ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

TRẦN ANH TUẤN

NGHIÊN CỨU VÀ TRIỂN KHAI HỆ THỐNG PRIVATE CLOUD CHO CÁC ỨNG DỤNG ĐÀO TẠO VÀ THỰC HÀNH DỰA TRÊN GIẢI PHÁP MÃ NGUỒN MỞ OPENSTACK

LUẬN VĂN THẠC SĨ CÔNG NGHỆ THÔNG TIN

Hà nội, 11/2019

ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

TRẦN ANH TUẤN

NGHIÊN CỨU VÀ TRIỂN KHAI HỆ THỐNG PRIVATE CLOUD CHO CÁC ỨNG DỤNG ĐÀO TẠO VÀ THỰC HÀNH DỰA TRÊN GIẢI PHÁP MÃ NGUỒN MỞ OPENSTACK

Quyết định số: 655/QĐ-CTSV

Ngành: Mạng máy tính và truyền thông dữ liệu

Chuyên ngành: Mạng máy tính và truyền thông dữ liệu

Mã số: 8480102.01

Giảng viên hướng dẫn: TS. Hoàng Xuân Tùng

LUẬN VĂN THẠC SĨ CÔNG NGHỆ THÔNG TIN

Hà nội, tháng 11/2019

LỜI CẢM ƠN

Trước tiên em xin dành lời cảm ơn chân thành đến thầy Hoàng Xuân Tùng, thầy đã hướng dẫn, khuyến khích, chỉ bảo và tạo cho em những điều kiện tốt nhất từ khi bắt đầu cho tới khi hoàn thành công việc của mình.

Em xin dành lời cảm ơn chân thành tới các thầy cô giáo khoa Công nghệ thông tin, trường Đại học Công nghệ, ĐHQGHN đã tận tình đào tạo, cung cấp cho em những kiến thức vô cùng quý giá và đã tạo điều kiện tốt nhất cho em trong suốt quá trình học tập, nghiên cứu tại trường để em hoàn thành khoá luận và là hành trang cho em sau này..

Cuối cùng em xin cảm ơn đến bạn bè, đồng nghiệp, người thân đã động viên, giúp đỡ, tạo điều kiện những khi vấp phải những khó khăn để em hoàn thành luận văn này.

Mặc dù đã rất cố gắng nhưng do kiến thức còn nhiều hạn chế nên luận văn của em không thể tránh khỏi những sai sót. Em rất mong nhận được sự góp ý của các thầy cô và các bạn để em có thể hoàn thiện và khắc phục những thiếu sót của mình.

Em xin chân thành cảm ơn!

LỜI CAM ĐOAN

Tôi là Trần Anh Tuấn, học viên K23 trường Đại học Công Nghệ - ĐHQGHN, xin cam đoan rằng luận văn thạc sĩ công nghệ thông tin "Nghiên cứu và triển khai hệ thống Private Cloud cho các ứng dụng đào tạo và thực hành dựa trên giải pháp mã nguồn mở Openstack" là luận văn nghiên cứu của tôi, được thầy Hoàng Xuân Tùng hướng dẫn và không sao chép lại của người khác. Tất cả những tài liệu trích dẫn đều có nguồn gốc rõ ràng.

Nếu có sai phạm, tôi xin chịu hoàn toàn trách nhiệm chịu mọi hình thức kỷ luật theo quy định cho lời cam đoan này.

Hà Nội, ngày …. tháng 11 năm 2019

Tác giả luận văn

Trần Anh Tuấn

MỤC LỤC

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG BIỂU

ĐẶT VẤN ĐỀ

Trong sự phát triển của công nghệ thông tin, đặc biệt là sự phát triển điện toán đám mây (Cloud Computing) trong những ứng dụng của cuộc sống chưa bao giờ phổ biến và tiện lợi như hiện nay. Việc ứng dụng điện toán đám mây trong các doang nghiệp, các đơn vị hành chính sự nghiệp, cơ sở giáo dục là nhu cầu cấp thiết trong việc xây dựng, thiết lập cơ sở hạng tầng và năng lực lưu trữ của các hệ thống hiện nay. Trên thế giới, điện toán đám mây là công nghệ đã phát triển khá lâu và đã được đẩy mạnh trong nhứng năm trở lại đây bởi các công ty công nghệ như Amazon, Google, Microsoft... Ngoài ra, nhiều doanh nghiệp tự xây dựng và tạo ra các dự án Opensource liên quan tới điện toán đám mây như Openstack, Cloudstack, Eucalyptus, PetiteCloud... Ở Việt Nam, các doang nghiệp đã triển khai hệ thống điện toán đám mây nhằm khai thác các dịch vụ trên đó như Viettel, FPT, CMC… Chính vì nhu cầu ứng dụng cao về xây dựng, triển khai và vận hành điện toàn đám mây có chất lượng cao thì nhu cầu đào tạo nguồn nhân lực có kiến thức và kỹ năng liên quan đến điện toán đám mây. Để có thể đào tạo nguồn nhân lực có trình độ cao, các cở sở giáo dục cần nghiên cứu, giảng dạy, đào tạo cũng như xây dựng các ứng dụng liên quan tới điện toàn đám mây. Đó là mối quan tâm đặc biệt trong môi trường giáo dục đại học vì đây là cơ sở nghiên cứu, đào tạo và thực hành. Chúng tôi sẽ nghiên cứu và triển khai hệ thống Private Cloud tại Bộ môn mạng của trường Đại học Công nghệ - Đại học Quốc gia Hà Nội.

Do các ứng dụng đào tạo và thực hành là các ứng dụng đặc thù trong giáo dục (đặc biệt là áp dụng cho trường đại học) nên đòi hỏi việc triển khai cho từng đơn vị là khác nhau. Các ứng dụng cho đào tạo và thực hành thường được sử dụng để phục vụ cho công tác quản lý, giảng dạy và học tập cho nên các đối tượng sử dụng rất đa dạng như các cán bộ quán lý của các phòng ban, các cán bộ quán lý của các khoa trong nhà trường hay các giảng viên, sinh viên… dẫn tới nhu cầu bảo mật phục vụ cho các đối tượng hay nhóm đối tượng là khác nhau nên khi triển khai Private cloud, hệ thống cần hoạt động ổn định, đảm bảo tính bảo mật nhưng không gây nhiều khó khăn cho quản trị viên khi vận hành và phát triển. Ngoài ra, do các ứng dụng đào tạo và thực hành phục vụ cho các mục đích và đối tượng người dùng khác nhau nên cần quy hoạch hệ thống hạ tầng cơ sở một cách tối ưu cũng là những thách thức và khó khăn khi triển khai hệ thống mạng truyền thống nên khi triển khai Private cloud, đặc biệt là dựa trên mà nguồn mở Openstack, hệ thống cần được quy hoạch một cách hiệu quả và tối ưu nhưng vẫn

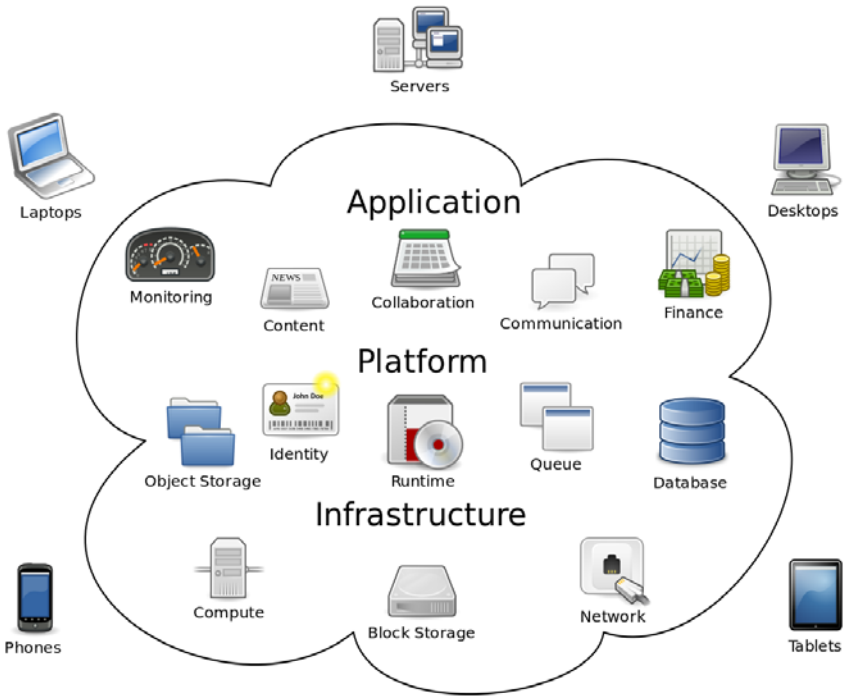
đáp ứng nhu cầu về nền tảng cơ sở hạ tầng cho công tác quản lý, giảng dạy và học tập. Luận văn này sẽ giới thiệu các khái niệm chung, hệ thống thực tế đang triển khai cũng như các bài toàn khi triển khai, cụ thể là quy hoạch hệ thống server đồng thời giải quyết bài toán quản trị và kết nối mạng nhằm phục vụ cho các hoạt động đào tạo và thực hành.

CHƯƠNG 1. GIỚI THIỆU CHUNG

1. 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. [1]

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. [1]



Hình 1-1. Mô hình Cloud Computing

Cloud computing đạt được hiệu quả kinh tế do sự chia sẻ tài nguyên, cụ thể là cho phép các doanh nghiệp giảm chi phí về cơ sở hạ tầng. Sự phát triển của các mạng công nghệ tốc độ cao, giá thành của máy tính và các thiết bị lưu trữ thấp cũng như việc triển khai rộng rãi ảo hóa phần cứng, kiến trúc hướng dịch vụ, mô hình tự động hóa và các tiện ích máy tính sẵn có đã dẫn đến sự hình thành và phát triển của cloud computing.

Theo NIST (Viện Quốc gia về tiêu chuẩn và công nghệ Mỹ), Cloud computing gồm năm đặc tính cơ bản: *On-demand self-service, Broad network access, Resource pooling, Rapid elasticity or expansion* và *Measured service*. Trong đó, ***On-demand self-service*** được hiểu là khách hàng có thể tự quản lý dịch vụ của bản thân mà không cần sự trợ giúp của đơn vị IT ngoài hoặc nhà cung cấp hosting. ***Broad network access*** được hiểu là các dịch vụ cloud cần được truy cập thông qua các công nghệ mạng thông thường. ***Resource pooling*** được hiểu là các dịch vụ chạy trong datacenter sử dụng chung hạ tầng và được chia sẻ với nhiều người dùng khác nhau. ***Rapid elasticity or expansion*** được hiểu là dịch vụ cloud có khả năng dễ dàng thay đổi theo đúng nhu cầu thực tế. Các dịch vụ phải được thêm hay bớt theo đúng nhu cầu. ***Measured service*** được hiểu là dịch vụ cloud có khả năng tối ưu tài nguyên sử dụng của người dùng và được cập nhật thường xuyên. [3]

Ba mô hình dịch vụ được NIST trình bày dùng để định nghĩa các dịch vụ cung cấp trong Cloud computing bao gồm: *Software as a Service (SaaS), Platform as a Service (PaaS)* và *Infrastructure as a Service (IaaS).* Trong mô hình *Software as a Service (SaaS),* các phần mềm chạy trên datacenter và được quản lý bởi nhà cung cấp dịch vụ. Microsoft Office 365 là ví dụ điển hình của mô hình SaaS. Trong mô hình *Platform as a Service (PaaS),* một server chạy trên datacenter và được quản lý bởi nhà cung cấp dịch vụ. Tuy nhiên người dùng được quản lý các ứng dụng và lưu trữ dữ liệu trên server. Ngoài Windows Azure, Amazon Web Service (AWS) là ví dụ điển hình trong mô hình này. Trong mô hình *Infrastructure as a Service (IaaS),* server chạy trên datacenter của nhà cung cấp dịch vụ nhưng được quản lý hoàn toàn bởi người dùng. Mọi chương trình, ứng dụng chạy trên server được quản lý bởi người dùng: bao gồm các OS (bất kỳ hệ điều hành nào), các ứng dụng và data lưu trữ trên Server. [3]

Ngoài ra, NIST cũng liệt kê bốn mô hình triển khai cho Cloud computing bao gồm: *Private Cloud, Public Cloud, Community Cloud* và *Hybrid Cloud*. Mô hình *Private Cloud* là mô hình mà cơ sở hạ tầng được triển khai dành cho chỉ duy nhất một khách hàng. Mỗi hệ thống *Private Cloud* có thể được đặt tại datacenter của người dùng nhưng cũng có thể được đặt tại datacenter của nhà cung cấp dịch vụ. Các hệ thống *Private Cloud* có thể được quản lý bởi người dùng, các nhà cung cấp dịch vụ hoặc một đơn vị thứ ba chuyên cung cấp các dịch vụ Cloud. Tuy nhiên, người dùng luôn phải chịu toàn bộ chi phí cho giải pháp. Các hệ thống *Public Cloud* là mô hình mà cơ sở hạ tầng được triển khai để tất cả mọi người, đều có thể sử dụng, không giới hạn số lượng, đối tượng người dùng (đó có thể là người dùng cá nhân hoặc các công ty lớn). *Public Cloud* được sử dụng phổ biến và dễ dàng. Mirosoft Office 365, Microsoft Azure, Amazon Web Service (AWS) và NTC Cloud Server là những ví dụ điển hình cho giải pháp *Public Cloud*. Mô hình *Community Cloud* là mô hình mà cơ sở hạ tầng được chia sẻ cho nhiều tổ chức hoặc người dùng có chung mục đích. Việc quản lý một *Community Cloud* có thể do một tổ chức hoặc một đơn vị thứ ba chuyên cung cấp các dịch vụ Cloud. Mô hình *Hybrid Cloud* là mô hình mà cơ sở hạ tầng được kết hợp từ 3 mô hình Cloud kể trên. Trong hệ thống Mirosoft Office 365, có thể các mailbox được lưu trữ trong hệ thống của Microsoft datacenter, nhưng cũng có thể kết hợp với Exchange Server và các mailbox dùng riêng. Kết hợp lại, tạo nên một hệ thống lai – hybrid messaging system. [3]

Trong phạm vi triển khai của luận văn, mô hình Private Cloud được sử dụng để triển khai cài đặt.

2. Tổng quan về Private Cloud:

Private Cloud được định nghĩa là các dịch vụ được cung cấp qua Internet hoặc mạng nội bộ và bị giới hạn người dùng thay vì cho phép truy cập công khai do vậy còn được gọi là Internal Cloud hay Corporate Cloud. Private Cloud hỗ trợ doanh nghiệp những tiện ích như self-service, khả năng mở rộng và tính linh hoạt như khi sử dụng Public Cloud. Ngoài ra, Private Cloud còn cung cấp tính riêng tư và độ bảo mật cấp độ cao thông qua các firewall và internal hosting để đảm bảo các hoạt động và dữ liệu quan trọng không thể truy cập bởi nhà cung cấp dịch vụ bên thứ ba. [2]

Hai mô hình dịch vụ được áp dụng trong Private Cloud bao gồm: *Platform as a Service (PaaS)* và *Infrastructure as a Service (IaaS)*. Mô hình đầu tiên là *Platform as a Service (PaaS)* cho phép một tổ chức cung cấp mọi thứ từ các ứng dụng miễn phí cho đến các ứng dụng trả phí. Mô hình thứ hai là *Infrastructure as a Serice (IaaS)* cho phép một tổ chức sử dụng tài nguyên của cơ sở hạ tầng như máy tính, hệ thống mạng và các thiết bị lưu trữ như một dịch vụ. [2]

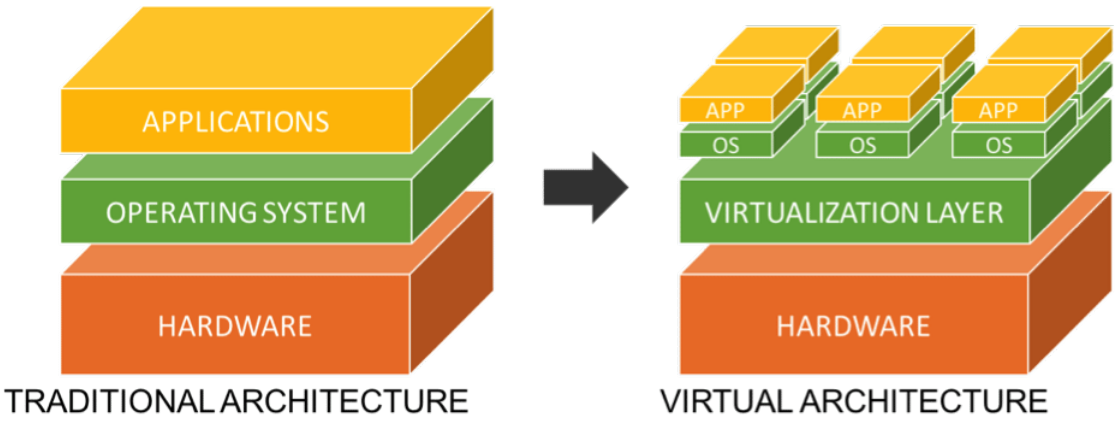
Private Cloud còn kết hợp với Public Cloud để tạo ra Hybrid Cloud cho phép doanh nghiệp tận dụng Cloud Bursting để tối ưu không gian và quy mô các dịch vụ Cloud Computing khi người dùng hay tổ chức tăng nhu cầu sử dụng. [2]

Openstack, Cloudstack là một trong những nền tảng phần mềm điển hình cho mô hình Private Cloud.

