최종 발표

원격 모니터링 및 배포 시스템

네트워크 프로그래밍 | 송해상 교수님
1조 [redacted] 신연준, [redacted]
2025.12.03

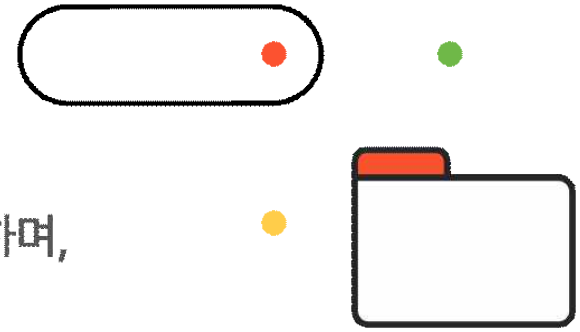
INDEX

목차

01

원격 모니터링 및 배포 시스템

중앙 서버가 여러 클라이언트(PC)와 통신하며,
각 PC의 상태를 확인하고, 명령(배포·재시도·재부팅 등)을 원격 실행하며,
공지(알림) 전송 내역을 관리하는 시스템.



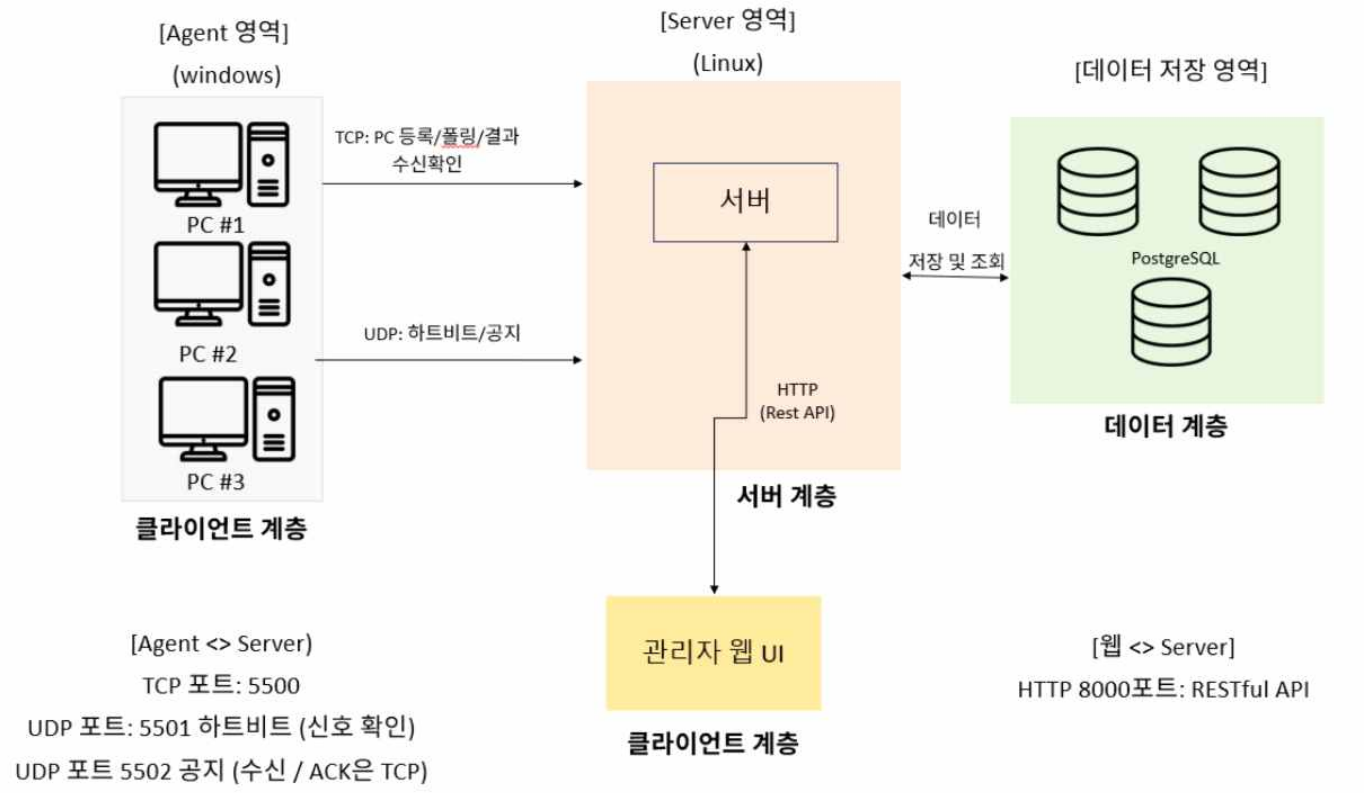
기능표

대분류	단위기능	상세 설명	비교/제약사항	개발우선순위
1. PC 관리	1.1. PC 등록	클라이언트에서 토큰을 이용하여 서버에 등록	최초 1회 TCP 등록, 토큰 분실 시 재발급 필요	P0
	1.2. PC 목록/검색	호스트명 검색, 정렬(이름/최근 체크인)	host_name GIN(trgm) 인덱스 활용	P0
	1.3. 온라인/오프라인 표시	마지막 체크인 기반 자동 판정	현재시간−last_check_in ≤ poll_interval×3	P0
	1.4. 상세 정보 패널	OS, 에이전트 버전, 최근 체크인, 폴링주기, IP 등 요약	DB 필드 그대로 노출	P0
2. 배포 관리	2.1. 배포 만들기	명령/PowerShell 템플릿 작성, 이름·설명 입력	command 필수, 위험 명령 가드	P0
	2.2. 대상 선택	전체/온라인/선택(체크한 PC)	tag:group 미사용	P0
	2.3. 실행(즉시/예약)	now 실행 또는 예약 시간 지정	예약은 서버 타이머 큐 필요	P0
	2.4. 진행 현황 보드	진행률·성공/실패 수, 최신 10개 로그 요약	-	P0
	2.5. 결과 상세	PC별 종료코드·시간 확인, 전체 로그 링크	전체 로그는 스토리지 URL	P0
	2.6. 재시도	실패 PC만 재실행	동일 고유키로 새 request 발급	P1
3. 공지 관리	3.1. 공지 작성/보내기	제목·내용 입력 후 전송 방식 선택(브로드캐스트/멀티캐스트)	멀티캐스트는 OS/네트워크 설정 필요	P0
	3.2. 수신 현황	대상 수/수신 수/수신률 표시, PC별 수신 시간	-	P0
	3.3. 재전송	미수신 PC만 재전송	중복 수신 방지	P1
	3.4. 공지 이력	최근 보낸 공지 목록/필터	보관 주기 설정 가능	P1

