6th Week

여섯 번째 뵙겠습니다 ?!

▷ 잠시만 기다렸다가 30분 되면 시작하겠습니다~^^

▷ 이제 반환점에 왔습니다~ 우리 끝까지 같이 힘내요 !!!

- 아자! 아자! 파이팅!!!

▷ Camera는 가급적 켜 주시면 대단히 감사하겠습니다 !!!

- 너무 부끄러우면 Snap Camera를 사용하시는 것 까지는~ ^^

▷ 오늘 수업 자료는 아래 링크에서 다운로드 받으실 수 있어요.

- https://github.com/whatwant-school/kubernetes

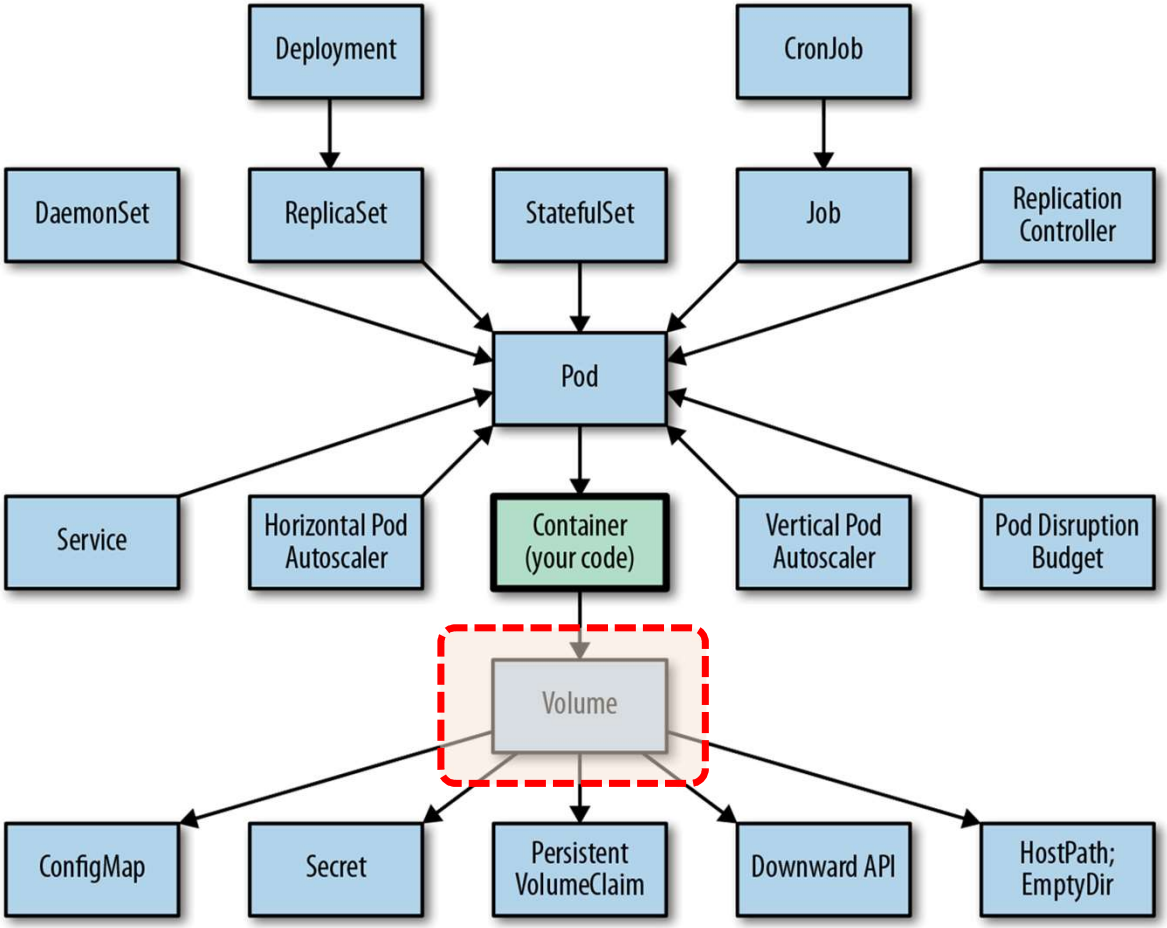
///

지난 수업 기억 나시나요?

https://kahoot.it/

///

Service



※ 참고 : https://www.oreilly.com/library/view/kubernetes-patterns/9781492050278/ch01.html

Flip Learning

(Volume - emptyDir / hostPath / (PV/PVC) / StorageClass)

이상윤 님

///

Kubernetes

Volume - Overview

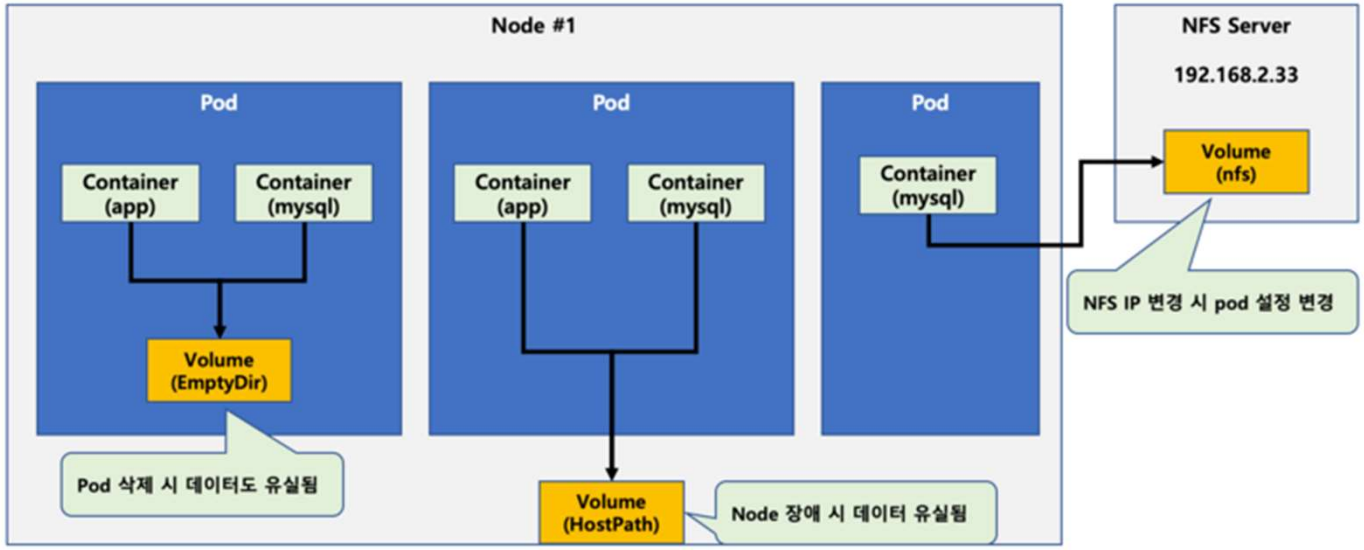
kinds of volume (2022.08.22 기준)

- awsElasticBlockStore (deprecated)
- azureDisk (deprecated)
- azureFile (deprecated)
- **cephfs**
- cinder (deprecated)
- **configMap**
- **downwardAPI**
- **emptyDir**
- **fc (파이버 채널)**
- flocker (deprecated)
- gcePersistentDisk (deprecated)
- gitRepo (deprecated)
- **glusterfs**
- **hostPath**
- **iscsi**
- **local**
- **nfs**
- **persistentVolumeClaim**
- **portworxVolume**
- **projected**
- quobyte (deprecated)
- **rbd**
- **secret**
- storageOS (deprecated)
- vsphereVolume (deprecated)

