`ARCHITECTURE DESIGN SPECIFICATION (SAD) · ISO/IEC/IEEE 42010 & 4+1 VIEW STANDARD`

[아키텍처 설계 명세서] Spring Boot · Embedded Tomcat · Jenkins · Nexus · Git 표준 배포 아키텍처 및 운영 가이드 (SAD)

문서 관리 항목	명세 내용	문서 관리 항목	명세 내용
문서 번호 (Doc ID)	`SAD-DEPLOY-2026-001`	**버전 (Version)**	`v1.0.0`
대상 시스템 / 범위	전사 Spring Boot 백엔드 표준 CI/CD 및 배포 체계	**문서 상태 (Status)**	`APPROVED (표준 배포 가이드 확정)`
설계 책임 (Architect)	플랫폼 엔지니어링 & DevOps 코어 그룹	**준거 표준 (Standard)**	`ISO/IEC/IEEE 42010 · 4+1 View · SLSA Level 3`
런타임 기준 스택	`Spring Boot 3.3 (Embedded Tomcat 10.1) + Jenkins + Nexus 3 + Git`	**보안 등급**	`대외비 (Internal Engineering Standard)`

문서 제·개정 이력 (Revision History)

버전	일자	변경 구분	작성 / 승인자	주요 설계 제·개정 내역
`v0.5.0`	2026-10-14	초안 작성	DevOps 아키텍처 TF	Git · Jenkins · Nexus 3 · Embedded Tomcat 기반 표준 CI/CD 파이프라인 및 디렉토리 규격 정의
`v0.9.0`	2026-10-21	검토 반영	SRE & 미들웨어 파트	Nginx Upstream Draining, Graceful Shutdown(30s), Atomic Symlink 무중단 롤백 스크립트 반영
`v1.0.0`	2026-10-26	표준 승인	아키텍처 심의 위원회	4대 아키텍처 뷰(논리·프로세스·데이터·배포) 및 배포 적합성 자동 검증 규칙 최종 승인

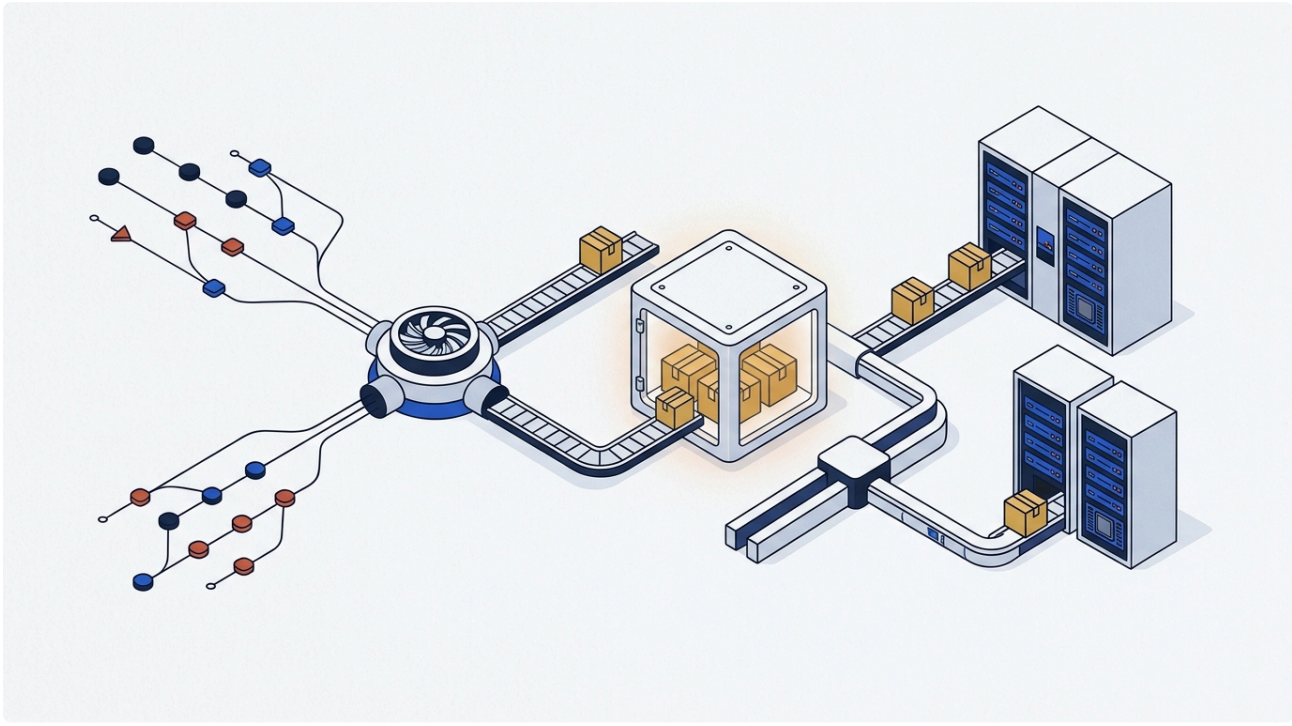
제1장. 설계 개요 및 아키텍처 원칙 (1. Architecture Overview & Principles)

> **[설계 핵심 정의 (Architecture Blueprint Summary)]**: 본 설계서 및 배포 가이드는 **Git(형상 관리) → Jenkins(CI/CD 파이프라인 오케스트레이터) → Sonatype Nexus 3(불변 아티팩트 저장소) → Spring Boot Embedded Tomcat(타겟 WAS 런타임)**으로 이어지는 전사 표준 소프트웨어 공급망(Software Supply Chain)과 무중단 롤링 배포(Zero-Downtime Rolling Deployment) 아키텍처를 명세함.

1.1 시스템 컨텍스트 및 불변 아티팩트(Immutable Artifact) 배포 개념

- **시스템 경계 및 빌드·런타임 분리 구조:**

- 모든 애플리케이션 소스코드는 **Git**에서 단일 진실 공급원(SSOT)으로 관리되며, **Jenkins** 빌드 클러스터에서 컴파일·단위 테스트·정적 보안 분석(SonarQube)을 통과한 뒤 **실행 가능한 단일 Fat JAR(Embedded Tomcat 내장)** 형태로 패키징됨.
- 생성된 바이너리는 **Sonatype Nexus 3** 저장소에 버전별 불변(Immutable) 아티팩트로 영구 봉인되며, 개발·검증·운영 타겟 서버는 재빌드 없이 오직 Nexus에서 검증된 동일 아티팩트를 다운로드하여 `systemd` 및 심볼릭 링크(`Symlink`) 스왑 방식으로 기동함.



