thndr

Thndr DevOps Task

Description

You are required to build a simple RESTful API that simulates a simple stocks technical analysis application.

Stocks technical analysis shows a prediction for the stock price change through a certain period of time.

API

Implement the RESTful API using python. You can pick any framework that you're comfortable with if you want to.

There will be two roles in the app; an Admin who is responsible for adding the analysis, and a User who can list the stocks data.

N.B For simplicity, you are not required to implement any authentication nor access control, it is enough that you differentiate between user's actions and admin's actions by prefixing admin endpoints (i.e /admins).

Streamer

The API server will be connected to a streamer that pushes stock price changes to a messaging queue. This will give you data for 3 stocks' data structured as below.

The task's folder includes a docker-compose file that runs both the messaging queue and the streamer service. You will only need to consume messages from the queue.

N.B The API Server should persist the stocks data coming from the streamer. It should also mark the stock's *target_hit* flag to true in the following cases:

- If price >= target and type == "UP"
- If price <= target and type == "DOWN"

Message Queue Topic: `thndr-trading`

Message Queue Data:

```
{
"stock_id": "6ffb8e62-92c1-40c7-9d38-5b976a346b62", // stock id
"name": "CIB", // stock name
"price": 28, // current stock price
"availability": 46, // current stock availability
"timestamp": "2019-12-15 14:36:33.462393" // current timestamp
}
```

API Spec

Endpoint: *GET /stocks*

Response:

```
[{
"stock_id": "6ffb8e62-92c1-40c7-9d38-5b976a346b62", // stock id
"name": "CIB", // stock name
"price": 28, // current stock price
"availability": 46, // current stock availability
}]
```

Endpoint: GET */stocks/<stock_id>*

Response:

```
{
    "stock_id": "6ffb8e62-92c1-40c7-9d38-5b976a346b62", // stock id
    "name": "CIB", // stock name
    "price": 28, // current stock price
    "availability": 46, // current stock availability
    "technical_analysis": {
        "target": 29, // predicted target price
        "type": "UP", // type is either UP or DOWN
        "target_hit": false // indicates if the target was reached
    }
}
```

Endpoint: POST */admin/stocks/<stock_id>/analysis*

Request:

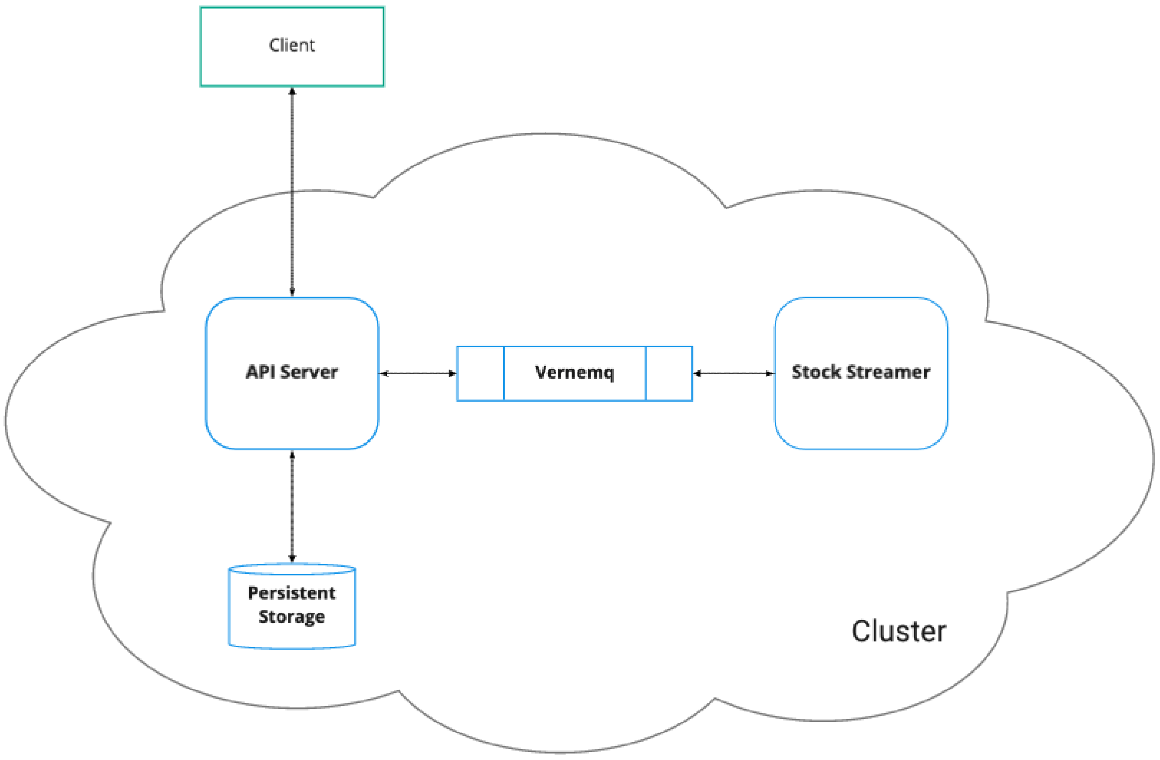
```
{
    "target": 29, // predicted target price
    "type": "UP", // type is either UP or DOWN
}
```

Response:

```
{
    "target": 29, // predicted target price
    "type": "UP", // type is either UP or DOWN
    "target_hit": false // indicates if the target was reached
}
```

Kubernetes Cluster

You are required to deploy the whole system on a minikube cluster. The system's architecture would look similar to the following figure.



Deployment notes:

- Only use declarative kubernetes manifests to make changes to the cluster.
- Only the API Server should be exposed publicly.
- Add Prometheus for metric collection.
- Using helm charts is a plus.
- Adding Grafana for metric visualization is a plus.