Docker와 Kubernetes 기반 ETL 자동화 및 대시보드 구현

한국폴리텍 강서캠퍼스

2025

스마트 금융과 노유성

목차

프로젝트 배경

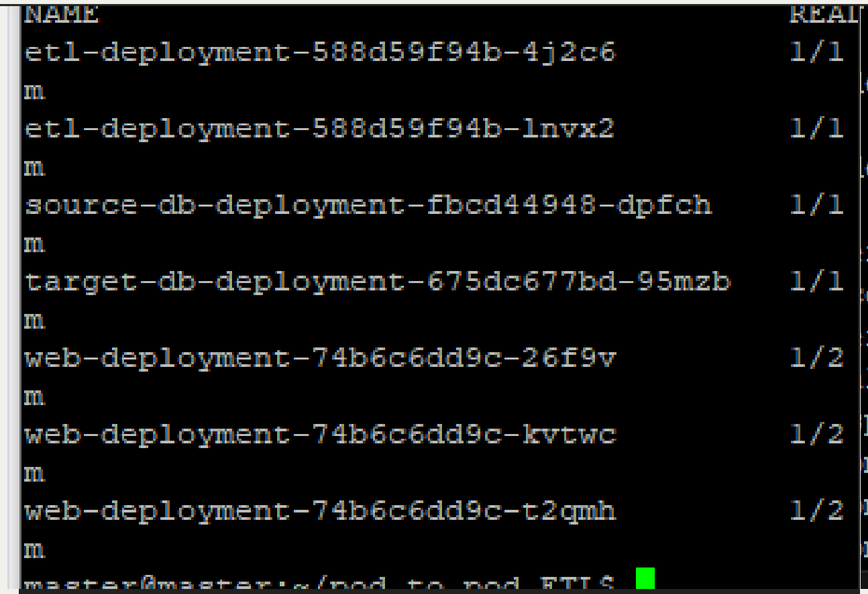
Crontab을 이용한 기존 ETL 작업의 자가 복구성 부족

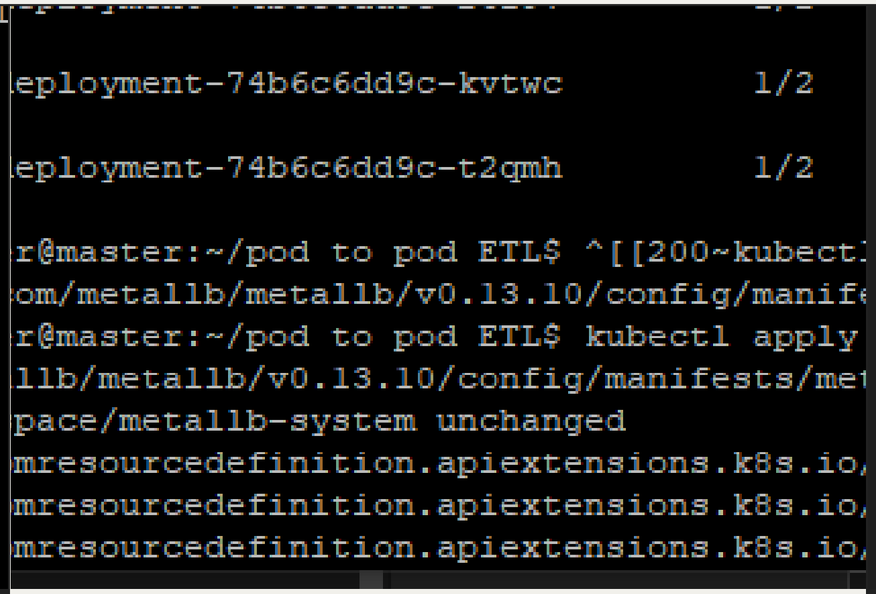
-서비스의 일관성을 위해 지속적으로 실행되어야할 ETL작업이 단순 ubuntu환경에서의 crontab과 sh파일을 이용한 자동화 구조로는 서버가 다운될 시 수동으로 복구를 해줘야한다는 불완전성이 존재
-쿠버네티스의 레플리카셋과 pod을 이용하면 자동으로 복구가 되기 때문에 서비스 단계에서 안전성을 확보 가능
-ETL 작업자체를 별도 컨테이너에서 실행함으로서 리소스 충돌이나 DB오염을 최소화

