BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC DÂN LẬP HẢI PHÒNG

VŨ TRỌNG CHIẾN

CÔNG NGHỆ ẢO HÓA DOCKER VÀ ỨNG DỤNG TẠI ĐẠI HỌC DÂN LẬP HẢI PHÒNG

LUẬN VĂN THẠC SĨ
NGÀNH CÔNG NGHỆ THÔNG TIN

CHUYÊN NGÀNH: HỆ THỐNG THÔNG TIN

MÃ SỐ: 60 48 01 04

NGƯỜI HƯỚNG DẪN KHOA HỌC:
TS. LÊ VĂN PHÙNG

LỜI CẢM ƠN

Để có thể hoàn thành đề tài luận văn thạc sĩ một cách hoàn chỉnh, bên cạnh sự nỗ lực cố gắng của bản thân còn có sự hướng dẫn nhiệt tình của quý Thầy Cô, cũng như sự động viên ủng hộ của gia đình và bạn bè trong suốt thời gian học tập nghiên cứu và thực hiện luận văn thạc sĩ.

Xin chân thành bày tỏ lòng biết ơn đến Thầy TS. Lê Văn Phùng người đã hết lòng giúp đỡ và tạo mọi điều kiện tốt nhất cho tôi hoàn thành luận văn này. Xin chân thành bày tỏ lòng biết ơn đến toàn thể quý thầy cô trong khoa Công Nghệ Thông Tin – Trường Đại học Dân lập Hải Phòng đã tận tình truyền đạt những kiến thức quý báu cũng như tạo mọi điều kiện thuận lợi nhất cho tôi trong suốt quá trình học tập nghiên cứu và cho đến khi thực hiện đề tài luận văn.

Xin chân thành bày tỏ lòng biết ơn đến Trung tâm Thông tin Thư viên – Trường Đại học Dân lập Hải Phòng đã không ngừng hỗ trợ và tạo mọi điều kiện tốt nhất cho tôi trong suốt thời gian nghiên cứu và thực hiện luận văn.

Cuối cùng, tôi xin chân thành cảm ơn đến gia đình, các anh chị và các bạn đồng nghiệp đã hỗ trợ cho tôi rất nhiều trong suốt quá trình học tập, nghiên cứu và thực hiện đề tài luận văn thạc sĩ một cách hoàn chỉnh.

Hải Phòng, ngày 05 tháng 12 năm 2017

Học viên thực hiện

Vũ Trọng Chiến

LỜI CAM ĐOAN

Tôi xin cam đoan mọi kết quả của đề tài: "Công nghệ ảo hóa Docker và ứng dụng tại Đại học Dân lập Hải Phòng" là công trình nghiên cứu của cá nhân tôi và chưa từng được công bố trong bất cứ công trình khoa học nào khác cho tới thời điểm này.

Hải Phòng, ngày 05 tháng 12 năm 2017

Tác giả luận văn

Vũ Trọng Chiến

Mục lục

DANH MỤC VIẾT TẮT

AD: Active Directory

CPU: Center processing unit

DHCP: Dynamic Host Configuration Protocol

DNS: Domain Name System

DPM: Distributed Power Manager

DRS: Distributed Resource Scheduler

HA: High Availability

HDD: Hard Disk Drive

IP: Internet Protocol

LAN: Local Area Network.

MAC: Medium Access Control

NIS: Network Information Server

NLB: Network Load Balancing

NTFS: New Technology File System

OS: Operation System

RAID : Redundant Arrays of Independent Disks

RAM: Random Access Memory

SAN: Storeage Area Network

SRM: Site Recovery Manager

SSH: Secure Shell

TCP/IP: Transmission Control Protocol and Internet

UDP: User Datagram Protocol

VCB: VMware Consolidated Backup

VLAN: Virtual LAN

VMFS: Virtual Machine File System

VSMP: Virtual Symmetric Multi-Processing

DANH MỤC HÌNH

MỞ ĐẦU

1. Tính cấp thiết của đề tài

Một số trung tâm dữ liệu chỉ sử dụng 10% đến 30% năng lực xử lý hiện có của họ. Ảo hóa đã giúp nhiều tổ chức có thể chia sẻ các tài nguyên công nghệ thông tin theo cách tốn ít giá thành nhất, làm cho cơ sở hạ tầng công nghệ thông tin trở nên linh động và bảo đảm cung cấp một cách tự động với những nhu cầu cần thiết. Các doanh nghiệp luôn tìm giải pháp để tiết kiệm hơn, đây cũng là lúc công nghệ ảo hóa tìm được chỗ đứng vững chắc trong lĩnh vực công nghệ thông tin trên thế giới.

Sử dụng công nghệ ảo hóa đã đem đến cho người dùng sự tiện ích, có thể chạy nhiều hệ điều hành, nhiều hệ thống đồng thời trên cùng một hệ thống phần cứng máy chủ, mở rộng khả năng lưu trữ, cung cấp tài nguyên phần cứng. Khả năng và lợi ích của ảo hoá còn hơn thế và nơi thành công và tạo nên thương hiệu của công nghệ ảo hóa đó chính là trong môi trường hệ thống máy chủ ứng dụng và hệ thống mạng.

Hiện nay có nhiều nhà cung cấp các sản phẩm máy chủ và phần mềm đều chú tâm đầu tư nghiên cứu và phát triển công nghệ này như là HP, IBM, Microsoft và VMware. Tại Việt Nam, ảo hóa máy chủ ngày càng được quan tâm, nhiều vấn đề về công nghệ ảo hóa đã được nghiên cứu và áp dụng thực tế, như là ảo hóa máy chủ ở mức cơ sở hạ tầng (IaaS). Tuy nhiên việc ảo hóa ứng dụng, do còn nhiều vấn đề về công nghệ và người dùng chưa thực sự quan tâm tới lợi ích và còn thiếu một đội ngũ am hiểu về công nghệ này nên việc áp dụng nó vào hệ thống là rất dè dặt.

Công nghệ ảo hóa Docker được đề cập vào năm 2013, được đánh giá là một công nghệ ảo hóa ứng dụng tương lai cho Linux, đến tháng 8/2014 ra mắt Docker Engine 1.2, và tháng 1/2016 đã công bố Docker Cloud. Docker đưa ra một giải pháp mới cho vấn đề ảo hóa, thay vì tạo ra các máy ảo con chạy độc lập kiểu hypervisors (tạo phần cứng ảo và cài đặt hệ điều hành lên đó), các ứng dụng sẽ được đóng gói lại thành các Container riêng lẻ. Các Container này chạy chung trên nhân hệ điều hành qua LXC (Linux Containers), chia sẻ chung tài nguyên của máy mẹ, do đó, hoạt động nhẹ và nhanh hơn các máy ảo dạng hypervisors.

Công nghệ ảo hóa Docker là công nghệ mới, có khả năng phát triển mạnh mẽ trong tương lai, Đó cũng là lý do mà em chọn đề tài "Công nghệ ảo hóa docker và ứng dụng tại Đại học Dân lập Hải phòng", đề tài giới thiệu được cái nhìn tổng quan về công nghệ này, đồng thời đưa ra những giải pháp, cách thức cơ bản để ứng dụng

công nghệ này cho một mô hình ảo hóa ứng dụng quy mô nhỏ tại trường Đại học Dân lập Hải Phòng.

2. Đối tượng và phạm vi nghiên cứu

Đối tượng nghiên cứu: Công nghệ ảo hóa.

Phạm vi nghiên cứu: Công nghệ Docker để ảo hóa ứng dụng tại Trường Đại học Dân lập Hải Phòng .

3. Hướng nghiên cứu của đề tài

Nghiên cứu về các loại ảo hóa, các công nghệ ảo hóa hệ thống, và nghiên cứu phần mềm tự do nguồn mở trong ảo hóa.

Nghiên cứu sâu về công nghệ Docker, ảo hóa ứng dụng với phần mềm tự do nguồn mở docker.

Ứng dụng công nghệ Docker để ảo hóa ứng dụng tại ĐH Dân lập Hải Phòng.

4. Phương pháp nghiên cứu

- Sưu tập và tổng hợp các nguồn tư liệu đã xuất bản, các tư liệu liên quan về vấn đề ảo hóa và khả năng ứng dụng ảo hóa trong môi trường đào tạo.
- Nghiên cứu thực nghiệm: phân tích thiết kế và cài đặt phần mềm, kiểm tra và đánh giá kết quả thử nghiệm.

5. Cấu trúc luận văn

Ngoài phần mở đầu và kết luận, luận văn chia làm 3 chương:

Chương 1- Tổng quan về ảo hóa và phần mềm tự do nguồn mở

Chương 2- Công nghệ ảo hóa Docker

Chương 3- Ứng dụng công nghệ Docker để ảo hóa ứng dụng tại ĐHDL Hải Phòng

CHƯƠNG 1
TỔNG QUAN VỀ ẢO HÓA VÀ PHẦN MỀM TỰ DO NGUỒN MỞ

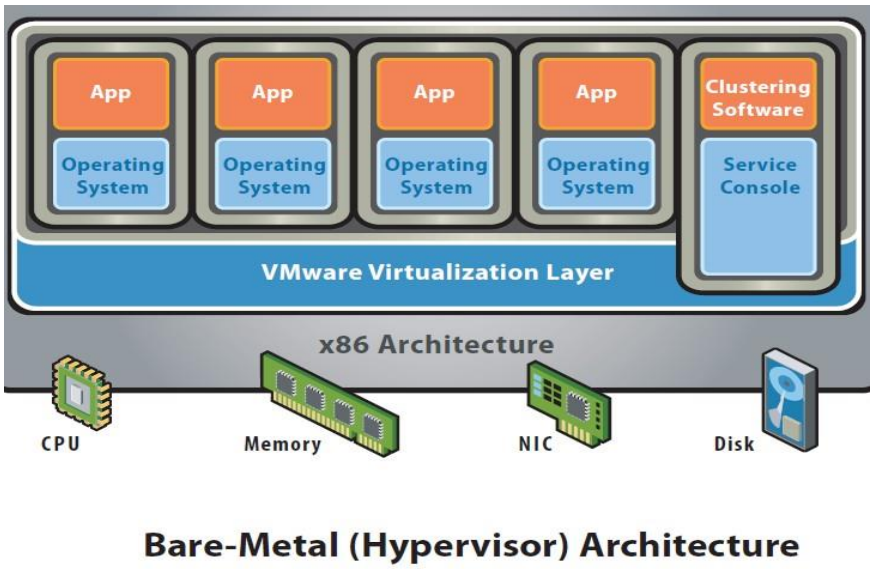
1.1.Tổng quan về ảo hóa

1.1.1. Vấn đề ảo hóa

1.1.1.1. Khái niệm về ảo hóa

Ảo hóa là việc chia phần cứng vật lý thành nhiều phần cứng ảo. Vì vậy, có thể nói ảo hóa là việc chia một máy vật lý thành nhiều máy con ảo.

Công nghệ ảo hóa là một công nghệ thực hiện ảo hóa trên máy tính, bao gồm các kỹ thuật và quy trình thực hiện ảo hóa. Các kỹ thuật và quy trình này để tạo ra một tầng trung gian giữa hệ thống phần cứng máy tính và phần mềm chạy trên nó. Ý tưởng ban đầu của công nghệ ảo hóa là từ một máy vật lý đơn lẻ có thể tạo thành nhiều máy ảo độc lập. Nó cho phép tạo nhiều máy ảo trên một máy chủ vật lý, mỗi một máy ảo cũng được cấp phát tài nguyên phần cứng như máy thật gồm có RAM, CPU, Card mạng, ổ cứng, các tài nguyên khác và hệ điều hành riêng. Khi chạy ứng dụng, người sử dụng không nhận biết được ứng dụng đó chạy trên lớp phần cứng ảo. người sử dụng chỉ chú ý tới khái niệm logic về tài nguyên máy tính hơn là khái niệm vật lí về tài nguyên máy tính[6].



Hình 1. 1: : Kiến trúc x86 Virtuallization

Máy chủ trong các hệ thống CNTT ngày nay thường được thiết kế để chạy một hệ điều hành và một ứng dụng. Điều này không khai thác triệt để hiệu năng của hầu hết các máy chủ rất lớn. Ảo hóa cho phép ta vận hành nhiều máy chủ ảo trên cùng một máy chủ vật lý, dùng chung các tài nguyên của một máy chủ vật lý qua nhiều môi trường khác nhau. Các máy chủ ảo khác nhau có thể vận hành nhiều hệ điều hành và ứng dụng khác nhau trên cùng một máy chủ vật lý.

1.1.1.2. Lịch sử phát triển

Kỹ thuật ảo hóa đã không còn xa lạ kể từ khi Vmware giới thiệu sản phẩm Vmware Workstation đầu tiên vào năm 1999. Sản phẩm này ban đầu được thiết kế để hỗ trợ việc phát triển và kiểm tra phần mềm. Nó đã trở lên phổ biến nhờ khả năng tạo ra những máy tính "ảo" chạy đồng thời nhiều hệ điều hành khác nhau trên cùng một máy tính "thực"(khác với chế độ "khởi động kép " - máy tính được cải nhiều hệ điều hành và có thể chọn lúc khởi động nhưng mỗi lúc chỉ làm việc được với một hệ diều hành).

Vmware đã được EMC – hãng chuyên về lĩnh vực thiết bị lưu trữ mua lại vào tháng 12 năm 2003. EMC đã mở rộng tầm hoạt động lĩnh vực ảo hóa từ máy tính để bàn đến máy chủ và hiện hãng vẫn giữ vai trò thống lĩnh thị trường ảo hóa, tuy nhiên Vmware không giữ vị trí "độc tôn" mà phải cạnh tranh với rất nhiề sản phẩm ảo hóa các hãng khách như Virtualization Engine của IBM, Hyper V – Microsoft, Virtuozzo của SWSoft và vitual iron của iron software… và ảo hóa cũng không còn bó hệp trong một lĩnh vực mà đã mở rộng cho toàn bộ hạ tầng công nghệ thông tin, từ phần cứng như chip xử lý cho đến hệ thống máy chủ và cả hệ thống mạng.

Hiện nay, Vmware là hãng đẫn đầu thị trường ảo hóa nhưng không phải là hãng tiên phong, vai trò thuộc về IBM với hệ thống ảo hóa VM/370 nổi tiếng được công bố vào năm 1972 và "ảo hóa" vẫn đang hiện diện trong các hệ thống máy chủ của IBM.

Giữa năm 1960, IBM's Cambridge Scientific Center đã tiến hành phát triển sản phẩm CP-40, sản phẩm đầu tiên của dòng CP/CMS. Nó được chính thức đưa vào sản xuất vào tháng 1 năm 1967. Ngay từ khi thiết kế CP-40 đã đặt mục đích phải sử dụng ảo hóa đầy đủ. Để làm được vấn đề này nó yêu cầu phần cứng và đoạn mã của

S/360-40 phải kết hợp hoàn chỉnh với nhau, nó phải cung cấp cách truy cập địa chỉ vung nhớ, tập lệnh CPU và các tính năng ảo hóa.

Năm 1970 IBM công bố sản phẩm System 370. Nhưng điều khiến người dùng thất vọng nhất về sản phẩm này do nó không có tính năng Virtial memory.

Vào tháng 8 năm 1999, Vmware giới thiệu sản phẩm ảo hóa đầu tiên hoạt dộng trên nên tảng x86. Vmware Virtual Platform…

Trước đây chúng ta phải mất tiền mua bản quyền sử dụng của Vmware's Workstion. Nhưng năm 2005 Vmware đã quyết định cung cấp sản phẩm ảo hóa chất lượng cao cho người dùng miễn phí. Tuy nhiên chức năng tạo máy chủ ảo và các tính năng phụ khác nhằm mục đích tăng hiệu suất sử dụng máy ảo đã bị lược bỏ.

Năm 2006 đây là năm ảo hóa có một bước tiến mới trong quá trình phát triển, đó là sự ra dời của Application Virtualization và Application Streaming.

Năm 2008, Vmware giới thiệu phiên bản Vmware workstation 6.5 beta, sản phẩm đầu tiên cho phép các chương trình cuả windows và linux được sử dụng Direct X9 để tăng tốc xử lý hình ảnh trong máy ảo Windows XP [9].

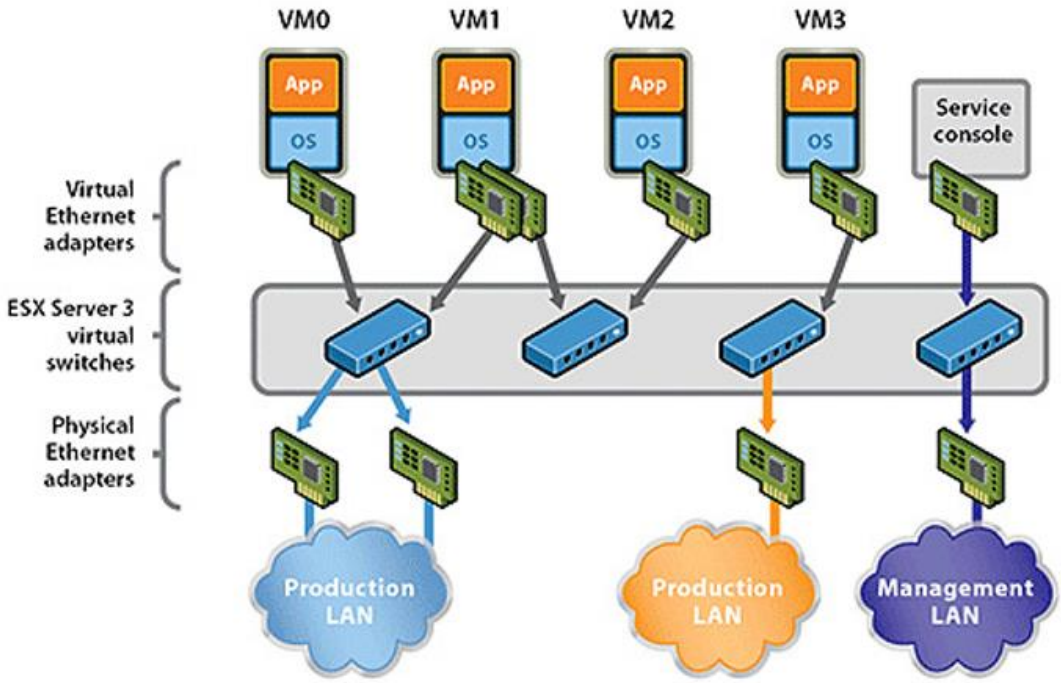
1.2. 1. Phân loại ảo hóa

1.1.2.1. Ảo hóa mạng

Ảo hóa hệ thống mạng là một tiến trình hợp nhất tài nguyên, thiết bị mạng cả phần cứng lẫn phần mềm thành một hệ thống mạng ảo. Sau đó, các tài nguyên này sẽ được phân chia thành các channel và gắn với một máy chủ hoặc một thiết bị nào đó [15].

Các thành phần mạng trong cơ sở hạ tầng mạng như Switch, Card mạng, được ảo hoá một cách linh động. Switch ảo cho phép các máy ảo trên cùng một máy chủ có thể giao tiếp với nhau bằng cách sử dụng các giao thức tương tự như trên thiết bị chuyển mạch vật lý mà không cần phần cứng bổ sung. Chúng cũng hỗ trợ VLAN tương thích với việc triển khai VLAN theo tiêu chuẩn từ nhà cung cấp khác, chẳng hạn như Cisco.

Một máy ảo có thể có nhiều card mạng ảo, việc tạo các card mạng ảo này rất đơn giản và không giới hạn số card mạng tạo ra.Ta có thể nối các máy ảo này lại với nhau bằng một Switch ảo. Điều đặc biệt quan trọng, tốc độ truyền giữa các máy ảo này với nhau thông qua các switch ảo được truyền với tốt độ rất cao theo chuẩn GIGABITE(1GB), đẫn đến việc đồng bộ giữa các máy ảo với nhau diễn ra rất nhanh.



Hình 1. 2: : Ảo hóa network

1.1.2.2 .Ảo hóa lưu trữ

Ảo hóa hệ thống lưu trữ về cơ bản là sự mô phỏng, giả lập việc lưu trữ từ các thiết bị lưu trữ vật lý. Các thiết bị này có thể là băng từ, ổ cứng hay kết hợp cả 2 loại. Việc làm này mang lại các ích lợi như việc tăng tốc khả năng truy xuất dữ liệu, do việc phân chia các tác vụ đọc, viết trong mạng lưu trữ. Ngoài ra, việc môt phỏng các thiết bị lưu trữ vật lý cho phép tiết kiệm thời gian hơn thay vì phải định vị xem máy chủ nào hoạt động trên ổ cứng nào để truy xuất [15].

Hiện nay các nhà lưu trữ đã cung cấp giải pháp lưu trữ hiệu suất cao cho khách hàng của họ trong một thời gian kha khá. Trong hình thức cơ bản nhất của nó, lưu trữ ảo hóa tồn tại trong việc ta lắp ráp nhiều ổ đĩa vật lý thành một thực thể duy nhất để

các máy chủ lưu trữ và chạy hệ điều hành chẳng hạn như triển khai RAID. Điều này có thể được coi là ảo bởi vì tất cả các ổ đĩa được sử dụng và tương tác như một ổ đĩa logic duy nhất, mặc dù bao gồm hai hoặc nhiều ổ đĩa trong.

Một công nghệ ảo hoá lưu trữ khá nổi bật là SAN (Storeage Area Network – lưu trữ qua mạng). SAN là một mạng được thiết kế cho việc thêm các thiết bị lưu trữ cho máy chủ một cách dễ dàng như: Disk Aray Controllers, hay Tape Libraries.

Với những ưu điểm nổi trội SAN đã trở thành một giải pháp rất tốt cho việc lưu trữ thông tin của doanh nghiệp hay tổ chức. SAN cho phép kết nối từ xa tới các thiết bị lưu trữ trên mạng như: Disks và Tape drivers. Các thiết bị lưu trữ trên mạng, hay các ứng dụng chạy trên đó được thể hiện trên máy chủ như một thiết bị của máy chủ (as locally attached divices).

Có hai đặc điểm cơ bản trong các thành phần của SAN:

1-Mạng (network) có tác dụng truyền thông tin giữa thiết bị lưu trữ và hệ thống máy tính. Một SAN bao gồm một cấu trúc truyền tin, nó cung cấp kết nối vật lý, và quản lý các lớp, tổ chức các kết nối, các thiết bị lưu trữ, và hệ thống máy tính sao cho dữ liệu truyền trên đó với tốc độ cao và tính bảo mật;

2-Một hệ thống lưu trữ bao gồm các thiết bị lưu trữ, hệ thống máy tính, hay các ứng dụng chạy trên nó, và một phần rất quan trọng là các phần mềm điều khiển, quá trình truyền thông tin qua mạng.

1.1.2.3. Ảo hóa máy chủ

Một máy chủ riêng ảo tiếng anh Virtual Private Server, máy chủ ảo hoá là một phương pháp phân vùng một máy chủ vật lý thành nhiều máy chủ ảo, mỗi máy chủ có khả năng của riêng của mình chạy trên máy tính dành riêng. Mỗi máy chủ ảo riêng của nó có thể chạy hệ điều hành khác nhau, và mỗi máy chủ ảo có thể được khởi động lại độc lập [14].

- ✓ Lợi thế của ảo hoá máy chủ:
    - Tiết kiệm được chi phí đầu tư máy chủ ban đầu;
    - Hoạt động hoàn toàn như một máy chủ riêng;

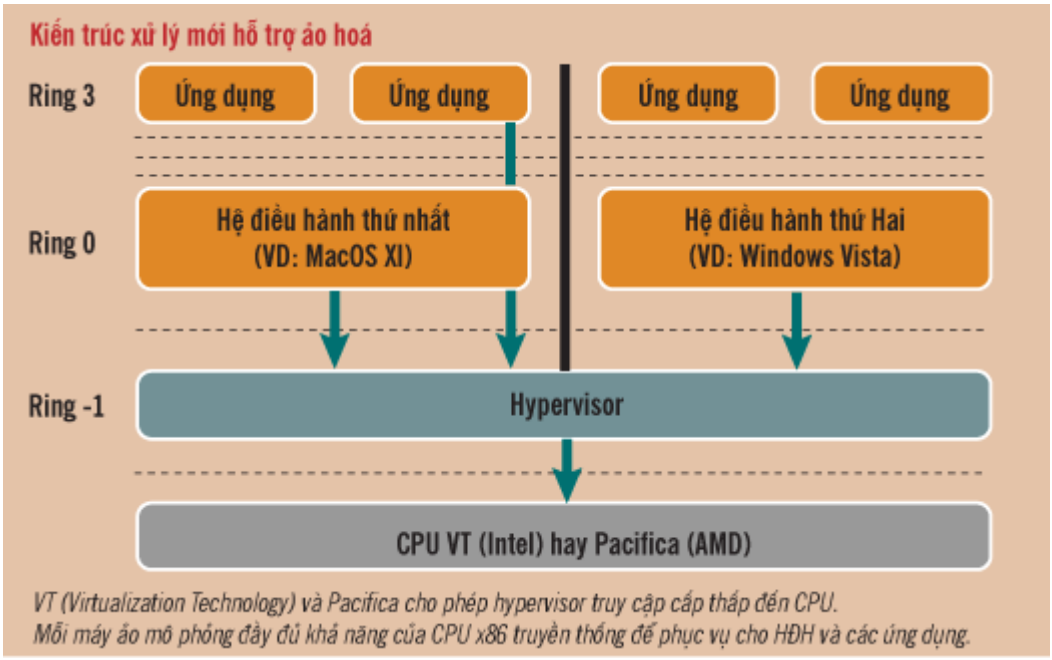
- Có thể dùng máy chủ ảo hoá cài đặt các ứng dụng khác tùy theo nhu cầu của doanh nghiệp;
- Bảo trì sửa chữa nâng cấp nhanh chóng và dễ dàng;
- Dễ dàng nâng cấp tài nguyên RAM, HDD, băng thông khi cần thiết;
- Có thể cài lại hệ điều hành từ 5-10 phút;
- Không lãng phí tài nguyên.

