올인원 패키지 Online.

CKA 쿠버네티스 자격증 과정

PART1 | CKA 자격증 소개
자격증 등록과 Hands-On 실습 환경 만들기

PART2 | Kubernetes 아키텍쳐
Etcd 백업/복구, Kubernetes 업그레이드, RBAC 인증 실습

PART3 | Workloads & Scheduling
Application deploy 후 scale/rollback, pod 스케줄링 실습

PART4 | Services & Networking
Service 운영과 접근제한(NetworkPolicy),Ingress 운영 실습

PART5 | Storage
StorageClass 적용한 PV, PVC 생성하여 pod에 적용하기 실습

PART6 | Troubleshooting
controller component 설정 정보 수정 및 node/pod 문제해결

PART7 | 실전문제풀이
실전 문제를 직접 풀어 보고, 답안 리뷰 확인

Part 5 Storage

01 Volume Mount - emptyDir
02 Volume Mount - hostPath
03 Storage Class
04 Persistent Volume
05 Persistent Volume Claim

Part 5 Storage

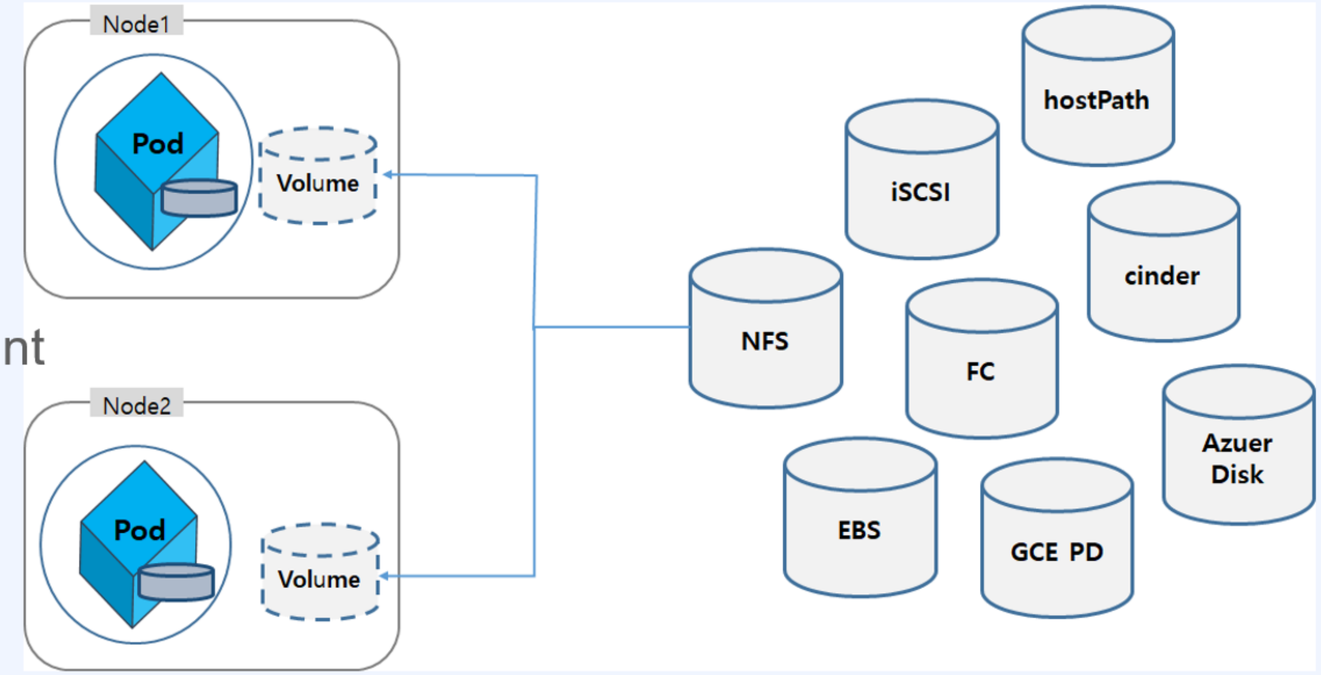
01 Volume Mount - emptyDir

- Volume은 kubernetes 스토리지의 추상화 개념
  - 컨테이너는 Pod에 바인딩 되는 볼륨을 마운트하고 마치 로컬 파일시스템에 있는 것처럼 스토리지에 접근한다.
- Kubernetes 스토리지

```
volumes:
- name: html
  hostPath:
    path: /hostdir_or_file
```

