AWS Cloud for beginner

Instructor: Linh Nguyen

(Engineering Consultant, AWS Cloud Solution Architect)

Level: Beginner

“Không có việc gì khó, chỉ sợ không biết làm”

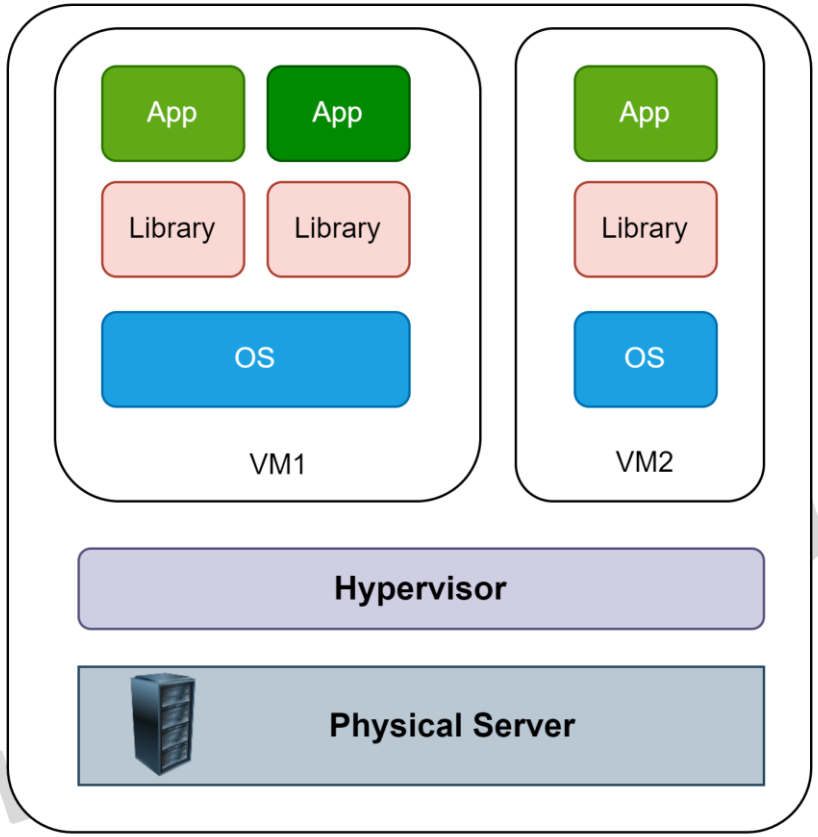
Container Services – ECS, ECR

Target

- Nắm được các khái niệm cơ bản về Container & Docker.
- Hiểu được các dịch vụ cơ bản liên quan container như ECS, ECR.
- Nắm được các thành phần cơ bản của một hệ thống sử dụng container.
- Nắm được các bước cơ bản để xây dựng & triển khai một ứng dụng container.
- Làm quen với ECS và ECR thông qua bài lab đơn giản.

Container là gì? Tại sao cần Container?

Cùng xét một mô hình về việc deployment sử dụng máy ảo (VM) hay các EC2 server.



Virtual Machine Architect

Container là gì? Tại sao cần Container?

Giả sử một ngày đẹp trời bạn được sếp giao nhiệm vụ install một software mới lên server production của công ty, bạn tìm hiểu rất kỹ càng những thư viện, runtime cần thiết của software này và đã tải về đầy đủ.

Đến ngày thực hiện, việc đầu tiên là bạn sẽ backup server (đương nhiên rồi ☺)

Bạn cài đặt một thư viện Jxxx version 20, tuy nhiên trong quá trình cài đặt Jxxx nói rằng không thể tồn tại 2 version song song trên cùng một máy và bạn phải gỡ version cũ hơn ra trước. Bạn gỡ version cũ là Jxxx 19 ra khỏi máy và cài Jxxx 20 vào (mới hơn tất nhiên là tốt hơn).

Sau khi hoàn thành thao tác cài đặt software, bạn thông báo với sếp là đã xong việc.