✓ Các môi trường ảo hóa máy chủ

Có hai môi trường máy chủ ảo hoá: ảo hoá toàn phần (Full virtualization) và ảo hoá một nửa (Paravirtualization).

1. Full-virtualization: Phần cứng được mô phỏng để mở rộng chạy những hệ điều hành khách trên nền tảng ảo hóa.Điều này có nghĩa rằng các thiết bị phần cứng khác nhau đều được mô phỏng.Thông thường, có nhiều nền tảng ảo hóa cố gắng chạy nhiều sự ủy nhiệm trên CPU chính (chạy nhanh hơn nhiều so với CPU mô phỏng) nhằm nắm bắt và xử lý các sự ủy nhiệm một cách thích hợp.

Một số nền tảng ảo hóa hỗ trợ hoặc yêu cầu CPU mở rộng để hỗ trợ ảo hóa. Trên 1 số những dòng chíp mới như x86 và x86_64 CPUs được cung cấp thông qua VT-X (Intel) và AMD-V (AMD).Chúng được gọi là Phần Cứng Hỗ Trợ Ảo Hóa (hardware-assisted full-virtualization).



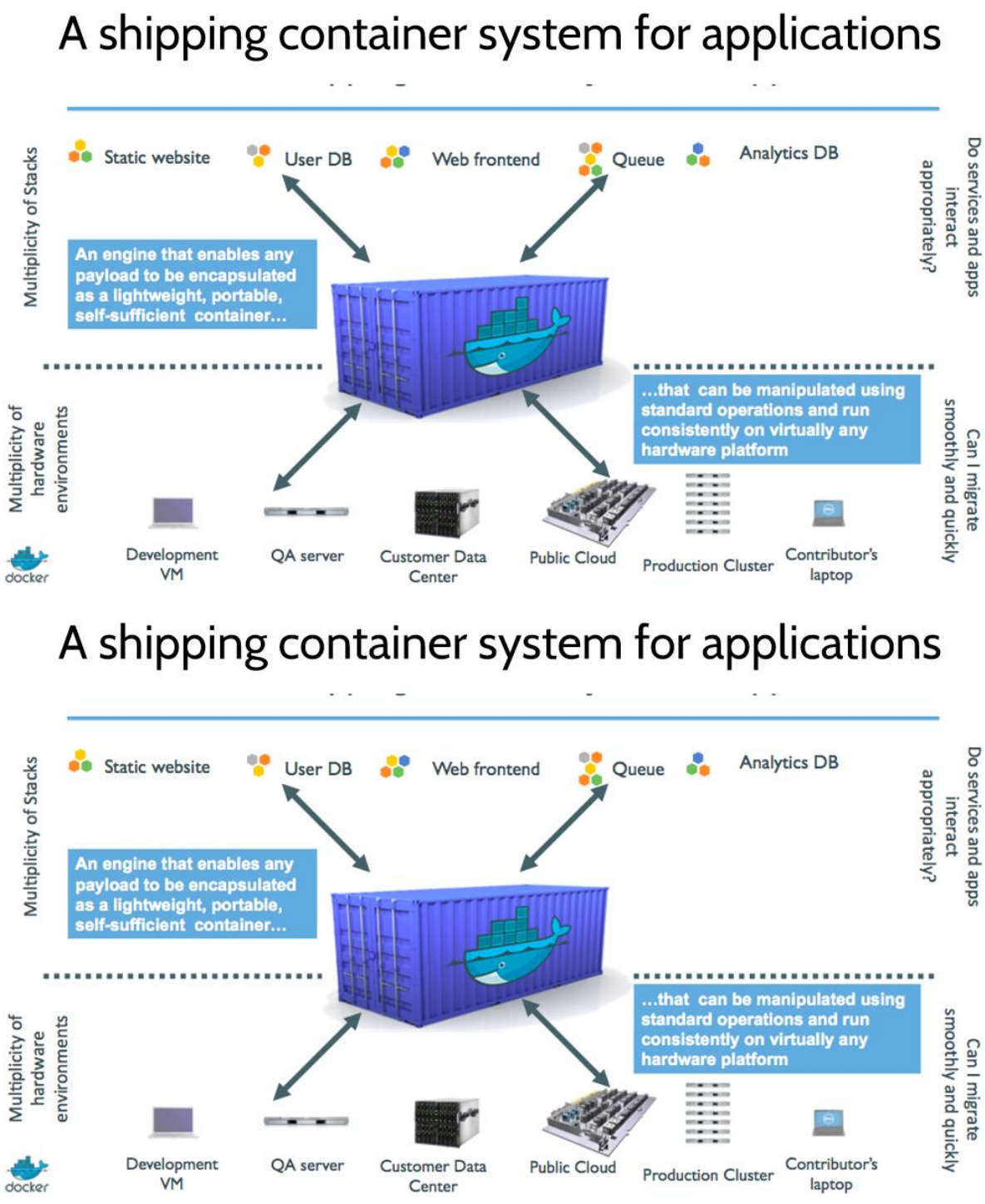
Hình 1. 3: Kiến trúc xử lý mới hổ trợ ảo hóa

2. Paravirtualization: là một phương pháp ảo hóa máy chủ khác. Với phương pháp ảo hóa này, thay vì mô phỏng một môi trường phần cứng hoàn chỉnh, phần mềm ảo hóa này là một lớp mỏng (Hypervisor) dồn các truy cập các hệ điều hành máy chủ vào tài nguyên máy vật lý cơ sở.

1.1.2.4. Ảo hóa ứng dụng

Ảo hóa ứng dụng là một dạng công nghệ ảo hóa khác cho phép chúng ta tách rời mối liên kết giữa ứng dụng và hệ điều hành và cho phép phân phối lại ứng dụng phù hợp với nhu cầu user. Môt ứng dụng được ảo hóa sẽ không được cài đặt lên máy tính một cách thông thường, mặc dù ở góc độ người sử dụng, ứng dụng vẫn hoạt động một cách bình thường. Việc quản lý việc cập nhật phần mềm trở nên dễ dàng hơn, giải quyết sự đụng độ giữa các ứng dụng và việc thử nghiệm sự tương thích của chúng cũng trở nên dễ dàng hơn. Hiện nay đã có khá nhiều chương trình ảo hóa ứng dụng như VirtualBox, Vagrant, Docker, Citrix XenApp, Microsoft Application Virtualization, Vmware ThinApp… [15].

Ảo hóa ứng dụng là giải pháp tiến đến công nghệ "điện tóan đám mây" cho phép ta sử dụng phần mềm của công ty mà không cần phải cài phần mềm này vào bất cứ máy tính con nào.



Hình 1. 4: Ảo hóa ứng dụng

Giải pháp Ảo Hóa Ứng Dụng cho ta những lợi ích nổi trội sau:

- ✓ Tất cả các máy tính đều có thể sử dụng phần mềm ảo như đang cài trên máy tính của mình mà không phải lo về cấu hình (ví dụ chạy Photoshop trên máy

P4 chỉ có 512 MB RAM). Tốc độ phần mềm luôn ổn định và ko phụ thuộc vào cấu hình từng máy;

- ✓ Các máy tính con luôn ở trong tình trạng sạch và chạy nhanh hơn. Loại bỏ hòan tòan việc phải sửa lỗi phần mềm do virus, spyware hoặc do người dùng sơ ý;
- ✓ Cho phép sử dụng phần mềm mà không phải quan tâm đến hệ điều hành ta đang sử dụng (ví dụ: ta có thể dùng Microsoft Office 2007 ngay trong Linux, Windows 98 hoặc MAC-OS);
- ✓ Ta có thể phân phối phần mềm 1 cách linh động đến 1 số cá nhân hoặc nhóm có nhu cầu sử dụng thay vì cài vào tất cả mọi máy như cách phổ thông. Việc phân phối hoặc gỡ bỏ phần mềm ra các máy tính có thể diễn ra chỉ trong vòng chỉ vài giây thay vì hàng tuần nếu như công ty có hàng chục máy tính;
- ✓ Thông tin luôn luôn được lưu trữ an tòan ở server trung tâm thay vì có thể phân tán ra từng máy con. Cho dù ta ở bất cứ nơi nào (tại 1 máy tính khác, tại nhà hay thậm chí ở internet cafe), việc truy nhập và sử dụng phần mềm của doanh nghiệp trở nên dễ dàng qua 1 hệ thống bảo mật hiện đại nhất.

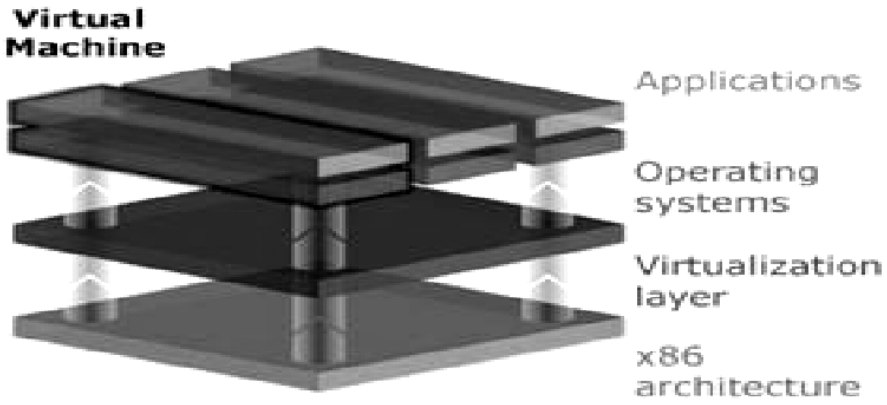
Ảo hóa ứng dụng là giải pháp cho phép sử dụng và quản lý phần mềm doanh nghiệp 1 cách hiệu quả có hệ thống. Tiết kiệm tối đa chi phí bảo trì, hỗ trợ kỹ thuật và quản lý từng máy tính.

Trong ảo hóa ứng dụng thì công nghệ ảo hóa Docker đang được đánh giá là tương lai của công nghệ ảo hoá (future of virtualization), Công nghệ này là sản phẩn của một dự án phần mềm tự do nguồn mở phát hành theo giấy phép Apache. Khác biệt lớn của Docker và các công nghệ ảo hóa khác đó là tiết kiệm đáng kể resource sử dụng. Với docker có thể chạy 20 container (tương tự như một hệ điều hành nhỏ) trên cùng một máy host (host machine), mà nếu sử dụng công nghệ ảo hóa khác như Vagrant sẽ cần một máy chủ với cấu hình rất lớn. Docker làm được điều này là bởi vì khác với Virtual Machine ở chỗ thay vì tách biệt giữa hai môi trường guest và host, thì các container của Docker chia sẻ các resource với host machine.

1.1.3.Các công nghệ giúp ảo hóa hệ thống

1.1.3.1. Công nghệ máy ảo

Máy ảo là một máy tính được cài trên một hệ điều hành khác hay một máy tính khác. Một máy ảo cũng bao gồm phần cứng, các ứng dụng phần mềm và hệ điều hành. Điều khác biệt ở đây là lớp phần cứng của máy ảo không phải là các thiết bị thường mà chỉ là một môi trường hay phân vùng mà ở đó nó được cấp phát một số tài nguyên như là chu kì CPU, bộ nhớ, ổ đĩa….Công nghệ máy ảo cho phép cài và chạy nhiều máy ảo trên một máy tính vật lý. Mỗi máy ảo có một hệ điều hành máy khách riêng lẻ và được phân bố tài nguyên, ổ cứng, card mạng và các tài nguyên phần cứng khác một cách hợp lý. Việc phân bố tài nguyên này phụ thuộc vào nhu cầu của từng máy ảo ứng dụng và cũng tùy thuộc vào phương pháp ảo hóa được dùng. Đặc biệt khi máy ảo cần truy xuất tài nguyên phần cứng thì nó hoạt động giống như một máy thật hoàn chỉnh. Vì chỉ là một tập tin được phân vùng trên ổ đĩa nên việc di chuyển các máy ảo từ máy chủ này sang máy chủ khác là rất dễ dàng và không cần quan tâm đến vấn đề tương thích phần cứng hay ảnh hưởng tới máy chủ.



Hình 1. 5: Mô hình các lớp tương tác trong hệ thống VMs

Trong kiến trúc của một bộ xử lý ảo hóa được chia thành 4 lớp . Lớp 0 là lớp có quyền cao nhất có thể truy cập và can thiệp sâu nhất đến tài nguyên phần cứng.

Lớp 0 thường là các hệ điều hành chủ được cài trên chính máy chủ. Lớp 1 là lớp ảo hóa Hypervisor. Lớp này dùng đề quản lý và phân phối tài nguyên đến các máy ảo. Lớp 2 là các hệ điều hành khách chạy trên các máy ảo. Để truy cập tài nguyên

phần cứng nó phải liên lạc với lớp ảo hóa và phải qua hệ điều hành máy chủ. Lớp có quyền can thiệp thấp nhất đến tài nguyên là lớp 3 là các ứng dụng hoạt động trên các máy ảo.

Trong các hệ thống máy tính lớn dùng để xử lý các ứng dụng thương mại và khoa học (mainframe), hệ điều hành chạy trên phần cứng máy thực ở chế độ ưu tiên vì chỉ có hệ điều hành chủ mới được phép sửa đổi và can thiệp vào phần cứng bên dưới nó. Còn máy ảo làm việc ở chế độ giới hạn vì phần cứng mà nó nhìn thấy chỉ là các thiết bị ảo. Khi máy ảo yêu cầu các lệnh hoặc tiến trình thông thường thì hệ điều hành chủ sẽ chuyển tiếp chúng đến bô xử lý để thực thi trực tiếp, còn đối với các lệnh hoặc các tiến trình đặc biệt nhạy cảm can thiệp sâu đến phần cứng bên dưới sẽ bị chặn lại vì có thể làm ảnh hưởng tới hệ thống và máy ảo còn lại. Hệ điều hành chủ sẽ thực thi lệnh với bộ xử lý trên máy thực rồi sau đó mô phỏng kết quả rồi trả về cho máy ảo. Đây là cơ chế nhằm cách ly máy ảo với máy thực để đảm bảo an toàn hệ thống.

1.1.3.2. Công nghệ cân bằng tải

Trong xu hướng công nghệ, máy chủ là trái tim của của mạng máy tính, nếu máy chủ mạng hỏng, hoạt động của hệ thống sẽ bị ngưng trệ. Do vậy, vấn đề đặt ra là cần có một giải pháp để đảm bảo cho hệ thống vẫn hoạt động tốt ngay cả khi có sự cố xảy ra đối với máy chủ mạng. Giải pháp hiệu quả được đưa ra là sử dụng một nhóm server cùng thực hiện một chức năng dưới sự điều khiển của một công cụ phân phối tải - Giải pháp cân bằng tải (Load Balancing). Có rất nhiều hãng đưa ra giải pháp cân bằng tải như Cisco, Coyote Point, Sun Microsystems... với rất nhiều tính năng phong phú.

Load Balancing là một công nghệ có khả năng chia tải và nâng cao khả năng chịu lỗi của hệ thống. Load Balancing không chỉ làm nhiệm vụ phân phối tải cho các server mà còn cung cấp cơ chế đảm bảo hệ thống server luôn khả dụng trước các client và nhu cầu truy cập từ internet.

Hiện nay có 2 loại cân bằng tải được áp dụng:

1-Cân bằng tải sử dụng phần cứng: sử dụng các mođun cắm thêm trên các thiết bị chuyên dụng như Bộ định tuyến (Router) hay bộ chuyển mạch (Switch) để chia tải theo luồng, thường hoạt động từ layer 4 trở xuống. Vì sử dụng thiết bị chuyên dụng nên có hiệu năng cao, tính ổn định cao, khả năng mở rộng tốt hơn nhưng khó phát triển được tính năng bảo mật phức tạp. Thường sử dụng các thiết bị của Cisco, F5, Citrix,…

2-Cân bằng tải sử dụng phần mềm: Sử dụng phần mềm cài trên server để kết hợp nhiều server một cách chặt chẽ tạo thành một server ảo (virtual server). Cách này có ưu điểm là có thể chia sẻ được nhiều tài nguyên trong hệ thống, theo dõi được trạng thái của các máy chủ trong nhóm để chia tải hợp lý. Tuy nhiên, do sử dụng phần mềm trên server, tính phức tạp cao nên khả năng mở rộng của giải pháp này bị hạn chế, phức tạp khi triển khai cũng như khắc phục khi xảy ra sự cố, có rào cản về tính tương thích, khó có được những tính năng tăng tốc và bảo mật cho ứng dụng. Thường sử dụng các giải pháp Proxy, DNS load balancing, Round Robin NDS,…

1.1.3.3. Công nghệ cân bằng tải mạng

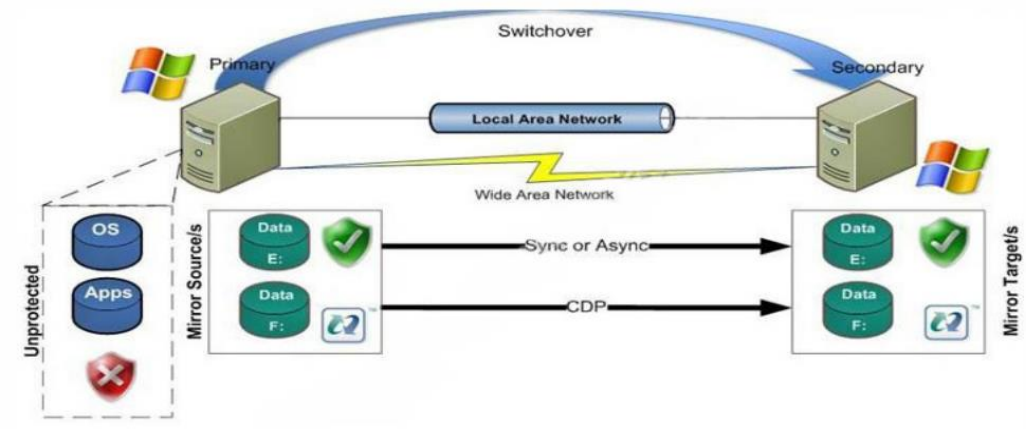
Công nghệ cân bằng tải mạng (Load Balancing) là một công nghệ có khả năng chia tải và nâng cao khả năng chịu lỗi của hệ thống. Được dùng cho các ứng dụng Stateless applications (các ứng dụng hoạt động mang tính nhất thời) như Web, File Tranfer Protocol (FTP), Virtual Private Network (VPN)… Trong hệ thống NLB sẽ bao gồm các cụm server được cấu hình tương tự nhau (có thể được đặt rải rác ở nhiều nơi) cùng hoạt động để phân phối khối lượng công việc giữa các máy chủ trong hệ thống, giúp hệ thống giảm bớt gánh nặng khi phân bố tải.

Nhược điểm của NLB là mỗi cụm server phải dùng riêng một nơi lưu trữ cục bộ (Local Storage) cho nên cần phải có quá trình đồng bộ hóa dữ liệu ở mỗi nơi lưu trữ, số lượng cụm server càng nhiều thì thời gian cho việc đồng bộ hóa càng lâu, chính vì điều này nên ta không nên triển khai các ứng dụng Stateful applications (các ứng dụng hoạt động thường xuyên trong thời gian dài) như các database server: Microsoft SQL Server, Microsoft Exchange Server, File and Print Server… trên kỹ thuật NLB này nhằm đảm bảo tính chính xác của dữ liệu.

1.1.3.4. Công nghệ cân bằng tải Clustering

Đây là công nghệ được dùng rộng rãi cho các hệ thống cần độ sẵn sàng phục vụ cao, đây là giải pháp được đặc biệt quan tâm do tính kinh tế, đa dạng và khả năng dịch vụ cao. Công nghệ này có thể sử dụng phần cứng chuyên dụng để cung cấp một môi trường với độ tin cậy cao đảm bảo cho các dịch vụ có thể hoạt động trơn tru mà không bị dừng bởi một vài lỗi nhỏ; hoặc cũng có thể được thiết kế để chạy trên các phần cứng thông dụng mà vẫn đạt được các yêu cầu:

- ✓ Tăng cường khả năng mở rộng;
- ✓ Nâng cao hiệu suất;
- ✓ Tính sẵn sàng cao và khắc phục sự cố.



Hình 1. 6: Mô hình cân bằng tải Clustering

Với hệ thống sử dụng công nghệ clustering, trong quá trình khởi tạo cấu hình của hệ thống sẽ bao gồm một hệ thống với các nút chính chủ động (active primary node) và một hệ thống với các nút phụ bị động sao lưu (passive backup node). Trong hệ thống này các nút chủ động và bị động cần có cùng cấu hình và được sử dụng công nghệ lưu trữ SAN hoặc Raid (Raid 1).

Nút đang hoạt động (active node) sẽ đáp lại các yêu cầu về dịch vụ thông qua một địa chỉ IP ảo (Virtual IP hay VIP). Địa chỉ VIP là một địa chỉ IP và nó chỉ khác so với địa chỉ IP thông thường của một nút đang hoạt động.

Hệ thống bị động (gồm các nút không hoạt động) sẽ không trực tiếp chạy dịch vụ, thay vào đó nó quản lí các dịch vụ của nút chủ động đang hoạt động, và đảm bảo chắc chắn là nút đang hoạt động vẫn phải đang còn hoạt động. Nếu nút không hoạt

động phát hiện ra 1 vấn đề nào đó với hoặc là nút hoạt động hoặc dịch vụ đang chạy trên nó, thì một thông báo lỗi sẽ được khởi tạo.Khi có lỗi, hệ thống clustering sẽ thực hiện các bước sau:

- ✓ Bước 1: Nút đang hoạt động sẽ trực tiếp ngắt hết các dịch vụ đang chạy và các kết nối;
- ✓ Bước 2: Nút không hoạt động sẽ khởi động các dịch vụ tương đương với các dịch vụ của máy chủ động;
- ✓ Bước 3: Nút đang hoạt động sẽ ngắt không sử dụng địa chỉ VIP;
- ✓ Bước 4: Nút không hoạt động bây giờ lại chuyển thành nút đang hoạt động, và ở chế độ sử dụng địa chỉ VIP;
- ✓ Bước 5: Nếu nó vẫn đang hoạt động và đang duy trì kết nối mạng, nút trước kia là chủ động thì bây giờ trở thành nút bị động, bắt đầu giám sát các dịch vụ của nút chủ động.

1.1.3.5. Công nghệ RAID

RAID (Redundant Array of Independent Disks) có nghĩa là sự tận dụng các phần dư trong các ổ cứng độc lập. Ban đầu, RAID được sử dụng như một giải pháp phòng hộ vì nó cho phép ghi dữ liệu lên nhiều đĩa cứng cùng lúc. RAID chính là sự kết hợp giữa các đĩa cứng vật lý bằng cách sử dụng một trình điều khiển đặc biệt RAID có thể sử dụng như là một phần cứng lẫn phần mềm.

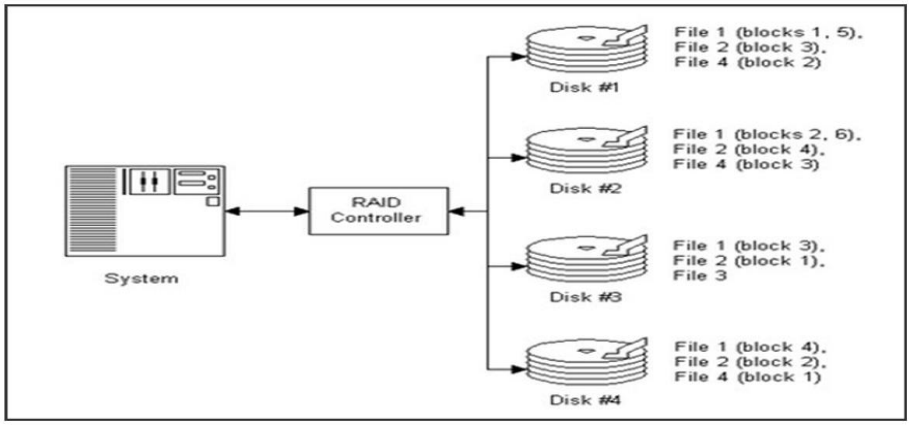
Hệ thống có sử dụng RAID được dùng trong việc đảm bảo an toàn dữ liệu khi có ổ đĩa bị lỗi và phục hồi lại các dữ liệu, có thể thay "nóng" ổ đĩa đối với một số loại RAID và cũng còn tùy thuộc vào máy chủ. RAID ngày càng trở nên cần thiết cho các hệ thống máy tính.

Vì RAID mang tính toàn vẹn dữ liệu cao , phục hồi nhanh chóng nên RAID chủ yếu được ứng dụng vào các máy chủ, không phải là các máy bàn không thể dùng Raid được mà là do chi phí đầu tư khá tốn kém nên chỉ ở các hệ thống lớn đòi hỏi độ an toàn cho dữ liệu phải cao mới sử dụng.

