WBX Spring Framework

On-Premise 설치 가이드

Installation Guide — On-Premise v1.0

아큐라시스템 | 2026년 3월

목차

1. 사전 요구사항 · 서버 사양
2. OS 설치 · 기본 설정 (Linux)
3. JDK 21 설치
4. 데이터베이스 설치 (Oracle / MSSQL / MySQL / PostgreSQL)
5. Redis 설치
6. 애플리케이션 배포
7. Nginx 리버스 프록시 설정
8. systemd 서비스 등록
9. SSL/TLS 인증서 설정
10. 방화벽 · 보안 설정
11. WBX FastAPI 동시 운영 (선택)
12. 백업 · 복구
13. 모니터링 (Prometheus + Grafana)
14. 트러블슈팅
A. 설치 체크리스트

1. 사전 요구사항 · 서버 사양

1-1. 최소 서버 사양

항목	최소 사양	권장 사양
OS	RHEL 8+ / Ubuntu 22.04+	RHEL 9 / Rocky Linux 9
CPU	4 vCPU	8 vCPU
RAM	8 GB	16 GB
Disk	50 GB SSD	100 GB SSD (RAID)
Network	1 Gbps	10 Gbps

1-2. 필수 소프트웨어

소프트웨어	버전	용도
JDK	21 (Temurin LTS)	Spring Boot 런타임
DB	Oracle 19c+ / MSSQL 2019+	애플리케이션 데이터
	MySQL 8.0+ / PG 14+	(고객 선택)
Redis	7.x	캐시, 세션
Nginx	1.24+	리버스 프록시, SSL

1-3. 네트워크 포트

포트	서비스	접근 범위	비고
443	HTTPS	외부	Nginx SSL
80	HTTP	외부→443 리다이렉트	
8080	Spring Boot	내부(localhost)	Nginx에서만 접근
8001	WBX FastAPI	내부(localhost)	선택, 동시 운영 시
5432	PostgreSQL	내부	DB 선택에 따라
1521	Oracle	내부	
1433	MSSQL	내부	
3306	MySQL	내부	
6379	Redis	내부	bind 127.0.0.1

3. JDK 21 설치

3-1. Eclipse Temurin 설치 (권장)

```
# RHEL/Rocky
sudo dnf install -y https://packages.adoptium.net/artifactory/rpm/₩
  centos/9/$(uname -m)/Packages/temurin-21-jdk-21.0.6+7-1.$(uname -m).rpm

# Ubuntu
sudo apt install -y wget apt-transport-https gpg
wget -qO - https://packages.adoptium.net/artifactory/api/gpg/key/public ₩
  | sudo gpg --dearmor -o /usr/share/keyrings/adoptium.gpg
echo 'deb [signed-by=/usr/share/keyrings/adoptium.gpg] ₩
  https://packages.adoptium.net/artifactory/deb jammy main' ₩
  | sudo tee /etc/apt/sources.list.d/adoptium.list
sudo apt update && sudo apt install -y temurin-21-jdk
```

3-2. 설치 확인

```
java -version
# openjdk version "21.0.6" 2025-01-21 LTS

# JAVA_HOME 설정
echo 'export JAVA_HOME=/usr/lib/jvm/temurin-21-jdk' >> /etc/profile.d/java.sh
source /etc/profile.d/java.sh
```

3-3. 자동 설치 (install.sh)

install.sh 스크립트 실행 시 JDK가 없으면 자동으로 Temurin 21을 설치합니다.

