

# CodeJam

실시간 협업 기반의 Zero-Config 코드 에디터



# 목차

## 01 서비스 기획 배경

- 가벼운 협업
- 면접을 위한 새로운 접근

## 02 서비스 소개

24시간 한정 실시간 협업 에디터

## 03 핵심 기능 시연

- 방 생성
- 멀티 커서
- 실시간 편집

## 04 기술 스택 및 아키텍처

## 05 기술적 도전

## 06 팀 회고

# 서비스 기획 배경

## 문제 인식

- 무거운 IDE 의 오래 걸리는 로딩
- 간단한 협업에도 설치해야하는 익스텐션
- 빠른 코드 공유와 실행의 어려움



# 서비스 기획 배경

## 해결 방향

- 설치 없이 즉시 사용 가능한 환경 제공
- 링크 하나로 협업 공간 생성
- 로그인 없는 간편한 접근성 구현

## 핵심 가치

- Speed: 즉시 시작 가능한 빠른 속도
- Simplicity: 복잡한 설정이 필요 없는 단순함
- Collaboration: 실시간 협업 지원

# 서비스 소개

로그인 없이 즉시 생성  
24시간 뒤 사라지는 안전한 실시간 협업 에디터

## Live Sync

- CRDT 기반의 완벽한 동시성 제어
- 여러 사용자의 실시간 편집 동기화
- 충돌 없는 협업 환경 제공

## Zero-Config

- 접속 즉시 고유 작업 공간 자동 생성
- 별도의 설치나 설정 과정 불필요
- URL 공유만으로 협업 시작

## Safe Execution

- 샌드박스 환경에서의 안전한 코드 실행
- 다중 언어 지원 (Python, JavaScript 등)
- 보안 위협으로부터 시스템 보호



# 핵심 기능 시연

<http://drive.google.com/file/d/1fWc3hU0TBZPl6oINI6dok8yqa3mSa8bC/view>

# 기술 스택 및 아키텍처

Frontend

React 19, CodeMirror 6, TailwindCSS

Backend

NestJS, Socket.io, Redis, PostgreSQL

Sync Engine

Yjs (CRDT)

Execution

Piston (Sandboxed Engine)

Infra/Ops

Turborepo, Docker, CI/CD, Slack 연동

# 기술적 도전



# 실시간 동기화와 일관성

## 기술 배경

- 여러 명의 편집자가 동시에 코드를 수정할 때 발생하는 데이터 충돌 문제
- 편집 순서 보장 및 일관성 유지의 어려움
- 중앙 서버 의존으로 인한 병목 가능성

## 해결 방안

- Yjs(CRDT) 라이브러리를 도입하여 충돌 없는 동시 편집 구현
- 서버가 메시지 처리를 최소화하고 중개자 역할을 수행하면서 데이터 동기화
- 데이터의 일관성 보장

## CRDT의 장점

- 서버 부하 감소 및 네트워크 지연에 강건한 구조
- 오프라인 편집 후 재연결 시 자동 병합
- 복잡한 충돌 해결 로직 없이 자동 조정

## 동기화 범위

- 텍스트 편집 내용의 실시간 동기화
- 파일 메타데이터 및 텍스트 변경 사항 반영
- 커서 위치 및 선택 영역까지 동기화 객체로 관리

# 문서 영속성 지원

## 기술 배경

- 메모리에서 CRDT 로직을 처리하고 있음
- 서버 재시작 시 문서가 유실되면 사용자들이 서비스를 신뢰하지 않게 됨
- 영속성 문제를 해결하기 위해 Redis 도입

