The Basic Stock Tracker

Connor Ryan, S4434200

Introduction

The Basic Stock Tracker Web application is about utilising the cloud capabilities of GCP's Compute Engine and VM to deploy a Flask based Stock tracker application in scale utilizing Docker and Docker Compose. This application is determined to bring a lightweight tool to store and monitor your favourite stocks and their relevant fundamental statistics.

The motivation behind this project is to provide a simple way to review a stock and its fundamentals over a period of time without all the unnecessary clutter that most modern sites possess today. Learning about stocks and relevant fundamentals can be quite difficult without proper financial knowledge, therefore having a single tool to look at and review a stock can make it much easier to gain insight.

The overall objectives of this project are to provide interested parties with a way to view a stocks price and stock fundamentals on the stocks across many different markets and countries. This will include a user login system till take a simple username and password to store in the database. This will allow the user to add stocks to their favourites for quick access once they are logged in. Users will have the ability to review a stock over a period of many days to years and return the stocks underlying fundamental values for reference if they are looking to perform analysis on the data.

Cloud computing brings many benefits to a project of this type. Due to the nature of the application requiring direct access to market data, utilising a web app is the easiest way to build the visualisation and implement the dataset from a web api to the widest user base. The ability to scale out and adjust the system requirements on the fly is also a key benefit that is brought to by cloud computing. With docker-compose and a virtual machine within Google's Cloud Computing platform, this gives us the ability to rapidly deploy new machines or upgrade existing ones due to the requirements from increasing user traffic or as the complexity of the application improves.

Technical Solutions

The front end technologies of this project are utilising a Bootstrap and client-side React from the Dash Plotly Python library. We utilised Bootstrap and React from within the Dash/Flask library to provide server side rendering and serving a completed page to the end user. This has the benefit of increasing site security to prevent from web scraping tools interacting with components as well as quickly adding new UI elements without the need for the incremental changes that are brought by raw HTML. Bootstrap and React are both utilised within Dashes comprehensive libraries to provide a minimalistic and fast deploying front end UI for the end user to work with.

The back end technologies of this project are utilising Flask and Dash. Flask is a micro web framework for Python in the web, it excels in building scalable web applications that will enable user interaction without the need for other libraries. This framework allows for readable code to control the user interaction and data flow within the app. With the utilisation of third party libraries as with psycopg2 and pandas we can also allow connections to the database, to utilise DDL and DML with postgres to manipulate the data to be displayed.

We are also using nginx to serve the web app and PostgreSQL as a database. Nginx is a web server that can effectively utilise load balancing to handle high levels of user requests to the web server. It allows users direct access to the flask web application without the need of entering the default ports. PostgreSQL is an open-source relational database that utilises extensibility and standard SQL compliance, this allows for quick entering and controlling of data with a large dataset for multi columned data as well as relational data across multiple tables.

The estimated cloud cost for this project is roughly $68.71 per month. This is currently utilising a Ubuntu 18.04 GCP VM located in Iowa (us-central1) with 8 vCPUs and 12.8 GBs of RAM (Refer to App. Figure 2.0. As all of the software being utilised is open-source, no costs are incurred from it.

