Creating MySQL DB containers for microservices

Remove these dependencies from configserver microservice  
D:\Experiments\Microservices\sb-bank-application\configserver\pom.xml  
<dependency>

            <groupId>org.springframework.cloud</groupId>

            <artifactId>spring-cloud-starter-bus-amqp</artifactId>

</dependency>

<dependency>

            <groupId>org.springframework.cloud</groupId>

            <artifactId>spring-cloud-config-monitor</artifactId>

            <version>4.1.2</version>

</dependency>

Remove these properties from configserver  
D:\Experiments\Microservices\sb-bank-application\configserver\src\main\resources\application.properties  
spring.cloud.rabbitmq.host=localhost

spring.cloud.rabbitmq.port=5672

spring.cloud.rabbitmq.username=guest

spring.cloud.rabbitmq.password=guest

Delete this dependency from accounts microservice

D:\Experiments\Microservices\sb-bank-application\accounts\pom.xml

<dependency>

            <groupId>org.springframework.cloud</groupId>

            <artifactId>spring-cloud-starter-bus-amqp</artifactId>

</dependency>

Remove these properties from the accounts  
D:\Experiments\Microservices\sb-bank-application\accounts\src\main\resources\application.properties

spring.rabbitmq.host=localhost

spring.rabbitmq.port=5672

spring.rabbitmq.username=guest

spring.rabbitmq.password=guest

Delete this dependency from loans microservice

D:\Experiments\Microservices\sb-bank-application\loans\pom.xml

<dependency>

            <groupId>org.springframework.cloud</groupId>

            <artifactId>spring-cloud-starter-bus-amqp</artifactId>

</dependency>

Remove these properties from the loans  
D:\Experiments\Microservices\sb-bank-application\loans\src\main\resources\application.properties

spring.rabbitmq.host=localhost

spring.rabbitmq.port=5672

spring.rabbitmq.username=guest

spring.rabbitmq.password=guest

Delete this dependency from cards microservice

D:\Experiments\Microservices\sb-bank-application\cards\pom.xml

<dependency>

            <groupId>org.springframework.cloud</groupId>

            <artifactId>spring-cloud-starter-bus-amqp</artifactId>

</dependency>

Remove these properties from the cards  
D:\Experiments\Microservices\sb-bank-application\cards\src\main\resources\application.properties

spring.rabbitmq.host=localhost

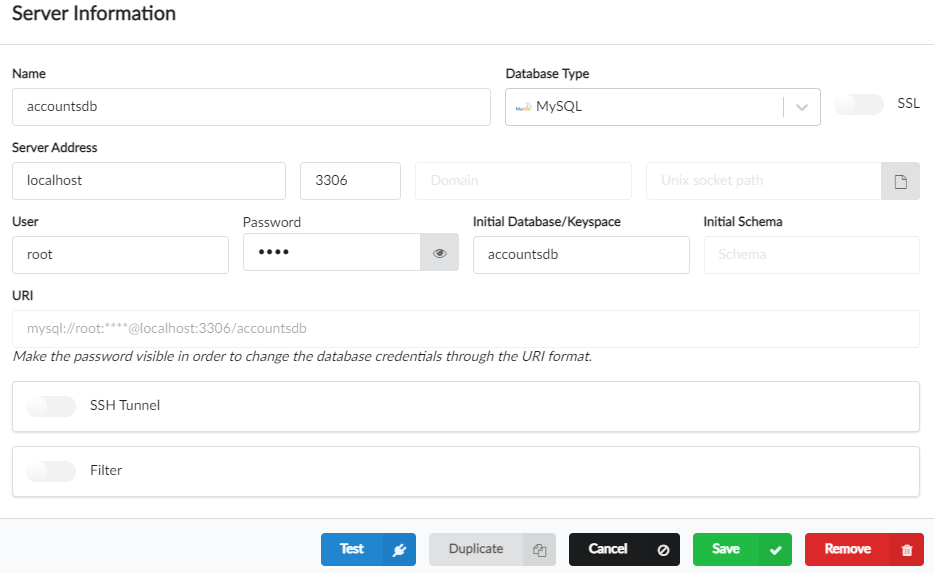
spring.rabbitmq.port=5672

spring.rabbitmq.username=guest

spring.rabbitmq.password=guest

Starting the MySQL docker container in local machine

PS D:\> docker run -p 3306:3306 --name accountsdb -e MYSQL\_ROOT\_PASSWORD=root -e MYSQL\_DATABASE=accountsdb -d mysql

To connect to the DB use a very super light client named SQLECTRON  
https://sqlectron.github.io/  
  


This database is empty does not have any tables.

Similarly create database for loans and cards microservice

PS D:\> docker run -p 3307:3306 --name loansdb -e MYSQL\_ROOT\_PASSWORD=root -e MYSQL\_DATABASE=loansdb -d mysql

PS D:\> docker run -p 3308:3306 --name cardsdb -e MYSQL\_ROOT\_PASSWORD=root -e MYSQL\_DATABASE=cardsdb -d mysql

Note: Restart the Sqlectron after creating the database for successful connection

Update microservices code to replace H2 DB with MySQL DB

Delete H2 related dependency from accounts, loans, and cards microservice

<dependency>

        <groupId>com.h2database</groupId>

        <artifactId>h2</artifactId>

        <scope>runtime</scope>

</dependency>

Add MySQL related dependency from accounts, loans, and cards microservice

<dependency>

        <groupId>com.mysql</groupId>

        <artifactId>mysql-connector-j</artifactId>

<scope>runtime</scope>

</dependency>

Modify all the H2 configurations with MySQL configurations in accounts, loans, and cards microservice

D:\Experiments\Microservices\sb-bank-application\accounts\src\main\resources\application.properties

….

spring.datasource.url=jdbc:mysql://localhost:3306/accountsdb

~~# spring.datasource.driver-class-name=org.h2.Driver~~

spring.datasource.username=root

spring.datasource.password=root

~~# spring.jpa.database-platform=org.hibernate.dialect.H2Dialect~~

~~# spring.jpa.hibernate.ddl-auto=update~~

spring.jpa.show-sql=true

spring.sql.init.mode=always

~~# spring.h2.console.enabled=true~~

…

D:\Experiments\Microservices\sb-bank-application\loans\src\main\resources\application.properties

…

spring.datasource.url=jdbc:mysql://localhost:3307/loansdb

~~# spring.datasource.driver-class-name=org.h2.Driver~~

spring.datasource.username=root

spring.datasource.password=root

~~# spring.jpa.database-platform=org.hibernate.dialect.H2Dialect~~

~~# spring.jpa.hibernate.ddl-auto=update~~

spring.jpa.show-sql=true

spring.sql.init.mode=always

~~# spring.h2.console.enabled=true~~

…

D:\Experiments\Microservices\sb-bank-application\cards\src\main\resources\application.properties

