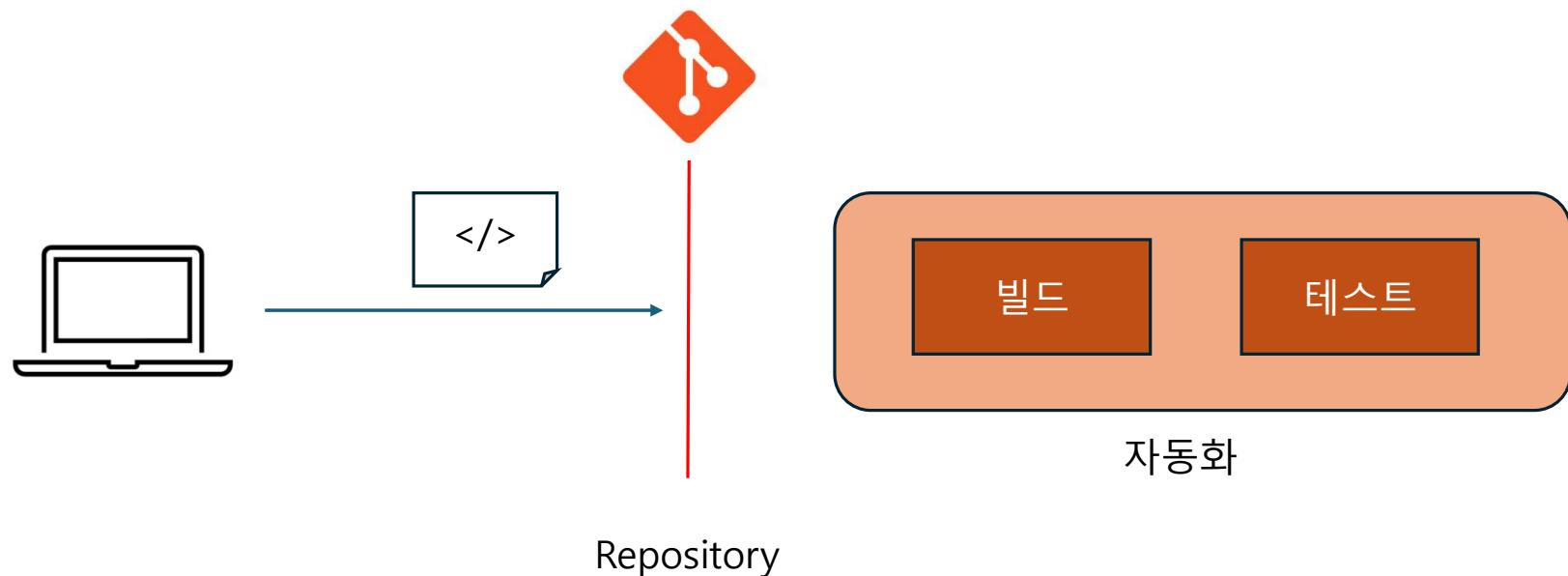


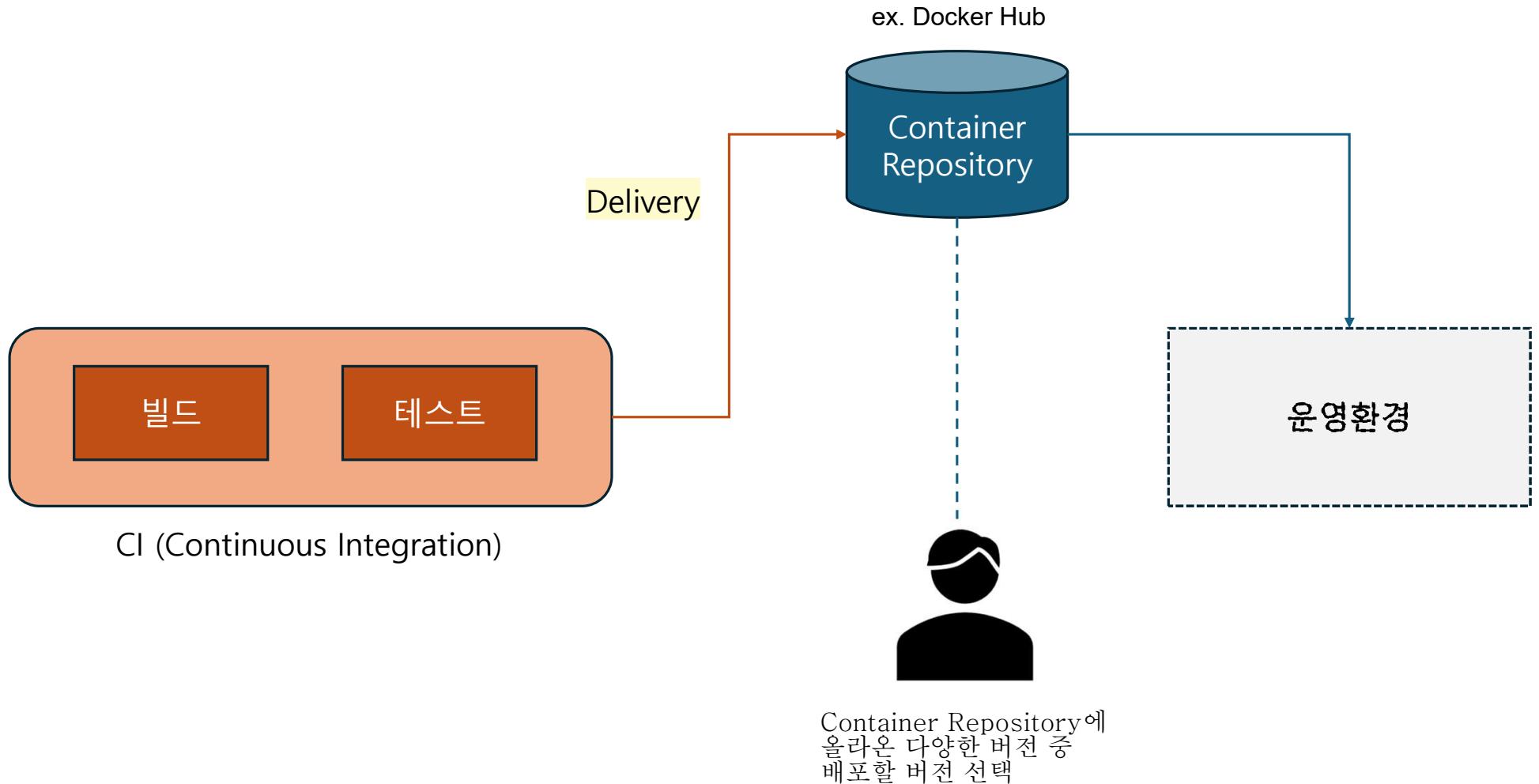
CI/CD
Jenkins

CI (Continuous Integration)

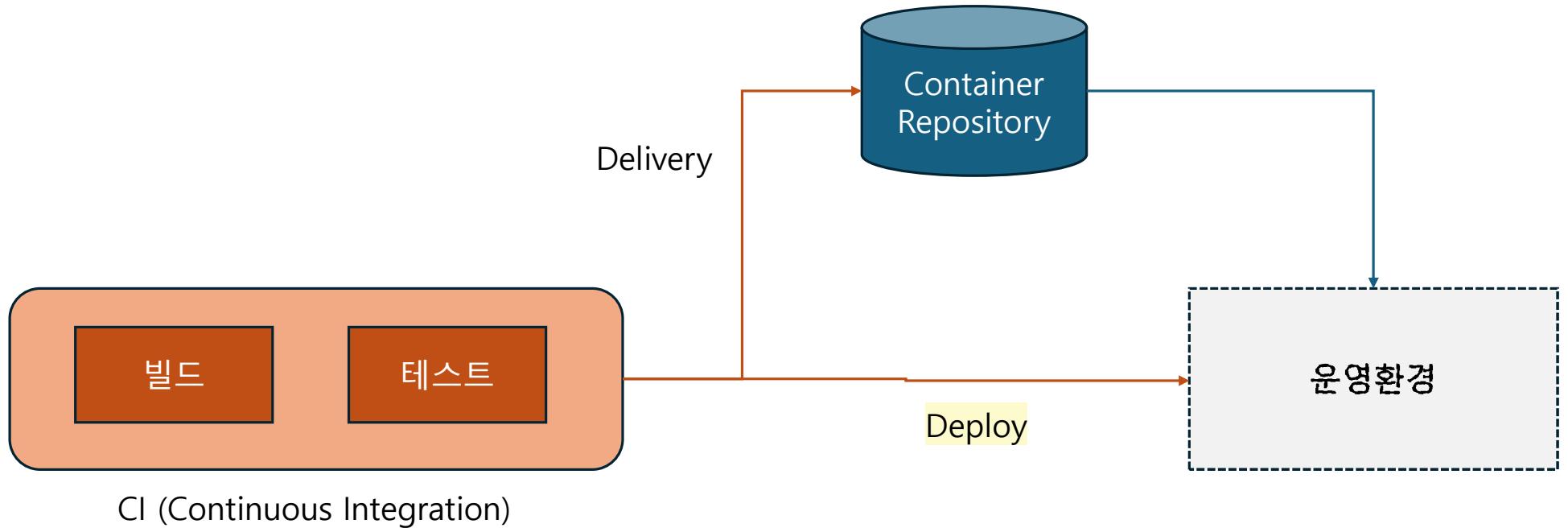
- 개발자는 코드 변경사항을 주기적으로 빈번하게 머지해야 한다.
- 빌드/테스트의 자동화



CD (Continuous Delivery)



CD (Continuous Deploy)