- 컨테이너 단위로 mount

```
volumeMounts:
- name: html
  mountPath: /webdata
```

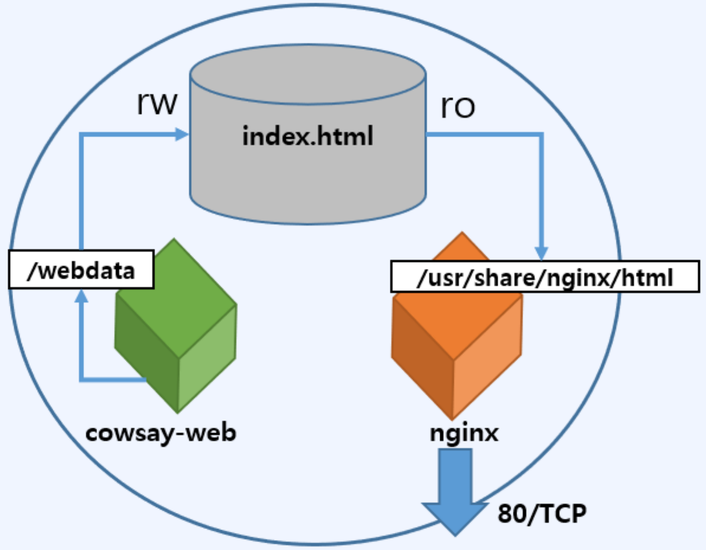


Kubernetes Volume type

- volume type
  - emptyDir
  - HostPath
  - gitRepo
  - nfs
  - gcePersistentDisk, awsElastic-BlockStore, azureDisk
  - cinder, cephfs, iscsi, flocker, glusterfs, quobyte, rbd, flexVolume, vsphere-volume, photonPersistentDisk, scaleIO
  - configMap, secret, downwardAPI
  - persistentVolumeClaim

emptyDir Volume

- Volume을 통해 컨테이너 간 데이터 공유
  - emptyDir 볼륨은 빈 디렉토리로 시작
  - Pod 내부에서 실행중인 애플리케이션은 필요한 모든 파일을 작성
  - Pod를 삭제하면 볼륨의 내용이 손실됨
  - 동일한 Pod에서 실행되는 컨테이너 간에 파일을 공유할 때 유용



volume-empty.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: dynamic-web
spec:
  containers:
  - image: smlinux/cowsay-web
    name: web-generator
    volumeMounts:
    - name: html
      mountPath: /webdata
  - image: nginx:1.14
    name: web-server
    volumeMounts:
    - name: html
      mountPath: /usr/share/nginx/html
      readOnly: true
    ports:
    - containerPort: 80
  volumes:
  - name: html
    emptyDir: {}
```

문제1:
emptyDir Volume을 공유하는 multi-pod 운영

■ 작업 클러스터 : **k8s**

- 다음 조건에 맞춰서 nginx 웹서버 pod가 생성한 로그파일을 받아서 STDOUT으로 출력하는 busybox 컨테이너를 운영하시오.
- Pod Name: weblog
- Web container:
  - Image: nginx:1.17
  - Volume mount : /var/log/nginx
  - readwrite
- Log container:
  - Image: busybox
  - Command: /bin/sh, -c, "tail -n+1 -f /data/access.log"
  - Volume mount : /data
  - readonly
- emptyDir 볼륨을 통한 데이터 공유

```
$ kubectl run weblog --image=nginx:1.17 --dry-run=client -o yaml > weblog.yaml
$ vi weblog.yaml
...
$ kubectl apply -f weblog.yaml
$ kubectl logs weblog -c log
$ kubectl describe pod weblog  | grep -A 3 -i mount
```

