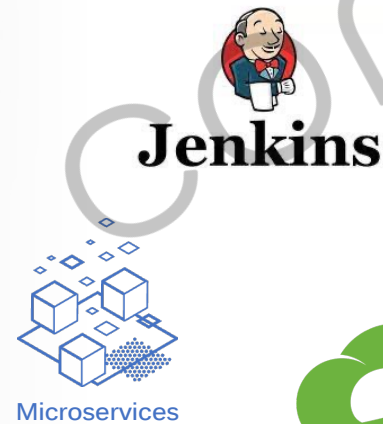


Jenkins를 이용한 CI/CD Pipeline 구축



```
class Book {
    private String title;
    private double price;
    private String author;

    public Book(String title, double price, String author) {
        this.title = title;
        this.price = price;
        this.author = author;
    }

    public String getTitle() {
        return title;
    }

    public double getPrice() {
        return price;
    }

    public String getAuthor() {
        return author;
    }
}

import java.io.*;
import java.util.*;

public class BookServlet {
    private Book book;

    public BookServlet() {
        this.book = new Book("Jenkins", 10.0, "Kenneth Lee");
    }

    public void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<h1> Jenkins CI/CD Pipeline </h1>");
    }

    public void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        // TODO: Implement POST logic
    }
}

@WebServlet("/book")
public class BookController extends HttpServlet {
    private BookServlet bookServlet;

    public BookController() {
        this.bookServlet = new BookServlet();
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        bookServlet.doGet(request, response);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        bookServlet.doPost(request, response);
    }
}

@InterfaceNextInnovationDelegate : NSObject <UIApplicationDelegate>
```

프로필

Dowon Lee



지식공유자 인증

5452 ☆ 4.8(420)

멘토링 활성



- 홈
- 강의
- 로드맵
- 수강후기
- 블로그

강의 (3)

최신순 ▼

20% 할인 (D-8)

Spring Cloud
with Microservices
Part 2

Spring Cloud로 개발하는 마이크로서비스 애플리케이션(MSA)

Dowon Lee

★★★★★ (174)

학습중

+3000명 독점 할인중

20% 할인 (D-8)

Spring Boot
with RESTful Web Services
Part 1

Spring Boot를 이용한 RESTful Web Services 개발

Dowon Lee

★★★★★ (308)

학습중

+2700명 독점 할인중

IntelliJ IDEA
Java Web Programming

웹 애플리케이션 개발을 위한 IntelliJ IDEA 설정

Dowon Lee

★★★★★ (228)

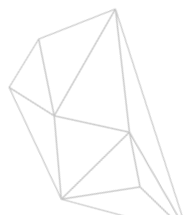
학습중

+3500명



목차

- Section 1: DevOps와 CI/CD
- Section 2: Jenkins를 이용한 CI/CD 사용
- Section 3: Jenkins + Infrastructure as Code
- Section 4: Jenkins + Ansible + Kubernetes 연동
- **Section 5: Advanced Jenkins 사용**
- Section 6: Public Cloud에 배포
- Appendix



Section 5.

Advanced Jenkins

- Delivery Pipeline 사용
- Jenkins Pipeline 구성
- SonarQube 사용
- Jenkins + SonarQube 연동
- Jenkins Master + Slaves 구성

Create a Pipeline

My-First-Project

- echo

My-Second-Project

- git
- maven build

My-Third-Project

- git
- maven build
- deploy on Tomcat

빌드 후 조치

Build other projects

Projects to build

My-Second-Project

- ☒ Trigger only if build is stable
- ☐ Trigger even if the build is unstable
- ☐ Trigger even if the build fails

빌드 후 조치 추가 ▾

빌드 후 조치

Build other projects

Projects to build

My-Third-Project

- ☒ Trigger only if build is stable
- ☐ Trigger even if the build is unstable
- ☐ Trigger even if the build fails

빌드 후 조치 추가 ▾

Create a Pipeline

njone company

- Manage Jenkins → Plugin Manager → Available
 - Delivery Pipeline

업데이트된 플러그인 목록

설치 가능

설치된 플러그인 목록

고급

Q Delivery Pipeline

Install

Name ↓

Delivery Pipeline 1.4.2

☐

User Interface

This plugin visualize Delivery Pipelines (Jobs with upstream/downstream dependencies)

Create a Pipeline

njone company

Dashboard >

+ 새로운 Item

👤 사람

📁 빌드 기록

🔗 프로젝트 연관 관계

🖨 파일 핑거프린트 확인

⚙ Jenkins 관리

👤 My Views

📁 새로운 뷰

빌드 대기 목록

빌드 대기 항목이 없습니다.

