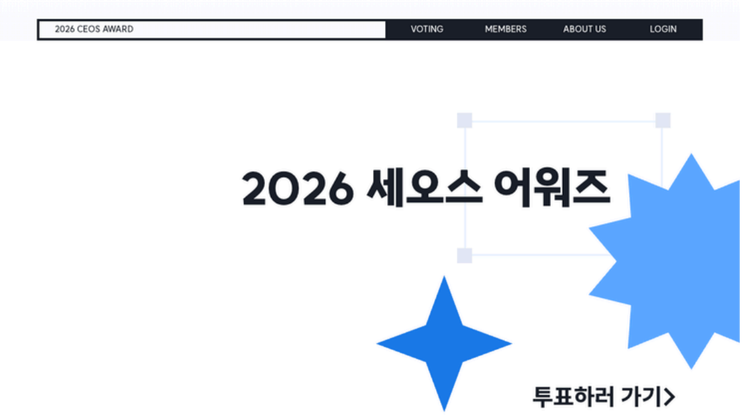
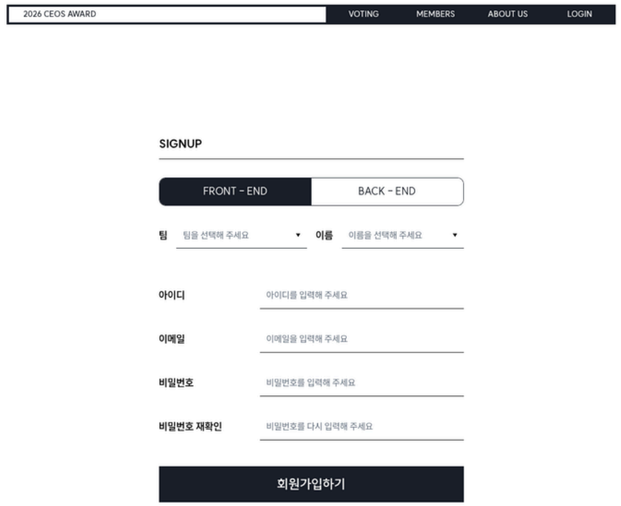
프론트엔드

