# [Hands-on] 11. Kubernetes Deployment strategies

이번 실습은 Deployment의 업데이트 방법 두 가지를 비교해보는 실습입니다.

그리고, 이번 실습은 Terminal이 두 개 필요합니다.
미리 준비해주세요.

## Recreate

첫 번째는 **Recreate** 입니다.
말 그대로 **다시 생성**하는 방법입니다. 기존에 서비스되고 있던 Pod들을 모두 정지하고, 새로운 Pod를 실행하는거죠.

Deployment 의 `.spec.strategy`를 아래와 같이 지정하면 됩니다.

```
spec:
  strategy:
    type: Recreate
```

아래는 **Recreate** 방식을 적용한 **Deployment** 예제파일입니다.
실습을 위해 아래 파일을 만들어주세요.

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: my-nginx
    tier: frontend
spec:
  replicas: 3
  strategy:
    type: Recreate
  selector:
    matchLabels:
      app: my-nginx
  template:
    metadata:
      labels:
        app: my-nginx
      name: my-nginx
    spec:
      containers:
      - image: nginx:1.18
        name: my-nginx
        ports:
        - containerPort: 80
```

> 파일명은 nginx-recreate.yaml로 합니다.

그리고, 다음과 같이 **Deployment**를 생성합니다.

```
ubuntu@ip-10-0-1-161:~$ kubectl apply -f nginx-recreate.yaml
deployment.apps/nginx-deployment created
```

> **명령어** : `kubectl apply -f nginx-recreate.yaml`

그리고, 생성된 Object들도 확인해 보겠습니다.

```
ubuntu@ip-10-0-1-161:~$ kubectl get all
NAME                                    READY   STATUS    RESTARTS   AGE
pod/nginx-deployment-5777d8dcc8-8s8cd   1/1     Running   0          109s
pod/nginx-deployment-5777d8dcc8-mbzpz   1/1     Running   0          109s
pod/nginx-deployment-5777d8dcc8-nhqmd   1/1     Running   0          109s

NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   3h20m

NAME                               READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/nginx-deployment   3/3     3            3           110s

NAME                                          DESIRED   CURRENT   READY   AGE
replicaset.apps/nginx-deployment-5777d8dcc8   3         3         3       109s
```

> **명령어** : `kubectl get all`

Spec에 정의된 대로 세 개의 Nginx Pod가 생성되어 있습니다.

우리가 생성한 nginx 버젼을 알아볼까요?

```
ubuntu@ip-10-0-1-161:~$ kubectl describe deployment nginx-deployment | grep -i image
    Image:        nginx:1.18
```

> **명령어** : `kubectl describe deployment nginx-deployment | grep -i image`

사용된 Image는 `nginx:1.18` 입니다.

이제 버젼을 변경하려고 합니다.
앞에서 배웠으니 **선언형(Declarative)**으로 해볼게요.

yaml파일의 버젼부분을 수정합니다. (`image: nginx:1.18` -> `image: nginx:1.19` , `sed` 명령어 사용)

```
ubuntu@ip-10-0-1-161:~$ sed -i 's/image: nginx:1.18/image: nginx:1.19/g' nginx-recreate.yaml
```

> **명령어** : `sed -i 's/image: nginx:1.18/image: nginx:1.19/g' nginx-recreate.yaml`

그리고, Pod들이 어떻게 변하는지 살펴보기 위해서 다음 명령어를 실행해주세요.
이 명령어는 두 번째 Terminal에서 실행해주세요.

```
ubuntu@ip-10-0-1-161:~/mspt2$ kubectl get pods --watch
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-5777d8dcc8-8s8cd   1/1     Running   0          11m
nginx-deployment-5777d8dcc8-mbzpz   1/1     Running   0          11m
nginx-deployment-5777d8dcc8-nhqmd   1/1     Running   0          11m
```

> **명령어** : `kubectl get pods --watch`
> `--watch` 는 앞의 명령어를 실행한 후 변경(Change)사항을 지속적으로 보여주는 Flag입니다.
> Watch를 멈추려면 Ctrl + c 를 입력합니다.

이제 다시 첫 번째 Terminal에서 아래와 같이 변경사항을 적용합니다.

그리고 변경된 yaml파일을 이용해서 업데이트를 합니다.

