

송철진

☎ +821026922709 @songcj92@gmail.com

Node.js 기반 서비스 최적화와 AWS 자동화 배포 경험을 바탕으로 서비스 안정성과 효율성 개선에 강점을 가진 3년차 백엔드 개발자입니다.

- Node.js, NestJS 기반 백엔드 개인 프로젝트 배포 경험
- Node.js, Express.js 기반 RESTful API 설계·개발·배포·유지보수 경험
- MySQL 모델링, Redis, RabbitMQ, AWS EC2/RDS/S3, WebSocket 등 인프라 연계 기반 백엔드 서비스 구축 경험
- 서버 리소스 관리, SSL 인증서 갱신, 로그 기반 모니터링 등 DevOps 업무 수행
- 협력사 및 타 직군(기획/디자인/QA)과 협업하여 기능 검증 경험
- 문제 해결 과정과 기술 지식을 기록·공유하는 개발 문화 주도
- L2/L3 스위치 기반 네트워크 구축 및 NMS 시스템 QA 경험

경력 5년 4개월



(주)오픈웍스

2025.09 - 재직중 (4개월) | 정규직 | 백엔드 개발 | 대리

AI CCTV 영상 다운로드 및 조회/만료 파일 삭제 기능 구현

2025.09 - 2025.09 | 백엔드 개발 | 대리

[프로젝트 개요]

AI CCTV가 차량 출입, 화재, 쓰러짐 등 이상 상황을 자동 감지하여 이벤트·썸네일·영상을 서버로 전송하고, 관제자가 이를 실시간으로 조회·알림·리플레이할 수 있도록 지원하는 기능을 구축하는 프로젝트입니다.

[팀 구성]

• Front-End 1명, Back-End 1명

[기술스택]

- 백엔드 개발: Node.js (Express.js), MySQL, WebSocket
- 개발 및 배포 환경: Windows Server, Nginx, PM2

[주요 역할]

- AI CCTV 이벤트(화재, 차량 감지 등) 발생 시 WebSocket으로 실시간 이미지·영상 다운로드 기능 구현
- 영상은 pre-time + post-time 기준으로 일정 시간 경과 후 다운로드되도록 지연 제어 로직 개발
- 다운로드 실패 시 최대 3회 재시도 로직 구현
- 파일을 image/video 디렉터리로 구분하고, 발생 연·월 기준으로 디렉토리 구조 개선
- 서버 리소스 관리를 위한 cron 기반 삭제 스크립트 구현(3개월 보관 정책)
- 관제 솔루션에서 빠르게 조회할 수 있도록 nginx 라우팅 디렉터리 설정
- 이벤트 다운로드 이력을 log 테이블에 저장하여 관제 서비스 이력관리 기능과 연동

[성과 및 배운점]

- 실시간 관제 성능 개선(WebSocket 전환 및 지연 최소화)

- 기존: API 폴링 기반으로 서버 부하 및 최대 1분 지연 발생하여 실시간 관제 서비스에 어려움
- 개선: WebSocket 실시간 이벤트 수신, 이미지 즉시 다운로드, 영상은 약 20초 대기 후 다운로드
- 결과: 실시간 이벤트 로깅과 신속한 조회 가능

- 정적 파일 저장 구조 최적화(디렉터리 샤딩 기반 성능 개선)

- 기존: image/video 파일이 하나의 디렉터리에 혼합 저장되고 파일 수가 지속적으로 누적됨에 따라 OS 파일 탐색 비용 증가 및 조회 지연 발생
- 개선: image/video 유형별 분리 + 연·월 단위 디렉터리 구조 정립으로 디렉터리당 파일 수를 안정적으로 분산시키고 파일시스템 lookup 성능 최적화
- 결과: 조회 속도 향상, 디스크 정리·백업·삭제 등 운영 작업의 관리 단순화, 대용량 파일 증가에도 성능 저하 없이 안정적인 서비스 제공 가능

- 시스템 아키텍처 문서화 및 내부 공유 체계 개선

- 기존: 시스템 구성 문서 미비로 부서 간 소통 및 신규 인력 온보딩·유지보수 과정에서 정보 단절과 작업 지연 발생
- 개선: 단말·서버 구조 및 전체 데이터 플로우를 시퀀스 다이어그램·구조도 등으로 도식화하여 표준 문서로 정리
- 결과: 내부 공유 활성화 및 유지보수 용이성 향상, 신규 인력 투입 시 빠른 이해 가능, 장애 대응 속도 개선