※ 참고 : https://kubernetes.io/ko/docs/concepts/storage/volumes/

Pod에서 직접 Volume을 지정하는 방식

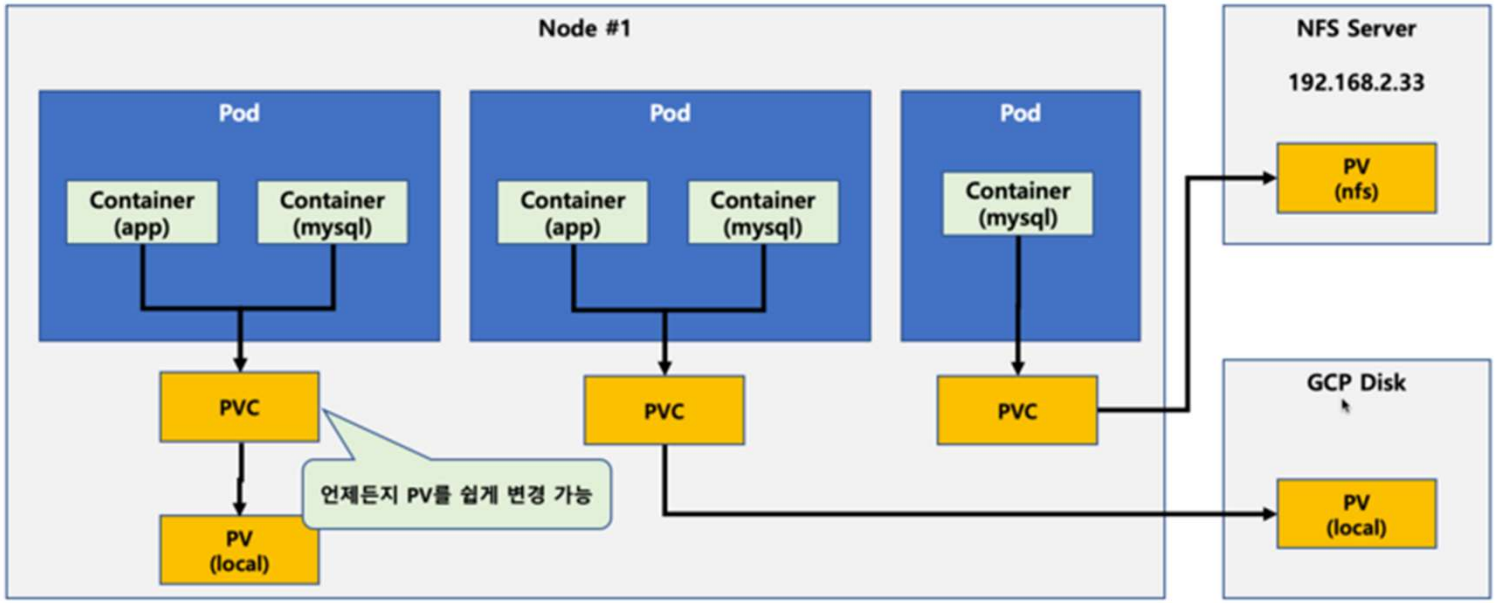
Temp	Local	Network
emptyDir	hostPath	nfs

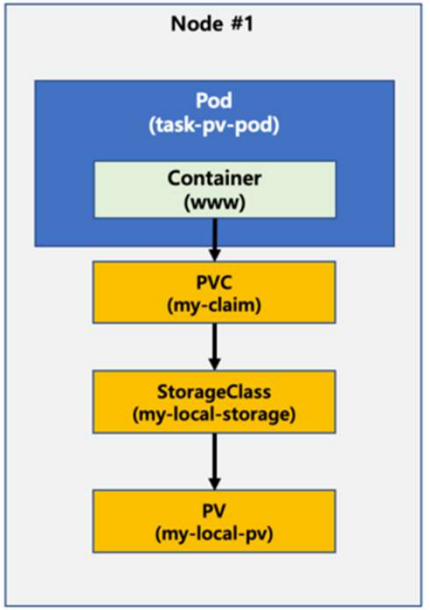


※ 참고 : https://www.slideshare.net/kubecon/kubecon-eu-2016-kubernetes-storage-101

※ 참고 : https://m.blog.naver.com/freepsw/222005161870

Pod에서 PersistentVolume을 활용하는 방식





※ 참고 : https://m.blog.naver.com/freepsw/222005161870

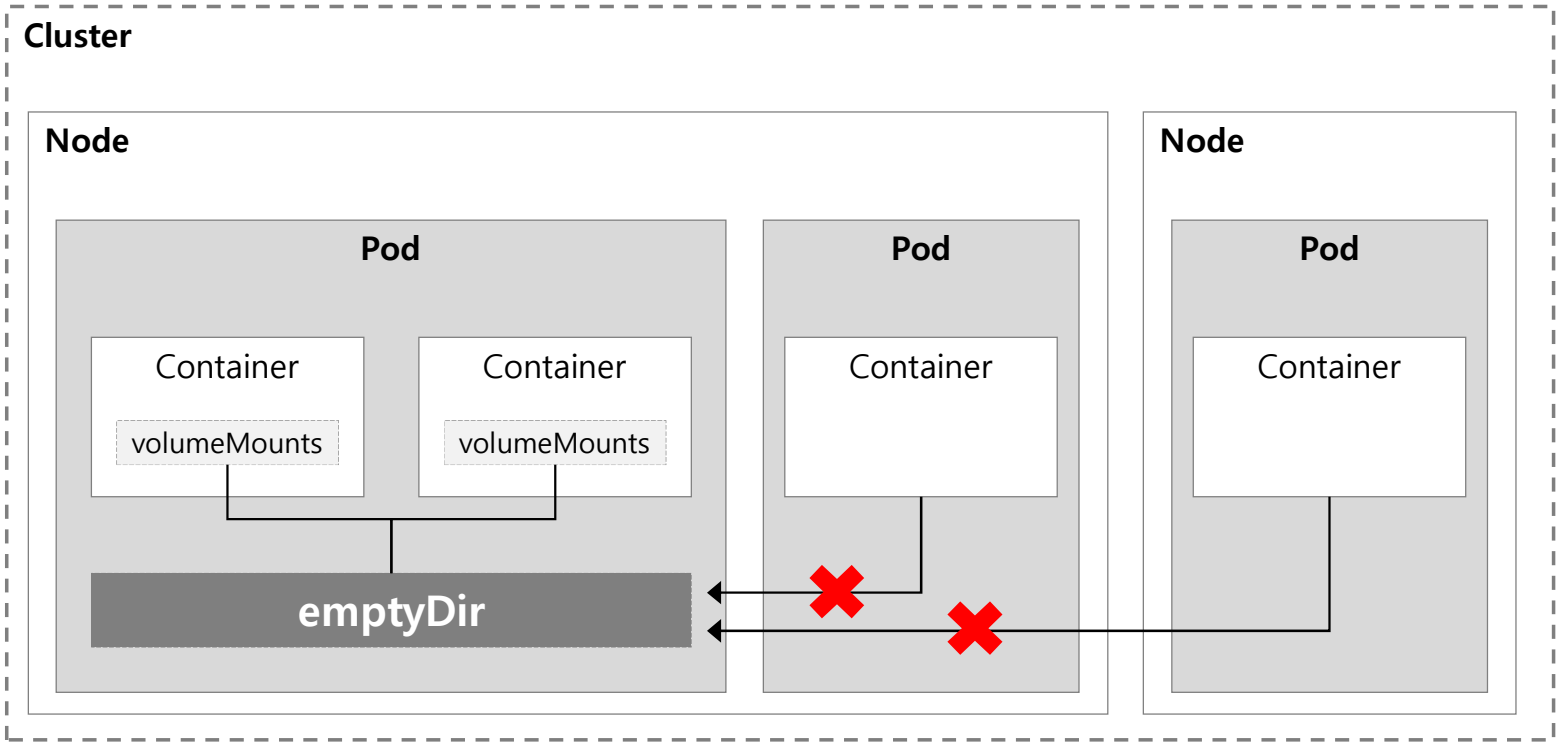
///

Volume - emptyDir

emptyDir

- Pod 안에서 Container끼리 공유 또는 Container가 재시작 하더라도 저장된 파일 유지

- Pod가 재시작 하는 경우에는 저장된 파일 유실



YAML

- 1개의 Pod 안에 2개의 container를 구성하고 각 container에서 emptyDir을 mount 하도록 구성

rs-emptyDir.yaml

```yaml
apiVersion: apps/v1
kind: ReplicaSet

metadata:
  name: rs-emptyDir

spec:
  replicas: 1

  selector:
    matchLabels:
      app: ubuntu

  template:
    metadata:
      labels:
        app: ubuntu
    spec:
      containers:
      - image: ubuntu:20.04
        name: worker1
        command: ["/bin/sleep", "3650d"]

        volumeMounts:
        - name: emptydir-demo
          mountPath: /data/emptyDir1

      - image: ubuntu:20.04
        name: worker2
        command: ["/bin/sleep", "3650d"]

        volumeMounts:
        - name: emptydir-demo
          mountPath: /data/emptyDir2

      volumes:
      - name: emptydir-demo
        emptyDir: {}
```