3. Tổng quan về Virtualization:

Ảo hóa (virtualization) được định nghĩa là sự triển khai một hệ thống máy tính ảo trên nền một hệ thống máy tính thật. Ngoài ra, ảo hóa đề cập đến việc giả lập mọi thiết bị bằng bao gồm sự ảo hóa các nền tảng phần cứng máy tính, các thiết bị lưu trữ, tài nguyên cũng như hệ thống mạng máy tính. Nói cách khác, ảo hóa cũng có thể được coi là một kỹ thuật cho phép người dùng chia sẻ một instance vật lý của một tài nguyên hoặc một ứng dụng giữa nhiều người dùng và tổ chức khác nhau. [4]

Ý tưởng của ảo hóa không phải là điều gì mới. Ý tưởng này được IBM giới thiệu vào năm 1960 khi các mainframe được sử dụng. Các mainframe hầu như không được sử dụng hết hiệu suất cũng như tính năng. Ảo hóa là một phương pháp tối ưu trong việc cung cấp các tài nguyên hệ thống cho các ứng dụng khác nhau hoạt động trên các mainframe.



Hình 1-2. 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.

Ảo hóa được phân thành rất nhiều loại, cụ thể như sau:

Bảng 1-1. Các loại ảo hóa

STT	Loại ảo hóa	Công nghệ ảo hóa	Mức độ quan tâm
1	Phần cứng (Hardware)	Server Virtualization	Cao
		Desktop Virtualization	Trung bình
2	Phần mềm (Software)	Application virtualization	Trung bình
		Workspace virtualization	Trung bình
		Service virtualization	Cao
3	Bộ nhớ (Memory)	Memory virtualization	Cao
		Virtual memory	Trung bình
4	Lưu trữ (Storage)	Storage virtualization	Cao
		Distributed file system	Trung bình
		Virtual file system	Trung bình
		Storage hypervisor	Cao
		Virtual disk	Trung bình
5	Dữ liệu (Data)	Data virtualization	Cao
		Database virtualization	Cao
6	Hệ thống mạng (Network)	Network virtualization	Cao
		Virtual private network	Trung bình

Nền tảng hypervisor được giới thiệu sau đây cũng sử dụng công nghệ ảo hóa.

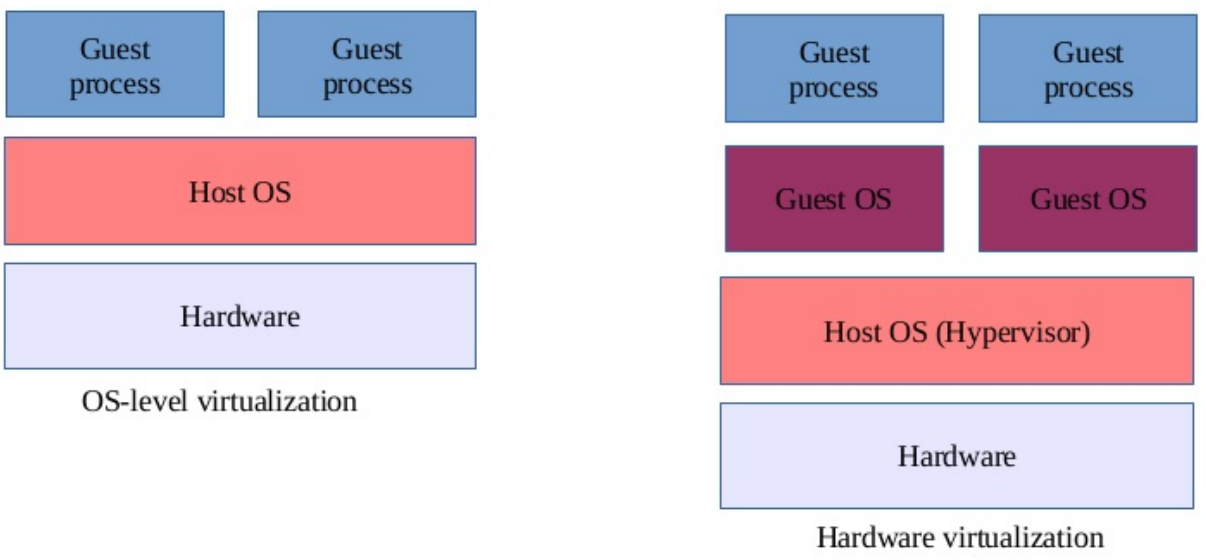
4. Tổng quan về Hypervisor:

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. [4]

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. [4]

Hypervisor là phần mềm máy tính, firmware hay phần cứng nhằm tạo và chạy máy ảo. Một máy tính mà một hypervisor chạy một hay nhiều máy ảo được gọi là máy *Host* và mỗi máy ảo được gọi là máy *Guest*. [1]

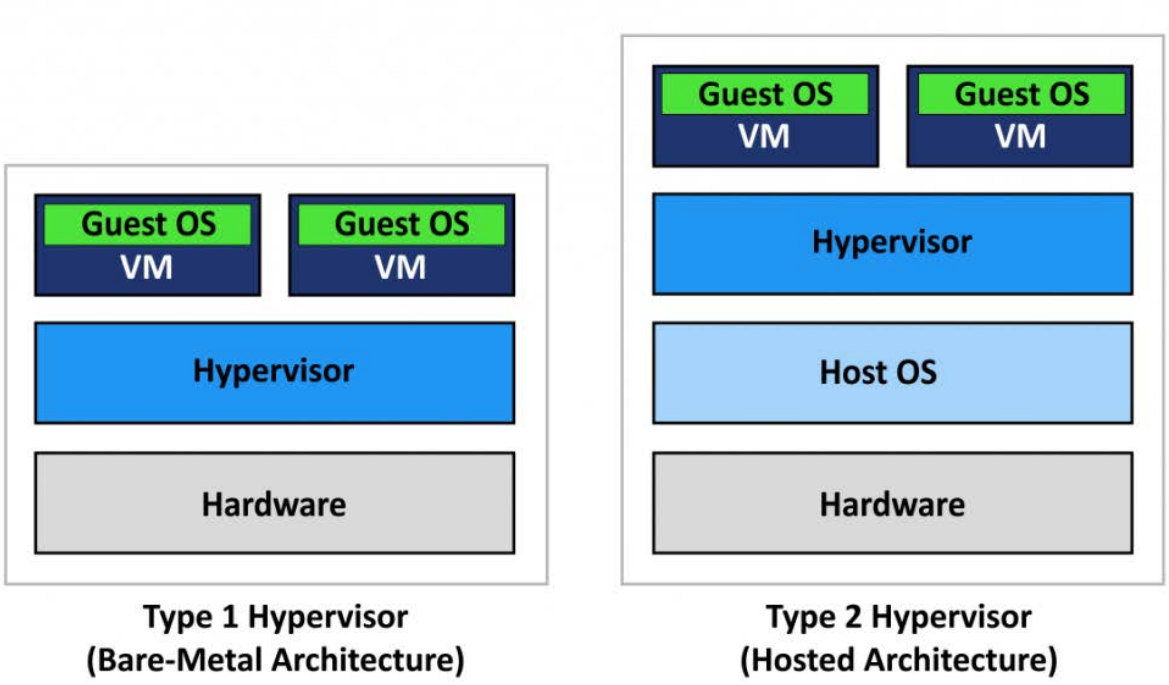
Hypervisor biểu diễn bằng một nền tảng vận hành ảo chứa Guest OS và quản lý vận hành Guest OS. Các instance trong các hệ điều hành có thể chia sẻ nhau các tài nguyên phần cứng ảo, chẳng hạn, các instance chứa Linux, Windows và macOS có thể cùng chạy trên một máy tính đơn x86. Cơ chế Hypervisor trái ngược 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. [1]



Hình 1-3. Hai cơ chế ảo hóa phần cứng

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. [4]

Dựa trên nền tảng đó, Hypervisor được phân thành 2 loại như sau: *Native hypervisor (Bare-metal hypervisor), Hosted hypervisor*.



Hình 1-4. Phân loại hypervisor

a. Native hypervisor (Bare-metal hypervisor)

Hypervisor chạy trực tiếp trên phần cứng server và quản lý các Guest OS. Các hypervisor được IBM phát triển vào những năm 1960, bao gồm phần mềm test *SIMON*, hệ điều hành *CP/CMS* hay Hệ điều hành Antsle, Xen, XCP-ng, SPARC Oracle VM Server, Oracle VM Server x86, Microsoft Hyper-V, Xbox One và VMware ESXi (phiên bản trước đó là VMware ESX). [1]

b. Hosted hypervisor

Hypervisor chạy thông qua một chương trình máy tính hay một hệ điều hành chạy nền. Mỗi Guest OS chạy như một vi xử lý của *host*. Hypervisor khái quát Guest OS từ Host OS, bao gồm VMware Workstation, VMware Player, VirtualBox, Parallels Desktop for Mac và QEMU. [1]

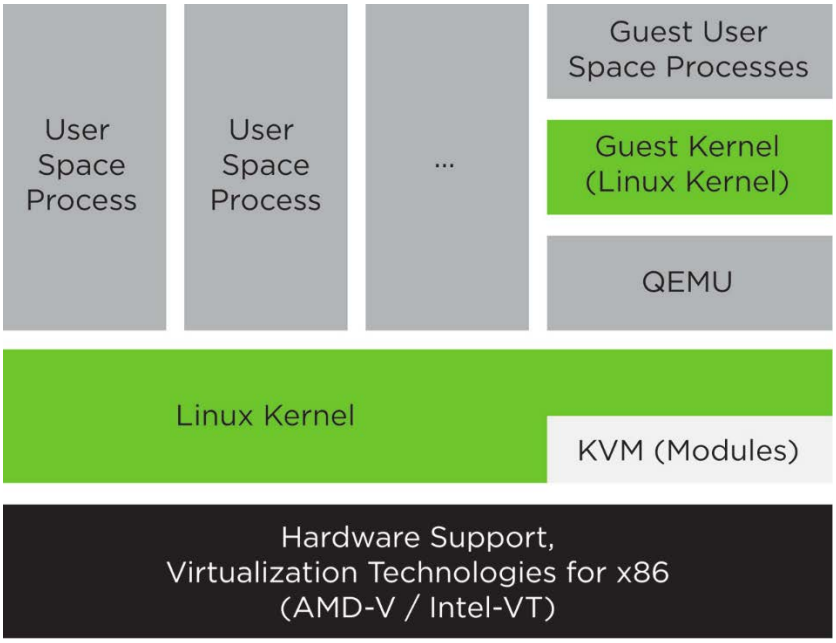
Tuy nhiên sự khác biệt của 2 loại hypervisor không phải lúc nào cũng rõ ràng. Ví dụ như Kernel-based Virtual Machine (KVM) của Linux hay Bhyve của FreeBSD là những module nhân có thể thao tác trực tiếp trên máy host như một *native hypervisor*. Đồng thời, do các phiên bản Linux và FreeBSD vẫn sử dụng nền tảng các hệ điều hành chung và cạnh tranh nhau về các ứng dụng cho các tài nguyên máy ảo nên KVM và Bhyve có thể được xem như một *hosted hypervisor*. [1]

CHƯƠNG 2. GIỚI THIỆU VỀ LIBVIRT- KVM, OPENSTACK, CLOUDSTACK

I. LIBVIRT- KVM

1. KVM

Kernel-based Virtual Machine (KVM) là một module ảo hóa nằm trong nhân Linux cho phép nhân thực hiện các chức năng như một hypervisor. KVM đã xuất hiện vào tháng 10 năm 2006 và được tích hợp vào trong nhân Linux trong phiên bản 2.6.20 được phát hành vào ngày 5 tháng 2 năm 2007. KVM đòi hỏi một vi xử lý có hỗ trợ phần cứng ảo hóa mở rộng như Intel VT hay AMD-V. KVM ban đầu được thiết kế cho bộ xử lý x86 và sau đó đã được chuyển sang S/390, PowerPC, IA-64 và ARM. [1]



Hình 2-1. Mô hình KVM

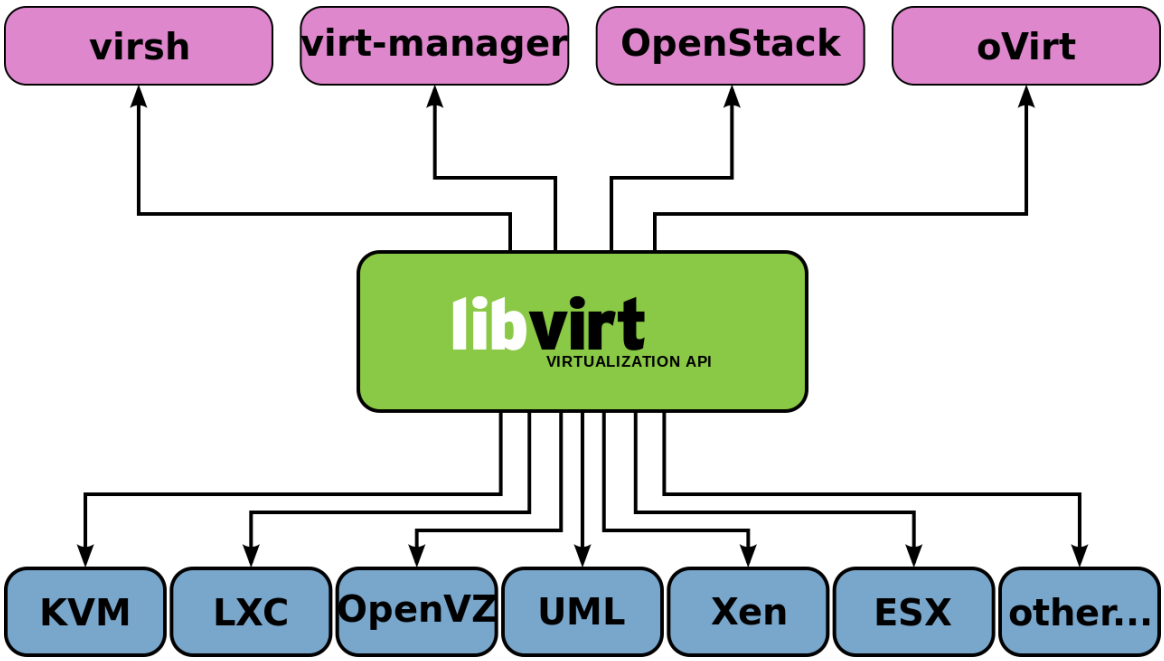
KVM cung cấp tính năng hỗ trợ ảo hóa phần cứng cho các Guest OS khác nhau bao gồm Linux, BSD, Solaris, Windows, Haiku, ReactOS, Plan 9, AROS Research Operating System và MacOS hay như Android 2.2, GNU/Hurd (Debian K16), Minix 3.1.2a, Solaris 10 U3 và Darwin 8.0.1 cũng như các hệ điều hành khác và các thế hệ hệ điều hành mới. [1]

KVM chuyển đổi nhân Linux thành một hypervisor. KVM sử dụng một phần mềm mô phỏng phần cứng là QEMU để tối ưu hóa các thành phần của hệ thống. QEMU mô phỏng vi xử lý và mô phỏng các thiết bị ngoại vi khác như ổ đĩa, hệ thống mạng, VGA, PCI, USB, các cổng nối tiếp/ song song… dùng cho việc thiết lập hoàn chỉnh hệ thống phần cứng ảo để mô phỏng và cài đặt các Guest OS. [7]

KVM là một hypervisor chính của Openstack compute (Nova) trong Openstack.[6]

2. LIBVIRT

Libvirt là một API mã nguồn mở, công cụ quản trị nền tảng ảo hóa. Libvirt được sử dụng để quản lý KVM, Xen, VMware ESXi, QEMU và các công nghệ ảo hóa khác. Các API này được sử dụng rộng rãi trong việc phát triển các giải pháp cloud-based nhờ lớp kiến trúc của các hypervisor. [1]



Hình 2-2. Mô hình mô tả vai trò Libvirt trong Hypervisor

Libvirt là một thư viện ngôn ngữ C kết hợp với các ngôn ngữ khác như Python, Perl, OCaml, Ruby, Java, JavaScript (đặc biệt là Node.js) và PHP. [1]

Libvirt hỗ trợ các lập trình viên sử dụng ngôn ngữ lập trình khác nhau các class/package được gọi là libvirtmod. Sự triển khai của libvirtmod được liên kết chặt chẽ các nền tảng kế thừa ngôn ngữ lập trình C/C++ với chính nói về mặt cú pháp và cấu trúc hàm. [1]

Dòng lệnh giao diện của libvirt được thực thi bởi câu lệnh *virsh*. Ngoài ra, libvirt còn được sử dụng trong các công cụ quản trị như oVirt hay virt-manager.[7]

Trong Openstack, libvirt giúp KVM hoạt động một cách tối ưu và hiệu quả. [6]

II. CLOUDSTACK:

Cloudstack (hay còn gọi là Apache Cloudstack) là một nền tảng phần mềm mã nguồn mở Cloud Computing, được phát triển theo mô hình *Infrastructure as a Service (IaaS)* dùng để quản lý tài nguyên hệ thống máy tính. [1]

