**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT THÀNH PHỐ HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN**

□□□□

Logo

Description automatically generated

**Môn học: ĐIỆN TOÁN ĐÁM MÂY**

**TÌM HIỂU KUBERNETES**

**VÀ VIẾT ỨNG DỤNG DEMO**

**BÁO CÁO ĐỒ ÁN CUỐI KỲ**

**Mã lớp học:**  CLCO332779\_22\_1\_09

**GVHD: TS. Huỳnh Xuân Phụng**

**SVTH MSSV**

**Hoàng Vũ Trường Giang 20110237**

**Trần Trung Hậu 20110642**

**Lê Duy Khiêm 20110661**

Tp. Hồ Chí Minh, tháng 12 năm 202

MỤC LỤC

[LỜI CẢM ƠN 6](#_Toc122315727)

[LỜI CAM KẾT 7](#_Toc122315728)

[MỞ ĐẦU 9](#_Toc122315729)

[Lý do chọn đề tài 9](#_Toc122315730)

[Mục tiêu 9](#_Toc122315731)

[Chức năng dự kiến 9](#_Toc122315732)

[Các Services dự kiến sử dụng 9](#_Toc122315733)

[NỘI DUNG 11](#_Toc122315734)

[Chương 1. CƠ SỞ LÝ THUYẾT 11](#_Toc122315735)

[1.1. Sơ lược về điện toán đám mây (Cloud Computing) 11](#_Toc122315736)

[1.1.1. Khái niệm điện toán đám mây. 11](#_Toc122315737)

[1.1.2. Phân loại mô hình cung cấp dịch vụ 12](#_Toc122315738)

[1.1.2.1. IAAS, PAAS và SAAS 12](#_Toc122315739)

[1.1.2.2. Private, Public and Hybric cloud 13](#_Toc122315740)

[1.2. Sơ lược về AWS 14](#_Toc122315741)

[1.2.1. Giới thiệu về AWS 14](#_Toc122315742)

[1.2.2. Các dịch vụ được hỗ trợ và sử dụng 14](#_Toc122315743)

[1.3. Sơ lược về Kubernetes 15](#_Toc122315744)

[1.3.1. Khái niệm Kubernetes 15](#_Toc122315745)

[1.3.2. Thành phần Kubernetes 15](#_Toc122315746)

[1.3.2.1. Master Node 16](#_Toc122315747)

[1.3.2.2. Worker Node: 17](#_Toc122315748)

[1.3.2.3. Một số thành phần và khái niệm khác. 17](#_Toc122315749)

[1.3.3. Lợi ích của Kubernetes 19](#_Toc122315750)

[1.3.4. Scaling trong Kubernetes 20](#_Toc122315751)

[1.3.4.1. HPA (Horizontal Pod Autoscaler) 20](#_Toc122315752)

[1.3.4.2. VPA (Vertical Pod AutoScaler) 20](#_Toc122315753)

[1.4. Helm - “The best way to find, share, and use software built for Kubernetes”. 21](#_Toc122315754)

[1.5. Sơ lược về Minikube 21](#_Toc122315755)

[1.6. Kubernetes volume là gì ? 21](#_Toc122315756)

[1.6.1. Persistent Volume (pv) 23](#_Toc122315757)

[1.6.2. PersistentVolumeClaim (pvc) 24](#_Toc122315758)

[1.7. Docker. 24](#_Toc122315759)

[1.7.1. Khái niệm Docker. 24](#_Toc122315760)

[1.7.2. Một số khái niệm xung quanh Docker. 24](#_Toc122315761)

[Chương 2. CÀI ĐẶT VÀ TRIỂN KHAI 25](#_Toc122315762)

[2.1. Cài đặt cụm Kubernetes. 25](#_Toc122315763)

[2.1.1. Cách bước tiến hành 25](#_Toc122315764)

[2.1.2. Sơ đồ thiết kế 27](#_Toc122315765)

[2.2. Các giao diện web (file html) 27](#_Toc122315766)

[2.2.1. Giao diện chính 27](#_Toc122315767)

[2.2.2. Giao diện deploy 28](#_Toc122315768)

[2.2.3. Giao diện thông tin 29](#_Toc122315769)

[2.2.4. Giao diện về các Charts 30](#_Toc122315770)

[2.3. Cài đặt chương trình 31](#_Toc122315771)

[2.4. Demo 32](#_Toc122315772)

[KẾT LUẬN 33](#_Toc122315773)

[Kết quả thu được 33](#_Toc122315774)

[Ưu, nhược điểm 33](#_Toc122315775)

[Ưu điểm 33](#_Toc122315776)

[Nhược điểm 33](#_Toc122315777)

[Hướng phát triển 34](#_Toc122315778)

[TÀI LIỆU THAM KHẢO 35](#_Toc122315779)

MỤC LỤC HÌNH ẢNH

[Hình 1. Cloud Computing. 12](#_Toc122315710)

[Hình 2. Ứng dụng của Cloud Computing theo IAAS, PAAS và SAAS. 14](#_Toc122315711)

[Hình 3. AWS và các dịch vụ phổ biến. 15](#_Toc122315712)

[Hình 4. Logo K8s 16](#_Toc122315713)

[Hình 5. Các thành phần K8s 17](#_Toc122315714)

[Hình 6. So sánh Master và Worker Node 18](#_Toc122315715)

[Hình 7. Lợi ích K8s, 21](#_Toc122315716)

[Hình 8. Cách thức hoạt động của Kubernetes Volume. 23](#_Toc122315717)

[Hình 9. Một số khái niệm liên quan đến Docker, 26](#_Toc122315718)

[Hình 10. Sơ đồ thiết kế cụm Kubernetes. 28](#_Toc122315719)

[Hình 11. Giao diện chính. 29](#_Toc122315720)

[Hình 12. Giao diện deploy. 30](#_Toc122315721)

[Hình 13. Giao diện thông tin. 30](#_Toc122315722)

[Hình 14. Giao diện về các Charts. 31](#_Toc122315723)

# LỜI CẢM ƠN

Lời nói đầu tiên, nhóm thức hiện đồ án xin phép chân thành cảm ơn thầy Huỳnh Xuân Phụng. Một đề tài rất hấp dẫn và đa dạng nhận được từ thầy đã giúp chúng em phát triển và lĩnh hiểu được nhiều kiến thức bổ ích mà đồ án mang lại, giúp cho con đường sau này của chúng em trở nên thuận tiện hơn.

Một lần nữa, xin cảm ơn thầy vì sự quan tâm và giúp đỡ từ thầy, cũng như là sự phản hồi tận tình để giúp đồ án của chúng em trở nên hoàn thiện hơn như bây giờ.

