



Devops on AWS for beginner

Instructor – Linh Nguyen

Engineering Consultant, AWS Cloud Solution Architect

Giới thiệu Kubernetes (k8s)

"Không có việc gì khó, chỉ sợ không biết làm!"

Copyright@Linh Nguyen on Udemy
All right reserved

Target

- Hiểu được Kubernetes là gì và tại sao nó được sử dụng phổ biến.
- Các thành phần cơ bản của Kubernetes.
- Các concept cơ bản trong Kubernetes.
- Thao tác cơ bản với Kubernetes sử dụng Minikube.
- Triển khai một cluster đơn giản sử dụng KOPS và EC2.
- Triển khai một số ứng dụng đơn giản lên Kubernetes cluster.

Copyright@Linh Nguyen on Udemy
All right reserved

Container Orchestration là gì?

Container Orchestration là quá trình tự động hóa việc triển khai, quản lý, mở rộng, di chuyển và giám sát các container. Các công nghệ như Kubernetes, Docker Swarm và Mesos được thiết kế để giúp trong việc quản lý các container.

Các công việc cụ thể mà container orchestration có thể thực hiện bao gồm:

- Quản lý vòng đời của container
- Cung cấp khả năng mở rộng tự động (Auto scaling).
- Cung cấp khả năng phục hồi (recovery).
- Load Balance trên các container
- Cung cấp kênh giao tiếp giữa các container (Networking).
- Cung cấp cơ chế bảo mật cho các container (Security).
- Quản lý việc cấp phát tài nguyên cho các container.

Copyright@Linh Nguyen on Udemy
All right reserved

Các công nghệ Container Orchestration

Có nhiều công nghệ Container Orchestration và Kubernetes là một trong số đó.



Giới thiệu Kubernetes.

Kubernetes – Container Orchestration phát triển bởi Google.

Kubernetes giúp quản lý và tự động hóa việc triển khai các container, đồng thời cung cấp khả năng mở rộng và linh hoạt cho ứng dụng của bạn. Nó là một công cụ quan trọng trong việc xây dựng hệ thống phân tán và hiệu quả.



kubernetes

Copyright@Linh Nguyen on Udemy
All right reserved

Giới thiệu Kubernetes – lịch sử của Kubernetes

1. 2003-2004: Sự ra đời của Hệ thống Borg:

1. Google giới thiệu **Hệ thống Borg** vào khoảng năm 2003-2004. Nó là một hệ thống quản lý **cluster** nội bộ quy mô lớn, chạy hàng trăm nghìn tasks từ nhiều ứng dụng khác nhau trên nhiều cụm máy chủ.

2. 2013: Từ Borg đến Omega:

1. Sau Borg, Google giới thiệu **Omega**, một hệ thống quản lý **cluster** linh hoạt có thể mở rộng cho các cụm máy tính lớn.

3. 2014: Google giới thiệu Kubernetes:

1. Giữa năm 2014, Google giới thiệu **Kubernetes** như một phiên bản mã nguồn mở của Borg.
2. Ngày 7 tháng 6, Kubernetes được push lên GitHub là phiên bản đầu tiên.
3. Ngày 10 tháng 7, Microsoft, RedHat, IBM và Docker tham gia cộng đồng Kubernetes.

4. 2015: Năm của Kube v1.0 & CNCF:

1. Tháng 7, Kubernetes v1.0 được phát hành.
2. Google hợp tác với Linux Foundation để thành lập **Cloud Native Computing Foundation (CNCF)**. CNCF nhằm mục đích xây dựng hệ sinh thái bền vững và thúc đẩy cộng đồng xung quanh các dự án chất lượng cao liên quan đến kiến trúc microservices.

5. 2016: Năm Kubernetes trở nên official:

1. Tháng 2, Helm - trình quản lý gói của Kubernetes - được phát hành lần đầu.
2. Kubernetes tiếp tục phát triển với sự tham gia của nhiều công ty như Deis, OpenShift, Huawei và Gondor.



kubernetes

Tại sao Kubernetes được sử dụng nhiều?

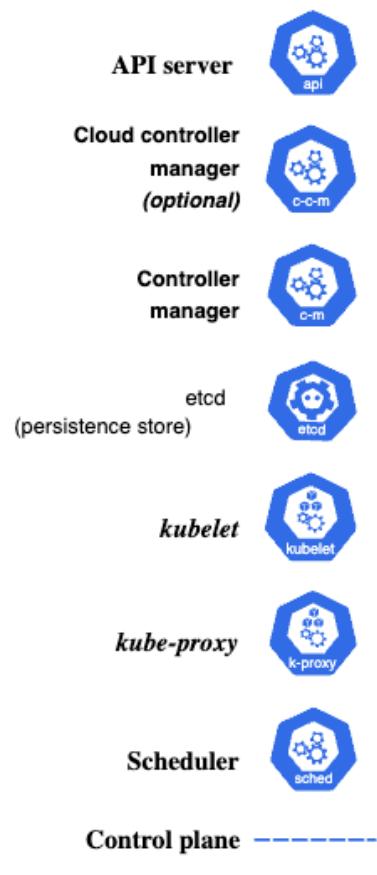
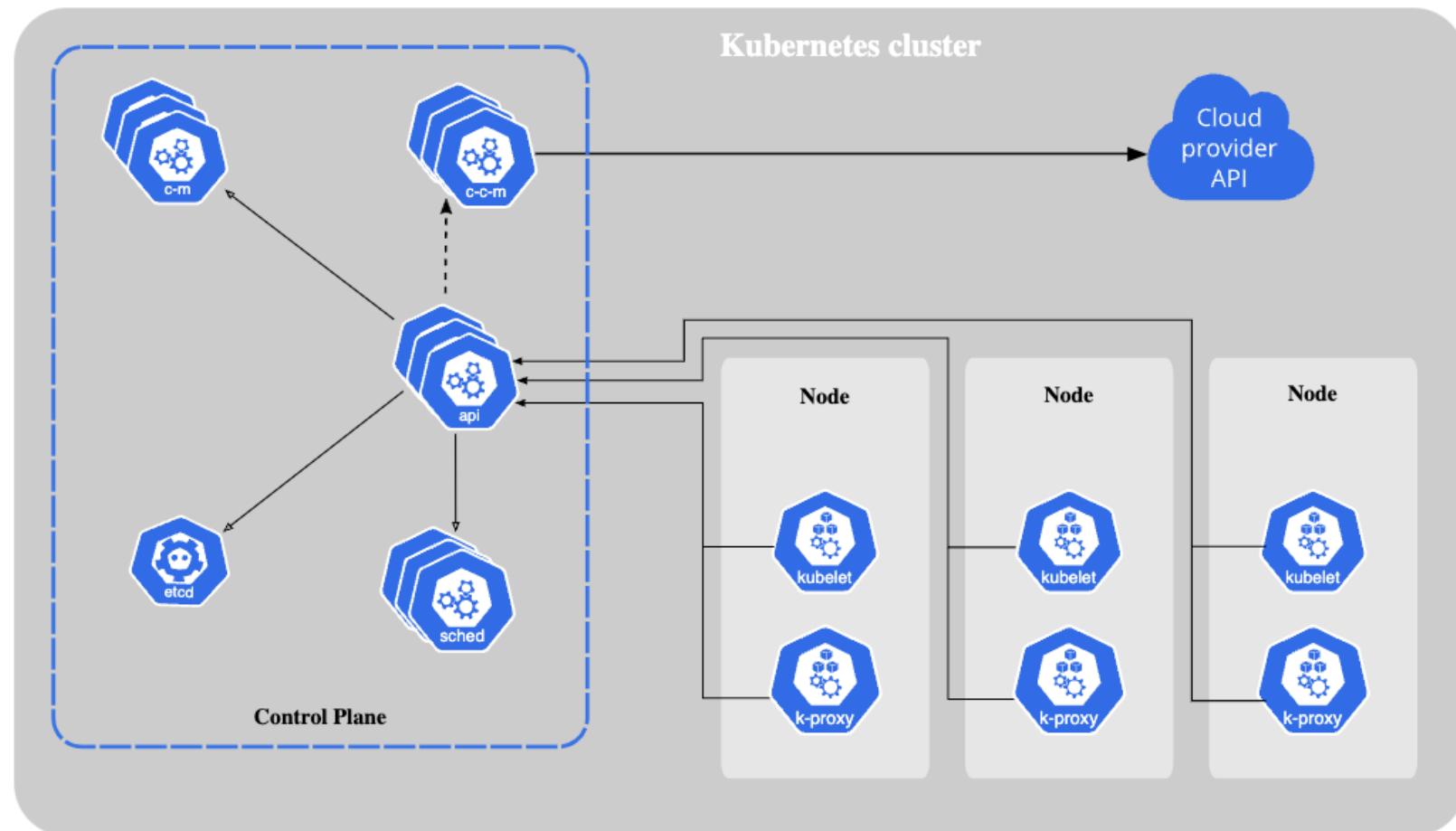
- Tự động hoá:** Kubernetes giúp tự động hoá việc triển khai, mở rộng, và quản lý các ứng dụng container. Bạn có thể định nghĩa các rule & policy để Kubernetes tự động thực hiện các thay đổi.
- Mở rộng linh hoạt:** Kubernetes cho phép bạn mở rộng ứng dụng dễ dàng bằng cách scale-out các container mới mà không làm gián đoạn dịch vụ.
- Cô lập và độ tin cậy:** Các container trong Kubernetes được cô lập với nhau, đảm bảo rằng một container không ảnh hưởng đến container khác. Nếu một container gặp sự cố, Kubernetes có thể tự động khởi động lại nó.
- Phân tán và hiệu quả:** Kubernetes giúp bạn triển khai ứng dụng trên nhiều máy chủ, tận dụng tối đa tài nguyên.
- Hỗ trợ cộng đồng mạnh mẽ:** Kubernetes là một dự án nguồn mở, có cộng đồng lớn mạnh, nhiều tài liệu hướng dẫn. Dễ dàng tìm kiếm giải pháp và hỗ trợ từ cộng đồng.



kubernetes

Copyright@Linh Nguyen on Udemy
All right reserved

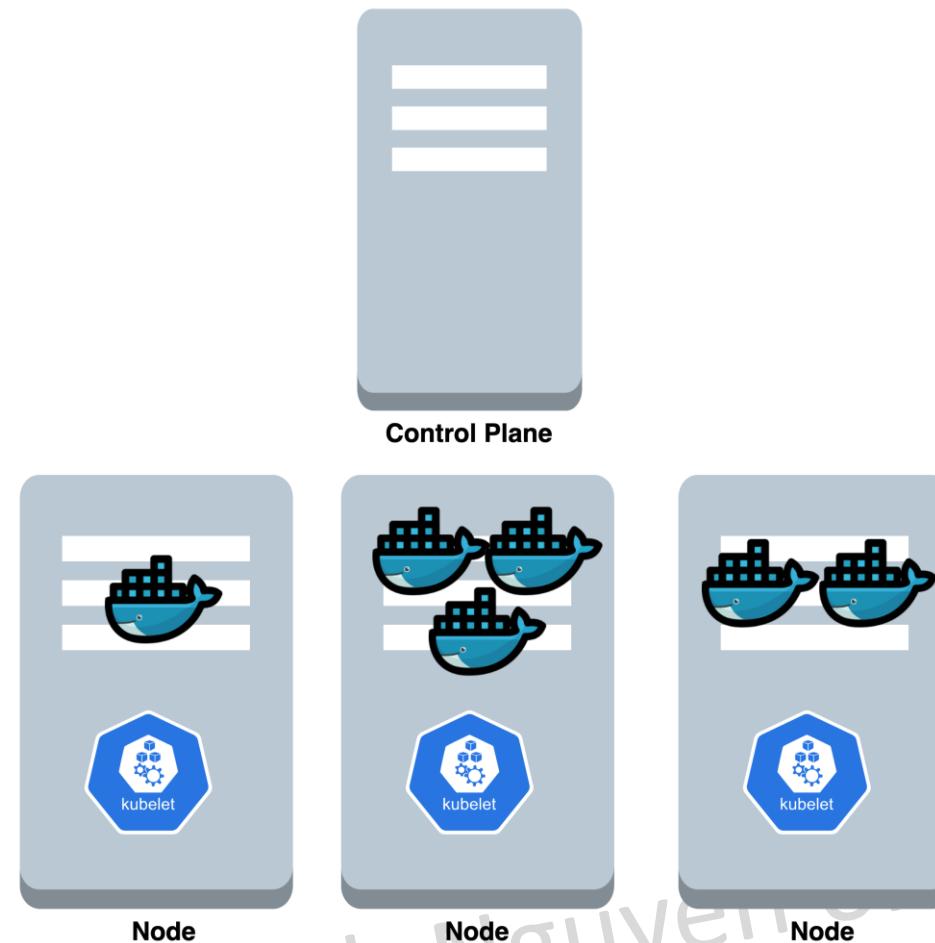
Các thành phần cơ bản của một Kubernetes cluster



*Nguồn: <https://kubernetes.io/docs/concepts/overview/components/>

Các thành phần cơ bản của một Kubernetes cluster

Mô hình đơn giản của một K8s cluster gồm Master nodes (Control Plane) và Worker nodes.



Các thành phần cơ bản của một Kubernetes cluster

Các thành phần bên trong Control Plane (Master node).



API Server: Entrypoint của K8s Cluster



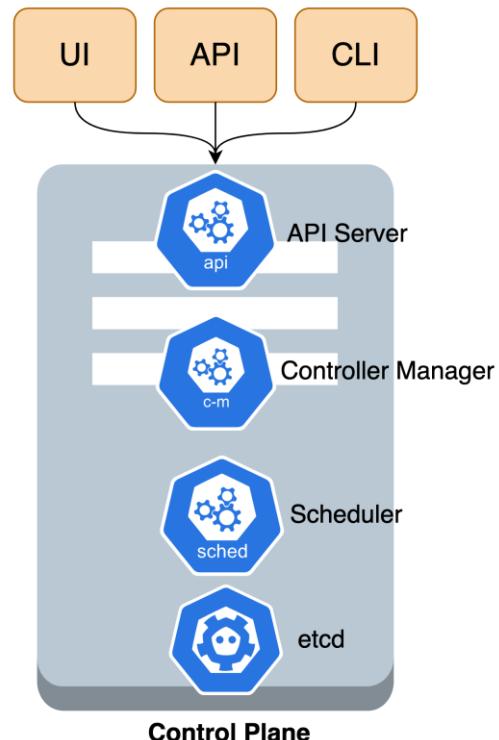
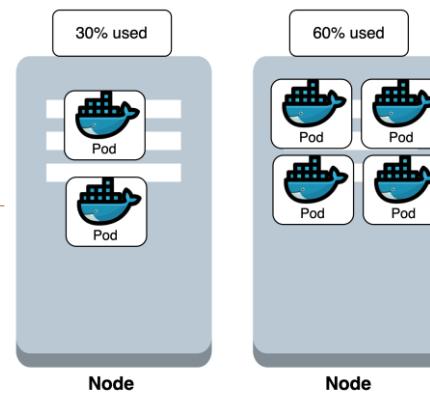
Controller Manager: Theo dõi tất cả những hoạt động đang diễn ra trên Cluster.



Scheduler: đảm bảo việc phân phối Pod



etcd: lưu trữ & đảm bảo tính nhất quán của cluster data (key:value).



Các thành phần cơ bản của một Kubernetes cluster

Các thành phần bên trong Worker node.



