

WBX Spring Framework

Cloud 설치 가이드

Azure · AWS · GCP

Installation Guide — Cloud v1.0

아큐라시스템 | 2026년 3월

목차

1. 클라우드 선택 가이드
2. Azure 배포 (상세)
3. AWS 배포 (상세)
4. GCP 배포 (상세)
5. Terraform 인프라 자동화 (공통)
6. Kubernetes 배포 (AKS / EKS / GKE)
7. CI/CD 파이프라인 연동
8. 시크릿 관리
9. 모니터링 · 로깅 · 알람
10. 비용 최적화
11. 보안 · 컴플라이언스
- A. 클라우드별 설치 체크리스트

1. 클라우드 선택 가이드

1-1. 플랫폼 비교

항목	Azure	AWS	GCP
DB (PaaS)	Azure SQL, PG	RDS, Aurora	Cloud SQL, AlloyDB
컨테이너	AKS, App Service	EKS, ECS Fargate	GKE, Cloud Run
SSO	Entra ID	Cognito	Cloud Identity
파일	Blob Storage	S3	Cloud Storage
시크릿	Key Vault	Secrets Manager	Secret Manager
모니터링	Monitor	CloudWatch	Cloud Monitoring
CDN	Front Door	CloudFront	Cloud CDN
IaC	Terraform/Bicep	Terraform/CDK	Terraform/Pulumi

1-2. 프로파일 조합

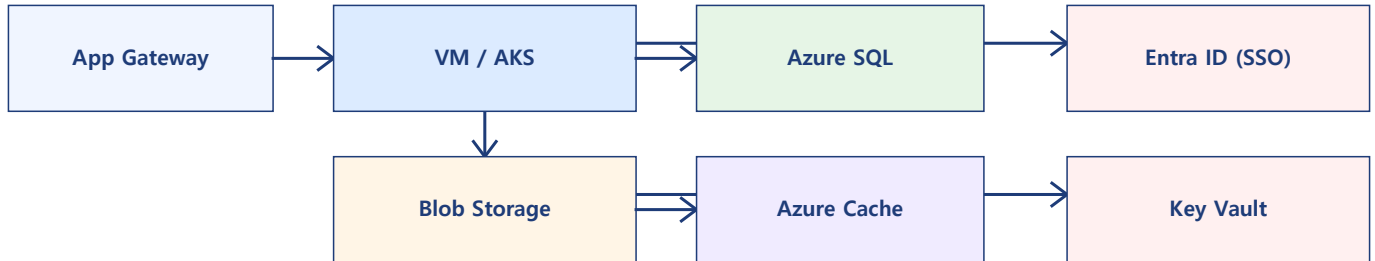
```
# Azure + Azure SQL
java -jar app.jar --spring.profiles.active=prod,azure,mssql

# AWS + PostgreSQL (Aurora)
java -jar app.jar --spring.profiles.active=prod,aws,postgresql

# GCP + PostgreSQL (Cloud SQL)
java -jar app.jar --spring.profiles.active=prod,gcp,postgresql
```

2. Azure 배포 (상세)

Azure 아키텍처



2-1. 리소스 생성 (Azure CLI)

```

# 리소스 그룹
az group create -n wbx-app-rg -l koreacentral

# Azure SQL Database
az sql server create -n wbx-sql -g wbx-app-rg \
  -l koreacentral --admin-user sqladmin --admin-password 'StrongP@ss'
az sql db create -n wbx_spring -s wbx-sql -g wbx-app-rg \
  --service-objective S1

# Azure Cache for Redis
az redis create -n wbx-redis -g wbx-app-rg -l koreacentral \
  --sku Basic --vm-size C0

# Azure Blob Storage
az storage account create -n wbxappstorage -g wbx-app-rg \
  -l koreacentral --sku Standard_LRS
az storage container create -n uploads --account-name wbxappstorage

# VM (App Server)
az vm create -n wbx-app-vm -g wbx-app-rg \
  --image Ubuntu2204 --size Standard_D2s_v3 \
  --admin-username azureuser --ssh-key-values ~/.ssh/id_rsa.pub

# Entra ID App Registration (SSO)
az ad app create --display-name 'WBX Spring App' \
  --web-redirect-uris 'https://app.company.com/api/auth/sso/callback'
  
```

2-2. application-azure.yml

```

spring:
  cloud:
  
```

```

azure:
  active-directory:
    enabled: true
    profile:
      tenant-id: ${AZURE_TENANT_ID}
  credential:
    client-id: ${AZURE_CLIENT_ID}
    client-secret: ${AZURE_CLIENT_SECRET}
  storage:
    blob:
      account-name: ${AZURE_STORAGE_ACCOUNT}
      account-key: ${AZURE_STORAGE_KEY}
  datasource:
    app:
      url: jdbc:sqlserver://${DB_HOST};1433;database=${DB_NAME};%
      encrypt=true;trustServerCertificate=false
wbx:
  spring:
    file:
      storage-type: azure-blob

