ĐẠI HỌC QUỐC GIA TP HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

BỘ MÔN HỆ THỐNG THÔNG TIN

**BÁO CÁO ĐỒ ÁN**

**CÁC HỆ THỐNG PHÂN TÁN**

**Đề tài :**

**KUBERNETES – K8S**

**GIẢNG VIÊN HƯỚNG DẪN :** TS. Thái Lê Vinh

**HỌC VIÊN THỰC HIỆN :**

20C12007 – Trần Đình Lâm

20C12030 – Huỳnh Lâm Phú Sĩ

**KHÓA :** K30

TP.HCM, 28 tháng 11 năm 2021

# **Thông tin chi tiết nhóm**

|  |  |  |  |
| --- | --- | --- | --- |
| **BẢNG PHÂN CÔNG & ĐÁNH GIÁ HOÀN THÀNH CÔNG VIỆC** | | | |
| **Người**  **thực hiện** | **Công việc thực hiện** | **Mức độ**  **hoàn thành** | **Đánh giá của nhóm** |
| 20C12007  Trần Đình Lâm |  | 70% | 70% |
|  |
|  |
|  |
|  |
| 20C12030  Huỳnh Lâm  Phú Sĩ |  | 70% | 70% |
|  |
|  |
|  |
|  |
|  |
|  |

Bảng 1: Thông tin chi tiết và phân công nhóm

1. **Tổng quan về Kubernetes**
2. **Đặt vấn đề**

Ngày nay, với sự phát triển của internet và nhu cầu thực hiện các tác vụ internet bùng nổ theo từng ngày, các hệ thống phân tán cũng ngày càng phát triển để theo kịp tiến độ phát triển đó. Các hệ thống phân tán ngày nay được phát triển để đáp ứng nhu cầu sử dụng to lớn của người dung cũng như như nhu cầu phát triển của các nhà phát triển. Do đó, nhằm mục đích đơn giản hóa quy trình phát triển và triển khai của các ứng dụng internet, nhiều công cụ mạnh mẽ đã được tạo ra. Trong đó, Kubernetes (K8s) là một hệ thống quản lí mã nguồn mở vô cùng mạnh mẽ được phát triển bởi Google

1. **Giới thiệu về Kubernetes**

Kubernet (K8s) là hệ thống quản lí container mã nguồn mở được Google phát triển và chính thức giới thiệu với cộng đồng vào năm 2014. Được phát triển dựa trên kinh nghiệm vận hành các sản phẩm có scale lớn của Google, nó là công cụ giúp các lập trình viên triển khai, mở rộng, quản lí các ứng dụng dưới dạng container, điều phối, theo dõi, xử lí và lên lịch vận hành các container ở các cluster và đảm bảo chúng vận hành trơn tru, đúng kế hoạch.

Kubernetes sau một thời gian phát triển đã trở thành một hệ sinh thái lớn, techstack lớn và công cụ hỗ trợ rộng rãi, mạnh mẽ.

1. **Triển khai ứng dụng bằng container**

Graphical user interface

Description automatically generated

Cách triển khai theo kiểu truyền thống: Ban đầu, các tổ chức vận hành ứng dụng 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 chạy trên một máy chủ vật lý, có thể có trường hợp một ứng dụng sẽ chiếm phần lớn tài nguyên và kết quả là các ứng dụng khác sẽ hoạt động kém hiệu quả. Giải pháp cho điều này là chạy từng ứng dụng trên một máy chủ vật lý khác nhau. Nhưng điều này không mở rộng quy mô do tài nguyên không được sử dụng đầy đủ và tốn kém để duy trì nhiều máy chủ vật lý.

Cách triển khai theo kiểu ảo hóa: Như một giải pháp khắc phục các điểm yếu cho cách triển khai truyền thống, ảo hóa đã ra đời. Nó cho phép nhà phát triển chạy nhiều Máy ảo (VM) trên một CPU của một máy chủ vật lý. Ảo hóa cho phép các ứng dụng được tách biệt giữa các máy ảo và cung cấp độ bảo mật cao vì thông tin của một ứng dụng này không thể truy cập tùy tiện bởi ứng dụng khác, giảm chi phí phần cứng và hơn thế nữa. Với ảo hóa, các nhà phát triển có thể trình bày một tập hợp các tài nguyên vật lý dưới dạng một cụm máy ảo dùng một lần. Mỗi máy ảo là một máy đầy đủ chạy tất cả các thành phần, bao gồm cả hệ điều hành riêng, trên phần cứng được ảo hóa.

