|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI  VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG  ──────── \* ───────  ĐỒ ÁN  **TỐT NGHIỆP ĐẠI HỌC**  NGÀNH HỆ THỐNG THÔNG TIN  **NGHIÊN CỨU VÀ XÂY DỰNG PLATFORM TRIỂN KHAI ỨNG DỤNG AUTOSCALING**   |  |  | | --- | --- | | Sinh viên thực hiện | : **Nguyễn Bá Cường** | | Lớp | : **CNTT-TT 2.3 – K56** | | Giáo viên hướng dẫn | : **TS. Nguyễn Bình Minh** |   Hà Nội, 05-2016 |

**PHIẾU GIAO NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP**

**1. Thông tin về sinh viên**

Họ và tên: **NGUYỄN BÁ CƯỜNG**

Điện thoại liên lạc: **0164-868-6894** Email: [**cuongnb14@gmail.com**](mailto:cuongnb14@gmail.com)

Lớp: **CNTT-TT 2.3 – K56** Hệ đào tạo: **Đại học chính quy**

Đồ án tốt nghiệp thực hiện tại: **Bộ môn Hệ thống thông tin, Viện Công nghệ thông tin và truyền thông, Đại học Bách Khoa Hà Nội.**

Thời gian làm ĐATN: Từ ngày 21/02/2016 đến 31/05/2016

Thực tập tại **Viện nghiên cứu quốc tế về khoa học và kĩ thuật tính toán (ICSE)**, thuộc trường đại học Bách Khoa Hà Nội.

Thời gian thực tập: Từ ngày 26/06/2015 đến 31/05/2016.

**2. Mục đích nội dung của ĐATN**

Xây dựng một nền tảng triển khai ứng dụng web có khả năng tự động mở rộng

**3. Các nhiệm vụ cụ thể của đồ án**

1. Nghiên cứu docker, đóng gói ứng dụng web bằng docker

2. Nghiên cứu và xây dựng nền tảng triển khai ứng dụng web từ mã nguồn

3. Xây dựng các dịch vụ giúp autoscaling ứng dụng web

**4. Lời cam đoan của sinh viên**

Tôi – *Nguyễn Bá Cường* - cam kết ĐATN là công trình nghiên cứu của bản thân tôi dưới sự hướng dẫn của TS. Nguyễn Bình Minh.

Các kết quả nêu trong ĐATN là trung thực, không phải là sao chép toàn văn của bất kỳ công trình nào khác.

Hà Nội, ngày tháng năm

Tác giả ĐATN

*Nguyễn Bá Cường*

**5. Xác nhận của giáo viên hướng dẫn về mức độ hoàn thành của ĐATN và cho phép bảo vệ:**

Hà Nội, ngày tháng năm

Giáo viên hướng dẫn

*TS. Nguyễn Bình Minh*

# TÓM TẮT NỘI DUNG ĐỒ ÁN TỐT NGHIỆP

Đồ án tiến hành nghiên cứu và xây dựng một nền tảng triển khai ứng dụng web có khả năng autoscaling.

**Đồ án giải quyết các vấn đề:**

1. Nghiên cứu docker, đóng gói ứng dụng web bằng docker
2. Nghiên cứu và xây dựng hệ thống mesosphere trên server cluster
3. Xây dựng hệ thống load balancing HAproxy cho ứng dụng web
4. Xây dựng các dịch vụ autoscaling cho ứng dụng web
5. Xây dựng giao diện người dùng cho phép người dùng triển khai ứng dụng web từ mã nguồn
6. Thực nghiệm và đánh giá kết quả

**Nội dung của đồ án tốt nghiệp gồm 5 chương.**

**Chương 1 – Mở đầu:** Trình bày các vấn đề gặp phải, cách giải quyết vấn đề và mục đích của đề tài

**Chương 2 – Kiến trúc hệ thống:** Kiến trúc hệ thống của một nền tảng autoscaling và kiến trúc của ứng dụng web

**Chương 3 – Phân tích và thiết kế hệ thống:** Cung cấp sự mô tả đầy đủ về tất cả các chức năng chính cũng như giới hạn của hệ thống.

**Chương 4 – Kiểm thử hệ thống:** Trình bày kịch bản kiểm thử, kết quả và đánh giá kiểm thử

**Chương 5 – Kết luận và hướng phát triển:** Tổng kết các kết quả đạt được. Trình bày các hướng phát triển trong tương lai

# ABSTRACT OF THESIS

This thesis researches and design a platform deploy web application with the autoscaling ability.

**The thesis solves issues:**

1. Research docker, containerize web application by docker
2. Research and design mesosphere system on server cluster
3. Design load balancing system HAproxy for web application
4. Design autoscaling services for web application
5. Design user interface allow user deploy web application from source code.
6. Experiment and evaluating results.

**This graduation thesis consists of 5 chapters:**

**Chapter 1 - Introduction:** The encountered issues and solutions and the target of this thesis

**Chapter 2 -** **Architecture system :** Architecture system of a platform autoscaling and architecture of web application

**Chapter 3 - Analyzing and designing system:** Providing full description of all main functions and the system’s limit.

**Chapter 4 - Testing System:** Presenting the test’s scenario and result, evluating this test

**Chapter 5 - Conclusion and Future work:** Conclusion based on collected results. Present

develop trend in the future.

# LỜI CẢM ƠN

Tôi xin cảm ơn các thầy cô trong trường đại học Bách Khoa Hà Nội, đặc biệt các thầy cô thuộc Viện công nghệ thông tin và truyền thông đã truyền đạt kiến thức và kinh nghiệm cho tôi trong năm năm đại học.

Tôi xin cảm ơn **TS. Nguyễn Bình Minh** đã hướng dẫn tận tình, theo sát quá trình thực hiện đồ án tốt nghiệp. Những kinh nghiệm, kĩ năng, tác phong nghiên cứu khoa học của thầy sẽ giúp ích cho tôi rất nhiều trong công việc cũng như trong cuộc sống.

Tôi xin cảm ơn các anh chị cán bộ viện ICSE đã tạo điều kiện thuận lợi để tôi nghiên cứu và hoàn thành đồ án.

Tôi xin cảm ơn anh **Trần Hữu Cường** đã giúp tôi trong việc trao đổi kiến thức về server cluster và docker

Tôi xin cảm ơn các bạn thuộc nhóm Cloud: **Bùi Ngọc Luân, Phạm Công Huân** cùng các bạn sinh viên K57 thực tập tại viện ICSE đã giúp đỡ, trao đổi kiến thức trong quá trình thực hiện đồ án.

Cuối cùng, con xin cảm ơn bố mẹ và các anh chị vì đã luôn quan tâm và chăm sóc con. Đó sẽ là điểm tựa, động lực để con đi tiếp.

Hà Nội, tháng 05 năm 2016.

Sinh viên thực hiện

*Nguyễn Bá Cường*

MỤC LỤC

[TÓM TẮT NỘI DUNG ĐỒ ÁN TỐT NGHIỆP 3](#_Toc452025656)

[ABSTRACT OF THESIS 4](#_Toc452025657)

[LỜI CẢM ƠN 5](#_Toc452025658)

[MỤC LỤC 6](#_Toc452025659)

[DANH MỤC HÌNH VẼ 9](#_Toc452025660)

[DANH MỤC BẢNG 11](#_Toc452025661)

[DANH MỤC TỪ VIẾT VIẾT TẮT VÀ THUẬT NGỮ 12](#_Toc452025662)

[CHƯƠNG 1: MỞ ĐẦU 13](#_Toc452025663)

[1.1 Đặt vấn đề 13](#_Toc452025664)

[1.2 Cách tiếp cận 14](#_Toc452025665)

[1.2.1 Cở sở hạ tầng 14](#_Toc452025666)

[1.2.2 Chiến thuật tăng giảm 17](#_Toc452025667)

[1.2.3 Tổng quan về một hệ thống autoscaling: 18](#_Toc452025668)

[1.3 Giới thiệu một số hệ thống autoscaling thực tế 19](#_Toc452025669)

[1.3.1 Openshift 19](#_Toc452025670)

[1.3.2 Microsoft Azure 19](#_Toc452025671)

[1.4 Mục đích đề tài 20](#_Toc452025672)

[CHƯƠNG 2: KIẾN TRÚC HỆ THỐNG 21](#_Toc452025673)

[2.1 Mô hình hệ thống autoscaling 21](#_Toc452025674)

[2.1.1 Monitoring 21](#_Toc452025675)

[2.1.2 Decider 22](#_Toc452025676)

[2.1.3 Scaler 23](#_Toc452025677)

[2.1.4 Dashboard 23](#_Toc452025678)

[2.2 Kiến trúc của một ứng dụng web trong hệ thống 23](#_Toc452025679)

[CHƯƠNG 3: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG 25](#_Toc452025680)

[3.1 Phân tích chức năng 25](#_Toc452025681)

[3.1.1 Các tác nhân tham gia hệ thống 25](#_Toc452025682)

[3.1.2 Danh sách các use case 25](#_Toc452025683)

[3.1.3 Mô tả các ca sử dụng 27](#_Toc452025684)

[3.2 Phân tích cấu trúc 41](#_Toc452025685)

[3.2.1 Lớp Boundaries (các lớp biên hay các lớp đối thoại): 42](#_Toc452025686)

[3.2.2 Lớp Controls (các lớp điều khiển ): 42](#_Toc452025687)

[3.2.3Lớp Entities (các lớp thực thể hay các lớp lĩnh vực): 43](#_Toc452025688)

[3.3 Phân tích hành vi người dùng 45](#_Toc452025689)

[3.3.1 Đăng nhập 45](#_Toc452025690)

[3.3.2 Thêm ứng dụng web mới 46](#_Toc452025691)

[3.3.3 Cập nhật ứng dụng web 47](#_Toc452025692)

[3.3.4 Thay đổi trạng thái ứng dụng web 48](#_Toc452025693)

[3.3.5 Thêm luật cho ứng dụng 49](#_Toc452025694)

[3.3.6 Xem biểu đồ nguồn tài nguyên 50](#_Toc452025695)

[3.3.6 Thêm ứng dụng database 50](#_Toc452025696)

[3.3.7 Thao tác với ứng dụng database 51](#_Toc452025697)

[3.4 Thiết kế cơ sở dữ liệu 52](#_Toc452025698)

[3.5 Thiết kế giao diện người dùng 53](#_Toc452025699)

[3.5.1 Bố cục giao diện 53](#_Toc452025700)

[3.5.2 Danh sách các giao diện 54](#_Toc452025701)

[3.5.3 Một số hình ảnh giao diện 54](#_Toc452025702)

[3.6 Thiết kế console tool 58](#_Toc452025703)

[CHƯƠNG 4: KIỂM THỬ HỆ THỐNG 59](#_Toc452025704)

[4.1 Cài đặt môi trường kiểm thử 59](#_Toc452025705)

[4.2 Tiến hành kiểm thử 59](#_Toc452025706)

[4.2.1 Tạo ứng dụng database 59](#_Toc452025707)

[4.2.2 Tạo một ứng dụng web 60](#_Toc452025708)

[4.2.3 Chức năng autoscaling 61](#_Toc452025709)

[CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 64](#_Toc452025710)

[5.1 Kết luận 64](#_Toc452025711)

[5.2 Hướng phát triển 65](#_Toc452025712)

[5.2.1 Áp dụng phương pháp dự đoán tiêu dùng tài nguyên (Prediction Resource Consumption) 65](#_Toc452025713)

[5.2.2 Migrate các instance dồn về một số node 65](#_Toc452025714)

[5.2.3 Autoscaling cho database 65](#_Toc452025715)

[TÀI LIỆU THAM KHẢO 66](#_Toc452025716)

[PHỤ LỤC 67](#_Toc452025717)

[Phần lục 1: Cài đặt và cấu hình hệ thống mesosphere 67](#_Toc452025718)

[Cài đặt 67](#_Toc452025719)

[Cấu hình 67](#_Toc452025720)

[Phần lục 2: Dockerfile cho apache-php images 68](#_Toc452025721)

[Phần lục 3: Hướng dẫn sử dụng console tool (ASClient) 68](#_Toc452025722)

# DANH MỤC HÌNH VẼ

[Hình 1: Nguồn tài nguyên cố định 13](#_Toc452024071)

[Hình 2: Nguồn tài nguyên động 14](#_Toc452024072)

[Hình 3: Docker với Virtual machine 15](#_Toc452024073)

[Hình 4: Kiến trúc của mesosphere 16](#_Toc452024074)

[Hình 5: Thành phần cơ bản của hệ thống autoscaling 18](#_Toc452024075)

[Hình 6: Kiến trúc hệ thống autoscaling 21](#_Toc452024076)

[Hình 7: Kiến trúc ứng dụng có khả năng scale 23](#_Toc452024077)

[Hình 8: UC01 Tổng quan hệ thống 27](#_Toc452024078)

[Hình 9: UC02 Đăng kí 27](#_Toc452024079)

[Hình 10: UC03 Đăng nhập 28](#_Toc452024080)

[Hình 11: UC04 Quản lí database 28](#_Toc452024081)

[Hình 12: UC04.1: Thêm database 29](#_Toc452024082)

[Hình 13: UC04.2: Xem thông tin database 29](#_Toc452024083)

[Hình 14: UC04.3: Đổi mật khẩu root 30](#_Toc452024084)

[Hình 15: UC04.4: Thao tác với database 30](#_Toc452024085)

[Hình 16: UC04.5: Xóa database 31](#_Toc452024086)

[Hình 17: UC05: Quản lí ứng dụn web 31](#_Toc452024087)

[Hình 18: UC05.1: Thêm ứng dụng 32](#_Toc452024088)

[Hình 19: UC05.2: Xem thông tin ứng dụng 32](#_Toc452024089)

[Hình 20: UC05.3: Cập nhật ứng dụng 33](#_Toc452024090)

[Hình 21: UC05.4: Thay đổi trạng thái ứng dụng 34](#_Toc452024091)

[Hình 22: UC05.4.1: Redeploy ứng dụng 34](#_Toc452024092)

[Hình 23: UC05.4.2: Scale ứng dụng 35](#_Toc452024093)

[Hình 24: UC05.4.3 Chạy / Dừng ứng dụng 35](#_Toc452024094)

[Hình 25: UC05.4.4 Bật/Tắt Autoscaling 36](#_Toc452024095)

[Hình 26: UC05.5: Xóa ứng dụng 36](#_Toc452024096)

[Hình 27: UC05.6: Quản lí bộ luật 37](#_Toc452024097)

[Hình 28: UC05.6.1: Thêm luật 37](#_Toc452024098)

