

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**  
**TRƯỜNG ĐẠI HỌC NGOẠI NGỮ - TIN HỌC**  
**THÀNH PHỐ HỒ CHÍ MINH**



**ĐỒ ÁN MÔN HỌC**  
**ĐIỆN TOÁN Đám Mây**

**Nguyễn Công Khang - 21DH110770**

**Phạm Đức Thiên Phúc - 21DH112813**

**Lê Viết Nam – 21DH112687**

**GVGD: Th.S Cao Tiến Thành**

# MỤC LỤC

<b>LỜI NÓI ĐẦU</b> .....	5
<b>Chương 1: Giới thiệu</b> .....	6
<b>Chương 2: Cơ sở lý thuyết và cài đặt</b> .....	7
<b>1.1 Container</b> .....	7
a. Khái niệm .....	7
b. Đặc điểm kĩ thuật.....	7
c. Ưu và nhược điểm.....	8
<b>1.2 Docker</b> .....	9
a. Khái niệm .....	9
b. Lợi ích của việc sử dụng Docker .....	11
c. Khi nào sử dụng Docker.....	11
<b>1.3 Kubernetes</b> .....	11
a. Khái niệm .....	11
b. Lợi ích của việc sử dụng Kubernetes .....	12
c. Sơ đồ Kubernetes.....	14
d. Kiến trúc tổng quan về Kubernetes .....	15
e. Nguyên lý hoạt động của Kubernetes .....	17
f. Ưu điểm nổi bật của Kubernetes .....	18
g. So sánh với các công nghệ tương đương.....	19
h. Cài đặt Kubernetes .....	20
<b>Chương 3: Triển khai dịch vụ</b> .....	27
○ Web tĩnh .....	27
○ Web động .....	30
Scan file Kubescape .....	34
Link demo chạy Kubernetes.....	38
<b>Kết Luận</b> .....	39
<b>Tài liệu tham khảo</b> .....	40

## Danh mục hình ảnh

Hình 1 Mô hình kiến trúc container.....	8
Hình 2 Mô hình tổng quát về Docker .....	10
Hình 3 Mô hình Kubernetes .....	14
Hình 4 Mô hình Master Node .....	16
Hình 5 Mô hình Worker Node .....	17
Hình 6 Update Ubuntu .....	20
Hình 7 Upgrade Ubuntu .....	21
Hình 8 Reboot hệ thống .....	21
Hình 9 Đặt hostname .....	21
Hình 10 Cài nano editor .....	21
Hình 11 Cập nhật file .....	21
Hình 12 Tắt phân vùng swap trên hệ thống .....	21
Hình 13 Kiểm tra xem swap đã disable chưa.....	21
Hình 14 Sau đó disable trên file trên .....	21
Hình 15 Tải kernel modules trên nodes.....	22
Hình 16 Tải kernel modules trên nodes.....	22
Hình 17 Thiết lập tham số.....	22
Hình 18 Reload sysctl .....	22
Hình 19 Cài đặt containerd .....	23
Hình 20 Cài đặt containerd .....	23
Hình 21 Cài đặt containerd .....	23
Hình 22 Cài đặt containerd .....	23
Hình 23 Cài đặt containerd .....	24
Hình 24 Cấu hình containerd .....	24
Hình 25 Cài đặt Kubernetes .....	24
Hình 26 Cài đặt Kubernetes .....	24
Hình 27 Key join giữa master với worker .....	25
Hình 28 Cấu hình Kubernetes .....	25

Hình 29 Paste key vào máy worker.....	25
Hình 30 Tải calico.....	25
Hình 31 Chỉnh sửa file calico.....	25
Hình 32 Kiểm tra trạng thái nodes .....	26
Hình 33 Liệt kê các node trong cluster .....	27
Hình 34 Liệt kê các pods trong cluster .....	27
Hình 35 Liệt kê các services trong cluster .....	27
Hình 36 Liệt kê các deployment trong cluster .....	27
Hình 37 Mở file my-k8s-project.....	28
Hình 38 Mở file deployment.....	28
Hình 39 Hình ảnh file deployment .....	28
Hình 40 Mở file configmap.....	28
Hình 41 Hình ảnh file deployment .....	29
Hình 42 Mở file service .....	29
Hình 43 Hình ảnh file service .....	29
Hình 44 Hình ảnh trang web tĩnh .....	30
Hình 45 Liệt kê các pods trong cluster .....	30
Hình 46 Liệt kê các pvc .....	30
Hình 47 Truy cập service chạy trong minikube.....	31
Hình 48 Liệt kê các node trong cluster .....	31
Hình 49 Liệt kê các service trong cluster .....	31
Hình 50 Hình ảnh trang web .....	31
Hình 51 Hình ảnh trang web .....	32
Hình 52 Hình ảnh trang web .....	32
Hình 53 Hình ảnh trang web .....	33
Hình 54 Hình ảnh trang web .....	33

## LỜI NÓI ĐẦU

Ngày nay, ngành công nghệ thông tin đang phát triển rất mạnh mẽ, góp phần thúc đẩy quá trình chuyển hóa trong thời đại công nghệ số. Đi cùng với đó là những vấn đề phát sinh liên quan đến sự phụ thuộc của phần mềm vào môi trường của hệ thống khi chuyển giao sản phẩm, ứng dụng là rất lớn. Cách giải quyết vấn đề này là công nghệ ảo hóa đang “làm mưa làm gió” trong cộng đồng công nghệ trong nước và thế giới hiện nay là “Container”. Vì vậy, trong báo cáo này, em xin trình bày những hiểu biết về Kubernetes và Docker Container trong điện toán đám mây.

Em xin cảm ơn **Th.S Cao Tiến Thành** đã hướng dẫn và giảng dạy làm đồ án trong kỳ học này. Trong quá trình làm báo cáo do còn hạn chế về kiến thức nên còn khó tránh khỏi những sai sót. Mong nhận được đánh giá và nhận xét của thầy để em rút kinh nghiệm và hoàn thiện đề tài này hơn.

Em xin chân thành cảm ơn ạ!

# Chương 1: Giới thiệu

## 1. Phân tích đề tài

Đề tài điện toán đám mây sử dụng Kubernetes là một lĩnh vực nghiên cứu và áp dụng ngày càng phổ biến trong thế giới công nghệ hiện đại. Đây là một lĩnh vực liên quan đến việc sử dụng Kubernetes, một nền tảng quản lý container mạnh mẽ, để tối ưu hóa và tự động hóa việc triển khai, quản lý, và mở rộng các ứng dụng trên các môi trường điện toán đám mây.

## 2. Tại sao Kubernetes lại quan trọng trong điện toán đám mây

**Quản lý đơn giản hóa:** Kubernetes cung cấp một cách tiếp cận thống nhất và quản lý tập trung cho việc triển khai và vận hành các ứng dụng trên các nền tảng đám mây công cộng như AWS, Azure, Google Cloud, hay các môi trường điện toán đám mây riêng (private cloud).

**Tự động hóa:** Với Kubernetes, các quy trình triển khai và quản lý tài nguyên có thể được tự động hóa, giúp tối ưu hóa sử dụng tài nguyên và giảm thiểu sự can thiệp thủ công.

**Mở rộng linh hoạt:** Kubernetes cho phép mở rộng linh hoạt theo nhu cầu, từ việc mở rộng các ứng dụng đơn lẻ đến mở rộng toàn bộ hệ thống với hàng trăm hoặc hàng nghìn container.

**Khả năng đám mây đa nền tảng:** Kubernetes không chỉ hỗ trợ các nền tảng công cộng, mà còn có thể triển khai trên các môi trường đám mây riêng (private cloud) và kết hợp giữa chúng để xây dựng các mô hình đa đám mây.

**Đảm bảo tính sẵn sàng và tin cậy:** Nhờ vào tính năng tự phục hồi và khả năng mở rộng, Kubernetes giúp đảm bảo rằng các ứng dụng luôn có sẵn và hoạt động ổn định.

## 3. Đối tượng nghiên cứu và ứng dụng của đề tài

**Nghiên cứu:** Nghiên cứu trong lĩnh vực này có thể tập trung vào tối ưu hóa hiệu suất và tài nguyên của Kubernetes, phát triển các công cụ quản lý và giám sát tiên tiến, nghiên cứu về an ninh và bảo mật trong môi trường Kubernetes.

**Ứng dụng thực tiễn:** Trong thực tế, đề tài này có thể áp dụng để xây dựng các hệ thống ứng dụng có khả năng mở rộng và linh hoạt cao trên nền tảng điện toán đám mây, từ các ứng dụng web, dịch vụ microservices đến các ứng dụng phân tích dữ liệu và trí tuệ nhân tạo.

## Chương 2: Cơ sở lý thuyết và cài đặt

### 1.1 Container

#### a. Khái niệm

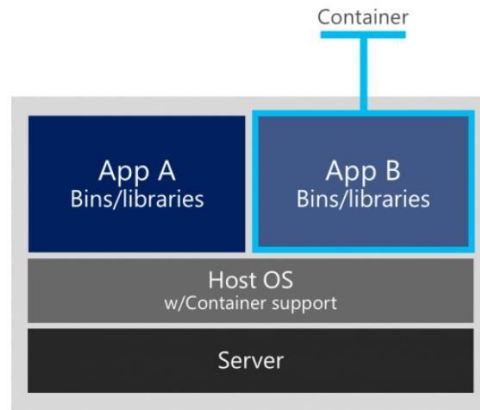
Container là giải pháp để giải quyết vấn đề làm sao để chuyển giao phần mềm một cách đáng tin cậy giữa các môi trường máy tính khác nhau. Chẳng hạn như giữa máy tính của lập trình viên với máy của tester, giữa môi trường staging (hay còn gọi là môi trường tiền thực tế) với môi trường thực tế hay thậm chí giữa máy chủ riêng đặt tại trung tâm dữ liệu với máy ảo trên Cloud.