Cách triển khai bằng container: Các container tương tự như máy ảo, nhưng chúng có đặc tính đóng để chia sẻ Hệ điều hành (OS) giữa các ứng dụng. Vì vậy, container ít chiếm tài nguyên hơn. Tương tự như một máy ảo, một vùng chứa có hệ thống tệp riêng của nó, chia sẻ CPU, bộ nhớ, không gian xử lý và hơn thế nữa. Khi chúng được tách ra khỏi cơ sở hạ tầng bên dưới, chúng có thể di động qua các đám mây và các hệ điều hành khác nhau.

Ưu điểm của triển khai ứng dụng bằng container:

* Tạo và triển khai ứng dụng linh hoạt: tăng tính đơn giản và hiệu quả của việc tạo image container so với việc sử dụng image máy ảo.
* Phát triển, tích hợp và triển khai liên tục (CI/CD): cung cấp giải pháp đáng tin cậy và ổn định cho việc build và deploy image container với khả năng rollback nhanh chóng và hiệu quả (do tính bất biến của image).
* Phân tách Dev và Ops: tạo image container ứng dụng tại thời điểm build / release hơn là thời điểm deploy, do đó tách ứng dụng khỏi cơ sở hạ tầng.
* Khả năng quan sát: không chỉ hiển thị thông tin và số liệu cấp hệ điều hành, mà còn hiển thị tình trạng ứng dụng và các tín hiệu khác.
* Tính nhất quán về môi trường xuyên suốt quá trình develop, test và production: Chạy trên laptop giống như chạy trên đám mây
* Khả năng phân tán trên OS và đám mây: Chạy trên Ubuntu, RHEL, CoreOS, trên các dịch vụ đám mây công cộng lớn và bất kỳ nơi nào khác.
* Quản lý tập trung vào ứng dụng: Nâng cao mức độ trừu tượng từ việc chạy một hệ điều hành trên phần cứng ảo sang chạy một ứng dụng trên một hệ điều hành sử dụng tài nguyên logic.
* Các micro-service được kết hợp, phân tán, giải phóng không phụ thuộc lẫn nhau: các ứng dụng được chia thành các phần nhỏ hơn, độc lập và có thể được triển khai và quản lý động - không phải là monolithic stack chạy trên một máy chủ đơn mục đích lớn.
* Cách ly tài nguyên: hiệu suất ứng dụng có thể dự đoán được.
* Sử dụng tài nguyên hiệu quả cao.

