강동현 · 백엔드 신입 (Platform / DevOps 친화)

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

Summary

Spring Boot 애플리케이션이 실제로 실행되는 경로까지 이해하기 위해 K3s, Nginx, Traefik, Keycloak, Vault를 직접 구성해 온 신입 백엔드 개발자입니다.

물리 서버 host에 설정을 하나씩 추가하다 보니 TLS 종료 위치, proxy 신뢰 범위, secret과 session의 저장 위치를 설명하기 어려워졌습니다. 이를 정리하기 위해 **외부 요청이 Nginx를 거쳐 Kubernetes Service까지 도달하는 경로와 각 컴포넌트의 책임을 분리**했습니다.

모든 내용은 개인 home lab / dev 환경에서 검증한 결과이며 운영 경험으로 확대해 표현하지 않았습니다. 현재 hyeonworks.com에서 TechLog frontend를 서비스하고 있으며, backend와 플랫폼의 최종 통합은 진행 중입니다.

Skills

Core

- Kubernetes / K3s, Nginx, Traefik, Kustomize, Helm / HelmChartConfig, NetworkPolicy

Secrets / Identity

- HashiCorp Vault 1.17.2, Vault Secrets Operator, Keycloak, oauth2-proxy, Kubernetes Auth

Data / Observability

- PostgreSQL / CloudNativePG, Redis, Prometheus, Grafana, Loki, Tempo, Blackbox Exporter

Projects

Home Platform — K3s 기반 서비스 실행 환경

> 멘토링 기반 개인 프로젝트 · home lab / dev hyeonworks.com

서비스를 하나씩 올리면서 host와 cluster의 책임이 섞이기 시작했습니다. 단순히 “접속된다”가 아니라 **요청이 어떤 신뢰 경계를 지나 Pod까지 도달하는지 설명할 수 있는 구조**로 다시 구성했습니다.

- 외부 HTTPS를 Host Nginx에서 종료하고 Traefik NodePort를 loopback에만 열어 `Internet → Nginx → 127.0.0.1 NodePort → Traefik → Service` 경로로 진입점을 제한했습니다.
- 외부가 보낸 `X-Forwarded-*`를 Nginx에서 다시 쓰고, Traefik은 실제 proxy source만 신뢰하도록 범위를 줄였습니다.
- 공개 서비스와 관리용 hostname을 분리하고 관리 경로에는 LAN / Tailscale source allowlist를 적용했습니다.
- CloudNativePG / PostgreSQL, Keycloak, workload별 Local PV를 구성하고 서비스별 DB / role / Secret / NetworkPolicy를 분리했습니다.
- Prometheus / Grafana / Loki / Tempo / Blackbox Exporter를 구성해 endpoint, node, log, trace, 인증서 상태를 각각 관측했습니다.
- manifest에 선언된 설정과 live cluster에서 실제 집행되는 상태를 대조해, 선언만 존재하는 항목을 완료로 판단하지 않도록 검증했습니다.

`K3s` `Nginx` `Traefik` `Kustomize` `CloudNativePG` `Prometheus` `Grafana`

Vault Secret Delivery — 애플리케이션 밖으로 secret 책임 분리

> 개인 lab · 현재 Vault는 single replica / file storage 구성

Secret 값을 Git에서 제거하는 것만으로는 누가 Vault에 접근하고, workload가 어떤 credential을 받는지 해결되지 않았습니다.
애플리케이션이 Vault API를 직접 알지 않도록 secret 전달 경계를 분리했습니다.

- Vault KV-v2를 source of truth로 두고 **Vault** → **VSO** → **Kubernetes Secret** → **workload** 흐름으로 구성했습니다.
- `VaultConnection` / `VaultAuth`와 Kubernetes Auth를 사용하고 role / policy를 경로별로 분리했습니다.
- VSO controller → Vault init / unseal → auth / policy → KV seed → destination Secret → workload 순으로 bootstrap dependency를 정리했습니다.
- 실행 중 생성되는 registry credential은 생성 후 Vault에 기록하고 consumer가 VSO를 통해 다시 받도록 흐름을 분리했습니다.
- 현재 구성은 Vault 1 replica / file storage / local-path PVC 기준이며 Raft HA, auto-unseal 운영 경험으로 표현하지 않았습니다.

`Vault 1.17.2` `Vault Secrets Operator` `Kubernetes Auth` `Kustomize`

Keycloak Multi-node — session과 장애의 실제 위치 확인

> 개인 lab · VM 2대 / 2-node K3s

Keycloak replica를 늘리면 session이 자동으로 공유된다고 보기 쉬웠습니다. 이를 확인하기 위해 두 VM에 Keycloak을 나눠 배치하고 DB, cache, network 조건을 바꿔가며 상태의 실제 위치를 확인했습니다.

- Keycloak 26 기본 `persistent-user-sessions`에서 교차 노드 session 처리가 **공유 PostgreSQL session 조회에 의존**하는 것을 SQL로 확인했습니다.
- persistent session을 끈 조건에서는 JGroups 연결, rolling restart, DB 상태가 결과에 미치는 영향이 달라지는 것을 비교했습니다.
- NetworkPolicy, iptables, `tc`를 적용했다고 바로 장애로 판단하지 않고 conntrack / packet counter / cluster state로 **장애 주입 자체가 성공했는지 먼저 검증**했습니다.
- Refresh Token Rotation 동시 경쟁과 replica별 JWKS cache 차이처럼 단일 인스턴스에서는 드러나지 않는 상태 경쟁도 확인했습니다.

`K3s` `Keycloak` `PostgreSQL` `Redis` `JGroups` `Linux networking`

Links

- **TechLog**: hyeonworks.com
- **GitHub**: github.com/donghyeon-ka
- **Blog**: velog.io/@donghyeon2