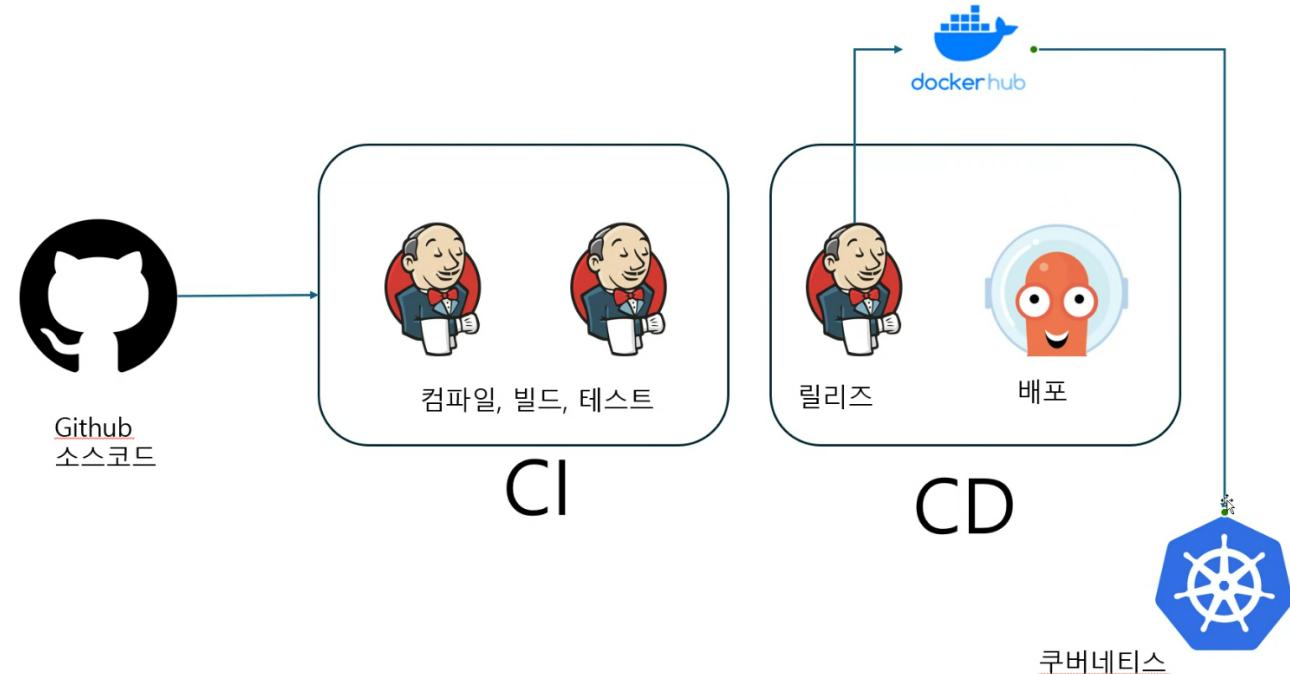
- + Build, Test, Delivery : Jenkins
- + Deploy : ArgoCD + Kubernetes Deploy

Jenkins 소개

- 소프트웨어 개발 프로세스를 자동화하는데 사용되는 오픈 소스 자동화 플랫폼
- 주로 CI/CD(Continuous Integration/Continuous Delivery) 파이프라인 구축에 활용

주요 기능

- 자동화된 빌드 및 테스트
- CI/CD 파이프라인 구축 (delivery)
- 다양한 플러그인 지원



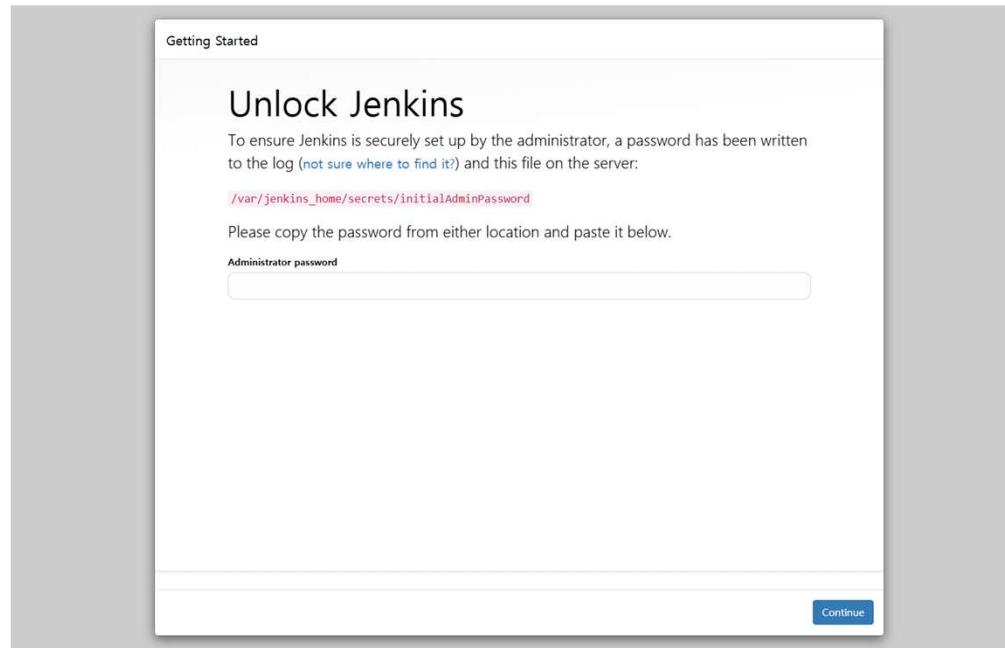
Install Suggested Plugins

```
version: "3"
services:
  jenkins:
    image: solarhc/jenkins:jdk17
    container_name: jenkins
    restart: always
    user: root
    volumes:
      - ./jenkins_home:/var/jenkins_home
      - /var/run/docker.sock:/var/run/docker.sock
    ports:
      - 8000:8080
    environment:
      - DOCKER_HOST=unix:///var/run/docker.sock
```

docker-compose.yml

- docker-compose.yml 파일을 C:\server\Jenkins 디렉토리에 생성
- C:\server\Jenkins 디렉토리에 jenkins_home 디렉토리 생성
 - C:\server\jenkins> **mkdir jenkins_home**
- C:\server\Jenkins 디렉토리에서 docker compose up 명령어 실행
 - C:\server\Jenkins> **docker compose up -d**

<http://localhost:8000> 접속



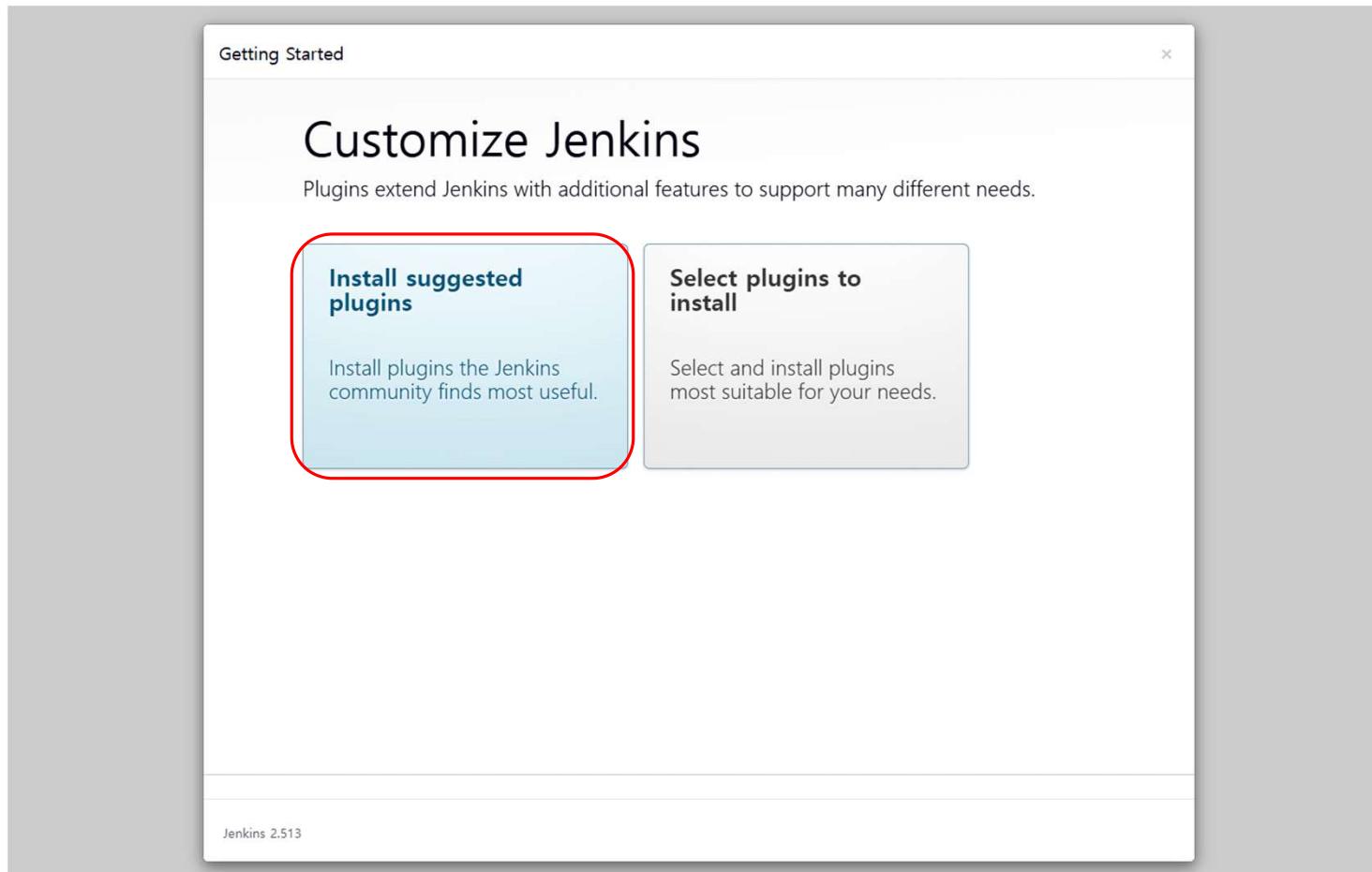
초기 비밀번호는 C:\server\jenkins\jenkins_home\secrets 디렉토리 initialAdminPassword 파일에 있음.

