

# (23.01.10) Deployment

## Deployment

### Deployment는 왜 필요할까?



운영환경에서 웹서버를 배포해야하는 여러가지 케이스를 생각해봅시다.

1. 최신 버전의 어플리케이션 빌드가 도커 레지스트리에서 사용 가능해질 때 마다 도커 인스턴스가 매끄럽게 업그레이드 되길 바랄 것 입니다. 하지만 어플리케이션에 액세스 하는 사용자에게 영향을 미칠 수 있으므로 한 번에 모든 인스턴스를 업그레이드 하고 싶지는 않고 하나씩 업그레이드 하고 싶습니다. (이걸 롤링업데이트라고 합니다.)
2. 수행한 업그레이드 중 하나에서 예기치 않은 오류가 발생해서 최근 변경 사항을 롤백하기를 원합니다.
3. 환경에 여러가지 변경사항을 반영하기를 원합니다. 하지만 각 변경 커맨드가 수행된 즉시 적용되기를 원하지는 않고, 모든 변경사항이 함께 돌아오도록 잠시 중단했다가 시작할 수 있도록 하고 싶습니다.



이 모든것이 deployment 를 통해 가능해집니다.

### Deployment란?

Deployment는 ReplicaSet을 소유하고 관리하는 object입니다. (쿠버네티스 계층구조에서 가장 높은곳에 위치한 object라 할 수 있겠습니다.) Deployment는 롤링업데이트를 사용해서 (계층 구조 하단의) instance들을 원활하게 업그레이드 하고, 업그레이드를 취소하고, 업그레이드를 중단했다 계속할수 있게합니다.

앞으로 우리는 ReplicaSet을 따로 생성하지 않습니다. ReplicaSet을 사용하고 싶으면 Deployment를 생성해서 Deployment를 통해 관리하세요. 이것이 쿠버네티스의 권고사항입니다.

# YAML로 Deployment 생성하기

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3
  template:
    metadata:
      name: nginx-pod
      labels:
        app: nginx-demo
    spec:
      containers:
        - name: nginx
          image: nginx
  selector:
    matchLabels:
      app: nginx-demo
```

```
minikube * 21:23:51
kubectrl create -f deployment-demo.yml
deployment.apps/nginx-deployment created