Vì khả năng còn hạn chế nên trong quá trình thực hiện đồ án sẽ còn nhiều thiếu sót không thể tránh khỏi và các dự định chưa làm được. Rất mong nhận được sự phản hồi từ để đồ án này cũng như là đồ án khác trong tương lai của chúng em sẽ chỉnh chu hơn.

# LỜI CAM KẾT

Chúng em xin cam đoan đồ án môn học Cloud Computing “TÌM HIỂU KUBERNETES VÀ VIẾT ỨNG DỤNG DEMO” là công trình nghiên cứu của nhóm 1. Những phần sử dụng tài liệu tham khảo trong đồ án đã được nêu rõ trong phần TÀI LIỆU THAM KHẢO. Các số liệu, kết quả trình bày trong đồ án là hoàn toàn trung thực, nếu sai sót và gian lận, chúng em xin chịu hoàn toàn trách nhiệm và chịu mọi kỷ luật của giáo viên bộ môn và nhà trường đề ra.

TP.HCM, ngày 18 tháng 12 năm 2022

|  |  |  |
| --- | --- | --- |
| Sinh viên 1  (Ký tên)  Hoàng Vũ Trường Giang | Sinh viên 2  (Ký tên)  Trần Trung Hậu | Sinh viên 3  (Ký tên)  Lê Duy Khiêm |

DANH MỤC TỪ VIẾT TẮT

|  |  |
| --- | --- |
| K8s | Kubernetes: chủ đề chính của đồ án. Được trình bày ở các phần sau. |
| AWS | Amazon Web Services là một công ty con của Amazon cung cấp các nền tảng điện toán đám mây |

# MỞ ĐẦU

## Lý do chọn đề tài

Những năm vừa qua, công nghệ ảo hóa container nói chung và Docker phát triển ngày càng mạnh mẽ. Một ứng dụng lớn và phức tạo có thể bao gồm từ vài chục cho đến hàng trăm hoặc thậm chí hàng ngàn container. Công việc quản lý, điều phối container đòi hỏi các công cụ, dịch vụ để xử lý công việc này một cách nhanh chóng và thuận tiện nhất. Vì vậy, Container orchestration đã bùng nổ và phát triển mạnh mẽ. Trong đó Kubernetes nổi tiếng và được sử dụng nhiều nhất.

Thế nhưng, khi chuyển qua từ những phương pháp deploy truyền thống sang Kubernetes thì sẽ trở thành thách thức lớn và khó khăn khi phải tốn một khoảng thời gian để đào tạo và tập huấn. Từ những điều đó, nhóm quyết định chọn đề tài và tạo ra một trang website tương tác với Kubernetes để có thể triển khai các ứng dụng.

## Mục tiêu

Với yêu cầu là tìm hiểu ứng dụng về Kubernetes thì nhóm sẽ thực hiện đồ án này với mục đích:

* Tìm hiểu rõ về Kubernetes
* Tìm hiểu và sử dụng các công cụ có thể hỗ trợ Kubernetes
* Tìm hiểu các dịch vụ của AWS cho việc tạo, sử dụng và phát triển Kubernetes
* Áp dụng Kubernetes để xây dựng một ứng dụng demo, cụ thể là một website có tương tác với Kubernetes để có thể hỗ trợ việc triển khai các ứng dụng

## Chức năng dự kiến

Nhóm dự kiến sẽ xây dựng một website hỗ trợ triển khai các helm chart trên cụm máy ảo của EC2. Helm chart dùng để đóng gói những thành phần, những tài nguyên cần thiết trong quá trình xây dựng một ứng dụng chạy trên nền tảng Kubernetes. Website này có các chức năng cơ bản là tạo chart, xóa, sửa, xem và tăng số lượng bản sao thông tin các ứng dụng.

## Các Services dự kiến sử dụng

Dịch vụ sẽ sử dụng: Amazon Elastic Compute Cloud (EC2)

EC2 là dịch vụ IaaS trong nhóm dịch vụ Compute, về cơ bản chúng ta có sử dụng các máy chủ ảo của AWS và chúng ta có thể cấu hình, chạy các dịch vụ của chúng ta trên đó. Ngoài ra EC2 cũng cung cấp các cơ chế backup hoặc restore một cách nhanh chóng giúp chúng ta restore hoặc scale hệ thống của mình một cách cực kỳ nhanh chóng và dễ dàng

Ngoài ra nhóm chúng em còn sử dụng một số công cụ khác để hỗ trợ như Docker.

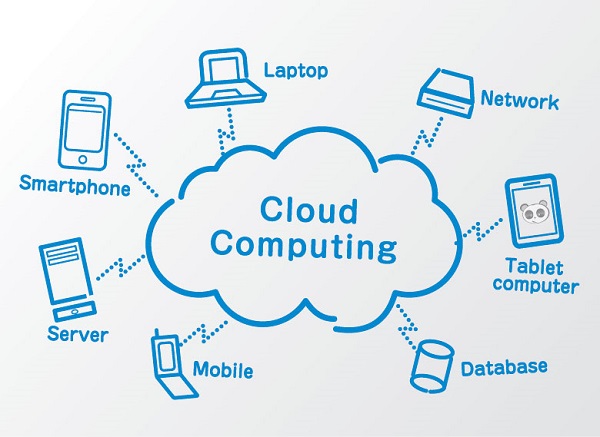
# NỘI DUNG

## CƠ SỞ LÝ THUYẾT

### Sơ lược về điện toán đám mây (Cloud Computing)

#### Khái niệm điện toán đám mây.

Điện toán đám mây là một khái niệm cho phép tài nguyên CNTT được phân phối theo nhu cầu của doanh nghiệp, tổ chức hoặc cá nhân thông qua Internet. Thay vì phải tốn nhiều tài nguyên, tiền bạc cho việc mua và quản lý tài nguyên CNTT, chúng ta có thể sử dụng các dịch vụ công nghệ, điện toán, lưu trữ và cơ sở dữ liệu, khi cần thiết, từ nhà cung cấp dịch vụ đám mây như Amazon, Google… Tất cả chi phí phải chi trả chỉ từ những gì chúng ta sử dụng (pay-as-you-go).



Hình 1. Cloud Computing.