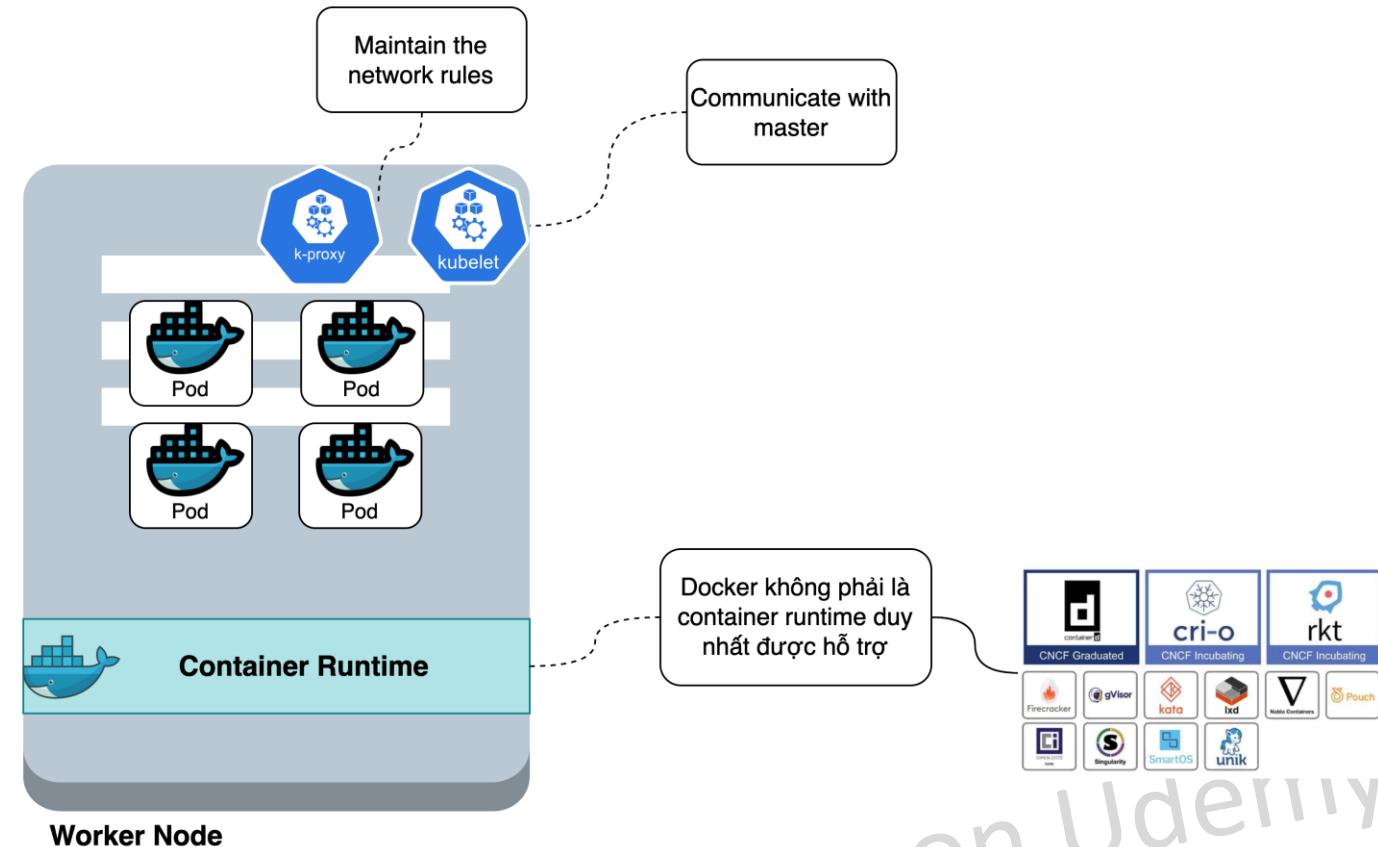
kubelet: agent chạy trên Worker node có nhiệm vụ giao tiếp với Master



k-proxy: phụ trách quản lý network communication

Container Runtime

runtime giúp chạy các container bên trên worker node.



Các thành phần cơ bản của một Kubernetes cluster

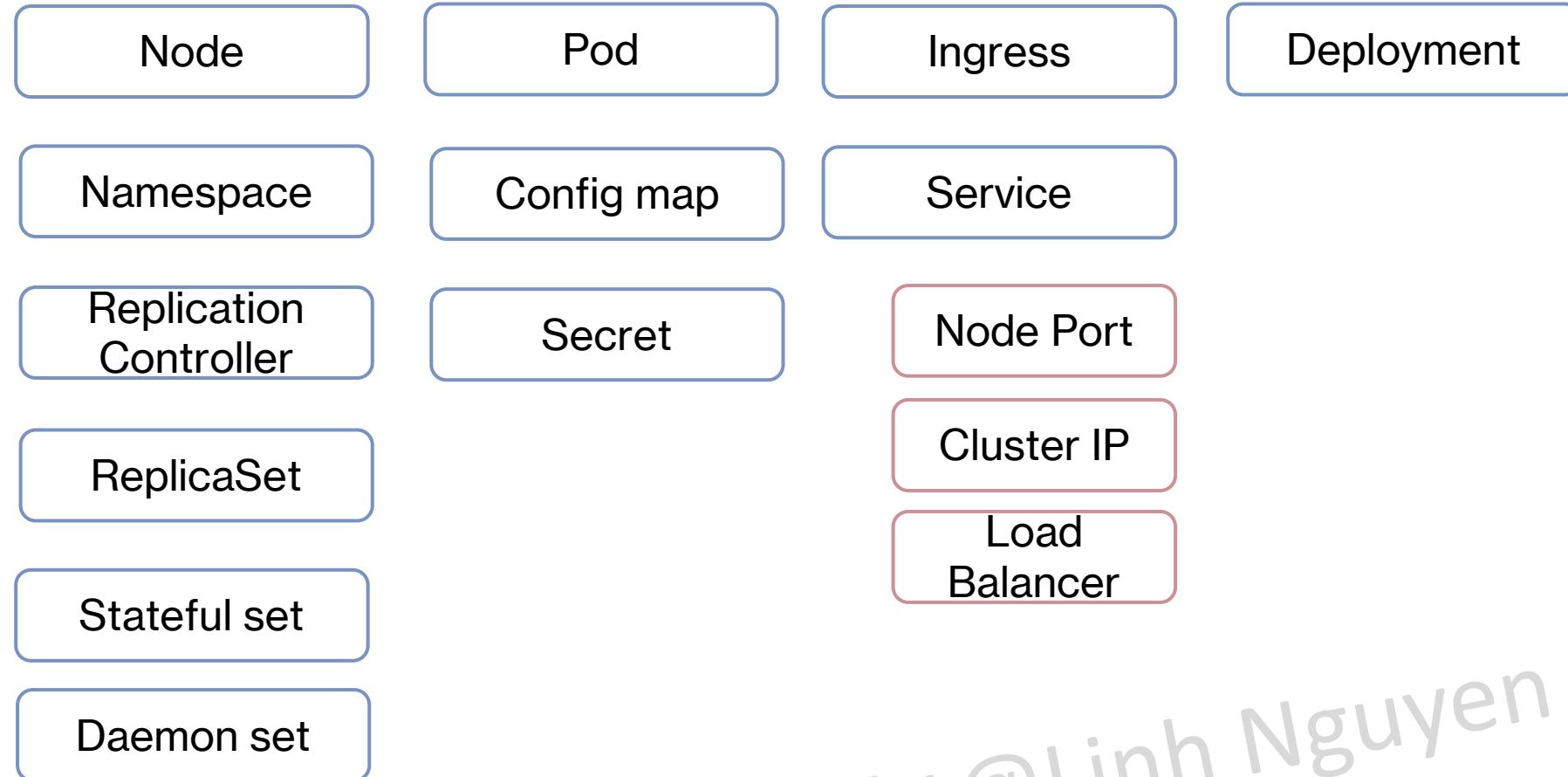
kubectl: command line tool giúp tương tác với Kubernetes cluster

```
kubectl run my-hello-kube  
kubectl get services  
kubectl describe nodes my-node  
kubectl apply -f myconfig.yaml
```



Copyright@Linh Nguyen on Udemy
All right reserved

Các khái niệm trọng Kubernetes cần nắm vững.



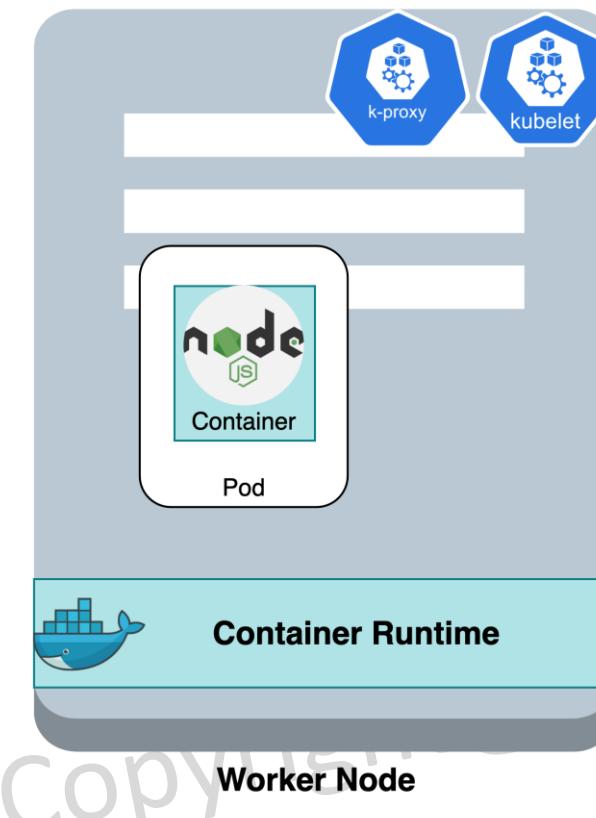
Copyright@Linh Nguyen on Udemy
All right reserved

Pod

Kubernetes chỉ có thể chạy các ứng dụng đã được build ra thành Docker image.

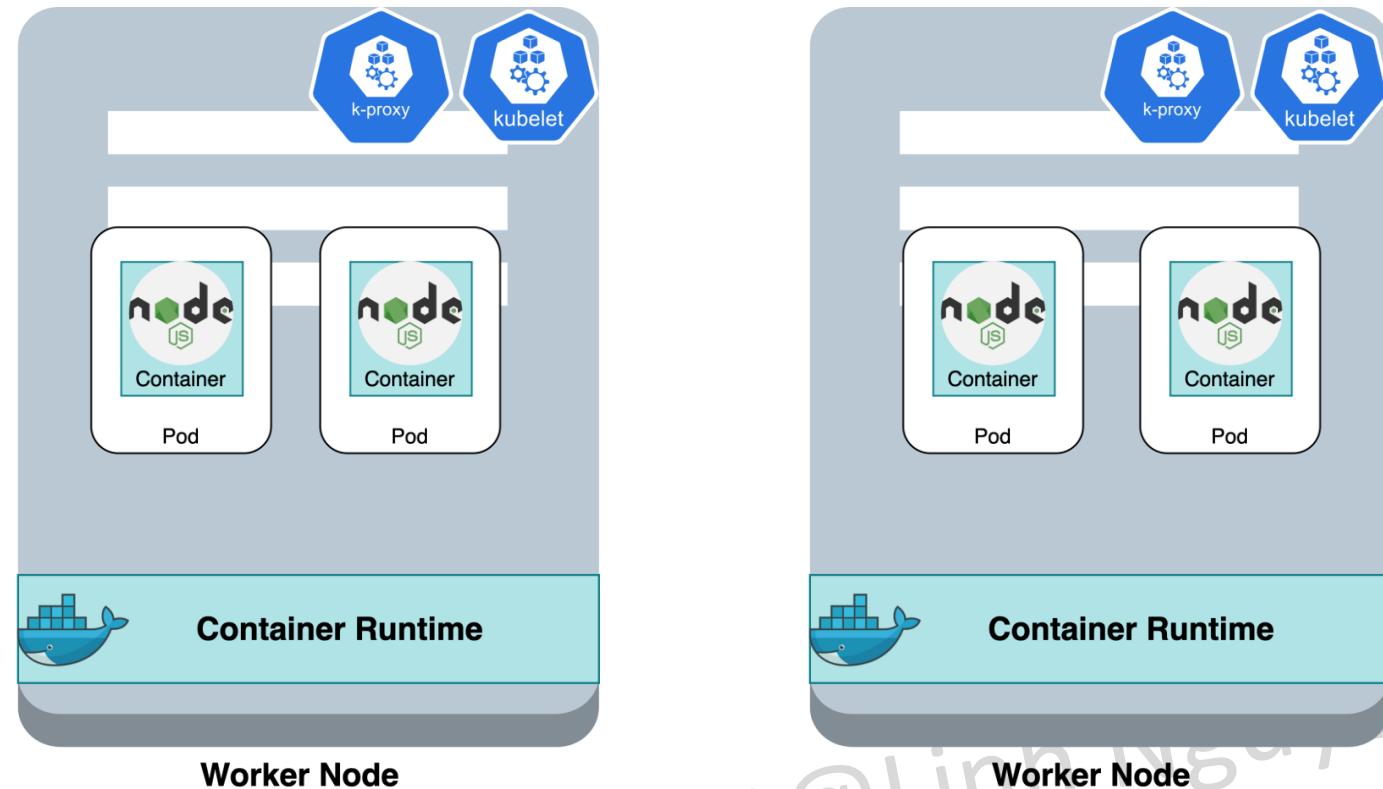
Image cần được chuẩn bị trước trên các Container Registry như ECR, DockerHub,...

Pod: đơn vị quản lý nhỏ nhất của Kubernetes. Một pod có thể chứa 1 hoặc nhiều container.



Pod

Khi scaling một ứng dụng, scale thêm pod hoặc kết hợp cả việc thêm node mới.



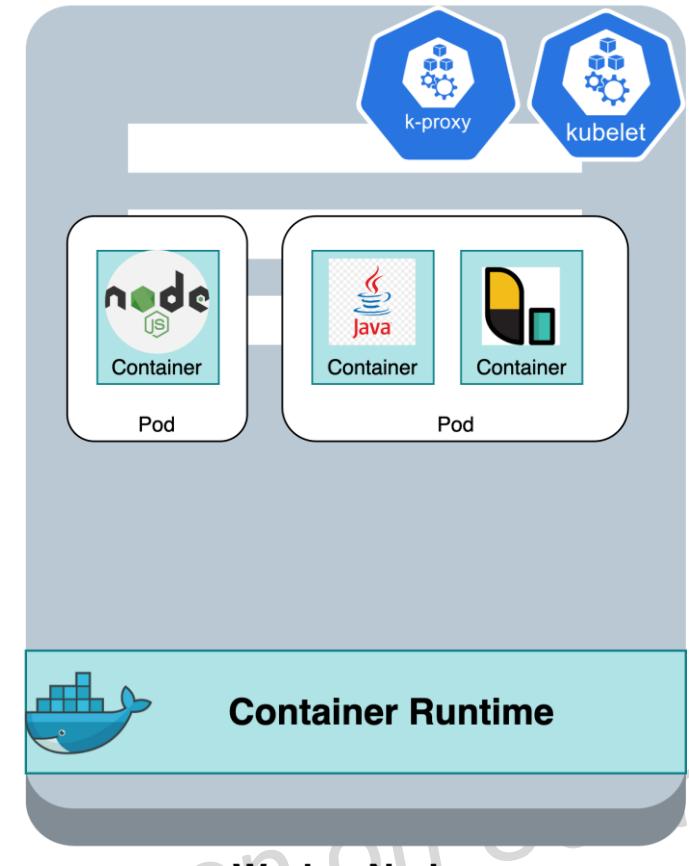
Pod

Có thể triển khai nhiều hơn 1 container trên 1 pod, vd 1 container chính & 1 container phụ làm nhiệm vụ hỗ trợ (sidecar).

VD hình bên có 1 Pod chạy 2 container, container chính chạy app Java và container phụ chạy Logstash có nhiệm vụ collect log.

Lưu ý: Không triển khai 2 container cùng loại trên 1 pod.

Containers trên cùng 1 pod sẽ share storage, network và có chung life cycle.



Lab1 – Cài đặt và sử dụng Kubectl, Minikube.

