

04. 컨피그맵(ConfigMap), 시크릿(Secret)을 활용해 환경변수 관리하기

1. [예제] 백엔드(Spring Boot) 서버에 환경변수 등록해 사용하기

✅ 백엔드(Spring Boot) 서버에 환경변수 등록해 사용하기

1. Spring Boot 프로젝트 셋팅(demo2)

start.spring.io

의존성 추가 web, devtools

2. 간단한 코드 작성

AppController

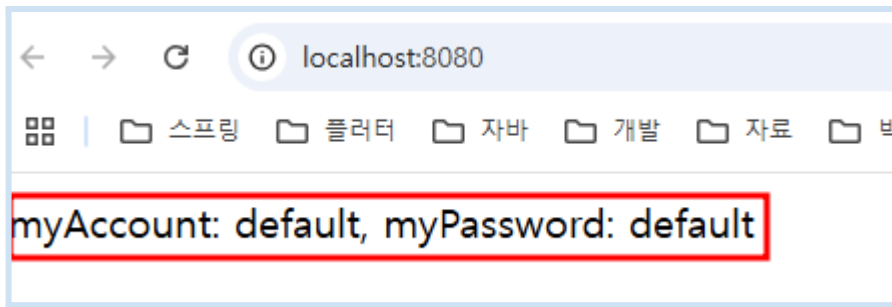
```
@RestController
public class AppController {

    @Value("${MY_ACCOUNT:default}")
    private String myAccount;

    @Value("${MY_PASSWORD:default}")
    private String myPassword;

    @GetMapping("/")
    public String home() {
        return "myAccount: " + myAccount + ", myPassword: " + myPassword;
    }
}
```

3. 프로젝트 실행시켜보기



4. Dockerfile 작성하기

Dockerfile

```
FROM openjdk:17-jdk
```

```
COPY build/libs/*SNAPSHOT.jar app.jar
```

```
ENTRYPOINT ["java", "-jar", "/app.jar"]
```

5. Spring Boot 프로젝트 빌드하기

```
$ ./gradlew clean build
```

```
PS C:\Users\Aiclass> cd C:\DevData\Kubernetes\demo2
PS C:\DevData\Kubernetes\demo2> ./gradlew clean build
OpenJDK 64-Bit Server VM warning: Sharing is only supported for boot loader classes because bootstrap classpath has been appended
BUILD SUCCESSFUL in 9s
8 actionable tasks: 7 executed, 1 up-to-date
PS C:\DevData\Kubernetes\demo2>
```

6. Dockerfile를 바탕으로 이미지 빌드하기

```
$ docker build -t spring-server .
```

```
PS C:\DevData\Kubernetes\demo2> docker image ls
REPOSITORY          IMAGE ID      CREATED        TAG      SIZE
spring-server       c7b4bad4c1c8  22 seconds ago  latest   766MB
nest-server         9b8134e46317  8 hours ago    1.1      1.87GB
nest-server         5a1bb3bb949e  8 hours ago    1.0      1.87GB
```

7. 매니페스트 파일 작성하기

spring-deployment.yaml

```
apiVersion: apps/v1
```

```
kind: Deployment
```

Deployment 기본 정보

```
metadata:
```

```
  name: spring-deployment # Deployment 이름
```

Deployment 세부 정보

```
spec:
```

```
  replicas: 3 # 생성할 파드의 복제본 개수
```

```
  selector:
```

```
    matchLabels:
```

```
      app: backend-app # 아래에서 정의한 Pod 중 'app: backend-app'이라는 값을
가진 파드를 선택
```

배포할 Pod 정의

```
template:
```

```
  metadata:
```

```
    labels: # 레이블 (= 카테고리)
```

```
      app: backend-app
```

spec:

containers:

- name: spring-container # 컨테이너 이름

image: spring-server # 컨테이너를 생성할 때 사용할 이미지

imagePullPolicy: IfNotPresent # 로컬에서 이미지를 먼저 가져온다. 없으면 레지스트리에서 가져온다.

