웹 소켓 기반 웹 터미널을 통한 쿠버네티스 Pod 접속 시스템 설계

서민재§, 한하영§, 김주령§, 이채혁§, 방인규▪†, 김태훈▪§

§국립한밭대학교 컴퓨터공학과, †국립한밭대학교 지능미디어공학과

{20201738, 20222517, 20222021, 20242073}@edu.hanbat.ac.kr, {ikbang, thkim}@hanbat.ac.kr

Architecture Design of a WebSocket-based Web Terminal for Access Kubernetes Pods

Minjae Seo§, Hayeong Han§, Juryeong Kim§, Chaehyeok Lee§, Inkyu Bang▪†, Taehoon Kim▪§

§Department of Computer Engineering, Hanbat National University

†Department of Intelligence Media Engineering, Hanbat National University

요 약

Kubernetes는 현대 클라우드 네이티브 인프라 환경에서 핵심적인 플랫폼으로 부상함에 따라, 컨테이너 기반 시스템에 대한 안전하고 효율적인 접근 방식의 중요성이 점차 강조되고 있다. 본 연구는 WebSocket 기반의 웹 터미널 시스템을 설계·구현하여, 사용자가 웹 브라우저를 통해 Kubernetes Pod의 컨테이너 셸 환경에 실시간으로 접근할 수 있는 구조를 제안한다. Presigned URL 기반의 이중 인증 절차를 통해 인증된 사용자에게 CLI 접근을 허용하며, Pod 내 실시간 명령어 실행 실험을 통해 기능의 안정성과 보안성을 함께 검증하였다.

Ⅰ. 서론

클라우드 네이티브 기술의 확산과 함께 Kubernetes는 현대 애플리케이션 배포 및 인프라 관리를 위한 핵심 플랫폼으로 자리잡았다. 특히 컨테이너 기반 환경의 확장성과 자동화된 오케스트레이션 기능은 다양한 산업에서 Kubernetes 채택을 가속화하고 있다[1].

그러나 실제 운영환경에서는 연결 설정 오류, 단일 세션 구조로 인한 협업의 어려움, 사용자 간 세션 분리가 불가능한 구조 등 다양한 실무적 제약이 발생한다. 특히 Kubernetes에서 기본적으로 제공하는 *kubectl exec* 방식은 로컬 환경 의존성, 플랫폼 간 호환성 문제, 그리고 다중 사용자 미지원 등으로 인해 확장성과 접근성에 한계를 지닌다[2].

이에 본 논문에서는 WebSocket 기반의 실시간 웹 터미널 시스템을 설계·구현하여, 사용자가 웹 브라우저를 통해 Kubernetes Pod의 웹 터미널 환경에 안전하게 접속하고 명령어를 실시간으로 입력하여 확인할 수 있는 구조를 제안한다. Presigned URL 기반 이중 인증 절차를 통해 사용자 인증을 강화하고, 실시간 명령어 스트리밍, 동적 TTY 할당, 멀티세션 지원 등을 통합적으로 제공한다.

Ⅱ. 시스템 구현

1. 시스템 준비

본 실험은 WebSocket을 이용하여 Kubernetes Pod에 웹 터미널로 접속하는 기능을 구현하고 검증하기 위해, Apple M3 칩이 탑재된 MacBook에서 로컬 환경을 구성하여 진행되었다.

프론트엔드는 React 프레임워크를 이용해 개발하였고, 웹 터미널 UI는 xterm.js 모듈을 활용하여 구현하였다. 백엔드는 Spring Boot 프레임워크로 개발하였고, Kubernetes 클러스터는 Parallels 드라이버를 이용한 Minikube 환경에서 실행하였다. 테스트에 사용된 Pod는 Ubuntu 22.04 LTS 이미지를 기반으로 생성되었으며, 실시간 터미널 상호작용을 위해 tty 및 stdin 기능을 활성화하였다.

