

Jenkins

Jenkins

❖ Jenkins

- ✓ Jenkins는 자바로 작성된 오픈 소스 자동화 서버
- ✓ Apache Tomcat 처럼 servlet container 내부에서 실행되는 서버 시스템
- ✓ 매우 활발한 커뮤니티의 참여와 방대한 플러그인 덕분에 지속적 통합 및 지속적 인도 프로세스를 구축하는 데 많이 사용되는 도구 중 하나
- ✓ Hudson이라고 불렸으나 Oracle이 Hudson을 인수하고 이를 독점 소프트웨어로 전환하기로 하자 이에 반발해 이름을 Jenkins로 변경한 후 지금까지 계속 오픈 소스 솔루션으로 개발을 지속
- ✓ Jenkins는 단순성, 유연성, 다양성 측면에서 매우 높은 역량을 갖고 있으며 지속적 통합 도구로 다른 솔루션에 비해 월등히 뛰어난 기능과 확장성 덕분에 현재 가장 널리 사용되는 소프트웨어



Jenkins

❖ Jenkins

✓ 특징

- 다양한 프로그래밍 언어 지원: Jenkins는 많은 플러그인을 갖고 있으며 대부분의 프로그래밍 언어와 프레임워크를 지원하고 거의 모든 종류의 셸 명령어와 소프트웨어를 사용할 수 있기 때문에 특정 프로그래밍 언어에 대한 지식 없이도 자동화 프로세스를 구축하는 데 지장이 없음(Java, Python, Node.js 등)
- 플러그인을 통한 확장: Jenkins는 훌륭한 커뮤니티를 보유하고 있고 수천 개가 넘는 플러그인을 이용할 수 있고 사용자가 직접 필요한 플러그인을 작성해 Jenkins의 기능을 확장하는 것도 가능(다양한 빌드 도구 지원 - Maven, Ant, Gradle 등)
- 이식성: Jenkins는 Java로 개발됐기 때문에 대부분의 운영체제에서 실행할 수 있음
- WAR 파일이나 Docker Image, 윈도우나 맥OS 또는 리눅스 용 바이너리도 제공하기 때문에 편리
- 대부분의 버전 관리 시스템을 지원: Jenkins는 현존하는 대부분의 소스 코드 관리 및 빌드 도구들과 통합이 가능(Git, SVN 등)
- 분산 처리 지원: Jenkins는 마스터/에이전트 모드를 지원하므로 여러 컴퓨터로 노드를 분산해 실행하는 것이 가능하고 다른 기종 환경에서도 실행되므로 노드마다 다른 운영체제가 설치돼도 문제 없이 사용할 수 있음

Jenkins

❖ Jenkins

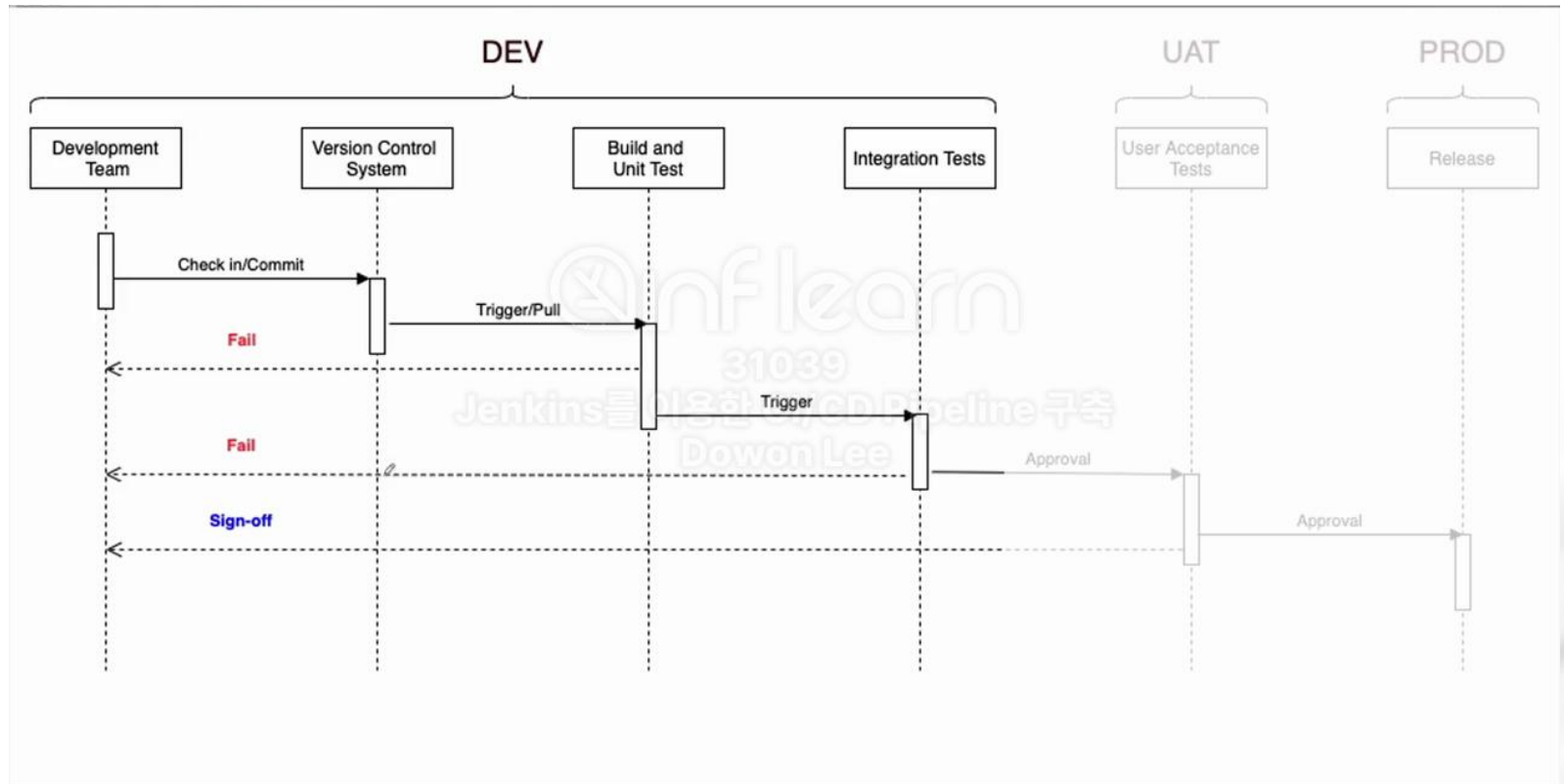
✓ 특징

- 단순성: 설치와 구성 과정이 단순해서 추가 데이터베이스나 소프트웨어를 설치할 필요가 없으며 GUI나 XML, 그루비 스크립트만으로도 구성 작업을 할 수 있음
- 코드 방식 지원: Jenkins 파이프라인은 코드로 저장되며 Jenkins의 구성도 YAML/XML 파일이나 그루비 스크립트로 가능하고 이런 코드를 소스 코드 리포지터리에 저장해 관리할 수 있고 자동화하는 데도 도움이 됨



