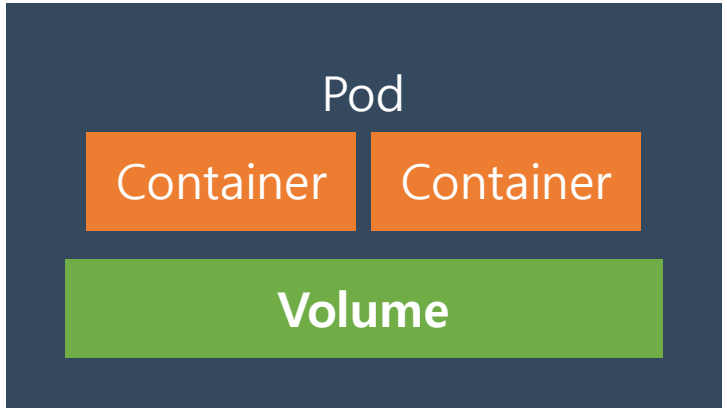
Volume은 Pod에 장착되어, 그 Pod에 있는 Container 간에 공유



Types of Volumes

<table>
<tr><th>임시 볼륨</th><th>로컬 볼륨</th><th>네트워크 볼륨</th><th>네트워크 볼륨
(Cloud dependent)</th></tr>
<tr><td rowspan="2">emptyDir</td><td rowspan="2">hostPath</td><td colspan="2">PersistentVolumeClaim</td></tr>
<tr><td>gitRepo, iSCSI, NFS
cephFS, glusterFS
.....</td><td>gcePersistentDisk,
AzureDisk,
Amazon EFS, Amazon EBS
...</td></tr>
<tr><td>✓ Pod내 컨테이너간 공유</td><td>✓ Host 디렉토리를 Pod와 공유해 사용하는 방식</td><td colspan="2">✓ 영구적으로 영속성 있는 데이터 관리 목적</td></tr>
<tr><td>✓ Pod가 삭제되면, emptyDir도 지워지므로 휘발성 데이터 저장 용도</td><td>✓ 컨테이너에 nodeSelecor를 지정안하면 매번 다른 호스트에 할당
✓ e.g. 호스트의 Metric 수집해야 하는 경우</td><td colspan="2">✓ 쿠버네티스는 PV와 PVC의 개념을 통해 Persistent 볼륨을 Pod에 제공

✓ gitRepo, iSCSI, NFS와 같은 표준 네트워크 볼륨과 Cloud Vendor가 제공하는 볼륨으로 구분</td></tr>
</table>

Volumes : emptyDir

kubectl apply -f volume-emptydir.yaml 아래 설정의 볼륨 생성

```
root@labs-910775232:/home/project/container-orchestration/yaml/volume/aws# cat volume-emptydir.yaml
apiVersion: v1
kind: Pod
metadata:
  name: shared-volumes
spec:
  containers:
  - image: redis
    name: redis
    volumeMounts:
    - name: shared-storage
      mountPath: /data/shared
  - image: nginx
    name: nginx
    volumeMounts:
    - name: shared-storage
      mountPath: /data/shared
  volumes:
  - name: shared-storage
    emptyDir: {}
```

emptyDir의 생명주기는 컨테이너 단위가 아닌 Pod 단위로 Container 재기동에도 계속 사용 가능

지정 컨테이너 접속 후, 파일 생성

```
root@labs-910775232:/home/project/container-orchestration/yaml/volume/aws# kubectl exec -it shared-volumes --container redis -- /bin/bash
root@shared-volumes:/data# cd /data/shared
root@shared-volumes:/data/shared# echo test\342\200\246 > test.txt
root@shared-volumes:/data/shared# exit
exit
```

동일 pod의 컨테이너간 볼륨을 공유함을 알 수 있음

다른 컨테이너로 접속 후, 파일 확인

```
root@labs-910775232:/home/project/container-orchestration/yaml/volume/aws# kubectl exec -it shared-volumes --container nginx -- /bin/bash
root@shared-volumes:/# cd /data/shared
root@shared-volumes:/data/shared# ls
test.txt
```

Volumes : hostPath

```
apiVersion: v1
kind: Pod
metadata:
  name: hostpath
spec:
  containers:
  - name: redis
    image: redis
    volumeMounts:
    - name: somepath
      mountPath: /data/shared
  volumes:
  - name : somepath
    hostPath:
      path: /tmp
      type: Directory
```

https://bcho.tistory.com/1259

- Node의 Local 디스크 경로를 Pod에 마운트
- 같은 hostPath에 있는 볼륨은 여러 Pod사이에서 공유
- Pod가 삭제되어도 hostPath에 있는 파일은 유지
- Pod가 재기동 되어 다른 Node에서 기동될 경우, 새로운 Node의 hostPath를 사용
- Node의 로그 파일을 읽는 로그 에이전트 컨테이너 등에 사용가능
- Pod 생성 및 확인 (Pod 내, ls -al /data/shared)

