쿠버네티스 발표

2장 파드와 디플로이먼트로 컨테이너 실행하기

26.01.11 David

2.1 쿠버네티스는 어떻게 컨테이너를 실행하고 관리하는가?

파드(Pod) - 쿠버네티스의 기본 단위

> “쿠버네티스는 컨테이너가 아닌, 파드(Pod)를 관리한다”

- 컨테이너 = 애플리케이션 구성요소 하나를 실행하는 가상화된 환경
- 파드(Pod) = 컨테이너를 감싸는 쿠버네티스의 최소 배포 단위
- 파드는 클러스터를 이루는 노드(Node) 중 하나에서 실행됨.

2.1 쿠버네티스는 어떻게 컨테이너를 실행하고 관리하는가?

파드(Pod) - 쿠버네티스의 기본 단위

"왜 쿠버네티스는 컨테이너가 아니라 파드를 관리할까?"

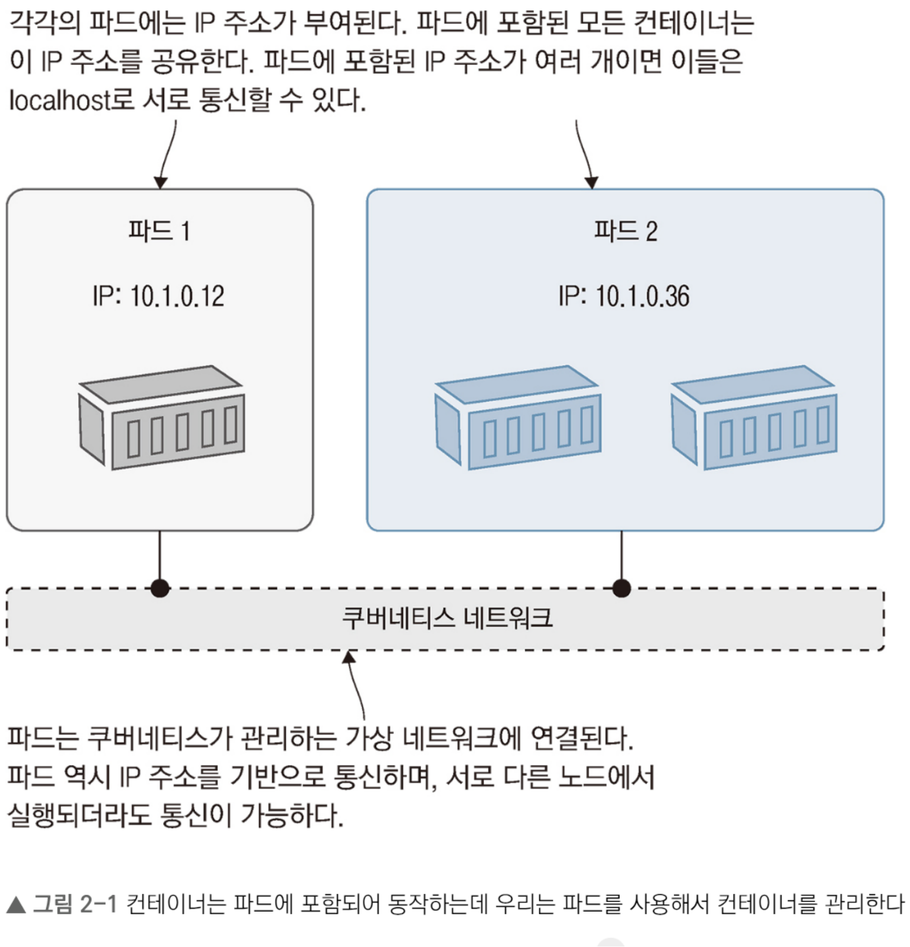
- 컨테이너 단위 관리 → 네트워크/스토리지 공유가 어려움
- 실제 서비스는 여러 프로세스가 한 덩어리로 움직여야 함
- 파드는 "함께 죽고, 함께 살아야 하는 컨테이너 묶음"

2.1 쿠버네티스는 어떻게 컨테이너를 실행하고 관리하는가?

