Provisioning Infrastructure To Facilitate Containerised Microservice Based High Frequency Trading (Hft) Application Using Docker In Google Cloud

Dr .M.VIMALA DEVI
M.E., PhD., PROFESSOR and HEAD
Department of Computer Science and Engineering
K S R Institute for Engineering and Technology
Tiruchengode- 637 215

Dharsha K, Dhevak V, Ganesh M, Harikrishnaa S
Department of computer science and Engineering
KSR institute for engineering and technology
Tiruchengode

Abstract—High-frequency trading (HFT) is a type of algorithmic trading that involves the use of advanced trading algorithms and computer systems to execute trades at high speeds and frequencies. To achieve this, we will use a micro services architecture that allows us to break down the application into smaller, independent services that can be developed, deployed, and scaled independently. The micro services will be deployed as Dockers containers running on Google Cloud, which provides a scalable and reliable infrastructure for managing and orchestrating containerized applications. The high-frequency trading application will be designed to ingest real-time market data and execute trades based on predefined trading strategies. These micro services will communicate with each other using a messaging system to facilitate the exchange of data and enable real-time decision-making. To ensure the security and reliability of the application, we will implement advanced security measures such as data encryption, identity and access management, and logging and monitoring. We will also use Google Cloud's managed services, such as Google Dockers to provide a scalable and cost-effective infrastructure for the application.
By leveraging micro services and Dockers in Google Cloud, we can create a high-performance, scalable, and reliable high-frequency trading application that can meet the demands of today's fast-paced trading environments.

I. 1.INTRODUCTION

The aim of this project is to provision a containerized HFT application on Google Cloud Platform (GCP) using Dockers to improve the efficiency and reliability of HFT applications while reducing infrastructure costs and improving security. The project will achieve this by selecting an appropriate HFT application, creating a Dockers file to build the container image, deploying the application on GCP, and configuring networking and storage. The expected outcome of this project is a robust and scalable containerized HFT application that can be easily deployed and managed on GCP using Dockers.

II. LITERATURE REVIEW

[1] The Design and Architecture of Micro services by Alan Sill (Texas Tech University):

The services like Design Methodology, Cloud Computing which is a basic one , Computer Architecture the main design , Standards Development, Service Computing which includes multiple services , Storage Automation, Micro services based containers, Networks Micro services are sweeping through cloud design architectures These are the current new technologies for emerging cloud security. This above services and methodologies are used as a unique or group to improve the Micro service Development.

[2] Micro service security: a systematic literature review by 1)Davide Berardi, 2)Saverio Giallorenzo,3) Jacopo Mauro, 4)Andrea Melis, 5)Fabrizio Montesi :

For developing distributed systems the Micro services is an arising paradigm. With their wide relinquishment, for the relation between micro services and security, more and more workshop are delved. Alas, this subject doesn't has a well- defined corpus as a literature it's thrown over numerous venues and composed of benefactions substantially addressing specific scripts or requirements. In this work, the field, by gathering 290 applicable

publications the methodical review is conducted — at the time of jotting, the largest curated dataset on the content. We assay our dataset along two lines(a) one is quantitatively- through publication metadata, which allows us to chart publication outlets, approaches, communities and dived issues;(b) another bone is qualitatively, we used to give an added up overview of the literature and to spot gaps left open through 20 exploration questions. Then we summaries our analyses in the form in the conclusion of a call for action to address the main open challenges.

[3] A Review of Container level Auto scaling for Micro services-based Applications by Mohamed Hedi Fourati Red CAD Laboratory, University of Sfax, Tunisia; Soumaya Marzouk; Mohamed Jmaiel:

This paper presents an overview of auto scaling results for micro services- grounded operations during elastic treatment. Actually, utmost of being work propose results dealing with plainness at the VM position. These results launch plainness when VM load is detected or prognosticated. Many results treat this issue at the vessel position and deal with plainness of micro services- grounded operations. In addition, these ultimate use ideas employed at the VM position without considering the particularity of micro service armature. In this paper, we study and classify being auto scalars dealing with holders and planting micro services- grounded operations. We explain the strength and the failings of each order. As a conclusion, we describe the challenges of auto scaling treatment and we give recommendations for unborn results.

