Kubernetes 2

Pods & Containers in Kubernetes

caodangsao@gmail.com

Module Target

Kết thúc bài học, học viên cần đạt được các kĩ năng sau:

- Có khả năng làm việc cụm Kubernetes với kubectl.
- Có hiểu biết về Pods và Containers trong Kubernetes.
- Nắm được cách quản lý cấu hình ứng dụng, quản lý tài nguyên containers.
- Có thể cấu hình giám sát được tình trạng của container trong cụm Kubernetes.
- Biết cách sử dụng Init Containers để tùy biến triển khai ứng dụng.

Nội dung

1 Pods và Containers in K8s

- Sử dụng kubectl để tương tác cụm K8s
- Quản lý cấu hình ứng dụng
- Quản lý tài nguyên containers
- Giám sát tình trạng container
- Self-Healing Pods
- Pods với nhiều container
- Pods với Init Containers

2 Summary

- Tóm tắt nội dung buổi học

Lab 1: Working with kubectl

Lab 2: manage app config

Lab 3: manage resource request

Lab 4: monitor container health

Lab 5: self healing pods

Lab 6: multi container pods

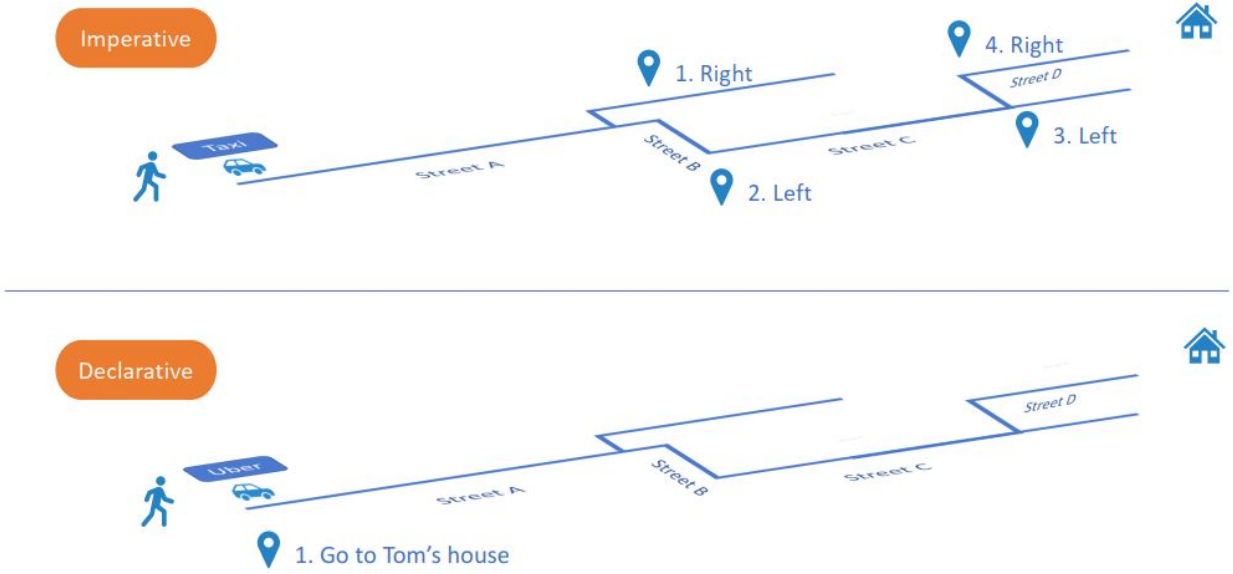
Lab 7: pod with init container

Section 1:

Pods và Containers in K8s

Imperative & Declarative

Ví dụ về việc di chuyển bằng taxi từ Công ty về nhà Tom



Imperative & Declarative

So sánh Imperative và Declarative trong Infrastructure as Code

Imperative

```
1. Provision a VM by the name ‘web-server’
2. Install NGINX Software on it
3. Edit configuration file to use port ‘8080’
4. Edit configuration file to web path ‘/var/www/nginx’
5. Load web pages to ‘/var/www/nginx’ from GIT Repo - X
6. Start NGINX server
```

Declarative

```
VM Name: web-server
Package: nnginx:1.18
Port: 8080
Path: /var/www/nginx
Code: GIT Repo - X
```