파드의 네트워킹

"모든 파드는 고유한 가상 IP를 가진다"

- 파드는 쿠버네티스가 관리하는 자신만의 가상 IP 주소를 보유
- 이 IP로 가상 네트워크에 접속된 다른 파드와 통신 가능
- 심지어 다른 노드에서 실행되는 파드와도 통신 가능



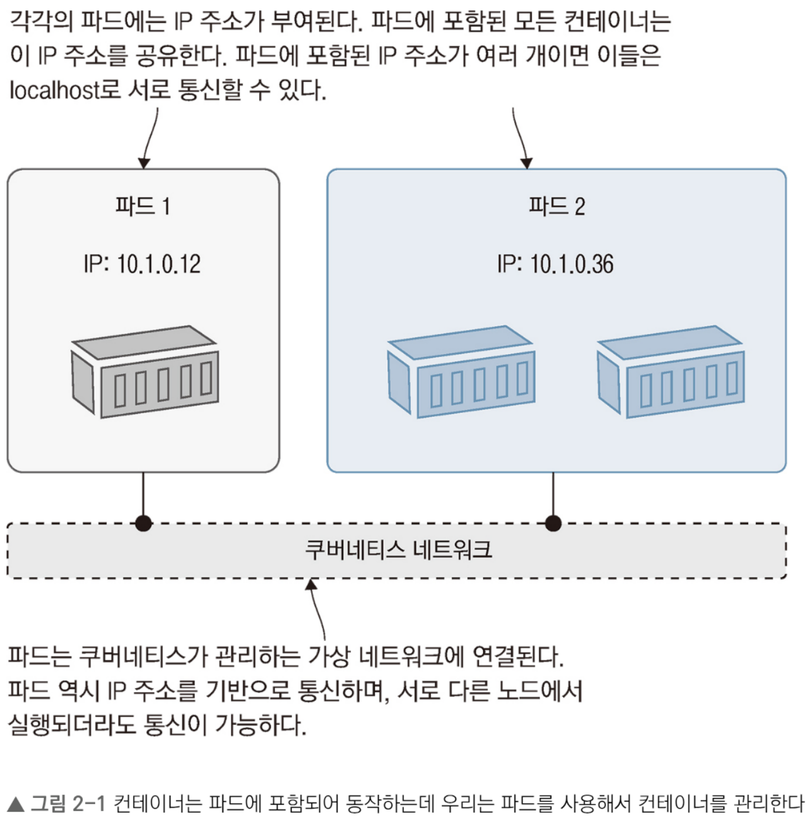
▲ 그림 2-1 컨테이너는 파드에 포함되어 동작하는데 우리는 파드를 사용해서 컨테이너를 관리한다

2.1 쿠버네티스는 어떻게 컨테이너를 실행하고 관리하는가?

멀티 컨테이너 파드

> "하나의 파드에 여러 컨테이너를 담을 수 있다"

- 파드 하나는 대개 컨테이너 하나를 포함
- 설정에 따라 여러 개의 컨테이너 포함 가능
- 같은 파드 내 컨테이너들은
  - 네트워크 공유 (같은 IP)
  - localhost로 서로 통신 가능
  - 스토리지 볼륨 공유 가능



▲ 그림 2-1 컨테이너는 파드에 포함되어 동작하는데 우리는 파드를 사용해서 컨테이너를 관리한다

2.1 쿠버네티스는 어떻게 컨테이너를 실행하고 관리하는가?

[실습] 명령어로 파드 실행하기

"간단한 파드는 YAML 없이도 바로 실행 가능"

```
david@davidui-MacBookAir ~ % kubectl run nginx --image nginx
pod/nginx created
david@davidui-MacBookAir ~ % kubectl wait --for=condition=Ready pod nginx
pod/nginx condition met
david@davidui-MacBookAir ~ % kubectl get pods
NAME    READY   STATUS    RESTARTS   AGE
nginx   1/1     Running   0          44s
```

-> 컨테이너 하나를 담은 파드 생성

-> 파드가 준비 될 때까지 대기

-> 클러스터에 있는 모든 파드 목록 출력

2.1 쿠버네티스는 어떻게 컨테이너를 실행하고 관리하는가?

[실습] 파드 상세 정보 확인

```
david@davidui-MacBookAir ~ % kubectl describe pod nginx
Name:             nginx
Namespace:        default
Priority:         0
Service Account:  default
Node:             docker-desktop/192.168.65.3
Start Time:       Sat, 10 Jan 2026 23:30:49 +0900
Labels:           run=nginx
Annotations:      <none>
Status:           Running
IP:               10.1.0.6
IPs:
  IP:  10.1.0.6
Containers:
  nginx:
    Container ID:   docker://d34b205c0d2bdf910a1a7c194fef61878531641194f20340feb6c9313a6ee5f5
    Image:          nginx
    Image ID:       docker-pullable://nginx@sha256:7272239bd21472f311aa3e86a85fdca0f1ad648995f983ab6e5e7dea665cd233
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Sat, 10 Jan 2026 23:30:56 +0900
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-p8wkv (ro)
```

-> IP 주소와 파드를 실행하는
노드 등 특정 파드에 대한 상세 정보 출력

2.1 쿠버네티스는 어떻게 컨테이너를 실행하고 관리하는가?