[Hình 29: UC05.6.2 Cập nhật luật 38](#_Toc452024099)

[Hình 30: UC05.6.3: Xóa luật 38](#_Toc452024100)

[Hình 31: UC05.7: Xem biểu đồ 39](#_Toc452024101)

[Hình 32: UC06: Quản lí tài khoản người dùng 40](#_Toc452024102)

[Hình 33: UC06.1: Thêm người dùng 40](#_Toc452024103)

[Hình 34: UC06.2: Xóa người dùng 41](#_Toc452024104)

[Hình 35: UC06.3: Cập nhật thông tin 41](#_Toc452024105)

[Hình 36: Sequence đăng nhập 45](#_Toc452024106)

[Hình 37: Sequence thêm ứng dụng web 46](#_Toc452024107)

[Hình 38: Sequence Cập nhập ứng dụng web 47](#_Toc452024108)

[Hình 39: Sequence thay đổi trạng thái ứng dụng 48](#_Toc452024109)

[Hình 40: Sequence thêm luật cho ứng dụng 49](#_Toc452024110)

[Hình 41: Sequence xem biểu đồ nguồn tài nguyên 50](#_Toc452024111)

[Hình 42: Sequence thêm ứng dụng database 51](#_Toc452024112)

[Hình 43: Sequence thao tác với database 51](#_Toc452024113)

[Hình 44: Cơ sở dữ liệu tổng quát 52](#_Toc452024114)

[Hình 45: Bố cục giao diện 53](#_Toc452024115)

[Hình 46: Giao diện trang chủ 54](#_Toc452024116)

[Hình 47: Giao diện quản lí ứng dụng web 55](#_Toc452024117)

[Hình 48: Giao diện quản lí luật 56](#_Toc452024118)

[Hình 49: Giao diện xem thông tin ứng dụng web 57](#_Toc452024119)

[Hình 50: Giao diện xem biểu đồ ứng dụng 57](#_Toc452024120)

[Hình 51: Trình tự thực hiện một dòng lệnh 58](#_Toc452024121)

[Hình 52: Biểu đồ metrics 63](#_Toc452024122)

# DANH MỤC BẢNG

[Bảng 1: Các thông số của một luật 22](#_Toc452024123)

[Bảng 2: Các tác nhân của hệ thống 25](#_Toc452024124)

[Bảng 3: Danh sách các usecase 26](#_Toc452024125)

[Bảng 4: Danh sách các bảng cơ sở dữ liệu 53](#_Toc452024126)

[Bảng 5: Danh sách giao diện 54](#_Toc452024127)

[Bảng 6: Các chức năng kiểm thử 59](#_Toc452024128)

[Bảng 7: Kết quả thực nghiệm chức năng tạo Database 60](#_Toc452024129)

[Bảng 8: Kết quả kiểm nghiệm chức năng tạo ứng dụng 61](#_Toc452024130)

[Bảng 9: Bộ luật kiểm thử 62](#_Toc452024131)

[Bảng 10: Kết quả kiểm thử chức năng autoscaling 62](#_Toc452024132)

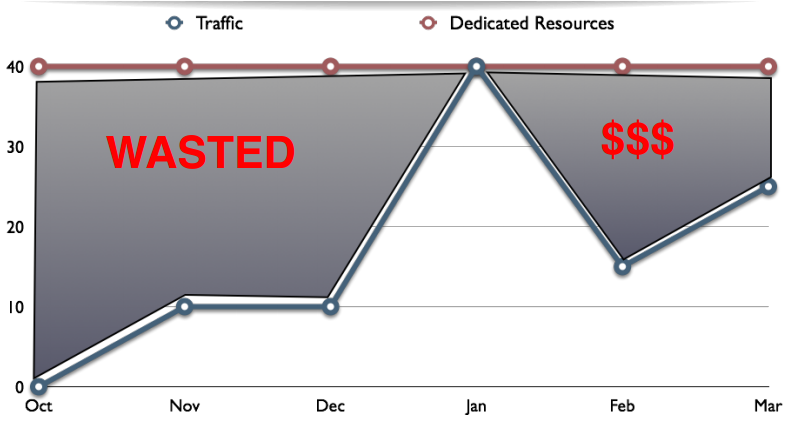
# DANH MỤC TỪ VIẾT VIẾT TẮT VÀ THUẬT NGỮ

|  |  |  |
| --- | --- | --- |
| Chữ viết tắt | Viết đầy đủ | Ý nghĩa |
|  | Instance | Là một trong những container đang chạy của ứng dụng |
|  | Server cluster | Là một mô hình bao gồm nhiều server riêng lẻ được liên kết và hoạt động cùng với nhau trong một hệ thống |
|  | Docker image /  image | Một template cố định chứa các thư viện và ứng dụng được cài đặt sẵn |
|  | Docker container /  container | Là một instance của docker image |
| REST | Representational State Transfer | Một kiểu kiến trúc được sử dụng trong việc giao tiếp giữa các máy tính |
| LXC | Linux Container | Là môi trường ảo ở tầng hệ điều hành |
| VM | Virtual Machine | Máy tính ảo do phần mềm ảo hóa giả lập |
|  | Hypervisors | Ảo hóa phần cứng |
|  | Scale | Mở rộng ứng dụng (tăng giảm số lượng instance của ứng dụng) |
|  | Mesos task | Một instance của ứng dụng chạy trên mesos |
| PaaS | Platform as a service | Nền tảng hỗ trợ việc triển khai ứng dụng mà không quan tâm đến chi phí hay sự phức tạp của việc trang bị và quản lý các lớp phần cứng và phần mềm bên dưới |
|  | Dockerfile | File dùng để xây dựng docker image |
|  | Time series data | Dữ liệu theo khoảng thời gian và liên tục |
|  | Time series database | Database quản lí dữ liệu time series |

# CHƯƠNG 1: MỞ ĐẦU

## 1.1 Đặt vấn đề

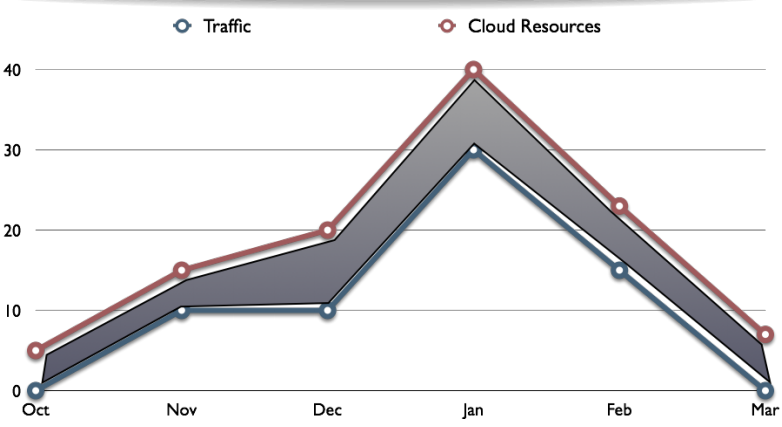
Theo thống kê thì lưu lượng truy cập của hầu hết mọi website là không đồng đều. Ví dụ như các trang web đọc báo thường có lượng truy cập rất nhiều vào buổi sáng, khi có các sự kiện lớn xảy ra như động đất, dịch bệnh,… .Các trang web bán hàng thì thường có lượng truy cập lớn khi có các sự kiện giảm giá, mặt hàng mới về, … Chính vì vậy nếu hệ thống website được thiết kế theo phương thức truyền thống (nguồn tài nguyên có định) thì tài nguyên của server ít nhất phải đáp ứng được lưu lượng truy cập lớn nhất của website để đảm bảo rằng website luôn luôn sẵn sàng đáp ứng được yêu cầu của người dùng. Từ đó nảy sinh ra một vấn đề là khi lưu lượng người truy cập ít hơn sẽ dẫn tới dư thừa tài nguyên của server gây lãng phí tiền bạc.



Hình 1: Nguồn tài nguyên cố định

Giải quyết vấn đề này, ta có thể nghĩ ngay tới một hệ thống có thể co dãn tài nguyên tùy vào nhu cầu sử dụng của người dùng. Để tạo được hệ thống ta cần giải quyết được 2 vấn đề chính:

* Hệ thống phải có cơ sở hạ tầng có cơ chế tăng giảm nguồn tài nguyên của ứng dụng dễ dàng
* Có chiến thuật tăng giảm nguồn tài nguyên phù hợp dựa trên nhu cầu sử dụng



Hình 2: Nguồn tài nguyên động

## 1.2 Cách tiếp cận

### 1.2.1 Cở sở hạ tầng

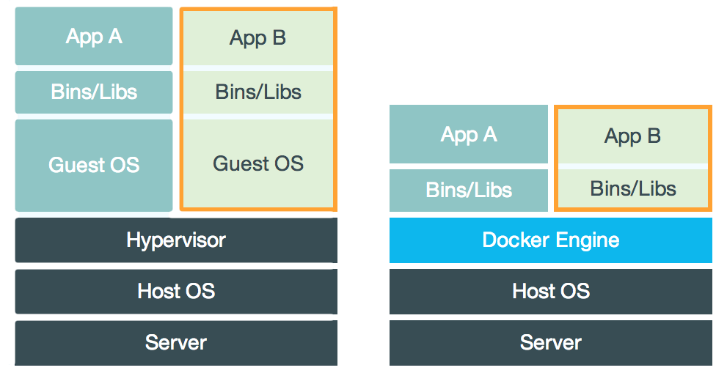
Để giải quyết bài toán cở sở hạ tầng có 2 phương án khá phổ biến. Thứ nhất là đóng gói ứng dụng vào một virtual machine (VM), khi đó nguồn tài nguyên của ứng dụng sẽ phụ thuộc vào nguồn tài nguyên trên mỗi con VM và số lượng VM của ứng dụng. Cách thứ 2 là đóng gói ứng dụng vào docker container thì nguồn tài nguyên sẽ phụ thuộc vào nguồn tài nguyên của mỗi container và số lượng container. Cả 2 cách đều có thể tăng, giảm nguồn tài nguyên của ứng dụng bằng cách tăng, giảm nguồn tài nguyên của mỗi instances (vm hoặc docker container) hoặc số lượng instances của mỗi ứng dụng. Tuy nhiên việc thay đổi số lượng instances phổ biến hơn vì thay đổi nguồn tài nguyên mỗi instances sẽ khiến ứng dụng phải khởi động lại.

#### 1.2.1.1 Giới thiệu về Docker

**Docker** là một Open Platform để xây dựng, vận chuyển và chạy các ứng dụng phân tán(Build-Ship-Run). Ban đầu viết bằng Python, hiện tại đã chuyển sang Go-lang.

Docker đưa ra một giải pháp mới cho vấn đề ảo hóa, thay vì tạo ra các máy ảo con chạy độc lập kiểu hypervisors (tạo phần cứng ảo và cài đặt hệ điều hành lên đó), các ứng dụng sẽ được đóng gói lại thành các Container riêng lẻ. Các Container này chạy chung trên nhân hệ điều hành qua LXC (Linux Containers), chia sẻ chung tài nguyên của máy mẹ, do đó, hoạt động nhẹ và nhanh hơn các máy ảo dạng hypervisors.

**So sánh Docker với Virtual machine:**



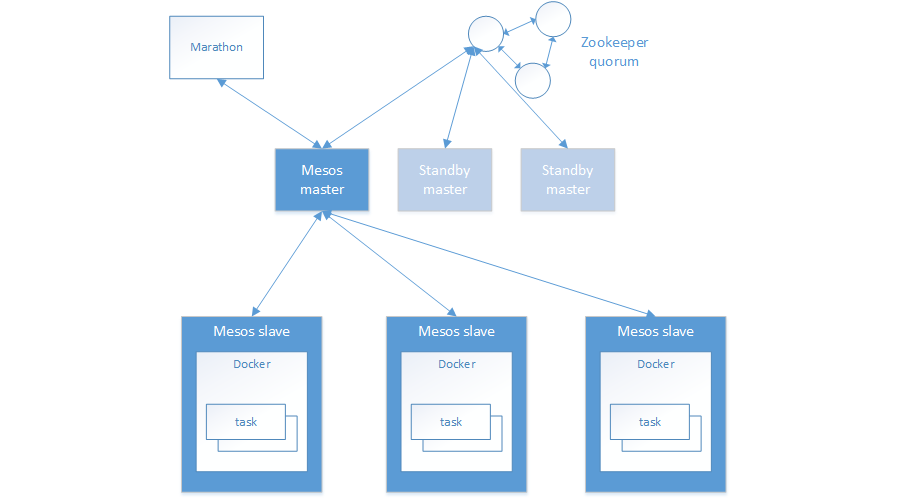
Hình 3: Docker với Virtual machine

Điểm khác biệt chính là các containers sử dụng chung kernel với Host OS nên các thao tác bật, tắt rất nhẹ nhàng, nhanh chóng.

* **Ưu điểm**: nhanh, nhẹ, có thể chia sẻ dễ dàng qua DockerHub

- **Nhược điểm**: còn mới, cập nhật thay đổi thường xuyên

#### 1.2.1.2 Giới thiệu về Mesosphere



Hình 4: Kiến trúc của mesosphere

**Mesosphere** là một giải pháp phần mềm, mở rộng thêm khả năng quản lí cluster của apache mesos bằng cách thêm các thành phần bổ sung để cung cấp một cách mới trong việc quản lí cơ sở hạ tầng máy chủ.

Kết hợp một số thành phần với **Mesos**, như là **Marathon** và **Chronos**, **Mesosphere** cho phép dễ dàng scale ứng dụng

Mesosphere cung cấp một số tính năng như:

* Lập lịch cho ứng dụng
* Scaling
* Khả năng chịu lỗi
* Tự sửa chữa

**Apache mesos**

Apache mesos là một phần mềm quản lí cluster mã nguồn mở, là trung tâm của hệ thống Mesosphere.

Mesos cung cấp một số tính năng như là:

* Mở rộng tới hơn 10.000 node
* Cô lập nguồn tài nguyên cho mỗi task thông qua Linux containers.
* Lập lịch sử dụng nguồn tài nguyên cpu và bộ nhớ hiệu quả
* Đảm bảo tính sẵn sàng cao (nhờ Apache ZooKeeper)
* Có giao diện web cho việc giám sát, theo dõi trạng thái cluster

**Marathon**

Marathon là một framework cho Mesos được thiết kết để khởi động long-running applications. Trong mesosphere nó đóng vai trò thay thế cho hệ thống init truyền thống.