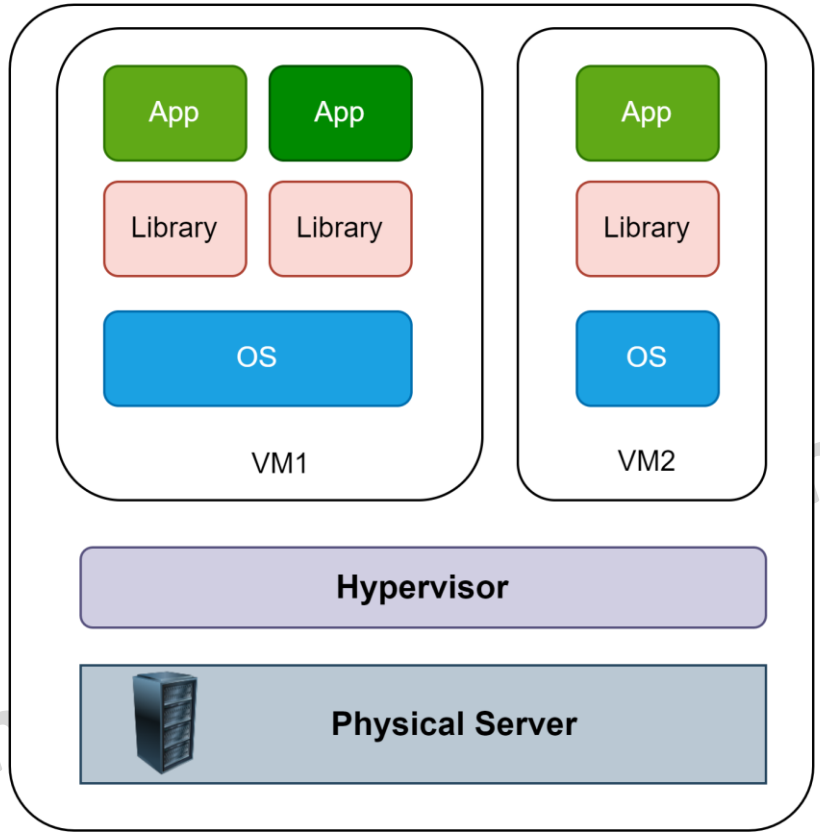
30 phút sau, khách hàng gọi điện và claim về việc một số software trước giờ vẫn sử dụng bình thường thì hôm nay không khởi động lên được và văng lỗi thiếu thư viện.

Và sau đó.... thôi không cần nói nữa... ☹

Container là gì? Tại sao cần Container?

Problem khi sử dụng VM

- Xung đột version giữa Library, Binary, Runtime cùng cài trên các VM.
- Không có khả năng độc lập môi trường giữa các ứng dụng.
- Mất thời gian trong việc triển khai (chuẩn bị OS, cài các library cần thiết, setup môi trường,vv)
- Khó đảm bảo tính nhất quán của ứng dụng được triển khai (ứng dụng work OK trên một môi trường nhưng không chắc sang môi trường khác chạy bình thường).

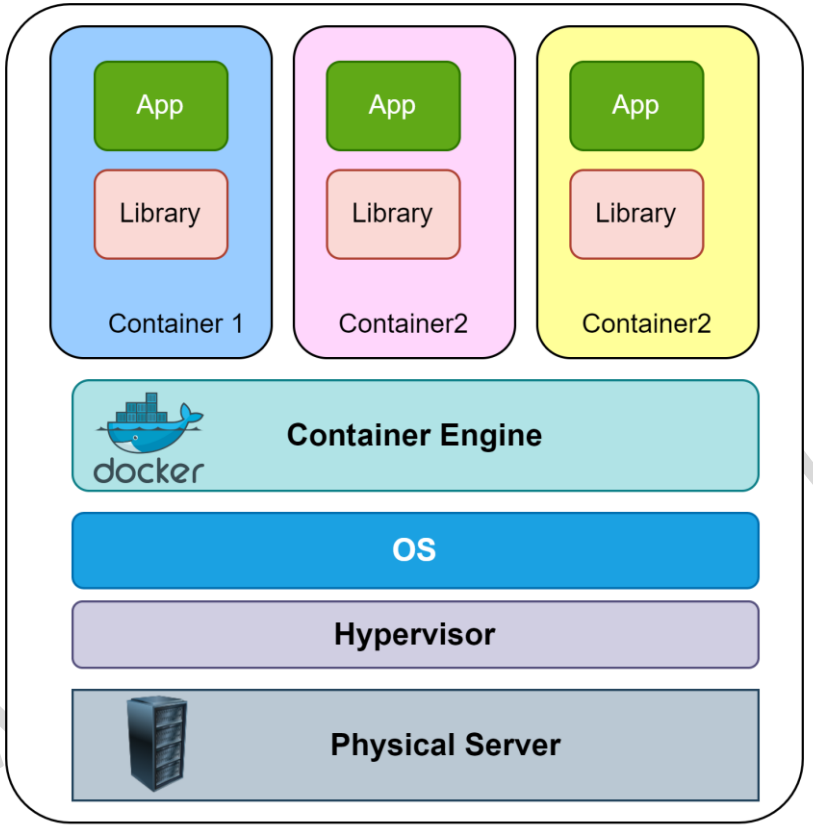


Virtual Machine Architect

Container là gì? Tại sao cần Container?