III. METHODOLOGY

TASK 1: DEFINE THE REQUIREMENTS OF THE HFT APPLICATION AND CREATE A DETAILED PROJECT PLAN.

Defining the requirements of the HFT application is a crucial first step in the project plan, as it will guide the development process and ensure that the application meets the needs of the users. The requirements should be gathered from stakeholders, including traders, analysts, and IT staff, and should be documented in a clear and concise manner.

The project plan should be developed based on the requirements, and should include the following tasks:

Design the architecture of the application, including the micro services and their interactions, and the deployment model on Cabernets.

Select and configure the necessary GCP services, including Google Cabernets Engine, Cloud Storage, Cloud SQL, Cloud Pub/Sub, and Cloud IAM

. Develop the micro services using a suitable programming language and frameworks, with a focus on high-performance and low latency. Implement monitoring and logging using Google Cloud Monitoring and Logging, to enable real-time performance and system health insights. Perform load and stress testing of the application to ensure its ability to handle high volumes of traffic...

TASK 2: SET UP A GCP ACCOUNT AND CREATE A NEW PROJECT.

To set up a GCP account and create a new project, follow these steps: Go to the google cloud console (console.cloud.google.com)

Click on the "Create Project" button in the top bar.

Enter a name for your project and select a billing account .Click on "Create" to create the project.

Once the project is created, you can access it by selecting it from the project dropdown menu in the top bar.

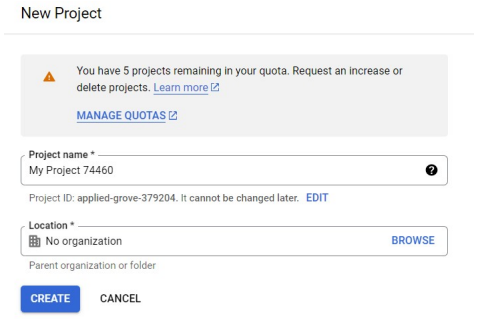

Fig.1.1. creating a new project in GCP console.

TASK 3: SET UP THE NECESSARY IAM ROLES AND PERMISSIONS FOR PROJECT MEMBERS.

To set up the necessary IAM places and warrants for design members in GCP, follow these way Go to the Google Cloud Console and elect your design. Click on the" IAM & admin" menu option in the left sidebar. Click on the" IAM" tab to see the current IAM places and warrants for the design. Click on the" Add" button at the top of the runner to add a new member to the design. Enter the dispatch address of the member you want to add and elect the applicable IAM part for them. Click on the" Save" button to add the member to the design with their assigned part.

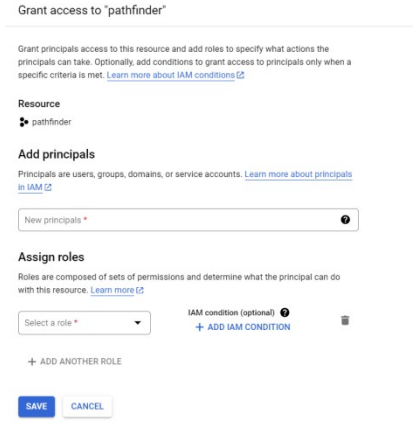

Fig.1.2. creating IAM role.

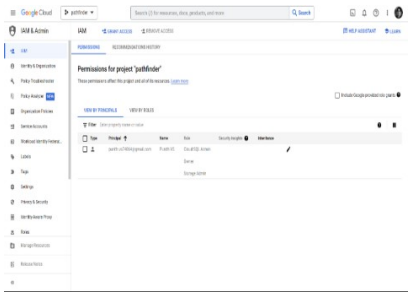

Fig.1.3. overview console of IAM.

