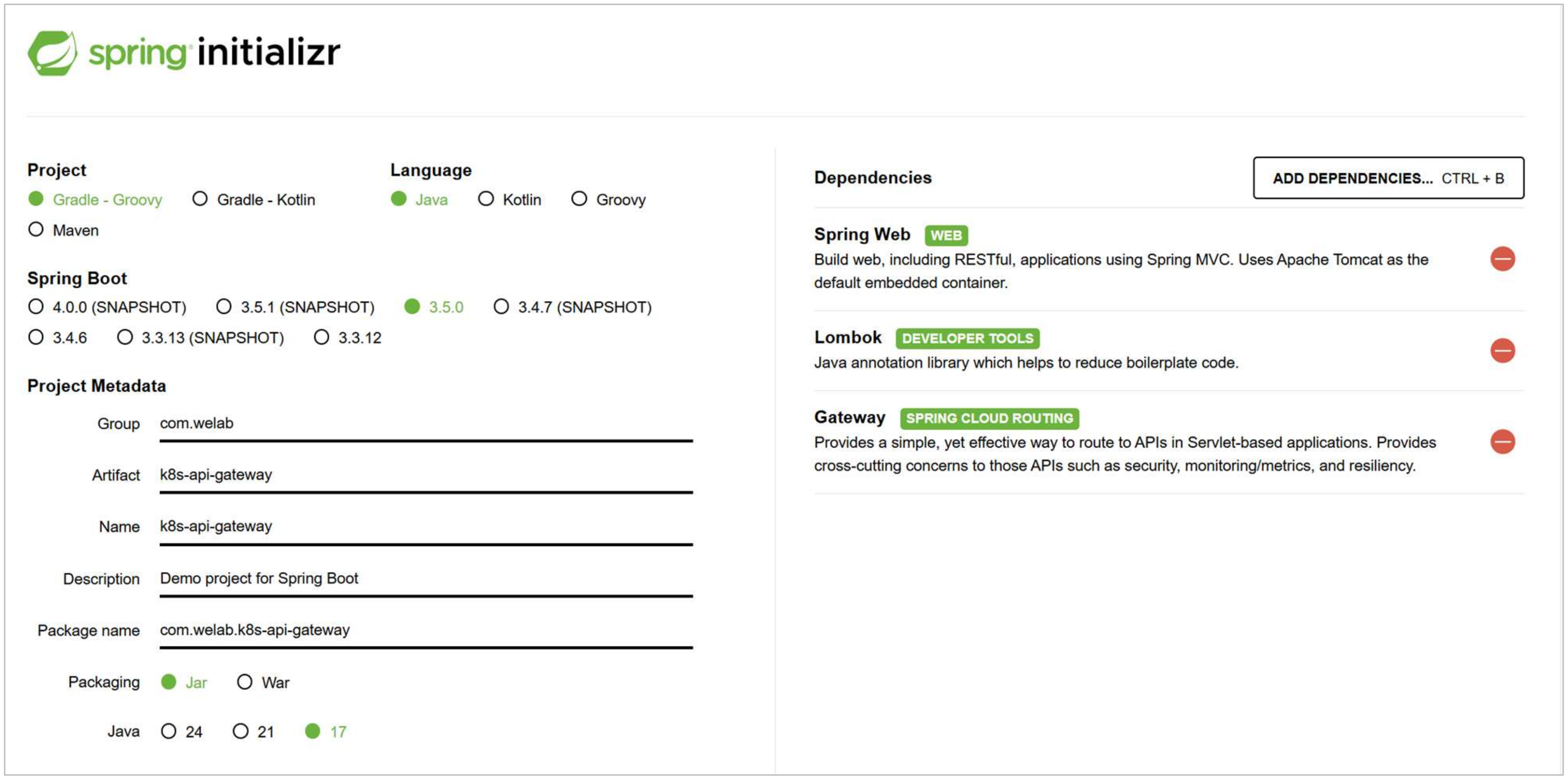
디플로이먼트, 노드포트, 서비스

api-gateway

api-gateway 프로젝트 생성

https://start.spring.io/ 접속

Artifact: k8s-api-gateway



spring initializr

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	com.welab
Artifact	k8s-api-gateway
Name	k8s-api-gateway
Description	Demo project for Spring Boot
Package name	com.welab.k8s-api-gateway
Packaging	[✓ Jar] War
Java	24 21 [✓ 17]

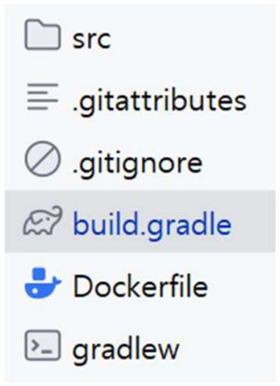
Dependencies

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.

Gateway SPRING CLOUD ROUTING
Provides a simple, yet effective way to route to APIs in Servlet-based applications. Provides cross-cutting concerns to those APIs such as security, monitoring/metrics, and resiliency.

api-gateway 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'
```

-SNAPSHOP 제거

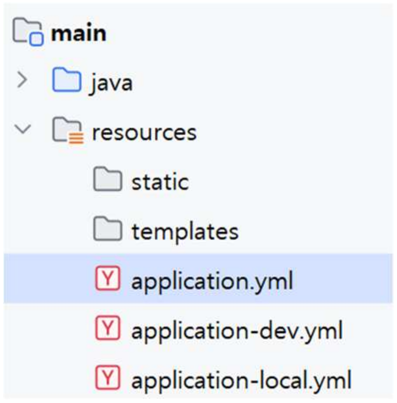
```
...