Yêu cầu:

*Tham khảo link: <https://minikube.sigs.k8s.io/docs/start/>

- Cài đặt Docker desktop (bỏ qua nếu đã cài).
- Cài đặt Kubectl trên máy cá nhân.
- Cài đặt Minikube trên máy cá nhân hoặc trên một máy ảo, EC2.
- Chạy câu lệnh sau:

>kubectl get nodes

Copyright@Linh Nguyen on Udemy
All right reserved

Lab 2 – Chạy một pod đơn giản sử dụng kubectl.

Sử dụng minikube để chạy một pod đơn giản từ image **nginx**.

Kiểm tra thông tin pod.

Xoá pod.

Copyright@Linh Nguyen on Udemy
All right reserved

Sử dụng file Yaml để triển khai resource lên Kubernetes cluster

Trong thực tế chúng ta sẽ không sử dụng command cho các tác vụ deploy từng resource riêng lẻ, thay vào đó sẽ định nghĩa chúng thông qua file Yaml.

Sau khi có được file config như mong muốn, chúng ta tiến hành apply cho kubernetes cluster thông qua lệnh:

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

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
spec:
  containers:
  - name: nginx
    image: nginx:1.14.2
    ports:
    - containerPort: 80
```

Copyright@Linh Nguyen on Udemy
All right reserved

Lab 3 – sử dụng file Yaml để triển khai Pod.

Tạo file Yaml định nghĩa pod.

Apply file config cho Kubernetes.

Kiểm tra pod được tạo ra.

Xoá toàn bộ pod.

Copyright@Linh Nguyen on Udemy
All right reserved

Namespace trong k8s

Namespace trong Kubernetes là một cách logical để phân chia tài nguyên cluster giữa các nhóm resource.

Mỗi namespace cung cấp một phạm vi cho các tên của tài nguyên. Tài nguyên trong một namespace là riêng biệt và không thể truy cập được từ các namespace khác.

Namespace không thể lồng nhau.

Khi tạo resource mà không chỉ định namespace, nó sẽ thuộc “**default**” namespace.

Namespace trong k8s

Use case của Namespace

- Khi bạn muốn phân chia tài nguyên của một cluster Kubernetes giữa nhiều người dùng hoặc nhóm người dùng.
- Khi bạn muốn áp dụng các chính sách quản lý quyền truy cập khác nhau cho các phần khác nhau của cluster.
- Khi bạn muốn phân biệt các môi trường phát triển khác nhau, như dev, test, và production trên cùng một Cluster.
- Khi bạn muốn cài đặt các ứng dụng hoặc dịch vụ khác nhau. Mỗi ứng dụng hoặc dịch vụ có thể được triển khai trong namespace riêng của nó, giúp giảm thiểu khả năng xung đột và tăng cường bảo mật.

Copyright@Linh Nguyen on Udemy
All right reserved

Namespace trong k8s

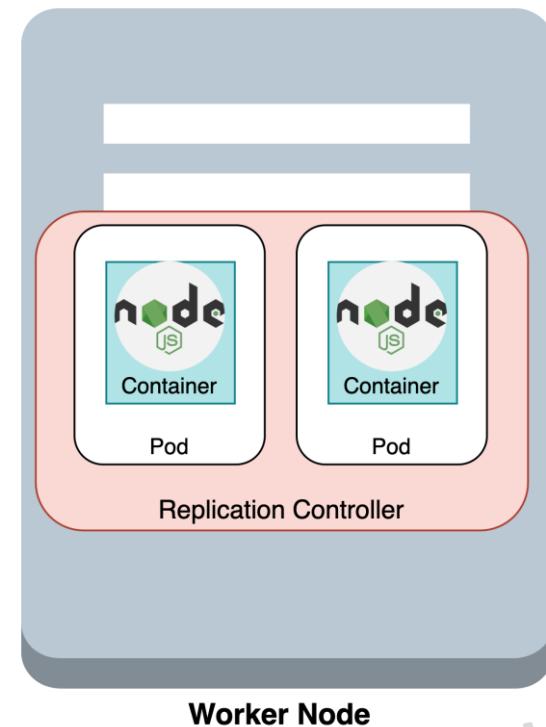
Một số resource không thể đưa vào trong namespace (không thuộc name space):

- Nodes: Mỗi node đại diện cho một máy chủ trong cluster Kubernetes và không thuộc về bất kỳ namespace nào.
- PersistentVolumes: Chúng tồn tại độc lập với bất kỳ namespace nào.
- Roles và ClusterRoles: Đây là tài nguyên liên quan đến quản lý quyền truy cập trong Kubernetes.
- RoleBindings và ClusterRoleBindings: Đây là tài nguyên liên quan đến việc gán quyền truy cập.

Copyright@Linh Nguyen on Udemy
All right reserved

Replication Controller & ReplicaSet

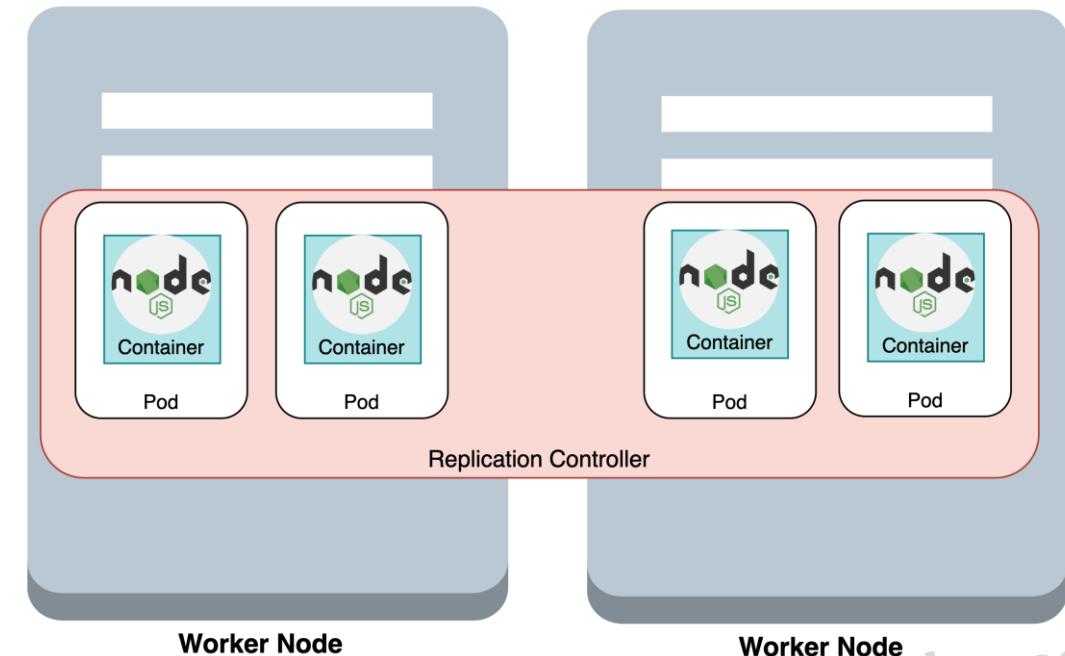
Replication Controller & Replica set có nhiệm vụ đảm bảo số lượng pod được run, tự tạo lại pod khi có vấn đề xảy ra.



Copyright@Linh Nguyen on Udemy
All right reserved

ReplicationController & ReplicaSet

Replication Controller & Replica trong trường hợp cluster có nhiều worker node.



ReplicationController & ReplicaSet

ReplicaSet có thêm thông tin về selector, cho phép chọn pod với label nhất định để điều hướng traffic

```

1  apiVersion: v1
2  kind: ReplicationController
3  metadata:
4    name: nginx-controller
5  spec:
6    replicas: 2
7    template:
8      metadata:
9        labels:
10       app: nginx
11    spec:
12      containers:
13        - name: nginx
14          image: nginx:1.14.2
15        ports:
16          - containerPort: 80
17

```

ReplicationController

```

1  apiVersion: apps/v1
2  kind: ReplicaSet
3  metadata:
4    name: nginx-set
5  spec:
6    replicas: 2
7    selector:
8      matchLabels:
9        app: nginx
10   template:
11     metadata:
12       labels:
13         app: nginx
14   spec:
15     containers:
16       - name: nginx
17         image: nginx:1.14.2
18       ports:
19         - containerPort: 80

```

ReplicaSet

Lab 4 – Tạo một ReplicaSet và apply cho kubernetes.

- Tạo một replica set với 4 pods nginx.
- Apply cho cluster
- Kiểm tra các pod được tạo ra.
- Thủ delete 1 pod sử dụng câu lệnh.
- Kiểm tra khả năng tự phục hồi.

Copyright@Linh Nguyen on Udemy
All right reserved

Deployment

Trong Kubernetes, Deployment là một đối tượng API cung cấp cơ chế cập nhật và quản lý trạng thái cho các version của ứng dụng. Deployment sẽ kiểm soát và cập nhật các Pod để đảm bảo trạng thái của chúng luôn ổn định và khớp với trạng thái mong muốn mà bạn đã khai báo cấu hình.

*Ngoài việc đảm bảo số lượng pod được duy trì như ReplicaSet, Deployment còn hỗ trợ phương pháp triển khai cũng như khả năng quản lý version cho phép rollback về version trước đó khi có sự cố.

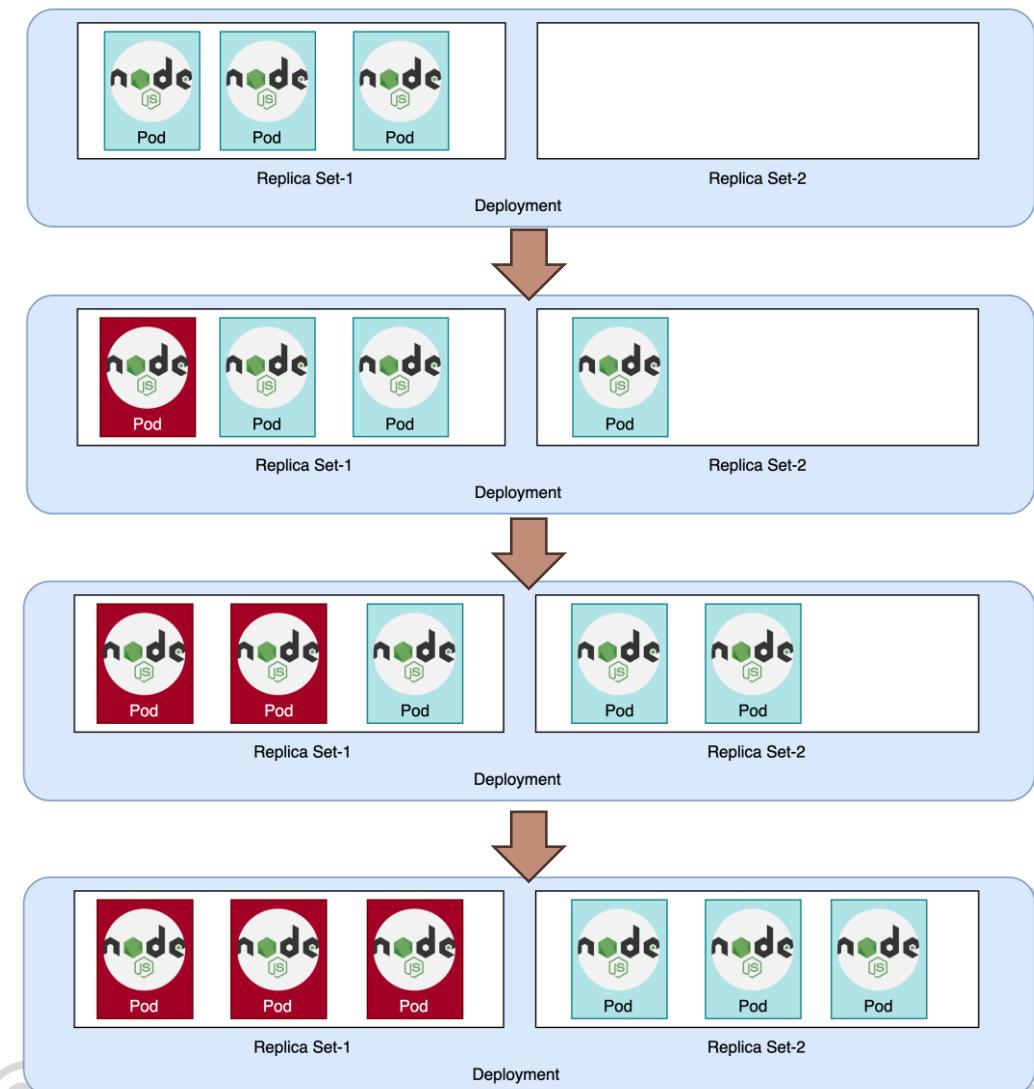
Copyright@Linh Nguyen on Udemy
All right reserved

Deployment - Update & Rollback.

Deployment có các chiến lược deploy vd:
Recreate, RollingUpdate

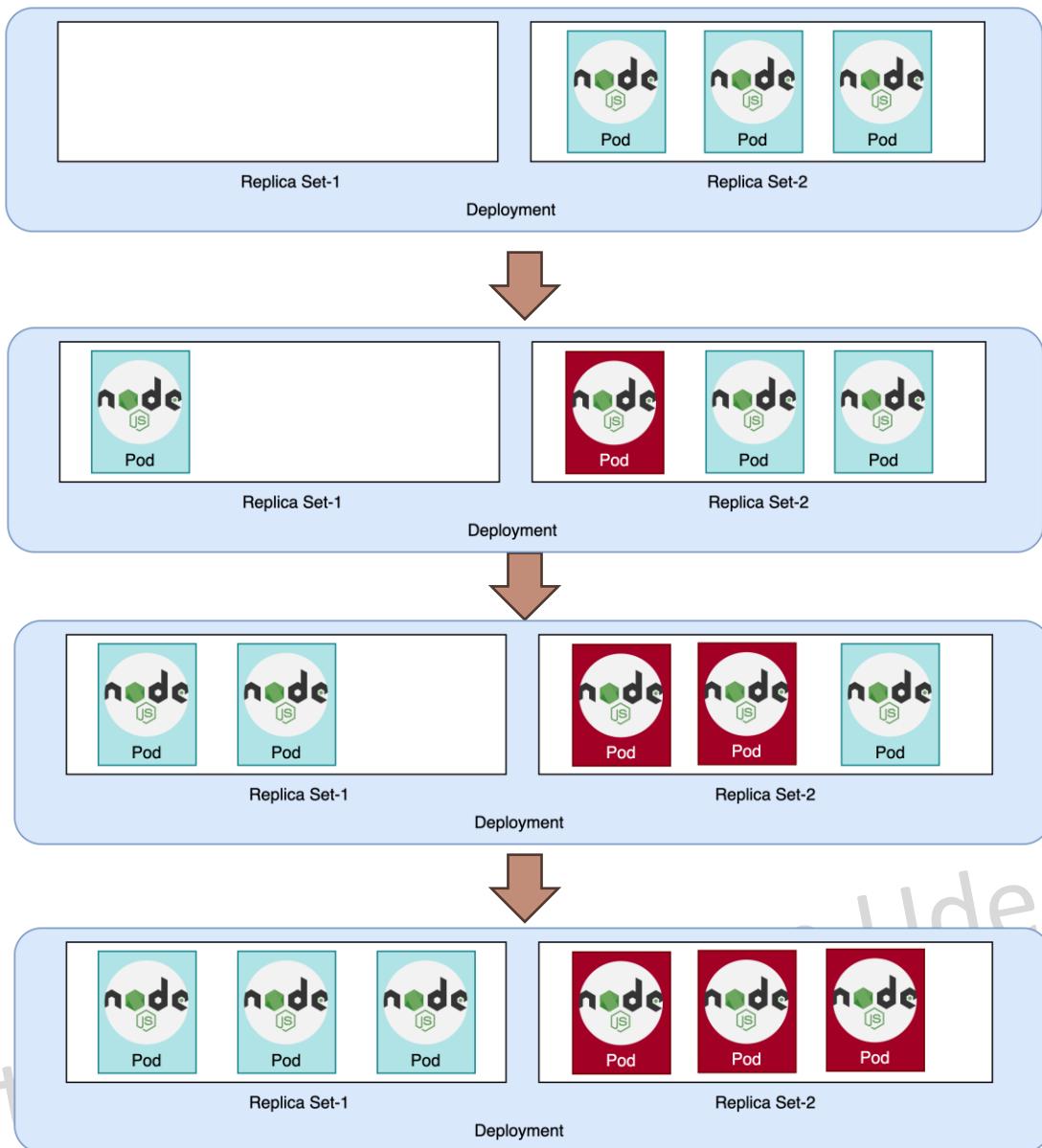
Khi Rolling Update deployment, k8s sẽ tạo
ra một Replica set mới.

