



kubernetes

쿠버네티스 구성

Autumn 2024



AI융합학과

Seongbok Baik
sbbaik@dju.ac.kr



한 권으로 배우는
도커 & 쿠버네티스
컨테이너 개념부터 쿠버네티스를 활용한 배포까지
DOCKER & KUBERNETES

장철원 지음



장철원 소프트웨어공학자

01 쿠버네티스 구성 정보 확인

클러스터의 핵심 서비스 운영 상태를 확인하고 클러스터가 올바르게 작동 중인지 진단

```
eevee@myserver01:~$ kubectl cluster-info ①
Kubernetes control plane is running at https://10.0.2.4:6443
CoreDNS is running at https://10.0.2.4:6443/api/v1/namespaces/kube-system/services/
kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
```

- Kubernetes Control Plane: 클러스터의 중앙 관리 구성 요소가 실행 중
 - https://10.0.2.4:6443 주소에서 실행 중
- CoreDNS: Kubernetes의 DNS 서버, 내부 서비스 검색에 사용됨
 - CoreDNS는 https://10.0.2.4:6443/api/...에서 실행 중

02 쿠버네티스 노드 정보 확인

클러스터 노드 정보 확인

```
eevee@myserver01:~$ kubectl get nodes ❶
```

NAME	STATUS	ROLES	AGE	VERSION
myserver01	Ready	control-plane	3d18h	v1.26.5
myserver02	Ready	<none>	3d17h	v1.26.5
myserver03	Ready	<none>	3d17h	v1.26.5

- 3 노드 (myserver01, 02, 03)가 ready 상태임을 확인

03 쿠버네티스 파드 정보 확인

클러스터 파드 정보 확인

```
eevee@myserver01:~$ kubectl get pod
```

①

NAME	READY	STATUS	RESTARTS	AGE
hello-world	0/1	Completed	0	8s

```
eevee@myserver01:~$ kubectl get pod -o wide
```

②

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE
hello-world	0/1	Completed	0	3d17h	192.168.131.2	myserver02
<none>	<none>					

- hello-world 파드가 생성되어 있음을 확인

04 쿠버네티스 파드 삭제

클러스터 파드 삭제

```
eevee@myserver01:~$ kubectl delete pod hello-world
```

①

```
pod "hello-world" deleted
```

```
eevee@myserver01:~$ kubectl get pod
```

②

```
No resources found in default namespace.
```

- hello-world 파드 삭제 및 삭제 확인

05 쿠버네티스 파드 실행 확인

```
eevee@myserver01:~$ kubectl run hello-world --image=hello-world --restart=Always ❶
pod/hello-world created
```

```
eevee@myserver01:~$ kubectl get all ❷
```

NAME	READY	STATUS	RESTARTS	AGE
pod/hello-world	0/1	CrashLoopBackOff	1 (12s ago)	18s

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	3d18h

- `kubectl run hello-world --image=hello-world --restart=Always`
 - hello-world라는 이름의 Pod를 생성
 - `--image=hello-world`는 Docker 이미지 이름
 - `--restart=Always`는 Pod를 항상 재시작하도록 설정
 - STATUS: CrashLoopBackOff: Pod가 계속해서 충돌(Crash)/재시작을 반복하는 문제
 - RESTARTS: Pod가 12초 전에 1번 재시작되었음
- `kubectl get all`: 클러스터 내 모든 리소스를 조회

06 파드 생성시 calico 에러 해결 방법

```
eevee@myserver01:~/work/ch09/ex14$ kubectl describe pod/hello-world
Failed to create pod sandbox: rpc error: code = Unknown desc = failed to setup
network for sandbox "56e9be09cf4062ca5b0177ec3ad0f04299875909dcb97315d9919f7a4
21fa497": plugin type="calico" failed (add): error getting ClusterInformation:
connection is unauthorized: Unauthorized
```