```
apiVersion: v1
kind: Pod
metadata:
  name: weblog
spec:
  containers:
  - image: nginx:1.17
    name: main
    volumeMounts:
    - mountPath: /var/log/nginx
      name: log-volume
  - image: busybox
    name: log
    args: [/bin/sh, -c, "tail -n+1 -F /data/access.log"]
    volumeMounts:
    - mountPath: /data
      name: log-volume
      readOnly: true
  volumes:
  - name: log-volume
    emptyDir: {}
```

Part 5 Storage

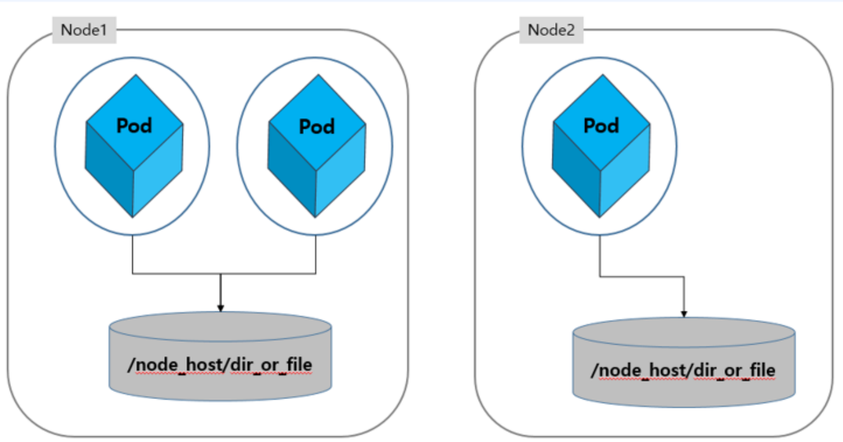
02 Volume Mount - hostPath

hostPath Volume

- hostPath
  - 노드의 파일시스템의 디렉토리나 파일을 컨테이너에 마운트
  - 노드에 디렉토리나 파일을 생성하여 마운트 가능
  - hostPath는 type 지시어를 이용해 mount 구성의 요구를 추가할 수 있다.

```
volumes:
- name: html
  hostPath:
    type: FileOrCreate
    path: /hostdir_or_file
```

DirectoryOrCreate	주어진 경로에 아무것도 없다면, 필요에 따라 kubelet의 소유권, 권한을 0755로 설정한 빈 디렉터리를 생성한다.
Directory	주어진 경로에 디렉터리가 있어야 함
FileOrCreate	주어진 경로에 아무것도 없다면, 필요에 따라 kubelet의 소유권, 권한을 0755로 설정한 file을 생성한다.
File	주어진 경로에 파일이 있어야 함




```
volumes:
- name: html
  hostPath:
    path: /hostdir_or_file
```

```
volumes:
- name: html
  local:
    path: /mount/dir
```

문제2:
HostPath Volume 구성

■ 작업 클러스터 : **k8s**

- /data/cka/fluentd.yaml 파일에 다음 조건에 맞게 볼륨 마운트를 설정하시오.
- Worker node의 도커 컨테이너 디렉토리를 동일 디렉토리로 pod에 마운트 하시오..
- Worker node의 /var/log 디렉토리를 fluentd Pod에 동일이름의 디렉토리 마운트하시오.

