ESCOLA SUPERIOR DE REDES

# Orquestração de Containers

## Armazenamento no Kubernetes

# Tópicos abordados

- Persistência de dados
- Volumes
- *PersistentVolumes e PersistentVolumeClaims*
- *StorageClasses*

# Persistência de dados

Por padrão, dados armazenados em *pods* são efêmeros

Em alguns cenários isso não é um problema, como em serviços de *cache* ou configuração *read-only*

Em outros, contudo, não ter os dados persistidos após um *crash* pode inviabilizar a aplicação

Pense, por exemplo, em um banco de dados ou um *file server*

# Volumes

Usados para persistir dados entre reinícios de um *pod*, ou compartilhamento de arquivos entre *pods*

Análogo ao conceito de *volumes* no Docker, porém com mais funcionalidades

Essencialmente, um *volume* pode ser entendido como um diretório

Como esse diretório é criado, em que meio físico, e seu conteúdo são determinados por seu *tipo*

# Tipos de volumes

awsElasticBlockStore
azureDisk
azureFile
cephfs
cinder
configMap
downwardAPI
emptyDir
fc (fibre channel)
flocker (deprecated)
gcePersistentDisk
gitRepo (deprecated)
glusterfs
hostPath
iscsi
local
nfs
persistentVolumeClaim
portworxVolume
projected
quobyte
rbd
scaleIO (deprecated)
secret
storageOS
vsphereVolume

# Um exemplo simples: *HostPath*

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: test-pd
spec:
  containers:
  - image: k8s.gcr.io/test-webserver
    name: test-container
    volumeMounts:
    - mountPath: /test-pd
      name: test-volume
  volumes:
  - name: test-volume
    hostPath:
      # directory location on host
      path: /data
      # this field is optional
      type: Directory
```

# Vamos testar?

Onde são criados os arquivos num volume *HostPath*?
Essa solução possui escalabilidade?

# *PersistentVolumes*

Provê uma API para usuários e administradores que abstrai provisionamento e consumo de *storage*

O ciclo de vida de um PV é independente do *pod* usuário, ao contrário de volumes

O objeto captura os detalhes de implementação, como NFS, iSCSI ou *cloud-based*

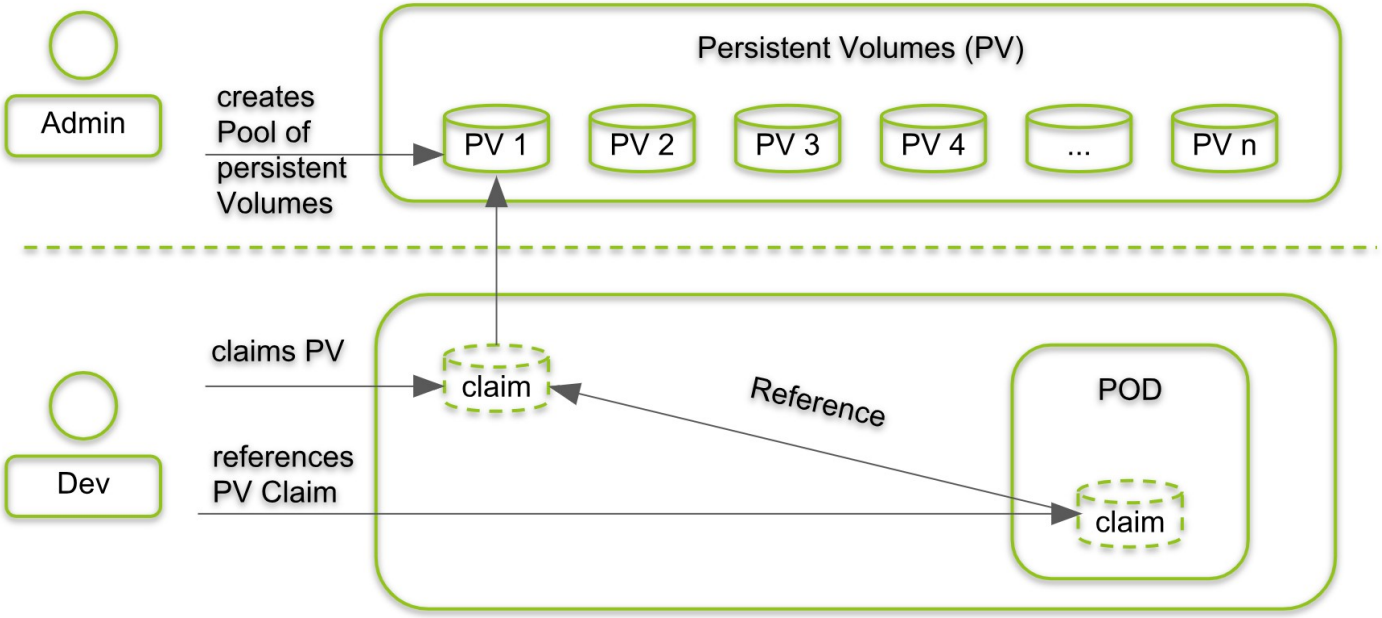
# *PersistentVolumeClaims*

Trata-se de um requerimento de *storage* por um usuário, similar a um *pod*

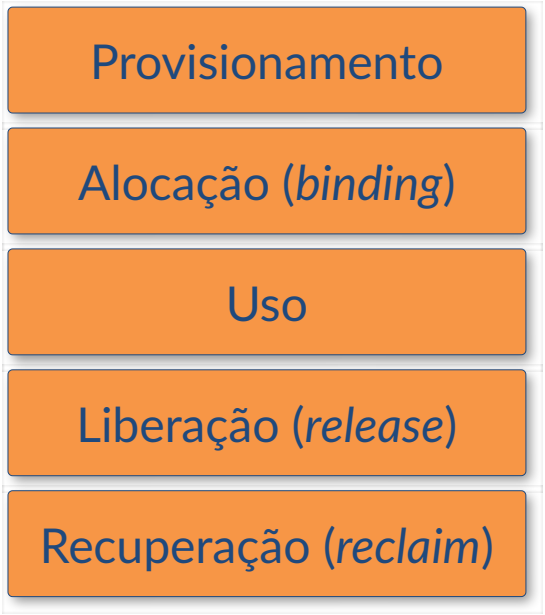
*Pods* consomem recursos de *nodes*, como CPU e RAM – PVCs, por outro lado, consomem recursos de PVs

PVCs podem especificar o tamanho do volume e o modo de acesso

# Exemplo esquemático



# Ciclo de vida de PVs e PVCs



# PVs: tipos de provisionamento

Estático

Dinâmico

# PVs: *volume modes*

*Filesystem*

*Block*

# PVs: modos de acesso

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

In the CLI, the access modes are abbreviated to:

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

# PVs: *reclaim policies*

- Retain -- manual reclamation
- Recycle -- basic scrub ( `rm -rf /thevolume/*` )
- Delete -- associated storage asset such as AWS EBS, GCE PD, Azure Disk, or OpenStack Cinder volume is deleted

# Tipos de PVs

- `awsElasticBlockStore` - AWS Elastic Block Store (EBS)
- `azureDisk` - Azure Disk
- `azureFile` - Azure File
- `cephfs` - CephFS volume
- `cinder` - Cinder (OpenStack block storage) (**deprecated**)
- `csi` - Container Storage Interface (CSI)
- `fc` - Fibre Channel (FC) storage
- `flexVolume` - FlexVolume
- `flocker` - Flocker storage
- `gcePersistentDisk` - GCE Persistent Disk
- `glusterfs` - Glusterfs volume

# Tipos de PVs

- `hostPath` - HostPath volume (for single node testing only; WILL NOT WORK in a multi-node cluster; consider using `local` volume instead)
- `iscsi` - iSCSI (SCSI over IP) storage
- `local` - local storage devices mounted on nodes.
- `nfs` - Network File System (NFS) storage
- `photonPersistentDisk` - Photon controller persistent disk. (This volume type no longer works since the removal of the corresponding cloud provider.)
- `portworxVolume` - Portworx volume
- `quobyte` - Quobyte volume
- `rbd` - Rados Block Device (RBD) volume
- `scaleIO` - ScaleIO volume (**deprecated**)
- `storageos` - StorageOS volume
- `vsphereVolume` - vSphere VMDK volume

# Criando PVs via arquivos YAML

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv0003
spec:
  capacity:
    storage: 5Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Recycle
  storageClassName: slow
  mountOptions:
    - hard
    - nfsvers=4.1
  nfs:
    path: /tmp
    server: 172.17.0.2
```