Container ra đời để giải quyết những vấn đề trên, bằng cách:

- Đóng gói ứng dụng cùng với những thứ cần thiết để chạy được ứng dụng đó thành một image có thể run ở bất cứ đâu có hỗ trợ container.
- Cung cấp môi trường & cơ chế cấp phát tài nguyên để image đó có thể run được.
- Cung cấp cơ chế & công cụ cho phép các nhà phát triển đóng gói, lưu trữ, phân phối và triển khai ứng dụng một cách thuận tiện.



Containers Architect

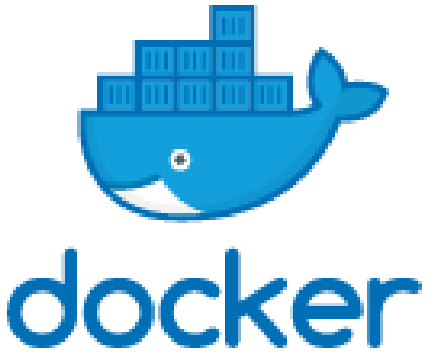
Container là gì? Tại sao cần Container?

Lợi ích của việc sử dụng Container

1. Độc lập với môi trường: Containers cung cấp một cách để đóng gói ứng dụng và tất cả các dependencies của nó, bao gồm OS, libraries, tools. Nó cho phép ứng dụng chạy một cách độc lập và nhất quán trên bất kỳ môi trường nào.
2. Đơn giản hóa quy trình triển khai: tính nhất quán, tốc độ, thuận tiện là những gì Containerlize mang lại khi so sánh với mô hình truyền thống.
3. Quản lý tài nguyên hiệu quả: triển khai ứng dụng bằng Container cho phép bạn chia sẻ và sử dụng tài nguyên của hệ thống một cách hiệu quả. Bằng cách chạy nhiều container trên cùng một máy chủ vật lý hoặc máy ảo, bạn có thể tận dụng tối đa khả năng tính toán và tài nguyên của hệ thống.
4. Linh hoạt và mở rộng: Containers cho phép bạn dễ dàng mở rộng ứng dụng theo nhu cầu. Bằng cách scale horizontal, ứng dụng có thể mở rộng để đáp ứng workload. Ngoài ra, việc triển khai nhiều version của một ứng dụng cùng lúc cũng trở nên dễ dàng.

