Sinh viên: Nguyễn Chí Thành

Báo Cáo Nghiên Cứu: So sánh Docker Compose, Docker Swarm và Kubernetes

Contents

[I. Giới Thiệu 2](#_Toc167634599)

[II. Docker Compose 2](#_Toc167634600)

[1. Tổng Quan 2](#_Toc167634601)

[2. Tính Năng 2](#_Toc167634602)

[3. Trường Hợp Sử Dụng 3](#_Toc167634603)

[III. Docker Swarm 3](#_Toc167634604)

[1. Tổng Quan 3](#_Toc167634605)

[2. Tính Năng 3](#_Toc167634606)

[3. Trường Hợp Sử Dụng 4](#_Toc167634607)

[IV. Kubernetes 4](#_Toc167634608)

[1. Tổng Quan 4](#_Toc167634609)

[2. Tính Năng 5](#_Toc167634610)

[3. Trường Hợp Sử Dụng 6](#_Toc167634611)

[V. So Sánh 6](#_Toc167634612)

[1. Dễ Sử Dụng 6](#_Toc167634613)

[2. Khả Năng Mở Rộng 6](#_Toc167634614)

[VI. Kết Luận 7](#_Toc167634615)

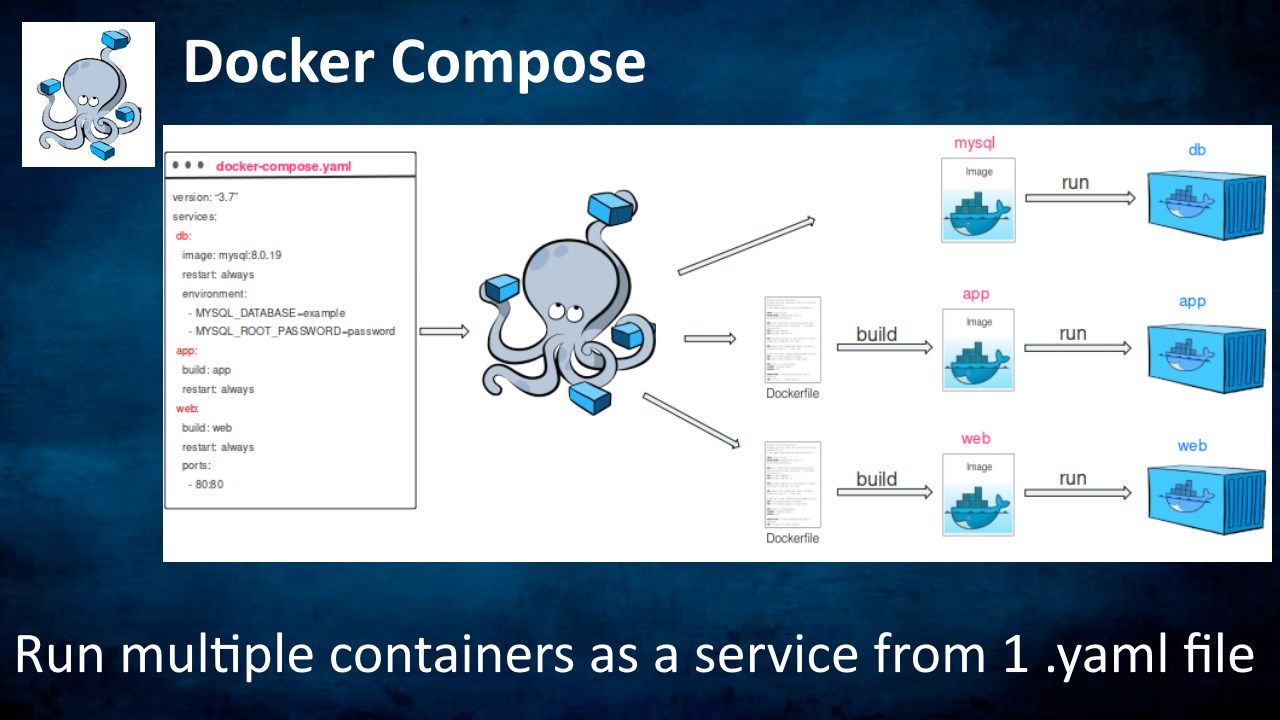
# Giới Thiệu

* Công nghệ container hóa đã cách mạng hóa cách các ứng dụng được phát triển, triển khai và vận hành. Docker Compose, Docker Swarm và Kubernetes là ba công cụ nổi bật trong lĩnh vực điều phối container, mỗi công cụ phục vụ các mục đích và trường hợp sử dụng khác nhau. Trong báo cáo này em sẽ tìm hiểu về các công cụ này, tính năng của chúng, ưu điểm và các trường hợp sử dụng.