# Criando PVCs via arquivos YAML

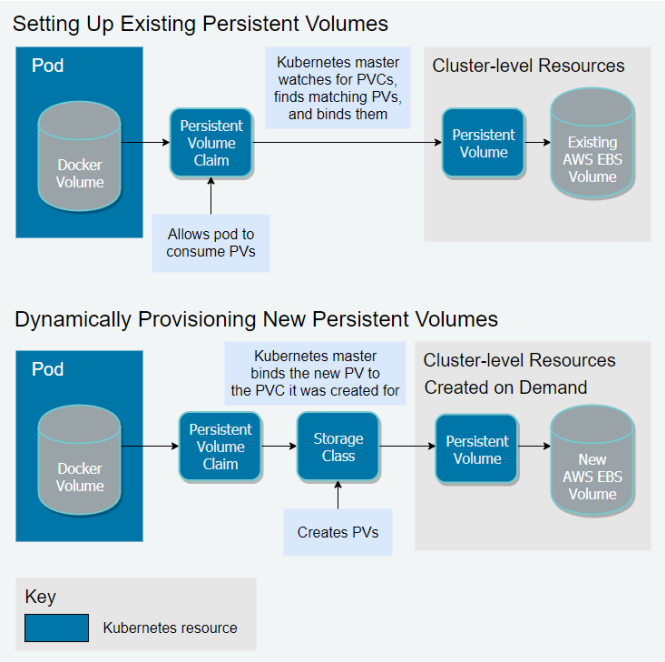
```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: myclaim
spec:
  accessModes:
    - ReadWriteOnce
  volumeMode: Filesystem
  resources:
    requests:
      storage: 8Gi
  storageClassName: slow
  selector:
    matchLabels:
      release: "stable"
    matchExpressions:
      - {key: environment, operator: In, values: [dev]}
```

