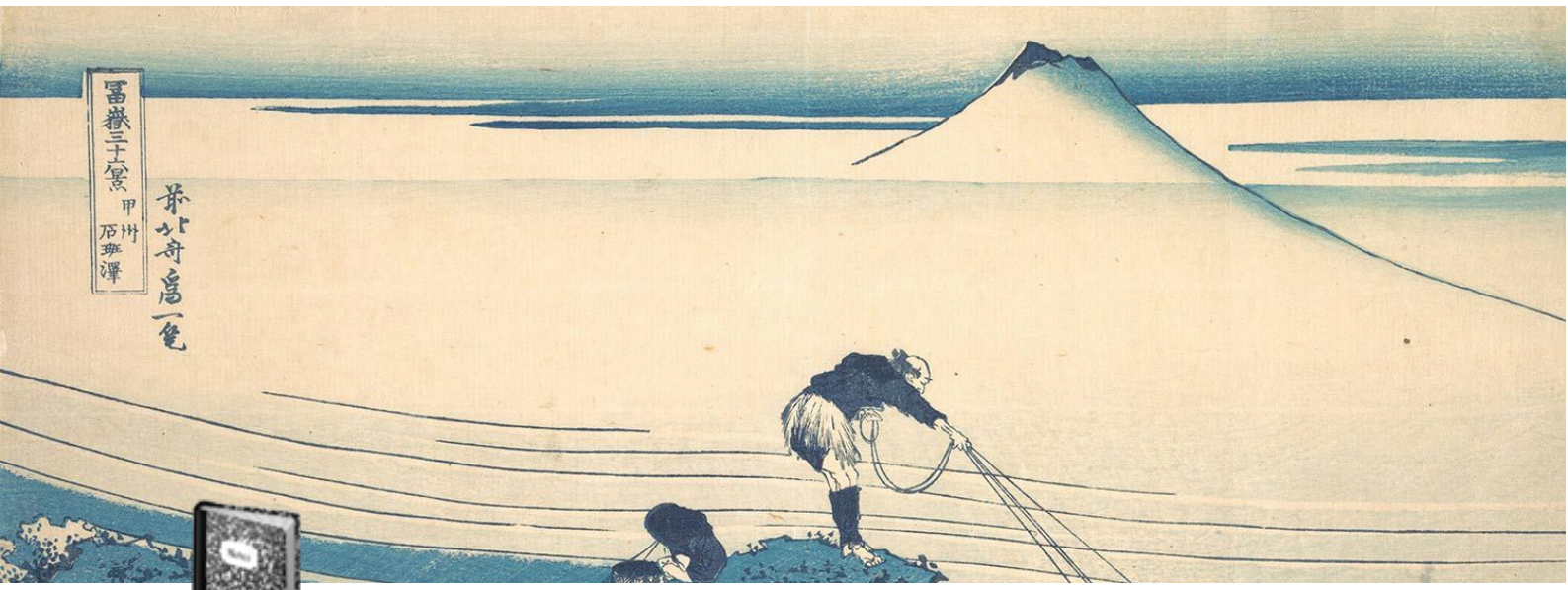

챕터2 컨테이너 저장소 활용과 컨테이너 롤링업데이트 전략 이해

Created 2021년 11월 6일 오후 2:48

Tags 비어 있음

▼ 목차

Secret, Configmap 마운트

Configmap 생성 yaml 예제

```
cat <<EOF >./example-redis-config.yaml apiVersion: v1 kind: ConfigMap metadata
: name: example-redis-config data: redis-config: | maxmemory 2mb maxmemory-pol
icy allkeys-lru EOF
```

Configmap을 사용하는 파드 예제 배포

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes/website/main/con
tent/en/examples/pods/config/redis-pod.yaml
```

redis cli를 통해서 설정 내용 확인

```
$ kubectl exec -it redis -- redis-cli 127.0.0.1:6379> CONFIG GET maxmemory-pol
icy 1) "maxmemory-policy" 2) "allkeys-lru" 127.0.0.1:6379> CONFIG GET maxmemor
y 1) "maxmemory" 2) "2097152"
```

Secret 생성 yaml 예제

```
echo -n admin > username echo -n 1q2w3e > password kubectl create secret gener
ic mysecret --from-file=username --from-file=password
```

Secret을 사용하는 파드 예제 배포

