Alexander Meau (alme9155)

CSCA-5028 Applications of Software Architecture for Big Data – Final Project

Project Title:

- AI-Powered Movie Sentiment Rating System

Project Requirements

(A- Level Work Rubics with Implement details)

Requirements (A-Level Work)	Status	Implementation
1. Web application basic form, reporting	☑	Implemented web form "/" to submit movie review, and report page "/report"
2. Data collection	☑	Implemented /collect method to fetch movie reviews from web API. Fetched data are stored in NoSQL and then push RabbitMQ.
3. Data analyzer	☑	Implemented fine-tuned NLP engine to provide movie ratings from sentiment analysis.
4. Unit tests	☑	Implemented Unit test for all server and components classes.
5. Data persistence any data store	☑	Implemented MongoDB NoSQL data store to store raw web API records and analyze results.
6. Rest collaboration internal or API endpoint	☑	Implemented web API end points to collaborate between frontend, analyze, and collect services. (e.g. /collect, /analyze, /publish)
7. Product environment	☑	Hosted on Google cloud.
8. Integration tests	☑	Implemented gradlew integration to submit review movie reviews to docker service.
9. Using mock objects or any test doubles	☑	Hosted on Google cloud.
10. Continuous integration	☑	Utilized github for continuous integration.
11. Production monitoring instrumenting	☑	Implemented Promethesus and Grafana to monitor /health and /metrics of ktor services.
12. Event collaboration	☑	Implemented RabbitMQ to handle movie reviews feteched from online API.
13. Continous delivery	☑	Implemented deploy.sh script to facilitate continuous delivery to google cloud.

Alexander Meau (alme9155)

Overview

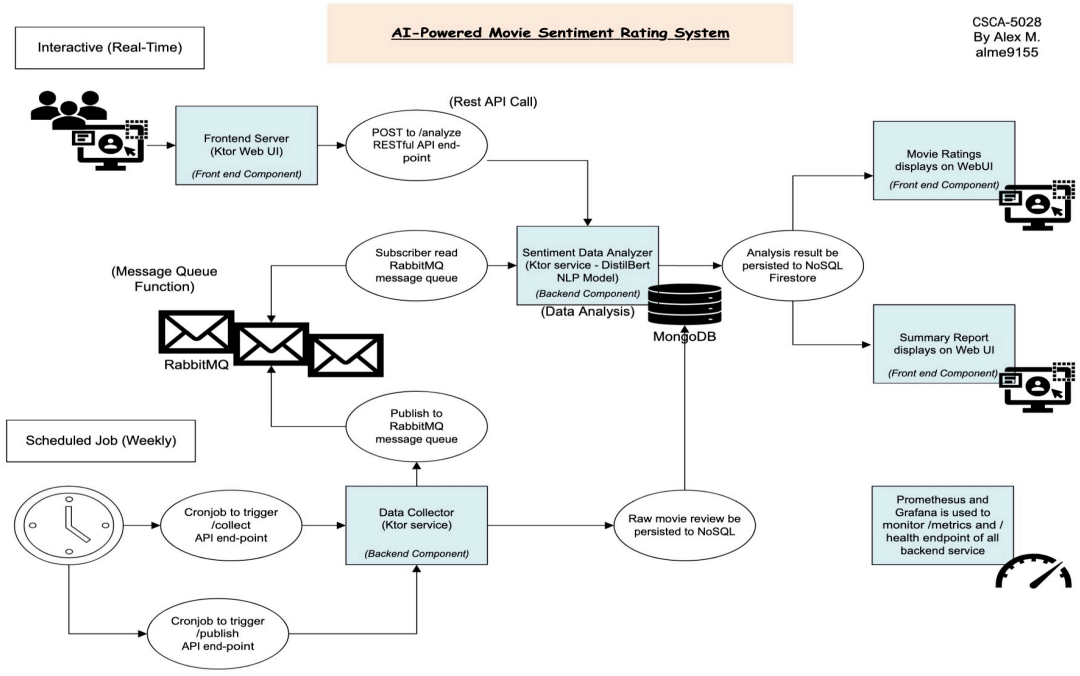
This project aims to convert user-submitted movie reviews into rating scores using sentiment analysis from a Natural Language Processor. Movie ratings and reviews can be sold to online movie platforms, such as Netflix, to provide movie recommendations.

1. ***What problem is the product aimed at solving?***
   This product aims to address the challenges of generating rating scores for movie review text.