Jenkins

- ❖ Jenkins
 - ✓ Pipeline



Jenkins

❖ Jenkins

✓ CI/CD 프로세스

- 다수의 개발자가 각자의 브랜치에서 변경 작업을 한 후 이를 중앙 리포지토리로 푸시하고 코드 리뷰가 끝나면 이를 다른 브랜치(개발 브랜치)에 병합
- 브랜치가 변경되었음을 젠킨스에 통보하거나 젠킨스가 인지
- 젠킨스가 통보를 수신하면 작업을 시작

✓ 젠킨스의 작업

- 소스 코드 관리 시스템에 맞는 플러그인을 사용해 리포지토리에서 변경된 파일을 가져옴
- Maven 같은 빌드 도구와 관련 젠킨스 플러그인을 사용해 변경된 파일을 컴파일
- 빌드 도구를 재사용해 컴파일된 코드의 단위/통합 테스트를 실행
- 정적 분석 도구를 실행해 코딩 표준을 준수하는지와 데드 코드가 있는지를 확인하는데 정적 분석 도구로는 소나큐브 등을 사용
- 컴파일과 테스트까지 완료된 파일들을 .jar 또는 .war 같은 라이브러리 형태로 번들링하는데 이 과정 역시 젠킨스 플러그인으로 실행되는 빌드 도구를 사용
- 빌드된 라이브러리 파일을 테스트/프로덕션 환경으로 배포
- E-E 테스트 자동화 도구(셀레늄이나 프로트랙터 등과 같은 UI 자동화 도구)를 사용해 배포된 애플리케이션을 대상으로 테스트를 실행
- 새로 생성된 애플리케이션의 상태와 E-E 테스트 결과가 포함된 이메일을 프로젝트 팀원들에게 전송

Jenkins

❖ Jenkins Architecture

✓ Master & Agent

- Jenkins를 운영하다 보면 예상보다 빠르게 과부하 상황이 발생하는데 용량이 크지 않은 마이크로서비스조차도 빌드하는 데 몇 분씩 걸리기도 하는데 그러다 보니 커밋을 자주 하는 팀은 Jenkins 인스턴스를 죽이는 일이 종종 발생하기 때문에 아주 소규모의 프로젝트가 아니라면 빌드 작업을 에이전트(슬레이브) 인스턴스에 위임해 실행하는 방식을 사용
- 현재 Jenkins를 실행하고 있는 Jenkins 마스터가 있고 실제 작업은 Jenkins 에이전트에 위임해 실행하는 방식을 사용

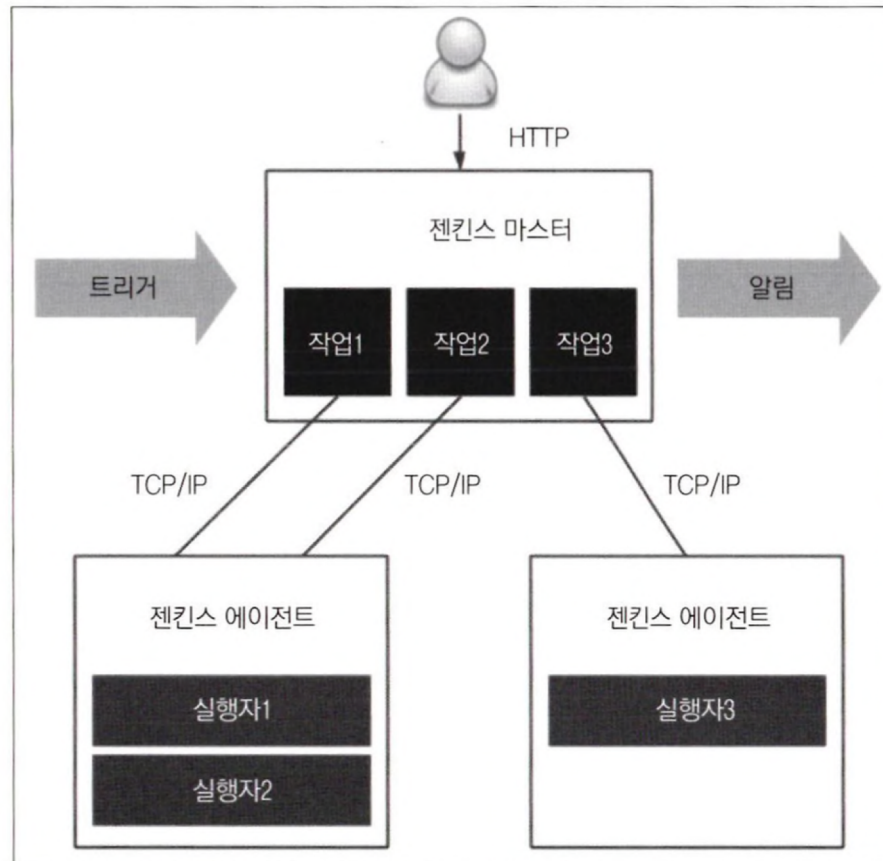


Jenkins

❖ Jenkins Architecture

✓ Master & Agent

● 마스터-에이전트의 동작 방식



Jenkins

❖ Jenkins Architecture

✓ Master & Agent

● Jenkins 마스터의 작업

- ◆ 빌드 시작 명령을 받음 - Github에 커밋이 발생한 직후
- ◆ 알림을 전송 - 빌드가 실패하면 이메일이나 슬랙 메시지로 알림
- ◆ 클라이언트와 통신을 하며 HTTP 요청을 처리
- ◆ 에이전트에서 실행 중인 작업의 우선순위 조정 등의 빌드 환경을 관리

