

## Contents

- ReplicaSet을 이용해 Pod 관리하기
- Deployment를 이용해 Pod 관리하기

# ReplicaSet

이제 Pod를 관리하는 다른 방법을 알아보겠습니다.

첫 번째는 ReplicaSet입니다. ReplicaSet 생성을 위해서 아래 파일을 작성합니다.

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: nginx-replicaset
  labels:
    app: my-nginx
    tier: frontend
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-nginx
  template:
    metadata:
      labels:
        app: my-nginx
        name: my-nginx
    spec:
      containers:
        - image: nginx:1.19.3
          name: my-nginx
          ports:
            - containerPort: 80
```

파일명은 nginx-replicaset.yaml로 합니다.

## Docker & Kubernetes - [Hands-on] 09. Kubernetes Workload(2)

spec부분을 보시면, 우리가 원하는 Pod에 대한 spec이 보이고, 그 위에 `replicas: 3` 이라는 부분이 보이네요. 이 부분이 핵심입니다.

나는 Nginx Pod를 세 개 원한다고 선언한 것입니다.

특별한 얘기가 없으면 yaml파일을 이용한 리소스 생성은 `kubectl apply` 명령어를 쓰시면 됩니다.

ReplicaSet을 생성해볼까요?

```
ubuntu@ip-10-0-1-14:~$ kubectl apply -f nginx-replicaset.yaml
replicaset.apps/nginx-replicaset created
```

명령어 : `kubectl apply -f nginx-replicaset.yaml`

조회도 해보시구요.

```
ubuntu@ip-10-0-1-14:~$ kubectl get replicaset -o wide
NAME          DESIRED   CURRENT   READY   AGE      CONTAINERS   IMAGES
nginx-replicaset   3         3         3    105s   my-nginx   nginx:1.19.3   app=my-nginx
```

명령어 : `kubectl get replicaset -o wide`

## Docker & Kubernetes - [Hands-on] 09. Kubernetes Workload(2)

상세조회 결과는 아래와 같습니다.

```
ubuntu@ip-10-0-1-14:~$ kubectl describe replicaset nginx-replicaset
Name:           nginx-replicaset
Namespace:      default
Selector:       app=my-nginx
Labels:         app=my-nginx
                tier=frontend
Annotations:    <none>
Replicas:       3 current / 3 desired
Pods Status:   3 Running / 0 Waiting / 0 Succeeded / 0 Failed
Pod Template:
  Labels:  app=my-nginx
  Containers:
    my-nginx:
      Image:      nginx:1.19.3
      Port:       80/TCP
      Host Port:  0/TCP
      Environment: <none>
      Mounts:     <none>
      Volumes:    <none>
  Events:
    Type      Reason          Age    From            Message
    ----      -----          ----   ----
    Normal    SuccessfulCreate 2m47s  replicaset-controller  Created pod: nginx-replicaset-4ljhf
    Normal    SuccessfulCreate 2m47s  replicaset-controller  Created pod: nginx-replicaset-xkltg
    Normal    SuccessfulCreate 2m47s  replicaset-controller  Created pod: nginx-replicaset-89tl8
```

명령어 : `kubectl describe replicaset nginx-replicaset`

## Docker & Kubernetes - [Hands-on] 09. Kubernetes Workload(2)

우리는 Pod를 생성하지는 않았지만 Pod도 생성됐습니다.  
ReplicaSet이 하는 일이 그런거니까요.

Pod도 조회해볼까요?

```
ubuntu@ip-10-0-1-14:~$ kubectl get pods --show-labels
NAME           READY   STATUS    RESTARTS   AGE   LABELS
nginx-replicaset-4ljhf  1/1     Running   0          4m34s  app=my-nginx
nginx-replicaset-89tl8  1/1     Running   0          4m34s  app=my-nginx
nginx-replicaset-xkltg  1/1     Running   0          4m34s  app=my-nginx
```

명령어 : `kubectl get pods --show-labels`

이제 뭔가 좀 자동으로 돌아가는 모양새가 나오네요~

이제 생성한 리소스들을 삭제해볼게요.

`apply` 를 `delete` 로 바꿔주시면 됩니다. (ง •̀ ∇ •́)ง

```
ubuntu@ip-10-0-1-14:~$ kubectl delete -f nginx-replicaset.yaml
replicaset.apps "nginx-replicaset" deleted
```

명령어 : `kubectl delete -f nginx-replicaset.yaml`

# Deployment

좀 더 나가서, 이번엔 Deployment 입니다. 먼저 YAML파일을 만들어보겠습니다.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: my-nginx
    tier: frontend
spec:
  replicas: 3
  strategy:
    type: RollingUpdate
  selector:
    matchLabels:
      app: my-nginx
  template:
    metadata:
      labels:
        app: my-nginx
        name: my-nginx
    spec:
      containers:
        - image: nginx:1.19.3
          name: my-nginx
          ports:
            - containerPort: 80
```

파일명은 nginx-deployment.yaml로 합니다.