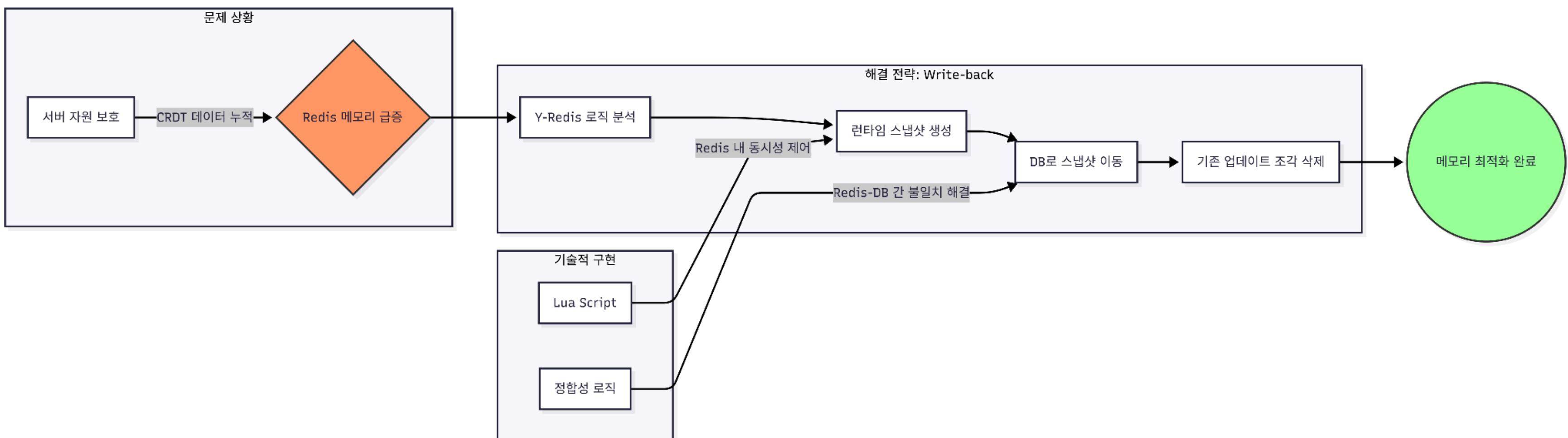
## Y-Redis

- Yjs 와 Redis 를 연결하는 라이브러리
- 업데이트 조각을 Redis 에 저장

## 스냅샷과 하드리밋

- 기존 Redis 를 사용했을 때 CRDT 데이터를 모두 저장하면서 **메모리 사용량이 빠르게 증가하는 문제** 존재
- 런타임에 **스냅샷을 DB 로 옮기고 Redis 업데이트 조각을 버리는 전략** 사용 (write back)
- Y-Redis 라이브러리 로직을 분석해서 런타임 스냅샷 쓰기 로직 추가 및 동시성 문제 처리
- 5MB 의 하드 리밋을 설정해서 시스템 메모리 자원 보호

# 문서 영속성 지원



# 한글 동시편집 시 글자 깨짐 방지

## 기술 배경

- 여러 명의 편집자가 같은 줄에서 한글을 입력할 때 글자가 깨지는 문제 발생

## 문제 원인

- 브라우저의 한글 입력기가 글자를 조합하는 과정 중에 원격 업데이트 사항이 사용자가 편집 중인 DOM 노드를 파괴하면서 발생

## 해결 방안

- 조합 시작과 끝 이벤트를 감지
- 조합 중일 때는 원격 업데이트를 Yjs 엔진이 처리하지 않도록 큐에 넣음
- 조합이 끝났을 때 큐를 비우면서 데이터 정합성 확보
- Yjs 엔진은 네트워크 지연으로 인한 문제인지 아니면 의도적으로 업데이트를 미룬 것인지 구별하지 못함

## 동시성 문제

- 신뢰할 수 있는 **Composition** 이벤트를 얻기 위해 **CodeMirror** 의 API 사용
- 한글은 조합 끝 시점과 조합 시작 시점의 동기적 타이밍을 가로채기 어려워서 rAF 타이밍 사용

# CI/CD

 Production 배포가 완료되었습니다!

배포 항목:

- Client
- Server

커밋: Merge pull request #344 from boostcampwm2025/dev  
dev -> main

URL: <https://lets-codejam.vercel.app>

작성자: son-hyejun

## Github workflows 를 이용하여 CI/CD 파이프라인을 구축

CI

- 1.코드 품질 검증
- 2.PR 코멘트
- 3.Vercel Preview 배포 (Client 변경 감지 시)

CD Staging

- 1.Client 배포 (Vercel CLI 빌드/Preview 환경 배포/Slack 알림)
- 2.Server 배포 (의존성 빌드/Docker 이미지 빌드 및 푸시/새 이미지 풀 및 컨테이너 실행/Slack 알림)

CD Production

- 1.Client 배포 (Production 환경 설정으로 빌드, Vercel Production 배포)
- 2.Server 배포(Staging과 동일 + 이전 이미지 정리)
- 3.통합 알림

