MỤC LỤC

DANH MỤC CÁC TỪ VIẾT TẮT

Amazon EC2: Amazon Elastic Compute Cloud.

Amazon S3: Amazon Simple Storage Service.

Amazon SimpleDB: Amazon SimpleDatabase.

Amazon SQS: Amazon Simple Queue Service.

AMI: Amazon Machine Image.

DAC: Direct-attached storage.

DHCP: Dynamic Host Configuration Protocol.

EMI: Eucalyptus Machine Image.

IaaS: Infrastructure as a Service.

LAN: Local Area Network.

NAS: Network-attached storage.

PaaS: Platform as a Service.

SaaS: Software as a Service.

VLAN: Virtual Local Area Network.

GIẢI THÍCH MỘT SỐ THUẬT NGỮ

Image: là tệp khuôn mẫu dùng để tạo các máy ảo trong Eucalyptus. Các máy ảo được từ cùng một tệp image sẽ có trạng thái ban đầu giống hệt nhau. Mỗi Image có một tên định danh riêng dạng emi-xxxxxx, với xxxxxx là 6 chữ số dạng hexa (Ví dụ: *emi-23409A*).

Instance: là máy ảo của Eucalyptus, được tạo ra từ tệp image. Mỗi Instance có một tên định danh riêng dạng i-xxxxxx, với xxxxxx là 6 chữ số dạng hexa (Ví dụ: *i-4353B4*).

Public IP: là địa chỉ IP được gán cho máy ảo khi vừa chạy lên. Địa chỉ này dùng để giao tiếp với các máy khác ở các cluster khác của hệ thống Private Cloud.

Private IP: là địa chỉ IP được gán cho máy ảo khi vừa chạy lên. Địa chỉ này dùng để giao tiếp với các máy khác trong cùng một cluster của hệ thống Private Cloud.

Volume: là phần dung lượng lưu trữ được gắn thêm vào máy ảo.

Snapshot: là ảnh chụp trạng thái của các volume. Có thể tạo ra một volume mới từ một snapshot đã có sẵn.

Worker: Khi kiểm thử khả năng chịu tải của CPU trong các máy ảo bằng cách gửi các HTTP request một cách tự động, các worker đóng vai trò như những chiếc máy tự động gửi các request. Số lượng request được yêu cầu gửi sẽ được chia đều cho số lượng worker.

Host: là một máy tính hay một thiết bị nào đó được giám sát bởi NRPE.

Monitoring host: là máy chủ thực hiện giám sát một máy tính khác trong hệ thống.

Remote host: là máy tính chịu sự giám sát bởi một máy tính khác.

Chương 1 – TÌM HIỂU LÝ THUYẾT CLOUD COMPUTING

1.1. Tổng quan về Cloud Computing

1.1.1. Hiện trạng thực tế

Ngày nay, đối với các công ty, doanh nghiệp, việc quản lý tốt, hiệu quả dữ liệu của riêng công ty cũng như dữ liệu khách hàng và đối tác là một trong những bài toán được ưu tiên hàng đầu và đang không ngừng gây khó khăn cho họ. Để có thể quản lý được nguồn dữ liệu đó, ban đầu các doanh nghiệp phải đầu tư, tính toán rất nhiều loại chi phí như chi phí cho phần cứng, phần mềm, mạng, chi phí cho quản trị viên, chi phí bảo trì, sửa chữa,… Ngoài ra họ còn phải tính toán khả năng mở rộng, nâng cấp thiết bị; phải kiểm soát việc bảo mật dữ liệu cũng như tính sẵn sàng cao của dữ liệu.

Các công ty lớn thường dành nhiều kinh phí và nguồn lực cho việc xây dựng cơ sở hạ tầng. Cơ sở hạ tầng càng tốt thì lợi thế cạnh tranh càng lớn. Phương châm làm việc của họ là *"Cứ tạo ra thật nhiều, rồi sau này sẽ dùng đến"*. Trong hầu hết các trường hợp, cách tiếp cận này: [1]

- ✓ Để lại một lượng lớn các tài nguyên tính toán không được sử dụng, làm tiêu tốn không gian trong các trung tâm dữ liệu lớn.
- ✓ Bắt buộc phải có ai đó trông giữ các máy chủ.
- ✓ Chi phí năng lượng rất tốn kém.
- ✓ Công suất tính toán không được sử dụng bị bỏ phí mà không có cách nào chuyển sang các công ty khác.

1.1.2. Nhu cầu của người dùng

Từ những nhu cầu thực tế của người dùng như: với một máy PC cấu hình hạn chế, nhưng muốn làm một việc mà hàng ngàn máy mới có thể làm được; máy PC chỉ có 100GB dung lượng ổ cứng, tuy nhiên lại muốn lưu trữ rất nhiều tấm ảnh và video chất lượng cao hàng terabyte, có thể truy cập được ở mọi nơi mà không cần phải dùng một thiết bị lưu trữ di động nào; mang chiếc PDA phone bé bằng bàn tay đi đây đi đó, nhưng lúc nào cũng có dữ liệu và thông tin cần thiết;… mà không cần phải bỏ ra một khoản tiền lớn ban đầu để mua

chúng; muốn tự thiết kế một trang web, hay một phần mềm chức năng nào đó mà không cần phải có kỹ năng lập trình.

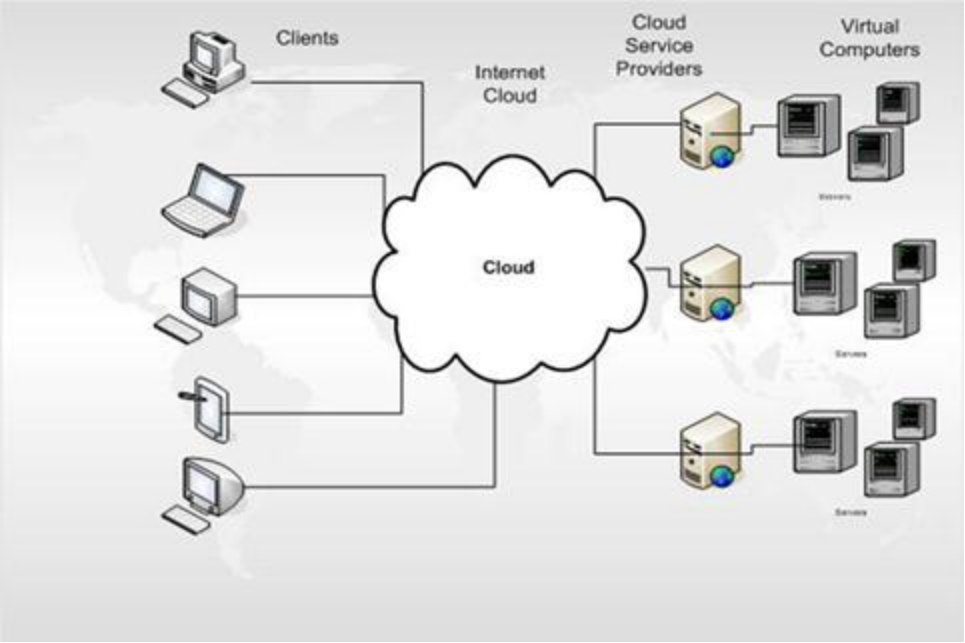
Tất cả những điều trên, có người cho rằng đó là điều không thể. Nhưng nó đã thành hiện thực nhờ một công nghệ mới, đó là Cloud Computing.

1.1.3. Định nghĩa Cloud Computing

Từ "Cloud" (đám mây) trong Cloud Computing thực chất chỉ là một phép ẩn dụ để mô tả Internet. Thực chất thì Cloud Computing là biện pháp sử dụng tài nguyên tính toán dựa trên kết nối Internet, nơi mà những người dùng chia sẻ cùng một mạng máy chủ, phần mềm và dữ liệu. Đây là định nghĩa cơ bản giúp chúng ta hiểu một cách cơ bản về Cloud Computing nhất.

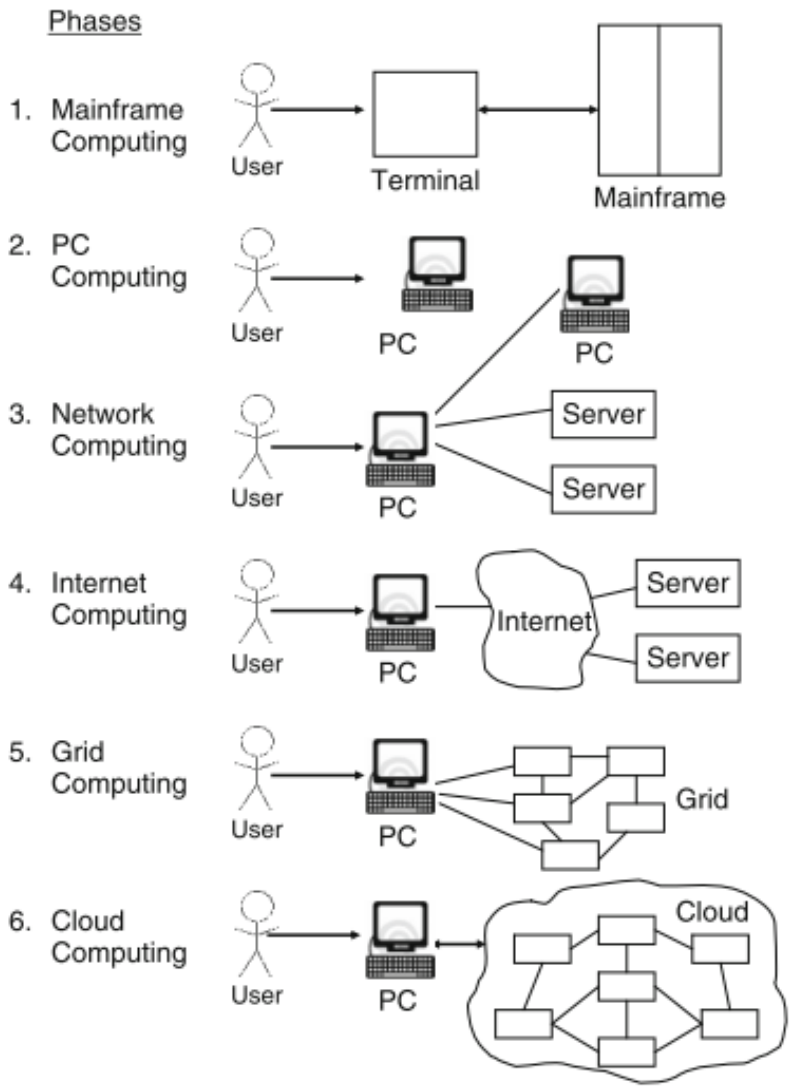
Với Cloud Computing, các máy tính dư thừa sẽ được đưa vào sử dụng và sinh lợi bằng cách cho khách hàng thuê lại. Việc chuyển đổi cơ sở hạ tầng công nghệ thông tin vào một tiện ích như vậy, sẽ có tác dụng hiệu quả trong hầu hết các trường hợp hoặc trong một số mức độ cho phép. Việc cạnh tranh khi đó sẽ trở thành cạnh tranh dựa trên ý tưởng, không còn là cạnh tranh về tài nguyên tính toán.

Theo Ian Foster: "Một mô hình điện toán phân tán có tính co giãn lớn, là nơi có sức mạnh tính toán, lưu trữ, và là nền tảng (platform) cho các dịch vụ trực quan, ảo hóa và co giãn linh động,… Tất cả được phân phối theo nhu cầu cho các khách hàng bên ngoài thông qua Internet". "Cloud Computing là một dạng thức điện toán cung cấp các tài nguyên ảo hóa và có quy mô dưới dạng dịch vụ qua mạng Internet. Người dùng không cần tới những kiến thức chuyên môn để quản lý hạ tầng công nghệ này bởi phần việc đó là dành cho các nhà cung cấp dịch vụ". [2]



Hình 1.1 – Định nghĩa về Cloud Computing.

Theo NIST (*National Institute of Standards and Technology*): "Cloud Computing là mô hình điện toán cho phép truy cập qua mạng để lựa chọn và sử dụng tài nguyên tính toán (ví dụ: mạng ảo, máy chủ, lưu trữ, ứng dụng và dịch vụ) theo nhu cầu một cách thuận tiện và nhanh chóng; đồng thời cho phép kết thúc sử dụng dịch vụ, giải phóng tài nguyên dễ dàng, giảm thiểu các giao tiếp với nhà cung cấp". [3]



Hình 1.2 – Sáu mô hình tính toán.[4]

Hình 1.2 ở trên cho thấy sáu giai đoạn hình thành và phát triển của các máy tính, gồm có các thiết bị đầu cuối (Terminal), máy tính lớn (Mainframe), máy tính cá nhân (PC), mạng máy tính (Network), Grid Computing và Cloud Computing.

- ✓ Giai đoạn 1, ban đầu người dùng sử dụng các máy tính có kích thước rất lớn thông qua thiết bị đầu cuối.
- ✓ Giai đoạn 2, máy tính cá nhân độc lập đã đủ mạnh để đáp ứng phần lớn các nhu cầu của con người.

- ✓ Giai đoạn 3, máy tính cá nhân, máy tính xách tay, và máy chủ kết nối với nhau thành mạng lưới cục bộ để chia sẻ dữ liệu và tăng hiệu suất.
- ✓ Giai đoạn 4, các mạng cục bộ nhỏ kết nối với nhau tạo thành mạng lưới Internet toàn cầu, người dùng có thể sử dụng các ứng dụng và nguồn tài nguyên từ các máy tính khác trên phạm vi toàn cầu.
- ✓ Giai đoạn 5, mô hình điện toán lưới – các máy tính trên thế giới có thể kết hợp với nhau để cùng giải quyết một vấn đề / một bài toán lớn.
- ✓ Giai đoạn 6, mô hình Cloud Computing – là mô hình điện toán lưới cải tiến, cung cấp cho người sử dụng Internet các dịch vụ xử lý tính toán, lưu trữ theo những cách thức tiện dụng và hiệu quả nhất.

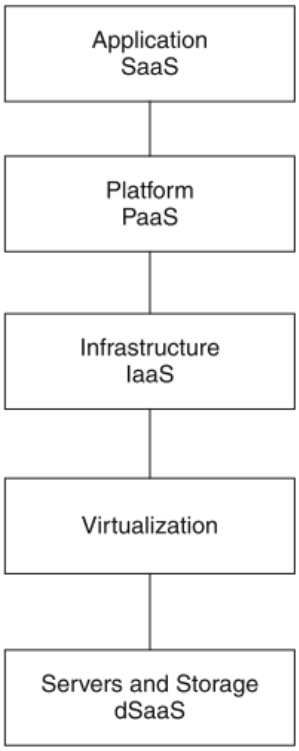
Nếu đem so sánh sáu mô hình tính toán này với nhau, trông có vẻ như mô hình Cloud Computing giống hệt với mô hình máy Mainframe ban đầu. Tuy nhiên, có sự khác biệt rất rõ giữa hai mô hình này. Đó là máy Mainframe có khả năng tính toán rất hạn chế, trong khi Cloud Computing cung cấp khả năng tính toán và lưu trữ rất lớn. Ngoài ra, trong mô hình máy Mainframe người dùng phải giao tiếp thông qua thiết bị đầu cuối rất khó sử dụng, trong khi Cloud Computing giao tiếp với người dùng từ chính máy tính cá nhân của họ thông qua các ứng dụng web hoặc các dịch vụ web.

1.2. Các mô hình dịch vụ của Cloud Computing

1.2.1. Tổng quan về các mô hình dịch vụ

Hiện tại có rất nhiều nhà cung cấp dịch vụ Cloud Computing cung cấp nhiều loại dịch vụ khác nhau. Tuy nhiên có ba loại dịch vụ Cloud Computing cơ bản là:

- ✓ Dịch vụ phần mềm (Software as a Service – SaaS).
- ✓ Dịch vụ cơ sở hạ tầng (Infrastructure as a Service – IaaS).
- ✓ Dịch vụ nền tảng (Platform as a Service – PaaS).



Hình 1.3 – Kiến trúc phân tầng dịch vụ của Cloud Computing.[4]

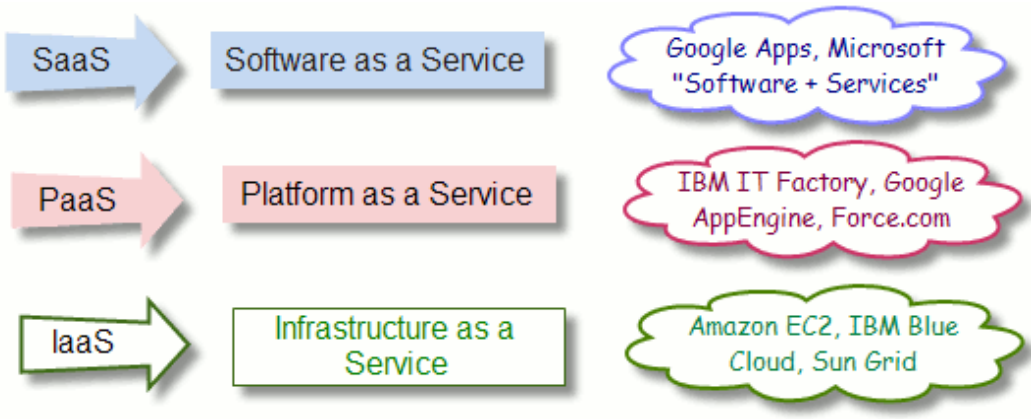
Tầng SaaS (Software as a Service) ở trên cùng cung cấp cho khách hàng một phần mềm dạng dịch vụ hoàn chỉnh, cho phép người dùng điều khiển từ xa các ứng dụng trong đám mây. Hiện nay SaaS được cung cấp rộng rãi bởi các công ty như SalesForce, 3Tera, Microsoft, Zoho.

Tầng IaaS (Infrastructure as a Service) cung cấp một dịch vụ bao gồm các máy tính ảo hóa và băng thông dành riêng cho lưu trữ và truy cập Internet.Nhà cung cấp IaaS thương mại nổi tiếng nhất là Amazon Elastic Compute Cloud (EC2).

Tầng PaaS (Platform as a Service) cung cấp một nền tảng (platform) phát triển ứng dụng cho khách hàng. Hiện nay nhà cung cấp PaaS nổi tiếng là Force.com của Salesforce.com, Google App Engine, Zoho. Thực chất tầng PaaS cũng chính là tầng IaaS, nhưng các máy chủ ở tầng này còn có thêm hệ điều hành và các chương trình nền tảng phục vụ cho các ứng dụng nhất định nào đó. Nói cách khác, PaaS chính là IaaS cộng thêm một phần mềm nền tảng được thiết kế riêng cho các ứng dụng chuyên biệt.

