Giới thiệu Cloud computing và triển khai trên OpenStack

Thực hiện: **Lê Minh Chí**

Nguyễn Sơn Tùng

Contents

Thuật ngữ viết tắt

CC	Cloud computing
IaaS	Infrastructure as a Service
PaaS	Platform as a Service
SaaS	Software as a Service
CSA	Cloud Security Alliance
SLA	Service Level Agreement
NIST	National Institute of Standard and Technology
AWS	Amazon Web Service
HĐH	Hệ Điều Hành
VMM	Virtual Machine Monitor

Tóm lược

Cloud computing (CC) đang là chủ đề được bàn luận sôi nổi nhất hiện nay, các công nghệ liên quan đến 'cloud' nhận được rất nhiều quan tâm từ người dùng và doanh nghiệp. Đã có khá nhiều sản phẩm thương mại cũng như nguồn mở miễn phí được giới thiệu cung cấp cho người dùng khả năng xây dựng các thành phần của CC, từ hạ tầng IaaS đến PaaS và SaaS. Tuy nhiên tất cả vẫn đang trong quá trình phát triển, sẽ rất sai lầm nếu chỉ nghe theo quảng cáo từ các nhà cung cấp đó. Để có nhận xét chính xác và chi tiết hơn về hiện trạng của các sản phẩm này, cách tốt nhất là hãy thử nghiệm chúng.

Một trong những ưu điểm của CC là nó sử dụng hiệu quả hơn các tài nguyên từ hệ thống vật lý và hiệu suất sử dụng năng lượng cao hơn. IaaS chính là thành phần quan trọng nhất giúp cho CC thực hiện được điều này. Là thành phần quản lý hạ tầng về phần cứng, mạng và phân phối lại các tài nguyên này, IaaS chính là phần cung cấp cho người dùng khả năng xây dựng hạ tầng cơ sở cho đám mây riêng của họ (Private Cloud).

Trong báo cáo này nhóm xin trình bày một số thử nghiệm bước đầu về một trong những IaaS đang được quan tâm nhất hiện nay: Openstack. Là một dự án nguồn mở được tham gia bởi hơn 160 công ty lớn trên thế giới, Openstack mang đến cho các doanh nghiệp khả năng xây dựng các đám mây riêng phục vụ cho công việc nội bộ hoặc lớn hơn là đám mây để cung cấp dịch vụ liên quan tới CC.

Trong phần đầu của báo cáo sẽ giới thiệu một số khái niệm về CC và các công nghệ ảo hóa. Phần tiếp theo xin được trình bày về Openstack, các công việc thử nghiệm đã và kết quả đạt được. Phần cuối sẽ phân tích security trong một hệ thống CC hoàn chỉnh để so sánh đánh giá với OpenStack. Các tài liệu tham khảo cũng như hướng dẫn chi tiết về cài đặt, cấu hình... được đính kèm trong phần phụ lục tham khảo.

Phần 1: Cloud Computing và các giải pháp

I. Cloud computing

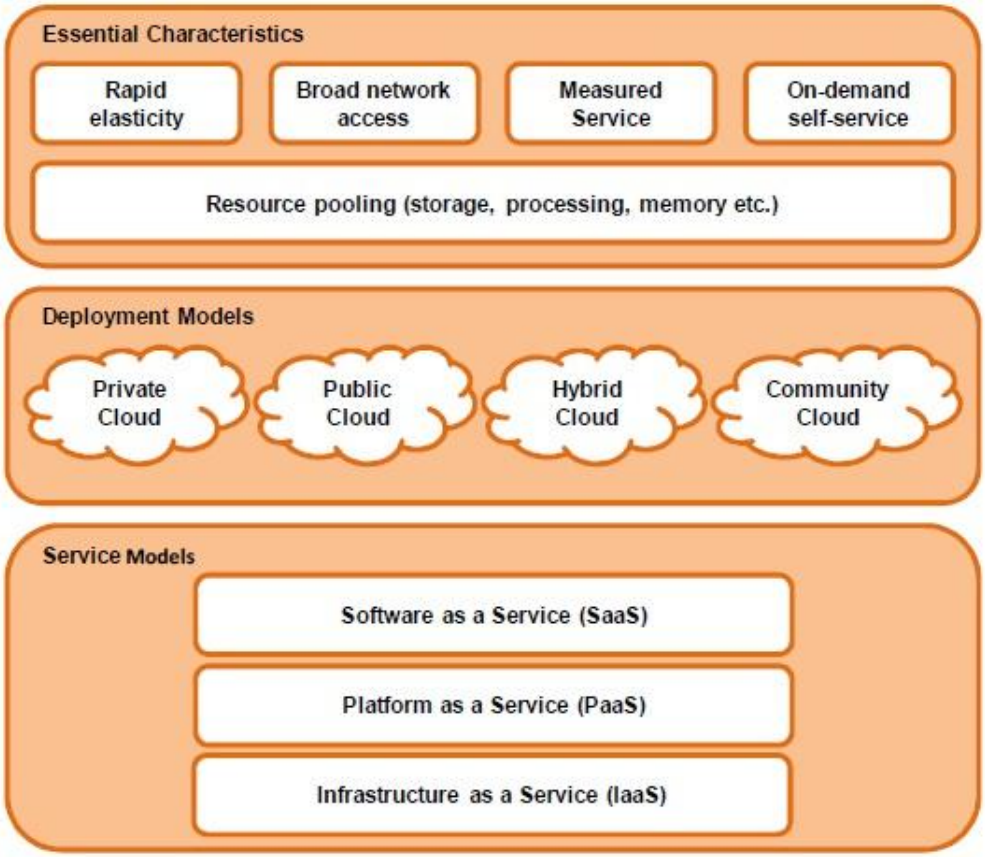
1. Giới thiệu về "Điện toán đám mây"

Điện toán đám mây (*cloud computing*) hay còn gọi là điện toán máy chủ ảo nơi các tính toán được "định hướng dịch vụ" và phát triển dựa vào Internet. Cụ thể hơn, trong mô hình điện toán đám mây, tất cả các tài nguyên, thông tin, và software đều được chia sẻ và cung cấp cho các máy tính, thiết bị, người dùng dưới dạng dịch vụ trên nền tảng một hạ tầng mạng công cộng (thường là mạng Internet) [1, 2]. Các users sử dụng dịch vụ như cơ sở dữ liệu, website, lưu trữ, … trong mô hình cloud computing không cần quan tâm đến vị trí địa lý cũng như các thông tin khác của hệ thống mạng đám mây - "điện toán đám mây trong suốt đối với người dùng".

Người dùng cuối truy cập và sử dụng các ứng dụng đám mây thông qua các ứng dụng như trình duyệt web, các ứng dụng mobile, hoặc máy tính cá nhân thông thường. Hiệu năng sử dụng phía người dùng cuối được cải thiện khi các phần mềm chuyên dụng, các cơ sở dữ liệu được lưu trữ và cài đặt trên hệ thống máy chủ ảo trong môi trường điện toán đám mây trên nền của "data center".

- *"Data center" là thuật ngữ chỉ khu vực chứ server và các thiết bị lưu trữ, bao gồm nguồn điện và các thiết bị khác như rack, cables, …có khả năng sẵn sàng và độ ổn định cao. Ngoài ra còn bao gồm các tiêu chí khác như: tính module hóa cao, khả năng mở rộng dễ dàng, nguồn và làm mát, hỗ trợ hợp nhất server và lưu trữ mật độ cao [3].*

Hình bên dưới mô tả một định nghĩa về CC bao gồm 5 tính năng chính, với 4 mô hình triển khai, và 3 mô hình dịch vụ.



Hình 1: Tổng quan Cloud Computing (NIST) [4]

- 5 tính năng trong CC tùy thuộc vào mô hình triển khai thực tế có thể khác nhau. Ví dụ trong mô hình private cloud, tài nguyên được sử dụng bởi chỉ 1 doanh nghiệp thì tính năng "On-demand service" hay "resource pool" sẽ khác so với các mô hình khác.
    - ***Rapid elasticity**: nhà cung cấp CC dễ dàng chỉ định cũng như thu hồi tài nguyên người dùng rất nhanh chóng. Về phía người dùng được phép yêu cầu một tài nguyên "không giới hạn" và chỉ việc chi trả theo tiền.*
    - ***Broad network access**: truy cập vào các tài nguyên máy tính dễ dàng thông qua các cơ chế network tiêu chuẩn.*
    - ***Measured service**: provider đảm bảo việc tính toán lượng tiêu dùng của khách hàng. Mô hình hướng đến là "pay as you go".*
    - ***On-demand self-service**: cho phép khách hàng tùy chỉnh tài nguyên sử dụng mà không cần phải thông báo hay qua bất kỳ sự can thiệp nào của provider.*
    - ***Resource pooling**: các loại tài nguyên vật lý và ảo của CC được chia sẻ với nhau và tự động cấp cho các users.*
- Có 3 mô hình triển khai điện toán đám mây chính là public (công cộng), private (riêng), và hybrid ("lai" giữa đám mây công cộng và riêng). Đám mây công cộng là mô hình đám mây mà trên đó, các nhà cung cấp đám mây cung cấp các dịch vụ như tài nguyên, platform, hay các ứng dụng lưu trữ trên đám mây và public ra bên ngoài. Các dịch vụ trên public cloud có thể miễn phí hoặc có phí [5]. Đám mây riêng thì các dịch vụ được cung cấp nội bộ và thường là các dịch vụ kinh doanh, mục đích nhắm đến cung cấp dịch vụ cho một nhóm người và đứng đằng sau firewall. Đám mây "lai" là môi trường đám mây mà kết hợp cung cấp các dịch vụ công cộng và riêng [5]. Ngoài ra còn có "community cloud" là đám mây giữa các nhà cung cấp dịch vụ đám mây.
- Về mô hình cung cấp dịch vụ có 3 loại chính là IaaS – cung cấp hạ tầng như một server, PaaS – cung cấp Platform như một service, và SaaS – cung cấp software như một service.

Trên đây là định nghĩa của NIST về CC, phần tiếp theo sẽ trình bày về các lợi ích của CC nhằm nổi bật các tính năng so với các mô hình truyền thống.

2. Những lợi ích của "Điện toán đám mây"

Có thể kể ra một số lợi ích cơ bản và đặc trưng của hệ thống "Điện toán đám mây" như sau [6]:

- Tăng sự linh hoạt của hệ thống (Increased Flexibility): khi cần thêm hay bớt một hay vài thiết bị (storaged devices, servers, computers, …) chỉ cần mất vài giây.
- Sử dụng tài nguyên theo yêu cầu (IT Resources on demand): tùy thuộc vào nhu cầu của khách hàng mà administrator setup cấu hình hệ thống cung cấp cho khách hàng.
- Tăng khả năng sẵn sàng của hệ thống (Increased availability): các ứng dụng và dịch vụ được cân bằng động để đảm bảo tính khả dụng. Khi một trong các hardware bị hư hỏng không làm ảnh hưởng đến hệ thống, chỉ suy giảm tài nguyên hệ thống.
- Tiết kiệm phần cứng (Hardware saving): mô hình truyền thống trong nhiều trường hợp cần một hệ thống riêng biệt cho mỗi tác vụ, dịch vụ. Điều này gây ra lãng phí,

trong mô hình "Điện toán đám mây", các tài nguyên IT được quản lý để đảm bảo sự không lãng phí này.
- Cung cấp các dịch vụ với độ sẵn sàng gần như 100% (taking down services in real time)
- Trả theo nhu cầu sử dụng thực tế (Paying-as-you-go IT): mô hình "Cloud computing" tích hợp với hệ thống billing để thực hiện việc tính cước dựa theo dung lượng người dùng đối với các tài nguyên như tốc độ CPU, dung lượng RAM, dung lượng HDD, …

Tóm lại, mô hình "Điện toán đám mây" đã khắc phục được 2 yếu điểm quan trọng của mô hình truyền thống về "khả năng mở rộng (scalability)" và "độ linh hoạt (flexibility)". Các tổ chức cũng như công ty có thể triển khai ứng dụng và dịch vụ nhanh chóng, chi phí giảm, và ít rủi ro[6]. Phần tiếp theo sẽ giới thiệu về ảo hóa – là công nghệ cốt lõi và được xem như là một bước đệm chuyển tiếp từ mô hình truyền thống sang CC.

3. Các công nghệ ảo hóa (Virtualization Technologies)

3.1. Kernel mode và User mode

Trước khi đi vào chi tiết các công nghệ ảo hóa xin được sơ lược một số khái niệm liên quan đến việc xử lý trên tài nguyên phần cứng của một hệ điều hành. Thông thường một HĐH khi được cài đặt sẽ có 2 modes hoạt động chính:

- ***Kernel mode:*** *đây là không gian được bảo vệ nơi mà "nhân" của HĐH xử lý và tương tác trực tiếp với phần cứng. Một ví dụ điển hình cho Kernel mode là các drivers của thiết bị. Khi có sự cố thì hệ thống ngưng hoạt động và thông báo lỗi như ở windows sẽ hiển thị màn hình xanh khi có lỗi giao tiếp phần cứng.*
- ***User mode:*** *đây là không gian nơi các ứng dụng chạy, ví dụ Office, MySQL, hay Exchange server. Khi có sự cố ở các ứng dụng thì chỉ có các ứng dụng ngưng hoạt động mà không ảnh hưởng gì đến server.*

Khi một ứng dụng cần truy cập vào tài nguyên phần cứng, ví dụ đĩa cứng hay network interface, ứng dụng đó cần giao tiếp với driver thích hợp chạy trong kernel mode. Sự chuyển đổi qua lại giữa User mode và Kernel mode cũng là những "tiến trình-process" và cũng chiếm dụng tài nguyên hệ thống (CPU, RAM, …).

3.2. Hypervisor

Tất cả các loại ảo hóa được quản lý bởi VMM (Virtual Machine Monitor). VMM về bản chất cũng được chia làm 2 loại là:

- *VMM đóng vai trò như một phần mềm trung gian chạy trên HĐH để chia sẻ tài nguyên với HĐH. Ví dụ: VMware workstation, Virtual PC, KVM.*
- *VMM đóng vai trò là một hypervisor chạy trên phần cứng. Ví dụ: VMware ESXi, Hyper-V, Xen.*

Hypervisor là một phần mềm nằm ngay trên phần phần cứng hoặc bên dưới HĐH nhằm mục đích cung cấp các môi trường tách biệt gọi là các phân vùng – partition. Mỗi phân vùng ứng với mỗi máy ảo-VM có thể chạy các HĐH độc lập.

Hiện nay có 2 hướng tiếp cận hypervisor khác nhau (loại 2 – hypervisor VMM) với tên gọi: Monolithic và Micro hypervisor.