Container giải quyết vấn đề trên bằng cách tạo ra một môi trường bị cô lập chứa mọi thứ mà phần mềm cần để có thể chạy được bao gồm mã nguồn, các thư viện runtime, các thư viện hệ thống, các công cụ hệ thống mà không bị các yếu tố liên quan đến môi trường hệ thống làm ảnh hưởng tới cũng như không làm ảnh hưởng tới các phần còn lại của hệ thống

#### b. Đặc điểm kỹ thuật

Mô hình kiến trúc của container bao gồm các thành phần chính là Server (máy chủ vật lý hoặc máy ảo), host OS (hệ điều hành cài đặt trên server) và các container. Mỗi một ứng dụng (App A và App B) sẽ có những sự phụ thuộc riêng của nó bao gồm cả về phần mềm lẫn cả phần cứng. Các ứng dụng này sẽ được Container Engine, một công cụ ảo hóa tinh gọn, được cài đặt trên host OS, nó sẽ cô lập sự phụ thuộc của các ứng dụng khác nhau bằng cách đóng gói chúng thành các container. Các tiến trình trong 1 container bị cô lập với các tiến trình của các container khác trong cùng hệ thống, tuy nhiên tất cả các container này đều chia sẻ kernel của host OS

Với mô hình trên, sự phụ thuộc của ứng dụng vào tầng OS cũng như cơ sở hạ tầng được loại bỏ giúp việc triển khai phương pháp “deploy anywhere” của container được hiệu quả hơn. Thêm vào đó, do chia sẻ host OS nên container có thể được tạo gần như một cách tức thì, giúp việc scale-up và scale-down theo nhu cầu được thực hiện một cách nhanh chóng.



*Hình 1 Mô hình kiến trúc container*

### c. Ưu và nhược điểm

#### Ưu điểm:

- **Có kích thước nhỏ và nhẹ:** Các container chia sẻ kernel của máy chủ lưu trữ, chúng chỉ chứa các thành phần thực sự cần thiết với hệ điều hành và thư viện. Đồng thời các container thường cũng chỉ giới hạn ở một chức năng duy nhất, nên có kích thước rất nhỏ. Nhờ vậy, việc xây dựng, triển khai cực kỳ nhanh chóng.
- **Triển khai nhanh:** Do kích thước nhỏ, các container có thể chỉ cần vài giây để khởi động, thậm chí là ít hơn, nên rất thích hợp cho các ứng dụng cần được đẩy lên và xuống liên tục, chẳng hạn như các ứng dụng “serverless”.
- **CI/CD:** Các container được thiết kế để có thể start và restart thường xuyên, nhờ vậy mà dễ dàng tiếp nhận các thay đổi.
- **Có tính di động:** Với thiết kế độc lập, việc di chuyển container giữa các máy có thể thực hiện tương đối dễ dàng, miễn là đúng vị trí kernel.
- **Đảm bảo tính nhất quán:** Giải pháp đóng gói ứng dụng bao gồm source code và các library, framework,... một lần duy nhất và đem chạy ở bất kỳ đâu.

#### Nhược điểm:

- **Có thể yêu cầu kết nối mạng phức tạp:** Thường thường các chức năng sẽ được chia thành nhiều container và cần phải giao tiếp với nhau. Việc số



lượng rất nhiều các container phải giao tiếp với nhau có thể phức tạp. Một số hệ thống điều phối như Kubernetes có các multi-container pods giúp việc trao đổi dễ dàng hơn một chút, nhưng được cho là vẫn phức tạp hơn so với sử dụng máy ảo. Thực tế thì mô hình mạng L3 trong Kubernetes đơn giản hơn nhiều so với mô hình L2 trong hạ tầng máy ảo OpenStack. Vì vậy, vấn đề nằm ở chỗ cần xác định được việc giao tiếp xảy ra giữa các chức năng hay giữa các máy ảo.

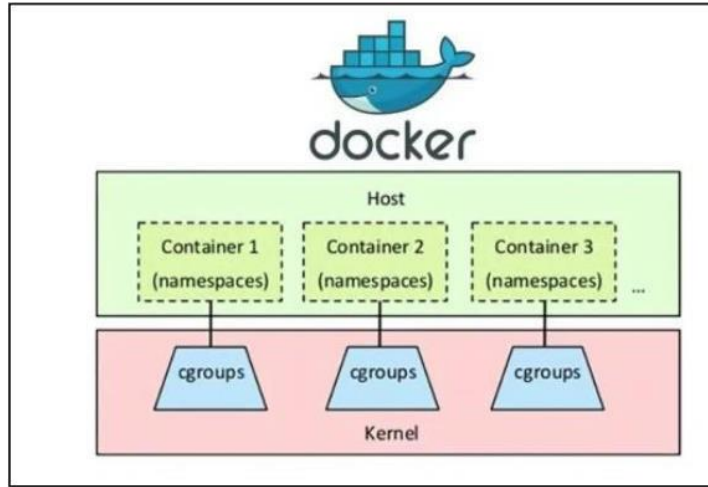
- **Có thể bảo mật kém:** Container vẫn là một công nghệ tương đối non trẻ và chưa đạt được mức độ bảo mật như VM, vậy nên nếu yêu cầu bảo mật cao là tối quan trọng thì đây là 1 trong những yếu tố cần cân nhắc có nên hay không sử dụng container.
- **Có thể cần thao tác nhiều hơn so với máy ảo:** Nếu sử dụng container, bạn sẽ phân tách ứng dụng thành các dịch vụ thành phần khác nhau, mặc dù việc này có ích lợi nhưng lại không cần thiết khi sử dụng VM.

## 1.2 Docker

### a. Khái niệm

Docker là một phần mềm tự động hóa công việc triển khai ứng dụng đóng gói trong các container. Chúng ta cần phải hiểu rằng Docker không “tạo” ra container, mà nó là một nền tảng mà thông qua nó việc tương tác với máy ảo container (khởi tạo, xóa bỏ, quản lý, ...) trở nên dễ dàng và thân thiện hơn rất nhiều.

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



*Hình 2 Mô hình tổng quát về Docker*

dưới dạng một package

Các khái niệm thường được sử dụng trong ảo hóa “container”:

- **Images:** là một bản mẫu (blueprint) của máy ảo container, Images tương tự như file ISO mà chúng ta hay bắt gặp khi cài hệ điều hành, nó chứa toàn bộ thông tin cũng như ứng dụng mà chúng ta triển khai trong đây
- **Container:** là phiên bản thực thi sinh ra từ file image, việc tạo ra container từ file image cũng tương tự như việc chúng ta sử dụng file ISO để cài đặt hệ điều hành để sử dụng
- **Docker Hub:** là kho chứa các Images đã được xây dựng sẵn tại địa chỉ [dockerhub.io](https://dockerhub.io), DockerHub tương tự như AppStore trong IOS hay PlayStore trong IOS hay PlayStore trong Android vậy, chúng ta có thể vào đây, tìm kiếm Image chứa ứng dụng thích hợp, tải xuống rồi dùng nó để tạo ra container
- **Docker Engine:** là thành phần chính của Docker, như một công cụ để đóng gói ứng dụng
- **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

## **b. Lợi ích của việc sử dụng Docker**

- **Tính dễ ứng dụng:** Docker rất dễ cho mọi người sử dụng từ lập trình viên, sys admin... nó tận dụng lợi thế của container để build, test nhanh chóng. Có thể đóng gói ứng dụng trên laptop của họ và chạy trên public cloud, private cloud... Câu thần chú là “Build once, run anywhere”.
- **Tốc độ:** Docker container rất nhẹ và nhanh, bạn có thể tạo và chạy docker container trong vài giây
- **Môi trường chạy và khả năng mở rộng:** Bạn có thể chia nhỏ những chức năng của ứng dụng thành các container riêng lẻ. Ví dụ Database chạy trên một container và Redis cache có thể chạy trên một container khác trong khi ứng dụng Node.js lại chạy trên một cái khác nữa. Với Docker, rất dễ để liên kết các container với nhau để tạo thành một ứng dụng, làm cho nó dễ dàng scale, update các thành phần độc lập với nhau.

## **c. Khi nào sử dụng Docker**

- Triển khai kiến trúc Microservices.
- Khi xây dựng ứng dụng và cần scale một cách linh hoạt.
- Khi muốn không tốn quá nhiều thời gian để config máy local và server cùng một môi trường để chạy được ứng dụng. Bạn chỉ cần build 1 lần chạy ở nhiều nơi mà thôi.
- Sản phẩm của công ty bạn cần một cách tiếp cận mới về xây dựng, đẩy lên server, thực thi ứng dụng một cách nhanh chóng dễ dàng.

## **1.3 Kubernetes**

### **a. Khái niệm**

Còn gọi là k8s, là một nền tảng mã nguồn mở tự động hóa việc quản lý,

scaling và triển khai ứng dụng dưới dạng container. Nó loại bỏ rất nhiều quy trình thủ công liên quan đến việc triển khai và mở rộng các containerized applications.