C:\server\jenkins\jenkins_home\secrets> **cat initialAdminPassword**

2adb81754e2440f886113c5218b75066



Install Suggested Plugins



Getting Started

Create First Admin User

계정명

암호

암호 확인

이름

이메일 주소

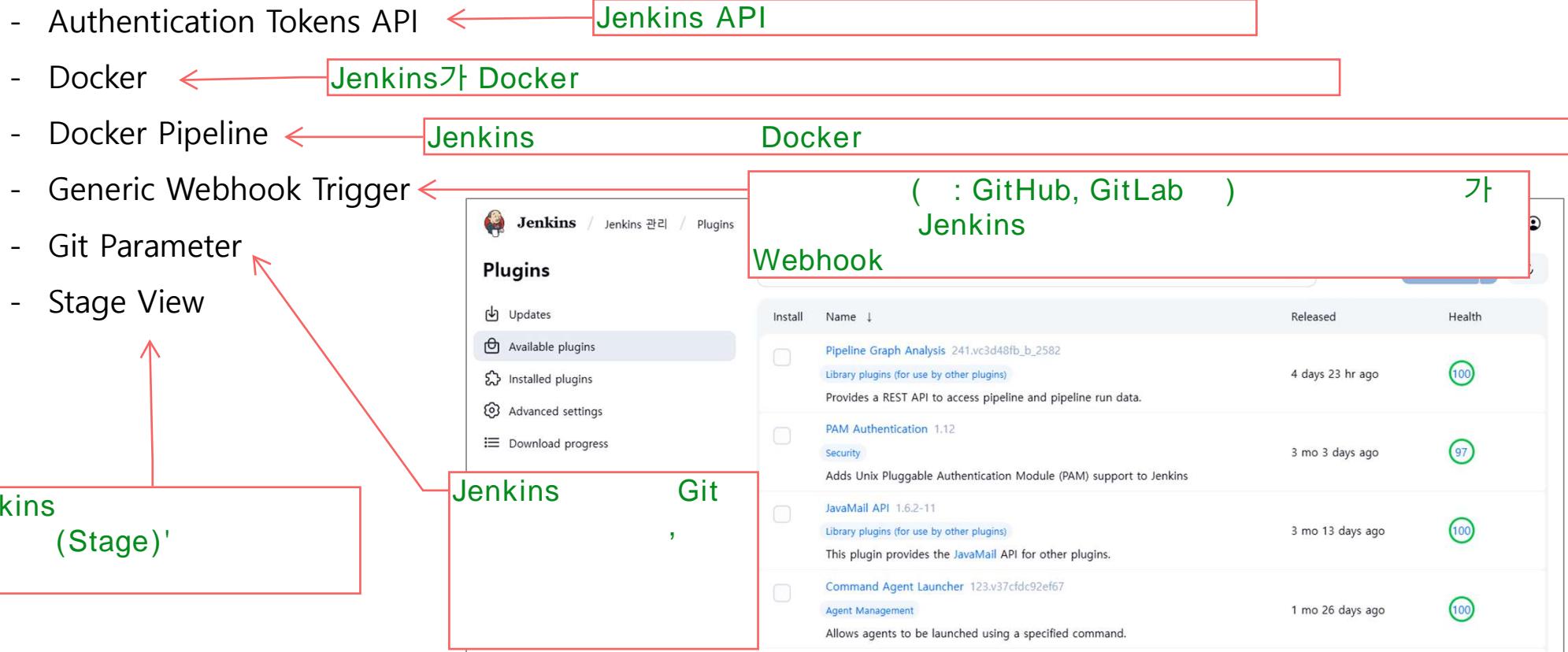
Jenkins 2.513

Skip and continue as admin

Save and Continue

추가 Plugin 설치

'Jenkins 관리 > 플러그인 관리'에서 다음 플러그인들을 설치



Global Tool 설정 - JDK

Jenkins 관리 > Tools

The screenshot shows the Jenkins Global Tool Configuration page for JDK installations. It displays two sections: a general view and a detailed configuration for a specific JDK instance.

General View:

- Header: JDK installations
- Button: Add JDK

Detailed Configuration (OpenJDK 17):

- JDK Name: OpenJDK 17
- Install automatically 선택 해제 (unchecked)
- JAVA_HOME: /opt/java/openjdk

A callout box highlights the JAVA_HOME field with the note: **(※ Jenkins 설정 > 시스템 정보 > java.home 값을 확인하여 입력)** (Note: Verify the value in Jenkins System Information > java.home before inputting).

Global Tool 설정 – Gradle

Jenkins 관리 > Tools

The screenshot shows the Jenkins Global Tool configuration page for Gradle. At the top left, there is a section titled "Gradle installations" with a "Add Gradle" button. Below this, a "Gradle" configuration card is expanded, showing the following details:

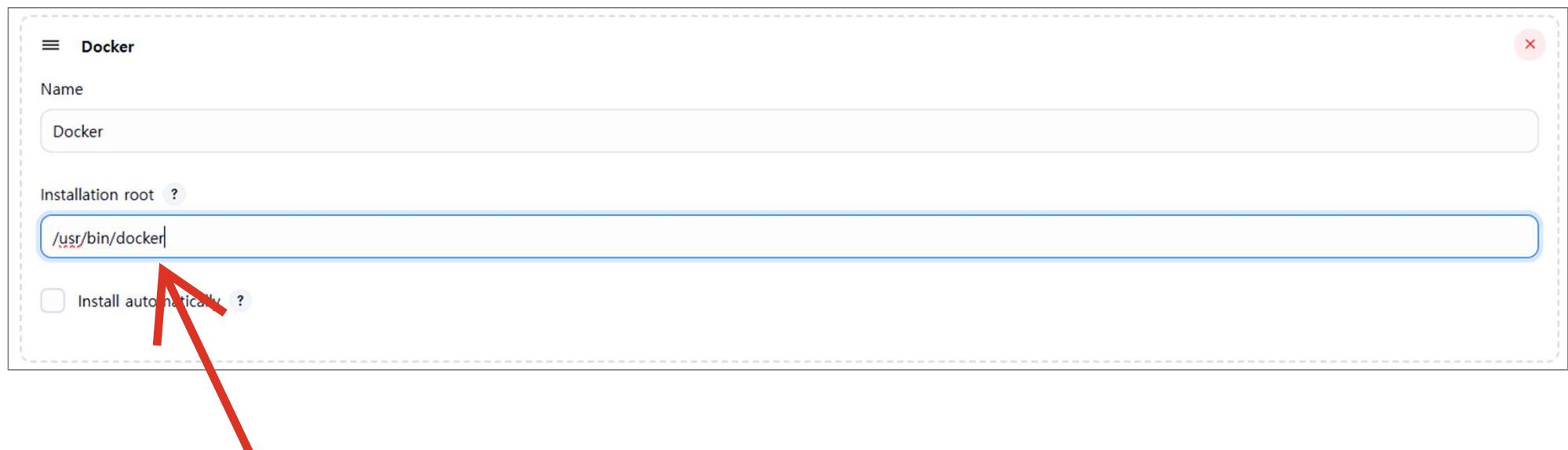
- name**: Gradle 8.14.2
- Install automatically**:
- Install from Gradle.org** section:
 - Version**: Gradle 8.14.2
- Add Installer** button

A callout box highlights the "Gradle name" and "Version" fields with the text: "- Gradle name: Gradle 8.14.2" and "- Version: Gradle 8.14.2 선택".