Marathon cung cấp một số tính năng để đơn giản hóa việc chạy một ứng dụng trên môi trường cluster như là:

* Tính sẵn sàng cao
* Có ràng buộc giữa các node
* Giao tiếp với người dùng thông qua REST API
* Giao diện web để quản lí
* Có chức năng scale

#### 1.2.1.3 Giới thiệu về HAproxy

Khi sử dụng nhiều instances để chạy một ứng dụng ta sẽ cần phải có một hệt thống cân bằng tải để phân tán lưu lượng tải trên các instances đó. **Haproxy** là một phần mềm như vậy.

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

### 1.2.2 Chiến thuật tăng giảm

Việc tăng giảm nguồn tài nguyên cho ứng dụng có thể lợi dung chu kỳ truy cập ứng dụng. Ví dụ như một trang web tin tức có số lượng người đọc rất nhiều vào buổi sáng, ít vào ban đêm thì ta sẽ có định số lượng instances cho ứng dụng vào buổi sáng là 6 instances, buổi đêm là 1, các buổi còn lại là 3. Tuy nhiên cách làm này không đảm bảo là sẽ có không dư thừa xảy ra hơn nữa không phải trang web nào cũng theo một chu kì nhất định hoặc nếu có thì đôi khi cũng sẽ có ngoại lệ.

Có một cách khác hay hơn đó là sẽ có một hệ thống monitoring để theo dõi nguồn tài nguyên sửa dụng của hệ thống sau đó sẽ áp dụng một thuật toán để tăng giảm nguồn tài nguyên cho phù hợp.

### 1.2.3 Tổng quan về một hệ thống autoscaling:



Hình 5: Thành phần cơ bản của hệ thống autoscaling

* Hệ thống autoscaling gồm có 3 thành phần chính:
* **Monitor**: Theo dõi thông tin sử dụng tài nguyên của ứng dụng
* **Decider**: Dựa vào thông tin của Monitor cung cấp để ra quyết định tăng giảm nguồn tài nguyên cho ứng dụng sau đó gửi tới Scaler
* **Scaler**: Thực thi quyết định do Decider cung cấp

Có 2 điểm cần chú ý ở đây. Thứ nhất là hệ thống monitor phải tốt, giám sát ứng dụng không quá lâu so với thực tế và phải có cơ chế lưu trữ phù hợp để decider có thể dễ dàng truy vấn. Thứ 2 là giải thuật mà decider sử dụng sẽ quyết định tới chất lượng của hệ thống autoscaling. Hiện nay có một số kĩ thuật được sử dụng có decider như là Reinforcement Learning (Q-Learning), Queuing Theory… và một kĩ thuật đơn giản khá phổ biển, đạt hiệu quả tốt là Static Threshold-based Rules.

* Giới thiệu kĩ thuật **Static Threshold-based Rules**:

Là kĩ thuật rất phổ biến. Đơn giản, trực quan hấp hẫn người sử dụng. Tuy nhiên việc thiết lập ngưỡng yêu cầu phải hiểu biết sâu về xu hướng khối lượng công việc.

Số lượng VMs (hoặc containers) trong ứng dụng đích sẽ thay đổi dựa trên tập các luật. Luật có thể dựa trên một hoặc nhiều thông số hiệu suất như là tốc độ các request, CPU load, .... Mỗi luật sẽ liên quan đến một số tham số do người dùng định nghĩa, cụ thể là: 1 ngưỡng trên thrUp,1 ngưỡng dưới thrDown, và 2 giá trị thời gian vUp và vDown để xác định điều kiện phải được đáp ứng bao lâu để kích hoạt hành động scaling, 2 khoảng thời gian tiếp theo (tùy chọn) sau khi thực hiện scaling inUp và inDown:

if x >thrUp for vUp seconds then

n = n + s and

do nothing for inUp seconds

if x <thrDown for vDown seconds then

n = n − s and

do nothing for inDown seconds

* Ưu điểm của hệ thống autoscaling:
* Đối với các công ty sử dụng cơ sở hạ tầng các máy chủ web của riêng họ , autoscaling thường có nghĩa là cho phép một vài máy chủ ở trạng thái sleep trong thời gian tải thấp, giúp tiết kiệm chi phí điện (cũng như chi phí nước nếu dùng nếu sử dụng hệ thống làm mát bằng nước). Còn với các công ty sử dụng cơ sở là tầng là cloud, autoscaling đồng nghĩa với việc chi phí thấp, vì hầu hết các nhà cung cấp điện toán đám mây tính phí dựa trên tổng số nguồn tài nguyên sử dụng chứ không phải là tài nguyên tối đa.
* Autoscaling khác với việc cố định lượng tài nguyên cố định theo hằng giờ, hằng ngày, hoặc theo chu kì đó là nó điều chỉnh nguồn tài nguyên theo nhu cầu thực tế.
* Nhược điểm:

- Cần cài đặt thêm một số dịch vụ giúp hệ thống autoscaling hoạt động

## 1.3 Giới thiệu một số hệ thống autoscaling thực tế

### 1.3.1 Openshift

OpenShift của hãng Red Hat là một Platform-as-a-Service (PaaS) cho phép các nhà phát triển nhanh chóng phát triển, host , và scale ứng dụng trong môi trường điện toán đám mây.

Chức năng scale của hệ thống này dựa trên lưu lượng truy cập trang web mà HAProxy cung cấp. Cách làm này sẽ có nhược điểm là không phải tất cả lượt truy cập cũng có nguồn tài nguyên sử dụng như nhau, có lượt thì cần nhiều cpu, memory có lượt lại không nhiều, vì vậy sẽ có một số trường hợp scale không hợp lí.

### 1.3.2 Microsoft Azure

Azure là nền tảng điện toán đám mây linh hoạt cho phép bạn nhanh chóng xây dựng, triển khai và quản lý các ứng dụng thông qua mạng lưới toàn cầu của trung tâm dữ liệu Microsoft.

Azure luôn đảm bảo tính sẵn sàng và có thiết kế tải cân bằng và có khả năng tự phục hồi khi phần cứng có sự cố. Bạn có thể sử dụng bất kỳ ngôn ngữ, công cụ hay nền tảng nào để xây dựng các ứng dụng. Và bạn có thể tích hợp các ứng dụng trên đám mây công cộng của bạn với môi trường IT có sẵn.

Hệ thống trên cũng cung cấp chức năng autoscaling dựa trên luật (sử dụng kĩ thuật Static Threshold-based Rules) áp dụng cho cpu và memory hoặc autoscaling dựa trên số lượng truy cập đang trong hàng đợi. Tuy nhiên các ứng dụng được đóng gói trong VM nên thao tác tắt đi bật lại khó khăn hơn khi sử dụng container.

## 1.4 Mục đích đề tài

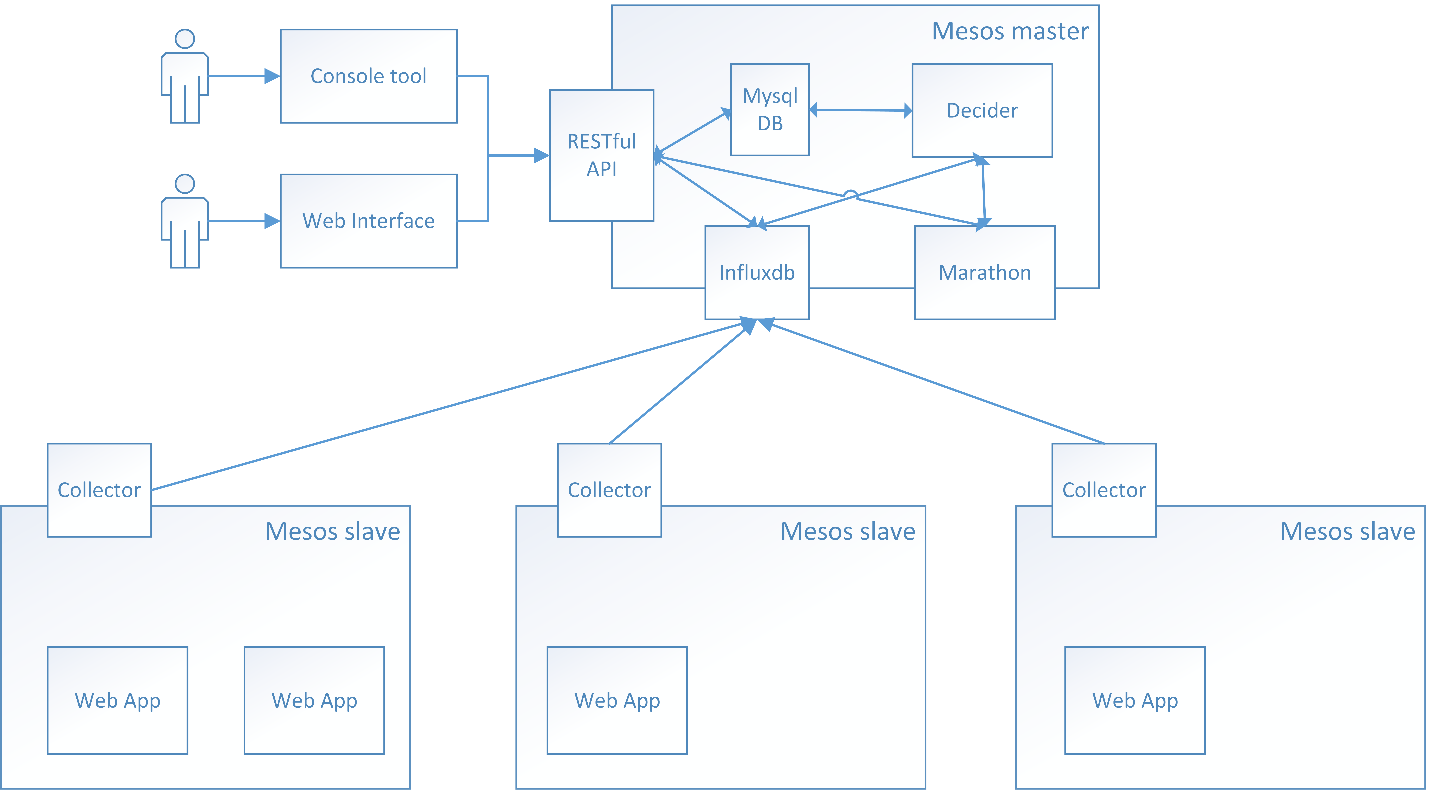
Mục đích của đồ án này là dựa trên docker và mesosphere, tạo ra một dịch vụ nền tảng giúp người dùng triển khai những ứng dụng web có thể scale từ mã nguồn ứng dụng web của họ, và cung cấp những công cụ cần thiết cho việc chạy và quản lí ứng dụng web như MySQL, PHPMyadmin.

Các chức năng chính của hệ thống:

* Xây dựng, đóng gói ứng dụng web vào docker container dựa trên đường dẫn github chứa mã nguồn ứng dụng
* Cung cấp chức năng autoscaling (sử dụng kĩ thuật Static Threshold-based Rules) cho ứng dụng dựa trên cpu và memory sử dụng của ứng dụng
* Cung cấp biểu đồ trực quan về cpu, mem, số lượng instances của ứng dụng theo thời gian để giúp người dùng điểu chỉnh luật phù hợp
* Cho phép người dùng quản lí các database MySQL
* Cung cấp PHPMyAdmin cho phép người dùng thao tác với database của họ

# CHƯƠNG 2: KIẾN TRÚC HỆ THỐNG

## 2.1 Mô hình hệ thống autoscaling



Hình 6: Kiến trúc hệ thống autoscaling

Hệ thống autoscaling được chia làm 4 thành phần chính và được chạy trên mesosphere:

**Phần 1: Monitoring** – Gồm các collector service được gán vào mỗi con mesos slave và influxdb

**Phần 2: Decider** – Lấy dữ liệu monitoring từ influxdb sau đó áp dụng các luật do người dùng định nghĩa (được lưu trong mysql db) và ra quyết định scale

**Phần 3: Scale** – Chính là marathon framework, có nhiệm scale các ứng dụng dựa trên quyết định của decider

**Phần 4: Dashboard**: bộ RESTful API giúp người dùng cuối sử dụng hệ thống. Họ có thể sử dụng RESTful API thông qua giao diện web hoặc console tool

### 2.1.1 Monitoring

Hệ thống monitoring gồm các service collector chạy ở mỗi mesos slave và database dạng time series influxdb

Mỗi collector ở mỗi slave sẽ thu thập thông tin của các mesos task và gửi về influxdb. Các thông tin mà collector thu thập bao gồm:

* Thời gian ghi nhận
* ID của mesos task
* Docker container id tương ứng với mesos task đó
* App uuid tương ứng với app\_uuid lưu trong dabase mysql
* Phần trăm cpu sử dụng tại thời điểm đó (tính trong 1 giây)
* Phần trăm memory sử dụng tại thời điểm đó (tính trong 1 giây)

Các thông tin này sẽ được thu thập 3s một lần và để nâng cao hiệu năng dữ liệu sẽ được gửi đi theo bath (cứ 10s thì gửi dữ liệu thu được trong 10s đó tới influxdb 1 lần)

### 2.1.2 Decider

Decider trong hệ thống trên sử dụng kĩ thuật Static Threshold-based Rules. Luật do người sử dụng tạo ra được lưu trong database.

Các thông số mà người dùng cần cung cấp cho mỗi luật là :

* Loại metrics áp dụng (cpu hoặc memory)
* Ngưỡng trên
* Ngưỡng dưới
* Số lượng instances giảm đi
* Số lượng instances tăng lên
* Thời gian nghỉ sau khi scale lên
* Thời gian nghỉ sau khi scale xuống

**Ví dụ:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| metric type | upper threshold | lower threshold | instances out | instances in | scale up wait | scale down wait |
| cpu | 70 | 30 | 1 | 2 | 60 | 80 |

Bảng 1: Các thông số của một luật

Có nghĩa là: Nếu cpu sử dụng trung bình trong 1 phút (tham số này cố định) lớn hơn 70% thì tăng thêm 2 instances sau đó decider sẽ nghỉ 60s (không thực hiện ra quyết định) hoặc nếu cpu nhỏ hơn 30% thì sẽ thực hiện giảm 1 instances sau đó decider nghỉ 60s.

Trong mỗi lần ra quyết định, tất cả các luật do người dùng định nghĩa cho ứng dụng đó sẽ được áp dụng, sau đó nếu có nhiều luật thỏa mãn sẽ ưu tiên chọn luật có scale lên và trong các luật scale lên thỏa mãn sẽ chọn luật có số lượng instances in lớn nhất. Còn nếu không có scale lên mà chỉ có scale xuống thì sẽ chọn luật có số lượng instances out nhỏ nhất.