…

spring.datasource.url=jdbc:mysql://localhost:3308/cardsdb

~~# spring.datasource.driver-class-name=org.h2.Driver~~

spring.datasource.username=root

spring.datasource.password=root

~~# spring.jpa.database-platform=org.hibernate.dialect.H2Dialect~~

~~# spring.jpa.hibernate.ddl-auto=update~~

spring.jpa.show-sql=true

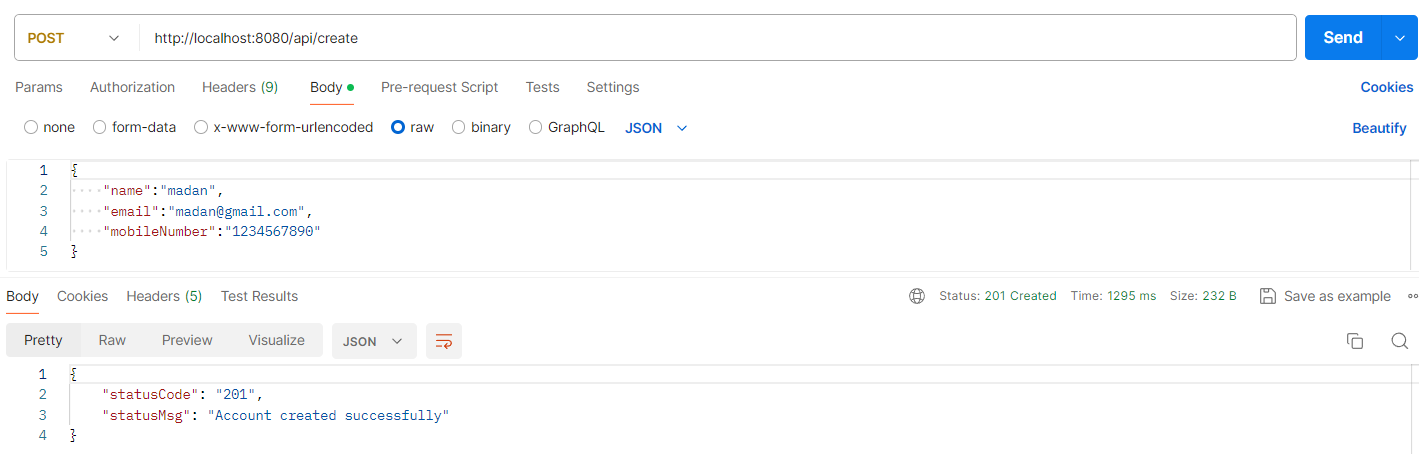
spring.sql.init.mode=always  
~~# spring.h2.console.enabled=true~~

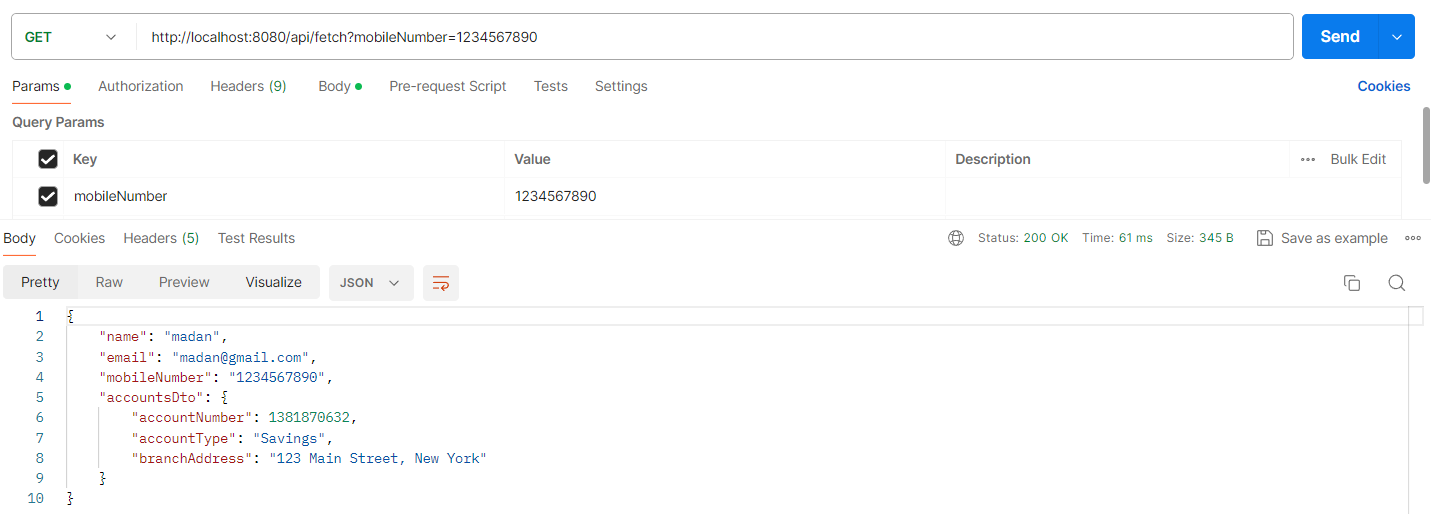
….

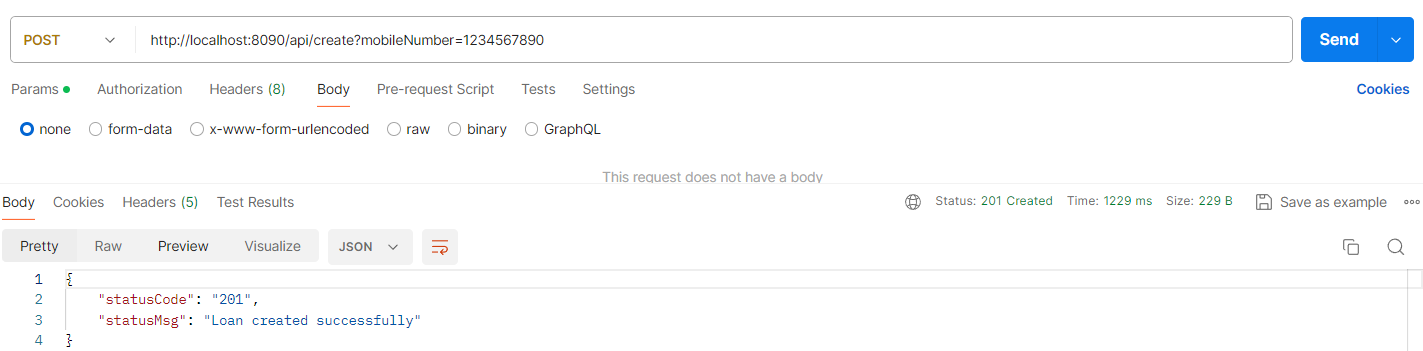
Start the config server  
Start accounts, loans, and cards microservice

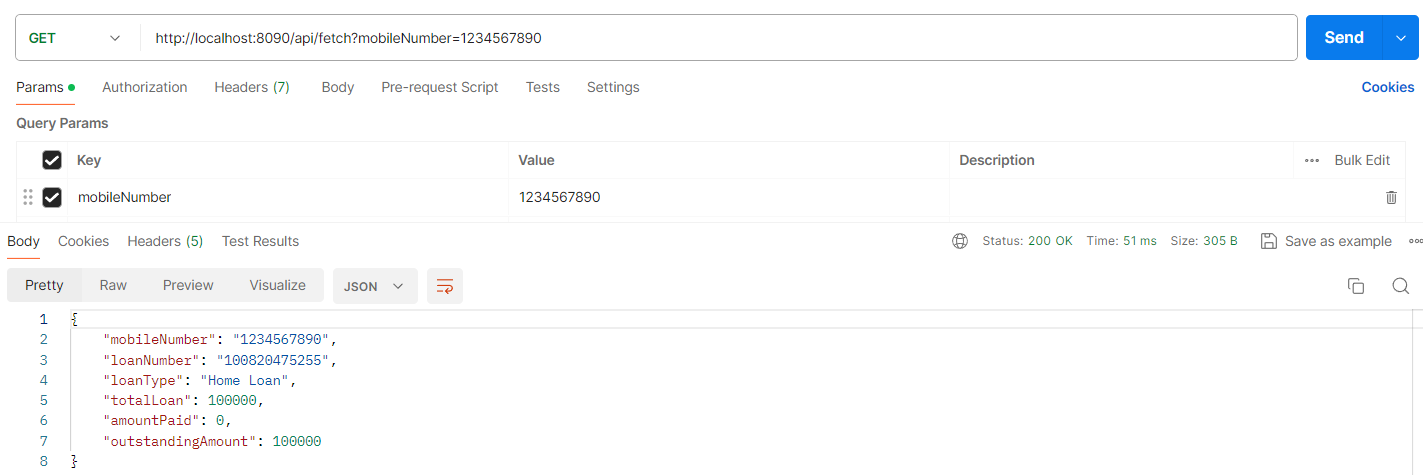
There will be tables created for each of the microservices on starting the service.

