PROFILE

01

PORTFOLIO

Proxmox 기반 개인 서비스 인프라 구축 및 운영

김종훈 · 신입 서버·시스템 엔지니어

Bosgame E5 Mini PC에 Proxmox VE와 Ubuntu Server 24.04 VM을 구성하고 개인 블로그를 배포했습니다. 서버 구축, 애플리케이션 개발, 배포 및 백업 자동화를 직접 수행했습니다. HTTPS로 블로그를 공개하고 GitHub Actions에서 버전 태그로 배포하도록 구성했습니다.

핵심 역량	프로젝트에서 적용한 내용
가상화·Linux	Proxmox VE에 Ubuntu VM을 생성하고 서비스를 배포했습니다.
컨테이너·네트워크	Nginx로 HTTPS 요청을 처리하고 Next.js·Spring Boot로 전달했습니다.
배포·백업 자동화	버전 검사와 배포 후 점검, DB·업로드 백업 작업을 스크립트로 작성했습니다.

기간: 2026-09-24 ~ 현재 · 구축·개선·운영 중

연락처: kjhreport3469@gmail.com

GitHub · Geniewiz

github.com/Geniewiz

공개 서비스 · shurikim.dev

shurikim.dev/

TECHNICAL SKILLS

02

사용 기술 및 적용 내용

VM 구성, 서비스 간 통신, 데이터 저장, 배포·백업 자동화에 아래 기술을 사용했습니다.

영역	적용 기술과 범위
가상화·OS	Proxmox VE · Ubuntu Server 24.04 / VM 구성, SSH 접속, systemd timer 기반 정기 백업
컨테이너·진입점	Docker Compose · Nginx · Certbot / 서비스 연결, HTTPS, 경로별 프록시
데이터	PostgreSQL 17 · Redis 8 / PostgreSQL 영속 데이터 저장·Redis 임시 상태 관리
자동화	GitHub Actions · Bash / 릴리스 검사, VM 배포, 백업 작업
관리 접속	Tailscale · OpenSSH / Tailscale 기반 SSH 원격 접속·배포

Next.js와 Spring Boot를 컨테이너로 배포했습니다. PostgreSQL·Redis 연결, 데이터 저장소, 서비스 시작 순서와 배포 순서를 설정했습니다.

단일 VM에서 여러 서비스를 운영하기 위해 Docker Compose를 선택했습니다. 서비스별 실행 설정과 의존 관계를 Compose 파일로 관리했습니다.

컨테이너 실행 상태

서비스	상태 · 호스트 포트 게시
app	실행 중 · healthy · 게시 없음
frontend	실행 중 · 게시 없음
Nginx	실행 중 · 80:80, 443:443
PostgreSQL	실행 중 · healthy · 게시 없음
Redis	실행 중 · healthy · 게시 없음

2026-10-29 기준 컨테이너 실행 상태 및 호스트 포트 게시 현황

INFRASTRUCTURE ARCHITECTURE

03

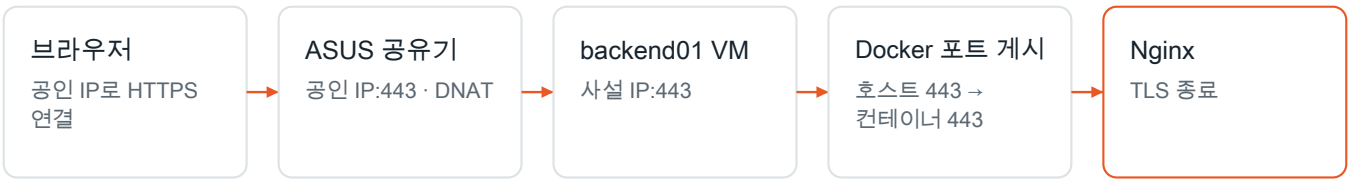
공개 요청과 관리 접속의 흐름

브라우저는 DNS에서 조회한 공인 IP로 HTTPS 연결을 시작합니다. 요청은 공유기의 포트 전달을 거쳐 backend01 VM의 Nginx에 도착합니다.