빌드 실행 상태

1 대기 중

2 대기 중

New view

조희명

My-First-Pipeline

Type

☒ Delivery Pipeline View

Continuous Delivery pipelines, perfect for visualization on information radiators. Shows one or more delivery pipeline instances, based on traditional Jenkins jobs with upstream/downstream dependencies.

☐ Delivery Pipeline View for Jenkins Pipelines

Continuous Delivery pipelines, perfect for visualization on information radiators. Shows one or more delivery pipeline instances, based on Jenkins pipelines (created using the Pipeline or Workflow plug

☐ List View

단순 목록 형식으로 작업을 보여줌. 조회에 표시될 작업을 선택할 수 있음.

☐ My View

이 조회는 현재 사용자가 접근하고 있는 모든 작업을 자동적으로 표시함.

Create

Pipelines

Components

Component

Name

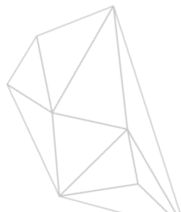
My-Pipeline

Initial Job

My-First-Project

Final Job (optional)

☐ Show upstream



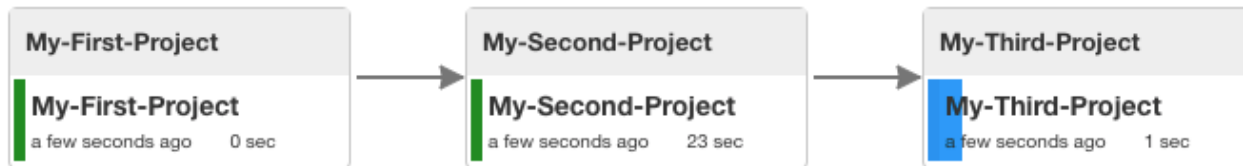
Create a Pipeline

njone company

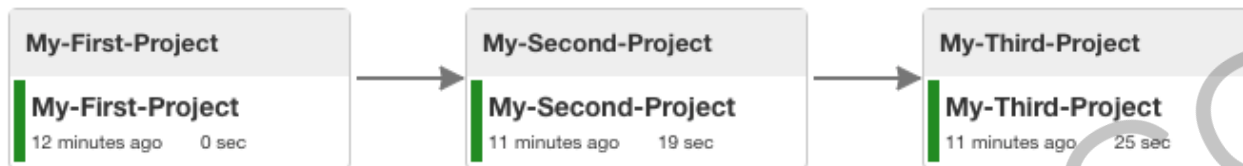
All My-First-Pipeline +

My-Pipeline

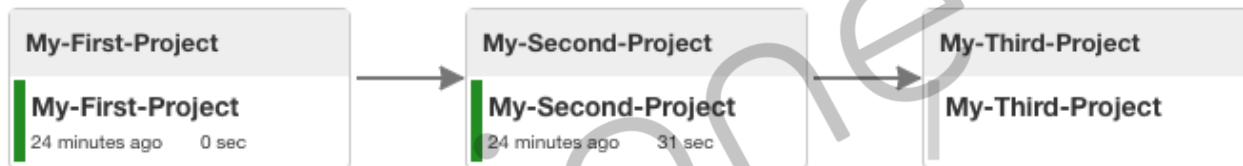
#8 triggered by user Administrator started a few seconds ago



#7 triggered by user Administrator started 12 minutes ago



#6 triggered by user Administrator started 24 minutes ago



Jenkins Pipeline

njone company

- Declarative
- Scripted (Groovy + DSL)
- 차이점
 - 시작 시 유효성 검사 유무
 - 특정 Stage 실행 가능 여부
 - 제어문
 - Option

```
pipeline {  
    agent any  
    stages {  
        steps('build') {  
            //  
        }  
  
        steps('test') {  
            //  
        }  
  
        steps('deploy') {  
            //  
        }  
    }  
}
```

```
node {  
    stage('build') {  
        //  
    }  
  
    stage('test') {  
        //  
    }  
  
    stage('deploy') {  
        //  
    }  
}
```

