

Contents

- PersistentVolumeClaim(PVC), PersistentVolume(PV) 생성하기
- Volume을 사용하여 Pod 생성하기

PersistentVolumeClaim(PVC), PersistentVolume(PV) 생성하기

도커에서와 마찬가지로 컨테이너의 데이터 저장을 위해서 Volume을 생성해 보겠습니다.
이번 실습은 다양한 방법 중에서

- Storage class를 이용한 [Dynamic Volume Provisioning](#) 을 적용하고,
- [hostPath](#) 유형의 Volume을 사용

해서 진행해 보겠습니다.

먼저 우리 환경이 준비가 되어있는지 확인해볼게요.
우리 Cluster에서 사용가능한 Storageclass가 있는지 확인합니다.

ubuntu@ip-172-31-20-30:~\$ kubectl get storageclasses						
NAME	PROVISIONER	RECLAIMPOLICY	VOLUMEBINDINGMODE	ALLOWVOLUMEEXPANSION	AGE	
standard (default)	k8s.io/minikube-hostpath	Delete	Immediate	false	11h	

명령어 : `kubectl get storageclasses`

Minikube는 기본적으로 위와같은 StorageClass가 있습니다.
간단히 테스트해볼 수 있도록, hostPath 타입의 Volume을 만들 수 있습니다.

Docker & Kubernetes - [Hands-on] 11. Kubernetes Storage

이제 PVC를 만들어볼게요.

아래와 같은 파일을 준비합니다.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: nginx-pvc
spec:
  storageClassName: standard
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 3Gi
```

파일명은 nginx-pvc.yaml로 합니다.

standard라는 이름의 StorageClass를 이용해서, 3Gi 용량의 PV를 요청(Claim)하는 것입니다.

그리고, 아래와 같이 적용합니다.

```
ubuntu@ip-172-31-20-30:~$ kubectl apply -f nginx-pvc.yaml
persistentvolumeclaim/nginx-pvc created
```

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

Docker & Kubernetes - [Hands-on] 11. Kubernetes Storage

만들어진 K8s 리소스들을 볼까요?

먼저 PVC를 확인해볼게요.

ubuntu@ip-172-31-20-30:~\$ kubectl get persistentvolumeclaims							
NAME	STATUS	VOLUME	CAPACITY	ACCESS MODES	STORAGECLASS	AGE	
nginx-pvc	Bound	pvc-5347efb3-5aaf-437e-ad7d-9be120c190fa	3Gi	RWO	standard	15s	

명령어 : `kubectl get persistentvolumeclaims` 또는 `kubectl get pvc`

결과를 보니 VOLUME(pvc-5347efb3-5aaf-437e-ad7d-9be120c190fa) 도 보이고, STATUS는 Bound네요.

그럼, 이번에는 PV를 볼까요?

ubuntu@ip-172-31-20-30:~\$ kubectl get persistentvolume									
NAME	CAPACITY	ACCESS MODES	RECLAIM POLICY	STATUS	CLAIM	STORAGECLASS	REASON	AGE	
pvc-5347efb3-5aaf-437e-ad7d-9be120c190fa	3Gi	RWO	Delete	Bound	default/nginx-pvc	standard		3m36s	

명령어 : `kubectl get persistentvolume` 또는 `kubectl get pv`

예상한대로 PVC를 생성했더니, PV가 동적으로 생성됐습니다.

일반적인 사용 유형이니 잘 익혀두세요.

Docker & Kubernetes - [Hands-on] 11. Kubernetes Storage

PV를 좀 더 자세히 볼까요?