# 사용자 인증 해결 과정



J143\_송상화 오전 2:54

CodeJam 의 Editor 권한을 해킹해 봤습니다 (!!)

```
// ==UserScript==  
// @name      CodeJam  
// @version   1.0.0  
// @description Spoof the 'editor'  
// @run-at    document-start  
// @match     http://49.50.138.93/room/prototype  
// ==/UserScript==  
  
(function () {  
  "use strict";  
  
  const reduce = Array.prototype.reduce;  
  
  Array.prototype.reduce = function (...args) {  
    const flag = this?.[0]?.ptId;  
    if (flag) this.forEach((e) => (e.role = "editor"));  
    return reduce.apply(this, args);  
  };  
})();
```

**Step 0 [배경].**  
참가자 정보 복원을 위한 참가자 ID 로컬스토리지에 저장

**Step 1. [보안]**  
LocalStorage 취약점 → HTTP-Only 쿠키 이관 및 재연결 흐름 구축

**Step 2. [충돌]**  
Vercel Rewrite 도입 → 프록시의 WS 프로토콜 미지원 확인

**Step 3. [병목]**  
쿠키를 얻으면 소켓이 끊기고, 소켓을 얻으면 쿠키가 막히는 기술적 딜레마

**Step 4. [완성]**  
API는 Rewrite, 소켓은 직결하는 하이브리드 구조로 모든 제약 해결

# 기타 기술적 도전

## 코드 실행

- Piston 엔진 기반의 독립된 샌드박스 환경 구축 및 다양한 언어 실행 지원
- 실행 시간, 메모리 제한 및 요청 횟수 제한 등으로 서버 리소스 보호
- NestJS 서버와 Piston 서버를 웹소켓 연결해서 실행 결과를 실시간으로 출력 => UX 개선

## 부하 테스트

- 100개의 방, 각 6명 (`TYPING_INTERVAL=1000, 2000`) 모두 성공
- Grafana, Prometheus, Loki 를 통한 서버 모니터링으로 부하 측정

## 요청 횟수 제한

- `@nestjs/throttler` 를 이용한 레이트 리밋 구현
- 방 생성 및 코드 실행에 도입

## > 실행 결과

Runtime: javascript 20.11.1

Running...

Process exited with code 0



참가자 14

참가자 검색 ...

● 입장순 ○ 이름순

- Host #6535** you
- HOST**
- bot-0 #8901**
- EDITOR**
- bot-1 #9367**
- EDITOR**
- bot-2 #1781**
- EDITOR**
- bot-3 #1868**
- EDITOR**
- bot-4 #8153**
- EDITOR**
- bot-5 #9851**
- EDITOR**
- bot-6 #5338**
- EDITOR**
- bot-7 #1523**
- EDITOR**
- bot-8 #9995**
- EDITOR**
- bot-9 #5868**
- EDITOR**
- bot-10 #4979**
- EDITOR**