```
kubectl apply -f deployment-file.yaml
kubectl rollout status deployment/<deployment-name>
kubectl rollout history deployment/<deployment-name>
```



Deployment - Update & Rollback.

Rollback Deployment



```
kubectl rollout status deployment/<deployment-name>
kubectl rollout history deployment/<deployment-name>
kubectl rollout undo deployment/<deployment-name>
```

Deployment - Update & Rollback - Summary

create

```
>kubectl create -f deployment-file.yaml
```

get

```
>kubectl get deployments
```

update

```
>kubectl apply -f deployment-file.yaml
```

```
>kubectl set image deployment/my-deployment nginx=nginx:v1.2.3
```

```
>kubectl edit deployment my-deployment --record
```

status

```
>kubectl rollout status deployment/my-deployment
```

```
>kubectl rollout history deployment/my-deployment
```

rollback

```
>kubectl rollout undo deployment/my-deployment
```

Lab 5 – Deployment

- Tạo một file cấu hình Deployment cho service nginx
 - Số lượng pod: 3
 - Phương pháp deploy: RollingUpdate.
- Apply Deployment cho Kubernetes cluster.
- Get thông tin của Deployment & Replicaset
- Tiến hành Update version cho image nginx
- Apply thay đổi cho Kubernetes cluster
- Get thông tin của Deployment & Replicaset

Copyright@Linh Nguyen on Udemy
All right reserved

Lab 5 – Deployment Update & Rollback

- List ra các version history của Deployment bằng câu lệnh:

```
>kubectl rollout history deployment/ nginx-deployment
```

- Tiến hành rollback về version cũ sử dụng câu lệnh:

```
>kubectl rollout undo deployment/ nginx-deployment
```

- Trong quá trình rollback, chạy nhanh câu lệnh sau để kiểm tra trạng thái:

```
>kubectl rollout status deployment/ nginx-deployment
```

Lab 5 – Deployment Update & Rollback

Giải thích về 2 thông số: maxSurge & maxUnavailable

In a Kubernetes Deployment, maxSurge and maxUnavailable are two optional fields that you can specify in the rollingUpdate strategy. They control the rolling update process when you perform a deployment.

- **maxSurge**: This is the maximum number of Pods that can be scheduled above the desired number of Pods. It can be an absolute number (for example, 5) or a percentage of desired Pods (for example, 10%). The value cannot be 0 if maxUnavailable is 0. The default value is 25%.
- **maxUnavailable**: This is the maximum number of Pods that can be unavailable during the update. It can be an absolute number (for example, 5) or a percentage of desired Pods (for example, 10%). The value cannot be 0 if maxSurge is 0. The default value is 25%.

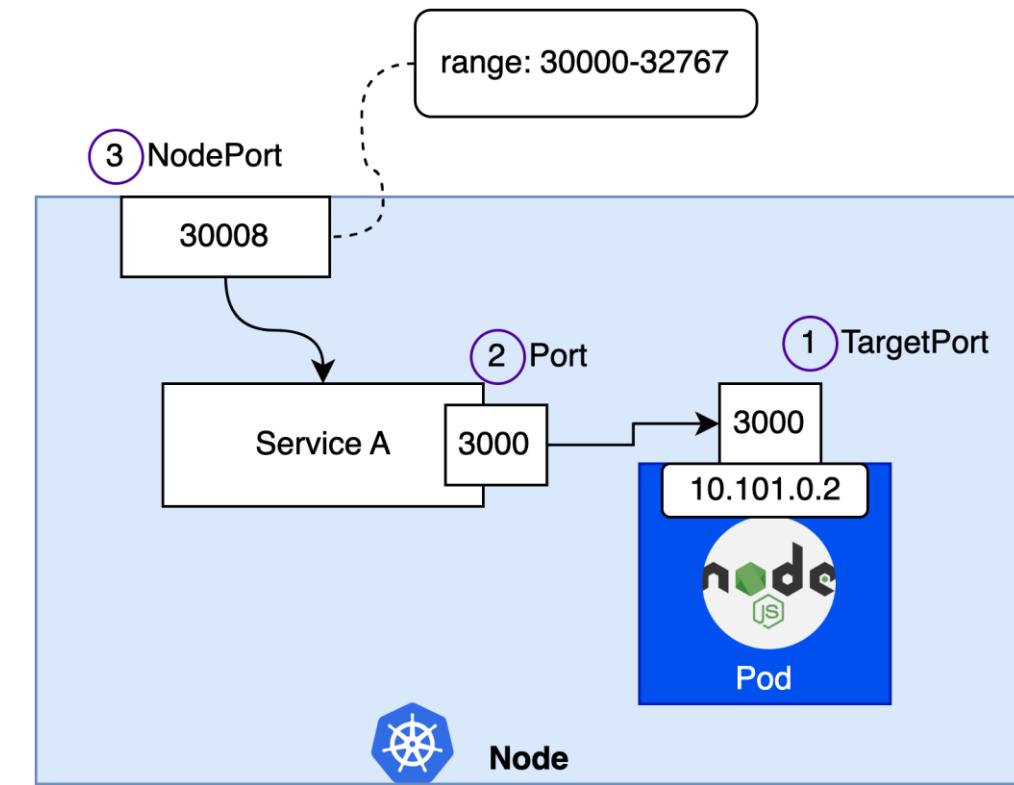
Service

- Service trong Kubernetes là một khái niệm trừu tượng mô tả một tập hợp các pod hoạt động cùng nhau như một dịch vụ. Các pod được chọn dựa trên các tiêu chí mà service định nghĩa.
- Service cho phép tương tác với các pod mà không cần biết chi tiết về cấu trúc mạng hoặc vị trí của chúng. Điều này rất hữu ích khi chúng ta muốn cung cấp High Availability và LoadBalance cho các ứng dụng.
- Có 3 dạng service của Kubernetes là **NodePort**, **ClusterIP** và **LoadBalancer**

Copyright@Linh Nguyen on Udemy
All right reserved

Service - NodePort

- **NodePort:** Đây là loại Service cơ bản nhất. Khi bạn tạo một Service với kiểu NodePort, Kubernetes sẽ cấp một cổng (port) cố định trên mỗi node trong cluster cho Service đó. Bất kỳ ai có thể kết nối đến cổng này trên bất kỳ node nào sẽ được chuyển hướng đến Service.
- NodePort: port trên node có range từ 30000-32676. Nếu không chỉ định sẽ dc set random.
- Port: port của service.
- Target Port: Port mà pod expose ra để truy cập. Nếu không chỉ định sẽ hiểu = Port.



Copyright@Linh Nguyen on Udemy
All right reserved

Service

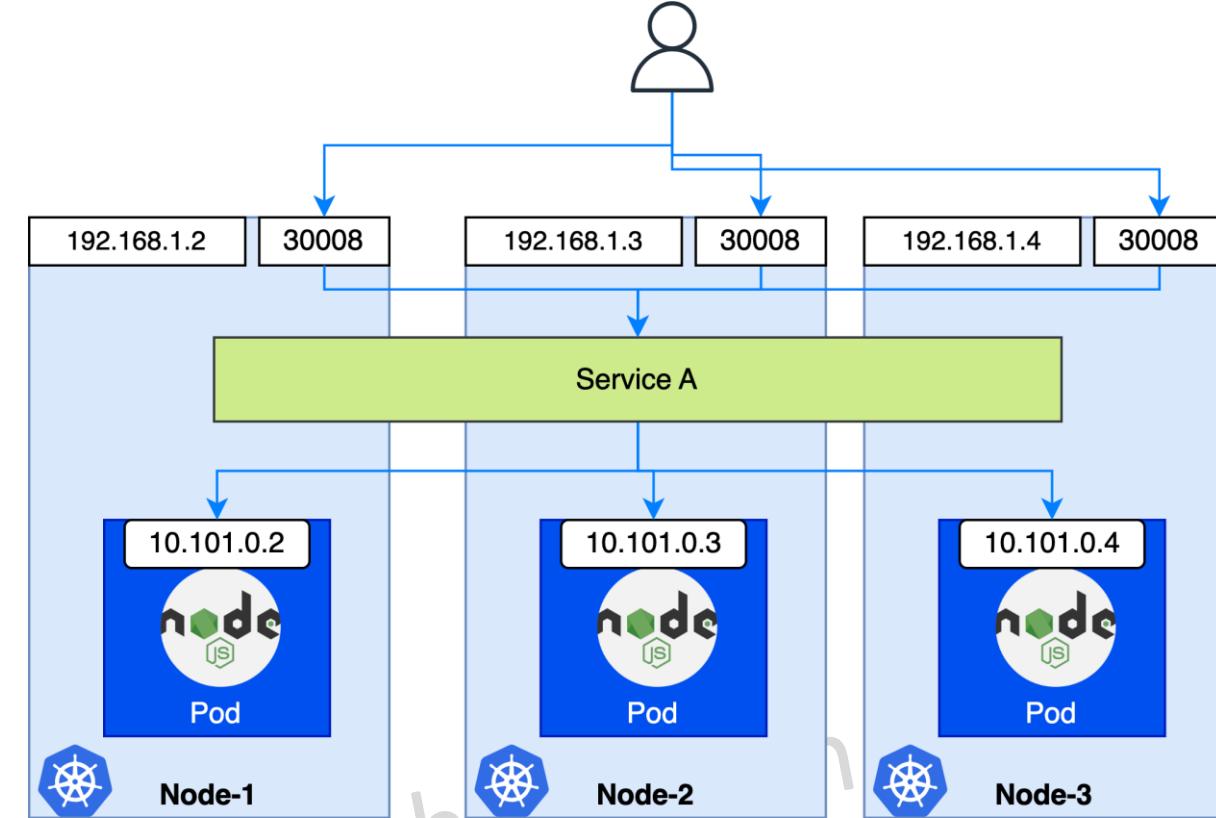
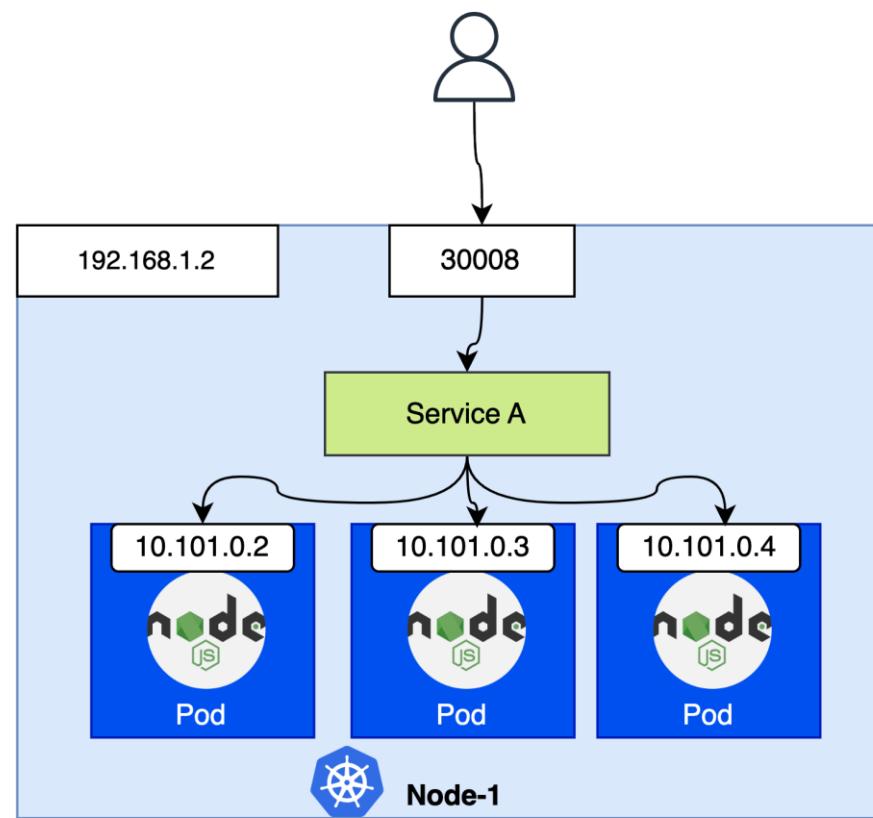
- Service điều hướng traffic tới pod sử dụng selector.

```
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: my-service
5  spec:
6    selector:
7      app: MyApp
8    ports:
9      - nodePort: 30001
10     port: 3000
11     targetPort: 3000
```

Copyright@Linh Nguyen on Udemy
All right reserved

Service NodePort trong single & multi Node cluster.

Trong multi node cluster, NodePort service sẽ được triển khai span trên các node, giúp có thể truy cập sử dụng IP của bất kỳ node nào.



Có một vấn đề: Khi các IP của node bị thay đổi (vd node chết và tạo lại), cần phải handle việc đó (tất nhiên sẽ có tool)

Lab 6 – Tạo một NodePort Service

Yêu cầu:

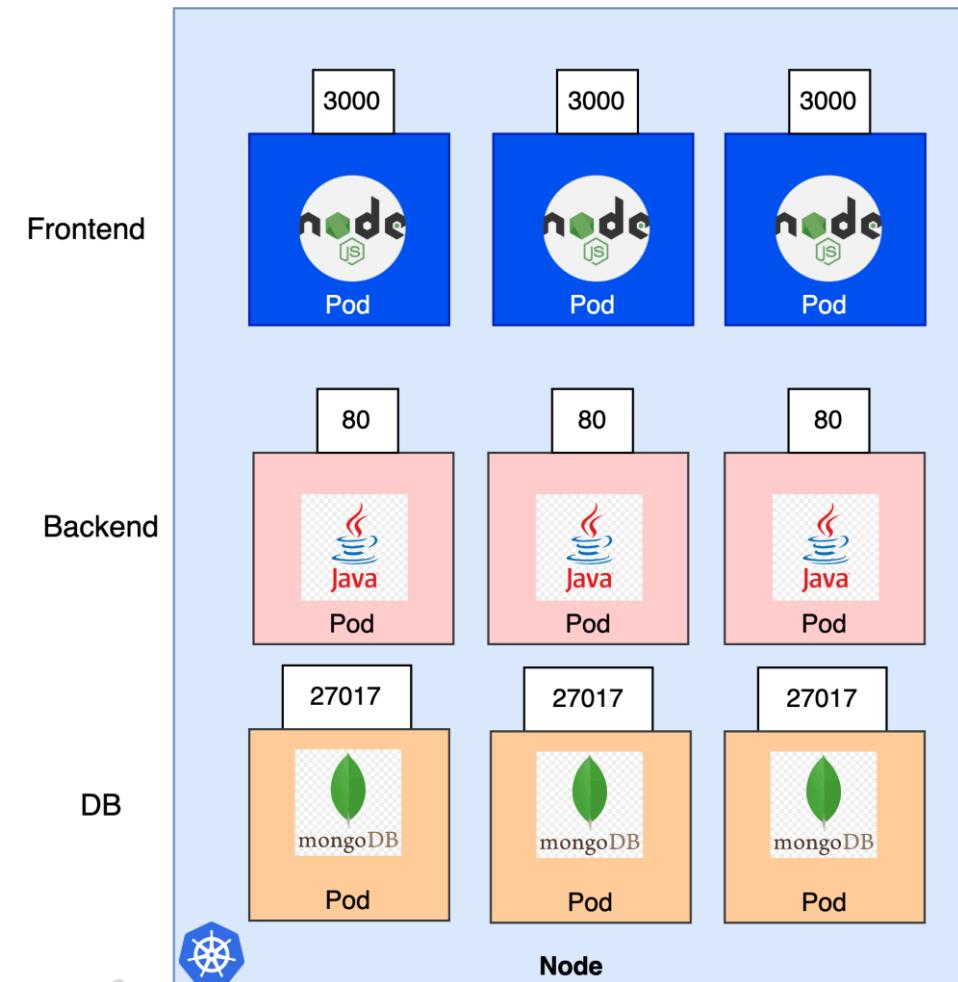
- Tạo một file cấu hình cho nginx service dạng NodePort (kèm theo deployment).
- Apply file cấu hình cho Kubernetes cluster.
- Kiểm tra Service & Deployment được tạo ra.
- Sử dụng câu lệnh sau để check IP của minikube, open địa chỉ <minikube_ip>:<node_port> trên trình duyệt để xem kết quả.