비상벨 연동 TTS/Live 방송 기능 구현

2025.10 - 2025.11 | 백엔드 개발 | 대리

[프로젝트 개요]

비상벨·TTS·마이크 방송 시스템을 관제 솔루션과 연동하여 웹에서 실시간 방송 제어와 단말 관리를 수행할 수 있도록 구축한 프로젝트입니다.

[팀 구성]

- Back-End 1명

[기술스택]

- 백엔드 개발: Node.js (Express.js), MySQL, Redis, WebSocket
- 개발 및 배포 환경: Windows Server, Nginx, PM2

[주요 역할]

- 새서울정보통신 비상벨 방송시스템과 연동하여 TTS·마이크 방송 기능을 관제 솔루션에 통합
- 방송 단말 목록 조회 및 상태(on/off/비상호출중/방송중/미등록) 관리 기능 구현
- 한글/영어/일시정지 규칙을 기반으로 TTS 텍스트 등록·조회·수정·삭제 기능 구현
- TTS 방송 시작 기능 및 마이크 방송 시작/종료 기능 구현
- 비상 상황 시 모든 단말로 일괄 방송할 수 있는 그룹 방송 기능 구현
- 방송 시스템 전체 구조 및 케이스별 흐름을 도식화하여 팀 내 공유
- 새서울 엔지니어 및 사내 기술팀과 협업하여 장비 제약, 라이선스 정책, 운영 환경 등을 분석하여 요구사항 확정

[성과 및 배운점]

- 단말 상태 고도화

- 기존: 단말 상태가 on/off 두 가지로만 표시됨
- 개선: 비상호출중 / 방송중 / 미등록 등 세분화된 상태 추가 정의 및 WebSocket 실시간 갱신 적용

- 결과: 상황별 실시간 알림 팝업 제공 및 방송 가능 여부 명확화 → 관제 효율성 향상

• TTS 문구 생성 자동화

- 기존: 관리자가 새서울 TTS 코드 규칙을 모두 숙지해야만 방송 문구 작성 가능

- 개선: 한글/영어 변환, 일시정지시간을 입력하면 실행용/표시용 포맷을 자동 생성하는 API 구현

- 결과: 비전문가도 손쉽게 TTS 문구 생성 가능, 서버의 포맷 오류·에러 핸들링 대폭 감소

• 요구사항 불명확·정보 불일치 문제 해결

- 기존: 여러 팀과 협업 시 정보 불일치로 개발 범위가 계속 변경되는 문제

- 개선: 전체 TTS/마이크/비상벨/방송 서버 구조를 직접 도식화하고, 장비 실험 기반으로 기술적 제약사항을 명확히 정리 (라이선스 필요 여부, 에이전트 설치 위치, 신호 흐름 등)

- 결과: 개발 방향을 명확하게 정립하고 프로젝트 주도권 확보, 실 개발 범위 확정

• 실장비 기반 문제 진단 및 방송 구조 검증

- 기존: 마이크 방송 테스트 시 원격데스크탑 환경·USB 마이크 등으로 인해 원인 파악 어려움

- 개선: TRS 마이크 연결 방식, 에이전트 설치 위치, 단말->앰프->스피커 실제 배선 등 실장비 환경에서 단계별 원인 분석 진행

- 결과: 방송 서버->단말 간 신호 흐름을 정확히 검증하고, 실제 동작 가능한 아키텍처 확립

• 에이전트 기반 방송 제약 해결

- 기존: 웹 브라우저 기반 관제 솔루션에서 어느 PC가 방송 가능한지 식별 불가

- 개선: 관리자 계정과 agent id/mic id 매핑 구조 설계, DB 기반 방송 가능 PC 지정 기능 구현

- 결과: 웹 기반 서비스에서도 안정적인 방송 제어 가능, 운영 관리 체계 수립

Amons 관제솔루션 서버 유지보수

2025.09 - 2025.12 | 백엔드 개발 | 대리

[프로젝트 개요]

