**Final Project**

**Problem Statement:**

**Background:**

A popular library ("Amazing Books") in the city is planning to launch a portal for expanding their reach. They hire you to help them, go online. They want to proceed with incremental but fast-paced releases to attract more readers. They have a few major functions which should be independently developed, tested, and released from other functions to enable fast-paced releases. Apart from this, they expect the user base to grow soon and thus must be highly scalable without compromising on availability, resiliency, and security.

The portal would need to support different types of devices eg. web, mobile, tablet, and thus would require RESTful APIs which you need to build to manage issuing of books to the readers. Development of GUI (Graphical User Interface) would be taken up by another team and the delivery of the books will be handled by another vendor.

So, there are three major functions that need to be implemented:

1. APIs to fetch/add/edit/delete books (H2 Database).

2. APIs to fetch/add/cancel issuing of the books to the customers (H2 Database).

3. Customers to be managed by OAuth2 server(in-memory)

Apart from these functional requirements, you would also have to take care of non-functional requirements:

1. It should be easy to scale.

2. It must be highly secured.

3. It should be resilient.

4. It should be performant.

5. It should support quick releases.

Considering the above-mentioned requirements (functional + non-functional), Microservices Architecture seems fit for this. Before the implementation do provide High Level Design.

Below is a list of important functions(microservices) and the model which you may consider for the implementation:

1. Book(bookms): fetch/add/edit/delete books

Model: isbn, title, publishedDate, totalCopies, issuedCopies, author Assumption: A book will have only one author

2. Issue books to customers(issuerms)

- Fetch(REST call) Book Details eg bookId and availableCopies(totalCopies - issuedCopies)

- Issue the book to the customer, if the book is available

- Update issuedCopies in bookms - To keep things simple, these operations may not necessarily be atomic

Model: isbn, custId, noOfCopies Note: A book(isbn) can be assigned to more than one customer

3. A service for dynamic service discovery

4. A service for handling cross-cutting concerns

5. A service for managing security

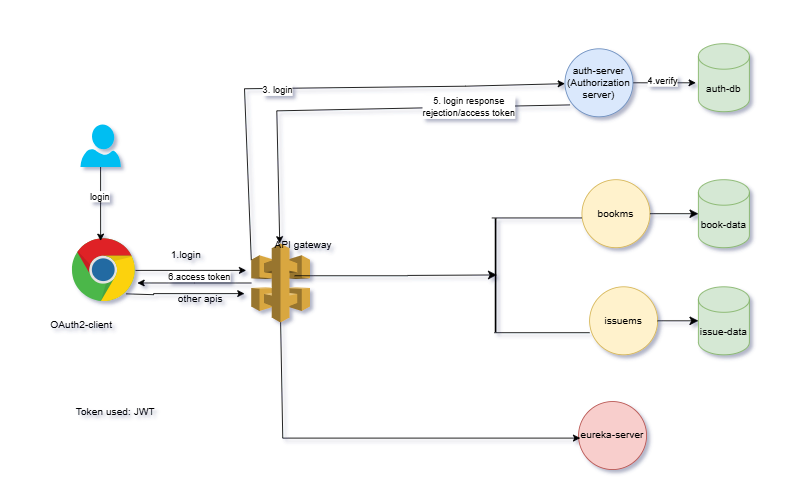
- Have one client application registered(in-memory)

- Add customers as in-memory users

**Solution:**

Code is available in **https://github.com/kalaivaniselvam/Edureka.git**

**High Level Design:**

****

**Microservice overview:**

**Booms:**

The CRUD operations are handled and also H2 database will be used for storage

**Issuems:**

This microservice will issue books to customers. The API will be invoked from here to bookms to fetch book details.

**OAuth2 Authorization server:**

Authentication and Authorization will be handled here. It stores customers and client app in-memory.

**Eurekams (Service Discovery):**

The microservices (issuems and bookms) will be registered here.

**API gateway(auth-server):**

It handles cross cutting concerns like routing, logging, monitoring etc.

Eureka server:

https://localhost:8761

A screenshot of a computer

AI-generated content may be incorrect.

