Pusan National University

Computer Science and Engineering

Technical Report 2023-10

클라우드 기반 원격 DaaS

- 교육 플랫폼을 위한 클라우드 DaaS 솔루션 -

저자 1 202025181 박소현

저자 2 202055512 김기해

저자 3 202055520 김수현

지도교수 김태운

목 차

1. 서론

1.1. 연구 배경

1.1.1. 에듀테크 산업의 성장과 교육기관의 단말기 제공

코로나 19 이후 원격 수업의 중요성이 부각되는 가운데, 에듀테크 산업에 대한 관심도 높아지고 있다. HolonIQ 보고서에 따르면 2018년 전체 세계 교육 시장 규모에서 에듀테크 시장의 규모는 2.5%(1,530억불)였으나, 2025년에는 4.3%(3,420억불)에 이를 것으로 예상된다.[1]그러나 교육기관에서 제공하는 스마트 단말기와 컴퓨터실 환경이 일관성 없이 운영되어 학생들의 교육 환경의 일관성을 유지하는데 어려움을 겪고 있다. 경남 교육청은 1500억원대 예산으로 학생용 스마트 단말기 '아이북'을 보급했으나 성능과 활용성에 대한 비판이 나왔다. 또한, 학교 컴퓨터실에서 제공하는 데스크탑 환경과의 불일치로 교육 예산이 낭비되고 비대면 교육의 환경 변화에 대응하는데 어려움이 발생하고 있다.[2]

1.1.2. 정부의 공공기관 클라우드 전환 사업

한편, 정부는 2026년까지 모든 행정과 공공기관의 정보 자원을 클라우드로 전환하는 계획을 추진하고 있다. 이는 민간 클라우드 기술과 서비스를 공공에 도입하여 다양하고 긴급한 행정 수요에 신속하고 유연하게 대응하기 위함이다.[3] 교육 분야에서도 디지털 환경 개선과 교육 불평등 해소를 위한 다양한 클라우드 프로젝트가 진행되고 있다. 글로벌 교육 시장에서 클라우드 컴퓨팅 분야만 따져도, 연간 약 24.6%의 연평균 성장률로 2027년에는 129조 원(895억 3천만 달러)을 기대하고 있다.[4]

1.1.3. 코로나 이후 변화된 교육 및 근무의 형태

또한, 코로나 19의 영향으로 인해 원격 및 재택 근무가 활발해지면서 원격 데스크탑 서비스에 대한 중요성이 부각되고 있다. 원격 데스크탑 서비스는 사무실이나 교실을 실제로 찾아가지 않고도 업무와 학습을 진행할 수 있으므로 안전하고 효율적인 업무 및 학습 환경을 제공한다. 이와 같이 일관된 비대면 교육 환경을 위해 클라우드 기술을 통한 원격 교육 및 원격 데스크탑 서비스 개선이 필요한 과제로 부상하고 있다.

---

[1] 참고문헌 [1]

[2] 참고문헌 [2]

[3] 참고문헌 [3]

[4] 참고문헌 [4]

1.2. 기존 문제점

1.2.1. 웹 기반 원격 데스크탑의 문제점

기존의 웹 기반 원격 데스크탑은 클라우드 기반이 아니므로 제한된 리소스로 작동하기에 성능, 안정성, 확장성이 제한된다. 특히, 동시에 다수의 사용자가 접속하는 경우 서버 부하 문제가 발생하여 사용자 경험을 저해한다. 이로 인해 학생들은 느린 속도와 불안정한 연결로 인한 어려움을 겪을 수 있다.

1.2.2. 클라우드 기반 원격 데스크탑 서비스의 문제점

클라우드 기반 원격 데스크탑 서비스는 클라우드 인프라를 기반으로 하여 필요에 따라 리소스를 동적으로 할당하고 조절할 수 있다. 이는 뛰어난 성능과 확장성을 제공하며 다수의 사용자가 동시에 원활한 환경에서 작업할 수 있도록 도움을 준다.

그러나 기존의 클라우드 기반 원격 데스크탑 서비스는 클라이언트 앱을 다운로드하여 설치해야 하는 불편함이 있다. 이로 인해 사용자는 정기적으로 애플리케이션을 업데이트하고 관리해야 하며 일부 사용자에게는 추가적인 접근성 문제가 발생할 수 있다. 클라이언트 앱의 설치가 번거롭고 복잡해 일부 사용자는 서비스에 접근하는 데 어려움을 겪는다. 특히, 교육 환경에서는 이러한 불편함이 사용자들에게 부담으로 다가올 수 있다.

1.2.3. 교육 기기 호환성 문제

교육 환경에서 교육자와 학생 간의 일관된 데스크탑 환경을 제공하는 것은 어려운 과제이다. 사용자의 기기 및 운영 체제의 차이로 인해 교육 경험이 일관되게 유지되지 않을 수 있다. 특정 운영 체제에 최적화된 서비스는 다른 운영 체제에서는 원활한 이용을 제공하지 못할 수 있다. 이는 교육자가 강의나 교육 컨텐츠를 효과적으로 전달하고 학생들의 학습 경험의 통일성 확보에 어려움을 줄 수 있다.

1.3. 연구 목표

본 과제는 코로나 19를 기점으로 원격 교육과 재택 근무 필요성이 급증한 현대 사회에서, 교육 분야의 비대면 교육 혁신이 요구되는 시점에 제안된다. 연구 목표는 현대 사회의 동향과 요구사항을 반영하여 **클라우드 기반의 원격 데스크탑 서비스(DaaS, Desktop-as-a-Service)와** 여러 학습 관리 기능을 결합한 **교육 플랫폼(LMS, Learning Management System)을** 구축하는 것이다. 이를 통해 비대면 교육뿐만 아니라 학습 환경의 품질을 향상시키고 효과적인 학습 관리 서비스를 제공하고자 한다.

2. 연구 배경

2.1. 배경 지식

2.1.1. 아마존 웹 서비스(Amazon Web Service, AWS)

AWS는 아마존닷컴의 클라우드 컴퓨팅 플랫폼으로 다양한 클라우드 서비스를 제공하는 세계적인 클라우드 서비스 제공 업체이다. AWS의 주요 요소는 다음과 같다.

서비스명	설명
Amazon EC2	가상 서버를 제공하며 다양한 인스턴스 유형과 운영 체제에서 실행 가능하다.
Amazon S3	객체 스토리지 서비스로 데이터를 안전하게 저장하고 검색할 수 있는 확장 가능한 스토리지를 제공한다.
Amazon LoadBalancer	클라우드 환경에서 서버 부하 분산 및 고가용성을 제공하여 웹 애플리케이션, 마이크로서비스 아키텍처 및 다른 클라우드 기반 애플리케이션의 확장 및 안전성을 향상시킨다.
Amazon CloudWatch	모니터링 및 로깅 서비스로 AWS 리소스 및 애플리케이션의 성능 및 상태에 대한 실시간 데이터 및 로그를 수집, 보관, 모니터링할 수 있는 도구이다.
Route 53	관리형 DNS 시스템 서비스로, 웹 애플리케이션을 호스팅하거나 AWS 내부 리소스에 대한 DNS 관리와 라우팅을 위한 도구이다.
Certificate Manager	관리형 공개 및 사설 SSL/TSL 인증서 서비스로, 웹 애플리케이션 및 서비스를 안전하게 운영하기 위해 SSL/TLS 인증서를 관리하고 배포하는데 사용된다.

2.1.2. 네이버 클라우드 플랫폼(Naver Cloud Platform, Ncloud)

네이버 클라우드는 안정적인 국내 인프라를 기반으로 제공되는 기업용 클라우드 서비스로, 국내의 금융 기관, 공공 기관, 기업 및 국제 기업을 대상으로 클라우드 서비스를 제공하고 있다. Ncloud는 Classic 환경과 VPC(Virtual Private Cloud) 환경의 2가지 환경을 제공한다. Classic 환경은 논리적인 네트워크 분리가 없는 환경이고 VPC 환경은 논리적으로 격리된 네트워크를 생성하고 관리할 수 있는 환경이다. Ncloud 구축 시에는 VPC환경의 서비스를 활용하였고, 주요 요소는 다음과 같다.

서비스명	설명
Kubernetes Service (VPC)	클라우드 환경에서 Kubernetes 컨테이너 오케스트레이션 플랫폼을 제공하는 서비스이다. 클라우드 기반에서 애플리케이션을 관리하고 배포하기 위한 환경을 제공하며, 애플리케이션을 효과적으로 운영하고 확장할 수 있도록 돕는 역할을 한다.
Route (VPC)	네트워크 트래픽을 관리하고 라우팅하는 서비스로, 네트워크와 서비스 간의 통신을 제어하며 데이터의 흐름 방식을 정의한다.
NAT Gateway (VPC)	NAT(Network Address Translation)을 제공하는 관리형 네트워크 서비스로 클라우드 환경에서 인터넷 연결을 허용하면서 내부 네트워크 리소스를 보호하는 데 사용된다.
Server (VPC)	가상화된 컴퓨팅 인프라를 제공하며, 클라우드 환경에서 다양한 운영 체제 실행 및 애플리케이션 호스팅에 사용된다.

2.1.3. 도커(Docker)

도커는 프로그램들을 프로세스 격리 기술들을 사용해 컨테이너로 실행하고 관리한다. 도커 이미지는 어떤 소프트웨어를 실행하기 위한 파일과 설정 등을 포함한 패키지로, 도커 엔진에서 실행될 때 컨테이너라 불리는 가상 환경을 생성한다. 이미지는 읽기 전용이며 변하지 않기 때문에 어디서든 일관된 환경을 제공하며 배포와 확장을 용이하게 한다. 도커 컨테이너는 이미지의 인스턴스로, 일종의 소프트웨어 실행에 필요한 모든 구성 요소를 포함하는 완전한 파일 시스템 안에 감싼다. 이는 격리된 환경에서 항상 일관된 애플리케이션이 실행을 보장한다.

2.1.4. 쿠버네티스(Kubernetes, k8s)

쿠버네티스는 컨테이너 기반의 애플리케이션을 개발하고 배포할 수 있도록 설계된 오픈소스 컨테이너 플랫폼으로, 여러 컨테이너화된 애플리케이션을 자동으로 배포, 스케일링 및 관리하는 역할을 수행한다. 쿠버네티스의 자가 치유(Self-healing) 기능은 오류가 발생한 컨테이너를 재시작하고, 노드(Node)가 죽었을 때, 컨테이너를 교체하기 위해 다시 스케줄을 하는 방식으로 정상적인 컨테이너만을 대상으로 애플리케이션을 운영할 수 있게 해준다. 또한, 쿠버네티스는 자원 사용률에 배포된 애플리케이션을 늘이거나 줄일 수 있어 효율적인 자원 사용이 가능하다. 이는 다양한 환경에서 실행되는 애플리케이션을 관리하며, 클러스터를 쉽게 확장할 수 있다. 쿠버네티스의 주요 요소는 다음과 같다.

리소스	설명
Cluster	쿠버네티스의 여러 리소스를 관리하기 위한 집합체이다. 마스터 노드와 워커 노드를 이용해 하나의 쿠버네티스 클러스터를 구성할 수 있다.
Master Node	쿠버네티스 클러스터 전체를 관리하는 시스템으로, 컨트롤 플레인(control plane)이라고도 한다.
Worker Node	마스터 노드에 의해 명령을 받아 파드를 생성하고 서비스하다고 해서 컴퓨팅 머신(computing machine)이라고도 한다. 워커노드에는 컨테이너 런타임이 포함되어 있다.
Container Runtime	파드를 실행하는 엔진으로 워커노드에 포함되어 있다. 대표적으로 도커가 있으며, 그 밖에도 컨테이너디, 크라이오 등이 런타임 엔진으로 많이 사용되고 있다.
Pod	쿠버네티스에서 가장 작은 배포 단위로 하나 이상의 컨테이너로 이루어져 있다. 동일한 pod 내의 컨테이너는 같은 네트워크 네임스페이스와 스토리지를 공유하고 함께 배포되고 스케일링된다.
Deployment	Pod의 세트를 정의하고 관리하는 쿠버네티스 리소스이다. Deployment를 사용하면 애플리케이션을 업데이트하거나 롤백하고, 자동으로 스케일링한다.
Service	Pod에 대한 로드 밸런싱과 네트워크 서비스를 제공하는 쿠버네티스 리소스이다. Service를 통해 Pod에 접근할 수 있다.

2.1.5. 하버(Harbor)

하버는 도커 이미지를 안전하게 저장하고 관리하는 도커 레지스트리 서비스이다. 하버는 외부 서비스를 사용하는 대신 도커 이미지 파일을 내부에 저장하므로, 이미지의 안전한 보관과 효율적인 관리를 지원한다.

2.1.6. 프로메테우스(Prometheus)

프로메테우스는 오픈 소스 기반의 모니터링 및 경고 시스템으로, 시계열 데이터를 기반으로 작동한다. 이 시스템은 서버의 성능, 리소스 사용, 애플리케이션 상태 등 다양한 메트릭 데이터를 지속적으로 수집하고 저장한다.

2.1.7. 그라파나(Grafana)

그라파나는 다양한 데이터 원본에서 데이터를 가져와 대시보드 및 시각화를 효과적으로 생성하는 도구이다. 그라파나는 프로메테우스와 통합하여 프로메테우스로부터 수집된 데이터를 시각적으로 표현할 수 있다.

2.1.8. 마이크로서비스 아키텍처(Microservices Architecture, MSA)

마이크로서비스 아키텍처는 소프트웨어 시스템을 작고 독립적인 서비스로 분할하는 소프트웨어 아키텍처 디자인 패턴이다. 각 서비스는 특정 기능 또는 업무를 담당하며, 서로 독립적으로 배포, 운영 및 확장될 수 있다.

2.1.9. 플라스크(Flask)

플라스크는 파이썬으로 웹 애플리케이션을 빠르고 간결하게 개발할 수 있도록 돕는 경량하고 간결한 웹 프레임워크이다. 플라스크는 파이썬의 간결한 문법과 함께 사용되며, 웹 애플리케이션을 빠르고 효율적으로 구축할 수 있도록 다양한 확장 기능과 라이브러리를 제공한다.

2.1.10. 스프링 부트(Spring Boot)

스프링 부트는 스프링 프레임워크 기반의 자바 기반 웹 애플리케이션을 빠르게 개발하고 구축할 수 있도록 도와주는 프레임워크이다. 스프링 프레임워크에서 제공하는 REST Template은 HTTP 통신을 단순화하는 도구로 스프링 부트 애플리케이션과 플라스크 애플리케이션 간의 통신에 사용된다.

2.1.11. 타임리프(Thymeleaf)

타임리프는 서버 사이드 자바 템플릿 엔진으로 스프링 부트와 자주 사용되는 템플릿 엔진 중 하나이다. HTML, XML, JavaScript, CSS 등의 마크업을 자바 코드와 통합하여 동적으로 생성한다.

2.1.12. MariaDB

MariaDB는 오픈 소스 기반의 관계형 데이터베이스 관리 시스템(RDBMS)으로, 다양한 애플리케이션과 웹 서비스에서 널리 사용되며 데이터 저장과 관리에 활용된다.

2.2. 서비스 기획

2.2.1. 기능 정의

서비스의 주요 기능과 사용자 요구사항을 구체적으로 정의하고 프로젝트 진행의 기준을 제시하기 위해 기능 정의서를 작성했다. 다음은 서비스에 대한 기능 정의서이다.

요구사항 ID	요구사항명	기능명	상세 설명	필수 데이터
MEM01	로그인	자체 로그인 기능	서비스를 사용하기 위해 로그인 필수	ID, 비밀번호
		연동 로그인 기능	구글 또는 네이버 연동으로 아이디, 비밀번호 입력 없이 간편하게 로그인 하는 기능	
		아이디 찾기	메일과 사용자 이름을 입력하면 해당하는 아이디 찾아주는 기능	이름, 이메일
		비밀번호 재설정	아이디와 메일을 입력하면 새로운 비밀번호 재설정	ID, 이메일
MEM02	회원가입	자체 회원가입 기능	서비스 사용을 위해 기본 정보를 입력해 회원가입 필수	아이디, 비밀번호, 이름, 닉네임, 이메일
		연동 회원가입 기능	구글 또는 네이버 연동 회원가입 가능	
MEM03	회원 정보 수정	회원 정보 수정	마이페이지에서 아이디와 이름 외의 정보는 변경 가능	
MEM04	회원 탈퇴	회원 탈퇴	마이페이지 회원 정보 수정에서 탈퇴하기 버튼 클릭시 팝업 박스로 한번 더 확인하고 비밀번호 입력 후 탈퇴	비밀번호, 탈퇴 체크
MEM05	로그아웃	로그아웃	로그아웃 기능	
MAIN01	가상 환경 생성	가상 환경 생성	메인 페이지에서 '생성하기' 버튼 클릭시 (원하는 스펙의) 가상 환경 생성되는 기능	가상환경 이름, 공개 범위, 접속자 권한 범위
MAIN02	가상 환경 접속	가상 환경 접속	자신의 가상 머신 목록 중에서 하나의 가상머신 선택 후 '접속하기' 버튼 클릭시 가상 환경에 접속	
MAIN03	가상 환경 상태 저장	가상 환경 상태 저장	가상 환경 접속한 상태에서 가상 환경 종료 전, 상단에 '저장' 버튼으로 가상 환경의 상태 저장	
MAIN04	가상 환경 종료	가상 환경 종료	가상 환경 접속한 상태에서 상단에 '종료' 버튼 클릭시 팝업 박스로 한 번 더 확인 후 가상 환경 종료	
MAIN05	가상 환경 내보내기	가상 환경 내보내기	메인 페이지에서 자신의 가상 머신 목록 중에서 하나의 가상 머신 선택 후 '내보내기' 버튼 클릭시 팝업 박스로 한번 더 확인 후 키(key) 값 생성	
MAIN06	가상 환경 로드	가상 환경 로드	메인 페이지에서 '로드하기'에 키(key) 값 입력 시 해당 가상머신 로드 가능	가상머신 key 값
MAIN07	가상 환경 수정	가상 환경 정보 수정	가상 머신 목록 중 하나의 가상 머신 선택 후 설정에서 가상 환경 이름, 공개범위, 접속자 권한 범위, 연결할 수업 변경 가능	가상 환경 이름, 공개 범위, 접속자 권한 범위
MAIN07	가상 환경 삭제	가상 환경 삭제	가상 머신 목록 중 하나의 가상 머신 선택 후 설정에서 '삭제하기' 버튼 클릭시 팝업 박스로 한 번 더 확인하고 가상 환경 삭제	

그림 1. 기능 정의서 1

CLASS01	강좌 생성	강좌 생성하기	강좌 페이지에서 '강좌 생성하기' 클릭하고 이름과 설명을 입력해 강좌 생성	강좌 이름, 강좌 설명
CLASS02	강좌 신청	강좌 신청하기	일반 사용자는 강좌 페이지에서 '강좌 신청하기' 누르면 신청 가능한 강좌 목록이 뜨고, 원하는 강좌에 '신청' 버튼으로 신청	강좌 이름, 강좌 생성자 아이디
CLASS03		강좌 신청 요청 수락	관리자는 마이페이지 '내 소식' 또는 강좌 페이지의 설정 및 관리 페이지에서 수강생들의 강좌 신청 승인 수락, 거절 버튼으로 수락, 거절 가능	
CLASS04	강좌 조회	강좌 목록 조회	강좌 페이지에서 '나의 강좌'에서 신청한 강좌 목록 확인 가능, 원하는 강좌 클릭시 해당 강좌 페이지로 이동	
CLASS05	강좌 수정	강좌 정보 수정	생성자는 해당 강좌 페이지에서 '설정 및 관리 페이지'에서 수강생 관리 및 설정할 수 있음	
CLASS06	강좌 취소	강좌 취소	해당 강좌 페이지에서 '강좌 취소하기' 버튼 클릭 시 팝업 박스로 다시 한 번 확인 후 강좌 취소	
CLASS07		강좌 삭제	커뮤니티 생성자는 마이페이지 안 커뮤니티 관리 페이지에서 해당 커뮤니티 삭제 버튼 클릭 시 팝업 박스로 다시 한 번 확인 후 커뮤니티 삭제	
POST01	공지사항 게시판	공지사항 글 작성	강좌 생성자만 공지사항 글 작성 가능 (only 전체 공개)	제목, 내용
POST02		공지사항 글 수정	강좌 생성자만 공지사항 글 수정	
POST03		공지사항 글 조회	해당 수강생들은 공지사항 글 조회 가능	
POST04		공지사항 글 삭제	강좌 생성자만 공지사항 글 삭제	
POST05	질문 게시판	질문 글 작성	강좌 수강생들은 질문 글 작성 가능 (전체 공개, 생성자 공개)	제목, 내용, 공개 범위
POST06		질문 글 수정	글 작성한 수강생만 질문 글 수정 가능	
POST07		질문 글 조회	글 공개 범위에 따라 질문 글 조회 가능	
POST08		질문 글 삭제	글 작성한 수강생만 질문 글 삭제 가능	
POST09	자유 게시판	자유 글 작성	해당 강좌 수강생들은 자유 글 작성 가능 (only 전체 공개)	제목, 내용, 공개 범위
POST10		자유 글 수정	글 작성한 수강생만 자유 글 수정 가능	
POST11		자유 글 조회	자유 글 조회 가능	
POST12		자유 글 삭제	글 작성한 수강생만 자유 글 삭제 가능	

그림 2. 기능 정의서 2

2.2.2. 화면 설계

사용자 경험을 고려한 직관적이고 효율적인 인터페이스를 제공하기 위해 화면 설계서를 작성한다.

그림 3. 화면 설계서 예시

2.2.3. DataBase 설계

데이터의 효율적인 저장, 검색을 위한 구조와 관계를 정의하고 데이터 일관성을 유지하기 위해 ERD를 설계한다.

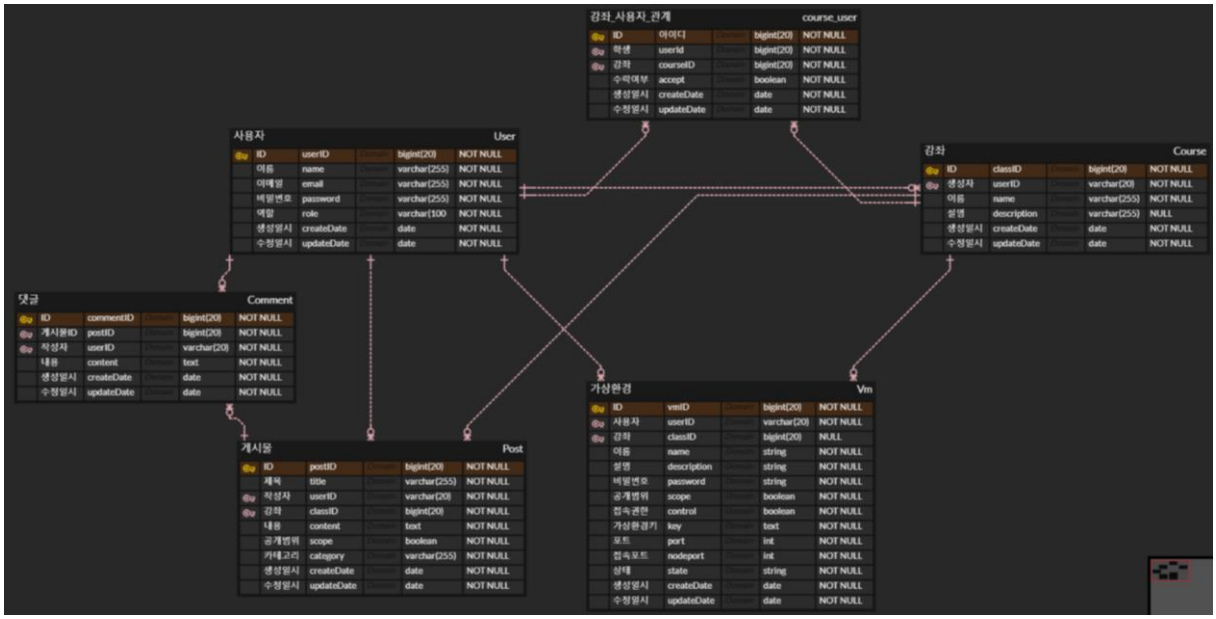

그림 4. ERD 설계

우선 사용자의 정보를 저장하는 User, 가상환경 정보를 저장하는 Vm, 강좌 정보를 저장하는 Course, 게시물 정보를 저장하는 Post, 댓글 정보를 저장하는 Comment 가 있다.

하나의 강좌에 여러 사용자가 속하고, 한 사용자는 여러 강좌에 속할 수 있으므로 이를 관리하기 위한 강좌-사용자-관계 테이블을 생성한다. 또한 가상환경의 생성자는 User 를 참조하고, 가상환경은 강좌에 속할 수 있으므로 Course 를 참조한다.

2.3. 개발 환경

분야	사용 기술 및 도구
Programming Languages	- Java 17 - Python 3.8.10
Framework	- Spring Boot 3.1.2 - Flask 3.0.0
Template Engine	- Thymeleaf 3.1.2
Build Tool	- Gradle 8.2.1
Cloud	- Docker 24.0.6 - Kubernetes 1.25.8 - Amazon Web Services(AWS) - Naver CloudPlatform
DataBase	- MariaDB 10.0
Registry	- Harbor 2.8.2 - AWS S3
Monitoring	- Prometheus 2.20.1 - Grafana 8.5.27 - AWS CloudWatch
Version Control	- Git 2.34.1

2.4. 용어 정리

프로젝트와 관련하여 사용되는 주요 용어들에 대한 정의이다. 연구 내용 소개 전 보고서의 이해를 돕기 위해 용어에 대한 설명을 제시한다.

용어	정의
사용자	해당 서비스를 이용하는 모든 개인
수강생	서비스에서 강좌를 신청하는 사용자
교육자	서비스에서 강좌를 생성하고 관리하는 사용자
접속자	특정 계정의 가상 환경을 관찰할 수 있는 권한을 가진 사용자
관리자	해당 서비스를 관리하는 운영 담당자

3. 연구 내용

3.1. 서비스 전체 구성도

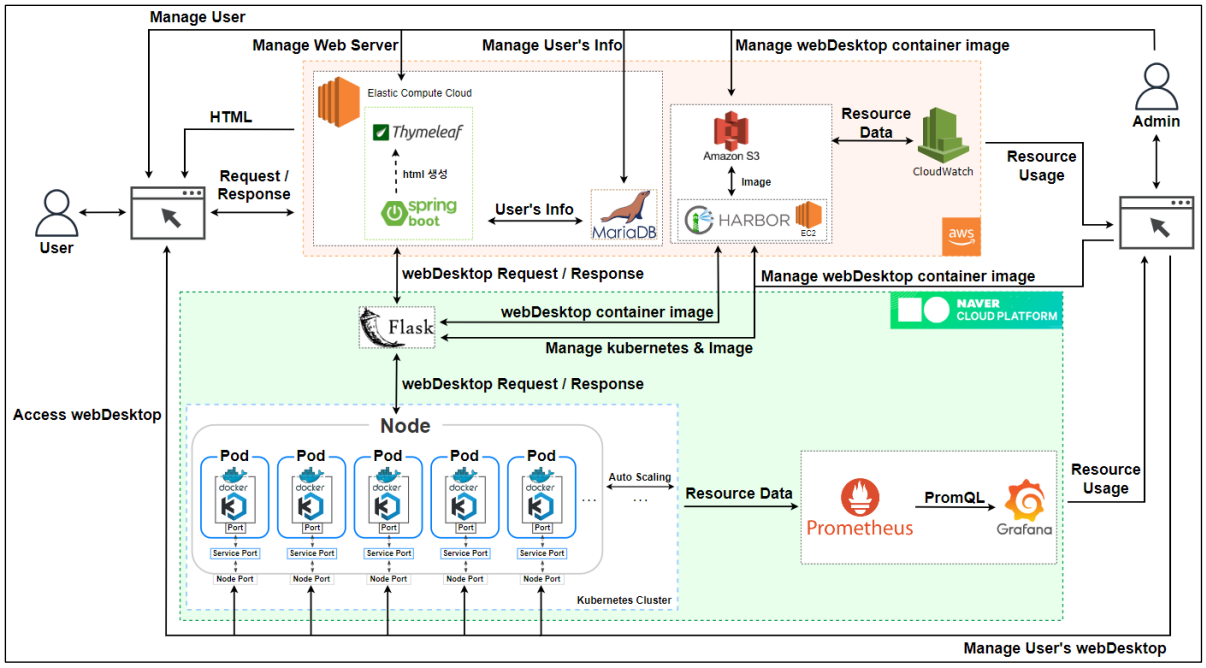

그림 5. 서비스 전체 구성도

구성	설명
멀티 클라우드	위 서비스는 멀티 클라우드 아키텍처로 구성 되어있으며, 클라우드 서비스 공급자로는 사용한 AWS와 Naver CloudPlatform을 선택했다. 멀티 클라우드 환경을 구축함으로써 다음과 같은 이점을 얻을 수 있다. **고가용성 및 신뢰성 향상** 두 가지 클라우드 플랫폼을 사용함으로써 서비스의 고가용성을 확보하고, 단일 클라우드 장애로 인한 중단을 방지한다. 한 클라우드에서 장애가 발생해도 다른 클라우드에서 서비스를 계속 운영할 수 있기 때문에 신뢰성이 향상된다. **비즈니스 리스크 분산** 클라우드 서비스 공급자 간에 계약 조건, 가용성, 가격, 보안 등이 다를 수 있기 때문에, 어떤 이슈가 한 클라우드에서 발생하더라도 다른 클라우드로 이전하여 서비스를 지속할 수 있다.

	유연성과 확장성 멀티 클라우드 아키텍처는 서비스의 확장성을 향상시키며, 미래에 대비해 유연성을 제공한다. 서비스 확장을 통해 서비스의 성능을 필요에 따라 조절할 수 있다. 또한, 새로운 클라우드 공급자를 도입하거나, 기존 클라우드 리소스를 확장함으로써 비즈니스 요구 사항이나 기술적인 변화에 더 잘 대응할 수 있도록 도와준다. **MSA(Micro Service Architecture) 설계** MSA(Micro Service Architecture)는 시스템을 관리하기 쉬운 작고 독립적인 서비스로 분할하는 설계로, 독립된 서비스는 API를 통해 서로 통신한다. 이는 시스템의 효율적인 리소스 관리, 확장성과 유지보수를 가능하게 한다.
AWS	AWS 환경에서는 다음의 서비스가 구축 되어있다. **Web Server** 웹 서버는 AWS의 EC2를 사용하고 있으며, 웹 애플리케이션 호스팅 및 웹 사이트 제공에 사용된다. 사용자의 웹 브라우저에서 요청을 받고, 웹 페이지 및 컨텐츠를 제공한다. 사용자의 정보는 MariaDB에 저장된다. MariaDB와 SpringBoot 웹 애플리케이션은 container화 된 애플리케이션으로 EC2 위에서 docker container로 실행된다. **Private Docker Registry** 오픈 소스 컨테이너 레지스트리인 Harbor와 AWS 클라우드 스토리지 서비스인 S3를 사용하여 Private Docker Registry를 구성하였다. Harbor는 AWS의 EC2위에서 multi docker container 형태로 실행된다. Harbor와 S3를 같이 사용함으로써 Docker Image를 private하게 저장할 수 있다. **S3 Monitoring** S3의 리소스 사용량을 보기 위한 Monitoring이다. AWS의 CloudWatch 서비스를 사용하여 S3의 리소스를 모니터링, 로깅, 이벤트 추적 등을 수행할 수 있다.

	이러한 서비스들을 활용하여 AWS 환경에서는 안정적인 웹 서버 호스팅, Docker Container Image 저장 및 관리, 그리고 S3 모니터링을 통해 실시간으로 적절한 대응하여 데이터를 안전하게 관리할 수 있다.
Naver CloudPlatform	Naver CloudPlatform에는 다음의 환경이 구축 되어있다. **Kubernetes & Web Desktop Image Manager** Kubernetes 관리 및 Web Desktop Image를 관리하는 역할을 수행하며, Flask 기반으로 실행된다. 사용자의 가상환경을 관리하는 서버에게 명령을 내려 Kubernetes를 관리한다. 사용자가 가상 환경을 사용하는 동안에는 해당 Web Desktop Image를 임시적으로 보관한다. **Web Desktop Manger** 실제 가상환경이 실행, 관리되는 곳으로, Kubernetes 기반으로 구축되어 있다. 여러 가상 환경을 관리하고, 사용자들의 리소스 사용량에 따라 Kubernetes 환경을 Scaling 하여 적절하게 대응한다. **Kubernetes Monitoring** Kubernetes의 리소스 사용량을 보기 위한 모니터링이다. Kubernetes 클러스터의 상태, 리소스 사용량 및 가상환경 성능을 실시간으로 모니터링하고 관리한다. 이러한 서비스들을 활용하여 Naver CloudPlatform은 사용자 가상 환경 관리, Kubernetes 모니터링을 통해 안정성과 성능을 제공한다.