[실습] 파드 기본 정보 확인

```
david@davidui-MacBookAir ~ % kubectl get pod nginx
NAME    READY   STATUS    RESTARTS   AGE
nginx   1/1     Running   0          10m
david@davidui-MacBookAir ~ % kubectl get pod nginx --output custom-columns=NAME:metadata.name,NODE_IP:status.hostIP,POD_IP:status.podIP
NAME    NODE_IP        POD_IP
nginx   192.168.65.3   10.1.0.6
david@davidui-MacBookAir ~ % kubectl get pod nginx -o jsonpath='{.status.containerStatuses[0].containerID}'
docker://d34b205c0d2bdf910a1a7c194fef61878531641194f20340feb6c9313a6ee5f5%
```

kubectl get pod nginx

-> nginx에 대한 기본적인 정보 확인

kubectl get pod nginx —output custom-columns=NAME:metadata.name,NODE_IP:…

-> 이름과 데이터의 JSON 형태로 항목 정의해서 출력, 현재는 NAME, NODE_IP, POD_IP를 예시로 듦

kubectl get pod nginx -o jsonpath=‘{.status.containerStatuses[0].containerID}’

-> 파드에 포함된 첫 번째 컨테이너의 컨테이너 식별자를 출력하라는 의미.

2.1 쿠버네티스는 어떻게 컨테이너를 실행하고 관리하는가?

[실습] 쿠버네티스의 파드로 실행된 컨테이너를 컨테이너 런타임이 어떻게 유지하나?

```
david@davidui-MacBookAir ~ % docker container ls -q --filter label=io.kubernetes.container.name=nginx
d34b205c0d2b
david@davidui-MacBookAir ~ % docker container rm -f $(docker container ls -q --filter label=io.kubernetes.container.name=nginx)
d34b205c0d2b
david@davidui-MacBookAir ~ % kubectl get pod nginx
NAME    READY   STATUS    RESTARTS   AGE
nginx   1/1     Running   1          13m
david@davidui-MacBookAir ~ % docker container ls -q --filter label=io.kubernetes.container.name=nginx
1b53a8246839
```

컨테이너의 식별자가 다름, 쿠버네티스가 새로운 컨테이너로 파드 복원함

docker container ls -q —filter label=io.kubernetes.container.name=nginx

-> 파드에 포함된 컨테이너 찾기 (쿠버네티스는 컨테이너를 만들 때 파드 이름을 컨테이너 레이블에 추가함)

docker container rm -f $(docker container ls -q —filter label=io.kubernetes…)

-> 해당 컨테이너 삭제하기

kubectl get pod nginx

-> 파드 상태 확인

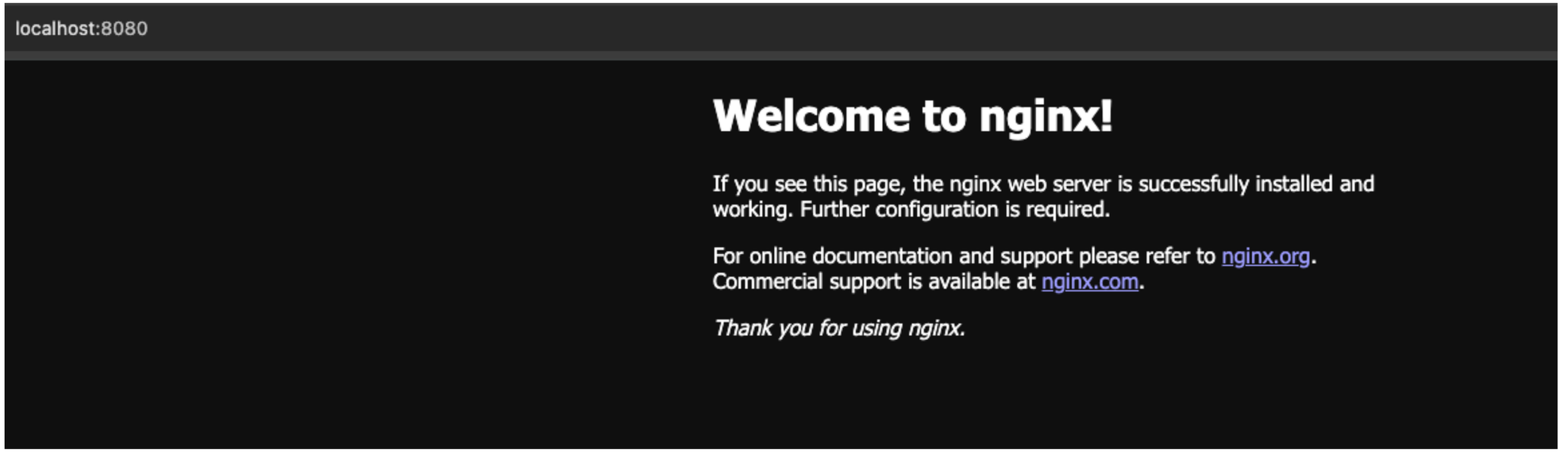
2.1 쿠버네티스는 어떻게 컨테이너를 실행하고 관리하는가?

