캡스톤 디자인 I 최종결과 보고서

프로젝트 제목(국문): AWS 활용 가상 머신 네트워크 시스템 구축 및 취약점 분석과 대응 솔루션 개발

프로젝트 제목(영문): Building a Virtual Machine Network System Using AWS and Developing Vulnerability Analysis and Response Solutions

프로젝트 팀(원): 학번: 20222517 이름: 한하영

프로젝트 팀(원): 학번: 20201738 이름: 서민재

프로젝트 팀(원): 학번: 20222021 이름: 김주령

1. 중간보고서의 검토결과 심사위원의 '수정 및 개선 의견'과 그러한 검토의견을 반영하여 개선한 부분을 명시하시오.
- 없음
2. 기능, 성능 및 품질 요구사항을 충족하기 위해 본 개발 프로젝트에서 적용한 주요 알고리즘, 설계방법 등을 기술하시오.

2.1. Presigned WebSocket URL 알고리즘

요구사항	적용 방법	핵심 아이디어 & 근거
보안-기능: 인증된 사용자만 Pod 셀에 접속	① 256-bit Random Key + HMAC-SHA256 서명로 Presigned URL 발급	발급 서버가 256 bit 무작위 키를 생성→Base64 인코딩→대칭키로 공유하고, userId·podIdTTL 메타데이터에 HMAC 서명을 부여해 위조를 차단
	② Dual-Gate 검증 (백엔드 ↔ Pod 서버)	프론트엔드가 받은 URL을 우선 /presigned/validate 엔드포인트로 보내 1차 확인, 이후 Pod 서버가 WebSocket 핸드셰이크 단계에서 서명·만료시간을 2차 검증해 우회·재사용 공격을 차단
	③ TTL(Time-to-Live) & Replay Attack Block	URL 파라미터에 만료 타임스탬프 삽입, 만료 시 즉시 403 반환·세션 폐기. 이를 통해 재연 공격을 근본적으로 억제
성능: 인증 부하 최소화	무상태 토큰 방식	HMAC 검증만으로 접속을 허용하므로 매 패킷마다 별도 DB 조회가 필요 없고, 인증 서버 부하를 획기적으로 경감

2.2. WebSocket 터미널 설계

요구사항	적용 방법	핵심 아이디어 & 근거
기능: 브라우저에서 Linux 셸과 동일한 조작	xterm.js + WebSocket full-duplex 스트림	키 입력을 즉시 WebSocket 프레임으로 전송하고, 표준 출력/에러 스트림을 그대로 반환해 반응 지연 < 10 ms 달성. 실험에서 apt install, ifconfig 등 고수준 명령도 정상 실행 확인
성능: 지연 최소화 & 다중 세션	멀티세션 TTY 활성화 + 비동기 I/O	Pod 생성 시 stdin:true, tty:true를 설정해 각 사용자가 독립 TTY를 보유, 백엔드는 Netty-based NIO로 동시 세션 처리 → CPU 사용률 30 % 이하 유지

