

# Cloud 기반 고가용성 아키텍처 구축

CI/CD를 통한 본사-지사 자동 배포 서비스

2025.04.14 – 2025.04.22

김혜수 신영민 신지혜

# 팀원 소개

---

김혜수

본사 담당 (EC2 Tomcat)

신영민

Jenkins 배포  
독일 지사 담당 (Docker Tomcat)

신지혜

일본 지사 담당 (Docker Tomcat)



# 목차

---

1

## 개요 및 시스템 구성

프로젝트 목적

서비스 구성 요소

2

## 구축

GitHub

Jenkins 서버 구성 / Jenkins

AWS

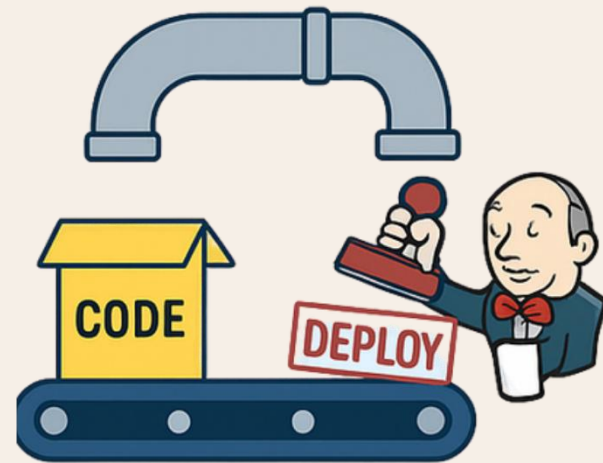
배포 결과

3

## 결론

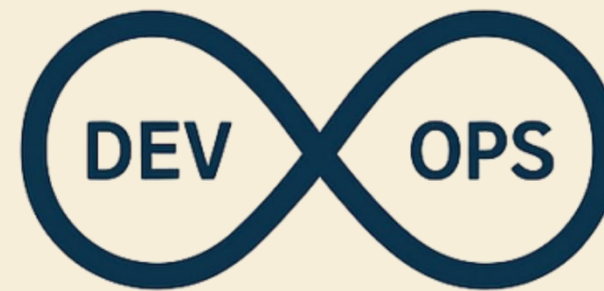
향후 발전 방향

# 개발과 배포 사이의 다리를 만든다



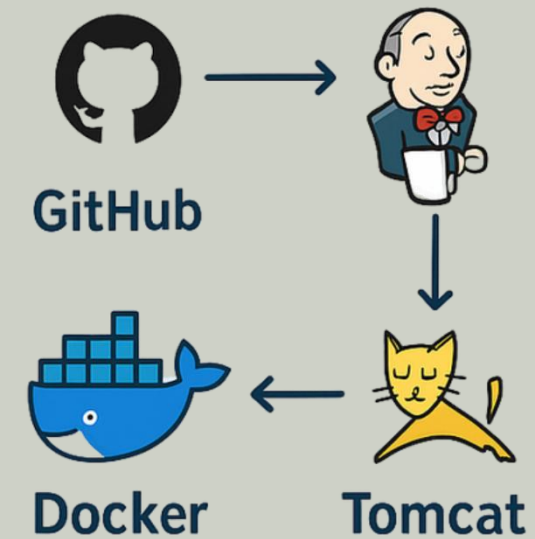
