

# Nội dung

<u>Ưu điểm</u> .....	2
<u>Kubernetes không phải là gì?</u> .....	2
<u>Các thành phần</u> .....	4
<u>Control plane - master node</u> .....	4
<u>Pod</u> .....	5
<u>Node - worker node</u> .....	5
<u>Kubernetes Namespace</u> .....	6
<u>Hướng dẫn cài đặt trên Linux</u> .....	7
<u>Triển khai hệ thống</u> .....	7
<u>Sử dụng Kubeadm để deploy cho hệ thống thật</u> .....	7
<u>Sử dụng Minikube để deploy trên localhost</u> .....	7
<u>kubectl</u> .....	8
<u>Triển khai ứng dụng</u> .....	9
<u>Triển khai LoadBalancer</u> .....	10
<u>Quản lý cluster</u> .....	11

# Kubernetes

Kubernetes hoặc k8s là một nền tảng mã nguồn mở giúp tự động hóa việc quản lý, mở rộng và triển khai ứng dụng dưới dạng container. K8s còn được gọi là Container Orchestration Engine (hiểu nôm na là công cụ điều phối container). Kubernetes loại bỏ rất nhiều các quy trình thủ công liên quan đến việc triển khai và mở rộng các containerized applications.

Kubernetes Orchestration cho phép người dùng xây dựng các dịch vụ ứng dụng mở rộng nhiều containers, lên lịch các containers đó trên một cụm máy chủ (cluster), mở rộng các containers và quản lý tình trạng của các containers theo thời gian.

Kubernetes ban đầu được các kỹ sư Google phát triển. Và Google cũng là một trong những cái tên tiên phong đóng góp cho công cuộc phát triển công nghệ Linux container.

## Ưu điểm

**Service discovery and load balancing:** Kubernetes có thể expose một container sử dụng DNS hoặc địa chỉ IP của riêng nó. Nếu lượng traffic truy cập đến một container cao, Kubernetes có thể cân bằng tải và phân phối lưu lượng mạng (network traffic) để việc triển khai được ổn định.

**Điều phối bộ nhớ:** Kubernetes cho phép bạn tự động mount một hệ thống lưu trữ mà bạn chọn, như local storages, public cloud providers, v.v.

**Tự động rollouts và rollbacks:** Bạn có thể mô tả trạng thái mong muốn cho các container được triển khai dùng Kubernetes và nó có thể thay đổi trạng thái thực tế sang trạng thái mong muốn với tần suất được kiểm soát. Ví dụ, bạn có thể tự động hóa Kubernetes để tạo mới các container cho việc triển khai của bạn, xoá các container hiện có và áp dụng tất cả các resource của chúng vào container mới.

**Đóng gói tự động:** Bạn cung cấp cho Kubernetes một cluster gồm các node mà nó có thể sử dụng để chạy các tác vụ được đóng gói (containerized task). Bạn cho Kubernetes biết mỗi container cần bao nhiêu CPU và bộ nhớ (RAM). Kubernetes có thể điều phối các container đến các node để tận dụng tốt nhất các resource của bạn.

**Tự phục hồi:** Kubernetes khởi động lại các containers bị lỗi, thay thế các container, xoá các container không phản hồi lại cấu hình health check do người dùng xác định và không cho các client biết đến chúng cho đến khi chúng sẵn sàng hoạt động.

**Quản lý cấu hình và bảo mật:** Kubernetes cho phép bạn lưu trữ và quản lý các thông tin nhạy cảm như: password, OAuth token và SSH key. Bạn có thể triển khai và cập nhật lại secret và cấu hình ứng dụng mà không cần build lại các container image và không để lộ secret trong cấu hình stack của bạn.