```
ubuntu@ip-172-31-20-30:~$ kubectl describe persistentvolume pvc-5347efb3-5aaf-437e-ad7d-9be120c190fa
Name:           pvc-5347efb3-5aaf-437e-ad7d-9be120c190fa
Labels:         <none>
Annotations:   hostPathProvisionerIdentity: f5c4df7d-df7c-41dd-ba21-7392f383138d
               pv.kubernetes.io/provisioned-by: k8s.io/minikube-hostpath
Finalizers:    [kubernetes.io/pv-protection]
StorageClass:  standard
Status:        Bound
Claim:         default/nginx-pvc
Reclaim Policy: Delete
Access Modes:  RWO
VolumeMode:   Filesystem
Capacity:     3Gi
Node Affinity: <none>
Message:
Source:
  Type:      HostPath (bare host directory volume)
  Path:      /tmp/hostpath-provisioner/default/nginx-pvc
  HostPathType:
Events:        <none>
```

명령어 : `kubectl describe persistentvolume [PV-NAME]` 또는 `kubectl describe pv [PV-NAME]`
[PV-NAME] 에는 앞에서 만들어진 PV의 Name을 넣어주세요.

Source아래 내용을 보시면 어디에 Volume영역이 할당되었는지 알 수 있습니다.

위의 경우는 HostPath타입을 이용했고, `/tmp/hostpath-provisioner/default/nginx-pvc`를 Volume의 위치로 사용하고 있습니다.

Volume을 사용하여 Pod 생성하기

이제 만들어진 PVC, PV를 사용하는 Pod를 생성해 보겠습니다.

아래와 같이 volumes와 volumeMounts를 정의하려고 합니다.

```
...
  volumes:
    - name: nginx-storage
      persistentVolumeClaim:
        claimName: nginx-pvc
  containers:
    - image: nginx:1.19.3
      name: my-nginx
      ports:
        - containerPort: 80
      volumeMounts:
        - mountPath: "/usr/share/nginx/html"
          name: nginx-storage
```

앞에서 만든 nginx-pvc 를 사용하고, 컨테이너의 /usr/share/nginx/html를 마운트합니다.

Docker & Kubernetes - [Hands-on] 11. Kubernetes Storage

다음과 같이 Deployment를 준비해주세요.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-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
    spec:
      volumes:
        - name: nginx-storage
          persistentVolumeClaim:
            claimName: nginx-pvc
      containers:
        - image: nginx:1.19.3
          name: my-nginx
          ports:
            - containerPort: 80
          volumeMounts:
            - mountPath: "/usr/share/nginx/html"
              name: nginx-storage
```

파일명은 nginx-deployment-volume.yaml로 합니다.

Docker & Kubernetes - [Hands-on] 11. Kubernetes Storage

다음은 Deployment와 앞에서 실습한 Service, Ingress까지 리소스를 생성해주세요.

```
ubuntu@ip-172-31-20-30:~$ kubectl apply -f nginx-deployment-volume.yaml
deployment.apps/my-nginx-deployment created
ubuntu@ip-172-31-20-30:~$ kubectl apply -f nginx-clusterip-service.yaml
service/nginx-clusterip-service created
ubuntu@ip-172-31-20-30:~$ kubectl apply -f nginx-ingress.yaml
ingress.networking.k8s.io/my-nginx-ingress created
```

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

명령어 : `kubectl apply -f nginx-clusterip-service.yaml`

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

아직 한 가지 더 할 일이 남았습니다.

```
ubuntu@ip-172-31-20-30:~$ echo '<h1>Hello kubernetes</h1>' >> /tmp/hostpath-provisioner/default/nginx-pvc/index.html
```

명령어 : `echo '<h1>Hello kubernetes</h1>' >> /tmp/hostpath-provisioner/default/nginx-pvc/index.html`

Nginx에서 보여줄 간단한 `index.html` 파일을 하나 만들었습니다.
혹시 PV의 경로가 다르다면 거기에 맞춰서 해주세요.

- 이 실습은 PVC, PV, Pod의 동작을 살펴보기 위한 것입니다. HostPath 유형의 사용상 주의사항은 [hostPath](#) 를 참고하세요.

Docker & Kubernetes - [Hands-on] 11. Kubernetes Storage

이제 브라우저에서 어떻게 나오나 볼까요?

<http://my-nginx.info>



Hello kubernetes

Pod의 파일시스템에도 위의 내용이 반영되어 있는지도 확인해보세요.

```
ubuntu@ip-172-31-20-30:~/mspt3/hands_on_files$ kubectl get pod
NAME                  READY   STATUS    RESTARTS   AGE
my-nginx-deployment-7cbbdb88f6-56r2d   1/1     Running   0          166m
my-nginx-deployment-7cbbdb88f6-ppbwmm  1/1     Running   0          166m
my-nginx-deployment-7cbbdb88f6-whzp8   1/1     Running   0          166m
ubuntu@ip-172-31-20-30:~/mspt3/hands_on_files$ kubectl exec -it my-nginx-deployment-7cbbdb88f6-56r2d -- cat /usr/share/nginx/html/index.html
<h1>Hello kubernetes</h1>
```

명령어 : `kubectl get pod`

명령어 : `kubectl exec -it [POD-NAME] -- cat /usr/share/nginx/html/index.html`

[POD-NAME] 에는 앞에서 조회한 POD중 하나의 이름을 넣어주세요.