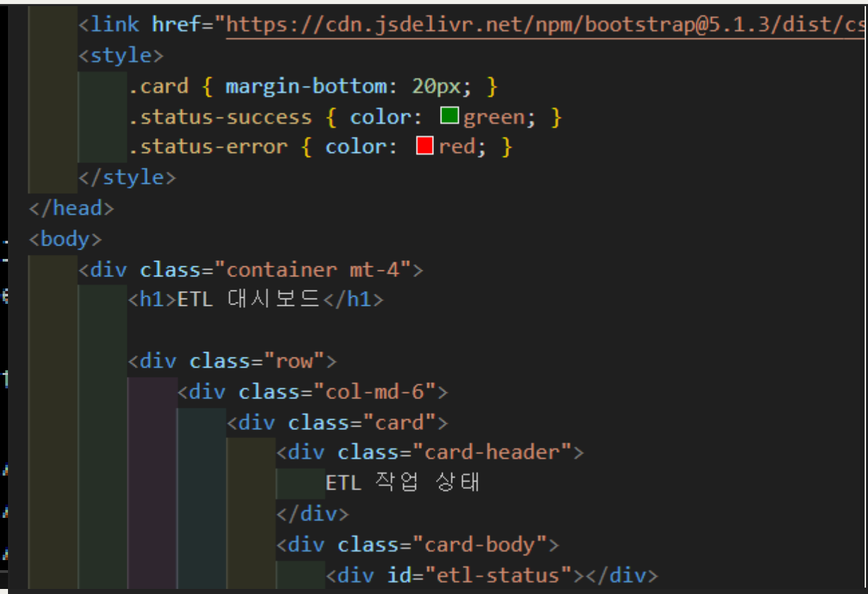
재현성의 부족으로 인한 데이터 통합 문제

-python환경, 라이브러리 버전, os설정 등에 따라 ETL 작업의 실패 가능성이 존재
-docker로 데이터셋을 컨테이너화하여 빌드된 이미지가 어디서든 실행할 수 있게 하여 서비스의 일관성과 재현성을 확보
-DB를 pvc화하여 컨테이너가 재시작 혹은 재배포되더라도 데이터를 안전하게 보존 가능

프로젝트 목표







ETL 작업의 지속성과 재현성 확보

-데이터가 이관될 pod 내부 pvc 빌드 후 데이터 베이스 생성 및 관리하여 데이터 보안성 확보

-데이터를 가공할 때마다 새로운 pod와 deployment 생성하여 각 자료의 독립성 유지

-ETL 작업의 지속성을 위해 레플리카셋을 통해 2개의 파드를 묶어 이중화하여 하나의 POD가 다운되더라도 다른 하나가 자동생성함으로 서비스의 안전성을 확보 가능

LoadBalancer활용 웹 과부하방지

-MetalLB통한 로컬 LoadBalancer 생성

-서버 과부하를 방지 및 데이터 분산화를 통한 안전성 확보

-포트 직접 노출없이 연결이 가능하므로 무결성을 확보할 수 있고, IP할당할 필요도 따로 없이 공인 IP를 제공 받기때문에 다른 서비스와 빠른 호환성이 기대. 데이터 보관 후 전처리 과정이 필수적인 ETL프로젝트에서 효과적

