클라우드컴퓨팅

텀 프로젝트 1: 제안서

대표학생 이름	오지웅
대표학생 학번	201924511
대표학생 소속 학과/대학	정보컴퓨터공학부 / 정보의생명공학대학
분반	059

<주의사항>

- 팀 과제 입니다. (팀 대표 또는 팀원 중 아무나 한 명만 제출하면 됩니다.)
- 각각의 질문 아래에 답을 작성 후 제출해 주세요.
    - 소스코드/스크립트 등을 제출하는 경우, 해당 파일의 이름도 적어주세요.
- PLATO제출 데드라인: 공지사항 참고
    - 데드라인을 지나서 제출하면 0점
    - 주말/휴일/학교행사 등 모든 날짜 카운트 함
    - 부정행위 적발 시, 원본(보여준 사람)과 복사본(베낀 사람) 모두 0점 처리함
    - 예외 없음
- PLATO에 아래의 파일을 제출 해 주세요
    - 보고서(PDF 파일로 변환 후 제출, 또는 워드문서 그대로 제출)
        - 보고서의 경우 PDF/DOC/DOCX 이외의 확장자는 제출 불가
        - 다른 확장자로 제출하였는데 채점자 컴퓨터에서 파일이 열리지 않으면 0점 처리
    - 보고서 파일명 및 보고서 첫 페이지에 이름과 학번을 입력 해 주세요.

<개요>

이번 과제는 텀 프로젝트 제안서를 제출하는 내용입니다.

<텀 프로젝트 : 제안서>

[1] 개발 내용 소개

텀 프로젝트 주제 및 텀 프로젝트에서 개발하고자 하는 시스템/서비스/SW에 대해 소개하세요.

답변 1(프로젝트 주제):

K3s 기반 다중 노드 Kubernetes 환경에서 안정적인 운영이 가능한 수강신청 서비스 개발

답변 2(프로젝트 소개):

본 프로젝트는 대학생의 입장에서 가장 친숙하면서 사용량의 급격한 변화가 잦은 **수강신청 서비스**에서, 사용자가 급격하게 몰리더라도 안정적인 운영이 가능하도록 자동 확장 및 부하 분산 기능을 갖춘 **수강신청 웹 서비스를 K3s 기반의 다중 노드 Kubernetes 환경에서 설계 및 구현**하는 것을 목표로 한다.

FastAPI 기반의 수강신청 서버를 다중 노드에 배포하고 **CPU 사용률 기반의 HPA(Horizontal Pod Autoscaler)**를 적용하여 자동 확장 및 부하 분산 기능을 구현하며, 그 과정을 Prometheus 및 Grafana를 통해 **실시간으로 시각화**하는 시스템을 구축한다. 로컬 네트워크 상의 3대의 노드(1 마스터 + 2 워커)로 구성된 K3s 클러스터를 구성하고, 수강신청 API 서버를 컨테이너화하여 클러스터에 배포한다. 이 서버는 실제 수강신청 서비스를 모델로 하며, 사용자 요청에 따라 수강 인원을 증가시키는 기능을 가진다. 추가로 원활한 시연을 위해 부하 유도용 엔드포인트도 함께 제공하여 트래픽의 급격한 증가 상황을 시뮬레이션할 수 있다.

HPA 설정을 통해 부하에 따라 Pod 수가 자동으로 증가하고, 이 Pod들이 워커 노드에 자동으로 분산 배포되는 오토스케일링 과정을 편하게 확인할 수 있도록 관리자 홈페이지를 개발하여 시각화한다. Prometheus를 활용해 클러스터 리소스 및 HPA 상태를 수집하고, Grafana 대시보드를 구성하여 Pod 수 변화, CPU 사용률, 노드별 부하 분산 현황을 시각적으로 표현함으로써 **Kubernetes 환경에서의 자동 확장과 안정성 확보 메커니즘을 시연 가능한 형태로 구현**한다.

[2] 필요성 및 활용방안.

개발하고자 하는 SW의 필요성에 대해 설명하세요 개발 결과물이 어떻게, 그리고 어떤 상황에서 사용될지(Use-Case) 설명하세요.

답변:

필요성

접속 폭주로 인한 서버 다운 사례

매 학기 수강신청 시작 직후 수천 명이 동시 접속하면서 시스템 응답 지연 또는 일시적인 접속 불가 현상이 반복됨. 실제로 커뮤니티(에브리타임 등)에는 수강신청 장애에 대한 불만이 자주 제기되며, 이는 학생들의 학사 계획에 직접적인 영향을 미침.

서버 다운 이유: 자체 IDC 기반 한계

수강신청 서버 ip 주소 확인 - $ nslookup sugang.pusan.ac.kr

```
(base) onjih8587@ohjiwoongui-MacBookPro k8s_test % nslookup sugang.pusan.ac.kr
Server:         168.126.63.1
Address:        168.126.63.1#53

Non-authoritative answer:
Name:   sugang.pusan.ac.kr
Address: 164.125.6.5
```

공공 ip (클라우드 서비스) 기반이 아님을 확인.

