

Jenkins

What is Jenkins

Jenkins는 소프트웨어의 빌드, 테스트, 배포에 관련된 모든 종류의 업무를 자동화하는 데 사용되는 오픈 소스 자동화 서버입니다.

Installing Jenkins using Docker

<https://www.jenkins.io/doc/book/installing/docker/#docker>

1. 터미널 윈도우를 실행합니다.

2. 다음 명령어를 실행하여 Docker에 브릿지 네트워크를 생성합니다.

```
docker network create jenkins
```

3. Jenkins 노드 내부에서 Docker 명령어를 실행하기 위해서는 다음 명령어를 실행하여 `docker:dind` 이미지를 다운로드 해야 합니다.

```
docker run \  
  --name jenkins-docker \ (1)  
  --rm \ (2)  
  --detach \ (3)  
  --privileged \ (4)  
  --network jenkins \ (5)  
  --network-alias docker \ (6)  
  --env DOCKER_TLS_CERTDIR=/certs \ (7)  
  --volume jenkins-docker-certs:/certs/client \ (8)  
  --volume jenkins-data:/var/jenkins_home \ (9)  
  --publish 2376:2376 \ (10)  
  docker:dind \ (11)  
  --storage-driver overlay2 (12)
```

(1) Docker 컨테이너 이름을 지정합니다.

(2) 컨테이너가 종료되었을 때 자동으로 해당 컨테이너를 제거합니다.

(3) Docker 컨테이너를 백그라운드로 실행합니다.

(4) 권한을 부여합니다.

(5) 앞 단계에서 생성한 브릿지 네트워크를 사용합니다.

(6) Docker 컨테이너 내부의 Docker를 jenkins 네트워크에서 호스트 이름 docker로 사용 가능하도록 합니다.

(7) Docker 서버에서 TLS의 사용을 활성화 합니다.

(8) 컨테이너 내부의 /certs/client 디렉토리를 상기에서 생성한 jenkins-docker-certs Docker 볼륨에 매핑합니다.

(9) 컨테이너 내부의 /var/jenkins_home 디렉토리를 상기에서 생성한 jenkins-data Docker 볼륨에 매핑합니다.

다.

(10) Docker 데몬 포트를 호스트 머신에 노출시킵니다.

(11) docker:dind 이미지

(12) Docker 볼륨을 위한 스토리지 드라이버

Note: 주석이 없는 하기 명령어를 이용하여 실행할 수 있습니다.

```
docker run --name jenkins-docker --rm --detach \
  --privileged --network jenkins --network-alias docker \
  --env DOCKER_TLS_CERTDIR=/certs \
  --volume jenkins-docker-certs:/certs/client \
  --volume jenkins-data:/var/jenkins_home \
  --publish 2376:2376 \
  docker:dind --storage-driver overlay2
```

4. 하기 두 단계를 통해서 공식 Jenkins Docker 이미지의 커스텀을 수행합니다.

하기 내용을 이용하여 Dockerfile을 생성합니다.

```
FROM jenkins/jenkins:2.375.2
USER root
RUN apt-get update && apt-get install -y lsb-release
RUN curl -fsSL /usr/share/keyrings/docker-archive-keyring.asc \
  https://download.docker.com/linux/debian/gpg
RUN echo "deb [arch=$(dpkg --print-architecture) \
  signed-by=/usr/share/keyrings/docker-archive-keyring.asc] \
  https://download.docker.com/linux/debian \
  $(lsb_release -cs) stable" > /etc/apt/sources.list.d/docker.list
RUN apt-get update && apt-get install -y docker-ce-cli
USER jenkins
RUN jenkins-plugin-cli --plugins "blueocean docker-workflow"
```

상기 Dockerfile을 이용하여 새로운 Docker 이미지를 빌드하고 의미있는 이름을 부여합니다.

```
docker build -t myjenkins-blueocean:2.375.2-1 .
```

참고로 위 단계를 최초로 수행할 때 공식 Jenkins Docker 이미지를 다운로드할 것입니다.