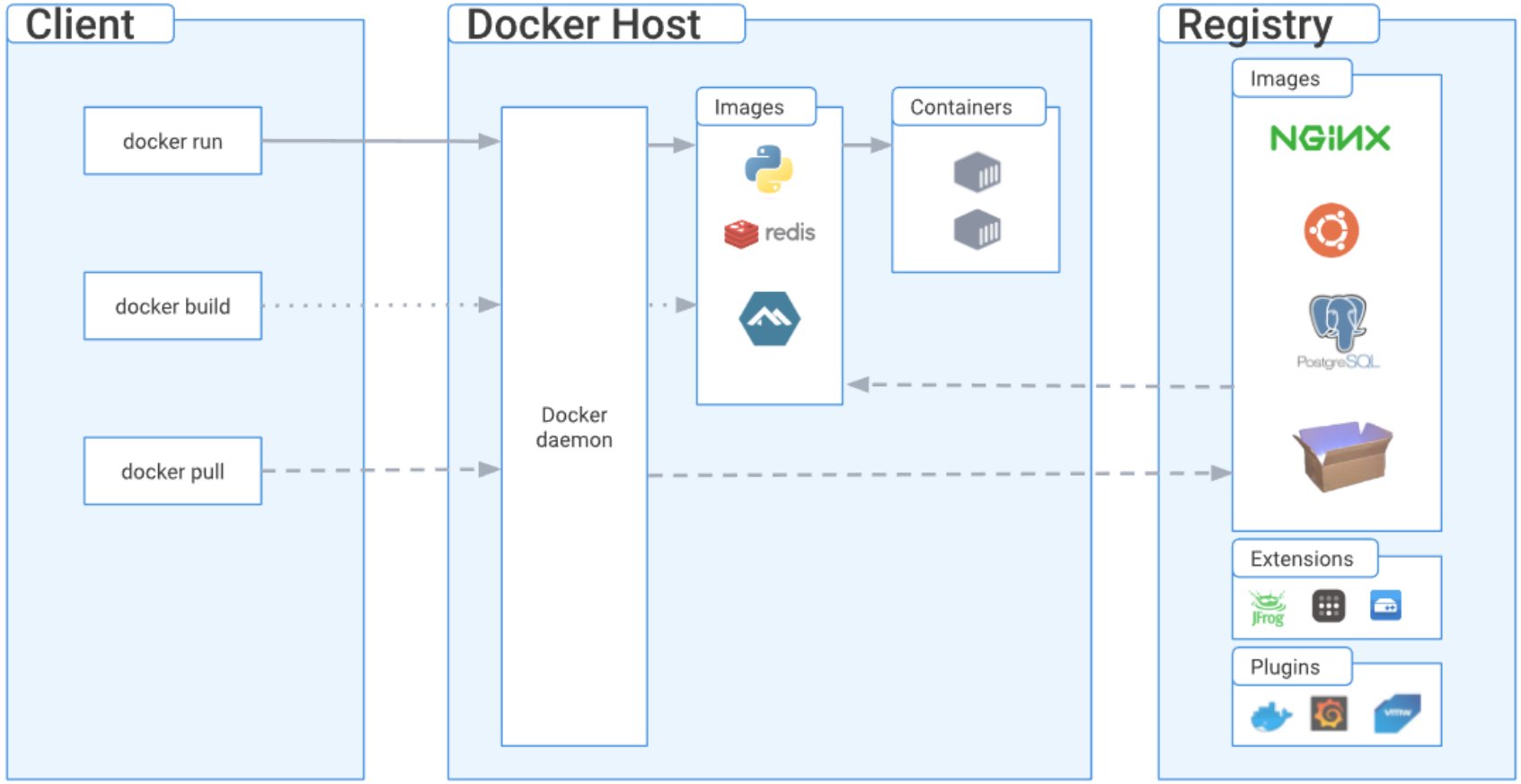
Docker là gì?

Docker là nền tảng phần mềm cho phép bạn dựng, kiểm thử và triển khai ứng dụng một cách nhanh chóng. Docker đóng gói phần mềm vào các đơn vị tiêu chuẩn hóa được gọi là container có mọi thứ mà phần mềm cần để chạy, trong đó có thư viện, công cụ hệ thống, mã và thời gian chạy. Bằng cách sử dụng Docker, bạn có thể nhanh chóng triển khai và thay đổi quy mô ứng dụng vào bất kỳ môi trường nào và biết chắc rằng mã của bạn sẽ chạy được.



*Lưu ý: Docker không phải là nền tảng duy nhất cho phép triển khai ứng dụng dưới dạng container.

Các thành phần cơ bản của Docker



*Nguồn: https://docs.docker.com/get-started/overview/

Các thành phần cơ bản của Docker

- Docker daemon: là nơi quản lý các thành phần của Docker như image, container, volume, network. Docker daemon nhận API từ Client để thực thi các nhiệm vụ.
- Docker Client: Cung cấp phương thức để tương tác với Docker daemon.
- Docker registry: nơi lưu trữ các docker image. Mặc định docker sẽ connect tới docker registry là Docker hub.

Khi bạn cài Docker Desktop, Daemon và Client sẽ cùng nằm trên máy tính của bạn.

Các bước cơ bản để xây dựng ứng dụng docker

1. **Chuẩn bị Dockerfile**: Dockerfile mô tả các bước cần thiết để tạo ra môi trường container chứa ứng dụng của bạn.
2. **Xây dựng Docker Image**: Sử dụng lệnh **docker build** để xây dựng Docker Image từ Dockerfile.
3. **Kiểm tra Docker Imag**e: Sử dụng lệnh **docker images** để kiểm tra danh sách các ảnh Docker có sẵn trên máy tính của bạn. Đảm bảo rằng Docker Image của ứng dụng của bạn đã được xây dựng thành công và xuất hiện trong danh sách.
4. **Chạy Docker Container**: Sử dụng lệnh **docker run** để chạy một container từ Docker Image.
5. **Kiểm tra ứng dụng**: Truy cập ứng dụng của bạn thông qua địa chỉ IP hoặc tên miền cùng với port đã chỉ định.