C/S 기능 분할

대분류	단위기능	상세 설명	에이전트 활동	서버 활동	관리자 활동
1. PC 관리	1.1. PC 등록	클라이언트에서 토큰을 이용하여 서버에 등록	PC 정보 수집 → 서버에 등록 요청 - id, hostname, os, address 정보 전달	등록 요청을 보낸 에이전트가 유효할 경우 토큰 발급, 전달받은 정보 DB에 저장 - 등록 성공 시 알림, 에이전트에 토큰 전달	신규 등록 PC 목록 확인 및 승인
	1.2. PC 목록/검색	호스트명 검색, 정렬(이름/최근 체크인)	-	DB에서 PC 정보 조회 및 정렬 결과 반환	등록된 PC 목록 검색, 필터링
	1.3. 온라인/오프라인 표시	마지막 체크인 기반 자동 판정	주기적 상태 보고(heartbeat) 전송	마지막 체크인 시각 비교 → 상태 계산	각 PC의 상태(온라인/오프라인) 확인
	1.4. 상세 정보 패널	OS, 에이전트 버전, 최근 체크인, 폴링주기, IP 등 요약	CPU·메모리·디스크 등 상세 정보 수집 후 보고	최신 정보 DB 업데이트	개별 PC 세부 정보 열람
2. 배포 관리	2.1. 배포 만들기	명령/PowerShell 템플릿 작성, 이름·설명 입력	-	요청받은 배포 설정을 DB에 저장	배포 템플릿 작성 및 저장후 서버에 요청 - deploy_name, target, command, by 전달
	2.2. 대상 선택	전체/온라인/선택(체크한 PC)	-	선택된 대상 PC 리스트를 매핑	배포 대상 지정(선택/전체/온라인)
	2.3. 실행(즉시/예약)	now 실행 또는 예약 시간 지정	명령 수신 → 실행 → 결과 로그 전송	배포 실행 요청 전달 - deploy_id, command, parameters 전달	즉시/예약 실행 설정 및 모니터링
	2.4. 진행 현황 보드	진행률·성공/실패 수, 최신 10개 로그 요약	실행 결과를 단계별로 보고	수집된 로그를 통계화 및 요약	배포 진행률, 성공/실패 현황 모니터링
	2.5. 결과 상세	PC별 종료코드·시간 확인, 전체 로그 링크	실행 로그 생성 및 업로드, 서버에 결과 전송 - deploy_id, agent_id, result, exit_code 전달	로그 저장소(URL) 생성 및 연결, DB에 작업 상태 갱신(deploy_id, status)	개별 PC 실행 결과 확인
	2.6. 재시도	실패 PC만 재실행	실패 항목 재요청 수신 후 재실행	실패 이력 기반 재배포 요청 생성	실패 항목 선택 후 재시도 명령
3. 공지 관리	3.1. 공지 작성/보내기	제목·내용 입력 후 전송 방식 선택(브로드캐스트/멀티캐스트)	공지 수신 후 확인 응답 전송	DB에 공지 정보 저장 이후 공지 발송 요청을 각 대상에 전달	공지 작성 및 전송 실행 - 서버에 notice_title, notice_body, target, by 전달
	3.2. 수신 현황	대상 수/수신 수/수신률 표시, PC별 수신 시간	공지 수신 → 응답(ACK) 전송	수신 여부 및 응답 시간 기록	수신률, 미응답 PC 상태 확인
	3.3. 재전송	미수신 PC만 재전송	재수신 후 응답 전송	미수신 대상만 필터링 후 재전송	재전송 요청 수행 및 상태 재확인
	3.4. 공지 이력	최근 보낸 공지 목록/필터	-	공지 이력 DB 조회 및 관리	보낸 공지 검색, 필터별 조회

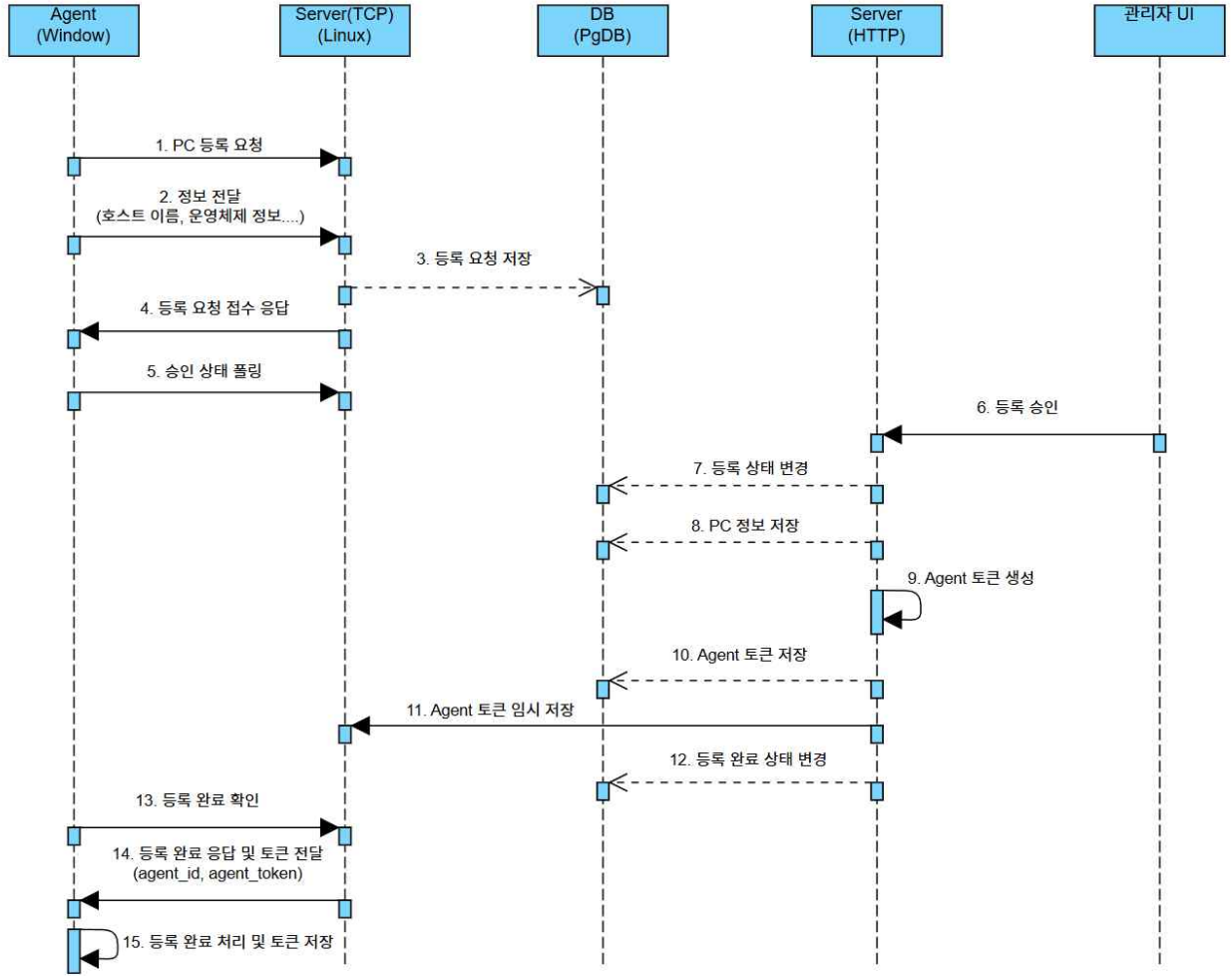
C/S 아키텍쳐


[Agent 영역]
(windows)
PC #1
PC #2
PC #3
클라이언트 계층
TCP: PC 등록/폴링/결과
수신확인
UDP: 하트비트/공지
[Server 영역]
(Linux)
서버
HTTP
(Rest API)
서버 계층
관리자 웹 UI
클라이언트 계층
데이터
저장 및 조회
[데이터 저장 영역]
PostgreSQL
데이터 계층
[Agent <> Server)
TCP 포트: 5500
UDP 포트: 5501 하트비트 (신호 확인)
UDP 포트 5502 공지 (수신 / ACK은 TCP)
[웹 <> Server]
HTTP 8000포트: RESTful API

C/S 프로토콜
- PC 등록 -

1. PC 등록 요청 (Agent → TCP Server) [동기: 요청 후 응답 대기]
2. 정보 전달 (호스트명, OS 정보...) (Agent → TCP Server) [동기: 1번과 함께 전송]
3. 등록 요청 저장 (TCP Server → DB) [비동기: async/await]
4. 등록 요청 접수 응답 (request_id, agent_id 반환) (TCP Server → Agent) [동기: 1번의 응답]
5. 승인 상태 폴링 (Agent → TCP Server) [동기: 요청 후 응답 대기, type: "status_check"]
6. 관리자 승인 (관리자 UI → HTTP API) [동기: HTTP 요청-응답, 웹 UI는 HTTP 유지]
7. 등록 요청 상태 변경 (HTTP Server → DB) [비동기: async/await, status: "approved"]
8. PC 정보 저장 (HTTP Server → DB) [비동기: async/await, pc 테이블 생성]
9. Agent 토큰 생성 (HTTP Server 내부) [동기: 내부 처리]
10. agent 토큰 저장 (hash) (HTTP Server → DB) [비동기: async/await]
11. agent 토큰 임시 저장 (HTTP Server → TCP Server._pending_tokens) [동기: 메모리 저장]
12. 등록 완료 상태 변경 (HTTP Server → DB) [비동기: async/await, status: "completed"]
13. 등록 완료 확인 (Agent → TCP Server) [동기: 요청 후 응답 대기, type: "status_check"]
14. 등록 완료 응답 및 토큰 수신 (TCP Server → Agent) [동기: 13번의 응답, agent_id, agent_token 반환]
15. 등록 완료 처리 및 토큰 저장 (Agent 내부) [동기: 내부 처리]