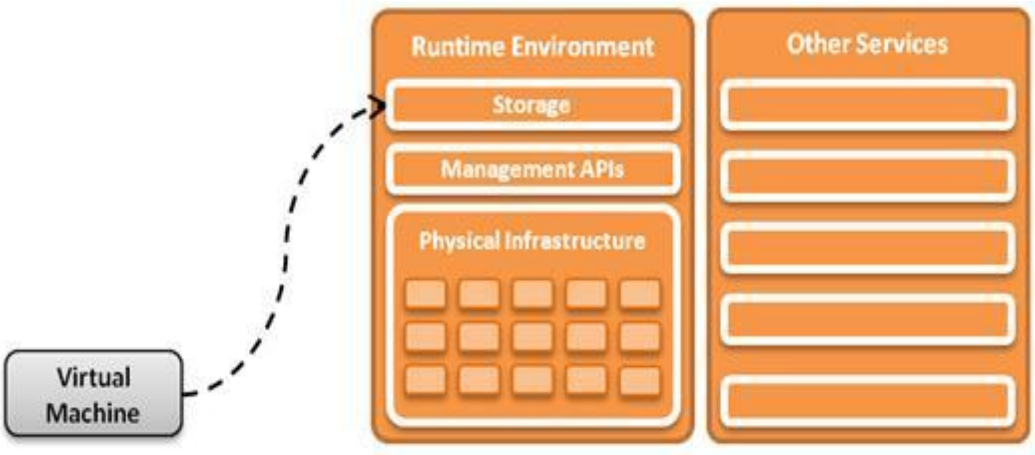
Tầng dSaaS (data-Storage-as-a-Service) là nơi chứa các máy chủ và không gian lưu trữ vật lý mà nhà cung cấp đã có sẵn.

Dựa trên các tầng dịch vụ, các nhà cung cấp lớn đã cho ra đời những dịch vụ mang tính Đám mây để phục vụ, giải quyết những nhu cầu cấp thiết hiện nay cho người sử dụng. Tiêu biểu như các nhà cung cấp lớn hàng đầu thế giới là Google, Microsoft, IBM, Amazon, Sun,… đang phát triển rất mạnh về các tầng dịch vụ Cloud Computing.



Hình 1.4 – Các hãng cung cấp dịch vụ Cloud Computing

1.2.2. Infrastructure as a Service – IaaS



Hình 1.5 – Infrastructure as a service.

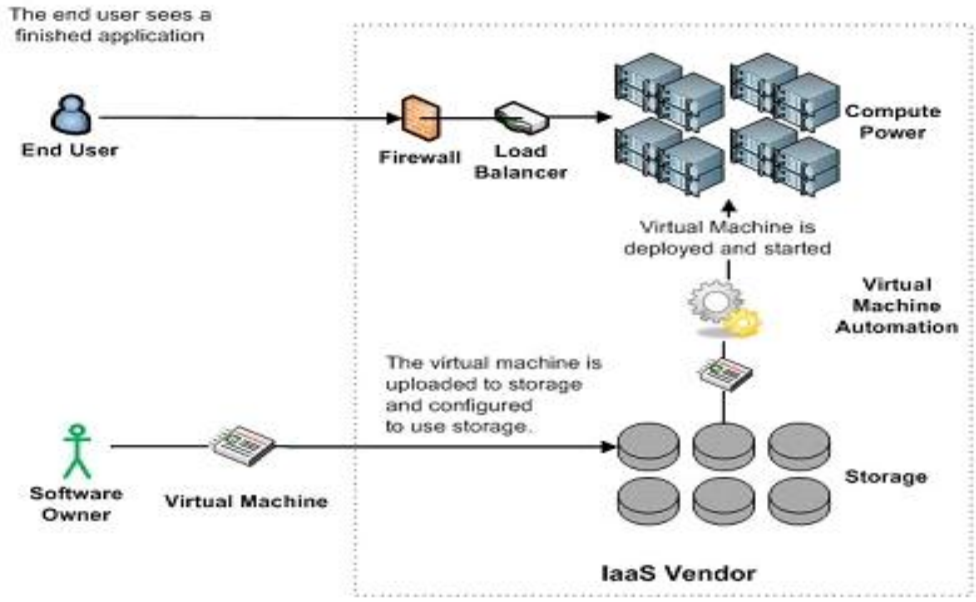
IaaS cung cấp một dịch vụ bao gồm các máy tính ảo hóa và băng thông dành riêng cho lưu trữ và truy cập Internet. Sever, router, hệ thống lưu trữ, các bộ chuyển đổi (switches), và những hệ thống khác được gom chung lại và luôn ở trạng thái sẵn sàng để đóng vai trò như một máy chủ ảo, đảm nhận các khối lượng công việc với khoảng cách từ các thành phần ứng dụng cho đến các ứng dụng điện toán cao cấp. Tầng này khác với PaaS ở chỗ: phần cứng ảo được cung cấp không kèm theo software stack. Đây là tầng thấp nhất trong các tầng dịch vụ của Cloud Computing.

Khách hàng sẽ cài hệ điều hành, triển khai ứng dụng và có thể kết nối các thành phần như tường lửa và bộ cân bằng tải. Nhà cung cấp dịch vụ sẽ quản lý cơ sở hạ tầng cơ bản bên dưới, khách hàng sẽ phải quản lý hệ điều hành, lưu trữ, các ứng dụng triển khai trên hệ thống, các kết nối giữa các thành phần.

Nhà cung cấp IaaS thương mại nổi tiếng nhất là Amazon Elastic Compute Cloud (Amazon EC2). Trong Amazon EC2, người dùng có thể chỉ định máy ảo (VM) đặc biệt của mình và triển khai các ứng dụng trên đó và chạy nó trên server. Người dùng chỉ phải trả tiền cho thời gian tính toán, dung lượng lưu trữ và băng thông mạng.

Ví dụ về các dịch vụ cơ sở hạ tầng bao gồm IBM Bluehouse của IBM, VMware, Amazon EC2(vendor là Amazon), Microsoft Azure Platform(vendor là Microsoft), Sun ParaScale Cloud Storage và nhiều hơn nữa…

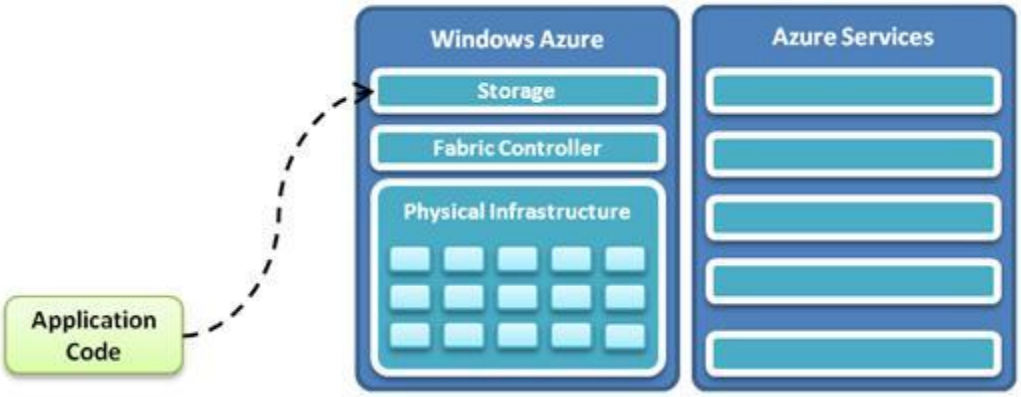
Cũng như với các dịch vụ nền tảng, sự ảo hóa là một phương pháp thường được sử dụng để tạo ra chế độ phân phối các nguồn tài nguyên theo yêu cầu. Nhà cung cấp IaaS sử dụng công nghệ ảo hóa để cung cấp sức mạnh điện toán. Vì vậy mỗi đơn vị triển khai là một máy ảo được xây dựng bởi chủ sở hữu phần mềm. Các kỹ thuật ảo hóa thường được sử dụng trong tầng này, nên có thể thấy rõ sự tiết kiệm chi phí do việc sử dụng nguồn lực hiệu quả mang lại. IaaS cũng rất mềm dẻo và có khả năng tùy biến theo kiến trúc cơ sở hạ tầng thay đổi. Ứng dụng được ảo hóa và được tải lên môi trường IaaS để chạy.



Hình 1.6 – Máy ảo được xây dựng và đưa lên môi trường IaaS.

Hình 1.6 ở trên minh họa làm thế nào một máy ảo được xây dựng cho một môi trường IaaS, được tải lên và cấu hình lại để các nhà cung cấp (IaaS Vendor) lưu trữ lại. Sau đó triển khai, phát triển, và cập nhật trong môi trường IaaS. Người dùng truy cập thông qua Internet và bắt đầu sử dụng.

1.2.3. Platform as a Service – PaaS

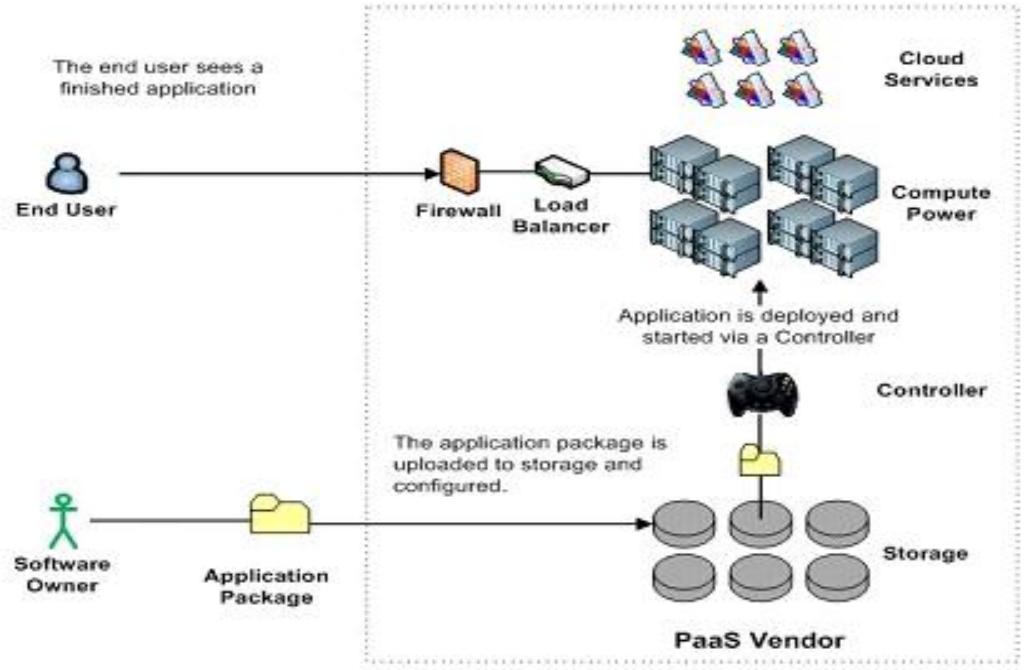


Hình 1.7 – Platform as a service.

Đây là tầng tiếp theo của các tầng dịch vụ. Trong tầng này, nhà cung cấp dịch vụ sẽ cung cấp một nền tảng (platform) cho khách hàng. Khách hàng sẽ tự phát triển ứng dụng của mình nhờ các công cụ và môi trường phát triển được cung cấp hoặc cài đặt các ứng dụng sẵn có trên nền platform đó. Khách hàng không cần phải quản lý hoặc kiểm soát các cơ sở hạ tầng bên dưới bao gồm cả mạng, máy chủ, hệ điều hành, lưu trữ, các công cụ, môi trường phát triển ứng dụng nhưng quản lý các ứng dụng mình cài đặt hoặc phát triển.

Nói cách khác, PaaS giống như IaaS nhưng gồm cả hệ điều hành và các dịch vụ cần thiết cho một ứng dụng chuyên biệt. Ví dụ PaaS ngoài server và lưu trữ ảo hóa cùng hệ điều hành đặc biệt và tập các ứng dụng (như một máy ảo) cùng các dịch vụ cần thiết như MySQL.

Môi trường PaaS cung cấp sức mạnh điện toán bằng việc cung cấp môi trường chạy cho ứng dụng. Đơn vị triển khai là các gói chứa mã nguồn ứng dụng hoặc phiên bản biên dịch của mã nguồn ứng dụng.



Hình 1.8 – Nền tảng ứng dụng được xây dựng và đưa lên môi trường PaaS.

Điều này có nghĩa là không phải cả máy ảo sẽ được xây dựng, cấu hình và tải lên mạng mà chỉ có các mã nguồn ứng dụng được tải lên mạng và sử dụng

Một số ví dụ điển hình về PaaS là Force.com của Salesforce.com, Google App Engine của nhà cung cấp Google, Yahoo Pipes của Yahoo hay Azure Service Platform của Microsoft…. PaaS có thể cung cấp cho mỗi giai đoạn phát triển phần mềm và thử nghiệm, hoặc họ có thể được chuyên về một khu vực cụ thể như quản lý nội dung. Ví dụ thương mại của PaaS bao gồm GAE (Google Apps Engine), phục vụ các ứng dụng trên cơ sở hạ tầng của Google. PaaS cung cấp mạnh mẽ để các ứng dụng trên máy chủ được triển khai, Google App Engine là một dịch vụ cho phép người dùng triển khai ứng dụng web của mình trên kiến trúc rất khả mở của Google. App Engine cung cấp một sandbox cho ứng dụng Python (các ngôn ngữ khác sẽ hỗ trợ sau) như là các API Python để lưu trữ và quản lý dữ liệu (dùng Google Query Language) bên cạnh các hỗ trợ về xác thực người dùng, thao tác hình ảnh và gửi email. Tuy nhiên, vẫn gặp hạn chế là khả năng mà nhà cung cấp Cloud cung cấp có thể không đáp ứng đủ.

Một ví dụ khác về PaaS là **10gen**, đó vừa là một nền tảng "đám mây" vừa là một gói phần mềm nguồn mở cho phép người dùng download để tạo ra "đám mây" của riêng mình. Software stack của **10gen** cũng giống như App Engine nhưng có vài điểm khác biệt: hỗ trợ các ngôn ngữ Java, Python, Ruby. Nền tảng của **10gen** cũng dùng khái niệm sandbox để cô lập các ứng dụng và cung cấp một môi trường đáng tin cậy trên nhiều máy tính (sử dụng Linux).

1.2.4. Software as a Service – SaaS

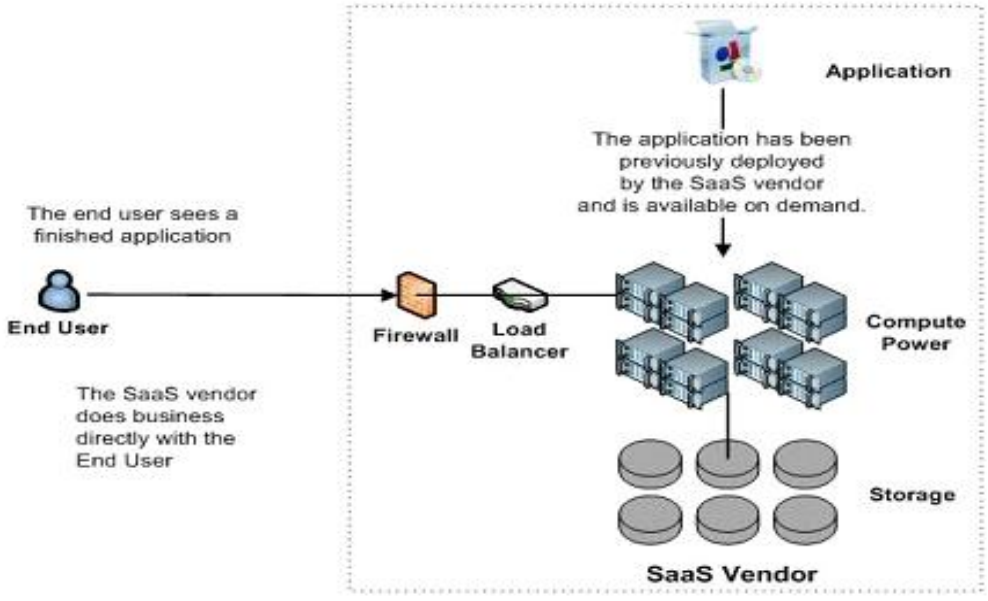
Đây là mô hình dịch vụ mà trong đó nhà cung cấp dịch vụ sẽ cung cấp cho khách hàng một phần mềm dạng dịch vụ hoàn chỉnh. Khách hàng chỉ cần lựa chọn ứng dụng phần mềm nào phù hợp với nhu cầu và chạy ứng dụng đó trên cơ sở hạ tầng Cloud. Mô hình này giải phóng người dùng khỏi việc quản lý hệ thống, cơ sở hạ tầng, hệ điều hành… tất cả sẽ do nhà cung cấp dịch vụ quản lý và kiểm soát để đảm bảo ứng dụng luôn sẵn sàng và hoạt động ổn định.

Các công ty hàng đầu trong ngành công nghiệp phần mềm đang dần dần di chuyển các ứng dụng cùng với các dữ liệu liên quan lên Internet và phân phối chúng thông qua SaaS. Google sử dụng nền tảng SaaS để cung cấp các ứng dụng web phục vụ liên lạc, cộng tác nhằm từng bước thay thế các ứng dụng tương tự chạy độc lập trên máy tính. Tương tự, Microsoft cũng

cung cấp cách phát triển của họ với dịch vụ cơ sở dữ liệu SaaS với tên mã là Microsoft Azure. SaaS được quảng bá rộng rãi bởi các công ty như SalesForce.com, 3Tera, Microsoft, Zoho và Amazon. Đấy là kết quả tất yếu của các dịch vụ kinh doanh tiên tiến.

Nói dễ hiểu hơn là: SaaS cho phép người sử dụng thuê một ứng dụng và chỉ trả tiền cho thời gian sử dụng. Một ví dụ tiêu biểu chính là Google Apps – Đây là dịch vụ của Google cung cấp các công cụ office cho người dùng Internet. Chỉ cần một máy tính kết nối Internet là ta đã có một bộ công cụ xử lý văn bản online, không cần cài đặt bất kỳ phần mềm nào khác ngoài web browser.

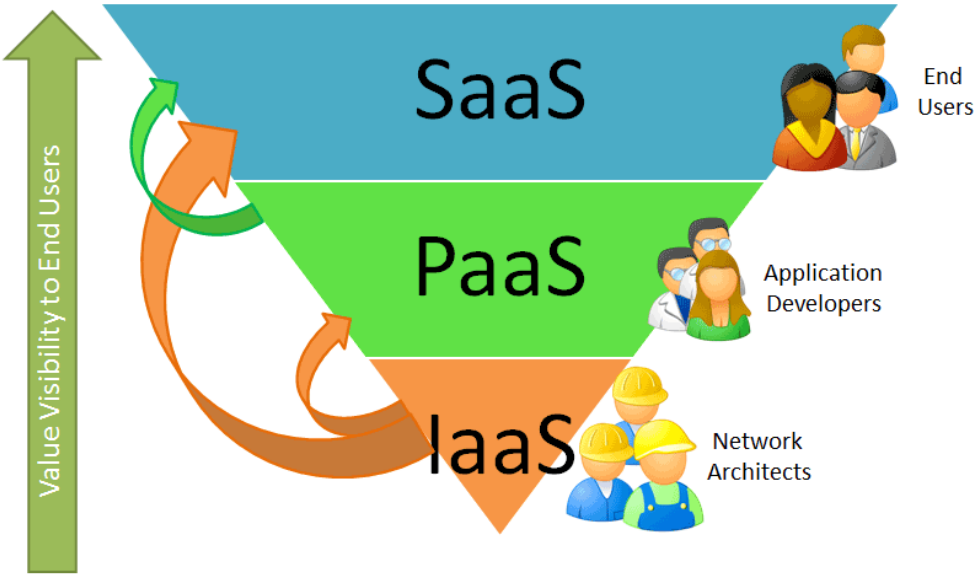
Góc độ khác về SaaS là việc sử dụng phần mềm chạy từ xa trên mạng. Phần mềm này có thể ở dạng Web services (các dịch vụ dùng bởi ứng dụng cục bộ) hay các ứng dụng từ xa mà có thể theo dõi kết quả thông qua trình duyệt web. Ví dụ tiêu biểu là Google Apps. Còn việc chạy ứng dụng từ xa thường dựa trên các Application server (là một software framework cung cấp các API – như quản lý giao dịch hay truy cập CSDL). Lấy ví dụ như Red Hat JBoss Application Server, Apache Geronimo, và IBM® WebSphere® Application Server.