Cloudstack sử dụng các hypervisor như KVM, VMware vSphere, VMware ESXi, VMware vCenter, XenServer/XCP, Oracle VM server và Microsoft Hyper-V cho các yêu cầu về ảo hóa. Cloudstack cung cấp các API của Amazon Web Services (AWS) cũng như giao diện Open Cloud Computing của tổ chức Open Grid Forum (OGF). [1]

Cloudstack là được pháp triển bởi tổ chức Apache Software Foundation dựa theo các điều khoản của Giấy phép Apache. [1]

Cloudstack được phát triển bởi Cloud.com và được Cloud.com phát hành vào tháng 5 năm 2010. Vào 12 tháng 7 năm 2011, Citrix Systems mua lại Cloud.com và đến tháng 8 năm 2011, Citrix Systems đã hợp tác với Apache Software Foundation để phát triển Cloudstack theo Giấy phép Apache. Vào tháng 2 năm 2012, Citrix Systems phát hành CloudStack 3.0. [8]

Người dùng có thể quản lý Cloudstack một cách dễ dàng thông qua giao diện web, công cụ dòng lệnh hoặc thông qua các API RESTful. Ngoài ra, Cloudstack cung cấp API tương thích với AWS EC2 và AWS S3 để các tổ chức có thể phát triển Hybrid Cloud. [9]

Cloudstack quản lý và tổ chức phần cứng theo cấu trúc phần tầng. Tầng thấp nhất gồm các hypervisor (máy ảo được coi là một *Host* và hệ thống lưu trữ ảo chung) được kết nối với nhau tạo thành một *Cluster*. Tầng thứ hai gồm các *Cluster* kết nối với nhau thông qua một switch layer 2 tạo thành một *Pod*. Tầng cuối cùng cũng là tầng cao nhất gồm các *Pod* kết nối với nhau thông qua một switch layer 3 và kết nối đến khối thiết bị trên server. [10]

III. OPENSTACK:

1. Tổng quan về Openstack:

Openstack là một nền tảng phần mềm mã nguồn mở Cloud Computing, 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, các công cụ dòng lệnh hoặc thông qua các API RESTful. [1]

Openstack.org là đơn vị phát hành Openstack dựa theo các điều khoản của Giấy phép Apache. [1]

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. [1]

a. Lịch sử hình thành và phát triển

Bảng 2-1. Lịch sử hình thành và phát triển của Openstack

Năm	Thời gian cụ thể	Sự kiện	Mục tiêu
2009		Bộ mã nguồn đầu tiên được phát triển từ nền tảng Nebula của NASA cũng như nền tảng Cloud Files của Rackspace (mô hình Cloud gốc được thiết kế bởi quản trị viên website NASA Ames là Megan A. Eskey, là một kiến trúc nguồn mở có tên OpenNASA v2.0). Các module Cloudstack và Openstack được kết hợp và phát hành dưới dạng nguồn mở bởi nhóm NASA Nebula phối hợp với Rackspace.	
2010	Tháng 7	Rackspace Hosting và NASA đã cùng nhau đưa ra một sáng kiến phần mềm Cloud nguồn mở được gọi là OpenStack.	Nhiệm vụ của OpenStack là "*nền tảng Cloud Computing nguồn mở phổ biến, đáp ứng nhu cầu của public cloud và private cloud bất kể quy mô,*

Năm	Thời gian cụ thể	Sự kiện	Mục tiêu
			bằng cách đơn giản để thực hiện và có thể mở rộng quy mô".
	Tháng 10	SUSE công bố công khai bản thương mại hóa đầu tiên dánh cho các thiết bị hỗ trợ OpenStack được cấu hình đầy đủ dựa trên bản phát hành OpenStack Diablo. Vào ngày 21/10, bản phát hành chính thức đầu tiên, có tên gọi Austin, được phát hành.	Dự án Openstack nhằm giúp các tổ chức cung cấp dịch vụ Cloud Computing chạy trên phần cứng tiêu chuẩn, với kế hoạch cập nhật phàn mềm thường xuyên sau vài tháng
2011		Các nhà phát triển bản Ubuntu-Linux đã sử dụng OpenStack với phiên bản không được hỗ trợ của Openstack Bexar được phát hành cho Ubuntu 11.04 (Natty Narwhal). Tổ chức tài trợ Ubuntu là Canonical ngay sau đó đã giới thiệu hỗ trợ đầy đủ cho các OpenStack Cloud, bắt đầu với việc phát hành Cactus của OpenStack. OpenStack đã có sẵn trong Debian Sid từ bản phát hành Openstack Cactus và bản phát hành đầu tiên của Debian chứa phiên bản OpenStack 2012.1 (Openstack Essex) là Debian 7.0 (Debian Wheezy).	
2012		NASA đã rút khỏi OpenStack với tư cách là thành viên đóng góp tích cực và thay vào đó đã đưa ra quyết định chiến lược sử dụng Amazon web service cho các dịch vụ cloud-based. Redhat ra mắt bản OpenStack phân tán của họ cũng bắt đầu với phiên bản Openstack Essex. HP bắt đầu triển khai HP Helion Public Cloud trên OpenStack	

Năm	Thời gian cụ thể	Sự kiện	Mục tiêu
	Tháng 8	SUSE đã ra mắt bản OpenStack phân tán dành cho doanh nghiệp được hỗ trợ thương mại dựa trên bản phát hành Openstack Essex.	
	Tháng 11	Dịch vụ kỹ thuật số của Chính phủ Vương quốc Anh (Government Digital Service - GDS) đã phát triển dựa trên phiên bản OpenNASA v2.0 là phiên bản Government as a Platform (GaaP).	
2013	Tháng 7	NASA đã tiến hành một cuộc kiểm toán nội bộ với lý do thiếu tiến bộ kỹ thuật và các yếu tố khác là lý do chính khiến cơ quan từ bỏ tư cách là nhà phát triển tích cực của dự án và thay vào đó tập trung vào việc sử dụng các Public Cloud.	
	Tháng 12	Oracle tuyên bố đã tham gia OpenStack với tư cách là Nhà phát triển và dự định mang OpenStack vào trong Oracle Solaris, Oracle Linux và nhiều sản phẩm của mình.	
2014		Tại Ngày hội Interop and Tech Field, software-defined networking (SDN) đã được Avaya trình bày bằng cách sử dụng shortest-path-bridging và OpenStack như một mô hình tự động, tự động mở rộng từ data center đến thiết bị, loại bỏ thao tác thủ công trong việc cung cấp dịch vụ.	
	Tháng 5	HP đã công bố HP Helion và phát hành bản OpenStack HP Helion, bắt đầu với phiên bản IceHouse.	
	Tháng 9	Vào ngày 24/9, Oracle cũng phát hành các phiên bản Oracle OpenStack là sự kết hợp Oracle Solaris và Oracle Linux tạo ra Openstack Icehouse	

Năm	Thời gian cụ thể	Sự kiện	Mục tiêu
2015	Tháng 3	Tính đến thời điểm này, NASA vẫn sử dụng OpenStack Private Cloud và có RFPs để hỗ trợ OpenStack Public Cloud.	
2016		OpenStack Foundation là tổ chức quản lý OpenStack.	

b. Các phiên bản của Openstack

Bảng 2-2. Các phiên bản của Openstack

STT	Phiên bản Openstack	Tình trạng	Thời gian phát hành	Giai đoạn tiếp theo	Ngày chấm dứt
1	Austin	Chấm dứt	21/10/2010		
2	Bexar	Chấm dứt	03/02/2011		
3	Cactus	Chấm dứt	15/04/2011		
4	Diablo	Chấm dứt	22/09/2011		06/05/2013
5	Essex	Chấm dứt	05/04/2012		06/05/2013
6	Folsom	Chấm dứt	27/09/2012		19/11/2013
7	Grizzly	Chấm dứt	04/04/2013		29/03/2014
8	Havana	Chấm dứt	17/10/2013		30/09/2014
9	Icehouse	Chấm dứt	17/04/2014		02/07/2015
10	Juno	Chấm dứt	16/10/2014		07/12/2015
11	Kilo	Chấm dứt	30/04/2015		02/05/2016
12	Liberty	Chấm dứt	15/10/2015		17/11/2016
13	Mitaka	Chấm dứt	07/04/2016		10/04/2017
14	Newton	Chấm dứt	06/10/2016		25/10/2017
15	Ocata	Hỗ trợ	22/02/2017	Ước tính không xác định	
16	Pike	Hỗ trợ	30/08/2017	Ước tính không xác định	
17	Queens	Duy trì	28/02/2018	Hỗ trợ đến 25/10/2019	
18	Rocky	Duy trì	30/08/2018	Hỗ trợ đến 24/02/2020	
19	Stein	Duy trì	10/04/2019	Hỗ trợ đến 10/10/2020	
20	Train	Duy trì	16/10/2019	Hỗ trợ đến 16/04/2021	
21	Ussuri	Phát triển	ước tính 13/05/2020	Duy trì ước tính 13/05/2020	

2. Cấu trúc dịch vụ:

Kiến trúc các module của Openstack ứng với các tên gọi của từng dịch vụ được cung cấp: [5]

Bảng 2-3. Các dịch vụ của Openstack

STT	Tên dịch vụ	Phân loại dịch vụ
1	Block storage (Cinder)	Chính
2	Compute (Nova)	Chính
3	Dashboard (Horizon)	Chính
4	Identity (Keystone)	Chính
5	Image (Glance)	Chính
6	Networking (Neutron)	Chính
7	Object storage (Swift)	Chính
8	Application Catalog (Murano)	Mở rộng
9	Backup, Restore and Disaster Recovery (Freezer)	Mở rộng
10	Bare Metal (Ironic)	Mở rộng
11	Clustering (Senlin)	Mở rộng
12	Container Infrastructure Management (Magnum)	Mở rộng
13	Containers (Zun)	Mở rộng
14	Data Processing (Sahara)	Mở rộng
15	Data Protection Orchestration (Karbor)	Mở rộng
16	Database (Trove)	Mở rộng
17	DNS (Designate)	Mở rộng
18	EC2 API compatibility	Mở rộng
19	Governance (Congress)	Mở rộng
20	Infrastructure Optimization (Watcher)	Mở rộng
21	Key Manager (Barbican)	Mở rộng
22	Load-balancer (Octavia)	Mở rộng
23	Messaging (Zaqar)	Mở rộng
24	Networking automation across Neutron (Tricircle)	Mở rộng
25	NFV Orchestration (Tacker)	Mở rộng
26	Orchestration (Heat)	Mở rộng
27	Placement (Placement)	Mở rộng
28	RCA (Root Cause Analysis) (Vitrage)	Mở rộng
29	Resource reservation (Blazar)	Mở rộng
30	Search (Searchlight)	Mở rộng
31	Shared File Systems (Manila)	Mở rộng
32	Software Development Lifecycle Automation (Solum)	Mở rộng

STT	Tên dịch vụ	Phân loại dịch vụ
33	Telemetry Alarming (Aodh)	Mở rộng
34	Telemetry Data Collection (Ceilometer)	Mở rộng
35	Telemetry Event (Panko)	Mở rộng
36	Workflow (Mistral)	Mở rộng

3. Các module chính được cung cấp trong Openstack:

a. Openstack identity module

OpenStack Identity (Keystone) cung cấp một danh mục các user được ánh xạ tới các dịch vụ Openstack để người dùng có thể truy nhập. OpenStack Identity hoạt động như một hệ thống xác thực chung trên toàn bộ hệ thống và có thể tích hợp với các dịch vụ có danh mục phụ trợ hiện có như Lightweight Directory Access Protocol (LDAP) hay Pluggable authentication module (PAM). OpenStack Identity hỗ trợ nhiều hình thức xác thực bao gồm thông tin đăng nhập tên người dùng và mật khẩu tiêu chuẩn, hệ thống token và phương thức truy nhập của AWS (tức là Amazon Web Services). Ngoài ra, OpenStack Identity còn cung cấp một danh sách có thể truy vấn của tất cả các dịch vụ được triển khai trên OpenStack trong một khởi tạo bình thường. Người dùng và các công cụ của bên thứ ba có thể xác lập tài nguyên được cấp quyền có thể truy cập thông qua OpenStack Identity. [1] [6]

b. Openstack compute module

OpenStack Compute (Nova) là một module điều khiển Cloud computing, là phần chính của hệ thống Openstack được phát triển theo mô hình dịch vụ *Infrastructure as a Service (IaaS)*. OpenStack Compute được thiết kế để quản lý và tự động tối ưu các tài nguyên máy tính cũng như có thể hoạt động với sự mở rộng các công nghệ ảo hóa có sẵn bao gồm các cấu hình của bare-metal server hay cấu hình của siêu máy tính. KVM, VMware và Xen là những lựa chọn có sẵn sử dụng công nghệ hypanneror (màn hình máy ảo), cùng với công nghệ Hyper-V và Linux container là LXC . [1]

OpenStack Compute phân tán các tác vụ hoạt động độc lập và riêng biệt như sau: [6]

Bảng 2-4. Các API trong Openstack Compute (Nova)

STT	API	Chức năng
1	nova-api	Tương tác giữa các API của hệ thống với người dùng
2	nova-compute	Cho phép người dùng có thể tạo và thực thi các VM instance với các API hypervisor (Libvirt KVM, Vmware API của Vmware)
3	nova-network	Cho phép người dùng quản lý các tác vụ liên quan đế mạng
4	nova-scheduler	Cho phép tối ưu các hoạt động của VM instance
5	nova-conductor	Cho phép người dùng truy nhập vào các node thông qua database

c. Openstack network module

OpenStack Networking (Neutron) có chức năng quản lý mạng và địa chỉ IP. OpenStack Networking đảm bảo mạng không bị tắc nghẽn hoặc thắt cổ chai trong khi triển khai Cloud và cung cấp cho người dùng khả năng cấu hình nội bộ và cấu hình qua mạng Internet. [1]

OpenStack Networking cung cấp các mô hình mạng cho các ứng dụng hoặc nhóm người dùng khác nhau. Các mô hình tiêu chuẩn bao gồm các flat network hoặc VLAN để phân tách các server với nhau và lưu lượng truyền dẫn. [1]

OpenStack Networking quản lý địa chỉ IP, hỗ trợ cả địa chỉ IP tĩnh hoặc địa chỉ IP động. Địa chỉ Floating IP cho phép lưu lượng truy cập được định tuyến lại một cách linh hoạt bất kỳ tài nguyên nào trong cơ sở hạ tầng, do đó người dùng có thể chuyển hướng lưu lượng trong quá trình bảo trì hoặc trong trường hợp xảy ra lỗi. [1]

Người dùng có thể tạo các mạng nội bộ, điều khiển lưu lượng, thiết lập kết nối tới các server và các thiết bị trong một hoặc nhiều mạng. Quản trị viên có thể sử dụng các công nghệ software-defined networking (SDN) như OpenFlow để hỗ trợ tối đa multi-tenancy và triển khai quy mô rộng. OpenStack Networking cung cấp một framework mở rộng có thể triển khai và quản lý các dịch vụ mạng thêm vào như hệ thống phát hiện xâm nhập (IDS), cân bằng tải, tường lửa và mạng riêng ảo (VPN). [1]

d. Openstack storage module

Openstack storage có 2 loại lưu trữ là Block storage (Cinder) và Object storage (Swift)

- *OpenStack Block Storage (Cinder)*

OpenStack Block Storage (Cinder) là một hệ thống lưu trữ block-level để sử dụng với các OpenStack compute instance. Hệ thống block storage quản lý việc tạo, gắn và tách các khối thiết bị trên các server. Các phân vùng block storage được tích hợp hoàn toàn vào OpenStack Compute và Dashboard cho phép người dùng cloud quản lý lưu trữ cần thiết của người dùng. Ngoài lưu trữ trên server Linux cục bộ, block storage có thể sử dụng các nền tảng lưu trữ bao gồm Ceph, CloudByte, Coraid, EMC (ScaleIO, VMAX, VNX and XtremIO), GlusterFS, Hitachi Data Systems, IBM Storage (IBM DS8000, Storwize family, SAN Volume Controller, XIV Storage System, and GPFS), Linux LIO, NetApp, Nexenta, Nimble Storage, Scality, SolidFire, HP (StoreVirtual, 3PAR StoreServ families) và Pure Storage. Block storage cũng được sử dụng cho các trường hợp phức tạp liên quan tới hiệu suất như lưu trữ cơ sở dữ liệu, hệ thống file mở rộng hoặc cung cấp cho server quyền truy cập vào block-level storage. Sự quản lý snapshot cung cấp hiệu quả chức năng để sao lưu dữ liệu được lưu trữ trên phân vùng block storage. Snapshot có thể được khôi phục hoặc tạo mới một phân vùng block storage. [1]

- *OpenStack Object Storage (Swift)*

OpenStack Object Storage (Swift) là một hệ thống lưu trữ dự phòng có thể mở rộng. Các object và file được ghi trên nhiều ổ đĩa trải đều các server trong data center với phần mềm OpenStack chịu trách nhiệm đảm bảo sao chép và toàn vẹn dữ liệu thông qua cluster. Các cluster lưu trữ phân bố đều khi thêm các server mới. Nếu server hoặc ổ cứng bị lỗi, OpenStack sẽ sao chép nội dung của nó từ các node hoạt động khác sang các vị trí mới trong cluster. Vì OpenStack sử dụng tính logic trong phần mềm để đảm bảo sao chép và phân tán dữ liệu trên các thiết bị khác nhau nên ổ cứng và server được sử dụng không cần đắt tiền. [1]