Ejs활용 Web 대시보드 기능 구성

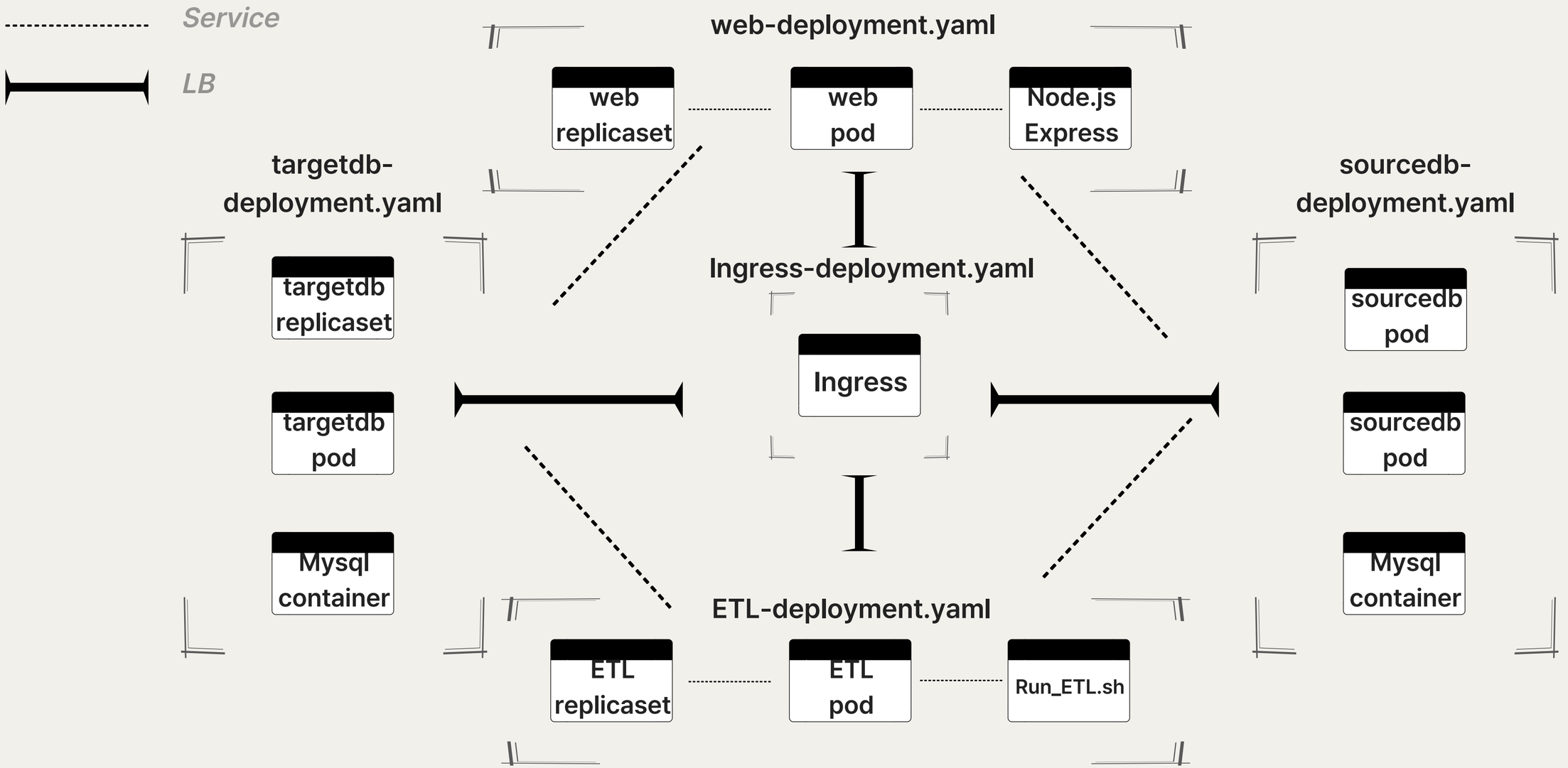
-ETL프로세스를 통해 정제된 데이터를 검색, 분석, 변별 하기위한 대시보드 기능