화면 목록

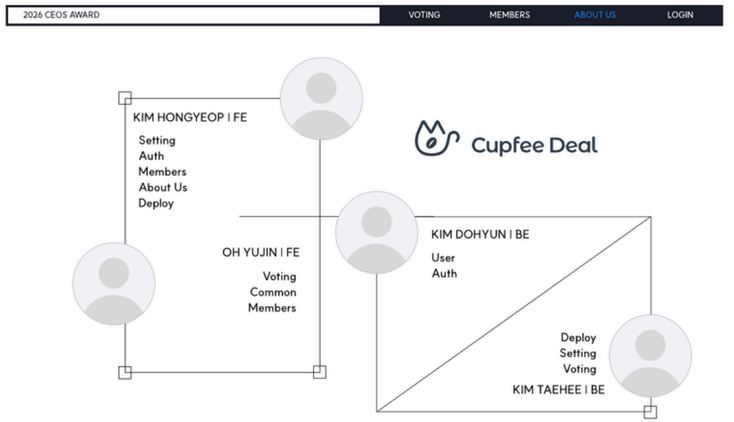
온보딩



회원가입/로그인

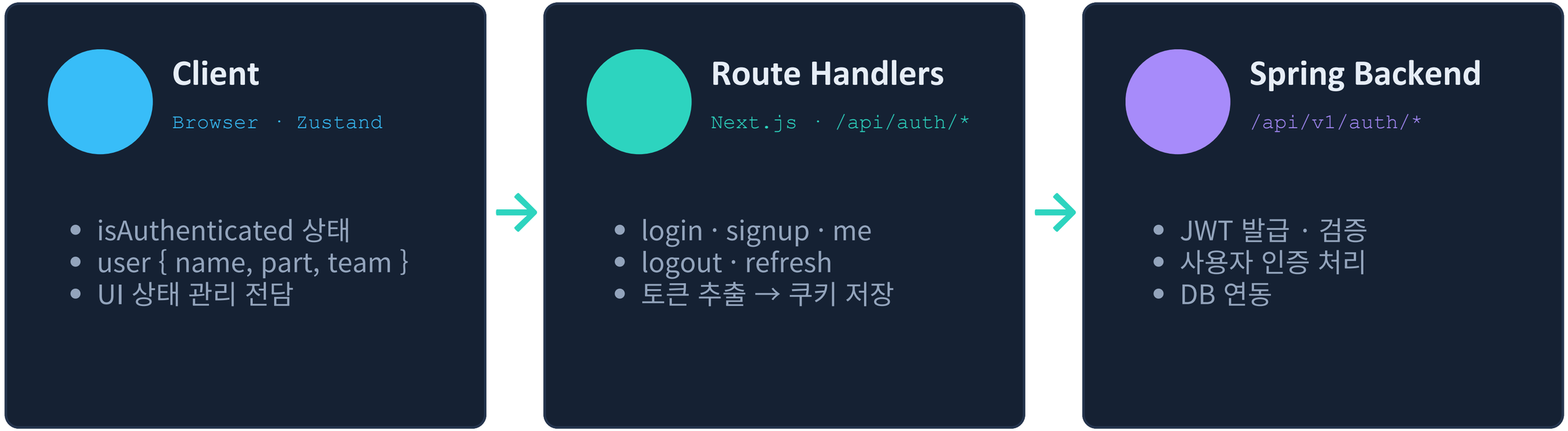


About Us



인증 아키텍처 개요

3-Layer 인증 아키텍처



클라이언트는 백엔드 URL을 모름

Route Handler가 중간 프록시

토큰 추출과 httpOnly 쿠키 저장 담당

인증 아키텍처 개요

httpOnly 쿠키 기반 토큰 관리

쿠키	httpOnly	secure	sameSite	path
accessToken	O	prod	lax	/
refreshToken	O	prod	lax	/api/auth

httpOnly

JS(document.cookie) 접근 불가 → XSS 토큰 탈취 차단

secure (prod)

HTTPS 연결에서만 전송 → 네트워크 도청 방지

sameSite: lax

외부 사이트 요청 시 쿠키 미포함 → CSRF 방어

인증 아키텍처 개요

새로고침 시 인증 상태 복원

문제

- Zustand는 메모리 기반 → 새로고침 시 상태 초기화

해결 — AuthProvider 패턴

1. RootLayout → AuthProvider (useEffect)
2. hydrate() 호출
3. GET /api/auth/me (백엔드 호출 없이 쿠키만 확인)
4. Zustand 상태 갱신 { isAuthenticated, user }

AuthProvider.tsx

```tsx
// AuthProvider.tsx
export default function AuthProvider(
  { children }) {
  const hydrate =
    useAuthStore((s) => s.hydrate);
  useEffect(() => { hydrate(); },
    [hydrate]);
  return <>{children}</>;
}
```

투표 API 연동

투표 흐름

조회

`GET`

- 진입 시 `results`로 후보 목록
- `votes/me`로 내 투표 여부

투표

`POST / PATCH`

- 선택
후보 클릭 → candidateId 저장
- 전송
투표하기 클릭 → POST(첫 투표) / PATCH(투표 변경)

결과

`GET`

`result/[pollId]` 진입
`results`로 득표 수 받아 화면에 Top 3 표시

결과 페이지를 동적 라우트 result/[pollId] 로
→ FE·BE·데모데이 3투표가 한 페이지 재사용

투표 API 연동

투표 수정 기능

배경

- 과제 조건 : 사용자는 한 번만 투표할 수 있으며, 중복 투표를 방지합니다.
- 중복되지 않으면 투표 내용을 변경할 수 있게 하는 것이 좋지 않을까?

해결 — 표를 추가가 아니라 변경하기

1. **PATCH**: 기존 후보 −1, 새 후보 +1
   → 계정 당 총 표 수 1 유지 = 중복 방지 ✅ + 변경 허용 ✅
2. votes/me로 투표 여부 확인 → 첫 투표 POST / 변경 PATCH
3. 재투표 진입 시 이전에 고른 후보를 미리 선택(호버처럼 강조)
   → 현재 내 표를 한눈에, 바로 변경 가능하도록 하여 UX 향상

voting/demoday/page.tsx

```
const handleSubmit = async () => {
 ...
 try {
 const res = await
fetch(API_ROUTES.vote.votes(POLL_ID), {
 // 처음 투표면 POST 이미 투표한 적 있으면 PATCH
 method: hasVoted ? 'PATCH' : 'POST',
 headers: { 'Content-Type':
'application/json' },
 body: JSON.stringify({ candidateId:
selectedId }),
 });
 ...
 };
```

