TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN ĐIỆN TỬ - VIỄN THÔNG

BÁO CÁO ĐỒ ÁN III

***ĐỀ TÀI**: Triển khai chuỗi SFC trên môi trường cloud*

Giảng viên hướng dẫn: PGS. TS. Nguyễn Hữu Thanh

Sinh viên thực hiện	*MSSV*
Đỗ Xuân Sơn	20133299
Nguyễn Thị Thanh Tâm	20133430

Lớp : KT ĐTTT 06 – K58

Hà Nội, tháng 01 năm 2018

LỜI MỞ ĐẦU

Hiện nay, để cung cấp một dịch vụ đầu cuối hoàn chỉnh cho khách hàng, các nhà cung cấp dịch vụ mạng viễn thông và công nghệ thông tin phải thiết lập và cấu hình hệ thống các dịch vụ mạng, chức năng mạng phù hợp để đảm bảo dịch vụ tới người dùng hoạt động ổn định, tin cậy.

Đa số các thiết bị viễn thông hiện nay chủ yếu dùng các thiết bị chuyên dụng của các nhà cung cấp thiết bị mạng, điều đó dẫn đến tổn hao chi phí lớn mua thiết bị và lệ thuộc vào nhà sản xuất.

Điện toán đám mây ra đời cùng với việc tích hợp với các công nghệ nổi bật như NFV và SFC để giải quyết vấn đề đó.

Nhóm đồ án xin chân thành cảm ơn PGS.TS. Nguyễn Hữu Thanh đã tận tình giúp đỡ, tạo điều kiện để nhóm thực hiện đồ án tốt nghiệp. Ngoài ra nhóm đồ án cũng xin chân thành cảm ơn các thành viên Future Internet Lab đặc biệt là các bạn trong nhóm Network Function Virtualization đã giúp đỡ, chia sẻ trong suốt thời gian qua.

MỤC LỤC

DANH MỤC HÌNH VẼ

1. CHƯƠNG 1: GIỚI THIỆU CHUNG

1.1. Đặt vấn đề

Sự gia tăng yêu cầu về kết nối trong thế giới hiện đại đặt ra những yêu cầu tới trong việc cung cấp dịch vụ, đặc biệt là với sự xuất hiện của khái niệm IoT (Internet of Things – Internet của vạn vật). Các nhà cung cấp dịch vụ, những người chịu trách nhiệm cung cấp các kết nối này đang phải đối mặt với sự gia tăng về lưu lượng mạng và số lượng thuê bao trong hệ thống mạng của họ. Những thách thức mà chọ phải đối mặt có thể kể tới như:

- Sự bùng nổ về số lượng.
- Không đủ khả năng cung cấp dịch vụ mới với sự gia tăng nhanh chóng và thay đổi liên tục về số lượng người dùng.
- Chi phí đầu tư và chi phí vận hành ngày càng tăng làm giảm doanh thu.

Các nhà cung cấp dịch vụ nói chung chưa thể tối ưu hóa tài nguyên sử dụng bởi họ trước hết phải đảm bảo phục vụ được tốc độ lưu lượng mạng ở mức đỉnh. Yêu cầu mới đặt ra là họ phải cải thiện hệ thống để quản lý việc cung cấp mạng lưới gồm rất nhiều ứng dụng đồng thời tối đa hóa được lượng tài nguyên sử dụng tránh lãng phí bằng việc triển khai cơ sở hạ tầng ảo hóa.

Đối với nhà cung cấp dịch vụ, để triển khai và cung cấp hoàn chỉnh một số dịch vụ đầu cuối cho khách hàng thường yêu cầu sử dụng nhiều thiết bị cung cấp các dịch vụ mạng. Các thiết bị này đặt ở nhiều nơi trong hệ thống mạng của nhà cung cấp, trong trung tâm dữ liệu hoặc giữa các trung tâm dữ liệu của nhà cung cấp với nhau.

Chuỗi chức năng mạng hiện nay phần lớn vẫn sử dụng các thiết bị chuyên dụng, dẫn đến tốn kém về chi phí và phụ thuộc vào nhà sản xuất.

Công nghệ "Ảo hóa các chức năng mạng" (NFV) áp dụng thành tựu của ảo hóa cho các thiết bị mạng truyền thống để giảm thiểu chi phí đầu tư và vận hành đồng thời cho phép cải thiện thời gian đưa dịch vụ vào thị trường. NFV đang thay đổi cách mà nhà cung cấp dịch vụ thiết kế và triển khai hệ thống mạng của họ. Kết hợp NFV với xây dựng chuỗi các chức năng mạng SFC sẽ làm tăng khả năng linh hoạt trong thiết lập và quản lý.

Trong đồ án này, nhóm thực hiện tiến hành tìm hiểu các vấn đề liên quan tới NFV và thiết kế một chuỗi SFC cơ bản.

1.2. Hướng giải quyết và phát triển đề tài

NFV ra đời thu hút được sự quan tâm đông đảo từ phía các nhà cung cấp dịch vụ công nghệ thông tin và viễn thông. Cùng với đó là sự ra đời của nhiều dự án công nghệ thông tin mã nguồn mở để hỗ trợ và đáp ứng được từng phần tử trong kiến trúc NFV. Chuỗi chức năng mạng hiện nay phần lớn vẫn sử dụng các thiết bị chuyên dụng, dẫn đến tốn kém về chi phí và phụ thuộc vào nhà sản xuất.

NFV ra đời đã giải quyết được vấn đề này bằng việc thay vì phải sử dụng các thiết bị chuyên dụng để làm các chức năng mạng, chúng ta có thể sử dụng các server vật lý được cài các phần mềm giả lập các chức năng mạng như firewall, loadblance,…

Những công việc chính trong đồ án:

- Tìm hiểu Openstack các khía cạnh liên quan.
- Triển khai Openstack với các project chính.
- Tìm hiểu các công nghệ NFV và một số VNF cơ bản.
- Xây dựng chuỗi SFC cơ bản.

2. CHƯƠNG 2: CƠ SỞ LÝ THUYẾT CHUNG

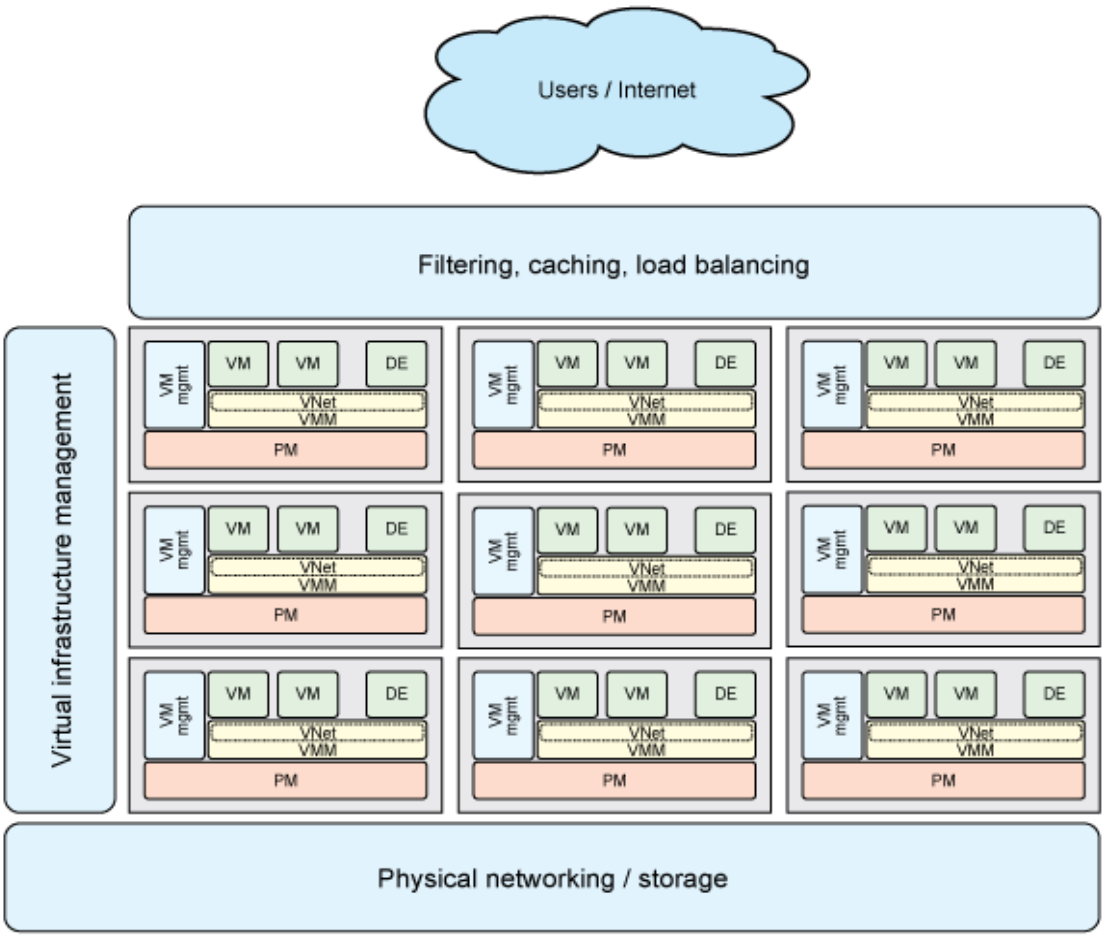
2.1. Tổng quan về Công nghệ điện toán đám mây

2.1.1. Giới thiệu

Điện toán đám mây (Cloud Computing), còn gọi là điện toán máy chủ ảo: là mô hình tính toán sử dụng các công nghệ máy tính và phát triển dựa vào mạng Internet.

Thuật ngữ ***"Cloud Computing"*** ra đời giữa năm 2007, dùng để khái quát lại các hướng phát triển của cơ sở hạ tầng CNTT vốn đã và đang diễn ra từ nhiều năm qua. Quan niệm này có thể được diễn giải một cách đơn giản như sau: các nguồn tài nguyên tính toán khổng lồ như các phần cứng (máy chủ), phần mềm, và các dịch vụ (chương trình ứng dụng), … sẽ nằm tại các máy chủ ảo (đám mây - cloud) trên Internet thay vì trong máy tính gia đình và văn phòng (trên mặt đất) để mọi người kết nối và sử dụng mỗi khi họ cần dù ở bất cứ đâu miễn có kết nối Internet.