Sáu lợi ích của việc sử dụng điện toán đám mây:

* Chi phí vốn thương mại: Số tiền phải chi trả cho lượng tài nguyên CNTT tiêu
* thụ sẽ thấp hơn số tiền mà một công ty sử dụng để mua, nâng cấp và bảo trì các
* tài sản vật chất CNTT.
* Hưởng lợi từ quy mô kinh tế lớn: chi phí biến đổi thấp hơn khi công ty lớn
* mạnh nhanh chóng. Điều này còn xảy ra khi dịch vụ AWS càng lớn, số tiền phải
* bỏ ra càng nhỏ.
* Ngừng việc dự đoán công suất: các dự đoán về nhu cầu cơ sở hạ tầng, tài nguyên
* nhàn rỗi hoặc đối phó với tài nguyên hạn chế sẽ được loại bỏ.
* Tăng tốc độ và sự nhanh chóng: Trong môi trường điện toán đám mây, thời
* gian để cấp phát tài nguyên cho các nhà phát triển chỉ trong vài phút.
* Ngừng tốn chi phí vào việc vận hành và duy trì các trung tâm dữ liệu: Điện
* toán đám mây cho phép tập trung vào khách hàng thay vì phải gánh nặng chi phí
* các máy chủ.
* Phát triển hệ thống toàn cầu trong thời gian ngắn: Các ứng dụng, tài nguyên
* có thể cung cấp độ trễ thấp hơn và trải nghiệm tốt hơn cho khách hàng một cách
* đơn giản và với chi phí tối thiểu.

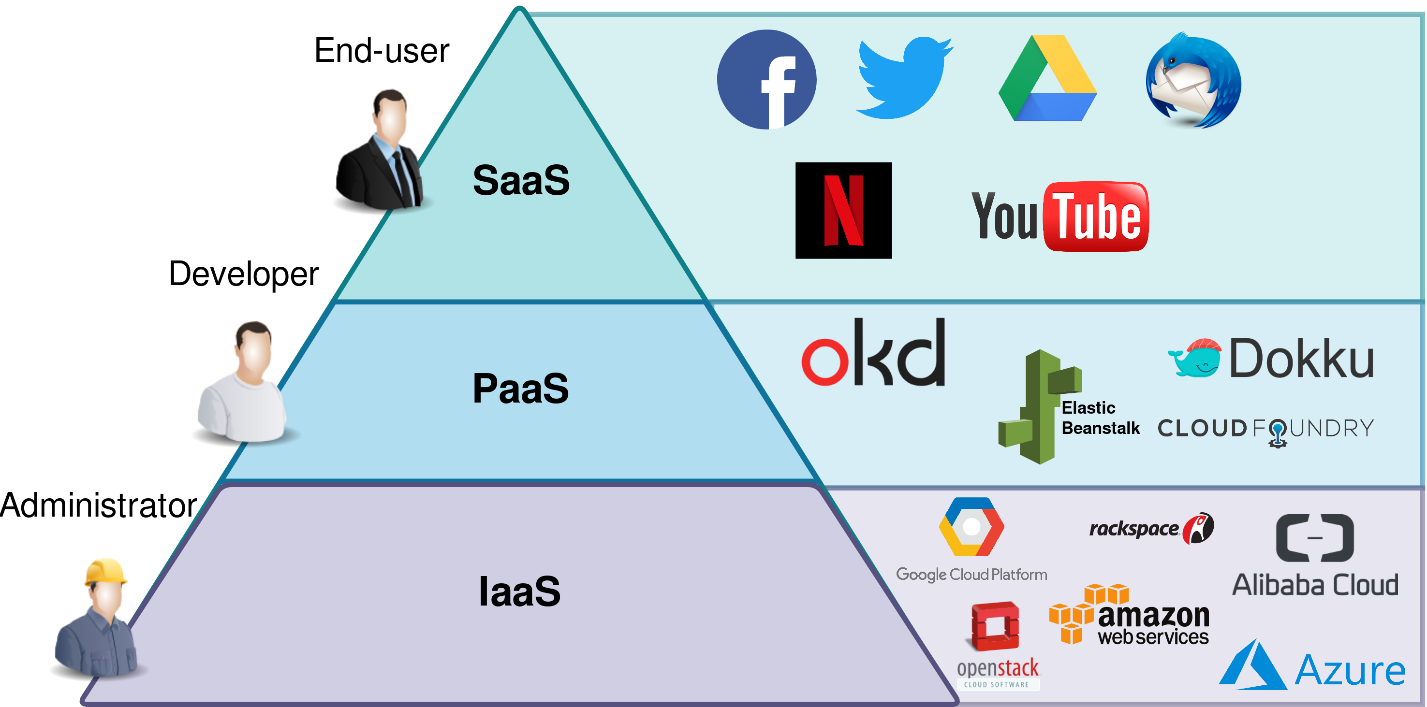
#### Phân loại mô hình cung cấp dịch vụ

##### IAAS, PAAS và SAAS

IAAS: Infrastructure as a Service (IAAS) là phương tiện cung cấp cơ sở hạ tầng máy tính dưới dạng dịch vụ theo yêu cầu. Nó là một trong ba hệ điều hành mạng lưu trữ mô hình dịch vụ đám mây cơ bản. Trong trường hợp người dùng mua máy chủ, không gian trung tâm dữ liệu phần mềm hoặc thiết bị mạng và thuê các tài nguyên đó như một dịch vụ thuê ngoài hoàn toàn có thể yêu cầu mô hình. Nó cho phép mở rộng quy mô động và các tài nguyên được phân phối như một dịch vụ. Nó thường bao gồm nhiều người dùng trên một phần cứng.

PAAS: Platform as a Service (PAAS) là một mô hình phân phối đám mây cho các ứng dụng bao gồm các dịch vụ do bên thứ ba quản lý. Nó cung cấp khả năng mở rộng ứng dụng của bạn một cách đàn hồi, cho phép các nhà phát triển xây dựng các ứng dụng và dịch vụ qua internet và các mô hình triển khai bao gồm công khai, riêng tư và kết hợp.

SAAS: Software as a Service (SAAS) cho phép người dùng chạy các ứng dụng trực tuyến hiện có và nó là một phần mềm mẫu được triển khai như một dịch vụ lưu trữ và được truy cập qua Đầu ra Văn bản được mô tả lại / viết lại trên internet hoặc mô hình phân phối phần mềm trong đó phần mềm và dữ liệu liên quan của nó được lưu trữ tập trung và được truy cập bằng ứng dụng khách của họ, thường là một trình duyệt trực tuyến trên web. Các dịch vụ SAAS được sử dụng để phát triển và triển khai các ứng dụng hiện đại.



Hình 2. Ứng dụng của Cloud Computing theo IAAS, PAAS và SAAS.

##### Private, Public and Hybric cloud

Private cloud thuộc về một tổ chức. Do đó, tổ chức vận hành đám mây đó và các dịch vụ chỉ khả dụng trong tổ chức. Một đám mây riêng cung cấp nhiều lợi thế. Vì khả năng truy cập được giới hạn cho tổ chức và công chúng không thể có quyền truy cập vào các tài nguyên, một đám mây riêng cung cấp bảo mật hơn cho dữ liệu và tài nguyên.