3.2. 사용자 서비스

3.2.1. Web Server 구성도

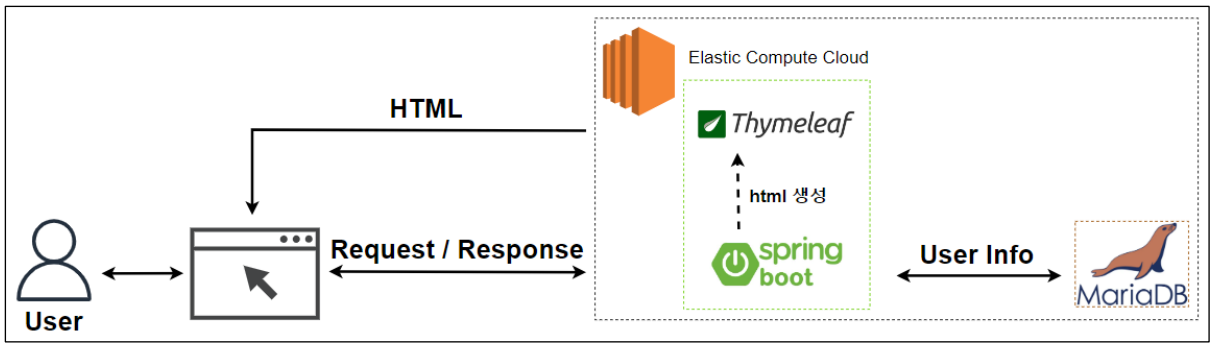

그림 6. Web Server 설계 구조

Web Server는 AWS의 EC2 인스턴스를 활용하여 웹 애플리케이션 호스팅과 웹 사이트를 제공한다. 웹 애플리케이션은 웹 페이지 내용을 서버에서 동적으로 생성한 후 브라우저로 전송하는 SSR(Server Side Rendering) 방식으로 구축되어 있다. Web Server가 수행하는 역할은 다음과 같다.

- **사용자 요청 처리** : 웹 페이지의 사용자 요청에 따른 응답 제공
- **콘텐츠 제공** : SSR을 통해 동적으로 생성된 웹 페이지와 콘텐츠를 웹 브라우저로 전송하여 사용자에게 화면을 제공
- **사용자 정보 저장** : 서비스 관련 정보 및 사용자 정보 MariaDB에 저장

MariaDB와 SpringBoot 웹 애플리케이션은 container화된 애플리케이션으로 AWS EC2 인스턴스 위에서 Docker Container형태로 실행된다. Container 기술은 애플리케이션 배포 및 관리를 단순화하고 표준화 하며, 이는 추후 CI/CD 관리에 편의성을 제공한다.

3.2.2. Web Server 구축 과정

[EC2 인스턴스 생성]

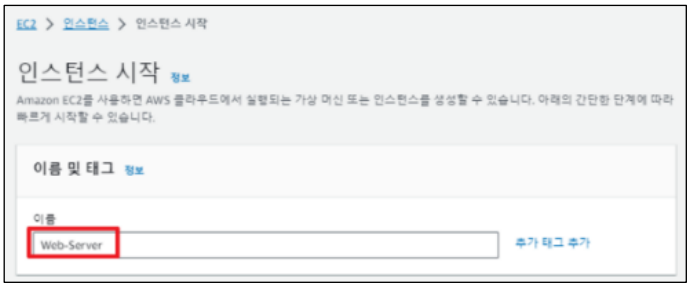

그림 7. EC2 인스턴스 이름

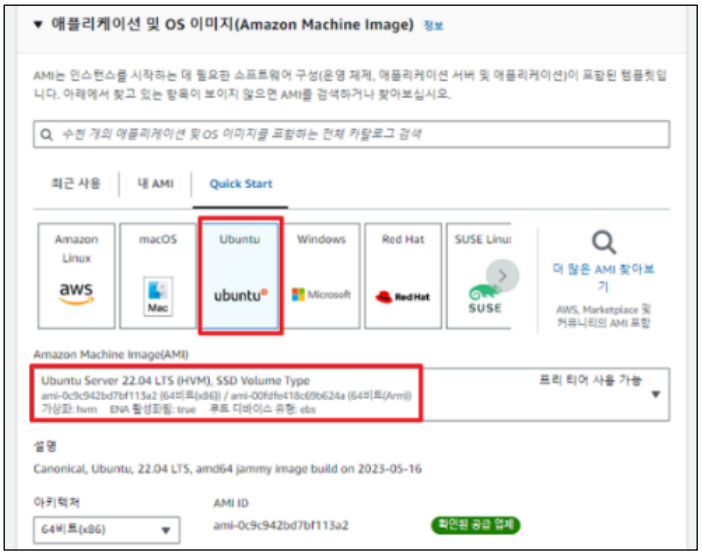

그림 8. EC2 인스턴스 서버 스펙

AWS의 EC2 인스턴스를 생성한다. EC2 인스턴스는 t2.micro를 선택하였다.

[도메인 등록]

그림 9. 구매한 메인 도메인에 EC2 인스턴스의 IP 등록

Route 53을 통해 EC2의 IP를 구매한 메인 도메인(p2kcloud.com)에 A 레코드 형식으로 등록하였다.

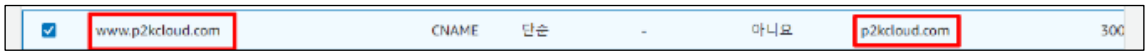

그림 10. 서브 도메인 추가해서 Route 53에 등록

메인 도메인(p2kcloud.com)을 CNAME 레코드 형식으로 서브도메인(www.p2kcloud.com)을 추가하여 Route 53에 등록한다.

[EC2 환경 설정]

생성된 EC2에 웹 애플리케이션인 java 프로젝트를 위해 JDK 17과 Git을 설치한다. Container 기술을 통해 런타임 환경을 격리하여 리소스를 효율적으로 관리하기 위해 Docker를 설치한다.

```
FROM openjdk:17

ARG JAR_FILE="./build/libs/p2k-0.0.1-SNAPSHOT.jar"

COPY ${JAR_FILE} p2k.jar

ENV PROFILE prod,aws,mariaDB

ENTRYPOINT ["java","-Dspring.profiles.active=${PROFILE}","-jar","/p2k.jar"]
```

그림 12. SpringBoot 애플리케이션을 Docker Image로 만들기 위한 Dockerfile

SpringBoot 애플리케이션을 Docker Image로 패키지화 함으로서 일관적인 배포 환경을 유지하기 위해 Dockerfile을 작성한다.

```
version: '3'

services:
  db:
    image: mariadb:10
    container_name: mariadb
    ports:
      - 3307:3306
    environment:
      MARIADB_DATABASE: p2k # DB 이름
      MARIADB_ROOT_PASSWORD: 'DB 접속 비밀번호'
      MARIADB_ROOT_HOST: "%"
      TZ: Asia/Seoul
    volumes:
      #  DB 서버 설정 파일
      - ./mariadb/conf.d:/etc/mysql/conf.d
      # DB 의 데이터가 파일 형태로 저장되는 공간
      - ./mariadb/data:/var/lib/mysql
      # Docker 컨테이너가 최초 실행 시 불러올 스크립트가 위치하는 공간
      - ./mariadb/initdb.d:/docker-entrypoint-initdb.d
    command: ['--character-set-server=utf8mb4', '--collation-
server=utf8mb4_unicode_ci', '--bind-address=0.0.0.0']

  application:
    build: .
    container_name: springboot
    ports:
      - 80:8080
    depends_on:
      - db
```

그림 11. MariaDB와 SpringBoot 웹 애플리케이션을 Container로 띄우기 위한 docker-compose.yml 파일

Container를 사용해 데이터베이스인 MariaDB와 SpringBoot 애플리케이션을 실행하기 위해 각각의 Container를 docker-compose.yml 파일에 정의한다.

```
spring:
  datasource:
    url: jdbc:mariadb://mariadb:3306/p2k
    driver-class-name: org.mariadb.jdbc.Driver
    username: root
    password: # password
  jpa:
    database-platform: org.hibernate.dialect.MariaDBDialect
    hibernate:
      ddl-auto: update
    show-sql: true
    generate-ddl: true
    properties:
```

그림 13. application-MariaDB.yml 파일

DB와 JPA(Java Persistence API) 관련 설정을 정의하여 Spring Boot 애플리케이션에서 DB 연결과 ORM(Object-Relational Mapping) 설정을 정의하기 위한 application-mariaDB.yml 파일을 작성한다.

3.2.3. 메인 페이지

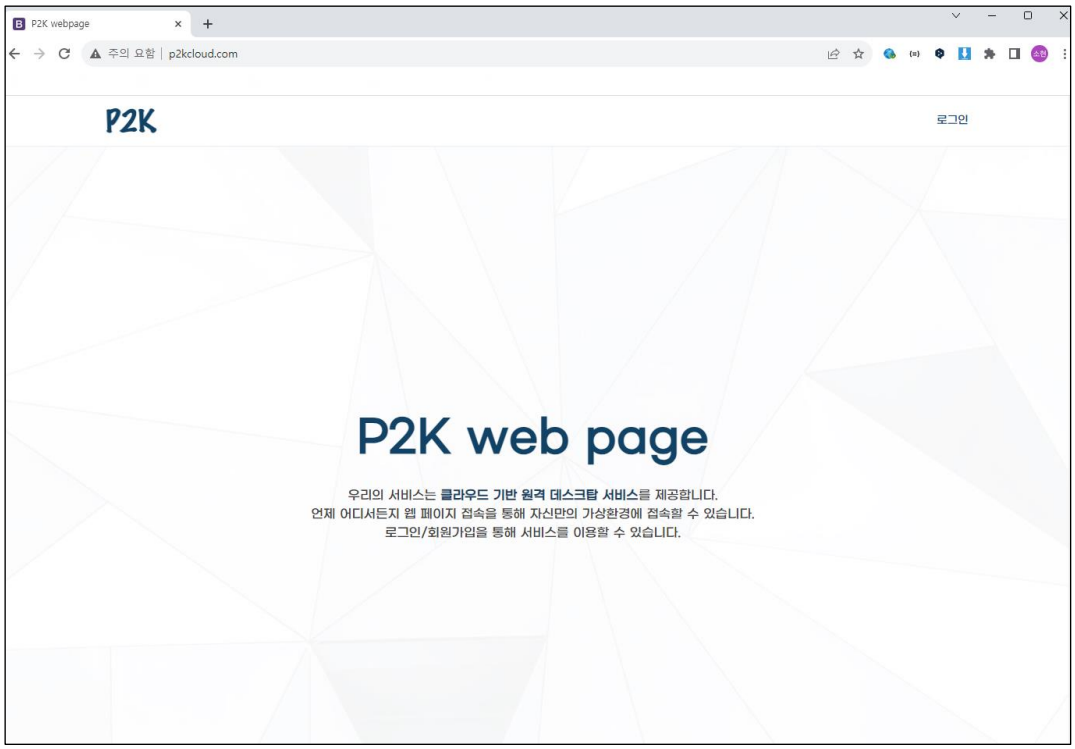

그림 14. 홈페이지 메인

설정한 도메인(www.p2kcloud.com)을 통해 외부 사용자는 웹 브라우저를 통해 EC2 인스턴스 내에서 호스팅되는 Docker 컨테이너 내의 애플리케이션에 접근할 수 있다.

3.2.4. 회원가입

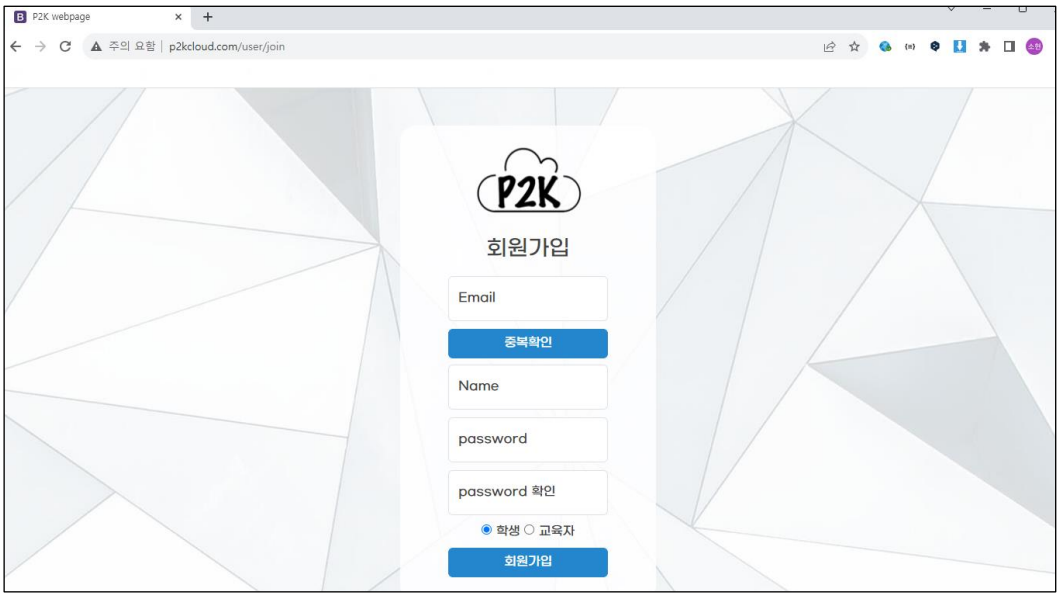

그림 15. 사용자 - 회원가입

사용자는 이메일, 이름, 비밀번호, 역할(학생, 교육자)을 선택한 후 회원가입을 할 수 있다. 교육자는 관리자의 승인을 받아야 권한이 부여된다.

3.2.5. 로그인/소셜 로그인

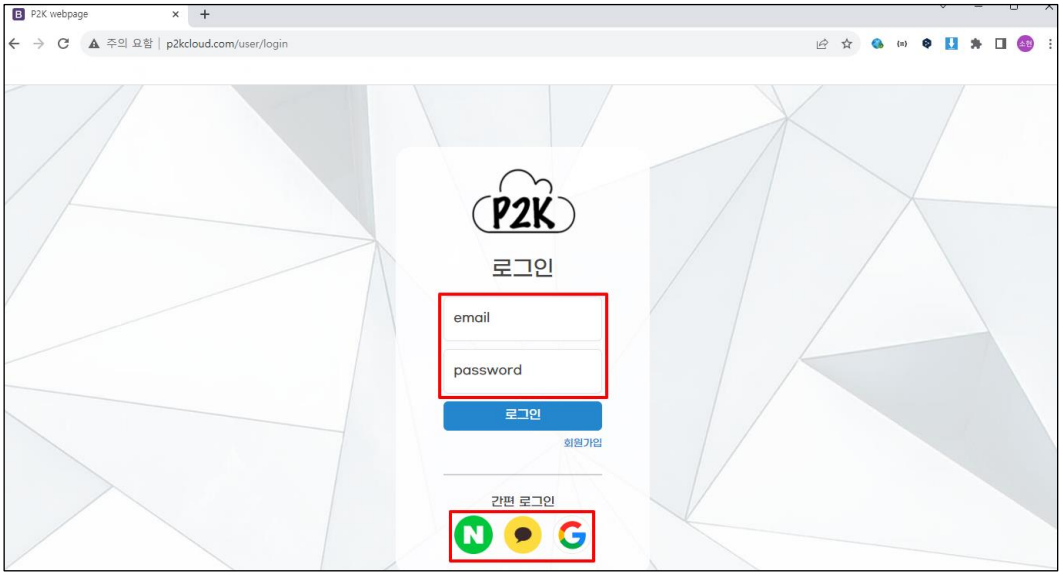

그림 16. 사용자 - 로그인

사용자는 가입된 이메일과 비밀번호로 로그인 할 수 있다. 또한 소셜 로그인을 제공해 간편하게 회원 가입 및 로그인을 할 수 있다.

3.2.6. 메인 화면

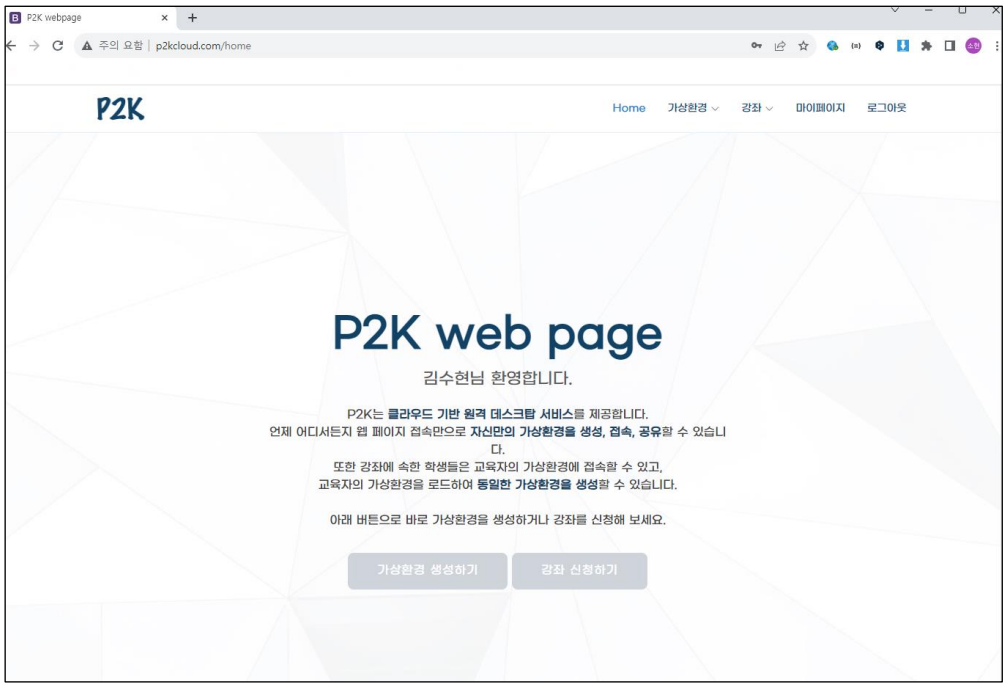

그림 17. 사용자 - 로그인 후 메인 페이지

처음 서비스를 이용하는 사용자들을 위해 메인 화면에서 서비스에 대한 간략한 소개와 설명을 제공한다. 또한 가상환경 생성하기, 강좌 신청하기 버튼을 제공하여 처음 서비스를 이용하는 사용자들에게 손쉬운 사용과 이해를 돕는다.

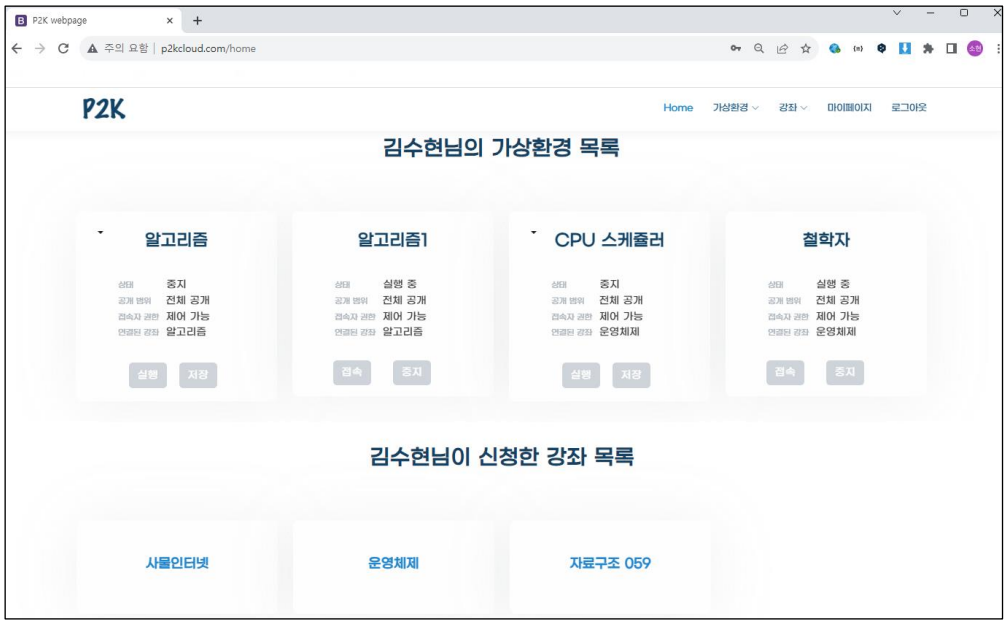

그림 18. 사용자 - 메인 페이지에서의 가상환경 목록과 신청한 강좌 목록

사용자는 스크롤로 자신이 생성한 가상환경 목록과 신청한 강좌 목록을 조회한다.

3.2.7. 사용자 정보 관리

[회원 정보 수정]

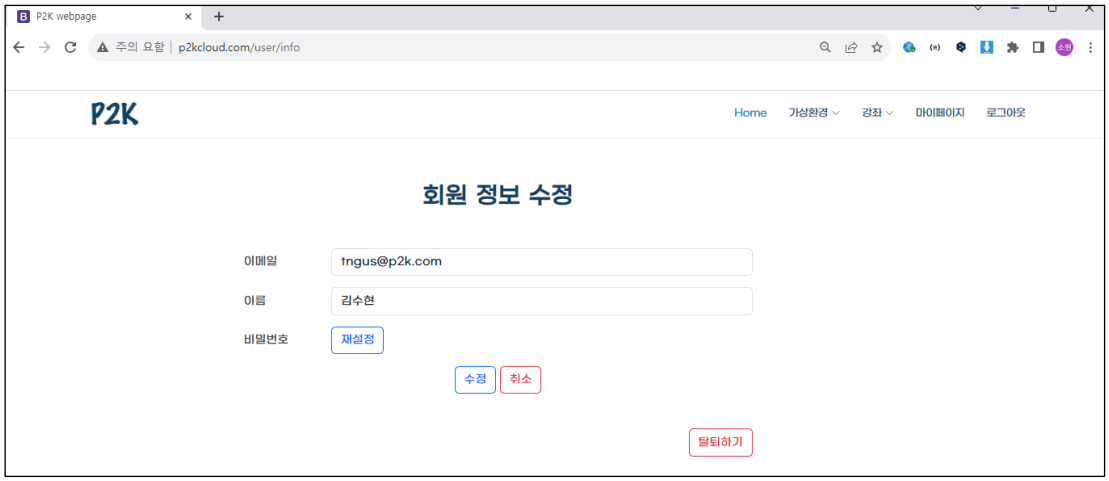

그림 19. 사용자 - 회원 정보 수정

사용자는 마이페이지에서 이메일, 이름 등의 정보를 변경할 수 있으며, 탈퇴하기 버튼을 통해 해당 서비스에서 탈퇴할 수 있다.

[비밀번호 재설정]

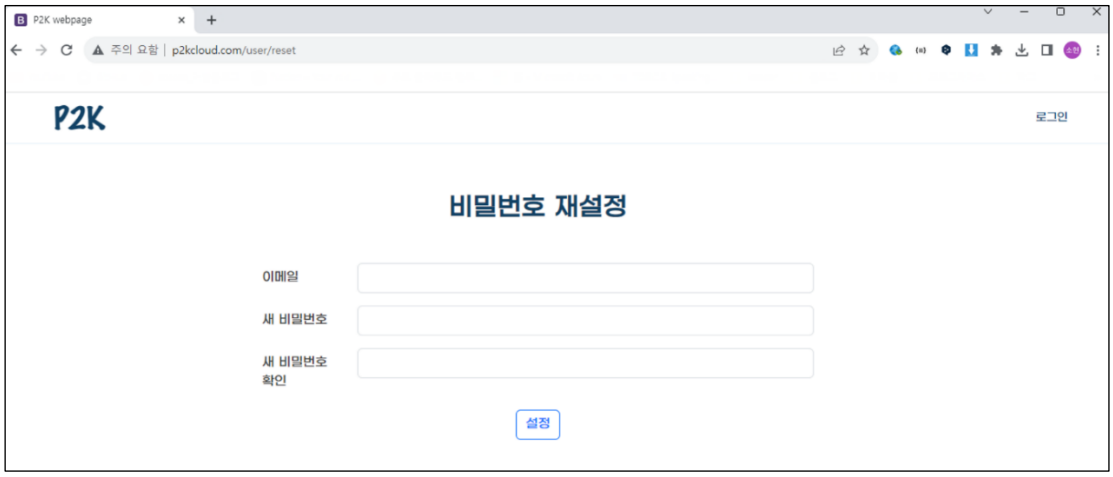

그림 20. 사용자 - 비밀번호 재설정

사용자는 비밀번호 재설정 페이지에서 이메일, 새 비밀번호, 새 비밀번호를 입력해 비밀번호를 변경할 수 있다.

3.3. 가상환경 서비스

3.3.1. 가상환경 Server 구성도

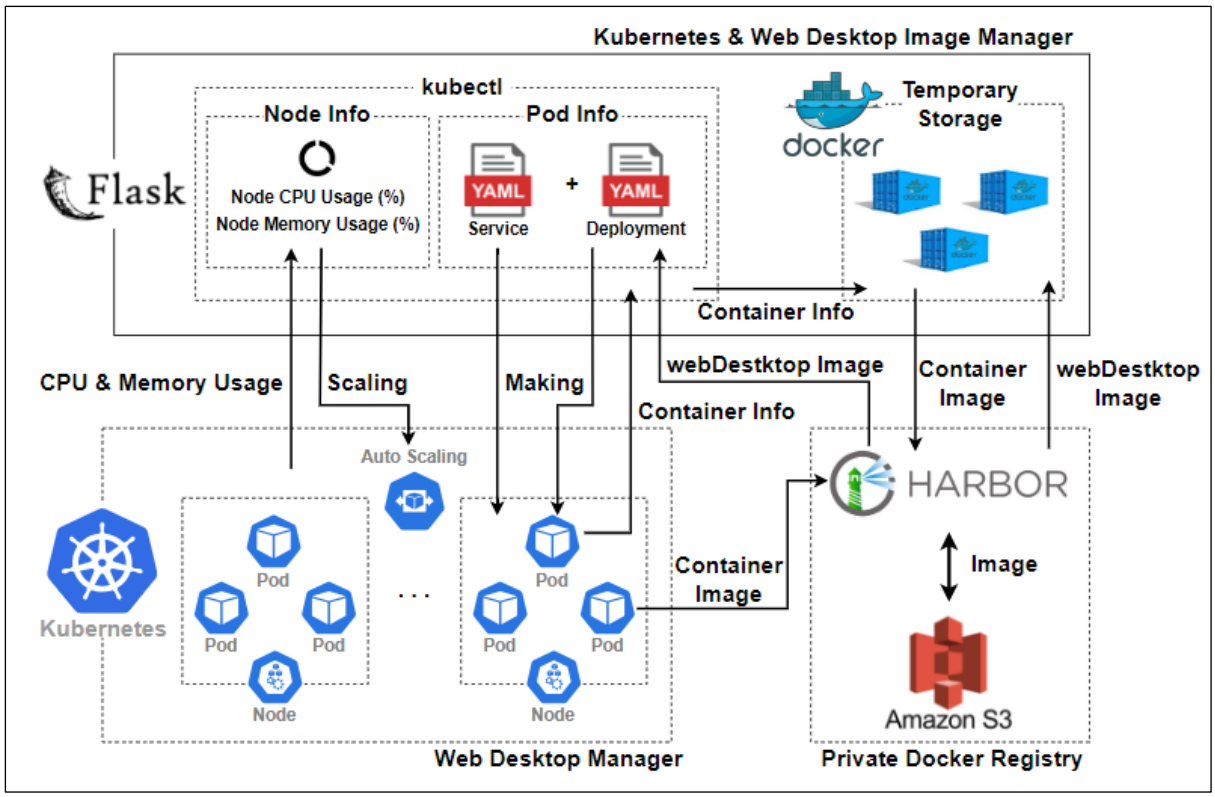

그림 21. 가상환경 서버 설계 구조

가상환경 서비스는 웹 서버의 요청에 따라 가상환경 생성, 실행, 접속, 삭제 등의 기능을 제공한다. 웹 서버, 즉 사용자로부터 api를 통해 요청이 오면, kubernetes의 명령어를 통해 가상환경을 생성, 삭제한다. 그리고 사용자의 docker image를 안전하게 저장한다.

가상환경 서버는 다음과 같이 구성된다.

- **Kubernetes & Web Desktop Image Manager (flask server)** : 사용자의 요청을 바탕으로 Kubernetes에 명령을 내리고, docker images를 관리하는 flask server (이하 flask 서버)
- **Web Desktop Manager (Kubernetes Server)** : 클라우드 환경에서 가상환경(kasm container)을 관리하기 위한 Kubernetes server (이하 k8s 서버)
- **Private Docker Registry (Harbor)** : Docker Image를 저장하는 private 저장소 (이하 harbor)

3.3.2. Kubernetes & Web Desktop Image Manager (flask server)

3.3.2.1. 세부 설명

Flask 기반으로 실행되며, Kubernetes 관리 및 Web Desktop Image를 관리한다. 세부 역할은 다음과 같다.

- **Kubernetes Node & Pod 관리**
- **Kubernetes의 Scaling에 따른 외부 접속 관리**
- **Web Docker image 관리**

3.3.2.2. Kubernetes Node & Pod 관리

Kubernetes 명령어(kubectl)를 이용해 node와 pod 생성, 삭제 등의 관리를 한다.

3.3.2.3. Kubernetes의 Scaling에 따른 외부 접속 관리

외부에서 pod(container)에 접근하기 위해서, 동적으로 변하는 pod의 ip를 직접 노출하지 않고, public node ip를 이용해 '[public node ip]:[pod의 port]'로 접근한다.

auto scaling이 진행되면, 새로운 노드가 생성되어 변경된 pod의 node ip를 알아야 한다. kubectl 명령어를 통해 전체 node의 목록('nodeName:nodeIp')과 pod의 목록('podName:nodeName')을 딕셔너리(Dictionary) 형태로 저장한다. 이를 통해 특정 pod가 속한 변경된 node의 ip를 찾는다. 아래는 위의 과정을 python과 Kubectl 명령어를 통해 작성한 코드이다.

```
# node 의 이름과 ip 를 추출하는 함수
def extractNodeInfo():
result = os.popen("kubectl get nodes -o wide --kubeconfig
/root/kubeconfig.yml").read()

nodeInfoList = result.split('\n')[1:-1]

for nodeInfo in nodeInfoList:
node = nodeInfo.split()
nodeName, nodeExternalIp = node[0], node[6]
extractNodeInfos[nodeName] = nodeExternalIp

return extractNodeInfos
```

그림 22. Kubernetes Server의 Node의 이름과 ip를 추출하는 함수

추출된 Node의 이름과 ip는 Node의 이름이 key, ip가 value인 Dictionary 형태로 저장되게 된다.

```
# pod 의 이름과 node 의 이름을 추출하는 함수
def extractPodInfo():

    result = os.popen("kubectl get pods -o wide --kubeconfig
/root/kubeconfig.yml").read()

    podInfoList = result.split('\n')[1:-1]

    for podInfo in podInfoList:
        pod = podInfo.split()
        podName, nodeName = pod[0], pod[6]
        extractPodInfos[podName] = nodeName

    return extractPodInfos
```

그림 23. Pod의 이름과 Pod가 속한 Node의 이름을 추출하는 함수

추출된 Pod의 이름과 Pod가 속한 Node의 이름도 Pod의 이름이 key, Node의 이름이 value인 Dictionary 형태로 저장된다.

저장된 Node와 Pod의 정보를 비교하면서, Pod가 속한 Node에 해당하는 ip를 추출하게 된다.

```
# pod 의 external ip 를 알기 위한 함수
def extractNodeIpOfPod(nodeList):

    podList = extractPodInfo()

    for _, nodeName in podList.items():
        if nodeName in nodeList:
            return extractNodeInfos[nodeName]

    # Node 의 external ip 를 찾지 못한 경우
    return "Not Found"
```

그림 24. Pod가 할당된 Node의 External IP를 찾는 함수

3.3.2.4. Web Desktop Image 관리

사용자가 생성한 가상환경에 대해 해당 Web Desktop image를 임시 보관 및 관리한다. 또한 k8s 서버에 장애가 발생하면 flask 서버가 대체해 docker 명령어로 Web Desktop image를 생성, 로드, 실행, 삭제 등의 관리할 수 있다.

3.3.2.5. Flask server 구축 과정

Flask 프로젝트를 배포하기 위해 ncloud에서 서버를 생성한다.

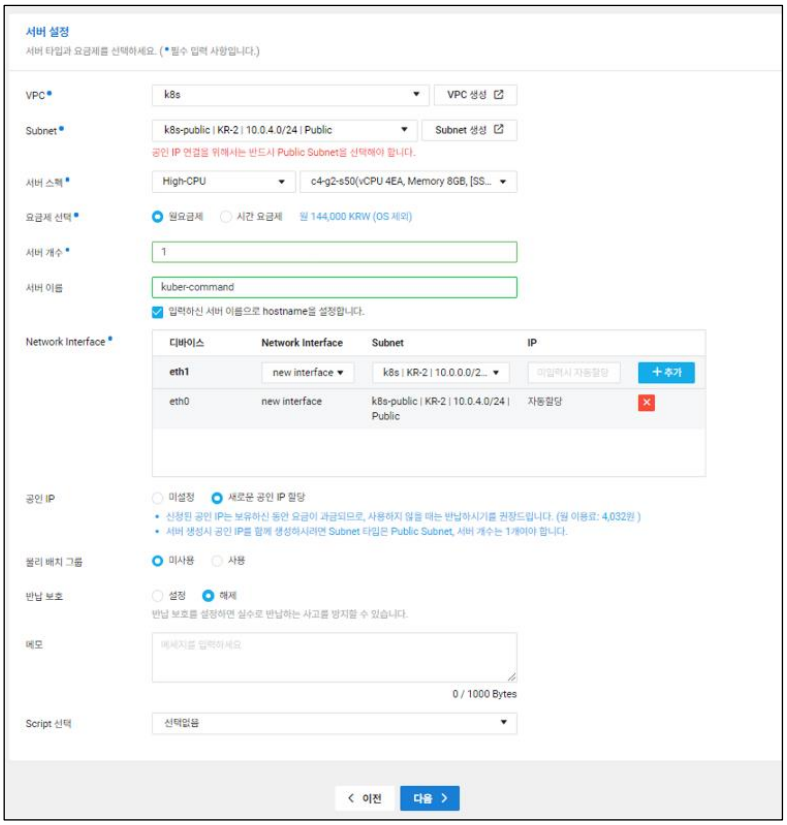

그림 25. Flask Server 설정

Docker와 가상환경의 중요 정보(container id, docker image 경로 등)를 암호화하기 위한 pycrypto 모듈을 설치한다. 또한 k8s 서버로 접속해 명령어를 실행하기 위해 Kubernetes의 명령 툴인 kubectl을 설치한다.

```
root@DESKTOP-16341DF:~# kubectl get namespaces --kubeconfig kubeconfig.yml
NAME              STATUS   AGE
default           Active   102m
kube-node-lease   Active   102m
kube-public       Active   102m
kube-system       Active   102m
```

그림 26. kubectl 명령어 테스트 - namespace 정보 확인

kubectl 명령어를 통해 Kubernetes Server와 통신이 잘 이루어지는 것을 확인하면, python 프로젝트를 실행해 flask 서버를 실행한다.

