<https://blog.couchbase.com/stateful-containers-kubernetes-amazon-ebs/>

<https://blog.couchbase.com/kubernetes-cluster-amazon-expose-service/>

<https://github.com/kubernetes-incubator/external-storage/tree/master/aws/efs>

<https://www.juandebravo.com/2018/09/28/aws-efs-kubernetes/>

https://containerjournal.com/topics/container-networking/using-ebs-and-efs-as-persistent-volume-in-kubernetes/

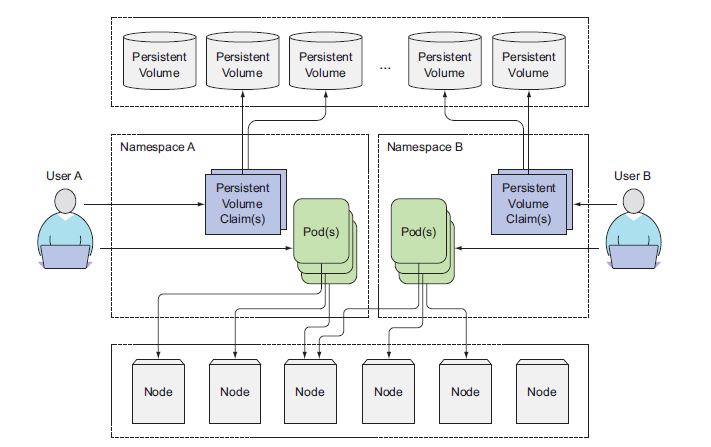
<https://medium.com/@while1eq1/using-amazon-efs-in-a-multiaz-kubernetes-setup-57922e032776>

<https://docs.giantswarm.io/guides/using-persistent-volumes-on-aws-with-efs/>

<https://github.com/kubernetes-incubator/external-storage/blob/master/aws/efs/deploy/manifest.yaml>

<https://rancher.com/running-highly-available-wordpress-mysql-kubernetes>

<https://medium.com/swlh/stupid-simple-kubernetes-persistent-volumes-explained-by-examples-29f8fec08c4>



===================EMPTY-DIR===========================================

apiVersion: v1

kind: Pod

metadata:

name: mypod

spec:

containers:

- name: mypod1

image: sreeharshav/rollingupdate:v1

volumeMounts:

- name: testvol

mountPath: /tmp/testvolume

volumes:

- name: testvol

emptyDir: {}

=================================================================================

apiVersion: v1

kind: Pod

metadata:

name: mypod

spec:

containers:

- name: mypod1

image: sreeharshav/rollingupdate:v1

volumeMounts:

- name: testvol

mountPath: /usr/share/nginx/html

volumes:

- name: testvol

gitRepo:

repository: https://github.com/mavrick202/dockertest1.git

revision: feature3

directory: .

====================================================================================

apiVersion: v1

kind: Pod

metadata:

name: gitrepo-volume-pod

labels:

name: gitrepo-volume-pod

spec:

containers:

- image: sreeharshav/rollingupdate:v3

name: web-server

ports:

- containerPort: 80

protocol: TCP

volumeMounts:

- name: html

mountPath: /usr/share/nginx/

- name: git-sync

image: k8s.gcr.io/git-sync:v3.0.1

volumeMounts:

- name: html

mountPath: /tmp/git/

env:

- name: GIT\_SYNC\_REPO

value: https://github.com/mavrick202/dockertest1.git

- name: GIT\_SYNC\_BRANCH

value: feature3

- name: GIT\_SYNC\_DEST

value: html

volumes:

- name: html

emptyDir: {}

EBS Tags => KubernetesCluster: sreek8s.xyz

<https://stackoverflow.com/questions/47278433/need-help-on-volume-mount-issue-with-kubernetes>

==================POD-RAW-EBS-MAPPING========================

apiVersion: v1

kind: Pod

metadata:

name: test-ebs

labels:

name: raw-volume-testing

spec:

containers:

- image: nginx:latest

name: test-container

volumeMounts:

- mountPath: /usr/share/nginx/html/

name: test-volume

volumes:

- name: test-volume

# This AWS EBS volume must already exist.

awsElasticBlockStore:

volumeID: vol-0cda9e7f423949407

fsType: ext4

=================DEPLOYMENT-WITH-RAW-DEVICE=================

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-deployment

labels:

app: mongodb

spec:

replicas: 1

selector:

matchLabels:

app: mongodb

name: mongodb

template:

metadata:

labels:

app: mongodb

name: mongodb

spec:

containers:

- image: mongo

name: mongodb

imagePullPolicy: Always

volumeMounts:

- name: mongodb-data

mountPath: /data/db

volumes:

- name: mongodb-data

awsElasticBlockStore:

volumeID: vol-0cfc58a58832bc418

fsType: ext4

EBS is a persistent block storage with a defined size (It is however possible to change it later).