### 2.1.3 Scaler

Việc scale ứng dụng sẽ thông qua marathon. Marathon cung cấp một bộ REST API, thông qua đó chúng ta sẽ thực hiện scale một cách dễ dàng.

Mỗi ứng dụng sẽ có một khoảng instances hạn chế do người dùng chọn lúc tạo ứng dụng. Mỗi lần scale đều phải đảm bảo rằng số lượng instances sẽ không nằm ngoài khoảng này.

### 2.1.4 Dashboard

Dashboard cung cấp cho người dùng cuối một interface cho phép dễ dàng thao tác với hệ thống. Đặc biệt với web interface người dùng có thể visualization các nguồn tài nguyên đã sử dụng (cpu và memory) và số lượng instances theo thời gian qua đó có thể điều chỉnh các luật scale cho ứng dụng một cách hợp lí và có hiệu quả nhất.

## 2.2 Kiến trúc của một ứng dụng web trong hệ thống



Hình 7: Kiến trúc ứng dụng có khả năng scale

Một ứng dụng có thể scale được sẽ có những đặc điểm riêng so với những ứng dụng thiết kế theo cách truyền thống. Cụ thể như sau:

1. **Phải được đóng gói thành container**

Đóng gói vào container sẽ giúp ứng dụng dễ dàng mở rộng số lượng instances một cách nhanh chóng

Docker là một nền tảng mới, do đó còn lạ lẫm đối với nhiều người vì thế việc hỗ trợ đóng gói ứng dụng vào container là điều cần thiết để giúp người dùng xây dựng được một ứng dụng có thể scale một cách dễ dàng.

Ở đây ta sẽ tạo một Dockerfile để build một docker images (tên là apache-php) có chứa apache và php được cấu hình sẵn, sau đó khi chạy container ta sẽ mount thư mục chứa mã nguồn của ứng dụng vào thư mục web trong container (/var/www/html).

Như vậy ở đây người dùng chỉ việc upload thư mục mã nguồn của họ lên hoặc cung cấp link github chứa mã nguồn và hệ thống sẽ tự động down load mã nguồn về sau đó chạy container của images apache-php sau đó mount mã nguồn vừa down vào thư mục /var/www/html.

1. **Có cơ chế đồng bộ các file dữ liệu giữa các container**

Vì ở trong môi trường cluster nên không thể mount thư mục ở ngoài máy vật lí vào trong container theo cách thông thường được. Để giải quyết vấn đề này ta sẽ sử dụng một docker volume plugin gọi là docker-volume-netshare. Plugin này sẽ giúp các container ở các máy khác mount được thư mục ở một máy chủ chứ dữ liệu.

1. **Có service loadbalancing**

Có nhiệm vụ phân bố đồng đều lưu lượng truy cập giữa các container của một ứng dụng. Trong trường hợp này chúng ta sẽ sử dụng HAproxy. Do các ứng dụng biến đổi liên tục nên ta không thể cấu hình cho HAproxy bằng tay được nên ta sẽ sử dụng marathon-lb. marathon-lb là một tool để quản lí Haproxy dựa trên trạng thái của các ứng dụng trên marathon.

# CHƯƠNG 3: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

## 3.1 Phân tích chức năng

Mục đích của phần này là cung cấp sự mô tả đầy đủ về tất cả các chức năng chính cũng như giới hạn của hệ thống.

### 3.1.1 Các tác nhân tham gia hệ thống

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên tác nhân | Kế thừa | Mô tả |
| 1 | Khách |  | Là người không có tài khoản đăng nhập |
| 1 | Người dùng |  | Là người có tài khoản đăng nhập vào hệ thống, có khả năng tạo và quản lí các ứng dụng của họ |
| 2 | Người quản trị | Người dùng | Là người có tài khoản đăng nhập và có toàn quyền trong hệ thống |

Bảng 2: Các tác nhân của hệ thống

### 3.1.2 Danh sách các use case

|  |  |  |  |
| --- | --- | --- | --- |
| Use case ID | Tên use case | Mô tả | Tên tác nhân tham gia |
| UC01 | Tổng quan | Use case thổng quan hệ thống | Người dùng, Người quản trị, Khách |
| UC02 | Đăng kí | Đăng kí tài khoản người dùng | Khách |
| UC03 | Đăng nhập | Đăng nhập vào hệ thống | Người dùng |
| UC04 | Quản lí database | Quản lí instances database | Người dùng |
| UC04.1 | Thêm database | Thêm một instance database(MySQL) mới | Người dùng |
| UC04.2 | Xem thông tin database | Xem thông tin về địa chỉ (host và port), mật khẩu tài khoản root của database | Người dùng |
| UC04.3 | Thay đổi mật khẩu database | Thây đổi mật khẩu của tài khoản root của database | Người dùng |
| UC04.4 | Thao tác database | Thực hiện các truy vấn , chèn,… với database | Người dùng |
| UC04.5 | Xóa database | Xóa database | Người dùng |
| UC05 | Quản lí ứng dụng web | Thêm, cập nhật, xóa, … ứng dụng web | Người dùng |
| UC05.1 | Thêm ứng dụng web | Thêm một ứng dụng web mới | Người dùng |
| UC05.2 | Xem thông tin ứng dụng | Xem các thông tin về địa chỉ, các biến môi trường, bộ luật | Người dùng |
| UC05.3 | Cập nhật thông tin cho ứng dụng web | Thay đổi thông tin của ứng dụng (như các biến môi trường, bộ luật…) | Người dùng |
| UC05.4 | Thay đổi trạng thái ứng dụng | Chạy / Dừng ứng dụng | Người dùng |
| UC05.4.1 | Redeploy ứng dụng | Redeploy ứng dụng web | Người dùng |
| UC05.4.2 | Chạy/Dừng ứng dụng | Chạy (khởi động) hoặc dừng ứng dụng web | Người dùng |
| UC05.4.2 | Bặt tắt autoscaling | Bật hoặc tắt chức năng autoscaling cho ứng dụng | Người dùng |
| UC05.5 | Xóa ứng dụng | Xóa một ứng dụng web | Người dùng |
| UC05.6 | Quản lí bộ luật | Thêm, chỉnh sửa, xóa luật cho ứng dụng | Người dùng |
| UC05.6.1 | Thêm luật | Thêm luật mới cho ứng dụng web | Người dùng |
| UC05.6.2 | Cập nhật luật | Thay đổi các tham số cho luật | Người dùng |
| UC05.6.3 | Xóa luật | Xóa bỏ luật không cần thiết | Người dùng |
| UC05.7 | Xem biểu đồ | Xem biểu đồ về nguồn tài nguyên sử dụng của ứng dụng | Người dùng |
| UC06 | Quản lí người dùng | Thêm xóa tài khoản người dùng | Người quản trị |
| UC06.1 | Thêm người dùng | Thêm tài khoản người dùng mới | Người quản trị |
| UC06.2 | Xóa người dùng | Xóa tài khoản người dùng | Người quản trị |
| UC06.3 | Cập nhật thông tin người dùng | Cập nhật thông tin tài khoản người dùng | Người quản trị |

Bảng 3: Danh sách các usecase

### 3.1.3 Mô tả các ca sử dụng

#### UC01: Tổng quan



Hình 8: UC01 Tổng quan hệ thống

Hệ thống có 3 tác nhân tham gia. Người dùng có các chức năng quản lí ứng dụng và quản lí database. Người quản trị có mọi chức năng mà người dùng có ngoài ra còn có thêm chức năng quản lí tài khoản người dùng. Hai người dùng này cần phải đăng nhập để thực hiện các chức năng của mình. Khách là người không có tài khoản trong hệ thống chỉ có chức năng đăng kí để trở thành người dùng trong hệ thống.

#### UC02: Đăng kí



Hình 9: UC02 Đăng kí

Khách có thể đăng kí tài khoản người dùng để sử dụng các chức năng trong hệ thống

Các bước thực hiện ca sử dụng:

* Truy cập vào trang đăng kí
* Cung cấp các thông tin cần thiết cho tài khoản
* Nhấn nút đăng kí để đăng kí tài khoản

#### UC03: Đăng nhập



Hình 10: UC03 Đăng nhập

Người dùng và người quản trị muốn sử dụng các chức năng của hệ thống đều phải qua bước đăng nhập

Các bước thực hiện ca sử dụng:

* Truy cập vào giao diện web của hệ thống
* Nhập mật khẩu và tài khoản
* Nhấn nút Đăng nhập để gửi thông tin xác thực về hệ thống

#### UC04: Quản lí database



Hình 11: UC04 Quản lí database

Người dùng sau khi đăng nhập có thể sử dụng các chức năng này để quản lí các instances database phục vụ cho ứng dụng web của mình.

Các bước thực hiện:

* Chọn danh mục All database trên giao diện web
* Chọn các chức năng muốn thực hiện trên giao diện mà hệ thống cung cấp

#### UC04.1: Thêm database



Hình 12: UC04.1: Thêm database

Người dùng có thể thêm instance database mới với một khẩu root do mình định nghĩa.

Các bước thực hiện:

* Trên giao diện quản lí database, chọn New Database
* Hệ thống hiện thị popup chứa form nhập liệu mật khẩu root
* Người sử dụng nhập mật khẩu tài khoản root và ấn nút Submit
* Hệ thống trả về kết quả

#### UC04.2: Xem thông tin database



Hình 13: UC04.2: Xem thông tin database

Người dùng có thể xem thông tin của các instances database do mình tạo như: Địa chỉ instance (bao gồm hostname và port), mật khẩu tài khoản root

Các bước thực hiện:

* Vào giao diện quản lí database
* Hệ thống sẽ cũng cấp mọi thông tin về các instances database

#### UC04.3: Đổi mật khẩu root



Hình 14: UC04.3: Đổi mật khẩu root

Người dùng có thể cập nhật lại mật khẩu root của instance database của mình

Các bước thực hiện:

* Tại giao diện quản lí database, ấn nút đổi mật khẩu của database tương ứng (mỗi database có một nút riêng)
* Hệ thống hiện thị popup chứa form nhập liệu mật khẩu root mới
* Người dùng nhập mật khẩu mới và nhấn submit
* Hệ thống trả về kết quả

#### UC04.4: Thao tác với database



Hình 15: UC04.4: Thao tác với database

Người dùng có thể thao tác với instances database như tạo database mới, import, export dữ liệu, sử dụng các câu lệnh sql

Các bước thực hiện:

* Trên giao diện quản lí database nhấn nút PHPMyadmin
* Hệ thống cung cấp giao diện đăng nhập phpmyadmin
* Người dùng nhập thông tin tài khoản đăng nhập vào database
* Sau khi đăng nhập thành công, người dùng thao tác với database trên giao diện do hệ thống trả về

#### UC04.5: Xóa database



Hình 16: UC04.5: Xóa database

Người dùng có thể xóa instance database khi không cần thiết

Các bước thực hiện:

* Tại giao diện quản lí database nhấn nút Delete tương ứng với instance database muốn xóa
* Hệ thống thực hiện xóa và trả về kết quả

#### UC05: Quản lí ứng dụn web



Hình 17: UC05: Quản lí ứng dụn web

Sau khi đăng nhập người dùng có thể tạo ứng dụng web của họ và có thể thực hiện các thao tác cập nhật, xóa bỏ ứng dụng…

Các bước thực hiện:

* Chọn danh mục All Application trên giao diện web
* Thực hiện các chức năng tương ứng trên giao diện này

#### UC05.1: Thêm ứng dụng



Hình 18: UC05.1: Thêm ứng dụng

Người dùng có thể dễ dàng tạo một ứng dụng web có khả năng scale bằng cách cung cấp địa chỉ github của mã nguồn

Các bước thực hiện:

* Tại trang quản lí ứng dụng nhấn “New Application”
* Hệ thống cung cấp form dữ liệu
* Người dùng cung cấp các thông tin cần thiết cho ứng dụng và ấn submit
* Hệ thống thực hiện tạo ứng dụng và thông báo kết quả cho người dùng

#### UC05.2: Xem thông tin ứng dụng



Hình 19: UC05.2: Xem thông tin ứng dụng

Người dùng có thể xem các thông tin chi tiết về ứng dụng do mình tạo ra bao gồm: tên ứng dụng, địa chỉ github, địa chỉ truy cập ứng dụng, các biến môi trường sử dụng trong ứng dụng, các thông tin về nguồn tài nguyên của ứng dụng (cpu, memory, số lượng instances, min max instances), bộ luật của ứng dụng dùng để scale

Các bước thực hiện:

* Vào trang quản lí ứng dụng
* Hệ thống cung cấp một số thông tin chủ yếu của ứng dụng
* Người dùng nhấn vào nút “Info”
* Hệ thống hiện thị popup có các thông tin chi tiết của ứng dụng

#### UC05.3: Cập nhật ứng dụng



Hình 20: UC05.3: Cập nhật ứng dụng

Người dùng có thể cập nhật các thông tin cơ bản cho ứng dụng của mình

Các bước thực hiện:

* Tại giao diện quản lí ứng dụng nhấn nút Edit tương ứng với ứng dụng muốn cập nhật
* Hệ thống trả về form dữ liệu
* Người dùng điền thông tin muốn cập nhật cho ứng dụng và ấn submit
* Hệ thống thực hiện cập nhật và trả về kết quả

#### UC05.4: Thay đổi trạng thái ứng dụng



Hình 21: UC05.4: Thay đổi trạng thái ứng dụng

Người dùng có thể thay đổi trạng thái của ứng dụng: dừng, chạy, scale (bằng tay) ứng dụng, bật tắt autoscaling.

