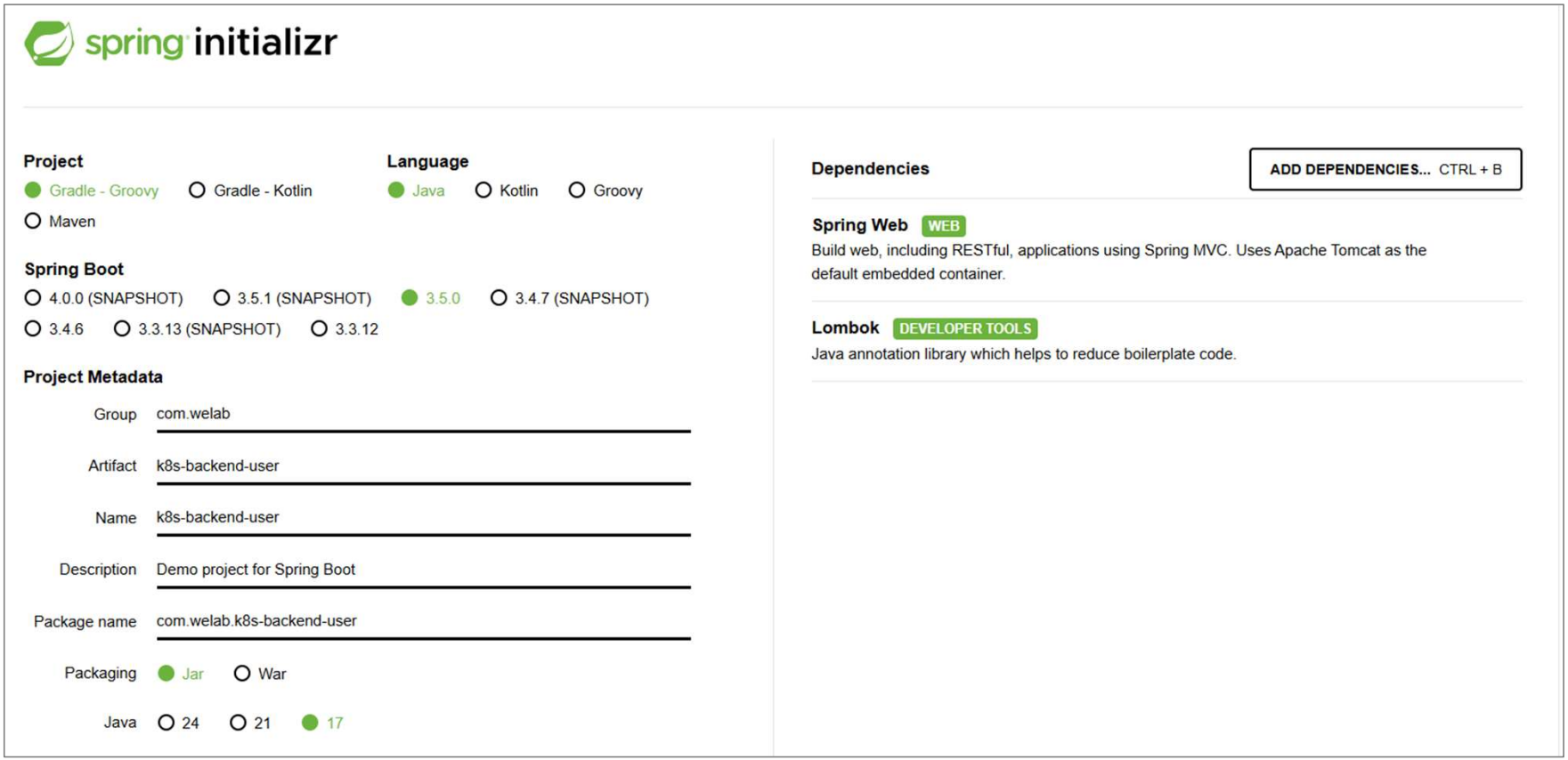
디플로이먼트, 노드포트, 서비스

backend-user

backend-user 프로젝트 생성

https://start.spring.io/ 접속

Artifact: k8s-backend-user



backend-user 프로젝트 build.gradle 수정

- src
- .gitattributes
- .gitignore
- build.gradle
- Dockerfile
- gradlew

```
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

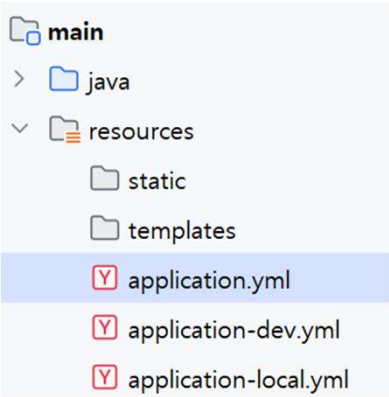
```
...