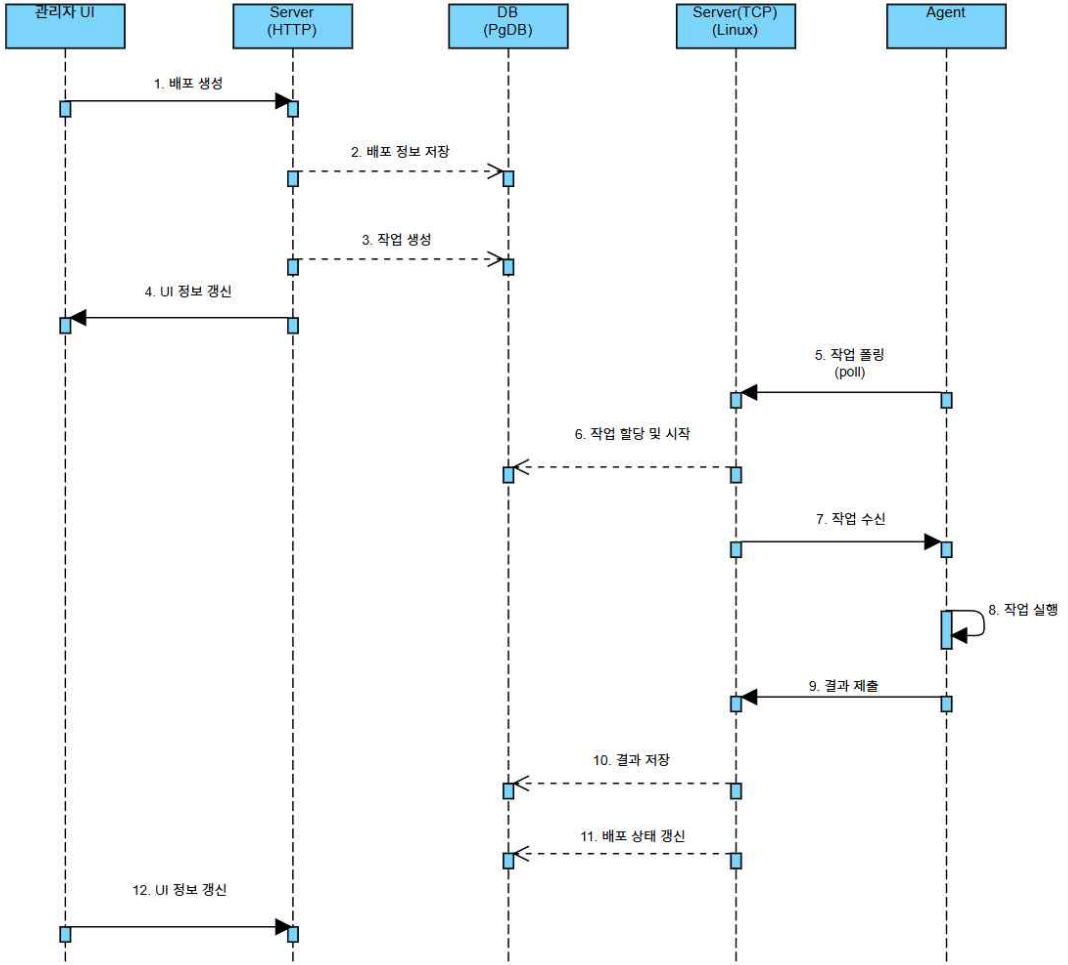
시퀀스 다이어그램
- PC 등록



C/S 프로토콜
- 배포 생성 및 처리 -

1. 배포 생성 (관리자 UI → HTTP API) [동기: HTTP 요청-응답]
2. 배포 정보 저장 (HTTP Server → DB) [비동기: async/await, deployments 테이블]
3. 작업 생성 (HTTP Server → DB) [비동기: async/await, 각 agent_id마다 tasks 테이블에 생성, status: "pending"]
4. UI 정보 갱신 (HTTP Server → 관리자 UI) [동기: 1번의 응답, deployment_id 반환]
5. 작업 폴링 (Agent → TCP Server) [동기: 요청 후 응답 대기, type: "poll", 주기적 반복]
6. 작업 할당 및 시작 (TCP Server → DB) [비동기: async/await, status: "pending" → "assigned" → "running"]
7. 작업 수신 (TCP Server → Agent) [동기: 5번의 응답, task_id, command, timeout, admin_required 반환]
8. 작업 실행 (Agent 내부) [동기: subprocess 실행, 명령어 실행 및 결과 수집]
9. 결과 제출 (Agent → TCP Server) [동기: 요청 후 응답 대기, type: "result", status, exit_code, log_summary 전송]
10. 결과 저장 (TCP Server → DB) [비동기: async/await, tasks 테이블 업데이트, status: "completed"/"failed"]
11. 배포 상태 갱신 (TCP Server → DB) [비동기: async/await, complete_task 내부에서 자동 호출, 모든 작업 완료 시 deployments 테이블 상태 업데이트]
12. UI 정보 갱신 (관리자 UI → HTTP API) [동기: HTTP GET 요청, 배포 상태 및 결과 조회]

