gcr.io/crypto-forecasting-app/crypto-forecasting-api-service      20211129010541   e64b373aed92   37 hours ago   2.63GB
gcr.io/crypto-forecasting-app/crypto-forecasting-frontend-react   20211129010541   edfc06298332   37 hours ago   162MB
postgres                                                        latest           577410342f45   12 days ago    374MB
nginx                                                           stable           aedf7f31bdab   13 days ago    141MB
connorcapitolo_g_harvard_edu@demo-cc:~$ sudo docker container ls
CONTAINER ID   IMAGE                                                                COMMAND                  CREATED        STATUS        PORTS                                   NAMES
7d2992b474fe   nginx:stable                                                         "/docker-entrypoint.…"   37 hours ago   Up 37 hours   0.0.0.0:80->80/tcp, 0.0.0.0:443->443/tcp   nginx
7d5bc228417a   gcr.io/crypto-forecasting-app/crypto-forecasting-worker-service:20211129010541   "/bin/bash ./docker-…"   37 hours ago   Up 37 hours                                           worker-service
78cf76469a44   gcr.io/crypto-forecasting-app/crypto-forecasting-api-service:20211129010541      "/bin/bash ./docker-…"   37 hours ago   Up 37 hours   0.0.0.0:9600->9000/tcp                  api-service
90b0a0ffbd4e   postgres:latest                                                      "docker-entrypoint.s…"   37 hours ago   Up 37 hours   0.0.0.0:5432->5432/tcp                  postgres
8ab2619b5094   gcr.io/crypto-forecasting-app/crypto-forecasting-frontend-react:20211129010541   "/docker-entrypoint.…"   37 hours ago   Up 37 hours   0.0.0.0:3000->80/tcp                    frontend
```

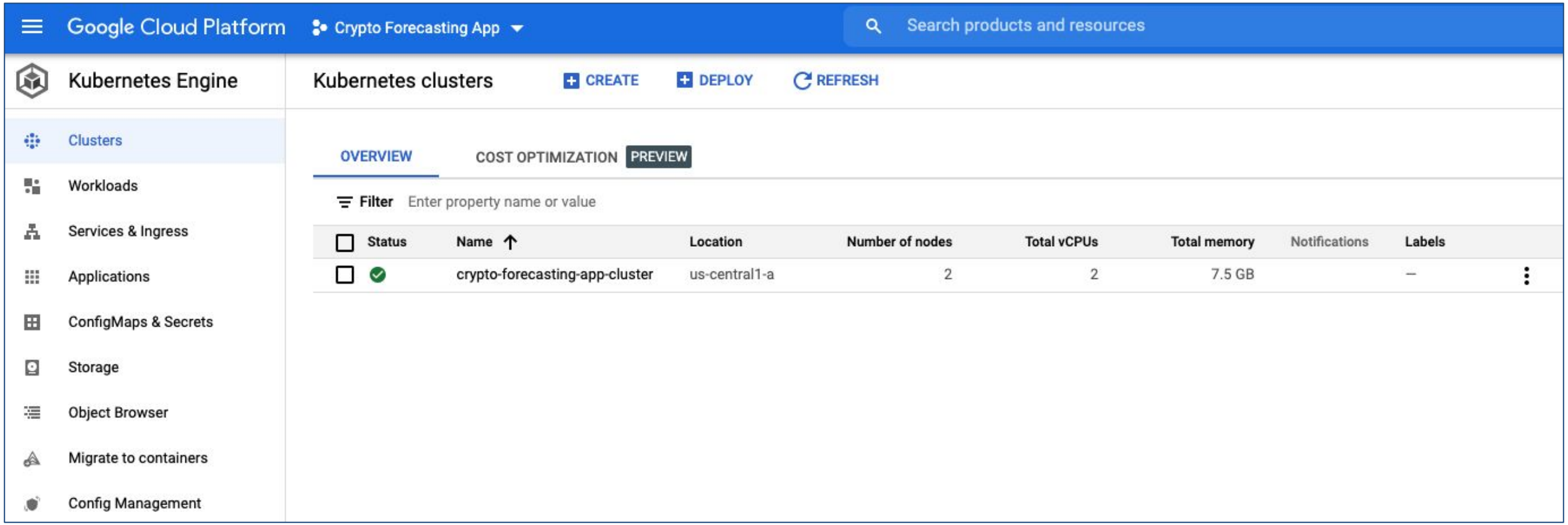
Deployment - Ansible Crypto Forecasting App



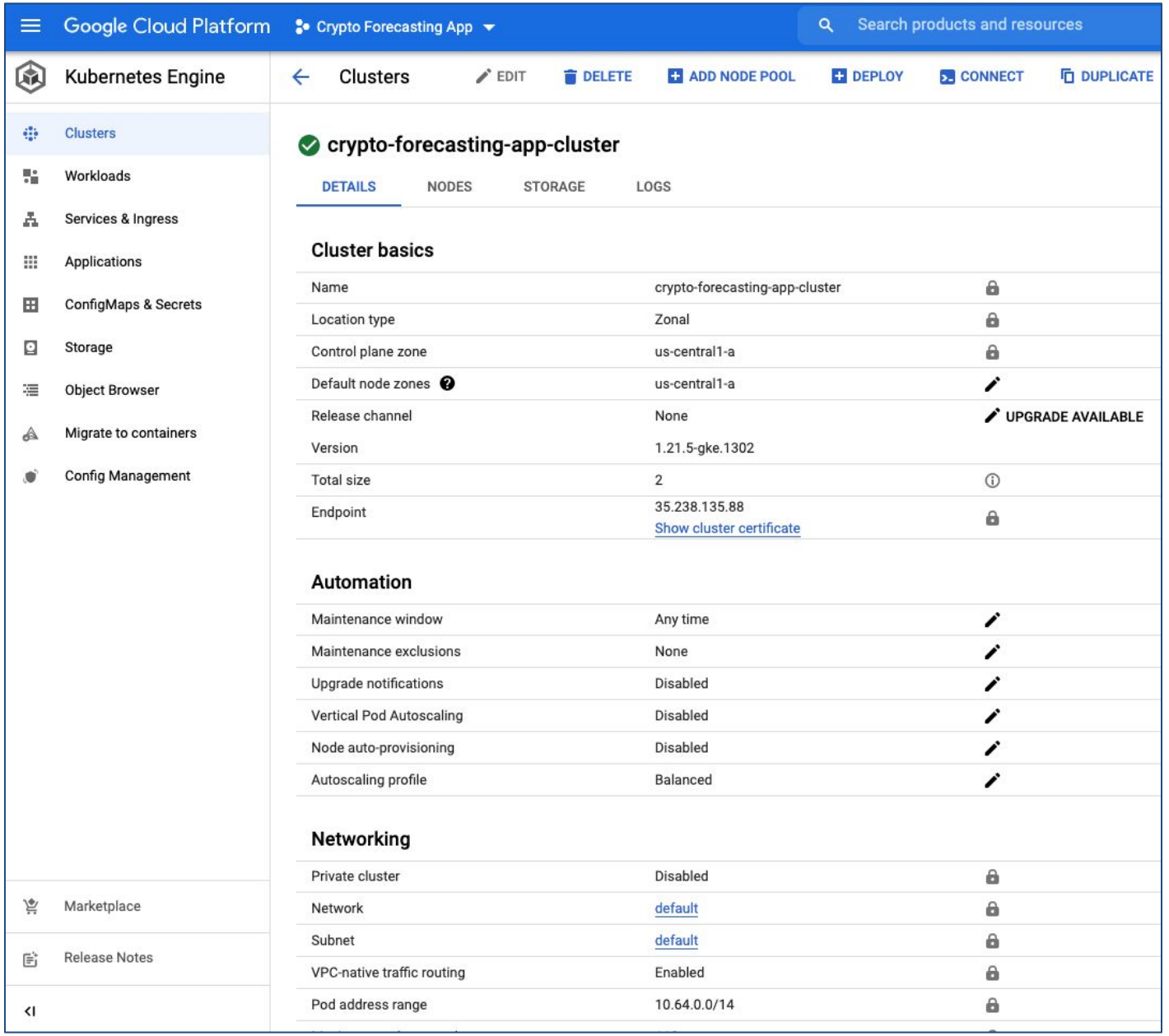
Deployment - Kubernetes Script (Partial View)