위 IP는 **AWS, GCP, Azure와 같은 공공 클라우드 주소가 아니며**, 부산대학교 소유의 자체 IDC에서 운영 중인 서버로 추정됨.

부산대학교 정보전산원은 자체 IDC를 기반으로 다음과 같은 인프라를 제공하고 있음:

1. 네트워크 연결 및 관제

2. 항온항습 유지

3. 방화벽 및 VPN 기반 보안

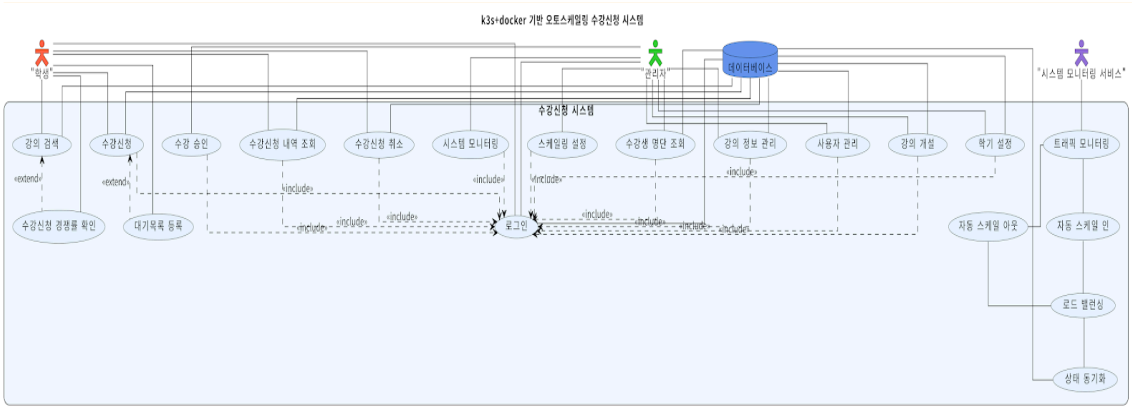
4. 자료 백업 및 복구

정보화본부에서는 학내 정보자원의 효율적 활용과 정보통신보안 강화를 위하여 대학통합전산센터(Integrated Data Center)를 구축하여 운영 중에 있음.

그러나 이러한 **정적 물리 인프라는 자동 확장(Auto Scaling)** 기능을 기본적으로 지원하지 않으며, 트래픽 변화에 민감하게 대응하기 어렵고 리소스 낭비가 발생할 수 있음.

민감한 데이터 가령 **학생 개인 정보(학번, 이름, 성적 등)는 내부 시스템에 유지하고**, 클라우드에서는 **수강신청 처리 로직과 UI만 분리 운영**하는 구조가 필요함.

사용 상황



공통 API

1. 서버 시간 조회
   - 엔드포인트: GET /api/getServerTime
   - 사용 목적: 모든 사용자가 서버의 현재 시간을 확인할 때 사용
   - response: {date, time, error}
   - YYYY-MM-DD, HH:mm:ss
2. 로그인
   - 엔드포인트: POST /api/login
   - 사용 목적: 모든 유형의 사용자가 시스템 접속 시 인증
3. request: JSON { id, password }
   - response: JSON { success, user, role, message }

학생 관점 API

1. 수강 요약 정보 조회
    - 엔드포인트: POST /api/summary
    - 사용 목적: 학생이 자신의 수강 신청 현황 요약 확인
    - request: JSON { id }
    - response: JSON { appliedCoursesCount, appliedCredits, availableCredits, remainingCredits, status }
    - ex) {6건, 18학점, 21학점, 3학점}
2. 등록된 과목 조회
    - 엔드포인트: POST /api/registeredCourses
    - 사용 목적: 학생이 자신이 수강 신청한 과목 목록 확인
    - request: JSON { id }
    - response: JSON { courseName, courseId, classSection, courseCategory, credit, limit, retake, scheduleSummary, changeSection, note }
3. 과목 검색
    - 엔드포인트: POST /api/course/search
    - 사용 목적: 학생이 수강 가능한 과목 검색
    - request: JSON { courseLevel, universityLevel, searchType, subjectCategory. subjectSubcategory, courseName, coreCompetency, isEnglishLecture }
    - response: JSON { courseName, courseId, classSection, courseCategory, credit, limit, waitingCount, professor, department, scheduleSummary, note}
4. 수강 신청
    - 엔드포인트: POST /api/register-course
    - 사용 목적: 학생이 특정 과목에 수강 신청
    - request: JSON { courseId, classSection, studentId }
    - response: JSON { status, message}