● 빌드 에이전트

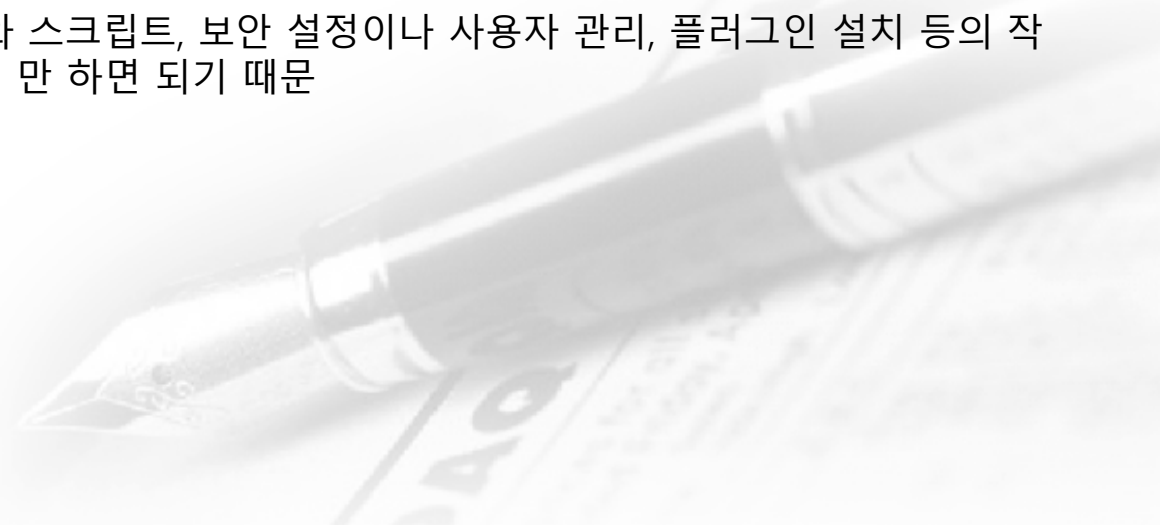
- ◆ 빌드가 시작된 이후에 발생하는 모든 작업을 처리하는 기기
- ◆ 에이전트는 가능한 한 범용성을 갖춘 기기여야 하는데 예를 들어 하나는 자바로 다른 하나는 파이썬으로 또 다른 하나는 루비로 작성된 프로젝트가 있다고 할 때 각 에이전트들에게 어떤 프로젝트를 할당해도 잘 수행할 수 있어야 하는데 이런 식으로 에이전트를 교차 사용할 수 있다면 가용 자원을 최적화하기가 수월
- ◆ 에이전트 머신이 모든 프로젝트에 대응할 만큼 충분하지 않다면 에이전트와 프로젝트에 레이블(태그)을 붙여 특정 빌드는 특정 유형의 에이전트에서 수행하도록 하는 방식도 가능

Jenkins

❖ Jenkins Architecture

✓ 확장성

- 소프트웨어 분야의 모든 것이 그렇듯이 Jenkins도 사용량 증가에 따라 급격히 과부하 상태가 되고 수행 속도도 떨어지므로 이런 문제가 발생하기 전에 미리 확장(Scaling) 계획을 세울 필요가 있으며 수직적 확장 과 수평적 확장 2가지 방식 있음
- 수직적 확장
 - ◆ 수직적 확장은 마스터의 부하 증가에 대응해 마스터 시스템에 자원을 추가하는 방식
 - ◆ 신규 프로젝트가 생길 때마다 램과 CPU를 추가하고 외장 하드 드라이브도 확장 하는 것
 - ◆ Jenkins 마스터를 단 한 대의 초 고성능의 하드웨어에서 운용한다면 유지보수 측면에서는 매우 효율적
 - ◆ 모든 업그레이드와 스크립트, 보안 설정이나 사용자 관리, 플러그인 설치 등의 작업을 단 한 곳에서 만 하면 되기 때문



Jenkins

❖ Jenkins Architecture

✓ 확장성

● 수평적 확장

- ◆ 수평적 확장은 조직이 늘어날 때마다 마스터 인스턴스의 수량을 늘려가는 방식으로 영리하게 인스턴스를 할당할 필요가 있으며 극단적으로는 각 팀마다 한 대씩의 마스터를 두게 될 수도 있는데 그렇게 된다면 아예 에이전트가 필요하지 않을 수도 있음
- ◆ 단점은 프로젝트 간의 통합 자동화가 어렵다는 것과 각 팀마다 Jenkins를 유지 보수하는 시간이 필요하다는 것
- ◆ 장점
 - 마스터 역할을 하는 컴퓨터의 하드웨어가 고 사양일 필요가 없음
 - 팀마다 각기 다른 설정을 할 수 있음(팀마다 다르게 플러그인 설치 가능)
 - 팀 전용 마스터 인스턴스가 있으므로 팀워크와 업무 효율이 향상
 - 마스터 인스턴스 하나에 문제가 생겨도 다른 팀에 영향을 끼치지 않음
 - 인프라를 표준 인프라와 필수 인프라로 구분해 구성할 수 있음