```
$ vi /data/cka/fluentd.yaml
...
$ kubectl get pod | grep fluentd

$ kubectl describe pod fluentd-xxx
```

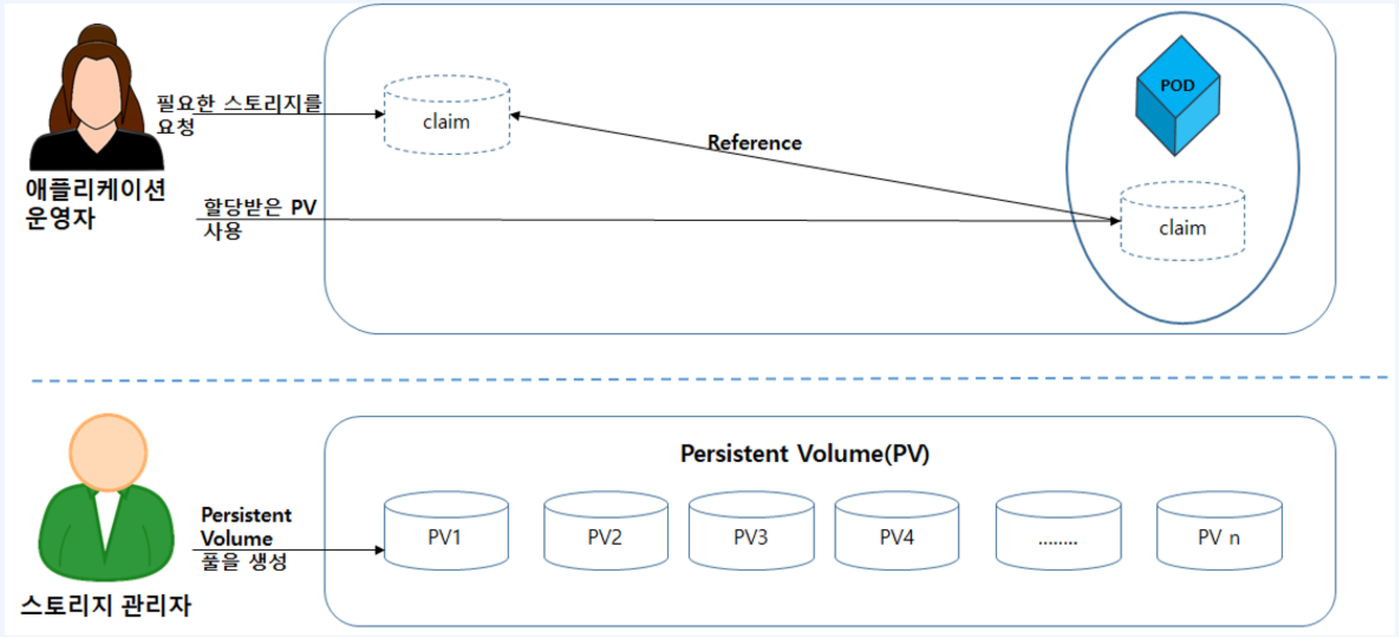
```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: fluentd
spec:
  selector:
    matchLabels:
      name: fluentd
  template:
    metadata:
      labels:
        name: fluentd
    spec:
      containers:
      - name: fluentd
        image: fluentd
        volumeMounts:
        - name: varlog
          mountPath: /var/log
        - name: varlibdockercontainers
          mountPath: /var/lib/docker/containers
          readOnly: true
      volumes:
      - name: varlog
        hostPath:
          path: /var/log
      - name: varlibdockercontainers
        hostPath:
          path: /var/lib/docker/containers
```

Part 5 Storage

03 Storage Class

Kubernetes Volume
운영환경 분리

- 기본 스토리지 운영환경을 분리
  - 관리자 : 스토리지 구성
  - 개발자 : 필요한 만큼 요구
- PersistentVolumes
- PersistentVolumeClaims



Dynamic Provisioning

- Dynamic Provisioning
  - 온-디맨드 방식으로 스토리지 볼륨을 생성
  - 사용자가 스토리지를 요청하면 자동으로 프로비저닝
  - 하나 이상의 StorageClass 오브젝트를 사전 생성

- StorageClass
  - 스토리지의 "classes"를 설명
  - StorageClass 에는 해당 StorageClass에 속하는 PV를 동적으로 프로비저닝 할 때 사용되는 Provisioner, parameters와 reclaimPolicy 필드가 포함
  - reclaimPolicy 가 지정되지 않으면 기본값은 Delete
  - 스토리지 클래스에 속하는 볼륨을 설명하는 parameters

- 프로비저너
  - 각 storageClass에는 PV 프로비저닝에 사용되는 볼륨 플러그인을 결정하는 프로비저너가 있다. 이 필드는 반드시 지정해야 한다.
  - https://kubernetes.io/ko/docs/concepts/storage/storage-classes/#%ED%94%84%EB%A1%9C%EB%B9%84%EC%A0%80%EB%84%88

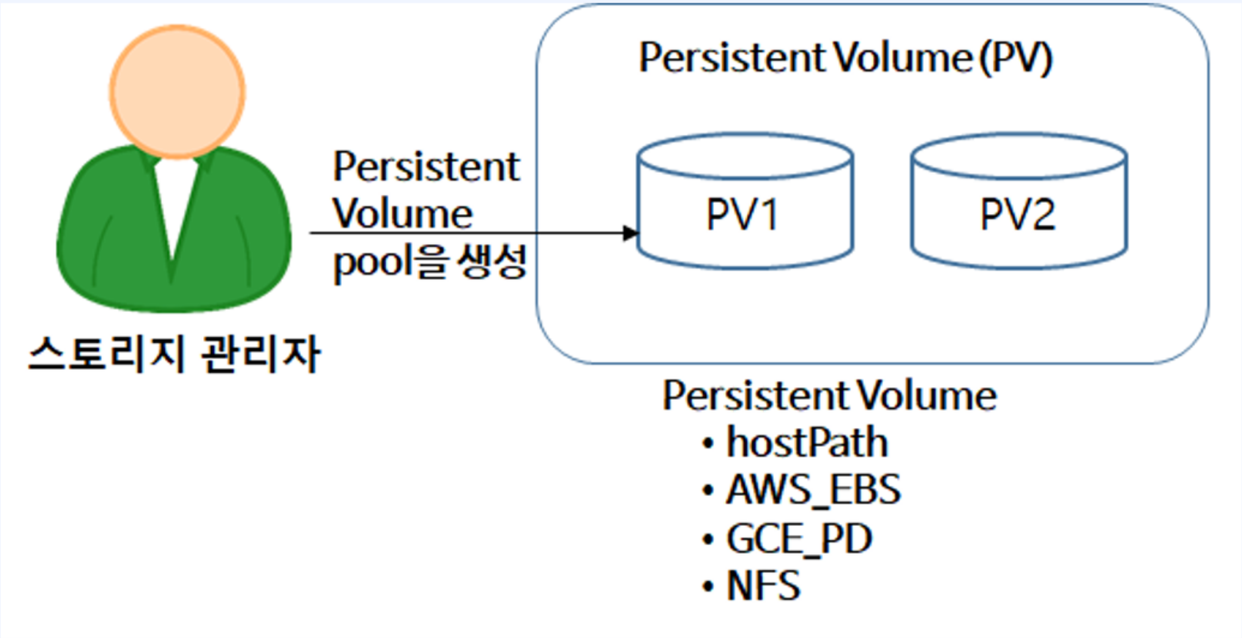
Part 5 Storage

04 Persistent Volume

PV(Persistent Volume) 만들기

pv.yaml

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pvname
spec:
  capacity:
    storage: <storage_size>
  accessModes:
  - ReadWriteOnce
  - ReadOnlyMany
  persistentVolumeReclaimPolicy: Retain
  nfs:
    server: <NFS_Server>
    path: <Share_Storage>
```