Imperative & Declarative

So sánh Imperative và Declarative trong Kubernetes

Imperative

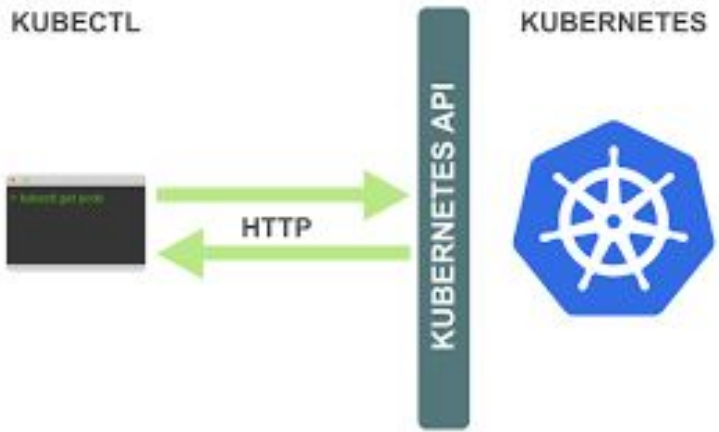
```
> kubectl run --image=nginx nginx
> kubectl create deployment --image=nginx nginx
> kubectl expose deployment nginx --port 80
> kubectl edit deployment nginx
> kubectl scale deployment nginx --replicas=5
> kubectl set image deployment nginx nginx=nginx:1.18
> kubectl create -f nginx.yaml
> kubectl replace -f nginx.yaml
> kubectl delete -f nginx.yaml
```

Declarative

```
> kubectl apply -f nginx.yaml
```

Using Kubectl

- ➔ Kubectl là công cụ dòng lệnh để thao tác với K8s.
- ➔ Kubectl sử dụng các API nội bộ của K8s để thực hiện các lệnh.



Using Kubectl - kubectl get

➔ Kubectl get được sử dụng để lấy các đối tượng trong cụm K8s.

```
kubectl get <object_type> <object_name> -o <output> --sort-by <JSONpath> --selector
<selector>
```

➔ -o — định dạng đầu ra YML/JSON

➔ - -sort-by — Sắp xếp đầu ra sử dụng JSON path Expression

➔ - -Selector — Lọc kết quả bằng label

Using Kubectl - kubectl describe

➔ Kubectl describe được sử dụng để lấy thông tin chi tiết đối tượng trong

kubectl describe <object_type> <object_name> -o <output>

Using Kubectl - kubectl create

- ➔ Kubectl create được sử dụng để tạo đối tượng trong cụm K8s.
- ➔ File mô tả có định dạng YAML

```
kubectl create -f <file_name>
```

Using Kubectl - kubectl apply

- ➔ Kubectl apply được sử dụng để tạo đối tượng trong cụm k8s tương tự như kubectl create.
- ➔ Nếu user sử dụng kubectl apply trên các đối tượng đã tồn tại. Nó sẽ sửa đổi tượng đó nếu có thể.

```
kubectl apply -f <file_name>
```

Using Kubectl - kubectl delete

➔ Kubectl delete được sử dụng để xóa đối tượng trong cụm k8s.

```
kubectl delete -f <file_name>
```

Using Kubectl - kubectl delete

- ➔ Kubectl exec được sử dụng để chạy câu lệnh trong container.
- ➔ K8s tài nguyên phải được chạy trong container, nơi mà lệnh kubectl exec đang thực hiện.

```
kubectl exec <pod_name> -c <container_name>
```

Labs

Lab 1: Working with kubectl

K8s Pods & Containers Overview

- ➔ Quản lý cấu hình ứng dụng
- ➔ Quản lý tài nguyên containers
- ➔ Giảm sát tình trạng container
- ➔ Tạo self-healing Pods

Manage Application Configuration

- ➔ Cấu hình ứng dụng là gì ?
- ➔ ConfigMaps
- ➔ Secrets
- ➔ Environment Variables
- ➔ Configuration Volumes
- ➔ Lab demo

Application Configuration

- ➔ Ta có thể hiểu "cấu hình" là các cài đặt ảnh hưởng đến hoạt động của ứng dụng.
- ➔ Kubernetes cho phép người dùng chuyển các giá trị cấu hình động cho ứng dụng lúc Runtime.
- ➔ Các cấu hình động này giúp người dùng kiểm soát luồng ứng dụng.