```
cat <<EOF | kubectl apply -f - apiVersion: v1 kind: Pod metadata: name: mypod
spec: containers: - name: mypod image: redis volumeMounts: - name: foo mountPa
th: "/etc/foo" readOnly: true volumes: - name: foo secret: secretName: mysecre
t EOF
```

NFS를 활용한 네트워크 스토리지

NFS 서버 구성하기

서버 설치 방법

```
apt-get update apt-get install nfs-common nfs-kernel-server portmap -y
```

공유할 디렉토리 생성

```
mkdir /home/nfs chmod 777 /home/nfs
```

/etc/exports 파일에 다음 내용 추가

```
/home/nfs 172.30.6.90(rw,sync,no_subtree_check) 172.30.5.197(rw,sync,no_subtre
e_check) 172.30.4.193(rw,sync,no_subtree_check)
```

서비스 재시작

```
service nfs-server restart showmount -e 127.0.0.1
```

NFS 클라이언트에서는 mount 명령어로 마운트해서 사용

```
mount -t nfs 172.30.5.16:/home/nfs /mnt
```

노드에서 NFS로 접속할 수 있도록 구성

nfs 관련 라이브러리 설치

```
apt-get update apt-get install nfs-common nfs-kernel-server portmap -y
```

/home/nfs에 index.html

```
<h1>This is Index Page</h1> <p>This is Index Page</p>
```

파드 yaml 파일을 사용해 nfs 볼륨을 마운트 하도록 구성

```
# nfs-httpd.yaml apiVersion: v1 kind: Pod metadata: name: nfs-httpd spec: cont
ainers: - image: httpd name: web volumeMounts: - mountPath: /usr/local/apache
2/htdocs name: nfs-volume readOnly: true volumes: - name: nfs-volume nfs: serv
er: 172.30.5.16 path: /home/nfs
```

포트포워드 실행해 포트를 오픈

```
kubectl port-foward nfs-httpd 8888:80
```

오픈된 포트로 접속 실행

```
curl 127.0.0.1:8888
```

rook-ceph를 활용한 프라이빗 클라우드 스토리지클래스

비어있는 스토리지(파티션이 할당되지 않은) 디스크를 하나 구성하고 다음 명령으로 확인

```
lsblk
```

루크로 ceph 설치하기

```
git clone --single-branch --branch release-1.7 https://github.com/rook/rook.gi
t cd rook/cluster/examples/kubernetes/ceph kubectl create -f crds.yaml -f comm
on.yaml -f operator.yaml kubectl create -f cluster.yaml
```

툴 박스와 ceph csi를 설치

```
kubectl create -f toolbox.yaml kubectl create -f csi/rbd/storageclass.yaml
```

툴박스 컨테이너로 상태 확인

```
$ kubectl -n rook-ceph exec -it $(kubectl -n rook-ceph get pod -l "app=rook-ceph-tools" -o jsonpath='{.items[0].metadata.name}') -- bash # 컨테이너 내부에서 다음 명령 실행 ceph status ceph osd pool stats
```

스토리지 클래스 확인

```
kubectl get sc
```

pvc를 다음과 같이 생성하고 디스크와 매칭되는지 확인

```
cat <<EOF | kubectl apply -f - apiVersion: v1 kind: PersistentVolumeClaim metadata: name: mongo-pvc spec: storageClassName: rook-ceph-block # 딜러지정 accessModes: - ReadWriteOnce resources: requests: storage: 2Gi EOF
```

파드를 생성해 최종적으로 파드와 연결

```
apiVersion: v1 kind: Pod metadata: name: mongodb spec: containers: - image: mongo name: mongodb volumeMounts: - name: mongodb-data mountPath: /data/db ports: - containerPort: 27017 protocol: TCP volumes: - name: mongodb-data persistentVolumeClaim: claimName: mongo-pvc
```

스토리지 클래스 연습문제

httpd를 사용할 수 있도록 pod, pvc, storageclass 정의하여 동적 프로비저닝을 수행하자.

httpd를 사용할 수 있도록 자동으로 pod, pvc, storageclass를 정의하고 생성하라.

▼ 풀이