Jenkinsfile

Jenkins Pipeline

njone company

■ Declarative

- Groovy script 없이 간단하게 시작

```
pipeline {
```

```
  agent any
```

```
  stages {
```

```
    steps('build') {
```

```
      //
```

```
    steps('test') {
```

```
      //
```

```
    steps('deploy') {
```

```
      //
```

```
  }
```

```
}
```

```
}
```

→ 실행가능한 Agent에서 Pipeline 실행

→ build 스테이지 선언

→ build 스테이지에 필요한 작업을 수행

→ test 스테이지 선언

→ test 스테이지에 필요한 작업을 수행

→ deploy 스테이지 선언

→ deploy 스테이지에 필요한 작업을 수행

Exercise 9# Jenkins Pipeline Job 1/4

njone company

- Item name → ***My-First-Pipeline***

- Pipeline
 - Script



General Build Triggers Advanced Project Options Pipeline

설명

My first pipeline using Declarative

[Plain text] [미리보기](#)

Pipeline

Definition

Pipeline script

Script

```
1 pipeline {
2   agent any
3   stages {
4     stage('Compile') {
5       steps {
6         echo "Compiled successfully!";
7       }
8     }
9
10    stage('JUnit') {
11      steps {
12        echo "JUnit passed successfully!";
13      }
14    }
15
16    stage('Code Analysis') {
17      steps {
18        echo "Code Analysis completed successfully!";
19      }
20    }
21
22    stage('Deploy') {
23      steps {
24        echo "Deployed successfully!";
25      }
26    }
27  }
28 }
```

Exercise 9# Jenkins Pipeline Job 2/4

njone company

■ Build Now

Pipeline My-First-Pipeline

My first pipeline using Declarative



Recent Changes

Stage View

Average stage times:
(Average full run time: ~1s)

Compile	JUnit	Code Analysis
58ms	33ms	48ms
58ms	33ms	48ms

고정링크

Stage Logs (Compile)

Print Message -- Compiled successfully! (self time 6ms)

Compiled successfully!

Status

Changes

Build Now

구성

Pipeline 삭제

Full Stage View

Rename

Pipeline Syntax

Build History

My first pipeline using Declarative



Recent Changes

Stage View

Average stage times:
(Average full run time: ~1s)

#1
Oct 06
10:18
No
Changes

Success
Compile

Logs

58ms

Exercise 9# Jenkins Pipeline Job 3/4

njone company

- Console Output

✓ 콘솔 출력

```
Started by user Administrator
Running in Durability level: MAX_SURVIVABILITY
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/jenkins_home/workspace/My-First-Pipeline
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Compile)
[Pipeline] echo
Compiled successfully!
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (JUnit)
[Pipeline] echo
```

```
Deployed successfully!
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Exercise 9# Jenkins Pipeline Job 4/4

njone company

■ Script 추가

- *post*

```
Deployed successfully!  
[Pipeline] }  
[Pipeline] // stage  
[Pipeline] stage  
[Pipeline] { (Declarative: Post Actions)  
[Pipeline] echo  
This will always run  
[Pipeline] echo  
This will run when the run finished successfully  
[Pipeline] }  
[Pipeline] // stage  
[Pipeline] }  
[Pipeline] // node  
[Pipeline] End of Pipeline  
Finished: SUCCESS
```

Pipeline

Definition

Pipeline script

Script

```
22 stage('Deploy') {  
23     steps {  
24         echo "Deployed successfully!";  
25     }  
26 }  
27  
28  
29  
30 post {  
31     always {  
32         echo 'This will always run'  
33     }  
34     success {  
35         echo 'This will run when the run finished successfully'  
36     }  
37     failure {  
38         echo 'This will run if failed'  
39     }  
40     unstable {  
41         echo 'This will run when the run was marked as unstable'  
42     }  
43     changed {  
44         echo 'This will run when the state of the pipeline has changed'  
45     }  
46 }
```

Exercise 10# Jenkins Pipeline Job 1/3

njone company

■ Github에 저장된 Script 실행

- https://github.com/joneconsulting/jenkins_pipeline_script