Jenkins

❖ Jenkins Architecture

✓ 테스트 인스턴스/프로덕션 인스턴스

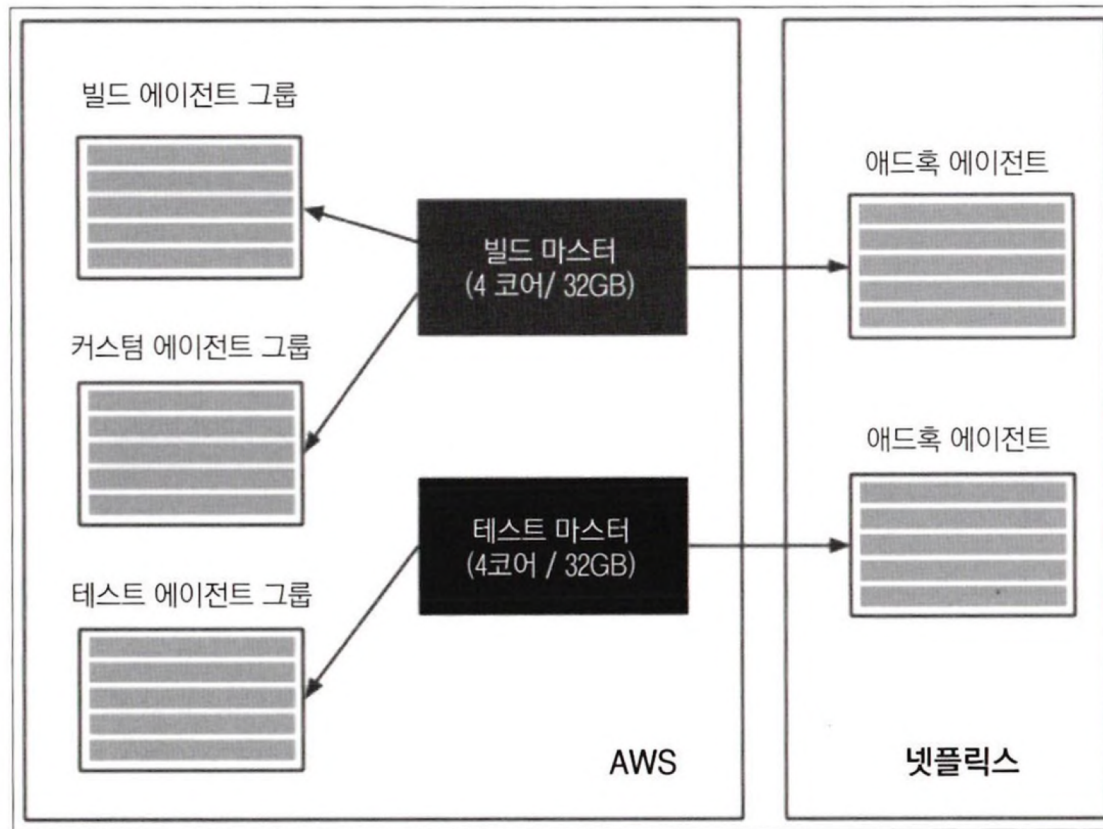
- 확장 방식 외에 고려할 점이 하나 더 있는데 Jenkins 업그레이드와 신규 플러그인, 파이프라인 정의를 어떻게 테스트
- Jenkins 시스템은 전사적으로 매우 중요한데 소프트웨어의 품질을 보증하며 지속적 인 도 프로세스에서는 프로덕션 서버에 배포하는 역할을 수행하는데 고 가용성이 매우 중요한 시스템으로 결코 테스트 목적으로 사용해서는 안됨
- Jenkins는 항상 동일한 두 개의 인스턴스를 테스트 용과 프로덕션 용으로 분리 운영
- 테스트 환경은 최대한 프로덕션 환경과 유사해야 하므로 연결된 에이전트의 개수도 비슷하게 맞춰야 함



Jenkins

❖ Jenkins Architecture

✓ 넷플릭스 아키텍처



Jenkins

❖ Jenkins Architecture

✓ 넷플릭스 아키텍처

- 넷플릭스는 테스트 용과 프로덕션 용 마스터 인스턴스를 운영
- 각 인스턴스에는 에이전트 풀 과 추가 애드혹 에이전트들이 존재
- 이 인프라에서는 하루에 2천 개의 빌드를 할 수 있음
- 인프라의 일부는 AWS에서 운영되며 일부는 자체 서버에서 운영



Jenkins

❖ Jenkins 설치

✓ 하드웨어 요구 사항

- 256MB의 메모리 용량
- 1GB 이상의 하드 디스크 용량 이상을 추천(Jenkins를 도커 컨테이너로 실행하는 경우에는 10GB 이상을 추천)

✓ 소프트웨어 요구 사항

- Java(요구되는 버전: <https://get.jenkins.io/war-stable/>)



Jenkins

❖ Jenkins 설치

✓ 설치 방법: <https://www.jenkins.io/doc/book/installing/>

- Servlet: Jenkins는 Java로 작성됐고 기본적으로 WAR 형식의 웹 애플리케이션으로 배포되는 애플리케이션 서버(아파치 톰캣 또는 글래스피시) 전용 솔루션이므로 모든 애플리케이션을 Servlet으로 배포하는 경우 이 방식을 선택
- Application: Jenkins WAR 파일은 Jetty 애플리케이션 서버를 포함하므로 Java 명령으로 직접 실행할 수 있는데 이때 필요한 것은 JRE 뿐으로 베어메탈 서버를 사용하거나 한 시스템에 여러 Jenkin 인스턴스를 설치해야 하는 경우 이 방식을 선택
- 전용 패키지: Jenkins는 전용 패키지(윈도우용 MSI, 맥OS용 Homebrew 패키지, 데비안/우분투용 deb 패키지 등) 형태로 대부분의 운영체제에 맞는 버전을 배포하는데 베어메탈 서버를 사용하는 경우 가장 간단하게 설치 및 구성을 하려면 이 방식을 선택
- Docker: Jenkins는 Docker 이미지 형태로도 배포되므로 Docker 실행 환경을 갖고 있다면 가장 간단한 설치 방식
- Kubernetes: Jenkins는 Kubernetes 클러스터에서 설치, 관리 및 확장을 단순화하는 데 헬름 차트와 Kubernetes 오퍼레이터를 제공하는데 가장 간단한 Jenkins 확장 및 관리를 위해서는 이 방식을 선택
- Cloud: Jenkins는 여러 플랫폼에서 SaaS 형식으로 호스팅 되므로 서버 유지관리 및 Jenkins 설치에 대해 고민하고 싶지 않다면 이 방식을 선택

