포팅 매뉴얼

- version
  - JAVA - 17.0.13
  - node.js - 20.18.0

▼ DB 및 모니터링

▼ docker-compose.yaml

```
name: toudeuk-docker-compose



services:
  database:
    image: mysql:8.0
    container_name: toudeuk-database
    environment:
      MYSQL_ROOT_PASSWORD: toudeuk
      MYSQL_DATABASE: toudeuk
      MYSQL_USER: toudeuk
      MYSQL_PASSWORD: toudeuk
      TZ: Asia/Seoul
    ports:
      - "3306:3306"
    volumes:
      - toudeuk-db:/var/lib/mysql

  db-admin:
    container_name: db-admin
    image: phpmyadmin/phpmyadmin
    ports:
      - "5000:80"
    environment:
      PMA_HOST: database
      MYSQL_ROOT_PASSWORD: toudeuk
```

```
    restart: always

  cache-database:
    image: redis
    container_name: toudeuk-cache-database
    ports:
      - "6379:6379"
    entrypoint: redis-server --requirepass toudeuk --ma

  prometheus:
    image: prom/prometheus:latest
    container_name: prometheus
    ports:
      - "9090:9090"
    volumes:
      - ./prometheus.yml:/etc/prometheus/prometheus.yml
      - prometheus-data:/prometheus

  grafana:
    image: grafana/grafana:latest
    container_name: grafana
    user: "$UID:$GID"
    ports:
      - "3001:3000"
    volumes:
      - grafana-data:/var/lib/grafana
    depends_on:
      - prometheus


  # !! Ngrinder 설정 나중에 서버를 분리하면 스프링 서버가 돌아가
  controller:
    container_name: ngrinder-controller
    image: ngrinder/controller:3.5.5
    restart: always
    ports:
      - "9091:80"
      - "16001:16001"
```

```
      - "12000-12009:12000-12009"
    volumes:
      - /tmp/ngrinder-controller:/opt/ngrinder-controll
  agent:
    container_name: ngrinder-agent
    image: ngrinder/agent:3.5.5
    restart: always
    links:
      - controller


  zookeeper-1:
    image: confluentinc/cp-zookeeper:latest
    ports:
      - '32181:32181'
    environment:
      ZOOKEEPER_CLIENT_PORT: 32181
      ZOOKEEPER_TICK_TIME: 2000
  kafka-1:
    image: confluentinc/cp-kafka:latest
    container_name: kafka-msa
    ports:
      - '9092:9092'
    depends_on:
      - zookeeper-1
    environment:
      KAFKA_BROKER_ID: 1
      KAFKA_ZOOKEEPER_CONNECT: zookeeper-1:32181
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: INTERNAL:PL
      KAFKA_INTER_BROKER_LISTENER_NAME: INTERNAL
      KAFKA_ADVERTISED_LISTENERS: INTERNAL://kafka-1:29
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
      KAFKA_GROUP_INITIAL_REBALANCE_DELAY_MS: 0
      #      KAFKA_DEFAULT_REPLICATION_FACTOR: 3
      KAFKA_NUM_PARTITIONS: 1

  kafka-ui:
    image: provectuslabs/kafka-ui
```

```
    container_name: kafka-ui
    ports:
      - "8989:8080"
    restart: always
    environment:
      - KAFKA_CLUSTERS_0_NAME=local
      - KAFKA_CLUSTERS_0_BOOTSTRAPSERVERS=kafka-1:29092
      - KAFKA_CLUSTERS_0_ZOOKEEPER=zookeeper-1:32181
volumes:
  toudeuk-db:
  grafana-data:
  prometheus-data:
```

▼ prometheus.yml

```
scrape_configs:
  - job_name: 'celuveatdev'
    metrics_path: '/actuator/prometheus'
    scrape_interval: 5s
    static_configs:
      - targets: ['toudeuk.kr:9093']
```