## Docker & Kubernetes - [Hands-on] 09. Kubernetes Workload(2)

일단 한번 생성해볼게요.

```
ubuntu@ip-10-0-1-14:~$ kubectl apply -f nginx-deployment.yaml  
deployment.apps/nginx-deployment created
```

명령어 : `kubectl apply -f nginx-deployment.yaml`

이번엔 새로운 명령어 `kubectl get all`을 해볼까요?

```
ubuntu@ip-10-0-1-14:~$ kubectl get all  
NAME                                     READY   STATUS    RESTARTS   AGE  
pod/nginx-deployment-596ff98864-7rcc5   1/1     Running   0          51s  
pod/nginx-deployment-596ff98864-8gzpg   1/1     Running   0          51s  
pod/nginx-deployment-596ff98864-ccgxd   1/1     Running   0          51s  
  
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE  
service/kubernetes   ClusterIP  10.96.0.1    <none>        443/TCP   5d11h  
  
NAME                                     READY   UP-TO-DATE   AVAILABLE   AGE  
deployment.apps/nginx-deployment   3/3     3           3           51s  
  
NAME                                     DESIRED   CURRENT   READY   AGE  
replicaset.apps/nginx-deployment-596ff98864  3         3         3         51s
```

명령어 : `kubectl get all`

오~ 다 나오네요... ㅋ('ㅁ'\*),

## Docker & Kubernetes - [Hands-on] 09. Kubernetes Workload(2)

Pod들을 Label까지 같이 보려면 아래와 같이 하면 됩니다.

```
ubuntu@ip-10-0-1-14:~$ kubectl get po --show-labels
NAME           READY   STATUS    RESTARTS   AGE   LABELS
nginx-deployment-596ff98864-7rcc5  1/1     Running   0          113s  app=my-nginx, pod-template-hash=596ff98864
nginx-deployment-596ff98864-8gzpg  1/1     Running   0          113s  app=my-nginx, pod-template-hash=596ff98864
nginx-deployment-596ff98864-ccgxd  1/1     Running   0          113s  app=my-nginx, pod-template-hash=596ff98864
```

명령어 : `kubectl get po --show-labels`

이제 Pod 하나를 삭제(delete)해 볼까요?

```
ubuntu@ip-10-0-1-14:~$ kubectl delete po nginx-deployment-596ff98864-7rcc5
pod "nginx-deployment-596ff98864-7rcc5" deleted
```

명령어 : `kubectl delete po [POD-NAME]`

[POD-NAME]에는 앞에서 조회된 POD 중 하나의 이름을 넣어주세요.

그리고, 다시 조회를 해보면...

```
ubuntu@ip-10-0-1-14:~$ kubectl get po --show-labels
NAME           READY   STATUS    RESTARTS   AGE   LABELS
nginx-deployment-596ff98864-8gzpg  1/1     Running   0          3m9s  app=my-nginx, pod-template-hash=596ff98864
nginx-deployment-596ff98864-ccgxd  1/1     Running   0          3m9s  app=my-nginx, pod-template-hash=596ff98864
nginx-deployment-596ff98864-8sqsn  1/1     Running   0          11s   app=my-nginx, pod-template-hash=596ff98864
```

명령어 : `kubectl get po --show-labels`

새롭게 하나의 POD가 생성된 걸 볼 수 있습니다. ReplicaSet이 자기 역할을 다하고 있는 듯 하네요~ 이제 든든합니다.

