

# Jenkins 소개



# Jenkins

<https://www.jenkins.io/>

## CNCF CI/CD 종류

<https://landscape.cncf.io/card-mode?category=continuous-integration-delivery&grouping=category>

---

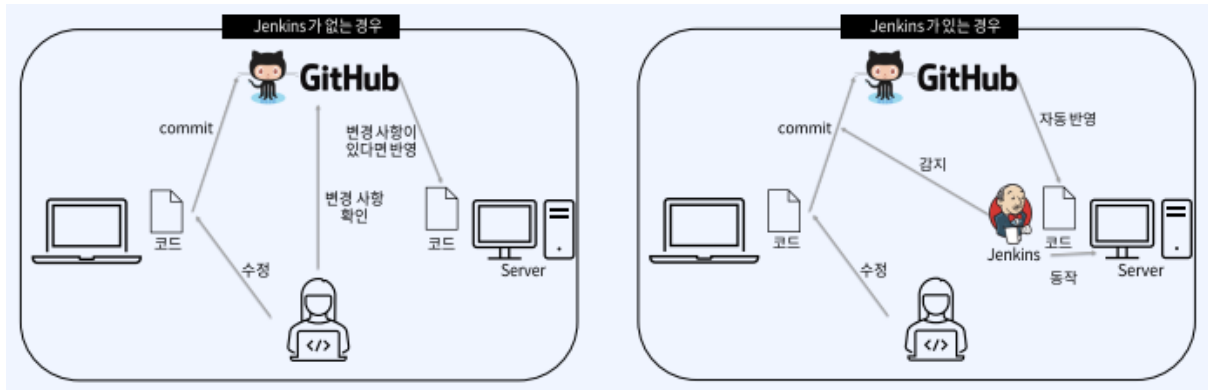
## What is Jenkins?

젠킨스는 소프트웨어 개발 시 빌드, 테스트, 배포 프로세스를 자동화해주는 **지속적인 통합 (CI, Continuous Integration)** 서비스를 제공하는 툴이다.

다수의 개발자들이 하나의 프로그램을 개발할 때 버전 충돌을 방지하기 위해 각자 작업한 내용을 공유 영역에 있는 저장소에 빈번히 업로드함으로써 지속적 통합이 가능하도록 해준다.

원래 허드슨 프로젝트에서 Java로 개발되었고, 허드슨의 개발은 2004년 여름 썬 마이크로 시스템즈에서 시작되었다. 그리고 2005년 2월에 java.net에 처음 출시되었다. 오라클과의 분쟁으로 허드슨으로부터 분기 후 2011년 2월 2일 Jenkins로 출시되었다.

젠킨스와 같은 CI툴이 등장하기 전에는 일정 시간마다 빌드를 실행하는 방식이 일반적이었다. 특히 개발자들이 당일 작성한 소스들의 커밋이 모드 끝난 심야 시간대에 이러한 빌드가 타이머에 의해 집중적으로 진행되었는데, 이를 nightly-build라 한다. 하지만, 젠킨스는 정기적인 빌드에서 한발 나아가 서버버전, Git 과 같은 버전관리시스템과 연동하여 소스의 커밋을 감지하면 자동적으로 자동화 테스트가 포함된 빌드가 작동되도록 설정할 수 있다.



## 젠킨스가 주는 이점

개발중인 프로젝트에서 커밋은 매우 빈번히 일어나기 때문에 커밋 횟수만큼 빌드를 실행하는 것이 아니라 작업이 큐잉되어 자신이 실행될 차례를 기다리게 된다. 코드의 변경과 함께 이뤄지는 이 같은 자동화된 빌드와 테스트 작업들은 다음과 같은 이점들을 가져다 준다.

- 프로젝트 표준 컴파일 환경에서의 컴파일 오류 검출
- 자동화 테스트 수행
- 정적 코드 분석에 의한 코딩 규약 준수 여부 체크
- 프로파일링 툴을 이용한 소스 변경에 따른 성능 변화 감시
- 결합 테스트 환경에 대한 배포작업

이 외에도 젠킨스는 1400여가지가 넘는 플러그인을 온라인으로 간단히 인스톨 할 수 있는 기능을 제공하고 있으며 파이썬과 같은 스크립트를 이용해 손쉽게 자신에게 필요한 기능을 추가 할 수도 있다.

소스코드 일관성 유지	자동 빌드	자동 테스트
<ul style="list-style-type: none"> <li>- 소스 관리 시스템은 필요</li> <li>- 정적 코드 분석에 의한 코딩 규약 준수여부 체크</li> </ul>	<ul style="list-style-type: none"> <li>- 커밋에 따른 자동 빌드</li> <li>- 시간 간격에 의한 빌드</li> <li>- 빌드 관리 툴이 있다면 효율성 극대화</li> </ul>	<ul style="list-style-type: none"> <li>- 빌드 과정에서의 테스트</li> <li>- 결합 테스트 환경 배포</li> <li>- 자동 수행으로 안전망 역할</li> </ul>
일일 체크아웃과 빌드	컴파일 오류 검출	프로파일링

## 각종 배치 작업의 간략화

프로젝트 기간 중에 개발자들은 순수한 개발 작업 이외에 DB셋업이나 환경설정, Deploy 작업과 같은 단순 작업에 시간과 노력을 들이는 경우가 빈번하다. 데이터베이스의 구축, 어플리케이션 서버로의 Deploy, 라이브러리 릴리즈와 같이 이전에 CLI로 실행되던 작업들이 젠킨스 덕분에 웹 인터페이스로 손쉽게 가능해졌다.