# Docker Compose

## Tổng Quan

* Docker Compose là một công cụ được sử dụng để định nghĩa và chạy các ứng dụng Docker đa container. Sử dụng một tệp YAML, người dùng có thể chỉ định các dịch vụ, mạng và volume cần thiết cho ứng dụng của mình.



## Tính Năng

* Cấu hình đa container đơn giản: Docker Compose cho phép người dùng định nghĩa nhiều dịch vụ trong một tệp YAML duy nhất.
* Dễ sử dụng: Được thiết kế cho môi trường phát triển, Docker Compose đơn giản hóa quá trình thiết lập và tháo dỡ các ứng dụng đa container.
* Tích hợp với Docker CLI: Nó tích hợp liền mạch với giao diện dòng lệnh của Docker, giúp dễ sử dụng cho những người đã quen với các lệnh Docker.

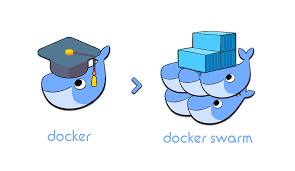
## Trường Hợp Sử Dụng

* Phát triển và kiểm thử: Lý tưởng cho các môi trường phát triển cục bộ nơi một nhà phát triển cần chạy nhiều dịch vụ.
* CI/CD Pipelines: Hữu ích trong các quy trình tích hợp liên tục và triển khai liên tục để kiểm thử các ứng dụng trong môi trường container hóa.

# Docker Swarm

## Tổng Quan

* Docker Swarm là công cụ điều phối và clustering native của Docker cho phép người dùng tạo và quản lý một swarm (một cụm các nút Docker).



## Tính Năng

* Tích hợp với Docker: Là một phần của hệ sinh thái Docker, Swarm tích hợp liền mạch với Docker CLI và các công cụ khác.
* Mô hình theo kiểu declarative: Cho phép người dùng định nghĩa trạng thái mong muốn của dịch vụ và Swarm sẽ quản lý phần còn lại.
* Khả năng mở rộng: Dễ dàng mở rộng hoặc thu nhỏ các dịch vụ với các lệnh đơn giản.
* Cân bằng tải: Tự động phân phối lưu lượng mạng giữa các container trong dịch vụ.

