5th Week

다섯번째 뵙겠습니다 ?!

▷ 출석 체크도 한 번 해보시면 어떠세요?!

- https://modulabs.co.kr/
- 모두연 홈페이지 → 로그인 → 마이페이지 → 참여한 랩·풀잎 → 자세히 보기 → 내 풀잎스쿨 출석 확인하기

▷ Ground Rule

- 가급적 지각/결석 하지 않기
- 가급적 Camera 켜 놓고 수업 참여하기
- 가급적 적극적으로 참여하기
- 3시간이 넘더라도 배고프다고 화내지 않기
- Slack 읽지 않기
- 꼭 끝까지 함께하기

잡담 &
지난 수업 관련 이야기

///

Agenda

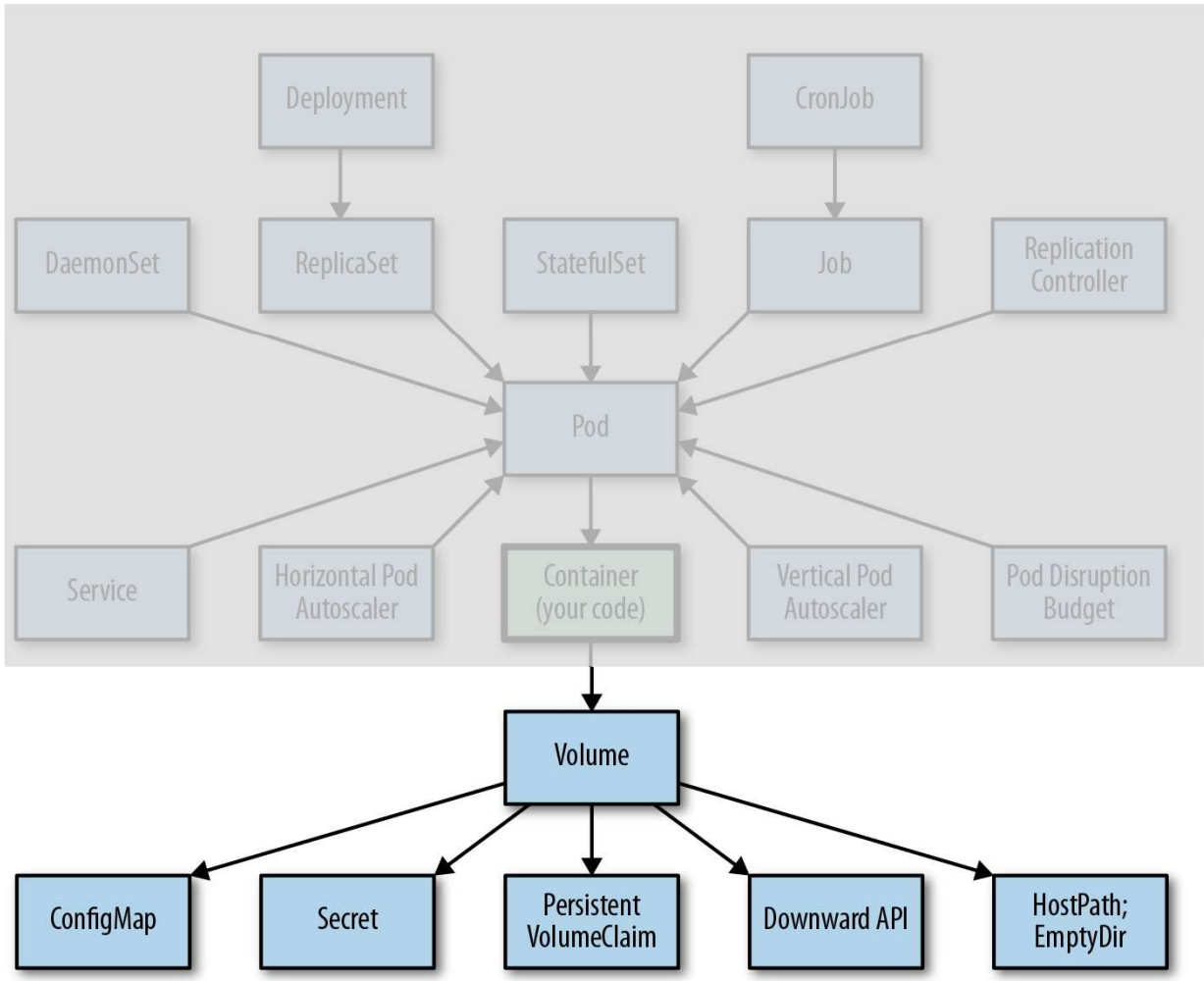
- ▷ [10m] make-friendship
- ▷ [30m] Flip - Volume (emptyDir/hostPath/PersistentVolume)
- ▷ [30m] Wrap-Up & Hands-On
- ▷ [10m] Break-Time

- ▷ [30m] Flip - Volume (configMap/Secret/downwardAPI)
- ▷ [30m] Wrap-Up & Hands-On
- ▷ [20m] REST API

- ▷ [20m] Quiz

Kubernetes

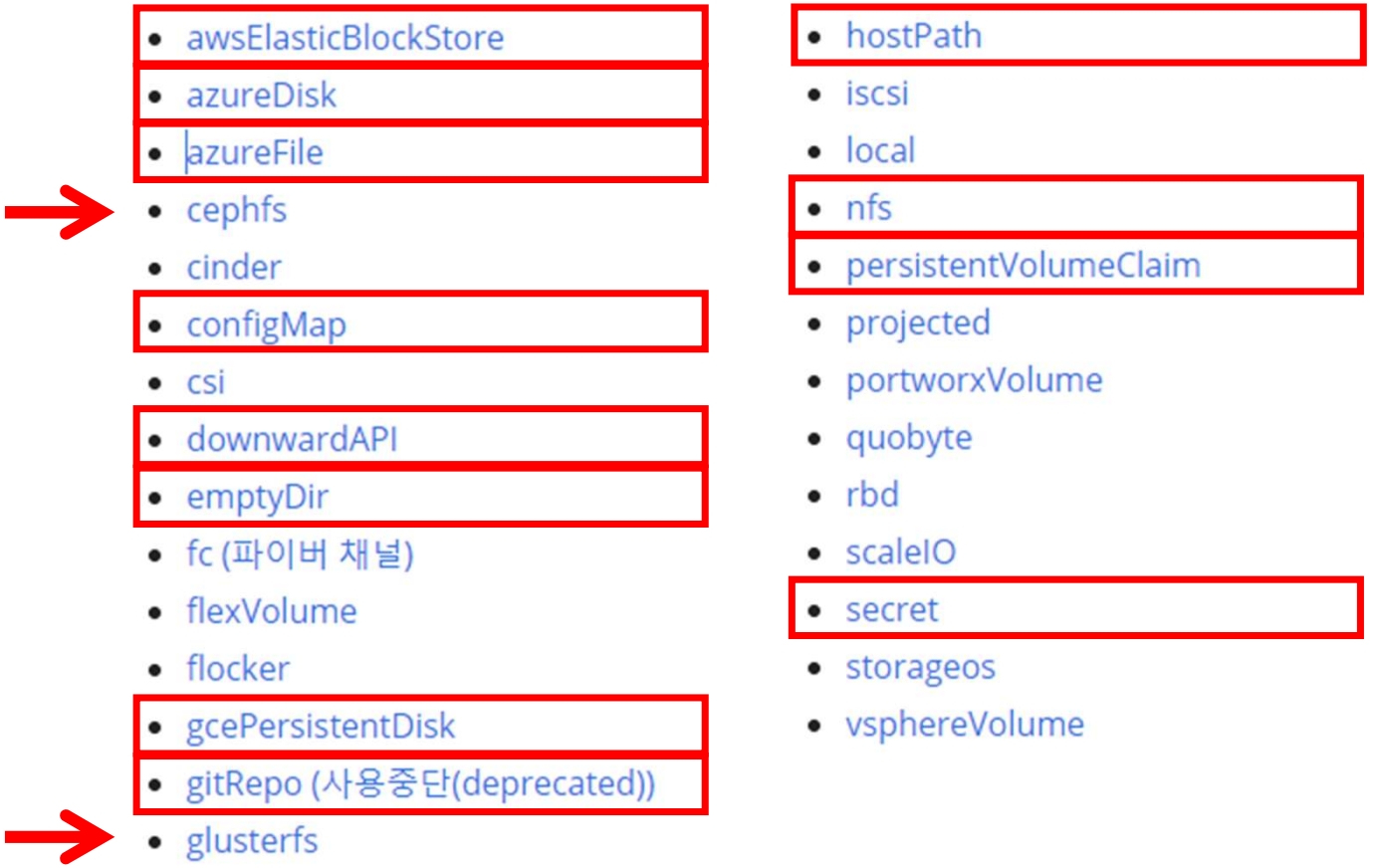
Volume - Overview


Deployment
CronJob
DaemonSet
ReplicaSet
StatefulSet
Job
Replication Controller
Pod
Service
Horizontal Pod Autoscaler
Container (your code)
Vertical Pod Autoscaler
Pod Disruption Budget
Volume
ConfigMap
Secret
Persistent VolumeClaim
Downward API
HostPath; EmptyDir