-Node.js + Express.js 통해 Api 처리 및 라우팅 제어

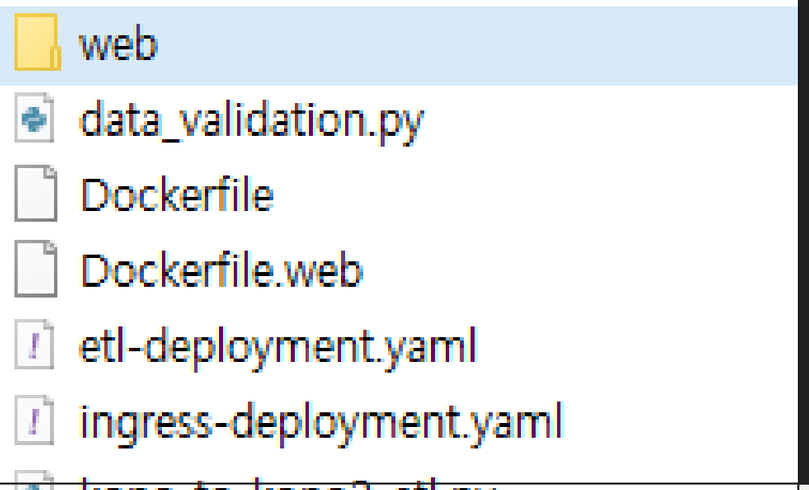
-ETL 파이프라인 상태 모니터링 위한 웹 기반 관제 시스템 구축

-분류기준별 시멘틱 구조로 정렬하여 한눈에 데이터를 알아보기 쉽게 구성

프로젝트 구조 MERMAID 다이어그램



Yaml파일 명세



1.etl-deployment.yaml

-Python 기반 ETL 컨테이너 배포
-주기적으로 Github으로 부터 위해상품 Exel 데이터를 수집
-Source DB로 이관. 무한루프 구조
-장애발생시 리스케쥴링
-run_etl.sh 실행으로 검증 로그 축적

3.source-db-deployment.yaml

-ETL 데이터 저장 목적 MySQL 데이터베이스
-Github에서 원천데이터 적재 PVC 마운트
-Excel → Pandas → SQL 단계를 거쳐 로컬 저장
-Pod 재시작 시에도 보존 가능

2. web-deployment.yaml

-Node.js 기반 Express 서버를 통해 사용자 대시보드 시각적 조회 목적 웹 UI 인터페이스 제공
-dashboard.ejs 동적 랜더링
-사용자가 Ingress를 통해 접근하여 web-service 통해 pod접근
-Express서버와 targetDB(MySQL) 연결

5. ingress-deployment.yaml

-외부 사용자의 웹 접속을 가능하게 하는 라우터 역할. 트래픽을 특정 서비스로 전달
-MetalLB 설치하여 로컬환경 LoadBalancer 가동
-내부 DNS 기반 K8s Service 연결

4.target-db-deployment.yaml

-souceDB에 저장된 원천데이터 가공(위해상품_위험도_분석, 위해상품_데이터셋) 및 저장
-웹 Dashboard의 Feature Source
-PVC 마운트하여 데이터 유실 방지
-REST API 응답에 사용

ETL 로직 실행

```
start_time = datetime.now()
logger.info("ETL 프로세스 시작")

# 처리할 Excel 파일 목록
EXCEL_FILES = {
    '위해상품': 'https://raw.githubusercontent.com/n
    '위해상품부적합검사': 'https://raw.githubusercont
    '위해상품업체': 'https://raw.githubusercontent.c
}

# 데이터베이스 연결 설정
LOCAL_DB_CONFIG = {
    'host': os.getenv('SOURCE_DB_HOST', '10.110.126
    'port': int(os.getenv('SOURCE_DB_PORT', '3306')
    'user': os.getenv('SOURCE_DB_USER', 'root'),
```