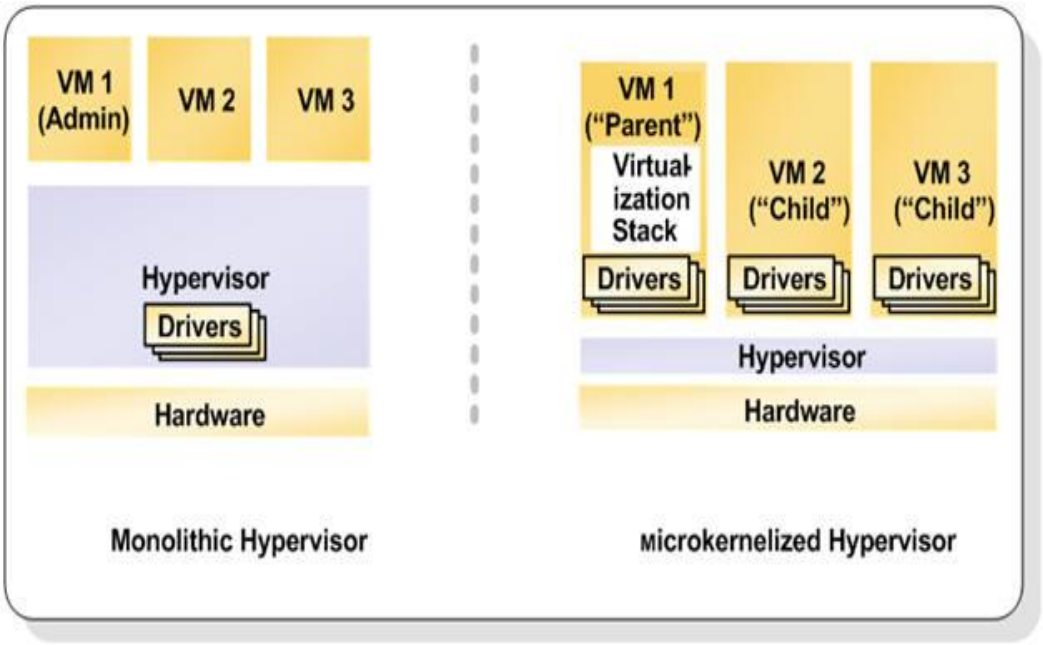


Figure 2: Monolithic và Microkernelized Hypervisor [7]

- ***Monolithic hypervisor****: hypervisor có driver riêng biệt để truy cập tài nguyên phần cứng bên dưới. Các VMs truy cập tài nguyên hệ thống thông qua drivers của hypervisor. Điều này mang lại hiệu suất cao, tuy nhiên khi driver trên hypervisor bị sự cố thì cả hệ thống ngưng hoạt động, hoặc phải đối mặt với vấn đề an ninh khi drivers có thể bị giả dạng bởi malware, một rủi ro trong môi trường ảo hóa.*
- ***Micro-kernelized hypervisor****: loại hypervisor này không có driver bên trong hypervisor mà chạy trực tiếp trên mỗi partition. Một VM sẽ đóng vai trò partition cha quản lý và khởi tạo các partition con (VM con). VM cha cũng bao gồm nhiều tính năng khác như quản lý memory, lưu trữ drivers, … Điều này mang lại sự an toàn và tin cậy. Tuy nhiên nó cũng gặp phải vấn đề về độ sẵn sàng (availability) khi partition cha gặp sự cố, hệ thống cũng bị ngưng trệ.*

3.3. Full-virtualization:

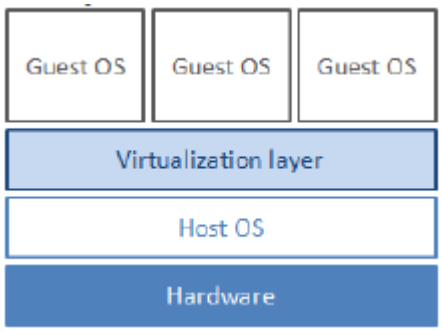


Figure 3: Full-virtualization

- Full-virtualization là công nghệ ảo hóa để cung cấp 1 loại hình máy ảo dưới dạng mô phỏng của 1 máy chủ thật với đầy đủ tất cả các tính năng bao gồm input/output operations, interrupts, memory access, … Hình 3 miêu tả mô hình ảo hóa Full-Virtualization với layer Virtualization để thực hiện chức năng ảo hóa, cung cấp các máy chủ ảo (Guest OS) [8]. Tuy nhiên mô hình ảo hóa này không thể khai thác tốt hiệu năng khi phải thông qua một trình quản lý máy ảo (Virtual Machines monitor

hay hypervisor) để tương tác đến tài nguyên hệ thống (mode switching). Vì vậy sẽ bị hạn chế bớt 1 số tính năng khi cần thực hiện trực tiếp từ CPU. Xen, VMWare workstation, Virtual Box, Qemu/KVM, và Microsoft Virtual Server hỗ trợ loại ảo hóa này [9].

3.4. Para-virtualization

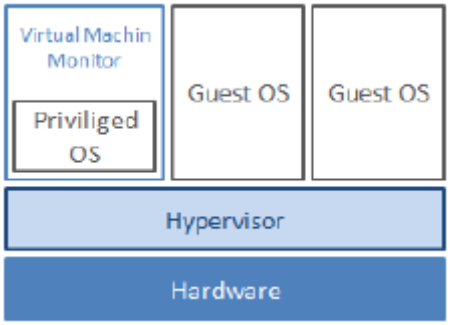


Figure 4: Para-virtualization

- Para-virtualization hay còn gọi là ảo hóa "một phần" là kỹ thuật ảo hóa được hỗ trợ và điều khiển bởi 1 hypervisor nhưng các Oss của guest thực thi các lệnh không phải thông qua Hypervisor (hay bất kỳ 1 trình quản lý máy ảo nào) nên không bị hạn chế về quyền hạn. Tuy nhiên nhược điểm của loại ảo hóa này là các OS biết đang chạy trên 1 nền tảng phần cứng ảo và khó cấu hình cài đặt. Ảo hóa Para-Virtualization được hỗ trợ bởi Xen, VMware, Hyper-V, và UML [9, 10].

3.5. OS-level virtualization (Isolation)

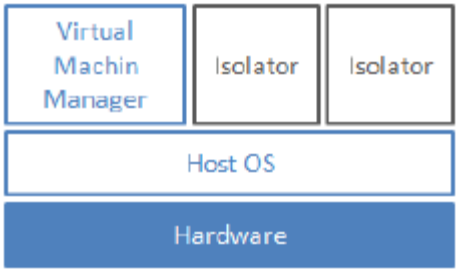


Figure 5: OS-Level virtualization (Isolation)

- OS level virtualization, còn gọi là containers Virtualization hay Isolation: là phương pháp ảo hóa mới cho phép nhân của hệ điều hành hỗ trợ nhiều instances được cách ly dựa trên một HĐH có sẵn cho nhiều users khác nhau, hay nói cách khác là tạo và chạy được nhiều máy ảo cách ly và an toàn (secure) dùng chung 1 HĐH. Ưu điểm của ảo hóa này là bảo trì nhanh chóng nên được ứng dụng rộng rãi trong các lĩnh vực hosting. OpenVZ, Virtuozzo, Linux-VServer, Solaris Zones, và FreeBSD Jails hỗ trợ loại ảo hóa này [9, 11]. Một lưu ý là loại ảo hóa Isolation này chỉ tồn tại trên HĐH Linux.

Nếu ảo hóa chỉ là công nghệ nền tảng của CC thì việc triển khai CC trong thực tế dựa vào 2 giải pháp cơ bản sau: sử dụng các sản phẩm thương mại cho CC như của VMware, Microsoft (Hyper-V), hoặc các sản phẩm nguồn mở như Eucalyptus và OpenStack. Phần kế sẽ trình bày về lợi ích của hướng tiếp cận triển khai CC dùng nguồn mở.

4. Hướng tiếp cận "Cloud computing" sử dụng công cụ nguồn mở

Với những lợi ích đã nêu của mô hình "Cloud computing" trong phần trước, đặc biệt là về "flexibility" và "cost benefits", đây sẽ là một xu hướng tiếp cận trong tương lai. Tuy nhiên, có rất nhiều công nghệ cho "Điện toán đám mây" với những chi phí và giải pháp khác nhau tùy vào mục đích sử dụng và ưu điểm của mỗi công nghệ như dễ dàng triển khai, khả năng mở rộng cao, giá rẻ, … Sử dụng công cụ mã nguồn mở để triển khai "Cloud computing" đạt được những ưu điểm sau [6]:

- Sự phụ thuộc vào các phần mềm đóng kín và bản quyền (Avoiding vendor lock-in): các giải pháp thương mại thường là 1 bộ giải pháp với các tiêu chuẩn của nhà sản xuất chẳng hạn các APIs đặc trưng, các kiểu định dạng image và lưu trữ riêng, … sẽ làm cho "cloud" không tương thích, hoặc không tận dụng được những cơ sở hạ tầng sẵn có. Hoặc các "đám mây vendor lock-in" trong tương lai sẽ đối mặt với vấn đề di chuyển (migration) một số dịch vụ sang những hệ thống cloud khác, sự khó khăn này là một hạn chế.
- Getting best-of-breed technology: các dự án về "open source cloud computing" luôn luôn được hỗ trợ và giúp đỡ bởi cộng đồng toàn thế giới với hàng ngàn người tham gia phát triển các functions mới và sửa lỗi bugs (fix bugs). Lợi thế này của open source sẽ không thể có được ở bất kỳ một công ty đơn lẻ nào.
- Khả năng mở rộng không hạn chế: chi phí là vấn đề nổi trội trong vấn đề mở rộng mạng "cloud" với giải pháp phần mềm bản quyền. Tuy nhiên với open source clouds, ví dụ mạng clouds sử dụng Ubuntu, hệ điều hành Ubuntu hỗ trợ "cloud computing" hoàn toàn miễn phí nên việc mở rộng rất dễ dàng.
- Aligning the cloud to specific business needs: khi giải pháp thương mại thiếu một chức năng gì đó, sẽ rất khó để tìm ra phương thức thay thế trừ khi chờ một phiên bản mới hơn hỗ trợ. Nhưng với kỹ thuật open source có thể thay đổi code để thêm các chức năng phù hợp cho mục đích kinh doanh của hệ thống.

II. Các giải pháp mã nguồn mở cho mô hình điện toán đám mây

	Eucalyptus	**OpenNebula**	**Nimbus**	**Xen Cloud Platform**	**AbiCloud**	**OpenStack**
Produced by	Santa Barbara university	Eucalyptus System Company European Union	University of Chicago	Citrix XenServer	Abico	Rackspace, NASA, Dell, Citrix, Cisco, Canonical etc.
Main purpose	EC2 Cloud	Build private Cloud	Cloud Computing scientific solution	- Evolution of Citrix XenServer	Cloud management	Offers Cloud Computing services
Users	Enterprise	Researchers on Cloud Computing and Virtualization	Scientific communities	Enterprise	Enterprise	Enterprises, service providers and researchers
Supported OS	Linux (Ubuntu, Fedora, CentOS, OpenSUSE et Debian)	Linux (Ubuntu, RedHat Enterprise Linux, Fedora et SUSE Linux, Enterprise Server)	Most Linux distributions	- Linux (Fedora, RedHat, CentOS et Suse Linux Enterprise Server) - Windows 7	Linux (Ubuntu et CentOS) - Windows XP - Mac OS	- Linux - Windows - Requires x86 Server
Architecture	- Hierarchical - Five components - Minimum two servers	- Centralized - Three components - Minimum two servers	- Centralized - Three components - Minimum two servers	- Centralized - Three components - Minimum two servers	- Centralized - Three components - Minimum two servers	Integration of OpenStack object and OpenStack compute
language	Java, C and python	Java, Ruby and C++	Python, java	Caml	Java, Ruby, C++, and python	Python
Storage	Walrus	- SCP - SQLite3	- GridFTP, Comulus (new version of GridFTP) - SCP	VastSky	HDFS	OpenStack Store
Network	DHCP server on the cluster controller	Manual configuration	DHCP server installed on nodes	Open vSwitch	WSManagement	OpenStack Compute
Access interface	- EC2 WS API - Tools as: HybridFox, ElasticFox	- EC2 WS API - OCCI API	- EC2 WS API - Nimbus WSRF	Command lines « XE » (XenCenter and Versiera (commercial solution for Windows)	Web interface with Adobe Flex	Web interface Web
User	- Zip file that	- Authentication	- X509 certificate	- Authentication	- Authentication	- Certification

	contains certifications - HTTPS connection			- SSH connection	(password stored in MD5 format)	
administrator	- SSH connection - Root required	Root (Only if necessary)	- SSL connection - Integrate Globus (certification)	- SSH connection	- Authentication	- Certification
Load balancing	The cloud controller	Nginx	Le context broker	XAPI	AbiServer	The cloud controller
Fault tolerance	Cluster controller's separation	Database backend (registers virtual machine information)	Periodic verification of cloud nodes	Virtual machine state's synchronization	×	Replication
Live migration	×	Shared FS	×	- Open Virtualization Format - Shared Storage		
VMs location	Node controller	Cluster node	Physical nodes	XCP Host	Clouds nodes	OpenStack Compute
Compatibility with EC2	Yes	Yes	Yes	Yes	No	No
Used by	NASA	Reservoir Project , NUBA	STAR		Active in Span	

1. Eucalyptus

Eucalyptus là một phần mềm nguồn mở Linux-based để triển khai "điện toán đám mây" với cả 2 loại hình private hay hybrid (private and public). Eucalyptus cung cấp IaaS (Infrastructure as a Service) thuận tiện cho việc chỉ định tài nguyên (phần cứng, dung lượng lưu trữ, và hạ tầng mạng) dựa trên yêu cầu sử dụng. Điểm mạnh của Eucalyptus là triển khai enterprise data centers mà không cần quá nhiều yêu cầu về cấu hình phần cứng. Hơn nữa, Eucalyptus hỗ trợ kết nối với dịch vụ đám mây nổi tiếng của Amazon – AWS (Amazone Web Services™) thông qua một giao diện lập trình chung. Kiến trúc của Eucalyptus đơn giản, linh hoạt (flexible), được module hóa (Modular) và đạt được nhiều ưu điểm như chức năng snapshot, self-service, …[12].

2. OpenNebula

OpenNebula là bộ công cụ nguồn mở sử dụng cho private, public, và hybrid cloud. OpenNebula hoạt động tương thích với các giải pháp của Xen, KVM, VMWare, và mới đây là Virtual Box [13, 14].

3. Nimbus

Nimbus là một dự án "điện toán đám mây" của Culumbus để cung cấp dịch vụ IaaS (Infrastructure as a Service). Nimbus hỗ trợ triển khai 2 loại ảo hóa là Xen và KVM [13].