```
root@425fa92ea59f:/app# ansible-playbook deploy-k8s-cluster.yml -i inventory.yml --extra-vars cluster_state=present
[DEPRECATION WARNING]: community.kubernetes.helm_repository has been deprecated. The community.kubernetes collection is being renamed to kube
rnetes.core. Please update your FQCNs to
kubernetes.core instead. This feature will be removed from community.kubernetes in version 3.0.0. Deprecation warnings can be disabled by set
ting deprecation_warnings=False in ansible.cfg.
[DEPRECATION WARNING]: community.kubernetes.helm has been deprecated. The community.kubernetes collection is being renamed to kubernetes.core
. Please update your FQCNs to kubernetes.core
instead. This feature will be removed from community.kubernetes in version 3.0.0. Deprecation warnings can be disabled by setting deprecation
_warnings=False in ansible.cfg.

PLAY [Create Kubernetes Cluster and deploy multiple containers] ****************************************************************************
*********************************************

TASK [Create a GKE cluster] ****************************************************************************************************************
*********************************************
changed: [localhost]

TASK [Create a Node Pool] ******************************************************************************************************************
*********************************************
changed: [localhost]

TASK [Connect to cluster (update kubeconfig)] **********************************************************************************************
*********************************************
changed: [localhost]

TASK [Create Namespace] ********************************************************************************************************************
*********************************************
changed: [localhost]

TASK [Add nginx-ingress helm repo] *********************************************************************************************************
*********************************************
ok: [localhost]

TASK [Install nginx-ingress] ***************************************************************************************************************
*********************************************
changed: [localhost]