[팀 구성]

• Front-End 3명, Back-End 3명

[기술스택]

• 백엔드 개발: Node.js (Express.js), MySQL, Redis, WebSocket

• 개발 및 배포 환경: Windows Server, Ubuntu, Nginx, PM2

[주요 역할]

• pm2 기반 Node.js 서비스 유지보수 및 배포

• Windows 서버 상태 모니터링(CPU/메모리/디스크) 및 로그 분석

• pm2 프로세스 구조 정리 및 충돌 해결

• MySQL 커넥션 처리 개선 및 쿼리 튜닝

• 긴급 장애 대응 및 운영 자동화 개선

[성과 및 배운점]

• 원격 접속 제한 해소 및 장애 대응 체계 개선

- 기존: 원격 데스크탑 1세션만 허용되어 동시 접근 불가, 장애 대응 지연

- 개선: 원격서버 SSH 활성화

- 결과: 여러 명이 동시에 서버 접근 가능, 긴급 장애 대응 속도 향상

• PM2 프로세스 충돌 방지 및 재부팅 관리 구조 정립

- 기존: 서버 재부팅 시 shell:startup의 bat 파일로 인해 삭제한 pm2 프로세스가 재실행되어 충돌 발생

- 개선: bat 파일 제거 후 pm2 save/resurrect/flush 구조로 통합 관리

- 결과: 재부팅 시 포트 충돌 및 중복 프로세스 문제 해결

• 서버 다운 원인 진단 및 안정화(리소스 모니터링·DB 세션·메모리 누수·백신 이슈 개선)

- 기존: Windows 서버가 약 5일 주기로 CPU 100%에 도달하며 다운되는 문제가 반복되었고, 원격 데스크톱 단일 세션 환경에서는 블랙스크린·무응답 현상으로 로그/리소스 확인이 어려워 IDC 담당자에게 재부팅을 반복 요청해야 하는 상황이 지속됨

- 개선:

+ SSH 접속을 활성화하여 여러 명이 동시에 서버에 접속해 로그 확인·원격 재부팅 등 장애 대응이 가능하도록 개선

+ 작업 관리자/프로세스 목록을 통해 CPU 사용률 상위 프로세스(1순위 ekrn, 2순위 MySQL) 식별

+ 10초 간격으로 CPU/Memory/Disk/Network 상태를 DB 테이블에 로깅하고 7일 보관 후 자동 삭제하는 모니터링 기능 구현

+ 수집한 로그를 기반으로 CPU 사용률을 그래프화하여 약 5일 주기로 40% → 100%까지 서서히 증가하는 패턴 분석

+ 모든 서비스의 MySQL 사용 코드를 재검토하여 connection.release 위치 오류 수정(sleep 세션 431 → 186 감소)

+ Promise 기반 API에서 응답 후 return 누락으로 인한 이중응답·메모리 누수 가능 구간을 점검·수정

+ ekrn(ESET 백신) 프로세스에 대해 스캔 범위 조정, 로그 수집·본사 분석 의뢰, 버전 업그레이드 등을 단계적으로 수행했으나 이슈 지속됨을 확인

+ IDC 담당자와 협의해 V3 백신 POC 버전을 1개월간 적용(ESET 제거)하여 동일 CPU 증가 현상 재현 여부를 검증하고, 문제 없음과 비용 절감(기존 대비 약 40%)을 확인

- 결과: CPU 급상승 및 주기적 서버 다운의 주 원인이 백신(ESET) 이슈임을 단계적으로 특정하고, V3로의 교체 방향과 모니터링 체계를 확립하여 재발 시에도 로그 분석·원격 재부팅이 가능한 안정적인 서버 운영 기반을 마련함



VVOLT Lab (개인 프로젝트)

2025.07 - 2025.08 (2개월) | 프리랜서 | 백엔드 개발

중고 거래 플랫폼 서비스 리팩토링

2025.07 - 2025.08 | 백엔드 개발

- 프로젝트 개요

- 기존 중고거래 플랫폼을 NestJS, Docker, AWS 기반으로 리팩토링

- 서비스 개발·배포 전 과정을 단기간 내 경험하는 데 집중

- 팀 구성: 초기 Front-End 4명, Back-End 3명 → 리팩토링 시 1인 개발

- 기술스택: Node.js (Express.js, NestJS), TypeORM, MySQL, React, Docker, AWS EC2/RDS/S3, GitHub Actions