Architecture Design

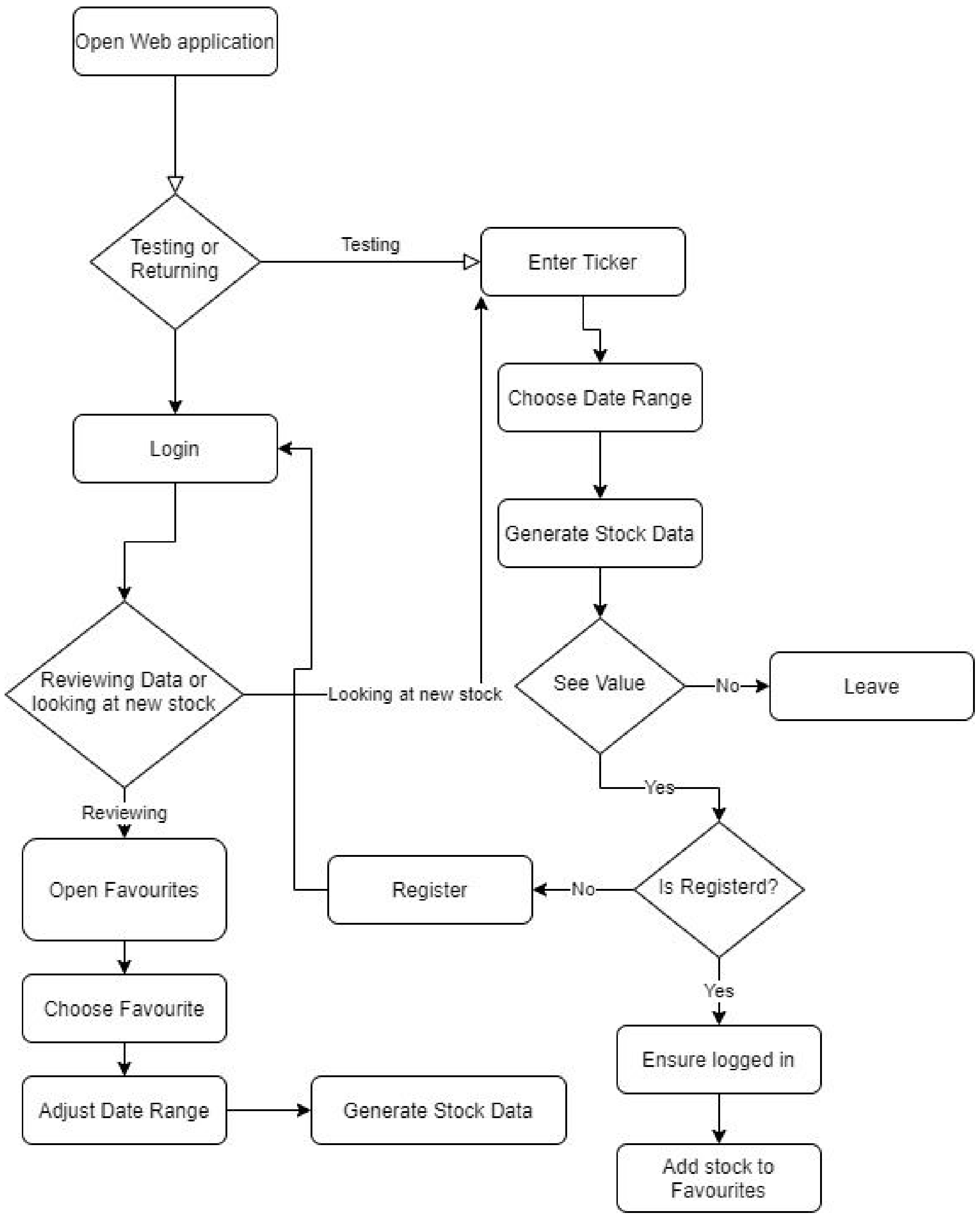
The architecture design of this project is based around the Docker-compose docker containers. There are 3 containers that are utilised in this web application; Postgres,webapp-flask and Nginx. These containers are dependent on each other in that order to help control for a race condition. The nginx web gateway receives incoming requests from the client browser which then passes it to the flask web app to be served to the user. The user is then able to make their interactions with the application as necessary, on specific events functions will fire that will allow the user to input or pull data from or into the postgres database. For detail around this schema, refer to figure 1.0

Utilising PostGreSQL, we have two primary tables users and userFavourites. These tables enable the main functionality around logging in and adding stocks to the users favourites to be pulled at a later date. The userId in user favourites is a foregin key that references the userId from the users table to perform its basic join on the data with a 1-M relationship. There is also a dateAdded column which is utilised as a logging system to refer to when a new user or stock is added.

users	[PK(UserId), UNIQUE(username), passwor, dateCreated]
userFavourites	[PK(id), FK(userId), ticker, dateAdded UNIQUE(userId,ticker)