Jenkins

❖ Jenkins 설치

✓ 설치 방식 비교

비교 항목	Docker 컨테이너 설치 (권장)	Linux 직접 설치 (Bare Metal / VM 패키지)
설치 및 초기 구성	간편함 (공식 이미지 및 docker-compose로 즉시 가동)	Java(JDK), 환경 변수, OS별 리포지토리 수동 설정 필요
Java 및 의존성 격리	컨테이너 내부 JVM 버전에 종속되므로 호스트 OS 오염 없음	호스트 OS의 Java 버전 업그레이드 시 충돌 위험
업데이트 및 롤백	이미지 태그 변경 및 컨테이너 재기동으로 즉시 완료	패키지(apk/yum) 업데이트 시 실패하거나 롤백이 복잡함
백업 및 서버 이전	데이터 볼륨(/var/jenkins_home)만 백업/마운트하면 끝	시스템 설정, 서비스 파일, 홈 디렉터리 등 분산 백업 필요
빌드 격리 (CI 실행 환경)	파이프라인에서 컨테이너 기반 빌드 환경 구성 용이	빌드 툴(Node, Python, Go 등)이 호스트 OS에 누적되어 충돌 유발

Jenkins

❖ Jenkins 설치

- ✓ docker 환경 – docker-compose.yml
version: '3.8'

services:

jenkins:

image: jenkins/jenkins:lts-jdk21

container_name: jenkins

restart: unless-stopped

user: root # 호스트 docker.sock 접근 권한 처리가 필요할 경우

ports:

- "8080:8080" # 웹 UI 포트
- "50000:50000" # Jenkins Agent 연결 포트

volumes:

- ./jenkins_home:/var/jenkins_home
- /var/run/docker.sock:/var/run/docker.sock

Jenkins

❖ Jenkins 설치

✓ docker 환경

● 초기 비밀번호 확인

◆ 로그 확인

▪ docker logs jenkins

```
*****  
*****
```

Jenkins initial setup is required. An admin user has been created and a password generated.

Please use the following password to proceed to installation:

654d91092df9404c8ae3d8ee48f40a9c

This may also be found at: /var/jenkins_home/secrets/initialAdminPassword

```
*****  
*****  
*****
```

Jenkins

❖ Jenkins 설치

✓ docker 환경

● 초기 비밀번호 확인

◆ 내부 파일을 이용해서 확인

- `docker exec -it jenkins /bin/bash`
- `cd /var/jenkins_home/secrets/`
- `cat initialAdminPassword`



Jenkins

❖ Jenkins 설치

✓ 전용 패키지로 설치

- ubuntu – <https://www.Jenkins.io/doc/book/installing/linux/#debianubuntu>

◆ JDK 설치

- sudo apt update
- sudo apt install fontconfig openjdk-21-jre
- java -version

◆ sudo wget -O /etc/apt/keyrings/jenkins-keyring.asc ₩

<https://pkg.jenkins.io/debian/jenkins.io-2026.key>

◆ echo "deb [signed-by=/etc/apt/keyrings/jenkins-keyring.asc]" ₩

<https://pkg.jenkins.io/debian/binary/> | sudo tee ₩

/etc/apt/sources.list.d/jenkins.list > /dev/null

◆ sudo apt update

◆ sudo apt install Jenkins

◆ 브라우저에서 8080 포트로 접속하면 초기 비밀번호 파일의 경로가 보임

Jenkins

❖ Jenkins 설치

✓ docker 환경

- 추가 설정 – localhost:8080 으로 접속해서 설정
 - ◆ 비밀번호는 docker logs Jenkins 의 로그에서 확인

Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

```
/var/jenkins_home/secrets/initialAdminPassword
```

Please copy the password from either location and paste it below.

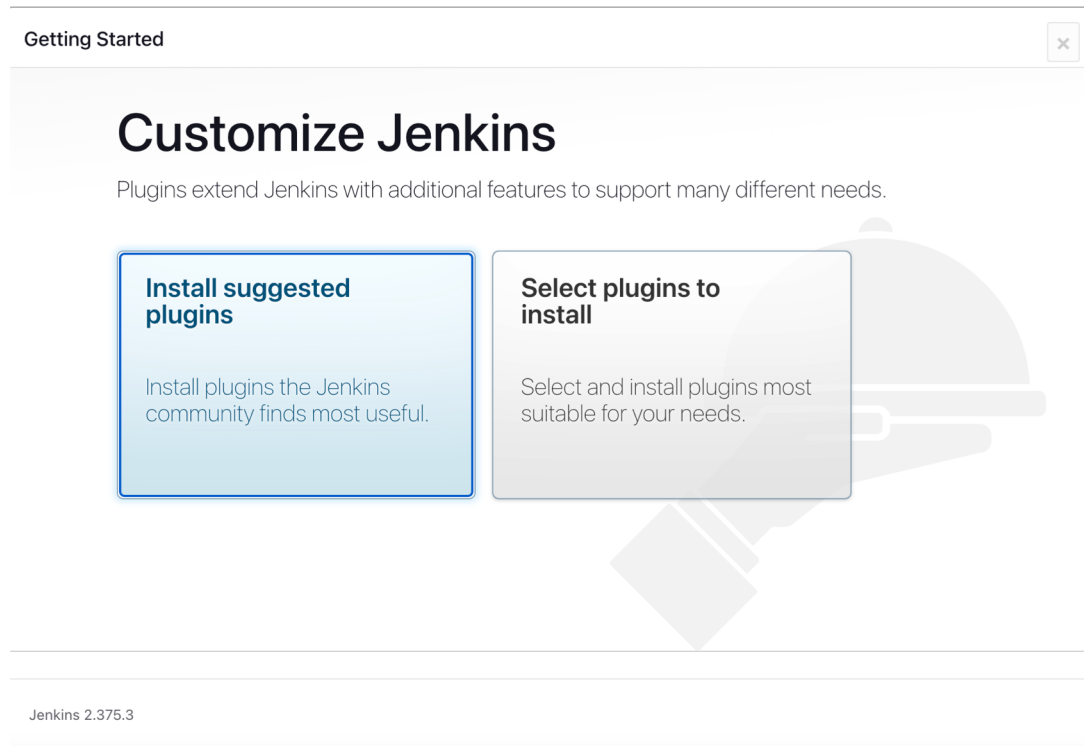
Administrator password

Continue

Jenkins

❖ Jenkins 설치

- ✓ 추가 설정 – localhost:8080 으로 접속해서 설정
 - 플러그 인 설치



Jenkins

❖ Jenkins 설치

✓ 추가 설정 – localhost:8080 으로 접속해서 설정

- 계정 생성

Getting Started

Create First Admin User

계정명

암호

암호 확인

이름

Jenkins 2.375.3

[Skip and continue as admin](#) [Save and Continue](#)