```
>minikube ip
```

Copyright@Linh Nguyen on Udemy
All right reserved

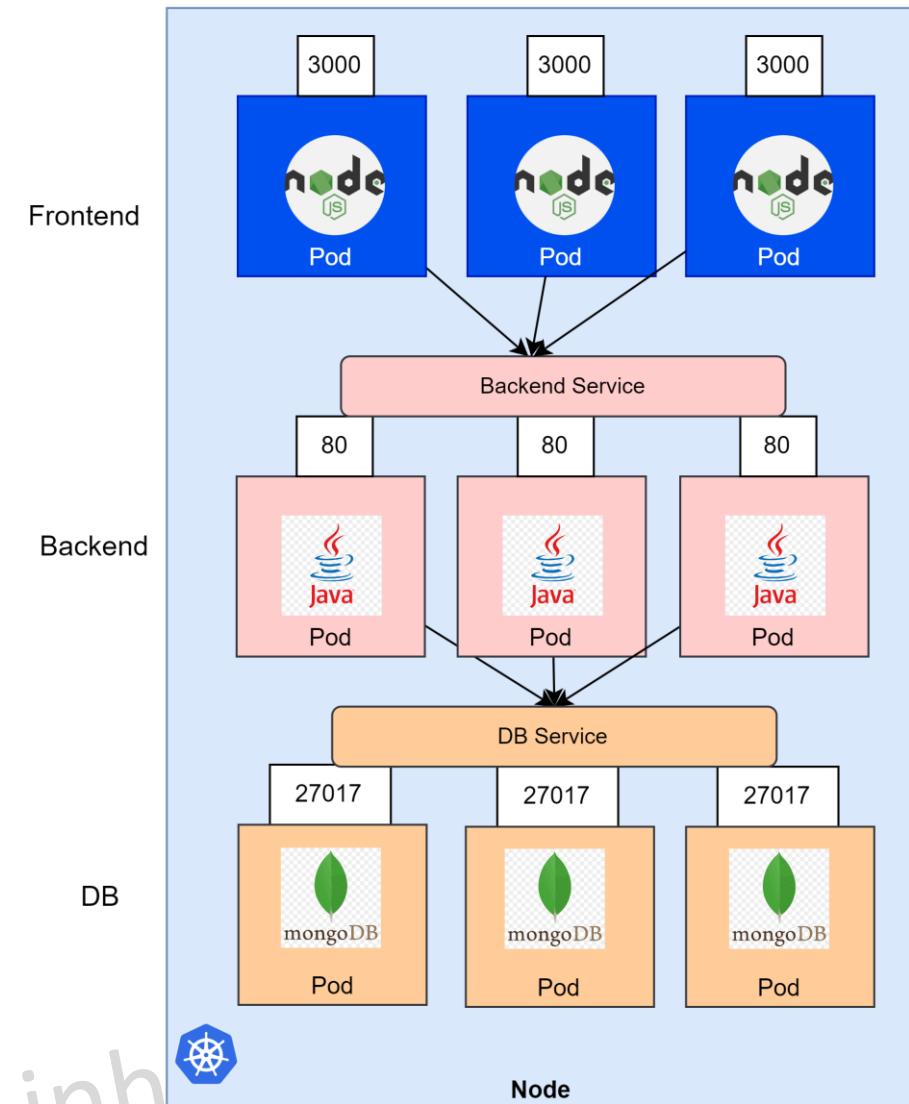
Service.- ClusterIP

- Cluster IP:** Khi bạn tạo một Service với kiểu ClusterIP, Kubernetes sẽ cấp một địa chỉ IP nội bộ (chỉ có thể truy cập từ bên trong cluster) cho Service đó. Điều này hữu ích cho việc cho phép các pod trong cùng một cluster giao tiếp với nhau.
- Vd hình bên, Frontend cần 1 connection point để giao tiếp với Backend, Backend cần 1 connection point để giao tiếp với DB.



Service.- ClusterIP

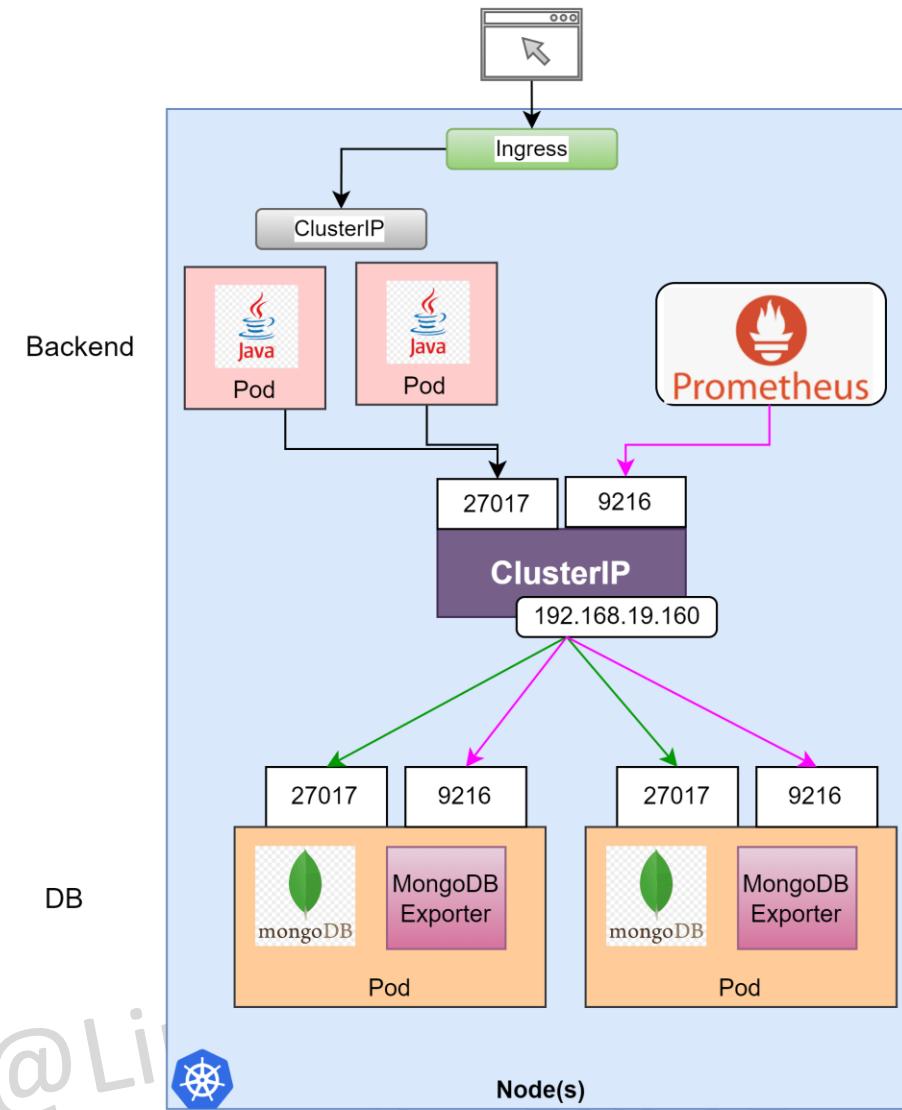
- **Cluster IP** service chỉ có thể được truy cập từ bên trong cluster.
- Các Pod bên trong cluster có thể truy cập tới Cluster IP service sử dụng IP hoặc service-name.



[Tham khảo thêm] Multi container in a Pod

Service có thể đứng trước một nhóm các pod mà trong đó mỗi pod có nhiều hơn 1 container.

Vd container Mongo có port 27017 và 1 container sidecar có port 9126.



[Tham khảo thêm] Multi container in a Pod

Trong trường hợp multi container, port cần được định nghĩa name.