Global Tool 설정 – Docker

Docker installations

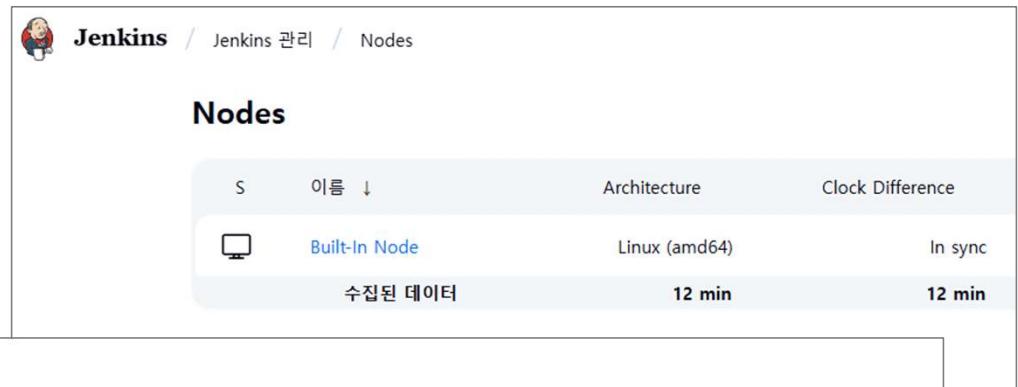
Add Docker



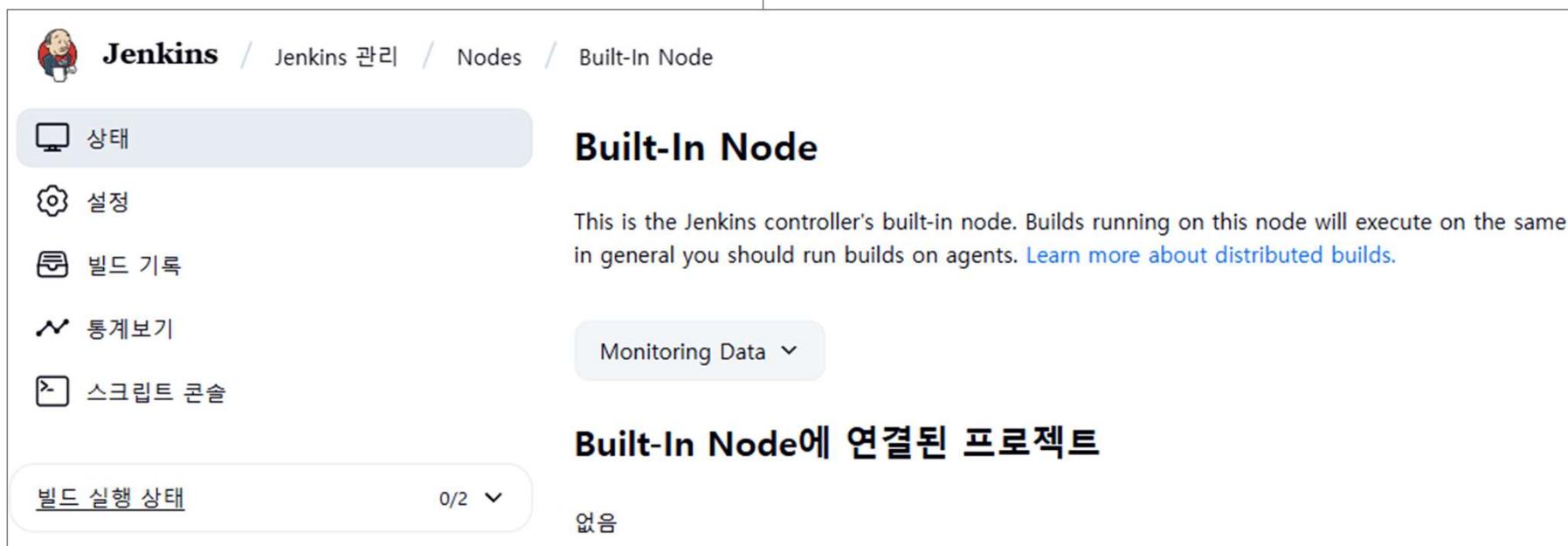
1. docker ps | findstr jenkins : jenkins 컨테이너 찾기
2. docker exec -it (container hash value) bash : Docker Bash 사용 (ex. 컨테이너 내의 OS 환경으로 접속)
3. which docker : docker 폴더 혹은 파일이 위치한 경로 찾기

노드 레이블 설정

Jenkins 관리 > Nodes > Build-In Node



S	이름	Architecture	Clock Difference
	Built-In Node	Linux (amd64)	In sync



The screenshot shows the Jenkins interface for managing nodes. On the left, there's a sidebar with links: 상태 (Status), 설정 (Configuration), 빌드 기록 (Build History), 통계보기 (Statistics), and 스크립트 콘솔 (Script Console). The main content area is titled "Built-In Node". It contains a message: "This is the Jenkins controller's built-in node. Builds running on this node will execute on the same machine as the Jenkins master. In general you should run builds on agents. [Learn more about distributed builds.](#)". Below this is a "Monitoring Data" section with a dropdown menu. At the bottom, there's a table titled "Built-In Node에 연결된 프로젝트" (Projects connected to Built-In Node) which is currently empty.

노드 레이블 설정 (계속)

The screenshot shows the Jenkins Node configuration interface for a 'Built-In Node'. The left sidebar has links for '상태', '설정' (which is highlighted with a red box), '빌드 기록', '통계보기', and '스크립트 콘솔'. The main area shows 'Number of executors' set to 2 and '라벨' set to 'master' (also highlighted with a red box). A yellow callout box says '라벨명을 master로 설정'. Below this, 'Usage' is set to 'Use this node as much as possible'. Under 'Node Properties', there are three unchecked checkboxes: 'Disable deferred wipeout on this node', 'Disk Space Monitoring Thresholds', and 'Environment variables'. At the bottom are 'Save' and 'Apply' buttons.

Jenkins / Nodes / Built-In Node / Configure

상태

설정

빌드 기록

통계보기

스크립트 콘솔

라벨

master

라벨명을 master로 설정

빌드 실행 상태 0/2

Node Properties

Disable deferred wipeout on this node

Disk Space Monitoring Thresholds

Environment variables

Save Apply

Credential 등록 – github access token 발급

The image shows two screenshots of the GitHub interface. The left screenshot is the GitHub Dashboard, featuring a search bar, a 'Dashboard' button, and a 'Top repositories' section. The right screenshot is the user profile page for 'solarhc', showing various profile links like 'Your profile', 'Your repositories', and 'Settings'. A red box highlights the 'Settings' icon in the top right corner of the dashboard header and the 'Settings' link in the sidebar of the user profile page.

Dashboard

Type to search

Top repositories

New

Home

Ask Copilot

Latest changes

9 hours ago

Fluency and coherence evaluators added

solarhc

Set status

Your profile

Your repositories

Your Copilot

Your projects

Your stars

Your gists

Your organizations

Your enterprises

Your sponsors

Try Enterprise

Feature preview

Settings

Credential 등록 – github access token 발급 (계속)

The screenshot shows the GitHub developer settings interface. On the left, a sidebar lists various options: Copilot, Pages, Saved replies, Security, Code security, Integrations, Applications, Scheduled reminders, Archives, Security log, Sponsorship log, and Developer settings. The 'Developer settings' option is highlighted with a red box. The main content area shows the 'GitHub Apps' section, which includes 'OAuth Apps', 'Personal access tokens' (which is currently selected and highlighted with a blue box), 'Fine-grained tokens', and 'Tokens (classic)'. Below this, a large box displays the message 'No personal access token created' with a key icon, followed by the text 'Need an API token for scripts or testing? Generate a personal access token for quick access to the GitHub API.' A 'Generate new token' button is shown, also highlighted with a red box. At the bottom right of the message box, there is a link to 'GitHub API'.

Credential 등록 – github access token 발급 (계속)

New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Note

Jenkins build

What's this token for?

Expiration

The token will expire on the selected date

Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

<input checked="" type="checkbox"/> repo	Full control of private repositories
<input type="checkbox"/> repo:status	Access commit status
<input type="checkbox"/> repo_deployment	Access deployment status
<input type="checkbox"/> public_repo	Access public repositories
<input type="checkbox"/> repo:invite	Access repository invitations
<input type="checkbox"/> security_events	Read and write security events

Credential 등록 – github access token 발급 (계속)

<input type="checkbox"/> admin:org	Full control of orgs and teams, read and write org projects
<input type="checkbox"/> write:org	Read and write org and team membership, read and write org projects
<input type="checkbox"/> read:org	Read org and team membership, read org projects
<input type="checkbox"/> manage_runners:org	Manage org runners and runner groups
<input type="checkbox"/> admin:public_key	Full control of user public keys
<input type="checkbox"/> write:public_key	Write user public keys
<input type="checkbox"/> read:public_key	Read user public keys
<input checked="" type="checkbox"/> admin:repo_hook	Full control of repository hooks
<input checked="" type="checkbox"/> write:repo_hook	Write repository hooks
<input checked="" type="checkbox"/> read:repo_hook	Read repository hooks
<input type="checkbox"/> admin:org_hook	Full control of organization hooks
<input type="checkbox"/> gist	Create gists

Credential 등록 – github access token 발급 (계속)

Personal access tokens (classic) Generate new token ▾

Tokens you have generated that can be used to access the [GitHub API](#).

⚠ Make sure to copy your personal access token now. You won't be able to see it again!

✓ Delete

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Credential 등록 – github access token 등록

Jenkins 관리 > Credentials

The screenshot shows the Jenkins 'Credentials' management page. At the top, there's a header with tabs T, P, Store (with a dropdown arrow), and a 'Domain' filter. Below this, a section titled 'Stores scoped to Jenkins' lists 'Domains'. Under 'Domains', there are two entries: 'System' (with a user icon) and '(global)' (with a gear icon). The '(global)' entry is highlighted with a red rectangle. On the left, there are icons for '아이콘:' followed by S, M, and L. On the right, there's a blue button labeled '+ Add Credentials' with a red rectangle around it. A message at the bottom says 'This credential domain is empty. How about [adding some credentials?](#)'.

Kind	Description
This credential domain is empty. How about adding some credentials?	

Credential 등록 – github access token 등록 (계속)

New credentials

Kind
Username with password

Scope ?
Global (Jenkins, nodes, items, all child items, etc)

Username ?
github-id (깃허브 아이디)

Treat username as secret ?

Password ?
..... (깃허브 접근 토큰)

ID ?
github-token (크레デン셜 식별자)

Description ?

Create

Credential 등록 – dockerhub access token 발급

The screenshot shows the Docker website interface. At the top, there's a blue header bar with the Docker logo and a user profile icon. Below the header, a message says "Welcome back, solarhc!". On the left, there's a section titled "Docker products" featuring "docker desktop" and "buildcloud". On the right, there's a sidebar with a user profile picture and the name "solarhc". A dropdown menu is open from the user profile icon, listing options like "What's new", "My profile", "Account settings" (which is highlighted with a red rectangle), "Billing", and "Sign out".

Welcome back, solarhc!

Docker products

docker desktop
Innovate with
Docker Desktop
Your command center for innovative local and cloud native container development.

[Go to download](#) [Launch Docker Desktop](#)

buildcloud
Build with
Docker Build Cloud
Accelerate image build times with access to cloud-based builders and shared cache.

[Go to Build Cloud →](#)

solarhc

- What's new
- My profile
- Account settings**
- Billing
- Sign out

[Go to Scout →](#)

Credential 등록 – dockerhub access token 발급 (계속)

The screenshot shows the Docker Account Center interface. On the left, a sidebar menu includes options like Account information, Email, Password, 2FA, Personal access tokens (which is selected and highlighted with a red box), Connected accounts, Convert, Privacy, and Deactivate. The main content area is titled "Personal access tokens" and contains a brief description: "You can use a personal access token instead of a password for Docker CLI authentication. Create multiple tokens, control their scope, and delete tokens at any time. [Learn more](#)". A prominent blue button labeled "Generate new token" is also highlighted with a red box. Below this, a table lists two existing tokens:

Description	Scope	Status	Source	Created	Last used	Expiration date
Generated through de...	Read, Write, Delete	Active	Auto-generated	Jun 04, 2025 at 17:39:35	Jun 06, 2025 at 15:33:33	Never
Generated through de...	Read, Write, Delete	Active	Auto-generated	Jun 03, 2025 at 21:35:46	Jun 04, 2025 at 08:59:55	Never

At the bottom right, there are pagination controls: "Rows per page: 10" with a dropdown arrow, "1–2 of 2", and navigation arrows.