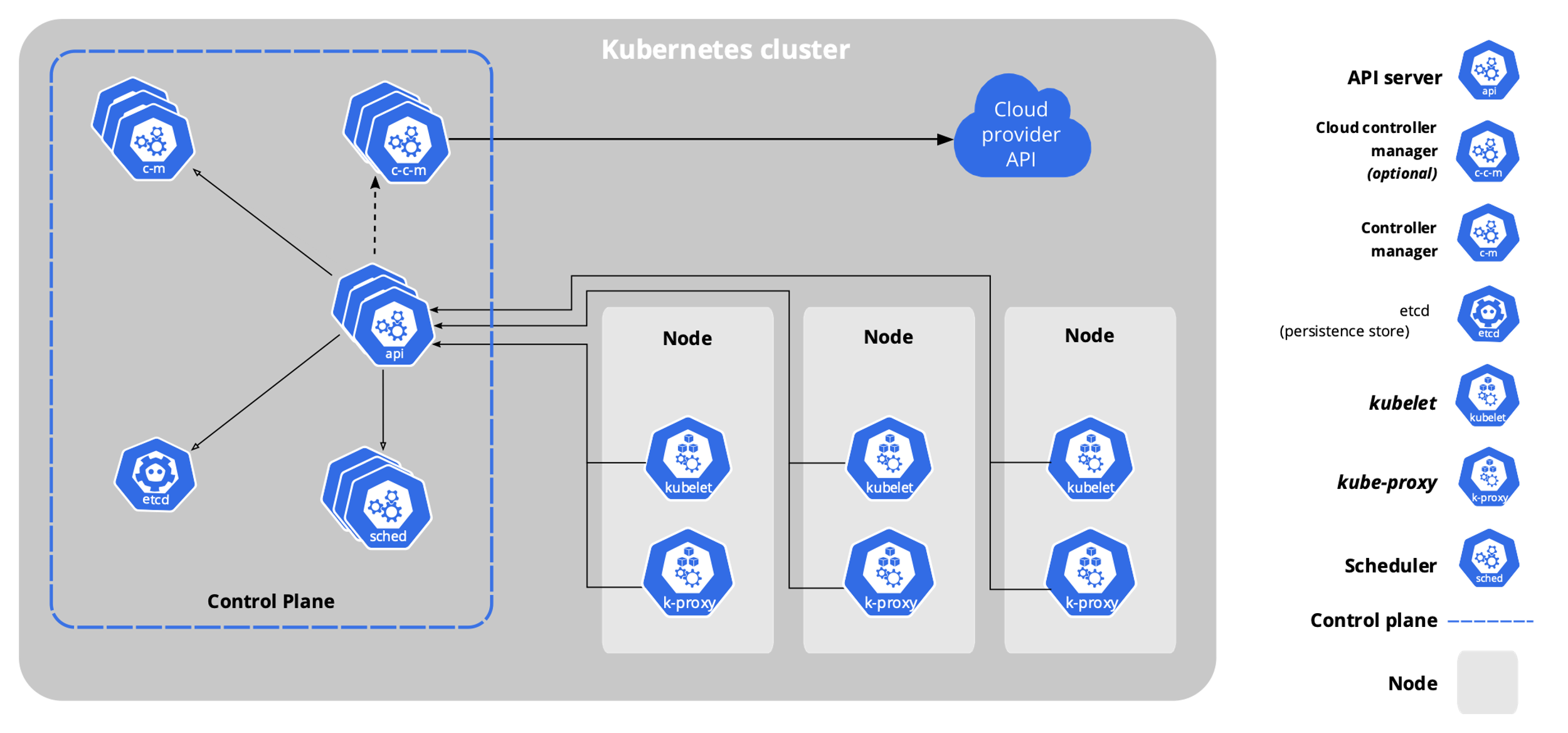
1. **Kubernetes cung cấp những tính năng gì?**

Với Kubernetes, các nhà phát triển có thể:

* Service discovery và load balancing: Kubernetes có thể công khai một container bằng tên DNS hoặc sử dụng địa chỉ IP của riêng chúng. Nếu lưu lượng truy cập vào vùng chứa cao, Kubernetes có thể cân bằng tải và phân phối lưu lượng mạng để việc triển khai ổn định.
* Điều phối storage của Kubernetes cho phép các nhà phát triển tự động gắn kết hệ thống lưu trữ phù hợp với nhu cầu, chẳng hạn như kho lưu trữ cục bộ, nhà cung cấp dịch vụ đám mây công cộng, v.v.
* Automatic Rollout và rollback: Bằng cách mô tả trạng thái mong muốn cho các container đã triển khai bằng Kubernetes và nó có thể thay đổi trạng thái thực tế thành trạng thái mong muốn với tốc độ có thể kiểm soát. Ví dụ: có thể tự động hóa Kubernetes để tạo các container mới cho việc triển khai, xóa các container hiện có và lấy tất cả tài nguyên của chúng cho container mới.
* Đóng gói tự động: Bằng cách cung cấp cho Kubernetes một nhóm các nod mà nó có thể sử dụng để chạy các tác vụ được đóng gói. Thiết lập cho Kubernetes mỗi container cần bao nhiêu CPU và bộ nhớ (RAM). Kubernetes điều chỉnh phù hợp các container vào các node để sử dụng tốt nhất các tài nguyên.
* Kubernetes tự phục hồi khởi động lại các container bị lỗi, thay thế các container, tắt các container không phản hồi với các kiểm tra sức khỏe do người dùng cung cấp và không triển khai chúng cho khách hàng cho đến khi chúng sẵn sàng.
* Kubernetes có cơ chế quản lí bằng configuration và secrets cho phép nhà phát triển lưu trữ và quản lý thông tin nhạy cảm, chẳng hạn như mật khẩu, mã thông báo OAuth và khóa SSH. Nhà phát triển có thể triển khai và cập nhật secrect cũng như cấu hình ứng dụng mà không cần xây dựng lại image container và không để lộ secrect của stack configuration.

