

ĐẠI HỌC QUỐC GIA HÀ NỘI  
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN  
KHOA VẬT LÝ

Sinh viên: Lã Anh Trúc

**XÂY DỰNG HỆ THỐNG TÍNH TOÁN  
SONG SONG DỰA TRÊN KIẾN TRÚC IRONIC  
OPENSTACK BAREMETAL VÀ HĐH DEBIAN**

Khóa luận tốt nghiệp đại học hệ chính quy

Ngành: Kỹ thuật điện tử và Tin học

(Chương trình đào tạo chuẩn)

**HÀ NỘI – 2024**

ĐẠI HỌC QUỐC GIA HÀ NỘI  
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN  
KHOA VẬT LÝ

Sinh viên: **Lã Anh Trúc**

**XÂY DỰNG HỆ THỐNG TÍNH TOÁN  
SONG SONG DỰA TRÊN KIẾN TRÚC IRONIC  
OPENSTACK BAREMETAL VÀ HỆ ĐIỀU HÀNH DEBIAN**

Khóa luận tốt nghiệp đại học hệ chính quy

Ngành: Kỹ thuật điện tử và Tin học

(Chương trình đào tạo chuẩn)

Cán bộ hướng dẫn: **TS. Nguyễn Hồng Quang**

**HÀ NỘI – 2024**

## **Lời cảm ơn**

Trước tiên, em xin gửi lời cảm ơn chân thành và sâu sắc nhất đến Tiến sĩ – giảng viên, thầy Nguyễn Hồng Quang vì đã tận tình giúp đỡ và hướng dẫn em trong suốt quá trình làm và bảo vệ đồ án tốt nghiệp. Sự hỗ trợ nhiệt tình và những lời khuyên quý báu của Thầy đã giúp em vượt qua những khó khăn, thử thách và hoàn thành tốt công việc của mình. Thầy đã luôn kiên nhẫn, tận tâm và sẵn lòng chia sẻ những kinh nghiệm, kiến thức quý báu mà Thầy đã tích lũy được qua nhiều năm. Điều này không chỉ giúp em nâng cao năng lực chuyên môn mà còn giúp em phát triển kỹ năng tư duy và làm việc.

Em cũng xin gửi lời cảm ơn sâu sắc đến tất cả các thầy cô trong khoa Vật Lý đã luôn nhiệt tình giảng dạy, hỗ trợ và tạo điều kiện tốt nhất cho em và các sinh viên khác trong suốt thời gian học tập tại trường. Những kiến thức và kinh nghiệm mà các thầy cô đã truyền đạt không chỉ giúp em hoàn thiện đồ án tốt nghiệp mà còn là nền tảng vững chắc cho sự phát triển nghề nghiệp trong tương lai.

Ngoài ra, em cũng muốn gửi lời cảm ơn đến trường đại học Khoa học tự nhiên, ĐHQGHN - nơi đã tạo ra môi trường học tập lý tưởng, cung cấp cơ sở vật chất hiện đại và đầy đủ, cũng như tổ chức nhiều hoạt động bổ ích giúp sinh viên phát triển toàn diện.

Lời cuối cùng, em xin kính chúc thầy Nguyễn Hồng Quang, các thầy cô và toàn thể nhà trường luôn mạnh khỏe, hạnh phúc và thành công trong sự nghiệp trồng người cao quý.

Hà Nội, tháng 6 năm 2024

Sinh viên,

Lã Anh Trúc

# Mục lục

Tên đầu mục	Trang
<b>Mở đầu</b> .....	1
<b>Nội dung</b> .....	2
<b>Chương 1: Tổng quan</b>	
1. Giới thiệu Openstack Bare Metal - Ironic	
1.1. Openstack và Ironic .....	2
1.2. Lịch sử hình thành và phát triển .....	3
1.3. Tại sao nên sử dụng Ironic?.....	4
2. Các thành phần chính trong Ironic	
2.1. Các thành phần cốt lõi.....	7
2.2. Các thành phần có thể tích hợp.....	8
2.3. Kiến trúc hệ thống Ironic .....	9
3. Các thành phần khác	
3.1. Hệ điều hành Debian.....	11
3.2. Hệ quản trị cơ sở dữ liệu MySQL/ MariaDB .....	12
3.3. Máy chủ dịch vụ web Apache2.....	13
3.4. Dịch vụ phân phối thông điệp RabbitMQ .....	14
 <b>Chương 2: Cài đặt hệ thống Ironic trên hệ điều hành Debian</b>	
1. Triển khai Ironic trên hệ điều hành Debian .....	15
2. Cài đặt hệ quản trị cơ sở dữ liệu MariaDB .....	16
3. Cài đặt dịch vụ xác thực Openstack Keystone .....	17
4. Cài đặt dịch vụ phân phối thông điệp RabbimtMQ .....	19
5. Cài đặt máy chủ web Apache2.....	20
6. Cài đặt thành phần chính trong Ironic.....	21
6.1. Nền tảng các công nghệ ảo hóa .....	21
6.2. Cấu hình các thành phần chính trong Ironic .....	23

## **Chương 3: Xây dựng hệ thống tính toán song song**

1. Khái niệm tính toán song song.....	24
2. Xây dựng bài toán tính toán song song.....	25
3. Triển khai hệ thống tính toán song song.....	26

<b>Kết luận</b> .....	30
-----------------------	----

<b>Danh mục tài liệu tham khảo</b> .....	32
--	----

## **Phụ lục**

1. Cài đặt MariaDB .....	34
1.1. Hướng dẫn cài đặt.....	34
1.2. Lỗi khi cài đặt MariaDB .....	34
2. Cài đặt Apache2.....	35
2.1. Hướng dẫn cài đặt.....	35
2.2. Lỗi khi cài đặt Apache2 .....	36
3. Cài đặt RabbitMQ.....	36
3.1. Hướng dẫn cài đặt.....	36
3.2. Lỗi khi cài đặt RabbitMQ.....	39
4. Cài đặt Keystone.....	41
4.1. Hướng dẫn cài đặt.....	41
4.2. Lỗi khi cài đặt Keystone.....	45
5. Cài đặt Ironic .....	47
5.1. Cài đặt dịch vụ ironic-api .....	47
5.2. Cấu hình dịch vụ ironic conductor.....	48
5.3. Các lỗi khi cài đặt.....	53
6. Cấu hình giao diện khởi động cho hệ thống .....	55
7. Cấu hình trình điều khiển cho hệ thống .....	58
8. Các lỗi khác .....	59
9. Tạo và cấu hình máy chủ vật lý.....	60
10. Triển khai máy chủ vật lý .....	62
11. Chương trình tính toán song song đơn giản.....	63

## Mở đầu

Trong thời đại số hóa hiện nay, việc tính toán song song trở thành một phần không thể thiếu của hầu hết các tổ chức và doanh nghiệp. Tính toán song song không chỉ đơn thuần là một công cụ để tăng cường hiệu suất tính toán mà còn mở ra những cơ hội mới trong việc phân tích dữ liệu, triển khai ứng dụng phức tạp, và nghiên cứu khoa học.

Việc quản lý và triển khai cơ sở hạ tầng tính toán song song đòi hỏi sự linh hoạt và hiệu quả cao. OpenStack Bare Metal (IroniC) nổi lên như một giải pháp linh hoạt và mạnh mẽ cho các tổ chức với nhu cầu sử dụng cơ sở hạ tầng vật lý để triển khai các ứng dụng và công việc tính toán một cách hiệu quả.

IroniC là một hệ thống cung cấp máy chủ vật lý mạnh mẽ, giúp người dùng quản lý và tự động hóa việc triển khai hạ tầng, quản lý tài nguyên đến vòng đời của các máy chủ. Thay vì phải xử lý các công việc triển khai máy chủ một cách thủ công và tốn kém, IroniC cho phép tự động hóa quy trình này, giúp tiết kiệm thời gian và tài nguyên của tổ chức.

Sự kết hợp giữa tính linh hoạt của IroniC và khả năng mở rộng của tính toán song song mở ra những cơ hội mới trong việc xây dựng và quản lý cơ sở hạ tầng máy chủ. Điều này không chỉ giúp tối ưu hóa việc sử dụng tài nguyên mà còn đảm bảo tính linh hoạt và hiệu suất của hạ tầng tính toán, tiết kiệm chi phí đầu tư.

Mục tiêu: Xây dựng hệ thống tính toán song song trên hệ điều hành Debian và kiến trúc Openstack Bare Metal.

Đối tượng và phạm vi nghiên cứu: Hệ thống Openstack Bare Metal IroniC trên hệ điều hành Debian.

Phương pháp nghiên cứu: Phương pháp phân tích – tổng hợp, phương pháp liệt kê, phương pháp so sánh.

# **Chương 1. Tổng quan**

## **1. Giới thiệu Openstack Bare Metal – Ironic**

### **1.1. Openstack và Ironic**

OpenStack là một nền tảng điện toán đám mây mã nguồn mở và miễn phí. Nó chủ yếu được triển khai cơ sở hạ tầng dưới dạng như một dịch vụ (infrastructure-as-a-service IaaS) trong cả đám mây công cộng và đám mây riêng, nơi các máy chủ ảo và các tài nguyên khác được cung cấp cho người dùng.

Nền tảng phần mềm này bao gồm các thành phần liên quan giúp kiểm soát các tài nguyên phần cứng đa dạng từ nhiều nhà cung cấp như xử lý, lưu trữ và mạng trong toàn bộ trung tâm dữ liệu. Người dùng có thể quản lý nó thông qua bảng điều khiển, các công cụ dòng lệnh, hoặc các dịch vụ web.

OpenStack Bare Metal hay còn được biết đến như Ironic, là một dự án quan trọng trong hệ sinh thái OpenStack, tập trung vào việc cung cấp các tài nguyên phần cứng vật lý thay vì các máy ảo. Nó có thể được coi là bộ giao diện lập trình ứng dụng cho việc cung cấp, quản lý và tương tác với các trình ảo hóa các máy vật lý.

Trái hoàn toàn với các hệ thống cung cấp máy ảo như KVM, XEN hay VMWare, Virtualbox thường hoạt động dựa trên bộ ảo hóa, Ironic là dịch vụ cung cấp các máy chủ vật lý mà có thể quản lý chúng như máy ảo. Các máy vật lý được cung cấp này đều được sử dụng toàn quyền bởi người dùng kể cả trong việc quản lý các yếu tố từ phần cứng đến phần mềm.

Bộ mã nguồn của Ironic được phát triển hoàn toàn dựa trên ngôn ngữ lập trình Python và là mã nguồn mở, được công khai trên các nền tảng quản lý và lưu trữ mã nguồn như OpenDev và Github. [1]

## 1.2. Lịch sử hình thành và phát triển

Vào tháng 7 năm 2010, Rackspace Hosting và NASA đã công bố một sáng kiến phần mềm đám mây mã nguồn mở được gọi là OpenStack. Tuyên bố sứ mệnh tạo ra nền tảng điện toán đám mây mã nguồn mở phổ biến đáp ứng nhu cầu của các đám mây công cộng và đám mây riêng bất kể kích thước, bằng cách dễ dàng triển khai và có khả năng mở rộng lớn.

Dự án nhằm giúp các tổ chức cung cấp dịch vụ điện toán đám mây chạy trên phần cứng tiêu chuẩn. Phiên bản chính thức đầu tiên của cộng đồng, có tên mã là Austin, xuất hiện ba tháng sau đó vào ngày 21 tháng 10 năm 2010, với kế hoạch phát hành các bản cập nhật phần mềm thường xuyên. Mã ban đầu đến từ nền tảng Nebula của NASA cũng như từ nền tảng Cloud Files của Rackspace. Các mô đun của cloud stack và open stack đã được hợp nhất và phát hành dưới dạng mã nguồn mở bởi đội ngũ NASA Nebula kết hợp với Rackspace.

Năm 2011, nhà tài trợ của Ubuntu là Canonical đã giới thiệu việc hỗ trợ đầy đủ cho các dịch vụ điện toán OpenStack, bắt đầu từ bản phát hành của OpenStack Cactus.

Cùng trong năm 2011, OpenStack phiên bản Cactus cũng đã có sẵn trong Debian 7.0.

Vào tháng 8 năm 2012, SUSE đã công bố phiên bản OpenStack doanh nghiệp được hỗ trợ thương mại dựa trên bản phát hành "Essex".

Sau một năm phát hành bản thử nghiệm tích hợp OpenStack (phiên bản "Essex") của mình, tập đoàn Red Hat đã giới thiệu hỗ trợ thương mại cho OpenStack với bản phát hành "Grizzly" vào tháng 7 năm 2013.

Năm 2012, OpenStack đã được chuyển sang quản lý bởi OpenStack Foundation, một tổ chức phi lợi nhuận được thành lập vào tháng 9 năm 2012 nhằm thúc đẩy phần mềm OpenStack và cộng đồng của nó.

Đến năm 2018, hơn 500 công ty đã tham gia dự án này. Năm 2020, tổ chức này thông báo đổi tên thành Open Infrastructure Foundation năm 2021. Đến năm 2024, OpenStack đã được hỗ trợ bởi hơn 560 nhà tổ chức, cung cấp



cho hơn 300 trung tâm điện toán đám mây công cộng, sở hữu cộng đồng phát triển trải dài khắp 180 nước với hơn 9000 người đóng góp xây dựng.

Ngày 30/4/2015, OpenStack Ironic được giới thiệu lần đầu tiên trong phiên bản OpenStack “Kilo”. Dự án này bắt đầu như một phần của OpenStack Nova, module quản lý tính toán của OpenStack, nhằm mục đích hỗ trợ triển khai và quản lý các máy chủ vật lý (bare metal) thay vì các máy ảo.

Hoàn cảnh ra đời của Ironic xuất phát từ nhu cầu ngày càng tăng của các tổ chức và doanh nghiệp về khả năng truy cập trực tiếp vào phần cứng thực để đạt được hiệu suất cao hơn, đặc biệt là trong các môi trường yêu cầu xử lý hiệu năng cao và các ứng dụng không thể ảo hóa. Ban đầu, việc quản lý máy chủ vật lý được thực hiện thông qua một tính năng của Nova có tên là "bare metal driver". Tuy nhiên, do tính phức tạp và yêu cầu chuyên biệt hóa trong việc quản lý phần cứng vật lý, cộng đồng OpenStack đã quyết định tách tính năng này ra thành một dự án riêng biệt, từ đó hình thành nên Ironic.