5. 하기 명령어를 이용하여 myjenkins-blueocean:2.375.2-1 이미지의 컨테이너를 실행합니다.

```
docker run \
  --name jenkins-blueocean \
  --restart=on-failure \
  --detach \
  --network jenkins \
  --env DOCKER_HOST=tcp://docker:2376 \
  --env DOCKER_CERT_PATH=/certs/client \
  --env DOCKER_TLS_VERIFY=1 \
  --publish 8080:8080 \
  --publish 50000:50000 \
  --volume jenkins-data:/var/jenkins_home \
  --volume jenkins-docker-certs:/certs/client:ro \
  myjenkins-blueocean:2.375.2-1
```

Note: 주석이 없는 하기 명령어를 이용하여 실행할 수 있습니다.

```
docker run --name jenkins-blueocean --restart=on-failure --detach \
  --network jenkins --env DOCKER_HOST=tcp://docker:2376 \
  --env DOCKER_CERT_PATH=/certs/client --env DOCKER_TLS_VERIFY=1 \
  --publish 8080:8080 --publish 50000:50000 \
  --volume jenkins-data:/var/jenkins_home \
  --volume jenkins-docker-certs:/certs/client:ro \
  myjenkins-blueocean:2.375.2-1
```

6. Post-installation setup wizard(<https://www.jenkins.io/doc/book/installing/docker/#setup-wizard>)를 진행합니다.

Post-installation setup wizard

위자드를 통해서 Jenkins를 연락하고 플러그인을 통해 커스터마이징을 수행하며 Jenkins에 접근할 수 있는 첫 번째 Admin 계정을 생성할 수 있습니다.

Unlocking Jenkins

맨 처음 Jenkins 인스턴스에 접근하면 자동으로 생성된 패스워드를 이용하여 연락하도록 요청을 받습니다.

1. `http://localhost:8080` 주소로 접근하여 **Unlock Jenkins** 페이지가 나타날 때까지 대기합니다.

2. **Jenkins 콘솔 로그**에서 자동으로 생성된 **alphanumeric** 패스워드를 복사합니다.

만약 공식 jenkins/jenkins Docker 이미지를 이용하여 Jenkins를 실행 중이라면 하기 명령어를 이용하여 패스워드를 확인할 수 있습니다.

```
sudo docker exec ${CONTAINER_ID or CONTAINER_NAME} cat
/var/jenkins_home/secrets/initialAdminPassword
```

3. 복사한 패스워드(`dcd742ec21a34cb68851ca0921a8c61e`)를 기입하여 **Continue** 버튼을 클릭합니다.

Customizing Jenkins with plugins

연락을 수행하고 나면 두 가지 선택을 할 수 있습니다.

- Install suggested plugins : 권장하는 플러그인을 설치합니다.
- Select plugins to install : 설치하고 싶은 플러그인을 선택합니다.

Note: 무엇을 설치해야 할지 잘 모르겠다면 Install suggested plugins을 선택합니다. 이후에 **Manage Jenkins > Manage Plugins** 를 통해서 추가로 플러그인 설치를 진행할 수 있습니다.

Creating the first administrator user

Customizing Jenkins with plugins을 끝내면 마지막으로 관리자 계정을 생성해야 합니다.

1. Create First Admin User 페이지가 나타나면 계정에 대한 자세항 사항을 필드에 기입하고 **Save and Continue** 를 클릭합니다.

계정명 : youngwoo

암호 : 1234

암호 확인 : 1234

이름 : Youngwoo Yoon

이메일 주소 : zerocow1995@gmail.com

2. Instance Configuration 페이지가 나타나면 **Jenkins URL**을 확인하여 **Save and Finish** 를 클릭합니다.

Jenkins URL : <http://192.168.53.9:8080/>

3. Jenkins is almost ready! 화면이 나타나면 **Restart** 버튼을 클릭하여 Jenkins를 재시작 합니다.


1분 후에 화면이 자동으로 새로고침 되지 않는다면 수동으로 새로고침 합니다.


4. 새로고침 후 Welcome to Jenkins! 화면이 나타납니다. 로그인을 수행합니다.







Welcome to Jenkins!