강의 관리자 관점 API

1. 강의 개설
    - 엔드포인트: POST /api/lecture/create
    - 사용 목적: 강의 관리자가 새로운 강의 등록
    - request: JSON { courseLevel, universityLevel, subjectCategory, subjectSubcategory, courseName, coreCompetency, isEnglishLecture, courseId, classSection, credit, limit, professor, department, scheduleSummary, note }
    - response: JSON { status, message}
2. 과목 검색
    - 엔드포인트: POST /api/course/search
    - 사용 목적: 강의 관리자가 개설된 강의 정보 확인

시스템 관리자 관점 API

1. 모니터링 관련(grafana에서 사용)
    - GET /api/monitoring/cpu-usage

- GET /api/monitoring/memory-usage
- GET /api/monitoring/pod-status
- GET /api/monitoring/node-status
- GET /api/monitoring/network-traffic

2. 시스템 설정 및 제어 api(ui에서 k3s hpa 관련 api 호출)
   - 엔드포인트: PATCH /api/hpa/patch/{deployment name}
   - 사용 목적: 시스템 관리자가 hpa 리소스 변경 등 변경
3. 부하 테스트용 api
   - 엔드포인트: POST /api/hpa/loadtest
   - 사용 목적: 시스템 관리자가 hpa 기반 부하 테스트 실행

[3] 선행기술 조사

개발하고자 하는 SW/시스템과 동일한 또는 유사한 기술이 이미 개발되었는지를 조사하세요. 관련 제품을 조사한 결과를 첨부하거나, 관련 논문/특허를 검색한 결과를 첨부해도 됩니다. 관련 제품/기술에 대한 간략한 소개 및 출처도 작성하세요.

답변:

수강신청은 티케팅 서비스와 유사한 프로세스이다. 국내의 경우 인터파크, 예스24, 멜론 티켓 등의 티켓팅 플랫폼이 존재한다.

인터파크 티케팅

- 인프라: 자체 IDC 기반 운영이나 빅데이터 EMR, AWS 클라우드를 하이브리드
- 컨테이너: Docker/K8s 기반 MSA
- 백엔드: 과거 모놀리식에서 MSA 변경 병행 및 DB는 Oracle과 MS-SQL 혼용
- 오토스케일링: AWS Auto Scaling(EC2, Spot), k8s HPA로 추정, F5/L7 Lb
- 트래픽 제어: NetFUNNEL 계열 가상 대기실, 캐싱, DB 트랜잭션 잠금

예스24 티켓팅 플랫폼

- 인프라: 온프레미스 Windows Server + IIS 기반 전살실
- 컨테이너: 미정
- 백엔드: 모놀리식 ASP.NET(C#), MS-SQL, Stored procedure
- 확장/부하분산: 사전 용량 확보, 하드웨어 및 IIS ARR 기반 LB
- 트래픽 제어: 외부 대기열 솔루션(NetFUNNEL 계열), DB 잠금

멜론티켓

- 인프라: 카카오 자체 DC 기반 Private Cloud
- 컨테이너: 카카오 내부 k8s 활용 추정
- 백엔드: 모듈화된 msa, RESTAPI
- 확장/부하분산: 카카오 클라우드 프로액티브 스케일링, L7 LB

- 트래픽 제어: NetFUNNEL 가상 대기실, AI 봇 차단, 캐싱, 빅데이터 예측

· 모놀리식(monolithic) 아키텍처: 하나의 애플리케이션 내에 UI, 비즈니스 로직, 데이터 접근 계층 등이 모두 통합되어 하나의 배포 단위로 운영되는 구조
· F5 L7 LB: F5 Networks 사의 L7 LB
· NetFUNNEL: Virtual Waiting Room 방식을 통해 웹 및 앱 과부하 방지하는 트래픽 관리
  - 원리
    1. 트래픽 급증 감지 시 신규 요청을 큐에 적재
    2. 사용자에게 대기 화면을 보여주며 순번 대기
    3. 설정된 속도로 순차적으로 서버에 요청 릴리스
  - 장점
    1. 서버 과부하 방지 및 안정성 보장
    2. 공정성 및 봇 차단
    3. 실시간 모니터링 관리