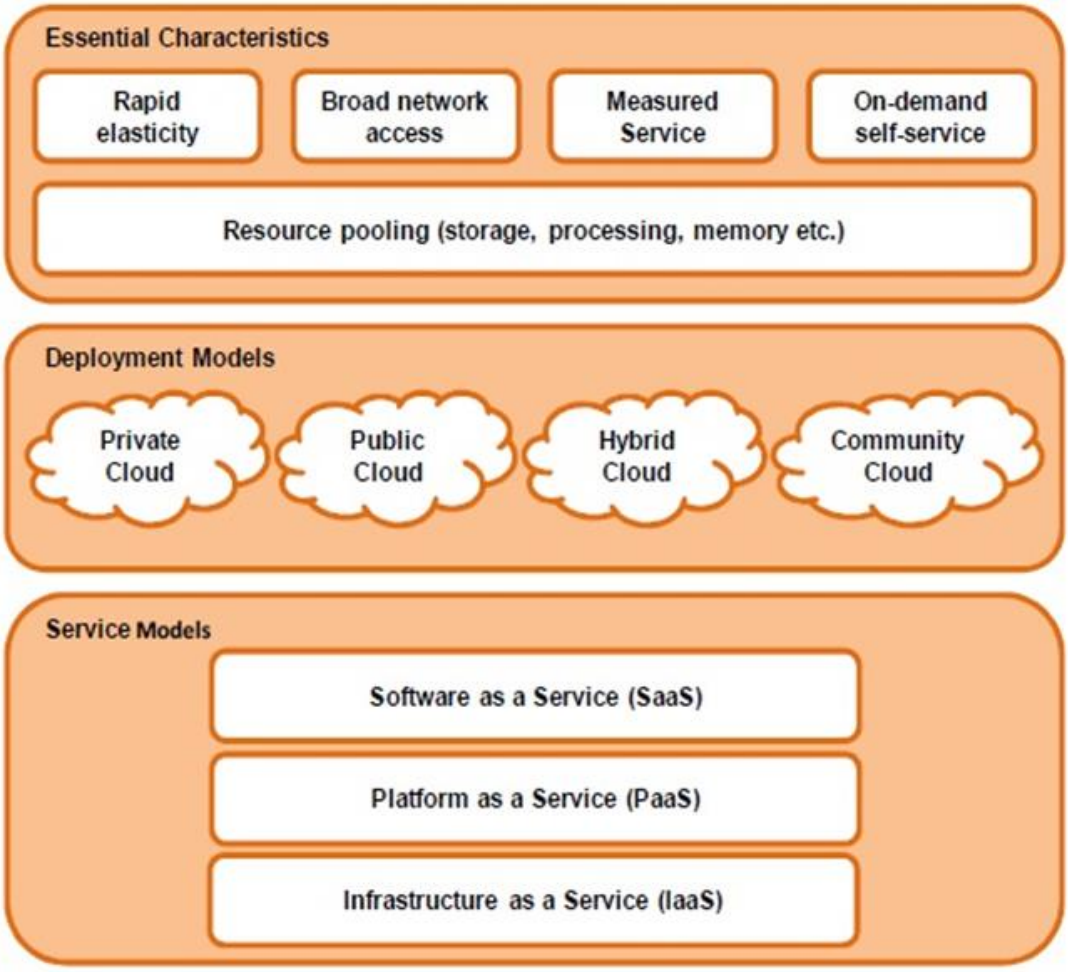
Nói cách khác, đây là một mô hình cho phép truy cập qua mạng để chia sẻ chung nguồn tài nguyên (mạng, dịch vụ, ứng dụng, máy chủ, không gian lưu trữ …) một cách nhanh chóng, thuận tiện và đồng thời cho phép kết thúc sử dụng dịch vụ, giải phóng tài nguyên dễ dàng, giảm thiểu sự ảnh hưởng quản lý và giao tiếp với nhà cung cấp. Mọi khả năng liên quan đến công nghệ thông tin đều được cung cấp dưới dạng các "dịch vụ", cho phép người sử dụng truy cập các dịch vụ công nghệ thông tin từ một nhà cung cấp nào đó "trong đám mây" mà không cần phải biết về công nghệ đó, cũng như không cần quan tâm đến các cơ sở hạ tầng phục vụ công nghệ đó.



Hình 2.1: Mô hình cơ sở hạ tầng của Cloud

2.1.2. Các đặc trưng của điện toán đám mây

Mô hình điện toán đám mây – Cloud Computing đặc trưng bởi 5 đặc tính, 3 mô hình dịch vụ và 4 mô hình triển khai.



Hình 2.2: Các đặc trưng của điện toán đám mây

5 đặc tính:

- ***On-demand self service :*** Khả năng tự phục vụ : người dùng có khả năng tự tính toán được nhu cầu sử dụng máy chủ lưu trữ, khi cần thiết có thể thực hiện tự động mà không cần phải có sự giao tiếp với nhà cung cấp dịch vụ.

- ***Broad network access:*** khả năng truy cập nhiều loại mạng, hỗ trợ nhiều nền tảng thiết bị , nhiều hạ tầng vật lý (như điện thoại di động, laptop, tablet, máy trạm..)
- ***Resource pooling:*** Dùng chung tài nguyền mạng. Tài nguyên tính toán được của nhà cung cấp được dùng chung cho nhiều khách hang sử dụng dịch vụ đa thuê bao, với những nguồn tài nguyên vật lý và ảo khác nhau được phân công và bố trí theo nhu cầu của người sử dụng. Khách hàng không thể xác định và biết được vị trí chính xác của nguồn tài nguyên nhưng mà có thể dễ dàng xác dịnhđược vị trí cung cấp nguồn tài nguyên một cách tương đối (quốc gia, liên bang, trung tâm dữ liệu).

  Gộp chung nguồn tài nguyên của nhiều máy chủ với nhau thành một khối thống nhất, rồi sau đó sẽ san sẻ tài nguyên cho người dùng. (mục đích dễ quản lý, cấp phát tài nguyên dễ dàng, nhanh chóng).

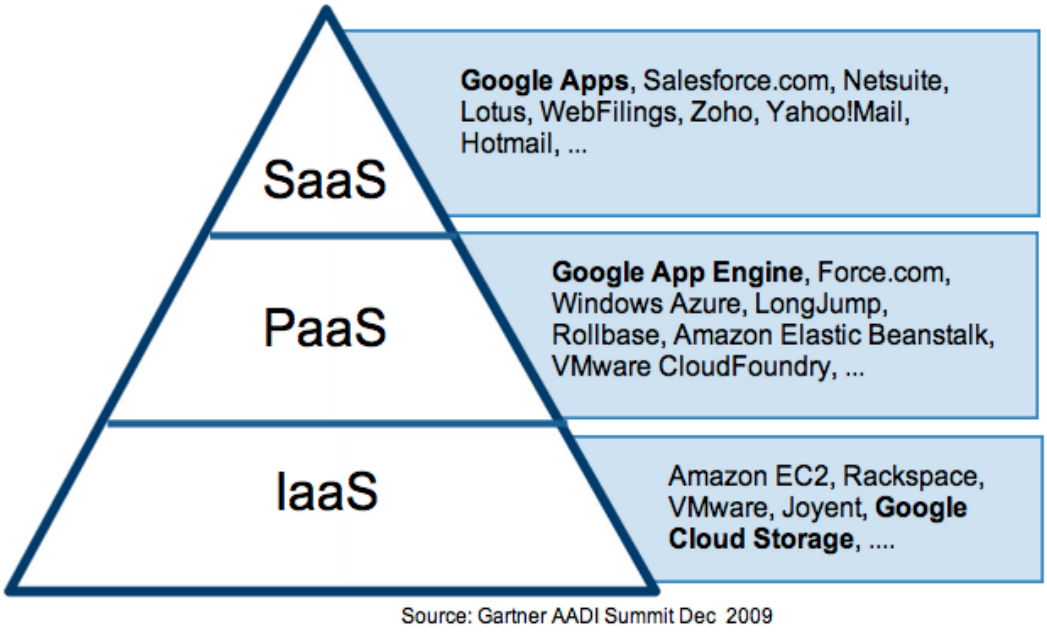
- ***Rapid Elasticity*** (có tính đàn hồi cao) : Có khả năng cung cấp và giải phóng tài nguyên một cách nhanh chóng, trong một số trường hợp có thể tự động thu hẹp hoặc mở rộng phạm vi tùy nhu cầu người dùng. Khách hàng có khả năng có thể được cung cấp không giới hạn và chiếm tài nguyên bất kì lúc nào .
- ***Measured service*** (Tính toán dịch vụ) : Hệ thống cloud tự động điều khiển và tối ưu hóa nguồn tài nguyên sử dụng bằng tận dụng khả năng đo lường ở một vài cấp độ dịch vụ. tính toán mức độ sử dụng dịch vụ, kiểm soát thời gian phục vụ, giám sát, điều khiển, báo cáo, etc. Từ đó có thể tính toán được chi phí của người sử dụng.

4 mô hình triển khai:

- ***Private cloud :*** Nền tảng cloud được cung cấp độc quyền cho 1 tổ chức bao gồm nhiều đối tượng (như công ty kinh doanh..). Nó có thể được làm chủ, điều khiển và vận hành bởi một tổ chức - môt thành viên thứ 3, hoặc là có sự kết hợp giữa 2 bên, có thể tồn tại hoặc không …

- ***Community cloud:*** Nền tảng cloud được cung cấp độc quyền cho một tổ chức người sử dùng đến từ các tổ chức có chung một mối quan tâm về mục đích, nhiệm vụ, chính sách. Nó có thể được quản lý, vận hành bởi một hoặc nhiều tổ chức trong cộng đồng kết hợp quản lý với nhau.
- ***Public Cloud:*** Hạ tầng Cloud đc cung cấp cho tất cả mọi người dùng thông thường. Được làm quản lý và vận hành bởi chính người dùng, có tính chất thương mại.. Có sự ràng buộc giữa người dùng và nhà cung cấp.
- ***Hybrid Cloud:*** Đây là hạ tầng cloud được phối hợp giữa 2 hoặc nhiều hạ tầng cloud riêng biêt. Những hạ tầng này vẫn giữ đặc điểm riêng biệt nhưng có thể phối hợp cùng nhau bởi việc được chuẩn hóa hoặc công nghệ phù hợp mà dữ liệu và ứng dụng có thể lưu động được.

3 mô hình dịch vụ:

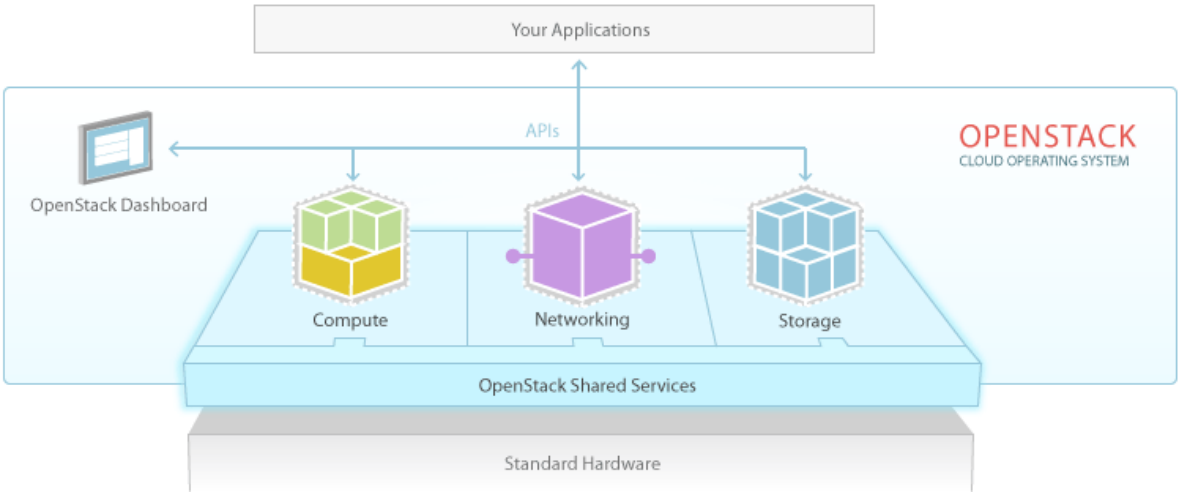


Hình 2.3: 3 mô hình dịch vụ của điện toán đám mây.

- ***SaaS (Software as a Service) :*** Khách hàng sử dụng những ứng dụng chạy trên nền tảng cloud của nhà cung cấp**.** Các ứng dụng có thể truy cập được từ các thiết bị thông qua giao diện với người dùng, Khách hang không có quyền quản lý và điều khiển kiến trúc hạ tầng của cloud như mạng, server, lưu trữ, hệ thống vẫn hành và thậm chí cả một vài ứng dụng cá nhân … do đó hạn chế người dùng về việc cáu hình đặc biệt cho các ứng dụng. (người dùng không cần quan tâm xem cloud triển khai như thé nào, chỉ cần thuê và sử dụng dịch vụ thông qua các phần mềm client (web browser..)
- ***PaaS (Platform as a Service) :*** Nền tảng như một dịch vụ. cung cấp các dịch vụ về nền tảng, môi trường lập trình, database, etc. để khách hàng phát triển các ứng dụng của mình (thông thường nền tảng này cung cấp cho các developer). Ví dụ: AWS, GAE, Azue, Bluemix, OpenShift, etc.. Nhưng khách hàng không được quản lý các kiến trúc hạ tầng của cloud, được phép điều khiển các ứng dụng triển khai và có thể đồng bộ cài đặt cho môi trường quản lý ứng dụng.
- ***IaaS (Infrastructure as a Service):*** Cung cấp các dịch vụ về hạ tầng, các máy chủ, tài nguyên tính toán (RAM, CPU), lưu trữ. Trên đó người dùng sẽ tạo các máy ảo với hệ điều hành, triển khai ứng dụng theo nhu cầu của mình. (Người dùng không được điều khiển hạ tầng cloud nhưng có thể điều khiển được hệ thống OS, khả năng lưu trữ, và các ứng dụng triển khai, và có thể có một số quyền kiểm soát chọn mạng.