① DNS 이름 해석

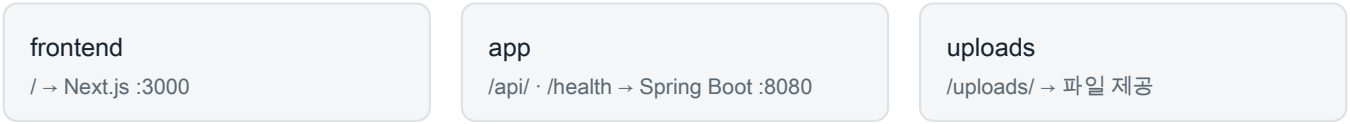


② HTTPS 요청 전달



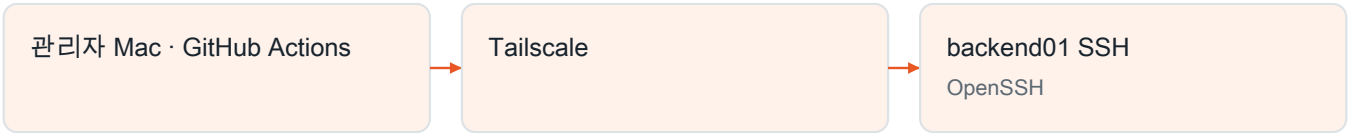
Proxmox VE 9.2.2 · nic1 ↔ vmbr0 ↔ backend01 VirtIO NIC (192.168.50.68)

Nginx의 요청 분기 · VM 내부



frontend → app:8080 · app → PostgreSQL:5432 / Redis:6379 / uploads

관리 접속 · SSH



DNS와 포트 전달 설정

Cloudflare는 DNS 조회에만 사용합니다. 공유기는 HTTP(80)와 HTTPS(443) 요청을 backend01 VM으로 전달하고, Nginx는 HTTP 요청을 HTTPS로 전환합니다. 내부 IP를 고정하는 DHCP/IP 바인딩과 외부 요청을 전달하는 포트포워딩은 별도로 설정했습니다. 컨테이너끼리는 Compose 기본 네트워크로 통신합니다.

IMPLEMENTATION · SERVER

04

Proxmox 기반 Ubuntu VM 구성

Proxmox VE 9.2.2에서 Ubuntu Server 24.04 VM 한 대로 개인 서비스를 운영합니다. VM은 물리 NIC nic1에 연결된 Linux 브리지 vmbr0를 사용합니다.

구분	사용 환경
물리 장비	Bosgame E5 · Ryzen 5300U · DDR4 16GB
VM 할당	4 vCPU · 메모리 최대 4GiB · 디스크 64GiB
네트워크	VirtIO → vmbr0 → nic1 · 게이트웨이 192.168.50.1

backend01 하드웨어 설정 · 주요 항목 발췌

Memory	2.00 GiB/4.00 GiB
Processors	4 (1 sockets, 4 cores) [x86-64-v2-AES]
Hard Disk (scsi0)	local-lvm:vm-100-disk-0,discard=on,iothread=1,size=64G,ssd=1
Network Device (net0)	virtio=BC:24:11:09:83:20,bridge=vmbr0,firewall=1

vmbr0의 물리 NIC·호스트 주소 · 해당 행 발췌

Ports/Slaves	Bond Mode	CIDR	Gateway
nic1		192.168.50.252/24	192.168.50.1

호스트 주소는 192.168.50.252/24, VM 주소는 192.168.50.68입니다. 관리자 Mac은 Tailscale을 거쳐 VM에 SSH로 접속합니다. 내부 테스트용 Nginx는 tailnet IP의 8080 포트에서 요청을 받도록 설정했습니다.

IMPLEMENTATION · CONTAINERS

05

실행 단위와 데이터 보관의 분리

애플리케이션 컨테이너와 영속 데이터 저장소를 분리했습니다. 컨테이너의 역할에 따라 볼륨의 읽기·쓰기 권한을 구분했습니다.

대상	구성과 선택 이유
frontend · app	Next.js와 Spring Boot 실행 단위입니다. 배포 시 새 버전으로 교체합니다.
PostgreSQL	콘텐츠·계정·세션·이력·미디어 메타데이터를 postgres-data에 보관합니다.
uploads	app이 원본을 쓰고 Nginx는 같은 볼륨을 읽기 전용으로 연결합니다.
Redis	OAuth 상태·티켓, 요청 제한, CSV 작업 상태를 다루며 redis-data 볼륨을 연결합니다.
인증서·webroot	Nginx에는 읽기 전용, Certbot 작업에는 쓰기를 허용합니다.