ports:

- containerPort: **8080** # 컨테이너에서 사용하는 포트를 명시적으로 표현

env: # 환경변수 등록

- name: **MY_ACCOUNT**

value: 본인ID

- name: **MY_PASSWORD**

value: pwd1234

spring-service.yaml

apiVersion: v1

kind: Service

Service 기본 정보

metadata:

name: spring-service

Service 세부 정보

spec:

type: NodePort # Service의 종류

selector:

app: backend-app # 실행되고 있는 파드 중 'app: backend-app'이라는 값을 가진 파드와 서비스를 연결

ports:

- protocol: TCP # 서비스에 접속하기 위한 프로토콜

port: 8080 # 쿠버네티스 내부에서 Service에 접속하기 위한 포트 번호 (Service

targetPort: 8080 # 매핑하기 위한 파드의 포트 번호

nodePort: 30000 # 외부에서 사용자들이 접근하게 될 포트 번호

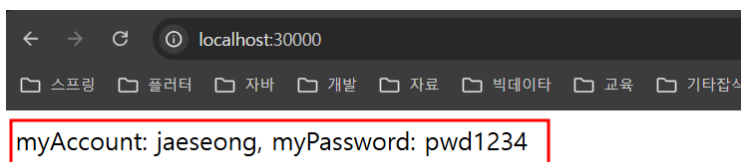
8. 매니페스트 기반으로 실행시키기

```
$ kubectl apply -f spring-deployment.yaml  
$ kubectl apply -f spring-service.yaml
```

확인하기

```
PS C:\DevData\Kubernetes\demo2> kubectl get service  
NAME                TYPE        CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE  
kubernetes           ClusterIP   10.96.0.1     <none>         443/TCP          8h  
nest-service         NodePort    10.98.178.59  <none>         3000:31000/TCP   7h51m  
spring-service       NodePort    10.96.231.117 <none>         8080:30000/TCP   3m7s  
PS C:\DevData\Kubernetes\demo2> kubectl get deployment  
NAME                READY   UP-TO-DATE   AVAILABLE   AGE  
nest-deployment      4/4     4             4           7h54m  
spring-deployment    3/3     3             3           4m52s  
PS C:\DevData\Kubernetes\demo2>
```

9. 환경변수가 잘 적용 됐는지 확인해보기



localhost:30000

myAccount: jaeseong, myPassword: pwd1234

10. 파드 내부로 접속해서 확인해보기

```
$ kubectl get pods # 파드명 확인하기  
$ kubectl exec -it [파드명] -- bash # 파드 내부로 접속하기  
  
$ env # 환경변수 조회
```

```
PS C:\DevData\Kubernetes\demo> kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nest-deployment-547f5764c8-n2fw9    1/1     Running   1 (10m ago) 7h42m
nest-deployment-547f5764c8-tzxlrv    1/1     Running   1 (10m ago) 7h42m
nest-deployment-547f5764c8-w4qj5     1/1     Running   1 (10m ago) 7h42m
nest-deployment-547f5764c8-w6spw     1/1     Running   1 (10m ago) 7h42m
spring-deployment-bcc5f479d-ghctt    1/1     Running   0           2m47s
spring-deployment-bcc5f479d-hxdmj    1/1     Running   0           2m47s
spring-deployment-bcc5f479d-splc5    1/1     Running   0           2m47s
```

```
PS C:\DevData\Kubernetes\demo> kubectl exec -it spring-deployment-586f9c7bb5-cr5tc bash
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD]
[COMMAND] instead.
hash-4 4# env
MY_ACCOUNT=jaeseong
NEST_SERVICE_PORT_3000_TCP_PROTO=tcp
LANG=C.UTF-8
HOSTNAME=spring-deployment-586f9c7bb5-cr5tc
JAVA_HOME=/usr/java/openjdk-17
SPRING_SERVICE_PORT_8080_TCP_ADDR=10.96.231.117
SPRING_SERVICE_PORT=tcp://10.96.231.117:8080
NEST_SERVICE_SERVICE_HOST=10.98.178.59
KUBERNETES_PORT_443_TCP_PROTO=tcp
KUBERNETES_PORT_443_TCP_ADDR=10.96.0.1
NEST_SERVICE_PORT_3000_TCP_PORT=3000
JAVA_VERSION=17.0.2
KUBERNETES_PORT=tcp://10.96.0.1:443
PWD=/
HOME=/root
KUBERNETES_SERVICE_PORT_HTTPS=443
KUBERNETES_PORT_443_TCP_PORT=443
NEST_SERVICE_PORT_3000_TCP=tcp://10.98.178.59:3000
NEST_SERVICE_PORT_3000_TCP_ADDR=10.98.178.59
KUBERNETES_PORT_443_TCP=tcp://10.96.0.1:443
MY_PASSWORD=pwd1234
TERM=xterm
SPRING_SERVICE_PORT_8080_TCP=tcp://10.96.231.117:8080
```