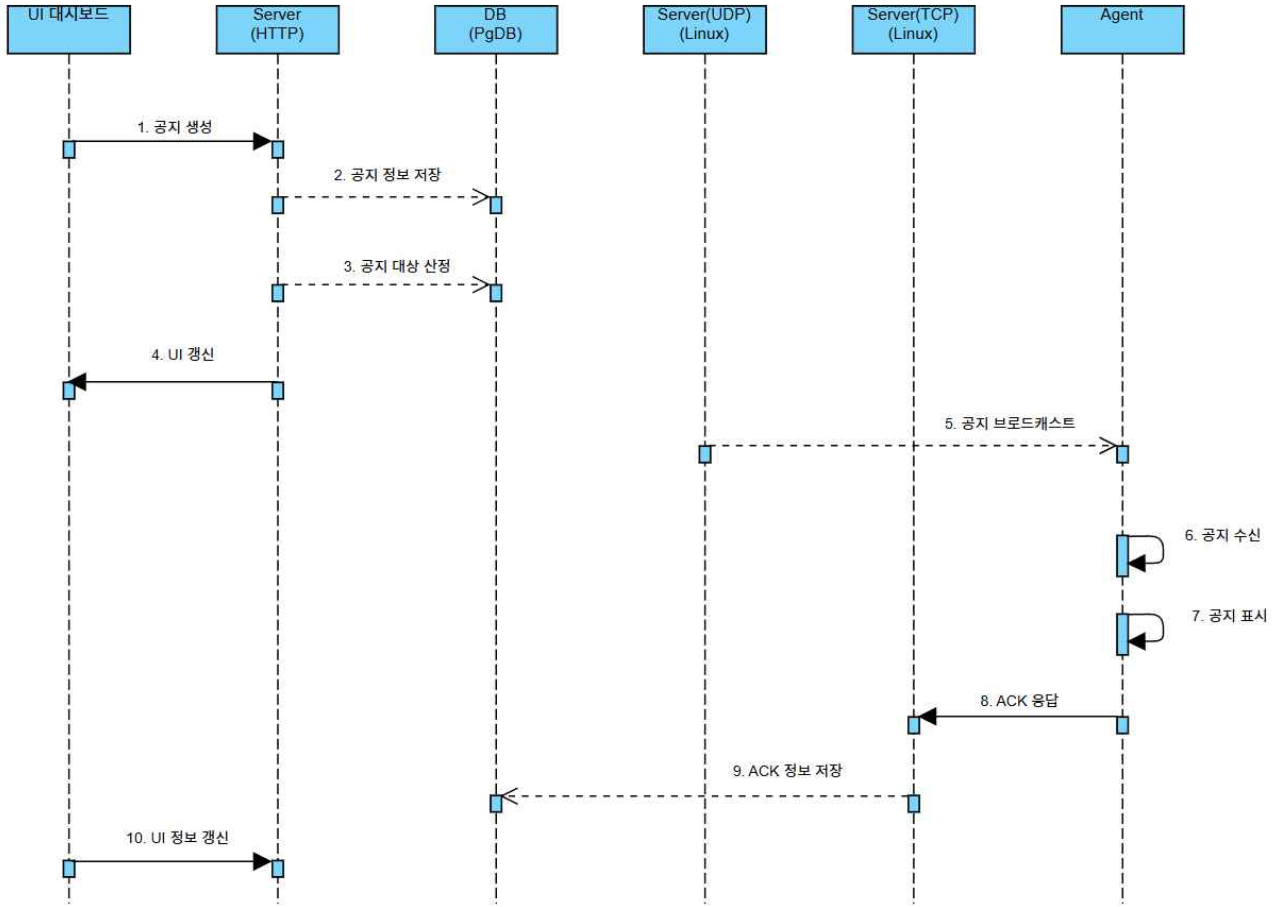
시퀀스 다이어그램
- 배포 생성 및 처리

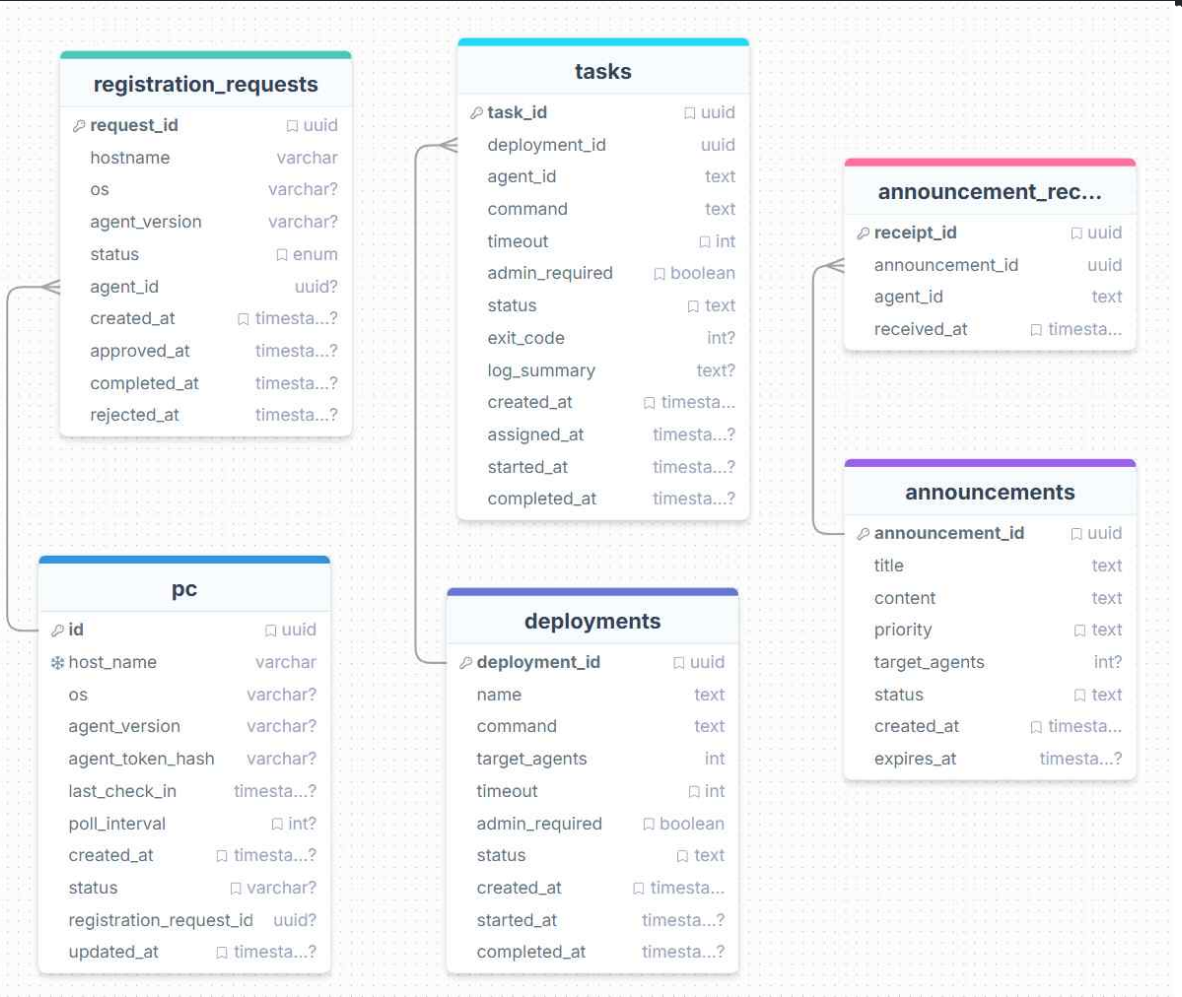


C/S 프로토콜
- 공지 생성 및 보내기 -

1. 공지 생성 (관리자 UI → HTTP API) [동기: HTTP 요청-응답]
2. 공지 정보 저장 (HTTP Server → DB) [비동기: async/await, announcements 테이블, status: "active"]
3. 공지 대상 산정 (HTTP Server → DB) [비동기: async/await, target_agents가 null이면 전체 에이전트 대상, pc 테이블 조회]
4. UI 정보 갱신 (HTTP Server → 관리자 UI) [동기: 1번의 응답, announcement_id 반환]
5. 공지 브로드캐스트 (UDP Server → Agent) [비동기: UDP 브로드캐스트, 포트 5502, 개별 전송 및 글로벌/서브넷 브로드캐스트]
6. 공지 수신 (Agent 내부) [비동기: UDP 포트 5502에서 수신 대기, 중복 수신 방지]
7. 공지 표시 (Agent 내부) [동기: Windows Toast Notification 표시, win10toast → MessageBox → PyQt6 순차적 fallback]
8. ACK 응답 (Agent → TCP Server) [동기: 요청 후 응답 대기, type: "notification_ack", announcement_id 전송]
9. ACK 정보 저장 (TCP Server → DB) [비동기: async/await, announcement_receipts 테이블, 멱등성 보장]
10. UI 정보 갱신 (관리자 UI → HTTP API) [동기: HTTP GET 요청, 공지 수신 통계 조회,
    target_count, received_count, receiptRate 계산]