Vào tháng 8 năm 2009, Rackspace đã bắt đầu phát triển OpenStack Object Storage như một sự thay thế hoàn toàn cho sản phẩm Cloud Files. Nhóm phát triển ban đầu bao gồm chín nhà phát triển. SwiftStack , một công ty phần mềm object storage,

hiện là nhà phát triển hàng đầu cho Swift với những đóng góp đáng kể từ HP, Red Hat, NTT, NEC, IBM…. [1]

e. Openstack image module

OpenStack Image (Glance) cung cấp dịch vụ trải nghiêm, tạo lập và cho phép sử dụng các image (ổ đĩa ảo). Các image lưu trữ được sử dụng như một template. OpenStack Image cũng có thể được sử dụng để lưu trữ và lập danh mục không giới hạn số lần sao lưu. Image Service có thể lưu trữ image trong nhiều loại back-end, bao gồm Swift. Image Service API cung cấp giao diện REST tiêu chuẩn để truy vấn thông tin về image ổ đĩa và cho phép các client tải các image sang server mới. [1]

OpenStack Image thêm nhiều cải tiến cho cơ sở hạ tầng truyền thống. Nếu được tích hợp với VMware, OpenStack Image giới thiệu các tính năng nâng cao cho tập các vSphere như vMotion, tính sẵn sàng cao và lập lịch tài nguyên động (DRS). vMotion là một công nghệ cho phép di chuyển trực tiếp một VM đang chạy, từ server vật lý này sang server vật lý khác mà không bị gián đoạn dịch vụ. Do đó, OpenStack Image cho phép một datacenter tự tối ưu việc tự động và điều phối, cho phép bảo trì phần cứng cho các server hoạt động kém hiệu suất mà không bị gián đoạn. [1]

Các module OpenStack khác cần tương tác với các image như Heat, phải giao tiếp với images metadata thông qua Glance. Ngoài ra, Nova có thể tiếp nhận thông tin về các image và sự thay đổi cấu hình trên image để tạo ra một instance. Tuy nhiên, Glance là module duy nhất có thể thêm, xóa, chia sẻ hoặc sao chép image. [1]

f. Openstack dashboard module

OpenStack Dashboard (Horizon) cung cấp cho quản trị viên và người dùng giao diện đồ họa để truy cập, cung cấp và triển khai tự động các tài nguyên cloud-based. Mô hình chứa các sản phẩm và dịch vụ của bên thứ ba như thanh toán, giám sát và các công cụ quản lý bổ sung. OpenStack Dashboard cũng có khả năng tạo sự khác biệt trong cách sử dụng cho các nhà cung cấp dịch vụ và các nhà cung cấp thương mại khác. OpenStack Dashboard là một trong các cách người dùng có thể tương tác với tài nguyên OpenStack. Các nhà phát triển có thể tự động truy cập hoặc xây dựng các công cụ để quản lý tài nguyên bằng API OpenStack gốc hoặc API tương thích EC2. [1]

4. Các thành phần chức năng chính của Openstack

Dựa trên các dịch vụ chính, Openstack đưa ra mô tả chi tiết các thành phần chức năng như sau:

Controller node là một node dùng để cài đặt hầu hết các dịch vụ liên quan đến quản trị, xác thực của Openstack cũng như các dịch vụ quản lý database cần thiết liên quan đến các image và các máy ảo cho hay là một control plane trong môi trường Openstack. *Controller node* chứa các module Keystone, Glance và Horizon.

Compute node là một node dùng đề cài đặt các dịch vụ quản lý các máy ảo. *Compute node* chứa module Nova.

Network node là một node dùng để cài đặt các dịch vụ quản lý đến hệ thống mạng và địa chỉ IP trong Openstack. *Network node* chứa module Neutron.

Storage node là một node dùng để cài đặt các dịch vụ liên quan đến quản lý lưu trữ các image, các máy ảo cũng như các file trong Openstack. *Storage node* chứa Cinder hoặc Swift hoặc cả Cinder lẫn Swift.

CHƯƠNG 3. TRIỂN KHAI CÀI ĐẶT HỆ THỐNG PRIVATE CLOUD CHO CÁC ỨNG DỤNG ĐÀO TẠO VÀ THỰC HÀNH DỰA TRÊN GIẢI PHÁP MÃ NGUỒN MỞ OPENSTACK

Chương này sẽ trình bày cách thức triển khai hệ thống Private cloud cho các ứng dụng đào tạo và thực hành dựa trên Openstack. Cụ thể chương này sẽ trình bày quy trình triển khai hệ thống Openstack. Trong quá trình cài đặt triển khai, các bài toàn phát sinh cần được giải quyết để tối ưu hóa hệ thống gồm: Khảo sát hệ thống phần cứng sẽ triển khai, quy hoạch và thiết lập mô hình cài đặt và triển khai, quản trị hệ thống.

Hệ thống phần cứng dùng để cài đặt triển khai trên cụm ba server vật lý và mỗi server được cài đặt hệ điều hành CentOS7. Tên mỗi máy vật lý cấu hình cho 2 card mạng gồm: một card mạng thiết lập kết nối ra Internet để quản trị viên có thể kết nối từng dịch vụ trong hệ thống và một card mạng thiết lập kết nối nội bộ (LAN) các máy với nhau thông qua các switch vật lý.

Mong muốn sau khi triển khai cài đặt Openstack, các server hoạt động với hiệu suất cao nhất do sự tối ưu tài nguyên của hệ thống mang lại. Ngoài ra, hệ thống cũng cho phép quản trị viên có thể tạo lập cơ chế quản trị một cách hiệu quả cũng như thiết lập kết nối từ bên ngoài đến từng máy ảo (hay cụm máy ảo) hoặc các kết nối các máy ảo với nhau.

I. Hệ thống phần cứng hiện có

Hệ thống phần cứng dùng để triển khai Openstack gồm ba server được cài đặt hệ điều hành CentOS7 được cấu hình sẵn để có thể kết nối tới nhau. Ngoài ra, các server này còn được cấu hình để có thể remote hay ssh với nhau. Mỗi server sẽ được mở cổng để cho phép hệ thống có thể kết nối với Internet nhằm giúp quản trị viên có thể quản trị hệ thống ngay cả khi không ở gần các server.

Để thực hiện triển khai giải pháp Private Cloud, hệ thống server được sử dụng bao gồm:

- Bảng các server, và số lượng:

STT	Tên server	Số lượng
1	Server Dell PowerEdge R540 Bronze 3106	1
2	Server Dell PowerEdge R740 Bronze 3106	1
3	Server HP DL380 G9 CTO E5-2630v4	1

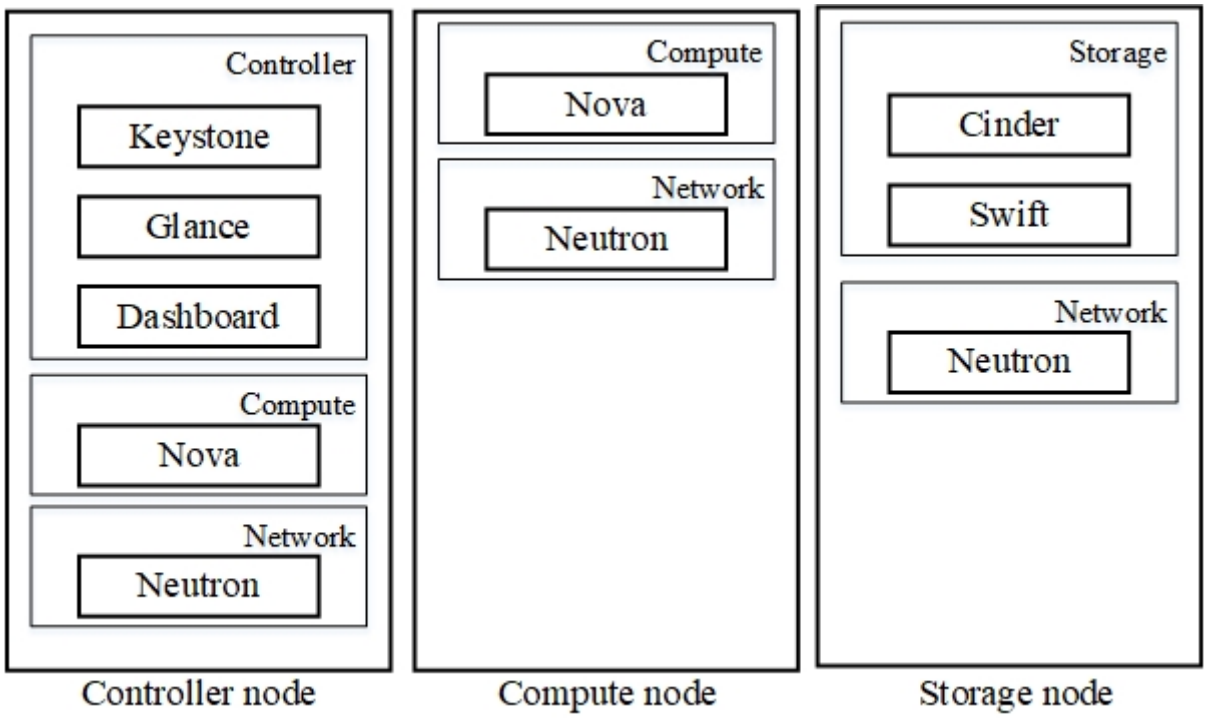
- Bảng cấu hình server (số CPU, RAM, Storage và Card mạng)

STT	Tên server	CPU	Core	RAM (GB)	Storage (TB)	Card mạng
1	Server Dell PowerEdge R540 Bronze 3106	Intel® Xeon® Bronze 3106	64	128	20	2
2	Server Dell PowerEdge R740 Bronze 3106	Intel® Xeon® Bronze 3106	64	128	20	2
3	Server HP DL380 G9 CTO E5-2630v4	Intel® Xeon® E5-2630 v4	64	128	20	2

II. Bài toán quy hoạch máy chủ

1. Mô hình triển khai tham chiếu

Theo yêu cầu của giải pháp Openstack, một mô hình Private Cloud sử dụng Openstack cần có: (1) 01 Controller node; (2) nhiều Compute node; (3) có thể có thêm các Storage node hoặc các thành phần phụ trợ khác. Trong đó, Controller node chịu trách nhiệm quản lý điều phối hoạt động của các Compute node và các thành phần khác phục vụ các tác vụ quản lý máy ảo (tạo, hủy, cấu hình máy ảo), quản lý network ảo. Còn các Compute node là nơi thực hiện việc tạo các máy ảo, host các máy ảo, cung cấp tài nguyên cho các máy ảo hoạt động. Ngoài ra, Storage node chịu trách nhiệm cấp phát tài nguyên để lưu trữ (các máy ảo, file chia sẻ)



Hình 3-1: Mô hình triển khai tham chiếu của Openstack

Các dịch vụ chính được cài đặt trong *Controller node* gồm:

Bảng 3-1: Các dịch vụ cài đặt trong *Controller node*

STT	Các module	Chức năng
1	Keystone	Là dịch vụ dùng để quản lý việc xác thực người dùng khi truy cập và sử dụng các dịch vụ của Openstack
2	Glance	Là dịch vụ dùng để quản lý image liên quan đến ổ đĩa ảo và cấu hình cài đặt trên các ổ đĩa ảo khi sử dụng Openstack
3	Dashboard	Là dịch vụ dùng để hỗ trợ người dùng quản lý hệ thống Openstack thông qua giao diện đồ họa
4	Nova (nova-api, nova-scheduler, nova-conductor)	Là dịch vụ dùng để quản lý việc truy xuất cơ sở dữ liệu và phục vụ các tác vụ liên quan tới máy ảo
5	Neutron (neutron-server, neutron-agent)	Là dịch vụ dùng để quản lý kết nối mạng giữa Controller node với các node khác trong Openstack

Các dịch vụ chính được cài đặt trong *Compute node* gồm:

Bảng 3-2: Các dịch vụ trong *Compute node*

STT	Các module	Chức năng
1	Nova (nova-compute)	Là dịch vụ dùng để quản lý các máy ảo cũng như cập nhật trạng thái của các máy ảo trong hệ thống Openstack
2	Neutron (neutron-agent)	Là dịch vụ dùng để quản lý kết nối mạng giữa Compute node với các node khác trong Openstack

Các dịch vụ chính được cài đặt trong *Storge node* gồm:

Bảng 3-3: Các dịch vụ trong *Storage node*

STT	Các module	Chức năng
1	Cinder	Là dịch vụ dùng để quản lý các thiết bị lưu trữ khối và cung cấp cho người dùng các API tự phục vụ theo yêu cầu và nhu cầu sử dụng
2	Swift	Là dịch vụ dùng để lưu trữ dự phòng và truy xuất dữ liệu
3	Neutron (neutron-agent)	Là dịch vụ dùng để quản lý kết nối mạng giữa Storge node với các node khác trong Openstack

Tuy nhiên, khi triển khai hệ thống Openstack, *Storage node* có thể được cài đặt chung với *Compute node*. Đồng thời, *Controller node* cũng cung cấp dịch vụ như SQL Database để lưu trữ thông tin về hệ thống, MQ (đặc biệt là RabbitMQ) để trao đổi thông tin với *Compute node* hay NTP (Network Time Protocol) để đồng bộ thời gian giữa các máy ảo với máy host. Ngoài ra, *Compute node* cũng cung cấp các dịch vụ tường lửa để đảm bảo an toàn cho các máy ảo. [5]

2. Bài toán quy hoạch máy chủ

Dựa theo mô hình triển khai tham chiếu đã nêu ở trên cũng như thông tin về các server nêu ở Mục I CHƯƠNG 3, chúng tôi thấy rằng việc bố trí cài đặt các thành phần chức năng (controller node, compute node,…) vào server nào với hình thức và phương pháp nào là một vấn đề quan trọng ảnh hưởng tới năng lực xử lý của hệ thống và hiệu suất sử dụng tài nguyên. Để minh họa sự ảnh hưởng của việc bố tri cài đặt controller node, compute node lên server vật lý tới năng lực xử lý và hiệu suất sử dụng tài nguyên, chúng ta xem xét các mô hình triển khai thông thường sau:

PhyComp-PhyCon (Physical Compute and Physical Controller): Trong mô hình này, các nút chức năng được triển khai trên các máy vật lý tách biệt nhau. Tại mỗi một máy vật lý chỉ có duy nhất một chức năng được thực hiện cài đặt. Như vậy với mô hình triển khai này, trong ba máy vật lý hiện có thì một máy sẽ được triển khai *Controller node*, hai máy còn lại sẽ được cài đặt *Compute node*. Mô hình này đơn giản, dễ cài đặt, việc cài đặt *Controller node* và *Compute node* chỉ cần tuân thủ đúng theo hướng dẫn cài đặt của tài liệu tham khảo Openstack. Việc cài đặt tách biệt trên các máy vật lý giúp cho tài nguyên quản lý, điều phối của hệ thống Cloud (sử dụng bởi Controller node) và tài nguyên cung cấp cho các máy ảo người dùng (quản lý bởi Compute node) là tuyệt đối tách biệt, không xung đột và gây ảnh hưởng lẫn nhau. Điều đó khiến hệ thống hoạt động ổn định.

Tuy nhiên, nếu triển khai theo mô hình ***PhyComp-PhyCon***, tài nguyên phần cứng của hệ thống sẽ bị lãng phí và không hiệu quả. *Controller node* sẽ được cài đặt lên server có cấu hình 64 core và 128GB RAM. Như vậy, *Controller node* sẽ được toàn quyền sử dụng 64 core và 128GB RAM. Tuy nhiên, để triển khai *Controller node* chỉ cần sử dụng 1-2 core và 2-4GB RAM. Ngoài ra, với cấu hình server khác tương tự, server sẽ được cài đặt một *Compute node* (chiếm 2-4 core và 4-8GB RAM) và khoảng 60 máy ảo (mỗi máy ảo sử dụng 1 core và 2GB RAM). Thêm nữa, khi cài đặt *Controller*

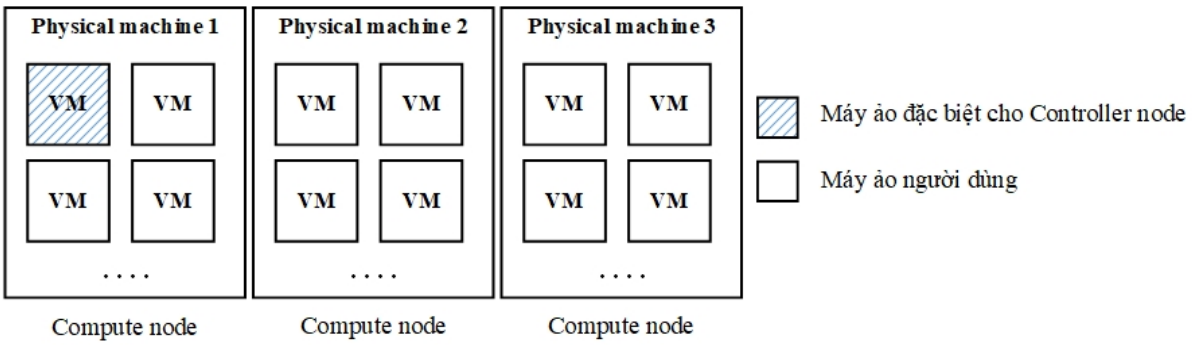
node lên một server tách biệt chỉ cần tối thiểu 10GB để lưu trữ dữ liệu trong khi theo cấu hình server hiện có, dung lượng ổ cứng của server là 20TB.