Các bước thực hiện:

* Vào giao diện quản lí ứng dụng
* Nhấn nút tương ứng với mỗi trạng thái cần thay đổi

#### UC05.4.1: Redeploy ứng dụng



Hình 22: UC05.4.1: Redeploy ứng dụng

Người dùng có thể deploy lại ứng dụng của mình (sau khi có cập nhật thông tin ứng dụng) để những thông tin cập có hiệu lực

Các bước thực hiện:

* Tại giao diện quản lí ứng dụng nhấn nút redeploy tương ứng với ứng dụng muốn redeploy
* Hệ thống thực hiện redeploy lại ứng dụng và thông báo kết quả

#### UC05.4.2: Scale ứng dụng



Hình 23: UC05.4.2: Scale ứng dụng

Người dùng có thể scale ứng dụng bằng tay

Các bước thực hiện:

* Truy cập trang quản lí ứng dụng
* Nhấn nút scale tương ứng với ứng dụng muốn scale
* Nhập số lượng instances của ứng dụng
* Nhấn nút submit để scale ứng dụng

#### UC05.4.3: Chạy / Dừng ứng dụng



Hình 24: UC05.4.3 Chạy / Dừng ứng dụng

Người dùng có thể khởi chạy hoặc dừng ứng dụng

Các bước thực hiện:

* Truy cập giao diện quản lí ứng dụng
* Nhấn nút Start/Stop tương ứng với với ứng dụng muốn thay đổi trạng thái
* Hệ thống thực hiện thay đổi trạng thái và trả về kết quả

#### UC05.4.4: Bật/Tắt Autoscaling



Hình 25: UC05.4.4 Bật/Tắt Autoscaling

Người dùng có thể bật autoscaling hoặc tắt nó nếu cảm thấy không cần thiết hoặc không hiệu quả

Các bước thực hiện:

* Truy cập giao diện quản lí ứng dụng
* Nhấn nút On/Off Autoscaling tương ứng với với ứng dụng muốn thay đổi trạng thái
* Hệ thống thực hiện thay đổi trạng thái và trả về kết quả

#### UC05.5: Xóa ứng dụng



Hình 26: UC05.5: Xóa ứng dụng

Người dùng có thể xóa bỏ ứng dụng của họ khi không dùng nữa

Các bước thực hiện:

* Truy cập giao diện quản lí ứng dụng
* Nhấn Delete tương ứng với ứng dụng muốn xóa bỏ
* Hệ thống thực hiện xóa ứng dụng và trả về kết quả

#### UC05.6: Quản lí bộ luật



Hình 27: UC05.6: Quản lí bộ luật

Người dùng có thể quản lí bộ luật dùng để scale cho mỗi ứng dụng của họ

Các bước thực hiện:

* Truy cập trang quản lí ứng dụng
* Nhấn nút edit tương ứng với ứng dụng muốn quản lí bộ luật
* Các thao tác quản lí bộ luật được thực hiện ở tab Rule trên giao diện hệ thống trả về

#### UC05.6.1: Thêm luật



Hình 28: UC05.6.1: Thêm luật

Người dùng có thể thêm mới luật cho ứng dụng của họ

Các bước thực hiện:

* Truy cập trang quản lí bộ luật
* Nhấn nút “Add Rule”
* Hệ thống trả về form nhập liệu
* Người dùng nhập các thông số của luật và nhấn submit
* Hệ thống thêm luật vào database và trả về kết quả

#### UC05.6.2 Cập nhật luật



Hình 29: UC05.6.2 Cập nhật luật

Người dùng có thể cập nhật luật do họ tạo ra

Các bước thực hiện:

* Truy cập trang quản lí bộ luật
* Nhấn nút edit tương ứng với luật muốn sửa đổi
* Hệ thống cung cấp form nhập liệu
* Người dùng cung cấp tham số mới cho luật và nhấn submit
* Hệ thống thực hiện cập nhật và trả về kết quả

#### UC05.6.3: Xóa luật



Hình 30: UC05.6.3: Xóa luật

Người dùng có thể xóa bộ luật khi không dùng nữa

Các bước thực hiện:

* Truy cập trang quản lí bộ luật
* Nhấn nút Delete tương ứng với luật muốn xóa
* Hệ thống thực hiện xóa luật và trả kết quả cho người dùng

#### UC05.7: Xem biểu đồ



Hình 31: UC05.7: Xem biểu đồ

Người dùng có thể xem biểu đồ (biểu đồ đường) thể hiện mức sử dụng tài nguyên của ứng dụng (cpu, memory, số lượng instances) trong một khoảng thời gian lựa chọn

Các bước thực hiện

* Truy cập trang quản lí ứng dụng
* Nhấn nút Chart tương ứng với ứng dụng muốn xem biểu đồ
* Hệ thống trả về biểu đồ cho người dùng với khoảng thời gian mặc định
* Người dùng có thể chọn khoảng thời gian cho biểu đồ bằng cách nhập khoảng thời gian vào ô input form

#### UC06: Quản lí tài khoản người dùng



Hình 32: UC06: Quản lí tài khoản người dùng

Người quản trị có quyền quản lí các tài khoản người dùng

Các bước thực hiện:

* Truy cập vào trang quản lí người dùng
* Thực hiện các thao tác quản lí trên trang này

#### UC06.1: Thêm người dùng



Hình 33: UC06.1: Thêm người dùng

Người quản trị có thể thêm tài khoản người dùng mới

Các bước thực hiện:

* Truy cập trang quản lí người dùng
* Chọn chức năng thêm người dùng
* Hệ thống trả về form dữ liệu
* Người quản trị nhập thông tin tài khoản mới và ấn submit
* Hệ thống tạo user mới và trả về kết quả

#### UC06.2: Xóa người dùng



Hình 34: UC06.2: Xóa người dùng

Người quản trị có thể xóa tài khoản người dùng

Các bước thực hiện

* Truy cập trang quản lí người dùng
* Nhấn nút Delete tương ứng với người dùng muốn xóa
* Hệ thống thực hiện xóa và trả về kết quả

#### UC06.3: Cập nhật thông tin



Hình 35: UC06.3: Cập nhật thông tin

Người quản trị có thể cập nhật thông tin của người dùng

Các bước thực hiện:

* Truy cập trang quản lí người dùng
* Nhấn nút Edit tương ứng với người dùng muốn cập nhật
* Hệ thống trả về form nhập liệu
* Người quản trị nhập thông mới cho tài khoản và nhấn nút submit
* Hệ thống thực hiện cập nhật và trả về kết quả

## 3.2 Phân tích cấu trúc

Phần này sẽ cung cấp các lớp chính tạo nên hệ thống

Có 3 loại lớp sẽ tham gia trong mô hình hệ thống: Boundaries, Controls, Entities

### 3.2.1 Lớp Boundaries (các lớp biên hay các lớp đối thoại):

* B\_Login: Giao diện login vào hệ thống
* B\_ManageDatabase: Giao diện quản lí database
* B\_InfoDatabase: Giao diện hiển thị thông tin database
* B\_CreateDatabase: Giao diện tạo database
* B\_ChangePassDatabase: Giao diện thay đổi mật khẩu root cho database
* B\_PHPMyadmin: Giao diện thao tác với database
* B\_DeleteDatabase: Giao diện xóa database
* B\_CreateApp: Giao diện tạo ứng dụng web
* B\_InfoApp: Giao diện hiển thị thông tin ứng dụng web
* B\_UpdateApp: Giao diện cập nhật thông tin ứng dụng web
* B\_ManageApp: Giao diện quản lí ứng dụng web (start/stop/redeploy …)
* B\_AddRule: Giao diện thêm luật cho ứng dụng
* B\_UpdateRule: Giao diện cập nhật luật
* B\_DeleteRule: Giao diện để xóa luật
* B\_ViewChart: Giao diện xem biểu đồ metrics của ứng dụng
* B\_AddUser: Giao diện thêm user mới
* B\_DeleteUser: Giao diện xóa user
* B\_UpdateUser: Giao diện cập nhật user

### 3.2.2 Lớp Controls (các lớp điều khiển ):

* C\_Database



* C\_App



* C\_Policy



* C\_User



- C\_Chart()



### 3.2.3Lớp Entities (các lớp thực thể hay các lớp lĩnh vực):

* E\_Database



* E\_Application



* E\_Policy



* E\_User



* E\_Metric



## 3.3 Phân tích hành vi người dùng

Phần này sẽ đi vào diễn tả sự tương tác giữa các đối tượng nhằm tạo ra các kịch bản của mỗi ca sử dụng của hệ thống dựa trên các biểu đồ tương tác

### 3.3.1 Đăng nhập



Hình 36: Sequence đăng nhập

Mô tả: Sau khi truy cập vào trang login, người dùng điền các thông tin cần thiết để xác thực và ấn nút submit, thông tin sẽ được gửi tới C\_User để thực hiện xác thực và trả về kết quả. Nếu đăng nhập thành công sẽ được chuyển hướng tới trang B\_Index, ngược lại thông báo lỗi cho người dùng.

### 3.3.2 Thêm ứng dụng web mới



Hình 37: Sequence thêm ứng dụng web

Mô tả:

Sau khi truy cập vào trang B\_CreateApp, người dùng điền các thông tin cần thiết để tạo ứng dụng mới. Thông tin sẽ được gửi tới C\_App. Trước khi tạo ứng dụng mới sẽ kiểm tra tên ứng dụng (ứng với người dùng đó) đã tồn tại hay chưa, nếu đã tồn tại sẽ gửi lỗi về, ngược lại sẽ tạo ứng dụng mới và trả kết quả về cho người dùng.

### 3.3.3 Cập nhật ứng dụng web



Hình 38: Sequence Cập nhập ứng dụng web

Mô tả:

Khi người dùng truy cập vào trang quản lí ứng dụng (B\_ManageApp) thống sẽ truy vấn vào database trả về tất cả ứng dụng của người dùng. Sau đó người dùng chọn ứng dụng muốn sửa đổi sẽ được chuyển sang trang chỉnh sửa ứng dụng (B\_EditApp), người dùng điền các thông tin cập nhật vào form và nhấn submit. Dữ liệu sẽ được gửi tới C\_App để xử lí và cập nhật vào database (updateApp()). Sau đó trả kết quả cập nhật về cho người dùng.

### 3.3.4 Thay đổi trạng thái ứng dụng web



Hình 39: Sequence thay đổi trạng thái ứng dụng

Mô tả:

Khi người dùng truy cập trang quản lí ứng dụng web (ManageApp) hệ thống sẽ trả về danh sách các ứng dụng của người dùng. Sau đó người dùng nhấn nút thay đổi trạng thái tương ứng (redeploy, start, stop, autoscaling). Thông tin sẽ được gửi tới C\_App. C\_App gửi yêu cầu thay đổi trạng thái tới E\_App và trả về kết quả cho người dùng.

### 3.3.5 Thêm luật cho ứng dụng



Hình 40: Sequence thêm luật cho ứng dụng

Mô tả:

Sau khi người dùng truy cập vào trang quản lí ứng dụng (ManageApp) hệ thống hiện thị danh sách các ứng dụng của người dùng. Người dùng chọn chỉnh sửa ứng dụn muốn thêm luật sẽ được chuyển tới trang chỉnh sửa ứng dụng (B\_EditApp). Tại đây người dùng click vào nút newPolicy sẽ được cung cấp một form để nhập các tham số của luật. Người dùng nhập tham số vào và ấn submit, dữ liệu được gửi tới C\_Policy. C\_Policy sẽ thực hiện thêm luật vào database (E\_Policy) và trả về kết quả, hiển thị cho người dùng.

### 3.3.6 Xem biểu đồ nguồn tài nguyên



Hình 41: Sequence xem biểu đồ nguồn tài nguyên

Mô tả:

Khi người dùng truy cập trang quản lí ứng dụng web (ManageApp) hệ thống sẽ trả về danh sách các ứng dụng của người dùng. Sau đó người dùng nhấn nút Chart tương ứng với ứng dụng muốn xem sẽ được chuyển qua trang xem biểu đồ (B\_ViewChart). Tiếp đó người dùng chọn khoản thời gian muốn xem, thông tin về ứng dụng và khoảng thời gian sẽ được gửi đến C\_Chart, tại đây C\_Chart sẽ truy vấn đến E\_Metric và lấy dữ liệu trả về cho B\_ViewChart. Cuối cùng từ dữ liệu được trả về B\_ViewChart sẽ vẽ biểu đồ đường cho mỗi loại nguồn tài nguyên.

### 3.3.6 Thêm ứng dụng database



Hình 42: Sequence thêm ứng dụng database

Mô tả:

Sau khi truy cập vào trang B\_CreateDatabase, người dùng điền các thông tin cần thiết để tạo ứng dụng database mới. Thông tin sẽ được gửi tới C\_Database. C\_Database sẽ yêu gửi yêu cầu tạo ứng dụng database mới tới E\_Database và trả kết quả từ E\_Database về cho người dùng.

### 3.3.7 Thao tác với ứng dụng database

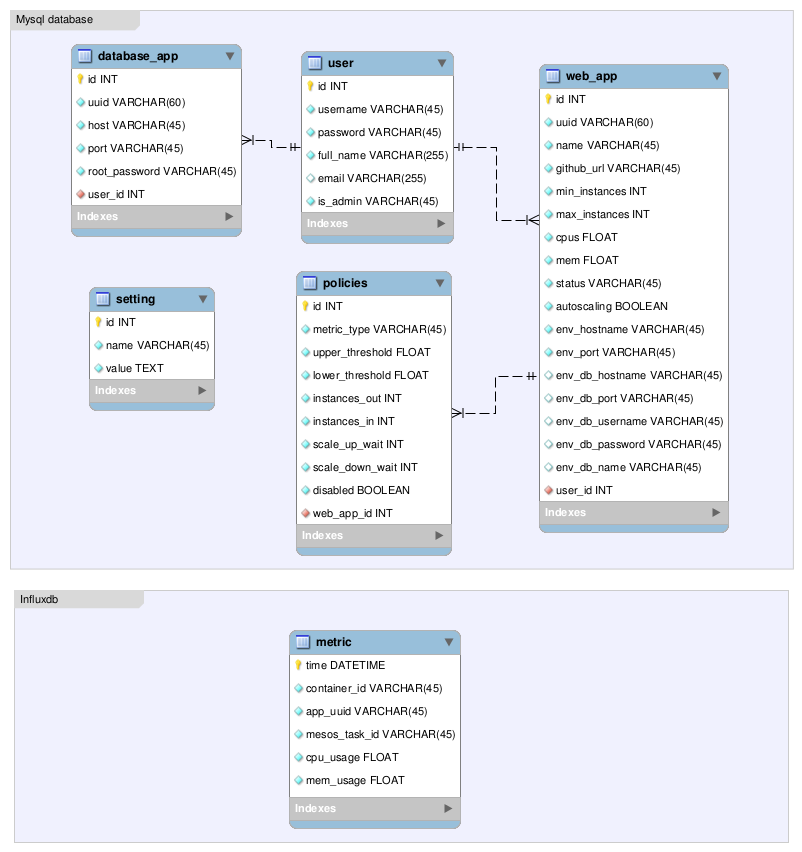


Hình 43: Sequence thao tác với database

Mô tả:

Người dùng truy cập vào trang quản lí database, hệ thống sẽ gửi thông tin các database của người dùng về và hiển thị cho người dùng. Sau đó người dùng nhấn nút PHPMyadmin để truy cập vào trang PHPMyadmin để thao tác với database.

