


디플로이먼트, 노드포트, 서비스  
**backend-user**

# backend-user 프로젝트 생성

<https://start.spring.io/> 접속

Artifact: k8s-backend-user



**Project**  
☒ Gradle - Groovy ☐ Gradle - Kotlin  
☐ Maven

**Language**  
☒ Java ☐ Kotlin ☐ Groovy

**Spring Boot**  
☐ 4.0.0 (SNAPSHOT) ☐ 3.5.1 (SNAPSHOT) ☒ 3.5.0 ☐ 3.4.7 (SNAPSHOT)  
☐ 3.4.6 ☐ 3.3.13 (SNAPSHOT) ☐ 3.3.12

**Project Metadata**  

Group

Artifact

Name

Description

Package name

Packaging ☒ Jar ☐ War

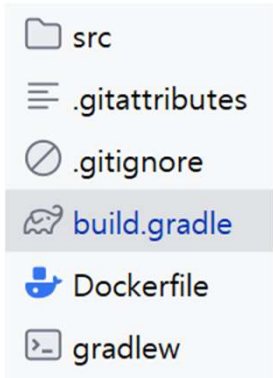
Java ☐ 24 ☐ 21 ☒ 17

**Dependencies** ADD DEPENDENCIES... CTRL + B

**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.

# backend-user 프로젝트 build.gradle 수정



```
plugins {  
    id 'java'  
    id 'org.springframework.boot' version '3.5.0'  
    id 'io.spring.dependency-management' version '1.1.7'  
}
```

```
group = 'com.welab'
```

```
version = '0.0.1'
```

-SNAPSHOT 제거

```
...
```

```
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}"  
    }  
}
```

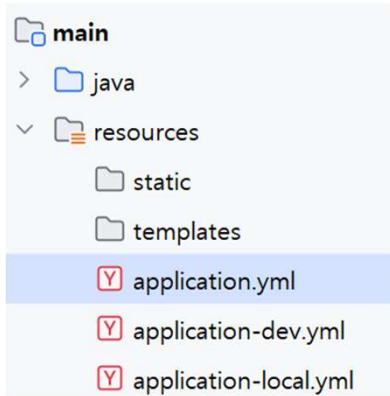
추가

## backend-user 프로젝트 기본 세팅

---

- ✓ 파일 > 프로젝트 구조 > SDK 확인
- ✓ application.yml, application-local.yml, application-dev.yml 설정 분리
  - application.properties는 삭제
- ✓ active profiles 지정
  - Community 버전: VM 옵션에 -Dspring.profiles.active=local 입력
  - Ultimate 버전: 활성 프로파일에 local 입력
- ✓ 기본 코드 세팅
  - ApiResponseDto (응답 메시지 정규화 @NoArgsConstructor 포함할 것)
  - ApiError, ClientError 등 Api Exception
  - ApiCommonAdvice (에러 응답 처리)

## backend-user 프로퍼티 설정



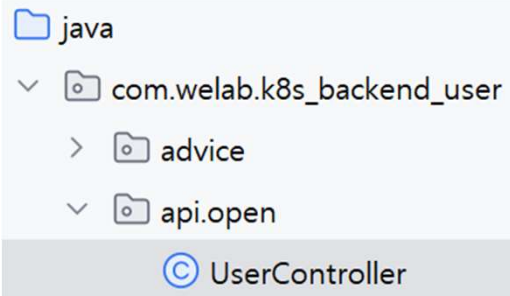
```
spring:  
  application:  
    name: k8s-backend-user
```

application.yml

```
server:  
  port: 8080
```

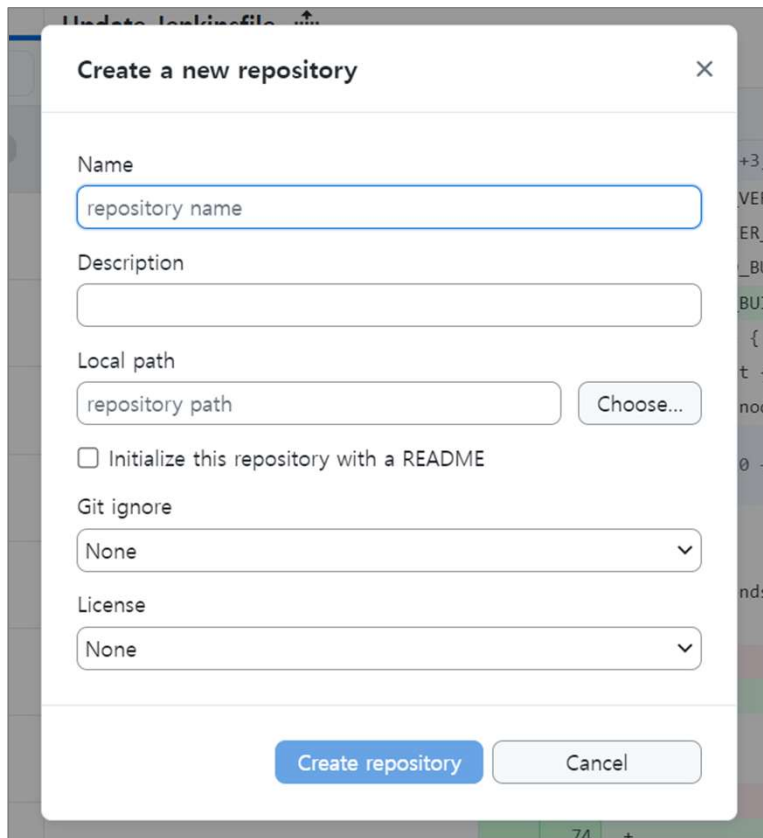
application-local.yml  
application-dev.yml