Kubernetes orchestration cho phép bạn xây dựng các dịch vụ ứng dụng mở rộng nhiều containers. Nó lên lịch các containers đó trên một cụm, mở rộng các containers và quản lý tình trạng của các containers theo thời gian. Các ứng dụng production thực tế mở rộng nhiều containers. Các containers đó phải được triển khai trên nhiều server hosts. Kubernetes cung cấp khả năng phối hợp và quản lý cần thiết để triển khai các containers theo quy mô cho các workloads đó.

Kubernetes ban đầu được phát triển và thiết kế bởi các kỹ sư tại Google, đây cũng là công nghệ đằng sau các dịch vụ đám mây của Google. Google đã và đang tạo ra hơn 2 tỉ container deployments mỗi tuần và tất cả đều được hỗ trợ bởi nền tảng nội bộ.

## **b. Lợi ích của việc sử dụng Kubernetes**

Kubernetes cung cấp:

- **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 storage, public cloud provide, v.v.

- **Tự động rollout 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 echo việc triển khai của bạn, xóa các

containere hiện có và áp dụng tất cả các resource của chúng vào containere 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 containere cần bao nhiêu CPU và bộ nhớ RAM. Kubernetes có thể điều phối các containere đến các node để tận dụng tốt nhất các resource của bạn.

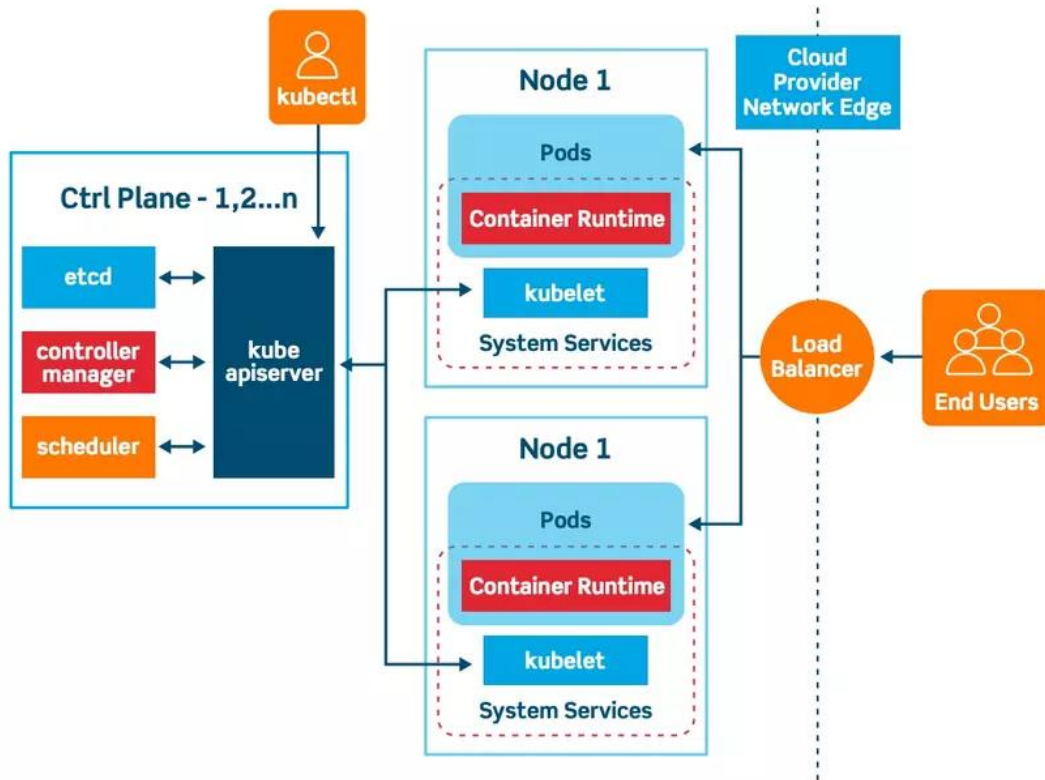
- **Tự phục hồi**

Kubernetes khởi động lại các containeré bị lỗi, thay thế các containere, xóa 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 lí cấu hình và bảo mật**

Kubernetes cho phép bạn lưu trữ và quản lí 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.

### c. Sơ đồ Kubernetes



Hình 3 Mô hình Kubernetes

#### etcd:

- **Mô tả:** etcd là một kho lưu trữ key-value phân tán, mạnh mẽ và nhất quán, được sử dụng để lưu trữ dữ liệu cấu hình và trạng thái của cụm Kubernetes.
- **Chức năng:** Lưu trữ tất cả thông tin cần thiết để quản lý và điều hành cụm Kubernetes, bao gồm thông tin về nodes, pods, services, config maps, secrets, và nhiều hơn nữa.
- **Đặc điểm:** Hỗ trợ tính nhất quán cao và chịu lỗi, rất quan trọng cho việc duy trì trạng thái của cụm Kubernetes.

#### Kubernetes API Server:

- **Mô tả:** Là thành phần trung tâm của cụm Kubernetes, chịu trách nhiệm xử lý tất cả các yêu cầu API.
- **Chức năng:** Tương tác với etcd để lấy và lưu trữ trạng thái của cụm. Tất

cả các thao tác quản lý cụm đều thông qua API Server.

### **Kubernetes Controller Manager:**

- **Mô tả:** Chứa các bộ điều khiển (controllers) khác nhau để quản lý các tác vụ và trạng thái của cụm.
- **Chức năng:** Các bộ điều khiển này theo dõi trạng thái của các đối tượng trong cụm và thực hiện các hành động cần thiết để duy trì trạng thái mong muốn.

### **Kube-Scheduler:**

- **Mô tả:** Thành phần chịu trách nhiệm lên lịch cho các pods trên các nodes trong cụm.
- **Chức năng:** Đọc thông tin từ API Server, quyết định nodes nào phù hợp để chạy các pods mới dựa trên tài nguyên và các ràng buộc khác.

### **Kubelet:**

- **Mô tả:** Chạy trên mỗi node trong cụm và chịu trách nhiệm cho việc khởi động và quản lý các pods được gán cho node đó.
- **Chức năng:** Tương tác với API Server để nhận các chỉ thị và báo cáo lại trạng thái của các pods trên node.

## **Tóm tắt**

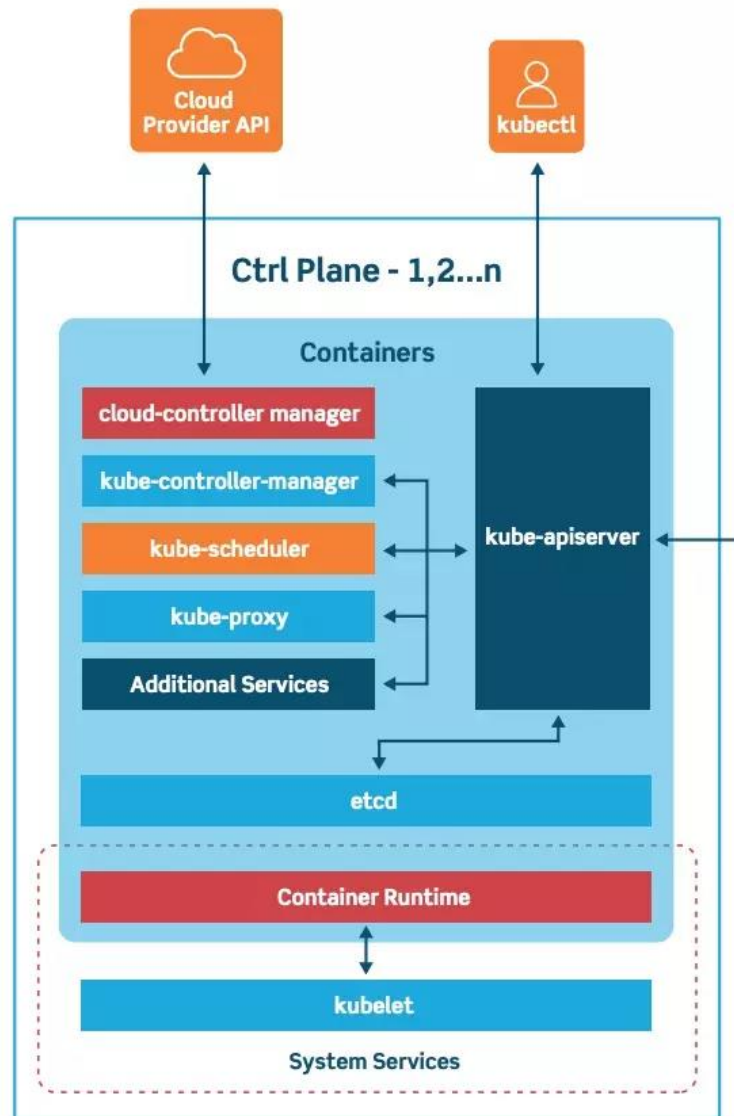
- **etcd:** Lưu trữ trạng thái của cụm.
- **Kubernetes API Server:** Giao diện trung tâm cho tất cả các hoạt động.
- **Kubernetes Controller Manager:** Đảm bảo trạng thái mong muốn của cụm.
- **Kube-Scheduler:** Quyết định nơi chạy các pods.
- **Kubelet:** Quản lý pods trên mỗi node.

## **d. Kiến trúc tổng quan về Kubernetes**

**Master Node (Control Plane):** Điều khiển toàn bộ cụm Kubernetes.

- **API Server:** Giao diện trung tâm cho tất cả các thao tác, nhận các yêu cầu từ người dùng, các công cụ CLI, và các thành phần bên trong khác.
- **etcd:** Lưu trữ trạng thái của cụm, như một kho lưu trữ key-value phân tán.