[실습] 포트 포워딩으로 접근

```
david@davidui-MacBookAir ~ % kubectl port-forward pod/nginx 8080:80
Forwarding from 127.0.0.1:8080 -> 80
Forwarding from [::1]:8080 -> 80
```

-> 로컬 컴퓨터의 8080번 포트를 주시하다가

이 포트로 들어오는 트래픽을 파드의 80번 포트로 전달



2.2 컨트롤러 객체와 함께 파드 실행하기

파드의 한계

> “파드는 일시적이므로, 죽으면 끝임”

문제점

- 파드가 장애로 종료되면? -> 자동 복구 기능 없음
- 트래픽이 증가하면? -> 자동 확장 없음
- 새 버전 배포하면? -> 수동으로 교체해야 함.

-> 파드는 자기 자신을 복구하지 못하므로 Deployment 같은 컨트롤러가 필요하다.

2.2 컨트롤러 객체와 함께 파드 실행하기

컨트롤러 객체란?

> “다른 리소스를 관리하는 쿠버네티스 리소스”

컨트롤러는 쿠버네티스의 API와 연동하며 시스템의 현재 상태를 감시하다가 '바람직한 상태'와 차이가 생기면 필요에 따라 그 차이를 바로 잡는다.

쿠버네티스에는 여러 가지 컨트롤러 객체들이 있다.

파드를 주로 관리하는 컨트롤러 객체는 디플로이먼트다.

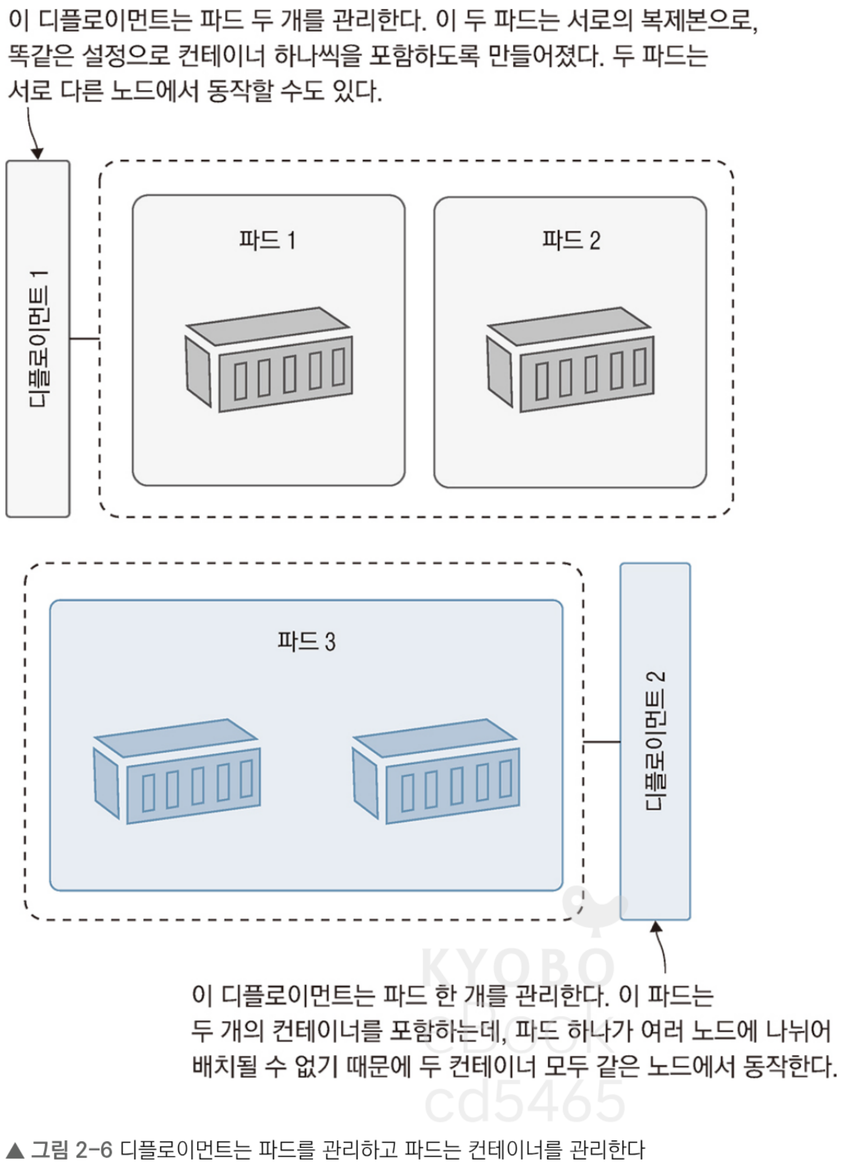
2.2 컨트롤러 객체와 함께 파드 실행하기

디플로이먼트(Deployment) - 파드의 관리자

> "디플로이먼트는 파드를 관리하는 컨트롤러다."

