K8s Volumes09 – Nguyễn Thái Dương

Tất cả file yaml được viết tương ứng với mỗi bài tập lap

```
PS D:\Devops_FPT_Foudations\K8s\final> ls


    Directory: D:\Devops_FPT_Foudations\K8s\final


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         9/13/2023   6:18 PM            804 exerise1-emtydir.yaml
-a----         9/13/2023   6:27 PM           1326 exerise2-configmap.yaml
-a----         9/13/2023   6:51 PM            875 exerise3-gitrepo.yaml
-a----         9/13/2023   6:57 PM           1312 exerise4-nginx-statefullset.yaml
-a----         9/13/2023   6:32 PM           2011 exerise5-pod-pvc-pv.yaml
-a----         9/13/2023   6:59 PM         556613 NguyenThaiDuong_K8S_ Topic 3_Volumes.docx
```

1 Create a nginx app with emptyDir volume (1.5)

Chúng ta tạo một file yaml trong đó có khởi tạo config 1 pod, 2 container được gắn với volume my-volume-emty, volume này sẽ được định nghĩa là emtyDir (volume này chia sẻ dữ liệu giữa các container cùng pod) sau đó chúng ta viết một file trong đường dẫn volume dùng chung để check

```yaml
kind: Pod
metadata:
  name: emptydir-pod # Name of the pod
spec:
  containers:
    - name: container-1 # Name of the container 1
      image: nginx:alpine # The image we will use for container 1
      resources: # Resource limits for the container 1
        limits:
          memory: "128Mi"
          cpu: "500m"
      volumeMounts:
        - name: my-volume-emty
          mountPath: /data/shared # Mount path within the container 1
    - name: container-2 # Name of the container 2
      image: busybox # The image we will use for the container 2
      resources: # Resource limits for the container 2
        limits:
          memory: "128Mi"
          cpu: "500m"
      volumeMounts:
        - name: my-volume-emty
          mountPath: /data/shared # Mount path within the container 2
      command: ["/bin/sh"]
      args: ["-c", "echo Hello from Container 2 > /data/shared/hello.txt"]
      # The command will wirte file hello.txt to emtyvolume that
      # 2 container will get shared data
  volumes:
    - name: my-volume-emty
      emptyDir: {} # Defines an emptyDir volume
```

Chúng ta apply file này và check xem đã có pod chạy chưa

```
PS D:\Devops_FPT_Foudations\K8s\final> kubectl apply -f exerise1-emtydir.yaml
pod/emptydir-pod created
NAME          READY   STATUS              RESTARTS   AGE   IP       NODE                  NOMINATED NODE   READINESS GATES
emptydir-pod  0/2     ContainerCreating   0          4s    <none>   duong1-control-plane  <none>           <none>
NAME          READY   STATUS    RESTARTS     AGE   IP           NODE                  NOMINATED NODE   READINESS GATES
emptydir-pod  1/2     NotReady  1 (4s ago)   12s   10.244.0.17  duong1-control-plane  <none>           <none>
NAME          READY   STATUS             RESTARTS     AGE   IP           NODE                  NOMINATED NODE   READINESS GATES
emptydir-pod  1/2     CrashLoopBackOff   1 (4s ago)   15s   10.244.0.17  duong1-control-plane  <none>           <none>
```

Pod đã chạy, vì config như trên thì ta đoán được conttaner 2 busy box đã chết nhưng nó có ghi được file cùng shared cho container 1 nginx không ta sẽ vào check

```
PS D:\Devops_FPT_Foudations\K8s\final> kubectl exec -it emptydir-pod -- /bin/sh
Defaulted container "container-1" out of: container-1, container-2
/ # ls
bin                  docker-entrypoint.sh  media                product_uuid         srv
data                 etc                   mnt                  root                 sys
dev                  home                  opt                  run                  tmp
docker-entrypoint.d  lib                   proc                 sbin                 usr
/ # cd data/shared
/data/shared # ls
hello.txt
/data/shared # cat hello.txt
Hello from Container 2
/data/shared #
```

Theo như hình ảnh thì ta đã vào được pod và được mặc định vào container 1 nginx, chúng ta check file ở đường dẫn cùng shared volume thì thấy thằng busybox lúc khởi tạo đã ghi được file vào đường dẫn cùng shared sử dụng emtydir volume mount