4. Xen Cloud Platform (XCP)

XCP là một platform nguồn mở cho việc triển khai ảo hóa máy chủ và điện toán đám mây trên nền tảng của Xen Hypervisor. XCP hỗ trợ nhiều Guest OS bao gồm windows và linux, hệ thống mạng và lưu trữ cũng như các công cụ quản trị nằm trong XCP appliance. XCP có nguồn gốc từ Citrix XenServer và được chứng nhận bản quyền bởi GNU General Public License (GPL2) [13, 15]

5. AbiCloud

AbiCloud là giải pháp "điện toán đám mây" private được phát triển bởi Abiquo cho phép người dùng có thể xây dựng môi trường IaaS. AbiCloud hỗ trợ các kỹ thuật ảo hóa Virtual Box, VMWare, XEN, và KVM [13, 16].

6. OpenStack

OpenStack là 1 dự án mở cộng đồng cho việc phát triển "điện toán đám mây" phù hợp với các nhà cung cấp (Cloud Providers) cũng như người dùng (Cloud Customers) được phát triển bởi Rackspace hosting và Nasa. OpenStack bao gồm 3 dự án chính: OpenStack Compute (để triển khai việc quản lý và chỉ định tài nguyên cho các instances ảo), OpenStack Object Storage (thực thi việc lưu trữ, backup), và OpenStack Image Service (đảm nhận việc phát hiện, đăng ký, truyền tải dịch vụ cho các images disk ảo) [13].

Hiện nay OpenStack đang được đánh giá là phần mềm nguồn mở xây dựng CC mạnh nhất hiện nay với sự hỗ trợ của các hãng mãy tính lớn trên thế giới như HP, Canonical, IBM, Cisco, Microsoft, …Đây cũng là bộ công cụ quan trọng đang được triển khai và sẽ được trình bày chi tiết trong các phần tiếp theo.

Phần 2: OpenStack

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

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

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

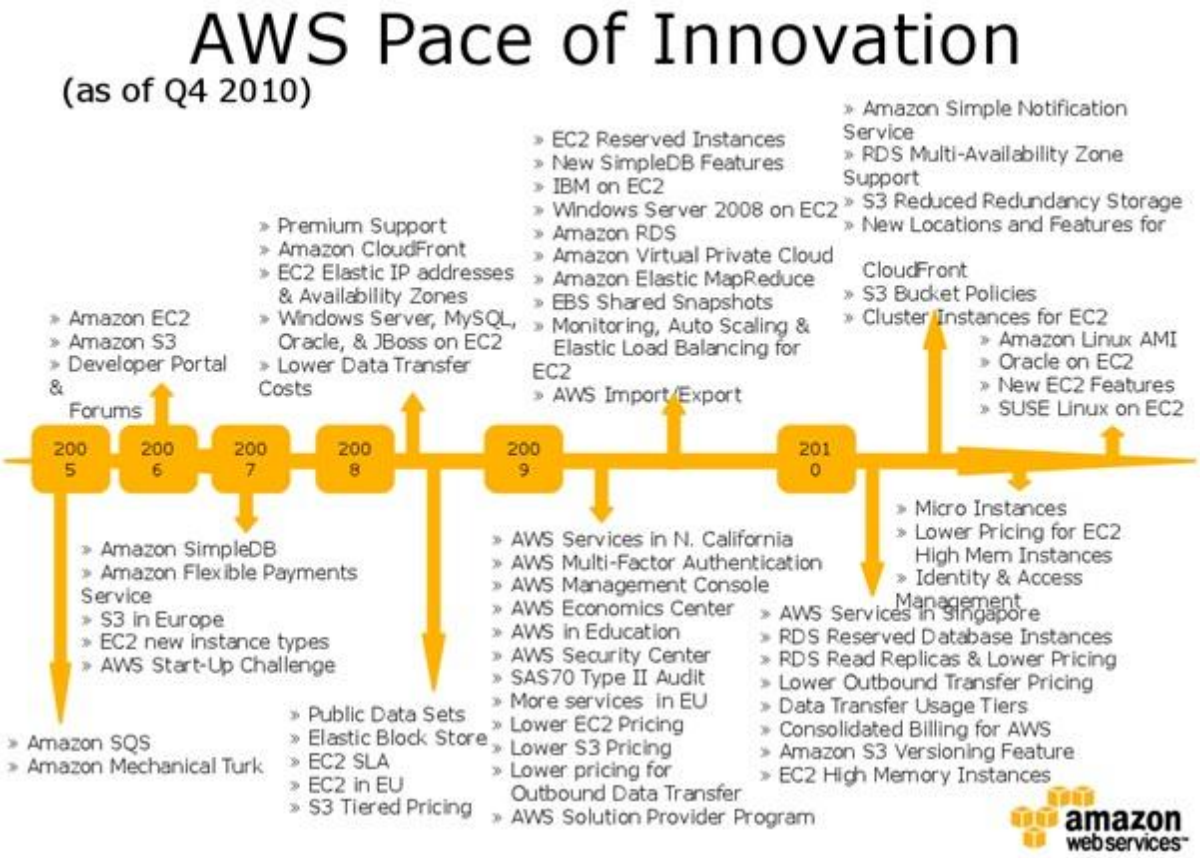
Saleforce đã bắt đầu từ rất sớm với CC, ngay từ năm 1999 hãng đã có định hướng phát triển về SaaS, từ việc cung cấp các dịch vụ quản lý khách hàng, kế toán, thống kê tài chính... Theo như báo cáo kinh doanh năm 2011, mảng dịch vụ về SaaS đã đem lại cho Saleforce hơn 3 tỉ USD đó là một con số đáng ngưỡng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

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

Figure 6: Management console AWS

Amazon thực sự đã xây dựng được một đế chế công nghệ hùng mạnh, cạnh tranh trực tiếp với các nhà cung cấp dịch vụ hosting truyền thống cũng như CC như Rackspace, GoDaddy, Google... Theo nhận định của giới chuyên môn Amazon đã tạo ra một kiến trúc về CC kinh điển AWS với đầy đủ các dịch vụ về tính toán, lưu trữ, cơ sở dữ liệu chuyên dụng... Thực tế cho thấy hầu hết các nền tảng khác như Eucalyptus, Openstack... đều được xây dựng theo một kiến trúc, các thành phần tựa như AWS. Tất nhiên chưa có một khẳng định nền tảng của ai tốt hơn một cách rõ ràng, nhưng với những đánh giá về tính ổn định, hiệu năng và quan trọng nhất là giá của dịch vụ. AWS vẫn đang là sản phẩm tốt nhất hiện nay.

Chúng ta sẽ lướt qua một số dịch vụ chính của AWS. Như trong hình dưới đây là cửa sổ quản lý dịch vụ của AWS.



AWS vẫn đang liên tục nghiên cứu cải thiện và bổ sung những tính năng mới cho tập các dịch vụ của họ. Do khuôn khổ của việc nghiên cứu thử nghiệm Openstack nên nhóm sẽ chỉ đưa ra một số giới thiệu cơ bản và nhận xét về các dịch vụ chính của Amazon. Từ đây sẽ có một cái nhìn trực quan hơn về Openstack và có một so sánh với 'đối thủ' lớn nhất của nó. Sau đây là một số mốc thời gian quan trọng của AWS:

Những dịch vụ chính của AWS phải kể đến là:

- Amazon Elastic Cloud Compute (EC2) cung cấp các instance (máy ảo) tùy theo nhu cầu, với khả năng tính toán, mở rộng vô cùng linh hoạt. Hiểu đơn giản, EC2 cung cấp cho người dùng khả năng tạo các máy ảo trên hạ tầng của Amazon, họ có thể cấp phát

tài nguyên (CPU, RAM) theo yêu cầu, và từ đó Amazon sẽ tính toán các chi phí. Các instance có các mức cấu hình khác nhau: nhỏ nhất là mirco instance (1 CPU, 613 MB RAM) và lớn nhất tới hơn 64GB RAM và 88 EC2 CPU (tương đương 2 x Intel Xeon E5-2670)

- Amazon Elastic Block (EBS) cung cấp khả năng lưu trữ độc lập, kết hợp với EC2. Hiểu đơn giản giống như việc sử dụng thêm các ổ đĩa mở rộng trên các máy vật lý. Khi mà có sự cố tại instance thì dữ liệu lưu trên EBS vẫn có thể sử dụng độc lập, và có thể chia sẻ giữa những instace khác nhau.

- Amazon Simple Storage Service (S3) cung cấp khả năng lưu trữ không hạn chế, cũng giống như EBS, S3 giải quyết vấn đề về lưu trữ, tuy nhiên EBS được sử dụng bởi các instance thì S3 được sử dụng như một ổ đĩa mạng. Thông qua một giao diện (web hay một GUI) người dùng có thể lưu trữ dữ liệu của mình, backup dữ liệu từ các nguồn khác nhau (từ chính EBS, EC2...) S3 sử dụng cơ sở dữ liệu Dynamo để quản lý việc lưu trữ, chứ không sử dụng các CSDL quan hệ truyền thống vì đối với dịch vụ lưu trữ, người dùng chủ yếu đọc và ghi dữ liệu nên nếu lưu theo mô hình quan hệ sẽ không giải quyết hiệu quả.

Vì các thành phần trong AWS hoạt động độc lập với nhau, để chúng có thể kết hợp lại cần có một phần trung gian giúp truyền các thông điệp và đồng bộ thời gian giữa các dịch vụ. Amazon đã phát triển riêng một dịch vụ tên Simple Queue Service - đây chính là thành phần đầu tiên mà Amazon phát triển, và phải mất tới 2 năm (2002-2004) mới cơ bản hoàn thiện. Tuy có vẻ không mấy quan trọng nhưng đây lại chính là một điểm mấu chốt giúp tạo nên sức mạnh của hệ thống các dịch vụ AWS.

Ngoài ra thì AWS đang cung cấp rất nhiều dịch vụ khác nữa như SimpleDB (lưu trữ truy vấn theo kiểu quan hệ truyền thống), Elastic MapReduce Service (áp dụng trong việc tính toán hiệu năng cao, xử lý dữ liệu lớn, thông qua S3 và EC2)...

Tùy theo lưu lượng sử dụng, tài nguyên hệ thống bạn cần...Amazon sẽ tính toán chi phí và yêu cầu bạn thanh toán. Về cơ bản bạn chỉ phải trả cho những gì bạn sử dụng. Khi bạn không cần dùng đến tài nguyên nào đó, bạn có thể 'dừng' nó lại và không phải trả phí trong thời gian đó. Đây chính là một trong những điểm thú vị có thể thấy với CC.

Amazon hiện cho phép người sử dụng thử nghiệm các dịch vụ cơ bản (ở quy mô nhỏ nhất) miễn phí trong một năm đầu tiên. Để đăng ký rất đơn giản, bạn cần khai báo tài khoản ngân hàng của mình, sẽ không mất một khoản phí nào nếu chú ý đọc điều khoản từ Amazon. Ví dụ khi sử dụng EC2 nếu bạn 'lỡ tay' chọn instance không phải loại micro, vậy là bạn đã mất phí rồi đấy. Nhóm sẽ demo một số chức năng chính của AWS trong buổi giới thiệu, bạn cũng có thể xem trong phần phụ lục.

Người dùng có thể tương tác với AWS thông qua AWS Management Console bằng cách đăng nhập với username và mật khẩu, sau đó với một giao diện Web người dùng có thể sử dụng các chức năng của AWS. Với từng dịch vụ cụ thể như EC2, S3... AWS sẽ cung cấp cho

người dùng các chứng chỉ, public/private key để chứng thực với hệ thống, sau đó người dùng có thể tương tác thông qua môi trường dòng lệnh (trong Linux sử dụng gói ec2tools).

AWS hỗ trợ một số ngôn ngữ lập trình cơ bản như Java, PHP, Ruby, .NET, Python... thông qua các API. Các lập trình viên có thể sử dụng những API này để tương tác, lập lịch, tự động khởi tạo mở rộng...với các dịch vụ của AWS. Theo đánh giá từ cộng đồng thì AWS API hoạt động rất tốt trên các nền tàng khác nhau. Ngôn ngữ được AWS khuyến cáo sử dụng là Python, Java.

II. Giới thiệu về OpenStack Projects

1. Lịch sử về Openstack

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

Trở lại mốc 2005 khi mà Amazon ra mắt thử nghiệm EC2, đó là một thành công lớn gây bất ngờ cho cộng đồng. Với sự ổn định của nó, các công ty khác có thể đơn giản "thuê" EC2 trong một vài giờ với một mức năng lực rất rất lớn để thực hiện các công việc tính toán cần tới hiệu năng cao của họ. Ví dụ mà Amazon thường đem ra so sánh là việc hợp tác giữa họ và NASDAQ - sàn chứng khoán cần xử lý một lượng dữ liệu tính toán cực lớn vào cuối tuần, thay vì đầu tư một hệ thống máy chủ phức tạp, họ chỉ thuê EC2 trong vài giờ và chi phí tiết kiệm rất rất nhiều hơn nữa hiệu quả công việc lại tốt hơn.

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

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

Như đã nói AWS chính là nguồn cảm hứng tạo nên Openstack ngày nay, AWS là nền tảng đóng của Amazon và Openstack là một nền tảng mở dành cho tất cả các công ty và cộng

đồng sử dụng. Mục đích của Openstack là cung cấp cho người dùng khả năng xây dựng một hạ tầng cho cả private cloud và public cloud. Đã có nhiều công ty sử dụng Openstack để xây dựng dịch vụ để phục vụ nhu cầu của chính họ và cho thuê như chính NASA và Rackspace.

2. Tổng quan về Openstack

Openstack có chu kỳ phát triển 6 tháng, đi cùng với sự phát triển của CC, với mỗi phiên bản Openstack lại bổ sung thêm thành phần mới tương ứng với những chức năng mới. Openstack hoàn toàn là nguồn mở, các thành phần của nó được viết trên Python - ngôn ngữ đang được đánh giá rất cao những năm gần đây.

2.1. Các phiên bản của OpenStack

Austin – 10/2010: là phiên bản đầu tiên của OpenStack bao gồm 2 projects là Object storage (còn gọi là Swift) và Compute (còn gọi là Nova). Project Compute trong phiên bản này chỉ ở mức độ testing và hạn chế nhiều tính năng khi triển khai.

Bexar – 2/2011: tích hợp 1 project mới là Image Service, đồng thời có nhiều sự thay đổi cải tiến trong Nova và Swift. Phiên bản này cho phép lưu trữ files lớn hơn 5Gb và tích hợp một service mới "swauth" cho việc chứng thực, thẩm quyền. Đồng thời cải tiến nhiều tính năng trong API cũng như mở rộng việc hỗ trợ các hypervisors cho ảo hóa.