## 배포 자동화 실습

GitHub에 코드를 올리면  
Jenkins가 자동으로 빌드하고,  
Docker로 배포되는  
전체 과정을 직접 실습함



## DevOps 경험 습득

개발과 운영을 연결하는 자동화 흐름을 구성하며,  
DevOps의 핵심 개념을 실제 사례를 통해  
체득함



## CI/CD 파이프라인 구현 경험

지속적인 통합과 배포 환경을 구축하고,  
코드 변경이 곧바로 서비스 반영으로  
이어지는 파이프라인을 완성

## 역할을 나누고, 흐름을 잇다

### Developer

작성한 코드를 GitHub에  
Push하면 자동으로 배포 시작

### GitHub

Webhook을 통해 Jenkins와 연결되어,  
코드 변경을 실시간으로 감지

### Jenkins

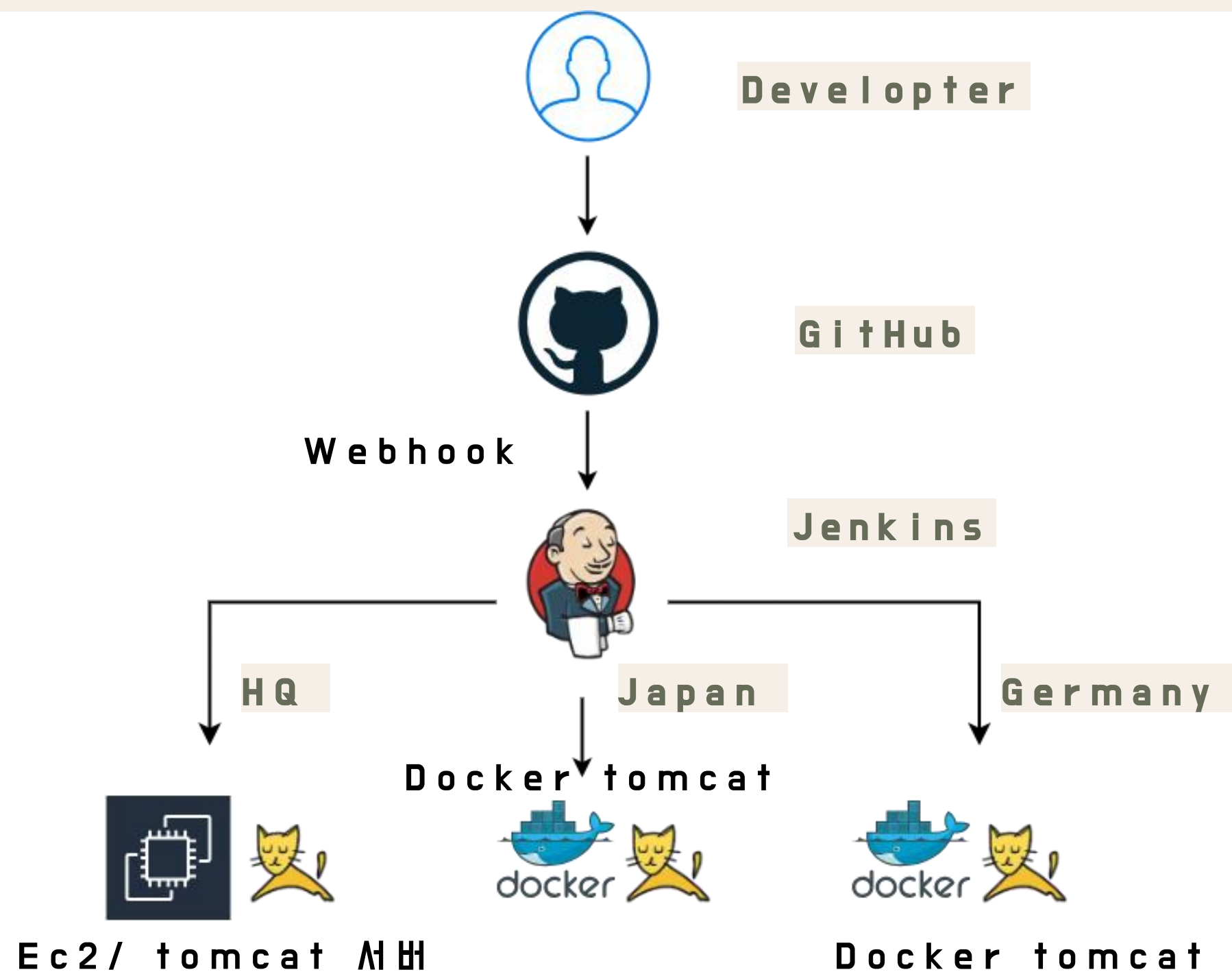
코드를 Build하고 Dockerfile을  
기반으로 이미지를 생성하며,  
컨테이너 실행을 자동화

