2024 년 전기 졸업과제 착수보고서

MSA 기반의 클라우드 티켓팅 플랫폼 개발

팀 명: clove

202125292 조용진

201824410 강찬석

201824543 이동현

지도교수: 염근혁 (인)

목차

1. 과제의 목표

1) 과제 배경

마이크로서비스 아키텍처(MSA)는 시스템을 작은 독립적인 서비스들로 분리하고, 각 서비스는 모듈 또는 프로젝트 단위로 나누어 개발 및 관리를 진행하는 소프트웨어 아키텍처이다. 각 서비스는 느슨하게 결합(Loosely Coupling)되어 다른 서비스와의 의존성이 최소화된다는 특징이 있다.

이러한 특징을 바탕으로 마이크로서비스는 서비스 간 독립성이 보장된다. 서비스 독립성의 특성 상 일부 서비스에서 실패 지점이 발생하더라도 전체 시스템에 큰 영향을 미치지 않는다. 실패가 발생한 서비스에 대해서만 복구 작업을 수행하면 서비스 회복에도 빠르게 대처할 수 있다는 장점이 있다. 또한 서비스의 확장성과 유연성이 높아진다는 장점이 있다.

티켓 예매 시스템의 사용자는 공연을 주최하여 티켓을 판매하는 판매자, 티켓을 구매하는 구매자로 나뉜다. 현재 운영되고 있는 티켓 예매 시스템은 티켓 판매 회사와 시스템 운영 회사 간의 계약서에 기반하여 위탁 판매 형태를 이루며, 적게는 수일에서 많게는 수십일에 걸쳐 계약-판매-정산의 과정이 이행된다. 이러한, 복잡한 의뢰 과정은 티켓 판매자가 원하는 요구사항이 서비스에 신속하게 반영되기 어렵다는 단점이 있다.

따라서 본 과제에서는 티켓 판매자의 요구사항이 반영된 티켓 예매 시스템을 제공하기 위해 마이크로서비스 아키텍처가 반영된 티켓 예매 시스템 구축 방법을 제시한다. 본 과제는 1) 기 준비된 컨테이너 풀을 통해 티켓 시스템을 구축하는 컨테이너 기반 마이크로서비스 배포 지원, 2) 컨테이너 인프라 및 마이크로서비스의 모니터링을 통해 안정적인 컨테이너 운영 지원, 3) 티켓 판매자의 요구사항을 컨테이너 배포 시 반영할 수 있는 맞춤형 마이크로서비스 배포 시스템을 목표로 수행한다.

2) 과제 세부 목표

① 티켓 판매 마이크로서비스 제공

② 마이크로서비스 및 배포 인프라 모니터링

③ 배포 상태에 기반한 마이크로서비스 관리

2. 대상 문제 및 요구사항 분석

1) 유사 시스템 분석

① 멜론티켓

멜론티켓은 멜론에서 운영하는 대한민국의 대표적인 티켓 예매 플랫폼 중 하나이다. 콘서트, 뮤지컬 및 연극, 전시 또는 행사 등 다양한 공연과 행사들에 대해 예매 서비스를 제공한다. 공연 주최자 입장에서는 멜론티켓 담당자와 상담 후 판매 대행 계약서와 공연 등록 의뢰서를 작성하여 티켓 판매를 시작하며, 공연 종료 후 멜론티켓 측에서 판매 정산서를 수령하게 된다.

② 티켓링크

티켓링크는 NHN 에서 운영하는 티켓 예매 플랫폼으로 공연, 전시, 스포츠 경기, 여행 등의 예매 서비스도 제공하고 있다. 멜론티켓과 마찬가지로 공연 주최자가 티켓링크 담당자와 상담 후 판매 대행 의뢰서를 작성하여 티켓 판매를 시작한다. 이 때 의뢰서를 크게 공연, 전시, 스포츠 세 가지로 분류하여 각 분야에 필요한 정보를 기입하도록 유도하였다.

③ 모놀리식 기반 티켓팅 시스템

마이크로서비스 아키텍처를 적용하지 않은 모놀리식 시스템은 사용자 인터페이스 계층, 비즈니스 로직 계층, 데이터베이스 접근 계층 등으로 구성된 단일 프로젝트 패키지 형태이다. 이러한 구조의 티켓 예매 시스템은 중앙화 된 서버와 DB 를 가지며 이를 통해 사용자가 로그인하고, 티켓을 예매하는 등 상호작용한다.

2) 문제점 분석

① 멜론티켓

공연 주최자가 티켓 판매를 의뢰하기 위해서는 멜론티켓 담당자와 판매 조건 등 판매 계약에 대해 상담한 후 계약서를 작성한 뒤 공연 등록 의뢰서를 작성한다. 작성한 의뢰서를 담당자에게 발송하면 3~4 일 이내에 멜론티켓에 공연이 등록되며, 관리자 계정과 비밀번호를 수령하게 된다. 공연이 종료되면 7 일 이내에 멜론티켓에서 판매 정산서를 수령하게 된다.

이러한 과정에서 공연 등록 의뢰서 작성 후 공연 등록까지 소요되는 시간이 3~4 일이내, 공연 종료 후 판매 정산서 수령까지 소요되는 시간이 7 일 이내로 짧지 않다는 문제점이 있다.

② 티켓링크

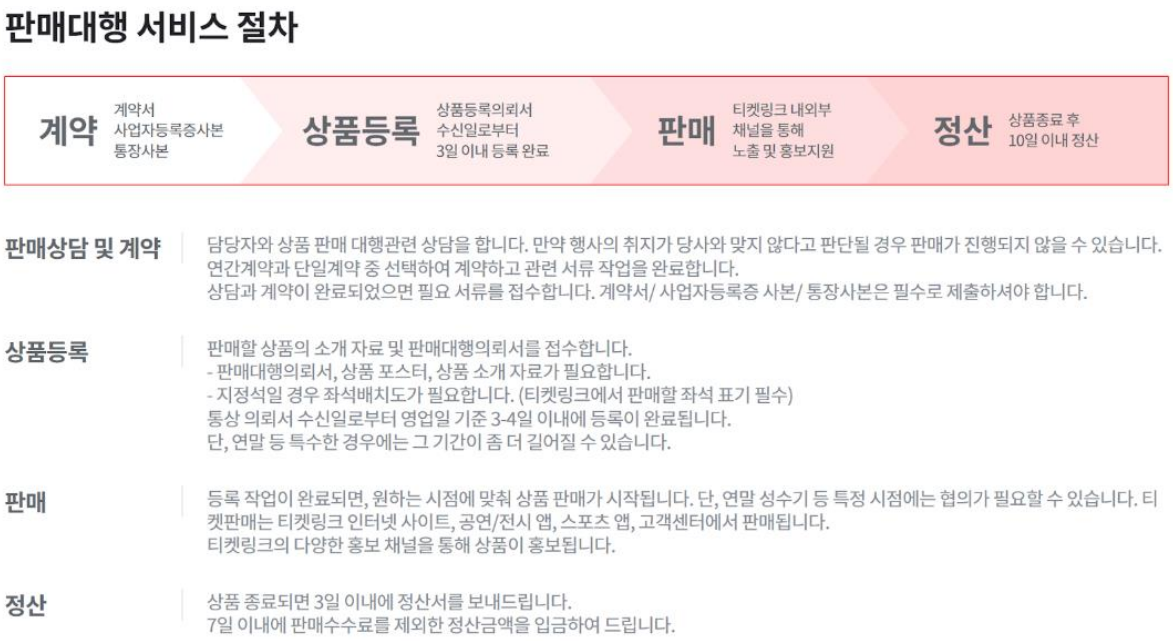


그림 1. 티켓링크의 판매대행 서비스 절차

그림 1 은 티켓링크의 판매대행 절차를 나타낸 것이다. 티켓링크는 의뢰서를 공연, 스포츠 등 분야별로 나누어 각 분야에 맞는 판매 의뢰자의 요구사항을 전달받아 판매대행 서비스를 진행한다.

그러나 이러한 방식은 공연, 전시, 스포츠 이외의 분야에 대한 요구사항을 반영하기에 부적절할 수 있다는 문제점과 의뢰서를 받은 티켓링크 측 담당자가 요구사항을 직접 시스템에 반영하여야 하므로 추가적인 인력이 소모되며, 이 과정에서 요구사항이 잘못 반영되는 경우가 생길 가능성이 있다는 문제점이 있다.

