디플로이먼트, 노드포트, 서비스

kafka 연동

backend-user : external kafka service yaml 작성 및 적용

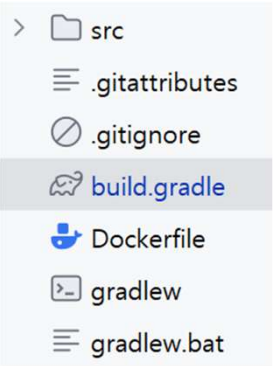
```
PS C:₩server₩service> ipconfig

Windows IP 구성

...

무선 LAN 어댑터 로컬 영역 연결* 3:

   미디어 상태 . . . . . . . . : 미디어 연결 끊김
   연결별 DNS 접미사. . . . :

무선 LAN 어댑터 로컬 영역 연결* 4:

   미디어 상태 . . . . . . . . : 미디어 연결 끊김
   연결별 DNS 접미사. . . . :

무선 LAN 어댑터 Wi-Fi:

   연결별 DNS 접미사. . . . :
   IPv4 주소 . . . . . . . . . . : 192.168.0.135
   서브넷 마스크 . . . . . . . : 255.255.255.0
   기본 게이트웨이 . . . . . . : 192.168.0.1
```

내 PC IP

내 PC IP 확인

```yaml
apiVersion: v1
kind: Service
metadata:
  name: k8s-external-kafka-service
spec:
  ports:
    - port: 9092

---

apiVersion: v1
kind: Endpoints
metadata:
  name: k8s-external-kafka-service
subsets:
  - addresses:
      - ip: 192.168.0.135
    ports:
      - port: 9092
```

backend-user : build.gradle 의존성 추가




```
dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-web'
    implementation 'org.springframework.cloud:spring-cloud-starter-openfeign'

    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
    runtimeOnly 'com.mysql:mysql-connector-j:8.4.0'

    implementation 'org.springframework.boot:spring-boot-starter-validation'

    implementation 'org.springframework.kafka:spring-kafka'
    testImplementation 'org.springframework.kafka:spring-kafka-test'

    compileOnly 'org.projectlombok:lombok'
    annotationProcessor 'org.projectlombok:lombok'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
    testRuntimeOnly 'org.junit.platform:junit-platform-launcher'
}
```

backend-user : application.yml Kafka 설정

```yaml
spring:
  application:
    name: k8s-backend-user

  kafka:
    listener:
      ack-mode: manual_immediate
    consumer:
      group-id: ${spring.application.name}
      key-deserializer: org.apache.kafka.common.serialization.StringDeserializer
      value-deserializer: org.springframework.kafka.support.serializer.JsonDeserializer
      enable-auto-commit: false
      auto-offset-reset: latest
      max-poll-records: 10
      properties:
        spring.json.trusted.packages: "*"
        spring.json.use.type.headers: false  # 헤더의 타입 정보 무시
       producer:
      key-serializer: org.apache.kafka.common.serialization.StringSerializer
      value-serializer: org.springframework.kafka.support.serializer.JsonSerializer
      properties:
        spring.json.add.type.headers: false  # 타입 헤더 추가 비활성화
```

application.yml

backend-user : application-dev.yml Kafka 접속 정보 입력

```yaml
server:
  port: 8080

spring:
  datasource:
    url: jdbc:mysql://k8s-external-user-mysql-
service:3306/user?serverTimezone=UTC&useSSL=true&autoReconnect=true&useUnicode=true&characterEncoding=utf-8
    username: user
    password: 1234
    driver-class-name: com.mysql.cj.jdbc.Driver
    hikari:
      connection-test-query: SELECT 1 # HikariCP 유효성 검사 추가
          validation-timeout: 5000
  jpa:
    hibernate:
      ddl-auto: create # 오직 테스트 환경에서만
        generate-ddl: true # 오직 테스트 환경에서만 (spring.jpa.generate-ddl)
    show-sql: true
    open-in-view: false

  kafka:
    bootstrap-servers: k8s-external-user-kafka-service:9092
```