## Build 자동화의 확립

빌드 툴의 경우 Java는 maven과 gradle이 자리잡고 있으며, 이미 빌드 관리 툴을 이용해 프로젝트를 진행하고 있다면 젠킨스를 사용하지 않을 이유가 하나도 없다. 젠킨스와 연동하여 빌드 자동화를 통해 프로젝트 진행의 효율성을 높일 수 있다.

## 자동화 테스트

자동화 테스트는 젠킨스를 사용해야 하는 가장 큰 이유 중 하나이며, 사실상 자동화 테스트가 포함되지 않은 빌드는 CI자체가 불가능하다고 봐도 무방하다. 젠킨스는 Subversion이나 Git과 같은 버전관리시스템과 연동하여 코드 변경을 감지하고 자동화 테스트를 수행하기 때문에 만약 개인이 미처 실시하지 못한 테스트가 있다 하여도 든든한 안전망이 되어준다. 제대로 테스트를 거치지 않은 코드를 커밋하게 되면 화난 젠킨스를 만나게 된다.

## 코드 표준 준수여부 검사

자동화 테스트와 마찬가지로 개인이 미처 실시하지 못한 코드 표준 준수 여부의 검사나 정적 분석을 통한 코드 품질 검사를 빌드 내부에서 수행함으로써 기술적 부채의 감소에도 크게 기여한다.

## 빌드 파이프라인 구성

2개 이상의 모듈로 구성되는 레이어드 아키텍처가 적용 된 프로젝트에는 그에 따는 빌드 파이프라인 구성이 필요하다. 예를 들면, 도메인 -> 서비스 -> UI와 같이 각 레이어의 참조 관계에 따라 순차적으로 빌드를 진행하지 않으면 안된다. 젠킨스에서는 이러한 빌드 파이프라인의 구성을 간단히 할 수 있으며, 스크립트를 통해서 매우 복잡한 제어까지도 가능하다.

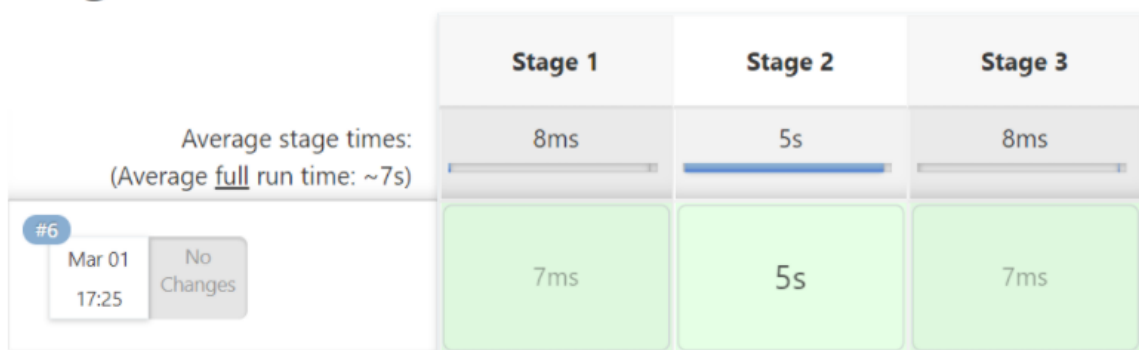
## Jenkins 특징

- Jenkinsfile

- **Jenkinsfile** 을 이용해 Job 혹은 파이프라인을 정의할 수 있다. **Jenkinsfile** 덕분에 일반 소스코드를 다루는 Github 업로드, Vscode 로 수정하는 것으로 파일을 이용할 수 있음
- 기본적으로 **Jenkinsfile** 을 통해 젠킨스를 실행함
- Scripted Pipeline (스크립트 파이프라인)
  - Jenkins 관련 구조를 자세히 가지지 않고 프로그램의 흐름을 **Groovy** 라는 동적 객체 지향 프로그래밍 언어를 이용해 관리되었음  
Java 가상머신에서 동작하고 Java와 유사함
  - 매우 유연하지만 Groovy를 알아야하므로 진입장벽이 있음.

```
node { ## 빌드를 수행할 node 또는 agent를 의미한다.
    stage("Stage 1"){
        echo "Hello"
    }
    stage("Stage 2"){
        echo "World"
        sh "sleep 5"
    }
    stage("Stage 3"){
        echo "Good to see you!"
    }
}
```

## Stage View



- Declarative Pipeline (선언적 파이프라인)
  - 2016년 경 **Cloudbees** 에서 개발
  - 사전에 정의된 구조만 사용할 수 있기 때문에 CI/CD 파이프라인이 단순한 경우에 적합하며 아직은 많은 제약사항이 따른다.
  - 공식 문서