**HỌC VIỆN KỸ THUẬT MẬT MÃ**

**KHOA CÔNG NGHỆ THÔNG TIN**

**---□**🕮**□---**



**THỰC TẬP CƠ SỞ CHUYÊN NGÀNH**

ĐỀ TÀI: BẢO MẬT TRÊN ĐÁM MÂY RIÊNG SỬ DỤNG OPENSTACK

Ngành: An toàn thông tin

Sinh viên thực hiện**:**

**BÙI VĂN NAM (AT160631)**

**TRẦN VĂN THIỆN (AT160647)**

**NGUYỄN THỊ TRÀ MY(AT160630)**

Giáo viên hướng dẫn:

**TS. PHẠM VĂN HƯỞNG**

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

Hà Nội, 2022

**NHẬN XÉT VÀ CHO ĐIỂM CỦA GIÁO VIÊN**

.................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................

MỤC LỤC

**MỤC LỤC** 3

**LỜI MỞ ĐẦU** 6

Tính cấp thiết của đề tài 6

Mục tiêu thực hiện đồ án 6

**DANH MỤC BẢNG BIỂU, HÌNH VẼ 7**

**CHƯƠNG I. GIỚI THIỆU VỀ CLOUD COMPUTING**

**1.1. Cloud computing là gì ?**

**1.2. Mô hình 5-4-3 trong cloud computing**

**1.2.1. Năm đặc điểm thiết yếu của cloud computing**

**1.2.2. Bốn mô hình triển khai**  9

**1.2.3. Ba Mô hình dịch vụ**  12

**1.2.4. Các nhà cung cấp điện toán đám mây hàng đầu**

**CHƯƠNG II. GIỚI THIỆU VỀ OPENSTACK**

**2.1 Tổng quan** 21

**2.2. Kiến trúc các thành phần chính của OpenStack** 21

**2.3. Tổng quan về các các thành phấn chính của OpenStack**

**2.3.1. Horizon ( Dashboard service )**

**2.3.1.1. Khái niệm**

**2.3.2. Keystone ( Identity Service )**

**2.3.2.1. Khái niệm**

## 2.3.2.2. Chức năng chính Keystone

#### 2.3.2*.*3 Ý nghĩa của các thành phần trên

**2.3.3. Nova ( Compute Service )**

**2.3.3.1. Khái niệm**

**2.3.3.2 Các dịch vụ của Nova**

## 2.3.3.2 Các thành phần của Nova

**2.3.4. Neutron ( Netwok Service )**

**2.3.4.1. Tổng quan**

**2.3.4.2. Neutron tích hợp với 1 số thành phần OpenStac khác**

**2.3.4.3. Các thành phần của Neutron**

**2.3.4.4. Các loại loại mạng neutron cung cấp**

**2.3.4.5. Các dịch vụ phổ biến**

**2.3.5. Glance ( Image Service )**

**2.3.5.1 Tổng quan**

**2.3.5.2. Thành phần của glance**

**2.3.5.3. Kiến trúc của Glance**

**2.3.5.4. Hộ trợ các loại format disk của Glance**

**2.3.6. Cinder ( Block Storage Service )**

**2.3.6.1. Khái niệm**

**2.3.6.2. Các thành phần của Cinder**

**2.3.6.3. Kiến trúc của Cinder**

**2.3.7 Swift (Object Storage Service )**

**2.3.7.1. Khái niệm**

**2.3.7.2. Kiến trúc logic**

**2.3.7.3. Kiến trúc vật lý**

**2.3.7.4. Kiến trúc xác thực trong swift**

**CHƯƠNG III. BẢO MẬT CHO ĐÁM MÂY SỬ DỤNG OPENSTACK**

## 3.1: Tổng quan về bảo mật trên đám mây

**3.1.1. Khái niệm**

**3.1.2. Mô hình bảo mật đám mây**

**3.2. Tăng cường bảo mật cho đám mây sử dụng Openstack**

**3.2.1. Theo từng mô hình**

**3.2.1.1. Public Cloud**

#### 3.2.1.2. Private Cloud

### 3.2.2. Theo từng vai trò

**3.2.2.1. Người vận hành OpenStack**

#### 3.2.2.2. Người sử dụng cuối

#### 

**CHƯƠNG IV. THỰC NGHIỆM TRIỂN KHAI OPENSTACK VÀ TĂNG CƯỜNG BẢO MẬT CHO ĐÁM MÂY IAAS SỬ DỤNG OPENSTACK**

# 

# LỜI MỞ ĐẦU

## Tính cấp thiết của đề tài

Cloud Secutity là thuật ngữ viết tắt từ Cloud Computing Security, nghĩa là bảo mật điện toán đám mây.Đây là một khái niệm dịch vụ không còn xa lạ gì trong các tổ chức doanh nghiệp trên thế giới, nói về giải pháp toàn diện cho các vấn đề công nghệ thông tin trên đám mây. Sự phát triển của dịch vụ điện toán đám mây là xu thế chung, và các doanh nghiệp bắt đầu chuyển và lưu giữ khối lượng dữ liệu lớn, các ứng dụng của tổ chức lên đám mây để dễ làm việc ở mọi nơi khi không có tại văn phòng… Đi kèm với sự phát triển, bên cạnh đó luôn luôn tồn tại các mối đe dọa an ninh dữ liệu. Vì vậy sự cần thiết Cloud Security càng quan trọng hơn, nếu tổ chức nào lỏng lẻo trong dịch vụ này, không chú trọng quan tâm coi như sẽ đánh mất nhiều lợi ích to lớn của nó và chắc chắn là đối mặt với nguy cơ sụp đổ rất nhanh chóng.

## Mục tiêu thực hiện đồ án

Mục tiêu đặt ra khi thực hiện đồ án: Triển khai thành công đám mây IaaS sử dụng OpenStack và cấu hình, triển khai các giải pháp an toàn thông tin cho OpenStack

Trong bản báo cáo này không thể tránh khỏi những thiếu sót nên em rất mong nhận được sự đóng góp của thầy để bài báo cáo của em được hoàn thiện hơn!

**LỜI CAM ĐOAN**

Em là Bùi Văn Nam, Trần Văn Thiện và Nguyễn Thị Trà My , sinh viên lớp AT16G, ngành An Toàn Thông Tin, Học viện Kỹ thuật mật mã. Em xin cam đoan đồ án này là công trình nghiên cứu, xây dựng của riêng em dưới sự hướng dẫn tận tình của thầy **TS. Phạm Văn Hưởng**. Các nội dung nghiên cứu, kết quả thực nghiệm trong đồ án hoàn toàn trung thực.

Để hoàn thiện đồ án này, em chỉ sử dụng những tài liệu đã được liệt kê trong mục tài liệu tham khảo, ngoài ra, em không sử dụng bất cứ tài liệu nào khác mà không được liệt kê.

Nếu sai, em xin chịu mọi hình thức kỷ luật theo quy định của Học viện cũng như pháp luật.

| *Hà Nội, ngày 01 tháng 04 năm 2021*  **Sinh viên thực thiện**  **Bùi Văn Nam**  **Trần Văn Thiện**  **Nguyễn Thị Trà My** |
| --- |

**DANH MỤC BẢNG BIỂU, HÌNH VẼ**

**Hình 1.1: Tổng quan cloud computing**

**Hình 1.2: Public Cloud**

**Hình 1.3: Hybrid Cloud**

**Hình 1.4: Mô hình dịch vụ của cloud computing**

**Hình 1.5: Thị phần dịch vụ đám mây công cộng năm 2019**

**Hình 2.1: Kiến trúc Openstack**

**Hình 2.2: Kiến trúc logic Openstack (conceptual)**

**Hình 2.3: Cách hoạt động của các bộ phân trong OpenStack**

**Hình 2.4: KeyStone xác thực người dùng**

**Hình 2.5: Các thành phần liên quan đến xác thực**

**Hình 2.6: Thành phần của Glance**

**Hình 2.7: Kiến trúc của Glance**

**Hình 2.8: Các loại disk format**

**Hình 2.9: Các loại container format**

**Hình 2.10: Kiến trúc của Cinder**

**Hình 2.11: Xác thực trong swift**

**Hình 3.1: An toàn thông tin trên đám mây**

**Hình 3.2: Triển khai giao thức TLS/SSL khi truy cập vào hệ thống**

**Hình 3.3: Hệ thống bảo mật Openstack an toàn**

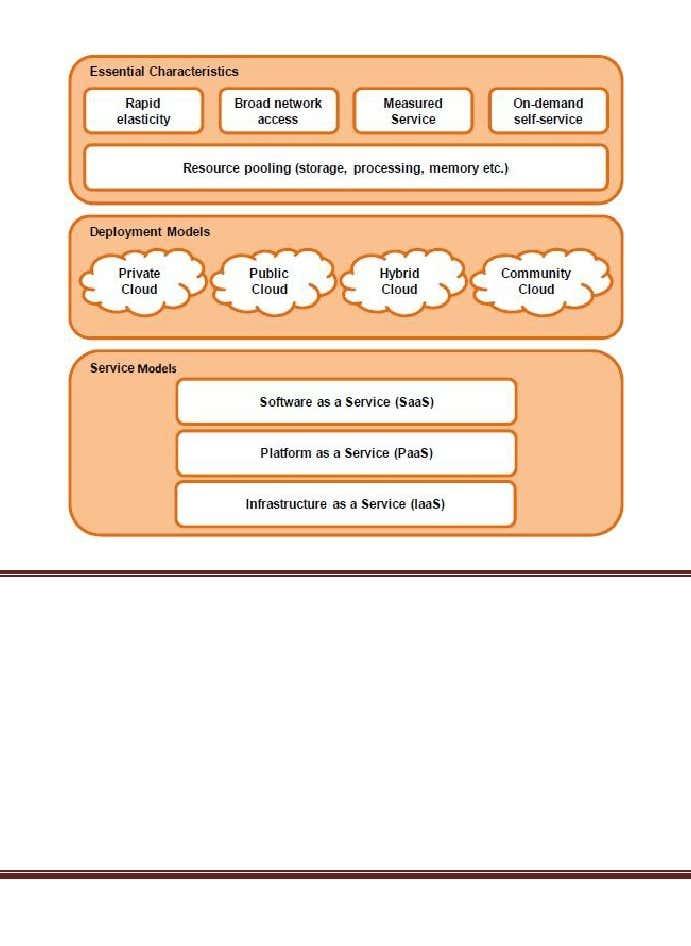
# CHƯƠNG I: GIỚI THIỆU VỀ CLOUD COMPUTING

## 1.1. Cloud computing là gì?

* Định nghĩa của Cloud Computing theo NIST:
* “Cloud Computing là mô hình dịch vụ cho phép người dùng truy cập tài nguyên điện toán dùng chung (mạng, sever, lưu trữ, ứng dụng, dịch vụ) thông qua kết nối mạng một cách dễ dàng, mọi lúc mọi nơi, theo yêu cầu. Tài nguyên điện toán đám mây này có thể được thiết lập hoặc hủy bỏ nhanh chóng bởi người dùng mà không cần sự can thiệp của Nhà cung cấp dịch vụ”.
* Người dùng có thể sử dụng các dịch vụ trên nền tảng điện toán đám mây mà không cần phải có kiến thức, kinh nghiệm về công nghệ và các hạ tầng đứng sau đó.

## 1.2. Mô hình 5-4-3 trong cloud computing

* Cloud computing bao gồm 5 đặc điểm thiết yếu, 4 mô hình triển khai, và 3 mô hình dịch vụ.



**Hình 1.1: Tổng quan cloud computing**

### 1.2.1. Năm đặc điểm thiết yếu của cloud computing :

* Tự phục vụ theo nhu cầu (On-demand self-service):
* Khi có nhu cầu, người dùng chỉ cần gửi yêu cầu thông qua trang web cung cấp dịch vụ, hệ thống của nhà cung cấp sẽ đáp ứng yêu cầu của người dùng.
* Người dùng có thể tự phục vụ yêu cầu của mình như tăng thời gian sử dụng server, tang dung lượng lưu trữ mà không cần phải tương tác trực tiếp với nhà cung cấp dịch vụ, mọi nhu cầu về dịch vụ đều được xử lý trên mỗi trường web
* Truy xuất diện rộng (Broad network access):
* Điện toán đám mây cung cấp các dịch vụ thông qua môi trường Internet Người dùng có kết nối Internet là có thể sử dụng dịch vụ.
* Điện toán đám mây ở dạng dịch vụ nên không đòi hỏi khả năng xử lý cao ở phía client. Vì vậy, người dùng có thể truy xuất bằng các thiết bị di động như điện thoại, PDA laptop,...
* Với điện toán đám mây người dùng không bị phụ thuộc vào vị trí, có thể truy xuất tư bất kỳ nơi nào, bất kỳ lúc nào có kết nối internet,
* Dùng chung tài nguyên (Resource pooling):
* Nhà cung cấp dịch vụ cho phép người dùng dùng chung tài nguyên do họ cung cấp dựa trên mô hình multi-tenant tài nguyên được phân phát rất linh hoạt tùy theo nhu cầu của người dùng.
* Khi như cầu của một người dùng nào đó giảm xuống, lập tức phần tài nguyên dư thừa sẽ được phục vụ cho người dùng khác.
* Các tài nguyên ảo này sẽ được cấp phát động theo sự thay đổi nhu cầu của từng khác hàng khác nhau. Nhờ đó mà nhà cung cấp dịch vụ có thể phục vụ nhiều khách hàng hơn so với cách cấp phát tài nguyên tĩnh truyền thông.
* Khả năng co gian (Rapid elasticity):
* Một đặc tính nổi bật của Cloud Computing là khả năng tự động mở rộng hoạt thu nhỏ hệ thống theo yêu cầu người dùng (hệ thống sẽ tự mở rộng hoặc thu hẹp bằng cách thêm hoặc giảm bớt tài nguyên). Một người dùng kỹ hợp đồng thuê một Server gom 4 CPU.
* Nếu lượng truy cập thấp chỉ cần 1 CPU là đủ, khi đó hệ thống quản lý của nhà cung cấp dịch vụ sẽ tự ngắt bớt 3 CPU, người dùng không phá trả phí cho 3 CPU nó trên và chúng được đưa sang phục vụ người dùng khác.
* Đến khi nhu cầu răng tức là lượng truy cấp tầng hệ thống ngay lập tức số tự động thêm CPU vào, nếu nhu cầu vượt quá 4 CPU thì người dùng trả phí theo hợp đồng đã kỵ với nhà cung cấp cloud computing service. Khả năng có giận nhanh và linh hoạt giúp cho nhà cung cấp dịch vụ cloud computing service tần dùng tại nguyên cụ thừa phục vụ được nhiều khách hàng người dùng giảm chi phí vì họ chỉ phải trả tiền cho những tài nguyên thực sự dùng.
* Điều tiết dịch vụ (Measured service):
* Hệ thống điện toán đám mây tự động kiểm soát và tối ưu hóa việ sử dụng tài nguyên. Lượng tài nguyên sử dụng có thể được theo dõi, kiểm soát và báo cáo một cách minh bạch cho cả hai phía nhà cung cấp dịch vụ và người sử dụng.

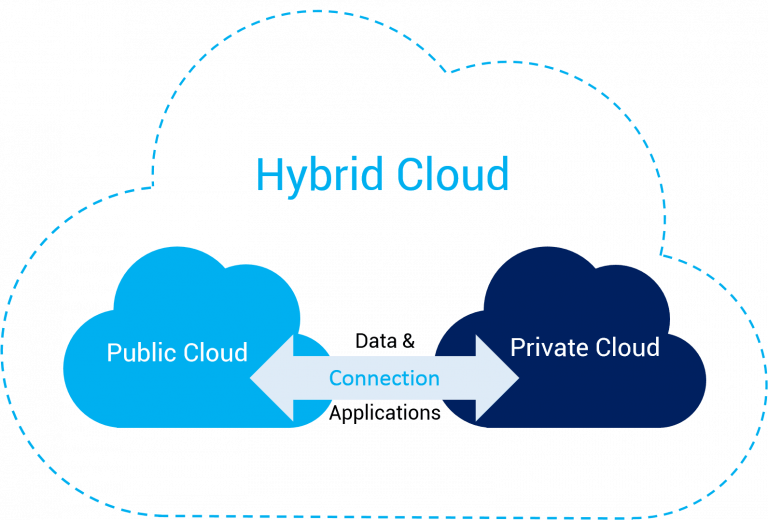
### 1.2.2. Bốn mô hình triển khai.

* Public Cloud: là các dịch vụ trên nền tảng Cloud Computing để cho các cá nhân và tổ chức thuê, họ dùng chung tài nguyên. Đây là mô hình triển khai được sử dụng phổ biến nhất hiện nay của Cloud Computing.