Credential 등록 – dockerhub access token 발급 (계속)

The screenshot shows the DockerHub 'Personal access tokens / New access token' page. On the left, a sidebar lists account management options: Account information, Email, Password, 2FA, Personal access tokens (which is selected and highlighted in grey), Connected accounts, Convert, Privacy, and Deactivate. The main area is titled 'Create access token' and contains the following fields:

- Access token description: Jenkins build
- Expiration date: None
- Access permissions: Read & Write

A note below states: "Read & Write tokens allow you to push images to any repository managed by your account." At the bottom are 'Cancel' and 'Generate' buttons.

Credential 등록 – dockerhub access token 발급 (계속)

The screenshot shows the Docker Hub 'Personal access tokens' page. On the left, a sidebar lists options: Account information, Email, Password, 2FA, Personal access tokens (which is selected and highlighted in grey), Connected accounts, Convert, Privacy, and Deactivate. The main content area shows a 'Copy access token' button, a note about using it as a password for Docker CLI, and a warning to copy it now as it's only displayed once. It also shows the token description ('Jenkins build'), expiration ('Never'), and access permissions ('Read & Write'). Below this, instructions for using the token with Docker CLI are provided, including a command line example and a placeholder for the token value.

Personal access tokens / New access token

Copy access token

Use this token as a password when you sign in from the Docker CLI client. [Learn more ↗](#)

Make sure you copy your personal access token now. Your personal access token is only displayed once. It isn't stored and can't be retrieved later.

Access token description
Jenkins build

Expires on
Never

Access permissions
Read & Write

To use the access token from your Docker CLI client:

1. Run

```
$ docker login -u solarhc Copy
```

2. At the password prompt, enter the personal access token.

```
dckr_pat_5djqkWD97K5txuJkD2s6f6ucjd0 Copy
```

[Back to access tokens](#)

Credential 등록 – dockerhub access token 등록

Jenkins 관리 > Credentials

The screenshot shows the Jenkins 'Credentials' management interface. At the top, there's a header with tabs T, P, Store (with a dropdown arrow), and a 'Domain' filter. Below this, a section titled 'Stores scoped to Jenkins' lists 'Domains'. Under 'Domains', there are two entries: 'System' (with a user icon) and '(global)' (with a gear icon). The '(global)' entry is highlighted with a red rectangle. On the left, there are icons for '아이콘:' followed by S, M, and L. On the right, there's a blue button labeled '+ Add Credentials' with a red rectangle around it. A message at the bottom says 'This credential domain is empty. How about adding some credentials?'.

Kind	Description
This credential domain is empty. How about adding some credentials?	

Credential 등록 – dockerhub access token 등록 (계속)

New credentials

Kind

Username with password

Scope ?
Global (Jenkins, nodes, items, all child items, etc)

Username ?
dockerhub-id (도커 아이디)

Treat username as secret ?

Password ?
..... (도커 접근 토큰)

ID ?
dockerhub-token (크레덴셜 식별자)

Description ?

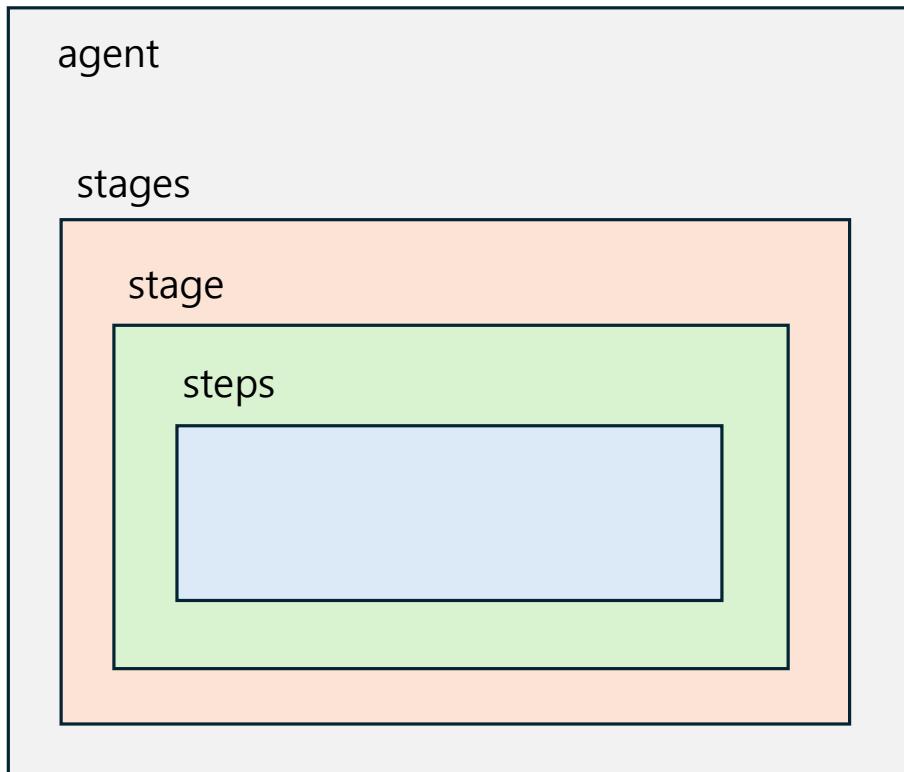
Create

This screenshot shows the Jenkins 'New credentials' configuration interface. The 'Kind' dropdown is set to 'Username with password'. The 'Scope' is set to 'Global'. The 'Username' field contains 'dockerhub-id' with the label '(도커 아이디)'. The 'Password' field is redacted with '.....' and has the label '(도커 접근 토큰)'. The 'ID' field contains 'dockerhub-token' with the label '(크레덴셜 식별자)'. A 'Create' button is at the bottom.

Pipeline

파이프라인 기본 구조

pipeline



```
pipeline {  
    agent any  
  
    stages {  
  
        stage('stage name') {  
  
            steps {  
                ...  
            }  
        }  
    }  
}
```

파이프라인 문법 – pipeline

```
pipeline {  
    /* insert Declarative Pipeline here */  
}
```

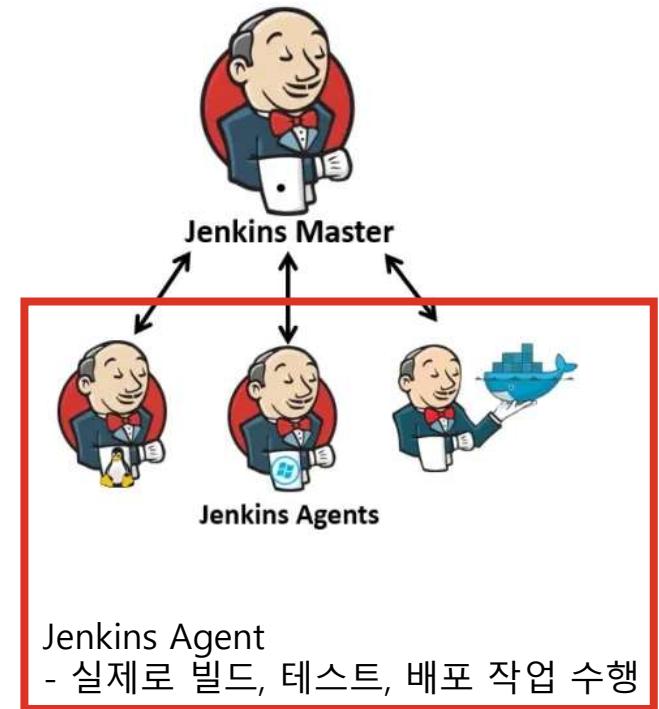
- 파이프라인을 정의하기 위해서는 반드시 pipeline을 포함해야 한다.
 - pipeline은 최상위 레벨이 되어야하며, {}으로 정의해야 한다.
 - pipeline은 Section, Directives, Steps 또는 assignment 문으로 구성되어야 한다.
-
- Section : agent, post, stages, steps
 - Directives: environment, options, parameters, triggers, tools

파이프라인 문법 – (section) agent

```
pipeline {  
    agent {  
        node {  
            label 'master'  
        }  
    }  
}
```

master인 노드에서 실행

- Jenkins 환경에서 전체 Pipeline이나 특정 stage가 실행될 위치를 지정
- pipeline {} 블록 내의 최상단에 정의
- 각 stage {} 블록의 최상단에도 정의가 가능하지만, 필수 사항은 아님
- agent any: 사용 가능한 agent.
- agent none: global agent는 설정 x. 각 stage에 설정 필요
- agent { node { ... } }



파이프라인 문법 – (section) stages

```
pipeline {  
    agent any  
  
    stages {  
        stage('Build') {  
            ...  
        }  
  
        stage('Test') {  
            ...  
        }  
    }  
}
```

- 하나 이상의 stage를 포함하는 컨테이너
- Pipeline이 말하는 work의 대부분이 위치하는 곳

파이프라인 문법 – (section) steps

```
pipeline {  
    agent any  
  
    stages {  
        stage('Build & Test Application') {  
            steps {  
                sh "gradle clean build"  
            }  
        }  
    }  
}
```

- stage 내부 실제 실행 명령을 정의

파이프라인 문법 – (section) post

```
pipeline {  
    agent any  
  
    stages {  
        stage('Build & Test Application') {  
            steps {  
                sh "gradle clean build"  
            }  
        }  
    }  
  
    post {  
        success { slackSend channel: '#deploy', message: 'Success!' }  
        failure { mail to: 'team@example.com', subject: 'Pipeline Failed!' }  
    }  
}
```

- 파이프라인/스테이지 실행 후 조건부 작업을 정의

파이프라인 문법 – (directives) environment

```
pipeline {  
    agent {  
        node {  
            label 'master'  
        }  
    }  
  
    environment {  
        GIT_URL = "https://github.com/solarhc/k8s-backend-user.git"  
        GITHUB_CREDENTIAL = "github-token"  
        DOCKERHUB_CREDENTIAL = 'dockerhub-token'  
    }  
    ...  
}
```