Create & Check

- 생성된 Pod 안에 있는 2개의 container에서 mount된 directory를 각각 확인

```
remote ❯ cd kubernetes/06-emptyDir-hostPath-PV/hands-on

remote ❯ kubectl create -f rs-emptyDir.yaml

replicaset.apps/rs-emptydir created


remote ❯ kubectl get pods -o wide

NAME                READY   STATUS    RESTARTS   AGE   IP              NODE      NOMINATED NODE   READINESS GATES
rs-emptydir-btf7g   2/2     Running   0          92s   10.233.103.95   worker2   <none>           <none>


remote ❯ kubectl exec -it rs-emptydir-btf7g -c worker1 -- ls -al /data

total 12
drwxr-xr-x 3 root root 4096 Aug 26 14:42 .
drwxr-xr-x 1 root root 4096 Aug 26 14:42 ..
drwxrwxrwx 2 root root 4096 Aug 26 14:42 emptyDir1


remote ❯ kubectl exec -it rs-emptydir-btf7g -c worker2 -- ls -al /data

total 12
drwxr-xr-x 3 root root 4096 Aug 26 14:42 .
drwxr-xr-x 1 root root 4096 Aug 26 14:42 ..
drwxrwxrwx 2 root root 4096 Aug 26 14:42 emptyDir2
```

Share Volume

- container 1번에서 생성한 파일이 container 2번에서도 공유되고 있음을 확인

```
remote ❯ kubectl exec -it rs-emptydir-btf7g -c worker1 -- ls -al /data/emptyDir1

total 8
drwxrwxrwx 2 root root 4096 Aug 26 14:42 .
drwxr-xr-x 3 root root 4096 Aug 26 14:42 ..

remote ❯ kubectl exec -it rs-emptydir-btf7g -c worker2 -- ls -al /data/emptyDir2

total 8
drwxrwxrwx 2 root root 4096 Aug 26 14:42 .
drwxr-xr-x 3 root root 4096 Aug 26 14:42 ..

remote ❯ kubectl exec -it rs-emptydir-btf7g -c worker1 -- touch /data/emptyDir1/wow

remote ❯ kubectl exec -it rs-emptydir-btf7g -c worker1 -- ls -al /data/emptyDir1

total 8
drwxrwxrwx 2 root root 4096 Aug 26 14:52 .
drwxr-xr-x 3 root root 4096 Aug 26 14:42 ..
-rw-r--r-- 1 root root    0 Aug 26 14:52 wow

remote ❯ kubectl exec -it rs-emptydir-btf7g -c worker2 -- ls -al /data/emptyDir2

total 8
drwxrwxrwx 2 root root 4096 Aug 26 14:52 .
drwxr-xr-x 3 root root 4096 Aug 26 14:42 ..
-rw-r--r-- 1 root root    0 Aug 26 14:52 wow
```

Restart

- Pod가 재시작 되었을 때, emptyDir 내용이 유지가 되는지를 보기 위한 실습이다. 당연히, 유지가 안된다.

```
remote ❯ kubectl exec -it rs-emptydir-btf7g -c worker1 -- ls -al /data/emptyDir1

total 8
drwxrwxrwx 2 root root 4096 Aug 26 14:52 .
drwxr-xr-x 3 root root 4096 Aug 26 14:42 ..
-rw-r--r-- 1 root root    0 Aug 26 14:52 wow


remote ❯ kubectl delete pods rs-emptydir-btf7g

pod "rs-emptydir-btf7g" deleted


remote ❯ kubectl get pods -o wide

NAME                READY   STATUS    RESTARTS   AGE   IP              NODE      NOMINATED NODE   READINESS GATES
rs-emptydir-pc6f9   2/2     Running   0          49s   10.233.103.96   worker2   <none>           <none>


remote ❯ kubectl exec -it rs-emptydir-pc6f9 -c worker1 -- ls -al /data/emptyDir1

total 8
drwxrwxrwx 2 root root 4096 Aug 26 15:40 .
drwxr-xr-x 3 root root 4096 Aug 26 15:40 ..
```

YAML - memory

- Memory 유형으로 emptyDir를 선언하면 tmpfs를 활용하게 된다. Memory 방식이기에 당연히 빠르다.

rs-emptyDir-memory.yaml

```yaml
apiVersion: apps/v1
kind: ReplicaSet

metadata:
  name: rs-emptydir-memory

spec:
  replicas: 1

  selector:
    matchLabels:
      app: ubuntu

  template:
    metadata:
      labels:
        app: ubuntu
    spec:
      containers:
      - image: ubuntu:20.04
        name: worker
        command: ["/bin/sleep", "3650d"]

        volumeMounts:
        - name: emptydir-demo
          mountPath: /data/emptyDir

      volumes:
      - name: emptydir-demo
        emptyDir:
          medium: Memory
```

