Lesson 5: Kubernetes Volumes and Storage

Lữ Thanh Tùng

Kubernetes Volumes

Volume là gì ?

- Volume là một thư mục có một số dữ liệu bên trong, có thể truy cập được vào các container trong pod

Tại sao phải cần Volume?

- Đối với container, những thứ ta ghi vào filesystem của nó thì chỉ tồn tại khi container còn chạy.
  Khi một Pod bị xóa và tạo lại, container mới sẽ được tao ra, lúc này các file lưu trữ ở container trước sẽ bị mất đi.
- Khi chạy nhiều container trong cùng một Pod, đôi khi ta cần các container khác nhau có thể truy cập vào cùng một folder để ghi dữ liệu, và sẽ có những container khác truy cập vào folder đó để lấy dữ liệu ra xử lý.

$\Rightarrow$**Kubernetes Volume giải quyết cả hai vấn đề trên.**

- **Volume có các dạng chính:**
  - Volume dùng để chia sẻ dữ liệu giữa các container trong Pod
  - Volume đính kèm vào trong filesystem một node
  - Volume đính kèm vào cluster và các node khác nhau có thể truy cập

Các loại Volume

1. **emptyDir**
   - Được tạo lần đầu tiên khi một Pod được gán cho một nút và tồn tại trong lifecycle của Pod
   - Khởi tạo một thư mục rỗng bên trong Pod, và các container trong một Pod có thể ghi và đọc dữ liệu từ nó.
2. **HostPath**
   - hostPath là loại volume sẽ tạo một mount point từ Pod ra ngoài filesystem của node.
   - Dùng để lưu trữ persistent data do dữ liệu lưu trong volume này chỉ tồn tại trên một worker node và sẽ không bị xóa đi khi Pod bị xóa
3. **gcePersistentDisk , awsElasticBlockStore, azureDiskVolume (cloud stogare) :**
   - Mount Persistent Disk của cloud vào Pod. Dữ liệu vẫn tồn tại ngay cả khi Pod bị dỡ khỏi node
4. **nfs**
   - nfs volume cho phép một hệ thống NFS tồn tại (Network File System) được mount vào pod. Dữ liệu trong một nfs volume không bị xoá khi Pod bị gỡ
   - Là volume duy nhất có thể unmount.
5. **configMap, secret, downwardAPI**
6. **PersistentVolumeClaim**

PersistentVolumes (PVs) and PersistentVolumeClaims (PVCs)

1. Persistent Storage

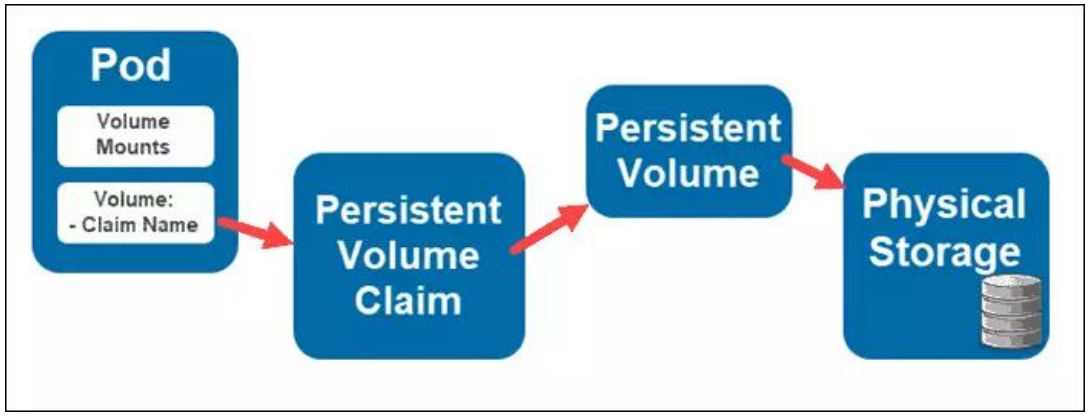
- Ephemeral Storage và Persistent Storage:
  - Ephemeral Storage (lưu trữ tạm thời) có thời gian tồn tại là thời gian tồn tại của pod và nó có thể bị mất đi sau khi pod bị xoá hoặc gặp sự cố, vô tình bị tắt
  - Persistent Storage (lưu trữ liên tục) có thời gian tồn tại độc lập với pod và không bị mất đi khi pod bị xoá hay gặp sự cố.

=> Persistent Storage là điều cần thiết với các ứng dụng quan trọng cần lưu trữ ổn định, bền bỉ vượt xa pod hoặc thậm chí là node mà pod đang chạy.

