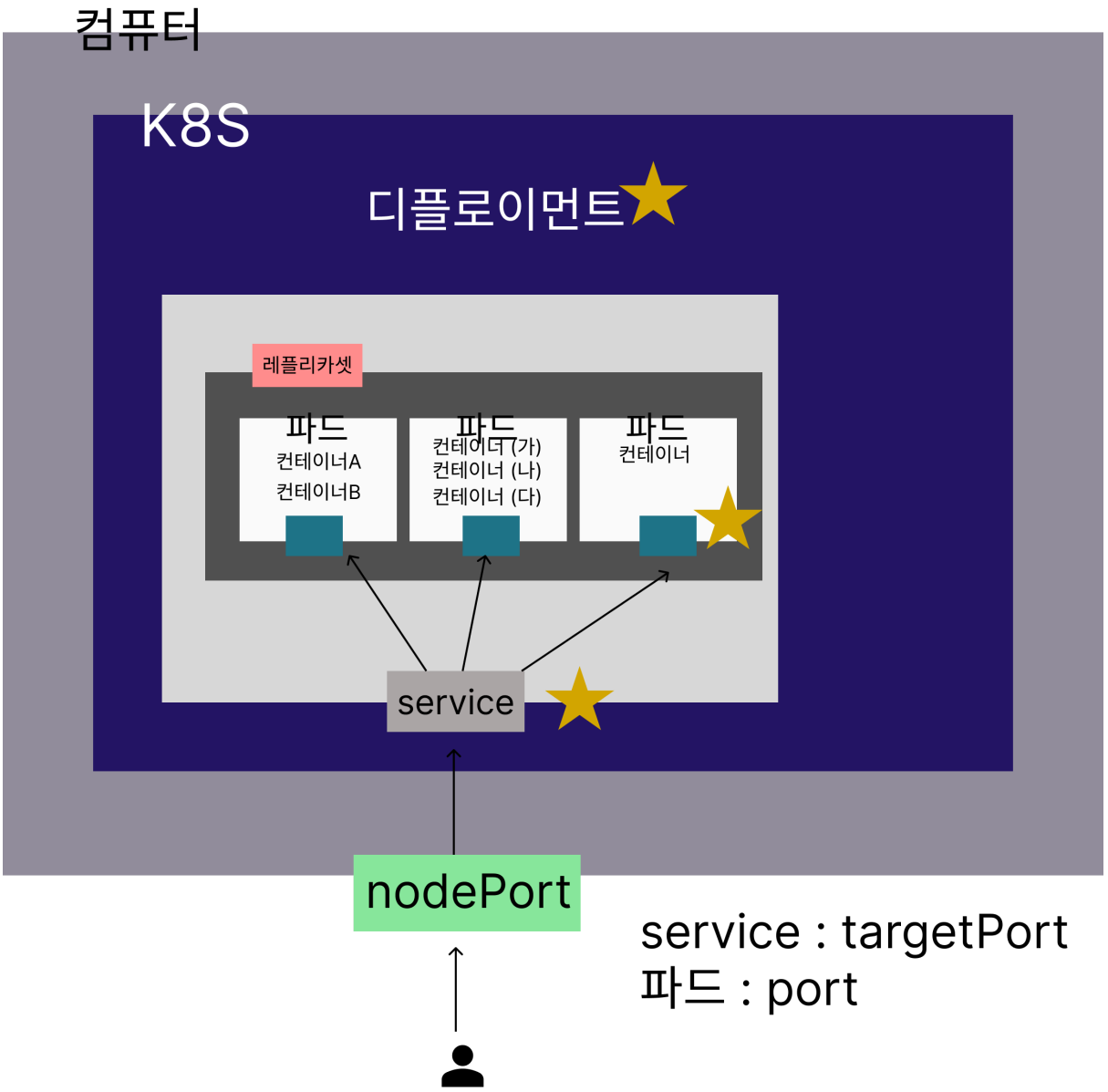
재정리



도커와 쿠버네티스

K8S에서는 kubectl + 디플로이먼트, 레플리카셋, 파드, 서비스 명령어를 쓴다.