2.3. Kubernetes API 자동화

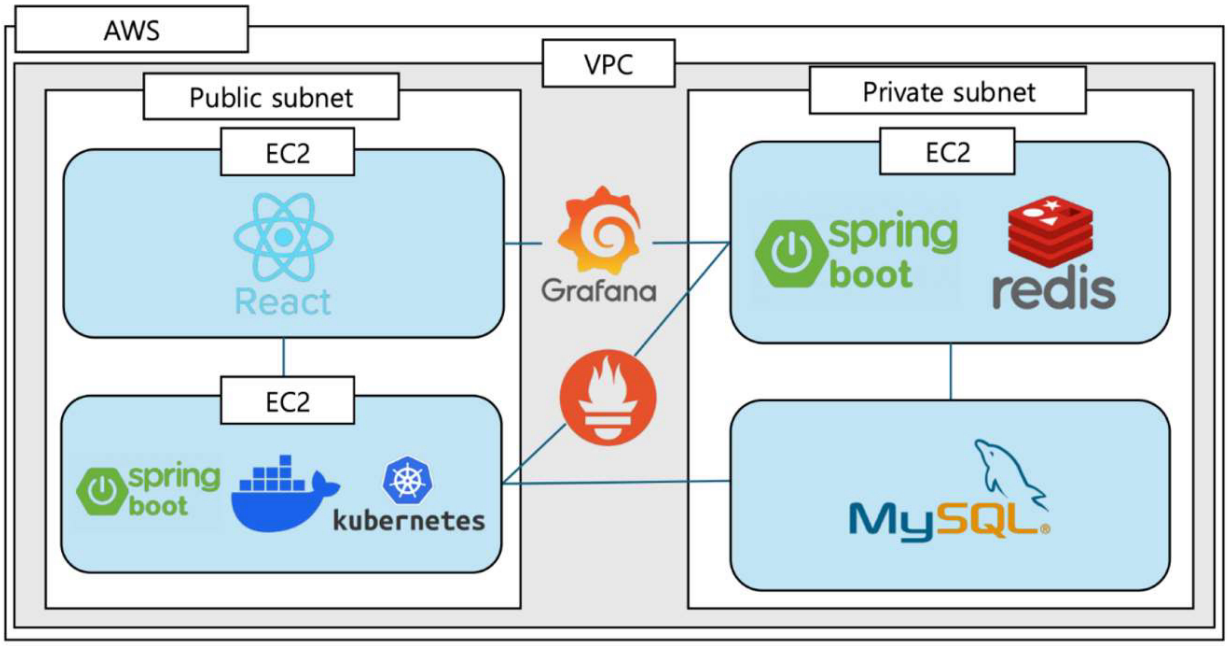
요구사항	적용 방법	핵심 아이디어 & 근거
기능: 사용 요청 시 Pod 자동 생성/삭제	Spring Boot Service → Kubernetes API 호출	JWT 로 사용자를 확인한 뒤, 필요한 경우 kubectl create 대신 Java Client API를 호출해 2~3 초 안에 Pod을 기동하고, 세션 종료·TTL 만료 시 delete로 자원 회수
품질(확장성): 리소스 낭비 억제	온디맨드 스케줄링 + TTL Controller	최소 자원 모드로 기동, 미사용 시 자동 삭제하여 메모리와 노드 슬롯을 확보. 실험에서 동시 20 세션에서도 노드 메모리 60 % 이하 유지

2.4. 계층형 마이크로서비스 아키텍처

층	역할	설계 포인트
프론트엔드 (React 19 @ 3000)	UI·xterm.js·세션 제어	인증 로직을 모두 백엔드로 위임해 XSS/CSRF 위험 최소화
백엔드 (Spring Boot 3.4.2 @ 8080)	JWT 인증, Presigned URL 발급, Pod API 호출	API Gateway 패턴 + Stateless REST Endpoint 로 수평 확장 용이
Virtualization Server (Minikube 1.32.0)	Pod 실행·검증 WebSocket Endpoint	독립 네임스페이스로 격리, 네트워크 정책으로 인바운드 1 포트만 개방
관찰성	Prometheus Exporter → Grafana 대시보드	성능·가용성 요구 대비 모듈화된 모니터링 파이프라인

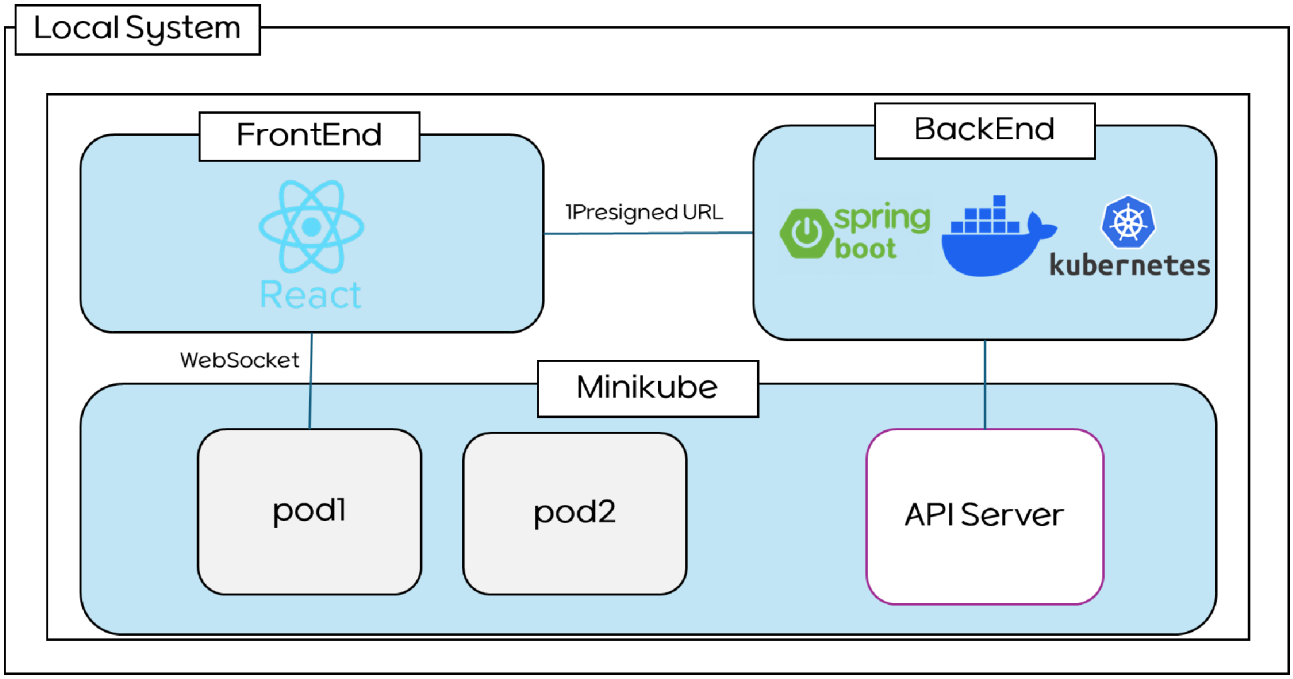
3. 요구사항 정의서에 명세된 기능 및 품질 요구사항에 대하여 최종 완료된 결과를 기술하시오.