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

api-gateway 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 @NoArgsContructor 포함할 것)
  - ApiError, ClientError 등 Api Exception
  - ApiCommonAdvice (에러 응답 처리)

api-gateway 프로퍼티 설정



```
main
  java
  resources
    static
    templates
    application.yml
    application-dev.yml
    application-local.yml
```

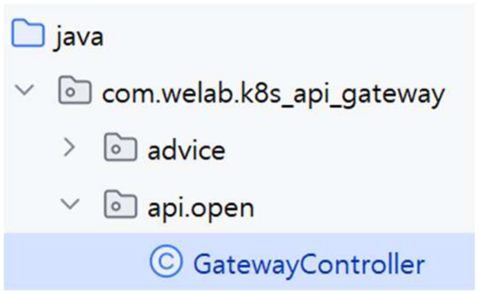
```yaml
spring:
  application:
    name: k8s-api-gateway
```

application.yml

```yaml
server:
  port: 8080
```

application-local.yml
application-dev.yml

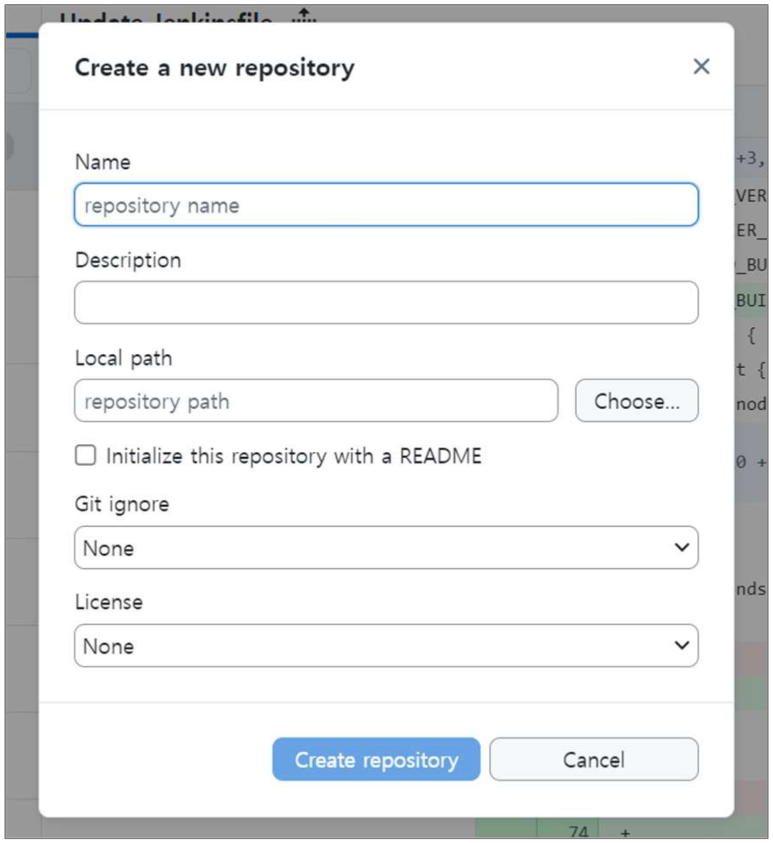
api-gateway : GatewayController 추가



```
java
com.welab.k8s_api_gateway
  advice
  api.open
    GatewayController
```

```java
@Slf4j
@RestController
@RequestMapping(value = "/api/gateway/v1", produces = MediaType.APPLICATION_JSON_VALUE)
@RequiredArgsConstructor
public class GatewayController {
    @GetMapping(value = "/hello")
    public ApiResponseDto<String> test() {
        return ApiResponseDto.createOk("안녕  쿠버네티스");
    }
}
```

api-gateway : Repository 생성 및 Publish



Create a new repository

Name
repository name

Description

Local path
repository path

Choose...

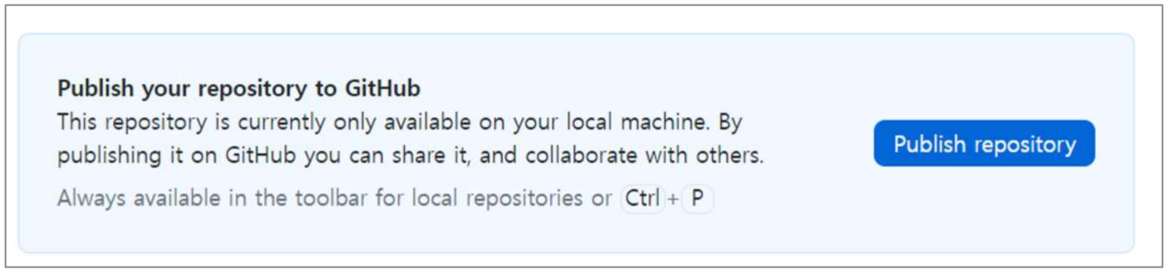
Initialize this repository with a README

Git ignore
None

License
None

Create repository
Cancel

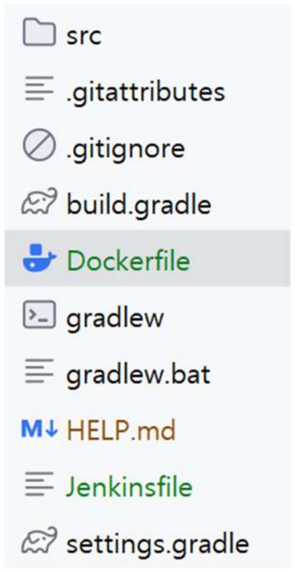
Name: k8s-api-gateway
Local path: C:₩Workspace₩k8s