- **Controller Manager:** Chạy các bộ điều khiển khác nhau để duy trì trạng thái mong muốn của cụm.
- **Scheduler:** Lên lịch cho các pods trên các nodes dựa trên tài nguyên và yêu cầu.



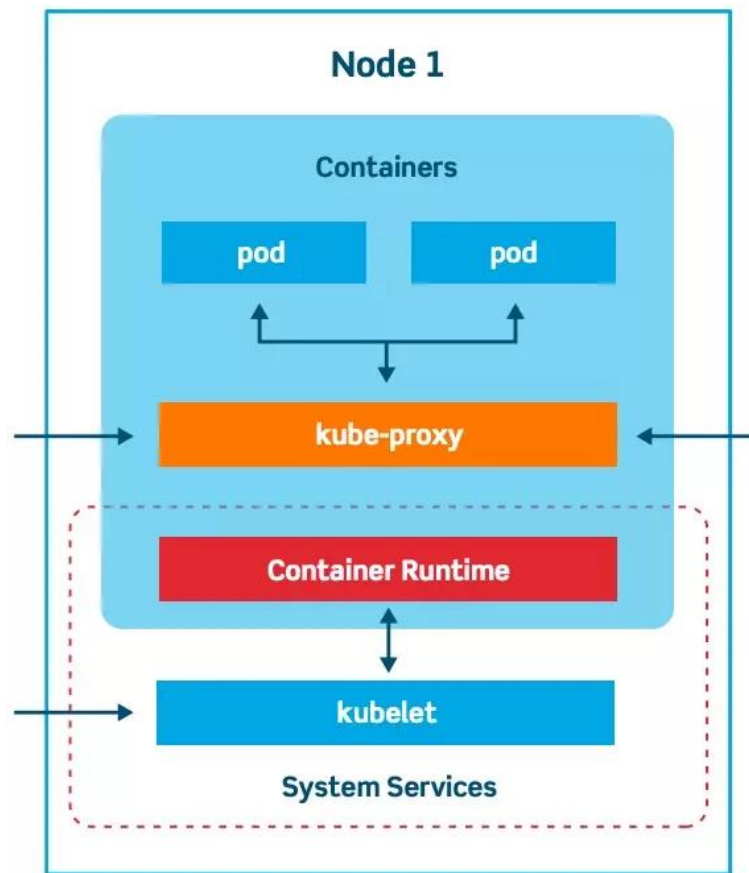
Hình 4 Mô hình Master Node

**Worker Node:** Chạy các ứng dụng trong các container.

- **Kubelet:** Đảm bảo các pods và containers chạy như mong muốn trên node.
- **Kube-Proxy:** Quản lý mạng, định tuyến lưu lượng đến các pods.



- **Container Runtime:** Chạy các containers, thường là Docker, containerd, hoặc CRI-O.



*Hình 5 Mô hình Worker Node*

## e. Nguyên lý hoạt động của Kubernetes

### **Bước 1: Khai báo trạng thái mong muốn:**

- Người dùng khai báo trạng thái mong muốn của ứng dụng và môi trường triển khai thông qua các tệp YAML hoặc JSON (ví dụ: Deployment, Service).
- Các tệp này được gửi đến API Server.

### **Bước 2: Lưu trữ và xác thực:**

- API Server xác thực và lưu trữ các khai báo này vào etcd.
- etcd lưu trữ trạng thái hiện tại và mong muốn của cụm.

### **Bước 3: Lên lịch và khởi tạo:**

- Scheduler đọc trạng thái mong muốn từ API Server và xác định node phù hợp để chạy các pods mới dựa trên tài nguyên có sẵn và các ràng buộc khác.
- Scheduler cập nhật API Server với quyết định lên lịch của mình.

#### **Bước 4: Thực thi và giám sát:**

- Kubelet trên mỗi worker node nhận lệnh từ API Server và khởi tạo các containers theo yêu cầu.
- Kubelet giám sát trạng thái của các pods và báo cáo lại cho API Server.

#### **Bước 5: Quản lý và điều chỉnh**

- Controller Manager chạy các bộ điều khiển để theo dõi trạng thái của các đối tượng trong cụm.
- Nếu có sự khác biệt giữa trạng thái hiện tại và trạng thái mong muốn, các bộ điều khiển sẽ thực hiện các hành động cần thiết để điều chỉnh (ví dụ: khởi động lại pods bị hỏng, mở rộng số lượng replicas).

### **f. Ưu điểm nổi bật của Kubernetes**

#### **Cung cấp nền tảng để lên lịch và chạy các Containers**

Một trong những ưu điểm chính của việc sử dụng K8S chính là mang đến 1 nền tảng vững chắc để lên lịch và chạy các Containers. Đặc biệt Kubernetes cực kỳ cần thiết nếu bạn đang tối ưu App Dev cho Cloud. Bởi vì nền tảng này sẽ cho phép người dùng lên lịch và chạy các Containers trên các Clusters của máy vật lý hoặc máy ảo.

Công nghệ Kubernetes sẽ giúp người dùng dễ dàng triển khai và hoạt động trên cơ sở hạ tầng Container – Based trong môi trường sản xuất. Việc Kubernetes có khả năng tự động hóa các hoạt động vận hành sẽ giúp người dùng có thể thực hiện nhiều tác vụ đối với Container. Ưu điểm này không chỉ hỗ trợ người dùng hoạt động hữu ích đối với Container mà còn có thể làm việc trên nhiều nền tảng ứng dụng khác hoặc các hệ quản lý khác.

Vì vậy với Kubernetes bạn hoàn toàn có thể thực hiện các nhiệm vụ sau:

- Điều hành, phân bố Container trên nhiều máy chủ

- Tận dụng phần cứng nhiều hơn để tối đa hóa tài nguyên cần thiết cho việc chạy các ứng dụng doanh nghiệp.
- Kiểm soát, tự động triển khai ứng dụng
- Cập nhật ứng dụng
- Gắn, bổ sung thêm bộ nhớ để chạy mượt các ứng dụng Stateful
- Mở rộng các ứng dụng chứa trong các Container
- Mở rộng tài nguyên của Container một cách nhanh chóng
- Health – Check và Self – Heal các ứng dụng bằng tính năng tự động phát hiện, sửa lỗi, dò tìm và mở rộng

### **Cung cấp không gian lưu trữ quy mô lớn**

Sở dĩ Kubernetes được đánh giá cao ưu thế này là vì Kubernetes được thiết kế xây dựng mang đến nhiều không gian lưu trữ khác nhau. Người dùng khi sử dụng sẽ có nhiều tùy chọn lưu trữ để lựa chọn. Chẳng hạn như các tùy chọn lưu trữ SAN cục bộ hay lưu trữ vào không gian đám mây công cộng,...

### **Vận hành mọi nơi**

Kubernetes – Hệ thống mã nguồn mở sở hữu ưu điểm là cho phép người dùng tự do tận dụng cơ sở hạ tầng tại chỗ. Ngoài ra, người dùng cũng có thể lựa chọn cơ sở hạ tầng điện toán đám mây để di chuyển khối lượng lớn công việc của mình đến những nơi quan trọng mà bạn mong muốn.

### **g. So sánh với các công nghệ tương đương**

	Hiệu suất	Giá thành	Dịch vụ
Kubernetes	Kubernetes cho phép bạn tự quản lý và tối ưu hóa hiệu suất của hạ tầng của mình. Tuy nhiên, điều này yêu cầu kiến thức kỹ thuật cao.	Kubernetes là mã nguồn mở và miễn phí. Tuy nhiên, chi phí triển khai và quản lý có thể tăng	Kubernetes tập trung vào việc quản lý container và không cung cấp các dịch vụ khác như lưu trữ dữ

		lên do yêu cầu kỹ thuật cao.	liệu, cơ sở dữ liệu, và dịch vụ AI.
AWS	AWS cung cấp hiệu suất cao và khả năng mở rộng linh hoạt.	Chi phí sử dụng dịch vụ AWS phụ thuộc vào việc sử dụng và cấu hình. Mặc dù chúng có một số lượng miễn phí cho các dịch vụ cơ bản, nhưng chi phí có thể tăng lên nếu sử dụng nhiều tài nguyên hơn.	AWS cung cấp một loạt các dịch vụ bao gồm lưu trữ, tính toán, cơ sở dữ liệu, dịch vụ AI và Machine Learning, IoT, và nhiều hơn nữa.
Vmware Cloud	VMware Cloud cũng cung cấp hiệu suất cao cho các môi trường ảo hóa và điện toán đám mây.	VMware Cloud có thể có chi phí cao hơn so với các nhà cung cấp đám mây công cộng do việc triển khai và quản lý hạ tầng riêng.	VMware Cloud tập trung vào việc cung cấp các giải pháp ảo hóa và điện toán đám mây, bao gồm quản lý và tự động hóa hạ tầng ảo.