kinds of volume

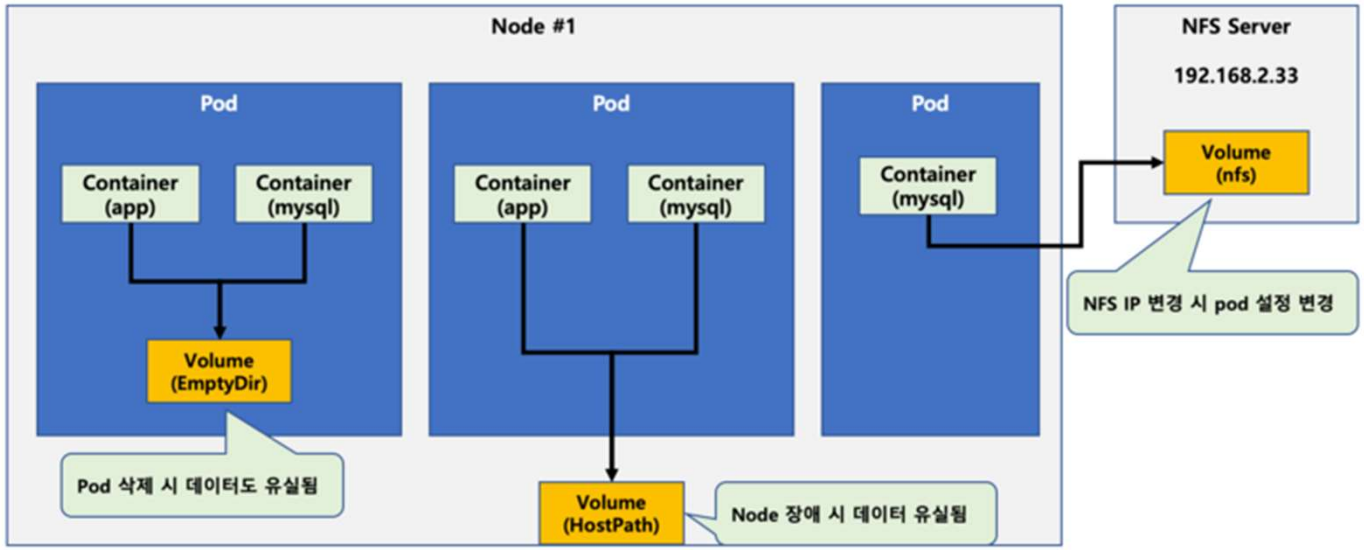
- "Kubernetes In Action"책에서 설명하는 범위

On-Premise 환경에서
별도 구축할 수 있는 solution



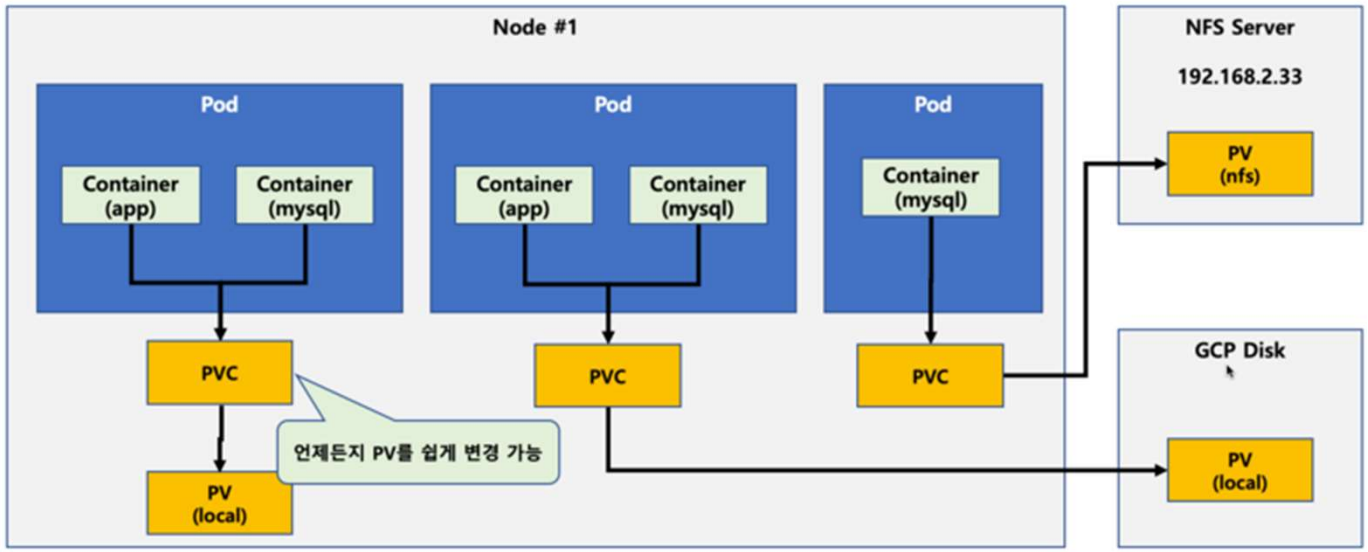
※ 참고 : https://kubernetes.io/ko/docs/concepts/storage/volumes/

Pod에서 직접 Volume을 지정하는 방식



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

Pod에서 Persistent Volume을 활용하는 방식



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

///

Flip Learning

(Volume - emptyDir/hostPath/PV)

이원준님

///

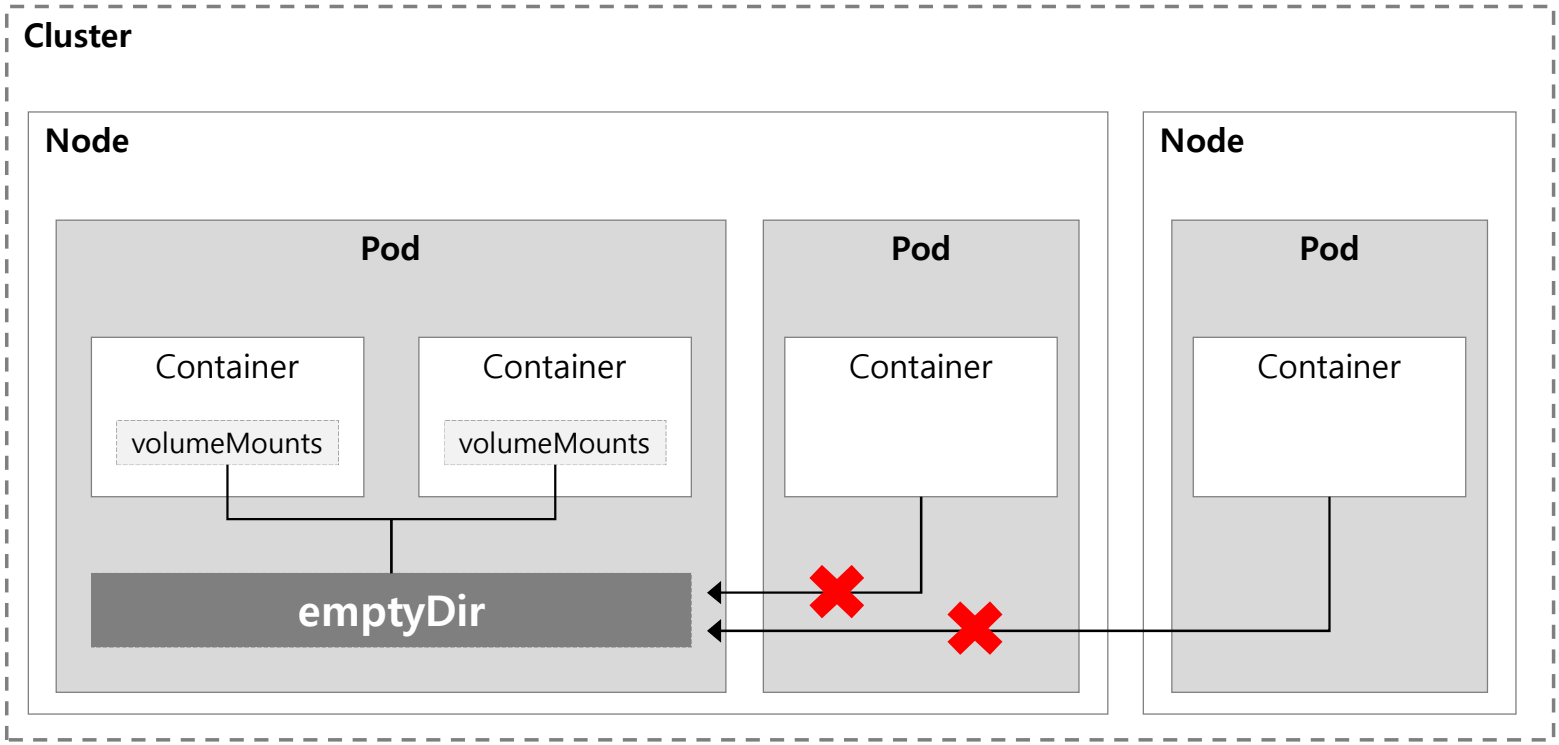
Kubernetes

Volume - emptyDir

emptyDir

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

- Pod가 재시작 하는 경우에는 저장된 파일 유실



Hands-On - 1/4

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: {}
```

1개의 Pod 안에 2개의 container를 구성하고

각 container에서 emptyDir을 mount 하도록 구성

Hands-On - 2/4

```
remote ❯ git clone https://github.com/whatwant-school/advanced-kubernetes.git
remote ❯ cd advanced-kubernetes

remote ❯ kubectl create -f ./05-week/emptyDir/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-wgfwx   2/2     Running   0          22s   10.233.103.40   worker2   <none>           <none>


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

total 12
drwxr-xr-x 3 root root 4096 Feb  2 15:44 .
drwxr-xr-x 1 root root 4096 Feb  2 15:44 ..
drwxrwxrwx 2 root root 4096 Feb  2 15:44 emptydir1


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

total 12
drwxr-xr-x 3 root root 4096 Feb  2 15:44 .
drwxr-xr-x 1 root root 4096 Feb  2 15:44 ..
drwxrwxrwx 2 root root 4096 Feb  2 15:44 emptydir2
```

생성된 각 container에

mount된 directory를 확인

Hands-On - 3/4

```
remote ❯ kubectl exec -it rs-emptydir-wgfwx -c worker1 -- ls -al /data/emptydir1

total 8
drwxrwxrwx 2 root root 4096 Feb  2 14:44 .
drwxr-xr-x 3 root root 4096 Feb  2 14:44 ..


remote ❯ kubectl exec -it rs-emptydir-wgfwx -c worker2 -- ls -al /data/emptydir2

total 8
drwxrwxrwx 2 root root 4096 Feb  2 14:44 .
drwxr-xr-x 3 root root 4096 Feb  2 14:44 ..


remote ❯ kubectl exec -it rs-emptydir-wgfwx -c worker1 -- touch /data/emptydir1/wow


remote ❯ kubectl exec -it rs-emptydir-wgfwx -c worker1 -- ls -al /data/emptydir1

total 8
drwxrwxrwx 2 root root 4096 Feb  2 15:21 .
drwxr-xr-x 3 root root 4096 Feb  2 14:44 ..
-rw-r--r-- 1 root root    0 Feb  2 15:21 wow


remote ❯ kubectl exec -it rs-emptydir-wgfwx -c worker2 -- ls -al /data/emptydir2

total 8
drwxrwxrwx 2 root root 4096 Feb  2 15:21 .
drwxr-xr-x 3 root root 4096 Feb  2 14:44 ..
-rw-r--r-- 1 root root    0 Feb  2 15:21 wow
```

container 1번에서 생성한 파일이

container 2번에서도 공유되고 있음을 확인

Hands-On - 4/4

```
remote ❯ kubectl get pods -o wide

NAME                READY   STATUS    RESTARTS   AGE   IP              NODE      NOMINATED NODE   READINESS GATES
rs-emptydir-wgfwx   2/2     Running   0          10m   10.233.103.40   worker2   <none>           <none>


remote ❯ kubectl exec -it rs-emptydir-wgfwx -c worker1 -- ls -al /data/emptydir1

total 8
drwxrwxrwx 2 root root 4096 Feb  2 15:53 .
drwxr-xr-x 3 root root 4096 Feb  2 15:44 ..
-rw-r--r-- 1 root root    0 Feb  2 15:53 wow


remote ❯ kubectl delete pods rs-emptydir-wgfwx

pod "rs-emptydir-wgfwx" deleted


remote ❯ kubectl get pods -o wide

NAME                READY   STATUS    RESTARTS   AGE   IP              NODE      NOMINATED NODE   READINESS GATES
rs-emptydir-5s8l8   2/2     Running   0          61s   10.233.103.41   worker2   <none>           <none>