## 3.4 Thiết kế cơ sở dữ liệu



Hình 44: Cơ sở dữ liệu tổng quát

**Danh sách các bảng**

|  |  |
| --- | --- |
| Tên bảng | Mô tả |
| user | Chứa các thông tin của người dùng và người quản trị |
| wep\_app | Chứa các thông tin của ứng dụng web |
| database\_app | Chứa các thông tin của ứng dụng database |
| policies | Chứa các thông số của luật cho ứng dụng web |
| setting | Chứa các thông tin cấu hình cho hệ thống |
| metric | Chứa dữ liệu về các nguồn tài nguyên sử dụng của ứng dụng web theo thời gian |

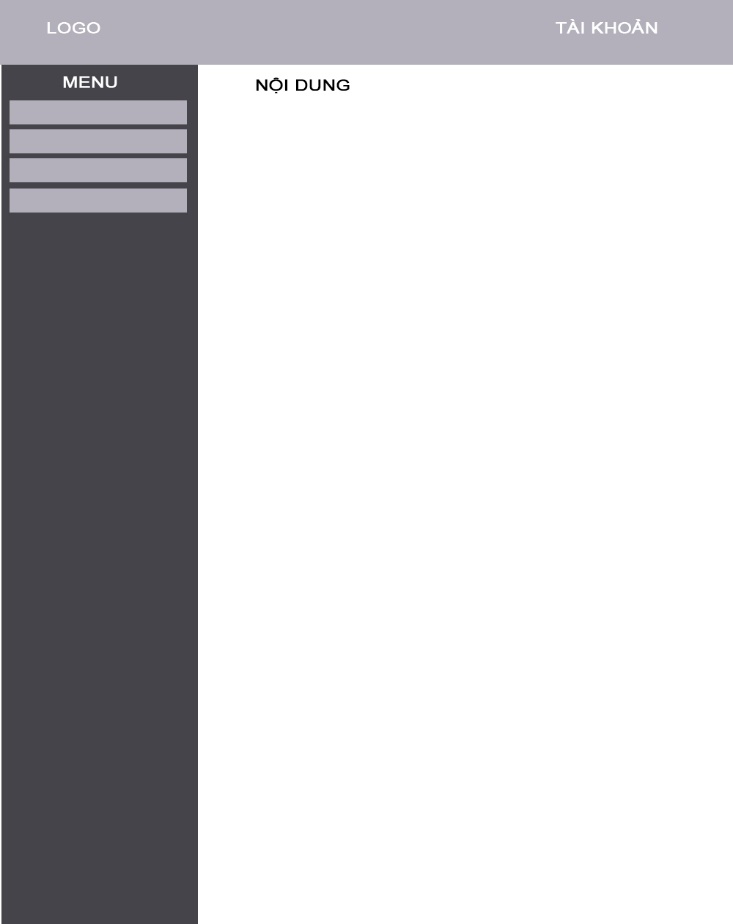
Bảng 4: Danh sách các bảng cơ sở dữ liệu

## 3.5 Thiết kế giao diện người dùng

Giao diện người dùng thiết kế dựa trên tiêu chí đơn giản dễ sửa dụng, các nút chức năng được thể hiện rõ ràng, dễ nhìn thấy.

Giao diện được thiết kế dựa trên template **AdminLTE**

### 3.5.1 Bố cục giao diện



Hình 45: Bố cục giao diện

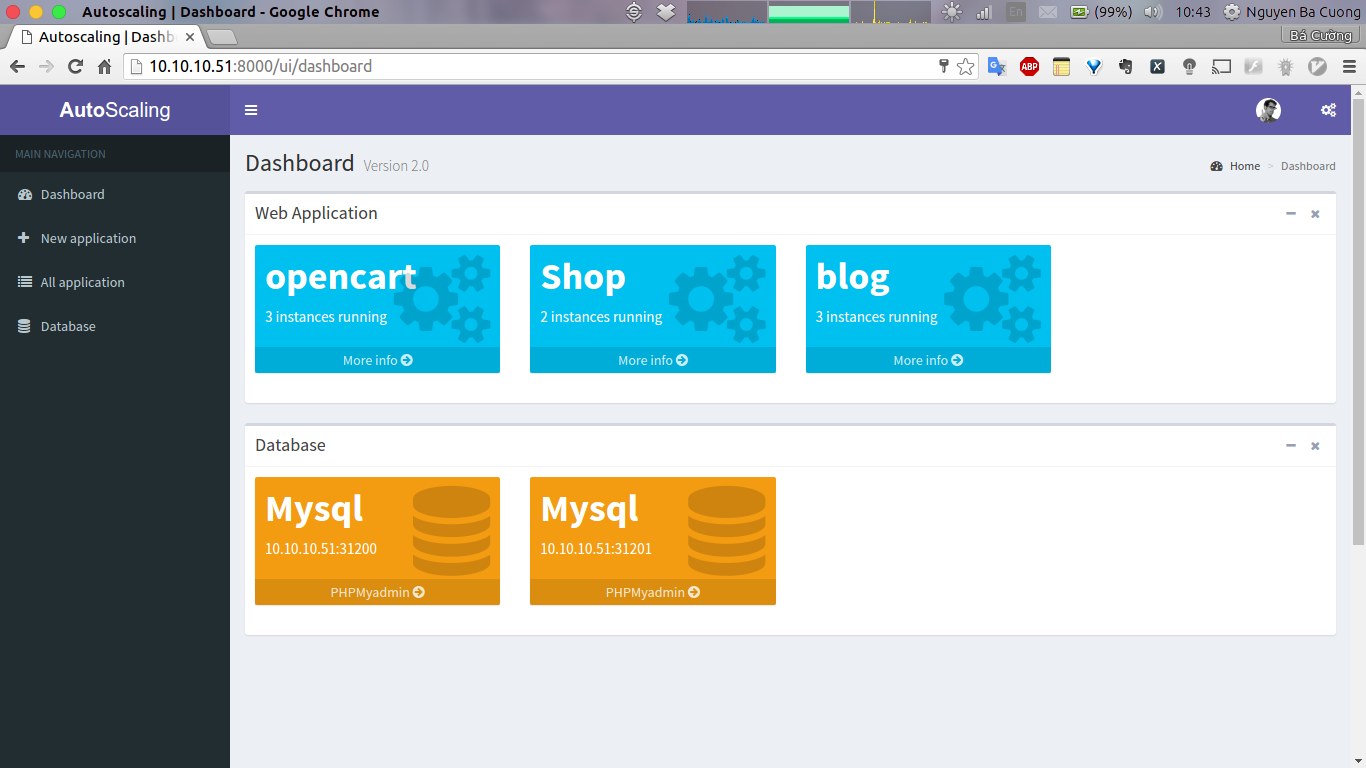
### 3.5.2 Danh sách các giao diện

|  |  |
| --- | --- |
| Tên giao diện | Mô tả giao diện |
| Đăng kí | Giao diện để đăng kí tài khoản người dùng |
| Đăng nhập | Giao diện đăng nhập vào hệ thống |
| Dashboard | Trang chủ hệ thống, hiện thị thông tin tổng quan về các ứng dụng web, ứng dụng database |
| Quản lí ứng dụng web | Hiển thị các ứng dụng web và các chức năng để quản lí ứng dụng web |
| Thêm ứng dụng web | Giao diện để thêm ứng dụng web mới |
| Chỉnh sửa ứng dụng web | Chỉnh sửa thông tin của ứng dụng web |
| Hiển thị thông tin ứng dụng web | Hiển thị các thông tin chi tiết về ứng dụng web |
| Quản lí ứng luật | Quản lí luật cho ứng dụng |
| Quản lí ứng dụng database | Hiển thị tất cả ứng dụng database của người dùng và các chức năng với ứng dụng web |
| Thao tác với database | Giao diện để thao tác với database (PHPMyadmin) |
| Xem biểu đồ nguồn tài nguyên | Biểu đồ nguồn tài nguyên của ứng dụng web |