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

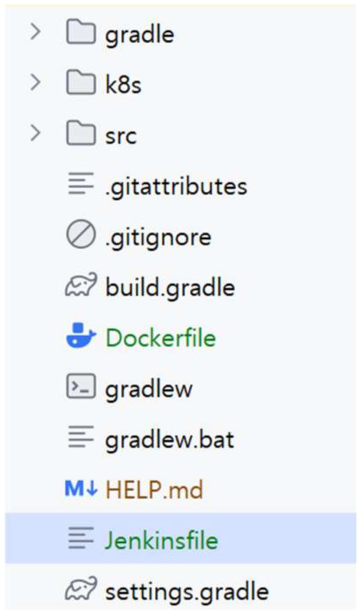
Publish repository

api-gateway : 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"]
```

api-gateway : Jenkins 파일을 프로젝트에 추가



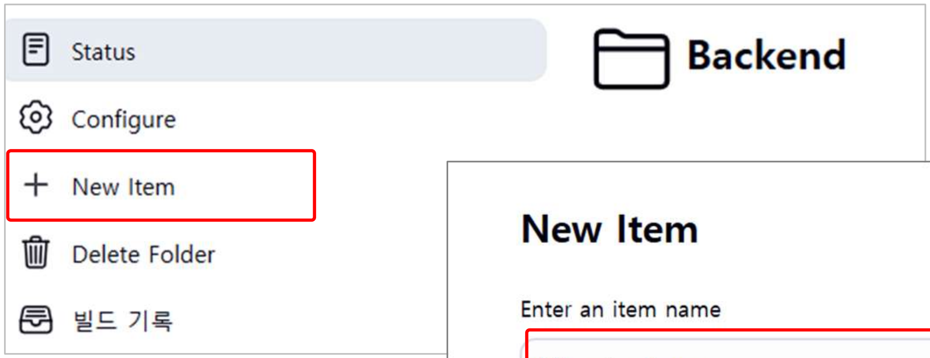
```
> gradle
> k8s
> src
  .gitattributes
  .gitignore
  build.gradle
  Dockerfile
  gradlew
  gradlew.bat
  HELP.md
  Jenkinsfile
  settings.gradle
```

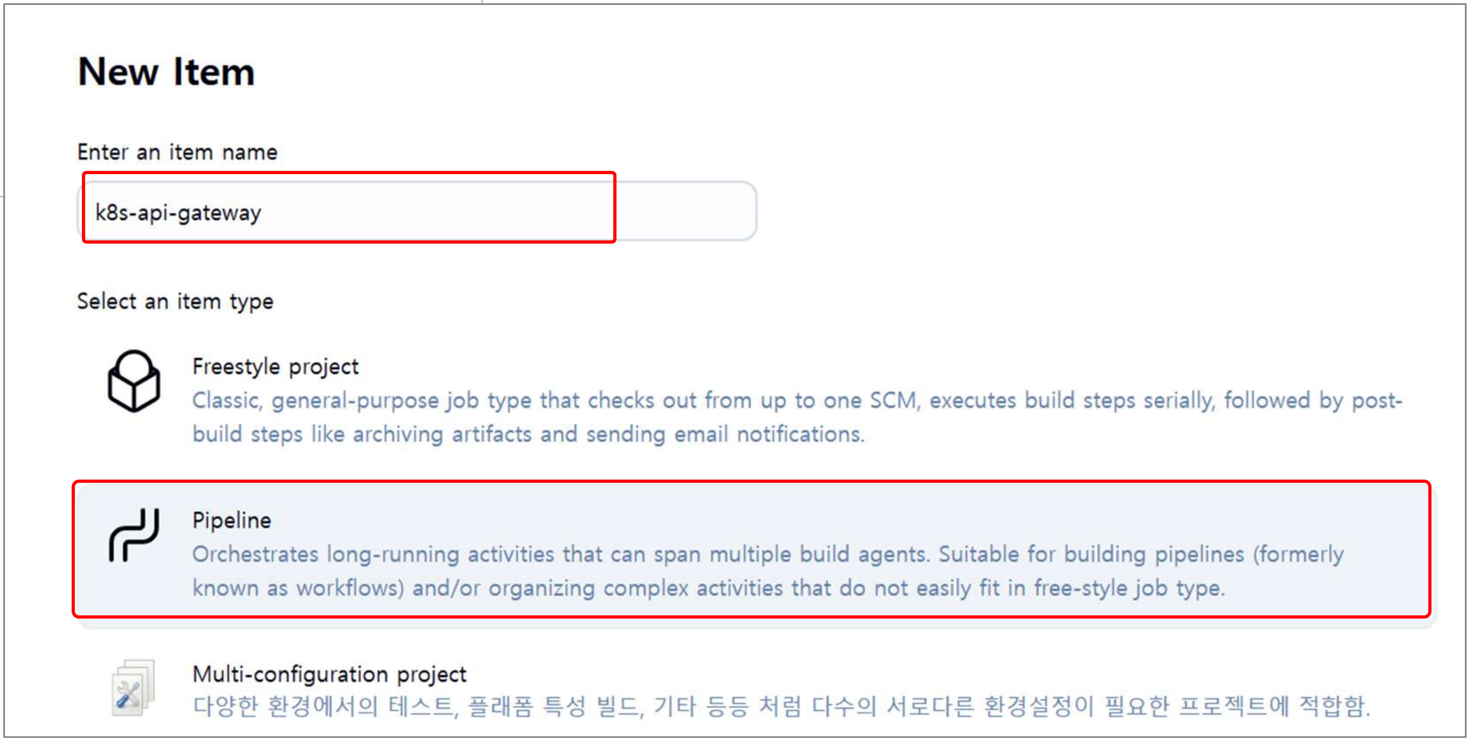
```
pipeline {
    …
    environment {
        GIT_URL = "https://github.com/solarhc/k8s-api-gateway.git"
        GITHUB_CREDENTIAL = "github-token"
        ARTIFACTS = "build/libs/**"
        DOCKER_REGISTRY = "solarhc"
        DOCKERHUB_CREDENTIAL = 'dockerhub-token'
    }