- RHEL/Rocky: yum + Adoptium 저장소 자동 추가
- Ubuntu/Debian: apt + Adoptium 저장소 자동 추가
- macOS: Homebrew (brew install --cask temurin@21)
- Windows: winget install EclipseAdoptium.Temurin.21.JDK

4. 데이터베이스 설치

4-1. PostgreSQL 14+ (권장)

```
# RHEL/Rocky
sudo dnf install -y postgresql16-server postgresql16
sudo /usr/pgsql-16/bin/postgresql-16-setup initdb
sudo systemctl enable --now postgresql-16

# DB / 사용자 생성
sudo -u postgres psql << SQL
CREATE USER wbxapp WITH PASSWORD 'StrongP@ss123';
CREATE DATABASE wbx_spring OWNER wbxapp ENCODING 'UTF8';
CREATE DATABASE wbx_gw OWNER wbxapp ENCODING 'UTF8';
GRANT ALL PRIVILEGES ON DATABASE wbx_spring TO wbxapp;
SQL
```

4-2. Oracle 19c+

```
# Oracle은 고객 DBA가 설치/관리
# 아래 정보를 전달받아 application-oracle.yml에 설정

# 필요 정보:
#   HOST: 192.168.1.100
#   PORT: 1521
#   SERVICE: WBXDB
#   USER: wbxapp
#   PASSWORD: (암호화 전달)

# 테이블스페이스 생성 (DBA)
CREATE TABLESPACE wbx_ts DATAFILE '/oradata/wbx_ts.dbf' SIZE 1G AUTOEXTEND ON;
CREATE USER wbxapp IDENTIFIED BY StrongP@ss DEFAULT TABLESPACE wbx_ts;
GRANT CONNECT, RESOURCE TO wbxapp;
```

4-3. MSSQL 2019+

```
# MSSQL on Linux (RHEL)
sudo curl -o /etc/yum.repos.d/mssql-server.repo ₩
  https://packages.microsoft.com/config/rhel/9/mssql-server-2022.repo
sudo dnf install -y mssql-server
sudo /opt/mssql/bin/mssql-conf setup  # SA 비밀번호 설정
sudo systemctl enable --now mssql-server

# DB 생성
sqlcmd -S localhost -U SA -P 'StrongP@ss123' -Q ₩
  'CREATE DATABASE wbx_spring; CREATE DATABASE wbx_gw;'
```

4-4. MySQL 8.0+

```
sudo dnf install -y mysql-server
sudo systemctl enable --now mysqld
sudo mysql_secure_installation

mysql -u root -p << SQL
CREATE DATABASE wbx_spring CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
CREATE DATABASE wbx_gw CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
CREATE USER 'wbxapp'@'localhost' IDENTIFIED BY 'StrongP@ss123';
GRANT ALL ON wbx_spring.* TO 'wbxapp'@'localhost';
GRANT ALL ON wbx_gw.* TO 'wbxapp'@'localhost';
SQL
```

5. Redis 설치

```
sudo dnf install -y redis
sudo systemctl enable --now redis

# 보안: bind + password
sudo sed -i 's/^bind .*/bind 127.0.0.1/' /etc/redis/redis.conf
echo 'requirepass RedisP@ss123' | sudo tee -a /etc/redis/redis.conf
sudo systemctl restart redis

# 확인
redis-cli -a RedisP@ss123 ping  # PONG
```

6. 애플리케이션 배포

자동 배포 스크립트 (권장)