```

2-3. App Service 배포 (대안)

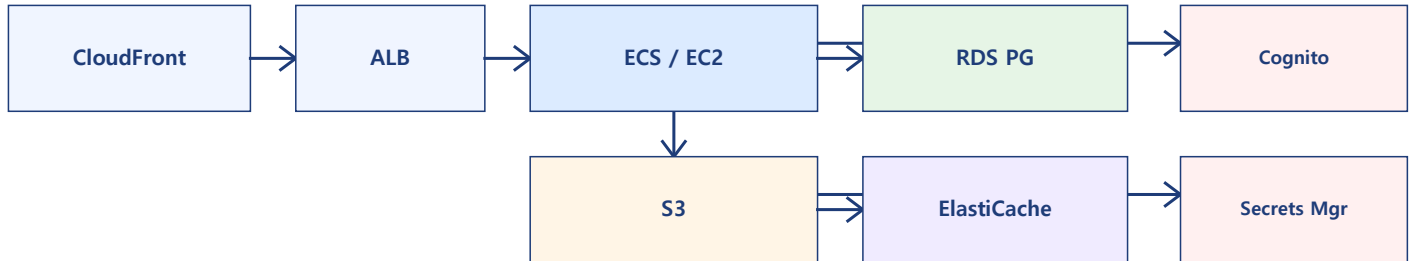
```

# Maven/Gradle 플러그인으로 직접 배포
az webapp create -n wbx-app -g wbx-app-rg %
  --runtime 'JAVA:21-java21' --plan wbx-plan
az webapp config appsettings set -n wbx-app -g wbx-app-rg %
  --settings SPRING_PROFILES_ACTIVE=prod,azure,mssql
az webapp deploy -n wbx-app -g wbx-app-rg %
  --src-path build/libs/app.jar --type jar

```

3. AWS 배포 (상세)

AWS 아키텍처



3-1. 리소스 생성 (AWS CLI)

```

# VPC (기존 사용 또는 생성)
aws ec2 create-vpc --cidr-block 10.0.0.0/16

# RDS PostgreSQL
aws rds create-db-instance ₩
  --db-instance-identifier wbx-db ₩
  --db-instance-class db.t3.medium ₩
  --engine postgres --engine-version 16 ₩
  --master-username dbadmin --master-user-password 'StrongP@ss'₩
  --allocated-storage 50

# ElastiCache Redis
aws elasticache create-cache-cluster ₩
  --cache-cluster-id wbx-redis ₩
  --engine redis --cache-node-type cache.t3.micro ₩
  --num-cache-nodes 1

# S3 Bucket
aws s3 mb s3://wbx-app-uploads --region ap-northeast-2

# EC2
aws ec2 run-instances ₩
  --image-id ami-0c55b159cbfafa1f0 ₩
  --instance-type t3.medium ₩
  --key-name wbx-key --security-groups wbx-sg
  
```

3-2. application-aws.yml

```

spring:
  cloud:
    aws:
  
```

```

region:
  static: ap-northeast-2
s3:
  bucket: ${AWS_S3_BUCKET}
datasource:
  app:
    url: jdbc:postgresql://${DB_HOST}:5432/${DB_NAME}
security:
  oauth2:
    client:
      registration:
        cognito:
          client-id: ${AWS_COGNITO_CLIENT_ID}
          client-secret: ${AWS_COGNITO_CLIENT_SECRET}
wbx:
  spring:
    file:
      storage-type: aws-s3

```

3-3. ECS Fargate 배포

```

# ECR 레지스트리에 이미지 Push
aws ecr get-login-password | docker login --username AWS ₩
--password-stdin {account}.dkr.ecr.ap-northeast-2.amazonaws.com
docker build -t wbx-app .
docker tag wbx-app:latest {account}.dkr.ecr.../wbx-app:latest
docker push {account}.dkr.ecr.../wbx-app:latest

# ECS 서비스 생성
aws ecs create-service ₩
--cluster wbx-cluster ₩
--service-name wbx-app ₩
--task-definition wbx-app:1 ₩
--desired-count 2 ₩
--launch-type FARGATE

```

4. GCP 배포 (상세)

4-1. 리소스 생성 (gcloud)

```
# 프로젝트
gcloud projects create wbx-app-prod
gcloud config set project wbx-app-prod

# Cloud SQL PostgreSQL
gcloud sql instances create wbx-db ₩
--database-version=POSTGRES_16 ₩
--tier=db-custom-2-8192 --region=asia-northeast3
gcloud sql databases create wbx_spring --instance=wbx-db

# Memorystore Redis
gcloud redis instances create wbx-redis ₩
--size=1 --region=asia-northeast3

# Cloud Storage
gsutil mb -l asia-northeast3 gs://wbx-app-uploads
```