application-dev.yml

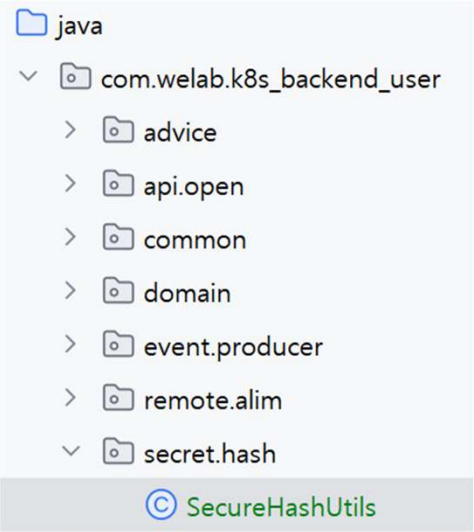
backend-user : application-local.yml Kafka 접속 정보 입력

```yaml
server:
  port: 8080

spring:
  datasource:
    url: jdbc:mysql://k8s-external-user-mysql-
service:3306/user?serverTimezone=UTC&useSSL=true&autoReconnect=true&useUnicode=true&characterEncoding=utf-8
    username: user
    password: 1234
    driver-class-name: com.mysql.cj.jdbc.Driver
    hikari:
      connection-test-query: SELECT 1 # HikariCP 유효성 검사 추가
          validation-timeout: 5000
  jpa:
    hibernate:
      ddl-auto: create # 오직 테스트 환경에서만
        generate-ddl: true # 오직 테스트 환경에서만 (spring.jpa.generate-ddl)
    show-sql: true
    open-in-view: false
  kafka:
    bootstrap-servers: 192.168.0.135:9092
#    bootstrap-servers: k8s-external-user-mysql-service:9092
```

자신의 IP

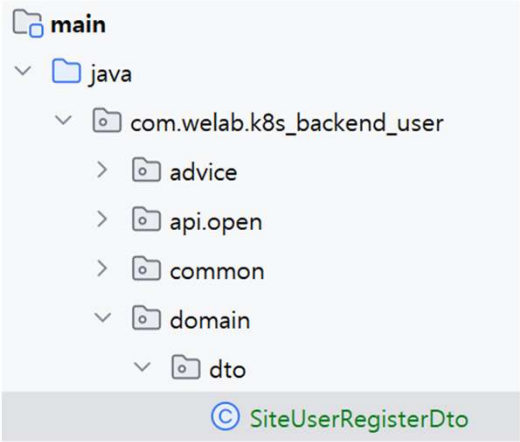
application-dev.yml

backend-user : SecureHashUtils 추가



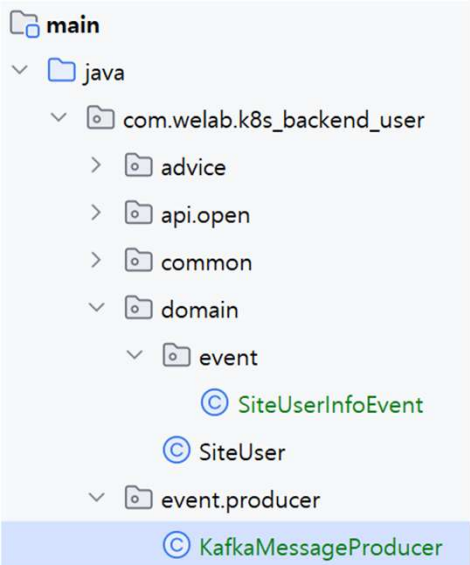

```java
public class SecureHashUtils {
    public static String hash(String message) {
        // TODO: message -> SHA-1 또는 SHA-256 해시 값으로 변환

            return message;
    }

    public static boolean matches(String message, String hashedMessage) {
        String hashed = hash(message);

        return hashed.equals(hashedMessage);
    }
}
```