### 제목 없음

HOST 01CEEX

```
main.js
```

```

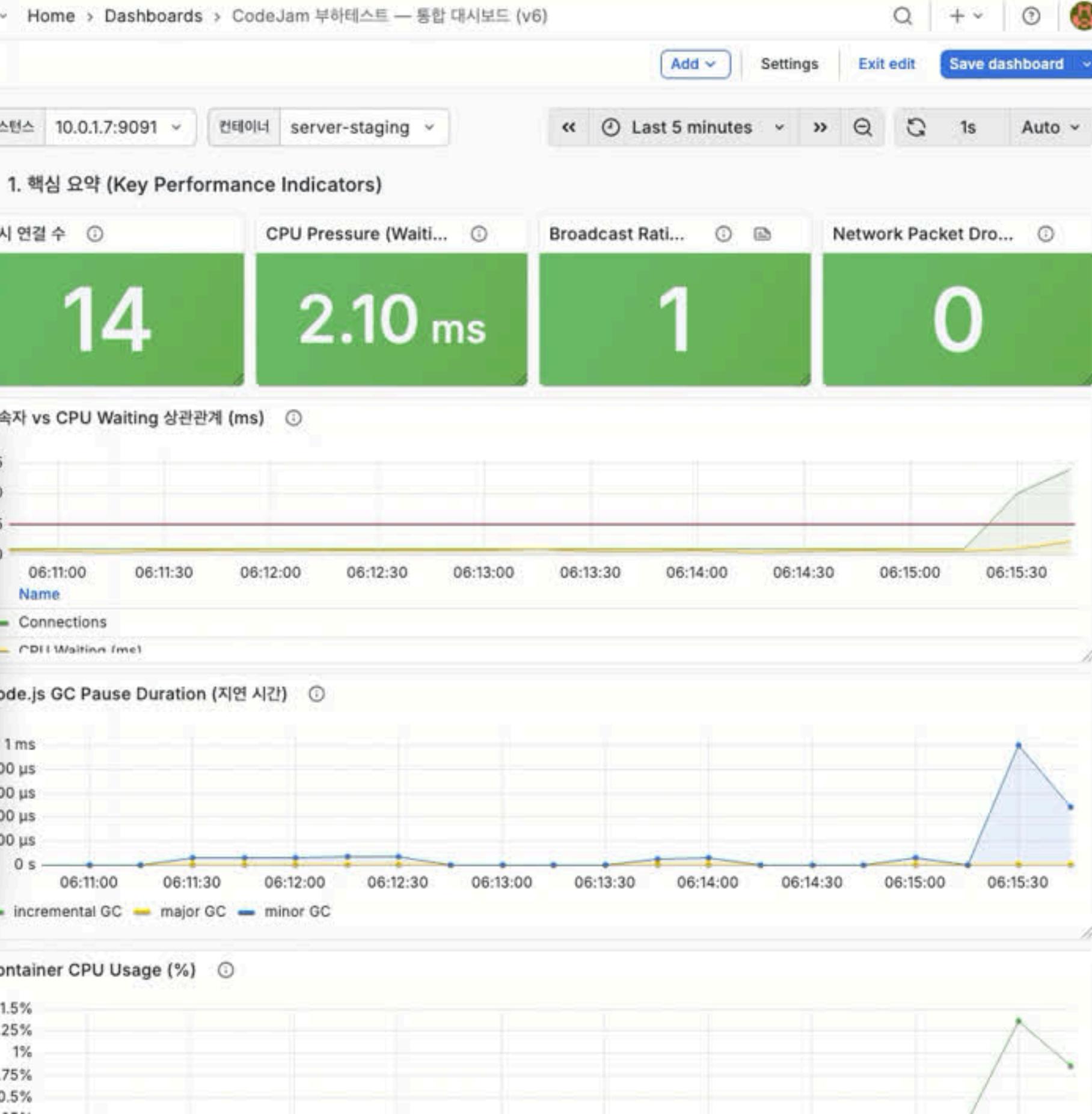
1 abcdefghijklm
2 abcdefghijklm
3 abcdefghijklm
4 abcdefghijklm
5 abcdefghijklm
6 abcdefghijklm
7 abcdefghijklm
8 abcdefghijklm
9 abcdefghijklm
10 abcdefghijklm
11 abcdefghijklm
12 abcdefghijklm
13