Nếu pod bị mất thì dữ liệu trong emty volume mà các container cùng được shared cũng mất

```
PS D:\Devops_FPT_Foudations\K8s\final> kubectl delete -f exerise1-emtydir.yaml
pod "emptydir-pod" deleted
PS D:\Devops_FPT_Foudations\K8s\final>
```

2 Create a nginx app with Config map volume (1.5)

Chúng ta viết file khởi tạo file yaml cho configmap có 2 biến learn và session với giá trị tương ứng của nó

```yaml
---
# Config map
apiVersion: v1
kind: ConfigMap
metadata:
  name: configmap-final # Name of the ConfigMap
data:
  # Define a data key "learn" with a value.
  learn: K8s-testing-is-some-thing-trying-too-code
  # Define a data key "session" with a value has "" beacuse it not a strings.
  session: "123456789"

```

Chúng ta sau đó viết file yaml config 1 pod tên “pod -configmap” với một container dùng nginx tên “container-1” ta có truyền vào container đó các biến đã tạo ở file configmap với biến learn trong container sẽ được đặt là biến môi tường COURSE và session được truyền vào container là SESSION

```yaml
# kubectl apply -f exerise2-configmap.yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod-configmap # Name of the pod
spec:
  containers:
    - name: container-1 # Name of the container
      image: nginx:alpine # Use the Nginx Alpine image
      resources:
        limits:
          memory: "128Mi"
          cpu: "500m"
      env:
        # Set an environment variable "COURSE" from the "learn" data in the "configmap-final ConfigMap.
        - name: COURSE
          valueFrom:
            configMapKeyRef:
              name: configmap-final
              key: learn
        # Set an environment variable "SESSION" from the "session" data in the "configmap-final" ConfigMap.
        - name: SESSION
          valueFrom:
            configMapKeyRef:
              name: configmap-final
              key: session
```

Chúng ta apply và check pod cùng với configmap

```
PS D:\Devops_FPT_Foudations\K8s\final> kubectl get pod -o wide
NAME            READY   STATUS    RESTARTS   AGE   IP            NODE                  NOMINATED NODE   READINESS GATES
pod-configmap   1/1     Running   0          49s   10.244.0.18   duong1-control-plane   <none>           <none>
PS D:\Devops_FPT_Foudations\K8s\final> kubectl get configmaps
NAME               DATA   AGE
configmap-final    2      62s
kube-root-ca.crt   1      81m
PS D:\Devops_FPT_Foudations\K8s\final>
```

Ta thấy configmap-final đã được tạo

Ta mô tả nó

```
PS D:\Devops_FPT_Foudations\K8s\final> kubectl describe configmap configmap-final
Name:         configmap-final
Namespace:    default
Labels:       <none>
Annotations:  <none>

Data
====
learn:
----
K8s-testing-is-some-thing-trying-too-code
session:
----
123456789

BinaryData
====

Events:  <none>
PS D:\Devops_FPT_Foudations\K8s\final>
```

xác nhận các biến trong yaml đã có

Ta vào container 1 nginx trong pod ta mới tạo

```
PS D:\Devops_FPT_Foudations\K8s\final> kubectl exec -it pod-configmap -- /bin/sh
/ # ls
bin                  etc                  mnt                  root                 sys
dev                  home                 opt                  run                  tmp
docker-entrypoint.d  lib                  proc                 sbin                 usr
```

Ta check các biến đã vào container chưa

```
/ # echo $COURSE
K8s-testing-is-some-thing-trying-too-code
/ # echo $SESSION
123456789
/ # echo Xac nhận hoàn thành
Xac nhận hoàn thành
/ #
```

Vậy các biến COURSE và SESSION đã nhận giá trị từ volume configmap-final

3 Create a nginx app with gitRepo volume (1)

Chúng ta viết file yaml để tạo gitrepo volume với nginx pod, gitRepo volume nó sẽ clone git repo của chúng ta vào đường dẫn volume trên container với mountPath: /usr/share/nginx/html

K8s > final > ! exerise3-gitrepo.yaml > ...