Hình 1.9 – Ứng dụng được xây dựng trên môi trường SaaS.

Các ứng dụng được cung cấp qua mô hình SaaS làm lợi cho người tiêu dùng bằng cách giải phóng cho họ khỏi việc cài đặt và bảo trì phần mềm và các ứng dụng có thể được sử dụng thông qua các mô hình cấp phép có hỗ trợ trả tiền để sử dụng. Các nhà cung cấp SaaS sở hữu các ứng dụng được tải lên, chạy theo yêu cầu của khách hàng và cũng chịu trách nhiệm cho môi trường lưu trữ.

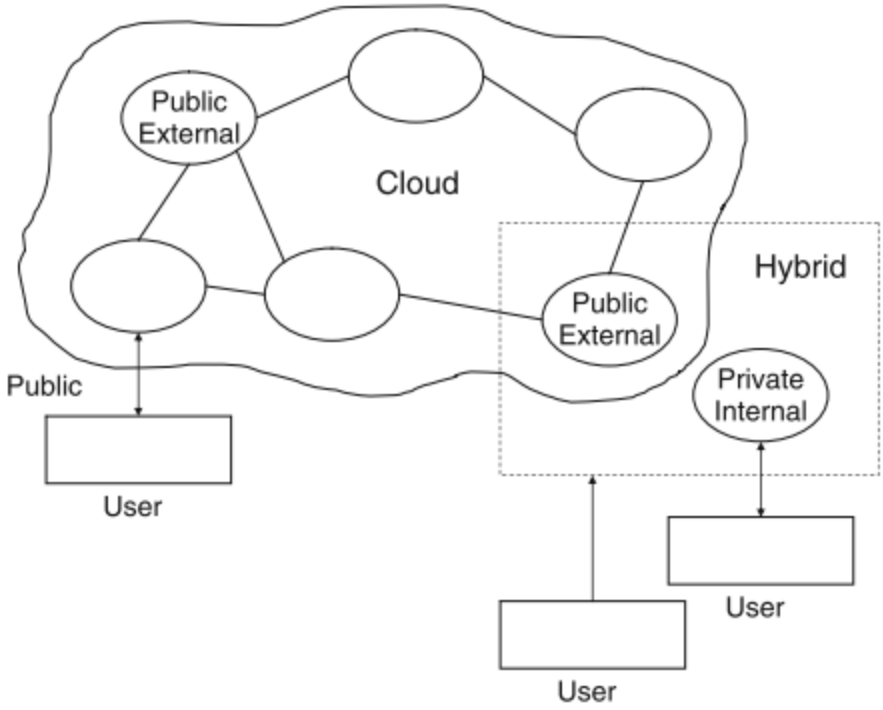
Giá trị của SaaS là những ứng dụng hoàn chỉnh có sẵn trên mạng Internet theo yêu cầu. Người sử dụng cuối cùng không cần giấy phép và hỗ trợ các phần mềm như trong mô hình truyền thống. Người dùng cuối cũng không cần mua hay cần hỗ trợ cơ sở hạ tầng mà ứng dụng chạy trên đó. Cơ cấu giá cho các dịch vụ SaaS thường là chi phí mỗi người sử dụng cho mỗi tháng.



Hình 1.10 – Mức độ người dùng trong các tầng dịch vụ.

1.3. Các mô hình triển khai Cloud Computing

Cho dù người dùng sử dụng loại mô hình dịch vụ nào đi nữa thì cũng có ba mô hình triển khai chính là: Public Cloud, Private Cloud và Hybrid Cloud.



Hình 1.11 – Các mô hình triển khai Cloud Computing.[4]

Public Cloud (hay External Cloud) là mô hình mà hạ tầng Cloud được một tổ chức sở hữu và cung cấp dịch vụ rộng rãi cho tất cả các khách hàng thông qua hạ tầng mạng Internet hoặc các mạng công cộng diện rộng.

Private Cloud (hay Internal Cloud) là mô hình mà cơ sở hạ tầng và các dịch vụ được xây dựng để phục vụ cho một tổ chức (doanh nghiệp) duy nhất. Doanh nghiệp đó sẽ hoàn toàn kiểm soát được dữ liệu, độ an toàn, và chất lượng của dịch vụ.

Hybrid Cloud là sự kết hợp của Public Cloud và Private Cloud. Sử dụng các dịch vụ Public Cloud, đồng thời, doanh nghiệp sẽ giữ lại các chức năng nghiệp vụ và dữ liệu tối quan trọng trong tầm kiểm soát (Private Cloud).

1.4. Lợi ích của Cloud Computing

1.4.1. Tính linh động

Người dùng có thể tự do lựa chọn các dịch vụ phù hợp với nhu cầu của mình, cũng như có thể bỏ bớt những thành phần mà mình không muốn. (Thay vì phải bỏ ra hàng trăm USD cho

phần mềm, ta có thể mua riêng lẻ từng phần hoặc chỉ trả một khoản phí rất nhỏ mỗi khi sử dụng một phần nào đó của nó).

Người dùng sẽ không còn bị bó hẹp với một thiết bị hay một vị trí cụ thể nào nữa. Với Cloud Computing, phần mềm, dữ liệu có thể được truy cập và sử dụng từ bất kì đâu, trên bất kì thiết bị nào mà không cần phải quan tâm đến giới hạn phần cứng cũng như địa lý. (Người sử có thể chơi những game 3D có đồ họa cực kì bát mắt trên iPad hoặc iPhone mà không cần quan tâm đến cấu hình của nó).

Với Cloud Computing, doanh nghiệp sẽ chuyển hầu hết trách nhiệm về kiểm soát hệ thống, quản lý hạ tầng, bảo mật, đảm bảo chất lượng dịch vụ… cho nhà cung cấp dịch vụ. Khi đó doanh nghiệp sẽ giảm rất nhiều chi phí và chỉ tập trung vào nhiệm vụ chính là kinh doanh, không phải bận tâm nhiều đến việc quản lý, kiểm soát hệ thống.

Khách hàng có thể lựa chọn nhà cung cấp dịch vụ nào đáp ứng tốt nhất cho nhu cầu của mình với giá cả và chất lượng dịch vụ hợp lý nhất.

1.4.2. Chi phí thấp

Người dùng không chỉ giảm bớt chi phí bản quyền mà còn giảm phần lớn chi phí cho việc mua và bảo dưỡng máy chủ. Việc tập hợp ứng dụng của nhiều tổ chức lại một chỗ sẽ giúp giảm chi phí đầu tư ban đầu, cũng như tăng hiệu năng sử dụng các thiết bị này một cách tối đa.

Khi doanh nghiệp sử dụng dịch vụ Cloud Computing, đặc biệt là Public Cloud, thì chi phí đầu tư ban đầu rất thấp. Nếu doanh nghiệp tự xây dựng một hệ thống quy mô lớn cho mình thì chi phí đầu tư rất lớn (mua phần cứng, quản lý nguồn điện, hệ thống làm mát, nguồn nhân lực vận hành hệ thống…). Và các dự án tốn kém như vậy thường cần rất nhiều thời gian để được phê chuẩn. Việc xây dựng một hệ thống như vậy cũng đòi hỏi nhiều thời gian. Giờ đây, nhờ Cloud Computing, mọi thứ đã được nhà cung cấp dịch vụ chuẩn bị sẵn sàng, doanh nghiệp chỉ cần thuê là có thể sử dụng được ngay mà không phải tốn chi phí đầu tư ban đầu.

Mọi phần mềm đều nằm trên server. Lúc này, người dùng sẽ không cần lo lắng cập nhật hay sửa lỗi phần mềm nữa. Và các lập trình viên cũng dễ dàng hơn trong việc cài đặt, nâng cấp ứng dụng của mình. Do đó đỡ tốn chi phí trong việc bảo trì.

1.4.3. Tăng cường độ tin cậy

Dữ liệu trong mô hình Cloud Computing được lưu trữ 1 cách phân tán tại nhiều cụm máy chủ tại nhiều vị trí khác nhau. Điều này giúp tăng độ tin cậy, độ an toàn của dữ liệu mỗi khi có sự cố hoặc thảm họa xảy ra.

Việc tập trung dữ liệu từ nhiều nguồn khác nhau sẽ giúp các chuyên gia bảo mật tăng cường khả năng bảo vệ dữ liệu của người dùng, cũng như giảm thiểu rủi ro bị ăn cắp toàn bộ dữ liệu. (Dữ liệu được đặt tại các 6 máy chủ khác nhau trong trường hợp hacker tấn công, người dùng cũng sẽ chỉ bị lộ 1/6. Đây là một cách chia sẻ rủi ro giữa các tổ chức với nhau).

1.4.4. Tăng lượng tài nguyên tính toán

Một trong những câu hỏi đau đầu của việc đầu tư tài nguyên (ví dụ máy chủ) là bao lâu thì nó sẽ hết khấu hao, tôi đầu tư như thế có lãi hay không, có bị outdate về công nghệ hay không… Khi sử dụng tài nguyên trên Cloud Computing thì người dùng sẽ không còn phải quan tâm tới điều này nữa.

1.4.5. Sử dụng tài nguyên hiệu quả hơn

Nhờ khả năng co giãn (elasticity) nên tài nguyên luôn được sử dụng một cách hợp lý nhất, theo đúng nhu cầu của khách hàng, không bị lãng phí hay dư thừa. Đối với nhà cung cấp dịch vụ, công nghệ ảo hóa giúp cho việc khai thác tài nguyên vật lý hiệu quả hơn, phục vụ nhiều khách hàng hơn.

1.5. Khó khăn của Cloud Computing

1.5.1. Data lock-in

Hiện nay các phần mềm đã được cải thiện khả năng tương tác giữa các nền tảng khác nhau, nhưng các hàm API của Cloud Computing vẫn còn mang tính độc quyền, chưa được chuẩn hóa. Do đó khi một khách hàng viết một ứng dụng trên một nền tảng do một nhà cung cấp dịch vụ thì ứng dụng đó sẽ chỉ được sử dụng trên các dịch đó, nếu đem ứng dụng đó qua một nền tảng khác do một nhà cung cấp dịch vụ khác cung cấp thì có thể không chạy được. Điều này dẫn đến người sử dụng phụ thuộc vào nhà cung cấp dịch vụ. Ngoài ra nhà cung cấp dịch vụ cũng sẽ tập trung hơn để phát triển dịch vụ của mình để phục vụ nhu cầu người sử dụng tốt hơn.

Ngoài ra việc sử dụng các dịch vụ của Cloud Computing cũng gây ra một vấn đề. Khi dữ liệu của người sử dụng dịch vụ lưu trữ trên hệ thống của nhà cung cấp dịch vụ thì không có gì đảm bảo cho người sử dụng là dữ liệu của mình sẽ an toàn, không bị rò rỉ ra bên ngoài. Hiện nay, về mặt kỹ thuật thì vẫn chưa có cách nào hiệu quả để giải quyết vấn đề trên. Điều này dẫn đến việc thực hiện hay sử dụng thường xảy ra đối với các nhà cung cấp dịch vụ có uy tín.

Ví dụ: tháng 8 năm 2008 khi dịch vụ lưu trữ dữ liệu trực tuyến của Linkup bị hỏng, sau khi phục hồi lại hệ thống thì phát hiện ra mất 45% dữ liệu của khách hàng. Sau sự cố này thì uy tín và doanh thu của công ty hạ xuống. Khoảng 20.000 người dùng dịch vụ của Linkup đã từ bỏ nhà cung cấp này để tìm đến một nhà cung cấp dịch vụ mới. Và sau đó dịch vụ này phải dựa trên một dịch vụ lưu trữ trực tuyến khác để tồn tại là Nirvanix. Hiện nay hai công ty này đã kết hợp với nhau trong việc cung cấp dịch vụ lưu trữ trực tuyến.

Từ ví dụ trên ta thấy nếu các các nhà cung cấp dịch vụ có cơ chế chuẩn hóa các API thì các nhà phát triển dịch vụ có thể triển khai dịch vụ trên nhiều nhà cung cấp dịch vụ. Khi đó nếu một nhà cung cấp dịch vụ nào đó bị hỏng, thì dữ liệu của các nhà phát triển không mất hết mà có thể nằm đâu đó trên không gian lưu trữ của các nhà cung cấp dịch vụ khác. Nếu như cách này được các nhà cung cấp dịch vụ thể hiện thì sẽ dẫn đến cuộc cạnh tranh về giá cung cấp. Hai tham số ảnh hưởng đến việc lựa chọn một dịch vụ lúc đó là:

- ✓ Tham số thứ nhất – Chất lượng dịch vụ tương xứng với giá mà người sử dụng trả cho nhà cung cấp dịch vụ. Hiện nay có một số nhà cung cấp dịch vụ có giá cao gấp 10 lần so với các nhà cung cấp khác, nhưng nó có chất lượng tốt cộng thêm các tính năng hỗ trợ người dùng như: tính dễ dùng, một số tính năng phụ khác…
- ✓ Tham số thứ hai, ngoài việc giảm nhẹ data lock – in, thì việc chuẩn hóa các API sẽ dẫn đến một mô hình mới: cơ sở hạ tầng cùng phần mềm có thể chạy trên Private Cloud hay Public Cloud.

1.5.2. Bảo mật

Việc để dữ liệu nhạy cảm của các công ty lên Cloud sẽ làm cho khả năng bị người khác truy xuất nhiều hơn. Và vấn đề này đang là một thách thức thực sự đối với công nghệ hiện đại

trong việc việc bảo mật dữ liệu. Hiện nay có một giải pháp là những người dùng dịch vụ phải mã hóa dữ liệu trước khi đưa lên hệ thống Cloud, và khi muốn sử dụng dữ liệu này thì phải thực hiện công việc giải mã này ở máy local. Mô hình này đã có những thành công nhất định đối với việc sử dụng TC3, đây là công ty về chăm sóc sức khỏe, dữ liệu của họ thường là những thông tin nhạy cảm (chủ yếu là các thông tin bệnh án của các bệnh nhân).

Ngoài ra, còn có thể thêm vào việc ghi nhận lại các thộng tin mà hệ thống đã làm, và sử dụng các hệ điều hành ảo khi cung cấp dịch vụ IaaS sẽ làm cho ứng dụng của mình khó bị tấn công hơn.

Việc bảo mật dữ liệu ngoài các vấn đề về kỹ thuật thì nó còn liên quan đến các vấn đề khác như con người, các đạo luật… Việc sử dụng các luật bảo vệ người sử dụng dịch vụ cloud khi họ đưa dữ liệu của mình lưu trữ trên Cloud, thì các nhà cung cấp dịch vụ phải bảo đảm dữ liệu của khách hàng không bị rò rỉ ra bên ngoài.

Thêm vào đó các nhà cung cấp dịch vụ SaaS còn cung cấp cho người dùng cơ chế lựa chọn vị trí mà người dùng muốn lưu trữ dữ liệu cũa mình. Ví dụ: Amazon cung cấp dịch S3, khi sử dụng dịch vụ này người dùng có thể lưu trữ dữ liệu vật lý của mình ở châu Âu hay ở Mỹ.

1.5.3. Truyền tải dữ liệu

Đối với các ứng dụng, mà lúc đầu ứng dụng bắt đầu chạy thường thì dữ liệu ít, và càng về sau thì dữ liệu càng nhiều. Và ngoài ra có thể có ứng dụng chạy trên Cloud mà dữ liệu có thể lưu ở các vị trí khác nhau. Khi lúc ứng dụng này chạy có thể dẫn đến việc vận chuyển giữa các dữ liệu (việc vận chuyển dữ liệu giữa các data center). Hiện nay giá của việc vận chuyển dữ liệu là 100$ đến 150$ cho mỗi terabyte vận chuyển. Khi ứng dụng chạy càng về sau thì chi phí này có thể càng tăng lên, làm cho chi phí truyền tải dữ liệu là một vấn đề quan trọng trong chi phí vận hành ứng dụng. Và vấn đề này cũng đã được các nhà cung cấp dịch vụ Cloud và những người sử dụng Cloud suy nghĩ đến. Và vấn đề này đã được giải quyết trong dịch vụ Cloudfront mà công ty Amazon đã phát triển.

Chương 2 – TÌM HIỂU AMAZON WEB SERVICES

2.1. Tổng quan

Amazon Web Services là tập hợp các dịch vụ cung cấp cho người lập trình khả năng truy cập tới hạ tầng kiến trúc tính toán của Amazon. Các máy tính có nền tảng cấu hình tốt được thiết kế và cài đặt qua nhiều năm của Amazon cho phép bất cứ ai sử dụng Internet cũng có thể truy cập tới. Amazon cung cấp một số dịch vụ Web nhưng báo cáo này chỉ tập trung vào các dịch vụ khối hợp nhất (building-block) cơ bản. Đó là những dịch vụ đáp ứng được một số yêu cầu cốt lõi của hầu hết các hệ thống như: lưu trữ, tính toán, truyền thông điệp và tập hợp dữ liệu.

Người dùng có thể thiết kế các ứng dụng phức tạp gồm nhiều phần khác nhau bằng cách sử dụng các chức năng phân tầng với các dịch vụ hiệu quả, đáng tin cậy do Amazon cung cấp. Các dịch vụ Web này tồn tại bên trong đám mây (phía ngoài của môi trường người dùng) và có tính sẵn sàng cao.

Người dùng sẽ chi trả dựa trên những gì đã sử dụng mà không cần phải trả trước các chi phí và không cần phải có vốn đầu tư ban đầu. Người dùng cũng không phải mất chi phí cho việc bảo trì phần cứng bởi việc này đã được Amazon đảm nhận.

Sự tự do từ việc hạn chế một lượng lớn vốn đầu tư cơ sở hạ tầng và chi phí bảo trì tạo cơ hội lớn cho sự đổi mới. Người dùng có thể tập trung vào các ý tưởng kinh doanh thay vì quan tâm đến sự hao mòn của một lượng lớn các máy chủ mình đang có (ví dụ như lo lắng về khả năng hết dung lượng đĩa,…). Theo ước tính của Amazon, các doanh nghiệp sử dụng hơn 70% thời gian của họ cho việc xây dựng và bảo trì cơ sở hạ tầng, trong khi chỉ sử dụng 30% thời gian để làm việc với các ý tưởng thực sự để tạo ra lợi nhuận cho doanh nghiệp. Amazon xử lý về các vấn đề chi tiết, cơ bản của phần cứng và cơ sở hạ tầng – và làm thế nào để có được hiệu quả cao – trong khi người dùng chỉ cần tập trung tới việc mang các ý tưởng của mình vào công việc và cuộc sống.

Các thành phần chính của cơ sở hạ tầng dạng dịch vụ Web này gồm:

Lưu trữ (Storage)

Mọi người đều cần phải lưu trữ - các tệp, các tài liệu, các dữ liệu tải về hoặc các bản sao lưu. Người dùng có thể tiến hành lưu trữ bất kỳ các ứng dụng cần thiết của mình trong Amazon Simple Storage Service (S3) và được hưởng các lợi ích của dịch vụ này như: khả năng mở rộng, sự tin cậy, tính sẵn sàng cao với mức chi phí thấp cho việc lưu trữ.