- calico 에러 이므로 재 실행 시도

07 파드 생성시 calico 에러 해결 방법

```
eevee@myserver01:~/work/ch09/ex14$ kubectl get pod --namespace calico-system ①
```

NAME	READY	STATUS	RESTARTS	AGE
calico-kube-controllers-8587d5cdf7-vl6fl	1/1	Running	1 (32h ago)	2d11h
calico-node-g26hp	1/1	Running	1 (32h ago)	2d11h ②
calico-node-mjhmt	1/1	Running	2 (39m ago)	2d11h ③
calico-node-z8zkx	1/1	Running	1 (16m ago)	2d11h ④
calico-typha-575b7c4bb6-77qzn	1/1	Running	4 (38m ago)	2d11h
calico-typha-575b7c4bb6-kgknx	1/1	Running	1 (16m ago)	2d11h
csi-node-driver-2tbkt	2/2	Running	2 (32h ago)	2d11h
csi-node-driver-fhgf9	2/2	Running	2 (16m ago)	2d11h
csi-node-driver-ssstk	2/2	Running	4 (39m ago)	2d11h

- calico-node-xxxx 세개 삭제 -> calico-node는 자동으로 생성

```
$ kubectl delete pod calico-node-g26hp --namespace calico-system
```

08 파드 강제 삭제

```
eevee@myserver01:~$ kubectl get all
```

①

NAME	READY	STATUS	RESTARTS	AGE
pod/hello-world	0/1	Terminating	0	3m44s

②

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	3d18h

- Terminating 상태로 중지되는 경우 있음

09 파드 강제 삭제

```
eevee@myserver01:~$ kubectl delete pod hello-world --grace-period=0 --force ①
```

Warning: Immediate deletion does not wait for confirmation that the running resource has been terminated. The resource may continue to run on the cluster indefinitely.

```
pod "hello-world" force deleted
```

```
eevee@myserver01:~$ kubectl get all ②
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	3d18h

- --force 옵션: 강제 삭제

10 매니페스트를 활용한 파드 실행

- 매니페스트: Kubernetes 오브젝트를 생성하기 위해 작성되는 메타정보 파일
- 주로 YAML 또는 JSON 형식으로 작성
- 매니페스트 파일에 실행할 파드(Pod)나 서비스(Service) 등의 정보를 입력

매니페스트 파일 저장을 위한 디렉토리 생성

```
eevee@myserver01:~$ cd work/ ①
eevee@myserver01:~/work$ ls ②
ch04  ch05  ch06
eevee@myserver01:~/work$ mkdir ch09 ③
eevee@myserver01:~/work$ ls ④
ch04  ch05  ch06  ch09
eevee@myserver01:~/work$ cd ch09 ⑤
eevee@myserver01:~/work/ch09$ mkdir ex01 ⑥
eevee@myserver01:~/work/ch09$ ls ⑦
ex01
eevee@myserver01:~/work/ch09$ cd ex01 ⑧
eevee@myserver01:~/work/ch09/ex01$
```

11 매니페스트를 활용한 파드 실행

```
eevee@myserver01:~/work/ch09/ex01$ vim nginx-test01.yml
apiVersion: v1
kind: Pod
metadata:
  name: nginx01
spec:
  containers:
  - name: nginx-test01
    image: nginx:latest
```

1
2
3
4
5
6
7
8
9

- nginx-test01.yml 이라는 매니페스트 파일 생성
- apiVersion: v1 -> Kubernetes API의 버전을 정의
- kind: Pod -> 생성하려는 Kubernetes 오브젝트의 종류
- metadata: -> 리소스에 대한 메타데이터 정보
- spec: -> Pod의 스펙(구체적인 설정)을 정의

12 매니페스트를 활용한 파드 실행