```
apiVersion: v1
kind: Service
metadata:
  name: mongodb-service
  ...
spec:
  selector:
    app: mongodb
  ports:
    - name: mongodb
      protocol: TCP
      port: 27017
      targetPort: 27017
    - name: mongodb-exporter
      protocol: TCP
      port: 9216
      targetPort: 9216
```

Copyright@Linh Nguyen All right reserved

Lab 7 – Tạo một ClusterIP Service

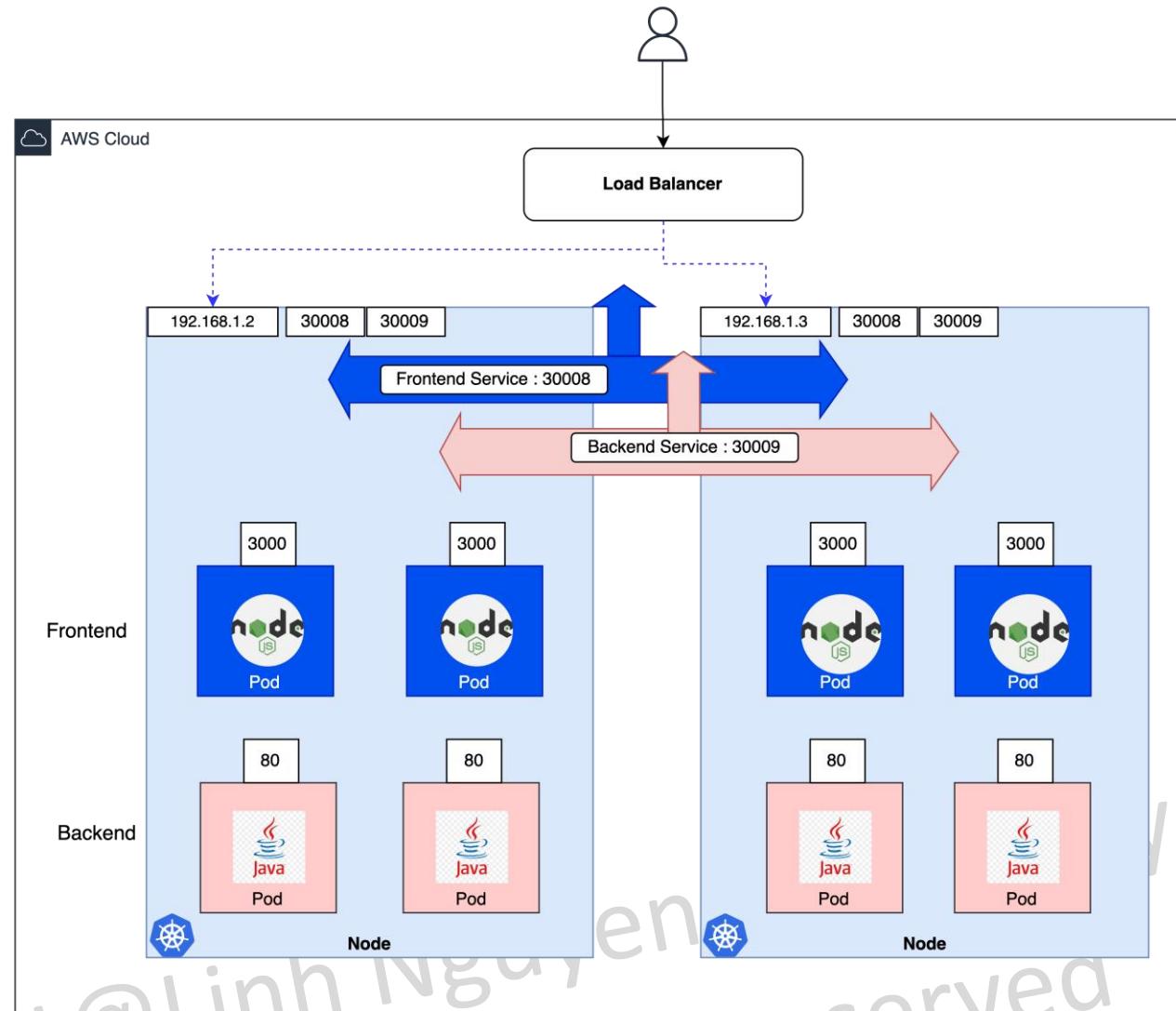
Yêu cầu tạo một ClusterIP Service cho MongoDB.

**Lưu ý bài lab chỉ mang tính chất demo vì trong thực tế các ứng dụng dạng Database như Mongo đòi hỏi cơ chế quản lý state phức tạp (StatefulSet).*

Copyright@Linh Nguyen on Udemy
All right reserved

Service - Load Balancer

- LoadBlalancer:** Đây là loại Service phức tạp nhất. Khi bạn tạo một Service với kiểu LoadBalancer, Kubernetes sẽ cấp một địa chỉ IP công khai cho Service đó và cấu hình một **LoadBalancer** trên Cloud Provider của bạn (nếu hỗ trợ) để điều hướng traffic đến Service. Điều này hữu ích cho việc expose Service ra bên ngoài Internet.



Service - Load Balancer

- Thông thường Kubernetes cluster sẽ chịu trách nhiệm config Load Balancer (vd AWS ALB) để có thể điều hướng tới các target group (đại diện cho các nhóm service khác nhau trên Cluster) nhằm điều hướng traffic từ client đến đúng các pod. Cần cấp IAM Role để Cluster có thể thao tác.

Copyright@Linh Nguyen on Udemy
All right reserved

Lab – Tạo một LoadBalancer

*Lab về Load Balancer sẽ được minh trình bày trong chương AWS EKS.

Copyright@Linh Nguyen on Udemy
All right reserved

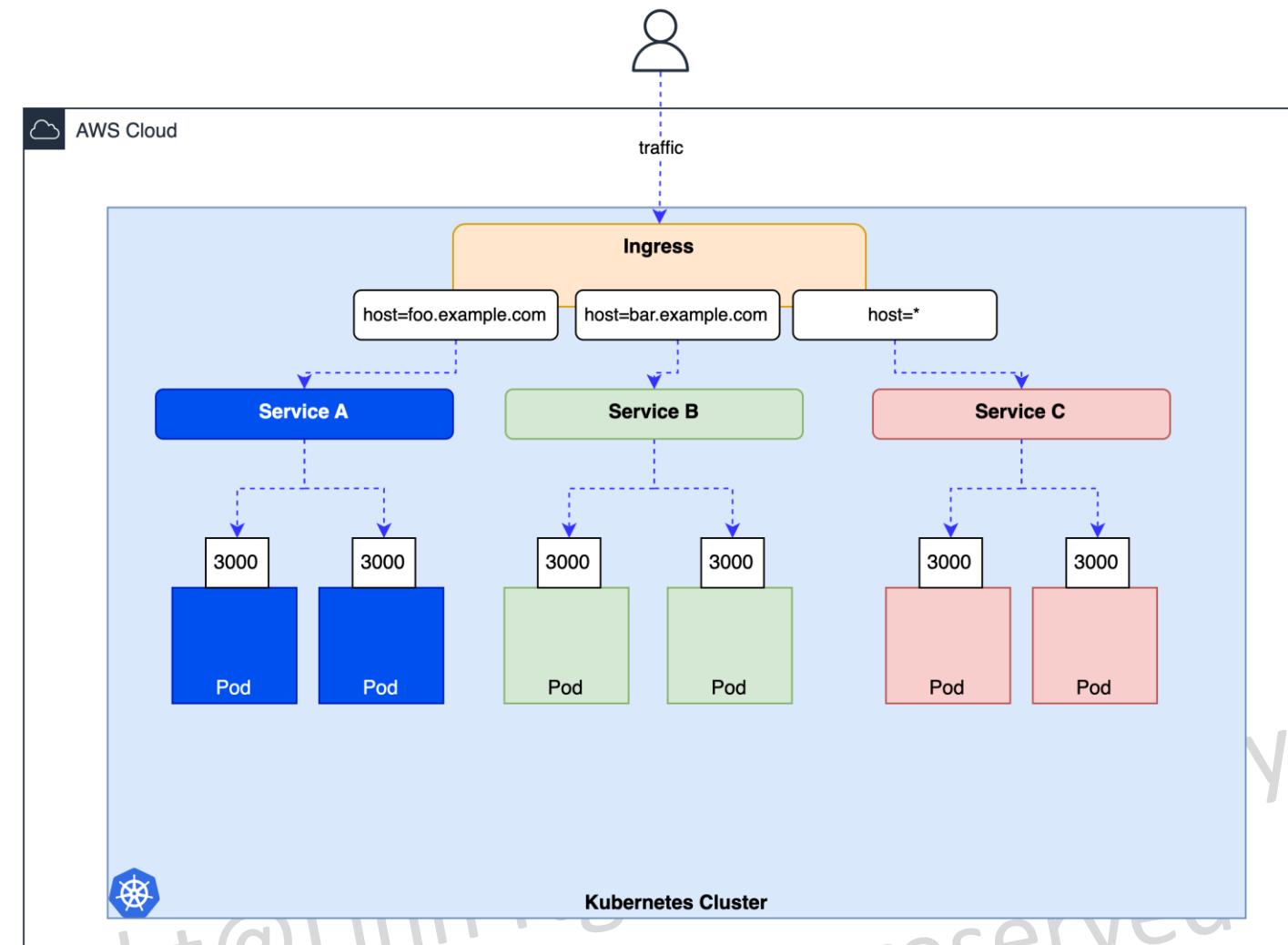
Ingress

Ingress trong Kubernetes là một API object cung cấp quy tắc định tuyến HTTP và HTTPS từ bên ngoài cluster đến các dịch vụ bên trong cluster.

Ingress cho phép bạn xác định:

- Hostname: quy tắc định tuyến dựa trên tên host.
- Path: quy tắc định tuyến dựa trên đường dẫn URL.
- Service: dịch vụ cần định tuyến đến.
- LoadBalancer: cân bằng tải network traffic đến các pod của dịch vụ.

Ingress cần một Ingress Controller để hoạt động, ví dụ như nginx, traefik, hoặc istio.



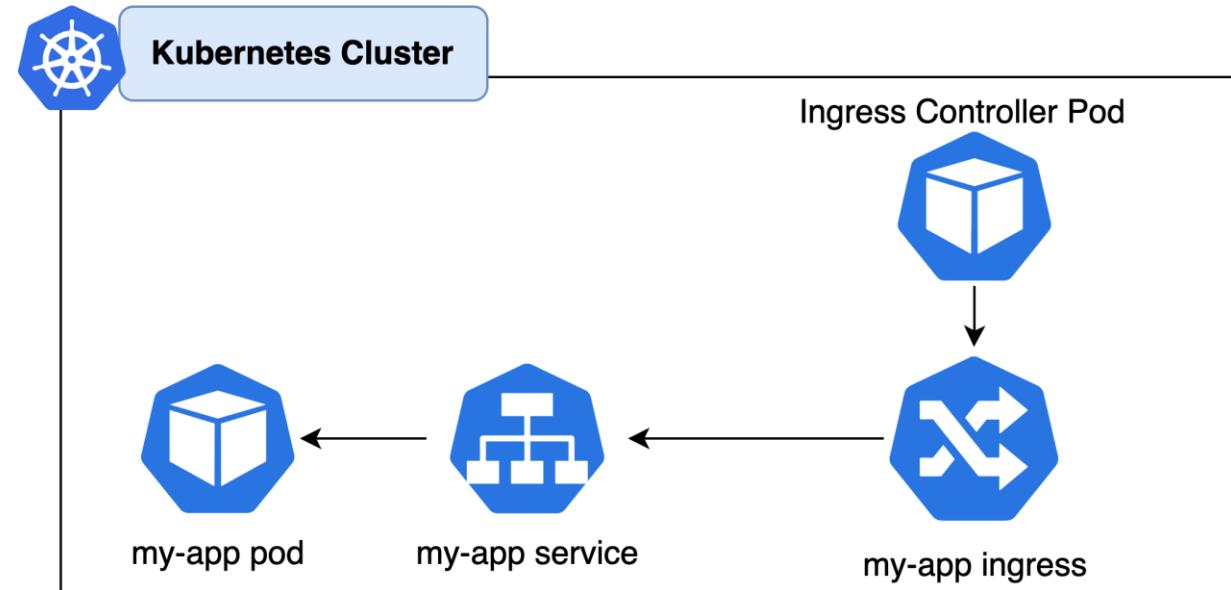
Ingress

Ingress cần một Ingress Controller để hoạt động.

Ingress Controller có nhiệm vụ đánh giá toàn bộ rule trong cluster, quản lý redirection, đóng vai trò như entrypoint của cluster.

Một số loại Ingress Controller:

- K8s Nginx Ingress Controller.
- Traefik
- HAProxy
- Kong
- Voyager
- Istio
- [AWS ALB Ingress Controller](#)
- [GCP GLBC/GCE-Ingress Controller](#)
- [AKS Application Gateway Ingress Controller](#).



So sánh: Load Balancer vs Ingress

- LoadBalancer: Khi bạn tạo một Service với kiểu LoadBalancer, Kubernetes sẽ cấp một địa chỉ IP công khai cho Service đó và cấu hình một load balancer trên cloud provider của bạn để điều hướng network traffic đến Service. Mỗi Service LoadBalancer sẽ có một địa chỉ IP riêng và một cân bằng tải riêng. Điều này có thể dẫn đến việc sử dụng tài nguyên không hiệu quả nếu bạn có nhiều Service cần expose ra bên ngoài.
- Ingress: Ingress là một API object trong Kubernetes cho phép bạn xác định các quy tắc định tuyến HTTP và HTTPS từ bên ngoài cluster đến các dịch vụ bên trong cluster. Ingress cho phép bạn quản lý truy cập vào các dịch vụ dựa trên URL path và host name, và nó sử dụng một Ingress Controller (vd nginx, traefik, hoặc istio) để thực hiện các quy tắc định tuyến. Ingress chỉ cần một địa chỉ IP công khai và một load balancer, không phụ thuộc vào số lượng dịch vụ, giúp tiết kiệm tài nguyên.

Nói cách khác, LoadBalancer thích hợp cho việc expose một số lượng nhỏ dịch vụ ra bên ngoài, trong khi Ingress thích hợp cho việc quản lý truy cập vào nhiều dịch vụ dựa trên URL path và host name.

Copyright@Linh Nguyen on Udemy
All right reserved

ALB Ingress controller trên AWS

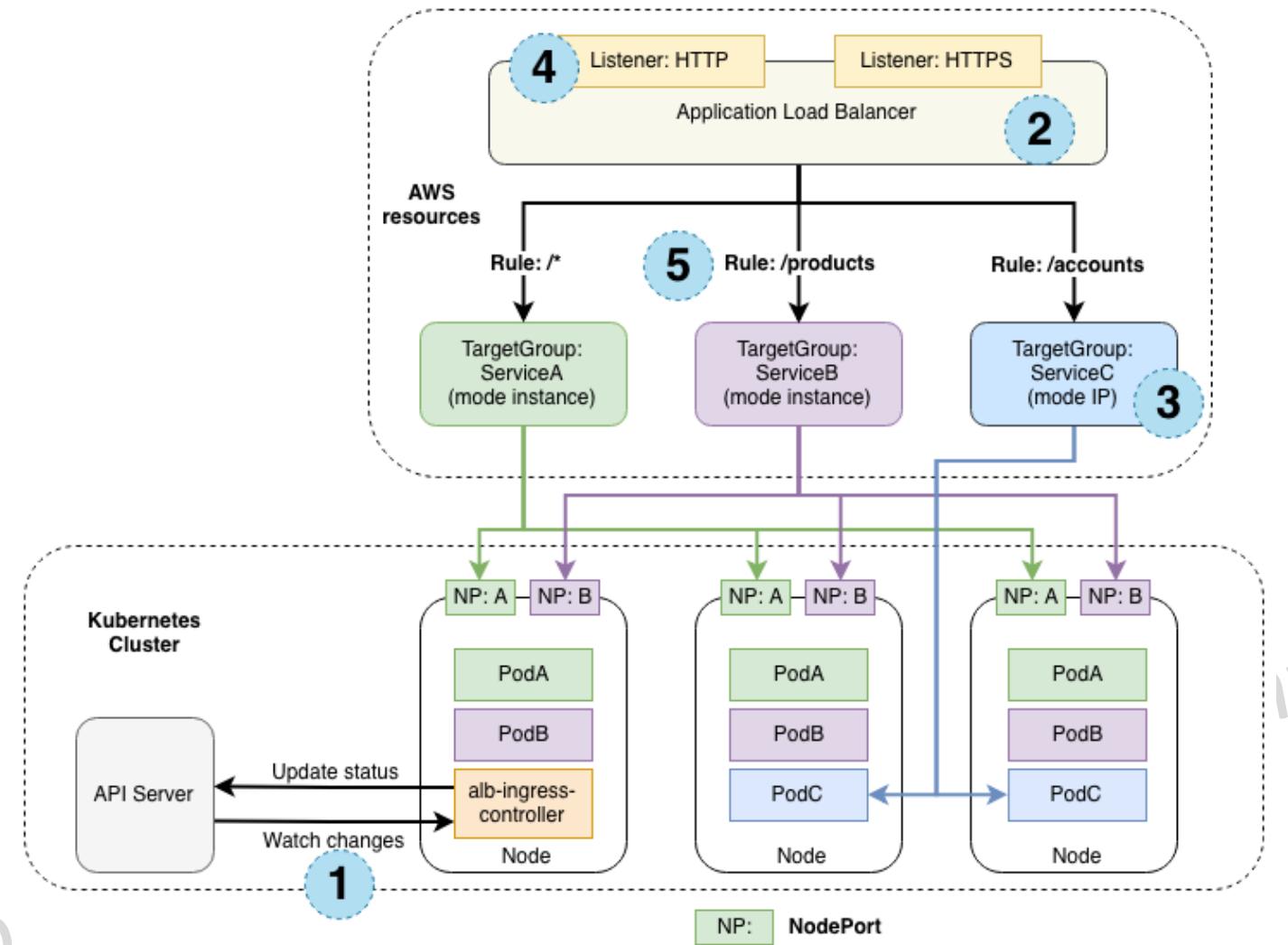
Tham khảo thêm: Mô hình của Ingress sử dụng ALB trên AWS.

1. Ingress controller listen các lệnh thay đổi từ API Server.
2. ALB được tạo ra trên AWS. ALB có thể là Internet facing hoặc Internal.
3. Target group được tạo ra cho mỗi service.
4. Listener được tạo ra trên ALB và mapping với 1 port. (vd 80, 443).
5. Rule được tạo ra trên ALB vd dựa trên path.

TargetGroup hỗ trợ 2 loại là:

Instance Mode (load balance trên các Node IP), trong mode này service phải được tạo ra theo kiểu NodePort.

IP Mode: redirect trực tiếp traffic xuống Pod. Cần cho phép gán secondary IP vào mỗi pod (Elastic Network Interface).



Lab 8 – Tạo một Ingress cho 2 service dựa trên request host (sub-domain)

- Chuẩn bị 2 Docker image dạng Nodejs, đơn giản print ra message Service-A, Service-B, push image lên DockerHub.
- Tạo một File cấu hình bao gồm các thông số:
 - Service A & Deployment_A
 - Service B & Deployment_B
 - File cấu hình ingress routing theo host: **service-a.localhost**, **service-b.localhost**
- Apply cấu hình cho Kubernetes cluster.
- Truy cập vào cluster thông qua ingress với sub domain:
 - **service-a.localhost**
 - **service-b.localhost**

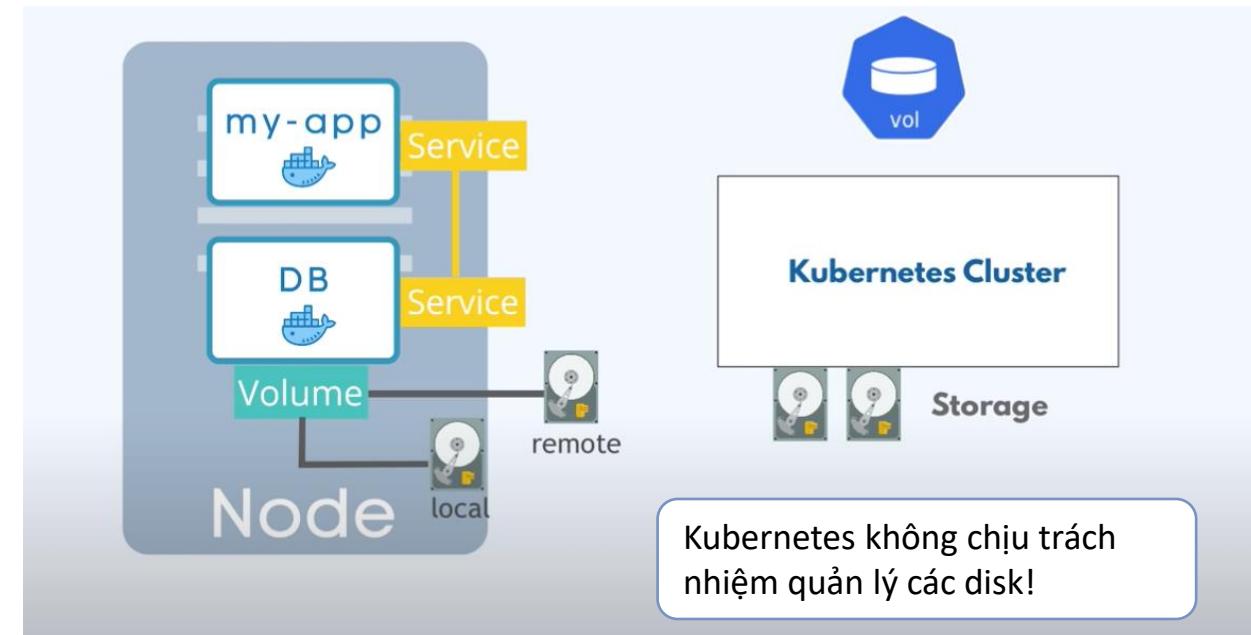
Copyright@Linh Nguyen on Udemy
All right reserved

Persistent Volume in Kubernetes

Đối với các ứng dụng cần có khu vực lưu trữ lâu dài (persistent) vd DB hoặc một service cần đọc/ghi file, ta cần cung cấp Persistent Volume cho chúng.

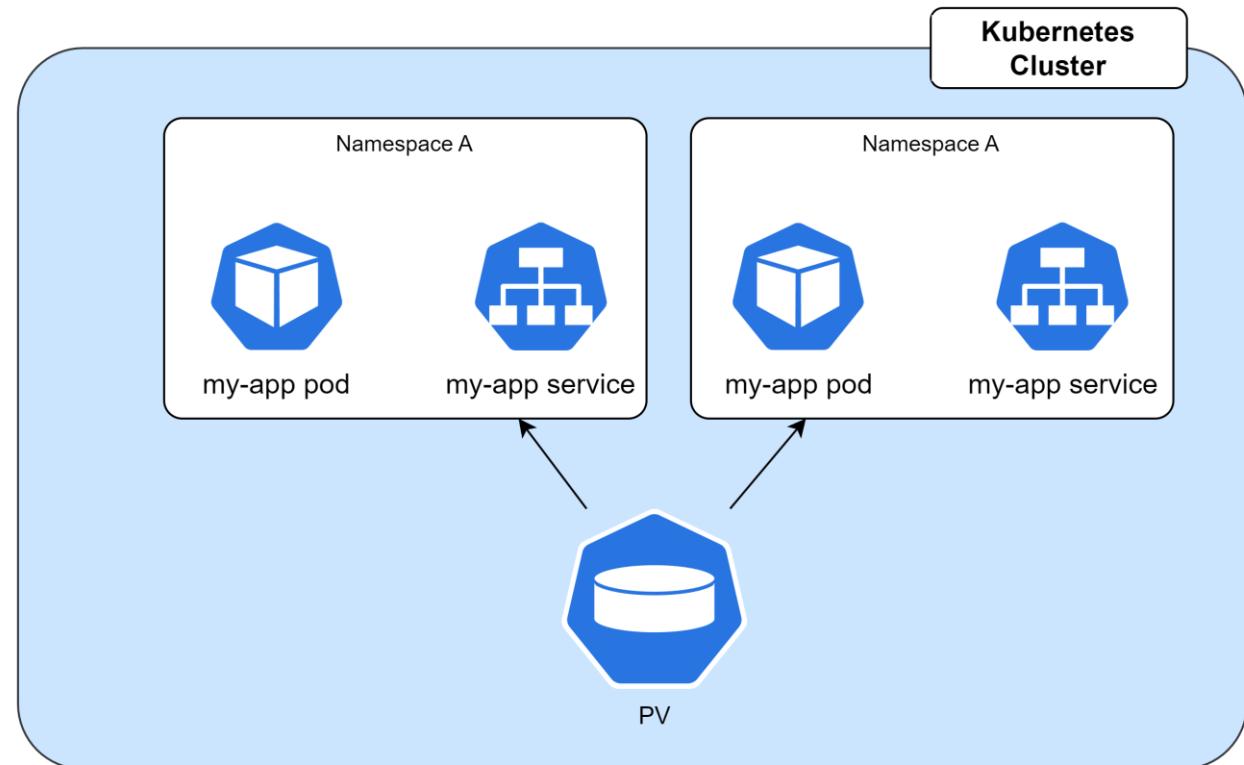
Persistent Volume có thể nằm trên ổ cứng của Node, nfs hoặc Cloud Service (EBS).

Lưu ý: Kubernetes không chịu trách nhiệm cho việc quản lý các disk.



Persistent Volume in Kubernetes

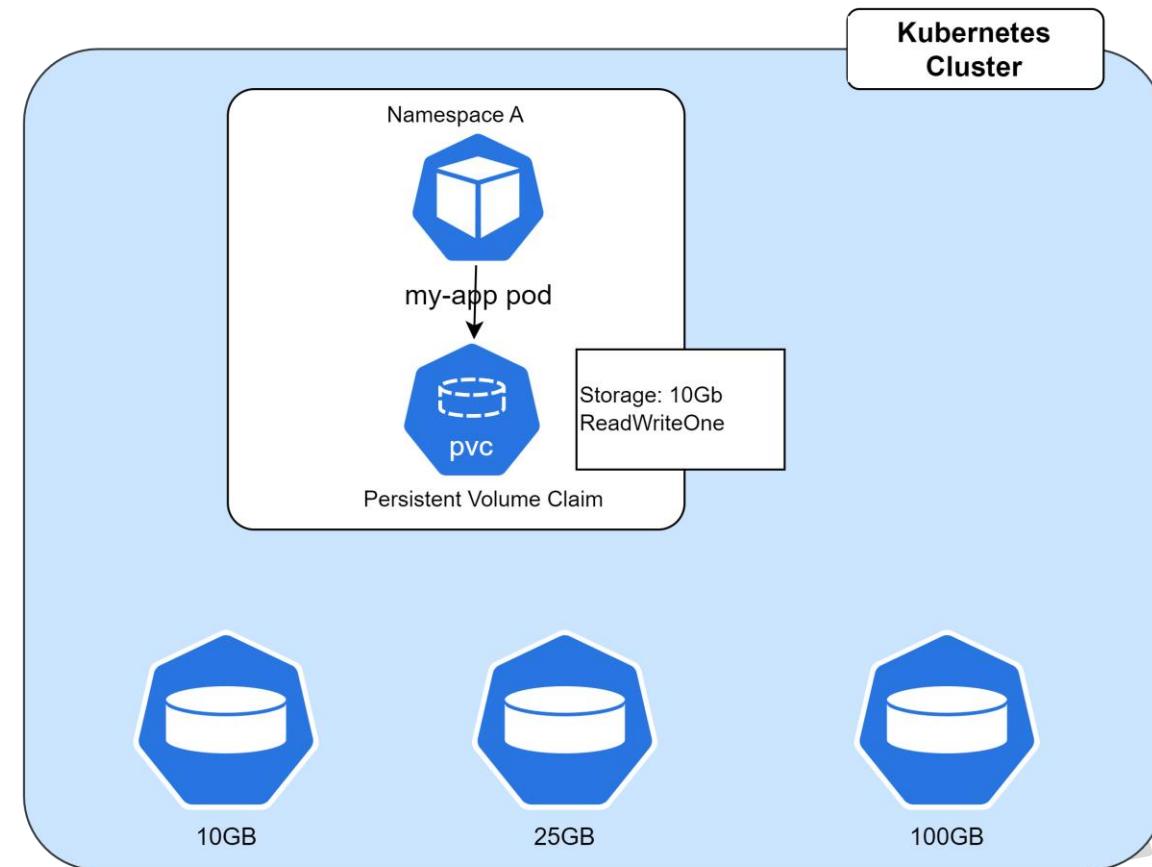
Persistent Volume là một resource nằm ngoài namespace.



Copyright@Linh Nguyen on Udemy
All right reserved

Persistent Volume in Kubernetes

Pod cần tạo một Persistent Volume Claim (pvc) để định nghĩa mapping giữa pod và volume thực tế.



Persistent Volume type in Kubernetes

Local volume

- Gắn trực tiếp vào 1 Node trong Cluster.
- Bị mất data nếu node đó crash.

Remote volume

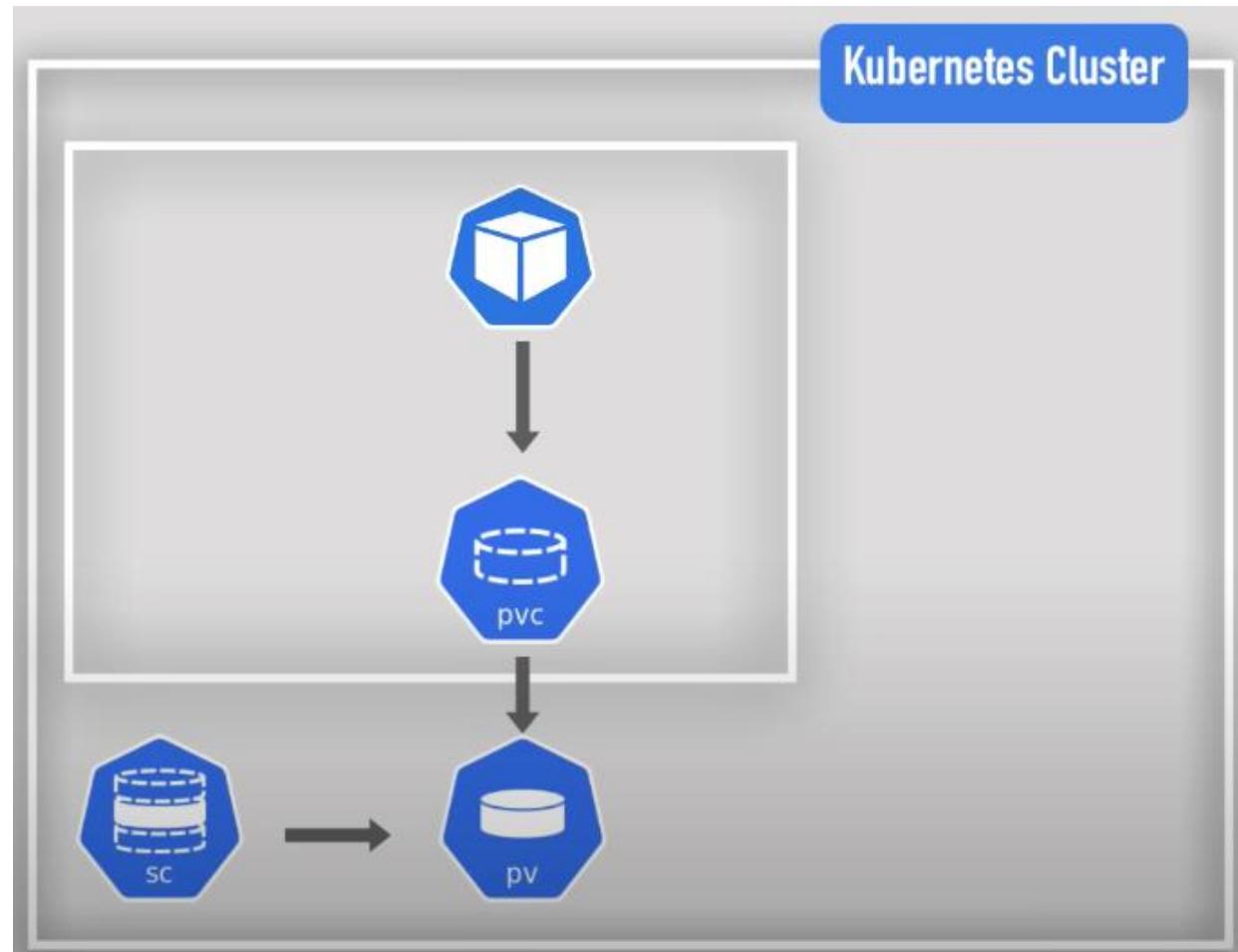
- Không nằm trên Node của Cluster

Copyright@Linh Nguyen on Udemy
All right reserved

Storage Class

Cung cấp cơ chế tạo & quản lý volume tự động (vd tự tạo và cấp phát EBS Disk nếu cần).

- Pod request volume thông qua pvc.
- PVC request volume thông qua Storage Class.
- Storage Class tạo ra các volume thoả mãn yêu cầu.



Copyright@Linh Nguyen
All right reserved

Lab 9 – Tạo một Persistent Volume, gán vào Pod.

Yêu cầu:

- Khai báo một Persistent volume & một persistent volume claim
- Khai báo một Service (NodePort) và 1 Deployment sử dụng MySQL, có mount volume tới persistent volume claim (pvc)
- Triển khai resource. Kết nối tới MySQL và insert một vài data.
- Thủ delete pod.
- Kiểm tra Pod được khôi phục.
- Truy cập lại MySQL kiểm tra xem data còn tồn tại?

Copyright@Linh Nguyen on Udemy
All right reserved

Configmap

ConfigMap trong Kubernetes là một đối tượng API được sử dụng để lưu trữ các configuration không nhạy cảm dưới dạng **key-value pairs**. ConfigMap cho phép bạn tách cấu hình từ các container và module của ứng dụng, giúp ứng dụng của bạn trở nên dễ dàng cấu hình và di chuyển.

Các ConfigMap có thể được sử dụng trong các pod thông qua các biến môi trường, thông qua args command-line, hoặc thông qua các file config trong một volume.

Copyright@Linh Nguyen on Udemy
All right reserved

Configmap

Ví dụ về Configmap

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: game-demo
data:
  # property-like keys; each key maps to a simple value
  player_initial_lives: "3"
  ui_properties_file_name: "user-interface.properties"
  # file-like keys
  game.properties: |
    enemy.types=aliens,monsters
    player.maximum-lives=5
```

