Kubernetes Essential

Agenda

- Assignment Review & Guides
- Kubernetes volume management
- Configmap and secret

Persistent in Pods

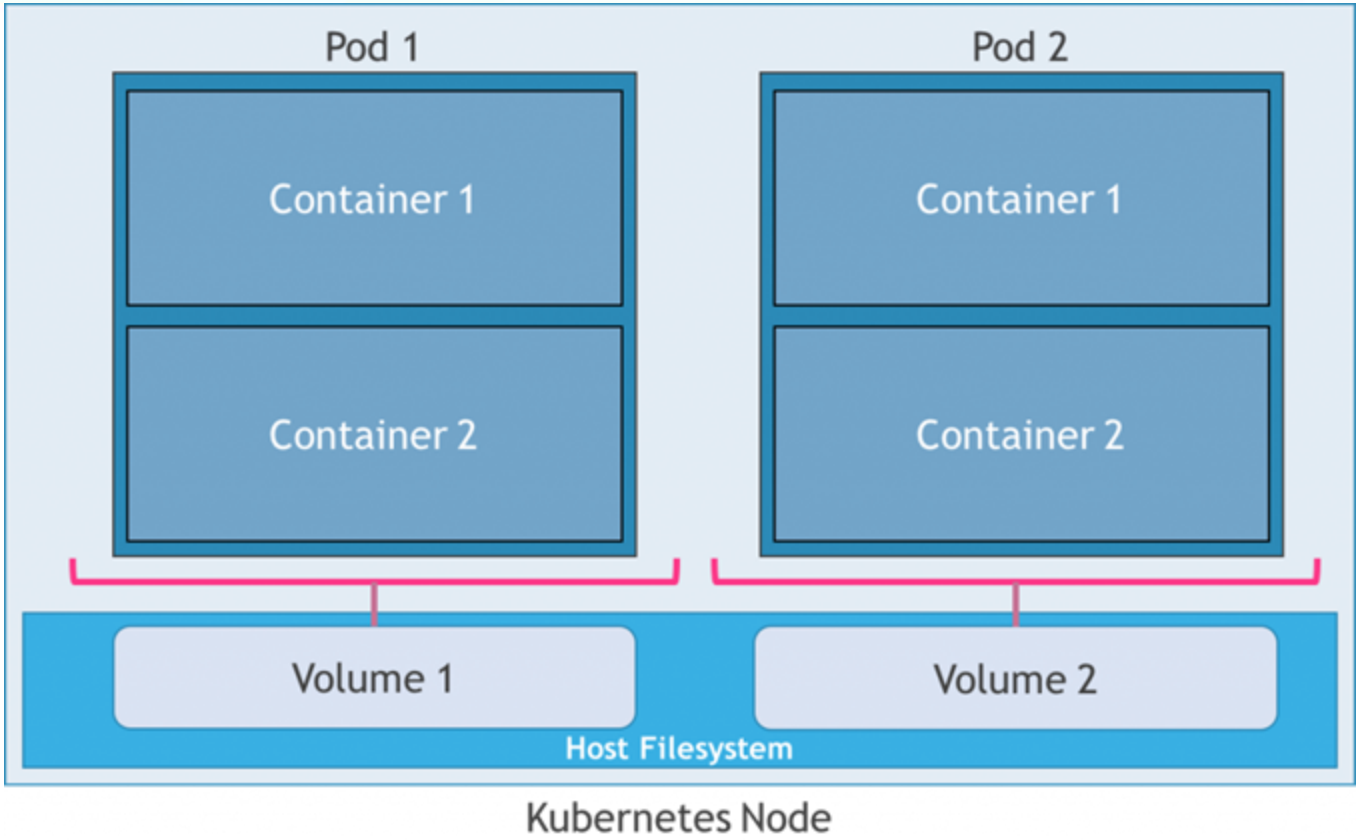
Issues:

- ***Pod*** are ephemeral and stateless: all files created or modified during a pod lifecycle is deleted when a ***pod*** crashed.
- Multiple containers in pod need to use shared files.