## h. Cài đặt Kubernetes

```
anm@anm-VMware-Virtual-Platform:~$ sudo apt update
[sudo] password for anm:
```

*Hình 6 Update Ubuntu*

```
anm@anm-VMware-Virtual-Platform:~$ sudo apt upgrade -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
```

*Hình 7 Upgrade Ubuntu*

```
anm@anm-VMware-Virtual-Platform:~$ sudo reboot
```

*Hình 8 Reboot hệ thống*

```
anm@anm-VMware-Virtual-Platform:~$ sudo hostnamectl set-hostname "k8s-master.phuc.local"
[sudo] password for anm:
```

*Hình 9 Đặt hostname*

```
anm@anm-VMware-Virtual-Platform:~$ sudo apt install -y nano
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
nano is already the newest version (7.2-2build1).
nano set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 11 not upgraded.
```

*Hình 10 Cài nano editor*

```
anm@anm-VMware-Virtual-Platform:~$ sudo nano /etc/hosts
anm@anm-VMware-Virtual-Platform:~$
```

*Hình 11 Cập nhật file*

```
anm@anm-VMware-Virtual-Platform:~$ sudo swapoff -a
anm@anm-VMware-Virtual-Platform:~$
```

*Hình 12 Tắt phân vùng swap trên hệ thống*

```
anm@anm-VMware-Virtual-Platform:~$ free -h
```

	total	used	free	shared	buff/cache	available
Mem:	3.8Gi	1.1Gi	2.1Gi	34Mi	842Mi	2.6Gi
Swap:	0B	0B	0B			

*Hình 13 Kiểm tra xem swap đã disable chưa*

```
anm@anm-VMware-Virtual-Platform:~$ sudo nano /etc/fstab
anm@anm-VMware-Virtual-Platform:~$ sudo mount -a
```

*Hình 14 Sau đó disable trên file trên*

```
anm@anm-VMware-Virtual-Platform:~$ sudo tee /etc/modules-load.d/containerd.conf
<<EOF
overlay
br_netfilter
EOF
overlay
br_netfilter
```

*Hình 15 Tải kernel modules trên nodes*

```
anm@anm-VMware-Virtual-Platform:~$ sudo modprobe overlay
anm@anm-VMware-Virtual-Platform:~$ sudo modprobe br_netfilter
```

*Hình 16 Tải kernel modules trên nodes*

```
anm@anm-VMware-Virtual-Platform:~$ sudo tee /etc/sysctl.d/kubernetes.conf <<EOF
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
EOF
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
```

*Hình 17 Thiết lập tham số*

```
anm@anm-VMware-Virtual-Platform:~$ sudo sysctl --system
* Applying /usr/lib/sysctl.d/10-apparmor.conf ...
* Applying /etc/sysctl.d/10-console-messages.conf ...
* Applying /etc/sysctl.d/10-ipv6-privacy.conf ...
* Applying /etc/sysctl.d/10-kernel-hardening.conf ...
* Applying /etc/sysctl.d/10-magic-sysrq.conf ...
* Applying /etc/sysctl.d/10-map-count.conf ...
* Applying /etc/sysctl.d/10-network-security.conf ...
* Applying /etc/sysctl.d/10-ptrace.conf ...
* Applying /etc/sysctl.d/10-zeropage.conf ...
* Applying /usr/lib/sysctl.d/30-tracker.conf ...
* Applying /usr/lib/sysctl.d/50-bubblewrap.conf ...
* Applying /usr/lib/sysctl.d/50-pid-max.conf ...
* Applying /usr/lib/sysctl.d/99-protect-links.conf ...
* Applying /etc/sysctl.d/99-sysctl.conf ...
* Applying /etc/sysctl.d/kubernetes.conf ...
* Applying /etc/sysctl.conf ...
```

*Hình 18 Reload sysctl*

```
anm@anm-VMware-Virtual-Platform:~$ sudo apt install -y curl gnupg2 software-properties-common apt-transport-https ca-certificates
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
software-properties-common is already the newest version (0.99.48).
software-properties-common set to manually installed.
ca-certificates is already the newest version (20240203).
ca-certificates set to manually installed.
The following NEW packages will be installed:
```

*Hình 19 Cài đặt containerd*

```
anm@anm-VMware-Virtual-Platform:~$ sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/trusted.gpg.d/docker.gpg
```

*Hình 20 Cài đặt containerd*

```
anm@anm-VMware-Virtual-Platform:~$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
Repository: 'deb [arch=amd64] https://download.docker.com/linux/ubuntu noble stable'
Description:
Archive for codename: noble components: stable
More info: https://download.docker.com/linux/ubuntu
Adding repository.
Press [ENTER] to continue or Ctrl-c to cancel.
```

*Hình 21 Cài đặt containerd*

```
anm@anm-VMware-Virtual-Platform:~$ sudo apt update
Hit:1 http://vn.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://vn.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Hit:3 http://vn.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 https://download.docker.com/linux/ubuntu noble InRelease
Get:5 http://vn.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [205 kB]
Get:6 http://vn.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [91.8 kB]
Get:7 http://vn.archive.ubuntu.com/ubuntu noble-updates/universe Translation-en [32.5 kB]
Hit:8 http://security.ubuntu.com/ubuntu noble-security InRelease
Fetched 455 kB in 1s (659 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
11 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

*Hình 22 Cài đặt containerd*

```
anm@anm-VMware-Virtual-Platform:~$ sudo apt install -y containerd.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  containerd.io
0 upgraded, 1 newly installed, 0 to remove and 11 not upgraded.
Need to get 30.5 MB of archives.
```

*Hình 23 Cài đặt containerd*

```
anm@anm-VMware-Virtual-Platform:~$ containerd config default | sudo tee /etc/con
tainerd/config.toml >/dev/null 2>&1
anm@anm-VMware-Virtual-Platform:~$ sudo sed -i 's/SystemdCgroup \= false/Systemd
Cgroup \= true/g' /etc/containerd/config.toml
anm@anm-VMware-Virtual-Platform:~$ sudo systemctl restart containerd
anm@anm-VMware-Virtual-Platform:~$ sudo systemctl enable containerd
```

*Hình 24 Cấu hình containerd*

```
anm@anm-VMware-Virtual-Platform:~$ echo "deb [signed-by=/etc/apt/keyrings/kubern
etes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.30/deb/ /" | sudo tee
/etc/apt/sources.list.d/kubernetes.list
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io
/core:/stable:/v1.30/deb/ /
anm@anm-VMware-Virtual-Platform:~$ curl -fsSL https://pkgs.k8s.io/core:/stable:/
v1.30/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-k
eyring.gpg
anm@anm-VMware-Virtual-Platform:~$ sudo apt update
sudo apt install -y kubelet kubeadm kubectl
```

*Hình 25 Cài đặt Kubernetes*

```
anm@anm-VMware-Virtual-Platform:~$ sudo apt install -y kubelet kubeadm kubectl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
kubelet is already the newest version (1.30.2-1.1).
kubeadm is already the newest version (1.30.2-1.1).
kubectl is already the newest version (1.30.2-1.1).
0 upgraded, 0 newly installed, 0 to remove and 11 not upgraded.
anm@anm-VMware-Virtual-Platform:~$ sudo apt-mark hold kubelet kubeadm kubectl
kubelet set on hold.
kubeadm set on hold.
kubectl set on hold.
```

*Hình 26 Cài đặt Kubernetes*



```
kubeadm join k8s-master.phuc.local:6443 --token mouh6k.5b42vv099y2u4z6i \
--discovery-token-ca-cert-hash sha256:2467127f1b7a3b65babede7205952bf49b
bc83233aecbc1eafd17360b08381b0
```

*Hình 27 Key join giữa master với worker*

```
anm@anm-VMware-Virtual-Platform:~$ mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

*Hình 28 Cấu hình Kubernetes*

```
anm@anm-VMware-Virtual-Platform:~$ sudo su
root@k8s-worker1:/home/anm# kubeadm join k8s-master.phuc.local:6443 --token mouh
6k.5b42vv099y2u4z6i \
--discovery-token-ca-cert-hash sha256:2467127f1b7a3b65babede7205952bf49b
bc83233aecbc1eafd17360b08381b0
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system g
et cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.y
aml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/ku
belet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-check] Waiting for a healthy kubelet. This can take up to 4m0s
[kubelet-check] The kubelet is healthy after 502.48022ms
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap
```

*Hình 29 Paste key vào máy worker*

```
anm@anm-VMware-Virtual-Platform:~$ curl https://raw.githubusercontent.com/projec
tcalico/calico/v3.25.0/manifests/calico.yaml -O
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left   Speed
100  232k  100  232k    0     0   326k      0 --:--:-- --:--:-- --:--:--  327k
```

*Hình 30 Tải calico*

```
anm@anm-VMware-Virtual-Platform:~$ sudo nano calico.yaml
[sudo] password for anm:
anm@anm-VMware-Virtual-Platform:~$ kubectl apply -f calico.yaml
```

*Hình 31 Chỉnh sửa file calico*

```
anm@k8s-master:~$ kubectl get nodes
NAME                                STATUS    ROLES    AGE   VERSION
k8s-master.phuc.local              Ready     control-plane   33m   v1.30.2
k8s-worker1.phuc.local             Ready     <none>         11m   v1.30.2
k8s-worker2.phuc.local             Ready     <none>         11m   v1.30.2
```

*Hình 32 Kiểm tra trạng thái nodes*

## Chương 3: Triển khai dịch vụ

### ○ Web tĩnh

Chúng em đã triển khai một dịch vụ trên nền cloud, bao gồm việc triển khai một trang web HTML tĩnh lên một node Kubernetes. Dịch vụ này sử dụng cổng 30004 để cung cấp nội dung trang web, giúp dễ dàng truy cập từ bên ngoài cụm. Việc triển khai này đảm bảo rằng trang web HTML tĩnh có thể được phục vụ một cách đáng tin cậy và hiệu quả trên môi trường cloud.

```
khang@k8s-master:~$ kubectl get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
k8s-master.khang.local	Ready	control-plane	4h52m	v1.30.2
k8s-worker1.khang.local	Ready	<none>	4h40m	v1.30.2
k8s-worker2.khang.local	Ready	<none>	4h42m	v1.30.2

Hình 33 Liệt kê các node trong cluster

