

BAN CƠ YẾU CHÍNH PHỦ  
**HỌC VIỆN KỸ THUẬT MẬT MÃ**

-----



BÁO CÁO THỰC TẬP CHUYÊN NGÀNH

**NGHIÊN CỨU, TRIỂN KHAI OPENSTACK SỬ DỤNG  
DOCKER**

Ngành: An toàn thông tin  
Mã số: 7.48.02.02

*Sinh viên thực hiện:*

**Dương Quang Huy**

Mã sv: AT150323

**Nguyễn Việt Hoàng**

Mã sv: AT150319

**Chu Viết Long Vũ**

Mã sv: AT150363

*Giáo viên hướng dẫn thực tập:*

**TS. Phạm Văn Hưởng**

Khoa Công nghệ thông tin – Học viện Kỹ thuật mật mã

**Hà Nội – 2021**

## MỤC LỤC

<b>MỤC LỤC</b> .....	<b>i</b>
<b>DANH MỤC KÍ HIỆU VÀ VIẾT TẮT</b> .....	<b>1</b>
<b>DANH MỤC HÌNH ẢNH</b> .....	<b>2</b>
<b>DANH MỤC BẢNG</b> .....	<b>3</b>
<b>LỜI CẢM ƠN</b> .....	<b>4</b>
<b>LỜI NÓI ĐẦU</b> .....	<b>5</b>
<b>CHƯƠNG 1: TỔNG QUAN VỀ ẢO HOÁ VÀ CLOUD COMPUTING</b> .....	<b>6</b>
1.1. Ảo hoá( Vitrualization) .....	6
1.1.1. Ảo hoá hỗ trợ phần cứng .....	7
1.1.2. Ảo hoá cấp độ hệ điều hành.....	11
1.2. Tổng quan về cloud computing.....	12
1.2.1. Những lợi ích của cloud computing:.....	13
<b>CHƯƠNG 2: GIỚI THIỆU VỀ DOCKER</b> .....	<b>14</b>
2.1. Khái niệm .....	14
2.2. Container trong Docker.....	15
2.3. Lý do nhóm chọn triển khai openstack bằng docker .....	15
<b>CHƯƠNG 3: GIỚI THIỆU VỀ OPENSTACK</b> .....	<b>16</b>
3.1. Amazon Web Service - nguồn cảm hứng cho sự ra đời của Openstack.....	16
3.2. Lịch sử về Openstack .....	17
3.3. Tổng quan về Openstack.....	18
3.4. Các thành phần của Openstack .....	19
3.5. Kiến trúc Conceptual.....	21
3.6. Kiến trúc Logic.....	22
3.7. Kiến trúc logic một số thành phần cơ bản.....	23
3.7.1. Identity(Keystone) .....	24
3.7.2. Compute(Nova) .....	25
3.7.3. Networking(Neutron) .....	28
3.7.4. Block Storage( Cinder) .....	29
3.7.5. Image Service( Glance).....	31
3.7.6. Dashboard( Horizon).....	32
<b>CHƯƠNG 4: TRIỂN KHAI OPENSTACK</b> .....	<b>32</b>
4.1. Yêu cầu hệ thống.....	32
4.2. Các bước triển khai .....	33
4.2.1. Cập nhật và nâng cấp các package ubuntu .....	33

4.2.2. Cài đặt python3 và các package để có thể tạo môi trường deploy bằng python.....	33
4.2.3. Cài đặt ansible ubuntu .....	34
4.2.4. Cài đặt và cấu hình kola-ansible cho ubuntu .....	34
4.2.5. Deploy region 1.....	36
4.2.6. Deploy region 2.....	36
<b>CHƯƠNG 5: KẾT QUẢ TRIỂN KHAI .....</b>	<b>38</b>
<b>KẾT LUẬN .....</b>	<b>45</b>
<b>DANH MỤC TÀI LIỆU THAM KHẢO.....</b>	<b>46</b>

## DANH MỤC KÍ HIỆU VÀ VIẾT TẮT

CC	Cloud computing	Điện toán đám mây
IaaS	Infrastructure as a Service	Hạ tầng như một dịch vụ
Paas	Platform as a Service	Môi trường như một dịch vụ
SaaS	Software as a Service	Phần mềm như một dịch vụ

## DANH MỤC HÌNH ẢNH

Hình 1: Sự khác biệt về kiến trúc máy tính giữa công nghệ truyền thống với công nghệ ảo hóa .....	6
Hình 2 Minh hoạ về ảo hoá hỗ trợ phần cứng.....	7
Hình 3: Hai cơ chế ảo hoá.....	8
Hình 4: Hai loại hypervisor.....	9
Hình 5: Minh hoạ về hypervisor type-1 .....	10
Hình 6: Minh hoạ về hypervisor type-2 .....	11
Hình 7: Minh hoạ về ảo hoá cấp độ hệ điều hành.....	11
Hình 8: Minh hoạ về mô hình cloud computing .....	13
Hình 9: Kiến trúc tổng thể các dịch vụ openstack.....	19
Hình 10: kiến trúc conceptual .....	22
Hình 11: kiến trúc logic phổ biến.....	23
Hình 12: Các thành phần trong keystone .....	24
Hình 13: Minh hoạ về kiến trúc nova trong openstack .....	27
Hình 14: Các thành phần trong neutron .....	28
Hình 15: Các thành phần trong cinder.....	30
Hình 16: Các thành phần trong glance .....	31
Hình 17: Trang login openstack.....	39
Hình 18: Trang overview.....	39
Hình 19: Danh sách các Image .....	40
Hình 20: Danh sách các Instance.....	40
Hình 21: Instance khi đang hoạt động .....	41
Hình 22: Console của Instance khi đang hoạt động.....	41
Hình 23: Log của Instance khi đang hoạt động .....	41
Hình 24: Danh sách các network đã triển khai.....	42
Hình 25: Mô hình Network đã triển khai.....	42
Hình 26: Danh sách các user đã tạo .....	43
Hình 27: Danh sách các service đang chạy trên node 1 .....	43
Hình 28: Danh sách các service đang chạy trên node 2 .....	44

## DANH MỤC BẢNG

<i>Bảng 1: Danh sách các openstack service .....</i>	<i>20</i>
<i>Bảng 2: Danh sách các openstack operations tooling .....</i>	<i>21</i>
<i>Bảng 3: Danh sách các openstack add-ons to services .....</i>	<i>21</i>
<i>Bảng 4: Danh sách các openstack add-ons to services .....</i>	<i>21</i>
<i>Bảng 5: Các thông tin cấu hình cơ bản của node1 .....</i>	<i>33</i>
<i>Bảng 6: Các thông tin cơ bản của node2 .....</i>	<i>33</i>

## LỜI CẢM ƠN

Trong thời gian làm báo cáo thực tập, chúng em xin chân thành cảm ơn quý thầy cô đã tận tình giảng dạy, giúp đỡ em có nền tảng kiến thức để hoàn thành đề tài này.

Đặc biệt chúng em xin chân thành cảm ơn sự nhiệt tình hướng dẫn và đóng góp ý kiến của thầy Phạm Văn Hưởng đã giúp chúng em hoàn thành tốt đề tài thực tập này.

Mặc dù đã cố gắng hoàn thành đề tài trong phạm vi và khả năng cho phép nhưng vẫn không tránh khỏi những thiếu sót. Kính mong nhận được sự thông cảm và tận tình đóng góp ý kiến của quý thầy cô và các bạn.

Em xin chân thành cảm ơn!

## SINH VIÊN THỰC HIỆN ĐỀ TÀI

*Dương Quang Huy*

*Nguyễn Việt Hoàng*

*Chu Viết Long Vũ*

## LỜI NÓI ĐẦU

Trước khi cloud computing ra đời, những công ty lớn, những tập đoàn lớn thường cài đặt tất cả các ứng dụng hay phần mềm trên những cụm máy chủ của họ. Một công ty sẽ có một hệ thống máy chủ, 2 công ty sẽ là 2, và 1000 công ty sẽ sở hữu con số máy chủ tương ứng dần dần số lượng, chi phí phát sinh từ hệ thống máy chủ ngày càng phình to. Bởi vậy, để giảm tải các chi phí phát sinh từ hệ thống máy chủ đồ sộ của các công ty riêng lẻ, cloud computing đã được ra đời. Cloud computing (CC) đang là chủ đề được bàn luận sôi nổi nhất hiện nay, các công nghệ liên quan đến “cloud” nhận được rất nhiều quan tâm từ các doanh nghiệp. Đã có khá nhiều sản phẩm thương mại cũng như nguồn mở miễn phí được giới thiệu cung cấp cho người dùng khả năng xây dựng các thành phần của CC, từ hạ tầng IaaS đến PaaS và SaaS.

Qua quá trình tìm hiểu các công nghệ và triển khai thử các IaaS, nhóm em đã tìm được một mã nguồn mở cho việc triển khai CC có khả năng chịu lỗi cũng như mở rộng cao của hệ thống, được nhiều tổ chức lớn sử dụng: Openstack. Ngoài việc giúp cho các doanh nghiệp khả năng xây dựng các đám mây riêng phục vụ cho công việc nội bộ, openstack còn có thể giúp doanh nghiệp xây dựng các đám mây để cung cấp dịch vụ liên quan tới CC.

Đề tài “Nghiên cứu, triển khai openstack sử dụng docker” nhằm mục đích nghiên cứu, triển khai một mô hình cho hệ thống điện toán đám mây. Đồng thời đóng gói hệ thống để cung cấp triển khai sang các cụm máy chủ khác nhau một cách dễ dàng, nhanh chóng, dễ dàng mở rộng và ổn định nhất có thể.

Trong phần đầu của báo cáo sẽ giới thiệu một số khái niệm về CC và các công nghệ ảo hóa.

Phần tiếp theo xin được trình bày về Openstack, các công việc thử nghiệm đã và kết quả đạt được trong việc tìm hiểu và triển khai bằng công cụ docker. Các tài liệu tham khảo cũng như hướng dẫn chi tiết về cài đặt, cấu hình... được đính kèm trong phần phụ lục tham khảo.

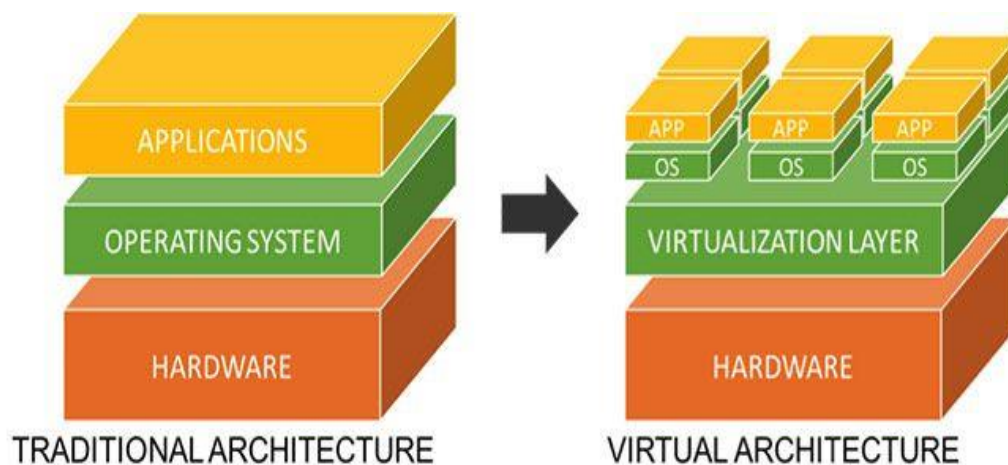


# CHƯƠNG 1: TỔNG QUAN VỀ ẢO HOÁ VÀ CLOUD COMPUTING

## 1.1. Ảo hoá (Virtualization)

Trong máy tính, ảo hóa (virtualization) có nghĩa là tạo một phiên bản ảo trên nền của tài nguyên máy tính thật như ram, cpu, network, ... Từ nền tảng của kỹ thuật ảo hoá này, chúng ta có thể tạo ra một instance vật lý hoặc một ứng dụng để chia sẻ cho nhiều người dùng và tổ chức khác nhau

Ý tưởng của ảo hoá không phải là thứ gì mới mẻ, ý tưởng này đã được IBM giới thiệu từ năm 1960 khi các server của công ty này hầu như không được sử dụng hết hiệu suất cũng như tính năng. Họ đã chọn phương pháp ảo hoá để tối ưu việc cung cấp tài nguyên hệ thống cho các ứng dụng khác nhau tùy vào mức độ sử dụng tài nguyên



Hình 1: Sự khác biệt về kiến trúc máy tính giữa công nghệ truyền thống với công nghệ ảo hóa

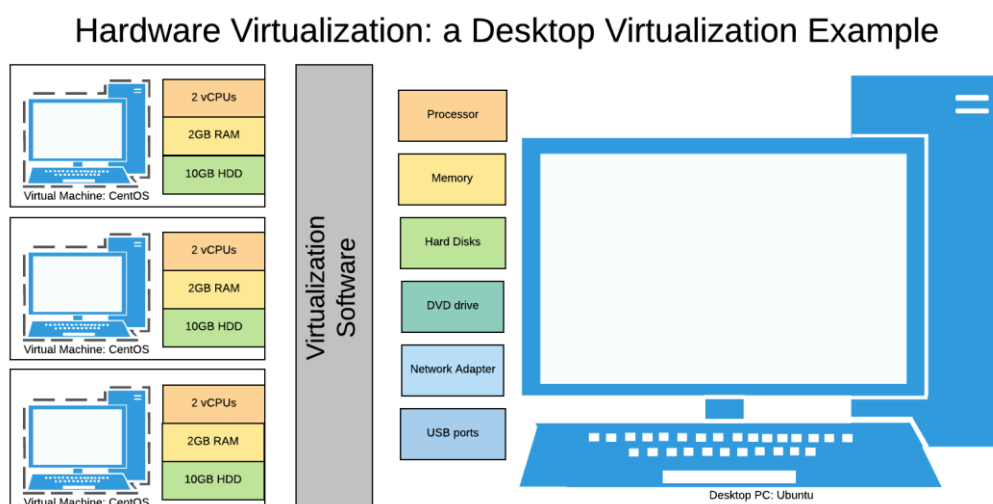
Do sự phát triển của công nghệ như Utility Computing và Cloud Computing, công nghệ ảo hóa được chú trọng hơn trong sự phát triển của các công nghệ mới gần đây.

Có 2 loại ảo hoá phổ biến và nhu cầu thị trường cao là ảo hoá hỗ trợ phần cứng và ảo hoá Ảo hóa cấp độ hệ điều hành chúng tôi sẽ nêu dưới đây

### 1.1.1. Ảo hoá hỗ trợ phần cứng

Ảo hóa hỗ trợ phần cứng là phương thức ảo hóa toàn phần (full virtualization), cho phép tạo các máy ảo hoạt động với tài nguyên vật lý độc lập.

Với phương thức ảo hóa này, các máy ảo sẽ làm việc giống y như máy chủ vật lý thật, sử dụng hoàn toàn các tài nguyên vật lý được cấp phát và có thể cài đặt, quản lý 100% hệ điều hành trên đó.



Hình 2 Minh hoạ về ảo hoá hỗ trợ phần cứng

Server sử dụng công nghệ ảo hóa phần cứng sẽ được sở hữu hoàn toàn các tài nguyên vật lý đã cấp cho nó mà không san sẻ cho các server ảo khác.