Git · Jenkins · Nexus · Spring Boot Embedded Tomcat 기반 표준 CI/CD 공급망 및 배포 개념도

1.2 배포 아키텍처 핵심 설계 원칙 명세 (Architecture Principles Specification)

원칙 ID	아키텍처 설계 원칙명	상세 설계 규격 및 강제 사항 (Design Rule Specification)	검증 방법 (Fitness Check)
`AP-01`	**단일 빌드 및 불변 아티팩트 승격 (Build Once, Deploy Anywhere)**	• 환경별(Dev/Stg/Prod) 재빌드를 전면 금지하며, **단 1회 빌드된 Fat JAR 아티팩트**를 Nexus `maven-releases`에 등록 후 전 환경에 동일 바이너리를 배포함 • 타겟 WAS 서버 내 `git clone` 또는 `gradle/maven` 컴파일 도구 설치를 엄격히 금지함	`DEPLOY-RULE-01` / SHA-256 대조
`AP-02`	**설정 외부화 및 아티팩트 환경 독립성 (Externalized Config)**	• JAR 패키지 내부에는 환경 종속 정보(DB 접속 URL, 패스워드, API Key) 포함을 금지함 • 모든 런타임 환경 설정은 타겟 서버의 `/app/{service}/config/application-{profile}.yml` 및 암호화 환경변수로 외부에서 주입함	`DEPLOY-RULE-02` / Spring Profile
`AP-03`	**무중단 롤링 배포 및 우아한 종료 (Zero-Downtime & Graceful Shutdown)**	• 배포 시 앞단 L7(Nginx/ALB)에서 대상 노드의 트래픽 유입을 선 차단(Draining)한 후 배포를 수행함 • Spring Boot `server.shutdown=graceful` 설정을 통해 진행 중인 인플라이트(In-flight) 요청을 최대 30초간 완결 처리한 뒤 프로세스를 종료함	`DEPLOY-RULE-03` / 부하 중 배포 테스트
`AP-04`	**심볼릭 링크 기반 원자적 스왑 및 즉시 롤백 (Atomic Symlink Rollback)**	• 실행 경로(`/app/{service}/current/app.jar`)는 버전별 보관 디렉토리(`releases/app-{ver}.jar`)를 가리키는 **심볼릭 링크(`ln -sfn`)**로 구성함 • 배포 후 헬스체크 실패 시 재빌드 없이 이전 버전 JAR로 심볼릭 링크를 즉시 환원하여 **15초 이내 롤백**을 보장함	`deploy.sh --rollback` 검증
`AP-05`	**최소 권한 데몬 격리 및 자동 복구 (Systemd Privilege Isolation)**	• Embedded Tomcat 프로세스는 `root` 계정 실행을 금지하며, 로그인 쉘이 차단된 전용 계정(`appuser:appgroup` , `/sbin/nologin`) 및 `systemd` 유닛으로 기동함 • 프로세스 비정상 종료 시 `Restart=on-failure` 정책으로 자동 복구함	`DEPLOY-RULE-04` / Linux CIS Audit

1.3 비기능 품질 속성 및 배포 SLA/SLO 명세 (Quality Attribute Specification)

요구사항 ID	품질 속성 분류	정량 목표 지표 (Target SLA / SLO)	아키텍처 실현 메커니즘 (Architectural Tactics)
`NFR-01`	**배포 무중단성 (Zero-Downtime)**	• 주간 운영 배포 시 **HTTP 502/503 에러 발생률 0.00%** • 인플라이트 트랜잭션 강제 단절 **0건**	• Nginx Upstream Draining(3초 대기) + Spring Boot `Graceful Shutdown(30s)` • `/actuator/health/readiness` HTTP 200 확인 후 로드밸런서 재투입
`NFR-02`	**배포 리드타임 (Lead Time)**	• 코드 푸시(Merge)부터 Nexus 등록까지 **CI 소요시간 ≤ 4분** • 타겟 WAS 그룹 전체 롤링 배포 완료 **CD 소요시간 ≤ 4분**	• Jenkins Agent 내 Nexus 로컬 캐시 미러링 및 Gradle Build Cache 활성화 • Group A → Group B 순차 롤링 스크립트 및 Actuator 워밍업 폴링 자동화
`NFR-03`	**롤백 복구 시간 (Rollback RTO)**	• 배포 실패 감지 시 **자동 롤백 완료 ≤ 15초** • 수동 특정 버전 지정 롤백 완료 **≤ 30초**	• 타겟 서버 로컬 `releases/` 디렉토리에 직전 5개 버전 골든 JAR 상시 보존 • `ln -sfn` 심볼릭 링크 포인터 스왑 + `systemctl restart` 즉시 실행
`NFR-04`	**WAS 런타임 처리량 (Tomcat Performance)**	• 노드당(4 vCPU, 8GB RAM) **≥ 1,500 TPS**, **P99 ≤ 100ms** • Full GC에 의한 Stop-The-World 지연 **≤ 10ms**	• Embedded Tomcat `max-threads=200`, `max-connections=8192`, `accept-count=100` 튜닝 • Java 21 LTS `G1GC` / `ZGC` 및 `Xms=Xmx=4g` 힙 메모리 고정 할당
`NFR-05`	**공급망 무결성 (Supply Chain Security)**	• 미검증 외부 라이브러리 직접 반입 **0건** • 빌드-배포 간 바이너리 위변조 발생률 **0.00%**	• 외부 인터넷 차단 및 Nexus `maven-central-proxy` 단일 경로 의존성 반입 • Nexus 업로드 시 생성된 `SHA-256` 체크섬과 타겟 서버 다운로드 파일 간 해시 대조 강제

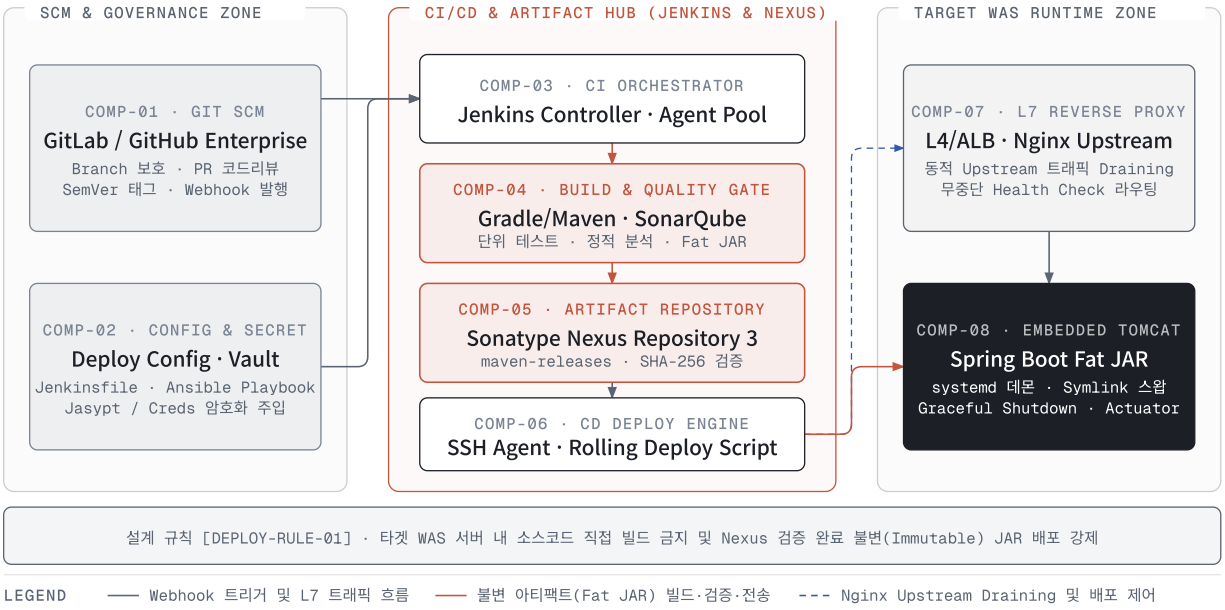
제2장. 논리 아키텍처 및 컴포넌트 설계 (2. Logical & Component View)

2.1 Git · Jenkins · Nexus · Embedded Tomcat 논리 아키텍처 구조도

- **구조 명세**: 전체 배포 시스템은 역할과 네트워크 보안 경계에 따라 **`SCM & Governance Zone (형상/설정 관리)`** → **`CI/CD & Artifact Hub (빌드·품질검증·아티팩트 금고)`** → **`Target WAS Runtime Zone (리버스 프록시 및 임베디드 톰캣 런타임)`**의 3대 영역·8개 표준 컴포넌트(`COMP-01` ~ `COMP-08`)로 구성됨.