Cactus – 4/2011: phiên bản này cũng bao gồm 3 projects như Bexar, tuy nhiên có sự cải tiến API và hỗ trợ thêm 2 công nghệ ảo hóa LXC containers và VMware. Glance giới thiệu công cụ command-line mới phục vụ việc truy cập dịch vụ, thêm các định dạng image, và thẩm định image đảm bảo toàn vẹn dữ liệu (integrity).

Diablo – 11/2011: đây là phiên bản đang được sử dụng thử nghiệm, cũng có 3 projects chính như phiên bản Cactus.

Essex – 4/2012: phiên bản mới vừa ra đời – sẽ thử nghiệm trong thời gian tới – với sự hỗ trợ và nâng cấp 2 projects mới là Identity và Dashboard.

2.2. OpenStack Diablo

Kiến trúc conceptual và logical

Sau đây là sơ đồ kiến trúc ở mức conceptual của Openstack:

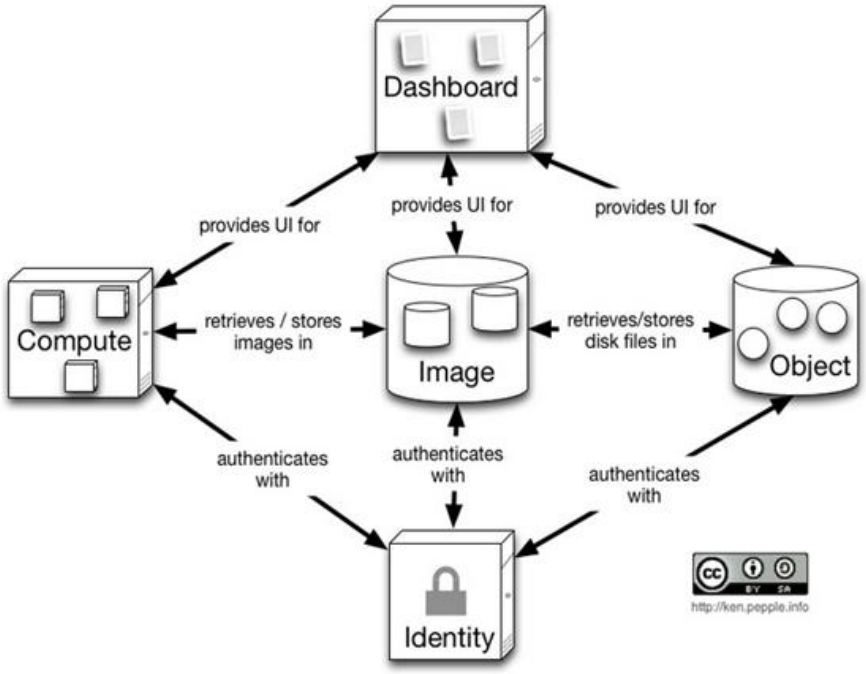


Figure 7: Kiến trúc Logic OpenStack (conceptual)

Trong thử nghiệm, nhóm sử dụng bản Openstack ra mắt ngày 22/11/2011 mã Diablo. Trong phiên bản này gồm ba thành phần chính:

- *Compute (tên mã Nova) cung cấp khả năng tính toán với những instance - tương ứng với EC2 của Amazon.*
- *Image Service (tên mã Glance) lưu trữ các file ảnh của các instance trước khi được 'bung' ra sử dụng bởi Nova - AWS cũng có một thành phần tương tự để quản lý các image tuy nhiên vì là nền tảng đóng, nên thông tin chi tiết về nó không được công bố rõ ràng.*
- *Object Storage (tên mã Swift) cung cấp khả năng lưu trữ - tương ứng với S3.*

Phiên bản mới nhất của Openstack ra mắt ngày 05/04/2012 với codename Essex, bổ sung thêm hai thành phần mới là:

- *Dashboard (tên mã Horizon) cung cấp giao diện web để quản lý Openstack.*
- *Identity (tên mã Keystone) cung cấp khả năng authentication và authorization cho các dịch vụ của Openstack.*

Ở mức kiến trúc logical, OpenStack được minh họa sau đây:

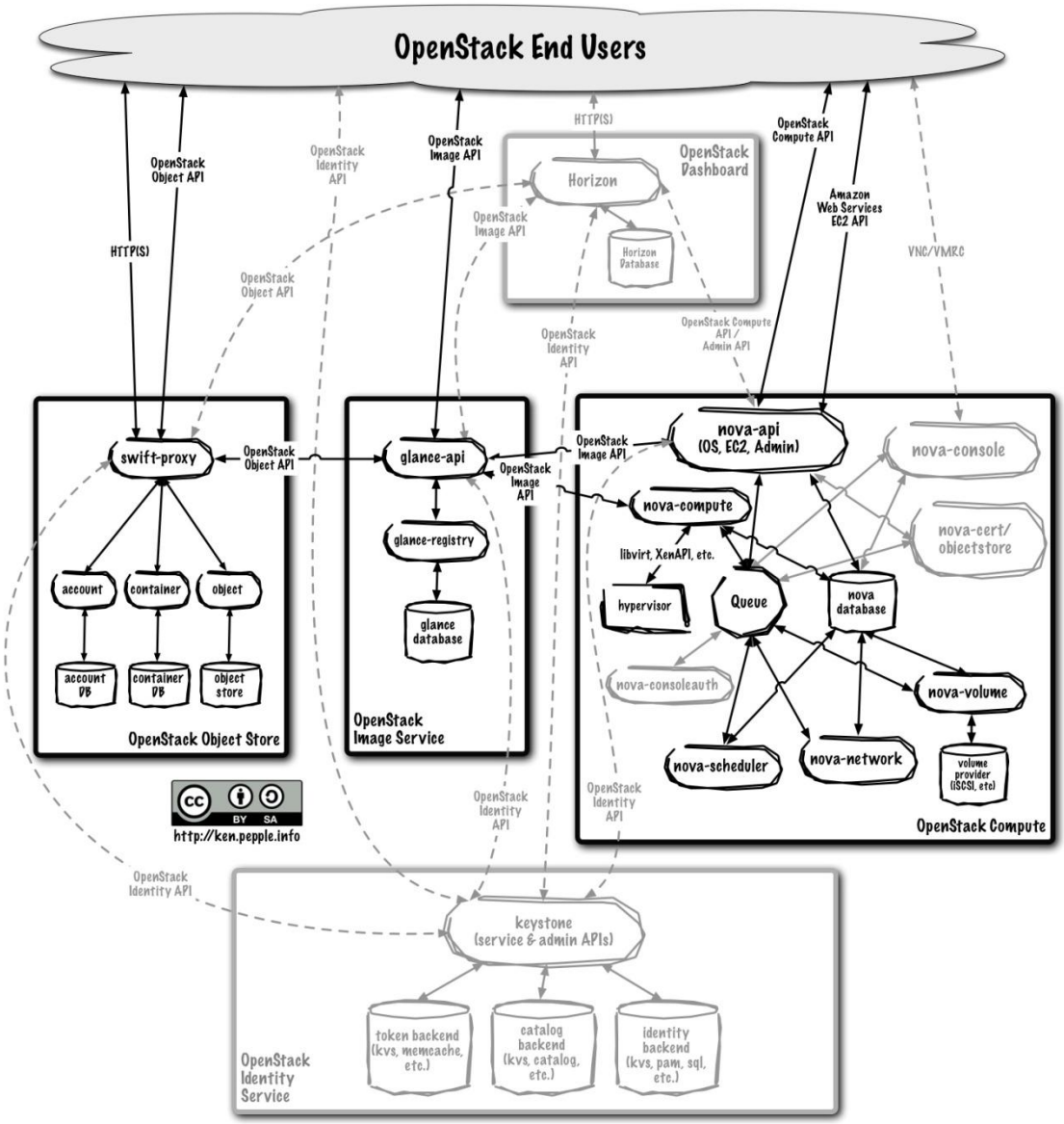


Figure 8: Logical Architecture

Mô hình kiến trúc logic của OpenStack được diễn giải qua 3 ý chính sau đây:

- Người dùng cuối tương tác thông qua 1 giao diện web (Horizon)
- Tất cả các services đều được chứng thực thông qua Keystone
- Các dịch vụ cá nhân riêng biệt tương tác với nhau thông qua các APIs tương ứng.

Cũng giống như AWS, các thành phần của Openstack hoạt động độc lập, do vậy cần phải có một phần trung gian ở giữa nhằm trung chuyển, đồng bộ thời gian, thông tin về tài nguyên cho cả hệ thống. Openstack hiện sử dụng Rabbit queue message để chuyển các thông điệp qua lại.

Trong phiên bản Diablo thử nghiệm hai thành phần Dashboard và Indentity chưa hoạt động tốt với 3 thành phần Nova, Swift, Glance nên hiện nay vẫn chưa thể cài đặt chúng hoạt động đúng.

Sau đây xin giới thiệu chi tiết hơn về các thành phần chính của Openstack.

2.1.1. OpenStack compute

Đây là phần cơ bản nhất của Openstack có chức năng điều khiển IaaS và phân phối lại tài nguyên hệ thống cho các instance với khả năng tính toán lưu trữ độc lập. Nó tương ứng với Amazon EC2.

Về cơ bản Nova cung cấp cho người dùng khả năng chạy các instance (máy ảo) và giao diện để quản lý các instance đó trên hạ tầng phần cứng. Tuy nhiên Nova không bao gồm bất cứ phần mềm ảo hóa nào. Cái nó làm là sử dụng lại các hypervisor (do người dùng tùy chọn cài đặt) để thực hiện việc ảo hóa tính toán. Người dùng có thể sử dụng các hypervisor khác nhau trong các zone khác nhau. Dưới đây là các hypervisor mà Nova hiện hỗ trợ:

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

Các tính năng chính của OpenStack Compute [17]

- ***Quản lý tài nguyên ảo hóa bao gồm CPU, memory, disks, network interfaces.*** *Tất cả các tài nguyên được hợp nhất vào trong 1 "bể" – "pool of computing". Việc này sẽ tăng tính tự động và tận dụng tài nguyên, đem lại lợi ích lớn về kinh tế.*
- ***Quản lý mạng nội bộ (LAN)*** *Flat, Flat DHCP, VLAN DHCP, IPv6*
  OpenStack được lập trình để chỉ định các địa chỉ IPs và VLAN (Virtual LAN). Chức năng này giúp cho việc cung cấp dịch vụ networking và nâng tính bảo mật khi các VLANs được tách rời nhau. Đồng thời tính linh hoạt trong mô hình mạng cũng phù hợp với mỗi ứng dụng cho mỗi user/group.
- ***API với nhiều tính năng và xác thực:*** *Được thiết kế tự động và an toàn để quản lý việc users truy cập vào các tài nguyên và ngăn chặn truy cập trái phép qua lại giữa các users.*
- ***Distributed and asynchronous architecture***
  Massively scalable and highly available system
  (for increased assurance of system uptime)
- ***Virtual Machine (VM) image management***
- ***Live VM management (Instance)*** *khởi tạo, khởi động, đóng băng, hay xóa instances. Ngoài ra còn có tính năng lifecycle management.*
- ***Floating IP addresses:***
- ***Security Groups***
- ***Role Based Access Control (RBAC)***
- ***Projects & Quotas***
- ***VNC Proxy through web browser***
- ***Advanced Scheduler*** *(Diablo v3 07/28 – Started)*

Nova có 7 thành phần chính:

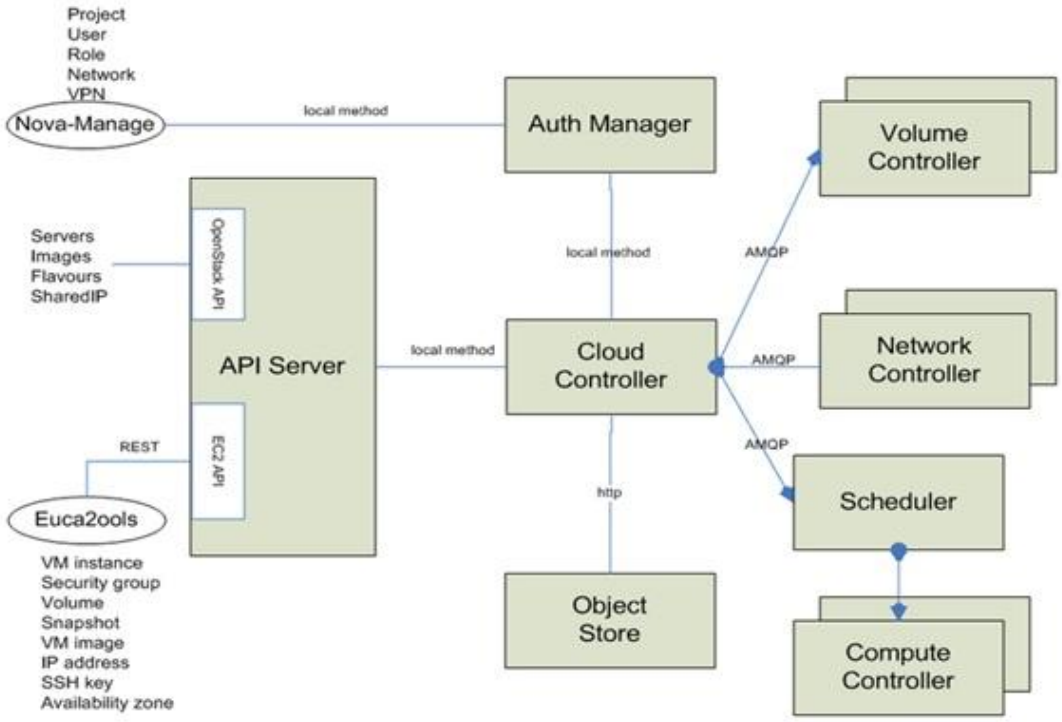


Figure 9: Các thành phần của Nova

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

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

Do được tạo thành từ nhiều thành phần khác nhau nên có một số chức năng đang được xây dựng lại, một số chức năng "bị lặp". Điển hình như trong Nova, thành phần Object Store dùng để lưu các image (file ảnh của các hệ điều hành ảo khi chưa được chạy), đồng thời Glance cũng là nơi để lưu trữ các image đó. Tuy nhiên việc này không ảnh hưởng gì nhiều

đến hệ thống. Người dùng có thể tùy chọn giữa các lựa chọn này. Theo khuyến cáo thì Glance vẫn được ưu tiên hơn.