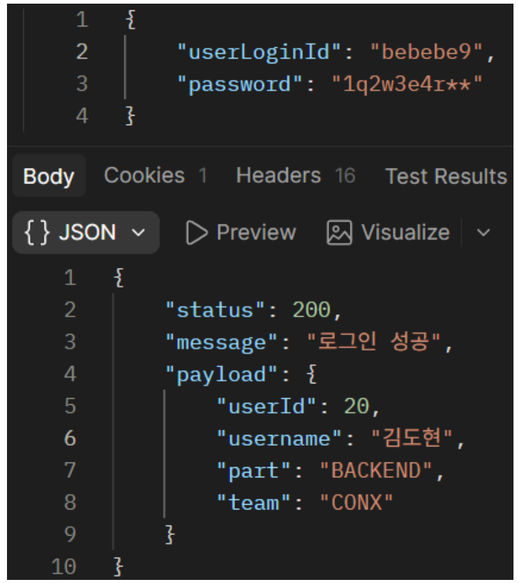
백엔드

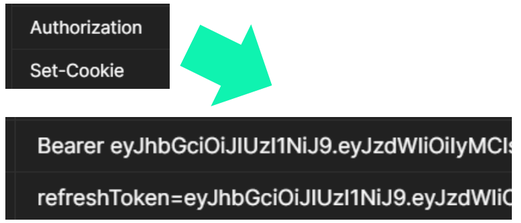
구현 API 기술 설명

```java
@Bean  김도현
public CorsConfigurationSource corsConfigurationSource() {
    CorsConfiguration configuration = new CorsConfiguration();
    configuration.setAllowedOrigins(List.of(
            "https://www.conx.co.kr",
            "https://conx.co.kr",
            "http://localhost:3000"
    ));
    configuration.setAllowedMethods(List.of("GET", "POST", "PUT", "DELETE", "PATCH", "OPTIONS"));
    configuration.setAllowedHeaders(List.of( e1: "*"));
    configuration.setAllowCredentials(true);
    configuration.setMaxAge(3600L);

    UrlBasedCorsConfigurationSource source = new UrlBasedCorsConfigurationSource();
    source.registerCorsConfiguration( pattern: "/**", configuration);
    return source;
}
```

CORS 설정





로그인 방식

폼로그인

+JWT토큰 인증

/api/v1/login

인증방식 accessToken - 요청헤더 (1시간 유효)

refreshToken - 쿠키 + 레디스 (3일 유효)

구현 API 기술 설명

회원가입
POST
/api/v1/auth/signup

요청 body
String userId - 로그인ID
String password - 비밀번호
String email - 이메일
Part part - 파트(프/벡)
String username - 이름
Team team - 팀

리프레시 토큰 재발급
POST
/api/v1/auth/refresh

투표 생성
POST
/api/v1/polls

요청 body
String title - 투표이름
VoteType voteType - 투표유형(데모데이/파트장)
List<CandidateCreateRequest> candidates - 투표자