backend-user : SiteUserRegisterDto 추가



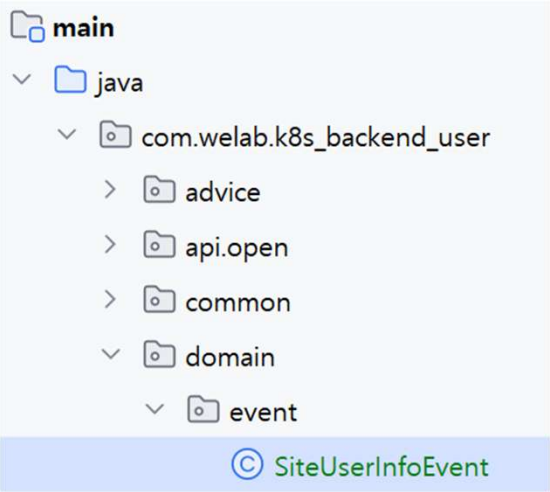

```java
@Getter
@Setter
public class SiteUserRegisterDto {
    @NotBlank(message = "아이디를  입력하세요.")
    private String userId;

    @NotBlank(message = "비밀번호를  입력하세요.")
    private String password;

    @NotBlank(message = "전화번호를  입력하세요.")
    private String phoneNumber;

    public SiteUser toEntity() {
        SiteUser siteUser = new SiteUser();

        siteUser.setUserId(this.userId);
        siteUser.setPassword(SecureHashUtils.hash(this.password));
        siteUser.setPhoneNumber(this.phoneNumber);

        return siteUser;
    }
}
```

backend-user : KafkaMessageProducer 추가



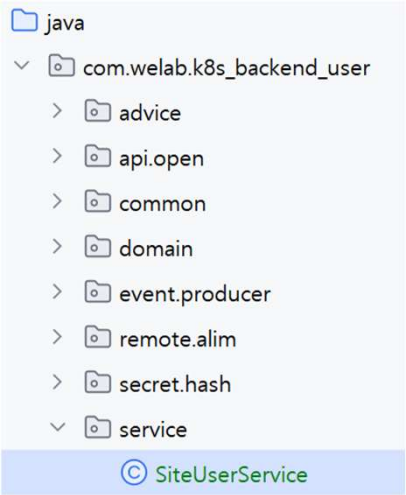

```java
@Slf4j
@Service
@RequiredArgsConstructor
public class KafkaMessageProducer {
    private final KafkaTemplate<String, Object> kafkaTemplate;

    public void send(String topic, Object message) {
        kafkaTemplate.send(topic, message);
    }
}
```

backend-user : SiteUserInfoEvent 추가 (구조 변경이 힘들기 때문에 처음부터 잘 구성하도록 노력)




```java
@Getter
@Setter
public class SiteUserInfoEvent {
    public static final String Topic = "userinfo";

    private String action;

    private String userId;

    private String phoneNumber;

    private LocalDateTime eventTime;

    public static SiteUserInfoEvent fromEntity(String action, SiteUser siteUser) {
        SiteUserInfoEvent event = new SiteUserInfoEvent();

        event.action = action;
        event.userId = siteUser.getUserId();
        event.phoneNumber = siteUser.getPhoneNumber();
        event.eventTime = LocalDateTime.now();

        return event;
    }
}
```