- key-value 형태로 파이프라인 내부에서 사용할 환경 변수로 선언

파이프라인 문법 – (directives) parameters

```
pipeline {
    agent {
        node {
            label 'master'
        }
    }

    parameters {
        booleanParam defaultValue: false, description: "", name: 'RELEASE'
    }
}
```

- 사용자 입력 매개변수를 정의

파이프라인 문법 – (directives) options

```
pipeline {  
    ...  
  
    options {  
        disableConcurrentBuilds()  
        buildDiscarder(logRotator(numToKeepStr: "30", artifactNumToKeepStr: "30"))  
        timeout(time: 120, unit: 'MINUTES')  
    }  
    ...  
}
```

- 파이프라인 전역 옵션을 설정

파이프라인 문법 – (directives) tools

```
pipeline {  
    agent {  
        node {  
            label 'master'  
        }  
    }  
    ...  
    tools {  
        gradle 'Gradle 8.14.2'  
        jdk 'OpenJDK 17'  
        dockerTool 'Docker'  
    }  
}
```

- 파이프라인에서 빌드 도구의 자동 설치 및 환경 구성은 지원하는 지시문
- 주로 JDK, Maven, Gradle 등 빌드에 필요한 도구를 지정
- Jenkins의 Global Tool Configuration에서 미리 정의된 도구만 사용 가능

파이프라인 문법 – (directives) when

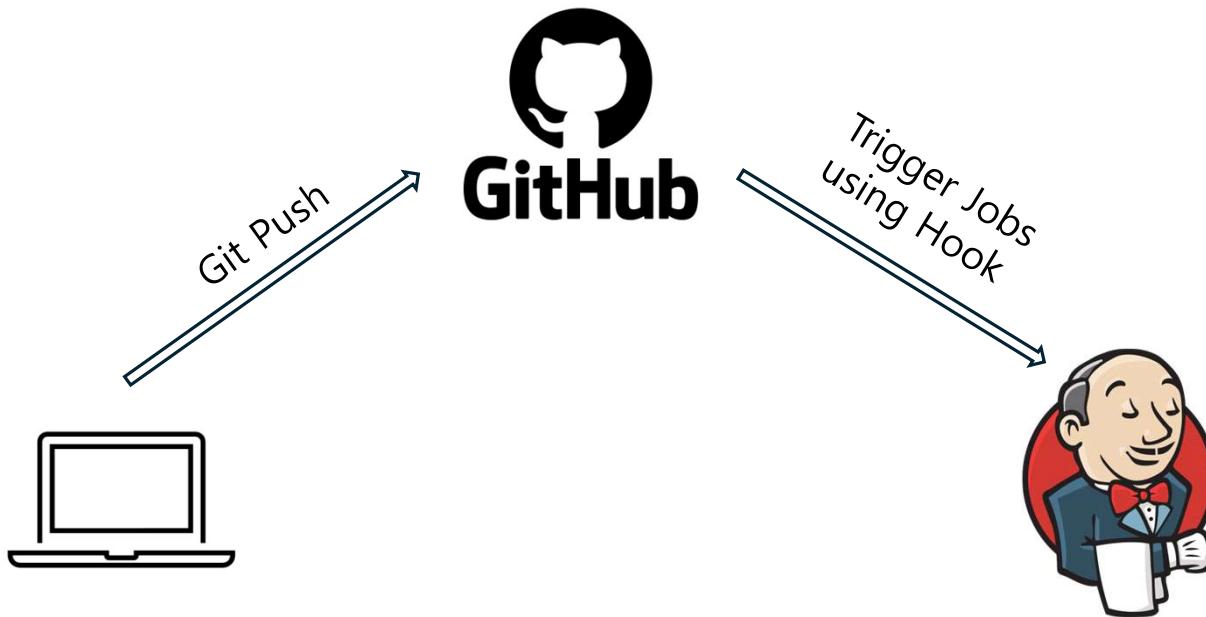
```
pipeline {
    ...
    stages {
        stage('Deliver to Prod Repository') {
            when {
                // 조건문 (예: 브랜치, 환경 변수, 파라미터 등)
                expression { params.ENV == 'prod' }
            }
            steps {
                script {
                    docker.withRegistry("", DOCKERHUB_CREDENTIALAL) {
                        docker.image("${DOCKER_IMAGE_NAME}").push()
                    }
                }
            }
        }
        ...
    }
}
```

실행 환경이 'prod'인 경우, docker image를 push하도록 실행

- 특정 조건이 충족될 때만 Stage를 실행하도록 제어하는 지시문
- 조건부 빌드, 환경별 배포, 브랜치/변수 기반 분기 처리에 유용

빌드 트리거

빌드 트리거 – Push Notification 방식

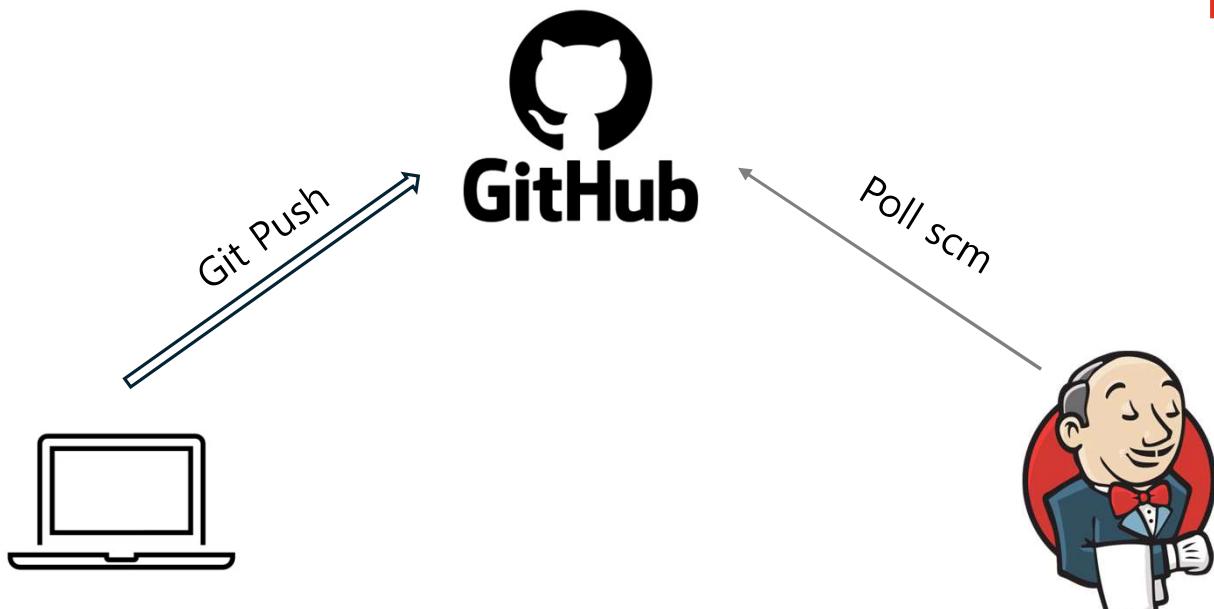


Jenkins 트리거

- Generic Webhook Trigger
- Github hook trigger for GITScm polling

주의) Jenkins 서버가 Github에서 접근 가능
해야 함 (공개 IP)

빌드 트리거 – Polling 방식



Jenkins 트리거
- Poll SCM

- 주기적으로 소스 코드 저장소를 폴링하여 변경사항을 감지하고, 변경이 있을 경우 빌드 트리거 하는 방식
- 크론(Cron) 방식으로 스케줄 설정
- 웹훅 설정이 불가능한 환경에서 사용

()

실습

프로젝트 생성 – User project

<https://start.spring.io/> 접속

Artifact: k8s-backend-user

The screenshot shows the Spring Initializr web interface with the following configuration:

- Project:** Maven (selected)
- Language:** Java (selected)
- Spring Boot:** 3.5.0 (selected)
- Dependencies:**
 - Spring Web** (WEB): Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.
 - Lombok** (DEVELOPER TOOLS): Java annotation library which helps to reduce boilerplate code.
- Project Metadata:**
 - Group: com.welab
 - Artifact: k8s-backend-user
 - Name: k8s-backend-user
 - Description: Demo project for Spring Boot
 - Package name: com.welab.k8s-backend-user
 - Packaging: Jar (selected)
 - Java: 17 (selected)

Create & Publish Repository

The screenshot shows a GitHub desktop application interface. On the left, there's a list of files with one change: "src...K8sBackendUserApp.java". The main area displays a "Create a new repository" dialog. The repository details are as follows:

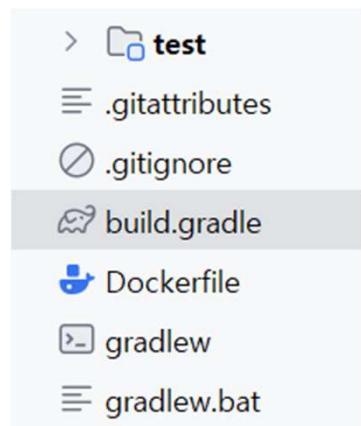
- Name:** k8s-backend-user
- Description:** (empty)
- Local path:** C:\Workspace\k8s
- Initialize this repository with a README:**
- Git ignore:** None
- License:** None

A note at the bottom states: "The repository will be created at C:\Workspace\k8s\test\k8s-backend-user." Below the dialog are two buttons: "Create repository" (highlighted in blue) and "Cancel".