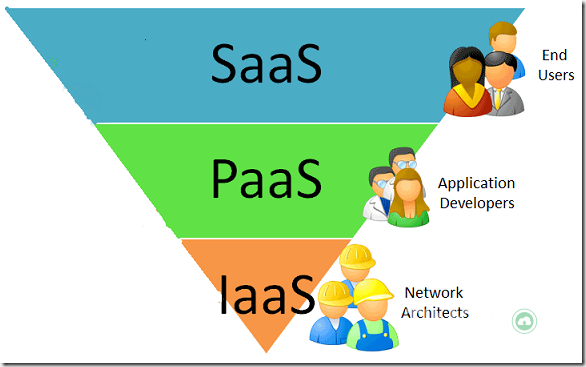
**Hình 1.2: Public Cloud**

* Private Cloud: dùng trong một doanh nghiệp và không chia sẻ với người dùng ngoài doanh nghiệp đó.
* Community Cloud: là các dịch vụ trên nền tảng Cloud Computing do các công ty cùng hợp tác xây dựng và cung cấp các dịch vụ cho cộng đồng.
* Hybrid Cloud: là thành phần của 2 hoặc nhiều các cloud khác (Private, Community hay Public Cloud) mà vẫn là các thực thể riêng biệt nhưng được liên kết với nhau, mang lại lợi ích của nhiều mô hình triển khai.



**Hình 1.3: Hybrid Cloud**

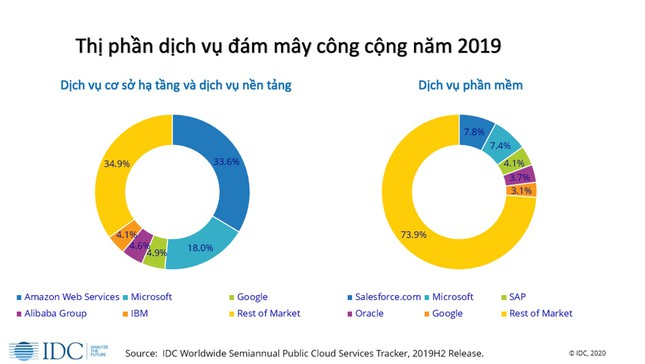
### 1.2.3. Ba mô hình dịch vụ:



**Hình 1.4: Mô hình dịch vụ của cloud computing**

* Infrastructure as a service (IaaS) – Cơ sở hạ tầng như một dịch vụ:
* IaaS cung cấp cho người dùng hạ tầng thô (thường dưới hình thức các máy ảo) như một dịch vụ.
* Người dùng có thể triển khai và chạy phần mềm trên các máy ảo như trên một máy chủ thực hay có thể đưa dữ liệu cá nhân lên “đám mây” và lưu trữ.
* Người dùng không có quyền kiểm soát hạ tầng thực bên trong “đám mây” tuy nhiên họ có toàn quyền quản lý và sử dụng tài nguyên mà họ được cung cấp, cũng như yêu cầu mở rộng lượng tài nguyên họ được phép sử dụng.
* Platform as a service (PaaS) – Nền tảng như một dịch vụ:
* PaaS cung cấp một môi trường cho các nhà phát triển ứng dụng mà họ có thể xây dựng và sử dụng để tạo ra các ứng dụng có thể điều chỉnh và tiếp tục phát triển.
* Người dùng không cần quan tâm đến hệ thống hạ tầng như hệ điều hành, mạng, lưu trữ,… đang vận hành bên dưới.
* Software as a Service (SaaS) – Phần mềm như một dịch vụ:
* Cung cấp cho người dùng việc sử dụng các ứng dụng của nhà cung cấp chạy trên cơ sở hạ tầng đám mây.
* Các ứng dụng có thể truy cập từ các thiết bị khách khác nhau thông qua trình duyệt web hoặc giao diện chương trình.
* Người dùng có quyền truy cập vào phần mềm ứng dụng và cơ sở dữ liệu. Các nhà cung cấp đám mây quản lý cơ sở hạ tầng và nền tảng chạy các ứng dụng

## 1.2.4. Các nhà cung cấp điện toán đám mây hàng đầu:

* **Amazon Web Service**
* [**Amazon Web Service (AWS)**](https://aws.amazon.com/vi/) là một nền tảng đám mây dùng để xây dựng các giải pháp doanh nghiệp bằng cách sử dụng những dịch vụ web tích hợp. AWS cung cấp một loạt các dịch vụ [**IaaS, PaaS**](https://hiepsharing.com/phan-biet-iaas-vs-paas-vs-saas/). Bao gồm Elastic Cloud Compute (EC2), Elastic Beanstalk, Simple Storage Service (S3) và Relational Database Service (RDS).
* AWS cung cấp quyền quản trị rộng rãi và luôn sẵn sàng thông qua một web client đảm bảo tính bảo mật. Người dùng có thể truy cập một số tính năng tại đây bao gồm việc tạo và kiểm tra một cặp khóa mã hóa (encryption key).
* AWS đồng thời cho phép bạn tối ưu hóa hạ tầng theo nhu cầu. Chi phí này sẽ thấp hơn so với việc xây dựng một hạ tầng tại chỗ (on premise).
* Một điểm cộng nữa của AWS là họ cung cấp 12 tháng miễn phí. Sau thời gian dùng thử, bạn sẽ nâng cấp lên gói trả phí hoặc hủy dịch vụ AWS của mình.
* **Microsoft Azure**
* [**Microsoft Azure**](https://azure.microsoft.com/en-us/?ranMID=24542&ranEAID=TnL5HPStwNw&ranSiteID=TnL5HPStwNw-6Q0.7znQvZOSgyvtM9BW0w&epi=TnL5HPStwNw-6Q0.7znQvZOSgyvtM9BW0w&irgwc=1&OCID=AID2000142_aff_7593_1243925&tduid=%28ir__xqe3jgfswwkftyizkk0sohzzx22xuu96vcyjogcx00%29%287593%29%281243925%29%28TnL5HPStwNw-6Q0.7znQvZOSgyvtM9BW0w%29%28%29&irclickid=_xqe3jgfswwkftyizkk0sohzzx22xuu96vcyjogcx00) đã hoạt động gần một thập kỷ trước, vào năm 2010. Người dùng có thể chạy bất kỳ dịch vụ nào trên cloud của họ hoặc kết hợp nó với trung tâm dữ liệu hoặc hạ tầng đang có (on premise).
* Microsoft Azure cung cấp hàng loạt các giải pháp phù hợp với mọi loại hình kinh doanh. Họ sẽ giúp bạn chọn một gói phù hợp với nhu cầu kinh doanh hiện tại.
* Với Azure, người dùng không cần máy chủ vật đặt tại cơ sở kinh doanh của họ. Điều này giúp tiết kiệm chi phí hạ tầng, chẳng hạn như một đội ngũ vận hành máy chủ tại chỗ.
* Azure Migration Center hỗ trợ quá trình di chuyển dữ liệu lên cloud dễ dàng và nhanh chóng hơn. Giải pháp này cũng hoàn toàn tương thích với Linux.
* Azure đồng thời mang đến hơn 25 dịch vụ miễn phí trọn đời và một cấp độ miễn phí giới hạn đối với nhiều dịch vụ phổ biến. Tất cả các gói và giá đều được hiển thị chi tiết trên trang chủ của họ. Trang này bao gồm bộ công cụ tính toán chi phí và dịch vụ dùng đến đâu trả đến đấy (**Pay as you go**).
* **Google Cloud**
* [**Google Cloud**](https://cloud.google.com/) cho phép người dùng tạo những giải pháp kinh doanh bằng các dịch vụ web mô-đun do Google cung cấp. Nó cung cấp một loạt các dịch vụ bao gồm những giải pháp IaaS, PaaS.
* Với hạ tầng bảo mật nhiều lớp của GCP, người dùng hoàn toàn yên tâm về mọi thứ họ tạo, mã nguồn hoặc lưu trữ sẽ được bảo vệ.
* Google Cloud bao gồm nhiều công cụ đảm bảo hiệu suất và quản lý nhất quán. Chúng bao gồm Compute Engine, App Engine, Container Engine, Cloud Storage và Big Query. Google đồng thời mang đến giải pháp di chuyển lên máy ảo của họ với một giá cả linh hoạt.

**Hình 1.5: Thị phần dịch vụ đám mây công cộng năm 2019**

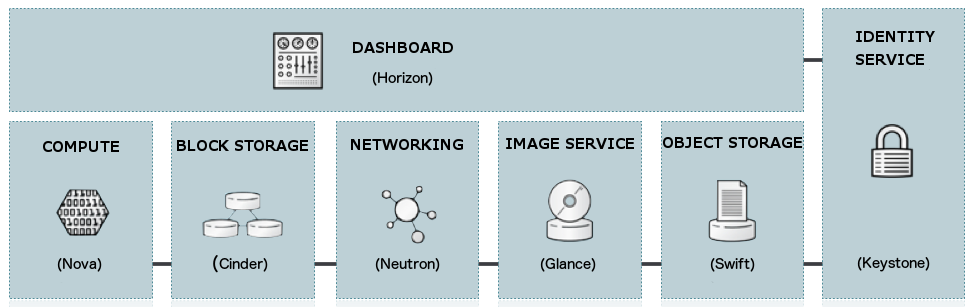
# CHƯƠNG II: GIỚI THIỆU VỀ OPENSTACK

## 2.1. Tổng quan

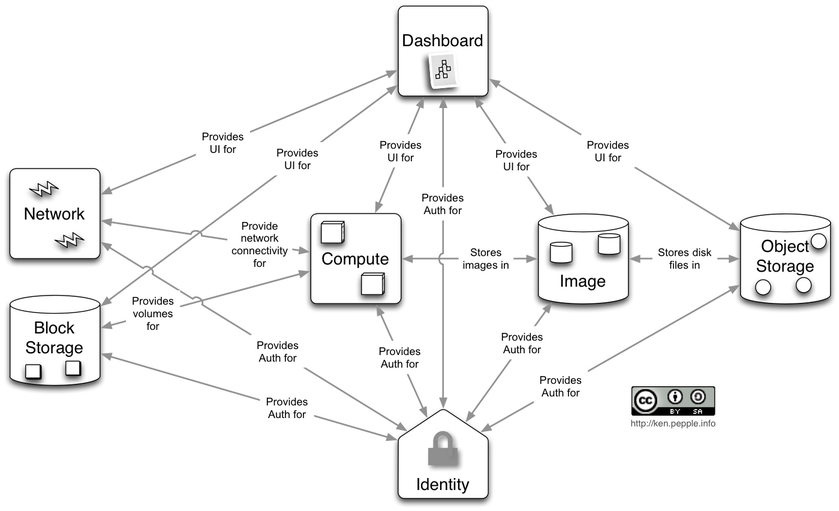
- OpenStack là một phần mềm mã nguồn mở, dùng để triển khai Cloud Computing

- Thiết kế theo hướng module. OpenStack là một project lớn là sự kết hợp của các project thành phần: nova, swift, neutron, glance, etc.  
- Mở về: Thiết kế/ Phát triển/ Cộng đồng/ Mã nguồn.  
- Chu kì 6 tháng một phiên bản mới.  
- 99.99% mã nguồn được viết bằng Python 2.x  
- Tên các phiên bản được đánh theo A, B, C (Austin, Bexar, Cactus, etc.)…. Tối đa 10 kí tự là danh từ.

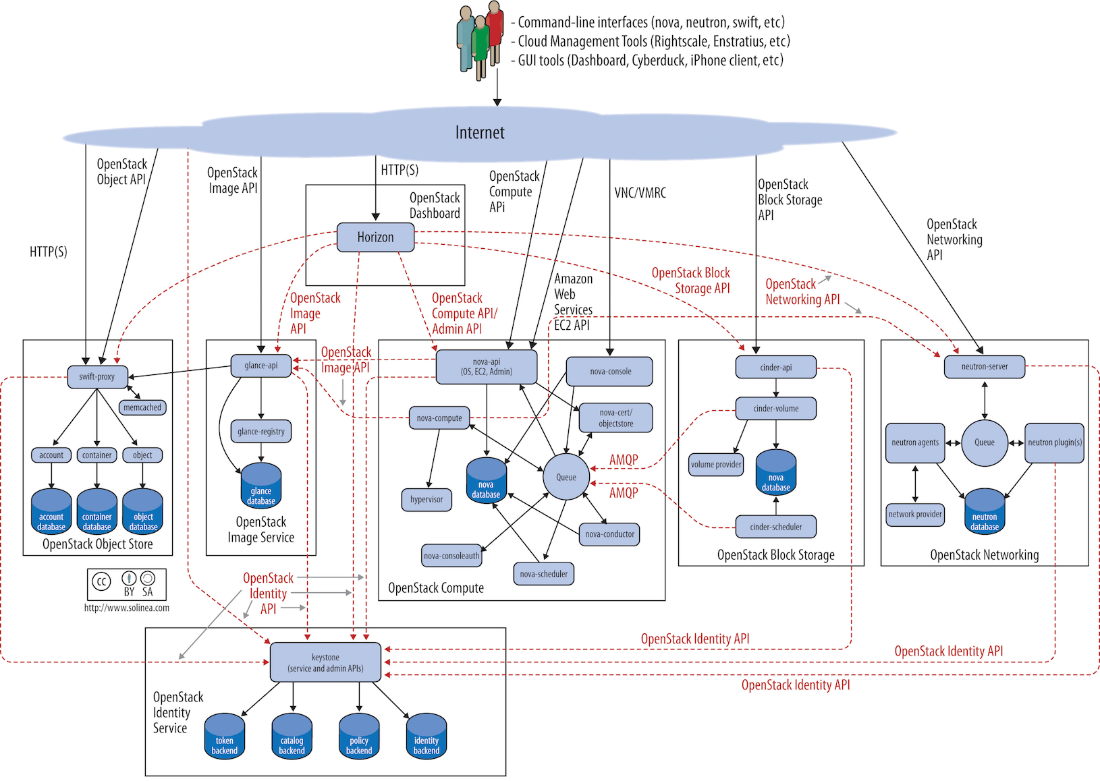
## 2.2. Kiến trúc các thành phần chính của OpenStack



**Hình 2.1: Kiến trúc Openstack**



**Hình 2.2: Kiến trúc logic Openstack (conceptual)**



**Hình 2.3: Cách hoạt động của các bộ phân trong OpenStack**

**2.3. Tổng quan về các project core của Openstack**

**2.3.1. Horizon ( Dashboard Service )**

**2.3.1.1. Khái niệm**

* Cung cấp giao diện web cho người dùng cuối và người quản trị cloud để tương tác, quản lí và theo dõi tài nguyên của đám mây
* Horizon viết dựa trên python django framework.
* Việc thiết kế có thể mở rộng giúp dễ dàng thêm vào các sản phẩm cũng như dịch vụ ngoài như billing, monitoring và các công cụ giám sát khác.

**2.3.2. Keystone ( Identity Service )**

**2.3.2.1. Khái niệm**

- Là dịch vụ xác thưc và cung cấp cho người dùng khả năng truy cập vào các tài nguyên và các dịch vụ trong đám mây

## 2.3.2.2. Chức năng chính Keystone:

### Identity:

* Nhận diện những người đang cố truy cập vào các tài nguyên cloud
* Trong keystone, identity thường được hiểu là User
* Tại những mô hình OpenStack nhỏ, identity của user thường được lưu trữ trong database của keystone. Đối với những mô hình lớn cho doanh nghiệp thì 1 external Identity Provider thường được sử dụng.
* Identity service trong keystone cung cấp các Actors. Nó có thể tới từ nhiều dịch vụ khác nhau như:

Federated Identity Provide LDAP

Multiple Backends SQL

### Authentication

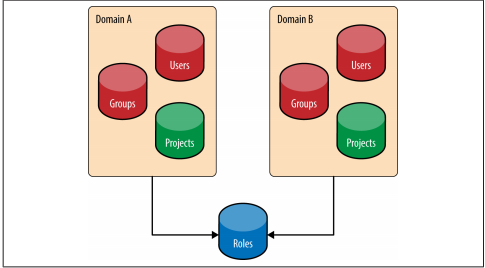
* Là quá trình xác thực những thông tin dùng để nhận định user (user's identity)
* Keystone có tính pluggable tức là nó có thể liên kết với những dịch vụ xác thực người dùng khác như LDAP hoặc Active Directory.
* Thường thì keystone sử dụng Password cho việc xác thực người dùng. Đối với những phần còn lại, keystone sử dụng tokens.
* OpenStack dựa rất nhiều vào tokens để xác thực và keystone chính là dịch vụ duy nhất có thể tạo ra tokens
* Token có giới hạn về thời gian được phép sử dụng. Khi token hết hạn thì user sẽ được cấp một token mới. Cơ chế này làm giảm nguy cơ user bị đánh cắp token.
* Hiện tại, keystone đang sử dụng cơ chế bearer token. Có nghĩa là bất cứ ai có token thì sẽ có khả năng truy cập vào tài nguyên của cloud. Vì vậy việc giữ bí mật token rất quan trọng.
* Có rất nhiều cách để xác thực với Keystone service, trong đó 2 phương thức được dùng nhiều nhất là: Password và Token

### Access Management (Authorization)

* Access Management hay còn được gọi là Authorization là quá trình xác định những tài nguyên mà user được phép truy cập tới.
* Trong OpenStack, keystone kết nối users với những Projects hoặc Domains bằng cách gán role cho user vào những project hoặc domain ấy.
* Các projects trong OpenStack như Nova, Cinder...sẽ kiểm tra mối quan hệ giữa role và các user's project và xác định giá trị của những thông tin này theo cơ chế các quy định (policy engine). Policy engine sẽ tự động kiểm tra các thông tin (thường là role) và xác định xem user được phép thực hiện những gì.
* Quản lí truy cập và quy định users được sử dụng APIs nào là một trong những yếu tố quyết định khiến Keystone trở nên quan trọng trong OpenStack. Về bản chất, Keystone sẽ tạo ra Role-Based
* Access Control (RBAC) policy trên mỗi một public APIs endpoints. Các policies này nằm trên file policy.json.



