

# Spring Cloud로 개발하는 マイクロ서비스 アプリケイ션



Microservices

+



Spring  
Cloud

```
class Book {
    def self, title, price, author;
    self.title = title
    self.price = price
    self.author = author
}

public static void main(String[] args)
{
    var fs = require('fs');
    fs.readFile('/JONE.txt' /* 1 */,
        function (err, data) {
            console.log(data); // 3
        });
}

<@interface NextInnovationDelegate : NSObject <UIApplicationDelegate> >

<@implementation NextInnovationDelegate >
<@end>
```



# 목차



- Section 9: 암호화 처리를 위한 Encryption과 Decryption
- Section 10: 마이크로서비스간 통신
- Section 11: 데이터 동기화를 위한 Kafka 활용 ①
- Section 12: 데이터 동기화를 위한 Kafka 활용 ②
- Section 13: 장애 처리와 Microservice 분산 추적
- Section 14: Microservice 모니터링
- Section 15: 애플리케이션 배포를 위한 컨테이너 가상화
- **Section 16: 애플리케이션 배포 – Docker Container**
- Appendix: Microservice 패턴

# Section 16.

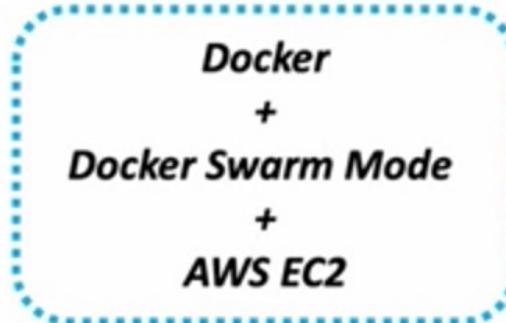
# 애플리케이션 배포

# Docker Container

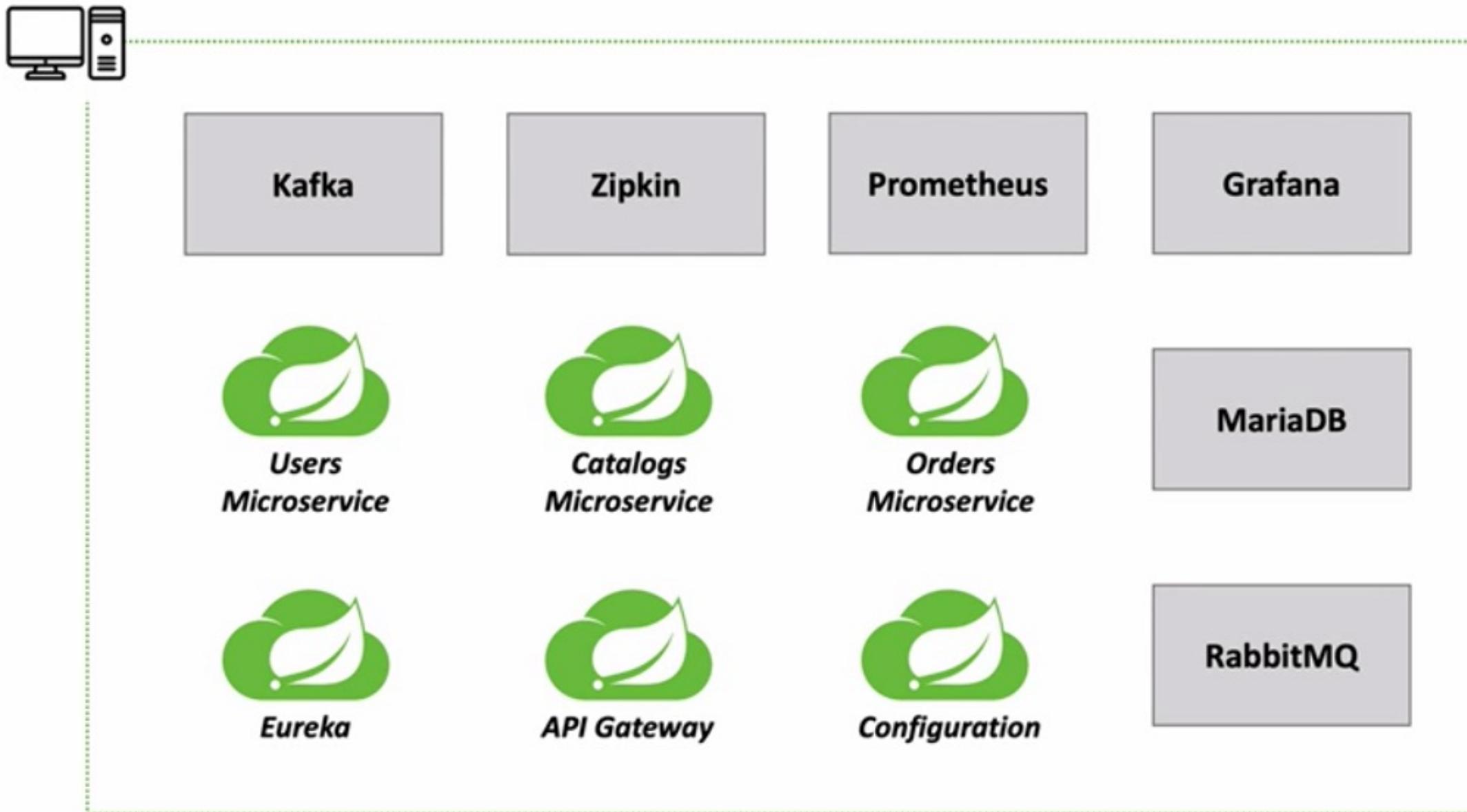
- Design Deployment
- Configuration server
- Eureka Discovery
- API Gateway
- MariaDB
- Kafka
- Zipkin
- Monitoring
- Microservices
- Multiple Environments