☐ 로그인 상태 유지로그인


 **Jenkins**


Q 검색 (CTRL+K) 


 1  Youngwoo Yoon   로그아웃


Dashboard >


+ 새로운 Item


 사람

 빌드 기록


 Jenkins 관리

 My Views

 블루 오션 열기


빌드 대기 목록 

빌드 대기 항목이 없습니다.

빌드 실행 상태 

1 대기 중


2 대기 중

 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 

REST API Jenkins 2.375.2

Note: 보안 이슈는 추후에 처리 요망

Q 검색 (CTRL+K)

?

 1

Youngwoo Yoon ▾

로그아웃

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

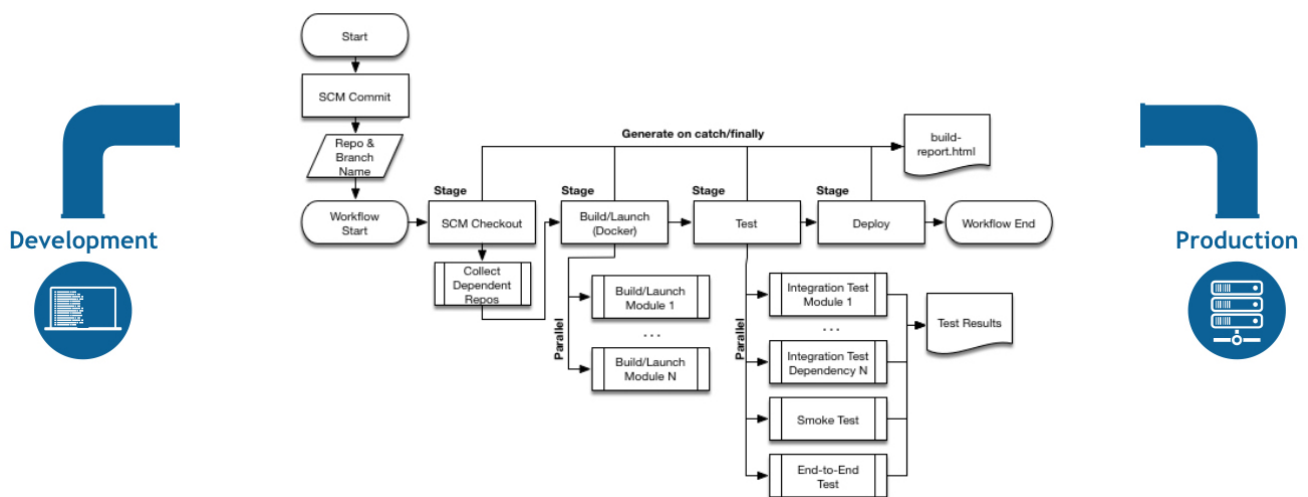
[Manage Jenkins](#)

Initial Settings

<https://www.jenkins.io/doc/book/installing/initial-settings/>

대부분의 Jenkins 구성 변경은 Jenkins 유저 인터페이스를 통해서 수행할 수 있습니다. 몇몇 구성 설정 값들은 Jenkins가 시작할 때만 수정할 수 있습니다. 상기 링크를 클릭하면 해당 구성 설정 값들을 어떻게 변경할 수 있는지 확인할 수 있습니다.

Flowchart of CD Scenario in Jenkins Pipeline



Pipeline Concepts

하기에 Pipeline 관련 용어 및 문법에 대해서 설명합니다.

Pipeline

Pipeline 은 사용자가 정의한 CD pipeline 모델입니다. Pipeline의 코드는 사용자의 전체 빌드 프로세스를 정의하는데, 일반적으로 애플리케이션을 빌드, 테스트, 배포하는 stages를 포함합니다. 또한 pipeline 블록은 Declarative Pipeline 문법의 핵심 요소 입니다.

Node

Node 는 Jenkins 환경의 일부인 머신이며 하나의 Pipeline을 수행할 수 있습니다. 또한 Node 블록은 Scripted Pipeline 문법의 핵심 요소 입니다.

Stage

Stage 블록은 전체 Pipeline(예를 들면, "Build", "Test", "Deploy")을 통해서 수행되는 업무들의 구별되는 부분 집합인데, 많은 플러그인들에 의해서 Jenkins Pipeline의 상태/진행 현황을 시각화 하거나 표현하는데 사용합니다.

