

# [쿠버네티스] 프로젝트 1

☰ 카테고리	Docker
☰ 태그	
☑ 공개	<input type="checkbox"/>
📁 난이도	

## Upbit Clone 프로젝트 최종 설계안 (Re-Revised)

### 1. 최종 기술 스택 및 데이터 정의

분류	기술 스택	세부 역할 및 저장 데이터
Infra	Ubuntu 24.04 / Tailscale	하이브리드 노드 통합, Tailscale Subnet Router(Bastion) 보안 구성
Platform	Kubernetes (v1.31+)	컨테이너 오케스트레이션 및 Patroni(DB HA) 운영
Backend	Node.js (Express)	업비트 Open API 수집, 비즈니스 로직, DB/Redis 커넥션 관리
Frontend	React	Tailwind CSS 기반의 실시간 시세 대시보드 및 차트 시각화
DB 1 (Cache)	Redis	<b>휘발성:</b> 실시간 호가, 현재가, API 요청 제한(Rate Limit) 관리
DB 2 (Main)	TimescaleDB	<b>시계열:</b> 분봉/일봉 캔들, 체결 내역 (Hypertable) <b>관계형:</b> 유저 정보, 자산 현황, 마켓 코드
Ops	ArgoCD / Prometheus	(일정 후반) GitOps 배포 파이프라인 및 클러스터 리소스 모니터링

### 2. 장비별 리소스 및 서비스 배치 (Updated)

- \*노트북(Control & View)\*\*과 \*\*데스크탑(Core Data & Logic)\*\*으로 역할을 완전히 분리했습니다.

🖥️ **노트북: Control & Frontend Zone (32GB)**

사용자와 직접 맞닿는(UI) 역할과 클러스터 제어를 담당합니다.

노드 이름 (Hostname)	역할 (Role)	주요 컴포넌트 & 스펙 (할당량)
<b>master</b>	<b>Control Plane</b>	<ul style="list-style-type: none"> <li>K8s Control Plane (API Server, Scheduler)</li> <li>(4GB)</li> </ul>
<b>worker-2</b>	<b>Frontend</b>	<ul style="list-style-type: none"> <li><b>React App (Web Server)</b></li> <li>Nginx Ingress Controller (외부 트래픽 유입)</li> <li>(10GB)</li> </ul>
<b>db-slave</b>	<b>DB Read Replica</b>	<ul style="list-style-type: none"> <li><b>TimescaleDB Slave</b> (Streaming Replication)</li> <li>읽기 전용 쿼리 처리 (차트 조회 등)</li> <li>(8GB)</li> </ul>

## 데스크탑: Backend & Data Core Zone (32GB)

무거운 연산(수집)과 데이터 저장(DB, Storage)을 전담합니다.

노드 이름 (Hostname)	역할 (Role)	주요 컴포넌트 & 스펙 (할당량)
<b>worker-1</b>	<b>Backend</b>	<ul style="list-style-type: none"> <li><b>Node.js App (Ingestor/API)</b></li> <li>업비트 소켓 연결 유지 및 데이터 가공</li> <li>(8GB)</li> </ul>
<b>db-master</b>	<b>DB Master</b>	<ul style="list-style-type: none"> <li><b>TimescaleDB Master</b> (모든 쓰기/수정)</li> <li><b>Redis</b> (인메모리 캐시)</li> <li>(12GB)</li> </ul>
<b>storage-nfs</b>	<b>Storage</b>	<ul style="list-style-type: none"> <li>NFS Server (PV/PVC 영구 데이터 저장소)</li> <li>DB 백업 파일 관리</li> <li>(4GB)</li> </ul>
<b>monitor-1</b> (New)	<b>Monitoring</b>	<ul style="list-style-type: none"> <li>Prometheus (메트릭 수집)</li> <li>Grafana (대시보드)</li> <li>(4GB)</li> </ul>

## 3. 데이터 흐름 (Data Flow)

변경된 노드 이름을 반영한 데이터 이동 경로입니다.

- 수집:** **worker-1** (Node.js)이 업비트에서 데이터를 받음.
- 저장/캐싱:** **worker-1** 이 **db-master** (Redis/TimescaleDB)에 데이터를 씀(Write).
- 복제:** **db-master** 의 데이터가 VPN을 타고 **db-slave** (노트북)로 실시간 복제됨.
- 조회:** 사용자가 접속하면 **worker-2** (React)가 **db-slave** (읽기) 또는 **db-master** (Redis)에서 데이터를 가져와 보여줌.

Flow: Upbit API → worker-1 → db-master (Write) ➡ (Replication) ➡ db-slave  
→ worker-2 (Read)

---

## 4. 상세 일정 (Schedule)

CI/CD 구축을 마지막(3-2단계)으로 미루어 현실적인 개발 시간을 확보했습니다.

- **1단계 (12/19 ~ 12/26): [인프라 & DB 기초]**
    - Ubuntu/Tailscale/K8s 설치 완료.
    - TimescaleDB Master ( **db-master** ) 구축 및 테이블 생성 (현재 진행 중).
    - 노드 라벨링 재정리 (위의 이름대로).
  - **2단계 (12/27 ~ 01/02): [데이터 파이프라인 & 백엔드]**
    - **db-slave** 복제 설정 (네트워크 이슈 해결 후).
    - Node.js 수집기 ( **worker-1** ) 개발 및 배포.
    - Redis 연동 및 실시간 시계 데이터 적재 테스트.
  - **3단계 (01/03 ~ 01/09): [프론트엔드 & 마무리]**
    - **3-1 (필수):** React ( **worker-2** ) 호가창/차트 UI 개발 및 API 연동.
    - **3-2 (선택/후순위):** ArgoCD (CI/CD) 파이프라인 구축 및 Prometheus ( **monitor-1** ) 연동.
    - 최종 발표 자료 정리 및 시연 시나리오 점검.
-