Kubernetes

1. **Kubernetes là gì?**
    - Hệ thống điều phối (orchestration) cho các cụm container.
    - Được thiết kế để triển khai, quản lý, mở rộng và cập nhật ứng dụng container hóa.
2. **Các chức năng chính của Kubernetes**
    - Triển khai ứng dụng container trên cụm máy chủ.
    - Tự động mở rộng (scale) ứng dụng.
    - Cập nhật phiên bản phần mềm không gây downtime.
    - Hỗ trợ debug ứng dụng container.
3. **Lợi ích khi dùng Kubernetes**
    - Đảm bảo ứng dụng luôn sẵn sàng 24/7.
    - Hỗ trợ triển khai và cập nhật liên tục.
    - Quản lý tài nguyên hiệu quả, tự động phân phối workload.
    - Nền tảng mã nguồn mở, sẵn sàng cho môi trường sản xuất.
4. **Kiến trúc Kubernetes Cluster**
    - Gồm 2 thành phần: Control Plane (quản lý cụm) và Node (chạy ứng dụng).
    - Control Plane điều phối, lên lịch, mở rộng, cập nhật ứng dụng.
    - Node là máy chủ (VM hoặc vật lý), mỗi node có Kubelet để giao tiếp với Control Plane.
    - Cluster nên có tối thiểu 3 node để đảm bảo tính sẵn sàng và dự phòng.
5. **Triển khai ứng dụng trên Cluster**
    - Control Plane nhận lệnh triển khai, lên lịch container chạy trên các node.
    - Giao tiếp qua Kubernetes API.
6. **Minikube là gì?**

- Minikube là công cụ tạo nhanh một cluster Kubernetes trên máy cá nhân (Mac, Linux, Windows) để phát triển và thử nghiệm.
- Hỗ trợ dashboard trực quan để quản lý tài nguyên.

7. **Quy trình triển khai ứng dụng mẫu với Minikube**
   - Khởi tạo cluster: `minikube start`
   - Mở dashboard: `minikube dashboard`
   - Tạo Deployment: `kubectl create deployment <tên> --image=<image>`
   - Kiểm tra trạng thái: `kubectl get deployments`, `kubectl get pods`
   - Xem log ứng dụng: `kubectl logs <tên-pod>`
   - Tạo Service để expose ứng dụng: `kubectl expose deployment <tên> --type=LoadBalancer --port=<port>`
   - Truy cập ứng dụng: `minikube service <tên-service>`
   - Có thể dùng `kubectl proxy` để truy cập API nội bộ qua localhost.
8. **Addons trong Minikube**
   - Có thể bật/tắt các addon như dashboard, metrics-server để mở rộng chức năng cluster.
   - Quản lý addon: `minikube addons list`, `minikube addons enable <addon>`, `minikube addons disable <addon>`
9. **File Deployment:**

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hms-api
spec:
  replicas: 1
  selector:
    matchLabels:
      app: hms-api
  template:
    metadata:
      labels:
        app: hms-api
```