```
eevee@myserver01:~/work/ch09/ex01$ kubectl apply -f nginx-test01.yml ①  
pod/nginx01 created ②  
  
eevee@myserver01:~/work/ch09/ex01$ kubectl get pod ③  
NAME      READY   STATUS    RESTARTS   AGE  
nginx01   1/1     Running   0           33s ④
```

- nginx-test01.yml이라는 매니페스트 파일을 사용해 Kubernetes 클러스터에 Pod를 생성
- pod/nginx01 created nginx01이라는 이름의 Pod가 성공적으로 생성

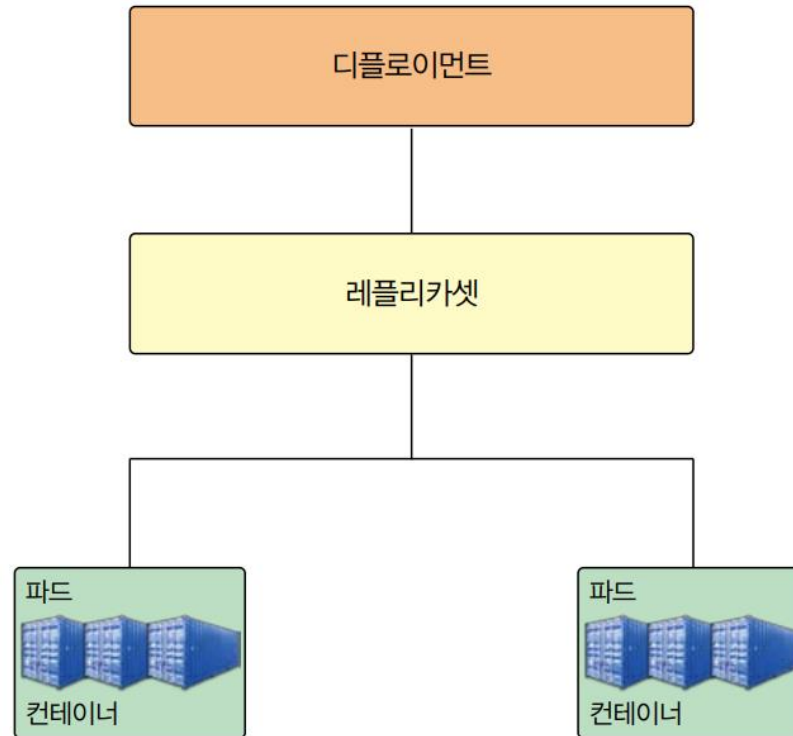
13 매니페스트를 활용한 파드 삭제

```
eevee@myserver01:~/work/ch09/ex01$ kubectl delete -f nginx-test01.yml ①  
pod "nginx01" deleted ②  
  
eevee@myserver01:~/work/ch09/ex01$ kubectl get pod ③  
No resources found in default namespace. ④
```

- nginx-test01.yml 매니페스트 파일을 사용해 Kubernetes 클러스터에 생성했던 Pod 삭제

14 디플로이먼트

디플로이먼트 개념



- 디플로이먼트는 레플리카셋을 관리함
- 레플리카셋은 여러 개의 Pod를 관리하며, 이를 통해 애플리케이션의 복제본을 유지
- 각 Pod는 내부적으로 하나 이상의 컨테이너를 실행

15 디플로이먼트 개념

- 디플로이먼트(Deployment):
 - Kubernetes에서 애플리케이션 배포를 관리하는 가장 상위 레벨의 객체
 - 선언적 방식으로 애플리케이션의 상태(예: 원하는 Pod 개수)를 정의하고, 이를 유지관리
 - 디플로이먼트는 레플리카셋을 생성하고 관리합니다.
- 레플리카셋(ReplicaSet):
 - 지정된 수의 Pod 복제본(레플리카)을 항상 유지하는 역할
 - 디플로이먼트가 레플리카셋을 관리하며, 필요에 따라 새로운 Pod를 생성/삭제
 - Pod의 개수를 조정(스케일링)하거나 실패한 Pod를 다시 생성
- 파드(Pod):
 - Kubernetes에서 가장 작은 배포 단위
 - 각 Pod는 하나 이상의 컨테이너를 포함
 - 동일한 네트워크 네임스페이스(IP 주소)를 공유하며, 보통 하나의 컨테이너와 함께 실행
- 컨테이너(Container):
 - Pod 내부에서 실제로 실행되는 애플리케이션
 - Docker와 같은 컨테이너 런타임을 사용해 애플리케이션을 캡슐화하고 실행

16 디플로이먼트 실행