minikube * 21:23:55
kubectrl get all
```

NAME	READY	STATUS	RESTARTS	AGE
pod/nginx-deployment-6d78d4997c-6djts	0/1	ContainerCreating	0	3s
pod/nginx-deployment-6d78d4997c-7gz4d	0/1	ContainerCreating	0	3s
pod/nginx-deployment-6d78d4997c-khn95	0/1	ContainerCreating	0	3s

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	9d

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/nginx-deployment	0/3	3	0	3s

NAME	DESIRED	CURRENT	READY	AGE
replicaset.apps/nginx-deployment-6d78d4997c	3	3	0	3s

Deployment를 생성해보면 ReplicaSet이 자동으로 생성되는 것을 볼 수 있습니다

## Rollout 과 Versioning

우리가 처음 deployment를 생성하면 rollout이 트리거 됩니다. 새로운 Rollout은 새로운 deployment revision을 생성합니다. 이것 revision1이라고 하겠습니다. 어플리케이션이 업그레이드 되어서 새로운 버전의 컨테이너가 업데이트되면 새로운 rollout이 트리거 되고 Revision2가 생성이 됩니다.

이를 통해 우리는 배포 변경사항을 추적하고, 이전 버전으로 롤백할 수 있게 됩니다.



즉, Deployment = ReplicaSet + Pod + **history** 라고 할 수 있습니다.

배포 전략에는 크가지 두 가지 전략이 있습니다. 여러개의 replica를 배포하는 상황을 가정해 봅시다.

### 1. Recreate 전략

모든 구버전 팟을 죽이고 새버전 팟을 띄우는 방법입니다. 이것에는 문제점이 있는데, 바로 다운타임이 있다는 것입니다. 이걸 Recreate 전략이라고 부르는데, 다행히도 이것은 deployment의 기본전략이 아닙니다.

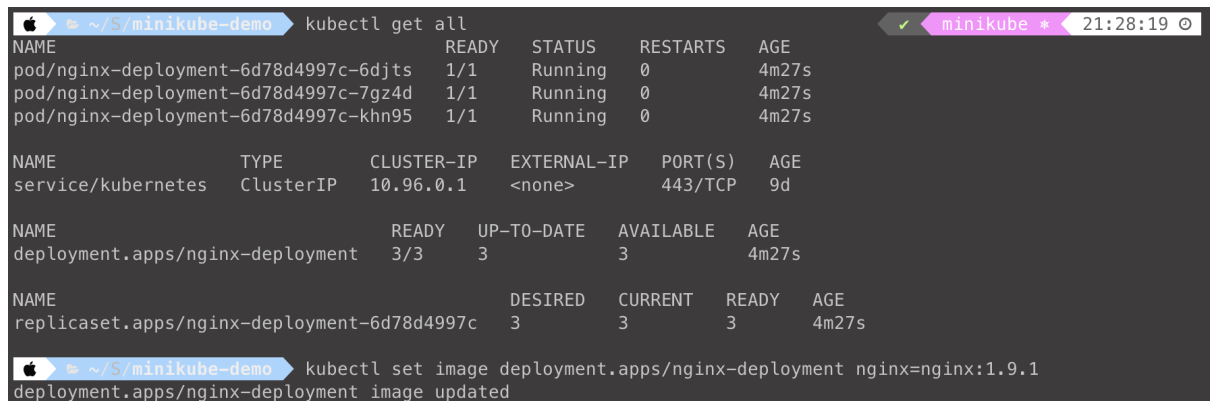
### 2. 롤링 업데이트 전략

하나를 죽이고, 하나를 살리고 - 이것을 반복하는 방법입니다. 이렇게 하면 다운타임이 없습니다. 이걸 롤링 업데이트 전략이라고 부릅니다. 만약 deployment를 생성할때에 전략을 명시해주지 않았다면 이것이 기본이 됩니다.

그러면 nginx 이미지 버전을 바꿈으로서 rollout을 발생시켜 봅시다.

```
# 방법(1)
# yaml의 이미지 버전을 수정하고
kubectl apply -f demo.yaml # 을 수행하는 방법

# 방법(2)
kubectl set image deployment/demo.yaml nginx=nginx:1.9.1
```



```
Apple ~ /S/minikube-demo kubectl get all
NAME                                READY   STATUS    RESTARTS   AGE
pod/nginx-deployment-6d78d4997c-6djts 1/1     Running   0           4m27s
pod/nginx-deployment-6d78d4997c-7gz4d 1/1     Running   0           4m27s
pod/nginx-deployment-6d78d4997c-khn95 1/1     Running   0           4m27s

NAME                                TYPE          CLUSTER-IP   EXTERNAL-IP   PORT(S)    AGE
service/kubernetes                  ClusterIP     10.96.0.1    <none>         443/TCP    9d

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/nginx-deployment    3/3     3             3           4m27s

NAME                                DESIRED   CURRENT   READY   AGE
replicaset.apps/nginx-deployment-6d78d4997c 3         3         3       4m27s

Apple ~ /S/minikube-demo kubectl set image deployment.apps/nginx-deployment nginx=nginx:1.9.1
deployment.apps/nginx-deployment image updated
```

nginx 이미지를 1.9.1 버전으로 변경했습니다. 이제 롤링업데이트가 시작됩니다.