4-2. Cloud Run 배포 (서버리스)

```
# Artifact Registry에 Push
gcloud builds submit --tag ₩
asia-northeast3-docker.pkg.dev/wbx-app-prod/repo/wbx-app

# Cloud Run 배포
gcloud run deploy wbx-app ₩
--image asia-northeast3-docker.pkg.dev/.../wbx-app ₩
--region asia-northeast3 ₩
--platform managed ₩
--set-env-vars SPRING_PROFILES_ACTIVE=prod,gcp,postgresql ₩
--set-env-vars DB_HOST=/cloudsql/wbx-app-prod:asia-northeast3:wbx-db ₩
--add-cloudsql-instances wbx-app-prod:asia-northeast3:wbx-db ₩
--allow-unauthenticated
```

TIP: Cloud Run은 자동 스케일링, SSL, 도메인 매핑을 기본 제공합니다.

5. Terraform 인프라 자동화

5-1. 디렉토리 구조

```
infra/terraform/
├── main.tf           # Provider, Resource Group
├── network.tf        # VNet/VPC, Subnet, NSG
├── compute.tf        # VM 또는 Container 서비스
├── database.tf       # 관리형 DB
├── redis.tf          # 캐시
├── storage.tf        # Blob / S3 / GCS
├── identity.tf       # SSO App Registration
├── monitoring.tf     # 모니터링
├── variables.tf      # 입력 변수
├── outputs.tf        # 출력 (URL, IP 등)
└── environments/
    ├── azure.tfvars
    ├── aws.tfvars
    └── gcp.tfvars
```

5-2. 실행

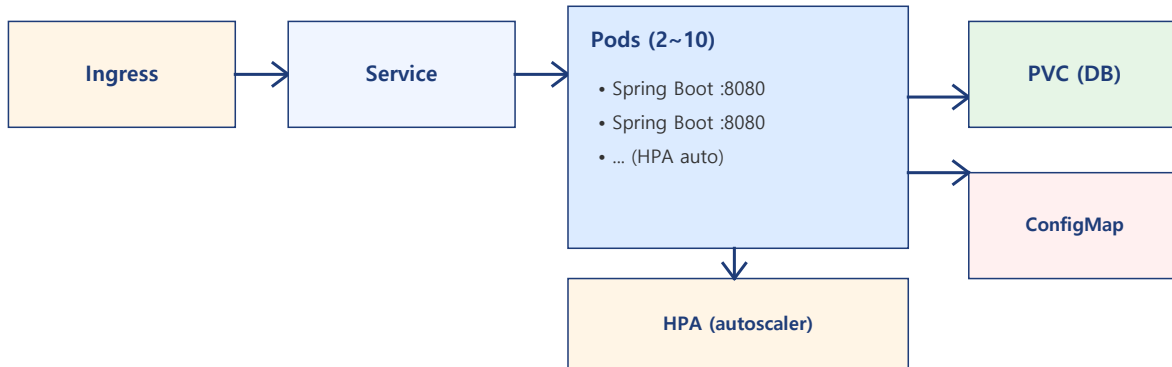
```
cd infra/terraform

# Azure
terraform init
terraform plan -var-file=environments/azure.tfvars
terraform apply -var-file=environments/azure.tfvars

# AWS
terraform plan -var-file=environments/aws.tfvars
terraform apply -var-file=environments/aws.tfvars
```

6. Kubernetes 배포

K8s 배포 구조



6-1. Helm Chart

```

# Azure AKS
helm install wbx-app ./helm/wbx-app -f values-azure.yaml

# AWS EKS
helm install wbx-app ./helm/wbx-app -f values-aws.yaml

# GCP GKE
helm install wbx-app ./helm/wbx-app -f values-gcp.yaml
  
```

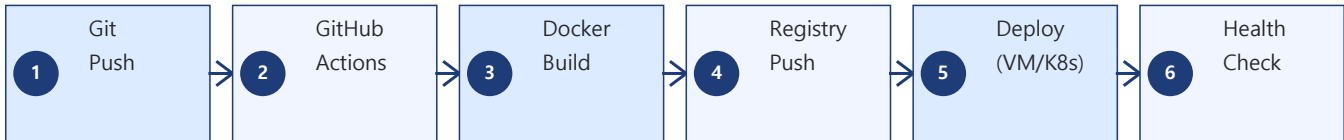
6-2. values.yaml 핵심 설정

```

replicaCount: 2
image:
  repository: myregistry/wbx-app
  tag: latest
env:
  SPRING_PROFILES_ACTIVE: prod,azure,mssql
resources:
  requests: { cpu: 500m, memory: 512Mi }
  limits: { cpu: 2000m, memory: 2Gi }
autoscaling:
  enabled: true
  minReplicas: 2
  maxReplicas: 10
  targetCPU: 70
health:
  liveness: /actuator/health/liveness
  readiness: /actuator/health/readiness
  