kopo_to_kopo2_etl.py , rush.sh 파일 실행

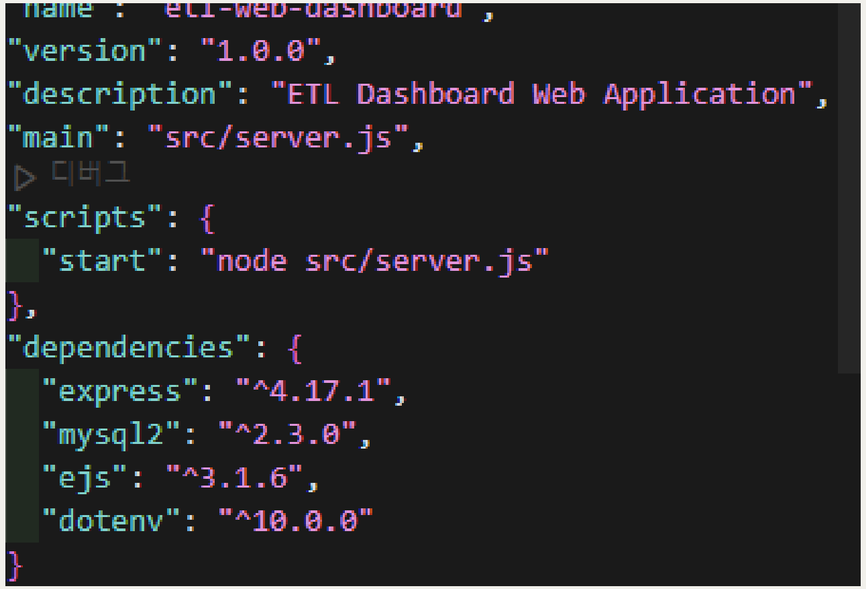
-원천 데이터 위치인 github raw파일 위치 경로 적어줌
-기존에 쓰던 local to ec2 ETL 구조를 제거하고, souce_deployment.yaml과 target_deployment.yaml에서 생성된 Mysql 8.0 버전의 환경변수나 서비스 명세에서의 cluster ip 를 적어준다.

```
master@master:~/pod to pod ETL$ ^C
master@master:~/pod to pod ETL$ kubectl exec -it target-d
5mzb -- bash
bash-5.1# locale
LANG=C.utf8
LC_CTYPE="C.utf8"
LC_NUMERIC="C.utf8"
LC_TIME="C.utf8"
LC_COLLATE="C.utf8"
LC_MONETARY="C.utf8"
LC_MESSAGES="C.utf8"
LC_PAPER="C.utf8"
LC_NAME="C.utf8"
LC_ADDRESS="C.utf8"
LC_TELEPHONE="C.utf8"
LC_MEASUREMENT="C.utf8"
LC_IDENTIFICATION="C.utf8"
LC_ALL=C.utf8
bash-5.1#
```

targetDB pod와 sourceDB pod 들어가서 locale 변환

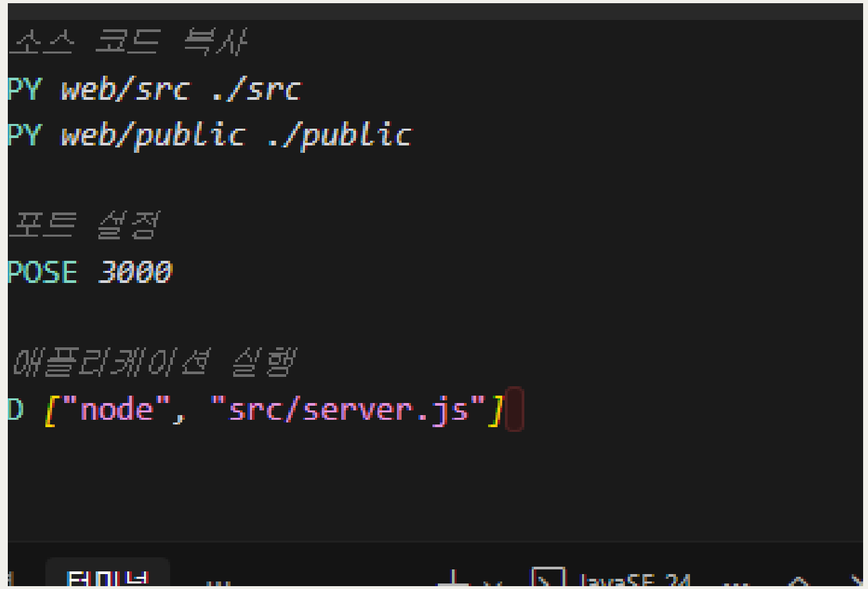
-처음 pvc로 db를 만들면 팟 내부의 로케일이 POSIX로 고정되어있는데, 이럴 경우 ETL 진행과정에서 DB가 칼럼의 글자를 인식하지 못하는 오류가 발생
-반드시 UTF8 계열로 바꾸고 진행

웹 서버 만들고 도커 이미지 push



웹 서버 실행과 엔드포인트 설정

-Python Flask, Express.js
-대시보드 UI 생성(HTML, CSS, JavaScript)하고 ejs 로 랜더링
targetDB의 데이터를 읽어오는 API 엔드포인트 설정
-/api/etl-status로 ETL상태조회, /api/dangerous-products 로 위해상품 데이터 조회