Ironic nhanh chóng trở thành một dự án độc lập và quan trọng trong hệ sinh thái OpenStack, cung cấp các công cụ mạnh mẽ để triển khai, quản lý và tái sử dụng các tài nguyên phần cứng vật lý trong môi trường đám mây. Điều này giúp các tổ chức không chỉ tận dụng tối đa hiệu suất của phần cứng mà còn duy trì được tính linh hoạt và khả năng mở rộng của các giải pháp điện toán đám mây. [2]

### 1.3. Tại sao nên sử dụng Ironic ?

Ironic là một công cụ mạnh mẽ và cần thiết cho việc quản lý và triển khai các máy chủ vật lý, phù hợp với việc xây dựng nền móng cho một hay nhiều hệ thống tính toán song song với những lý do có thể kể đến như sau:

- Mã nguồn mở và sử dụng miễn phí: Ironic là một dự án mã nguồn mở, nghĩa là nó không chỉ miễn phí mà còn cho phép người dùng tùy chỉnh và mở rộng theo nhu cầu riêng. Điều này giúp tiết kiệm chi phí đầu tư ban đầu và chi phí duy trì hệ thống.

- Cộng đồng sử dụng mạnh mẽ: Với một cộng đồng lớn và năng động, người dùng Ironic có thể dễ dàng tìm thấy sự hỗ trợ, tài liệu hướng dẫn và các giải pháp từ cộng đồng. Cộng đồng OpenStack thường xuyên cập nhật và cải tiến phần mềm, giúp người dùng tiếp cận được những tính năng mới nhất và các bản vá lỗi kịp thời.

- Hỗ trợ lâu dài từ OpenStack: OpenStack là một trong những công ty hàng đầu về công nghệ đám mây, với sự hỗ trợ lâu dài và cam kết từ nhiều công ty công nghệ lớn trên thế giới. Việc sử dụng Ironic giúp đảm bảo rằng hệ thống của người dùng luôn được cập nhật và hỗ trợ tốt nhất.

- Hỗ trợ kỹ thuật và tài liệu chi tiết: OpenStack cung cấp tài liệu chi tiết và hỗ trợ kỹ thuật chuyên sâu cho người dùng Ironic. Điều này giúp đảm bảo rằng người dùng có thể triển khai và quản lý hệ thống một cách hiệu quả và giải quyết các vấn đề phát sinh nhanh chóng.

- Tính bảo mật và tuân thủ quy định: Ironic cho phép kiểm soát hoàn toàn phần cứng vật lý, giúp đáp ứng các yêu cầu khắt khe về bảo mật và tuân thủ các quy định pháp lý. Điều này đặc biệt quan trọng trong các lĩnh vực như tài chính, y tế và chính phủ, nơi bảo mật dữ liệu là ưu tiên hàng đầu.

- Tính linh hoạt và khả năng mở rộng: Ironic cung cấp khả năng quản lý và triển khai linh hoạt các máy chủ vật lý, cho phép mở rộng hoặc thu hẹp quy mô hệ thống theo nhu cầu thực tế. Điều này giúp tối ưu hóa việc sử dụng tài nguyên và giảm thiểu chi phí vận hành.

- Cung cấp các cụm máy tính vật lý hiệu năng cao: Đối với những hệ thống yêu cầu sức mạnh tính toán lớn và hiệu suất cao, việc sử dụng máy chủ vật lý là rất cần thiết. Ironic giúp triển khai và quản lý các cụm máy tính vật lý này một cách hiệu quả, đảm bảo khả năng xử lý khối lượng công việc lớn với độ trễ thấp và hiệu suất tối ưu.

- Phục vụ linh hoạt các tác vụ tính toán đòi hỏi quyền truy cập vào các thiết bị phần cứng không thể ảo hóa: Một số ứng dụng và tác vụ yêu cầu truy cập trực tiếp vào phần cứng, chẳng hạn như các thiết bị thẻ đồ họa rời (GPU), mạch tích hợp cỡ lớn FPGA, hoặc các thiết bị đặc thù khác mà không thể ảo hóa. Ironic cho phép triển khai và quản lý các máy chủ vật lý có khả năng truy cập trực tiếp đến các thiết bị này, đáp ứng nhu cầu đặc thù của các ứng dụng chuyên dụng.

- Lưu trữ cơ sở dữ liệu hiệu quả: Một số cơ sở dữ liệu có hiệu suất kém trong môi trường ảo hóa do yêu cầu về đầu vào, đầu ra và tài nguyên phần cứng cao. Sử dụng máy chủ vật lý với Ironic giúp đảm bảo hiệu suất cao và ổn định cho các cơ sở dữ liệu này, cải thiện tốc độ truy xuất và xử lý dữ liệu.

- Đảm bảo phần cứng dành riêng cho một máy chủ duy nhất: Đối với những trường hợp yêu cầu về hiệu suất, bảo mật và độ tin cậy cao, việc sử dụng phần cứng dành riêng cho một máy chủ duy nhất là rất quan trọng. Ironic hỗ trợ việc phân bổ phần cứng một cách chính xác và an toàn, đáp ứng các yêu cầu khắt khe về bảo mật và tuân thủ quy định của các tổ chức.

- Triển khai cơ sở hạ tầng đám mây một cách nhanh chóng: Ironic giúp tự động hóa quá trình triển khai và quản lý các máy chủ vật lý, cho phép xây dựng cơ sở hạ tầng đám mây một cách nhanh chóng và hiệu quả. Điều này giúp tiết kiệm thời gian và chi phí, đồng thời tăng khả năng mở rộng và linh hoạt của hệ thống.

## 2. Các thành phần chính trong Ironic

### 2.1. Các thành phần cốt lõi

**Ironic-api:** Là một API RESTful (giao diện lập trình ứng dụng quản lý tài nguyên) xử lý các yêu cầu ứng dụng bằng cách gửi chúng đến Ironic-conductor qua phương thức gọi hàm từ xa (RPC - remote procedure call). Nó có thể chạy qua cổng giao diện máy chủ web (WSGI - Web Server Gateway Interface) hoặc như một tiến trình riêng biệt.

**Ironic-conductor:** Là nơi tập trung phần lớn việc xử lý logic của dịch vụ Ironic. Đây là bộ thành phần mạnh mẽ giúp thêm/sửa/xóa/bật/tắt các máy chủ vật lý bằng IPMI (giao diện quản lý nền tảng thông minh) hoặc giao thức riêng biệt khác, cung cấp/triển khai/dọn dẹp các máy chủ vật lý. Ironic-conductor sử dụng trình điều khiển (drivers) để thực hiện các thao tác trên phần cứng.

**Ironic-python-agent:** Một dịch vụ python chạy trong một đĩa RAM tạm thời để cung cấp các dịch vụ Ironic-Conductor và Ironic-Inspector với khả năng truy cập từ xa, kiểm soát phân và kiểm tra cấu hình phần cứng.

**Ironic-Inspector:** Một dịch vụ liên quan thực hiện việc xem xét phần cứng bằng cách sử dụng PXE boot để khởi động phần cứng chưa đăng ký vào đĩa ram đang chạy trên Ironic-python-agent. [3]

## 2.2. Các thành phần có thể tích hợp

Ironic cũng có thể được sử dụng độc lập hoặc là một phần của OpenStack Cloud khi tích hợp với các dịch vụ đi kèm như:

- OpenStack Keystone (Dịch vụ xác thực): Keystone là một dịch vụ OpenStack cung cấp xác thực ứng dụng khách (API Client), khám phá dịch vụ và ủy quyền các bên được phân phối bằng cách triển khai API nhận dạng của OpenStack.

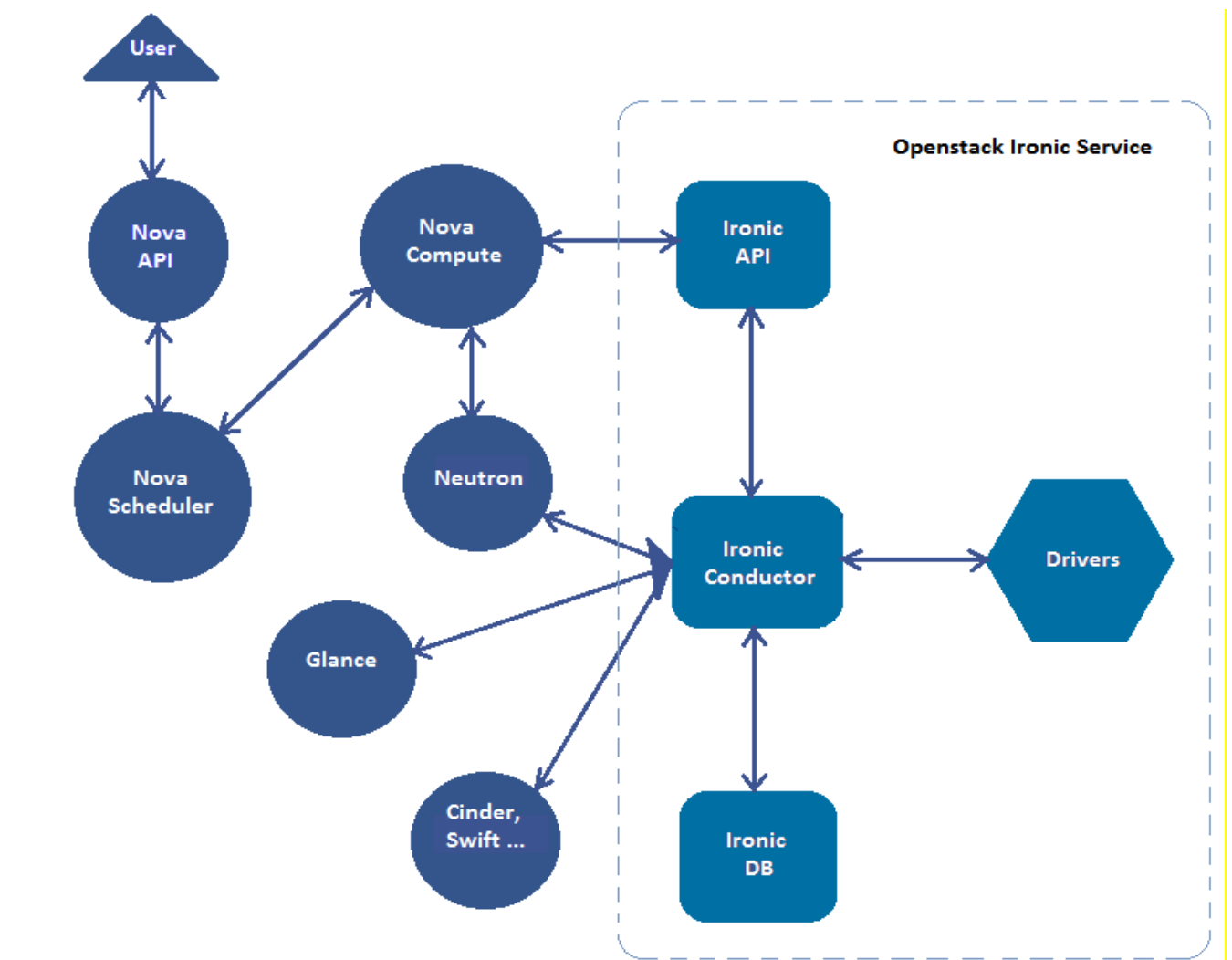
- OpenStack Nova (Dịch vụ điện toán): Nova là dự án OpenStack cung cấp các phiên bản điện toán (còn gọi là máy chủ ảo). Ngoài ra Nova hỗ trợ tạo máy ảo, máy chủ baremetal (thông qua việc sử dụng Ironic) và hỗ trợ hạn chế cho các bộ chứa hệ thống. Nova chạy như một tập hợp các daemon (trình nền) trên các máy chủ Linux hiện có để cung cấp dịch vụ đó.

- OpenStack Neutron (Các dịch vụ mạng): Neutron là một dự án OpenStack nhằm cung cấp "kết nối mạng như một dịch vụ" giữa các thiết bị giao diện được quản lý bởi các dịch vụ Openstack khác như Nova.

- OpenSack Glance (Dịch vụ lưu trữ ảnh đĩa): Các dịch vụ liên quan tới image bao gồm khám phá, đăng ký và truy xuất image máy ảo (VM). Glance có API RESTful cho phép truy vấn siêu dữ liệu image VM cũng như truy xuất image thực tế.

- Openstack Swift (Dịch vụ quản lý đối tượng): Swift là một bộ lưu trữ đối tượng/blob (tệp dữ liệu thuần bất biến) có tính sẵn sàng cao, được phân phối và nhất quán một cách hiệu quả, an toàn và tiết kiệm. [4]

### 2.3. Kiến trúc hệ thống Ironic



Hình 1.1: Kiến trúc logic của hệ thống Ironic  
(Nguồn: [Openstack Document](#))

Sơ đồ trong hình 1.1 cho thấy kiến trúc logic của hệ thống. Nó hiển thị các thành phần cơ bản hình thành nên dịch vụ Ironic, mối quan hệ của dịch vụ Ironic với các dịch vụ OpenStack khác và luồng logic của việc cung cấp máy chủ vật lý để xử lý các yêu cầu từ người dùng.

Đầu tiên, yêu cầu khởi chạy dịch vụ của người dùng được chuyển đến dịch vụ tính toán thông qua giao diện lập trình ứng dụng điện toán (Nova API) và bộ lập lịch điện toán. Dịch vụ điện toán sử dụng các trình điều khiển ảo (đã được cấu hình trong Ironic) để chuyển yêu cầu xử lý đến các thành phần chính của Ironic, trong đó yêu cầu sẽ bắt đầu chuyển từ giao diện lập trình của Ironic (Ironic-api) sang tới Ironic-conductor và cuối cùng là qua các trình điều khiển (Drivers) để cung cấp thành công máy chủ vật lý cho người dùng. Sau đó, dịch vụ Ironic cũng giao tiếp với các dịch vụ Openstack khác đã được tích hợp như bộ lưu trữ đĩa mềm (Glance), mạng (Neutron), đối tượng (Swift),.. nhằm cung cấp phiên bản máy chủ vật lý. [5]

### **3. Các thành phần khác**

#### **3.1. Hệ điều hành Debian**

Debian là một hệ điều hành máy tính phổ biến được cấu thành hoàn toàn từ phần mềm tự do, được phát triển từ sự cộng tác của các tình nguyện viên trên khắp thế giới trong Dự án Debian. Dự án Debian lần đầu tiên được công bố vào ngày 16 tháng 8 năm 1993 bởi Ian Murdock. Debian 0,01 được phát hành vào ngày 15 tháng 9 năm 1993, và bản phát hành stable (ổn định) đầu tiên được phát hành vào năm 1996. Nhánh phát hành stable là phiên bản phổ biến nhất cho máy tính cá nhân và máy chủ sử dụng Debian, và được sử dụng làm nền tảng cho nhiều bản phân phối khác.