CandidateCreateRequest
String name - 이름
Part part - 파트
Team team - 팀

투표 결과보기 (내림차순)
GET
/api/v1/polls/{pollId}results

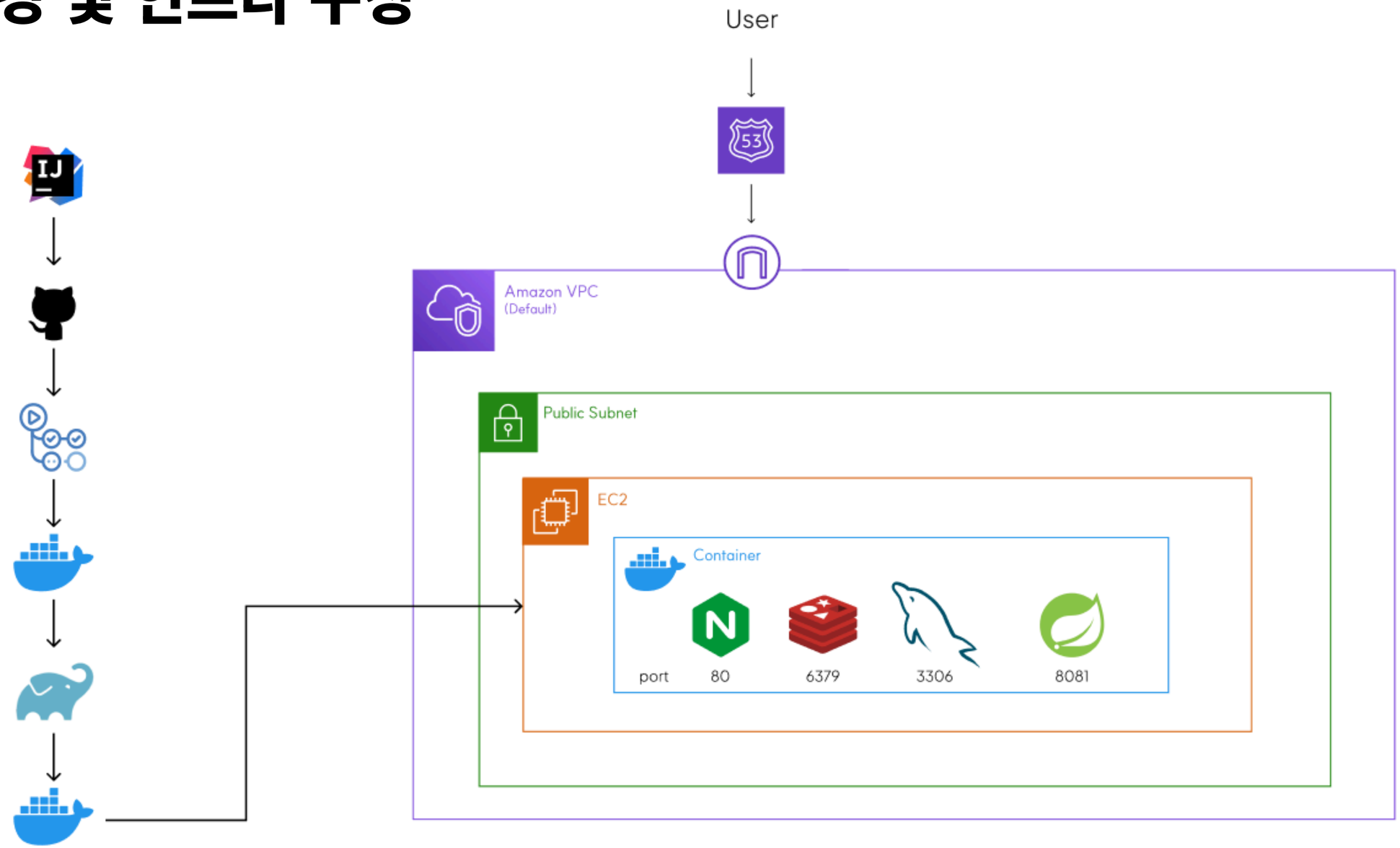
투표하기
POST | PATCH(재투표)
/api/v1/polls/{id}/votes