Jenkins

❖ Jenkins 설치

- ✓ 추가 설정 – localhost:8080 으로 접속해서 설정
 - URL 확인

Getting Started

Instance Configuration

Jenkins URL:

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the `BUILD_URL` environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Jenkins 2.375.3

[Not now](#) [Save and Finish](#)

Jenkins

❖ 초기 접속 화면

 **Jenkins**

🔍  1  itstudy  로그아웃

Dashboard >

+ 새로운 Item

 빌드 기록

 Jenkins 관리

 My Views

빌드 대기 목록 

빌드 대기 항목이 없습니다.

빌드 실행 상태 0/2 

상세 내용 입력

Jenkins에 오신 것을 환영합니다.

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

Create a job 

Set up a distributed build

Set up an agent 

Configure a cloud 

Learn more about distributed builds 



Jenkins

❖ 플러그인 관리

✓ 개요

- Jenkins는 소프트웨어 개발 프로세스의 여러 단계를 자동화하는 도구로 Source Code Repository에서 Checkout, Compile, 단위 테스트 실행, 산출물 패키징 및 배포 등의 기능을 제공
- 무료로 자유롭게 사용할 수 있는 라이선스를 갖는 오픈 소스 소프트웨어
- 소프트웨어 빌드 수명 주기는 소스 관리 도구에서 소스 코드를 입수하고 컴파일하고 단위/통합 테스트를 수행하고 라이브러리를 빌드하고 배포하는 단계로 구성되는데 이러한 단계들을 수행하려면 SCM, 단위/통합 테스트 도구, 빌드 도구 등 다양한 종류의 도구를 사용해야 함
- Jenkins는 이 모든 도구와 인터페이스하면서 빌드 수명 주기의 처음부터 끝까지의 과정을 실행하는데 이때 사용되는 것이 Jenkins 플러그인



Jenkins

❖ 플러그인 관리

✓ 많이 사용되는 플러그인

● Git

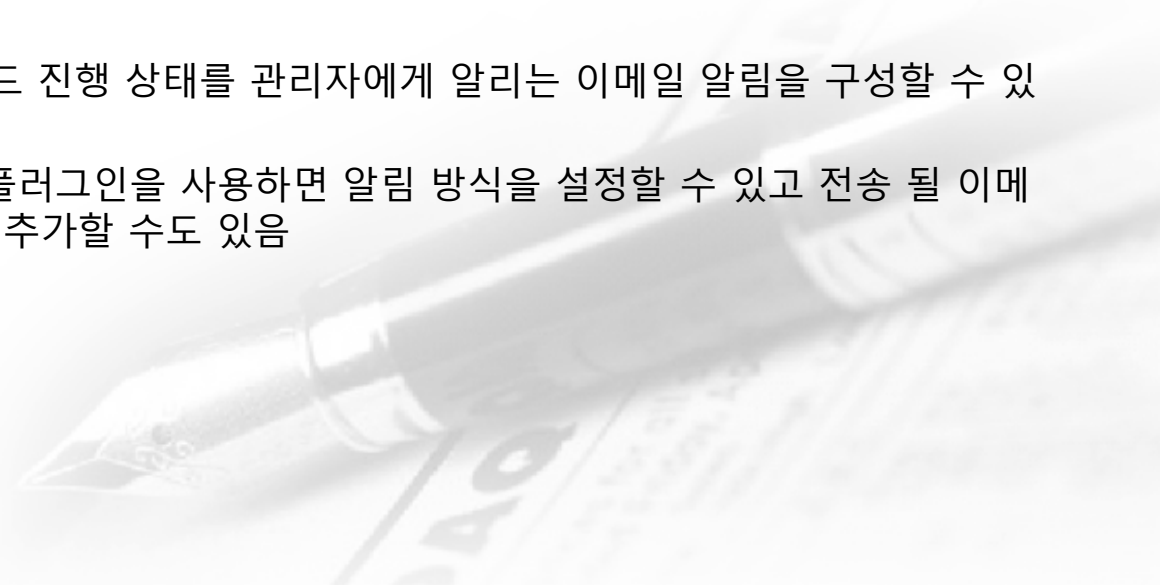
- ◆ Git 플러그인은 깃 버전 관리 시스템과 통합에 사용
- ◆ Git은 분산형 버전 관리 시스템으로써 여러 개발자가 함께 소스 코드를 관리할 수 있는 기능을 제공

● Maven Integration

- ◆ Maven Integration 플러그인은 메이븐 빌드 도구와 Jenkins를 통합하는 데 사용
- ◆ Maven은 컴파일, 패키징, 테스트 등과 같은 핵심 빌드 단계를 자동화하는 데 사용되는 빌드 도구

● Email Extension Plugin

- ◆ Jenkins에서는 빌드 진행 상태를 관리자에게 알리는 이메일 알림을 구성할 수 있음
- ◆ Email Extension 플러그인을 사용하면 알림 방식을 설정할 수 있고 전송 될 이메일의 세부 정보를 추가할 수도 있음