▼ 환경변수

▼ build.gradle

```
plugins {
    id 'java'
    id 'org.springframework.boot' version '3.3.4'
    id 'io.spring.dependency-management' version '1.
}

group = 'com.toudeuk'
version = '0.0.1-SNAPSHOT'

java {
    toolchain {
        languageVersion = JavaLanguageVersion.of(17)
    }
```

```
}

configurations {
    compileOnly {
        extendsFrom annotationProcessor
    }
}

repositories {
    mavenCentral()
}

dependencies {


    // WEB
    implementation 'org.springframework.boot:spring-
    developmentOnly 'org.springframework.boot:spring

    // JSON Converter
    implementation group: 'com.google.code.gson', na
    implementation 'com.fasterxml.jackson.core:jacks

    // TEST
    testImplementation 'org.springframework.boot:spr
    testRuntimeOnly 'org.junit.platform:junit-platfo
    testImplementation 'com.google.code.gson:gson:2.

    // DB
    implementation 'org.springframework.boot:spring-
    implementation 'org.springframework.boot:spring-
    runtimeOnly 'com.mysql:mysql-connector-j'

    //Java TimeModule
    implementation 'com.fasterxml.jackson.datatype:j

    // Lombok
    compileOnly 'org.projectlombok:lombok'
```

```
    annotationProcessor 'org.projectlombok:lombok'

    // WebSocket
    implementation 'org.springframework.boot:spring-
    implementation 'org.webjars:sockjs-client:1.0.2'
    implementation 'org.webjars:stomp-websocket:2.3.

    // Temp for Web Socket test
    implementation 'org.webjars:webjars-locator-core
    implementation 'org.webjars:bootstrap:3.3.7'
    implementation 'org.webjars:jquery:3.1.1-1'

    // QueryDSL
    implementation 'com.querydsl:querydsl-jpa:5.0.0:
    annotationProcessor "com.querydsl:querydsl-apt:5
    annotationProcessor "jakarta.annotation:jakarta.
    annotationProcessor "jakarta.persistence:jakarta
    implementation 'com.querydsl:querydsl-spatial'

    // JWT
    implementation 'io.jsonwebtoken:jjwt-api:0.12.3'
    implementation 'io.jsonwebtoken:jjwt-impl:0.12.3
    implementation 'io.jsonwebtoken:jjwt-jackson:0.1

    // Security
    implementation 'org.springframework.boot:spring-
    testImplementation 'org.springframework.security
    implementation 'org.springframework.boot:spring-

    // Kafka
    implementation 'org.springframework.kafka:spring



    // Monitoring
    implementation 'org.springframework.boot:spring-
    implementation 'io.micrometer:micrometer-registr

    //마이크로미터 설정
```

```
    runtimeOnly "io.micrometer:micrometer-registry-p


    // OAuth2
    implementation 'org.springframework.boot:spring-

    // Swagger
    implementation 'org.springdoc:springdoc-openapi-

    // AWS S3
    implementation 'org.springframework.cloud:spring
}

tasks.named('test') {
    useJUnitPlatform()
}
```