Tính toán (Computing)

Amazon Elastic Compute Cloud (EC2) cung cấp khả năng mở rộng tài nguyên tính toán của người dùng nhiều lên hoặc ít xuống dựa trên nhu cầu và cung cấp dịch vụ cho thuê máy chủ ảo một cách nhanh chóng, dễ dàng.

Gửi thông điệp (Messaging)

AWS có thể thực hiện tách riêng các thành phần ứng dụng của người dùng bằng cách sử dụng khả năng truyền thông điệp không giới hạn được cung cấp bởi Amazon Simple Queue Service (SQS).

Tập hợp dữ liệu (Datasets)

Amazon SimpleDB (SDB) cung cấp khả năng mở rộng, lập chỉ mục, khả năng lưu trữ mà không cần bảo trì, cùng với việc thực hiện xử lý và truy vấn với tập hợp dữ liệu.

Người dùng có thể sử dụng kết hợp các dịch vụ khi cần thiết; các dịch vụ này được thiết kế để có thể làm việc chung với nhau. Do người dùng đang làm việc trong môi trường của Amazon nên các thao tác liên lạc của các dịch vụ này sẽ được thực hiện một cách nhanh chóng.

Các doanh nghiệp có thể xây dựng các ứng dụng có khả năng mở rộng và đáng tin cậy bằng cách phân nhánh vào cơ sở hạ tầng ảo. Việc này tiêu tốn ít chi phí hơn nhiều so với cách tiếp cận dựa trên nền ứng dụng máy chủ truyền thống (thường yêu cầu một lượng lớn máy chủ).

Amazon Web Services có hai hình thức hỗ trợ cho người dùng:

- ✓ Hỗ trợ từ diễn đàn của Amazon.
- ✓ Hỗ trợ qua điện thoại. Cách yêu cầu hỗ trợ này đảm bảo hơn.

Amazon cung cấp các giao diện chuẩn dựa trên SOAP và REST để tương tác với từng dịch vụ. Các thư viện phát triển hoặc là từ Amazon hoặc từ một trong các ngôn ngữ cho phép, như Ruby, Python, Java™, Erlang và PHP, để thực hiện trao đổi với các dịch vụ này. Các

công cụ dòng lệnh cũng có thể để thực hiện quản lý tài nguyên tính toán trên EC2. Giao diện REST khá dễ sử dụng, người dùng có thể sử dụng chương trình bên phía máy khách được viết bằng bất cứ ngôn ngữ nào dưới dạng giao thức HTTP để thực hiện gửi yêu cầu tới các dịch vụ Web.

2.2. Những ứng dụng thành công của Amazon Web Services

Animoto, chương trình tạo video trình diễn trực tuyến cần một lượng lớn các máy tính tính toán lớn để thực hiện xử lý video, gần đây đã chống cự thành công một lưu lượng truy cập web cực lớn (lưu lượng này có thể làm tiêu tan hầu hết các hệ thống của các công ty khác) bằng cách tăng công suất xử lý nhanh sử dụng EC2. Tại một thời điểm họ đã sử dụng hơn 3,500 máy ảo chạy cùng lúc với nhau.

SmugMug, là ứng dụng lưu trữ ảnh trực tuyến, cho phép lưu trữ hơn một nửa petabyte dữ liệu trong S3, chi phí dịch vụ và lưu trữ ước tính có thể tiết kiệm được tới 1 triệu đôla. Đây là khách hàng lớn của dịch vụ Elastic Compute Cloud (EC2).

37Signals, là phần mềm phổ biến để quản lý dự án trực tuyến, sử dụng S3 để lưu trữ.

Tạp chí **New York Times** sử dụng sức mạnh của EC2 để xử lý nhiều terabyte dữ liệu lưu trữ bằng cách dùng hàng trăm máy ảo EC2 trong 36 giờ.

2.3. Tính toán co giãn với Amazon EC2

Amazon EC2 là dịch vụ Web cho phép người dùng gửi các yêu cầu thiết lập máy ảo trong vòng một vài phút và dễ dàng thay đổi dung lượng lưu trữ lên hoặc xuống tùy theo nhu cầu. Người dùng chỉ cần trả chi phí cho khoảng thời gian đã sử dụng. Nếu muốn tăng tốc độ tính toán lên, người dùng có thể nhanh chóng khởi tạo các máy ảo và sau đó tắt chúng đi khi nhu cầu giảm xuống.

Những máy ảo chạy trên Linux® và có thể chạy bất kỳ ứng dụng hoặc phần mềm nào người dùng muốn. Người dùng có thể điều khiển từng máy ảo cụ thể. Môi trường của EC2 được xây dựng ở tầng trên cùng của hệ thống mã nguồn mở Xen hypervisor (phát triển tại trường Đại học Cambridge). Amazon cho phép người dùng tạo ra các tệp Amazon Machine Images (AMIs), hoạt động với vai trò làm khuôn mẫu cho các máy ảo của người dùng. Việc truy cập vào các máy ảo được kiểm soát bằng việc xác định các quyền cho phép. Người dùng cũng có

thể làm bất cứ việc gì mình muốn với các máy ảo tuy chỉ có một hạn chế duy nhất là yêu cầu cần phải sử dụng nền tảng hệ điều hành Linux. Thời gian gần đây, Amazon đã nhận được sự hỗ trợ từ Open Solaris, đồng thời có quan hệ đối tác với Sun Microsystem, nhưng phần lớn các bản thương mại hay miễn phí của các máy ảo được thiết lập cho EC2 đều dựa trên nền tảng Linux.

Amazon EC2 thực hiện cung cấp dịch vụ dựa trên nền tảng web. Điều này giúp cho việc thay đổi quy mô tài nguyên tính toán của người dùng tăng lên hoặc giảm xuống được thực hiện dễ dàng. Người dùng hoàn toàn kiểm soát được môi trường tính toán hoạt động trong trung tâm dữ liệu của Amazon. Amazon cung cấp năm kiểu của máy chủ; người dùng có thể chọn lựa một trong các loại này sao cho phù hợp với ứng dụng của mình. Các máy chủ cung cấp từ loại đơn lõi x86 đến loại tám lõi x86_64. Người dùng có thể thiết lập các máy ảo trong một hoặc nhiều vùng địa lý khác nhau để đảm bảo hệ thống hoạt động ổn định. Amazon cũng đưa ra khái niệm IP động để cấp IP cho các máy ảo của người dùng.

2.4. Lưu trữ với Amazon S3

Amazon Simple Storage Service (S3) cung cấp các giao diện dịch vụ Web cho việc lưu trữ và khôi phục dữ liệu. Dữ liệu có thể ở bất kỳ dạng nào và được lưu trữ, truy cập đến từ bất kỳ vị trí nào thông qua Internet. Người dùng có thể lưu trữ không giới hạn một lượng lớn các đối tượng trong S3 với kích thước của mỗi đối tượng trong khoảng từ 1 byte tới 5 GB. Vị trí máy chủ lưu trữ có thể ở Hoa Kỳ hoặc ở các nước thuộc Liên Minh Châu Âu. Người dùng có thể chọn vị trí lưu trữ cho các đối tượng của mình khi người dùng tạo ra buckets (tương tự như khái niệm của thư mục trong hệ thống xử lý của người dùng).

Sự hạn chế truy cập có thể được xác định cho từng đối tượng người dùng lưu trữ trong S3, và các đối tượng này có thể được truy cập với các yêu cầu HTTP đơn giản. Thậm chí người dùng có thể tạo ra các đối tượng để tải về bằng cách sử dụng giao thức BitTorrent.

S3 giải phóng hoàn toàn cho người dùng về các vấn đề không gian lưu trữ, truy cập vào dữ liệu, hoặc bảo vệ dữ liệu. Người dùng thậm chí không phải quan tâm tới chi phí của việc bảo trì các máy chủ lưu trữ.

Amazon đảm bảo ở mức độ cao khả năng lưu trữ các tệp của người dùng, vì thế các tệp luôn luôn sẵn sàng bất kỳ khi nào người dùng cần đến chúng. Các thỏa thuận cấp độ dịch vụ được cung cấp bởi Amazon cho S3 lên tới 99,9% thời gian chạy máy hàng tháng.

2.5. Khả năng truyền thông điệp tin cậy của Amazon Simple Queue Service

Amazon Simple Queue Service (SQS) cung cấp khả năng truy cập tới hạ tầng kiến trúc gửi thông điệp của Amazon. Người dùng có thể thực hiện gửi và nhận các thông điệp từ bất cứ nơi nào bằng cách sử dụng các yêu cầu HTTP đơn giản dựa trên REST. Không cần phải cài đặt gì, không cần phải thực hiện cấu hình, người dùng có thể tạo ra một lượng không giới hạn của các hàng đợi và gửi một lượng không giới hạn các tin nhắn. Các tin nhắn được lưu trữ bởi Amazon thông qua nhiều máy chủ và các trung tâm dữ liệu để đảm bảo sự tin cậy cho hệ thống nhắn tin. Mỗi một tin nhắn có thể chứa lên tới 8KB dữ liệu văn bản. Chỉ sử dụng các ký tự dạng Unicode với nguyên tắc như sau:

#x9 | #xA | #xD | [#x20 tới #xD7FF] | [#xE000 tới #xFFFD] | [#x10000 tới #x10FFFF].

Mỗi một hàng đợi có thể sẽ phải xác định khoảng thời gian timeout (hạn thời gian), mục đích là để điều khiển việc truy cập tới hàng đợi từ nhiều người đọc khác nhau. Mỗi một ứng dụng đọc một thông điệp từ hàng đợi, thông điệp này sẽ không được đọc bởi bất kỳ một người đọc khác cho đến khi khoảng thời gian timeout kết thúc. Thông điệp sẽ được lặp lại trong hàng đợi sau một khoảng thời gian timeout được xác định, sau đó nó được xử lý bởi một bộ xử lý đọc khác.

SQS tích hợp rất tốt với các dịch vụ Amazon Web Services khác. Nó cung cấp cách thức xây dựng hệ thống riêng biệt giúp cho các máy ảo EC2 của người dùng có thể trao đổi, tương tác với nhau bằng cách gửi thông điệp tới SQS và phối hợp các luồng làm việc. Người dùng cũng có thể sử dụng các hàng đợi để xây dựng hệ thống hạ tầng máy ảo EC2 có khả năng tự phục hồi, tự động mở rộng cho các ứng dụng của mình. Người dùng có thể bảo đảm các thông điệp trong hàng đợi của mình tránh bị truy cập trái phép bằng cách sử dụng cơ chế xác thực được cung cấp bởi SQS.

2.6. Xử lý tập hợp dữ liệu với Amazon SimpleDB

Amazon SimpleDB (SDB) là dịch vụ Web cho phép lưu trữ, xử lý và truy vấn tập hợp dữ liệu có cấu trúc. Ở đây không phải là một cơ sở dữ liệu quan hệ theo cách tiếp cận truyền thống mà ở mức độ cao hơn dưới dạng các sơ đồ, với dữ liệu ít cấu trúc lưu trữ trong các đám mây và trong đó người dùng có thể sử dụng để lưu trữ và khôi phục các giá trị khóa. Mỗi một tập hợp các giá trị khóa cần phải có một tên mục (item) duy nhất; các mục được phân chia vào từng miền khác nhau. Mỗi một mục có thể lưu giữ lên tới 256 cặp giá trị khóa của dữ liệu. Người dùng có thể thực thi các truy vấn dựa trên tập dữ liệu của người dùng trong từng miền khác nhau. Các truy vấn dựa theo từng miền không được hỗ trợ bởi SDB.

SDB rất đơn giản để sử dụng và cung cấp hầu hết các chức năng của cơ sở dữ liệu quan hệ. Việc bảo trì cũng đơn giản hơn nhiều so với cơ sở dữ liệu truyền thống bởi vì không cần phải cài đặt hoặc định dạng. Amazon đảm nhận khá nhiều công việc liên quan đến quản trị. Dữ liệu được tự động đánh chỉ mục bởi Amazon và có sẵn cho người dùng tại bất kỳ thời điểm nào từ bất kỳ nơi nào. Một lợi thế quan trọng của các lược đồ trong SDB là khả năng chèn dữ liệu vào khi đang hoạt động và thêm các cột hoặc các khóa động.

SDB là một phần của cơ sở hạ tầng Amazon, và khả năng mở rộng được thực hiện một cách tự động đối với người dùng tùy từng tình huống. Người dùng có thể tự do tập trung vào những việc khác quan trọng hơn. Và đương nhiên, người dùng chỉ phải trả chi phí cho tập dữ liệu tài nguyên mà mình sử dụng.

2.7. Khả năng mở rộng kiến trúc

Amazon Web Services có thể giúp người dùng kiến tạo hệ thống mở rộng bằng cách cung cấp:

Sự tin cậy (Reliability)

Các dịch vụ hoạt động trong trung tâm dữ liệu với tính sẵn sàng cao và được kiểm tra qua thực tế.

Bảo mật (Security)

Cơ chế bảo mật và xác thực cơ bản đã có sẵn, người dùng có thể sử dụng chúng khi cần và áp dụng cho từng tầng ứng dụng riêng ở các mức trên của dịch vụ.

Các lợi ích về giá cả (Cost benefits)

Không hề có chi phí cố định hoặc các chi phí về bảo trì. Người dùngchỉ phải chi trả các dịch vụ đã sử dụng, và hoàn toàn có khả năng thay đổi nguồn tài nguyên khi cần thiết.

Dễ dàng phát triển (Ease of development)

Các hàm APIs đơn giản cho phép người dùng khai thác được sức mạnh của toàn bộ cơ sở hạ tầng ảo cũng như các thư viện có sẵn trong hầu hết các ngôn ngữ lập trình được sử dụng rộng rãi.

Tính mềm dẻo (Elasticity)

Quy mô các nguồn lực máy tính của người dùng thay đổi tăng lên hoặc giảm xuống dựa trên nhu cầu. Người dùng có thể thay đổi sử dụng một hoặc nhiều máy một cách nhanh chóng để phục vụ nhu cầu ứng dụng của mình.

Sự gắn kết (Cohensiveness)

Có bốn khối dịch vụ cơ bản (Lưu trữ, Tính toán, Truyền thông điệp, và Tập hợp dữ liệu) được thiết kế từ nhóm phát triển để làm việc hiệu quả với nhau, và cung cấp giải pháp hoàn thiện trên một lượng ứng dụng tên miền.

Cộng đồng (Community)

Cộng đồng người sử dụng năng động đang chi phối sự phổ cập của các dịch vụ web và đang tạo ra các ứng dụng độc nhất được xây dựng trên cơ sở hạ tầng này.

Chương 3 – TÌM HIỂU EUCALYPTUS

3.1. Tổng quan về Eucalyptus

Eucalyptus – **E**lastic **U**tility **C**omputing **A**rchitecture **L**inking **Y**our **P**rograms **T**o **U**seful **S**ystems - là phần mềm mã nguồn mở dạng IaaS dùng để thiết lập và quản lý các tài nguyên ảo dựa trên hệ thống máy chủ và hệ thống cung cấp dịch vụ có sẵn.

Eucalyptus do một nhóm nghiên cứu của đại học California tại Santa Barbara viết ra bằng ngôn ngữ C và Java.

Eucalyptus hiện đang có 2 phiên bản:

- ✓ Phiên bản mã nguồn mở (open-source) - *Eucalyptus 2.0.3*.
- ✓ Phiên bản thương mại dành cho doanh nghiệp - *Eucalyptus Enterprise Edition 2.0.0* - được phát triển dựa trên phiên bản mã nguồn mở.

Bài báo cáo này chỉ đề cập đến phiên bản mã nguồn mở - Eucalyptus 2.0.3.

Eucalyptus có tính co dãn, được thiết kế gồm nhiều module, nhiều thành phần tương hợp với hệ thống Amazon Web Service (AWS). Eucalyptus cho phép triển khai hệ thống Private Cloud mà không đòi hỏi phải có những linh kiện phần cứng đặc biệt hay phải thực hiện các thao tác cấu hình phức tạp trước đó.

Eucalyptus có tính khả chuyển cao - có thể được cài đặt trên nhiều bản phân phối của hệ điều hành Linux như: Ubuntu, Red Hat Enterprise Linux (RHEL), OpenSuse, Debian, Fedora, và CentOS và làm việc tốt với các phần mềm ảo hóa khác nhau: Xen, KVM, VMWare.

Hiện nay đã có khá nhiều ứng dụng hỗ trợ của hãng thứ 3 dành cho Eucalyptus, nổi bật nhất là các ứng dụng:

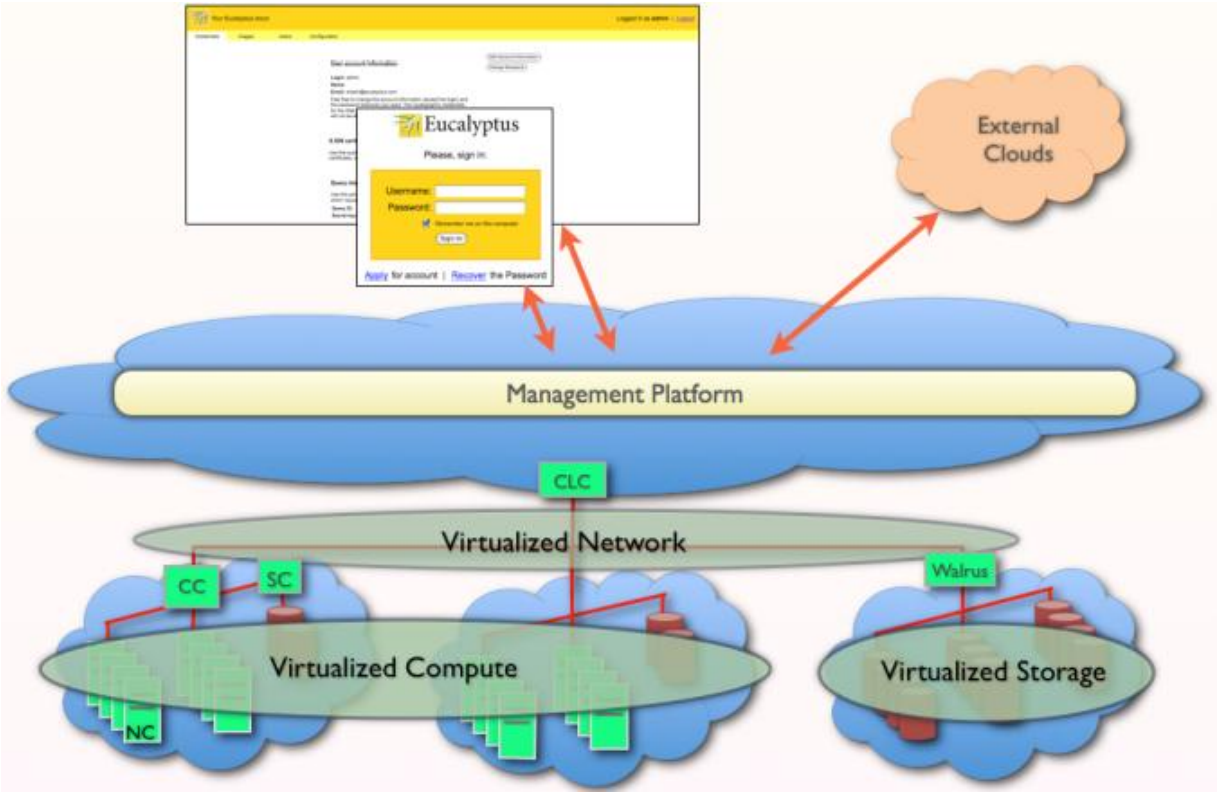
- ✓ ***Ubuntu Enterprise Cloud:*** hệ điều hành nhân Linux dạng tất cả trong một (all-in-one) có tích hợp sẵn Eucalyptus.
- ✓ ***RightScale:*** một ứng dụng web giúp kiểm soát một số thành phần của Eucalyptus.
- ✓ ***CloudBerry:*** công cụ giúp người quản trị dễ dàng quản lý các bucket trong thành phần Walrus của Eucalyptus.