```

$ kubectl get all
NAME                                READY    STATUS              RESTARTS   AGE
pod/nginx-deployment-6d78d4997c-6djts 1/1      Running             0           5m10s
pod/nginx-deployment-6d78d4997c-7gz4d 1/1      Terminating        0           5m10s
pod/nginx-deployment-844595f8cd-2kmvg 0/1      ContainerCreating    0           0s
pod/nginx-deployment-844595f8cd-f6dzp 1/1      Running             0           5s
pod/nginx-deployment-844595f8cd-nn4g8 1/1      Running             0           17s

NAME                TYPE        CLUSTER-IP    EXTERNAL-IP    PORT(S)    AGE
service/kubernetes  ClusterIP   10.96.0.1     <none>         443/TCP    9d

NAME                                READY    UP-TO-DATE    AVAILABLE    AGE
deployment.apps/nginx-deployment  3/3      3              3            5m10s

NAME                                DESIRED    CURRENT    READY    AGE
replicaset.apps/nginx-deployment-6d78d4997c 1          1          1        5m10s
replicaset.apps/nginx-deployment-844595f8cd 3          3          2        17s

```

롤링업데이트가 일어나고 있는 모습입니다. 1개의 컨테이너가 새로 생성되고, 기존 컨테이너 1개를 죽이고 있는 모습을 볼 수 있습니다.

```

$ kubectl get all
NAME                                READY    STATUS              RESTARTS   AGE
pod/nginx-deployment-844595f8cd-2kmvg 1/1      Running             0           98s
pod/nginx-deployment-844595f8cd-f6dzp 1/1      Running             0           103s
pod/nginx-deployment-844595f8cd-nn4g8 1/1      Running             0           115s

NAME                TYPE        CLUSTER-IP    EXTERNAL-IP    PORT(S)    AGE
service/kubernetes  ClusterIP   10.96.0.1     <none>         443/TCP    9d

NAME                                READY    UP-TO-DATE    AVAILABLE    AGE
deployment.apps/nginx-deployment  3/3      3              3            6m48s

NAME                                DESIRED    CURRENT    READY    AGE
replicaset.apps/nginx-deployment-6d78d4997c 0          0          0        6m48s
replicaset.apps/nginx-deployment-844595f8cd 3          3          3        115s

```

롤링업데이트가 모두 완료되었습니다. 그런데 왜 처음 생성되었던 레플리카셋이 그대로 살아있는 것일까요?

새로운 레플리카셋을 만들어서 롤링업데이트를 했는데, 왜 아직 처음 생성되었던 레플리카셋이 그대로 살아있는 걸까요? 우리는 디플로이먼트를 통해 롤백을 할 수 있다고 했었습니다.

이전의 레플리카셋을 그대로 남겨두고, 만약 롤백을 수행하면 이전 레플리카셋에 찢을 생성하는 방식으로 롤백이 일어납니다. 롤백을 해보겠습니다.

```

$ kubectl set image deployment.apps/nginx-deployment nginx=nginx:1.9.0
deployment.apps/nginx-deployment image updated

$ kubectl rollout status deployment nginx-deployment
Waiting for deployment "nginx-deployment" rollout to finish: 1 old replicas are pending termination...
Waiting for deployment "nginx-deployment" rollout to finish: 1 old replicas are pending termination...
deployment "nginx-deployment" successfully rolled out

$ kubectl rollout history deployment nginx-deployment
deployment.apps/nginx-deployment
REVISION  CHANGE-CAUSE
1          <none>
2          <none>
3          <none>

$ kubectl rollout status deployment nginx-deployment
deployment "nginx-deployment" successfully rolled out

```

nginx 이미지를 1.9.1 버전으로 변경했습니다. kubectl rollout history 명령어로 총 3개의 REVISION이 생성된것을 볼 수 있습니다. 현재는 REVISION 3입니다.

REVISION 3에서 REVISION 2로 롤백 해보겠습니다. `kubectl rollout undo` 명령어를 통해 이전 버전으로 롤백할 수 있습니다.