remote ❯ kubectl exec -it rs-emptydir-5s8l8 -c worker1 -- ls -al /data/emptydir1

total 8
drwxrwxrwx 2 root root 4096 Feb  2 15:55 .
drwxr-xr-x 3 root root 4096 Feb  2 15:55 ..
```

Pod가 재시작 되었을 때,

emptyDir 내용이 유지가 되는지를 보기 위한 실습이다.

당연히, 유지가 안된다.

///

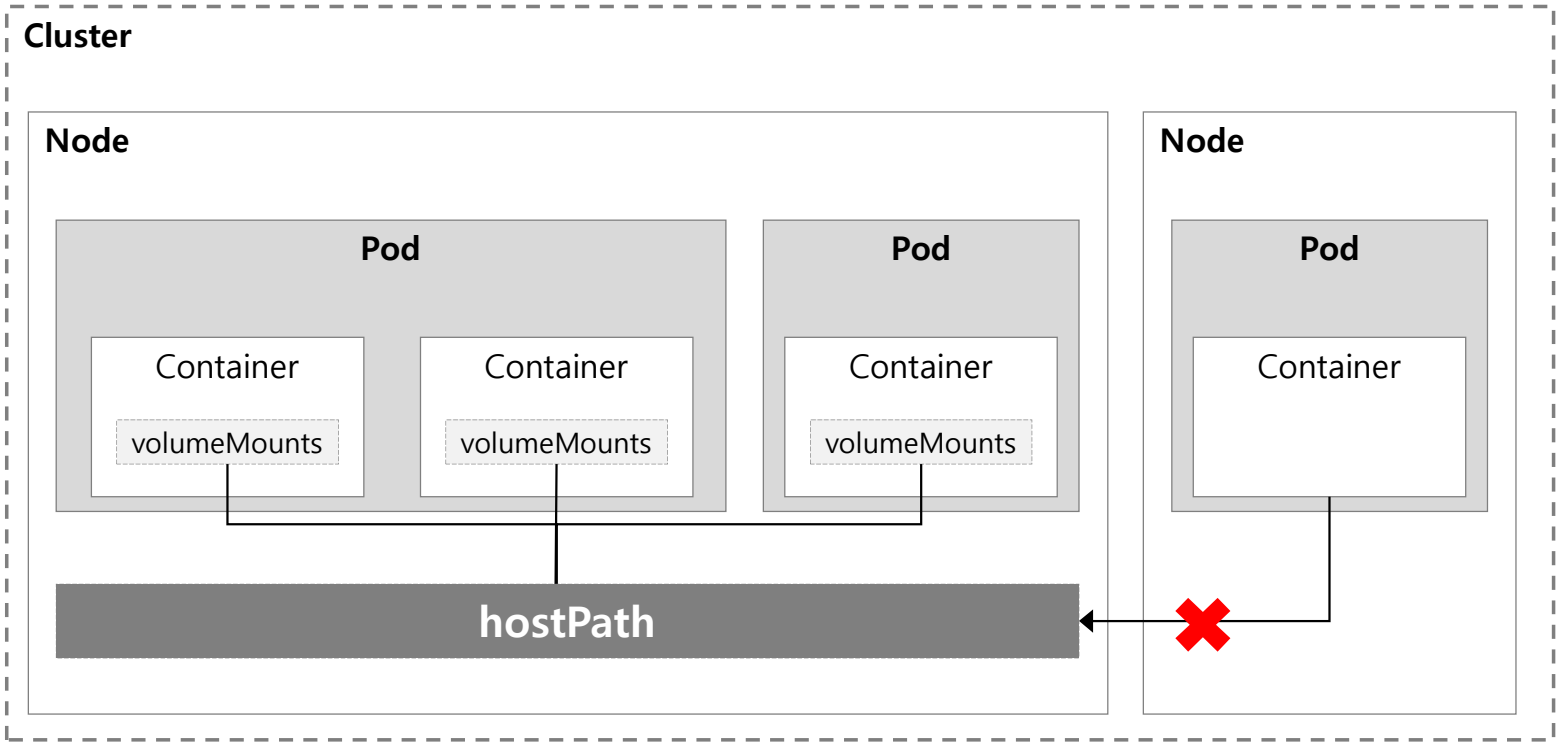
Kubernetes

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/

Hands-On - 1/5

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
```

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

hostPath을 mount 하도록 구성한 Pod 3개를 생성

Hands-On - 2/5

```
remote ❯ git clone https://github.com/whatwant-school/advanced-kubernetes.git
remote ❯ cd advanced-kubernetes

remote ❯ kubectl create -f ./05-week/hostPath/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-2xpmw  1/1     Running   0          13s   10.233.110.57   worker1   <none>           <none>
rs-hostpath-nlgdh  1/1     Running   0          13s   10.233.103.42   worker2   <none>           <none>
rs-hostpath-pddz5  1/1     Running   0          13s   10.233.103.43   worker2   <none>           <none>


remote ❯ kubectl exec -it rs-hostpath-2xpmw -- ls -al /data

total 12
drwxr-xr-x 3 root root 4096 Feb  2 16:45 .
drwxr-xr-x 1 root root 4096 Feb  2 16:45 ..
drwxr-xr-x 2 root root 4096 Feb  2 16:45 hostpath


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

total 12
drwxr-xr-x 3 root root 4096 Feb  2 16:45 .
drwxr-xr-x 1 root root 4096 Feb  2 16:45 ..
drwxr-xr-x 2 root root 4096 Feb  2 16:45 hostpath
```

서로 다른 Node에 있는 Pod 모두

일단 hostPath가 mount되어 있는 것을 확인할 수 있다

Hands-On - 3/5

```
remote ❯ kubectl get pods -o wide

NAME                READY   STATUS    RESTARTS   AGE   IP              NODE      NOMINATED NODE   READINESS GATES
rs-hostpath-2xpmw   1/1     Running   0          13s   10.233.110.57   worker1   <none>           <none>
rs-hostpath-nlgdh   1/1     Running   0          13s   10.233.103.42   worker2   <none>           <none>
rs-hostpath-pddz5   1/1     Running   0          13s   10.233.103.43   worker2   <none>           <none>

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

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

total 8
drwxr-xr-x 2 root root 4096 Feb  2 16:57 .
drwxr-xr-x 3 root root 4096 Feb  2 16:45 ..
-rw-r--r-- 1 root root    0 Feb  2 16:57 wow


remote ❯ kubectl exec -it rs-hostpath-2xpmw -- ls -al /data/hostpath

total 8
drwxr-xr-x 2 root root 4096 Feb  2 16:57 .
drwxr-xr-x 3 root root 4096 Feb  2 16:45 ..


remote ❯ kubectl exec -it rs-hostpath-pddz5 -- ls -al /data/hostpath

total 8
drwxr-xr-x 2 root root 4096 Feb  2 16:57 .
drwxr-xr-x 3 root root 4096 Feb  2 16:45 ..
-rw-r--r-- 1 root root    0 Feb  2 16:57 wow
```

Worker2에 있는 Pod에서 파일을 하나 생성했고,

실제로 확인해봐도 잘 생성되어 있다.

Worker1에 있는 Pod에서는

파일이 안보이지만

Worker2에 있는 다른 Pod에서는

파일이 잘 보인다.

Hands-On - 4/5

```
remote ❯ kubectl get pods -o wide

NAME                READY   STATUS    RESTARTS   AGE   IP              NODE      NOMINATED NODE   READINESS GATES
rs-hostpath-2xpmw   1/1     Running   0          30m   10.233.110.57   worker1   <none>           <none>
rs-hostpath-nlgdh   1/1     Running   0          30m   10.233.103.42   worker2   <none>           <none>
rs-hostpath-pddz5   1/1     Running   0          30m   10.233.103.43   worker2   <none>           <none>

remote ❯ kubectl exec -it rs-hostpath-pddz5 -- ls -al /data/hostpath

total 8
drwxr-xr-x 2 root root 4096 Feb  2 16:57 .
drwxr-xr-x 3 root root 4096 Feb  2 16:45 ..
-rw-r--r-- 1 root root    0 Feb  2 16:57 wow

remote ❯ kubectl delete pods rs-hostpath-pddz5

pod "rs-hostpath-pddz5" deleted

remote ❯ kubectl get pods -o wide

NAME                READY   STATUS    RESTARTS   AGE   IP              NODE      NOMINATED NODE   READINESS GATES
rs-hostpath-2xpmw   1/1     Running   0          31m   10.233.110.57   worker1   <none>           <none>
rs-hostpath-5v9r8   1/1     Running   0          42s   10.233.103.44   worker2   <none>           <none>
rs-hostpath-nlgdh   1/1     Running   0          31m   10.233.103.42   worker2   <none>           <none>

remote ❯ kubectl exec -it rs-hostpath-5v9r8 -- ls -al /data/hostpath

total 8
drwxr-xr-x 2 root root 4096 Feb  2 16:57 .
drwxr-xr-x 3 root root 4096 Feb  2 17:16 ..
-rw-r--r-- 1 root root    0 Feb  2 16:57 wow
```

기존 Pod가 삭제되고, 다른 Pod가 생성되었음에도

hostPath 내용이 유지가 되는지를 보기 위한 실습이다.

당연히, 유지가 된다.

Hands-On - 5/5

```
remote ❯ ssh vagrant@192.168.100.201


worker1 ❯ ls -al /tmp/hostPath-demo

total 8
drwxr-xr-x  2 root root 4096  2월  3 01:57 .
drwxrwxrwt 12 root root 4096  2월  3 02:25 ..


worker1 ❯ exit


remote ❯ ssh vagrant@192.168.100.202


worker2 ❯ ls -al /tmp/hostpath-demo

total 8
drwxr-xr-x  2 root root 4096  2월  3 01:57 .
drwxrwxrwt 12 root root 4096  2월  3 02:26 ..
-rw-r--r--  1 root root    0  2월  3 01:57 wow
```

Worker1 Node에서도 hostPath로 지정한 디렉토리가 생성이 되었다.

하지만, 파일 내용은 없다.

Worker2 Node에서도 hostPath로 지정한 디렉토리가 생성이 되었다.

Worker2 Node에 있는 Pod에서 파일을 생성했기에

여기에서는 파일이 보인다.

hostPath의 내용은 해당 Node host에서 그대로 확인이 가능하다.

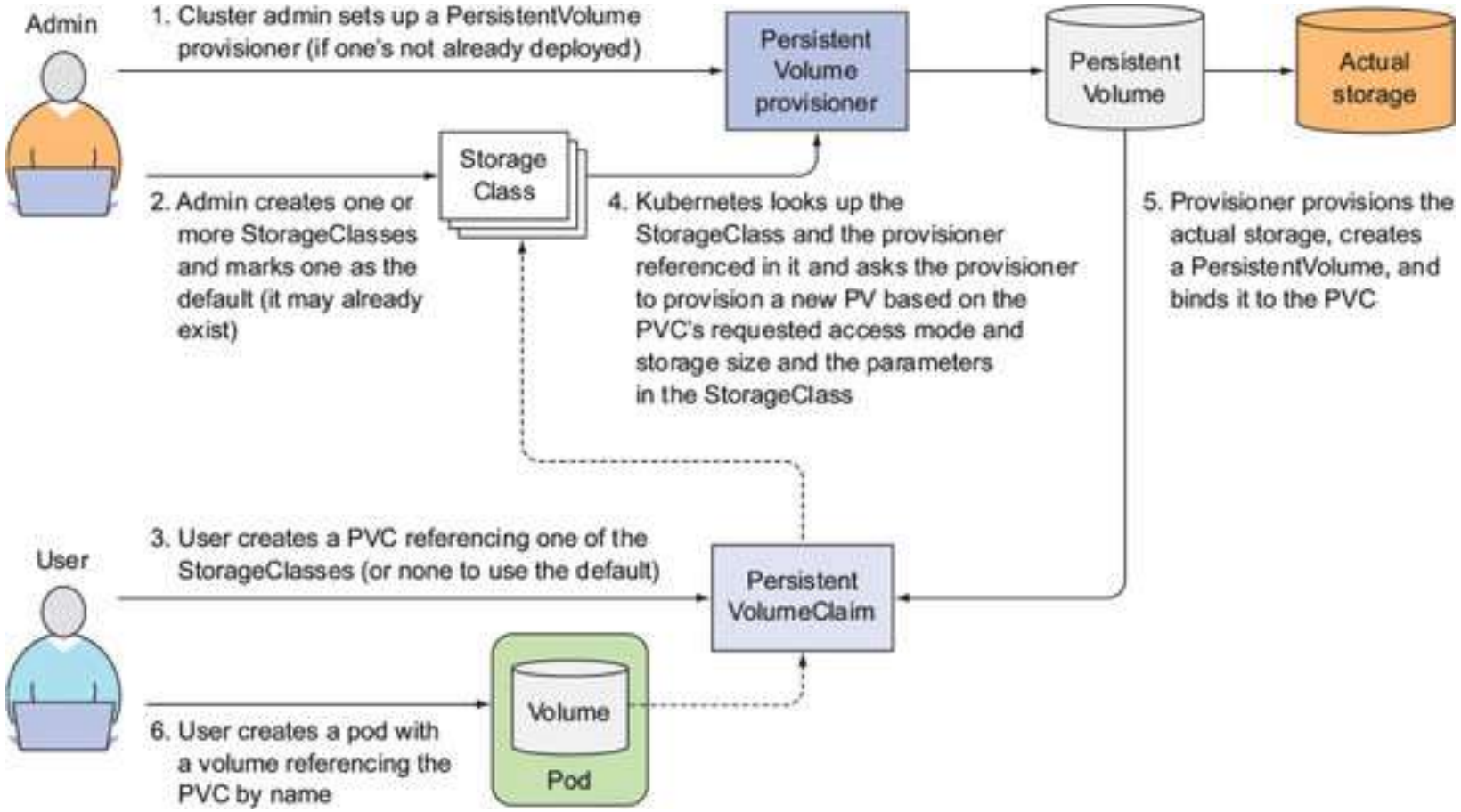
다르게 생각해보면, host에서 Pod로 파일을 공유할 수 있는 방법이 될 수도 있다.