1.1.3.5.1. Striping

Là một trong những chuẩn RAID có hiệu năng cao nhất, nó giúp ta tăng tốc độ truy cập lên tối đa bằng cách ghi song song dữ liệu lên các ổ đĩa này. Kỹ thuật này sẽ chia các tệp dự liệu ra và ghi đồng thời lên ổ đĩa cứng trong cùng một thời gian. Và khi đọc thì cũng đọc cùng lúc trên tất cả các ổ đĩa làm cho tốc độ đọc và hiệu suất cao.

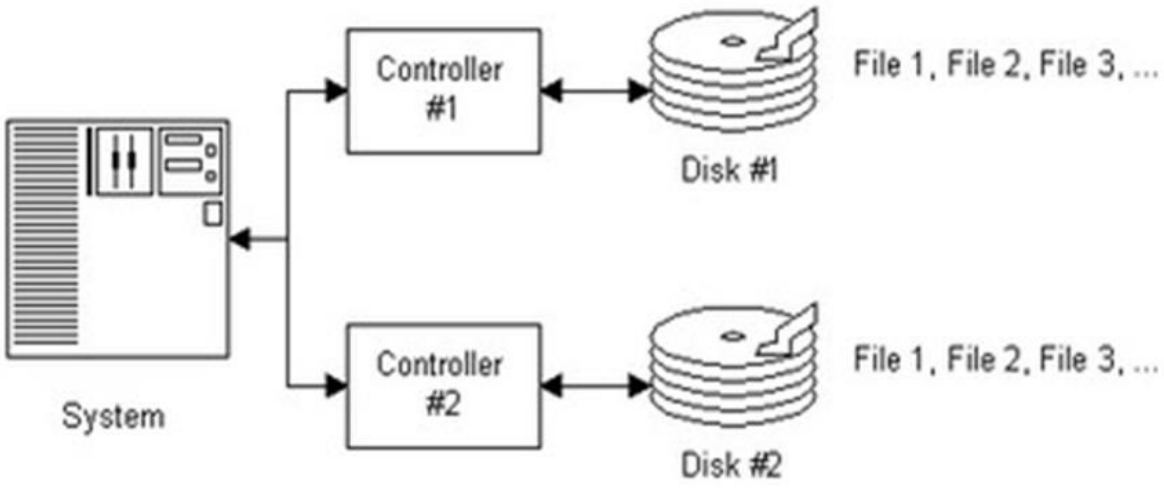
Ở cấp độ byte Striping chia ra thành từng gói nhỏ có kích thước một byte và bộ điều khiển sẽ ghi byte này lên ổ đĩa, trong cấp độ Block thì tập tin cũng bị chia nhỏ, lúc này chia nhỏ như thế nào thì tùy theo kích thước của Block nó như thế nào, tập tin sẽ được lưu và phân bổ trên các Block này.



Hình 1. 7: RAID Song hành

1.1.3.5.2. Duplexing

Đây là chuẩn mở rộng của Mirroring. Dữ liệu cũng được ghi trên hai ổ cứng nhưng phải có 2 bộ điều khiển RAID kết nối với 2 đĩa cứng. Từ đây ta đã thấy chuẩn này khá tốn kém. Nhưng có một đặc tính là Duplexing mang tính bảo mật cao hơn Mirroing vì ở đây nó dùng tới 2 card điều khiển RAID.



Hình 1. 8: RAID Ghép đôi (soi gương)

1.1.3.5.3. Parity RAID

Đây là phương pháp bảo vệ an toàn cho dữ liệu, sử dụng các thông tin mang tính chẵn lẻ bằng cách lưu giữ một con số nhị phân 0 hoặc 1 cho biết tổng các bit trong gói tin là chẵn hay lẻ. Nếu dùng chuẩn này thì lợi ích lớn nhất của nó là không yêu cầu hệ thống RAID bớt đi một phần dung lượng để lưu trữ dữ liệu. Nhưng cũng có khuyết điểm của nó là phải yêu cầu hệ thống có một phần cứng thật mạnh.

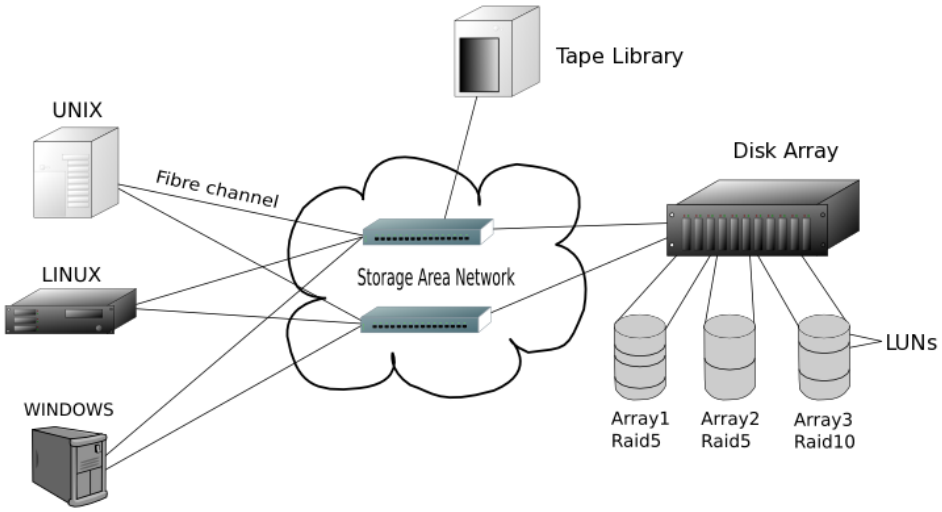
1.1.3.5.4. JBOD

JBOD được viết tắt từ "Just a Bunch of Disks" và không phải là hệ thống RAID chính thống, nó không có mục đích cải thiện hiệu suất của ổ cứng hay độ tin cậy. Nó dùng để ghép những ổ cứng có dung lượng khác nhau thành một dung lượng lưu trữ duy nhất.

Từ các chuẩn Raid trên cho ra đời các loại Raid như Raid 0, Raid 1, Raid 2, Raid 3, Raid 4, Raid 5, Raid 6, Raid 0+1, Raid 10,…là các ứng dụng dựa trên các công nghệ của những chuẩn RAID.

1.1.3.6. Công nghệ lưu trữ SAN

SAN là một hệ thống mạng lưu trữ chuyên dụng kết nối nhiều Server và nhiều thiết bị lưu trữ, với mục đích chính là truyền tải dữ liệu giữa hệ thống máy tính và phần tử lưu trữ và giữa các phần tử lưu trữ với nhau.



Hình 1. 9: Mô hình lưu trữ SAN

Trong ảo hóa công nghệ lưu trữ mạng được dùng làm trung tâm của dữ liệu và cũng có thể làm nơi chứa các máy ảo khi cần thiết. Nó hỗ trợ các máy chủ có thể lấy dữ liệu từ nó để khởi động.

Lợi ích của SAN:

- ✓ Lưu trữ tập trung dữ liệu trên một hệ thống đơn nhất;
- ✓ Cho phép truy cập dữ liệu với tốc độ rất cao (2Gb/s; 4Gb/s tương lai lên tới 10Gb/s);
- ✓ Tính ổn định liên tục của thiết bị rất cao (99,999%);
- ✓ Dễ dàng nâng cấp, mở rộng trong tương lại (tăng thêm số lượng máy chủ, hay mở rộng dung lượng lưu trữ lên rất lớn và uyển chuyển): bảo vệ sự đầu tư thông qua khả năng tương thích ngược với các switch thế hệ cũ hơn, và khả năng nâng cấp firmware của thiết bị mà không phải dừng hoạt động hệ thống, đảm bảo mọi dịch vụ đều được duy trì liên tục 100%;
- ✓ Có khả năng sao lưu dữ liệu trong nội bộ hệ thống SAN, mà không phải dừng dịch vụ của máy chủ để sao lưu như các hệ thống lưu trữ khác, đồng thời không hề ảnh hưởng băng thông của mạng LAN khi thực hiện các thao tác backup;
- ✓ Bảo mật tốt: xác thực, xác quyền, điều khiển truy xuất và khả năng quản lý theo vùng tăng thêm mức bảo mật mạng;

- ✓ Hỗ trợ nhiều hệ điều hành và môi trường cluster, cho phép thiết kế linh động và bảo vệ sự đầu tư;
- ✓ Uyển chuyển trong khoảng cách, kết nối và hiệu suất, với khả năng hỗ trợ nhiều hơn 3000 port. Hỗ trợ cơ sở hạ tầng đa giao thức gồm FC, iSCSI, và FCIP.

1.2. Phần mềm tự do nguồn mở

Phần mềm bao gồm một tập hợp các thành phần: Các tài liệu (phân tích thiết kế, hướng dẫn cài đặt, sử dụng, bảo trì, nâng cấp); CSDL được cài đặt trong môi trường tích hợp; Chương trình máy tính khả thi, phù hợp với hệ quản trị CSDL đã cài đặt CSDL; Các tiện ích số hóa đi kèm hỗ trợ cho chương trình máy tính[4].

Phần mềm đã trở thành một tài nguyên của xã hội manh tính chiến lược trong vài thập kỷ gần đây. Sự nổi lên của Phần mềm Tự do (PMTD), mà nó đã thâm nhập vào các khu vực chủ chốt của thị trường công nghệ thông tin và truyền thông (ICT), đang thay đổi một cách mạnh mẽ nền kinh tế của việc sử dụng và phát triển phần mềm. PMTD - đôi khi còn được gọi như là "Nguồn mở" hoặc "Phần mềm Tự do" - có thể được sử dụng, sao chép, sửa đổi và phân phối một cách tự do. Nó đưa ra sự tự do học và dạy mà không có việc lôi kéo vào những sự phụ thuộc vào bất kỳ nhà cung cấp công nghệ duy nhất nào. Những sự tự do này được xem xét như là điều kiện cơ bản tiên quyết cho sự phát triển bền vững và tổng thể của một xã hội thông tin.

Khái niệm PMTD, như được hiểu bởi Richard Stallman trong định nghĩa của ông, tham chiếu tới các quyền tự do được đảm bảo cho người nhận nó, được nhắc tới 4 quyền sau:

1-Tự do chạy chương trình ở bất cứ đâu, vì bất kỳ mục đích gì và vĩnh viễn;

2-Tự do nghiên cứu cách mà nó làm việc và để áp dụng nó cho các nhu cầu của chúng ta. Điều này cần tới sự truy cập vào mã nguồn;

3-Tự do phân phối lại các bản sao, sao cho chúng ta có thể giúp được những bạn bè và hàng xóm của chúng ta;

4-Tự do cải tiến chương trình và đưa những cải tiến đó ra cho công chúng. Điều này cũng cần tới mã nguồn.

Cơ chế mà đảm bảo cho những quyền tự do này, theo pháp luật hiện hành, là sự phân phối theo một giấy phép đặc biệt. Thông qua giấy phép này, tác giả trao các quyền cho người nhận chương trình để thi hành các quyền tự do này, cũng như bổ sung thêm bất kỳ sự hạn chế nào mà tác giả có thể mong muốn áp dụng (như để công nhận các tác giả ban đầu trong trường hợp của một sự phân phối lại). Để giấy phép được xem là tự do, những hạn chế này phải không làm mất tác dụng các quyền tự do đã được nêu ở trên[3].

1.2.1. Lịch sử phát triển

Trong những năm 50, 60, 70 thì người sử dụng máy tính đã có quyền tự do sử dụng các phần mềm miễn phí. Phần mềm miễn phí được những người sử dụng máy tính chia sẻ miễn phí với nhau và cũng do chính các nhà sản xuất chế tạo máy tính vì họ phấn khởi do có nhiều người đang cùng họ sáng tạo ra những phần mềm làm cho máy tính của họ có ích, không phải là những cục sắt vô dụng. Những tổ chức người tiêu dùng và nhà sản xuất được lập nên để tạo điều kiện cho việc trao đổi phần mềm ví dụ như SHARE. Vào những năm cuối của thập kỉ 60 thì xuất hiện những thay đổi đáng ngại: giá phần mềm tăng lên nhanh chóng, giữa nhà sản xuất phần cứng có cài đặt sẵn và nhà sản xuất phần mềm cũng xuất hiện sự cạnh tranh gay gắt để mở rộng thị trường vì khi đó phần mềm miễn phí vì chi phí của nó đã nằm trong giá phần cứng.Nhưng việc cài đặt sẵn những phần mềm như vậy lại không đem lại lợi ích gì cho việc bán phần mềm và người sử dụng đôi khi lại không cần những thứ được cài sẵn nên họ không muốn phải chi trả cho những thứ không xài tới. Trong bài "Nước Mỹ và IBM" United States vs. IBM, ngày 17/1/1969 chính quyền đã cho rằng việc cài đặt phần mềm đi kèm phần cứng khi bán ra thị trường là một kiểu cạnh tranh không lành mạnh. Tuy rằng vẫn có nhiều phần mềm là hoàn toàn miễn phí nhưng đa phần vẫn chỉ là những sản phẩm thương mại. Trong suốt quãng thời gian những năm 70 và thời kì đầu những năm 80,nền công nghệ phần mềmbắt đầu sử dụng các tiêu chuẩn về công nghệ (ví dụ như chỉ cho phân phối các phiên bản sử dụng, các bản sao nhị phân binary copies của chương trình máy tính) nhằm ngăn người sử dụng máy tính nghiên cứu và chỉnh sửa các phần mềm. Năm 1980 bộ luật quyền tác giả được mở rộng sang phần mềm máy tính.

Năm 1983, Richard Stallman, là thành viên lâu năm của cộng đồng hacker của MIT Artificial Intelligence Laboratory, chính ông cũng đã khởi xướng dự án GNU. Stallman nói rằng ông thấy chán nản vì những tác động thay đổi về văn hóa trong nền công nghiệp máy tính và người dùng máy. Sự phát triển các phần mềm cho hệ điều hành GNU,GNU operating system, bắt đầu từ 1/1984, và Tổ chức phần mềm tự do Free Software Foundation (FSF) được thành lập năm 1985. Ông đã phát triển một định nghĩa riêng cho phần mềm tự dovà khái niệm "copyleft".

Và tiềm năng thương mại của các phần mềm tự do được các công ty lớn nhìn thấy như IBM, Red Hat, và Sun Microsystems. Cũng có rất nhiều công ty không thuộc lĩnh vực công nghệ thông tin chọn các phần mềm miễn phí để làm các trang web thông tin và thương mại của họ vì chi phí đầu tư thấp và khả năng tự do đóng gói dữ kiện của các phần mềm dạng này. Ngoài ra cũng có những công ty trong các ngành công nghiệp phi phần mềm sử dụng các công nghệ tương tự như công nghệ phát triển phần mềm tự do trong quá trình nghiên cứu và phát triển. Một ví dụ minh chứng là các nhà khoa học cũng luôn mong muốn có một quy trình nghiên cứu tiên tiến hơn những công nghệ hiện tại và đã xuất hiện nhiều thiết bị phần cứng như microchips với giấy phép copyleft. Creative Commons cũng bị ảnh hưởng rất nhiều bởi trào lưu phần mềm tự do.

1.2.2.Ưu thế của phần mềm tự do mã nguồn mở so với phần mềm nguồn đóng

Lợi ích phần mềm nguồn mở thể hiện rỏ nhất ở tính kinh tế, sử dụng phần mềm nguồn mở tiết kiệm được nguồn tiền khổng lồ, nguồn tiền tiết kiệm trên sẽ giúp các nước đang phát triển hạn chế được hiện tượng chảy máu chất xám, khi mà các sinh viên được đào tạo về khoa học máy tính và phần mềm không còn đi tìm những công việc phù hợp với khả năng của họ tại các nước khác mà có thể làm việc tại đất nước mình.

Ở vấn đề giáo dục đào tạo phần mềm nguồn mở là là nền tảng cho việc giáo dục về khoa học máy tính, nếu dạy học về phần mềm sở hữu độc quyền, thì người học biết sẽ biết cách sử dụng phần mềm đó, nhưng nếu dạy và học về phần mềm nguồn mở thì người học không những biết cách sử dụng phần mềm nguồn mở mà còn biết thêm thông tin hoạt động của của phần mềm đó như thế nào. Song đôi lúc người ta lựa chọn phần mềm không chỉ dựa vào tính kinh phí phần mềm đó mà còn dựa vào

độ chất lượng và ứng dụng của nó. Xét về phần mềm nguồn mở nó có các đặc điểm sau đây: tính an toàn, tính ổn định và đáng tinh cậy, giảm lệ thuộc vào xuất khẩu, vấn đề vi phạm bản quyền, quyền sở hữu trí tuệ và tính tuân thủ WTO, bản địa hóa, các chuẩn mở và sự không lệ thuộc vào nhà cung cấp, phát triển năng lực ngành công nghiệp địa phương.

Ở phần mềm nguồn mở hầu như không có Virus gây hại cho máy tính, đây cũng là vấn đề khiến mã nguồn mở ngày được quan tâm hơn so với phần mềm sử dụng mã đóng như Window ví dụ như khi mua máy cài bản quyền Window thì phải mua thêm phần mềm diệt Virus lại tiếp tục tốn tiền mua bản quyền phần mềm này.

Những ưu điểm phần mềm nguồn mở nói trên thể hiện như sau[2]:

1. Tính an toàn

Mã nguồn được phổ biến rộng rãi: việc mã nguồn được phổ biến rộng rãi giúp người lập trình và người sử dụng dễ phát hiện, khắc phục các lỗ hỏng an toàn trước khi chúng bị lợi dụng. Đa phần các lỗi hệ thống của phần mềm nguồn mở được phát hiện trong quá trình rà soát định kỳ và được sửa trước khi gây ra bất kỳ thiệt hại nào. Các hệ thống phần mềm nguồn mở thường có quy trình rà soát chủ động chứ không phải rà soát đối phó.

Ưu tiên về tính an toàn đặt trên tiêu chí tiện dụng: có thể nói phần mềm nguồn mở được dùng để điều hành một phần lớn mạng internet và do đó nó nhấn mạnh nhiều đến tính bền vững, chức năng vận hành thay vì tính dễ sử dụng. Trước khi thêm bất cứ tính năng nào vào một ứng dụng phần mềm nguồn mở, bao giờ người ta cũng cân nhắc đến khía cạnh an toàn, và tính năng đó sẽ chỉ được đưa vào nếu không làm yếu đi tính an toàn của hệ thống.

Các hệ thống phần mềm nguồn mở chủ yếu dựa trên mô hình của Unix: nhiều người sử dụng, thuận tiện cho kết nối mạng. Do đó, chúng được thiết kế với một cấu trúc an toàn bảo mật cao. Điều này là đặc biệt quan trọng khi có nhiều người cùng chia sẻ quyền sử dụng một máy chủ cấu hình mạnh, bởi vì nếu hệ thống có độ an toàn thấp, một người sử dụng bất kỳ có thể đột nhập vào máy chủ, đánh cắp dữ liệu cá nhân của người khác, hoặc làm cho mọi người không tiếp cận được với các dịch vụ

do hệ thống cung cấp. Kết quả của mô hình thiết kế này là chỉ có rất ít vụ tấn công được thực hiện thành công với các phần mềm nguồn mở.

Vậy tóm lại một gói phần mềm được tạo ra bởi một vài nhà thiết kế, hay một gói phần mềm do hàng nghìn nhà thiết kế sáng tạo nên người sử dụng sẽ chọn lựa như thế nào. Do phần mềm mã nguồn mở được sáng tạo bởi vô số các nhà thiết kế và người sử dụng nên độ bảo mật của chúng sẽ được cải thiện, cũng như chúng cũng sẽ được mang thêm nhiều tính năng mới và những cải tiến mới nên phần mềm mã mở sẽ dễ chú ý sử dụng hơn.

2. Tính ổn định và đáng tin cậy

Các phần mềm nguồn mở thường ổn định và đáng tin cậy đó là kết luận từ những cuộc phân tích, đánh giá và so sánh với các phần mềm nguồn đóng khác. Ví du như: một cuộc thử nghiệm theo phương pháp chọn ngẫu nhiên được tiến hành vào năm 1995, tập trung thử nghiệm 7 hệ điều hành thương mại và GNU/Linux. Người ta nạp vào các hệ điều hành này những tính năng ngẫu nhiên theo một trình tự lộn xộn, bắt chước hành động của những người sử dụng kém hiểu biết. Kết quả là các hệ điều hành thương mại có tỷ lệ xung đột hệ thống trung bình là 23% trong khi Linux chỉ bị lỗi vận hành trong 9% số lần thử nghiệm. Các tiện ích của GNU (phần mềm do FSF xây dựng trong khuôn khổ dự án GNU) bị lỗi vận hành có 6% số lần thử nghiệm. Nhiều năm sau, một nghiên cứu tiếp nối còn cho thấy tất cả những lỗi gặp trong cuộc thử nghiệm nói trên đều đã được khắc phục với hệ điều hành FOSS (FOSS là một thuật ngữ bao gồm bao gồm cả phần mềm tự do và phần mềm nguồn mở), trong khi với các phần mềm đóng thì vẫn hầu như chưa được đụng đến.

3. Giảm lệ thuộc vào nhập khẩu

Một trong những động cơ quan trọng khiến các quốc gia đang phát triển nhiệt tình hưởng ứng phần mềm nguồn mở chính là chi phí khổng lồ của giấy phép sử dụng các phần mềm đóng. Vì hầu như toàn bộ phần mềm của các nước đang phát triển đều được nhập khẩu, tiền mua những phần mềm này sẽ làm tiêu hao quỹ dự trữ ngoại tệ hết sức quý báu mà lẽ ra có thể được sử dụng hiệu quả hơn cho những mục tiêu phát triển khác. Công trình phần mềm nguồn mở tự do: nghiên cứu và khảo sát còn cho biết mô hình phần mềm nguồn mở này thiên nhiều hơn về dịch vụ công, do đó chi

phí cho phần mềm cũng là để phục vụ những hoạt động của cơ quan Chính phủ chứ không phải cho mục đích lợi nhuận của các công ty đa quốc gia. Điều này có ảnh hưởng tích cực đến tạo công ăn việc làm cho xã hội, mở rộng năng lực đầu tư nội địa, và tăng thu cho ngân sách địa phương…

4. Vấn đề vi phạm bản quyền, quyền sở hữu trí tuệ, và tính tuân thủ WTO

Nạn sao chép phần mềm là vấn đề mà hầu như quốc gia nào trên thế giới cũng gặp phải. Tổ chức Business Software Alliance ước tính riêng trong năm 2002, tệ nạn này làm nước Mỹ thiệt hại mất 13,08 tỷ đôla. Ngay với các quốc gia phát triển, nơi mà trên lý thuyết giá phần mềm còn vừa túi tiền người dân, tỷ lệ sao chép phần mềm vẫn ở mức rất cao (24% ở Mỹ và 35% ở Châu Âu). Tại các quốc gia đang phát triển, nơi mà mức thu nhập thấp khiến cho phần mềm trở thành một thứ hàng xa xỉ, thì tỷ lệ sao chép có thể đạt tới 90%. Nạn sao chép phần mềm và hệ thống luật pháp lỏng lẻo sẽ gây thiệt hại cho một quốc gia trên nhiều phương diện. Quốc gia nào yếu trong việc thực thi bảo vệ quyền sở hữu trí tuệ sẽ kém hấp dẫn với các nhà đầu tư nước ngoài. Quyền gia nhập WTO và khả năng tiếp cận những lợi ích mà tổ chức này mang lại bị ảnh hưởng khá nhiều bởi mức độ bảo vệ quyền sở hữu trí tuệ mà một quốc gia đạt được. Nạn sao chép phần mềm còn gây hại cho nền công nghiệp phần mềm nội địa, do các nhà lập trình địa phương giờ đây chẳng còn mấy động cơ để xây dựng những phần mềm bản địa.

5. Bản địa hóa

"Bản địa hoá là thích ứng một sản phẩm, làm cho nó phù hợp về mặt ngôn ngữ và văn hoá với thị trường mục tiêu (quốc gia hoặc địa phương), nơi sản phẩm được tiêu thụ và sử dụng". Bản địa hoá là một trong những lĩnh vực nơi phần mềm nguồn mở tỏ rõ ưu thế của mình. Người sử dụng phần mềm nguồn mở có thể tự do sửa đổi để phần mềm trở nên thích ứng với những nhu cầu riêng biệt của một khu vực văn hoá đặc thù, bất kể quy mô kinh tế của khu vực đó. Chỉ cần một nhóm nhỏ những người có đủ trình độ kỹ thuật là đã có thể tạo ra một phiên bản nội địa ở mức độ thấp cho bất kỳ phần mềm nguồn mở nào. Còn việc xây dựng một hệ điều hành đã bản địa hóa hoàn chỉnh, mặc dù không đơn giản, nhưng ít ra cũng là khả thi. Việc Microsoft vào năm 1998 quyết định không xây dựng phiên bản Window 98 cho Iceland có thể đã gây nên những tác hại khó lường nếu như không có giải pháp thay thế của phần

mềm nguồn mở. (Windows 98 là cải tiến của phiên bản trước, nó khá giống Windows 95. Một số cải tiến hữu ích như hỗ trợ USB, chia sẻ kết nối Internet).

6. Các chuẩn mở và việc không phải lệ thuộc vào nhà cung cấp

Sẵn có mã nguồn: với mã nguồn được phổ biến công khai, người ta lúc nào cũng có thể tái thiết kế và tích hợp lại bộ chuẩn của một ứng dụng. Mọi khả năng tuỳ biến đều đã thể hiện rõ trong mã nguồn, khiến cho không ai có thể giấu một chuẩn riêng trong một hệ thống phần mềm nguồn mở. Đối với phần mềm đóng thì việc tái thiết kế sẽ khó hơn. Một số mã còn được viết ra để đánh lạc hướng người dùng.