## Docker & Kubernetes - [Hands-on] 09. Kubernetes Workload(2)

이번엔 scale in/out 방법을 알아보겠습니다. (replicas를 조정)

명령형 커맨드 방식으로는 이렇게 할 수 있습니다.

```
ubuntu@ip-10-0-1-14:~$ kubectl scale deployment nginx-deployment --replicas=5  
deployment.apps/nginx-deployment scaled
```

명령어 : `kubectl scale deployment nginx-deployment --replicas=5`

조회결과도 보겠습니다.

```
ubuntu@ip-10-0-1-14:~$ kubectl get all  
NAME                                     READY   STATUS    RESTARTS   AGE  
pod/nginx-deployment-596ff98864-6m6nz   1/1     Running   0          41s  
pod/nginx-deployment-596ff98864-8gzpg   1/1     Running   0          8m50s  
pod/nginx-deployment-596ff98864-8sqsn   1/1     Running   0          5m52s  
pod/nginx-deployment-596ff98864-ccgxd   1/1     Running   0          8m50s  
pod/nginx-deployment-596ff98864-sh7sh   1/1     Running   0          41s  
  
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE  
service/kubernetes   ClusterIP  10.96.0.1   <none>        443/TCP   5d11h  
  
NAME                                     READY   UP-TO-DATE   AVAILABLE   AGE  
deployment.apps/nginx-deployment   5/5     5           5           8m50s  
  
NAME                                     DESIRED   CURRENT   READY   AGE  
replicaset.apps/nginx-deployment-596ff98864   5         5         5         8m50s
```

명령어 : `kubectl get all`

명령형 커맨드에서 지정한 대로 Pod의 개수가 다섯개가 되었습니다. 새롭게 생성된 두 개의 Pod를 볼 수 있습니다.

## Docker & Kubernetes - [Hands-on] 09. Kubernetes Workload(2)

`kubectl edit deployment nginx-deployment` 명령어로 생성된 리소스를 수정할 수도 있습니다.

마치 vi editor를 이용하여 YAML파일을 수정하는 것과 동일합니다.

한 번 해보세요. (vi가 익숙하지 않은 경우, 이 실습은 생략해도 됩니다.)

명령어 : `kubectl edit deployment nginx-deployment`

`replicas` 를 2로 바꾸고 저장후 빠져나옵니다. ( `:wq` )

조회결과는 아래와 같습니다.

```
ubuntu@ip-10-0-1-14:~/mspt2/hands_on_files$ kubectl get all
NAME                           READY   STATUS    RESTARTS   AGE
pod/nginx-deployment-596ff98864-8gzpg   1/1     Running   0          16m
pod/nginx-deployment-596ff98864-8sqsn   1/1     Running   0          13m

NAME           TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.96.0.1   <none>        443/TCP   5d11h

NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/nginx-deployment   2/2     2           2           16m

NAME                           DESIRED   CURRENT   READY   AGE
replicaset.apps/nginx-deployment-596ff98864   2         2         2       16m
```

명령어 : `kubectl get all`

## Docker & Kubernetes - [Hands-on] 09. Kubernetes Workload(2)

그리고, 마지막으로 선언형 방법을 적용하려면 처음 사용된 yaml파일을 수정해주시면 됩니다.  
editor를 이용하여 `.spec.replicas` 부분을 수정하면 됩니다. (4로 변경)

그리고, 마법의 주문 `kubectl apply` 를 하는거죠.

```
ubuntu@ip-10-0-1-14:~$ kubectl apply -f nginx-deployment.yaml  
deployment.apps/nginx-deployment configured
```

명령어 : `kubectl apply -f nginx-deployment.yaml`

조회결과는 아래와 같습니다.

```
ubuntu@ip-10-0-1-14:~$ kubectl get all  
NAME                                         READY   STATUS    RESTARTS   AGE  
pod/nginx-deployment-596ff98864-74x29      1/1     Running   0          63s  
pod/nginx-deployment-596ff98864-8gzpg       1/1     Running   0          21m  
pod/nginx-deployment-596ff98864-8sqsn       1/1     Running   0          18m  
pod/nginx-deployment-596ff98864-tsqfk        1/1     Running   0          63s  
  
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE  
service/kubernetes   ClusterIP  10.96.0.1    <none>        443/TCP   5d11h  
  
NAME                                         READY   UP-TO-DATE   AVAILABLE   AGE  
deployment.apps/nginx-deployment   4/4     4           4           21m  
  
NAME                                         DESIRED  CURRENT   READY   AGE  
replicaset.apps/nginx-deployment-596ff98864  4        4         4         21m
```

명령어 : `kubectl get all`

## Docker & Kubernetes - [Hands-on] 09. Kubernetes Workload(2)

마지막으로 생성한 리소스들을 삭제하고 마치겠습니다. ^\_^

```
ubuntu@ip-10-0-1-14:~$ kubectl delete -f nginx-deployment.yaml
deployment.apps "my-nginx-deployment" deleted
```

명령어 : `kubectl delete -f nginx-deployment.yaml`