### Docker

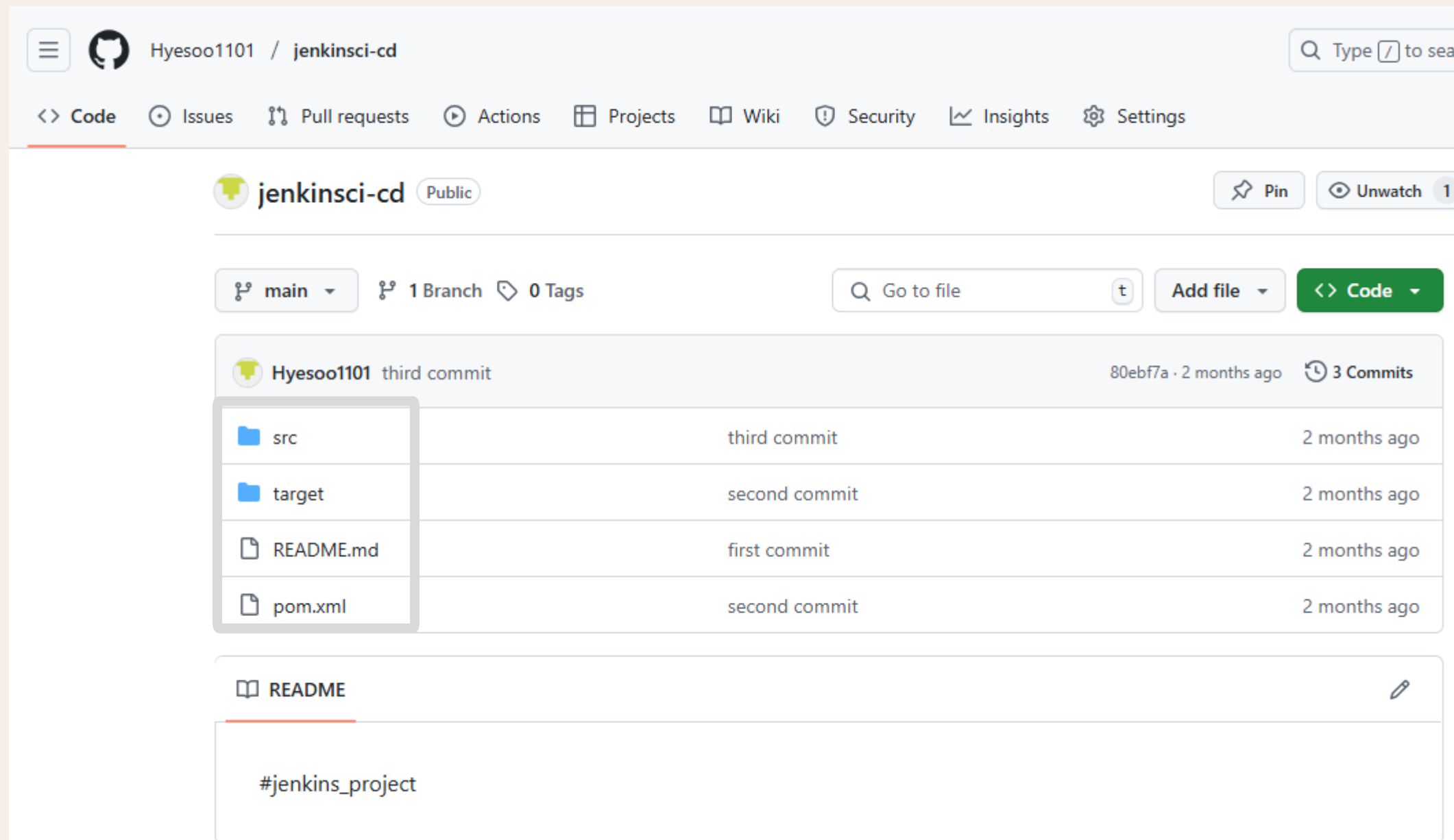
Jenkins에서 생성한 이미지를  
실행하여 Tomcat 서버로 배포

### Tomcat

최종적으로 애플리케이션이  