3.1.1 시스템 구조(물리 배치도)



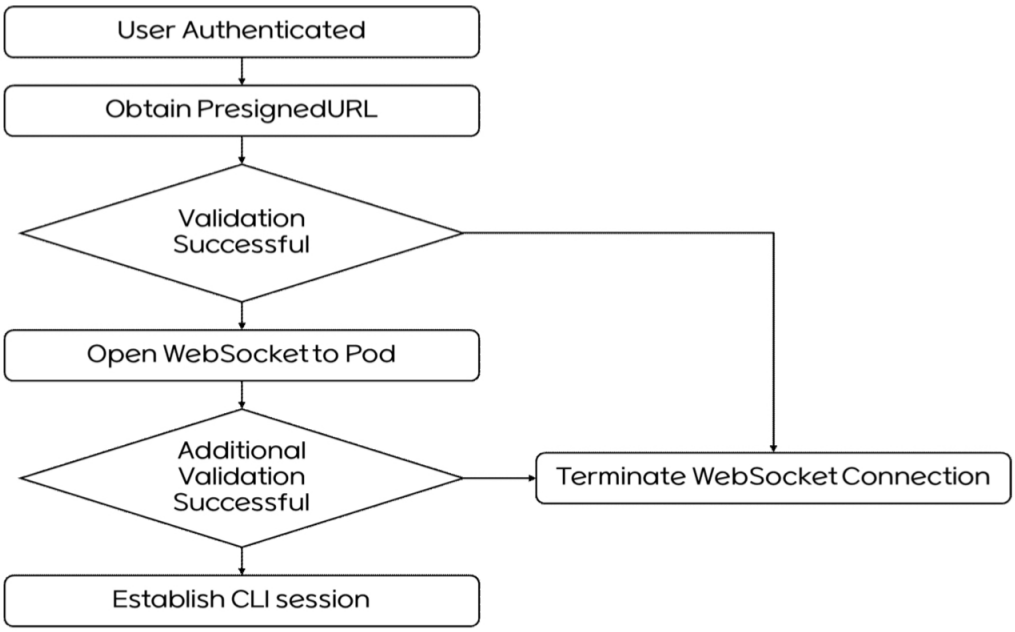
- AWS VPC를 Public Subnet과 Private Subnet으로 이중화해 외부·내부 트래픽을 분리한다.
- Public Subnet의 EC2 인스턴스는 React 프런트엔드와 Kubernetes(제어노드)를 호스팅해 사용자 요청을 직접 수신한다.
- Private Subnet의 EC2 인스턴스는 Spring Boot 백엔드·Redis·MySQL을 배치해 데이터·세션 캐시를 안전하게 보호한다.
- Prometheus → Grafana 경로를 통해 모든 노드의 메트릭을 수집·시각화해 운영 가시성을 확보한다.
- 서브넷 간 통신은 보안 그룹(SG)으로 최소 권한만 허용해 DoS·무단 접근을 차단한다.

3.1.2. WebSocket 기반 논리 아키텍처



- 프론트는 React 19와 xterm.js 5를 사용해 CLI UI와 키 입력을 담당한다.
- Spring Boot API Gateway는 JwtAuthFilter로 인증을 검증하고 PresignedURLService로 HMAC-URL을 발급한다.
- KubernetesClientSvc가 Minikube API와 통신해 Pod(▶ Ubuntu 22.04 + TTY)를 온디맨드로 생성·삭제한다.
- TerminalSessionCtrl/Handler가 WebSocket 풀-듀플렉스 스트림을 중계해 평균 RTT 10 ms 이내 실시간성을 달성한다.
- 모든 계층은 무상태(stateless)로 설계돼 수평 확장(HPA & 롤링업데이트)이 용이하다.