ConfigMap

- ➔ Giữ dữ liệu không nhạy cảm trong ConfigMap, dữ liệu này có thể được chuyển đến container của ứng dụng.
- ➔ Config Map lưu trữ data ở định dạng Key-Value.
- ➔ ConfigMaps cho phép ta tách các cấu hình của mình khỏi Pod và các thành phần.
- ➔ ConfigMap giúp làm cho các cấu hình dễ thay đổi và quản lý hơn, đồng thời ngăn mã hóa cứng dữ liệu cấu hình thành các thông số kỹ thuật của Pod.

ConfigMap Commands

➔ Tạo ConfigMaps qua File cấu hình.

```
kubectl create configmap [NAME] --from-file [/PATH/TO/FILE.PROPERTIES] --from-file
[/PATH/TO/FILE2.PROPERTIES]
```

```
kubectl create configmap [NAME] --from-file [/PATH/TO/DIRECTORY]]
```

➔ Lấy ConfigMap qua CLI

```
kubectl get configmap <Config_map_name> -o yaml/json
```

Secrets

- Secrets tương tự như ConfigMap nhưng được thiết kế để giữ các dữ liệu nhạy cảm.
- Tạo Secrets từ file.

```
kubectl create secret generic db-user-pass --from-file =./username.txt --from-file
=./password.txt
```

- Lưu ý: Các ký tự đặc biệt như $,\,* và ! yêu cầu escaping.

Secrets

➔ Lấy Secrets.

```
kubectl get secrets
```

➔ Mô tả Secrets.

```
kubectl describe secrets <Secret_name>
```

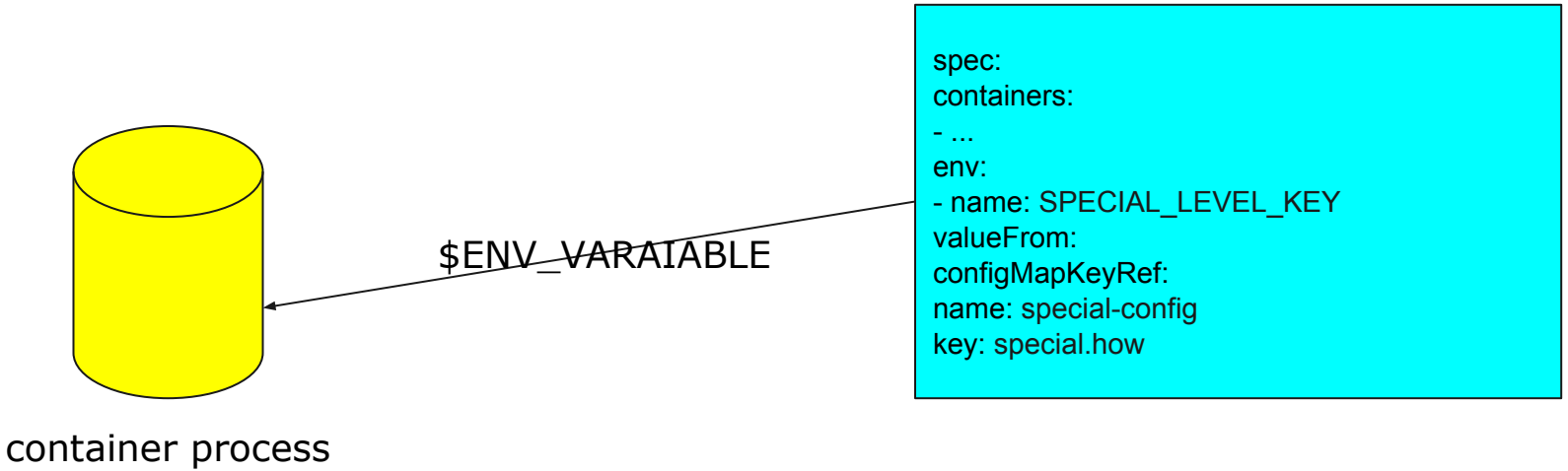
K8s Secrets

➔ Ví dụ mẫu về file YAML Secrets.

```
apiVersion: v1
kind: Secret
metadata:
name: mysecret-manifest
type: Opaque
data:
username: YW5zaHVsY2hhdWhhbg==
password: VGVzdGt1VybmV0ZXMxMjM0NQ==
```

Pass ConfigMap and Secrets to Containers

➔ User có thể chuyển Secrets và Config Map đến Container sử dụng Environment Variables.