Docker & Kubernetes - [Hands-on] 11. Kubernetes Storage

아래와 같이 사용한 리소스들을 정리해주세요.

```
ubuntu@ip-172-31-20-30:~$ kubectl delete -f nginx-ingress.yaml
ingress.networking.k8s.io "my-nginx-ingress" deleted
ubuntu@ip-172-31-20-30:~$ kubectl delete -f nginx-clusterip-service.yaml
service "nginx-clusterip-service" deleted
ubuntu@ip-172-31-20-30:~$ kubectl delete -f nginx-deployment-volume.yaml
deployment.apps "my-nginx-deployment" deleted
ubuntu@ip-172-31-20-30:~$ kubectl delete -f nginx-pvc.yaml
persistentvolumeclaim "nginx-pvc" deleted
```

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

명령어 : `kubectl delete -f nginx-clusterip-service.yaml`

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

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

이번 실습은 여기까지입니다. ^_^

끝~

Docker & Kubernetes - [Hands-on] 11. Kubernetes Storage

보너스 실습

앞의 실습까지 하고도 시간이 남으면 해보세요.

도커 실습에서 사용한 ToDo App을 PVC를 사용해서 데이터를 저장하는 방법입니다.
Docker Volumes 실습의 Kubernetes 버전이라고 보시면 될 것 같아요.

실습에 필요한 파일은 모두 hands_on_files 아래에 있습니다.

아래 참고해서 해보세요.

PVC 생성 > Deployment 생성 > Service 생성 > Ingress 생성

```
ubuntu@ip-172-31-20-30:~/mspt3/hands_on_files$ kubectl apply -f todo-pvc.yaml
persistentvolumeclaim/todo-pvc created
ubuntu@ip-172-31-20-30:~/mspt3/hands_on_files$ kubectl apply -f todo-deployment-volume.yaml
deployment.apps/todo-app-deployment created
ubuntu@ip-172-31-20-30:~/mspt3/hands_on_files$ kubectl apply -f todo-clusterip-service.yaml
service/todo-clusterip-service created
ubuntu@ip-172-31-20-30:~/mspt3/hands_on_files$ kubectl apply -f todo-ingress.yaml
ingress.networking.k8s.io/todo-app-ingress created
```

명령어 : `kubectl apply -f todo-pvc.yaml`

명령어 : `kubectl apply -f todo-deployment-volume.yaml`

명령어 : `kubectl apply -f todo-clusterip-service.yaml`

명령어 : `kubectl apply -f todo-ingress.yaml`

Docker & Kubernetes - [Hands-on] 11. Kubernetes Storage

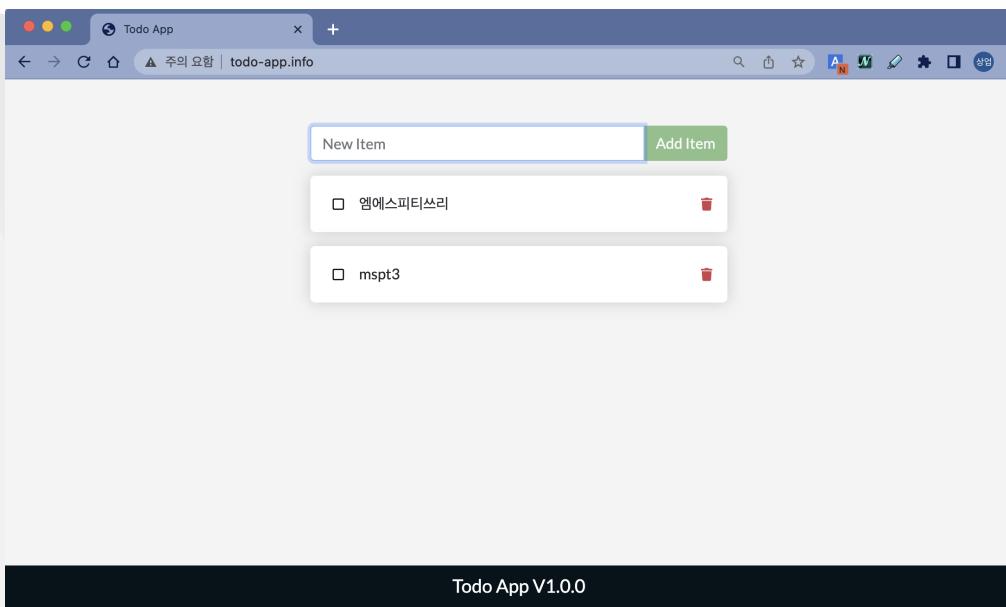
ToDo App 접속을 위해서 hosts파일에 다음과 같이 하나(*todo-app.info*)를 추가합니다.