반면, 쓸모 없어진 hostPath 내용이 자동으로 삭제되지도 않고, 이름이 중복되어 잘못 노출될 수 있는 위험도 있고,

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

///

Provisioning of PersistentVolumes



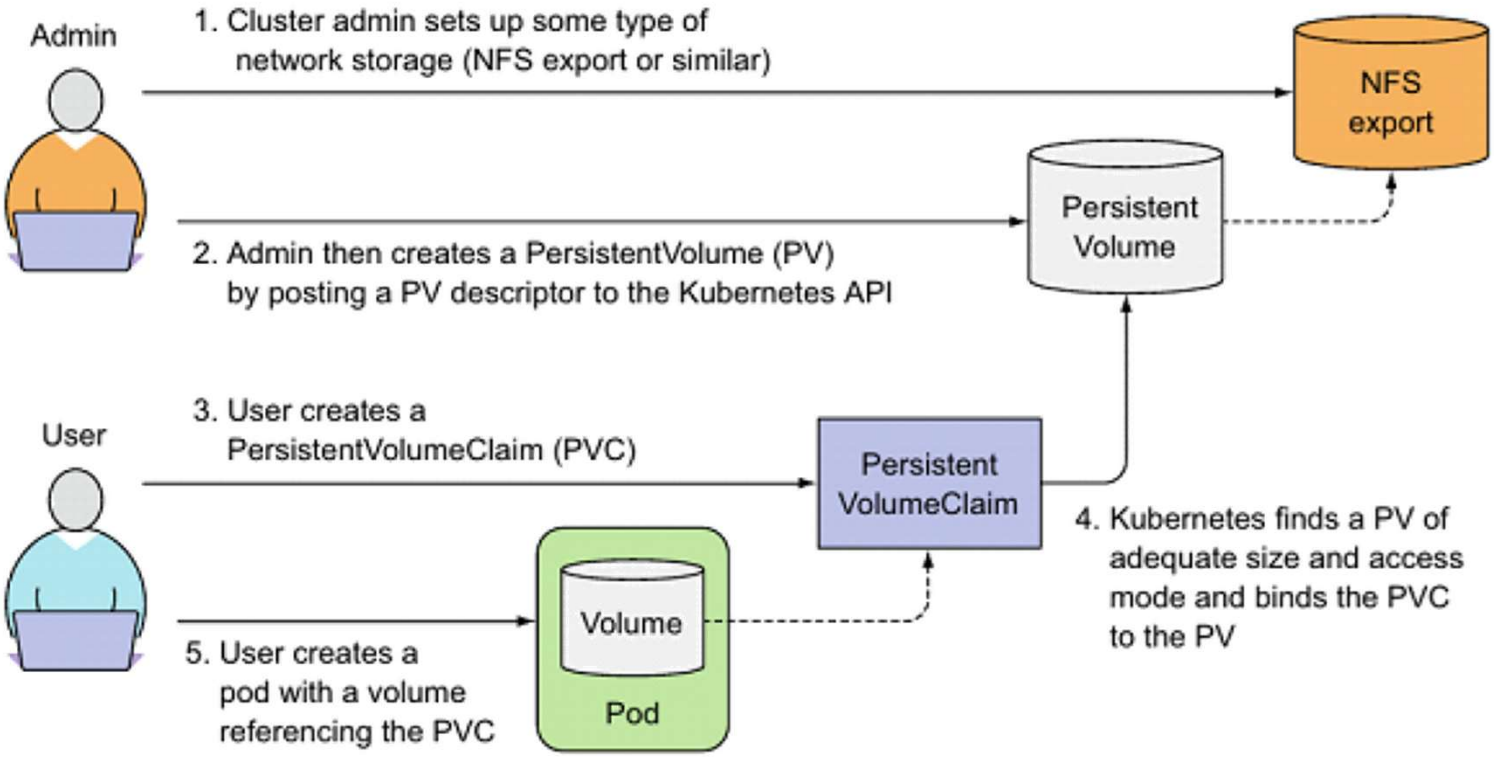
※ 참고 : https://livebook.manning.com/concept/kubernetes/persistentvolumes

///

Kubernetes

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/

PersistentVolume (hostPath)

- 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
```

```
remote ❯ git clone https://github.com/whatwant-school/advanced-kubernetes.git
remote ❯ cd advanced-kubernetes

remote ❯ kubectl create -f ./05-week/persistentvolume/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                  13s   Filesystem
```

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/

PersistentVolumeClaim

- PersistentVolume을 요청하는 주문서라고 생각하면 된다

pvc-50mi.yaml

```yaml
apiVersion: v1
kind: PersistentVolumeClaim

metadata:
  name: pvc-50mi

spec:
  storageClassName: manual

  accessModes:
    - ReadWriteOnce

  resources:
    requests:
      storage: 50Mi
```

```
remote ❯ git clone https://github.com/whatwant-school/advanced-kubernetes.git
remote ❯ cd advanced-kubernetes

remote ❯ kubectl create -f ./05-week/persistentvolume/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         6s    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                  6h15m   Filesystem
```

50Mi 용량으로 주문을 했는데, 주문 내용과 맞는 PV를 찾아보니 100Mi 용량짜리가 있어서 그렇게 matching 된 것이다.

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

volumeMounts

rs-pvc-static.yaml

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: rs-pvc-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: pv-claim
          mountPath: /data/pv

      volumes:
        - name: pv-claim
          persistentVolumeClaim:
            claimName: pvc-50mi
```

```
remote ❯ git clone https://github.com/whatwant-school/advanced-kubernetes.git
remote ❯ cd advanced-kubernetes

remote ❯ kubectl create -f ./05-week/persistentvolume/rs-pvc-static.yaml

replicaset.apps/rs-pvc-static created

remote ❯ kubectl get replicasets -o wide

NAME            DESIRED   CURRENT   READY   AGE   CONTAINERS   IMAGES          SELECTOR
rs-pvc-static   1         1         1       15s   ubuntu       ubuntu:20.04    app=ubuntu

remote ❯ kubectl get pods -o wide

NAME                  READY   STATUS    RESTARTS   AGE   IP              NODE      NOMINATED NODE   READINESS GATES
rs-pvc-static-7xv6m   1/1     Running   0          23s   10.233.103.47   worker2   <none>           <none>

remote ❯ kubectl exec -it rs-pvc-static-7xv6m -- touch /data/pv/wow

remote ❯ ssh whatwant@192.168.100.202

worker2 ❯ ls -al /tmp/pv-data

total 8
drwxr-xr-x  2 root root 4096  2월  6 21:50 .
drwxrwxrwt 13 root root 4096  2월  6 21:50 ..
-rw-r--r--  1 root root    0  2월  6 21:50 wow
```

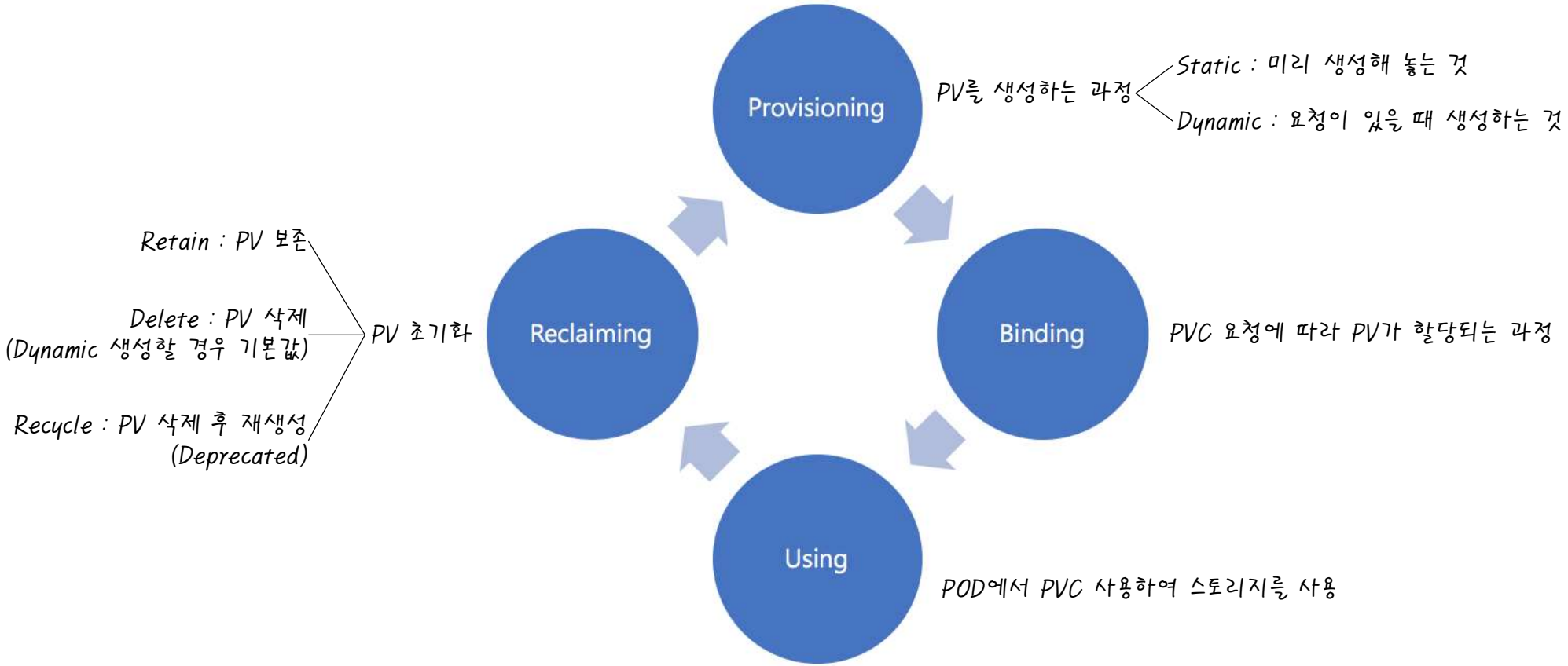
claimName 지정만으로 PV와 연결이 된다.

///

Kubernetes

Volume - Dynamic Provisioning (StorageClass)

“PV & PVC” Life Cycle (생명주기)



※ 참고 : https://arisu1000.tistory.com/27849

StorageClass (local) & Ready Storage

- 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
```

```
remote ❯ git clone https://github.com/whatwant-school/advanced-kubernetes.git
remote ❯ cd advanced-kubernetes

remote ❯ kubectl create -f ./05-week/persistentvolume/sc-local.yaml

storageclass.storage.k8s.io/sc-local created


remote ❯ kubectl get storageclasses -o wide

NAME       PROVISIONER                    RECLAIMPOLICY   VOLUMEBINDINGMODE      ALLOWVOLUMEEXPANSION   AGE
sc-local   kubernetes.io/no-provisioner   Retain          WaitForFirstConsumer   false                  32s
```

- 실제 파일 저장을 할 물리적인 공간을 확보해야 한다. 여기에서는 woker1 Node에 준비를 해봤다.