Refer to figure 1.1 for the database schema diagram,

Workflow figure



Appendix

Figure 1.0

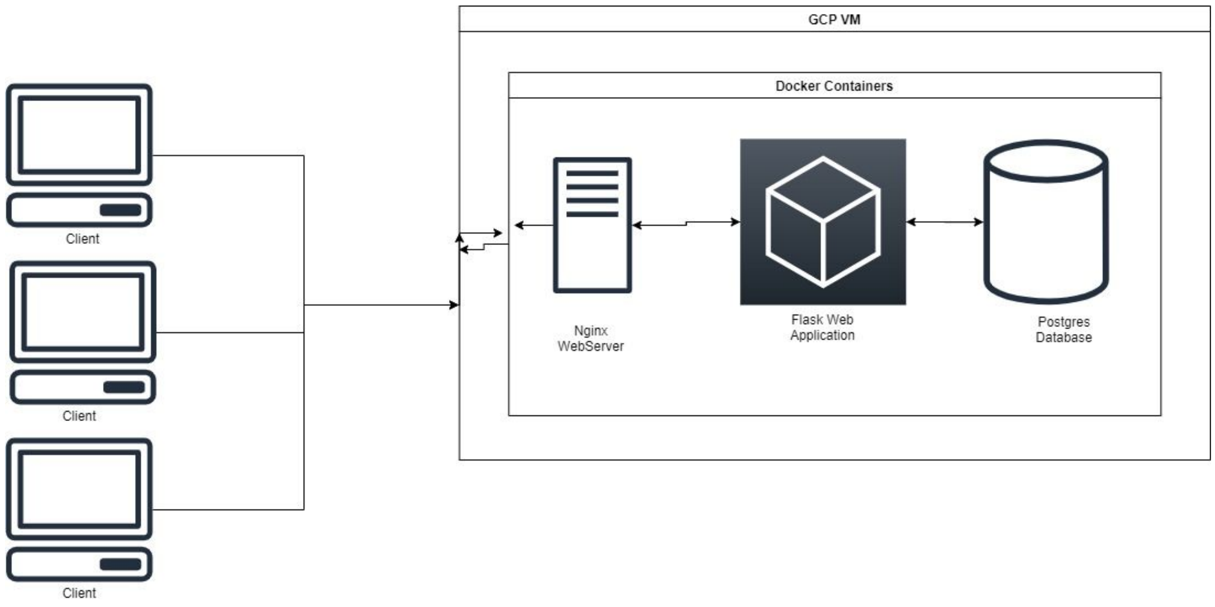
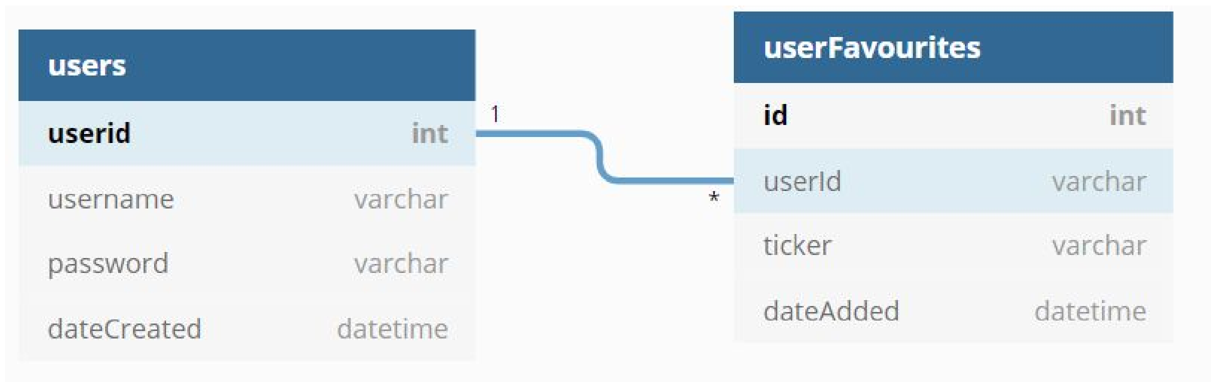


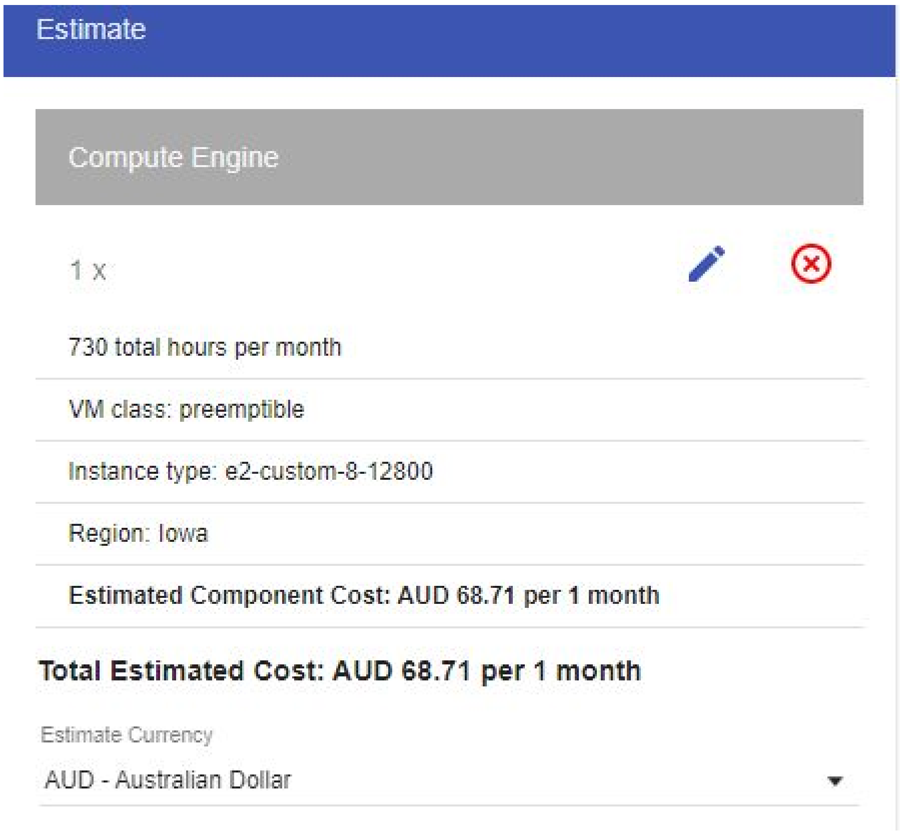
Figure 1.1



users	
userid	int
username	varchar
password	varchar
dateCreated	datetime

userFavourites	
id	int
userId	varchar
ticker	varchar
dateAdded	datetime

Figure 2.0



Estimate

Compute Engine

1 x

730 total hours per month

VM class: preemptible

Instance type: e2-custom-8-12800

Region: Iowa

Estimated Component Cost: AUD 68.71 per 1 month

Total Estimated Cost: AUD 68.71 per 1 month

Estimate Currency

AUD - Australian Dollar