User và Project

Nova được thiết kế để sử dụng cho nhiều đối tượng khác nhau, nó sử dụng các quy tắc phân quyền cơ bản thông qua Role-Based Access Control (RBAC) bao gồm 5 luật:

- ***Cloud Aministrator (admin):*** *Global role. User với quyền này có toàn quyền với cả hệ thống.*
- ***IT Security (itsec):*** *Global role. Quyền này hạn chế hơn so với admin. Nó cho phép user giữ và cách ly các instance trong bất cứ project nào nếu có vấn đề.*
- ***Project Manager (projectmanager)****: Project role. Người sở hữu một project nào đó, người có quyền này có thể thêm user vào project, tương tác với các image, chạy và kết thúc (terminate) các instance trong vùng project quản lý.*
- ***Network Administrator (netadmin)****: Project role. Định vị (allocate) và gán public IP cho instance. Thay đổi các luật của firewall.*
- ***Developer (developer):*** *Project role. Đây là quyền mặc định được gán cho người dùng.*

Nova-network

Thành phần này tương tác với nova-compute, có nhiệm vụ kết nối giữa các instance với nhau và các instance với public network. Cũng giống như AWS hay Eucalyptus một instance trong Openstack có thể có 2 IP. Một private IP được dùng để kết nối giữa các instance và public IP được dùng để kết nối instance với Internet (public network).

nova-network có ba cách quản lý khác nhau:

- Flat Network: tạo một giao diện bridge dựa trên ethernet adapter để giao tiếp giữa các node. Khi chọn cấu hình là Flat Network, Nova sẽ không quản lý các thao tác về networking của các instance. Đơn giản lúc đó IP sẽ được gán cho các instance thông qua file system. Các metadata phải được cấu hình thủ công trên các gateway nếu là yêu cầu của mạng nội bộ. Hình sau đây mô tả về cách cấu hình này trên nhiều node khác nhau thông qua một ethernet adapter:

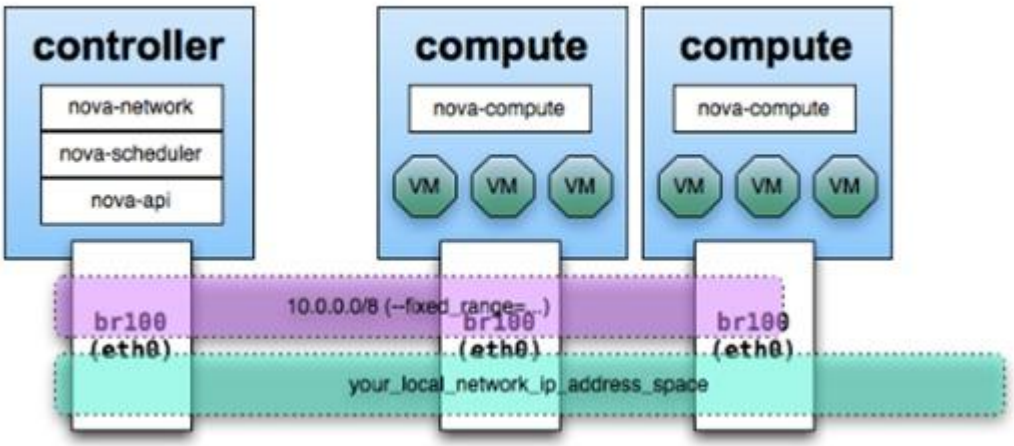


Figure 10: Ví dụ Flat Network

- Flat DHCP Networking: với kiểu cấu hình này thì host chạy nova-network sẽ đóng vai trò như một gateway cho các virtual node.

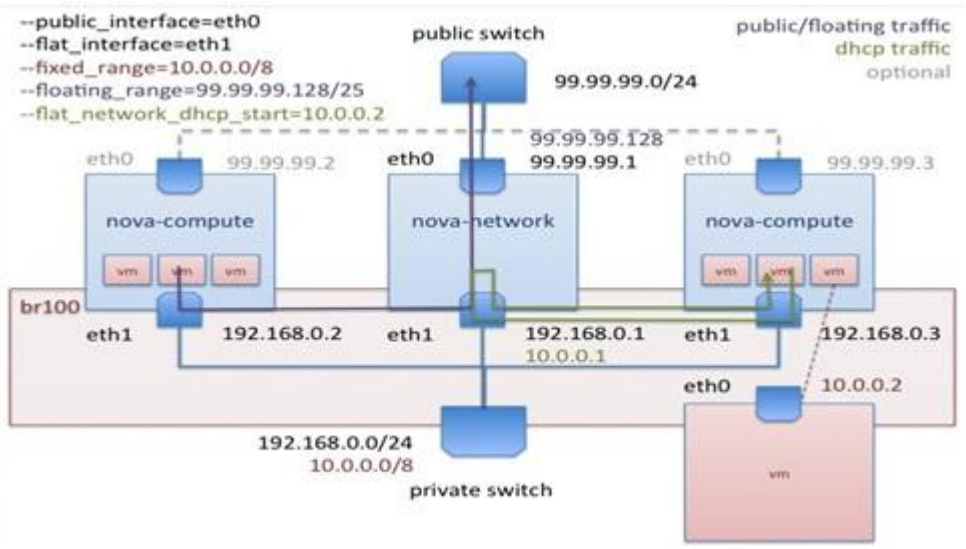


Figure 11: Flat DHCP networking

- VLAN Networking: là cấu hình mặc định của nova. Nó cho phép admin gán các vùng private network cho mỗi project. Và instance có thể được truy cập thông qua VPN từ ngoài Internet. Trong kiểu cấu hình này, mỗi project sẽ có một VLAN riêng, một Linux networking bridge và subnet. Subnet được chỉ định bởi admin và được gán động cho project khi được yêu cầu. Một DHCP server được chạy quản lý cho mỗi VLAN để gán IP cho mỗi instance trong vùng subnet được gán cho project. Tất cả các instance thuộc cùng project được đặt trong một VLAN riêng.

2.1.2. OpenStack Object Storage

OpenStack Object Storage hay còn gọi là Swift được Rackspace open-source từ năm 2010, nó chính là công nghệ được sử dụng đằng sau Rackspace's Cloud Files một trong những giải pháp lưu trữ thương mại rất tốt hiện nay cạnh tranh với Amazon S3.

Swift là phần mềm nguồn mở để tạo ra các phiên bản giống nhau cho việc lưu trữ dữ liệu, đồng thời với việc mở rộng lưu trữ rất linh hoạt và sử dụng cơ chế clusters, khả năng của swift có thể lưu trữ lên đến petabytes dữ liệu truy cập. Swift không chỉ là một hệ thống data thời gian thực, nó còn là một hệ thống lưu trữ lớn với tính chất "lâu dài – long term" với một lượng dữ liệu cực lớn mà vẫn đảm bảo việc truy xuất, phân cấp, và nâng cấp (retrieved, leveraged, and updated). Các đối tượng lưu trữ (Object Storage) sử dụng kiến trúc phân tán so với mô hình tập trung, nên sẽ không có điểm trung tâm. Việc này giúp cho nâng khả năng mở rộng, backup, và duy trì (scalability, redundancy and permanence). Các đối tượng được ghi lên nhiều thiết bị phần cứng khác nhau mà trong đó, OpenStack đóng vai trò chịu trách nhiệm đảm bảo việc tái tạo, sao nguyên, và toàn vẹn của dữ liệu qua các cluster. Mặt khác, các cụm lưu trữ dữ liệu có thể được mở rộng theo "chiều ngang" dễ dàng qua việc thêm các nodes lưu trữ mới. Nếu 1 nodes trục trặc, hoạt động của OpenStack ngay lập tức tái tạo lại nội dung của nó từ một nodes đã được active khác. Tất cả các công việc trên được OpenStack thực hiện về mặt logic mà không phụ thuộc vào bất kỳ thiết bị phần cứng nào, việc này đảm bảo chắc chắn hơn trong việc tái tạo, sao chép dữ liệu đồng thời tránh việc phụ thuộc vào thiết bị phần cứng, đặc biệt các thiết bị chuyên dụng giá thành cao.

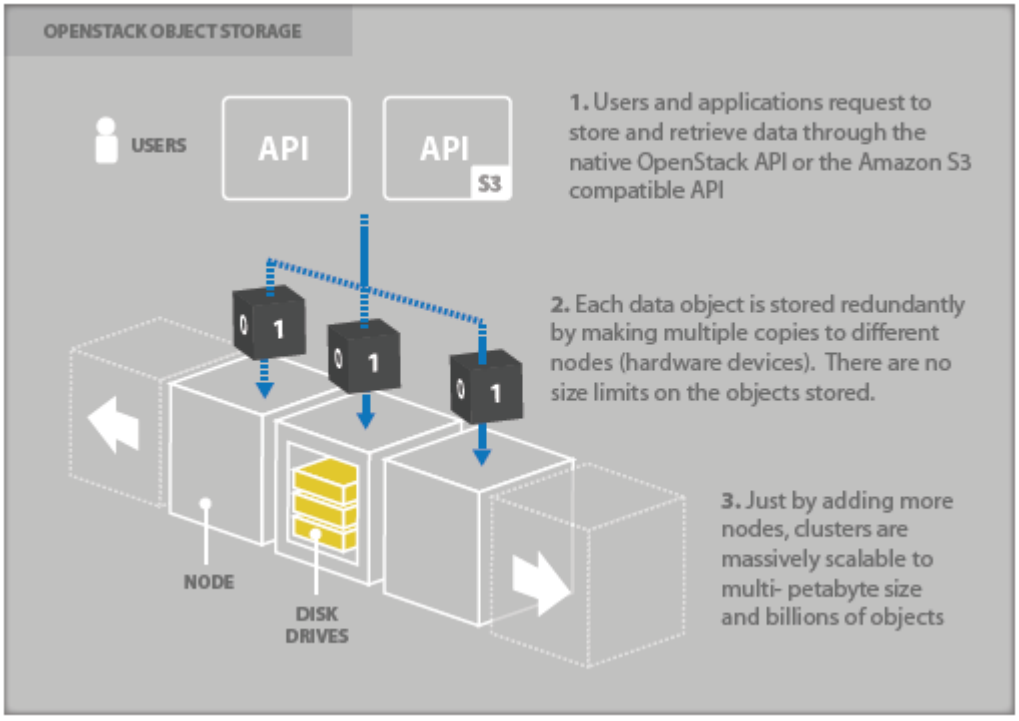


Figure 12: Tổng quan OpenStack Object Storage

Các tính năng của OpenStack Object Storage [18]

- ***Store and Manage files programmatically via API:*** *quản lý file thông qua giao diện API*
- ***Create Public or Private containers***
- ***Leverages Commodity hardware***
- ***HDD/node failure agnostic:*** *đảm bảo không mất dữ liệu bằng các cơ chế backup và sao lưu tự động*
- ***Unlimited Storage:*** *lưu trữ không hạn chế*
- ***Multi-dimensional scalability (scale out architecture)***

- ***Account/Container/Object structure:*** *cho phép mở rộng đến nhiều Peta-bytes, và hàng tỷ objects*
- ***Built-in Replication:*** *N copies các accounts, container, và objects*
- ***Easily add capacity unlike RAID resize***
- ***No central database:*** *hiệu suất cao, tránh được thắt cổ chai*
- ***RAID not required***
- ***Built-in Mgmt. utilities:*** *Acct. Management: Create, add, verify, delete users*
  Container Management: upload, download, verify
  Monitoring: Capacity, Host, Network, Log trawling, cluster health
- ***Drive auditing:*** *cho phép kiểm tra các ổ đĩa để phát hiện hư hỏng.*
- ***VNC Proxy through web browser***

Hình dưới đây mô tả kiến trúc logic của Swift:

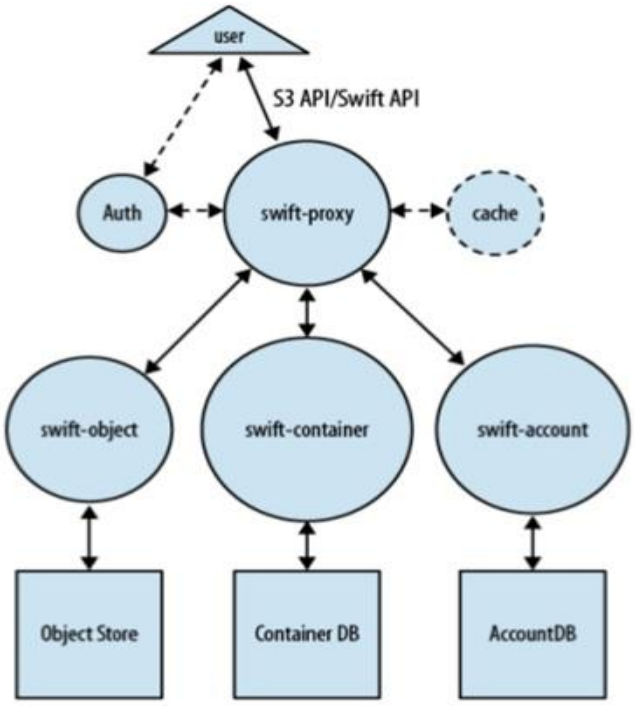


Figure 13: Kiến trúc Logic của Swift

Các thành phần chính được miêu tả cụ thể như sau:

- Proxy Server - nhận các request và chứng thực user. Sau khi quá trình chứng thực hoàn tất, dữ liệu sẽ được chuyển trực tiếp từ (hoặc tới) user. Proxy server sẽ không kiểm tra chúng.

- Object Server - lưu trữ, quản lý các đối tượng được lưu. Các object sẽ được lưu theo dạng binary cùng với metadata miêu tả về dữ liệu đó.

- Container Server - lưu trữ thông tin và trả về danh sách các object đang được lưu bên Object Store.Nó không biết chính xác object được lưu ở đâu nhưng nó biết cụ thể object được lưu tại container nào. Dữ liệu được lưu mặc định trong một CSDL Sqlite,

nếu Swift được cài đặt trên các cluster khác nhau thì CSDL này sẽ được tạo thêm các bản sao tương tự.

- Account Server - cũng giống như Container Server nhưng nhiệm vụ của nó là quản lý danh sách các Container chứ không phải là object.

- The Ring - Thành phần này sẽ tạo một ánh xạ giữa tên của các thực thể được lưu trên đĩa cứng và địa chỉ vật lý của nó. Có nhiều ring khác nhau cho account, container và object. Khi mà các thành phần khác cần sử dụng bất cứ thao tác nào trê object, container hay account thì cần phải tương tác với ring tương ứng để tìm ra đúng địa chỉ lưu trữ trên cluster. Ring được sử dụng bởi proxy server và các tiến trình khác chạy trong background.