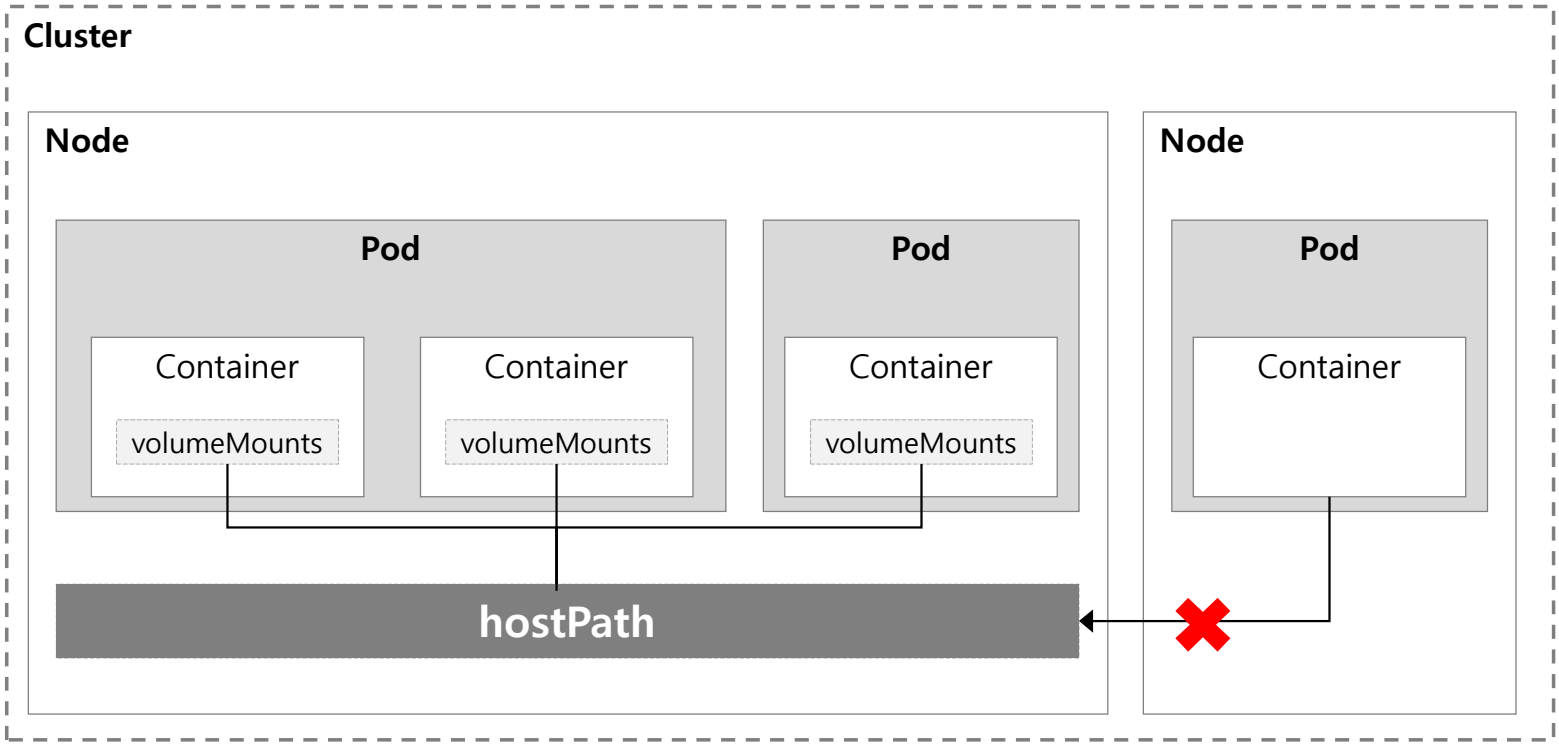
///

Volume - hostPath

hostPath

- 같은 Node에 있는 Pod끼리 서로 공유할 수 있고, 다른 Node에서는 참조할 수 없다

- Pod가 재시작 되더라도 hostPath에 저장된 내용은 유지 된다



Type of `hostPath`

값	행동
	빈 문자열 (기본값)은 이전 버전과의 호환성을 위한 것으로, hostPath 볼륨은 마운트 하기 전에 아무런 검사도 수행되지 않는다.
DirectoryOrCreate	만약 주어진 경로에 아무것도 없다면, 필요에 따라 Kubelet이 가지고 있는 동일한 그룹과 소유권, 권한을 0755로 설정한 빈 디렉터리를 생성한다.
Directory	주어진 경로에 디렉터리가 있어야 함
FileOrCreate	만약 주어진 경로에 아무것도 없다면, 필요에 따라 Kubelet이 가지고 있는 동일한 그룹과 소유권, 권한을 0644로 설정한 빈 디렉터리를 생성한다.
File	주어진 경로에 파일이 있어야 함
Socket	주어진 경로에 UNIX 소켓이 있어야 함
CharDevice	주어진 경로에 문자 디바이스가 있어야 함
BlockDevice	주어진 경로에 블록 디바이스가 있어야 함

※ 참고 : https://kubernetes.io/ko/docs/concepts/storage/volumes/#hostpath

YAML

- hostPath를 mount 하도록 구성한 Pod 3개를 생성

rs-hostPath.yaml

```yaml
apiVersion: apps/v1
kind: ReplicaSet

metadata:
  name: rs-hostpath

spec:
  replicas: 3

  selector:
    matchLabels:
      app: ubuntu

  template:
    metadata:
      labels:
        app: ubuntu
    spec:
      containers:
      - image: ubuntu:20.04
        name: worker1
        command: ["/bin/sleep", "3650d"]

        volumeMounts:
        - name: hostpath-demo
          mountPath: /data/hostpath

      volumes:
      - name: hostpath-demo
        hostPath:
          path: /tmp/hostpath-demo
          type: DirectoryOrCreate
```

Create & Check

- 서로 다른 Node에 있는 Pod 모두, 일단 hostPath가 mount되어 있는 것을 확인할 수 있다

- node가 다르더라도 일단 volume은 모두 mount 되어 있다.

```
remote ❯ cd kubernetes/06-emptyDir-hostPath-PV/hands-on
remote ❯ kubectl create -f rs-hostPath.yaml

replicaset.apps/rs-hostpath created

remote ❯ kubectl get pods -o wide

NAME               READY   STATUS    RESTARTS   AGE   IP              NODE      NOMINATED NODE   READINESS GATES
rs-hostpath-89rmm   1/1     Running   0          34s   10.233.103.98   worker2   <none>           <none>
rs-hostpath-n9z2t   1/1     Running   0          34s   10.233.110.41   worker1   <none>           <none>
rs-hostpath-xm6s8   1/1     Running   0          34s   10.233.103.97   worker2   <none>           <none>

remote ❯ kubectl exec -it rs-hostpath-n9z2t -- ls -al /data

total 12
drwxr-xr-x 3 root root 4096 Aug 26 16:11 .
drwxr-xr-x 1 root root 4096 Aug 26 16:11 ..
drwxr-xr-x 2 root root 4096 Aug 26 16:11 hostpath

remote ❯ kubectl exec -it rs-hostpath-89rmm -- ls -al /data

total 12
drwxr-xr-x 3 root root 4096 Aug 26 16:11 .
drwxr-xr-x 1 root root 4096 Aug 26 16:11 ..
drwxr-xr-x 2 root root 4096 Aug 26 16:11 hostpath
```

Share Volume

- Node가 같으면 같은 volume이 share 되지만, Node가 다른 경우에는 volume이 share되지 않는다.

```
remote ❯ kubectl get pods -o wide

NAME                READY   STATUS    RESTARTS   AGE   IP              NODE      NOMINATED NODE   READINESS GATES
rs-hostpath-89rmm   1/1     Running   0          34s   10.233.103.98   worker2   <none>           <none>
rs-hostpath-n9z2t   1/1     Running   0          34s   10.233.110.41   worker1   <none>           <none>
rs-hostpath-xm6s8   1/1     Running   0          34s   10.233.103.97   worker2   <none>           <none>

remote ❯ kubectl exec -it rs-hostpath-89rmm -- touch /data/hostpath/wow

remote ❯ kubectl exec -it rs-hostpath-89rmm -- ls -al /data/hostpath

total 8
drwxr-xr-x 2 root root 4096 Aug 26 16:19 .
drwxr-xr-x 3 root root 4096 Aug 26 16:11 ..
-rw-r--r-- 1 root root    0 Aug 26 16:19 wow

remote ❯ kubectl exec -it rs-hostpath-xm6s8 -- ls -al /data/hostpath

total 8
drwxr-xr-x 2 root root 4096 Aug 26 16:19 .
drwxr-xr-x 3 root root 4096 Aug 26 16:11 ..
-rw-r--r-- 1 root root    0 Aug 26 16:19 wow

remote ❯ kubectl exec -it rs-hostpath-n9z2t -- ls -al /data/hostpath

total 8
drwxr-xr-x 2 root root 4096 Aug 26 16:11 .
drwxr-xr-x 3 root root 4096 Aug 26 16:11 ..
```

Restart

- 기존 Pod가 삭제되고, 다른 Pod가 생성되었음에도 hostPath 내용이 유지가 되는지를 보기 위한 실습이다. 당연히, 유지가 된다.

```
remote ❯ kubectl get pods -o wide

NAME                READY   STATUS    RESTARTS   AGE   IP              NODE      NOMINATED NODE   READINESS GATES
rs-hostpath-89rmm   1/1     Running   0          34s   10.233.103.98   worker2   <none>           <none>
rs-hostpath-n9z2t   1/1     Running   0          34s   10.233.110.41   worker1   <none>           <none>
rs-hostpath-xm6s8   1/1     Running   0          34s   10.233.103.97   worker2   <none>           <none>


remote ❯ kubectl delete pods rs-hostpath-89rmm

pod "rs-hostpath-89rmm" deleted


remote ❯ kubectl get pods -o wide

NAME                READY   STATUS    RESTARTS   AGE   IP              NODE      NOMINATED NODE   READINESS GATES
rs-hostpath-n9z2t   1/1     Running   0          16m   10.233.110.41   worker1   <none>           <none>
rs-hostpath-rs52v   1/1     Running   0          97s   10.233.103.99   worker2   <none>           <none>
rs-hostpath-xm6s8   1/1     Running   0          16m   10.233.103.97   worker2   <none>           <none>


remote ❯ kubectl exec -it rs-hostpath-rs52v -- ls -al /data/hostpath

total 8
drwxr-xr-x 2 root root 4096 Aug 26 16:19 .
drwxr-xr-x 3 root root 4096 Aug 26 16:26 ..
-rw-r--r-- 1 root root    0 Aug 26 16:19 wow
```

filesystem

- hostPath의 내용은 해당 Node host에서 그대로 확인이 가능하다.