- 주요 개발 내용

- 카카오 소셜 로그인/회원가입 구현

- 인터셉터 구조 기반 AWS S3 단일/다중 이미지 업로드 구현

- JWT 인증 기반 상품 등록/조회/삭제/찜하기 등 주요 거래 기능 구현

- 공통 응답 포맷 유틸 및 전역 예외 필터 구현

- Docker Compose 기반 개발/프로덕션 환경 통일

- AWS EC2/RDS/S3 배포 환경 구성 및 가비아 DNS 연결

- Nginx Reverse Proxy 설정 및 Winston 기반 로그 시스템 구축

- 상품 삭제 오류 방지 및 거래 UX 개선 (FE)

- 성과
 - NestJS 기반 REST API 개발부터 AWS 실서비스 배포까지 백엔드 전 과정 단독 수행
 - API 응답 포맷 일원화로 프론트엔드 협업 효율 향상 및 코드 재사용성 증가
 - Swagger 기반 API 문서화로 백엔드-프론트엔드 간 커뮤니케이션 시간 절감
 - JWT 인증 가드 공통 모듈화로 일관된 보안 처리 구조 확립
 - Docker 기반 환경 구성으로 로컬/운영 서버 간 환경 차이 문제 사전 방지
 - GitHub Actions + PM2 CI/CD 자동화 구축 및 클라우드 인프라 운영 경험 확보



씨케이신택 주식회사

2023.07 - 2025.07 (2년 1개월) | 정규직 | 백엔드 개발자 | 사원

재난 신고 및 알림 서비스 서버 구축

2024.01 - 2025.06 | 백엔드 개발자 | 사원

[프로젝트 개요]

Rainbird(레인버드)는 전처리 서버와 리전 서버로 구성된 위성자료 기반 재난 감지·알림 플랫폼으로, 각국 사용자에게 실시간 푸시알림과 재난 정보를 제공하는 글로벌 대응형 프로젝트입니다

[팀 구성]

- 초기: 앱 개발자 3명, Back-End 4명(Back-End로 참여)
- 현재: 앱 개발자 2명, Back-End 1명(Back-End로 참여)

[기술스택]

- 백엔드 개발: Node.js (Express.js), MySQL, Redis, RabbitMQ, Python
- 개발 및 배포 환경: OCI(Ubuntu), PM2
- 기타: Git, Winston(로깅)

[주요 역할]

- 전처리 서버와 리전 서버 ERD 설계 및 클라우드 인프라 기반 구축
- Node.js에서 위성자료 Crop을 Python 라이브러리로 비동기 호출하고, 처리 결과를 DB에 기록하는 비동기 배치 프로세스 설계
- 위성/유저 재난 신고에 의한 5개 언어별 알림 및 위치 기반 재난 목록 서비스 구현
- BMKG, APCC 등 기상청 오픈 API 연동하여 이상기후 제공 서비스 구현
- 위성자료 기반 캄보디아 폭염 개인화 알림 및 다국어 행동요령 조회 기능 구현
- 사용자 권한별 재난 신고 및 알림 기능 구현
- PM2, Git tag 기반 수동 배포 환경 운영
- 기획팀, 앱 개발팀과 유기적으로 협업하여 개발기 앱 기반 수동 통합/E2E 테스트 검증
- 정기적인 서버 리소스 점검, SSL인증서 갱신

[성과]

- 월간 트래픽 42% 절감 시스템 구축
 - 기존: 전구(전 지구) 영역 전체 파일 전송 시 월간 무료 트래픽 한도(10TB)의 초과 예상.
 - 개선: 필요한 영역만 위성자료를 추출 및 전송하여, 전송량 최소화하는 전처리 시스템 구현
 - 결과: 월간 파일 전송량 약 42% 절감, 무료 트래픽 한도 내에서 안정적인 운영 가능, 추가 네트워크 비용 방지