VIEW 01 · LOGICAL & COMPONENT ARCHITECTURE (SCM · CI/CD HUB · EMBEDDED TOMCAT RUNTIME)

[논리 뷰] Git · Jenkins · Nexus · Embedded Tomcat 컴포넌트 구조도



View 1 논리 뷰 — Git · Jenkins · Nexus · Embedded Tomcat 배포 아키텍처 컴포넌트 구조도

2.2 컴포넌트 카탈로그 및 책임 명세표 (Component Specification Catalog)

컴포넌트 ID	컴포넌트명	소속 영역 / 노드	핵심 설계 책임 (Core Responsibilities)	입출력 인터페이스 및 보안·의존성 제약
`COMP-01`	**Git SCM Repository**	`SCM Zone` (GitLab / GitHub Enterprise)	• 애플리케이션 소스코드, `Jenkinsfile`, DB 마이그레이션 SQL 버전 관리 • GitFlow/Trunk 브랜치 보호 규칙 적용 및 Merge/Tag 이벤트 시 Webhook 발행	• 출력: Jenkins Webhook (`POST /generic-webhook-trigger/invoke`) • 제약: `main` / `release/*` 브랜치 직접 Push 금지 (최소 2인 PR 승인 필수)
`COMP-02`	**Config & Secret Vault**	`SCM & Secret Zone` (Git Config / Credentials)	• 환경별 외부 설정 파일 (`application-{profile}.yml`) 및 배포 스크립트 관리 • DB 계정, JWT 서명키, Jasypt 마스터 패스워드의 암호화 보관 및 런타임 주입	• 연동: Jenkins Credentials Binding (`withCredentials`) • 제약: 평문 비밀번호·개인키의 Git 커밋 원천 차단 (`gitleaks` 사전 검증)
`COMP-03`	**Jenkins CI/CD Controller**	`CI/CD Hub` (Jenkins LTS Master/Agent)	• 선언적 파이프라인 (`Jenkinsfile`) 파싱, 빌드 스케줄링 및 스테이지 상태 관리 • 운영 환경(Prod) 배포 시 승인자 수동 승인(`input` Gate) 및 배포 감사 로그 기록	• 구조: Controller는 스케줄링만 담당하며 실제 빌드는 **격리된 Build Agent 노드**에서 수행 • 인증: SSO/RBAC 기반 배포 권한 분리
`COMP-04`	**Build & Quality Gate**	`CI/CD Hub` (Gradle 8 / Maven 3.9 + SonarQube)	• Java 21 컴파일, JUnit 5 단위/통합 테스트 실행 및 Spring Boot Executable Fat JAR 패키징 • SonarQube 정적 코드 분석 및 OWASP 의존성 취약점 (CVE) 스캔 수행	• 의존성 조회: 오직 `COMP-05 (Nexus maven-public)` 경유 • 차단 기준: 테스트 실패, 커버리지 `< 80%` , Critical 취약점 발견 시 즉시 빌드 중단
`COMP-05`	**Sonatype Nexus 3 Vault**	`CI/CD Hub` (Nexus Repository Manager 3)	• 외부 오픈소스 캐싱 (`maven-central-proxy`) 및 사내 빌드 산출물 (`snapshots` / `releases`) 영구 저장 • 배포 대상 Fat JAR의 버전 관리, 메타데이터(`maven-metadata.xml`) 및 `SHA-256` 해시 제공	• 정책: `maven-releases` 저장소는 **Write-Once (`Allow redeploy: Disable`)**로 설정하여 기존 배포 버전 덮어쓰기 원천 차단
`COMP-06`	**CD Rolling Deploy Engine**	`CI/CD Hub` (Ansible / SSH Deploy Runner)	• 타겟 WAS 서버군을 Group A / Group B로 분할하여 순차적 무중단 롤링 배포 오케스	• 프로토콜: 배포 전용 Bastion 채널 `SSH (TCP 22, Ed25519 Key 인증)`

컴포넌트 ID	컴포넌트명	소속 영역 / 노드	핵심 설계 책임 (Core Responsibilities)	입출력 인터페이스 및 보안·의존성 제약
			트레이션 수행 • Nexus 아티팩트 다운로드 명령 하달, Nginx 트래픽 제어, 헬스체크 폴링 및 실패 시 자동 롤백 트리거	• 타임아웃: 노드당 헬스체크 최대 60초 대기, 미통과 시 즉시 배포 중단 및 롤백
`COMP-07`	**L7 Reverse Proxy (Nginx/ALB)**	`Target Runtime Zone` (DMZ / Web Tier)	• 외부 클라이언트 HTTPS 종단, 정적 리소스 처리 및 뒷단 Embedded Tomcat 업스트림 로드밸런싱 • 배포 대상 노드의 트래픽 유입을 무중단으로 차단(`down` / Draining)하고 복귀(`up`) 시키는 스위치 역할 수행	• 설정: `proxy_next_upstream error timeout http_502 http_503` • 연동: 배포 스크립트에 의한 업스트림 설정 전환 및 `nginx -s reload` (무중단 설정 반영)
`COMP-08`	**Spring Boot Embedded Tomcat**	`Target Runtime Zone` (Private WAS Tier)	• 외장 WAS 설치 없이 `spring-boot-starter-web` 에 내장된 **Apache Tomcat 10.1** 엔진으로 서블릿 컨테이너 구동 • `systemd` 서비스 유닛으로 관리되며, `SIGTERM` 수신 시 Graceful Shutdown 및 Actuator 헬스 엔드포인트 노출	• 포트: 애플리케이션 `8080/tcp` , 관리/Actuator `8081/tcp` (내부망 전용 분리) • 실행 계정: `appuser` (Non-root 전용 계정)

2.3 Git 리포지토리 및 타겟 WAS 서버 표준 디렉토리 구조 명세

```TEXT
[1] Git 소스 리포지토리 표준 구조 (Source Repository Layout)
{service-name}-backend/
├── Jenkinsfile                          # [COMP-03] 선언적 CI/CD 파이프라인 정의서 (Stage 1~8)
├── build.gradle.kts                     # [COMP-04] Spring Boot 3.3 + Embedded Tomcat + Nexus
├── gradle.properties                    # 그룹 ID, 아티팩트 ID, 기본 버전(SemVer) 명세
├── deploy/
│   ├── scripts/
│   │   └── deploy-rolling.sh            # [COMP-06] Nexus 다운로드 · Symlink 스왑 · Nginx Drainin
│   ├── systemd/
│   │   └── spring-app.service           # [COMP-08] 리눅스 systemd 데몬 유닛 템플릿
│   └── nginx/
│       └── upstream-app.conf            # [COMP-07] Nginx Upstream 로드밸런싱 및 Draining 설정 템플
└── src/
    ├── main/java/...                    # 비즈니스 애플리케이션 소스코드
    └── main/resources/
        └── application.yml              # 공통 기본 설정 + Embedded Tomcat 튜닝 + Graceful Shutdow

[2] 타겟 리눅스 WAS 서버 표준 디렉토리 구조 (Target Server Filesystem: /app/{service})
/app/order-service/
├── current/
│   └── app.jar -> /app/order-service/releases/order-service-1.4.2.jar   # [원자적 Symlink 포인트
├── releases/                            # [Nexus에서 다운로드한 불변 Fat JAR 보관소 — 최근 5개 버전 유지]
│   ├── order-service-1.4.2.jar          # 현재 운영 버전 (+ order-service-1.4.2.jar.sha256)
│   ├── order-service-1.4.1.jar          # 직전 롤백 대기 버전 (Golden Backup)
│   └── order-service-1.4.0.jar
├── config/                              # [패키지 외부 주입 설정 — 권한 chmod 600 appuser:appgroup]
│   ├── application-prod.yml             # 운영 환경 전용 설정 오버라이드 파일
│   └── env.conf                         # systemd EnvironmentFile (JAVA_OPTS, JASYPT_ENCRYPTOR
├── bin/
│   └── deploy-rolling.sh                # 로컬 실행용 배포/롤백 제어 스크립트
└── run/
    └── order-service.pid                # 프로세스 PID 파일
/logs/order-service/                     # [OS 루트 파티션 보호를 위해 별도 디스크 볼륨(/logs) 마운트]
├── app.json.log                         # 실시간 애플리케이션 구조화(JSON) 로그
├── access_log.2026-10-26.log            # Embedded Tomcat 내장 Access Log
└── dump/
    └── heapdump_oom.hprof               # OutOfMemoryError 발생 시 자동 덤프 저장 경로
```