- Để tạo và sử dụng Persistent Storage, Kubernetes cung cấp hai loại API resource là PersistentVolume (PV) and PersistentVolumeClaim (PVC) .
- Trên thực tế, Kubernetes hỗ trợ rất nhiều loại volume storage khác nhau và một số các volume thuần vẫn có thể có persistent storage. Tuy nhiên, PV và PVC là rất hữu ích cho việc quản lý tài nguyên lưu trữ cho các dự án lớn. Nó giúp trừu tượng đi các chi tiết cơ sở hạ tầng, có độ phức tạp cao hơn so với chỉ sử dụng volume thuần.

PersistentVolumes (PVs) and PersistentVolumeClaims (PVCs)

2. Tại sao sử dụng PVs và PVCs



- Trên thực tế, Kubernetes hỗ trợ rất nhiều loại volume storage khác nhau và một số các volume thuần vẫn có thể có persistent storage.
- Tuy nhiên, PV và PVC là rất hữu ích cho việc quản lý tài nguyên lưu trữ cho các dự án lớn. Nó giúp trừu tượng đi các chi tiết cơ sở hạ tầng, có độ phức tạp cao hơn so với chỉ sử dụng volume thuần. Và mang lại một số lợi ích:
  - **Di chuyển ứng dụng dễ dàng giữa các môi trường**: Tái sử dụng PVC và liên kết chúng với PV đã cấu hình trước đó trong môi trường mới mà không cần sửa đổi ứng dụng hay cấu hình lưu trữ.
  - **Quản lý dữ liệu độc lập**: Cho phép thay đổi các thiết lập lưu trữ như dung lượng, loại lưu trữ, hoặc quyền truy cập mà không ảnh hưởng đến ứng dụng.
  - **Quản lý tài nguyên lưu trữ hiệu quả**: Dễ dàng xác định và theo dõi các PV và PVC đã được cấu hình trong cluster, và kiểm soát số lượng và dung lượng tài nguyên lưu trữ đã được sử dụng
  - **Tính bảo mật**: Thiết lập các chính sách truy cập lưu trữ theo yêu cầu của ứng dụng. Bằng cách thiết lập quyền truy cập vào PV và PVC để bảo vệ dữ liệu khỏi truy cập trái phép.
  - **Khả năng mở rộng:** Thêm mới PV và PVC để đáp ứng nhu cầu lưu trữ của ứng dụng mà không cần thay đổi cấu hình ứng dụng hoặc ảnh hưởng đến các phần khác trong cluster.

PersistentVolumes (PVs) and PersistentVolumeClaims (PVCs)

3. Persistent Volumes

- PersistentVolume là một resource sẽ tương tác với kiến trúc storage bên dưới.
- Tồn tại hoàn toàn độc lập với bất kỳ pod nào sử dụng PV.
- Hiện tại, Kubernetes hỗ trợ rất nhiều các loại PersistentVolume khác nhau được cài đặt dưới dạng plugin như glusterfs, nfs, csi, ...

4. PersistentVolumeClaims

- PVCs yêu cầu không gian lưu trữ của PVs
- Để tạo một PVCs, sẽ tạo một manifest chỉ định số lượng, loại lớp lưu trữ (storage class), yêu cầu các mức tài nguyên CPU, bộ nhớ,...
- PVCs còn có thể xác định các chế độ quyền truy cập cụ thể vào vùng lưu trữ:
  - ReadWriteOnce: Volume chỉ được đọc –ghi bởi một node.
  - ReadOnlyMany: Volume có thể được đọc bởi nhiều node.
  - ReadWriteMany: Volume có thể được đọc-ghi bởi nhiều node.

1. Create a PersistentVolumeClaim (PVC) and mount it to a Pod to provide persistent storage.

- Tạo PV để lưu trữ
- Tạo PVC
- Tạo Pod sử dụng PVC

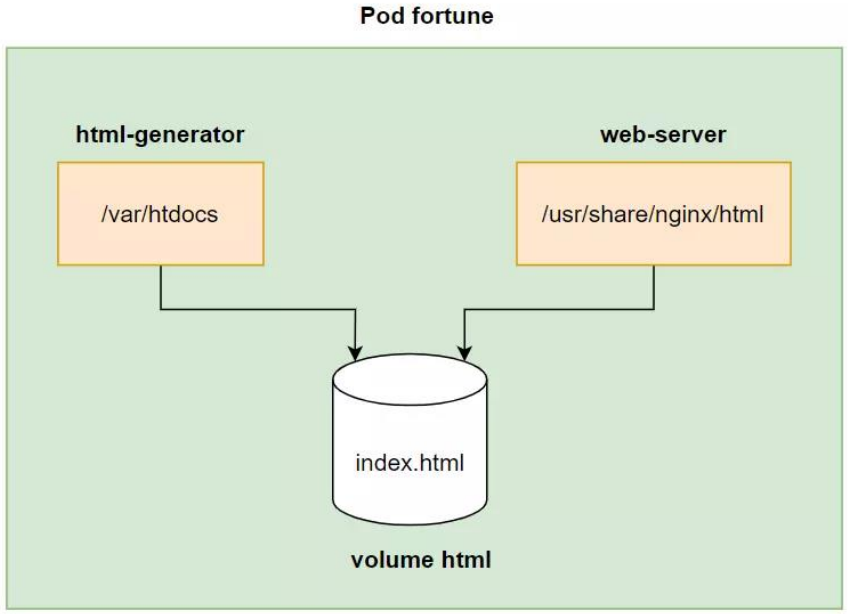
```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-lab1
  labels:
    type: local
spec:
  storageClassName: hostpath
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/data"
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod-lab1
spec:
  containers:
  - name: task-pv-container
    image: nginx
    ports:
      - containerPort: 80
        name: "http-server"
    volumeMounts:
      - mountPath: "/usr/share/ngnix"
        name: lab-storage
  volumes:
    - name: lab-storage
      persistentVolumeClaim:
        claimName: pvc-lab1
```