- Windows라면 C:\Windows\System32\drivers\etc\hosts 파일에,
- Linux계열은 /etc/hosts 파일에 추가하면 됩니다.

```
#mspt3  
11.22.33.44 my-nginx.info todo-app.info
```

11.22.33.44 대신 여러분 EC2 Instance의 Public IPv4 address를 써주세요.

이제 브라우저에서 <http://todo-app.info/> 로 접속하면, 아래와 같이 접속 가능할거예요.



Docker & Kubernetes - [Hands-on] 11. Kubernetes Storage

그리고, 아래처럼 Pod들이 삭제와 생성을 반복해도 데이터는 사라지지 않고 유지될거예요.

```
ubuntu@ip-172-31-20-30:~/mspt3/hands_on_files$ kubectl get pod
NAME                  READY   STATUS    RESTARTS   AGE
todo-app-deployment-55464569cf-4zmfj   1/1     Running   0          9m38s
todo-app-deployment-55464569cf-9ndh8   1/1     Running   0          9m38s
todo-app-deployment-55464569cf-9psmb   1/1     Running   0          9m38s
ubuntu@ip-172-31-20-30:~/mspt3/hands_on_files$ kubectl delete pod --all
pod "todo-app-deployment-55464569cf-4zmfj" deleted
pod "todo-app-deployment-55464569cf-9ndh8" deleted
pod "todo-app-deployment-55464569cf-9psmb" deleted
ubuntu@ip-172-31-20-30:~/mspt3/hands_on_files$ kubectl get pod
NAME                  READY   STATUS    RESTARTS   AGE
todo-app-deployment-55464569cf-5wmcx   1/1     Running   0          9s
todo-app-deployment-55464569cf-d8sx4   1/1     Running   0          9s
todo-app-deployment-55464569cf-dlsrp   1/1     Running   0          9s
```

명령어 : `kubectl get pod`

명령어 : `kubectl delete pod --all`

명령어 : `kubectl get pod`

Docker & Kubernetes - [Hands-on] 11. Kubernetes Storage

다 해보셨으면, 깨끗이 정리하고 마칠게요.

```
ubuntu@ip-172-31-20-30:~/mspt3/hands_on_files$ kubectl delete -f todo-ingress.yaml
ingress.networking.k8s.io "todo-app-ingress" deleted
ubuntu@ip-172-31-20-30:~/mspt3/hands_on_files$ kubectl delete -f todo-clusterip-service.yaml
service "todo-clusterip-service" deleted
ubuntu@ip-172-31-20-30:~/mspt3/hands_on_files$ kubectl delete -f todo-deployment-volume.yaml
deployment.apps "todo-app-deployment" deleted
ubuntu@ip-172-31-20-30:~/mspt3/hands_on_files$ kubectl delete -f todo-pvc.yaml
persistentvolumeclaim "todo-pvc" deleted
```

명령어 : `kubectl delete -f todo-ingress.yaml`

명령어 : `kubectl delete -f todo-clusterip-service.yaml`

명령어 : `kubectl delete -f todo-deployment-volume.yaml`

명령어 : `kubectl delete -f todo-pvc.yaml`

정말로 끝~ ^_^