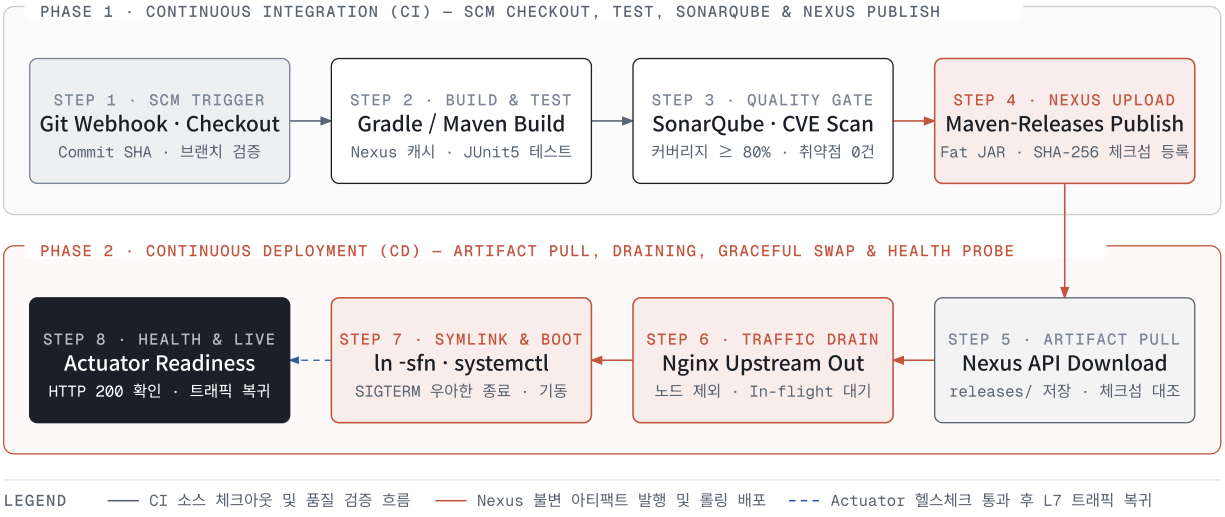
제3장. 프로세스 및 런타임 파이프라인 설계 (3. Process & Runtime Sequence View)

3.1 표준 8단계 CI/CD 빌드 및 무중단 롤링 배포 시퀀스 다이어그램

- **동적 파이프라인 흐름**: 전체 배포 공정은 **Phase 1(CI: `STEP 1` ~ `STEP 4`)**에서 소스 검증 및 불변 아티팩트를 생성하여 Nexus에 봉인하고, **Phase 2(CD: `STEP 5` ~ `STEP 8`)**에서 타겟 WAS 서버로 아티팩트를 반입한 뒤 Nginx 트래픽 Draining, Symlink 원자적 스왑, Actuator 헬스 프로브 검증을 거쳐 무중단으로 배포를 완결함.

VIEW 02 · PROCESS & CI/CD RUNTIME PIPELINE VIEW (8-STAGE SEQUENCE)

[프로세스 뷰] 표준 8단계 CI/CD 빌드 및 무중단 롤링 배포 파이프라인



View 2 프로세스 뷰 — 표준 8단계 CI/CD 빌드 및 무중단 롤링 배포 파이프라인 시퀀스

3.2 단계별 CI/CD 파이프라인 실행 계약 및 실패 시 대응 명세표

단계 ID	단계명 (Stage)	수행 주체	입력 계약 (Input)	상세
`STEP 1`	**SCM Trigger & Checkout**	Git → Jenkins Controller	Git Tag(`v1.4.2`) 또는 `release/*` Merge 이벤트	• Git 체크아 • `gi` 합으로
`STEP 2`	**Compile & Unit Test**	Jenkins Build Agent (`Gradle/Maven`)	소스코드 + Nexus `maven-public` 의존성	• `./` `inte` • 외부 **저장소**
`STEP 3`	**SonarQube & Security Gate**	SonarQube Scanner + Trivy/Dependency-Check	컴파일 클래스 + 의존성 트리(`build.gradle`)	• 정적 Code `80%`) • `wai` `true`
`STEP 4`	**Nexus Artifact Publish**	Jenkins Agent → Nexus 3 (`COMP-05`)	검증 완료된 Fat JAR + `pom.xml`	• `./` `mave` 드 • 업로 `SHA-` **(`.ja`**
`STEP 5`	**Target Artifact Pull & Verify**	Jenkins SSH Runner → Target WAS (`COMP-08`)	아티팩트 버전(`1.4.2`) + Nexus Read 자격증명	• 타겟 `/app` 및 `.s` • **`sh`** 다운로 대조
`STEP 6`	**Nginx Upstream Traffic Drain**	Deploy Script → Nginx (`COMP-07`)	배포 대상 노드 ID (예: `WAS-Node-1`)	• 앞단 `down` `relo` • 이미 **초간 유**
`STEP 7`	**Graceful Stop &**	`systemd` + Embedded Tomcat	`SIGTERM(15)` 시그널 + 신규 Fat JAR 경로	• `sy` Tomc

단계 ID	단계명 (Stage)	수행 주체	입력 계약 (Input)	상세
	Symlink Swap	(`COMP-08`)		**요청** **Shut** • `ln` `/app` `/app` 링크
`STEP 8`	**Actuator Probe & Traffic Live**	Deploy Script → Actuator (`8081`) → Nginx	`GET http://127.0.0.1:8081/actuator/health/readiness`	• 2초 드포인 확인 • UP 재투입 포로