표 1 실험 환경 구성 요약

구분	항목	내용
Hardware	CPU	Apple M3(8-core CPU)
	RAM	16GB
Software	OS	macOS Sequoia 15.4.1
	Frontend	React 19.0.0
		xterm.js 5.3.0
	Backend	Spring Boot 3.4.2
	Kubernetes Platform	Minikube(Parallels driver)
Pod Spec	Kubernetes	1.32.0
	OS	Ubuntu 22.04 LTS
	Envionment	tty, stdin 활성화

2. 시스템 구조

본 시스템은 로컬 환경에서 구성되었으며, React는 3000번 포트, Spring Boot는 8080번 포트에서 각각 구동된다. Kubernetes 클러스터는 Minikube를 통해 내부 IP 주소 10.211.55.6에서 구동된다. 인증을

▪ Corresponding Authors: Inkyu Bang (ikbang@hanbat.ac.kr), Taehoon Kim (thkim@hanbat.ac.kr)

완료한 사용자는 Pod를 동적으로 생성하거나 삭제할 수 있으며, 생성된 Pod에 대해 접근할 수 있는 권한이 부여된다. 그림 1은 전체적인 시스템 아키텍처 구조를 시각적으로 나타낸 것이다.

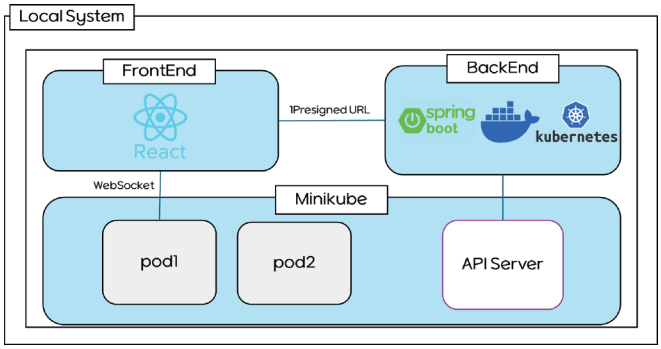

그림 1. WebSocket 기반 시스템 아키텍처

3. 시스템 시나리오

그림 2는 시스템 시나리오를 순서도로 나타낸 것이다. 사용자는 로그인 절차를 통해 인증이 완료되면 자신만의 Pod에 접속할 수 있는 권한이 부여된다. Pod에 접속하기 위해 사용자는 백엔드로부터 Presigned URL을 전달받고, 이를 */presigned/validate* 엔드포인트에서 1차 검증한다. 검증이 성공하면 해당 Presigned URL로부터 Pod 정보를 추출하고 WebSocket 연결을 생성하며, 이 과정에서 인증 우회 공격을 방지하기 위한 2차 Presigned URL 검증이 추가로 수행된다. 검증에 실패한 경우, WebSocket 연결은 즉시 종료되며, 세션도 함께 소멸한다. 반면, 모든 검증을 통과하면 사용자는 WebSocket을 통해 Pod의 bash 셸 환경에 연결되고, 프론트엔드의 xterm.js UI을 통해 웹 기반 터미널에서 실시간으로 명령어를 입력하고 그 결과를 확인할 수 있다. 이러한 기능들은 React, Spring Boot, Minikube 기반의 시스템 구조 위에서 구현되었으며, 사용자 요청에 따라 Pod를 동적으로 생성하고 실시간 터미널 접속을 제공하도록 설계되었다.

