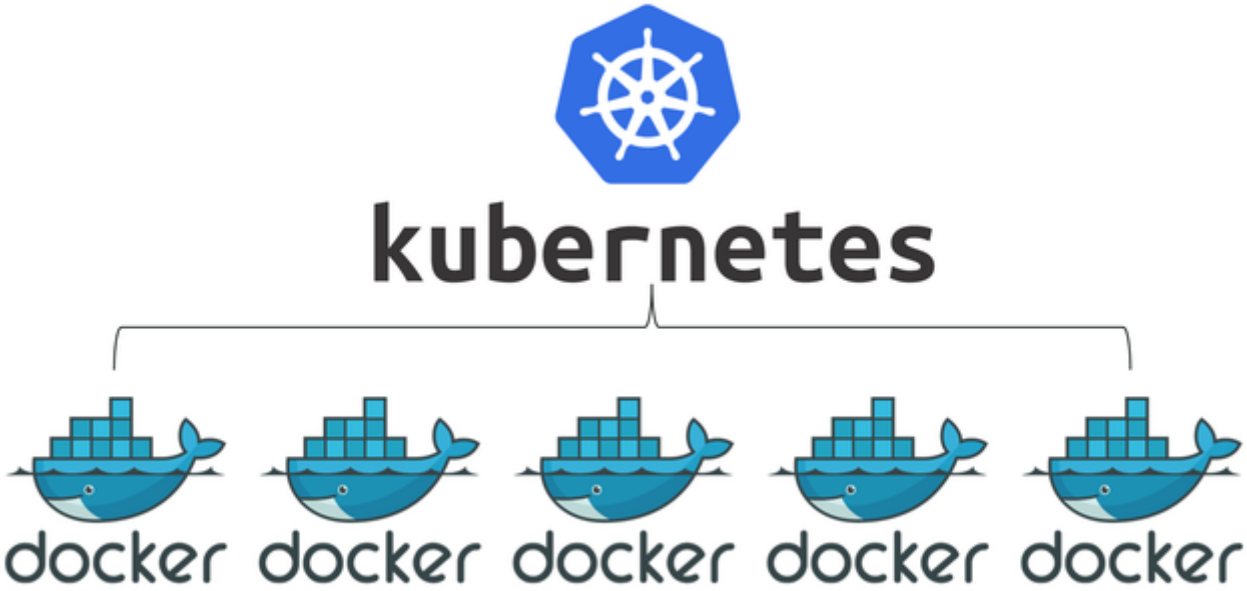
Giới thiệu Kubernetes

1. Kubernetes là gì?

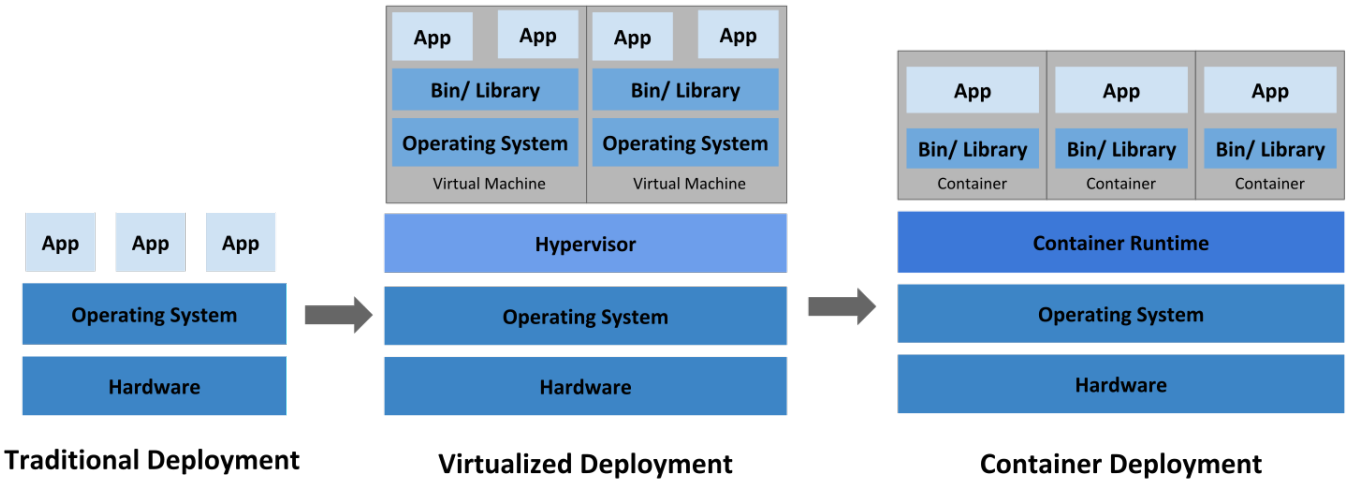
Kubernetes là một Platform được áp dụng để tự động hoá việc quản lý, scaling(mở rộng) và triển khai ứng dụng dưới dạng container hoá cụ thể là các docker, Kubernetes còn gọi là Container orchestration engine.