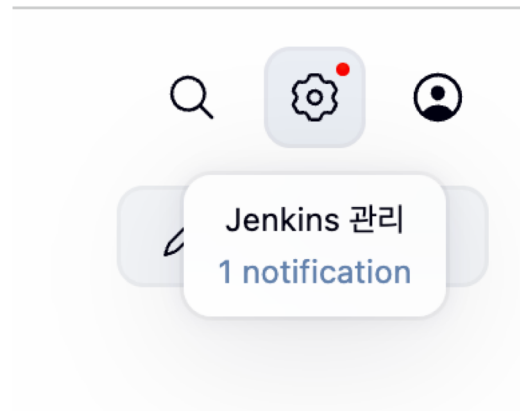


Jenkins

❖ 플러그인 관리

✓ 플러그인 설치

- Jenkins에 로그인해서 대시보드 페이지 확인
- Dashboard 에서 Manage Jenkins 를 클릭

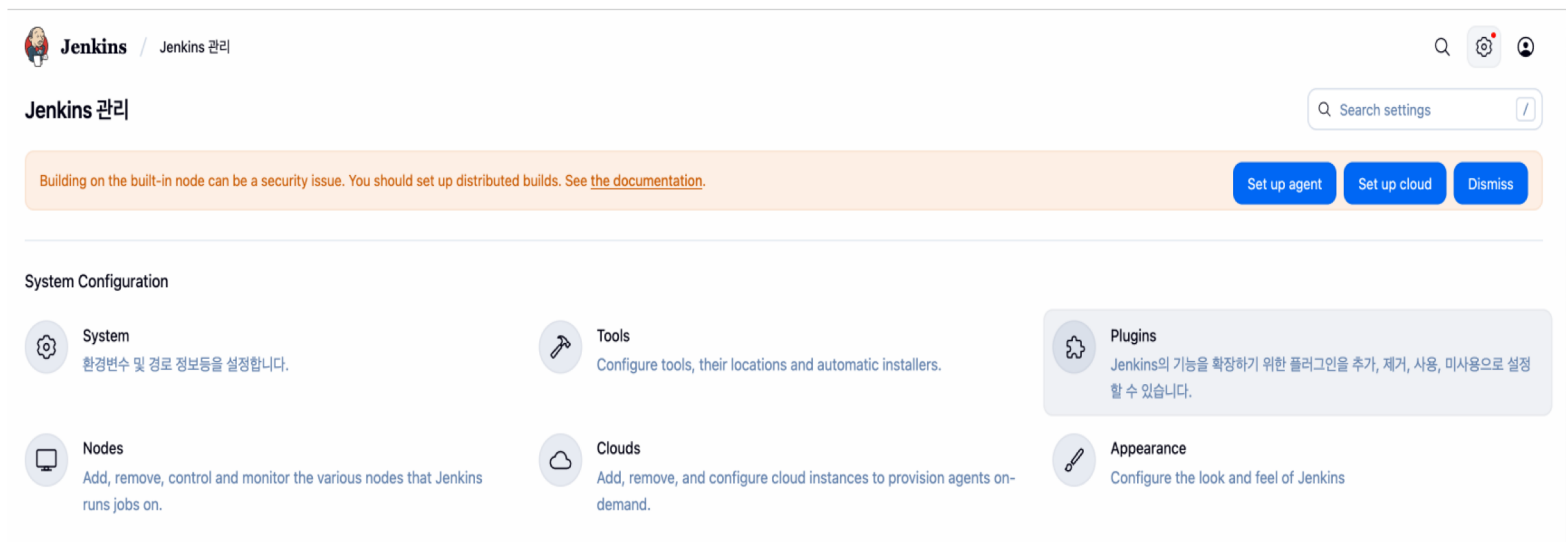


Jenkins

❖ 플러그인 관리

✓ 플러그인 설치

● Plugins 클릭



Jenkins / Jenkins 관리

Jenkins 관리

Building on the built-in node can be a security issue. You should set up distributed builds. See [the documentation](#).

Set up agent Set up cloud Dismiss

System Configuration

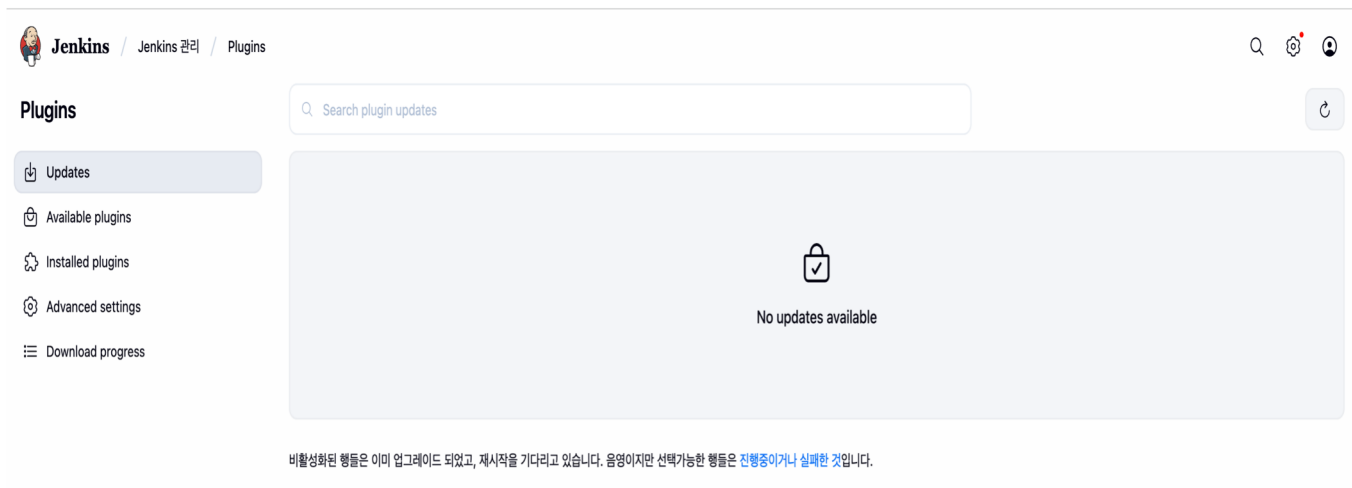
- System**
환경변수 및 경로 정보등을 설정합니다.
- Tools**
Configure tools, their locations and automatic installers.
- Plugins**
Jenkins의 기능을 확장하기 위한 플러그인을 추가, 제거, 사용, 미사용으로 설정할 수 있습니다.
- Nodes**
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.
- Clouds**
Add, remove, and configure cloud instances to provision agents on-demand.
- Appearance**
Configure the look and feel of Jenkins

Jenkins

❖ 플러그인 관리

✓ 플러그인 설치

● Plugins 클릭



Jenkins

❖ 플러그인 관리

✓ 플러그인 설치

● Plugin Manager 항목

◆ Updates

- 설치된 플러그인 중 업데이트 가능한 플러그인을 표시
- 업데이트하고 싶은 플러그인 이름 앞에 체크박스를 선택한 후 Down now and install after restart 버튼을 클릭해 업데이트 버전을 설치할 수 있음