# Running Microservices

- IntelliJ IDEA
- Exported JAR file
- Docker Container



# *Running Microservices in Local*



# Create Bridge Network



## ▪ **Bridge network**

- \$ docker network create --driver bridge [브릿지 이름]

## ▪ **Host network**

- 네트워크를 호스트로 설정하면 호스트의 네트워크 환경을 그대로 사용
- 포트 포워딩 없이 내부 어플리케이션 사용

## ▪ **None network**

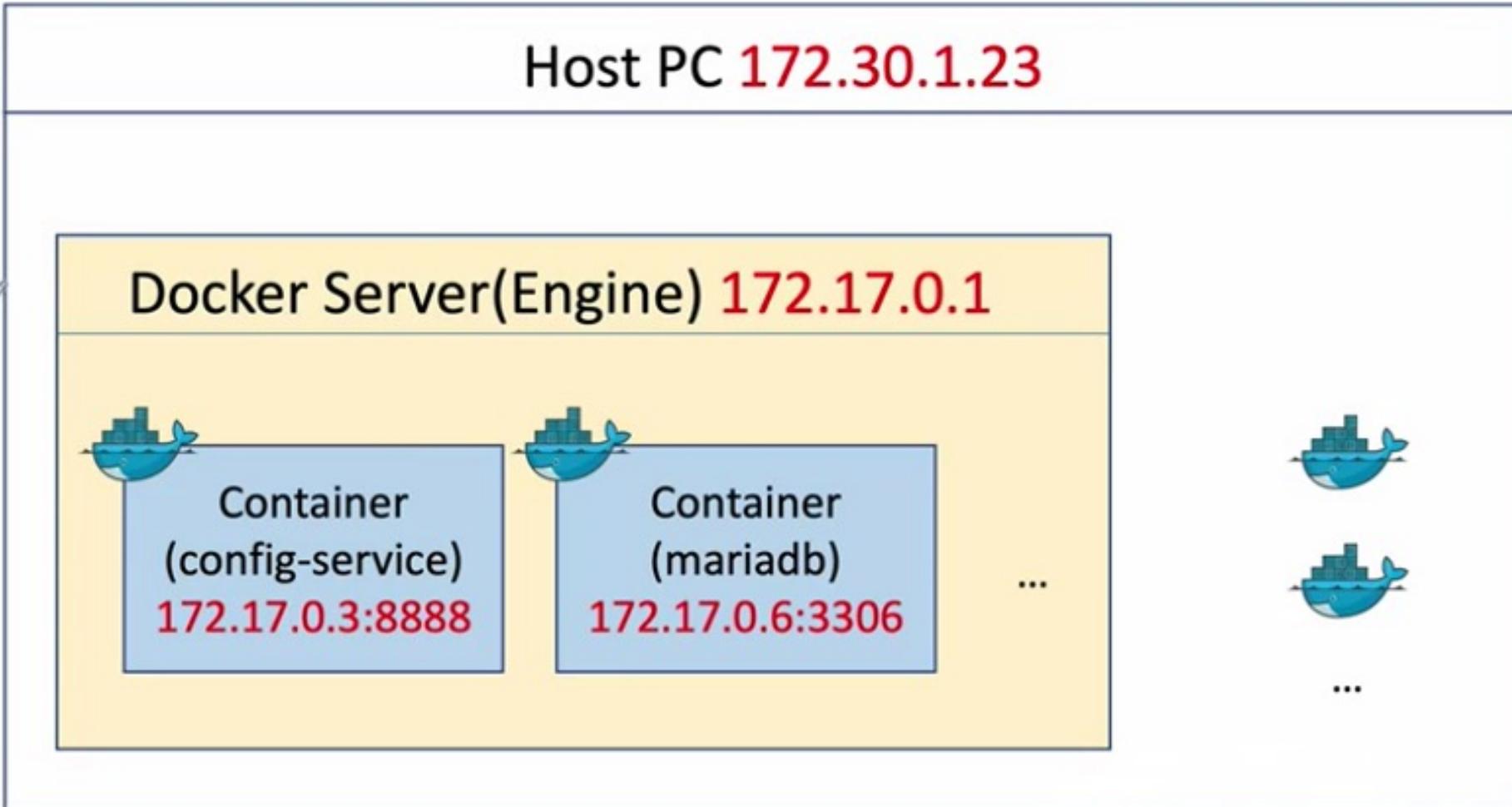
- 네트워크를 사용하지 않음
- Io 네트워크만 사용, 외부와 단절

```
$ docker network create ecommerce-network
```

```
$ docker network ls
```