3.3.3. Web Desktop Manager (Kubernetes Server)

3.3.3.1. 세부 설명

클라우드 환경에서 가상환경(docker container)를 실행하기 위해 container 오케스트레이션 플랫폼인 Kubernetes를 사용한다. 가상환경은 Kubernetes에서 단일 container pod 구조(하나의 pod는 docker container 하나로 구성)로 관리한다. 세부 역할은 다음과 같다.

- **Pod 실행 및 관리**
- **Auto Scaling 지원**

3.3.3.2. Pod 실행 및 관리

Flask를 사용하여 Kubernetes Pod내에서 Web Docker Container를 실행하도록 명령을 실행한다. 즉 사용자의 가상환경이 할당된 상태이다.

3.3.3.3. Auto Scaling

Auto Scaling이란 클라우드에서 서비스의 리소스를 동적으로 조절하는 기술이다. 이는 예상치 못한 트래픽에 대비할 수 있고, 효율적으로 리소스를 관리할 수 있다.

Kubernetes에서는 리소스 사용량에 따라 자동으로 Scaling을 진행한다.

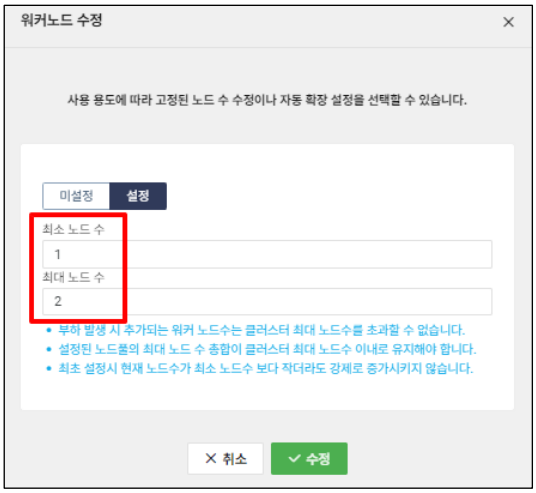

그림 27. Auto Scaling을 위한 워커 노드 설정

Ncloud에서는 Scaling 시 생성되는 최대 node의 수와 최소 node 수를 설정할 수 있다. 현재 본 과제에서는 최대 2개의 node가 생성될 수 있게 하였다.

Auto Scaling이 정상 작동 여부를 확인하기 위해 k8s Node에 여러 개의 Pod를 생성하는 테스트 과정을 진행한다. 테스트 시간 단축을 위해 노드풀의 스펙을 vCPU 4EA, Memory 8GB에서 vCPU 2EA, Memory 4GB로 축소했다.

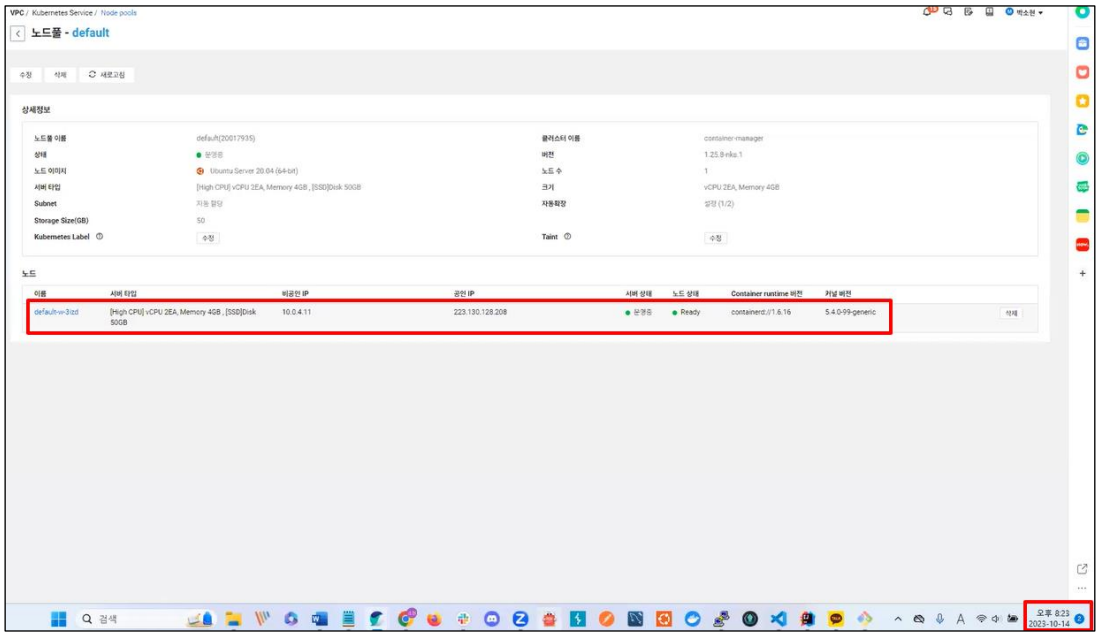

그림 28. 2023/10/14 20:23 - 현재 노드 개수 1개

현재(2023/10/14 20:23 기준) k8s에는 하나의 Node가 있다.

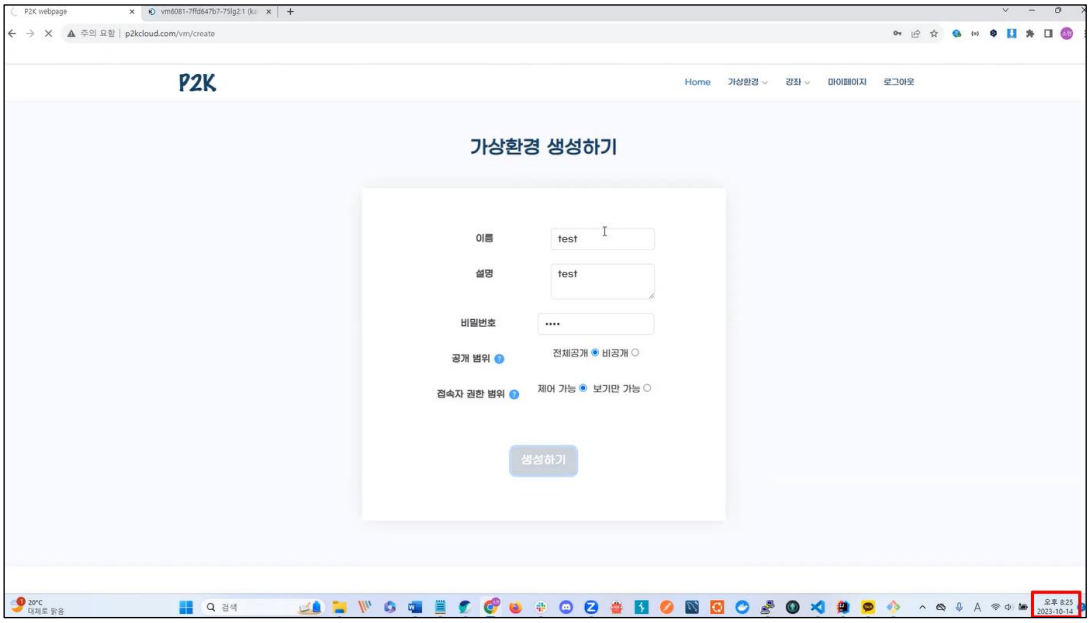

그림 29. 2023/10/14 20:25 - 사용자 가상환경 생성

사용자들이 서비스를 이용하면서 가상환경을 생성한다. (2023/10/14 20:25 기준 - 2번째 가상환경 생성)

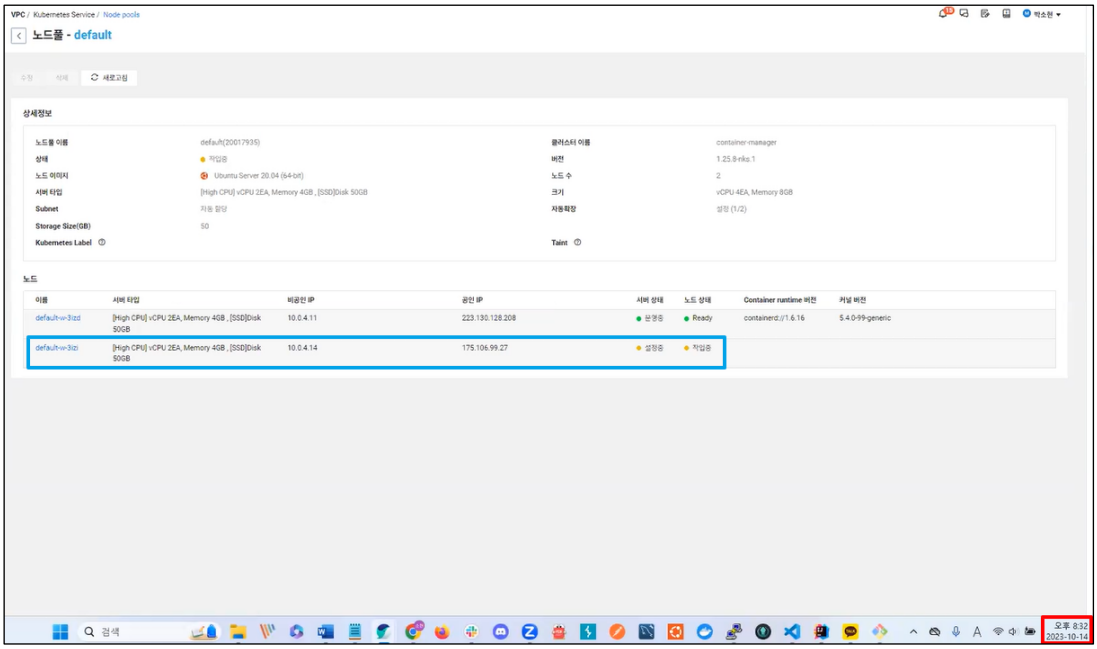

그림 30. 2023/10/14 20:32 - 현재 노드 개수 2개 (1개 생성 중)

여러 Pod가 생성되면 k8s에서 자동으로 Scaling이 진행되는 것을 확인할 수 있다. (2023/10/14 20:32 기준 1개 생성 중)

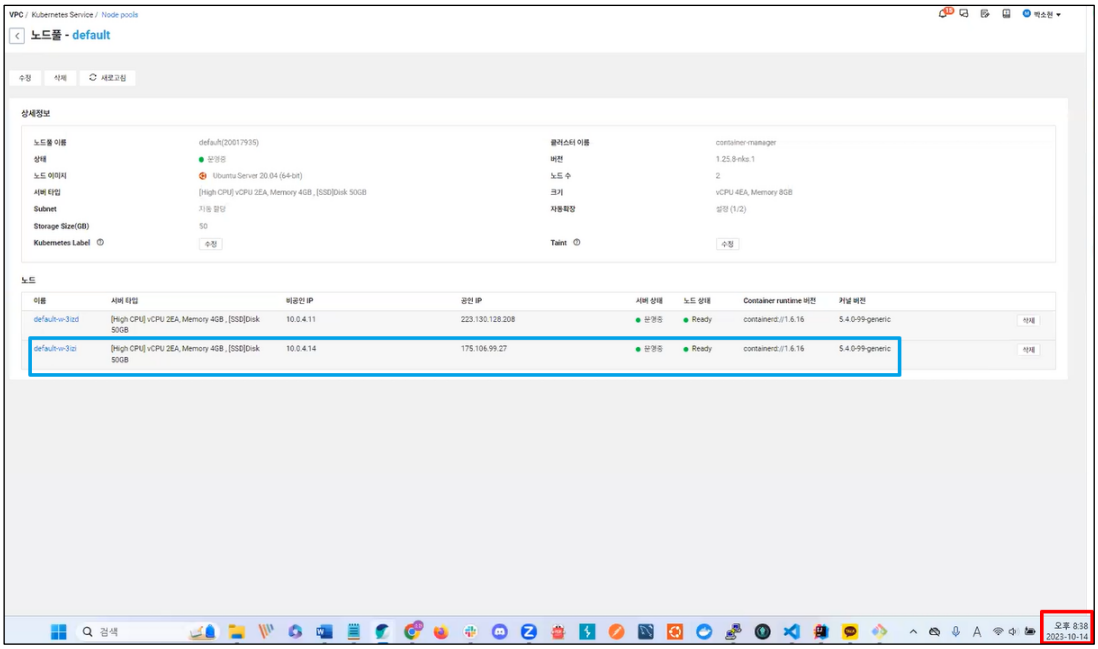

그림 31. 2023/10/14 20:38 - 현재 노드 개수 2개 (1개 생성 완료)

새로운 Node가 생성된 것을 확인할 수 있다. (2023/10/14 20:32 기준 1개 생성 완료)

```
root@kuber-command:~/vm-server# kubectl get pods -o wide
NAME                    READY   STATUS    RESTARTS   AGE     IP             NODE              NOMINATED NODE   READINESS GATES
vm6081-7ffd647b7-751g2  1/1     Running   0          38m     198.18.1.91    default-w-3izd    <none>           <none>
vm6082-64f7cd5fcb-gnbsw 1/1     Running   0          17m     198.18.0.115   default-w-3izi    <none>           <none>
vm6083-545f5df5c7-cgs9p 1/1     Running   0          33m     198.18.1.104   default-w-3izd    <none>           <none>
vm6084-5dfd988854-k8jz9 1/1     Running   0          30m     198.18.1.117   default-w-3izd    <none>           <none>
vm6085-8d4cd94f9-41cfw  1/1     Running   0          27m     198.18.1.34    default-w-3izd    <none>           <none>
vm6086-5675d48d48-w65tl 1/1     Running   0          23m     198.18.1.217   default-w-3izd    <none>           <none>
vm6087-5678768cd9-gmlwb 1/1     Running   0          21m     198.18.1.143   default-w-3izd    <none>           <none>
vm6088-7458b7495d-2phpc 1/1     Running   0          19m     198.18.1.76    default-w-3izd    <none>           <none>
vm6089-8584c784ff-lg2t6 1/1     Running   0          17m     198.18.1.148   default-w-3izd    <none>           <none>
vm6090-77cbc8754f-b9dlv 1/1     Running   0          15m     198.18.0.58    default-w-3izi    <none>           <none>
vm6091-5454d847b4-zmqt2 1/1     Running   0          13m     198.18.0.133   default-w-3izi    <none>           <none>
vm6092-8567cb7c7f-854tq 1/1     Running   0          11m     198.18.0.186   default-w-3izi    <none>           <none>
vm6093-574b88744b-s42f7 1/1     Running   0          10m     198.18.0.65    default-w-3izi    <none>           <none>
vm6095-65f45f87dd-gpmqs 1/1     Running   0          8m57s   198.18.0.63    default-w-3izi    <none>           <none>
vm6097-77f84ccf74-lsdw2 1/1     Running   0          6m57s   198.18.0.178   default-w-3izi    <none>           <none>
root@kuber-command:~/vm-server#
```

그림 32. Pod들의 할당 Node 확인

'Kubectl get pods -o wide' 명령어를 통해 생성된 각 Pod들이 어느 Node에 위치하는지 확인할 수 있다. (2023/10/14 20:58 기준 - 2번째 파드 생성 시각을 기준으로 33분 뒤)

3.3.3.4. Kubernetes Server 구축 과정

ncloud에서 서버를 생성해 Kubernetes 서버를 구축한다.

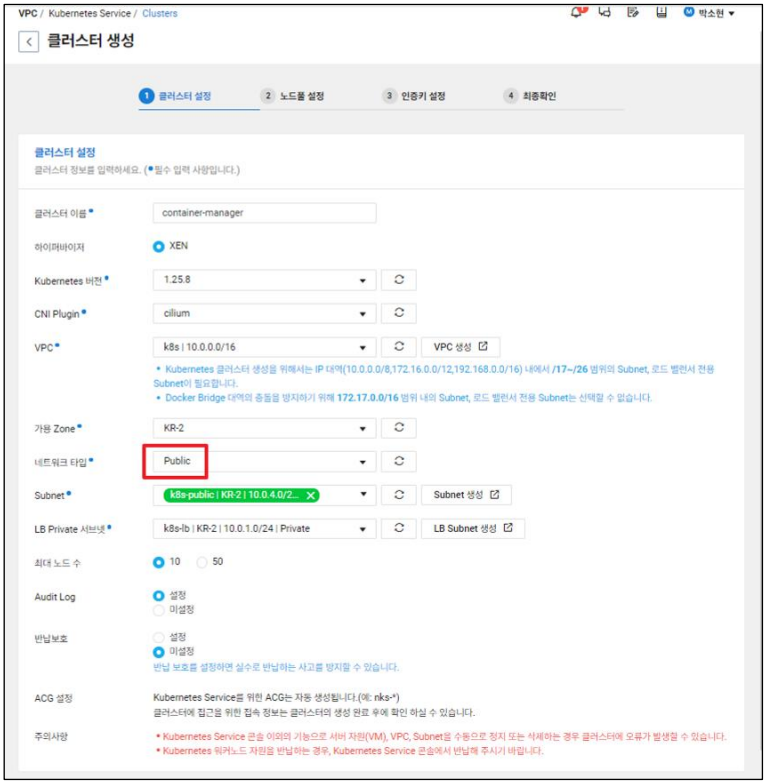

그림 33. Kubernetes Server의 Cluster 설정

Cluster 설정은 Kubernetes Server의 k8s 버전, 네트워크 설정이다. k8s의 Node에 접근하기 위해 네트워크 타입을 public으로 설정하여 public Node ip를 할당한다. 노드풀 설정은 Node의 스펙을 결정하는 설정으로, 현재 Node의 스펙은 vCPU 4EA, 8GB이다.

그림 34. Kubernetes Server의 NAT Gateway 설정

NAT gateway와 NAT gateway에 배치할 subnet을 생성 후 Route 설정을 통해 k8s 서버에 접속을 위한 네트워크 환경을 구성한다.

3.3.4. Private Docker Registry

3.3.4.1. Private Docker Registry 구성도

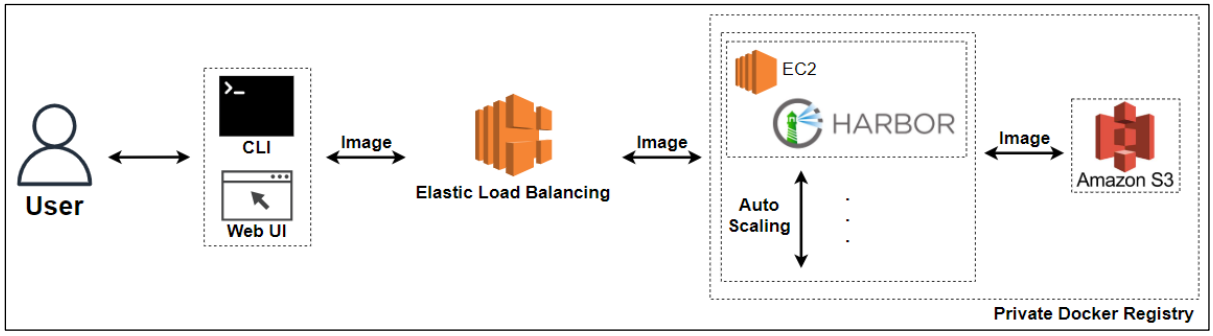

그림 35. Private Docker Registry 구성도

오픈 소스 컨테이너 레지스트리인 Harbor와 AWS 클라우드 스토리지 서비스인 S3를 결합하여 구축한다. Harbor는 AWS의 EC2 인스턴스 위에서 Multi Container 형태로 실행되며, 이를 통해 Docker Image를 private하게 저장할 수 있다. 사용자의 docker image 관리는 민감한 문제이므로 트래픽 분산과 서비스 무중단을 위해, Auto Scaling Group으로 구성하고 Load Balancer를 연동한다. Harbor와 S3와의 연동은 Harbor 데이터의 복제 및 백업을 지원하여 시스템 장애에 대비할 수 있다. 사용자는 CLI와 Web UI로 접근할 수 있다.

3.3.4.2. Private Docker Registry 구축 과정

AWS의 Route 53에서 domain(p2kcloud.com) 발급하고 Certificate Manager에서 domain(p2kcloud.com)을 바탕으로 한 SSL 인증서 발급한다. 그리고 Load Balancer를 생성해, 해당 Load Balancer를 domain(registry.p2kcloud.com)과 연동한다.

Harbor를 구축하고, S3를 생성해 서로 연동하고, Harbor와 Load Balancer도 연동한다.

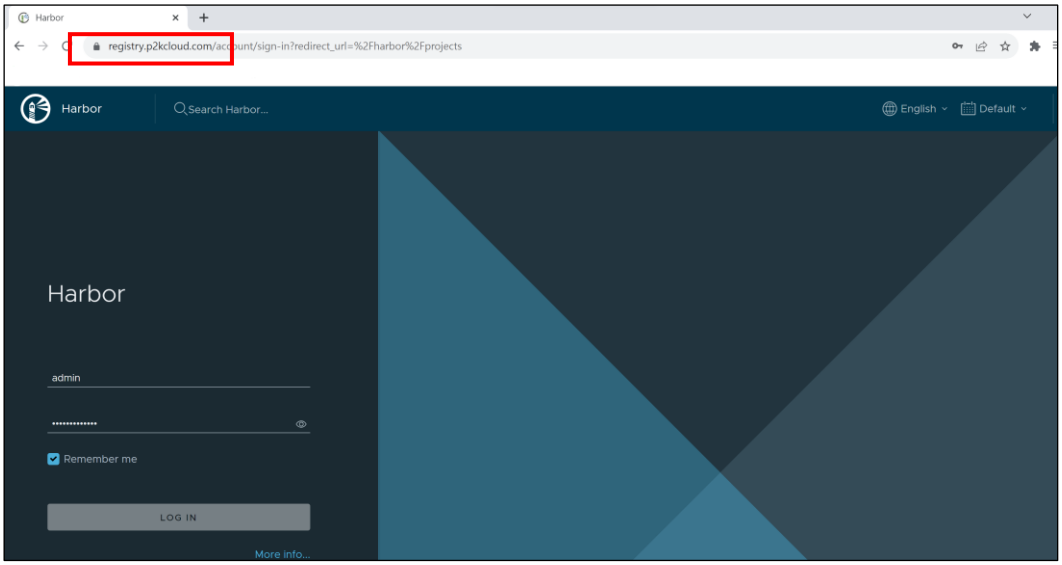

그림 36. Private Docker Registry (registry.p2kcloud.com) 화면

registry.p2kcloud.com에 접속하면 구축한 private docker registry인 Harbor에 접속할 수 있다.

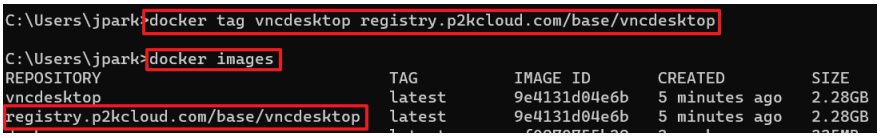

그림 37. Docker Image Tagging 작업

Harbor에 이미지 업로드(push)가 제대로 동작하는지 확인하기 위해 우선 Harbor에 저장될 project를 생성한다. 그리고 해당 경로로 push 하기 위해 tagging으로 이미지의 이름을 경로를 지정한다.

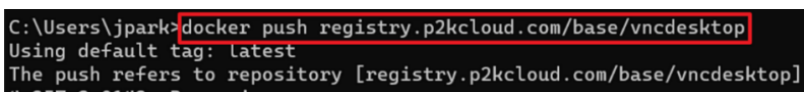

그림 38. Tagging 작업이 된 Docker Image를 Private Docker Registry에 push하기

Harbor_address/project명/push할 docker 이미지 로 tagging을 진행하면, 해당 경로에 맞게 docker 이미지가 push된다.

[Image Push 결과]

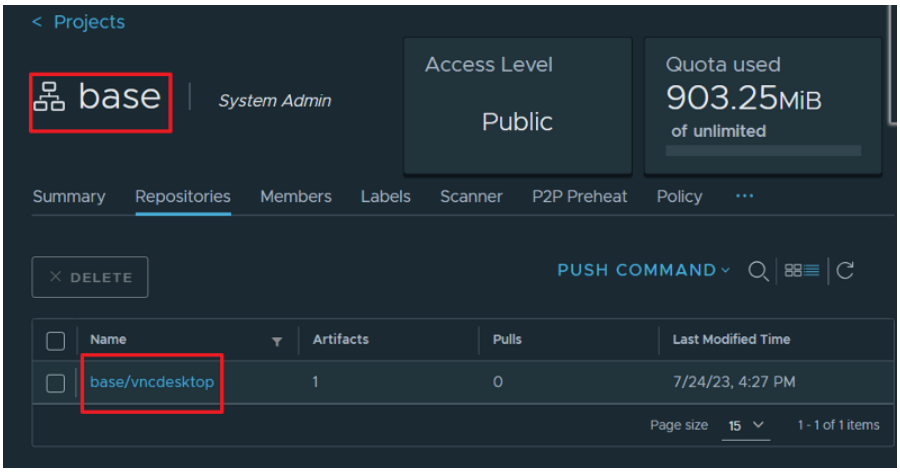

그림 39. Tagging한 Docker Image push 결과

Tagging된 경로에 맞게 Image가 저장된 모습이다.

```
C:\Users\jpark>docker pull registry.p2kcloud.com/base/vncdesktop
Using default tag: latest
latest: Pulling from base/vncdesktop
Digest: sha256:4dfb0e71ca6f55d8c44b7546164949706d953e32bce1a4b949abaa807e68dfa1
Status: Downloaded newer image for registry.p2kcloud.com/base/vncdesktop:latest
registry.p2kcloud.com/base/vncdesktop:latest

C:\Users\jpark>docker images
REPOSITORY                                TAG          IMAGE ID       CREATED          SIZE
vncdesktop                                latest       9e4131d04e6b   16 minutes ago   2.28GB
registry.p2kcloud.com/base/vncdesktop     latest       9e4131d04e6b   16 minutes ago   2.28GB
docker                                    latest       cf0970755b29   2 weeks ago      335MB
mysql                                     latest       041315a16183   2 weeks ago      565MB
ubuntu_novnc_port                         latest       563beef5a7ac   3 weeks ago      1.11GB
ubuntu                                    22.04        5a81c4b8502e   3 weeks ago      77.8MB
theasp/novnc                              latest       1f3e73f5b0da   3 weeks ago      608MB
mariadb                                   latest       2cd1259c6b0b   3 weeks ago      403MB
kakao-shop-front-frontend                 latest       ed51cfa3f9cd   3 weeks ago      575MB
kakao-shop-front-backend                  latest       86d8cb01afa4   3 weeks ago      273MB
ubuntu                                    latest       99284ca6cea0   6 weeks ago      77.8MB
kasmweb/desktop                           1.7.0-edge   249937e8107e   6 months ago     2.18GB
```

그림 40. Private Docker Registry에 저장된 Docker Image pull 작업

또한 pull을 통해 Harbor에서 docker 이미지를 가져올 수 있다.

이와 같이 push와 pull을 통해 Harbor로 이미지를 관리하여 보안성을 강화한다.

3.3.5. Kasm Docker Container

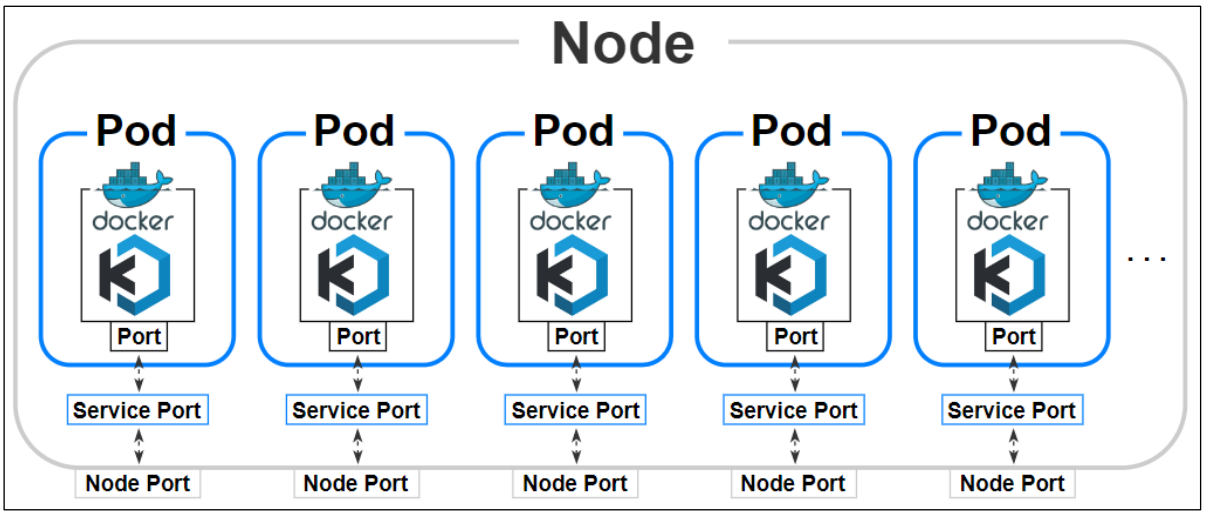

그림 41. Kasm Docker Container 구조도

사용자마다 가상환경을 할당하기 위해 Docker container를 사용한다. 가상머신(VM)이 아닌 Container로 가상환경을 할당하는 이유는 가상머신은 많은 저장 공간 등 컴퓨팅 자원을 차지하고, 오버헤드가 많이 일어나 이런 단점을 보완하기 위해 container를 사용한다. 이를 위해 Kasm docker container를 사용하는데, 이는 Desktop, 애플리케이션 및 웹 서비스에 대한 브라우저 기반 액세스를 제공하는 Docker Container Streaming Platform이다. CDI(Containerized Desktop Infrastructure) 기술을 통해 웹 브라우저로 액세스할 수 있는 주문형 일회용 docker container를 생성한다.

3.3.6. 가상환경 상태와 제공하는 기능 리스트

사용자는 가상환경에 대해 여러 정보를 설정할 수 있다.

정보	설명
이름	가상환경의 이름
설명	가상환경에 대한 설명
비밀번호	가상환경 접속을 위한 비밀번호
공개 범위	가상환경을 공개하는 범위 - **전체 공개** : 이름으로 검색이 가능하며, 모든 사람이 접속 가능 - **비공개** : 검색 불가능하며, 자신만 접속 가능

접속자 권한 범위	가상환경을 전체 공개로 설정했을 때, 접속자의 권한 범위 - **제어 가능** : 접속자도 화면을 제어 가능 - **보기만 가능** : 제어 권한 없이 관전만 가능 ** 접속자는 이름: guest, 비밀번호: guest로 접속한다.*
연결된 강좌	가상환경과 강좌를 연결하는 것으로, 강좌와 연결하면 해당 강좌에 가상환경이 속해 학습 관리를 할 수 있다.

가상환경은 중지, 실행중 2가지 상태로 나뉜다. 각 상태에 따라 여러 기능을 제공한다.

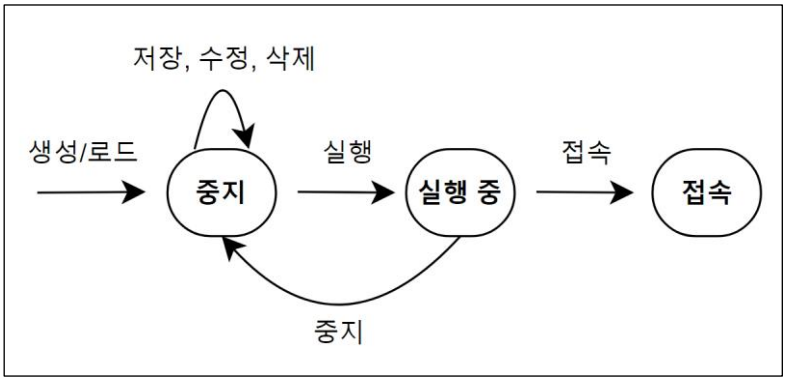

그림 42. 가상환경 상태에 따른 기능

상태	제공하는 기능	설명
중지	**실행하기**	가상환경을 실행한다.
	저장하기	가상환경을 private registry인 harbor에 저장한다.
	내보내기	가상환경의 로드 key를 가져온다. 로드 key가 있으면 동일한 가상환경을 생성할 수 있다.
	수정하기	가상환경 관련 정보(이름, 설명, 비밀번호, 공개 범위, 접속자 권한 범위, 연결된 강좌)를 수정한다.
	삭제하기	가상환경을 삭제한다.
실행 중	**접속하기**	실행 중인 가상환경에 접속한다.
	중지하기	실행 중인 가상환경을 중지한다. 중지된 상태는 접속이 불가능하다.
	생성하기	가상환경을 생성한다.
	로드하기	로드 key로 동일한 가상환경을 생성한다.