2.2. Tổng quan về Openstack

2.2.1. Giới thiệu Openstack



Hình 2.4: Openstack

OpenStack là một dự án phần mềm mã nguồn mở dùng để triển khai private và public cloud. Nó bao gồm nhiều thành phần (project) do các công ty, tổ chức và các lập trình viên tự nguyện xây dựng và phát triển.

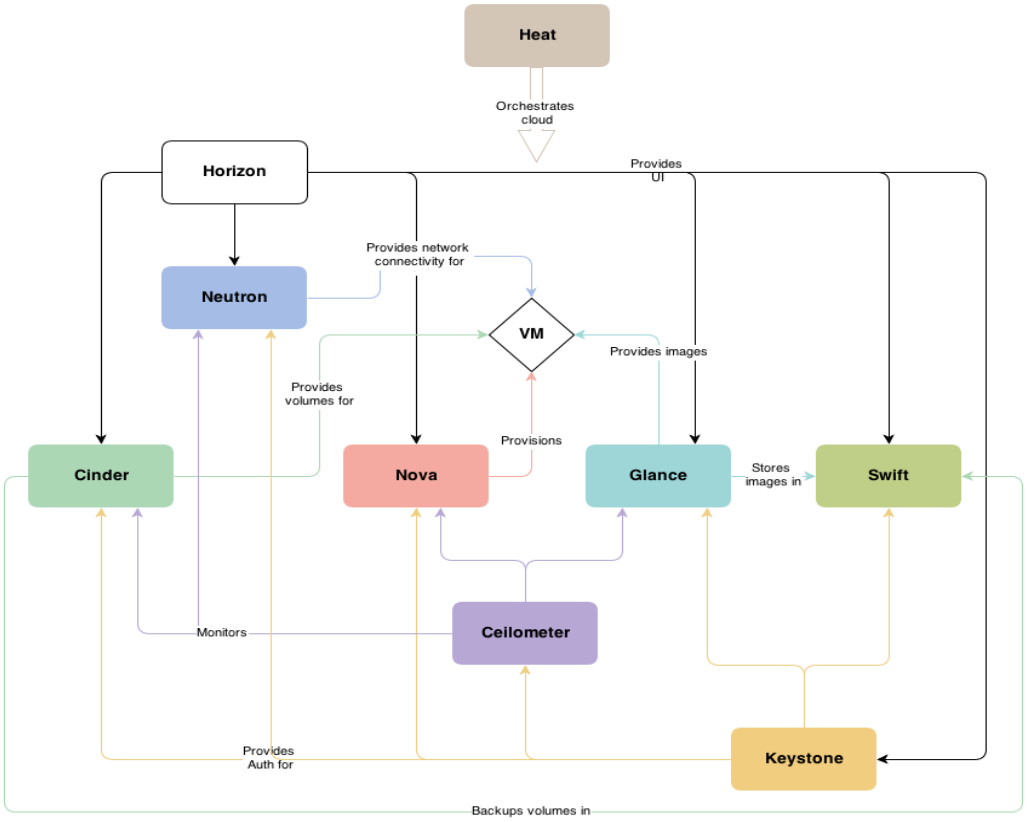
OpenStack hoạt động theo hướng mở: công khai lộ trình phát triển, công khai mã nguồn mở…

OpenStack được phát triển và phát hành phiên bản mới trong vòng 6 tháng, hiện tại đã có 13 phiên bản OpenStack. Tên các phiên bản được đặt theo thứ tự chữ cái A, B, C, … phiên bản hiện tại là Pike. Tới tháng 4/2018 sẽ có phiên bản mới là Queen.

Phần lớn mã nguồn của OpenStack là Python.

Có thể coi OpenStack như một hệ điều hành cloud có nhiệm vụ kiểm soát các tài nguyên tính toán (compute), lưu trữ (storage) và networking trong hệ thống lớn Datacenter, tất cả đều có thể được kiểm soát qua giao diện dòng lệnh hoặc một dashboard (do project

horizon cung cấp). Ở thời điểm hiện tại, OpenStack có 6 core project và 35 project tùy chọn cài đặt theo nhu cầu. 6 core project của OpenStack bao gồm: KEYSTONE, GLANCE, NOVA, NEUTRON, CINDER, HORIZON.



Hình 2.5: Mô hình kiến trúc mức khái niệm các Project chính trong Openstack.

Ngoài ra còn một số Project tùy chọn để triển khai NFV như Tacker, Heat.

2.2.2. Một số project chính trong Openstack

KEYSTONE - Identity Service

Một trong những tính năng quan trọng của môi trường cloud là cung cấp tài nguyên một cách bảo mật, truy cập điều khiển an toàn vào các tài nguyên có giá trị. Trong Openstack (OPS) Keystone là dịch vụ có khả năng cung cấp sự truy cập an toàn tới tất cả các tài nguyên cloud. Keystone đã chứng minh được sự cần thiết của nó trong môi trường cloud bảo mật.

Keystone đảm nhận các chức năng:

- Cung cấp dịch vụ xác thực và ủy quyền cho các dịch vụ khác của OpenStack, cung cấp danh mục của các endpoints cho tất các dịch vụ trong OpenStack. Cụ thể hơn:
- Xác thực user và vấn đề token để truy cập vào các dịch vụ
- Lưu trữ user và các tenant cho vai trò kiểm soát truy cập(cơ chế role-based access control - RBAC)
- Cung cấp catalog của các dịch vụ (và các API enpoints của chúng) trên cloud
- Tạo các policy giữa user và dịch vụ
- Mỗi chức năng của Keystone có kiến trúc pluggable backend cho phép hỗ trợ kết hợp với LDAP, PAM, SQL.

GLANCE - Image Service

Glance là Image services của Openstack bao gồm việc tìm kiếm, đăng ký, thu thập các images của các máy ảo. Glance là một reposity tập trung các image ảo. Dịch vụ OpenStack Compute sẽ sử dụng chúng trong suốt quá trình dự phòng instances.

Các chức năng chính của GLANCE:

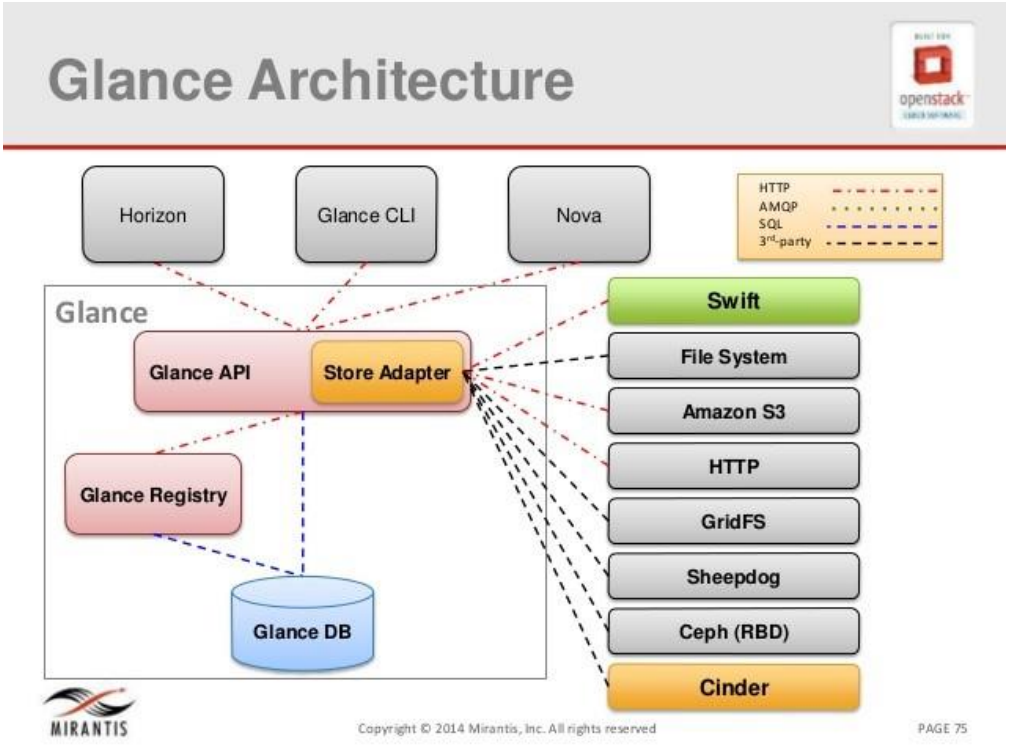
Glance cung cấp RESTful API cho phép truy vấn metadata của image máy ảo cũng như thu thập các image thực sự.

Images của máy ảo thông qua Glance có thể lưu trữ ở nhiều vị trí khác nhau từ hệ thống file thông thường cho tới hệ thống object-storage như project Swift trong OpenStack.

Trong Glance, các images được lưu trữ giống như các template. Các Template này sử dụng để vận hành máy ảo mới. Glance là giải pháp để quản lý các disk image trên cloud.

Glance được thiết kế để có thể là dịch vụ hoạt động độc lập cho sự cần thiết các tổ chức lớn lưu trữ và quản lý các dish image ảo.

Nó cũng có thể tạo các bản snapshots từ các máy ảo đang chạy để thực hiện dự phòng cho các VM và trạng thái các máy ảo đó.



Hình 2.6: Kiến trúc Glance trong Openstack

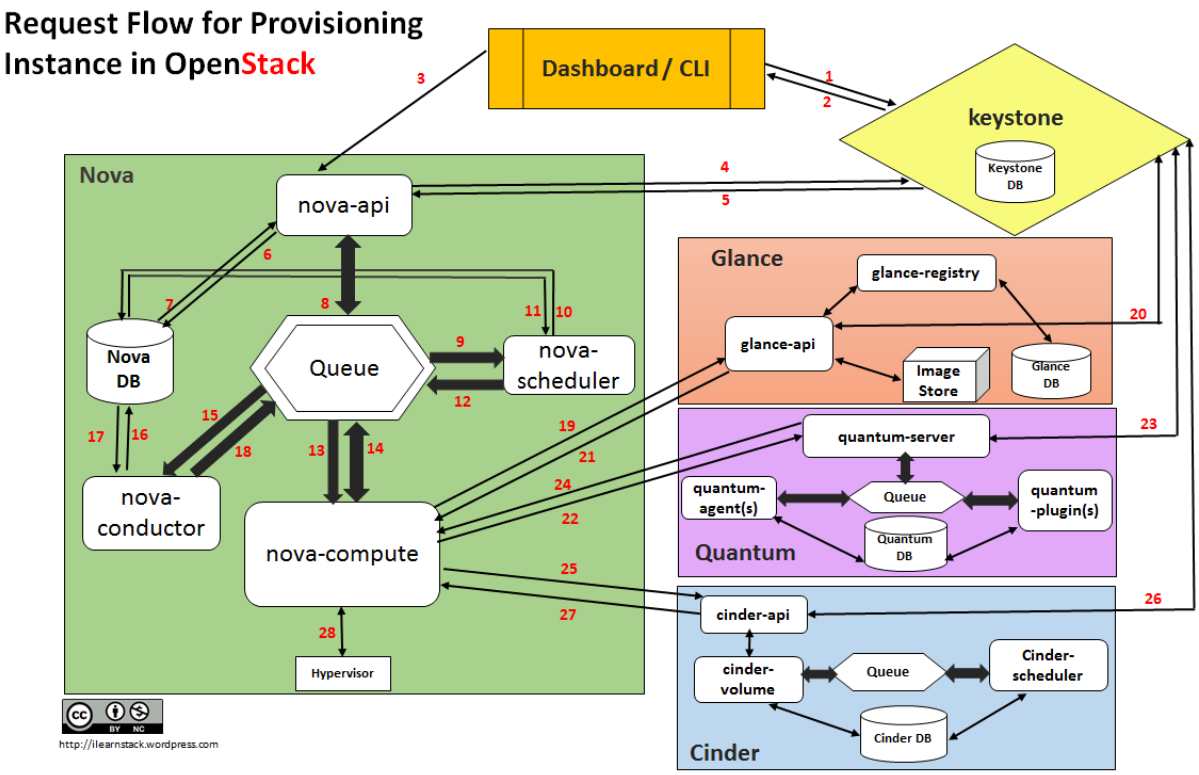
Glance bao gồm các thành phần sau:

- **`glance-api`**: tiếp nhận lời gọi API để tìm kiếm, thu thập và lưu trữ image
- **`glance-registry`**: thực hiện tác vụ lưu trữ, xử lý và thu thập metadata của images
- **`database`**: cơ sở dữ liệu lưu trữ metadata của image
- **`storage repository`**: được tích hợp với nhiều thành phần khác trong OpenStack như hệ thống file thông thường, Amazon và HTTP phục vụ cho chức năng lưu trữ images

NOVA - Compute service

Là service chịu trách nhiệm chứa và quản lý các máy ảo trong hệ thống cloud computing. OpenStack Compute chính là phần chính quan trọng nhất trong kiến trúc hệ thống Infrastructure-as-a-Service (IaaS). Phần lớn các modules của Nova được viết bằng Python.