**API sample request and response:**

**Bookms:**

**http://localhost:8082/swagger-ui/index.html**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Issuems:**

**http://localhost:8083/swagger-ui/index.html**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Bookms:**

**Add a book:**

**API:** http://localhost:8082/booksInventory/addBook

{

"id": 1,

"isbn": "123e",

"title": "Siragugal",

"publishedDate": "10/10/2000",

"totalCopies": 200,

"issuedCopies": 150,

"author": "KALAM",

"price": 250

}

**A screenshot of a computer

AI-generated content may be incorrect.**

**A black and white rectangular object

AI-generated content may be incorrect.**

**Update book details:**

<http://localhost:8082/booksInventory/updateBook/1>

Updating the issued copies:

{

"id": 1,

"isbn": "123e",

"title": “Kids Stories”,

"publishedDate": 10/12/2001,

"totalCopies": 200,

"issuedCopies": 2,

"author": "KALAM",

"price": 250

}

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**View all books:**

<http://localhost:8082/booksInventory/AllBooks>

A black and white rectangle

AI-generated content may be incorrect.

**View single book:**

http://localhost:8082/booksInventory/individualBook/1

A screenshot of a computer

AI-generated content may be incorrect.

Delete the book details:

http://localhost:8082/booksInventory/delete/1

A screenshot of a computer

AI-generated content may be incorrect.

**Issuems**:

Calling bookms from issuems service to fetch the books details. After that, the issued copies will be updated in bookms.

A screenshot of a computer

AI-generated content may be incorrect.

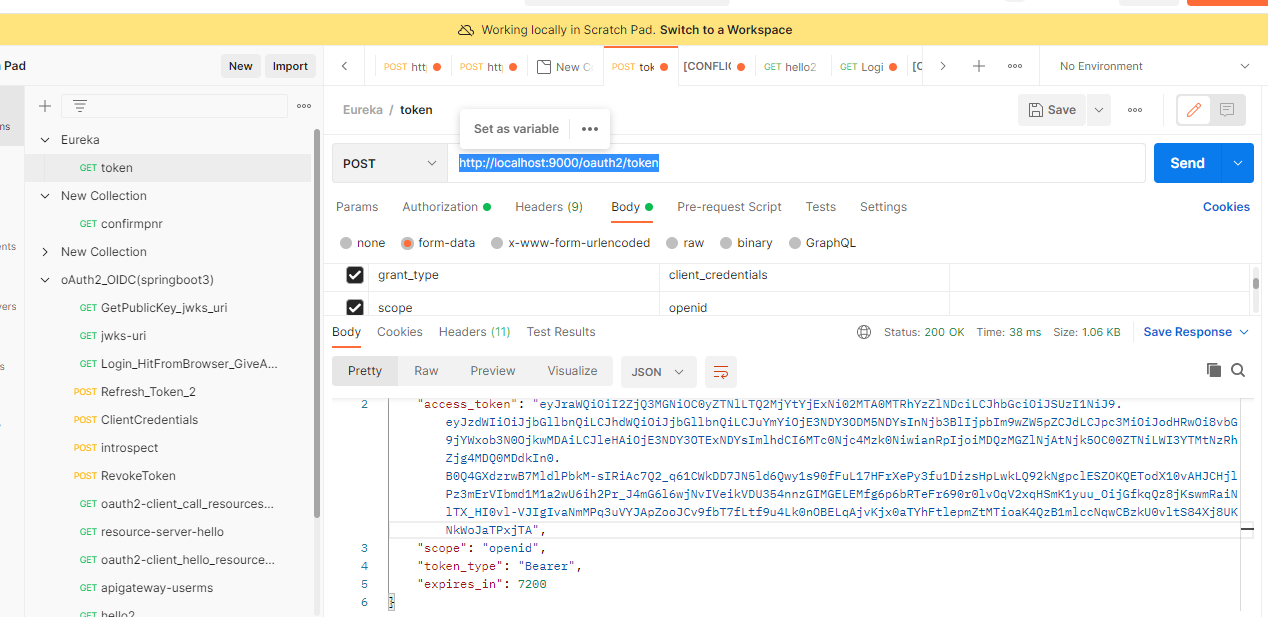
A screenshot of a computer

AI-generated content may be incorrect.