2. ***Who is the product geared towards (targeted audience)?***
   The primary target audience is online movie platforms, such as Netflix or Amazon Prime, to use these ratings in their recommender system.

3. ***How is the product unique?***
   This product is unique in its application of **transfer learning**, where a pre-trained NLP model is **fine-tuned for the domain of movie reviews**.

Architecture



Project Components

1. **Frontend Server**
   - Frontend server is a Ktor web application to provide an HTML frontend to allow end-users to submit movie reviews. It also has a reporting page to display top 10 movies based on sentiment ratings.
2. **Data Collector**
   - Data Collector is a Ktor service responsible for downloading critics reviews using an external API. Downloaded data first persisted in a NoSQL document store and then published to message queue for further processing.
3. **Sentiment Data Analyzer**
   - Data analyzer is a Ktor web service that loads a pre-trained Hugging Face NLP model (i.e. DistilBERT) to perform sentiment analysis. Data analyzer services accept user input submitted by front-end server and have a worker pool to process movie review data published to message queue.
4. **Data Persistence**
   - A Mongo NoSQL Document store is used within the entire system. It stores raw data collected from the web API and sentiment analysis results for reporting purposes.
5. **Message Queue**
   - Implemented RabbitMQ service to process data collected from external web API using publish/subscribe workers.

Key Design Decisions

- The project is set up using the Kotlin/Ktor class template. The class template has well-established class responsibilities using the MVC design pattern: model: Data collector Server, **view: frontend-server, controller: web endpoint to communicate between services.**
- For the persistence layer, a **NoSQL MongoDB database is chosen**. In module 3 of the assignment, there is a project requirement to fetch data from an external Web API and persist it to local storage. A NoSQL database would be ideal for this project requirement because we do not want to risk the system failing when the schema of the external Web API is modified. In most industry practice, it is the responsibility of the ingest module to understand the schema of collected data. **Due to this class requirement, choosing a NoSQL database provides flexibility without sacrificing performance.**
- For the data analyzer module, I have decided to fine-tune a pre-trained BERT natural language processing model from Hugging Face and adapt it for sentiment analysis

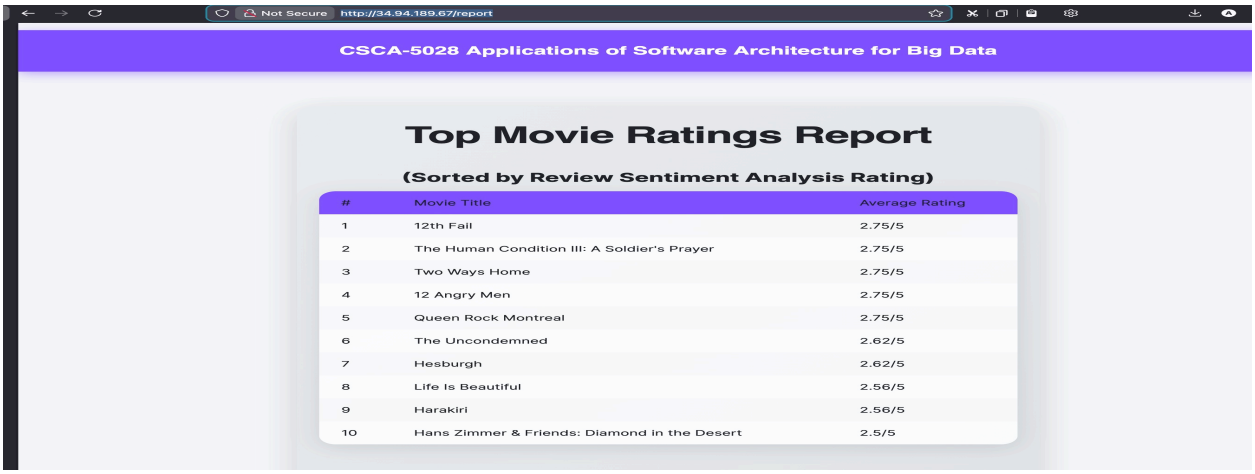
in the domain of movie reviews. My collab workbook has been checkin into GitHub and attached in the source zip file for review (ie, docs/Movie Ratings Sentiment Analysis v3 Collab.pdf)