A diagram of a server

Description automatically generated

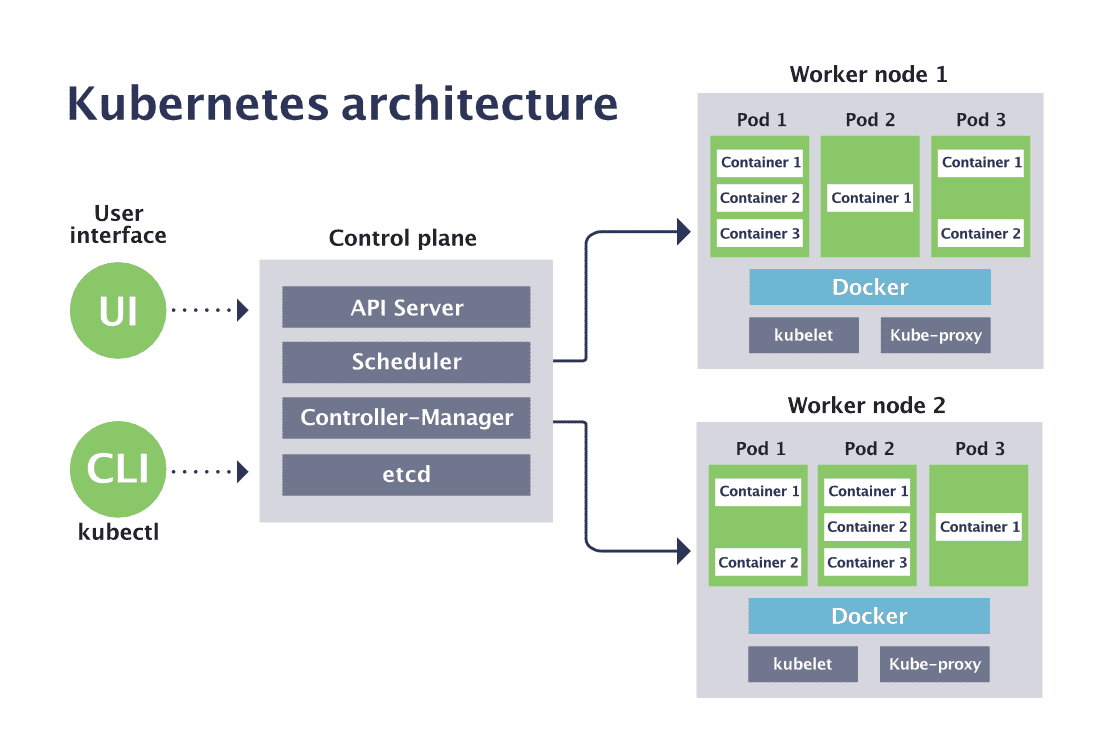
## Trường Hợp Sử Dụng

* Triển khai quy mô nhỏ đến trung bình: Phù hợp cho các cụm nhỏ hơn và các môi trường không quá phức tạp nơi Kubernetes có thể là quá mức cần thiết.
* Nhu cầu điều phối đơn giản: Đối với các nhóm đã sử dụng Docker rộng rãi, Swarm cung cấp một giải pháp điều phối ít phức tạp hơn so với Kubernetes.