**Hình 2.4: KeyStone xác thực người dùng**



**Hình 2.5: Các thành phần liên quan đến xác thực**

#### 2.3.2*.*3 Ý nghĩa của các thành phần liên quan đến KeyStone

#### Projects

+ Trong Keystone, Project được dùng bởi các services của OpenStack để nhóm và cô lập các nguồn tài nguyên. Nó có thể được hiểu là 1 nhóm các tài nguyên mà chỉ có một số các user mới có thể truy cập và hoàn toàn tách biệt với các nhóm khác.

+ Ban đầu nó được gọi là tenants sau đó được đổi tên thành projects.

+ Mục đích cơ bản nhất của keystone chính là nơi để đăng kí cho các projects và xác định ai được phép truy cập projects.

+ Bản thân projects không sở hữu users hay groups mà users và groups được cấp quyền truy cập tới project sử dụng cơ chế gán role.

+ Trong một vài tài liệu của OpenStack thì việc gán role cho user còn được gọi là "grant".

#### Domains

+ Là một tập hợp các users, groups, và projects. Nó cho phép người dùng chia nguồn tài nguyên cho từng tổ chức sử dụng mà không phải lo xung đột hay nhầm lẫn.

#### Users and User Groups (Actors)

+ Users và User Groups là những đối tượng được cấp phép truy cập tới các nguồn tài nguyên được cô lập trong domains và projects.

#### Roles

+ Được dùng để hiện thực hóa việc cấp phép trong keystone. Một actor có thể có nhiều roles đối với từng project khác nhau.

#### Assignment

+ Role assignment là sự kết hợp của actor, target và role.

+ Role assignment được cấp phát, thu hồi, và có thể được kế thừa giữa các users, groups, project và domains.

#### Targets

+ Projects và Domains đều giống nhau ở chỗ cả hai đều là nơi mà role được "gán" lên.Vì thế chúng được gọi là targets.

#### Token

+ Để user truy cập bất cứ OpenStack API nào thì user cần chúng minh họ là ai và họ được phép gọi tới API. Để làm được điều này, họ cần có token và "dán" chúng vào "API call". Keystone chính là service chịu trách nhiệm tạo ra tokens.

+ Sau khi được xác thực thành công bởi keystone thì user sẽ nhận được token. Token cũng chứa các thông tin ủy quyền của user trên cloud.

+ Token có cả phần ID và payload. ID được dùng để đảm bảo rằng nó là duy nhất trên mỗi cloud và payload chứa thông tin của user.

#### Catalog

+ Chứa URLs và endpoints của các services khác nhau.

+ Nếu không có Catalog, users và các ứng dụng sẽ không thể biết được nơi cần chuyển yêu cầu để tạo máy ảo hoặc lưu dữ liệu.

+ Service này được chia nhỏ thành danh sách các endpoints và mỗi một endpoint sẽ chứa admin URL, internal URL, and public URL.

+ Service catalog là cần thiết cho OpenStack cloud. Nó chứa URLs and endpoint của các Cloud services khác nhau. Không có catalog, users và applications không biết nơi để route các requests để tạo VMs hoặc store objects. Service catalog chia thành a list of endpoint, mỗi endpoint chia nhỏ thành admin URL, internal URL, và public URL, có thể giống nhau.

#### 2.3.3 Nova ( Compute Service )

**2.3.3.1. Khái niệm**

- Là service chịu trách nhiệm chứa và quản lí các hệ thống cloud computing.

- OpenStack Compute chính là phần chính quan trọng nhất trong kiến trúc hệ thống Infrastructure-as-a-Service (IaaS).

- Nova giao tiếp với keyStone để xác thực, với Glane để lấy images và Horizon để lấy giao diện cho người dùng và người quản trị.

- 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 năn thông qua một web-based API.

- Nova cho phép bạn điều khiển các máy ảo và networks, bạn cũng có thể quản lý các truy cập tới cloud từ users và projects.

- Nó hỗ trợ nhiều hypervisors gồm KVM, QEMU, LXC, XenServer, vSphere, Hyper-V …

- Phần lớn các modules của Nova được viết bằng Python.

**2.3.3.2 Các dịch vụ của Nova:**

**- API server :**

* API server là trái tim của cloud framwork, nơi thực hiện các lệnh và điều khiển hypervisor, storage và networking sẵn có đến users.
* API endpoint cơ bản là HTTP web services, xử lý authentication, authorization, và các câu lệnh cơ bản và chức năng điều khiển sử dụng API interface khác nhau của Amazon, Rackspace, và các mô hình liên quan khác. Điều này cho phép sự tương thích API với nhiều bộ tool để tương tác với các vendor khác nhau.

**- Message queue :**

* Messaging queue được xem như môi giới tương tác giữa các compute nodes, networking controllers, API endpoints, scheduler (xác định physical hardware cần phân bố cho tài nguyên ảo) và thành phần tương tự.
* Giao tiếp đến và từ could controller được xử lý bởi HTTP request thông qua nhiều API endpoints.
* Một sự kiện bắt đầu với thông điệp tiêu biểu mà API server nhận được từ yêu cầu của user. API server chứng thực user và đảm bảo user được phép đưa ra lệnh đó. Nếu các object liên quan đến yêu cầu là sẵn có thì yêu cầu sẽ được chuyển đến queuing engine cho những worker liên quan. Workers liên tục lắng nghe quêu dựa trên role của họ. Khi đến lượt yêu cầu được thực hiện, worker phân conng nhiệm vụ và thực hiện yêu cầu đó. Sau khi hoàn thành, phản hồi được gửi đến queue được nhận bởi API server và chuyển tiếp tới user. Các thao tác truy vấn, nhập và xóa là cần thiết trong quá trình.
* **Compute worker :**
* Compute worker quản lý computing instance trên host machine. API truyền lệnh đến compute worker để hoàn thành các tác vụ:
* Run instances - Delete instances
* Reboot instances - Attach volumes
* Detach volumes - Get console output
* **Network Controller**
* Network Controller quản lý tài quyền networking trên host machine. API server gửi các lệnh thông qua message queue, sau đó được xử lý bởi Network Controllers.
* Cấp phát fixed IP address
* Cấu hình VLANs cho projects
* Cấu hình network cho compute node

## 2.3.3.2 Các thành phần của Nova:

* **APIs:**
* **nova-api** : Là service tiếp nhận và phản hồi các compute API calls từ user. Service hỗ trợ OpenStack Compute API, Amazon EC2 API và Admin API đặc biệt được dùng để user thực hiện các thao tác quản trị. Nó cũng có một loạt các policies và thực hiện hầu hết các orchestration activities ví dụ như chạy máy ảo.
* **nova-api-metadata** : Là service tiếp nhận các metadata request từ máy ảo. Service này thường được dùng khi chạy multi-host kết hợp với nova-network.
* **nova-compute** : Là service chịu trách nhiệm tạo và hủy các máy ảo qua hypervisors APIs.
* **nova-placement-api** : được dùng để theo dõi thống kê và mức độ sử dụng của mỗi một resource provider.
* **nova-scheduler** : Service này sẽ lấy các yêu cầu máy ảo đặt vào queue và xác định xem chúng được chạy trên compute server host nào.
* **nova-conductor** : Là module chịu trách nhiệm về các tương tác giữa nova-compute và database. Nó sẽ loại bỏ tất cả các kết nối trực tiếp từ nova-compute tới database.
* **nova-consoleauth** : Xác thực token cho user mà console proxies cung cấp. Dịch vụ này buộc phải chạy cùng với console proxies. Bạn có thể chạy proxies trên 1 nova-consoleauth service hoặc ở trong 1 cluster configuration.
* **nova-novncproxy** : Cung cấp proxy cho việc truy cập các máy ảo đang chạy thông qua VNC connection. Nó hỗ trợ các trình duyệt based novnc clients.
* **nova-spicehtml5proxy** : Cung cấp proxy để truy cập các máy ảo đang chạy thông qua SPICE connection. Nó hỗ trợ các trình duyệt based HTML5 client.
* **nova-xvpvncproxy** : Cung cấp proxy cho việc truy cập các máy ảo đang chạy thông qua VNC connection. Nó hỗ trợ OpenStack-specific Java client.
* **The queue**: Trung tâm giao tiếp giữa các daemons. Thường dùng RabbitMQ hoặc các AMQP message queue khác như ZeroMQ.
* **SQL database** : Dùng để lưu các trạng thái của hạ tâng caloud bảo gồm:
* Các loại máy ảo có thể chạy
* Các máy ảo đang được dùng
* Các network khả dụng
* Projects

**2.3.4 Neutron ( Netwok Service )**

**2.3.4.1 Tổng quan**

* OpenStack Networking xử lý tạo và quản lý cơ sở hạ tầng mạng ảo, bao gồm networks, switches, subnets, và router cho devices được quản lý bởi OpenStack Compute service (Nova). Các services nâng cao như firewalls hoặc virtual private networks (VPNs) có thể được sử dụng.
* Sử dụng kiến trúc “plug-in”, Các plug-in được thực thi trên nhiều kiến trúc khác nhau (NVP, Open vSwitch, Linux bridge, Cisco)
* OpenStack Networking gồm neutron-server, database cho lưu trữ persistent, và bất kỳ số plug-in agent, cung cấp 1 số services khác như giao tiếp với native Linux networking mechanisms, external devices và SDN controller.  
  - OpenStack Networking là hoàn toàn độc lập và có thể triển khai trên 1 server độc lập.
* OpenStack Networking cho phép bạn tạo và quản lí các network objects ví dụ như networks, subnets, và 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.
* Dịch vụ Networking trong OpenStack (neutron) cũng 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.

**2.3.4.2. Neutron tích hợp với 1 số thành phần OpenStac khác**

* OpenStack Identity service (keystone) được sử dụng cho việc xác thực và ủy quyề của API request.
* OpenStack Compute service (nova) được sử dụng cho việc cắm mõi virtual NIC trên VM đến 1 mạng cụ thể.
* OpenStack Dashboard (horizon) được sử dụng bởi admin và user để tạo và quản lý network service thông qua giao diện web.

**2.3.4.3. Các thành phần của Neutron**

* **API server**
* OpenStack Networking API hỗ trợ Layer2 networking và IP address management (IPAM - quản lý địa chỉ IP), cũng như một extension để xây dựng router Layer 3 cho phép định tuyến giữa các networks Layer 2 và các gateway để ra mạng bên ngoài. OpenStack Networking cung cấp một danh sách các plug-ins (đang ngày càng tăng lên) cho phép tương tác với nhiều công nghệ mạng mã nguồn mở và cả thương mại, bao gồm các routers, switches, switch ảo và SDN controller.
* **OpenStack Networking plug-in and 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 RPC requests 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.
* Để cấp hình topology mạng phong phú, bạn có thể tạo và cấu hình networks và subnets và hướng dẫn các dịch vụ OpenStack khác như Compute để gắn các **virtual device** đến ports trên networks này. **OpenStack Compute** là đối tượng sử dụng **OpenStack Networking** để cung cấp kết nối cho instqance của nó. Đặc biệt, OpenStack Networking hỗ trợ mỗi project có nhiều private network và cho phép projects chọn địa chỉ IP của riêng họ, ngay cả khi những địa chỉ IP đó chống chéo với những projects khác sử dụng. Có 2 loại network, **project** và **provider** network.

**2.3.4.4. Các loại mạng neutron cung cấp**

* Provider networks
* Provider network cung cấp khả năng kết nối layer-2 đến instance với sự hỗ trợ tùy chọn cho DHCP và metadata service. Mạng này kết nối, hoặc map, với các mạng layer-2 hiện có trong data center, thường sử dụng tính năng VLAN (802.1q) tagging để xác định và tách chúng.
* Provider network cung cấp nói chung cung cấp sự đơn giản, hiệu năng và độ tin cậy với chi phí linh hoạt. Bởi mặc định, chỉ có admin có thể tạo hoặc cập nhật các **provider network** bởi vì cấu hình yêu cầu của cở sở hạ tầng physical network.
* Việc tạo và sửa đổi **provoder network** cho phép sử dụng các tài nguyên mạng vật lý như VLANs. Cho phép những thay đổi này chỉ dành cho những tenants đáng tin cậy :
* Ngoại ra, **provider network** chỉ xử lý kết nối layer-2 cho các instances, do đó thiếu sự hỗ trợ như routers và địa chỉ floating IP.  
  - Nói chung, các thành phàn phần mềm OpenStack Networking xử lý các hoạt động layer-3 ảnh hướng đến hiệu năng và độ tin cậy. Để cải hiện hiệu năng và độ tin cậy, provider network di chuyển hoạt động layer-3 đến cơ sở hạ tầng mạng vật lý.+
* Trong 1 trường hợp cụ thể, triển khai OpenStack tồn tại trong 1 môi trường hỗ hợp với virtualization và bare-metal hosts sử dụng 1 cơ sở hạ tầng mạng vật lý cỡ lớn. Ứng dụng chạy trong triển khai OpenStack có thể yêu cầu truy cập trực tiếp layer-2, thông thường sử dụng VLAN, cho các ứng dụng bên ngoài triển khai.

### Routed provider network

* Định tuyến **provider network** cung cấp kết nối layer-3 đến instances. Network này map mạng layer-3 đã tồn tại trong data center. Chính xác hơn, network maps đến nhiều segments layer-2, mỗi trong số đó về cơ bản là provider network.
* Self-service networks:
* **Self-service networks** chủ yếu cho phép các projects chung (không có đặc quyền) để quản lý mạng mà không cần đến admin. Các mạng này hoàn toàn là virtual và yêu câu virtual routers tương tác với provider và các magnj bên ngaofi Internet. **Self-service networks** cũng thường cung cấp các DHCP và metadata services đến instances.+
* Trong hầu hết các trường hợp, self-service networks sử dụng overlay protocol như VXLAN và GRE bởi vì chugn có thể hỗ trợ nhiều mạng hơn layer-2 segementation bwangf cách sử dụng VLAN tagging (802.1q). Hơn nữa, các VLAN thường yêu cầu cấu hình bổ sung của cơ sở hạ tầng mạng vật lý.
* IPv4 self-service network sử dụng dải địa chỉ private IP và tương tác với provider network thông qua source NAT trên virtual routers. Địa chỉ Floating IP cho phép truy cập đến instance từ provider networks thông destionation NAT trên virtual router.
* IPv6 self-service network luôn sử dụng dải địa chỉ IP public và tương tác với các provider network thông qua virtual router với các static route.+
* Networking serive thực hện routers bằng cách sử dụng layer-3 agent, thường là ít nhất 1 network node. Trái ngược với provider network cung cấp kết nối instances đến mạng vật lý tại layer-2, self-service networks phải đi qua later-3 agent. Users tạo ra các project network cho kết nối trong projects. Bởi mặc định, chúng được cô lập hoàn toàn và không được chia sẻ các project khác. OpenStack Networking hỗ trợ csac loại công nghệ mậng cô lập và overlay.
* Flat :
* Tất cả instances tren cùng 1 mạng, cũng có thể chia sẻ với các hosts. Không có VLAN tagging hoặc phần chia mạng.
* VLAN:
* Mạng cho phéo user tạo nhiều provider hoặc project sử dụng VLAN IDs (802.1Q) tương ứng với các VLAN có trong mạng vật lý.
* GRE và VXLAN:
* VXLAN và GRE là giao thức đóng gói tọa ra overkay network để kích hoạt và kiểm soát giữa compute instance. 1 networking router được yêu cầu để cho phép luông lưu lượng đu ra ngoài mạng GRE hoặc VXLAN. 1 router cũng được yêu cầu để kết nối trực tiếp project network với external network. Router cung cấp khả năng kết nối trực tiếp instance từ external network sử dụng địa chỉ floating IP.

### Subnets:

* Một khối các địa chỉ IP và trạng thái cấu hình liên quan. Đây cũng được gọi là native IPAM (IP Address Management) cung cấp bởi dịch vụ mạng cho project và provider network. Subnets được sử dụng để phần bổ địa chỉ IP khi các ports mới được tạo trên mạng.