    …
}
```

api-gateway : 코드 commit & push

Commit & Push to Github

api-gateway : Jenkins Pipeline 생성




New Item

Enter an item name

k8s-api-gateway

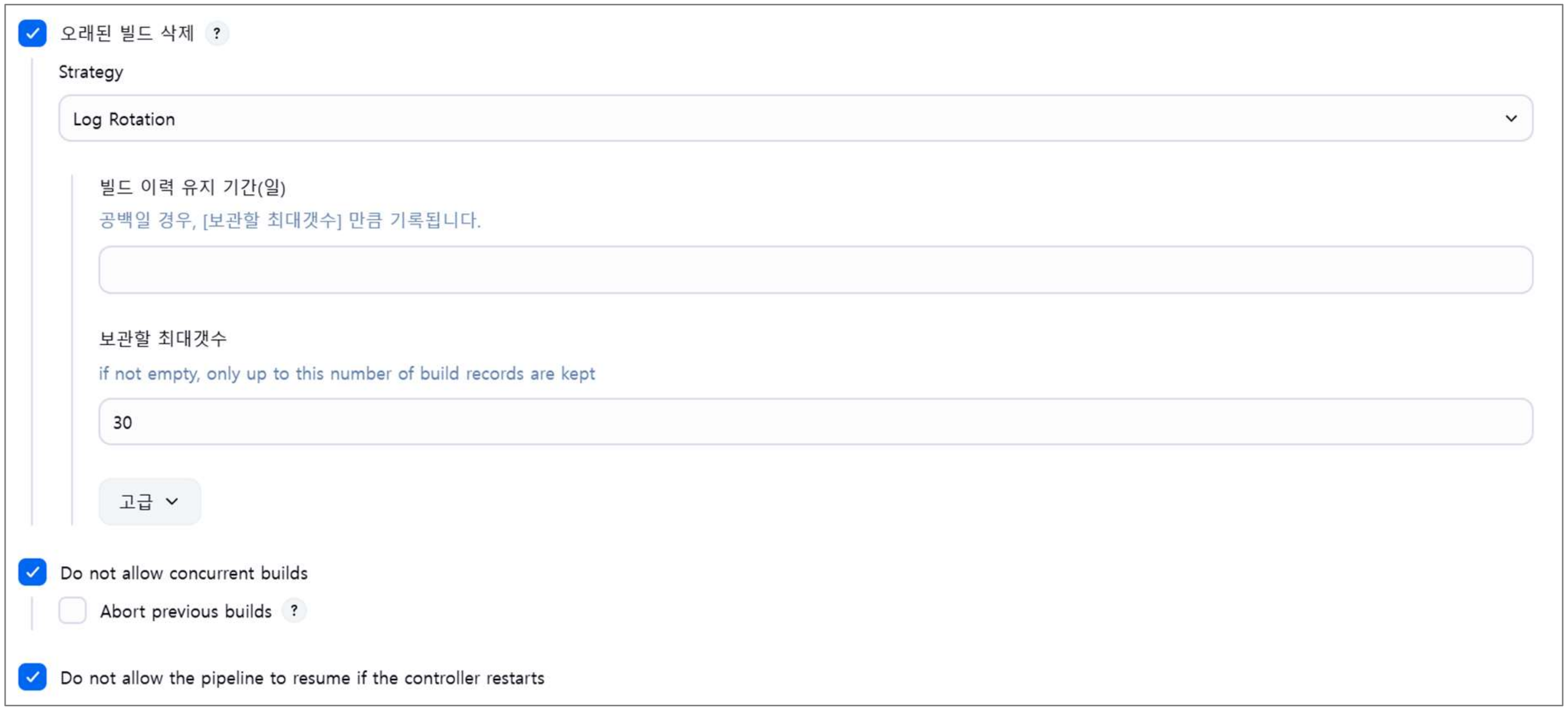
Select an item type

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.

api-gateway : Jenkins Pipeline General 설정



[✓] 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

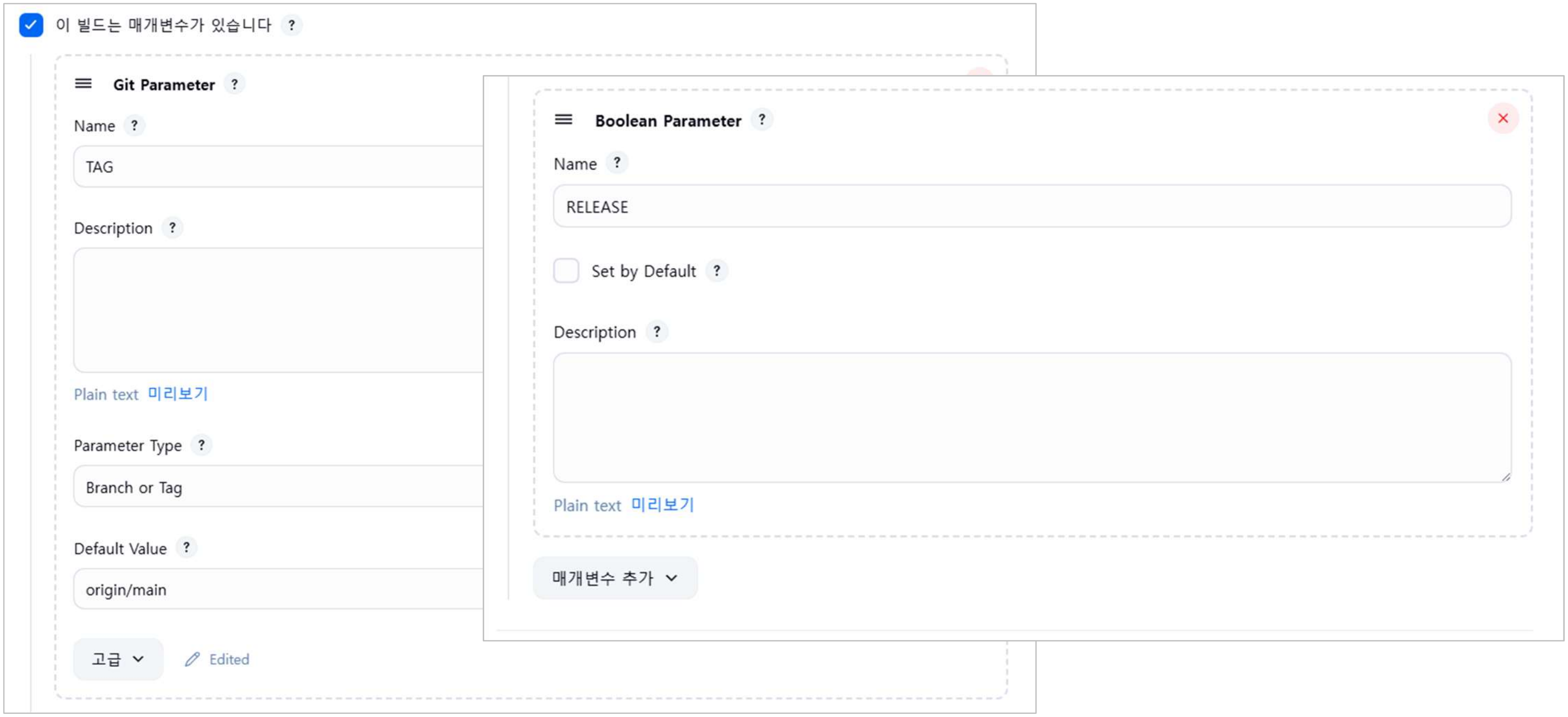
30

[✓] Do not allow concurrent builds

[ ] Abort previous builds ?

[✓] Do not allow the pipeline to resume if the controller restarts

api-gateway : Jenkins Pipeline 매개변수 설정



☑ 이 빌드는 매개변수가 있습니다 ?

Git Parameter ?

Name ?

TAG

Description ?

Plain text 미리보기

Parameter Type ?

Branch or Tag