디플로이먼트 핵심기능 3가지

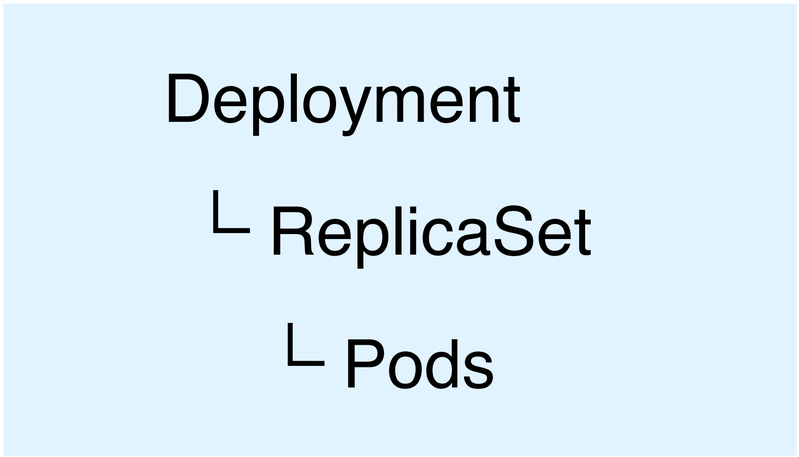
기능	설명
Self-healing	파드가 죽으면 자동으로 되살림
Scaling	파드 개수를 원하는 만큼 복제
Rolling Update	무중단 버전 업데이트 (하나씩 교체)



▲ 그림 2-6 디플로이먼트는 파드를 관리하고 파드는 컨테이너를 관리한다

2.2 컨트롤러 객체와 함께 파드 실행하기

디플로이먼트(Deployment) 내부 구조



설명

- Deployment는 직접 파드를 만들지 않는다.
- ReplicaSet을 통해 파드를 관리한다.
- 롤링 업데이트, 롤백의 핵심이 Deployment에 있음.

2.2 컨트롤러 객체와 함께 파드 실행하기

[실습] 디플로이먼트(Deployment) - 파드의 관리자

```
david@davidui-MacBookAir ~ % kubectl create deployment nginx2 --image=nginx
deployment.apps/nginx2 created
david@davidui-MacBookAir ~ % kubectl get pods
NAME                      READY   STATUS    RESTARTS   AGE
nginx                     1/1     Running   1          48m
nginx2-78f4b4ddd7-h4xhw   1/1     Running   0          7s
```