OpenStack Compute giao tiếp với OpenStack Identity để xác thực, OpenStack Image để lấy images và OpenStack Dashboard để lấy giao diện cho người dùng và người quản trị.



Hình 2.7: Luồng làm việc cảu Nova trong Openstack

Nova cho phép bạn điều khiển các máy ảo và networks, bạn cũng có thể quản lí các truy cập tới cloud từ users và projects. OpenStack Compute không chứa các phần mềm ảo hóa. Thay vào đó, nó sẽ định nghĩa các drivers để tương tác với các kĩ thuật ảo hóa khác chạy trên hệ điều hành của bạn và cung cấp các chức năng thông qua một web-based API.

Chức năng

Quản lí các máy ảo trong môi trường OpenStack, chịu trách nhiệm khởi tạo, lập lịch, bật, tắt hoạt động của các máy ảo theo yêu cầu.

- Nova bao gồm nhiều tiến trình trên server, mỗi tiến trình lại thực hiện một chức năng khác nhau.

- Nova cung cấp REST API để tương tác với user, các thành phần bên trong Nova truyền thông với nhau thông qua cơ chế truyền RPC message.

- API server xử lý các REST request, thường liên quan đến việc đọc/ghi databases, tùy chọn gửi RPC messages đến các Nova services khác, và tạo ra các trả lời cho REST calls. RPC message thực hiện thông qua thư viện oslo.messaging.

- Hầu hết các thành phần chính của Nova có thể chạy trên nhiều server, và có manager lắng nghe RPC messages. Ngoài trừ nova-compute, tiến trình duy nhất chạy trên hypervisor mà nó quản lý(ngoài trừ sử dụng VMware hoặc Ironic drivers).

- Nova có thể sử dụng central database được chia sẻ giữa tất các thành phần. Tuy nhiên, để hỗ trợ upgrade, DB được truy cập thông qua một object layer để đảm bảo các thành phần kiểm soát đã upgrade có thể giao tiếp với nova-compute chạy ở phiên bản trước. Để làm điều này, nova-compute ủy quyền các yêu cầu tới DB thông qua RPC tới một trình quản lý trung tâm, chính là dịch vụ nova-conductor.

NEUTRON – Networking Service

OpenStack Networking cho phép bạn tạo và quản lí các đối tượng trong network ví dụ như các mạng (networks), các lớp mạng con (subnets), và các cổng (ports) cho các services khác của OpenStack sử dụng. Với kiến trúc plugable, các plug-in có thể được sử dụng để triển khai các thiết bị và phần mềm khác nhau, nó khiến OpenStack có tính linh hoạt trong kiến trúc và triển khai.

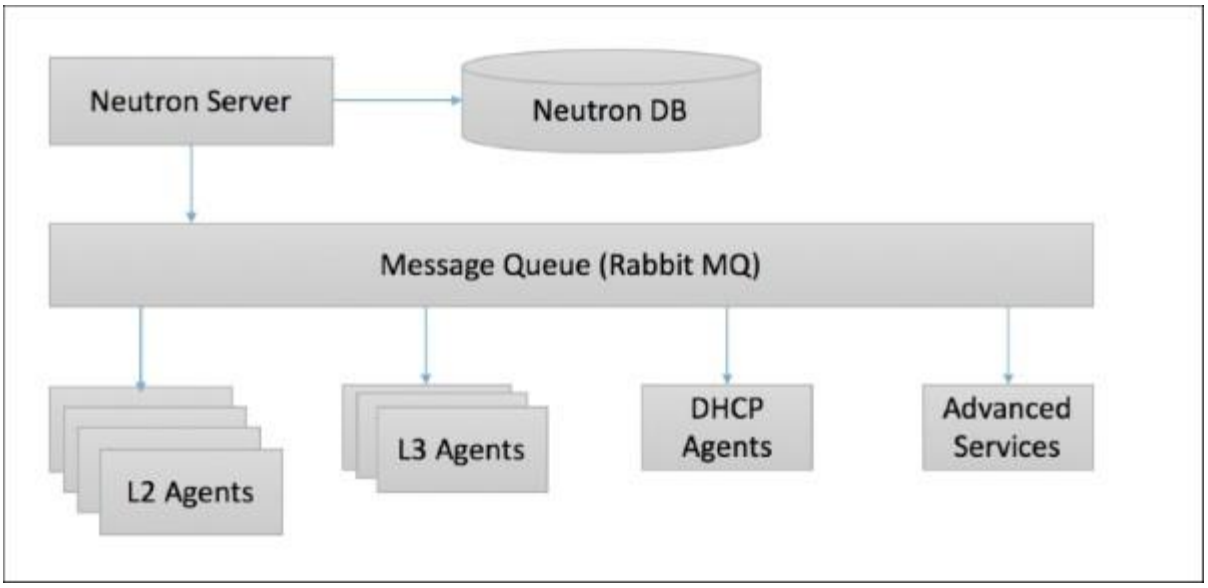
Các chức năng của Neutron:

- Dịch vụ Networking trong OpenStack (neutron) cung cấp API cho phép bạn định nghĩa các kết nối mạng và gán địa chỉ ở trong môi trường cloud.
- Cho phép các nhà khai thác vận hành các công nghệ networking khác nhau cho phù hợp với mô hình điện toán đám mây của riêng họ.

- Neutron cũng cung cấp một API cho việc cấu hình cũng như quản lí các dịch vụ networking khác nhau từ L3 forwarding, NAT cho tới load balancing, perimeter firewalls, và virtual private networks.

Các thành phần:

- API server: OpenStack Networking API bao gồm hỗ trợ Layer 2 networking và quản lý các địa chỉ IP (IP address management - IPAM), cũng như 1 phần mở rộng cho Layer 3 router cho phép định tuyến giữa các giữa các mạng lớp 2. OpenStack Networking bao gồm 1 danh sách các plug-in cho phép tương tác với các công nghệ mạng mã nguồn mở khác nhau, bao gồm routers, virtual switches, và Software-Defined Networking (SDN) controllers.
- Các plug-in và agents: Các plugin và các agent này cho phép gắn và gỡ các ports, tạo ra network hay subnet, và đánh địa chỉ IP. Lựa chọn plugin và agents nào là tùy thuộc vào nhà cung cấp và công nghệ sử dụng trong hệ thống cloud nhất định. Điều quan trọng là tại một thời điểm chỉ sử dụng được một plug-in.
- Messaging queue: Tiếp nhận và định tuyến các yêu cầu giữa các agents để hoàn thành quá trình vận hành API. Các Message queue được sử dụng trong ML2 plugin để thực hiện truyền thông RPC giữa neutron server và các neutron agents chạy trên mỗi hypervisor, cụ thể là các ML2 driver cho Open vSwitch và Linux bridge.



Hình 2.8: Các thành phần của Neutron

Với Neutron, bạn có thể tạo và cấu hình các networks, subnets và thông báo tới Compute để gán các thiết bị ảo vào các ports của mạng vừa tạo. OpenStack Compute chính là "khách hàng" của neutron, chúng liên kết với nhau để cung cấp kết nối mạng cho các máy ảo. Cụ thể hơn, OpenStack Networking hỗ trợ cho phép các projects có nhiều private networks và các projects có thể tự chọn danh sách IP cho riêng mình, kể cả những IP đã được sử dụng bởi một project khác. Có hai loại network đó là project network (selfservice) và provider network.

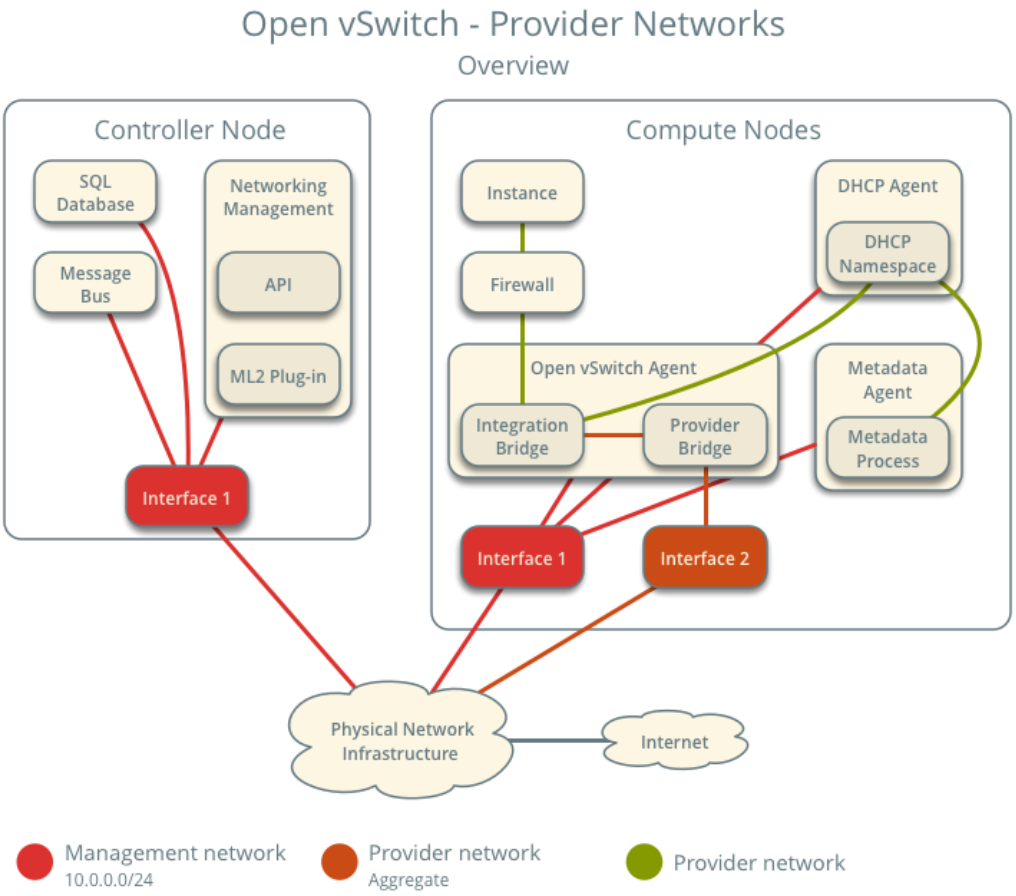
- Project network là mạng được tạo bởi các user thường và các chi tiết về việc thực hiện mạng sẽ bị ẩn đi khỏi người dùng.
- Provider network được tạo với quyền nhà quản trị hệ thống, xác định cách thức hoạt động vật lý của mạng, và thường khớp với mạng đã tồn tại trong trung tâm dữ liệu. Là mạng cung cấp khả năng truy cập ra ngoài Internet cho tất cả các máy ảo trong hệ thống Openstack.

Provider Networks

Mạng nhà cung cấp cho phép các quản trị viên tạo ra các mạng mà ánh xạ trực tiếp tới các mạng vật lý sẵn có trong trung tâm dữ liệu. Nên nó thường được sử dụng để cung cấp khả năng truy cập ra mạng public cho các project. Nó cũng có thể được sử dụng để tích hợp với VLAN đã có sẵn trong mạng vật lý của hệ thống.

Nhìn chung, Provider networks cũng cấp sự đơn giản, hiệu quả và sự minh bạch, linh hoạt trong chi phí. Mặc định chỉ có duy nhất người quản trị mới có thể tạo hoặc cập nhật provider networks bởi nó yêu cầu phải cấu hình thiết bị vật lí.

Bên cạnh đó, provider networks chỉ quản lí kết nối ở layer 2 cho máy ảo, vì thế nó thiếu đi một số tính năng ví dụ như định tuyến và gán floating IP.