동작하며 브라우저에서 접근



# CI 파이프라인의 첫 번째 단계



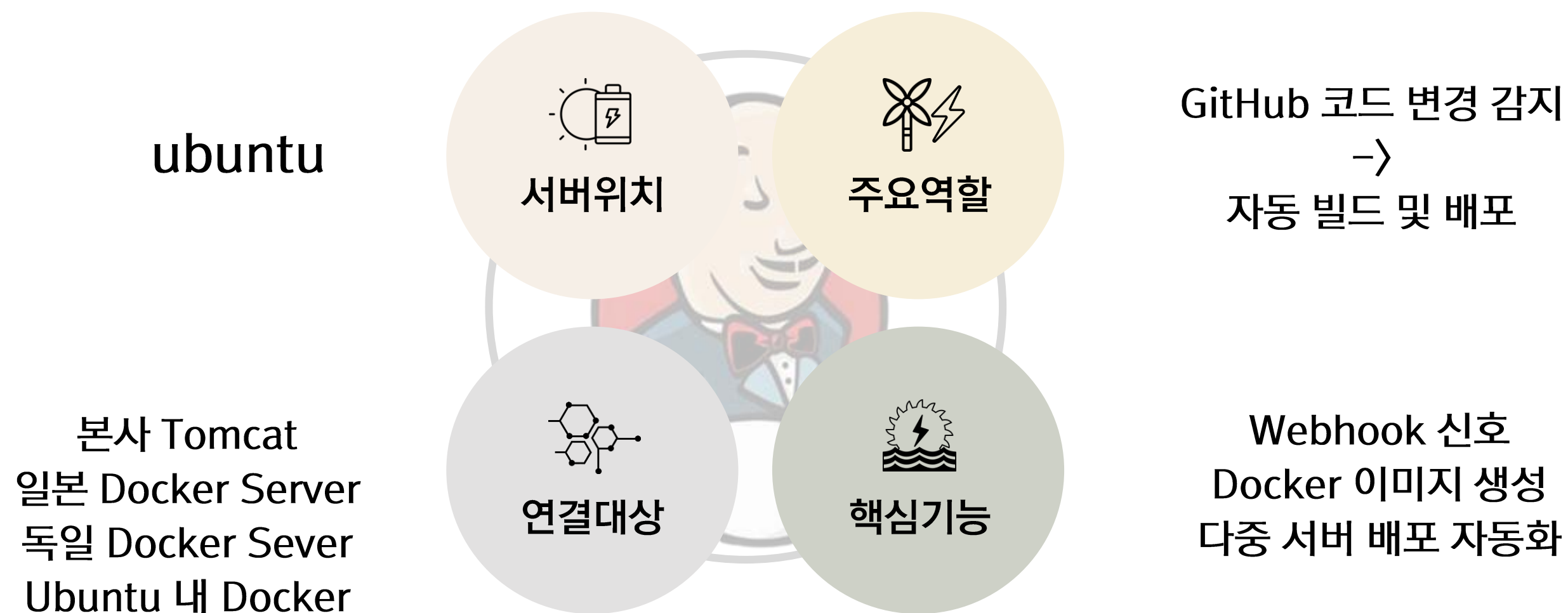
개발자는 로컬에서 작업한 Java 프로젝트를 GitHub 원격 저장소에 Push 하여 소스코드 관리