- 다르게 생각해보면, host에서 Pod로 파일을 공유할 수 있는 방법이 될 수도 있다.

```
remote ❯ ssh vagrant@192.168.100.201


worker1 ❯ ls -al /tmp/hostPath-demo

total 8
drwxr-xr-x  2 root root 4096  8월 27 01:11 .
drwxrwxrwt 12 root root 4096  8월 27 01:35 ..


worker1 ❯ exit
```

```
remote ❯ ssh vagrant@192.168.100.202


worker2 ❯ ls -al /tmp/hostpath-demo

total 8
drwxr-xr-x  2 root root 4096  8월 27 01:19 .
drwxrwxrwt 12 root root 4096  8월 27 01:36 ..
-rw-r--r--  1 root root    0  8월 27 01:19 wow


worker2 ❯ exit
```

- 사용자가 임의의 경로를 설정할 수 있기 때문에, hostPath는 보안에 취약하다.

- 또한, 기존 hostPath 내용이 삭제되지 않았으면, 잘못 노출될 수 있는 위험도 있고,

  같은 ReplicaSet에 속한 Pod인데도 위치한 Node에 따라 다른 내용이 저장된다는 문제도 있다.

```
      volumes:
      - name: hostpath-demo
        hostPath:
          path: /home/vagrant
          type: DirectoryOrCreate
```

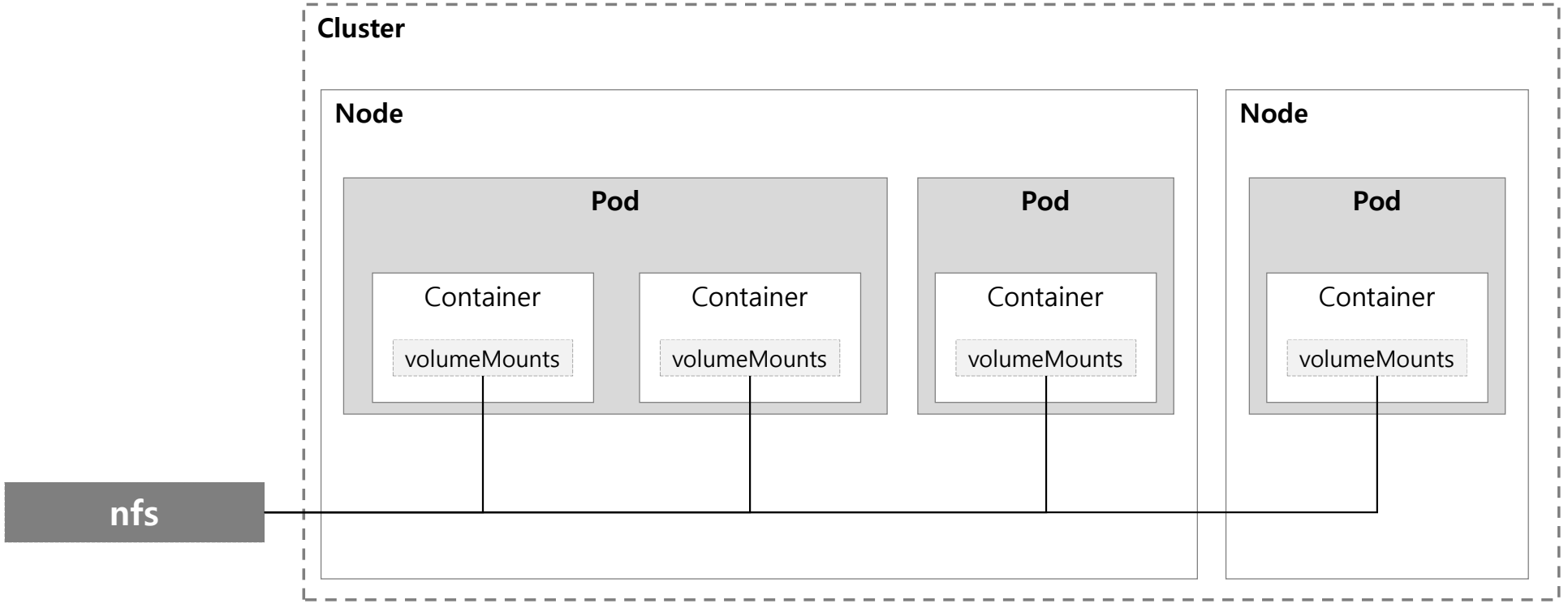
///

Volume - nfs

nfs

- NFS(Network File System)은 네트워크 상에서 파일시스템을 공유하도록 설계된 파일 시스템이다.

- Node와 무관하게 volume을 공유할 수 있으며, Pod/Node 모두 재시작 하더라도 파일을 유지할 수 있다.



NFS Server 구성

- Working 用 Ubuntu PC에 설치하는 것으로 진행해 보겠다. (K8s Cluster 바깥에 NFS Server가 있는 상황)

```
// 기본 패키지 설치
remote ❯ sudo apt install nfs-common nfs-kernel-server portmap


// NFS Server가 사용할 디렉토리 준비 (위치는 개인 취향)
remote ❯ cd /srv
remote ❯ sudo mkdir nfs
remote ❯ sudo chmod 777 ./nfs


// 접근 가능한 호스트 등록
remote ❯ sudo nano /etc/exports

        ...
        /srv/nfs 192.168.100.*(rw,sync,no_root_squash,no_subtree_check)

// NFS Server Service 재시작
remote ❯ sudo service nfs-kernel-server restart


// 확인
remote ❯ showmount -e 127.0.0.1

Export list for 127.0.0.1:
/srv/nfs 192.168.100.*
```

Worker Node Setting for NFS

- NFS를 volume으로 사용하는 Pod가 생성되는 worker node에 설치되어야 하는 패키지가 있다.

```
remote ❯ ssh vagrant@192.168.100.201
worker1 ❯ sudo apt install nfs-common
worker1 ❯ exit


remote ❯ ssh vagrant@192.168.100.202
worker2 ❯ sudo apt install nfs-common
worker2 ❯ exit
```

YAML

- nfs를 mount 하도록 구성한 Pod 3개를 생성

rs-nfs.yaml

```yaml
apiVersion: apps/v1
kind: ReplicaSet

metadata:
  name: rs-nfs

spec:
  replicas: 3

  selector:
    matchLabels:
      app: ubuntu

  template:
    metadata:
      labels:
        app: ubuntu
    spec:
      containers:
      - image: ubuntu:20.04
        name: worker1
        command: ["/bin/sleep", "3650d"]

        volumeMounts:
        - name: nfs-demo
          mountPath: /data/nfs

      volumes:
      - name: nfs-demo
        nfs:
          path: /srv/nfs
          server: 192.168.100.250
```