**Auth-Server:**

<http://localhost:9000/oauth2/token>

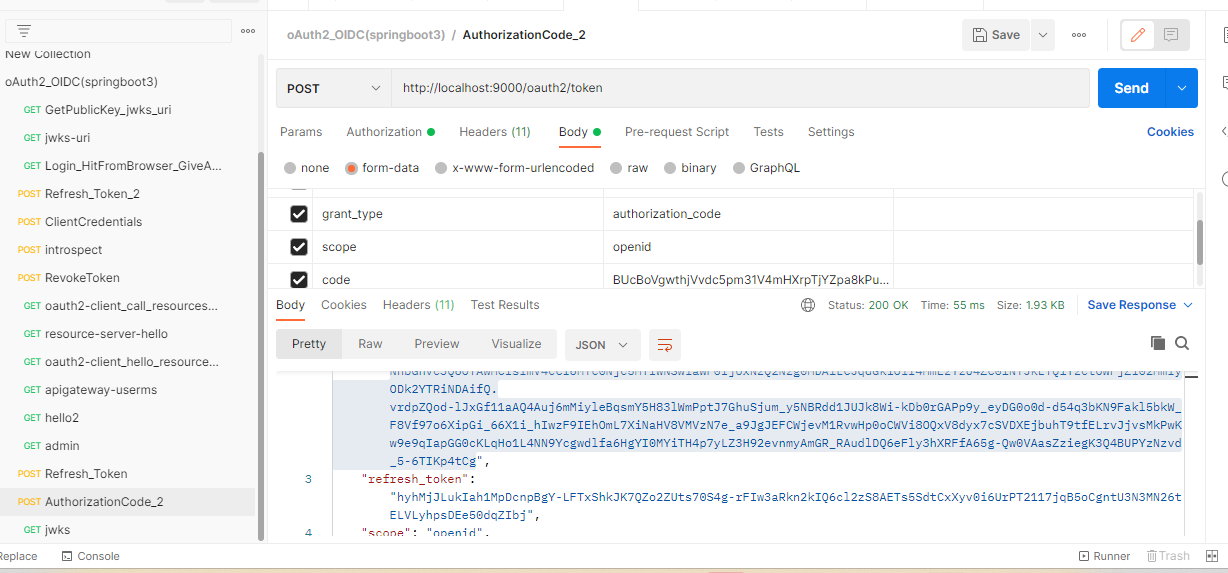
grant\_type = client\_credentials



<http://localhost:9000/oauth2/token>

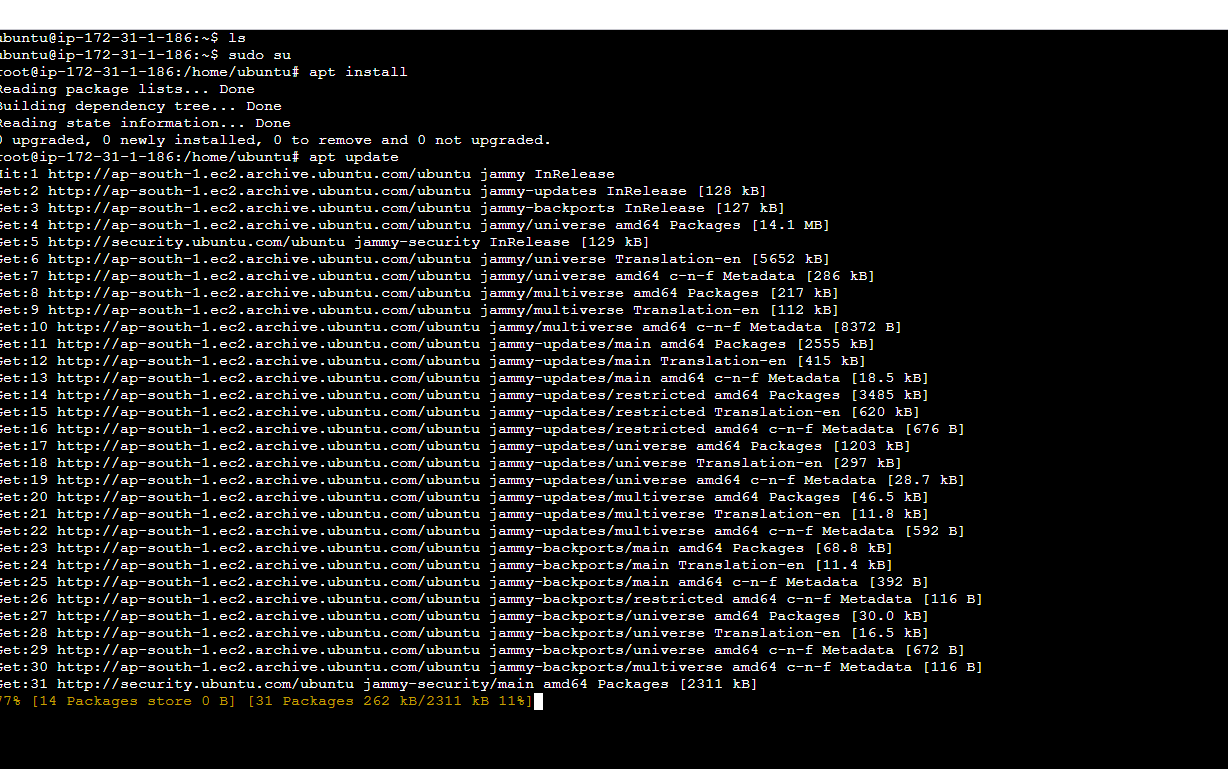
code : BUcBoVgwthjVvdc5pm31V4mHXrpTjYZpa8kPu0vrPfVSpNpuSkO\_aveAJqkq5OMGhdllE\_5PC50BADc\_FNCHIpKqghq6IeeMlbpg