```

7. CI/CD 파이프라인

CI/CD 파이프라인



7-1. GitHub Actions → 클라우드 배포

```

# .github/workflows/deploy.yml
name: Deploy
on:
  push:
    branches: [main]
jobs:
  build-and-deploy:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-java@v4
        with: { java-version: '21', distribution: temurin }
      - run: ./gradlew test bootJar
      - run: docker build -t app .
      # Azure: ACR Push + VM/AKS Deploy
      # AWS: ECR Push + ECS/EKS Deploy
      # GCP: Artifact Registry Push + Cloud Run Deploy
  
```

7-2. Azure DevOps + 승인 게이트

Azure DevOps Pipeline에서 Staging → Production 2단계 배포.

Production 환경에 승인 게이트(수동 확인) 설정.

상세: plans/wbx-spring/07-infra-deploy.md

8. 시크릿 관리

시크릿	Azure	AWS	GCP
DB 비밀번호	Key Vault	Secrets Manager	Secret Manager
JWT Secret	Key Vault	Secrets Manager	Secret Manager
SSO Credential	Key Vault	Secrets Manager	Secret Manager
Storage Key	Key Vault	IAM Role	Service Account

주의: 시크릿은 코드/환경변수에 직접 노출하지 않고, 클라우드 시크릿 매니저를 통해 주입합니다.

9. 모니터링 · 로깅 · 알람

기능	Azure	AWS	GCP
메트릭	Azure Monitor	CloudWatch	Cloud Monitoring
로그	Log Analytics	CloudWatch Logs	Cloud Logging
트레이싱	App Insights	X-Ray	Cloud Trace
알람	Monitor Alerts	CloudWatch Alarms	Alerting Policies

권장 알람 규칙

- CPU > 80% (5분 지속)
- Memory > 85%
- 5xx 에러 > 10건/분
- 응답 시간 > 3초 (P95)
- DB 연결 풀 사용률 > 80%
- 디스크 사용률 > 85%

10. 비용 최적화

10-1. 월 예상 비용 (소규모)

항목	Azure	AWS	GCP
VM (2vCPU/8GB)	~\$70	~\$65	~\$60
관리형 DB	~\$100	~\$90	~\$85
Redis (1GB)	~\$40	~\$35	~\$35
Storage (100GB)	~\$2	~\$2	~\$2
LB	~\$20	~\$20	~\$18
합계	~\$232/월	~\$212/월	~\$200/월

10-2. 절감 팁

- 예약 인스턴스 (1년/3년) → 30~60% 할인
- Spot/Preemptible VM → 개발/테스트 환경
- 서버리스 (Cloud Run, Fargate) → 사용한 만큼만 과금
- 자동 스케일링 → 피크 시에만 스케일 아웃
- DB 자동 일시 중지 (Azure Serverless SQL)

11. 보안 · 컴플라이언스

- 네트워크: VNet/VPC + Private Endpoint + NSG/Security Group
- 데이터: DB 암호화 (TDE/KMS), 전송 중 TLS 1.2+
- 접근: IAM 최소 권한, 서비스 계정 분리
- 감사: 클라우드 감사 로그 활성화 (Activity Log / CloudTrail / Audit Log)
- WAF: Azure WAF / AWS WAF / Cloud Armor
- DDoS: Azure DDoS Protection / AWS Shield / Cloud Armor
- 취약점 스캔: Defender / Inspector / Security Command Center

A. 클라우드별 설치 체크리스트

Azure

1. Resource Group 생성
2. Azure SQL 또는 PostgreSQL PaaS 생성
3. Azure Cache for Redis 생성
4. Blob Storage 계정 + Container 생성
5. Entra ID App Registration (SSO)
6. Key Vault에 시크릿 저장 (DB PW, JWT Secret)
7. VM 생성 또는 AKS 클러스터 생성
8. 앱 배포 (JAR 또는 Docker)
9. application-azure.yml + application-mssql.yml 설정
10. NSG 방화벽 규칙 (443 오픈, DB 내부만)
11. Health Check 확인
12. Monitor + Log Analytics 연동

AWS

1. VPC + Subnet + Security Group 생성
2. RDS PostgreSQL 또는 Aurora 생성
3. ElastiCache Redis 생성
4. S3 Bucket 생성
5. Cognito User Pool (SSO) 또는 AD Connector
6. Secrets Manager에 시크릿 저장
7. EC2 또는 ECS Fargate Task 생성
8. 앱 배포 (JAR 또는 ECR Docker)
9. application-aws.yml + application-postgresql.yml
10. ALB + Target Group + Health Check
11. CloudWatch 알람 설정

Cloud 배포 완료

문의: accura@accurasoft.co.kr