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

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

backend-user 프로퍼티 설정



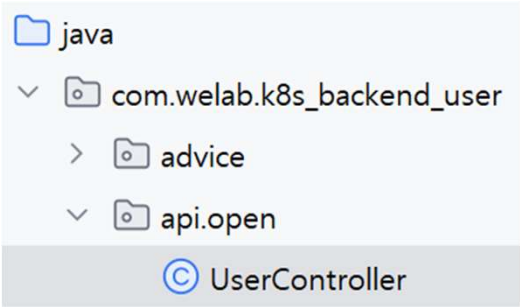

```yml
spring:
  application:
    name: k8s-backend-user
```

application.yml

```yml
server:
  port: 8080
```

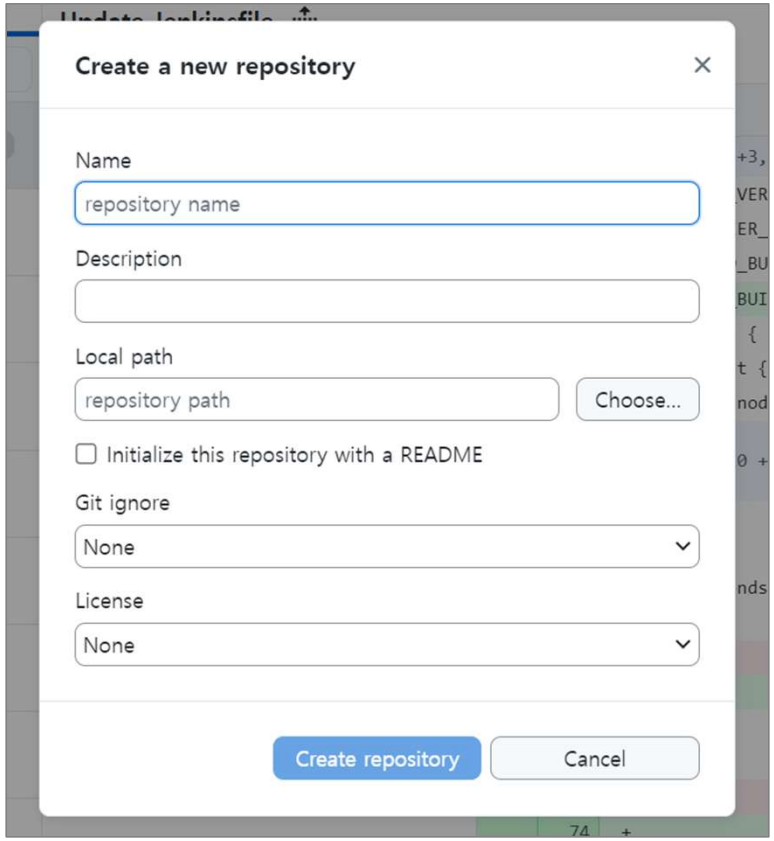
application-local.yml
application-dev.yml