```
ubuntu@ip-10-0-1-161:~$ kubectl apply -f nginx-recreate.yaml
deployment.apps/nginx-deployment configured
```

> **명령어** : `kubectl apply -f nginx-recreate.yaml`

두 번째 Terminal에서 어떤 일이 일어나는지 유심히 보세요. 아마도, 있던 Pod들이 모두 삭제되고 새로운 Pod들이 생길거예요.

```
ubuntu@ip-10-0-1-161:~$ kubectl get pods --watch
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-5777d8dcc8-8s8cd   1/1     Running   0          11m
nginx-deployment-5777d8dcc8-mbzpz   1/1     Running   0          11m
nginx-deployment-5777d8dcc8-nhqmd   1/1     Running   0          11m

nginx-deployment-5777d8dcc8-nhqmd   1/1     Terminating   0          17m
nginx-deployment-5777d8dcc8-8s8cd   1/1     Terminating   0          17m
nginx-deployment-5777d8dcc8-mbzpz   1/1     Terminating   0          17m
nginx-deployment-5777d8dcc8-nhqmd   0/1     Terminating   0          17m
nginx-deployment-5777d8dcc8-nhqmd   0/1     Terminating   0          17m
nginx-deployment-5777d8dcc8-nhqmd   0/1     Terminating   0          17m
nginx-deployment-5777d8dcc8-mbzpz   0/1     Terminating   0          17m
nginx-deployment-5777d8dcc8-8s8cd   0/1     Terminating   0          17m
nginx-deployment-5777d8dcc8-mbzpz   0/1     Terminating   0          17m
nginx-deployment-5777d8dcc8-mbzpz   0/1     Terminating   0          17m
nginx-deployment-5777d8dcc8-8s8cd   0/1     Terminating   0          17m
nginx-deployment-5777d8dcc8-8s8cd   0/1     Terminating   0          17m
nginx-deployment-6866dc769c-zjv2b   0/1     Pending       0          0s
nginx-deployment-6866dc769c-lk922   0/1     Pending       0          0s
nginx-deployment-6866dc769c-zjv2b   0/1     Pending       0          0s
nginx-deployment-6866dc769c-vzgxc   0/1     Pending       0          0s
nginx-deployment-6866dc769c-lk922   0/1     Pending       0          0s
nginx-deployment-6866dc769c-vzgxc   0/1     Pending       0          0s
nginx-deployment-6866dc769c-zjv2b   0/1     ContainerCreating   0          0s
nginx-deployment-6866dc769c-lk922   0/1     ContainerCreating   0          1s
nginx-deployment-6866dc769c-vzgxc   0/1     ContainerCreating   0          2s
nginx-deployment-6866dc769c-zjv2b   1/1     Running             0          9s
nginx-deployment-6866dc769c-lk922   1/1     Running             0          12s
nginx-deployment-6866dc769c-vzgxc   1/1     Running             0          14s
```

Deployment에 어떤 변화가 생겼나 볼까요?

```
ubuntu@ip-10-0-1-161:~$ kubectl describe deployment nginx-deployment | grep -i image
    Image:        nginx:1.19
```

> **명령어** : `kubectl describe deployment nginx-deployment | grep -i image`

그리고, 새로 생성된 Pod도 한번 보구요.