2.1.3. OpenStack Image Service

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

Các tính năng hiện tại [19]:

- ***Image-as-a-service***
- ***Multi-format/container support***
- ***Image status***
- ***Scalable API***
- ***Metadata***
- ***Image Checksum***
- ***Extensive Logging***
- ***Integrated testing***
- ***Back-end store options***
- ***Version control***
- ***CLI access***
- ***Built-in Mgmt. utilities***
- ***Drive auditing***
- ***VNC Proxy through web browser***

Như đã giới thiệu Glance là một trong những thành phần chính của Openstack, nhiệm vụ của nó là lưu và cung cấp các file ảnh của các máy ảo (instance).

Glance gồm có ba phần:

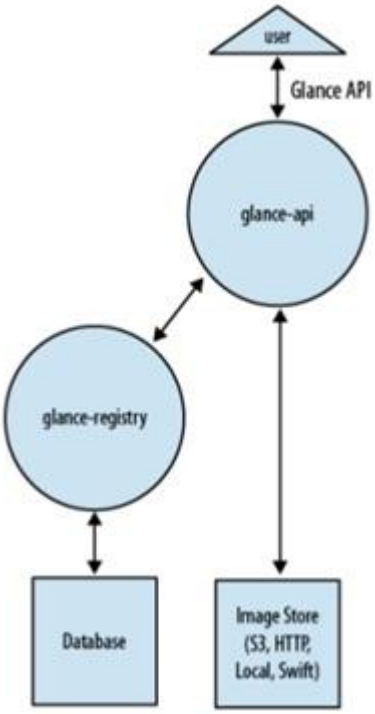


Figure 14: Các thành phần của Glance

- Glance API server - nhận các hàm gọi API, tương tự như nova-api, nó chờ các API request sau đó giao tiếp với các thành phần khác (glance-registry và image store) sau đó thực hiện các công việc được yêu cầu: truy vấn, upload, delete image...
- Glance Registry server - lưu và cung cấp các thông tin (metadata) về image (định dạng, ID, dung lượng...) Mặc định sử dụng Sqlite để lưu các metadata. Ngoài ra glance-registry luôn nghe cổng 9191.
- Image Storage - lưu trữ các file image

Glance hỗ trợ một số định dạng sau:

Disk Format	Notes
Raw	Unstructured disk format
VHD	Most common format supported by most OpenStack virtualization technologies except KVM.
VMDK	Format popularized by VMware.
qcow2	QEMU image format, native format for KVM and QEMU. Supports advanced functions.
VDI	Virtual disk image format originated by Oracle VM VirtualBox.
ISO	Archive format for optical disks.
AMI, ARI, AKI	Amazon machine, ramdisk, and kernel images (respectively). See more information at the Amazon EC2 User Guide.

Figure 15: Định dạng Glance

Để mô tả chức năng của Glance, đơn giản ta có thể miêu tả bằng sơ đồ hoạt động như sau:

Figure 16: Hoạt động của Glance

Trong phần thử nghiệm nhóm cũng sử dụng ba thành phần Nova, Glance và Swift. Về cơ bản các file image của instance sẽ được upload lên Glance server, sau đó Nova sẽ gọi tới Glance và yêu cầu lấy một trong những file image đó để khởi tạo instance bên trong nova-compute. Nếu có dữ liệu cần lưu riêng (backup, dữ liệu dùng chung giữa các instance) thì sẽ được lưu trên Swift. Ba thành phần này độc lập với nhau, nhưng có thê kết hợp với nhau để hoạt động như một thể thống nhất.

2.1.4. OpenStack Dashboard (Horizon) OpenStack Identity

Trong lần thử nghiệm này vẫn chưa hoàn thiện được việc cài đặt hai thành phần này cùng với Nova, Glance, Swift. Trong phiên bản Essex hy vọng hai thành phần này sẽ hoạt động tốt hơn. Sau đây là một số thông tin cơ bản về Keystone và Dashboard.

Keystone là thành phần để chứng thực, token, catalog và policy service cho tất cả các dịch vụ khác của Openstack. Nó được triển khai thông qua Identity API của Openstack.

Dashboard cung cấp một giao diện web nhằm tương tác quản lý các thành phần còn lại của Openstack, nó kết hợp với Keystone để chứng thực user. Được phát triển dựa trên Django framework. Nó cung cấp một giao diện tương tự như AWS management console.

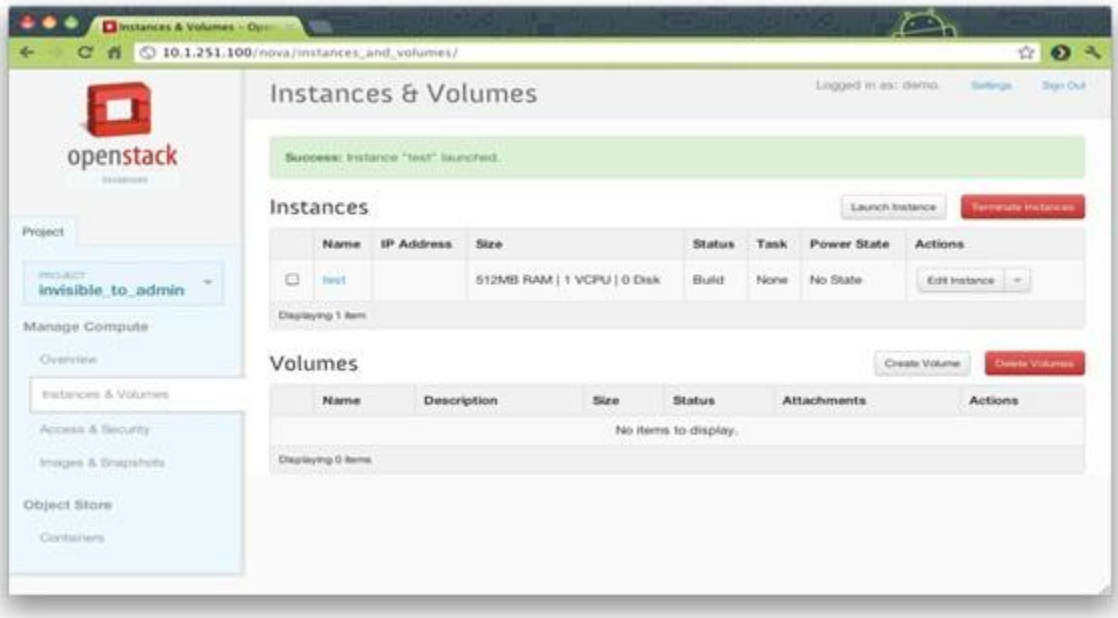


Figure 17: OpenStack DashBoard

Thông qua Dashboard chúng ta có thể thực hiện hầu hết các thao tác đối với các thành phần của Openstack.

II. Mô hình triển khai OpenStack

1. Các công cụ sử dụng

Mọi triển khai CC được thực hiện trên một server DELL T710

- Ubuntu 11.10 server amd64
- Các thành phần của Openstack được cài từ repository của Ubuntu, tương ứng với phiên bản Diablo của Openstack.
- Tất cả các thành phần Nova, Glance, Swift được cài đặt trên một server duy nhất. Do vậy một số thành phần phụ như đồng bộ thời gian giữa các node là không cần thiết (không cần sử dụng ntp server)

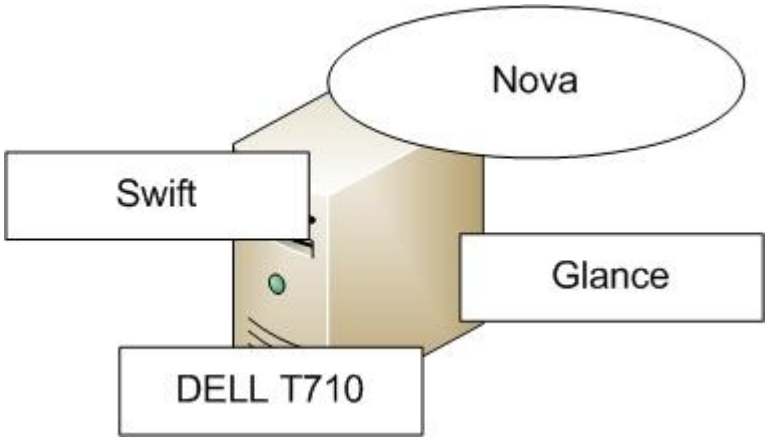


Figure 18: Mô hình triển khai

Thêm nữa chỉ cần sử dụng một adapter duy nhất cho việc cấu hình nova-network. Các instance sẽ được gán hai dải IP như sau:

- Public IP dùng để kết nối các instance ra Internet: 172.17.2.64/27
- Private IP dùng để kết nối các instnace với nhau (mặc định lúc khởi tạo sẽ gán cho mỗi instance một địa chỉ): 10.0.0.0/22 32 32

Các gói chính sẽ được cài đặt:

- Các gói phụ như unzip để giải nén các image, vnc và rất nhiều gói liên quan tới Python: python-software-properties memcached xfsprogs python-setuptools curl vncproxy unzip
- Chúng ta sẽ sử dụng MySql server cho tất cả các dịch vụ: mysql-server
- Message queue server nhằm chuyển các thông điệp giữa các thành phần của Openstack: rabbitmq-server

- Bộ công cụ dùng để tương tác với Openstack thông qua dòng lệnh, ban đầu nó được thiết kế cho Eucalyptus nên mới có tiền tố euca: euca2ools
- Các gói liên quan tới Nova: nova-volume nova-api nova-nova-ajax-console-proxy nova-doc nova-scheduler nova-objectstore nova-network nova-compute
- Gói liên quan tới Glance: glance
- Gói liên quan tới Swift: swift swift-account swift-container swift-object swift-proxy

Chú ý: vì đây không phải là "tất cả" các gói sẽ được cài đặt, sẽ có những gói phụ nữa được tự động cài đặt kèm theo mà chúng ta chưa cần quan tâm. Ở đây tài liệu chỉ xin nêu ra các gói cơ bản nhất để xây dựng một đám mây Openstack trên một node.

Phần này không đi chi tiết vào quá trình cài đặt mà chỉ nêu các bước cần thiết để có thể cấu hình Openstack chạy được với ba thành phần cơ bản. Trong đó chúng tôi sẽ giải thích rõ hơn một số điểm mấu chốt. Phần hướng dẫn cài đặt xin xem thêm trong phần phụ lục.

2. Các bước cài đặt trong thử nghiệm

Sau đây là một số bước chính để cài đặt hệ thống Openstack trong thử nghiệm của nhóm.

*** Cài đặt MySql server***

Mặc định thì Glance và Swift sẽ sử dụng Sqlite server để lưu các metadata cũng như các dữ liệu liên quan, chúng tôi chọn MySql là một CSDL khá phổ biến và quen thuộc với các nhu cầu sử dụng bình thường hiện nay tại môi trường chúng tôi làm việc.

Chúng tôi sẽ sử dụng ba dịch vụ là Nova, Glance và Swift do vậy cần tạo 3 CSDL nova_db, glance_db, swift_db tương ứng với các user: nova, glance, swift để sử dụng trong những cấu hình ở phần sau.

*** Cài đặt các gói cơ bản như unzip, rabbitmq-server, euca2tools...***

Các cài đặt này không đòi hỏi thay đổi tham số gì. Chỉ đơn giản là cài và chúng sẽ hoạt động theo đúng kịch bản.

*** Cài đặt và cấu hình Glance***

Các file cấu hình của Glance sẽ được lưu trong /etc/glance/ chúng ta sẽ thay đổi một số thông tin ví dụ như trong glance-registry.conf. Về CSDL sử dụng để lưu trữ từ Sqlite sang MySql. Ngoài ra trong thử nghiệm này, không cần thiết phải chỉnh thêm thông số nào khác của Glance.

*** Cài đặt và cấu hình Nova***

Các file cấu hình của Nova được lưu tại /etc/nova/.

Trong thử nghiệm này để đơn giản hóa, nhóm sẽ chỉ thay đổi thông số trong /etc/nova/nova.conf. Đây là nơi lưu các cấu hình quan trọng nhất của Nova như thông tin về CSDL, kiểu cấu hình nova-network...

Cấu hình nova-network ở chế độ VLAN. Chỉ sử dụng một interface eth0 từ server DELL nhằm kết nối các instance ra Internet và tạo VLAN cho mỗi project.

Như đã nói ở trên, chúng ta sử dụng dải private IP là: 10.0.0.0/22 32 32 và dải public IP là 172.17.2.64/27 đây là dải IP thực cùng lớp với IP của host (server DELL) do vậy khi cấu hình xong và chạy các instance, chúng ta có thể 'nhìn' thấy các instance đó thông qua dải IP này.

*** Tạo một Nova project***

Như đã nói trong phần user và role trong Nova, chúng ta sẽ tạo một user với tên: testuser và sau đó sẽ gán cho testuser quyền sysadmin. Tiếp đó chúng ta tạo một project tên testproject và gán nó cho testuser với toàn quyền.

*** Tạo các chứng chỉ (credential) access key.***

Với mỗi project, Nova sẽ cung cấp các chứng chỉ và access key cho user nhằm thực hiện việc chứng thực. Các thông tin quan trọng nhất nằm trong file novarc. File này được sử dụng để tạo một 'môi trường' với những tham số trỏ tới server mà Nova được cài đặt.

Giả sử chúng ta dùng một máy client khác để thực hiện các truy vấn trên Nova. Trên client này chúng ta cũng phải lấy các chứng chỉ và access key này về. Từ đó client này có thể dễ dàng tương tác và 'nói chuyện' với server chạy Nova.

Chúng ta cần mở cổng 22 cho SSH service và cổng 80 cho HTTP service.

Sau đó khởi động lại tất cả các dịch vụ của Nova và Glance. Nếu không có lỗi thì từ bây giờ chúng ta đã có thể sử dụng Nova và Glance.

*** Upload image và khởi chạy instance***

Các image có sẵn từ các server của Ubuntu, Stackops... đây cũng là những file chuẩn mà các nhà cung cấp này đã tạo sẵn cho người sử dụng. Chúng ta sẽ cần phải lấy chúng về và upload lên Glance (hoặc nova object-store)

Với mỗi instance chúng ta cần gán cho nó một cặp public/private key. Mục đích là để người dùng có thể sử dụng chúng để đăng nhập tới instance. Public key sẽ được gán vào instace còn private key thì người dùng sẽ lưu lại (đó là một file .pem). Từ client chỉ cần sử dụng private key tương ứng với instace bạn có thể SSH tới instance đó và thực hiện các việc cài đặt thông qua dòng lệnh.

*** Cài đặt và cấu hình Swift***

Như đã giới thiệu, để đảm bảo việc lưu trữ an toàn và hiệu quả, Swift lưu một object (dữ liệu) trên nhiều zone khác nhau. Ở thử nghiệm này do chúng ta chỉ có một server nên tưởng chừng cơ chế này không mấy hiệu quả, nhưng hãy tưởng tượng chúng ta có nhiều server và xa hơn nữa các server nằm tại các vị trí địa lý khác nhau. Sẽ an toàn và dễ dàng hơn rất nhiều trong việc lưu trữ và cải thiện chất lượng dịch vụ đối với người dùng.