▶ docker network ls				
NETWORK ID	NAME	DRIVER	SCOPE	
2879c2a94c9f	bridge	bridge	local	
22e70794e399	ecommerce-network	bridge	local	
de014ae20e46	host	host	local	
2624444d0133	none	null	local	

# Create Bridge Network



# Run RabbitMQ

```
$ docker run -d --name rabbitmq --network ecommerce-network \
-p 15672:15672 -p 5672:5672 -p 15671:15671 -p 5671:5671 -p 4369:4369 \
-e RABBITMQ_DEFAULT_USER=guest \
-e RABBITMQ_DEFAULT_PASS=guest rabbitmq:management
```

```
Starting RabbitMQ 3.8.11 on Erlang 23.2.3
Copyright (c) 2007-2020 VMware, Inc. or its affiliates.
Licensed under the MPL 2.0. Website: https://rabbitmq.com

## ##      RabbitMQ 3.8.11
## ##
##### Copyright (c) 2007-2020 VMware, Inc. or its affiliates.
#####
##### Licensed under the MPL 2.0. Website: https://rabbitmq.com

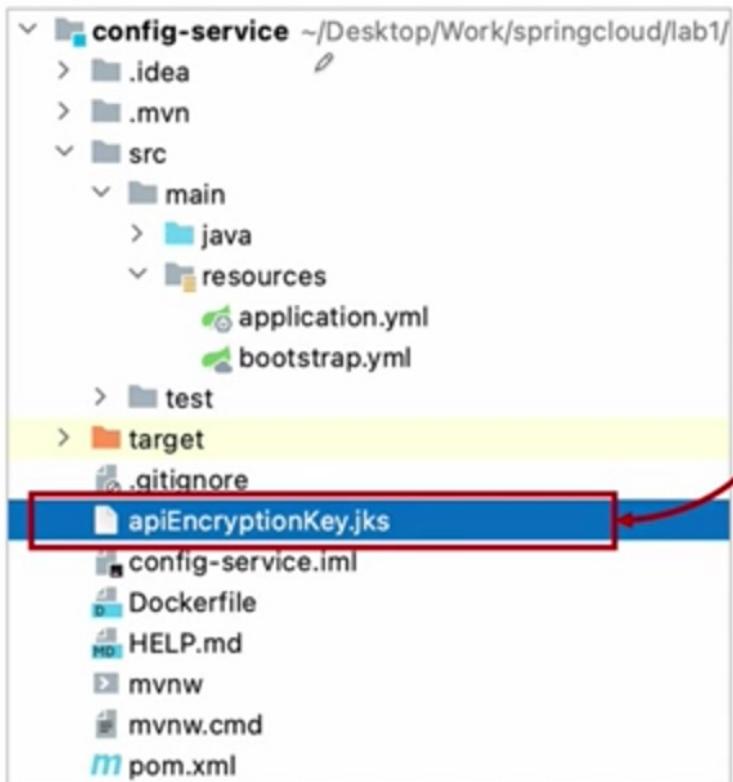
Doc guides: https://rabbitmq.com/documentation.html
Support: https://rabbitmq.com/contact.html
Tutorials: https://rabbitmq.com/getstarted.html
Monitoring: https://rabbitmq.com/monitoring.html
```

The screenshot shows the RabbitMQ Management Interface at the URL `127.0.0.1:15672/#/`. The interface is for RabbitMQ version 3.8.11 running on Erlang 23.2.3. The main navigation bar includes links for Overview, Connections, Channels, Exchanges, Queues, and Admin. The Overview page displays various metrics and statistics. At the top, it shows 'Queued messages last minute' and 'Currently idle'. Below that, it shows 'Message rates last minute' and 'Currently idle'. It also displays 'Global counts' with metrics for Connections (0), Channels (0), Exchanges (7), Queues (0), and Consumers (0). The 'Nodes' section lists a single node named 'rabbit@d3eaa89fc187' with detailed resource usage statistics.