요청 body
long candidateId - 투표자id

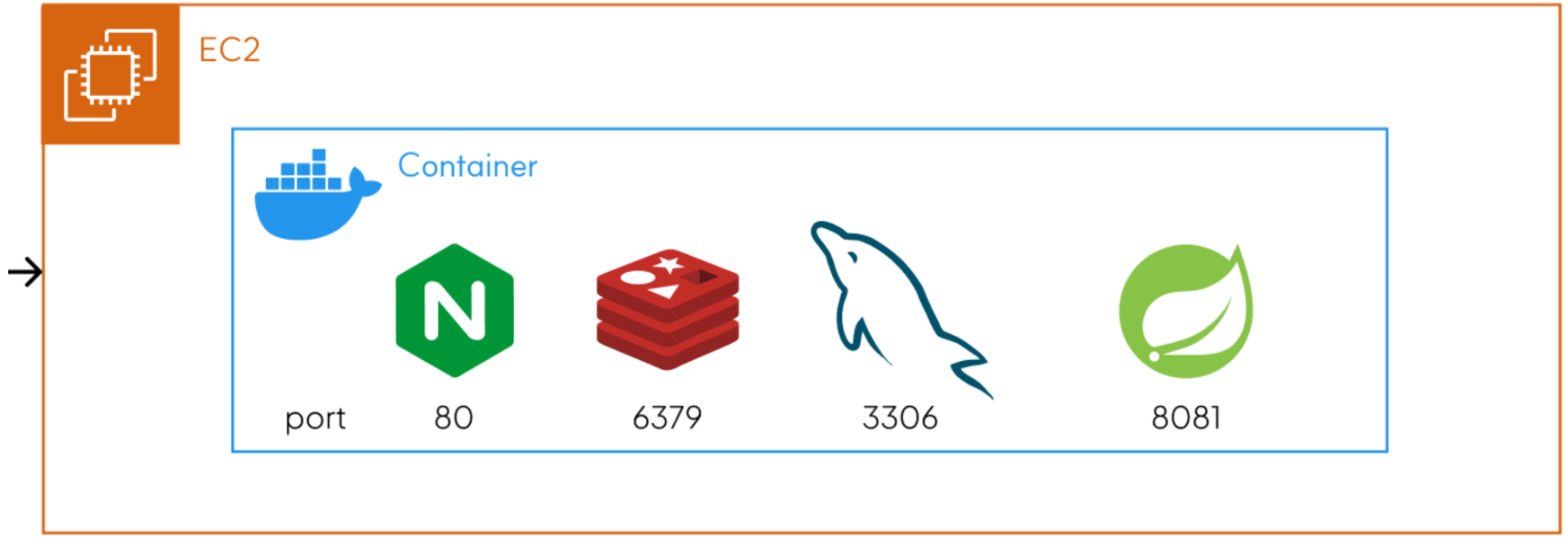
내 투표보기
GET
/api/v1/polls/{id}/votes/me

투표취소
DELETE
/api/v1/polls/{id}/votes

배포 환경 및 인프라 구성


User
53
Amazon VPC
(Default)
Public Subnet
EC2
Container
port
80
6379
3306
8081

트러블슈팅



t3.micro에서 도커/레디스/MySQL/스프링을 돌리려니 버벅임...!

트러블슈팅

EC2

```
ubuntu@ip-[redacted]:~/spring-vote-23rd$ ps aux --sort=-%mem|head -10
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root       46555 21.0 30.2 2863756 281024 ?       Ssl  10:21   0:27 java -jar app.jar
dnsmasq    46300  1.8 13.3 1779200 123940 ?       Ssl  10:20   0:02 mysqld
root         915  0.2  4.7 2416340 44420 ?       Ssl  07:43   0:27 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
root         695  0.3  3.1 1868956 29268 ?       Ssl  07:43   0:30 /usr/bin/containerd
root         610  0.0  1.7 1885160 15988 ?       Ssl  07:43   0:01 /usr/lib/snapd/snapd
root       46531  0.0  1.1 1233872 11040 ?       Sl   10:21   0:00 /usr/bin/containerd-shim-runc-v2 -namespace moby -id e96b4c36045543b9175a1be07c57
3d0c20d265ab672bde62d5f24c48a8652d88 -address /run/containerd/containerd.sock
root           1  0.0  1.0  25100  9540 ?        Ss   07:43   0:04 /sbin/init
root         177  0.0  0.9 279208  9044 ?        SLsl 07:43   0:00 /usr/sbin/multipathd -d -s
root       46250  0.0  0.9 1233872 8920 ?        Sl   10:20   0:00 /usr/bin/containerd-shim-runc-v2 -namespace moby -id a7398b1e487c59b5bfb58d31b1e9
cf5a9aea808b1255ba84a4e29e78fef646c8 -address /run/containerd/containerd.sock
```

Java 프로젝트 (약 281MB)

MySQL (약 124MB)

도커 (약 93MB)

t3.micro에서 도커/레디스/MySQL/스프링을 돌리려니 버벅임...!

자바 + MySQL + 도커 합쳐서 약 500MB 사용 중 (55%)

트러블슈팅

```
ubuntu@ip-[redacted]:~/spring-vote-23rd$ free -h
               total        used        free      shared  buff/cache   available
Mem:           908Mi       682Mi        53Mi       2.1Mi       281Mi       226Mi
Swap:          2.0Gi       438Mi       1.6Gi
```

t3.micro에서 도커/레디스/MySQL/스프링을 돌리려니 버벅임...!

자바 + MySQL + 도커 합쳐서 약 500MB 사용 중 (55%)

swap 메모리를 사용하여 RAM 2GB 확보

→ 자바 프로젝트, 레디스, MySQL까지 구동 성공

당장 필요한 RAM 용량이 부족한 상황이므로

느린 속도를 감수하더라도 swap 메모리를 사용

트러블슈팅

```
ubuntu@ip-[redacted]:~/spring-vote-23rd$ free -h
               total        used        free      shared  buff/cache   available
Mem:           908Mi       682Mi        53Mi       2.1Mi       281Mi       226Mi
Swap:          2.0Gi       438Mi       1.6Gi
```

t3.micro에서 도커/레디스/MySQL/스프링을 돌리려니 버벅임...!