Các đặc điểm chính của Kubernetes bao gồm:

- Quản lý các container cluster
- Cung cấp các công cụ để triển khai các ứng dụng
- Mở rộng các ứng dụng khi cần thiết
- Quản lý các thay đổi đối với các ứng dụng được container hóa hiện có
- Giúp tối ưu hóa việc sử dụng phần cứng cơ bản bên dưới container
- Cho phép thành phần ứng dụng khởi động lại và di chuyển trên toàn hệ thống khi cần thiết

2. Các mô hình triển khai ứng dụng



Mô trình truyền thống: Ban đầu, các ứng dụng được chạy trên các máy chủ vật lý. Không có cách nào để xác định ranh giới tài nguyên cho các ứng dụng trong máy chủ vật lý và điều này gây ra sự cố phân bổ tài nguyên. Ví dụ, nếu nhiều ứng dụng cùng chạy trên một máy chủ vật lý, có thể có những trường hợp một ứng dụng sẽ chiếm phần lớn tài nguyên hơn và kết quả là các ứng dụng khác sẽ hoạt động kém đi. Một giải pháp cho điều này sẽ là chạy từng ứng dụng trên một máy chủ vật lý khác nhau. Nhưng giải pháp này không tối ưu vì tài nguyên không được sử dụng đúng mức và rất tốn kém cho các tổ chức để có thể duy trì nhiều máy chủ vật lý như vậy.

Mô hình ảo hóa: Như một giải pháp, ảo hóa đã được giới thiệu. Nó cho phép bạn chạy nhiều Máy ảo (VM) trên CPU của một máy chủ vật lý. Ảo hóa cho phép các ứng dụng được cô lập giữa các VM và cung cấp mức độ bảo mật vì thông tin của một ứng dụng không thể được truy cập tự do bởi một ứng dụng khác.

Ảo hóa cho phép sử dụng tốt hơn các tài nguyên trong một máy chủ vật lý và cho phép khả năng mở rộng tốt hơn vì một ứng dụng có thể được thêm hoặc cập nhật dễ dàng, giảm chi phí phần cứng và hơn thế nữa. Với ảo hóa, bạn có thể có một tập hợp các tài nguyên vật lý dưới dạng một cụm các máy ảo sẵn dùng.

Mỗi VM là một máy tính chạy tất cả các thành phần, bao gồm cả hệ điều hành riêng của nó, bên trên phần cứng được ảo hóa.

Môi trường Container: Các container tương tự như VM, nhưng chúng có tính cô lập để chia sẻ Hệ điều hành (HĐH) giữa các ứng dụng. Do đó, container được coi là nhẹ (lightweight). Tương tự như VM, một container có hệ thống tệp (filesystem), CPU, bộ nhớ, process space, v.v. Khi chúng được tách rời khỏi cơ sở hạ tầng bên dưới, chúng có thể khả chuyển (portable) trên cloud hoặc các bản phân phối Hệ điều hành.