```
remote ❯ ssh whatwant@192.168.100.201

worker1 ❯ mkdir -p /tmp/pv-local

worker1 ❯ chmod 777 /tmp/pv-local
```

자꾸 /tmp 디렉토리를 이용하는 이유는

해당 Node가 재시작 하면 자동으로 해당 내용들을 삭제해주기 때문이다.

만약, Node가 재시작 되더라도 유지되게 하고 싶다면, 다른 디렉토리를 이용해야 한다.

PersistentVolume (local)

- 지금 만들어야 하는 PV는 앞에서 선언한 StorageClass와 미리 확보해 놓은 저장 공간을 사용하도록 해야 한다

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
```

```
remote ❯ git clone https://github.com/whatwant-school/advanced-kubernetes.git
remote ❯ cd advanced-kubernetes

remote ❯ kubectl create -f ./05-week/persistentvolume/pv-local.yaml

persistentvolume/pv-local created


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               8s    Filesystem


remote ❯ kubectl get storageclasses -o wide

NAME       PROVISIONER                    RECLAIMPOLICY   VOLUMEBINDINGMODE      ALLOWVOLUMEEXPANSION   AGE
sc-local   kubernetes.io/no-provisioner   Retain          WaitForFirstConsumer   false                  31m
```

PV는 아직 생성되지 않았다.

앞에서 미리 만들어 놓은 저장 공간이

worker1 Node의 `/tmp/pv-local`이었으므로

`nodeAffinity`를 이용해서 생성 Node를 제한한다.

PersistentVolumeClaim

- Static Provisioning과는 분명한 차이가 있다

pvc-local.yaml

```yaml
apiVersion: v1
kind: PersistentVolumeClaim

metadata:
  name: pvc-local

spec:
  storageClassName: sc-local

  accessModes:
    - ReadWriteOnce

  resources:
    requests:
      storage: 50Mi
```

```
remote ❯ git clone https://github.com/whatwant-school/advanced-kubernetes.git
remote ❯ cd advanced-kubernetes

remote ❯ kubectl create -f ./05-week/persistentvolume/pvc-local.yaml

persistentvolumeclaim/pvc-local created


remote ❯ kubectl get persistentvolumeclaims -o wide

NAME        STATUS    VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS   AGE   VOLUMEMODE
pvc-local   Pending                                      sc-local       17s   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               11m   Filesystem
```

PVC를 생성했음에도 PV와 Binding 되지 않았다.

PV는 아직도 Available 상태이고, PVC는 Pending 상태이다.

volumeMounts

rs-pvc-dynamic.yaml

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: rs-pvc-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 ./05-week/persistentvolume/rs-pvc-dynamic.yaml

replicaset.apps/rs-pvc-dynamic created

remote ❯ kubectl get persistentvolumeclaims -o wide

NAME        STATUS   VOLUME     CAPACITY   ACCESS MODES   STORAGECLASS   AGE   VOLUMEMODE
pvc-local   Bound    pv-local   100Mi      RWO            sc-local       12m   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                23m   Filesystem

remote ❯ kubectl get pods -o wide

NAME                   READY   STATUS    RESTARTS   AGE   IP              NODE      NOMINATED NODE   READINESS GATES
rs-pvc-dynamic-htrvf   1/1     Running   0          97s   10.233.110.63   worker1   <none>           <none>

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

remote ❯ ssh whatwant@192.168.100.201

worker1 ❯ ls -al /tmp/pv-local

total 8
drwxrwxrwx  2 whatwant whatwant 4096  2월  6 23:45 .
drwxrwxrwt 13 root     root     4096  2월  6 23:48 ..
-rw-r--r--  1 root     root        0  2월  6 23:45 wow
```

Pod가 생성되어서 volume을 요청하는 시점에

PVC ~ PV Binding이 이루어진다.

///

Break

///

Flip Learning

(Volume - ConfigMap/Secret/downwardAPI)

김남형님

///

Kubernetes

Environment Variables & Arguments

Environment Variables

- 환경 변수를 Container에 선언하기 위해서는 `env:`를 사용하면 된다.

pod-env.yaml

```yaml
apiVersion: v1
kind: Pod

metadata:
  name: pod-env

spec:
  containers:
  - name: print-env
    image: bash

    env:
    - name: FIRST
      value: "One"
    - name: SECOND
      value: "Two"
    - name: THIRD
      value: "Three"

    command: ["echo"]
    args: ["$(FIRST), $(SECOND), $(THIRD)"]
```

```
remote ❯ git clone https://github.com/whatwant-school/advanced-kubernetes.git
remote ❯ cd advanced-kubernetes

remote ❯ kubectl create -f ./05-week/env/pod-env.yaml

pod/pod-env created


remote ❯ kubectl get pods -o wide

NAME      READY   STATUS      RESTARTS   AGE   IP              NODE      NOMINATED NODE   READINESS
GATES
pod-env   0/1     Completed   0          3s    10.233.103.33   worker2   <none>           <none>


remote ❯ kubectl logs pod-env

One, Two, Three
```

Dockerfile에서 환경 변수를 정의할 수 있지만,

그렇게 되면 환경 변수 값이 변경될 때 이미지를 변경해야 하기에 추천하지 않는다.

※ 참고 : https://kubernetes.io/ko/docs/tasks/inject-data-application/define-environment-variable-container/

Arguments in Docker

- Container에서 arguments를 어떻게 사용하는지 먼저 살펴보고, Image를 만들어 보자.

fortuneloop.sh

```
#!/bin/bash
trap "exit" SIGINT
INTERVAL=$1
echo generate every $INTERVAL seconds
mkdir -p /var/htdocs

while :
do
  echo $(date) Writing to /var/htdocs/index.html
  /usr/games/fortune > /var/htdocs/index.html
  sleep $INTERVAL
done
```

Dockerfile

```
FROM ubuntu:latest

RUN apt-get update
RUN apt-get -y install fortune

ADD fortuneloop.sh /bin/fortuneloop.sh
RUN chmod +x /bin/fortuneloop.sh

ENTRYPOINT ["/bin/fortuneloop.sh"]
CMD ["10"]
```

```
remote ❯ git clone https://github.com/whatwant-school/advanced-kubernetes.git
remote ❯ cd advanced-kubernetes/05-week/arguments

remote ❯ docker build -t fortune:v1.0 .

...

remote ❯ docker run -it --name fortune fortune:v1.0

generate every 10 seconds
Sat Feb 12 18:28:10 UTC 2022 Writing to /var/htdocs/index.html
Sat Feb 12 18:28:20 UTC 2022 Writing to /var/htdocs/index.html
Sat Feb 12 18:28:30 UTC 2022 Writing to /var/htdocs/index.html
^C


( DockerHub에서 fortune repository 생성 )

remote ❯ docker login

remote ❯ docker tag fortune:v1.0 whatwant/fortune:v1.0

remote ❯ docker push whatwant/fortune:v1.0
```

10초 간격으로 동작하는 것을 볼 수 있다.

※ 참고 : https://github.com/luksa/kubernetes-in-action/tree/master/Chapter07

Arguments in Kubernetes - 1/2

- 이번에는 Kubernetes에서 arguments를 어떻게 사용하는지 살펴보자.

pod-fortune.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod-fortune
  labels:
    app: fortune
spec:
  containers:
  - name: html-generator
    image: whatwant/fortune:v1.0
    args: ["5"]
    volumeMounts:
    - name: web-fortune
      mountPath: /var/htdocs
  - name: web-server
    image: nginx:alpine
    volumeMounts:
    - name: web-fortune
      mountPath: /usr/share/nginx/html
      readOnly: true
    ports:
    - containerPort: 80
  volumes:
  - name: web-fortune
    emptyDir: {}
```

```
remote ❯ git clone https://github.com/whatwant-school/advanced-kubernetes.git
remote ❯ cd advanced-kubernetes

remote ❯ kubectl create -f ./05-week/arguments/pod-fortune.yaml

pod/pod-fortune created


remote ❯ kubectl logs pod-fortune -c html-generator

generate every 5 seconds
Sat Feb 12 19:31:43 UTC 2022 Writing to /var/htdocs/index.html
Sat Feb 12 19:31:48 UTC 2022 Writing to /var/htdocs/index.html
Sat Feb 12 19:31:53 UTC 2022 Writing to /var/htdocs/index.html
```

5초 간격으로 변경된 것을 확인할 수 있다.

설명	도커 필드 이름	쿠버네티스 필드 이름
컨테이너에서 실행되는 커맨드	Entrypoint	command
커맨드에 전달되는 인자들	Cmd	arg

command 값과 args 값을 동시에 정의한다면, Docker 이미지 안에 정의된 기본 EntryPoint 값과 기본 Cmd 값이 덮어 쓰여진다.

※ 참고 : https://kubernetes.io/ko/docs/tasks/inject-data-application/define-command-argument-container/

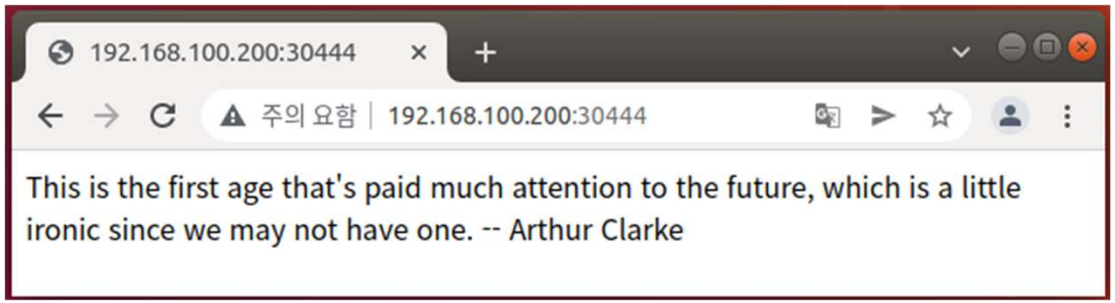
Arguments in Kubernetes - 2/2

- 이왕 만들어진 App이니, Service까지 해서 어떤 결과로 나오는지 확인해보자.

svc-fortune.yaml

```yaml
apiVersion: v1
kind: Service

metadata:
  name: svc-fortune

spec:
  type: NodePort

  ports:
  - port: 80
    targetPort: 80
    nodePort: 30444

  selector:
    app: fortune
```

```
remote ❯ git clone https://github.com/whatwant-school/advanced-kubernetes.git
remote ❯ cd advanced-kubernetes

remote ❯ kubectl create -f ./05-week/arguments/svc-fortune.yaml

service/svc-fortune created


remote ❯ kubectl get services -o wide

NAME          TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)        AGE   SELECTOR
kubernetes    ClusterIP   10.233.0.1     <none>        443/TCP        22d   <none>
svc-fortune   NodePort    10.233.0.134   <none>        80:30444/TCP   17h   app=fortune
```



///

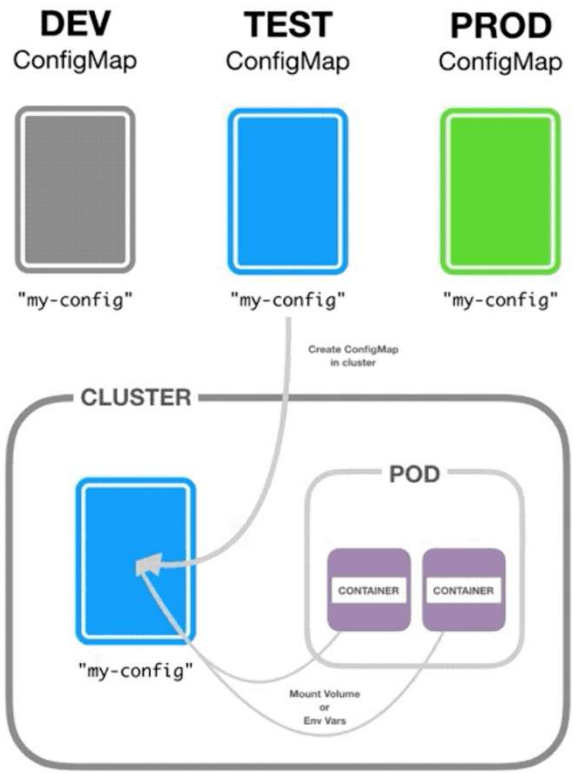
Kubernetes

Volume - configMap

Why configMap ?