③ 모놀리식 기반 티켓팅 시스템

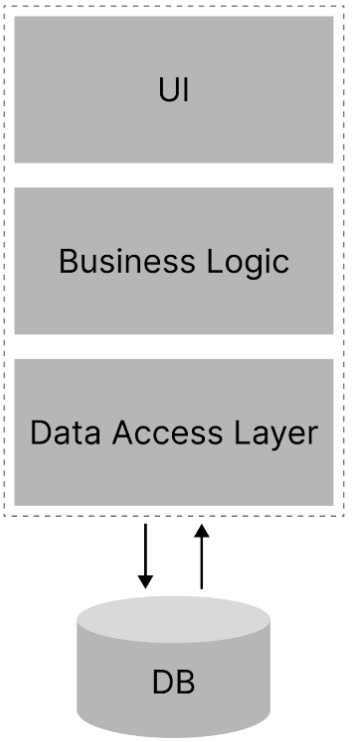
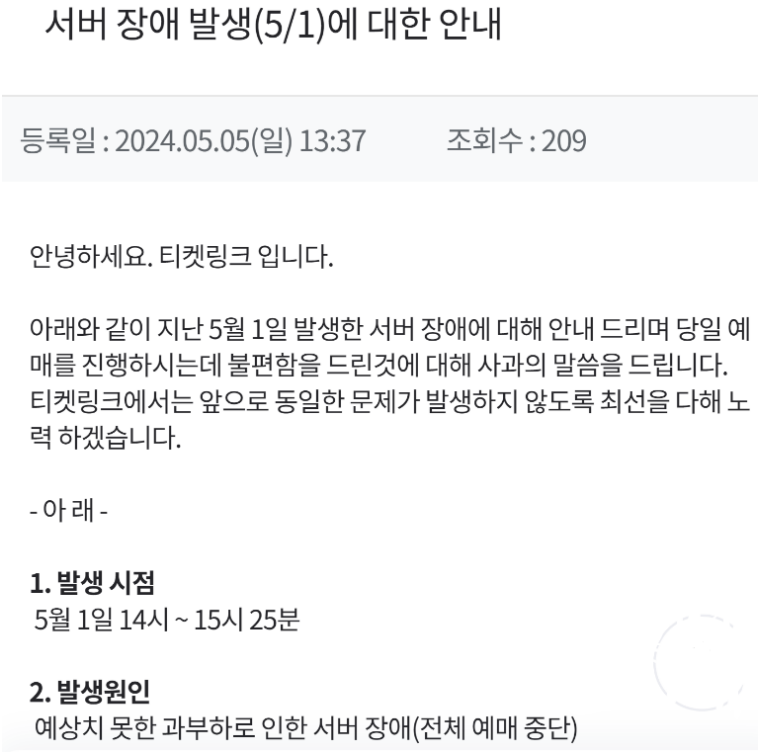


그림 2. 모놀리식 아키텍처 구조


서버 장애 발생(5/1)에 대한 안내

등록일 : 2024.05.05(일) 13:37 조회수 : 209

안녕하세요. 티켓링크 입니다.

아래와 같이 지난 5월 1일 발생한 서버 장애에 대해 안내 드리며 당일 예매를 진행하시는데 불편함을 드린것에 대해 사과의 말씀을 드립니다.
티켓링크에서는 앞으로 동일한 문제가 발생하지 않도록 최선을 다해 노력 하겠습니다.

- 아 래 -

1. 발생 시점
5월 1일 14시 ~ 15시 25분

2. 발생원인
예상치 못한 과부하로 인한 서버 장애(전체 예매 중단)

그림 3. 서버 오류로 인한 서비스 중단 사례

모놀리식 기반 티켓팅 시스템은 분산되지 않고 하나의 통합된 시스템을 가진다. 따라서 시스템의 일부분의 오류로 인해 전체 서비스가 영향을 받을 수 있다는 문제점이 있다.

3) 시스템의 필요성

① 티켓 판매 의뢰자의 요구사항을 유연하고 신속하게 반영할 수 있는 시스템이 필요하다.

- 관람 유형, 좌석 유형 등 공연 분야의 특성을 반영할 수 있는 컨테이너 풀을 구성하여 요구사항에 맞게 조합하여 배포
- 추가적인 인력 개입 없이 시스템이 자동으로 배포 구성

② 일부분의 오류가 전체에 영향을 끼치지 않는 독립된 시스템이 필요하다.

- 공연 별로 서버를 구성하여 다른 공연의 서비스 상태에 영향을 받지 않도록 시스템 구성

③ 전문적 지식이 없는 사용자에게도 서비스 모니터링 정보를 쉽고 빠르게 제공해 줄 수 있는 시스템이 필요하다.

- 배포한 공연 예매 서비스의 리소스 사용량을 모니터링할 수 있는 시스템 구성
- 리소스 사용량에 따른 요금 정보를 실시간으로 제공하는 시스템 구성

4) 요구사항 분석

① 기능적 요구사항

표 1 은 시스템이 제공해야 할 기능들에 대한 요구사항을 나타낸다.

표 1. 기능적 요구사항

기능		설명
시스템 접근 제어	회원가입	사용자 생성 기능이다. 회원가입시 판매자와 구매자로 구분되는 계정이 생성된다.
	로그인	사용자가 시스템 기능을 이용하기 위한 로그인 기능이다.
	로그아웃	사용자가 시스템 이용을 그만하기 위한 로그아웃 기능이다.
	회원탈퇴	생성한 사용자 정보를 영구 삭제하는 기능이다.
마이크로서비스 배포 및 운영	서비스 배포	판매자는 공연에 관한 간단한 요구사항 입력을 통해 자동화된 티켓 예매 서비스를 배포할 수 있다.
	요구사항 업데이트	판매자는 배포한 공연의 서비스에 대해 요구사항이 변경되었을 경우 해당 내용을 업데이트할 수 있어야 한다.
	배포 템플릿 구성	판매자의 요구사항 입력을 바탕으로 마이크로서비스를 배포하기 위한 템플릿을 생성한다.
	서비스 삭제	판매자는 자신이 관리하는 서비스의 배포를 중단할 수 있어야 한다.
	서비스 모니터링	판매자는 자신이 배포한 서비스의 리소스 사용량을 조회할 수 있어야 한다.
마이크로서비스 관리	마이크로서비스 이미지 추가	컨테이너 풀에 새로운 마이크로서비스의 이미지를 등록한다.
	마이크로서비스 이미지 목록 조회	컨테이너 풀에 등록되어 있는 마이크로서비스의 이미지를 조회한다.
	마이크로서비스 이미지 삭제	컨테이너 풀에 등록되어 있는 마이크로서비스의 이미지를 삭제한다.
	시스템 모니터링	시스템 관리자는 시스템 환경 전체에 대한 리소스 사용량을 조회할 수 있어야 한다.
	마이크로서비스 리소스 이상 탐지	마이크로서비스의 리소스 사용량 모니터링 정보를 바탕으로 임계치를 넘는 리소스 사용량이 일어나는 컨테이너를 탐지하는 기능이다.
	마이크로서비스 리소스 스케일링	마이크로서비스의 리소스를 확장 또는 축소하여 안정적인 서비스가 제공될 수 있도록 지원하는 기능이다.
	마이크로서비스 재구조화	마이크로서비스의 서비스간 재결합 또는 재배치를 통해 안정적인 서비스가 제공될 수 있도록 지원하는 기능이다.

티켓팅 서비스 이용	공연 목록 조회	구매자는 전체 공연의 목록을 조회할 수 있어야 한다.
	잔여 좌석 조회	구매자는 잔여 좌석 정보를 조회할 수 있어야 한다.
	공연 정보 조회	구매자는 공연에 대한 상세 정보를 조회할 수 있어야 한다.
	공연 예매	구매자는 잔여 좌석에 대해 예매를 할 수 있어야 한다.
	실시간 중복 예매 검사	구매자가 예매 정보를 입력하는 사이에 다른 사람이 같은 좌석에 대해 예매를 진행하는 것을 막기 위해 실시간 중복 예매 여부를 검사하는 기능이다.
	예매 티켓 조회	티켓 구매자는 자신이 예매한 티켓을 조회할 수 있다.
	티켓 결제	티켓 구매자는 예매한 티켓을 확정하기 위해 외부 시스템에 결제를 요청한다.
	티켓 환불	티켓 구매자가 결제를 완료한 확정된 티켓을 취소 및 환불받는 기능이다.
	예매자 목록 조회	판매자는 티켓을 예매한 구매자들의 정보를 확인할 수 있어야 한다.
	공연 정보 수정	판매자는 자신이 배포한 티켓팅 서비스의 정보를 수정할 수 있어야 한다.

② 비기능적 요구사항

다음 표 2 는 시스템이 만족해야 할 비기능적 요구사항이다.