# Kubernetes

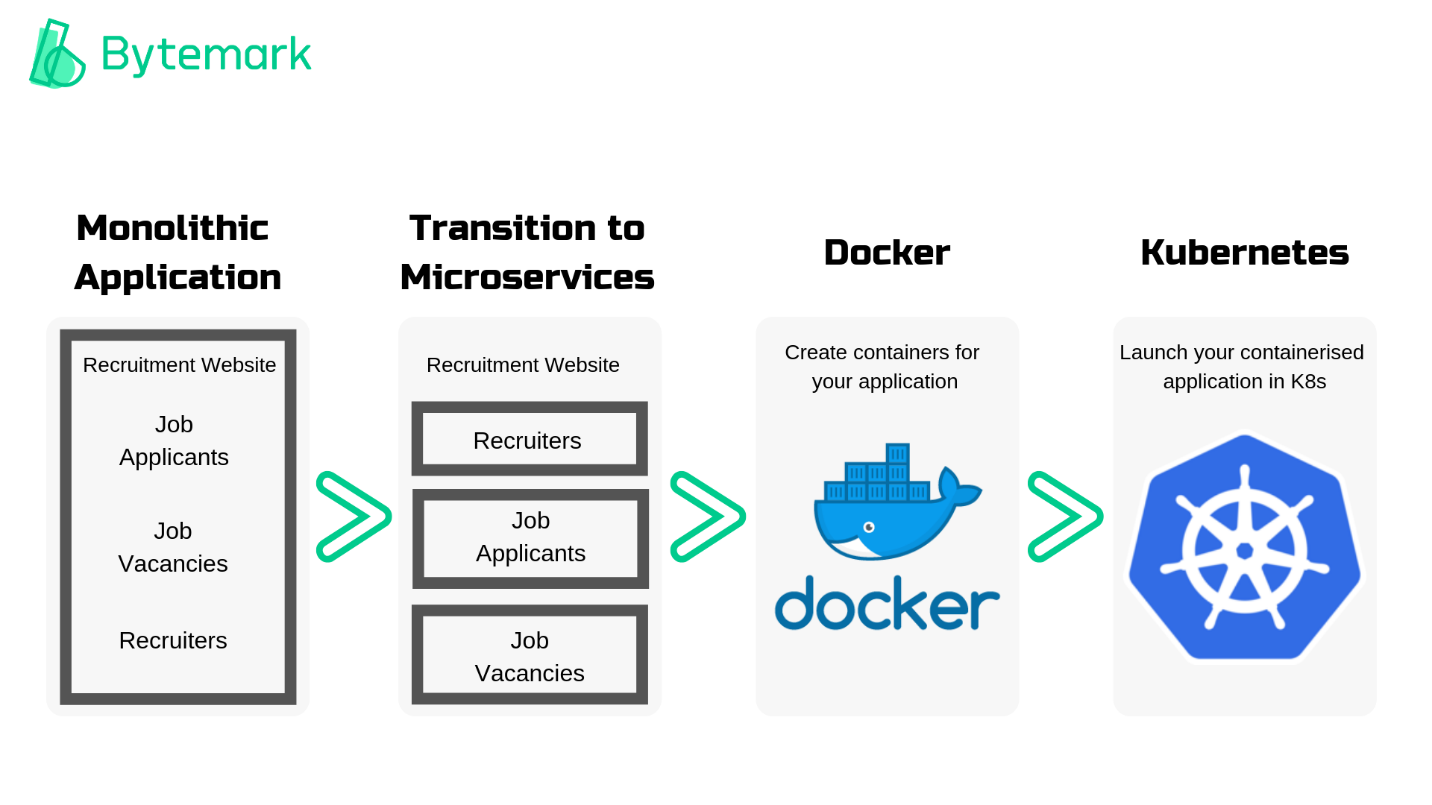
## Tổng Quan

* Kubernetes là một nền tảng điều phối container mã nguồn mở được thiết kế để tự động hóa việc triển khai, mở rộng và vận hành các ứng dụng container hóa.



## Tính Năng

* Khả năng mở rộng cao: Được thiết kế để xử lý các triển khai quy mô lớn, sản xuất.
* Hệ sinh thái phong phú: Được hỗ trợ bởi một cộng đồng lớn và tích hợp với nhiều công cụ và dịch vụ.
* Tự phục hồi: Tự động khởi động lại các container bị lỗi, thay thế và lên lịch lại container khi các nút chết, và tiêu diệt các container không đáp ứng các kiểm tra sức khỏe do người dùng định nghĩa.
* Cập nhật và rollback: Tạo điều kiện cho việc cập nhật ứng dụng mà không có thời gian chết và rollback trong trường hợp xảy ra lỗi.
* Cân bằng tải: Cân bằng tải tự động.



## Trường Hợp Sử Dụng

* Ứng dụng quy mô lớn: Lý tưởng để quản lý các ứng dụng lớn, phức tạp với nhiều dịch vụ và phụ thuộc.
* Kiến trúc microservices: Thích hợp cho các ứng dụng được thiết kế với kiến trúc microservices, nơi mỗi dịch vụ có thể được quản lý và mở rộng độc lập.
* Ứng dụng cloud-native: Được thiết kế để hỗ trợ các ứng dụng hiện đại trên nền tảng đám mây và có thể chạy trên bất kỳ nhà cung cấp đám mây nào.

# So Sánh

## Dễ Sử Dụng

* Docker Compose: Dễ sử dụng nhất, chủ yếu dành cho môi trường phát triển và kiểm thử cục bộ.
* Docker Swarm: Phức tạp vừa phải, phù hợp cho các triển khai quy mô nhỏ đến trung bình.
* Kubernetes: Tốt nhất cho các môi trường lớn, phức tạp.

## Khả Năng Mở Rộng

* Docker Compose: Giới hạn trong một máy host.
* Docker Swarm: Mở rộng tốt nhưng không mạnh mẽ bằng Kubernetes.
* Kubernetes: Được thiết kế để mở rộng lớn và có tính khả dụng cao.

# Kết Luận

* Mỗi công cụ trong số này phục vụ các nhu cầu cụ thể trong không gian điều phối container. Docker Compose tốt nhất cho các môi trường phát triển, Docker Swarm phù hợp cho các môi trường sản xuất nhỏ hơn, ít phức tạp, và Kubernetes là lý tưởng cho các ứng dụng lớn, phức tạp.