### Subnet pools:

* End users có thể tạo subnets với bất kỳ đại chỉ IP hợp lệ. Tuy nhiên, 1 trong số các trường hợp, nó là tốt đẹp cho các admin hoặc project được định nghĩa trước pool của địa chỉ từ đó để tạo subnets với tự động cấp phát địa chỉ IP.
* Sử dụng **subnet pools** sẽ hạn chế những địa chỉ nào có thể sử dụng bằng cách yêu cầu mọi subnet phải nằm trong pool được xác định. Nó cũng ngăn ngừa việc tài sử dụng địa chỉ hoặc 2 subnets từ cùng pool.
* Ports:
* Port là điểm để kết nối để gắn 1 thiết bị, chẳng hạn NIC của virtual server, đến virtual network. Port cũng mô tả cấu hình mạng liên quan, chả hạn như MAC và địa chỉ IP được sử dụng trên port đó.

### Routers:

* Router cung cấp virtual layer-3 services như routing và NAT giữa **self-service network** và **provider network** hoặc giữa các **self-service networks** phụ thuộc vào project. Networking service sử dụng **layer-3 agent** để quản lý routers thông qua namespaces.

### Security group:

* Cung cấp vùng lưu trữ cho virtual firewall rules để kiểm soát lưu lượng truy cập ingress (inbound to instance) và egress (outbound from instance) mạng ở mức port. **Security groups** sử dụng default deny policy và chỉ chứa các rules đồng ý phép lưu lượng cụ thể. FIrewall driver chuyển các group rule đến cấu hình cho ciing chệ lọc gói bên dưới như iptables.
* Mỗi project có chứa 1 default security group mà cho phép tát cả lưu lượng egress và từ chối tất cả các lưu lượng ingress. Bạn có thể thay đổi rules trong default security group. Nó bạn launch instance mà không có security group chỉ định, default security group sẽ tự động được áp dụng cho instance đó. Tương tự, nếu bạn tạo 1 port mà không chỉ định security group, default security group tự động được áp cho port đó.
* Security group rules:
* Là stateful. Do đó, cho phép ingress TCP port 22 cho SSH tự động tạo lưu lượng engress và thông điệp ICMP error liên quan đến kết nối TCP.
* Mặc định, tất cả các security groups chứa 1 loạt các rules cơ bản và chống lừa đảo

### 2.3.4.5. Các dịch vụ phổ biến:

#### VPNaaS:

* Virtual Private Network-as-a-Service (VPNaaS) là phần mở rộng của neutron giới thiệu tính năng VPN.

#### LBaaS:

* Load-Balancer-as-a-Service (LBaaS) API cung cấp và cấu hình load balancers. Việc thực hiện tham chiếu dựa trên HAProxy software load balancer.

#### FWaaS:

#### Firewall-as-a-Service (FWaaS) API là API thực nghiệm cho phép adopters và nhà cung cấp thử nghiệm triển khia mạng của họ.

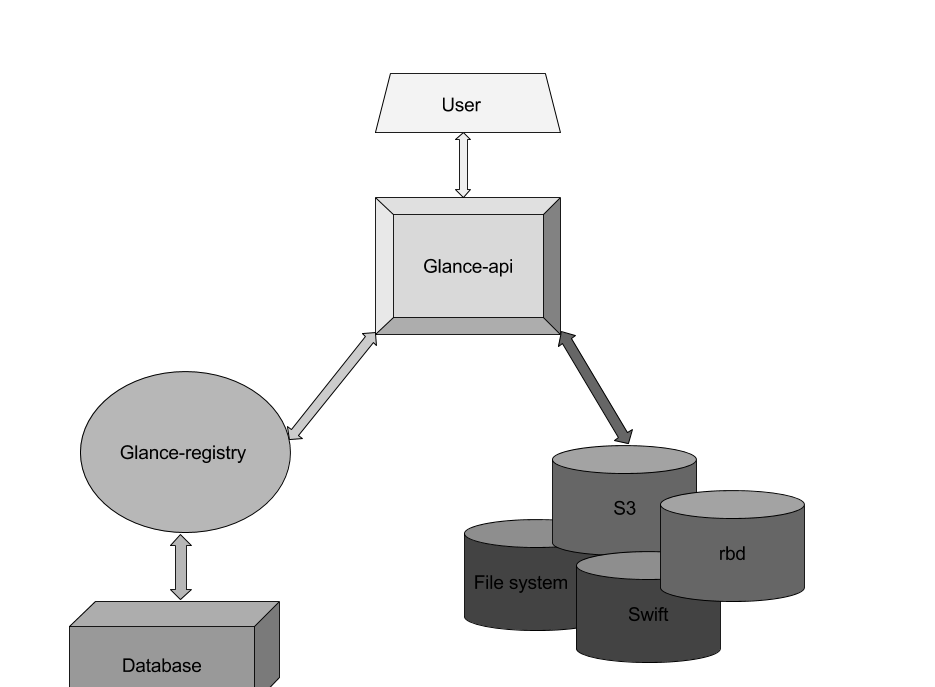
* Plug-in Firewall-as-a-Service (FWaaS) áp dụng firewall cho các đối tượng OpenStack như projects, routers, và router ports.

**2.3.5 Glance ( Image Service )**

**2.3.5.1 Tổng quan**

* Glane (Image Service) là image service cung cấp khả năng discovering, registering (đăng ký), retrieving (thu thập) các image cho virtual machine. OpenStack Glance là central repository cho virtual image.
* Glance cung cấp RESTful API cho phép querying VM image metadata cũng như thu thập các actual image.
* VM image có sẵn thông qua Glance có thể stored trong nhiều vị trí khác nhau, từ file system đến object storage system như OpenStack Swift OpenStack.
* Trong Glance, images được lưu trữ như template, được sử dụng để launching new instances. Glance được thiết kế để trở thành một service độc lập đối với các user cần tổ chức large virtual disk images. Glance cung cấp giải pháp end-to-end cho cloud disk image management. Nó cũng có thể lấy các bản snapshots từ running instance cho việc backing up VM và các states của VM.
* Glance chấp nhận API request cho image từ end-users hoặc Nova components và có thể stores nó trong object storage service,swift hoặc storage repository khác.
* Image server hỗ trợ các back-end stores:
* File system: OpenStack Image server lưu trữ virtual machine images trong file system back end là default.Đây là back end đơn giản lưu trữ image files trong local file system.
* Object Storage: Là hệ thống lưu trữ do OpenStack Swift cung cấp - dịch vụ lưu trữ có tính sẵn sàng cao , lưu trữ các image dưới dạng các object.
* Block Storage: Hệ thống lưu trữ có tính sẵn sàng cao do OpenStack Cinder cung cấp, lưu trữ các image dưới dạng block.
* HTTP: OpenStack Image service có thể đọc virtual machine images mà có sẵn trên Internet sử dụng HTTP. Đây là store chỉ đọc.
* RADOS Block Device (RBD): Stores images trong Ceph storage cluster sử dụng Ceph’s RBD interface.
* Sheepdog: A distributed storage system dành cho QEMU/KVM.
* GridFS: Stores images sử dụng MongoDB.

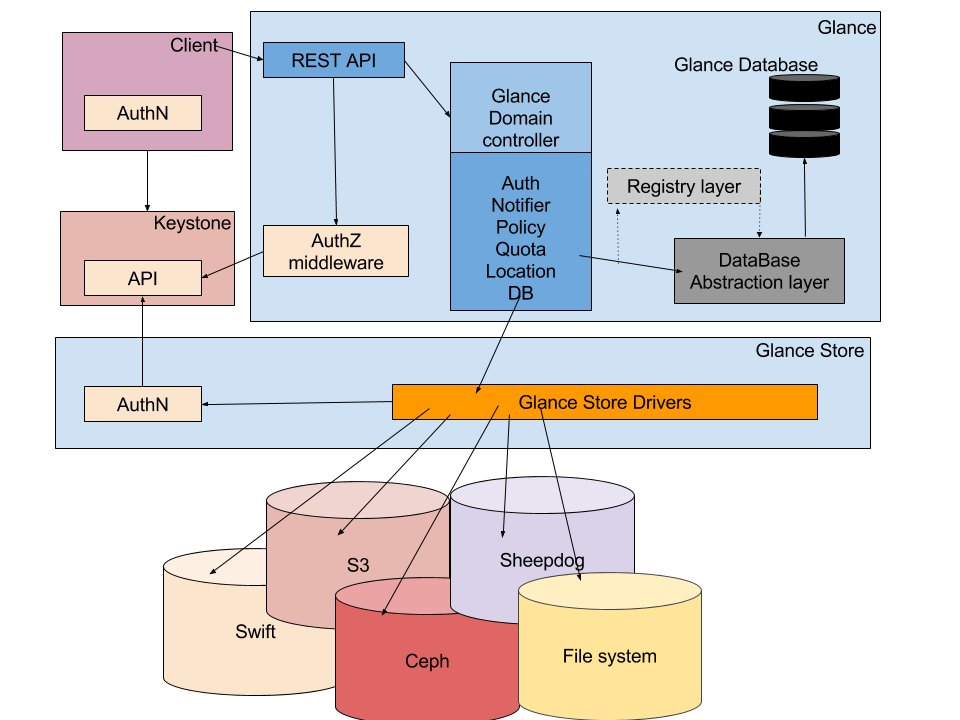
**2.3.5.2. Thành phần của glance**



**Hình 2.6: Thành phần của Glance**

* glance-api: chấp nhận API calls cho việc tìm kiếm, lấy và lưu trữ image.
* glance-registry: thực hiện lưu trữ, xử lý và lấy thông tin metadata của image.
* database: lưu trữ metadata của image.
* storage repository: tích hợp với nhiều thanh phần của OpenStack như file systems, Amazon S3 và HTTP cho image storages.

**2.3.5.3. Kiến trúc của Glance**

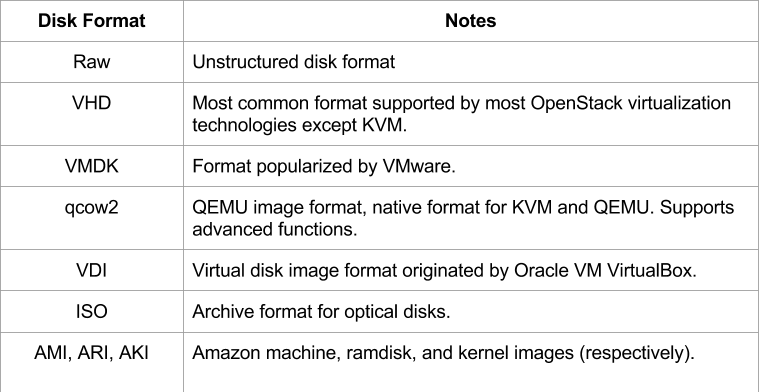


**Hình 2.7: Kiến trúc của Glance**

* Khi chúng ta upload image cho glance, chúng ta cần chỉ định format của Virtual machine images. Glance hỗ trợ nhiều format khác nhau như Disk Formats và Container Formats.
* Virtual disk là tương tự physical server’s boot driver.
* Các loại virtualization khác nhau hỗ trợ disk formats khác nhau.

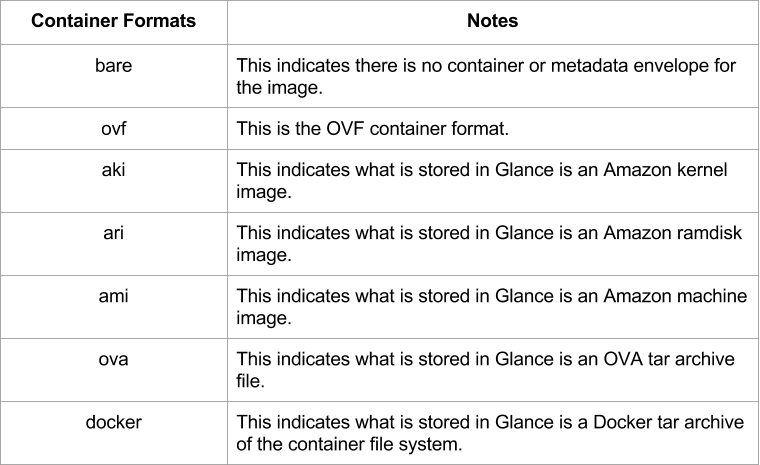
**2.3.5.4 Hộ trợ các loại format disk của Glance**

* Disk Formats:
* Disk Formats của virtual machine image là format của underlying disk image. Disk formats được hỗ trợ bởi OpenStack glance.



**Hình 2.8: Các loại disk format**

* Container Formats
* OpenStack glance hỗ trợ container format, mô tả file formats và chứa các thông tin metadata về actual virtual machine.  
  Container formats được hỗ trợ trong OpenStack glance :



**Hình 2.9: Các loại container format**

**2.3.6. Cinder ( Block Storage Service )**

**2.3.6.1. Khái niệm**

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

**2.3.6.2. Các thành phần của Cinder**

* Back-end Storage Device : Dịch vụ Block Storage yêu cầu một vài kiểu của back-end storage mà dịch vụ có thể chạy trên đó. Mặc định là sử dụng LVM trên một local volume group tên là "cinder-volumes"
* User và Project :
* Cinder được dùng bởi các người dùng hoặc khách hàng khác nhau (project trong một shared system), sử dụng chỉ định truy cập dưa vào role (role-based access). Các role kiểm soát các hành động mà người dùng được phép thực hiện.
* Trong cấu hình mặc định, phần lớn các hành động không yêu cầu một role cụ thể, nhưng sysad có teher cấu hình trong file policy.json để quản lý các rule. Một truy cập của người dùng có thể bị hạn chế bởi project, nhưng username và pass được gán chỉ định cho mỗi user.
* Key pairs cho phép truy cập tới một volume được mở cho mỗi user, nhưng hạn ngạch để kiểm soát sự tiêu thu tài nguyên trên các tài nguyên phần cứng có sẵn là cho mỗi project.
* Volume :
* Các tài nguyên block storage được phân phối có thể gán vào máy ảo như một ổ lưu trữ thứ 2 hoặc có thể dùng như là vùng lưu trữ cho root để boot máy ảo.
* Volume là các thiết bị block storage R/W bền vững thường được dùng để gán vào compute node thông qua iSCSI.
* Snapshot :
* Một bản copy trong một thời điểm nhất định của một volume. Snapshot có thể được tạo từ một volume mà mới được dùng gaafnd ây trong trạng thái sẵn sàng.
* Snapshot có thể được dùng để tạo một volume mới thông qua việc tạo từ snapshot.
* Backup : Một bản copy lưu trữ của một volume thông thường được lưu ở Swift.

## 2.3.6.3. Kiến trúc của Cinder

**Hình 2.10. Kiến trúc của Cinder**

## Cinder-api : Xác thực và định tuyến các yêu cầu xuyên suốt dịch vụ Cinder.

* Cinder-scheduler : Lên lịch và định tuyến các yêu cầu tới dịch vụ volume thích hợp. Tùy thuộc vào cách cấu hình, có thể chỉ là dùng round-robin để định ra việc sẽ dùng volume service nào, hoặc có thể phức tạp hơn bằng cách dùng Filter Scheduler. Filter Scheduler là mặc định và bật các bộ lọc như Capacity, Avaibility Zone, Volume Type, và Capability.
* Cinder-volume : Quản lý thiết bị block storage, đặc biệt là các thiết bị back-end
* Cinder-backup : Cung cấp phương thức để backup một Block Storage volume tới Openstack Object Storage (Swift)
* Driver : Chứa các mã back-end cụ thể để có thể liên lạc với các loại lưu trữ khác nhau.
* Storage : Các thiết bị lưu trữ từ các nhà cung cấp khác nhau.
* SQL DB : Cung cấp một phương tiện dùng để back up dữ liệu từ Swift/Celp, etc,..