```
eevee@myserver01:~$ kubectl create deployment deploy-hello --image=hello-world ❶
deployment.apps/deploy-hello created
```

```
eevee@myserver01:~$ kubectl get all ❷
```

NAME	READY	STATUS	RESTARTS	AGE
pod/deploy-hello-54df7ff57c-qgncl	0/1	CrashLoopBackOff	4 (89s ago)	3m4s

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	17d

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/deploy-hello	0/1	1	0	3m4s ❸

NAME	DESIRED	CURRENT	READY	AGE
replicaset.apps/deploy-hello-54df7ff57c	1	1	0	3m4s

- 디플로이먼트는 레플리카셋을 관리함
- 레플리카셋은 여러 개의 Pod를 관리하며, 이를 통해 애플리케이션의 복제본을 유지
- 각 Pod는 내부적으로 하나 이상의 컨테이너를 실행

17 디플로이먼트 실행 결과

Pod (pod/deploy-hello-54df7ff57c-qgncl)

- Pod는 Kubernetes에서 실행 가능한 최소 배포 단위
- 하나 이상의 컨테이너를 포함하며, 동일한 네트워크 네임스페이스를 공유
- Pod는 컨테이너를 실행하고 관리하며, 디플로이먼트나 레플리카셋이 이를 제어

메시지 분석

- "pod/deploy-hello-54df7ff57c-qgncl": 디플로이먼트가 생성한 Pod의 이름
- 이름의 구조:
 - deploy-hello(디플로이먼트 이름)
 - 54df7ff57c(레플리카셋 식별자)
 - qgncl(Pod 식별자)

18 디플로이먼트 실행 결과

Deployment (deployment/deploy-hello)

- Deployment는 Kubernetes에서 애플리케이션의 선언적 배포를 관리하는 상위 레벨 리소스
- 사용자가 애플리케이션의 원하는 상태(Desired State)를 정의하면, Deployment가 이를 유지 관리

특징:

- 디플로이먼트는 내부적으로 레플리카셋(ReplicaSet)을 생성하고 관리
- Pod 수를 조정(스케일링)하거나, 새로운 버전으로 애플리케이션을 업그레이드

역할:

- 사용자가 정의한 애플리케이션 상태를 유지하고, 이를 레플리카셋을 통해 구현
- Pod가 충돌하거나 삭제되면, 디플로이먼트가 이를 복구

19 디플로이먼트 실행 결과

ReplicaSet (replicaset.apps/deploy-hello-54df7ff57c)

- ReplicaSet은 Kubernetes에서 지정된 수의 Pod를 항상 유지하는 역할
- 디플로이먼트가 생성 및 관리하는 하위 리소스

특징

- 레플리카셋은 Pod의 복제본(레플리카)을 관리
- DESIRED/CURRENT/READY 필드를 통해 현재 실행 중인 Pod의 수를 모니터링
- replicaset.apps/deploy-hello-54df7ff57c는 디플로이먼트가 생성한 레플리카셋의 이름

역할

- 디플로이먼트가 원하는 Pod의 수를 유지하며, Pod의 생성, 삭제 및 상태를 관리
- Pod를 생성하고, 충돌 시 새로운 Pod를 생성

20 각 리소스 간 관계

Deployment

- 가장 상위 레벨의 관리 리소스
- 사용자가 디플로이먼트를 생성하면 Kubernetes가 아래 리소스를 자동으로 생성하고 관리

ReplicaSet

- 디플로이먼트가 원하는 Pod의 복제본을 유지하기 위해 레플리카셋을 생성
- Pod의 수를 관리하며, 디플로이먼트의 중간 관리자 역할

Pod

- 레플리카셋이 생성하고 관리하는 실제 실행 단위
- 컨테이너를 포함하며, 애플리케이션을 실제로 실행

21 Deployment 요약

- Pod: 애플리케이션이 실행되는 실제 단위.
- ReplicaSet: Pod의 복제본 수를 관리.
- Deployment: 사용자가 정의한 애플리케이션 상태를 유지하고 관리

[Deployment > ReplicaSet > Pod] 구조를 통해 Kubernetes는 애플리케이션의 상태를 선언적으로 유지

22 자원 별 정보 확인