Name	File descriptors ?	Socket descriptors ?	Erlang processes	Memory ?	Disk space	Uptime
rabbit@d3eaa89fc187	102 1048576 available	0 943629 available	557 1048576 available	102 MiB 796 MiB high watermark	43 GiB 48 MiB low watermark	1m 54s

# Create Config Server Docker Image

```
FROM openjdk:17-ea-11-jdk-slim  
VOLUME /tmp  
COPY apiEncryptionKey.jks apiEncryptionKey.jks  
COPY target/config-service-1.0.jar ConfigService.jar  
ENTRYPOINT ["java", "-jar", "ConfigService.jar"]
```



```
<groupId>com.example</groupId>  
<artifactId>config-service</artifactId>  
    <version>0.0.1-SNAPSHOT</version>-->  
<version>1.0</version>  
<name>config-service</name>
```

- **copy to docker container**

# Create Config Server Docker Image

```
encrypt:  
# key: abcdefghijklmnopqrstuvwxyz0123456789  
key-store:  
# location: file://${user.home}/Desktop/Work/keystore/apiEncryptionKey.jks  
location: file:/apiEncryptionKey.jks  
password: test1234  
alias: apiEncryptionKey
```

- *change the path*

```
$ mvn clean compile package -DskipTests=true
```

```
[INFO] --- maven-surefire-plugin:2.22.2:test (default-test) @ config-service ---  
[INFO] Tests are skipped.  
[INFO]  
[INFO] --- maven-jar-plugin:3.2.0:jar (default-jar) @ config-service ---  
[INFO] Building jar: /Users/downonlee/Desktop/Work/springcloud/lab1/config-service/target/config-service-1.0.jar  
[INFO]  
[INFO] --- spring-boot-maven-plugin:2.4.2:repackage (repackage) @ config-service ---  
[INFO] Replacing main artifact with repackaged archive  
[INFO] -----  
[INFO] BUILD SUCCESS  
[INFO] -----  
[INFO] Total time: 3.616 s  
[INFO] Finished at: 2021-03-08T09:05:27+09:00  
[INFO] -----
```

# Build Eureka Discovery Docker Image



```
FROM openjdk:17-ea-11-jdk-slim  
VOLUME /tmp  
COPY target/discoveryservice-1.0.jar DiscoveryService.jar  
ENTRYPOINT ["java", "-jar", "DiscoveryService.jar"]
```

- docker build --tag= edowon0623/discovery-service .
- docker push edowon0623/discovery-service

The screenshot shows a Docker registry interface. At the top, there is a search bar with the text 'discovery-service'. To the left of the search bar is a dropdown menu showing 'edowon0623'. On the right, there is a blue 'Create Repository' button. Below the search bar, a list of repositories is shown. One repository, 'edowon0623 / discovery-service', is highlighted with a red border. This repository was updated 2 minutes ago. To its right, there are status indicators: 'Not Scanned', '0 stars', '1 download', and a 'Public' link.

edowon0623

discovery-service

Create Repository

edowon0623 / discovery-service

Updated 2 minutes ago

Not Scanned

0

1

Public



# Run Eureka Discovery

```
$ docker run -d -p 8761:8761 --network ecommerce-network \
-e "spring.cloud.config.uri=http://config-service:8888" \
--name discovery-service edowon0623/discovery-service
```

```
$ docker logs discovery-service
```

```
INFO 1 --- [      Thread-10] o.s.c.n.e.server.EurekaServerBootstrap : isAws returned false
INFO 1 --- [      Thread-10] o.s.c.n.e.server.EurekaServerBootstrap : Initialized server context
INFO 1 --- [      Thread-10] c.n.e.r.PeerAwareInstanceRegistryImpl : Got 1 instances from neighboring DS node
INFO 1 --- [      Thread-10] c.n.e.r.PeerAwareInstanceRegistryImpl : Renew threshold is: 1
INFO 1 --- [      Thread-10] c.n.e.r.PeerAwareInstanceRegistryImpl : Changing status to UP
INFO 1 --- [      Thread-10] e.s.EurekaServerInitializerConfiguration : Started Eureka Server
INFO 1 --- [          main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8010 (http) with context path ''
INFO 1 --- [          main] .s.c.n.e.s.EurekaAutoServiceRegistration : Updating port to 8010
```