```
▶ ls *.bat
build.bat  deploy.bat  quality.bat  unit.bat
(base) downlee ~/Desktop/git/jenkins_pipeline_script ▶ master
▶ ls *.sh
build.sh  deploy.sh  quality.sh  unit.sh
(base) downlee ~/Desktop/git/jenkins_pipeline_script ▶ master
```

■ Pipeline 추가

- **Pipeline Syntax** 이용

```
Script
1 pipeline {
2   agent any
3   stages {
4     stage('Git clone') {
5       steps {
6         git 'https://github.com/joneconsulting/jenkins_pipeline_script';
7       }
8     }
9   }
}
```

Pipeline Syntax

Overview

This Snippet Generator will help you learn the Pipeline Script code which can be **Generate Pipeline Script**, and you will see a Pipeline Script statement that would script, or pick up just the options you care about. (Most parameters are optional a

Steps

Sample Step

git: Git

git

Repository URL

Branch

Credentials

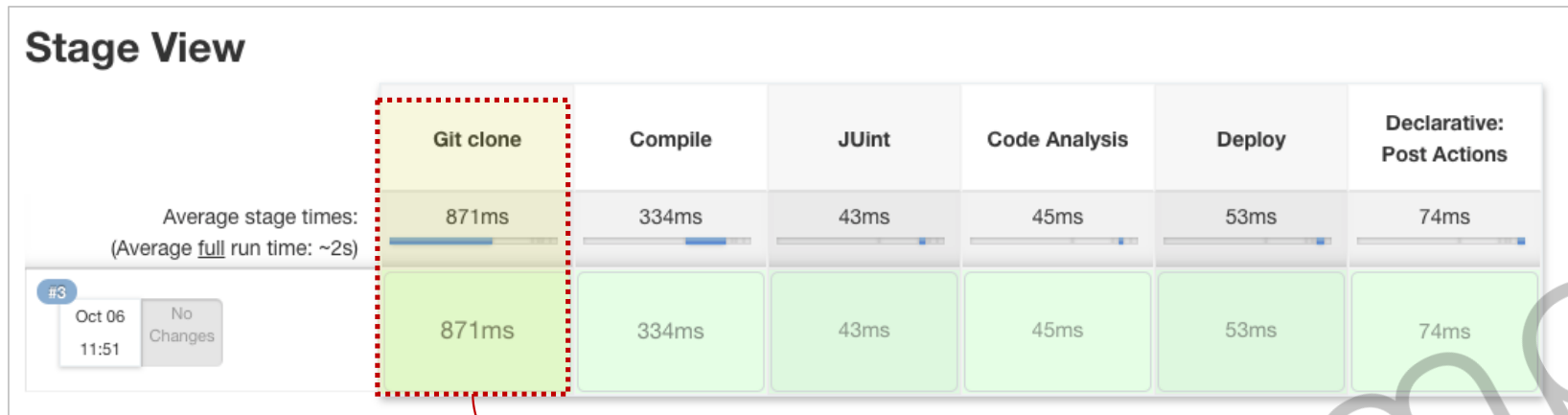
☒ Include in polling?

☒ Include in changelog?

Exercise 10 # Jenkins Pipeline Job 2/3

njone company

■ Build Now



```
Cloning the remote Git repository
Cloning repository https://github.com/joneconsulting/jenkins_pipeline_script
> git init /var/jenkins_home/workspace/My-First-Pipeline # timeout=10
Fetching upstream changes from https://github.com/joneconsulting/jenkins_pipeline_script
> git --version # timeout=10
> git --version # 'git version 2.30.2'
> git fetch --tags --force --progress -- https://github.com/joneconsulting/jenkins_pipeline_script
timeout=10
> git config remote.origin.url https://github.com/joneconsulting/jenkins_pipeline_script # timeout=
> git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision 5552317c4dbf21a896b27e707dcbe85168273377 (refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout -f 5552317c4dbf21a896b27e707dcbe85168273377 # timeout=10
> git branch -a -v --no-abbrev # timeout=10
> git checkout -b master 5552317c4dbf21a896b27e707dcbe85168273377 # timeout=10
Commit message: "upload batch files"
First time build. Skipping changelog.
```


Exercise 10 # Jenkins Pipeline Job 3/3

njone company

■ Pipeline 수정 → Build Now