grant-type = authorization\_code



**Deploying docker and microservice into AWS:**

Apt update



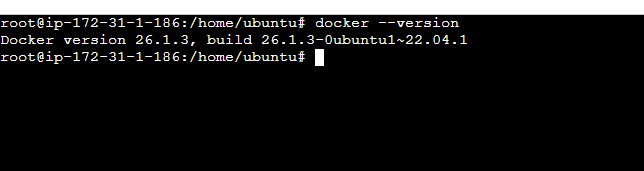
Docker install:

Apt install docker.io

A screenshot of a computer program

AI-generated content may be incorrect.

Docker version



Docker images

A black screen with white text

AI-generated content may be incorrect.

Uploading the microservice jar to aws using git bash



Docker and jar are in aws

A screenshot of a computer

AI-generated content may be incorrect.

Build the image

A computer screen with white text

AI-generated content may be incorrect.

Docker images

A screen shot of a computer

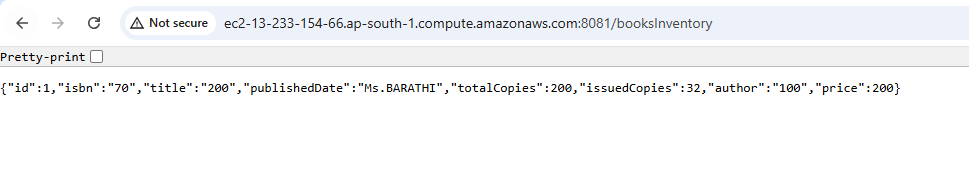
AI-generated content may be incorrect.

Docker run

docker run -d -p 8081:8081 -t bookms:1.0



AWS instance output:



Swagger url from aws:

A screenshot of a computer

AI-generated content may be incorrect.

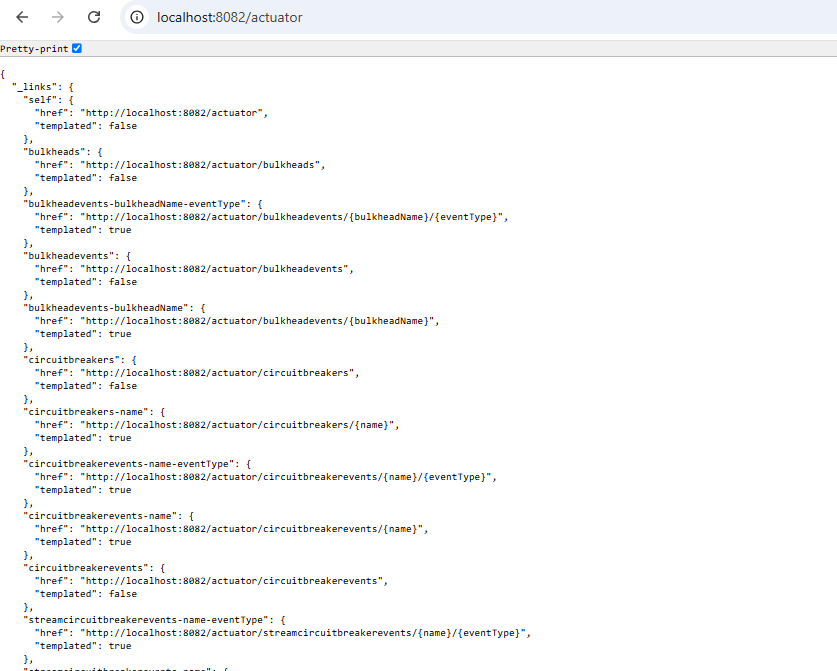
Peer:

A screenshot of a computer

AI-generated content may be incorrect.