Hình 2.9: Mô hình kiến trúc của provider network

Project network (hay Self-service network)

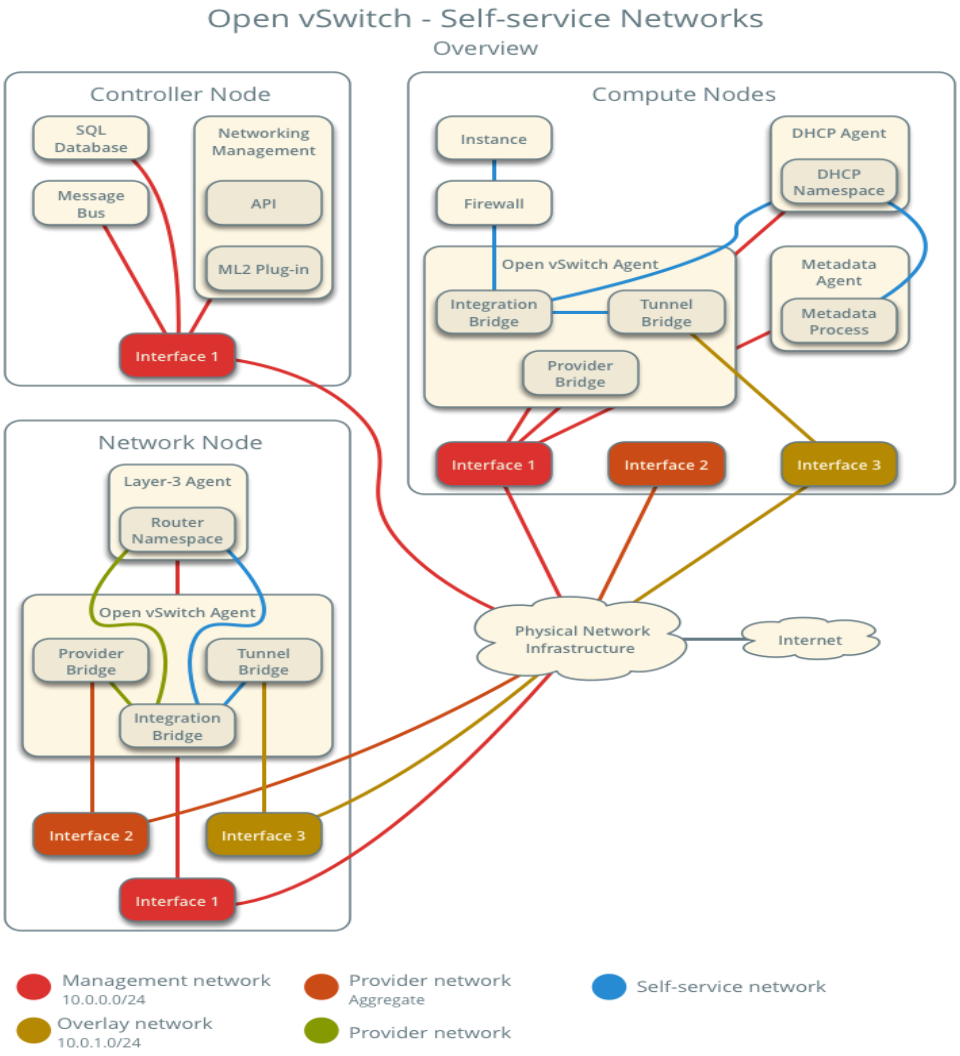
Self-service networks được ưu tiên ở các projects thông thường để quản lí các mạng mà không cần quản trị viên (quản lí network trong project). Các mạng này là ảo và nó yêu cầu các routers ảo để giao tiếp với provider và external networks. Self-service networks cũng đồng thời cung cấp dịch vụ DHCP và metadata cho máy ảo.

Trong hầu hết các trường hợp, self-service networks sử dụng các giao thức như VXLAN hoặc GRE bởi chúng hỗ trợ nhiều hơn là VLAN tagging (802.1q). Bên cạnh đó, Vlans cũng thường yêu cầu phải cấu hình thêm ở tầng vật lí.

Với IPv4, self-service networks thường sử dụng dải mạng riêng và tương tác với provider networks thông qua cơ chế NAT trên router ảo. Floating IP sẽ cho phép kết nối tới máy ảo thông qua địa chỉ NAT trên router ảo. Trong khi đó, IPv6 self-service networks thì lại sử dụng dải IP public và tương tác với provider networks bằng giao thức định tuyến tĩnh qua router ảo.

Trái ngược lại với provider networks, self-service networks buộc phải đi qua layer-3 agent. Vì thế việc gặp sự cố ở một node có thể ảnh hưởng tới rất nhiều các máy ảo sử dụng chúng.

Các user có thể tạo các project networks cho các kết nối bên trong project. Mặc định thì các kết nối này là riêng biệt và không được chia sẻ giữa các project.

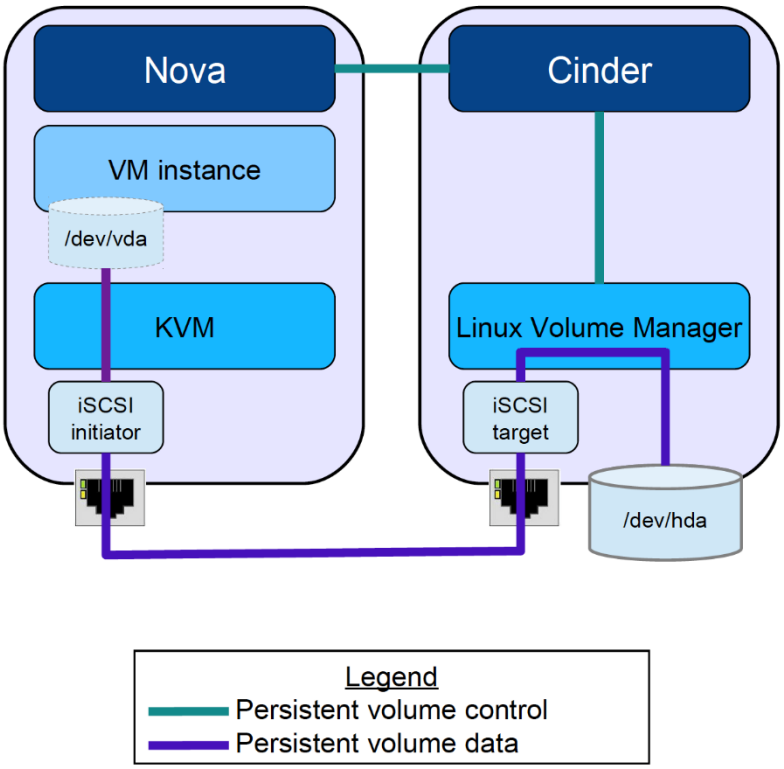


Hình 2.10: Mô hình kiến trúc của Self-service network

CINDER – Block Storage Service

Dịch vụ block storage – CINDER : cung cấp các tài nguyên lưu trữ dạng block bền vững mà máy ảo compute có thể sử dụng. Đây là dịch vụ lưu trữ thứ cấp giống với dịch vụ lưu trữ Amazone Elastic Block Storage (EBS). Ngoài ra, bạn có thể ghi image vào một thiết bị lưu trữ Block storage cho Compute để sử dụng như là boot một máy ảo.

Dịch vụ Block storage khác với Amazon EBS. Dịch vụ Block storage không cung cấp giải pháp lưu trữ dùng chung như NFS. Với dịch vụ Block Storage, bạn có thể gán một thiết bị chỉ với một máy ảo.



Hình 2.11: Cinder cung cấp Block storage cho các máy ảo nova

Các chức năng:

- Cung cấp và quản lý các tài nguyên lưu trữ dạng persistent block storage (volume) cho các máy ảo.
- Các volume này có thể được tách từ máy ảo này và gán lại vào một máy ảo khác, mà dữ liệu được giữ nguyên không bị thay đổi. (Hiện tại, một volume chỉ có thể được gán (attached) và một máy ảo tại một thời điểm)
- Các volume tồn tại độc lập với các máy ảo (tức là khi máy ảo bị xóa thì volume không bị xóa).
- Phân chia tài nguyên lưu trữ thành các khối gọi là Cinder volume.
- Cung cấp các API như là tạo, xóa, backup, restore, tạo snapshot, clone volume và nhiều hơn nữa. Những API này thực hiện bởi các backend lưu trữ mà được cinder hỗ trợ.
- Các kiến trúc plugin drive cho phép nhiều lựa chọn làm backend storage.

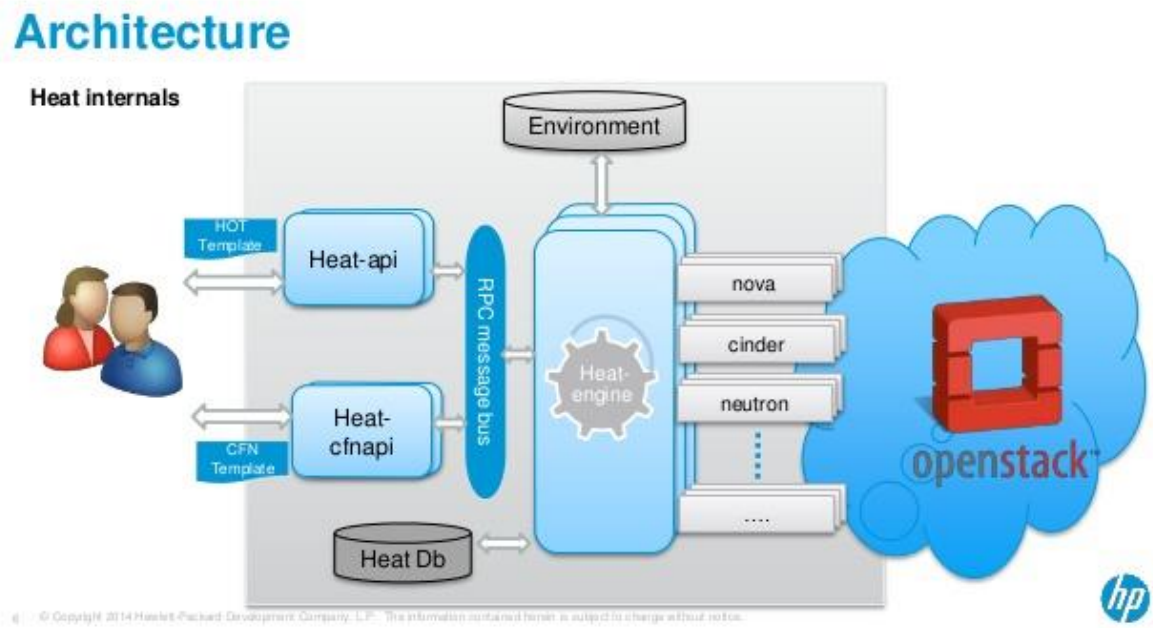
2.2.3. Một số project triển khai NFV trong Openstack

HEAT - Orchestration service

Heat là dịch vụ để triển khai các ứng dụng sử dụng định dạng template thông qua OpenStack Rest API.

Heat là sự phối hợp của các thành phần để triển khai OPS cơ bản nhất. Heat quản lý về hạ tầng (infrac) nhưng nó cũng hỗ trợ các phần mềm quản lý cấu hình. Heat cung cấp khả năng cho phép người dùng tự định nghĩa ứng dụng của mình thông qua các template.