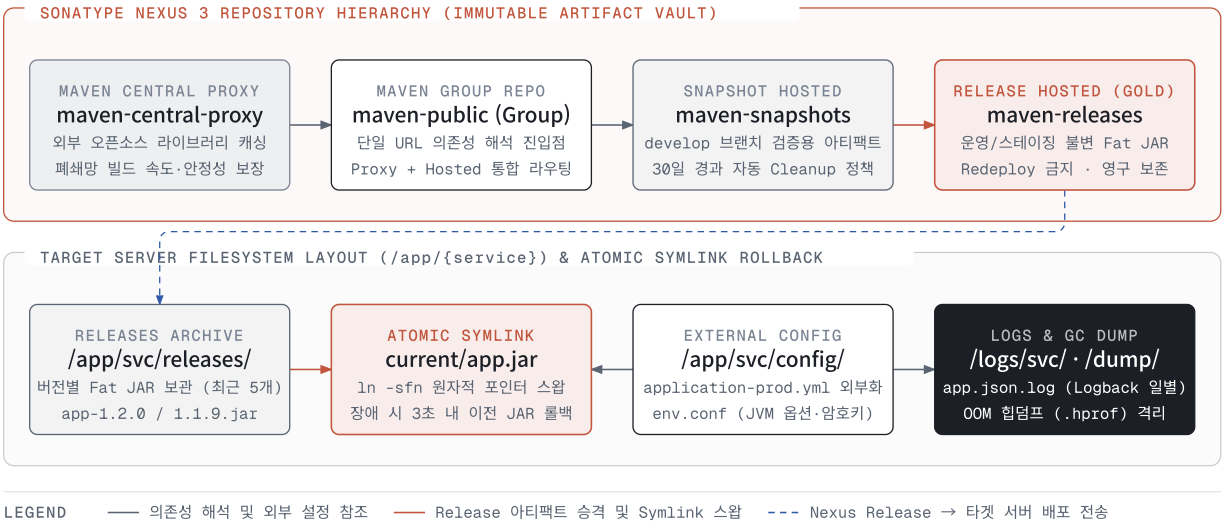
제4장. 데이터 및 영속성 아키텍처 설계 (4. Data & Persistence View)

4.1 Nexus 3단 아티팩트 저장소 계층 및 타겟 서버 파일시스템 구조도

- **배포 데이터 뷰 명세**: 배포 아키텍처에서의 핵심 영속 데이터는 **빌드 바이너리 아티팩트(Nexus Repository)**, **타겟 서버 릴리스 디렉토리 및 심볼릭 링크 상태(`/app/{service}`)**, **런타임 로그(`access_log` , `app.json.log`)**, 그리고 **무중단 배포를 위한 하위 호환 데이터베이스 스키마(Flyway Migration)**로 구성됨.

VIEW 03 · NEXUS REPOSITORY HIERARCHY & TARGET SERVER FILESYSTEM ARCHITECTURE

[데이터 뷰] Nexus 저장소 계층 및 타겟 서버 디렉토리·Symlink 구조도



View 3 데이터 뷰 — Sonatype Nexus 3 저장소 계층 및 타겟 서버 디렉토리·Symlink 구조도

4.2 아티팩트 저장소, 설정·로그 볼륨 및 DB 마이그레이션 정책 명세표

구분 ID	저장소 / 데이터 계층	적용 기술 / 경로	저장 대상 및 네이밍 규격	보존 주기 (Retention) 및 정합성 규격
`DEPLOY-DATA-01`	**Nexus Release Repo (`maven-releases`)**	Sonatype Nexus 3 (Blob Store: S3 / Enterprise NAS)	운영/스테이징 배포용 정식 Fat JAR (`com/company/{svc}/{ver}/{svc}-{ver}.jar`)	• **배포 정책:** `Disable redeploy` **(덮어쓰기 절대 불가)** • 보존 정책: 최근 2년간 모든 Release 버전 영구 보존 (감사 추적용)
`DEPLOY-DATA-02`	**Nexus Snapshot & Proxy (`snapshots` / `proxy`)**	Sonatype Nexus 3 (`maven-snapshots` , `maven-central-proxy`)	• 개발 브랜치 수시 빌드본 (`*-SNAPSHOT.jar`) • 외부 Maven Central 오픈소스 캐시	• Snapshot 보존: **최근 30일 또는 마지막 10개 빌드만 유지** 후 Nexus Cleanup Task로 자동 삭제 • Proxy 캐시: 외부 저장소 장애 시에도 로컬 캐시로 100% 빌드 보장
`DEPLOY-DATA-03`	**타겟 WAS 로컬 릴리스 및 설정 (`/app/{service}`)**	Linux XFS/EXT4 전용 파티션 (`/app`)	• `releases/app-{ver}.jar` (실행 바이너리) • `current/app.jar` (Symlink) • `config/application-{profile}.yml`	• **로컬 JAR 보존: 최근 5개 배포 버전만 유지** (배포 성공 시 오래된 JAR 자동 삭제 스크립트 실행) • 설정 파일은 `chmod 600` 으로 비인가 열람 차단

구분 ID	저장소 / 데이터 계층	적용 기술 / 경로	저장 대상 및 네이밍 규격	보존 주기 (Retention) 및 정합성 규격
`DEPLOY-DATA-04`	**애플리케이션·톰캣 로그 및 힙덤프 (`/logs/{service}`)**	별도 마운트 볼륨 (`/logs`) + Logback / Tomcat AccessLog	• `app.json.log` (비즈니스 로그) • `access_log.%d.log` (HTTP 요청/응답 시간) • `dump/heapdump.hprof` (OOM 덤프)	• **OS 영역 (`/`) 및 앱 영역(`/app`)과 물리 디스크 파티션 분리** (로그 폭주 시 OS 마비 방지) • 일별 롤링 + `gzip` 압축, 로컬 14일 보관 (장기 보관은 Promtail → Loki 전송
`DEPLOY-DATA-05`	**하위 호환 DB 스키마 버전 (`flyway_schema_history`)**	Flyway + RDBMS (`V{버전}__{설명}.sql`)	롤링 배포 중 구버전(`v1.4.1`)과 신버전 (`v1.4.2`) WAS가 동시에 바라보는 DB 스키마	• **Expand-and-Contract (확장 후 축소) 원칙 강제** 롤링 배포 중 구버전 파드가 죽지 않도록 컬럼 삭제 (`DROP`)/이름 변경 (`RENAME`) 금지, `NULL` 허용 컬럼 추가(`ADD`)만 허용

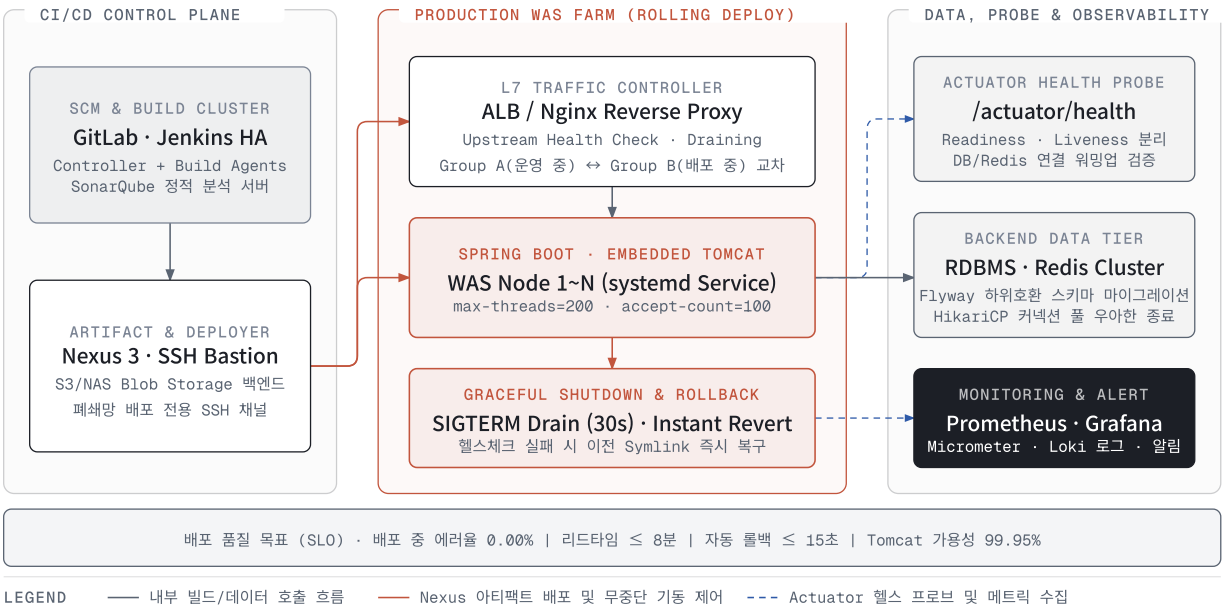
제5장. 물리 배포 및 회복탄력성·관측성 설계 (5. Deployment, Resilience & Observability View)