Để vận hành ảo hóa hỗ trợ phần cứng ta có Hypervisor:

#### Hypervisor là gì?

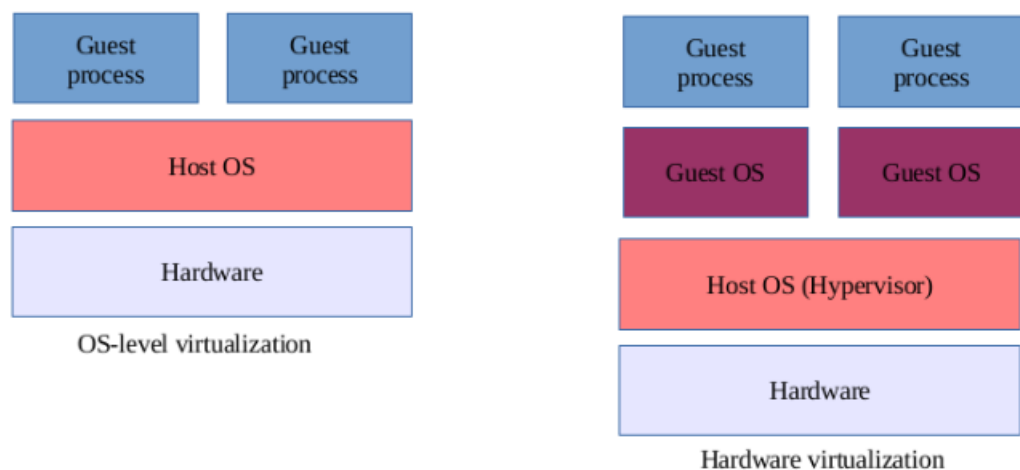
Theo Redhat, hypervisor là phần mềm khái quát phần cứng từ một hệ điều hành cho phép nhiều hệ điều hành cùng chạy trên cùng nền tảng phần cứng. Hypervisor chạy trên hệ thống cho phép các máy ảo chạy trên nền phần cứng của host.

Theo VMWare, hypervisor là phần mềm cung cấp các tính năng phân vùng ảo chạy trực tiếp trên phần cứng, nhưng ảo hóa các dịch vụ mạng ở mức tối đa.

Vai trò của Hypervisor là tạo ra một môi trường giả lập máy thật, nhờ đó mà các hệ điều hành khách (Guest OS) có thể chạy trên các máy ảo y hệt trên máy vật lý.

Hypervisor có thể là hardware, nhưng thường là software, hoặc firmware (phần mềm cấp thấp, chạy trực tiếp trên phần cứng không cần hệ điều hành).

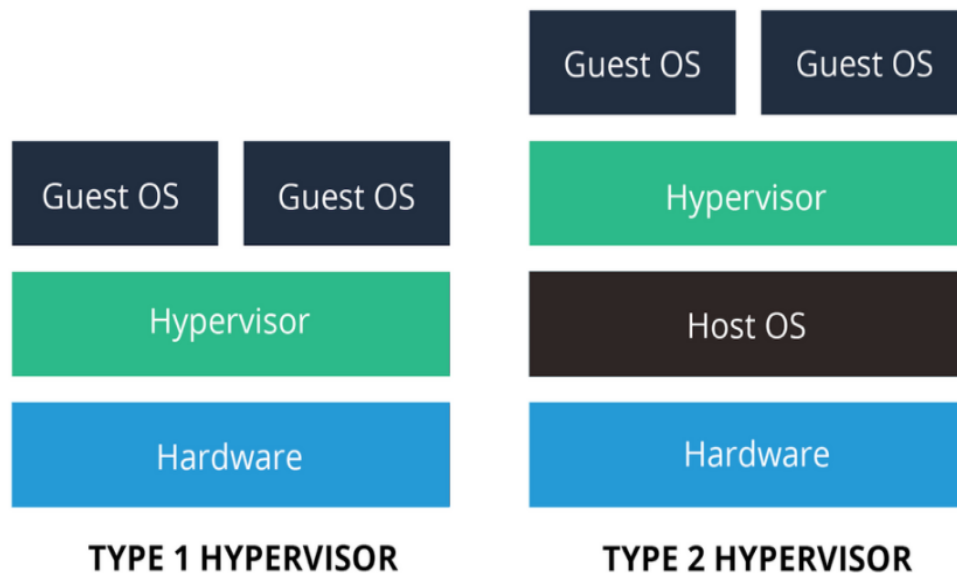
Hypervisor biểu diễn bằng một nền tảng vận hành ảo chứa Guest OS và quản lý vận hành Guest OS. Các instance trong các hệ điều hành có thể chia sẻ nhau các tài nguyên phần cứng ảo, chẳng hạn, các instance chứa Linux, Windows và macOS có thể cùng chạy trên một máy tính đơn x86. Cơ chế Hypervisor trái ngược với OS virtualization, tại đó tất cả các instance (container) phải chia sẻ cùng một nhân (kernel) thông qua Guest OS để phân chia các không gian sử dụng khác nhau



Hình 3: Hai cơ chế ảo hoá

Do sự phát triển của công nghệ ảo hóa nên các nền tảng phần cứng cũng có sự thay đổi. Intel hay AMD đã thiết kế các hệ thống vi xử lý mới mở rộng từ kiến trúc x86 tương ứng với những công nghệ được biết ngày nay là Intel VT-x hay AMD-V. Chipset Intel 80286 đã được giới thiệu về 2 phương thức về địa chỉ bộ nhớ: địa chỉ bộ nhớ thực (real mode) và địa chỉ bộ nhớ ảo (protected mode). Địa chỉ bộ nhớ ảo cung cấp các tính năng hỗ trợ multicasting như phần cứng hỗ trợ bộ nhớ ảo và thành phần vi xử lý.

Dựa trên nền tảng đó, hypervisor được phân thành 2 loại: Type 1 hypervisor(bare-metal) và type 2 hypervisor(hosted)

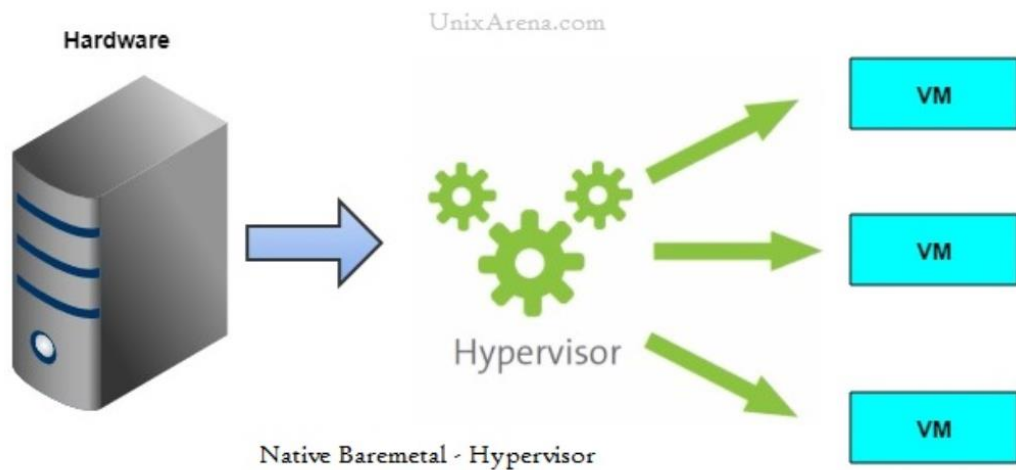


Hình 4: Hai loại hypervisor

\* )Hypervisor Type-1:

Một hypervisor ở dạng native (hay còn gọi “bare-metal”) chạy trực tiếp trên phần cứng. Nó nằm giữa phần cứng và một hoặc nhiều hệ điều hành khách (guest operating system). Nó được khởi động trước cả hệ điều hành và tương tác trực tiếp với kernel. Điều này mang lại hiệu suất cao nhất có thể vì không có hệ điều hành chính nào cạnh tranh tài nguyên máy tính với nó. Tuy nhiên, nó cũng đồng nghĩa với việc hệ thống chỉ có thể được sử dụng để chạy các máy ảo vì hypervisor luôn phải chạy ngầm bên dưới.

Các hypervisor dạng native này có thể kể đến như VMware ESXi, Microsoft Hyper-V, XEN và Apple Boot Camp.



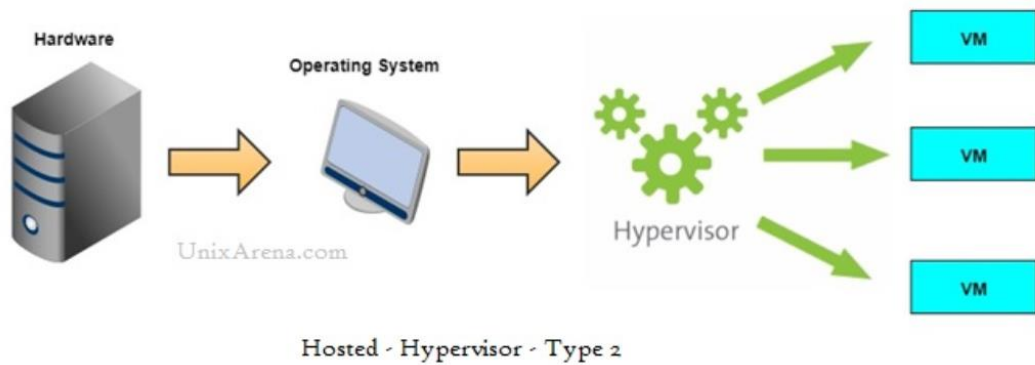
Hình 5: Minh họa về hypervisor type-1

**\*) Hypervisor Type-2:**

Một hypervisor dạng hosted được cài đặt trên một máy tính chủ (host computer), mà trong đó có một hệ điều hành đã được cài đặt. Nó chạy như một ứng dụng cũng như các phần mềm khác trên máy tính. Hầu hết các hypervisor dạng hosted có thể quản lý và chạy nhiều máy ảo cùng một lúc. Lợi thế của một hypervisor dạng hosted là nó có thể được bật lên hoặc thoát ra khi cần thiết, giải phóng tài nguyên cho máy chủ. Tuy nhiên, vì chạy bên trên một hệ điều hành nên hiệu suất sẽ không cao bằng một hypervisor ở dạng native.

Hypervisor Type-2 thì đa số dân IT đều biết, ví dụ bạn muốn cài Linux (Ubuntu) trên Windows 7, bạn cần cài phần mềm máy ảo VirtualBox, còn muốn cài Windows trên MacOS, thì dùng Parallels Desktop for Mac... đây đều là các phần mềm đóng vai trò của Hypervisor.

Các loại Hypervisor Type -2 phổ biến là VirtualBox, Parallels Desktop cho Mac, dùng cho Server thì có VMware Workstation, VMware Player, QEMU



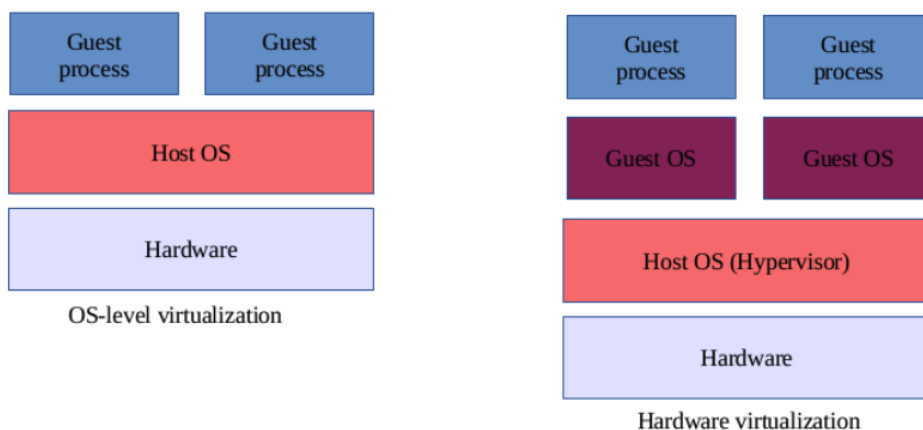
Hình 6: Minh hoạ về hypervisor type-2

Một số công nghệ ảo hoá dựa trên phần cứng tiêu biểu như: Xen, KVM, VMware ESXi, Hyper-V

### 1.1.2. Ảo hoá cấp độ hệ điều hành

Ảo hóa cấp độ hệ điều hành – OS-level virtualization là phương thức ảo hóa thực hiện trực tiếp trên hệ điều hành, mỗi máy ảo sẽ chạy trên một ‘trạng thái’ hệ điều hành riêng và chia sẻ với nhau toàn bộ tài nguyên vật lý của máy chủ.

Phương thức ảo hóa này không cấp phát ‘tài nguyên cứng’ cho mỗi máy chủ ảo như trong ảo hóa hỗ trợ phần cứng.



Hình 7: Minh hoạ về ảo hoá cấp độ hệ điều hành

Tùy vào công nghệ áp dụng mà ‘trạng thái hệ điều hành’ này được gọi là Instance, Container, Docker...

Ở mỗi máy chủ ảo, cho phép cô lập về phần mềm và quản lý – giới hạn tài nguyên sử dụng. Tức là bạn có thể set cho một máy ảo lượng tài nguyên tối đa (CPU, RAM, I/O, Network..) mà nó có thể sử dụng, khi nó không sử dụng hết lượng này, tài nguyên vật lý có thể được chuyển sang cho các máy chủ ảo khác.

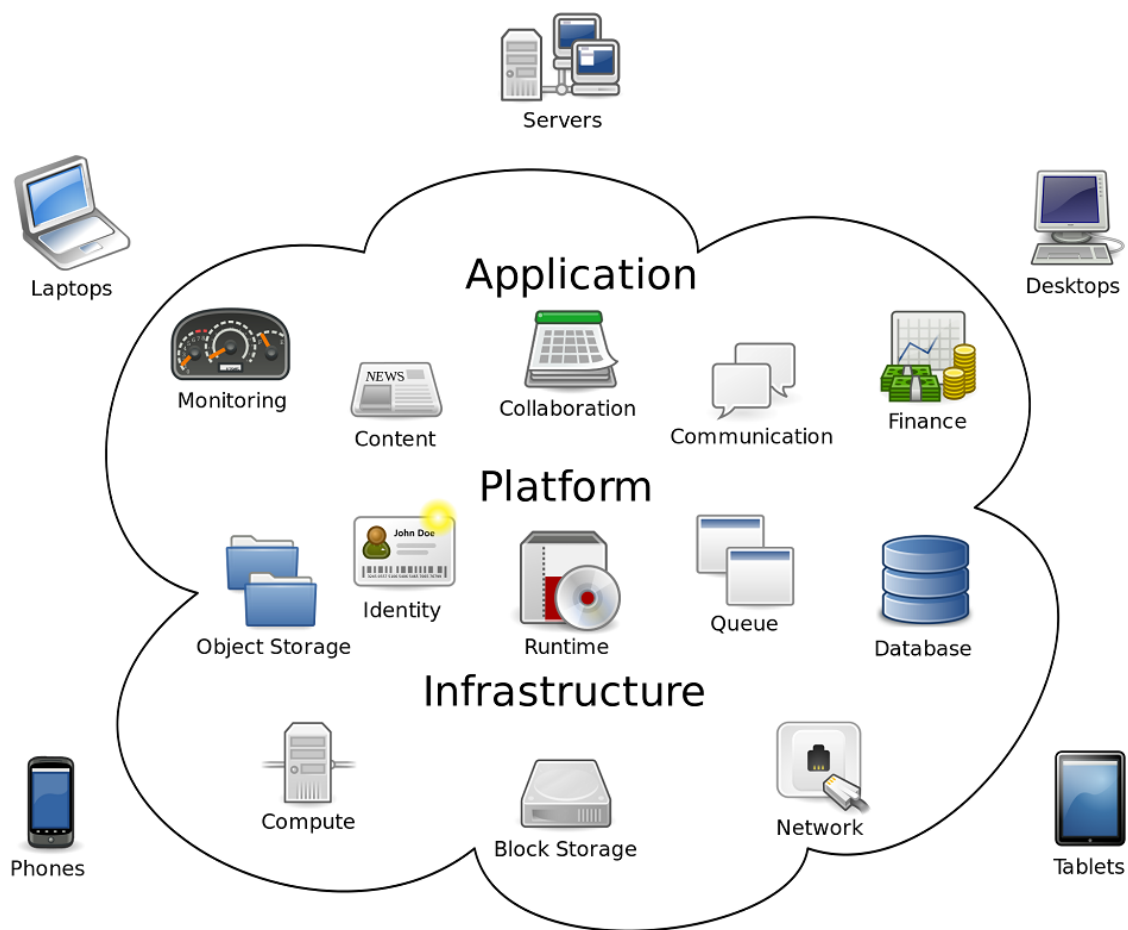
Một số công nghệ Ảo hóa cấp độ hệ điều hành: OS-level virtualization, OpenVZ, Virtuozzo, LXC, Docker

Trong đề tài lần này nhóm sử dụng công nghệ docker để triển khai openstack, nhóm sẽ nói về docker rõ hơn ở chương 2

## **1.2. Tổng quan về cloud computing**

Cloud computing là sự tổng hòa các khái niệm như Web service, Web 2.0 và các khái niệm mới khác cũng như các xu hướng công nghệ nổi bật, dựa trên nền tảng Internet nhằm đáp ứng nhu cầu của người sử dụng. Ví dụ, dịch vụ Google Application Engine hay Amazon EC2 cung cấp những ứng dụng liên quan đến mua bán trực tuyến, được truy nhập từ một trình duyệt web, còn các phần mềm và dữ liệu đều được lưu trữ trên các server hay các datacenter.