```
ubuntu@ip-10-0-1-161:~$ kubectl describe pod nginx-deployment-6866dc769c-lk922 | grep -i image
    Image:          nginx:1.19
    Image ID:       docker-pullable://nginx@sha256:df13abe416e37eb3db4722840dd479b00ba193ac6606e7902331dcea50f4f1f2
  Normal  Pulling    5m46s  kubelet            Pulling image "nginx:1.19"
  Normal  Pulled     5m36s  kubelet            Successfully pulled image "nginx:1.19" in 9.660078609s
```

> **명령어** : `kubectl describe pod [POD-NAME] | grep -i image`
> [POD-NAME] 에는 앞에서 조회된 POD 중 하나의 이름을 넣어주세요.

어떤가요? 업데이트가 잘 이루어졌나요?

이제 두 번째 터미널은 Ctrl + c 를 눌러 Watch를 멈추겠습니다.

업데이트에 문제가 생기면 **롤백**도 할 수 있습니다.
이번에는 Deployment의 **롤백** 방법을 알아보겠습니다.

먼저 업데이트 History는 아래와 같이 확인해볼 수 있습니다.

```
ubuntu@ip-10-0-1-161:~$ kubectl rollout history deployment nginx-deployment
deployment.apps/nginx-deployment
REVISION  CHANGE-CAUSE
1         <none>
2         <none>
```

> **명령어** : `kubectl rollout history deployment nginx-deployment`

최초 생성된 **Revision #1**과 한 번 업데이트 후의 **Revision #2**가 보입니다.
그 중 하나의 Revision을 콕 집어서 자세히 볼 수도 있습니다.

```
ubuntu@ip-10-0-1-161:~$ kubectl rollout history deployment nginx-deployment --revision=1
deployment.apps/nginx-deployment with revision #1
Pod Template:
  Labels:       app=my-nginx
        pod-template-hash=5777d8dcc8
  Containers:
   my-nginx:
    Image:      nginx:1.18
    Port:       80/TCP
    Host Port:  0/TCP
    Environment:        <none>
    Mounts:     <none>
  Volumes:      <none>
```

> **명령어** : `kubectl rollout history deployment nginx-deployment --revision=1`

```
ubuntu@ip-10-0-1-161:~$ kubectl rollout history deployment nginx-deployment --revision=2
deployment.apps/nginx-deployment with revision #2
Pod Template:
  Labels:       app=my-nginx
        pod-template-hash=6866dc769c
  Containers:
   my-nginx:
    Image:      nginx:1.19
    Port:       80/TCP
    Host Port:  0/TCP
    Environment:        <none>
    Mounts:     <none>
  Volumes:      <none>
```

> **명령어** : `kubectl rollout history deployment nginx-deployment --revision=2`

역시 두 번째 Terminal에 어떤 변화가 일어날지 모니터할 준비를 하고,

```
ubuntu@ip-10-0-1-161:~$ kubectl get pods --watch
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-6866dc769c-lk922   1/1     Running   0          17m
nginx-deployment-6866dc769c-vzgxc   1/1     Running   0          17m
nginx-deployment-6866dc769c-zjv2b   1/1     Running   0          17m
```

> **명령어** : `kubectl get pods --watch`
> `--watch` 는 앞의 명령어를 실행한 후 변경(Change)사항을 지속적으로 보여주는 Flag입니다.
> Watch를 멈추려면 Ctrl + c 를 입력합니다.

첫 번째 Terminal에서 revision1으로 롤백 합니다.

```
ubuntu@ip-10-0-1-161:~$ kubectl rollout undo deployment nginx-deployment --to-revision=1
deployment.apps/nginx-deployment rolled back
```

> **명령어** : `kubectl rollout undo deployment nginx-deployment --to-revision=1`

두 번째 터미널에는 업데이트 할 때와 비슷한 변경내용을 볼 수 있을겁니다.
Pod들을 먼저 삭제하고, 새로은 Pod들을 만드는 걸 볼 수 있습니다.

이전 버젼으로 롤백이 잘 됐는지 아래 명령어로 확인해보세요.

```
ubuntu@ip-10-0-1-161:~$ kubectl describe po nginx-deployment-5777d8dcc8-9ds76 | grep -i image
    Image:          nginx:1.18
    Image ID:       docker-pullable://nginx@sha256:e90ac5331fe095cea01b121a3627174b2e33e06e83720e9a934c7b8ccc9c55a0
  Normal  Pulled     3m50s  kubelet            Container image "nginx:1.18" already present on machine
```

> **명령어** : `kubectl describe pod [POD-NAME] | grep -i image`
> [POD-NAME] 에는 앞에서 조회된 POD 중 하나의 이름을 넣어주세요.

다 해보셨으면 다음 실습을 위해 Object들을 삭제해주세요.
아시죠? **선언형(Declarative)**...