## Kubernetes không phải là gì?

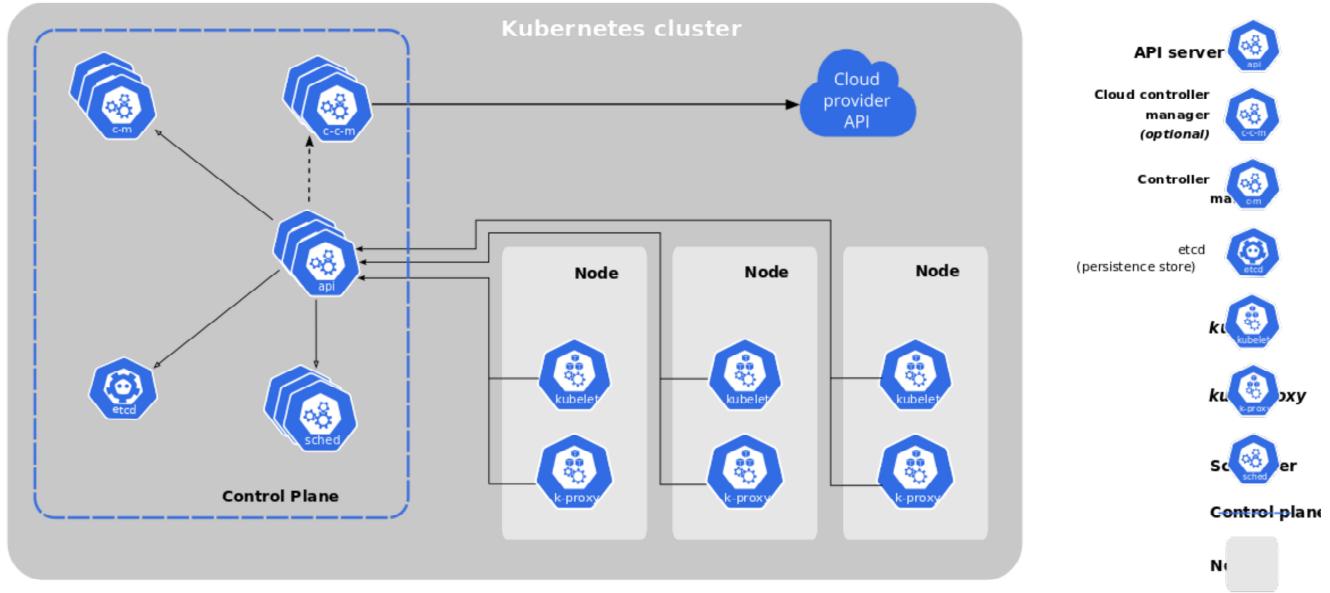
Kubernetes không phải là một hệ thống PaaS (Nền tảng như một Dịch vụ) truyền thống, toàn diện. Do Kubernetes hoạt động ở tầng container chứ không phải ở tầng phần cứng, nó cung cấp một số tính năng thường áp dụng chung cho các dịch vụ PaaS, như triển khai, nhân rộng, cân bằng tải, ghi nhật ký và giám sát. Tuy nhiên, Kubernetes không phải

là cấu trúc nguyên khôi và các giải pháp mã định này là tùy chọn và có thể cắm được (pluggable).

Kubernets:

- Không giới hạn các loại ứng dụng được hỗ trợ. Kubernetes nhằm mục đích hỗ trợ một khối lượng công việc cực kỳ đa dạng, bao gồm cả stateless, stateful và xử lý dữ liệu. Nếu một ứng dụng có thể chạy trong một container, nó sẽ chạy rất tốt trên Kubernetes.
- Không triển khai mã nguồn và không build ứng dụng của bạn. Quy trình CI/CD được xác định bởi tổ chức cũng như các yêu cầu kỹ thuật.
- Không cung cấp các service ở mức ứng dụng, như middleware (ví dụ, các message buses), các framework xử lý dữ liệu (ví dụ, Spark), cơ sở dữ liệu (ví dụ, MySQL), bộ nhớ cache, cũng như hệ thống lưu trữ của cluster (ví dụ, Ceph). Các thành phần như vậy có thể chạy trên Kubernetes và/hoặc có thể được truy cập bởi các ứng dụng chạy trên Kubernetes thông qua các cơ chế di động, chẳng hạn như Open Service Broker.
- Không bắt buộc các giải pháp ghi lại nhật ký (logging), giám sát (monitoring) hoặc cảnh báo (alerting). Nó cung cấp một số sự tích hợp như proof-of-concept, và cơ chế để thu thập và xuất các số liệu.
- Không cung cấp, không bắt buộc một cấu hình ngôn ngữ/hệ thống (ví dụ: Jsonnet). Nó cung cấp một API khai báo có thể được targeted bởi các hình thức khai báo tùy ý.
- Không cung cấp cũng như áp dụng bất kỳ cấu hình toàn diện, bảo trì, quản lý hoặc hệ thống tự phục hồi.
- Ngoài ra, Kubernetes không phải là một hệ thống điều phối đơn thuần. Trong thực tế, nó loại bỏ sự cần thiết của việc điều phối. Định nghĩa kỹ thuật của điều phối là việc thực thi một quy trình công việc được xác định: đầu tiên làm việc A, sau đó là B rồi sau chót là C. Ngược lại, Kubernetes bao gồm một tập các quy trình kiểm soát độc lập, có thể kết hợp, liên tục điều khiển trạng thái hiện tại theo trạng thái mong muốn đã cho. Nó không phải là vấn đề làm thế nào bạn có thể đi được từ A đến C. Kiểm soát tập trung cũng không bắt buộc. Điều này dẫn đến một hệ thống dễ sử dụng hơn, mạnh mẽ hơn, linh hoạt hơn và có thể mở rộng.

# Các thành phần



## Control plane - master node

Là server điều khiển các máy Worker chạy ứng dụng. Master node bao gồm 4 thành phần chính:

- Kubernetes API Server: là thành phần giúp các thành phần khác liên lạc nói chuyện với nhau. Lập trình viên khi triển khai ứng dụng sẽ gọi API Kubernetes API Server này.
- Scheduler: Thành phần này lập lịch triển khai cho các ứng dụng; theo dõi các Pod mới được tạo nhưng không có node được chỉ định và chọn một node để chúng chạy.
- Controller Manager: Thành phần đảm nhiệm quản lý các node, kiểm tra các node sống hay chết, đảm nhận việc nhân bản ứng dụng... Có các loại điều khiển như:
  - Node controller: Chịu trách nhiệm thông báo và phản hồi khi các node gặp sự cố.
  - Job controller: Xem xét các Job objects có các tác vụ chỉ xảy ra một lần, sau đó tạo các pod để chạy các ứng dụng đó cho đến khi hoàn thành.
  - Endpoints controller: Quản lý các endpoint.
  - Service Account & Token controllers: Tạo tài khoản mặc định và các token truy cập API cho namespaces mới.
- Etcd: Đây là cơ sở dữ liệu của Kubernetes, tất cả các thông tin của Kubernetes được lưu trữ cố định vào đây.

- Cloud controller manager:

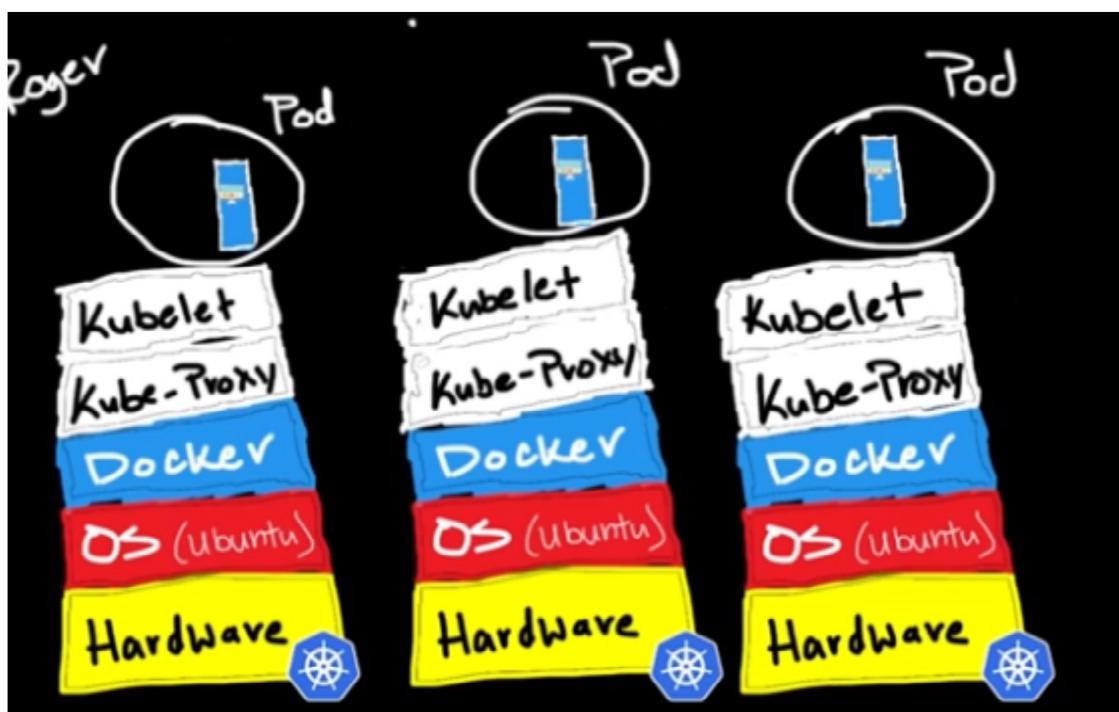
- o Cho phép bạn liên kết cluster của mình với API của nhà cung cấp đám mây và tách các thành phần tương tác với nền tảng đám mây đó khỏi các thành phần chỉ tương tác với cluster của bạn.
- o Cloud-controller-manager chỉ chạy các bộ điều khiển dành riêng cho nhà cung cấp dịch vụ đám mây của bạn. Nếu bạn đang chạy Kubernetes tại cơ sở của riêng mình hoặc trong môi trường học tập bên trong PC của riêng bạn, cluster không có cloud controller manager.

## Pod

Pod là khái niệm cơ bản và quan trọng nhất trên Kubernetes. Bản thân Pod có thể chứa 1 hoặc nhiều hơn 1 container. Pod chính là nơi ứng dụng được chạy trong đó. Pod là các tiến trình nằm trên các Worker Node. Bản thân Pod có tài nguyên riêng về file system, cpu, ram, volumes, địa chỉ network... Nhóm này chia sẻ không gian lưu trữ, địa chỉ IP với nhau. Pod thì được tạo ra hoặc xóa tùy thuộc vào yêu cầu của dự án.

## Node - worker node

Các thành phần node chạy trên mọi node, duy trì các nhóm đang chạy và cung cấp môi trường chạy Kubernetes.



kubelet: Kubebet đảm bảo rằng các container đang chạy trong một pod. Kubelet lấy một bộ PodSpec được cung cấp thông qua các cơ chế khác nhau và đảm bảo rằng các container được mô tả trong các PodSpec đó đang chạy và khỏe mạnh. Kubelet không quản lý các container không được tạo ra bởi Kubernetes.

kube-proxy: là một proxy mạng chạy trên mỗi node trong cluster của bạn, triển khai một phần của khái niệm dịch vụ Kubernetes. Kube-Proxy duy trì các quy tắc mạng trên các node. Các quy tắc mạng này cho phép giao tiếp mạng đến Pod của bạn từ các phiên mạng bên trong hoặc bên ngoài cluster.

## Kubernetes Namespace

Kubernetes hỗ trợ nhiều cluster ảo được cung cấp bởi cùng một cluster vật lý. Những cụm ảo này được gọi là namespace. Nó cho phép bạn phân vùng tài nguyên thành các nhóm được đặt tên hợp lý.

## Hướng dẫn cài đặt trên Linux

### Triển khai hệ thống

#### Sử dụng Kubeadm để deploy cho hệ thống thật

Phương pháp này có thể deploy trên các nền tảng sau: AWS, GCE, Azure, Joyent, OpenStack, VMWare, Bare Metal and localhost. Kubeadm được hỗ trợ chính thức bởi Kubernetes, do vậy chúng ta sẽ sử dụng phương pháp này để deploy cho hệ thống thật.