Cloud computing còn được định nghĩa là mô hình cung cấp các tài nguyên hệ thống máy tính (như network, server, storage, ứng dụng và dịch vụ), đặc biệt là khả năng lưu trữ và khả năng tự động xử lý mà người dùng không quản trị một cách trực tiếp. Cloud computing còn được mô tả việc nhiều người dùng sử dụng tài nguyên của các data center thông qua Internet. Các hệ thống Cloud computing thường phân tán các tính năng tại các vị trí khác nhau trong các cụm server



Hình 8: Minh họa về mô hình cloud computing

### 1.2.1. Những lợi ích của cloud computing:

Tiết kiệm chi phí chính là lợi ích đầu tiên mà Cloud Computing mang lại cho người dùng. Thay vì phải bỏ chi phí đầu tư cả một hệ thống máy chủ để lưu trữ dữ liệu, chịu các chi phí vận hành hay bảo dưỡng hàng năm thì bạn chỉ cần phải dành một khoản tiền nhỏ để duy trì chúng. Bạn sẽ tập trung hơn vào công việc của mình thay vì phải lo lắng đến những việc như bảo dưỡng hay vận hành hệ thống.

Lợi ích thứ hai của điện toán đám mây đó là khả năng mở rộng linh hoạt về quy mô của nó. Điều này có nghĩa là bạn sẽ được phân phối đúng lượng tài nguyên cần thiết với nhu cầu sử dụng của mình. Bạn có thể bổ sung tài nguyên bất cứ khi nào phát sinh nhu cầu và tại đúng vị trí địa lý mà bạn mong muốn.

Lợi ích thứ ba là vấn đề hiệu năng. Các dịch vụ điện toán đám mây lớn nhất chạy trên mạng lưới trung tâm dữ liệu an toàn trên toàn thế giới. Và đương nhiên chúng được nâng cấp thường xuyên để tăng hiệu quả và mức độ bảo mật. So với



mô hình trung tâm dữ liệu quy mô nhỏ như một công ty thì điều này mang lại những lợi ích tích cực hơn như giảm độ trễ mạng và tăng tính kinh tế khi áp dụng theo quy mô lớn hơn.

Lợi ích thứ tư là tính bảo mật. Nhiều nhà cung cấp dịch vụ đám mây cung cấp một loạt các chính sách, công nghệ và kiểm soát nhằm củng cố tính bảo mật của bạn. Qua đó nó giúp bảo vệ dữ liệu, ứng dụng và cơ sở hạ tầng của bạn khỏi các mối đe dọa tiềm ẩn hay các cuộc tấn công mạng vẫn thường xuyên xảy ra.

Lợi ích thứ năm là tốc độ. Hầu hết các dịch vụ Cloud Computing hiện nay đều được cung cấp dịch vụ theo yêu cầu. Nghĩa là bạn cần bao nhiêu thì có bấy nhiêu. Thậm chí ngay cả một lượng lớn tài nguyên máy tính cũng có thể được cung cấp chỉ trong vài phút. Như vậy bạn sẽ không cần phải quá áp lực trong việc lên kế hoạch tính toán công suất cho phù hợp nữa. Thay vào đó bạn có thể sử dụng bình thường, khi cần có thể bổ sung ngay tức thì chỉ với một vài cú nhấp chuột.

Lợi ích thứ sáu là vấn đề về năng suất. Đối với Cloud Computing, bạn sẽ không phải dành chi phí cũng như cắt cử nhân viên cho các tác vụ quản lý, bảo dưỡng và duy trì hệ thống công nghệ thông tin. Như vậy bạn có thể tập trung đội ngũ cho những việc chuyên môn phục vụ cho kinh doanh của mình nhiều hơn.

Lợi ích cuối cùng là tính tin cậy của hệ thống. Các đơn vị trung gian chuyên cung cấp các dịch vụ về Cloud, Máy chủ, ... như Viettel IDC luôn có các biện pháp giúp người dùng sao lưu và bảo vệ dữ liệu. Thậm chí họ còn có các trung tâm DC/DR giúp khôi phục dữ liệu khi bị tấn công mạng và duy trì hoạt động liên tục của hệ thống. Đây là những thứ mà trong phạm vi nhỏ như một công ty đơn lẻ sẽ khó có thể đáp ứng được.

## **CHƯƠNG 2: GIỚI THIỆU VỀ DOCKER**

### **2.1. Khái niệm**

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

## 2.2. Container trong Docker

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

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

Theo một cách nào đó, Docker khá giống virtual machine. Nhưng tại sao Docker lại phát triển, phổ biến nhanh chóng? Đây là những nguyên nhân:

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

## 2.3. Lý do nhóm chọn triển khai openstack bằng docker

- Triển khai kiến trúc Microservices của openstack nhanh chóng, hiệu quả
- Có thể scale một hoặc một số thành phần trong openstack một cách linh hoạt
- Dễ dàng triển khai trên các cụm máy chủ khác nhau sau khi đóng gói các image
- Dễ dàng kiểm soát tài nguyên của một container

## CHƯƠNG 3: GIỚI THIỆU VỀ OPENSTACK

### 3.1. Amazon Web Service - nguồn cảm hứng cho sự ra đời của Openstack

Phần này sẽ giới thiệu sơ lược về một trong những nhà cung cấp dịch vụ về CC hàng đầu hiện nay – Amazon. Amazon đã xây dựng được một hệ thống dịch vụ AWS cơ bản khá hoàn chỉnh và ổn định về IaaS và các dịch vụ đi kèm. Tiếp nữa AWS chính là nguồn cảm hứng để tạo ra những nền tảng về IaaS như Eucalyptus, Openstack...sau này. Tại sao lại như vậy? Chúng ta sẽ lướt qua một số mốc thời gian, trở lại khoảng 10 năm trước tại thời điểm mà hầu như chưa có mấy công ty có khái niệm về CC, tuy nhiên đã có một số người có ý tưởng về việc cung cấp phần mềm, hạ tầng...như là một dịch vụ.

Nhắc đến CC chúng ta thường nghĩ ngay đến những tên tuổi như Google, Microsoft, Apple... Tuy nhiên thực tế, họ không phải là những người đi đầu trong công nghệ cũng như ứng dụng về Cloud computing. Thực sự về tầm nhìn sớm và mức độ ứng dụng về CC thì phải nói đến Salesforce và tiếp đó là Amazon.

Salesforce đã bắt đầu từ rất sớm với CC, ngay từ năm 1999 hãng đã có định hướng phát triển về SaaS, từ việc cung cấp các dịch vụ quản lý khách hàng, kế toán, thống kê tài chính... Theo như báo cáo kinh doanh năm 2011, mảng dịch vụ về SaaS đã đem lại cho Salesforce hơn 3 tỉ USD đó là một con số đáng ngưỡng mộ. Ngay cả Google hay Microsoft những tên tuổi 'non trẻ' trong cùng mảng kinh doanh về CC cũng phải ghen tị với thành tích này.

Không dừng lại ở mức độ cung cấp về SaaS như Salesforce, Amazon từ một công ty bán lẻ các mặt hàng dân dụng, điện tử, sách...đã dần vươn lên và có thể nói là tên tuổi lớn nhất hiện nay về dịch vụ hạ tầng cho CC. Cách đây hơn 10 năm, sau khi tồn tại qua đợt khủng hoảng bong bóng dot com, Amazon đã dần chứng minh phương châm bán hàng qua mạng của họ là đúng đắn. Là công ty có tốc độ phát triển nhanh nhất sau 5 năm đầu tiên (từ năm 1995-2000 doanh thu là 2.8 tỉ USD) vượt xa Google (1998-2003 doanh thu 1.5 tỉ USD). Ban đầu tưởng chừng đối thủ cạnh tranh của Amazon chỉ là Walmart hay BestBuy, eBay - những công ty bán lẻ. Giờ đây Amazon đã lấn sân và kinh doanh trong 16 lĩnh vực khác nhau trong đó mạnh nhất vẫn là lĩnh vực bán lẻ tiếp đến là các dịch vụ về CC.

Amazon thực sự đã xây dựng được một đế chế công nghệ hùng mạnh, cạnh tranh trực tiếp với các nhà cung cấp dịch vụ hosting truyền thống cũng như CC như Rackspace, GoDaddy, Google... Theo nhận định của giới chuyên môn Amazon đã tạo ra một kiến trúc về CC kinh điển AWS với đầy đủ các dịch vụ về tính toán, lưu trữ, cơ sở dữ liệu chuyên dụng... Thực tế cho thấy hầu hết các nền tảng khác như Eucalyptus, Openstack... đều được xây dựng theo một kiến trúc, các thành phần tựa như AWS. Tất nhiên chưa có một khẳng định nền tảng của ai tốt

hơn một cách rõ ràng, nhưng với những đánh giá về tính ổn định, hiệu năng và quan trọng nhất là giá của dịch vụ. AWS vẫn đang là sản phẩm tốt nhất hiện nay.

*Một số dịch vụ chính của AWS:*

- Amazon Elastic Cloud Compute (EC2) cung cấp các instance (máy ảo) tùy theo nhu cầu, với khả năng tính toán, mở rộng vô cùng linh hoạt. Hiểu đơn giản, EC2 cung cấp cho người dùng khả năng tạo các máy ảo trên hạ tầng của Amazon, họ có thể cấp phát tài nguyên (CPU, RAM) theo yêu cầu, và từ đó Amazon sẽ tính toán các chi phí. Các instance có các mức cấu hình khác nhau: nhỏ nhất là micro instance (1 CPU, 613 MB RAM) và lớn nhất tới hơn 64GB RAM và 88 EC2 CPU (tương đương 2 x Intel Xeon E5-2670)
- Amazon Elastic Block (EBS) cung cấp khả năng lưu trữ độc lập, kết hợp với EC2. Hiểu đơn giản giống như việc sử dụng thêm các ổ đĩa mở rộng trên các máy vật lý. Khi mà có sự cố tại instance thì dữ liệu lưu trên EBS vẫn có thể sử dụng độc lập, và có thể chia sẻ giữa những instance khác nhau
- Amazon Simple Storage Service (S3) cung cấp khả năng lưu trữ không hạn chế, cũng giống như EBS, S3 giải quyết vấn đề về lưu trữ, tuy nhiên EBS được sử dụng bởi các instance thì S3 được sử dụng như một ổ đĩa mạng. Thông qua một giao diện (web hay một GUI) người dùng có thể lưu trữ dữ liệu của mình, backup dữ liệu từ các nguồn khác nhau (từ chính EBS, EC2...) S3 sử dụng cơ sở dữ liệu Dynamo để quản lý việc lưu trữ, chứ không sử dụng các CSDL quan hệ truyền thống vì đối với dịch vụ lưu trữ, người dùng chủ yếu đọc và ghi dữ liệu nên nếu lưu theo mô hình quan hệ sẽ không giải quyết hiệu quả

### **3.2. Lịch sử về Openstack**

Trong phần giới thiệu về AWS ở trên, chúng ta cơ bản nắm được một số chức năng mà một sản phẩm thương mại hiện tại đang cung cấp được cho khách hàng, từ đó ta có thể so sánh một cách tương đối giữa những chức năng mà gói công cụ nguồn mở này đã thực hiện được. Để làm rõ thêm lý do lấy AWS làm 'đối chiếu', xin được trích qua một số mốc quan trọng dẫn tới sự ra đời của Openstack.

Trở lại mốc 2005 khi mà Amazon ra mắt thử nghiệm EC2, đó là một thành công lớn gây bất ngờ cho cộng đồng. Với sự ổn định của nó, các công ty khác có thể đơn giản “thuê” EC2 trong một vài giờ với một mức năng lực rất rất lớn để thực hiện các công việc tính toán cần tới hiệu năng cao của họ. Ví dụ mà Amazon thường đem ra so sánh là việc hợp tác giữa họ và NASDAQ - sàn chứng khoán cần

xử lý một lượng dữ liệu tính toán cực lớn vào cuối tuần, thay vì đầu tư một hệ thống máy chủ phức tạp, họ chỉ thuê EC2 trong vài giờ và chi phí tiết kiệm rất rất nhiều hơn nữa hiệu quả công việc lại tốt hơn.

Một trong những công ty cần sử dụng khả năng tính toán hiệu năng cao kiểu như thế là NASA. Họ có kế hoạch tái cấu trúc lại trung tâm dữ liệu của họ, và họ cần một nền tảng IaaS để có thể sử dụng tốt hơn hạ tầng vật lý mà họ có. Amazon EC2 là một tấm gương tốt đáng ngưỡng mộ. Vào khoảng năm 2008 NASA bắt đầu sử dụng tham gia vào Eucalyptus một dự án nhằm cung cấp một IaaS giống như AWS (EC2 và S3). Tuy nhiên không như mong muốn của NASA, Eucalyptus không phải là một dự án mở hoàn toàn, công ty đỡ đầu cho nó không cho phép NASA xem một số thành phần đóng kín của Eucalyptus. Rạn nứt bắt đầu từ đây.

Sau đó NASA bắt đầu nghiên cứu dự án riêng của họ cũng với mục đích xây dựng một hạ tầng như Amazon EC2, và codename của dự án là Nebula. Với sự tác động từ nhiều phía khác nhau, cuối cùng vào năm 2010 NASA quyết định công bố mã nguồn của Nebula và phát triển nó dưới dạng nguồn mở với codename là Nova. Sau đó Rackspace tiếp tục đóng góp nền tảng lưu trữ của họ vào dự án với codename Swift. Dự án Openstack được thành lập với cam kết phát triển theo hướng mở. Nó nhanh chóng nhận được sự đồng thuận từ rất nhiều hãng công nghệ khác và cộng đồng. Hiện nay đã có hơn 160 công ty tham gia vào dự án này với hầu hết các tên tuổi lớn như: NASA, Rackspace, Cisco, Citrix, Microsoft, HP, Dell, Canonical...