표 2. 비기능적 요구사항

요건	설명
가용성	시스템은 24 시간, 365 일 내내 사용 가능해야 함
안전성	사용자 정보가 외부로 노출되지 않아야 함
성능	시스템은 많은 사용자가 접속하더라도 성능 저하 없이 사용자에게 서비스를 제공하여야 함
	동시에 여러 명이 같은 티켓을 예매하는 경우가 없어야 함
보안성	등록된 사용자만이 시스템에 접근할 수 있어야 함
	등록된 사용자의 역할(권한)에 맞는 동작만 수행할 수 있어야 함

5) 유스케이스 분석

① 유스케이스 다이어그램

그림 4 는 마이크로서비스 관리 시스템의 유스케이스 다이어그램이다.

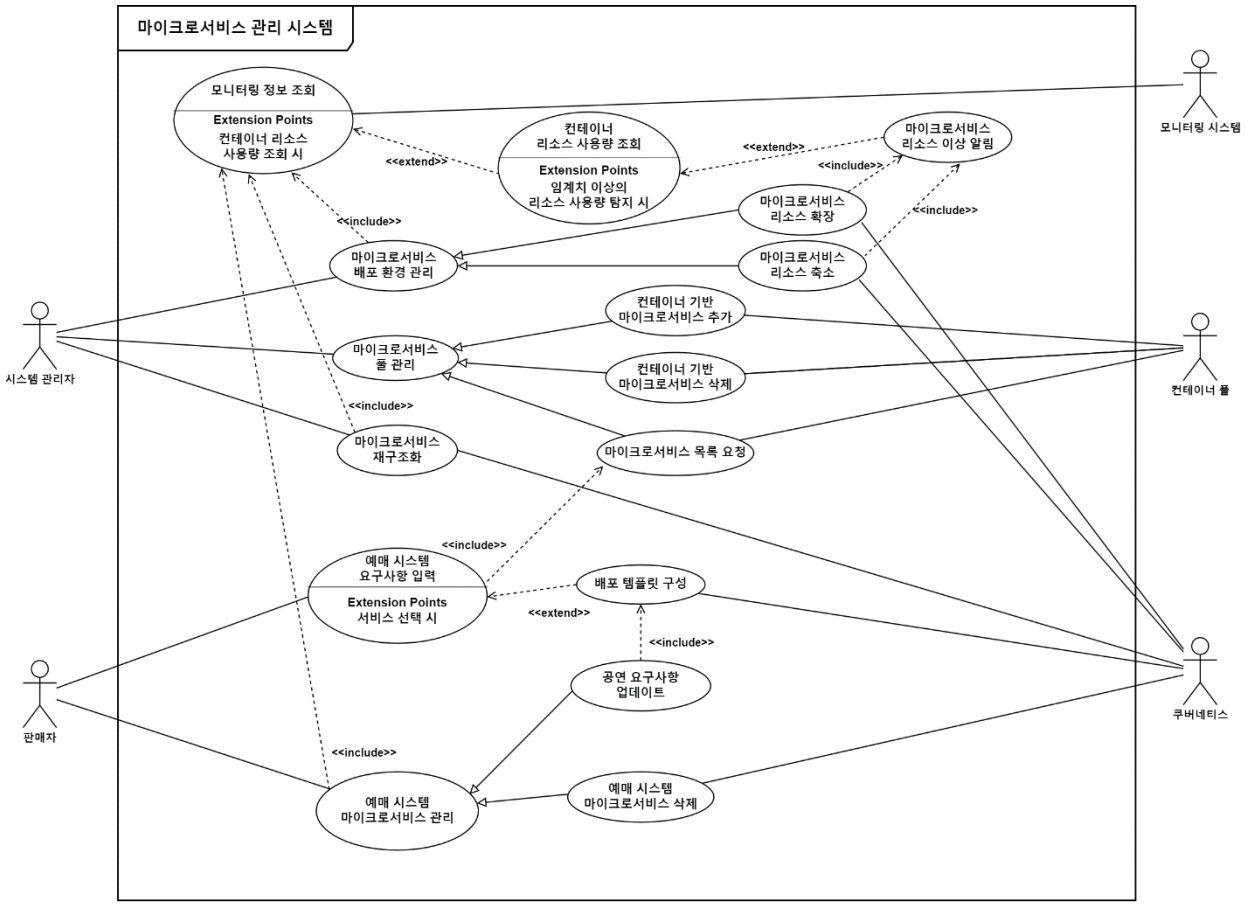


그림 4. 마이크로서비스 관리 시스템 유스케이스 다이어그램

그림 5 는 티켓팅 시스템의 유스케이스 다이어그램이다.

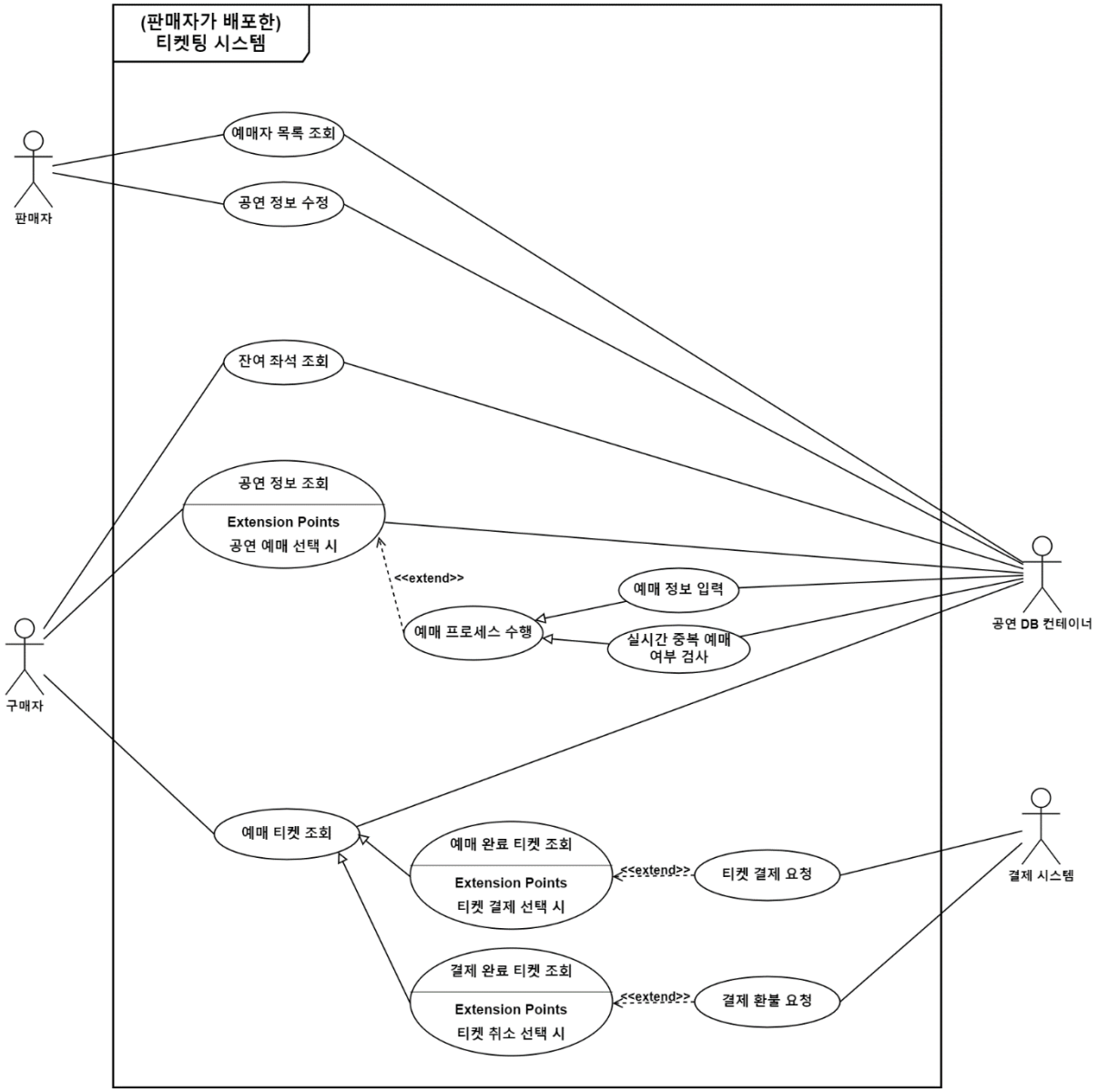


그림 5. 티켓팅 시스템 유스케이스 다이어그램

표 3 은 Actor 에 대한 설명을 나타낸 표이다.