Pass ConfigMap and Secrets to Containers

- ➔ Cấu hình Mount Volume là cách khác để chuyển dữ liệu Config và Secrets đến Container.
- ➔ Dữ liệu Config sẽ hiển thị dưới dạng Files trong hệ thống file của Container

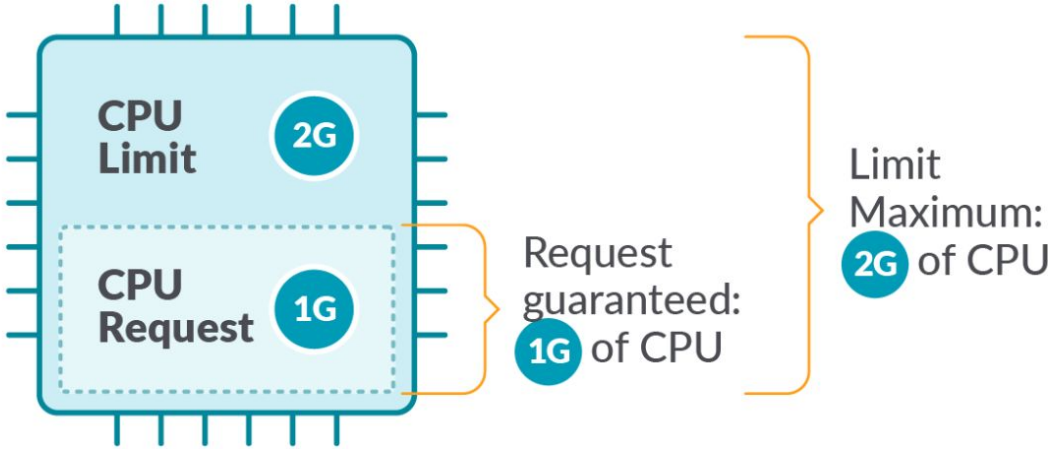
```
volumes:
- name: config-volume
configMap:
name: special-config
```

Labs

Lab 2: Manage Application Configuration

Manager Container Resources

- ➔ Resource Request
- ➔ Resource Limit
- ➔ Lab demo



Resource Request

- ➔ Resource request cho phép ta khai báo giới hạn tài nguyên sử dụng mà mình mong muốn container sử dụng.
- ➔ Kube Scheduler sẽ quản lý các yêu cầu tài nguyên và sẽ tránh lập lịch trên node đó, để tránh không đủ tài nguyên.
- ➔ Lưu ý: Containers được cho phép sử dụng nhiều hơn hay ít hơn tài nguyên yêu cầu. Resource Request chỉ để quản lý việc lập lịch.

Resource Request

- ➔ Memory sử dụng dưới dạng Bytes. Tuy nhiên, User được phép khai báo dưới dạng MegaBytes.
- ➔ Requests cho tài nguyên CPU được đo dưới dạng đơn vị cpu. 1vCPU bằng 1000 CPU đơn vị.

```
apiVersion: v1
kind: Pod
metadata:
name: frontend
spec:
containers:
- name: app
image: nginx
resources:
requests:
memory: "64Mi"
cpu: "250m"
```

Resource Limit

- ➔ Resource Limits được sử dụng để giới hạn tài nguyên mà container sử dụng.
- ➔ Limits được áp dụng lúc khởi chạy container.

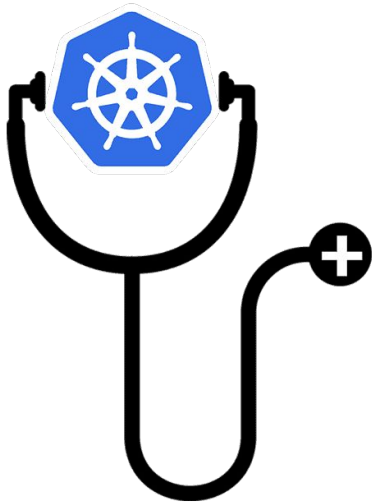
```
apiVersion: v1
kind: Pod
metadata:
name: frontend
spec:
containers:
- name: app
image: nginx
resources:
limits:
memory: "128Mi"
cpu: "500m"
```

Labs

Lab 3: manage resource request

Monitor Containers

- ➔ Container Health
- ➔ Liveness Probe
- ➔ StartUp Probe
- ➔ Readiness Probe
- ➔ Lab demo