- Due to GitHub's size limit, I have uploaded my fine-tuned model to Hugging Face for public access. This fine-tuned NLP model will be downloaded during the git build process. (URL: https://huggingface.co/alme9155/distilbert-sst5-finetuned-v3)
- **Rest API endpoints are used between server components.**
- Unit tests have been added to all server applications and component classes. Mock objects and test doubles are implemented using the **io.mockk library.**
- **Integration tests are implemented following the Gradle standard**. It assumes all Docker services are running and then submits the user response to the frontend service. The frontend services will call the analyzer service for sentiment analysis. At the end, the content of the return web page is examined using assert statements.
- **For continuous integration,** all source code has been checked into the **GitHub repo: https://github.com/alme9155/csca-5028-sentiment-analysis/**
- For product environment, **Google cloud is chosen because student has free credit on Google cloud.** Since this project uses a Prometheus Docker image, it is best to host the entire project within a virtual machine; this makes it much easier for the code reviewer to examine both the local and cloud environments.
- **For deployment,** a shell script deploy.sh has been written to allow **GitHub Actions to deploy the project into Google Cloud.**

Major Project Workflow:

- **Interactive (Real-Time):** Users can submit movie reviews via the web UI. The web UI will internally pass forward the review text to the RESTful API for the sentiment analyzer services. The NLP model will then analyze the review text and return the result ratings to be displayed on the webpage.

- **Batch (Weekly)**: A scheduled job will be executed periodically to trigger the data collector service to retrieve recent movie critics' reviews from the external web API. The data collected online will first be persisted in a MongoDB NoSQL database, then pushed to a message queue for service workers to analyze and process. The final rating results will be stored in a NoSQL database. The user can review the top 10 movies based on rating score from the report web page.

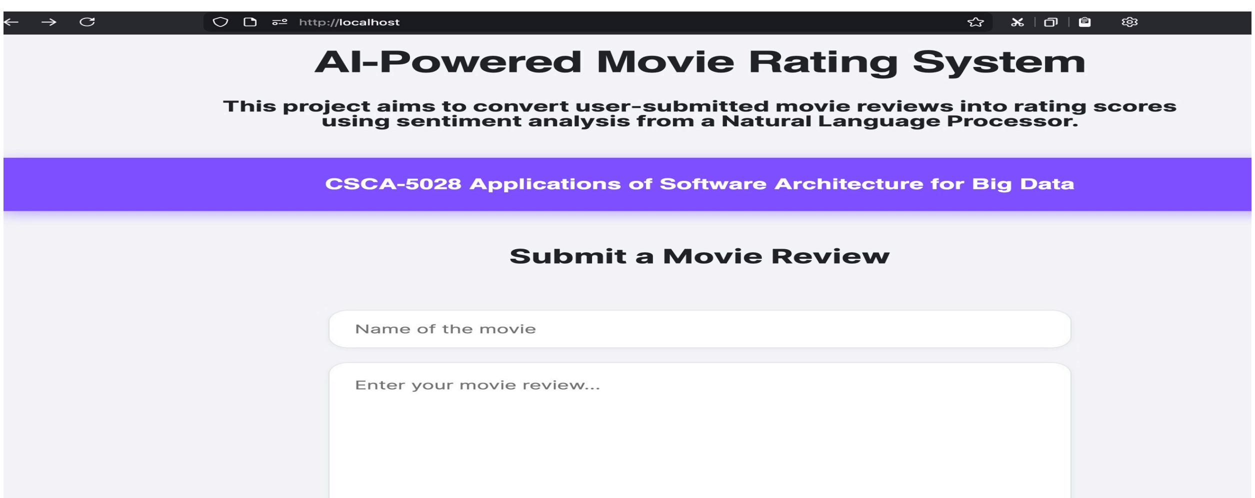


Build, and Test Process:

- >./gradlew clean build
- > docker compose build –no-cache
- > docker compose up –d
- > sh scripts/post-collect.sh
- > sh scripts/post-publish.sh
- Then, you can examine the web form (http://localhost) or report (http://localhost/report)

```
superman:csca-5028-sentiment-analysis ameau$ ./gradlew clean build
Starting a Gradle Daemon (subsequent builds will be faster)

> Task :components:sentiment:downloadCustomModel
Downloading DistilBERT SST-5 model into /Users/ameau/Downloads/temp-test/csca-5028-sentiment-analysis/components/senti
ment/models/distilbert-sst5-finetuned-v3
  [GET ] https://huggingface.co/alme9155/distilbert-sst5-finetuned-v3/resolve/main/config.json
  [OK  ] Saved config.json
  [GET ] https://huggingface.co/alme9155/distilbert-sst5-finetuned-v3/resolve/main/tokenizer.json
  [OK  ] Saved tokenizer.json
  [GET ] https://huggingface.co/alme9155/distilbert-sst5-finetuned-v3/resolve/main/tokenizer_config.json
  [OK  ] Saved tokenizer_config.json
  [GET ] https://huggingface.co/alme9155/distilbert-sst5-finetuned-v3/resolve/main/special_tokens_map.json
  [OK  ] Saved special_tokens_map.json
  [GET ] https://huggingface.co/alme9155/distilbert-sst5-finetuned-v3/resolve/main/vocab.txt
  [OK  ] Saved vocab.txt
  [GET ] https://huggingface.co/alme9155/distilbert-sst5-finetuned-v3/resolve/main/model.safetensors
  [OK  ] Saved model.safetensors
  [GET ] https://huggingface.co/alme9155/distilbert-sst5-finetuned-v3/resolve/main/training_args.bin
  [OK  ] Saved training_args.bin

> Task :components:messaging:compileKotlin
w: file:///Users/ameau/Downloads/temp-test/csca-5028-sentiment-analysis/components/messaging/src/main/kotlin/cu/csca50
28/alme9155/messaging/ReviewsDataHandler.kt:207:21 This is a delicate API and its use requires care. Make sure you ful
ly read and understand documentation of the declaration that is marked as a delicate API.
Java HotSpot(TM) 64-Bit Server VM warning: Sharing is only supported for boot loader classes because bootstrap classpa
th has been appended
Java HotSpot(TM) 64-Bit Server VM warning: Sharing is only supported for boot loader classes because bootstrap classpa
th has been appended
Java HotSpot(TM) 64-Bit Server VM warning: Sharing is only supported for boot loader classes because bootstrap classpa
th has been appended

Deprecated Gradle features were used in this build, making it incompatible with Gradle 9.0.

You can use '--warning-mode all' to show the individual deprecation warnings and determine if they come from your own
scripts or plugins.

For more on this, please refer to https://docs.gradle.org/8.7/userguide/command_line_interface.html#sec:command_line_w
arnings in the Gradle documentation.

BUILD SUCCESSFUL in 2m 9s
65 actionable tasks: 52 executed, 13 up-to-date
```

```
superman:csca-5028-sentiment-analysis ameau$ docker compose build --no-cache
[+] Building 19.0s (27/27) FINISHED                                                        docker:desktop-linux
 => [data-collector internal] load build definition from Dockerfile                                        0.0s
 => => transferring dockerfile: 1.90kB                                                                     0.0s
 => [frontend-server internal] load build definition from Dockerfile                                       0.0s
 => => transferring dockerfile: 1.90kB                                                                     0.0s
 => [data-analyzer internal] load build definition from Dockerfile                                         0.0s
 => => transferring dockerfile: 1.90kB                                                                     0.0s
 => [data-collector internal] load metadata for docker.io/library/alpine:3.20                              1.1s
 => [data-analyzer internal] load metadata for docker.io/library/eclipse-temurin:21-alpine                 1.1s
 => [data-collector auth] library/eclipse-temurin:pull token for registry-1.docker.io                      0.0s
 => [data-collector auth] library/alpine:pull token for registry-1.docker.io                               0.0s
 => [frontend-server internal] load .dockerignore                                                          0.0s
 => => transferring context: 2B                                                                            0.0s
 => [data-analyzer internal] load .dockerignore                                                            0.0s
 => => transferring context: 2B                                                                            0.0s
 => [data-collector internal] load .dockerignore                                                           0.0s