#### Sử dụng Minikube để deploy trên localhost

Minikube là một bộ cài đặt Kubernetes bằng cách tạo ra một máy ảo trên máy tính của bạn và triển khai một cluster đơn giản bên trong máy ảo đó chỉ bao gồm một Node. Minikube có cho Linux, macOS, và Windows.

Yêu cầu:

- 2 CPU trớ lén
- 2GB bộ nhớ trống
- 20GB dung lượng đĩa trống
- Kết nối Internet
- Trình quản lý vùng chứa hoặc máy ảo, chẳng hạn như: Docker, Hyperkit, Hyper-V, KVM, Parallels, Podman, VirtualBox hoặc VMware Fusion / Workstation

\*Ghi chú: Trong tài liệu này, sẽ hướng dẫn bạn cài đặt Kubernetes trên Linux.

Để kiểm tra xem việc ảo hóa (virtualization) có được hỗ trợ trên Linux không, chạy lệnh sau và chắc chắn rằng kết quả trả về là non-empty:

```
grep -E --color 'vmx|svm' /proc/cpuinfo
```

Yêu cầu máy tính đã cài đặt docker. Thêm USER vào docker group:

```
sudo usermod -aG docker $USER && newgrp docker
```

Chuyển Docker storage driver sang overlay2, để minikube hoạt động tương thích hơn.

Cài minikube phiên bản ổn định mới nhất trên kiến trúc x86-64 bằng RPM pakage:

```
curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-latest.x86_64.rpm  
sudo rpm -Uvh minikube-latest.x86_64.rpm
```

Bắt đầu với cluster

```
minikube start
```

## kubectl

kubectl là công cụ quản trị Kubernetes, được cài đặt trên các máy trạm, cho phép các lập trình viên đầy đủ các ứng dụng mô tả triển khai vào Kubernetes cluster, cũng như là cho phép các quản trị viên có thể quản trị được cluster.

1. Tải về phiên bản mới nhất với câu lệnh:

```
curl -LO https://storage.googleapis.com/kubernetes-release/release/`curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt`/bin/linux/amd64/kubectl
```

2. Tạo kubectl binary thực thi:

```
chmod +x ./kubectl
```

3. Đưa bản binary vào biến môi trường PATH của bạn:

```
sudo mv ./kubectl /usr/local/bin/kubectl
```

4. Kiểm tra chắc chắn rằng phiên bản bạn cài là mới nhất:

```
kubectl version
```

5. Kiểm tra kubectl được cấu hình đúng bằng việc xem trạng thái của cluster:

```
kubectl cluster-info
```

Nếu bạn trông thấy một URL phản hồi, thì kubectl đã được cấu hình đúng để truy cập vào cluster của bạn.

Nếu bạn trông thấy một tin nhắn tương tự bên dưới, thì kubectl chưa được cấu hình đúng hoặc chưa thể kết nối với Kubernetes cluster.

```
The connection to the server <server-name:port> was refused - did you specify the right host or port?
```

Ví dụ như, nếu bạn đang định chạy một Kubernetes cluster trên laptop của bạn (locally), bạn sẽ cần một công cụ như Minikube được cài trước đó và chạy lại các câu lệnh bên trên.

Nếu kubectl cluster-info trả về url nhưng bạn không thể truy cập vào cluster của bạn, thì hãy kiểm tra nó đã được cấu hình đúng hay chưa, bằng cách:

```
kubectl cluster-info dump
```

Kiểm tra danh sách các Pod:

```
kubectl get pods --all-namespaces
```

Kiểm tra danh sách các node:

```
kubectl get nodes
```

Kiểm tra danh sách docker container:

```
sudo docker ps
```

Dừng cluster:

```
minikube stop
```

Xóa cluster:

```
minikube delete
```

Truy cập Kubernetes dashboard:

```
minikube dashboard
```

## Triển khai ứng dụng

Tạo một ứng dụng và hiển thị nó trên cổng 8080:

```
kubectl create deployment hello-minikube --image=k8s.gcr.io/echoserver:1.4  
kubectl expose deployment hello-minikube --type=NodePort --port=8080
```