Default Value ?

origin/main

고급 ⌄ Edited

Boolean Parameter ?

Name ?

RELEASE

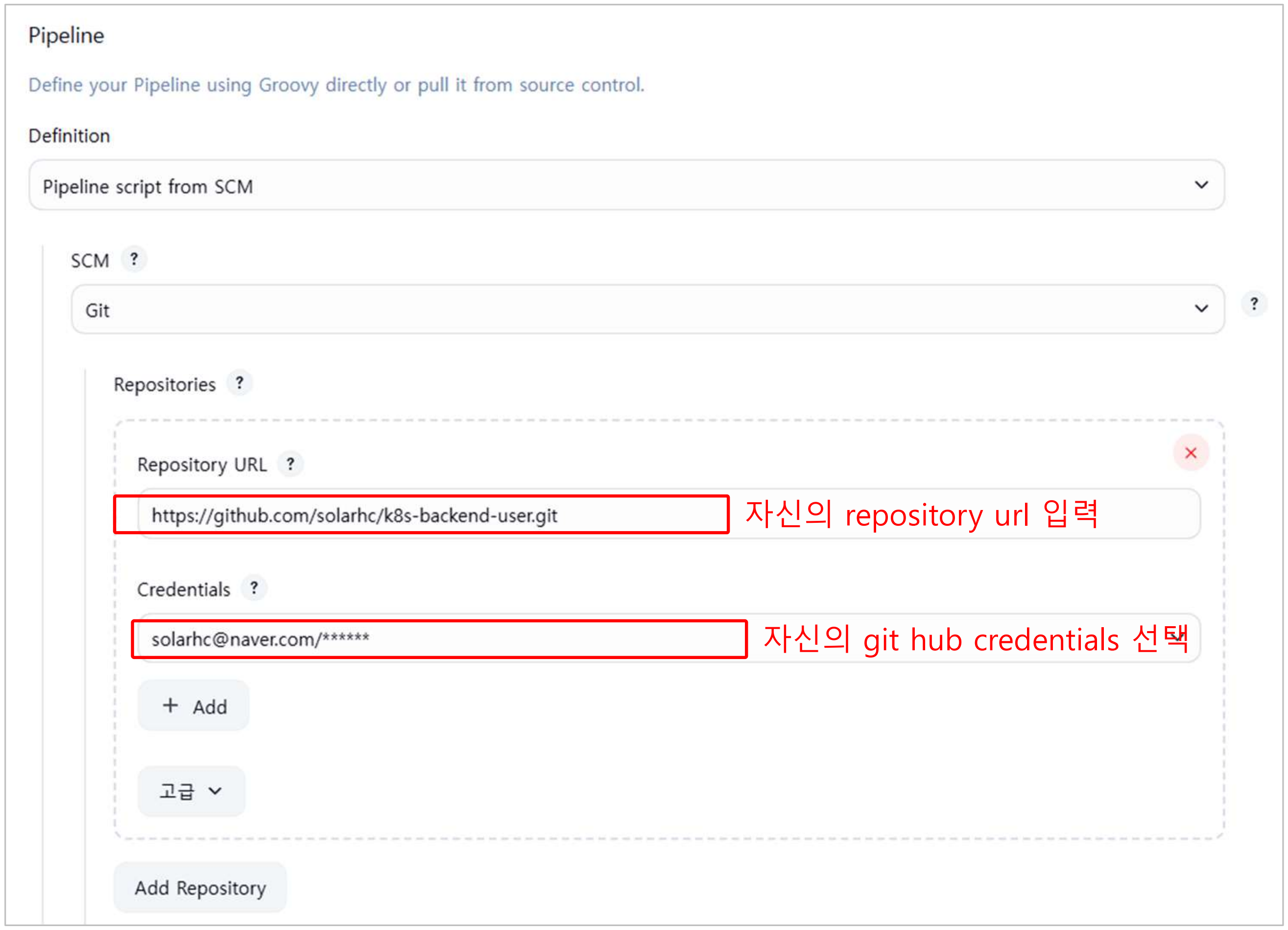
☐ Set by Default ?

Description ?

Plain text 미리보기

매개변수 추가 ⌄

api-gateway : Jenkins Pipeline SCM 설정



Pipeline

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

Definition: Pipeline script from SCM

SCM: Git

Repositories

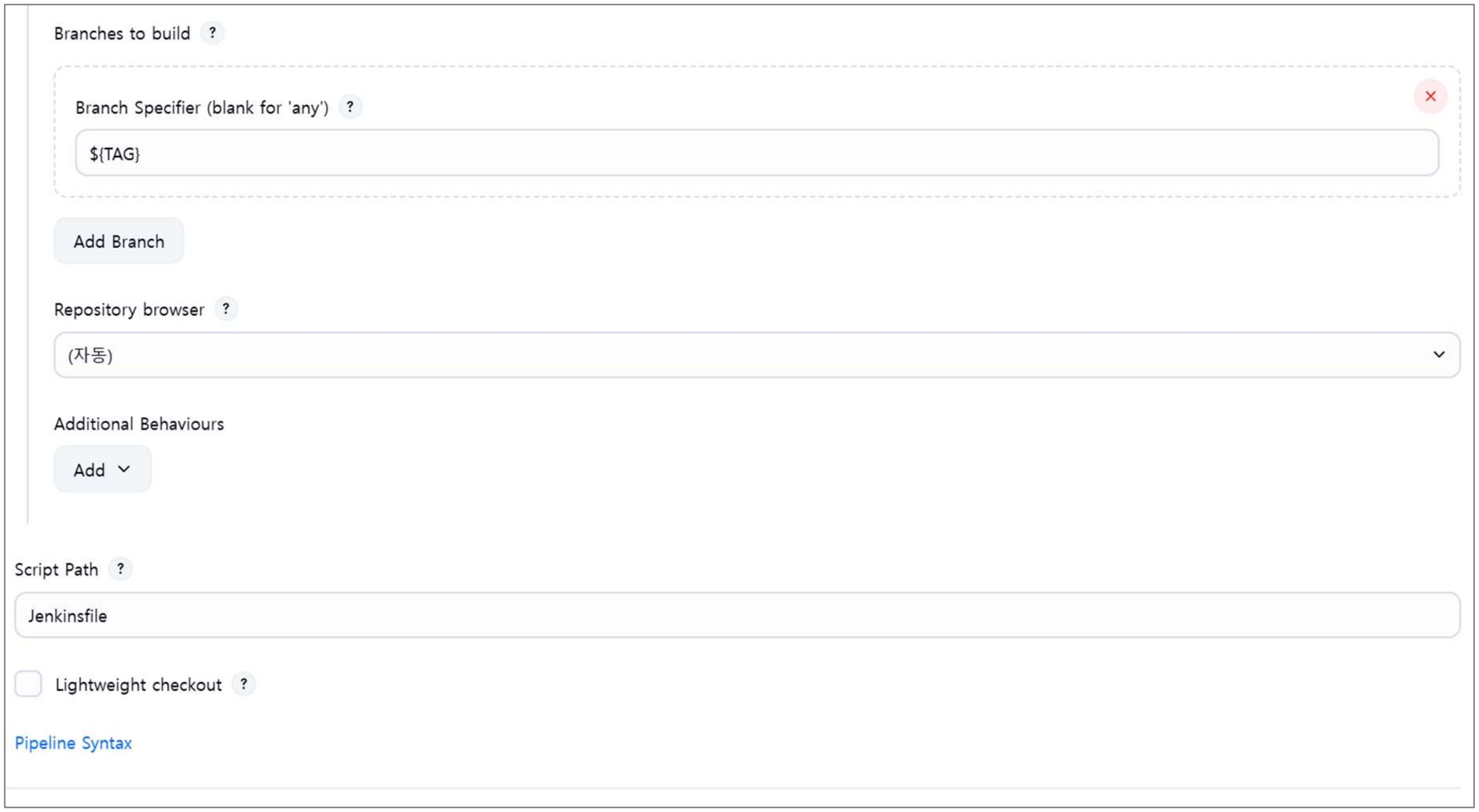
Repository URL: https://github.com/solarhc/k8s-backend-user.git

자신의 repository url 입력

Credentials: solarhc@naver.com/******

자신의 git hub credentials 선택

api-gateway : Jenkins Pipeline SCM 설정 (계속)



Branches to build ?

Branch Specifier (blank for 'any') ?

${TAG}

Add Branch

Repository browser ?

(자동)