Docker에서는

도커, 왜 쓸까?

매번 귀찮은 설치 과정 일일이 거치지 않아도 됨
항상 일관되게 프로그램 설치 : 버전/환경설정/옵션/운영체제

각 프로그램이 독립적인 환경에서 실행되기 때문에, 프로그램간에 서로 충돌이 일어나지 않는다.
docker run -d -p 8080:80 --name web-server nginx:1.25 // 이미지로부터 실제 실행 환경 만들기.

K8S, 왜 쓸까?

도커의 명령어를 yaml 파일에 저장, 사용함으로써 더 빠른 인프라 구축 가능
[nginx-deployment.yaml]========================== // 이미지로부터 실제 실행 환경 만들기.
apiVersion: apps/v1 # 사용할 Kubernetes API 버전 (Deployment는 apps/v1)
kind: Deployment # 이 리소스의 종류는 Deployment (애플리케이션 배포용)
metadata:
name: web-server # 이 Deployment의 이름
spec:
replicas: 1 # 실행할 파드 수 (복제 수)
selector:
matchLabels:
app: web-server # 어떤 파드를 관리할지 기준이 되는 라벨
template:
metadata:
labels:
app: web-server # 위 selector와 매칭되도록 동일한 라벨 지정
spec:
containers:
- name: nginx # 컨테이너 이름
image: nginx:1.25 # 사용할 Docker 이미지 (nginx 버전 1.25)
ports:
- containerPort: 80 # 컨테이너 내부에서 열리는 포트

[nginx-service.yaml]============================== // 이미지로부터 실제 실행 환경 만들기.
apiVersion: v1 # 서비스 리소스는 core API 그룹(v1) 사용
kind: Service # 리소스 종류는 Service (네트워크 접근을 위한)
metadata:
name: web-server-service # 서비스 이름
spec:

selector:
app: web-server # 이 서비스가 연결할 Pod의 라벨 (Deployment와 일치해야 함)
ports:
- protocol: TCP # 통신 프로토콜 (HTTP는 TCP 기반)
port: 8080 # 클러스터 내부에서 사용할 포트
targetPort: 80 # 실제 Pod의 컨테이너에서 노출된 포트
type: NodePort # 외부에서 접근 가능하도록 NodePort 방식 사용

```
## 디플로이먼트를 사용하면 애플리케이션의 **파드(Pod)**와
**레플리카셋(ReplicaSet)**을 생성하고,
업데이트하며, 롤백까지 손쉽게 수행할 수 있습니다.
## 서비스? 외부 트래픽을 균등 분배 == 로드 밸런서
```

kubectl apply -f nginx-deployment.yaml
kubectl apply -f nginx-service.yaml

[바로 위 실행단계]
mvnw clean install // 윈도우의 경우. // linux ubuntu는 mvn clean install

docker build -t spring-server .
// docker는 도커 이미지를 생성(build) 하라는 명령입니다.

kubectl delete pod --all
kubectl delete deployment --all
kubectl delete service --all
// service, deployment, pod 는 k8s 내의 구성요소이다

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

==========
단어장

kubectl get pods

kubectl get service

kubectl get deployment

kubectl delete pod [pod 이름] —all // 모두 삭제 (—a)

kubeclt delete service [service name] —all

kubectl delete deployment [deployment name]

docker ps -a // 컨테이너 모두 출력 (중지,실행 전부 다)

docker rm -f [컨테이너 이름]

docker rmi -f [이미지 이름]

파드 조회
kubectl get pods

파드 포트 포워딩
kubectl port-forward pod/파드명 로컬에서의포트:파드에서의포트
kubectl port-forward pod/nginx-pod 80:80