Public cloud là một cách tiếp cận phát triển ứng dụng thay thế cho các kiến trúc CNTT tại chỗ truyền thống. Trong mô hình điện toán đám mây công cộng cơ bản, nhà cung cấp bên thứ ba lưu trữ các tài nguyên CNTT theo yêu cầu, có thể mở rộng và cung cấp chúng cho người dùng qua kết nối mạng, qua internet công cộng hoặc mạng chuyên dụng.

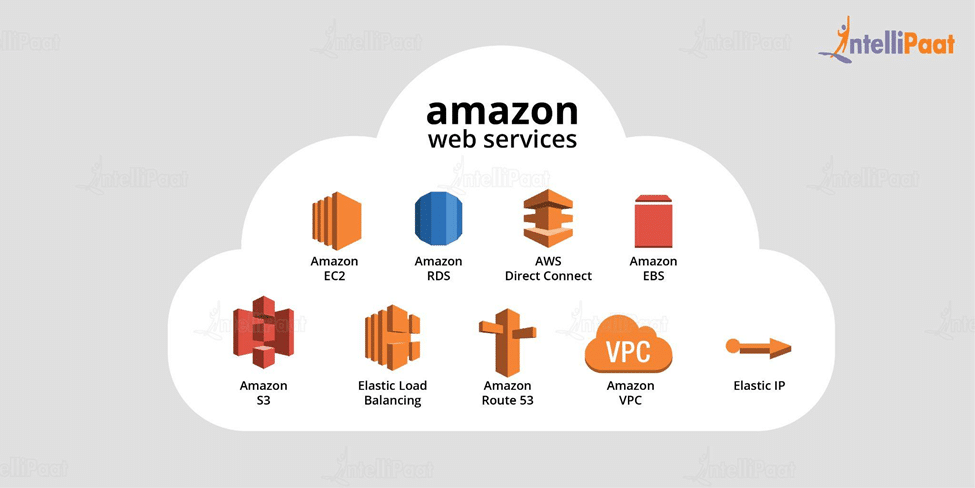
Đám mây lai là sự kết hợp của đám mây công cộng và đám mây riêng. Nó cung cấp các tính năng của cả hai. Đám mây riêng thực hiện các hoạt động quan trọng trong khi đám mây công cộng thực hiện các hoạt động không quan trọng. Nó yêu cầu chi phí tối thiểu hơn so với một đám mây riêng. Hơn nữa, đám mây riêng của một đám mây lai cung cấp bảo mật cho dữ liệu và tài nguyên.

### Sơ lược về AWS

#### Giới thiệu về AWS

Amazon Web Services (AWS) là một nền tảng đám mây an toàn cung cấp rất nhiều các sản phẩm dựa trên cơ sở hạ tầng đám mây trên toàn cầu do Amazon cung cấp, bao gồm hỗn hợp cơ sở hạ tầng dưới dạng dịch vụ (IAAS), nền tảng dưới dạng dịch vụ (PAAS) và phần mềm đóng gói dưới dạng dịch vụ (SAAS).

Để có thể tương tác với AWS: AWS Management Console - tương tác trực quan đơn giản, AWS Commàn Line Interface (AWS CLI) và Software Development Kits (SDKs) - phục vụ cho các lập trình viên.



Hình 3. AWS và các dịch vụ phổ biến.

#### Các dịch vụ được hỗ trợ và sử dụng

Các dịch vụ trên AWS vô cùng đa dạng (khoảng hơn 200 dịch vụ). Trong đó, các dịch vụ cơ bản như: Compute, Storage, Database, Networking, Content delivery, Các công cụ cho Developer trên AWS, các dịch Management Service, Analytics, IoT, Machine learning… Bên cạnh đó còn nhiều dịch vụ quản lý và sử dụng các tài nguyên khác trên AWS, các dịch vụ này hoàn toàn miễn phí. Không những thế, AWS còn cung cấp nhiều gói dịch vụ miễn phí cho các nhà phát triển nhỏ.

Trong phạm vi nghiên cứu của đề tài, chúng em tập chung sử dụng dịch vụ EC2.

**EC2** là dịch vụ IaaS trong nhóm dịch vụ Compute, về cơ bản chúng ta có sử dụng các máy chủ ảo của AWS và chúng ta có thể cấu hình, chạy các dịch vụ của chúng ta trên đó. Ngoài ra EC2 cũng cung cấp các cơ chế backup hoặc restore một cách nhanh chóng giúp chúng ta restore hoặc scale hệ thống của mình một cách cựckỳ nhanh chóng và dễ dàng.

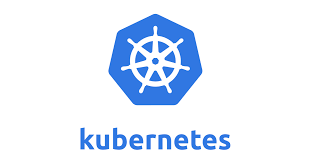
### Sơ lược về Kubernetes

#### Khái niệm Kubernetes

Kubernetes là một nền tảng nguồn mở, khả chuyển, có thể mở rộng để quản lý các ứng dụng được đóng gói và các service, giúp thuận lợi trong việc cấu hình và tự động hóa việc triển khai ứng dụng. Kubernetes là một hệ sinh thái lớn và phát triển nhanh chóng. Các dịch vụ, sự hỗ trợ và công cụ có sẵn rộng rãi.

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.



Hình 4. Logo K8s

#### Thành phần Kubernetes

Có 2 khái niệm chính trong kubernetes: Cluster và Node.

Node là một máy chủ (vật lý hoặc máy ảo). Các Node sẽ được phân chia và phân bổ các nhiệm vụ riêng biệt trong một hệ thống. Các Node còn bao gồm các tài nguyên như là compute, RAM, storage và network. Có 2 loại node: master node và worker node.

Cluster là một cụm bao gồm nhiều node mà trong đó các node sẽ cung cấp. Một hệ thống ở quy mô lớn có thể bao gồm nhiều Cluster. Các cụm là tập hợp sự tính toán, lưu trữ của nhiều Node lại giúp cho hệ thống hoạt động trơn tru.

Timeline

Description automatically generated

Hình 5. Các thành phần K8s

##### 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, ưng dụng được đặt vào Worker nào để chạy
* Controller Manager: Thành phần đảm nhiệm phần quản lý các Worker, kiểm tra các Worker sống hay chết, đảm nhận việc nhân bản ứng dụng…
* Kube Controller Manager: Là một tập hợp các controller khác nhau để thuận lợi cho việc theo dõi các cập nhật trạng thái của Kubernetes Cluster thông qua API và nhanh chóng thực hiện các thay đổi đối với Cluster sao cho phù hợp với hệ thống.
* Etcd: Đây là cơ sở dữ liệu của Kubernetes, tất cả các thông tin của Kubernetes được lưu trữ cố định vào đây.