도커 이미지 만들고 push하기

-웹 애플리케이션을 Docker 이미지로 패키징
-Dockerfile 작성
-이미지 빌드 및 테스트

Kubernetes 리소스 생성

```yaml
apiVersion: v1
kind: Service
metadata:
  name: web-service
  namespace: default
spec:
  selector:
    app: web
  ports:
  - port: 80
    targetPort: 80
  type: NodePort
```

웹 관련 명세를 정리한 web-deployment.yaml apply

-Deployment 생성
-Service 생성해서 Cluster IP 확인
-외부접근용 객체인 Ingress 객체 생성하고 metalLB다운로드
-kubectl get all or kubectl get services 명령어로 작동 확인

```
r@master:~/pod to pod ETL$ ^C
r@master:~/pod to pod ETL$ kubectl get s
                  TYPE        CLUSTER-IP
ervice            ClusterIP   10.109.107.30
netes             ClusterIP   10.96.0.1
e-db-service      ClusterIP   10.110.126.86
t-db-service      ClusterIP   10.98.128.61
ervice            NodePort    10.100.92.194
r@master:~/pod to pod ETL$
r@master:~/pod to pod ETL$ kubectl get c
                  DATA   AGE
root-ca.crt       1      11h
lb-excludel2      1      11h
```