```

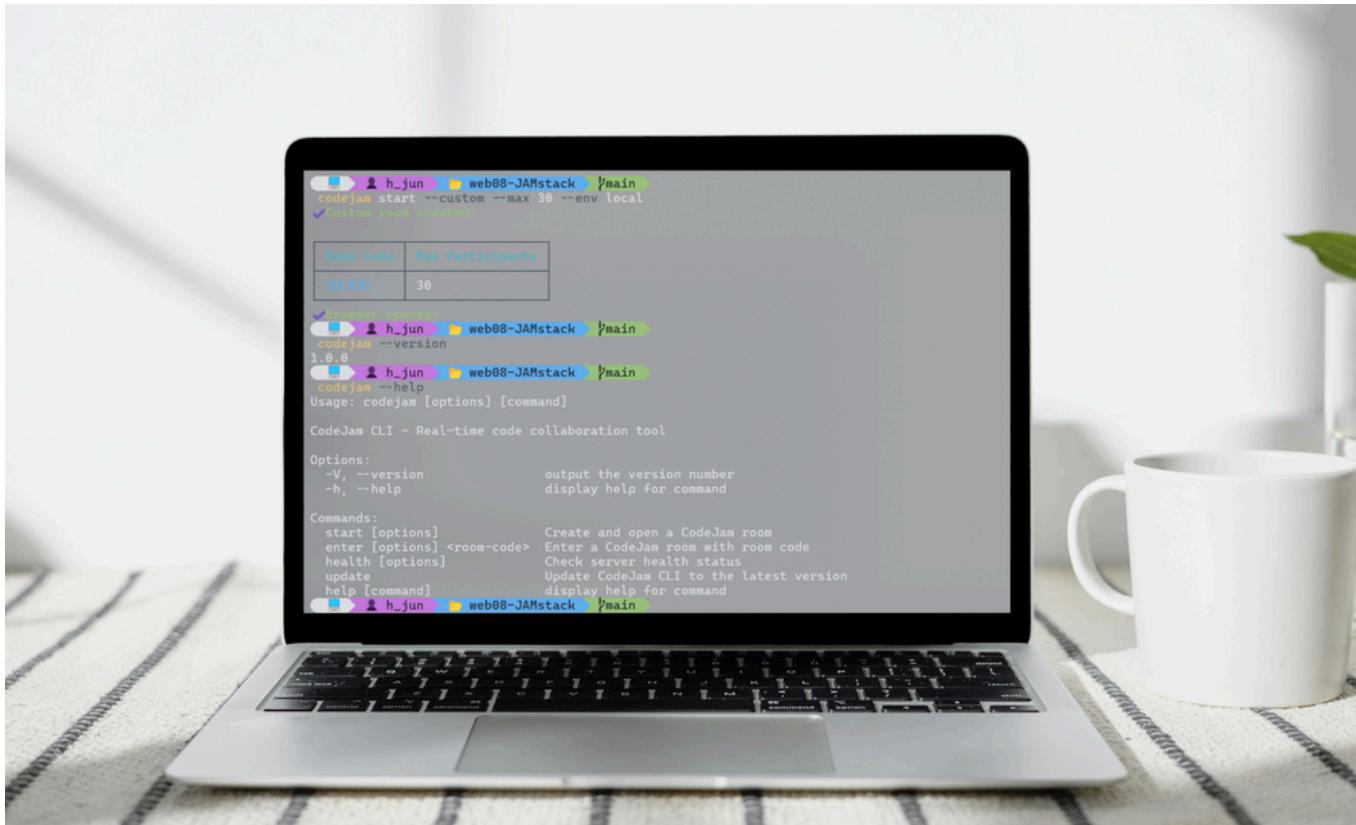
```

bash run-test.sh O1CEEX 13
user@MacBook-Pro ~/Desktop/GroupProject/scripts bash run-test.sh O1CEEX 13
===== 단일 프로세스 부하 테스트 시작 =====
방 코드 : O1CEEX
총 인원 : 13 명
===== 부하 테스트 시작 | 방 : O1CEEX | 목표 : 13명 =====
서버 : http://211.188.55.175:3001
설정 : Ramp-up 10s | Hold 30s | Interval 200ms
Custom Room 모드 : 권한 부여 대기 10초
===== 13명 생성 완료 =====
===== 브라우저에서 가상 유저들에게 "편집 권한"을 부여해 주세요!
10... 5... 4... 3... 2... 1...
대기 완료! 타이핑을 시작합니다...
===== 타이핑 시작 (부하 발생 중)...
[status] Connected: 13, Typing: 13, Emits: 129

```



# 사용자 피드백 및 개선사항



## CLI 도구 개발

- 터미널 환경에서 빠르게 CodeJam 세션을 생성하고 관리할 수 있는 **codejam-cli** 개발
- 개발자 워크플로우에 자연스럽게 통합되는 보조 도구 제공

## 사용성 개선

- 모바일 환경에서도 원활하게 사용 가능하도록 반응형 뷰 최적화
- 다크모드 지원으로 다양한 사용 환경에 대응
- 사용자 인터페이스 개선 및 접근성 향상

# 팀 회고

7주간의 프로젝트를 통해 실시간 협업 시스템 구축과 팀 협업의 중요성을 배웠습니다.  
코드 리뷰와 기술 문서화의 가치를 체감했으며,  
앞으로 더 풍부한 협업 기능을 추가하여 서비스를 고도화할 계획입니다.

## 회고 - 배운 점

- 코드 리뷰의 중요성: 팀원 간 지식 공유 및 코드 품질 향상
- 기술 문서화: 체계적인 문서 관리로 온보딩 시간 단축
- 실시간 협업 시스템 구축 경험 축적

## 향후 계획

- 마크다운 프리뷰 지원
- SonarQube 를 활용한 리팩토링 및 React Compiler 를 활용한 프론트엔드 최적화
- 유저 상호간 기능을 추가 (스포트라이트, Follow 등)

7주 동안 다들 수고 많으셨습니다