Chủ động tương thích chuẩn: khi đã có những chuẩn được thừa nhận rộng rãi, ví dụ như HyperText Markup Language (HTML) bộ chuẩn quy định cách thức hiển thị các trang web, thì các dự án phần mềm nguồn mở luôn chủ động bám sát những chuẩn này. Khi sử dụng các hệ thống phần mềm nguồn mở để thoát khỏi việc lệ thuộc vào nhà cung cấp.

Ví dụ: các doanh nghiệp có thể biến đổi một phần của gói phần mềm mã nguồn mở để biến chúng phù hợp với những nhu cầu của mình. Nhờ vào tính mở của các mã nguồn mà người sử dụng chỉ cần thay đổi mã nguồn để đạt được tính năng như ý muốn. Họ không thể làm được điều đó với các phần mềm có bản quyền.

1.2.3. Các khía cạnh pháp lý của phần mềm tự do nguồn mở

Về phương diện pháp lý mà nói, thì các chương trình tự do so với các phần mềm sở hữu độc quyền là không khác nhau: chúng cả 2 đều được phân phối theo một giấy phép. Sự khác biệt nằm ở những gì giấy phép này cho phép. Trong trường hợp các giấy phép của chương trình tự do, nó không hạn chế việc sử dụng, phân phối lại và sửa đổi chúng.

Có nhiều loại giấy phép tự do, Có thể phân chia các giấy phép của PMTD thành 2 họ lớn. Họ đầu bao gồm các giấy phép mà không đặt ra những điều kiện đặc biệt lên việc phân phối lại lần 2, là các giấy phép dễ dãi. Họ thứ 2, gọi là các giấy phép mạnh (hoặc các giấy phép copyleft), bao gồm những giấy phép mà chúng ở dạng của GNU GPL, áp đặt những điều kiện trong trường hợp muốn phân phối lại phần

mềm, có mục đích để đảm bảo tuân thủ với những điều kiện của giấy phép sau lần phân phối lại lần đầu.

1.2.3.1. Những giấy phép dễ dãi

Những giấy phép dễ dãi, cũng còn đôi khi được biết như là những giấy phép hào phóng hoặc tối thiểu, hầu như không áp đặt bất kỳ điều kiện nào lên người nhận phần mềm, và vâng, cho phép sử dụng, phân phối lại và sửa đổi. Từ quan điểm cụ thể nào đó, tiếp cận này có thể được nhìn nhận như một sự đảm bảo sự tự do tối đa cho người nhận chương trình. Nhưng từ quan điểm khác, nó cũng có thể được hiểu như sự cẩu thả tối đa xét về việc đảm bảo rằng một khi ai đó nhận được chương trình, thì người đó đảm bảo những quyền tự do y hệt khi phân phối lại chương trình đó. Trong thực tế, những giấy phép này thường cho phép phần mềm phân phối lại bằng một giấy phép sở hữu độc quyền .

Một số giấy phép dễ dãi[3]:

- ✓ Trong số những giấy phép này, thì giấy phép BSD là nổi tiếng nhất, ở một chừng mực nào đó thì những giấy phép dễ dãi thường sẽ được tham chiếu tới như các giấy phép loại BSD. Giấy phép BSD (Berkeley Software Distribution) bắt nguồn từ xuất bản phẩm của các phiên bản khác nhau của Unix của Đại học Berkeley, California tại Mỹ. Bổn phận duy nhất mà nó đưa ra là để công nhận các tác giả, trong khi nó cho phép phân phối lại ở cả các định dạng nhị phân và mã nguồn, mà không bắt ép theo bất kỳ cách nào trong mọi trường hợp. Nó cũng trao quyền để thực hiện bất kỳ sự thay đổi nào và được tích hợp vào trong các chương trình khác mà hầu như không có bất kỳ hạn chế nào;
- ✓ Giấy phép X Window, phiên bản 11 (X11). Đây là giấy phép được sử dụng để phân phối hệ thống X Window, hệ thống các cửa sổ được sử dụng rộng rãi nhất trong thế giới các môi trường của Unix, và cũng của GNU/Linux. Nó rất tương tự như giấy phép BSD, mà nó cho phép phân phối lại, sử dụng và sửa đổi mà thực tế không có bất kỳ hạn chế nào. Đôi khi nó được gọi là giấy phép MIT (với một sự thiếu chính xác nguy hiểm, vì MIT đã sử dụng các dạng giấy phép khác). Các công việc dẫn xuất từ hệ thống X Window, như XFree86 cũng được phân phối theo giấy phép này;
- ✓ Zope Public License 2.0. Giấy phép này (thường được tham chiếu như là ZPL) được sử dụng cho sự phân phối Zope (một máy chủ ứng dụng) và các sản phẩm liên

quan khác. Nó tương tự như BSD, với tính năng kỳ lạ rằng nó chỉ cốt để cấm sử dụng các thương hiệu được đăng ký bởi tập đoàn Zope;

✓ Giấy phép Apache Đây là giấy phép mà theo đó hầu hết các chương trình được sản xuất bởi dự án Apache được phân phối. Nó tương tự như giấy phép BSD. Có một số chương trình tự do mà không được phân phối với một giấy phép cụ thể, hay đúng hơn là tác giả công bố một cách rõ ràng chúng thuộc về miền công cộng. Hệ quả chính của tuyên bố này là việc tác giả khước từ mọi quyền đối với chương trình này, mà nó có thể vì thế được sửa đổi, phân phối lại, sử dụng, …, theo bất kỳ cách nào. Theo những điều khoản thực tế, nó rất tương tự như một chương trình theo một giấy phép loại BSD. Phần mềm mã nguồn mở Docker phát hành theo giấy phép này [12].

1.2.3.2. Các giấy phép mạnh

GNU General Public Licence (GNU GPL) Trong những điều khoản cơ bản, giấy phép GPL cho phép phân phối lại ở dạng nhị phân và dạng mã nguồn, dù trong trường hợp sự phân phối lại bằng mã nhị phân thì sự truy cập tới mã nguồn cũng là bắt buộc. Nó còn cho phép những sửa đổi được thực hiện mà không có bất kỳ hạn chế nào. Tuy nhiên, điều này chỉ có thể để phân phối lại mã nguồn được cấp phép theo GPL tích hợp được với mã nguồn khác (ví dụ, việc liên kết mã nguồn) nếu nó có một giấy phép tương thích.

The GNU Lesser General Public Licence (GNU LGPL) LGPL cho phép các chương trình tự do được sử dụng với các phần mềm sở hữu độc quyền. Bản thân chương trình này được phân phối lại là theo giấy phép GPL, sự tích hợp của nó với bất kỳ gói phần mềm nào khác được cho phép mà hầu như không có bất kỳ hạn chế nào.

Các giấy phép mạnh khác đáng lưu ý gồm[3]:

✓ Giấy phép con mèo ngủ Sleepycat. Đây là giấy phép theo đó công ty Sleepycat (http://www.sleepycat.com/) đã phân phối các chương trình của hãng. Nó ép buộc một số điều kiện cụ thể bất kỳ khi nào chương trình hoặc công việc dẫn xuất từ chương trình được phân phối lại. Đặc biệt, nó bắt mã nguồn phải được đưa ra, (bao gồm cả những sửa đổi trong trường hợp của một công việc dẫn xuất) và phân phối lại thì buộc phải đưa vào cùng những điều kiện y hệt cho người nhận;

✓ eCos License 2.0. Đây là giấy phép mà theo đó eCos, một hệ điều hành thời gian thực, được phân phối. Nó là một sửa đổi của GNU GPL mà không coi mã nguồn được liên kết tới các chương trình mà nó bảo vệ, phải tuân theo các điều khoản của GNU GPL nếu được phân phối lại. Từ quan điểm này, những hiệu lực của nó là tương tự như của GNU LGPL;

✓ Affero General Public License. Đây là một sửa đổi thú vị của GNU GPL mà nó coi trường hợp các chương trình đưa ra các dịch vụ thông qua web, hoặc nói chung, thông qua các mạng máy tính. Dạng chương trình này đại diện cho một vấn đề từ quan điểm của các giấy phép mạnh. Vì việc sử dụng chương trình không ngụ ý phải nhận nó thông qua một sự phân phối lại, ngay cả dù nó được cấp phép theo GNU GPL, ví dụ vậy, mà không có việc phân phối lại theo bất kỳ cách thức nào, và vì thế, không có bổn phận, ví dụ vậy, phải phân phối mã nguồn của nó. Affero GPL có một mệnh đề bắt buộc rằng nếu chương trình có một phương tiện cho việc đưa ra mã nguồn của nó thông qua web cho bất kỳ ai sử dụng nó; tính năng này có thể không được vô hiệu hóa. Điều này có nghĩa là nếu tác giả gốc ban đầu đưa khả năng này vào trong mã nguồn, thì bất kỳ người sử dụng nào cũng có thể có được nó, và cộng với việc phân phối lại phải tuân thủ các điều kiện của giấy phép này. FSF đang xem xét đưa vào các khoản tương tự vào phiên bản 3 của GNU GPL;

✓ IBM Public License 1.0. Đây là một giấy phép mà nó cho phép một sự phân phối lại nhị phân của các công việc dẫn xuất chỉ nếu (giữa những điều kiện khác) một cơ chế được định sẵn trước cho người nhận chương trình để nhận mã nguồn. Sự phân phối lại mã nguồn phải được thực hiện theo cùng giấy phép này. Giấy phép này cũng thú vị vì nó bắt bên tham gia phân phối lại chương trình với những sửa đổi phải cấp phép tự động và không được lấy tiền đối với bất kỳ bằng sáng chế nào có ảnh hưởng tới những sửa đổi như vậy và chúng là sở hữu của người phân phối cho bên nhận chương trình;

✓ Mozilla Public License 1.1 là một ví dụ về giấy phép tự do được thiết kế bởi một công ty. Đây là một sự tiến bộ của giấy phép tự do đầu tiên mà Netscape Navigator đã có, mà nó đã từng là rất quan trọng trong những ngày đó vì nó đã là lần đầu tiên mà một công ty nổi tiếng đã quyết định phân phối một chương trình theo giấy phép tự do của riêng hãng.

1.2.4. Các môi trường và công nghệ phát triển phần mềm tự do nguồn mở cũng như ứng dụng của chúng

Những đặc tính chung của môi trường và công nghệ phát triển phần mềm tự do nguồn mở:

- ✓ Đầu tiên, dù không nhất thiết là một yếu tố quyết định, thì thông thường đối với môi trường, các công cụ phát triển (và ngay cả máy tính ảo đích, nếu có) đều là tự do. Điều này không phải lúc nào cũng vậy. Ví dụ, dự án GNU, với mục tiêu để thay thế Unix, phải được phát triển trong và cho những hệ thống Unix sở hữu độc quyền cho tới khi Linux và FreeBSD xuất hiện. Hiện nay, đặc biệt khi PMTD được phát triển như một phần của một mô hình kinh doanh, thì xu thế là việc máy tính đích cũng có thể là một hệ thống sở hữu độc quyền, thường thông qua những máy ảo được xen kẽ (Java, Python, PHP, …). Trong mọi trường hợp, môi trường và máy ảo cần phải đủ chung và rẻ để cùng mang đến cho các đồng lập trình viên đủ để có cùng các công cụ.
- ✓ Thứ 2, cũng để lôi cuốn số lượng lớn nhất có thể các đồng lập trình viên, các công cụ cần phải là đơn giản, nổi tiếng và có khả năng hoạt động trong các máy tiết kiệm. Có lẽ vì những lý do này mà thế giới PMTD khá là bảo thủ khi nói về các ngôn ngữ, công cụ và môi trường.
- ✓ Thứ 3, mô hình phát triển của PMTD có xu hướng được phân tán cao độ, với nhiều người cộng tác tiềm năng trải rộng ra khắp thế giới. Vì lý do này thường các công cụ cộng tác không đồng bộ là cần thiết, mà chúng cùng một lúc cho phép sự phát triển tiến bộ dễ dàng, bất chấp số lượng và nhịp độ công việc của từng cộng tác viên với hàng loạt kiến trúc khác nhau trong đó chúng có thể biên dịch và kiểm thử các chương trình của chúng [3].

1.2.4.1. Ngôn ngữ

Hầu hết PMTD được viết trong ngôn ngữ C, không chỉ vì C là ngôn ngữ tự nhiên của bất kỳ biến thể Unix nào (nền tảng thường thấy của PMTD), mà cũng vì nó được phổ biến rộng rãi, cả trong tâm trí của mọi người. Các ngôn ngữ khá tương tự khác là C++, Java cũng phổ biến vì nó cho phép những phát triển trên nhiều nền tảng.

1.2.4.2. Môi trường phát triển tích hợp

Môi trường phát triển tích hợp là một hệ thống mà nó làm cho công việc của lập trình viên phần mềm dễ dàng hơn bằng việc tích hợp việc xuất bản có định hướng các

ngôn ngữ, trình biên dịch hoặc phiên dịch, dò tìm và sửa lỗi, đo lường sự thực thi, hợp nhất các mã nguồn tới một hệ thống kiểm soát nguồn, …, thường theo dạng các module. Một số môi trường: Eclipse, Kdevelop, Anjuta, Netbeans, Code::Blocks, GitHub.

1.2.4.3. Các cơ chế cộng tác cơ bản

PMTD là một hiện tượng bởi sự cộng tác của các cộng đồng phân tán và vì thế, đòi hỏi các công cụ để làm cho sự cộng tác đó có hiệu quả.

Giữa thập kỷ 70, UUCP (giao thức truyền tệp của Unix) đã được phát triển cho các máy tính giao tiếp thông qua quay số dial - up và các đường dây chuyên dụng, và trên đó thư điện tử đã được xây dựng, và vào năm 1979, kết nối đầu tiên của USENET qua UUCP. thông tin của USENET, một hệ thống diễn đàn có cấu trúc thứ bậc được phân tán bằng việc gây ngập tràn tới các site được sắp xếp theo thứ bậc, đã đóng một vai trò cơ bản trong sự phát triển của PMTD.

Hiện hành, với sự phổ biến của web, nhiều nhóm thảo luận là những nhóm thảo luận thuần túy web hoặc weblogs, như ShashDot, Barrapunto hoặc wikis, để xây dựng một tài liệu cộng tác, như là đặc tả kỹ thuật cho một chương trình, một module hoặc một hệ thống.

Cuối cùng, phải nhắc tới các cơ chế tương tác được sử dụng bởi các lập trình viên để nói chuyện trong thời gian thực. Công cụ thường được sử dụng nhất là IRC (Internet Relay Chat).

1.2.4.4. Quản lý mã nguồn

Bất kỳ dự án phát triển chương trình nào phải lưu trữ lịch sử của nó, vì một sửa đổi có thể tạo ra một lỗi ẩn được phát hiện sau này. Nếu dự án được phát triển bởi vài người, thì tác giả của từng thay đổi cũng sẽ cần phải được ghi lại. Nếu các phiên bản có đánh số của một dự án được thực hiện, thì chúng ta cần biết chính xác những phiên bản nào của từng module tạo nên một phần của từng phiên bản. Thường thì, một dự án sẽ giữ một phiên bản ổn định và phiên bản thực nghiệm khác; cả 2 phiên bản này cần phải được duy trì, gỡ rối, và sửa các lỗi được truyền từ phiên bản này sang phiên bản khác. Tất cả những điều này có thể được thực hiện bằng việc lưu

và dán nhãn cho mỗi tệp của phiên bản. Những gì mà một hệ thống kiểm soát nguồn, còn được biết tới như một hệ thống quản lý phiên bản, thường làm, là để lưu giữ lịch sử các tệp như một tập hợp các khác biệt đối với một phiên bản, thường là phiên bản gần nhất, vì tính hiệu quả, cũng như việc dán nhãn cho từng sự khác biệt với siêu dữ liệu cần thiết. Một số hệ thống quản lý mã nguồn tiêu biểu: CVS, Subversion, arch, bazaar, BitKeeper, Git.

1.2.4.5. Tài liệu

DocBook: DocBook là một ứng dụng SGML ban đầu được phát triển cho các tài liệu kỹ thuật về công nghệ thông tin và bây giờ là một phương án của XML. Hiện tại, DocBook là chuẩn định dạng tài liệu tự do cho nhiều dự án (Linux Documentation Project, KDE, GNOME, Mandriva Linux, etc.) và là một mục tiêu để đạt được đối với những dự án khác (Linux, *BSD, Debian, etc).

Wikis: một cơ chế về sự cộng tác cho sự chuẩn bị tài liệu trực tuyến thông qua web đã trở nên phổ biến, được gọi là wiki, và được sáng tạo bởi Ward Cunningham. Lần đầu tiên được đưa vào phục vụ năm 1995 và bây giờ được sử dụng nhiều trong việc chuẩn bị các tài liệu rất năng động, không được thiết kế cho việc in ấn và thường với một vòng đời ngắn (ví dụ, tổ chức hội nghị).

1.2.4.6. Quản lý lỗi và các vấn đề khác

Một trong những điểm mạnh của mô hình phát triển tự do là việc cộng đồng đóng góp với những báo cáo lỗi và cảm thấy rằng những báo cáo hoặc giải pháp này đưa ra được sự chú ý. Điều này đòi hỏi một cơ chế báo cáo lỗi đơn giản, sao cho các lập trình viên có thể nhận được những thông tin đầy đủ, theo một cách có hệ thống và chứa đựng tất cả các chi tiết cần thiết, hoặc được cung cấp bởi người đóng góp, với một sự giải thích về những gì đang xảy ra, mức độ quan trọng và giải pháp có thể, hoặc thông qua một cơ chế tự động mà nó xác định, ví dụ, phiên bản và môi trường của chương trình trong đó nó hoạt động. Các lỗi cũng phải được lưu giữ trong một cơ sở dữ liệu mà có thể tra cứu được, để xem liệu một lỗi đã được giao tiếp, được sửa hay chưa, mức độ quan trọng của nó, … Có một vài hệ thống như thế này, một số thông qua web, số khác thông qua thư điện tử, thông qua một số chương trình trung gian. Tất cả chúng có một giao diện web cho tra cứu. Một số cho phép các báo cáo

nặc danh, trong khi số khác yêu cầu sự xác thực (một địa chỉ thư điện tử hợp lệ) để ngăn ngừa nhiễu. Mặc dù những thủ tục web dường như là đơn giản nhất, chúng không dễ dàng có được thông tin một cách tự động trong môi trường lỗi. Ví dụ, hệ thống của Debian cung cấp các chương trình như báo cáo lỗi reportbug.

1.2.4.7. Hỗ trợ cho các kiến trúc khác

Tối thiểu nhất các trình biên dịch phải cho phép chương trình được biên dịch trên các kiến trúc và các hệ điều hành khác nhau. Ví dụ, SourceForge đã xuất ra các môi trường của Debian GNU/Linux cho Intel x86, DEC Alpha, PowerPC và SPARC, và đã bổ sung thêm vào các môi trường của Solaris và Mac OS/X.

1.2.4.8. Các site hỗ trợ phát triển:

Về dạng dịch vụ này, một trong những thứ đầu tiên đã được thiết lập, và nổi tiếng nhất, là SourceForge ngoài ra còn có Savannah chuyên dụng cho dự án GNU và cho các chương trình khác với các giấy phép dạng copyleft, hoặc BerliOS - một điểm họp cho các lập trình viên và các công ty PMTD, Launchpad được sử dụng bởi Ubuntu cho việc phát triển từng phiên bản của phát tán này.

Kết luận chương

Công nghệ ảo hóa là một công nghệ được ra đời nhằm khai thác triệt để khả năng làm việc của các phần cứng trong một hệ thống máy chủ. Tuy nhiên nó cũng có những ưu và nhược điểm nhất định do vậy tùy vào từng hệ thống máy chủ mà thiết kế hệ thống ảo hóa phù hợp như VMM-Hypervisor, Hybrid, Monolithic Hypervisor, Microkernelized Hypervisor. Công nghệ ảo hóa phục vụ cho nhiều mục đích phong phú như: hợp nhất máy chủ, hợp nhất cho các môi trường triển khai và thử nghiệm, đơn giản hóa kế hoạch đối phó và khôi phục thảm họa, một trung tâm dữ liệu động, ...

Một hệ thống ảo hóa bao gồm các thành phần: tài nguyên vật lý, các phần mềm ảo hóa, máy ảo, hệ điều hành. Các công nghệ hỗ trợ ảo hóa bao gồm: công nghệ cân bằng tải, cân bằng tải mạng, cân bằng clustering, công nghệ RAID, công nghệ lưu trữ mạng SAN. Các kiểu ảo hóa cơ bản bao gồm: ảo hóa hệ thống mạng, ảo hóa hệ thống lưu trữ, ảo hóa ứng dụng, ảo hóa hệ thống máy chủ.

Trong các công nghệ ảo hóa ứng dụng, công nghệ ảo hóa Docker là sản phẩn của một dự án phần mềm tự do nguồn mở phát hành theo giấy phép Apache, công nghệ này đang được đánh giá là tương lai của công nghệ ảo hoá ứng dụng. Công nghệ này sẽ được trình bày chi tiết trong chương 2.

Phần mềm tự do nguồn mở có nhiều ưu thế so với phần mềm nguồn đóng như: tính an toàn, ổn định và đáng tin cậy, giảm lệ thuộc nhà cung cấp, không bị vấn đề vi phạm bản quyền, sở hữu trí tuệ, bản địa hóa, các tiêu chuẩn mở. Phần mềm tự do nguồn mở được phát hành theo giấy phép, được chia làm 2 loại giấy phép dễ dãi và giấy phép mạnh. Phát triển phần mềm tự do nguồn mở phải tuân thủ các môi trường và công nghệ về: ngôn ngữ, môi trường tích hợp, các cơ chế cộng tác cơ bản, quy định quản lý mã nguồn, tài liệu, quản lý lỗi và phiên bản, hỗ trợ các kiến trúc khác, hỗ trợ các site phát triển.

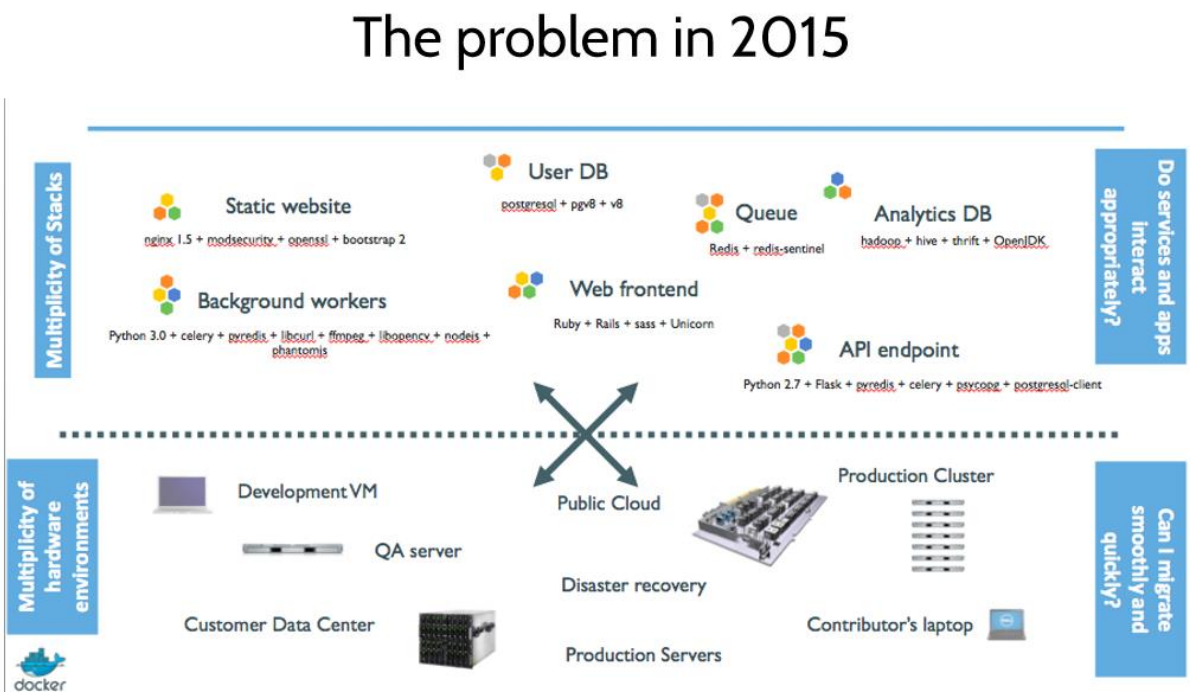
CHƯƠNG 2
CÔNG NGHỆ ẢO HÓA DOCKER

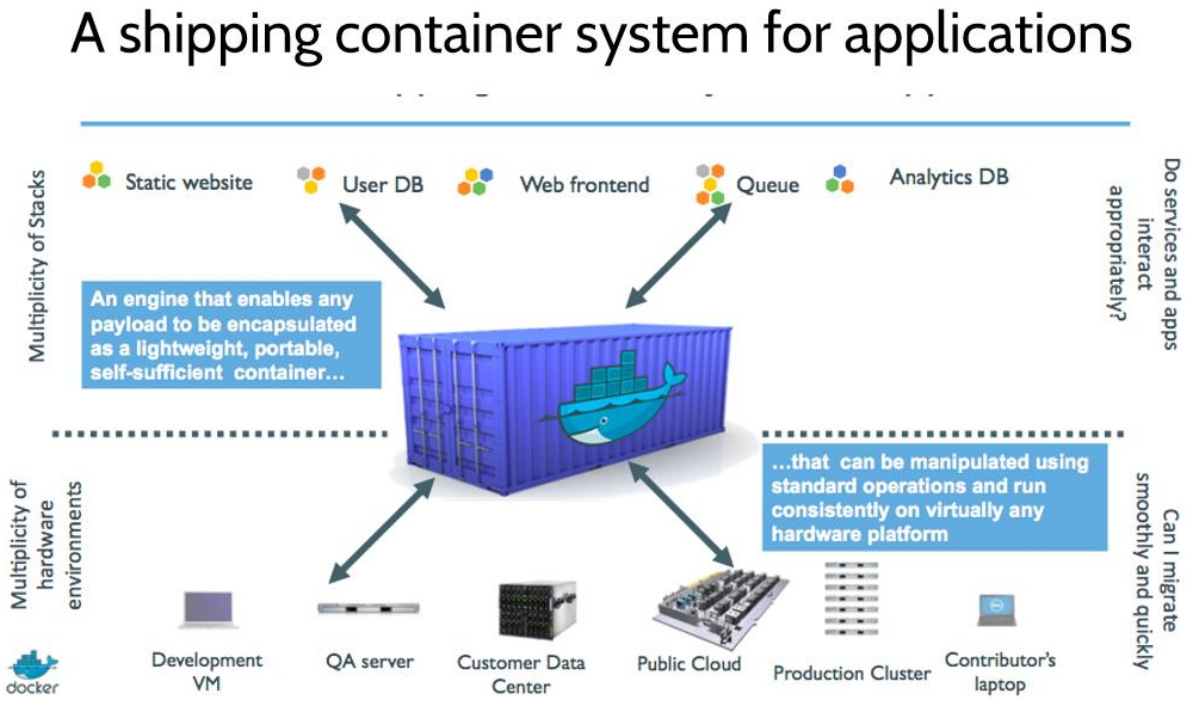
2.1. Khái niệm về Công nghệ ảo hóa Docker

2.1.1. Định nghĩa

Docker là một nền tảng mở dành cho các lập trình viên, quản trị hệ thống dùng để xây dựng, vận chuyển và chạy các ứng dụng phân tán. Ban đầu viết bằng Python, hiện tại đã chuyển sang Go-lang [7].