다섯 서비스에 restart: unless-stopped를 설정했습니다. Compose로 처음 시작할 때 DB와 Redis의 상태 검사를 통과하면 app을 시작하고, app의 상태 검사를 통과하면 frontend를 시작합니다.

게시글·회원·미디어 정보는 PostgreSQL에 저장했습니다. JPA로 데이터에 접근하고 Flyway로 DB 스키마 변경을 관리했습니다. Redis에는 인증 상태와 일회성 티켓을 저장하고, TTL로 만료시키며 티켓은 한 번만 사용할 수 있도록 했습니다.

컨테이너 교체 후 데이터 보관

데이터를 볼륨에 저장해 컨테이너를 교체한 뒤에도 유지되도록 했습니다. 볼륨도 같은 호스트의 디스크를 사용하므로 디스크 손상에 대비할 백업은 별도로 구성했습니다.

SECURITY & DEPLOYMENT

06

HTTPS 구성 및 자동 배포 절차

Certbot으로 인증서를 발급받고 Nginx에서 HTTPS 연결을 처리했습니다. GitHub Actions는 버전 태그와 CI 결과를 검사한 뒤 Tailscale·SSH로 VM에 접속해 배포합니다.

1. 대상 검증과 접속
   태그가 가리키는 커밋이 main에 포함되어 있는지, CI 검사를 통과했는지 확인한 뒤 Tailscale·OpenSSH로 VM에 접속합니다.
2. 교체 전 검사와 빌드
   docker compose config --quiet로 Compose 구성을 검사하고 app·frontend를 빌드합니다.
3. 애플리케이션 교체
   app·frontend를 교체한 뒤 app 상태와 frontend HTTP 응답을 확인합니다.
4. Nginx 교체와 사후 검사
   Nginx를 강제 재생성한 뒤 실행 중인 컨테이너에서 nginx -t를 수행하고 내부·공개 HTTPS와 페이지를 검사합니다.

HTTP 요청은 HTTPS로 전환하고 TLS 1.2·1.3을 허용합니다. 외부에서 /actuator/ 경로로 들어온 요청에는 404를 반환하고 /health는 별도로 제공합니다. 접근 로그에는 쿼리와 Referer가 남지 않도록 설정했습니다.

배포 방식의 한계

단일 VM에서 컨테이너를 순차적으로 교체하므로 일시적인 접속 중단이 발생할 수 있습니다. Compose 설정 검사나 이미지 빌드가 실패하면 기존 컨테이너를 유지합니다. 교체 이후 오류에 대한 자동 롤백은 구현하지 않았습니다.

개선 계획

컨테이너 교체 전에 Nginx 설정을 검사하고, 배포 실패 단계별 복구 절차를 문서화할 계획입니다.

BUILD VALIDATION

07

CI 검증 및 운영 배포 결과

릴리스 조건, Backend·Frontend, Browser E2E, 운영 스크립트를 검사한 뒤 운영 서버에 배포했습니다.

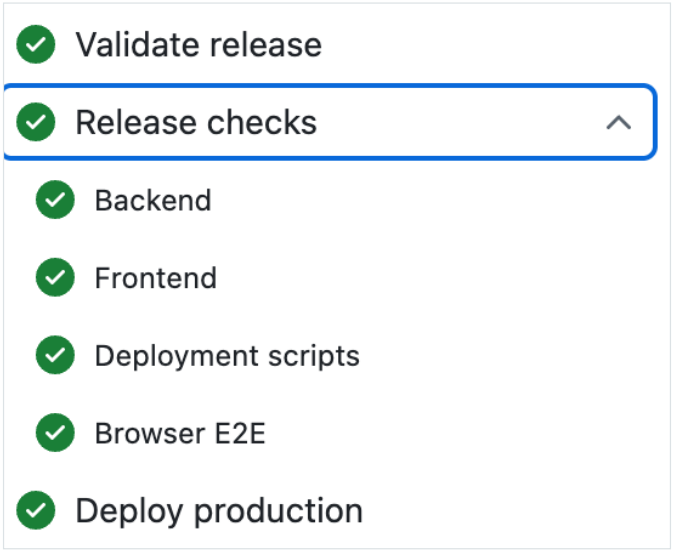
E2E 테스트는 운영 환경과 분리해 Next.js·Spring Boot·PostgreSQL·Redis를 함께 실행했습니다. 외부 소셜 로그인에는 테스트용 구현을 사용했습니다. 배포·운영 스크립트는 외부 명령을 모의 실행해 오류 처리 절차를 검사했습니다.