backend-user : SiteUserService 추가

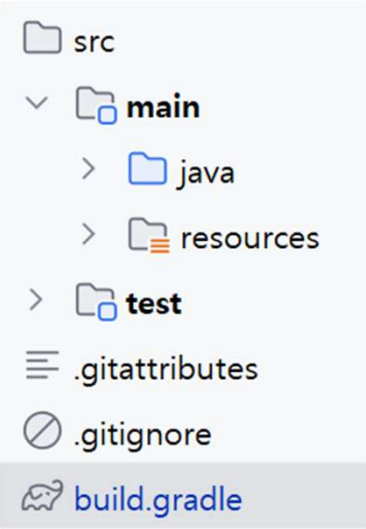



```java
@Slf4j
@Service
@RequiredArgsConstructor
public class SiteUserService {
    private final SiteUserRepository siteUserRepository;
    private final KafkaMessageProducer kafkaMessageProducer;

    @Transactional
    public void registerUser(SiteUserRegisterDto registerDto) {
        SiteUser siteUser = registerDto.toEntity();

        siteUserRepository.save(siteUser);

        SiteUserInfoEvent event = SiteUserInfoEvent.fromEntity("Create", siteUser);
        kafkaMessageProducer.send(SiteUserInfoEvent.Topic, event);
    }
}
```

backend-user : UserAuthController 추가


```java
@Slf4j
@RestController
@RequestMapping(value = "/api/user/v1/auth", produces = MediaType.APPLICATION_JSON_VALUE)
@RequiredArgsConstructor
public class UserAuthController {
    private final SiteUserService siteUserService;

    @PostMapping(value = "/register")
    public ApiResponseDto<String> register(@RequestBody @Valid SiteUserRegisterDto registerDto) {
        siteUserService.registerUser(registerDto);
        return ApiResponseDto.defaultOk();
    }
}
```

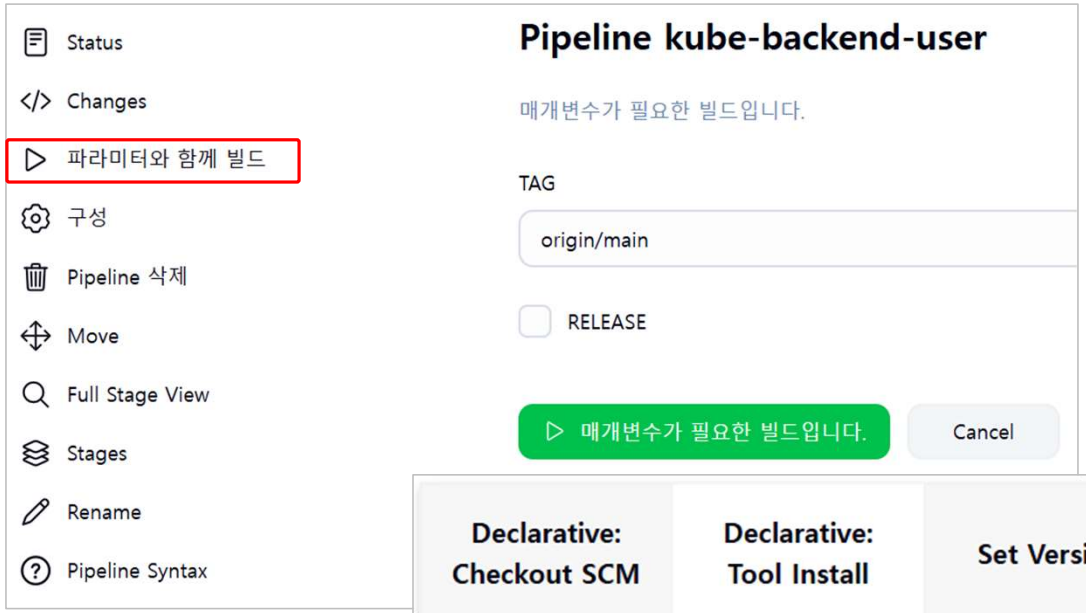
backend-user : 앱 버전 업그레이드




```
plugins {
    id 'java'
    id 'org.springframework.boot' version '3.5.0'
    id 'io.spring.dependency-management' version '1.1.7'
}

group = 'com.welab'
version = '0.0.4'

…
```