Hiện tại có rất nhiều hệ điều hành Linux được xây dựng dựa trên Debian GNU/Linux, trong đó có Ubuntu, Linux Mint, Knoppix, MEPIS, DreamLinux, Damn Small Linux và các hệ điều hành khác. Debian nổi tiếng với hệ thống quản lý gói của nó, mà cụ thể APT (công cụ quản lý gói cao cấp, Advanced Packaging Tool), chính sách nghiêm ngặt đối với chất lượng các gói và bản phát hành, cũng như tiến trình phát triển và kiểm tra mở. Cách thức làm việc này đã giúp cho việc nâng cấp giữa các bản phát hành và việc cài đặt hay gỡ bỏ các gói phần mềm được dễ dàng hơn.

Debian luôn luôn có ít nhất ba bản trong chế độ bảo trì tích cực, gọi là stable (ổn định), testing (thử nghiệm) và unstable (không ổn định). Trong đó, các bản được phát hành chính thức mới nhất của Debian, được xem là bản ổn định và dùng cho môi trường sản xuất, hoạt động chính thức.

Ngoài ra Debian còn cung cấp phiên bản hỗ trợ dài hạn (LTS) nhằm kéo dài thời gian tồn tại của tất cả các bản phát hành ổn định Debian lên (ít nhất) 5 năm. Debian LTS không được xử lý bởi nhóm bảo mật của Debian mà bởi một nhóm tình nguyện viên và các công ty riêng biệt với mong muốn làm cho nó thành công. [6]



### 3.2. Hệ quản trị cơ sở dữ liệu MySQL/ MariaDB

MySQL được tạo ra bởi một công ty Thụy Điển, MySQL AB, được thành lập bởi người Thụy Điển David Axmark, Allan Larsson và Michael "Monty" Widenius người Phần Lan. Ban đầu MySQL được phát triển bởi Widenius và Axmark vào năm 1994. Nó được tạo ra để phục vụ mục đích sử dụng cá nhân từ với tên gọi là mSQP dựa trên ngôn ngữ cấp thấp ISAM. Tuy nhiên ISAM được cho là quá chậm và không linh hoạt, do đó các nhà sáng lập sau đó đã tạo một giao diện SQL mới trong khi vẫn giữ API giống như mSQL. Bằng cách giữ API nhất quán với hệ thống mSQL, nhiều nhà phát triển đã có thể sử dụng MySQL thay vì mSQL tiền thân (được cấp phép độc quyền).

MySQL nổi tiếng với khả năng xử lý nhanh chóng các truy vấn do đánh đổi việc bảo mật lấy hiệu năng, tính ổn định tốc độ cao, và khả năng mở rộng tốt. Được sử dụng rộng rãi trong các ứng dụng web và doanh nghiệp, MySQL cung cấp các tính năng mạnh mẽ như sao lưu, phục hồi, bảo mật, và quản lý người dùng.

MariaDB là một sản phẩm mã nguồn mở tách ra từ mã mở do cộng đồng phát triển của hệ quản trị cơ sở dữ liệu quan hệ MySQL nhằm theo hướng không phải trả phí với giấy phép phần mềm cộng đồng tự do (GNU GPL). MariaDB được phát triển từ sự dẫn dắt của những nhà phát triển ban đầu của MySQL, do lo ngại khi MySQL bị Oracle Corporation mua lại sẽ bị đóng mã cho một số plugins (các trình bổ trợ). MariaDB được thiết kế để thay thế trực tiếp MySQL mà không cần thay đổi mã nguồn của ứng dụng với mục tiêu duy trì tính tương thích hoàn toàn với MySQL trong khi cung cấp hiệu năng tốt hơn và thêm nhiều tính năng mới. Có thể nói đến chi tiết như câu lệnh khởi chạy dịch vụ MariaDB vẫn sử dụng cú pháp bắt đầu với từ khóa "mysql". [7]

### 3.3. Máy chủ dịch vụ web Apache2

Apache hay là chương trình máy chủ HTTP là một chương trình dành cho máy chủ đối thoại qua giao thức truyền tải siêu văn bản (HTTP) được phát hành theo các điều khoản giấy phép Apache2.0. Nó được phát triển và duy trì bởi một cộng đồng các nhà phát triển dưới sự bảo trợ của Quỹ phần mềm Apache. Apache đóng một vai trò quan trọng trong quá trình phát triển của mạng web thế giới (World Wide Web).

Khi được phát hành lần đầu, Apache là chương trình máy chủ mã nguồn mở duy nhất có khả năng cạnh tranh với chương trình máy chủ tương tự của Netscape Communications Corporation mà ngày nay được biết đến qua tên thương mại Sun Java System Web Server. Từ đó trở đi, Apache đã không ngừng tiến triển và trở thành một phần mềm có sức cạnh tranh mạnh so với các chương trình máy chủ khác về mặt hiệu suất và tính năng phong phú. Từ tháng 4 năm 1996, Apache trở thành một chương trình máy chủ HTTP thông dụng nhất. Hơn nữa, Apache thường được dùng để so sánh với các phần mềm khác có chức năng tương tự. Tính đến tháng 1 năm 2007 thì Apache chiếm đến 60% thị trường các chương trình phân phối trang web. Tính đến tháng 3 năm 2022, Netcraft ước tính rằng Apache phục vụ 23,04% trong số hàng triệu trang web bận rộn nhất.

Apache cung cấp các tính năng như xử lý các yêu cầu tĩnh (static) và động (dynamic), quản lý session, tăng tốc độ xử lý bằng caching và nhiều tính năng khác. Apache cũng được sử dụng rộng rãi để phục vụ các ứng dụng web phức tạp, bao gồm cả các ứng dụng được viết bằng các ngôn ngữ lập trình như PHP, Python và Java.

Apache được phát triển và duy trì bởi một cộng đồng mã nguồn mở dưới sự bảo trợ của Apache Software Foundation. Apache được phát hành với giấy phép Apache License và là một phần mềm tự do và miễn phí. [8]

### **3.4. Dịch vụ phân phối thông điệp RabbitMQ**

RabbitMQ là một dịch vụ phân phối thông điệp mã nguồn mở được phát triển dựa trên giao thức xếp hàng tin nhắn nâng cao (Advanced Message Queuing Protocol). AMQP là giao thức mạnh mẽ có cơ chế bảo mật cao, mỗi tin nhắn được gửi đều có tính năng tự mô tả chính nó nên luôn đảm bảo đầy đủ thông tin. RabbitMQ được biết đến với khả năng xử lý các tác vụ truyền thông tin giữa các hệ thống một cách hiệu quả, đáng tin cậy và dễ dàng quản lý. Với kiến trúc linh hoạt và khả năng mở rộng cao, RabbitMQ hỗ trợ nhiều giao thức nhắn tin, cung cấp tính năng đảm bảo thứ tự thông điệp, xác nhận thông điệp và độ bền của dữ liệu. Từ đó mở rộng với kiến trúc plugins để hỗ trợ giao thức nhắn tin định hướng văn bản trực tuyến (STOMP), truyền tải từ xa MQ (MQTT) và các giao thức khác.

Được phát triển ban đầu bởi Rabbit Technologies Ltd., khởi đầu là một liên doanh giữa LShift và CohesiveFT vào năm 2007, RabbitMQ đã được SpringSource, một bộ phận của VMware, mua lại vào tháng 4 năm 2010. Dự án đã trở thành một phần của Pivotal Software vào tháng 5 năm 2013. Sau đó đã được VMWare mua lại vào tháng 12 năm 2019.

Kể từ tháng 11 năm 2020, đã có thêm các dịch vụ thương mại của RabbitMQ dành cho các tính năng hỗ trợ và doanh nghiệp: "VMware RabbitMQ OVA", "VMware RabbitMQ" và "VMware RabbitMQ for Kubernetes" (các cấp tính năng khác nhau so với bản miễn phí). [9]

## **Chương 2. Cài đặt hệ thống Ironic trên hệ điều hành Debian**

### **1. Triển khai Ironic trên hệ điều hành Debian**

Khi triển khai Ironic, việc lựa chọn hệ điều hành phù hợp là một yếu tố tiên quyết, quan trọng để đảm bảo hệ thống hoạt động hiệu quả, ổn định, an toàn và tiết kiệm. Debian, với những đặc điểm nổi bật của mình, trở thành một trong những lựa chọn hàng đầu cho việc cài đặt Ironic:

- Trang chủ của Openstack cung cấp tài liệu chi tiết về cách cài đặt và sử dụng OpenStack, bao gồm cả Ironic trên hệ điều hành Debian.

- Không giống như Red Hat Enterprise Linux, một hệ điều hành hướng tới thị trường thương mại và có tính phí sử dụng. Debian là một hệ điều hành mã nguồn mở và miễn phí, giúp tối thiểu hóa chi phí dịch vụ kỹ thuật.

- Nổi tiếng với sự ổn định và tin cậy. Các phiên bản của Debian được kiểm tra kỹ lưỡng và duy trì lâu dài với các bản vá bảo mật và cập nhật định kỳ.

- Debian có một cộng đồng người dùng và nhà phát triển rộng lớn, sẵn sàng cung cấp hỗ trợ qua các diễn đàn và tài liệu phong phú.

- Debian chú trọng đến bảo mật, với các chính sách nghiêm ngặt về việc cập nhật và vá lỗi bảo mật.

- Hỗ trợ nhiều kiến trúc phần cứng nhất, từ các hệ thống x86, ARM đến các kiến trúc ít phổ biến hơn. Điều này làm cho Debian trở nên linh hoạt và phù hợp với nhiều loại phần cứng, giúp người dùng dễ dàng triển khai OpenStack Ironic trên các môi trường phần cứng đa dạng.

- Debian sử dụng hệ thống quản lý gói APT mạnh mẽ và hiệu quả, cho phép dễ dàng cài đặt, cập nhật và quản lý các gói phần mềm. Với kho phần mềm phong phú, Debian cung cấp hầu hết các phần mềm cần thiết cho việc triển khai và vận hành OpenStack Ironic.

## 2. Cài đặt hệ quản trị cơ sở dữ liệu MariaDB

Trước khi bắt đầu xây dựng hệ thống Ironic, cần cài đặt dịch vụ quản trị cơ sở dữ liệu trước để làm nền tảng cho các thành phần theo sau có thể phụ thuộc việc lưu trữ và truy xuất dữ liệu. Đảm bảo tính nhất quán và ổn định hiệu suất cho hệ thống, tránh xảy ra các lỗi khách quan trong quá trình cài đặt hệ thống.

Một hệ thống CSDL tốt không chỉ đảm bảo dữ liệu được tổ chức một cách hợp lý mà còn hỗ trợ khả năng mở rộng, xử lý các giao dịch nhanh chóng và đảm bảo tính toàn vẹn của dữ liệu.

Trong quá trình triển khai và vận hành hệ thống Ironic, việc lựa chọn hệ cơ sở dữ liệu phù hợp là vô cùng quan trọng để đảm bảo tính ổn định, hiệu năng và khả năng mở rộng của hệ thống. Ironic có khả năng kết nối với nhiều loại cơ sở dữ liệu khác nhau, trong đó MariaDB là lựa chọn tối ưu mang tính phù hợp cao với hệ thống với những lý do:

- Tích hợp tốt với OpenStack: MariaDB là hệ cơ sở dữ liệu mặc định được khuyến nghị sử dụng bởi OpenStack, bao gồm cả Ironic. Điều này có nghĩa là MariaDB đã được kiểm tra và tối ưu hóa để hoạt động tốt với OpenStack.

- Tính ổn định và độ tin cậy cao: MariaDB là một trong những hệ quản trị cơ sở dữ liệu được sử dụng rộng rãi nhất trên thế giới. Với lịch sử phát triển lâu dài và cộng đồng hỗ trợ mạnh mẽ.

- Hiệu năng cao và khả năng mở rộng tốt: Điều này phù hợp đối với hệ thống Ironic khi được xây dựng trong mạng cục bộ nơi có thể phụ thuộc việc bảo mật vào hệ thống tường lửa. Có thể vận hành các tác vụ quản lý và triển khai các máy chủ vật lý đòi hỏi sự phản hồi nhanh chóng và hiệu quả. Hơn nữa, MariaDB có khả năng mở rộng tốt, cho phép hệ thống Ironic phát triển mà không gặp phải các vấn đề về hiệu năng.

- Cộng đồng hỗ trợ mạnh mẽ : Cộng đồng người dùng và phát triển MariaDB rất lớn và năng động, cung cấp một lượng lớn tài liệu, diễn đàn, và các nguồn tài nguyên hỗ trợ.

\* Chi tiết cài đặt có thể tham khảo Phụ lục, mục 1.

### 3. Cài đặt dịch vụ xác thực Openstack Keystone

Là một trong những thành phần không thể thiếu của hệ thống Ironic, việc cài đặt OpenStack Keystone theo sau đó là rất quan trọng để có thể phục vụ việc xác thực danh tính khi truy cập và sử dụng các dịch vụ của hệ thống. Tránh xảy ra các lỗi xác thực và cấu hình trong quá trình cài đặt khi yêu cầu được gửi đến. Chi tiết có thể đề cập đến như:

- Tích hợp liên mạch với các thành phần OpenStack khác: Keystone là thành phần xác thực trung tâm của OpenStack, tương tác với tất cả các dịch vụ khác như Nova, Glance, Neutron và Ironic. Việc sử dụng Keystone trong Ironic giúp đảm bảo sự nhất quán và liên mạch trong việc xác thực và quản lý quyền truy cập trên toàn bộ hệ thống OpenStack.

- Khuyến nghị và hỗ trợ từ cộng đồng OpenStack: Keystone là thành phần được khuyến nghị sử dụng bởi cộng đồng OpenStack và nhận được sự hỗ trợ mạnh mẽ từ cộng đồng này. Điều này đảm bảo rằng người dùng sẽ luôn có sự hỗ trợ và các tài liệu hướng dẫn chi tiết khi triển khai và sử dụng Keystone trong hệ thống Ironic. Hơn nữa, các tài liệu trên trang chủ OpenStack luôn luôn tham chiếu đến việc cài đặt và sử dụng Keystone.

- Bảo mật cao: Keystone cung cấp các cơ chế xác thực mạnh mẽ và đa dạng, bao gồm xác thực bằng tên người dùng/mật khẩu, token và hỗ trợ các giao thức xác thực bên thứ ba như LDAP, Kerberos. Điều này đảm bảo rằng chỉ những người dùng hợp lệ mới có thể truy cập vào hệ thống, tăng cường bảo mật cho các tài nguyên vật lý. Đặc biệt khi sử dụng dịch vụ cơ sở dữ liệu MariaDB với điểm yếu về việc bảo mật.