그림 43. 가상환경 상태에 따른 정보와 기능 (좌 – 중지/실행에 따른 가상환경 정보, 우 – 중지 상태일 때 사용가능 기능)

위의 사진은 웹 페이지에서 가상환경에 대한 정보와 기능을 제공하는 모습이다. 지금부터 각 기능들의 구체적인 동작 과정과 구현 방법에 대해 설명한다.

3.3.7. 가상환경 접속 범위와 권한 관리

Kasm container는 실행할 때마다 container 내부 vnc_startup.sh를 실행한다. vnc_startup.sh 파일을 분석해 kasm container 동작 과정을 살핀다.

```
# switch passwords to local variables
tmpval=$VNC_VIEW_ONLY_PW
unset VNC_VIEW_ONLY_PW
VNC_VIEW_ONLY_PW=$tmpval
tmpval=$VNC_PW
unset VNC_PW
VNC_PW=$tmpval
```

그림 44. Kasm 접속 비밀번호 설정 관련 코드

```
# first entry is control, second is view (if only one is valid for both)
mkdir -p "$HOME/.vnc"
PASSWD_PATH="$HOME/.kasmpasswd"
if [[ -f $PASSWD_PATH ]]; then
    echo -e "\n---------  purging existing VNC password settings  ---------"
    rm -f $PASSWD_PATH
fi
VNC_PW_HASH=$(python3 -c "import crypt; print(crypt.crypt('${VNC_PW}', '\$5\$kasm\$'));")
VNC_VIEW_PW_HASH=$(python3 -c "import crypt; print(crypt.crypt('${VNC_VIEW_ONLY_PW}', '\$5\$kasm\$'));")
echo "kasm_user:${VNC_PW_HASH}:ow" > $PASSWD_PATH
echo "kasm_viewer:${VNC_VIEW_PW_HASH}:" >> $PASSWD_PATH
chmod 600 $PASSWD_PATH
```

그림 45. Kasm 접속 비밀번호 암호화와 권한 설정 관련 코드

Kasm container 접속에 대해 kasm_user와 kasm_viewer가 존재하고, 비밀번호를 암호화해 .kasmpasswd 파일에 저장하는 것을 확인했다.

그림 46. .kasmpasswd 파일 안 비밀번호, 권한 관리

그리고 공식 문서를 확인한 결과, 암호화된 비밀번호 뒤에 :o (관리자), :w (쓰기권한), :r (읽기권한)을 추가해 사용자들의 권한을 설정할 수 있다.

위의 내용을 바탕으로 가상환경(kasm container) 접근에 대해 사용자(user)와 접속자(guest)를 나눠 관리한다. 접속하는 사용자의 이름과 비밀번호 설정 위해 스크립트 파일 start.sh를 작성한다. start.sh는 사용자가 가상환경에 설정한 공개 범위(scope)와 접속자 제어 권한(control), 가상환경 비밀번호를 매개변수로 받는다. Scope가 true이면 공개이므로 control(접속권한)에 따라 guest의 권한을 w(제어 가능) 또는 r(보기만 가능)로 설정한다. 그리고 guest의 비밀번호는 guest를 암호화하고 user의 비밀번호는 매개변수로 받은 가상환경 비밀번호를 암호화해 .kasmpasswd 에 입력한다.

```
#!/bin/sh
#!/bin/bash

# scope: 1, control: 2, password: 3

PASSWD_PATH="$HOME/.kasmpasswd"
VNC_VIEW_ONLY_PW="guest"

VNC_PW_HASH=$(python3 -c "import crypt; print(crypt.crypt('$3', '\$5\$kasm\$'));")
VNC_VIEW_PW_HASH=$(python3 -c "import crypt;
print(crypt.crypt('${VNC_VIEW_ONLY_PW}', '\$5\$kasm\$'));")

sed -i "1 i user:${VNC_PW_HASH}:ow" ~/.kasmpasswd

if [ $1 = "True" ]; then

if [ $2 = "True" ]; then
sed -i "2 i guest:${VNC_VIEW_PW_HASH}:w" ~/.kasmpasswd
fi
if [ $2 = "False" ]; then
sed -i "2 i guest:${VNC_VIEW_PW_HASH}:r" ~/.kasmpasswd

fi
fi

sed -i "3,4d" ~/.kasmpasswd
```

그림 47. 접속자 권한에 따른 접속 제어 코드

즉 Kasm container 실행 시 내부에서 start.sh를 실행해 kasm container의 .kasmpasswd를 조작하여 가상환경 접속 정보를 관리한다.

3.3.8. 가상환경 목록 조회

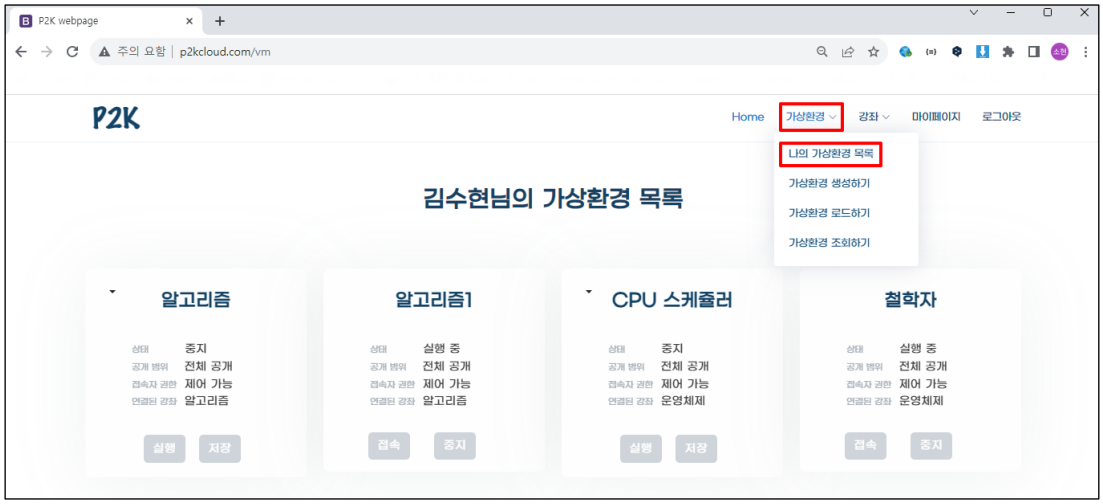

그림 48. 사용자 - 가상환경 목록

사용자는 [가상환경] – [나의 가상환경 목록]에서 자신의 가상환경 목록을 조회한다.

3.3.9. 가상환경 상태에 따른 기능 제공

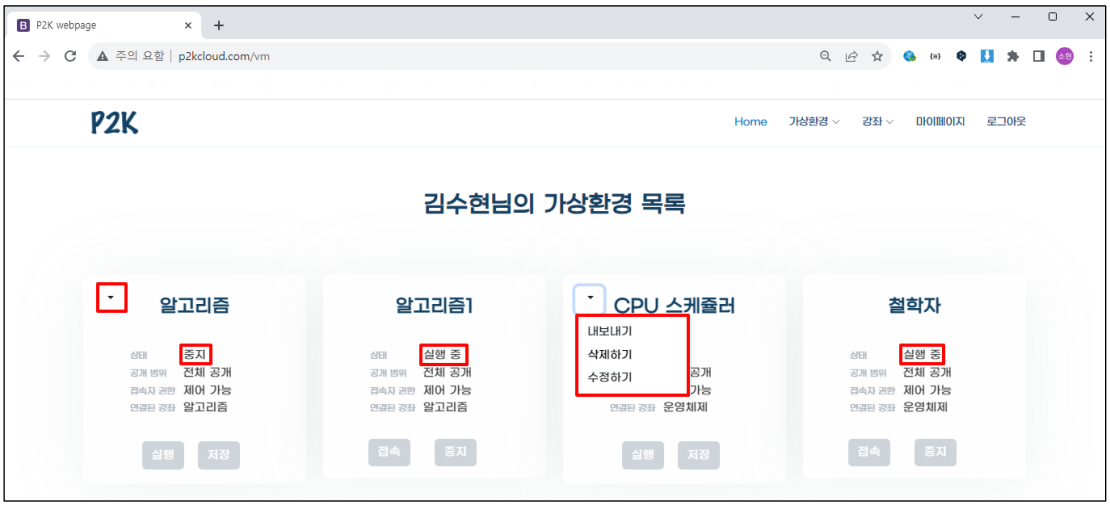

그림 49. 사용자 - 가상환경 상태에 따른 기능 제공

사용자는 가상환경 목록을 통해 가상환경에 대한 정보를 제공받을 수 있다. 사용자는 가상환경 상태에 따라 다른 기능을 제공받는다.

3.3.10. 가상환경 생성하기

사용자는 웹 페이지에서 가상환경 생성을 위한 정보를 입력 후 가상환경을 생성한다.

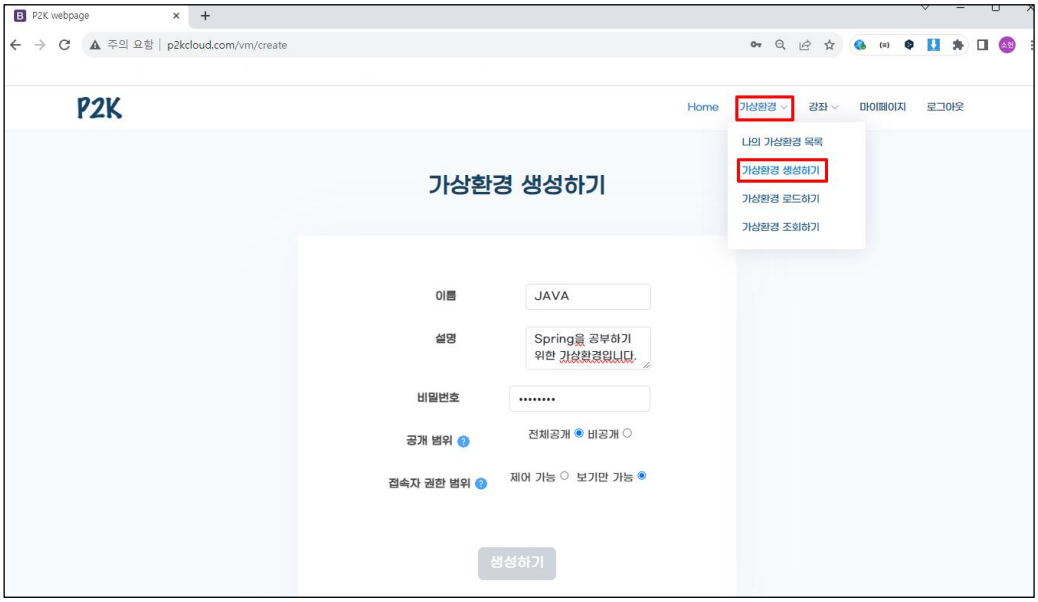

그림 50. 사용자 - 가상환경 생성하기

사용자는 [메인 페이지] – [가상환경 생성하기] 또는 [가상환경] – [가상환경 생성하기]에서 가상환경 생성을 위한 정보를 입력 후 가상환경을 생성한다.

사용자의 가상환경은 외부 접속이 가능해야 한다. 따라서 사용자에게 할당된 Pod는 외부로 노출하기 위해, Deployment, Service 2가지의 Pod를 생성한다. 생성 요청을 받은 flask 서버는 harbor에 존재하는 기본 kasm base image를 가져와(pull) deployment pod를 생성하기 위한 deployment.yaml 파일을 생성한다. 그리고 pod를 노출시키기 위한 service.yaml 파일을 생성한다. 이때 외부에서의 접속을 위해 yaml 파일에서는 3가지 port를 정의한다.

- **node port** : 외부에서 사용자가 Kubernetes node로 접속하기 위한 port
- **port** : cluster 안에서 내부적으로 service 객체에 접근하기 위한 port
- **target port** : service 객체로 전달된 요청을 pod로 전달할 때 사용하는 port

가상환경마다 고유하게 가지는 node port는 30000~32768 사이의 포트 번호를 차례대로 할당하고, port는 6080 이후의 숫자를 차례대로 할당하고, target port는 kasm container의 port 번호인 6091으로 지정한다. 즉, nodePort, port, targetport를 설정해 외부(사용자) - pod - kasm container 로 연결한다.

다음은 Pod를 만들기 위한 yaml 파일 작성 함수이다.

```
# deployment pod 만드는 함수
def generateDeploymentPodYaml(deploymentName, containerName, imageName,
servicePort) :
    deploymentDefinition = {
        "apiVersion": "apps/v1",
        "kind": "Deployment",
        "metadata": {"name": deploymentName},
        "spec": {
            "replicas": 1,
            "selector": {
                "matchLabels": {
                    "app": "webdesktop", # service 에서 pod 를 선택할 때 구별하는 용도
                    "port": str(servicePort) # port label 추가
                }
            },
            "template": {
                "metadata": {
                    "labels": {
                        "app": "webdesktop", # 새로운 pod 가 생성될 때 template 정의
                        "port": str(servicePort) # port label 추가
                    }
                },
                "spec": {
                    "containers": [
                        {
                            "name": containerName,
                            "image": imageName,
                            "ports": [{"containerPort": 6901}], # container 포트는
이미지 받을 때부터 열려있었던 포트인 6901 로 접속해야 가능
                        }
                    ],
                    "imagePullSecrets": [{"name": "harbor"}] # harbor 라는 이름의
kubeconfig.yaml 파일
                }
            }
        }
    }
    # YAML 로 변환하여 문자열로 반환합니다.
    deploymentYaml = yaml.dump(deploymentDefinition, default_flow_style=False)
    return deploymentYaml
```

그림 51. Deployment Pod를 만들기 위한 yaml 작성 함수

```
# service pod 를 만드는 함수 (pod 를 외부로 노출하기 위함)
def generateServiceYaml(serviceName, servicePort, nodePort):
    # Service 의 기본 구조를 딕셔너리로 정의합니다.
    serviceDefinition = {
        "apiVersion": "v1",
        "kind": "Service",
        "metadata": {"name": serviceName},
        "spec": {
            "type": "NodePort",
            "selector": {
                "app": "webdesktop",
                "port": str(servicePort) # port label 추가
            },
            "ports": [
                {
                    "port": int(servicePort),
                    "targetPort": 6901, # deployment 의 containerPort 와 일치해야 함
                    "nodePort": int(nodePort) # node_port 는 30000~32768
                }
            ]
        }
    }

    # YAML 로 변환하여 문자열로 반환합니다.
    serviceYaml = yaml.dump(serviceDefinition, default_flow_style=False)
```

그림 52. Service Pod를 만들기 위한 yaml 작성 함수

또한 가상환경과 관련된 보안상 노출되면 안 되는 정보(container id, docker image path 등)는 flask 서버에서 암호화를 진행해 웹 서버로 응답 보낸다.

Flask 서버는 k8s 서버에 pod 생성, 실행, 중지, 그리고 사용 중이던 사용자의 container image 저장을 위한 명령어를 수행한다. 또한, 사용자의 kasm docker image와 container를 임시로 저장하고 관리하는 역할도 수행한다. 이는 Harbor로 docker image를 업로드(push)하기 전 docker image들의 직접적인 관리와 백업을 위한 것이다. 따라서 flask 서버는 동시에 docker 명령어 create와 commit을 사용해 kasm base image에서 container와 image를 생성한다.

다음으로 제공되는 코드는 가상환경 정보를 암호화하는 코드이다.

```
class AESCipher(object):
    def __init__(self, key):
        self.key = hashlib.sha256(key.encode()).digest()

    def encrypt(self, message):
        message = message.encode()
        raw = pad(message)
        cipher = AES.new(self.key, AES.MODE_CBC, self.__iv().encode('utf8'))
        enc = cipher.encrypt(raw)
        return base64.b64encode(enc).decode('utf-8')

    def decrypt(self, enc):
        enc = base64.b64decode(enc)
        cipher = AES.new(self.key, AES.MODE_CBC, self.__iv().encode('utf8'))
        dec = cipher.decrypt(enc)
        return unpad(dec).decode('utf-8')

    def __iv(self):
        return chr(0) * 16
```

그림 53. 가상환경 정보 암호화

3.3.11. 가상환경 내보내기/로드하기

[내보내기]

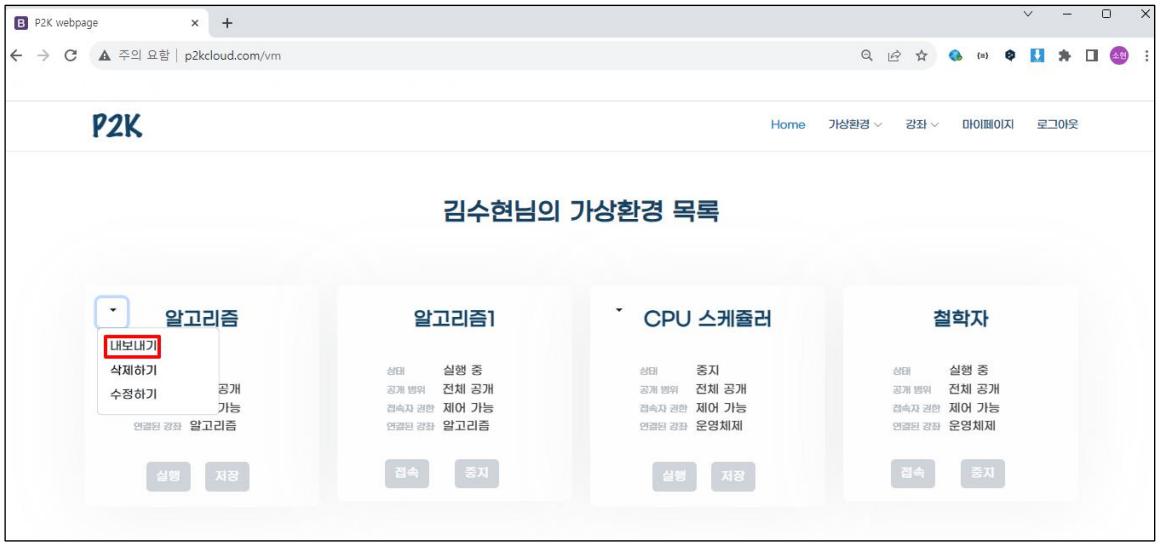

그림 54. 사용자 - 가상환경 내보내기

사용자는 가상환경 내보내기 기능을 통해 자신의 가상환경을 내보낼 수 있다.

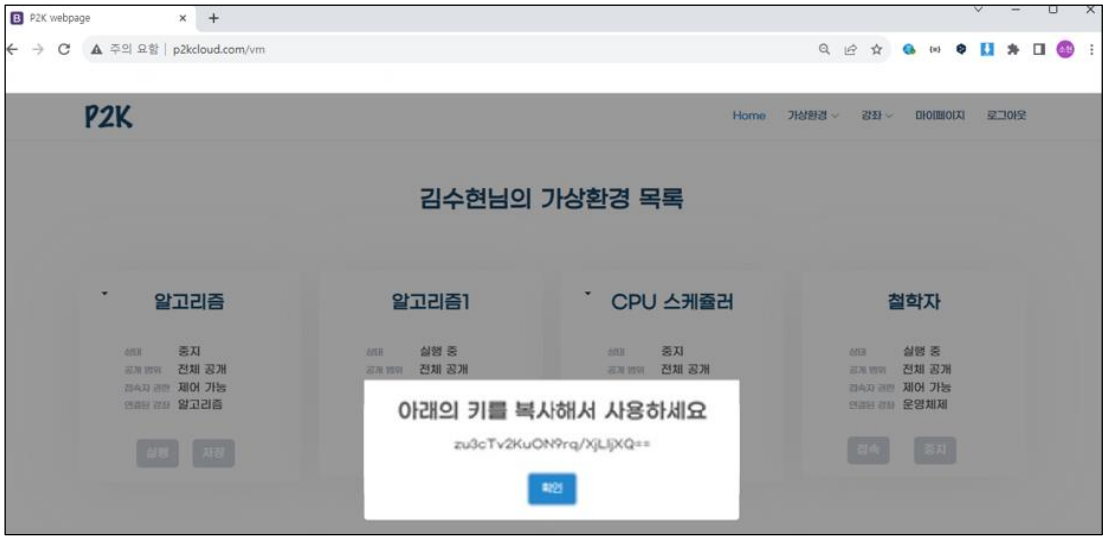

그림 55. 사용자 - 가상환경 내보내기 키 발급

즉 가상환경 내보내기를 클릭하면 key값을 받을 수 있는데, 해당 key로 현재 가상환경과 동일한 가상환경을 생성할 수 있다. 이때 key 값은 harbor에 올라간 kasm docker image의 경로를 암호화한 값이다. 암호화된 key 값은 웹 서버 내 MariaDB에 저장되어 있다. Harbor에 올라간 경로를 통해 docker image를 생성하므로 가상환경을 내보내기 전에 가상환경 저장하기를 해야 한다.

[로드하기]

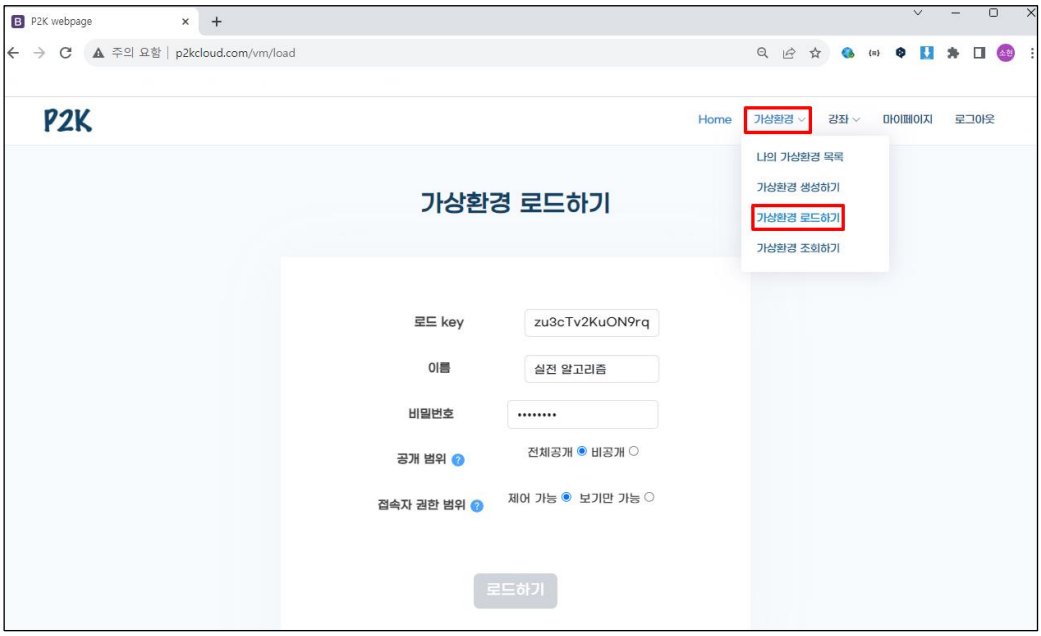

그림 56. 사용자 - 가상환경 로드하기

가상환경 로드하기 기능은 로드 key를 이용해 key와 일치하는 가상환경을 그대로 복사해 동일한 가상환경을 생성하는 기능이다. 여기서 로드 key값은 내보내기를 통해 받은 key값이다. 가상환경 로드 key 값, 비밀번호 등을 입력하고 로드하기를 누르면 /load api를 통해 가상환경 로드하기 요청을 보낸다.

flask 서버에서는 해당 key를 복호화하여 harbor에 존재하는 복사 대상 kasm docker image의 위치를 알아낸다. 찾은 harbor 내 경로를 통해 deployment.yaml 파일과 service.yaml 파일을 생성한다.

그 결과 사용자는 로드 key값과 동일한 가상환경을 생성해 할당 받을 수 있다.

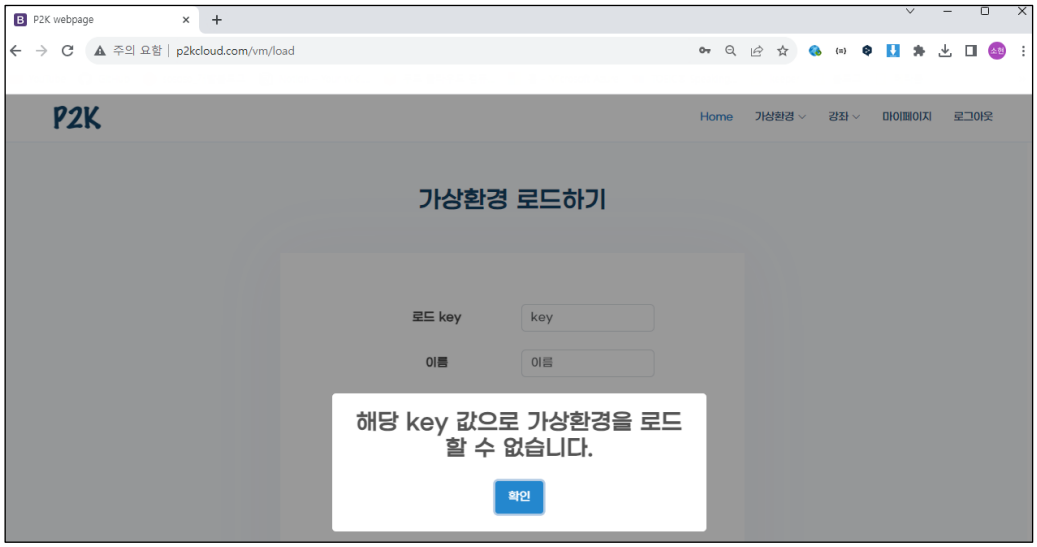

그림 57. 사용자 - 잘못된 로드 key 입력 시 오류

만약 잘못된 로드 key로 가상환경 로드를 시도하면 로드할 수 없다는 경고창이 나온다.

3.3.12. 가상환경 실행하기

그림 58. 실행할 가상환경

중지 상태의 가상환경에서 실행 버튼을 누르면 웹 서버에서 flask 서버로 /start api를 통해 요청을 보낸다.

flask 서버에서는 kubernates 명령 툴인 kubectl의 apply를 이용해 이전에 생성한 deployment.yaml 및 service.yaml 파일로 Deployment Pod와 Service Pod를 생성한다. 이때 가상환경 공개 범위와 접속자의 권한 범위를 설정하기 위해 작성했던 start.sh 스크립트 파일을 적용해야 한다. 이를 위해 먼저 pod의 label를 사용해 조회하고 pod의 이름을 가져온다. Pod 이름을 사용해 flask 서버에 생성해둔 start.sh 파일을 pod 안 container 내부로 복사(copy)한다. 그리고 exec 명령어로 매개변수와 함께 container 내의 start.sh를 실행한다.

이로써 Kasm Container, 즉 가상환경은 k8s 서버에서 실행되어 사용자가 설정한 범위와 접속자 권한으로 설정된 채 외부에서 접속 가능한 상태가 된다.

3.3.13. 가상환경 접속하기

그림 59. 접속할 가상환경

실행 중인 가상환경에 접속 버튼을 눌러 사용자 이름(user)와 비밀번호(사용자가 설정한 가상환경 비밀번호)를 입력해 가상환경에 접속한다.

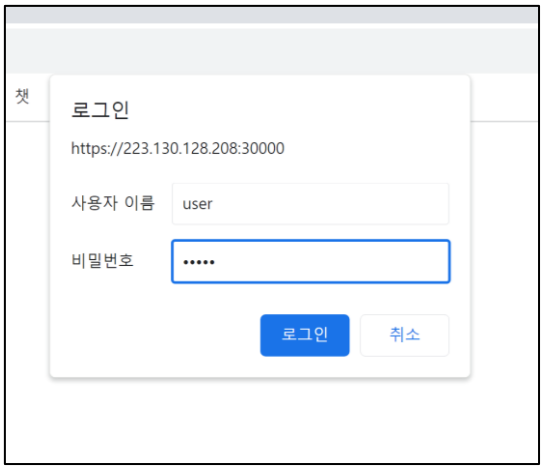

그림 60. 가상환경 접속 시 접속자 정보 입력

사용자는 접속 권한에 따라 사용자 이름과 비밀번호를 다르게 입력한다.

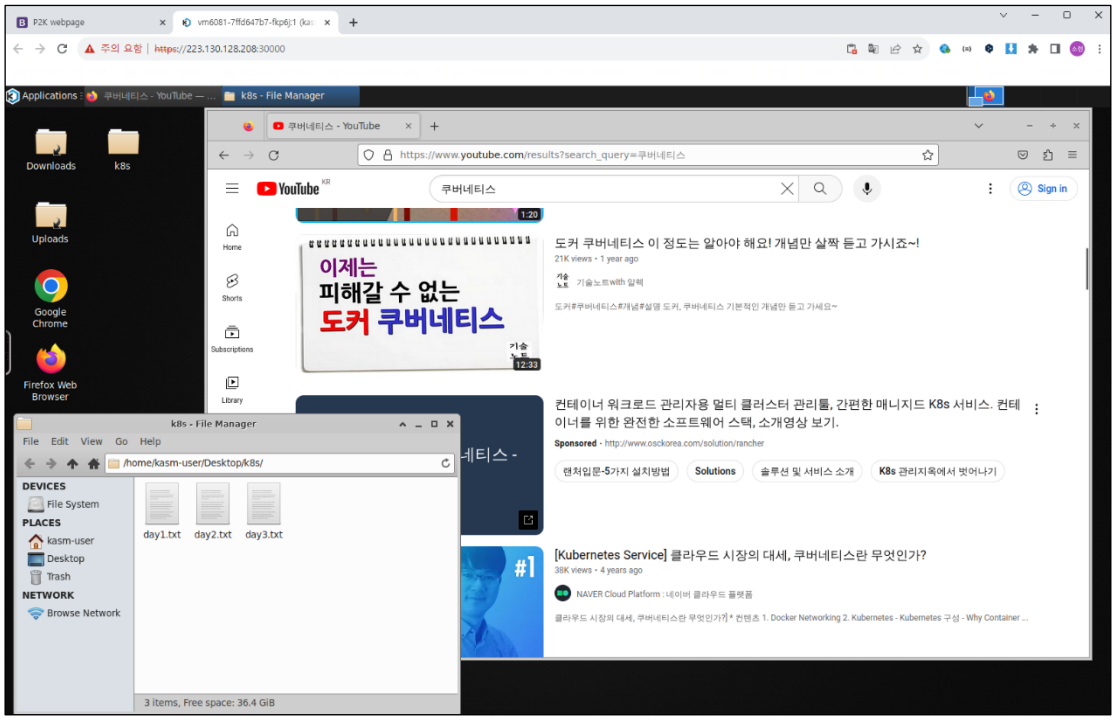

그림 61. 접속한 가상환경의 모습

위의 사진은 해당 가상환경에 접속한 화면이다.

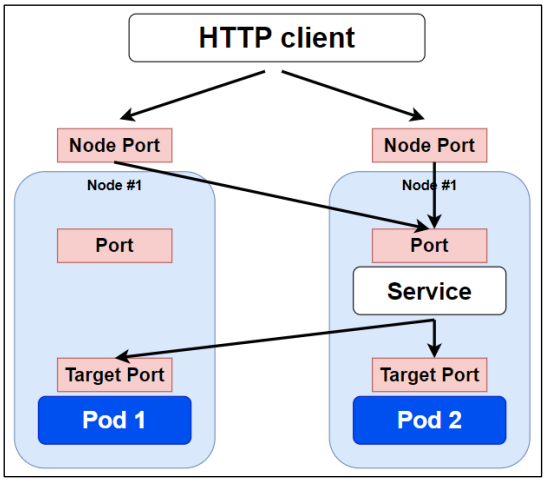

그림 62. Kubernetes Server의 Port 종류

k8s 서버에 생성된 pod, 즉 kasm container에 접속하기 위해서 node port – port – target port 를 거쳐 가상환경에 접속한다. 이때 사용자에게 노출되는 port는 node port이다.

3.3.14. 가상환경 중지하기

그림 63. 중지하고 싶은 가상환경

실행 중인 가상환경에서 중지 버튼을 누르면 웹 서버에서 flask 서버로 /stop api를 통해 요청을 보낸다. flask 서버에서는 kubectl exec 명령어를 통해 pod 내부로 접근해 만들어진 shell script를 통해 container를 중지하게 된다. 사용자가 접속을 중단해도 pod는 상태를 유지하고 있게 된다.

3.3.15. 가상환경 저장하기

그림 64. 저장하고 싶은 가상환경

중지된 가상환경에서 저장 버튼을 누르면 웹 서버에서 flask 서버로 /save api를 이용해 요청을 보낸다. 가상환경 저장 요청이 오면, 현재 가상환경의 상태를 안전하게 저장하기 위해 Kubernetes 서버에서 harbor에 container image를 push 한다. Docker 명령어인 push를 사용해 kasm docker image를 harbor에 업로드한다.