3.2. Kiến trúc của Eucalyptus

3.2.1. Xét về mặt công nghệ

Eucalyptus được thiết kế gồm nhiều module giúp cho việc cài đặt được dễ dàng hơn. Người quản trị có thể cài đặt cả hệ thống Eucalyptus trên một máy server duy nhất hoặc cài đặt trên cụm gồm nhiều server khác nhau. Phần khung (framework) của Eucalyptus được tạo bởi sự kết hợp theo hướng module hóa của nhiều dịch vụ web (web services) cùng nhau làm việc thông qua các giao thức chuẩn (HTTPS, FTP,…).Với bộ khung này, Eucalyptus có thể thiết lập các máy ảo và nguồn tài nguyên lưu trữ ảo nối liền với nhau, nằm ở lớp mạng biệt lập so người dùng (lớp phía trên) và hệ thống máy chủ vật lý có sẵn (lớp phía dưới).

Các trình duyệt web, các giao diện ứng dụng SOAP và REST đều có thể tương tác được với Eucalyptus từ phía người dùng.



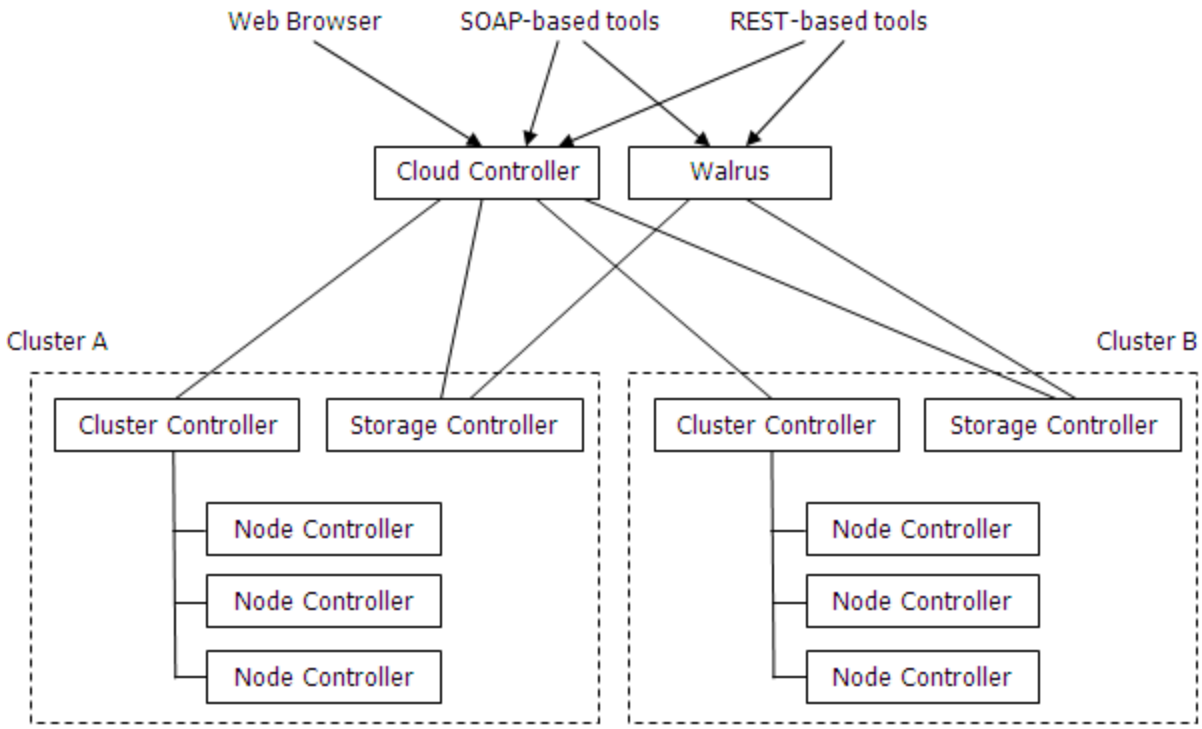
Hình 3.1 – Mô hình khái niệm của Eucalyptus.[10]

CLC - Cloud Controller.

CC - Cluster Controller.

NC - Node Controller.

SC - Storage Controller.



Hình 3.2 – Mô hình logic của Eucalyptus[11]

3.2.2. Năm thành phần trừu tượng trong Eucalyptus

Node Controller (NC) được cài trên mỗi máy server thực của hệ thống. NC kiểm soát việc cài đặt, cấu hình và tắt/mở các máy ảo. NC cũng thực hiện chức năng hủy bỏ và xóa đi các thành phần được tạo ra từ các tệp image (như tệp kernel, tệp root hệ thống, và tệp image ramdisk).

Cluster Controller (CC) thực hiện nhiệm vụ tổng hợp thông tin của các máy ảo, lập lịch cho các máy ảo chạy trên NC nào đó, và quản lý hệ thống mạng ảo trong Eucalyptus. Tất cả các node được quản lý bởi một CC phải nằm trong cùng một vùng quảng bá (broadcast domain).

Storage Controller (SC) cung cấp khả năng lưu trữ và truy xuất dữ liệu theo dạng khối (block-level) cho các máy thực, giúp cho việc truy xuất trực tiếp đến các máy này được thực hiện dễ dàng hơn. SC tương đương với Amazon EBS.

Walrus (put/get storage) làm nhiệm vụ lưu trữ các tệp image của các máy ảo, chứa dữ liệu lâu dài của người dùng, tổ chức phân hoạch theo các bucket và object. Walrus cho phép người dùng có thể tạo, xóa, liệt kê bucket, đặt(put), lấy(get), xóa bucket, và thiết lập các quy

định cho việc truy xuất đến các tệp image. Walrus tương đương với Amazon S3, và có hỗ trợ phần giao diện quản lý định dạng image Amazon Machine Image (AMI) của Amazon.

Cuối cùng, **Cloud Controller** (CLC) thực hiện vai trò là cầu nối giữa cả hệ thống Eucalyptus với thế giới bên ngoài. CLC có khả năng đưa ra các yêu cầu lập lịch ở mức cao và gửi yêu cầu đó đến cho CC thực hiện, đồng thời có thể chuyển các câu lệnh yêu cầu của người dùng đến cho CC. Về bản chất, CLC chịu trách nhiệm cho việc hiển thị bên ngoài, cũng như quản lý nguồn tài nguyên ảo bên dưới (các máy chủ, hệ thống mạng, hệ thống lưu trữ) thông qua thư viện chuẩn API của Amazon EC2 và phần giao diện web dùng để giao tiếp với người dùng.

Do hệ thống Eucalyptus co dãn tốt, nên các thành phần trừu tượng kể trên có thể được cài đặt trong một máy server duy nhất, hoặc cài đặt trong một phạm vi lớn gồm nhiều máy server khác nhau (datacenter).

3.3. Euca2ools – Công cụ hỗ trợ quản lý máy ảo (VM)

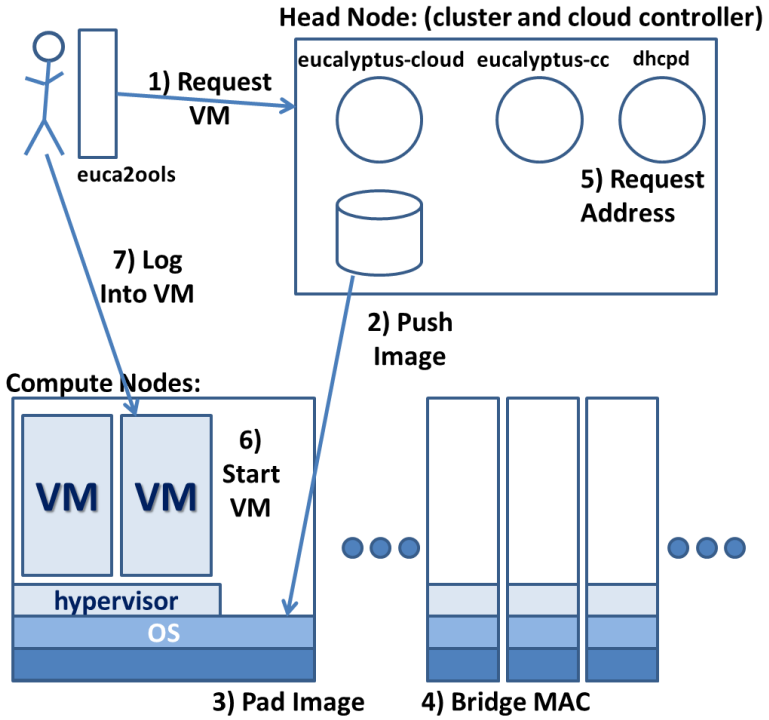
Euca2ools là bộ công cụ dòng lệnh có thể tương tác với Web Services sử dụng dịch vụ REST, hoặc các lệnh truy vấn dựa trên bộ thư viện API của Amazon EC2 và S3. Các lệnh trong Euca2ools được tạo ra dựa theo ý tưởng xây dựng bộ công cụ dòng lệnh của chính Amazon (api-tools và ami-tools). Tuy nhiên, Euca2ools đã được viết lại bằng Python, bộ thư viện boto và bộ công cụ M2Crypto.

Euca2ools hỗ trợ các chức năng sau đây:

- ✓ Manage Images (bundle, upload, register, delete, …)
- ✓ Manage Instances (start, reboot, terminate, list, …)
- ✓ Manage Volumes & Snapshots (attach, detach, list, create, delete, …)
- ✓ Manage IP addresses (associate, disassociate, list, ….)
- ✓ Manage SSH key pairs (add, delete, list)
- ✓ Manage Security groups (add, delete, list, add rules)
- ✓ Query availability zones (clusters)

Euca2ools có thể sử dụng cho Eucalyptus, đồng thời cũng hỗ trợ Amazon EC2 và S3.

3.4. Quy trình tạo ra máy ảo (VM) trong hệ thống Cloud sử dụng Eucalyptus



Hình 3.4 – Quy trình tạo ra máy ảo (VM) [12]

1) Người dùng sử dụng Euca2ools gửi yêu cầu tạo máy ảo (VM) đến Eucalyptus.
2) Tệp ảnh đĩa (image) lưu trong Walrus được đưa vào máy Node (máy cài Node Controller) để làm khuôn mẫu tạo máy ảo.
3) Từ tệp ảnh đĩa vừa đưa vào, hệ thống sẽ canh chỉnh dung lượng theo đúng yêu cầu của người dùng và copy thành các tệp mới. Các tệp này sẽ được lưu trong một thư mục có tên là *ID* của chính máy ảo.
4) Máy Node thiết lập môi trường mạng, cung cấp một card mạng ảo (virtual NIC) và một địa chỉ MAC ảo.
5) DHCPD cung cấp địa chỉ IP.
6) Hypervisor - trình quản lý và theo dõi máy ảo (Xen, KVM, VMWare,…) chính thức tạo ra máy ảo từ các file đã copy trong bước 3; đồng thời sử dụng card mạng ảo, địa chỉ MAC ảo tạo ở bước 4, địa chỉ IP tạo ở bước 5, gán cho máy ảo vừa

được tạo ra. Lúc này máy ảo sẽ có một tên định danh riêng gồm tiếp đầu ngữ *i-* và 8 chữ số hexa phía sau (ví dụ: *i-4892A3BD*).

7) Người dùng có thể SSH trực tiếp vào máy ảo bằng cách sử dụng khóa xác nhận của mình.

Ví dụ minh họa:

Khi người dùng thực hiện lệnh yêu cầu Eucalyptus tạo ra máy ảo bằng lệnh,

```
euca-run-instances -k mykey --kernel eki-B0850F7D --ramdisk eri-E345106A emi-3E921241
```

Màn hình sẽ hiển thị kết quả:

```
RESERVATION     r-51B008CA      admin   default
INSTANCE        i-4438092A      emi-3E921241    192.168.1.30
10.10.1.2       pending         mykey   0       m1.small    2
011-12-19T06:28:08.727Z         cc1     eki-B0850F7D    eri-
E345106A
```

Máy ảo khi đó ở trạng thái **pending,** có ID là *i-4438092A*, có public IP là 192.168.1.30, có private IP là 10.10.1.2.

Sau một khoảng thời gian, máy ảo *i-4438092A* sẽ hoạt động, trạng thái khi đó trở thành **running**.

```
euca-describe-instances
```

```
RESERVATION     r-51B008CA      admin   default
INSTANCE        i-4438092A      emi-3E921241    192.168.1.30
10.10.1.2       running         mykey   0       m1.small    2
011-12-19T06:28:08.727Z         cc1     eki-B0850F7D    eri-
E345106A
```

Lúc này người dùng có thể sử dụng khóa xác nhận của mình để login vào máy ảo vừa được tạo ra.

```
ssh -i ~/.ssh/id_mykey root@192.168.1.30
```

3.5. Những thuận lợi của Eucalyptus

Eucalyptus cung cấp khả năng ảo hóa các máy server, hệ thống mạng, và hệ thống lưu trữ ở mức độ bảo mật cao, nhằm giảm thiểu chi phí, tăng khả năng bảo trì, và cung cấp cho người

dùng chế độ tự phục vụ (người dùng có thể sự dụng khóa xác nhận (Credential key) của mình để chạy và đăng nhập vào các máy ảo).

Việt thiết kế theo hướng module hóa giúp Eucalyptus tạo ra sự thay đổi về mặt giao diện người dùng, đem những lợi ích của công nghệ ảo hóa đến một phạm vi người dùng rộng lớn, bao gồm: quản trị viên (admins), các nhà phát triển (developers), người quản lý (managers), khách hàng thuê host (hosting customers). Và cung cấp nền tảng cho các nhà cung cấp dịch vụ để họ có thể phác thảo ra mô hình giá cả phục vụ cho người dùng cuối.

Tính năng tạo ra các máy ảo (VM), các volumn và các snapshot giúp cải thiện độ tin cậy của hệ thống, thực hiện các thao tác theo một khuôn mẫu hoặc cho phép tự động hóa hoàn toàn. Điều này giúp cho hệ thống trở nên dễ sử dụng hơn, giảm đi thời gian làm quen tiếp cận hệ thống đối với người dùng bình thường, và giảm thời gian thực hiện lại (nếu có) đối với các dự án làm việc trên hệ thống.

Phần nhân của Eucalyptus sẽ luôn được duy trì là mã nguồn mở, giúp cho người dùng có thể sử dụng, phát triển mã nguồn cũng như có cơ hội sử dụng các bản phân phối từ cộng đồng các nhà phát triển.

Hệ thống các phần mềm hỗ trợ cho Amazon AWS sẽ được thúc đẩy phát triển nhanh. Như RightScale, CohesiveFT, Zmanda, rPath,… tất cả đều là các đối tác cung cấp các giải pháp cho Amazon AWS, đều có thể làm việc thử nghiệm trên Eucalyptus.

3.6. Những khó khăn của Eucalyptus

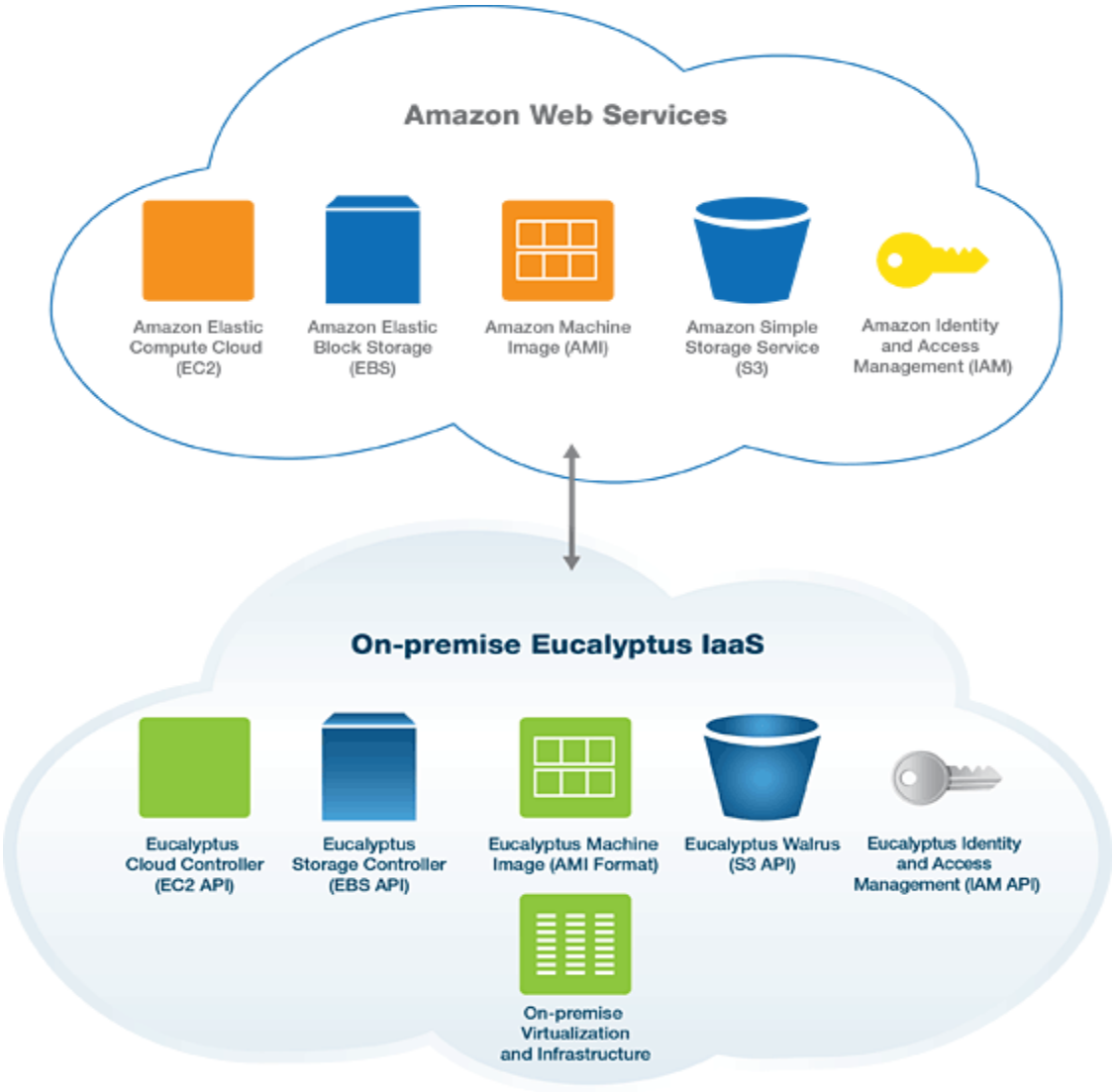
Eucalyptus hiện chỉ có thể triển khai ở dạng Private Cloud, chưa cung cấp các tiện ích tính toán cho người dùng bên ngoài hệ thống (người dùng internet).

Eucalyptus còn thiếu các tính năng tự động hóa việc lập lịch cho các máy ảo, không có khả năng ưu tiên cấp phát tài nguyên cho máy ảo cụ thể nào đó đang thực sự cần được cấp phát. Trong khi đó, OpenNebula với sự kết hợp của Haize đã làm tốt được việc này.