```
Apple ~/Study/k8s ➤ kubectl rollout undo deployment/nginx-deploy
deployment.apps/nginx-deploy rolled back

Apple ~/Study/k8s ➤ kubectl rollout history deployment/nginx-deploy
deployment.apps/nginx-deploy
REVISION  CHANGE-CAUSE
1          <none>
3          <none>
4          <none>
```

새로운 REVISION 4 번호로 바뀌었습니다만, REVISION 2로 돌아간 것입니다.

```
Apple ~/Study/k8s ➤ kubectl get all
```

NAME	READY	STATUS	RESTARTS	AGE
pod/nginx-deploy-844595f8cd-tbp6t	1/1	Running	0	8m7s
pod/nginx-deploy-844595f8cd-vrk74	1/1	Running	0	7m56s
pod/nginx-deploy-844595f8cd-znf78	1/1	Running	0	8m1s

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	175m

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/nginx-deploy	3/3	3	3	91m

NAME	DESIRED	CURRENT	READY	AGE
replicaset.apps/nginx-deploy-6d78d4997c	0	0	0	91m
replicaset.apps/nginx-deploy-844595f8cd	3	3	3	84m
replicaset.apps/nginx-deploy-db7c4b8ff	0	0	0	82m

레플리카셋에 생성된 팟의 개수를 보면, 이전 레플리카셋으로 롤백된것을 알 수 있습니다.

#### ▼ rollout history를 볼때에 왜 변경된 사유가 <none> 인걸까요?

CHANGE-CAUSE를 <none> 이 아니도록 하고싶다면 deployment를 생성할 때나, rollout을 일으키는 명령어를 날릴때에 `--record` 옵션을 사용합니다. Describe 를 통해 확인해보면 annotations 항목에 어떤 명령어로 생성된 Revision인지 기록되어 있는걸 볼 수 있습니다. (현재 이 명령어를 치면 deprecated되었다고 나옵니다.)

아니면 deployment를 생성하고 `kubectl annotate deployment/nginx-deployment`  
`kubernetes.io/change-cause="어쩌구저쩌구"` 로 주석을 달거나 수동으로 리소스 매니페스트를 편집합니다.

## pod-template-hash

어떻게 각 Revision 마다 팟이 겹치지 않고 관리될 수 있는걸까요? 분명 ReplicaSet을 공부할 때, Selector는 기존에 생성되어있던 팟이 동일한 Label을 가지고 있는 걸 고려하지 않는다고 했었는데?

이것은 pod-template-hash를 통해 관리됩니다. Deloyment를 통해 ReplicaSet을 생성하면 ReplicaSet의 이름에 자동으로 해시값이 붙어서 생성이 되는데, 이 해시값은 ReplicaSet의 Selector에 자동으로 추가가 되고, 이 ReplicaSet이 생성한 팟들은 이 해시값을 라벨로 가집니다.

```
🍏 ~/Study/k8s ➤ kubectl describe rs nginx-deploy-6d78d4997c
Name:          nginx-deploy-6d78d4997c
Namespace:     default
Selector:      app=nginx-demo, pod-template-hash=6d78d4997c
Labels:        app=nginx-demo
               pod-template-hash=6d78d4997c
Annotations:   deployment.kubernetes.io/desired-replicas: 3
               deployment.kubernetes.io/max-replicas: 4
               deployment.kubernetes.io/revision: 1
Controlled By: Deployment/nginx-deploy
Replicas:      3 current / 3 desired
Pods Status:   3 Running / 0 Waiting / 0 Succeeded / 0 Failed
Pod Template:
  Labels:  app=nginx-demo
           pod-template-hash=6d78d4997c
  Containers:
    nginx:
      Image:      nginx
      Port:       <none>
      Host Port:  <none>
      Environment: <none>
```

우리가 디플로이먼트 YAML을 작성할때는 없었던 pod-template-hash가 셀렉터로 추가되어있습니다.

## 스케일링



\_\_\_\_\_아직 작성중  
\_\_\_\_\_