Seletores podem ser usados para filtrar por volumes elegíveis para *bind*

# Consumindo PVCs

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: mypod
spec:
  containers:
    - name: myfrontend
      image: nginx
      volumeMounts:
      - mountPath: "/var/www/html"
        name: mypd
  volumes:
    - name: mypd
      persistentVolumeClaim:
        claimName: myclaim
```

# De volta ao provisionamento dinâmico...



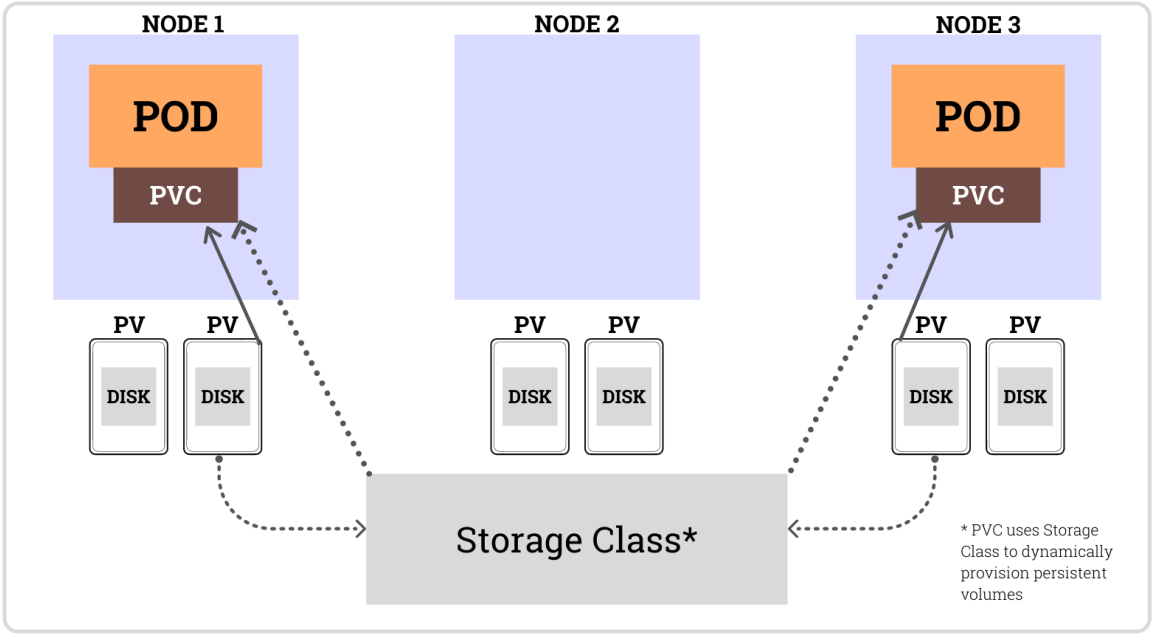
# *StorageClasses*

Provê uma maneira para administradores descreverem classes de armazenamento

Por exemplo, seria possível diferenciar armazenamento com tempo de acesso rápido ou lento

Deve-se selecionar um *plugin* de provisionamento, bem como outras opções semelhantes ao PV

# Exemplo esquemático



# *StorageClasses*: *plugins* de provisionamento

| | | |
|---|---|---|
| AWSElasticBlockStore | ✓ | AWS EBS |
| AzureFile | ✓ | Azure File |
| AzureDisk | ✓ | Azure Disk |
| CephFS | - | - |
| Cinder | ✓ | OpenStack Cinder |
| FC | - | - |
| FlexVolume | - | - |
| Flocker | ✓ | - |
| GCEPersistentDisk | ✓ | GCE PD |
| Glusterfs | ✓ | Glusterfs |
| iSCSI | - | - |
| Quobyte | ✓ | Quobyte |
| NFS | - | - |
| RBD | ✓ | Ceph RBD |
| VsphereVolume | ✓ | vSphere |
| PortworxVolume | ✓ | Portworx Volume |
| ScaleIO | ✓ | ScaleIO |
| StorageOS | ✓ | StorageOS |
| Local | - | Local |

# Criando *StorageClasses* via arquivos YAML

```yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: standard
provisioner: kubernetes.io/aws-ebs
parameters:
  type: gp2
reclaimPolicy: Retain
allowVolumeExpansion: true
mountOptions:
  - debug
volumeBindingMode: Immediate
```

# Um exemplo de *provider* dinâmico

https://github.com/kubernetes-sigs/nfs-subdir-external-provisioner

ESCOLA SUPERIOR DE REDES

# Armazenamento no Kubernetes

RNP ORGANIZAÇÃO SOCIAL DO MCTI

MINISTÉRIO DO TURISMO

MINISTÉRIO DA DEFESA

MINISTÉRIO DA SAÚDE

MINISTÉRIO DA EDUCAÇÃO

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÕES

PÁTRIA AMADA BRASIL GOVERNO FEDERAL