##### Worker Node:

Là server chạy ứng dụng trên đó. Bao gồm 3 thành phần chính:

* Container runtime: Là thành phần giúp chạy các ứng dụng dưới dạng Container. Thông thường người ta sử dụng Docker.
* Kubelet: đây là thành phần giao tiếp với Kubernetes API Server, và cũng quản lý các container
* Kubernetes Service Proxy: Thành phần này đảm nhận việc phân tải giữa các ứng dụng

Graphical user interface

Description automatically generated

Hình 6. So sánh Master và Worker Node

##### Một số thành phần và khái niệm khác.

**PODS**

Một nhóm gồm một hoặc nhiều containers được triển khai cho single node. Tất cả các containers trong một pod dùng chung một IP address, IPC, hostname, và những nguồn khác. Pods tách network và storage ra khỏi container bên dưới. Điều này cho phép bạn di chuyển các containers xung quanh cluster dễ dàng hơn.

**SERVICES**

Là phần mạng (network) của Kubernetes giúp cho các Pod gọi nhau ổn định hơn, hoặc để Load Balancing giữa nhiều bản sao của Pod, và có thể dùng để dẫn traffic từ người dùng vào ứng dụng (Pod), giúp người dùng có thể sử dụng được ứng dụng.

**LABELS**

Labels cung cấp metadata để identity cho các object trong Kubernetes. Labels cho phép tổ chức và group các object lại với nhau trong Cluster. Một object có thể có nhiều Label và mỗi Label có thể được gán cho nhiều object khác nhau có thể xem đây là mối quan hệ MxN.

**ANNOTATIONS**

Annotations khá giống với Label. Nhưng Annotations thì chỉ cung cấp metadata để máy tính hoặc các ứng dụng sử dụng chứ không phải con người.

Ví dụ: Chúng ta có thể sử dụng Annotations để bổ sung các thông tin như là timestamp, Git branch, Release IDs, ...

**REPLICASETS**

Replica Set đảm bảo rằng application sẽ luôn chạy đúng loại và số lượng Pod trong cluster.

Replica Set được thiết kế cho stateless application (ví dụ như Web Server).

**CONFIGMAPS**

Một ConfigMap là một Kubernetes API object có thể được sử dụng để lưu trữ dữ liệu dưới dạng các cặp key-value. Kubernetes có thể sử dụng ConfigMaps đã tạo dưới dạng:

* Configuration file
* Environment variable
* Command-line argument

ConfigMaps cung cấp khả năng làm cho các ứng dụng linh động bằng cách tách các cấu hình dành riêng cho môi trường khỏi các containers.

**SECRETS**

- Secrets tương tự như ConfigMap nhưng được sử dụng để cung cấp thông tin cho container.

Ví dụ: API token, MySQL DB Password, ...

- Sử dụng Secrets cho phép bạn tạo container mà không cần đóng gói dữ liệu trong

container.

- Thông qua Manifest file thì Configmap và Secrets có thể được truyền vào Pod và

Kubernetes API

**NAMESPACES**

Trong Kubernetes, Namespace cung cấp một cơ chế để cô lập các nhóm tài

nguyên trong một cụm duy nhất. Tên của các tài nguyên cần phải là duy nhất

trong một Namespace, nhưng không phải giữa các Namespace.

- Phạm vi Namespace-based chỉ áp dụng cho các đối tượng Namespace (ví dụ: Services, Deployments, …) và không áp dụng cho các đối tượng toàn cụm (ví dụ: PresentlyVolumes, StorageClass, Nodes, ….)

- Các namespaces sẽ được deploy tất cả các object trong Kubernetes vào.

- Việc quản lý các object trong Cluster thì Kubernetes sử dụng Namespace để quản lý.

#### Lợi ích của Kubernetes

- Phát triển application nhanh chóng nhưng vẫn duy trì sự ổn định.

- Việc scale dễ dàng hơn rất nhiều

- Tự động khôi phục nếu có sự cố.

- Trên cùng một máy thì nhiều application có thể chạy mà không ảnh hưởng đến nhau.

- Việc phân phối các ứng dụng trên toàn cụm đều sẽ tự động hóa.

Graphical user interface

Description automatically generated

Hình 7. Lợi ích K8s,

#### Scaling trong Kubernetes

Auto-scaling là cách thức tự động tăng hoặc giảm tài nguyên CNTT mà được cấp cho ứng dụng tại bất kỳ thời điểm nào. Vì thế, giúp tạo ra những hệ thống cloud có khả năng mở rộng 24/24 nhằm đáp ứng nhu cầu sử dụng.

ReplicaSet trong phần Controller - nó đảm bảo ổn định các pod của một ứng dụng khi đang chạy.

##### HPA (Horizontal Pod Autoscaler)

Horizontal pod autoscaling là cách ta tăng giá trị replicas ở trong các scalable resource (Deployment, ReplicaSet, ReplicationController, hoặc StatefulSet) để scale số lượng Pod. Công việc này được thực hiện bởi Horizontal controller khi ta tạo một HorizontalPodAutoscaler (HPA) resource. Horizontal controller sẽ thường xuyên kiểm tra metric của Pod, và tính toán số lượng pod replicas phù hợp dựa vào metric kiểm tra của Pod hiện tại với giá trị metric mà ta đã chỉ định ở trong HPA resource, sau đó sẽ thay đổi trường replicas của các scalable resource (Deployment, ReplicaSet, ReplicationController, or StatefulSet) nếu nó thấy cần thiết.

##### VPA (Vertical Pod AutoScaler)

VPA là một quá trình tự động scale theo yêu cầu CPU và bộ nhớ tự động và điều chỉnh giới hạn dựa trên các phép đo sử dụng tài nguyên. Kubernetes Vertical Pod Autoscaler tự động điều chỉnh cpu và bộ nhớ định trước để giúp điều chỉnh ‘'đúng kích thước’ các ứng dụng. Điều chỉnh này có thể cải thiện việc sử dụng tài nguyên cụm và giải phóng CPU và bộ nhớ cho các Pod khác

### Helm - “The best way to find, share, and use software built for Kubernetes”.

Helm là một trình quản lý gói ứng dụng cho Kubernetes, điều phối tải xuống, cài đặt và triển khai ứng dụng.

Helm chart là cách để bạn xác định một ứng dụng bao gồm các tài nguyên nào trong Kubernetes.