Trong thực tế khi triển khai lên môi trường Cloud sẽ bao gồm nhiều bước phức tạp hơn, sẽ được trình bày sau.

Lab 1 – làm quen với Docker helloworld!

Yêu cầu đã cài sẵn Docker Desktop (Mac or Windows đều được)

Link: https://docs.docker.com/desktop/install/windows-install/

Yêu cầu đăng ký sẵn tài khoản DockerHub. https://hub.docker.com/signup

1. Run một docker image **hello-world**
2. Kiểm tra output.
3. Giới thiệu các câu lệnh cơ bản của Docker.

Làm quen với Dockerfile

- Khai báo Base image: Dockerfile thường bắt đầu bằng một chỉ thị **FROM** để chỉ định base image mà new Docker image sẽ dựa trên. Ví dụ: **FROM ubuntu:latest.**
- Sao chép các tệp và thư mục: Sử dụng chỉ thị **COPY** hoặc **ADD**, bạn có thể sao chép các tệp và thư mục từ máy chủ nơi Dockerfile được chạy vào bên trong Docker image
  Ví dụ: **COPY app.py /app**.
- Chỉ thị RUN. bạn có thể thực thi các lệnh bên trong hình ảnh Docker để cài đặt phần mềm, cập nhật gói phần mềm hoặc thực hiện các tác vụ khác.
  Ví dụ: **RUN apt-get update && apt-get install -y python**.
- Thiết lập biến môi trường: Bằng cách sử dụng chỉ thị **ENV**, bạn có thể định nghĩa các biến môi trường cho hình ảnh Docker. Ví dụ: **ENV LOGLEVEL=DEBUG**.
- Mở port: Bằng cách sử dụng chỉ thị **EXPOSE**, bạn có thể xác định các cổng mà ứng dụng trong hình ảnh Docker sẽ lắng nghe. Ví dụ: **EXPOSE** 8080.
- Chạy ứng dụng: Bằng cách sử dụng chỉ thị **CMD** bạn có thể chỉ định lệnh mà Docker sẽ chạy khi khởi động một container từ hình ảnh.
  Ví dụ: **CMD ["python", "/app/app.py"]**

```
1  FROM httpd
2
3  COPY index.html /usr/local/apache2/htdocs/
4
5  RUN chown www-data:www-data /usr/local/
   apache2/htdocs/index.html
6  ENV LOGLEVEL=DEBUG
7  EXPOSE 80
8  CMD ["httpd-foreground"]
9
```

Lab 2 – Build and run custom image

Yêu cầu đã cài sẵn Docker Desktop (Mac or Windows đều được)

1. Tạo một Dockerfile với base image **httpd**
2. Build image
3. Kiểm tra image đã được build
4. Chạy một container từ image đã được build

Giới thiệu Elastic Container Registry

Elastic Container Registry (ECR) là một dịch vụ của AWS cung cấp khả năng quản lý và lưu trữ các Docker Image. ECR là một Registry dựa trên cloud computing, được thiết kế đặc biệt để làm việc với các container hóa và môi trường chạy container của AWS.

- **Registry**: đơn vị quản lý của ECR (giống một repository). Thông thường một Registry sẽ chỉ lưu image của một ứng dụng.
- **Image**: Tương tự Docker Image. Các Image trên Registry cần đánh tag để quản lý.

Amazon Elastic Container Registry (Amazon ECR)

Registry

Image

Giới thiệu Elastic Container Service

Elastic Container Service (ECS) là một dịch vụ quản lý container đám mây do Amazon Web Services (AWS) cung cấp. Nó cho phép bạn chạy và quản lý các ứng dụng container trên nền tảng AWS một cách dễ dàng và linh hoạt.

Amazon Elastic Container Service (Amazon ECS)

Giới thiệu Elastic Container Service

Amazon Elastic Container Service (Amazon ECS)