Bảng 5: Danh sách giao diện

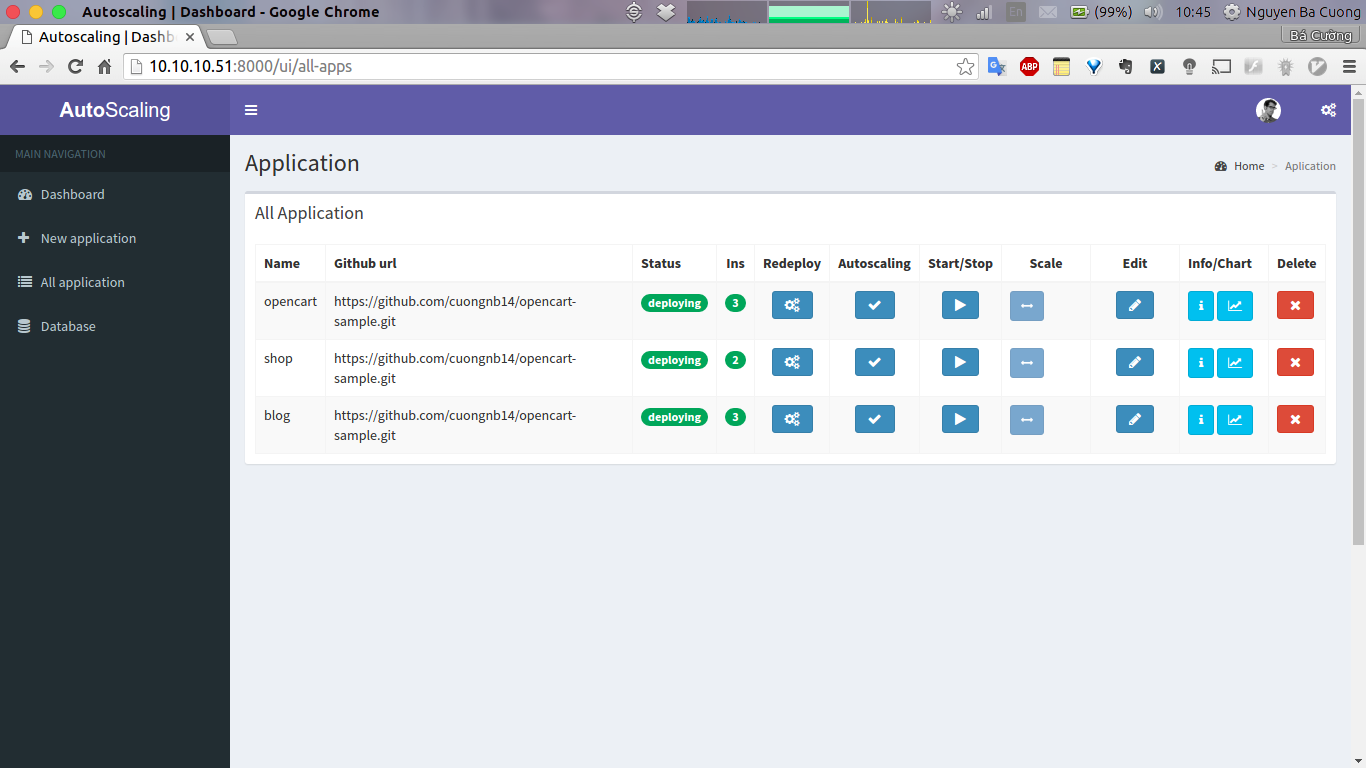
### 3.5.3 Một số hình ảnh giao diện

**Dashboard (Trang chủ)**



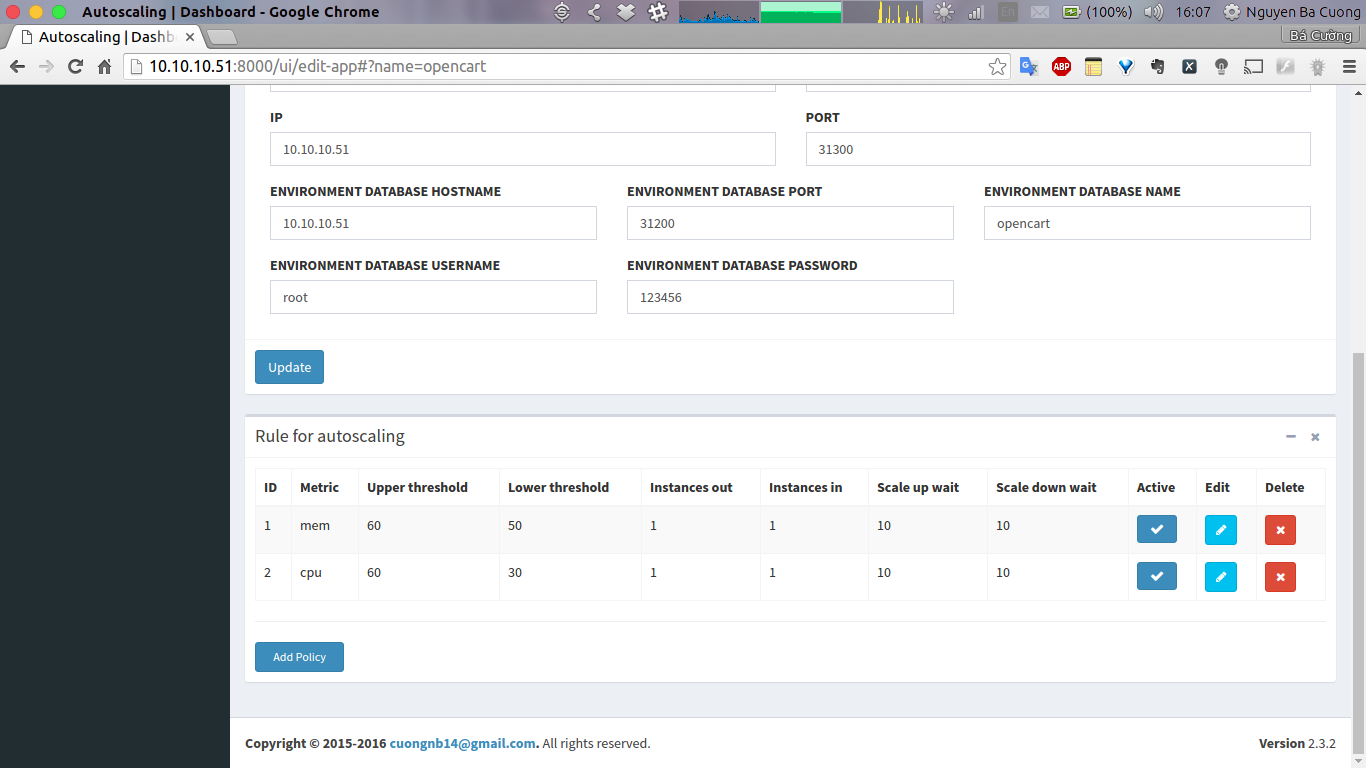
Hình 46: Giao diện trang chủ

**Quản lí ứng dụng web**



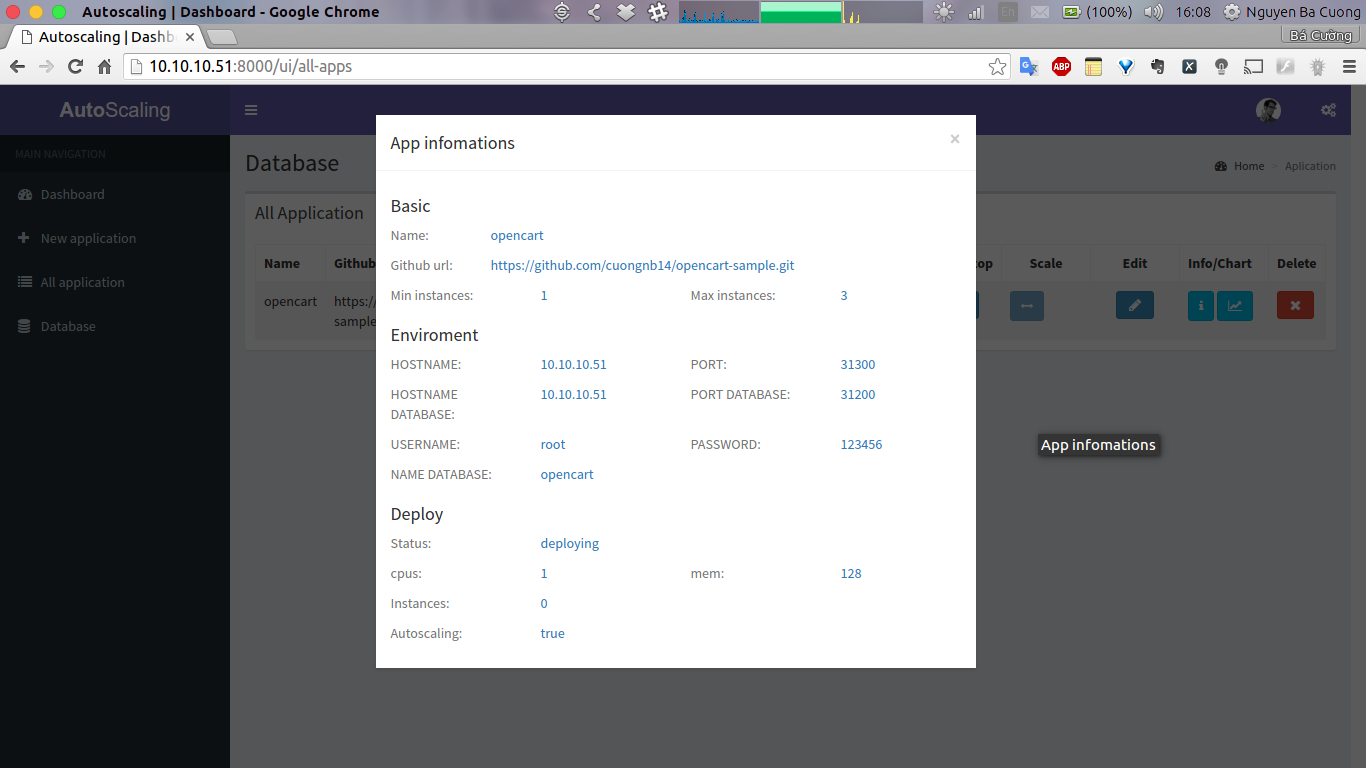
Hình 47: Giao diện quản lí ứng dụng web

**Quản lí luật**

****

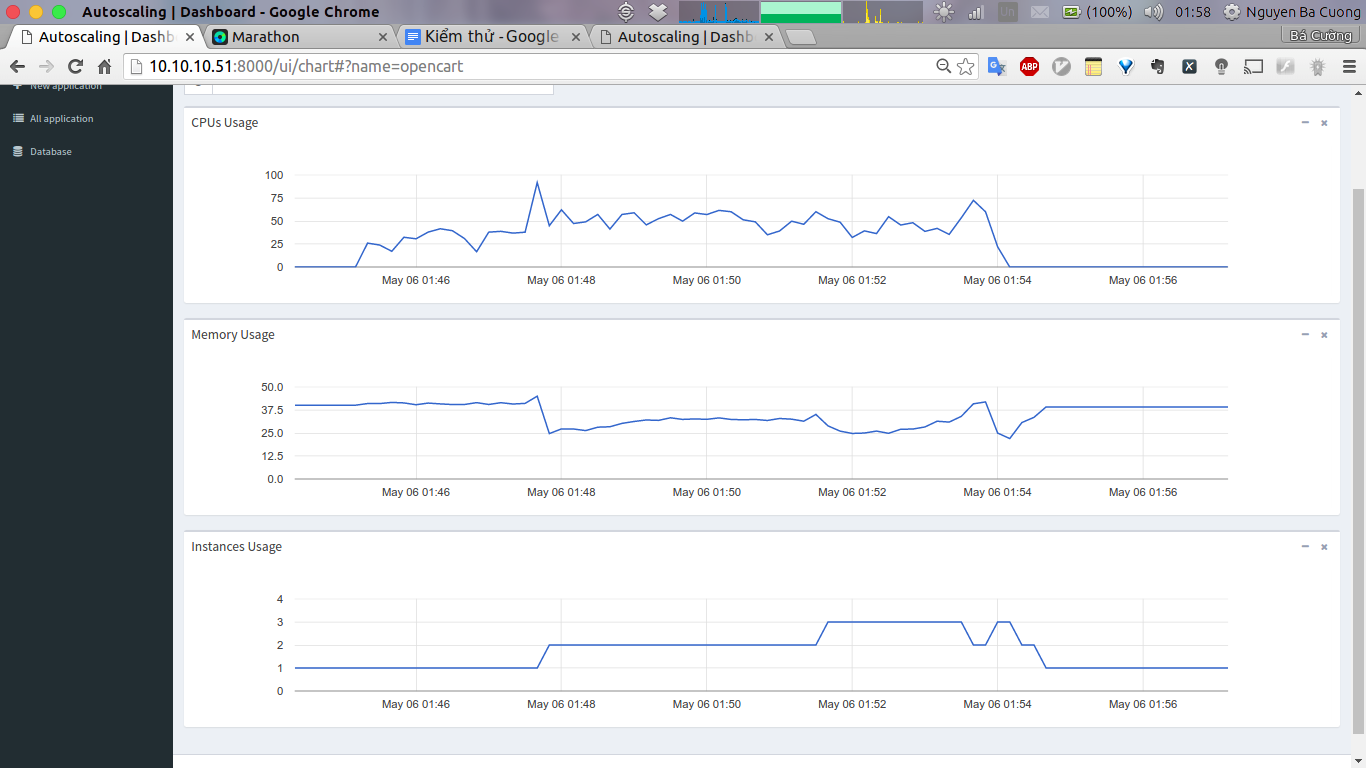
Hình 48: Giao diện quản lí luật

**Xem thông tin ứng dụng web**



Hình 49: Giao diện xem thông tin ứng dụng web

**Xem biểu đồ ứng dụng**



Hình 50: Giao diện xem biểu đồ ứng dụng

## 3.6 Thiết kế console tool

Console tool là một giao diện dòng lệnh giúp người dùng có thể thao tác nhanh chóng với hệ thống thông qua các dòng lệnh

Các trình tự xử lí một câu lệnh:



Hình 51: Trình tự thực hiện một dòng lệnh

Các quá trình trên được thực hiện một cách đồng bộ.

# CHƯƠNG 4: KIỂM THỬ HỆ THỐNG

## 4.1 Cài đặt môi trường kiểm thử

- Hệ thống kiểm thử gồm 3 node ( 3 virtual machine ) chạy hệ điều hành ubuntu server 14.04 với cấu hình:

* Node 1: 2Gb ram, 2 cpu, ổ cứng HDD 7200rpm
* Node 2: 1Gb ram, 2 cpu, ổ cứng HDD 7200rpm
* Node 3: 1Gb ram, 2 cpu, ổ cứng HDD 7200rpm
* Tải mã nguồn từ github <https://github.com/cuongnb14/autoscaling-paas.git> về node 1 và cho vào thư mục /autoscaling
* Sử dụng các script được cung cấp sẵn để cài đặt các phần mềm cần thiết lên các node.
* Các phềm mềm được cài đặt trên hệ thống là:
  + Node 1: mesos master, slave, database cho các dịch vụ autoscaling (mysql, influxdb) được chạy trên marathon, RESTful server của autoscaling, dịch vụ collector
  + Node 2, Node 3: mesos slave, dịch vụ collector

## 4.2 Tiến hành kiểm thử

Danh sách các chức năng được kiểm thử

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên nhóm chức năng | Mô tả | Mục đích |
| 1 | Tạo ứng dụng database | Tạo một instance database cho ứng dụng web | Kiểm tra chức năng có hoạt động tốt và ổn định hay không |
| 2 | Tạo ứng dụng web | Tạo một ứng dụng web từ mã nguồn trên github | Kiểm tra chức năng có hoạt động tốt và ổn định hay không |
| 3 | Chức năng autoscaling | Giả lập hành vi người dùng để kiểm tra chức năng autoscaling | Kiểm tra chức năng có hoạt động hay không và độ chính xác của chức năng. |

Bảng 6: Các chức năng kiểm thử

### 4.2.1 Tạo ứng dụng database

* Điều kiện tiên quyết: Tạo database thành công
* Kết quả mong đợi: Instance database được khởi tạo nhanh, các thao tác với database ổn định
* Các bước thực hiện:

1. Đăng nhập và truy cập trang quản lí database
2. Nhấn nút tạo database
3. Nhập mật khẩu cho database và nhấn submit
4. Ghi lại kết quả hệ thống trả về và vào trang quản trị của marathon để tính thời gian khởi tạo database (được tính từ lúc nhấn submit đến lúc instance database được khởi tạo thành công trên marathon).
5. Nhấn nút PHPMyadmin để truy cấp vào trang phpmyadmin
6. Thao tác một số chức năng ngẫu nhiên (import, select, …) và ghi lại kết quả
7. Làm lại các bước trên 5 lần

* Kết quả thực nghiệm:

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tạo database | Thời gian | Thao tác với database |
| 1 | Thành công | 3s | Ổn định |
| 2 | Thành công | 4s | Ổn định |
| 3 | Thành công | 4s | Ổn định |
| 4 | Thành công | 3s | Ổn định |
| 5 | Thành công | 3s | Ổn định |

Bảng 7: Kết quả thực nghiệm chức năng tạo Database

* Nhận xét: Chức năng tạo ứng dụng database hoạt động rất tốt, thời gian tạo một instance database rất nhanh

### 4.2.2 Tạo một ứng dụng web

* Điều kiện tiên quyết: Tạo ứng dụng thành công
* Kết quả mong đợi: Thời gian tạo ứng dụng nhanh
* Giới thiệu ứng dụng opencart:

Ứng dụng được tạo trong quá trình kiểm thử này được xây dựng từ mã nguồn ở <https://github.com/cuongnb14/opencart-sample.git> . Đây là một ứng dụng mua hàng online được xây dựng trên mã nguồn mở opencart. Kích thước mã nguồn là 9.26MB (kích thước càng lớn sẽ càng ảnh hưởng tới thời gian tải mã nguồn).

* Các bước thực hiện:

1. Đăng nhập và tạo một ứng dụng database sau đó import dữ liệu trong file opencart.sql (ở thư mục gốc của mã nguồn) vào database
2. Chọn chức năng tạo ứng dụng web
3. Cung cấp các thông tin cần thiết cho ứng dụng

* github\_url: https://github.com/cuongnb14/opencart-sample.git
* cpu: 1
* mem: 128
* Biến môi trường phù hợp với thông số của databse tạo ở bước 1
* Các trường còn lại để mặc định

1. Nhấn nút submit và ghi lại kết quả trả về, tính thời gian khởi tạo ứng dụng thành công (bao gồm thời gian clone mã nguồn dữ liệu trên git và thời gian triển khai lên marathon):

* Thời gian clone được tính từ lúc chạy câu lệnh git clone đến khi báo clone xong. Các thông tin này lấy từ log của dashboard
* Thời gian triển khai lên marathon được tính từ lúc clone xong đến khi instance được triển khai thành công trên marathon (Vào trang quản trị của marathon để xem trạng thái triển khai ứng dụng).

1. Truy cập vào trang web vừa tạo và thực hiện một số thao tác, ghi lại nhận xét
2. Làm lại các bước 2 đến 4 năm lần

* Kết quả kiểm thử

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | Tạo ứng dụng web | Thời gian clone | Thời gian triển khai ứng dụng | Sử dụng ứng dụng |
| 1 | Thành công | 16s | 9s | Ổn định |
| 2 | Thành công | 20s | 7s | Ổn định |
| 3 | Thành công | 18s | 6s | Ổn định |
| 4 | Thành công | 18s | 7s | Ổn định |
| 5 | Thành công | 17s | 7s | Ổn định |

Bảng 8: Kết quả kiểm nghiệm chức năng tạo ứng dụng

* Nhận xét: Chức năng tạo ứng dụng web hoạt động tốt. Thời gian tạo ứng dụng nhanh, chủ yếu phụ thuộc vào kích thước của mã nguồn và tốc độ mạng của máy chủ.