표 3. Actor

요건	설명
시스템 관리자	전체 시스템 및 인프라를 유지 관리하는 관리자
판매자	판매자로 로그인하여 공연을 판매하는 티켓 예매 시스템 사용자
구매자	구매자로 로그인하여 시스템에 등록된 공연의 티켓을 구매하는 사용자
쿠버네티스	도커 기반의 컨테이너 관리 도구
컨테이너 풀	티켓 예매 시스템을 구성하는 컨테이너화 된 마이크로서비스 템플릿 저장소
모니터링 시스템	프로메테우스, 그라파나를 이용한 시스템과 서비스 모니터링 도구
공연 DB 컨테이너	판매자가 배포한 마이크로 서비스 중 하나로 공연에 관한 정보를 저장하는 DB
결제 시스템	티켓 구매 시 결제를 처리해주는 외부 시스템

② 유스케이스 명세

- 예매 시스템 요구사항 입력

유스케이스명	예매 시스템 요구사항 입력
개요	판매자는 공연에 관한 요구사항 입력을 통해 자동화된 티켓 예매 시스템을 배포할 수 있다.
관련 액터	판매자, 컨테이너 풀, 쿠버네티스
선행 조건	없음
이벤트 흐름	기본 흐름 1. 판매자는 예매 시스템 배포 버튼을 선택한다. 2. 판매자는 판매할 공연의 분야, 좌석 형태를 입력한다. 3. 마이크로서비스 관리 시스템은 컨테이너 풀에 등록되어 있는 마이크로서비스 목록을 보여준다. 대안 흐름 3.1 서비스 선택 시 1. 판매자는 목록에서 원하는 마이크로서비스를 선택한다. 2. 마이크로서비스 관리 시스템은 판매자가 선택한 마이크로서비스를 바탕으로 배포 템플릿을 구성한다. 3. 마이크로서비스 관리 시스템은 구성한 배포 템플릿을 포함하여 쿠버네티스에 예매 시스템 배포 요청을 보낸다. 4. 쿠버네티스는 판매자의 네임스페이스를 생성한다. 5. 쿠버네티스는 배포 템플릿을 이용하여 예매 시스템을 배포한 뒤 배포 성공 여부를 마이크로서비스 관리 시스템에 전송한다. 6. 마이크로서비스 관리 시스템은 전달받은 배포 성공 여부를 판매자에게 알린다.

- 공연 요구사항 업데이트

유스케이스명	공연 요구사항 업데이트
개요	판매자는 배포한 공연 서비스에 대한 요구사항을 업데이트할 수 있다.
관련 액터	판매자, 모니터링 시스템, 컨테이너 풀, 쿠버네티스
선행 조건	없음
이벤트 흐름	기본 흐름 1. 판매자는 예매 시스템 마이크로서비스 관리 메뉴를 선택한다. 2. 마이크로서비스 관리 시스템은 판매자가 배포한 예매 시스템 마이크로서비스를 모니터링 정보 조회를 통해 보여준다. 3. 판매자는 업데이트할 마이크로서비스를 선택한 뒤 업데이트 할 공연 요구사항 정보를 입력한다. 4. 마이크로서비스 관리 시스템은 업데이트 된 공연 요구사항을 바탕으로 배포 템플릿을 구성한다. 5. 마이크로서비스 관리 시스템은 구성한 배포 템플릿을 포함하여 쿠버네티스에 예매 시스템 배포 요청을 보낸다. 6. 쿠버네티스는 배포 템플릿을 이용하여 판매자의 네임스페이스에 예매 시스템을 배포한 뒤 배포 성공 여부를 마이크로서비스 관리 시스템에 전송한다. 7. 마이크로서비스 관리 시스템은 전달받은 배포 성공 여부를 판매자에게 알린다.

- 예매 시스템 마이크로서비스 삭제

유스케이스명	예매 시스템 마이크로서비스 삭제
개요	판매자는 배포한 예매 시스템 마이크로서비스를 삭제할 수 있다.
관련 액터	판매자, 모니터링 시스템, 쿠버네티스
선행 조건	없음
이벤트 흐름	기본 흐름 1. 판매자는 예매 시스템 마이크로서비스 관리 메뉴를 선택한다. 2. 마이크로서비스 관리 시스템은 판매자가 배포한 예매 시스템 마이크로서비스를 모니터링 정보 조회를 통해 보여준다. 3. 판매자는 삭제할 마이크로서비스를 선택한 뒤 예매 시스템 마이크로서비스 삭제 버튼을 선택한다. 4. 마이크로서비스 관리 시스템은 쿠버네티스에 판매자가 선택한 예매 시스템 마이크로서비스 삭제 요청을 보낸다. 5. 쿠버네티스는 요청받은 예매 시스템 마이크로서비스를 판매자의 네임스페이스에서 삭제한다. 6. 쿠버네티스는 판매자의 네임스페이스를 삭제한 뒤 예매 시스템 마이크로서비스 삭제 성공 여부를 마이크로서비스 관리 시스템에 전송한다. 7. 마이크로서비스 관리 시스템은 전달받은 삭제 성공 여부를 판매자에게 알린다.

- 모니터링 정보 조회

유스케이스명	모니터링 정보 조회
개요	배포중인 서비스에 대한 모니터링 정보를 조회할 수 있다.
관련 액터	시스템 관리자, 판매자, 모니터링 시스템
선행 조건	없음
이벤트 흐름	기본 흐름 1. 마이크로서비스 관리 시스템은 모니터링 시스템에 현재 배포중인 마이크로서비스에 대한 정보를 요청한다. 2. 모니터링 시스템은 현재 배포중인 마이크로서비스에 대한 정보를 마이크로서비스 관리 시스템에 전송한다. 3. 마이크로서비스 관리 시스템은 전달받은 마이크로서비스 정보를 보여준다. 대안 흐름 3.1 컨테이너 리소스 사용량 조회 시 1. 마이크로서비스 관리 시스템은 컨테이너 리소스 사용량 중 CPU 사용량, 메모리 사용량, 디스크 사용량, 트래픽 사용량을 대시보드를 통해 보여준다. 대안 흐름 3.1.1 임계치 이상의 리소스 사용량 탐지 시 1. 마이크로서비스 관리 시스템은 시스템 관리자에게 임계치 이상의 리소스 사용을 탐지하였다고 알린다.

- 컨테이너 기반 마이크로서비스 추가

유스케이스명	컨테이너 기반 마이크로서비스 추가
개요	시스템 컨테이너 풀에 컨테이너 기반의 새로운 마이크로서비스를 등록한다.
관련 액터	시스템 관리자, 컨테이너 풀
선행 조건	없음
이벤트 흐름	기본흐름 1. 시스템 관리자는 마이크로서비스 풀 관리 메뉴에서 컨테이너 기반 마이크로서비스 추가 버튼을 선택한다. 2. 시스템 관리자는 추가할 컨테이너 기반 마이크로서비스의 이름, 별칭, 버전을 입력한다. 3. 시스템 관리자는 추가할 마이크로서비스를 컨테이너 이미지 형태로 업로드한다. 4. 마이크로서비스 관리 시스템은 업로드 된 마이크로서비스 추가 요청을 컨테이너 풀로 보낸다. 5. 컨테이너 풀은 전달받은 마이크로서비스를 추가한 뒤 추가 성공 여부를 마이크로서비스 관리 시스템에 전송한다. 6. 마이크로서비스 관리 시스템은 전달받은 마이크로서비스 추가 성공 여부를 시스템 관리자에게 알린다. 예외흐름 2.1 기존의 마이크로 서비스와 중복된 정보를 입력할 경우 1. 기존의 마이크로서비스와 중복된 부분의 정보를 잘못된 입력이라고 표시한다. 2. 시스템 관리자는 표시된 부분을 수정한다.

- 마이크로서비스 목록 요청

유스케이스명	마이크로서비스 목록 요청
개요	컨테이너 풀에 등록되어 있는 컨테이너 기반 마이크로서비스 목록을 조회한다.
관련 액터	시스템 관리자, 컨테이너 풀
선행 조건	없음
이벤트 흐름	기본흐름 1. 시스템 관리자는 컨테이너 풀 관리 메뉴에서 마이크로서비스 목록 조회 버튼을 선택한다. 2. 마이크로서비스 관리 시스템은 컨테이너 풀에 현재 등록되어 있는 마이크로서비스 목록을 요청한다. 3. 컨테이너 풀은 현재 등록되어 있는 마이크로서비스 목록을 마이크로서비스 관리 시스템으로 전송한다. 4. 마이크로서비스 관리 시스템은 전달받은 마이크로서비스 목록을 보여준다. 5. 시스템 관리자는 원하는 마이크로서비스의 상세 조회 버튼을 누른다. 6. 선택된 마이크로서비스의 이름, 별칭, 버전 등의 상세 정보를 보여준다. 예외흐름 2.1 등록된 마이크로서비스가 없을 경우 1. 마이크로서비스 목록을 빈 상태로 보여준다.