Create & Check

- 서로 다른 Node에 있는 Pod 모두, nfs가 mount되어 있는 것을 확인할 수 있다

- node가 다르더라도 일단 volume은 모두 mount 되어 있다.

```
remote ❯ cd kubernetes/06-emptyDir-hostPath-PV/hands-on
remote ❯ kubectl create -f rs-nfs.yaml

replicaset.apps/rs-nfs created

remote ❯ kubectl get pods -o wide

NAME           READY   STATUS    RESTARTS   AGE   IP              NODE      NOMINATED NODE   READINESS GATES
rs-nfs-8ktgl   1/1     Running   0          10s   10.233.103.101   worker2   <none>           <none>
rs-nfs-9hzs2   1/1     Running   0          10s   10.233.110.42    worker1   <none>           <none>
rs-nfs-f2vjt   1/1     Running   0          10s   10.233.103.100   worker2   <none>           <none>


remote ❯ kubectl exec -it rs-nfs-9hzs2 -- ls -al /data

total 12
drwxr-xr-x 3 root root 4096 Aug 26 19:02 .
drwxr-xr-x 1 root root 4096 Aug 26 19:02 ..
drwxrwxrwx 2 root root 4096 Aug 26 19:00 nfs

remote ❯ kubectl exec -it rs-nfs-8ktgl -- ls -al /data

total 12
drwxr-xr-x 3 root root 4096 Aug 26 19:02 .
drwxr-xr-x 1 root root 4096 Aug 26 19:02 ..
drwxrwxrwx 2 root root 4096 Aug 26 19:00 nfs
```

Share Volume

- Node가 달라도 같은 volume이 share 된다.

```
remote ❯ kubectl get pods -o wide

NAME          READY   STATUS    RESTARTS   AGE   IP              NODE      NOMINATED NODE   READINESS GATES
rs-nfs-8ktgl   1/1     Running   0          10s   10.233.103.101   worker2   <none>           <none>
rs-nfs-9hzs2   1/1     Running   0          10s   10.233.110.42    worker1   <none>           <none>
rs-nfs-f2vjt   1/1     Running   0          10s   10.233.103.100   worker2   <none>           <none>

remote ❯ kubectl exec -it rs-nfs-8ktgl -- touch /data/nfs/wow
remote ❯ kubectl exec -it rs-nfs-8ktgl -- ls -al /data/nfs

total 8
drwxrwxrwx 2 root root 4096 Aug 26 19:22 .
drwxr-xr-x 3 root root 4096 Aug 26 19:02 ..
-rw-r--r-- 1 root root    0 Aug 26 19:22 wow

remote ❯ kubectl exec -it rs-nfs-f2vjt -- ls -al /data/nfs

total 8
drwxrwxrwx 2 root root 4096 Aug 26 19:22 .
drwxr-xr-x 3 root root 4096 Aug 26 19:02 ..
-rw-r--r-- 1 root root    0 Aug 26 19:22 wow

remote ❯ kubectl exec -it rs-nfs-9hzs2 -- ls -al /data/nfs

total 8
drwxrwxrwx 2 root root 4096 Aug 26 19:22 .
drwxr-xr-x 3 root root 4096 Aug 26 19:02 ..
-rw-r--r-- 1 root root    0 Aug 26 19:22 wow
```

///

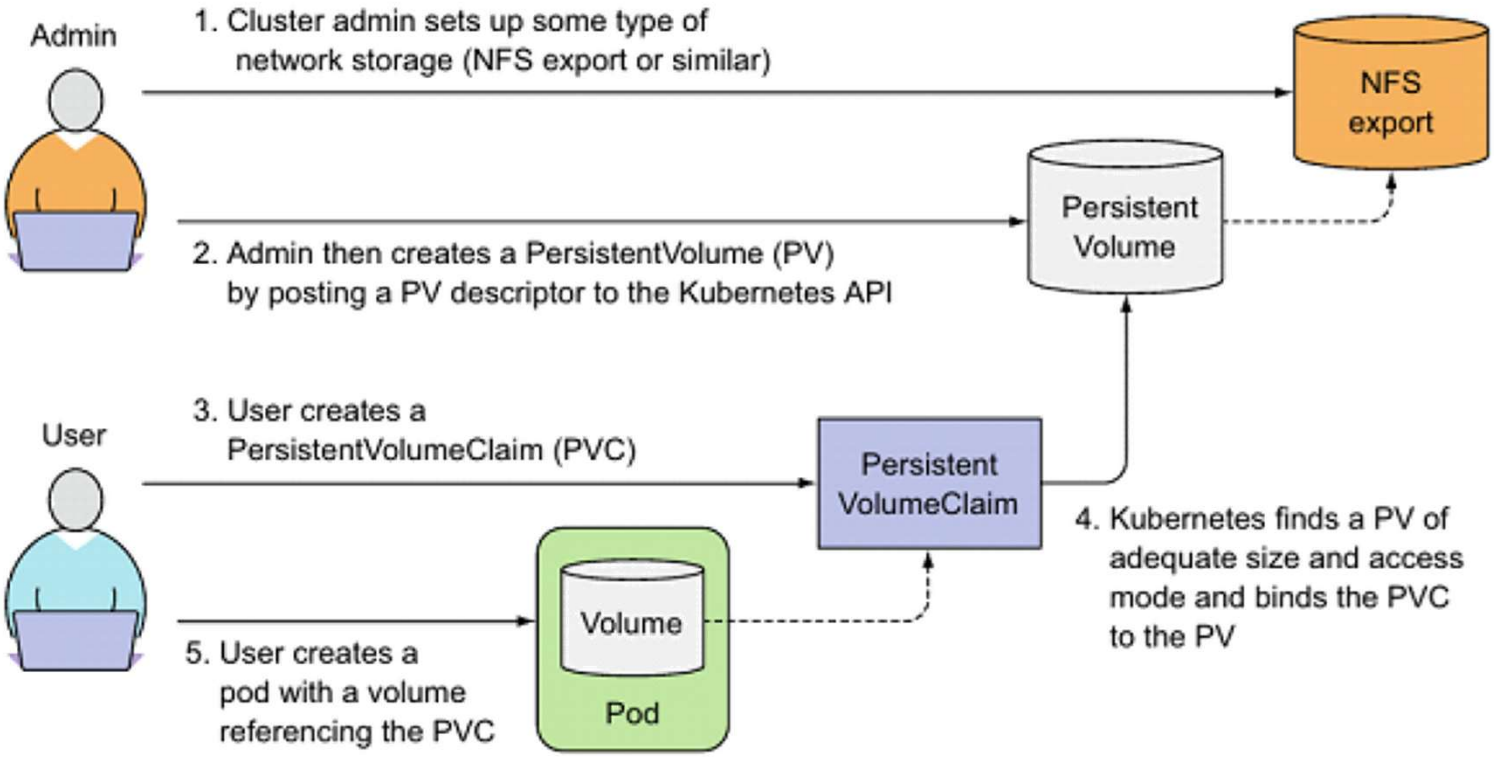
Break

돌아오셨으면 채팅창에
복귀!
타이핑하기!

///

Volume - Static Provisioning (PV / PVC)

PersistentVolume(PV) & PersistentVolumeClaim(PVC)



▲ 그림 6.6 클러스터 관리자가 퍼시스턴트볼륨을 프로비저닝하면 파드는 퍼시스턴트볼륨클레임을 통해 이를 사용한다.