파드 삭제
kubectl delete pod 파드명
kubectl delete pod nginx-pod

파드 디버깅

1. 파드 세부 정보 조회하기
   kubectl describe pods 파드명
   kubectl describe pods nginx-pod
2. 파드 로그 확인하기
   kubectl logs 파드명
   kubectl logs nginx-pod
3. 파드 내부로 접속하기
   kubectl exec -it 파드명 — bash
   kubectl exec -it nginx-pod — bash

메니페스트 파일에 적혀져 있는 리소드(파드 등) 생성

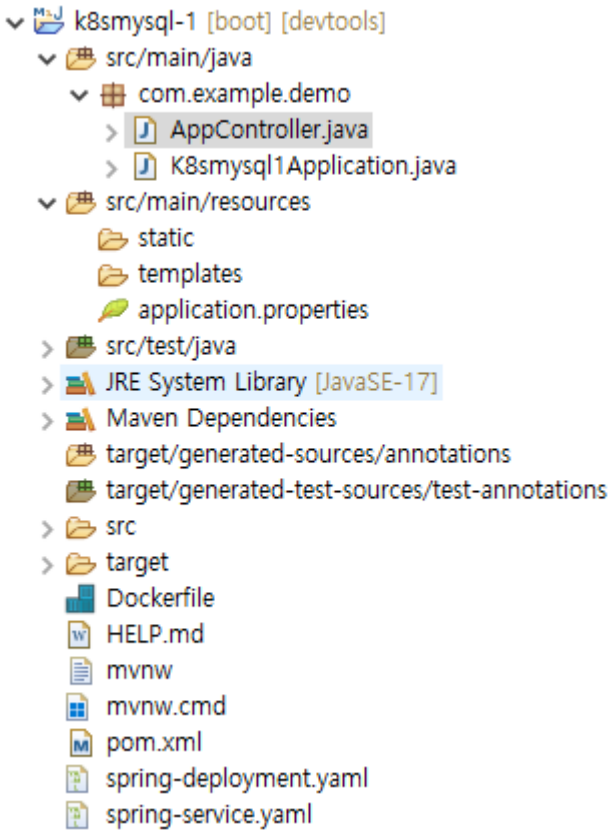
kubectl apply -f 파일명
kubectl apply -f nginx-pod.yaml

```
mvnw clean install  // 윈도우의  경우. // linux ubuntu는 mvn clean install

docker build -t spring-server .
// docker는 도커 이미지를 생성(build) 하라는 명령입니다.
// 현재 디렉토리에 있는 Dockerfile을 기반으로

kubectl delete pod --all
kubectl delete deployment --all
kubectl delete service --all
// service, deployment, pod 는 k8s 내의 구성요소이다


kubectl apply -f spring-deployment.yaml  // 변경적용
kubectl apply -f spring-service.yaml
```




```
package com.example.demo;

import org.springframework.web.bind.annotation.GetMapping;
```

```java
import org.springframework.web.bind.annotation.RestController;

@RestController
public class AppController {

    @GetMapping("/")
    public String home () {
        return "hi there, how are you doing?";
    }
}
```

Dockerfile

```
FROM openjdk:17-jdk

COPY target/*SNAPSHOT.jar app.jar

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

spring-deployment.yaml

```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 가

  # 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 레지
        ports:
          - containerPort: 8080  # 컨테이너에서 사용하는 포트를 명시적으로 표현
```

```
apiVersion: v1
kind: Service

# Service 기본 정보
metadata:
  name: spring-service # Service 이름

# Service 세부 정보
spec:
  type: NodePort # Service의 종류
  selector:
    app: backend-app # 실행되고 있는 파드 중 'app: backend-app'이라는 값을 가진
  ports:
    - protocol: TCP # 서비스에 접속하기 위한 프로토콜
      port: 8080 # 쿠버네티스 내부에서 Service에 접속하기 위한 포트 번호
      targetPort: 8080 # 매핑하기 위한 파드의 포트 번호
      nodePort: 30000 # 외부에서 사용자들이 접근하게 될 포트 번호
```