디플로이먼트가 생성한 파드 이름은 컨트롤러 객체 이름 뒤에 무작위 문자열을 덧붙임

kubectl create deployment nginx2 —image=nginx

-> nginx2라는 이름의 디플로이먼트 생성, 파드의 복제본 수를 지정하지 않아 기본값인 파드 1개 생성

2.2 컨트롤러 객체와 함께 파드 실행하기

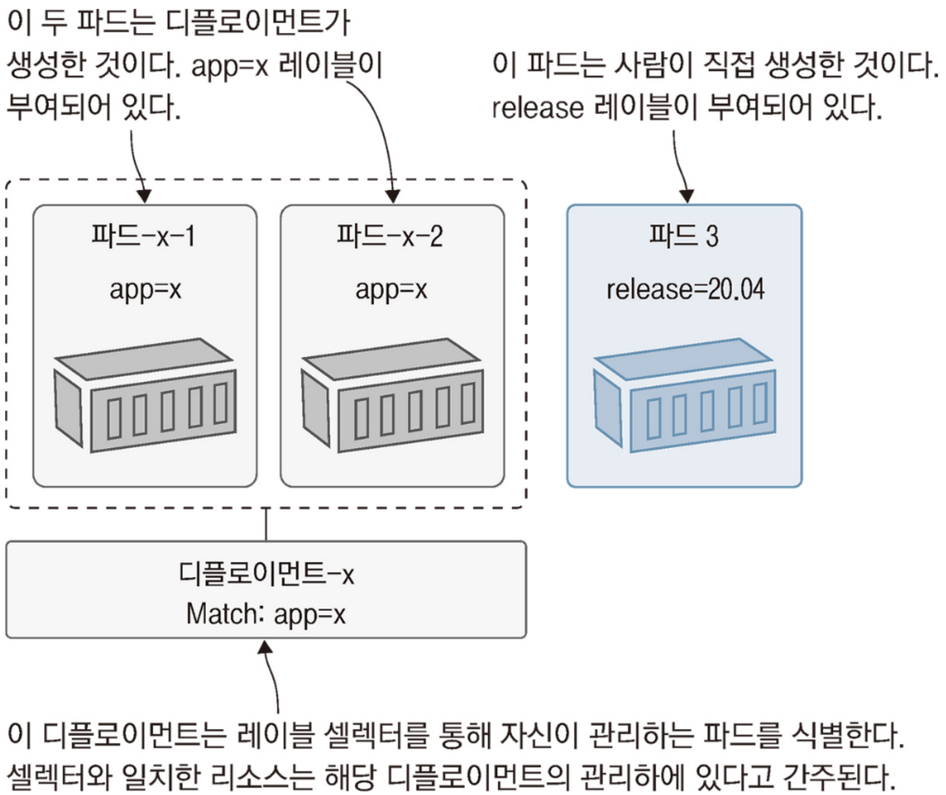
디플로이먼트와 파드의 연결 고리

"Deployment는 내 파드를 어떻게 찾을까?"

이름(Name)이 아닌 라벨(Label)로 관리

Label Selector

- 서비스(Service) <-> 파드(Pod) 연결
- 디플로이먼트(Deployment) <-> 파드(Pod) 관리
- 모든 연결의 핵심



▲ 그림 2-9 컨트롤러 객체는 레이블 셀렉터를 통해 자신이 관리하는 리소스를 식별한다

2.2 컨트롤러 객체와 함께 파드 실행하기

[실습] 디플로이먼트가 관리하는 리소스 추적하는 방법

```
david@davidui-MacBookAir ~ % kubectl get deploy nginx2 -o jsonpath='{.spec.template.metadata.labels}'
{"app":"nginx2"}%
david@davidui-MacBookAir ~ % kubectl get pods -l app=nginx2
NAME                      READY   STATUS    RESTARTS   AGE
nginx2-78f4b4ddd7-h4xhw   1/1     Running   0          90s
```

kubectl get deploy nginx2 -o jsonpath=‘{.spec.template.metadata.labels}’

-> 디플로이먼트가 부여한 파드의 레이블 출력

kubectl get pods -l app=nginx2

-> 레이블 셀렉터를 사용하여 레이블 이름은 “app”, 값은 “nginx2”인 파드 목록을 출력

2.2 컨트롤러 객체와 함께 파드 실행하기

[실습] 파드의 레이블을 수정해보자

```
david@davidui-MacBookAir ~ % kubectl get pods -o custom-columns=NAME:metadata.name,LABELS:metadata.labels
NAME                      LABELS
nginx                     map[run:nginx]
nginx2-78f4b4ddd7-h4xhw   map[app:nginx2 pod-template-hash:78f4b4ddd7]
david@davidui-MacBookAir ~ % kubectl label pods -l app=nginx2 --overwrite app=nginxx
pod/nginx2-78f4b4ddd7-h4xhw labeled
david@davidui-MacBookAir ~ % kubectl get pods -o custom-columns=NAME:metadata.name,LABELS:metadata.labels
NAME                      LABELS
nginx                     map[run:nginx]
nginx2-78f4b4ddd7-h4xhw   map[app:nginxx pod-template-hash:78f4b4ddd7]
nginx2-78f4b4ddd7-lx976   map[app:nginx2 pod-template-hash:78f4b4ddd7]
```

레이블이 수정되면서 디플로이입장에서는 파드가 유실되었으므로 새로운 파드 생성함.

kubectl get pods -o custom-columns=NAME:metadata.name,LABELS:metadata.labels

-> 모든 파드 이름과 레이블 확인

kubectl label pods -l app=nginx2 —overwrite app=nginxx

-> 디플로이먼트가 생성한 파드의 ‘app’ 레이블 수정

- **컨트롤러 객체는 사라진 파드를 대신하여 대체 파드를 생성하고, 애플리케이션은 성능 손실 없이 계속 동작 가능.**

2.2 컨트롤러 객체와 함께 파드 실행하기

[실습] 디플로이먼트가 쿠버네티스 API를 통해 중복 삭제 과정

```
david@davidui-MacBookAir ~ % kubectl get pods -l app -o custom-columns=NAME:metadata.name,LABELS:metadata.labels
NAME                      LABELS
nginx2-78f4b4ddd7-h4xhw   map[app:nginxx pod-template-hash:78f4b4ddd7]
nginx2-78f4b4ddd7-lx976   map[app:nginx2 pod-template-hash:78f4b4ddd7]

david@davidui-MacBookAir ~ % kubectl label pods -l app=nginxx --overwrite app=nginx2
pod/nginx2-78f4b4ddd7-h4xhw labeled
david@davidui-MacBookAir ~ % kubectl get pods -l app -o custom-columns=NAME:metadata.name,LABELS:metadata.labels
NAME                      LABELS
nginx2-78f4b4ddd7-h4xhw   map[app:nginx2 pod-template-hash:78f4b4ddd7]
```

하나는 디플로이먼트가 관리, 다른 하나는 레이블 수정 후 디플로이먼트 관리에서 벗어남.

kubectl get pods -o custom-columns=NAME:metadata.name,LABELS:metadata.labels

-> 모든 파드 이름과 레이블 확인

kubectl label pods -l app=nginxx —overwrite app=nginx2

-> 디플로이먼트 관리 벗어난 파드의 ‘app’ 레이블을 원래대로 수정

2.2 컨트롤러 객체와 함께 파드 실행하기

쿠버네티스의 컨테이너 상태 관리: Probes

"단순히 프로세스만 살아있다면 건강한 걸까?"