※ 참고 : https://thenewstack.io/strategies-running-stateful-applications-kubernetes-persistent-volumes-claims/

Byte Size

바이트 크기 v • d • e • h					
SI 접두어		전통적 용법		이진 접두어	
기호(이름)	값	기호	값	기호(이름)	V값
kB (킬로바이트)	$1000^1 = 10^3$	KB	$1024^1 = 2^{10}$	KiB (키비바이트)	2^{10}
MB (메가바이트)	$1000^2 = 10^6$	MB	$1024^2 = 2^{20}$	MiB (**메비바이트**)	2^{20}
GB (기가바이트)	$1000^3 = 10^9$	GB	$1024^3 = 2^{30}$	GiB (기비바이트)	2^{30}
TB (테라바이트)	$1000^4 = 10^{12}$	TB	$1024^4 = 2^{40}$	TiB (테비바이트)	2^{40}
PB (페타바이트)	$1000^5 = 10^{15}$	PB	$1024^5 = 2^{50}$	PiB (페비바이트)	2^{50}
EB (엑사바이트)	$1000^6 = 10^{18}$	EB	$1024^6 = 2^{60}$	EiB (엑스비바이트)	2^{60}
ZB (제타바이트)	$1000^7 = 10^{21}$	ZB	$1024^7 = 2^{70}$	ZiB (제비바이트)	2^{70}
YB (요타바이트)	$1000^8 = 10^{24}$	YB	$1024^8 = 2^{80}$	YiB (요비바이트)	2^{80}

※ 참고 : https://ko.wikipedia.org/wiki/메비바이트

PersistentVolume (hostPath) - 1/2

- Volume 사용을 위해서는 물리적인 저장 공간이 필요 → 지금은 가장 기본적인 hostPath를 이용해서 살펴보자

pv-hostPath.yaml

```yaml
apiVersion: v1
kind: PersistentVolume

metadata:
  name: pv-hostpath
  labels:
    type: local

spec:
  storageClassName: manual
  persistentVolumeReclaimPolicy: Retain

  capacity:
    storage: 100Mi

  accessModes:
    - ReadWriteOnce

  hostPath:
    path: "/tmp/pv-data"
    type: DirectoryOrCreate
```

Reclaim Policy (반환 정책)

구분	설명
Retain	수동 반환 (default)
Delete	삭제
Recycle	Deprecated

accessModes

구분		설명
ReadWriteOnce	RWO	하나의 노드에서 볼륨을 읽기-쓰기
ReadOnlyMany	ROX	여러 노드에서 볼륨을 읽기 전용
ReadWriteMany	RWX	여러 노드에서 볼륨을 읽기-쓰기

※ 참고 : https://kubernetes.io/ko/docs/tasks/configure-pod-container/configure-persistent-volume-storage/

PersistentVolume (hostPath) - 2/2

- PersistentVolume을 생성하면 일단 Available 상태가 된다.

```
remote ❯ kubernetes/06-emptyDir-hostPath-PV/hands-on


remote ❯ kubectl create -f pv-hostPath.yaml

persistentvolume/pv-hostpath created


remote ❯ kubectl get persistentvolumes -o wide

NAME          CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM   STORAGECLASS   REASON   AGE   VOLUMEMODE
pv-hostpath   100Mi      RWO            Retain           Available           manual                  46s   Filesystem
```

PersistentVolumeClaim

- PersistentVolume을 요청하는 주문서라고 생각하면 된다.
- 앞에서 준비한 PersistentVolume 중에서 주문 내역에 맞는 것이 사용된다. (50Mi를 요청했을 때 딱 맞는게 없으면, 100Mi가 사용된다)
- 사용된 PersistentVolume의 상태는 Bound가 된다.

pvc-50Mi.yaml

```yaml
apiVersion: v1
kind: PersistentVolumeClaim

metadata:
  name: pvc-50mi

spec:
  storageClassName: manual

  accessModes:
    - ReadWriteOnce

  resources:
    requests:
      storage: 50Mi
```

```
remote ❯ cd kubernetes/06-emptyDir-hostPath-PV/hands-on

remote ❯ kubectl create -f pvc-50Mi.yaml

persistentvolumeclaim/pvc-50mi created


remote ❯ kubectl get persistentvolumeclaims -o wide

NAME       STATUS   VOLUME        CAPACITY   ACCESS MODES   STORAGECLASS   AGE   VOLUMEMODE
pvc-50mi   Bound    pv-hostpath   100Mi      RWO            manual         28s   Filesystem


remote ❯ kubectl get persistentvolumes -o wide

NAME          CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM              STORAGECLASS   REASON   AGE     VOLUMEMODE
pv-hostpath   100Mi      RWO            Retain           Bound    default/pvc-50mi   manual                  7m11s   Filesystem
```

※ 참고 : https://kubernetes.io/ko/docs/tasks/configure-pod-container/configure-persistent-volume-storage/

volumeMounts

rs-static.yaml

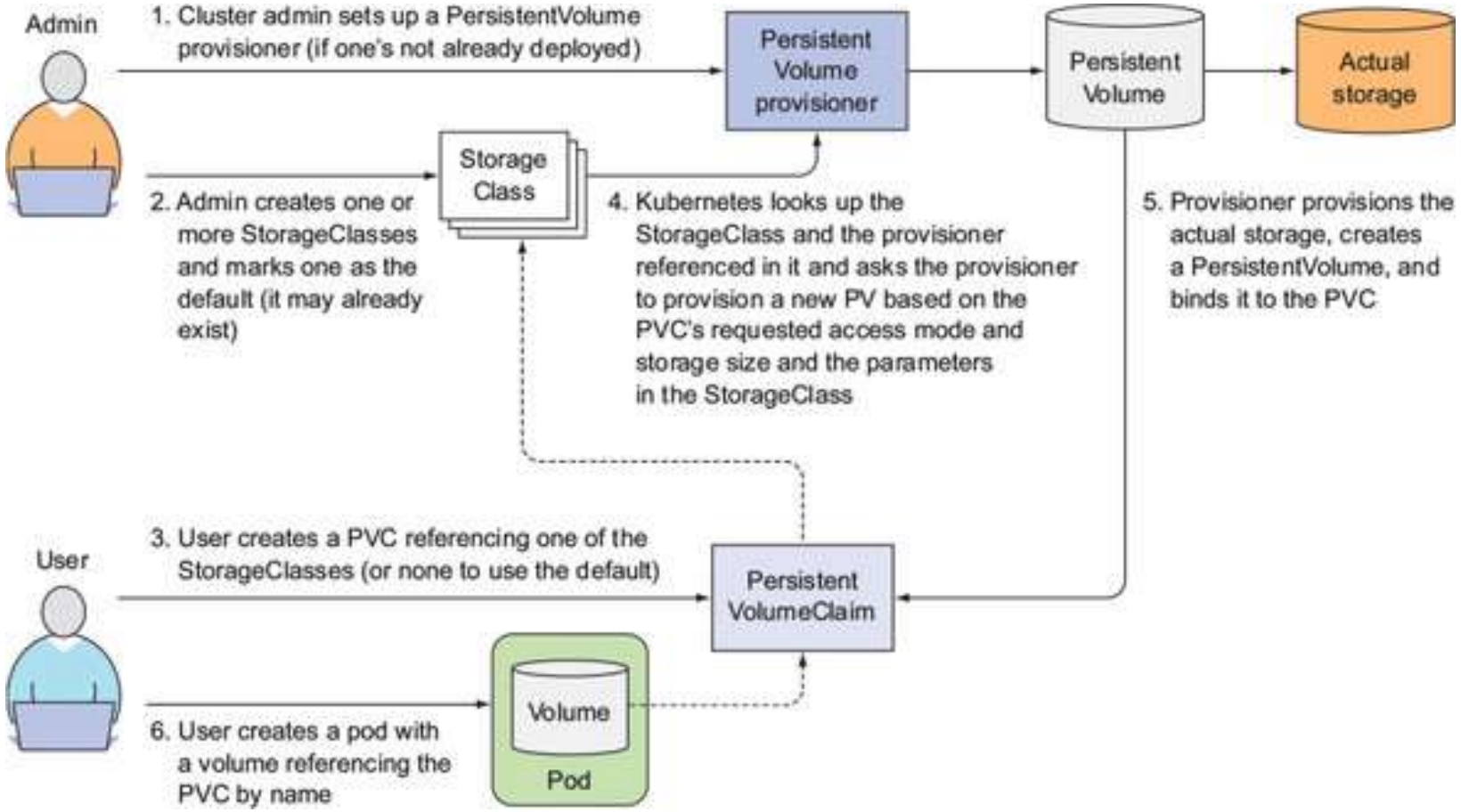
```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: rs-static

spec:
  replicas: 1
  selector:
    matchLabels:
      app: ubuntu

  template:
    metadata:
      labels:
        app: ubuntu

    spec:
      containers:
      - image: ubuntu:20.04
        name: ubuntu
        command: ["/bin/sleep", "3650d"]

        volumeMounts:
        - name: pvc-static
          mountPath: /data/pv

      volumes:
        - name: pvc-static
          persistentVolumeClaim:
            claimName: pvc-50mi
```

```
remote ❯ cd kubernetes/06-emptyDir-hostPath-PV/hands-on

remote ❯ kubectl create -f rs-static.yaml

replicaset.apps/rs-static created

remote ❯ kubectl get pods -o wide

NAME              READY   STATUS    RESTARTS   AGE   IP              NODE      NOMINATED NODE   READINESS GATES
rs-static-9nfj6   1/1     Running   0          48s   10.233.103.102   worker2   <none>           <none>

remote ❯ kubectl get persistentvolumeclaims -o wide

NAME       STATUS   VOLUME        CAPACITY   ACCESS MODES   STORAGECLASS   AGE     VOLUMEMODE
pvc-50mi   Bound    pv-hostpath   100Mi      RWO            manual         4h21m   Filesystem

remote ❯ kubectl describe pods rs-static-9nfj6
...
Volumes:
  pvc-static:
    Type:       PersistentVolumeClaim (a reference to a PersistentVolumeClaim in the same namespace)
    ClaimName:  pvc-50mi
    ReadOnly:   false
  kube-api-access-c9s9j:
    Type:                    Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
...
```