**2.3.7 Swift (Object Storage Service )**

**2.3.7.1. Khái niệm**

* Dùng để quản lý lưu trữ dạng ĐỐI TƯỢNG .
* Nó là một hệ thống lưu trữ phân tán cho quản lý tất cả các dạng của lưu trữ như: archives, user data, virtual machine image …
* Cung cấp dịch vụ lưu trữ giống S3 Amazon và tương thích với S3 API
* Có khả năng mở rộng, lưu trữ phân tán, có backup

**2.3.7.2. Kiến trúc logic**

* Tổ chức logic:
* Mỗi tenant được gán một **Account**.
* Trong account, có thể chứa nhiều **container** (tương tự như folder).
* Container sẽ chứa các **Object**.
* Người dùng lưu trữ các đối tượng mà không phải quan tâm đến phần cứng.
* Vấn đề đặt tên: các account phải có tên khác nhau và là duy nhất. Trong một account, tên của các containers cũng phải khác nhau, nhưng trên 2 account khác nhau thì tên container có thể trùng nhau. Với object cũng tương tự như thế.
* Trong lưu trữ swift, các containers là ngang hàng nhau. Như vậy sẽ không có tạo subcontainer. Tuy nhiên, swift hỗ trợ cơ chế giả thư mục để hỗ trợ cho việc quản lý.
* Chi tiết về từng thành phần:
* Account
* Account là gốc của vị trí lưu trữ data.
* Thường so sánh với filesystem volume, account là tên duy nhất trong hệ thống.
* Mỗi account có một database lưu trữ metada của bản thân nó và cả danh sách các containers ở trong account đó.
* Metada headers của account được lưu với cặp key-value đảm bảo lưu trữ lâu dài cùng với tất cả dữ liệu trong swift.
* Cơ sở dữ liệu account không chứa các objects và metada của object
* Cơ sở dữ liệu của account được truy cập khi người dùng (người được cho phép hoặc services) yêu cầu metada về một account hoặc danh sách containers trong account.
* Account có thể được chỉ định cho đơn người dùng truy cập hoặc nhiều người dùng truy cập tùy thuộc vào quyền mà nó được thiết lập.
* Đơn người dùng với người dùng các nhân hoặc các dịch vụ accounts như là backups, đa người dùng truy cập với project nơi nhiều thành viên của nhóm truy cập account.
* Người dùng swift với các quyền chính xác có thể tạo, thay đổi, xóa các container và các objects trong container với một account.
* Trong swift, account là lưu trữ account, không phải là định danh user.
* Container
* Container được người dùng định nghĩa phân đoạn của không gian tên account mà cung cấp vị trí lưu trữ nơi các objects được tìm thấy.
* Mặc dù các container không thể lồng vào nhau, nhưng có khái niệm tương tự như thư mục trong filesystem.
* Container không có ràng buộc về tên duy nhất.
* Không có giới hạn về số lượng container trong một account.
* Cũng như cơ sở dữ liệu account, mỗi container database chứa metada của container và danh sách các object trong container đó và metada headers cũng được lưu dưới dạng key-value.
* Database container không chứa objects thực và metada của object.
* Database container được truy cập khi user yêu cầu danh sách tất cả các object trong container hoặc metada về container đó.
* Trong một account, bạn có thể tạo và sử dụng các containers để nhóm dữ liệu theo cách logic. Bạn cũng có thể gán metada cho containers, và sử dụng vài tính năng Swift để cho các thuộc tính containers khác nhau.

## Objects.

* Object là dữ liệu được lưu trong Openstack Swift. Chúng có thể là ảnh, videos, tài liệu, logfiles, database backups, fileysystem snapshots, hoặc bất kỳ dữ liệu khác. Mỗi đối tượng thường bao gồm bản thân dữ liệu của nó và metada.
* Metada headers được lưu với cặp key-value, đảm bảo lâu dài với dữ liệu object, và đòi hỏi không thêm chút độ trễ để khôi phục. Metada có thể cung cấp các thông tin quan trọng về object. Ví dụ, một nhà sản xuất video có thể lưu định dạng video và khoảng thời gian dọc theo nội dụng; một tài liệu có thể chứa thông tin tác giả.
* Mỗi object phải thuộc một container. Khi một object được lưu trong cụm, người dugnf sẽ luôn để ý đến vị trí lưu trữ object (/account/container/object). Không có giới hạn số lượng object mà user có thể lưu trong container.
* Vị trí lưu trữ object là nơi để tìm object. Swift sẽ lưu nhiều bản sao của object để đảm bảo sự tin cậy và độ khả dụng của dữ liệu. Swift làm điều này bằng cách đặt đối tượng trong nhóm logic gọi là partition. Một partition sẽ liên kết đến nhiều thiết bị, mỗi thiết bị sẽ có object được ghi vào. User truy cập object thông qua vị trí của nó. User không biết và không cần biết nơi nào trong hệ thống dữ liệu được đặt.

**2.3.7.3. Kiến trúc vật lý**

* Region: Tại mức cao nhất, Swift lưu trữ dữ liệu trong các regions (gọi là khu vực) bị chia cắt về mặt vật lý - phân chia theo địa lý.
* Zone: Bên trong các regions là các zones. Zone là tập các máy chủ riêng biệt, các máy chủ này được gọi là node storage. Khi có máy chủ xảy ra lỗi, Swift sẽ tách biệt các zone có máy chủ bị lỗi đó với các zone khác.-
* Storage server (máy chủ lưu trữ): Trong các zones sẽ chứa các máy chủ để lưu trữ.
* Disk (ổ đĩa hoặc là thiết bị lưu trữ): Là một phần của storage server, có thể là các đĩa cứng cắm trực tiếp bên trong storage server hoặc được liên kết qua mạng.

**2.3.7.4. Tổng quan về kiến trúc phần mềm trong Swift.**

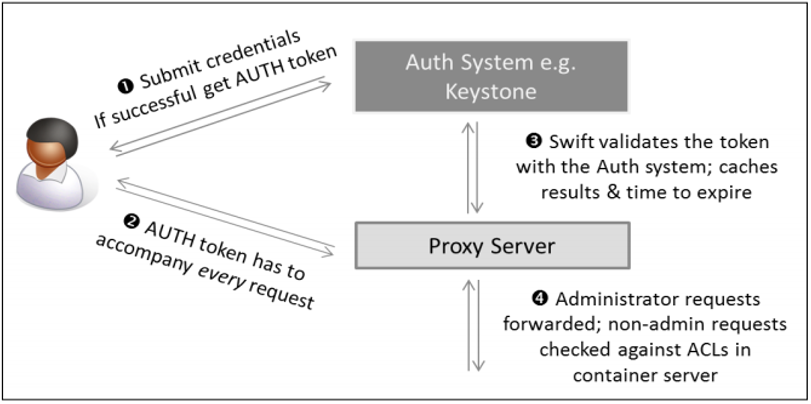
* Trong Swift có 4 service chính. Document của swift gọi là Servers:
* Account server
* Proxy server
* Account server
* Container server
* Object server-
* Thông thường, proxy server được cài đặt trên một node riêng, còn 3 server còn lại sẽ được cài đặt trên cùng một node gọi là storage server.
* Chức năng chính của từng server:
* Proxy server:
* Chịu trách nhiệm nhận các yêu cầu HTTP từ người dùng. Nó sẽ tìm kiếm vị trí của storage server nơi mà yêu cầu cần được chuyển tiếp bằng cách sử dụng ring phù hợp.
* Proxy server tính toán các node bị hư hại bằng cách tìm kiếm các handoff node và thực hiện đọc/ghi.
* Account server:
* Theo dõi tên các containers có trong account. Dữ liệu được lưu trữ trong cơ sở dữ liệu SQL, các files cơ sở dữ liệu sẽ được lưu trong filesystem. Server này cũng theo dõi thống kê nhưng không có bất kỳ thông tin nào về vị trí của các containers.
* Proxy server xác định vị trí container dựa trên container ring. Thông thường, account server sẽ được cài đặt với container server và object server trên cùng một máy chủ vật lý. Tuy nhiên, với một hệ thống lớn thì có thể cài đặt các server trên các máy chủ riêng biệt.
* Container server:
* Giống như account server, ngoại trừ container server quản lý các objects bên trong nó.
* Object server:
* Đơn giản là server quản lý lưu trữ đối tượng. Trên các ổ đĩa có các filesystem, và các đối tượng được lưu trên đó.

**2.3.7.4. Kiến trúc xác thực trong swift**

# Xác thực là một phần quan trọng trong swif

# Các thành phần trong swift đều dạng module, do vậy thành phần xác thực cũng là một project riêng.

# Người dùng có thể lựa chọn các hệ thống xác thực khác nhau. Keystone là project xác thực ở trong Openstack mà có thể được sử dụng cho Swift.



**Hình 2.11. Xác thực trong swift**

* Luồng làm việc của Swift với keystone:
  + Người dùng cung cấp thông tin xác thực cho hệ thống xác thực. Điều này được thực hiện bằng cách thực hiện thông qua lời gọi HTTP REST API.
  + Hệ thống xác thực cung cấp cho người dùng một mã (token) thông báo AUTH.
  + Mã thông báo AUTH không phải là duy nhất cho mọi yêu cầu và hết hạn sau một khoảng thời gian nhất định. Mọi yêu cầu được gửi cho Swift đều phải kèm theo token.
  + Swift kiểm tra mã thông báo với hệ thống xác thực và lưu trữ kết quả vào bộ nhớ đệm. Kết quả được làm sạch khi hết hạn.
  + Thông thường, hệ thống xác thực có khái niệm tài khoản quản trị và không quản trị. Các yêu cầu của quản trị viên rõ ràng sẽ được thông qua.
  + Các yêu cầu không quản trị được kiểm tra dựa vào danh sách kiểm soát truy cập mức kho chứa (ACL – access control lists). Các danh sách này cho phép quản trị viên thiết lập các quyền đọc ghi cho người dùng không phải quản trị.
  + Như vậy, với các yêu cầu không phải quản trị, ACL được kiểm tra trước khi proxy server có các bước xử lý tiếp theo với yêu cầu này.

# CHƯƠNG III. BẢO MẬT CHO ĐÁM MÂY SỬ DỤNG OPENSTACK

## 3.1: Tổng quan về bảo mật trên đám mây

### 3.1.1. Khái niệm:

* Bảo mật điện toán đám mây (cloud computing security hay viết tắt cloud security) là một lĩnh vực nghiên cứu bao gồm các vấn đề chính sách, công nghệ, và điều khiển để bảo vệ dữ liệu và kiến trúc của điện toán đám mây.

### 3.1.2. Mô hình bảo mật đám mây:

* Bảo mật đám mây gồm 2 mảng: các vấn đề bảo mật của nhà cung cấp điện toán đám mây (các tổ chức cung cấp phần mềm, platform hay cơ sở hạ tầng) gặp phải và các vấn đề bảo mật của khách hàng gặp phải. Các nhà cung cấp điện toán đám mây phải đảm bảo rằng cơ sở hạ tầng mà họ cung cấp phải an toàn với ý nghĩa là các dữ liệu và ứng dụng của khách hàng được bảo vệ.
* Trong khi đó, khách hàng phải đảm bảo rằng các nhà cung cấp đã áp dụng biện pháp an ninh phù hợp để bảo vệ thông tin của họ.

**3.2. Tăng cường bảo mật cho đám mây sử dụng Openstack :**

### 3.2.1. Theo từng mô hình:

#### Hình 3.1: An toàn thông tin trên đám mây

#### 3.2.1.1. Public Cloud

#### Chia sẻ trách nhiệm :

#### Nhiệm vụ bảo mật Public Cloud được phân chia giữa nhà cung cấp và người dùng đám mây, được nêu trong mô hình trách nhiệm chung.

#### Khuôn khổ này chỉ định các khía cạnh cụ thể của bảo mật - và trách nhiệm giải trình - đối với nhà cung cấp và người dùng. Các nhiệm vụ chi tiết cụ thể trong một thỏa thuận bảo mật khác nhau tùy thuộc vào nhà cung cấp đã chọn và mô hình Public Cloud.

#### Những thách thức về bảo mật Public Cloud

* Các tổ chức phải hiểu có rất nhiều thách thức liên quan đến bảo mật đám mây để bảo vệ các ứng dụng được lưu trữ trên đám mây. Public Cloud yêu cầu bảo vệ chống lại các mối đe dọa bên ngoài, chẳng hạn như các cuộc tấn công độc hại và vi phạm dữ liệu, cũng như các rủi ro bảo mật nội bộ, bao gồm các tài nguyên được định cấu hình sai và các chính sách quản lý truy cập.
* Bảo mật đám mây lai tạo ra thêm một loạt thách thức khác nữa. Những phức tạp như bảo mật dữ liệu khi truyền qua internet công cộng và các thành phần mạng cho các môi trường khác nhau đòi hỏi các biện pháp tăng cường bảo vệ.
* Các công cụ và phương pháp bảo mật
* Các dịch vụ và công nghệ bảo mật của nhà cung cấp đám mây bao gồm mã hóa và các công cụ quản lý truy cập và nhận dạng (IAM). Một chiến lược bảo mật toàn diện dựa trên sự kết hợp của những yếu tố này.
* Giám sát bảo mật đám mây là một phần quan trọng của chiến lược bảo mật để cung cấp khả năng phát hiện mối đe dọa.
* Các công cụ giám sát bảo mật quét và quan sát các dịch vụ và tài nguyên trong môi trường đám mây của bạn và tạo cảnh báo khi phát sinh vấn đề bảo mật tiềm ẩn.
* Kiểm soát truy cập cũng rất quan trọng đối với bảo mật Public Cloud. Thiết lập các chính sách IAM mạnh mẽ chỉ phân bổ ở mức quyền cần thiết.
* Cập nhật nhất quán các chính sách IAM và xóa quyền truy cập đối với những người dùng không còn yêu cầu các quyền nhất định. Sử dụng xác thực đa yếu tố để tăng cường xác minh người dùng.

- Ngoài các công cụ và chính sách bảo mật, đội ngũ nhân viên CNTT được đào tạo tốt là không thể thiếu để đảm bảo một môi trường đám mây an toàn. Nhiều lỗ hổng là sản phẩm của việc cấu hình sai tài nguyên do lỗi của con người. Đảm bảo nhân viên CNTT của bạn luôn cập nhật các chính sách bảo mật và thực tiễn cấu hình phù hợp

#### 3.2.1.2. Private Cloud

##### Chức năng mạng ảo bảo mật (VNF)

##### VNFs quản lý các chức năng mạng chạy trên máy ảo. Thông thường, các tổ chức sử dụng nhiều VNF để xây dựng một dịch vụ kết nối mạng quy mô đầy đủ.

* Các VNF mang lại giá trị khác nhau cho các tổ chức tùy theo ngành. Ví dụ, các công ty dịch vụ công nghệ thường sử dụng chúng để triển khai nhanh chóng các dịch vụ mạng mới, tăng doanh thu. Trong khi đó, các tổ chức khác sử dụng chúng như một cách để giảm thời gian tiếp thị cho các sáng kiến mới.+
* Khi các tổ chức tìm kiếm các giải pháp để giải quyết các thách thức về đám mây riêng của họ, họ nên xem xét liệu một công nghệ VNF có thể cho phép bảo mật mạnh mẽ hơn hay không.
* Kết nối với các trình điều phối, như Amdocs, Nuage và OpenStack.
* Sử dụng các khả năng của SDN để tạo một chuỗi dịch vụ gồm các dịch vụ mạng được kết nối.
* Kết nối chuỗi dịch vụ mạng thành một chuỗi ảo.
* Kết hợp hệ thống ngăn chặn xâm nhập (IPS), chống vi-rút, lọc web, tính năng Secure SD-WAN, bảo mật email, WAF và phân tích hộp cát.

### Tuân thủ và quy định

### Các tổ chức phải tuân thủ luật pháp, tiêu chuẩn ngành, kiểm soát nội bộ hoặc một số kết hợp của cả ba. Trong các ngành có quy định cao quản lý lượng lớn dữ liệu cá nhân, sự tuân thủ có thể là động lực chính để triển khai một đám mây riêng.

### Để bảo mật dữ liệu và các hoạt động tuân thủ tài liệu, các tổ chức có đám mây riêng thường sử dụng các giải pháp quản lý sự cố và sự kiện bảo mật (SIEM) hoặc các giải pháp điều phối, tự động hóa và ứng phó bảo mật (SOAR). Khi tìm kiếm một giải pháp cho phép giám sát tuân thủ và lập hồ sơ, các tổ chức phải đảm bảo rằng nó bao gồm các khả năng sau:

### Thu thập và phân tích dữ liệu không xác định từ các nguồn khác nhau, bao gồm nhật ký, số liệu hiệu suất, SNMP Traps, cảnh báo bảo mật và thay đổi cấu hình.

### Máy học (ML) để phân tích hành vi người dùng và thực thể (UEBA) để phát hiện hoạt động bất thường.

### Chấm điểm rủi ro của người dùng và thiết bị.

### 

### \

### Thời gian thực, cơ sở hạ tầng tự động và khám phá ứng dụng cho cơ sở hạ tầng vật lý và ảo.

### Bản đồ truy cập và nhận dạng động kết hợp người dùng, vai trò và các thuộc tính theo ngữ cảnh.

### Giảm thiểu hoặc loại bỏ sự cố tự động.

### Báo cáo và ghi nhật ký hoạt động mạng.

### Tối ưu hóa hoạt động bảo mật, mạng và đám mây

### Khi các tổ chức triển khai và xây dựng các chiến lược đám mây riêng và đám mây lai của họ, số lượng các giải pháp bảo mật có thể sẽ tăng lên đến mức các điểm mù và sự phức tạp sẽ được đưa vào môi trường.