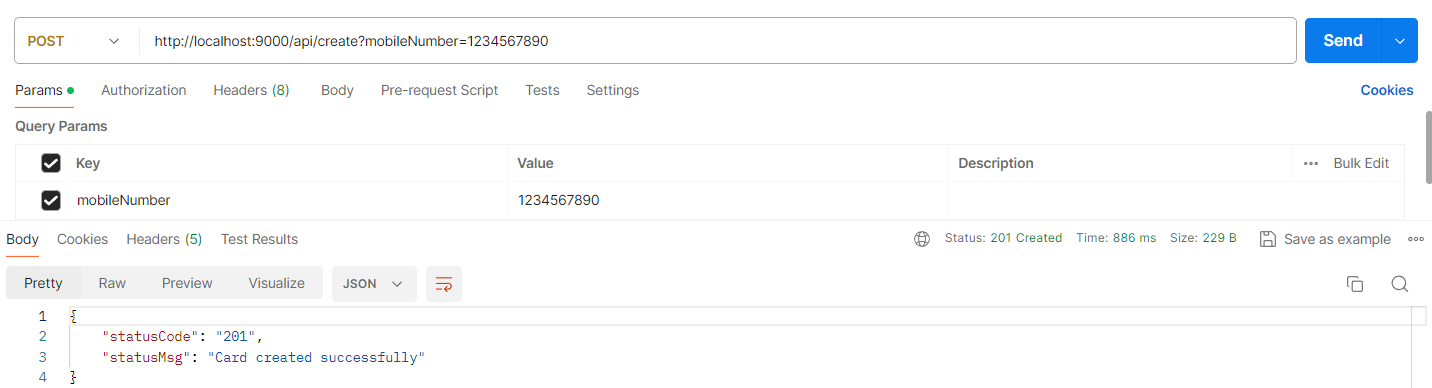
Test the databases using postman

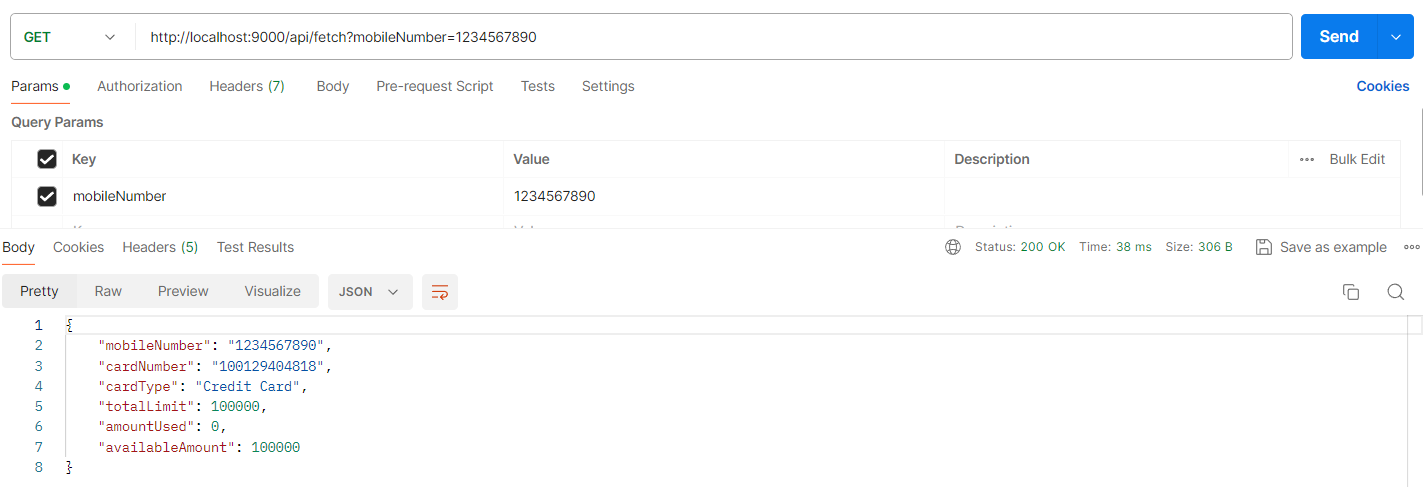


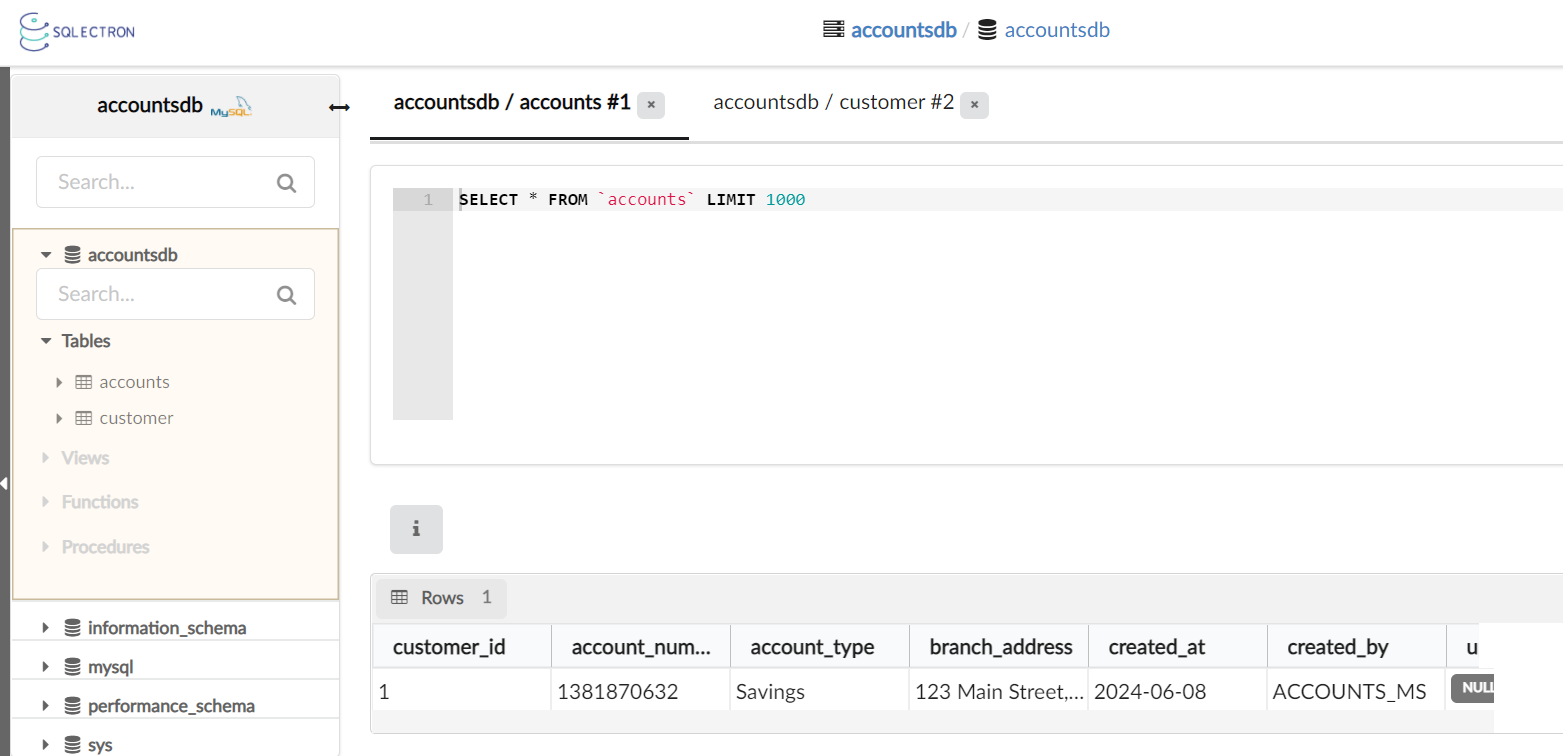












If we stop the container, data will not be deleted, but if we delete the container, all the data will be lost.