src/ 디렉토리 및 pom.xml, README.md 등이 커밋되어 있는 상태

Git Hub 저장소는 Jenkins 와 연동되어 있어, Push 이벤트 발생 시 자동으로 Jenkins에서 빌드 시작됨

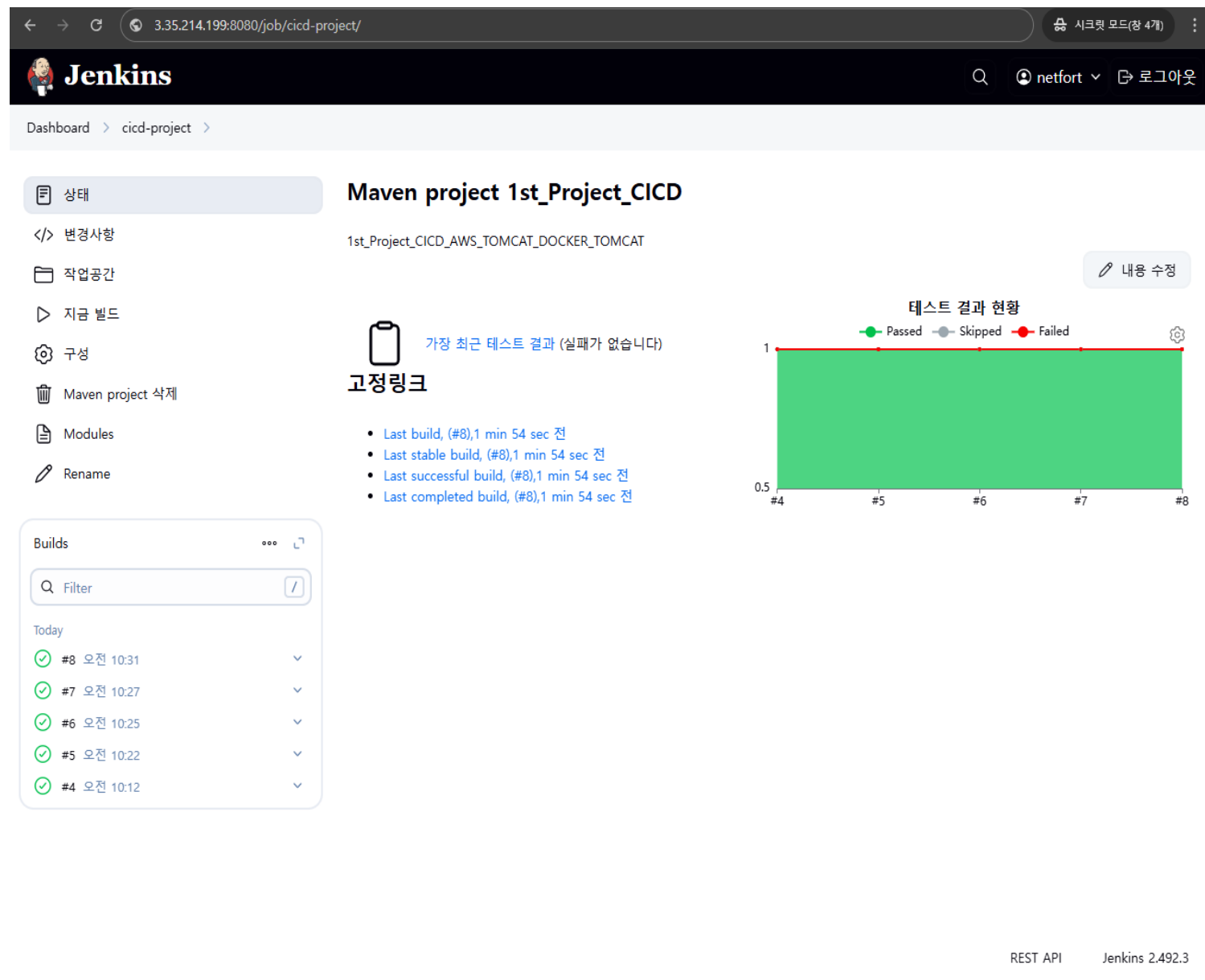
## 자동화의 시작, 빌드의 중심

Jenkins는 전체 자동화 파이프라인의 중심으로, 코드 변경 감지부터 이미지 생성 및 서버 배포까지 모든 단계를 자동으로 수행합니다





# 빌드 자동화, 이젠 Jenkins가 알아서



## Jenkins 자동 빌드 트리거

- 프로젝트명: 1st\_Project\_CICD
- 트리거 방식: GitHub
- 빌드 도구: Maven
- 빌드 성공 횟수: 5회 연속 성공 (#4 ~ #8)
- 테스트 상태: All Passed (성공)
- Jenkins 버전: 2.492.3

GitHub에 소스코드를 Push하면, Jenkins는 GitHub Webhook을 통해 변경 사항을 감지

# AWS 인프라 기반 CI/CD 서버 구성

The screenshot displays the AWS Management Console interface for EC2 instances. The left sidebar shows the navigation menu with '인스턴스' (Instances) selected. The main content area shows a list of three EC2 instances, all of which are in the 'running' state (실행 중). The instances are named 'Japan-Branch', 'HQ', and 'Germany-Branch'. Each instance has a unique ID, is using the 't2.micro' instance type, and has a public IPv4 address assigned. The 'Japan-Branch' instance is highlighted with a red box, 'HQ' with a blue box, and 'Germany-Branch' with an orange box. The table below summarizes the instance details.

Name	인스턴스 ID	인스턴스 상태	인스턴스 유형	상태 검사	경보 상태	가용 영역	퍼블릭 IPv4
Japan-Branch	i-082f8cb8cfdcb1202	실행 중	t2.micro	2/2개 검사 통과	경보 보기	ap-northeast-2c	ec2-13-209.
HQ	i-00e31834f6b74e7c2	실행 중	t2.micro	2/2개 검사 통과	경보 보기	ap-northeast-2c	ec2-52-79-2
Germany-Branch	i-035cdf228248caba2	실행 중	t2.micro	2/2개 검사 통과	경보 보기	ap-northeast-2c	ec2-43-201.