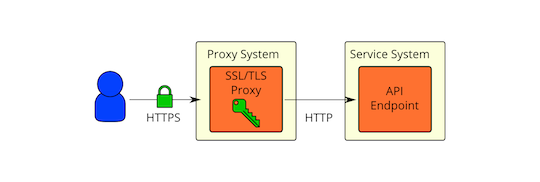
### Thông thường, các tổ chức sẽ coi việc mất khả năng hiển thị, bảo mật và hiệu quả hoạt động như một sự đánh đổi có thể chấp nhận được đối với giá trị kinh doanh thu được từ việc chuyển sang đám mây. Đây là một cách tiếp cận cực kỳ rủi ro để áp dụng vì ngay cả khi mất kiểm soát và khả năng hiển thị trong chốc lát cũng có thể dẫn đến một thỏa hiệp thành công có thể hủy bỏ bất kỳ lợi ích kinh doanh nào mà một tổ chức có thể thu được thông qua việc di chuyển đám mây của họ. Và tệ nhất, điều này thậm chí có thể dẫn đến thua lỗ kinh doanh hoặc đặt một tổ chức vào thế đan xen của các trách nhiệm pháp lý và quy định.

### 3.2.2. Theo từng vai trò:

#### 3.2.2.1. Người vận hành OpenStack

#### Cấu hình bảo mật cho các thành phần trong openStack :

#### Sử dụng https và triển khai giao thức TLS/SSL cho toàn cloud Compute API endpoints.

**Hình 3.2: Triển khai giao thức TLS/SSL khi truy cập vào hệ thống** 

#### Triển khai xác thực 2 yếu tố khi xác thực

#### Triển khai thêm các thiết bị mạng bảo mật cho cloud :

#### Triển khai hệ thống IDS/IPS ( snort)

* Triển khai hệ thống chống tấn công DDOS

#### Sử dụng tường lửa Iptable

#### 

**Hình 3.3: Hệ thống bảo mật Openstack an toàn**

#### Luôn cập nhật phiên bản mới nhất của OpenStack.

* Thiết lập các policy trong hệ thống mạng
* Thiết lập các máy chủ web:
  + Máy chủ web DNS
  + Máy chủ web server apache

#### Hướng dẫn cập nhật tài liệu cho người dung cuối

#### 3.2.2.2. Người sử dụng cuối

#### Triển khai các service được cung cấp bởi OpenStack

#### Firewall as a Service

#### Load Balancing as a Service

#### Back up service , Monitoring & audit Service…

#### Cấu hình , triển khai đám mây an toàn :

#### Phân chia quyền truy cập vào tài nguyên

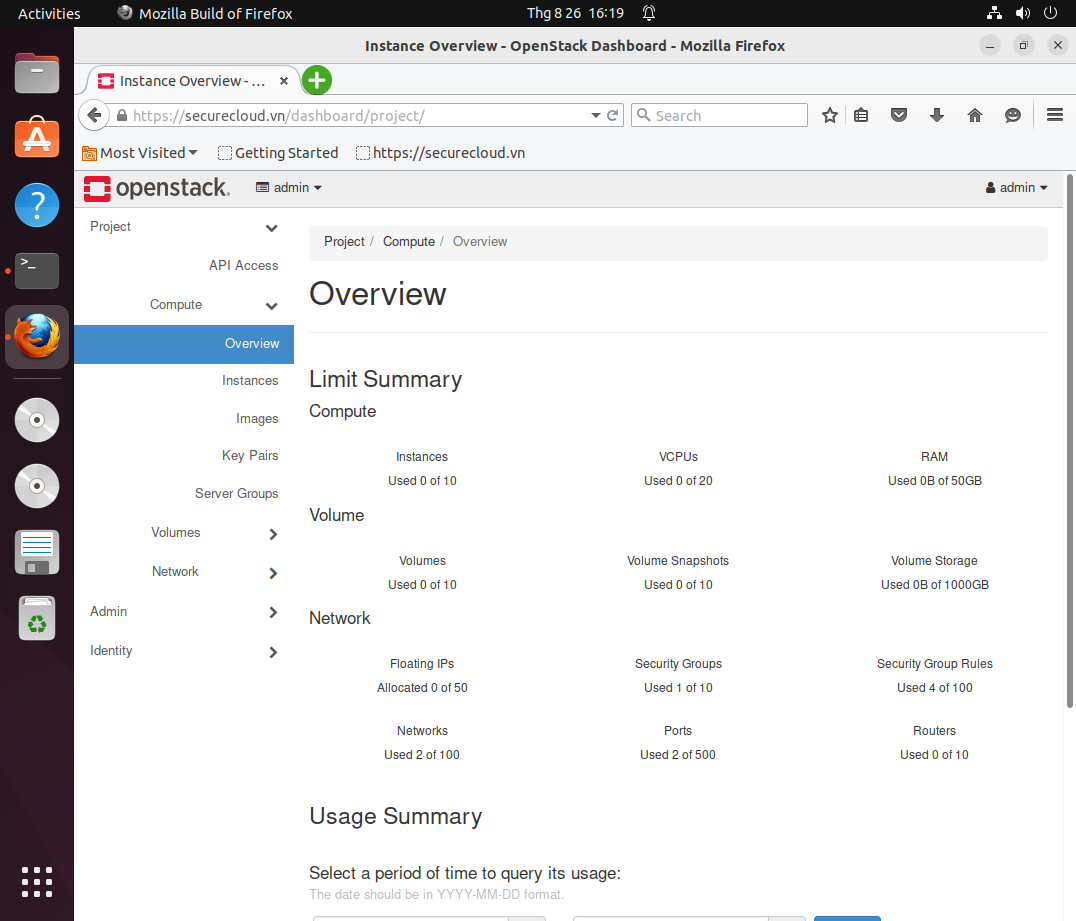
#### Cập nhật phiên bản mới nhất của nhà cung cấp cloud

#### Tuân thủ các policy của cloud provider

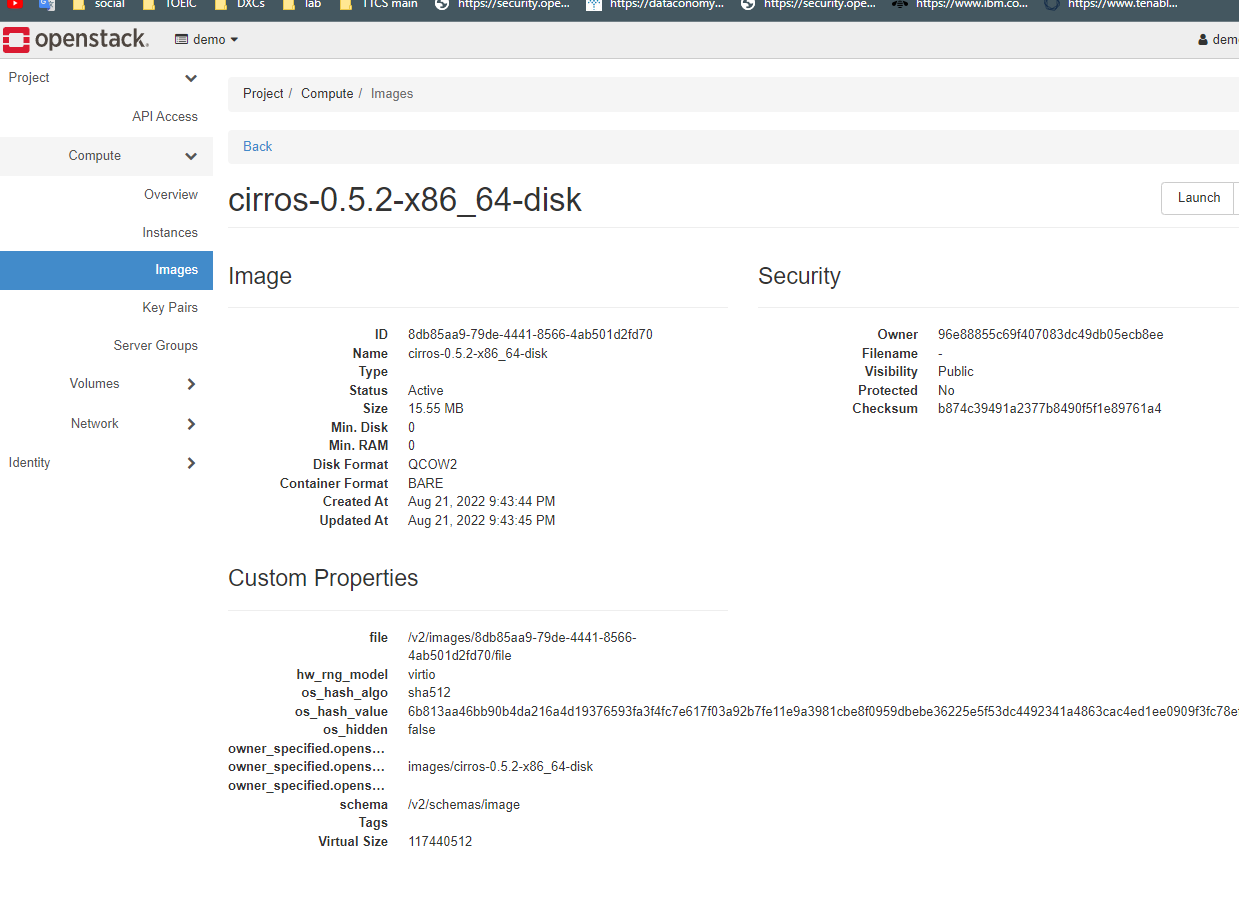
**CHƯƠNG IV. THỰC NGHIỆM TRIỂN KHAI OPENSTACK VÀ TĂNG CƯỜNG BẢO MẬT CHO ĐÁM MÂY IAAS SỬ DỤNG OPENSTACK**

**4.1. Quá trình làm thực nghiệm**

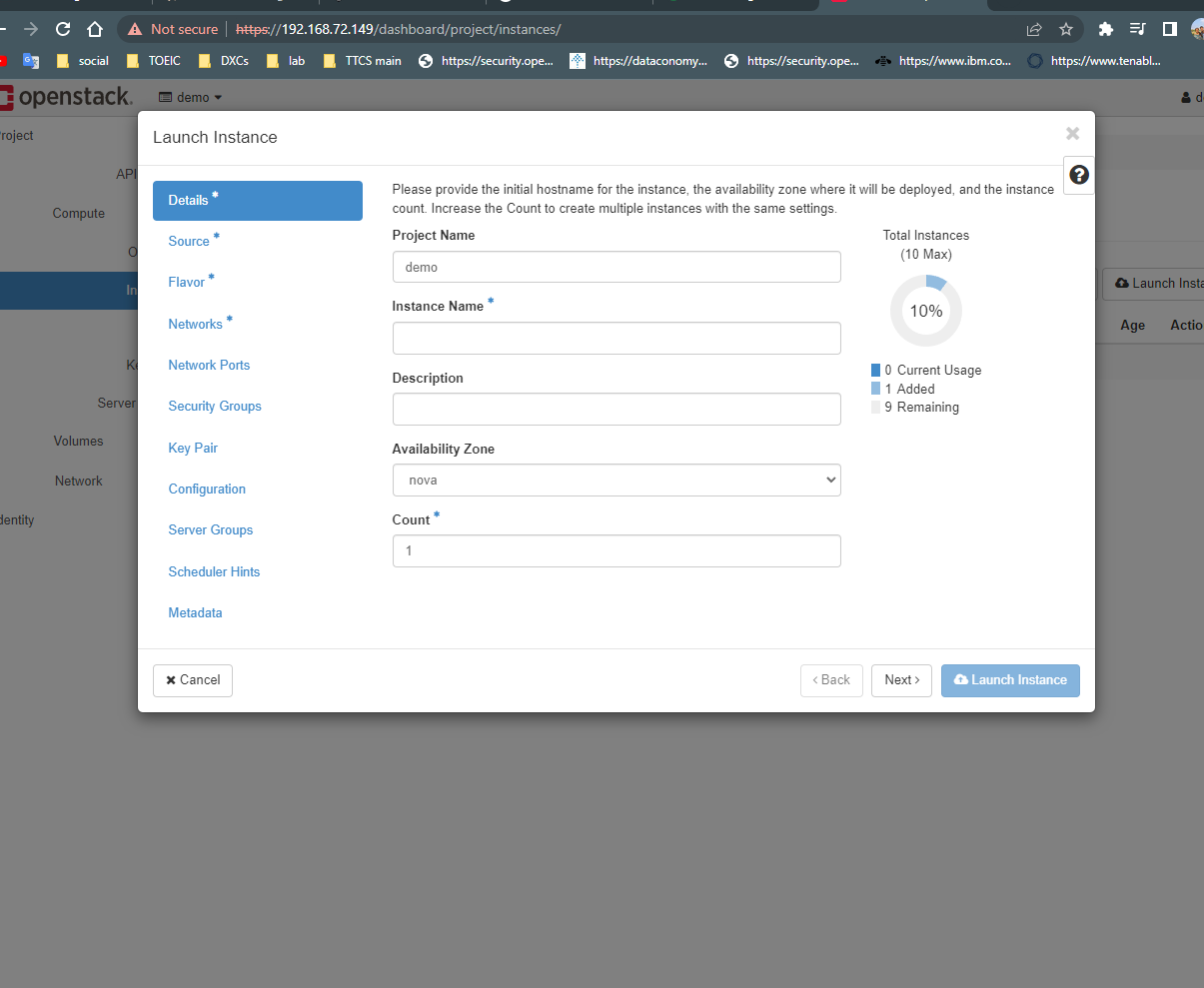
* Chuẩn bị máy ảo sử dụng hệ điều hành ubuntu có cấu hình 8gb ram và 100gb bộ nhớ
* Cài đặt các thành phần chủ yếu chính của OpenStack:
* Web UI Dashboard ( Horizon )

 Hình 4.1 Giao diện của openstack sau khi đăng nhập

* Image Service ( Glance )

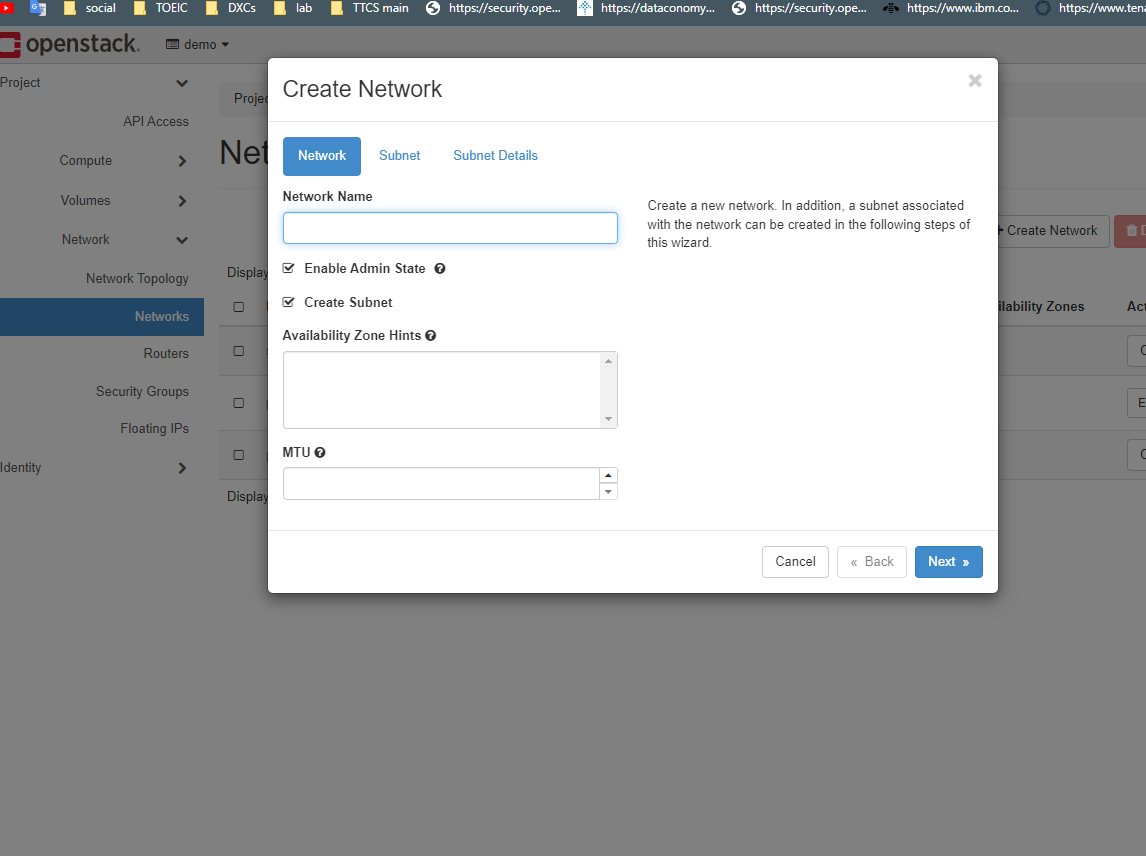
 Hình 4.2 Image Cirros có sẵn của openStack

* Compute Service ( Nova )