Prometheus



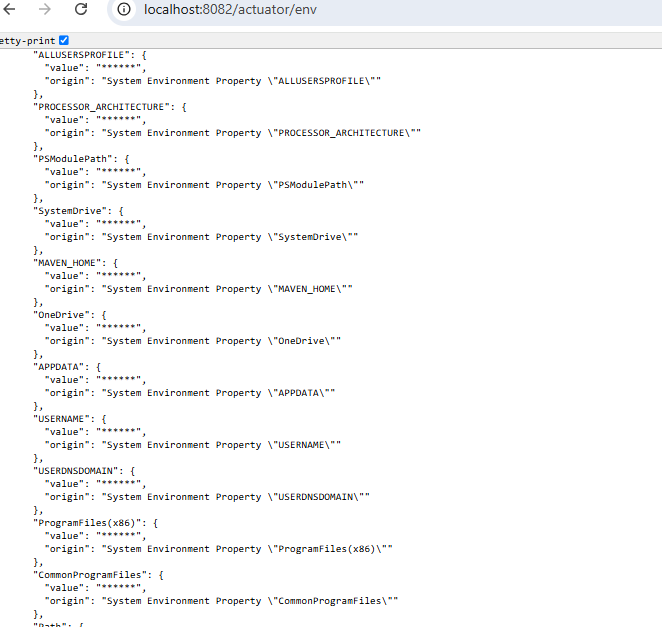


Health check:

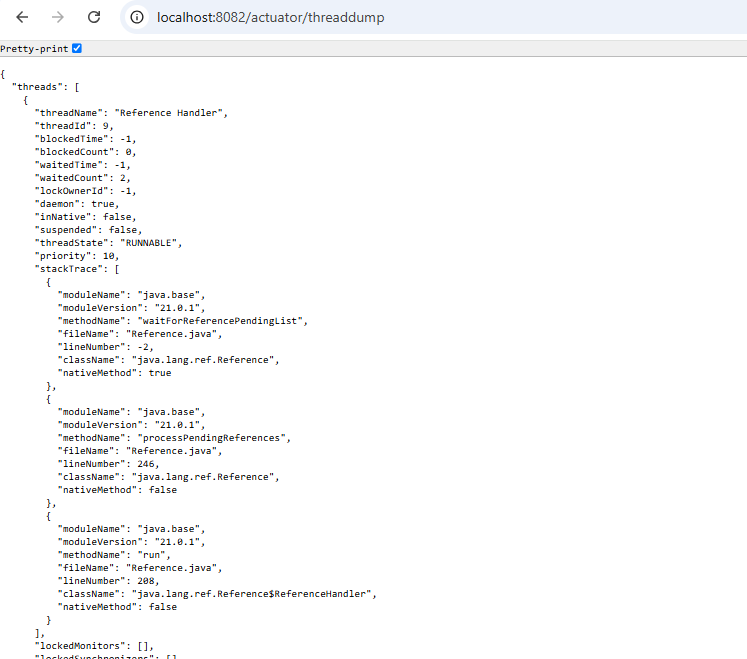
A screenshot of a computer screen

AI-generated content may be incorrect.

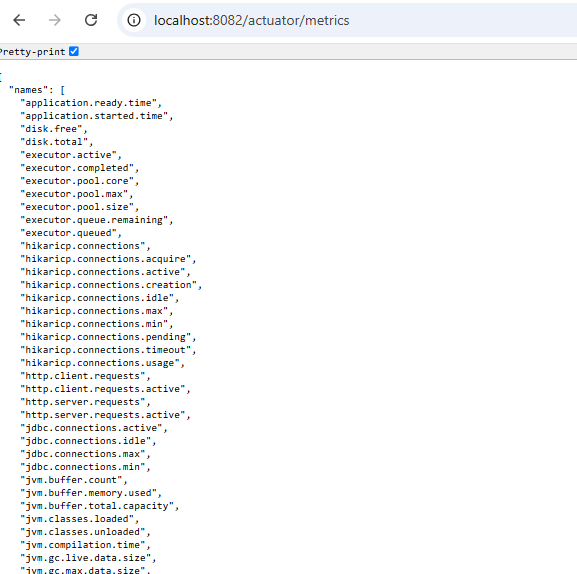
Environmental details:



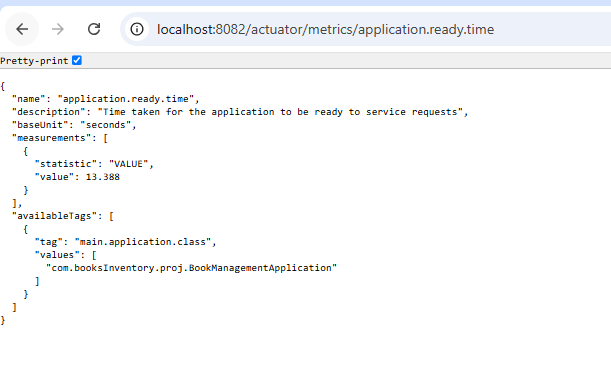
Thread dump:



Metrics:



Application ready time:

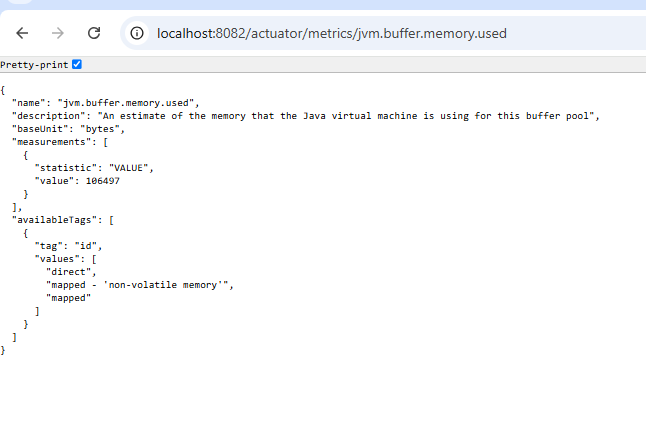


Application started time:

A screenshot of a computer program

AI-generated content may be incorrect.

Memory used:



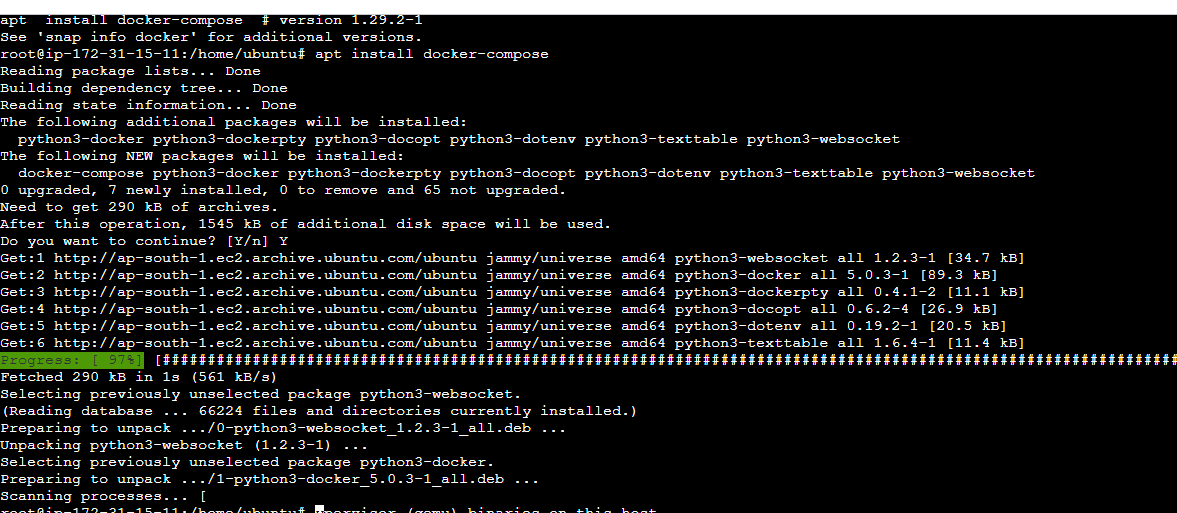
Premetheus view:

A screenshot of a computer

AI-generated content may be incorrect.