```
 spec:
   containers:
   - name: hms-api
     image: hms-api:latest
     imagePullPolicy: IfNotPresent
     ports:
     - containerPort: 80
     envFrom:
     - configMapRef:
         name: hms-api-config
     - secretRef:
         name: hms-api-secret
     volumeMounts:
     - name: epic-privatekey
       mountPath: /src/privatekey.pem
       subPath: privatekey.pem
   volumes:
   - name: epic-privatekey
     secret:
       secretName: epic-privatekey
```

- `apiVersion: apps/v1`

  → Khai báo version API cho loại tài nguyên Deployment.

- `kind: Deployment`

  → Định nghĩa đây là một Deployment (quản lý việc tạo, cập nhật, scale Pod).

- `metadata:`

  → Thông tin định danh cho Deployment (tên, nhãn...).

- `name: hms-api`

  → Tên của Deployment.

- `spec:`

  → Định nghĩa cấu hình cho Deployment.

- `replicas: 1`

  → Số lượng Pod sẽ được chạy (ở đây là 1).

- `selector:`

  → Chọn Pod nào thuộc về Deployment này dựa trên nhãn.
- `matchLabels: app: hms-api`

  → Pod có label `app: hms-api` sẽ thuộc về Deployment này.
- `template:`

  → Mẫu cấu hình cho Pod được tạo bởi Deployment.
- `metadata: labels: app: hms-api`

  → Gán label cho Pod để Service có thể tìm và kết nối.
- `spec: containers:`

  → Định nghĩa danh sách container trong Pod.
- `name: hms-api`

  → Tên container.
- `image: hms-api:latest`

  → Docker image sử dụng cho container.
- `imagePullPolicy: IfNotPresent`

  → Chỉ pull image nếu chưa có sẵn trên node.
- `ports: containerPort: 80`

  → Mở cổng 80 trong container để nhận request.
- `envFrom:`

  → Lấy biến môi trường từ ConfigMap và Secret.
- `configMapRef: name: hms-api-config`

  → Lấy biến môi trường từ ConfigMap `hms-api-config`.
- `secretRef: name: hms-api-secret`

  → Lấy biến môi trường từ Secret `hms-api-secret`.
- `volumeMounts:`

  → Mount file hoặc thư mục vào container.
- `name: epic-privatekey`

  → Tên volume mount.
- `mountPath: /src/privatekey.pem`

→ Đường dẫn file trong container.

- `subPath: privatekey.pem`

  → Chỉ mount file `privatekey.pem`.

- `volumes:`

  → Định nghĩa volume cho Pod.

- `secret: secretName: epic-privatekey`

  → Volume lấy từ Secret `epic-privatekey`.

10. **File Service:**

```
apiVersion: v1
kind: Service
metadata:
  name: hms-api-service
spec:
  type: LoadBalancer
  selector:
    app: hms-api
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

- `apiVersion: v1`

  → Khai báo version API cho loại tài nguyên Service.

- `kind: Service`

  → Định nghĩa đây là một Service (cổng truy cập vào Pod).

- `metadata: name: hms-api-service`

  → Tên của Service.

- `spec:`

  → Định nghĩa cấu hình cho Service.

- `type: LoadBalancer`

  → Service sẽ expose ra ngoài cluster (Minikube sẽ tạo tunnel).

- `selector: app: hms-api`

  → Service sẽ forward request tới Pod có label `app: hms-api`.

- `ports:`

  → Định nghĩa các cổng của Service.

- `protocol: TCP`

  → Sử dụng giao thức TCP.

- `port: 80`

  → Cổng bên ngoài Service.

- `targetPort: 80`

  → Cổng bên trong container (Pod).

11. **LoadBalancer và ClusterIP**
    - **ClusterIP (mặc định):**
      - Service chỉ có IP nội bộ trong cluster.
      - Chỉ các Pod/service khác trong cluster mới truy cập được.
      - Dùng cho giao tiếp nội bộ giữa các microservice.
    - **LoadBalancer:**
      - Service sẽ được expose ra ngoài cluster.
      - Kubernetes sẽ yêu cầu cloud provider tạo một IP public (Minikube sẽ tạo tunnel).
      - Dùng để truy cập từ bên ngoài (ví dụ: từ trình duyệt, Postman).

12. **Lệnh docker + minikube hay dùng**
    - Set docker vào minikube: `eval $(minikube -p minikube docker-env)`
    - Unset docker khỏi minikube: `eval $(minikube docker-env -u)`
    - Build Docker image trong Minikube: `docker build -t hms-api:latest -f HospitalManagementSystem.API/Dockerfile .`
    - Check Images của docker: `docker images`
    - Tạo ConfigMap và Secret từ file .env và privatekey:

      ```
      kubectl create configmap hms-api-config --from-env-file=.env
      kubectl create secret generic hms-api-secret \
      ```

`--from-literal=REDIS-PASSWORD=redis123 \`
`--from-literal=JWT-SECRET=HMS_SuperSecretKey_ForDevelopment_2024_MustBe32CharsOrMore!`
`kubectl create secret generic epic-privatekey --from-file=privatekey.pem=src/privatekey.pem`