 => => transferring context: 2B                                                                            0.0s
 => CACHED [data-analyzer stage-1 1/7] FROM docker.io/library/alpine:3.20@sha256:765942a4039992336de8dd5db68058  0.0s
 => CACHED [data-analyzer jre-build 1/2] FROM docker.io/library/eclipse-temurin:21-alpine@sha256:c4799f335a65b1  0.0s
 => [frontend-server internal] load build context                                                          2.4s
 => => transferring context: 51.99MB                                                                       2.3s
 => [data-collector stage-1 2/7] RUN apk add --no-cache ca-certificates tini                               4.0s
 => [data-collector internal] load build context                                                           2.2s
 => => transferring context: 51.85MB                                                                       2.2s
 => [frontend-server jre-build 2/2] RUN /opt/java/openjdk/bin/jlink     --add-modules java.base,java.logging,j  15.8s
 => [data-analyzer internal] load build context                                                            2.6s
 => => transferring context: 51.83MB                                                                       2.6s
 => [data-collector stage-1 3/7] RUN addgroup -g 1001 kotlin && adduser -S -D -u 1001 -G kotlin kotlin     0.4s
 => [data-analyzer stage-1 4/7] COPY --from=jre-build /javaruntime /opt/java/openjdk                       0.3s
 => [frontend-server stage-1 5/7] WORKDIR /app                                                             0.1s
 => [data-analyzer stage-1 6/7] RUN mkdir -p /opt/applications && chown kotlin:kotlin /opt/applications    0.3s
 => [data-analyzer stage-1 7/7] COPY applications/data-analyzer-server/build/libs/*.jar /opt/applications/app.j  0.7s
 => [frontend-server stage-1 7/7] COPY applications/frontend-server/build/libs/*.jar /opt/applications/app.jar  0.7s
 => [data-collector stage-1 7/7] COPY applications/data-collector-server/build/libs/*.jar /opt/applications/app  0.7s
 => [data-analyzer] exporting to image                                                                     0.5s
 => => exporting layers                                                                                    0.5s
 => => writing image sha256:58b57abd6854f0d6d7adce9dda81c7d3e53b519a0feff708a8a9a9975b0e6c71               0.0s
 => => naming to docker.io/library/csca-5028-sentiment-analysis-data-analyzer                              0.0s
 => [data-collector] exporting to image                                                                    0.5s
 => => exporting layers                                                                                    0.5s
 => => writing image sha256:7b309095b1f765b0d4fe8e49e2112086dd18ccfd9cbb55f9b93ed7b66e42cbce               0.0s
 => => naming to docker.io/library/csca-5028-sentiment-analysis-data-collector                             0.0s
 => [frontend-server] exporting to image                                                                   0.5s
 => => exporting layers                                                                                    0.5s
 => => writing image sha256:51be3fb795cccec14366baff2c86480cac8b3dcb10e4cd842125a17be57e9765               0.0s
 => => naming to docker.io/library/csca-5028-sentiment-analysis-frontend-server                            0.0s
superman:csca-5028-sentiment-analysis ameau$ docker compose up -d
[+] Running 7/8
 :. Network csca-5028-sentiment-analysis_default  Created                                                 24.6s
 ✔ Container rabbitmq                             Healthy                                                 13.7s
 ✔ Container mongodb                              Healthy                                                 24.2s
 ✔ Container prometheus                           Started                                                  1.6s
 ✔ Container grafana                              Started                                                  2.2s
 ✔ Container sentiment-data-collector             Started                                                  2.3s
 ✔ Container sentiment-analyzer-frontend          Started                                                  2.4s
 ✔ Container sentiment-analyzer                   Started                                                 24.2s
superman:csca-5028-sentiment-analysis ameau$ sh scripts/post-collect.sh
{"fetchedCount":100,"upsertedCount":100}superman:csca-5028-sentiment-analysis ameau$
superman:csca-5028-sentiment-analysis ameau$ sh scripts/post-publish.sh
{"dbCount":100,"msgCount":245}superman:csca-5028-sentiment-analysis ameau$
```