Eucalyptus không hỗ trợ tính năng *migration* – là khả năng vay mượn các máy ảo từ một cluster khác trong hệ thống cloud. Trong khi đó, OpenNebula có hỗ trợ tính năng này, và đây cũng chính là một lợi thế rất lớn của OpenNebula.

3.7. Sự tương hợp giữa Eucalyptus và Amazon Web Services

Eucalyptus có giao diện tương thích với cơ sở hạ tầng Cloud Computing của AWS, nghĩa là có thể sử dụng lại các công cụ hoặc script dùng cho AWS đã có trước đó để quản lý các hệ thống Private Cloud của người dùng. Thư viện API (Application Programming Interface) của Amazon được cài đặt ở tầng trên cùng của kiến trúc Eucalyptus, vì vậy nên bất kỳ công cụ nào giao tiếp được với Amazon AWS cũng đều có thể giao tiếp với Eucalyptus.



Hình 3.5 – Những thành phần tương hợp giữa Eucalyptus và AWS.[13]

Những thành phần của Eucalyptus ở hình 3.5 đều ở dạng mã nguồn mở và giao tiếp với nhau thông qua các dịch vụ web, cùng với một lớp giao tiếp được thêm vào để thể hiện sự tương hợp về mặt giao diện với Amazon.

Những thuận lợi có được từ sự tương hợp

- ✓ Eucalyptus có thể sử dụng những công cụ hỗ trợ dùng cho AWS.
- ✓ Eucalyptus hỗ trợ nhiều phần mềm ảo hóa: Xen, KVM, VMware. (Hiện tại Vmware chỉ hỗ trợ cho phiên bản Eucalyptus Enterprise Edition).
- ✓ Eucalyptus chạy được các tệp image của Amazon chỉ trong vài phút.
- ✓ Eucalyptus có thể chạy liên tục cùng một ứng dụng ở cả trong hoặc ngoài hệ thống quản lý tài nguyên ảo.

Chương 4 – SỬ DỤNG NAGIOS ĐỂ GIÁM SÁT CÁC MÁY ẢO

4.1. Giới thiệu Nagios

Nagios là bộ công cụ mã nguồn mở khá phổ biến, dùng để giám sát (monitor) hoạt động của hệ thống máy tính, và giám sát cơ sở hạ tầng của phần mềm ứng dụng. Nagios cung cấp nhiều tính năng giám sát khác nhau; đưa ra cảnh báo chính xác về hoạt động của các máy chủ, thiết bị chuyển mạch, các ứng dụng, và các dịch vụ. Nagios giám sát các máy chủ, dịch vụ, và cảnh báo cho người dùng biết khi hệ thống gặp sự cố, và cũng thông báo khi hệ thống đã hoạt động tốt hơn.[16]

Hình 4.1 – Giao diện chính của Nagios

Nagios ban đầu được tạo ra với tên gọi NetSaint, được viết bởi Ethan Galstad. Còn **N.A.G.I.O.S** là từ viết tắt của câu: "*Nagios Ain't Gonna Insist On Sainthood*". Từ "Sainthood" có nghĩa là "Vị thánh", được dùng để ám chỉ đến cái tên ban đầu NetSaint đã được thay đổi để đáp ứng với một rào cản pháp lý. Do trước đó một nhãn hiệu hàng hoá cũng có tên tương tự với NetSaint. "Agios" cũng là một từ phiên âm từ chữ *άγιος* trong tiếng Hy Lạp, có nghĩa là "vị thánh".[16]

Nagios ban đầu được thiết kế để chạy trên Linux. Nhưng sau đó nó được phát triển thêm để chạy tốt trên các bản phân phối khác của Unix. Nagios là phần mềm miễn phí, được cấp giấy phép theo các điều khoản của GNU General Public License phiên bản 2 (xuất bản bởi Tổ chức Phần mềm Tự do - *Free Software Foundation*).

4.2. Tổng quan về một số đặc điểm/chức năng của Nagios

- ✓ Giám sát các dịch vụ mạng (SMTP, POP3, HTTP, NNTP, ICMP, SNMP, FTP, SSH). Nagios có thể kiểm tra cùng lúc (song song) các dịch vụ này.

- ✓ Giám sát các tài nguyên của các máy trạm (processor load, disk usage, system logs), và gửi dữ liệu thu thập được qua mạng thông qua các plugins.
- ✓ Giám sát máy trạm thông qua script chạy từ xa nhờ plugin *Nagios Remote Plugin Executor* – NRPE.
- ✓ Có thể giám sát từ xa thông qua SSH và có hỗ trợ mã hóa SSL.
- ✓ Các plugin có sẵn được thiết kế đơn giản, cho phép người dùng dễ dàng phát triển các lệnh kiểm tra dịch vụ tùy theo nhu cầu của riêng họ, bằng cách sử dụng các công cụ, ngôn ngữ phổ biến (Shell Script, C++, Perl, Ruby, Python, PHP, C#, ...).
- ✓ Có khả năng xác định hệ thống phân cấp máy chủ trong mạng bằng cách sử dụng khái niệm "*parent host*", cho phép phát hiện và phân biệt giữa các máy chủ không còn hoạt động và những máy chủ không thể truy cập.
- ✓ Luân chuyển tập tin đăng nhập tự động.
- ✓ Hỗ trợ thực hiện giám sát các máy chủ dư thừa.
- ✓ Các tùy chọn trong giao diện web giúp người dùng xem tình trạng mạng hiện tại, các thông báo, lịch sử truy cập, các tập tin đăng nhập.
- ✓ Việc lưu trữ dữ liệu được thực hiện trong các tập tin văn bản mà không phải là cơ sở dữ liệu.

4.3. Một số plugin phổ biến dùng để hỗ trợ cho Nagios

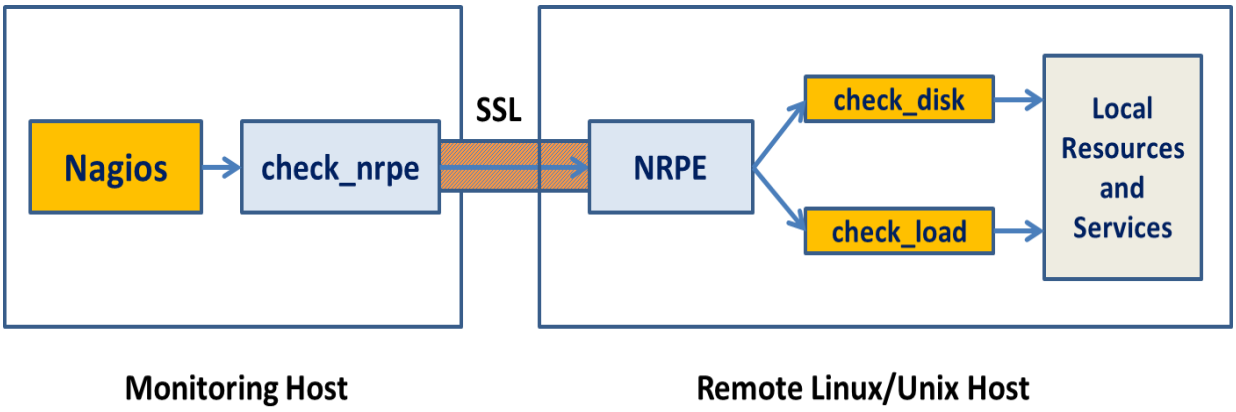
4.3.1. Nagios Remote Plugin Executor - NRPE

Nagios Remote Plugin Executor (NRPE) là một công cụ dùng cho Nagios, cho phép hệ thống theo dõi (Monitoring Host) giám sát từ xa một máy chủ khác (Remote Host).

NRPE cho phép giám sát các tài nguyên như:

- ✓ *Check_disk* (dung lượng đĩa),
- ✓ Check_load (kiểm tra tải của CPU),
- ✓ Check_users (đếm số user đang đăng nhập),
- ✓ *Check_procs* (đếm số process đang chạy),…

Nagios thực hiện thăm dò định kỳ các tài nguyên trên hệ thống từ xa (Remote Host) bằng cách sử dụng plugin check_nrpe (có trong gói cài đặt NRPE).



Hình 4.2 – Nagios kiểm tra tài nguyên của Remote Host thông qua NRPE.[14]

Check_nrpe là một plugin có sẵn trong gói cài đặt của NRPE, được cài đặt vào máy Monitoring Host.

Khi Nagios yêu cầu check _nrpe thực thi lệnh giám sát, check_nrpe sẽ gửi yêu cầu đó cho NRPE. NRPE thực hiện các lệnh được yêu cầu (lệnh đó phải cài sẵn trong plugin của NRPE) trên cùng một máy và truyền thông tin thu thập được cho check_nrpe. Sau cùng thông tin này được gửi về cho Nagios để hiển thị.

Chú ý: Tùy thuộc vào CPU / hệ điều hành của máy Remote Host, ta sẽ phải biên dịch lại NRPE và bổ sung thêm một số tùy chọn (Ví dụ như cho phép giao thức SSL có thể hoạt động).

4.3.2. NSClient++

Chương trình này được sử dụng chủ yếu để giám sát các máy tính sử dụng hệ điều hành Windows. NSClient++ được cài đặt trên hệ thống giám sát từ xa (Remote system) – giao tiếp bằng cổng TCP 1248. Nagios plugin được sử dụng để thu thập thông tin từ addon này gọi là *check_nt*. Giống như NRPE, NSClient++ cho phép giám sát các dịch vụ dùng riêng trong Remote System (dung lượng bộ nhớ, tải CPU, dung lượng đĩa cứng, tiến trình đang chạy,…).[16]

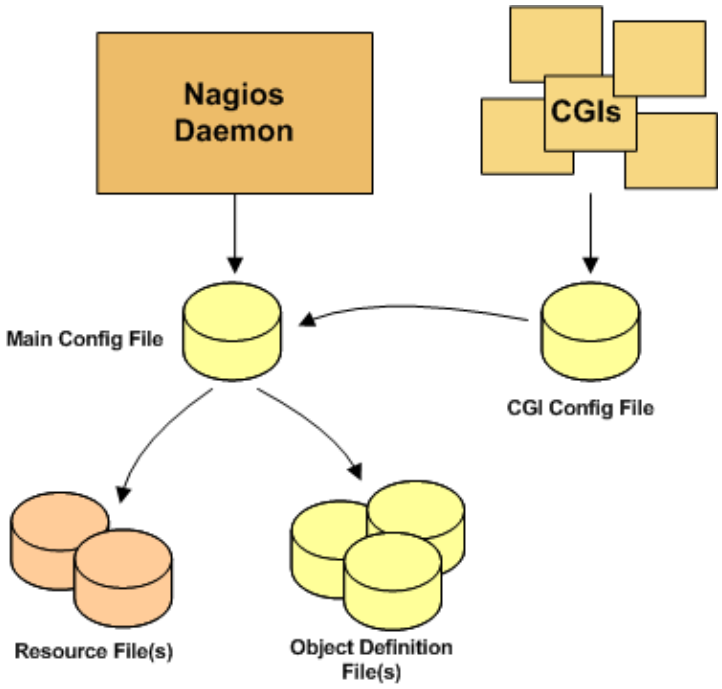
Do bài báo cáo chỉ đề cập đến các hệ thống sử dụng hệ điều hành Linux/Unix nên chúng ta không đi sâu vào NSClient++.

4.4.Tổng quan về các tệp cấu hình trong Nagios

4.4.1. Giới thiệu

Có một số tập tin cấu hình khác nhau mà người dùng cần phải tạo mới hoặc chỉnh sửa trước khi bắt đầu giám sát bất cứ đối tượng gì (máy chủ, dịch vụ, máy in…). Việc cấu hình cho Nagios có thể mất nhiều thời gian, đặc biệt là nếu người dùng chỉ mới sử dụng lần đầu. Nhưng một khi khi đã hiểu được cách làm việc của Nagios, người dùng sẽ tận dụng được nhiều tính năng rất hay của phần mềm hữu ích này.

Chú ý: các tệp cấu hình mẫu được lưu sẵn trong thư mục /usr/local/nagios/etc.



Hình 4.3 – Hệ thống các tệp cấu hình trong Nagios.[17]

4.4.2. Main Configuration File

Main Configuration File – chứa các chỉ thị có ảnh hưởng đến hoạt động của Nagios Daemon. Tập tin cấu hình này được đọc bởi Nagios Daemon và Nagios CGIs.

4.4.3. Resource File(s)

Resource File(s) – được sử dụng để lưu trữ các macro do người dùng định nghĩa. Chức năng chính của các tập tin tài nguyên là để lưu trữ các thông tin cấu hình nhạy cảm (như mật khẩu), mà không để cho phần cấu hình trong CGIs can thiệp vào.

4.4.4. Object Definition Files

Object Definition Files – được sử dụng để định nghĩa các host (máy chủ), services, hostgroups, contacts, contactgroups, commands,… Đây chính là nơi người dùng định nghĩa tất cả những đối tượng họ muốn giám sát và cách thức để giám sát những đối tượng đó.

Người dùng có thể chỉ định một hoặc nhiều tập tin định nghĩa đối tượng bằng cách sử dụng cfg_file và/hoặc chỉ thị cfg_dir trong tập tin cấu hình *Main Conf File* của người dùng.

4.4.5. CGI Configuration File

Tệp cấu hình CGI (Common Gateway Interface) – có chứa một số chỉ thị có ảnh hưởng đến hoạt động của CGIs (phần giao diện của Nagios). Tệp này tham chiếu đến tệp cấu hình chính, vì vậy CGIs biết người dùng đã cấu hình Nagios thế nào và phần định nghĩa các đối tượng được lưu trữ ở đâu.

Chương 5 – SỬ DỤNG APACHE BENCH ĐỂ KIỂM THỬ SỰ HOẠT ĐỘNG CỦA CÁC MÁY ẢO

5.1. Giới thiệu Apache Bench

ApacheBench (AB) là chương trình máy tính cung cấp các dòng lệnh thực thi theo kiểu đơn luồng, dùng để đo hiệu suất của web server. Ban đầu AB được thiết kế chỉ để kiểm thử các máy chủ sử dụng Apache HTTP web server, nhưng sau đó đã được phát triển để kiểm thử rất nhiều web server khác nhau.

AB được phát triển đi kèm với các tiêu chuẩn phân phối mã nguồn của Apache. Và tương tự như web server của Apache, AB là phần mềm miễn phí, mã nguồn mở và phân phối theo các điều khoản của Giấy phép Apache (Apache License).

5.2. Cách hoạt động của Apache Bench

Xét một câu lệnh ví dụ của AB

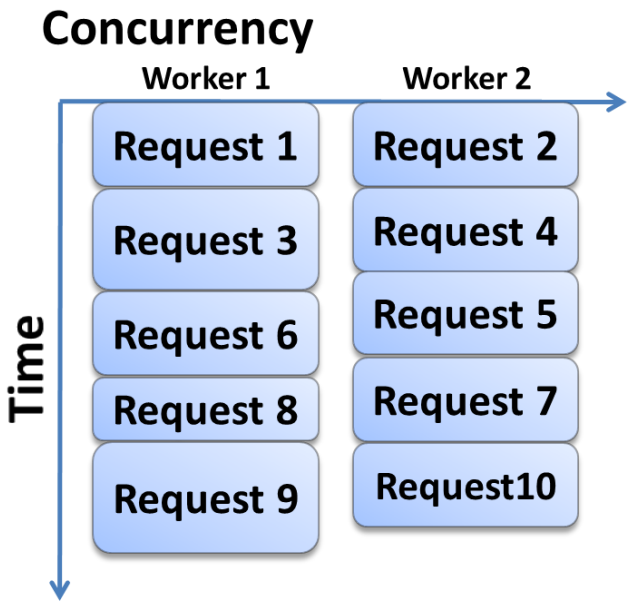
```
ab -c 2 -n 10 http://www.apache.org/
```

Có 3 tùy chọn ở câu lệnh trên:

- ✓ Concurrency (-c 2) – Số worker hoạt động đồng thời.

- ✓ Number of request (-n 10) – Số request sẽ gửi.
- ✓ URL (http://apache.org/) – Địa chỉ URL đích sẽ gửi request đến.

Cách hoạt động là AB tạo ra các worker làm việc đồng thời. Mỗi worker gửi lần lượt các request, và không xảy ra trường hợp 2 request được gửi cùng lúc. Quá trình này thực hiện liên tục cho đến khi toàn bộ request được gửi hết.



Hình 5.1 – Sơ đồ hoạt động của Apache Bench[18]

Mỗi request có khoảng thời gian thực hiện hơi chênh lệch nhau. Mỗi worker phải chờ cho đến khi request nó đang thực hiện đã hoàn thành, rồi mới có thể bắt đầu thực hiện request tiếp theo, và worker đó sẽ bắt đầu gọi request khác gần như ngay lập tức. Dễ thấy trên *hình 5.1*, khi có 2 worker cùng gửi request thì thời gian thực hiện gọi 10 request sẽ chỉ bằng một nửa so với việc chỉ có 1 worker thực hiện gọi tuần tự từng request.

Việc gửi request đồng thời như trên (Concurrency) là cách tốt nhất để kiểm tra và mô phỏng khả năng chịu tải (load) trong các ứng dụng web. Khi ứng dụng web có nhiều người dùng gửi request đến, lượng request sẽ tăng lên nhưng nó sẽ không đi quá tỷ lệ 1 – 1. Do hầu hết các request có thể được xử lý trong một khoảng thời gian ngắn cho một người dùng nhất định. Không giống như Apache Bench, người sử dụng thật sự không gửi liền request khác sau khi kết thúc một request trước đó càng nhanh càng tốt. Chính vì vậy mà việc mô phỏng

cho trường hợp này là tương đối khó khăn, và Apache Bench đã tạo ra trường hợp xấu nhất có thể để kiểm tra khả năng chịu tải của web server.

5.3. Kết quả hiển thị của Apache Bench

Lệnh gửi request,

```
ab -c 2 -n 10 http://www.apache.org/
```

Khi gửi xong request. AB sẽ hiển thị ra kết quả như sau.

```
1.  This is ApacheBench, Version 2.3 <$Revision: 655654 $>
2.  Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
3.  Licensed to The Apache Software Foundation, http://www.apache.org/
4.
5.  Benchmarking www.apache.org (be patient).....done
6.
7.
8.  Server Software:        Apache/2.3.15-dev
9.  Server Hostname:        www.apache.org
10. Server Port:            80
11.
12. Document Path:          /
13. Document Length:        36449 bytes
14.
15. Concurrency Level:      2
16. Time taken for tests:   0.637 seconds
17. Complete requests:      10
18. Failed requests:        0
19. Write errors:           0
20. Total transferred:      368500 bytes
21. HTML transferred:       364490 bytes
22. Requests per second:    15.71 [#/sec] (mean)
```

```
Time per request:       127.332 [ms] (mean)
Time per request:       63.666 [ms] (mean, across all concurrent requests)
Transfer rate:          565.24 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:       24   25   0.6     25      26
Processing:   101  102   0.8    102     104
Waiting:       25   26   0.3     26      26
Total:        125  127   1.4    127     129

Percentage of the requests served within a certain time (ms)
50%    127
66%    127
75%    127
80%    129
90%    129
95%    129
98%    129
99%    129
100%   129 (longest request)
```

Giải thích ý nghĩa cho từng dòng,[18]

Dòng	Mô tả của AB	Giải thích chi tiết
1-7	-	Phần giới thiệu thông tin về Apache Bench và thông báo quá trình gửi request đang bắt đầu.
8-12	-	Thông tin về server/URL được kiểm thử.
13	Document Length	Chiều dài của nội dung (tính bằng byte, không bao gồm HTTP headers).
15	Concurrency Level	Tham số mà người dùng chỉ định với –c, là tổng số lượng worker.
16	Time taken for tests	Tổng thời gian của toàn bộ quá trình kiểm tra.
17	Complete requests	Tham số mà người dùng chỉ định với –n, là tổng số lượng request.
18	Failed requests	Bất kỳ request nào không hoàn thành hoặc không hợp lệ (mã lỗi HTTP như 404 được báo là "Non-2xx responses" nếu nó xảy ra trong quá trình thử nghiệm).
19	Write errors	TCP socket thông báo lỗi.
20	Total transferred	Tổng số byte tải về từ máy chủ (bao gồm phần header).
21	HTML transferred	Số byte response đã tải về (không bao gồm phần header) = (dòng 13) * (dòng 17).
22	Requests per second	Số request trong một giây = (dòng 17)/(dòng 16)
23	Time per request [#1]	Tổng thời gian của tất cả request chia cho số lượng request (trường hợp chỉ có 1 worker).
24	Time per request [#2]	Thời gian trung bình của mỗi request = (dòng 16)/(dòng 17). Trường hợp này có nhiều worker, số worker do người dùng chỉ định.

