

[Hands-on] 14. Kubernetes Volume

PVC를 통해서 PV를 동적으로 Provisioning하고 구성하는 실습입니다.

이 실습은 실행 환경에 따라 동적 생성이 안될 수도 있으니, 그 경우에는 실습 환경에 맞게 설정을 변경하여 진행하시기 바랍니다.

먼저 Wordpress 실행을 위한 파일들을 준비하겠습니다.

```
ubuntu@ip-10-0-1-161:~$ curl -LO https://k8s.io/examples/application/wordpress/mysql-deployment.yaml
% Total    % Received % Xferd Average Speed   Time    Time     Time  Current
           Dload  Upload   Total   Spent    Left  Speed
100  178  100  178    0     0   423      0 --:--:-- --:--:-- --:--:--   424
100 1193  100 1193    0     0  1588      0 --:--:-- --:--:-- --:--:--  1588

ubuntu@ip-10-0-1-161:~$ curl -LO https://k8s.io/examples/application/wordpress/wordpress-deployment.yaml
% Total    % Received % Xferd Average Speed   Time    Time     Time  Current
           Dload  Upload   Total   Spent    Left  Speed
100  178  100  178    0     0   538      0 --:--:-- --:--:-- --:--:--   539
100 1278  100 1278    0     0  1789      0 --:--:-- --:--:-- --:--:--  1789

ubuntu@ip-10-0-1-161:~$ ls -al *-deployment.yaml
-rw-rw-r-- 1 ubuntu ubuntu 1193 Jul  7 16:46 mysql-deployment.yaml
-rw-rw-r-- 1 ubuntu ubuntu 1278 Jul  7 16:47 wordpress-deployment.yaml
```

명령어1 : `curl -LO https://k8s.io/examples/application/wordpress/mysql-deployment.yaml`

명령어2 : `curl -LO https://k8s.io/examples/application/wordpress/wordpress-deployment.yaml`

Docker & Kubernetes - [Hands-on] 14. Kubernetes Volume

MySQL 실행에 필요한 Secret을 하나 만들겠습니다.

```
ubuntu@ip-10-0-1-161:~$ kubectl create secret generic mysql-pass --from-literal=password=pwd
secret/mysql-pass created
```

명령어1 : `kubectl create secret generic mysql-pass --from-literal=password=pwd`

이제 MySQL을 실행할텐데요, 미리 받아둔 파일(mysql-deployment.yaml)을 이용합니다.

파일 내용은 아래와 같습니다.

```
apiVersion: v1
kind: Service
metadata:
  name: wordpress-mysql
  labels:
    app: wordpress
spec:
  ports:
    - port: 3306
  selector:
    app: wordpress
    tier: mysql
  clusterIP: None
---
```

MySQL의 Service

Docker & Kubernetes - [Hands-on] 14. Kubernetes Volume

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mysql-pv-claim
  labels:
    app: wordpress
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 20Gi
---
```

MySQL에서 사용할 PVC (동적 Provisioning)

Docker & Kubernetes - [Hands-on] 14. Kubernetes Volume

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wordpress-mysql
  labels:
    app: wordpress
spec:
  selector:
    matchLabels:
      app: wordpress
      tier: mysql
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: wordpress
        tier: mysql
    spec:
      containers:
        - image: mysql:5.6
          name: mysql
          env:
            - name: MYSQL_ROOT_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: mysql-pass
                  key: password
          ports:
            - containerPort: 3306
              name: mysql
          volumeMounts:
            - name: mysql-persistent-storage
              mountPath: /var/lib/mysql
      volumes:
        - name: mysql-persistent-storage
          persistentVolumeClaim:
            claimName: mysql-pv-claim
```

Docker & Kubernetes - [Hands-on] 14. Kubernetes Volume

설치를 진행합니다.

```
ubuntu@ip-10-0-1-161:~$ kubectl apply -f mysql-deployment.yaml
service/wordpress-mysql created
persistentvolumeclaim/mysql-pv-claim created
deployment.apps/wordpress-mysql created
```

명령어1 : `kubectl apply -f mysql-deployment.yaml`

설치된 Object들은 다음과 같이 조회합니다.

```
ubuntu@ip-10-0-1-161:~$ kubectl get all
```

NAME	READY	STATUS	RESTARTS	AGE
pod/wordpress-mysql-668d75584d-mjb6v	1/1	Running	0	71s

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	3d5h
service/wordpress-mysql	ClusterIP	None	<none>	3306/TCP	72s

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/wordpress-mysql	1/1	1	1	72s

NAME	DESIRED	CURRENT	READY	AGE
replicaset.apps/wordpress-mysql-668d75584d	1	1	1	72s

명령어1 : `kubectl get all`

Docker & Kubernetes - [Hands-on] 14. Kubernetes Volume

그리고 이번에는 PVC와 PV를 조회해보겠습니다.

```
ubuntu@ip-10-0-1-161:~$ kubectl get pvc
NAME          STATUS    VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
mysql-pv-claim Bound     pvc-a2771824-0d68-42d3-a2ac-072e8aef2265  20Gi       RWO            standard       5s

ubuntu@ip-10-0-1-161:~$ kubectl get pv
NAME          CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM          STORAGECLASS   REASON   AGE
pvc-a2771824-0d68-42d3-a2ac-072e8aef2265  20Gi       RWO            Delete           Bound    default/mysql-pv-claim  standard              9s
```

명령어1 : `kubectl get pvc` , `kubectl get pv`

PVC에 의해서 동적으로 PV가 생성된 걸 확인할 수 있습니다.