2. 컨피그맵(ConfigMap)을 활용해 환경변수 분리하기

✓ 컨피그맵(ConfigMap)이란?

Spring Boot에서는 설정값을 **application.yml**으로 분리해서 관리한다. Nest.js에서도 설정값을 **.env**으로 분리해서 관리한다. 별도의 파일로 분리를 해서 관리함으로써 유지보수가 편리해지고 개발, 테스트, 프로덕션과 같은 환경 분리가 편해진다.

쿠버네티스에서는 파드(Pod), 디플로이먼트(Deployment), 서비스(Service)가 각각의 역할을 가지고 있는 것처럼 환경 변수를 관리하는 역할을 가진 오브젝트가 따로 존재한다. 그게 바로 **컨피그맵(ConfigMap)**이다.

✓ 디플로이먼트(Deployment)에 환경 변수의 정보를 같이 작성했을 때의 단점

spring-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment

# Deployment 기본 정보
metadata:
  name: spring-deployment # Deployment 이름

# Deployment 세부 정보
spec:
  replicas: 3 # 생성할 파드의 복제본 개수
  selector:
    matchLabels:
      app: backend-app # 아래에서 정의한 Pod 중 'app: backend-app'이라는 값을 가진 파드를 선택

# 배포할 Pod 정의
template:
  metadata:
    labels: # 레이블 (= 카테고리)
      app: backend-app
  spec:
    containers:
      - name: spring-container # 컨테이너 이름
        image: spring-server # 컨테이너를 생성할 때 사용할 이미지
        imagePullPolicy: IfNotPresent # 로컬에서 이미지를 먼저 가져온다. 없으면 레지스트리에서 가져온다.
        ports:
          - containerPort: 8080 # 컨테이너에서 사용하는 포트를 명시적으로 표현
```