io.k8s.api.core.v1.Pod (v1@pod.json)

```yaml
# kubectl apply -f exerise3-gitrepo.yaml
# Define a Kubernetes Pod named "nginx-gitrepo-pod" that runs an Nginx container.
apiVersion: v1
kind: Pod
metadata:
  name: pod-gitrepo # Name of the Pod
spec:
  containers:
    - name: container-1 # Name of the container
      image: nginx:alpine # The Nginx container image to use
      resources: # Define resource limits for the container
        limits:
          memory: "128Mi" # Maximum memory limit for the container
          cpu: "500m" # Maximum CPU limit for the container
      volumeMounts:
        - name: gitrepo-final
          mountPath: /usr/share/nginx/html
          readOnly: true
      ports:
        - containerPort: 80 # Expose port 80 on the container
  volumes:
    - name: gitrepo-final # Name of the volume
      gitRepo:
        repository: "https://github.com/NTD151298/NTD1512.github.io.git" # Git repository URL to mount
        revision: "main" # Specify the branch or commit you want to mount
        directory: .
# kubectl port-forward pod-gitrepo 9999:80

```

Ta apply và đợi container khởi tạo, theo việc gán vào gitRepo nó sẽ clone đường dẫn git URL in yaml

```
PS D:\Devops_FPT_Foudations\K8s\final> kubectl apply -f exerise3-gitrepo.yaml
pod/pod-gitrepo configured
PS D:\Devops_FPT_Foudations\K8s\final> kubectl get pod -o wide
NAME          READY   STATUS              RESTARTS   AGE     IP       NODE                 NOMINATED NODE   READINESS GATES
pod-gitrepo   0/1     ContainerCreating   0          2m16s   <none>   duong1-control-plane   <none>           <none>
PS D:\Devops_FPT_Foudations\K8s\final>
```

4 Create a nginx app with statefullset (Example)(1.5)

Chúng ta tạo pod nginx với statefullset với replica là 3 có service, volume với tên là nginx-data

```yaml
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: nginx-statefulset  # Name of the StatefulSet
spec:
  replicas: 3  # Desired number of replicas for the StatefulSet
  selector:
    matchLabels:
      app: nginx  # Selector for matching pods with the "app: nginx" label
  serviceName: "nginx"  # Name of the Kubernetes service associated with this StatefulSet
  template:
    metadata:
      labels:
        app: nginx  # Label applied to pods created from this template
    spec:
      containers:
        - name: nginx
          image: nginx:latest  # Docker image to use for the Nginx container
          ports:
            - containerPort: 80  # Port to expose in the container
          volumeMounts:
            - name: nginx-data  # Name of the volume to mount
              mountPath: /usr/share/nginx/html  # Mount path in the container
  volumeClaimTemplates:
    - metadata:
        name: nginx-data  # Name of the volume claim template
      spec:
        accessModes: ["ReadWriteOnce"]  # Access mode for the volume claim
        storageClassName: "standard"  # Storage class name for dynamic provisioning
        resources:
          requests:
            storage: 1Gi  # Requested storage size for each replica's volume
```

Ta apply và đợi xem kết quả

```
PS D:\Devops_FPT_Foudations\K8s\final> kubectl apply -f exerise4-nginx-statefullset.yaml
statefulset.apps/nginx-statefulset created
PS D:\Devops_FPT_Foudations\K8s\final> kubectl get pod -o wide
NAME                READY   STATUS              RESTARTS   AGE   IP           NODE                 NOMINATED NODE   READINESS GATES
nginx-statefulset-0   1/1     Running             0          4s    10.244.0.20   duong1-control-plane   <none>           <none>
nginx-statefulset-1   0/1     ContainerCreating   0          1s    <none>       duong1-control-plane   <none>           <none>
PS D:\Devops_FPT_Foudations\K8s\final> kubectl get pod -o wide
NAME                READY   STATUS    RESTARTS   AGE   IP           NODE                 NOMINATED NODE   READINESS GATES
nginx-statefulset-0   1/1     Running   0          17s   10.244.0.20   duong1-control-plane   <none>           <none>
nginx-statefulset-1   1/1     Running   0          14s   10.244.0.21   duong1-control-plane   <none>           <none>
nginx-statefulset-2   1/1     Running   0          10s   10.244.0.22   duong1-control-plane   <none>           <none>
PS D:\Devops_FPT_Foudations\K8s\final>
```

Ta đã tạo được 3 replica container nginx đều có port là 80 và được gán volume tên là nginx-data, volume này yêu cầu sức chứa là 1Gb

