Lesson 1: Introduction to Kubernetes

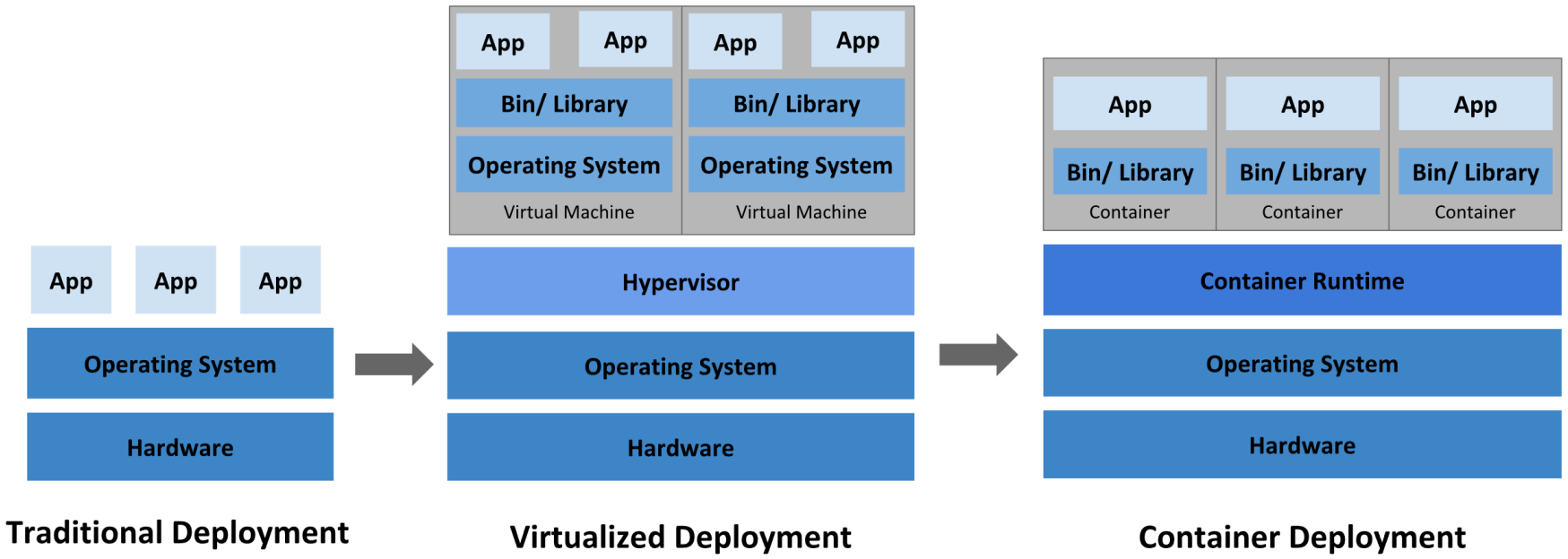
Lữ Thanh Tùng

Kubernetes là gì ?

- Kubernetes (K8s) là một nền tảng mã nguồn mở của Google từ năm 2014.
- Nó có tính khả chuyển (portable), có thể mở rộng (extensible), dùng để quản lý các ứng dụng và dịch vụ (service) được đóng gói (container), giúp thuận lợi trong việc cấu hình và tự động hoá việc triển khai ứng dụng.
- Kubernetes là một hệ sinh thái lớn và phát triển nhanh chóng với các dịch vụ, sự hỗ trợ và công cụ có sẵn rộng rãi.
- Tên gọi Kubernetes có nguồn gốc từ tiếng Hy Lạp, có ý nghĩa là người lái tàu hoặc hoa tiêu.
- Kubernetes là xây dựng dựa trên 15 năm kinh nghiệm mà Google với việc vận hành một khối lượng lớn công việc trong thực tế, kết hợp với các ý tưởng và thực tiễn tốt nhất từ cộng đồng.

Containerization là gì?

1. Lịch sử phát triển của việc triển khai (deployment) phần mềm



Containerization là gì?

2. Triển khai theo cách truyền thống *(Traditional deployment)*

- Các ứng dụng được chạy trên các máy chủ vật lý
- Không có cách để xác định ranh giới tài nguyên cho các ứng dụng trên cùng một máy chủ và gây ra vấn đề cấp phát 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 là chạy từng ứng dụng trên một máy chủ vật lý khác nhau.
- => Không thể mở rộng quy mô vì tài nguyên không được sử dụng hợp lý và rất tốn kém để có thể duy trì nhiều máy chủ vật lý .