Như đã nói AWS chính là nguồn cảm hứng tạo nên Openstack ngày nay, AWS là nền tảng đóng của Amazon và Openstack là một nền tảng mở dành cho tất cả các công ty và đồng sử dụng. Mục đích của Openstack là cung cấp cho người dùng khả năng xây dựng một hạ tầng cho cả private cloud và public cloud. Đã có nhiều công ty sử dụng Openstack để xây dựng dịch vụ để phục vụ nhu cầu của chính họ và cho thuê như chính NASA và Rackspace

### **3.3. Tổng quan về Openstack**

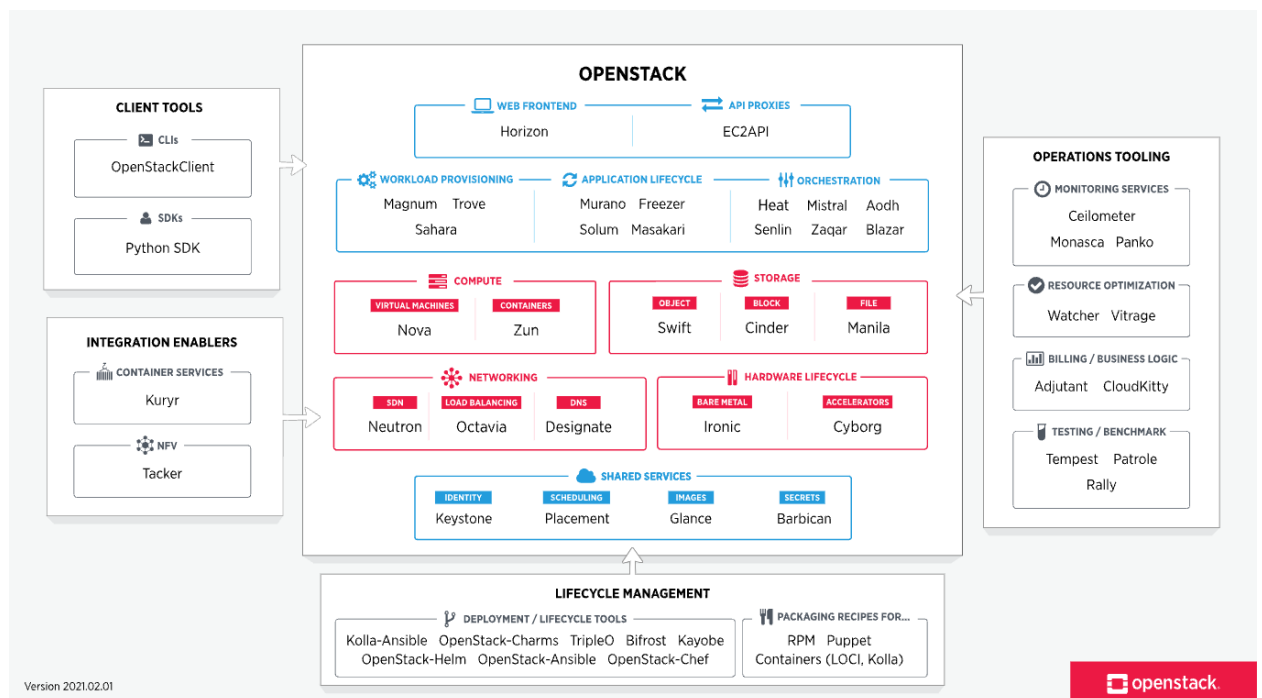
Openstack có chu kỳ phát triển 6 tháng, đi cùng với sự phát triển của CC, với mỗi phiên bản Openstack lại bổ sung thêm thành phần mới tương ứng với những chức năng mới. Openstack hoàn toàn là nguồn mở, các thành phần của nó được viết trên Python - ngôn ngữ đang được đánh giá rất cao những năm gần đây. Openstack được phát triển theo mô hình Infrastructure as a Service (IaaS) quản lý tài nguyên hệ thống máy tính và cung cấp tài nguyên (các server ảo và các tài nguyên khác) cho người dùng. Nền tảng phần mềm bao gồm một nhóm các chức năng liên quan đến nhau điều khiển xử lý các nhóm phần cứng, lưu trữ và hệ thống

mạng trong data center. Người sử dụng quản lý thông qua một dashboard dựa trên nền web, thông các công cụ dòng lệnh hoặc thông qua các API RESTful.

Openstack.org là đơn vị phát hành Openstack dựa theo các điều khoản của Giấy phép Apache. OpenStack là một dự án chung của Rackspace Hosting và của NASA vào năm 2010. Kể từ năm 2016, OpenStack được quản lý bởi OpenStack Foundation (một tổ chức phi lợi nhuận) được thành lập vào tháng 9 năm 2012 để quảng bá phần mềm OpenStack và cộng đồng người dùng và hơn 500 công ty đã tham gia dự án.

### 3.4. Các thành phần của Openstack

Các thành phần chính của openstack bao gồm openstack service, Operations tooling, Add-Ons to Services và Integration enablers.



Hình 9: Kiến trúc tổng thể các dịch vụ openstack

STT	Chức năng, tên dịch vụ	Phân loại dịch vụ
1	Compute(Nova)	Compute
2	Containers(ZUN)	Compute
3	Bare Metal Provisioning(Ironic)	Hardware Lifecycle
4	Lifecycle management of accelerators(Cyborg)	Hardware Lifecycle
5	Object store(Swift)	Storage

6	Block storage(Cinder)	Storage
7	Shared filesystems(Manila)	Storage
8	Networking(Neutron)	Networking
9	Load balancer(Octavia)	Networking
10	DNS(Designate)	Networking
11	Identity(Keystone)	Shared Services
12	Placement(Placement)	Shared Services
13	Image(Glance)	Shared Services
14	Key management(Barbican)	Shared Services
15	Orchestration(Heat)	Orchestration
16	Clustering(Senlin)	Orchestration
17	Workflow(Mistral)	Orchestration
18	Messaging(Zaqar)	Orchestration
19	Resource reservation(Blazar)	Orchestration
20	Alarming(Aodh)	Orchestration
21	Container Orchestration Engine(Magnum)	Workload Provisioning
22	Big Data Processing Framework(Sahara)	Workload Provisioning
23	Database as a Service(Trove)	Workload Provisioning
24	Instances High Availability(Masakari)	Application Lifecycle
25	Application Catalog(Murano)	Application Lifecycle
26	Software Develop Lifecycle Automation(Solum)	Application Lifecycle
27	Backup, Restore, and Disaster Recovery(Freezer)	Application Lifecycle
28	EC2 API proxy(EC2API)	API Proxies
29	Dashboard(Horizon)	Web Frontend

*Bảng 1: Danh sách các openstack service*

STT	Chức năng, tên tool	Phân loại tool
1	Metering & Data Collection(Ceilometer)	Monitoring services
2	Event, Metadata Indexing(Panko)	Monitoring services
3	Monitoring(Monasca)	Monitoring services
4	Optimization(Watcher)	Resource optimization
5	Root Cause Analysis(Vitrage)	Resource optimization
6	Operations processes automation(Adjutant)	Billing / Business Logic
7	Billing and chargebacks(Cloudkitty)	Billing / Business Logic
8	Benchmarking tool(Rally)	Testing / Benchmark
9	OpenStack Integration Test Suite(Tempest)	Testing / Benchmark

10	OpenStack RBAC Integration Test Suite(Patrole)	Testing / Benchmark
----	--	---------------------

*Bảng 2: Danh sách các openstack operations tooling*

STT	Chức năng, tên Add-Ons	Phân loại Add-Ons
1	Computable object storage(Storlets)	Swift add-ons

*Bảng 3: Danh sách các openstack add-ons to services*

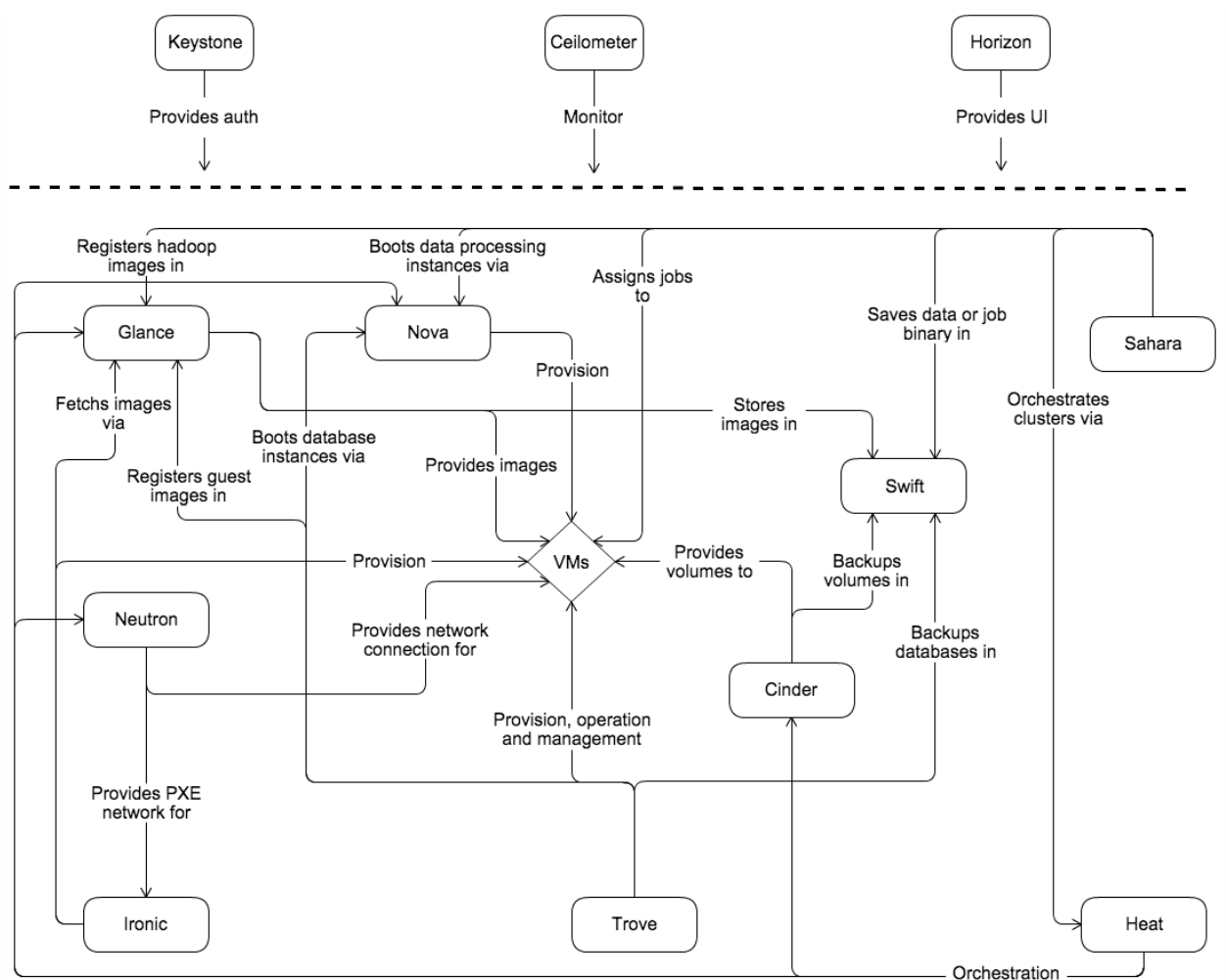
STT	Chức năng, tên Integration enablers	Phân loại Add-Ons
1	Networking integration for containers(Kuryr)	Containers
2	NFV Orchestration(Tacker)	NFV

*Bảng 4: Danh sách các openstack add-ons to services*

### 3.5. Kiến trúc Conceptual

Kiến trúc Conceptual thể hiện mối quan hệ giữa các dịch vụ trong OpenStack

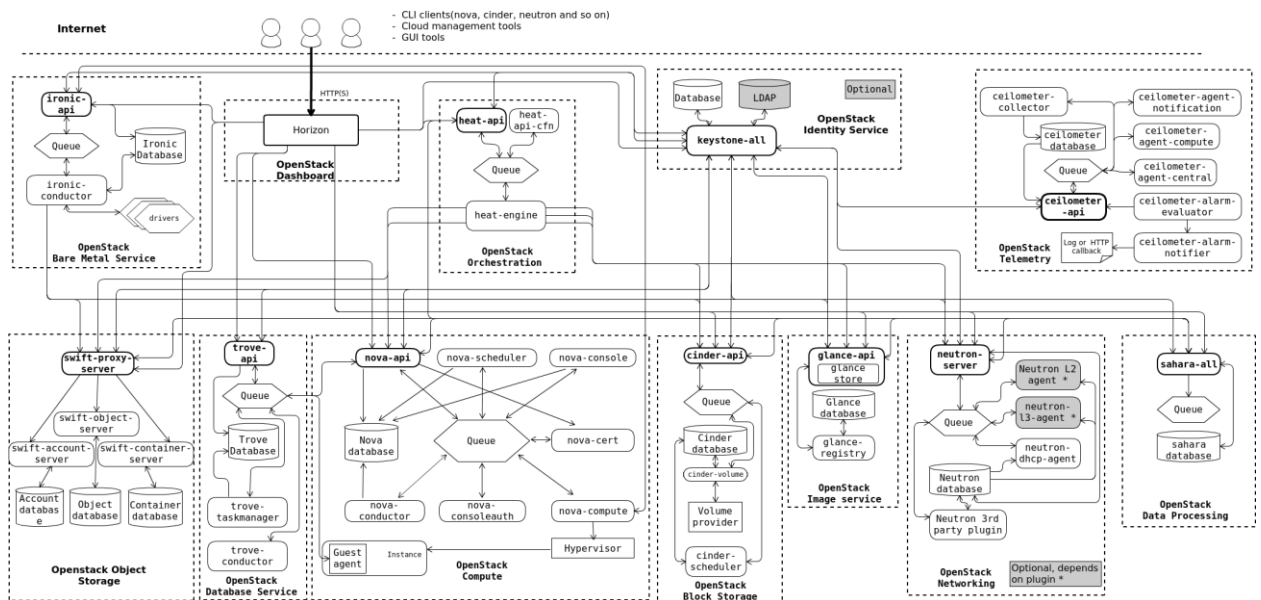




Hình 10: kiến trúc conceptual

### 3.6. Kiến trúc Logic

Để hiểu, triển khai, cấu hình openstack thì người quản trị cần phải hiểu được kiến trúc logic



Hình 11: kiến trúc logic phổ biến

Như được trong kiến trúc conceptual, OpenStack bao gồm một số phần độc lập, được đặt tên là các OpenStack service. Tất cả các dịch vụ đều được xác thực thông qua một Identity service chung. Các dịch vụ riêng lẻ tương tác với nhau thông qua các API công khai, trừ khi các lệnh của quản trị viên đặc quyền là cần thiết.

Bên trong, các OpenStack service bao gồm một số process. Tất cả các service đều có ít nhất một API process, quy trình này sẽ lắng nghe các yêu cầu API, xử lý trước chúng và chuyển chúng đến các phần khác của service.

Để giao tiếp giữa các process của một service, một tin nhắn AMQP được sử dụng. Trạng thái của service được lưu trữ trong cơ sở dữ liệu. Khi triển khai và định cấu hình đám mây OpenStack, bạn có thể chọn trong số một số giải pháp cơ sở dữ liệu và message broker, chẳng hạn như RabbitMQ, MySQL, MariaDB và SQLite.