# *Build ApiGateway Service Docker Image*

```
FROM openjdk:17-ea-11-jdk-slim  
VOLUME /tmp  
COPY target/apigateway-1.0.jar ApiGateway.jar  
ENTRYPOINT ["java", "-jar", "ApiGateway.jar"]
```

- docker build --tag=edowon0623/apigateway-service .
- docker push edowon0623/apigateway-service

The screenshot shows a Docker registry interface. At the top, there is a search bar with the text "apigateway-service" and a dropdown menu showing "edowon0623". To the right of the search bar is a blue "Create Repository" button. Below the search bar, the repository details are displayed: "edowon0623 / apigateway-service", "Updated 2 minutes ago", "Not Scanned", "0 stars", "1 download", and "Public".

edowon0623 | ▾

apigateway-service

Create Repository

edowon0623 / apigateway-service

Updated 2 minutes ago

Not Scanned

0

1

Public

# Run ApiGateway Service

```
spring:  
  cloud:  
    config:  
      uri: http://127.0.0.1:8888  
      name: ecommerce
```

```
eureka:  
  client:  
    register-with-eureka: true  
    fetch-registry: true  
    service-url:  
      defaultZone: http://localhost:8761/eureka
```

```
$ docker run -d -p 8000:8000 --network ecommerce-network \  
-e "spring.cloud.config.uri=http://config-service:8888" \  
-e "spring.rabbitmq.host=rabbitmq" \  
-e 'eureka.client.serviceUrl.defaultZone=http://discovery-service:8761/eureka/' \  
--name apigateway-service \  
edowon0623/apigateway-service
```

# *Build, Run MariaDB*

```
FROM mariadb
ENV MYSQL_ROOT_PASSWORD test1357
ENV MYSQL_DATABASE mydb
COPY ./mysql /var/lib/mysql
EXPOSE 3306
ENTRYPOINT ["mysqld"]
```

```
▶ mysql.server start
Starting MariaDB
210308 23:25:30 mysqld_safe Logging to '/usr/local/var/mysql/DOWONui-MacBookPro.local.err'.
210308 23:25:30 mysqld_safe Starting mariadb daemon with databases from /usr/local/var/mysql
SUCCESS!
```

```
C:\Work\mariadb-10.5.8-winx64>.\bin\mariadb-install-db.exe --datadir=C:\Work\mariadb-10.5.8-winx64\data --service=mariaDB --port=3306 --password=test1357
Running bootstrap
2021-02-21 12:45:55 0 [Note] C:\Work\mariadb-10.5.8-winx64\bin\mysqld.exe (mysqld 10.5.8-MariaDB) starting as process 24056 ...
Removing default user
Setting root password
Creating my.ini file
Registering service 'mariaDB'
Creation of the database was successful

C:\Work\mariadb-10.5.8-winx64>
```

```
$ docker build -t edowon0623/my_mariadb -f Dockerfile_mariadb .
```

```
$ docker run -d -p 3306:3306 --network ecommerce-network --name mariadb edowon0623/my_mariadb
```

# Run Kafka Server

## ■ Zookeeper + Kafka Standalone

- docker-compose로 실행
- git clone <https://github.com/wurstmeister/kafka-docker>
- docker-compose-single-broker.yml 수정

```
$ docker-compose -f docker-compose-single-broker.yml up -d
```

```
version: '2'
services:
  zookeeper:
    image: wurstmeister/zookeeper
    ports:
      - "2181:2181"
    networks:
      my-network:
        ipv4_address: 172.18.0.100
  kafka:
    # build: .
    image: wurstmeister/kafka
    ports:
      - "9092:9092"
    environment:
      KAFKA_ADVERTISED_HOST_NAME: 172.18.0.101
      KAFKA_CREATE_TOPICS: "test:1:1"
      KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
    depends_on:
      - zookeeper
    networks:
      my-network:
        ipv4_address: 172.18.0.101
networks:
  my-network:
    name: ecommerce-network
```



# *Run Zipkin*

- <https://zipkin.io/pages/quickstart>

```
$ docker run -d -p 9411:9411 \
  --network ecommerce-network \
  --name zipkin \
  openzipkin/zipkin
```