backend-user : UserController 추가

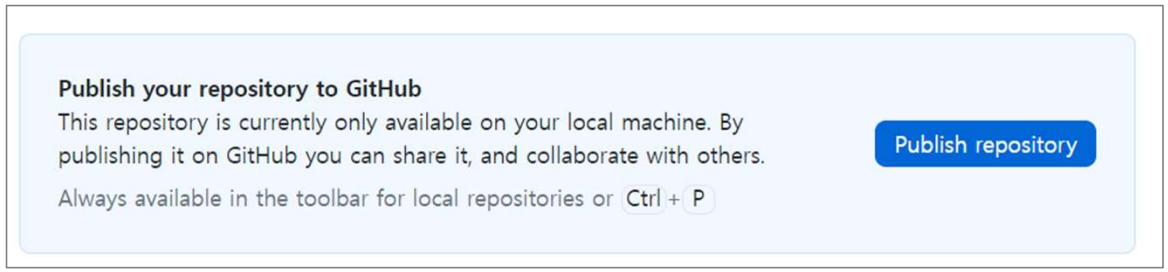



```java
@Slf4j
@RestController
@RequestMapping(value = "/api/user/v1", produces = MediaType.APPLICATION_JSON_VALUE)
@RequiredArgsConstructor
public class UserController {
    @GetMapping(value = "/hello")
    public ApiResponseDto<String> hello() {
        return ApiResponseDto.createOk("웰컴 투 백엔드 유저");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

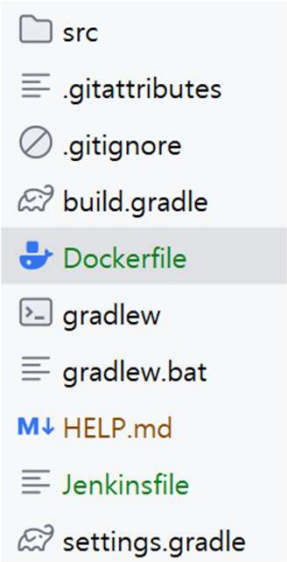
api-gateway : Repository 생성 및 Publish



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

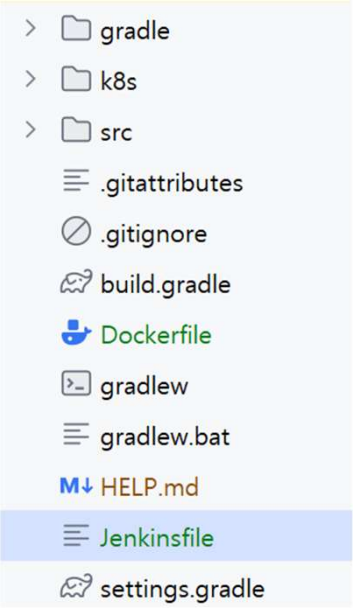


backend-user : Dockerfile 파일을 프로젝트에 추가