Lợi ích:

* Deployment đơn giản hơn mang tính chất lặp lại với chỉ vài câu lệnh ngắn
* Quản lý sự phụ thuộc của ứng dụng với các version cụ thể
* Thực hiện nhiều deployment với các môi trường khác nhau như: test, staging, production ...
* Thực thi các jobs liên quan đến chạy ứng dụng trước khi deployment
* Dễ dàng update rollback và test deployment khi có vấn đề xảy ra hay muốn cập nhật phiên bản mới (zero downtime server)

### Sơ lược về Minikube

- Minikube là một công cụ cho phép chạy Kubernetes cục bộ.

- Minikube chạy cụm Kubernetes một node trên máy tính cá nhân (bao gồm PC Windows, macOS và Linux) để dùng thử Kubernetes hoặc cho công việc phát triển.

- Là công cụ tốt nhất để phát triển kubernetes, nó hỗ trợ tất cả các tính năng phù hợp.

- Là công cụ hữu ích cho người mới bắt đầu sử dụng kubernetes.

### Kubernetes volume là gì ?

Kubernetes volume là một thư mục lưu trữ dữ liệu có thể truy cập được vào các container trong một Pod nhất định trong cụm Kubernetes. Khi các container gặp phải sự cố phải restart thì dữ liệu log hoặc database trong các Pod này sẽ bị mất, vì vậy cần một nơi lưu trữ lại những dữ liệu này. Có thể coi Volume là bộ nhớ vật lý hoặc những remote storage sử dụng chung của các Pod. Volume cung cấp một cơ chế plug-in để kết nối các container có dạng empheral với các kho dữ liệu liên tục ở nơi khác.

Các loại Volumes mà Kubernetes hỗ trợ: hostPath, awsElasticBlockStore, gcePersistentDisk, azureDisk, cinder, configMap, cephfs, rbd, glusterfs.

Trong đồ án lần này, **Hostpath** là loại Kubernetes Volumes được áp dụng.

Một hostPath volume sẽ mount 1 file hoặc 1 thư mục từ filesystem trên node host vào trong pod. Đây không phải là điều mà hầu hết các pod sẽ cần nhưng nó cung cấp 1 lối thoát mạnh mẽ cho một số ứng dụng.

Một số trường hợp sử dụng của hostPath:

* Chạy 1 container cần truy cập nội bộ Docker → ta sử dụng hostPath đến /var/lib/docker
* Chạy cAdvisor trong 1 container → sử dụng hostPath đến /sys
* Cho phép 1 pod chỉ định liệu 1 hostPath cụ thể có được tồn tại trước khi Pod chạy hay không, liệu nó có được tạo hay không và nó nên tồn tại như thế nào.

Diagram

Description automatically generated

Hình 8. Cách thức hoạt động của Kubernetes Volume.

#### Persistent Volume (pv)

PersistentVolume (pv) là một phần không gian lưu trữ dữ liệu trong cluster, các Persistent Volume giống với Volume bình thường tuy nhiên nó tồn tại độc lập với POD (pod bị xóa PV vẫn tồn tại), có nhiều loại Persistent Volume có thể triển khai như NFS, Glusterfs ... (xem tại Các kiểu PersistentVolume).

Kubernetes hỗ trợ nhiều persistent volume khác nhau như:

- NFS: AWS Block Storage, Google Cloud Storage.

- iSCSI: FreeNAS-iscsi-provisioner.

Chu kỳ chính của Volume:

- Provisioning: statically or dynamically.

- Binding: đảm bảo rằng PV đáp ứng nhu cầu của người dùng mà không lãng phí tài nguyên khối lượng

- Using: Các pod sử dụng các volume.

Cấu hình PV cũng như những component khác được tạo bằng cách sử dụng YAML file thông qua các keyword:

- Capacity: PV sẽ có dung lượng lưu trữ cụ thể bằng cách sử dụng thuộc tính capacity của PV.

- Volume mode: Có 2 loại Filesystem và Block.

- Access Modes bao gồm: ReadWriteOnce, ReadWriteOnce, ReadOnlyMany, ReadWriteMany, ReadWriteOncePod.

- Class: được định nghĩa bởi storageClassName đại diện cho thuộc tính, StorageClass.

- Reclaim Policy: Các chính sách để thu hồi lại các PV bao gồm (Retain, Recycle, Delete)

- Mount Options: Người quản trị Kubernetes có thể chỉ định các Mount Option bổ sung Persistent Volume khác nhau được cài đặt dưới dạng plugin như glusterfs, nfs, csi, ...

#### PersistentVolumeClaim (pvc)

PersistentVolumeClaim (pvc) là yêu cầu sử dụng không gian lưu trữ (sử dụng PV). Hình dung PV giống như Node, PVC giống như POD. POD chạy nó sử dụng các tài nguyên của NODE, PVC hoạt động nó sử dụng tài nguyên của PV.

Tổng quát lại quá trình sử dụng storage của các Pod.

Để sử dụng các PVC trong các pod, các developer thường thông qua các YAML file. Trong file YAML này, các thuộc tính cũng thông qua một số keyword như trong PV.

### Docker.

#### Khái niệm Docker.

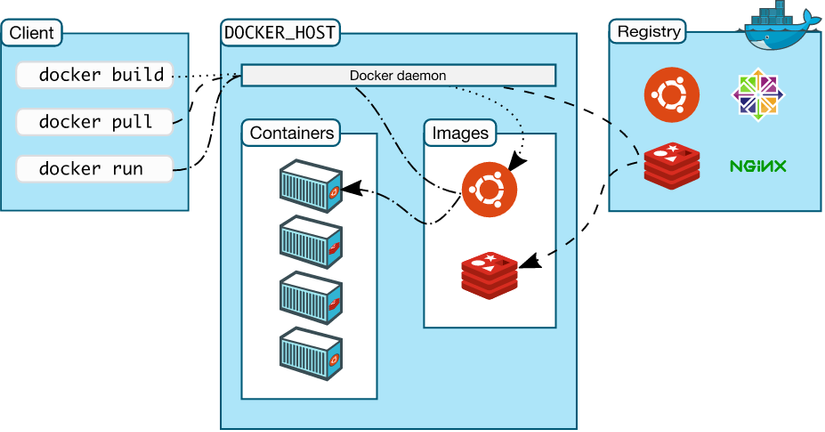
Docker là một nền tảng để cung cấp cách để building, deploying và running ứng dụng dễ dàng hơn bằng cách sử dụng các containers (trên nền tảng ảo hóa). Ban đầu viết bằng Python, hiện tại đã chuyển sang Golang..

Các containers cho phép lập trình viên đóng gói một ứng dụng với tất cả các phần cần thiết, chẳng hạn như thư viện và các phụ thuộc khác, và gói tất cả ra dưới dạng một package.