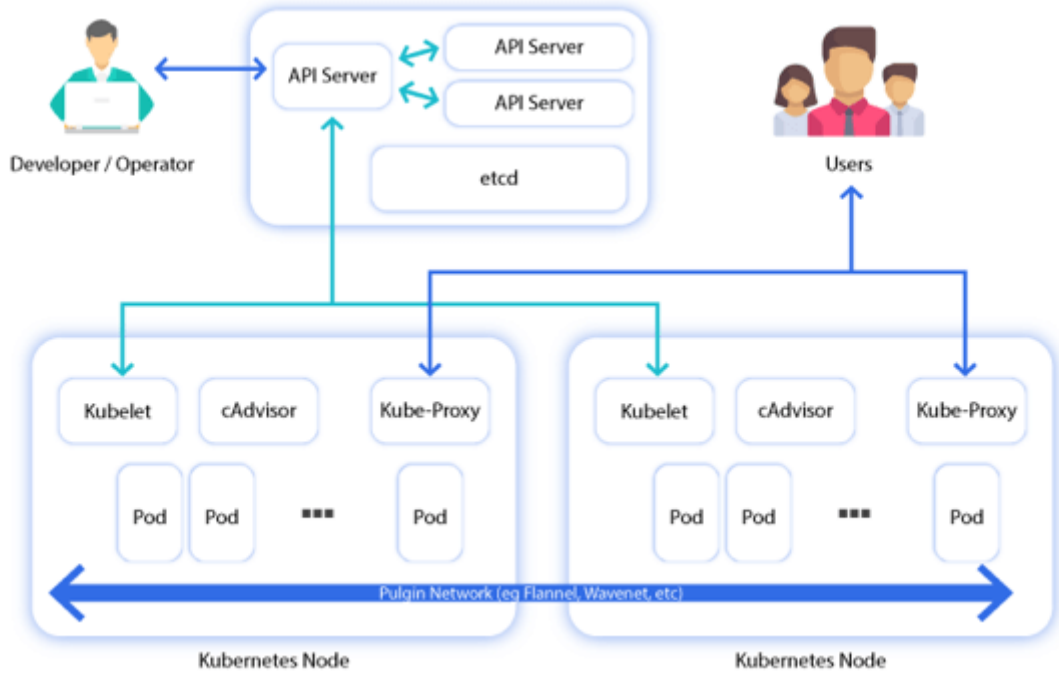
Các container đã trở nên phổ biến vì chúng có thêm nhiều lợi ích, chẳng hạn như:

- Tạo mới và triển khai ứng dụng Agile: gia tăng tính dễ dàng và hiệu quả của việc tạo các container image so với việc sử dụng VM image.
- Phát triển, tích hợp và triển khai liên tục: cung cấp khả năng build và triển khai container image thường xuyên và đáng tin cậy với việc rollbacks dễ dàng, nhanh chóng.

- Phân biệt giữa Dev và Ops: tạo các images của các application container tại thời điểm build/release thay vì thời gian triển khai, do đó phân tách các ứng dụng khỏi hạ tầng.
- Khả năng quan sát không chỉ hiển thị thông tin và các metric ở mức Hệ điều hành, mà còn cả application health và các tín hiệu khác.
- Tính nhất quán về môi trường trong suốt quá trình phát triển, testing và trong production: Chạy tương tự trên laptop như trên cloud.
- Tính khả chuyển trên cloud và các bản phân phối HĐH: Chạy trên Ubuntu, RHEL, CoreOS, on-premises, Google Kubernetes Engine và bất kì nơi nào khác.
- Quản lý tập trung ứng dụng: Tăng mức độ trừu tượng từ việc chạy một Hệ điều hành trên phần cứng ảo hóa sang chạy một ứng dụng trên một HĐH bằng logical resources.
- Các micro-services phân tán, elastic: ứng dụng được phân tách thành các phần nhỏ hơn, độc lập và thể được triển khai và quản lý một cách linh hoạt - chứ không phải một app nguyên khối (monolithic).
- Cô lập các tài nguyên: dự đoán hiệu năng ứng dụng
- Sử dụng tài nguyên: hiệu quả

3. Thành phần cơ bản của Kubernetes

Kubernetes cung cấp nhiều hơn ngoài framework cơ bản, cho phép người dùng chọn loại framework ứng dụng, ngôn ngữ, công cụ giám sát, logging và các công cụ khác mà họ chọn. Mặc dù nó không phải là Platform as a Service, nhưng nó có thể được sử dụng làm cơ sở cho một PaaS hoàn chỉnh.