```
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"]
```

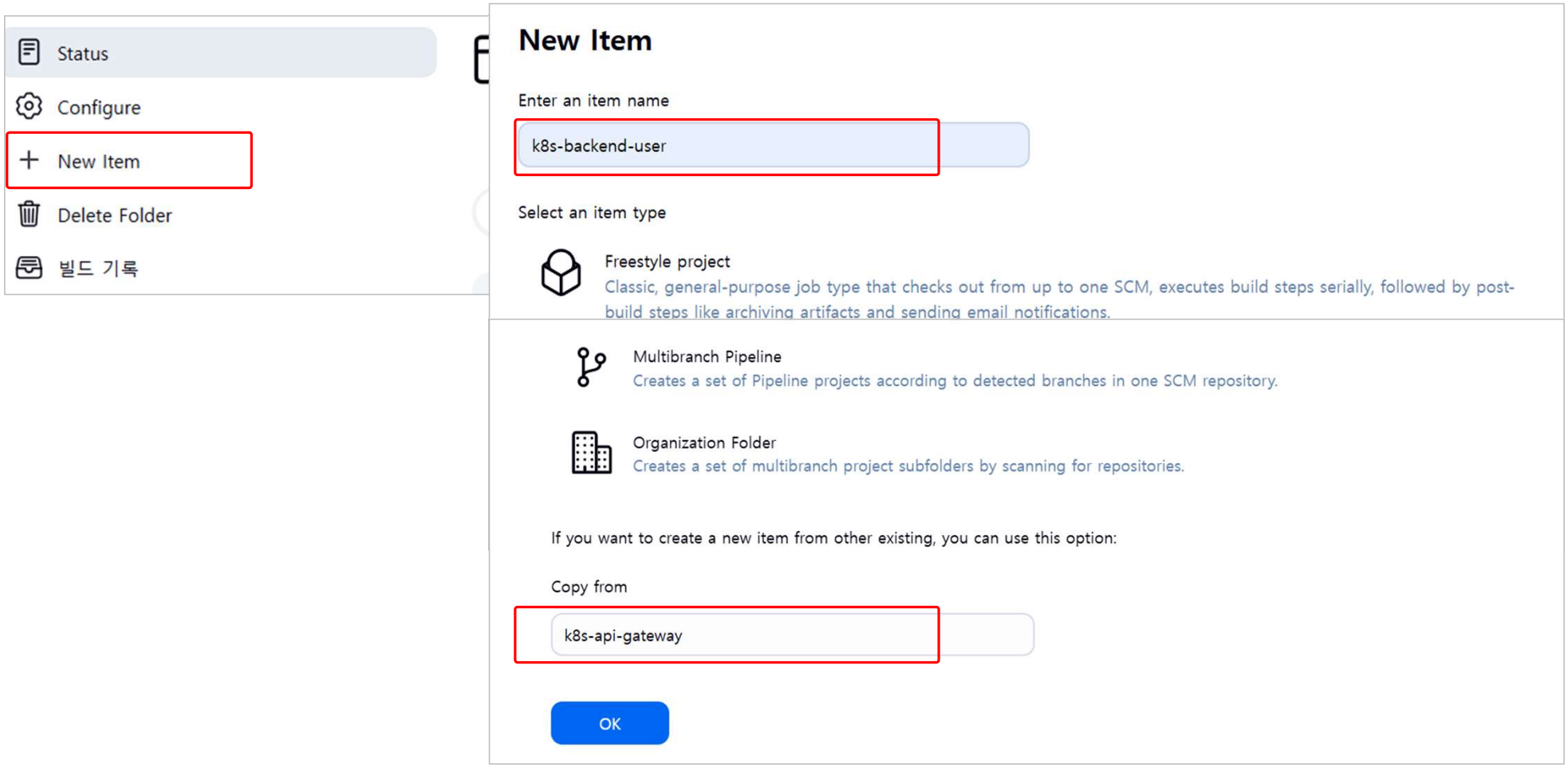
backend-user : Jenkins 파일을 프로젝트에 추가



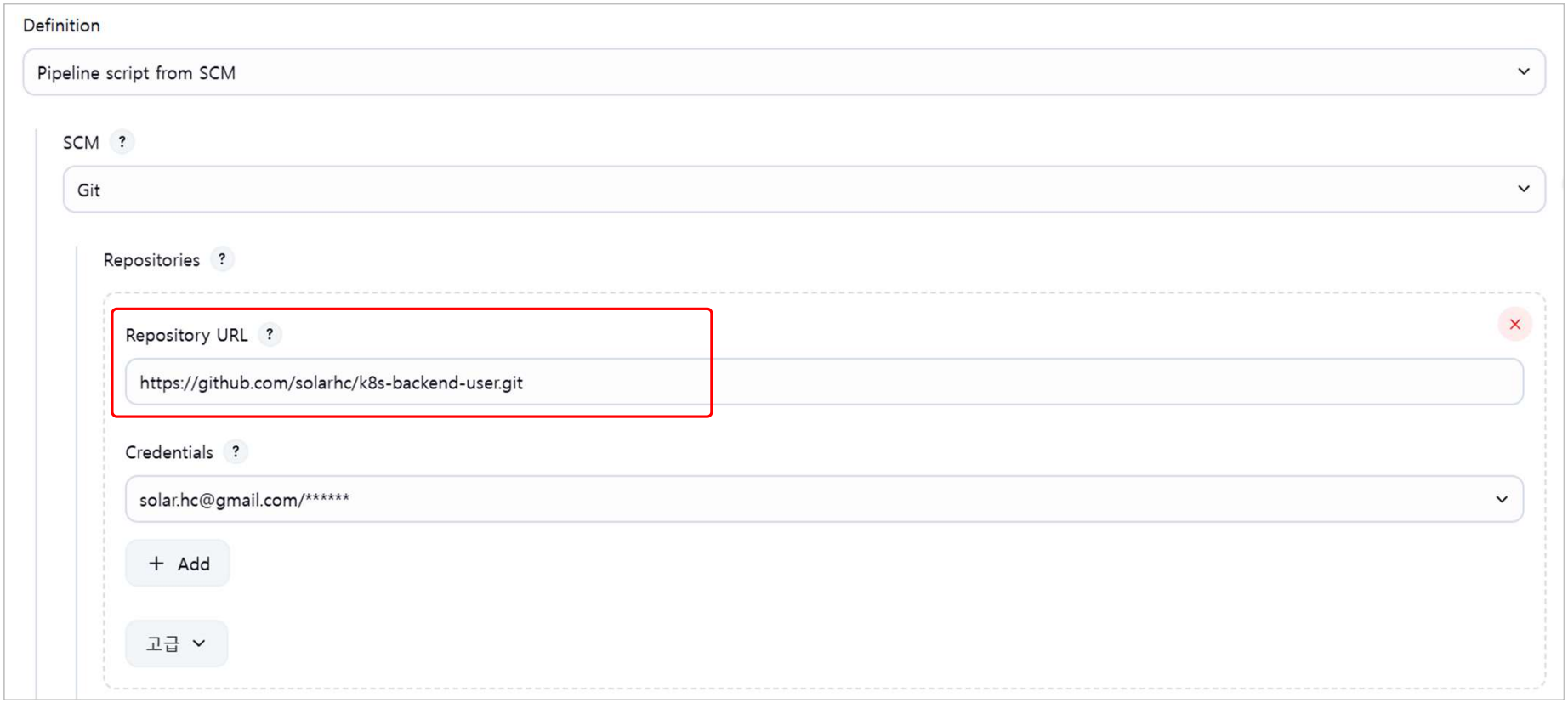