TASK [Copy docker tag file] ****************************************************************************************************************
*********************************************
ok: [localhost]
```

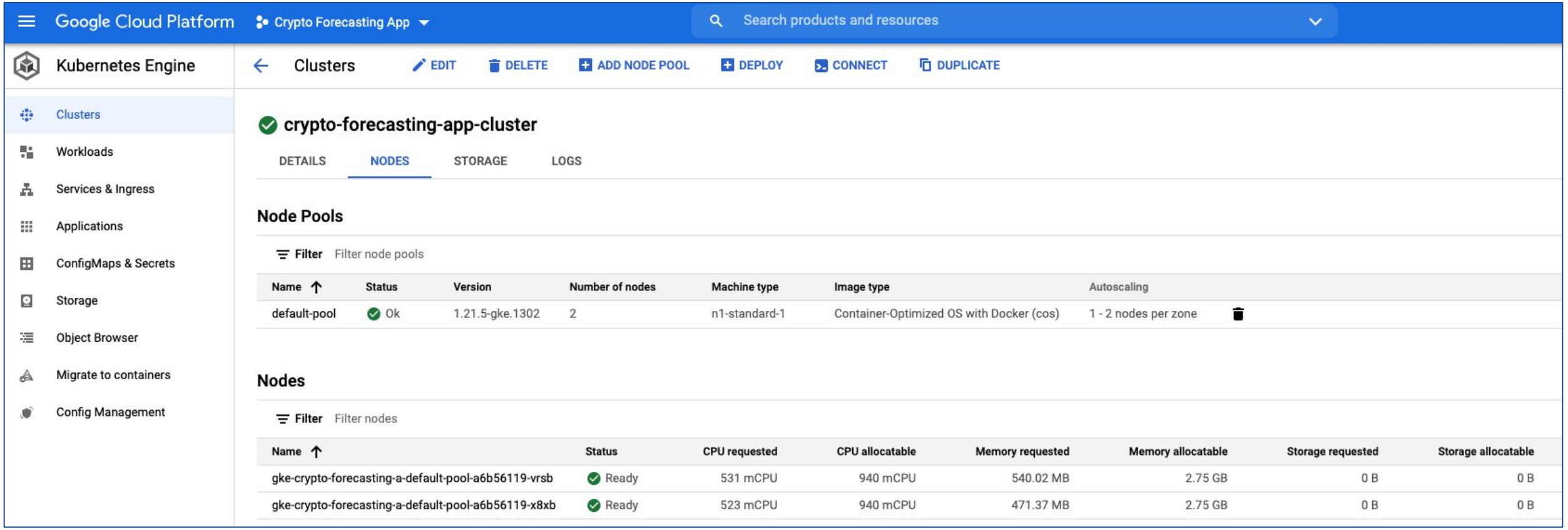
Deployment - Kubernetes Engine Homepage



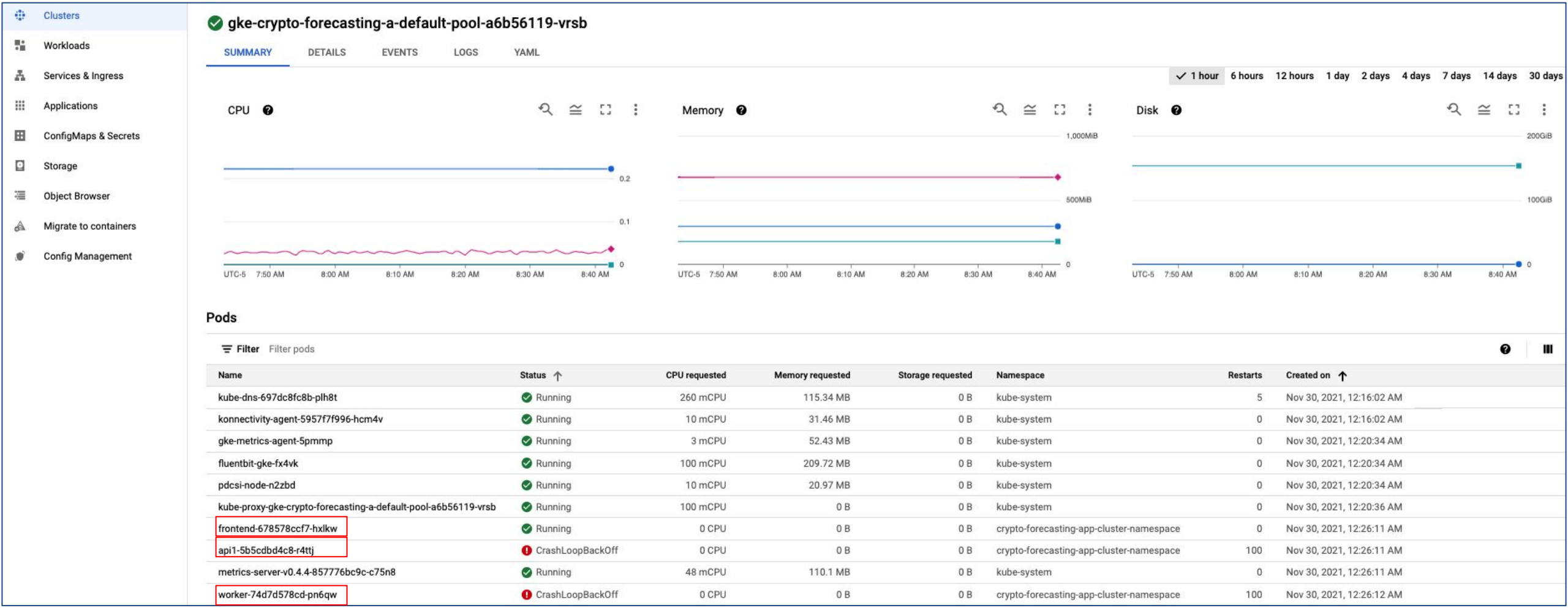
Deployment - Kubernetes Cluster