Fig 1.2 shows a principle with the privilege of “project owner”, “cloud storage admin” and “cloud SQL admin”. The complete IAM roles that need to be implemented for the proposed project "Building a Micro service High-Frequency Trading Application using Google Cabernets Engine" are as follows:

Project Owner: The Project Owner has full control over all resources within the project, including the ability to add and remove members, create and delete resources, and modify IAM roles.

Monitoring Editor: The Monitoring Editor role is required to view monitoring data and create monitoring dashboards.

Logging Admin: The Logging Admin role is required to view and manage logs in Cloud Logging.

Service Account User: The Service Account User role is required to use service accounts to authenticate with GCP APIs and services.

Note: that these roles should be assigned to specific individuals or groups based on their responsibilities and access requirements. Additionally, the roles should be granted the minimum necessary permissions to perform their respective tasks in order to maintain the security of the project.

Note: It is important to carefully consider the roles and permissions you assign to project members to ensure that they have the appropriate level of access and control over project resources.

TASK 4: CREATE A PRIVATE VPC NETWORK AND SUBNETWORKS FOR THE PROJECT.

Go to the Google Cloud Console and elect your design. In the left- hand navigation menu, elect" VPC network" and also" VPC networks". Click on" produce VPC network" and give that network a name. elect" Custom" for" IPv4 CIDR range" and enter the CIDR block you want to use for your network. Under" Subnets", click on" Add subnet". Give your subnet a name and elect the region and zone where you want to emplace your HFT operation. Enter the CIDR range you want to use for your subnet. Make sure the range is a subset of the VPC network CIDR block. Elect" Private subnet" to insure that your subnet isn't intimately accessible from the internet. Reprise way 5- 8 for

each fresh subnet you want to produce in different regions zones. Once you have created all the necessary subnets, you can configure firewall rules to control access to your coffers in the VPC network.

TASK 5: CONFIGURE CLOUD LOAD BALANCER, AND NECESSARY FIREWALL RULES.
Configuring firewall rules:
Go to the VPC network section in the Google Cloud Console. Click on the Firewall rules tab.
Click on the Create Firewall Rule button.
Enter a name for the firewall rule. Choose the network where you want to apply the firewall rule.
Enter the IP range that you want to apply the firewall rule to. For example, you can specify the range of IP addresses for your subnets. Select the protocols and ports that you want to allow or deny. Choose the action you want to take when the rule is matched (allow or deny).
Click the Create button to create the firewall rule. Simple insert statements. The decision is yours, but you must load all 500 rows.
Configuring Health Checks:
Create the HTTP firewall rule
Create a firewall rule to allow HTTP traffic to back ends.
In the Cloud Console, navigate to Navigation menu (☰) > VPC network > Firewall.
Notice the existing ICMP, internal, RDP, and SSH firewall rules.
Click Create Firewall Rule.
Set the following values, leave all other values at their defaults:

Make sure to include the /0 in the Source IPv4 ranges to specify all networks.
5. Click Create
Create the health check firewall rules:
Health checks determine which instances of a load balancer can receive new connections. For HTTP load balancing, the health check probes to your load balanced instances come from addresses in the ranges <configured VPC subnet IP range>. Your firewall rules must allow these connections. 1. Still in the Firewall page, click Create Firewall Rule. 2. Set the following values, leave all other values at their defaults:],
TASK 6: CREATE A DOCKER IMAGE FOR THE HFT APPLICATION AND PUSH IT TO THE GOOGLE CONTAINER REGISTRY.
In this section, you will build a Dockers image that's based on a simple node application.
Execute the following command to create and switch into a folder named HFT

```
mkdir hft && cd hft
Create a Dockerfile:
  Cat > Dockerfile <<EOF
# Use an official Node runtime as the parent image
FROM node:lts
# create app directory
WORKDIR /app
# Copy package.json and package-lock.json
COPY package*.json ./
# Install app dependencies
RUN npm install
# Copy app source code
COPY. .
# Expose the port that the app is listening on
EXPOSE 3000
# Start the app
CMD [ "npm", "start" ]
EOF
```