Người dùng có thể truy cập OpenStack thông qua giao diện người dùng dựa trên web thông qua Horizon Dashboard, thông qua các command-line clients và bằng cách đưa ra các yêu cầu API thông qua các công cụ như plug-ins của browser hoặc curl. Đối với các ứng dụng, một số SDK có sẵn. Cuối cùng, tất cả các phương pháp truy cập này đưa ra các lệnh gọi REST API tới các dịch vụ OpenStack khác nhau.

### 3.7. Kiến trúc logic một số thành phần cơ bản

Do được tạo nên từ các service độc và là mã nguồn mở nên ai cũng có thể thêm các thành phần bổ sung vào openstack để đáp ứng nhu cầu của họ. Tuy nhiên

thì openstack có một số service chính mà hầu như hệ thống nào triển khai openstack đều có

Các thành phần đó là:

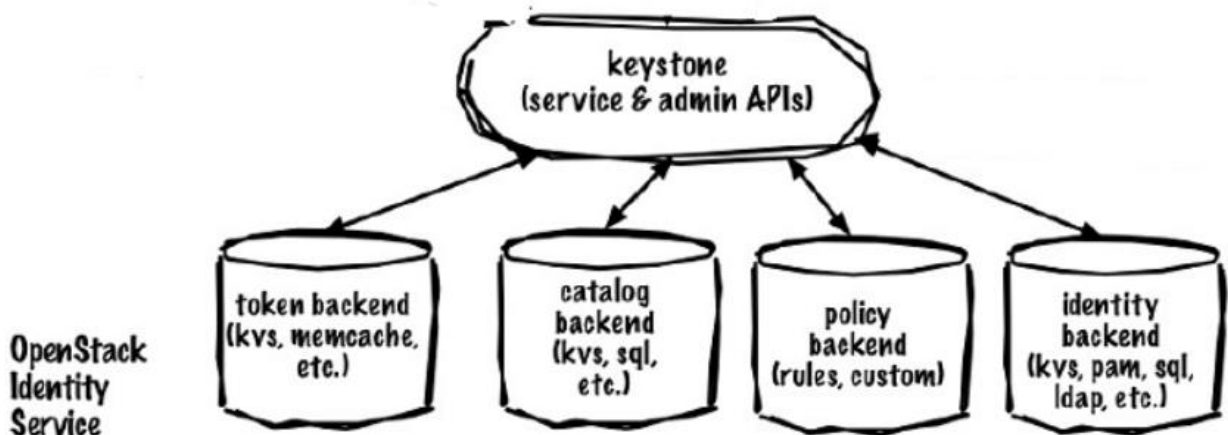
- Identity( Keystone)
- Compute( Nova)
- Networking( Neutron)
- Block storage( Cinder)
- Image (Glance)
- Dashboard( Horizon)

### 3.7.1. Identity(Keystone)

Tất cả các service trong Openstack đều được xác thực thông qua Identity service, do đó identity là thành phần gần như không thể thiếu trong bất kỳ hệ thống nào triển khai Openstack

*\*) Các tính năng chính của Keystone:*

- Quản lý user và những quyền mà user được phép làm.
- Cung cấp các dịch vụ nhận dạng và xác thực.
- Tạo các chính sách giữa user và dịch vụ.



Hình 12: Các thành phần trong keystone

*\*) Các tính thành phần của Keystone:*

- **Identity:** cung cấp xác thực và dữ liệu về người dùng và nhóm.
- **Token:** xác thực và quản lý các token để xác thực yêu cầu khi thông tin đăng nhập của người dùng đã được xác minh.
- **Catalog:** cho phép các API client tự động phát hiện và điều hướng đến các dịch vụ đám mây.
- **Policy:** Cung cấp các chính sách của Keystone. Mỗi dịch vụ OpenStack định nghĩa các chính sách truy cập cho các tài nguyên của nó trong một tệp policy liên quan.

### **3.7.2. Compute(Nova)**

Đây là phần cơ bản nhất của Openstack có chức năng điều khiển IaaS và phân phối lại tài nguyên hệ thống cho các instance với khả năng tính toán lưu trữ độc lập. Về cơ bản Nova cung cấp cho người dùng khả năng chạy các instance (máy ảo) và giao diện để quản lý các instance đó trên hạ tầng phần cứng. Tuy nhiên Nova không bao gồm bất cứ phần mềm ảo hóa nào. Cái nó làm là sử dụng lại các hypervisor (do người dùng tùy chọn cài đặt) để thực hiện việc ảo hóa tính toán. Người dùng có thể sử dụng các hypervisor khác nhau trong các zone khác nhau. Dưới đây là các hypervisor( Phần mềm giám sát máy ảo) mà Nova hiện hỗ trợ:

- Hyper-V 2008
- KVM - Kernel-based Virtual Machine
- LXC - Linux Containers (through libvirt)
- QEMU - Quick EMUlator
- UML - User Mode Linux
- VMWare ESX/ESXi 4.1 update 1
- Xen - XenServer 5.5, Xen Cloud Platform (XCP)

*\*) Các tính năng chính của Nova:*

- **Quản lý tài nguyên ảo hóa** bao gồm CPU, memory, disks, network interfaces  
Tất cả các tài nguyên được hợp nhất vào trong 1 “bể” – “pool of computing”.

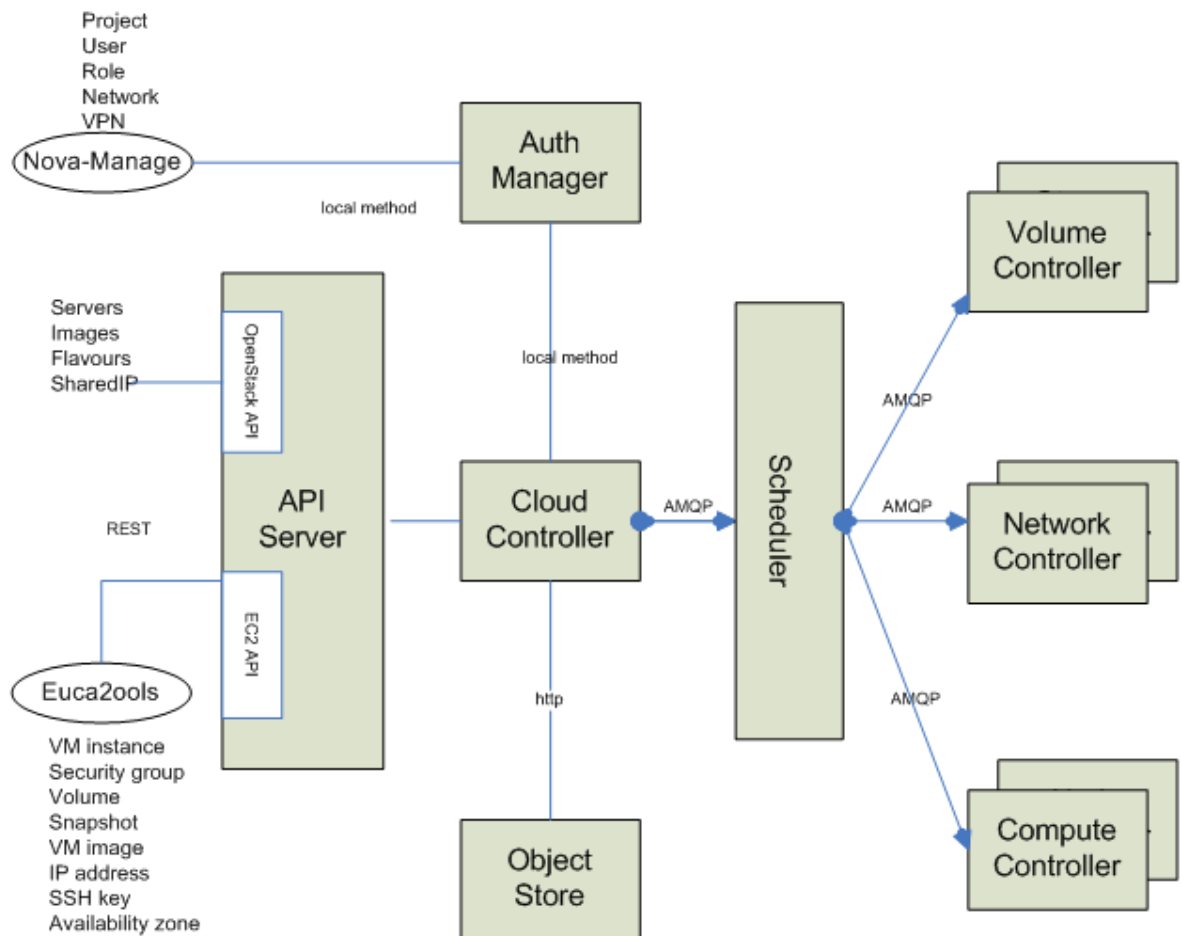
Việc này sẽ tăng tính tự động và tận dụng tài nguyên, đem lại lợi ích lớn về kinh tế.

- **Quản lý mạng nội bộ (LAN)** Flat, Flat DHCP, VLAN DHCP, IPv6, OpenStack được lập trình để chỉ định các địa chỉ IPs và VLAN (Virtual LAN). Chức năng này giúp cho việc cung cấp dịch vụ networking và nâng tính bảo mật khi các VLANs được tách rời nhau. Đồng thời tính linh hoạt trong mô hình mạng cũng phù hợp với mỗi ứng dụng cho mỗi user/group.
- **API với nhiều tính năng và xác thực:** Được thiết kế tự động và an toàn để quản lý việc users truy cập vào các tài nguyên và ngăn chặn truy cập trái phép qua lại giữa các users.
- **Live VM management (Instance)** khởi tạo, khởi động, đóng băng, hay xóa instances. Ngoài ra còn có tính năng lifecycle management.
- **Floating IP addresses:** Một ip có thể được gán lại ngay lập tức từ node này sang node khác nhằm giảm downtime hệ thống
- **Security Groups:** Như một tường lửa ảo cho instance để kiểm soát lưu lượng vào và ra
- **Role Based Access Control (RBAC):** Có thể dùng hạn chế quyền truy cập mạng dựa trên vai trò của một người trong tổ chức
- **Advanced Scheduler (Diablo v3 07/28 – Started)**

*\*) Nova có 7 thành phần chính:*

- **Cloud Controller** - quản lý và tương tác với tất cả các thành phần của Nova
- **API Server** - giống như một Web service đầu cuối của Cloud Controller
- **Compute Controller** - cung cấp, quản lý tài nguyên từ các instance. Object Store - cung cấp khả năng lưu trữ, thành phần này đi cùng với Compute Controller
- **Auth Manager** - dịch vụ authentication và authorization
- **Volume Controller** - lưu trữ theo block-level - giống như Amazon EBS
- **Network Controller** - tạo quản lý các kết nối trong virtual network để các server có thể tương tác với nhau và với public network

- **Security Groups:** Như một tường lửa ảo cho instance để kiểm soát lưu lượng vào và ra
- **Scheduler** - chọn ra compute controller thích hợp nhất để lưu instance



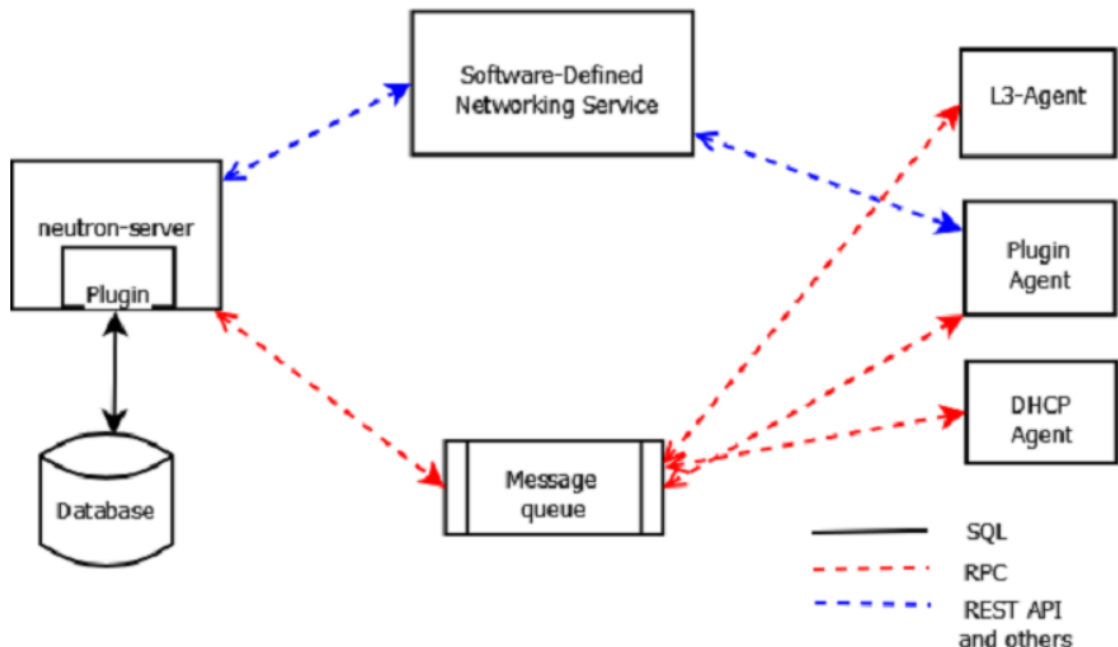
Hình 13: Minh họa về kiến trúc nova trong openstack

Các thành phần của Nova hoạt động độc lập, kết nối với nhau bằng các thông điệp (message-based architecture). Các thành phần Compute Controller, Volume Controller, Network Controller và Object Store có thể cài đặt trên các server vật lý khác nhau. Như trong hình trên có thể thấy Cloud Controller giao tiếp với Object Store thông qua HTTP nhưng giao tiếp với Scheduler thông qua AMQP (Advanced Message Queue Protocol) Để tránh việc tắc nghẽn khi chờ đợi các thành phần phản hồi, Nova sử dụng các hàm gọi không đồng bộ (asynchronous), với một call-back được gửi khi mà response đã được nhận

### 3.7.3. Networking(Neutron)

OpenStack Networking cho phép bạn tạo và quản lý các network objects ví dụ như networks, subnets, và ports cho các services khác của OpenStack sử dụng. Với kiến trúc pluggable, các plug-in có thể được sử dụng để triển khai các thiết bị và phần mềm khác nhau, nó khiến OpenStack có tính linh hoạt trong kiến trúc và triển khai.

Dịch vụ Networking trong OpenStack (neutron) cung cấp API cho phép bạn định nghĩa các kết nối mạng và gán địa chỉ ở trong môi trường cloud. Nó cũng cho phép các nhà khai thác vận hành các công nghệ networking khác nhau cho phù hợp với mô hình điện toán đám mây của riêng họ. Neutron cũng cung cấp một API cho việc cấu hình cũng như quản lý các dịch vụ networking khác nhau từ L3 forwarding, NAT cho tới load balancing, perimeter firewalls, và virtual private networks.



Hình 14: Các thành phần trong neutron

\*) Các tính năng chính của Neutron:

- Quản lý mạng và địa chỉ IP, đảm bảo mạng không bị hiện tượng nút cổ chai và đem đến cho người dùng khả năng tự phục vụ.
- Cung cấp một framework mở rộng có thể triển khai và quản lý các dịch vụ mạng bổ sung, chẳng hạn như hệ thống phát hiện xâm nhập (IDS), cân bằng tải, tường lửa và mạng riêng ảo (VPN)...