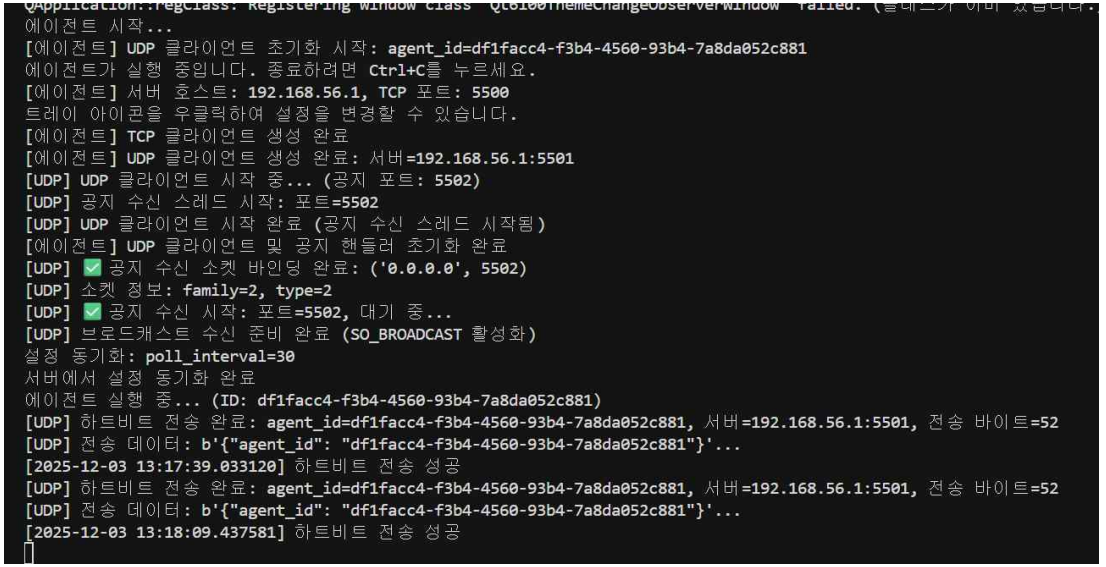
시퀀스 다이어그램
- 공지 생성 및 보내기



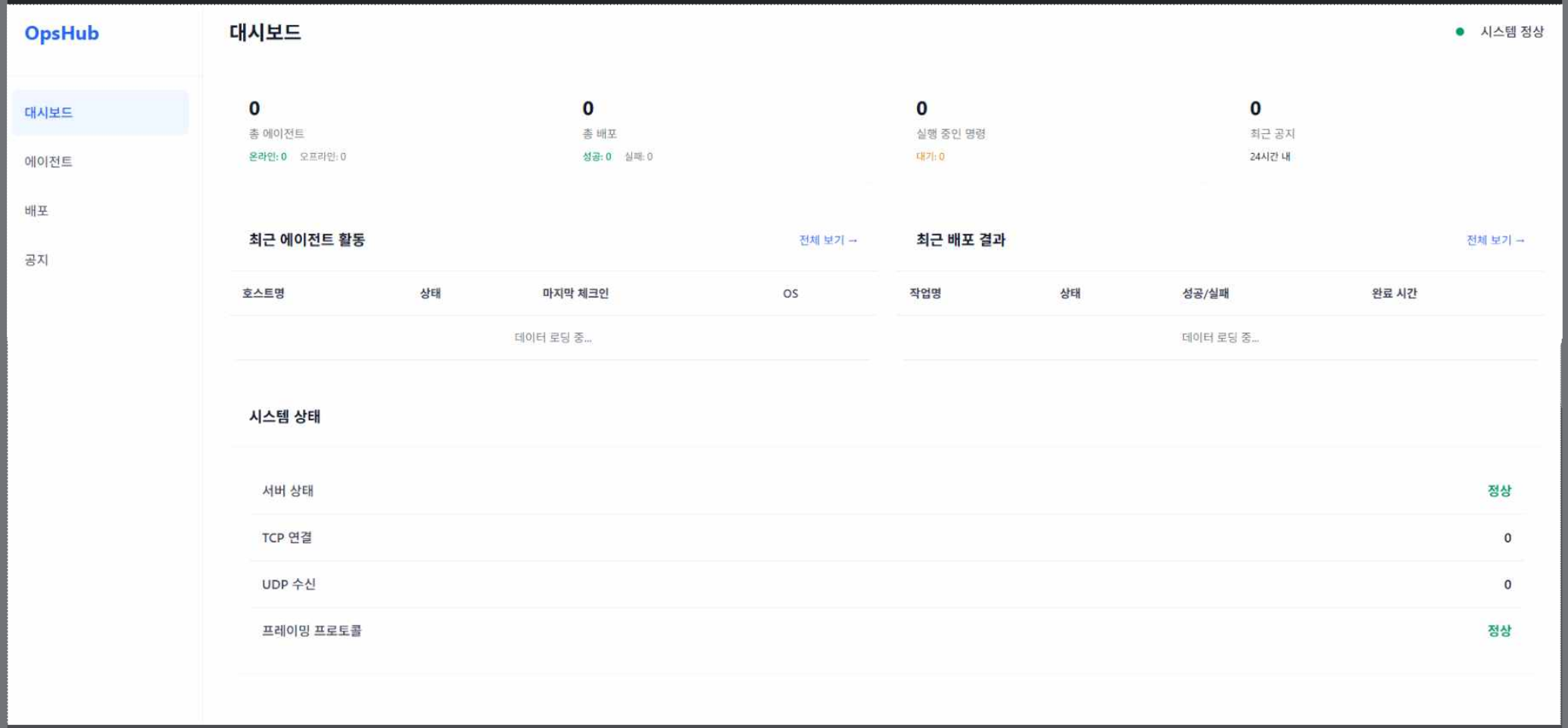

registration_requests
request_id uuid
hostname varchar
os varchar?
agent_version varchar?
status enum
agent_id uuid?
created_at timesta...?
approved_at timesta...?
completed_at timesta...?
rejected_at timesta...?
tasks
task_id uuid
deployment_id uuid
agent_id text
command text
timeout int
admin_required boolean
status text
exit_code int?
log_summary text?
created_at timesta...
assigned_at timesta...?
started_at timesta...?
completed_at timesta...?
announcement_rec...
receipt_id uuid
announcement_id uuid
agent_id text
received_at timesta...
announcements
announcement_id uuid
title text
content text
priority text
target_agents int?
status text
created_at timesta...
expires_at timesta...?
pc
id uuid
host_name varchar
os varchar?
agent_version varchar?
agent_token_hash varchar?
last_check_in timesta...?
poll_interval int?
created_at timesta...?
status varchar?
registration_request_id uuid?
updated_at timesta...?
deployments
deployment_id uuid
name text
command text
target_agents int
timeout int
admin_required boolean
status text
created_at timesta...
started_at timesta...?
completed_at timesta...?