5.1 Multi-Node 무중단 롤링 배포 물리 토폴로지 및 관측성 구조도

- **물리 배치 및 런타임 제어 명세**: 배포 제어망(`CI/CD Control Plane`)의 Jenkins와 Nexus는 배포 전용 보안 채널(SSH)을 통해 운영망(`Production WAS Farm`)의 Nginx 및 다중 WAS 노드(`Group A` / `Group B`)를 순차 제어하며, 모든 Embedded Tomcat 인스턴스는 Actuator 헬스 프로브와 Prometheus/Grafana 관측성 스택에 실시간 연결됨.

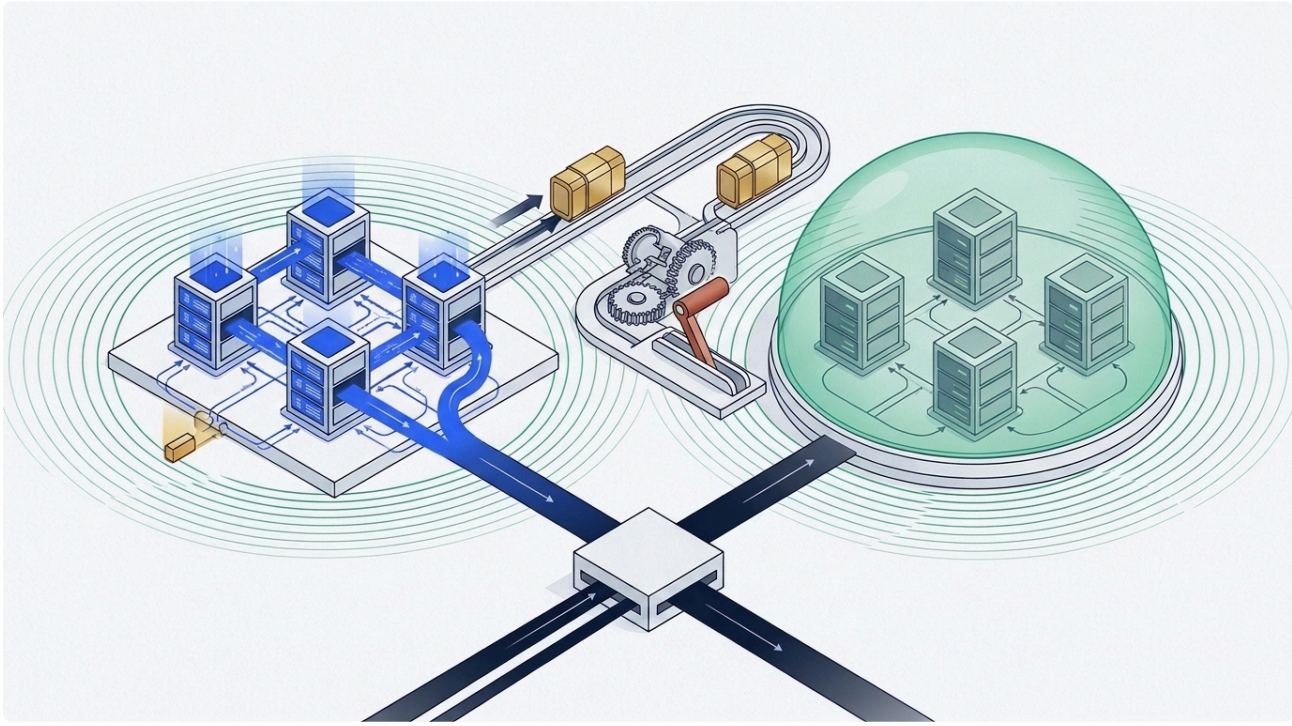
VIEW 04 · ZERO-DOWNTIME ROLLING DEPLOYMENT TOPOLOGY & EMBEDDED TOMCAT TUNING

[배포 뷰] 무중단 롤링 배포 런타임 토폴로지 및 관측성 설계도



View 4 배포 뷰 — Multi-Node 무중단 롤링 배포 물리 토폴로지 및 Embedded Tomcat 관측성 설계도

5.2 무중단 트래픽 드레이닝(Draining), Embedded Tomcat 튜닝 및 자동 롤백 설계



무중단 롤링 배포 트래픽 스위칭, Graceful Shutdown 및 심볼릭 링크 즉시 롤백 개념도

- **Embedded Tomcat · JVM · 무중단 배포 엔지니어링 파라미터 명세표 (Runtime Tuning Specification)**

규격 ID	설계 영역	적용 대상 컴포넌트	표준 설정 파라미터 및 임계치 명세 (Parameter Specification)	동
`TOMCAT-RES-01`	**Embedded Tomcat 스레드풀 및 커넥션 튜닝**	`application.yml` (`server.tomcat.*`)	• `server.tomcat.threads.max: 200`, `min-spare: 50` • `server.tomcat.max-connections: 8192` (NIO Poller 최대 동시 소켓) • `server.tomcat.accept-count: 100` (OS SYN/Accept 대기열) • `server.tomcat.connection-timeout: 5000ms`, `keep-alive-timeout: 20000ms`	•
`TOMCAT-RES-02`	**Graceful Shutdown 및 시스템 데몬 종료 규격**	Spring Boot 3.3 + Linux `systemd`	• `server.shutdown: graceful` • `spring.lifecycle.timeout-per-shutdown-phase: 30s` • `systemd` 유닛: `KillSignal=SIGTERM`, `TimeoutStopSec=40`, `SuccessExitStatus=143`	•
`TOMCAT-RES-03`	**프로덕션 JVM 메모리 및 GC/OOM 보호 규격**	OpenJDK 21 LTS (`config/env.conf`)	• `-Xms4g -Xmx4g -XX:MetaspaceSize=256m -XX:MaxMetaspaceSize=512m` • `-XX:+UseG1GC -XX:MaxGCPauseMillis=100 -XX:+ParallelRefProcEnabled` • `-XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=/logs/{svc}/dump/` • `-XX:+ExitOnOutOfMemoryError -Djava.security.egd=file:/dev/./urandom`	•
`TOMCAT-RES-04`	**Actuator 포트 분리 및 관측성 (Observability)**	Spring Boot Actuator + Micrometer	• `management.server.port: 8081` (외부 Nginx 미노출, 내부 관리망 전용) • `management.endpoint.health.probes.enabled: true` (`liveness` / `readiness`) • `management.metrics.enable.tomcat: true` (`tomcat_threads_busy_threads` 수집)	•

제6장. 표준 인터페이스 및 코드 템플릿 명세 (6. Standard Interface & Code Specification)