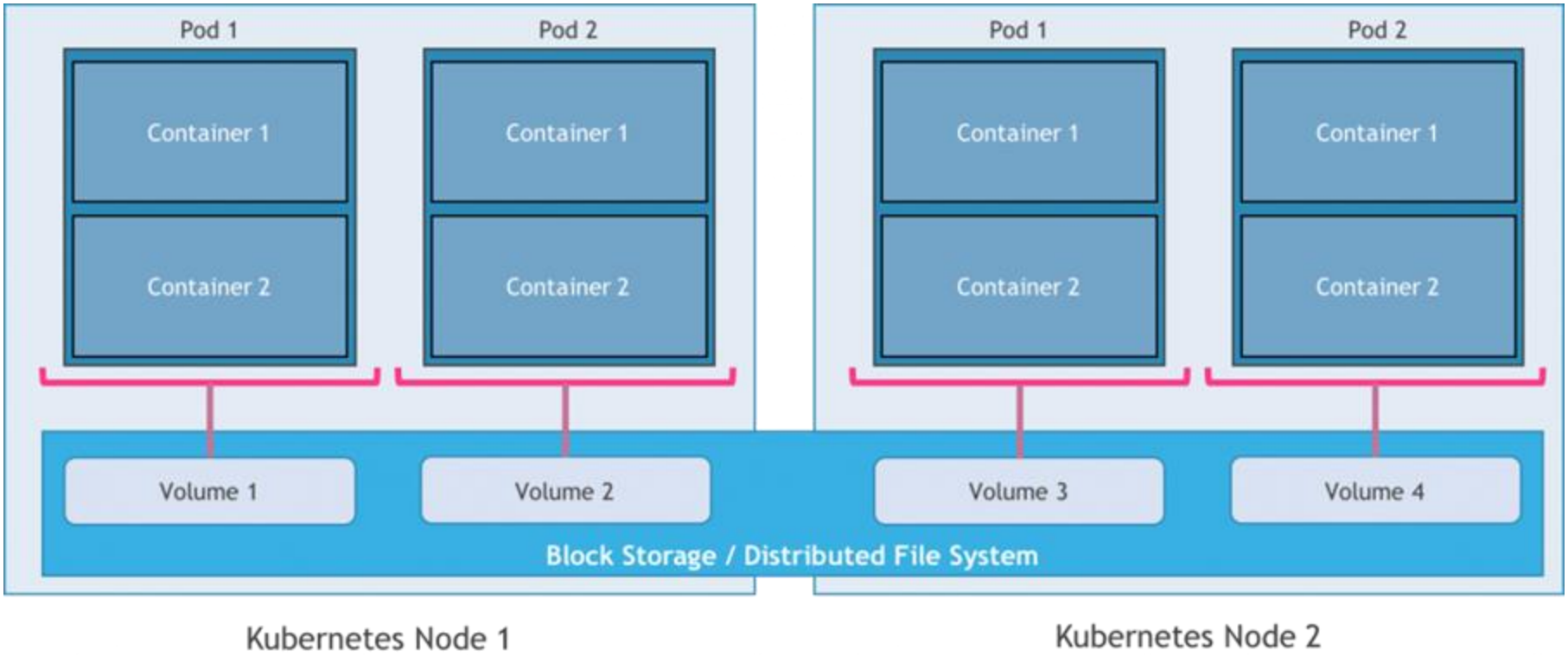
Solution: use ***Volumes***

- Kubernetes support a number of volume types, and a ***pod*** can use any number of volume types simultaneously.
- Ephemeral volume types: exists during the lifetime of a ***pod***.
- Persistent volume types: exist beyond the lifetime of a ***pod***.
- ***Volumes*** is accessible to all containers in a pod that it is mounted to.
- In all cases, container restarts do not affect to created ***volumes***.

Pod and Volumes



Pod and Volumes



Type of Volumes

Ephemeral volume types:

- emptyDir
- configmap, secret
- …

Persistent volume types:

- hostPath
- local
- NFS
- …

Volume type: emptyDir

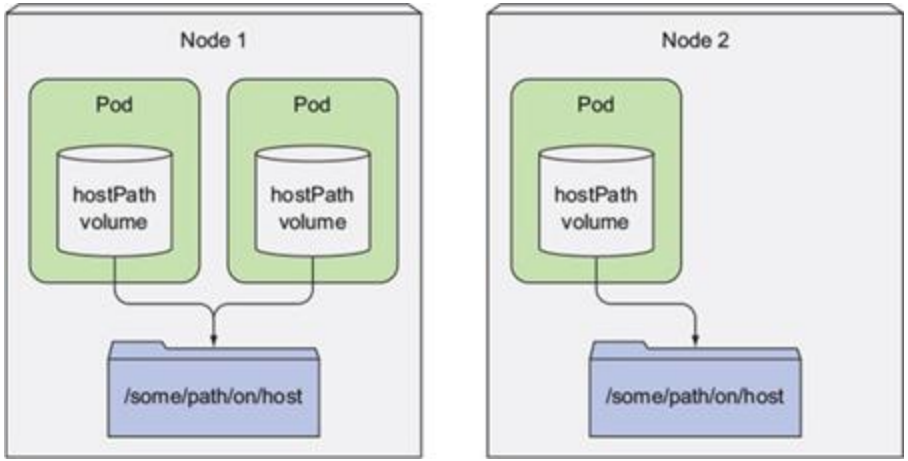
emptyDir:

- Created when a pod is assigned to a node, and exists as long as that pod is running on that node.
- Initially empty.
- When a Pod is removed from a node for any reason, the data in the emptyDir is deleted permanently.

Note: A container crashing does *not* remove a Pod from a node. The data in an `emptyDir` volume is safe across container crashes.

Volume type: hostPath

A *hostPath* volume mounts a file or directory from the host node's filesystem into the pod.



StorageClass

- A *StorageClass* is an object that defines the type of storage that can be dynamically provisioned by the cluster.
- Different classes might map to quality-of-service levels, or to backup policies, or to arbitrary policies determined by the cluster administrators

StorageClass

A *StorageClass* defines several attributes, including:

- Provisioner: The name of the storage provider or backend that will be used to provision the storage.
- Parameters: Any additional configuration parameters that are required by the provisioner.
- Reclaim Policy: How the storage should be reclaimed when it is no longer needed. This can be set to retain the data, delete the data, or archive the data.

StorageClass

There are a variety of provisioners:

- AWSElasticBlockStore
- GCEPersistentDisk
- AzureDisk
- NFS
- …

PersistentVolume

- A ***PersistentVolume*** **(PV)** is a piece of storage in the cluster that has been provisioned by an administrator or dynamically provisioned using Storage Classes.
- ***PersistentVolume*** has a lifecycle independent of any individual Pod that uses that ***PersistentVolume***.

PersistentVolume

- A **PersistentVolume** (PV) represents a storage resource.
- PVs are a **cluster wide resource** linked to a backing storage provider: NFS, EBS, etc.
- Their lifecycle is handled independently from a pod
- **CANNOT** be attached to a Pod directly. Relies on a **PersistentVolumeClaim** (PVC).

PersistentVolumeClaim

- A ***PersistentVolumeClaim*** (PVC) is a request for storage by a user.
- Claims can request specific size and access modes: *ReadWriteOnce*, *ReadOnlyMany* or *ReadWriteMany*.