Docker đưa ra một giải pháp mới cho vấn đề ảo hóa, thay vì tạo ra các máy ảo con chạy độc lập kiểu hypervisors (tạo phần cứng ảo và cài đặt hệ điều hành lên đó), các ứng dụng sẽ được đóng gói lại thành các Container (Công-ten-nơ) riêng lẻ. Các Container này chạy chung trên nhân hệ điều hành qua LXC (Linux Containers), chia sẻ chung tài nguyên của máy mẹ, do đó, hoạt động nhẹ và nhanh hơn các máy ảo dạng hypervisors [10].





Hình 2. 1: Công nghệ ảo hóa Docker

2.1.2. Các thành phần chính

Các thành phần chính của Docker bao gồm:

- ✓ Docker Engine: là thành phần chính của Docker, như một công cụ để đóng gói ứng dụng;
- ✓ Docker Hub: là dịch vụ cloud để chia sẻ ứng dụng và tự động hóa chuỗi các công việc liên tục, có thể thao tác pull/push với các images.

2.1.3. Một số khái niệm

Một số khái niệm phổ biến về Docker:

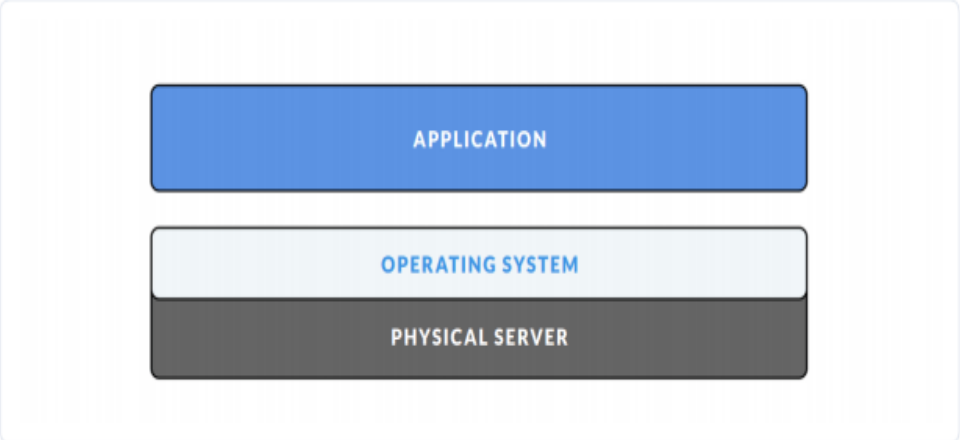
- ✓ Docker images: là một "read-only template". Chẳng hạn, một image chứa hệ điều hành Ubuntu đã cài đặt sẵn Apache và ứng dụng web;
- ✓ Docker registries: là kho chứa images. Người dùng có thể tạo ra các images của mình và tải lên đây hoặc tải về các images được chia sẻ;
- ✓ Docker container: hoạt động giống như một thư mục (directory), chứa tất cả những thứ cần thiết để một ứng dụng có thể chạy được. Mỗi một docker

container được tạo ra từ một docker image. Các thao tác với một container: chạy, bật, dừng, di chuyển, và xóa;

- ✓ Dockerfile: là một file chứa tập hợp các lệnh để Docker có thể đọc và thực hiện để đóng gói một image theo yêu cầu người dùng;
- ✓ Orchestration : là các công cụ, dịch vụ dùng để điều phối và quản lý nhiều containers sao cho chúng làm việc hiệu quả nhất[8].

2.1.4. So sánh Docker với Virtual machine

Ngày xưa, mô hình máy chủ thường là 1 máy chủ vật lý + 1 hệ điều hành (OS) + 1 application.

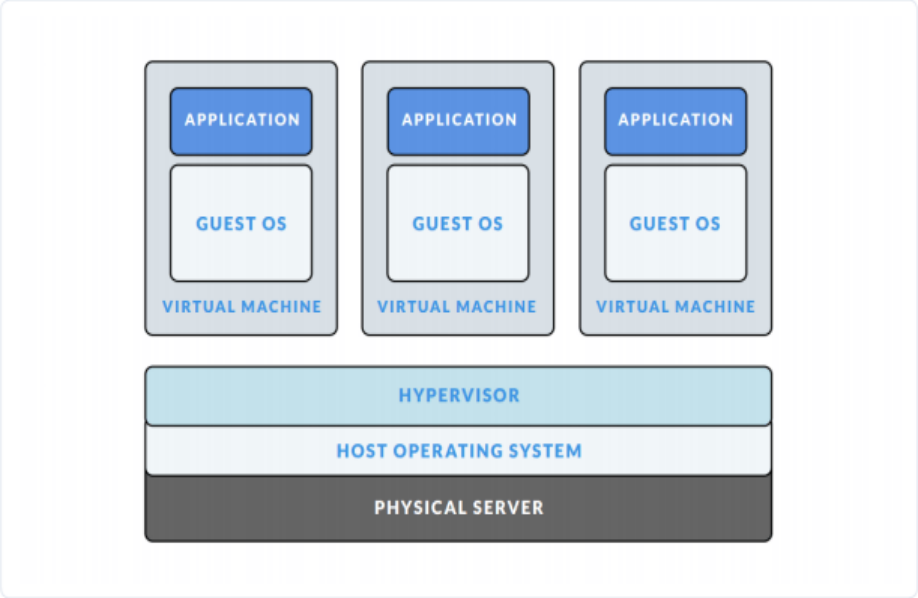


Hình 2. 2: Mô hình máy chủ truyền thống

Khi ứng dụng phát triển lên, mô hình này nảy sinh ra nhiều vấn đề, ví dụ:

- ✓ Lãng phí tài nguyên: mặc dù cấu hình máy khỏe, ổ cứng dung lượng lớn, nhưng hệ thống lại không tận dụng được hết lợi thế này;
- ✓ Khó khăn trong việc mở rộng hệ thống: muốn mở rộng phải thuê thêm server, cấu hình, cân bằng tải (load balacing), …

Lúc này, công nghệ ảo hóa (vitualization) ra đời.

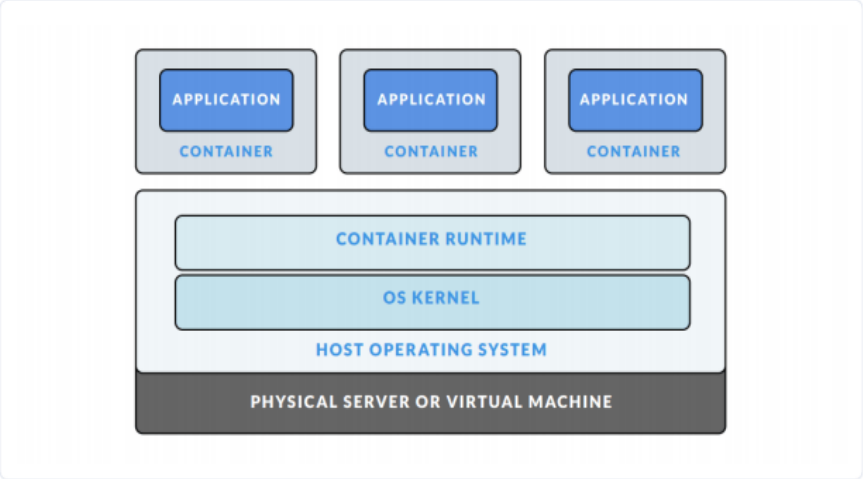


Hình 2. 3: Mô hình máy ảo VMs

Với công nghệ ảo hóa, trên cùng 1 máy chủ vật lý, có thể tạo ra nhiều OS, tức là sẽ chạy được nhiều application. Vậy là tài nguyên của máy được tận dụng tốt hơn. Tuy nhiên, việc ảo hóa này lại nảy sinh vấn đề mới:

- ✓ Ngốn tài nguyên: khi chạy 1 máy ảo, nó sẽ luôn chiếm 1 phần tài nguyên cố định. Vd: máy chủ có 512GB SSD, 16GB RAM. Tạo ra 4 máy ảo Linux, mỗi máy cấp 64GB SSD và 2GB RAM. Như vậy, sẽ mất 256 GB SSD để chứa 4 máy ảo, và khi chạy cùng 4 máy ảo lên cùng lúc, chúng sẽ chiếm 8GB RAM. Mặc dù chỉ chạy lên để không đó thôi, chưa dùng gì cả nhưng nó vẫn chiếm từng đó;
- ✓ Tốn thời gian thực thi: thời gian khởi động, shutdown của các máy ảo sẽ lâu, thường là hàng phút;
- ✓ Cồng kềnh: việc phải chịu tải cho cả 1 nhóm máy ảo như vậy thì server không thể chạy hết hiệu suất được.

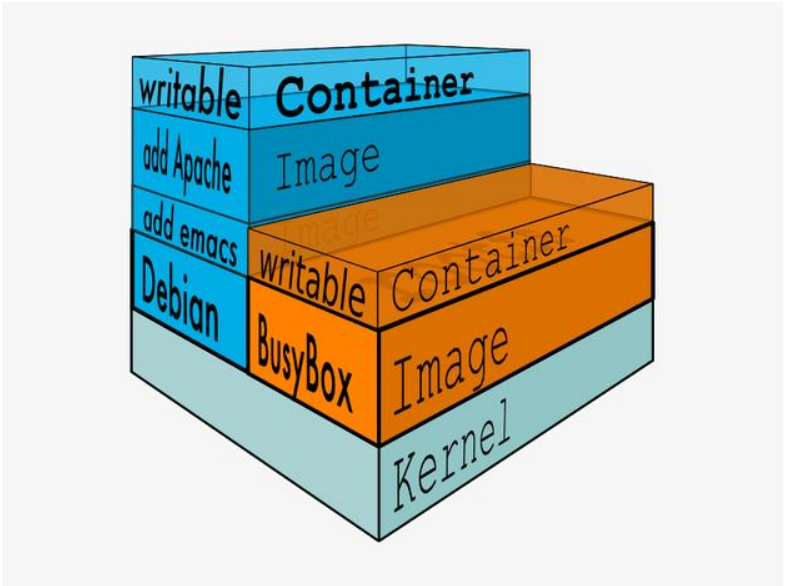
Bước tiến hóa tiếp theo, người ta phát minh ra containerlization



Hình 2. 4: Mô hình ảo hóa Container

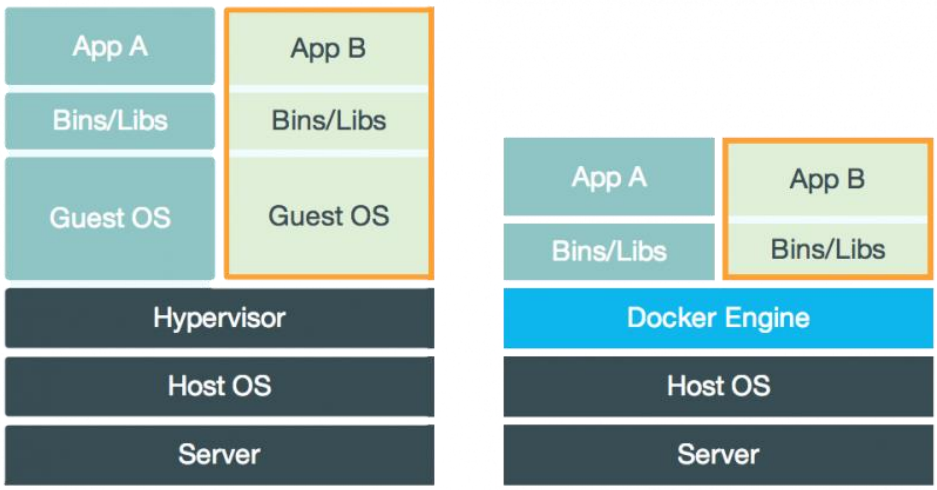
Với công nghệ này, trên một máy chủ, ta sẽ sinh ra được nhiều máy con (giống với ảo hóa), nhưng điều đặc biệt là các máy con (Guess OS) này đều dùng chung phần nhân của máy mẹ (host OS) và chia sẻ với nhau tài nguyên của máy mẹ (RAM chẳng hạn). Như vậy việc tận dụng tài nguyên sẽ được tối ưu hơn.

Ngoài ra, việc sử dụng hệ thống file cắt lớp (layer file system) sẽ khiến việc tối ưu tài nguyên hiệu quả hơn.



Hình 2. 5: Hệ thống file cắt lớp Container

Cụ thể, mỗi máy con (container) mới, nó sẽ được xây dựng dựa trên 1 file ảnh (image) dạng chỉ đọc (read-only). Trong mỗi máy con sẽ có thêm 1 lớp bọc có-thể-ghi-được (writabe-layer), các thay đổi trong máy con sẽ được ghi lên đây. Như vậy, từ 1 image ban đầu, ta có thể tạo nhiều máy con mà chỉ tốn rất ít dung lượng ổ đĩa.



Hình 2. 6: Khác biệt giữa Docker và VMs

Điểm khác biệt chính là các containers sử dụng chung kernel với Host OS nên các thao tác bật, tắt rất nhẹ nhàng, nhanh chóng.

- ✓ Ưu điểm: nhanh, nhẹ, có thể chia sẻ dễ dàng qua DockerHub;
- ✓ Nhược điểm : mới, cập nhật thay đổi thường xuyên.

2.2. Cài đặt, sử dụng Docker

2.2.1. Cài đặt Docker

Sau đây là quá trình cài đặt Docker trên Ubuntu 12.04:

Bước 1- Upgrade kernel 3

```
$ sudo apt-get update

$ sudo apt-get install

$ linux-image-generic-lts-raring
```

```
$ linux-headers-generic-lts-raring
```

```
$ reboot
```

Bước 2- update package

```
$ sudo apt-get update
```

Bước 3- install docker

```
$ sudo apt-get install linux-image-generic-lts-trusty
```

Bước 4- reboot

```
$ sudo reboot
```

2.2.2. Sử dụng Docker

1-Quản lý images/containers

Các lệnh thường dùng để quản lý images/containers:

- ✓ List các images đang có:

  `$ docker images -a`

- ✓ List các containers:

  `$ docker ps -a`

- ✓ List các container đang chạy:

  `$ docker ps`

- ✓ Remove image:

  `$ docker rmi < image id>`

- Remove container:

```
$ docker rm < container id>
```

2-Dùng Dockerfile build image

Nội dung của 1 Dockerfile để cài openssh-server trên ubuntu 12.04 image

```
$ cat Dockerfile
```

-- FROM: base on this image

FROM ubuntu:12.04

-- MAINTAINER: Dockerfile author

MAINTAINER Vu Trong Chien

-- ENV: to set an environment variable

ENV USER root

ENV HOME /root

-- RUN: run a command

RUN apt-get update

RUN apt-get install -y openssh-server

RUN mkdir /var/run/sshd

RUN echo "root:123456" | chpasswd

-- EXPOSE: exposes port for container

EXPOSE 22

-- ENTRYPOINT: make container executable with command

ENTRYPOINT /usr/sbin/sshd -D

cd đến folder chứa Dockerfile và dùng **docker build -t < image tag> .** để build image

```
docker build -t vtchien/ssh .
```

Uploading context 10.24 kB

Step 1 : FROM ubuntu:12.04

Pulling repository ubuntu

8dbd9e392a96: Download complete

---> 8dbd9e392a96

Step 2 : MAINTAINER Vu Trong Chien

---> Running in 99ea761f56a8

---> 62e7555f00bd

Step 3 : ENV USER root

---> Running in 1f190f72706a

---> 6c558132b77f

Step 4 : ENV HOME /root

---> Running in 44c613dd2012

---> b58d9d956fb7

Step 5 : RUN apt-get update

---> Running in 25ed0a4b26ac

.....

---> 3c82fd19a88a

Step 6 : RUN apt-get install -y openssh-server

---> Running in 6370346bc2bd

.....

---> be807df511bd

Step 7 : RUN mkdir /var/run/sshd

---> Running in 161ff5c313d0

---> 13cc03230011

Step 8 : RUN echo "root:123456" | chpasswd

---> Running in bf827247b5f4

---> 8eedcdc3afe9

Step 9 : EXPOSE 22

---> Running in f70d33303fff

---> 47728b435ca6

Step 10 : ENTRYPOINT /usr/sbin/sshd -D

---> Running in 61b8e19d9574

---> 8a1bc4691754

Successfully built 8a1bc4691754

3-Kiểm tra kết quả:

```
# docker images -a
```

REPOSITORY | TAG | IMAGE ID | CREATED | VIRTUAL SIZE|
|
< none> | < none> | 8eedcdc3afe9 | 23 seconds ago | 181.7 MB
vtchien/ssh | latest | 8a1bc4691754 | 23 seconds ago | 181.7 MB
< none> | < none> | 13cc03230011 | 23 seconds ago | 181.7 MB
< none> | < none> | 47728b435ca6 | 23 seconds ago | 181.7 MB
< none> | < none> | be807df511bd | 24 seconds ago | 181.7 MB
< none> | < none> | 3c82fd19a88a | 30 seconds ago | 157 MB
< none> | < none> | 62e7555f00bd | 31 seconds ago | 128 MB
< none> | < none> | 6c558132b77f | 31 seconds ago | 128 MB
< none> | < none> | b58d9d956fb7 | 31 seconds ago | 128 MB
ubuntu | 12.04 | 8dbd9e392a96 | 8 months ago | 128 MB

```
# docker ps -a
```

CONTAINER ID | IMAGE | COMMAND | CREATED | STATUS | PORTS | NAMES|
|
|61b8e19d9574 | 47728b435ca6 | /bin/sh -c /usr/sbin | 56 seconds ago | Exit 0 | |
f70d33303fff | 8eedcdc3afe9 | /bin/sh -c # (nop) EX | 56 seconds ago| Exit 0 | |
bf827247b5f4 | 13cc03230011 | /bin/sh -c echo "roo | 56 seconds ago | Exit 0 | |
161ff5c313d0 | be807df511bd | /bin/sh -c mkdir /va | 57 seconds ago | Exit 0 | |
6370346bc2bd | 3c82fd19a88a | /bin/sh -c apt-get i | About a minute ago| Exit 0 | |
25ed0a4b26ac | b58d9d956fb7 | /bin/sh -c apt-get u | About a minute ago | Exit 0 | |
44c613dd2012 | 6c558132b77f | /bin/sh -c #(nop) EN | About a minute ago | Exit 0 |
1f190f72706a | 62e7555f00bd | /bin/sh -c #(nop) EN | About a minute ago | Exit 0 | |
99ea761f56a8 | ubuntu:12.04 | /bin/sh -c #(nop) MA | About a minute ago| Exit 0 | |

```
# docker ps
```

CONTAINER ID | IMAGE | COMMAND | CREATED | STATUS | PORTS | NAMES
|

docker ps cho thấy chưa có container nào chạy.
run image và assign port:

```
# docker run -d -p 22 vtchien/ssh
```
be7238580063e1c5da400fe35b4d45497aac4d83850eb00d7a601098da500e13

```
# docker port
```

be7238580063e1c5da400fe35b4d45497aac4d83850eb00d7a601098da500e13 22
0.0.0.0:49153

```
# docker ps
```

CONTAINER ID | IMAGE | COMMAND | CREATED | STATUS | PORTS | NAMES
|
be7238580063 | vtchien/ssh:latest | /bin/sh -c /usr/sbin | 15 seconds ago | Up 15 seconds | 0.0.0.0:49153->22/tcp |

Giờ đã thấy container **be7238580063** đang chạy và port **49153** của host được map với port 22 của container.

Ta có thể ssh vào container qua port 49153 với user **root** và pass **123456** đã khai báo trong Dockerfile:

```
# ssh root@0.0.0.0 -p 49153
```

The authenticity of host '[0.0.0.0]:49153 ([0.0.0.0]:49153)' can't be established.

ECDSA key fingerprint is 5d:c0:af:8b:64:b2:c4:71:0c:35:5d:0e:29:9d:87:00.

Are you sure you want to continue connecting (yes/no)? yes

Warning: Permanently added '[0.0.0.0]:49153' (ECDSA) to the list of known hosts.

root@0.0.0.0's password:

Welcome to Ubuntu 12.04 LTS (GNU/Linux 3.8.0-29-generic x86_64)

** Documentation: https://help.ubuntu.com/

The programs included with the Ubuntu system are free software;

the exact distribution terms for each program are described in the individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by applicable law.

```
# ip a

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN

link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00

inet 127.0.0.1/8 scope host lo

inet6 ::1/128 scope host

   valid_lft forever preferred_lft forever

25: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000

link/ether 2a:5c:64:3c:f7:5a brd ff:ff:ff:ff:ff:ff

inet 172.17.0.6/16 scope global eth0

inet6 fe80::285c:64ff:fe3c:f75a/64 scope link

   valid_lft forever preferred_lft forever

# uname -a

Linux be7238580063 3.8.0-29-generic #42~precise1-Ubuntu SMP Wed Aug 14 16:19:23 UTC 2013 x86_64 x86_64 x86_64 GNU/Linux
```

```
# ping 8.8.8.8

-bash: ping: command not found

# ls /

bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin
selinux  srv  sys  tmp  usr  var
```

4-Debug

Log của docker lưu lại **/var/log/upstart/docker.log**
Cần quyền root để xem log này.

Log level mặc định là INFO, muốn chuyển sang DEBUG cần tắt service và khởi động bằng tay với **option –D.**

```
$sudo initctl stop docker

$sudo /usr/bin/docker -d -D
```

2.3. Các thành phần và công nghệ ảo hóa ứng dụng trong Docker

2.3.1. Các thành phần

1-Docker images:

- ✓ Là file ảnh, file nền của một hệ điều hành, một nền tảng, một ngôn ngữ (vd: ubuntu image, ruby image, rails image, mysql image…);
- ✓ Từ các image này, bạn sẽ dùng nó để tạo ra các container;
- ✓ Các image là dạng file-chỉ-đọc (read only file);
- ✓ Tự bạn cũng có thể tạo image cho riêng mình;
- ✓ Một image có thể được tạo từ nhiều image khác (vd: bạn tạo 1 image chạy ubuntu, có cài sẵn ruby 2.3 và rails 5, image này của bạn được tạo nên bởi 3 image khác).

2-Docker registries: Là kho chứa images. Người dùng có thể tạo ra các images của mình và tải lên đây hoặc tải về các images được chia sẻ.

3-Docker container: hoạt động giống như một thư mục (directory), chứa tất cả những thứ cần thiết để một ứng dụng có thể chạy được. Mỗi một docker container được tạo ra từ một docker image. Các thao tác với một container: chạy, bật, dừng, di chuyển, và xóa.