Các tính năng của ECS

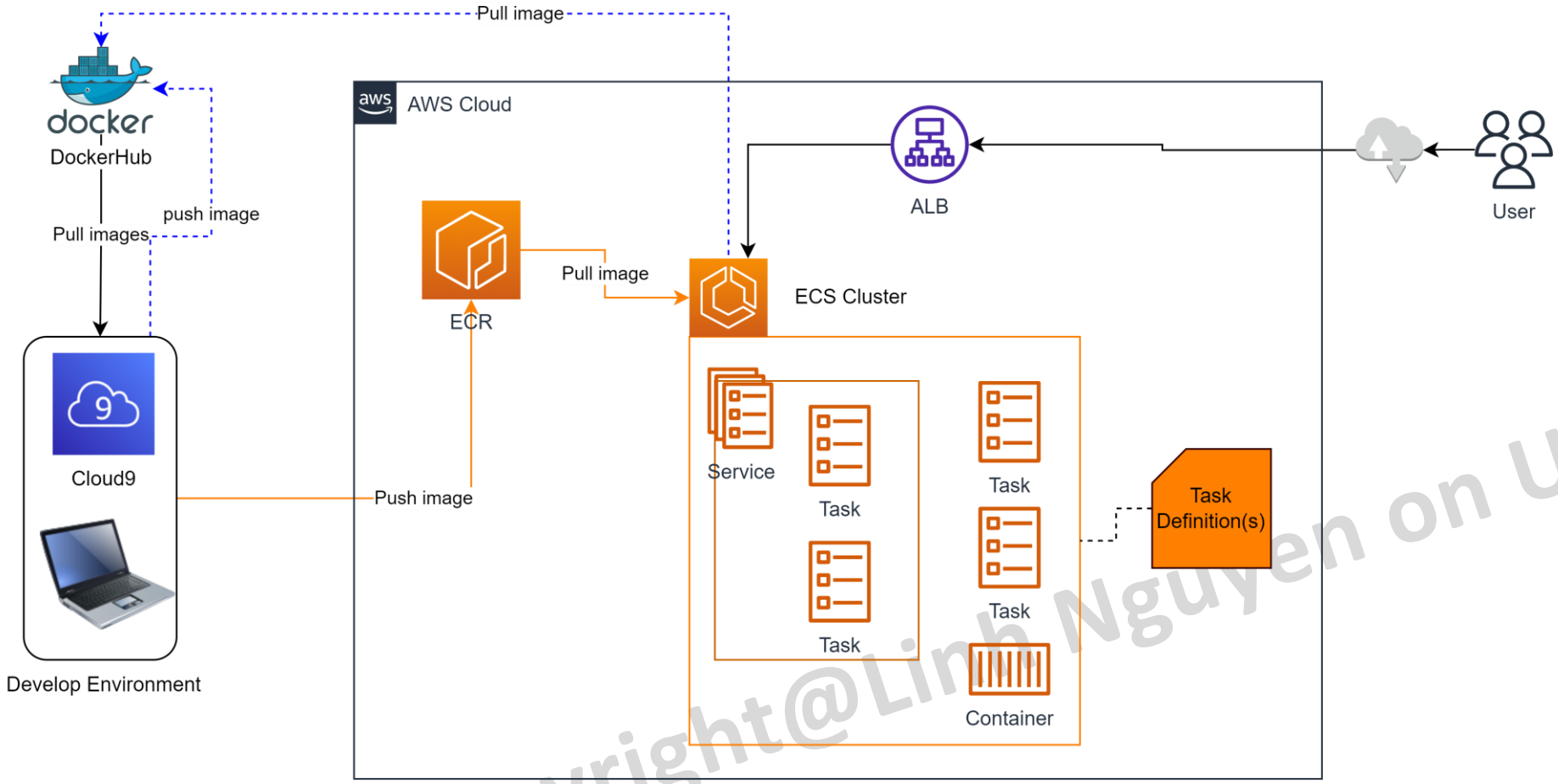
1. Quản lý đơn giản.
2. Tích hợp với công cụ container: ECS tích hợp tốt với Docker, cho phép bạn chạy các container Docker trực tiếp trên nền tảng AWS mà không cần thay đổi mã nguồn hoặc cấu hình.
3. Mở rộng linh hoạt: Kết hợp với AutoScaling, ECS cho phép scale in-out linh hoạt dựa trên nhu cầu workload.
4. Tích hợp với các dịch vụ AWS khác: ECS tích hợp tốt với các dịch vụ AWS khác như Elastic Load Balancer (ELB), Elastic Container Registry (ECR), IAM, CloudWatch và nhiều dịch vụ khác.
5. Sự linh hoạt về kiến trúc: ECS hỗ trợ hai kiểu triển khai: EC2 Launch Type và Fargate Launch Type. Giúp bạn dễ dàng lựa chọn dựa theo nhu cầu cũng như khả năng customize của đội dự án.