6.1 CI/CD 파이프라인 및 타겟 서버 배포 구성 표준 템플릿 (Reference Implementation Templates)

- **[Template 1] Spring Boot + Embedded Tomcat 런타임 및 Nexus 배포 설정 (`build.gradle.kts` & `application-prod.yml`)**

```yaml
# File: /app/order-service/config/application-prod.yml
server:
  port: 8080
  shutdown: graceful                                    # [필수] 우아한 종료 활성화 (SIGTERM 수신 시
  tomcat:
    threads:
      max: 200                                          # 최대 워커 스레드 수
      min-spare: 50                                     # 상시 유지 최소 스레드 수
    max-connections: 8192                               # NIO 최대 동시 연결 수
    accept-count: 100                                   # 스레드 포화 시 OS 소켓 큐 대기열 크기
    connection-timeout: 5000ms
    keep-alive-timeout: 20000ms                         # 앞단 Nginx keepalive_timeout(15s)보다
    max-keep-alive-requests: 1000
    basedir: /app/order-service/work                    # OS /tmp 자동 삭제(tmpwatch)로 인한 Mul
    accesslog:
      enabled: true
      directory: /logs/order-service
      prefix: access_log
      suffix: .log
      file-date-format: .yyyy-MM-dd
      pattern: '%h %l %u %t "%r" %s %b %D "%{X-Forwarded-For}i" "%{X-B3-TraceId}i"'

spring:
  lifecycle:
    timeout-per-shutdown-phase: 30s                     # Graceful Shutdown 최대 대기 시간 (30초
  jpa:
    open-in-view: false

management:
  server:
    port: 8081                                          # [보안] 관리/헬스체크 포트 분리 (외부 노출 차
  endpoint:
    health:
      probes:
        enabled: true                                   # /actuator/health/liveness, /actuat
      show-details: when-authorized
  endpoints:
    web:
      exposure:
        include: health,info,prometheus,metrics,threaddump
```

- **[Template 2] Jenkins 선언적 CI/CD 파이프라인 (`Jenkinsfile` — Git → Gradle → SonarQube → Nexus → Rolling Deploy)**

GROOVY

```groovy
// File: Jenkinsfile (Declarative Pipeline for Spring Boot + Nexus + Embedded Tomcat)
pipeline {
    agent { label 'linux-jdk21-agent' }
    options {
        timeout(time: 15, unit: 'MINUTES')
        disableConcurrentBuilds()                    // 동일 파이프라인 동시 배포 충돌 방지
        buildDiscarder(logRotator(numToKeepStr: '30'))
    }
    parameters {
        choice(name: 'TARGET_ENV', choices: ['staging', 'prod'], description: '배포 대상 환경
        string(name: 'RELEASE_VERSION', defaultValue: '1.4.2', description: '배포 아티팩트 버전
        booleanParam(name: 'ROLLBACK_MODE', defaultValue: false, description: '긴급 롤백 실행
    }
    environment {
        APP_NAME       = 'order-service'
        GROUP_PATH     = 'com/company/order'
        NEXUS_BASE_URL = 'https://nexus.internal.company.com/repository/maven-releases'
    }
    stages {
        stage('1. SCM Checkout & Build Info') {
            when { expression { !params.ROLLBACK_MODE } }
            steps {
                checkout scm
                sh "./gradlew --version"
            }
        }
        stage('2. Unit Test & Fat JAR Package') {
            when { expression { !params.ROLLBACK_MODE } }
            steps {
                withCredentials([usernamePassword(credentialsId: 'nexus-maven-cred', usern
                    sh "./gradlew clean test bootJar -Pversion=${params.RELEASE_VERSION} -
                }
            }
            post {
                always { junit '**/build/test-results/test/*.xml' }
            }
        }
        stage('3. SonarQube Quality Gate') {
            when { expression { !params.ROLLBACK_MODE } }
            steps {
                withSonarQubeEnv('SonarQube-Enterprise') {
                    sh "./gradlew sonar -Dsonar.projectVersion=${params.RELEASE_VERSION}"
                }
                timeout(time: 5, unit: 'MINUTES') {
                    waitForQualityGate abortPipeline: true
                }
            }
        }
        stage('4. Publish Immutable JAR to Nexus') {
            when { expression { !params.ROLLBACK_MODE } }
            steps {
                withCredentials([usernamePassword(credentialsId: 'nexus-maven-cred', usern
                    sh "./gradlew publish -Pversion=${params.RELEASE_VERSION} --no-daemon"
                }
```

```
            }
        }
        stage('5. Production Approval & Rolling Deploy') {
            steps {
                script {
                    if (params.TARGET_ENV == 'prod' && !params.ROLLBACK_MODE) {
                        input message: "[운영 배포 승인] ${APP_NAME} v${params.RELEASE_VERSION
                    }
                    def targetHosts = (params.TARGET_ENV == 'prod') ? ['10.10.20.11', '10.
                    def actionFlag  = params.ROLLBACK_MODE ? "--rollback" : "--deploy"

                    withCredentials([usernamePassword(credentialsId: 'nexus-maven-cred', u
                        sshagent(credentials: ['was-deploy-ssh-key']) {
                            for (host in targetHosts) {
                                echo ">>> Rolling Deploy on Target WAS Node: ${host}"
                                sh """
                                    ssh -o StrictHostKeyChecking=no appuser@${host} \
                                        "/app/${APP_NAME}/bin/deploy-rolling.sh ${actionFl
                                         --version ${params.RELEASE_VERSION} \
                                         --nexus-url ${NEXUS_BASE_URL}/${GROUP_PATH}/${APP
                                         --nexus-user ${NEXUS_USER} --nexus-pass '${NEXUS_
                                """
                            }
                        }
                    }
                }
            }
        }
    }
}
```

- **[Template 3] 타겟 WAS `systemd` 데몬 유닛 및 무중단 롤링 배포·롤백 스크립트 (`spring-app.service` & `deploy-rolling.sh`)**

INI

```
# File: /etc/systemd/system/order-service.service
[Unit]
Description=Spring Boot Embedded Tomcat Service (order-service)
After=network-online.target
Wants=network-online.target

[Service]
Type=simple
User=appuser
Group=appgroup
WorkingDirectory=/app/order-service
EnvironmentFile=/app/order-service/config/env.conf
ExecStart=/usr/lib/jvm/temurin-21-jdk/bin/java $JAVA_OPTS \
    -jar /app/order-service/current/app.jar \
    --spring.profiles.active=prod \
    --spring.config.additional-location=optional:file:/app/order-service/config/
KillMode=process
KillSignal=SIGTERM
TimeoutStopSec=40
SuccessExitStatus=143
Restart=on-failure
RestartSec=5
LimitNOFILE=65536
LimitNPROC=16384

[Install]
WantedBy=multi-user.target
```