Ta cũng tạo được 3 pv và 3 pvc theo yêu cầu của yaml statefullset

```
PS D:\Devops_FPT_Foudations\K8s\final> kubectl get pv
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM                                      STORAGECLASS   REASON   AGE
pvc-0243b602-c3bb-4d12-bce5-7610affaa5d3   1Gi        RWO            Delete           Bound    default/nginx-data-nginx-statefulset-2     standard                113m
pvc-03ff5c79-0919-42ec-b313-5898a640bfd2   1Gi        RWO            Delete           Bound    default/nginx-data-nginx-statefulset-0     standard                113m
pvc-af9c725b-4ccf-4761-a9ac-4a4c957f2f34   1Gi        RWO            Delete           Bound    default/nginx-data-nginx-statefulset-1     standard                113m
PS D:\Devops_FPT_Foudations\K8s\final> kubectl get pvc
NAME                           STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
nginx-data-nginx-statefulset-0   Bound    pvc-03ff5c79-0919-42ec-b313-5898a640bfd2   1Gi        RWO            standard       114m
nginx-data-nginx-statefulset-1   Bound    pvc-af9c725b-4ccf-4761-a9ac-4a4c957f2f34   1Gi        RWO            standard       113m
nginx-data-nginx-statefulset-2   Bound    pvc-0243b602-c3bb-4d12-bce5-7610affaa5d3   1Gi        RWO            standard       113m
PS D:\Devops_FPT_Foudations\K8s\final>
```

5 Create a nginx app using Persisten Volumes Claim to claim Persisten Volume resource (1.5)

Ta viết file yaml config pod se được gán vào một Persisten Volumes Claim tên my-volume-5

K8s > final > ! exerise5-pod-pvc-pv.yaml > {} metadata > name

```yaml
# kubectl apply -f exerise5-pod-pvc-pv.yaml

# Pod vp
# Specifying the Kubernetes resource as a Pod, and giving it the name "vp-pod".
apiVersion: v1
kind: Pod
metadata:
  name: vp-pod
spec:
  # Defining the containers that will run inside the Pod. In this case, there's one container with the name "nginx-vp-con".
  containers:
    - name: nginx-vp-con
      # Specifying the Docker image to be used for this container, which is the lightweight "nginx:alpine" image.
      image: nginx:alpine
      # Setting resource limits for the container, including memory and CPU limits.
      resources:
        limits:
          memory: "128Mi"
          cpu: "500m"
      # Specifying volume mounts for this container, in this case, a volume with the name "my-volume-1" will be mounted.
      volumeMounts:
        - name: my-volume-5
          # Defining the path where the volume will be mounted inside the container.
          mountPath: /var/www/html
  # Defining a volume with the name "my-volume-1" and associating it with a PersistentVolumeClaim (PVC) named "vpcday5".
  volumes:
    - name: my-volume-5
      persistentVolumeClaim:
        claimName: vpcday-final
```

File config yaml của my-volume-5 với yêu cầu 1Gb và sau đó nó sẽ tìm 1 Persisten Volumes đáp ứng

```yaml
# VPC
# This section defines a PersistentVolumeClaim (PVC) named "vpcday5" with resource requests for 1 gigabyte of
# storage. It specifies "ReadWriteOnce" access mode.
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: vpcday-final
spec:
  resources:
    requests:
      storage: 1Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
---
```

Ta tạo file Persisten Volumes được mount trên node host file có sức chứa là 5gb

```yaml
# VP
# This section defines a PersistentVolume (PV) named "pvday5" with a capacity of 5 gigabytes. It also specifies
#"ReadWriteOnce" access mode and uses a hostPath at "/tmp/pv-data" for storage.
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-final
spec:
  capacity:
    storage: 5Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
  storageClassName: standard
  hostPath:
    path: "/tmp/pv-data"

```

Ta apply

```
PS D:\Devops_FPT_Foudations\K8s\final> kubectl apply -f exerise5-pod-pvc-pv.yaml
pod/vp-pod created
persistentvolumeclaim/vpcday-final created
persistentvolume/pv-final created
PS D:\Devops_FPT_Foudations\K8s\final> kubectl get pod -o wide
NAME     READY   STATUS    RESTARTS   AGE   IP           NODE                NOMINATED NODE   READINESS GATES
vp-pod   1/1     Running   0          6s    10.244.0.7   duong1-control-plane   <none>           <none>
PS D:\Devops_FPT_Foudations\K8s\final>
```