- N+1 쿼리 제거로 댓글 목록 성능 1s -> 120ms 개선
 - 기존: 신고된 재난의 댓글 조회 시 N+1 쿼리로 API 응답 1s 이상 소요
 사용자 프로필/신고 사진을 file 테이블에서 매 댓글마다 개별 조회.
 - 개선: comment · user · file 테이블을 JOIN으로 한 번에 조회하도록 쿼리 리팩터링.
 - 결과: 불필요한 쿼리 제거로 응답 지연 해소(1s -> 120ms), 목록 조회 성능 및 DB 부하 개선.
- 재난 알림 중복 표시 최소화 로직 개선
 - 기존: 모니터링 반경 내 동일 시간/타입 재난 발생 시, 알림 수신 사용자 수만큼 중복 표시 -> 피드백(댓글·사진) 분산 문제 발생
 - 개선: 동일 시간·동일 타입 재난의 경우, 목록 조회 시 가장 가까운 1개의 재난만 대표로 표시되도록 로직 최적화
 - 결과: 중복재난 표시 최대 80% 감소(5개 -> 1개), 사용자 경험 개선 및 피드백 집중화 -> 서비스 활용도 향상
- 서버 해킹 대응 및 복구 경험
 - 기존: 사용자 수 대비 과도한 클라우드 청구 비용 발생. CPU 과부하 및 아웃바운드 트래픽 50Mbps 이상 발생.
 - 대응 및 개선:
 - htop으로 과부하 프로세스 탐지 및 종료
 - iftop으로 중국 통신사 IP 확인 후 차단
 - rkhunter로 악성코드 검사 및 제거
 - 익일 동일 증상 발생 -> 고객사 서버 관리자와 협의하여 서버 중단 및 신규 서버 생성
 - OpenSSH 버전 업그레이드 및 계정 비밀번호 변경
 - 결과: 보안 취약점 제거, 악성 트래픽 차단으로 추가 비용 방지 및 서버 보안 강화.

IoT 냉장고 자가진단 및 시뮬레이션 서버 구축

2024.09 - 2025.06 | 백엔드 개발자 | 사원

[프로젝트 개요]

IoT 기반 상업용 냉장고의 누적 수신된 센서/상태 정보를 기반으로 11종 진단 케이스 알림을 전송하여 사용자가 제품의 이상을 감지하고 문제를 사전예방하기 위한 자가진단 서비스 개발 프로젝트입니다

[팀 구성]

Front-End 2명, Back-End 2명(Back-End 개발자로 참여)

[기술스택]

- 백엔드 개발: Node.js (Express.js), MySQL, Redis, RabbitMQ, Mosquitto
- 개발 및 배포 환경: AWS EC2(Ubuntu), PM2
- 기타: Git, jQuery(시뮬레이터 서버), Winston(로깅)

[주요 역할]

- 사용자가 냉장고 센서/상태 이상을 감지하고 식재료 이상 문제를 사전 예방하기 위한 각 모델별 11종 진단 케이스의 자가진단 로직 구현
- 서버 성능 최적화 및 고속 처리를 위해 센서/상태 데이터를 DB 저장/조회 방식이 아닌 고정길이 순환 버퍼 메모리 기반 저장/조회 후 진단 조건값 계산 기능 구현
- 12종 모델별 11종 진단 케이스의 우선순위를 고려한 알림 로직 구현
- 반복된 알림에 사용자가 불편을 느끼지 않도록 6시간 내 동일 알림 전송 방지 기능 구현
- 최대 1만대 제품의 수신조건을 효율적 테스트하기 위한 시뮬레이션 서버 구축
- Front-End팀, 실제제품QA팀과 유기적으로 협업하여 진단로직 개선 및 기능 확장 반영

[성과]