Ở thử nghiệm này, nhóm sử dụng một phân vùng đĩa cứng khác để Swift lưu trữ dữ liệu trên đó. Nhóm sẽ tạo ra 4 server mô phỏng cho việc lưu trữ object trên 4 node khác nhau. Bốn node này tất nhiên sẽ cùng IP nhưng sẽ sử dụng các cổng khác nhau cho từng dịch vụ như vậy có thể tạm thời mô phỏng được cách thức hoạt động của Swift.

Trong thử nghiệm chúng tôi sử dụng Swauthkey nhằm đơn giản hơn việc tạo user trong Swift, chúng ta có thể dùng Swauthkey để tạo mới user ngay trên dòng lệnh chứ không cần phải thay đổi thông số trong /etc/swift/proxy-server.conf.

Chúng ta có thể tương tác với Swift thông qua dòng lệnh. Một số trình quản lý FTP như CyberDuck có thể tương tác với Swift khá tốt.

Trên đây tài liệu đã trình bày qua các bước chính và ý nghĩa của chúng theo như kịch bản cài đặt thử nghiệm. Ba thành phần này đều hoạt động, tuy nhiên vẫn có rất nhiều trục trặc, lỗi phát sinh trong quá trình thử nghiệm sẽ được trình bày ở phần tổng kết.

Phần 3: Security trong "Cloud computing"

Để đánh giá về tính bảo mật trong CC, cần xác định các yêu cầu cũng như các định nghĩa về security trong các giải pháp CC. Điều này đồng nghĩa với việc cần nghiên cứu về các tiêu chuẩn đặt ra trong security. Phần này sẽ trình bày và so sánh các tiêu chuẩn CC security của CSA, NIST, và các nghiên cứu khác. Tiếp theo sẽ đề xuất các giải pháp trong CC và cuối cùng là đánh giá độ an toàn của OpenStack Diablo.

I. CSA

CSA là tổ chức phi chính phủ được thành lập năm 2008 nhằm mục đích nghiên cứu các vấn đề về security trong CC với sự hợp tác của rất nhiều công ty lớn trên thế giới như Microsoft, Google, IBM, VMware, … Phiên bản đầu tiên của CSA là 1.0 ra đời tháng 4/2009, sau đó là phiên bản 2.1 ra đời tháng 12 cùng năm với những nguy cơ security được thêm mới như Information Lifecycle Management và Storage. Hiện nay CSA đang ở phiên bản 3.0 với một số cải tiến và mở rộng, chẳng hạn Security as a Service.

Các tiêu chính đánh giá của CSA đều dựa trên các nghiên cứu và được thẩm định trong giới học thuật (peer-review) trước khi được công bố thành các phiên bản. Phần này sẽ tóm lược các yêu cầu về security mà CSA đưa ra để có một cái nhìn về lĩnh vực rộng lớn security trong CC. CSA chia các yêu cầu về security ra 2 phần chính với các vấn đề như sau:

1. Quản lý trong CC (5 phần)

Phần này CSA khuyến cáo với nhiều phân mục [20], tuy nhiên trong khuôn khổ tài liệu chỉ đề cập đến:

- ***Governance and Enterprise Risk Management**: khả năng quản lý và kiểm soát các mối nguy hại trong môi trường kinh doanh (Enterprise), các vấn đề liên quan đến nhận thức người dùng, sự phối hợp giữa Provider với người dùng trong trách nhiệm bảo vệ các dữ liệu bí mật.*
- ***Information Management and Data Security**: quản lý dữ liệu lưu trữ trong cloud, việc định danh dữ liệu, chống thất thoát, mất dữ liệu khi di chuyển data. Đảm bảo tính bảo mật, toàn vẹn, và sẵn sàng cho dữ liệu (confidentiality, integrity, availability).*
- ***Interoperability and Portability**: di chuyển dữ liệu và service từ 1 nhà cung cấp sang 1 nhà cung cấp khác - interoperability, cũng như đem toàn bộ data back-in-house.*

Ngoài ra còn có nội dung về SLA – được xem như là các quy định đảm bảo mức độ an toàn và sẵn sàng của hệ thống đối với người dùng, tùy chi phí mà SLA có thể cao hay thấp.

2. Hoạt động trong CC (8 phần)

- ***Traditional Security, Business Continuity, and Disaster Recovery**: CC cũng phải đối mặt với các hiểm họa an ninh như các hệ thống truyền thống khác cũng như vấn đề back-up hệ thống và recovery khi có sự cố xảy ra.*
- ***Data Center Operations**: đảm bảo sự hoạt động của Data Center với đầy đủ các tính năng đồng thời với độ ổn định cao và lâu dài.*
- ***Incident Response**: kiểm soát, báo hiệu trong việc xảy ra các sự cố.*
- ***Application security***

- ***Encryption and Key Management**: bảo vệ việc truy cập vào tài nguyên hệ thống cũng như bảo vệ dữ liệu.*
- ***Identity, Entitlement, and Access Management**: đảm bảo quản lý việc truy cập và chứng thực user.*
- ***Virtualization**: những nguy hại liên quan đến VM isolation, lỗ hổng của các Hypervisor.*
- ***Security as a Service**: đây là 1 yêu cầu mới về security trong phiên bản này về xác nhận đảm bảo security từ các hãng thứ 3 có uy tín trên thế giới thông qua việc kiểm tra security. Việc này đảm bảo sự tin tưởng của khách hàng cũng như tạo ra 1 động lực để nâng cao security cho hệ thống.*

Phần này đã tóm lược các tính năng liên quan đến security trong cloud được CSA khuyến cáo khi muốn triển khai CC. Phần kế sẽ cùng chủ đề nhưng được khuyến nghị bởi NIST.

II. NIST

NIST – National Institute of Standards and Technology đã khuyến nghị 1 guidelines hoành chỉnh cho security và privacy trong CC vào tháng 12/2011. Mục đích của NIST là cung cấp một cái nhìn tổng quan về CC và các thách thức bảo mật trong CC [21]. Các vấn đề được NIST đưa ra là:

- ***Governance***
- ***Compliance***
- ***Trust**: các vấn đề về data ownweship, Insider Access, hay Risk management*
- ***Architecture**: thiết lập bảo vệ cho các máy ảo (VM), mạng ảo (Virtual Network), phía người dùng và phía server.*
- ***Identity and Access Management**: thực thi Authentication và Access control*
- ***Software Isolation***
- ***Data Protection***
- ***Availability**: bảo vệ chống lại các mối nguy hại liên quan đến độ sẵn sàng của hệ thống như DoS, outages (đảm bảo hệ thống điện, nguồn).*
- ***Incident Response***

Nhìn chung, các tiêu chí của NIST và CSA khá giống nhau khi gần như rà soát toàn bộ các yêu cầu đảm bảo cho một hệ thống an toàn, ví dụ về đảm bảo chứng thực và quyền truy cập (authentication và access control), hay các phản ứng khi có sự cố, cũng như software/application security, … Về các giải pháp và các khuyến nghị (recommendation) cụ thể của CSA hay NIST sẽ được trình bày trong phần IV.

III. Các nghiên cứu từ các trường đại học

Một mô hình "điện toán đám mây", theo các chuyên gia của đại học Azad, Iran và các nghiên cứu đưa ra trong hội nghị lần thứ 8-2009 của IEEE, cần thỏa mãn các yêu cầu sau về security [5]:

- ***Availability management**: độ sẵn sàng của hệ thống trong mọi trường hợp*
- ***Access control management**: quản lý việc truy cập*
- ***Vulnerability and problem management**: khả năng ngăn cản các lỗ hổng và thâm nhập*

- ***Patch and configuration management**: vấn đề update hệ thống thường xuyên ngay khi có bản vá và cấu hình*
- ***Countermeasure**: các biện pháp đối phó khi gặp sự cố về security*
- ***Cloud system using and access monitoring**: quản lý việc sử dụng và truy cập của user với cloud.*

Đặc thù của mạng đám mây với nhiều loại hình khách hàng khác nhau từ người dùng phổ thông (ordinary users), giới nghiên cứu (academia) hay các doanh nghiệp kinh doanh (enterprise). Sự tỷ lệ nghịch giữa "security" và "performance" luôn là vấn đề cần phải đặt ra và giải quyết nếu muốn đảm bảo 1 hệ thống đám mây "an toàn" với "hiệu suất cao". Tuy nhiên với các loại hình khách hàng khác nhau, nhu cầu cũng khác nhau. Chẳng hạn với khách hàng enterprise, nhu cầu về security được ưu tiên hàng đầu trên cả performance trong khi giới academia ưu tiên vấn đề hiệu suất cao. Về bản chất, "cloud computing" cũng là một môi trường mạng public như các mạng truyền thống nên vẫn phải đối mặt với các vấn đề an ninh cơ bản như các lỗ hỗng của web application (SQL injection hay Cross-site scripting), DNS poisoning hay ARP poisoning, … Tuy nhiên, vấn đề về security trong "cloud computing" đặt trọng tâm vào việc đánh giá qua Information Security Policies – các chính sách bảo vệ thông tin và Cloud RAS (reliability, availability, and security) issues – các nguy hại về an ninh gặp phải trong đặc thù môi trường cloud.

1. Information Security Policies

Một số điểm yếu về Information security policies có thể kể ra như sau [22]:

- ***Privileged user access***
- ***Regulatory compliance:** khách hàng (users) phải chịu trách nhiệm hoàn toàn về tính "an toàn" và "toàn vẹn (integrity)" của dữ liệu. Trong mô hình truyền thống việc này được sự giúp đỡ của các tổ chức kiểm toán (external audits) và các security certifications.*
- ***Data location:** người sử dụng cloud sẽ không thể biết được chính xác dữ liệu của họ lưu trữ ở đâu. Cơ chế lưu trữ phân tán gây ra sự mất điều khiển cho người dùng và điều này là mối lo khi chuyển từ lưu trữ local sang lưu trữ trên đám mây.*
- ***Data segregation**: Dữ liệu người dùng được lưu trữ chung với nhau và việc bảo vệ được thực thi bằng mã hóa. Tuy nhiên phương pháp cổ điển này không thể đảm bảo vấn đề "an toàn thông tin". Tuy nhiên hiện nay vẫn chưa có 1 giải pháp hoàn mỹ cho vấn đề này.*
- ***Recovery**: mọi vấn đề về backup và khôi phục dữ liệu được người dùng mong muốn thực hiện thông qua Cloud Provider mà không phải 1 hãng thứ 3.*
- ***Investigative support***
- ***Long-term viability***

2. Cloud RAS issues

Sự phát triển của cloud đến mức nào đó đòi hỏi phải có sự kết hợp giữa các nhà cung cấp cloud với người dùng để phát triển các ứng dụng. Việc chia sẻ này đồng nghĩa với việc gia tăng các mối hiểm họa bảo mật và đòi hỏi nhiều thách thức trong việc quản lý bảo mật cho đội ngũ IT. Các nguy cơ tiềm ẩn mới trong mạng cloud có thể kể đến:

2.1. Data Leakage

- Khi chuyển hướng sang mô hình cloud sẽ có 2 sự thay đổi lớn về mặt dữ liệu của người dùng cần được quan tâm sát sao: dữ liệu sẽ được lưu trữ cách xa người dùng,

không còn trên ổ đĩa local truyền thống và dữ liệu sẽ được lưu trữ trên nhiều nguồn (multi-tenant environment) thay vì một đầu mối như trước đây (single-tenant environment) [23]. Đây chính là mối quan tâm chính trong vấn đề bảo mật.

2.2. Cloud security issues

- Bản chất của cloud provider cũng là sự truyền thông trên Internet sử dụng giao thức TCP/IP mà trong đó các user được định danh bởi địa chỉ IP. Cũng giống như mạng thuần vật lý, mỗi máy ảo trên Internet cũng được định danh bằng ít nhất 1 địa chỉ IP mà có thể dễ dàng tìm thấy bởi người dùng hay attackers. Tương tự như máy vật lý, attackers có thể xâm nhập từ máy ảo qua máy chủ vật lý.
- Attacks in cloud: ngày nay có rất nhiều loại hình tấn công mạng, và về lý thuyết, tất cả các loại hình đó có thể được áp dụng để đe dọa cloud – tùy thuộc mức độ khác nhau. Chẳng hạn khi 2 users trong cùng mạng cloud sử dụng máy ảo, có thể xem như 2 máy vật lý chung 1 network.
- DDoS attacks against Cloud: tấn công DDoS là kiểu tấn công với số lượng lớn gói tin IP đến 1 mạng nhất định với mục đích làm ngưng trệ toàn bộ hệ thống mạng đó. Với đặc điểm rất nhiều người dùng trong 1 mạng điện toán đám mây, sự nguy hại nếu hệ thống bị ngưng trệ là rất lớn hơn trong mô hình kiến trúc đơn điểm [5]. Phần lớn các mạng không thể nào bảo vệ để chống lại tấn công DDoS bởi vì lượng traffic đổ về từ hàng ngàn, vạn máy trên internet, đồng thời cũng rất khó để phân biệt bad traffic và good traffic. Hệ thống IPS rất hữu hiệu để ngăn chặn DDoS nhưng với các kiểu tấn công đã được nhận dạng hoặc với các gói tin, tập tin nhiễm độc đã được lưu trữ (pre-existing signature). Tuy nhiên với những gói tin hợp lệ mang nội dung xấu vẫn được cho qua. Giải pháp firewall cũng không còn hữu hiệu với DDoS khi các gói tin bypass firewall còn dễ dàng hơn IDS/IPS [5].