1. **Kiến trúc của Kubernetes**
2. **Các thành phần (component) của Kubernetes**

Khi triển khai bằng Kubernetes, nhà phát triển sẽ nhận được một Cluster. Kubernetes Cluster bao gồm một tập hợp các Worker Machine, được gọi là các Node, chạy các ứng dụng được chứa trong Container. Mỗi Cluster đều có ít nhất một Worker Node. (Các) Worker Node làm host chứa các Pod là Component của Application Workload. Control Plane quản lý các Worker node và các Pod trong Cluster. Trong môi trường Production, Control Plane thường chạy trên nhiều máy tính và một Cluster thường chạy nhiều Nod, cung cấp khả năng chịu lỗi và tính sẵn sàng cao.



2.1.1. Các thành phần của Control Plane (Control Plane Components)

Các component của Control Plane đưa ra quyết định chung về Cluster (như lập lịch), cũng như phát hiện và phản hồi các sự kiện của Cluster (ví dụ: khởi động Pod mới khi deployment’s replica field không thỏa yêu cầu nào đó). Các component của Control Plane có thể chạy trên bất kỳ máy nào trong Cluster. Tuy nhiên, để đơn giản, setup scripts thường khởi động tất cả các Control Plane Component trên cùng một máy và không chạy Container của người người dùng trên máy này.

* kube-apiserver: API Server là một Component Kubernetes Control Plane để public ra các API Kubernetes. API Server giống như front end của Kubernetes Control Plane
* etcd: Là một Key-value store nhất quán và sẵn sàng cao được sử dụng làm “xương sống” cho tất cả dữ liệu Cluster. Nếu Kubernetes Cluster sử dụng etcd
* kube-scheduler: Component thuộc Control Plane theo dõi các Pod mới được tạo mà không có Node được gán và chọn một Node cho Pod đó để chạy. Các yếu tố được tính đến để đưa ra quyết định lập lịch bao gồm: yêu cầu tài nguyên của riêng nó và tổng thể hệ thống, các ràng buộc về phần cứng / phần mềm / chính sách, thông số kỹ thuật chung,…
* kube-controller-manager: Component của Control Plane chạy các process của controller. Về mặt lý thuyết, mỗi controller là một process riêng biệt, nhưng để giảm độ phức tạp, tất cả chúng đều được biên dịch thành một tệp nhị phân duy nhất và chạy trong một process duy nhất. Có nhiều loại controller-manager như: Node controller, Job controller, Endpoint controller, Service Account & Token controller,…
* cloud-controller-manager: Component của Control Plane được chứa logic điều khiển dành riêng cho đám mây. Cloud Controller Manager cho phép liên kết cluster của nhà phát triểnvới API của nhà cung cấp dịch vụ 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 nhà phát triển. Tương tự như với kube-controller-manager, cloud-controller-manager kết hợp tiến trình khiển độc lập về mặt logic thành một tệp nhị phân duy nhất mà chạy như một quy trình duy nhất. Nhà phát triển có thể scale theo chiều ngang (chạy nhiều hơn một bản sao) để cải thiện hiệu suất hoặc tăng khả năng chịu lỗi.

2.1.2. Các thành phần của Node (Node Components)

Các Node Component chạy trên mọi Node, duy trì các cluster đang chạy và cung cấp cho chúng Kubernetes Runtime Environment.

* Kubelet:
* Kube-proxy
* Container runtime:

# **Tài liệu tham khảo**

|  |  |
| --- | --- |
| [1] | Yuan Zhu, Sami Demiroluk, Kaan Ozbay, Kun Xie, Hong Yang, Di Sha, "SAVE-T: Safety Analysis Visualization and Evaluation Tool," *Journal of Advanced Transportation,* vol. 2021, 2021. |
| [2] | U. o. C. B. 2. afe Transportation Research and Education Center, "Transportation Injury Mapping System (TIMS)," University of California, Berkeley, 2021. [Online]. |
| [3] | "California Traffic Collision Data from SWITRS," 2021. [Online]. Available: http://iswitrs.chp.ca.gov/ . |