- Quản lý người dùng và quyền hạn: Keystone cho phép quản lý người dùng và quyền hạn một cách hiệu quả. Người dùng có thể dễ dàng tạo, sửa đổi, và xóa người dùng, cũng như phân quyền truy cập tới các dịch vụ và tài nguyên khác nhau trong hệ thống Ironic. Điều này giúp kiểm soát chặt chẽ quyền truy cập và đảm bảo rằng mỗi người dùng chỉ có thể thực hiện những tác vụ mà họ được phép.

- Đăng nhập một lần (Single Sign-On - SSO): Keystone hỗ trợ cơ chế đăng nhập một lần, giúp người dùng chỉ cần đăng nhập một lần để truy cập tất cả các dịch vụ trong hệ thống OpenStack. Điều này cải thiện trải nghiệm người dùng và tăng hiệu suất làm việc, đồng thời giảm thiểu rủi ro liên quan đến việc quản lý nhiều tài khoản và mật khẩu.

- Tính mở rộng và khả năng tùy biến cao: Keystone được thiết kế với khả năng mở rộng và tùy biến cao, phù hợp với các hệ thống từ nhỏ đến lớn. Người dùng có thể dễ dàng tích hợp Keystone với các hệ thống xác thực hiện có của doanh nghiệp, đồng thời mở rộng khả năng xác thực khi nhu cầu tăng lên.

- Ghi nhật ký và theo dõi: Keystone cung cấp các công cụ ghi nhật ký và theo dõi hoạt động xác thực, giúp người dùng giám sát và phân tích các hành động truy cập trong hệ thống. Điều này rất hữu ích trong việc phát hiện và ngăn chặn các hành vi truy cập trái phép, đảm bảo an ninh cho toàn bộ hệ thống Ironic.

\* Chi tiết cài đặt có thể tham khảo Phụ lục, mục 4.

#### 4. Cài đặt dịch vụ phân phối thông điệp RabbimtMQ

Trong các hệ thống phần mềm hiện đại, bên cạnh việc chọn lựa cơ sở dữ liệu phù hợp, việc sử dụng dịch vụ phân phối thông điệp (message broker) cũng không kém phần quan trọng. Message broker chịu trách nhiệm trung gian trong việc truyền tải thông điệp giữa các dịch vụ và ứng dụng, giúp các thành phần trong hệ thống giao tiếp với nhau một cách hiệu quả, đáng tin cậy.

Trong quá trình triển khai và vận hành các hệ thống như OpenStack, việc lựa chọn message broker phù hợp là vô cùng quan trọng để đảm bảo tính ổn định và hiệu quả trong giao tiếp giữa các dịch vụ. Và RabbitMQ là một lựa chọn tối ưu với nhiều lý do:

- RabbitMQ được chính thức khuyến nghị sử dụng bởi OpenStack do khả năng tương thích cao và hiệu suất vượt trội. Nhiều thành phần cốt lõi của OpenStack, bao gồm Nova, Neutron và Cinder, đã được thử nghiệm và xác nhận hoạt động ổn định với RabbitMQ.

- Hiệu năng cao và khả năng mở rộng tốt: RabbitMQ được thiết kế để xử lý hàng triệu thông điệp mỗi giây, với khả năng mở rộng dễ dàng để đáp ứng nhu cầu ngày càng tăng của hệ thống.

- Độ tin cậy và tính bền vững: RabbitMQ đảm bảo độ tin cậy cao với các tính năng như xác nhận thông điệp (message acknowledgements), lưu trữ thông điệp trên đĩa (persistent messages), và hỗ trợ cơ chế khôi phục (high availability). Đảm bảo rằng không có thông điệp nào bị mất mát trong quá trình truyền tải, ngay cả khi xảy ra sự cố hệ thống.

- Tính linh hoạt và đa dạng: RabbitMQ hỗ trợ nhiều giao thức nhắn tin và mô hình truyền thông khác nhau như publish/subscribe, point-to-point, và routing, đáp ứng nhu cầu đa dạng của các ứng dụng và dịch vụ trong hệ thống.

- Quản lý và giám sát dễ dàng: RabbitMQ đi kèm với giao diện quản lý trực quan và các công cụ giám sát mạnh mẽ, cho phép quản trị viên dễ dàng theo dõi và quản lý các hàng đợi, kết nối và thông điệp.

\* Chi tiết cài đặt có thể tham khảo Phụ lục, mục 3.



## 5. Cài đặt máy chủ web Apache2

Tiếp đến, việc cài đặt và triển khai máy chủ web là bước quan trọng để có thể triển khai các dịch vụ web cần thiết cho các thành phần khác của hệ thống. Một máy chủ web đáng tin cậy như Apache 2 sẽ đảm bảo rằng các dịch vụ được đẩy lên có thể hoạt động một cách hiệu quả và ổn định.

Việc lựa chọn máy chủ web phù hợp để sử dụng cho hệ thống Ironic mang lại nhiều lợi ích quan trọng, đảm bảo tính hiệu quả và ổn định cho hệ thống. Apache2 là một trong những máy chủ web đáng lựa chọn với nhiều tính năng ưu việt:

- Hỗ Trợ WSGI: Apache2 hỗ trợ tốt cho WSGI (Web Server Gateway Interface), một tiêu chuẩn giao tiếp giữa máy chủ web và các ứng dụng Python. Ironic, như một thành phần của OpenStack, thường sử dụng WSGI để triển khai các dịch vụ web. Apache2 kết hợp với mô-đun `mod_wsgi` cho phép triển khai các ứng dụng WSGI một cách mượt mà và hiệu quả.

- Được hỗ trợ chi tiết bởi OpenStack: Trên trang chủ của OpenStack, có sẵn các tài liệu chi tiết hướng dẫn cài đặt và cấu hình Ironic với Apache2. Giảm thiểu thời gian cấu hình và đảm bảo triển khai đúng cách theo các khuyến nghị và best practices.

- Bảo Mật Cao: Với khả năng cập nhật bảo mật thường xuyên và hỗ trợ SSL/TLS, Apache2 giúp bảo vệ các dịch vụ web khỏi các mối đe dọa tiềm ẩn.

- Tính ổn định và hiệu năng cao: Apache2 đã được chứng minh qua nhiều năm sử dụng trong các môi trường sản xuất với khả năng xử lý lượng lớn yêu cầu một cách ổn định.

- Khả năng tùy biến cao: Apache2 hỗ trợ nhiều mô-đun mở rộng, cho phép tùy chỉnh và tối ưu hóa theo nhu cầu cụ thể của hệ thống.

- Cộng đồng mạnh mẽ: Apache2 có một cộng đồng sử dụng và phát triển rộng lớn, cung cấp tài liệu phong phú cũng như hỗ trợ kỹ thuật nhanh chóng, giúp giải quyết các vấn đề một cách nhanh chóng và hiệu quả.

- \* Chi tiết cài đặt có thể tham khảo Phụ lục, mục 2.

## 6. Cài đặt thành phần chính trong Ironic

### 6.1. Các nền tảng công nghệ ảo hóa

Bảng so sánh các công nghệ ảo hóa

Tiêu chí	Virtual Machine (KVM, Xen)	Container (Docker, LXC)	Physicalization (Ironic)
Định nghĩa	Ao hóa toàn bộ hệ điều hành và phần cứng ảo.	Ao hóa tại cấp độ hệ điều hành, chia sẻ các tài nguyên với máy chủ.	Quản lý và kiến trúc các máy chủ vật lý.
Kiến trúc	Được phân ra 2 loại ảo hóa loại 1 và loại 2.	Sử dụng container để đóng gói và phân phối ứng dụng, có môi trường người dùng riêng biệt.	Các máy chủ vật lý được quản lý và cấp phát trực tiếp mà không có lớp ảo hóa trung gian.
Tính chất	Phù hợp cho các môi trường ảo hóa phức tạp.	Chỉ bao gồm ứng dụng của người dùng. Không thể can thiệp các thành phần khác.	Hiệu suất cao. Xử lý công việc không thể ảo hóa.
Tài nguyên	Quản lý tài nguyên mạnh mẽ nhưng phân bổ nhiều cho Hypervisor.	Quản lý tài nguyên kém so với VM do chia sẻ tài nguyên trực tiếp với máy chủ.	Sử dụng tài nguyên vật lý nhưng đòi hỏi quản lý chi tiết và phức tạp hơn.
Hiệu suất	Thấp nhất do có overhead cao, cần tài nguyên cho Hypervisor.	Cao hơn so với Virtual Machine.	Cao nhất (không Hypervisor)
Bảo mật	Cao (mỗi VM là một máy ảo riêng).	Trung bình (1 lỗ hổng có thể ảnh hưởng đến toàn bộ containers).	Cao nhất (không Hypervisor)
Khả năng mở rộng	Mở rộng tốt nhưng chậm.	Mở rộng nhanh chóng và linh hoạt.	Linh hoạt nhưng khó khăn hơn trong việc triển khai.

Hình 2.1: Bảng so sánh các công nghệ ảo hóa

Có thể thấy, Ironic có nhiều ưu điểm vượt trội so với các công nghệ ảo hóa khác. Tùy vào nhu cầu sử dụng và triển khai, người dùng có thể lựa chọn công nghệ ảo hóa phù hợp. [10]

Ngoài ra, trong bảng 2.1 có đề cập đến các công nghệ ảo hóa loại 1 và loại 2. Trong đó, công nghệ ảo hóa loại 1 sử dụng dịch vụ bare metal để triển khai máy ảo. Khi so sánh với hệ thống Ironic cũng sử dụng dịch vụ bare metal, người dùng có thể thấy nhiều điểm khác biệt:

Bảng so sánh các công nghệ ảo hóa loại 1 và 2 với Ironic

<b>Tiêu chí</b>	<b>Ảo hóa loại 1</b>	<b>Ảo hóa loại 2</b>	<b>Ironic</b>
Định nghĩa	Ảo hóa loại 1 (KVM, XEN) chạy trực tiếp trên phần cứng, không có hệ điều hành lưu trữ.	Ảo hóa loại 2 (VMware, VirtualBox) chạy trên một hệ điều hành lưu trữ đã được cài đặt.	Quản lý và kiến trúc các máy chủ vật lý trong môi trường đám mây.
Giao diện	Có giao diện quản lý phần cứng ảo hóa.	Có giao diện quản lý phần mềm ảo hóa.	Ironic có giao diện quản lý máy chủ vật
Hiệu suất	Có hiệu suất cao hơn ảo hóa loại 2 do không có lớp Hypervisor trung gian.	Thấp nhất do có lớp Hypervisor trung gian.	Hiệu suất cao. Xử lý công việc không thể ảo hóa.
Độ phức tạp	Phức tạp trong việc cài đặt và quản lý.	Đơn giản hơn trong việc cài đặt và quản lý	Phức tạp trong việc cài đặt và quản lý trong Cloud.
Khả năng di động	Di động giữa các nền tảng ảo hóa loại 1 khác nhau.	Di động giữa các nền tảng ảo hóa loại 2 khác nhau.	Không di động giữa các môi trường đám mây khác nhau.

Hình 2.2: Bảng so sánh các công nghệ ảo hóa loại 1 và 2 với dịch vụ Ironic

## 6.2. Cấu hình các thành phần chính trong Ironic

Các thành phần chính cần cài đặt khi triển khai Ironic bao gồm:

- Ironic-api và Ironic conductor: 2 thành phần không thể thiếu trong việc giao tiếp và xử lý hệ thống.

- PXE (Preboot Execution Environment): Là cơ chế cho phép khởi động một phần mềm lấy qua mạng. Trong ngữ cảnh server vật lý để có thể khởi động PXE thì phía máy client cần NIC card mạng hỗ trợ PXE và hệ thống mạng có máy chủ DHCP và TFTP để cấp IP và phân bổ các image hệ điều hành xuống client. [11]

- DHCP (Dynamic Host Configuration Protocol): Là giao thức cấu hình máy chủ). DHCP có nhiệm vụ giúp quản lý nhanh, tự động và tập trung việc phân phối địa chỉ IP bên trong một mạng. Ngoài ra DHCP còn giúp đưa thông tin đến các thiết bị hợp lý hơn cũng như việc cấu hình subnet mask hay cổng mặc định. [12]

- TFTP (Trivial File Transfer Protocol): Là một công nghệ chuyển các file giữa các thiết bị mạng và là phiên bản đơn giản hóa của giao thức FTP. [13]

- HTTP (Hyper Text Transfer Protocol): Là giao thức truyền tải siêu văn bản, cho phép tìm nạp tài nguyên. [14]

- IPMI (Intelligent Platform Management Interface): Là giao diện máy tính cho phép người dùng tự do quản lý và giám sát máy chủ từ xa. [15]

Tóm lại, dịch vụ Ironic quản lý phần cứng thông qua các giao thức quản lý từ xa phổ biến và riêng biệt (ví dụ như: PXE và IPMI). Nó cung cấp cho người sử dụng một giao diện nhất quán ngay cả với một nhóm máy chủ không đồng nhất. Qua đó, cung cấp và cho phép quản lý các máy chủ vật lý như thể chúng là máy ảo.

## **Chương 3. Xây dựng hệ thống tính toán song song**

### **1. Khái niệm tính toán song song**

Tính toán song song (Parallel computing), là một hình thức tính toán trong đó nhiều phép tính và tiến trình được thực hiện đồng thời, hoạt động trên nguyên tắc là những vấn đề lớn đều có thể chia thành nhiều phần nhỏ hơn, sau đó được giải quyết tương tranh (race condition).

Có nhiều hình thức khác nhau của tính toán song song: song song cấp bit, song song cấp lệnh, song song dữ liệu, và song song tác vụ. Xử lý song song đã được sử dụng từ nhiều năm qua, chủ yếu là trong lĩnh vực tính toán hiệu năng cao. Gần đây hình thức tính toán này được quan tâm nhiều hơn, do những hạn chế vật lý ngăn chặn việc tăng hiệu năng tính toán chỉ bằng cách tăng tần số. Vì việc tiêu hao điện năng (dẫn đến sinh nhiệt) từ máy tính đã trở thành một mối lo ngại trong những năm gần đây, tính toán song song đã trở thành mô hình thống trị trong lĩnh vực kiến trúc máy tính, phần lớn là dưới dạng bộ xử lý đa nhân.