총 3개의 EC2 인스턴스를 통해 분산형 CI/CD 구성  
모든 인스턴스는 퍼블릭 IP를 통해 접근 가능

# AWS 인프라 기반 CI/CD 서버 구성

인바운드 규칙   아웃바운드 규칙   공유 -신규   VPC 연결 -신규   태그						
인바운드 규칙 (5)						
Q 검색						
<input type="checkbox"/>	Name ▾	보안 그룹 규칙 ID ▾	IP 버전 ▾	유형 ▾	프로토콜 ▾	포트 범위 ▾
<input type="checkbox"/>	HQ - tomcat	sgr-0228d198a125f0...	IPv4	사용자 지정 TCP	TCP	8082
<input type="checkbox"/>	Germany-branch	sgr-0ca9f81c0c654f5a1	IPv4	사용자 지정 TCP	TCP	8083
<input type="checkbox"/>	-	sgr-016bca7f3307a6afe	IPv4	SSH	TCP	22
<input type="checkbox"/>	Jenkins	sgr-072c53f95682aea...	IPv4	사용자 지정 TCP	TCP	8080
<input type="checkbox"/>	Japan-Branch	sgr-00aa7cd208ba53...	IPv4	사용자 지정 TCP	TCP	8081

EC2 인스턴스 별 포트 오픈 설정을 통해  
외부에서 각 서버에 접근 가능하도록 구성

- 포트 8080: Jenkins 서버 접근용 (HQ)
- 포트 8081: 일본 지사 Docker Tomcat
- 포트 8082: HQ Tomcat 서버 접근용
- 포트 8083: 독일 지사 Docker Tomcat
- 포트 22: SSH 원격 접속용 (공통)

보안 그룹에서 명시적으로 각 포트를 사용자 지정 TCP로 개방  
운영 환경 전환 시 IP 제한 또는 프라이빗 서브넷 전환 고려

## 결국 확인하는 건 화면 하나



The screenshot shows the Tomcat Web Application Manager interface. The browser address bar indicates the URL is `localhost:8082/manager/html/`. The page title is "Tomcat 웹 애플리케이션 매니저". Below the title, there is a message box showing "메시지: OK". The main content area is titled "매니저" and contains a table of applications.

경로	버전	표시 이름	실행 중	세션들	명령들
<a href="#">/germany</a>	지정 안됨	Germany Branch	true	0	<a href="#">시작</a> <a href="#">중지</a> <a href="#">다시 로드</a> <a href="#">배치된 것을 제거</a> <a href="#">세션들을 만료시키기</a> idle 값 ≥ 30 분
<a href="#">/host-manager</a>	지정 안됨	Tomcat Host Manager Application	true	0	<a href="#">시작</a> <a href="#">중지</a> <a href="#">다시 로드</a> <a href="#">배치된 것을 제거</a> <a href="#">세션들을 만료시키기</a> idle 값 ≥ 30 분
<a href="#">/hq</a>	지정 안됨	HQ App	true	0	<a href="#">시작</a> <a href="#">중지</a> <a href="#">다시 로드</a> <a href="#">배치된 것을 제거</a> <a href="#">세션들을 만료시키기</a> idle 값 ≥ 30 분
<a href="#">/japan</a>	지정 안됨	Japan Branch App	true	0	<a href="#">시작</a> <a href="#">중지</a> <a href="#">다시 로드</a> <a href="#">배치된 것을 제거</a> <a href="#">세션들을 만료시키기</a> idle 값 ≥ 30 분
<a href="#">/manager</a>	지정 안됨	Tomcat Manager Application	true	1	<a href="#">시작</a> <a href="#">중지</a> <a href="#">다시 로드</a> <a href="#">배치된 것을 제거</a> <a href="#">세션들을 만료시키기</a> idle 값 ≥ 30 분