This is a simple HTTP server that listens on port 3000 and returns "Current Exchange Rates Data".
Now build the image
Note again the ".", which means current directory so you need to run this command from within the directory that has the Dockerfile:

docker build -t node-app:0.1 .
It might take a couple of minutes for this command to finish executing. When it does, your output should resemble the following:
Running the Docker Images
use this code to run containers based on the image you built:
docker run -p 4000:3000 --name my-app node-app:0.1
Open terminal (in Cloud Shell, click the + icon), and test the server:
curl http://localhost:4000
The container will run as long as the initial terminal is running. If you want the container to run in the background (not tied to the terminal's session), you need to specify the -d flag.
Close the initial terminal and then run the following command to stop and remove the container
docker stop my-app && docker rm my-app
Now run the following command to start the container in the background
docker run -p 4000:80 --name my-app -d node-app:0.1
docker ps
Publishing Docker Images
Now you're going to push your image to the Google Artifact Registry. After that you'll remove all containers and images to simulate a fresh environment, and then pull and run your containers. This will demonstrate the portability of Docker containers.
Create the target Docker repository
You must create a repository before you can push any images to it. Pushing an image can't trigger creation of a repository and the Cloud Build service account does not have permissions to create repositories.
From the Navigation Menu, under CI/CD navigate to Artifact Registry > Repositories.
Click Create Repository.
Specify my-repository as the repository name.
Choose Docker as the format.
Under Location Type, select Region and then choose the location us-central1 (Iowa).
Click Create.
Configure authentication
Before you can push or pull images, configure Docker to use the Google Cloud CLI to authenticate requests to Artifact Registry.
To set up authentication to Docker repositories in the region us-central1, run the following command in Cloud Shell:
gcloud auth configure-docker us-central1-docker.pkg.dev
Enter Y when prompted.
The command updates your Docker configuration. You can now connect with Artifact Registry in your Google Cloud project to push and pull images

Push the container to Artifact Registry.
Run the following commands to set your Project ID and change into the directory with your Dockerfile.
export PROJECT_ID=$(gcloud config get-value project)
cd ~/hft
docker build -t us-central1-docker.pkg.dev/$PROJECT_ID/my-repository/node-app:0.1 .
Run the following command to check your built Docker images
docker images
Push this image to Artifact Registry
docker push us-central1-docker.pkg.dev/$PROJECT_ID/my-repository/node-app:0.1
Test the image
Stop and remove all containers
docker stop $(docker ps -q)
docker rm $(docker ps -aq)
Run the following command to remove all of the Docker images.
docker rmi us-central1-docker.pkg.dev/$PROJECT_ID/hft-repository/node-app:0.1
docker rmi node:lts
docker rmi -f $(docker images -aq) # remove remaining images
docker images
Pull the image and run it

```
docker pull us-central1-docker.pkg.dev/$PROJECT_ID/hft-repository/node-app:0.1
docker run -p 4000:3000 -d us-central1-docker.pkg.dev/$PROJECT_ID/hft-repository/node-app:0.1
curl http://localhost:4000
```

IV. CONCLUSIONS

Proposed project for "provisioning a containerized HFT application using Dockers in Google Cloud" is a promising solution for addressing the challenges faced by HFT firms in terms of scalability, portability, and security. By adopting a containerized micro services architecture and leveraging the benefits of cloud infrastructure, the proposed project can improve the efficiency, scalability, and security of HFT applications. The use of Dockers containers for encapsulating the HFT application and its dependencies provides a more portable and scalable solution, enabling the application to be easily deployed on any system that supports Dockers and Cabernets. By adopting a micro services architecture, the application can be broken down into smaller, independent services that can be developed and deployed separately, providing greater flexibility and scalability. By deploying the containerized HFT application on GCP using Cabernets, the proposed project can take advantage of the benefits of cloud infrastructure, such as automatic scaling, load balancing, and pay-as-you-go pricing models. This can help to reduce infrastructure costs and improve the efficiency and reliability of the application. In addition, the principle of containerization provides an additional layer of security for the HFT application. By isolating the application from the host operating system and other applications running on the system, containerization can mitigate the risks of security vulnerabilities and ensure the integrity of the application. The built-in security features of GCP, such as firewalls and access controls, can further improve the security of the HFT application.