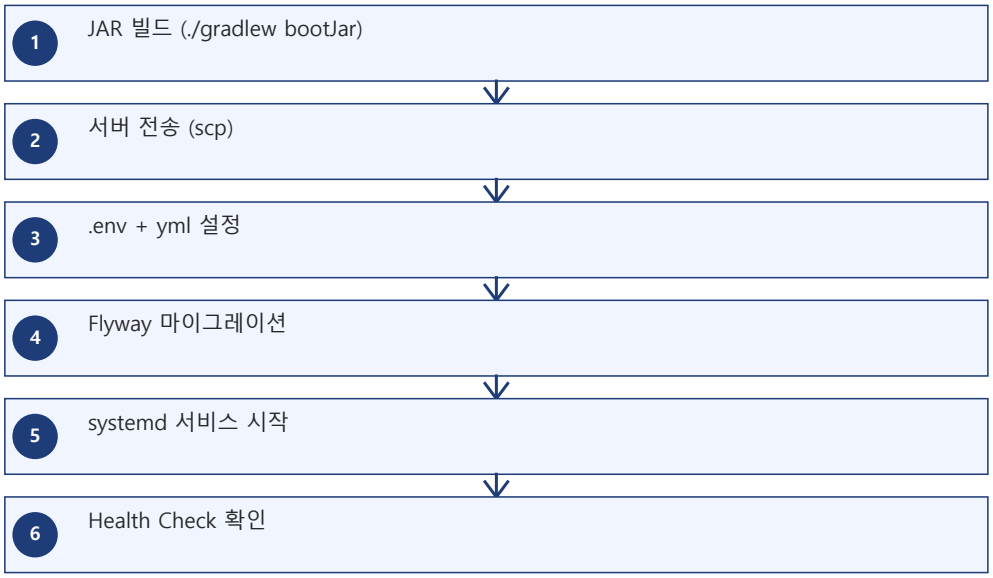
프로젝트에 포함된 deploy-prod.sh 스크립트가 서비스 계정 생성, 디렉토리 구조, systemd 등록, 백업 cron까지 자동 처리합니다.

```
# 소스에서 빌드 후 배포 스크립트 실행
./gradlew bootJar
sudo scripts/deploy-prod.sh

# 스크립트가 자동 처리하는 항목:
# - 서비스 계정(wbxapp) 생성
# - /opt/wbx-app 디렉토리 생성
# - .env 템플릿 생성
# - systemd 서비스 등록
# - DB 백업 cron 등록
# - 방화벽/SELinux 설정
```

TIP: 스크립트 실행 후 .env 파일의 JWT_SECRET, DB_PASS를 반드시 수정하세요.

수동 배포 플로우



6-1. 디렉토리 구조

```
/opt/wbx-app/
  ├── app.jar                # Spring Boot Fat JAR
  ├── application-prod.yml   # 프로덕션 설정
  ├── application-{db}.yml   # DB 프로필 (oracle/mssql/mysql/postgresql)
```

```
├── .env                # 환경변수 (시크릿)
├── logs/               # 로그
├── uploads/            # 파일 업로드
└── backup/             # 백업
```

6-2. JAR 배포

```
# 빌드 서버에서 JAR 생성
./gradlew bootJar

# 운영 서버로 전송
scp build/libs/wbx-spring-core-*.jar wbxapp@server:/opt/wbx-app/app.jar
```

6-3. application-prod.yml

```
server:
  port: 8080
  forward-headers-strategy: native

spring:
  profiles:
    include: ${DB_PROFILE:postgresql}   # oracle/mssql/mysql/postgresql

wbx:
  spring:
    api-prefix: /api
    jwt:
      secret: ${JWT_SECRET}
      expiration: 28800
    admin-ui:
      enabled: true          # WBX 없으면 true
    cors:
      allowed-origins:
        - https://app.company.com

springdoc:
  swagger-ui:
    enabled: false           # 프로덕션 비활성화
```

6-4. .env (시크릿)

```
SPRING_PROFILES_ACTIVE=prod,mysql
SERVER_CONTEXT_PATH=/
JWT_SECRET=your-256bit-secret-key-here
DB_HOST=localhost
DB_PORT=3306
DB_NAME=wbx_spring
DB_USER=wbxapp
DB_PASS=StrongP@ss123
CORS_ORIGINS=https://app.company.com
```

```
LOG_PATH=/opt/wbx-app/logs/app.log
```

주의: .env 파일 권한: chmod 600 .env (소유자만 읽기)

6-5. Flyway 마이그레이션 실행

```
# 첫 실행 시 자동 마이그레이션
java -jar app.jar --spring.profiles.active=prod,mysql

# 로그에서 확인
# Flyway: Successfully applied 12 migrations
```