Kiến trúc của Heat



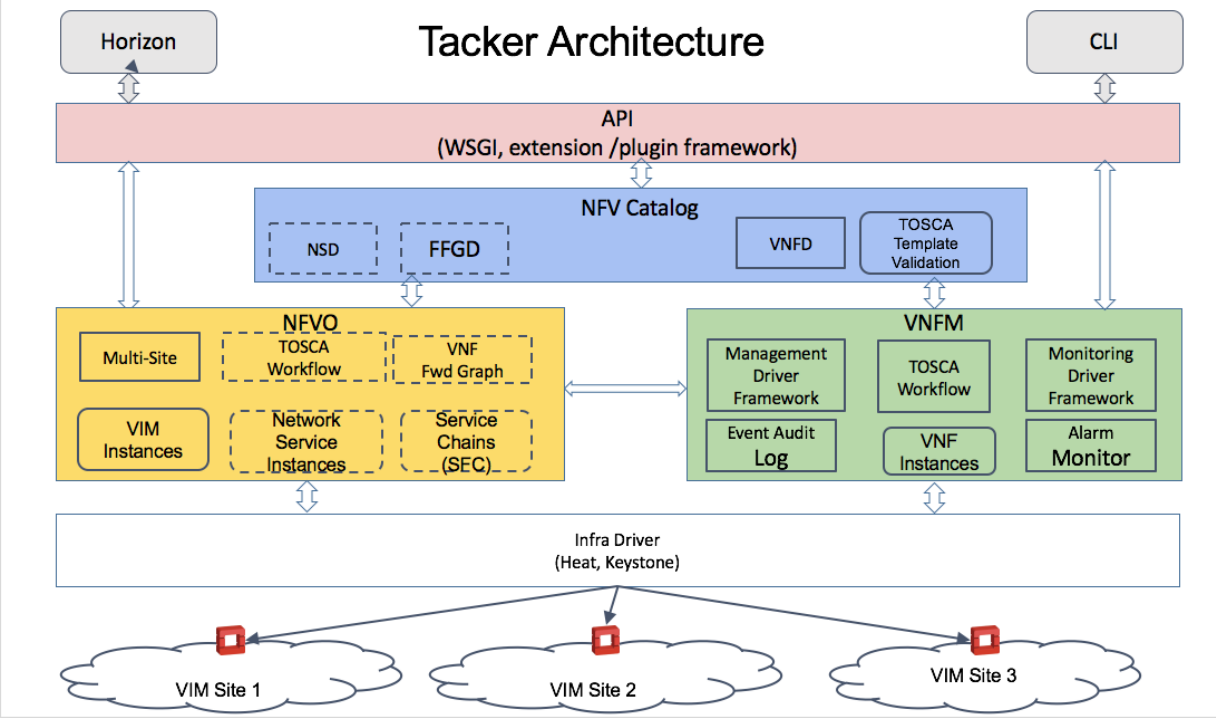
Hình 2.12: Kiến trúc của project HEAT

Các tính năng của Heat:

- Cung cấp một template dựa trên Orchestrator để mô tả ứng dụng trên cloud bằng cách xử lý các lời gọi tới các Openstack APIs thích hợp để tạo ra các ứng dụng trên cloud.
- Mô tả kiến trúc của các ứng dụng cloud bằng các file văn bản mà con người có thể đọc và viết được, quản lý chúng thông qua các version control tools.
- Templates xác định mối quan hệ giữa các nguồn tài nguyên (ví dụ như volume nào đc nối với server nào). Điều này cho phép Heat gọi đến các Openstack APIs để tạo nên tất cả cơ sở hạ tầng một cách chính xác để bạn có thể chạy các ứng dụng của mình
- Tích hợp với tất cả các thành phần khác của Openstack. Template cho phép tạo hầu hết các loại tài nguyên của Openstack (như máy ảo, floating ips, volumes, security groups, users, ...) cũng như cung cấp một số tính năng cao hơn.
- Heat chủ yếu quản lý cơ sở hạ tầng, nhưng cũng có một số template tương thích tốt với các công cụ cấu hình phần mềm (ví dụ: Puppet, Ansible)
- Các nhà khai thác có thể tùy chỉnh Heat với các tính năng bổ sung.

TACKER - NFV Orchestration Service

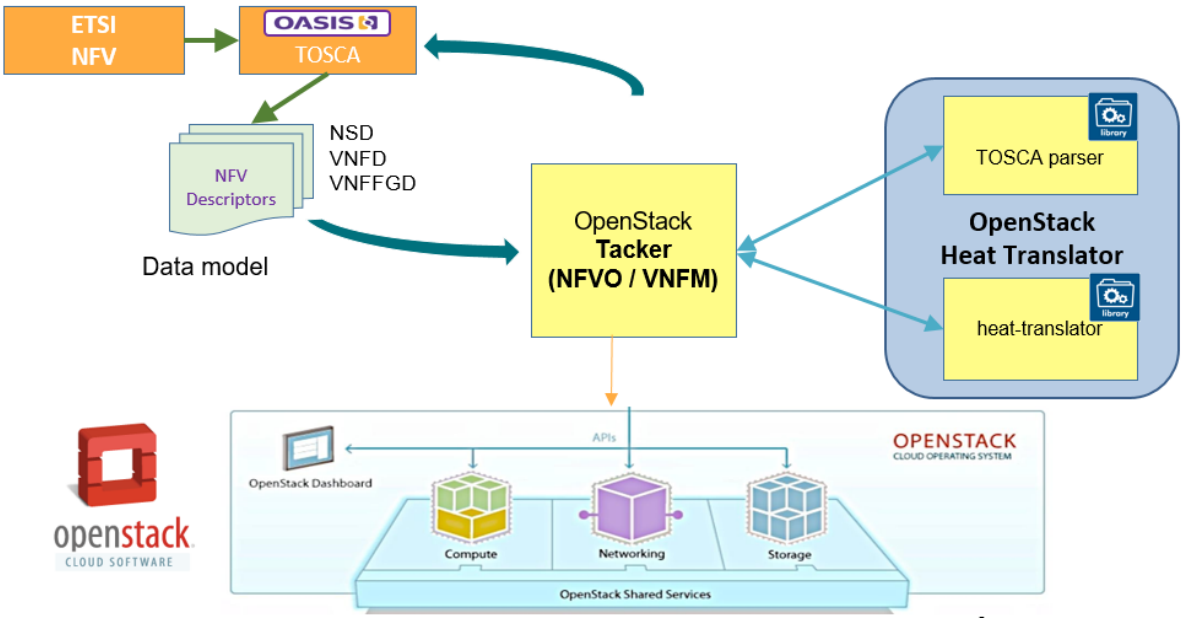
Tacker là một dự án chính thức của OpenStack được sử dụng để xây dựng một khối quản lý chung cho các VNF (General VNF Manager - VNFM) và NFV Orchestrator (NFVO) để triển khai và vận hành các dịch vụ mạng và các chức năng mạng ảo trên một nền tảng cơ sở hạ tầng NFV như OpenStack.



Hình 2.13: Kiến trúc Tacker trong Openstack

Tacker sử dụng các file TOSCA như là template để tạo ra các VNF.

TOSCA, viết tắt của Topology and Orchestration Specification for Cloud Applications, là một ngôn ngữ được phát triển gần đây bởi OASIS, một tổ chức tiêu chuẩn, để mô tả tô pô của một đám mây và các ứng dụng chạy trên nó.



Hình 2.14: Mối liên hệ giữa file Tosca và Tacker trong Openstack

Các file TOSCA NFV xác định một mô hình dữ liệu NFV cụ thể sử dụng ngôn ngữ TOSCA.

2.3. Tổng quan về Công nghệ Ảo hóa chức năng mạng – NFV

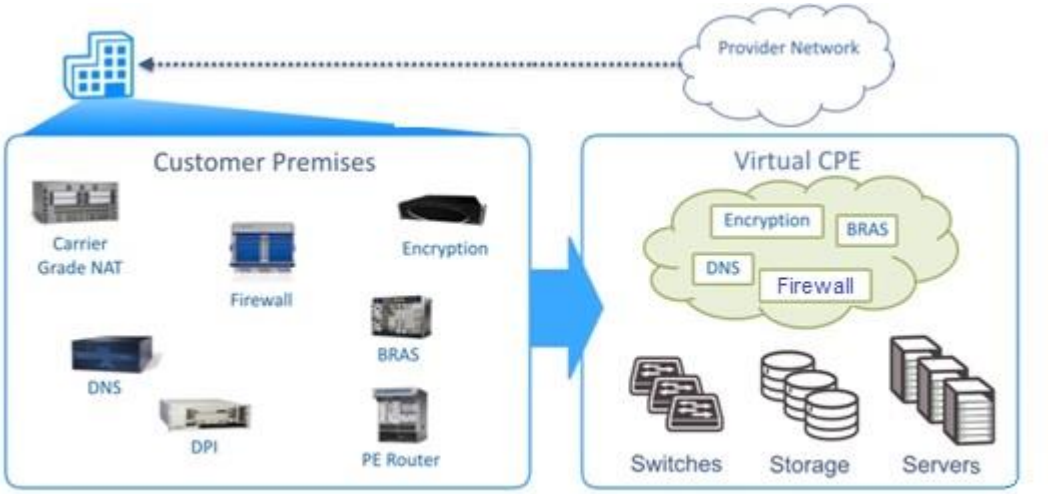
Hệ thống mạng viễn thông hiện tại được vận hành sử dụng các thiết bị phần cứng độc quyền của nhiều nhà cung cấp khác nhau. Việc vận hành dịch vụ mạng mới đồng nghĩa với việc sử dụng thêm nhiều thiết bị hơn, đòi hỏi phải mở rộng không gian để triển khai, đặt ra vấn đề về việc chi phí năng lượng ngày càng tăng, thách thức về vốn đầu tư, yêu cầu các kỹ năng cần thiết để thiết kế, tích hợp và vận hành các thiết bị mạng vật lý càng ngày càng phức tạp. Ngoài ra vòng đời các thiết bị phần cứng cũng không dài, yêu cầu có kế hoạch về chu kì thiết kế - tích hợp - triển khai phù hợp. Tệ hơn, vòng đời của phần cứng đang ngày một ngắn dần do sự phát triển nhanh chóng của các dịch vụ và công nghệ, gây khó khăn cho việc triển khai các network services mới để thu về lợi nhuận, hạn chế sự đổi mới bởi vì xu hướng hiện tại là hướng về các giải pháp mạng lưới tập trung.

Network Functions Virtualization (NFV) ra đời mang đến cách thức mới để thiết kế, triển khai và quản lý các dịch vụ mạng, sử dụng các công nghệ ảo hóa tiêu chuẩn hiện tại để hợp nhất nhiều loại thiết bị mạng trên các máy chủ, switches và storages theo tiêu chuẩn công nghiệp được đặt trong các trung tâm dữ liệu, các nút mạng và tại nhà của người dùng cuối. NFV tách biệt các chức năng mạng (NAT, firewalling, intrusion detection, DNS, caching) khỏi các thiết bị vật lý và triển khai dưới hình thức phần mềm và có thể chạy trên các máy chủ vật lý truyền thống, đồng thời có thể di trú hoặc được khởi tạo trên nhiều vị trí trong hệ thống mạng theo yêu cầu mà không cần phải triển khai thiết bị mới như trước đây.

Những lợi ích của NFV có thể kể tới như:

- Giảm chi phí vốn: giảm chi phí để mua những phần cứng chuyên dụng để triển khai các chức năng mạng, hỗ trợ mô hình pay-as-you-grow (chi trả theo mức độ mở rộng), bớt lãng phí dự phòng không cần thiết.

- Giảm chi phí vận hành: thu hẹp không gian, chi phí về năng lượng và làm mát cho các thiết bị, đơn giản hóa việc quản lý các dịch vụ mạng.
- Cung cấp nhanh chóng và linh hoạt: NFV cho phép nhanh chóng mở rộng hoặc thu hẹp quy mô dịch vụ để giải quyết những thay đổi trong yêu cầu của khách hàng bằng việc triển khai trong các máy chủ tiêu chuẩn công nghiệp.
- Tăng tốc đưa dịch vụ mới vào thương mại: giảm bớt thời gian triển khai dịch vụ mạng mới đáp ứng thay đổi của doanh nghiệp, nắm bắt những cơ hội thị trường mới và tăng lợi nhuận khi đầu tư vào các dịch vụ mới đó. Thêm vào đó, nhà cung cấp có thể thử nghiệm các dịch vụ mới mà ít gặp rủi ro hơn và đáp ứng tốt hơn nhu cầu ngày một thay đổi của khách hàng.