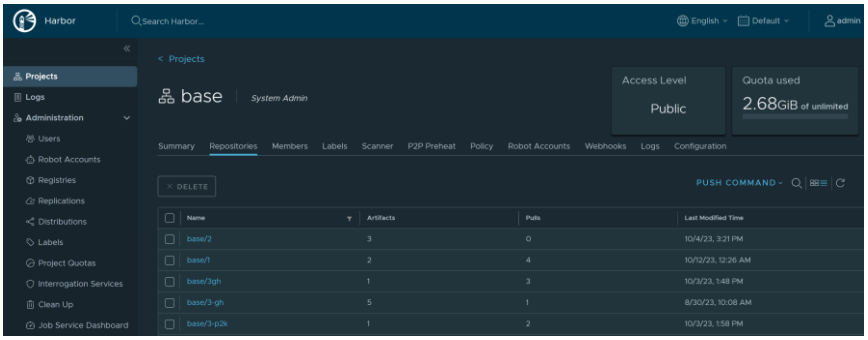

그림 65. 저장하기 기능을 통해 가상환경의 Container Image가 Private Docker Registry에 저장된 모습

직접 구현한 private registry harbor에 이미지가 push되는 방식은 base 폴더 내부에서 사용자의 id로 폴더를 구분하고, 해당 폴더 안에 한 사용자의 이미지들이 저장된다.

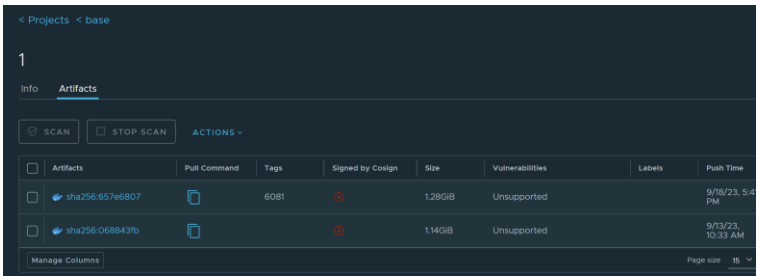

그림 66. 사용자별로 Container Image가 저장된 모습

각 사용자의 base/{userId} 안에 고유한 port 번호를 tags로 가지는 이미지들이 저장된다.

3.3.16. 가상환경 수정하기

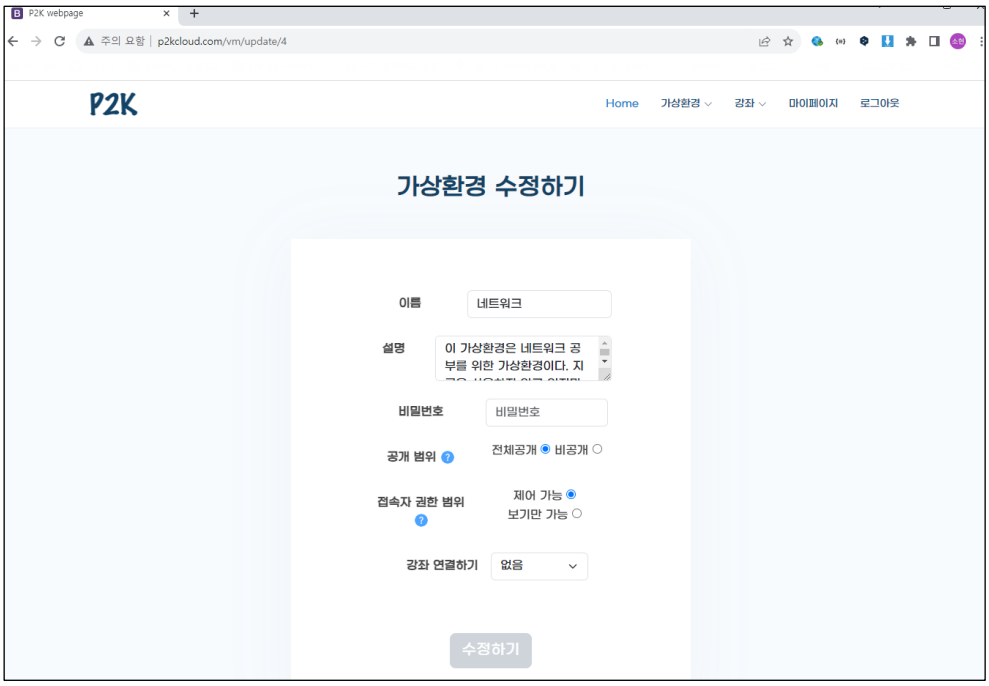

그림 67. 가상환경 수정하기

중지된 가상환경은 정보를 수정할 수 있다. 가상환경 이름, 설명, 접속 비밀번호, 공개 범위, 접속자 권한 범위를 수정하면 웹 서버에 연동된 데이터베이스(MariaDB)에 정보가 수정된다. 변경된 정보는 가상환경을 실행할 때 flask 서버로 함께 보내, start.sh가 실행될 때 수정된 정보가 반영된다.

3.3.17. 가상환경 삭제하기

삭제하기를 누르면 웹 서버에서 flask 서버로 /delete api를 이용해 요청을 보낸다. flask 서버에서는 kubectl의 delete 명령어를 이용해 deployment pod와 service pod를 제거하고 작성한 deployment.yaml 파일과 service.yaml 파일도 삭제한다. Flask 서버로부터 정상적으로 제거되었다는 응답을 받으면 웹 서버의 데이터베이스에서도 해당 가상환경에 관한 정보를 삭제한다.

3.3.18. 가상환경 조회하기

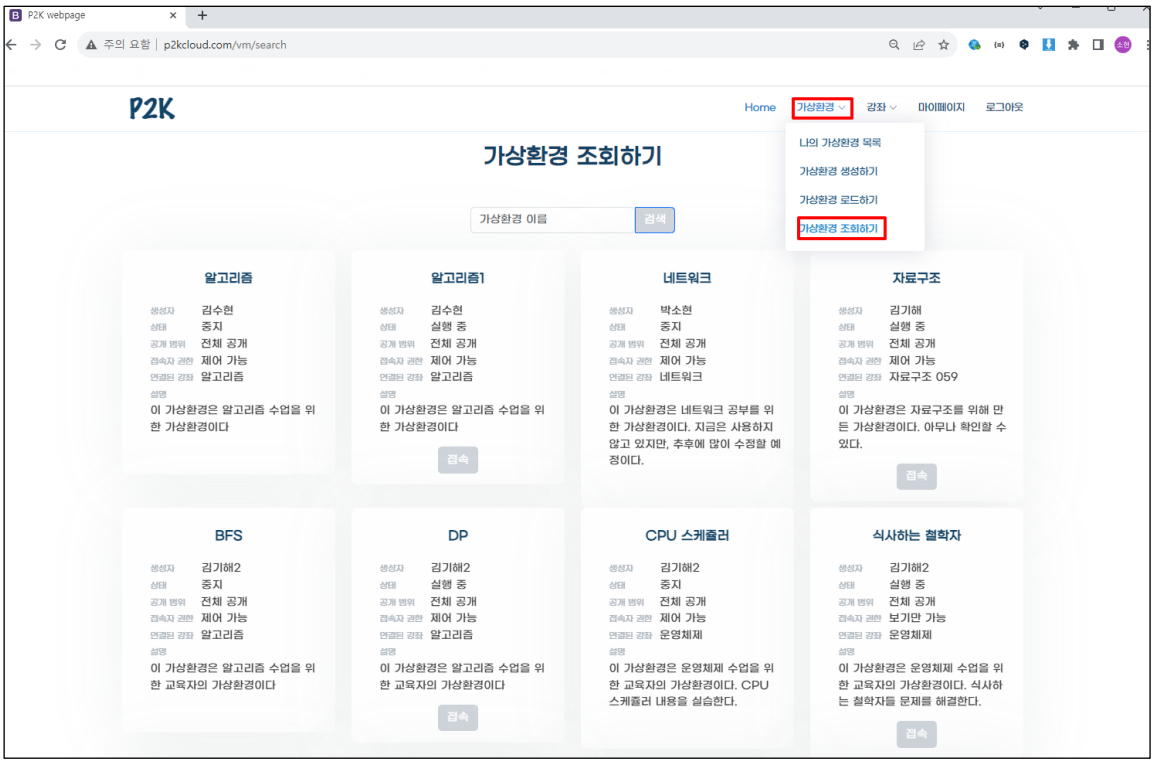

그림 68. 사용자 - 가상환경 조회하기

공개 범위가 전체 공개로 설정된 가상환경의 전체 목록을 조회하고 검색할 수 있다. 실행 중인 가상환경에만 접속 버튼이 활성화되어, 접속자로 접속할 수 있다.

3.4. 강좌 서비스

3.4.1. 강좌 서비스 제공

교육자와 수강생을 분리한 강좌 서비스를 제공하여 학습 관리 시스템(LMS)을 구축한다. 교육자는 해당 강좌 서비스를 활용하여 강좌를 생성하고 수업을 운영할 수 있다. 강좌에 등록된 수강생들을 조회하고 수강생의 학습 진행 상황을 손쉽게 파악할 수 있다. 또한, 교육자는 학습 환경 설정 후 로드 key를 게시판에 업로드하여 수강생에게 일관된 학습 환경 제공을 제공할 수 있다. 수강생들은 수업의 학습 환경과 자료를 제공받을 수 있다. 또한 교육자의 가상 환경에 접속하여 수업에 실시간으로 참여할 수 있어, 수업의 이해와 참여도를 높인다. 질문, 자유 게시판의 질문과 답변을 통한 수업 구성원들 간의 소통을 돕는다. 이로써 수업의 효율성을 극대화하고 효과적인 학습 경험을 제공한다.

3.4.2. 강좌 관리

3.4.2.1. 수강생

1) 강좌 신청

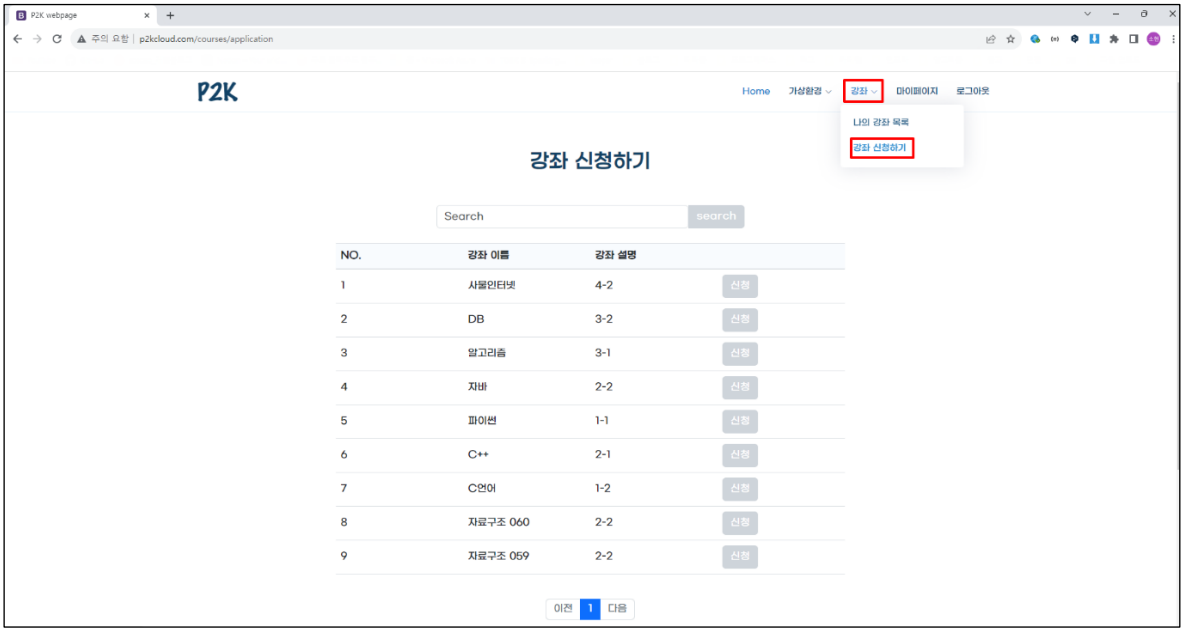

그림 69. 강좌 신청 페이지

[강좌] – [강좌 신청하기] 탭을 통해 들어가게 되면, 강좌 신청하기 페이지가 나오게 되고, 해당 페이지를 통해 수강생은 강좌를 신청할 수 있다.

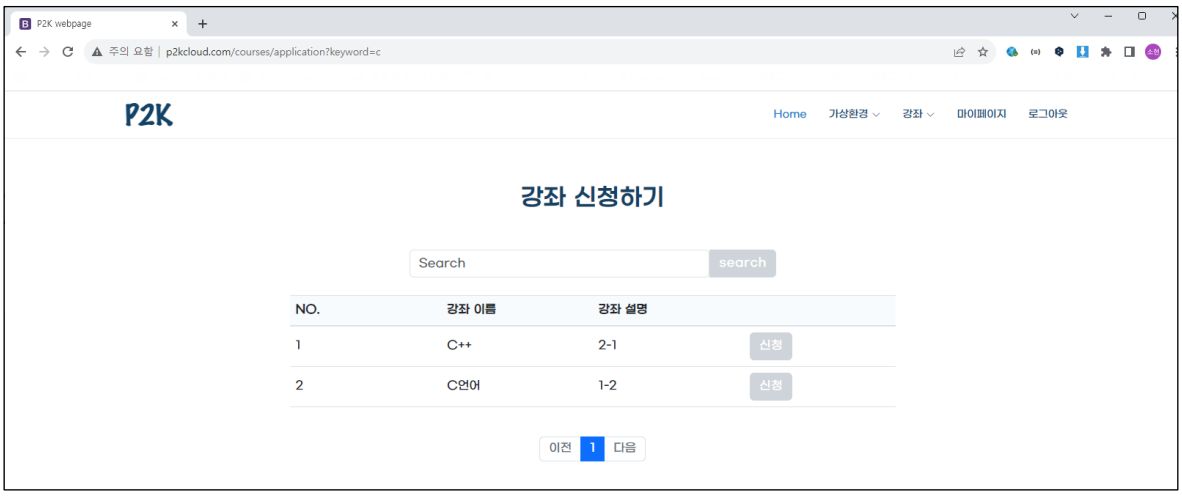

그림 70. C만 입력했을 때 나오는 강좌 목록

수강생은 강좌 신청하기 페이지에서 강좌를 강좌 이름으로 검색할 수 있다. 강좌 이름에 검색 키워드를 포함한 모든 강좌가 조회된다.

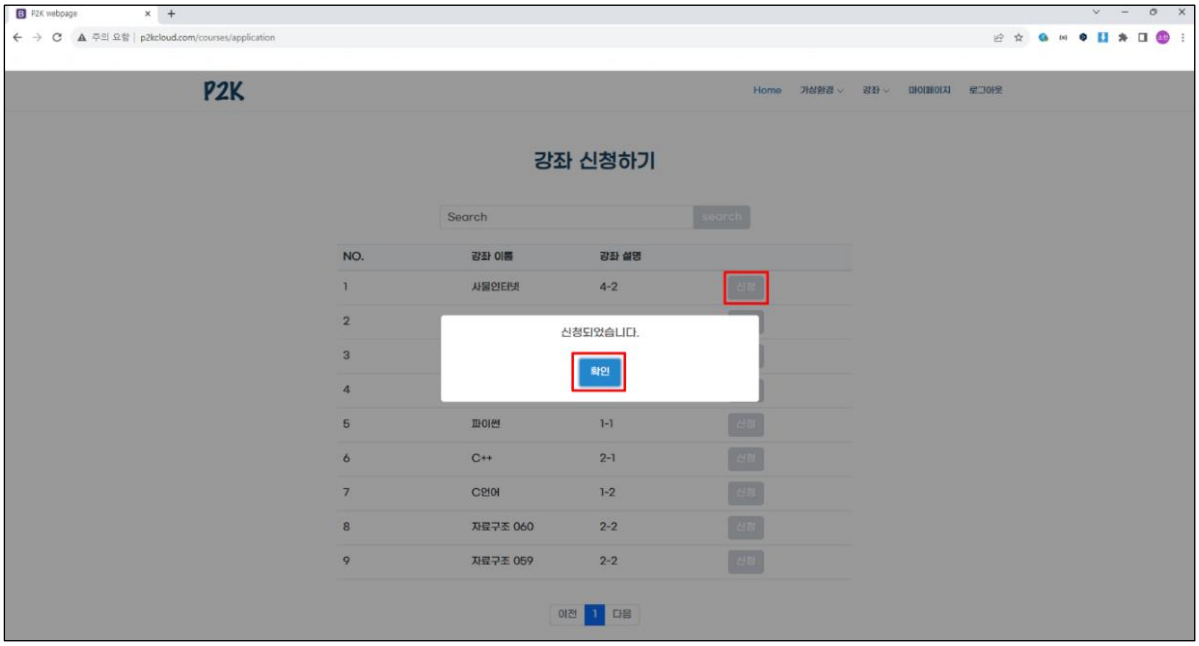

그림 71. 강좌 신청 확인 모달창

원하는 강좌의 신청 버튼을 누르면 강좌 신청이 되고, 신청 확인 창이 뜬다.

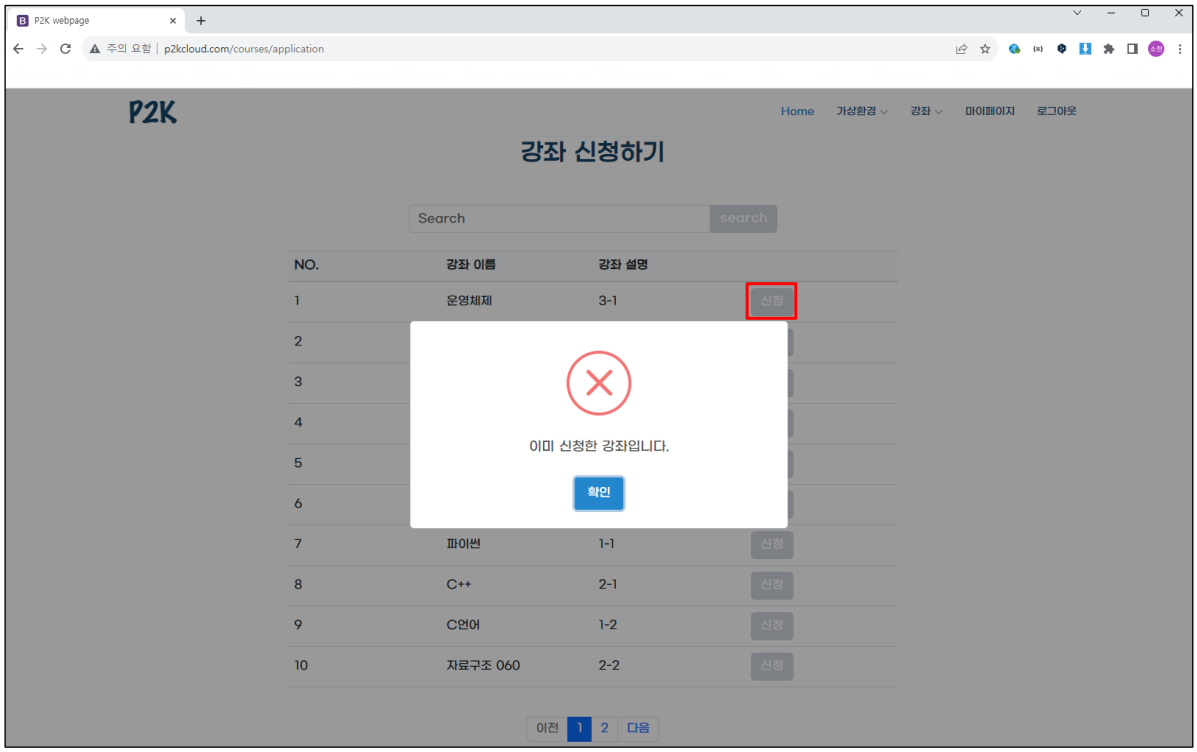

그림 72. 신청한 강좌 재신청 시 뜨는 신청 오류 모달창

이미 신청한 강좌일 경우에는 신청 오류 창이 뜬다.

2) 강좌 목록 조회

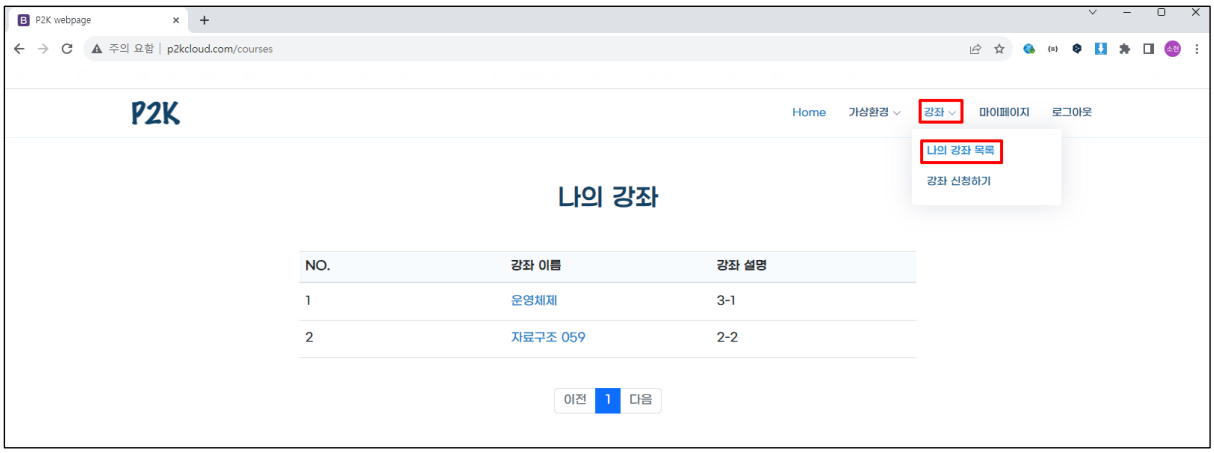

그림 73. 학생 신청 강좌 목록

[강좌] – [나의 강좌 목록] 탭을 통해 학생은 자신이 속한 강좌의 목록을 조회할 수 있다. 교육자의 승인을 받은 강좌만 조회할 수 있다.

3) 강좌에 연결된 나의 가상 환경 접속

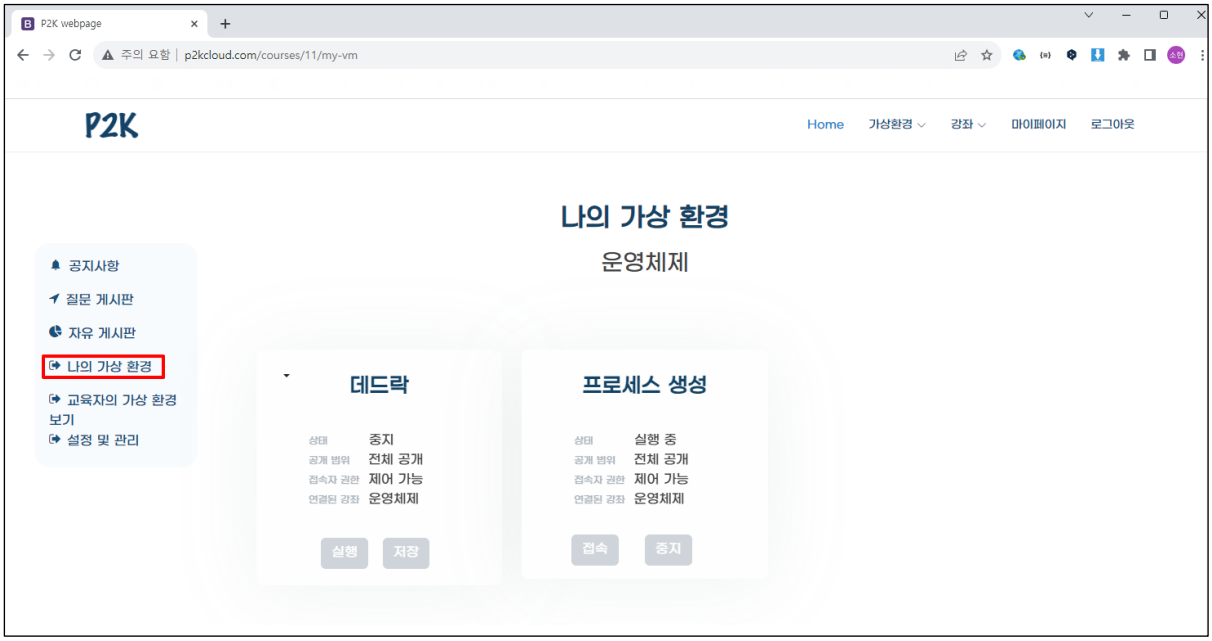

그림 74. 강좌와 연결된 나의 가상 환경 (운영체제 수업 선택 예시)

수강생은 신청한 강좌 목록에서 [운영체제] – [나의 가상 환경] 탭을 통해 나의 가상 환경 페이지에서 해당 강좌에 연결한 자신의 가상 환경 목록을 조회할 수 있다.

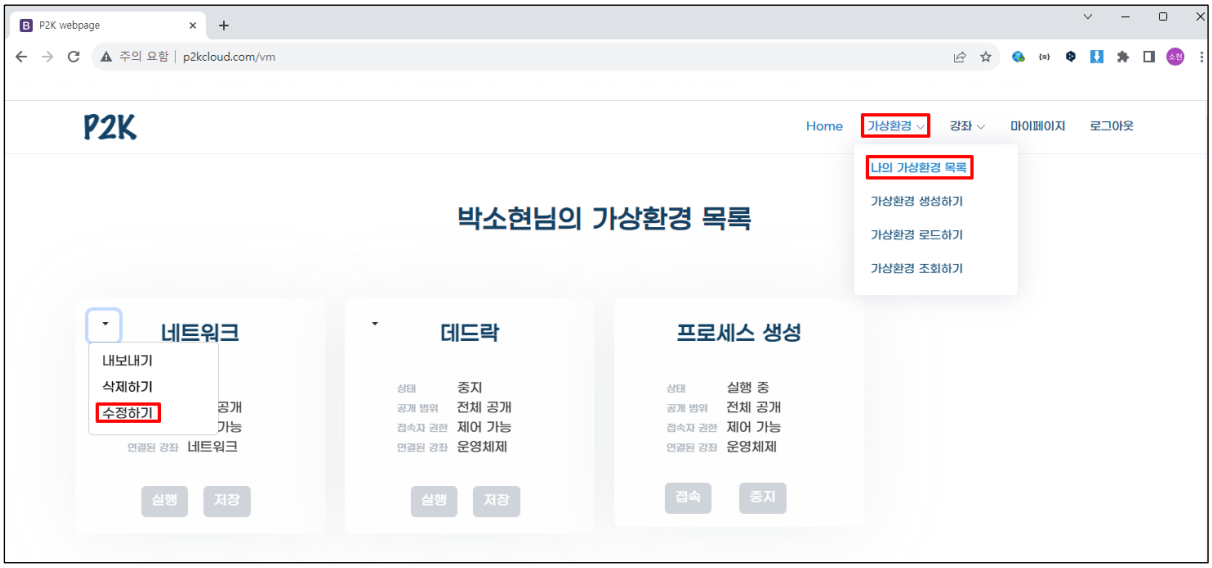

그림 75. 가상 환경 수정페이지 접속 방법

[가상환경] – [나의 가상환경 목록] – [수정하기]를 통해 가상 환경 수정 페이지로 넘어가게 된다.

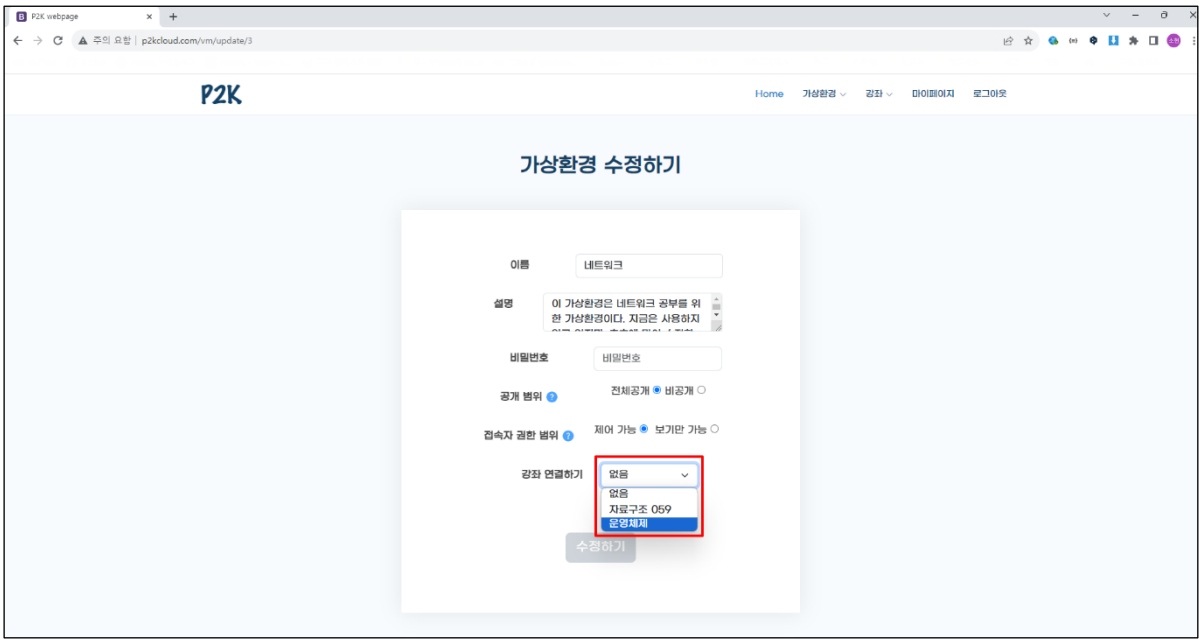

그림 76. 가상 환경 수정하기에서의 강좌 연결하기

가상 환경 수정하기에서 강좌 연결하기를 통해 가상 환경이 연결될 강좌를 자신이 신청한 강좌 목록에서 선택할 수 있다. 강좌 페이지 내부에서 강좌에 연결된 가상 환경의 목록을 접속할 수 있으므로 편리한 학습환경을 유도한다.

4) 강좌에 연결된 교육자의 가상 환경 접속

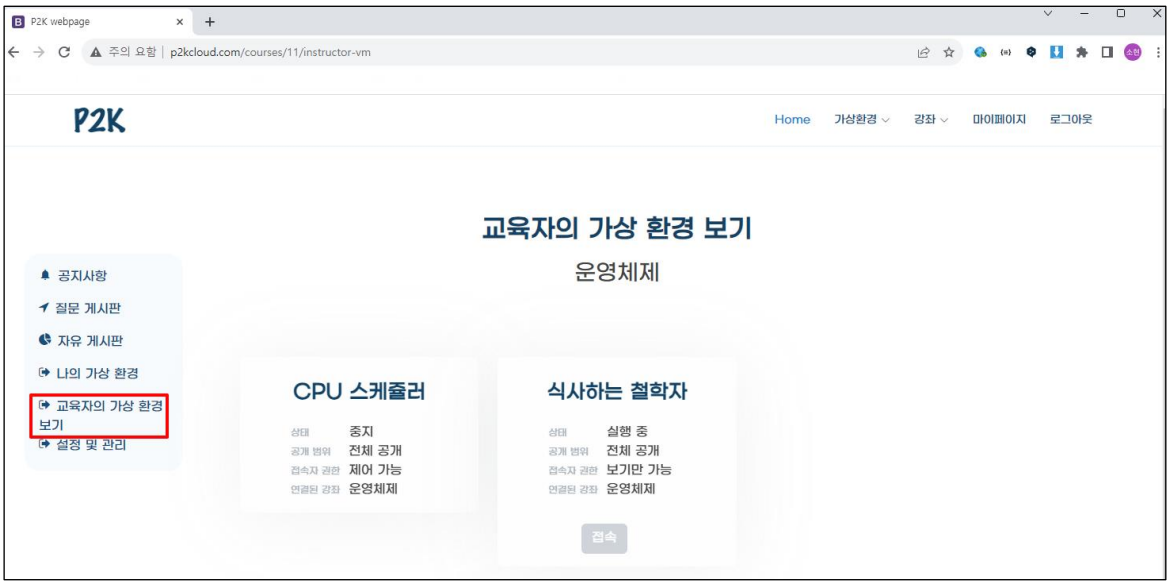

그림 77. 강좌에 연결된 교육자의 가상 환경

수강생은 [교육자의 가상 환경 보기]에서 해당 강좌에 연결된 교육자의 가상 환경에 동시 접속하여 교육자의 실습 환경에 실시간으로 접속하고 참여할 수 있다. 교육자가 설정한 권한에 따라, 제어 가능한 경우에는 수강생이 해당 가상 환경을 조작할 수 있고, 보기만 가능한 경우에는 읽기 전용(read-only) 모드로만 열람할 수 있다.