- 독립적인 리소스로 동일한 Pod라도 여러 개의 configMap을 선택하여 사용 가능

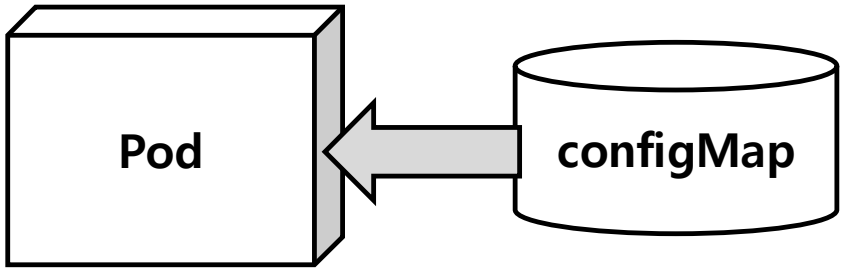
  . configMap만 변경하여 Pod를 손쉽게 테스트



※ 참고 : https://timewizhan.tistory.com/entry/Kubernetes-ConfigMap

How configMap ?

- configMap을 생성해 놓으면, Pod에서 가져다 사용하는 방식



- **env**
- **envFrom**
- **volumes**

configMap을 **설정**하는 방법

configMap을 **생성**하는 방법

- **from directory :** --from-file=/dir
- **from file :** --from-file=/dir/file
- **from literal :** --from-literal=key=value

Hands-On : YAML - 1/2

- YAML을 이용한 configMap 사용법을 알아보자

envFrom & volumes 의 2가지 방식으로
configMap을 사용하는 방법을
살펴보고자 한다.

cm-envfrom.yaml

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: cm-envfrom

data:
  map-hash-bucket-size: "128"
  ssl-protocols: SSLv2
```

cm-volume.yaml

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: cm-volume

data:
  index.html: |-
     <html>
     <h1>Hello from ConfigMap</h1>
     </html>
```

pod-configmap.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod-configmap
  labels:
    app: nginx

spec:
  containers:
  - name: nginx
    image: nginx:1.14.2
    ports:
    - containerPort: 80

    envFrom:
      - configMapRef:
          name: cm-envFrom

    volumeMounts:
    - name: nginx-volume
      subPath: index.html
      mountPath: /usr/share/nginx/html/index.html
```

```yaml
  volumes:
  - name: nginx-volume
    configMap:
      name: cm-volume
```

svc-configmap.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  name: svc-configmap

spec:
  type: NodePort
  ports:
  - port: 80
    targetPort: 80
    nodePort: 30501

  selector:
    app: nginx
```

※ 참고 : https://github.com/tncad/k8s-app-cpd/blob/a41111ae84c45fecb5c21ba96fec0ee113beff81/021-kustomize/full/deployment.yaml

Hands-On : YAML - 2/2

```
remote ❯ git clone https://github.com/whatwant-school/advanced-kubernetes.git
remote ❯ cd advanced-kubernetes

remote ❯ kubectl create -f ./05-week/configMap/cm-envfrom.yaml
remote ❯ kubectl create -f ./05-week/configMap/cm-volume.yaml
remote ❯ kubectl create -f ./05-week/configMap/pod-configmap.yaml
remote ❯ kubectl create -f ./05-week/configMap/svc-configmap.yaml

remote ❯ kubectl get pods -o wide

NAME            READY   STATUS    RESTARTS   AGE    IP              NODE      NOMINATED NODE   READINESS GATES
pod-configmap   1/1     Running   0          4m5s   10.233.103.35   worker2   <none>           <none>


remote ❯ kubectl exec -it pod-configmap -- printenv

PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=pod-configmap
NGINX_VERSION=1.14.2-1~stretch
NJS_VERSION=1.14.2.0.2.6-1~stretch
map-hash-bucket-size=128
ssl-protocols=SSLv2
...


remote ❯ kubectl get services -o wide

NAME            TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)        AGE    SELECTOR
kubernetes      ClusterIP   10.233.0.1    <none>        443/TCP        22d    <none>
svc-configmap   NodePort    10.233.37.4   <none>        80:30501/TCP   6m20s  app=nginx
```

Hands-On : CLI

```
remote ❯ git clone https://github.com/whatwant-school/advanced-kubernetes.git
remote ❯ cd advanced-kubernetes/05-week/configMap

remote ❯ kubectl get configmaps

NAME              DATA   AGE
kube-root-ca.crt   1      22d

remote ❯ cat ./cm-env/map-hash-bucket-size

128

remote ❯ cat ./cm-env/ssl-protocols

SSLv2

remote ❯ cat ./cm-vol/index.html

<html>
<h1>Hello from ConfigMap</h1>
</html>

remote ❯ kubectl create configmap cm-envfrom --from-file=./cm-env/

configmap/cm-envfrom created

remote ❯ kubectl create configmap cm-volume --from-file=./cm-vol/index.html

configmap/cm-volume created
```

앞에서 작업한

모든 것들 지워진 상태

```
remote ❯ kubectl get configmaps

NAME               DATA   AGE
cm-envfrom         2      4m18s
cm-volume          1      7s
kube-root-ca.crt   1      22d


remote ❯ kubectl create -f ./pod-configmap.yaml

pod/pod-configmap created


remote ❯ kubectl create -f ./svc-configmap.yaml

service/svc-configmap created


remote ❯ kubectl exec -it pod-configmap -- printenv

...
map-hash-bucket-size=128
ssl-protocols=SSLv2
...
```

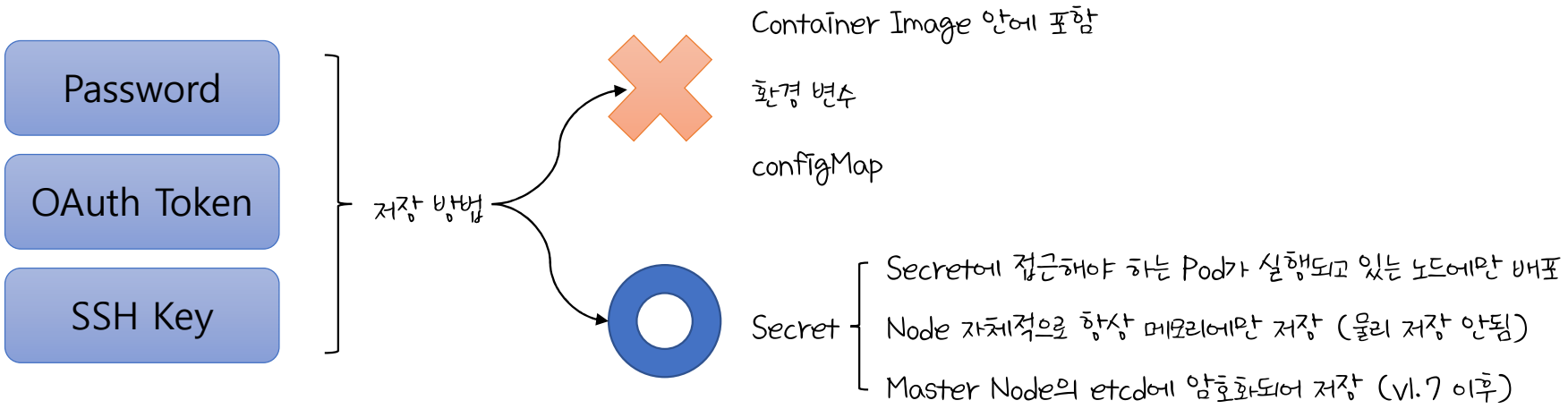
///

Kubernetes

Volume - Secret

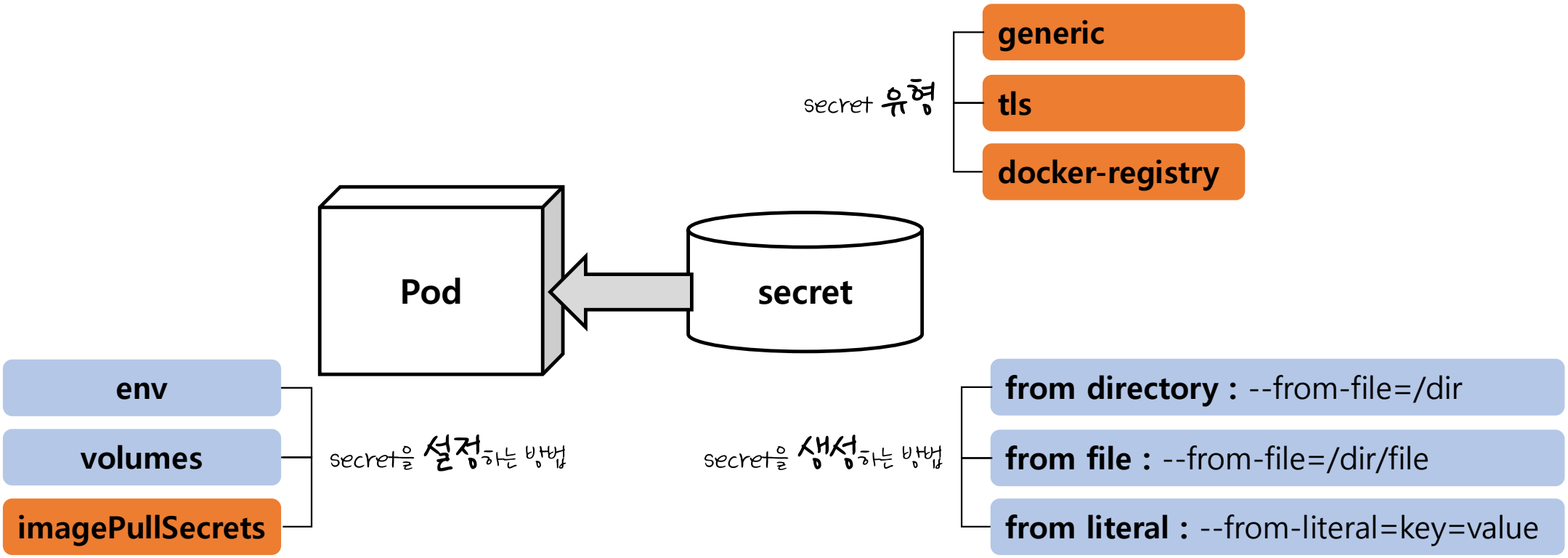
Why secret ?

- 노출되면 안되는 정보를 저장하기 위한 방법 필요



How secret ?