Các thành phần cơ bản của Kubernetes bao gồm:

- **Kubernetes Master:** Đây là đơn vị điều khiển chính quản lý workloads và liên lạc trên toàn hệ thống. Mỗi thành phần của nó có một quy trình khác nhau có thể chạy trên một master node hoặc trên nhiều master node. Thành phần của nó là:
  - **Etcd Storage**: Đây là kho lưu trữ dữ liệu key-value, là nguồn mở được phát triển bởi nhóm CoreOS và có thể được truy cập bởi tất cả các node trong cluster. Kubernetes

sử dụng "Etcd" để lưu trữ dữ liệu cấu hình của cluster để thể hiện trạng thái chung của cluster bất cứ lúc nào.
    - **API-Server**: API server là thực thể quản lý trung tâm nhận các REST requests để cập nhật các thông tin cấu hình, đóng vai trò là front-end để điều khiển cluster. Hơn nữa, đây là thứ duy nhất giao tiếp với cluster Etcd, và đảm bảo rằng dữ liệu được lưu trữ trong Etcd.
    - **Scheduler**: Nó giúp lên lịch các pod trên các node khác nhau dựa trên việc sử dụng tài nguyên và quyết định nơi nào triển khai dịch vụ nào. Bộ lập lịch có các thông tin liên quan đến các tài nguyên có sẵn cho các thành viên cũng như tài nguyên còn lại để cấu hình dịch vụ để chạy.
    - **Controller Manager**: Thực hiện một số quy trình điều khiển riêng biệt trong nền để điều chỉnh trạng thái chia sẻ của cluster và thực hiện một tác vụ theo quy luật. Khi có bất kỳ thay đổi nào trong dịch vụ, bộ điều khiển sẽ phát hiện ra sự thay đổi và bắt đầu làm việc theo trạng thái mong muốn mới.

- **Worker Node:** Đây còn được gọi là Minion node, nó chứa thông tin để quản lý kết nối mạng giữa các container như Docker và liên lạc giữa master node khi gán tài nguyên cho các container theo lịch trình
    - **Kubelet**: Kubelet đảm bảo rằng tất cả các container trong node đang chạy và ở trạng thái healthy. Kubelet theo dõi trạng thái của một pod nếu nó không ở trạng thái mong muốn. Nếu một node thất bại, bộ điều khiển sao chép sẽ quan sát sự thay đổi này và khởi chạy các pod trên một pod healthy khác.
    - **Container**: Container là mức thấp nhất của microservice, được đặt bên trong pod và cần địa chỉ IP bên ngoài để xem xét quy trình bên ngoài.
    - **Kube Proxy**: Nó hoạt động như một proxy mạng và bộ cân bằng tải. Ngoài ra, nó chuyển tiếp yêu cầu tới các pod chính xác trên các mạng bị cô lập trong một cluster.
    - **cAdvisor**: Hoạt động như một trợ lý chịu trách nhiệm theo dõi và thu thập dữ liệu về việc sử dụng tài nguyên và số liệu hiệu suất trên mỗi node.

4. Ưu điểm của Kubernetes

- **Nguồn mở và di động:** Kubernetes có thể chạy các container trên một hoặc nhiều môi trường public cloud, máy ảo hoặc trên bare metal, điều đó có nghĩa là nó có thể được triển khai trên bất kỳ cơ sở hạ tầng nào. Hơn nữa, nó tương thích trên nhiều nền tảng, giúp cho chiến lược đa cloud trở nên linh hoạt và có thể sử dụng được.
- **Khả năng mở rộng workloads:** Kubernetes cung cấp một số tính năng hữu ích cho mục đích mở rộng:
    - **Horizontal Infrastructure Scaling:** Các thao tác được thực hiện ở cấp máy chủ riêng lẻ để thực hiện chia tỷ lệ theo chiều ngang. Máy chủ mới có thể được thêm hoặc gỡ bỏ dễ dàng.
    - **Auto-Scaling:** Dựa trên việc sử dụng tài nguyên CPU hoặc các số liệu ứng dụng khác, có thể sửa đổi số lượng container đang chạy.
    - **Manual Scaling:** Có thể thay đổi số lượng container đang chạy thông qua một lệnh hoặc giao diện.
    - **Replication Controller:** Bộ điều khiển sao chép đảm bảo rằng cluster có số lượng pod tương đương được chỉ định trong điều kiện đang chạy. Nếu có quá nhiều pod, bộ