Additional Behaviours

Add

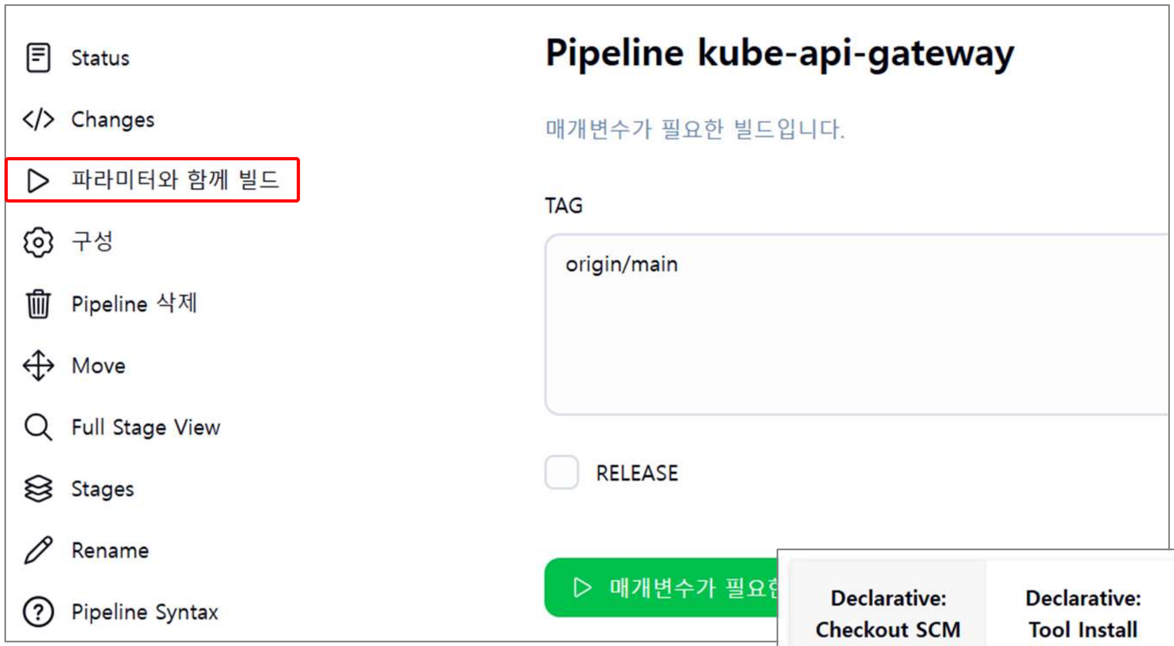
Script Path ?

Jenkinsfile

Lightweight checkout ?

Pipeline Syntax

api-gateway : Jenkins Pipeline 빌드



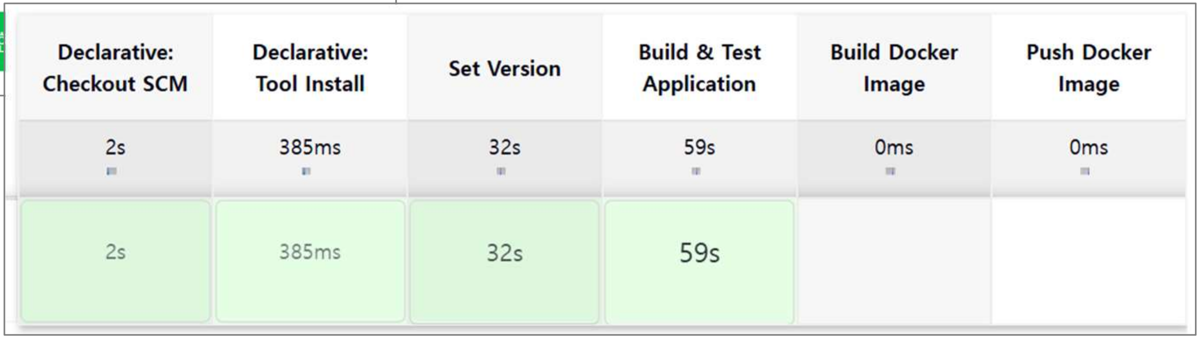
Pipeline kube-api-gateway

매개변수가 필요한 빌드입니다.

TAG

origin/main

RELEASE

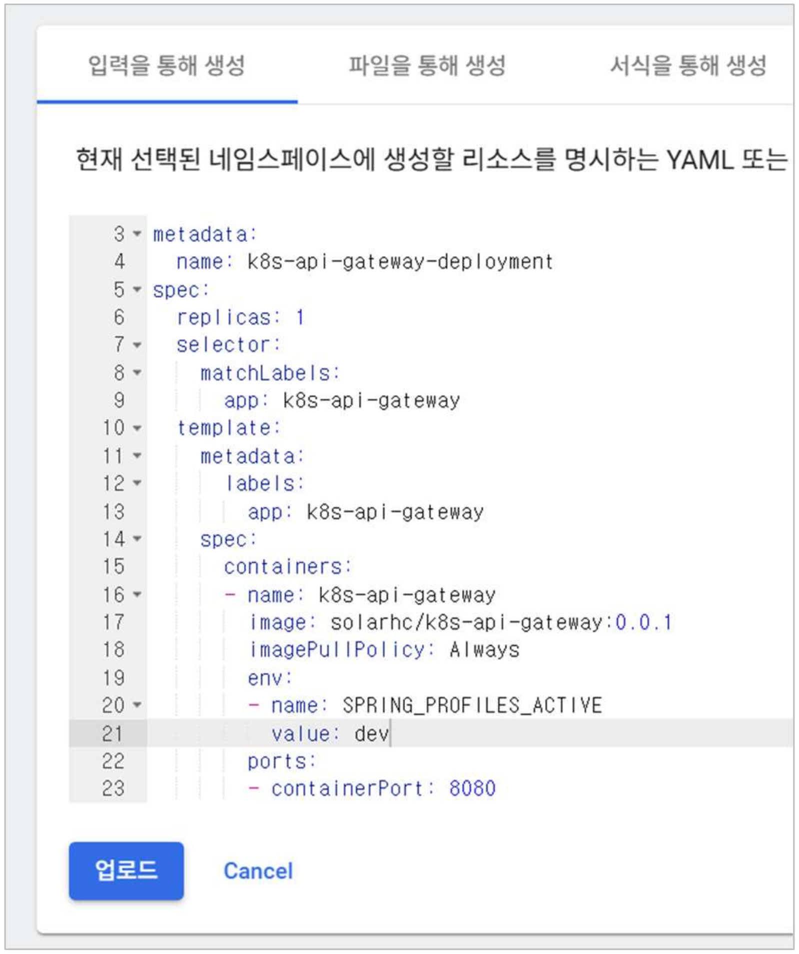


Declarative: Checkout SCM	Declarative: Tool Install	Set Version	Build & Test Application	Build Docker Image	Push Docker Image
2s	385ms	32s	59s	0ms	0ms
2s	385ms	32s	59s		

api-gateway : K8S deployment.yaml 작성 및 적용

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

k8s-api-gateway-deploy.yaml



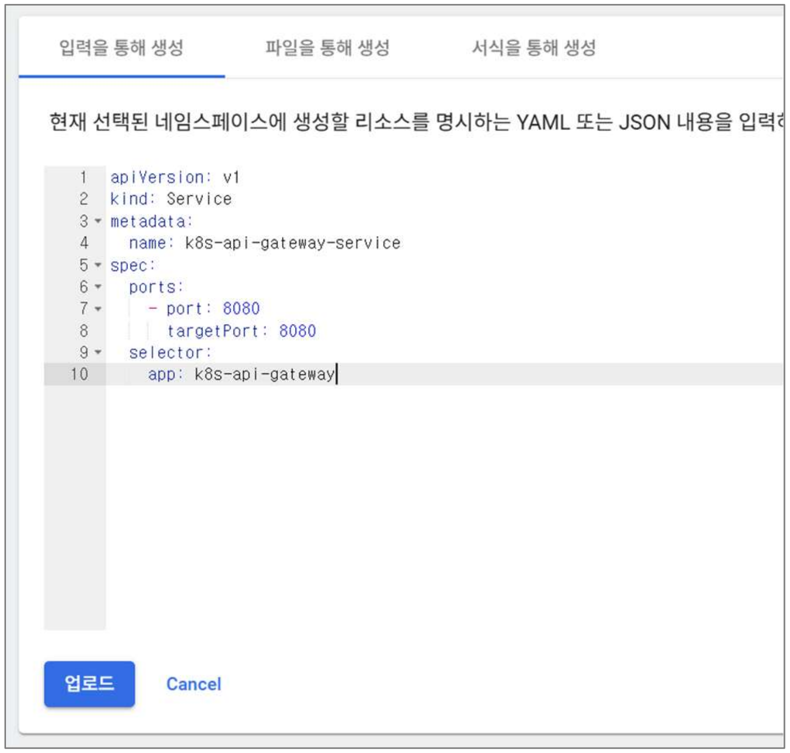
현재 선택된 네임스페이스에 생성할 리소스를 명시하는 YAML 또는