Các máy tính song song có thể được phân loại tùy theo cấp độ hỗ trợ cho song song của phần cứng, với những chiếc máy tính đa nhân và đa xử lý có bộ phận đa xử lý trong một máy đơn lẻ, trong khi cụm máy tính, xử lý song song hàng loạt, và điện toán lưới sử dụng nhiều máy tính để xử lý cùng một công việc. Những kiến trúc máy tính song song chuyên dụng thỉnh thoảng cũng sử dụng các bộ xử lý truyền thống nhằm tăng tốc độ cho những công việc đặc trưng. [16]

## 2. Xây dựng bài toán tính toán song song

Sau khi đã cài đặt và cấu hình thành công dịch vụ Ironic. Người dùng có thể thiết lập hệ thống các máy chủ vật lý phục vụ hệ thống tính toán song song.

Bước 1: Tạo máy chủ vật lý: \*Chi tiết triển khai tham khảo Phụ lục, mục 9.

Bước 2: Cài đặt và cấu hình các công cụ phục vụ công việc tính toán song song khác:

- Ngôn ngữ lập trình Python: Là một ngôn ngữ lập trình phổ biến, mạnh mẽ, cung cấp rất nhiều thư viện toán học như numpy,... dễ dàng tham gia vào việc xây dựng hệ thống tính toán song song.

- MPI (Message Passing Interface): Là một tiêu chuẩn giao tiếp được thiết kế để hỗ trợ việc lập trình song song trên các hệ thống máy tính phân tán. Nó cung cấp các giao thức và chức năng để truyền thông tin giữa các tiến trình khác nhau, thường chạy trên các nút khác nhau trong một cụm máy tính (cluster).

- MPI được thiết kế để: Cung cấp chức năng truyền tải thông điệp giữa các tiến trình. Hỗ trợ các mô hình lập trình song song. Tương thích và hoạt động trên nhiều hệ điều hành và kiến trúc phần cứng khác nhau.

- Thư viện mpi4py: Là một gói thư viện cho Python cung cấp các ràng buộc Python cho chuẩn MPI. Điều này có nghĩa là người dùng có thể sử dụng MPI từ ngôn ngữ lập trình Python thông qua mpi4py. Nó giúp tận dụng các khả năng của MPI một cách dễ dàng hơn mà không cần viết mã C hoặc Fortran.

- Bài toán thực hiện: Sử dụng ngôn ngữ Python viết chương trình nhân 2 ma trận với kích thước phù hợp, giúp đảm bảo không gây quá tải tài nguyên đã cung cấp cho các máy tính toán.



Thời gian tính toán khi thiết lập 2 máy chạy song song:

```

root@ironic:~# mpirun -np 2 --host 192.168.149.137,
Thời gian tính toán của từng tiến trình: 7.33 giây
Ma trận đã được tính xong!
Thời gian tính toán: 8.46 giây
root@ironic:~# mpirun -np 2 --host 192.168.149.137,
Thời gian tính toán của từng tiến trình: 6.71 giây
Ma trận đã được tính xong!
Thời gian tính toán: 8.16 giây
root@ironic:~# mpirun -np 2 --host 192.168.149.137,
Thời gian tính toán của từng tiến trình: 3.45 giây
Ma trận đã được tính xong!
Thời gian tính toán: 3.97 giây

```

Hình 3.3. Thời gian tính toán khi thiết lập  
2 máy chạy song song.

Tài nguyên sử dụng khi thực hiện tính toán song song:

CPU[                                     100.0%]											Tasks: 24, 59 thr, 95 k	
Mem[                                 163M/926M]											Load average: 0.63 0.41	
Swp[     32.2M/975M]											Uptime: 00:09:13	
Main		I/O										
PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command	
1038	root	20	0	246M	43216	22532	R	97.4	4.6	0:01.62	python3 /tinhtoansongs	
979	root	20	0	7964	3996	3176	R	0.6	0.4	0:01.91	htop	
1	root	20	0	99M	5668	4184	S	0.0	0.6	0:00.91	/sbin/init	
246	root	20	0	32964	6664	6340	S	0.0	0.7	0:00.33	/lib/systemd/systemd-j	
266	root	20	0	26028	416	368	S	0.0	0.0	0:00.15	/lib/systemd/systemd-u	
309	systemd-ti	20	0	90056	1656	1604	S	0.0	0.2	0:00.06	/lib/systemd/systemd-t	
CPU[                                     100.0%]											Tasks: 23, 59 thr, 96 kthr;	
Mem[                                 556M/926M]											Load average: 0.98 0.48 0.22	
Swp[     211M/975M]											Uptime: 00:10:32	
Main		I/O										
PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command	
1079	root	20	0	850M	412M	8080	R	99.3	44.6	0:05.35	python3 /tinhtoansongsong.s	
924	root	20	0	17976	192	16	S	0.7	0.0	0:00.32	sshd: root@pts/0	
1008	root	20	0	7964	1612	832	R	0.7	0.2	0:01.52	htop	
1	root	20	0	99M	2236	916	S	0.0	0.2	0:00.96	/sbin/init	

Hình 3.4. Tài nguyên sử dụng khi thực hiện  
tính toán song song.



Qua so sánh, có thể thấy tốc độ tính toán trung bình khi sử dụng một máy tính toán đơn rơi vào khoảng 17,91 giây sau 3 lần thử nghiệm. Trong khi đó, tốc độ tính toán trung bình của hệ thống 2 máy chạy song song sẽ là 6.86 giây và 6,03 cho thời gian tính toán trung bình của từng tiến trình.

Tài nguyên được sử dụng trong khi thực hiện tính toán song song luôn cao hơn so với khi thực hiện tính toán đơn lẻ. Điều này chứng tỏ việc hệ thống tính toán song song luôn tận dụng hiệu quả các tài nguyên đang trong quá trình không được sử dụng để tối ưu hóa quá trình tính toán và xử lý công việc.

Ngoài ra, tốc độ tính toán trên hệ thống song song có vẻ như được cải thiện qua mỗi lần chạy. Qua đó cho thấy sự linh hoạt và mạnh mẽ trong việc áp dụng tính toán song song thay cho tính toán đơn lẻ.

Tóm lại có thể thấy, việc áp dụng tính toán song song mang lại nhiều lợi ích so với tính toán đơn (tính toán tuần tự), đặc biệt khi mở rộng quy mô trong việc xử lý các tác vụ phức tạp và lớn. Chi tiết có thể nói đến như:

- Tăng tốc độ xử lý:

- + Phân chia công việc: Trong tính toán song song, một tác vụ lớn được chia thành nhiều tác vụ nhỏ hơn, có thể được thực hiện đồng thời trên nhiều bộ xử lý hoặc máy tính khác nhau.

- + Thời gian xử lý ngắn hơn: Thời gian tổng cộng để hoàn thành công việc giảm đáng kể, đặc biệt là với các tác vụ có thể được chia thành các phần độc lập.

- Sử dụng tài nguyên một cách hiệu quả:

- + Tận dụng tài nguyên nhàn rỗi: Tính toán song song giúp tận dụng tối đa các tài nguyên nhàn rỗi, như nhiều lõi CPU hoặc nhiều máy tính trong một cụm.

- + Tăng cường hiệu suất: Sử dụng nhiều tài nguyên đồng thời giúp cải thiện hiệu suất tổng thể của hệ thống.

- Khả Năng Mở Rộng:

- + Tăng khả năng xử lý: Khi khối lượng công việc tăng, có thể thêm nhiều tài nguyên (như thêm máy tính hoặc lõi CPU) để duy trì hoặc tăng tốc độ xử lý.

+ Mở rộng dễ dàng: Các hệ thống tính toán song song có thể được mở rộng một cách linh hoạt để đáp ứng nhu cầu ngày càng tăng.

- Xử lý các tác vụ lớn và phức tạp:

+ Giải quyết vấn đề lớn hơn: Tính toán song song cho phép giải quyết các vấn đề mà tính toán tuần tự không thể xử lý do giới hạn về thời gian hoặc tài nguyên.

+ Ứng dụng trong nhiều lĩnh vực: Đặc biệt hữu ích trong các lĩnh vực như mô phỏng khoa học, học máy, phân tích dữ liệu lớn, và mô hình hóa tài chính.

- Tính tin cậy và khả năng chịu lỗi:

+ Độ tin cậy cao hơn: Hệ thống tính toán song song có thể được thiết kế để chịu lỗi tốt hơn, vì lỗi của một phần hệ thống không làm ảnh hưởng toàn bộ hệ thống.

+ Phục hồi nhanh chóng: Hệ thống có thể tự phục hồi từ các lỗi bằng cách tái phân chia và tiếp tục công việc từ điểm bị gián đoạn.

- Tăng Cường Độ Chính Xác và Chi Tiết:

+ Phân tích chi tiết hơn: Cho phép thực hiện các phân tích chi tiết hơn và chính xác hơn do khả năng xử lý lượng dữ liệu lớn trong thời gian ngắn.

+ Mô phỏng chính xác: Đặc biệt hữu ích trong các mô phỏng yêu cầu độ chính xác cao như trong nghiên cứu khoa học và kỹ thuật.

- Tiết Kiệm Chi Phí Dài Hạn:

+ Hiệu quả năng lượng: Sử dụng nhiều tài nguyên đồng thời có thể tiết kiệm năng lượng hơn so với việc sử dụng một tài nguyên liên tục trong thời gian dài.

+ Giảm thời gian phát triển: Với khả năng xử lý nhanh hơn, thời gian phát triển và thử nghiệm ứng dụng giảm, dẫn đến tiết kiệm chi phí.

- Khả Năng Thực Hiện Các Tác Vụ Đồng Thời:

+ Thực hiện nhiều tác vụ cùng lúc: Cho phép thực hiện nhiều tác vụ đồng thời, tăng hiệu quả xử lý và tiết kiệm thời gian.

+ Đáp ứng tốt hơn: Cải thiện khả năng đáp ứng của hệ thống đối với các tác vụ thời gian thực và yêu cầu cao về tốc độ xử lý.

## Kết luận

Qua bài viết trên, người đọc đã được tìm hiểu về việc xây dựng hệ thống tính toán song song trên hệ điều hành Debian, sử dụng kiến trúc OpenStack Baremetal Ironic. Qua đó, có thể thấy rằng Ironic đóng vai trò vô cùng quan trọng và không thể thiếu trong việc triển khai và quản lý các tài nguyên phần cứng vật lý một cách hiệu quả và linh hoạt.

Ironic giúp quản lý và tự động hóa việc triển khai các máy chủ vật lý, biến chúng thành các tài nguyên điện toán mạnh mẽ và có khả năng mở rộng. Điều này là nền tảng cho việc xây dựng các hệ thống tính toán song song hiệu quả, giúp tối ưu hóa hiệu suất và tận dụng tối đa tài nguyên phần cứng sẵn có. Với Ironic, việc triển khai các hệ thống tính toán phức tạp trở nên dễ dàng hơn nhờ khả năng tự động hóa cao, giúp giảm thiểu thời gian và công sức quản trị.

Hơn nữa, việc sử dụng Debian làm hệ điều hành cho hệ thống tính toán song song mang lại nhiều lợi ích như tính ổn định, bảo mật cao, và sự hỗ trợ lâu dài từ cộng đồng mã nguồn mở. Debian không chỉ tương thích tốt với đa dạng phần cứng mà còn là một nền tảng lý tưởng để triển khai các dịch vụ và ứng dụng đòi hỏi hiệu suất cao.

Tuy nhiên, Ironic cũng có một số hạn chế cần lưu ý:

- Yêu cầu phần cứng mạnh: Để triển khai Ironic, hệ thống cần có phần cứng vật lý phù hợp và được hỗ trợ. Điều này có thể đòi hỏi đầu tư lớn vào cơ sở hạ tầng.

- Khó khăn trong việc cài đặt và sử dụng: Việc cài đặt và cấu hình Ironic khá phức tạp, yêu cầu kiến thức chuyên sâu về OpenStack và các thành phần liên quan.

- Yêu cầu chuyên môn cao: Quản trị và vận hành Ironic đòi hỏi đội ngũ kỹ thuật có chuyên môn cao, hiểu rõ về cả phần cứng và phần mềm.

- Đôi khi gặp hạn chế về khả năng mở rộng: Mặc dù Ironic giúp quản lý tài nguyên vật lý hiệu quả, việc mở rộng quy mô hệ thống vẫn gặp khó khăn do phụ thuộc vào phần cứng cụ thể.

Tóm lại, sự kết hợp giữa Debian và OpenStack Baremetal Ironic tạo nên một giải pháp mạnh mẽ và hiệu quả cho việc xây dựng các hệ thống tính toán song song. Ironic đóng vai trò trung tâm trong việc quản lý và phân phối tài nguyên phần cứng, giúp hệ thống hoạt động ổn định và tối ưu, đồng thời cung cấp khả năng mở rộng linh hoạt cho các nhu cầu tính toán ngày càng phức tạp và đa dạng.

Mặc dù có một số hạn chế như yêu cầu phần cứng cụ thể, khó khăn trong việc cài đặt và vận hành, và cần chuyên môn cao, nhưng những lợi ích mà Ironic mang lại vẫn rất đáng kể. Việc hiểu rõ và áp dụng các công nghệ này không chỉ giúp tối ưu hóa nguồn lực mà còn tạo ra những bước tiến quan trọng trong việc phát triển và triển khai các hệ thống tính toán hiện đại. Sự kết hợp giữa tính ổn định và bảo mật của Debian với khả năng quản lý tài nguyên hiệu quả của Ironic chắc chắn sẽ là nền tảng vững chắc cho bất kỳ hệ thống tính toán song song nào.

## DANH MỤC TÀI LIỆU THAM KHẢO

- [1] Wikipedia, "OpenStack History," [Online]. Available:  
<https://en.wikipedia.org/wiki/OpenStack#>.
- [2] Wikipedia, "OpenStack & Ironic," [Online]. Available:  
<https://en.wikipedia.org/wiki/OpenStack#>.
- [3] O. Ironic, "Ironic Bare Metal service components," [Online]. Available:  
[https://docs.openstack.org/ironic/latest/install/get\\_started.html](https://docs.openstack.org/ironic/latest/install/get_started.html).
- [4] Wikipedia, "OpenStack Components," [Online]. Available:  
<https://en.wikipedia.org/wiki/OpenStack#Components>.
- [5] OpenStack, "Ironic Bare Metal service overview," [Online].  
Available:  
[https://docs.openstack.org/ironic/latest/install/get\\_started.html#Logical%20architecture](https://docs.openstack.org/ironic/latest/install/get_started.html#Logical%20architecture).
- [6] Wikipedia, "Debian Wikipedia," [Online]. Available:  
<https://en.wikipedia.org/wiki/Debian>.
- [7] Wikipedia, "MariaDB Wikipedia," [Online]. Available:  
<https://en.wikipedia.org/wiki/MariaDB>.
- [8] Wikipedia, "Apache HTTP Server," [Online]. Available:  
[https://en.wikipedia.org/wiki/Apache\\_HTTP\\_Server](https://en.wikipedia.org/wiki/Apache_HTTP_Server).
- [9] Wikipedia, "RabbitMQ Wikipedia," [Online]. Available:  
<https://en.wikipedia.org/wiki/RabbitMQ>.
- [10] A. W. Service, "What's the Difference Between Type 1 and Type 2 Hypervisors?," [Online]. Available:  
[https://aws.amazon.com/compare/the-difference-between-type-1-and-type-2-hypervisors/?nc1=h\\_ls](https://aws.amazon.com/compare/the-difference-between-type-1-and-type-2-hypervisors/?nc1=h_ls).