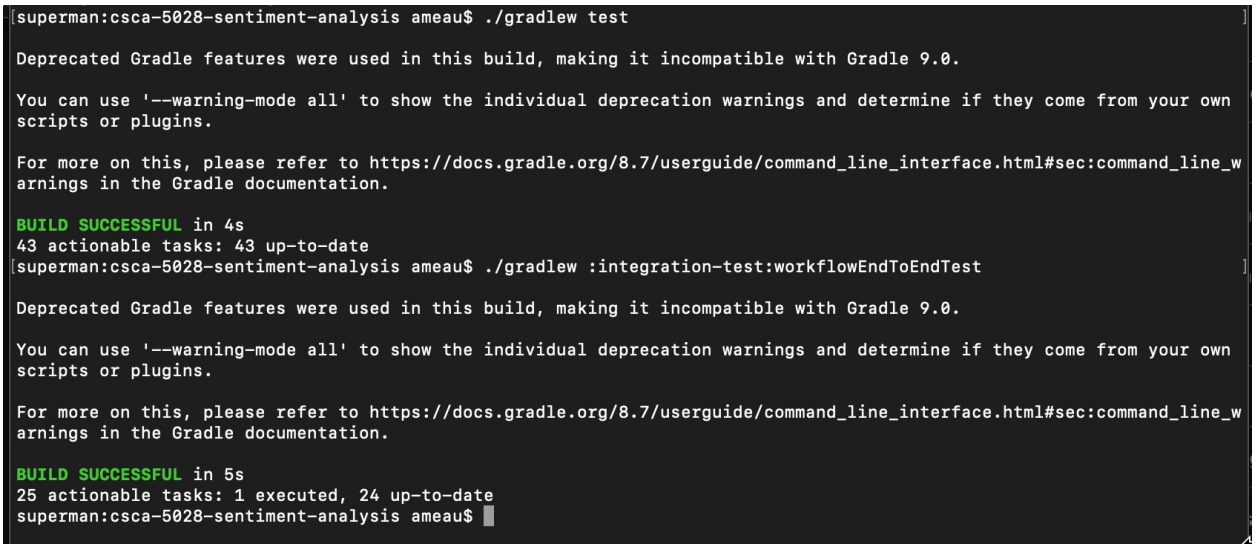
Unit Test and Integration Test

- Unit Test are executed as part of build process, or using the *command*.

  > ./gradlew test

- Assuming source code are built and all the docker containers are running, then execute the gradlew integration test