Commit & Push to Github

backend-user : Jenkins Pipeline 빌드



Declarative: Checkout SCM	Declarative: Tool Install	Set Version	Build & Test Application	Build Docker Image	Push Docker Image
4s	367ms	1min 14s	49s	7s	11s
4s	367ms	1min 14s	49s	7s	11s

backend-user : Deployment 이미지 버전 업데이트

Deployments

이름	이미지	레이블	파드	생성 시간 ↑
k8s-backend-alim-deployment	solarhc/k8s-backend-alim:0.0.1	-	1 / 1	16 minutes ago
k8s-backend-user-deployment	solarhc/k8s-backend-user:0.0.1	-	1 / 1	2 hours ago
arhc/k8s-api-gateway:0.0.3		-	1 / 1	2 hours

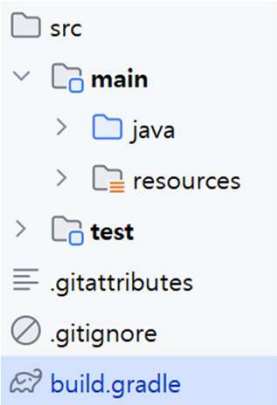
Scale
Edit
Restart
Delete

```yaml
spec:
  replicas: 1
  selector:
    matchLabels:
      app: k8s-backend-user
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: k8s-backend-user
    spec:
      containers:
        - name: k8s-backend-user
          image: solarhc/k8s-backend-user: 0.0.4
          ports:
```

이 액션은 다음 커맨드와 동일합니다. `kubectl apply -f <spec.yaml>`

Update Cancel

backend-alim : build.gradle 의존성 추가




```
dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-web'
    implementation 'org.springframework.cloud:spring-cloud-starter-openfeign'

    implementation 'org.springframework.kafka:spring-kafka'
    testImplementation 'org.springframework.kafka:spring-kafka-test'

    compileOnly 'org.projectlombok:lombok'
    annotationProcessor 'org.projectlombok:lombok'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
    testRuntimeOnly 'org.junit.platform:junit-platform-launcher'
}
```

backend-alim : application.yml Kafka 설정

```yaml
spring:
  application:
    name: k8s-backend-alim

  kafka:
    listener:
      ack-mode: manual_immediate
    consumer:
      group-id: ${spring.application.name}
      key-deserializer: org.apache.kafka.common.serialization.StringDeserializer
      value-deserializer: org.springframework.kafka.support.serializer.JsonDeserializer
      enable-auto-commit: false
      auto-offset-reset: latest
      max-poll-records: 10
      properties:
        spring.json.trusted.packages: "*"
        spring.json.use.type.headers: false  # 헤더의 타입 정보 무시
         producer:
      key-serializer: org.apache.kafka.common.serialization.StringSerializer
      value-serializer: org.springframework.kafka.support.serializer.JsonSerializer
      properties:
        spring.json.add.type.headers: false  # 타입 헤더 추가 비활성화
```

application.yml

backend-alim : application-dev.yml Kafka 접속 정보 입력

```yaml
server:
  port: 8080

spring:
  kafka:
    bootstrap-servers: 192.168.0.135:9092
#    bootstrap-servers: k8s-external-user-mysql-service:9092
```

자신의 IP

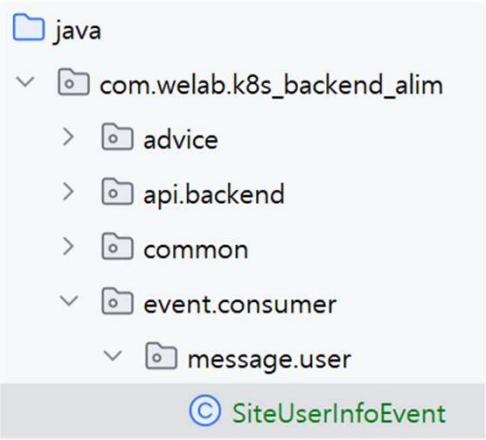
application-dev.yml

backend-alim : application-local.yml Kafka 접속 정보 입력

```yaml
server:
  port: 8080

spring:
  kafka:
    bootstrap-servers: localhost:9092
```