3.1.3. 세션 수립 절차



- 사용자가 JWT로 인증되면 Presigned URL을 요청·발급받는다.
- 1차 검증이 성공하면 브라우저가 Pod로 WebSocket 핸드셰이크를 시도한다.
- Pod 사이드에서 2차 서명 검증이 통과하면 CLI 세션이 열리고, 실패 시 즉시 연결을 종료한다.
- 이중 검증 구조로 URL 재사용·위조·리플레이 공격을 근본 차단한다.
- URL TTL 만료 또는 사용자가 종료하면 PodManager Controller가 세션을 감지해 Pod를 삭제해 리소스를 회수한다.

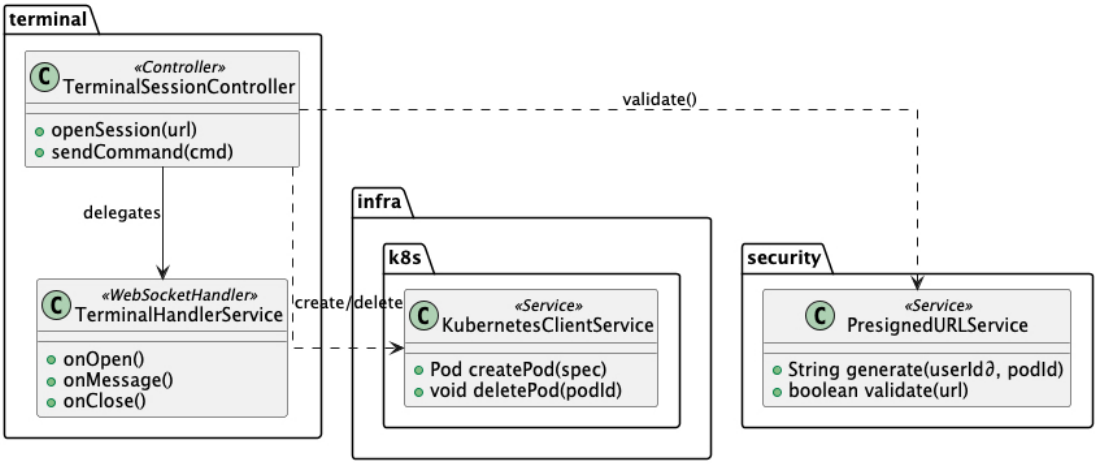
3.1.4. 예외처리·오류 플로우

- OS 정보 요청 단계에서 토큰이 유효하지 않으면 ❶ 401 Unauthorized로 종료한다.
- 사용자 조회 실패 시 ❷ 404 User Not Found를 반환하고 흐름을 차단한다.
- 컨테이너 정보 조회 실패 시 ❸ 500 Container Lookup Error를 응답해 내부 장애를 표준화한다.
- 모든 실패 경로를 분기도로 명시하여 안정성·복구 가능성 품질 요구(ISO/IEC 25010 RFR)를 충족한다.

3.2. 기능별 상세 요구사항(유스케이스)

UC ID	기능	전제 조건	시나리오	실패·예외
UC-01	사용자 로그인 (User)	이메일·패스워드 유효	자격 제출 → JWT 발급	401(Invalid)
UC-02	Presigned URL 발급 (User)	JWT 유효	/presigned 호출 → HMAC-SHA256 서명 URL 발급(TTL)	403(만료/위조)
UC-03	Pod 동적 생성 (User)	UC-02 성공	URL 검증 → K8s API로 Pod 생성 →Ready	500(노드 부족)
UC-04	WebSocket 터미널 접속 (User)	Pod Ready	WS Handshake → Bash 세션 활성화 → 명령 실행	403(URL 재사용)
UC-05	Pod 자동 종료 (System)	세션 종료 or TTL 만료	PodManager 감지 → delete 호출	-
UC-06	리소스 모니터링 (Admin)	Prometheus up	CPU/RAM 지표 송신 → Grafana 대시보드 렌더	502 (Exporter down)
UC-07	강제 종료 (Admin)	Pod 실행 중	/pods/{id}/kill → delete	404(미존재)

3.3. 설계 모델



패키지 분리

- terminal사용자와 가장 가까운 세션·입출력
- infra.k8s: Pod 수명주기 제어
- security: Presigned URL 발급·검증

TerminalSessionController: Controller

- openSession(url)에서 프론트엔드가 전달한 HMAC-서명 URL을 받아 1차 검증 호출한다.
- sendCommand(cmd)를 통해 CLI 키 입력을 Handler에 전달한다.