DB 설계

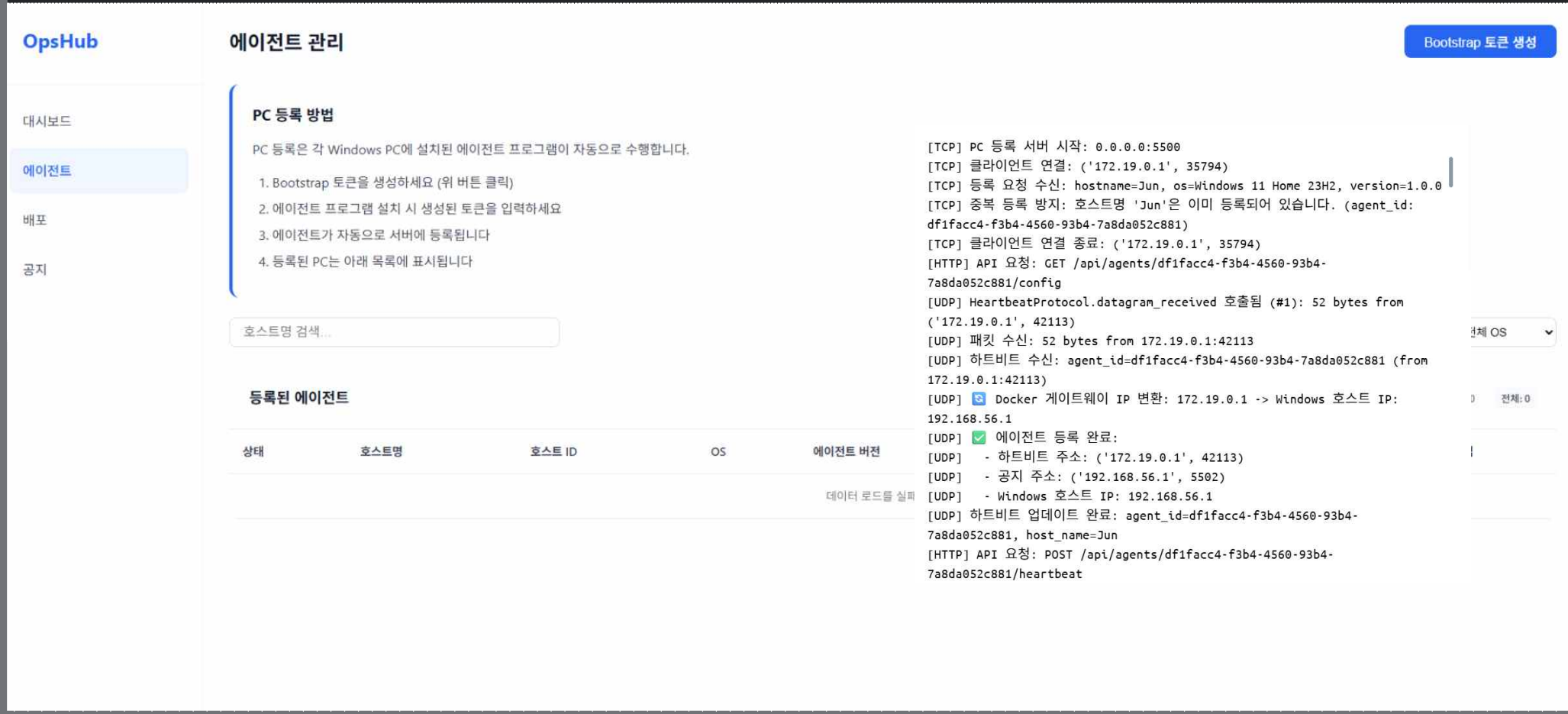
구현 결과


에이전트 시작...
[에이전트] UDP 클라이언트 초기화 시작: agent_id=df1facc4-f3b4-4560-93b4-7a8da052c881
에이전트가 실행 중입니다. 종료하려면 Ctrl+C를 누르세요.
[에이전트] 서버 호스트: 192.168.56.1, TCP 포트: 5500
트레이 아이콘을 우클릭하여 설정을 변경할 수 있습니다.
[에이전트] TCP 클라이언트 생성 완료
[에이전트] UDP 클라이언트 생성 완료: 서버=192.168.56.1:5501
[UDP] UDP 클라이언트 시작 중... (공지 포트: 5502)
[UDP] 공지 수신 스레드 시작: 포트=5502
[UDP] UDP 클라이언트 시작 완료 (공지 수신 스레드 시작됨)
[에이전트] UDP 클라이언트 및 공지 핸들러 초기화 완료
[UDP] ✅ 공지 수신 소켓 바인딩 완료: ('0.0.0.0', 5502)
[UDP] 소켓 정보: family=2, type=2
[UDP] ✅ 공지 수신 시작: 포트=5502, 대기 중...
[UDP] 브로드캐스트 수신 준비 완료 (SO_BROADCAST 활성화)
설정 동기화: poll_interval=30
서버에서 설정 동기화 완료
에이전트 실행 중... (ID: df1facc4-f3b4-4560-93b4-7a8da052c881)
[UDP] 하트비트 전송 완료: agent_id=df1facc4-f3b4-4560-93b4-7a8da052c881, 서버=192.168.56.1:5501, 전송 바이트=52
[UDP] 전송 데이터: b'{"agent_id": "df1facc4-f3b4-4560-93b4-7a8da052c881"}'...
[2025-12-03 13:17:39.033120] 하트비트 전송 성공
[UDP] 하트비트 전송 완료: agent_id=df1facc4-f3b4-4560-93b4-7a8da052c881, 서버=192.168.56.1:5501, 전송 바이트=52
[UDP] 전송 데이터: b'{"agent_id": "df1facc4-f3b4-4560-93b4-7a8da052c881"}'...
[2025-12-03 13:18:09.437581] 하트비트 전송 성공

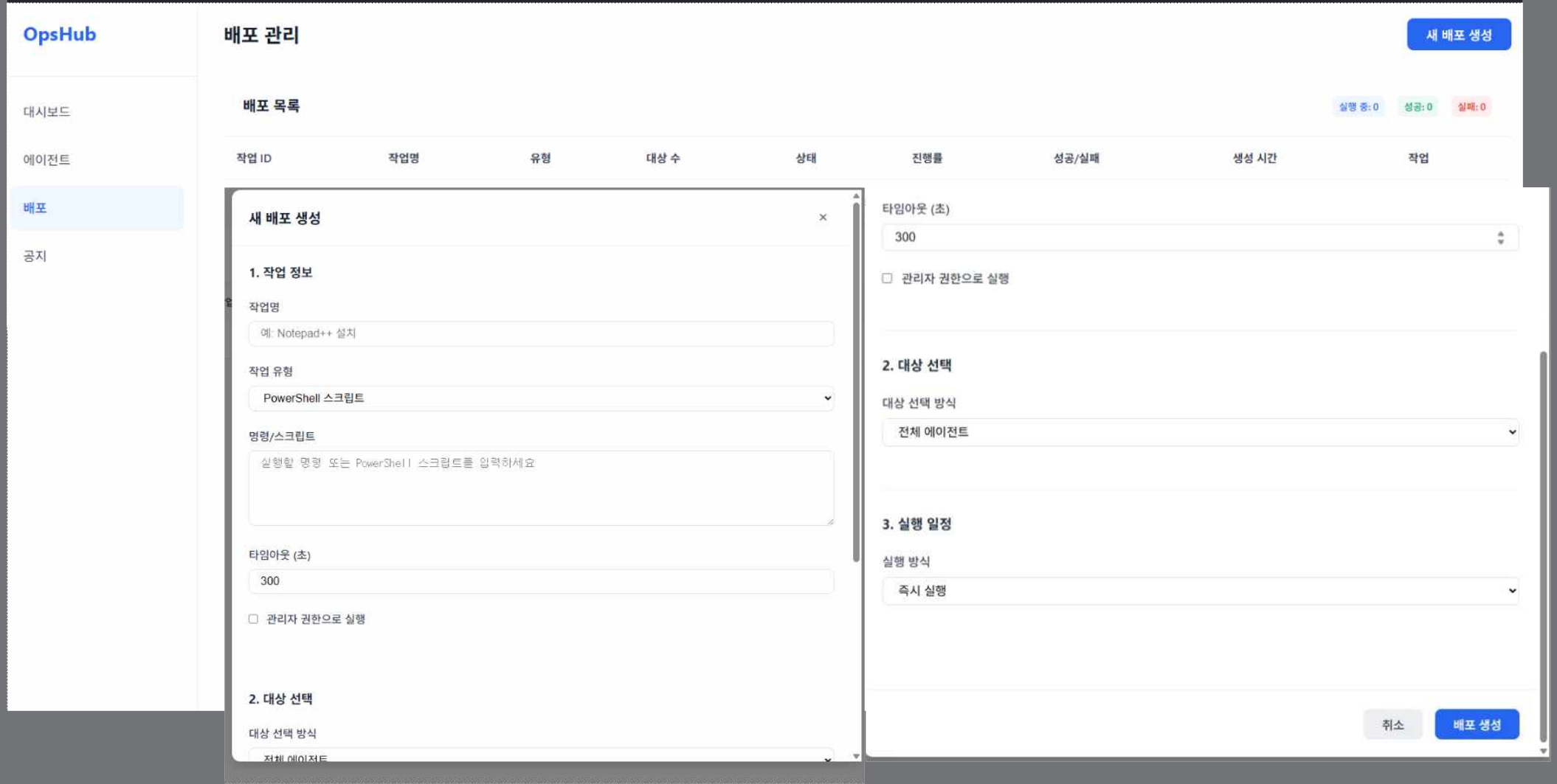
메인 대시보드


OpsHub
대시보드
에이전트
배포
공지
대시보드
시스템 정상
0
총 에이전트
온라인: 0 오프라인: 0
0
총 배포
성공: 0 실패: 0
0
실행 중인 명령
대기: 0
0
최근 공지
24시간 내
최근 에이전트 활동
전체 보기 →
호스트명
상태
마지막 체크인
OS
데이터 로딩 중...
최근 배포 결과
전체 보기 →
작업명
상태
성공/실패
완료 시간
데이터 로딩 중...
시스템 상태
서버 상태
정상
TCP 연결
0
UDP 수신
0
프레이밍 프로토콜
정상

에이전트 관리 UI


OpsHub
에이전트 관리
Bootstrap 토큰 생성
대시보드
에이전트
배포
공지
PC 등록 방법
PC 등록은 각 Windows PC에 설치된 에이전트 프로그램이 자동으로 수행합니다.
1. Bootstrap 토큰을 생성하세요 (위 버튼 클릭)
2. 에이전트 프로그램 설치 시 생성된 토큰을 입력하세요
3. 에이전트가 자동으로 서버에 등록됩니다
4. 등록된 PC는 아래 목록에 표시됩니다
[TCP] PC 등록 서버 시작: 0.0.0.0:5500
[TCP] 클라이언트 연결: ('172.19.0.1', 35794)
[TCP] 등록 요청 수신: hostname=Jun, os=Windows 11 Home 23H2, version=1.0.0
[TCP] 중복 등록 방지: 호스트명 'Jun'은 이미 등록되어 있습니다. (agent_id: df1facc4-f3b4-4560-93b4-7a8da052c881)
[TCP] 클라이언트 연결 종료: ('172.19.0.1', 35794)
[HTTP] API 요청: GET /api/agents/df1facc4-f3b4-4560-93b4-7a8da052c881/config
[UDP] HeartbeatProtocol.datagram_received 호출됨 (#1): 52 bytes from ('172.19.0.1', 42113)
[UDP] 패킷 수신: 52 bytes from 172.19.0.1:42113
[UDP] 하트비트 수신: agent_id=df1facc4-f3b4-4560-93b4-7a8da052c881 (from 172.19.0.1:42113)
[UDP] Docker 게이트웨이 IP 변환: 172.19.0.1 -> Windows 호스트 IP: 192.168.56.1
[UDP] 에이전트 등록 완료:
[UDP] - 하트비트 주소: ('172.19.0.1', 42113)
[UDP] - 공지 주소: ('192.168.56.1', 5502)
[UDP] - Windows 호스트 IP: 192.168.56.1
[UDP] 하트비트 업데이트 완료: agent_id=df1facc4-f3b4-4560-93b4-7a8da052c881, host_name=Jun
[HTTP] API 요청: POST /api/agents/df1facc4-f3b4-4560-93b4-7a8da052c881/heartbeat
호스트명 검색...
전체 OS
등록된 에이전트
전체: 0
상태
호스트명
호스트 ID
OS
에이전트 버전
데이터 로드를 실패