```
env: # 환경변수 등록
- name: MY_ACCOUNT # Key 값
  value: 본인ID # Value 값
- name: MY_PASSWORD
  value: pwd1234
```

위 매니페스트 파일을 보면 디플로이먼트(Deployment)에 대한 내용과 환경 변수에 관련된 내용을 같이 작성했다. 이렇게 환경 변수를 디플로이먼트(Deployment) 내부에 작성하면 다른 환경(개발, 테스트, 프로덕션 등)에서 서버를 실행할 때 유연하게 설정 값을 변경하기 어려워진다.

✅ 컨피그맵(ConfigMap)을 활용해 환경변수 분리하기

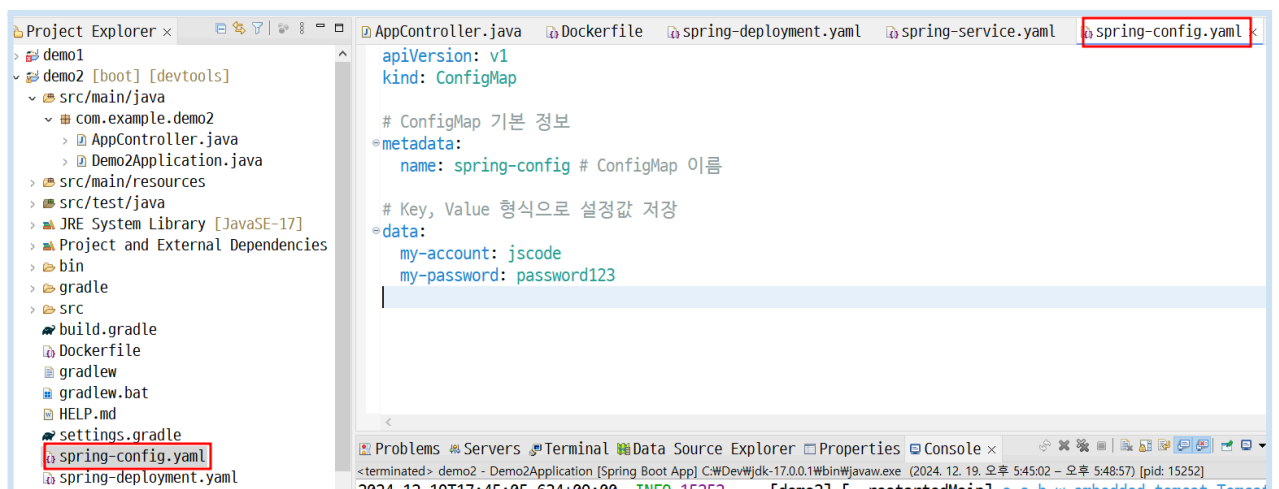
1. ConfigMap 매니페스트 파일 생성하기

spring-config.yaml

```
apiVersion: v1
kind: ConfigMap

# ConfigMap 기본 정보
metadata:
  name: spring-config # ConfigMap 이름

# Key, Value 형식으로 설정값 저장
data:
  my-account: kmdadoo1
  my-password: password123
```



- 참고) 컨피그맵이 잘 적용됐는 지 확인하기 위해 **my-account, my-password** 값 변경

2. Deployment 매니페스트 파일 수정하기

```
apiVersion: apps/v1
kind: Deployment

# Deployment 기본 정보
metadata:
  name: spring-deployment # Deployment 이름

# Deployment 세부 정보
spec:
  replicas: 5 # 생성할 파드의 복제본 개수
  selector:
    matchLabels:
      app: backend-app # 아래에서 정의한 Pod 중 'app: backend-app'이라는 값을
                        # 가진 파드를 선택

# 배포할 Pod 정의
template:
  metadata:
    labels: # 레이블 (= 카테고리)
      app: backend-app
  spec:
    containers:
      - name: spring-container # 컨테이너 이름
        image: spring-server # 컨테이너를 생성할 때 사용할 이미지
        imagePullPolicy: IfNotPresent # 로컬에서 이미지를 먼저 가져온다. 없으면
        레지스트리에서 가져온다.
        ports:
          - containerPort: 8080 # 컨테이너에서 사용하는 포트를 명시적으로 표현
        env:
          - name: MY_ACCOUNT
            valueFrom:
              configMapKeyRef:
                name: spring-config # ConfigMap의 이름
                key: my-account # ConfigMap에 설정되어 있는 Key값
          - name: MY_PASSWORD
            valueFrom:
              configMapKeyRef:
                name: spring-config
                key: my-password
```

3. 매니페스트 파일 반영하기

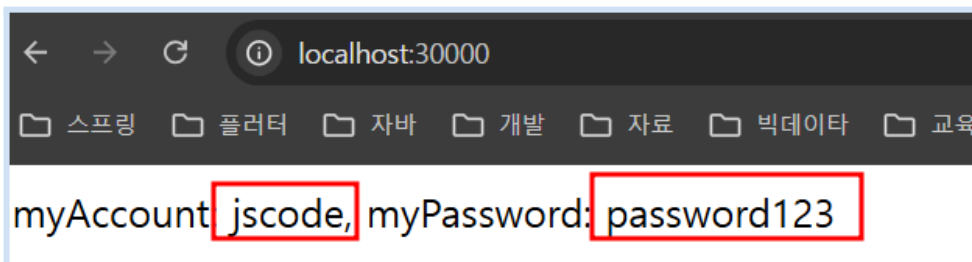
```
$ kubectl apply -f spring-config.yaml
$ kubectl apply -f spring-deployment.yaml
```

kubectl rollout restart deployment [디플로이먼트명]

\$ kubectl rollout restart deployment spring-deployment # Deployment 재시작