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-lab1
spec:
  storageClassName: hostpath
  accessModes:
    - ReadWriteOnce
  resources:
      requests:
        storage: 2Gi
```

2. Experiment with different types of Volumes - EmptyDir

- Ta sẽ tạo 2 container, 1 cái ghi dữ liệu vào volume, cái còn lại đọc dữ liệu từ đó




```
#!/bin/bash
trap "exit" SIGINT
mkdir /var/htdocs

while :
do
  echo $(date) Writing fortune to /var/htdocs/index.html
  /usr/games/fortune > /var/htdocs/index.html
  sleep 10
done
```

Code của script trong container luksa/fortune

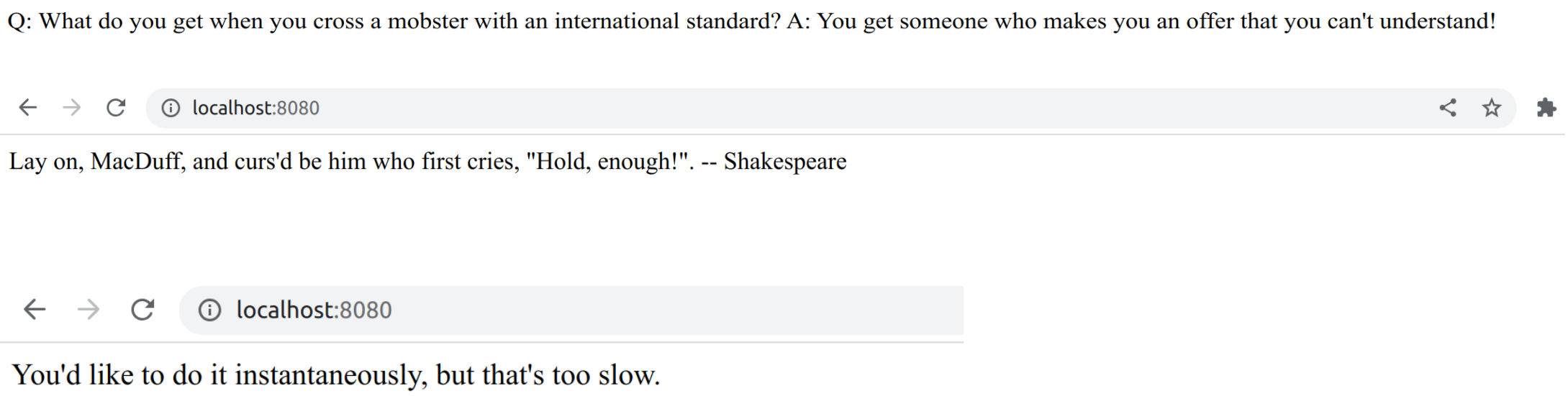
2. Experiment with different types of Volumes - EmptyDir

- Ta sẽ tạo 2 container, 1 cái ghi dữ liệu vào volume, cái còn lại đọc dữ liệu từ đó

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: fortune
spec:
  containers:
    - name: html-generator
      image: luksa/fortune
      volumeMounts:
        - name: html # The volume called html is mounted at /var/htdocs in the container
          mountPath: /var/htdocs
    - name: web-server
      image: nginx:alpine
      ports:
        - containerPort: 80
          protocol: TCP
      volumeMounts:
        - name: html # The volume called html is mounted at /usr/share/nginx/html in the container
          mountPath: /usr/share/nginx/html
          readOnly: true
  volumes: # define volumes
    - name: html # name of the volumes
      emptyDir: {} # define type is emptyDir
```

2. Experiment with different types of Volumes - EmptyDir

- **Khi apply minifest và dùng port-forward pod qua cổng 8080, truy cập trên trình duyệt web localhost:8080, ta được kết quả**



2. Experiment with different types of Volumes - HostpathEmptyDir

- **Ta sẽ tạo một volume để storage dữ liệu từ container**

```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: DirectoryOrCreate
```