- 컨테이너 기반 마이크로서비스 삭제

유스케이스명	컨테이너 기반 마이크로서비스 삭제
개요	시스템 컨테이너 풀에 등록되어 있는 컨테이너 기반 마이크로서비스를 삭제한다.
관련 액터	시스템 관리자, 컨테이너 풀
선행 조건	컨테이너 풀에 최소 1 개 이상의 컨테이너 기반 마이크로서비스가 있어야 한다.
이벤트 흐름	기본흐름 1. 마이크로서비스 관리 시스템은 현재 컨테이너 풀에 등록되어 있는 컨테이너 기반 마이크로서비스 목록을 보여준다. 2. 시스템 관리자는 마이크로서비스 목록에서 삭제할 마이크로서비스를 선택한다. 3. 마이크로서비스 관리 시스템은 컨테이너 풀에 시스템 관리자가 선택한 마이크로서비스에 대한 삭제 요청을 보낸다. 4. 컨테이너 풀은 요청받은 마이크로서비스의 컨테이너 이미지를 삭제한 뒤 삭제 성공 여부를 마이크로서비스 관리 시스템에 전송한다. 5. 마이크로서비스 관리 시스템은 전달받은 마이크로서비스 삭제 성공 여부를 시스템 관리자에게 알린다.
기타 요구사항	현재 배포중인 마이크로서비스의 컨테이너의 이미지를 컨테이너 풀에서 삭제할 경우, 서비스 동작은 계속하되 삭제 및 중단 시 재생성이 불가함을 알린다.

- 마이크로서비스 리소스 확장

유스케이스명	마이크로서비스 리소스 확장
개요	시스템 관리자는 마이크로서비스의 리소스를 확장할 수 있다.
관련 액터	시스템 관리자, 쿠버네티스
선행 조건	없음
이벤트 흐름	기본 흐름 1. 시스템 관리자는 모니터링 정보 조회에서 컨테이너 리소스 사용량을 조회한다. 2. 마이크로서비스 관리 시스템은 임계치 이상의 리소스 사용을 탐지하고 시스템 관리자에게 알린다. 3. 시스템 관리자는 확장할 마이크로서비스, 확장할 크기를 입력하고 마이크로서비스 리소스 확장 버튼을 선택한다. 4. 마이크로서비스 관리 시스템은 시스템 관리자가 입력한 정보와 함께 쿠버네티스에 마이크로서비스 확장 요청을 보낸다. 5. 쿠버네티스는 요청받은 마이크로서비스의 리소스 확장 작업을 처리한 뒤 확장 성공 여부를 마이크로서비스 관리 시스템에 전송한다. 6. 마이크로서비스 관리 시스템은 전달받은 확장 성공 여부를 시스템 관리자에게 알린다.

- 마이크로서비스 리소스 축소

유스케이스명	마이크로서비스 리소스 축소
개요	시스템 관리자는 마이크로서비스의 리소스를 확장할 수 있다.
관련 액터	시스템 관리자, 쿠버네티스
선행 조건	없음
이벤트 흐름	기본 흐름 1. 시스템 관리자는 모니터링 정보 조회에서 컨테이너 리소스 사용량을 조회한다. 2. 마이크로서비스 관리 시스템은 임계치 이상의 리소스 사용을 탐지하고 시스템 관리자에게 알린다. 3. 시스템 관리자는 축소할 마이크로서비스, 축소할 크기를 입력하고 마이크로서비스 리소스 축소 버튼을 선택한다. 4. 마이크로서비스 관리 시스템은 시스템 관리자가 입력한 정보와 함께 쿠버네티스에 마이크로서비스 축소 요청을 보낸다. 5. 쿠버네티스는 요청받은 마이크로서비스의 리소스 축소 작업을 처리한 뒤 확장 성공 여부를 마이크로서비스 관리 시스템에 전송한다. 6. 마이크로서비스 관리 시스템은 전달받은 축소 성공 여부를 시스템 관리자에게 알린다.

- 마이크로서비스 재구조화

유스케이스명	마이크로서비스 재구조화
개요	시스템 관리자는 마이크로서비스의 서비스간 재결합 또는 재배치를 통해 안정적인 서비스를 제공할 수 있다.
관련 액터	시스템 관리자, 쿠버네티스, 모니터링 시스템
선행 조건	없음
이벤트 흐름	기본 흐름 1. 시스템 관리자는 마이크로서비스 재구조화 버튼을 선택한다. 2. 마이크로서비스 관리 시스템은 모니터링 정보 조회를 통해 현재 배포중인 마이크로서비스에 대한 정보를 보여준다. 3. 마이크로서비스 관리 시스템은 컨테이너 리소스 사용량 중 CPU 사용량, 메모리 사용량, 디스크 사용량, 트래픽 사용량을 대시보드를 통해 보여준다. 4. 시스템 관리자는 현재 배포중인 마이크로서비스 정보와 컨테이너 리소스 사용량을 참고하여 작업 노드를 변경할 컨테이너를 선택한다. 5. 시스템 관리자는 선택한 컨테이너를 재배치할 노드를 선택한 뒤 작업 노드 변경 버튼을 선택한다. 6. 마이크로서비스 관리 시스템은 작업 노드를 변경할 컨테이너 정보, 해당 컨테이너가 현재 동작하고 있는 노드, 해당 컨테이너를 재배치할 노드 정보를 담아 쿠버네티스에 컨테이너 작업 노드 변경을 요청한다. 7. 쿠버네티스는 전달받은 정보를 통해 컨테이너를 재배치하고 성공 여부를 마이크로서비스 관리 시스템에 전송한다. 8. 마이크로서비스 관리 시스템은 전달받은 컨테이너 재배치 성공 여부를 시스템 관리자에게 알린다.

- 예매자 목록 조회

유스케이스명	예매자 목록 조회
개요	판매자는 티켓을 예매한 구매자들의 정보를 확인할 수 있어야 한다.
관련 액터	판매자, 공연 DB 컨테이너
선행 조건	1. 판매자는 로그인 상태여야 한다. 2. 판매자는 최소 1 개 이상의 공연을 판매하고 있어야 한다.
이벤트 흐름	기본 흐름 1. 메인 화면에서 예매자 목록 조회 버튼을 선택한다. 2. 판매중인 공연 중에서 예매자 목록을 확인할 공연을 선택한다. 3. 공연 DB 컨테이너에서 해당 공연에 대한 예매자 목록을 요청한다. 4. 공여 DB 컨테이너에서 받은 정보를 웹 UI 에서 표시한다.

- 공연 정보 수정

유스케이스명	공연 정보 수정
개요	판매자는 자신이 배포한 티켓팅 서비스의 정보를 수정할 수 있어야 한다.
관련 액터	판매자, 공연 DB 컨테이너
선행 조건	1. 판매자는 로그인 상태여야 한다. 2. 판매자는 최소 1 개 이상의 공연을 판매하고 있어야 한다.
이벤트 흐름	기본 흐름 1. 판매자는 공연 정보 수정 버튼을 선택한다. 2. 공연 정보 조회 기능을 통해 판매중인 공연을 확인한다. 3. 수정할 공연을 선택한다. 4. 선택한 공연에 대해 정보를 수정한다. 5. 수정한 정보를 공연 DB 컨테이너에 저장한다.

- 잔여 좌석 조회

유스케이스명	잔여 좌석 조회
개요	구매자는 잔여 좌석 정보를 조회할 수 있어야 한다.
관련 액터	구매자, 공연 DB 컨테이너
선행 조건	구매자는 로그인 상태여야 한다.
이벤트 흐름	기본 흐름 1. 구매자는 메인 화면에서 잔여 좌석 조회 버튼을 선택한다. 2. 공연 DB 컨테이너에 잔여 좌석에 대한 정보를 요청한다. 3. 공연 DB 컨테이너에서 받은 정보를 웹 UI 에서 표시한다.

- 공연 목록 조회

유스케이스명	공연 목록 조회
개요	구매자는 전체 공연의 목록을 조회할 수 있어야 한다.
관련 액터	구매자, 공연 DB 컨테이너
선행 조건	구매자는 로그인 상태여야 한다.
이벤트 흐름	기본 흐름 1. 구매자는 메인 화면에서 공연 정보 조회 버튼을 선택한다. 2. 공연 DB 컨테이너에 판매중인 공연에 대한 정보를 요청한다. 3. 구매자는 웹 UI 상에서 판매중인 공연에 대한 정보를 확인할 수 있다. 대안흐름 3.1 공연 예매선택 시 1. 예매 프로세스를 수행한다. 2. 시스템은 구매자에게 예매 정보 입력을 요청한다. 3. 실시간 중복 예매 여부 검사 기능으로 하나의 좌석에 하나의 유저만 접근 가능하게 한다.