- Delete config map: `kubectl delete configmap hms-api-config`
- Apply services lên minikube: `kubectl apply -f redis-deployment.yaml`
- Lấy list pods: `kubectl get pods`
- Lấy list services: `kubectl get services`
- Lấy logs của 1 pod: `kubectl logs hms-api-bc5c9fb5d-plkwr`
- Delete 1 pod: `kubectl delete pod hms-api-bc5c9fb5d-8f9j5`
- Expose service type LoadBalancer: `minikube service hms-api-service`
- Get ReplicaSets: `kubectl get rs`
- Kiểm tra trạng thái ReplicaSets: `kubectl describe rs/hms-api-bc5c9fb5d`
- Ví dụ set deployment sang phiên bản mới hơn: `kubectl set image deployment/hms-api hms-api=hms-api:v2`
- Liệt kê image trong minikube: `minikube ssh -- docker images | grep hms-api`
- Xoá image: `minikube ssh -- docker rmi <IMAGE_ID>`
- Scale số lượng pod để đảm bảo 24/7: `kubectl scale deployment hms-api --replicas=3`
- Rollback phiên bản cũ: `kubectl rollout undo deployment/hms-api`
- Kiểm tra lịch sử phiên bản: `kubectl rollout history deployment/hms-api`

Hướng dẫn từng bước tạo một Deployment và Service trên Minikube

Bước 1: Cài đặt kubectl, Minikube và Docker

- Cài kubectl: https://kubernetes.io/docs/tasks/tools/install-kubectl-macos/
- Cài Minikube: https://minikube.sigs.k8s.io/docs/start/?arch=%2Fmacos%2Fx86-64%2Fstable%2Fbinary+download
- Cài Docker Desktop: https://www.docker.com/products/docker-desktop/

Bước 2: Khởi động Minikube

`minikube start`

> Lệnh này sẽ tạo một cluster Kubernetes trên máy bạn.

Bước 3: Chuyển Docker context sang Minikube

`eval $(minikube -p minikube docker-env)`

> Để build image Docker nằm trong Minikube, giúp Kubernetes sử dụng được image này.

Bước 4: Build Docker image cho ứng dụng

`docker build -t hms-api:latest -f HospitalManagementSystem.API/Dockerfile .`

> Tạo image tên `hms-api:latest` từ Dockerfile.

Bước 5: Tạo ConfigMap và Secret từ file cấu hình

```
kubectl create configmap hms-api-config --from-env-file=.env
kubectl create secret generic hms-api-secret \
--from-literal=REDIS-PASSWORD=redis123 \
--from-literal=JWT-SECRET=HMS_SuperSecretKey_ForDevelopment_2024_MustBe32CharsOrMore!
kubectl create secret generic epic-privatekey --from-file=privatekey.pem=src/privatekey.pem
```

> Lưu các biến môi trường và file nhạy cảm vào Kubernetes.

Bước 6: Tạo file Deployment (hms-api-deployment.yaml)

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hms-api
spec:
  replicas: 1
  selector:
    matchLabels:
```

```
      app: hms-api
  template:
    metadata:
      labels:
        app: hms-api
    spec:
      containers:
      - name: hms-api
        image: hms-api:latest
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 80
        envFrom:
        - configMapRef:
            name: hms-api-config
        - secretRef:
            name: hms-api-secret
        volumeMounts:
        - name: epic-privatekey
          mountPath: /src/privatekey.pem
          subPath: privatekey.pem
      volumes:
      - name: epic-privatekey
        secret:
          secretName: epic-privatekey
```

> File này định nghĩa cách Kubernetes chạy ứng dụng của bạn. (Đi tới giải thích)

Bước 7: Tạo file Service (hms-api-service.yaml)

```
apiVersion: v1
kind: Service
metadata:
  name: hms-api-service
spec:
```

```
  type: LoadBalancer
  selector:
    app: hms-api
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

> File này giúp ứng dụng của bạn có thể truy cập từ bên ngoài/nội bộ cluster.
> (Đi tới giải thích)

Bước 8: Deploy ứng dụng lên Minikube

```
kubectl apply -f hms-api-deployment.yaml
kubectl apply -f hms-api-service.yaml
```

> Tạo Pod và Service trên Kubernetes.