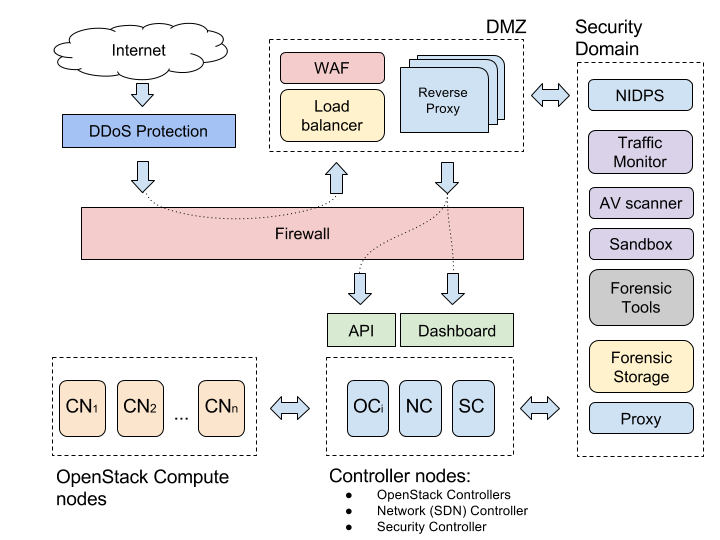
Hình 4.3 Quá trình tạo 1 instance trong openstack

* Network Service ( Neutron )



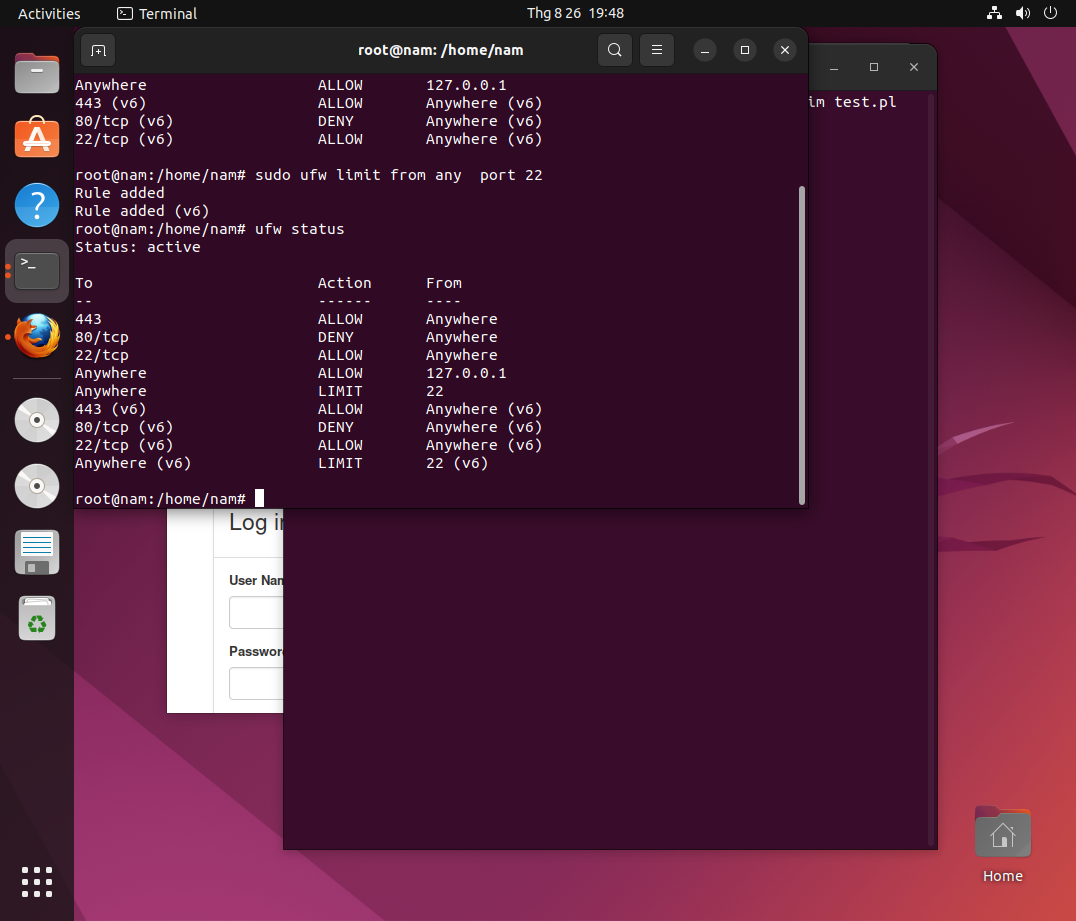
Hình 4.4 Tạo network trong openstack

* Block Storage Service ( Cinder )
* Object Storage Service ( Swift )
* Triển khai hệ thống mạng an toàn :



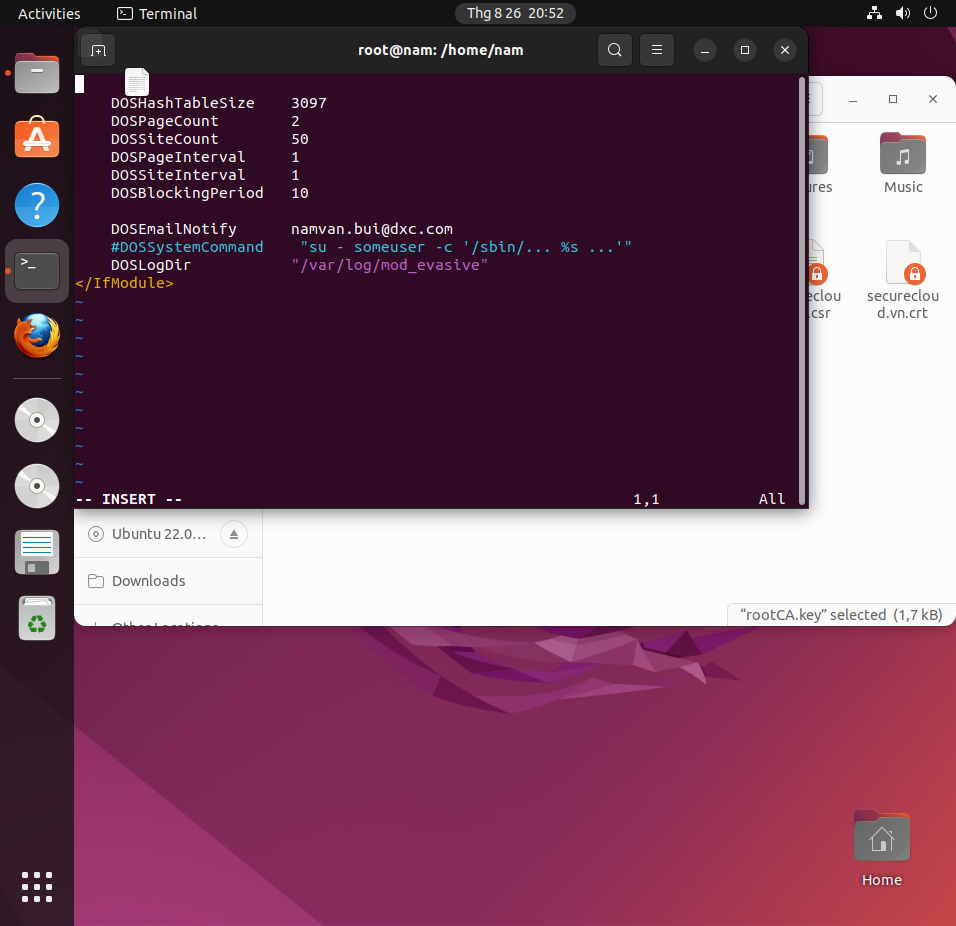
Hình 4.5 Triển khai mô hình mạng an toàn mạng

* Cấu hình tường lửa iptables

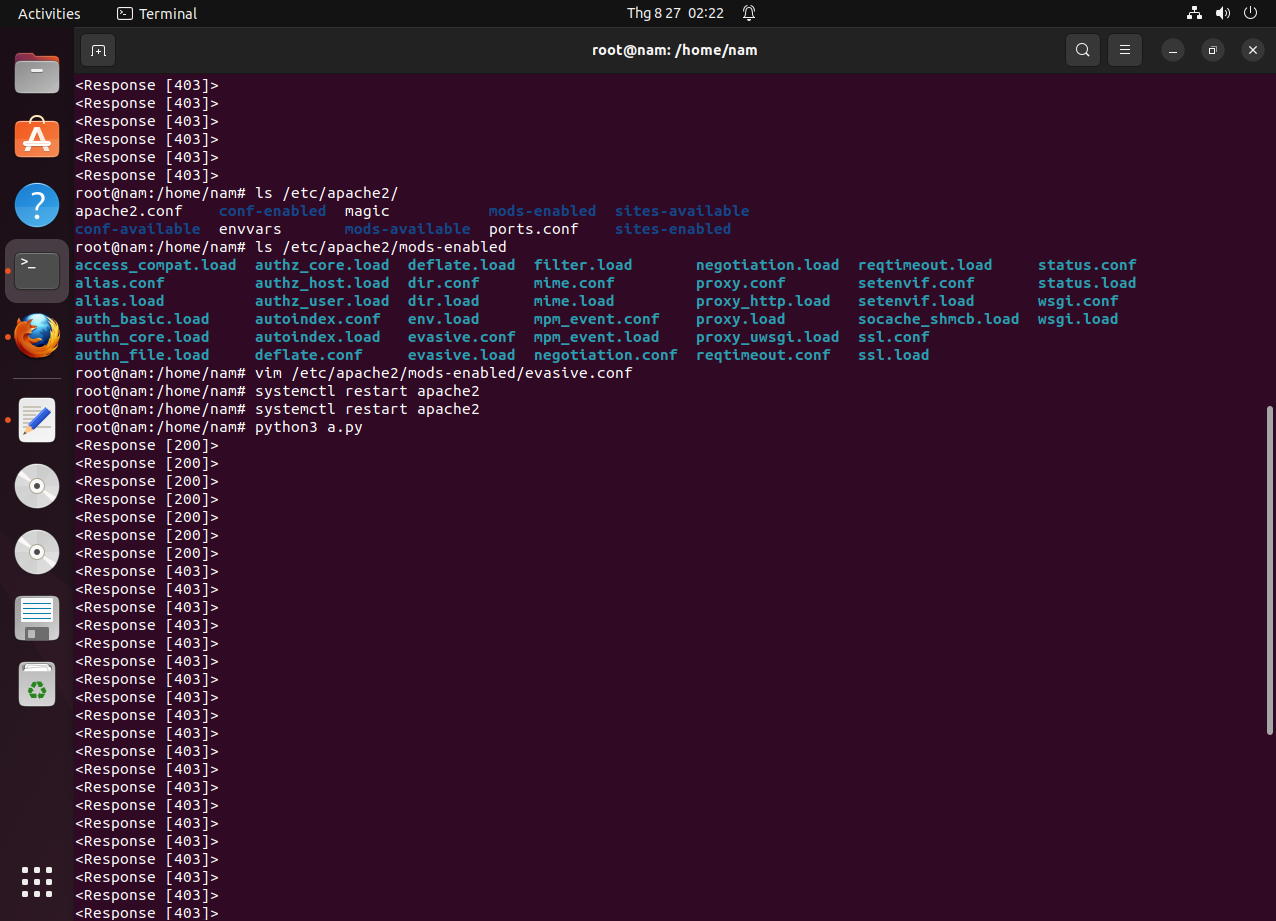
****

Hình 4.6 Cấu hình tường lửa trong mạng

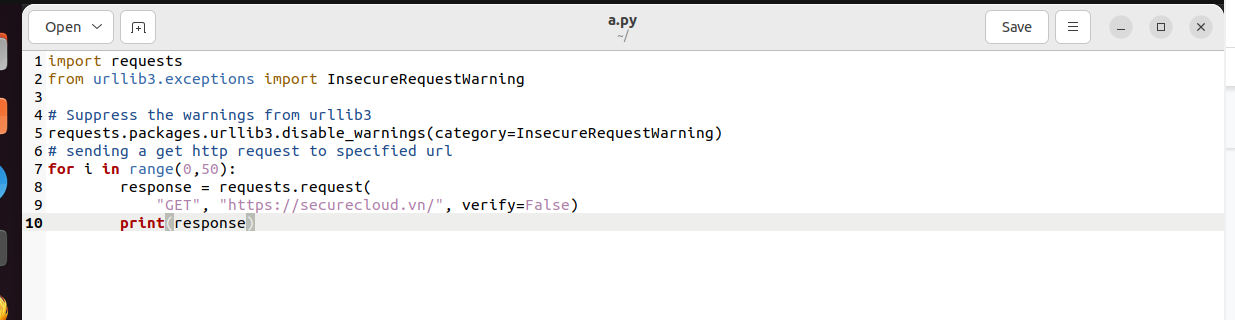
* Sử dụng hệ thống chống DDOS ( DDOS protection)
  + Sử dụng lệnh apt install libapache2-mod-evasive để cài đặt evasive
  + Vào file /etc/apache2/mods-enabled/evasive.conf để cấu hình