PhyComp-CoPhyCon (Physical Compute and Colocatable Physical Controller): Trong mô hình này, các nút chức năng được triển khai trên cùng một máy vật lý. Trong số các máy vật lý có duy nhất một máy được cài đặt đầy đủ các chức năng. Như vậy với mô hình triển khai này, trong ba máy vật lý hiện có thì một máy sẽ được triển khai đồng thời *Controller node* và *Compute node*, hai máy còn lại sẽ chỉ được cài đặt *Compute node*. Mô hình này dễ tiếp cận trong giai đoạn đầu của việc triển khai thử nghiêm thực tế lần đầu, việc cài đặt *Controller node* và *Compute node* trên một máy sẽ không đòi hỏi nhiều trong việc chuẩn bị tài nguyên. Việc cài đặt chung các chức năng trên cùng một máy vật lý còn giúp cho hệ thống khắc phục, giải quyết được sự lãng phí về tài nguyên cung cấp cho nhu cầu xử lý (CPU, RAM, Core) và lưu trữ (của hệ thống, việc cài đặt *Controller node* và *Compute node* trên cùng một máy sẽ sử dụng tối đa tài nguyên của hệ thống.

Tuy nhiên, mô hình ***PhyComp-CoPhyCon*** là mô hình khó cấu hình, việc cấu hình *Controller node* và *Compute node* trên cùng một máy vật lý có thể dẫn tới việc xung đột. Việc cài đặt chung trên máy vật lý làm cho tài nguyên quản lý, điều phối của hệ thống Cloud (sử dụng bởi Controller node) và tài nguyên cung cấp cho các máy ảo người dùng (quản lý bởi Compute node) có thể ảnh hưởng lẫn nhau do không được tách biệt và gây ảnh hưởng tới hiệu suất hoạt động của hệ thống. Như vậy, với máy server cài chung chức năng *Controller node* và *Compute node*, hệ thống sẽ ưu tiên cấp phát tài nguyên cho *Controller node* nhằm đảm bảo nhu cầu hoạt động. Tuy nhiên, tài nguyên CPU và RAM cấp phát cho các máy ảo trong *Compute node* sẽ bị chia sẻ dẫn tới hiệu suất hoạt động của các máy ảo bị giảm sút hoặc có thể một số máy ảo vì không được cung cấp tối thiểu CPU và RAM dẫn tới bị treo hệ thống cài đặt trên các máy ảo. Tương tự, khi số lượng máy ảo lớn cần hoạt động thì hệ thống có thể sẽ ưu tiên cấp phát tài nguyên như RAM và CPU cho các máy ảo trong *Compute node* hoạt động ổn định. Như vậy, tài nguyên CPU và RAM cấp phát *Controller node* sẽ bị chia sẻ hoặc khi cần thêm tài nguyên cho nhu cầu xử lý các tác vụ, hệ thống sẽ không cấp phát thêm làm giảm hiệu suất hoạt động của toàn bộ hệ thống.

Qua hai mô hình vừa trình bày bên trên, chúng ta thấy các thành phần chức năng được bố trí vào server và phương thức triển khai là một vấn đề quan trọng và là một thách thức trong bài toán quy hoạch máy chủ sao cho hệ thống hoạt động tối ưu về năng lực xử lý và hiệu suất sử dụng. Để giải quyết vấn đề tận dụng tối đa tài nguyên phần cứng của các máy chủ vật lý nhưng vẫn tách biệt được tài nguyên dùng cho quản lý hệ thống Cloud (*Controllller node*) và tài nguyên cho các máy ảo người dùng (*Compute node*), chúng tôi đề xuất mô hình ***PhyComp-VirCon*** trong đó *Controller node* được triển khai vào một máy ảo dành riêng ký sinh trên một máy vật lý có cài *Compute node*. Cụ thể mô hình ***PhyComp-VirCon*** được mô tả như dưới đây.

PhyComp-VirCon (Physical Compute and Virtual Controller): Trong mô hình này, một máy ảo đặc biệt được thiết lập trước, tách biệt tài nguyên với phần còn lại của một trong ba máy vật lý. *Compute node* sẽ được cài đặt trên ba máy vật lý. Trên máy ảo đặc biệt, *Controller node* được cài đặt. Như vậy với mô hình triển khai này, các nút chức năng sẽ được cài đặt tách biệt nhau. Mô hình này dễ cài đặt các chức năng, việc cài đặt *Controller node* và *Compute node* chỉ cần tuân thủ đúng theo hướng dẫn cài đặt của tài liệu tham khảo Openstack. Việc cài đặt tách biệt trên các máy vật lý giúp cho tài nguyên quản lý, điều phối của hệ thống Cloud (sử dụng bởi Controller node) và tài nguyên cung cấp cho các máy ảo người dùng (quản lý bởi Compute node) là tuyệt đối tách biệt, không xung đột và gây ảnh hưởng lẫn nhau. Điều đó khiến hệ thống hoạt động ổn định. Điểm khác biệt duy nhất giữa mô hình ***PhyComp-PhyCon*** và ***PhyComp-VirCon*** là *Controller Node* được cài lên một server ảo. Việc cấu hình tài nguyên cho máy ảo khá đơn giản, nhanh chóng và khi cần thiết có thể dễ dàng mở rông. Khi cài đặt máy ảo, quản trị viên sẽ cài đặt máy ảo KVM trên CentOS7, cấu hình cho KVM tối thiểu là 2 core và 4GB RAM và dùng libvirt để điều khiển máy ảo KVM. Khi cần tăng core và RAM để xử lý hệ thống, quản trị viên sẽ sửa file chứa cấu hình và khởi động lại máy ảo. Thêm vào đó, các máy vật lý được cài đặt trên các máy vật lý sẽ tận dụng toàn bộ tài nguyên phần cứng hiện có (CPU, RAM, Core và Storage).



Hình 3-2.Mô hình PhyComp-VirCon

Tuy nhiên, trong quá trình nghiên cứu, chúng tôi cũng thấy được những hạn chế khi triển khai mô hình này do mô hình phức tạp nên quản trị viên cần là người có kinh nghiêm trong việc quản trị server và hệ thống thực tế, việc cấu hình, kết nối và quản trị hệ thống giữa server ảo với server vật lý cũng như giữa server ảo với các máy ảo cũng là một thách thức cần giải quyết.

Dưới đây là bảng tóm tắt khi triển khai các mô hình quy hoạch máy chủ:

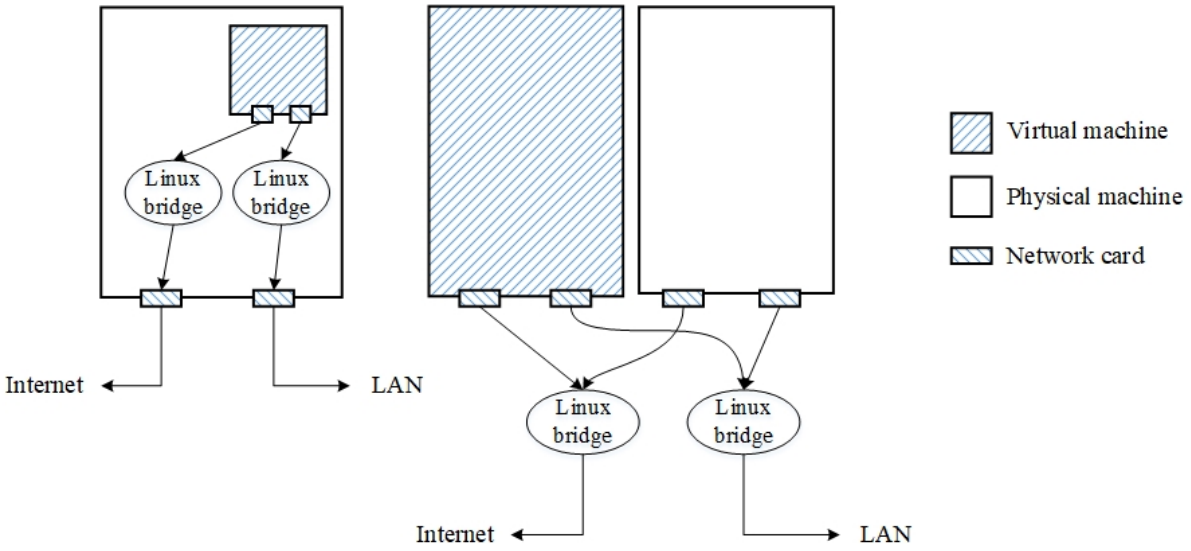
Bảng 3-4. Bảng so sánh các mô hình quy hoạch máy chủ

STT	Mô hình	Ưu điểm	Nhược điểm
1	PhyComp-PhyCon	- Mô hình đơn giản, dễ cài đặt. - Hệ thống sau khi triển khai hoạt động ổn định.	- Gây ra sự lãng phí tài nguyên.
2	PhyComp-CoPhyCon	- Ban đầu dễ tiếp cận hệ thống - Khắc phục sự lãng phí và tối ưu tài nguyên so với mô hình PhyComp-PhyCon.	- Mô hình khó cấu hình cho các nút. - Các nút chức năng dễ xảy ra xung đột và ảnh hưởng tới nhau làm giảm hiệu suất hoạt động.
3	PhyComp-VirCon	- Mô hình tách biệt, các nút chức năng hoạt động độc lập. - Hệ thống sau khi triển khai hoạt động ổn định. - Tối ưu tài nguyên hệ thống. - Dễ dàng mở rộng hệ thống.	- Mô hình phức tạp, quản trị viên cần có kinh nghiệm về quản trị hệ thống. - Khó khăn trong việc quản trị và kết nối mạng đến từng máy ảo.

III. Quy trình triển khai quy hoạch máy chủ theo mô hình *PhyComp-VirCon*

1. Triển khai Openstack trên nền tảng cơ sở hạ tầng sẵn có

Để thực hiện mô hình ***PhyComp-VirCon***, giải pháp chúng tôi đề xuất là tạo một máy ảo bên trong một máy chủ vật lý. Sau đó, triển khai một mạng bridge để kết nối máy ảo ra bên ngoài thông qua máy vật lý chứa nó như minh họa trên *Hình 3-3*. Cách bố trí như vậy đem lại các lợi ích sau. Thứ nhất, máy ảo mặc dù ở bên trong máy vật lý nhưng lại được kết nối mạng ngang hàng với máy vật lý. Điều này khiến việc cài đặt và triển khai chức năng *Controller node* vào máy ảo đơn giản như triển khai *Controller node* vào một máy chủ vật lý độc lập (mô hình ***PhyComp-PhyCon***). Thứ hai, tách biệt *Controller node* vào một máy ảo khiến tài nguyên dành cho điều khiển Private Cloud được tách biệt tối đa với các máy ảo người dùng.



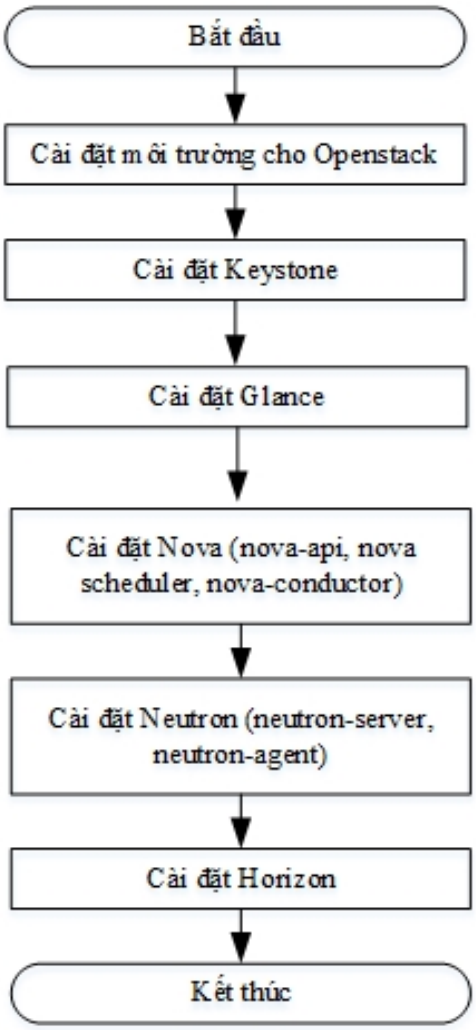
Hình 3-3. Triển khai máy ảo cho Controller node theo mô hình PhyComp-VirCon

Cụ thể mô hình ***PhyComp-VirCon*** của hệ thống thí nghiệm được triển khai như sau. Trên nền hệ điều hành CentOS7 của ba server vật lý, chọn một trong ba máy vật lý, một máy ảo (server ảo) được thiết lập và cấu hình bởi KVM và libvirt được cài đặt lên trên máy vật lý (server vật lý). Máy ảo sẽ được tạo và cấu hình mạng (cấu hình máy ảo ban đầu gồm 2 core, 4GB RAM và 2 card mạng ảo). Việc cài đặt server ảo lên trên server vật lý thiết lập kết nối tương đương với hai máy vật lý kết nối với nhau (*như Hình 3-3*). Server ảo sẽ được thiết lập để kết nối với server vật lý thông qua các switch ảo, cụ thể trong hệ điều hành CentOS7, các switch ảo chính là *Linux bridge* [11] [12]. Server ảo sẽ kết nối với server vật lý qua *Linux bridge* được thiết lập với các kết nối gồm: một kết

nối ra Internet (được gọi là ***Provider network***) và một kết nối nội bộ (được gọi là ***LAN network***). Lúc này, server ảo có đầy đủ chức năng và cấu hình như một server vật lý. Hệ điều hành CentOS7 cũng được cài đặt lên máy ảo vừa tạo và máy ảo sẽ được thiết lập địa chỉ IP để kết nối tới máy vật lý.

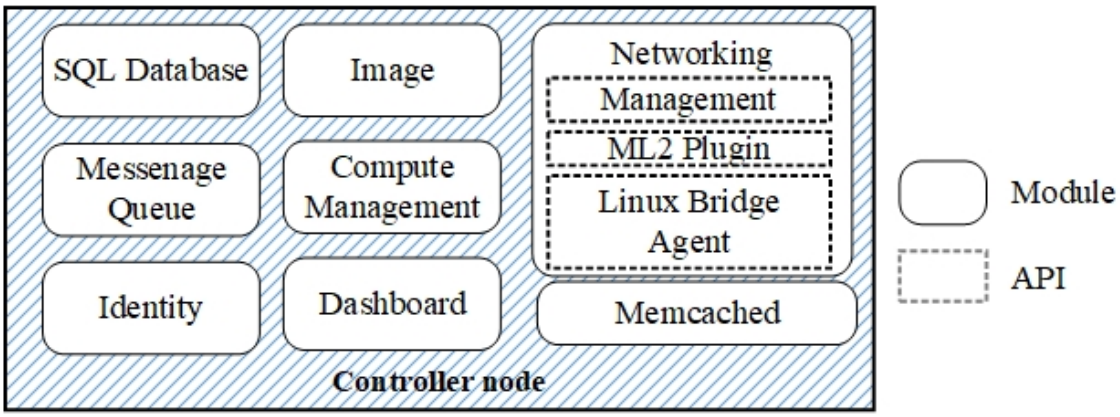
Trong mô hình ***PhyComp-VirCon***, server ảo sẽ được thiết lập để cài đặt làm *Controller node* và server vật lý sẽ được thiết lập để cài đặt làm *Compute node*. Trên server ảo, các module được cài đặt cho *Controller node* bao gồm: OpenStack Identity (Keystone), OpenStack Compute (Nova), OpenStack Networking (Neutron), OpenStack Image (Glance) và OpenStack Dashboard (Horizon). Trên các máy server vật lý, các module được cài đặt cho *Compute node* bao gồm: OpenStack Compute (Nova), OpenStack Networking (Neutron). Dưới đây là chi tiết quy trình triển khai và cài đặt *Controller node* và *Compute node*.

2. Triển khai *Controller node* theo mô hình *PhyComp-VirCon*



Hình 3-4. Sơ đồ quy trình cài đặt *Controller node*

Để triển khai cài đặt *Controller node*, cần cài đặt các môi trường nền phục vụ cho việc triển khai các module của hệ thống. Sau khi cài đặt xong môi trường nền, sẽ tiến hành cài đặt các module theo tuần tự theo tài liệu hướng dẫn của Openstack: Keystone → Glance → Nova → Neutron → Horizon.



Hình 3-5. Các module được triển khai cho *Controller node*

Các phần mềm được cài đặt để tạo môi trường cho Openstack gồm Openstack Repository (phiên bản được sử dụng là phiên bản Openstack Rocky), Openstack Client, Openstack SELinux, SQL Database (trong hệ thống này là MySQL) hay Messenger Queue (là RabbitMQ) và Memcached. Trong quá trình cài đặt các module chính của Openstack việc đầu tiên cần khởi tạo và gán quyền cho các database của “keystone”, “glance”, “nova”, “neutron” trong *Controller node*.