### 4.2.3 Chức năng autoscaling

* Điều kiện tiên quyết: Chức năng hoạt động được
* Kết quả mong đợi: Ứng dụng được scale hoạt động ổn định trong quá trình kiểm thử, số request lỗi thấp
* Giới thiệu về jmeter:

Jmeter là công cụ để đo độ tải và performance của đối tượng, có thể sử dụng để test performance trên cả nguồn tĩnh và nguồn động, có thể kiểm tra độ tải và hiệu năng trên nhiều loại server khác nhau như: Web – HTTP, HTTPS, SOAP, Database via JDBC, LDAP, JMS, Mail – SMTP(S), POP3(S) and IMAP(S)…

Trong lần kiểm thử này chúng ta sẽ sử dụng jmeter để giả lập các hành vi người dùng truy cập vào một ứng dụng web (opencart) trên hệ thống. Các hành vi này theo thứ tự là:

1. Truy cập vào trang chủ
2. Thực hiện tìm kiếm sản phẩm
3. Nhấn nút mua hàng
4. Truy cập vào giỏ hàng
5. Nhấn nút thanh toán

Kịch bản cho các hành vi trên được cung cấp tại <https://github.com/cuongnb14/autoscaling-paas.git>

* Các bước thực hiện:

1. Tạo một ứng dụng opencart (như trong lần kiểm thử tạo ứng dụng) và cung cấp bộ luật autoscaling cho ứng dụng như sau

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| metric type | upper threshold | lower threshold | Instances out | Instances in | Scale up wait | Scale down wait |
| cpu | 60 | 30 | 1 | 1 | 10 | 10 |
| mem | 60 | 40 | 1 | 1 | 10 | 10 |

Bảng 9: Bộ luật kiểm thử

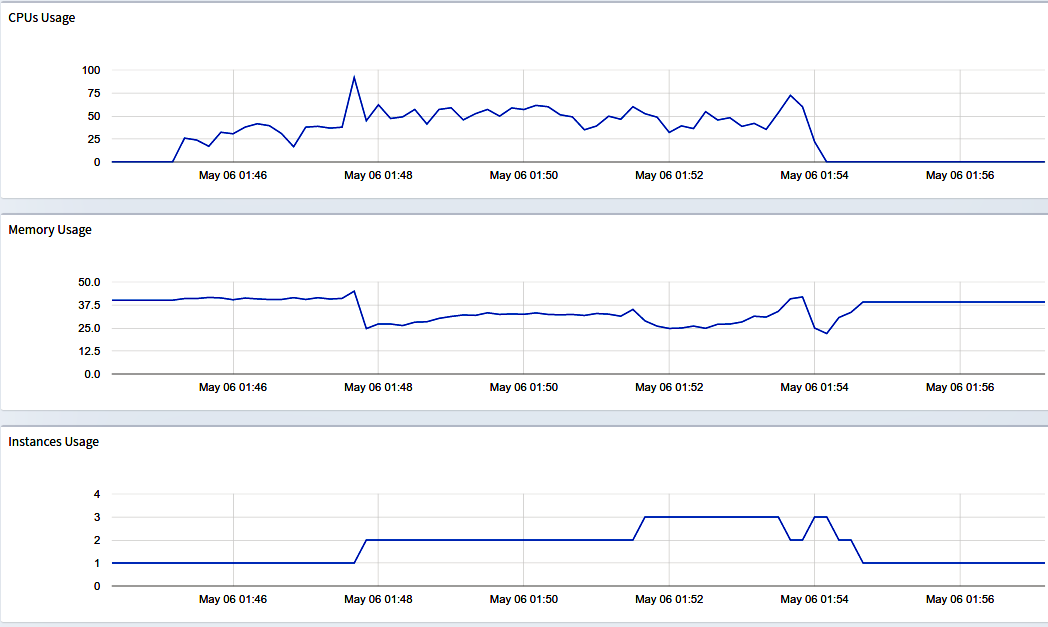
1. Kích hoạt chức năng autoscaling cho ứng dụng
2. Chạy kịch bản được cung cấp trên git bằng jmeter. Kịch bản sẽ được chạy 4 lần, mỗi lần sẽ thay đổi số lượng thread (số lượng người dùng giả lập) gửi request đến ứng dụng lần lượt là 0, 1, 5, 15.
3. Ghi lại kết quả về số lượng instances, cpu trung bình, memory trung bình của ứng dụng, số request lỗi.

* Kết quả kiểm thử:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Số thread | Số instances | Cpu trung bình | Memory trung bình | Request lỗi (%) |
| 0 | 1 | 0.001% | 36% | 0% |
| 1 | 1 | 34% | 37% | 0% |
| 5 | 2 | 55% | 32% | 0.08% |
| 15 | 3 | 48% | 28% | 0.36% |

Bảng 10: Kết quả kiểm thử chức năng autoscaling

Biểu đồ thể hiện các metrics sử dụng của ứng dụng trong quá trình kiểm thử:



Hình 52: Biểu đồ metrics

* Nhận xét: Chức năng autoscaling hoạt động tốt, số request lỗi là rất thấp.

# CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## 5.1 Kết luận

**Đồ án đã thu được những kết quả sau:**

1. Xây dựng hệ thống quản lí cluster mesosphere

Hệ thống được cài đặt trên 3 node (3 con VM). 1 node vừa là master vừa là slave, 2 node còn lại là slave.

1. Xây dựng các service giúp autoscaling ứng dụng

Các service bao gồm:

* Collector: được viết bằng python, cài đặt trên mỗi node slave, dùng để theo dõi nguồn tài nguyên sử dụng của ứng dụng web và lưu trữ nó vào influxdb
* Autoscaling: Dựa vào tài nguyên sử dụng của ứng dụng web và bộ luật mà người dùng cung cấp để ra quyết định scale ứng dụng tới marathon.

1. Đóng gói ứng dụng web từ mã nguồn trên github

* Tạo một docker image (apache-php) từ Dockerfile và quy trình đóng gói ứng dụng từ mã nguồn vào image này.

1. Xây dựng được kiến trúc ứng dụng web có khả năng scale

Bao gồm 2 phần:

* Đồng bộ các file dữ liệu của các instance ứng dụng bằng cách sử dụng docker volumn plugin
* Hệ thống cân bằng tải HAproxy

1. Xây dựng giao diện người dùng giúp người dùng tương tác với hệ thống

Bao gồm các phần:

* Bộ RESTful API để tương tác với hệ thống
* Giao diện web bằng angular
* Giao diện dòng lệnh bằng python

Cả giao diện web và dòng lệnh đều tương tác thông qua bộ RESTful API

1. Chạy thử nghiệm hệ thống trên local

* Cài đặt hệ thống thử nghiệm với mô hình 3 node
* Sử dụng jmeter để giả lập thao tác người dùng
* Thử nghiệm các chức năng: Tạo ứng dụng database, Tạo ứng dụng web và autoscaling ứng dụng web

**Đóng góp của đồ án là:**

Cung cấp một open source platform autoscaling được public tại địa chỉ web https://github.com/cuongnb14/autoscaling-paas.

## 5.2 Hướng phát triển

### 5.2.1 Áp dụng phương pháp dự đoán tiêu dùng tài nguyên (Prediction Resource Consumption)

**Mô tả:**

Hiện tại hệ thống sẽ bắt đầu salce ứng dụng khi nguồn tài nguyên đạt ngưỡng, khi đó hệ thống sẽ phải chịu quá tải trong một thời gian. Thực hiện dự đoán trước tài nguyên tiêu dùng để tiến hành scale trước, tránh vấn đề này.

**Hướng tiếp cận:**

Sử dụng mô hình học máy mạng nơ-ron nhân tạo để dự đoán cpu và memory sẽ được ứng dụng sử dụng. Đầu vào sẽ là cpu và memory sử dụng trung bình của n khoảng thời gian trước đó và đầu ra là cpu và memory sử dụng của khoảng thời gian tiếp theo.

### 5.2.2 Migrate các instance dồn về một số node

**Mô tả:**

Khi di chuyển các instance dồn về một số node, một số node còn lại sẽ không có instance nào chạy, ta có thể tắt điện những node đó để tiết kiệm điện, bảo trì server, …

**Hướng tiếp cận:**

Chúng ta có thể thêm các service để theo dõi nguồn tài nguyên sử dụng của các node và thực hiện tắt một số node khi thấy nguồn tài nguyên dư thừa quá nhiều. Sau khi tắt marathon sẽ tự động khởi tạo một số instances trên các node đang còn chạy thay cho các instances nằm trên các node bị tắt.

### 5.2.3 Autoscaling cho database

**Mô tả:**

Áp dụng autoscaling cho database để tránh việc database bị quá tải khi có nhiều kết nối đến. Việc scale database sẽ phải giải quyết được vấn đề nhất quán dữ liệu cho các instances database.

**Hướng tiếp cận:**

*Giới thiệu Galera:* Galera cluster là một giải pháp multi master cho database. Sử dụng galera cluster, application có thể read/write database trên bất cứ node nào.

Chúng ta có thể sử dụng các dịch vụ autoscaling có sẵn để thực hiện scale database và sử dụng Galera để đồng bộ dữ liệu giữa các instances.

# TÀI LIỆU THAM KHẢO

**SÁCH BÁO**

[1] Tania Lorido-Botrán, José Miguel-Alonso, José A. Lozano. Auto-scaling Techniques for Elastic Applications inCloud Environments. Technical Report EHU-KAT-IK-09-12. Department of Computer Architecture and TechnologyUniversity of the Basque Country, 2012

**INTERNET**

[1] Marc Cluet. URL: <http://www.slideshare.net/lynxmanuk/autoscaling-best-practices>, truy cập lần cuối 2/4/2016

[2] Brian Christner. URL: <https://www.brianchristner.io/how-to-setup-docker-monitoring>, truy cập lần cuối 20/4/2016

[3] Jérôme Petazzoni. URL: <http://www.slideshare.net/jpetazzo/docker-and-containers-for-development-and-deployment-scale12x>, truy cập lần cuối 21/4/2016

**PHẦN MỀM**

1. Mesosphere. URL: <https://mesosphere.com/>
2. Influxdb. URL: <https://influxdata.com/>
3. Docker. URL: <https://www.docker.com/>
4. Jmeter. URL: <http://jmeter.apache.org/>
5. Django. URL: <https://www.djangoproject.com/>
6. Angularjs. URL: <https://angularjs.org/>
7. marathon-python. URL: <https://github.com/thefactory/marathon-python>
8. marathon-lb. URL: <https://github.com/mesosphere/marathon-lb>
9. Opencart. URL: <http://www.opencart.com/>
10. AdminLTE: URL: <https://github.com/almasaeed2010/AdminLTE>
11. Netshare. URL: <http://netshare.containx.io/>

# PHỤ LỤC

## Phần lục 1: Cài đặt và cấu hình hệ thống mesosphere

Mesosphere có nhiều thành phần và để các thành phần kết nối được với nhau ta cần các bước cấu hình rất phức tạp. Để dễ dàng hơn trong việc cài đặt và cấu hình ta có thể sử dụng các script có sẵn tại địa chỉ:

<https://github.com/cuongnb14/autoscaling-paas/tree/master/setup> .

### Cài đặt

Tất cả các node đều cài đặt gói mesosphere và docker theo 2 script là install-mesos.sh và docker-1.9.sh. Nếu gặp lỗi “mesosphere : Depends: marathon but it is not going to be installed” thì do node chưa được cài jdk8, giải quyết bằng cách cài jdk8 sau đó chạy lại script.

Sau khi chạy script các node sẽ được cài mesos-master, mesos-slave, zookeeper, marathon và docker.

### Cấu hình

* Với master node sử dụng script master.sh để cài đặt ( chú ý mặc định khi chạy script này sẽ cấu hình mesos-master và mesos-slave chạy trên node nếu chỉ muốn node làm master thì comment phần cấu hình cho mesos-slave lại). Quá trình cấu hình trong script lần lượt theo:
* Cấu hình thông tin kết nối Zookeeper cho Mesos
* Cấu hình mesos-master
* Cấu hình marathon
* Cấu hình mesos-slave
* Cấu hình khởi động cùng server cho các dịch vụ trên.

Nếu gặp lỗi không khởi động được slave thì là do cache của slave, giải quyết bằng cách xóa thư mục / tmp/mesos

* Với slave node sử dụng script slave.sh để cấu hình. Quá trình cấu hình lần lượt theo:
* Cấu hình thông tin kết nối Zookeeper cho Mesos
* Dừng các dịch vụ mesos master, zookeeper, marathon
* Cấu hình cho mesos-slave

## Phần lục 2: Dockerfile cho apache-php images

Dockerfile là 1 file text chứa các lệnh thực thi để build 1 image cho docker. Với dockerfile chúng ta có một phương tiện tiện dụng để build các image tự động và chia sẻ với nhau dễ dàng.

Docker file apache-php sẽ có các phần chủ yếu sau:

1. FROM php:5.6-apache: Image apache-php được xây dựng dựa trên image php:5.6-apache có sẵn. Khi đó apache-php sẽ được cài sẵn php phiên bản 5.6 và apache server.
2. RUN a2enmod rewrite: Bật mod rewrite cho apache server
3. RUN set -xe \

&& apt-get update \

&& apt-get install -y libpng12-dev libjpeg-dev libmcrypt-dev \

&& rm -rf /var/lib/apt/lists/\* \

&& docker-php-ext-configure gd --with-png-dir=/usr --with-jpeg-dir=/usr \

&& docker-php-ext-install gd mcrypt mbstring mysqli zip \

&& chown -R www-data:www-data /var/www/html

Cài đặt các thư viện, các gói mở rộng cần thiết cho ứng dụng web

Chi tiết file Dockerfile xem tại:

<https://github.com/cuongnb14/autoscaling-paas/blob/master/project/Dockerfile>

## Phần lục 3: Hướng dẫn sử dụng console tool (ASClient)

**ASClient** là một tool sử dụng giao diện dòng lệnh để thao tác với hệ thống autoscaling được viết bằng python.

Vì có giao diện dòng lệnh nên sẽ không đầy đủ tính năng như trên giao diện web (ví dụ như chức năng xem biểu đồ nguồn tài nguyên).

**Yêu cầu hệ thống:**

* Được cài đặt sẵn python phiên bản 3 trở lên

**Cài đặt:**

* Tải file **autoscaling\_tool.py** tại địa chỉ: <https://raw.githubusercontent.com/cuongnb14/autoscaling-paas/master/project/console_tool/autoscaling_tool.py>

**Cách sử dụng:**

Bước 1: Vào terminal và chạy file autoscaling\_tool.py

Bước 2: Nhập thông tin xác thực tài khoản người dùng (username và password)

Bước 3: Nếu xác thực thành công người dùng có thể bắt đầu thực hiện các thao tác với hệ thống bằng các dòng lệnh.

**Tổng hợp các lệnh:**

|  |  |
| --- | --- |
| Command | Mô tả |
| app --all | Hiển thị danh sách các ứng dụng web |
| app --name <app\_name> | Hiển thị thông tin của ứng dụng <app\_name> |
| app create --file <file\_name> | Tạo ứng dụng web mới với các thông tin cấu hình trong file <file\_name> |
| app delete --name <app\_name> | Xóa ứng dụng web có tên <app\_name> |
| app policy --list <app\_name> | Hiển thị danh sách luật của ứng dụng <app\_name> |
| app policy --delete <app\_name> <pid> | Xóa luật có id là <pid> của ứng dụng <app\_name> |
| app policy --add <app\_name> --file <file\_name> | Thêm luật mới cho ứng dụng web <app\_name> với các thông tin trong file <file\_name> |
| exit | Thoát khỏi chương trình |