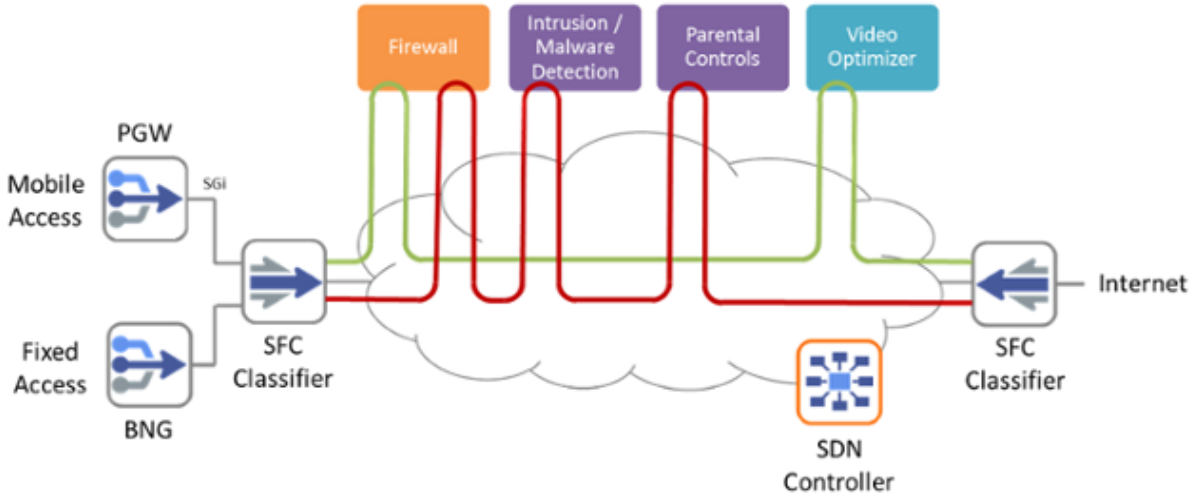


Hình 2.15: Ảo hóa chức năng mạng

2.4. Tổng quan về chuỗi các chức năng mạng (Service Function Chaining – SFC)

Đối với nhà cung cấp dịch vụ, để triển khai và cung cấp hoàn chỉnh một dịch vụ đầu cuối cho khách hàng thường yêu cầu sử dụng nhiều thiết bị cung cấp các dịch vụ mạng. Các thiết bị này đặt ở nhiều nơi trong hệ thống mạng của nhà cung cấp, trong trung tâm dữ liệu hoặc giữa các trung tâm dữ liệu của nhà cung cấp với nhau.

Trong đó bao gồm hai loại: thiết bị chuyển tiếp (forwarding device) như router, switch và thiết bị biến đổi, điều tra, lọc và xử lý lưu lượng (middlebox) như NAT (Network Address Translators), firewall (tường lửa), DPI (Deep Packet Inspection – Kiểm soát gói mức độ sâu), IDS/IPS (Intrusion Detection System/ Intrusion Prevention System – Hệ thống phát hiện / ngăn chặn xâm nhập). Service Function Chaining (SFC) hay chuỗi các dịch vụ mạng là tập hợp các dịch vụ mạng theo trật tự nhất định và điều khiển lưu lượng đi qua chuỗi dịch vụ đó.



Hình 2.16: Một chuỗi các chức năng mạng

Mối quan hệ giữa NFV và SFC

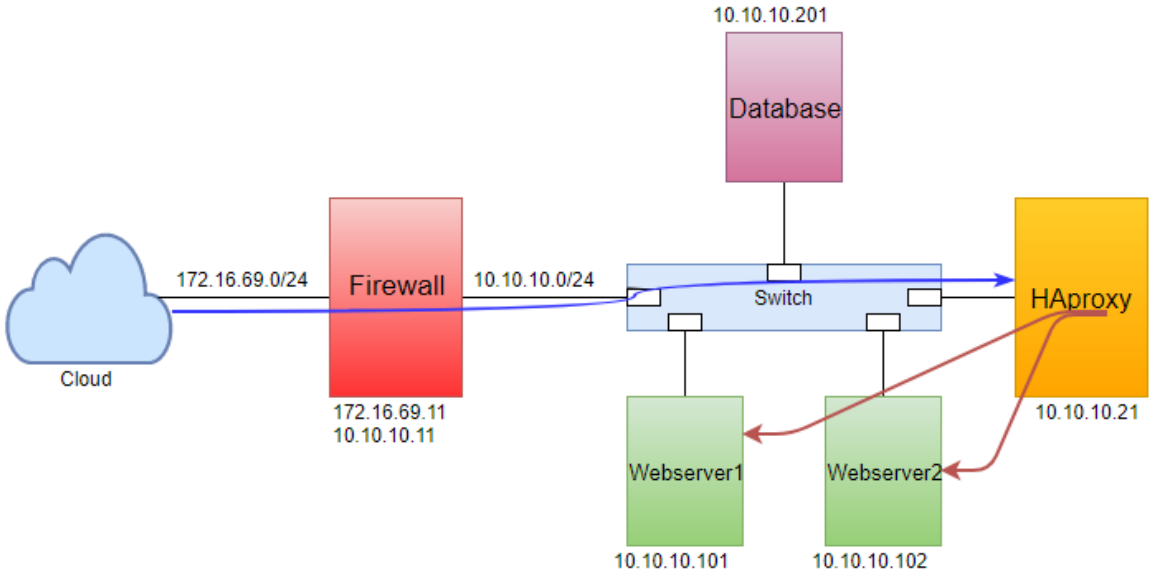
Chuỗi chức năng mạng hiện nay phần lớn vẫn sử dụng các thiết bị chuyên dụng, dẫn đến tốn kém về chi phí và phụ thuộc vào nhà sản xuất.

NFV ra đời đã giải quyết được vấn đề này bằng việc thay vì phải sử dụng các thiết bị chuyên dụng để làm các chức năng mạng, chúng ta có thể sử dụng các server vật lý được cài các phần mềm giả lập các chức năng mạng như firewall, loadblance,… Qua đó, các nhà mạng có thể tiết kiệm chị phí vận hành, và cũng bớt lệ thuộc vào các nhà sản xuất thiết bị mạng.

3. CHƯƠNG 3: XÂY DỰNG CHUỖI SFC DỰA TRÊN CÔNG NGHỆ NFV

3.1. Xây dựng mô hình triển khai

3.1.1. Mô hình



Mô hình gồm 1 Firewall, 1 HA proxy, 2 Webserver và 1 Database.

- Firewall: 3 CPU, 4GB RAM
- HA proxy: 3CPU, 4GB RAM
- Webserver1: 2CPU, 2GB RAM
- Webserver2: 2CPU, 2GB RAM
- Database: 1CPU, 1 GB RAM

Các thông số mạng như hình vẽ.

Kịch bản:

- Người dùng sẽ request đến IP public của Firewall là 172.16.69.11. Firewall sẽ thực hiện chức năng NAT và chuyển tiếp gói tin sang HA Proxy.
- HA Proxy thực hiện cân bằng tải chuyển tiếp gói sang Webserver1 hoặc Webserver2.

3.1.2. Khái quát về các khối chức năng trong chuỗi

Firewall

- Chúng ta sử dụng **`iptables`** làm Firewall.

- Iptables là chương trình tiện ích user-space cho phép người quản trị hệ thống cấu hình tables được cung cấp bởi Linux kernel firewall (được thực hiện bởi các modules Netfilters khác nhau) và chains và rules nó lưu trữ. Modules kernels và chương trình khác nhau được sử dụng cho các giao thức khác nhau: iptables áp dụng cho IPv4, ip6tables áp dụng cho IPv6, arptables áp dụng cho ARP và ebtables cho Ethernet frames.

- Netfilter/Iptables được tích hợp vào Linux kernel 2.3.x. Trước iptables, Linux dùng ipchains trong Linux kernel 2.2.x và ipfwadm trong Linux kernel 2.0.x.

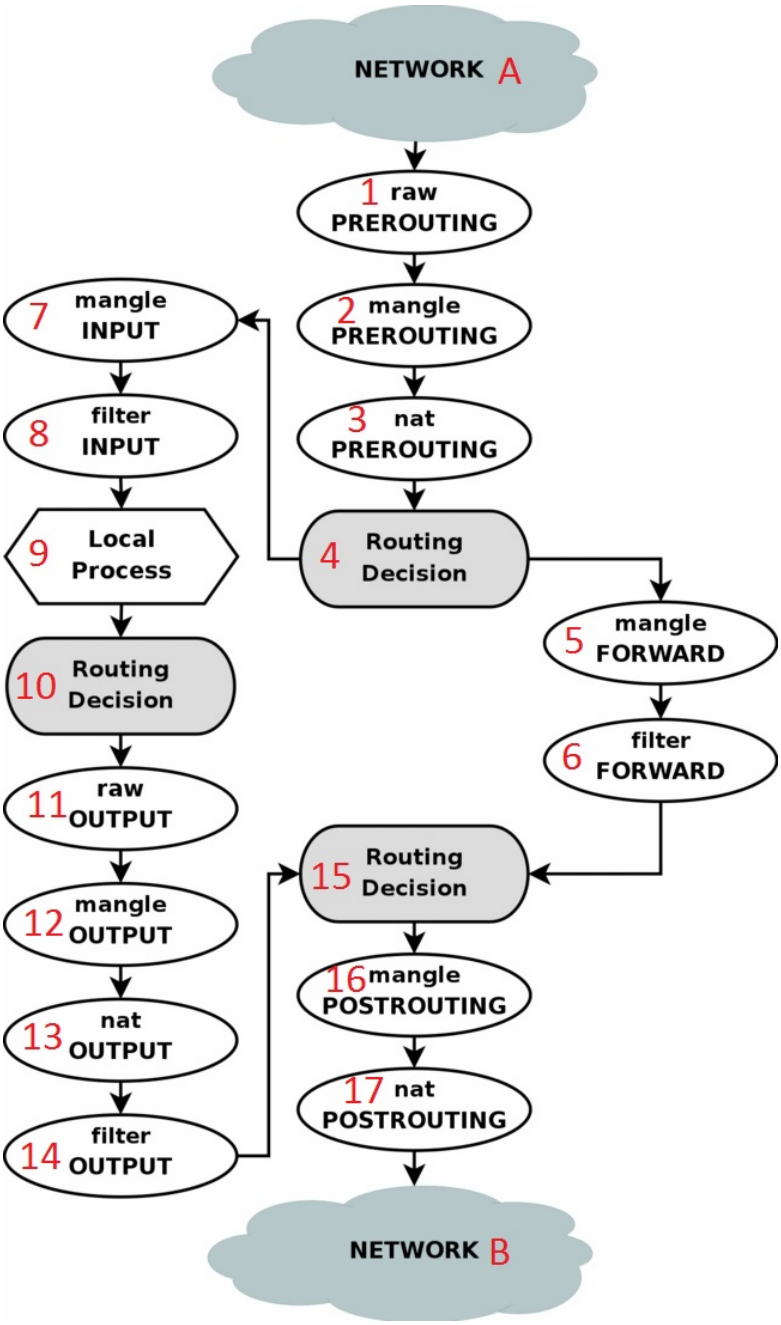
- Iptables yêu cầu quyền ưu tiên được vận hành và phải được thực hiện bởi user root.

- Stateless và Stateful Packet Filtering

- Stateless Packet Filtering là dạng bộ lọc không biết được quan hệ của những packet vào với packet đi trước nó hoặc đi sau nó, gọi là cơ chế lọc không phân biệt được trạng thái của các packet, trong kernel 2.0 và 2.2 thì ipfwadm và ipchains chỉ thực hiện được đến mức độ này. Loại firewall này khó có thể bảo vệ được mạng bên trong trước các kiểu tấn công phá hoạt như DoS, SYN flooding, SYN cookie, ping of death, packet fragmentaion hay các hacker chỉ cần dùng công cụ nmap chẳng hạn là có thể biết được trạng thái của các host nằm sau firewall. Điều này không xảy ra với Stateful firewall.

- Stateful Packet Filtering Với mọi packet đi vào mà bộ lọc có thể biết được quan hệ của chúng như thế nào đối với packet đi trước hoặc đi sau nó, ví dụ như các trạng thái bắt tay 3 bước trước khi thực hiện 1 phiên kết nối giao thức TCP/IP (SYN, SYN/ACK, ACK) gọi là firewall có thể phân biệt được trạng thái của các packet. Với loại firewall này, chúng ta có thể xây dựng các quy tắc lọc để có thể ngăn chặn được ngay cả các kiểu tấn công phá hoại như SYN flooding hay Xmas tree... Iptables thuộc loại firewall này. Hơn thế nữa Iptables còn hỗ trợ khả năng giới hạn tốc độ kết nối đối với các kiểu kết nối khác nhau từ bên ngoài, cực kỳ hữu hiệu để ngăn chặn các kiểu tấn công từ chối phục vụ (DoS) mà hiện nay vẫn là mối đe doạ hàng đầu đối vói các website trên thế giới. Một đặc điểm nổi bật nữa của Iptables là nó hỗ trợ chức năng dò tìm chuỗi tương ứng (string pattern matching), chức năng cho phép phát triển firewall lên một mức cao hơn, có thể đưa ra quyết định loại bỏ hay chấp nhận packet dựa trên việc giám sát nội dung của nó. Chức năng này có thể được xem như là can thiệp được đến mức ứng dụng như HTTP, TELNET, FTP... mặc dù thực sự Netfilter Iptables vẫn chỉ hoạt động ở mức mạng (lớp 3 theo mô hình OSI 7 lớp).