To the right of the dialog, a modal window titled "No local changes" provides suggestions:

- Publish your repository to GitHub:** This repository is currently only available on your local machine. By publishing it on GitHub you can share it, and collaborate with others. [Publish repository](#)
- Open the repository in your external editor:** Select your editor in [Options](#). Repository menu or [Ctrl + Shift + A](#). [Open in Visual Studio Code](#)
- View the files of your repository in Explorer:** Repository menu or [Ctrl + Shift + F](#). [Show in Explorer](#)

User 프로젝트 - build.gradle 설정



settings.gradle
rootProject.name

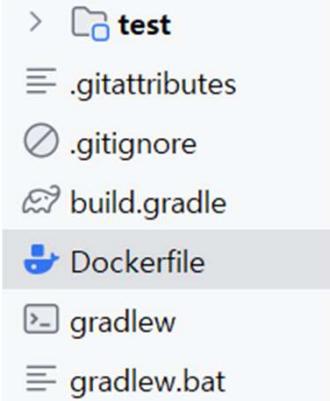
```
tasks.named('test') {
    useJUnitPlatform()
}

jar {
    enabled = false // plain.jar 생성 완전히 비활성화
}

tasks.register('getAppName') {
    doLast {
        println "${rootProject.name}"
    }
}

tasks.register('getAppVersion') {
    doLast {
        println "${project.version}"
    }
}
```

User 프로젝트 - Dockerfile 작성



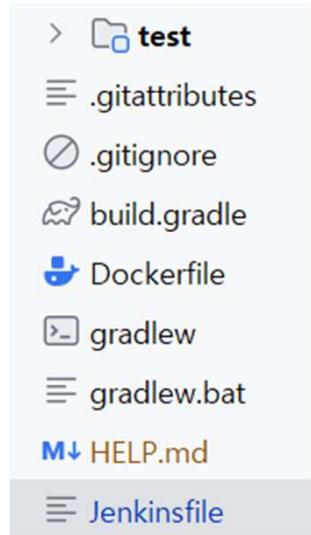
```
FROM amazoncorretto:17
MAINTAINER dev@welab.com
VOLUME /tmp
EXPOSE 8080
COPY build/libs/*.jar /app.jar
ENTRYPOINT ["java", "-Djava.security.egd=file:/dev/.urandom", "-jar", "/app.jar"]
```

User 프로젝트 - Jenkinsfile 작성

```
> test
≡ .gitattributes
🚫 .gitignore
⚡ build.gradle
/dockerfile
gradlew
≡ gradlew.bat
M HELP.md
≡ Jenkinsfile
```

```
def APP_NAME
def APP_VERSION
def DOCKER_IMAGE_NAME
def PROD_BUILD = false
pipeline {
    agent {
        node {
            label 'master'
        }
    }
    parameters {
        gitParameter branch: '',
            branchFilter: '.*',
            defaultValue: 'origin/main',
            description: '',
            listSize: '0',
            name: 'TAG',
            quickFilterEnabled: false,
            selectedValue: 'DEFAULT',
            sortMode: 'DESCENDING_SMART',
            tagFilter: '*',
            type: 'PT_BRANCH_TAG'
    }
    booleanParam defaultValue: false, description: '', name: 'RELEASE'
}
```

User 프로젝트 - Jenkinsfile 작성 (계속)



■ : 자신의 것으로 수정 必

```
environment {
    GIT_URL = "https://github.com/solarho/k8s-backend-user.git"
    GITHUB_CREDENTIAL = "github-token"
    ARTIFACTS = "build/libs/**"
    DOCKER_REGISTRY = "solarho"
    DOCKERHUB_CREDENTIAL = 'dockerhub-token'
}

options {
    disableConcurrentBuilds()
    buildDiscarder(logRotator(numToKeepStr: "30", artifactNumToKeepStr: "30"))
    timeout(time: 120, unit: 'MINUTES')
}

tools {
    gradle 'Gradle 8.14.2'
    jdk 'OpenJDK 17'
    dockerTool 'Docker'
}
```

User 프로젝트 - Jenkinsfile 작성 (계속)

```
> test
≡ .gitattributes
🚫 .gitignore
build.gradle
Dockerfile
gradlew
≡ gradlew.bat
M HELP.md
≡ Jenkinsfile
```

```
stages {
    stage('Set Version') {
        steps {
            script {
                APP_NAME = sh (
                    script: "gradle -q getAppName",
                    returnStdout: true
                ).trim()
                APP_VERSION = sh (
                    script: "gradle -q getAppVersion",
                    returnStdout: true
                ).trim()
                DOCKER_IMAGE_NAME = "${DOCKER_REGISTRY}/${APP_NAME}:${APP_VERSION}"
                sh "echo IMAGE_NAME is ${APP_NAME}"
                sh "echo IMAGE_VERSION is ${APP_VERSION}"
                sh "echo DOCKER_IMAGE_NAME is ${DOCKER_IMAGE_NAME}"
            }
        }
    }
}
```

build.gradle에서 정의한 명령
(출력 형식으로 반환됨)

User 프로젝트 - Jenkinsfile 작성 (계속)

```
> test
  .gitattributes
  .gitignore
  build.gradle
  Dockerfile
  gradlew
  gradlew.bat
  HELP.md
  Jenkinsfile
```

```
stage('Build & Test Application') {
    steps {
        sh "gradle clean build"
    }
}

stage('Build Docker Image') {
    steps {
        script {
            docker.build "${DOCKER_IMAGE_NAME}"
        }
    }
}

stage('Push Docker Image') {
    steps {
        script {
            docker.withRegistry("", DOCKERHUB_CREDENTIAL) {
                docker.image("${DOCKER_IMAGE_NAME}").push()
            }
            sh "docker rmi ${DOCKER_IMAGE_NAME}"
        }
    }
}
```

Jenkins – Backend Folder 생성

The Jenkins dashboard is shown. At the top left is the Jenkins logo. Below it is a large red-bordered button labeled '+ 새로운 Item'. To its right are three other items: '빌드 기록', '프로젝트 연관 관계', and '파일 핑거프린트 확인'. Below these are two sections: '빌드 대기 목록' and '빌드 실행 상태'. The '빌드 실행 상태' section shows two builds: 'Backend » k8s-backend-user' (status: #17, Push Docker Image) and 'Backend » k8s-backend-user' (status: #17). Both builds have a progress bar at the bottom.

The 'New Item' dialog is open. At the top is a text input field labeled 'Enter an item name' containing 'Backend'. Below it is a section titled 'Select an item type' with five options: 'Freestyle project', 'Pipeline', 'Multi-configuration project', 'Folder', and 'Multibranch Pipeline'. The 'Folder' option is highlighted with a red border. Its description states: 'Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.' Below the 'Folder' option are descriptions for 'Multibranch Pipeline' and 'Organization Folder'.

Jenkins – k8s-backend-user 파이프라인 생성

The screenshot shows the Jenkins web interface. On the left, there's a sidebar with icons for Status, Configure, New Item (which is highlighted with a red box), Delete Folder, and Build History. The main area shows a folder named "Backend". A modal window titled "New Item" is open, prompting the user to "Enter an item name" with the value "k8s-backend-user" highlighted by a red box. Below this, there are three options for "Select an item type": "Freestyle project", "Pipeline", and "Multi-configuration project". The "Pipeline" option is also highlighted with a red box. Descriptions for each type are provided in Korean.

Status

Configure

+ New Item

Delete Folder

빌드 기록

Backend

New Item

Enter an item name

k8s-backend-user

Select an item type

Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
다양한 환경에서의 테스트, 플래폼 특성 빌드, 기타 등등 처럼 다수의 서로다른 환경설정이 필요한 프로젝트에 적합함.

Jenkins – k8s-backend-user 파이프라인 설정

General

Enabled 

설명

Plain text [미리보기](#)

Do not allow concurrent builds
 Abort previous builds ?

Do not allow the pipeline to resume if the controller restarts

GitHub project

Pipeline speed/durability override ?

Preserve stashes from completed builds ?

Throttle builds ?

Jenkins – k8s-backend-user 파이프라인 설정 (계속)

오래된 빌드 삭제 ?

Strategy

Log Rotation

▼

빌드 이력 유지 기간(일)
공백일 경우, [보관할 최대갯수] 만큼 기록됩니다.

보관할 최대갯수
if not empty, only up to this number of build records are kept

30

고급 ▾

Edited

Jenkins – k8s-backend-user 파이프라인 설정 (계속)

이 빌드는 매개변수가 있습니다 ?

☰ Git Parameter ?