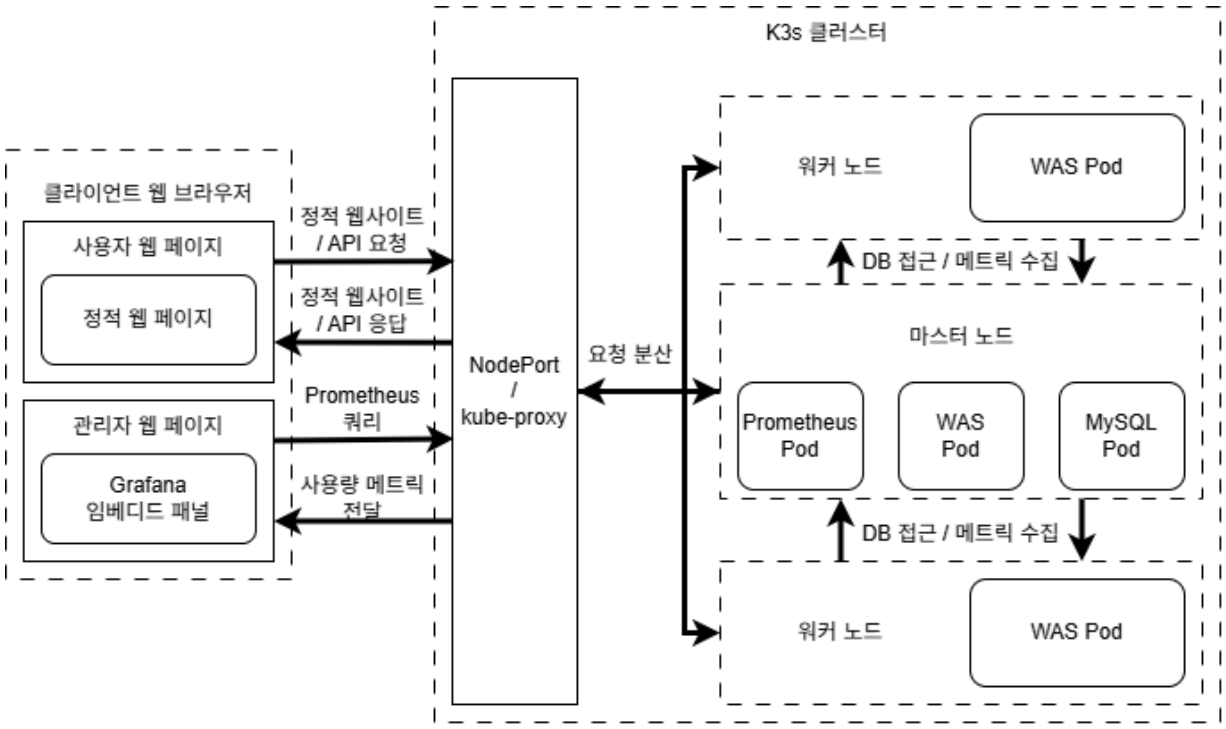
출처

- https://aws.amazon.com/ko/solutions/case-studies/interpark/(인터파크 티켓)
- https://www.kakaocorp.com/page/detail/8563(멜론 티켓)
- https://medium.com/stclab-tech-blog/how-a-ticketing-latecomer-gained-3x-more-users-e761d0f88905(멜론 티켓 트래픽 폭증)

[4] 제안하는 SW/시스템의 구성도

최종 결과물의 구성 및 동작방식을 다이어그램을 그려서 설명하세요. 예를 들어, [WAS]+[DB]로 구성된 SW인 경우, [사용자], [WAS], [DB]를 나타내는 다이어그램을 각각 그리고 [사용자]가 [WAS]로 어떤 요청을 보내는지, [WAS]가 사용자에게 어떤 답변을 보내는지, [WAS]와 [DB]사이에 어떤 요청/응답이 오가는지를 설명하는 다이어그램을 그리세요. 그림에 텍스트를 충분히 추가하여 어떤 데이터/명령이 오고 가는지 나타내세요.

답변:



- **사용자 웹 페이지**
  - 일반 사용자용 수강신청 UI
  - 정적 웹 페이지에서 fetch 요청

- **Grafana**
  - Prometheus에서 수집한 데이터 시각화

- **관리자 페이지**
  - 별도 웹 UI 제공
  - Grafana 대시보드를 iframe 등으로 임베딩하여 시각적 모니터링 제공

- **WAS Pod**
  - 수강 신청 API 제공 (/courses, /enroll, /load 등)
  - HPA에 의해 부하 증가 시 자동으로 복제 및 확장

- **MySQL Pod**
  - 수강 내역 저장
  - 모든 WAS Pod가 공유 접근

- **Prometheus Pod**
  - WAS Pod, 마스터 및 워커 노드, HPA 등에서 사용량 메트릭 수집

[5] 동작 환경

최종 결과물이 클라우드, PC/노트북, 모바일 기기(스마트폰 등), 라즈베리파이 등 어느 플랫폼에서 동작하는지에 대한 설명도 추가하세요.

답변:

클라우드 환경: 메인 서비스가 배포되는 환경이며, 오토스케일링을 통해 대량 요청에 대응

PC/노트북(웹브라우저): 수강신청 페이지에 접속해 신청하는 사용자 인터페이스(UI)를 제공

모바일 기기(스마트폰, 태블릿): 모바일 브라우저로 수강신청 가능

끝! 수고하셨습니다 ☺