쿠버네티스의 기능 (Probes)

- Liveness Probe (헬스체크)

  "너 살아 있지?" -> 대답 없으면 파드를 재시작

- Readiness Probe (준비 가능한 상태인지 체크)

  "트래픽 받을 준비 되어 있지?" -> 아직이라면 트래픽 차단(로딩 중 보호)

2.3 애플리케이션 매니페스트에 배포 정의하기

매니페스트(Manifest)란?

> “쿠버네티스 오브젝트를 정의한 설계도(명세서)”

특징

- 주로 YAML 형식으로 작성
- 모든 배포를 코드로 관리 (Infrastructure as Code)
- 버전 관리 시스템(Git)으로 추적 가능

2.3 애플리케이션 매니페스트에 배포 정의하기

[예시] 기후동행 디플로이먼트 YAML

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: climate-card          # 디플로이먼트 이름
spec:
  replicas: 2                 # 파드 개수(기본값1)
  selector:
    matchLabels:              # 레이블 셀렉터 정의
      app: climate-card
```

YAML의 필수 요소

필드	설명	예시
apiVersion	API 버전	apps/v1
kind	오브젝트 종류	Deployment
metadata	이름, 라벨 등 메타정보	name: climate-card
spec	바라는 상태 정의	이미지, 파드 개수 등

2.3 애플리케이션 매니페스트에 배포 정의하기

[예시] 기후동행 디플로이먼트 YAML

```
template:                          #. 파드 템플릿
  metadata:
    labels:
      app: climate-card
  spec:                            # 파드에는 컨테이너 이름과 이미지 이름 정의
    containers:
    - name: app
      image: traefik/whoami        # 컨테이너 이미지
      ports:
      - containerPort: 80
```

YAML의 필수 요소

필드	설명	예시
apiVersion	API 버전	`apps/v1`
kind	오브젝트 종류	`Deployment`
metadata	이름, 라벨 등 메타정보	`name: climate-card`
spec	바라는 상태 정의	이미지, 파드 개수 등

2.4 파드에서 실행 중인 애플리케이션에 접근하기

파드 내부 상황 파악하기

"서비스가 안될 때, 파드 안에는 어떻게 들어갈까?"

kubectl exec: 컨테이너 내부 접속(대화형 셸)

원격 서버에 접속하는 파드 내부로 들어가서 명령어 실행

설정 파일 확인(cat), 네트워크 연결(ping, curl) 테스트 가능

명령어:

kubectl exec -it [파드 이름] — sh (-i는 표준 입력, -t는 터미널 할당)

kubectl logs: 애플리케이션 로그 확인

가장 쉽고 빠른 문제 해결 수단(디버깅용)

컨테이너가 화면(표준 출력)에 나타내는 로그를 열람

명령어: Kubectl logs [파드 이름]

2.4 파드에서 실행 중인 애플리케이션에 접근하기

파드와 파일 주고받기

> “컨테이너 안에 있는 파일을 내 컴퓨터로 가져오려면?”

kubectl cp: 파일 시스템 접근

로컬 컴퓨터와 파드 컨테이너 사이의 파일 복사

컨테이너 내부의 로그 파일을 로컬로 가져오거나, 수정된 설정 파일을 밀어 넣을때 사용

명령어:

kubectl cp [로컬파일] [파드이름]:[컨테이너내부경로]

kubectl cp [파드이름]:[컨테이너내부경로] [로컬경로]

2.5 쿠버네티스의 리소스 관리 이해하기

[실습] 삭제한 리소스가 살아나는 경우

```
david@davidui-MacBookAir ~ % kubectl get pods
NAME                      READY   STATUS    RESTARTS   AGE
nginx                     1/1     Running   1          72m
nginx2-78f4b4ddd7-h4xhw   1/1     Running   0          24m
david@davidui-MacBookAir ~ % kubectl delete pods --all
pod "nginx" deleted from default namespace
pod "nginx2-78f4b4ddd7-h4xhw" deleted from default namespace
david@davidui-MacBookAir ~ % kubectl get pods
NAME                      READY   STATUS    RESTARTS   AGE
nginx2-78f4b4ddd7-w968s   1/1     Running   0          5s
```

kubectl delete pods —all

-> 모든 유형의 리소스를 한 번에 삭제하였으나, 파드가 1개가 다시 생성됨.

디플로이먼트가 자신이 관리하던 파드가 삭제되자 이를 대체할 새로운 파드 생성함.

2.5 쿠버네티스의 리소스 관리 이해하기

[실습] 삭제한 리소스가 살아나는 경우