Step

하나의 업무를 의미합니다. 예를 들면, shell 명령어인 make를 실행하기 위해서 `sh step: sh 'make'` 을 사용합니다.

Pipeline syntax overview

하기에 Declarative Pipeline syntax 및 Scripted Pipeline syntax 문법의 차이를 보여줍니다.

Declarative Pipeline fundamentals

선언형 파이프라인 문법에서는 `pipeline` 블록이 사용자의 전체 파이프라인을 통해서 완료되는 모든 업무를 정의합니다.

```
Jenkinsfile (Declarative Pipeline)
pipeline {
  agent any → (1)
  stages {
    stage('Build') { → (2)
      steps {
        // → (3)
      }
    }
    stage('Test') { → (4)
      steps {
        // → (5)
      }
    }
    stage('Deploy') { → (6)
      steps {
        // → (7)
      }
    }
  }
}
```

- (1) 어떠한 agent도 해당 파이프라인 또는 어떠한 스테이지도 수행할 수 있습니다.
- (2) "Build" 스테이지를 정의합니다.
- (3) "Build" 스테이지와 관련된 스텝들을 수행합니다.
- (4) "Test" 스테이지를 정의합니다.
- (5) "Test" 스테이지와 관련된 스텝들을 수행합니다.
- (6) "Deploy" 스테이지를 정의합니다.
- (7) "Deploy" 스테이지와 관련된 스텝들을 수행합니다.

Scripted Pipeline fundamentals

```
Jenkinsfile (Scripted Pipeline)
node { → (1)
    stage('Build') { → (2)
        // → (3)
    }
    stage('Test') { → (4)
        // → (5)
    }
    stage('Deploy') { → (6)
        // → (7)
    }
}
```

- (1) 어떠한 agent도 해당 파이프라인 또는 어떠한 스테이지도 수행할 수 있습니다.
- (2) "Build" 스테이지를 정의합니다.
- (3) "Build" 스테이지와 관련된 스텝들을 수행합니다.
- (4) "Test" 스테이지를 정의합니다.
- (5) "Test" 스테이지와 관련된 스텝들을 수행합니다.
- (6) "Deploy" 스테이지를 정의합니다.
- (7) "Deploy" 스테이지와 관련된 스텝들을 수행합니다.

Pipeline example

하기 예시는 선언형 파이프라인 문법을 사용한 Jenkinsfile 입니다.

```
Jenkinsfile (Declarative Pipeline)
pipeline { → (1)
    agent any → (2)
    options {
        skipStagesAfterUnstable()
    }
    stages {
        stage('Build') { → (3)
            steps { → (4)
                sh 'make' → (5)
            }
        }
        stage('Test'){
            steps {
                sh 'make check'
                junit 'reports/**/*.*xml' → (7)
            }
        }
        stage('Deploy') {
            steps {
                sh 'make publish'
            }
        }
    }
}
```


- (1) `pipeline` is Declarative Pipeline-specific syntax that defines a "block" containing all content and instructions for executing the entire Pipeline.
- (2) `agent` is Declarative Pipeline-specific syntax that instructs Jenkins to allocate an executor (on a node) and workspace for the entire Pipeline.
- (3) `stage` is a syntax block that describes a stage of this Pipeline. Read more about stage blocks in Declarative Pipeline syntax on the Pipeline syntax page. As mentioned above, stage blocks are optional in Scripted Pipeline syntax.
- (4) `steps` is Declarative Pipeline-specific syntax that describes the steps to be run in this stage.
- (5) `sh` is a Pipeline step (provided by the Pipeline: Nodes and Processes plugin) that executes the given shell command.
- (6) `junit` is another Pipeline step (provided by the JUnit plugin) for aggregating test reports.
- (7) `sh` is a Pipeline step (provided by the Pipeline: Nodes and Processes plugin) that executes the given shell command.

더 자세한 파이프라인 문법에 대해서는 하기 링크를 참고한다.

<https://www.jenkins.io/doc/book/pipeline/syntax/>