```
ubuntu@ip-10-0-1-161:~$ kubectl delete -f nginx-recreate.yaml
deployment.apps "nginx-deployment" deleted
```

> **명령어** : `kubectl delete -f nginx-recreate.yaml`

## RollingUpdate

이번엔 RollingUpdate 입니다.
기존에 서비스되고 있던 Pod들을 새로운 Pod로 조금씩(N개씩) 업데이트 하는 방식입니다.

Deployment 의 `.spec.strategy`를 아래와 같이 지정하면 됩니다.

```
spec:
  strategy:
    type: RollingUpdate
```

아래는 **RollingUpdate** 방식을 적용한 **Deployment** 예제파일입니다.
실습을 위해 아래 파일을 만들어주세요.

```yaml
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:1.18
        name: my-nginx
        ports:
        - containerPort: 80
```

> 파일명은 nginx-rollingupdate.yaml로 합니다.

다음과 같이 Deployment를 생성합니다.

```
ubuntu@ip-10-0-1-161:~$ kubectl apply -f nginx-rollingupdate.yaml
deployment.apps/nginx-deployment created
```

> **명령어 :** `kubectl apply -f nginx-rollingupdate.yaml`

그리고, 생성된 Object들도 확인해 보겠습니다.

```
ubuntu@ip-10-0-1-161:~$ kubectl get all
NAME                                    READY   STATUS    RESTARTS   AGE
pod/nginx-deployment-5777d8dcc8-bl2f2   1/1     Running   0          90s
pod/nginx-deployment-5777d8dcc8-p2qtl   1/1     Running   0          90s
pod/nginx-deployment-5777d8dcc8-sj4b2   1/1     Running   0          90s

NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   4h12m

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/nginx-deployment    3/3     3            3           90s

NAME                                           DESIRED   CURRENT   READY   AGE
replicaset.apps/nginx-deployment-5777d8dcc8    3         3         3       90s
```

> **명령어 :** `kubectl get all`

생성된 Deployment의 정보를 보고 현재 실행된 이미지를 확인해봅니다.

```
ubuntu@ip-10-0-1-161:~$ kubectl describe deployment nginx-deployment | grep -i image
    Image:        nginx:1.18
```

> **명령어 :** `kubectl describe deployment nginx-deployment | grep -i image`

사용된 Image는 `nginx:1.18` 입니다.

업데이트를 위해서 Deployment yaml파일에서 버젼을 변경하구요.

```
ubuntu@ip-10-0-1-161:~$ sed -i 's/image: nginx:1.18/image: nginx:1.19/g' nginx-rollingupdate.yaml
```

> **명령어 :** `sed -i 's/image: nginx:1.18/image: nginx:1.19/g' nginx-rollingupdate.yaml`

두 번째 Terminal에는 확인할 명령어를 실행한 후에

```
ubuntu@ip-10-0-1-161:~$ kubectl get pods --watch
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-5777d8dcc8-bl2f2   1/1     Running   0          7m21s
nginx-deployment-5777d8dcc8-p2qtl   1/1     Running   0          7m21s
nginx-deployment-5777d8dcc8-sj4b2   1/1     Running   0          7m21s
```

> **명령어 :** `kubectl get pods --watch`

첫 번째 Terminal에서 업데이트를 합니다.

```
ubuntu@ip-10-0-1-161:~$ kubectl apply -f nginx-rollingupdate.yaml
deployment.apps/nginx-deployment configured
```

> **명령어 :** `kubectl apply -f nginx-rollingupdate.yaml`

두 번째 Terminal은 아래와 비슷한 걸 볼 수 있을겁니다. **Recreate**때와는 달리 Pod들이 순차적으로 변경되는 걸 볼 수 있습니다.