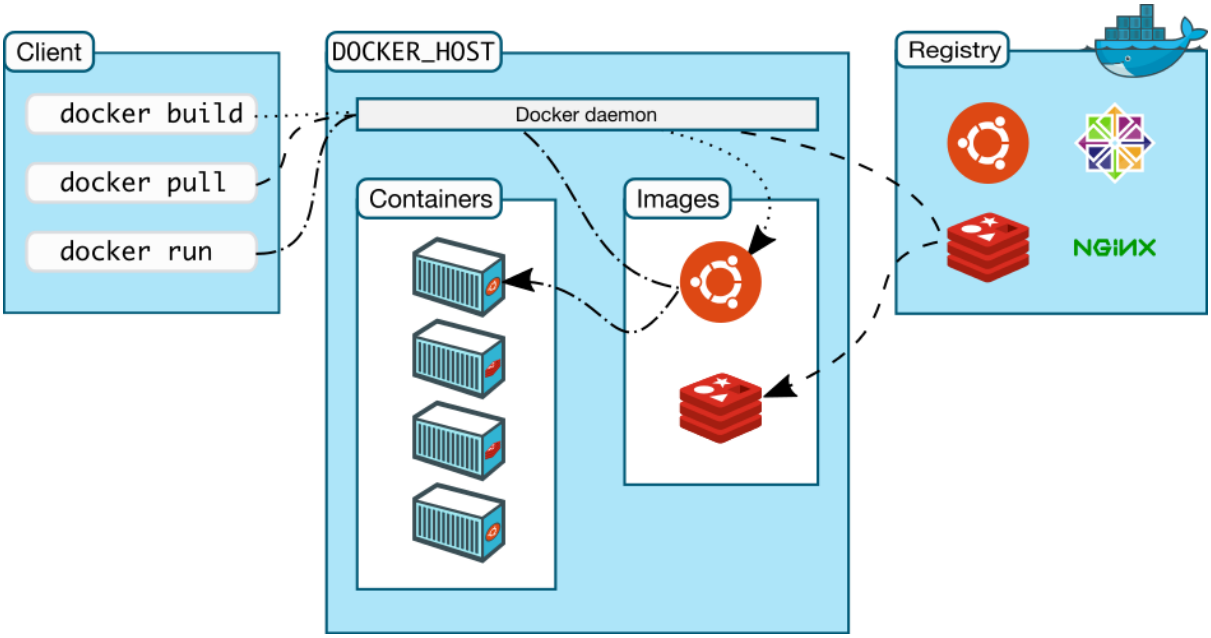
- ✓ Là một máy ảo, được cấu thành từ 1 image và được đắp thêm 1 lớp "trang trí" writable-file-layer. Các thay đổi trong máy ảo này (cài thêm phần mềm, tạo thêm file…) sẽ được lưu ở lớp trang trí này;
- ✓ Các container này sẽ dùng chung tài nguyên của hệ thống (RAM, Disk, Network…), chính nhờ vậy, những container của bạn sẽ rất nhẹ, việc khởi động, kết nối, tương tác sẽ rất nhanh gọn;
- ✓ Nếu ánh xạ sang hướng đối tượng, thì image chính là class, còn container chính là instance-1 thể hiện của class đó. Từ 1 class ta có thể tạo ra nhiều instance, tương tự, từ 1 image ta cũng có thể tạo ra được nhiều container hoàn toàn giống nhau.

4-Dockerfile: là một file chứa tập hợp các lệnh để Docker có thể đọc và thực hiện để đóng gói một image theo yêu cầu người dùng.

5-Orchestration : là các công cụ, dịch vụ dùng để điều phối và quản lý nhiều containers sao cho chúng làm việc hiệu quả nhất.

2.3.2. Kiến trúc của Docker

Docker sử dụng kiến trúc client-server. Docker client sẽ nói liên lạc với các Docker daemon, các Docker daemon sẽ thực hiện các tác vụ build, run và distribuing các Docker container. Cả Docker client và Docker daemon có thể chạy trên cùng 1 máy, hoặc có thể kết nối theo kiểu Docker client điều khiển các docker daemon như hình dưới. Docker client và daemon giao tiếp với nhau thông qua socker hoặc RESTful API.



Hình 2. 7: Kiến trúc Docker

1-Docker Daemon

Như thể hiện trên biểu đồ phía trên, Docker daemon chạy trên các máy host. Người dùng sẽ không tương tác trực tiếp với các daemon, mà thông qua Docker Client.

2-Docker Client

Là giao diện người dùng của Docker, nó cung cấp cho người dùng giao diện dòng lệnh và thực hiện phản hồi với các Docker daemon.

3-Docker images

Là một template chỉ cho phép đọc, ví dụ một image có thể chứa hệ điều hành Ubuntu và web app. Images được dùng để tạo Docker container. Docker cho phép chúng ta build và cập nhật các image có sẵn một cách cơ bản nhất, hoặc bạn có thể download Docker images của người khác.

4-Docker Container

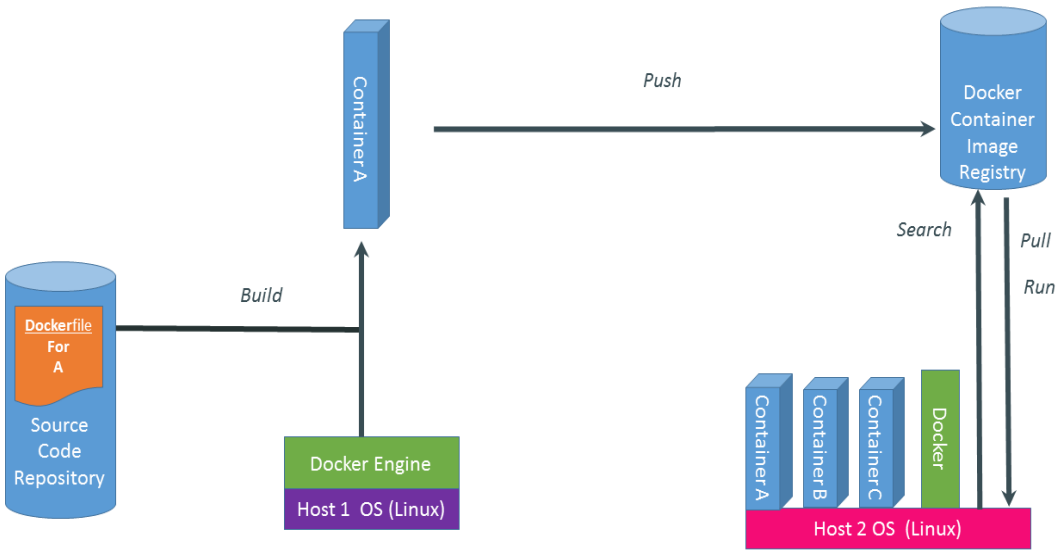
Docker container có nét giống với các directory. Một Docker container giữ mọi thứ chúng ta cần để chạy một app. Mỗi container được tạo từ Docker image. Docker container có thể có các trạng thái run, started, stopped, moved và deleted.

2.3.3. Ưu điểm hình thức đóng gói thành Container.

Việc đóng gói thành các container này có thể giải quyết được nhiều vấn đề mà ta chưa đề cập tới.

- ✓ Ví dụ như trước kia ta không thể dùng chung Port, thì ở đây 2 ứng dụng với 2 container khác nhau. Ta có thể cấu hình Port trùng nhau cho ứng dụng này;
- ✓ Tiếp theo là về việc quản lí phiên bản. Ta khó có thể cài 2 phiên bản của 1 phần mềm trên cùng 1 máy hypervisor. Tuy nhiên với Container, ta có thể cài mỗi phiên bản trên 1 Container và chạy một cách trơn tru;
- ✓ Khả năng khởi động nhanh của Docker cũng là một lợi thế rất lớn;
- ✓ Tiếp theo nói về tài nguyên, Docker sẽ ngốn ít tài nguyên hơn các máy hypervisor.

2.3.4. Quy trình thực thi của một hệ thống sử dụng Docker.



Như trong hình vẽ, một hệ thống Docker được thực thi với 3 bước chính:

Build -> Push -> Pull,Run

Cụ thể hơn về nguyên lí 3 bước này.

1-Build.

Đầu tiên chúng ta sẽ tạo một dockerfile, trong dockerfile này chính là code của chúng ta.

Dockerfile này sẽ được Build tại một máy tính đã cài đặt Docker Engine.

Sau khi build ta sẽ thu được Container, trong Container này chứa bộ thư viện và ứng dụng của chúng ta.

2- Push.

Sau khi có được Container, chúng ta thực hiện push Container này lên đám mây và lưu trữ ở đó.

Việc push này có thể thực hiện qua môi trường mạng Internet.

3- Pull, Run

Giả sử một máy tính muốn sử dụng Container chúng ta đã push lên đám mây (máy đã cài Docker Engine) thì bắt buộc máy phải thực hiện việc Pull container này về máy. Sau đó thực hiện Run Container này.

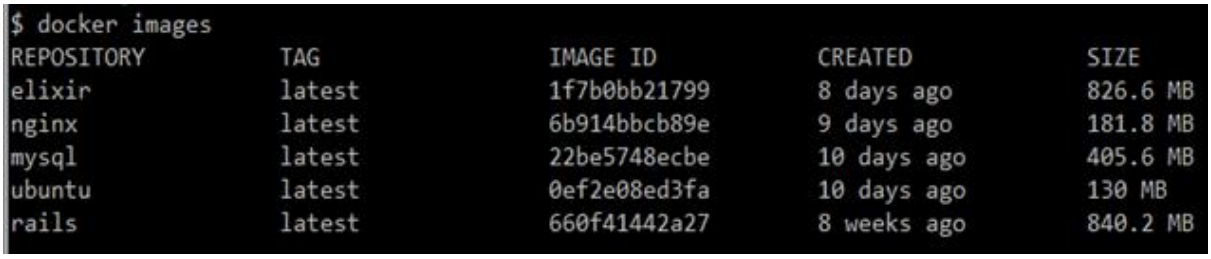
Đó chính là quy trình 3 bước rất đơn giản và rõ ràng miêu tả hoạt động của một hệ thống sử dụng Docker.

2.4. Các lệnh cơ bản thường dùng

Các lệnh cơ bản thường dùng bao gồm:

1-Hiển thị danh sách các images :

Docker images



```
$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
elixir              latest              1f7b0bb21799        8 days ago          826.6 MB
nginx               latest              6b914bbcb89e        9 days ago          181.8 MB
mysql               latest              22be5748ecbe        10 days ago         405.6 MB
ubuntu              latest              0ef2e08ed3fa        10 days ago         130 MB
rails               latest              660f41442a27        8 weeks ago         840.2 MB
```

2-Tải image về local

docker pull <name_image:tag>

(phần :tag là options, nếu để trống thì mặc định download bản latest)

Ví dụ: *docker pull ubuntu* => download ubuntu latest

docker pull ubuntu:14.04 => download ubuntu version 14.04

Truy cập trang https://hub.docker.com/ , nơi lưu trữ các images tập trung để tìm images cần dùng.

3- Chạy một image

docker run –name <tên_container> -v <thư mục trên máy tính>:<thư mục trong container> -p<port_máy tính>:<port_container> <image name> bash

Ví dụ: *docker run –name eva_nginx -p 80:80 -d nginx*

- docker run : lệnh chạy của docker;
- –name: đặt tên cho container ở đây là eva_nginx . Name này là duy nhất, không thể tạo trùng, nếu không đặt thì docker tự genate;
- -p mở port container ra ngoài IP public;
- -d bật chế độ chạy background;
- nginx: tên images.

4- Liệt kê các container

docker ps -a (liệt kê tất cả các container)

docker ps (chỉ liệt kê các container đang chạy background)

5-Dừng container đang chạy

docker stop <container_id hoặc name_container>

docker stop $(docker ps –a –q) **(**Dừng tất cả các docker đang dùng)

6- Khởi động lại container đã dừng

docker start <container_id hoặc name_container>

7- Xóa container không còn sử dụng

docker rm <container_id hoặc name_container>

docker rm $(docker ps -a -q) **(**Xóa tất cả các docker)

8-Truy cập vào 1 container đang chạy

docker exec -it <container_id hoặc name_container> bash

9- Export bản container

docker export <container_id hoặc name_container> | gzip > file_export.tar.gz

10- Import container => image

*zcat file_export.tar.gz | docker <new_name_image>*Sau khi chạy xong. Chạy lệnh *docker images* để kiểm tra lại trong danh sách list images.

Ngoài ra còn nhiều lệnh khác. Sử dụng lệnh *docker -h* sẽ liệt kê chi tiết.

2.5. Ảo hóa ứng dụng với phần mềm tự do nguồn mở Docker

Phần này đưa ra các bước để ảo hóa 1 ứng dụng bằng cách build 1 image cho riêng mình, image chạy rails app. Với các bước tương tự có thể build các image khác nhau tùy thuộc vào mục đích sử dụng.

Bước 1 - Lưu thay đổi trên image sẵn có

Khi chạy 1 container lên, cần cài thêm 1 vài phần mềm, tiện ích vào đó, hay đơn giản chỉ là muốn lưu công việc lại để bữa sau làm tiếp, có thể dùng lệnh commit:

```
$ docker commit {container_id} {author}/{image_name}:{tag}
```

Vd chạy 1 ubuntu, cài ruby và vim vào, sau đó lưu lại:

```
$ docker commit c3f279d17e0a vtchien/ubuntu_ruby_vim:1.0
```

Thực tế cách này chỉ là biện pháp nóng, còn để tạo 1 image thực thụ, docker cung cấp Dockerfile.

Bước 2 - Tạo 1 rails app

Dockerfile Là file định nghĩa tất cả những gì sẽ có trong image.
Trước khi tiếp cận với Dockerfile, cần 1 vài thao tác chuẩn bị.

rails app này sau sẽ chạy trong image.

$ rails new demo

$ cd demo

$ bundle install

$ rails server

Đến đây là khởi động được rails server trên máy local, có thể truy cập vào localhost:3000 để thấy kết quả.

Bước 3 - Tạo Dockerfile

Vẫn ở thư mục gốc của demo, tạo file Dockerfile với nội dung:

```
FROM ubuntu:14.04

FROM ruby:2.3

RUN apt-get update && apt-get install -y \

  build-essential \

  nodejs

RUN mkdir -p /app

WORKDIR /app

COPY Gemfile Gemfile.lock ./

RUN gem install bundler && bundle install --jobs 20 --retry 5

COPY . ./

EXPOSE 3000

CMD ["bundle", "exec", "rails", "server", "-b", "0.0.0.0"]
```

Chú ý mỗi dòng thường bắt đầu bằng 1 chữ IN HOA (FROM, RUN…) đó là các từ khoá của Dockerfile, docker sẽ dựa vào các từ khoá này để build image, giải thích từng dòng:

- Dòng 1: FROM ubuntu:14.04 chỉ ra rằng image sẽ dùng ubuntu:14.04 image làm nền.
- Dòng 2: tương tự dòng 1, tức là image sẽ được cài sẵn ruby:2.3
- Dòng 3,4,5: lệnh RUN tức là sẽ chạy mấy lệnh đó sau khi container được khởi động. Cụ thể ở đây sẽ thiết lập môi trường để chạy được rails app
- Dòng 6: tương tự như trên.
- Dòng 7: WORKDIR thư mục làm việc, nghĩa là sau này, những lệnh như RUN, COPY hay ENTRYPOINT sẽ làm việc tại đây.
- Dòng 8: chép 2 file cần thiết của rails app vào WORKDIR mà đã chỉ định ở trên.
- Dòng 9: tiếp tục cài đặt môi trường cho rails .

- Dòng 10: copy mã nguồn của rails app, trong bài viết này là thư mục demo vào thư mục /app trong container.
- Dòng 11: EXPOSE 3000: mặc định rails sẽ chạy ở cổng 3000, cần phải mở cổng 3000 của container ra có thể truy cập từ bên ngoài (host – tức là máy local).
- Dòng 12: Chạy rails server .
  Có được Dockerfile rồi thì có thể build image được.

Bước 4 - Build image

Cú pháp:

```
$ docker build [OPTION] [path/to/Dockerfile]
```

Lệnh dưới sẽ build một image từ file Dockerfile ở thư mục hiện hành, image có tên và tag là demo:1.0 (-t là tag).

```
$ docker build -t demo:1.0
```

Lệnh trên sẽ show ra rất nhiều thông tin về tiến trình build image.

Sau khi có image rồi thì chạy container.

Bước 5 - Chạy container

```
$ docker run -itP demo:1.0
```

Tham số -P là để mở tất cả những cổng được định nghĩa trong Dockerfile.

Bước 6 - Truy cập vào rails app trong container

Liệt kê các container đang chạy:

$ docker ps

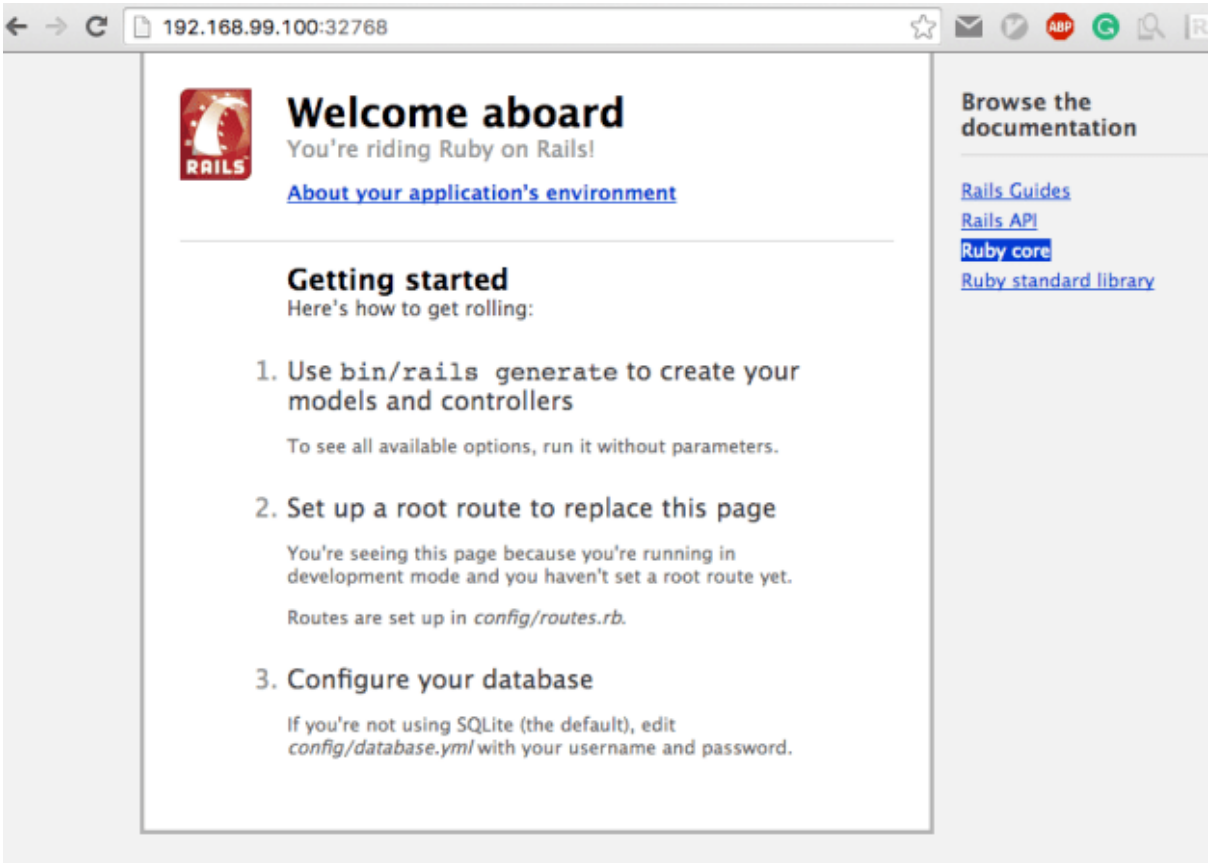
CONTAINER
ID IMAGE COMMAND CREATED STATUS PORTS
NAMES

ede938634b3a demo:1.0 "bundle exec rails se" 9 minutes ago Up 9 minutes 0.0.0.0:32768->3000/tcp

Như thấy ở trên, cột PORTS, container đang mở cổng 3000 và trỏ (NAT) đến cổng 32768, tức là có thể truy cập vào thông qua cổng này.

Với ubuntu, chỉ cần truy cập: http://localhost:32768 là vào được rails server.

Ở máy cá nhân, địa chỉ máy VM là: 192.168.99.100, có thể truy cập vào 192.168.99.100:32768 để kiểm tra kết quả.



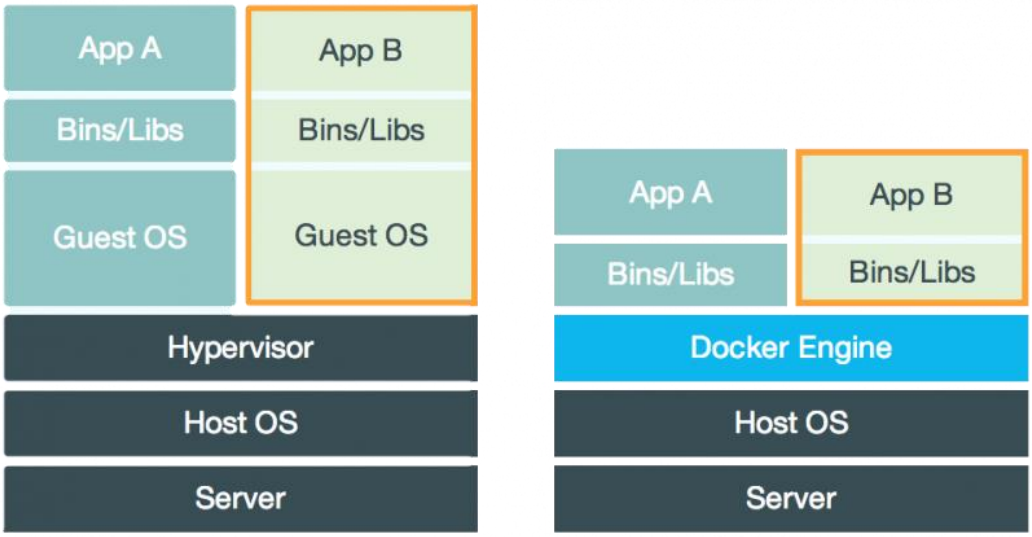
2.6. Ưu thế của Docker so với các phần mềm ảo hóa ứng dụng khác

Có 3 loại phần mềm ảo hóa ứng dụng phổ biến: Virtual Machine, Vagrant, Docker

1-Virtual Machine: Virtual machine hay còn gọi là phần mềm tạo máy ảo. Phần mềm này cho phép tạo lập và chạy một hệ điều hành (hay máy ảo) trên máy chủ (host machine). Ví dụ máy tính bạn chạy Windows nhưng cần dev app chạy trên Linux thì bạn có thể sử dụng VirtualBox để tạo một máy ảo Linux trên Windows. VirtualBox là phần mềm để tạo máy ảo phổ biến hiện nay trên PC. Ngoài ra còn có Vmware, MicroSoft Vitual PC. Với dòng phần mềm này tài nguyên máy tính phải đáp ứng đủ cho cả OS thật và OS ảo.

2-Vagrant: Là một công nghệ dựa trên nền ảng của Virtual Machine tuy nhiên cung cấp thêm tính năng để cấu hình và thiết lập môi trường cho các máy ảo (hay các box). Ví dụ team có 15 người đang sử dụng máy Windows nhưng cần phát triển ứng dụng chạy trên server Linux. Server này cần cài đặt Apache web server, PHP, MySQL... Sử dụng Vagrant có thể tạo một box có sẵn các phần mềm trên (và cấu hình cần thiết nếu có) để khi 15 team member muốn tạo môi trường giả lập họ chỉ cần sử dụng box đã cấu hình mà không cần tạo box và cài lại từng phần mềm một (tưởng tượng team có 100 nhân viên thì việc này sẽ giảm thiểu được nhiều thời gian đáng kể tuy nhiên phải cần 1 máy chủ rất khỏe).

3-Docker: Đây là công nghệ đang được đánh giá là tương lai của công nghệ ảo hoá (future of virtualization). Khác biệt lớn của Docker và Vagrant (hay Virtual Machine nói chung) đó là tiết kiệm đáng kể resource sử dụng. Với docker bạn có thể chạy 20 container (tương tự như một hệ điều hành nhỏ) trên cùng một máy host (host machine) điều mà nếu sử dụng Vagrant sẽ cần một máy chủ với cấu hình cực khủng. Docker làm được điều này là bởi vì khác với Virtual Machine ở chỗ thay vì tách biệt giữa hai môi trường guest và host, thì các container của Docker chia sẻ các resource với host machine.



Ngoài ra Docker là một nền tảng mở (open platform), dùng để phát triển và vận hành ứng dụng, nó còn có 1 số ưu điểm sau:

1. Với sự hỗ trợ của docker, việc coding, testing, deploying trở nên đơn giản hơn.

2. Khả năng di động (portable): môi trường develop được dựng lên bằng docker có thể chuyển từ người này sang người khác mà không làm thay đổi cấu hình ở trong.

3. Application-centric: docker được dùng trên nhiều môi trường, đặc biệt tương thích trên môi trường develop, hướng đến việc coding thuận tiện nhất.

4. Versioning: docker được tích hợp VCS-git, để tracking các dòng lệnh thiết lập, hay đánh dấu version.

5. Component re-use: nghĩa là docker có khả năng sử dụng lại resource trước đó, bằng cách đánh dấu những resources giống nhau bằng một mã ID. Các môi trường được dựng lên sau đó sẽ kiểm tra các mã ID trước đó, nếu trùng docker sẽ sử dụng lại.