Copyright@Linh Nguyen on Udemy
All right reserved

Configmap

Passing Configmap value vào trong Pod

```
env:
- name: MY_ENV_VAR
  valueFrom:
    configMapKeyRef:
      name: my-configmap
      key: my-key
```

Sử dụng như biến môi trường

```
args:
- --arg-from-config=$(MY_ENV_VAR)
env:
- name: MY_ENV_VAR
  valueFrom:
    configMapKeyRef:
      name: my-configmap
      key: my-key
```

Sử dụng như command line arguments

```
volumes:
- name: config-volume
  configMap:
    name: my-configmap
containers:
- name: my-container
  volumeMounts:
- name: config-volume
  mountPath: /etc/config
```

Mount thành 1 file trong volume. Trong trường hợp này, mỗi cặp key:value sẽ thành 1 file trong volume.

Lab 10 – Tạo một ConfigMap

- Tạo một configmap lưu một số key: value đơn giản.
- Tạo một Service MySQL (kiểu NodePort) và passing configmap vào Env.
- Apply cho Kubernetes cluster.
- Thủ truy cập vào MySQL bằng thông tin trong ConfigMap.

Copyright@Linh Nguyen on Udemy
All right reserved

Secret

Secret trong Kubernetes là một đối tượng API được sử dụng để lưu trữ thông tin nhạy cảm như mật khẩu, token, private key,... Điều này giúp bảo vệ thông tin quan trọng và ngăn chặn rò rỉ thông tin nhạy cảm.

Secrets được sử dụng bởi các Pod để truy cập thông tin nhạy cảm mà chúng cần để hoạt động. Chúng có thể được sử dụng trong các Pod thông qua các biến môi trường, thông qua args command-line, hoặc thông qua các file config trong một volume.

Copyright@Linh Nguyen on Udemy
All right reserved

Secret

Ví dụ về Secret

*Lưu ý trong Kubernetes, giá trị của secret phải được mã hoá base64 tại thời điểm khai báo.

```
apiVersion: v1
kind: Secret
metadata:
  name: mysecret
  type: Opaque
data:
  username: YWRtaW4= # Base64 encoded value of 'admin'
  password: MWYyZDFlMmU2N2Rm # Base64 encoded value of '1f2d1e2e67df'
```

Copyright@Linh Nguyen on Academy
All right reserved

Secret

Tương tự như Configmap, Secret cũng có thể được pass vào Pod thông qua 2 cách: biến môi trường và volume mount.

```
apiVersion: v1
kind: Pod
metadata:
  name: secret-env-pod
spec:
  containers:
    - name: mycontainer
      image: redis
      env:
        - name: SECRET_USERNAME
          valueFrom:
            secretKeyRef:
              name: mysecret
              key: username
        - name: SECRET_PASSWORD
          valueFrom:
            secretKeyRef:
              name: mysecret
              key: password
  restartPolicy: Never
```

```
apiVersion: v1
kind: Pod
metadata:
  name: secret-volume-pod
spec:
  containers:
    - name: mycontainer
      image: redis
      volumeMounts:
        - name: secret-volume
          mountPath: "/etc/secret"
          readOnly: true
  volumes:
    - name: secret-volume
      secret:
        secretName: mysecret
  restartPolicy: Never
```

Lab 11 – Tạo một secret

Tạo một secret username/password

Tạo một Service MySQL (kiểu NodePort) và passing secret vào Env.