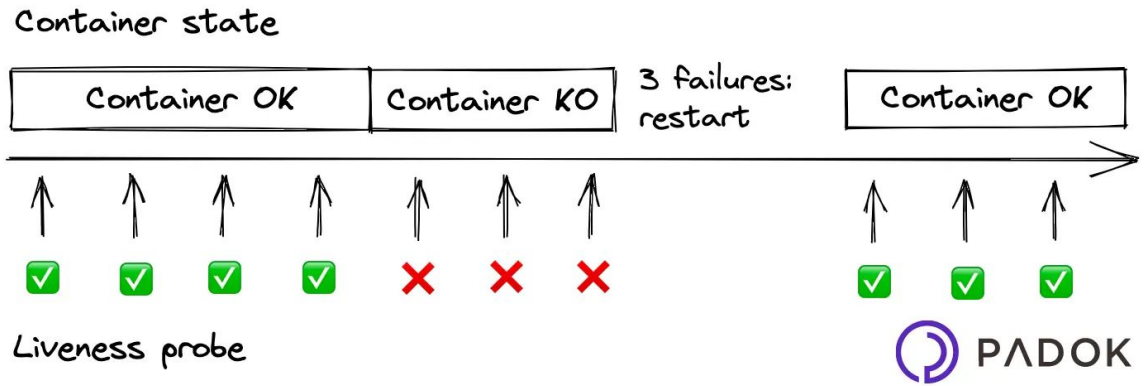


Container Health

- ➔ Kubernetes có các tính năng hỗ trợ giám sát containers.
- ➔ Chủ động giám sát cụm K8s để quyết định trạng thái container và restart tự động trong trường hợp Container lỗi.

Liveness Probe

- ➔ Liveness probe giúp xác định Container State.
- ➔ Liveness probe giúp user cải thiện và tùy chỉnh cơ chế giám sát Container.
- ➔ User có thể thực thi hai loại Liveness probes - Chạy câu lệnh trong container hoặc định kỳ gửi health check qua HTTP.



Liveness Probe

- ➔ Cấu hình Liveness qua file manifest ở phần câu lệnh container.
- ➔ initialDelaySeconds: thời gian chờ trước khi gửi một probe sau khi container chạy.
- ➔ periodSeconds: thời gian định kỳ gửi một probe.

```
livenessProbe:
exec:
command:
-some command
hereinitialDelaySeconds: 5
periodSeconds: 5
```

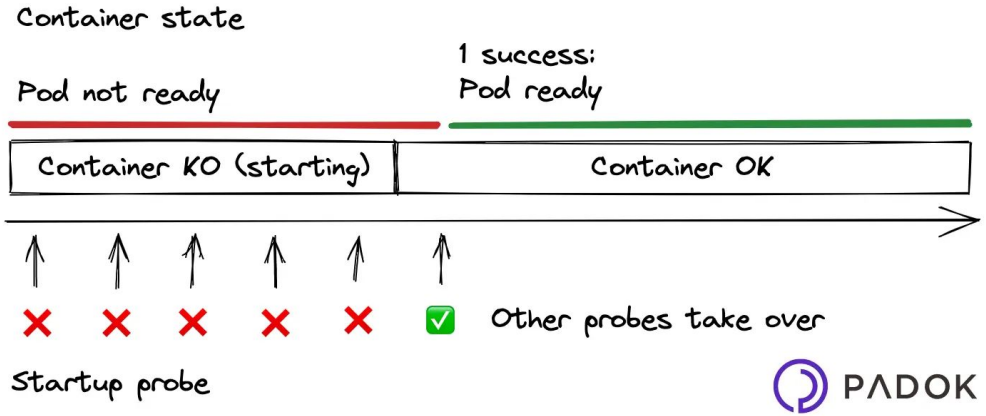
Liveness Probe

- ➔ Liveness qua HTTP requests ở tệp manifest.
- ➔ timeoutSeconds: Sau bao lâu một request lấy được response trước khi nó được xem là lỗi.

```
livenessProbe:
httpGet:
path: /health.html
port: 8080
httpHeaders:
- name: Custom-Header
value: Awesome
initialDelaySeconds: 3
periodSeconds: 3
timeoutSeconds: 1
```

StartUp Probe

- ➔ Việc cấu hình Liveness probe thì rất khó với các ứng dụng có thời gian khởi tạo lâu. StartUp probe chạy lúc container bắt đầu StartUp và ngừng chạy sau khi container chạy thành công.
- ➔ Sau khi StartUp probe đã thành công một lần, liveness probe sẽ tiếp tục để cung cấp phản hồi nhanh đối với trường hợp container bị lỗi.