Name ?
TAG

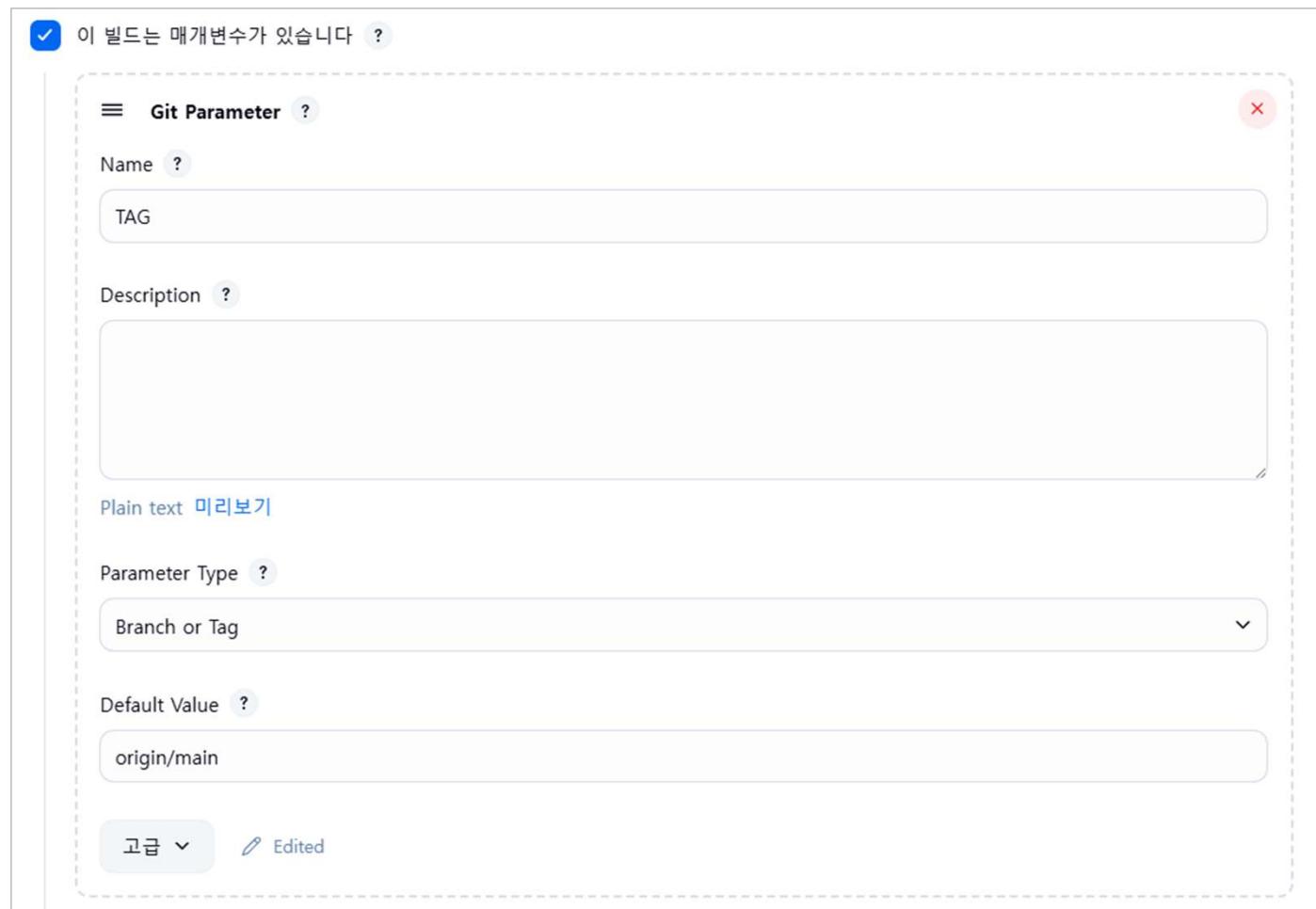
Description ?

Plain text [미리보기](#)

Parameter Type ?
Branch or Tag

Default Value ?
origin/main

고급 ▾ Edited



Jenkins – k8s-backend-user 파이프라인 설정 (계속)

≡ Boolean Parameter ? ×

Name ?
RELEASE

Set by Default ?

Description ?

Plain text 미리보기

매개변수 추가 ▾

Jenkins – k8s-backend-user 파이프라인 설정 (계속)

Triggers

Set up automated actions that start your build based on specific events, like code changes or scheduled times.

- Build after other projects are built [?](#)
- Build periodically [?](#)
- Generic Webhook Trigger [?](#)
- GitHub hook trigger for GITScm polling [?](#)
- Poll SCM [?](#)
- 빌드를 원격으로 유발 (예: 스크립트 사용) [?](#)

Jenkins – k8s-backend-user 파이프라인 설정 (계속)

Pipeline

Define your Pipeline using Groovy directly or pull it from source control.

Definition

Pipeline script from SCM

SCM ?

Git

Repositories ?

Repository URL ?

자신의 repository url 입력

Credentials ?

자신의 git hub credentials 선택

+ Add

고급 ▾

Add Repository

The screenshot shows the Jenkins Pipeline configuration page. It's set to 'Pipeline script from SCM' and 'Git'. Under 'Repositories', there's one entry with 'Repository URL' as 'https://github.com/solarhc/k8s-backend-user.git' and 'Credentials' as 'solarhc@naver.com/********'. Both the URL and credentials fields are highlighted with red boxes, with red Korean annotations '자신의 repository url 입력' and '자신의 git hub credentials 선택' placed next to them. Other UI elements like 'Add' and '고급' are also visible.

Jenkins – k8s-backend-user 파이프라인 설정 (계속)

Branches to build ?

Branch Specifier (blank for 'any') ?

**/main X

Add Branch

Repository browser ?

(자동) ▼

Additional Behaviours

Add ▼

Script Path ?

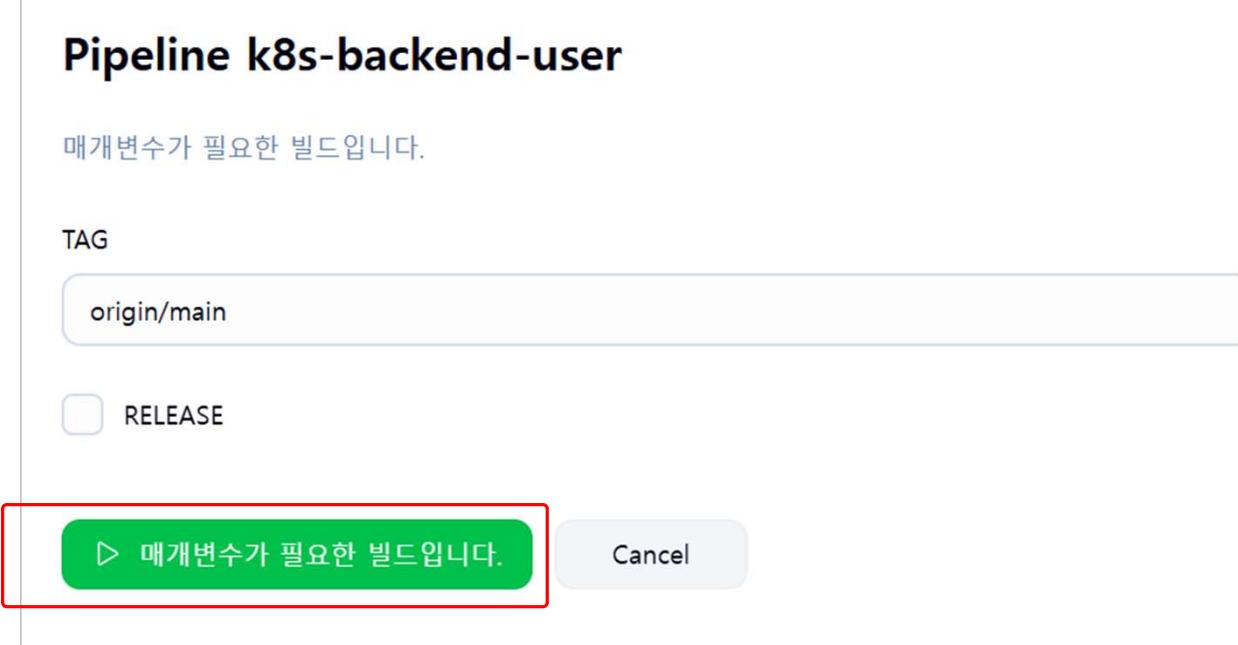
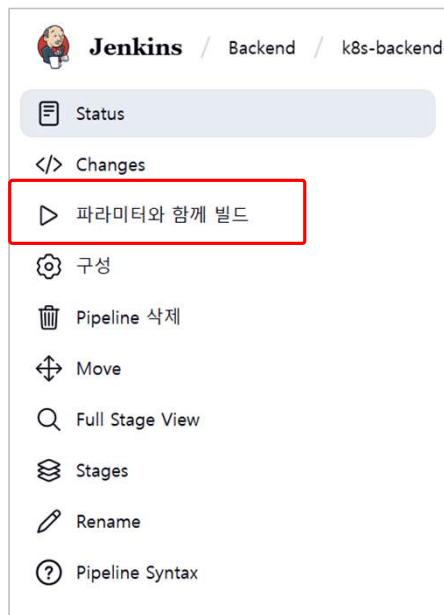
Jenkinsfile

Lightweight checkout ?

Pipeline Syntax

This screenshot shows the Jenkins Pipeline configuration screen. It includes fields for specifying branches to build (with a placeholder for 'any' and a specific entry for '**/main'), adding additional branches, selecting a repository browser (set to '자동' or automatic), defining additional behaviors (with an 'Add' button), setting the script path to 'Jenkinsfile', and enabling lightweight checkout.

Jenkins 파이프라인 빌드



조건부 Stage 실행

Jenkinsfile

```
def PROD_BUILD = false
def TAG_BUILD = false
pipeline {
    ...
    stages {
        stage('Set Version') {
            steps {
                script {
                    ...
                    if( params.TAG.startsWith('origin') == false && params.TAG.endsWith('/main') == false ) {
                        if( params.RELEASE == true ) {
                            DOCKER_IMAGE_NAME += '-RELEASE'
                            PROD_BUILD = true
                        } else {
                            DOCKER_IMAGE_NAME += '-TAG'
                            TAG_BUILD = true
                        }
                    }
                }
            }
        }
        ...
        stage('Build Docker Image') {
            when {
                expression { PROD_BUILD == true || TAG_BUILD == true }
            }
            steps {
                script {
                    docker.build "${DOCKER_IMAGE_NAME}"
                }
            }
        }
    }
}
```

Main branch가 아닌 TAG를 지정하여 빌드하는 경우
PROD_BUILD 또는 TAG_BUILD 가 true로 설정됨

PROD_BUILD 또는 TAG_BUILD 가 true 인 경우에만 docker build를 수행함

Jenkins Pipeline 구성 변경

위치: Jenkins Pipeline 구성 > Pipeline > SCM > Branches to build > Branch Specifier



Branch Specifier를 */main에서 \${TAG}로 변경

조건부 Stage 실행 결과

Stage View

	Declarative: Checkout SCM	Declarative: Tool Install	Set Version	Build & Test Application	Build Docker Image	Push Docker Image
Average stage times: (full run time: ~2min 17s)						
#10 6월 09 일 19:02	2s	2s	39s	51s	12s	18s
#9 6월 09 일 19:00	2s	2s	39s	49s	13s	21s
#8 6월 09 일 18:53	3s	2s	38s	49s		
	2s	2s	33s	47s	11s	20s

TAG 지정 빌드

TAG

0.0.2_2025-06-09

RELEASE

▷ 매개변수가 필요한 빌드입니다.

main branch 빌드

TAG

origin/main

RELEASE

▷ 매개변수가 필요한 빌드입니다.