```
khang@k8s-master:~$ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
nginx-deployment-8b986464f-dwptj	1/1	Running	0	57m
nginx-deployment-8b986464f-grnw9	1/1	Running	0	57m
nginx-deployment-8b986464f-hbw7s	1/1	Running	0	57m
nginx-deployment-8b986464f-jcszw	1/1	Running	0	57m
nginx-deployment-8b986464f-mkbzx	1/1	Running	0	57m
wordpress-7d7846bd75-tvrzk	0/1	Pending	0	4h1m
wordpress-mysql-6dd978bc8b-dzrkx	0/1	Pending	0	4h1m

Hình 34 Liệt kê các pods trong cluster

```
khang@k8s-master:~$ kubectl get svc
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	4h52m
nginx-service	NodePort	10.107.84.211	<none>	80:30004/TCP	4h35m
wordpress	LoadBalancer	10.98.149.14	<pending>	80:32160/TCP	4h1m
wordpress-mysql	ClusterIP	None	<none>	3306/TCP	4h1m

Hình 35 Liệt kê các services trong cluster

```
khang@k8s-master:~$ kubectl get deployment
```

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
nginx-deployment	5/5	5	5	4h35m
wordpress	0/1	1	0	4h1m
wordpress-mysql	0/1	1	0	4h1m

Hình 36 Liệt kê các deployment trong cluster

```
khang@k8s-master:~$ cd my-k8s-project/
```

Hình 37 Mở file my-k8s-project

```
khang@k8s-master:~/my-k8s-project$ sudo nano deployment.yaml  
[sudo] password for khang:
```

Hình 38 Mở file deployment

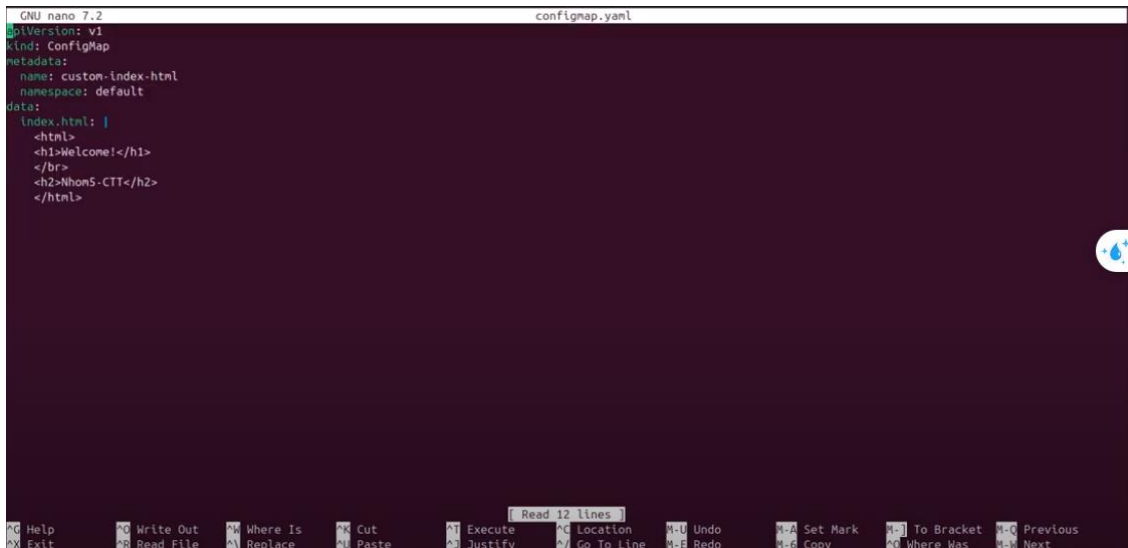
```
GNU nano 7.2 deployment.yaml  
apiVersion: apps/v1  
kind: Deployment  
metadata:  
  name: nginx-deployment  
  namespace: default  
spec:  
  selector:  
    matchLabels:  
      app: nginx  
  replicas: 5  
  template:  
    metadata:  
      labels:  
        app: nginx  
    spec:  
      containers:  
      - name: nginx  
        image: nginx:latest  
        ports:  
        - containerPort: 80  
[ Read 27 lines ]  
^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location  
^X Exit      ^R Read File ^_ Replace   ^U Paste     ^J Justify   ^_ Go To Line
```

Hình 39 Hình ảnh file deployment

- Tạo một Deployment với 5 container Nginx, mỗi container sử dụng nội dung HTML tùy chỉnh từ ConfigMap custom-index-html để hiển thị trang web.

```
khang@k8s-master:~/my-k8s-project$ sudo nano configmap.yaml  
khang@k8s-master:~/my-k8s-project$ sudo nano service.yaml
```

Hình 40 Mở file configmap

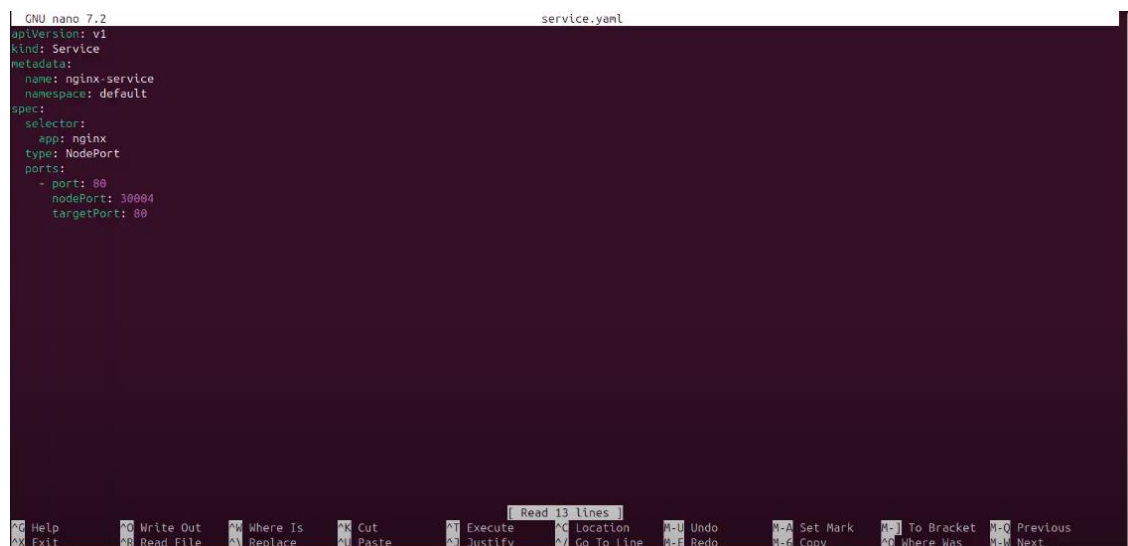


Hình 41 Hình ảnh file configmap

- File YAML này định nghĩa một ConfigMap trong Kubernetes với tên custom-index-html. ConfigMap này chứa dữ liệu tĩnh để cấu hình ứng dụng.
- ConfigMap này được sử dụng để cung cấp cấu hình hoặc nội dung tĩnh cho các Pod trong Kubernetes. Ở đây nó cung cấp một trang HTML tùy chỉnh để hiển thị bởi một ứng dụng web (ví dụ như Nginx).

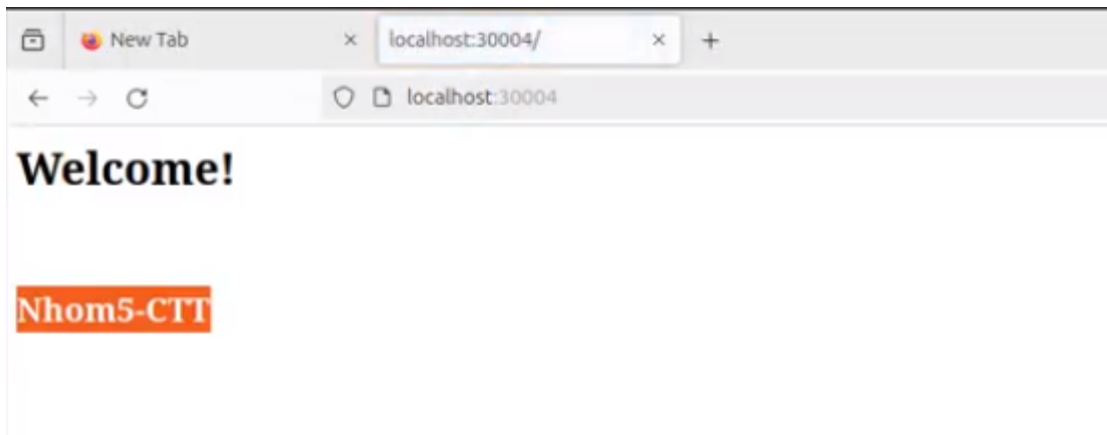


Hình 42 Mở file service



Hình 43 Hình ảnh file service

- File YAML này tạo một Service trong Kubernetes để truy cập các Pod Nginx từ bên ngoài cụm Kubernetes
- Service này cho phép truy cập vào các Pod Nginx thông qua cổng 30004 của các Node trong cụm Kubernetes.



*Hình 44 Hình ảnh trang web tĩnh*

### ○ Web động

Trong phần triển khai dịch vụ gia tăng, chúng tôi sẽ cài đặt MySQL và WordPress để tạo một hệ thống quản lý nội dung mạnh mẽ và dễ sử dụng. MySQL sẽ cung cấp cơ sở dữ liệu vững chắc và hiệu quả, trong khi WordPress, nền tảng CMS phổ biến nhất thế giới, sẽ mang lại trải nghiệm người dùng thân thiện và nhiều tính năng tùy biến. Việc tích hợp này giúp xây dựng và quản lý website một cách nhanh chóng, tiện lợi và bảo mật.