- [11] Cloud365, "Cloud365," [Online]. Available: <https://news.cloud365.vn/cobbler-tong-quan-ve-pxe-preboot-execution-environment/>.
- [12] totolink, "totolink," [Online]. Available: <https://www.totolink.vn/article/111-dhcp-la-gi-tim-hieu-ve-dhcp.html>
- [13] vietnix, "vietnix," [Online]. Available: <https://vietnix.vn/tftp-la-gi/>.
- [14] topdev, "topdev," [Online]. Available: <https://topdev.vn/blog/http-la-gi/>.
- [15] 123host, "123host.vn," [Online]. Available: <https://123host.vn/kien-thuc/giao-dien-ipmi-la-gi.html>.
- [16] Wikipedia, "TÍNH toán song song Wikipedia," [Online]. Available: [https://vi.wikipedia.org/wiki/T%C3%ADnh\\_to%C3%A1n\\_song\\_song](https://vi.wikipedia.org/wiki/T%C3%ADnh_to%C3%A1n_song_song).

## Phụ lục

### 1. Cài đặt MariaDB

#### 1.1 Hướng dẫn cài đặt

Bước 1: Cài đặt MariaDB

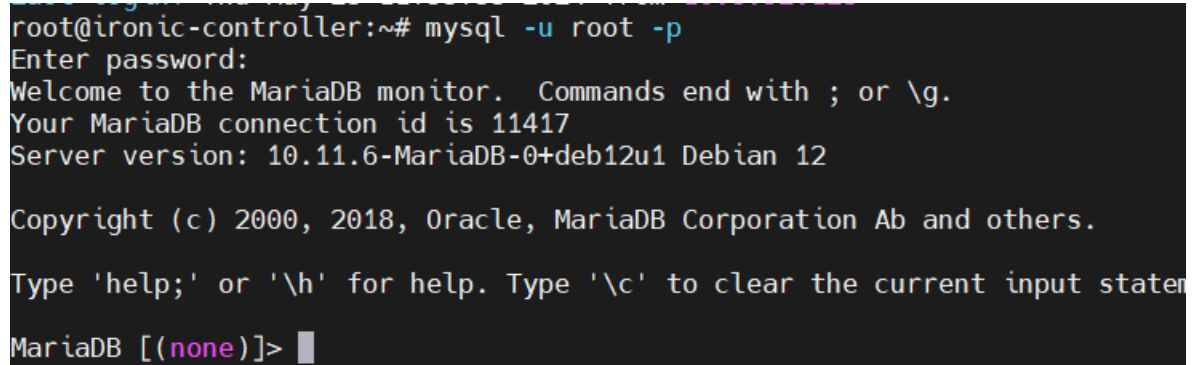
```
$ sudo apt-get install mariadb-server mariadb-client
```

Bước 2: Bảo mật quá trình cài đặt MariaDB

```
$ mysql_secure_installation
```

Bước 3: Kiểm tra dịch vụ MariaDB đã cài đặt thành công

```
$ mysql -u root -p
```



```
root@ironic-controller:~# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 11417
Server version: 10.11.6-MariaDB-0+deb12u1 Debian 12

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

Hình 1.1. Dịch vụ MariaDB đã cài đặt thành công

#### 1.2. Lỗi khi cài đặt MySQL/MariaDB

##### 1.2.1. Unable to locate package.

Nguyên nhân: Linux không tìm thấy package muốn cài đặt.

Cách xử lý: Kiểm tra và cài đặt package có tên “mariadb-server” và “mariadb-client”.

```
$ sudo apt-get install mariadb-server maria-client.
```

### 1.2.2. Port 3306 already in use.

Nguyên nhân: Cổng mặc định của MySQL (3306) đã được sử dụng bởi tiến trình khác.

Cách xử lý: Kiểm tra và đổi lại cổng hoạt động cho dịch vụ MySQL.

```
$ sudo nano /etc/mysql/my.cnf
```

Thay đổi “bind-address” bằng cổng khác chưa được sử dụng.

```
$ sudo service mysqld restart
```

## 2. Cài đặt Apache2

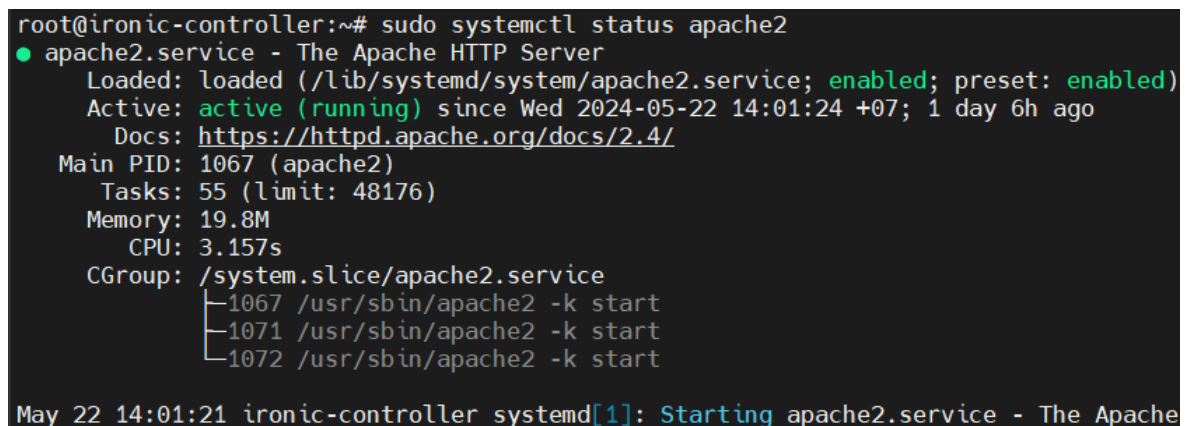
### 2.1 Hướng dẫn cài đặt

Bước 1: Cài đặt Apache2

```
$ sudo apt-get install apache2
```

Bước 2: Kiểm tra dịch vụ MariaDB đã cài đặt thành công

```
$ sudo systemctl status apache2
```



```
root@ironic-controller:~# sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; preset: enabled)
   Active: active (running) since Wed 2024-05-22 14:01:24 +07; 1 day 6h ago
     Docs: https://httpd.apache.org/docs/2.4/
   Main PID: 1067 (apache2)
     Tasks: 55 (limit: 48176)
    Memory: 19.8M
       CPU: 3.157s
    CGroup: /system.slice/apache2.service
            └─1067 /usr/sbin/apache2 -k start
              └─1071 /usr/sbin/apache2 -k start
                └─1072 /usr/sbin/apache2 -k start

May 22 14:01:21 ironic-controller systemd[1]: Starting apache2.service - The Apache
```

Hình 2.1. Dịch vụ Apache2 đã cài đặt thành công.



## 2.2. Lỗi khi cài đặt Apache2

### 2.2.1. Port 80 or 443 already in use.

Nguyên nhân: Cổng mặc định của Apache2 đã được sử dụng bởi tiến trình khác.

Cách xử lý: Kiểm tra và đổi lại cổng hoạt động cho dịch vụ MySQL.

```
$ sudo nano /etc/apache2/ports.conf
```

Thay đổi “Listen” bằng cổng khác chưa được sử dụng.

```
$ sudo service apache2 restart
```

### 2.2.2. Permissions not granted.

Nguyên nhân: Lỗi xảy ra khi người dùng không có đủ quyền để cài đặt hoặc chạy Apache2.

Cách xử lý: Đảm bảo chạy lệnh cài đặt và cấu hình với quyền sudo.

Kiểm tra và thay đổi quyền của các thư mục và tệp cấu hình liên quan:

```
$ sudo chown -R www-data:www-data /var/www/html
```

```
$ sudo chmod -R 755 /var/www/html
```

## 3. Cài đặt RabbitMQ

### 3.1 Hướng dẫn cài đặt

Bước 1: Cài đặt các phụ thuộc thiết yếu

```
$ sudo apt-get update -y
```

```
$ sudo apt-get install curl gnupg -y
```

Bước 2: Cài đặt apt HTTPS Transport

```
$ sudo apt-get install apt-transport-https
```

Bước 3: Thêm mã khóa GPG vào kho lưu trữ ứng dụng RabbitMQ

```
$ sudo apt-get install curl gnupg apt-transport-https -y
```

#### Bước 4: Thêm tệp danh sách mã nguồn

```
1. curl -sLf "https://keys.openpgp.org/vks/v1/by-fingerprint/0A9AF2115F4687BD29803A206B73A36E6026DFC" |
2.  sudo gpg --dearmor |
3.  sudo tee /usr/share/keyrings/com.rabbitmq.team.gpg
4.  > /dev/null
5.  curl -sLf https://github.com/rabbitmq/signing-
   keys/releases/download/3.0/cloudsmith.rabbitmq-
   erlang.E495BB49CC4BBE5B.key |
6.  sudo gpg --dearmor |
7.  sudo tee
   /usr/share/keyrings/rabbitmq.E495BB49CC4BBE5B.gpg
8.  > /dev/null
9.  curl -sLf https://github.com/rabbitmq/signing-
   keys/releases/download/3.0/cloudsmith.rabbitmq-
   server.9F4587F226208342.key |
10. sudo gpg --dearmor |
11. sudo tee /usr/share/keyrings/rabbitmq.9F4587F226208342.gpg
12. > /dev/null
```

#### Bước 5: Cập nhật các gói phần mềm vừa thêm

```
$ sudo apt-get update -y
```

#### Bước 6: Cài đặt Erlang

```
1. $ sudo apt-get install -y erlang-base \
2.   erlang-asn1 erlang-crypto erlang-eldap \
3.   erlang-ftp erlang-inets \
4.   erlang-mnesia erlang-os-mon \
5.   erlang-parsetools erlang-public-key \
6.   erlang-runtime-tools \
7.   erlang-snmp erlang-ssl \
8.   erlang-syntax-tools erlang-tftp \
9.   erlang-tools erlang-xmerl
```

Bước 7: Hoàn tất quá trình cài đặt RabbitMQ

```
$ sudo apt-get install rabbitmq-server -y --fix-missing
```

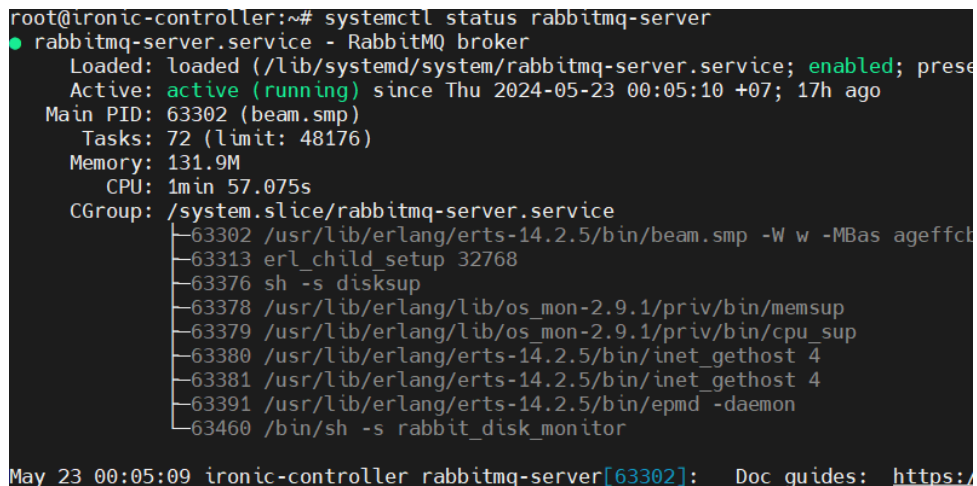
Bước 8: Khởi chạy dịch vụ RabbitMQ

```
$ sudo systemctl enable rabbitmq-server
```

```
$ sudo systemctl start rabbitmq-server
```

Bước 9: Kiểm tra dịch vụ nếu RabbitMQ đã cài đặt thành công

```
$ sudo systemctl status rabbitmq-server
```



```
root@ironic-controller:~# systemctl status rabbitmq-server
● rabbitmq-server.service - RabbitMQ broker
   Loaded: loaded (/lib/systemd/system/rabbitmq-server.service; enabled; prese
   Active: active (running) since Thu 2024-05-23 00:05:10 +07; 17h ago
     Main PID: 63302 (beam.smp)
        Tasks: 72 (limit: 48176)
      Memory: 131.9M
         CPU: 1min 57.075s
    CGroup: /system.slice/rabbitmq-server.service
            └─63302 /usr/lib/erlang/erts-14.2.5/bin/beam.smp -W w -MBas agefft
            └─63313 erl_child_setup 32768
            └─63376 sh -s disksup
            └─63378 /usr/lib/erlang/lib/os_mon-2.9.1/priv/bin/memsup
            └─63379 /usr/lib/erlang/lib/os_mon-2.9.1/priv/bin/cpu_sup
            └─63380 /usr/lib/erlang/erts-14.2.5/bin/inet_gethost 4
            └─63381 /usr/lib/erlang/erts-14.2.5/bin/inet_gethost 4
            └─63391 /usr/lib/erlang/erts-14.2.5/bin/epmd -daemon
            └─63460 /bin/sh -s rabbit_disk_monitor

May 23 00:05:09 ironic-controller rabbitmq-server[63302]: Doc guides: https://
```

Hình 3.1. Dịch vụ RabbitMQ đã cài đặt thành công

Bước 10: Thiết lập tài khoản người dùng ironic trong RabbitMQ