6. Sharing: với Docker Hub (public registry), các developer có thể tìm và sử dụng các môi trường được dựng sẵn.

Các OS có thể dùng Docker:

1. Linux: Ubuntu 12.04+, Fedora 19/20+, RHEL 6.5+, CentOS 6+, Gentoo, ArchLinux, openSUSE 12.3+, CRUX 3.0+

2. Cloud: Amazon EC2m Google Compute Engine, Microsoft Azure,…

3. Max OS X

4. Windows: Windows 7+

Bên cạnh những ưu điểm kể trên, docker cũng có những nhược điểm:

- ✓ Tốn thời gian học tập;
- ✓ Mới, cập nhật thay đổi thường xuyên;
- ✓ Công cần thiết để biết viết code tại một môi trường gần như không kém hơn so với phát triển trên local;
- ✓ Xét về tính an toàn:
  - o Do dùng chung OS nên nếu có lỗ hổng nào đấy ở kernel của host OS thì nó sẽ ảnh hưởng tới toàn bộ container có trong host OS đấy;
  - o Ngoài ra với host OS là Linux, nếu trong trường hợp ai đấy hoặc một ứng dụng nào đấy có trong container chiếm được quyền superuser. Về lý thuyết thì tầng OS sẽ bị crack và ảnh hưởng trực tiếp tới máy host bị hack cũng như các container khác trong máy đó (hacker sử dụng quyền chiếm được để lấy dữ liệu từ máy host cũng như từ các container khác trong cùng máy host bị hack chẳng hạn).

Kết luận chương

Docker được phát hành dạng mã nguồn mở trong tháng 3 năm 2013. Tới 2015, dự án Docker trở thành top 20 dự án có số sao cao nhất rên GitHub. Năm 2017, Có thể nói chưa bao giờ docker có nhiều thay đổi như năm 2017. Hiện nay phần mềm tạo ảo hóa của Docker được đánh giá là tương lai của công nghệ ảo hoá và nó đã hỗ trợ nhiều hệ điều hành khác nhau như GNU/Linux, Cloud, Mac OS X và Microsoft Windows.

Docker đưa ra một giải pháp mới cho vấn đề ảo hóa, thay vì tạo ra các máy ảo con chạy độc lập kiểu hypervisors (tạo phần cứng ảo và cài đặt hệ điều hành lên đó), các ứng dụng sẽ được đóng gói lại thành các Container riêng lẻ. Các Container này chạy chung trên nhân hệ điều hành qua LXC, chia sẻ chung tài nguyên của máy mẹ, do đó, hoạt động nhẹ và nhanh hơn các máy ảo dạng hypervisors. Docker có 2 thành phần chính là Docker Enginne và Docker Hub, Có các đối tượng được quy định: Docker images, Docker registries, Docker container, Dockerfile, Orchestration.

Một hệ thống Docker được thực thi với 3 bước chính : Build -> Push -> Pull,Run.

Quy trình ảo hóa ứng dụng sử dụng Docker thực hiện qua các bước: Tạo Dockerfile; Build image; Chạy container.

So với các công nghệ ảo hóa phổ biến khác hiện nay như Virtual Machine và Vagrant thì công nghệ Docker có nhiều ưu điểm hơn như nhanh hơn, nhẹ hơn, chia sẻ dễ dàng, tốn ít tài nguyên hơn,…

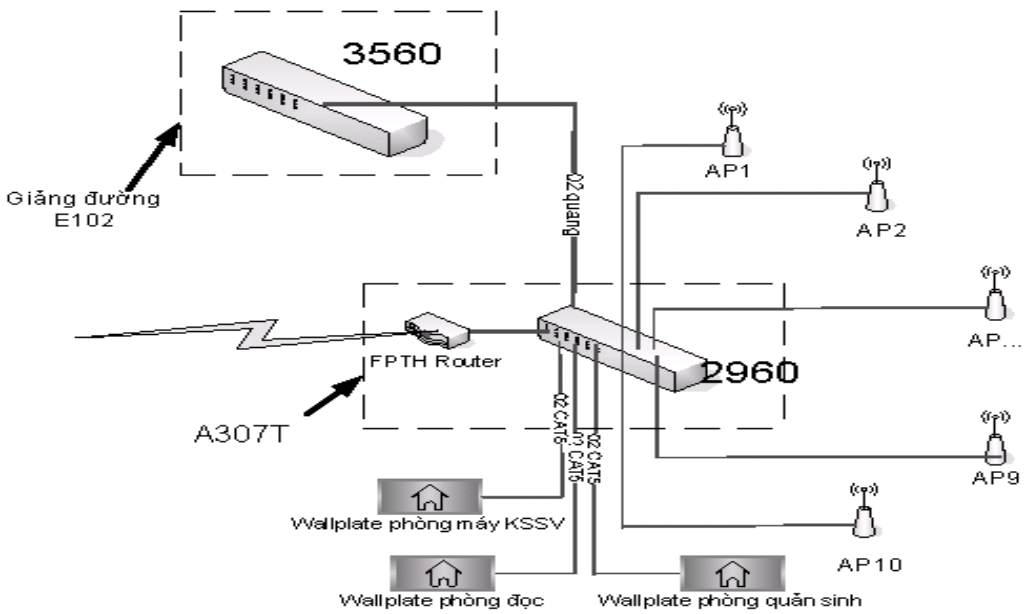
CHƯƠNG 3
ỨNG DỤNG CÔNG NGHỆ DOCKER ĐỂ ẢO HÓA ỨNG DỤNG TẠI ĐHDL HẢI PHÒNG

3.1. Hiện trạng hệ thống và nhu cầu ảo hóa tại ĐH Dân lập HP

3.1.1. Hiện trạng hệ thống

3.1.1.1. Hiện trạng hệ thống mạng

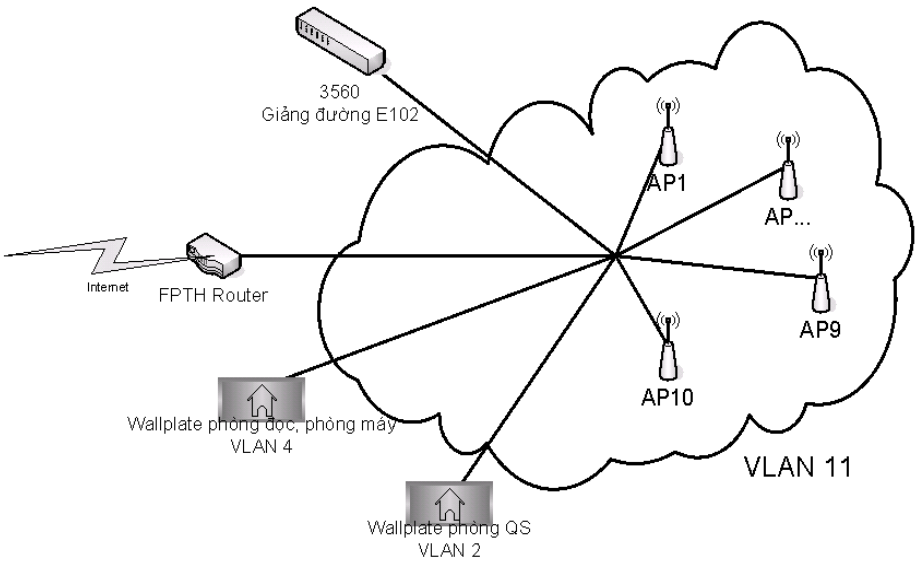
1-Sơ đồ kết nối vật lý



Hình 3. 1: Sơ đồ kết nối vật lý mạng HPU

Tất cả các nút mạng đều được nối về Switch 2960 - phòng A307T và Switch 3560 – Phòng E102 khu giảng đường, để phục vụ cho công việc của cán bộ giảng viên và nhu cầu truy cập mạng của sinh viên, các nút mạng có thể là các điểm truy cập không dây, có dây, các switch.

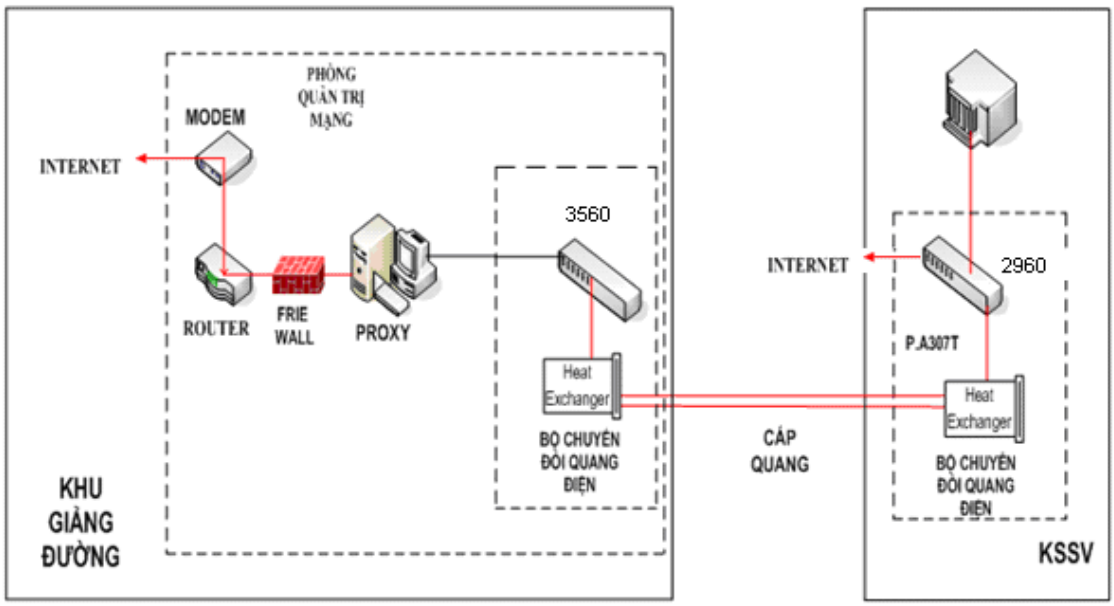
2-Sơ đồ logic



Hình 3. 2: Sơ đồ logic mạng HPU

Toàn bộ hệ thống mạng của nhà trường chia thành các VLAN khác nhau dựa theo chức năng và nhiệm vụ, ví dụ khối thư viện thuộc VLAN4, khối phòng ban thuộc VLAN2, khối mạng không dây thuộc VLAN 11. Switch lõi trung tâm 3560 đảm nhiệm vài trò định tuyến giữa các VLAN với nhau.

3-Kết nối giữa khu giảng đường sang khách sạn sinh viên



Hình 3. 3: Kết nối giữa khu GĐ và KSSV

Kết nối giữa các khu vực khác nhau trong nhà trường sử dụng cáp quang, việc sử dụng 02 sợi cáp quang giúp hạn chế rủi ro khi đứt cáp và đảm bảo tổng dung lượng băng thông giữa các switch trong lõi là 2Gb.

3.1.1.2. Hiện trạng hệ thống máy chủ

Danh sách server:

STT	Modem	Date	Info	Vị trí	Hệ điều hành	App	DNS
2	Compaq		1. Kiểm tra lại	E102		ftp server	
3	Compaq			E102			
4	IBM	17/08/2005		E102	pfsense	Gateway/ Firewall/ Proxy/ VNP Server	gateway.hpu.edu.vn vpn.hpu.edu.vn
5	IBM	17/08/2005		E102	debian 7tr/năm	mikrotik	
7	IBM			E102			
8	IBM			E102			
9	IBM	15/05/2009	IBM x3650M1 Xeon, 16G RAM, 1x500G HDD	E102	Window 2008	Dang ky mon hoc Server 2, QLGD, xét tuyển, tham do, dieu kien,webservice, decuong, phiếu cá nhân giảng dạy	qlgd.hpu.edu.vn thamdo.hpu.edu.vn dk.hpu.edu.vn , dieukien.hpu.edu.vn s3.hpu.edu.vn, xettuyen.hpu.edu.vn
10	IBM	15/05/2009	IBM x3650M1 Xeon, 16G RAM, 1x500G HDD	E102	Window 2008	libol, vp, ho tro, niên chế-tín chỉ(chuyển điểm qlgd)	s2.hpu.edu.vn vp2.hpu.edu.vn vp.hpu.edu.vn libol.hpu.edu.vn hotro.hpu.edu.vn
11	IBM	15/05/2009	IBM x3650M1 Xeon, 16G RAM, 1x500G HDD	E102	Ubuntu	ACC	acc.hpu.edu.vn s1.hpu.edu.vn

12	IBM	30/072010	IBM x3650M2 2 chip, 18G RAM, 2x300G HDD	E102	Ubuntu,	Tester	
13	IBM	30/072010	IBM x3650M2 2 chip, 18G RAM, 2x300G HDD	E102	Window 2008	Nhật ký Edu, EDUMng	
14	IBM	30/072010	IBM x3650M2 2 chip, 18G RAM, 2x300G HDD	E102	Window 2008	Tuyển sinh, xết tuyển, nhân sự, ESP - EDU	
15	FPT	Mar-14	4G RAM/ 2x500G HDD	G3	Window 2008	forum	diendan.hpu.edu.vn
16	FPT	Mar-14	4G RAM/ 2x500G HDD	G3	Window 2008	web cũ	web.hpu.edu.vn
17	NAS	15/05/2009	Ready NAS TM 1100 4TB Dual Gigabit Rackmount	E102		NAS NetGear	
18	SuperMicro	2012	Xeon 3.1 RAM 8G HDD 2x2TB	VDC	Ubuntu	website mới, eng, wikihpu > qlgd1	
19	SuperMicro	2012	Xeon 3.1 RAM 8G HDD 2x2TB	VDC	Window 2008	dspace test > dspace	lib.hpu.edu.vn
20	FPT Elead	2009	2G RAM	E102	clound	Tester	
21	FPT Elead	2009	2G RAM	E102	clound	Tester	

22	Dell	18/12/2015	RAM 8G; HDD 2x1TB	FPT	Window 2008	http://hpu.edu.vn	is.hpu.edu.vn hpu.edu.vn eng.hpu.edu.vn/ m.eng.hpu.edu.vn, hpuwiki.hpu.edu.vn
23	Dell	18/12/2015	RAM 8G; HDD 2x1TB	FPT	Window 2008	http://img.hpu.edu.vn	img.hpu.edu.vn
23		Apr-16	RAM 64GB;	FPT	Ảo hóa	Server 1: Window 2008	
23		Apr-16	RAM 64GB;	FPT	Ảo hóa	Server 2: Window 2008	

Hình 3. 4: Danh sách máy chủ

Dễ dàng nhận thấy số lượng máy chủ khá lớn, có thời gian mua khác nhau, cấu hình cao thấp không đồng đều, một số máy yếu không sử dụng được chuyển sang mục đích nghiên cứu, một số máy cấu hình cao thì số lượng các ứng dụng cài đặt trên đó không đồng đều, có máy cài 1 ứng dụng, có máy cài nhiều ứng dụng, các ứng dụng thuộc nhiều nền tảng khác nhau, gây lên việc sử dụng các máy không hết hiệu năng hoặc quá tải.

3.1.1.3. Hiện trạng sử dụng

3.1.1.1. Phân tích hiện trạng

Về khả năng đáp ứng

Để đáp ứng những yêu cầu không ngừng về việc triển khai, phát triển các dịch vụ và ứng dụng mới, nhà trường phải thường gia tăng thêm số lượng máy chủ cả về số lượng lẫn chất lượng.

Về an toàn thông tin

Như trong mục hiện trạng mạng máy tính nội bộ của Trường ĐH DL Hải Phòng, ta thấy rằng, hiện tại các máy tính kết nối trong mạng nội bộ và thực hiện chia sẻ thông tin theo mô hình Peer-to-peer. Với mô hình kết nối peer-to-peer việc quản lý hoạt động của các user là không thể thực hiện được. Nó sẽ gây ra những khó khăn trong quá trình quản lý cũng như khai thác hệ thống:

- ✓ Không có phương thức quản lý tập trung, các máy tính thực hiện cài đặt và khai thác mạng một cách đơn lẻ;
- ✓ Việc triển khai các chính sách phân tán do không có máy chủ và phương thức thực hiện tập trung, sẽ gây khó khăn trong việc thực hiện;
- ✓ Không có cơ chế phân quyền hợp lý và quản lý người dùng có thể gây ra những nguy cơ về mất mát dữ liệu và những vi phạm trong nội bộ mà việc phát hiện là hết sức khó khăn.

Khả năng quản lý, mở rộng

Quy mô máy chủ gia tăng thì những vấn đề khó khăn lại càng thể hiện rõ nét hơn.

Các hệ quả phát sinh có thể kể đến như:

- ✓ Chi phí tăng: chi phí mua máy chủ / thiết bị, chi phí thuê không gian chỗ đặt, chi phí vận hành, chi phí quản lý…
- ✓ Hiệu quả đầu tư thấp: mỗi máy chủ hầu như chỉ sử dụng một phần tài nguyên chuyên dụng (RAM hoặc CPU hoặc ổ cứng) trong toàn bộ máy chủ vật lý tương ứng trong khi các tài nguyên khác có khi chưa dùng hết 20%
- ✓ Khó khăn trong quản lý và vận hành.

3.1.2. Yêu cầu ảo hóa đối với hệ thống

Yêu cầu về thiết bị

- ✓ Tận dụng được thiết bị trong hệ thống cũ.

Yêu cầu về hiệu năng

- ✓ Tận dụng được hiệu năng dư thừa ở các máy chủ;
- ✓ Giảm tải cho 1 số máy chủ đang chạy với công xuất lớn.

Yêu cầu về an toàn thông tin

- ✓ Hệ thống phải được đánh giá có khả năng chống chịu tốt với các nguy cơ tấn công mạng.

Yêu cầu về mở rộng

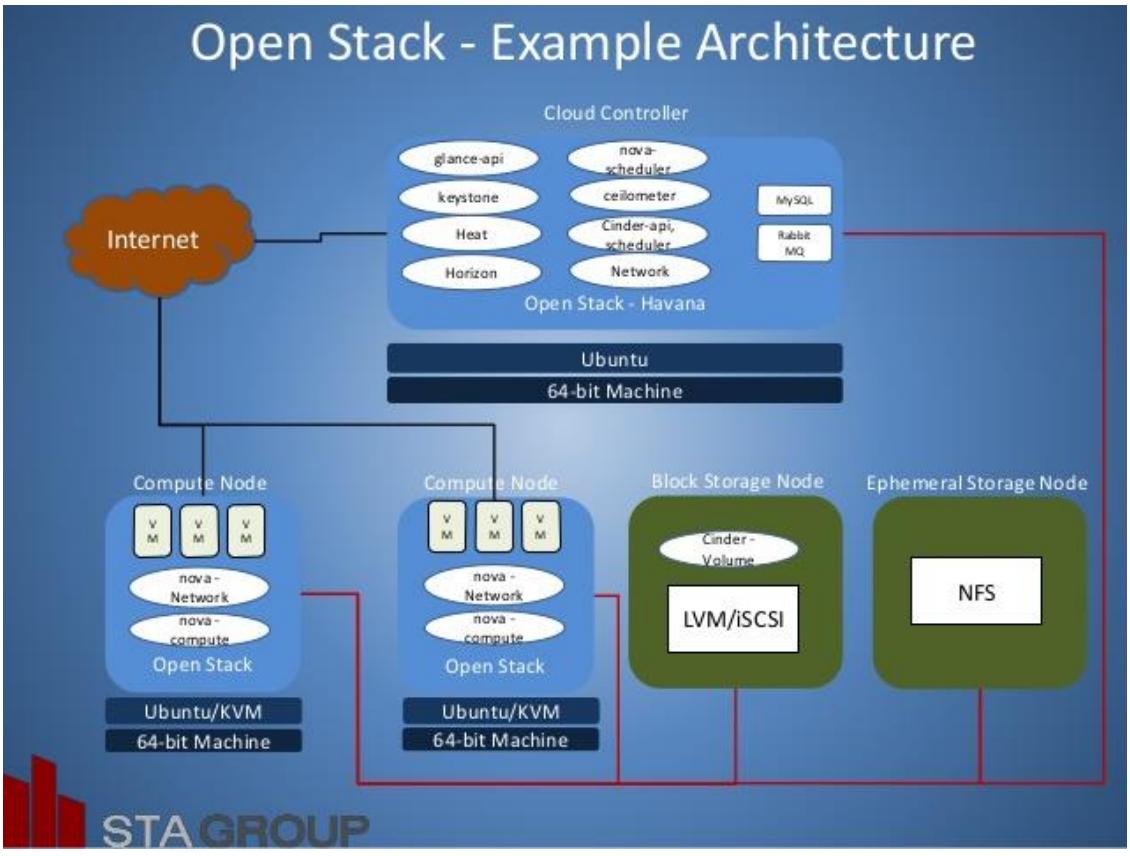
- ✓ Đáp ứng được yêu cầu mở rộng trong tương lai: hỗ trợ nhiều dịch vụ khác nhau, hỗ trợ số lượng người dùng tăng, nâng cấp hệ thống một cách dễ dàng.

3.2. Lựa chọn công nghệ ảo hóa

3.2.1. Công nghệ đề xuất

Giai đoạn 1:

Tận dụng các máy chủ sẵn có để xây dựng thành 1 (vài) máy ảo mạnh, đây là ảo hóa ở kiểu IaaS, có thể sử dụng Ubuntu Openstack.



Hình 3. 5: Kiến trúc của Ubuntu Opentack

Trong mô hình này có 3 thành phần:

- Sẽ sử dụng 1 server làm chức năng Cloud Controller;

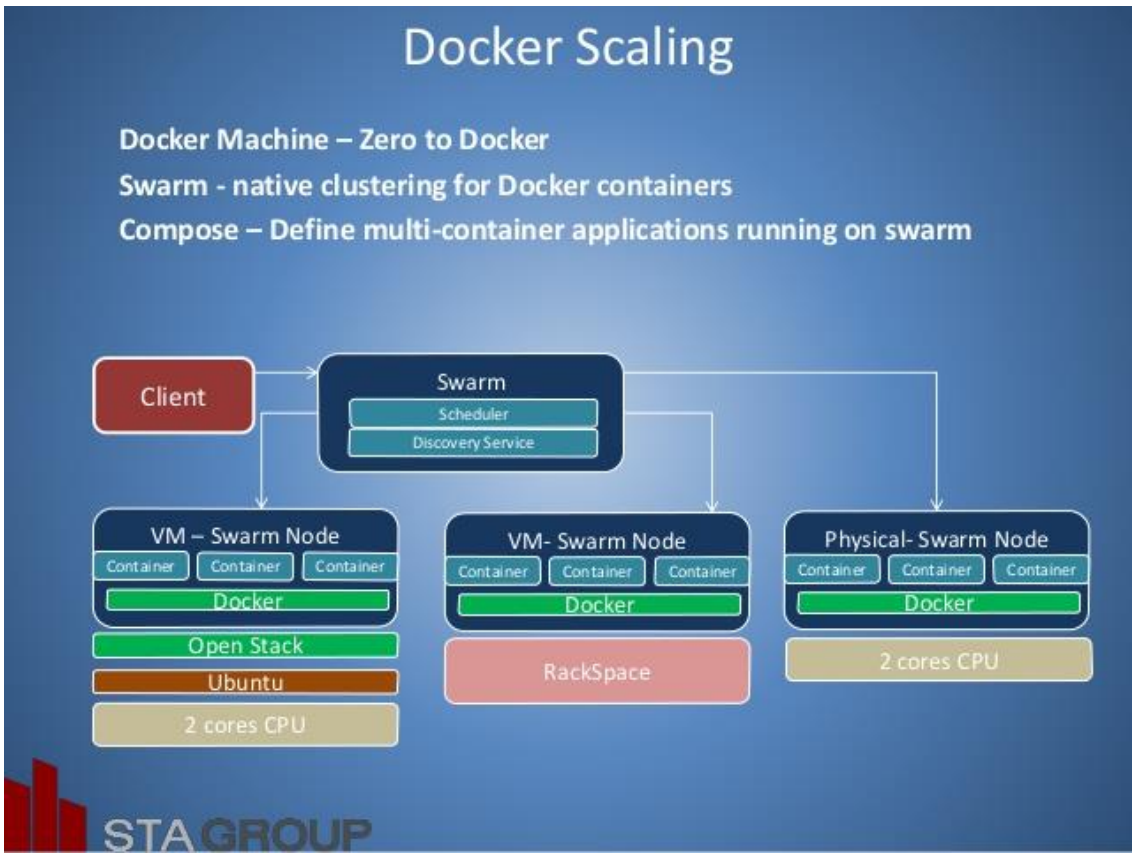
- Các server khác sẽ làm chức năng tính toán, xử lý Computer Node;

- Thành phần lưu trữ các bản ảnh có thể dùng 1 server dung lượng HDD lớn hoặc các thiết bị lưu trữ ngoài.

Lưu ý các server phải đáp ứng được các yêu cầu cài đặt của Ubuntu Cloud https://help.ubuntu.com/community/Installation/SystemRequirements#Ubuntu_Server_.28CLI.29_Installation

Giai đoạn 2:

Trên các máy ảo VMs mạnh từ kết quả của giai đoạn 1, hoặc trên 1 server vật lý cấu hình mạnh ta có thể sử dụng ảo hóa Docker.