문제3:
PersistentVolume 만들기

■ 작업 클러스터 : **hk8s**

- pv001라는 이름으로 size 1Gi, access mode ReadWriteMany를 사용하여 persistent volume을 생성합니다.
- volume type은 hostPath이고 위치는 /tmp/app-config입니다.

```
$ kubectl config use-context hk8s
$ vi pv001.yaml
...

$ kubectl apply -f pv001.yaml

$ kubectl get pv

$ kubectl describe pv pv001
```

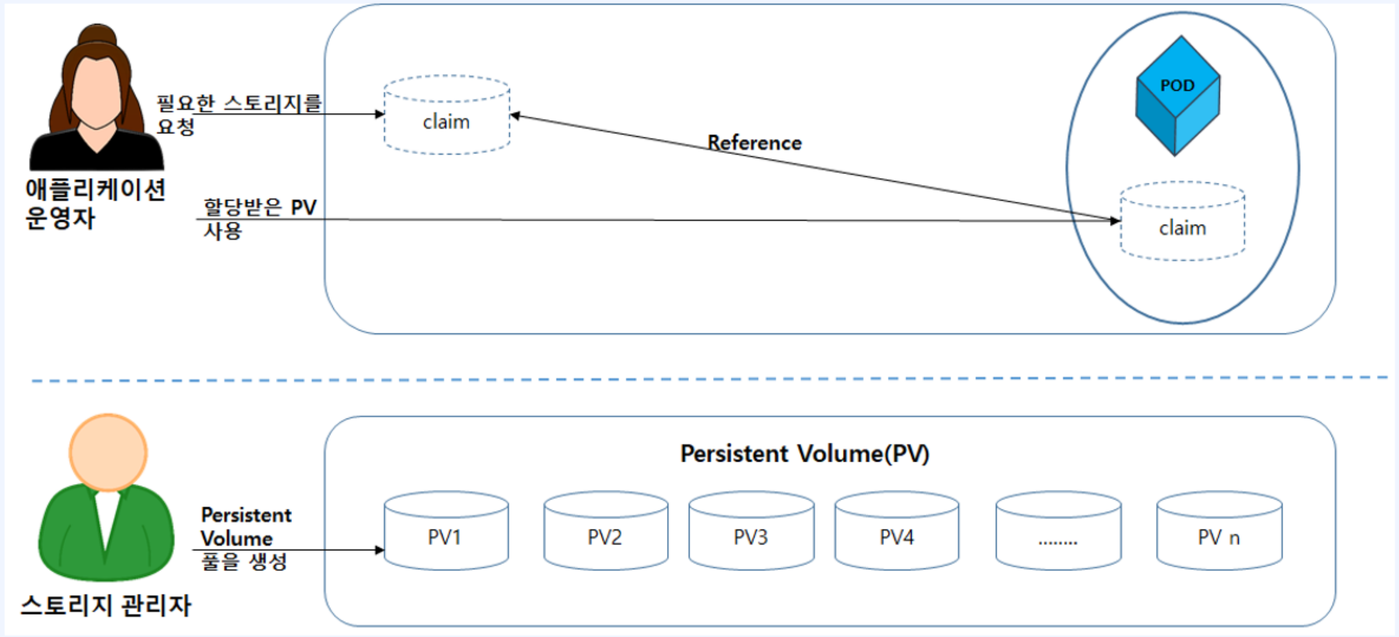
```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv001
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteMany
  hostPath:
    path: /tmp/app-config
```