```
david@davidui-MacBookAir ~ % kubectl get deploy
NAME     READY   UP-TO-DATE   AVAILABLE   AGE
nginx2   1/1     1            1           24m
david@davidui-MacBookAir ~ % kubectl delete deploy --all
deployment.apps "nginx2" deleted from default namespace
david@davidui-MacBookAir ~ % kubectl get pods
No resources found in default namespace.
david@davidui-MacBookAir ~ % kubectl get all
NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   75m
```

kubectl get deploy

-> nginx2라는 디플로이먼트 확인

kubectl delete deploy —all

-> deploy 모두 삭제 후 쿠버네티스 API만 남아 있는 것을 확인

2.6 연습문제

기후동행 앱을 배포하는 디플로이먼트 YAML을 작성하고, 실행 후 접속을 확인하기

climate-card-deploy.yaml U

solution > climate-card-deploy.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: climate-card
spec:
  replicas: 1
  selector:
    matchLabels:
      app: climate-card
  template:
    metadata:
      labels:
        app: climate-card
    spec:
      containers:
      - name: climate-card
        image: traefik/whoami
        ports:
        - containerPort: 80

```

필드	설명
apiVersion: apps/v1	Deployment API 버전
kind: Deployment	리소스 종류
metadata.name	Deployment 이름 (climate-card)
replicas: 1	유지할 파드 개수
selector.matchLabels	관리할 파드를 식별하는 라벨
template	생성할 파드의 설계도
image	컨테이너에 사용할 Docker 이미지
containerPort	컨테이너가 수신할 포트

2.6 연습문제

기후동행 앱을 배포하는 디플로이먼트 YAML을 작성하고, 실행 후 접속을 확인하기

```
david@davidui-MacBookAir 기후동행 % kubectl apply -f solution/climate-card-deploy.yaml
deployment.apps/climate-card created
david@davidui-MacBookAir 기후동행 % kubectl get deployments
NAME           READY   UP-TO-DATE   AVAILABLE   AGE
climate-card   1/1     1            1           6s
david@davidui-MacBookAir 기후동행 % kubectl get pods
NAME                            READY   STATUS    RESTARTS   AGE
climate-card-647cd55d89-vlzzn   1/1     Running   0          11s
```

```
david@davidui-MacBookAir 기후동행 % kubectl port-forward deploy/climate-card 8080:80
Forwarding from 127.0.0.1:8080 -> 80
Forwarding from [::1]:8080 -> 80
curl http://localhost:8080
kubectl get pods -o custom-columns=NAME:metadata.name

kubectl get pods -o custom-columns=NAME:metadata.name
^C%
david@davidui-MacBookAir 기후동행 % kubectl get pods -o custom-columns=NAME:metadata.name
NAME
climate-card-647cd55d89-vlzzn
```

배포

Deployment 상태 확인

파드 상태 확인

포트 포워딩

접속 테스트

파드 이름 확인

2.6 연습문제

동일한 파드임을 검증하기

```
david@davidui-MacBookAir 기후동행 % kubectl get pods -o custom-columns=NAME:metadata.name

NAME
climate-card-647cd55d89-vlzzn
```

파드 이름 출력

```
david@davidui-MacBookAir 기후동행 % curl http://localhost:8080
Hostname: climate-card-647cd55d89-vlzzn
IP: 127.0.0.1
IP: ::1
IP: 10.1.0.24
IP: fe80::5cb6:b5ff:fee4:2eae
RemoteAddr: 127.0.0.1:36798
GET / HTTP/1.1
Host: localhost:8080
User-Agent: curl/8.7.1
Accept: */*
```

curl 응답의 Hostname 출력

이름이 동일하므로 파드가 정상 동작하는 것을 알 수 있음.

2.7 핵심정리

개념	설명
파드(Pod)	컨테이너를 감싸는 최소 배포 단위
디플로이먼트	파드를 관리하는 컨트롤러 (자동 복구, 스케일링)
매니페스트	쿠버네티스 오브젝트 정의서 (YAML)
리소스 관리	의도한 상태 유지

- 쿠버네티스는 컨테이너가 아니라 파드를 관리
- 파드는 일시적이기 때문에 컨트롤러가 필요
- Deployment는 선언적 상태 관리의 핵심
- 삭제된 리소스도 컨트롤러(ReplicaSet)가 감지하여 즉시 복구함