*\*) Các thành phần chính của Neutron:*

- Neutron server (neutron-server và neutron-\*-plugin): chạy trên các node mạng để phục vụ Networking API và các mở rộng của nó. Ngoài ra nó cũng thực thi các dịch vụ mạng và đặt địa chỉ IP cho mỗi port.
- Plugin agent (neutron-\*-agent): Chạy trên mỗi compute node để quản lý cấu hình chuyển mạch ảo cục bộ (vswitch).
- DHCP agent (neutron-dhcp-agent): Cung cấp dịch vụ DHCP cho các tenant network. Tác nhân này giống nhau trên tất cả các plugin và chịu trách nhiệm duy trì cấu hình DHCP.
- L3 agent (neutron-l3-agent): Cung cấp chuyển tiếp L3 / NAT cho truy cập mạng bên ngoài của các máy ảo trên các mạng thuê.
- Network provider services (SDN server/services): Cung cấp các dịch vụ mạng bổ sung cho các mạng thuê.

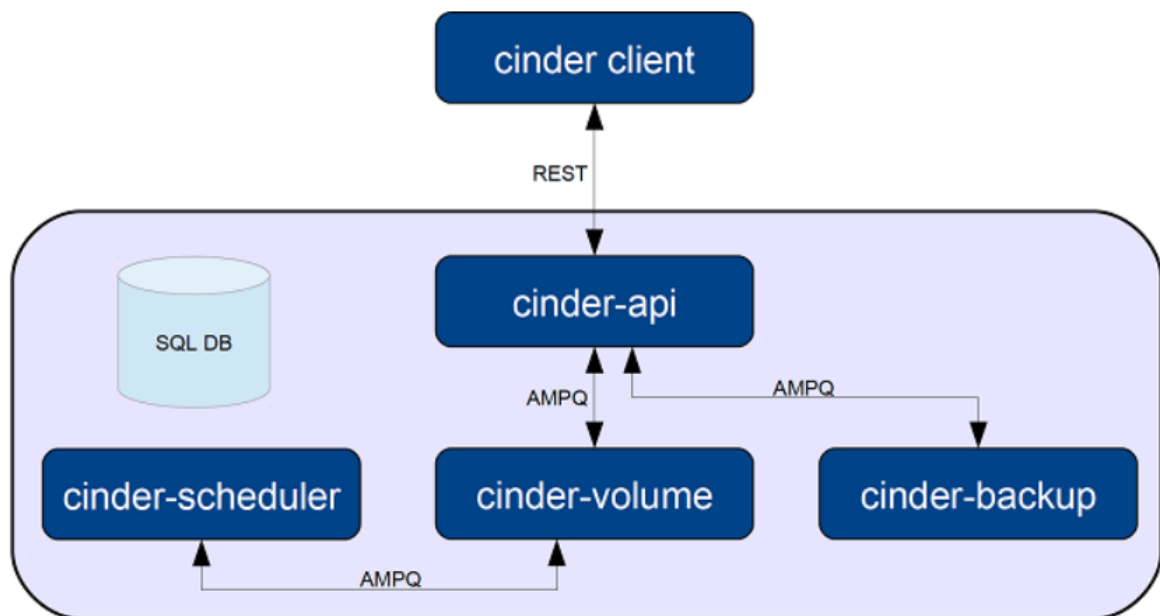
#### **3.7.4. Block Storage( Cinder)**

Cinder là dịch vụ Block Storage trong Openstack. Nó được thiết kế để người dùng cuối có thể thực hiện việc lưu trữ bởi Nova, việc này được thực hiện bởi LVM hoặc các plugin driver cho các nền tảng lưu trữ khác. Cinder ảo hóa việc quản lý các thiết bị block storage và cung cấp cho người dùng cuối một API đáp ứng được nhu cầu tự phục vụ cũng như tiêu thụ các tài nguyên đó mà không cần biết quá nhiều kiến thức chuyên sâu.

*\*) Các tính năng chính của Cinder:*



- Cung cấp các thiết bị lưu trữ khối (Block Storage) để sử dụng cùng với khối Compute, dùng lưu trữ cơ sở dữ liệu, hệ thống tệp có thể mở rộng...
- Ảo hóa việc quản lý các thiết bị lưu trữ khối.
- Quản lý việc tạo, gắn và tách các thiết bị khối với các máy chủ. Block Storage Volume được tích hợp đầy đủ vào Compute và Dashboard cho phép người dùng đám mây quản lý nhu cầu lưu trữ của riêng họ.



Hình 15: Các thành phần trong cinder

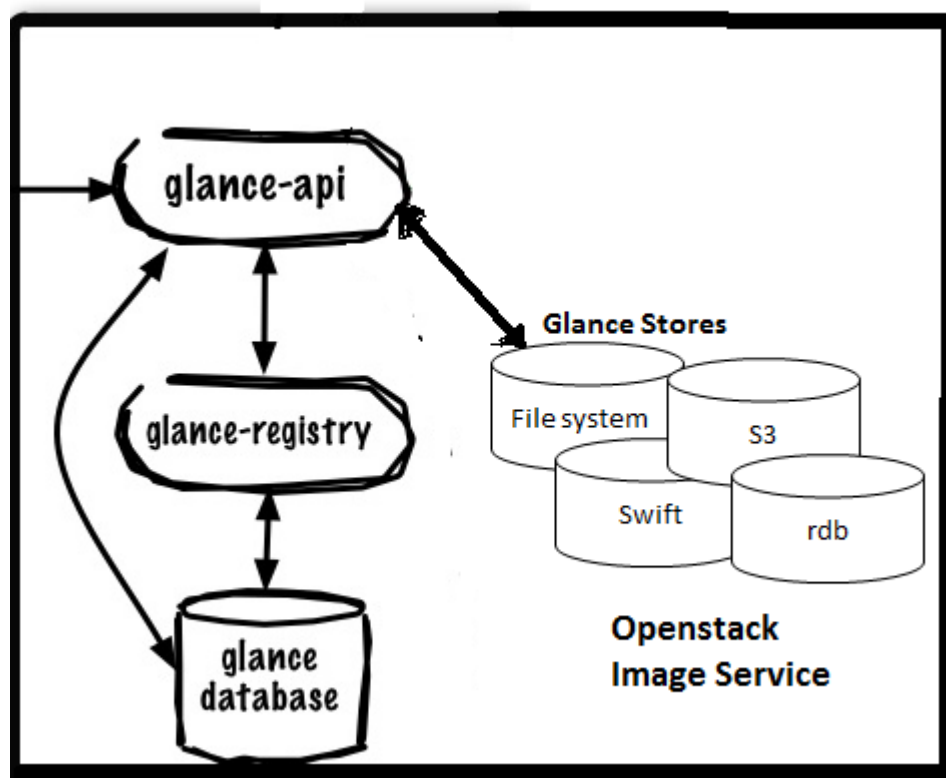
*\*) Các thành phần chính của Cinder:*

- **Cinder-api:** Nhận API request và định tuyến chúng đến cinder-volume để thực thi.
- **Cinder-volume:** Tương tác trực tiếp với dịch vụ Lưu trữ khối và các tiến trình như cinder-scheduler.
- **Cinder-scheduler:** Dựa trên request được điều hướng tới, cinder-scheduler chuyển request tới Cinder Volume Service thông qua giao thức AMQP (Advanced Message Queue Protocol). Cinder-scheduler chọn nơi cung cấp bộ nhớ tối ưu để tạo volume. Nó tương tự như nova-scheduler.

- **Cinder-backup**: cung cấp khả năng sao lưu cho bất kỳ loại nào cho backup storage provider.
- **Messaging queue**: chuyển tiếp thông tin giữa các tiến trình trong block storage.

### 3.7.5. Image Service( Glance)

OpenStack Image Service( Glance) cung cấp các tính năng về discovery, đăng ký (registration), và vận chuyển (delivery) các dịch vụ cho các đĩa images ảo. API của OpenStack Image Service cung cấp một giao diện tiêu chuẩn cho các thông tin truy vấn về các đĩa image ảo lưu trữ trong các back-end, bao gồm luôn cả OpenStack Object Storage. Clients có thể đăng ký một đĩa image ảo với các dịch vụ có sẵn, thực hiện việc truy vấn thông tin



Hình 16: Các thành phần trong glance

\*) Một số tính năng hiện tại của Glance:

- Cho phép người dùng khám phá, đăng ký và truy xuất các VM image.
- Cung cấp API REST cho phép ta truy vấn VM image metadata và truy xuất một image thực tế

*\*) Các thành phần chính của Glance:*

- **Glance-api:** Nhận REST call cho việc tìm kiếm, tiếp nhận và lưu trữ các image.
- **Glance-registry:** lưu trữ, thực thi và thu thập metadata về các image.
- **Database:** là cơ sở dữ liệu lưu trữ image metadata.
- **Storage repository:** tích hợp với các thành phần OpenStack khác nhau bên ngoài như hệ thống tệp thông thường, Amazon S3 và HTTP cho kho lưu trữ hình ảnh...

### 3.7.6. Dashboard( Horizon)

*\*) Một số tính năng chính của dashboard:*

- Cung cấp cho quản trị viên và người dùng một giao diện đồ họa dựa trên web để truy cập, cung cấp và tự động triển khai các tài nguyên dựa trên đám mây.

## CHƯƠNG 4: TRIỂN KHAI OPENSTACK

### 4.1. Yêu cầu hệ thống

Để triển khai được một mô hình openstack với 2 region mỗi region có 1 node all in one tối thiểu chúng ta cần 2 máy ảo mỗi máy có số yêu cầu cấu hình cơ bản sau:

- Ít nhất 2 network interface
- Máy có ít nhất 6GB ram
- Ổ cứng tối thiểu 40G

Sau khi có đủ hệ thống tối thiểu nhóm đã cấu hình cho 2 máy ảo một số thông số như sau:

<b>Node1</b>	
Interfaces	Enp0s8: 10.10.10.11 Enp0s3: 10.0.2.15 Enp0s9: Chưa gán ip
Ram	8G
vCPUs	3
Storage	/dev/mapper/controller--vg-root: 66,5GB
Virtualization Platform	KVM
Operating System	Ubuntu 18.04 LTS
User	stack (non root user with sudo rights )

*Bảng 5: Các thông tin cấu hình cơ bản của node1*

<b>Node2</b>	
Interfaces	Enp0s8: 10.10.10.111 Enp0s3: 10.0.2.15 Enp0s9: Chưa gán ip
Ram	6G
vCPUs	1
Storage	/dev/mapper/controller--vg-root: 66,5GB
Virtualization Platform	KVM
Operating System	Ubuntu 18.04 LTS
User	stack (non root user with sudo rights )

*Bảng 6: Các thông tin cơ bản của node2*

## 4.2. Các bước triển khai

### 4.2.1. Cập nhật và nâng cấp các package ubuntu

*\$ sudo apt update*

*\$ sudo apt upgrade*

### 4.2.2. Cài đặt python3 và các package để có thể tạo môi trường deploy bằng python

- Cài đặt các package cần thiết:

*\$sudo apt install python3-dev python3-venv libffi-dev gcc libssl-dev git*

-Tạo môi trường ảo cho kola-ansible( một công cụ deploy của cộng đồng openstack):

```
$ python3 -m venv $HOME/kolla-openstack
```

-Active môi trường vừa tạo:

```
$ source $HOME/kolla-openstack/bin/activate
```

-Cập nhật công cụ quản lý package của python

```
(kolla-openstack) pip install -U pip
```

#### **4.2.3. Cài đặt ansible ubuntu**

-Cài đặt ansible sử dụng pip của python:

```
(kolla-openstack) pip install 'ansible<2.10'
```

-Tạo file cấu hình ansible:

```
(kolla-openstack) vim $HOME/ansible.cfg
```

-File cấu hình có nội dung:

```
[defaults]  
host_key_checking=False  
pipelining=True  
forks=100
```

#### **4.2.4. Cài đặt và cấu hình kola-ansible cho ubuntu**

- Cài đặt bản kola-ansible 10.0.0 (bản ansible phát hành đầu tiên cho openstack ussuri)

```
(kolla-openstack) pip install kolla-ansible==10.0.0
```

- Tạo direactory /etc/kola để chứa các file cấu hình kola-ansible

```
(kolla-openstack) sudo mkdir /etc/kolla
```

- Cấp quyền truy cập cho user vừa dùng để tạo môi trường ảo:

```
(kolla-openstack) sudo chown $USER:$USER /etc/kolla
```

- Sao chép file cài đặt(globals.yml) và file chứa mật khẩu(passwords.yml) vào directory cài đặt kolla ở trên

```
(kolla-openstack)\  
cp $HOME/kolla-openstack/share/kollaansible/etc_examples/kolla/*\ /etc/kolla/
```

- Sao chép file inventory all in one(file chứa danh sách các server cần deploy) ra thư mục hiện tại:

```
(kolla-openstack)\  
cp $HOME/kolla-openstack/share/kolla-ansible/ansible/inventory/all-in-one .
```

- Thiết lập một số thông tin cài đặt đặc biệt trong file /etc/kolla/globals.yml:

```
config_strategy: "COPY_ALWAYS"  
kolla_base_distro: "ubuntu"  
kolla_install_type: "binary"  
openstack_release: "ussuri"  
kolla_internal_vip_address: "10.10.10.11"  
kolla_internal_fqdn: "{{ kolla-openstack.kifarunix-demo.com }}"  
kolla_external_vip_address: "{{ kolla_internal_vip_address }}"  
kolla_external_fqdn: "{{ kolla_internal_fqdn }}"  
network_interface: "enp0s8"  
neutron_external_interface: "enp0s9"  
neutron_plugin_agent: "openvswitch"  
enable_haproxy: "yes"  
enable_cinder: "yes"  
enable_cinder_backend_lvm: "yes"  
keystone_token_provider: 'fernet'  
cinder_volume_group: "controller-vg"  
nova_compute_virt_type: "qemu"
```

- Cập nhật lại các thay đổi vào biến môi trường:

```
(kolla-openstack) source $HOME/kolla-openstack/bin/activate
```

- Tạo các mật khẩu quản trị, các mật khẩu này sẽ được lưu vào file `/etc/kolla/passwords.yml`:

*(kolla-openstack) kolla-genpwd*

#### **4.2.5. Deploy region 1**

- Khởi động cấu hình trước khi deploy:

*(kolla-openstack) kolla-ansible -i all-in-one bootstrap-servers*

- Kiểm tra các cấu hình hệ thống xem đủ các điều kiện để deploy chưa:

*(kolla-openstack) kolla-ansible -i all-in-one prechecks*

- Nếu như tất cả các điều kiện đều được thông qua thì bắt đầu deploy:

*(kolla-openstack) kolla-ansible -i all-in-one deploy*

- Sau khi deploy có thể kiểm tra trạng thái hoạt động của các service bằng lệnh:

*(kolla-openstack) docker ps*

- Cài đặt command line administration tools của openstack:

*(kolla-openstack) pip install python-openstackclient python-neutronclient python-glanceclient*

- Tạo OpenStack admin user credentials:

*(kolla-openstack) kolla-ansible post-deploy*

*(kolla-openstack) source /etc/kolla/admin-openrc.sh*

- Tạo OpenStack network, images, nova keys:

*(kolla-openstack) kolla-openstack/share/kolla-ansible/init-runonce*

#### **4.2.6. Deploy region 2**

- Trên node1 cấu hình enable keystone và horizon trong `/etc/kolla/globals.yml`:

```
enable_keystone: "yes"
enable_horizon: "yes"
```

- Khai báo thêm region thứ hai có tên RegionTwo:

```
openstack_region_name: "RegionOne"
multiple_regions_names:
  - "{{ openstack_region_name }}"
  - "RegionTwo"
```

- Deploy lại openstack sau khi config thêm RegionTwo:

```
kolla-ansible -i all-in-one reconfigure
```

- Trên node2 chúng ta tiến hành lại như các bước deploy trên node1 nhưng khi cấu hình file /etc/kolla/globals.yml chúng ta phải khai báo để kết nối được sang node1:

```
kolla_internal_fqdn_r1: 10.10.10.254

keystone_admin_url: "{{ admin_protocol }}://{{ kolla_internal_fqdn_r1 }}:{{ keystone_admin_port }}"
keystone_internal_url: "{{ internal_protocol }}://{{ kolla_internal_fqdn_r1 }}:{{ keystone_public_port }}"

openstack_auth:
  auth_url: "{{ admin_protocol }}://{{ kolla_internal_fqdn_r1 }}:{{ keystone_admin_port }}"
  username: "admin"
  password: "{{ keystone_admin_password }}"
  project_name: "admin"
  domain_name: "default"
```

- Khai báo đường dẫn ghi đè config:

```
node_custom_config: "/etc/kolla/config"
```

- Tạo file global.conf trong folder /etc/kolla/config với nội dung:



```
[keystone_authtoken]
www_authenticate_uri = {{ keystone_internal_url }}
auth_url = {{ keystone_admin_url }}
```

- Tạo file nova.conf trong folder /etc/kolla/config/nova với nội dung:

```
[placement]
auth_url = {{ keystone_admin_url }}
```

- Tạo file heat.conf trong folder /etc/kolla/config/heat với nội dung:

```
[trustee]
www_authenticate_uri = {{ keystone_internal_url }}
auth_url = {{ keystone_internal_url }}
```

```
[ec2authtoken]
www_authenticate_uri = {{ keystone_internal_url }}
```

```
[clients_keystone]
www_authenticate_uri = {{ keystone_internal_url }}
```

- Tạo file ceilometer.conf trong folder /etc/kolla/config/ceilometer với nội dung:

```
[service_credentials]
auth_url = {{ keystone_internal_url }}
```

- Thay đổi tên region trong /etc/kolla/globals.yml của node2:


```
openstack_region_name: "RegionTwo"
```

- Khai báo cho node 2 không chạy service keystone và horizon:

```
enable_keystone: "no"
enable_horizon: "no"
```

## CHƯƠNG 5: KẾT QUẢ TRIỂN KHAI

*Dưới đây là một số hình ảnh sau khi triển khai openstack sử dụng docker*



openstack.