```
$ sudo rabbitmqctl add_user ironic 'password'
```

\*Thay thế 'password' bằng mật khẩu phù hợp.

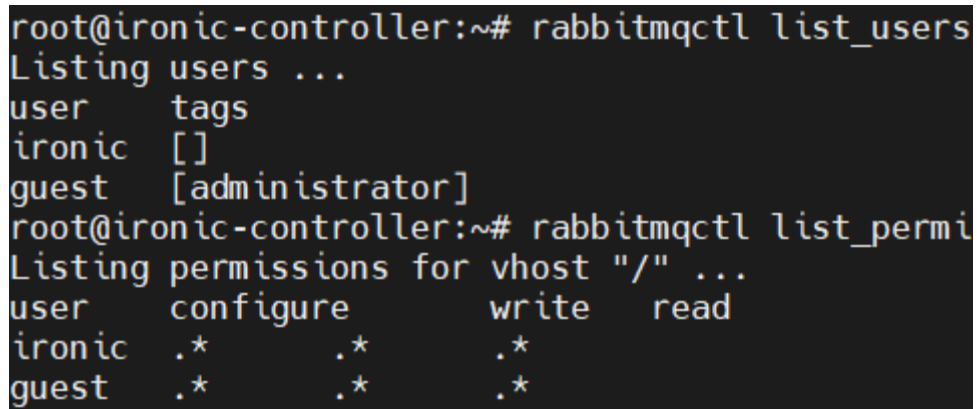
Phân quyền cho tài khoản Ironic

```
$ sudo rabbitmqctl set_permissions -p / ironic ".*" ".*" ".*"
```

Kiểm tra tài khoản đã được tạo thành công

```
$ rabbitmqctl list_users
```

```
$ rabbitmqctl list_permissions
```



The screenshot shows a terminal window with the following output:

```
root@ironic-controller:~# rabbitmqctl list_users
Listing users ...
user      tags
ironic    []
guest     [administrator]
root@ironic-controller:~# rabbitmqctl list_permissions
Listing permissions for vhost "/" ...
user      configure      write      read
ironic    .*              .*         .*
guest     .*              .*         .*
```

Hình 3.2. Tài khoản ironic đã được tạo và phân quyền thành công.

## 3.2. Lỗi khi cài đặt RabbitMQ

### 3.2.1. Unable to start RabbitMQ-service.

Nguyên nhân: Lỗi xảy ra do trong quá trình cài đặt RabbitMQ gặp vấn đề.

Cách xử lý:

Kiểm tra lỗi chi tiết trong nhật ký dịch vụ RabbitMQ:

```
$ sudo tail -n 50 /var/log/rabbitmq/rabbit@<hostname>.log
```

Kiểm tra lại tệp mã nguồn chứa danh sách khóa ký

```
$ sudo nano /etc/apt/sources.list.d/rabbitmq.list
```

Cấu hình tệp etc/ironic/ironic.conf:

```
$ transport_url = rabbit://ironic:password@localhost:5672/
```

Cài đặt lại gói Erlang bị lỗi:

```
1. $ sudo apt-get install -y erlang-base \
2. erlang-asn1 erlang-crypto erlang-eldap \
3. erlang-ftp erlang-inets \
4. erlang-mnesia erlang-os-mon \
5. erlang-parsetools erlang-public-key \
6. erlang-runtime-tools \
7. erlang-snmp erlang-ssl \
8. erlang-syntax-tools erlang-tftp \
9. erlang-tools erlang-xmerl
```

Cài đặt lại các phụ thuộc của RabbitMQ

```
$ sudo apt-get install rabbitmq-server -y --fix-missing
```

### 3.2.2. (403) ACCESS\_REFUSED - Login was refused using authentication mechanism AMQPLAIN.

Nguyên nhân: Lỗi xảy ra do RabbitMQ từ chối truy cập do vấn đề xác thực. Thường do nhập sai thông tin trong tệp cấu hình /etc/ironic/ironic.conf

Cách xử lý:

Kiểm tra lỗi chi tiết trong nhật ký dịch vụ RabbitMQ:

```
$ sudo tail -n 50 /var/log/rabbitmq/rabbit@<hostname>.log
```

Tạo và phân quyền cho tên đăng nhập ironic:

```
$ sudo rabbitmqctl add_user ironic password
```

```
$ sudo rabbitmqctl set_permissions -p / ironic ".*" ".*" ".*"
```

Cấu hình tệp etc/ironic/ironic.conf:

```
$ transport_url = rabbit://ironic:password@localhost:5672
```

Khởi động lại dịch vụ

```
$ sudo systemctl restart rabbitmq-server ironic-api ironic-conductor
```

## 4. Cài đặt Keystone

### 4.1 Hướng dẫn cài đặt

Bước 1: Tạo cơ sở dữ liệu cho Keystone

```
$ mysql -u root -p  
$ MariaDB [(none)]> CREATE DATABASE keystone;
```

Bước 2: Cấp quyền truy cập dữ liệu cho cơ sở dữ liệu Keystone

```
1. MariaDB [(none)]> GRANT ALL PRIVILEGES ON  
  \  
2. keystone.* TO 'keystone'@'localhost'\  
3. IDENTIFIED BY 'KEYSTONE_DBPASS';  
4. MariaDB [(none)]> GRANT ALL PRIVILEGES ON  
  \  
5. keystone.* TO 'keystone'@'%'  
6. IDENTIFIED BY 'KEYSTONE_DBPASS';
```

\*Thay thế **'KEYSTONE\_DBPASS'** bằng mật khẩu phù hợp.

Bước 3: Cài đặt dịch vụ Openstack Keystone

```
$ apt-get install keystone
```

Bước 4: Cấu hình dịch vụ Openstack Keystone

Truy cập tập tin cấu hình dữ liệu của dịch vụ keystone

```
$ nano /etc/keystone/keystone.conf
```

Tiến hành kết nối với cơ sở dữ liệu trong phần [database]

```
connection=mysql+pymysql://keystone:KEYSTONE_DBP  
ASS@controller/keystone
```

Thay thế **KEYSTONE\_DBPASS** bằng mật khẩu đã nhập ở Bước 2.

Cấu hình nhà cung cấp trong phần [token]

```
provider = fernet
```

Bước 5: Đồng bộ cơ sở dữ liệu cho dịch vụ Openstack Keystone

```
$ su -s /bin/sh -c "keystone-manage db_sync" keystone
```

Bước 6: Khởi tạo kho lưu trữ khóa Fernet

```
$ keystone-manage fernet_setup --keystone-user keystone -  
-keystone-group keystone
```

```
$ keystone-manage credential_setup --keystone-user  
keystone --keystone-group keystone
```

Bước 7: Khởi động dịch vụ Openstack Keystone

```
1. $ keystone-manage bootstrap --bootstrap-password  
ADMIN_PASS  
2. --bootstrap-admin-url http://controller:35357/v3/ \  
3. --bootstrap-internal-url http://controller:5000/v3/ \  
4. --bootstrap-public-url http://controller:5000/v3/ \  
5. --bootstrap-region-id RegionOne
```

\* Thay thế **ADMIN\_PASS** bằng mật khẩu đã nhập ở **Bước 2**.

Bước 8: Cấu hình máy chủ web Apache2 cho Keystone

Tạo và chỉnh sửa tệp tin cấu hình apache2

```
$ nano /etc/sysconfig/apache2
```

```
APACHE_SERVERNAME="controller"
```

Tạo file cấu hình máy chủ web cho dịch vụ Keystone

```
$ nano /etc/apache2/conf.d/wsgi-keystone.conf
```

Cấu hình file ‘wsgi-keystone.conf’ vừa tạo với nội dung:

```
1. Listen 5000
2. Listen 35357
3.
4. <VirtualHost *:5000>
5.     WSGIDaemonProcess keystone-public processes=5
   threads=1 user=keystone group=keystone display-
   name=%{GROUP}
6.     WSGIProcessGroup keystone-public
7.     WSGIScriptAlias / /usr/bin/keystone-wsgi-public
8.     WSGIApplicationGroup %{GLOBAL}
9.     WSGIPassAuthorization On
10.    ErrorLogFormat "%{cu}t%M"
11.    ErrorLog /var/log/apache2/keystone.log
12.    CustomLog /
13.    <Directory /usr/bin>
14.        Require all granted
15.    </Directory>
16. </VirtualHost>
17.
18. <VirtualHost *:35357>
19. WSGIDaemonProcess keystone-admin processes=5
   threads=1 user=keystone group=keystone display-
   name=%{GROUP}
20. WSGIProcessGroup keystone-admin
21. WSGIScriptAlias / /usr/bin/keystone-wsgi-admin
22. WSGIApplicationGroup %{GLOBAL}
23. WSGIPassAuthorization On
24. ErrorLogFormat "%{cu}t%M"
25. ErrorLog /var/log/apache2/keystone.log
26. CustomLog /var/log/apache2/keystone_access.log
   combined
27.
28. <Directory /usr/bin>
29.     Require all granted
30. </Directory>
31. </VirtualHost>
```

Thay đổi quyền sở hữu cho thư mục /etc/keystone

```
$ chown -R keystone:keystone /etc/keystone
```



## Bước 9: Hoàn tất việc cài đặt

### Khởi động dịch vụ máy chủ web apache2

```
$ systemctl enable apache2.service
```

```
$ systemctl start apache2.service
```

### Cấu hình tài khoản quản trị

1. `export OS_USERNAME=admin`
2. `export OS_PASSWORD=ADMIN_PASS`
3. `export OS_PROJECT_NAME=admin`
4. `export OS_USER_DOMAIN_NAME=Default`
5. `export OS_PROJECT_DOMAIN_NAME=Default`
6. `export OS_AUTH_URL=http://controller:35357/v3`
7. `export OS_IDENTITY_API_VERSION=3`

\* Thay thế **ADMIN\_PASS** bằng mật khẩu đã nhập ở Bước 2.

## Bước 10: Kiểm tra dịch vụ Keystone đã hoạt động

```
$ systemctl start apache2.service
```

```
root@ironic-controller:~# sudo systemctl status keystone
● keystone.service - OpenStack Keystone API (keystone)
   Loaded: loaded (/lib/systemd/system/keystone.service; enabled; preset: enabled)
   Active: active (running) since Wed 2024-05-22 14:01:32 +07; 1 day 7h ago
     Docs: man:keystone(1)
   Main PID: 1229 (uwsgi_python311)
    Tasks: 257 (limit: 48176)
   Memory: 1.1G
      CPU: 1min 23.714s
   CGroup: /system.slice/keystone.service
           └─1229 /usr/bin/uwsgi_python311 --http-socket :5000 --ini /etc/keystone/keystone-uwsgi
           └─1246 /usr/bin/uwsgi_python311 --http-socket :5000 --ini /etc/keystone/keystone-uwsgi
           └─1248 /usr/bin/uwsgi_python311 --http-socket :5000 --ini /etc/keystone/keystone-uwsgi
           └─1250 /usr/bin/uwsgi_python311 --http-socket :5000 --ini /etc/keystone/keystone-uwsgi
           └─1252 /usr/bin/uwsgi_python311 --http-socket :5000 --ini /etc/keystone/keystone-uwsgi
           └─1254 /usr/bin/uwsgi_python311 --http-socket :5000 --ini /etc/keystone/keystone-uwsgi
           └─1256 /usr/bin/uwsgi_python311 --http-socket :5000 --ini /etc/keystone/keystone-uwsgi
           └─1258 /usr/bin/uwsgi_python311 --http-socket :5000 --ini /etc/keystone/keystone-uwsgi
           └─1259 /usr/bin/uwsgi_python311 --http-socket :5000 --ini /etc/keystone/keystone-uwsgi

May 22 14:01:32 ironic-controller systemd[1]: Started keystone.service - OpenStack Keystone API (key
May 22 14:01:33 ironic-controller keystone[1229]: [uwsgi] implicit plugin requested python311
May 22 14:01:33 ironic-controller keystone[1229]: [uWSGI] getting INI configuration from /etc/keysto
May 23 11:56:52 ironic-controller systemd[1]: keystone.service: Dependency After=keystone.service is
May 23 11:56:54 ironic-controller systemd[1]: keystone.service: Dependency After=keystone.service is
May 23 17:19:07 ironic-controller systemd[1]: keystone.service: Dependency After=keystone.service is
root@ironic-controller:~#
```

Hình 4.1. Dịch vụ Keystone đã hoạt động

## 4.2. Lỗi khi cài đặt Openstack Keystone

### 4.2.1. AttributeError: 'NoneType' object has no attribute 'getcurrent'

Nguyên nhân: Lỗi trong mã nguồn python.

Cách xử lý:

Truy cập và sửa tệp tin thread.py:

```
nano /usr/lib/python3/dist-  
packages/eventlet/green/thread.py
```

Tìm và thay thế đoạn code:

```
1. def get_ident(gr=None):  
2.     if gr is None:  
3.         return id(greenlet.getcurrent())  
4.     else: return id(gr)
```

Bằng đoạn code bên dưới để bắt ngoại lệ:

```
1. def get_ident(gr=None):  
2.     try  
3.         if gr is None:  
4.             return id(greenlet.getcurrent())  
5.         else:  
6.             return id(gr)  
7.     except:  
8.         return id(gr)
```

Tiếp tục quá trình cài đặt

```
$ su -s /bin/bash keystone -c "keystone-manage db_sync"
```

### 4.2.2. Missing value auth-url required for auth plugin password

Nguyên nhân: Lỗi xảy ra do máy chủ thiếu các thông tin xác thực để kết nối tới dịch vụ Keystone

Cách xử lý:

Tạo và thiết lập các biến môi trường trong tệp */openrc.sh*:

```
1. $ export OS_AUTH_URL=http://keystone_url:5000/v3
2. $ export OS_USERNAME=username
3. $ export OS_PASSWORD=password
4. $ export OS_PROJECT_NAME=project
5. $ export OS_PROJECT_DOMAIN_NAME=Default
6. $ export OS_USER_DOMAIN_NAME=Default
7. $ export OS_IDENTITY_API_VERSION=3
```

Đọc và thực thi tệp openrc.sh

```
$ source /path/to/your/openrc.sh
```

#### 4.2.3. The request you have made requires authentication. (HTTP 401)

Nguyên nhân: Máy chủ cung cấp sai thông tin xác thực

Cách xử lý:

Kiểm tra lại các thông tin đã khai báo trong tệp */openrc.sh*

Kiểm tra nhật ký hệ thống của Keystone

```
$ sudo tail -f /var/log/keystone/keystone.log
```

Khởi động lại các dịch vụ

```
1. $ sudo systemctl restart keystone
2. $ sudo systemctl restart ironic-api
3. $ sudo systemctl restart ironic-conductor
```

## 5. Cài đặt Ironic

### 5.1. Cài đặt dịch vụ ironic-api

Bước 1: Thiết lập cơ sở dữ liệu cho hệ thống Ironic.

Truy cập dịch vụ mysql.