Containerization là gì?

3. Triển khai theo cách ảo hóa *(Virtualized deployment)*

- Ảo hóa cho phép chạy nhiều Máy ảo (VM) trên CPU của một máy chủ vật lý
- Các ứng dụng được cô lập giữa các VM và cung cấp một lớp bảo mật (thông tin của một ứng dụng không thể được truy cập tự do bởi một ứng dụng khác)
- Sử dụng tốt hơn các tài nguyên trong một máy chủ vật lý và cho phép khả năng mở rộng tốt hơn (một ứng dụng có thể được thêm hoặc cập nhật dễ dàng, giảm chi phí phần cứng)
- Với ảo hóa, các tài nguyên vật lý có thể dùng dưới dạng một cụm các máy ảo . 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.
- **Nhược điểm:** Mỗi máy ảo chiếm nhiều tài nguyên do luôn có hệ điều hành kèm theo

Containerization là gì?

4. Triển khai theo container *(Container deployment)*

- Các container tương tự như VM, nhưng chúng có tính cô lập để chia sẻ Hệ điều hành (HĐH) giữa các ứng dụng thay vì dùng hệ điều hành riêng cho như ở VM.
- => 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 của Hệ điều hành (OS distributions).

Tại sao Kubernetes quan trọng?

- Các container là một cách tốt để đóng gói và chạy các ứng dụng. Tuy nhiên cần quản lý các container chạy các ứng dụng và đảm bảo rằng không có khoảng thời gian chết.
    - 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.

=> Kubernetes đã giải quyết chúng.

- Kubernetes cung cấp một framework để chạy các hệ phân tán một cách mạnh mẽ. Nó đảm nhiệm việc mở rộng và cung cấp khả năng chống lỗi cho ứng dụng, cung cấp các mẫu deployment

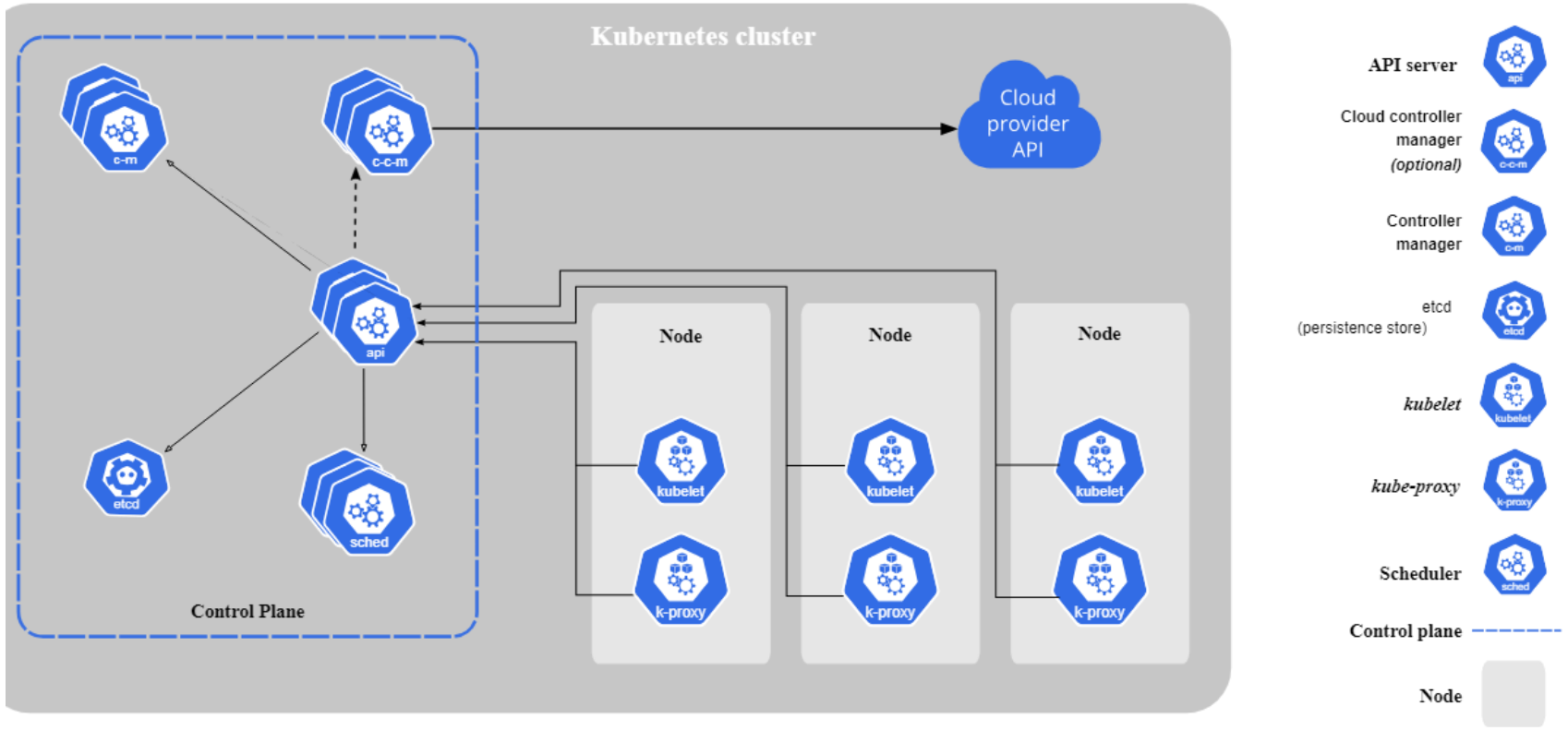
Tại sao Kubernetes quan trọng?