REFERENCES

[1]. "Containerized Microservices Orchestration and Provisioning in Cloud Computing: A Conceptual Framework and Future Perspectives"by Abdul Saboor, Mohd Fadzil Hassan, Rehan Akbar

[2]. G.Neelakrishnan, P.Iraianbu, T.Abishek, G.Rajesh, S.Vignesh, "IOT Based Monitoring in Agricultural" International Journal of Innovative Research in Science, Engineering and Technology, March 2020, Volume 9, Issue 3, pp:814-819

[3]. "Microservices: architecture, container, and Challenges" by Guozhi Liu, Bi Huang, Minmin Qin.

[4]. C.Nagarajan and M.Madheswaran - 'Experimental verification and stability state space analysis of CLL-T Series Parallel Resonant Converter' - *Journal of ELECTRICAL ENGINEERING,* Vol.63 (6), pp.365-

[5]. "Container-based Microservice Architecture for Cloud Applications" byVindeep Singh Department of Computer Science & Engineering Indian Institute of Technology Roorkee-Roorkee, India and Sateesh K Peddoju Department of Computer Science & Engineering, Indian Institute of Technology Roorkee -Roorkee, India.

[6]. Nagarajan and M.Madheswaran - 'Experimental Study and steady state stability analysis of CLL-T SeriesParallel Resonant Converter with Fuzzy controller using State Space Analysis'- *Iranian Journal of Electrical & Electronic Engineering*, Vol.8 (3), pp.259-267, September 2012.

[7]. C. Pahl, A. Brogi, J. Soldani and P. Jamshidi, "Cloud Container Technologies: A State-of-the-Art Review," in IEEE Transactions on Cloud Computing

[8]. C.Nagarajan and M.Madheswaran - 'Stability Analysis of Series Parallel Resonant Converter with Fuzzy Logic Controller Using State Space Techniques'- *Taylor & Francis, Electric Power*

[9]. .C. Pahl, "Containerization and the PaaS Cloud," in IEEE Cloud Computing,

[10]. G. Granchelli, M. Cardarelli, P. Di Francesco, I. Malavolta, L. Iovino and A. Di Salle, "MicroART: A Software Architecture Recovery Tool for Maintaining Microservice-Based Systems," 2017 IEEE International Conference on Software Architecture Workshops (ICSAW), Gothenburg, 2017, pp. 298-302, doi: 10.1109/ICSAW.2017.9.

[11]. G.Neelakrishnan, R.S.Jeevitha, P.Srinisha, S.Kowsalya, S.Dhivya, "Smart Gas Level Monitoring, Booking and Gas Leakage Detector over IOT" International Journal of Innovative Research in Science, Engineering and Technology, March 2020, Volume 9, Issue 3, pp: 825-836

[12]. N. Alshuqayran, N. Ali and R. Evans, "A Systematic Mapping Study in Microservice Architecture," 2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA), Macau, 2016, pp. 44-51, doi: 10.1109/SOCA.2016.15.

[13]. X. Zhou et al., "Fault Analysis and Debugging of MicroserviceSystems: Industrial Survey, Benchmark System, and Empirical Study,"in IEEE Transactions on Software Engineering, doi:10.1109/TSE.2018.2887384.

[14]. C.Nagarajan and M.Madheswaran - 'Performance Analysis of LCL-T Resonant Converter with Fuzzy/PID Using State Space Analysis'- *Springer, Electrical Engineering*, Vol.93 (3), pp.167-178, September 2011.

[15]. F. Dai, H. Chen, Z. Qiang, Z. Liang, B. Huang, L. "Wang, Automatic Analysis of Complex Interactions in Microservice Systems," Complexity, 2020, doi: 10.1155/2020/2128793.

[16]. M. Villamizar et al. Cost comparison of running web applications in the cloud using monolothic, microservice, and AWs Lambada architecture [C]// Proceedings of the Service Oriented Computing and Applications.