```
PS C:\DevData\Kubernetes\demo2> kubectl apply -f spring-config.yaml
configmap/spring-config created
PS C:\DevData\Kubernetes\demo2> kubectl get configmap
NAME          DATA  AGE
kube-root-ca.crt 1      3d8h
spring-config   2      58s
PS C:\DevData\Kubernetes\demo2> kubectl apply -f spring-deployment.yaml
deployment.apps/spring-deployment configured
PS C:\DevData\Kubernetes\demo2> kubectl rollout restart deployment spring-deployment
deployment.apps/spring-deployment restarted
PS C:\DevData\Kubernetes\demo2> kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nest-deployment-547f5764c8-n2fw9    1/1     Running   1 (36m ago)  8h
nest-deployment-547f5764c8-tzxlv    1/1     Running   1 (36m ago)  8h
nest-deployment-547f5764c8-w4qj5    1/1     Running   1 (36m ago)  8h
nest-deployment-547f5764c8-w6spw    1/1     Running   1 (36m ago)  8h
spring-deployment-58dbdc6986-6zf87  1/1     Running   0           73s
spring-deployment-58dbdc6986-74jcc  1/1     Running   0           73s
spring-deployment-58dbdc6986-dmdnb  1/1     Running   0           73s
spring-deployment-58dbdc6986-l9p8h  1/1     Running   0           62s
spring-deployment-58dbdc6986-rv4pz  1/1     Running   0           59s
```

4. 잘 반영 됐는지 확인하기



← → ↻ ⓘ localhost:30000

스프링 플러터 자바 개발 자료 빅데이터 교육

myAccount jscode, myPassword: password123

3. 시크릿(**Secret**)을 활용해 ‘민감한 값’을 환경 변수로 분리하기

✓ 시크릿(**Secret**)이란?

시크릿(**Secret**)은 컨피그맵(**ConfigMap**)과 비슷하게 환경 변수를 분리해서 관리하는 오브젝트이다. 차이점은 시크릿(**Secret**)은 비밀번호와 같이 보안적으로 중요한 값을 관리하기 위한 오브젝트이다.

✓ 시크릿(**Secret**)을 활용해 ‘민감한 값’을 따로 분리하기

1. 기존 매니페스트 파일 살펴보기

spring-config.yaml

```
apiVersion: v1
kind: ConfigMap

# ConfigMap 기본 정보
metadata:
  name: spring-config # ConfigMap 이름

# Key, Value 형식으로 설정값 저장
data:
  my-account: thejoeun
  my-password: password123
```

위 매니페스트 파일에서 **my_password**의 값이 보안적으로 중요한 값이라고 가정해보자. 그러면 **my_password**의 값은 컨피그맵(**ConfigMap**)이 아닌 시크릿(**Secret**)으로 관리해야 한다.

2. 기존 매니페스트 파일 수정하기 / 새로운 매니페스트 파일 생성하기 수정 (spring-config.yaml)

```
apiVersion: v1
kind: Secret

# ConfigMap 기본 정보
metadata:
  name: spring-config # ConfigMap 이름
```

```
# Key, Value 형식으로 설정값 저장
data:
  my-account: jscore
  my-password: password123
```

생성 (spring-secret.yaml)

```
apiVersion: v1
kind: Secret

# Secret 기본 정보
metadata:
  name: spring-secret # Secret 이름

# Key, Value 형식으로 값 저장
stringData:
  my-password: my-secret-password
```

- 참고) 시크릿이 잘 적용 됐는지 확인하기 위해 **my-password** 값 변경

수정 (spring-deployment.yaml)