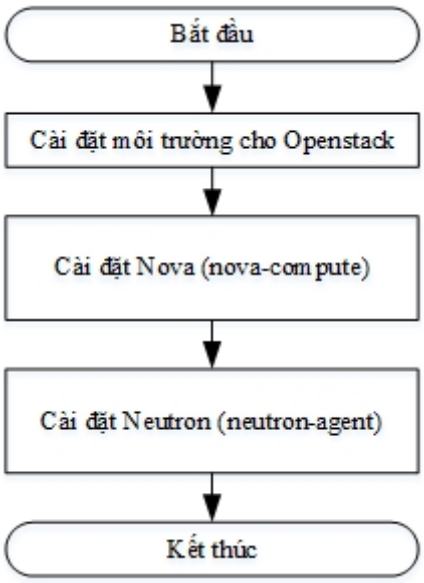
Trong *Controller node*, module đầu tiên là module Keystone. Keystone được cài đặt trong gói phần mềm *openstack-keystone* và được tinh chỉnh các thành phần trong file *keystone.conf* cho phép kết nối với database cũng như cài đặt keystone-manage nhằm khởi tạo và cập nhật dữ liệu trong Keystone để cung cấp dịch vụ định danh. Tiếp theo, cấu hình của Apache HTTP nhằm thiết lập và cung cấp dịch vụ web bằng cách cài đặt gói phần mềm *httpd* và *mod_wsgi*, tinh chỉnh các thành phần trong file *httpd.conf* và chạy dịch vụ *httpd* đồng thời cấu hình tài khoản Admin với file *admin-openrc*. Dịch vụ định danh cung cấp tính năng xác thực cho mọi dịch vụ của Openstack. Tính năng xác thực được dùng kết hợp với các domain, project, user và role. Như vây, các domain, project, user và role có thể được khởi tạo bởi người dùng, bước đầu tiên cần khởi tạo domain và dựa trên cơ sở các domain được khởi tạo, các project và user *Keystone* cũng sẽ được khởi tạo và đồng thời khởi tạo các role. Các role được thêm vào cho các project và các user *Keystone*. Để kết thúc việc cài đặt, Keystone cần được kiểm tra và xác thực

thông tin lưu trữ trong database so với thông tin trong các bước khởi tạo, nếu kết quả trùng khớp thì xác lập việc cài đặt thành công và chuyển sang bước cài đặt tiếp theo. Ngoài môi trường tập lệnh mặc định của Openstack dành cho admin, Openstack cho phép người dùng tạo và cấu hình môi trường tập lệnh Openstack client để tương tác với dịch vụ định danh. Module thứ hai là module Glance. Dựa trên việc tạo user *Glance* và các project *Keystone* sẽ gán tới các role, ngoài ra, các service glance cần được khởi tạo để quản lý các image đồng thời khởi tạo các API endpoint cho Glance. Glance cài đặt gói phần mềm và tinh chỉnh các thành phần trong các file *glance-api.conf, glance-registry.conf* để kết nối với database cũng như cài đặt *glance-manage* để quản lý và cấu hình việc cài đặt Glance với database, tiếp đó cần khởi động lại các Image service. Để kết thúc việc cài đặt, glance cần được kiểm tra và xác nhận việc hoạt động của Image service để phục vụ vấn đề quản lý và lưu trữ chính xác các thông tin liên quan image máy ảo so với thông tin trong các bước khởi tạo, nếu kết quả trùng khớp thì xác lập việc cài đặt thành công và chuyển sang bước cài đặt tiếp theo. Module thứ ba là module Nova. Nova cần được cấu hình theo tài khoản admin với file *admin-openrc*. Khởi tạo các user *Nova* đồng thời thêm các role cho user *Nova* và các project *Keystone*, khởi tạo các service và các API endpoint cho Nova. Bước tiếp sẽ cài đặt gói phần mềm và tinh chỉnh các thành phần trong file cấu hình của Nova (bao gồm Nova API, Nova Scheduler và Nova Conductor) và khi cài đặt và cấu hình xong sẽ khởi động lại các dịch vụ của Nova. Khi các *Compute node* được thiết lập xong *Controller node* cần được quét và cập nhật các danh sách vào database qua service của Nova. Để kết thúc việc cài đặt, Nova cần được kiểm tra và xác thực thông tin lưu trữ trong database so với thông tin trong các bước khởi tạo, nếu kết quả trùng khớp thì xác lập việc cài đặt thành công và chuyển sang bước cài đặt tiếp theo. Module thứ tư là module Neutron. Neutron cũng cần được cấu hình theo tài khoản admin. Khởi tạo các user *Neutron* đồng thời thêm các role cho user *Neutron* và các project *Keystone* cũng như khởi tạo các service và các API endpoint cho Neutron. Sau đó, gói phần mềm được cài đặt và tinh chỉnh các thành phần trong các file *neutron.conf, ml2_conf.ini* của Neutron. Các API được cài đặt và tinh chỉnh gồm API Openstack Neutron và API Openstack Neutron ML2 dành cho Neutron server nhằm cung cấp các dịch vụ liên quan tới Networking API còn API Openstack Neutron Linuxbridge dành cho Neutron Agent nhằm quản lý cấu hình các switch ảo (vswitch). Sau cùng, Neutron service cần được khởi động lại. Để kết thúc việc cài đặt, Neutron

cần được kiểm tra và xác thực thông tin lưu trữ trong database so với thông tin trong các bước khởi tạo, nếu kết quả trùng khớp thì xác lập việc cài đặt thành công và chuyển sang bước cài đặt tiếp theo. Module cuối cùng là module Horizon. Gói phần mềm phục vụ Horizon là *openstack-dashboard* cần được cài đặt, tinh chỉnh các file *local_settings.py* và *openstack-dashboard.conf* theo hướng dẫn tài liệu của Openstack. Sau đó, hệ thống cần khởi động lại web server và session storage service. Để kết thúc việc cài đặt, horizon cần được kiểm tra và xác nhận việc hoạt động bằng việc truy cập qua địa chỉ http://controller/dashboard đồng thời xác thực user và mặc định tên miền.

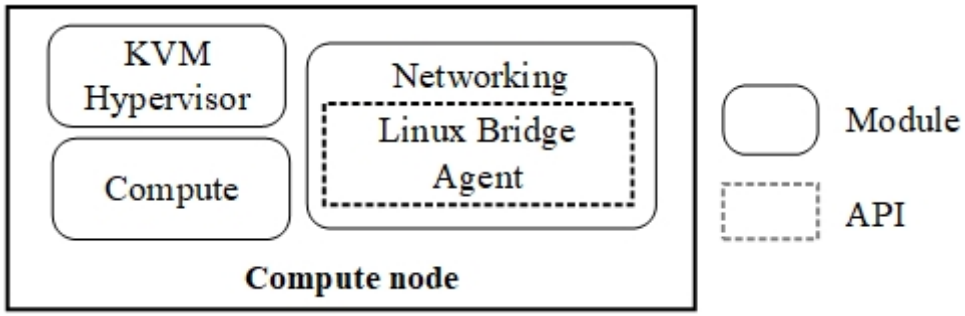
Trong toàn bộ các file cấu hình để cung cấp mã token cần thiết lập thông qua port 11211, để truy nhập xác thực thông qua port 5000 khi kết nối với Keystone. Ngoài ra để truy cập vào Glance để thao tác với image thông qua port 9292.

3. Triển khai *Compute node* theo mô hình *PhyComp-VirCon*



Hình 3-6. Sơ đồ quy trình cài đặt *Compute node*

Để triển khai cài đặt *Compute node*, cần cài đặt các môi trường nền phục vụ cho việc triển khai các module của hệ thống. Sau khi cài đặt xong môi trường nền, sẽ tiến hành cài đặt các module theo tuần tự theo tài liệu hướng dẫn của Openstack: Nova → Neutron.



Hình 3-7. Các module được triển khai cho *Compute node*

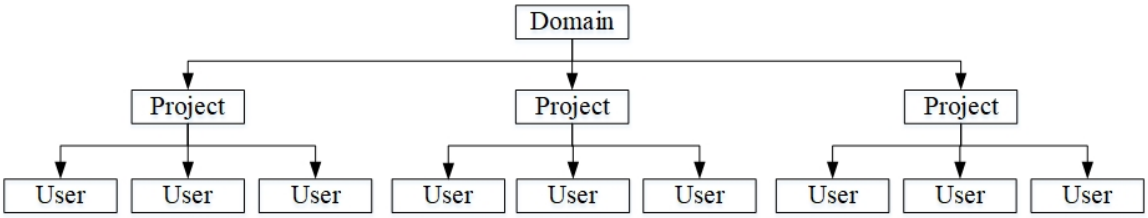
Các phần mềm được cài đặt để tạo môi trường cho Openstack gồm Openstack Repository (phiên bản được sử dụng là phiên bản Openstack Rocky), Openstack Client, Openstack SELinux.

Trong *Compute node*, module đầu tiên là module Nova. Nova được cài đặt trong gói phần mềm *openstack-nova-compute* và được cấu hình trong file *nova.conf* và khi cấu hình xong sẽ khởi động lại các dịch vụ của Nova. Việc quản lý các tác vụ liên quan đến máy ảo sau khi máy ảo (KVM Hypervisor) được cài đặt trên *Compute node* được phụ trách bởi *Controller node*. Openstack cho phép người dùng có thể truy cập từ xa bằng công cụ VNC thông qua port 6080. Module đầu tiên là module Neutron. Các API được cài đặt và tinh chỉnh gồm API Openstack Neutron Linuxbridge dành cho Neutron Agent nhằm quản lý cấu hình các switch ảo (vswitch) phục vụ cho việc kết nối các máy ảo với nhau trong các file *nova.conf*, *neutron.conf* và *linuxbridge_agent.ini*. Sau cùng, Neutron service và Nova service cần được khởi động lại.

Sau khi cài đặt xong các module của Openstack, quản trị viên sẽ quản trị hệ thống qua OpenStack Dashboard (Horizon) thông qua website. Tuy nhiên, quản trị viên cấu hình địa chỉ IP cho website quản trị có thể truy cập theo địa chỉ ứng với tên miền Internet. Sau khi đã cấu hình xong và quản trị viên đăng nhập vào hệ thống qua tài khoản *admin*, quan trị viên tạo ra các máy ảo và được cài sẵn hệ điều hành để người dùng có thể triển khai các ứng dụng (cụ thể là các phòng ban sẽ cài đặt các dịch vụ và ứng dụng liên quan tới đào tạo và thực hành ứng với các nhiệm vụ chức năng như công tác quản lý, giảng dạy và thực hành…). Quản trị viên cũng có thể tạo ra các rule để thiết lập bảo mật cũng như kết nối cho toàn bộ các máy ảo hay từng máy ảo trên Openstack.

IV. Sử dụng Openstack trong quản trị hệ thống Private Cloud cho trường đại học

Hệ thống Openstack được quản trị theo domain. Các domain được quản trị bởi tài khoản *admin* trong domain *Default*. Ngoài ra, mỗi domain có tài khoản quản trị riêng của domain đó và tách biệt độc lập với các domain khác. Khi quản trị một domain, tài khoản quản trị được phép tạo nhiều project và nhiều user khác nhau được phân quyền (role) khác nhau (như admin, user, …) để tương tác với hệ thống Cloud.



Hình 3-8. Mô hình quản trị Openstack

Với việc triển khai hệ thống Private Cloud cùng hệ thống phần cứng hiện có để cung cấp tài nguyên cho các đơn vị trong trường học với mục đích sử dụng các ứng dụng đào tạo và thực hành, vai trò phân cấp chức năng của hệ thống quản trị cũng cần phải tương ứng với mô hình cơ cấu quản lý đào tạo giáo dục của nhà trường.

Mỗi domain được vận hành tương ứng với một cloud dùng để phục vụ cho các đơn vị trong nhà trường như khối hành chính, các khoa – viện – bộ môn. Mỗi domain cung cấp dịch vụ cho phép triển khai công tác nghiệp vụ, quản lý cũng như học tập giảng dạy của các đơn vị trong nhà trường. Mỗi domain hoạt động tách biệt với nhau và được kiểm soát bởi một tài khoản quản trị.

Mỗi project được vận hành tương ứng với cụm server được dùng cho từng phòng ban trong khối hành chính như phòng Đào tạo, phòng Kế toán… hay các dự án của khoa – viện, các môn học trong từng bộ môn. Mỗi project được quản trị nhằm phục vụ cho việc triển khai các hệ thống đặc thù của từng đơn vị. Mỗi domain hoạt động tách biệt với nhau và được sử dụng bởi một hay nhiều user với các quyền khác nhau.

Việc tạo các role và user trong hệ thống đáp ứng được nhu cầu phân cấp người dùng, cụ thể như trong một môn học, giảng viên có thể tạo ra các hệ thống thực nghiệm (gồm những máy ảo, cụm máy ảo đã được thiết lập sẵn môi trường và điều kiện thực hành, nghiên cứu) là các bài tập cho phép sinh viên, học viên vào để học tập, nghiên cứu theo như yêu cầu môn học mà không cần phải thiết lập lại hệ thống hoặc giảng viên có thể kiểm tra việc thực hành của người học theo yêu cầu hay cho phép học viên, sinh

viên có quyền quản trị, theo dõi, giám sát và thao tác với hệ thống nhỏ (là những máy ảo, cụm máy ảo đã được thiết lập sẵn môi trường và các dịch vụ theo yêu cầu) trong dự án, đề tài do bộ môn, khoa – viện đang triển khai.

Để quản trị hệ thống Openstack, quản trị viên cần tiến hành việc khởi tạo và quản trị với hệ thống lần lượt theo thứ tự domain → project → role → user. Sau khi cài đặt thành công Openstack, quản trị viên sẽ đăng nhập với domain *Default* cùng với username và password là tài khoản được cấu hình trong file *admin-openrc*. Ngoài domain mặc định *Default* để có thể quản trị hệ thống nhằm cung cấp các dịch vụ trong Openstack, quản trị viên có thể khởi tạo domain khác cho hệ thống. Domain được quản trị viên khởi tạo bằng hai cách: Cách đầu tiên là thông qua chạy giao diện dòng lệnh (cli) trong *Controller node*. Với thao tác dùng giao diện dòng lệnh, quản trị viên cần truy cập vào server ảo đã cài đặt *Controller node* và chạy câu lệnh ***openstack domain create --description <ghi chú cho domain> <tên domain>.*** Cách thứ hai là quản trị thông qua Dashboard. Với thao tác khởi tạo domain qua Dashboard, quản trị viên chọn menu Identity → Domain và chọn *Create Domain* đồng thời nhập thông tin về tên, mô tả về domain. Mỗi domain trong Openstack quản lý chung nhiều project. Như vậy, nhiều project khác nhau được khởi tạo bởi một domain. Để khởi tạo một project, quản trị viên cần chọn menu Identity → Projects và chọn *Create Project* đồng thời nhập thông tin về tên, mô tả về project. Ngoài ra, trong quá trình quản trị project, quản trị viên có thể thay đổi hay xóa từng project. Với mỗi project, quản trị viên cũng sẽ tạo và quản lý nhiều user với các role khác nhau. Sau khi khởi tạo project, quản trị viên cần khởi tạo các quyền nhằm gán quyền khi tạo tài khoản người dùng. Role được quản trị viên khởi tạo bằng cách chọn menu Identity → Roles và chọn *Create Role* và nhập thông tin về tên Role. Để khởi tạo một User, quản trị viên cần chọn menu Identity → Users và chọn *Create User* đồng thời nhập thông tin về tên, mô tả, mật khẩu đăng nhập của user, project của user cũng như quyền (role) tương ứng của người dùng. Trong quá trình tạo User, quản trị viên cần biết chính xác Project mà tài khoản người dùng cần khởi tạo cũng như vai trò của người dùng trong Project thông qua role. Sau khi đã khởi tạo một User xong, người dùng sẽ đăng nhập vào Openstack, điền chính xác tên domain được khởi tạo để quản lý user và thông tin tài khoản của người dùng được khởi tạo. Trong quá trình quản trị, quản trị viên có thể thay đổi role cho user cũng như có thể vô hiệu

hóa tài khoản user mà không nhất thiết phải xóa tài khoản user nhằm bảo đảm tính an toàn của hệ thống.

Ngoài việc quản trị theo mô hình phân cấp chức năng của hệ thống Cloud. Openstack còn cung cấp các cơ chế để quản trị theo mô hình hướng người dùng. Người dùng có thể quy hoạch tạo và thay đổi hệ thống các cụm server do bản thân quản lý bằng việc thiết lập hệ thống mạng bao gồm việc kết nối cho từng máy ảo, việc đóng mở các port, các luật nhằm bảo mật kết nối và thiết lập hệ thống máy ảo bao gồm lựa chọn hệ điều hành, cấu hình máy ảo, vị trị máy ảo trong hệ thống.