- 초단위 시뮬레이션 기반 진단 로직 검증 환경 구축
 - 기존: 실제 장비 환경에서 진단 로직 검증 시 최소 36시간 소요 -> 테스트 주기 비효율적
또한, 매 5분마다 최대 1만 대 장비 동시 처리 가능 여부를 검증할 환경 부재
 - 개선: 실제 장비와 동일하게 MQTT 송신 가능한 시뮬레이션 JS 스크립트 구현
 - 성과: MQTT 송신 간격을 조절하여 초단위 시뮬레이션 가능 -> 테스트 시간 대폭 단축
1대 장비를 33ms 간격으로 송신해 1만 대 장비의 5분 주기와 동일한 처리량 재현
- 진단 로직 테스트 효율화를 통한 테스트 시간 66% 절감
 - 기존: 모델별 진단 종류·온도 센서 등 입력 변수가 많아 스크립트 구분과 파일 관리 번거로움
유의미한 테스트 케이스 생성 시 과도한 시간·노력 소모
 - 개선: jQuery 기반 UI 테스트 툴 개발 -> 모델, 진단 케이스, 센서값을 시각적으로 입력·관리 가능
 - 성과: 계산기 기능을 통해 실행 전 결과 예측 가능
반복적 스크립트 작성 없이 테스트 수행 가능 -> 복잡도·시간 66% 절감
수동 입력도 지원하여 다양한 테스트 시나리오에 유연하게 대응
- 자가진단 로직 리팩토링: 기능 단위 모듈화
 - 기존: 각 진단 조건마다 동일한 데이터(평균 등)를 반복 계산하여 성능 저하
진단부, 계산부, 알림부의 기능이 명확히 분리되지 않아 가독성이 낮고 유지보수 어려움
불만족 조건 원인 정보를 확인하기 어려워 디버깅 및 테스트 불편
 - 개선: 공통 계산부 분리: 모듈화 구조로 성능 및 유지보수성 향상
기능 단위 분리: 계산/진단/알림 등 기능 단위 함수화
조건 실패 로깅 도입: if~return 구조로 로깅
 - 성과: 중복 연산 제거로 처리 성능 최적화 및 서버 부하 감소: 최대 25회 -> 6회 (74% 감소)
우선순위알림 등 알림 기능 확장, 진단 종류 확장 용이
개발 환경에서 실제 환경과 동일 데이터를 재현하여 로그 기반 실패 조건 확인 가능

파세코 IoT 에어컨 AWS서버 운용 및 유지보수

2024.01 - 2025.06 | 서비스 운영 및 인프라 관리 | 사원

[프로젝트 개요]

시즌별 IoT 에어컨/사용자의 증감에 따른 서버 리소스 사용량 변화에 대응하여 서버의 리소스 할당을 동적으로 조정하고, 인프라 안정성을 확보한 프로젝트입니다.

[팀구성]

Front-End 2명, Back-End 2명(Back-End 개발자로 참여)

[기술스택]

- 백엔드 개발: Node.js (Express.js), MySQL, Mosquitto, Redis
- 개발 및 배포 환경: AWS EC2(Ubuntu)/RDS, Forever
- 기타: Git, Winston(로깅)

[담당 역할]

- 여름철(5~6월) 성수기를 대비해 선제적인 서버 스케일업·스케일아웃 수행
- 겨울철(11~12월) 비수기를 대비해 선제적인 서버 스케일다운·스케일인 수행
- 스케일링 전후 자원 모니터링 및 서비스 안정성 점검
- 프라이빗 대시보드 서버(Ubuntu) 기반으로 제품 커백션 수, Active 사용자 수 추이, CPU 사용량 등 리소스 분석
- 이슈 발생 시 실시간 로그 모니터링 기반 에러 분석 및 보고

[성과]

- 계절별 사용자 수 변동에 맞춘 선제적 리소스 대응으로 트래픽 급증 시에도 100% 무중단 서비스 유지
- 서버 이상 징후를 사전 감지 및 대응함으로써 주요 서비스 장애를 사전에 차단
- AWS클라우드 인프라 서버 운영 및 보안 대응 역량 강화
- 보고 체계 및 협업 커뮤니케이션 능력 향상



주식회사혜성테크윈(HyesungTechwinCo.,Ltd) 

2019.11 - 2022.08 (2년 10개월) | 정규직 | 네트워크 엔지니어 | 대리

광선로 감시 시스템 개발

2019.11 - 2022.08 | 네트워크 엔지니어 | 대리

[주요 역할]

- 사내 개발 OTDR 기반 광선로 감시 시스템의 장비 연동 테스트 및 성능 검증 수행
- 제품 출시 전 현장 환경에서 최종 필드테스트 주도 및 이슈 리포트 관리

[성과]