Bằng cách đó, nhờ vào container, ứng dụng sẽ chạy trên mọi máy Linux khác bất kể mọi cài đặt tùy chỉnh mà máy có thể có khác với máy được sử dụng để viết code.

#### Một số khái niệm xung quanh Docker.

* **Docker Engine** : là thành phần chính của Docker, như một công cụ để đóng gói ứng dụng
* **Docker Hub** : là một “github for docker images”. Trên DockerHub có hàng ngàn public images được tạo bởi cộng đồng cho phép bạn dễ dàng tìm thấy những image mà bạn cần. Và chỉ cần pull về và sử dụng với một số config mà bạn mong muốn.
* **Images**: là một khuôn mẫu để tạo một container. Thường thì image sẽ dựa trên 1 image có sẵn với những tùy chỉnh thêm. Ví dụ bạn build 1 image dựa trên image Centos mẫu có sẵn để chạy Nginx và những tùy chỉnh, cấu hình để ứng dụng web của bạn có thể chạy được. Bạn có thể tự build một image riêng cho mình hoặc sử dụng những image được chia sẽ từ cộng đồng Docker Hub. Một image sẽ được build dựa trên những chỉ dẫn của Dockerfile.
* **Container**: là một instance của một image. Bạn có thể create, start, stop, move or delete container dựa trên Docker API hoặc Docker CLI.
* **Docker Client:** là một công cụ giúp người dùng giao tiếp với Docker host.
* **Docker Daemon**: lắng nghe các yêu cầu từ Docker Client để quản lý các đối tượng như Container, Image, Network và Volumes thông qua REST API. Các Docker Daemon cũng giao tiếp với nhau để quản lý các Docker Service.
* **Dockerfile**: là một tập tin bao gồm các chỉ dẫn để build một image .
* **Volumes**: là phần dữ liệu được tạo ra khi container được khởi tạo.



Hình 9. Một số khái niệm liên quan đến Docker,

## CÀI ĐẶT VÀ TRIỂN KHAI

### Cài đặt cụm Kubernetes.

#### Cách bước tiến hành

Bước 1:

- Tại mỗi server, tiến hành cài đặt các docker

- Thiết lập các Repository

- Cài đặt các docker engine

● Bước 2:

- Tắt Swap và Bật Kernel modules

● Bước 3:

- Trên mỗi server, cài đặt Kubernetes

+ Update và install các package cần thiết

+ Download the Google Cloud public signing key

+ Thêm the Kubernetes apt repository

+ Update lại và install kubelet, kubeadm and kubectl

● Bước 4:

- Trên máy master, khởi tạo một master nodes

- Sau khi câu lệnh kết thúc command, copy kubeadm để join các máy cluster

● Bước 5:

- Thiết lập tệp cấu hình kubernetes để sử dụng chung

● Bước 6:

- Áp dụng một plugin mạng chung. Trong trường hợp này, Flannel

kubectl apply -f

<https://raw.githubusercontent.com/coreos/flannel/2140ac876ef134e0ed5af15c65e414cf26827915/Documentation/kube-flannel.yml>

Text

Description automatically generated

* Bước 7:

- Tiến hành join các cluster vô cụm Kubernetes vừa tạo

● Bước 8:

- Kiểm tra xem mọi thứ đã ổn chưa

kubectl get nodes

#### Sơ đồ thiết kế

Graphical user interface, diagram

Description automatically generated

Hình 10. Sơ đồ thiết kế cụm Kubernetes.

### Các giao diện web (file html)

#### Giao diện chính

<body>

<h1 class="txt\_title">Xin chào mừng bạn đến với kubernetes</h1>

<img src="{{url\_for('static',filename = 'img/index.png')}}" alt="" class="bg\_image">

<div class="btn\_group">

<form action="/deploy\_apps" method="POST">

<input type="submit" name="submit\_button" value="Start cluster" class="btn\_action"></input>

<input type="submit" name="submit\_button" value="Stop cluster" class="btn\_action"></input>

<input type="submit" name="submit\_button" value="Delete cluster" class="btn\_action"></input>

</form>

</div>

</body>

Diagram

Description automatically generated

Hình 11. Giao diện chính.

#### Giao diện deploy

<body class="bg\_deploy">

  <h1 class="txt\_title">Triển khai ứng dụng trên minikube</h1>

    <h3>Sử dụng trình quản lý gói helm để deploy ứng dụng</h3>

    <div class="container">

        <br>

        <form action="/show\_all\_deploy" method="POST">

            <input type="submit" name="submit\_button" value="Show all applications" class="btn\_action"></input>

            <input type="submit" name="submit\_button" value="Delete all applications" class="btn\_action" id ="btn\_del"></input>

        </form>

    </div>

    <div class="wrapper">

        <!--Make sure the form has the autocomplete function switched off:-->

        <form class="search" autocomplete="off" action="/deploy\_option" method="post">

            <div class="autocomplete" >

            <input id="myInput" type="text" name="application" placeholder=" Search applications..." >

            </div>

            <button>Deploy</button>

        </form>

    </div>

    <script>

        autocomplete(app);

    </script>

</body>

Graphical user interface, text, website

Description automatically generated

Hình 12. Giao diện deploy.

#### Giao diện thông tin

<body class="bg\_deploy">

  <h1 class="txt\_title">Thông tin applications</h1>

    <h3>Thay đổi replicas sử dụng kubectl</h3>

    <div class="wrapper">

        <!--Make sure the form has the autocomplete function switched off:-->

    {% set url = '/scaling?app=' + app|join('') %}

    <form class="search" autocomplete="on" action={{url}} method="post">

            <input type="number" name="replicas" value={{number\_rs}}>

            <button>Change</button>

        </form>

    </div>

    <center><h3>Thông tin chi tiết: </h3>

    <div >

        <textarea name="replicas" rows="20" cols="100" readonly>{{mydata}}</textarea>

    </div>

    </center>

</body>

Graphical user interface

Description automatically generated

Hình 13. Giao diện thông tin.