7. Nginx 리버스 프록시

7-1. 설치

```
sudo dnf install -y nginx
sudo systemctl enable --now nginx
```

7-2. 설정 파일

```
# /etc/nginx/conf.d/wbx-app.conf
upstream spring_app {
    server 127.0.0.1:8080;
    keepalive 32;
}

server {
    listen 80;
    server_name app.company.com;
    return 301 https://$host$request_uri;
}

server {
    listen 443 ssl http2;
    server_name app.company.com;

    ssl_certificate     /etc/ssl/certs/app.pem;
    ssl_certificate_key /etc/ssl/private/app.key;

    # Security Headers
    add_header X-Frame-Options SAMEORIGIN always;
    add_header X-Content-Type-Options nosniff always;
    add_header Strict-Transport-Security max-age=63072000 always;

    # Spring Boot API
    location /api/ {
        proxy_pass http://spring_app;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        client_max_body_size 50m;
    }

    # SSE
    location /api/notifications/stream {
        proxy_pass http://spring_app;
        proxy_http_version 1.1;
```

```
        proxy_set_header Connection '';
        proxy_buffering off;
        proxy_read_timeout 3600s;
    }

    # Admin Console
    location /admin {
        proxy_pass http://spring_app;
    }

    # Health
    location /health {
        proxy_pass http://spring_app;
    }

    # Static (Frontend SPA)
    location / {
        root /var/www/wbx-app/frontend;
        try_files $uri $uri/ /index.html;
    }
}
```

7-3. 확인

```
sudo nginx -t
sudo systemctl reload nginx
```

8. systemd 서비스 등록

서비스 관리 명령어

시작:	systemctl start wbx-app
중지:	systemctl stop wbx-app
재시작:	systemctl restart wbx-app
상태:	systemctl status wbx-app
로그:	journalctl -u wbx-app -f

```
# /etc/systemd/system/wbx-app.service
[Unit]
Description=WBX Spring Application
After=network.target

[Service]
Type=simple
User=wbxapp
WorkingDirectory=/opt/wbx-app
EnvironmentFile=/opt/wbx-app/.env
ExecStart=/usr/bin/java ₩
  -XX:+UseG1GC ₩
  -XX:MaxRAMPercentage=75.0 ₩
  -Dspring.profiles.active=${SPRING_PROFILES_ACTIVE} ₩
  -jar /opt/wbx-app/app.jar
Restart=always
RestartSec=5
StandardOutput=append:/opt/wbx-app/logs/app.log
StandardError=append:/opt/wbx-app/logs/app.log

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

```
sudo systemctl daemon-reload
sudo systemctl enable --now wbx-app
sudo systemctl status wbx-app

# 로그 확인
journalctl -u wbx-app -f
```

9. SSL/TLS 인증서

9-1. 고객 제공 인증서 (권장)

```
# 고객 인증서 파일 배치
sudo cp app.pem /etc/ssl/certs/
sudo cp app.key /etc/ssl/private/
sudo chmod 600 /etc/ssl/private/app.key
```

9-2. Let's Encrypt (테스트/내부)

```
sudo dnf install -y certbot python3-certbot-nginx
sudo certbot --nginx -d app.company.com

# 자동 갱신
sudo systemctl enable --now certbot-renew.timer
```

10. 방화벽 · 보안 설정

10-1. firewalld

```
sudo firewall-cmd --permanent --add-service=http
sudo firewall-cmd --permanent --add-service=https
sudo firewall-cmd --reload

# DB 포트는 외부 차단 (기본)
# 내부 접근만 허용
```

10-2. SELinux

```
# Nginx → Spring Boot 프록시 허용
sudo setsebool -P httpd_can_network_connect 1
```

10-3. 보안 체크리스트

- SSH 키 기반 인증 (비밀번호 로그인 비활성화)
- 서비스 계정(wbxapp)에 sudo 권한 없음
- DB 포트 외부 차단 (localhost만 접근)
- Redis bind 127.0.0.1 + requirepass
- .env 파일 600 권한
- Swagger UI 프로덕션 비활성화
- Nginx 보안 헤더 (HSTS, X-Frame, X-Content-Type)

11. WBX FastAPI 동시 운영 (선택)

WBX Groupware(FastAPI)와 Spring Boot를 동시 운영하는 경우, Nginx에서 URL prefix로 분기합니다.