Part 5 Storage

05 Persistent Volume Claim

Kubernetes Volume
운영환경 분리

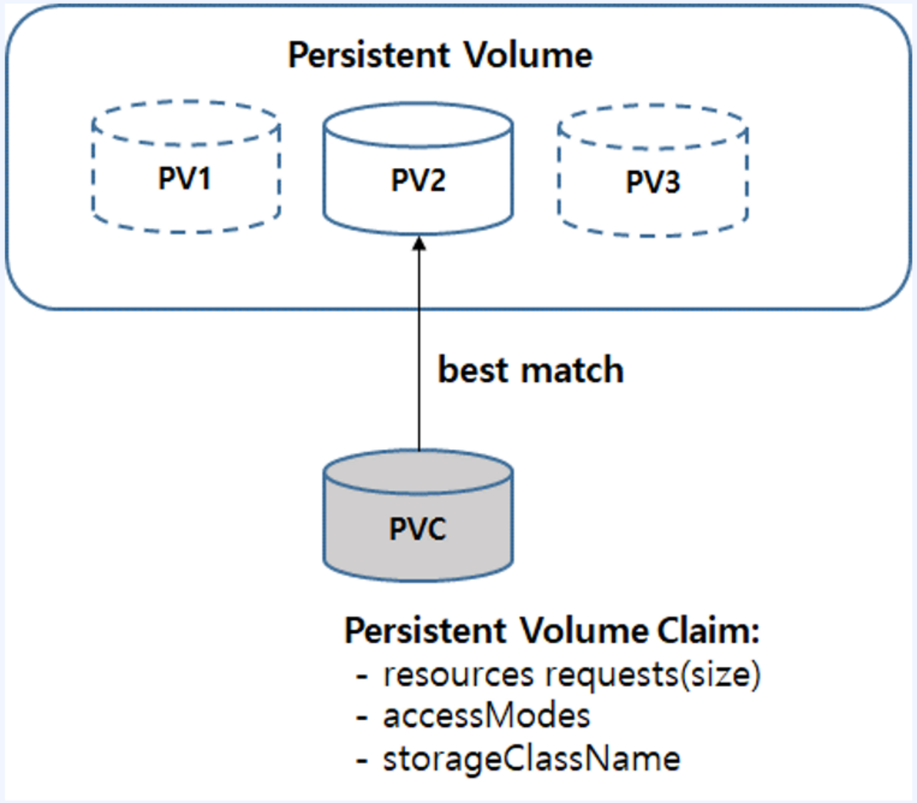
- 기본 스토리지 운영환경을 분리
  - 관리자 : 스토리지 구성
  - 개발자 : 필요한 만큼 요구
- PersistentVolumes
- PersistentVolumeClaims



