강동현 · 백엔드 신입 (Platform / DevOps 친화)

Email	donghyeonka68@gmail.com
GitHub	github.com/donghyeon-ka
Blog	velog.io/@donghyeon2

Summary

Spring Boot 애플리케이션이 실제로 실행되는 환경까지 이해하려고, K3s 기반 로컬 / dev 플랫폼을 직접 구성해 온 신입 백엔드 개발자입니다.

Ingress 인증 게이트, Vault 기반 secret 주입, default-deny NetworkPolicy, PSS Restricted baseline을 함께 구성하며 애플리케이션 코드 밖의 책임을 명확히 나누는 데 집중했습니다. 백엔드는 JWT 검증과 도메인 처리에 집중하고, 로그인 / 세션 / secret / 네트워크 경계는 플랫폼 계층에서 다루도록 설계했습니다.

아래 내용은 운영 경험이 아니라 로컬 / dev 환경에서 실행 구조를 검증한 결과입니다. 대신 각 결정이 어떤 책임을 분리하기 위한 것인지 드러나도록 정리했습니다.

Skills

Core

- Kubernetes (K3s 1.30), Kustomize, Traefik, NetworkPolicy default-deny, PSS Restricted

Secrets / Auth Gateway

- HashiCorp Vault 1.17.2, Vault Secrets Operator, Keycloak, oauth2-proxy, Traefik ForwardAuth, HelmChartConfig

Validation / Backend

- `kustomize build`, `kubeconform -strict`, `kube-linter`, Java 21, Spring Boot 4, Spring Security Resource Server, PostgreSQL, Flyway

Projects

Project-Infra — Kubernetes 기반 백엔드 실행 환경

> 개인 프로젝트 · 2025.11 ~ 현재 · 1 인 · 설계 / 구현 / 로컬 검증 github.com/donghyeon-ka/project-infra

Spring Boot 인증 서버가 동작하는 K3s 환경을 구성하고, 인증 게이트 / secret 주입 / 네트워크 정책 / manifest 검증 책임을 분리한 프로젝트입니다.

- Traefik ForwardAuth + oauth2-proxy + Keycloak 조합으로 미인증 요청을 Ingress 계층에서 먼저 차단하도록 구성했습니다.
- auth-server는 Ingress 뒤에서 Keycloak JWT를 Resource Server로 다시 검증하게 하여, 로그인 세션 처리와 API 권한 검증을 분리했습니다.
- Vault KV-v2 값을 VSO가 Kubernetes Secret으로 동기화하도록 구성하고, 도메인별 Vault role / policy를 나눠 secret 접근 범위를 제한했습니다.
- default-deny NetworkPolicy를 기본값으로 두고, 필요한 컴포넌트 간 통신만 명시적으로 허용했습니다.
- K3s packaged Traefik manifest를 직접 수정하지 않고 `HelmChartConfig` overlay로 설정을 관리해, 재부팅 시 설정이 사라지는 문제를 피했습니다.
- PSS Restricted baseline을 유지하기 위해 Vault의 `disable_mlock` 설정을 선택했고, 그에 따른 swap-off 운영 전제를 명시했습니다.
- 주요 overlay에 대해 `kustomize build`, `kubeconform -strict`, `kube-linter` 검증을 실행하는 스크립트를 구성했습니다.
- MinIO Operator 설치, namespace / PSS 라벨 적용, Vault init / unseal, VSO 설치 흐름을 반복 실행 가능한 bootstrap script로 분리했습니다.

`Kubernetes` `K3s` `Kustomize` `Traefik` `oauth2-proxy` `Keycloak` `Vault` `VSO` `NetworkPolicy` `PSS`

Project-Auth-Server — 플랫폼 뒤에서 동작하는 Spring Boot 백엔드

> 개인 프로젝트 · 2025.11 ~ 현재 · 1 인 · 설계 / 구현 / 로컬 검증 github.com/donghyeon-ka/project-auth-server

플랫폼 계층이 로그인 / 세션을 처리하고, 백엔드는 Resource Server로 JWT를 검증하는 구조의 Spring Boot 인증 서버입니다.

- `oauth2-client` 가 아니라 `oauth2-resource-server` 를 사용해 백엔드 책임을 JWT 검증과 권한 매핑으로 좁혔습니다.
- Clean / Hexagonal 5 개 모듈 구조와 ArchUnit 테스트로 계층 의존성 위반을 빌드 단계에서 탐지하도록 했습니다.
- application 코드가 HTTP status 같은 web 프로토콜 세부사항을 직접 알지 않도록 `ErrorCode` 와 HTTP status 매핑을 분리했습니다.
- K8s 환경에서는 공식 `flyway/flyway` 이미지를 initContainer로 실행해, 자체 migration 이미지 빌드 사이클을 줄였습니다.

`Java 21` `Spring Boot` `Spring Security` `Keycloak` `JPA` `Flyway` `ArchUnit`

feed-postgresql-query-tuning — PostgreSQL 성능 검증

> 개인 프로젝트 · 2025.03 ~ 2025.04 · 1 인 · 분석 / 최적화 / 검증 github.com/donghyeon-ka/feed-postgresql-query-tuning

플랫폼 관점에서도 애플리케이션 병목을 관찰할 수 있도록, DB 쿼리 실행 계획과 부하 테스트를 함께 정리한 프로젝트입니다.

- `EXPLAIN ANALYZE` 기준 피드 목록 조회 쿼리 실행 시간을 **1.77 s → 7.66 ms**로 개선했습니다.
- JOIN 구조 변경, 페이지네이션 쿼리 분리, 복합 / 커버링 인덱스 적용을 단계적으로 검증했습니다.
- 단일 인스턴스 Locust 1,000 동시 사용자 시나리오에서 약 **900~1,100 RPS** 구간을 확인했습니다.

`PostgreSQL` `EXPLAIN ANALYZE` `Index design` `Redis` `Locust`

Links

- **GitHub**: github.com/donghyeon-ka
- **메인 프로젝트**: project-infra
- **인증 서버**: project-auth-server
- **쿼리 튜닝**: feed-postgresql-query-tuning
- **Blog**: velog.io/@donghyeon2