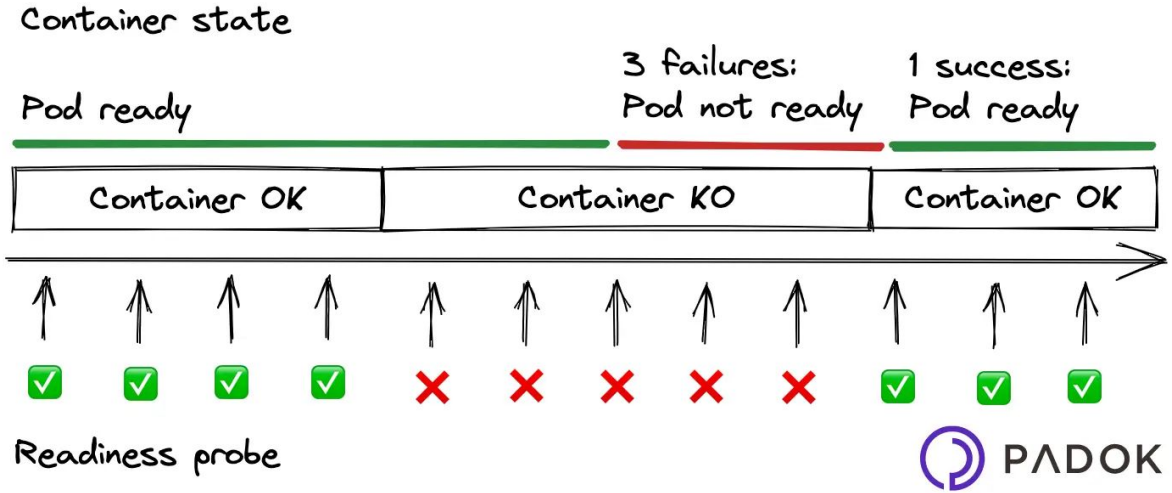
StartUp Probe

- ➔ Cấu hình StartUp probe qua HTTP request manifest file.
- ➔ failureThreshold: Khi một probe lỗi, K8s sẽ thử lại từng này lần trước khi dừng lại.
- ➔ Ứng dụng sẽ có tối đa 5 phút (30 * 10 = 300s) để kết thúc startup probe này.

```
startupProbe:
httpGet:
path: /health.html
port: 8080
failureThreshold: 30
periodSeconds: 10
```

Readiness Probe

- ➔ Readiness probe sử dụng để phát hiện nếu một container đã sẵn sàng để nhận traffic.
- ➔ Không có traffic gửi đến một Pod cho đến khi container vượt qua được Readiness Probe.



Readiness Probe

- ➔ Cấu hình Readiness probe qua anifest file.
- ➔ Cấu hình cho HTTP readiness probes cũng tương tự như cấu hình cho liveness probes.
- ➔ Readiness và liveness probes có thể được sử dụng song song trong cùng container.

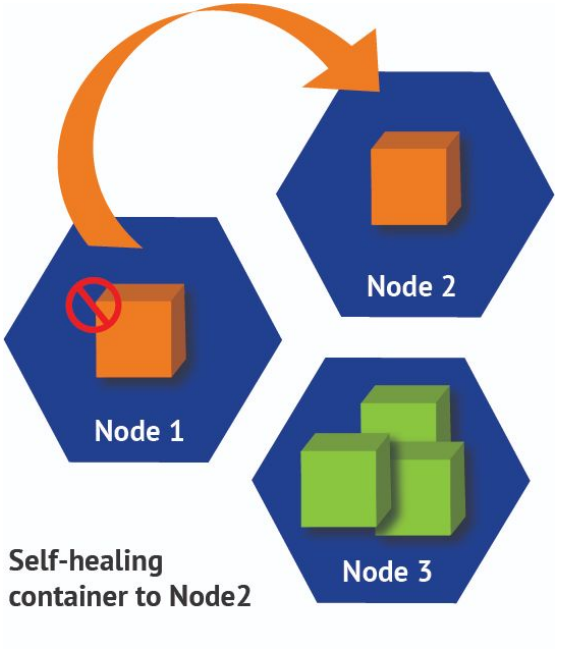
```
readinessProbe:
exec:
command:
- cat
- /tmp/healthy
initialDelaySeconds: 5
periodSeconds: 5
```

Labs

Lab 4: monitor container health

Self Healing Pods

- ➔ Container Restart Policies
- ➔ Always
- ➔ OnFailure
- ➔ Never
- ➔ Lab demo



Restart Policies

- ➔ Kubernetes có khả năng tự động restart conatainer khi nó lỗi.
- ➔ RestartPolicies cho phép tùy chỉnh cơ chế restart conainer của k8s.
- ➔ K8s có 3 loại restartPolicies - Always, OnFailure và Never

Always Restart Policy

- ➔ Always là chính sách restart mặc định trong k8s.
- ➔ Với Always Policy, containers luôn restart ngay cả khi container đã chạy hoàn tất thành công.
- ➔ Sử dụng policy này khi mà các container nên luôn ở trạng thái running.