Giới thiệu Elastic Container Service

Các thành phần của ECS

- Cluster: Đơn vị lớn nhất của ECS, có nhiệm vụ cung cấp tài nguyên cần thiết (EC2, Fargate) để chạy ứng dụng.
- Task: Một đơn vị được cấp phát tài nguyên (CPU,RAM) trong mode Fargate. Một task có thể chứa 1 hoặc nhiều container.
- Service: Một nhóm các Task có chung nhiệm vụ được expose ra bên ngoài hoặc nội bộ cluster.
- Container: tương tự Docker Container, một runable image.
- ECS Connect Service: Cung cấp cơ chế service-to-service communication
- Task Definition (không có trong hình): chỉ dẫn để ECS biết phải tạo một task như thế nào.

Sample hệ thống triển khai trên sd ECS, ECR



Lưu ý: các tool như Jenkins, các service phục vụ deployment CICD chưa thể hiện trong hình.

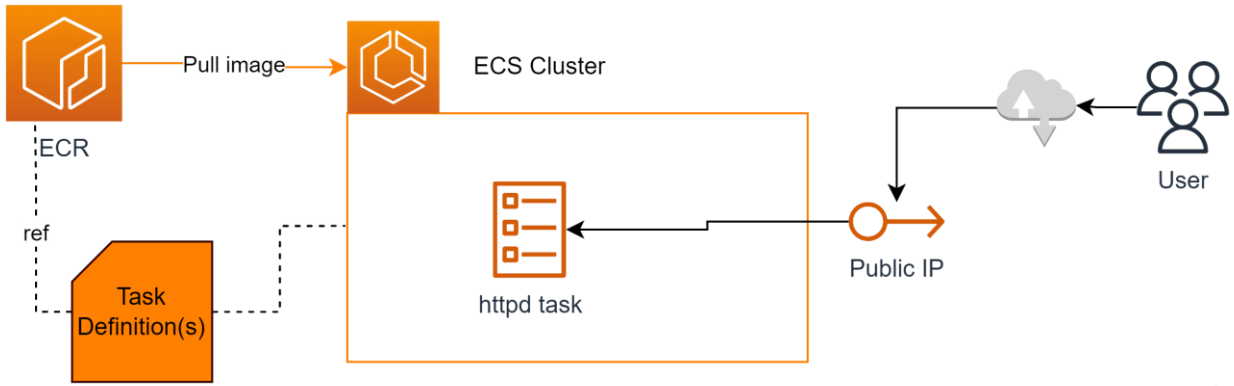
Lab 3 – Tạo và push image lên ECR

Yêu cầu: Build một custom image từ httpd, copy file index.html vào image. Tag và push image đó lên ECR.

1. Tạo một ECR repository vd: **my-httpd**
2. Tạo một **Dockerfile** theo mẫu cung cấp sẵn.
3. Build, tag image, push lên ECR theo chỉ dẫn.
4. Kiểm tra image đã được tạo ra trên ECR repository.

Lab 4 – ECS Cluster - Run task

Yêu cầu: Sử dụng docker image đã build và push lên ECR từ bài lab trước, chạy một task trên ECS cluster, test truy cập.