5) 강좌 구성원 조회 및 강좌 취소

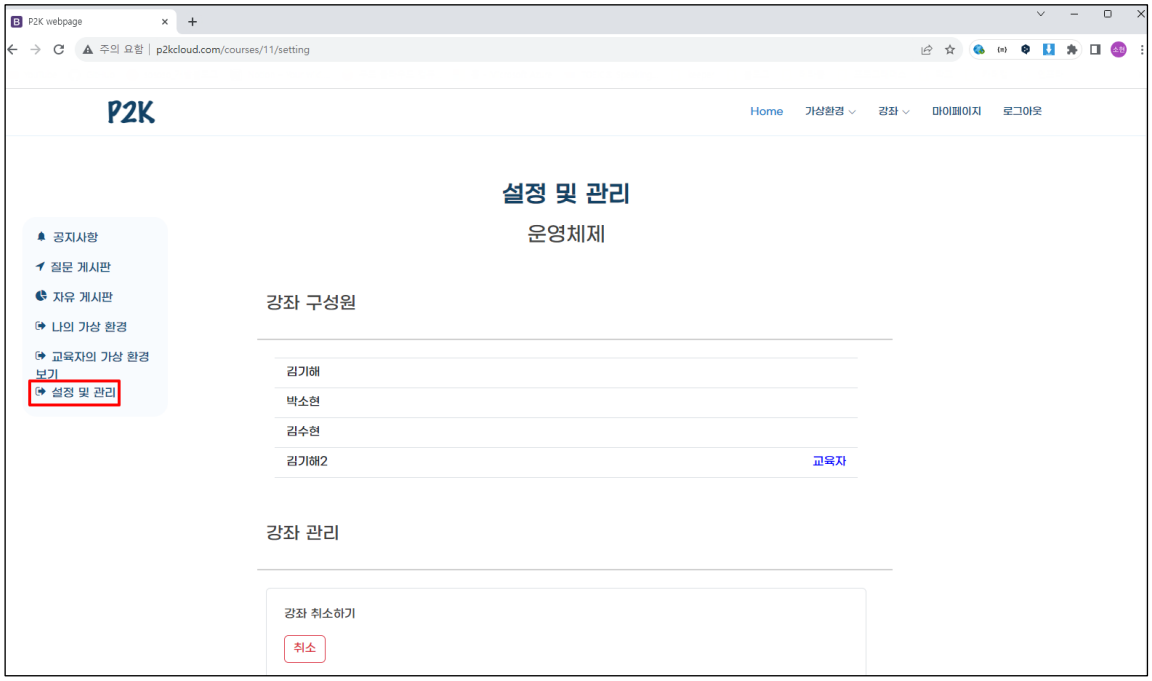

그림 78. 강좌 구성원 조회 및 강좌 취소

[구성원 조회]

수강생은 [설정 및 관리]에서 강좌 구성원 목록을 조회할 수 있다. 교육자일 경우 교육자임을 확인하는 부가적인 표시가 나타난다.

[수강 취소]

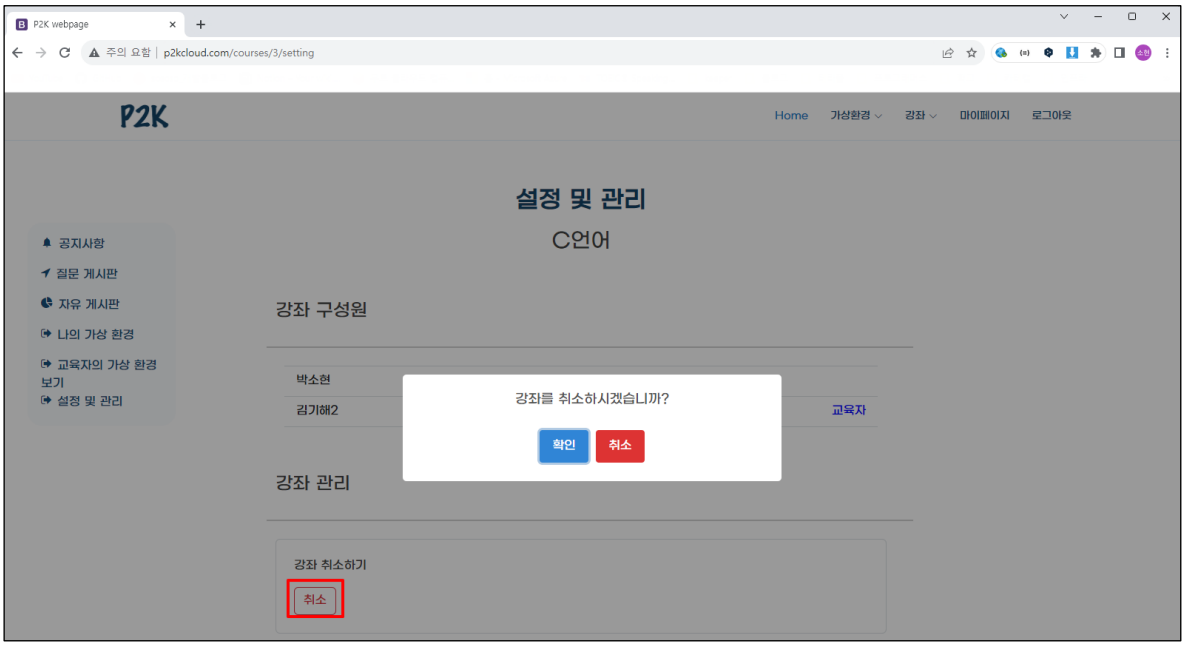

그림 79. 강좌 취소 모달창

[설정 및 관리]의 [강좌 취소하기] – [취소]를 누르면 강좌 취소 모달창이 뜬다.

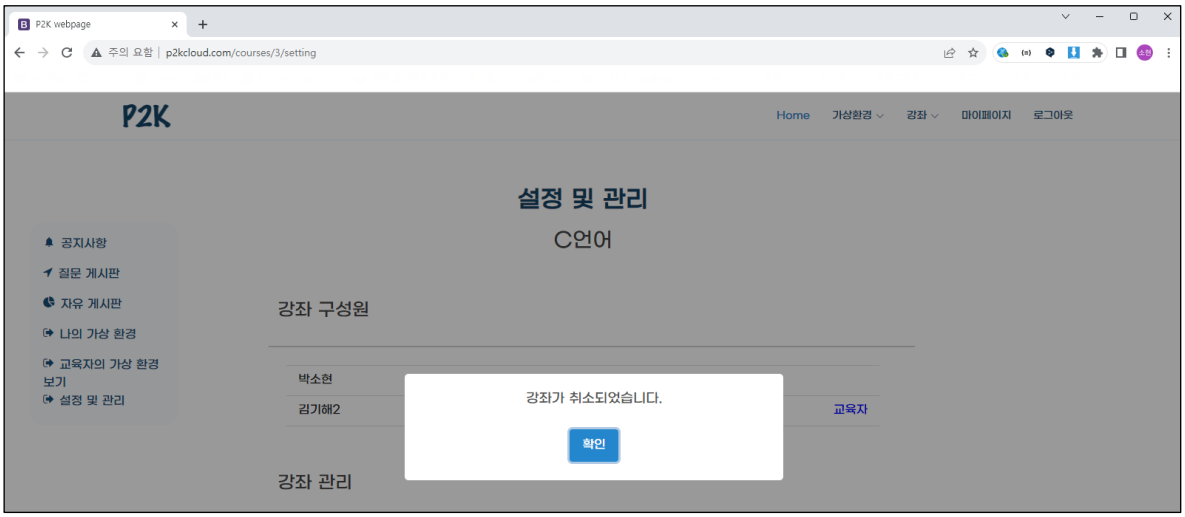

그림 80. 강좌 취소 확인 모달창

수강생은 강좌 취소 모달창의 [확인] 버튼을 누르고 강좌를 취소할 수 있다. 강좌가 취소되면, 나의 강좌 페이지로 리다이렉트 된다.

3.4.2.2. 교육자

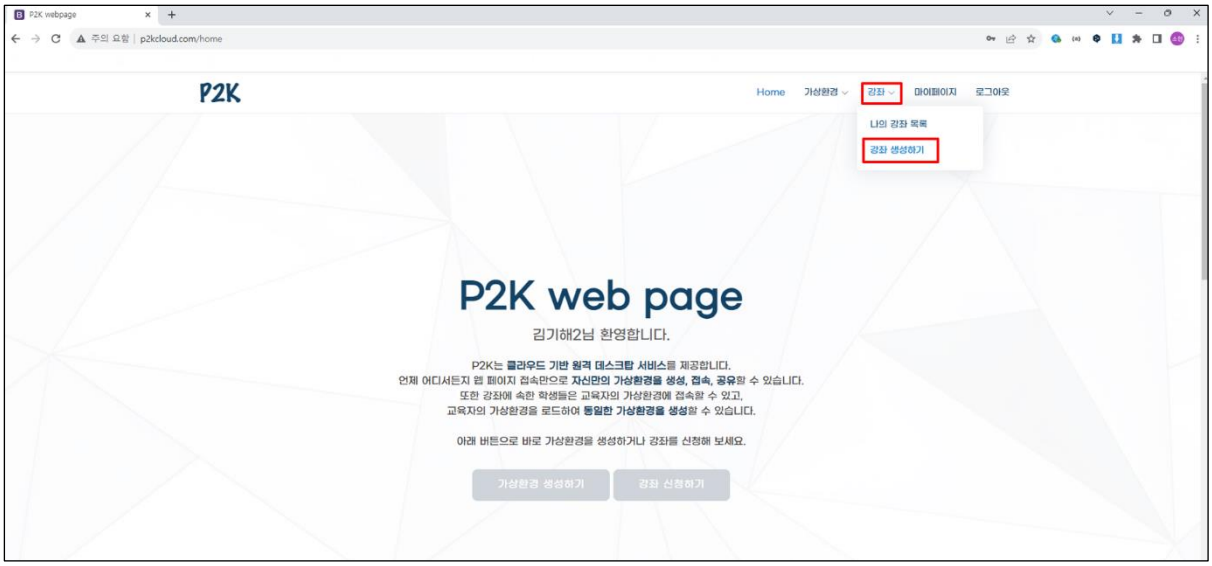

그림 81. 교육자의 메인 페이지

교육자의 메인 페이지이다. 교육자는 학생과는 다르게 [강좌] – [강좌 신청하기]가 아닌, [강좌] – [강좌 생성하기] 탭이 있다.

1) 강좌 생성

그림 82. 강좌 생성하기

관리자의 승인을 받은 교육자는 [강좌] – [강좌 생성하기] 탭을 통해 강좌 생성하기 페이지에서 강좌를 생성할 수 있다. 강좌 이름, 설명(학년-학기)을 모두 기입해야 강좌를 생성할 수 있다.

2) 강좌 목록 조회

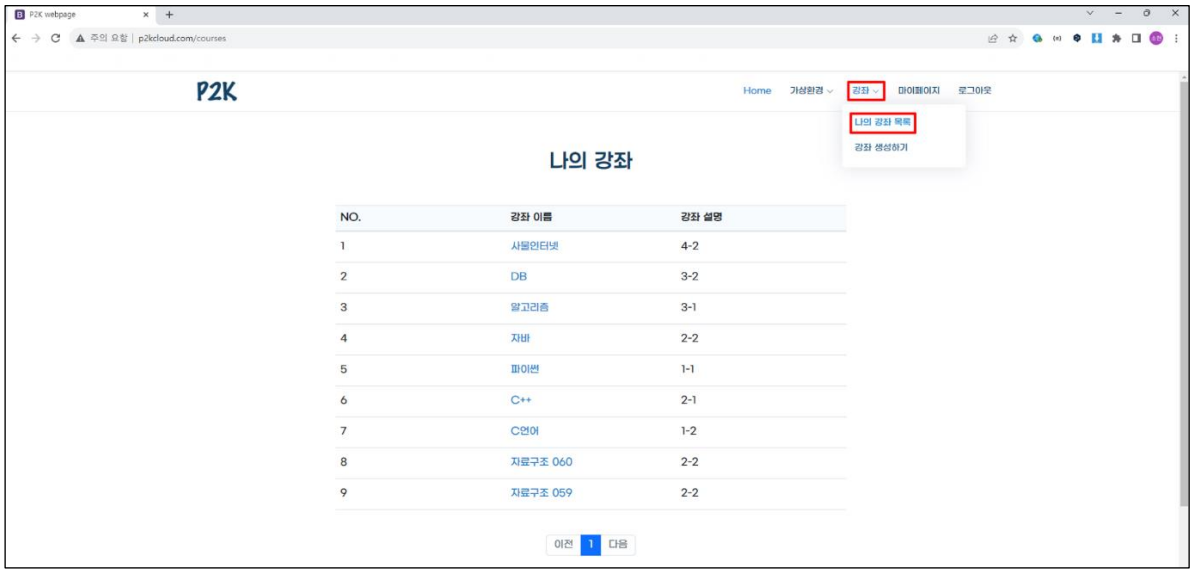

그림 83. 교육자 강좌 목록 조회

교육자는 [강좌] – [나의 강좌목록]에서 자신이 생성한 강좌의 목록을 조회할 수 있다. 자세한 내용은 수강생의 강좌 목록 조회와 동일하다.

3) 강좌 신청 수락/거절

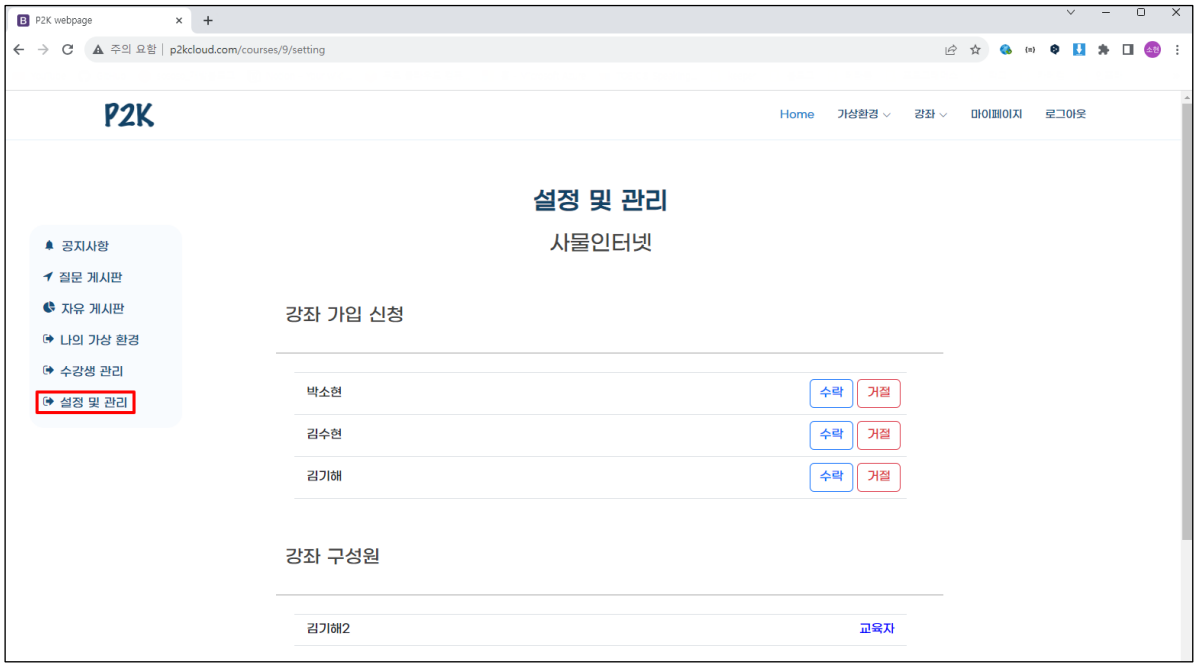

그림 84. 교육자 강좌 설정 및 관리 (사물인터넷 예시)

교육자는 [사물인터넷] - [설정 및 관리]의 강좌 가입 신청 목록에서 강좌 신청인원에 대한 강좌 수락/거절을 할 수 있다.

[강좌 신청 수락]

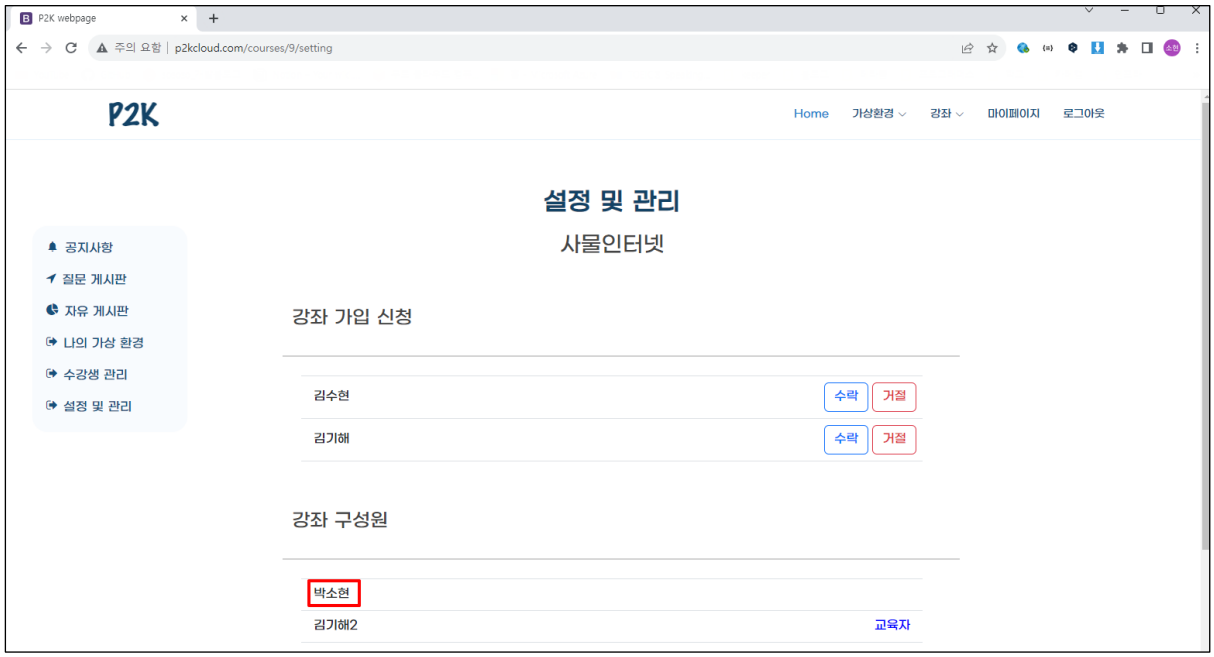

그림 85. 강좌 가입 신청 수락

강좌 신청 수강생에 대한 수락을 할 경우, 해당 학생은 강좌의 수강생이 되며 강좌 가입 신청 목록에서 제외되고 강좌 구성원으로 포함된다.

[강좌 신청 거절]

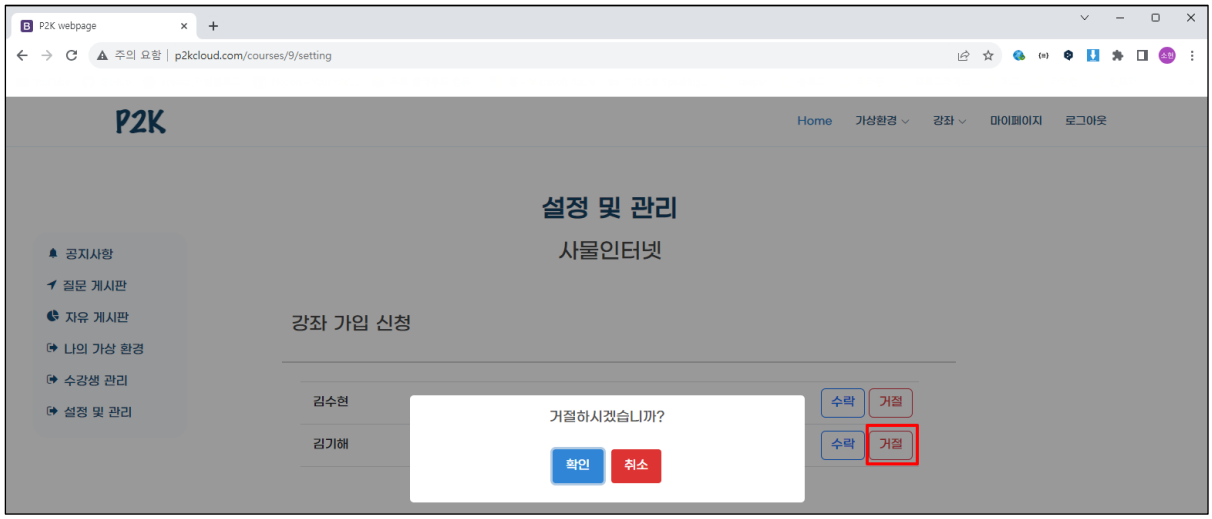

그림 86. 강좌 가입 신청 거절 모달창

강좌 신청 수강생에 대한 거절을 할 경우, 강좌 가입 신청 거절 모달창이 뜨게 된다. 거절을 하고 싶은 경우, [거절] 버튼을 클릭하면 해당 학생의 가입 신청이 거절된다.

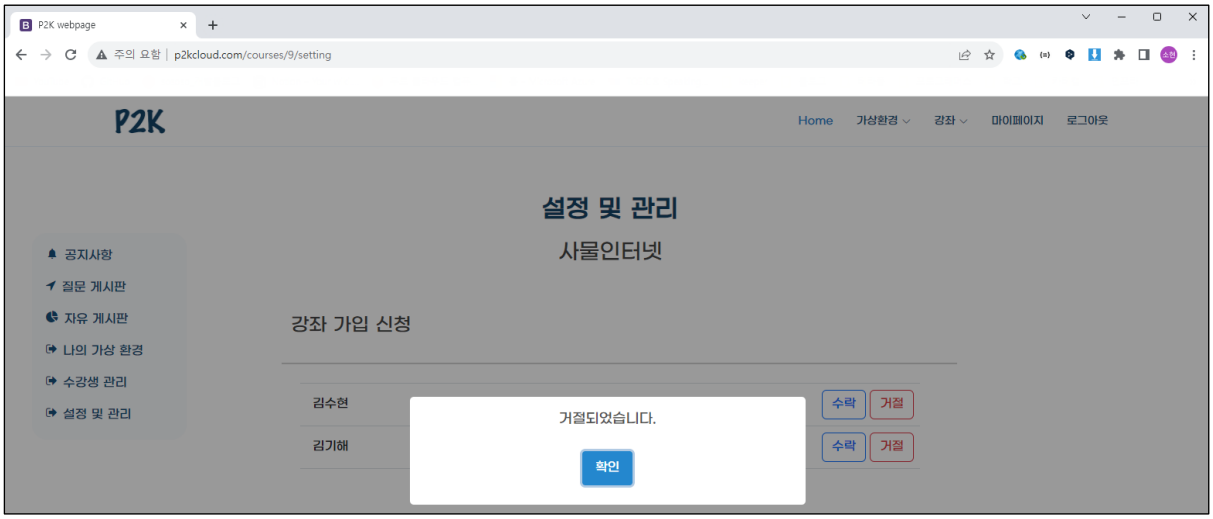

그림 87. 강좌 신청 거절 확인 모달창

[거절] 버튼을 클릭하면, 강좌 신청 거절 확인 모달창이 뜨게 되고 강좌 가입 신청 목록에서 제외된다. 해당 가입 신청 학생은 강좌에 속할 수 없다.

4) 강좌 구성원 조회

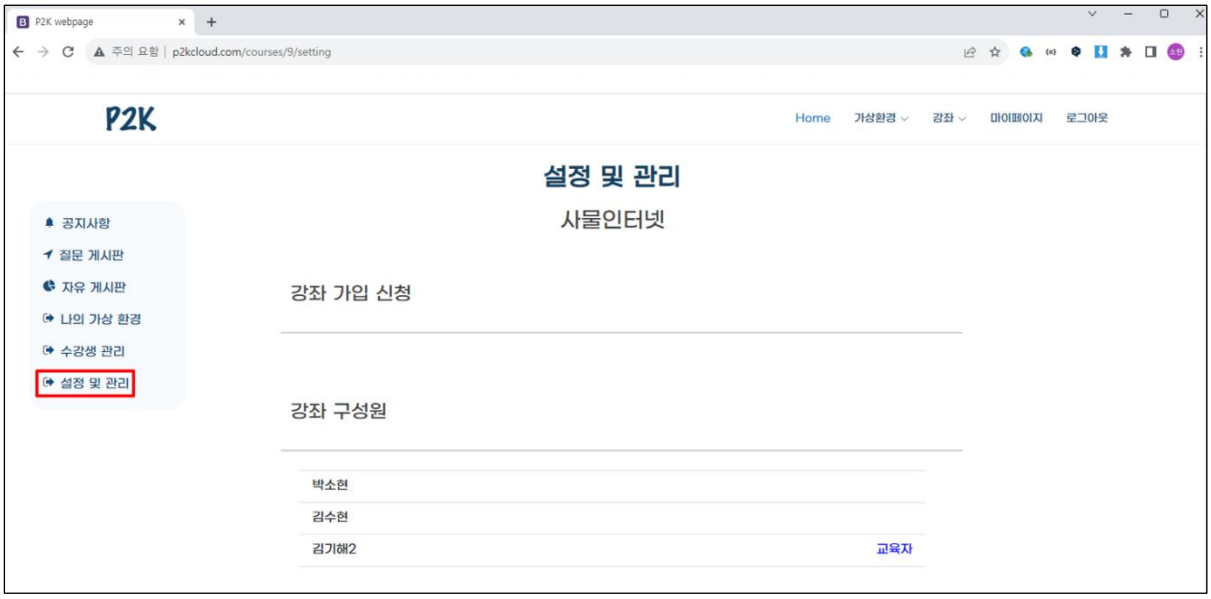

그림 88. 교육자 - 강좌 구성원 확인

교육자는 [설정 및 관리]의 [강좌 구성원]에서 해당 강좌에 속한 구성원 목록을 조회할 수 있다. 자세한 내용은 수강생의 강좌 구성원 조회와 동일하다.

5) 강좌에 연결된 나의 가상 환경 접속

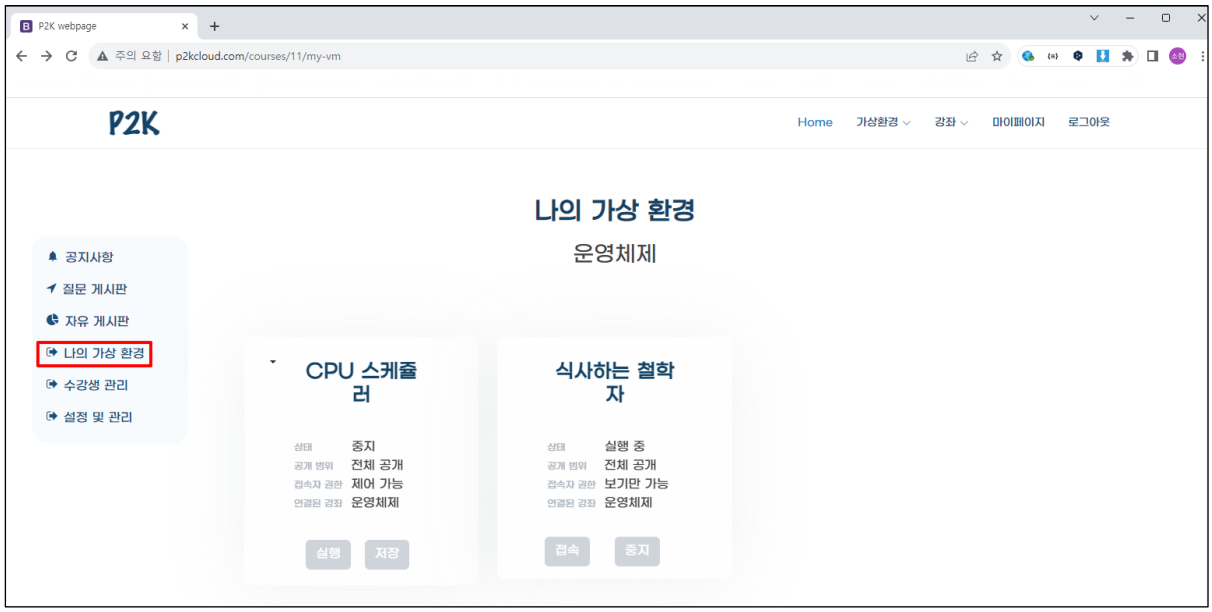

그림 89. 교육자 - 강좌에 연결된 나의 가상 환경 조회 (운영 체제 예시)

교육자는 [나의 가상 환경]에서 해당 강좌에 연결한 자신의 가상 환경 목록을 조회한다. 자세한 내용은 수강생의 강좌에 연결된 나의 가상 환경 접속과 동일하다.

6) 강좌에 연결된 나의 가상 환경 공개 범위/접속자 권한 수정

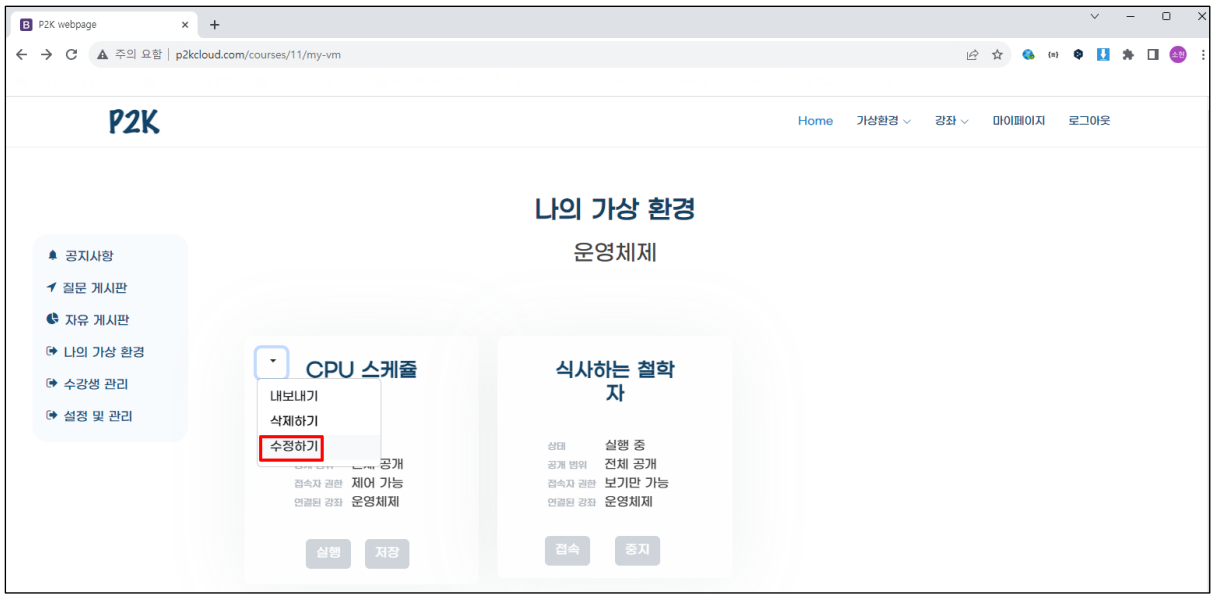

그림 90. 교육자 - 나의 가상환경 공개 범위 수정 (CPU 스케줄러 예시)

교육자는 [나의 가상 환경] – [CPU 스케줄러] – [수정하기]를 통해 가상환경을 수정할 수 있다.

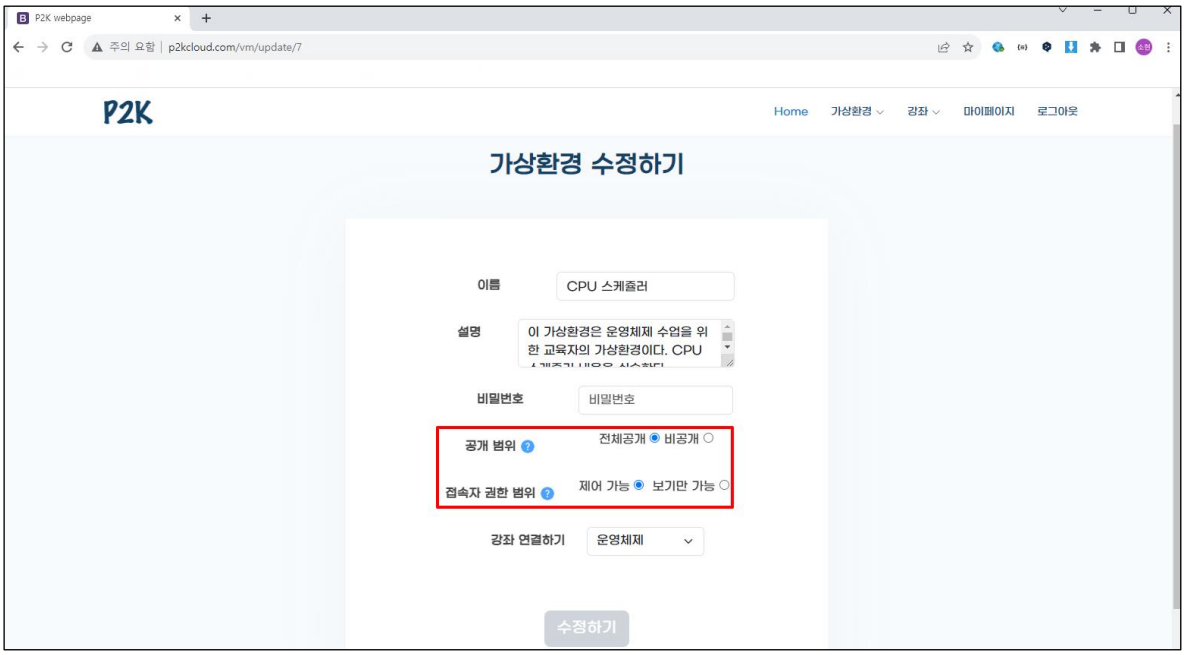

그림 91. 교육자 - 가상 환경 공개 범위/접속자 권한 범위 수정

교육자는 강좌에 연결된 가상 환경의 공개 범위와 접속자 권한을 수정할 수 있다.