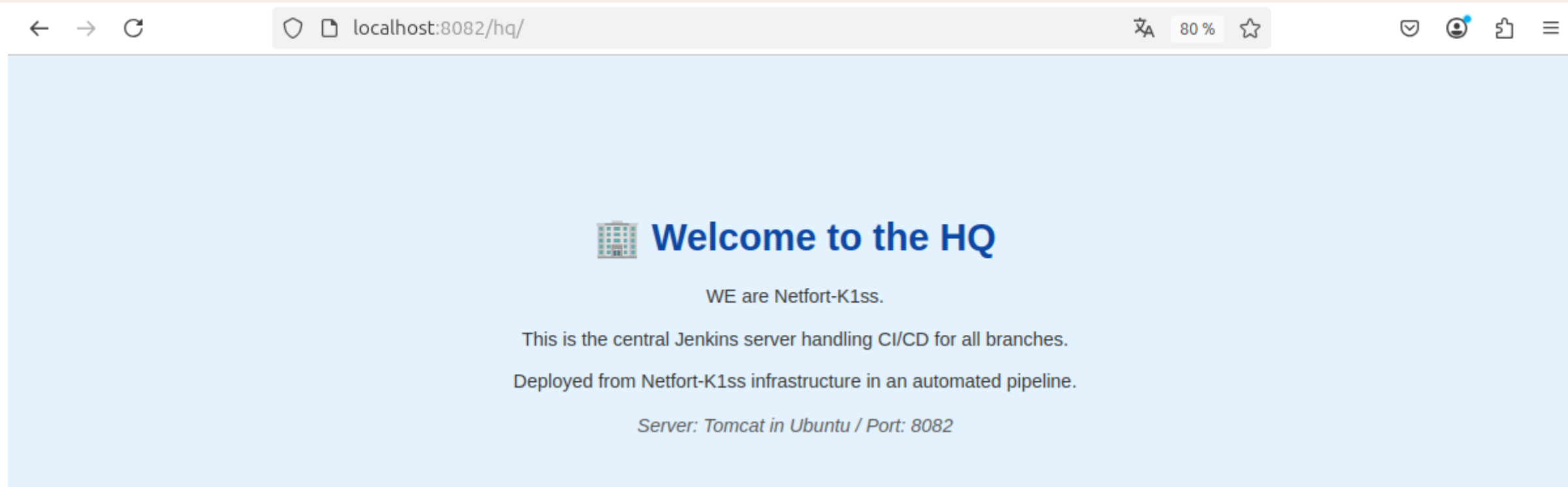
Tomcat 웹 애플리케이션 매니저

Jenkins를 통한 자동 배포 이후,  
WAR 파일이 tomcat에 배포되어  
각 지사 별 애플리케이션이 정상적으로  
기동됨을 시각적으로 확인

- [/hq](#): 본사(HQ) 애플리케이션 작동
- [/germany](#): 독일 지사 서비스 작동
- [/japan](#): 일본 지사 서비스 작동

상태: 모두 true (실행 중)

## 결국 확인하는 건 화면 하나



배포 확인 : HQ (본사) 애플리케이션

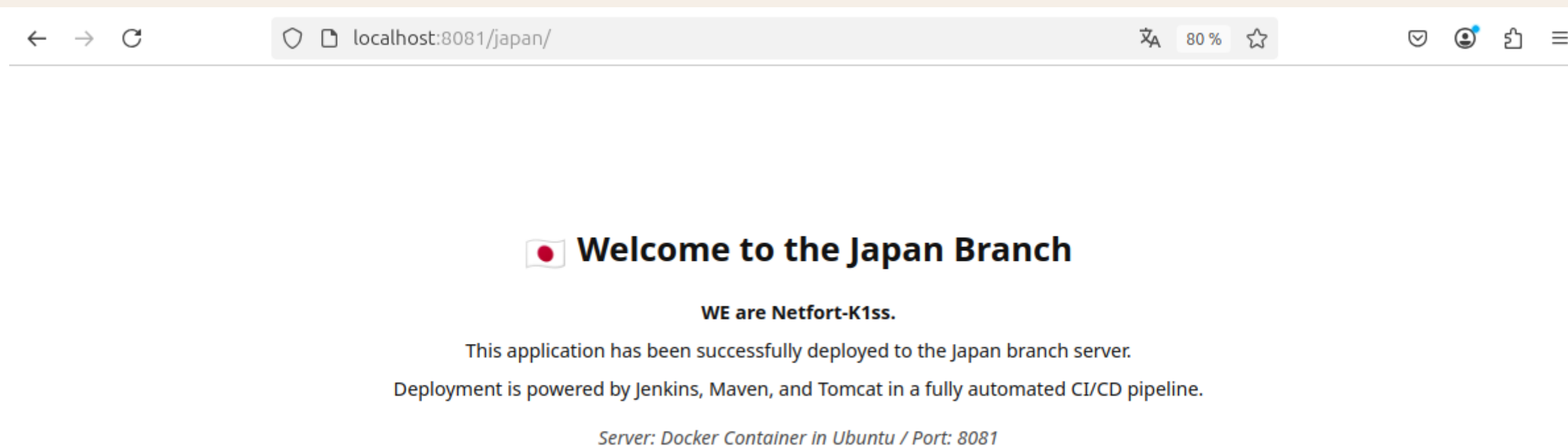
URL : <http://localhost:8082/hq/>

서버 위치 : Tomcat in Ubuntu (내부 테스트용 본사 서버)