#### Giao diện về các Charts

<body class="bg\_redis">

    <h1 class="txt\_title">DEPLOYMENT</h1>

    <table border="1" class="table1">

        <tr>

            <th colspan="1">Name</th>

            <th colspan="1">Namespace</th>

            <th colspan="1">Revision</th>

            <th colspan="1">Updated</th>

            <th colspan="1">Status</th>

            <th colspan="1">Chart</th>

            <th colspan="1">App Version</th>

             <th colspan="1">Info</th>

            <th colspan="1">Action</th>

        </tr>

        <tr>

            {%  for i in range(0,mydata|length) %}

                <tr>

                {% set url\_del ='/delete\_deploy?app=' + mydata[i][0] %}

                {% set url = '/deploy\_modify?app=' + mydata[i][0] %}

                {% for column in mydata[i] %}

                    <td>{{column}}</td>

                {% endfor %}

        <td> <form action={{url}} method="post"><button class="btn\_ac">

            <i class="fa fa-tasks"></i></button></form> </td>

    <td>     <form action={{url\_del}} method="post"><button class="btn\_red">

        i class="fa fa-trash"></i></button></form>

            </td>

            </tr>

        {% endfor %}

        </tr>

    </table>

</body>

A picture containing table

Description automatically generated

Hình 14. Giao diện về các Charts.

### Cài đặt chương trình

Clone git từ link: https://github.com/hautran-02/cloud-gr1-kubernetes.git

● Đảm bảo đã có một kubernetes cluster

● Thực thi trên Master node:

Cài đặt flask: pip install flask

**git clone https://github.com/hautran-02/cloud-gr1-kubernetes.git**

Chuyển directory đến Kubernestes\_G1/app

Tiến hành chạy app bằng lệnh : python index.py

- Kết quả :

Kết quả: Deploy lên web với port 5000

Text

Description automatically generated

### Demo

Tham khảo qua các video sau:

|  |  |  |
| --- | --- | --- |
| **Thời điểm** | **Link video (Youtbe)** | **Đoạn demo trong video** |
| Báo cáo đồ án giữa kỳ | https://www.youtube.com/watch?v=RpyooYMwfXs&t=520s | từ phút 9:32 |
| Báo cáo đồ án cuối kỳ phần 1 | https://www.youtube.com/watch?v=MJ6aUaK7\_94&t=852s | từ phút 14:48 |
| Báo cáo đồ án cuối kỳ phần 2 | https://www.youtube.com/watch?v=WLSn\_I\_HS40&t=30s | từ phút 14:14 |

# KẾT LUẬN

## Kết quả thu được

Sau một học kỳ học tập và nghiên cứu về K8s, nhóm chúng em đã tìm hiểu và lĩnh hội được rất nhiều kiến thức về K8s và các khía cạnh xung quanh nó như tầm quan trọng, thành phần. công dụng, quá trình cài đặt. Ngoài ra, tìm hiểu K8s giúp chúng em hiểu thêm được các khái niệm khác như Docker, Helm, Scaling, Kubernetes Volume. Từ đó giúp chúng em hoàn thành được các yêu cầu của đồ án học phần:

- Hiểu và nắm bắt các kiến thức quan trọng về Container, Docker, Kubernetes

- Tạo và phát triển một cụm Kubernetes bao gồm 3 node.

- Deploy các application trên cụm Kubernetes.

- Tạo một trang web giúp giảm bớt công việc triển khai và quản lý các kubernetes.

Ngoài ra cũng không thể nhắc đến AWS và Cloud Computing (Điện toán đám mây) vì đây là một nền tảng vô cùng quan trọng và đa dạng, cần phải lĩnh hội được trước khi đến với K8s.

## Ưu, nhược điểm

Sau khi xem lại toàn bộ đồ án cũng như là quá trình làm đồ án thì nhóm chúng em đã nhận thấy được một số ưu nhược điểm từ đồ án.

### Ưu điểm

- Giúp quá trình tạo các helm chart được dễ dàng hơn

- Triển khai đơn giản các chart application chỉ bằng vài thao tác đơn giản

- Trực quan hóa giao diện để dễ dàng quản lý các ứng dụng trên Kubernetes.

### Nhược điểm

- Đồ án chỉ mô phỏng vào quá trình cài đặt và sử dụng cơ bản của K8s nhưng chưa đi sâu vào cách thức quản lý và cấu hình bên trong file YAML.

- Chưa vận dụng được nhiều về lý thuyết khi sử dụng PersistentVolumeClaim hoặc PersistentVolume: các ứng dụng sử dụng đến lớp storage thì chưa được hoàn chỉnh, còn nhiều thiếu sót.

- Website còn đơn giản, chưa có nhiều tùy chọn nên chỉ demo được một số chức năng cơ bản của K8s.

## Hướng phát triển

- Sử dụng thêm tính năng của K8s như K8s Volume để thêm nhiều loại Chart có thể Deloy trên web.

- Kết hợp với autoscale của AWS để tự mở rộng máy ảo khi các máy hiện tại quá tải.

- Thiết kế lại trang web bắt mắt, nhiều tính năng hơn.

# TÀI LIỆU THAM KHẢO

1. Sử dụng Persistent Volume (pv) và Persistent Volume Claim (pvc) trong Kubernetes, <https://xuanthulab.net/su-dung-persistent-volume-pv-va-persistent-volume-claim-pvc-trong-kubernetes.html?fbclid=IwAR1NVaoYi_aAxKH8TjskiMWP-jiFiKsV4QdFH-K5vXAHZAR9xaLiwb0T9ZU>
2. Nguyen Van Tam. Hiểu cơ bản khái niệm Auto-scale của K8S. Truy cập ngày 19/12/2022 tại: <https://viblo.asia/p/hieu-co-ban-khai-niem-auto-scale-cua-k8s-maGK73rbKj2>
3. WheeBoo, Helm là gì ? Nó có liên quan gì đến series này ?. Truy cập ngày 19/12/2022 tại: <https://viblo.asia/p/helm-la-gi-no-co-lien-quan-gi-den-series-nay-Do754oAQlM6>
4. Cloud Native Viet, Volumes. Truy cập ngày 19/12/2022 tại: [https://www.cloudnativeviet.net/docu ments/kubernetes/storage/volumes/#hostpath](https://www.cloudnativeviet.net/docu%20ments/kubernetes/storage/volumes/#hostpath)
5. Quân Huỳnh, Kubernetes Series - Bài 16 - Automatic scaling Pod và cluster. Truy cập ngày 19/12/2022 tại: <https://viblo.asia/p/kubernetes-series-bai-16-automatic-scaling-pod-va-cluster-YWOZrGyRlQ0>
6. Le Thi Ngoc Tram, Tìm hiểu về EBS (Elastic Block Store) của AWS. Truy cập ngày 19/12/2022 tại: <https://viblo.asia/p/tim-hieu-ve-ebs-elastic-block-store-cua-aws-gGJ596EDKX2>
7. Amazon, Docker là gì?. Truy cập ngày 19/12/2022: <https://aws.amazon.com/vi/docker/>
8. TopDev, Docker là gì? Tìm hiểu về Docker. Truy cập ngày 19/12/2022 tại: https://topdev.vn/blog/docker-la-gi/