◆ Available plugins

- 다운로드 및 설치할 수 있는 플러그인 표시
- 새로운 플러그인을 설치하려면 검색 창에 플러그인 이름을 입력한 후 나타나는 목록에서 이름 앞에 체크박스를 선택

◆ Installed plugins

- Jenkins에 설치된 모든 플러그인의 목록과 상세 버전 정보 등이 표시되며 플러그인을 제거할 수 있는데 검색 창에 이름을 입력하거나 직접 목록에서 해당 플러그인을 찾아 Uninstall 버튼을 클릭
- JENKINS_HOME/plugins 디렉터리에서 제거를 원하는 플러그인의 .HPI 파일을 삭제하는 방법도 있음
- 플러그인을 제거하지 않고 비활성화하는 방법도 있는데 이 경우 비록 플러그인이 설치돼 있어도 시작되지 않으며 확장 기능도 제공되지 않는데 플러그인을 비활성화하려면 Enabled을 클릭해서 비활성화 상태로 변경

Jenkins

❖ 플러그인 관리

✓ 플러그인 설치

● Plugin Manager 항목

◆ Advanced Settings

- 기업이나 조직에 따라서는 프록시 서버를 거쳐야만 인터넷을 사용할 수 있는 경우가 있는데 프록시 서버가 구성되지 않으면 인터넷에 직접 접속할 수 없음
- 프록시 서버를 거쳐야만 인터넷 접속 요청이 실제 네트워크로 연결되는 구성이기 때문인데 조직 내에 이런 프록시 서버가 있다면 Jenkins에서 수행하는 작업들이 인터넷에 접속할 수 없음
- Jenkins 서버가 조직의 프록시 서버 뒤에 위치한다면 먼저 Jenkins에서 프록시 구성을 해야 하는데 이 때 사용



Jenkins

❖ 플러그인 관리

✓ 설치 문제 해결

● 오류 메시지

java.io.IOException: Downloaded file /var/lib/jenkins/plugins/*.tmp does not match expected SHA-1, expected 'f2ncNlydUUSPrk6SoG255v+2kQU=', actual '!ZR3co4Ouvlj0AG4Aet7HadHg/Q='

◆ 에러 이유

- Jenkins 업데이트 센터가 Jenkins와 동기화되지 않는 문제
 - Manage Jenkins > Manage Plugin을 선택한 후 Advanced settings 메뉴를 클릭하고 페이지 아래 부분에 Update Site 섹션으로 이동하고
 - Check Now 즉시 점검 버튼을 클릭하면 Jenkins 서버와 업데이트 센터가 동기화 됨
 - Jenkins 서버를 재시작하고 플러그인을 다시 설치
- Jenkins 시스템이 프록시 뒤에서 운영되는 문제
 - Manage Jenkins > Manage Plughr을 선택한 후 Advanced settings 메뉴를 클릭하고 HTTP Proxy Configuration 섹션에 인증이 설정돼 있지 않으면 Username과 Password 항목을 비움
 - Advanced... 버튼을 클릭하면 나타나는 Test URL 항목에 Jenkins 업데이트 센터의 URL을 입력한 후 Validata Proxy 버튼을 클릭하고 프록시 검증이 성공하면 Submit 버튼을 클릭하고 다시 설치

Jenkins

❖ 플러그인 관리

✓ 설치 문제 해결

● 오류 메시지

javax.net.ssl.SSLHandshakeException: PKIX path building failed:

sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to requested target

◆ 에러 이유

- Jenkins 업데이트 센터 URL에서는 HTTPS를 기본으로 사용하기 때문에 보안 인증서와 같은 추가 보안 조치가 필요
 - Manage Jenkins > Manage Plugin을 선택하고 Advanced settings 메뉴를 선택한 후 페이지 아래에 Update Site 섹션으로 이동해서 기존 주소의 https를 http로 수정한 후 재설치
 - 최신 버전의 자바를 다운로드하고 설치한 후 PATH 환경 변수에 JAVA_HOME/bin 디렉토리를 추가

Jenkins

❖ Jenkins 작업의 이해

✓ Jenkins의 작업

- Jenkins의 작업은 Jenkins에게 무엇을 언제 해야 하는지를 지시하는 일련의 명령 집합
- 작업은 다른 말로 Jenkins 프로젝트라고 할 수도 있음
- 어떤 종류의 작업을 구성하든 다음 세 가지 유형의 지시 사항을 포함해야 함
 - ◆ 작업을 수행하는 시점(트리거): 사용자는 작업에서 수행할 태스크가 언제 시작될 지를 Jenkins에게 지시할 수 있는데 이것을 Jenkins에서는 Trigger 라고 부름
 - ◆ 작업을 구성하는 단계별 태스크(빌드 스텝): 사용자는 특정 목표를 수행하기 위한 태스크를 단계별(Step)로 구성할 수 있으며 이것을 Jenkins에서는 Build Step 이라고 부름
 - ◆ 태스크가 완료 후 수행 할 명령(Post-Build Action): 사용자는 태스크 실행이 완료 된 후에 Jenkins가 수행할 작업을 구성할 수 있는데 이것을 Jenkins에서는 Post-Build Action 이라고 부름

✓ Jenkins의 빌드

- Jenkins의 빌드는 Jenkins 작업의 특정 실행 버전
- 사용자는 Jenkins 작업을 여러 번 실행할 수 있는데 실행될 때마다 고유한 빌드 번호가 부여됨
- 작업 실행 중에 생성된 아티팩트, 콘솔 로그 등 특정 실행 버전과 관련된 모든 세부 정보가 해당 빌드 번호로 저장됨

Jenkins

❖ Jenkins 작업의 이해

✓ 작업 종류

- Jenkins에서는 필요에 따라 Pipeline Job 이나 Freestyle Job 등과 같은 다양한 유형의 작업을 만들 수 있음
- 프리스타일 작업은 일반적인 형태의 빌드 작업(또는 태스크) 이라 할 수 있는데 이를 통해 일반적으로 테스트 실행, 빌드 및 패키징, 보고서 전송 같은 간단한 작업을 할 수 있음