```
9 stage('Compile') {
10     steps {
11         echo "Compiled successfully!";
12         sh './build.sh'
13     }
14 }
15
16 stage('JUnit') {
17     steps {
18         echo "JUnit passed successfully!";
19         sh './unit.sh'
20     }
21 }
22
23 stage('Code Analysis') {
24     steps {
25         echo "Code Analysis completed successfully!";
26         sh './quality.sh'
27     }
28 }
29
30 stage('Deploy') {
31     steps {
32         echo "Deployed successfully!";
33         sh './deploy.sh'
34     }
35 }
36 }
```

```
JUnit passed successfully!
[Pipeline] sh
+ ./unit.sh
Running Unit Test Cases : Wed Oct 6 04:34:20 UTC 2021
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Code Analysis)
[Pipeline] echo
Code Analysis completed successfully!
[Pipeline] sh
+ ./quality.sh
Code Quality Check : Wed Oct 6 04:34:20 UTC 2021
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Deploy)
[Pipeline] echo
Deployed successfully!
[Pipeline] sh
+ ./deploy.sh
Deploying the Project : Wed Oct 6 04:34:21 UTC 2021
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Declarative: Post Actions)
[Pipeline] echo
This will always run
[Pipeline] echo
This will run when the state of the pipeline has changed
[Pipeline] echo
This will run when the run finished successfully
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Exercise 11 # Maven build

- 이전에 작업했던 cicd-web-project 예제를 Maven 빌드 하기

```
1 pipeline {  
2   agent any  
3   tools {  
4     maven 'Maven3.8.5'  
5   }  
6   stages {  
7     stage('github clone') {  
8       steps {  
9         git branch: 'main', url: 'https://github.com/joneconsulting/cicd-web-project.git';  
10      }  
11    }  
12  
13    stage('build') {  
14      steps {  
15        sh '''  
16          echo build start  
17          mvn clean compile package -DskipTests=true  
18        '''  
19      }  
20    }  
21  }
```

Exercise 12 # Deploy on Tomcat 1/2

njone company

- Exercise 11번 작업의 결과물을 tomcat9 서버에 배포하기

Sample Step

deploy: Deploy war/ear to a container

deploy

WAR/EAR files ?

**/*.war

Containers

Tomcat 9.x Remote

Credentials

deployer/***** (user to deploy on tomcat VM)

+ Add

Tomcat URL ?

http://10.48.9.201:8080

고급...

Exercise 12 # Deploy on Tomcat 2/2

njone company

- Exercise 11번 작업의 결과물을 tomcat9 서버에 배포하기

```
stage('deploy') {  
    steps {  
        deploy adapters: [tomcat9(credentialsId: 'deployer_user', path: '', url: 'http://10.48.9.201:8088/']], contextPath:  
null, war: '**/*.war'  
    }  
}
```



Exercise 13 # Deploy on Docker server 1/2

njone company

- Exercise 11번 작업의 결과물을 Docker server에 배포하기

SSH Publishers

SSH Server
Name ?

docker-server

고급...

Transfers

Transfer Set
Source files ?

target/*.war

Remove prefix ?

target

Remote directory ?

.

Exec command ?

```
docker build -t edowon0623/devops_exam1 -f Dockerfile .
```

Exercise 13 # Deploy on Docker server 2/2

njone company

- Exercise 11번 작업의 결과물을 Docker server에 배포하기

```
stage('ssh publisher') {  
    steps {  
        sshPublisher(publishers: [sshPublisherDesc(configName: 'docker-server', transfers: [sshTransfer(cleanRemote:  
false, excludes: '', execCommand: 'docker build -t edowon0623/devops_exam1 -f Dockerfile .', execTimeout: 120000,  
flatten: false, makeEmptyDirs: false, noDefaultExcludes: false, patternSeparator: '[, ]+', remoteDirectory: '.',  
remoteDirectorySDF: false, removePrefix: 'target', sourceFiles: 'target/*.war']], usePromotionTimestamp: false,  
useWorkspaceInPromotion: false, verbose: false)])  
    }  
}
```