25	Transfer rate	(dòng 20)/(dòng 16)
27-32	Connection Times	Phần phân tích của mỗi request.
29	Connect	Thời gian cần để thiết lập kết nối TCP (không bao gồm giao thức HTTP).
30	Processing	Tổng số thời gian trừ đi thời gian của phần Connect. Bao gồm tất cả các IO của HTTP, cộng với thời gian chờ đợi. Thời gian của IO có thể tính bằng cách lấy thời gian *Processing – Waiting*.
31	Waiting	Đây là thời gian từ lúc ghi byte cuối cùng của request đến lúc nhận byte đầu tiên của response. Lưu ý rằng thời gian của phần Processing cũng nằm trong khoảng thời gian này.
32	Total	Tổng thời gian của tất cả request, bao gồm cả thời gian mở kết nối TCP.
34-43	Percentage of the requests…	Đây là phân thống kê phân phối tích lũy. Thống kê này được bắt đầu ở mức 50% bởi vì khi server đang tải / thực hiện thử nghiệm, người dùng thường chỉ quan tâm đến các request chậm để biết được có bao nhiêu phần trăm request đang gặp vấn đề (nhưng thường thì không phải tất cả request đều gặp vấn đề).

5.4. Tải của hệ thống – CPU load

Trong UNIX, *tải* (*load*) chính là thước đo khối lượng công việc mà một hệ thống máy tính thực hiện. *Tải trung bình* (*load average*) là đại lượng đại diện cho tải của một hệ thống hoạt động trong một khoảng thời gian nào đó.

Thông thường phần hiển thị của *tải trung bình* gồm có ba chữ số, đó là tải của hệ thống trong một phút, năm phút, và mười lăm phút gần nhất.

Người dùng có thể dễ dàng truy vấn kết quả hiện tại bằng cách chạy lệnh uptime:

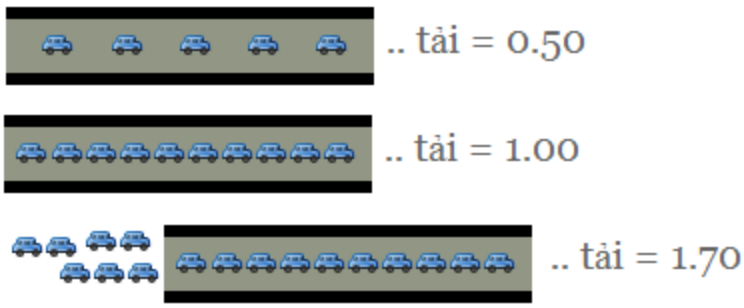
```
$ uptime
 14:34:03 up 10:43,  4 users,  load average:  0.06, 0.11, 0.09
```

Lệnh w và lệnh top cũng hiển thị cùng một trị số load average như vậy.

5.4.1.Trường hợp CPU chỉ có 1 nhân (single-core)

Tải có thể nhận các giá trị sau đây:

- ✓ [Giá trị của tải] = 0: Khi CPU ở trạng thái idle.
- ✓ 0 < [Giá trị của tải] < 1.00 : Khi CPU xử lý số lượng tiến trình **chưa đạt tới** ngưỡng tối đa cho phép.
- ✓ [Giá trị của tải] = 1.00 : Khi CPU xử lý số lượng tiến trình **bằng đúng** ngưỡng tối đa cho phép.
- ✓ [Giá trị của tải] > 1.00 : Khi CPU xử lý số lượng tiến trình **vượt quá** ngưỡng tối đa cho phép. Khi đó các tiến trình vượt quá sẽ phải chờ đến lượt được xử lý.



Hình 5.2 – Các giá trị của tải – trường hợp CPU có 1 nhân.[21]

5.4.2. Trường hợp CPU có 2 nhân (dual-core)

Tương tự trường hợp CPU có 1 nhân, ngưỡng cho phép ở trường hợp 2 nhân là 2.00.

- ✓ [Giá trị của tải] = 0 : Khi CPU ở trạng thái idle.
- ✓ 0 < [Giá trị của tải] < 2.00 : Khi CPU xử lý số lượng tiến trình **chưa đạt tới** ngưỡng tối đa cho phép.
- ✓ [Giá trị của tải] = 2.00 : Khi CPU xử lý số lượng tiến trình **bằng đúng** ngưỡng tối đa cho phép.
- ✓ [Giá trị của tải] > 2.00 : Khi CPU xử lý số lượng tiến trình **vượt quá** ngưỡng tối đa cho phép. Khi đó các tiến trình vượt quá sẽ phải chờ đến lượt được xử lý.

Hình 5.3 – Giá trị của tải – trường hợp CPU có 2 nhân.[21]

Một ví dụ sử dụng plugin check_nrpe, dùng lệnh giám sát check_load để kiểm tra tải của một máy có địa chỉ IP là 192.168.1.45.

```
#/usr/local/nagios/libexec/check_nrpe –H 192.168.1.45 –c check_load

OK – load average: 13.94, 4.45, 1.48| load1=13.940;15.000;30.000;0;
   load5=4.450;10.000;25.000;0;    load15=1.840;5.000;20.000;0;
```

Chú ý: để kiểm tra CPU của máy chủ UNIX/LINUX đang hoạt động có bao nhiêu nhân, ta có thể xem trong tệp /proc/cpuinfo.[21]

```
grep 'model name' /proc/cpuinfo | wc –l
```

Kết quả sẽ là số nhân của CPU, ví dụ:

```
2
```

Chương 6 – TÓM TẮT QUY TRÌNH TRIỂN KHAI HỆ THỐNG PRIVATE CLOUD VÀ HỆ THỐNG GIÁM SÁT CÁC MÁY ẢO

6.1. Tóm tắt tổng quát toàn bộ quy trình

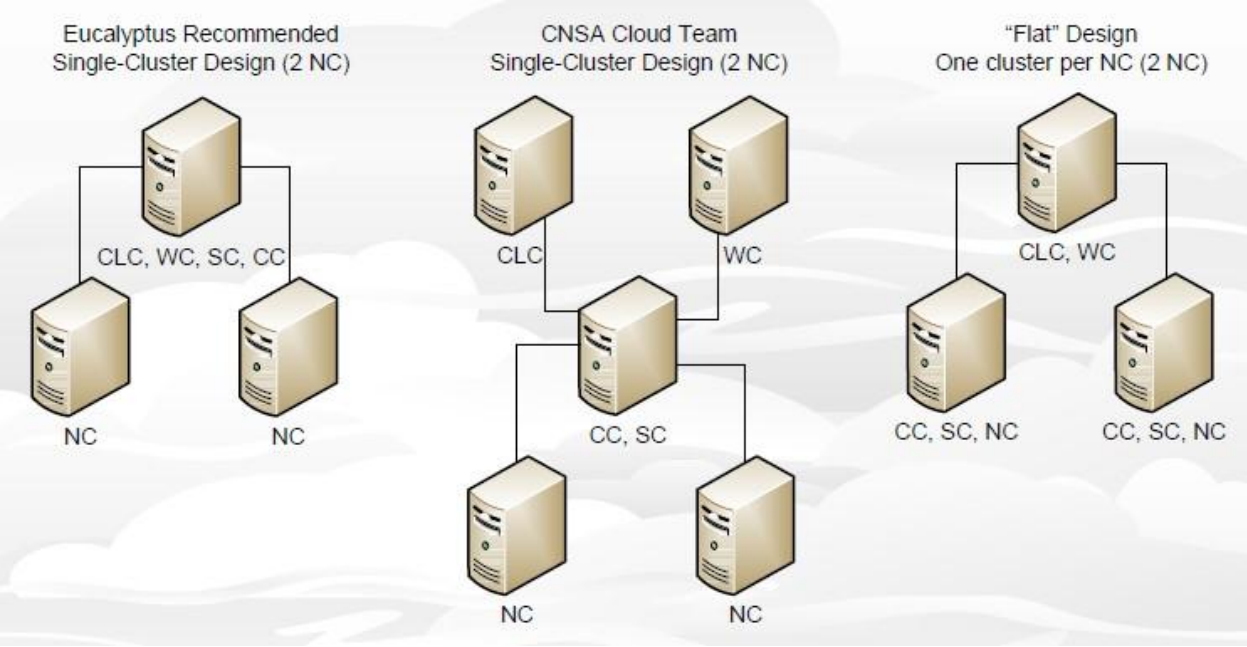
1. Triển khai Eucalyptus trên hệ điều hành CentOS 5.6. Tạo ra các máy ảo.
2. Ssh vào các máy ảo.
3. Cài web server trên các máy ảo.
4. Triển khai hệ thống Nagios và NRPE để giám sát tự động các máy ảo.
5. Triển khai Apache Bench (AB) để gửi tự động HTTP request đến các máy ảo.
6. Nhân bản các máy ảo đã được cấu hình.

Sau khi triển khai xong bước 4 và bước 5, hệ thống có thể theo dõi tự động các máy ảo, người quản trị có thể đánh lệnh gửi HTTP request trong AB để gửi tự động các request đến các máy ảo. Các bước còn lại (bước 1, bước 2, bước 3, bước 6) đều phải thực hiện thủ công.

Sau đây là phần tóm tắt chi tiết thực hiện của từng bước quá trình triển khai. Nhưng trước tiên chúng ta cần đề cập đến các mô hình triển khai Private Cloud dựa trên Eucalyptus thường được áp dụng.

6.2. Giới thiệu các mô hình triển khai Private Cloud dựa trên Eucalyptus

Khi triển khai Eucalyptus, có rất ít sự ràng buộc về vị trí cài đặt cho các thành phần của cả hệ thống. Eucalyptus được thiết kế đặc biệt theo hướng module hóa theo từng thành phần nên việc cài đặt trở nên hết sức linh động. Các thành phần của Eucalyptus có thể được cài đặt trong một máy server duy nhất, hoặc cài đặt trong một phạm vi lớn gồm nhiều máy server khác nhau.



Hình 6.1 – Các mô hình triển khai Eucalyptus.[11]

Có 3 mô hình triển khai phổ biến dành cho Eucalyptus.[11]

Mô hình 1 – Eucalyptus Recommended – Single-cluster Design (2 NC)

Do nhóm phát triển Eucalyptus tại đại học UCSB đề nghị: Tất cả các thành phần (ngoại trừ NC) được cài đặt trên một máy duy nhất là Front-end. Chi phí triển khai cho mô hình này là rất thấp. Tuy nhiên, việc mở rộng trong tương lai có thể sẽ khó khăn, và các dịch vụ đang chạy sẽ làm chậm hiệu suất của máy Front-end (máy cài CLC, CC, SC, Walrus) đáng kể.

Mô hình 2 – CNSA Cloud Team – Single-cluster Design (2 NC)

Do nhóm CNSA Cloud Team đề xuất. Mô hình này giúp dễ dàng để mở rộng hệ thống trong tương lai. Nhưng các chi phí đầu tư cho máy tính lại cao hơn (vì phải đầu tư mua nhiều máy hơn).

Mô hình 3 - "flat" design – One cluster per NC (2 NC)

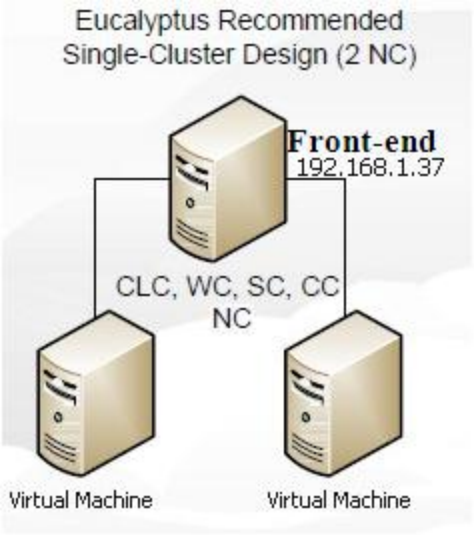
Trong mô hình này, thành phần NC được triển khai chung với CC và SC trên cùng một máy, CLC và Walrus được cài trên máy Front-end. Ưu điểm lớn nhất của mô hình này đó là tốc độ truy xuất đĩa nhanh hơn (do không gian lưu trữ là DAC chứ không phải NAS [11]). Tuy

nhiên, việc có quá nhiều thành phần CC và SC trong mô hình có thể làm chậm khả năng xử lý của máy Front-end.

6.3. Triển khai Eucalyptus để tạo ra máy ảo

Để việc cài đặt, triển khai trở nên đơn giản nhưng vẫn đáp ứng yêu cầu thực tế, chúng ta sẽ cài đặt toàn bộ các thành phần của Eucalyptus trên 1 máy thực duy nhất.

- ✓ Ta sẽ cần 1 máy thực, gọi tên là **Front-end**, dùng làm nơi cài CLC, Walrus. Để việc tạo máy ảo được thực hiện dễ dàng, chúng ta sẽ làm việc trong phạm vi một cluster và thiết lập Cluster Controller (CC), Storage Controller (SC), Node Controller (NC) trên máy Front-end. Máy Front-end có một card mạng với địa chỉ IP **192.168.1.37**. Một máy thực thường **chỉ tạo tối đa 2 máy ảo**.



Hình 6.2 – Mô hình triển khai thử nghiệm Eucalyptus.

Khi máy ảo chạy lên, nó sẽ tự động được DHCP gán địa chỉ IP nằm trong khoảng 192.168.1.30 đến 192.168.1.60 (dải địa chỉ IP này do người dùng tự cấu hình trong DHCP ở tệp /etc/dhcpd.conf và tệp /etc/eucalyptus/eucalyptus.conf).

Ví dụ minh họa:

Khi người dùng thực hiện lệnh yêu cầu Eucalyptus tạo ra máy ảo bằng lệnh,

```
euca-run-instances -k mykey --kernel eki-B0850F7D --ramdisk eri-E345106A emi-3E921241
```

Màn hình sẽ hiển thị kết quả:

```
RESERVATION     r-51B008CA      admin   default
INSTANCE        i-4438092A      emi-3E921241    192.168.1.30
10.10.1.2       pending         mykey   0       m1.small    2
011-12-19T06:28:08.727Z         cc1     eki-B0850F7D    eri-
E345106A
```

Máy ảo khi đó ở trạng thái **pending,** có ID là *i-4438092A*, có public IP là 192.168.1.30, có private IP là 10.10.1.2.

Sau một khoảng thời gian, máy ảo *i-4438092A* sẽ hoạt động, trạng thái khi đó trở thành **running**.

```
euca-describe-instances
```

```
RESERVATION     r-51B008CA      admin   default
INSTANCE        i-4438092A      emi-3E921241    192.168.1.30
10.10.1.2       running         mykey   0       m1.small    2
011-12-19T06:28:08.727Z         cc1     eki-B0850F7D    eri-
E345106A
```

Lúc này người dùng có thể sử dụng khóa xác nhận của mình để login vào máy ảo vừa được tạo ra.

6.4. Login vào máy ảo

Để login vào máy ảo, ta cần sử dụng đến khóa xác nhận (credential key) của người dùng. Nếu không có khóa xác nhận này. Khi login, máy ảo sẽ yêu cầu người dùng nhập mật khẩu đăng nhập. Nhưng lúc này ta lại không có mật khẩu nên không đăng nhập được.

Khóa xác nhận chính là khóa lưu ở máy của người quản trị, tương ứng với khóa được dùng trong quá trình tạo máy ảo.

Ví dụ: Khi tạo khóa xác nhận, đánh lệnh

```
euca-add-keypair  mykey > ~/.ssh/id_mykey
```

Thì khóa mykey sẽ lưu trên hệ thống của Eucalyptus, dùng để chạy máy ảo như ví dụ ở phần 6.3. Còn khóa ~/.ssh/id_mykey dùng để login vào máy ảo.

Thực hiện login bằng lệnh ssh như sau:

```
ssh -i ~/.ssh/id_mykey root@192.168.1.30
```

Lúc này ta đã có thể login vào được máy ảo mà không cần phải nhập mật khẩu. Tiếp theo ta sẽ cài web server vào máy ảo.

6.5. Cài web server trên máy ảo

Để cài được web server trên máy ảo, trước hết ta cần đánh lệnh sau.

```
yum clean all
```

Sau đó cài đặt Apache web server bằng lệnh

```
yum install -y httpd
```

6.6.Triển khai hệ thống Nagios, NRPE để giám sát hoạt động của các máy ảo

- Nagios là ứng dụng dùng để giám sát một hệ thống mạng máy tính.

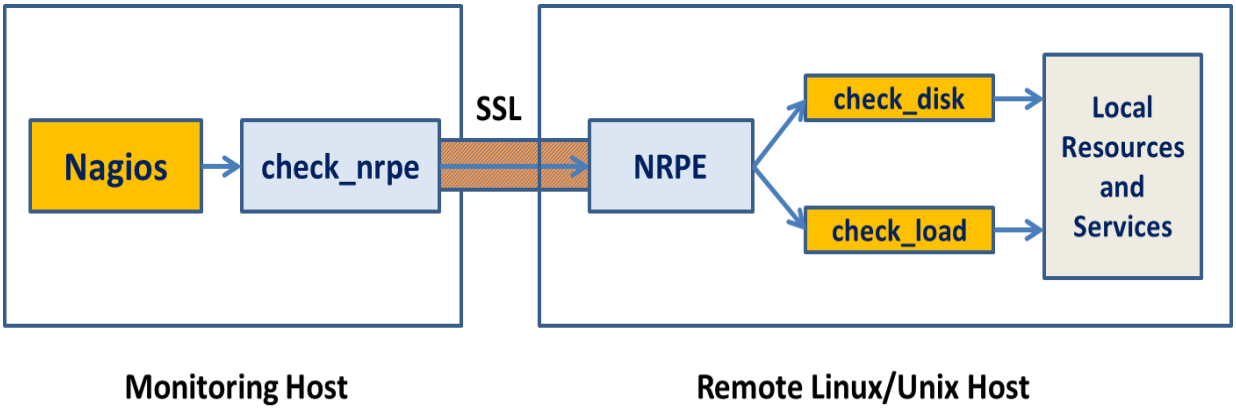
- Tính năng nổi bật của Nagios là giám sát các dịch vụ mạng như: HTTP, SMTP, SSH, Telnet.

Để giám sát các máy ảo, ta cần phải triển khai thêm NRPE (*Nagios Remote Plugin Executor*).