Bước 9: Kiểm tra trạng thái Pod và Service

```
kubectl get pods
kubectl get services
```

> Đảm bảo Pod đã chạy và Service đã được tạo.

Bước 10: Truy cập ứng dụng từ bên ngoài

```
minikube service hms-api-service
```

> Lệnh này sẽ mở trình duyệt tới địa chỉ ứng dụng của bạn.

Bước 11: Kiểm tra log ứng dụng nếu gặp lỗi

```
kubectl logs <tên-pod-hms-api>
```

> Xem log để biết ứng dụng có lỗi không.

Bước 12: Scale số lượng Pod để đảm bảo ứng dụng luôn chạy

`kubectl scale deployment hms-api --replicas=3`

> Đảm bảo ứng dụng luôn sẵn sàng 24/7.

Tóm tắt quy trình

1. Cài Minikube và Docker.
2. Khởi động Minikube.
3. Chuyển Docker context sang Minikube.
4. Build image Docker cho ứng dụng.
5. Tạo ConfigMap và Secret.
6. Tạo file Deployment và Service.
7. Apply lên Minikube.
8. Kiểm tra trạng thái.
9. Truy cập ứng dụng qua Minikube service.
10. Kiểm tra log nếu cần.
11. Scale Pod để tăng tính sẵn sàng.

Các lỗi hay gặp

- Build image sai context (không phải Minikube).
- Sai tên image hoặc tag trong deployment.
- Thiếu hoặc sai ConfigMap/Secret.
- Sai selector trong Service.
- Pod không chạy do thiếu tài nguyên hoặc cấu hình sai.
- Quên apply lại sau khi sửa file YAML.
- Quên mở terminal khi dùng `minikube service` với Docker driver.

Hướng dẫn chi tiết cập nhật phiên bản mới và rollback trên Minikube

A. Cập nhật phiên bản mới cho ứng dụng (Deployment)

Bước 1: Build image mới với tag mới (ví dụ v2)

```
eval $(minikube -p minikube docker-env)  # Đảm bảo build trong Minikube
docker build -t hms-api:v2 -f HospitalManagementSystem.API/Dockerfile .
```

> Tạo image mới tên `hms-api:v2`.

Bước 2: Cập nhật Deployment để dùng image mới

- Mở file: hms-api-deployment.yaml
- Sửa image trong container

```
image: hms-api:latest
```

→

```
image: hms-api:v2
```

- Chạy lệnh apply

  ```
  kubectl apply -f hms-api-deployment.yaml
  ```

  > Lệnh này sẽ cập nhật Pod của Deployment sang image mới.

Bước 3: Kiểm tra trạng thái Pod

```
kubectl get pods
kubectl describe pod <tên-pod-hms-api>
```

> Đảm bảo Pod mới đã chạy với image mới.

Bước 4: Kiểm tra log ứng dụng

```
kubectl logs <tên-pod-hms-api>
```

> Xem log để kiểm tra ứng dụng có lỗi không.

Bước 5: Truy cập lại ứng dụng

```
minikube service hms-api-service
```

> Kiểm tra ứng dụng đã cập nhật phiên bản mới.

B. Rollback về phiên bản cũ nếu phiên bản mới bị lỗi

Bước 1: Kiểm tra lịch sử các phiên bản (revision) của Deployment

```
kubectl rollout history deployment/hms-api
```

> Xem các phiên bản đã từng deploy.

Bước 2: Rollback về phiên bản trước

```
kubectl rollout undo deployment/hms-api
```

> Kubernetes sẽ tự động chuyển về image của phiên bản trước

Bước 3: Kiểm tra lại trạng thái Pod và log

```
kubectl get pods
```

```
kubectl logs <tên-pod-hms-api>
```

> Đảm bảo pod đã chạy lại với phiên bản cũ và không còn lỗi

Bước 4: Truy cập lại ứng dụng để xác nhận rollback thành công

```
minikube service hms-api-service
```

Lưu ý quan trọng khi update/rollback

- Luôn build image mới trong docker context của minikube.
- Kiểm tra log sau khi cập nhật để phát hiện lỗi sớm.
- Rollback chỉ hoạt động khi đã có ít nhất 2 phiên bản (2 lần cập nhật image).
- Sau rollback, pod sẽ tự động chạy lại với image cũ.