```
cat <<EOF | kubectl apply -f - # pod apiVersion: v1 kind: Pod metadata: na
me: httpd spec: containers: - name: httpd image: httpd volumeMounts: - mou
ntPath: "/usr/local/apache2/htdocs/" name: htdocs volumes: - name: htdocs
persistentVolumeClaim: claimName: httpd-pvc --- # pvc apiVersion: v1 kind:
PersistentVolumeClaim metadata: name: httpd-pvc spec: accessModes: - ReadW
riteOnce volumeMode: Filesystem resources: requests: storage: 8Gi storageC
lassName: my-sc --- # storageclass # rook/cluster/examples/kubernetes/cep
h/csi/rbd/storageclass.yaml apiVersion: storage.k8s.io/v1 kind: StorageCla
ss metadata: name: my-sc provisioner: rook-ceph.rbd.csi.ceph.com parameter
s: clusterID: rook-ceph # namespace:cluster pool: replicapool imageFormat:
"2" imageFeatures: layering csi.storage.k8s.io/provisioner-secret-name: ro
ok-csi-rbd-provisioner csi.storage.k8s.io/provisioner-secret-namespace: ro
ok-ceph # namespace:cluster csi.storage.k8s.io/controller-expand-secret-na
me: rook-csi-rbd-provisioner csi.storage.k8s.io/controller-expand-secret-n
amespace: rook-ceph csi.storage.k8s.io/node-stage-secret-name: rook-csi-rb
d-node csi.storage.k8s.io/node-stage-secret-namespace: rook-ceph # namespa
ce:cluster csi.storage.k8s.io/fstype: ext4 allowVolumeExpansion: true recl
aimPolicy: Delete EOF
```

ceph 디스크 확인 방법

생성된 디스크 확인하기

```
kubectl get pv -o yaml | grep vol
```

툴박스 실행 후 디스크 이미지 리스트 확인

```
$ kubectl -n rook-ceph exec -it $(kubectl -n rook-ceph get pod -l "app=rook-
ceph-tools" -o jsonpath='{.items[0].metadata.name}') -- bash 다음 내용은 ceph 툴박
스에서 실행 # rbd ls -p replicapool csi-vol-c1c3dccc-47c5-11ec-a94b-0aa186c35891
```

디스크 상세 정보 확인

```
# rbd info csi-vol-c1c3dccc-47c5-11ec-a94b-0aa186c35891 -p replicapool rbd
image 'csi-vol-c1c3dccc-47c5-11ec-a94b-0aa186c35891': size 2 GiB in 512
objects order 22 (4 MiB objects) snapshot_count: 0 id: 84e9c5c99c1e
block_name_prefix: rbd_data.84e9c5c99c1e format: 2 features: layering
op_features: flags: create_timestamp: Wed Nov 17 16:45:22 2021
access_timestamp: Wed Nov 17 16:45:22 2021 modify_timestamp: Wed Nov 17
16:45:22 2021
```

ceph 디스크 사용량 확인

```
# ceph df --- RAW STORAGE --- CLASS SIZE AVAIL USED RAW USED %RAW USED hdd 300
GiB 300 GiB 78 MiB 78 MiB 0.03 TOTAL 300 GiB 300 GiB 78 MiB 78 MiB 0.03 ---
POOLS --- POOL ID PGS STORED OBJECTS USED %USED MAX AVAIL
device_health_metrics 1 64 0 B 0 0 B 0 95 GiB replicapool 2 32 4.2 MiB 26 13
MiB 0 95 GiB
```

스테이트풀셋

https://kubernetes.io/ko/docs/concepts/workloads/controllers/statefulset/
스테이트풀셋 예제를 사용해 배포해보자.

```
# stateful-nginx.yaml apiVersion: v1 kind: Service metadata: name: nginx label
s: app: nginx spec: ports: - port: 80 name: web clusterIP: None # headless sel
ector: app: nginx --- apiVersion: apps/v1 kind: StatefulSet metadata: name: we
b spec: selector: matchLabels: app: nginx # has to match .spec.template.metada
ta.labels serviceName: "nginx" replicas: 3 # by default is 1 template: metadat
a: labels: app: nginx # has to match .spec.selector.matchLabels spec: terminat
ionGracePeriodSeconds: 10 containers: - name: nginx image: k8s.gcr.io/nginx-sl
im:0.8 ports: - containerPort: 80 name: web volumeMounts: - name: www mountPat
h: /usr/share/nginx/html volumeClaimTemplates: - metadata: name: www spec: acc
essModes: [ "ReadWriteOnce" ] storageClassName: "rook-ceph-block" resources: r
equests: storage: 1Gi
```

스케일 명령을 사용해 파드를 늘리거나 축소해보자.

```
kubectl scale statefulset web --replicas=5 kubectl scale statefulset web --
replicas=1
```

디플로이먼트 업데이트

업데이트 준비

v1 디플로이먼트와 서비스를 준비한다.