Trong mỗi tài khoản người dùng, có thể tạo một hay nhiều máy ảo. Người dùng khi được cấp quyền cho phép truy nhập vào các project khác nhau cũng có thể thao tác với các máy ảo được tạo có trong các project đó. Để có thể cài đặt máy ảo, quản trị viên cần khởi tạo source image là các bản cài đặt hệ điều hành cho máy ảo. Để khởi tạo *Image*, quản trị viên chọn menu Project → Compute → Image, chọn Create Image để nhập thông tin cũng như tải file image là các bản cài đặt hệ điều hành và để nên định dạng của image là QCOW2. Để định dạng cấu hình cho việc cấp phát tài nguyên cho máy ảo, quản trị viên cần tạo các *flavor*. Để khởi tạo *Flavor*, quản trị viên chọn menu Admin → Compute → Flavors, chọn Create Flavor và nhập thông tin về tên flavor, cấu hình cho máy ảo (CPU, RAM, dung lượng lưu trữ). Trong quá trình cài đặt máy ảo, quản trị viên cần thiết lập kết nối mạng cho máy ảo. Openstack hỗ trợ quản trị viên việc quy hoạch kết nối hệ thống mạng cho các máy ảo bằng mô hình giao diện đồ họa ở mục Project → Network → Network Topology. Để khởi tạo các kết nối mạng trong hệ thống, quản trị viên chọn menu Project → Network → Networks. Tại đây, quản trị viên có thể thiết lập tên mạng, cấu hình dải mạng (subnet) cho máy ảo hay cụm máy ảo kết nối với nhau. Ngoài ra, khi đã thiết lập các dải mạng, để kết nối chúng với nhau hay tạo lập kết nối với dải mạng cho phép truy cập Internet, quản trị viên tạo và cấu hình cho các *router* qua thao tác chọn menu Project → Network → Routers. Router được thiết lập để kết nối với đường mạng cho phép kết nối với Internet. Nếu muốn kết nối router với đường mạng LAN trong hệ thống, quản trị viên chọn router cần kết nối và chọn Interface → Add Interface và chọn subnet cần kết nối. Để quản trị viên có thể truy cập vào từng máy ảo trong hệ thống, Openstack hỗ trợ việc kết nối ssh với các công cụ như putty, openssh… bằng việc tạo key pair. Để khởi tạo *Key pair*, quản trị viên chọn menu

Compute → Key Pairs, chọn Create Key Pair và nhập tên key pair. Hệ thống sẽ tự động cho phép người dùng tải key pair ngay sau khi khởi tạo. Cuối cùng là thao tác khởi tạo các máy ảo (instance). Để khởi tạo Instance, quản trị viên chọn menu Compute → Instances, chọn Launch Instance và nhập các thông tin về tên cũng như mô tả cho instance, số lượng instance cần tạo, hình thức boot hệ điều hành vào máy ảo, dung lượng ổ ảo, hệ điều hành cần tạo cho instance, loại flavor, dải mạng sẽ kết nối, các cơ chế bảo mật cho instance, key pair (nếu sử dụng ssh). Sau khi tạo instance, quản trị viên có thể kiểm tra thông tin của instance như port ssh của máy ảo, nội dung hoạt động của máy ảo và các thao tác người dùng trên máy ảo, cũng như truy cập vào máy ảo trên nền web qua console.

Thông qua việc khởi tạo các instance (chính là các máy ảo) trong Project một cách đơn giản và nhanh chóng, quản trị viên có thể thiết lập các hệ thống mạng khác nhau gồm các cụm máy ảo nhằm phục vụ việc triển khai cài đặt các ứng dụng hoặc hệ thống phần mềm cụ thể của các đơn vị phòng ban chức năng của Nhà trường. Ngoài ra, các cụm máy ảo do giảng viên thiết lập sinh viên có thể truy cập từ xa để học tập, thực hành các bài tập trên hệ thống các máy ảo đó qua đó giảng viên cũng sẽ tương tác và theo dõi tiến trình học tập thực hành của sinh viên tại từng môn học đòi hỏi nhu cầu thực hành lớn. Ngoài việc sử dụng cho học tập, thực hành và quản lý của các bộ môn và các phòng ban trong Nhà trường, việc thiết lập các máy ảo dùng cho các dự án liên quan đến các khoa – viện đòi hỏi tính linh hoạt cũng như khả năng mở rộng hệ thống. Openstack đã hỗ trợ quản trị viên chuyển đổi các instance một cách thuận tiện như mở rộng cấu hình máy ảo, thiết lập thêm các kết nối mạng, điều chính các chính sách bảo mật trong khi máy ảo vẫn đang hoạt động và không gây gián đoạn hệ thống.

Ngoài việc khởi tạo và thiết lập các instance, Openstack cũng hổ trợ quản trị viên quản lý tài nguyên khi các instance tạm dừng hay hủy bỏ nhằm bảo toàn tài nguyên phần cứng một cách tối đa. Đặc biệt, nhu cầu hủy bỏ các máy ảo dùng cho sinh viên thực hành của giảng viên trong các môn học khác nhau tại từng kỳ học khác nhau.

CHƯƠNG 4. KẾT QUẢ ĐẠT ĐƯỢC VÀ KẾT LUẬN

Trong ngữ cảnh đặc thù của ngành giáo dục nói chung, đại học nói riêng. Các ứng đào tạo và thực hành là các ứng dụng đặc thù trong giáo dục nên đòi hỏi việc triển khai cho từng đơn vị và đối tượng là khác nhau. Nên việc triển khai và ứng dụng Private Cloud để quy hoạch và tối ưu tài nguyên hệ thống server cho hoạt động quản lý, học tập, giảng dạy là nhu cầu chính đáng. Chúng ta cần xây dựng quy trình triển khai cài đặt hệ thống bao gồm khảo sát, quy hoạch, triển khai hệ thống thực tế cũng như quản trị và kết nối các máy ảo trong hệ thống.

Trong luận văn, chúng tôi đã khảo sát hệ thống phần cứng hiện có dùng để triển khai, xây dựng mô hình triển khai tham chiếu cũng như đưa ra các bài toán quy hoạch máy chủ. Trong quá trình xem xét các mô hình ***PhyComp-PhyCon*** và ***PhyComp-CoPhyCon*** khi triển khai trên các máy chủ vật lý cũng như việc chúng tôi đề xuất mô hình ***PhyComp-VirCon*** để đánh giá năng lực xử lý của hệ thống và hiệu suất sử dụng tài nguyên khi bố trí các khối chức năng vào vị trí cũng như phương thức triển khai trên các server hiện có. Qua so sánh các ưu điểm và nhược điểm của các mô hình, chúng tôi đã lựa chọn mô hình ***PhyComp-VirCon*** là mô hình để giải quyết bài toán quy hoạch máy chủ. Dựa trên mô hình ***PhyComp-VirCon***, chúng tôi đã thực hiện triển khai thực nghiệm máy ảo cho *Controller node* lên trên một trong các máy vật lý sẽ cài đặt *Compute node* cũng như mô tả quy trình cài đặt bằng sơ đồ, mô hình các module sẽ được cài đặt cho *Controller node* và *Compute node* cùng với mô tả chi tiết các bước cài đặt theo tài liệu hướng dẫn của Openstack. Ngoài ra, chúng tôi cũng đã đưa ra quy trình quản trị hệ thống Private Cloud sử dụng Openstack, từ việc mô tả phân cấp chức năng của hệ thống gồm domain, project, user, role cũng như phân tích và áp dụng mô hình phân cấp chức năng váo trong mô hình triển khai các ứng dụng đào tạo và thực hành trong môi trường đại học thông qua việc phân tích quy trình ứng dụng hệ thống Openstack cũng như các thao tác kỹ thuật liên quan tới cài đặt và quản trị hệ thống.

TÀI LIỆU THAM KHẢO

[1] Cloud computing, Virtualization, Hypervisor, Openstack – Wikipedia (wikipedia.org).

[2] What is a private cloud? – Microsoft Azure (azure.microsoft.com).

[3] Mell, P., & Grance, T. (2011). The NIST definition of cloud computing.

[4] Desai, A., Oza, R., Sharma, P., & Patel, B. (2013). Hypervisor: A survey on concepts and taxonomy. International Journal of Innovative Technology and Exploring Engineering, 2(3), 222-225.

[5] Openstack – Openstack (openstack.org).

[6] Khedher, O., & Chowdhury, C. D. (2017). Mastering OpenStack. Packt Publishing Ltd.

[7] Chirammal, H. D., Mukhedkar, P., & Vettathu, A. (2016). Mastering KVM virtualization. Packt Publishing Ltd.

[8] Kumar, R., Jain, K., Maharwal, H., Jain, N., & Dadhich, A. (2014). Apache cloudstack: Open source infrastructure as a service cloud computing platform. Proceedings of the International Journal of advancement in Engineering technology, Management and Applied Science, 111-116.

[9] Cloudstack – Cloudstack (cloudstack.apache.org).

[10] Barkat, A., dos Santos, A. D., & Ho, T. T. N. (2014, September). Open stack and cloud stack: Open source solutions for building public and private clouds. In 2014 16th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (pp. 429-436). IEEE.

[11] Böhme, U., & Buytenhenk, L. (2000). Linux BRIDGE− STP− HOWTO. Revision 0.4. The Linux Documentation Project.

[12] Makita, T. (2014). Virtual switching technologies and linux bridge.

PHỤ LỤC 1: CÁC BƯỚC TRIỂN KHAI OPENSTACK

Các thao tác triển khai cài đặt Openstack:

1. Triển khai Controller node

a. Chuẩn bị

- Cài đặt Openstack Repository (Rocky)

  yum install centos-release-openstack-rocky

- Cài đặt Openstack client

  yum install python-openstackclient

- CentOS enable SELinux by default. Install the openstack-selinux package to automatically manage security policies for OpenStack services

  yum install openstack-selinux

- Cài đặt MySQL ...

  wget http://repo.mysql.com/mysql-community-release-el7-5.noarch.rpm

  rpm -ivh mysql-community-release-el7-5.noarch.rpm

  yum install mysql-server

  yum install mariadb-server

  systemctl start mysqld

  systemctl start mariadb

  mysql_secure_installation

- Cài đặt RabbitMQ

  yum install rabbitmq-server

  systemctl enable rabbitmq-server.service

  systemctl start rabbitmq-server.service

  rabbitmqctl add_user openstack RABBIT_PASS

  rabbitmqctl set_permissions openstack "." ".*" ".*"*

- Cài đặt Memcached

  yum install memcached python-memcached

  OPTIONS="-l 127.0.0.1,::1,controller"

  systemctl enable memcached.service

  systemctl start memcached.service

b. Cài đặt Keystone

- Tạo database và user "keystone". Gán quyền cho keystone.

  CREATE DATABASE keystone;

GRANT ALL PRIVILEGES ON keystone. TO 'keystone'@'localhost' IDENTIFIED BY 'KEYSTONE_DBPASS';*

GRANT ALL PRIVILEGES ON keystone. TO 'keystone'@'%' IDENTIFIED BY 'KEYSTONE_DBPASS';*

- Cài đặt và cấu hình các thành phần

yum install openstack-keystone httpd mod_wsgi

- Sửa file keystone

vi /etc/keystone/keystone.conf

{

[database]

...

connection=mysql+pymysql://keystone:KEYSTONE_DBPASS@controller/keystone

[token]

...

provider=fernet

}

- Cài đặt keystone-manage

su -s /bin/sh -c "keystone-manage db_sync" keystone

keystone-manage fernet_setup --keystone-user keystone --keystone-group keystone

keystone-manage credential_setup --keystone-user keystone --keystone-group keystone

keystone-manage bootstrap --bootstrap-password ADMIN_PASS --bootstrap-admin-url http://controller:5000/v3/ --bootstrap-internal-url http://controller:5000/v3/ --bootstrap-public-url http://controller:5000/v3/ --bootstrap-region-id RegionOne

- Tùy chỉnh Apache HTTP
  - Sửa file httpd

vi /etc/httpd/conf/httpd.conf

{

ServerName controller

}

ln -s /usr/share/keystone/wsgi-keystone.conf /etc/httpd/conf.d/

- Bật service Apache HTTP

  systemctl enable httpd.service

  systemctl start httpd.service

- Config tài khoản ADMIN

  vi admin-openrc

  {

  export OS_USERNAME=admin

  export OS_PASSWORD=ADMIN_PASS

  export OS_PROJECT_NAME=admin

  export OS_USER_DOMAIN_NAME=Default

  export OS_PROJECT_DOMAIN_NAME=Default

  export OS_AUTH_URL=http://controller:5000/v3

  export OS_IDENTITY_API_VERSION=3

  }

  . admin-openrc

- Tạo a domain, projects, users, and roles

  - Tạo domain

    openstack domain create --description "An Example Domain" example

  - Tạo project

    openstack project create --domain default --description "Service Project" service

    openstack project create --domain default --description "Demo Project" myproject

  - Tạo user

    openstack user create --domain default --password-prompt myuser

  - Tạo role

    openstack role create myrole

  - Thêm role “myrole” cho project “myproject” và user “myuser “

    openstack role add --project myproject --user myuser myrole

- Verify operation
  - Loại bỏ giá trị biến OS_AUTH_URL và OS_PASSWORD

    unset OS_AUTH_URL OS_PASSWORD
  - Authentication token

    openstack --os-auth-url http://controller:5000/v3 --os-project-domain-name Default --os-user-domain-name Default --os-project-name admin --os-username admin token issue

    openstack --os-auth-url http://controller:5000/v3 --os-project-domain-name Default --os-user-domain-name Default --os-project-name myproject --os-username myuser token issue
- Tạo OpenStack client environment scripts
  - Config tài khoản DEMO

    vi demo-openrc

    {

    export OS_PROJECT_DOMAIN_NAME=Default

    export OS_USER_DOMAIN_NAME=Default

    export OS_PROJECT_NAME=myproject

    export OS_USERNAME=myuser

    export OS_PASSWORD=MYUSER_PASS

    export OS_AUTH_URL=http://controller:5000/v3

    export OS_IDENTITY_API_VERSION=3

    export OS_IMAGE_API_VERSION=2

    }

    . demo-openrc

    openstack token issue

c. Cài đặt Glance
- Tạo database và user "glance". Gán quyền cho glance.

  CREATE DATABASE glance;

  GRANT ALL PRIVILEGES ON glance. TO 'glance'@'localhost' IDENTIFIED BY 'GLANCE_DBPASS';*

  GRANT ALL PRIVILEGES ON glance. TO 'glance'@'%' IDENTIFIED BY 'GLANCE_DBPASS';*
- Tạo user

openstack user create --domain default --password-prompt glance

- Thêm role “admin” cho user “glance” và project “service”

  openstack role add --project service --user glance admin

- Tạo service

  openstack service create --name glance --description "OpenStack Image" image

- Tạo API endpoint

  openstack endpoint create --region RegionOne image public http://controller:9292

  openstack endpoint create --region RegionOne image internal http://controller:9292

  openstack endpoint create --region RegionOne image admin http://controller:9292

- Cài đặt và cấu hình các thành phần

  yum install openstack-glance

- Sửa file glance-api.conf

  vi /etc/glance/glance-api.conf

  {

  [database]

  ...

  connection=mysql+pymysql://glance:GLANCE_DBPASS@controller/glance

  [keystone_authtoken]

  ...

  www_authenticate_uri =http://controller:5000

  auth_url=http://controller:5000

  memcached_servers=controller:11211

  auth_type=password

  project_domain_name=Default

  user_domain_name=Default

  project_name=service

  username=glance

  password=GLANCE_PASS

[paste_deploy]

...

flavor=keystone

[glance_store]

...

stores=file,http

default_store=file

filesystem_store_datadir=/var/lib/glance/images/

}

- Sửa file glance-registry.conf

vi /etc/glance/glance-registry.conf

{

[database]

...

connection=mysql+pymysql://glance:GLANCE_DBPASS@controller/glance

[keystone_authtoken]

...

www_authenticate_uri=http://controller:5000

auth_url=http://controller:5000

memcached_servers=controller:11211

auth_type=password

project_domain_name=Default

user_domain_name=Default

project_name=service

username=glance

password=GLANCE_PASS

[paste_deploy]

...

flavor=keystone

}

- Cài đặt glance-manage

  su -s /bin/sh -c "glance-manage db_sync" glance

- Khởi động và cấu hình Image service

  systemctl enable openstack-glance-api.service openstack-glance-registry.service

  systemctl start openstack-glance-api.service openstack-glance-registry.service

- Verify operation

  . admin-openrc

  wget http://cloud-images.ubuntu.com/xenial/current/xenial-server-cloudimg-amd64-disk1.img

  openstack image create "ubuntu" --file xenial-server-cloudimg-amd64-disk1.img --disk-format qcow2 --container-format bare --public

  openstack image list

d. Cài đặt Nova

- Tạo database và user "nova". Gán quyền cho nova.