It is fully configurable and could be the most performant solution of Amazon.

The major drawback is that it can be only mounted to one EC2 instance.

This also means that it can't be shared between multiple pods.

https://kubernetes.io/docs/concepts/storage/persistent-volumes/

https://blog.abyssale.com/shared-storage-volume-on-amazon/

PV & PVC:

* RWO - ReadWriteOnce
* ROX - ReadOnlyMany
* RWX - ReadWriteMany

The access modes are:

* ReadWriteOnce -- the volume can be mounted as read-write by a single node
* ReadOnlyMany -- the volume can be mounted read-only by many nodes
* ReadWriteMany -- the volume can be mounted as read-write by many nodes

A volume will be in one of the following phases:

* Available -- a free resource that is not yet bound to a claim
* Bound -- the volume is bound to a claim
* Released -- the claim has been deleted, but the resource is not yet reclaimed by the cluster
* Failed -- the volume has failed its automatic reclamation

----------------PV---------------------

---

apiVersion: v1

kind: PersistentVolume

metadata:

name: aws-pv1

labels:

type: aws-pv1

spec:

storageClassName: gp2

persistentVolumeReclaimPolicy: Delete

capacity:

storage: 2Gi

accessModes:

- ReadWriteOnce

awsElasticBlockStore:

volumeID: vol-04180df2e5277b4ea

fsType: ext4

---

apiVersion: v1

kind: PersistentVolume

metadata:

name: aws-pv2

labels:

type: aws-pv2

spec:

storageClassName: gp2

persistentVolumeReclaimPolicy: Delete

capacity:

storage: 2Gi

accessModes:

- ReadWriteOnce

awsElasticBlockStore:

volumeID: vol-04180df2e5277b4ea

fsType: ext4

---

apiVersion: v1

kind: PersistentVolume

metadata:

name: aws-pv3

labels:

type: aws-pv3

spec:

storageClassName: gp2

persistentVolumeReclaimPolicy: Delete

capacity:

storage: 2Gi

accessModes:

- ReadWriteOnce

awsElasticBlockStore:

volumeID: vol-04180df2e5277b4ea

fsType: ext4

---

apiVersion: v1

kind: PersistentVolume

metadata:

name: aws-pv4

labels:

type: aws-pv4

spec:

storageClassName: gp2

persistentVolumeReclaimPolicy: Delete

capacity:

storage: 2Gi

accessModes:

- ReadWriteOnce

awsElasticBlockStore:

volumeID: vol-04180df2e5277b4ea

fsType: ext4

---

apiVersion: v1

kind: PersistentVolume

metadata:

name: aws-pv5

labels:

type: aws-pv5

spec:

storageClassName: gp2

persistentVolumeReclaimPolicy: Delete

capacity:

storage: 2Gi

accessModes:

- ReadWriteOnce

awsElasticBlockStore:

volumeID: vol-04180df2e5277b4ea

fsType: ext4

------------PVC----------------------

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: task-pv-claim1

spec:

storageClassName: gp2

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 2Gi

---

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: task-pv-claim2

spec:

storageClassName: gp2

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 2Gi

---

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: task-pv-claim3

spec:

storageClassName: gp2

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 2Gi

---

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: task-pv-claim4

spec:

storageClassName: gp2

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 2Gi

---

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: task-pv-claim5

spec:

storageClassName: gp2

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 2Gi

----------------PV---------------------

apiVersion: v1

kind: PersistentVolume

metadata:

name: aws-pv1

labels:

type: aws-pv1

spec:

storageClassName: gp2

persistentVolumeReclaimPolicy: Delete

capacity:

storage: 2Gi

accessModes:

- ReadWriteOnce

awsElasticBlockStore:

volumeID: vol-0eb77fa6ddf3c1d79

fsType: ext4

---

apiVersion: v1

kind: PersistentVolume

metadata:

name: aws-pv2

labels:

type: aws-pv2

spec:

storageClassName: gp2

persistentVolumeReclaimPolicy: Delete

capacity:

storage: 2Gi

accessModes:

- ReadWriteOnce

awsElasticBlockStore:

volumeID: vol-0eb77fa6ddf3c1d79

fsType: ext4

---

apiVersion: v1

kind: PersistentVolume

metadata:

name: aws-pv3

labels:

type: aws-pv3

spec:

storageClassName: gp2

persistentVolumeReclaimPolicy: Delete

capacity:

storage: 2Gi

accessModes:

- ReadWriteOnce

awsElasticBlockStore:

volumeID: vol-0eb77fa6ddf3c1d79

fsType: ext4

---

apiVersion: v1

kind: PersistentVolume

metadata:

name: aws-pv4

labels:

type: aws-pv4

spec:

storageClassName: gp2

persistentVolumeReclaimPolicy: Delete

capacity:

storage: 2Gi

accessModes:

- ReadWriteOnce

awsElasticBlockStore:

volumeID: vol-0eb77fa6ddf3c1d79

fsType: ext4

---

apiVersion: v1

kind: PersistentVolume

metadata:

name: aws-pv5

labels:

type: aws-pv5

spec:

storageClassName: gp2

persistentVolumeReclaimPolicy: Delete

capacity:

storage: 2Gi

accessModes:

- ReadWriteOnce

awsElasticBlockStore:

volumeID: vol-0eb77fa6ddf3c1d79

fsType: ext4

#------------PVC----------------------

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: task-pv-claim1

spec:

storageClassName: gp2

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 2Gi

---

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: task-pv-claim2

spec:

storageClassName: gp2

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 2Gi

---

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: task-pv-claim3

spec:

storageClassName: gp2

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 2Gi

---

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: task-pv-claim4

spec:

storageClassName: gp2

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 2Gi

---

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: task-pv-claim5

spec:

storageClassName: gp2

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 2Gi

------------POD-WITH-PVC--------------------------

apiVersion: v1

kind: Pod

metadata:

name: task-pv-pod

spec:

volumes:

- name: task-pv-storage

persistentVolumeClaim:

claimName: task-pv-claim1

containers:

- name: task-pv-container

image: nginx

ports:

- containerPort: 80

name: "http-server"

volumeMounts:

- mountPath: "/usr/share/nginx/html"

name: task-pv-storage

Reclaiming

When a user is done with their volume, they can delete the PVC objects from the API that allows reclamation of the resource. The reclaim policy for a PersistentVolume tells the cluster what to do with the volume after it has been released of its claim. Currently, volumes can either be Retained, Recycled, or Deleted.

Retain

The Retain reclaim policy allows for manual reclamation of the resource. When the PersistentVolumeClaim is deleted, the PersistentVolume still exists and the volume is considered "released". But it is not yet available for another claim because the previous claimant's data remains on the volume.

kubectl patch pv aws-pv5 -p '{"spec":{"claimRef": null}}'

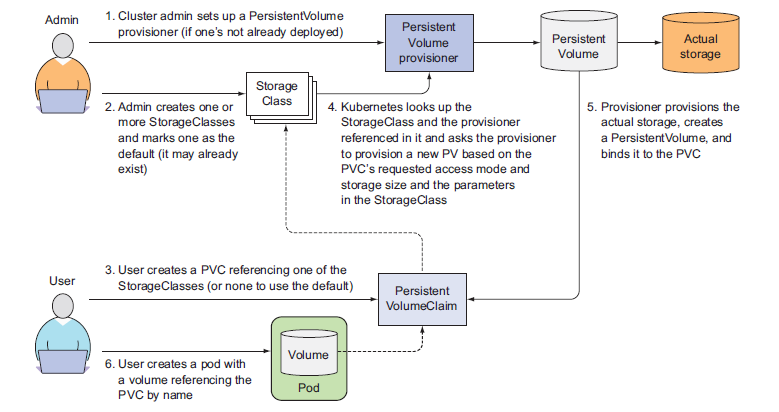
Delete

For volume plugins that support the Delete reclaim policy, deletion removes both the PersistentVolume object from Kubernetes, as well as the associated storage asset in the external infrastructure, such as an AWS EBS, GCE PD, Azure Disk, or Cinder volume. Volumes that were dynamically provisioned inherit the reclaim policy of their StorageClass, which defaults to Delete.

Recycle

Warning: The Recycle reclaim policy is deprecated. Instead, the recommended approach is to use dynamic provisioning.

DYNAMIC-STORAGE-PROVISIONING



===============DYNAMIC-PROVISIONING====================

---

kind: StorageClass

apiVersion: storage.k8s.io/v1beta1

metadata:

name: slow

provisioner: kubernetes.io/aws-ebs

parameters:

type: io1

#zone: us-east-1a

zones: us-east-1a, us-east-1b

iopsPerGB: "5"

---

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: auto-assign-pv-1

spec:

storageClassName: high

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 4Gi

---

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: auto-assign-pv-2

spec:

storageClassName: medium

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 1Gi