NRPE cho phép giám sát các tài nguyên như:

- ✓ *Check_disk* (dung lượng đĩa),
- ✓ Check_load (kiểm tra tải của CPU),
- ✓ Check_users (đếm số user đang đăng nhập),
- ✓ *Check_procs* (đếm số process đang chạy),…

Nagios thực hiện thăm dò định kỳ các tài nguyên trên hệ thống từ xa (Remote Host) bằng cách sử dụng plugin check_nrpe (có trong gói cài đặt NRPE).



Hình 6.3 – Nagios kiểm tra tài nguyên của Remote Host thông qua NRPE.[13]

Check_nrpe là một plugin có sẵn trong gói cài đặt của NRPE, được cài đặt vào máy Monitoring Host.

Khi Nagios yêu cầu check _nrpe thực thi lệnh giám sát, check_nrpe sẽ gửi yêu cầu đó cho NRPE. NRPE thực hiện các lệnh được yêu cầu (lệnh đó phải cài sẵn trong plugin của NRPE) trên cùng một máy và truyền thông tin thu thập được cho check_nrpe. Sau cùng thông tin này được gửi về cho Nagios để hiển thị.

Tiếp theo ta cần thực hiện gửi các HTTP request đến máy ảo một cách tự động để kiểm thử khả năng chịu tải của các máy ảo.

6.7. Triển khai Apache Bench để gửi tự động HTTP request đến các máy ảo

Cài đặt Apache Bench (AB) bằng câu lệnh sau

```
yum install –y apr apr-util
```

Sau đó có thể thực hiện gửi HTTP request như sau

```
ab -c 2 -n 10 http://www.apache.org/
```

Có 3 tùy chọn ở câu lệnh trên:

- ✓ Concurrency (-c 2) – Số worker hoạt động đồng thời.
- ✓ Number of request (-n 10) – Số request sẽ gửi.

URL (http://apache.org/) – Địa chỉ URL đích sẽ gửi request đến.

Ta có thể gửi liên tiếp thật nhiều request để kiểm thử máy ảo.

```
ab -c 100 -n 100000 http://192.168.1.33:80/
```

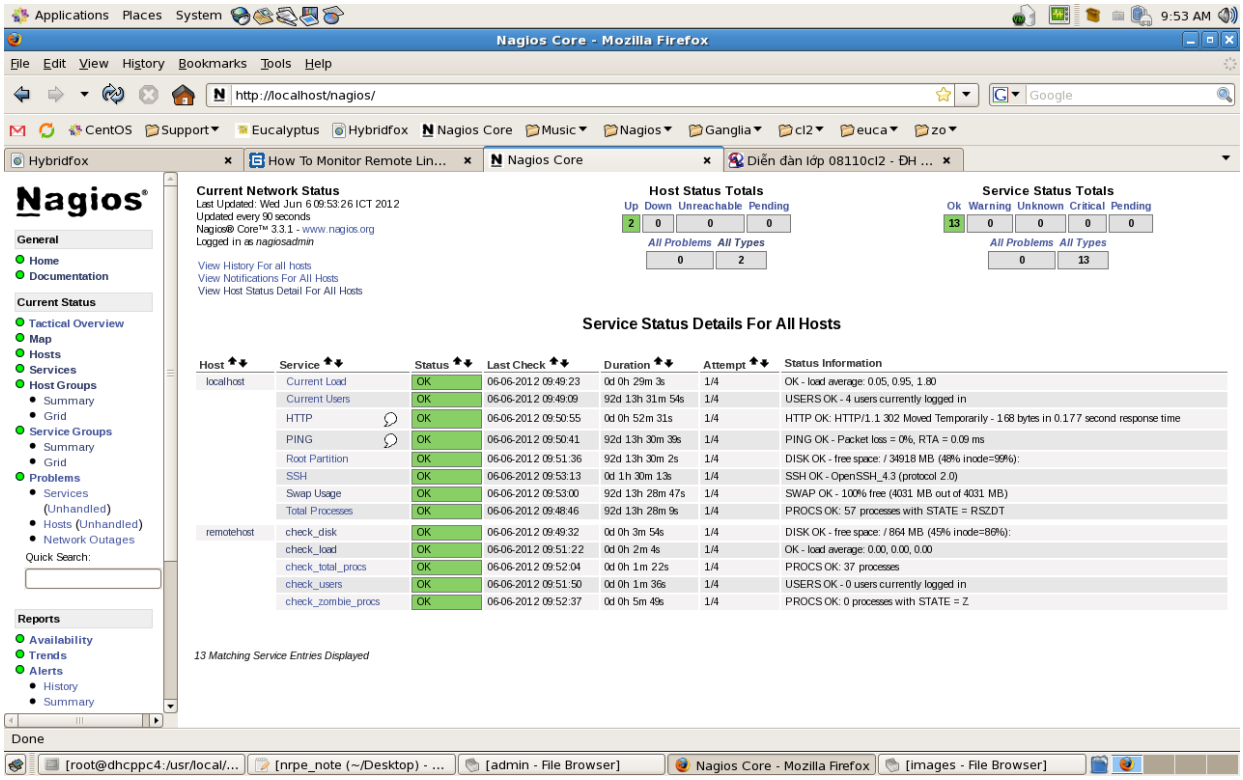
Sau đó dùng lệnh check_load của check_nrpe để kiểm tra tải của máy ảo.

```
#/usr/local/nagios/libexec/check_nrpe –H 192.168.1.33 –c check_load
```

Kết quả

```
OK – load average: 13.94, 4.45, 1.48| load1=13.940;15.000;30.000;0;
    load5=4.450;10.000;25.000;0;  load15=1.840;5.000;20.000;0;
```

Hoặc có thể theo dõi bằng hình ảnh trực quan trên Nagios. Cứ sau một khoảng thời gian nhất định (do người quản trị thiết lập), hệ thống Nagios sẽ tự cập nhật thông tin giám sát.



Hình 6.4 – Giám sát hoạt động của máy ảo từ Nagios thông qua các lệnh của NRPE.

Đến đây, ta đã hiện thực hóa thành công toàn bộ hệ thống bao gồm các chức năng:

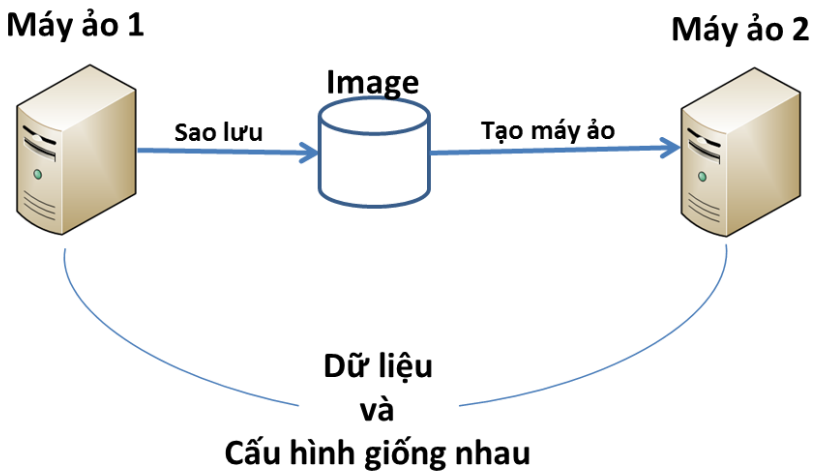
- ✓ Tạo ra các máy ảo.
- ✓ Giám sát các máy ảo.
- ✓ Gửi tự động HTTP request đến máy ảo.

Tiếp theo cần phải nhân bản máy ảo đang chạy, để lưu lại các cấu hình hiện tại của máy ảo ra thành một tệp image mới. Sau này khi chạy máy ảo mới từ tệp image mới này, ta sẽ không cần phải cấu hình lại từ đầu nữa.

6.8. Nhân bản các máy ảo đã được cấu hình

Nhân bản máy ảo là thao tác lưu trữ lại toàn bộ dữ liệu, trạng thái cấu hình hiện tại của một máy ảo đang chạy. Tất cả nội dung lưu trữ đó được lưu vào một tệp image mới. Khi một máy ảo khác chạy lên từ tệp image mới đó, nó sẽ có dữ liệu, trạng thái cấu hình giống hệt với máy ảo ban đầu.

Sơ đồ:



Hình 6.5 – Sơ đồ định nghĩa về Nhân bản máy ảo.

Chú ý: Image mới tạo mặc định sẽ được lưu trên máy ảo ban đầu, sau đó được upload lên Walrus của Eucalyptus.

6.8.1. Các bước thực hiện trên máy Front-end.

Bước 1: Tạo volume.

Bước 2: Attach volume vừa tạo và thiết bị lưu trữ ngoại vi vào máy ảo đang chạy.

Bước 3: Định dạng hệ thống tập tin (file system) ext3 cho thiết bị lưu trữ vừa attach.

Bước 4: Gửi khóa xác nhận (credential key, ví dụ euca2-xxxxx-x509.zip) của người dùng hiện tại vào máy ảo.

6.8.2.Các bước thực hiện trên máy ảo

Bước 1: Đăng nhập vào máy ảo.

Bước 2: Định dạng lại hệ thống tập tin (file system) ext3 cho thiết bị lưu trữ.

Bước 3: Cài Euca2ools và một số gói phần mềm cần thiết.

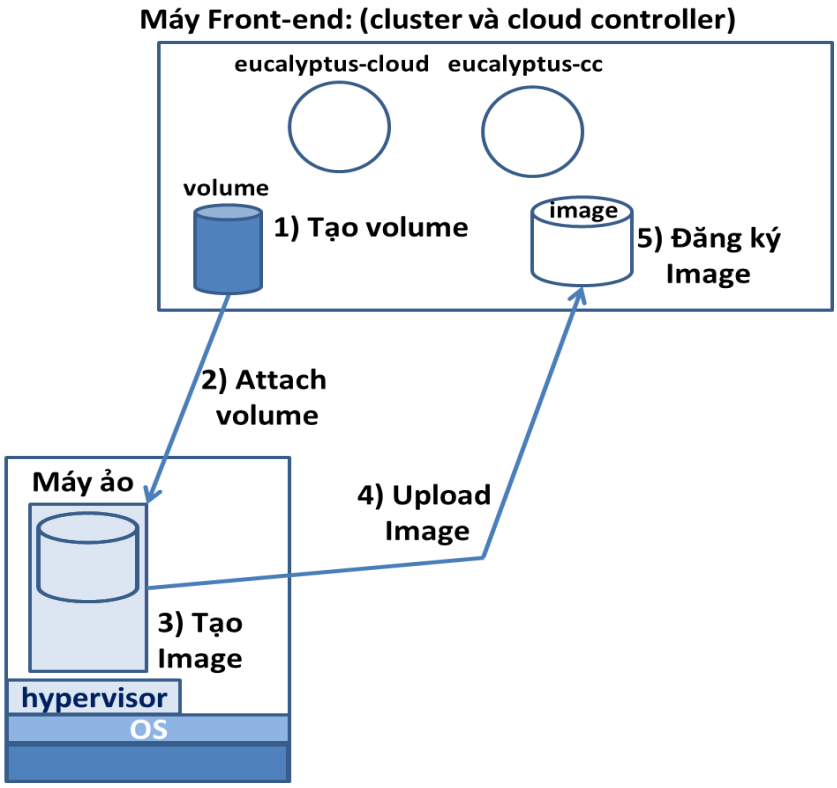
Bước 4: Thay đổi password của người dùng root.

Bước 5: Thực hiện thao tác bundle volume (tạo ra image mới).

Bước 6: Upload tệp image vừa tạo lên Walrus của hệ thống Eucalyptus.

Bước 7: Register tệp image vừa upload.

Dưới đây là sơ đồ tóm tắt quá tình tạo ra tệp Image từ máy ảo đang chạy.



Hình 6.6 – Tóm tắt quá trình tạo ra tệp image.

TÀI LIỆU THAM KHẢO

[1] http://www.ibm.com/developerworks/library/ar-cloudaws1/

[2] Ian Foster, Yong Zhao, Ioan Raicu, Shiyong Lu, *Cloud Computing and Grid Computing 360-Degree Compared*, Grid Computing Environments Workshop, 2008.

[3] Peter Mell, Timothy Grance. The NIST Definition of Cloud Computing.

[4] Borko Furht - Department of Computer and Electrical Engineering and Computer Science - Florida Atlantic University. Cloud Computing Fundamentals.

[5] Sun MicroSystems, Introduction to Cloud ComputingArchitecture 1st Edition, June 2009.

[6] John W.Rittinghouse – James F.Ransome ,Cloud Computing - Implementation, management and security CRC Press.

[7] Judith Hurwitz – Robin Bloor – Marcia Kaufman – Fern Haipper, *Cloud Computing for Dummies*.

[8] Voas, J., & Zhang, J. (March/April 2009). Cloud computing: New wine or just a new bottle? IEEEITPro, 15–17.

[9] Nurmi, D.l, Wolski, R., Grezegorczyk, C., Obertelli, G., & Soman, S. (2009). The eucalyptus open-source cloud-computing system.

[10] Eucalyptus Open-Source Cloud Computing Infrastructure - An Overview.

[11] http://cnsa-cloud-project.wikidot.com/eucalyptus

[12] Peter Sempolinski and Douglas Thain - University of Notre Dame. A Comparison and Critique of Eucalyptus, OpenNebula and Nimbus.

[13] http://www.eucalyptus.com/resources/AmazonAWS.

[14] http://nagios.sourceforge.net/docs/3_0/monitoring-linux.html

[15] http://blogs.plexibus.com/

[16] http://nagios.sourceforge.net/docs/3_0/config.html

PHỤ LỤC

Script cài đặt Nagios.

Hướng dẫn: Đoạn script này cần được lưu vào một tệp .sh (ví dụ: nagios_install.sh). Sau đó copy tệp nagios-3.3.1.tar.gz và nagios-plugins-1.4.15.tar.gz vào cùng thư mục với tệp cài đặt ở trên (nagios_install.sh).

```
#!/bin/ksh

NAGIOSSRC=nagios-3.3.1
NAGIOSPLUGINSRC=nagios-plugins-1.4.15
NAGIOSCONTACTSCFG=/usr/local/nagios/etc/objects/contacts.cfg
NAGIOSPASSWD=/usr/local/nagios/etc/htpasswd.users
PASSWD=cluster
OS=foo

function buildNagios {
 if [ -e $NAGIOSSRC.tar.gz ]
 then
  echo "found $NAGIOSSRC.tar.gz building and installing Nagios"
 else
  echo "could not find $NAGIOSSRC.tar.gz in current directory."
  echo "Please run $0 in the same directory as the source files."
  exit 1
 fi
 echo "Extracting Nagios..."
 tar zxf $NAGIOSSRC.tar.gz
 cd nagios
 echo "Configuring Nagios..."
```

```
 if ./configure --with-command-group=nagcmd  > config.LOG.$$ 2>&1
 then
  echo "Making  Nagios..."
  if make all -j8 > make.LOG.$$ 2>&1
  then
    make install > make.LOG.$$ 2>&1
    make install-init > make.LOG.$$ 2>&1
    make install-config > make.LOG.$$ 2>&1
    make install-commandmode  > make.LOG.$$ 2>&1
    make install-webconf > make.LOG.$$ 2>&1
  else
    echo "make  all failed.  See log:"
    echo "$NAGIOSSRC/make.LOG.$$"
    exit 1
  fi
 else
  echo "configure of Nagios failed. Please read
   $NAGIOSSRC/config.LOG.$$  for details."
  exit 1
 fi
 echo "Done  Making  Nagios!"
 cd ..
}



function buildNagiosPlug  {

 if [ -e $NAGIOSPLUGINSRC.tar.gz  ]
```

```
then
  echo "found $NAGIOSPLUGINSRC.tar.gz  building and installing Nagios"
else
  echo "could not find $NAGIOSPLUGINSRC.tar.gz  in current directory."
  echo "Please run $0 in the same directory as the source files."
  exit 1
fi
echo "Extracting Nagios Plugins..."
tar zxf $NAGIOSPLUGINSRC.tar.gz
cd nagios-plugins-1.4.15
echo "Configuring Nagios Plugins..."
if ./configure --with-nagios-user=nagios --with-nagios-group=nagios --enable-
   ssl
    #-prefix=/usr/local/nagios > config.LOG.$$ 2>&1
then
  echo "Making Nagios Plugins..."
  if make -j8 > make.LOG.$$ 2>&1
  then
    make install > make.LOG.$$ 2>&1
  else
   echo "Make failed of Nagios plugins. See
   $NAGIOSPLUGINSRC/make.LOG.$$"
    exit 1
  fi
else
  echo "configure of Nagios plugins failed. See config.LOG.$$"
  exit 1
fi
echo "Successfully built and installed Nagios Plugins!"
```

```
  cd ..

}


function configNagios {
  echo "We'll now configure Nagios."
  LOOP=1
  while [[ $LOOP -eq 1 ]]
  do
    echo "You'll need to put in a user name. This should be the person"
    echo "who will be receiving alerts. This person should have an account"
    echo "on this server. "
    print "Type in the userid of the person who will receive alerts (e.g. bob)> \c"
    read NAME
    print "What is ${NAME}'s email?> \c"
    read EMAIL
    echo
    echo
    echo "Nagios alerts will be sent to $NAME at $EMAIL"
    print "Is this correct? [y/N] \c"
    read YN
    if [[ "$YN" = "y" ]]
    then
      LOOP=0
    fi
  done
  if [ -r $NAGIOSCONTACTSCFG ]
  then
```

```
  perl -pi -e "s/nagiosadmin/$NAME/g"  $NAGIOSCONTACTSCFG
  EMAIL=$(echo  $EMAIL | sed s/\@/\\\\@/g)
  perl -pi -e "s/nagios\@localhost/$EMAIL/g"  $NAGIOSCONTACTSCFG
 else
  echo "$NAGIOSCONTACTSCFG  does not exist"
  exit 1
 fi

 echo "setting ${NAME}'s  password to be 'cluster' in Nagios"
 echo "    you can change this later by running:  "
 echo "    htpasswd  -c $NAGIOSPASSWD  $Name)'"
 htpasswd  -bc $NAGIOSPASSWD  $NAME  cluster
 if [ "$OS"  = "rh"  ]
 then
  service httpd restart
 fi

}


function preNagios {

 if [ "$OS"  = "rh"  ]
 then
  echo "making  sure prereqs are installed"
  yum  -y install httpd gcc glibc glibc-common gd gd-devel  perl-TimeDate
  /usr/sbin/useradd  -m nagios
  echo $PASSWD  | passwd  --stdin nagios
  /usr/sbin/groupadd  nagcmd
```

```
    /usr/sbin/usermod -a -G nagcmd  nagios
    /usr/sbin/usermod -a -G nagcmd  apache
  fi

}
function postNagios {
  if [ "$OS"  = "rh"  ]
  then
    chkconfig --add nagios
    chkconfig nagios on
    # touch this file so that if it doesn't exist we won't  get errors
    touch /var/www/html/index.html
    service nagios start
  fi
  echo "You  may now be able to access Nagios at the URL below:"
  echo "http://localhost/nagios"

}




if [ -e /etc/redhat-release ]
then
  echo "installing  monitoring on Red Hat system"
  OS=rh
fi

# make  sure you're root:
ID=$(id -u)
```

```
if [ "$ID" != "0" ]
then
  echo "Must run this as root!"
  exit
fi

preNagios
buildNagios
buildNagiosPlug
configNagios
postNagios
```