  > ./gradlew :integration-test:workflowEndToEndTest



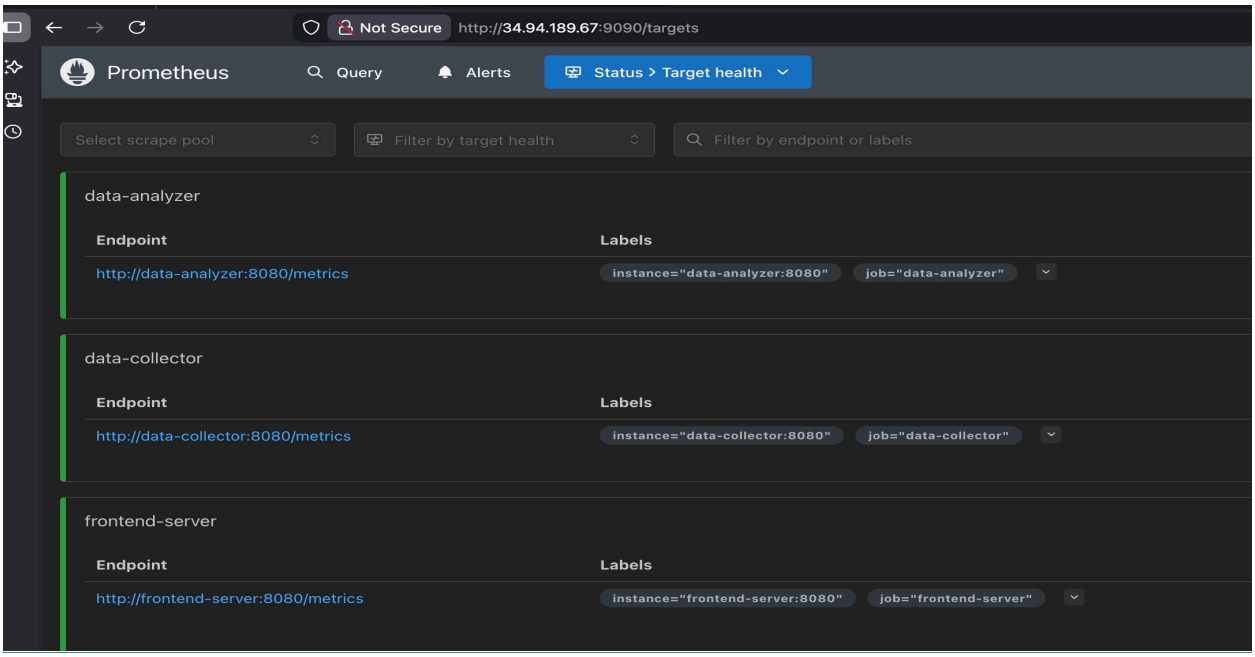
Product environment

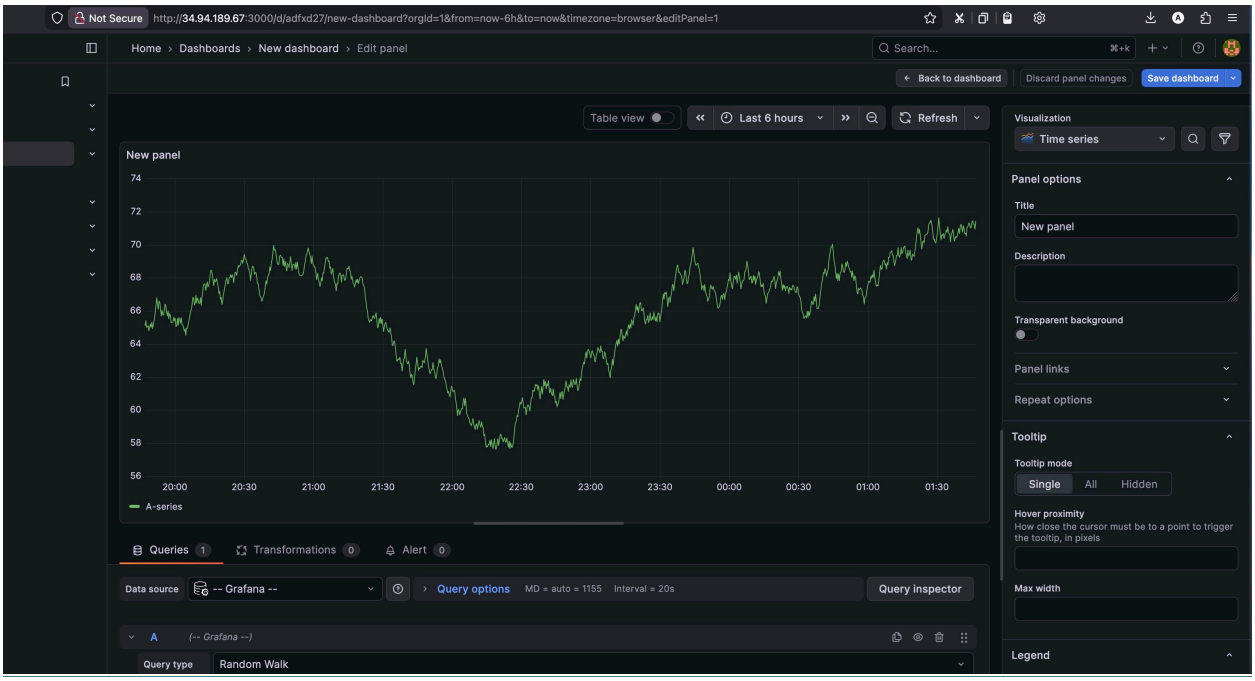
- Project has been deployed to Google cloud.
- Web form and report can be accessed http://34.94.189.67/

(Note: ONLY HTTP protocol is supported)

Montoring

- Promethsus can be used to track service health: http://localhost:9090/targets
- Granfana can be login using (admin / password): http://localhost:3000


Not Secure
http://34.94.189.67:9090/targets
Prometheus
Query
Alerts
Status > Target health
Select scrape pool
Filter by target health
Filter by endpoint or labels
data-analyzer
Endpoint
Labels
http://data-analyzer:8080/metrics
instance="data-analyzer:8080"
job="data-analyzer"
data-collector
Endpoint
Labels
http://data-collector:8080/metrics
instance="data-collector:8080"
job="data-collector"
frontend-server
Endpoint
Labels
http://frontend-server:8080/metrics
instance="frontend-server:8080"
job="frontend-server"


Not Secure
http://34.94.189.67:3000/d/adfxd27/new-dashboard?orgId=1&from=now-6h&to=now&timezone=browser&editPanel=1
Home
Dashboards
New dashboard
Edit panel
Search...
Back to dashboard
Discard panel changes
Save dashboard
Table view
Last 6 hours
Refresh
New panel
74
72
70
68
66
64
62
60
58
56
20:00
20:30
21:00
21:30
22:00
22:30
23:00
23:30
00:00
00:30
01:00
01:30
A-series
Queries 1
Transformations 0
Alert 0
Data source
-- Grafana --
Query options
MD = auto = 1155
Interval = 20s
Query inspector
A
(-- Grafana --)
Query type
Random Walk
Visualization
Time series
Panel options
Title
New panel
Description
Transparent background
Panel links
Repeat options
Tooltip
Tooltip mode
Single
All
Hidden
Hover proximity
How close the cursor must be to a point to trigger the tooltip, in pixels
Max width
Legend