-Packet flow của iptables



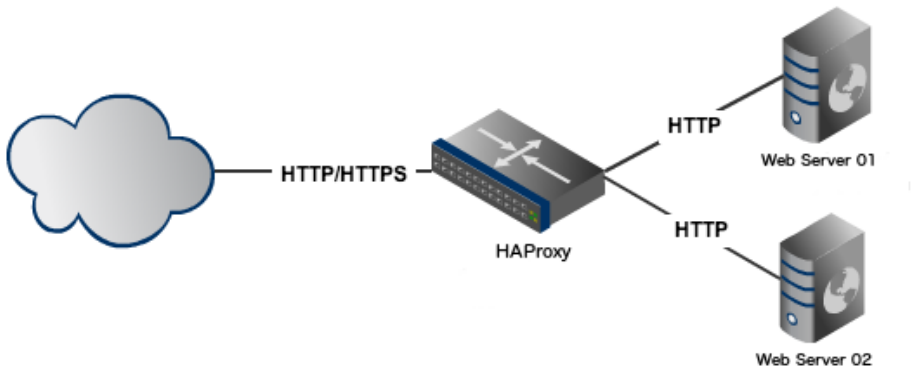
Hình 3.1: Packet Flow của IPtables

HAproxy

High Availability hay còn được gọi là HA có nghĩa là "Độ sẵn sàng cao", những máy chủ, thiết bị loại này luôn luôn sẵn sàng phục vụ, người sử dụng không cảm thấy nó bị trục trặc, hỏng hóc gây gián đoạn. Để đảm bảo được điều đó, tối thiểu có một cặp máy, thiết bị chạy song song, liên tục liên lạc với nhau, cái chính hỏng, cái phụ sẽ lập tức biết và tự động thay thế. Một ví dụ đơn giản nhất là một số máy chủ có hai bộ nguồn, tự động thay thế nóng cho nhau.

Load Balancing hay cân bằng tải là một phương pháp phân phối khối lượng tải trên nhiều máy tính hoặc một cụm máy tính để có thể sử dụng tối ưu các nguồn lực, tối đa hóa thông lượng, giảm thời gian đáp ứng và tránh tình trạng quá tải trên máy chủ.

HAProxy, viết tắt của High Availability Proxy, là phần mềm mã nguồn mở cân bằng tải TCP/HTTP và giải pháp proxy mã nguồn mở phổ biến, có thể chạy trên Linux, Solaris, và FreeBSD. Nó thường dùng để cải thiện hiệu suất (performance) và sự tin cậy (reliability) của môi trường máy chủ bằng cách phân tán lưu lượng tải (workload) trên nhiều máy chủ (như web, application, database). Nó cũng thường dùng cho môi trường cao cấp gồm: GitHub, Imgur, Instagram, và Twiter.



Hình 3.2: HAproxy

3.2. Dựng hệ thống triển khai SFC

Trên Firewall, thực hiện các thao tác sau:

- Thực hiện DNAT với những gói tin TCP có port đích là 80, nat IP đích 172.16.69.11 thành IP 10.10.10.21.
- Thực hiện SNAT để nat IP nguồn của gói tin TCP đi từ ngoài mạng internet vào, có port đích là 80 thành IP 10.10.10.11.
- Đồng ý cho các bản tin ICMP từ ngoài mạng internet tới.
- Chặn mọi bản tin khác.

Trên HAproxy:

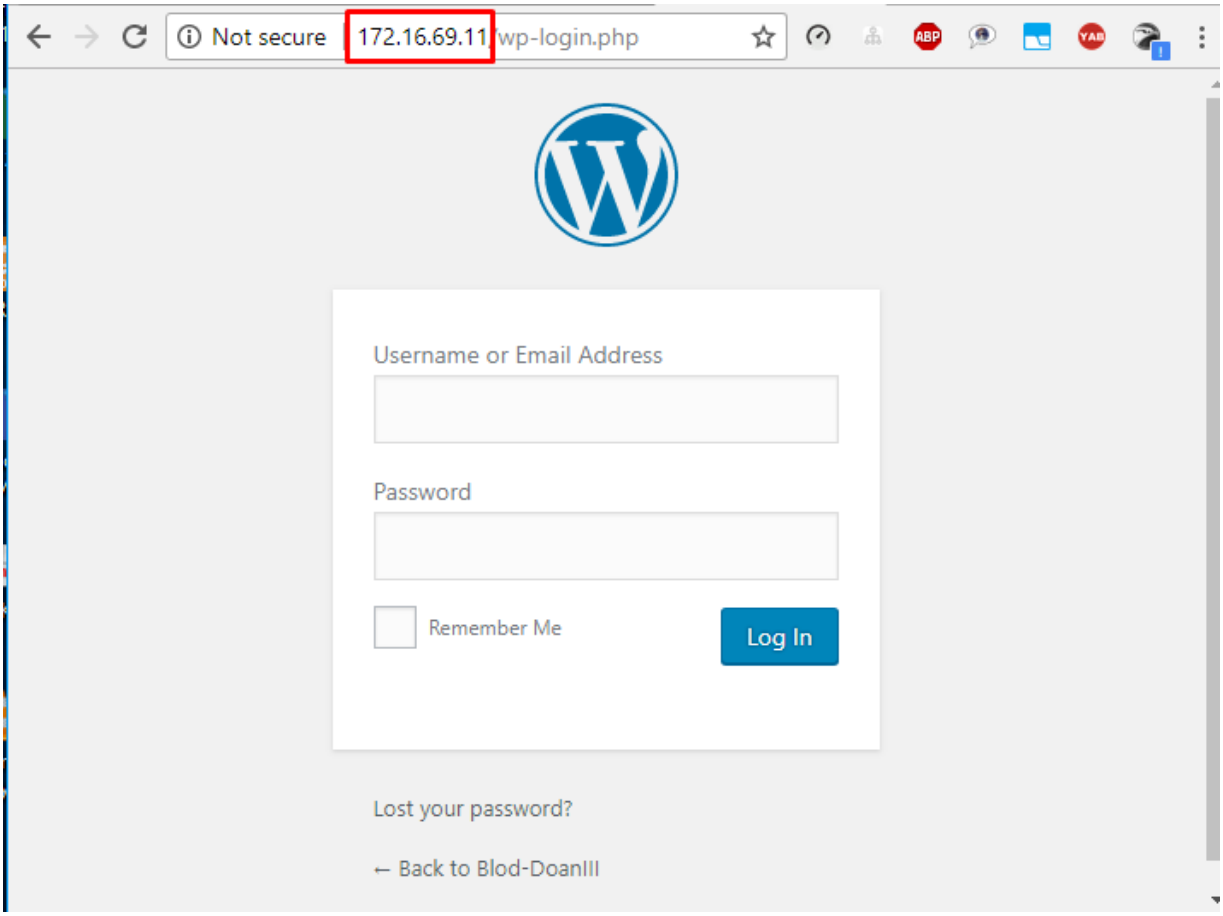
- Thực hiện cấu hình cho phép cân bằng các yêu cầu gửi tới web server.
- Sử dụng thuật toán roundrobin để chuyển các yêu cầu lần lượt tới 2 web server.
- Nếu một web server gặp sự cố bị down thì chuyển tất cả các yêu cầu tới web server còn lại.

Trên 2 máy Webserver:

- Sử dụng phần mềm apache2.
- Cài thêm wordpress và cùng truy vấn đến Database.

Trên máy Database: Cài mysql-server làm hệ quản trị cơ sở dữ liệu cho 2 web server.

3.3. Kết quả và đánh giá



Hình 3.3: Kết quả thu được khi client gửi request tới hệ thống

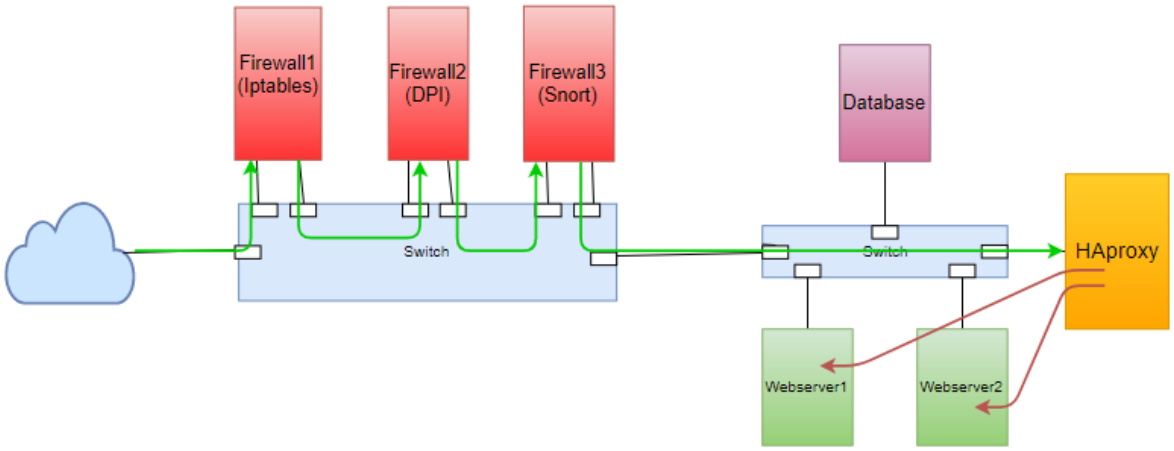
Thực hiện thành công đưa luồng dữ liệu đi đúng hướng như chuỗi SFC đề ra. Khi truy cập vào địa chỉ của Firewall, luồng dữ liệu sẽ được chuyển tiếp tới HAproxy và phân tải đều cho 2 web server.

Khó khăn:

- Gặp khó khăn trong điều khiển luồng http trong Openvswitch.

3.4. Định hướng phát triển

Triển khai chuỗi SFC trên OpenStack sử dụng tacker.



Kết hợp SDN và NFV để triển khai chuỗi SFC trên môi trường Openstack sử dụng Tacker, như hình vẽ, bao gồm:

- 3 Firewall: gồm có Iptables, DPI, Snort
- 1 HA proxy
- 2 Webserver
- 1 Database

KẾT LUẬN

Dựa vào những kết quả đạt được và hạn chế còn tồn tại trong tương lai, nhóm thực hiện sẽ tiếp tục phát triển đề tài này theo các hướng sau:

- Xây dựng hệ thống NFV kết hợp SFC với nền tảng VIM đảm bảo tính sẵn sàng cao.
- Đo đạc tham số thực tế của các chức năng mạng và chuỗi dịch vụ mạng, từ đó làm đầu vào cho nghiên cứu học thuật sau này.
- Đưa ra vấn đề học thuật liên quan tới việc cấp phát tài nguyên và điều phối vị trí triển khai dịch vụ mạng đầu cuối hoàn chỉnh nhằm thỏa mãn một số tiêu chí như: tối ưu năng lượng tiêu thụ, tối ưu băng thông và tài nguyên sử dụng của hạ tầng vật lý. Mô hình hóa hệ thống và đưa ra thuật toán thực hiện giải quyết vấn đề học thuật đưa ra.

TÀI LIỆU THAM KHẢO

[1] https://docs.openstack.org/pike/ truy cập cuối cùng ngày 10/01/2018

[2] https://wiki.openstack.org/wiki/Tacker truy cập cuối cùng ngày 10/01/2018

[3] http://cbonte.github.io/haproxy-dconv/1.9/configuration.html truy cập cuối cùng ngày 10/01/2018

[4] http://www.faqs.org/docs/iptables/ truy cập lần cuối ngày 10/01/2018