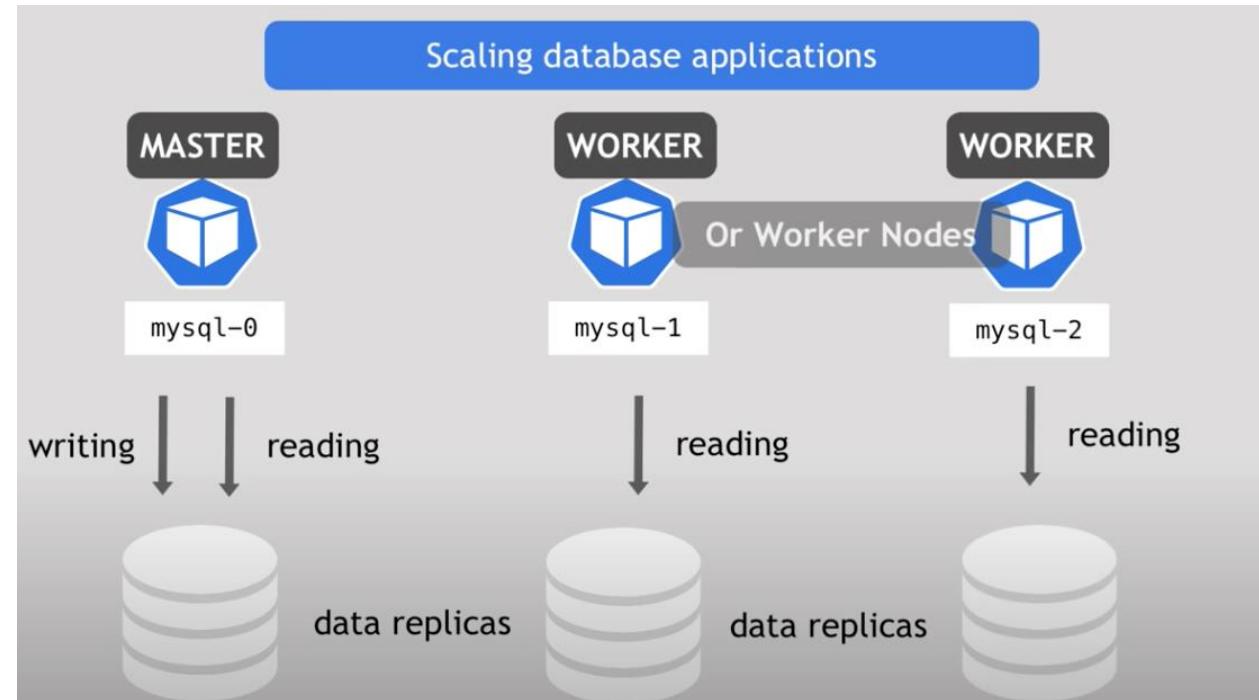
Apply cho Kubernetes cluster.

Thử truy cập vào MySQL bằng thông tin trong secret.

Copyright@Linh Nguyen on Udemy
All right reserved

Stateful Set

Đối với các ứng dụng có data và cần quản lý chặt chẽ quan hệ của các Pod, ta cần một cơ chế khác với các ứng dụng thông thường.

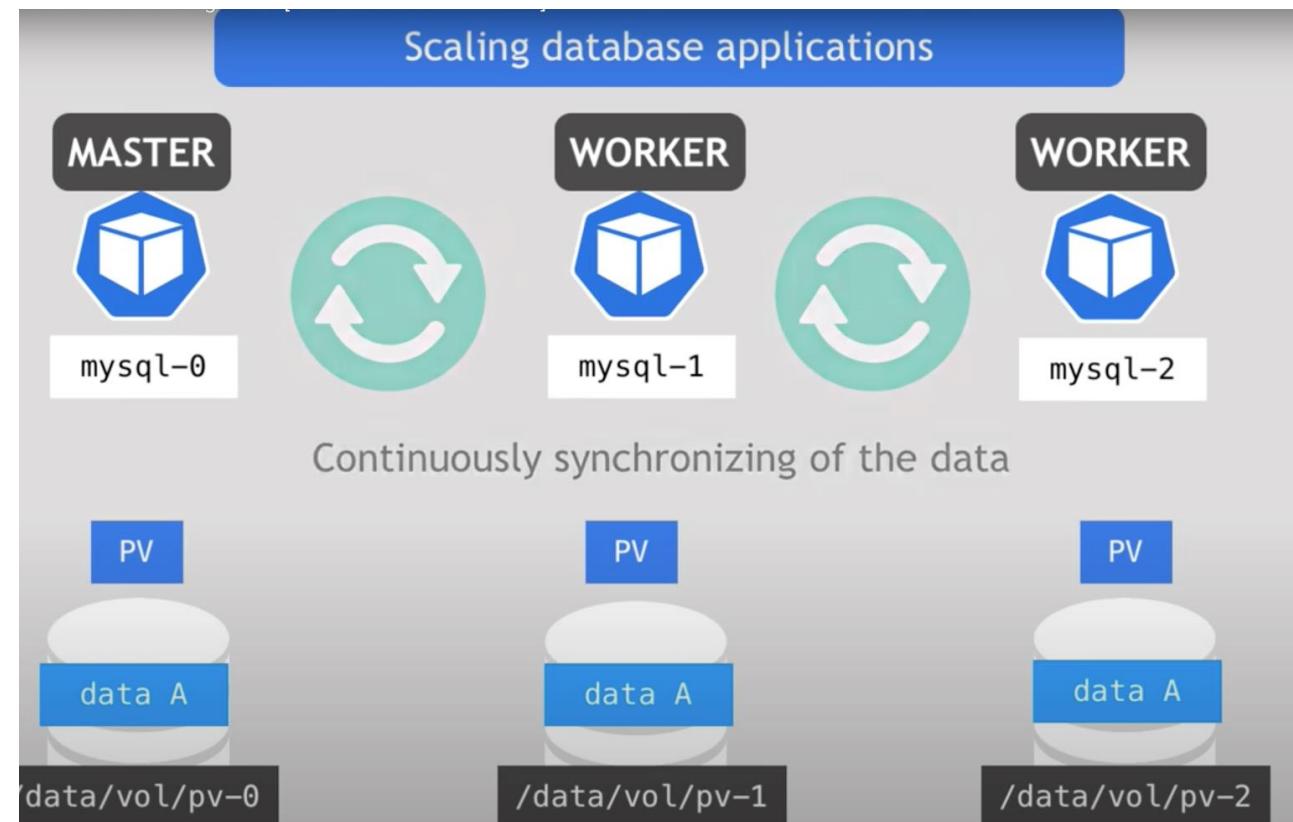


Xét một ví dụ về Database MySQL, trong service sẽ có pod đóng vai trò Master và pod đóng vai trò Worker (read-replica).

Stateful Set

VD hình bên, data cần được sync giữa các pod, chỉ có pod master được ghi data.

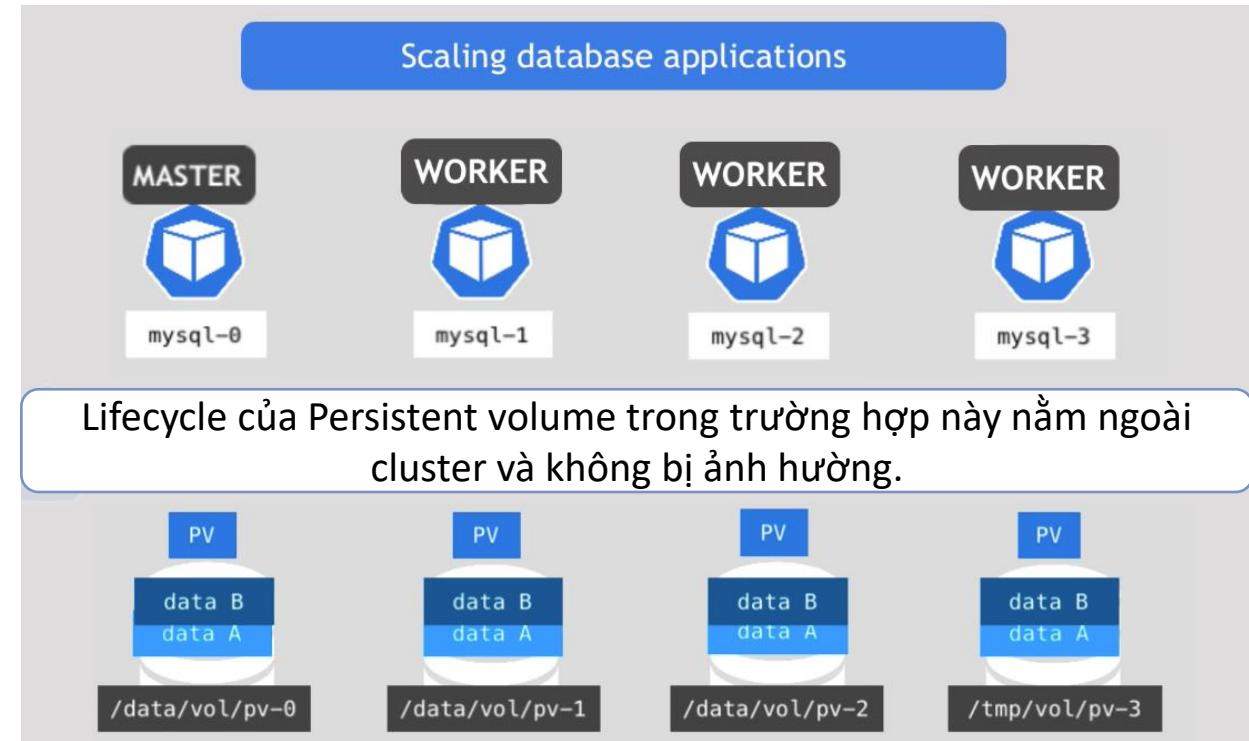
Việc replicate data này vẫn có 1 rủi ro khi toàn bộ nodes die cùng lúc => mất data.



Xét một ví dụ về Database MySQL, trong service sẽ có pod đóng vai trò Master và pod đóng vai trò Worker (read-replica).

Stateful Set

Sử dụng volume persistent nằm ngoài cluster đảm bảo data sẽ không bị mất khi các node có sự cố.



Mặc dù Kubernetes cung cấp cơ chế để quản lý các statefull set (dành cho ứng dụng statefull) tuy nhiên việc setup vẫn tốn nhiều công sức và khó có một giải pháp triệt để.
=>Nên sử dụng các dịch vụ Cloud cho DB, Cache, ElasticSearch,...

Giới thiệu về KOPS

Kops, viết tắt của Kubernetes Operations, là một công cụ giúp bạn tạo, xóa, nâng cấp và duy trì cụm Kubernetes trên các nền tảng điện toán đám mây như AWS, GCE và DigitalOcean.

Kops giúp bạn quản lý cụm Kubernetes của mình một cách dễ dàng và hiệu quả.

Copyright@Linh Nguyen on Udemy
All right reserved

Giới thiệu về KOPS

Một số tính năng nổi bật:

- Automates the provisioning of Highly Available Kubernetes clusters
- Built on a state-sync model for **dry-runs** and automatic **idempotency**
- Ability to generate Terraform

...

***automatic idempotency**: This means that you can run it multiple times and the end state of the cluster will be the same. Kops will ensure that the necessary resources are created or updated to match your cluster specification, but if they already match, Kops won't make any changes. This is what we mean by automatic idempotency.

Lab 12 – Sử dụng KOPS tạo một K8S Cluster và kết nối

Yêu cầu:

- Cài đặt Kubectl, AWS CLI, Kops lên một EC2 Ubuntu
- Cấp quyền Admin cho EC2.
- Tạo một Domain sử dụng Route53
- Tạo một cluster Kubernetes với 1 node master, 3 node worker.
- Kết nối với Cluster từ EC2 instance.
- Download cấu hình và kết nối tới Cluster từ máy cá nhân.

Copyright@Linh Nguyen on Udemy
All right reserved

Lab 13 – Triển khai một ứng dụng mẫu lên Kubernetes cluster

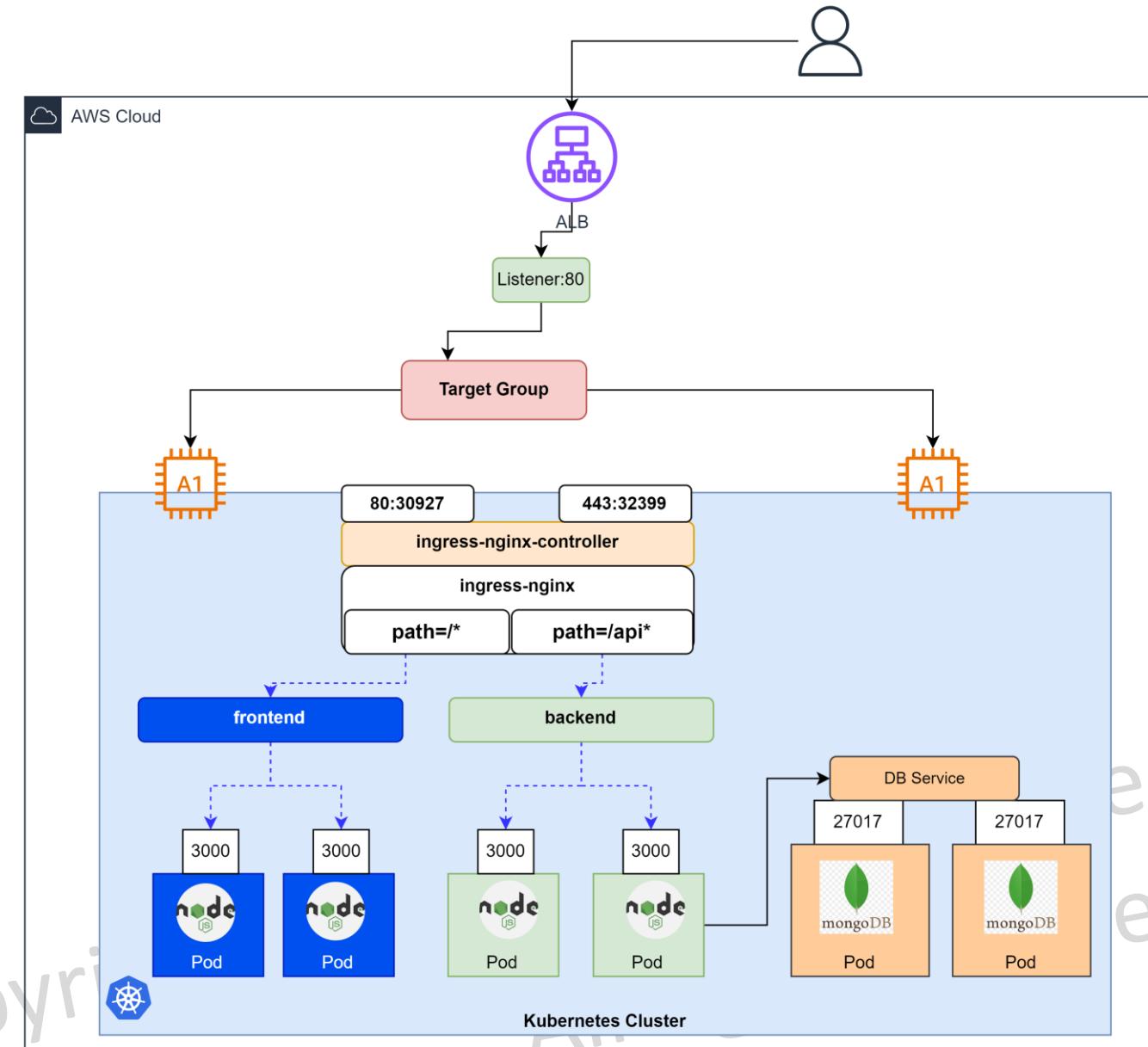
Các bước thực hiện:

- Chuẩn bị một Kubernetes Cluster (tham khảo bài Lab 12 sd Kops).
- Tìm một ứng dụng mẫu trên Github (keyword “awsome compose github”)
- Chạy thử tại máy local bằng Docker Compose để confirm hoạt động.
- Phân tích các thành phần & mô hình hệ thống cho ứng dụng.
- Build các Docker Image cần thiết và push lên ECR.
- Tạo file Kubernetes Config cho các hạng mục.
- Apply và kiểm tra.
- Tạo ALB, Target Group trỏ vào Kubernetes cluster.
- Test truy cập thông qua ALB DNS (cấu hình security group nếu cần)

Copyright@Linh Nguyen on Udemy
All right reserved

Lab 13 – Triển khai một ứng dụng mẫu lên Kubernetes cluster

Sơ đồ hệ thống của bài lab



Tổng kết & Clear resources

- Terminate EC2 instance.
- Xoá hết các EBS Volume nếu còn lại.
- Kiểm tra các resource khác như NAT Gateway, Elastic IP.
- ECR Repository: có thể giữ lại để sử dụng sau này (tốn phí nhưng không đáng kể).

Copyright@Linh Nguyen on Udemy
All right reserved

Thanks you and see you in the next chapter!

Copyright@Linh Nguyen on Udemy
All right reserved