


8. Pipeline 사용

1. item 생성


Enter an item name

My-First-Pipeline


» Required field

 **Freestyle project**


이것은 Jenkins의 주요 기능입니다. Jenkins은 어느 빌드 시스템과 어떤 SCM(형상관리)으로 묶인 당신의 프로젝트를 빌드할 것이고, 소프트웨어 빌드보다 다른 어떤 것에 자주 사용될 수 있습니다.

 **Maven project**


Maven 프로젝트를 빌드합니다. Jenkins은 POM 파일의 이점을 가지고 있고 급격히 설정을 줄입니다.

 **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, 플랫폼 특성 빌드, 기타 등등 처럼 다수의 서로다른 환경설정이 필요한 프로젝트에 적합함.

 **Folder**

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.

 **Multibranch Pipeline**

Creates a set of Pipeline projects according to detected branches in one SCM repository.

 **Organization Folder**

Creates a set of multibranch project subfolders by scanning for repositories

OK

2. pipeline 생성 및 빌드 테스트

- 2-1)

```
pipeline {  
    agent any  
    stages {  
        stage('Compile') {
```

```

    steps {
        echo "Compiled successfully!";
    }
}

stage('JUnit') {
    steps {
        echo "JUnit passed successfully!";
    }
}

stage('Code Analysis') {
    steps {
        echo "Code Analysis completed successfully!";
    }
}

stage('Deploy') {
    steps {
        echo "Deployed successfully!";
    }
}
}
}

```

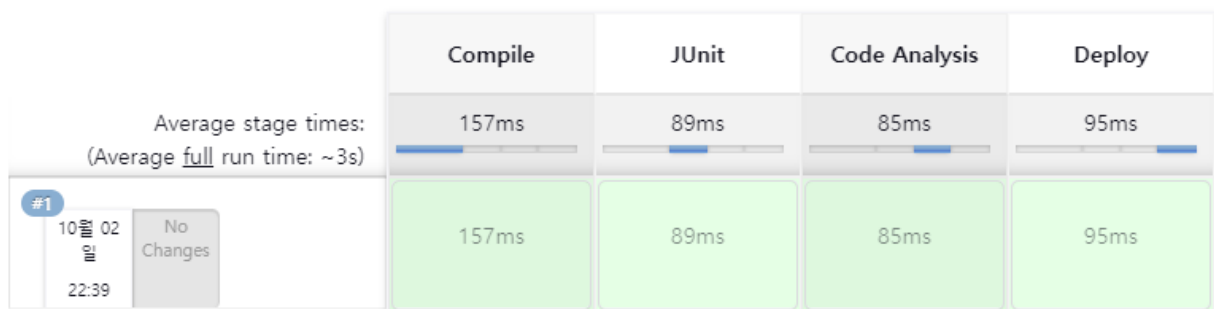
- 빌드 테스트

Pipeline My-First-Pipeline

 상세 내용 입력

프로젝트 중지하기

Stage View



- 2-2) post 단계 추가

```
pipeline {
    agent any
    stages {
        stage('Compile') {
            steps {
                echo "Compiled successfully!";
            }
        }

        stage('JUnit') {
            steps {
                echo "JUnit passed successfully!";
            }
        }

        stage('Code Analysis') {
            steps {
                echo "Code Analysis completed successfully!";
            }
        }

        stage('Deploy') {
            steps {
                echo "Deployed successfully!";
            }
        }
    }

    post {
        always {
            echo "This will always run"
        }
        success {
            echo "This will run when the run finished successfully"
        }
        failure {
            echo "This will run if failed"
        }
        unstable {
            echo "This will run when the run was marked as unstable"
        }
        changed {
            echo "This will run when the state of the pipeline has changed"
        }
    }
}
```

```
}  
}  
}
```

- 2-2) 실행 결과

Stage Logs (Declarative: Post Actions)

☑ Print Message -- This will always run (self time 32ms)

☑ Print Message -- This will run when the run finished successfully (self time 22ms)

3. maven build pipeline

- 새로운 item 생성

- pipeline syntax 사용하여 script 생성 (repo가 public 설정되어 있어서 계정 정보 X)

Pipeline Syntax

Sample Step

git: Git

git ?

Repository URL ?

https://github.com/aheeej/cicd-web-project.git

Branch ?

master

Credentials ?

- none -

Add ▾

☒ Include in polling? ?

☒ Include in changelog? ?

Generate Pipeline Script

git 'https://github.com/aheeej/cicd-web-project.git'

- pipeline 생성

```
pipeline {  
  agent any  
  tools {
```

```
    maven 'Maven3.9.4'
  }
  stages {
    stage('github clone') {
      steps {
        git 'https://github.com/aheeej/cicd-web-project.git'
      }
    }
    stage('build'){
      steps {
        sh '''
          echo build start
          mvn clean compile package -DskipTests=true
          ...
        '''
      }
    }
  }
}
```

- 빌드 테스트 정상 성공

- docker에 war파일 ssh로 배포 단계 추가를 위해 pipeline syntax에서 스크립트 확인

sshPublisher ?

SSH Publishers

SSH Server

Name ?

docker-server

고급 ▾

Transfers

Transfer Set

Source files ?

target/*.war

Remove prefix ?

target

Remote directory ?

.

Exec command ?

docker build --tag aheeej/exam_01 -f Dockerfile .

All of the transfer fields (except for Exec timeout) support substitution of [Jenkins environment variables](#)

고급 ▾

- 파이프라인 추가