OnFailure Restart Policy

- ➔ OnFailure chỉ thực hiện khi tiến trình container kết thúc với mã lỗi.
- ➔ Nó cũng thực hiện nếu liveness probe của container xác định container là unhealthy.
- ➔ Sử dụng policy này trên ứng dụng cần chạy thành công và sau đó stop.

Never Restart Policy

- ➔ Never cho phép container không bao giờ restart kể cả khi container liveness probe lỗi.

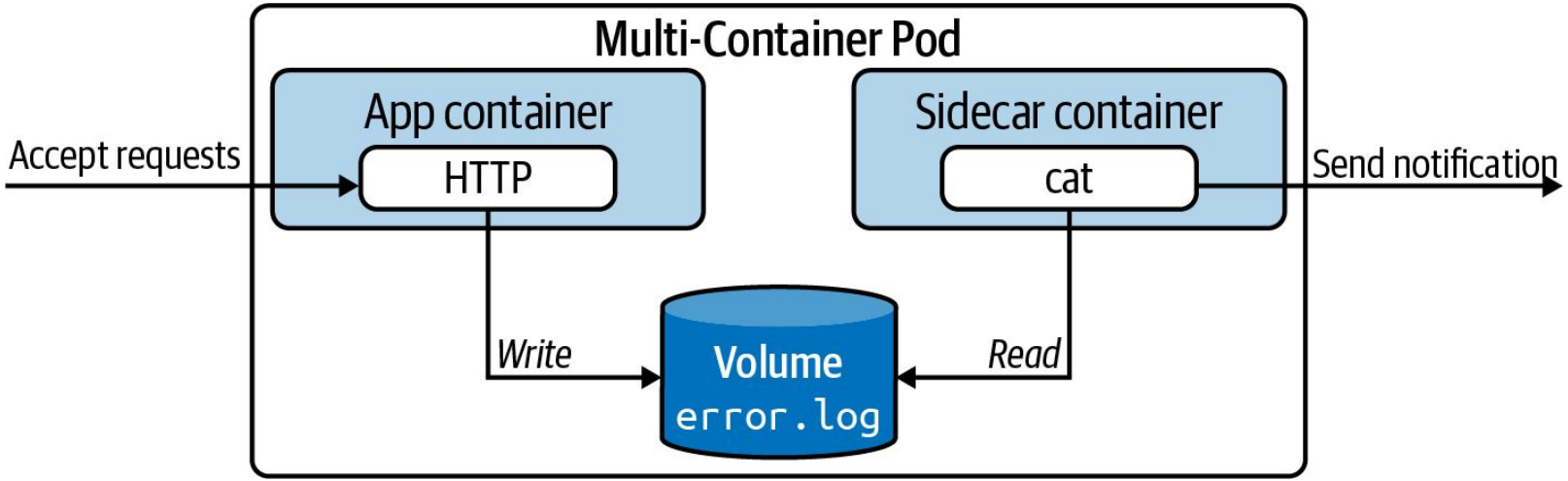
- ➔ Sử dụng policy này trên ứng dụng chạy một lần và không cần chạy lại một cách tự động.

Labs

Lab 5: self healing pods

Multi Container Pods

- ➔ Multi Container Pod là gì
- ➔ Giao tiếp chéo các container trong Pod
- ➔ Ví dụ
- ➔ Demo lab



Multi Container Pods

- ➔ Kubernetes Pods có thể có một hoặc nhiều Containers.
- ➔ Trong Pod có nhiều container, các container chia sẻ chung tài nguyên như network và storage, có thể giao tiếp qua Local Host.
- ➔ Lưu ý: Tốt nhất ta nên giữ các container trong các Pod riêng biệt, trừ khi chúng ta muốn các container chia sẻ chung tài nguyên.