- configMap 사용법과 유사



default token

- 기본적으로 생성되어 있는 secret을 확인해보자

```
remote ❯ kubectl get secrets -o wide

NAME                  TYPE                                  DATA   AGE
default-token-xf884   kubernetes.io/service-account-token   3      23d

remote ❯ kubectl describe secrets default-token-xf884

Name:         default-token-xf884
Namespace:    default
Labels:       <none>
Annotations:  kubernetes.io/service-account.name: default
              kubernetes.io/service-account.uid: e72b51f8-e9f6-44b4-abf1-4f38867c35ee

Type:  kubernetes.io/service-account-token

Data
====
ca.crt:     1099 bytes
namespace:  7 bytes
token:
eyJhbGciOiJSUzI1NiIsImtpZCI6IjJHUXBHX0NzeUl0a0xVSDh5Nk9CRVNleVBKcmxWdThRTVFubWR4d0pBc0UifQ.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJkZWZhdWx0Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZWNyZXQubmFtZSI6ImRlZmF1bHQtdG9rZW4teGY4ODQiLCJrdWJlcm5ldGVzLmlvL3NlcnZpY2VhY2NvdW50L3NlcnZpY2UtYWNjb3VudC5uYW1lIjoiZGVmYXVsdCIsImt1YmVybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VydmljZS1hY2NvdW50LnVpZCI6ImU3MmI1MWY4LWU5ZjYtNDRiNC1hYmYxLTRmMzg4NjdjMzVlZSIsInN1YiI6InN5c3RlbTpzZXJ2aWNlYWNjb3VudDpkZWZhdWx0OmRlZmF1bHQifQ.lmbn_Ucr-gKqrxkU4zYQgqxeABzFac4ml21HPz4GexHlAHknGPg8--pnlihCQq32zrTmRgL-Y6CYd6_Wc4XTEu_fv14Yqp71wiN9k4IvAlqxY-X591-0Vr_LMIyG8S13-DZnwKrwciWKhu-SG557X1U29hjP9gbsQryBoI69RgODONgQNdpxPeKQZwHdf5uhOF8vpd5rDzE2tK_dHvTz8zpofKKf9g1cDqMnJBOONag2Rm2xSwDvKdRsWUYdzuhGgrx3iYxdOqUQuN064u-jE4pb4KLMXJyR42lTX8iw5xOecTav1CXYlJ_PBFqUqGmmHPbD2nLDV7oTvS4qgZA7lA
```

3개의 secret 데이터 존재

Kubernetes API 통신을 위한 3종 데이터 정보

default token in Pod

- Pod를 생성하면 기본적으로 API Server와 통신하기 위한 secret이 mount 된다.

```
remote ❯ kubectl run web --image=nginx:latest

pod/web created

remote ❯ kubectl describe pods web

Name:         web
Namespace:    default
Priority:     0
Node:         worker2/192.168.100.202
Start Time:   Tue, 15 Feb 2022 00:11:11 +0900
Labels:       run=web
Annotations:  cni.projectcalico.org/containerID: 6...
              cni.projectcalico.org/podIP: 10.233....
              cni.projectcalico.org/podIPs: 10.23....
Status:       Running
IP:           10.233.103.39
IPs:
  IP:  10.233.103.39
Containers:
  web:
    Container ID:   containerd://48f16fc7f85e54703...
    Image:          nginx:latest
    Image ID:       docker.io/library/nginx@sha256...
    Port:           <none>
    Host Port:      <none>
...
```

CLI 활용한 Pod 생성

```
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-ggr4n (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  kube-api-access-ggr4n:
    Type:                    Projected (a volume that contains injected data from multiple ...)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
QoS Class:                   BestEffort
Node-Selectors:              <none>
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                             node.kubernetes.io/unreachable:NoExecute op=Exists for 300s

Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  17s   default-scheduler  Successfully assigned default/web to worker2
  Normal  Pulling    17s   kubelet            Pulling image "nginx:latest"
  Normal  Pulled     7s    kubelet            Successfully pulled image "nginx:latest" in 9.1...
  Normal  Created    7s    kubelet            Created container web
  Normal  Started    7s    kubelet            Started container web
```

///

docker registry : Private Image

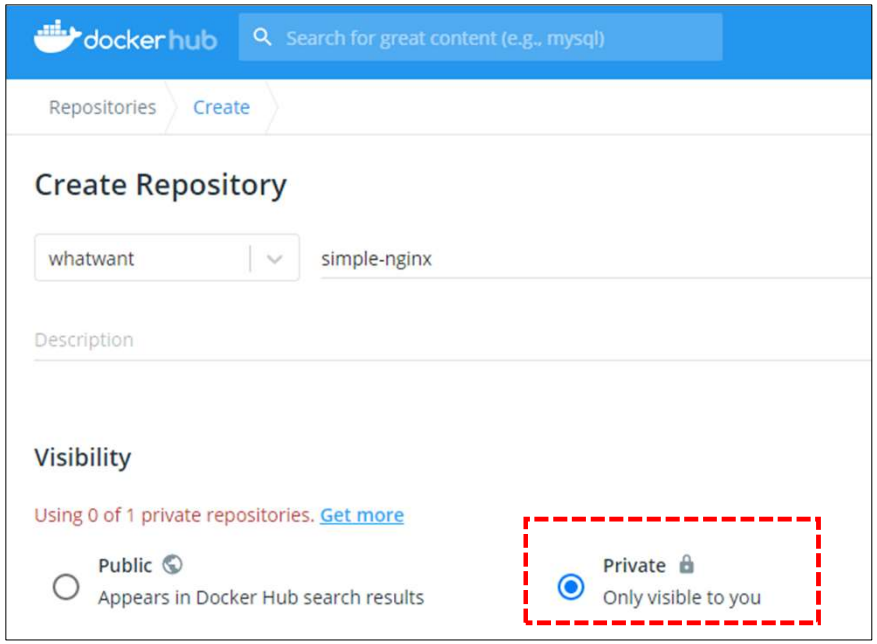
- Private Image를 하나 만들어서 Docker Hub에 업로드 해놓자.

Index.html

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Docker Nginx</title>
</head>
<body>
  <h2>Hello from Nginx container</h2>
</body>
</html>
```

Dockerfile

```
FROM nginx:latest

COPY ./04-index.html /usr/share/nginx/html/index.html
```




```
remote ❯ git clone https://github.com/whatwant-school/advanced-kubernetes.git
remote ❯ cd advanced-kubernetes/05-week/secret

remote ❯ docker login

remote ❯ docker build -t whatwant/simple-nginx:v0.1 -f 04-Dockerfile .

remote ❯ docker push whatwant/simple-nginx:v0.1
```

local에 build 한 뒤, tagging 하고 push를 해왔는데,

이렇게 바로 DockerHub 用 tagging으로 build를 바로 해버릴 수도 있다.

docker registry : Failed to pull image

- 앞에서 등록한 Private Image를 그냥 내려 받으면 어떻게 되는지 확인해보자.

pod-private-fail.yaml

```yaml
apiVersion: v1
kind: Pod

metadata:
  name: pod-private
  labels:
    app: nginx

spec:
  containers:
  - name: nginx
    image: whatwant/simple-nginx:v0.1
```

```
remote ❯ git clone https://github.com/whatwant-school/advanced-kubernetes.git
remote ❯ cd advanced-kubernetes

remote ❯ kubectl create -f ./05-week/secret/pod-private-fail.yaml

pod/pod-private created


remote ❯ kubectl get pods -o wide

NAME          READY   STATUS         RESTARTS   AGE   IP              NODE      NOMINATED NODE   READINESS GATES
pod-private   0/1     ErrImagePull   0          16s   10.233.103.42   worker2   <none>           <none>
```

```
remote ❯ kubectl describe pods pod-private
...
Events:
  Type     Reason     Age                From               Message
  ----     ------     ----               ----               -------
  Normal   Scheduled  2m10s              default-scheduler  Successfully assigned default/pod-private to worker2
  Normal   Pulling    31s (x4 over 2m9s)  kubelet            Pulling image "whatwant/simple-nginx:v0.1"
  Warning  Failed     28s (x4 over 2m5s)  kubelet            Failed to pull image "whatwant/simple-nginx:v0.1": rpc error: code = Unknown desc = failed
to pull and unpack image "docker.io/whatwant/simple-nginx:v0.1": failed to resolve reference "docker.io/whatwant/simple-nginx:v0.1": pull access denied,
repository does not exist or may require authorization: server message: insufficient_scope: authorization failed
  Warning  Failed     28s (x4 over 2m5s)  kubelet            Error: ErrImagePull
  Warning  Failed     16s (x6 over 2m5s)  kubelet            Error: ImagePullBackOff
  Normal   BackOff    4s (x7 over 2m5s)   kubelet            Back-off pulling image "whatwant/simple-nginx:v0.1"
```

docker registry : Create secret

- Registry Server 인증 정보를 secret으로 등록하자

pod-private-success.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod-private
  labels:
    app: nginx

spec:
  containers:
  - name: nginx
    image: whatwant/simple-nginx:v0.1

  imagePullSecrets:
    - name: my-docker-hub
```

```
kubectl create secret docker-registry <secret-name> \
                    [ --docker-server=<your-registry-server> \ ]
                      --docker-username=<your-name> \
                      --docker-password=<your-password> \
                      --docker-email=<your-email>
```

```
remote ❯ kubectl create secret docker-registry my-docker-hub \
                            --docker-username=whatwant \
                            --docker-password='xxx' \
                            --docker-email='whatwant@gmail.com'
secret/my-docker-hub created


remote ❯ kubectl get secrets -o wide

NAME                  TYPE                                  DATA   AGE
default-token-xf884   kubernetes.io/service-account-token   3      24d
my-docker-hub         kubernetes.io/dockerconfigjson        1      19s
```

```
remote ❯ kubectl create -f ./05-week/secret/pod-private-success.yaml

pod/pod-private created


remote ❯ kubectl get pods

NAME          READY   STATUS    RESTARTS   AGE
pod-private   1/1     Running   0          3m4s
```

Type of Secret

빌트인 타입	사용처
Opaque	임의의 사용자 정의 데이터
kubernetes.io/service-account-token	서비스 어카운트 토큰
kubernetes.io/dockercfg	직렬화 된(serialized) ~/.dockercfg 파일
kubernetes.io/dockerconfigjson	직렬화 된 ~/.docker/config.json 파일
kubernetes.io/basic-auth	기본 인증을 위한 자격 증명(credential)
kubernetes.io/ssh-auth	SSH를 위한 자격 증명
kubernetes.io/tls	TLS 클라이언트나 서버를 위한 데이터
bootstrap.kubernetes.io/token	부트스트랩 토큰 데이터

※ 참고 : https://kubernetes.io/ko/docs/concepts/configuration/secret/

///

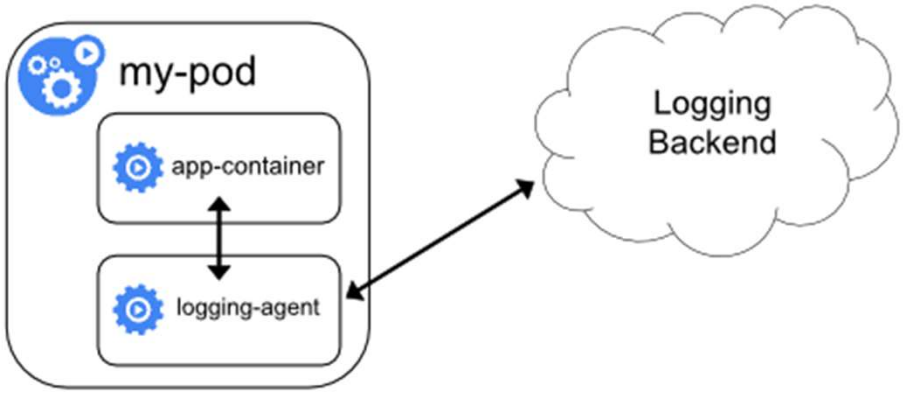
volume gitRepo: Deprecated → ‘git-sync’ Sidecar Container

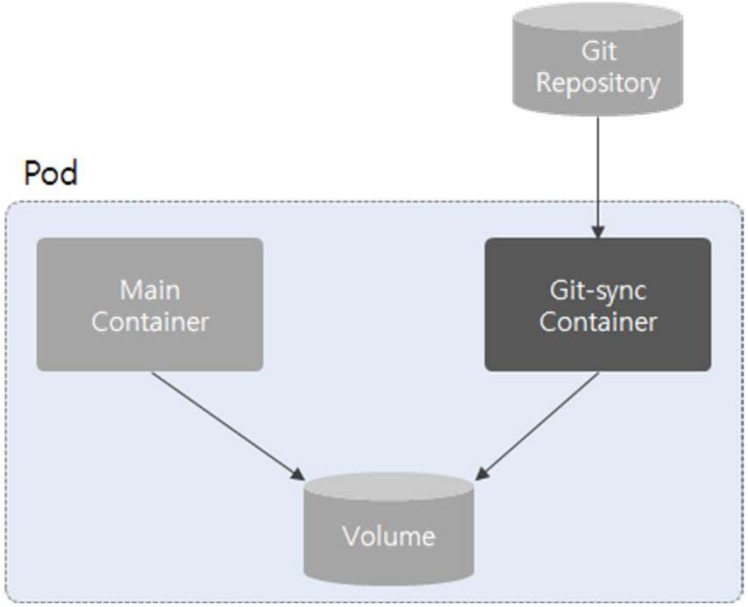
- 사이드카 패턴 (Sidecar Pattern)

  . 기본 컨테이너의 기능을 확장하거나 강화하는 용도의 컨테이너를 추가하는 패턴

  . 기본 컨테이너에는 원래 목적의 기능에만 충실하고 나머지 부가적인 공통 기능들은 사이드카 컨테이너를 추가해서 사용

- deprecate 된 gitRepo volume 대신 git-sync의 sidecar container를 구성하는 것으로 구현해야 한다





※ 참고 : https://arisu1000.tistory.com/27863

※ 참고 : https://github.com/kubernetes/git-sync

secret with SSH

- ssh-key 정보와 known_hosts 정보를 secret으로 등록해보자

  . ssh-key를 아직 생성하지 않았다면, `ssh-keygen`으로 생성 후 진행하면 된다.

```
remote ❯ ssh-keyscan github.com > ./known_hosts

# github.com:22 SSH-2.0-babeld-e1420b26
# github.com:22 SSH-2.0-babeld-e1420b26
# github.com:22 SSH-2.0-babeld-e1420b26
```

github.com 접근을 위해 미리 known_hosts에 등록해야 한다.