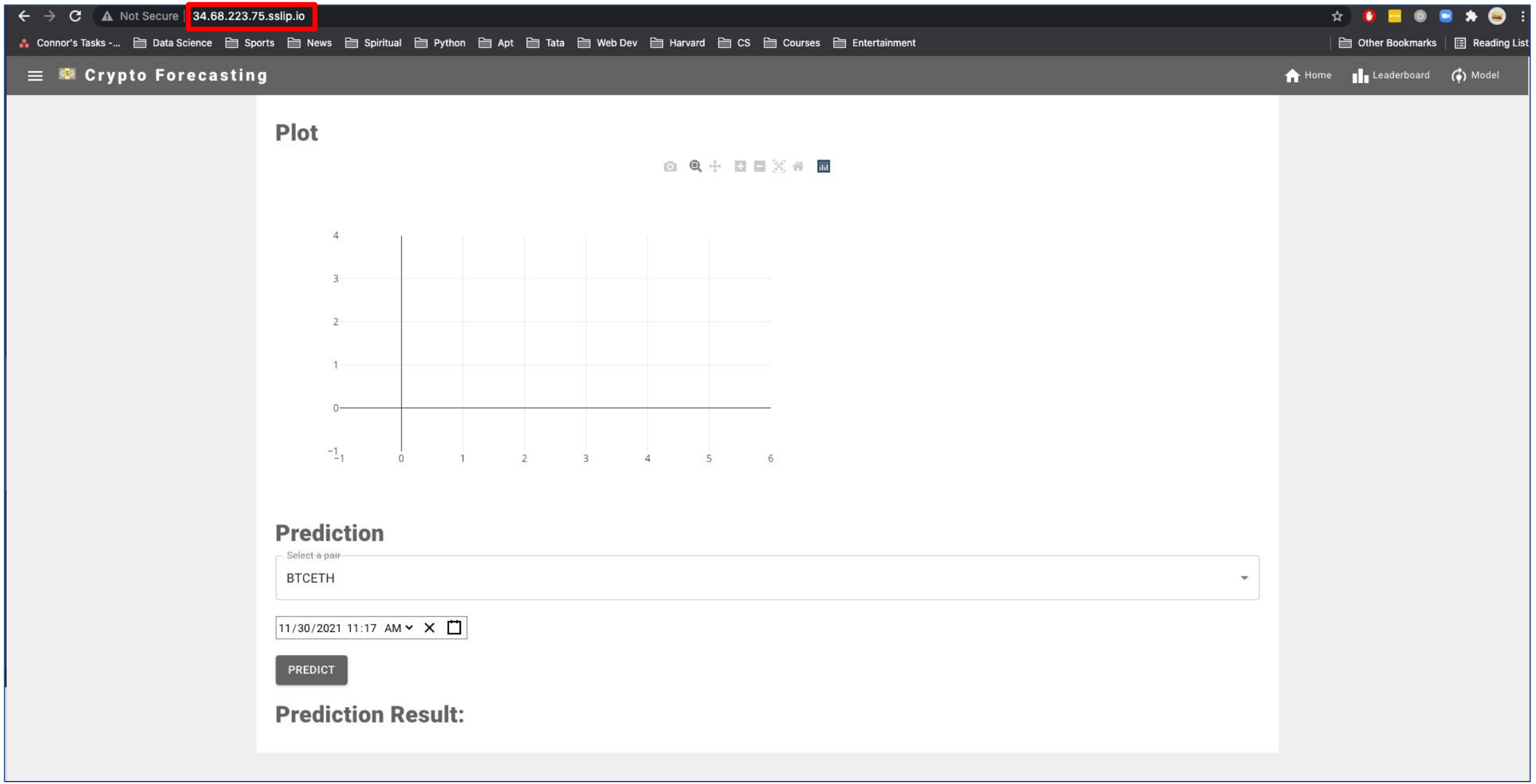
Deployment - Kubernetes Node Pool & Nodes



Deployment - Kubernetes Pod



Deployment - Kubernetes Crypto Forecasting App



Next Steps

- Kubernetes Completion (connecting Postgres database)
- Further LSTM tuning
- Connecting GCS bucket with models to API
- Building out frontend API calls
- Polish frontend design and plots visualization