Log in

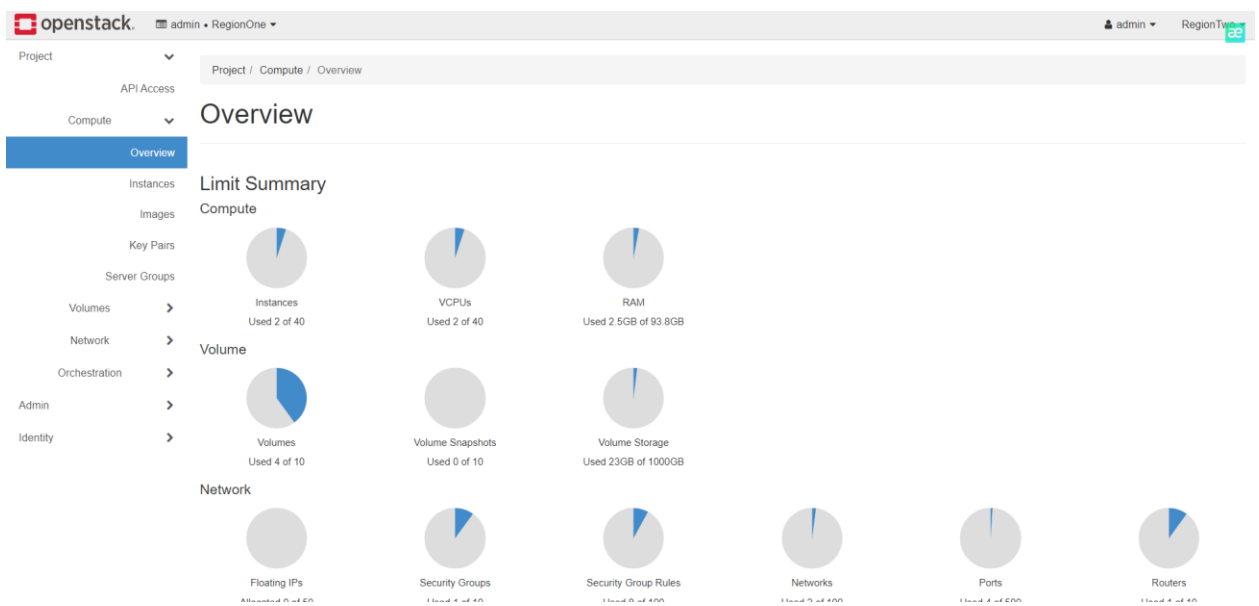
User Name

Password

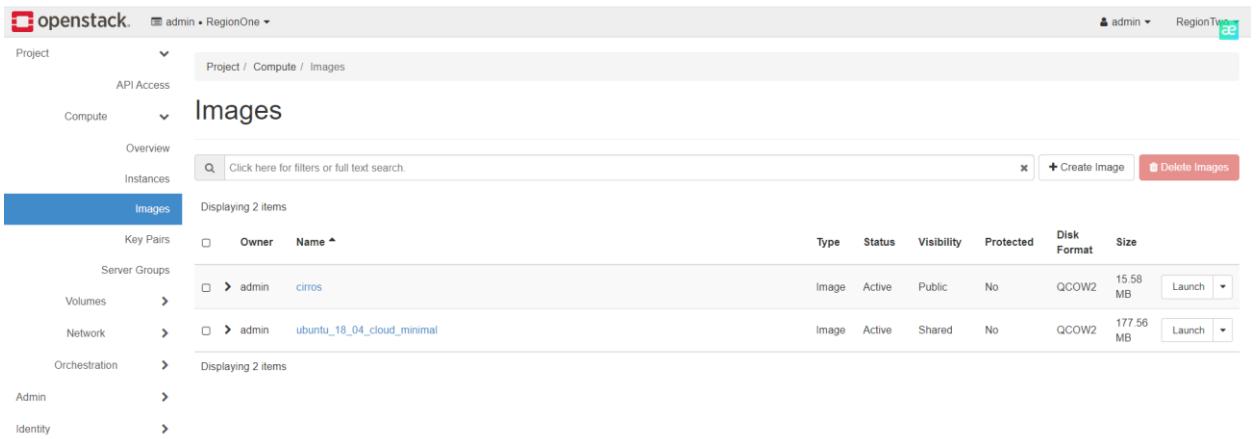
Region

[Sign In](#)

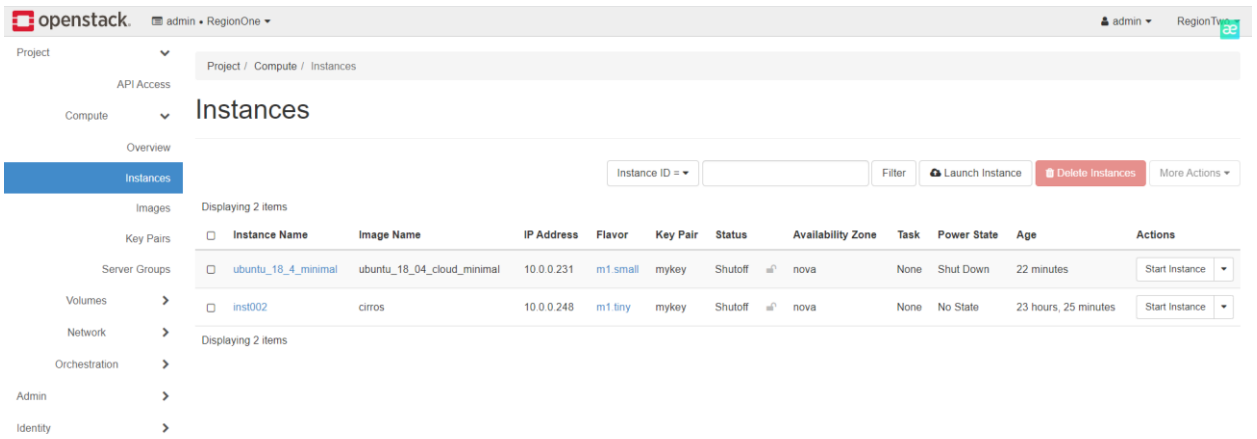
Hình 17: Trang login openstack



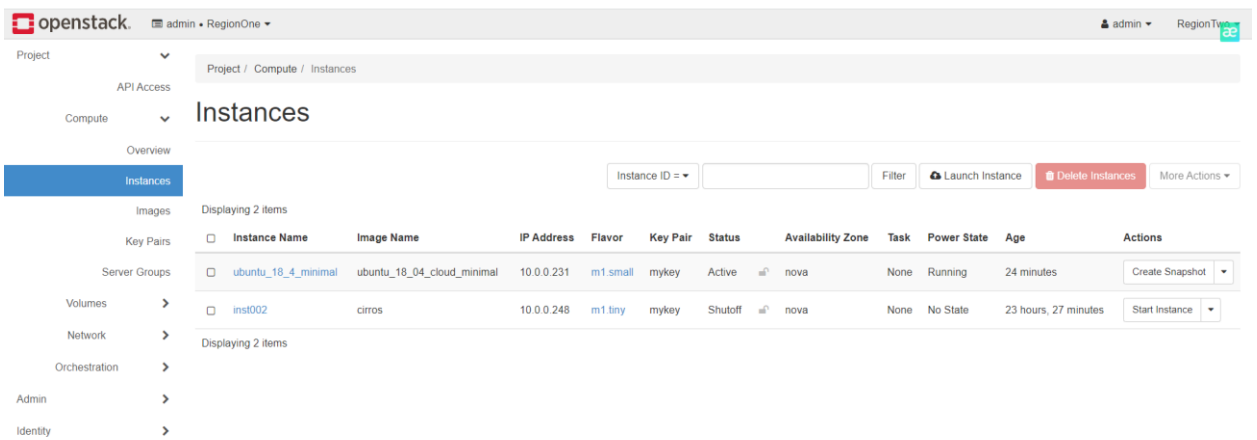
Hình 18: Trang overview



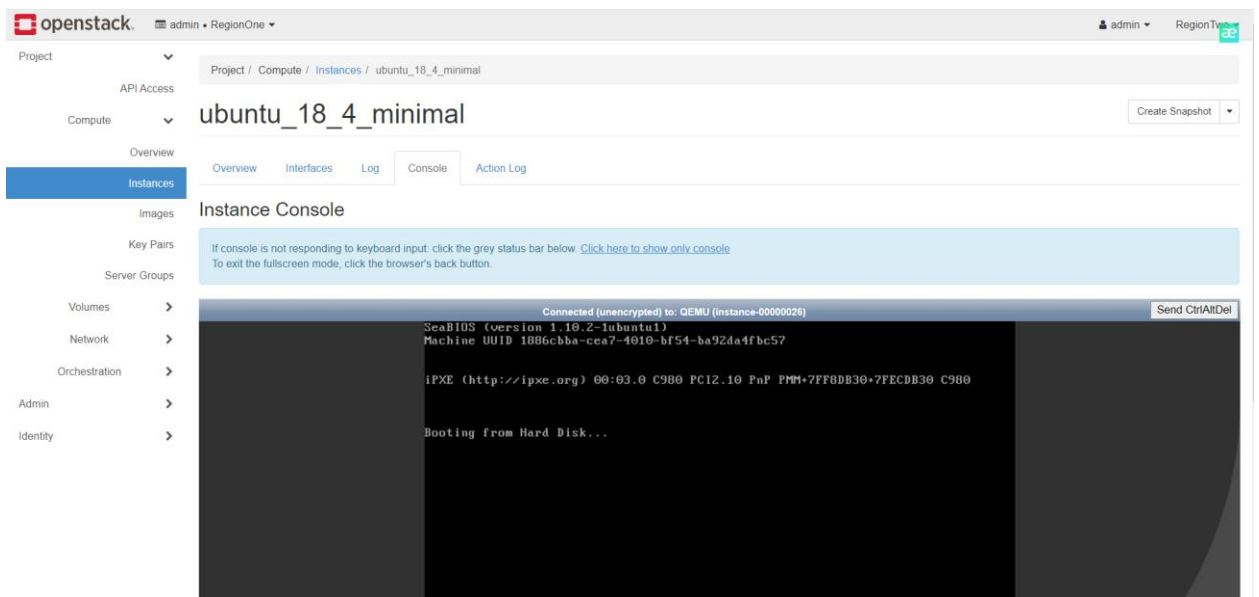
Hình 19: Danh sách các Image



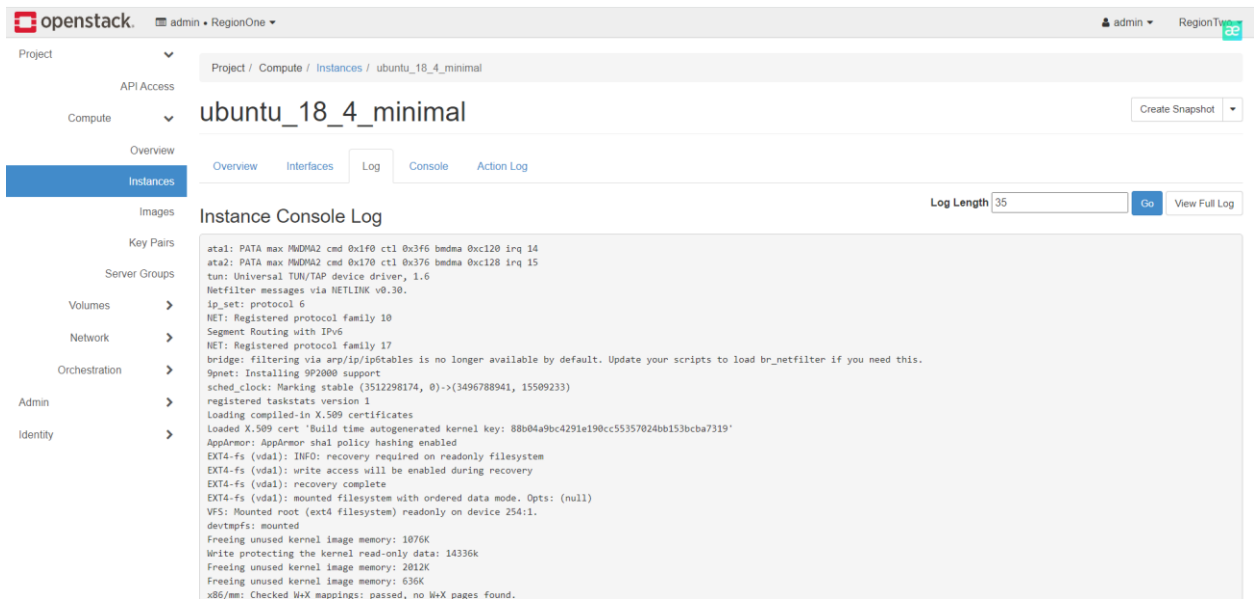
Hình 20: Danh sách các Instance



Hình 21: Instance khi đang hoạt động



Hình 22: Console của Instance khi đang hoạt động



Hình 23: Log của Instance khi đang hoạt động

openstack. admin • RegionOne

Project / Network / Networks

## Networks

Displaying 2 items

Name	Subnets Associated	Shared	External	Status	Admin State	Availability Zones	Actions
demo-net	demo-subnet 10.0.0.0/24	No	No	Active	UP	nova	<a href="#">Edit Network</a>
public1	public1-subnet 10.0.2.0/24	No	Yes	Active	UP	nova	<a href="#">Edit Network</a>