- 예매 티켓 조회

유스케이스명	예매 티켓 조회
개요	구매자는 자신이 예매한 티켓을 조회할 수 있다.
관련 액터	판매자, 공연 DB 컨테이너
선행 조건	구매자는 로그인 상태여야 한다.
이벤트 흐름	기본 흐름 1. 메인 화면에서 예매 티켓 조회 버튼을 선택한다. 2. 공연 DB 컨테이너에 티켓에 관한 정보를 요청한다. 예외 흐름 2.1 티켓 결제 버튼 선택 시 1. 결제 시스템에 티켓 결제 프로세스를 요청한다. 2.2 티켓 취소 버튼 선택 시 1. 결제시스템에 결제 환불 요청 프로세스를 요청한다.

3. 현실적 제약 사항 분석 결과 및 대안

1) 현실적 제약 사항

① 테스트 시, 티켓을 예매하는 과정을 수행하는 대규모의 사람을 모으기 힘들다.

② 티켓 예매 시 실제로 결제를 진행하는 데 어려움이 있다.

③ 모든 공연 장소에 대한 좌석 정보를 반영하는 데 어려움이 있다.

2) 대안

① 실제 사람이 아닌 예매 봇 또는 매크로 작업을 수행하는 별도의 프로세스를 구성해 테스트를 진행할 수 있다.

② 테스트 결제 시스템을 사용하거나, 결제 없이 예매 절차를 진행할 수 있다.

③ 모든 공연 장소가 아닌 일부 장소의 좌석 정보만을 선택하여 시스템에 반영할 수 있다.

4. 시스템 구성

1) 시스템 구성도

그림 6 은 본 시스템의 전체 구성을 나타낸 그림이다.

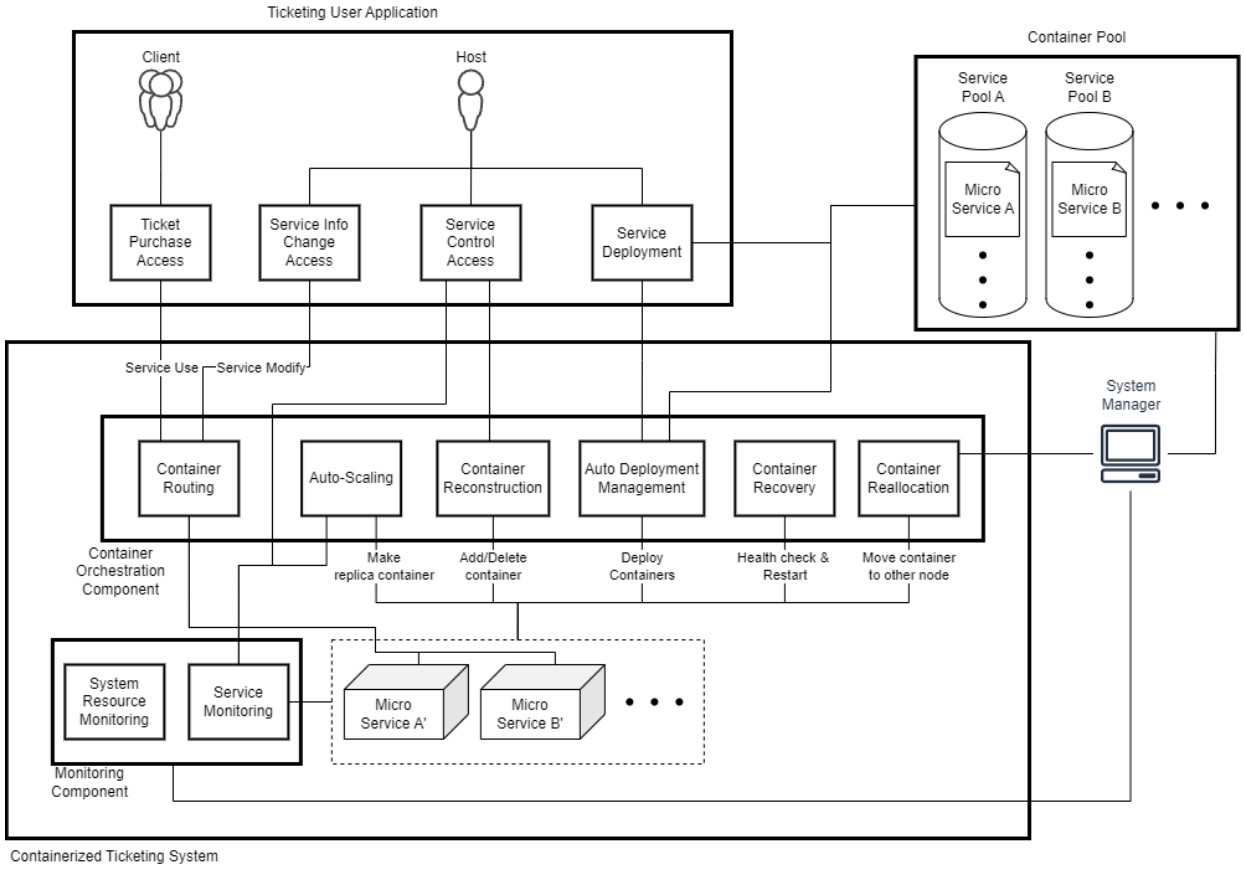


그림 6. 시스템 구성도

사용자는 판매자와 구매자로 나뉘고 구매자는 티켓 구매 작업을, 판매자는 판매를 위한 서비스 배포, 제어, 수정 작업을 수행하게 된다. 시스템 관리자는 컨테이너 풀과 시스템을 종합적으로 관리한다. 컨테이너 풀은 컨테이너화 된 마이크로서비스의 이미지를 관리하는 저장소이다. 시스템 관리자는 컨테이너 풀에 마이크로서비스의 이미지의 저장 및 관리를 수행한다.

클라우드 환경에서 컨테이너 기반의 마이크로서비스 배포 및 관리를 통해 시스템이 운영된다. 컨테이너들을 관리하기 위한 컨테이너 오케스트레이션의 기능을 정의한다. 컨테이너 오케스트레이션에서는 1) 자동 배포 관리 2)컨테이너 라우팅 3) 컨테이너 재구성 4) 오토-스케일링 5) 컨테이너 복구 6) 컨테이너 재배치를 수행한다.

1) 자동 배포 관리에서는 판매자가 입력한 정보를 바탕으로 컨테이너 풀에서 이미지를 가져오고 배포를 위한 명세로의 변환 작업을 수행해 서비스 컨테이너들의 배포가 이루어진다. 2) 컨테이너 라우팅은 외부에서 컨테이너에 대한 접근을 시도할 때 적절한 컨테이너로의 라우팅을 수행한다. 구매자의 티켓 구매 작업과 판매자의 서비스 수정 작업을 위해서는 내부 컨테이너로의 접근이 필요한데 이를 컨테이너 라우팅이 도와준다. 3) 컨테이너 재구성에서는 판매자가 자신이 배포한 서비스의 모니터링 정보를 확인하고 서비스 추가 및 삭제를 원할 때 이를 반영하여 명세를 재변환하고 배포가 이루어진다. 4) 오토-스케일링은 각 서비스에 대한 모니터링 정보를 주기적으로 확인하며 정의된 규칙에 따라 서비스들을 스케일-아웃하는 작업을 수행한다. 5) 컨테이너

복구는 각 서비스들에 대한 주기적인 헬스 체크를 통해 서비스가 잘 동작하는지 체크하며 서비스가 중단되었다고 판단하면 컨테이너를 재실행하여 복구하는 작업을 수행한다. 6) 컨테이너 재배치는 컨테이너의 워커 노드를 변경 및 이동하는 작업을 수행한다. 시스템 관리자가 시스템과 서비스들의 리소스 사용량을 모니터링하며 시스템 환경 상태를 확인하고 모니터링 정보를 통해 노드들의 리소스 사용량을 분석해 일부 노드의 특정 리소스 사용량이 집중되어 있다고 판단되면, 시스템 관리자는 현재 운영중인 마이크로서비스들을 컨테이너 재배치를 통해 안정적이고 효율적인 클라우드 환경을 유지할 수 있게 한다.

2) 개발 환경

표 4 는 시스템의 개발에서 사용되는 관련 기술들을 보여준다.

표 4. 개발 환경

개발환경	사용기술
소프트웨어 형상관리(SCM)	Github
개발 IDE	VSCode
DBMS	MySQL
Front-end	React, HTML, CSS, JavaScript
Back-end	Spring Boot
API	REST API
프로젝트 문서화 관리	Notion
메트릭 분석	Prometheus, Grafana
CI / CD	GitHub Actions

3) 사용 기술

① Docker

그림 7 은 도커의 아키텍처를 나타낸 그림이다.