Cách lấy file từ PVC bằng lệnh

Bước 1: Lấy tất cả các pods

`kubectl get pods`

Bước 2: Copy tên pod pvc cần lấy file và exec bash

`kubectl exec -it central-postgres-559c566f66-xc7pd -- bash`

Bước 3: List file trong đường dẫn volume được mount trong pod

`ls /var/lib/postgresql/data`

> Lệnh liệt kê tên tất cả file trong thư mục

Bước 4: Copy file ra ngoài

`kubectl cp central-postgres-559c566f66-xc7pd:/var/lib/postgresql/data/<tên-file> ./<tên-file>`

> Lệnh để copy file ra thư mục gốc

Hướng dẫn chi tiết để lấy file từ PVC bằng cách tạo một pod và service pvc-reader

Bước 1: tạo Pod pvc-reader để mount PVC

Mục đích:

Tạo một pod mới, mount PVC chứa dữ liệu cần lấy (ví dụ của PostgreSQL), và chạy một HTTP server để truy cập file qua HTTP.

File cấu hình:

Tạo file pvc-reader.yaml với nội dung sau:

```
apiVersion: v1
kind: Pod
metadata:
  name: pvc-reader
  labels:
    app: pvc-reader
spec:
  containers:
  - name: pvc-reader
    image: python:3.11-slim
    command: ["python3", "-m", "http.server", "8080"]
    workingDir: /data
    ports:
    - containerPort: 8080
    volumeMounts:
    - name: data
      mountPath: /data
  volumes:
  - name: data
    persistentVolumeClaim:
      claimName: central-postgres-pvc
```

Giải thích:

- Pod này dùng image Python, chạy HTTP server tại data (nơi mount PVC).
- PVC `central-postgres-pvc` là nơi lưu dữ liệu của PostgreSQL.
- Nghĩa là, mọi file trong `/var/lib/postgresql/data` của pod central-postgres **cũng xuất hiện ở** data của pod pvc-reader.

Bước 2: Tạo Service pvc-reader kiểu ClusterIP

Mục đích:

Expose pod pvc-reader ra mạng nội bộ Kubernetes để các pod khác (ví dụ API server) có thể truy cập.

Thêm vào file pvc-reader.yaml:

```
---
apiVersion: v1
kind: Service
metadata:
  name: pvc-reader
spec:
  selector:
    app: pvc-reader
  ports:
  - protocol: TCP
    port: 8080
    targetPort: 8080
  type: ClusterIP
```

Giải thích:

- Service này chỉ cho phép truy cập nội bộ trong cluster (không public ra ngoài).
- Các pod khác có thể gọi `http://pvc-reader:8080/<file-name>` để lấy file.

Bước 3: Deploy Pod và Service pvc-reader

Chạy lệnh sau để tạo pod và service:

`kubectl apply -f pvc-reader.yaml`

Kiểm tra pod và service đã chạy:

`kubectl get pods`
`kubectl get svc`

Bước 4: Truy cập file qua API nội bộ

```
[HttpGet("db-file")]
[Authorize(Roles = "Admin")]
public async Task<IActionResult> GetDbFile([FromQuery] string fileName)
{
    // Địa chỉ pvc-reader (giả sử đã port-forward hoặc có service)
    var pvcReaderUrl = $"http://pvc-reader:8080/{fileName}";
    using var httpClient = new HttpClient();
    var resp = await httpClient.GetAsync(pvcReaderUrl);
    if (!resp.IsSuccessStatusCode)
        return NotFound("File not found");

    var stream = await resp.Content.ReadAsStreamAsync();
    return File(stream, "application/octet-stream", fileName);
}
```