BASH

```bash
#!/usr/bin/env bash
# File: /app/order-service/bin/deploy-rolling.sh
set -euo pipefail

APP_NAME="order-service"
APP_HOME="/app/${APP_NAME}"
RELEASES_DIR="${APP_HOME}/releases"
CURRENT_LINK="${APP_HOME}/current/app.jar"
NGINX_DRAIN_FLAG="/etc/nginx/drain/${APP_NAME}.drain"
HEALTH_URL="http://127.0.0.1:8081/actuator/health/readiness"

MODE="${1:---deploy}"
VERSION="${3:-}"
NEXUS_JAR_URL="${5:-}"
NEXUS_USER="${7:-}"
NEXUS_PASS="${9:-}"

drain_traffic() {
    echo "[STEP 6] Draining L7 traffic from current WAS node..."
    touch "${NGINX_DRAIN_FLAG}" && sudo /usr/sbin/nginx -s reload
    sleep 3
}

undrain_traffic() {
    echo "[STEP 8] Restoring L7 traffic to current WAS node..."
    rm -f "${NGINX_DRAIN_FLAG}" && sudo /usr/sbin/nginx -s reload
}

wait_for_readiness() {
    for i in $(seq 1 30); do
        STATUS=$(curl -sf "${HEALTH_URL}" | grep -o '"status":"UP"' || true)
        if [[ "${STATUS}" == '"status":"UP"' ]]; then
            echo "[OK] Embedded Tomcat Readiness UP verified (attempt ${i}/30)."
            return 0
        fi
        sleep 2
    done
    return 1
}

rollback_previous() {
     PREV_JAR=$(ls -1t "${RELEASES_DIR}"/${APP_NAME}-*.jar | sed -n '2p')
    echo "[ROLLBACK] Reverting symlink to previous golden artifact: ${PREV_JAR}"
    ln -sfn "${PREV_JAR}" "${CURRENT_LINK}"
    sudo /bin/systemctl restart "${APP_NAME}.service"
    wait_for_readiness && undrain_traffic
}

if [[ "${MODE}" == "--rollback" ]]; then
    drain_traffic
    rollback_previous
    exit 0
fi
```

```
# [STEP 5] Nexus에서 불변 아티팩트 및 SHA-256 체크섬 다운로드 후 무결성 대조
TARGET_JAR="${RELEASES_DIR}/${APP_NAME}-${VERSION}.jar"
curl -fSL -u "${NEXUS_USER}:${NEXUS_PASS}" -o "${TARGET_JAR}" "${NEXUS_JAR_URL}"
curl -fSL -u "${NEXUS_USER}:${NEXUS_PASS}" -o "${TARGET_JAR}.sha256" "${NEXUS_JAR_URL}.sha256
echo "$(cat "${TARGET_JAR}.sha256")  ${TARGET_JAR}" | sha256sum -c -

# [STEP 6 & 7] 트래픽 Draining → Graceful Shutdown → Symlink 원자적 스왑 → 기동
drain_traffic
sudo /bin/systemctl stop "${APP_NAME}.service"
ln -sfn "${TARGET_JAR}" "${CURRENT_LINK}"
sudo /bin/systemctl start "${APP_NAME}.service"

# [STEP 8] Readiness 헬스체크 검증 (실패 시 자동 롤백)
if wait_for_readiness; then
    undrain_traffic
    ls -1t "${RELEASES_DIR}"/${APP_NAME}-*.jar | tail -n +6 | xargs -r rm -f
    echo "[SUCCESS] Deployment of ${APP_NAME} v${VERSION} completed with zero downtime."
else
    echo "[ERROR] Readiness check failed! Initiating automatic rollback..."
    rollback_previous
    exit 1
fi
```

제7장. 아키텍처 적합성 검증 및 기술 스택 명세 (7. Architecture Fitness & Tech Stack BOM)

7.1 배포 파이프라인 및 런타임 적합성 자동 검증 규칙 (Deployment Fitness Functions)

모든 Spring Boot 서비스 리포지토리와 타겟 WAS 서버는 아래 5대 배포 적합성 검증 규칙(`DEPLOY-RULE-01` ~ `DEPLOY-RULE-05`)을 준수해야 하며, Jenkins 파이프라인 사전 검증(Pre-flight Check) 단계에서 단 1건이라도 위반 시 배포가 즉시 차단됨.

규칙 ID	배포 적합성 검증 규칙명	검증 대상	위반 판정 조건 (Violation Condition)	강제 수준
`DEPLOY-RULE-01`	**Nexus Release 덮어쓰기 및 SNAPSHOT 운영 배포 금지**	`Jenkinsfile`, `build.gradle.kts`, Nexus Repo	운영(`prod`) 배포 버전에 `-SNAPSHOT` 이 포함되거나, Nexus `maven-releases` 에 동일 버전 재배포가 시도될 시	**Blocker (배포 차단)**
`DEPLOY-RULE-02`	**소스 리포지토리 내 평문 시크릿 및 환경 IP 하드코딩 금지**	`src/main/resources/*.yml`, Git Commit	`application.yml` 내부에 평문 DB 비밀번호, JWT Secret, 운영 DB IP가 하드코딩되어 있을 시	**Blocker (CI 차단)**
`DEPLOY-RULE-03`	**Embedded Tomcat Graceful Shutdown 및 임시 디렉토리 분리 필수**	`application.yml`, `spring-app.service`	`server.shutdown: graceful` 이 누락되거나, `systemd TimeoutStopSec` 가 Spring `timeout-per-shutdown-phase` 이하로 설정될 시	**Blocker (CI 차단)**
`DEPLOY-RULE-04`	**타겟 WAS `root` 계정 기동 금지 및 체크섬 검증 강제**	`spring-app.service`, `deploy-rolling.sh`	`systemd` 유닛의 `User=root` 로 설정되거나, 배포 스크립트에서 `sha256sum -c` 무결성 검증이 생략될 시	**Blocker (배포 차단)**
`DEPLOY-RULE-05`	**Actuator 관리 포트 분리 및 Readiness 프로브 활성화**	`application-prod.yml`	비즈니스 포트(`8080`)와 Actuator 포트(`8081`)가 분리되지 않거나, `probes.enabled: true` 가 비활성화될 시	**Critical (PR 차단)**

7.2 전사 표준 빌드·배포·런타임 기술 스택 명세서 (Standard CI/CD & Runtime BOM)

기술 분류	표준 채택 기술 (Artifact / Engine)	표준 버전 (BOM)	아키텍처 역할 및 필수 구성 규격
Source Control (SCM)	GitLab Enterprise / GitHub Enterprise	**Git 2.45+ (SemVer 2.0)**	Protected Branch(`main` , `release/*`), Merge Request 코드리뷰, Webhook 트리거
CI/CD Orchestrator	Jenkins LTS (Controller + Agent Pool)	**v2.462.x LTS**	Declarative Pipeline(`Jenkinsfile`), Credentials Binding, SSH Agent, RBAC 승인 게이트
Build & Code Quality	Gradle (Kotlin DSL) + SonarQube LTS	**Gradle 8.10 / Sonar 9.9**	`spring-boot-gradle-plugin` , JUnit 5 커버리지 (JaCoCo ≥ 80%), `maven-publish` 플러그인
Artifact Repository	Sonatype Nexus Repository Manager 3	**v3.70.x**	`maven-central-proxy` , `maven-snapshots` , `maven-releases` (Redeploy 금지, SHA-256 검증)
Application & Embedded WAS	Spring Boot + Embedded Apache Tomcat	**Boot 3.3.x / Tomcat 10.1.x**	Executable Fat JAR, `server.shutdown=graceful` , NIO Connector(`max-threads=200`)
Java Runtime (JVM)	Eclipse Temurin OpenJDK	**Java 21 LTS**	`-Xms4g -Xmx4g -XX:+UseG1GC` , `-XX:+HeapDumpOnOutOfMemoryError` , `-XX:+ExitOnOutOfMemoryError`
Reverse Proxy & Process Manager	Nginx Stable + Linux `systemd`	**Nginx 1.26 / RHEL 9**	Upstream Draining(`keepalive 64`), `SIGTERM` 우아한 종료(`TimeoutStopSec=40`), `ln -sfn` Symlink 스왑
Health Probe & Observability	Spring Boot Actuator + Prometheus + Loki	**Micrometer 1.13.x**	관리 포트 분리(`8081`), `/actuator/health/readiness` , Tomcat 스레드/커넥션 메트릭 및 JSON 로그 수집