  CREATE DATABASE nova_api;

  CREATE DATABASE nova;

  CREATE DATABASE nova_cell0;

  CREATE DATABASE placement;

  GRANT ALL PRIVILEGES ON nova_api. TO 'nova'@'localhost' IDENTIFIED BY 'NOVA_DBPASS';*

  GRANT ALL PRIVILEGES ON nova_api. TO 'nova'@'%' IDENTIFIED BY 'NOVA_DBPASS';*

  GRANT ALL PRIVILEGES ON nova. TO 'nova'@'localhost' IDENTIFIED BY 'NOVA_DBPASS';*

  GRANT ALL PRIVILEGES ON nova. TO 'nova'@'%' IDENTIFIED BY 'NOVA_DBPASS';*

  GRANT ALL PRIVILEGES ON nova_cell0. TO 'nova'@'localhost' IDENTIFIED BY 'NOVA_DBPASS';*

  GRANT ALL PRIVILEGES ON nova_cell0. TO 'nova'@'%' IDENTIFIED BY 'NOVA_DBPASS';*

  GRANT ALL PRIVILEGES ON placement. TO 'placement'@'localhost' IDENTIFIED BY 'PLACEMENT_DBPASS';*

GRANT ALL PRIVILEGES ON placement. TO 'placement'@'%' IDENTIFIED BY 'PLACEMENT_DBPASS';*

- Config tài khoản ADMIN

  . admin-openrc

- Tạo user

  openstack user create --domain default --password-prompt nova

- Thêm role “admin” cho user “nova”

  openstack role add --project service --user nova admin

- Tạo service

  openstack service create --name nova --description "OpenStack Compute" compute

- Tạo API endpoint

  openstack endpoint create --region RegionOne compute public http://controller:8774/v2.1

  openstack endpoint create --region RegionOne compute internal http://controller:8774/v2.1

  openstack endpoint create --region RegionOne compute admin http://controller:8774/v2.1

- Thêm role “admin” vào project “service”

  openstack user create --domain default --password-prompt placement

- Tạo service

  openstack service create --name placement --description "Placement API" placement

- Tạo API endpoint

  openstack endpoint create --region RegionOne placement public http://controller:8778

  openstack endpoint create --region RegionOne placement internal http://controller:8778

  openstack endpoint create --region RegionOne placement admin http://controller:8778

- Cài đặt và cấu hình các thành phần

  yum install openstack-nova-api openstack-nova-conductor openstack-nova-console openstack-nova-novncproxy openstack-nova-scheduler openstack-nova-placement-api

- Sửa file nova.conf

vi /etc/nova/nova.conf

{

[DEFAULT]

...

enabled_apis=osapi_compute,metadata

[api_database]

...

connection=mysql+pymysql://nova:NOVA_DBPASS@controller/nova_api

[database]

...

connection=mysql+pymysql://nova:NOVA_DBPASS@controller/nova

[placement_database]

...

connection=mysql+pymysql://placement:PLACEMENT_DBPASS@controller/placement

[DEFAULT]

...

transport_url=rabbit://openstack:RABBIT_PASS@controller

[api]

...

auth_strategy=keystone

[keystone_authtoken]

...

auth_url=http://controller:5000/v3

memcached_servers=controller:11211

auth_type=password

project_domain_name=default

user_domain_name=default

project_name=service

username=nova

password=NOVA_PASS

[DEFAULT]

...

my_ip=10.10.0.33

[DEFAULT]

...

use_neutron=true

firewall_driver=nova.virt.firewall.NoopFirewallDriver

[vnc]

enabled=true

...

server_listen=$my_ip

server_proxyclient_address=$my_ip

[glance]

...

api_servers=http://controller:9292

[oslo_concurrency]

...

lock_path=/var/lib/nova/tmp

[placement]

...

region_name=RegionOne

project_domain_name=Default

project_name=service

auth_type=password

user_domain_name=Default

auth_url=http://controller:5000/v3

username=placement

password=PLACEMENT_PASS

}

- Sửa file 00-nova-placement-api.conf

vi /etc/httpd/conf.d/00-nova-placement-api.conf

{

<Directory /usr/bin>

<IfVersion >= 2.4>

Require all granted

</IfVersion>

<IfVersion < 2.4>

Order allow,deny

Allow from all

</IfVersion>

</Directory>

}

systemctl restart httpd

su -s /bin/sh -c "nova-manage api_db sync" nova

su -s /bin/sh -c "nova-manage cell_v2 map_cell0" nova

su -s /bin/sh -c "nova-manage cell_v2 create_cell --name=cell1 --verbose"

nova 109e1d4b-536a-40d0-83c6-5f121b82b650

su -s /bin/sh -c "nova-manage db sync" nova

su -s /bin/sh -c "nova-manage cell_v2 list_cells" nova

systemctl enable openstack-nova-api.service openstack-nova-consoleauth

openstack-nova-scheduler.service openstack-nova-conductor.service

openstack-nova-novncproxy.service

systemctl start openstack-nova-api.service openstack-nova-consoleauth

openstack-nova-scheduler.service openstack-nova-conductor.service

openstack-nova-novncproxy.service

- Thêm compute node vào cell database

. admin-openrc

openstack compute service list --service nova-compute

- Tìm kiếm compute hosts

su -s /bin/sh -c "nova-manage cell_v2 discover_hosts --verbose" nova

vi /etc/nova/nova.conf

{

[scheduler]

discover_hosts_in_cells_interval=300

}

e. Cài đặt Neutron

- Tạo database và user "neutron". Gán quyền cho neutron.

  CREATE DATABASE neutron;

  GRANT ALL PRIVILEGES ON neutron. TO 'neutron'@'localhost' IDENTIFIED BY 'NEUTRON_DBPASS';*

  GRANT ALL PRIVILEGES ON neutron. TO 'neutron'@'%' IDENTIFIED BY 'NEUTRON_DBPASS';*

- Config tài khoản ADMIN

  . admin-openrc

- Tạo user

  openstack user create --domain default --password-prompt neutron

- Thêm role “admin” cho user “nova”

  openstack role add --project service --user neutron admin

- Tạo service

  openstack service create --name neutron --description "OpenStack Networking" network

- Tạo API endpoint

  openstack endpoint create --region RegionOne network public http://controller:9696

  openstack endpoint create --region RegionOne network internal http://controller:9696

  openstack endpoint create --region RegionOne network admin http://controller:9696

➢ PROVIDER NETWORK

- Cài đặt và cấu hình các thành phần

  yum install openstack-neutron openstack-neutron-ml2 openstack-neutron-linuxbridge ebtables

- Sửa file neutron.conf

  vi /etc/neutron/neutron.conf

  {

  [database]

...

connection=mysql+pymysql://neutron:NEUTRON_DBPASS@controller/neutron

[DEFAULT]

...

core_plugin=ml2

service_plugins =

[DEFAULT]

...

transport_url=rabbit://openstack:RABBIT_PASS@controller

[DEFAULT]

...

auth_strategy=keystone

[keystone_authtoken]

...

www_authenticate_uri=http://controller:5000

auth_url=http://controller:5000

memcached_servers=controller:11211

auth_type=password

project_domain_name=default

user_domain_name=default

project_name=service

username=neutron

password=NEUTRON_PASS

}

- Sửa file ml2_conf.ini

vi /etc/neutron/plugins/ml2/ml2_conf.ini

{

ln -s /etc/neutron/plugins/ml2/ml2_conf.ini /etc/neutron/plugin.ini

}

su -s /bin/sh -c "neutron-db-manage --config-file /etc/neutron/neutron.conf --config-file /etc/neutron/plugins/ml2/ml2_conf.ini upgrade head" neutron

- Khởi động Compute API service

  systemctl restart openstack-nova-api.service

- Chạy Networking service

  systemctl enable neutron-server.service neutron-linuxbridge-agent.service neutron-dhcp-agent.service neutron-metadata-agent.service

  systemctl start neutron-server.service neutron-linuxbridge-agent.service neutron-dhcp-agent.service neutron-metadata-agent.service

  systemctl enable neutron-l3-agent.service

  systemctl start neutron-l3-agent.service

f. Cài đặt Dashboard

- Cài đặt và cấu hình các thành phần

  yum install openstack-dashboard

- Sửa file local_settings.py

  vi /etc/openstack-dashboard/local_settings.py

```
{
        OPENSTACK_HOST="controller"
        ALLOWED_HOSTS=['one.example.com', 'two.example.com']
        SESSION_ENGINE='django.contrib.sessions.backends.cache'
        CACHES={
            'default': {
                    'BACKEND':
                    'django.core.cache.backends.memcached.MemcachedCache',
                    'LOCATION': 'controller:11211',
             }
        }
        OPENSTACK_KEYSTONE_URL="http://%s:5000/v3" %
        OPENSTACK_HOST
        OPENSTACK_KEYSTONE_MULTIDOMAIN_SUPPORT=True
        OPENSTACK_API_VERSIONS={
                "identity": 3,
                "image": 2,
                "volume": 2,
        }
```

OPENSTACK_KEYSTONE_DEFAULT_DOMAIN="Default"

OPENSTACK_KEYSTONE_DEFAULT_ROLE="user"

OPENSTACK_NEUTRON_NETWORK={

...

'enable_router': False,

'enable_quotas': False,

'enable_distributed_router': False,

'enable_ha_router': False,

'enable_lb': False,

'enable_firewall': False,

'enable_vpn': False,

'enable_fip_topology_check': False,

}

TIME_ZONE="TIME_ZONE"

}

- Sửa file openstack-dashboard.conf

  vi /etc/httpd/conf.d/openstack-dashboard.conf

  {

  WSGIApplicationGroup %{GLOBAL}

  }

- Khởi động lại web server và session storage service

  systemctl restart httpd.service memcached.service

- Verify operation

  http://controller/dashboard.

  Xác thực user admin hay demo và mặc định tên miền

g. Cài đặt Cinder

- Tạo database và user " cinder". Gán quyền cho cinder.

  CREATE DATABASE cinder;

  GRANT ALL PRIVILEGES ON cinder. TO 'cinder'@'localhost' IDENTIFIED BY 'CINDER_DBPASS';*

  GRANT ALL PRIVILEGES ON cinder. TO 'cinder'@'%' IDENTIFIED BY 'CINDER_DBPASS';*

- Config tài khoản ADMIN

  . admin-openrc

- Tạo user

  openstack user create --domain default --password-prompt cinder

- Thêm role “admin” cho user “cinder”

  openstack role add --project service --user cinder admin

- Tạo service

  openstack service create --name cinderv2 --description "OpenStack Block Storage" volumev2

  openstack service create --name cinderv3 --description "OpenStack Block Storage" volumev3

- Tạo API endpoint

  openstack endpoint create --region RegionOne volumev2 public http://controller:8776/v2/%\(project_id\)s

  openstack endpoint create --region RegionOne volumev2 internal http://controller:8776/v2/%\(project_id\)s

  openstack endpoint create --region RegionOne volumev2 admin http://controller:8776/v2/%\(project_id\)s

  openstack endpoint create --region RegionOne volumev3 public http://controller:8776/v3/%\(project_id\)s

  openstack endpoint create --region RegionOne volumev3 internal http://controller:8776/v3/%\(project_id\)s

  openstack endpoint create --region RegionOne volumev3 admin http://controller:8776/v3/%\(project_id\)s

- Cài đặt và cấu hình các thành phần

  yum install openstack-cinder

- Sửa file cinder.conf

  vi /etc/cinder/cinder.conf

  {

  [database]

  ...

  connection = mysql+pymysql://cinder:CINDER_DBPASS@controller/cinder

  [DEFAULT]

  ...

transport_url = rabbit://openstack:RABBIT_PASS@controller

[DEFAULT]

...

auth_strategy = keystone

[keystone_authtoken]

...

www_authenticate_uri = http://controller:5000

auth_url = http://controller:5000

memcached_servers = controller:11211

auth_type = password

project_domain_id = default

user_domain_id = default

project_name = service

username = cinder

password = CINDER_PASS

[DEFAULT]

...

my_ip = 10.0.0.11

[oslo_concurrency]

...

lock_path = /var/lib/cinder/tmp

}

- Sửa file cinder.conf

vi /etc/nova/nova.conf

{

[cinder]

os_region_name = RegionOne

}

- Khởi động lại các service

systemctl restart openstack-nova-api.service

systemctl enable openstack-cinder-api.service openstack-cinder-scheduler.service

systemctl start openstack-cinder-api.service openstack-cinder-scheduler.service

2. Triển khai Compute node

a. Chuẩn bị

- Cài đặt Openstack Repository (Rocky)

  yum install centos-release-openstack-rocky

- Cài đặt Openstack client

  yum install python-openstackclient

- CentOS enable SELinux by default. Install the openstack-selinux package to automatically manage security policies for OpenStack services

  yum install openstack-selinux

b. Cài đặt Nova

- Cài đặt và cấu hình các thành phần

  yum install openstack-nova-compute

- Sửa file nova.conf

  vi /etc/nova/nova.conf

  {

  [DEFAULT]

  ...

  enabled_apis=osapi_compute,metadata

  [DEFAULT]

  ...

  transport_url=rabbit://openstack:RABBIT_PASS@controller

  [api]

  ...

  auth_strategy=keystone

  [keystone_authtoken]

  ...

  auth_url=http://controller:5000/v3

  memcached_servers=controller:11211

  auth_type=password

project_domain_name=default

user_domain_name=default

project_name=service

username=nova

password=NOVA_PASS

[DEFAULT]

...

my_ip=MANAGEMENT_INTERFACE_IP_ADDRESS

[DEFAULT]

...

use_neutron=true

firewall_driver=nova.virt.firewall.NoopFirewallDriver

[vnc]

...

enabled=true

server_listen=0.0.0.0

server_proxyclient_address=$my_ip

novncproxy_base_url=http://controller:6080/vnc_auto.html

[glance]

...

api_servers=http://controller:9292

[oslo_concurrency]

...

lock_path=/var/lib/nova/tmp

[placement]

...

region_name=RegionOne

project_domain_name=Default

project_name=service

auth_type=password

user_domain_name=Default

auth_url=http://controller:5000/v3

username=placement

password=PLACEMENT_PASS

[libvirt]

...

virt_type=qemu

}

egrep -c '(vmx|svm)' /proc/cpuinfo

systemctl enable libvirtd.service openstack-nova-compute.service

systemctl start libvirtd.service openstack-nova-compute.service

- Thêm compute node vào cell database (nêu Cài đặt Controller node)

c. Cài đặt Neutron

- Cài đặt và cấu hình các thành phần

  yum install openstack-neutron-linuxbridge ebtables ipset

- Sửa file neutron.conf

  vi /etc/neutron/neutron.conf

  {

  [DEFAULT]

  ...

  transport_url=rabbit://openstack:RABBIT_PASS@controller

  [DEFAULT]

  ...

  auth_strategy=keystone

  [keystone_authtoken]

  ...

  www_authenticate_uri=http://controller:5000

  auth_url=http://controller:5000

  memcached_servers=controller:11211

  auth_type=password

  project_domain_name=default

  user_domain_name=default

  project_name=service

username=neutron

password=NEUTRON_PASS

[oslo_concurrency]

...

lock_path=/var/lib/neutron/tmp

}

- Sửa file linuxbridge_agent.ini

vi /etc/neutron/plugins/ml2/linuxbridge_agent.ini

{

[linux_bridge]

physical_interface_mappings=provider:PROVIDER_INTERFACE_NAME

[vxlan]

enable_vxlan=false

[securitygroup]

...

enable_security_group=true

firewall_driver=neutron.agent.linux.iptables_firewall.IptablesFirewallDriver

}

modprobe br_netfilter

sysctl -p

sysctl net.bridge

- Sửa file nova.conf

vi /etc/nova/nova.conf

{

[neutron]

...

url=http://controller:9696

auth_url=http://controller:5000

auth_type=password

project_domain_name=default

user_domain_name=default

region_name=RegionOne

project_name=service

username=neutron

password=NEUTRON_PASS

}

systemctl enable openstack-nova-compute.service

systemctl restart openstack-nova-compute.service

systemctl enable neutron-linuxbridge-agent.service

systemctl start neutron-linuxbridge-agent.service

- PROVIDER NETWORK
  - Sửa file neutron.conf

vi /etc/neutron/neutron.conf

{

[DEFAULT]

...

notify_nova_on_port_status_changes=true

notify_nova_on_port_data_changes=true

[nova]

...

auth_url=http://controller:5000

auth_type=password

project_domain_name=default

user_domain_name=default

region_name=RegionOne

project_name=service

username=nova

password=NOVA_PASS

[oslo_concurrency]

...

lock_path=/var/lib/neutron/tmp

}

3. Triển khai Storage node

a. Chuẩn bị

- Cài đặt gói phần mềm LVM

  yum install lvm2 device-mapper-persistent-data

- Khởi động dịch vụ LVM

  systemctl enable lvm2-lvmetad.service

  systemctl start lvm2-lvmetad.service

- Tạo volume LVM

  pvcreate /dev/sdb

  vgcreate cinder-volumes /dev/sdb

- Sửa file lvm.conf

  vi /etc/lvm/lvm.conf

  {

  devices {

  ...

  filter = ["a/sdb/", "r/./"]*

  }

  }

b. Cài đặt Storage

- Cài đặt và cấu hình các thành phần

  yum install openstack-cinder targetcli python-keystone

- Sửa file cinder.conf

  vi /etc/cinder/cinder.conf

  {

  [database]

  ...

  connection =

  mysql+pymysql://cinder:CINDER_DBPASS@controller/cinder

  [DEFAULT]

  ...

  transport_url = rabbit://openstack:RABBIT_PASS@controller

  [DEFAULT]

...

auth_strategy = keystone

[keystone_authtoken]

...

www_authenticate_uri = http://controller:5000

auth_url = http://controller:5000

memcached_servers = controller:11211

auth_type = password

project_domain_id = default

user_domain_id = default

project_name = service

username = cinder

password = CINDER_PASS

[DEFAULT]

...

my_ip = MANAGEMENT_INTERFACE_IP_ADDRESS

[lvm]

volume_driver = cinder.volume.drivers.lvm.LVMVolumeDriver

volume_group = cinder-volumes

iscsi_protocol = iscsi

iscsi_helper = lioadm

[DEFAULT]

...

enabled_backends = lvm

[DEFAULT]

...

glance_api_servers = http://controller:9292

[oslo_concurrency]

...

lock_path = /var/lib/cinder/tmp

[DEFAULT]

...

backup_driver = cinder.backup.drivers.swift

backup_swift_url = SWIFT_URL

}

- Để hiển thị SWIFT_URL

  openstack catalog show object-store

- Khởi động lại các service

  systemctl enable openstack-cinder-volume.service target.service

  systemctl start openstack-cinder-volume.service target.service

  systemctl enable openstack-cinder-backup.service

  systemctl start openstack-cinder-backup.service

- Verify operation

  . admin-openrc

  openstack volume service list