자바 + MySQL + 도커 합쳐서 약 500MB 사용 중 (55%)

swap 메모리를 사용하여 RAM 2GB 확보

→ 자바 프로젝트, 레디스, MySQL까지 구동 성공

만약 DB 및 레디스에 데이터가 증가해 RAM이 부족해진다면

자바 프로젝트에서 용량확보 또는 인스턴스 업그레이드가 필요할 것...!

우리 프로젝트에 적용하기

```
ubuntu[redacted]~$ ps aux --sort=-%mem | head -10
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root     2659975  0.9 40.2 2867852 374556 ?      Ssl  08:57   1:02 java -jar app.jar
root        6439  0.0  5.0 2556876 46604 ?       Ssl  May22  43:59 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
root        6292  0.1  2.8 2092852 26640 ?       Ssl  May22  66:55 /usr/bin/containerd
root     3139322  0.0  2.4  99400 23232 ?        S<s  Jun10   0:21 /usr/lib/systemd/systemd-journald
root        1124  0.0  1.5 1832756 14316 ?       Ssl  May22   6:12 /snap/amazon-ssm-agent/13009/amazon-ssm-agent
root      355006  0.0  1.4 1851356 13576 ?       Ssl  May24  11:57 /snap/snapd/current/usr/lib/snapd/snapd
ubuntu   2672259  0.3  1.3  21876 12612 ?        Ss   10:43   0:00 /usr/lib/systemd/systemd --user
root     2672254  0.0  1.2  18112 11620 ?        Ss   10:43   0:00 sshd-session: ubuntu [priv]
root           1  0.0  1.1  25700 10244 ?        Ss   May22   1:51 /usr/lib/systemd/systemd --system --deserialize=96
```

Java 프로젝트 (약 374MB)

도커 (약 72MB)

```
ubuntu@ip[redacted]:~$ free -h
               total        used        free      shared  buff/cache   available
Mem:           908Mi       710Mi        53Mi       2.3Mi       258Mi       197Mi
Swap:          2.0Gi       162Mi       1.8Gi
```

CONX 서버 인스턴스 RAM 사용현황 (스왑메모리 사용 중)

우리 프로젝트에 적용하기

```
ubuntu[redacted]~$ ps aux --sort=-%mem | head -10
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root     2659975  0.9 40.2 2867852 374556 ?      Ssl  08:57   1:02 java -jar app.jar
root        6439  0.0  5.0 2556876 46604 ?       Ssl  May22  43:59 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
root        6292  0.1  2.8 2092852 26640 ?       Ssl  May22  66:55 /usr/bin/containerd
root     3139322  0.0  2.4  99400 23232 ?        S<s  Jun10   0:21 /usr/lib/systemd/systemd-journald
root        1124  0.0  1.5 1832756 14316 ?       Ssl  May22   6:12 /snap/amazon-ssm-agent/13009/amazon-ssm-agent
root      355006  0.0  1.4 1851356 13576 ?       Ssl  May24  11:57 /snap/snapd/current/usr/lib/snapd/snapd
ubuntu   2672259  0.3  1.3  21876 12612 ?        Ss   10:43   0:00 /usr/lib/systemd/systemd --user
root     2672254  0.0  1.2  18112 11620 ?        Ss   10:43   0:00 sshd-session: ubuntu [priv]
root           1  0.0  1.1  25700 10244 ?        Ss   May22   1:51 /usr/lib/systemd/systemd --system --deserialize=96
```

Java 프로젝트 (약 374MB)

도커 (약 72MB)

```
ubuntu@ip[redacted]:~$ free -h
               total        used        free      shared  buff/cache   available
Mem:           908Mi       710Mi        53Mi       2.3Mi       258Mi       197Mi
Swap:          2.0Gi       162Mi       1.8Gi
```

CONX 서버 인스턴스 RAM 사용현황 (스왑메모리 사용 중)

→ 메모리를 아껴쓰고, 메모리가 부족한 경우
느린 속도를 감수하더라도 swap 메모리를 사용해 메모리를 확보하자

(라고하지만 아직 저는 느린속도 체감을 못해봤습니다..)