## backend-user : UserController 추가



```
@Slf4j
@RestController
@RequestMapping(value = "/api/user/v1", produces = MediaType.APPLICATION_JSON_VALUE)
@RequiredArgsConstructor
public class UserController {
    @GetMapping(value = "/hello")
    public ApiResponseDto<String> hello() {
        return ApiResponseDto.createOk("웰컴 투 백엔드 유저");
    }
}
```

# api-gateway : Repository 생성 및 Publish



**Create a new repository**

Name  
repository name

Description

Local path  
repository path Choose...

☐ Initialize this repository with a README

Git ignore  
None

License  
None

Create repository Cancel

Name: k8s-backend-user  
Local path: C:\Workspace\k8s

## 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.

Always available in the toolbar for local repositories or **Ctrl + P**

Publish repository

## backend-user : Dockerfile 파일을 프로젝트에 추가

- src
- .gitattributes
- .gitignore
- build.gradle
- Dockerfile**
- gradlew
- gradlew.bat
- HELP.md
- Jenkinsfile
- settings.gradle

```
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"]
```



## backend-user : Jenkins 파일을 프로젝트에 추가

- > gradle
- > k8s
- > src
- ≡ .gitattributes
- 🚫 .gitignore
- 🐘 build.gradle
- 🐳 Dockerfile
- 📄 gradlew
- ≡ gradlew.bat
- 📖 HELP.md
- ≡ Jenkinsfile
- 🐘 settings.gradle

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

backend-user : 코드 commit & push

---

## **Commit & Push to Github**

# backend-user : Jenkins Pipeline 생성

Status

Configure

New Item

Delete Folder


빌드 기록


## 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.

 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.

If you want to create a new item from other existing, you can use this option:

Copy from

k8s-api-gateway

OK

## backend-user : Jenkins Pipeline SCM URL 수정

Definition

Pipeline script from SCM

SCM ?

Git

Repositories ?

Repository URL ?

`https://github.com/solarhc/k8s-backend-user.git`

Credentials ?

`solar.hc@gmail.com/*****`

+ Add

고급 ▾

# backend-user : Jenkins Pipeline 빌드

Status

Changes

**▶ 파라미터와 함께 빌드**

구성

Pipeline 삭제

Move

Full Stage View

Stages

Rename

Pipeline Syntax

## Pipeline kube-backend-user

매개변수가 필요한 빌드입니다.

TAG

origin/main

☐ RELEASE

**▶ 매개변수가 필요한 빌드입니다.** Cancel

Declarative: Checkout SCM	Declarative: Tool Install	Set Version	Build & Test Application	Build Docker Image	Push Docker Image
4s	367ms	1min 14s	49s	7s	11s
4s	367ms	1min 14s	49s	7s	11s

## backend-user : K8S deployment.yaml 작성 및 적용

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: k8s-backend-user-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: k8s-backend-user
  template:
    metadata:
      labels:
        app: k8s-backend-user
    spec:
      containers:
        - name: k8s-backend-user
          image: solarhc/k8s-backend-user:0.0.1
          imagePullPolicy: Always
          env:
            - name: SPRING_PROFILES_ACTIVE
              value: dev
          ports:
            - containerPort: 8080
```

k8s-backend-user-deploy.yaml



## backend-user : K8S service.yaml 작성 및 적용

```
apiVersion: v1
kind: Service
metadata:
  name: k8s-backend-user-service
spec:
  ports:
    - port: 8080
      targetPort: 8080
  selector:
    app: k8s-backend-user
```

k8s-backend-user-service.yaml

입력을 통해 생성    파일을 통해 생성    서식을 통해 생성

현재 선택된 네임스페이스에 생성할 리소스를 명시하는 YAML 또는 JSON

```
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: k8s-backend-user-service
5  spec:
6    ports:
7      - port: 8080
8        targetPort: 8080
9    selector:
10      app: k8s-backend-user
```

업로드    Cancel