`kubectl describe` 명령으로 상세 내용도 확인해보세요.

```
ubuntu@ip-10-0-1-161:~$ kubectl describe pvc mysql-pv-claim
Name:          mysql-pv-claim
Namespace:     default
StorageClass:  standard
Status:        Bound
Volume:        pvc-a2771824-0d68-42d3-a2ac-072e8aef2265
Labels:        app=wordpress
Annotations:    pv.kubernetes.io/bind-completed: yes
                pv.kubernetes.io/bound-by-controller: yes
                volume.beta.kubernetes.io/storage-provisioner: k8s.io/minikube-hostpath
                volume.kubernetes.io/storage-provisioner: k8s.io/minikube-hostpath
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:      20Gi
Access Modes:  RWO
VolumeMode:    Filesystem
Used By:       wordpress-mysql-668d75584d-vwthh
Events:
  Type     Reason              Age   From                                     Message
  ----     -
  Normal   ExternalProvisioning  103s  persistentvolume-controller            waiting for a volume to be created, either by external provisioner "k8s.io/minikube-hostpath" or manually created by system administrator
  Normal   Provisioning         103s  k8s.io/minikube-hostpath_minikube_df641629-e39a-4e31-be5a-1750cf12e60d  External provisioner is provisioning volume for claim "default/mysql-pv-claim"
  Normal   ProvisioningSucceeded 103s  k8s.io/minikube-hostpath_minikube_df641629-e39a-4e31-be5a-1750cf12e60d  Successfully provisioned volume pvc-a2771824-0d68-42d3-a2ac-072e8aef2265
```

명령어1 : `kubectl describe pvc mysql-pv-claim`

Docker & Kubernetes - [Hands-on] 14. Kubernetes Volume

```
ubuntu@ip-10-0-1-161:~$ kubectl describe pv pvc-a2771824-0d68-42d3-a2ac-072e8aef2265
Name:          pvc-a2771824-0d68-42d3-a2ac-072e8aef2265
Labels:        <none>
Annotations:   hostPathProvisionerIdentity: dc1ac9b8-fa36-405d-83f9-747ba7d2c23f
                pv.kubernetes.io/provisioned-by: k8s.io/minikube-hostpath
Finalizers:    [kubernetes.io/pv-protection]
StorageClass:  standard
Status:        Bound
Claim:         default/mysql-pv-claim
Reclaim Policy: Delete
Access Modes:  RWX
VolumeMode:    Filesystem
Capacity:      20Gi
Node Affinity: <none>
Message:
Source:
  Type:        HostPath (bare host directory volume)
  Path:         /tmp/hostpath-provisioner/default/mysql-pv-claim
  HostPathType:
Events:        <none>
```

명령어1 : `kubectl describe pv pvc-a2771824-0d68-42d3-a2ac-072e8aef2265`

Docker & Kubernetes - [Hands-on] 14. Kubernetes Volume

그리고, Pod의 내용도 확인해볼까요?

```
ubuntu@ip-10-0-1-161:~$ kubectl get po
NAME                                READY   STATUS    RESTARTS   AGE
wordpress-mysql-668d75584d-vwthh   1/1     Running   0           4m15s

ubuntu@ip-10-0-1-161:~$ kubectl describe po wordpress-mysql-668d75584d-vwthh
Name:                                wordpress-mysql-668d75584d-vwthh
Namespace:                           default
Priority:                              0
Node:                                 minikube/192.168.49.2
...생략...
Environment:
  MYSQL_ROOT_PASSWORD: <set to the key 'password' in secret 'mysql-pass'> Optional: false
Mounts:
  /var/lib/mysql from mysql-persistent-storage (rw)
  /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-25qk8 (ro)
...생략...
Volumes:
  mysql-persistent-storage:
    Type:                PersistentVolumeClaim (a reference to a PersistentVolumeClaim in the same namespace)
    ClaimName:            mysql-pv-claim
    ReadOnly:             false
  kube-api-access-25qk8:
    Type:                 Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds: 3607
    ConfigMapName:         kube-root-ca.crt
    ConfigMapOptional:     <nil>
    DownwardAPI:           true
... 생략 ...
```

명령어1 : `kubectl get po` , `kubectl describe po [POD_NAME]`

Volumes부분과 Mounts 부분을 잘 확인해보세요.

Docker & Kubernetes - [Hands-on] 14. Kubernetes Volume

이제 Wordpress도 실행합니다.

```
ubuntu@ip-10-0-1-161:~$ kubectl apply -f wordpress-deployment.yaml
service/wordpress created
persistentvolumeclaim/wp-pv-claim created
deployment.apps/wordpress created
```

명령어1 : `kubectl apply -f wordpress-deployment.yaml`

그리고, MySQL과 마찬가지로 생성된

- PVC
- PV
- Pod

의 내용을 확인해보세요.

이번 실습은 여기까지 입니다. ~~~