```
khang@k8s-master:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
wordpress-7d7846bd75-65cwg         1/1     Running   0           36m
wordpress-mysql-6dd978bc8b-kfs7n   1/1     Running   0           37m
```

*Hình 45 Liệt kê các pods trong cluster*

```
khang@k8s-master:~$ kubectl get pvc
NAME                                STATUS    VOLUME                                     CAPACITY
ACCESS MODES                       STORAGECLASS          VOLUMEATTRIBUTESCLASS  AGE
mysql-pv-claim                     Bound     pvc-d747a1ae-c49c-4d60-925a-a6ef5b9ea4cc  20Gi
RWO                                 standard             <unset>                 37m
wp-pv-claim                        Bound     pvc-88f44564-e55a-4d27-a100-203d27df4362  20Gi
RWO                                 standard             <unset>                 36m
```

*Hình 46 Liệt kê các pvc*

```
khang@k8s-master:~$ minikube service wordpress --url  
http://192.168.59.100:31596  
khang@k8s-master:~$ kubectl get nodes
```

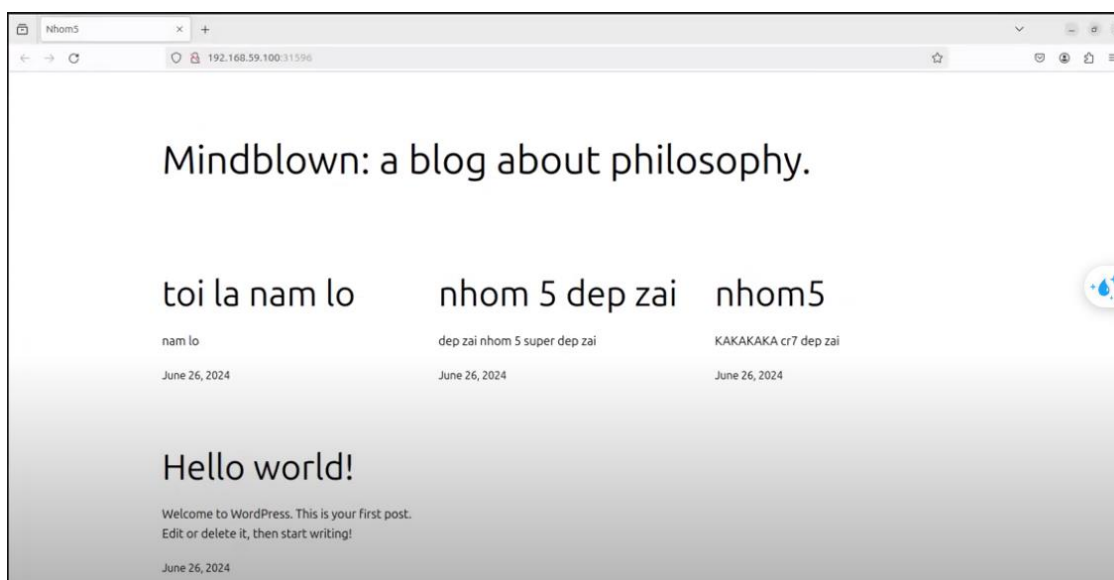
*Hình 47 Truy cập service chạy trong minikube*

```
khang@k8s-master:~$ kubectl get nodes  
NAME          STATUS    ROLES    AGE   VERSION  
minikube      Ready    control-plane  45m   v1.30.0
```

*Hình 48 Liệt kê các node trong cluster*

```
khang@k8s-master:~$ kubectl get service  
NAME          TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE  
kubernetes    ClusterIP     10.96.0.1     <none>         443/TCP          46m  
wordpress     LoadBalancer 10.99.64.33   <pending>      80:31596/TCP     43m  
wordpress-mysql ClusterIP     None         <none>         3306/TCP         44m
```

*Hình 49 Liệt kê các service trong cluster*



*Hình 50 Hình ảnh trang web*



Comments

Leave a Reply

Your email address will not be published. Required fields are marked \*

Comment \*

that tuyet voi

Name \*

ctt

Email \*

ctt@gmail.d

Website

*Hình 51 Hình ảnh trang web*

Posted June 26, 2024 in Uncategorized by nhom5

Tags:

Comments

ctt  
June 26, 2024

Your comment is awaiting moderation.

that tuyet voi

Reply

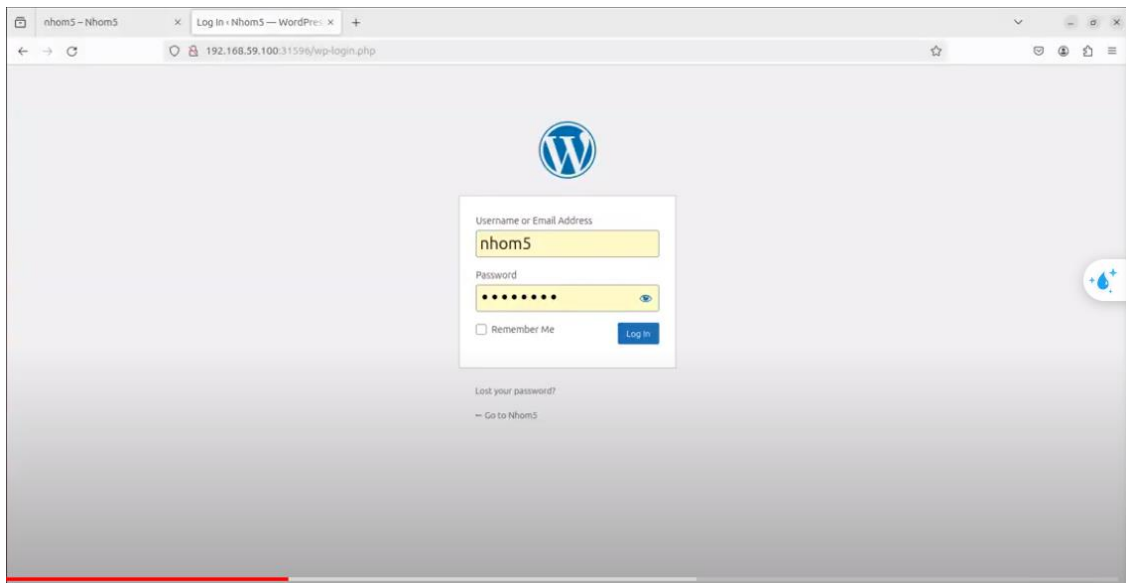
Leave a Reply

Your email address will not be published. Required fields are marked \*

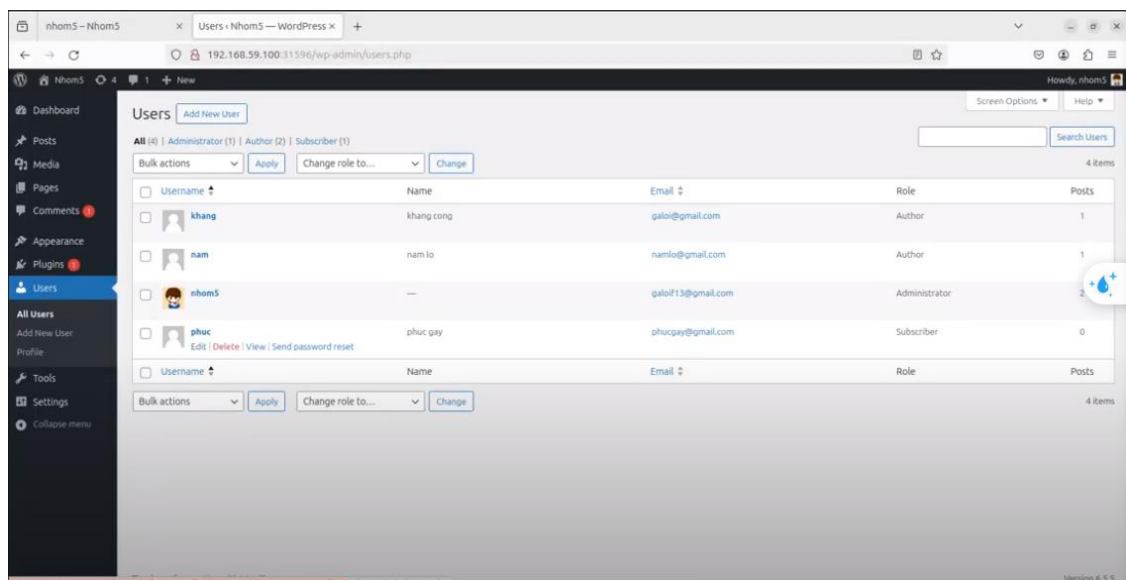
Comment \*

*Hình 52 Hình ảnh trang web*





Hình 53 Hình ảnh trang web



Hình 54 Hình ảnh trang web

## ○ Bảo mật

Ở đây chúng ta sử dụng và triển khai phần mềm KubeScape. Kubescape là một công cụ mã nguồn mở dùng để quét và đánh giá bảo mật cho các ứng dụng và cấu hình Kubernetes. Công cụ này kiểm tra cấu hình Kubernetes so với các chuẩn bảo mật phổ biến như CIS (Center for Internet Security), NSA (National Security Agency), và các quy định bảo mật khác. Dưới đây là một giải thích ngắn gọn và đầy đủ về Kubescape:

### Chức năng chính

- **Quét bảo mật:** Tự động quét và đánh giá các cấu hình Kubernetes.
- **Đối chiếu chuẩn bảo mật:** Kiểm tra so với các chuẩn bảo mật phổ biến (CIS, NSA).
- **Phát hiện lỗ hổng:** Tìm và báo cáo các lỗ hổng bảo mật trong cấu hình Kubernetes.
- **Báo cáo chi tiết:** Cung cấp báo cáo chi tiết và các khuyến nghị để khắc phục vấn đề bảo mật.

## Lợi ích

- **Dễ sử dụng:** Giao diện dòng lệnh đơn giản.
- **Tự động hóa:** Có thể tích hợp vào các quy trình CI/CD để quét bảo mật tự động.
- **Mã nguồn mở:** Miễn phí và có thể tùy chỉnh theo nhu cầu của tổ chức.

## Scan file Kubescape