7) 수강생의 가상 환경에 접속

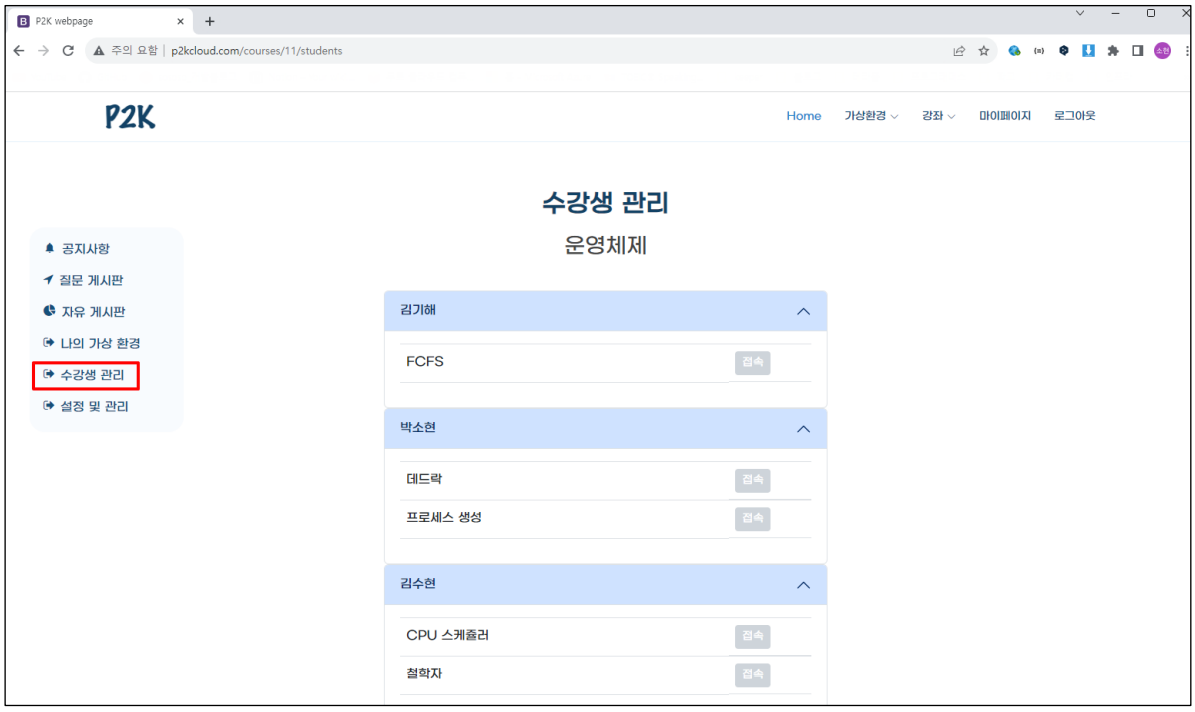

그림 92. 교육자 - 수강생 관리에서의 학생 가상 환경 접속

교육자는 [수강생 관리]에서 수강생이 해당 강좌에 연결한 가상 환경의 목록을 조회할 수 있다. 교육자는 수강생의 학습 현황을 파악하기 위해 원하는 수강생의 가상 환경 [접속] 버

튼을 눌러 해당 가상 환경으로 접속한다. 가상 환경의 접속자 권한이 제어 가능 상태라면 교육자는 해당 가상 환경을 제어할 수 있고 접속자 권한이 보기만 가능 상태라면 교육자는 읽기 모드(read-only)로 조회만 가능하다.

8) 강좌 삭제

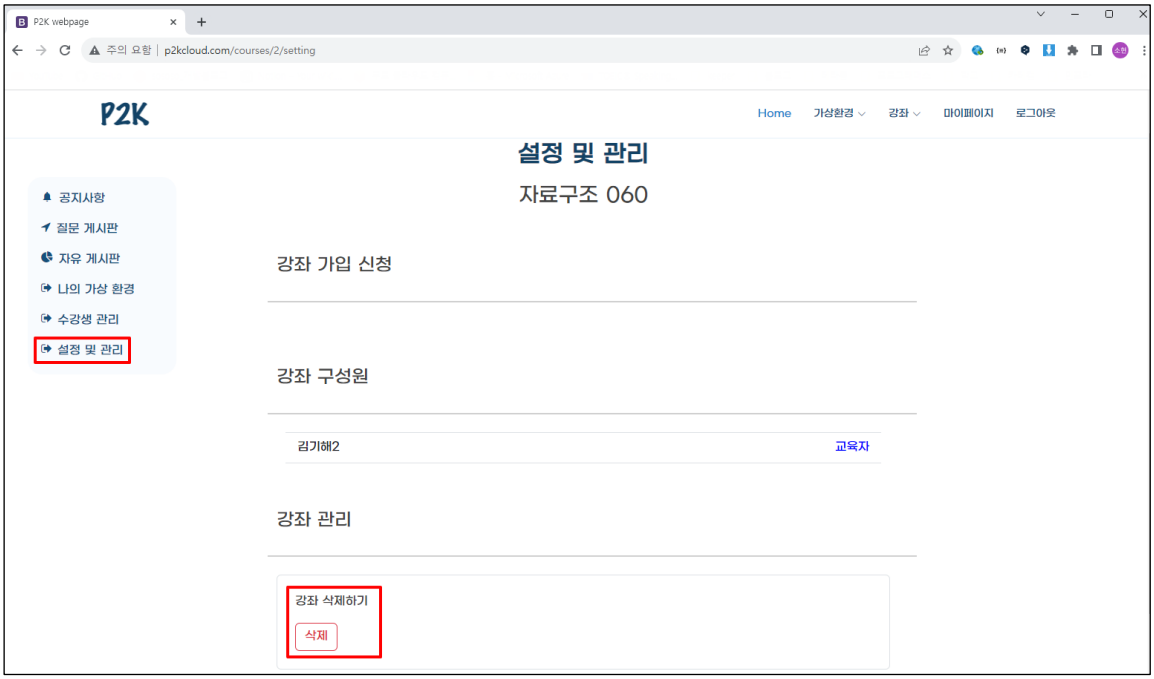

그림 93. 교육자 - 강좌 삭제하기

교육자는 강좌의 [설정 및 관리]에서 [강좌 삭제하기] – [삭제]를 통해 자신이 생성한 강좌를 삭제할 수 있다.

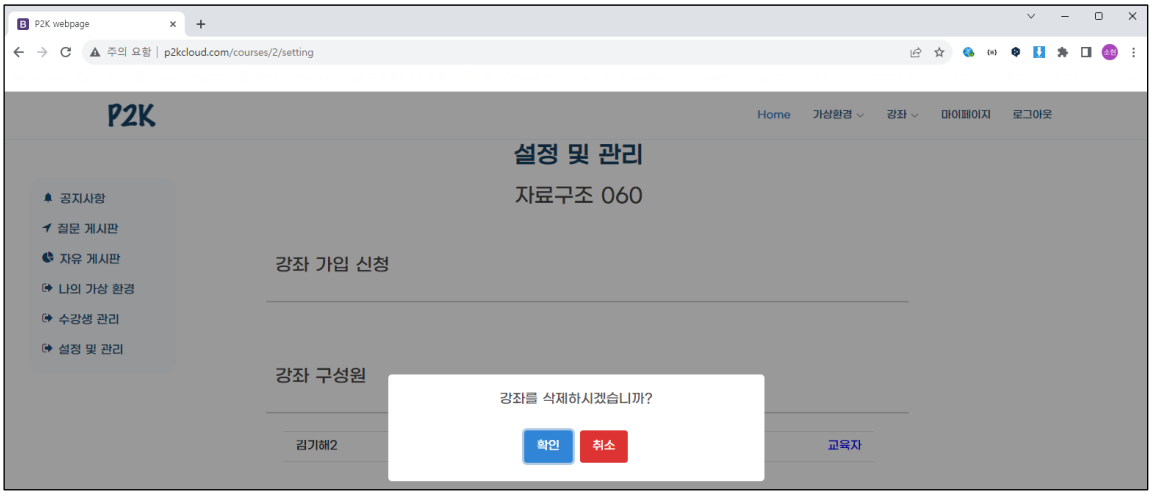

그림 94. 교육자 - 강좌 삭제 모달창

[강좌 삭제하기] – [삭제]를 누르면 강좌 삭제 팝업 창이 뜬다.

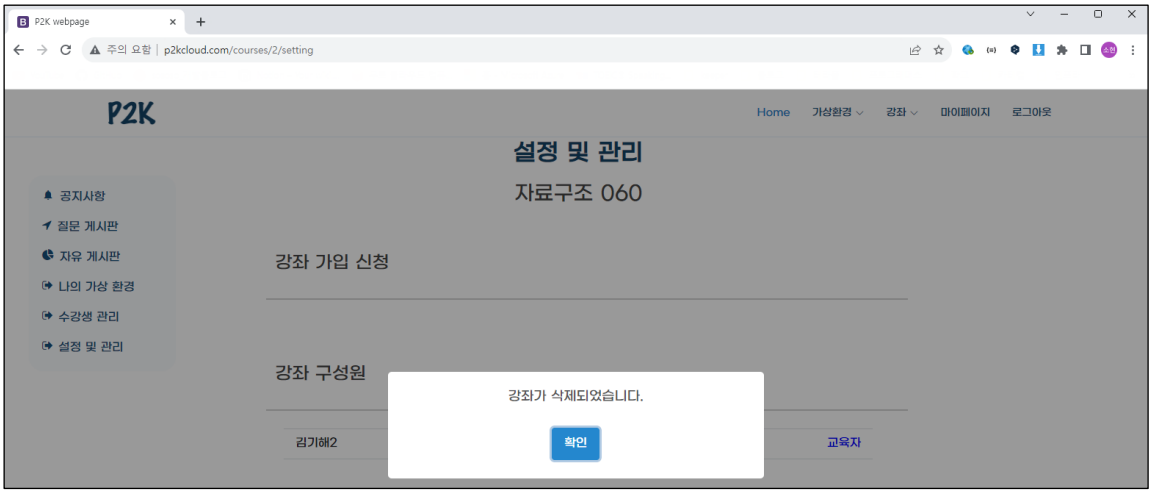

그림 95. 교육자 - 강좌 삭제 확인 모달창

강좌 삭제 팝업 창에서 [확인]을 누르면 해당 강좌의 글, 댓글이 삭제되고 가상 환경에 연결된 강좌도 연결 해제된다.

3.4.3. 게시판 관리

강좌에 속한 교육자, 수강생들은 공지사항, 질문, 자유 게시판을 이용할 수 있다.

1) 공지사항 게시판

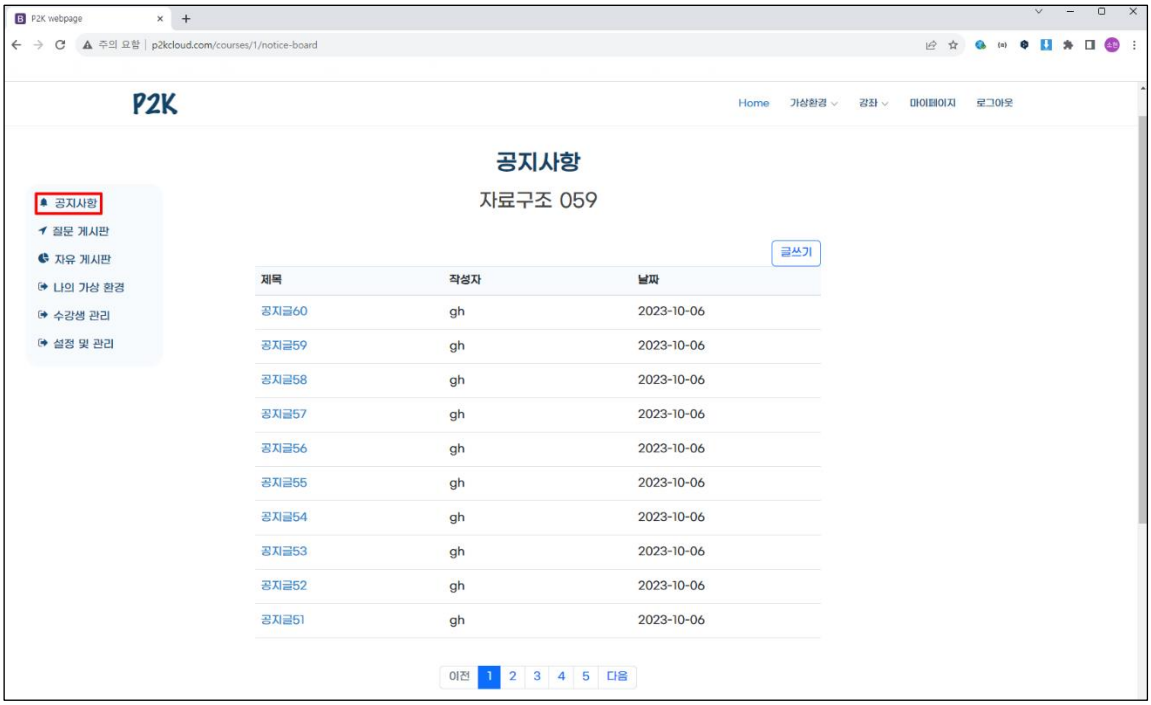

그림 96. 수강자/교육자 - 강좌 공지사항 목록

강좌의 공지사항 글은 강좌 구성원이 모두 조회할 수 있다.

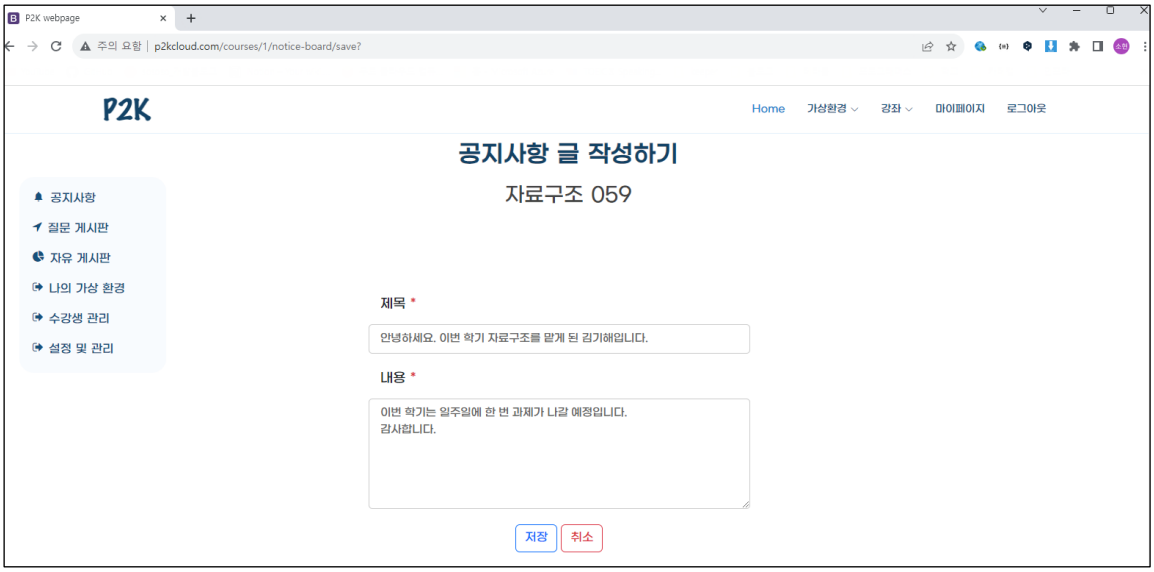

그림 97. 교육자 – 강좌 공지사항 등록

해당 강좌의 교육자는 [공지사항] – [글쓰기]를 통해 공지사항을 작성할 수 있다.

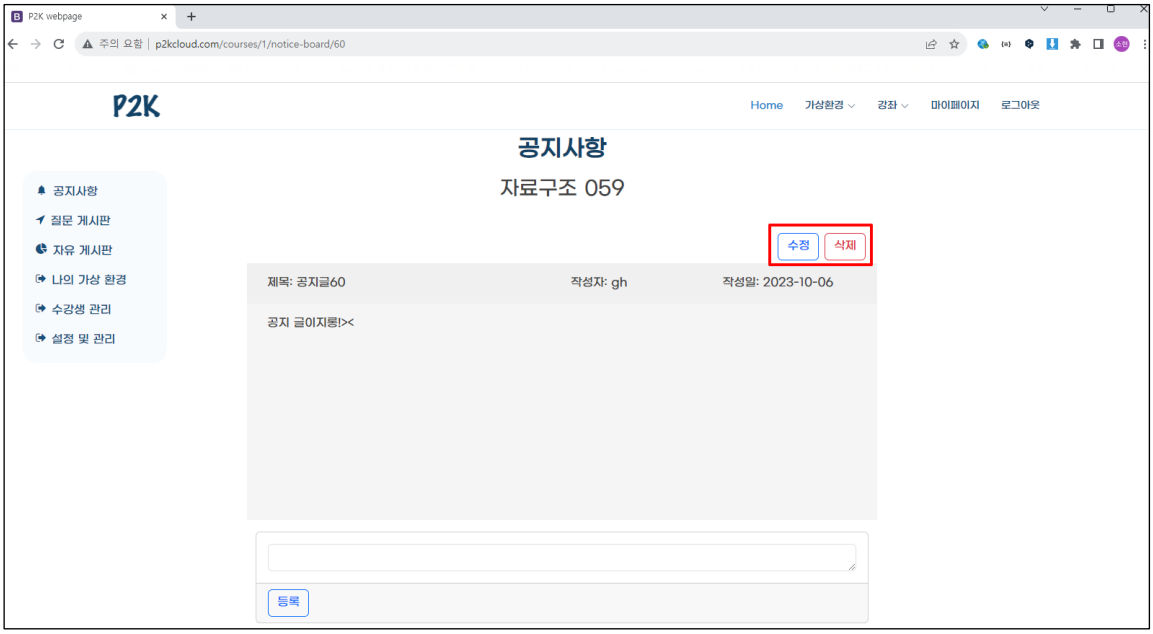

그림 98. 교육자 - 강좌 공지글 수정/삭제

해당 강좌의 교육자는 [공지사항] – [공지글] – [수정]/[삭제]를 통해 공지글을 수정/삭제할 수 있다.

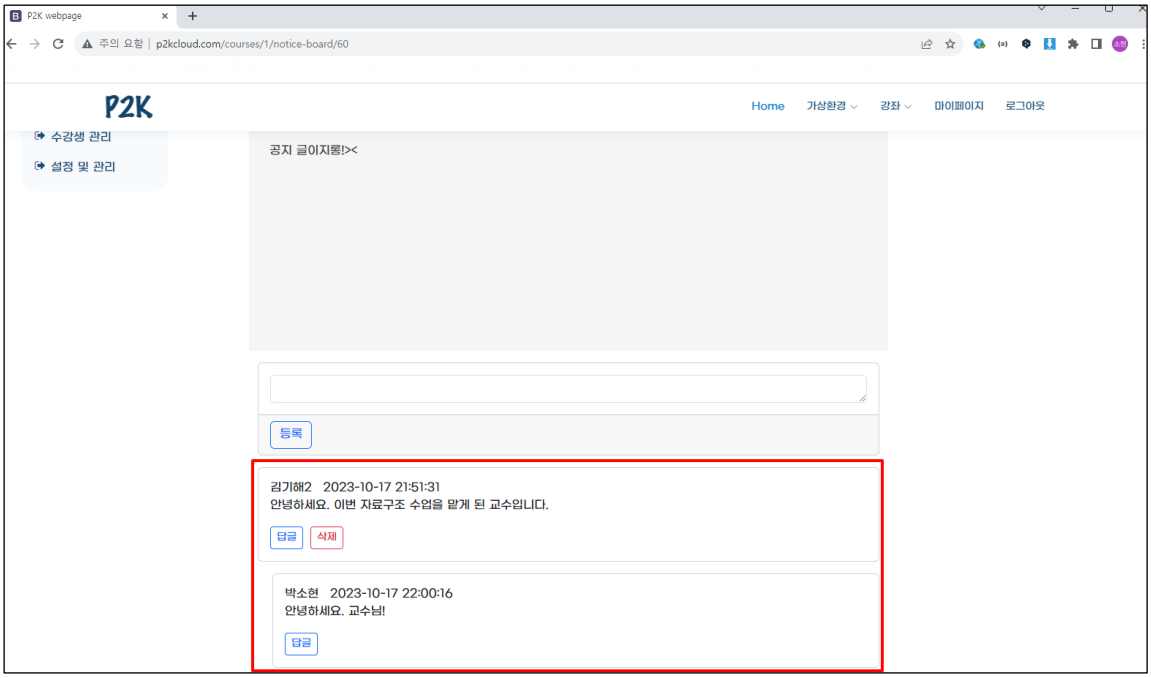

그림 99. 수강자/교육자 - 공지사항 댓글 작성

공지사항의 댓글은 강좌의 구성원 모두 작성 가능하다.

2) 질문 게시판

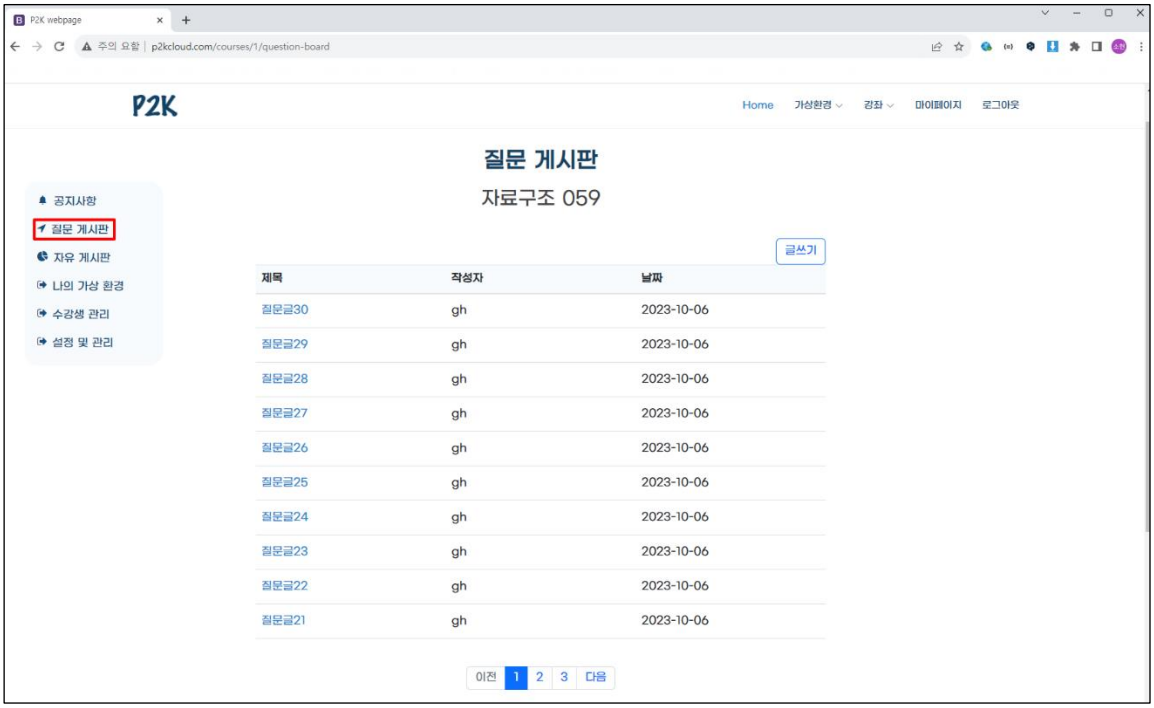

그림 100. 수강생/교육자 - 강좌 질문 게시판 목록

질문 게시판은 강좌 구성원 모두 조회할 수 있다.

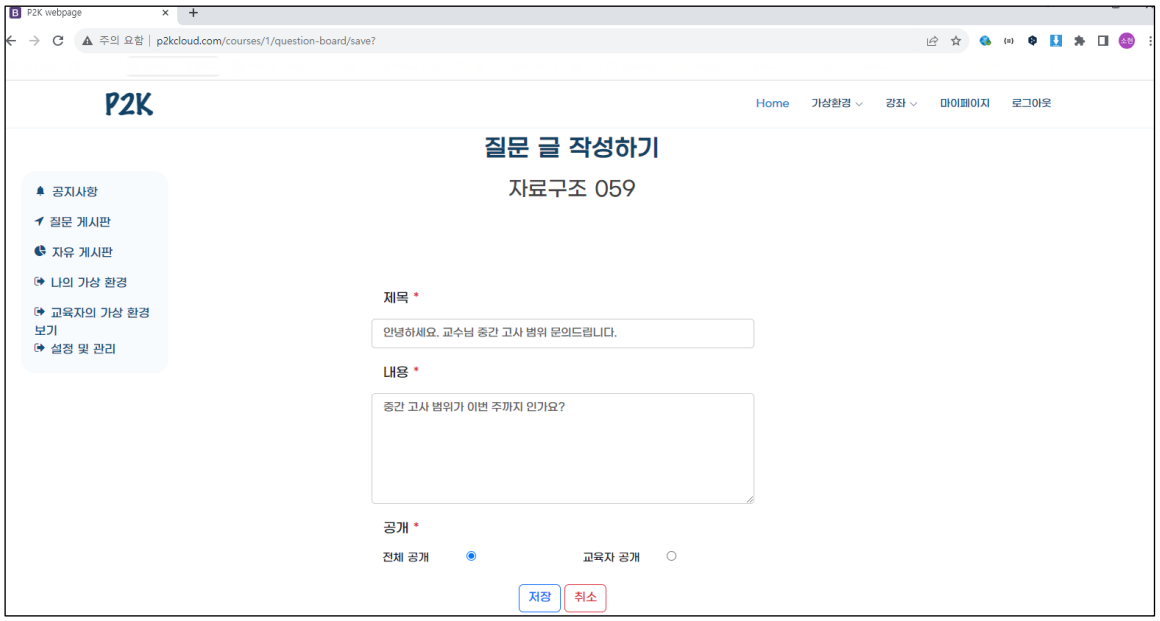

그림 101. 수강자/교육자 - 질문 글 작성하기

[질문 게시판] – [글쓰기]를 통해 강좌의 구성원은 질문 글을 작성할 수 있다. 질문 글의 공개 범위가 전체 공개인 경우는 강좌 구성원 모두 조회 가능, 교육자 공개로 설정할 경우 강좌의 교육자만 조회 가능하다.

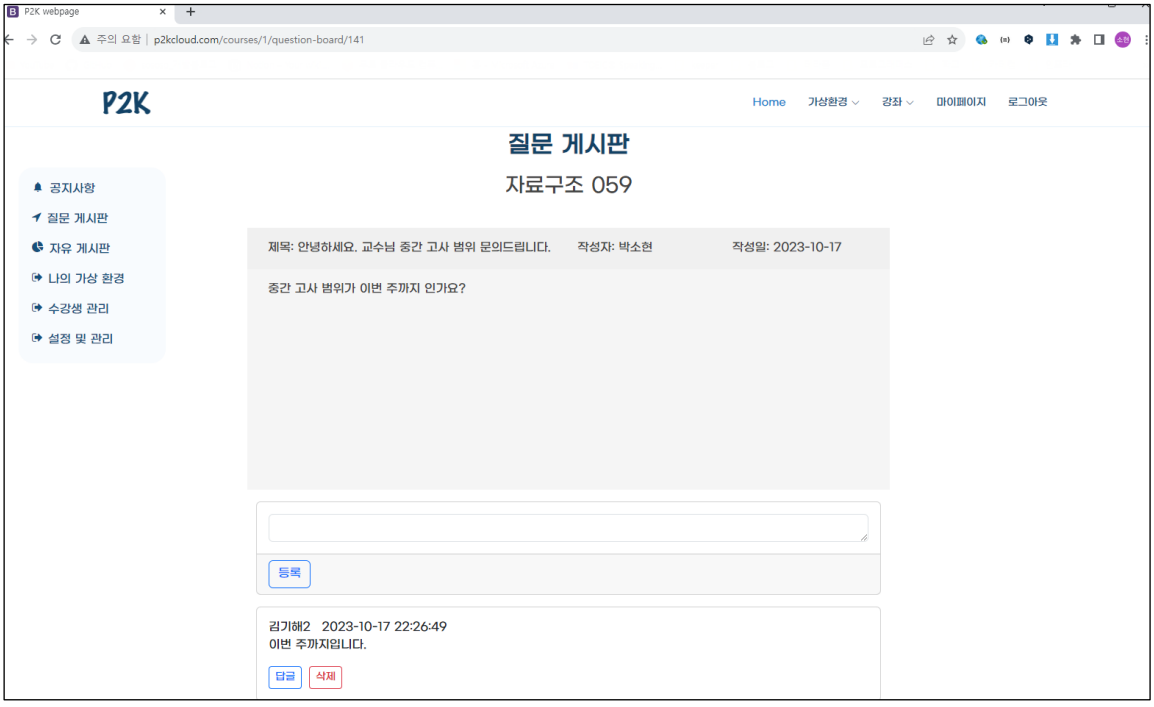

그림 102. 수강자/교육자 - 질문글 수정/삭제 및 댓글 작성 (교육자 화면 예시)

질문 글은 해당 글 작성자만 수정/삭제가 가능하다. 댓글은 강좌의 구성원 모두 작성 가능하다.

3) 자유 게시판

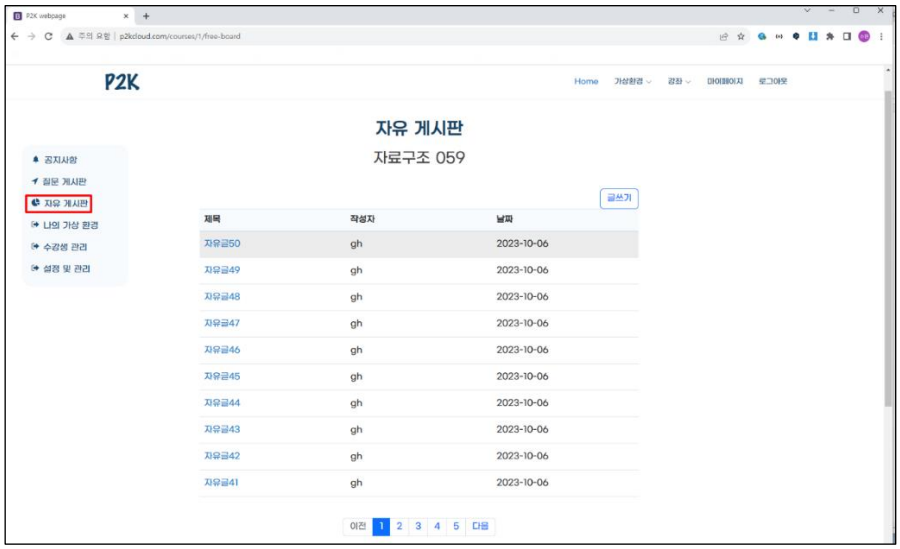

그림 103. 수강자/교육자 - 자유 게시판 목록

자유 게시판의 글은 강좌 구성원 모두 작성할 수 있다. 자유 글의 공개 범위를 전체 공개로 설정할 경우 강좌 구성원 모두 조회할 수 있고, 공개 범위를 교육자 공개로 설정할 경우 강좌의 교육자만 조회할 수 있다. 자유 글 또한, 해당 글의 작성자만 수정, 삭제가 가능하다. 댓글은 강좌의 구성원 모두 작성 가능하다.

자유 게시판의 경우, 질문 게시판과 방법이 동일하기 때문에 자세한 내용은 생략한다.

4) 게시글

[수정]

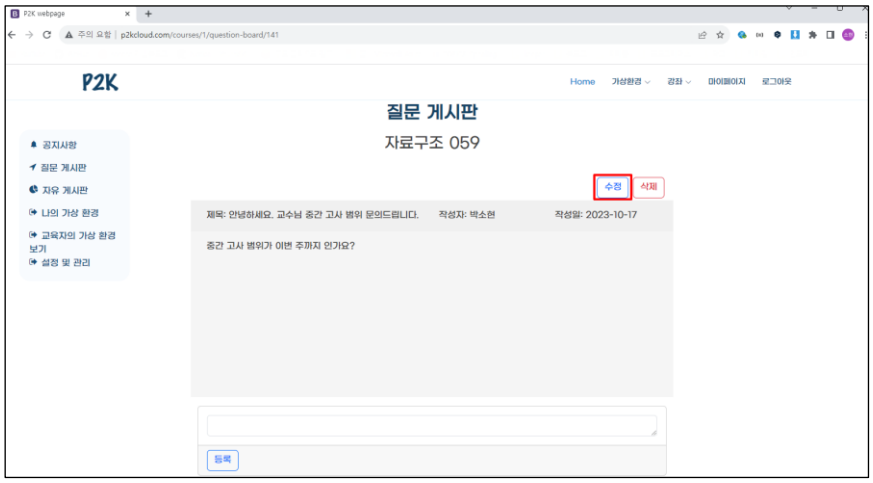

그림 104. 수강자/교육자 - 게시글 수정 (질문 게시판 예시)

게시글에서 [수정]을 눌러 게시글 수정 페이지로 이동한다.

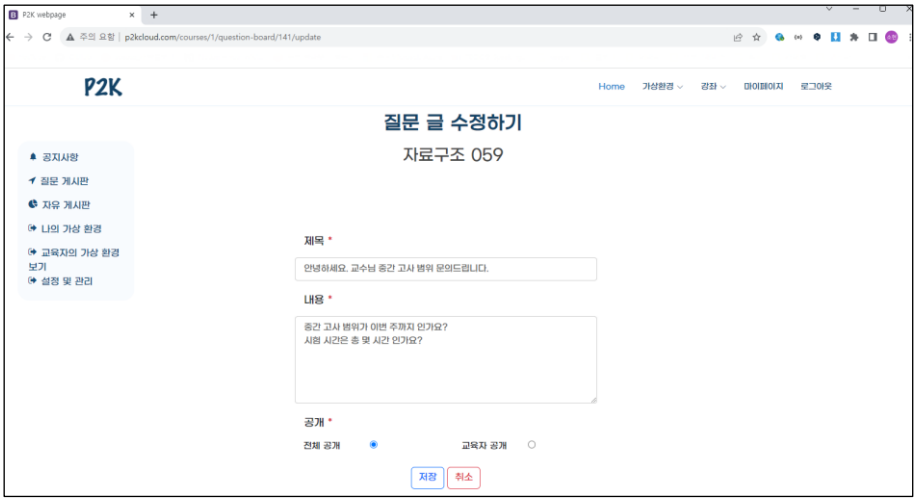

그림 105. 수강자/교육자 - 게시글 수정 (질문 게시판 예시)

글을 수정한 후 [저장]을 누르면 수정 사항이 반영된다.