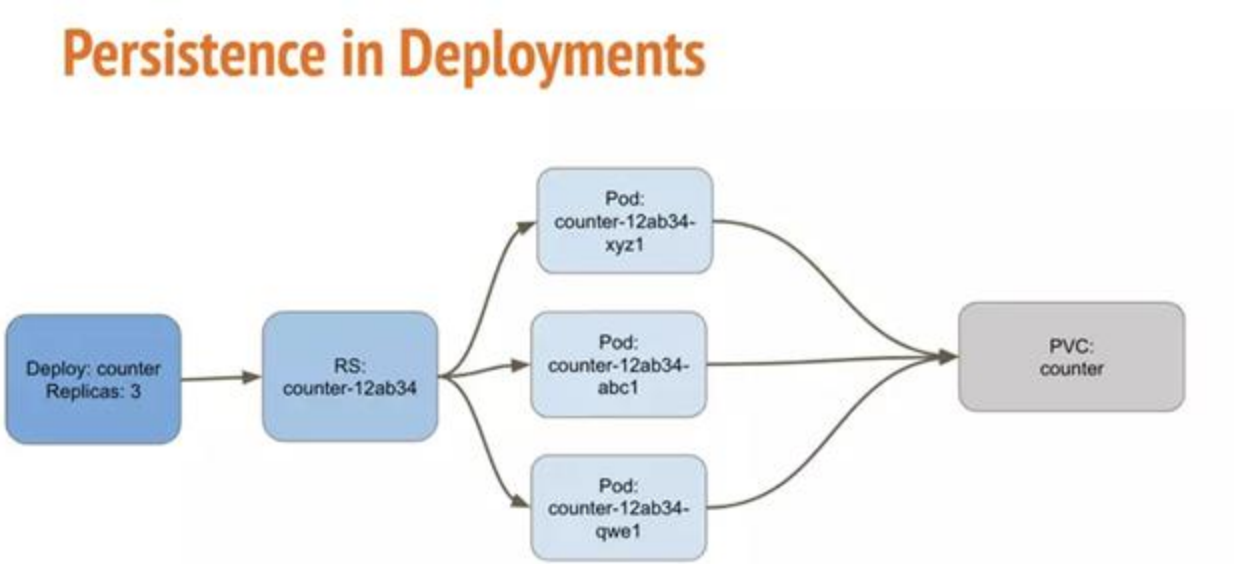
PersistentVolumeClaim

PersistenVolumeClaim can be statically or dynamically provisioned:

- **Static**: A cluster administrator creates a number of PVs. They carry the details of the real storage, which is available for use by cluster users. They exist in the Kubernetes API and are available for consumption.
- **Dynamic**: When none of the static PVs the administrator created match a user's PersistentVolumeClaim, the cluster may try to dynamically provision a volume specially for the PVC. This provisioning is based on StorageClasses

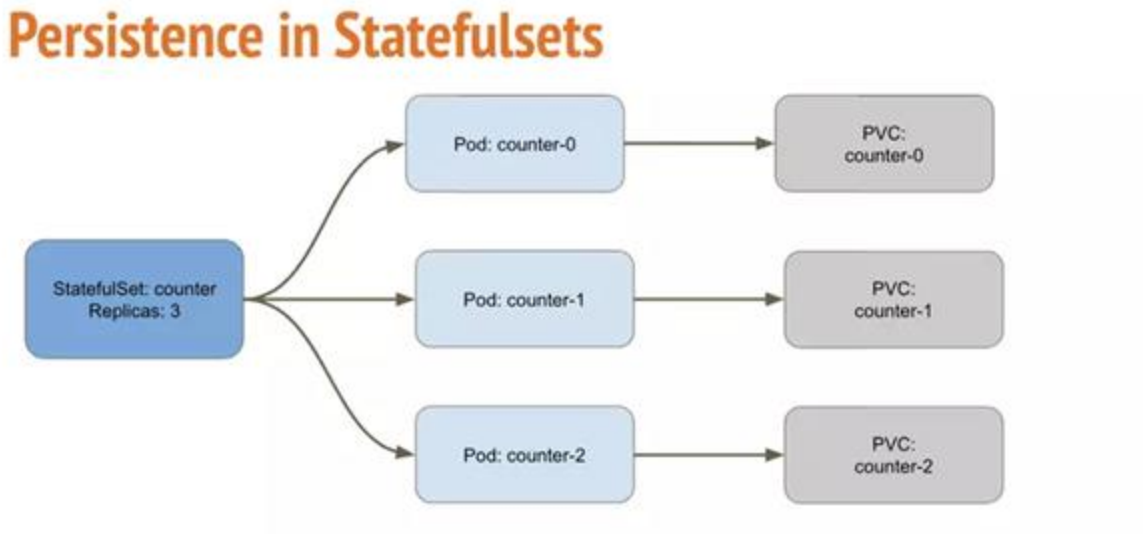
StatefulSet

Issues with ***ReplicaSet***: Each pod can not have it's own state or datastore.



StatefulSet

Solution: ***StatefulSet,*** each pod will have it's own volume.



CONFIGURATION

ConfigMaps

- An API object used to store non-confidential data in key-value pairs
- Can be referenced through several different means:
  - environment variable
  - a command line argument
  - injected as a configuration file into a volume mount
- Can be created from a manifest, literals, directories, or files directly.

```
apiVersion: v1
kind: ConfigMap
metadata:
 name: my-configmap
data:
 server: localhost
 name: bastion
```

Secrets

- Functionally identical to a ConfigMap.
- Used to stored credentials as base64 encoded contents.

```
---
apiVersion: v1
kind: Secret
metadata:
 name: my-secret
type: Opaque
data:
 username: dXNlcm5hbWU=
 password: cGFzc3dvcmQ=
```

Q&A