배포 관리 UI


OpsHub
대시보드
에이전트
배포
공지
배포 관리
새 배포 생성
배포 목록
실행 중: 0
성공: 0
실패: 0
작업 ID
작업명
유형
대상 수
상태
진행률
성공/실패
생성 시간
작업
새 배포 생성
1. 작업 정보
작업명
예: Notepad++ 설치
작업 유형
PowerShell 스크립트
명령/스크립트
실행할 명령 또는 PowerShell 스크립트를 입력하세요
타임아웃 (초)
300
관리자 권한으로 실행
2. 대상 선택
대상 선택 방식
전체 에이전트
타임아웃 (초)
300
관리자 권한으로 실행
2. 대상 선택
대상 선택 방식
전체 에이전트
3. 실행 일정
실행 방식
즉시 실행
취소
배포 생성

배포 관리 UI


OpsHub
대시보드
에이전트
배포
공지
배포 관리
새 배포 생성
배포 목록
작업 ID
작업명
유형
대상 수
상태
데이터
성공: 0
실패: 0
작업
[2025-12-03 13:19:40.345184] 하트비트 전송 성공
[작업] 작업 수신: task_id=17fbb986-f076-4adf-b192-65e6335e7d9b
작업 실행 시작: 17fbb986-f076-4adf-b192-65e6335e7d9b
명령: ipconfig
[작업] 결과 제출 성공: 17fbb986-f076-4adf-b192-65e6335e7d9b
[DeploymentService] 작업 시작: task_id=17fbb986-f076-4adf-b192-65e6335e7d9b
[DeploymentService] 작업 정보 조회 성공: task_id=17fbb986-f076-4adf-b192-65e6335e7d9b, command=ipconfig
[TCP] 작업 할당: agent_id=df1facc4-f3b4-4560-93b4-7a8da052c881, task_id=17fbb986-f076-4adf-b192-65e6335e7d9b
[TCP] 클라이언트 연결 종료: ('172.19.0.1', 40684)
[TCP] 클라이언트 연결: ('172.19.0.1', 40700)
[배포 상태] 작업 상태 확인: deployment_id=ed811a73-286f-463d-b81f-8e5e8e3b4b5b, 전체=1, 완료=1
[배포 상태] 배포 완료: deployment_id=ed811a73-286f-463d-b81f-8e5e8e3b4b5b, status=completed, 실패=0
[배포 상태] 배포 상태 업데이트 완료: deployment_id=ed811a73-286f-463d-b81f-8e5e8e3b4b5b
[TCP] 작업 결과 저장 완료: agent_id=df1facc4-f3b4-4560-93b4-7a8da052c881, task_id=17fbb986-f076-4adf-b192-65e6335e7d9b, status=success
[TCP] 클라이언트 연결 종료: ('172.19.0.1', 40700)
[HTTP] API 요청: POST /api/deployments
[HTTP] 배포 생성 요청 데이터: {'name': '테스트', 'type': 'powershell', 'command': 'ipconfig', 'timeout': 300, 'target_type': 'all', 'schedule_type': 'now', 'target_agents': [], 'admin_required': False}
[HTTP] 전체 에이전트 선택: 1개 에이전트
[HTTP] 배포 생성 파라미터: name=테스트, command=ipconfig, target_agents=['df1facc4-f3b4-4560-93b4-7a8da052c881'], timeout=300, admin_required=False
[배포 생성] 작업 생성 시작: deployment_id=ed811a73-286f-463d-b81f-8e5e8e3b4b5b, target_agents=1개
[배포 생성] 작업 생성 성공: task_id=17fbb986-f076-4adf-b192-65e6335e7d9b, agent_id=df1facc4-f3b4-4560-93b4-7a8da052c881
[배포 생성] 모든 작업 생성 완료: deployment_id=ed811a73-286f-463d-b81f-8e5e8e3b4b5b
[HTTP] 배포 생성 성공: deployment_id=ed811a73-286f-463d-b81f-8e5e8e3b4b5b


PC 등록 실행 프로그램
OpsHub
공지 관리
새 공지 발송
대시보드
에이전트
배포
공지
발송된 공지
전체: 0
수신 성공: 0
공지 ID
제목
전송 방식
대상
수신률
발송 시간
상태
작업
데이터 로딩 중...
새 공지 발송
공지 제목
예: 오늘 20:00 유지보수 예정
공지 내용
공지 내용을 입력하세요
전송 방식
브로드캐스트 (전체 서브넷)
우선순위
일반
트레이 알림 표시
로그에 기록
취소
공지 발송

PC 등록 실행 프로그램

OpsHub

공지 관리

새 공지 발송

전체: 0 수신 성공: 0

수신률	발송 시간	상태	작업

```
[UDP] 전송 데이터: b'{"agent_id": "df1facc4-f3b4-4560-93b4-7a8da052c881"}'...
[2025-12-03 13:20:11.139807] 하트비트 전송 성공
[UDP] ✅ UDP 패킷 수신 (#1): 207 bytes from 192.168.56.1:51686
[UDP] 수신 데이터 (처음 100바이트): b'{"type": "announcement", "announcement_id": "f856dce7-4bae-40e2-a471-a8a5649fa154",
"title": "\xed\x85\x8c\xec\x8a\xa4'
[UDP] ✅ 공지 데이터 파싱 완료:
[UDP]   - type: announcement
[UDP]   - title: 테스트
[UDP]   - announcement_id: f856dce7-4bae-40e2-a471-a8a5649fa154
[UDP] ✅ 공지 수신 확인: 테스트 from ('192.168.56.1', 51686)
[UDP] 공지 콜백 호출 시작...

================================================
[공지] 테스트
================================================
안녕하세요
================================================
```

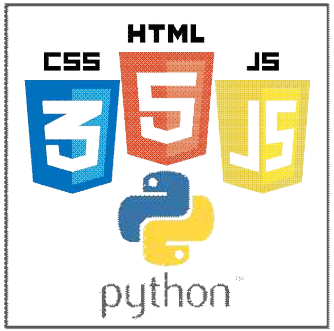
```
[HTTP] API 요청: POST /api/announcements
[UDP] 연결된 에이전트 목록: [('192.168.56.1', 5502)]
[UDP] Windows 호스트 IP: 192.168.56.1
[UDP] 공지 전송 시도: 192.168.56.1:5502
[UDP] ✅ 개별 전송 성공: ('192.168.56.1', 5502)
[UDP] 브로드캐스트 전송: ('255.255.255.255', 5502)
[UDP] Windows 호스트 서브넷 브로드캐스트 전송: ('192.168.56.255', 5502)
[UDP] Docker 서브넷 브로드캐스트 전송: ('172.19.0.255', 5502)
[UDP] 공지 전송 완료: 브로드캐스트 + 1/1개 에이전트에 개별 전송
[TCP] 클라이언트 연결: ('172.19.0.1', 60616)
[TCP] 공지 ACK 저장 완료: agent_id=df1facc4-f3b4-4560-93b4-7a8da052c881,
notification_id=f856dce7-4bae-40e2-a471-a8a5649fa154
[TCP] 클라이언트 연결 종료: ('172.19.0.1', 60616)
```

```
[공지] ACK 전송 완료: announcement_id=f856dce7-4bae-40e2-a471-a8a5649fa154
[UDP] ✅ 공지 콜백 호출 완료
[UDP] ✅ UDP 패킷 수신 (#2): 207 bytes from 192.168.56.1:51687
[UDP] 수신 데이터 (처음 100바이트): b'{"type": "announcement", "announcement_id": "f856dce7-4bae-40e2-a471-a8a5649fa15
"title": "\xed\x85\x8c\xec\x8a\xa4'
[UDP] ✅ 공지 데이터 파싱 완료:
[UDP]   - type: announcement
[UDP]   - title: 테스트
[UDP]   - announcement_id: f856dce7-4bae-40e2-a471-a8a5649fa154
[UDP] ✅ 공지 수신 확인: 테스트 from ('192.168.56.1', 51687)
[UDP] 공지 콜백 호출 시작...
[공지] 이미 수신한 공지입니다: f856dce7-4bae-40e2-a471-a8a5649fa154
[UDP] ✅ 공지 콜백 호출 완료
WNDPROC return value cannot be converted to LRESULT
TypeError: WPARAM is simple, so must be an int object (got NoneType)
[공지] ✅ Windows Toast Notification 표시 완료 (win10toast): 테스트
```