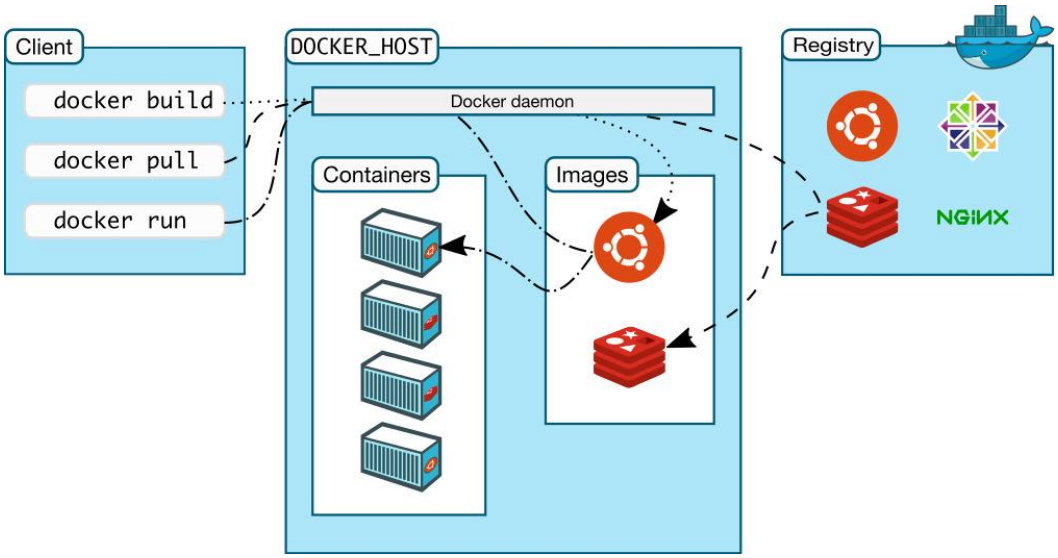


그림 7. 도커의 아키텍처

도커(Docker)는 리눅스 컨테이너에 리눅스 어플리케이션을 프로세스 격리기술을 사용하여 더 쉽게 컨테이너로 실행하고 관리할 수 있게 해주는 오픈소스 프로젝트이다. 도커는 일반적으로 도커 엔진(Docker Engine) 혹은 도커에 관련된 모든 프로젝트를 말한다.

도커의 대표적인 컴포넌트로는 Docker daemon, Docker Client, Docker Registry, Docker object, Docker image, Container 등이 있고 각 컴포넌트의 기능은 다음과 같다.

I. Docker daemon

도커 API 요청을 수신하고 이미지, 컨테이너, 네트워크 및 볼륨과 같은 도커 개체를 관리한다.

II. Docker Client

사용자가 도커와 상호 작용하기 위한 미들웨어로 이미지 생성, 컨테이너 생성 및 실행과 같은 명령어들을 위생 할 수 있으며, 해당 명령어들은 도커 데몬과의 통신을 통하여 수행되어진다.

III. Docker Desktop

도커 데스크톱 지원용 프로그램이다. 도커 데몬, 도커 클라이언트와 쿠버네티스(minikube) 와 같은 다양한 응용프로그램들이 함께 탑재되어 있다.

IV. Docker Registry

도커 이미지를 저장하는 레지스트리로 도커 데스크탑 설치 시 개인 레지스트리가 함께 설치되며, 공용 레지스트리인 Docker hub 에서 이미지 검색을 통하여 이미지를 다운 받을 수 있다.

V. Docker object

도커를 사용하는데 있어서 필요한 이미지, 컨테이너, 네트워크, 볼륨, 플러그인들이 여기에 속한다.

VI. Docker image

도커 컨테이너를 만드는데 사용되어지는 템플릿이다. 다른 이미지에 사용자 설정을 추가하여 사용할 수 있는 구성 세부 정보 설치 기능을 수행할 수 있다.

VII. Container

이미지의 실행 가능한 인스턴스로 Docker API 혹은 CLI 를 통하여 생성, 중지, 시작, 이동, 삭제가 가능하다.

② Kubernetes

그림 8 은 쿠버네티스의 아키텍처를 나타낸 그림이다.

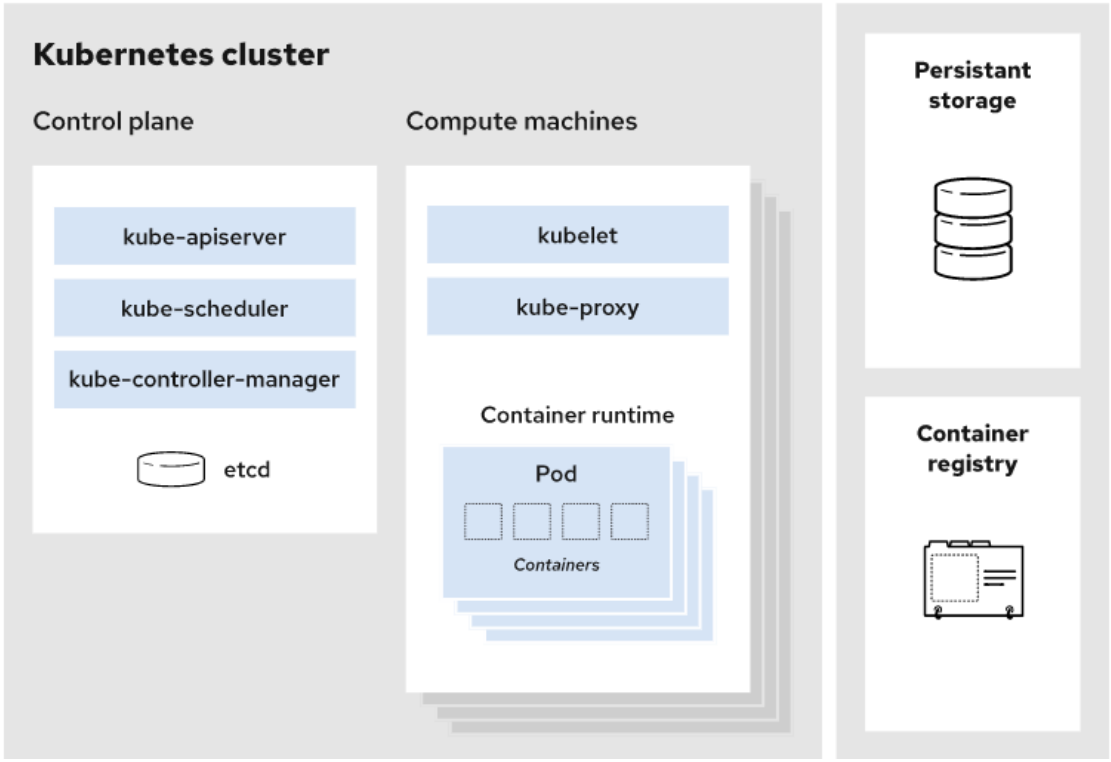


그림 8. 쿠버네티스 아키텍처

쿠버네티스는 컨테이너화 된 워크로드와 서비스를 관리하기 위한 이식성이 있고, 확장 가능한 오픈소스 플랫폼이다. 쿠버네티스는 선언적 구성과 자동화를 모두 용이하게 해준다. 쿠버네티스는 크고, 빠르게 성장하는 생태계를 가지고 있다. 쿠버네티스 서비스, 기술 지원 및 도구는 어디서나 쉽게 이용할 수 있다.

다음은 쿠버네티스의 주요 구성요소에 대한 설명이다.

I. Control plane

쿠버네티스 클러스터 전체를 컨트롤 하는 시스템으로 API Server 를 통해 쿠버네티스를 관리하고, 모든 컴포넌트들은 API Server 를 통해서 통신한다.

II. kube-apiserver

쿠버네티스 내부의 모든 컴포넌트들이 서로 호출하기 위한 컴포넌트로 쿠버네티스의 모든 기능들은 REST

API 로 제공하고 그에 대한 명령을 처리하는 컴포넌트이다.

III. kube-controller-manager

쿠버네티스의 ReplicaSet, Deployment 등 Contoller 를 관리 및 적절한 노드에 할당하는 컴포넌트로 각 컨트롤러에게 pod 의 복제/배포 명령 수행한다.

IV. etcd

쿠버네티스 클러스터의 데이터베이스 역할을 하는 컴포넌트로 클러스터의 모든 설정, 상태 데이터를 저장하는 컴포넌트이다. object 를 key-value 형태로 저장한다.

V. Compute machines

마스터에 의해 명령을 받고 실제 워크 로드를 생성해서 서비스하는 컴포넌트로 실제 컨테이너들이 생성되는 가상머신 또는 물리적인 서버이다.