Kubernets cung cấp các tính năng:

- **Service discovery**: Hiển thị một container sử dụng DNS hoặc địa chỉ IP của riêng nó.
- **Cân bằng tải:** Nếu lưu lượng 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**: Cho phép tự động gắn một hệ thống lưu trữ mà người dung chọn như local storages, public cloud providers, …
- **Tự động rollouts và rollbacks**: Các container có thể thay đổi trạng thái thực tế sang trạng thái mong muốn với tần suất được kiểm soát.
  - Ví dụ, Tự động hoá Kubernetes để tạo mới các container cho việc triển khai, xoá các container hiện có và áp dụng tất cả các resource của chúng vào container mới.
- **Đóng gói tự động**: Nếu cung cấp một cluster gồm các node dùng để chạy các tác vụ được đóng gói (containerized task), Kubernetes sẽ điều phối các container đã biết trước tài nguyên(CPU, RAM) đến các node đó sao cho tận dụng tốt nhất các resource .
- **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:** Lưu trữ và quản lý các thông tin nhạy cảm như: password, OAuth token và SSH key. 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 .

Kubernetes Components

- Khi deploy Kubernetes sẽ tạo ra một cụm cluster
- Một cụm Kubernetes bao gồm một tập hợp các máy worker (gọi là nodes), chạy các containerized application. Mỗi cluster có ít nhất một worker node.
- Worker node kết nối (host) đến các thành phần của application workload (gọi là Pods).



- Control plane quản lý các worker node và Pod trong cluster. Trong thực tế, Control plane thường chạy trên nhiều máy tính và một cluster chạy nhiều nút để đảm bảo tính chống chịu lỗi và khả năng sẵn sàng cao

Control Plane Components

- Control plane's components đưa ra các quyết định toàn cục cho cụm cluster (VD: lập lịch (scheduling)), phát hiện và phản hồi các sự kiện (cluster events) (VD: khởi động một pod mới khi một bản sao (replicas) được deploy không thể đáp ứng)
- Control plane components có thể chạy trên nhiều máy trong cluster. Tuy nhiên, các script thường sẽ khởi động tất cả các control plane components trên cùng một máy để tiện quản lý và không chạy user containers trên máy đó để tránh tranh chấp tài nguyên giữa control plane component và user container, đảm bảo hiệu suất hoạt động của chúng

Các component của Control Plane

- **kube-apiserver**
  - API sever là front end của Kubernetes control plane.
  - Việc thực thi chính của Kubernetes API server là kube-apiserver. kube-apiserver được thiết kế để scale horizontally (scale bằng cách triển khai thêm nhiều phiên bản (instance) hơn, có thể chạy nhiều instances của kube-apiserver and chúng có thể tự cân bằng bang thông giữa các instance)
- **etcd:** Kho lưu trữ key-value có tính nhất quán và khả dụng cao cho tất cả các cluster data
- **kube-scheduler:** theo dõi các Pod mới được tạo mà không đăng kí node và chọn cho nó một node để chạy.
- **kube-controller-manager: Gồm các loại controller:**
  - Node controller: Chịu trách nhiệm thông báo và phản hồi khi các node ngừng hoạt động
  - Job controller: quan sát các Job object (task chạy một lần), sau đó tạo Pods để chạy hoàn thành task đó
  - EndpointSlice controller: cung cấp liên kết giữa Services và Pods.
  - ServiceAccount controller: tao ra ServiceAccounts mặc định cho không gian mới.
- **cloud-controller-manager:** Liên kết giữa cluster và các API của nhà cung cấp cloud và tách các thành phần tương tác với cloud ra khỏi các thành phần chỉ tương tác với cluster