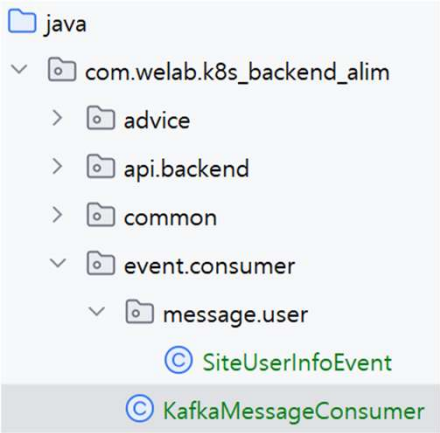
application-dev.yml

backend-alim : application-local.yml Kafka 접속 정보 입력



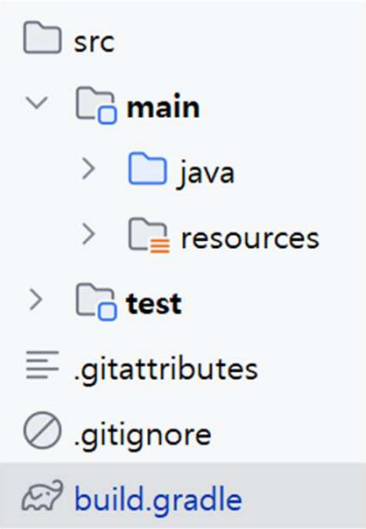

```java
@Getter
@Setter
public class SiteUserInfoEvent {
    public static final String Topic = "userinfo";

    private String action;

    private String userId;

    private String phoneNumber;

    private LocalDateTime eventTime;
}
```

backend-alim : application-local.yml Kafka 접속 정보 입력




```
@Slf4j
@Service
@RequiredArgsConstructor
public class KafkaMessageConsumer {
    @KafkaListener(
            topics = SiteUserInfoEvent.Topic,
            properties = {
                    JsonDeserializer.VALUE_DEFAULT_TYPE
                            + ":com.welab.k8s_backend_alim.event.consumer.message.user.SiteUserInfoEvent"
            }
    )
    void handleSiteUserInfoEvent(SiteUserInfoEvent event, Acknowledgment ack) {
        log.info("SiteUserInfoEvent 처리. userId={}", event.getUserId());

        ack.acknowledge();
    }

}
```

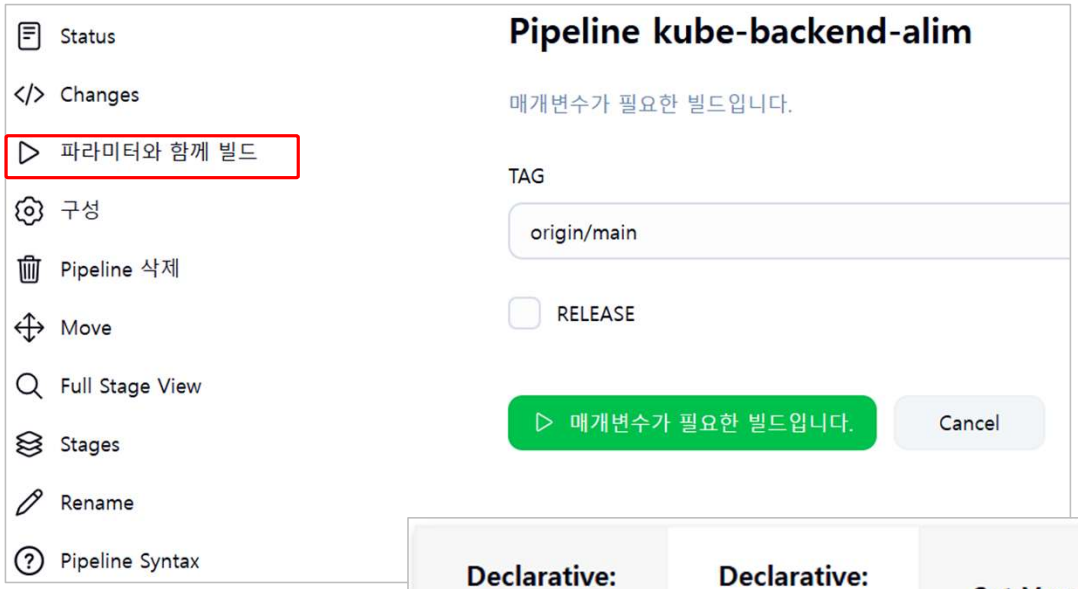
backend-alim : 앱 버전 업그레이드




```
plugins {
    id 'java'
    id 'org.springframework.boot' version '3.5.0'
    id 'io.spring.dependency-management' version '1.1.7'
}

group = 'com.welab'
version = '0.0.2'

…
```