Update docker compose file to create and use MySQL DB

Mandatory plugin:   
<plugin>

        <groupId>com.google.cloud.tools</groupId>

        <artifactId>jib-maven-plugin</artifactId>

        <version>3.4.1</version>

        <configuration>

              <to>

                <image>nileshzarkar/${project.artifactId}:s7</image>

              </to>

           </configuration>

</plugin>

PS D:\Experiments\Microservices\sb-bank-application\accounts> mvn compile jib:dockerBuild

PS D:\Experiments\Microservices\sb-bank-application\cards> mvn compile jib:dockerBuild

PS D:\Experiments\Microservices\sb-bank-application\loans> mvn compile jib:dockerBuild

PS D:\Experiments\Microservices\sb-bank-application\configserver> mvn compile jib:dockerBuild

Push the images to docker hub

PS D:\Experiments\Microservices\sb-bank-application\docker-compose> docker image push docker.io/nileshzarkar/accounts:s7

PS D:\Experiments\Microservices\sb-bank-application\docker-compose> docker image push docker.io/nileshzarkar/loans:s7

PS D:\Experiments\Microservices\sb-bank-application\docker-compose> docker image push docker.io/nileshzarkar/cards:s7

PS D:\Experiments\Microservices\sb-bank-application\docker-compose> docker image push docker.io/nileshzarkar/configserver:s7

Delete the local docker cache images

D:\Experiments\Microservices\sb-bank-application\docker-compose\default\docker-compose.yaml

Remove all rabbitmq related changes from docker-compose

….

rabbit:

    image: rabbitmq:3.12-management

    hostname: rabbitmq

    ports:

      - "5672:5672"

      - "15672:15672"

    healthcheck:

      test: rabbitmq-diagnostics check\_port\_connectivity

      interval: 10s

      timeout: 5s

      retries: 10

      start\_period: 5s

    extends:

      file: common-config.yaml

      service: network-deploy-service

…..

    depends\_on:

      rabbit:

        condition: service\_healthy

….

D:\Experiments\Microservices\sb-bank-application\docker-compose\default\common-config.yaml

D:\Experiments\Microservices\sb-bank-application\docker-compose\default\docker-compose.yaml

Running microservices & MySQL DB containers using docker compose file

D:\Experiments\Microservices\sb-bank-application\docker-compose\default\common-config.yaml

services:

  network-deploy-service:

    networks:

      - eazybank

  microservice-db-config:

    extends:

      service: network-deploy-service

    image: mysql

    healthcheck:

      test: [ "CMD", "mysqladmin" ,"ping", "-h", "localhost" ]

      timeout: 10s

      retries: 10

      interval: 10s

      start\_period: 10s

    environment:

      MYSQL\_ROOT\_PASSWORD: root

  microservice-base-config:

    extends:

      service: network-deploy-service

    deploy:

      resources:

        limits:

          memory: 700m

  microservice-configserver-config:

    extends:

      service: microservice-base-config

    depends\_on:

      configserver:

        condition: service\_healthy

    environment:

      SPRING\_PROFILES\_ACTIVE: default

      SPRING\_CONFIG\_IMPORT: configserver:http://configserver:8071/

      SPRING\_DATASOURCE\_USERNAME: root

      SPRING\_DATASOURCE\_PASSWORD: root

D:\Experiments\Microservices\sb-bank-application\docker-compose\default\docker-compose.yaml

services:

  accountsdb:

    container\_name: accountsdb

    ports:

      - 3306:3306

    environment:

      MYSQL\_DATABASE: accountsdb

    extends:

      file: common-config.yaml

      service: microservice-db-config

  loansdb:

    container\_name: loansdb

    ports:

      - 3307:3306

    environment:

      MYSQL\_DATABASE: loansdb

    extends:

      file: common-config.yaml

      service: microservice-db-config

  cardsdb:

    container\_name: cardsdb

    ports:

      - 3308:3306

    environment:

      MYSQL\_DATABASE: cardsdb

    extends:

      file: common-config.yaml

      service: microservice-db-config

  configserver:

    image: "nileshzarkar/configserver:s7"

    container\_name: configserver-ms

    ports:

      - "8071:8071"

    healthcheck:

      test: "curl --fail --silent localhost:8071/actuator/health/readiness | grep UP || exit 1"

      interval: 10s

      timeout: 5s

      retries: 10

      start\_period: 10s

    extends:

      file: common-config.yaml

      service: microservice-base-config

  accounts:

    image: "nileshzarkar/accounts:s7"

    container\_name: accounts-ms

    ports:

      - "8080:8080"

    environment:

      SPRING\_APPLICATION\_NAME: "accounts"

      SPRING\_DATASOURCE\_URL: "jdbc:mysql://accountsdb:3306/accountsdb"

    depends\_on:

      accountsdb:

        condition: service\_healthy

    extends:

      file: common-config.yaml

      service: microservice-configserver-config

  loans:

    image: "nileshzarkar/loans:s7"

    container\_name: loans-ms

    ports:

      - "8090:8090"

    environment:

      SPRING\_APPLICATION\_NAME: "loans"

      SPRING\_DATASOURCE\_URL: "jdbc:mysql://loansdb:3306/loansdb"

    depends\_on:

      loansdb:

        condition: service\_healthy

    extends:

      file: common-config.yaml

      service: microservice-configserver-config

  cards:

    image: "nileshzarkar/cards:s7"

    container\_name: cards-ms

    ports:

      - "9000:9000"

    environment:

      SPRING\_APPLICATION\_NAME: "cards"

      SPRING\_DATASOURCE\_URL: "jdbc:mysql://cardsdb:3306/cardsdb"

    depends\_on:

      cardsdb:

        condition: service\_healthy

    extends:

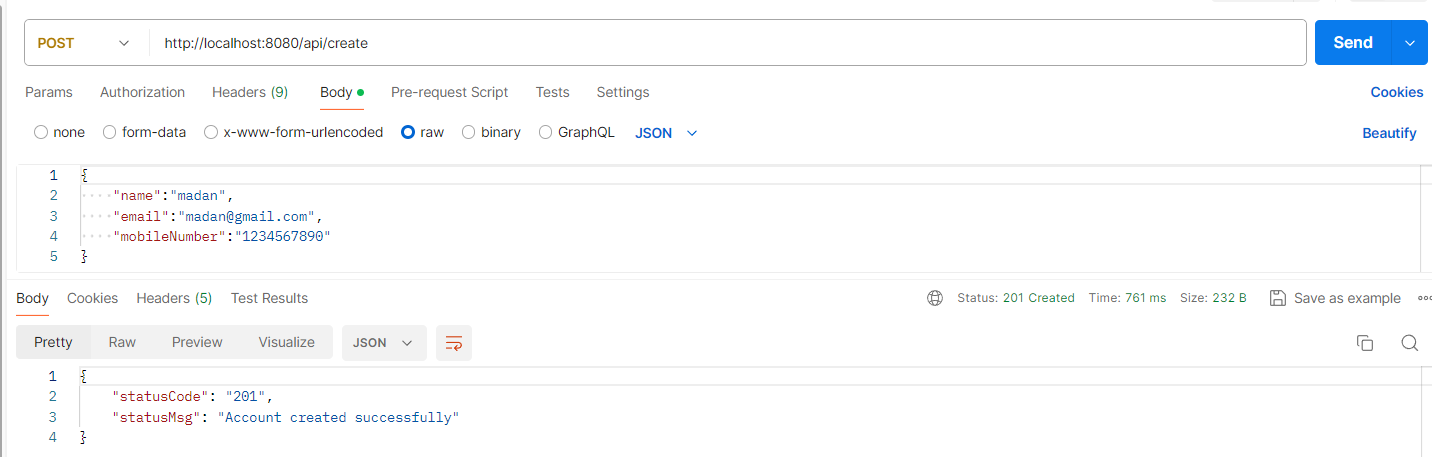
      file: common-config.yaml

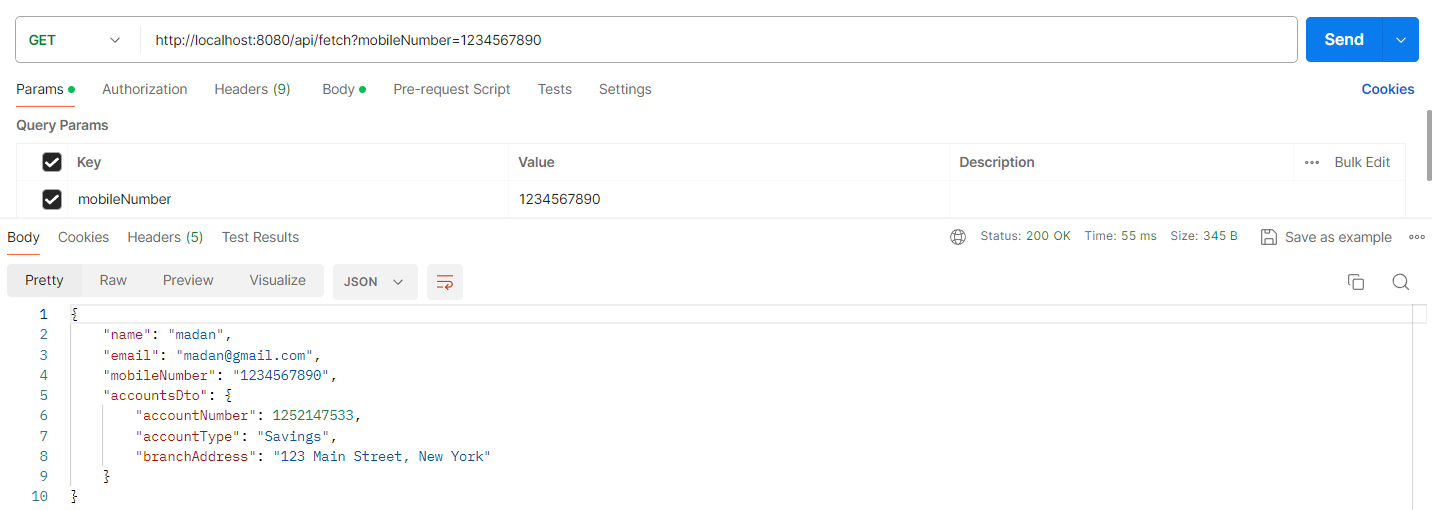
      service: microservice-configserver-config

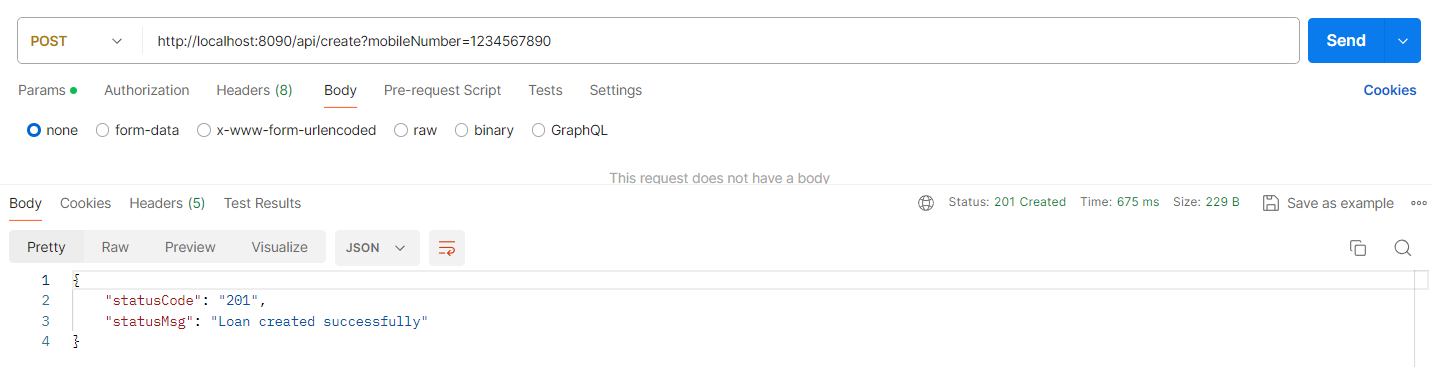
networks:

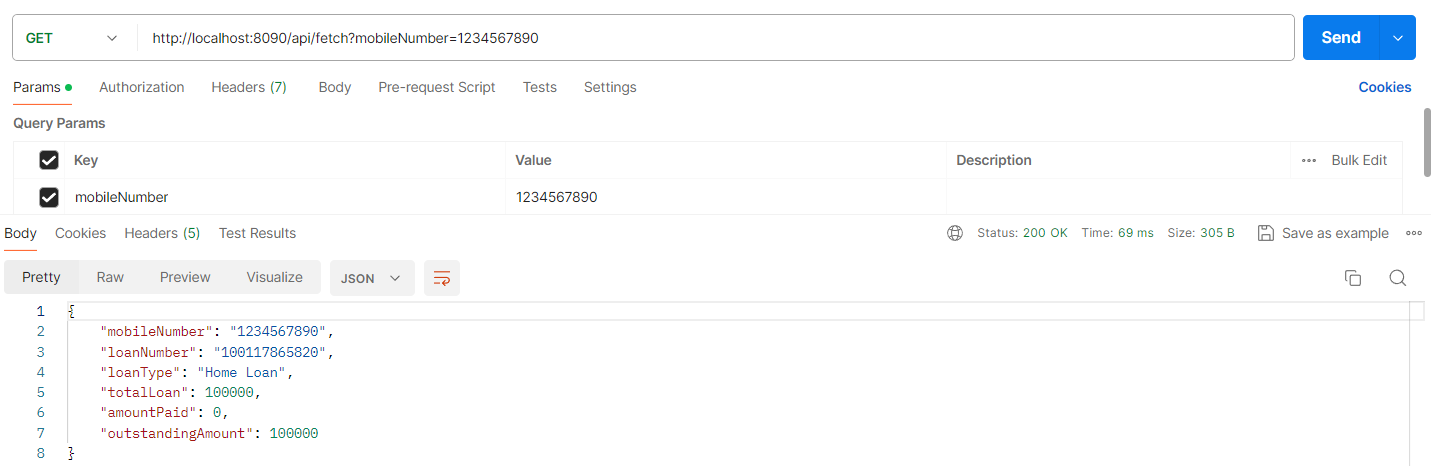
  eazybank:

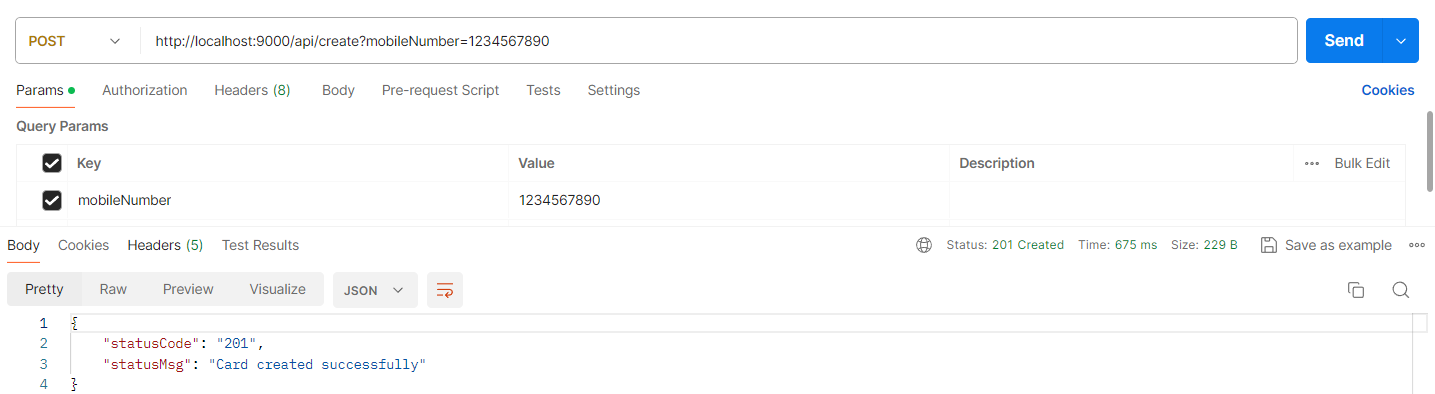
    driver: "bridge"