Sơ đồ hệ thống

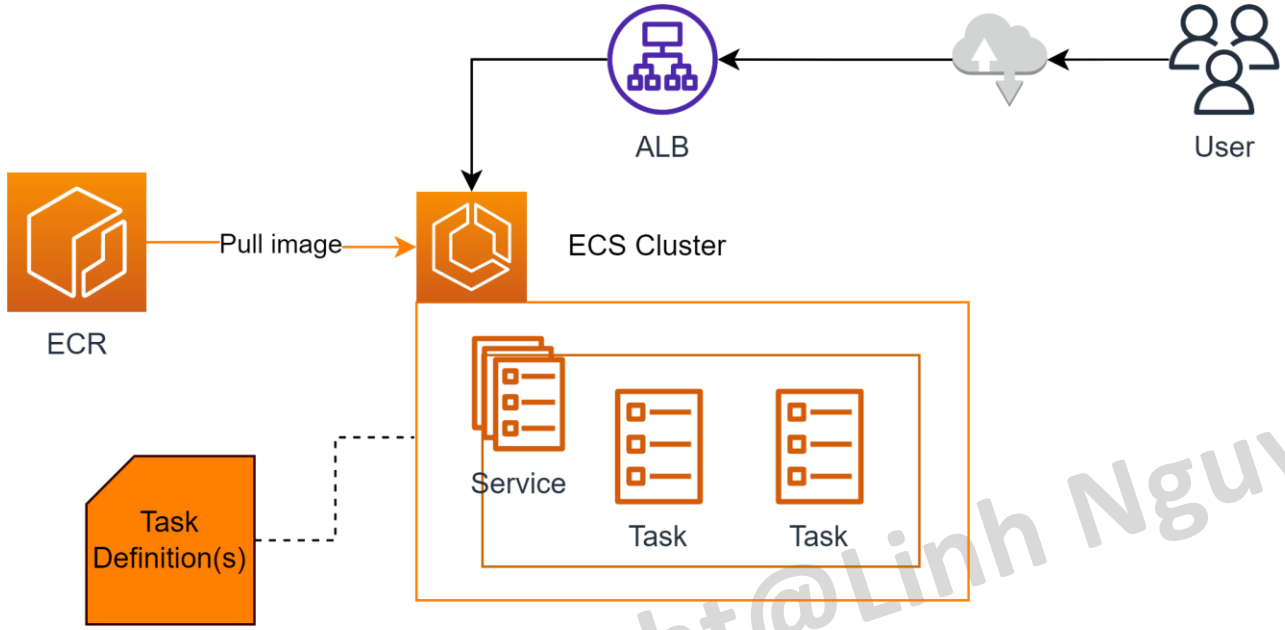
Lab 4 – ECS Cluster - Run task

Steps:

1. Tạo một ECS Cluster với mode Fargate
2. Tạo một task definition với các thông số cơ bản.
3. Truy cập vào Cluster, run một task với task definition đã tạo. Lưu ý enable public IP để có thể test truy cập.
4. Cấu hình security group nếu cần.
5. Thử truy cập tới public IP của Task (lưu ý số port).

Lab 5 – ECS Cluster - Run Service

Yêu cầu: Tạo một service https thay vì chạy task đơn lẻ, kết nối với ALB để có thể truy cập từ bên ngoài. Test việc điều chỉnh capacity của service.



Sơ đồ hệ thống

Lab 4 – ECS Cluster - Run Service

Steps:

1. Tạo task definition (có thể tái sử dụng ở bài lab trước).
2. Tạo một Target group với type là IP, không add bất cứ gì.
3. Tạo một Application Load Balancer có listener 80 trỏ tới Target Group
4. Tạo một service với task definition đã tạo, trong lúc tạo nhớ chọn ALB và Target Group tương ứng, enable Public IP (nếu không có NAT).
5. Đợi Service launch task thành công, test truy cập thông qua ALB.
6. Thử điều chỉnh số lượng task trong service (scale).

Giới thiệu Cloud Map

- AWS Cloud Map là một “Cloud Service Discovery”. Với Cloud Map, bạn có thể xác định tên tùy chỉnh cho các tài nguyên ứng dụng của mình và dịch vụ này duy trì vị trí cập nhật của những tài nguyên thay đổi linh hoạt đó. Điều này làm tăng tính sẵn sàng của ứng dụng vì dịch vụ web của bạn luôn tìm ra vị trí mới nhất của các nguồn tài nguyên.
- Cloud Map cho phép bạn đăng ký bất kỳ tài nguyên ứng dụng nào, chẳng hạn như DB, queue, micro service và cloud resource khác, với tên tùy chỉnh. Sau đó, Cloud Map liên tục kiểm tra tình trạng của tài nguyên để đảm bảo vị trí được cập nhật. Sau đó, ứng dụng có thể query đến registry để biết vị trí của các tài nguyên cần thiết dựa trên phiên bản ứng dụng và môi trường triển khai.
- Một trong những ứng dụng phổ biến của AWS Cloud Map là trong kiến trúc dựa trên microservices. Bằng cách sử dụng Cloud Map, bạn có thể quản lý và theo dõi các dịch vụ ứng dụng microservice một cách linh hoạt và tự động.

AWS Cloud Map

Giới thiệu Cloud Map

Giới thiệu Cloud Map trên AWS console

AWS Cloud Map

Tổng kết

- Container là gì? tại sao cần dùng Container.
- Các thành phần cơ bản của Docker cũng như các ứng dụng Container.
- Các bước cơ bản để build và triển khai một ứng dụng Container.
- Làm quen với ECR, ECS.
- Biết cách thao tác với ECR, ECS ở mức độ cơ bản.

Clear resources

1. Login to AWS console.
2. Stop toàn bộ ECS task.
3. Xoá toàn bộ ECS service
4. Xoá ECS Cluster.
5. Xoá Application Load Balancer.
6. Xoá Repository trên ECR.
7. Xoá Cloud9 environment.