Cross Container Communication

- ➔ Các Container trong cùng một Pod, có thể tương tác với tài nguyên dùng chung.
- ➔ Network : Các container chia sẻ cùng network và giao tiếp trên Port bất kỳ, trừ khi port được expose đến cụm.
- ➔ Storage: Các container có thể sử dụng Volume chung để chia sẻ dữ liệu trọng một Pod.

Labs

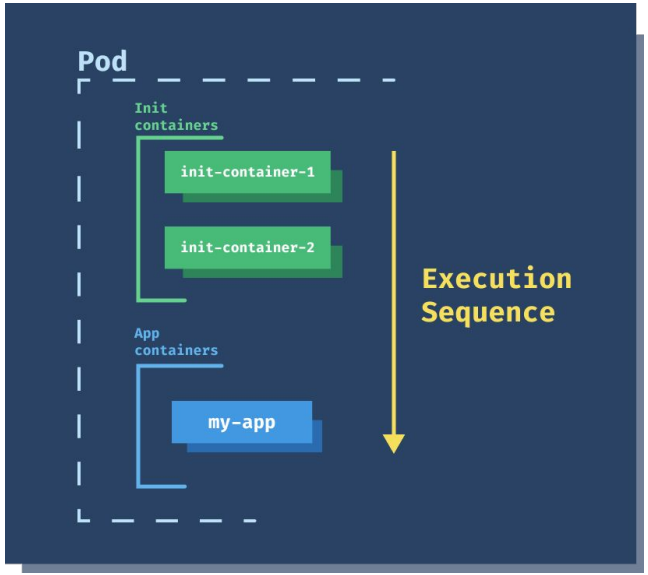
Lab 6: multi container pods

Container Initialization

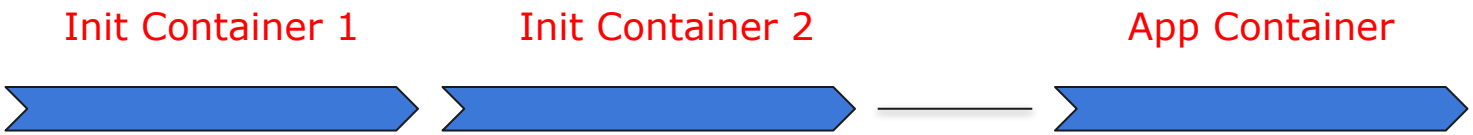
- ➔ Init Containers là gì
- ➔ Khi nào sử dụng Init Container
- ➔ Demo lab

Init Containers

- ➔ Init Containers là các container có nhiệm vụ đặc thù chạy trước các container ứng dụng trong một Pod.
- ➔ Init Container chỉ chạy một lần trong suốt quá trình start-up của Pod.
- ➔ Init Containers có thể chứa các script ứng dụng hoặc cấu hình không có trong image của ứng dụng.
- ➔ User có thể khai báo nhiều Init Container trong Pod.



Init Containers



Container chạy đến khi hoàn thành trước khi chuyển qua Init Container tiếp theo hoặc Container ứng dụng bắt đầu.

User có thể có nhiều Init Container và chúng sẽ thực thi tuần tự.

Containers của ứng dụng chỉ bắt đầu khi tất cả các Init Container đều hoàn thành.

Init Containers - Use Case

- ➔ Để thiết lập tập lệnh khởi tạo ứng dụng hoặc các setup ban đầu.
- ➔ Init Containers cung cấp cơ chế chặn hoặc trì hoãn việc khởi động container ứng dụng cho đến khi đáp ứng một loạt các điều kiện tiên quyết.
- ➔ Init Containers có thể chạy các tiện ích hoặc code tùy chỉnh một cách an toàn, nếu không sẽ làm cho image container ứng dụng kém an toàn hơn.
- ➔ Điền Dữ liệu vào Volume dùng chung trước khi Khởi động ứng dụng.

Labs

Lab 7: pod with init container

Section 2:

Summary

Mục tiêu

Kết thúc bài học, học viên cần đạt được các kĩ năng sau:

- Có khả năng làm việc cụm Kubernetes với kubectl.
- Có hiểu biết về Pods và Containers trong Kubernetes.
- Nắm được cách quản lý cấu hình ứng dụng, quản lý tài nguyên containers.
- Có thể cấu hình giám sát được tình trạng của container trong cụm Kubernetes.
- Biết cách sử dụng Init Containers để tùy biến triển khai ứng dụng.

Tài liệu tham khảo

Thank you