```
ubuntu@ip-10-0-1-161:~/mspt2$ kubectl get pods --watch
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-5777d8dcc8-bl2f2   1/1     Running   0          7m21s
nginx-deployment-5777d8dcc8-p2qtl   1/1     Running   0          7m21s
nginx-deployment-5777d8dcc8-sj4b2   1/1     Running   0          7m21s

nginx-deployment-6866dc769c-tld28   0/1     Pending   0          0s
nginx-deployment-6866dc769c-tld28   0/1     Pending   0          0s
nginx-deployment-6866dc769c-tld28   0/1     ContainerCreating   0          0s
nginx-deployment-6866dc769c-tld28   1/1     Running             0          1s
nginx-deployment-5777d8dcc8-bl2f2   1/1     Terminating         0          8m11s
nginx-deployment-6866dc769c-vznpg   0/1     Pending             0          0s
nginx-deployment-6866dc769c-vznpg   0/1     Pending             0          0s
nginx-deployment-6866dc769c-vznpg   0/1     ContainerCreating   0          0s
nginx-deployment-6866dc769c-vznpg   1/1     Running             0          1s
nginx-deployment-5777d8dcc8-bl2f2   0/1     Terminating         0          8m12s
nginx-deployment-5777d8dcc8-sj4b2   1/1     Terminating         0          8m12s
nginx-deployment-6866dc769c-t2p6n   0/1     Pending             0          0s
nginx-deployment-5777d8dcc8-bl2f2   0/1     Terminating         0          8m12s
nginx-deployment-6866dc769c-t2p6n   0/1     Pending             0          0s
nginx-deployment-5777d8dcc8-bl2f2   0/1     Terminating         0          8m12s
nginx-deployment-6866dc769c-t2p6n   0/1     ContainerCreating   0          0s
nginx-deployment-5777d8dcc8-sj4b2   0/1     Terminating         0          8m13s
nginx-deployment-5777d8dcc8-sj4b2   0/1     Terminating         0          8m13s
nginx-deployment-5777d8dcc8-sj4b2   0/1     Terminating         0          8m13s
nginx-deployment-6866dc769c-t2p6n   1/1     Running             0          1s
nginx-deployment-5777d8dcc8-p2qtl   1/1     Terminating         0          8m13s
nginx-deployment-5777d8dcc8-p2qtl   0/1     Terminating         0          8m14s
nginx-deployment-5777d8dcc8-p2qtl   0/1     Terminating         0          8m14s
nginx-deployment-5777d8dcc8-p2qtl   0/1     Terminating         0          8m14s
```

첫 번째 Terminal에서 아래와 같이 업데이트 이후의 변경사항도 확인해보세요.

```
ubuntu@ip-10-0-1-161:~$ kubectl describe deployment nginx-deployment | grep -i image
    Image:        nginx:1.19
```

> **명령어 :** `kubectl describe deployment nginx-deployment | grep -i image`

새로 생성된 Pod의 정보도 확인해봅니다.

```
ubuntu@ip-10-0-1-161:~$ kubectl describe pod nginx-deployment-6866dc769c-t2p6n | grep -i image
    Image:          nginx:1.19
    Image ID:       docker-pullable://nginx@sha256:df13abe416e37eb3db4722840dd479b00ba193ac6606e7902331dcea50f4f1f2
  Normal  Pulled     6m55s  kubelet            Container image "nginx:1.19" already present on machine
```

> **명령어 :** `kubectl describe pod [POD-NAME] | grep -i image`
> [POD-NAME] 에는 앞에서 조회된 POD 중 하나의 이름을 넣어주세요.

앞에서와 마찬가지로 롤백도 해보세요.. 자세한 설명은 생략합니다.

`kubectl rollout history deployment nginx-deployment`

`kubectl get pods --watch`

`kubectl rollout undo deployment nginx-deployment --to-revision=1`

`kubectl describe pod [POD-NAME] | grep -i image`

`kubectl delete -f nginx-rollingupdate.yaml`