개발 환경

운영체제

윈도우11, 우분투



언어

html, css, js, python

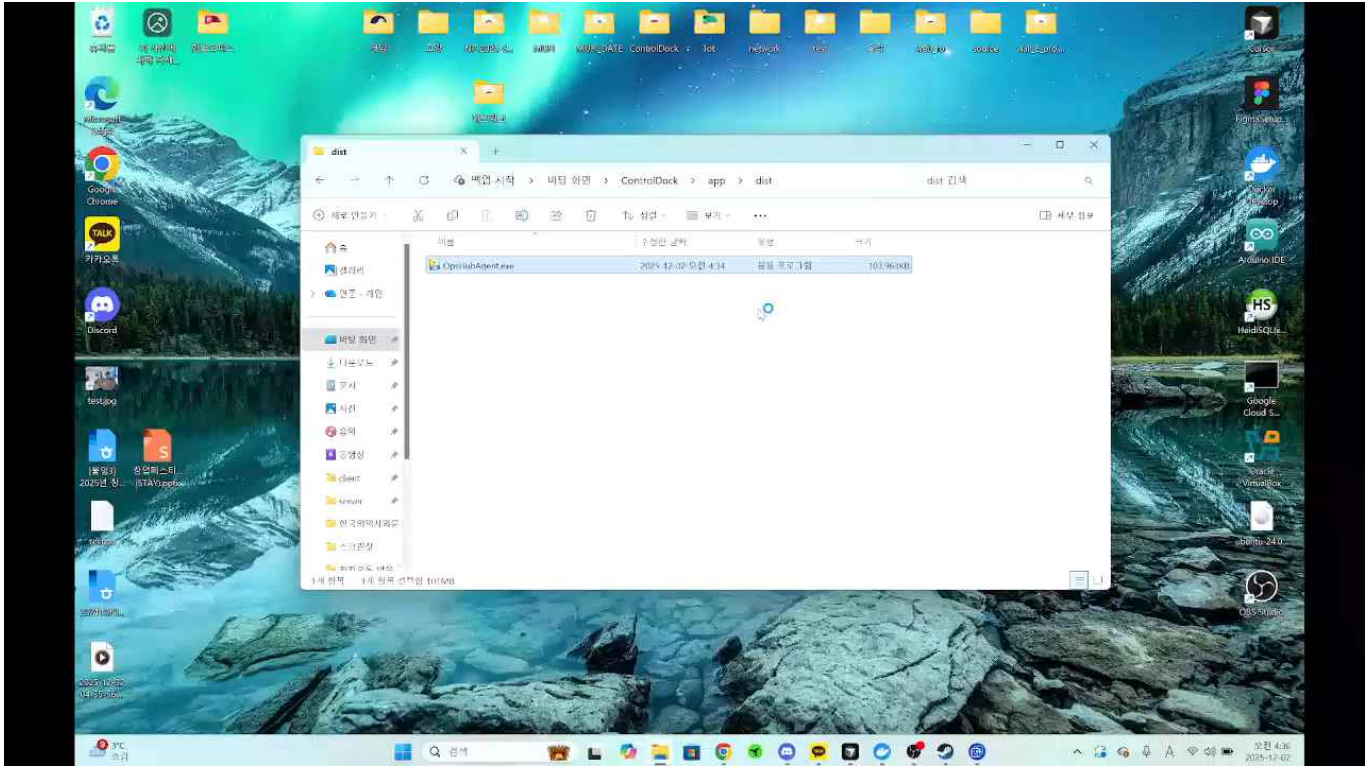
DBMS

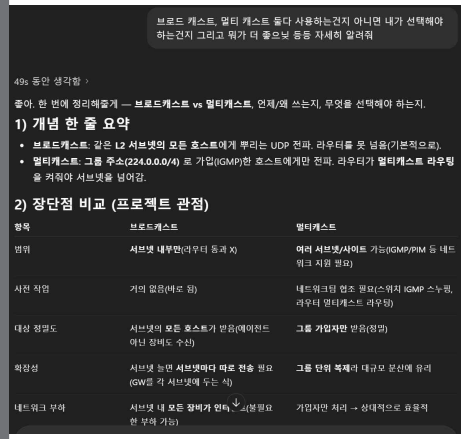
PostgreSQL

가상 컨테이너

docker

구현 결과





AI 적용 방법 및 사례&결론

ChatGPT

아이디어 정리 및 요구사항 구조화

프로토콜/기능 아이디어 구체화

=> 설계 단계에서의 시행착오 감소

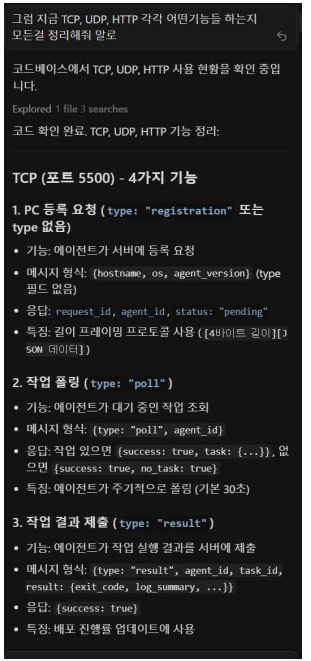
+

Cursor

단위 테스트 코드 생성

리팩토링 제안

=> 반복 작업감소, 코드 품질 유지



핵심 비즈니스 로직과 네트워크 설계에 집중

설계·구현 전 과정에서 개발자의 생산성과 판단력을 보조하는 강력한 파트너

역할분담 및 느낀점

https://github.com/shinyeonjun/ControlDock

이호현

데이터베이스 설계 및 백엔드 서비스 로직

네트워크 프로그래밍이라는 과목을 접하면서 생각보다 서버-클라이언트 구조 간의 통신이 어렵다는 것을 느꼈습니다. 그래서 이번 서버-클라이언트 구조의 프로젝트를 직접 설계·개발해가는 과정 속에서 많은 것을 배울 수 있었습니다. 쉽지 않은 주제였지만 결국 완성할 수 있어 뿌듯합니다.

신연준

백엔드 서버 구현 (HTTP/TCP/UDP 서버)

백엔드 서버를 구현하면서 HTTP, TCP, UDP 등 서로 다른 통신 방식이 요구하는 구조와 제약을 비교하며 설계해야 한다는 점이 인상적이었습니다. 구현한 서버가 정상적으로 동작하고 팀의 다른 모듈과 무리 없이 통신하는 것을 확인했을 때 큰 보람을 느꼈습니다. 이번 프로젝트를 통해 백엔드 전반에 대한 이해가 한층 넓어졌다고 느낍니다.

윤동현

에이전트 클라이언트 구현 및 배포 시스템

이번 프로젝트에서 에이전트 클라이언트 구현과 배포를 담당하면서, 네트워크 구조가 단순히 요청과 응답만으로 이루어지는 것이 아니라 다양한 예외 상황과 상태 전이를 세밀하게 고려해야 한다는 점을 깊이 체감했습니다. 또한 경험을 통해 네트워크 프로그래밍 전반에 대한 이해가 한층 확장된 것 같아 의미 있는 시간이었습니다.

서하제

프론트엔드 웹 UI 구현

프론트엔드 웹 UI를 구현하면서 사용자 경험을 고려한 화면 구조 설계가 얼마나 중요한지 다시 한 번 느낄 수 있었습니다. 또한 이번 프로젝트를 통해 프론트엔드와 서버 측 로직이 어떻게 연결되는지 더 깊이 이해할 수 있었고, 실제 작동하는 UI를 구현해보며 많은 성장을 경험했습니다.

1조 이호현, 서하제, 신연준, 윤동현

원격 모니터링 및 배포 시스템

감사합니다