PersistentVolumeClaim 만들기

pvc.yaml

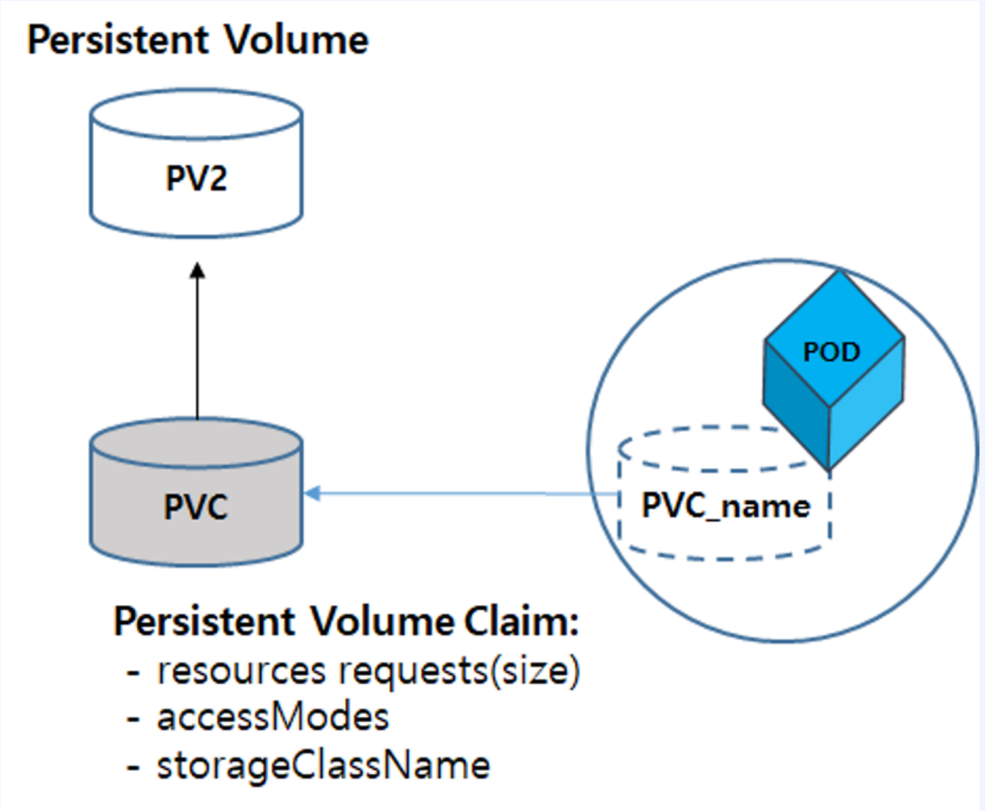
```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-name
spec:
  resources:
    requests:
      storage: size
  accessModes:
  - ReadWriteOnce
  - ReadOnlyMany
  storageClassName: "manuel“
```



PersistentVolumeClaim 을 pod에서 사용하기

pod.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: pod
spec:
  containers:
  - image: image
    name: container-name
    volumeMounts:
    - name: volume-name
      mountPath: /mount/dir
  volumes:
  - name: volume-name
    persistentVolumeClaim:
      claimName: pvc-name
```



문제5:
PVC를 사용하는 애플리케이션 Pod 운영

■ 작업 클러스터 : **k8s**

- 다음의 조건에 맞는 새로운 PersistentVolumeClaim 생성하시오.
  - Name: pv-volume
  - Class: app-hostpath-sc
  - Capacity: 10Mi
- 앞서 생성한 pv-volume PersistentVolumeClaim을 mount하는 Pod 를 생성하시오.
  - Name: web-server-pod
  - Image: nginx
  - Mount path: /usr/share/nginx/html
- Volume에서 ReadWriteMany 액세스 권한을 가지도록 구성합니다.

```
$ kubectl config use-context k8s
$ vi pvc.yaml
...
$ kubectl apply -f pvc.yaml
$ kubectl get pvc,pv

$ kubectl run web-server-pod --image=nginx --dry-run=client -o yaml > pod.yaml
$ vi pod.yaml
...
$ kubectl apply -f pod.yaml
$ kubectl get pod
$ kubectl describe pod web-server-pod
```

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pv-volume
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Mi
  storageClassName: app-hostpath-sc
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: web-server-pod
spec:
  containers:
    - name: web-server
      image: nginx
      volumeMounts:
      - mountPath: "/usr/share/nginx/html"
        name: pv-volume
  volumes:
    - name: pv-volume
      persistentVolumeClaim:
        claimName: pv-volume
```