Có thể mất một chút thời gian, nhưng triển khai của bạn sẽ sớm hiển thị khi bạn chạy câu lệnh:

```
kubectl get services hello-minikube
```

Cách dễ nhất để truy cập ứng dụng này là để minikube khởi chạy trình duyệt web cho bạn:

```
minikube service hello-minikube
```

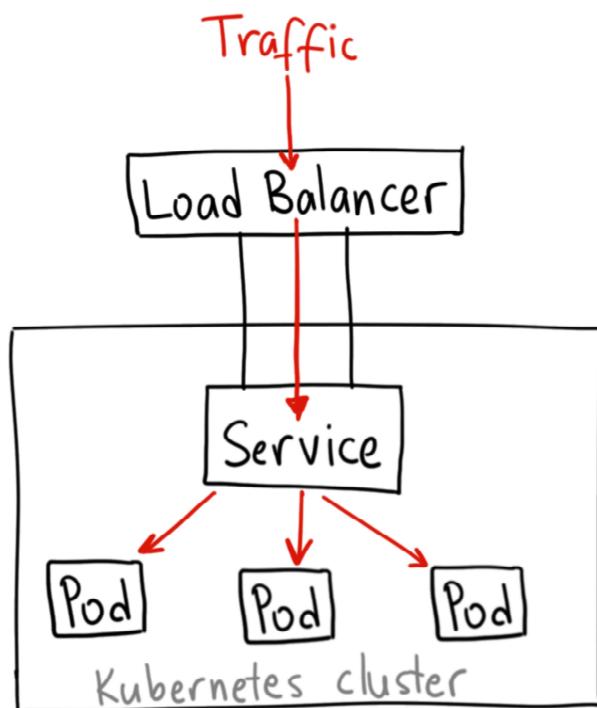
Ngoài ra, để dễ truy cập, sử dụng kubectl để chuyển tiếp cổng:

```
kubectl port-forward service/hello-minikube 7080:8080
```

Ứng dụng của bạn hiện tại có thể truy cập qua đường dẫn <http://localhost:7080>

## Triển khai LoadBalancer

Một LoadBalancer Service là cách thức chuẩn để expose service với Internet.



Nếu bạn muốn trực tiếp expose một service, đây là phương thức mặc định. Tất cả các traffic trên port bạn đã chỉ định sẽ được chuyển tiếp tới service. Sẽ không có filter, routing,... Do đó bạn có thể gửi hầu hết các loại traffic đến nó, như HTTP, TCP, UDP, WebSockets, gRPC.

Để truy cập triển khai LoadBalancer, hãy sử dụng lệnh “minikube tunnel”. Đây là một ví dụ triển khai:

```
kubectl create deployment balanced --image=k8s.gcr.io/echoserver:1.4
```

```
kubectl expose deployment balanced --type=LoadBalancer --port=8080
```

Trong một cửa sổ terminal khác, dùng tunnel để tạo một IP có thể định tuyến cho việc triển khai 'cân bằng':

```
minikube tunnel
```

Để tìm IP có thể định tuyến, hãy chạy lệnh này và kiểm tra cột EXTERNAL-IP

```
kubectl get services balanced
```

Triển khai của bạn hiện có sẵn tại <EXTERNAL-IP>: 8080

## Quản lý cluster

Tạm dừng Kubernetes mà không ảnh hưởng đến các ứng dụng đã triển khai:

```
minikube pause
```

Bỏ tạm dừng một phiên bản đã tạm dừng:

```
minikube unpause
```

Tạm dừng cluster:

```
minikube stop
```

Tăng giới hạn bộ nhớ mặc định (yêu cầu khởi động lại):

```
minikube config set memory 16384
```

Duyệt qua danh mục các add-ons đã được cài đặt:

```
minikube addons list
```

Tạo một cluster thứ hai chạy Kubernetes cũ hơn:

```
minikube start -p aged --kubernetes-version=v1.16.1
```

Xóa tất cả các minikube cluster:

```
minikube delete --all
```