```
khang@k8s-master:~$ touch pod.yaml
khang@k8s-master:~$ sudo nano pod.yaml
[sudo] password for khang:
```

Hình 55 Tạo tệp pod.yaml và mở trình soạn thảo văn bản nano

```
GNU nano 7.2 pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx:latest
```

Hình 56 Thêm dữ liệu vào tệp pod.yaml

```
khang@k8s-master:~$ kubectl apply -f pod.yaml
Warning: Detected changes to resource nginx which is currently being deleted.
pod/nginx unchanged
```

Hình 57 Triển khai ứng dụng mẫu lên cụm Kubernetes



Critical	0
High	10
Medium	38
Low	4

Run with '--verbose'/'-v' to see control failures for each resource.

Severity	Control name	Failed resources	All Resources	Compliance score
Critical	Disable anonymous access to Kubelet service	0	0	Action Required **
Critical	Enforce Kubelet client TLS authentication	0	0	Action Required **
High	Ensure CPU limits are set	5	9	44%
High	Ensure memory limits are set	5	9	44%
Medium	Prevent containers from allowing command execution	2	58	97%
Medium	Non-root containers	5	9	44%
Medium	Allow privilege escalation	4	9	56%
Medium	Ingress and Egress blocked	9	9	0%
Medium	Automatic mapping of service account	11	17	35%
Medium	Administrative Roles	2	58	97%
Medium	Cluster internal networking	1	3	67%
Medium	Linux hardening	4	9	56%
Medium	Secret/etcd encryption enabled	0	0	Action Required *
Medium	Audit logs enabled	0	0	Action Required *
Low	Immutable container filesystem	4	9	56%
Low	PSP enabled	0	0	Action Required *
Resource Summary		18	84	59.80%

\* failed to get cloud provider, cluster: kubernetes-admin-kubernetes

\*\* This control is scanned exclusively by the Kubescape operator, not the Kubescape CLI. Install the Kubescape operator: <https://kubescape.io/docs/install-operator/>.

Hình 60 Chạy kiểm tra bảo mật của Kubescape

## Giải thích chi tiết sau khi quét

### - Ensure CPU limits are set

**Ensure CPU limits are set:** Đây là một kiểm tra bảo mật nhằm đảm bảo rằng tất cả các container trong cụm Kubernetes của bạn có giới hạn CPU được đặt. Việc đặt giới hạn CPU giúp ngăn chặn một container đơn lẻ tiêu thụ toàn bộ tài nguyên CPU, từ đó giúp duy trì sự ổn định và hiệu suất của toàn bộ cụm.

- **Kết quả:** 3 tài nguyên đã thất bại trong kiểm tra này, có nghĩa là chúng không có giới hạn CPU được đặt.
- **Tóm lại:** Kiểm tra này đã không đạt yêu cầu, bạn nên xem xét việc đặt giới hạn CPU cho tất cả các container để đảm bảo tài nguyên được quản lý hiệu quả và ngăn chặn việc tiêu thụ quá mức.

### - Ensure memory limits are set

**Ensure memory limits are set:** Kiểm tra này nhằm đảm bảo rằng tất cả các container có giới hạn bộ nhớ được đặt. Giới hạn bộ nhớ giúp ngăn chặn một container đơn lẻ tiêu thụ toàn bộ tài nguyên bộ nhớ, gây ảnh hưởng đến các container khác và có thể làm sập hệ thống.

- **Kết quả:** 3 tài nguyên đã thất bại trong kiểm tra này.

- **Tóm lại:** Kiểm tra này đã không đạt yêu cầu, bạn nên thiết lập giới hạn bộ nhớ cho tất cả các container để ngăn ngừa tình trạng cạn kiệt tài nguyên bộ nhớ.

- **Allow privilege escalation**

**Allow privilege escalation:** Kiểm tra này nhằm đảm bảo rằng các container không được phép tăng quyền hạn (privilege escalation) từ người dùng không phải root lên root hoặc từ người dùng có quyền hạn thấp hơn lên cao hơn. Việc cho phép tăng quyền hạn có thể tạo ra các lỗ hổng bảo mật nghiêm trọng.

- **Kết quả:** 3 tài nguyên đã thất bại trong kiểm tra này.
- **Tóm lại:** Bạn cần cấu hình các container sao cho không cho phép tăng quyền hạn, điều này sẽ giúp tăng cường bảo mật cho hệ thống của bạn.

- **Automatic mapping of service account**

**Automatic mapping of service account:** Kiểm tra này nhằm đảm bảo rằng các tài khoản dịch vụ (service accounts) không được ánh xạ tự động. Điều này ngăn ngừa việc lạm dụng các tài khoản dịch vụ để truy cập không đúng vào các tài nguyên khác trong cụm.

- **Kết quả:** 3 tài nguyên đã thất bại trong kiểm tra này, trong tổng số 5 tài nguyên được kiểm tra.
- **Tóm lại:** Bạn nên kiểm tra và cấu hình lại các tài khoản dịch vụ để không cho phép ánh xạ tự động, điều này sẽ giúp ngăn ngừa lạm dụng quyền truy cập.

- **Linux hardening**

**Linux hardening:** Kiểm tra này nhằm đảm bảo rằng các container được cấu hình với các biện pháp tăng cường bảo mật của Linux, chẳng hạn như không cho phép mount hệ thống tập tin proc.

- **Kết quả:** 3 tài nguyên đã thất bại trong kiểm tra này.
- **Tóm lại:** Bạn cần thực hiện các biện pháp tăng cường bảo mật cho các container của mình để giảm thiểu các lỗ hổng bảo mật.

- **Immutable container filesystem**

**Immutable container filesystem:** Kiểm tra này nhằm đảm bảo rằng hệ thống tập tin của container là không thay đổi (immutable). Điều này có nghĩa là sau khi container được triển khai, không ai có thể thay đổi hệ thống tập tin của nó, giúp ngăn chặn các cuộc tấn công và thay đổi không mong muốn.

- **Kết quả:** 3 tài nguyên đã thất bại trong kiểm tra này.
- **Tóm lại:** Bạn nên cấu hình các container của mình để hệ thống tập tin không thể thay đổi sau khi triển khai, điều này sẽ giúp bảo vệ container khỏi các thay đổi trái phép và tăng cường bảo mật.

## **Link demo chạy Kubernetes**

Cài đặt Kubernetes

<https://youtu.be/dbWsd5BOvyc>

Triển khai dịch vụ

<https://youtu.be/ykT5gm2z-h8>

Triển dịch dịch vụ gia tăng

<https://youtu.be/AGnbKkwXYP0>

Demo scan Kuberscape

<https://youtu.be/pw4oQ7HYcBk>

## **Kết Luận**

Trong thời đại công nghệ số phát triển mạnh mẽ hiện nay, điện toán đám mây đã trở thành một phần không thể thiếu trong việc triển khai và quản lý hạ tầng công nghệ thông tin. Đồ án này đã cung cấp một cái nhìn tổng quan về cách sử dụng công nghệ container, cụ thể là Docker và Kubernetes, để triển khai các dịch vụ một cách hiệu quả và bảo mật trên môi trường đám mây.

Trong quá trình thực hiện, nhóm đã triển khai thành công một trang web HTML tĩnh trên nền tảng Kubernetes, tích hợp MySQL và WordPress để xây dựng một hệ thống quản lý nội dung mạnh mẽ. Bên cạnh đó, nhóm đã sử dụng Kubescape để kiểm tra bảo mật và đảm bảo các container được cấu hình đúng đắn, không sử dụng quyền root và có giới hạn tài nguyên CPU và bộ nhớ.

Qua đồ án này, chúng em đã học được nhiều kiến thức quý báu về công nghệ container và điện toán đám mây, cũng như các kỹ năng thực tế trong việc triển khai và quản lý dịch vụ trên nền tảng này. Chúng em hy vọng rằng những kinh nghiệm và kiến thức tích lũy được sẽ là nền tảng vững chắc cho những nghiên cứu và phát triển trong tương lai.

Cuối cùng, chúng em xin gửi lời cảm ơn sâu sắc đến Th.S Cao Tiến Thành đã tận tình hướng dẫn và hỗ trợ chúng em trong suốt quá trình thực hiện đồ án. Chúng em rất mong nhận được những góp ý quý báu từ thầy để hoàn thiện hơn nữa đề tài này.

## **Tài liệu tham khảo**

- Bartlett, J. (2023). Cloud Native Applications with Docker and Kubernetes: Design and Build Cloud Architecture and Applications with Microservices, EMQ, and Multi-Site Configurations. Apress. <https://doi.org/10.1007/978-1-4842-8876-4>
- Kubernetes Documentation. (2024). Kubernetes Documentation - Supported Documentation Versions. Kubernetes. Retrieved from <https://kubernetes.io/docs/home/supported-doc-versions/>