```
eevee@myserver01:~$ kubectl get pod ①
```

NAME	READY	STATUS	RESTARTS	AGE
deploy-hello-54df7ff57c-qgncl	0/1	CrashLoopBackOff	3 (43s ago)	92s


```
eevee@myserver01:~$ kubectl get replicaset ②
```

NAME	DESIRED	CURRENT	READY	AGE
deploy-hello-54df7ff57c	1	1	0	15s


```
eevee@myserver01:~$ kubectl get deployment ③
```

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deploy-hello	0/1	1	0	2m41s

- kubectl get all 명령 대신에 각 세부 자원별 정보 확인
 - kubectl get pod / replicaset / deployment

23 자원 별 정보 확인

```
eevee@myserver01:~$ kubectl get deployment,replicaset,pod
```

1

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/deploy-hello	0/1	1	0	4m22s

NAME	DESIRED	CURRENT	READY	AGE
replicaset.apps/deploy-hello-54df7ff57c	1	1	0	4m22s

NAME	READY	STATUS	RESTARTS	AGE
pod/deploy-hello-54df7ff57c-qgncl	0/1	CrashLoopBackOff	5 (75s ago)	4m22s

- 여러 자원 한꺼번에 정보 확인

24 자원 별 정보 확인

```
eevee@myserver01:~$ kubectl get deploy,rs,po
```

2

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/deploy-hello	0/1	1	0	5m25s

NAME	DESIRED	CURRENT	READY	AGE
replicaset.apps/deploy-hello-54df7ff57c	1	1	0	5m25s

NAME	READY	STATUS	RESTARTS	AGE
pod/deploy-hello-54df7ff57c-qgnc1	0/1	CrashLoopBackOff	5 (2m18s ago)	5m25s

- deployment 줄임말 -> deploy
- replicaset -> rs
- pod -> po

25 리플리카셋 조정

```
eevee@myserver01:~$ kubectl create deployment deploy-nginx --image=nginx --replicas=3 ①  
deployment.apps/deploy-nginx created
```

- (1) 디플로이먼트 생성, replicas 옵션을 활용하여 파드 개수를 설정
- 리플리카셋은 원하는 파드만큼 유지시켜주는 역할을 하는 컨트롤러
- 리플리카셋을 통해 파드 개수 조정
- 파드 개수 조절은 리플리카셋이 아니라 디플로이먼트를 통해 조절

26 디플로이먼트 자원 확인