Deploying microservice, Prometheus and graphana into the aws using Docker compose:

Apt install docker-compose:



All dokcer-compose done

docker-compose up -d

A screenshot of a computer

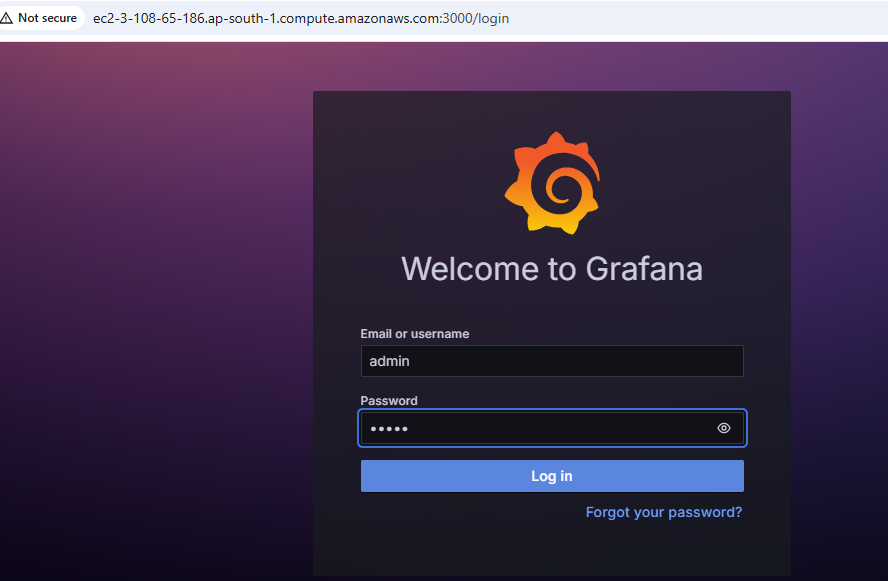
AI-generated content may be incorrect.

3 containers are running

A black screen with white text

AI-generated content may be incorrect.

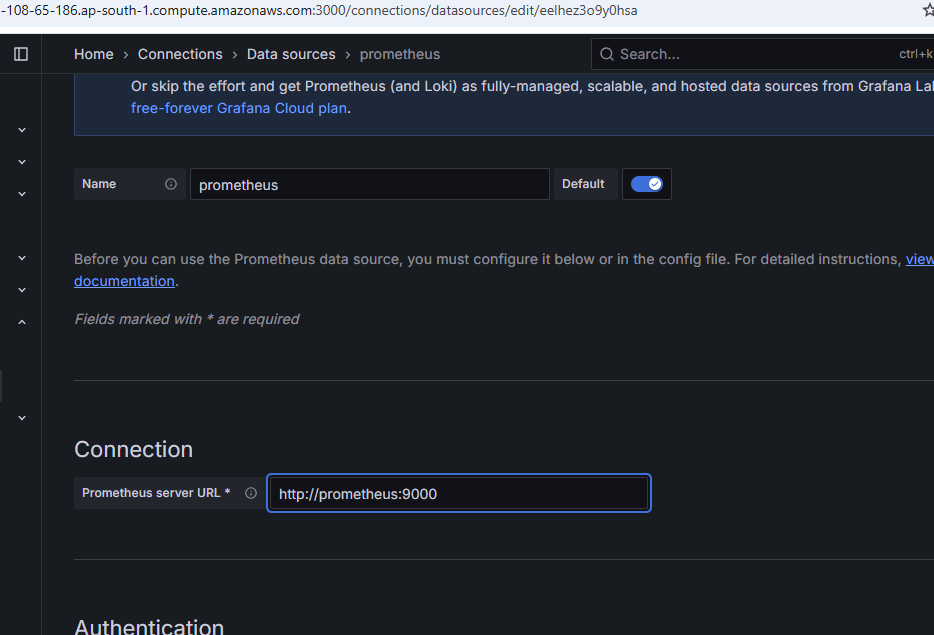
Graphana:



A screenshot of a computer

AI-generated content may be incorrect.

Add new data source:



A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

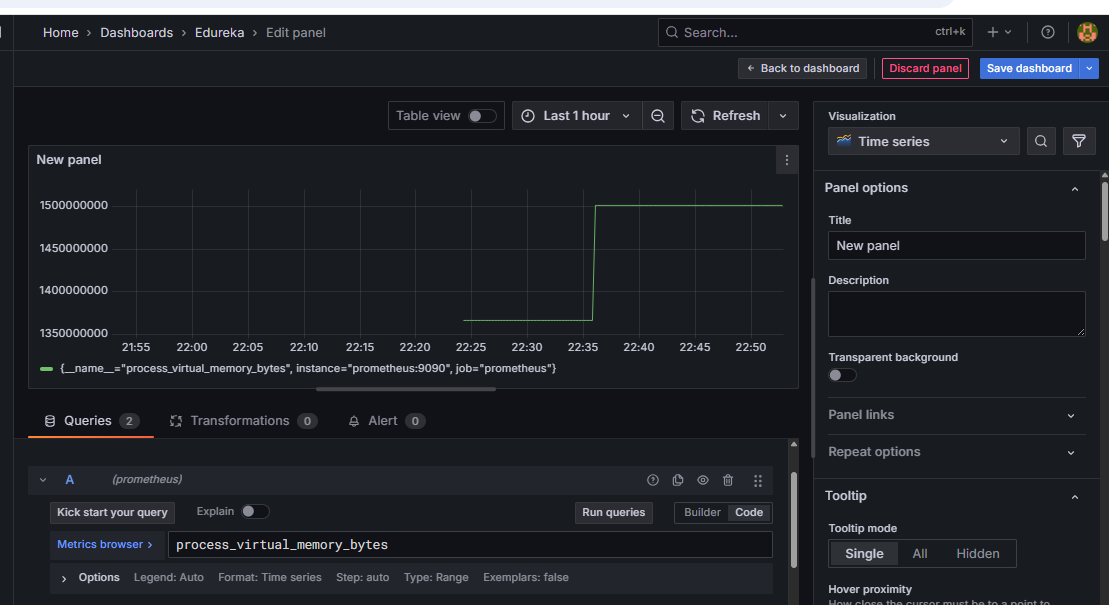
AI-generated content may be incorrect.

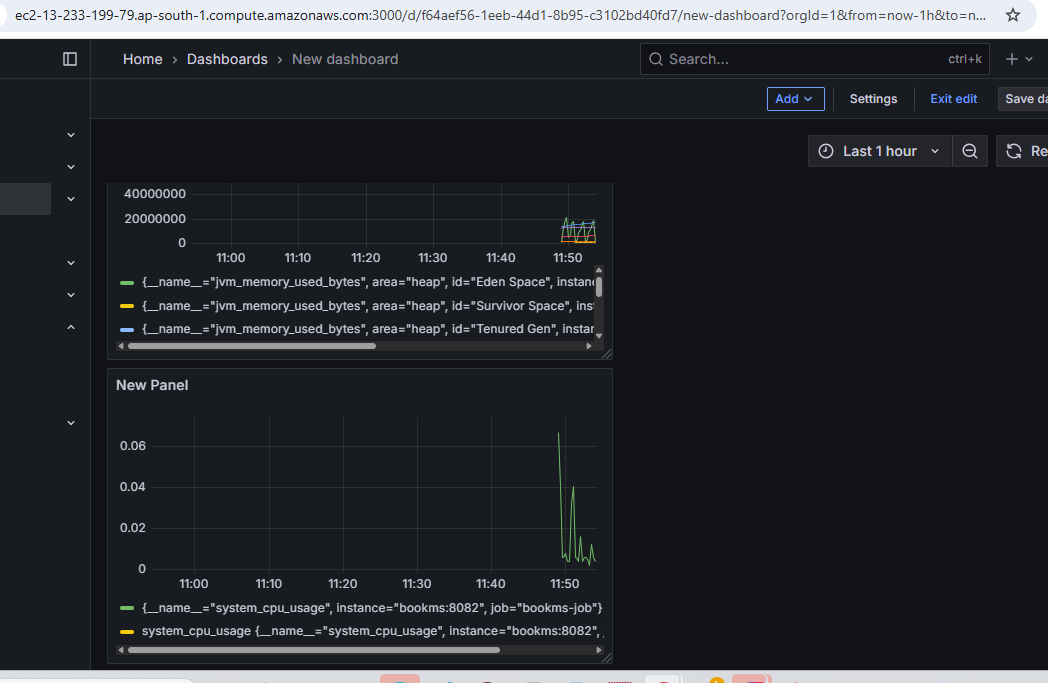
A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.





A screenshot of a computer

AI-generated content may be incorrect.

Prometheus from aws instance:

A screenshot of a computer

AI-generated content may be incorrect.

**Improvements done:**

1. Implemented load balancer and circuit breaker.
2. Implemented zipkin for the peer server.
3. Implemented swagger to view the results
4. Used docker container
5. Used Prometheus and Grafana for monitoring the servers