[요구사항정의서]

VMware & K8s 기반 배관 수리 예약 시스템

1. 프로젝트 주제

- **주 주제:** Nginx 기반의 로드밸런싱 아키텍처 구축
- **세부 주제:** 배관 서비스 플랫폼을 위한 3-Tier 고가용성(HA) 시스템 설계 (VMware 가상화 환경)
- **배경:** 삼촌 사업체의 웹 서비스 구축을 목적으로, VMware 가상화 기술을 활용하여 실무 수준의 고가용성(HA) 인프라를 로컬 서버 내에 구현하고자 합니다.

2. 요구사항 정의

2.1 사용자 요구사항

- **일반 고객:** 비회원 견적 신청 및 예약 현황 조회 (성함/연락처/예약번호 기반).
- **관리자:** 예약 승인, 전체 일정 캘린더 관리, 고객 연락처 기반 DB 관리. 운영 관제

2.2 시스템 및 네트워크 요구사항 (VMware)

- **네트워크 구성:** VMware Virtual Network Editor를 활용한 논리적 망 분리.
  - **외부망 (NAT/Bridged):** 사용자 접속용 VIP(Virtual IP) 및 외부 패키지 업데이트용.
  - **내부망 (Custom VMNet):** WAS ↔ DB 통신용 전용 대역 (보안 격리).
- **고정 IP 관리:** 모든 서버에 고정 IP를 할당하여 서비스 간 통신 안정성 확보.
- **보안:** OS 레벨 방화벽(UFW) 필수 적용 및 SSH Key 기반 인증.

2.3 운영 및 관제 요구사항 (Observability)

- **모니터링:** Prometheus(지표), Grafana(시각화), Loki(로그) 통합 관제.
- **알림:** 서비스 중단 또는 임계치 초과 시 Slack 등 알림 연동.

3. 프로젝트 목표

- 무중단 서비스 가용성(HA) 확보: Nginx와 Keepalived를 활용한 로드밸런싱 및 VIP 절체 시스템을 구축하여 단일 장애점(SPOF)을 제거하고 무중단 서비스를 지향함.
- 데이터 보호 및 복구 복원력: MySQL Master-Slave 복제 구성을 통해 데이터 무결성을 확보하고, 재해 복구 능력을 증명함.
- 통합 관제 가시성 구현: 인프라 자원 및 애플리케이션 로그를 실시간 시각화하여 장애 발생 전 선제적 대응 체계를 마련함.
- 가상화 기반 보안 설계: 가상 네트워크 분리를 통해 DB 등 핵심 자원의 외부 노출을 최소화하고, 전 구간 암호화 통신을 실현함.
- 자원 효율성 최적화: 한정된 물리 자원 내에서 VM별 최적 스펙 할당 및 컨테이너 기술을 활용해 효율적인 인프라 운영을 달성함.

4. 상세 기술 스펙 (VMware 인프라)

4.1 가상 머신(VM) 사양 및 IP 계획

구분	VM 호스트명	OS	vCPU / RAM	네트워크 (IP)	주요 역할
Web 1	`lb-master`	Ubuntu 22.04	1 / 1GB	NAT (192.168.10.11)	Nginx (Active), Keepalived
Web 2	`lb-slave`	Ubuntu 22.04	1 / 1GB	NAT (192.168.10.12)	Nginx (Standby), Keepalived
App 1	`app-server-01`	Ubuntu 22.04	2 / 2GB	Internal (10.0.0.21)	Node.js API, Docker
App 2	`app-server-02`	Ubuntu 22.04	2 / 2GB	Internal (10.0.0.22)	Node.js API, Docker
DB 1	`db-master`	Ubuntu 22.04	2 / 2GB	Internal (10.0.0.31)	MySQL Master (Write/Read)
DB 2	`db-slave`	Ubuntu 22.04	2 / 2GB	Internal (10.0.0.32)	MySQL Slave (Read Only)
VIP	**Virtual IP**	-	-	**192.168.10.100**	서비스 대표 IP

5. 장애 대응 및 Failover 시나리오

5.1 로드밸런서 장애 (L4/L7 Failover)

- **상황:** `lb-master` 서버 다운.
- **대응:** Keepalived가 수 초 이내에 VIP(192.168.10.100)를 `lb-slave`로 자동 승계.
- **결과:** 사용자는 끊김 없이 서비스 이용 가능.

5.2 애플리케이션 장애 (App Failover)

- **상황:** `app-server-01` 프로세스 종료 또는 서버 다운.
- **대응:** Nginx의 Upstream Health Check 기능이 비정상 서버를 인지하고 `app-server-02`로만 트래픽 전달.
- **결과:** 에러 페이지 없이 정상 응답 유지.

5.3 데이터베이스 장애 (DB Failover)

- **상황:** `db-master` 장애 발생.
- **대응:** `db-slave`를 Read/Write 모드로 변경(Promotion)하고 WAS의 DB 접속 정보를 업데이트. (수동 또는 스크립트 대응)
- **결과:** 데이터 유실 없이 서비스 재개.

6. 프로젝트 산출물 및 구현 순서

6.1 주요 산출물

- `nginx.conf` / `keepalived.conf`: 로드밸런싱 및 VIP 설정
- `docker-compose.yml`: App 및 모니터링 스택 컨테이너 정의서
- `backup_script.sh`: MySQL 자동 백업 및 무결성 검증 스크립트

6.2 구현 순서

1. **VM 생성:** 6대의 가상 머신에 고정 IP 및 기본 보안(SSH, UFW) 설정.
2. **Network 격리:** VMware VMNet 설정을 통해 내부 전용 통신망 구축.
3. **DB 복제:** MySQL Master-Slave 동기화 및 백업 스케줄링.
4. **App 배포:** WAS 컨테이너화 및 내부 통신 테스트.
5. **HA 구성:** Nginx 및 Keepalived를 통한 이중화 및 헬스 체크 설정.
6. **관제 구축:** Grafana 대시보드 구성 및 부하 테스트 수행.