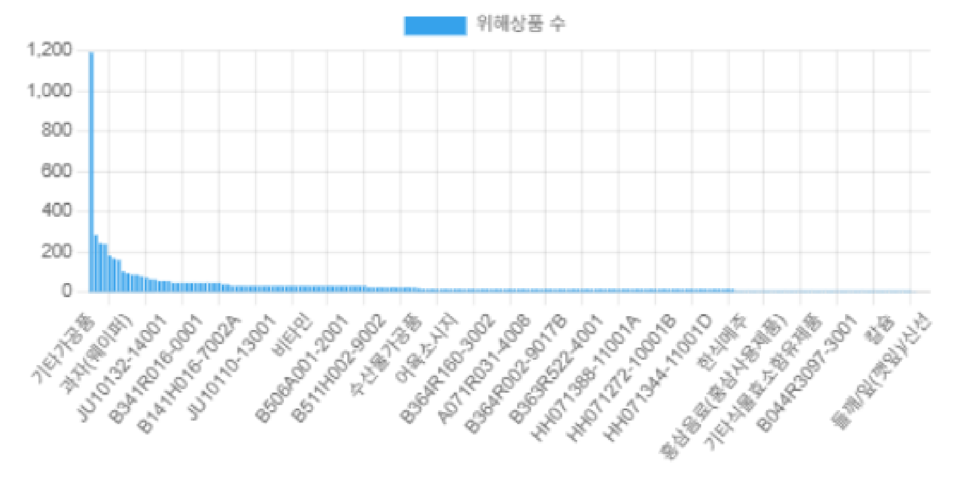
서버 연결 및 LoadBalancer 및 service 통신 테스트

-웹 어플리케이션 테스트
-MetalLB 설정 확인
-Ingress 설정 테스트
-대시보드 접속 테스트

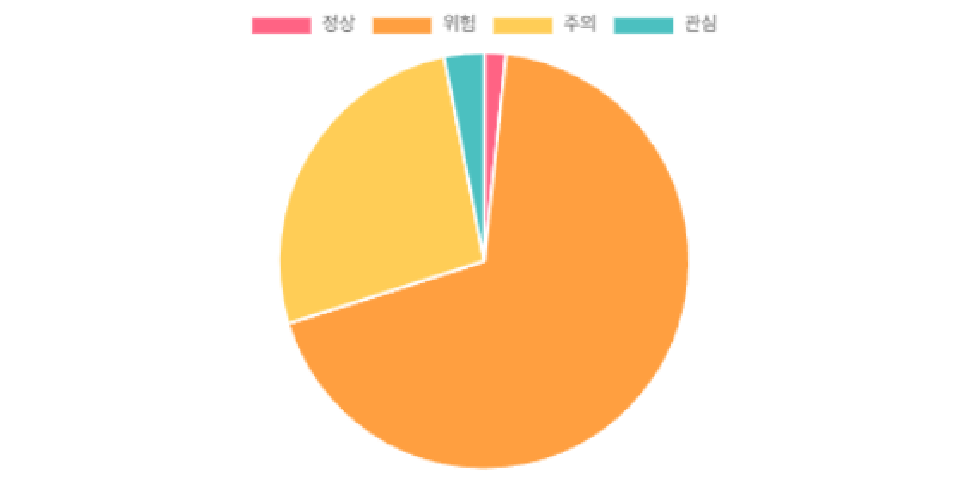
산출물 및 결과

위해상품 모니터링 대시보드

제품유형별 위해상품 현황



위험도등급 분포

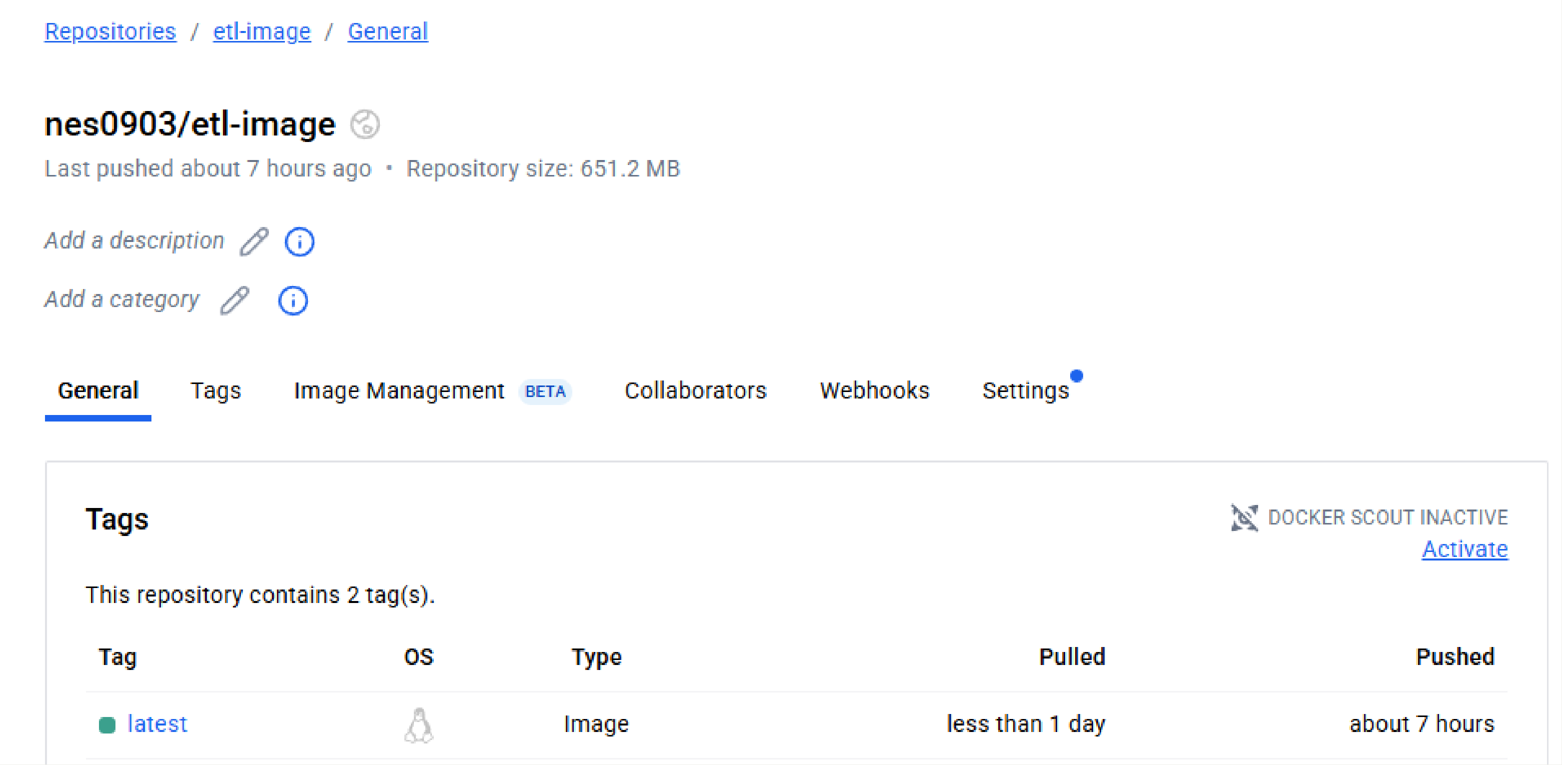


위해상품 목록

제품명	제품유형	제조국	위험도등급	위해상품여부	회수조치내용
얼티메이트 오메가-3	오메가-3지방산함유유지 제품	미국	관심	예	기타
얼티메이트 오메가-3	오메가-3지방산함유유지 제품	미국	관심	예	기타
얼티메이트 오메가-3	오메가-3지방산함유유지 제품	미국	관심	예	기타
얼티메이트 오메가-3	오메가-3지방산함유유지 제품	미국	관심	예	기타
얼티메이트 오메가-3	오메가-3지방산함유유지 제품	미국	관심	예	기타
얼티메이트 오메가-3	오메가-3지방산함유유지 제품	미국	관심	예	기타
얼티메이트 오메가-3	오메가-3지방산함유유지 제품	미국	관심	예	기타
얼티메이트 오메가-3	오메가-3지방산함유유지 제품	미국	관심	예	기타
얼티메이트 오메가-3	오메가-3지방산함유유지 제품	미국	관심	예	기타

도커, 쿠버네티스 기반 ETL 데이터셋 대시보드 웹페이지

-Express.js + EJS 기반의 SSR 웹 서비스 구현
-도표, 분석 기능을 추가해 보기 쉽고, 쿠버네티스 자동화 프로세스에 따라 변화를 감지하기 쉽게 의도
-DockerHub를 통한 버전 관리 및 배포 가능한 이미지 생성
-환경 의존성 제거한 즉시 실행가능한 컨테이너 기반 인프라 형성
-PVC활용한 데이터 영속성 보장 서비스