GitHub Actions 배포 실행

Geniewiz pushed	d8746d3 v0.4.0	Success	7m 41s	–

2026-10-28(KST) 실행 · v0.4.0 · 커밋 d8746d3 · 전체 실행 7분 41초

릴리스 검사와 운영 배포



릴리스 검증, Backend, Frontend, Deployment scripts, Browser E2E, 운영 배포가 모두 통과했습니다.

BACKUP AUTOMATION

08

DB와 업로드 파일의 백업 자동화

PostgreSQL 데이터와 업로드 파일을 함께 백업하고 검사한 뒤 S3에 저장하도록 자동화했습니다.

1. 백업 생성
   app을 중지하고 PostgreSQL custom 형식 덤프와 uploads 아카이브를 만듭니다.
2. 서비스 재시작
   app을 다시 시작하고 상태를 확인합니다. 백업이 실패하거나 중단돼도 재시작을 시도합니다.
3. 백업 무결성 및 복원 검증
   체크섬을 검사하고 격리 DB에 복원합니다. uploads 압축 해제까지 통과한 백업을 최종 디렉터리로 이동합니다.
4. 보관과 외부 저장
   S3 업로드·다운로드 비교가 통과하면 보관 기간이 지난 로컬 백업을 정리합니다.

매일 03:30 KST에 systemd timer로 백업합니다. 로컬 백업은 14일 보관하고, S3 현재 버전은 생성 후 30일에 만료되도록 설정했습니다. 실패 알림의 수신 테스트를 완료했으며, 실제 백업 장애로 알림이 발생한 사례는 없습니다.

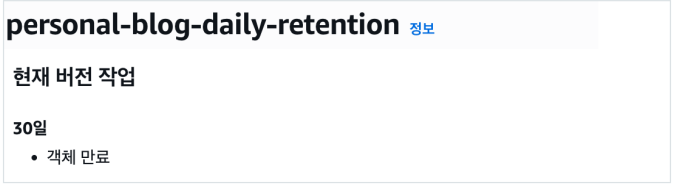
검사에 실패한 백업은 완료 처리하지 않고 이전 백업도 정리하지 않습니다. 백업 중 app을 멈추는 동안 서비스 이용에 영향이 생길 수 있습니다.

격리 DB 복원 및 업로드 검사

```
Restore verification passed: 21 tables; uploads extracted successfully.
```

2026-10-29 03:30 KST · 테이블 21개 복원, uploads 압축 해제 통과

S3 Lifecycle 보관 설정



personal-blog/daily/ 대상 규칙 · 현재 버전 생성 후 30일 만료

LESSONS LEARNED

09

구축 과정에서 확인한 점과 개선 방향

서비스별로 컨테이너를 나눴지만 모두 같은 VM과 물리 서버에서 실행됩니다. VM이나 물리 서버에 문제가 생기면 서비스 전체가 영향을 받을 수 있습니다.

격리 DB 복원 검사는 통과했습니다. 실제 서비스의 복구 여부를 확인하려면 애플리케이션을 다시 실행하고 접속까지 검사해야 합니다. 이 복구 실습은 아직 하지 않았습니다.

다음 작업

배포 전 Nginx 설정 검사를 추가하고, 장애를 재현해 로그로 원인을 분석할 계획입니다. 별도 환경에서 서비스 복구와 부하 테스트를 진행하겠습니다.

GitHub · Geniewiz

github.com/Geniewiz

공개 서비스 · shurikim.dev

shurikim.dev/

프로젝트 소스코드는 비공개입니다. 주요 설정과 실행 화면은 이 문서에 담았습니다.