Commit & Push to Github

backend-alim : Jenkins Pipeline 빌드



Declarative: Checkout SCM	Declarative: Tool Install	Set Version	Build & Test Application	Build Docker Image	Push Docker Image
4s	367ms	1min 14s	49s	7s	11s
4s	367ms	1min 14s	49s	7s	11s

backend-alim : Deployment 이미지 버전 업데이트

Deployments

이름	이미지	레이블	파드	생성 시간 ↑
k8s-backend-alim-deployment	solarhc/k8s-backend-alim:0.0.1	-	1 / 1	16 minutes ago
k8s-backend-user-deployment	solarhc/k8s-backend-user:0.0.1	-	1 / 1	2 hours ago
k8s-api-gateway-deployment	solarhc/k8s-api-gateway:0.0.3	-	1 / 1	2 hours

Scale
Edit
Restart
Delete

```yaml
spec:
  replicas: 1
  selector:
    matchLabels:
      app: k8s-backend-alim
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: k8s-backend-alim
    spec:
      containers:
        - name: k8s-backend-alim
          image: solarhc/k8s-backend-alim:0.0.2
          ports:
```

이 액션은 다음 커맨드와 동일합니다. `kubectl apply -f <spec.yaml>`

Update Cancel

kafka docker-compose.yml 수정 및 재시작

```yaml
version: "2"
services:
  kafdrop:
    image: obsidiandynamics/kafdrop:3.31.0
    restart: "always"
    ports:
      - "9000:9000"
    environment:
      KAFKA_BROKERCONNECT: "kafka:29092"
      JVM_OPTS: "-Xms16M -Xmx48M -Xss180K -XX:-TieredCompilation -XX:+UseStringDeduplication -noverify"
    depends_on:
      - "kafka"
  kafka:
    image: obsidiandynamics/kafka
    restart: "always"
    ports:
      - "2181:2181"
      - "9092:9092"
    environment:
      KAFKA_LISTENERS: "INTERNAL://:29092,EXTERNAL://:9092"
      KAFKA_ADVERTISED_LISTENERS: "INTERNAL://kafka:29092,EXTERNAL://192.168.0.135:9092"
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: "INTERNAL:PLAINTEXT,EXTERNAL:PLAINTEXT"
      KAFKA_INTER_BROKER_LISTENER_NAME: "INTERNAL"
      KAFKA_ZOOKEEPER_SESSION_TIMEOUT: "6000"
      KAFKA_RESTART_ATTEMPTS: "10"
      KAFKA_RESTART_DELAY: "5"
      ZOOKEEPER_AUTOPURGE_PURGE_INTERVAL: "0"
```

자신의 IP

```
PS C:₩server₩kafka> docker compose down -v
PS C:₩server₩kafka> docker compose up -d
```