```yaml
metadata:
  name: k8s-api-gateway-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: k8s-api-gateway
  template:
    metadata:
      labels:
        app: k8s-api-gateway
    spec:
      containers:
      - name: k8s-api-gateway
        image: solarhc/k8s-api-gateway:0.0.1
        imagePullPolicy: Always
        env:
        - name: SPRING_PROFILES_ACTIVE
          value: dev
        ports:
        - containerPort: 8080
```

api-gateway : K8S service.yaml 작성 및 적용

```yaml
apiVersion: v1
kind: Service
metadata:
  name: k8s-api-gateway-service
spec:
  ports:
    - port: 8080
      targetPort: 8080
  selector:
    app: k8s-api-gateway
```

k8s-api-gateway-service.yaml



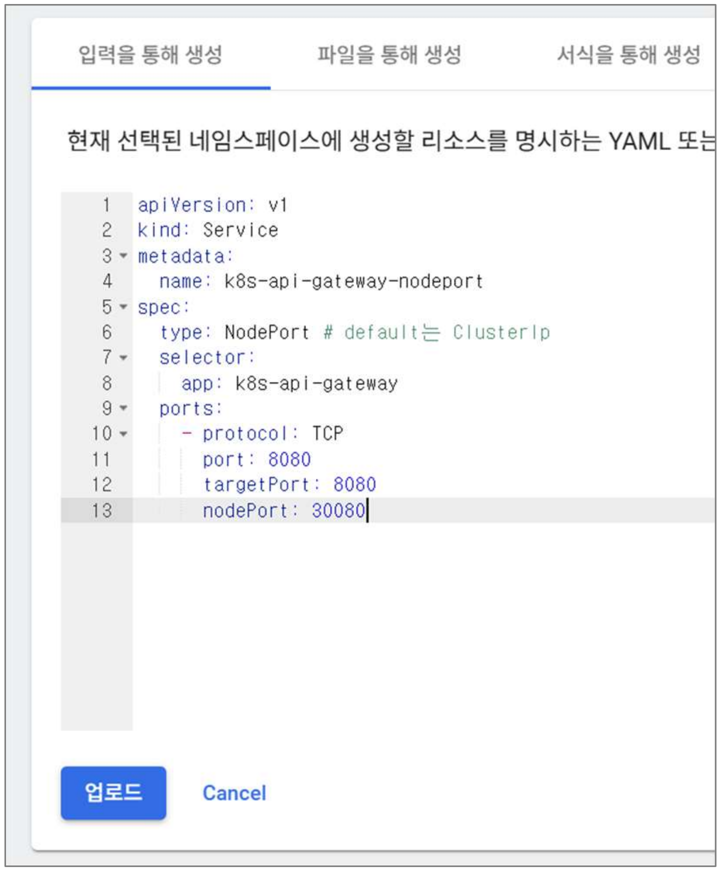
현재 선택된 네임스페이스에 생성할 리소스를 명시하는 YAML 또는 JSON 내용을 입력