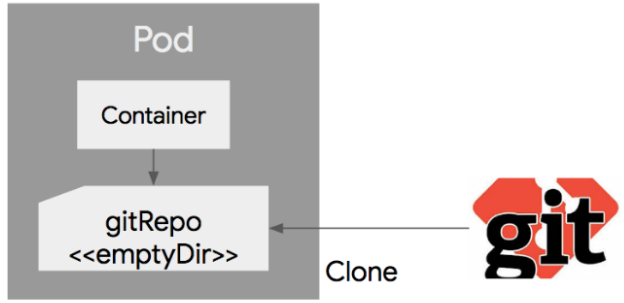
```
drwxrwxrwt 8 redis root  4096 Feb 17 03:08 .
drwxr-xr-x 3 redis redis 4096 Feb 17 03:07 ..
drwxrwxrwt 2 redis root  4096 Feb 11 05:39 .ICE-unix
drwxrwxrwt 2 redis root  4096 Feb 11 05:39 .Test-unix
drwxrwxrwt 2 redis root  4096 Feb 11 05:39 .X11-unix
drwxrwxrwt 2 redis root  4096 Feb 11 05:39 .XIM-unix
drwxrwxrwt 2 redis root  4096 Feb 11 05:39 .font-unix
drwx------ 3 redis root  4096 Feb 11 05:44 systemd-private-fe55104f60e34b2ea4
```

Volumes example : gitRepo

```
apiVersion: v1
kind: Pod
metadata:
 name: gitrepo-volume-pod
spec:
 containers:
 - image: nginx:alpine
   name: web-server
   volumeMounts:
   - name: html
     mountPath: /usr/share/nginx/html
     readOnly: true
   ports:
   - containerPort: 80
     protocol: TCP
 volumes:
 - name: html
   gitRepo:
       repository: https://github.com/luksa/kubia-website-
example.git
       revision: master
       directory: .
```

https://bcho.tistory.com/1259

- Pod 생성시 지정된 Git 리파지토리의 특정 리비전을 Cloning하여 디스크 볼륨 생성
- 물리적으로는 emptyDir이 생성되고 Git Clone 수행



- HTML 같은 정적 파일 및 Nodejs 같은 스크립트 기반 코드 배포에 유용

StorageClass - Dynamic PV Provisioning

StorageClass 등록, 조회

```
root@labs-910775232:/home/project/container-orchestration/yaml/volume/aws# cat efs-storageclass.yaml
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: aws-efs
provisioner: my-aws.com/aws-efsroot@labs-910775232:/home/project/container-orchestration/yaml/volume/aws# kubectl apply -f efs-storageclass.yaml
storageclass.storage.k8s.io/aws-efs created
```

```
root@labs-910775232:/home/project/container-orchestration/yaml/volume/aws# kubectl get sc
NAME            PROVISIONER             RECLAIMPOLICY   VOLUMEBINDINGMODE      ALLOWVOLUMEEXPANSION   AGE
aws-efs         my-aws.com/aws-efs      Delete          Immediate              false                  39s
gp2 (default)   kubernetes.io/aws-ebs   Delete          WaitForFirstConsumer   false                  6h25m
```

PVC 생성

```
root@labs-910775232:/home/project/container-orchestration/yaml/volume/aws# cat volume-pvc.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: aws-efs
  labels:
    app: test-pvc
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Mi
  storageClassName: aws-efsroot@labs-910775232:/home/project/container-orchestration/yaml/volume/aws# kubectl apply -f volume-pvc.yaml
persistentvolumeclaim/aws-efs created
```

PVC 조회

```
root@labs-910775232:/home/project/container-orchestration/yaml/volume/aws# kubectl get pvc
NAME      STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
aws-efs   Bound    pvc-9a767d38-5f68-4c57-ad2f-61537c4d5553   1Mi        RWX            aws-efs        46s
```

볼륨이 PVC에 의해 동적으로 만들어짐

```
root@labs-910775232:/home/project/container-orchestration/yaml/volume/aws# kubectl get pv
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM             STORAGECLASS   REASON   AGE
pvc-9a767d38-5f68-4c57-ad2f-61537c4d5553   1Mi        RWX            Delete           Bound    default/aws-efs   aws-efs                 106s
```

Create Pod with PersistentVolumeClaim

kubectl apply -f pod-with-pvc.yaml
kubectl get pod

```
root@labs-910775232:/home/project/container-orchestration/yaml/volume/aws# kubectl get pod
NAME                              READY   STATUS              RESTARTS   AGE
efs-provisioner-5f658489d5-tbsjv  1/1     Running             0          10m
gitrepo-volume-pod                0/1     ContainerCreating   0          8m55s
mypod                             0/1     ContainerCreating   0          5s
shared-volumes                    2/2     Running             0          73m
```

pod-with-pvc.yaml 내용 -> mountPath가 /mnt/aws

```
root@labs-910775232:/home/project/container-orchestration/yaml/volume/aws# cat pod-with-pvc.yaml
kind: Pod
apiVersion: v1
metadata:
  name: mypod
spec:
  containers:
  - name: mypod
    image: nginx:1.15.5
    resources:
      requests:
        cpu: 100m
        memory: 128Mi
      limits:
        cpu: 250m
        memory: 256Mi
    volumeMounts:
    - mountPath: "/mnt/aws"
      name: volume
  volumes:
    - name: volume
      persistentVolumeClaim:
        claimName: aws-efs
```

kubectl describe pod mypod

mountPath 위치에 생성

```
Problems      aws ×   namesapce

root@mypod:/# cd /mnt/aws
root@mypod:/mnt/aws# echo test > test.txt
root@mypod:/mnt/aws# cat test.txt
test
root@mypod:/mnt/aws# ls
test.txt
root@mypod:/mnt/aws#
```