-생산형 데이터 분석 플랫폼



한계점 및 개선방안

데이터셋의 도메인 종속성

한계점: 위해 상품이라는 특정 도메인 최적화 서비스로서 다른 도메인으로서의 범용성 및 재사용성 낮음
고정적 스키마 구조로인해 데이터 소스 변경 시 ETL 파이프라인 전체 수정이 필요

개선방안: 테이블-컬럼 정의 메타데이터 기반 추상화 필요. 파이프라인 유연성 심화필요. 도메인 일반화를 위해 다양한 분야를 적용할 수 있도록 탭을 추가하고 pod추가생성통해 서버 다중화작업 요망

Ingress 인증 및 보안 미적용

한계점: Ingress 구조는 트래픽을 효과적으로 분산 시킬 수 있지만, 현재 TLS 인증서나 인증 미들웨어가 구성되어 있지 않아 전송 계층 보안이 미흡한 상태. 외부 공격에 취약

개선방안: Enscrypt 기반 TLS 인증서 자동 발급과 갱신을 적용하여 HTTPS 기반 Ingress 구성
JMT 기반 인증 게이트 웨이와 연동하여 접근 제어 로직 추가

이미지 버전 관리 및 Rollback 포인트 부재

한계점: Docker 이미지에 latest 태그 또는 수동 버전 관리 사용 중이라 과거 상태로의 복원이 어렵고 배포 이력 추적 불가
잘못된 버전 배포시 재해 복구 불가

개선방안: Git 커밋 해시 또는 CI 배포 ID 기반으로 정형화된 이미지 버전 태깅 도입
Helm rollback 적용하여 자동화된 버전 롤백 체계 구축

민감 정보에 대한 정보 보호 미흡

한계점: 위해 상품 데이터에 업체명, 사업자 번호 등 민감 정보가 포함되어있음
법적 문제의 가능성,법인 도용, 위치 추적, 제품 이미지 손상에 따른 명예훼손 법적 공방 예상

개선방안:API 응답 필드에 대한 조건부 마스킹, IP 접근 제어, RBAC 기반 역할별 열람 권한 구분, 데이터 처리, 접근자 판별 감사 로직

감사합니다