```
eevee@myserver01:~$ kubectl get deploy,rs,po
```

2

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/deploy-nginx	3/3	3	3	21s

3

NAME	DESIRED	CURRENT	READY	AGE
replicaset.apps/deploy-nginx-7d74c85c6f	3	3	3	21s

4

NAME	READY	STATUS	RESTARTS	AGE
pod/deploy-nginx-7d74c85c6f-967d7	1/1	Running	0	21s
pod/deploy-nginx-7d74c85c6f-c5kvn	1/1	Running	0	21s
pod/deploy-nginx-7d74c85c6f-qzvlx	1/1	Running	0	21s

5

- (2) 디플로이먼트, 레플리카셋, 파드 정보를 확인
- (3) 디플로이먼트 정보 확인
- (4) 레플리카셋 정보를 확인하며 파드 세 개가 실행되는 상황 확인
- (5) 각 파드 정보 확인

27 디플로이먼트 자원 확인

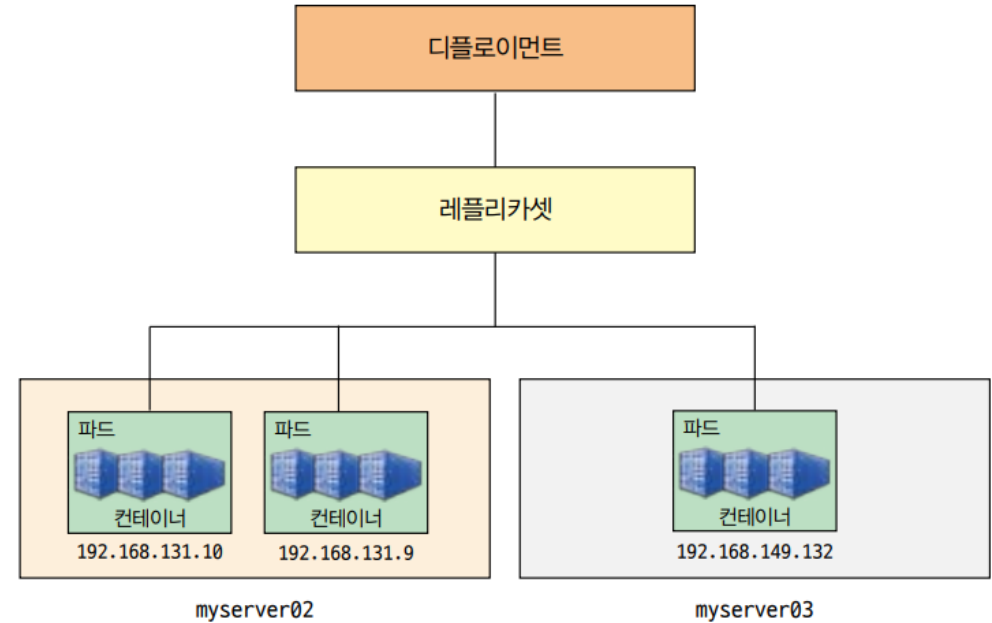
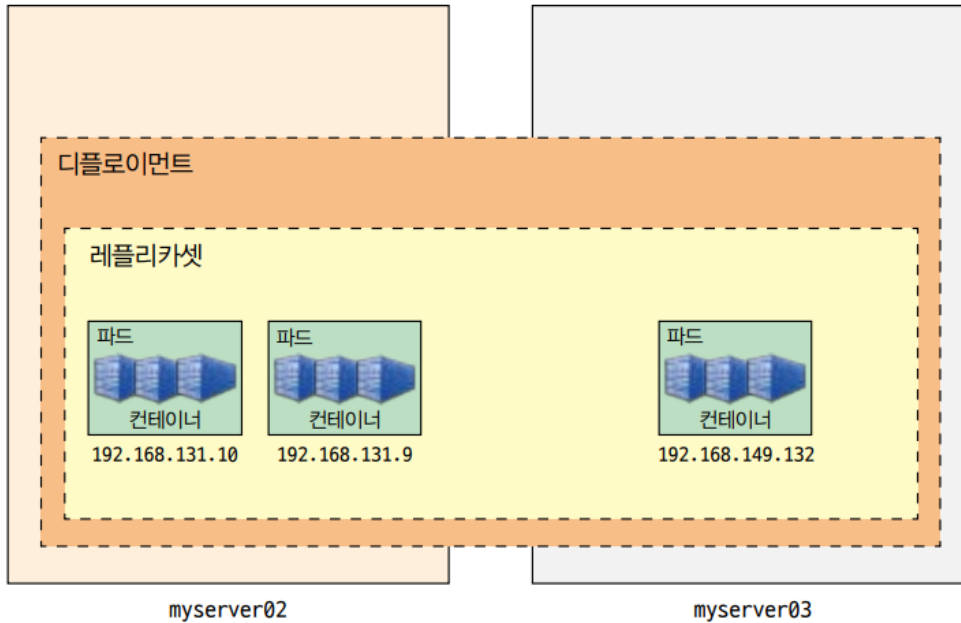
```
eevee@myserver01:~$ kubectl get deploy,rs,po -o wide
```

NAME	READY	UP-TO-DATE	AVAILABLE	AGE	CONTAINERS
deployment.apps/deploy-nginx	3/3	3	3	2m32s	nginx
nginx	app=deploy-nginx				

NAME	DESIRED	CURRENT	READY	AGE
replicaset.apps/deploy-nginx-7d74c85c6f	3	3	3	2m32s
nginx	app=deploy-nginx,pod-template-hash=7d74c85c6f			

NAME	READY	STATUS	RESTARTS	AGE	IP
pod/deploy-nginx-7d74c85c6f-967d7	1/1	Running	0	2m32s	
192.168.149.132	myserver03	<none>	<none>		
pod/deploy-nginx-7d74c85c6f-c5kvn	1/1	Running	0	2m32s	
192.168.131.10	myserver02	<none>	<none>		
pod/deploy-nginx-7d74c85c6f-qzvlx	1/1	Running	0	2m32s	192.168.131.9
myserver02	<none>	<none>			

28 디플로이먼트 레플리카셋 개념



- 파드들은 myserver02와 myserver03에 각각 흩어져 설치
- 하나의 리플리카셋, 하나의 디플로이먼트에 속함

29 디플로이먼트 구성 파드 삭제