[삭제]

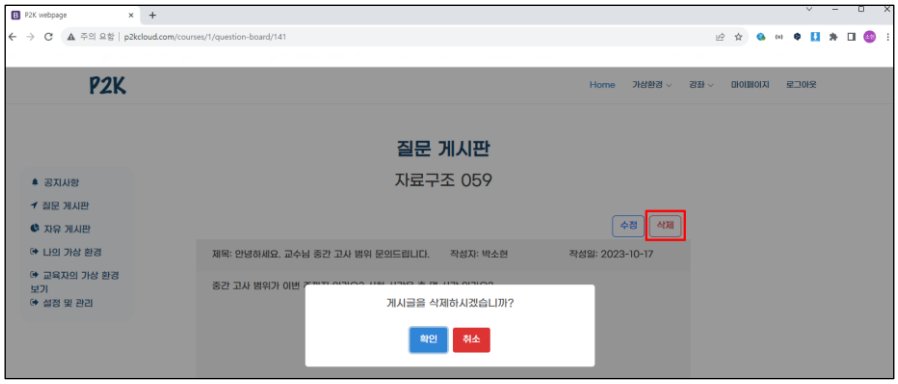

그림 106. 수강자/교육자 - 게시글 삭제 모달창 (질문 게시판 예시)

게시글에서 [삭제]을 누르면 게시글 삭제 모달창이 뜬다.

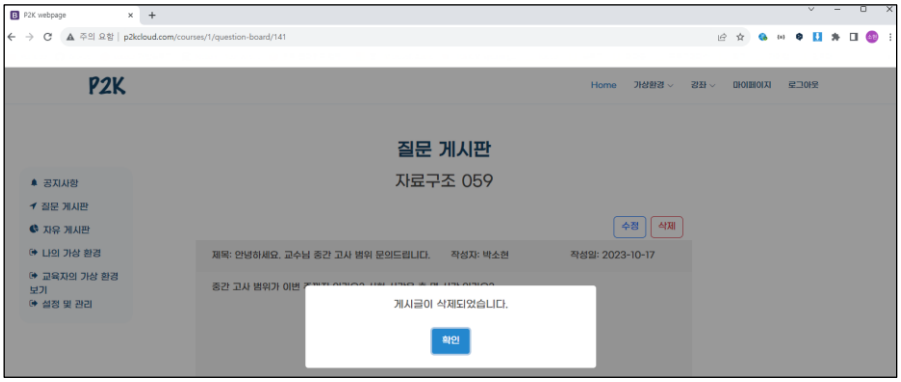

그림 107. 수강자/교육자 - 게시글 삭제 확인 모달창 (질문 게시판 예시)

게시글 삭제 확인 모달창에서 [확인]을 누르면 해당 게시글의 글과 댓글이 삭제된다.

5) 댓글

[작성]

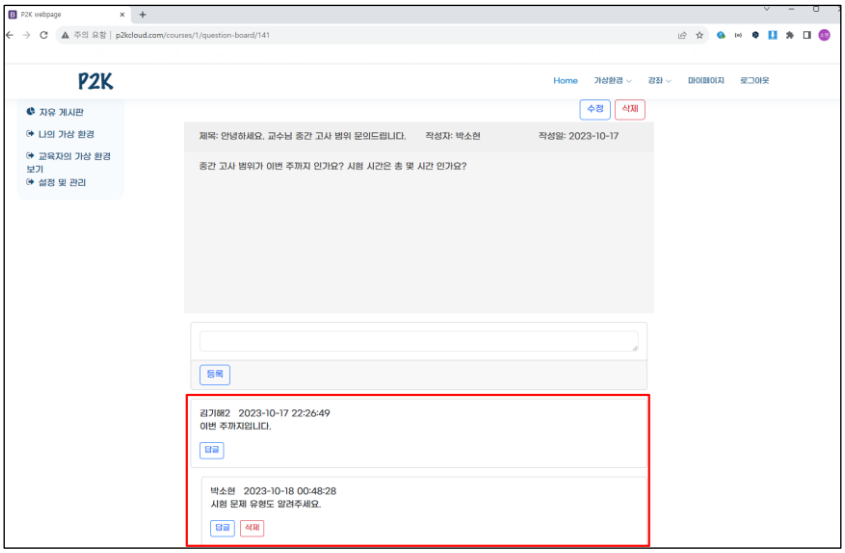

그림 108. 수강자/교육자 - 댓글 작성

게시글에서 댓글을 작성한다. 댓글에 대한 댓글(대댓글)을 계층적으로 작성할 수 있다. 대댓글은 최대 15 계층까지 생성될 수 있다. 댓글과 대댓글을 구분하지 않고 한 페이지 당 10개의 댓글이 조회되도록 구현했다.

[삭제]

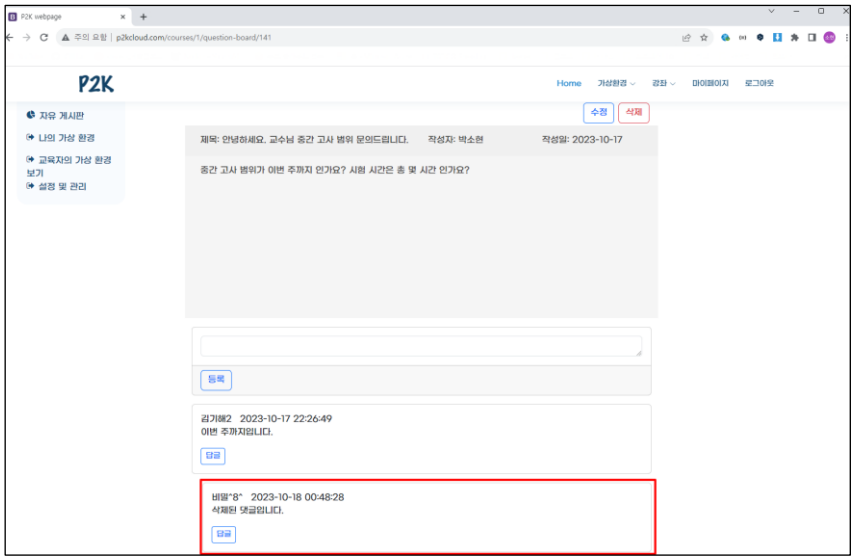

그림 109. 수강자/교육자 - 댓글 삭제

게시글에서 댓글의 [삭제]을 눌러 댓글을 삭제한다. 댓글을 삭제하지 않고 작성자 및 내용을 비공개로 변경한다. 삭제 버튼은 댓글이 삭제된 이후 사라진다.

3.5. 관리자 서비스

3.5.1. 관리자 서비스 제공

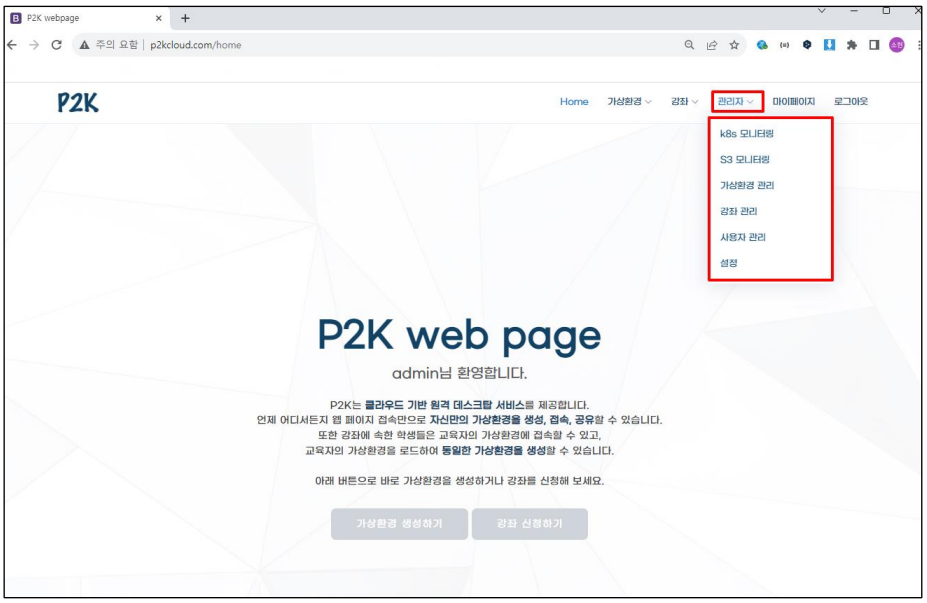

그림 110. 관리자 - 메인 페이지

관리자의 메인 페이지이다. 관리자는 다른 사용자들과는 달리 [관리자]탭이 존재하여 서비스를 전체적으로 관리할 수 있다.

3.5.2. 관리자 서비스 구성도

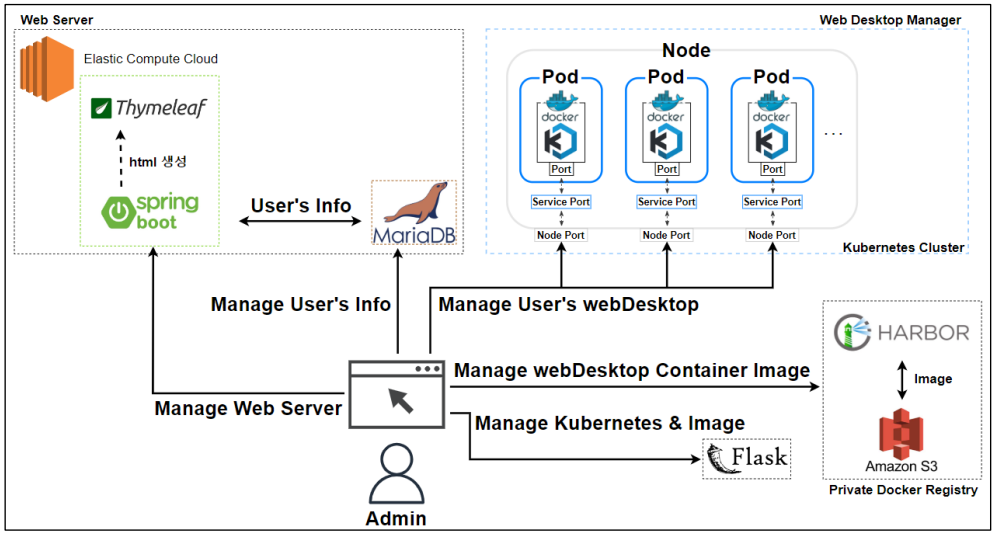

그림 111. 관리자 서비스 구성도

관리자는 서비스를 유지 보수하고, 서비스 사용자를 관리한다. 관리자가 접근할 수 있는 서비스는 다음과 같다.

1. **Web Server**
    - SpringBoot 웹 애플리케이션과 MariaDB를 관리하여 사용자의 웹 사용 서비스와 사용자 정보를 관리
2. **사용자의 Web Desktop**
    - 사용자의 가상환경에 직접 접근하여 관리, 사용자 지원 제공
    - 사용자의 가상환경을 모니터링, 문제가 발생할 경우 조치를 취해 사용자의 요구에 대응
3. **Kubernetes & Web Desktop Image Manager**
    - Kubernetes 서버 관리
    - Kubernetes Scaling을 통한 성능 최적화
    - 사용자의 이미지를 관리하며 사용자의 가상환경을 제어
4. **Private Docker Registry**
    - Harbor와 S3에 접근 가능, 사용자의 Web Desktop Image를 관리

3.5.3. k8s 모니터링

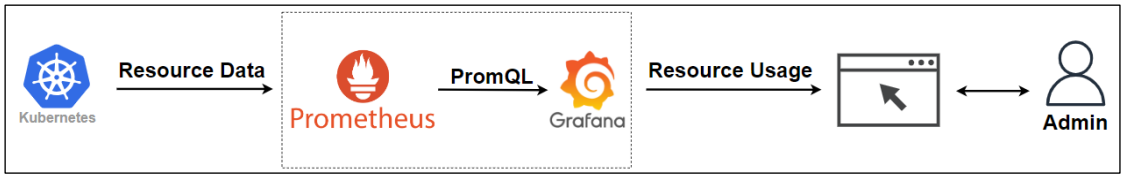

그림 112. k8s 모니터링 구성도

k8s를 모니터링하는 그라파나(Grafana)를 웹 페이지에 임베딩한다.

그림 113. 관리자 모니터링 기능

이를 통해 관리자는 편리하게 k8s 리소스를 모니터링할 수 있다. Grafana를 이용해 모니터링 할 수 있는 리소스는 다음과 같다.

리소스	모니터링
노드(Node) 리소스	CPU 사용량 및 부하
	메모리 사용량 및 부하
	디스크 사용량 및 부하
	네트워크 입출력 (네트워크 대역폭, 패킷 송수신 등)
파드(Pod) 리소스	CPU 및 메모리 사용량
	네트워크 트래픽 및 지연
	파드 상태 (활성, 비정상, 중지 등)
서비스(Service) 및 엔드포인트(Endpoint) 정보	서비스의 트래픽
	엔드포인트 상태와 연결 수
Kubernetes 이벤트	클러스터 이벤트 및 상태 변화 (파드 생성, 삭제, 스케일링 등)
Deployment 및 Stateful Set 리소스	Deployment 및 StatefulSet의 Replica 상태 및 업데이트 상태

3.5.4. s3 모니터링

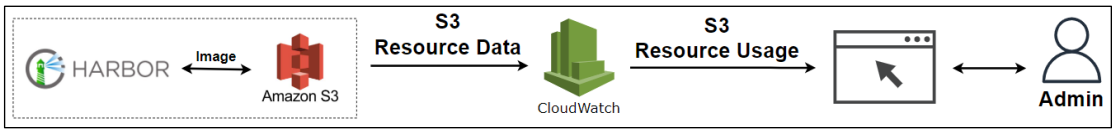

그림 114. S3 모니터링 구성도

관리자는 [관리자] – [s3 모니터링]에서 s3 리소스를 모니터링할 수 있다.

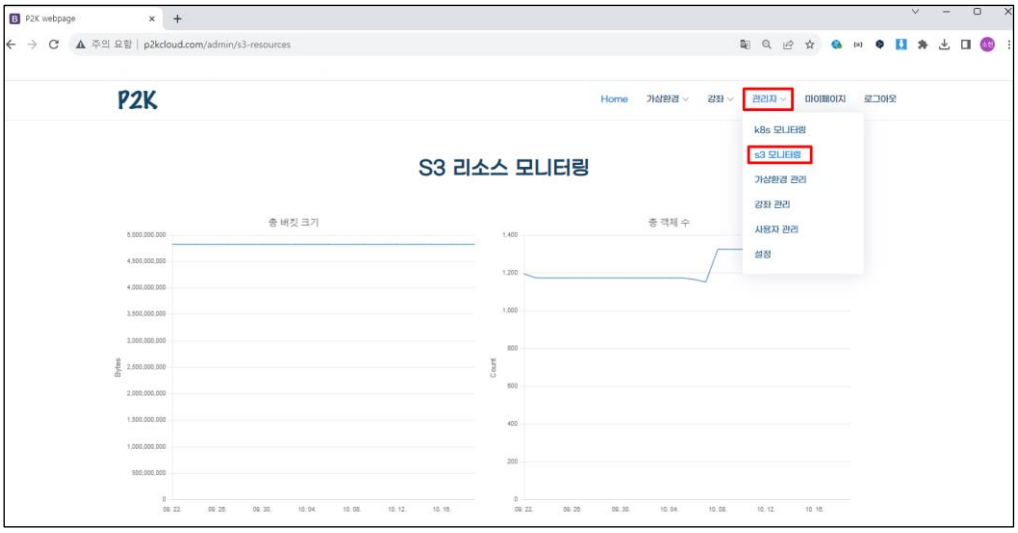

그림 115. 관리자 - S3 모니터링

AWS SDK를 사용하여 Spring Boot 애플리케이션에서 AWS CloudWatch API를 호출해서 CloudWatch로부터 S3 버킷의 메트릭을 주기적으로 가져온다. 현재 설정에 따르면, 지난 2주 동안 최대 100개의 데이터를 수집한다. 모니터링 할 수 있는 버킷 레벨 지표 리소스는 다음과 같다.

메트릭	설명
BucketSizeBytes	버킷 내의 모든 객체의 크기 합계(Bytes)
NumberOfObjects	버킷 내의 객체 수(Count)

조회된 메트릭 데이터의 시간 정보를 담고 있는 리스트인 timestamps와 해당 시간에 대응하는 메트릭 데이터의 값을 담고 있는 리스트인 values를 CloudWatch API를 사용해서 가져온다. timestamps와 values를 Chart.js를 사용해서 웹 페이지에 그래프로 나타낸다.

이러한 메트릭들은 CloudWatch에서 모니터링이 가능하며, S3 버킷의 운영 및 성능 최적화에 도움이 되고 용량 관리나 객체 수의 증가 추세를 파악하는 데 사용된다. 설정된 주기와 데이터 양은 애플리케이션의 요구 사항에 따라 조절 가능하다.

3.5.5. 가상 환경 관리

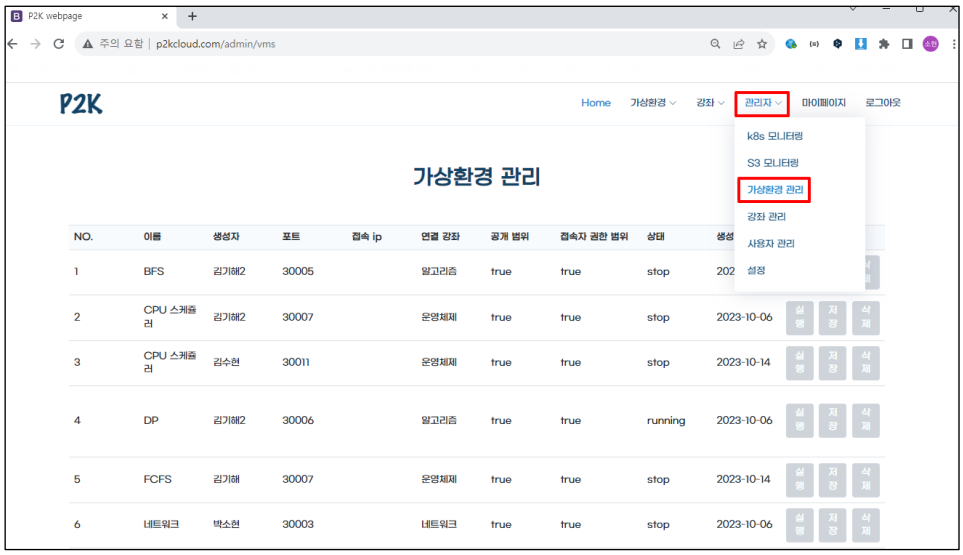

그림 116. 관리자 - 가상환경 관리

관리자는 [관리자] – [가상환경 관리]에서 현재 생성된 모든 가상 환경을 조회할 수 있다. 관리자는 가상 환경 기능인 실행, 접속, 중지, 저장, 삭제를 편리하게 실행할 수 있다.

3.5.6. 강좌 관리

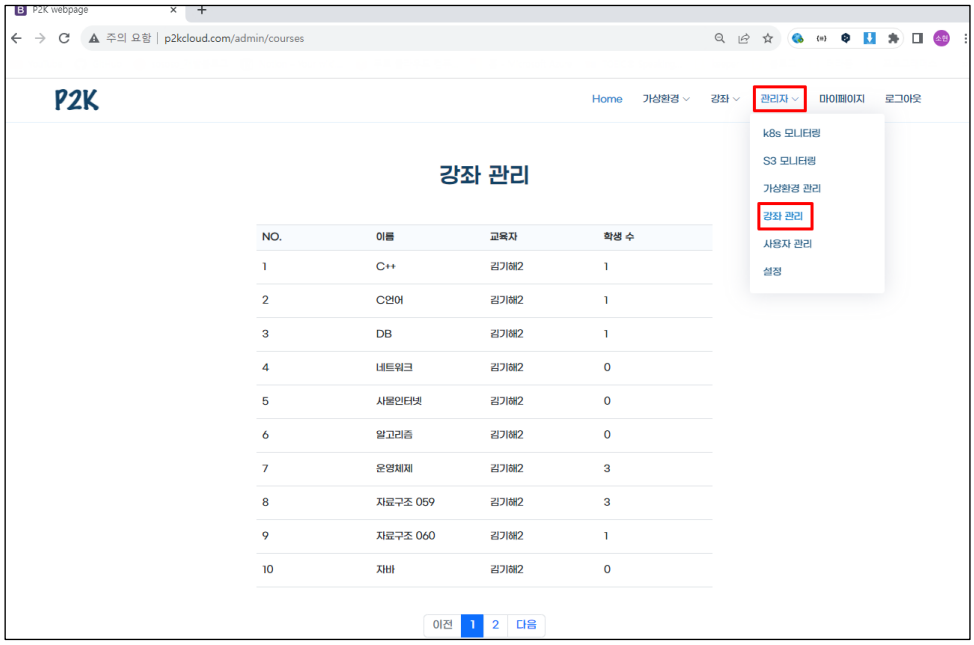

그림 117. 관리자 - 강좌 관리

관리자는 [관리자] – [강좌 관리]에서 현재 생성된 모든 강좌 정보를 조회할 수 있다.

3.5.7. 사용자 관리

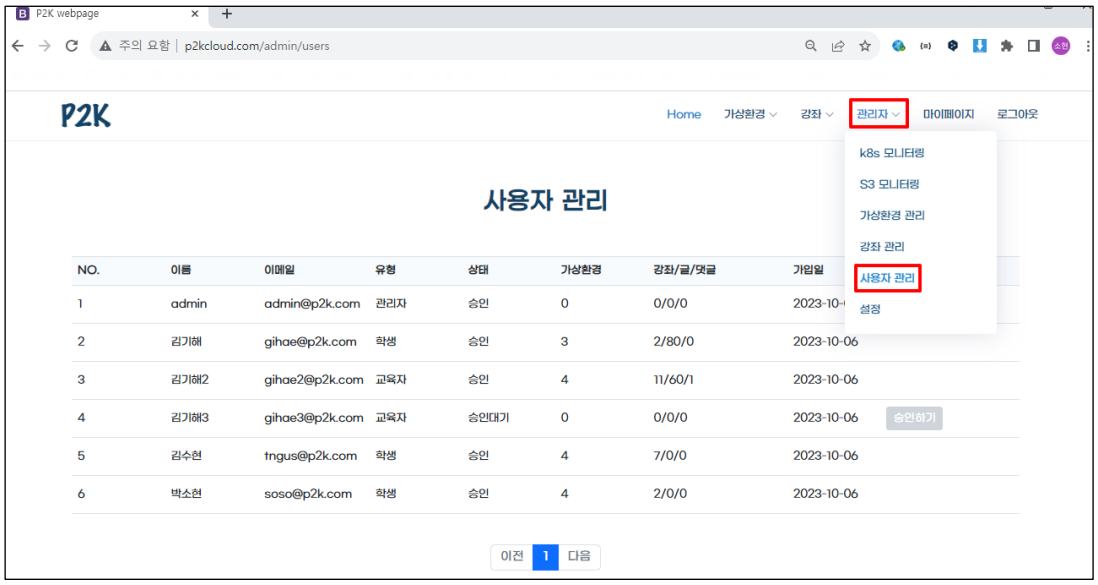

그림 118. 관리자 - 사용자 관리

관리자는 [관리자] – [사용자 관리]에서 현재 가입되어 있는 모든 사용자의 정보를 조회할 수 있다. 관리자의 승인을 받지 않은 교육자는 상태에 '승인대기'가 나타난다. 관리자는 승인되지 않은 교육자를 승인하기 버튼을 통해 승인할 수 있다.

3.5.8. 서비스 설정

그림 119. 관리자 - 설정 페이지

관리자는 [관리자] – [설정]에서 각 사용자의 생성 가능한 가상 환경 최대 개수, 생성 가능한 강좌 최대 개수, 신청 가능한 강좌 최대 개수를 설정할 수 있다. 관리자의 설정 기능은 시스템 전체의 성능에 따라 가상 환경 개수를 조절하며 서비스 안정성에 기여한다.

4. 연구 결과 분석 및 평가

4.1. 언제 어디서든 접근 가능한 데스크탑 환경

학생들은 인터넷 접속만으로 언제 어디서나 접근 가능한 원격 데스크탑을 통해 학습 환경에 손쉽게 접근할 수 있다. 사용자들은 pc 뿐 만 아니라 휴대폰, 태블릿 등의 다양한 기기로 접속한 후 로그인을 하면 자신이 사용하던 데스크탑 환경을 어디서든지 유지하고 사용할 수 있다.

4.2. 웹 기반 서비스 제공

클라우드 기반의 원격 데스크탑 서비스를 웹 서비스의 형태로 제공하여 사용자들이 별도의 애플리케이션 설치나 복잡한 설정 없이도 쉽게 사용할 수 있도록 한다. 이는 플랫폼 독립적인 웹 환경에서 간편한 이용을 가능하게 한다.

4.3. Kubernetes를 통한 효율적이고 안정적인 서비스 제공

Kubernetes를 사용하여 서버 리소스를 효율적으로 운용하고 안정적인 서비스를 제공한다. Kubernetes는 가상 환경을 동적으로 관리하는 auto scaling을 제공해 예기치 못한 트래픽 증가나 장애 상황에 대응해 안정성을 유지하고 중단없이 서비스를 제공한다.

4.4. MSA 구조

서버의 부하를 분산시키기 위해 가상 환경 이미지 관리 서버, 가상 환경 접속 서버, 웹 서버를 분리하여 마이크로서비스 아키텍처(MSA, MicroService Architecture)를 실현한다. 뿐만 아니라 기능 별 독립적인 서버는 추후 서비스의 확장성을 확보할 수 있다.

4.5. 학습 관리 시스템 제공

강좌 신청, 생성을 통한 교육 플랫폼을 제공하여 교육자가 강좌를 효과적으로 관리하고 학생들이 학습 자료와 정보에 쉽게 접근할 수 있다. 게시판과 댓글 기능도 제공해 강좌 커뮤니티를 생성하고 적절한 학습 환경을 제공한다.

4.6. 일관된 학습 환경 제공

기존에는 교육자가 교육에 필요한 환경을 제공하기 위해 압축 파일 또는 설명서를 제공해 학생들이 직접 환경 설정을 해야 했다. 하지만 우리의 서비스에서는 교육가는 가상 환경에서 학습 환경을 구성한 후 학생은 해당 가상 환경을 가져와서 교육자와 동일한 학습 환경을 쉽게 구성한다. 이를 통해 교육자는 교육 컨텐츠를 효과적으로 전달할 수 있으며, 학생마다 다르게 환경을 설정하는 일을 방지하고, 번거롭고 복잡한 환경 설정 과정을 크게 간소화한다.

4.7. 가상환경 접속자 권한 제공

사용자는 다른 사람의 가상환경 검색 또는 강좌에 속한 다른 수강생들의 가상 환경을 조회해 접속자로 접속하여 둘러보거나, 한 가상환경에서 함께 작업할 수 있다. 이는 화면 공유를 통한 원활한 팀 활동을 가능하게 하며, 더불어 수강생들의 가상환경 접속을 통한 교육자의 효과적인 학습 지도 및 관리 환경을 제공한다.

4.8. 안전한 가상환경 관리

harbor와 S3을 사용한 private docker registry를 구축하여 사용자들의 가상 환경인 docker image를 안전하게 관리한다. 이는 관리자가 소유하고 운영하는 private 공간에 docker image가 저장되고 관리되어 사용자들의 가상 환경 관리에 보안과 안정성을 강화한다.

4.9. 관리자의 실시간 대응

kubernetes로 구축한 서버를 프로메테우스(Prometheus)와 그라파나(Grafana)를 활용하여 모니터링하며 동시에 private docker registry에 사용된 S3를 AWS CloudWatch로 모니터링한다. 관리자는 관리자 페이지에서 모니터링과 자료를 제공받을 수 있다. 이를 통해 시스템의 성능을 지속적으로 확인하고 실시간으로 대응이 가능해 안정적인 서비스 운영에 기여한다.

5. 결론 및 향후 연구 방향

우리는 학생들에게 클라우드 기반 원격 데스크탑 서비스를 제공함으로써 웹 페이지만으로 언제 어디서든 자신의 가상환경에 접속할 수 있고, 자신의 pc가 아니여도 작업을 연속적으로 수행할 수 있다. 이는 뛰어난 접근성과 물리적 기기의 제약을 벗어난 작업의 연속성을 제공하여 코로나 19 이후의 사회적 변화와 요구사항에 대응한다. 뿐만 아니라 교육 플랫폼을 제공하여 수강생 관리 및 강좌 커뮤니티 제공, 동일한 학습 환경 일괄 배포, 간편한 학습 환경 구축 및 생성, 가상환경 동시 접속을 통한 실습 또는 팀 활동의 편의성 등을 제공한다. 이런 여러 가지 학습 관리 기능들을 제공해 독립적이고 안정된 학습 환경을 구축하고, 교육자가 학습 지도와 관리를 효과적으로 진행할 수 있게 한다.

현재는 사용자당 최대 3개의 가상환경을 생성할 수 있게 하여, 클라우드 자원을 관리하고 있다. 본 과제를 더 확장하여, 충분한 자원을 확보할 수 있다면 공공기관 또는 공공사업에서 진행하는 클라우드 전환 사업에 맞게 사회적, 경제적 약자를 위한 학습 관리 시스템으로 배포 가능성을 기대하고 있다. 또한 학생과 교육자를 위한 교육 플랫폼에서 확장하여, 재택 근무나 출장 등 작업 환경이 수시로 변경되는 직장인들에게도 근무의 연속성과 물리적 공간의 제약을 극복한 최적화된 서비스를 제공할 수 있을 거라 기대한다.

6. 역할 분담

팀원	역할
박소현	**인프라 설계 및 구축** - 전체 구성도 설계, 세부 구성 설계, 멀티 클라우드 적용 - 도메인 설계 및 구축 - 서비스 배포 **Private Docker Registry 설계 및 구축** - Harbor 구축 및 S3 연동 - Private Docker Registry AutoScaling 적용 - 모니터링 설정 **Web Desktop Manager 설계 및 구축** - Kubernetes 설계 및 적용, Kubernetes 환경 AutoScaling 적용 - Pod 생성 함수 설계 및 구현 - Pod 노출 방식 결정 및 노출 방법 구현 - 모니터링 설정 **Kubernetes & Web Desktop Image Manger 설계 및 구축** - Flask 구축, Container Base Image 선정, Image 관리 방법 수정 - Kubernetes 서버 통신 방법 설계 및 구현 - Flask와 Kubernetes 사이의 통신 방법 구현 (가상환경 중지, 저장)
김기해	**UI 설계** - 요구 사항 기능서를 바탕으로 한 UI 설계서 작성, 서비스 용어 정리 **Back-End** - 강좌 API, 게시판 API, 관리자 API 구현, 모니터링 기능 **Front-End** - 강좌, 게시판, 관리자 페이지 구현 **원격 접속** - noVNC, Websockify를 적용한 로컬 가상환경 통신 테스트
김수현	**설계** - 데이터 베이스 설계 및 schema 작성 - 요구사항 기능서 작성, 서비스 용어 정리 **Front-End** - 로그인, 회원 가입, 메인, 사용자별 가상 환경 관리 페이지 구현 **Back-End** - 로그인, 회원 가입 구현, 가상환경 API 구현 - 웹 서버와 가상환경 서버 API 통신 구현 - Flask 내의 Container Image 관리 구현 - 가상환경 접속 범위, 접속자 관리 기능

7. 개발 일정

구분	작업 일정																						
	5월		6월					7월				8월					9월				10월		
	3	4	1	2	3	4	5	1	2	3	4	1	2	3	4	5	1	2	3	4	1	2	3
환경 및 요구사항 분석																							
로컬 테스트																							
전체 구성도 설계																							
도커 레지스트리 구축																							
UI 설계 및 DB 설계																							
UI 개발																							
사용자 API 구현																							
중간 보고서																							
강좌 API 구현																							
게시판 API 구현																							
가상 환경 API 구현																							
관리자 API 구현																							
플라스크 서버 구축																							
쿠버네티스 서버 구축																							
모니터링																							
웹 서버 구축																							
배포																							
최종 보고서																							

8. 참고 문헌

[1] KDB Future Strategy Research Institute, Future Strategy Development Department, "Changes in the Trend of Non-face-to-face Education after COVID-19," pp. 2, May. 2020.

[2] Ahn, D., "Gyeongnam Office of Education Distributes Smart Terminals Worth 150 Billion Won to Students," JoongAng Ilbo, 2022.

[3] Ahn, H., "Public Cloud Transition: A Boon for Government Services," Electronic Times, 2022, p. 18.

[4] OpsNow, "EdTech and Cloud," [Online]. Available: https://www.opsnow.com/%EC%97%90%EB%93%80%ED%85%8C%ED%81%AC%EC%99%80-%ED%81%B4%EB%9D%BC%EC%9A%B0%EB%93%9C/ (Accessed Sep. 10, 2023)