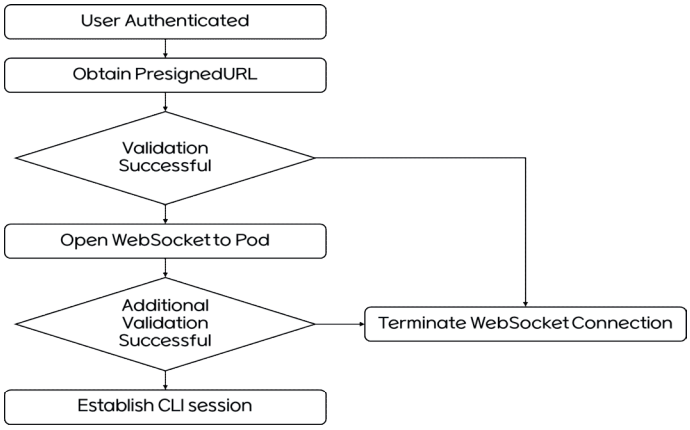

그림 2 시스템 시나리오 순서도

Ⅲ. 테스트 및 결과

그림 3은 Presigned URL을 통해 인증에 성공한 사용자가 WebSocket을 이용해 생성된 Pod의 터미널 환경에 접속하여 명령어를 실행한 결과(좌)와, Presigned URL 인증에 실패했을 경우 출력되는 접속 불가 화면(우)을 보여준다. 인증에 성공한 경우, 사용자는 웹 터미널 환경에서 ls, apt install net-tools, ifconfig 등의 명령어를 입력하였으며, 이는 실제 Ubuntu 22.04 운영체제에서와 동일하게 정상 동작하였다.

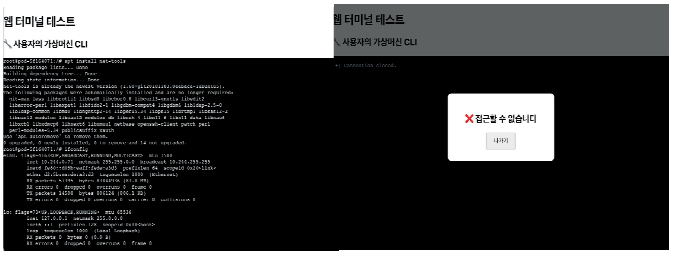

그림 3 웹 터미널 접속 성공 화면(좌)과 실패 화면(우)

이를 통해 사용자가 웹 브라우저 기반의 터미널 UI를 통해 Pod의 터미널 환경에 실시간으로 접속하고, 일반적인 리눅스 명령어를 실행할 수 있음을 확인하였다. 특히, apt와 같은 패키지 관리 도구가 정상 작동하는 것은 네트워크 연결과 셸 환경이 안정적으로 제공되고 있음을 의미하며, 본 시스템의 터미널 기능이 단순한 출력 시뮬레이션이 아니라 실제 시스템 자원에 대한 상호작용이 가능함을 검증한다. 한편, 인증에 실패한 경우에는 WebSocket 연결은 즉시 종료되며, 사용자에게는 접근 불가 메시지가 출력된다. 이와 같은 흐름을 통해 웹 소켓 프로토콜을 활용한 인증–통신 제어가 효과적으로 구현되었음을 확인할 수 있다.

Ⅳ. 결론

본 논문에서는 사용자가 웹 브라우저를 통해 Kubernetes Pod에 실시간으로 접속할 수 있는, WebSocket 기반의 웹 터미널 시스템을 설계하고 구현하였다. Presigned URL 기반의 이중 인증 구조를 적용하여, 인증된 사용자만 명령어 실행 및 패키지 설치 등 Pod 내부 시스템 자원에 대한 접근을 안전하게 제공할 수 있도록 구현하였다. 향후 다중 사용자 지원, 로그 관리, 실제 서버 배포를 진행하여 클라우드 기반 교육 플랫폼 또는 테스트 환경으로의 실용적 확장을 제안한다.

ACKNOWLEDGMENT

본 연구는 국립한밭대학교 공학교육혁신센터 “창의융합형공학인재양성지원사업”, 2025년 과학기술정통신부 및 정보통신기획평가원의 SW중심대학사업의 지원으로 수행되었음(2022-0-01068)

참 고 문 헌

[1] Cloud Native Computing Foundation, “CNCF Survey Report 2020”, Cloud Native Computing Foundation, Nov. 2020.

[2] D, Di Nucci and D. A.Tamburri, “RADON Adoption Handbook”, RADON H2020 Project, Sep. 2021.