```
pipeline {
    …
    environment {
        GIT_URL = "https://github.com/solarhc/k8s-backend-user.git"
        GITHUB_CREDENTIAL = "github-token"
        ARTIFACTS = "build/libs/**"
        DOCKER_REGISTRY = "solarhc"
        DOCKERHUB_CREDENTIAL = 'dockerhub-token'
    }

    …
}
```

Commit & Push to Github

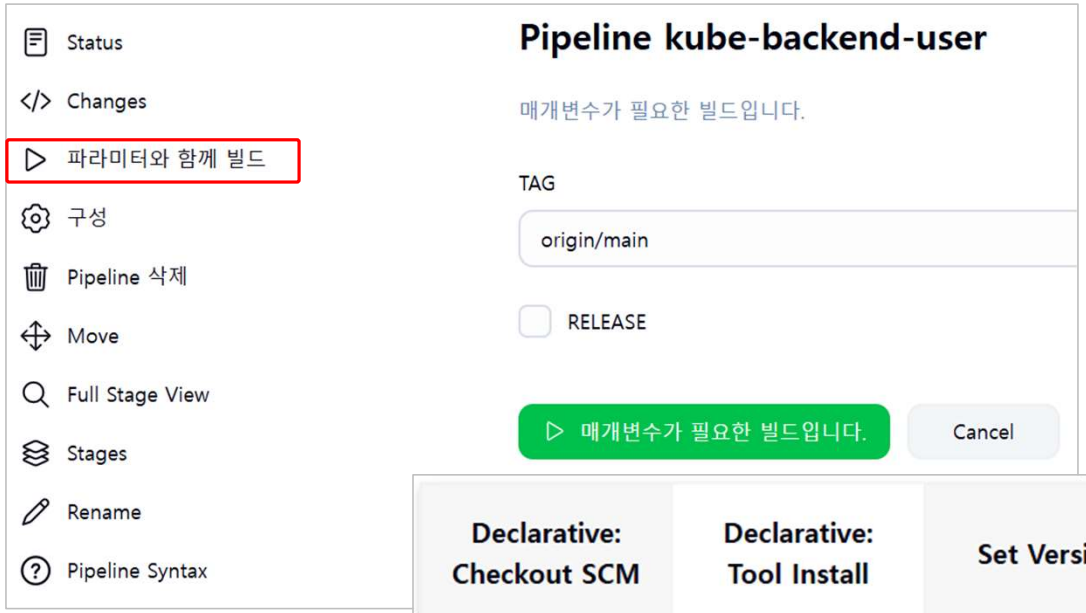
backend-user : Jenkins Pipeline 생성



backend-user : Jenkins Pipeline SCM URL 수정



backend-user : Jenkins Pipeline 빌드



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

```yaml
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

```yaml
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