- OTDR 연동 테스트 및 펌웨어 검증을 통해 제품 초기 통신 오류 3건 조기 발견 및 개선
- 시범 현장에서의 검증을 바탕으로 감시 장비 상용화 의사결정 지원

네트워크 종합관제 소프트웨어 QA

2019.11 - 2022.08 | 네트워크 엔지니어 | 대리

[주요 역할]

- NMS(네트워크 관제 시스템) 기능 테스트 수행을 통해 초기 오류 152건 발견
- 개발자와의 유기적 협업을 통해 전 건 수정 완료(100%)

[성과]

- 고양, 송도 등 5개 지자체 폐쇄망 프로젝트에서 성공적 배포 수행

광 네트워크 구축

2019.11 - 2022.08 | 네트워크 엔지니어 | 대리

[주요 역할]

- L2/L3 스위치 기반 광통신망 구축 및 VLAN 세그먼트 구성

[성과]

- 고양, 송도, 의정부 등 8개 지자체 광통신 구축 프로젝트 성공적 수행

학력



충남대학교

2011.03 - 2018.02 | 졸업 | 정보통신공학과 학사

컴퓨터 네트워크, 데이터베이스, 통신이론, 알고리즘 등 과목 이수

스킬

ExpressJS

MySQL

JavaScript

NodeJS

GitHub

Git

Redis

RabbitMQ

리눅스 서버

Ubuntu

TypeScript

Docker

Slack

Notion

Trello

HTML5

CSS 자바 스크립트

jQuery

AWS

아마존 S3

아마존 EC2

Nginx

VMware Server

수상/자격증/기타



부트캠프 Back-End 과정 수료

2023.01 | 교육

- 백엔드 개발자 교육과정 수료(3개월 과정)
- Node.js, ExpressJS, MySQL, Amazon EC2/RDS/S3를 사용한 SNS 웹사이트, 향수 웹스토어, 중고거래 마켓을 모델링하여 서비스 기획, ERD 설계, API, DB 구축
- GitHub을 이용한 협업



정보처리기사

2016.07 | 자격증

한국산업인력공단, 162021303571



정보통신기사

2017.05 | 자격증

한국방송통신전파진흥원, 17-71-4-0023



CCNA

2020.02 | 자격증

(2023.02 만료) CISCO, Y7FSXGVL0G441J9N



네트워크관리사2급

2021.12 | 자격증

링크

🔗 포트폴리오

https://www.canva.com/design/DAGq1FwoZlc/uHhR8WCnYWGz0mC_XpH62g/view?utm_content=DAGq1FwoZlc&utm_campaign=designshare&utm_medium=link2&utm_source=uniquelinks&utlId=h05bd4e5d79

🔗 [깃허브] 중고거래 플랫폼 브이볼트마켓(vvolt market)

<https://github.com/Gyelanjjim/nestjs-vvolt-market>

🔗 [안드로이드] 재난신고알림서비스 레인버드 아시아(Rainbird - Asia)

<https://play.google.com/store/apps/details?id=com.rainbirdgeo.app>

🔗 [안드로이드] 재난신고알림서비스 레인버드 인도네시아(AwasBencana)

https://play.google.com/store/apps/details?id=com.rainbirdgeo.save_the_children

🔗 [안드로이드] 재난신고알림서비스 레인버드 태평양(Rainbird - Pacific)

<https://play.google.com/store/apps/details?id=com.rainbirdgeo.unep>

🔗 [안드로이드] 재난신고알림서비스 레인버드 아프리카(Rainbird - Africa)

<https://play.google.com/store/apps/details?id=com.rainbirdgeo.af.app>

🔗 [깃허브] 백준/프로그래머스 알고리즘 학습

<https://github.com/Gyelanjjim/baekjoon>

🔗 [기술블로그] 비상벨/TTS/마이크 방송 시스템 연계 회고

<https://velog.io/@scroll0908/%EC%8B%A4%EC%8B%9C%EA%B0%84-%EB%B0%A9%EC%86%A1TTS-%EC%8B%9C%EC%8A%A4%ED%85%9C-%ED%86%B5%ED%95%A9-%EA%B3%BC%EC%A0%95%EC%97%90%EC%84%9C-%EB%B0%B0%EC%9A%B4-%EA%B2%83%EB%93%A4>