```
kubectl delete all --all --force kubectl create deploy http-go --image=gasbug
s/http-go:v1 --dry-run=client -oyaml > http-go-v1-deploy.yaml kubectl apply -f
http-go-v1-deploy.yaml --record=true kubectl expose deploy http-go --type=Node
Port --port=80 --target-port=8080 kubectl scale deploy http-go --replicas=3
```

모니터링 스크립트를 구성한다. 포트와 IP를 본인 것을 확인하고 적용하도록 한다.

```
while true; do curl 172.30.6.7:31129; sleep 1; done;
```

업데이트 속도 조절

```
kubectl patch deployment http-go -p '{"spec": {"minReadySeconds": 10}}'
```

업데이트 시작

set image 명령으로 업데이트가 가능하다. v2로 업데이트를 실행한다.

```
kubectl set image <resource_type> <resource_name> <container_name>=<image_name
> kubectl set image deploy http-go http-go=gasbugs/http-go:v2 --record=true
```

yaml 파일을 사용해 v3로 업데이트를 진행한다. http-go-v1-deploy.yaml 복제해 http-go-v3-deploy.yaml를 생성하고 v3로 이미지 버전을 변경한다. apply 명령을 사용해 업데이트할 수 있다.

```
cp http-go-v1-deploy.yaml http-go-v3-deploy.yaml vim http-go-v3-deploy.yaml # replicas의 개수를 3개로 설정 # image에서 v3로 변경 kubectl apply -f http-go-v3-deploy.yaml --record=true
```

업데이트 히스토리 확인하기

```
$ kubectl rollout history deploy http-go deployment.apps/http-go REVISION CHANGE-CAUSE 1 kubectl apply --filename=http-go-v1-deploy.yaml --record=true 2 kubectl set image deploy http-go http-go=gasbugs/http-go:v2 --record=true 3 kubectl apply --filename=http-go-v3-deploy.yaml --record=true $ kubectl get rs NAME DESIRED CURRENT READY AGE http-go-5bcb89d449 0 0 0 13m http-go-7887c68b47 3 3 3 2m45s http-go-7dfd967844 0 0 0 6m37s
```

롤백 실행하기

이전 버전으로 롤백 수행하기

```
kubectl rollout undo deploy http-go
```

특정 리비전으로 롤백 수행하기

```
kubectl rollout undo deploy http-go --to-revision=1
```

연습문제

다음 mongo 이미지를 사용하여 업데이트와 롤백을 실행하라.

1. 모든 revision 내용은 기록돼야 한다.

```
--record=true
```

2. mongo:4.2 이미지를 사용하여 deployment를 생성하라.
    - Replicas: 10
    - maxSurge: 50%
    - maxUnavailable: 50%

▼ 풀이

```
# mongo-4.2-deploy.yaml # kubectl apply -f mongo-4.2-deploy.yaml --
record=true apiVersion: apps/v1 kind: Deployment metadata: name: mongo
labels: app: mongo spec: strategy: rollingUpdate: maxSurge: 50%
maxUnavailable: 50% type: RollingUpdate replicas: 10 selector:
matchLabels: app: mongo template: metadata: labels: app: mongo spec:
containers: - name: mongo image: mongo:4.2
```

생성 후 현재 상황을 확인한다.

```
$ kubectl get deploy,pod NAME READY UP-TO-DATE AVAILABLE AGE
deployment.apps/mongo 10/10 10 10 7s NAME READY STATUS RESTARTS AGE
pod/mongo-6866645f8d-4r5hf 1/1 Running 0 6s pod/mongo-6866645f8d-6fgtk 1/1
Running 0 7s pod/mongo-6866645f8d-b679r 1/1 Running 0 6s pod/mongo-
6866645f8d-chxsx 1/1 Running 0 7s pod/mongo-6866645f8d-frnbt 1/1 Running 0
6s pod/mongo-6866645f8d-hxnf7 1/1 Running 0 6s pod/mongo-6866645f8d-lqb8z
1/1 Running 0 6s pod/mongo-6866645f8d-nr4gn 1/1 Running 0 7s pod/mongo-
6866645f8d-r8xxw 1/1 Running 0 7s pod/mongo-6866645f8d-ts62w 1/1 Running 0
6s
```

1. mongo:4.4 롤링 업데이트를 수행하라.

   ```
   kubectl set image deploy mongo mongo=mongo:4.4 --record=true
   ```

   포드들의 이미지 정보를 확인해본다.

   ```
   kubectl get pod -ojsonpath='{..image}'
   ```

2. mongo:4.2로 롤백을 수행하라.

   ```
   kubectl rollout undo deploy mongo
   ```