```
apiVersion: apps/v1
kind: Deployment

# Deployment 기본 정보
metadata:
  name: spring-deployment # Deployment 이름

# Deployment 세부 정보
spec:
  replicas: 5 # 생성할 파드의 복제본 개수
  selector:
    matchLabels:
      app: backend-app # 아래에서 정의한 Pod 중 'app: backend-app'이라는 값을
      가진 파드를 선택

# 배포할 Pod 정의
template:
  metadata:
    labels: # 레이블 (= 카테고리)
      app: backend-app
```

```
spec:
  containers:
    - name: spring-container # 컨테이너 이름
      image: spring-server # 컨테이너를 생성할 때 사용할 이미지
      imagePullPolicy: IfNotPresent # 로컬에서 이미지를 먼저 가져온다. 없으면
레지스트리에서 가져온다.
      ports:
        - containerPort: 8080 # 컨테이너에서 사용하는 포트를 명시적으로 표현
      env:
        - name: MY_ACCOUNT
          valueFrom:
            configMapKeyRef:
              name: spring-config # ConfigMap의 이름
              key: my-account # ConfigMap에 설정되어 있는 Key값
        - name: MY_PASSWORD
          valueFrom:
            secretKeyRef:
              name: spring-secret
              key: my-password
```

3. 매니페스트 파일 반영하기

```
$ kubectl apply -f spring-secret.yaml
$ kubectl apply -f spring-config.yaml
$ kubectl apply -f spring-deployment.yaml
$ kubectl rollout restart deployment spring-deployment
```

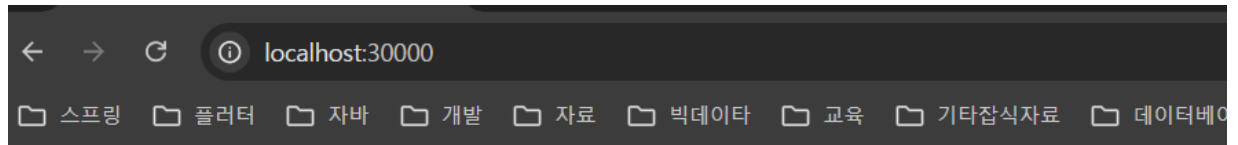
```
PS C:\DevData\Kubernetes\demo2> kubectl apply -f spring-secret.yaml
secret/spring-secret created
PS C:\DevData\Kubernetes\demo2> kubectl get secret
NAME          TYPE      DATA  AGE
spring-secret  Opaque    1      3s
```

```
PS C:\DevData\Kubernetes\demo2> kubectl apply -f spring-config.yaml
configmap/spring-config configured
```

```
PS C:\DevData\Kubernetes\demo2> kubectl apply -f spring-deployment.yaml
deployment.apps/spring-deployment configured
PS C:\DevData\Kubernetes\demo2> kubectl rollout restart deployment spring-deployment
deployment.apps/spring-deployment restarted
```

```
PS C:\DevData\Kubernetes\demo2> kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nest-deployment-547f5764c8-n2fw9    1/1     Running   2 (13m ago) 23h
nest-deployment-547f5764c8-tzxlv    1/1     Running   2 (13m ago) 23h
nest-deployment-547f5764c8-w4qj5    1/1     Running   2 (13m ago) 23h
nest-deployment-547f5764c8-w6spw    1/1     Running   2 (13m ago) 23h
spring-deployment-594f467ccb-9jx8q   1/1     Running   0           51s
spring-deployment-594f467ccb-f7jfv   1/1     Running   0           65s
spring-deployment-594f467ccb-hgm9w   1/1     Running   0           65s
spring-deployment-594f467ccb-rrkx4   1/1     Running   0           51s
spring-deployment-594f467ccb-rvbqv   1/1     Running   0           65s
```

4. 잘 반영 됐는지 확인하기



myAccount: thejoeun, myPassword: my-secret-password