그렇지 않으면, 처음 접근하는 서버의 등록 과정으로 문제가 발생하게 된다.

```
remote ❯ kubectl create secret generic my-ssh --from-file=ssh=$HOME/.ssh/id_rsa --from-file=known_hosts=./known_hosts

secret/my-ssh created


remote ❯ kubectl get secrets -o wide

NAME                  TYPE                                  DATA   AGE
default-token-xf884   kubernetes.io/service-account-token   3      24d
my-ssh                Opaque                                2      12s
```

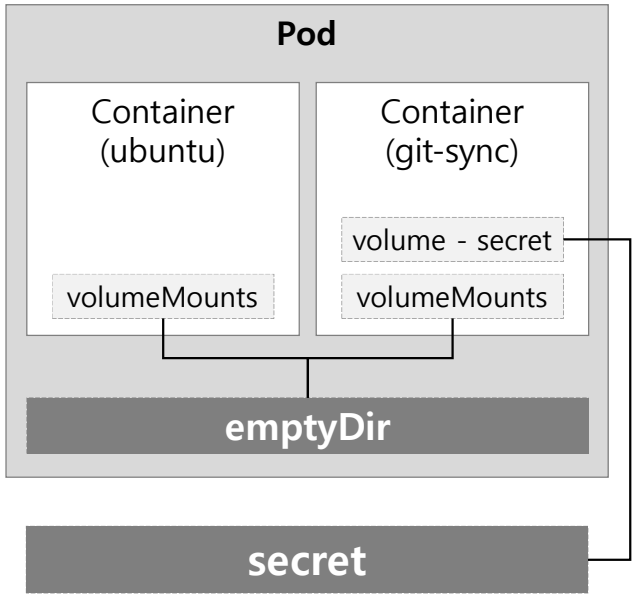
‘git-sync’ Sidecar Container - 1/2

pod-git-sync.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: git-sync

spec:
  containers:
  - name: ubuntu
    image: ubuntu:latest
    args:
    - sleep
    - "370000"

    volumeMounts:
    - name: myrepo
      mountPath: "/repo"

  - name: git-sync
    image: k8s.gcr.io/git-sync/git-sync:v3.2.2
    args:
    - -ssh
    - -repo=git@github.com:whatwant-school/private-repo.git
    - -root=/repo
    - -dest=private-repo
    - -branch=main
    - -depth=1
    volumeMounts:
    - name: myrepo
      mountPath: "/repo"

    - name: git-secret
      mountPath: /etc/git-secret
      readOnly: true

    securityContext:
      runAsUser: 65533

  securityContext:
    fsGroup: 65533

  restartPolicy: Never

  volumes:
  - name: myrepo
    emptyDir: {}

  - name: git-secret
    secret:
      secretName: my-ssh
      defaultMode: 0400
```

※ 참고 : https://github.com/kubernetes/git-sync/blob/release-3.x/docs/ssh.md

‘git-sync’ Sidecar Container - 2/2




```
remote ❯ git clone https://github.com/whatwant-school/advanced-kubernetes.git
remote ❯ cd advanced-kubernetes

remote ❯ kubectl create -f ./05-week/secret/pod-git-sync.yaml

pod/git-sync created

remote ❯ kubectl get pods -o wide

NAME       READY   STATUS    RESTARTS   AGE   IP              NODE      NOMINATED NODE   READINESS GATES
git-sync   2/2     Running   0          32s   10.233.103.50   worker2   <none>           <none>

remote ❯ kubectl exec -it git-sync -c ubuntu -- /bin/bash

groups: cannot find name for group ID 65533
root@git-sync:/# ls -al /repo
total 16
drwxrwsrwx 4 root  65533 4096 Feb 15 17:56 .
drwxr-xr-x 1 root  root  4096 Feb 15 17:56 ..
drwxr-sr-x 9 65533 65533 4096 Feb 15 17:56 .git
lrwxrwxrwx 1 65533 65533   44 Feb 15 17:56 private-repo -> rev-56af7aa4197fee5da30ccdd562cc2ea5e751af76
drwxr-sr-x 2 65533 65533 4096 Feb 15 17:56 rev-56af7aa4197fee5da30ccdd562cc2ea5e751af76

root@git-sync:/# ls -al /repo/private-repo/
total 16
drwxr-sr-x 2 65533 65533 4096 Feb 15 17:56 .
drwxrwsrwx 4 root  65533 4096 Feb 15 17:56 ..
-rw-r--r-- 1 65533 65533   71 Feb 15 17:56 .git
-rw-r--r-- 1 65533 65533   60 Feb 15 17:56 README.md
```

///

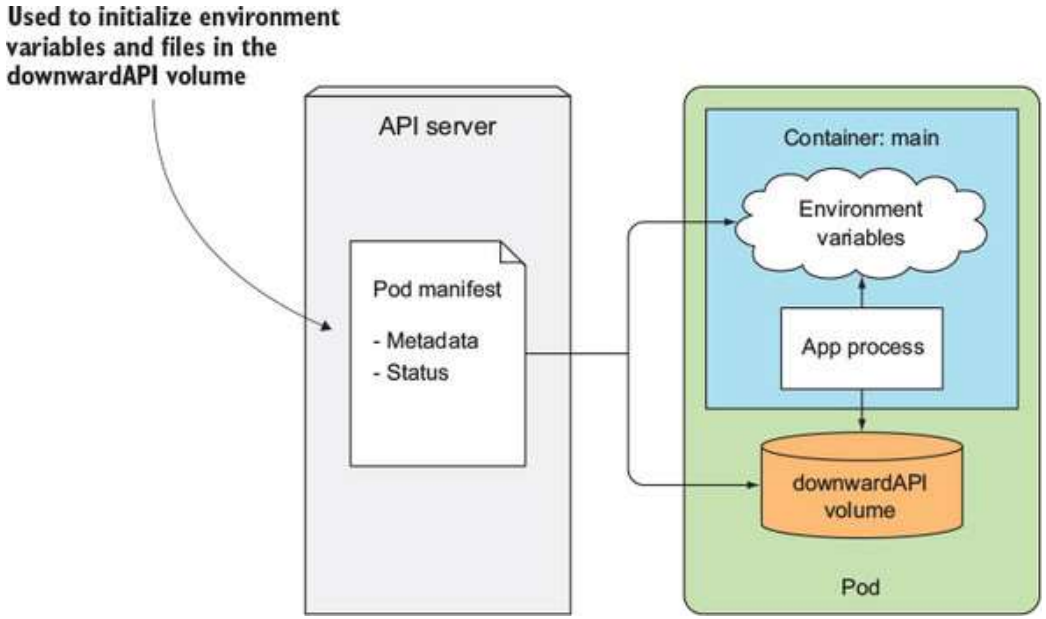
Kubernetes

Volume - downwardAPI

Why downwardAPI

- Pod의 IP, 호스트 노드 이름, Pod 자체의 이름과 같이 실행 시점까지 알려지지 않은 데이터를 얻기 위한 방법

  . downwardAPI는 애플리케이션이 호출해서 데이터를 가져오는 REST Endpoint와는 다르다.



※ 참고 : https://livebook.manning.com/book/kubernetes-in-action/chapter-8/11

capabilities of downwardAPI

Information available via	item	comment
fieldRef:	metadata.name	the pod's name
	metadata.namespace	the pod's namespace
	metadata.uid	the pod's UID
	metadata.labels['<KEY>']	the value of the pod's label <KEY> (for example, metadata.labels['mylabel'])
	metadata.annotations['<KEY>']	the value of the pod's annotation <KEY> (for example, metadata.annotations['myannotation'])
resourceFieldRef:	A Container's CPU limit	
	A Container's CPU request	
	A Container's memory limit	
	A Container's memory request	
	A Container's hugepages limit	providing that the DownwardAPIHugePages feature gate is enabled
	A Container's hugepages request	providing that the DownwardAPIHugePages feature gate is enabled
	A Container's ephemeral-storage limit	
	A Container's ephemeral-storage request	
downwardAPI volume fieldRef:	metadata.labels	all of the pod's labels, formatted as label-key="escaped-label-value" with one label per line
	metadata.annotations	all of the pod's annotations, formatted as annotation-key="escaped-annotation-value" with one annotation per line
environment variables:	status.podIP	the pod's IP address
	spec.serviceAccountName	the pod's service account name, available since v1.4.0-alpha.3
	spec.nodeName	the node's name, available since v1.4.0-alpha.3
	status.hostIP	the node's IP, available since v1.7.0-alpha.1

※ 참고 : https://kubernetes.io/docs/tasks/inject-data-application/downward-api-volume-expose-pod-information/#capabilities-of-the-downward-api

downwardAPI volume example - 1/2

dapi-volume.yaml

```yaml
apiVersion: v1
kind: Pod

metadata:
  name: kubernetes-downwardapi-volume-example
  labels:
    zone: us-est-coast
    cluster: test-cluster1
    rack: rack-22
  annotations:
    build: two
    builder: john-doe

spec:
  containers:
    - name: client-container
      image: k8s.gcr.io/busybox
      command: ["sh", "-c"]

      args:
      - while true; do
          if [[ -e /etc/podinfo/labels ]]; then
            echo -en '₩n₩n'; cat /etc/podinfo/labels; fi;
          if [[ -e /etc/podinfo/annotations ]]; then
            echo -en '₩n₩n'; cat /etc/podinfo/annotations; fi;
          sleep 5;
        done;
      volumeMounts:
        - name: podinfo
          mountPath: /etc/podinfo

  volumes:
    - name: podinfo
      downwardAPI:
        items:
          - path: "labels"
            fieldRef:
              fieldPath: metadata.labels
          - path: "annotations"
            fieldRef:
              fieldPath: metadata.annotations
```

※ 참고 : https://k8s.io/examples/pods/inject/dapi-volume.yaml

///

Tip

쿠버네티스 어린이 그림 가이드

※ 참고 : https://www.youtube.com/watch?v=4ht22ReBjno

///

https://kahoot.it/

[Score]

이민준 (7)
김남형 (4)
박남준 (3)
이혜정 (3)
김상호 (2)
이원준 (2)
정현찬 (1)
김정은 (1)