▼ application.properties

```
                spring.application.name=server
# JPA
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
spring.jpa.properties.hibernate.dialect=org.hibernat
spring.jpa.open-in-view=false
logging.level.org.hibernate.SQL=ERROR
# MySQL
spring.datasource.url=jdbc:mysql://localhost:3306/to
spring.datasource.username=toudeuk
spring.datasource.password=toudeuk
spring.datasource.driver-class-name=com.mysql.cj.jdb
# Redis
spring.data.redis.host=localhost
spring.data.redis.port=6379
spring.data.redis.password=toudeuk
#swagger
springdoc.swagger-ui.path=/api-test
```

```
springdoc.swagger-ui.groups-order=desc
springdoc.swagger-ui.tags-sorter=alpha
springdoc.swagger-ui.operations-sorter=method
springdoc.paths-to-match=/api/v1/**
springdoc.use-fqn=true
#
spring.jwt.access-expire=1000000000
spring.jwt.refresh-expire=1000000000
spring.jwt.secret=toudeukasdfasdfasdfasdfasdfasdfasdf
spring.config.import=optional:application-secret.pro
jwt.issuer=secretkey@gmail.com
management.endpoints.web.exposure.include=*
management.server.port=9093
server.tomcat.mbeanregistry.enabled=true
management.endpoint.prometheus.enabled=true
# Kafka
producers.topics.click.name=click
consumers.topics.click.name=click
producers.topics.item-buy.name=item-buy
consumers.topics.item-buy.name=item-buy

producers.topics.charge-cash.name=charge-cash
consumers.topics.charge-cash.name=charge-cash

producers.topics.game-cash-log.name=game-cash-log
consumers.topics.game-cash-log.name=game-cash-log

consumers.group-id.topics.click.name=toudeuk

spring.kafka.consumer.auto-offset-reset=earliest
spring.kafka.consumer.key-deserializer=org.apache.ka
spring.kafka.consumer.value-deserializer=org.apache.
spring.kafka.producer.key-serializer=org.apache.kafk
spring.kafka.producer.value-serializer=org.springfra
# ?? ?? ??? ? ??
server.tomcat.threads.max=400
server.tomcat.threads.min-spare=50
server.tomcat.accept-count=200
```

```
server.tomcat.max-connections=10000
# ??? ? ?? (HikariCP ?? ?)
spring.datasource.hikari.maximum-pool-size=80
spring.datasource.hikari.minimum-idle=80
```

▼ application-secret.properties

```
#registration - kakao
spring.security.oauth2.client.registration.kakao.cli
spring.security.oauth2.client.registration.kakao.cli
spring.security.oauth2.client.registration.kakao.red
spring.security.oauth2.client.registration.kakao.aut
spring.security.oauth2.client.registration.kakao.cli
spring.security.oauth2.client.registration.kakao.sco

# Kakao OAuth2 Provider
spring.security.oauth2.client.provider.kakao.authori
spring.security.oauth2.client.provider.kakao.token-u
spring.security.oauth2.client.provider.kakao.user-in
spring.security.oauth2.client.provider.kakao.user-na

# KAKAO PAY API
# 카카오페이 개발자센터에서 발급받아서 넣어주면 됨
kakaopay.api.secret.key=DEV86281C93C936CA199E81EBF2F
cid=TC0ONETIME
# 카카오페이 개발자센터에서 등록한 url로 교체 후 사용
kakaopay.api.host=
kakaopay.redirect.url=

# S3
cloud.aws.region.static=ap-northeast-2
cloud.aws.stack.auto-=false
# 아래 두개의 키는 s3 발급 받아서 내야함
cloud.aws.credentials.accessKey=
cloud.aws.credentials.secretKey=
# 경로는 알맞게 바꿔야함
cloud.aws.s3.bucket=toudeuk/images
# 클라우드 프론트 등록후 넣어주면 됨
```

```
cloud.aws.s3.cloudFrontDomain=https://d2uaylxt6f2ffw

# multipart
spring.servlet.multipart.enabled=true
spring.servlet.multipart.file-size-threshold=2MB
spring.servlet.multipart.max-file-size=10MB
spring.servlet.multipart.max-request-size=10MB
```

▼ .env

```
#NEXT_PUBLIC_SENTRY_DSN=https://873bd5cfbb5745397111
#NEXT_PUBLIC_SENTRY_ENVIRONMENT=development
#SENTRY_TRACES_SAMPLE_RATE=0.1
#SENTRY_AUTH_TOKEN=sntrys_eyJpYXQiOjE3MzAwMjY3MDAuMz
```

▼ .env.production

```


NEXT_PUBLIC_API_URL="https://toudeuk.kr"
# NEXT_PUBLIC_API_URL="http://localhost:8080"
NEXT_PUBLIC_BASE_URL="https://toudeuk.kr"
```

- 외부 서비스 정보
  - 카카오 로그인
  - 카카오 페이

- 시연 시나리오
  -