```
# Nginx 추가 설정
upstream wbx_fastapi {
    server 127.0.0.1:8001;
}

location /api/gw/ {
    proxy_pass http://wbx_fastapi;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
}
```

주의: JWT SECRET_KEY를 두 시스템이 동일하게 설정해야 SSO가 작동합니다.

12. 백업 · 복구

12-1. DB 백업 (cron)

```
# /opt/wbx-app/backup/db-backup.sh
#!/bin/bash
DATE=$(date +%Y%m%d_%H%M%S)
pg_dump -U wbxapp wbx_spring | gzip > /opt/wbx-app/backup/wbx_spring_${DATE}.sql.gz
find /opt/wbx-app/backup -name '*.gz' -mtime +30 -delete

# crontab -e
0 2 * * * /opt/wbx-app/backup/db-backup.sh
```

12-2. 복구

```
# 복구
gunzip -c wbx_spring_20260324.sql.gz | psql -U wbxapp wbx_spring
```

12-3. 앱 롤백

```
# 이전 JAR로 롤백
cp /opt/wbx-app/app.jar /opt/wbx-app/app.jar.current
cp /opt/wbx-app/backup/app-prev.jar /opt/wbx-app/app.jar
sudo systemctl restart wbx-app
```

13. 모니터링 (Prometheus + Grafana)

```
# Spring Boot Actuator → Prometheus 메트릭
# application-prod.yml:
management:
  endpoints:
    web:
      exposure:
        include: health,info,metrics,prometheus

# Prometheus scrape config
# prometheus.yml:
scrape_configs:
  - job_name: wbx-app
    metrics_path: /actuator/prometheus
    static_configs:
      - targets: ['localhost:8080']
```

Grafana에서 Spring Boot 대시보드 Import: ID 12900 (JVM Micrometer)

14. 트러블슈팅

증상	원인	해결
서비스 시작 실패	포트 충돌	lsof -i :8080 확인, 기존 프로세스 종료
DB 연결 실패	방화벽/인증	telnet DB_HOST PORT, .env 확인
502 Bad Gateway	앱 미기동	systemctl status wbx-app, logs 확인
Flyway 실패	DDL 호환	DB 프로필 확인 (oracle/mssql/mysql/pg)
SSL 오류	인증서 경로	nginx -t, 인증서 만료 확인
OOM (메모리)	힙 부족	-XX:MaxRAMPercentage=75 또는 RAM 증설

A. 설치 체크리스트

- [ ] 1. OS 업데이트 및 기본 패키지 설치
- [ ] 2. 서비스 계정 (wbxapp) 생성
- [ ] 3. 타임존 Asia/Seoul, 로케일 UTF-8
- [ ] 4. JDK 21 설치 및 JAVA_HOME 설정
- [ ] 5. 데이터베이스 설치 및 DB/사용자 생성
- [ ] 6. Redis 설치, bind 127.0.0.1, requirepass
- [ ] 7. 앱 디렉토리 /opt/wbx-app 생성
- [ ] 8. app.jar 배포
- [ ] 9. application-prod.yml, .env 설정
- [ ] 10. systemd 서비스 등록 및 시작
- [ ] 11. Nginx 설치 및 리버스 프록시 설정
- [ ] 12. SSL 인증서 설치
- [ ] 13. 방화벽 80/443 오픈, DB 포트 내부만
- [ ] 14. SELinux httpd_can_network_connect
- [ ] 15. Health Check: curl https://app.company.com/health
- [ ] 16. Swagger 비활성화 확인
- [ ] 17. DB 백업 cron 등록
- [ ] 18. 모니터링 Prometheus 연동 (선택)

설치 완료

문의: accura@accurasoft.co.kr