Node Components

- Node components chạy trên mọi node, duy trì hoạt động các pod và cung cấp môi trường runtime Kubernetes

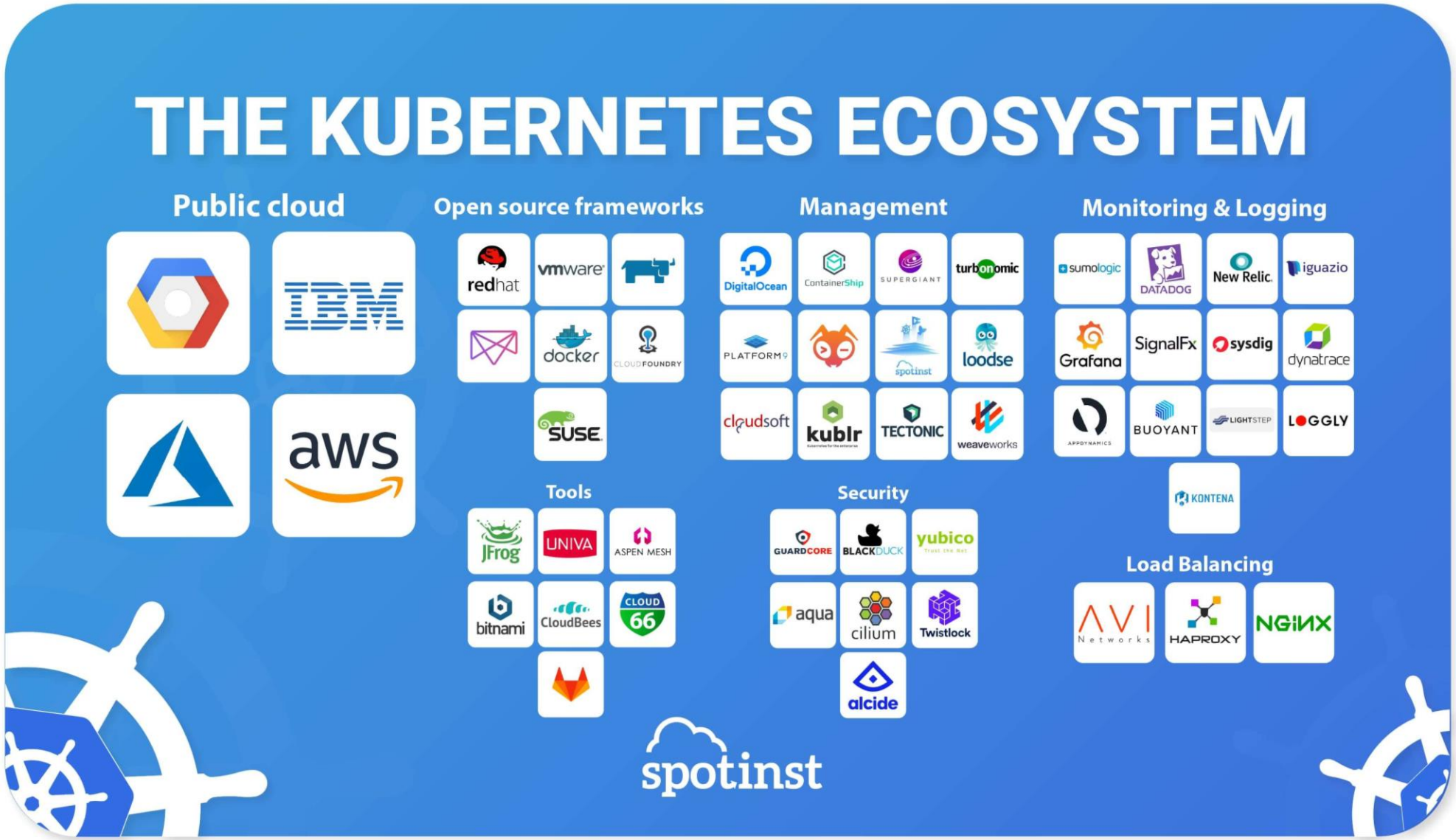
Các component của Node Component

- **kubelet**
  - Là tác tử chạy trên mỗi nút trong cluster, nó đảm bảo rằng các cointainer đang chạy trên một Pod
  - Kubelet nhận một tập các PodSpecs được cung cấp thông qua các cơ chế khác nhau và đảm bảo các cointainer được mô tả trong PodSpec luôn chạy và healthy. kubelet không quản lý các containers không được tạo bởi Kubernetes.
- **kube-proxy**
  - Là network proxy chạy trên mỗi nút trong cluster, thực thi một phần khái niệm của Kubernetes Service
  - kube-proxy duy trì quy tắc mạng trong nodes. Các quy tắc này cho phép các giao tiếp mạng đến các Pods từ các network sessions trong và ngoài cụm cluster
- **Container runtime**
  - Container runtime là phần mềm chịu trách nhiệm cho việc chạy các container.
  - Kubernetes hố trợ các container runtime như containerd, CRI-O, và bất kỳ triển khai nào khác của Kubernetes CRI (Container Runtime Interface)

Các component khác

- **Addons**: sử dụng các tài nguyên của Kubernetes (DaemonSet, Deployment, etc) để thực hiện các đặc trưng cluster.
- **DNS:** Tất cả các Kubernetes cluster đều phải có DNS**.** Cluster DNS là một DNS server.
  - Containers bắt đầu tự động bởi Kubernetes sẽ bao gồm DNS server trong DNS searches.
- **Web UI (Dashboard)**
  - Dashboard là một giao diện người dùng dựa trên web cho Kubernetes clusters. Nó cho phép người sử dụng quản lý và khắc phục sự cố các ứng dụng chạy trong cụm cluster cũng như cụm đó
- **Container Resource Monitoring**
  - Container Resource Monitoring ghi lại các chỉ số thời gian về containers trong một cơ sở dữ liệu trung tâm, và cung cấp một giao diện người dung (UI) để duyệt dữ liệu đó.
- **Cluster-level Logging**
  - Lưu lại các container log vào kho lưu trữ log với giao diện search/browsing.