Displaying 2 items

Hình 24: Danh sách các network đã triển khai

openstack. admin • RegionOne

Network Topology

Topology Graph

Resize the canvas by scrolling up/down with your mouse/trackpad on the topology. Pan around the canvas by clicking and dragging the space behind the topology

[Toggle Labels](#) [Toggle Network Collapse](#) [Center Topology](#)

10.10.10.12/home/

Hình 25: Mô hình Network đã triển khai

openstack

admin • RegionOne

admin

RegionOne

Project

Admin

Identity

Identity / Users

Users

Users

User Name =  Filter 

Create User Delete Users

Displaying 8 items

	User Name	Description	Email	User ID	Enabled	Domain Name	Actions
	admin	-		4f24179a65e84cb7bbf743c183979683	Yes	Default	Edit
	glance	-		b977cdd4f5b04cc4aad7d4f3d4b6a2e09	Yes	Default	Edit
	cinder	-		7c7de5de250249cabfcc5c68c3cf252	Yes	Default	Edit
	placement	-		208265153f1441b99de1998e05777dfa	Yes	Default	Edit
	nova	-		78ac44b63da24a0dbfbctde8b1f92841	Yes	Default	Edit
	neutron	-		a33a999168434bf3bd89142522cce423	Yes	Default	Edit
	heat	-		648d14349d36421ab1b4cda6ad17d729	Yes	Default	Edit
	heat_domain_admin	-		7ecc62174216461cabe6c11d050bb011	Yes	-	Edit

Displaying 8 items

Hình 26: Danh sách các user đã tạo

```
root@kolla-ansible:~# docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED          STATUS          PORTS          NAMES
c89469223333   kolla/ubuntu-source-horizon:ussuri  "dumb-init --single-..." 59 minutes ago   Up 59 minutes   horizon        horizon
cebe85fc4caf   kolla/ubuntu-source-heat-engine:ussuri "dumb-init --single-..." 59 minutes ago   Up 59 minutes   heat_engine     heat_api_cfn
352902ce9f98   kolla/ubuntu-source-heat-api-cfn:ussuri "dumb-init --single-..." 59 minutes ago   Up 59 minutes   heat_api_cfn    heat_api_
8f49ec412b96   kolla/ubuntu-source-heat-api:ussuri   "dumb-init --single-..." About an hour ago Up 59 minutes   heat_api        neutron_metadata_agent
54e2bb572630   kolla/ubuntu-source-neutron-metadata-agent:ussuri "dumb-init --single-..." About an hour ago Up About an hour neutron_metadata_agent
41fbc272464    kolla/ubuntu-source-neutron-l3-agent:ussuri "dumb-init --single-..." About an hour ago Up About an hour neutron_l3_agent
20184799287d   kolla/ubuntu-source-neutron-dhcp-agent:ussuri "dumb-init --single-..." About an hour ago Up About an hour neutron_dhcp_agent
097bcc69320cb kolla/ubuntu-source-neutron-openvswitch-agent:ussuri "dumb-init --single-..." About an hour ago Up About an hour neutron_openvswitch_agent
a6112effc3cc   kolla/ubuntu-source-neutron-server:ussuri "dumb-init --single-..." About an hour ago Up About an hour neutron_server  neutron_server
d758cb30378f   kolla/ubuntu-source-openvswitch-vswitchd:ussuri "dumb-init --single-..." About an hour ago Up About an hour openvswitch_vswitchd
f91c6ffa163e   kolla/ubuntu-source-openvswitch-db-server:ussuri "dumb-init --single-..." About an hour ago Up About an hour openvswitch_db  nova_compute
d3f1e5c08cca   kolla/ubuntu-source-nova-compute:ussuri  "dumb-init --single-..." About an hour ago Up About an hour nova_compute    nova_libvirt
fcb9a1e5f916   kolla/ubuntu-source-nova-libvirt:ussuri  "dumb-init --single-..." About an hour ago Up About an hour nova_libvirt     nova_ssh
95c5529ee76f   kolla/ubuntu-source-nova-ssh:ussuri      "dumb-init --single-..." About an hour ago Up About an hour nova_ssh         nova_novncproxy
c28d9d152124   kolla/ubuntu-source-nova-novncproxy:ussuri "dumb-init --single-..." About an hour ago Up About an hour nova_novncproxy  nova_conductor
2fa91ab5ea3f   kolla/ubuntu-source-nova-conductor:ussuri "dumb-init --single-..." About an hour ago Up About an hour nova_conductor   nova_api
4638e0d93755   kolla/ubuntu-source-nova-api:ussuri      "dumb-init --single-..." About an hour ago Up About an hour nova_api         nova_scheduler
303d239ccfd8   kolla/ubuntu-source-nova-scheduler:ussuri "dumb-init --single-..." About an hour ago Up About an hour nova_scheduler   placement_api
8f19b284dd8f   kolla/ubuntu-source-placement-api:ussuri  "dumb-init --single-..." About an hour ago Up About an hour placement_api    cinder_backup
13882a086f34   kolla/ubuntu-source-cinder-backup:ussuri  "dumb-init --single-..." About an hour ago Up About an hour cinder_backup    cinder_volume
1638a35a7f59   kolla/ubuntu-source-cinder-volume:ussuri  "dumb-init --single-..." About an hour ago Up About an hour cinder_volume    cinder_scheduler
b182a9d1e83c   kolla/ubuntu-source-cinder-scheduler:ussuri "dumb-init --single-..." About an hour ago Up About an hour cinder_scheduler cinder_api
af5ea47c9f16   kolla/ubuntu-source-cinder-api:ussuri     "dumb-init --single-..." About an hour ago Up About an hour cinder_api
80b870700704   kolla/ubuntu-source-glance-api:ussuri     "dumb-init --single-..." About an hour ago Up About an hour glance_api       keystone_fernet
b4c64b134341   kolla/ubuntu-source-keystone-fernet:ussuri "dumb-init --single-..." About an hour ago Up About an hour keystone_fernet  keystone_ssh
f7572643fae2   kolla/ubuntu-source-keystone-ssh:ussuri   "dumb-init --single-..." About an hour ago Up About an hour keystone_ssh     rabbitmq
2b55815c85d7   kolla/ubuntu-source-keystone:ussuri       "dumb-init --single-..." About an hour ago Up About an hour keystone         tgtd
8878cc7ea7be   kolla/ubuntu-source-rabbitmq:ussuri       "dumb-init --single-..." About an hour ago Up About an hour rabbitmq         iscsid
6f2d6a20a490   kolla/ubuntu-source-tgtd:ussuri           "dumb-init --single-..." About an hour ago Up About an hour tgtd             memcached
cfeeed53abb9   kolla/ubuntu-source-iscsid:ussuri         "dumb-init --single-..." About an hour ago Up About an hour iscsid           mariadb_clustercheck
602c4aeca14c   kolla/ubuntu-source-memcached:ussuri      "dumb-init --single-..." About an hour ago Up About an hour memcached        mariadb
452b69aaf555   kolla/ubuntu-source-mariadb-clustercheck:ussuri "dumb-init --single-..." About an hour ago Up About an hour mariadb_clustercheck
f602c15f20c    kolla/ubuntu-source-mariadb:ussuri        "dumb-init --single-..." About an hour ago Up About an hour mariadb          keepalived
d0cfd0599a5    kolla/ubuntu-source-keepalived:ussuri     "dumb-init --single-..." About an hour ago Up About an hour keepalived       haproxy
2f3a0c8da7b6   kolla/ubuntu-source-haproxy:ussuri        "dumb-init --single-..." About an hour ago Up About an hour haproxy          chrony
1445259b0cb1   kolla/ubuntu-source-chrony:ussuri         "dumb-init --single-..." About an hour ago Up About an hour chrony           cron
b49b14417462   kolla/ubuntu-source-cron:ussuri           "dumb-init --single-..." About an hour ago Up About an hour cron            kolla_toolbox
ab17b42e53da   kolla/ubuntu-source-kolla-toolbox:ussuri  "dumb-init --single-..." About an hour ago Up About an hour kolla_toolbox    fluentd
c9a9c12abfce   kolla/ubuntu-source-fluentd:ussuri        "dumb-init --single-..." About an hour ago Up About an hour fluentd
```

Hình 27: Danh sách các service đang chạy trên node 1

```

root@kolla-ansible:~# docker ps

```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
0223e2977631	kolla/ubuntu-source-heat-engine:ussuri	"dumb-init --single-_"	46 hours ago	Up 2 hours		heat_engine
e2b6645b9d3f	kolla/ubuntu-source-heat-api-cfn:ussuri	"dumb-init --single-_"	46 hours ago	Up 2 hours		heat_api_cfn
fd3d6739b7f	kolla/ubuntu-source-heat-api:ussuri	"dumb-init --single-_"	46 hours ago	Up 2 hours		heat_api
64ebaec76f01	kolla/ubuntu-source-neutron-metadata-agent:ussuri	"dumb-init --single-_"	46 hours ago	Up 2 hours		neutron_metadata_agent
8481f5ed6202	kolla/ubuntu-source-neutron-l3-agent:ussuri	"dumb-init --single-_"	46 hours ago	Up 2 hours		neutron_l3_agent
1a042ce115ff	kolla/ubuntu-source-neutron-dhcp-agent:ussuri	"dumb-init --single-_"	46 hours ago	Up 2 hours		neutron_dhcp_agent
fab2d64ce7cb	kolla/ubuntu-source-neutron-openvswitch-agent:ussuri	"dumb-init --single-_"	46 hours ago	Up 2 hours		neutron_openvswitch_agent
25cdee70e882	kolla/ubuntu-source-neutron-server:ussuri	"dumb-init --single-_"	46 hours ago	Up 2 hours		neutron_server
b6663d4bc9cf	kolla/ubuntu-source-openvswitch-vswitchd:ussuri	"dumb-init --single-_"	46 hours ago	Up 2 hours		openvswitch_vswitchd
3056f14e9ef9	kolla/ubuntu-source-openvswitch-db-server:ussuri	"dumb-init --single-_"	46 hours ago	Up 2 hours		openvswitch_db
b8550c3815c	kolla/ubuntu-source-nova-compute:ussuri	"dumb-init --single-_"	46 hours ago	Up 35 seconds		nova_compute
b55debb56ae2	kolla/ubuntu-source-nova-libvirt:ussuri	"dumb-init --single-_"	46 hours ago	Up About a minute		nova_libvirt
fcdecf300c5e	kolla/ubuntu-source-nova-ssh:ussuri	"dumb-init --single-_"	46 hours ago	Up 2 hours		nova_ssh
687a8eeda8d1	kolla/ubuntu-source-nova-novncproxy:ussuri	"dumb-init --single-_"	46 hours ago	Up 2 hours		nova_novncproxy
147c707b2318	kolla/ubuntu-source-nova-conductor:ussuri	"dumb-init --single-_"	46 hours ago	Up About an hour		nova_conductor
27eef2c38791	kolla/ubuntu-source-nova-api:ussuri	"dumb-init --single-_"	46 hours ago	Up 2 hours		nova_api
85b0154cb3fe	kolla/ubuntu-source-nova-scheduler:ussuri	"dumb-init --single-_"	46 hours ago	Up About an hour		nova_scheduler
347f20d094d2	kolla/ubuntu-source-placement-api:ussuri	"dumb-init --single-_"	46 hours ago	Up 2 hours		placement_api
3226749cd3f3	kolla/ubuntu-source-cinder-backup:ussuri	"dumb-init --single-_"	46 hours ago	Up 2 hours		cinder_backup
a2e5d1238da7	kolla/ubuntu-source-cinder-volume:ussuri	"dumb-init --single-_"	46 hours ago	Up 2 hours		cinder_volume
00ef08eda564	kolla/ubuntu-source-cinder-scheduler:ussuri	"dumb-init --single-_"	46 hours ago	Up 2 hours		cinder_scheduler
dfcf2b7e0f08f	kolla/ubuntu-source-cinder-api:ussuri	"dumb-init --single-_"	46 hours ago	Up 2 hours		cinder_api
928ffcf8825c	kolla/ubuntu-source-glance-api:ussuri	"dumb-init --single-_"	46 hours ago	Up 2 hours		glance_api
337eb1319f17	kolla/ubuntu-source-rabbitmq:ussuri	"dumb-init --single-_"	46 hours ago	Up 2 hours		rabbitmq
d3a4102fd8e4	kolla/ubuntu-source-tgtd:ussuri	"dumb-init --single-_"	46 hours ago	Up 2 hours		tgtd
57d402ca13f4	kolla/ubuntu-source-iscsid:ussuri	"dumb-init --single-_"	46 hours ago	Up 2 hours		iscsid
5123a7541409	kolla/ubuntu-source-memcached:ussuri	"dumb-init --single-_"	46 hours ago	Up 2 hours		memcached
fb8cfd3c0c8f	kolla/ubuntu-source-keepalived:ussuri	"dumb-init --single-_"	47 hours ago	Up 2 hours		keepalived
57853fc11a2d	kolla/ubuntu-source-haproxy:ussuri	"dumb-init --single-_"	47 hours ago	Up 2 hours		haproxy
3ab578015dd1	kolla/ubuntu-source-mariadb-clustercheck:ussuri	"dumb-init --single-_"	47 hours ago	Up 2 hours		mariadb_clustercheck
43b010a26ad2	kolla/ubuntu-source-mariadb:ussuri	"dumb-init --single-_"	47 hours ago	Up 2 hours		mariadb
1e602ecd56b7	kolla/ubuntu-source-cron:ussuri	"dumb-init --single-_"	47 hours ago	Up 2 hours		cron
f1732ec35174	kolla/ubuntu-source-kolla-toolbox:ussuri	"dumb-init --single-_"	47 hours ago	Up 2 hours		kolla_toolbox
a99003fd5ef3	kolla/ubuntu-source-fluentd:ussuri	"dumb-init --single-_"	47 hours ago	Up 2 hours		fluentd

Hình 28: Danh sách các service đang chạy trên node 2

## KẾT LUẬN

Như vậy, thông qua quá trình nghiên cứu và năm chương của đề tài “Nghiên cứu, triển khai Openstack sử dụng docker” về cơ bản đạt được một số mục tiêu sau:

- Hiểu được những kiến thức cơ bản và tổng quan về điện toán đám mây
- Hiểu được những kiến thức cơ bản về một số loại ảo hoá
- Hiểu được những kiến thức cơ bản về docker, openstack
- Triển khai được các dịch vụ của openstack bằng docker và ansible
- Xây dựng được một đám mây dựa trên mã nguồn mở OpenStack trên môi trường ảo hóa Vitruabox.

Tuy nhiên, do một số hạn chế về tài liệu, kiến thức và điều kiện thực tế, nên báo cáo còn một số vấn đề chưa thể khắc phục được:

- Máy ảo tạo được trên đám mây còn ở dạng đơn giản
- Mô hình triển khai chưa thực sự sát với thực tế

Do đó, trong tương lai, chúng em sẽ tiếp tục nghiên cứu, phát triển đề tài này một cách hoàn thiện hơn, có thể áp dụng vào các cơ quan tổ chức, doanh nghiệp

Lời cuối, chúng em xin gửi lời biết ơn chân thành tới Thầy giáo, TS. Phạm Văn Hưởng, đã giúp đỡ chúng em trong suốt thời gian qua. Qua đây em cũng mong nhận được sự đóng góp ý kiến của thầy cô, cũng như các bạn có cùng mối quan tâm để đề tài được hoàn thiện hơn. Em xin chân thành cảm ơn!



## DANH MỤC TÀI LIỆU THAM KHẢO

- Github: [https://github.com/huyduong2792000/openstack\\_docker.git](https://github.com/huyduong2792000/openstack_docker.git)