```yaml
apiVersion: v1
kind: Service
metadata:
  name: k8s-api-gateway-service
spec:
  ports:
    - port: 8080
      targetPort: 8080
  selector:
    app: k8s-api-gateway
```

api-gateway : K8S nodeport.yaml 작성 및 적용

```yaml
apiVersion: v1
kind: Service
metadata:
  name: k8s-api-gateway-nodeport
spec:
  type: NodePort # default는 ClusterIp
  selector:
    app: k8s-api-gateway
  ports:
    - protocol: TCP
      port: 8080
      targetPort: 8080
      nodePort: 30080
```

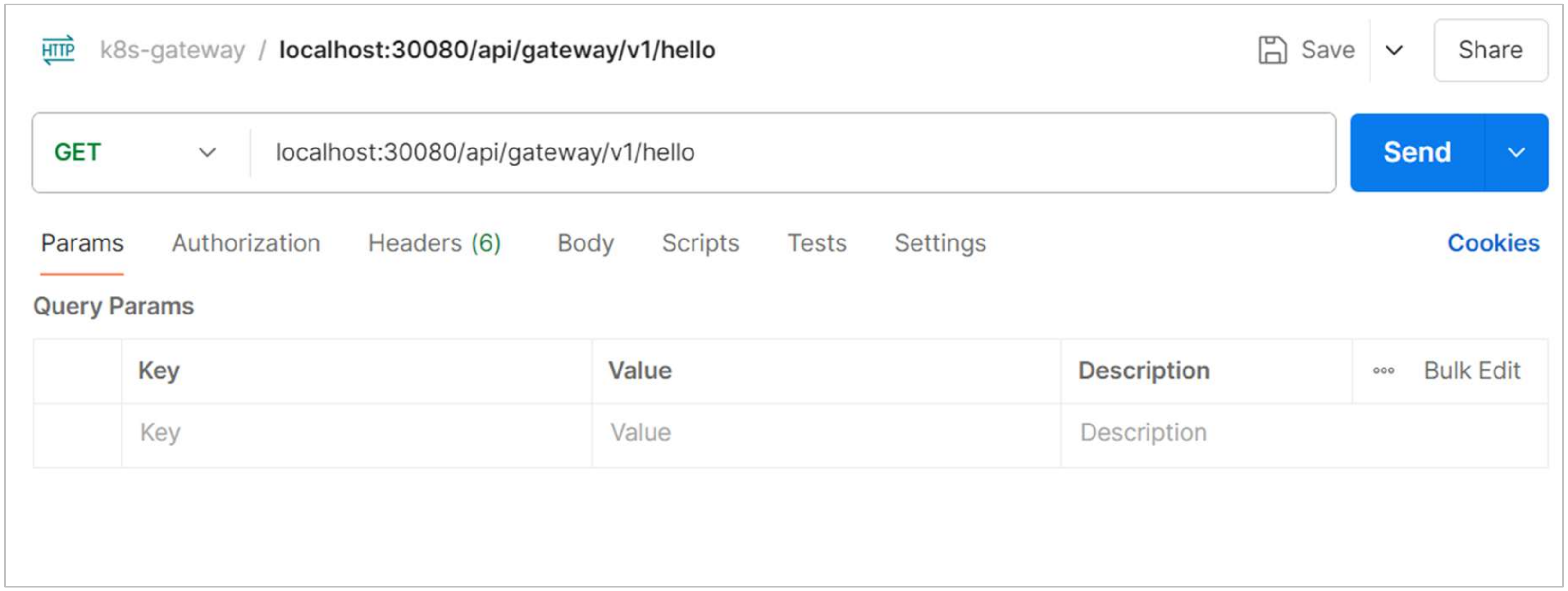
k8s-api-gateway-nodeport.yaml



현재 선택된 네임스페이스에 생성할 리소스를 명시하는 YAML 또는

```yaml
apiVersion: v1
kind: Service
metadata:
  name: k8s-api-gateway-nodeport
spec:
  type: NodePort # default는 ClusterIp
  selector:
    app: k8s-api-gateway
  ports:
    - protocol: TCP
      port: 8080
      targetPort: 8080
      nodePort: 30080
```

api-gateway : nodeport 동작 확인


HTTP k8s-gateway / localhost:30080/api/gateway/v1/hello

GET localhost:30080/api/gateway/v1/hello

Query Params

	Key	Value	Description	ooo Bulk Edit
	Key	Value	Description	