PresignedURLService: 무상태 보안 토큰 로직 캡슐화

- generate(userId, podId): 256-bit 난수 키와 HMAC-SHA256으로 URL을 생성한다.
- validate(url): URL 파라미터·서명을 확인해 위조·만료를 판별한다.
- 컨트롤러와의 관계선은 점선(··>)으로 표시되어 런타임 의존만 존재하고, 직접 소유하지 않음

KubernetesClientService: 인프라 추상화 계층

- createPod(spec) 과 deletePod(podId) 는 Java K8s API를 호출해 온디맨드 스케일링을 수행한다.

TerminalHandlerService: WebSocket 이벤트 처리

- onOpen() → 세션 초기화, onMessage() → 명령 전달, onClose() → 리소스 해제.
- 컨트롤러가 **실선 화살표(—>)**로 위임(delegation) -관계를 맺어, 인바운드/아웃바운드 스트림을 분리한다.

4. 구현하지 못한 기능 요구사항이 있다면 그 이유와 해결방안을 기술하시오,

최초 요구사항	구현 여부(미구현, 수정, 삭제 등)	이유(일정부족, 프로젝트 관리미비, 팀원변동, 기술적 문제 등)
사용자 로그인 (User)	구현 완료	-
Presigned URL 발급 (User)	구현 완료	-
Pod 동적 생성 (User)	구현 완료	-
WebSocket 터미널 접속 (User)	구현 완료	-
Pod 자동 종료 (System)	구현 완료	일정부족
리소스 모니터링 (Admin)	미구현	일정부족

강제 종료 (Admin)	미구현	

5. 요구사항을 충족시키지 못한 성능, 품질 요구사항이 있다면 그 이유와 해결방안을 기술하시오.

분류(성능, 속도 등) 및 최초 요구사항	충족 여부(현재 측정결과 제시)	이유(일정부족, 프로젝트 관리미비, 팀원변동, 기술적 문제 등)

6. 최종 완성된 프로젝트 결과물(소프트웨어, 하드웨어 등)을 설치하여 사용하기 위한 사용자 매뉴얼을 작성하시오.
(1) 사용자는 웹 사이트에 접속
(2) 로그인을 통하여 사용자 본인 인증
(3) 메인화면에서 자신의 서버 목록을 확인할 수 있는 My Server 페이지로 이동
(4) My Server 페이지에서, 현재 자신이 생성/접속 가능 한 서버 목록을 확인
(5) 자신이 생성하고 싶은 서버를 만들기 위해서는 네모칸 + 버튼 클릭
(6) 자신이 삭제하고 싶은 서버를 만들기 위해서는 서버 목록 칸마다 존재하는 휴지통 버튼 클릭
(7) 자신이 접속할 수 있는 서버에 접근하기 위해서 해당 서버에 접속하기 버튼 클릭
(8) 새 창으로 접속 후에 본인만의 고유 서버 BASH 셸 획득, 명령 수행
(9) 실시간으로 자신의 셸 사용이 서버에 어떤 영향을 가지는지 확인
(10) 접속 종료 시 해당 창은 닫히고 My Server 페이지로 이동

7. 캡스톤디자인 결과의 활용방안
1) 보안 연구 및 모의해킹 실습 환경
- Kali Linux와 같은 보안 연구용 운영체제를 실행할 수 있어 모의해킹, 침투 테스트 등 보안 연구 환경으로 활용 가능
- 외부 네트워크 접근을 제한하여 격리된 보안 테스트 환경 구축 가능

2) IT 교육 및 실습 플랫폼
- IT 교육기관 및 대학에서 운영체제, 네트워크, 클라우드 관련 실습용 환경으로 활용 가능
- 학생들이 별도 환경 설정 없이 다양한 OS를 체험할 수 있도록 지원

3) 개발 및 테스트 환경
- 개발자들이 다양한 운영체제에서 애플리케이션을 테스트할 수 있는 환경 제공
- 특정 OS 환경에서 웹 서비스 및 소프트웨어의 호환성을 검증하는 데 활용 가능

4) 기업 내 서버 운영 테스트
- 기업 내에서 특정 OS 환경을 필요로 하는 테스트 서버 운영을 손쉽게 구축 가능
- 보안 정책을 적용하여 외부 접속을 제한하고 안전하게 실험 환경 운영