điều khiển sao chép có thể loại bỏ các pod bổ sung hoặc ngược lại.
- **Tính sẵn sàng cao:** Kubernetes có thể xử lý sự sẵn có của cả ứng dụng và cơ sở hạ tầng. Nó đã khắc phục được:
  - **Health Checks:** Kubernetes đảm bảo rằng ứng dụng không bị lỗi bằng cách liên tục kiểm tra tình trạng của các mode và container. Kubernetes cung cấp khả năng tự phục hồi và tự động thay thế nếu pod bị hỏng do lỗi.
  - **Định tuyến lưu lượng và cân bằng tải:** Bộ cân bằng tải Kubernetes phân phối tải trên nhiều tải, cho phép cân bằng tài nguyên nhanh chóng trong lưu lượng ngẫu nhiên hoặc xử lý hàng loạt.

5. Tại sao bạn cần Kubernetes và nó có thể làm những gì?

Các container là một cách tốt để đóng gói và chạy các ứng dụng của bạn. Trong môi trường production, bạn cần quản lý các container chạy các ứng dụng và đảm bảo rằng không có khoảng thời gian downtime. Ví dụ, nếu một container bị tắt đi, một container khác cần phải khởi động lên. Điều này sẽ dễ dàng hơn nếu được xử lý bởi một hệ thống.

Đó là cách Kubernetes đến với chúng ta. Kubernetes cung cấp cho bạn một framework để chạy các hệ phân tán một cách mạnh mẽ. Nó đảm nhiệm việc nhân rộng và chuyển đổi dự phòng cho ứng dụng của bạn, cung cấp các mẫu deployment và hơn thế nữa. Ví dụ, Kubernetes có thể dễ dàng quản lý một triển khai canary cho hệ thống của bạn.

Kubernetes cung cấp cho bạn:

- **Service discovery và cân bằng tải**
  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 hoá 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.

6. Được thiết kế để triển khai

Triển khai ứng dụng theo mô hình container hóa có khả năng tăng tốc quá trình xây dựng, thử nghiệm và phát hành phần mềm và tính năng hữu ích bao gồm:

- **Tự động triển khai và khôi phục:** Kubernetes xử lý phiên bản mới và cập nhật cho ứng dụng mà không có downtime, đồng thời theo dõi health trong quá trình triển khai. Nếu bất kỳ lỗi nào xảy ra trong quá trình, nó sẽ tự động khôi phục.
- **Triển khai Canary:** Kubernetes kiểm tra song song việc sản xuất triển khai mới và phiên bản trước đó, trước khi tăng quy mô triển khai mới và đồng thời giảm quy mô triển khai trước đó.
- **Hỗ trợ Framework và ngôn ngữ lập trình:** Kubernetes hỗ trợ hầu hết các ngôn ngữ và framework lập trình như Java, .NET, v.v. và cũng nhận được sự hỗ trợ lớn từ cộng đồng phát triển. Nếu một ứng dụng có khả năng chạy trong một container, nó cũng có thể chạy trong Kubernetes.
- **Và nhiều hơn nữa:** Kubernetes cung cấp quản lý DNS, giám sát tài nguyên, ghi nhật ký, sắp xếp lưu trữ và cũng giải quyết vấn đề bảo mật. Chẳng hạn, nó đảm bảo rằng thông tin như mật khẩu hoặc ssh keys được lưu trữ an toàn trong các Kubernetes secret. Các tính năng mới được phát hành liên tục và có thể có trên Kubernetes GitHub.

7. 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 khối và các giải pháp mặ định này là tùy chọn và có thể cắm được (pluggable).

Kubernetes:

- 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

---

Revision #4
Created 2 December 2021 15:18:13 by Laptrinh.vn
Updated 21 February 2022 14:35:47 by Laptrinh.vn