```
eevee@myserver01:~$ kubectl delete pod deploy-nginx-7d74c85c6f-qzvlx
pod "deploy-nginx-7d74c85c6f-qzvlx" deleted
```

①

```
eevee@myserver01:~$ kubectl get deploy,rs,po
```

②

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/deploy-nginx	3/3	3	3	13m

NAME	DESIRED	CURRENT	READY	AGE
replicaset.apps/deploy-nginx-7d74c85c6f	3	3	3	13m

NAME	READY	STATUS	RESTARTS	AGE
pod/deploy-nginx-7d74c85c6f-967d7	1/1	Running	0	13m
pod/deploy-nginx-7d74c85c6f-c5kvn	1/1	Running	0	13m
pod/deploy-nginx-7d74c85c6f-wxnwr	1/1	ContainerCreating	0	4s

③

- (1) 앞서 확인한 파드 이름을 토대로 해당 파드 삭제
- (3) 정보 확인을 통해 해당 파드가 삭제되고 다른 파드를 생성

디플로이먼트를 생성할 때 리플리카셋을 3으로 설정 -> 파드 3개로 유지

30 디플로이먼트 구성 파드 삭제 복구

```
eevee@myserver01:~$ kubectl get deploy,rs,po
```

①

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/deploy-nginx	3/3	3	3	13m

NAME	DESIRED	CURRENT	READY	AGE
replicaset.apps/deploy-nginx-7d74c85c6f	3	3	3	13m

NAME	READY	STATUS	RESTARTS	AGE
pod/deploy-nginx-7d74c85c6f-967d7	1/1	Running	0	13m
pod/deploy-nginx-7d74c85c6f-c5kvn	1/1	Running	0	13m
pod/deploy-nginx-7d74c85c6f-wxnwr	1/1	Running	0	15s

②

- 수 초 후 다시 디플로이먼트, 레플리카셋, 파드 정보 확인 -> 해당 파드 재 생성

서로에게, 자신에게 친절합니다 - 허준이