Hình 4.7 Cấu hình file evasive để chặn tấn công ddos

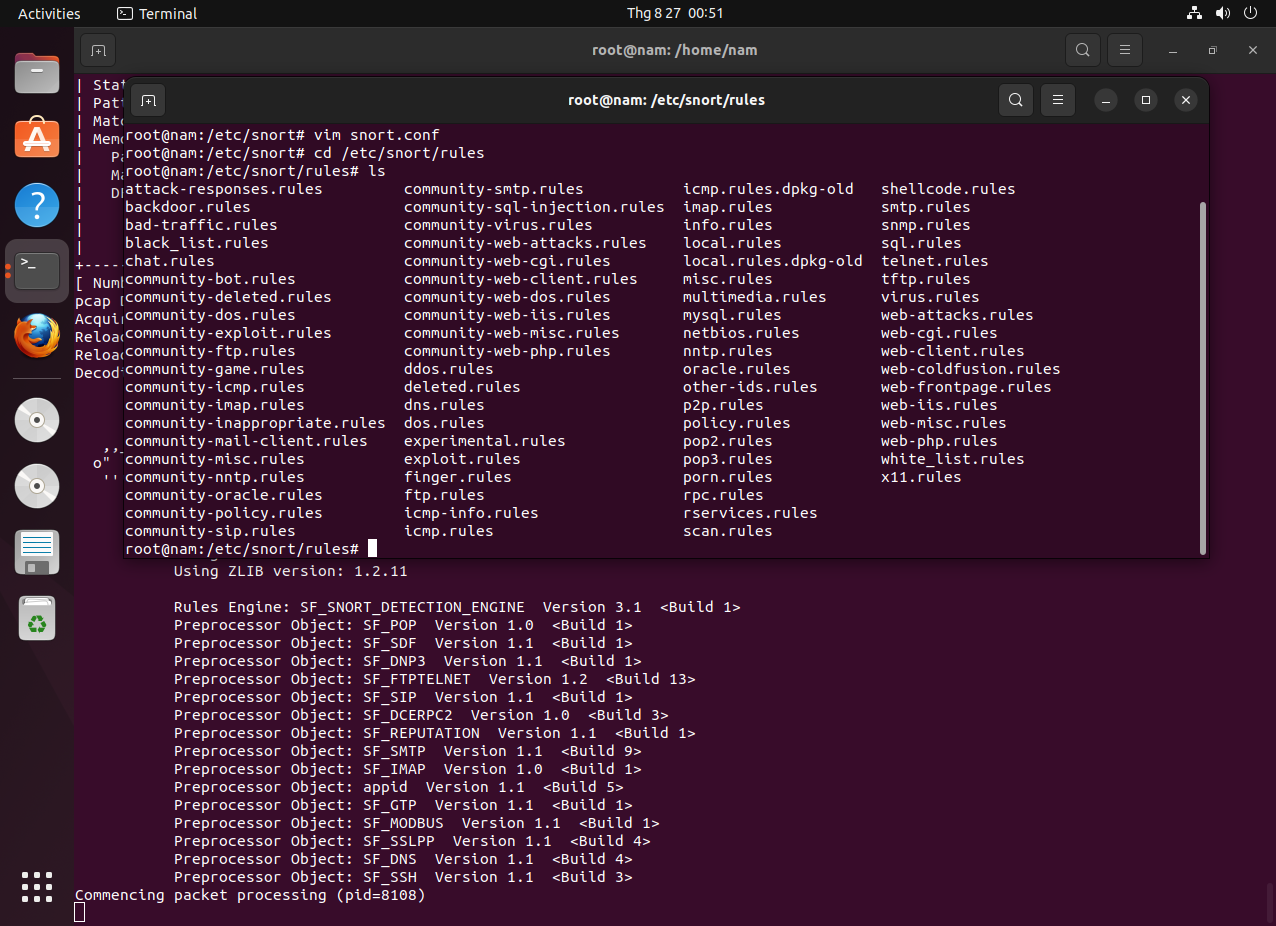


Hình 4.8 Hệ thống DDos protection đã chặn khi gửi quá số request cho phép

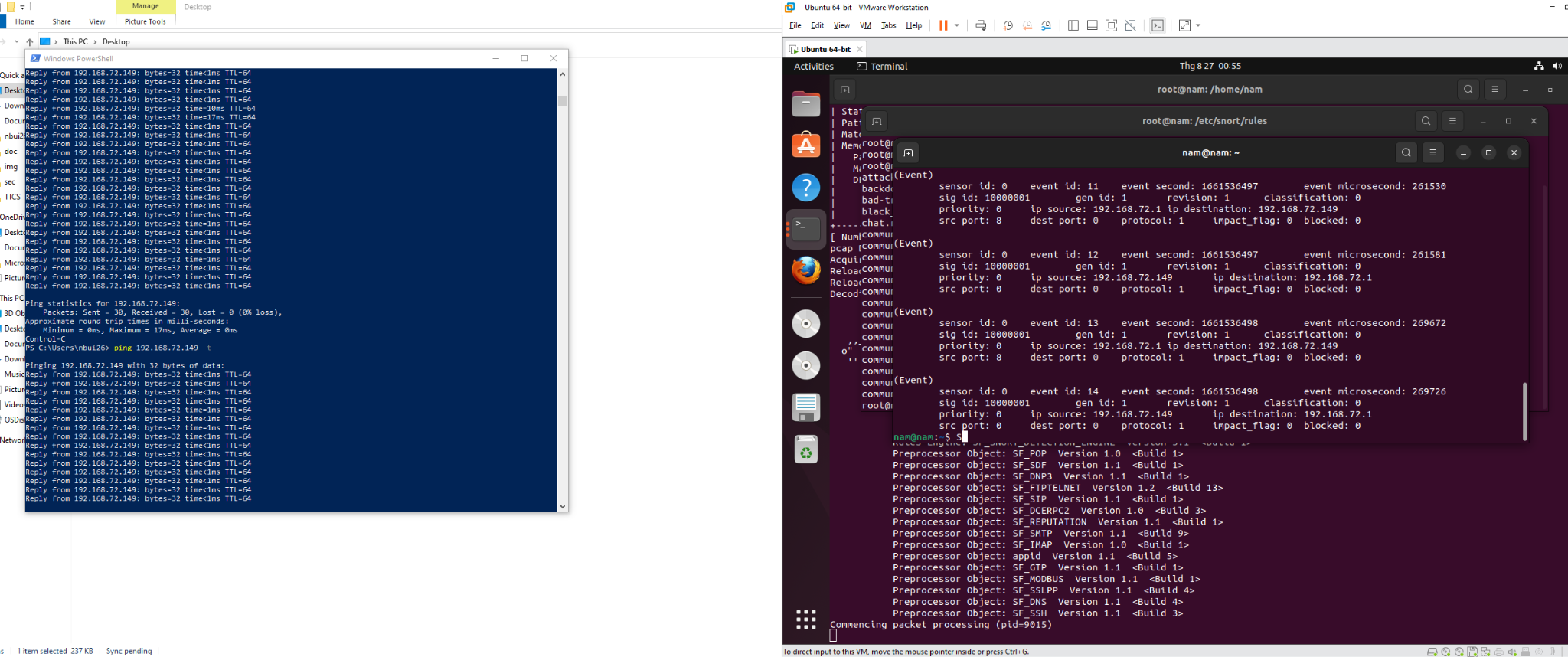


Hình 4.9 File gửi request đến server sử dụng python

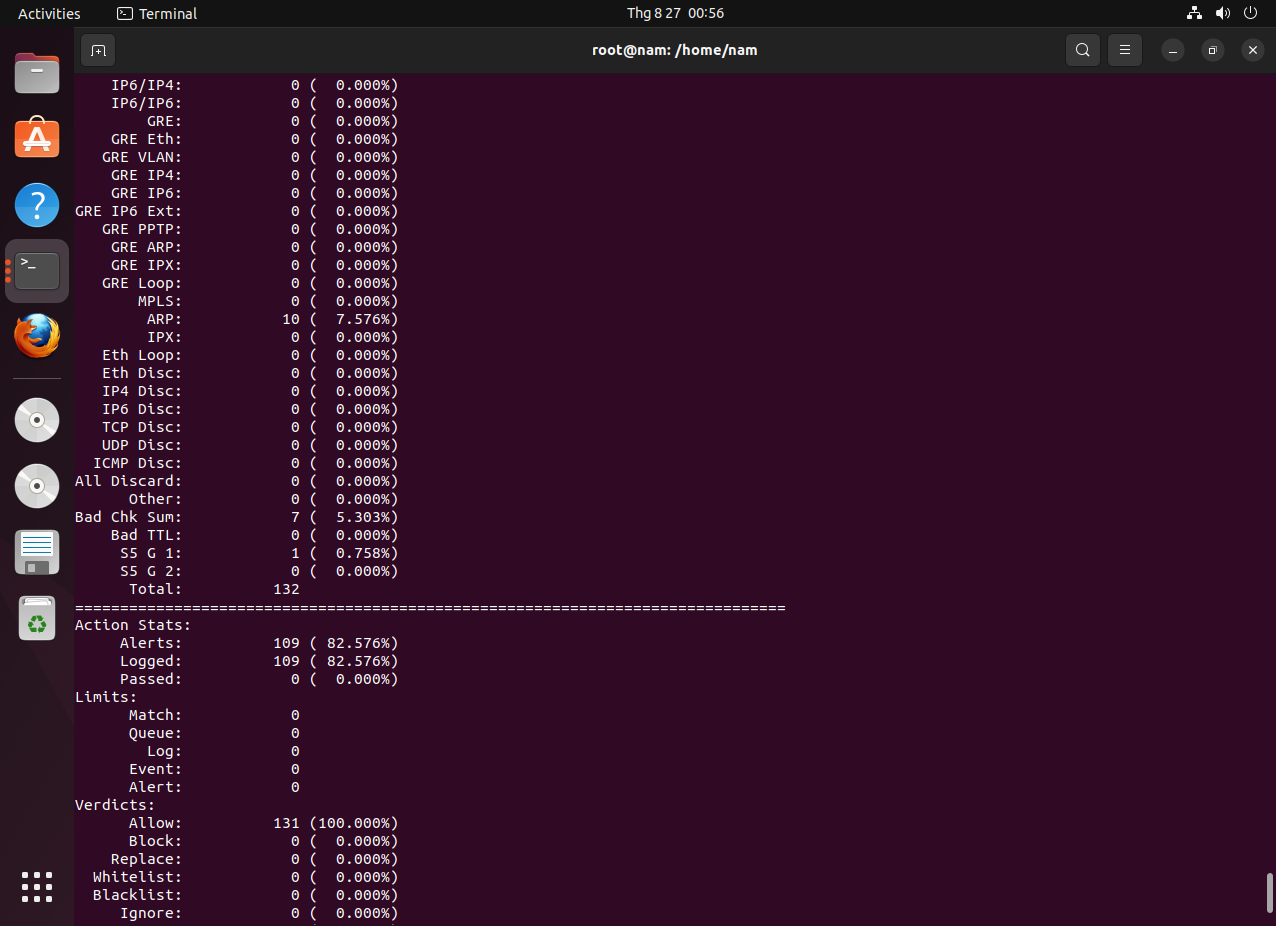
Triển khai hệ thống IDS/ IPS



Hình 4.10 Các rules đã được cài đặt

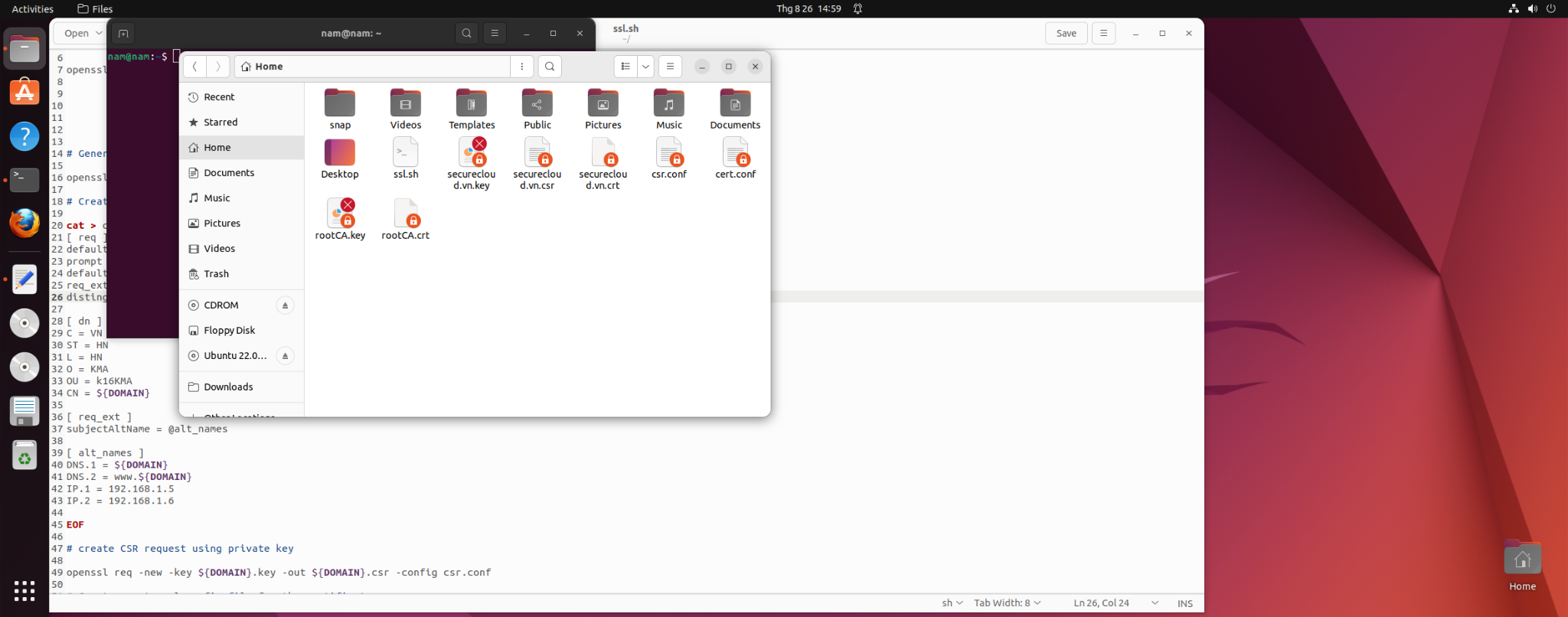


Hình 4.11 Kiểm tra thử icmp ping



Hình 4.12 ết quả có alert về icmp

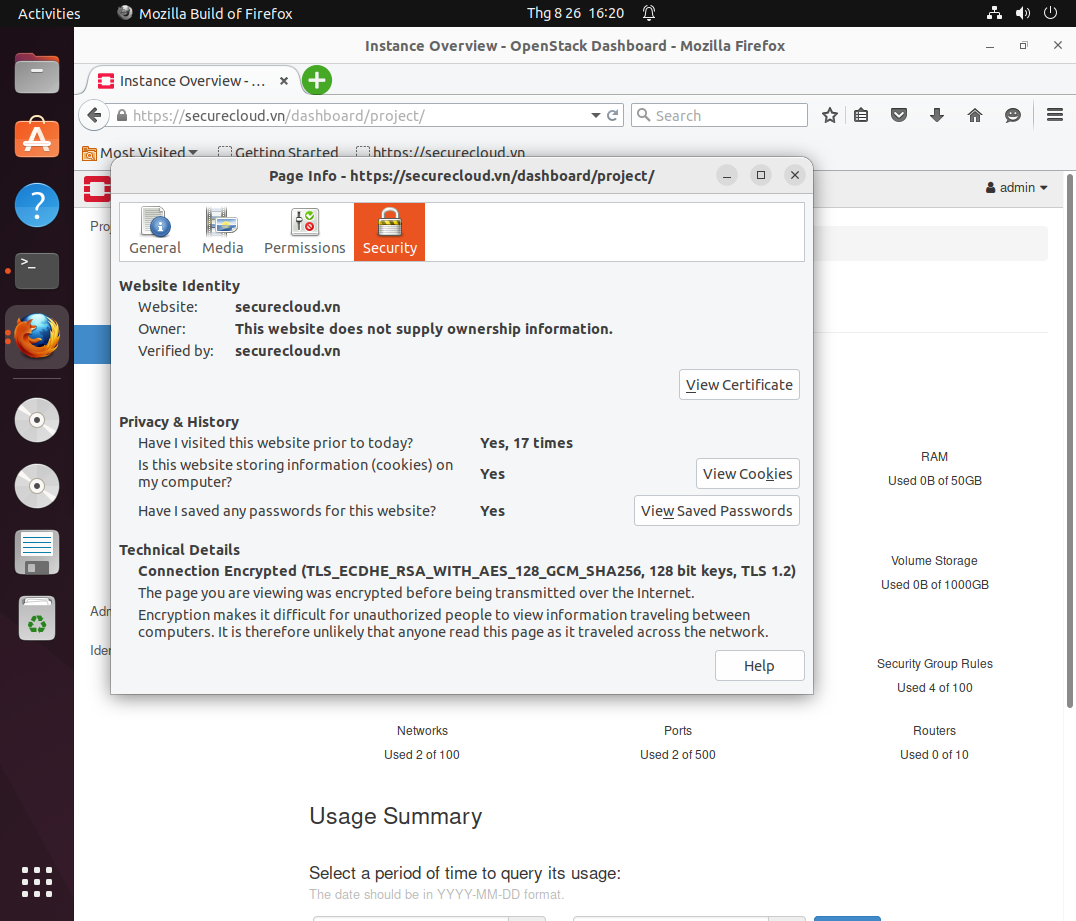
* Sử dụng giao thức HTTPS cho Web Interface và API
* Tạo certificate authority sử dụng trong giao thức TLS/ SSL trong OpenStack



Hình 4.13 Tạo private key , Cert authority

****

Hình 4.14 Cấu hình horizon.conf để nhận Cert Authority vừa tạo

****

Hình 4.15 SSL đã được kích hoạt khi truy cập Openstack bằng dashboard

**KẾT LUẬN**

Về lý thuyết, đồ án đã trình bày các khái niệm, thông tin liên quan liên quan đến triển khai đám mây IaaS bằng OpenStack và an toàn thông tin trong OpenStack. Đưa ra đề xuất một mô hình bảo mật an toàn đối với đám mây tự triển khai. Mô tả chi tiết quá trình triển khai hệ thống đám mây, cấu hình các tính năng, module cần thiết. Ngoài ra, đồ án đã trình bày các công nghệ, kiến thức trọng tâm trong được sử dụng trong công nghệ điện toán đám mây và bảo mật đám mây sử dụng bởi OpenStack

Về thực nghiệm, đồ án đã xây dựng thành công một hệ thống đám mây cơ bản. Hệ thống được tích hợp thành công những tính năng nâng cao, khó cài đặt. Bên cạnh đó, đồ án cũng xây dựng thành công một hệ thống đám mây sử dụng những thiết bị để đảm bảo an toàn thông tin trong đám mây tự tạo. Thực nghiệm trong đồ án mang tính khả thi, đáp ứng mục tiêu đã đề ra và sẵn sàng áp dụng sử dụng vào thực tế.

Dù vậy, đồ án vẫn còn một số hạn chế. Trước hết,vì hầu hết tài liệu là tiếng Anh nên Việt hoá chưa thực sự đầy đủ nên nhiều chỗ còn xuất hiện tiếng Anh. Tuy các chủ đề rất đa dạng và mỗi chủ đề lại cho phép chúng ta tuỳ biến rất sâu, nhưng chung quy lại, các cấu trúc xương sống của chủ đề vẫn không thay đổi, dẫn đến việc có sự tương đồng nhất định giữa các hệ thống sử dụng cùng chủ đề. Ngoài ra, đồ án chỉ xây dựng trong môi trường hệ thống chưa công khai được. Việc giải quyết những phần hạn chế này cũng chính là hướng phát triển tương lai của đồ án.

Chúng em xin cảm ơn thầy giáo đã theo dõi , góp ý , hướng dẫn , đánh giá khách quan nhất bài báo cáo này giúp chúng em rút kinh nghiệm làm tốt cho những bài tiếp theo . Một lần nữa , chúng em xin chân thành cảm ơn !

**TÀI LIỆU THAM KHẢO**

<https://www.academia.edu/31107357/Cloud_computing_v%C3%A0_OpenStack_Gi%E1%BB%9Bi_thi%E1%BB%87u_Cloud_computing_v%C3%A0_tri%E1%BB%83n_khai_tr%C3%AAn_OpenStack_Cloud_computing_v%C3%A0_OpenStack>

<https://viettelco.vn/dien-toan-dam-may-voi-openstack/>

<https://hiepsharing.com/top-nhung-nha-cung-cap-dien-toan-dam-may-2021/>

<https://github.com/openstack/security-doc>

<https://docs.openstack.org/security-guide/>

<https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/13/html/security_and_hardening_guide/index>

<https://docs.mirantis.com/mcp/q4-18/mcp-security-best-practices/single/index.html>

https://github.com/hocchudong/thuctap012017/tree/master/XuanSon/OpenStack