///

Volume - Dynamic Provisioning (StorageClass)

Provisioning of PersistentVolumes



※ 참고 : https://livebook.manning.com/concept/kubernetes/persistentvolumes

YAML

- local storage를 사용하는 경우, provisioner를 no-provisioner로 지정하면 된다

sc-local.yaml

```yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass

metadata:
  name: sc-local

provisioner: kubernetes.io/no-provisioner

reclaimPolicy: Retain

volumeBindingMode: WaitForFirstConsumer
```

pvc-local.yaml

```yaml
apiVersion: v1
kind: PersistentVolumeClaim

metadata:
  name: pvc-local

spec:
  storageClassName: sc-local

  accessModes:
    - ReadWriteOnce

  resources:
    requests:
      storage: 50Mi
```

pv-local.yaml

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-local

spec:
  capacity:
    storage: 100Mi

  accessModes:
  - ReadWriteOnce

  persistentVolumeReclaimPolicy: Retain

  storageClassName: sc-local

  local:
    path: /tmp/pv-local

  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: In
          values:
          - worker1
```

Ready Local Volume & Create StorageClass/PVC/PV

- local 유형의 PersistentVolume을 사용하기 위해서는 실제로 Node에 해당 경로가 존재해야 한다. (지금은 worker1 활용)

- /tmp 디렉토리를 이용하는 이유는 Node가 재시작 하면 자동으로 삭제되기 때문에 편리해서... (유지하려면 다른 경로 이용 필요)

```
remote ❯ ssh vagrant@192.168.100.201

worker1 ❯ mkdir -p /tmp/pv-local
worker1 ❯ chmod 777 /tmp/pv-local
```

```
remote ❯ cd kubernetes/06-emptyDir-hostPath-PV/hands-on

remote ❯ kubectl create -f sc-local.yaml
remote ❯ kubectl create -f pvc-local.yaml
remote ❯ kubectl create -f pv-local.yaml

remote ❯ kubectl get storageclasses -o wide

NAME       PROVISIONER                    RECLAIMPOLICY   VOLUMEBINDINGMODE      ALLOWVOLUMEEXPANSION   AGE
sc-local   kubernetes.io/no-provisioner   Retain          WaitForFirstConsumer   false                  23s

remote ❯ kubectl get persistentvolumeclaims -o wide

NAME        STATUS    VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS   AGE   VOLUMEMODE
pvc-local   Pending                                      sc-local       66s   Filesystem

remote ❯ kubectl get persistentvolumes -o wide

NAME       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM   STORAGECLASS   REASON   AGE     VOLUMEMODE
pv-local   100Mi      RWO            Retain           Available           sc-local               2m44s   Filesystem
```

volumeMounts

rs-dynamic.yaml

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: rs-dynamic

spec:
  replicas: 1
  selector:
    matchLabels:
      app: ubuntu

  template:
    metadata:
      labels:
        app: ubuntu

    spec:
      containers:
      - image: ubuntu:20.04
        name: ubuntu
        command: ["/bin/sleep", "3650d"]

        volumeMounts:
        - name: pv-claim
          mountPath: /data/pv

      volumes:
        - name: pv-claim
          persistentVolumeClaim:
            claimName: pvc-local
```

```
remote ❯ kubectl create -f rs-dynamic.yaml

remote ❯ kubectl get persistentvolumeclaims -o wide

NAME        STATUS   VOLUME     CAPACITY   ACCESS MODES   STORAGECLASS   AGE   VOLUMEMODE
pvc-local   Bound    pv-local   100Mi      RWO            sc-local       10m   Filesystem

remote ❯ kubectl get persistentvolumes -o wide

NAME       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM               STORAGECLASS   REASON   AGE   VOLUMEMODE
pv-local   100Mi      RWO            Retain           Bound    default/pvc-local   sc-local                11m   Filesystem

remote ❯ kubectl get pods -o wide

NAME              READY   STATUS    RESTARTS   AGE     IP              NODE      NOMINATED NODE   READINESS GATES
rs-dynamic-ntckh   1/1     Running   0          2m17s   10.233.110.43   worker1   <none>           <none>

remote ❯ kubectl exec -it rs-dynamic-ntckh -- touch /data/pv/wow

remote ❯ ssh vagrant@192.168.100.201

worker1 ❯ ls -al /tmp/pv-local

total 8
drwxrwxrwx  2 vagrant vagrant 4096  8월 27 09:49 .
drwxrwxrwt 13 root    root    4096  8월 27 09:50 ..
-rw-r--r--  1 root    root       0  8월 27 09:49 wow
```

///

Tip

k9s

- Kubernetes CLI To Manage Your Clusters In Style!

- LinuxBrew(HomeBrew) Install

```
remote ❯ sudo apt-get install build-essential curl file git

remote ❯ sh -c "$(curl -fsSL https://raw.githubusercontent.com/Linuxbrew/install/master/install.sh)"
```

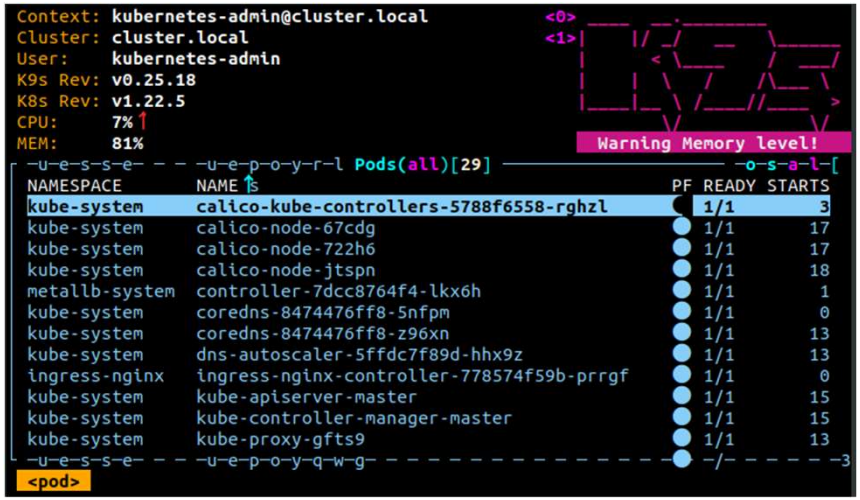
```
remote ❯ nano ~/.zshrc

...

remote ❯ source ~/.zshrc
```

~/.zshrc

```
export PATH="/home/linuxbrew/.linuxbrew/bin:$PATH"
export MANPATH="/home/linuxbrew/.linuxbrew/share/man:$MANPATH"
export INFOPATH="/home/linuxbrew/.linuxbrew/share/info:$INFOPATH"
```

- k9s Install

```
remote ❯ brew install derailed/k9s/k9s

remote ❯ k9s
```



※ 참고 : https://k9scli.io/

///