- **Network Plugins**
  - Là thành phần phần mềm triển khai đặc tả container giao diện mạng(CNI)
  - Cấp phát địa chỉ IP đến các pod và cho phép chúng giao thiếp với những pod khác trong cụm cluster

Kubernetes Ecosystem



Vai trò của Kubernetes trong việc phát triển các ứng dụng hiện đại

- **Khả năng mở rộng** : Hiệu quả trong quá trình phát triển các ứng dụng cần mở rộng theo cả chiều dọc và chiều ngang (horizontally and vertically). Bằng cách sử dụng các cơ chế tích hợp sẵn như autoscaling, rolling updates, helf-healing, nó đáp ứng các thay đổi về nhu cầu và đảm bảo mức sử dụng tài nguyên tối ưu.
- **Tính di động**: Do chạy các ứng dụng một cách nhất quán trên nhiều môi trường khác nhau, nó cho phép triển khai trên mọi cơ sở hạ tầng được hỗ trợ, bao gồm: on-premises data centers, public clouds, hoặc hybrid setups.
- **Tự động hóa**: Kubernetes tự động hóa nhiều khía cạnh quản lý và triển khai ứng dụng. Điều này giúp giảm thời gian và công sức can thiệp thủ công, và các nhóm phát triển có thể tập trung vào việc cung cấp các tính năng và các cải tiến mới.
- **Khả năng phục hồi:** Kubernetes có các cơ chế tích hợp để phát hiện và khôi phục các sự cố liên quan đến ứng dụng hoặc cơ sở hạ tầng. Do đó, người dung có thể dựa vào tính khả dụng và khả năng đáp ứng này, ngay cả khi gặp sự cố không mong muốn.
- **Tốc độ phát triển nhanh**: Developer có thể tạo và triển khai các ứng dụng nhanh hơn với có tính nhất quán cao. Kubernetes cho phép mở rộng quy mô và cho phép lặp lại các tính năng và bản cập nhật mới nhanh hơn, giảm thời gian giữa thay đổi code và triển khai.
- **Tiết kiệm chi phí**: Bằng cách tận dụng các tính năng của Kubernetes như intelligent resource allocation, autoscaling, efficient infrastructure usage, ta có thể tiết kiệm chi phí đáng kể so với các phương pháp triển khai truyền thống. Điều này đặc biệt đúng đối với cơ sở hạ tầng dựa trên cloud trả tiền khi sử dụng.