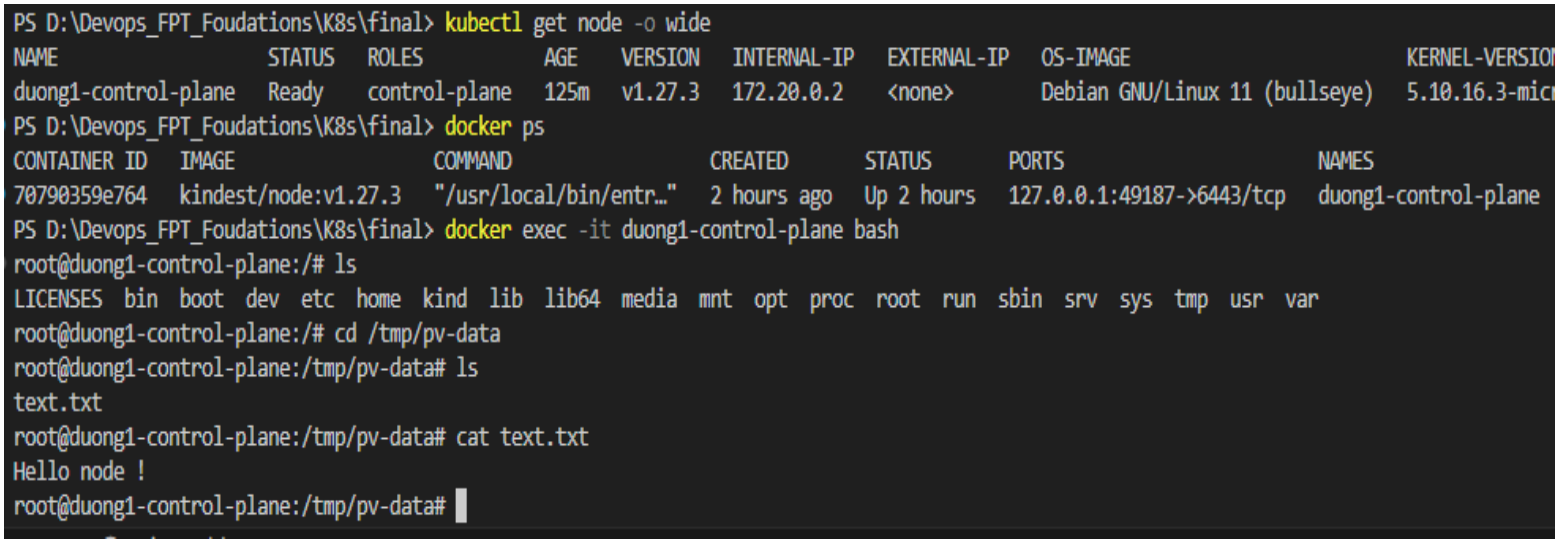
Ta sau đó check pv và pvc

```
PS D:\Devops_FPT_Foudations\K8s\final> kubectl get pvc
NAME           STATUS   VOLUME     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
vpcday-final   Bound    pv-final   5Gi        RWO            standard       3m47s
PS D:\Devops_FPT_Foudations\K8s\final> kubectl get pv
NAME       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM                  STORAGECLASS   REASON   AGE
pv-final   5Gi        RWO            Retain           Bound    default/vpcday-final   standard                3m50s
PS D:\Devops_FPT_Foudations\K8s\final>
```

Ta vào container trong pod và viết vào đường dẫn pvc volume (/var/www/html) file text.txt

```
PS D:\Devops_FPT_Foudations\K8s\final> kubectl exec -it vp-pod -- /bin/sh
/ # ls
bin                  etc                  mnt                  root                 sys
dev                  home                 opt                  run                  tmp
docker-entrypoint.d  lib                  proc                 sbin                 usr
docker-entrypoint.sh media                product_uuid         srv                  var
/ # cd /var/www/html
/var/www/html # ls
/var/www/html # touch text.txt
/var/www/html # echo "Hello node !" > text.txt
```

File này theo đường dẫn sẽ được đưa vào Persisten Volumes Claim và truyền thẳng vào node theo đường dẫn “/tmp/pv-data” nhờ vào thằng Persisten Volumes đã được mount vào node file systems



Theo như kết quả trên ta vào liệt kê node và sau đó vào thẳng node chứa pod chúng ta mới tạo đang chạy ở đây là duong1-control-plane sau đó chúng ta vào thẳng đường dẫn tmp/pv-data để check file

Text.txt chúng ta đã tạo khi ở trong container pod. File truyền qua PVC PV rồi vào file của Node