상태 : 정상 기동 (자동 배포 완료)

Jenkins Pipeline을 통해 자동 배포된 WAR 파일이 tomcat에 정상적으로 등록되어 실행 중임

## 결국 확인하는 건 화면 하나

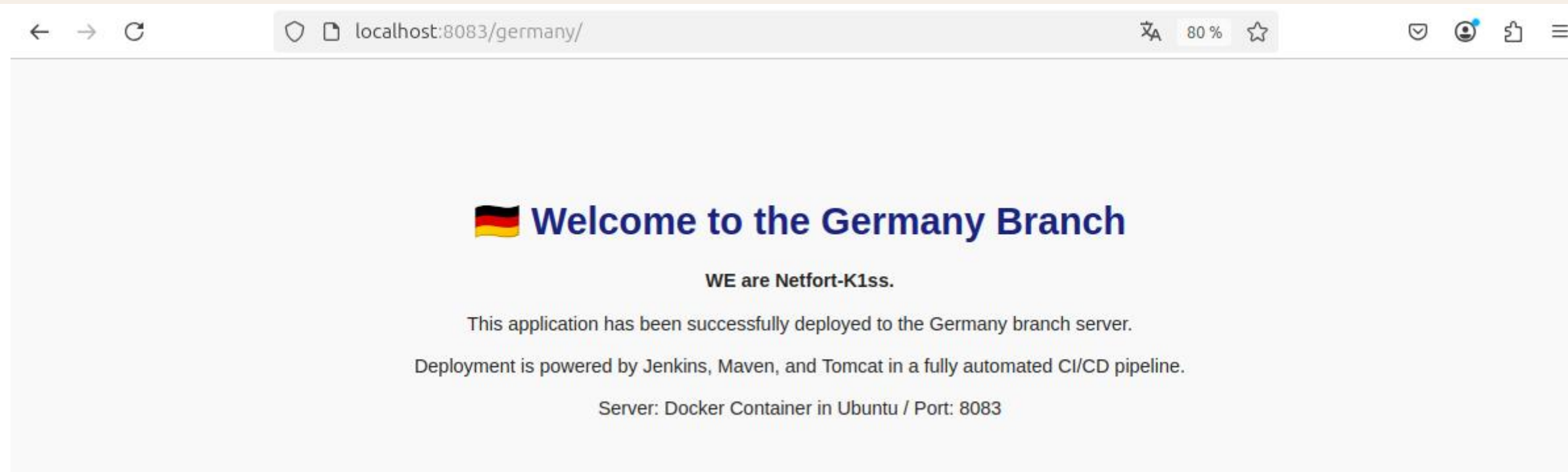


배포 확인 : Japan Branch (일본 지사)  
애플리케이션

URL : <http://localhost:8081/japan/>  
서버 위치 : Ubuntu Docker Container  
상태 : 정상 기동 (자동화 파이프라인을 통해 배포)

Jenkins를 통해 GitHub → Maven 빌드 → Docker 재기동으로 이어지는 자동 배포

## 결국 확인하는 건 화면 하나



배포 확인 : Germany Branch (독일 지사)  
애플리케이션

URL : <http://localhost:8083/germany/>  
서버 위치 : Ubuntu Docker Container  
상태 : 정상 기동 (자동화 파이프라인을 통해 배포)

Jenkins를 통해 GitHub → Maven 빌드 → Docker 재기동으로 이어지는 자동 배포

## 더 강하고, 더 똑똑하게 - DevOps의 다음 챕터

### 보안 강화

- 보안 그룹 세분화
- VPC/VPN 도입
- 인증 토큰 기반 Webhook 보안

### 모니터링 도입

- 서버 상태 자동 수집
- 서버 대시보드화
- 알림 시스템 연동

### 고가용성 강화

- Jenkins 이중화
- 대체 CI 도구 도입

### 테스트 자동화

- 단위 테스트, 통합 테스트 파이프라인 확장

앞으로 보안, 모니터링, 고가용성, 테스트 자동화 측면에서 시스템을 더욱 고도화하며 실제 서비스 운영에 적합한 안정적이고 확장 가능한 CI/CD 환경으로 발전시킬 수 있음



감사합니다