VI. kubelet

노드에 배포되는 agent 로 마스터의 API 서버와 통신한다. 마스터의 API Server 로부터 수행할 명령을 받아서 Worker Node 를 수행시킨다. Worker Node 의 상태를 마스터로 전달한다.

③ Prometheus, Grafana

그림 9 는 프로메테우스와 그라파나의 동작 방식을 나타낸 그림이다.

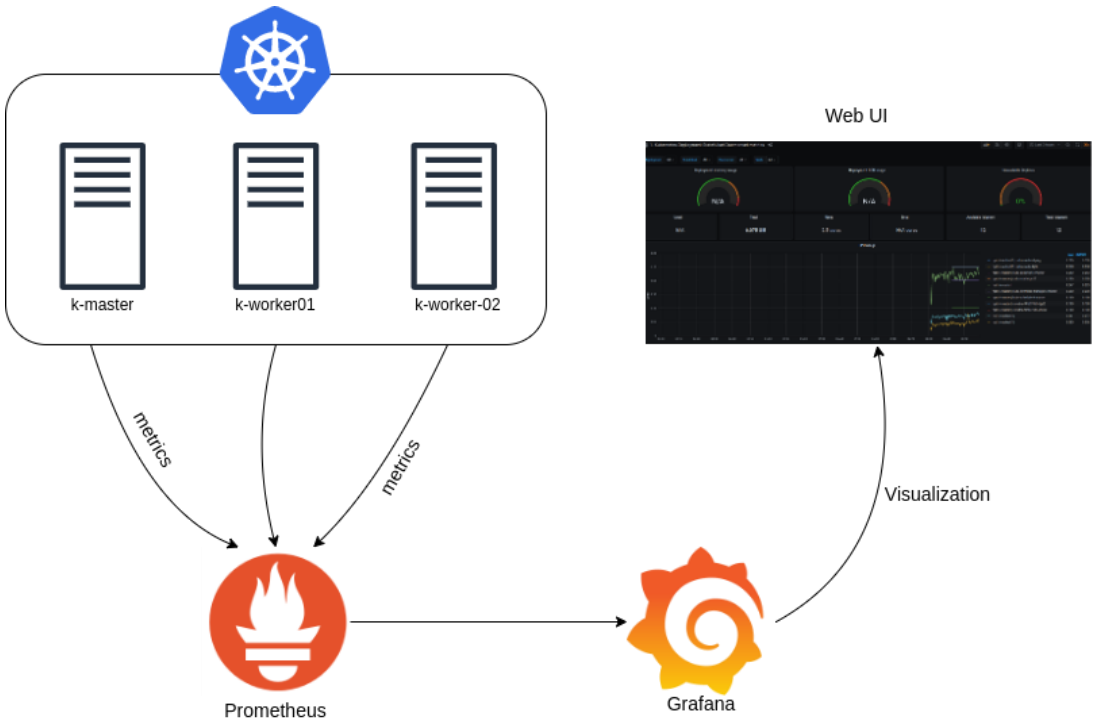


그림 9. 프로메테우스와 그라파나의 동작 방식

프로메테우스는 오픈소스 모니터링 툴로 지표수집을 통한 모니터링이 주요 기능이다. 지표를 수집하여 모니터링 할 수 있고, 기본적으로 Pull 방식으로 데이터를 수집하는데, 즉 프로메테우스가 주기적으로 모니터링

대상에서 지표를 읽어오는 것이다. 그라파나는 오픈소스 메트릭 데이터 시각화 도구로 메트릭 분석 플랫폼이다. 외부 데이터 소스를 정의하고 해당 데이터 소스에 쿼리를 통해서 데이터를 동적으로 가지고 와서 시각화 한다.

다음은 프로메테우스의 특징에 대한 설명이다.

I. 간편한 통합과 운영

Kubernetes 와 같은 컨테이너 오케스트레이션 시스템과의 통합이 용이하며, 설치 및 운영이 간편하여 IT 인프라의 모니터링과 관리를 단순화한다. 기본 구성 요소 설치뿐만 아니라 모니터링 대시보드, 알람 룰까지 모두 단일 헬름 차트를 이용하여 설치할 수 있다.

II. 서비스 디스커버리

동적으로 확장되고 축소되는 쿠버네티스 환경에서 프로메테우스는 개별 모니터링 대상을 서비스 엔드 포인트에 등록하여 자동으로 변경 내역을 감지한다. 서비스 리소스를 생성하면 자동으로 엔드 포인트 개체가 등록되는데 해당 엔드 포인트를 기준으로 프로메테우스도 모니터링 한다.

III. 다양한 애플리케이션 익스포터(exporter) 제공

MySQL, Elastic, Kafka, Redis 등 거의 모든 애플리케이션이 프로메테우스에서 사용할 수 있는 메트릭 정보를 제공한다. 사용자는 애플리케이션 설치 시 별도의 추가 시간을 들이지 않고 익스포터를 이용하여 손쉽게 모니터링 할 수 있다.

IV. 자체 검색 언어 PromQL(Prometheus Query Language) 제공

다양한 레이블 사용이 가능한 메트릭을 조회할 수 있도록 프로메테우스는 자체 검색 언어를 제공한다. 시각화 솔루션 그라파나에서 PromQL 으로 다양하게 자료를 조회하여 원하는 그래프 형태로 나타낼 수 있다. 뿐만 아니라 로깅 솔루션 로키에서도 비슷한 검색 언어(LogQL)를 사용하여 편의성이 뛰어나다는 장점이 있다.

V. 사용자 지정 및 유연성

사용자의 특정 요구에 맞게 경고 규칙, 데이터 수집 방법, 저장 기간 등을 맞춤 설정할 수 있어, 다양한 환경과 요구 사항에 유연하게 대응할 수 있다.

VI. Pull 방식

변경이 잦은 개별 모니터링 대상을 프로메테우스는 에이전트를 설치하고 에이전트가 중앙 서버로 모니터링 정보를 전달하는(Push) 방식이 아닌 중앙의 프로메테우스 서버가 모니터링 대상의 정보를 직접 가져오는(Pull) 방식을 사용한다.

VII. Container

이미지의 실행 가능한 인스턴스로 Docker API 혹은 CLI 를 통하여 생성, 중지, 시작, 이동, 삭제가 가능하다.

다음은 그라파나의 특징에 대한 설명이다.

I. 패널

히스토그램, 그래프, 지리 지도, 열 지도 등을 사용하여 사용자가 원하는 대로 데이터를 시각화 한다.

II. 플러그인

기존 데이터 소스와 연결된 패널 플러그인을 통해 데이터 마이그레이션 없이 사용자 친화적인 API 에서 실시간으로 데이터를 렌더링한다. 또한 데이터 소스 플러그인을 생성하여 모든 사용자 정의 API 에서 메트릭을 검색할 수 있다.

III. 경고

단일 사용자 인터페이스에서 모든 경고를 생성, 통합, 제어할 수 있다.

IV. 변환

데이터 소스 및 쿼리 전반에서 이름 변경, 요약, 결합, 계산을 수행할 수 있다.

V. 주석

다양한 데이터 소스의 풍성한 이벤트를 사용하여 그래프에 주석을 추가할 수 있다.

VI. 패널 편집기

패널을 구성 및 커스터마이징 하기 위한 일관된 사용자 인터페이스이다.

5. 개발 일정 및 역할 분담

1) 개발 일정

표 5 는 시스템 개발 일정을 나타낸 표이다.

표 5. 개발 일정

기간 / 수행내용	5 월		6 월				7 월					8 월				9 월				
	3	4	1	2	3	4	1	2	3	4	5	1	2	3	4	1	2	3	4	5
필요 지식 습득	All																			
마이크로서비스 구조 패키징 및 배포 기능 구축					이동현															
컨테이너 인프라 관리 기능					조용진															
마이크로서비스 배포 환경 관리 기능 구현					강찬석															
컨테이너 기반 청사진 구축												강찬석, 이동현								
요구사항 기반 마이크로서비스 구현												강찬석, 조용진								
테스트 및 보완														All						
최종보고서 작성																	All			

2) 역할 분담

표 6 은 개인별 시스템 역할 분담을 나타낸 표이다.

표 6. 역할 분담

이름	역할
공통	필요 지식 습득, 테스트 및 보완, 중간보고서, 최종보고서 작성
강찬석	마이크로서비스 배포 환경 관리 기능 구현, 컨테이너 기반 청사진 구축, 요구사항 기반 마이크로서비스 구현
조용진	컨테이너 인프라 관리 기능, 요구사항 기반 마이크로서비스 구현
이동현	마이크로서비스 구조 패키징 및 배포 기능 구축, 컨테이너 기반 청사진 구축