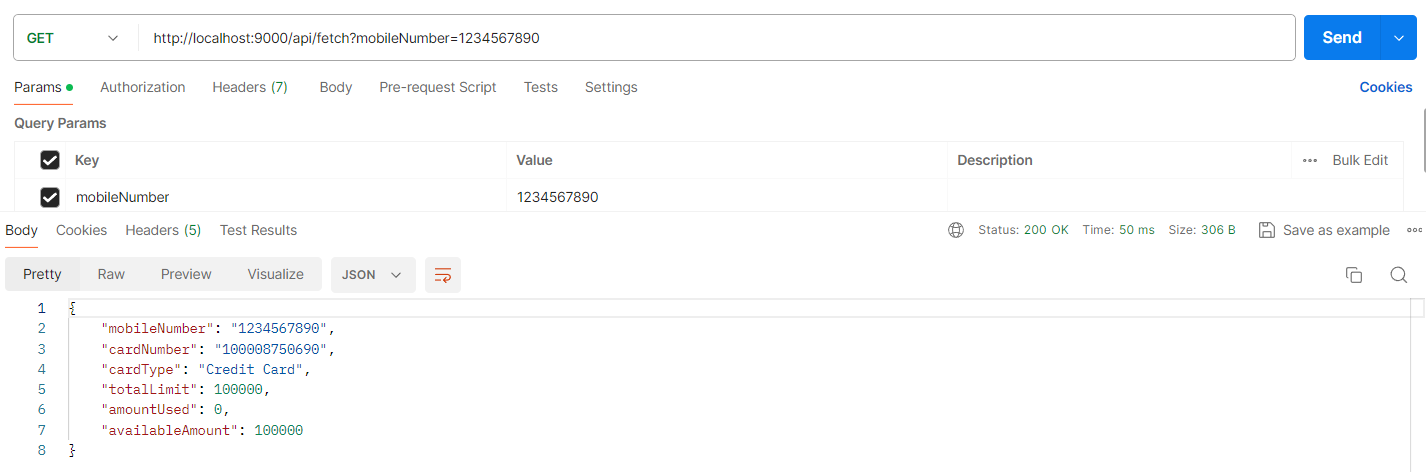


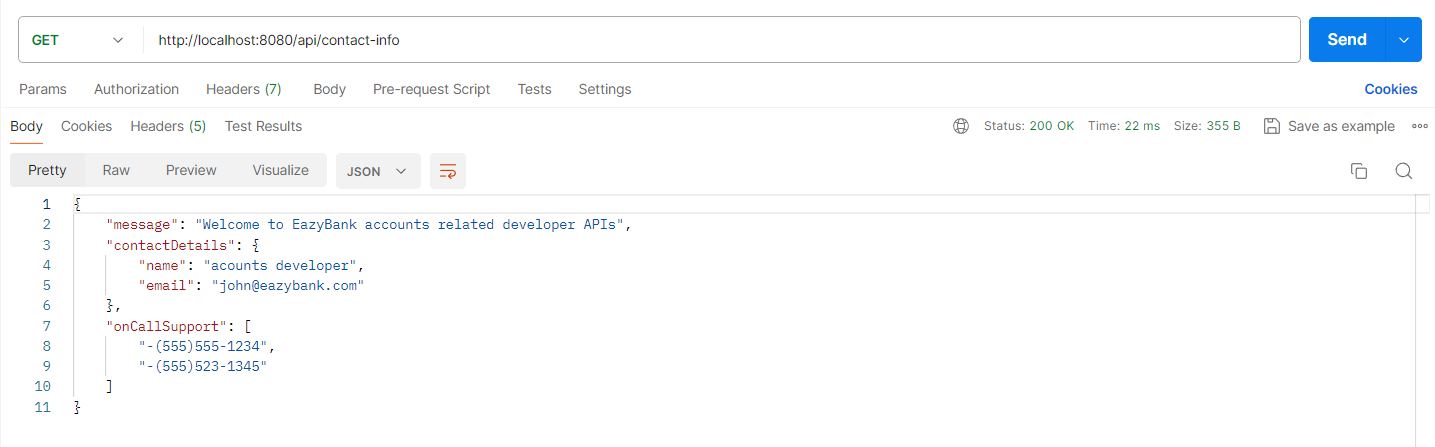


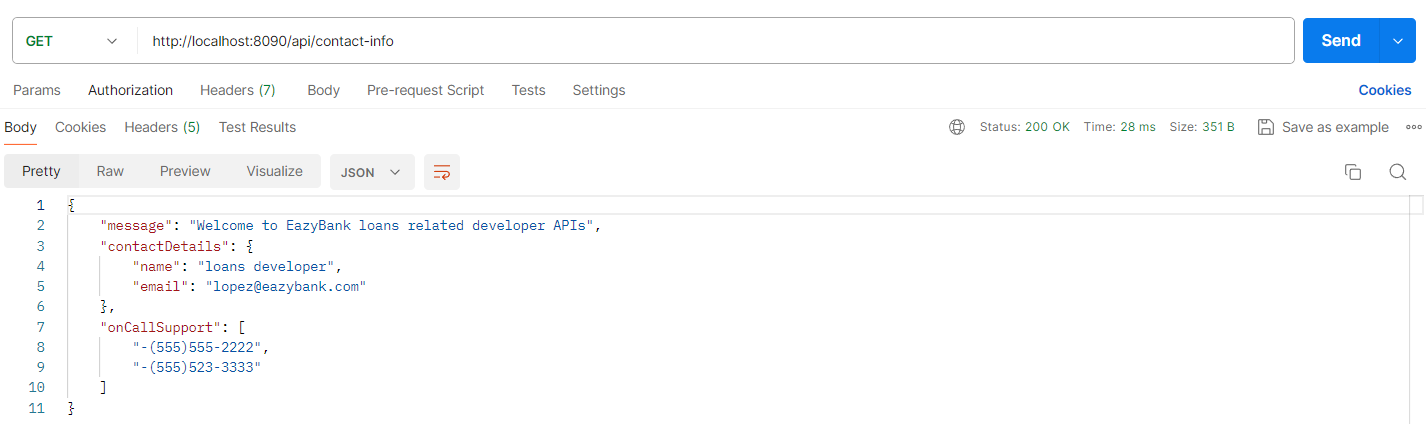


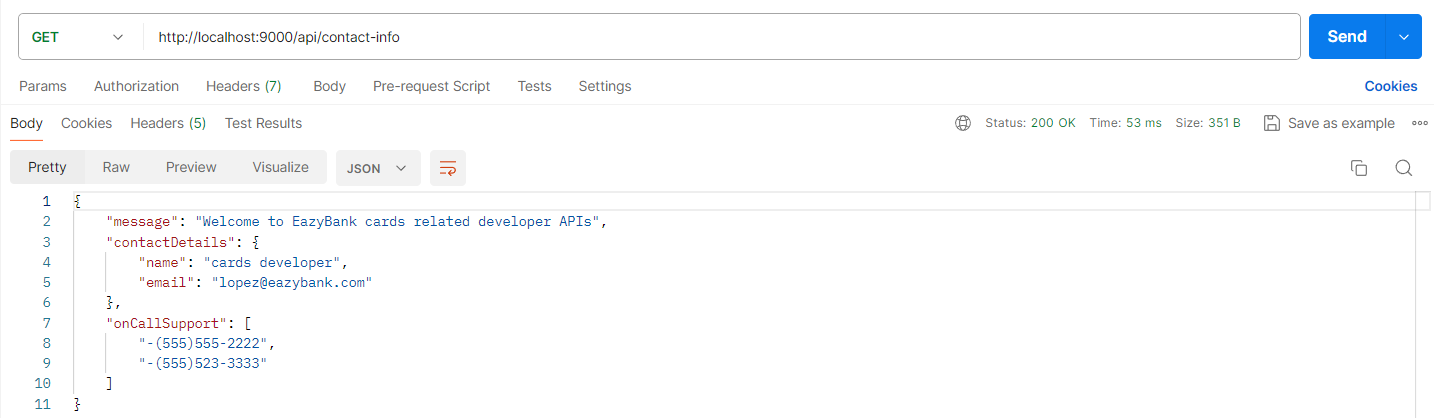












Preparing same docker compose files for QA and PROD

QA

D:\Experiments\Microservices\sb-bank-application\docker-compose\qa\common-config.yaml

services:

  network-deploy-service:

    networks:

      - eazybank

  microservice-db-config:

    extends:

      service: network-deploy-service

    image: mysql

    healthcheck:

      test: [ "CMD", "mysqladmin" ,"ping", "-h", "localhost" ]

      timeout: 10s

      retries: 10

      interval: 10s

      start\_period: 10s

    environment:

      MYSQL\_ROOT\_PASSWORD: root

  microservice-base-config:

    extends:

      service: network-deploy-service

    deploy:

      resources:

        limits:

          memory: 700m

  microservice-configserver-config:

    extends:

      service: microservice-base-config

    depends\_on:

      configserver:

        condition: service\_healthy

    environment:

      SPRING\_PROFILES\_ACTIVE: qa

      SPRING\_CONFIG\_IMPORT: configserver:http://configserver:8071/

      SPRING\_DATASOURCE\_USERNAME: root

      SPRING\_DATASOURCE\_PASSWORD: root

D:\Experiments\Microservices\sb-bank-application\docker-compose\qa\docker-compose.yaml

services:

  accountsdb:

    container\_name: accountsdb

    ports:

      - 3306:3306

    environment:

      MYSQL\_DATABASE: accountsdb

    extends:

      file: common-config.yaml

      service: microservice-db-config

  loansdb:

    container\_name: loansdb

    ports:

      - 3307:3306

    environment:

      MYSQL\_DATABASE: loansdb

    extends:

      file: common-config.yaml

      service: microservice-db-config

  cardsdb:

    container\_name: cardsdb

    ports:

      - 3308:3306

    environment:

      MYSQL\_DATABASE: cardsdb

    extends:

      file: common-config.yaml

      service: microservice-db-config

  configserver:

    image: "nileshzarkar/configserver:s7"

    container\_name: configserver-ms

    ports:

      - "8071:8071"

    healthcheck:

      test: "curl --fail --silent localhost:8071/actuator/health/readiness | grep UP || exit 1"

      interval: 10s

      timeout: 5s

      retries: 10

      start\_period: 10s

    extends:

      file: common-config.yaml

      service: microservice-base-config

  accounts:

    image: "nileshzarkar/accounts:s7"

    container\_name: accounts-ms

    ports:

      - "8080:8080"

    environment:

      SPRING\_APPLICATION\_NAME: "accounts"

      SPRING\_DATASOURCE\_URL: "jdbc:mysql://accountsdb:3306/accountsdb"

    depends\_on:

      accountsdb:

        condition: service\_healthy

    extends:

      file: common-config.yaml

      service: microservice-configserver-config

  loans:

    image: "nileshzarkar/loans:s7"

    container\_name: loans-ms

    ports:

      - "8090:8090"

    environment:

      SPRING\_APPLICATION\_NAME: "loans"

      SPRING\_DATASOURCE\_URL: "jdbc:mysql://loansdb:3306/loansdb"

    depends\_on:

      loansdb:

        condition: service\_healthy

    extends:

      file: common-config.yaml

      service: microservice-configserver-config

  cards:

    image: "nileshzarkar/cards:s7"

    container\_name: cards-ms

    ports:

      - "9000:9000"

    environment:

      SPRING\_APPLICATION\_NAME: "cards"

      SPRING\_DATASOURCE\_URL: "jdbc:mysql://cardsdb:3306/cardsdb"

    depends\_on:

      cardsdb:

        condition: service\_healthy

    extends:

      file: common-config.yaml

      service: microservice-configserver-config

networks:

  eazybank:

    driver: "bridge"

PROD

D:\Experiments\Microservices\sb-bank-application\docker-compose\prod\common-config.yaml

services:

  network-deploy-service:

    networks:

      - eazybank

  microservice-db-config:

    extends:

      service: network-deploy-service

    image: mysql

    healthcheck:

      test: [ "CMD", "mysqladmin" ,"ping", "-h", "localhost" ]

      timeout: 10s

      retries: 10

      interval: 10s

      start\_period: 10s

    environment:

      MYSQL\_ROOT\_PASSWORD: root

  microservice-base-config:

    extends:

      service: network-deploy-service

    deploy:

      resources:

        limits:

          memory: 700m

  microservice-configserver-config:

    extends:

      service: microservice-base-config

    depends\_on:

      configserver:

        condition: service\_healthy

    environment:

      SPRING\_PROFILES\_ACTIVE: prod

      SPRING\_CONFIG\_IMPORT: configserver:http://configserver:8071/

      SPRING\_DATASOURCE\_USERNAME: root

      SPRING\_DATASOURCE\_PASSWORD: root

D:\Experiments\Microservices\sb-bank-application\docker-compose\prod\docker-compose.yaml

services:

  accountsdb:

    container\_name: accountsdb

    ports:

      - 3306:3306

    environment:

      MYSQL\_DATABASE: accountsdb

    extends:

      file: common-config.yaml

      service: microservice-db-config

  loansdb:

    container\_name: loansdb

    ports:

      - 3307:3306

    environment:

      MYSQL\_DATABASE: loansdb

    extends:

      file: common-config.yaml

      service: microservice-db-config

  cardsdb:

    container\_name: cardsdb

    ports:

      - 3308:3306

    environment:

      MYSQL\_DATABASE: cardsdb

    extends:

      file: common-config.yaml

      service: microservice-db-config

  configserver:

    image: "nileshzarkar/configserver:s7"

    container\_name: configserver-ms

    ports:

      - "8071:8071"

    healthcheck:

      test: "curl --fail --silent localhost:8071/actuator/health/readiness | grep UP || exit 1"

      interval: 10s

      timeout: 5s

      retries: 10

      start\_period: 10s

    extends:

      file: common-config.yaml

      service: microservice-base-config

  accounts:

    image: "nileshzarkar/accounts:s7"

    container\_name: accounts-ms

    ports:

      - "8080:8080"

    environment:

      SPRING\_APPLICATION\_NAME: "accounts"

      SPRING\_DATASOURCE\_URL: "jdbc:mysql://accountsdb:3306/accountsdb"

    depends\_on:

      accountsdb:

        condition: service\_healthy

    extends:

      file: common-config.yaml

      service: microservice-configserver-config

  loans:

    image: "nileshzarkar/loans:s7"

    container\_name: loans-ms

    ports:

      - "8090:8090"

    environment:

      SPRING\_APPLICATION\_NAME: "loans"

      SPRING\_DATASOURCE\_URL: "jdbc:mysql://loansdb:3306/loansdb"

    depends\_on:

      loansdb:

        condition: service\_healthy

    extends:

      file: common-config.yaml

      service: microservice-configserver-config

  cards:

    image: "nileshzarkar/cards:s7"

    container\_name: cards-ms

    ports:

      - "9000:9000"

    environment:

      SPRING\_APPLICATION\_NAME: "cards"

      SPRING\_DATASOURCE\_URL: "jdbc:mysql://cardsdb:3306/cardsdb"

    depends\_on:

      cardsdb:

        condition: service\_healthy

    extends:

      file: common-config.yaml

      service: microservice-configserver-config

networks:

  eazybank:

    driver: "bridge"

Demo of Docker network concept

As of now, we make sure that all our containers and services, they are getting started inside a

same network.

So, if you see all our microservices, they are extending these **network-deploy-service** and the

network that we attach to all our containers is **eazybank**.

Now let me show you a demo where I will try to detach the databases from this network.

And with that I will show you a demo that the communication is failing. And post that,

I will also show a demo that the communication between your microservices and databases will fail.

D:\Experiments\Microservices\sb-bank-application\docker-compose\qa\common-config.yaml

You can see as of now our **microservice-db-config** is extending a service with the name **network-deploy-service**.

That means all the containers which is using this service.

They will get started inside this network.

….

  microservice-db-config:

    extends:

      service: network-deploy-service

    image: mysql

    healthcheck:

      test: [ "CMD", "mysqladmin" ,"ping", "-h", "localhost" ]

      timeout: 10s

      retries: 10

      interval: 10s

      start\_period: 10s

    environment:

      MYSQL\_ROOT\_PASSWORD: root

….

So, if I try to remove this here now my database containers, they will not start inside the network

of **easybank** and with that the communication will fail.

PS D:\Experiments\Microservices\sb-bank-application\docker-compose\qa> docker compose up

✔ Network qa\_eazybank Created

✔ Container loansdb Created

✔ Container accountsdb Created

✔ Container cardsdb Created

✔ Container configserver-ms Created

✔ Container accounts-ms Created

✔ Container loans-ms Created

✔ Container cards-ms Created

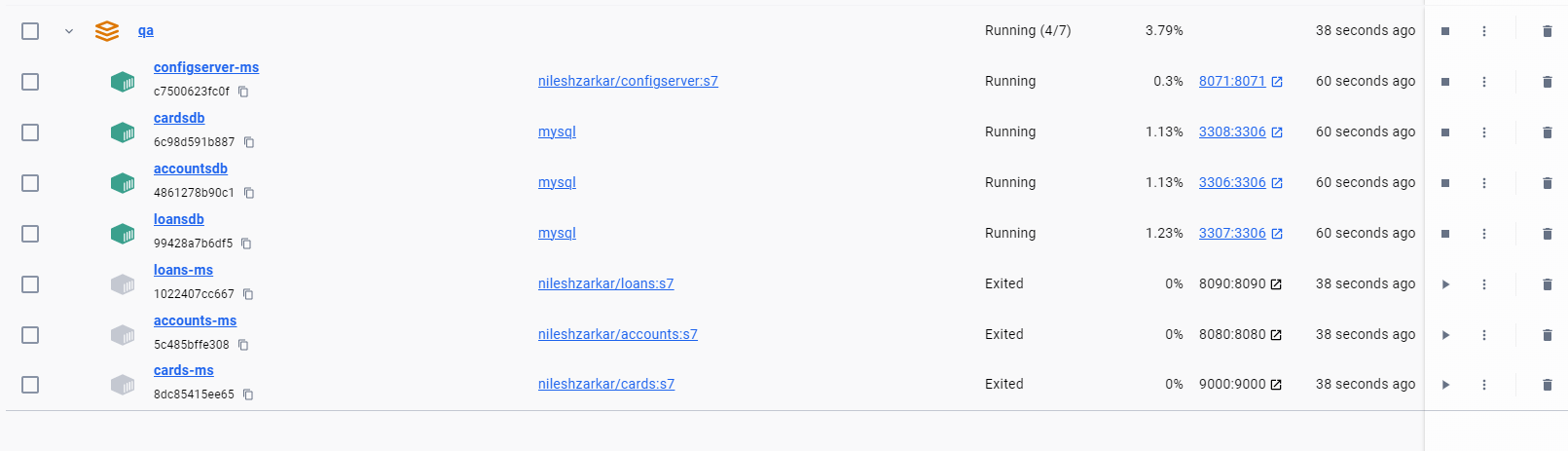
Attaching to accounts-ms, accountsdb, cards-ms, cardsdb, configserver-ms, loans-ms, loansdb

…

So here under the qa profile you can see my loans, database accounts, database, and cards database. They started successfully.

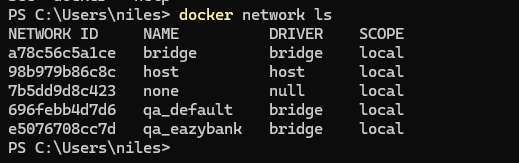
Config server microservices also started successfully, but coming to the other containers like accounts microservice, cards microservice and loans microservice.

They initially they are in the running status, but after some time they turned into exited status because if you go and open them, the communication is failing with the database, they are not able to create a connection.



I am hoping you are understanding how the Docker network works.

We can also try to see the list of Docker networks created behind the scenes.



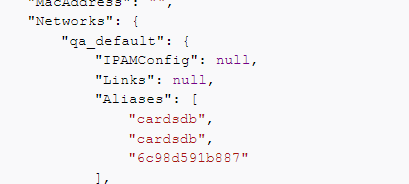
So, this will list you all the Dockers networks that are created.

So, you can see we have a network created with the name **qa\_easybank** with the qa profile,

and

at the same time there is also other networks that got created.

accountsdb, loansdb, and cardsdb are attached to **qa\_default** network



accounts-ms, loans-ms, and cards-ms are attached to **qa\_eazybank** network  


Which means inside a Docker compose network.

Whenever you are not mentioning any network, they will all get assigned to a default network, whereas my accounts microservice and all other microservice, since we have created and attached a different network and since these are two different network, the communication is not going to work.

So, this is the power of Docker.

You will have complete isolation until unless you allow the network communication, the communication will never happen.