IV. Các giải pháp security cho mô hình "Cloud Computing"

Với các đặc trưng riêng của cloud đã là giải pháp cho một số vấn đề truyền thống về security, chẳng hạn vấn đề downtime hệ thống, backup, lưu trữ phân tán, hay DoS. Một số giải pháp trong cloud được đề xuất để đảm bảo an toàn như sau:

1. Access control and management

Thiết lập 1 cơ chế điều khiển việc truy cập là rất cần thiết cho việc "an toàn thông tin" để ngăn chặn việc truy xuất trái phép. Ví dụ việc chỉ định quyền hạn cho user sử dụng các dữ liệu và dịch vụ. Một lưu ý cho cơ chế này là phải bao trùm tất cả các quá trình của 1 user từ khi mới bắt đầu khởi tạo (initial registration) cho đến khi kết thúc là không truy cập vào hệ thống và dịch vụ nữa(de-registration). Theo tiêu chuẩn của Information Technology Infrastructure Library (ITIL) và ISO 27001/27002 về bảo mật, một hệ thống "Security management" phải đảm bảo các chức năng sau [24]:

- ***Control access to information****: truy cập vào thông tin*
- ***Manage user access rights****: quản lý quyền hạn người dùng*
- ***Encourage good access practices***
- ***Control access to network services.***
- ***Control access to operating systems.***
- ***Control access to applications and systems***

Việc quản lý truy cập trong clouds được chia ra 3 phần theo mô hình cung cấp dịch vụ của clouds

- ***SaaS:*** *mặc dù cloud providers quản lý tất cả các lĩnh vực bao gồm mạng, servers, và hạ tầng ứng dụng. Tuy nhiên trong mô hình SaaS khi ứng dụng được cung cấp dưới dạng dịch vụ thông thường qua trình duyệt web, việc quản lý network-based gần như không liên quan mà tập trung vào vấn đề quản trị người dụng, các cơ chế chứng thực mạnh và sử dụng one-time password [5, 24], Single Sign On [25], quản lý quyền hạn, …*
- ***PaaS:*** *Khác với mô hình SaaS, trong PaaS, việc quản lý trọng tâm vào tầng network, servers, và các platform hạ tầng ứng dụng. Tuy nhiên trong trường hợp này, người dùng phải chịu trách nhiệm cho việc quản lý các ứng dụng đặt trên platform PaaS. Tuy nhiên, việc truy cập vào các ứng dụng phải được quản lý, chỉ định, và chứng thực.*
- ***IaaS:*** *các khách hàng của IaaS phải chịu hoàn toàn trách nhiệm cho việc quản lý truy cập đến tài nguyên của họ trên cloud. Việc truy cập vào các server ảo, network ảo, hệ thống lưu trữ ảo, và các ứng dụng trên một IaaS platform sẽ được thiết kế và quản lý bởi khách hàng. Việc quản lý truy cập ở mô hình IaaS bao gồm 2 phần chính: quản lý host, network, và ứng dụng thuộc sở hữu của cloud provider trong khi người dùng phải quản lý việc truy cập đến các server ảo, lưu trữ ảo, networks ảo, và các ứng dụng chạy trên các virtual servers [24, 25].*

2. Các biện pháp đối phó khi xảy ra các vấn đề về security

Một trong những điểm quan trọng của cloud security là tìm ra các vấn đề và lỗ hỗng bảo mật tồn tại, sau đó triển khai các biện pháp thích hợp để đối phó. Nhìn chung, hệ thống cloud được xây dựng trên một bộ nhiều engines lưu trữ với khả năng hỗ trợ "high availability" đáp ứng được việc backup qua lại cho các server ảo và thật nếu có sự cố xảy ra. Để đạt được độ linh hoạt, khả năng mở rộng và hiệu suất sử dụng, cloud providers phải đối mặt với những vấn đề trong việc phân tích và tính toán để phân bổ hợp lý tài nguyên cho các công việc tính toán khác nhau.

- ***Partitioning****: một ví dụ khi muốn nâng cao hiệu suất tính toán của các ứng dụng trên cloud là chia dữ liệu ra nhiều partitions để thực hiện tính toán trên nhiều nodes nhằm mục đích tăng hiệu suất của các query và transaction. Vì thế, các kết quả được tính toán rất nhanh chóng và trả về.*
- ***Migration****: Sự linh hoạt là một trong những yêu cầu chính của cloud, trong ngữ cảnh cung cấp các dịch vụ cloud cần linh hoạt trong việc sử dụng tài nguyên. Ví dụ tài nguyên phải được dành riêng cho các hoạt động cần thiết và quan trọng nhất. Chính điều này làm cho việc quá tải của các node trong cloud không xảy ra khi có sự di chuyển (migration) của hệ thống, đặc biệt là hệ thống CSDL lớn, đặc biệt vẫn đảm bảo duy trì hoạt động của hệ thống khi migration xảy ra.*
- ***Workload analysis and allocation***

Ngoài ra cần tính đến các giải pháp disaster recovery khi có các sự cố bất ngờ xảy ra như thiên tai, lũ lụt, cháy nổ, …

3. DDoS

Như đã trình bày, với các hệ thống có đầy đủ Firewall và IDS/IPS vẫn có thể bị tấn công DDoS. Tuy nhiên, nếu với một hạ tầng mạng đủ mạnh vẫn có thể chịu được với lưu lượng

DDoS cực lớn. Hạ tầng cloud đảm bảo cho điều này khi toàn bộ hạ tầng là sự liên kết của hàng trăm, ngàn máy tính. Điều này giúp cho quản trị viên có đủ thời gian để giải quyết sự cố tìm ra nguyên nhân khắc phục. Ví dụ hệ thống IPS sẽ học được các quy tắc tấn công mới hay quản trị viên tiến hành phân tích gói và thiết lập các rules để "drop" các gói tin đến vi phạm.

IV. OpenStack Security

Bất kỳ sự triển khai CC nào trong thực tế dù là sản phẩm thương mại hay opensource cũng phải đáp ứng các yêu cầu cũng như biện pháp xử lý đặt ra về security. Hiện nay với mục tiêu sử dụng bộ công cụ nguồn mở OpenStack, việc triển khai chỉ mới hoàn thiện ở mức độ cơ bản (3 projects đầu – trình bày ở phần 4) nên các công cụ hỗ trợ security trong OpenStack vẫn còn tiếp cận ở mức hạn chế.

- ***Keystone*** *(hay OpenStack Identity) chính là thành phần chính cho security với các chức năng chứng thực, chính sách, … đã trình bày sơ lược ở trên.*
- ***User và Project****: việc tạo các user và project cũng đảm bảo việc truy cập chứng thực khi user không thể truy cập vào các project không thuộc chủ quản của mình – chức năng User và Project trong Nova.*
- ***Keypairs****: Tạo các khóa để gán cho instance khi khởi tạo cũng là 1 công cụ đảm bảo security khi chỉ có user được cấp khóa mới đủ thẩm quyền truy cập instance.*

Việc triển khai keystone hiện nay chưa thành công nên mức độ đánh giá chưa chính xác. Nhưng các chức năng security trong keystone sẽ đảm bảo an toàn cho việc triển khai một IaaS.

Phần 4: Tổng kết

Trong 8 tuần thử nghiệm nhóm đã gặp khá nhiều khó khăn trong quá trình cài đặt cấu hình các thành phần của Openstack. Đây là một hệ thống phức tạp nhiều thành phần đang trong quá trình phát triển nên tài liệu về nó có nhiều chỗ không được chính xác và cập nhật kịp thời.

Sau khi tham khảo nhiều hướng dẫn từ những người khác cũng đang thử nghiệm Openstack, nhóm quyết định chỉ tập trung thực hiện việc thử nghiệm ba thành phần Nova, Glance và Swift. Một phần vì hai thành phần Dashboard và Keystone chưa hoạt động tốt trong phiên bản Diablo, một phần vì thời gian ra mắt phiên bản mới Essex cũng khớp với thời điểm kết thúc thử nghiệm. Do vậy việc thử nghiệm tất cả các thành phần kể trên của Openstack sẽ được kiểm tra với phiên bản Essex trong thời gian tới để tránh việc lãng phí thời gian. Sau đây là một số điểm tổng kết lại sau quá trình thử nghiệm:

**** Những việc đã đạt được

- Đã hoàn tất cài đặt và vận hành được ba thành phần chính của Openstack là Nova, Glance và Swift tương ứng với hai dịch vụ quan trọng nhất là cung cấp tài nguyên về Cloud compute và Cloud storage.
- Đã thử nghiệm các máy ảo của Linux cơ bản như Ubuntu, Debian, CentOs. Tuy nhiên, chỉ sử dụng các file image được cung cấp sẵn từ các hãng này. Trong đó Ubuntu là nhà cung cấp tốt nhất và đầy đủ nhất về các phiên bản. (Lí do một phần vì họ là đối tác chính của AWS và Openstack, thêm nữa công ty này đang đầu tư rất lớn vào CC)
- Swift - thành phần lưu trữ – hoạt động bình thường, tương tác được với một số GUI như CyberDuck trên Windows và MacOS.
- Kiểm tra được các chức năng cơ bản thông qua API và dòng lệnh (euca2tools)

Bản thân các gói Nova, Glance và Swift cũng đã cung cấp các chức năng xem thông tin, upload, quản lý instance và dữ liệu, tuy nhiên vì tên của các câu lệnh khá 'rời rạc' nên thông thường để tiện cho việc sử dụng người ta hay dùng euca2tools. Gói này tương thích với cả Eucalyptus và Openstack. Tất cả thao tác trong thử nghiệm đều sử dụng các câu lệnh euca* này.

**** Những việc chưa đạt được

- Chưa thử nghiệm được hai thành phần Keystone và Dashboard. Các thao tác quản lý chỉ thực hiện được thông qua dòng lệnh.
- Với tài nguyên về tính toán: đã có thể chạy các instance (máy ảo) Linux khá ổn định. Tuy nhiên với các máy ảo chạy Windows vẫn chưa thành công. Vẫn có nhiều lỗi phát sinh chẳng hạn như bị mất instance hoặc không thể đăng nhập vào instance. Các lỗi này theo đánh giá có thể xuất phát từ tính chưa ổn định của Openstack, thứ nữa là do

trình độ của nhóm chưa hiểu rõ về Linux và Openstack để có thể tìm hiểu và giải quyết triệt để.

- Với tài nguyên về lưu trữ: mới chỉ kiểm tra được sơ bộ cách thức hoạt động, chưa có tìm hiểu được sâu về khả năng lưu trữ mở rộng, lưu trữ file kích thước lớn...
- Ngoài ra vì giới hạn bởi điều kiện phần cứng, thời gian triển khai nên vẫn chưa kiểm tra được hiệu năng của các thành phần. Một số chức năng cần có nhiều hơn một máy chủ để kiểm tra như việc di chuyển một instance, cân bằng tải...vẫn chưa được thực hiện.

**** Kế hoạch trong việc thử nghiệm kế tiếp

Phiên bản mới Essex của Openstack mới ra mắt đầu tháng 04/2012 hứa hẹn nhiều cải tiến, và ổn định hơn. Hai thành phần Keystone và Dashboard cũng được giới thiệu tương thích tốt hơn với các thành phần còn lại.

Kế hoạch tiếp theo là chuyển sang thử nghiệm phiên bản này, kết hợp với việc mở rộng hạ tầng vật lý. Với một mô hình lớn hơn, khả năng kiểm tra hiệu năng của các thành phần trong Openstack sẽ chi tiết và được đánh giá chính xác, đúng hơn.

Phụ lục:

Phụ lục 1: Tutorial cài đặt OpenStack trên Ubuntu 11.10 64 bits

Xem file PDF đi kèm

Phụ lục 2: Một số link tham khảo khác

Security Solutions for Cloud Computing http://infosecisland.com/blogview/5449-Security-Solutions-for-Cloud-Computing.html

---

Security Issues and Solutions in Cloud Computing
http://wolfhalton.info/2010/06/25/security-issues-and-solutions-in-cloud-computing/

http://www.hastexo.com/resources/docs/installing-openstack-essex-20121-ubuntu-1204-precise-pangolin

References:

1. *Điện toán đám mây*. Available from: http://vi.wikipedia.org/wiki/%C4%90i%E1%BB%87n_to%C3%A1n_%C4%91%C3%A1m_m%C3%A2y.
2. *Cloud computing*. Available from: http://en.wikipedia.org/wiki/Cloud_computing.
3. *Data center*. Available from: http://en.wikipedia.org/wiki/Data_center.
4. Wind, S. *Open source cloud computing management platforms: Introduction, comparison, and recommendations for implementation*. in *Open Systems (ICOS), 2011 IEEE Conference on*. 2011.
5. Sabahi, F. *Cloud computing security threats and responses*. in *Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on*. 2011.
6. *Ubuntu Cloud: Technologies for future-thinking companies*. 2012; Available from: http://www.canonical.com/about-canonical/resources/white-papers/ubuntu-cloud-technologies-future-thinking-companies.
7. Wesselius, J. *Windows Server Virtualisation: Hyper-V, an Introduction*. 2009; Available from: http://www.simple-talk.com/sysadmin/virtualization/windows-server-virtualisation-hyper-v,-an-introduction/.
8. *Full virtualization*. Available from: http://en.wikipedia.org/wiki/Full_virtualization.
9. *Ảo hóa trong điện toán đám mây*. 2012; Available from: http://hpcc.hut.edu.vn/forum/archive/index.php/thread-658.html.
10. *Paravirtualization*. Available from: http://en.wikipedia.org/wiki/Paravirtualization.
11. *Operating system-level virtualization*. Available from: http://en.wikipedia.org/wiki/OS-level_virtualization.
12. *Eucalyptus open source cloud computing infrastructure - Overview*. 2011; Available from: http://go.eucalyptus.com/Eucalyptus-Open-Source-Cloud-Computing-Infrastructure-An-Overview-Download.html.
13. Mahjoub, M., et al. *A Comparative Study of the Current Cloud Computing Technologies and Offers*. in *Network Cloud Computing and Applications (NCCA), 2011 First International Symposium on*. 2011.
14. Leads, O.P. *OpenNebula 3.2 Key Features and Functionality*. 2012 [cited 2012 22nd March]; Available from: http://opennebula.org/documentation:features.
15. Systems, C. *Xen Cloud Platform Project*. 2012 [cited 2012 22nd March]; Available from: http://www.xen.org/products/cloudxen.html.
16. Abiquo. *Abiquo Overview*. 2012 [cited 2012 22nd March]; Available from: http://www.abiquo.com/products/abiquo-overview.php.
17. *OpenStack Compute*. 2012; Available from: http://openstack.org/projects/compute/.
18. *OpenStack Object Storage*. 2012; Available from: http://openstack.org/projects/storage/.
19. *OpenStack Image Service.* 2012.
20. Archer;, J., et al., *Security Guidance for Critical Areas of Focus in Cloud Computing v3.0.* Cloud Security Alliance, 2011.
21. Jansen, W. and T. Grance., *Guidelines on security and privacy in public cloud computing*, 2011.
22. Brodkin, J. *Gartner Seven cloud-computing security risks*. 2008; Available from: http://www.infoworld.com/d/security-central/gartner-seven-cloud-computing-security-risks-853.
23. Stokes, J., *T-Mobile and Microsoft/Danger data loss is bad for the cloud.* 2009.
24. *Security management in the cloud*. 2010; Available from: http://mscerts.programming4.us/programming/Security%20Management%20in%20the%20Cloud.aspx.

25. *Security Management in the Cloud - Access Control*. 2012; Available from: http://mscerts.programming4.us/programming/Security%20Management%20in%20the%20Cloud%20-%20Access%20Control.aspx.