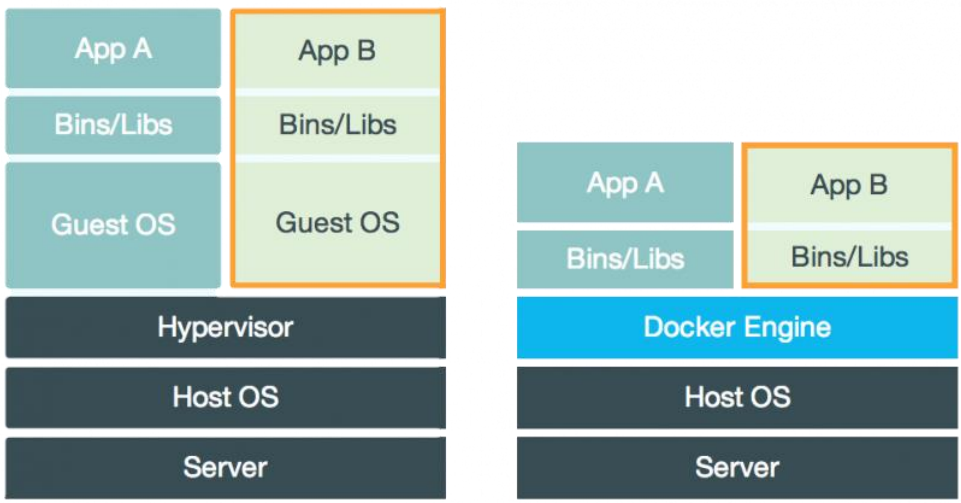


Hình 3. 6: Ảo hóa Docker

Trong phạm vi của đề tài này em xin chỉ tập trung vào phương án ảo hóa ứng dụng sử dụng Docker và thử nghiệm sử dụng Docker trên 1 máy chủ vật lý để ảo hóa thư viện số Dspace của trường Đại học Dân lập Hải Phòng.

3.2.2. Tính khả thi của giải pháp

- ✓ Không trang bị thêm trang thiết bị mới, tận dụng được thiết bị trong hệ thống cũ;
- ✓ Các ứng dụng dùng chung tài nguyên máy VMs hoặc máy chủ vật lý, tận dụng được hiệu năng dư thừa ở các máy chủ;
- ✓ Sử dụng các phần mềm mã nguồn mở được đánh giá có khả năng chống chịu tốt với các nguy cơ tấn công mạng;
- ✓ Đáp ứng được yêu cầu mở rộng trong tương lai: hỗ trợ nhiều dịch vụ khác nhau, hỗ trợ số lượng người dùng tăng, đưa thêm các ứng dụng mới một cách dễ dàng;
- ✓ Sử dụng Docker để ảo hóa được thực tế đánh giá tiêu tốn tài nguyên máy chủ ít hơn VMs.



Hình 3. 7: So sánh VMS và Docker

3.3. Thiết kế mô hình ứng dụng công nghệ Docker cho ĐHDL Hải Phòng

3.3.1. Mục tiêu

Ảo hóa toàn bộ hệ thống ứng dụng để loại trừ:

- ✓ Thời gian trì trệ đầu tư thiết bị máy chủ mới khi triển khai một ứng dụng mới;
- ✓ Thời gian chết (downtime) khi bảo trì hay nâng cấp phần cứng cho hệ thống ứng dụng;
- ✓ Khai thác triệt để hiệu năng cũng như công năng của công nghệ và sức mạnh phần cứng máy chủ hiện nay;
- ✓ Quản lý tập trung tại một điểm duy nhất và giảm thiểu các thao tác quản trị;
- ✓ Dễ dàng và linh động triển khai các yêu cầu mới ngay lập tức và sao lưu dự phòng toàn bộ hệ thống.

3.3.2. Các yêu cầu cần thực hiện

Cần thực hiện các yêu cầu sau:

- ✓ Đưa ra phương án sử dụng Docker ảo hóa một số ứng dụng đang sử dụng tại trường Đại học Dân lập Hải Phòng;
- ✓ Thực nghiệm sử dụng Docker ảo hóa thư viện số Dspace.

3.3.3. Sơ đồ thiết kế

Danh sách các ứng dụng tại Đại học Dân lập Hải Phòng

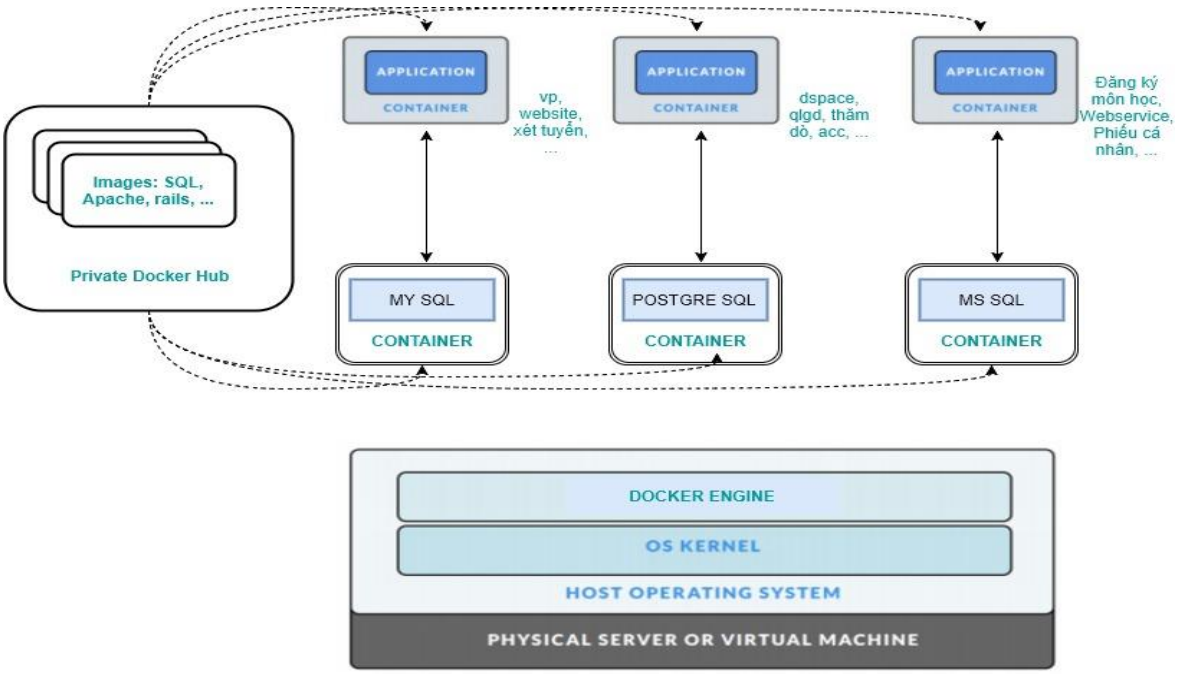
STT	Tên ứng dụng	Nền tảng web	Database
1	Dang ky mon hoc Server 2	IIS	MS SQL server
2	webservice	IIS	MS SQL server
3	phiếu cá nhân giảng dạy	IIS	MS SQL server
4	libol	IIS	MS SQL server
5	web cũ	IIS	MS SQL server
6	Nhật ký Edu		MS SQL server
7	nhân sự		MS SQL server
8	ESP - EDU		MS SQL server
9	EDUMng		MS SQL server
10	xét tuyển	Apache	My SQL
11	Vp	Apache	My SQL

12	ho tro	Apache	My SQL
13	niên chế-tín chỉ(chuyển điểm qlgd)	Apache	My SQL
14	Tuyển sinh	Apache	My SQL
15	forum	Apache	My SQL
16	website mới	Apache	My SQL
17	Eng	Apache	My SQL
18	dspace		PostGreSQL
19	QLGD	Ruby on Rails	PostGreSQL
20	tham do	Ruby on Rails	PostGreSQL
21	dieu kien	Ruby on Rails	PostGreSQL
22	decuong	Ruby on Rails	PostGreSQL
23	ACC	Ruby on Rails	PostGreSQL

Hình 3. 8: Các ứng dụng tại HPU

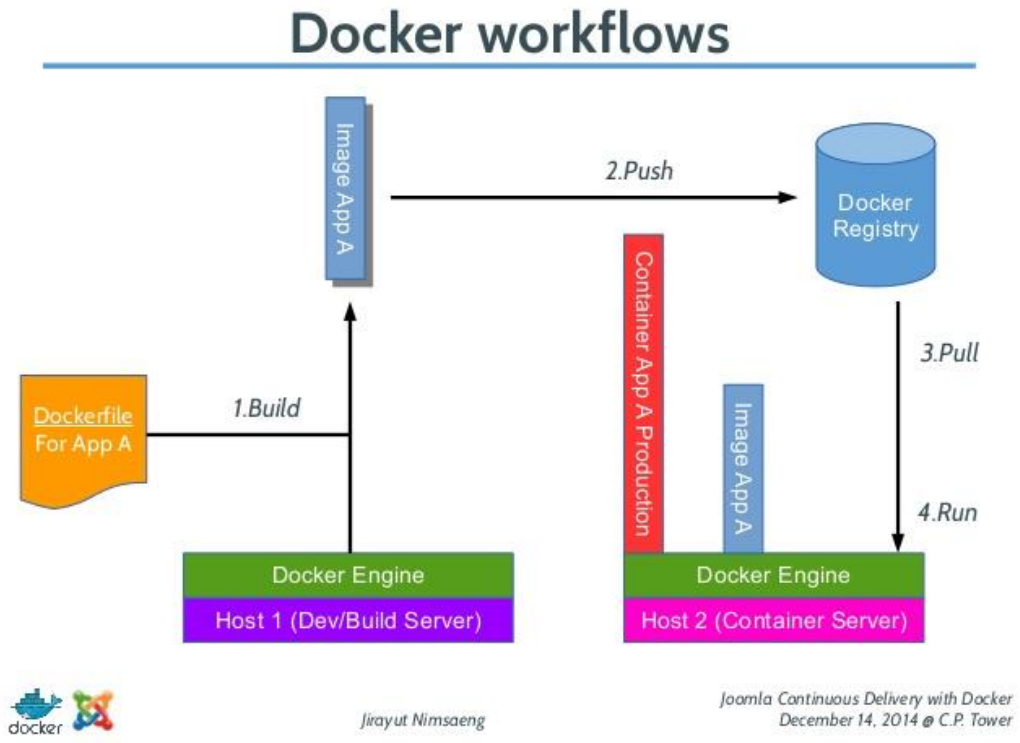
Có thể phân loại như sau:

- ✓ Nhóm cùng loại CSDL: như vậy trong Docker ta có thể tạo ra các conteinner Database dựa trên cùng các Image: MS SQL server, My SQL, PostGreSQL, CSDL của các ứng dụng sẽ đặt tương ứng tại các conteinner Database này;
- ✓ Tương tự như trên cũng tạo các conteinner cho các ứng dụng, nhóm cùng loại nền tảng web có thể dùng chung các images web app , các conteinner chứa ứng dụng sẽ kết nối với các conteinner Database tương ứng của từng ứng dụng.



Hình 3. 9: Sơ đồ thiết kế ảo hóa ứng dụng tại HPU

3.4. Quy trình thực hiện ảo hóa theo công nghệ Docker



Hình 3. 10: Quy trình ảo hóa trong Docker

Quy trình trên có thể diễn giải như sau:

- ✓ Tạo Dockerfile cho ứng dụng trên máy (dockerfile là code của ứng dụng);
- ✓ Việc tạo các Image được thực hiện ở Docker Hub;
- ✓ Bản ảnh cho ứng dụng được tạo, và sẵn sằng cho Pull;
- ✓ Chạy Container từ Image được tạo.

3.5. Sử dụng Docker ảo hóa thư viện số Dspace

3.5.1. Cài đặt Docker

Để cài đặt Docker trên Ubuntu 14.04 sử dụng câu lệnh sau:

Thêm Docker Repository

```
sudo apt-key adv --keyserver hkp://p80.pool.sks-keyservers.net:80 --recv-keys
58118E89F3A912897C070ADBF76221572C52609D
sudo apt-add-repository 'deb https://apt.dockerproject.org/repo ubuntu-xenial main'
```

Cài Docker Engine

```
sudo apt-get update && sudo apt-get install -y docker-engine
```

Sử dụng docker không cần lệnh sudo

Bước 1: Add user hiện tại vào group docker.

```
sudo gpasswd -a ${USER} docker
sudo service docker restart
```

Bước 2: Logout hoặc restart lại để user nhận group mới.

Thay đổi thư mục data của Docker

Bước 1: Tạo thư mục chứa data docker: /data/docker
Bước 2: Thêm dòng sau vào cuối nội dung file /etc/fstab

```
/data/docker /var/lib/docker none bind 0 0
```

Bước 3: Restart lại OS.

3.5.2. Tạo các Container

B1: Containner Dspace

Có một Dspace Docker Containner có sẵn trên Docker Hub. Trước tiên lấy Containner Dspace

```
docker pull unixelias/docker-dspace
```

B2: Containner PostgreSQL

Chúng ta có thể sử dụng một cơ sở dữ liệu bên ngoài hoặc tạo ra một containner PostgreSQL liên kết với conteinner DSpace. Có một PostkreSQL Docker Image có sẵn trên Docker Hub. Trước tiên tạo một Containner PostgreSQL.

```
docker run -d --name dspace_db -p 5432:5432 unixelias/postgres-dspace
```

tiếp theo chạy lệnh container PostgreSQL liên kết với container Dspace.

```
docker run -d --link dspace_db:postgres -p 8080:8080 docker-dspace
```

Sau khi tạo xong ta kiểm tra lại bằng lệnh:

docker ps để xem các containner đang chạy.

docker images: để xem các bản ảnh trên máy.

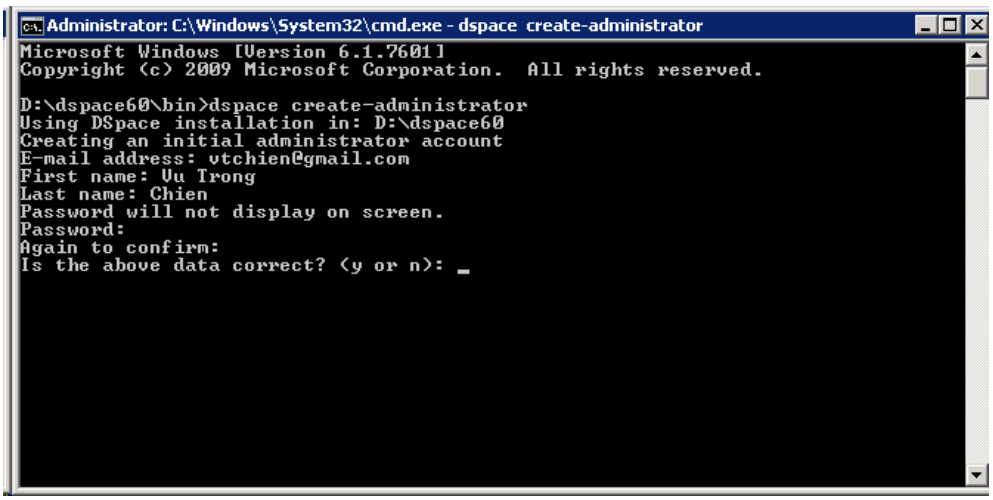
```
REPOSITORY              TAG             IMAGE ID        CREATED         SIZE
postgres                latest          f2a2d73f5ce3    3 days ago      287MB
unixelias/docker-dspace latest          a5f8f1b618b8    5 days ago      1.18GB
1science/dspace         latest          1f04de60ca05    2 months ago    1.42GB
```

Theo mặc định, CSDL được tạo với tên dspace với người dùng là dspace và mật khẩu dspace , nhưng có thể ghi đè cài đặt mặc định này.

```
docker run -d --link dspace_db:postgres \
    -e POSTGRES_SCHEMA=my_dspace \
    -e POSTGRES_USER=my_user \
    -e POSTGRES_PASSWORD=my_password \
    -p 8080:8080 unixelias/docker-dspace
```

B3: Tạo tài khoản Admin cho dspace

Dùng lệnh dspace create-administrator để tạo tài khoản admin cho Dspace



Hình 3. 11: Tạo tài khoản admin cho dspace

3.5.3. Chuyển dữ liệu từ Dspace cũ sang Docker dspace

Việc chuyển dữ liệu từ Dspace cũ sang Docker dspace về mặt kỹ thuật khá đơn giản. Trên Dspace cũ ta chỉ cần Export bộ sưu tập số ra file metadata, bao gồm cả các file tài nguyên. Trên Docker dspace chỉ việc Import vào là được. Tuy nhiên nếu số lượng nguồn tài nguyên lớn công việc này tốn khá nhiều thời gian và công sức.

3.5.4. Kết quả và đánh giá hiệu qủa

1. Kết quả

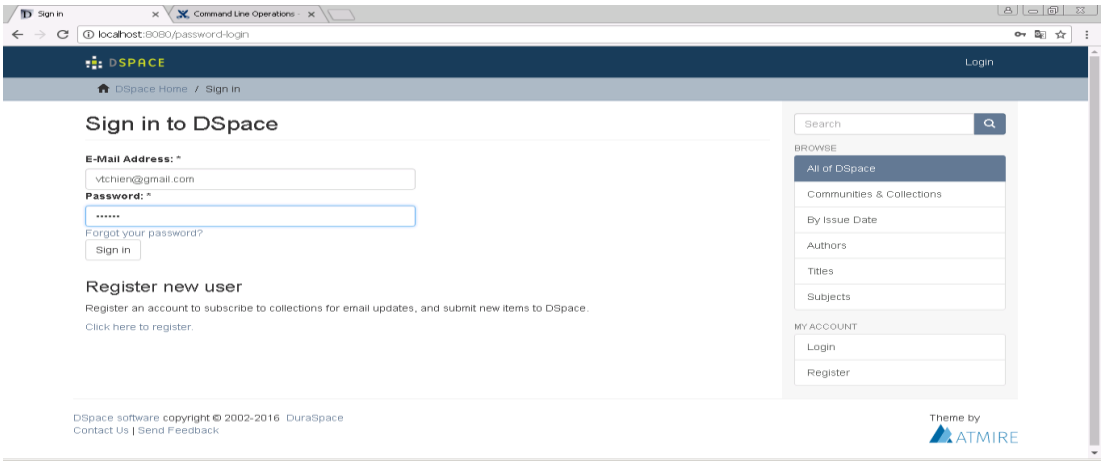
Hệ thống Dspace trên nền tảng Docker được tác giả dựng tại máy chủ có địa chỉ:

http://203.162.241.123:8080/xmlui/

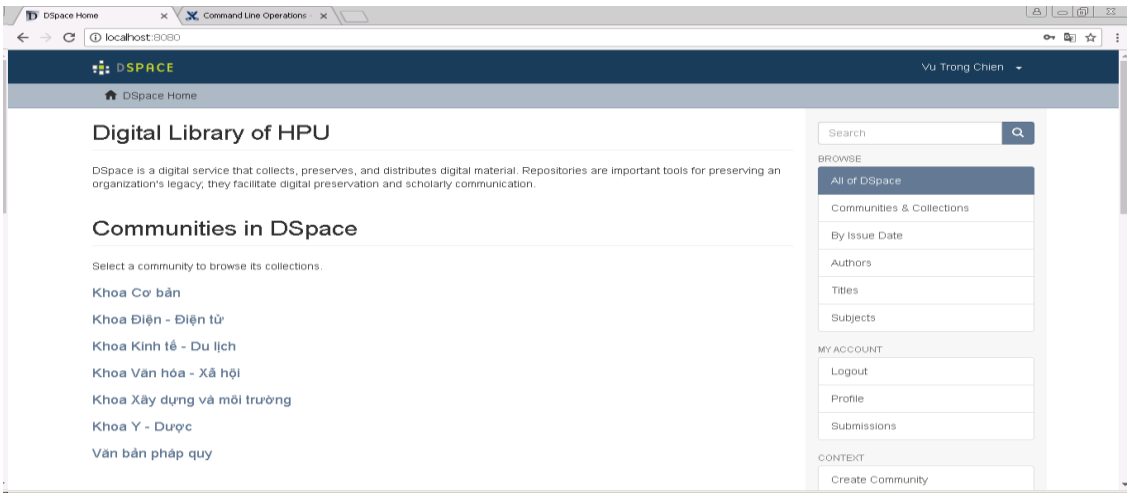
Tài khoản đăng nhập: *vtchien@gmail.com*

Mật khẩu: *123654*

Một số hình ảnh của Docker Dspace sau khi chạy:



Hình 3. 12: Giao diện đăng nhập của Dspace



Hình 3. 13: Giao diện trang chủ của Dspace

2. Đánh giá hiệu quả

So sánh giữa Dspace cài đặt bình thường trên 1 máy chủ và dùng Docker Dspace, thì Docker Dspace có những ưu điểm sau:

- ✓ Docker Dspace cài đặt dễ dàng hơn rất nhiều so với Dspace cài đặt bình thường trên 1 máy chủ;
- ✓ Việc nghiên cứu các ứng dụng đơn giản hơn, trên docker hub có rất nhiều các bản ảnh của các ứng dụng, chỉ cần lấy về và chạy;
- ✓ Về tài nguyên, Docker Dspace sẽ ngốn ít tài nguyên hơn, trên máy cài Docker Dspace có thể cài thêm các containner của các ứng dụng khác để tận dụng nguồn tài nguyên phần cứng;
- ✓ Docker Dspace Hoạt động nhanh và nhẹ hơn;
- ✓ Chia sẻ dễ dàng: như có thể nhân bản image đưa sang máy khác cho các mục đích chuyên biệt như nghiên cứu và phát triển, hay có thể tạo một bản Images và triển khai cho rất nhiều khách hàng.

Kết luận chương

Dựa vào kết quả khảo sát và phân tích về hệ thống mạng, máy chủ, ứng dụng, hiện trạng sử dụng, công nghệ ảo hóa thử nghiệm cho Trường Đại học Dân lập Hải Phòng được thực hiện theo 2 giải pháp:

1. Tận dụng các máy chủ sẵn có để xây dựng thành 1 (vài) máy ảo mạnh, có thể sử dụng Ubuntu Openstack.

2. Trên các máy ảo VMs mạnh từ kết quả bước trên, hoặc trên 1 server vật lý cấu hình mạnh sử dụng Docker để ảo hóa các ứng dụng của nhà trường.

Riêng với giải pháp thứ 2, quá trình thiết kế mô hình ứng dụng công nghệ Docker cho ĐH DL HP theo các bước: Xác định yêu cầu, mục tiêu; Đưa ra sơ đồ thiết kế; Quy trình để thực hiện ảo hóa ứng dụng sử dụng công nghệ ảo hóa Docker.

Kết quả thử nghiệm Sử dụng Docker ảo hóa thư viện số Dspace của nhà trường đã cho kết quả tốt và hiệu quả.

KẾT LUẬN

Luận văn tập chung nghiên cứu về công nghệ ảo hóa Docker và ứng dụng tại Đại học Dân lập Hải Phòng từ tháng 4 năm 2017. Cho đến nay hệ thống phần mềm quản lý tài liệu số vẫn hoạt động ổn định và có hiệu quả. Sau khi ảo hóa, tháng 7-2017 Thư viện số Dspace của Trường đã được tổ chức quốc tế Webometric xếp hạng số 1 trong số các thư viện số các trường Đại học tại Việt Nam.

- Kết quả đạt được của luận văn:

Luận văn đã đạt được những kết quả sau:

- ✓ Tổng quan được các công nghệ ảo hóa và đã lựa chọn được công nghệ ảo hóa phù hợp với Đại học Dân lập Hải Phòng
- ✓ Khảo sát và phân tích hiện trạng của hệ thống mạng, máy chủ, ứng dụng tại Đại học Dân lập Hải Phòng
- ✓ Đề xuất về ảo hóa máy chủ thích hợp và sử dụng Docker cho việc ảo hóa ứng dụng tại trường
- ✓ Đưa ra quy trình và cách thức ảo hóa ứng dụng tại Đại học Dân lập Hải Phòng
- ✓ Thực nghiệm Sử dụng Docker ảo hóa thư viện số Dspace

- Hạn chế của luận văn:

- ✓ Các dữ liệu còn giả định
- ✓ Chưa làm được an ninh bảo mật

- Hướng phát triển, mở rộng:

- ✓ Nghiên cứu sâu hơn về Docker áp dụng cho các mục đích khác nhau: nghiên cứu, phát triển ứng dụng, thương mại, ...
- ✓ Xây dựng dịch vụ ảo hóa máy chủ cho trường Đại học Dân lập Hải Phòng.

Tài liệu tham khảo

A.Tiếng Việt

[1] Trần Văn Đoàn (2013), Luận văn thạc sỹ, *Công nghệ ảo hoá và ứng dụng*, Học viện công nghệ bưu chính viễn thông.

[2] Lê Trung Nghĩa dịch (2010), *Giới thiệu Phần mềm Tự do*, Nhà xuất bản Thông tin và Truyền thông.

[3] Lê Trung Nghĩa dịch (2010), *Triển khai các hệ thống phần mềm tự do*, Nhà xuất bản Thông tin và Truyền thông.

[4] Lê Văn Phùng (2014), *Kỹ nghệ phần mềm*, Nhà xuất bản Thông tin và Truyền thông.

[5] Lê Văn Phùng (2014), *Hệ thống thông tin quản lý*, Nhà xuất bản Thông tin và Truyền thông.

[6] Trần Hải Phương (2015), Luận văn thạc sỹ, *Nghiên cứu công nghệ ảo hóa và ứng dụng xây dựng hệ thống thông tin doanh nghiệp*, Viện Đại học mở Hà Nội.

B.Tiếng Anh

[7] James Turnbull (2014), *TheDocker Book v1.0.7*, ISBN 978-0-9888202-0-3.

[8] James Turnbull (2015), *TheDocker Book v1.9.1*, ISBN 978-0-9888202-0-3.

[9] Linux Foundation (2015), Guide to the Open Cloud, [Internet], Available from: URL: www.linuxfoundation.org

C. Internet

[10] http://docker.com, truy cập 06/2017.

[11] https://github.com/docker, truy cập 06/2017.

[12] https://www.wikipedia.org, truy cập 06/2017.