```
$ mysql -u root -p
```

Tiến hành thiết lập cơ sở dữ liệu.

```
1. CREATE DATABASE ironic CHARACTER SET utf8mb3;  
2. GRANT ALL PRIVILEGES ON ironic.* TO  
   'ironic'@'localhost'\  
3. IDENTIFIED BY 'IRONIC_DBPASSWORD';  
4. GRANT ALL PRIVILEGES ON ironic.* TO 'ironic'@'%'\  
5. IDENTIFIED BY 'IRONIC_DBPASSWORD';
```

Bước 2: Cài đặt các gói thành phần trong Ironic.

```
$ sudo apt-get install ironic-api ironic-conductor python3-  
ironicclient
```

Bước 3: Cấu hình các thành phần trong Ironic.

Truy cập tệp tin cấu hình của dịch vụ Ironic và cấu hình ironic-api.

```
$ nano etc/ironic/ironic.conf
```

Thiết lập kết nối dịch vụ Ironic với cơ sở dữ liệu Mysql

```
1. [database]  
2. connection=mysql+pymysql://ironic:IRONIC_DBPA  
   SSWORD@DB_IP/ironic? charset=utf8
```

\* Thay thế **IRONIC\_DBPASSWORD** bằng mật khẩu đã nhập ở đã thiết lập cho tài khoản 'ironic' thiết lập ở bước 1.

\* Thay thế **DB\_IP** bằng địa chỉ IP của máy chủ MySQL.

Thiết lập kết nối dịch vụ ironic-api với dịch vụ RabbitMQ

```
1. [DEFAULT]
2. transport_url =
   rabbit://ironic:RPC_PASSWORD@RPC_HOST:RPC_PORT
```

\* Thay thế trường ***RPC\_PASSWORD*** bằng mật khẩu đã thiết lập cho tài khoản ‘ironic’ ở phần cài đặt dịch vụ **RabbitMQ**.

## 5.2. Cấu hình dịch vụ ironic-conductor

```
1. [DEFAULT]
2. #...
3. my_ip=HOST_IP
```

\* Thay thế ***HOST\_IP*** bằng địa chỉ IP của máy chủ ironic-conductor đang chạy.

Cấu hình dịch vụ ironic-api xác thực thông tin qua dịch vụ Keystone

```
1.  [DEFAULT]
2.
3.  auth_strategy=keystone
4.
5.  [keystone_authtoken]
6.
7.  # Authentication type to load (string value)
8.  auth_type=password
9.
10. # Complete public Identity API endpoint (string value)
11. www_authenticate_uri=http://PUBLIC_IDENTITY_IP:5000
12.
13. # Complete admin Identity API endpoint. (string value)
14. auth_url=http://PRIVATE_IDENTITY_IP:5000
15.
16. # Service username. (string value)
17. username=ironic
18.
19. # Service account password. (string value)
20. password=IRONIC_PASSWORD
21.
22. # Service tenant name. (string value)
23. project_name=service
24.
25. # Domain name containing project (string value)
26. project_domain_name=Default
27.
28. # User's domain name (string value)
29. user_domain_name=Default
30.
31. # memcached setting (string value)
32. memcached_servers=MEMCACHED_SERVER:11211
```

\* Thay thế **PUBLIC\_IDENTITY\_IP** và **PRIVATE\_IDENTITY\_IP** bằng địa chỉ IP công khai và IP riêng tư của dịch vụ Keystone.

\* Thay thế **IRONIC\_PASSWORD** bằng mật khẩu của người dùng ironic đã khai báo trong Keystone.

\* Thay thế **MEMCACHED\_SERVER** bằng tên hoặc địa chỉ của máy chủ memcached.

Cấu hình dịch vụ ironic-api để giao tiếp với dịch vụ ironic-conductor sử dụng JSON RPC

```
1. [DEFAULT]
2.
3. rpc_transport = json-rpc
4.
5. [json_rpc]
6.
7. # Authentication type to load (string value)
8. auth_type = password
9.
10. # Authentication URL (string value)
11. auth_url=https://IDENTITY_IP:5000/
12.
13. # Username (string value)
14. username=ironic
15.
16. # User's password (string value)
17. password=IRONIC_PASSWORD
18.
19. # Project name to scope to (string value)
20. project_name=service
21.
22. # Domain ID containing project (string value)
23. project_domain_id=default
24.
25. # User's domain id (string value)
26. user_domain_id=default
```

\* Thay thế **IDENTITY\_IP** bằng tên hoặc địa chỉ của máy chủ Keystone.

\* Thay thế **IRONIC\_PASSWORD** bằng mật khẩu của người dùng ironic đã khai báo trong Keystone.

Đồng bộ hóa cơ sở dữ liệu của dịch vụ Ironic.

```
$ ironic-dbsync --config-file /etc/ironic/ironic.conf create_schema
```

Cấu hình ironic-api sử dụng mô-đun mod\_wsgi trong apache.

```
1. $ sudo tee /etc/apache2/ironic <<EOF
2. Listen 6385
3.
4. <VirtualHost *:6385>
5.     WSGIDaemonProcess ironic user=stack group=stack
   threads=10 display-name=%{GROUP}
6.     WSGIScriptAlias / /usr/local/bin/ironic-wsgi-api
7.
8.     SetEnv APACHE_RUN_USER stack
9.     SetEnv APACHE_RUN_GROUP stack
10.    WSGIProcessGroup ironic
11.
12.    ErrorLog /var/log/apache2/ironic_error.log
13.    LogLevel info
14.    CustomLog /var/log/apache2/ironic_access.log combined
15.
16.    <Directory /opt/stack/ironic/ironic/api>
17.        WSGIProcessGroup ironic
18.        WSGIApplicationGroup %{GLOBAL}
19.        AllowOverride All
20.        Require all granted
21.    </Directory>
22. </VirtualHost>
23. EOF
```

```
$ sudo cp etc/apache2/ironic /etc/apache2/sites-
available/ironic.conf
```

Khởi động lại dịch vụ Ironic và apache2

```
1. $ sudo systemctl restart ironic-api
2. $ sudo systemctl restart apache2
3. $ sudo a2ensite ironic
```



Kiểm tra các dịch vụ thành phần chính của Ironic đã hoạt động

```
$ sudo systemctl status ironic-api
```

```
root@ironic-controller:~# systemctl status ironic-api
● ironic-api.service - OpenStack Ironic Service API (ironic-api)
   Loaded: loaded (/lib/systemd/system/ironic-api.service; enabled; preset: enabled)
   Active: active (running) since Thu 2024-05-23 00:05:57 +07; 21h ago
     Docs: man:ironic-api(1)
  Main PID: 63580 (uwsgi_python311)
    Tasks: 13 (limit: 48176)
   Memory: 851.4M
      CPU: 1min 17.731s
   CGroup: /system.slice/ironic-api.service
           └─63580 /usr/bin/uwsgi_python311 --http-socket :6385 --ini /etc/ironic/ironic-api-uwsgi.ini
           └─63587 /usr/bin/uwsgi_python311 --http-socket :6385 --ini /etc/ironic/ironic-api-uwsgi.ini
           └─63588 /usr/bin/uwsgi_python311 --http-socket :6385 --ini /etc/ironic/ironic-api-uwsgi.ini
           └─63589 /usr/bin/uwsgi_python311 --http-socket :6385 --ini /etc/ironic/ironic-api-uwsgi.ini
           └─63590 /usr/bin/uwsgi_python311 --http-socket :6385 --ini /etc/ironic/ironic-api-uwsgi.ini
           └─63591 /usr/bin/uwsgi_python311 --http-socket :6385 --ini /etc/ironic/ironic-api-uwsgi.ini
           └─63592 /usr/bin/uwsgi_python311 --http-socket :6385 --ini /etc/ironic/ironic-api-uwsgi.ini
           └─63593 /usr/bin/uwsgi_python311 --http-socket :6385 --ini /etc/ironic/ironic-api-uwsgi.ini
           └─63594 /usr/bin/uwsgi_python311 --http-socket :6385 --ini /etc/ironic/ironic-api-uwsgi.ini

May 23 00:05:57 ironic-controller systemd[1]: Started ironic-api.service - OpenStack Ironic Service API (iro
May 23 00:05:57 ironic-controller ironic-api[63580]: [uwsgi] implicit plugin requested python311
May 23 00:05:57 ironic-controller ironic-api[63580]: [uwsgi] getting INI configuration from /etc/ironic/ironic
```

Hình 5.1. Dịch vụ ironic-api đã hoạt động

```
$ sudo systemctl status ironic-conductor
```

```
root@ironic-controller:~# sudo systemctl status ironic-conductor
● ironic-conductor.service - OpenStack Ironic Conductor (ironic-conductor)
   Loaded: loaded (/lib/systemd/system/ironic-conductor.service; enabled; preset: enabled)
   Active: active (running) since Thu 2024-05-23 00:17:58 +07; 21h ago
     Docs: man:ironic-conductor(1)
  Main PID: 64695 (ironic-conductor)
    Tasks: 1 (limit: 48176)
   Memory: 121.9M
      CPU: 1min 51.656s
   CGroup: /system.slice/ironic-conductor.service
           └─64695 /usr/bin/python3 /usr/bin/ironic-conductor --config-file=/etc/ironic/ironic-conductor.conf

May 23 21:36:03 ironic-controller ironic-conductor[64695]: /usr/lib/python3/dist-packages/ironic/conductor/
```

Hình 5.2. Dịch vụ ironic-conductor đã hoạt động

Kiểm tra đường dẫn tới các dịch vụ đang hoạt động

```
$ openstack endpoint list
```

```
root@ironic-controller:~# openstack endpoint list
```

ID	Region	Service Name	Service Type	Enabled	Interface
1b7eb1e98be740bcb0f9bb2a67237ecc	regionOne	ironic	baremetal	True	internal
4a6e20a3e2d243299fe86794d15a8076	regionOne	keystone	identity	True	internal
8296d476dd4441f5b2a4304115ad9b7c	regionOne	keystone	identity	True	admin
9875ac1df35845cc98832e9eed6414a4	regionOne	ironic	baremetal	True	admin
d25146c3e65f4367838bcf8706087fd3	regionOne	keystone	identity	True	public
f33bab868db94ce198f71b3a8705541e	regionOne	ironic	baremetal	True	public

Hình 5.3. Các dịch vụ thành phần chính của Ironic đang hoạt động.

### 5.3. Các lỗi khi cài đặt Ironic

#### 5.3.1. [Errno 2] No such file or directory: '/httpboot/boot.ipxe'

Nguyên nhân: Chưa cấu hình trình giao diện khởi động hệ thống

Cách xử lý: Xem phụ lục [phần 6](#).

#### 5.3.2. ipmi\_si: Unable to find any System Interface(s)

Nguyên nhân: mô đun ipmi không tìm thấy trình điều khiển IPMI

Cách xử lý:

Kiểm tra lại các mô đun đã được cài đặt và thiết lập

1. \$ **sudo** modprobe ipmi\_msghandler
2. \$ **sudo** modprobe ipmi\_devintfs
3. \$ **sudo** modprobe ipmi\_si

Truy cập BIOS và kích hoạt IPMI hoặc BMC trong cài đặt

Sử dụng IPMI qua LAN thông qua máy ảo

1. \$ virsh start ipmi-vm
2. \$ ipmitool -I lanplus -H <bmc-ip> -U ironic -P password chassis power status

Khởi động lại dịch vụ của IPMI

1. \$ **sudo** systemctl restart ipmiev.service
2. \$ **sudo** systemctl status ipmiev.service

### 5.3.3. Openstack baremetal node deploy: error: unrecognized arguments --driver-info/instance-info/name-interface

Nguyên nhân: Do khai báo sai thuộc tính phần cứng khi tạo máy chủ

Cách xử lý:

Kiểm tra tên phần cứng cần khai báo và trình điều khiển tương ứng

Khai báo lại các thành phần cần cung cấp cho máy vật lý

```
1. $ openstack baremetal node set <node-name> \  
2.                               --driver-info \  
3.                               --instance-info \  
4.                               --*-interface
```

```
$ openstack baremetal node rebuild <node-name>
```

## 6. Cấu hình giao diện khởi động cho hệ thống

Việc cấu hình giao diện khởi động (boot interface) sẽ giúp hệ thống Ironic nhận diện được các giao diện khởi động chính trong việc triển khai các nút máy chủ vật lý. Đồng thời, boot interface cũng tham gia vào các quá trình quản lý và triển khai đĩa bộ nhớ cùng các tệp lưu trữ hệ điều hành.

Cấu hình máy chủ TFTP.

Bước 1: Tạo thư mục gốc cho tftp server có thể giao tiếp với máy chủ ironic-conductor.

1. `$ sudo mkdir -p /tftpboot`
2. `$ sudo chown -R ironic /tftpboot`

Cài đặt máy chủ tftp.

- ```
$ sudo apt-get install xinetd tftpd-hpa
```

Bước 2: Tạo hoặc chỉnh sửa file `/etc/xinetd.d/tftp` với nội dung dưới:

1. `{`
2. `protocol = udp`
3. `port = 69`
4. `socket_type = dgram`
5. `wait = yes`
6. `user = root`
7. `server = /usr/sbin/in.tftpd`
8. `server_args = -v -v -v -v -v --map-file`  
`/tftpboot/map-file /tftpboot`
9. `disable = no`
10. `flags = IPv4`
11. `}`

Bước 3: Tạo các map-file trong thư mục gốc

1. `$ echo 're ^(/tftpboot/) /tftpboot\2' > /tftpboot/map-file`
2. `$ echo 're ^/tftpboot/ /tftpboot/' >> /tftpboot/map-file`
3. `$ echo 're ^(^) /tftpboot\1' >> /tftpboot/map-file`
4. `$ echo 're ^([^\]) /tftpboot\1' >> /tftpboot/map-file`

#### Bước 4: Cài đặt UEFI – Grub

Cài đặt Grub2 và các gói shim

```
$ sudo apt-get install grub-efi-amd64-signed shim-signed
```

Copy các file grub và shim boot vào thư mục /tftpboot

1. \$ sudo cp /usr/lib/shim/shimx64.efi.signed /tftpboot/bootx64.efi
2. \$ sudo cp /usr/lib/grub/x86\_64-efi-signed/grubnetx64.efi.signed /tftpboot/grubx64.efi

#### Bước 5: Cài đặt bộ nạp PXE

Tạo các thư mục và phân quyền

1. \$ sudo mkdir -p /tftpboot
2. \$ sudo mkdir -p /httpboot
3. \$ sudo chown -R ironic /tftpboot
4. \$ sudo chown -R ironic /httpboot

Tạo các map-file trong thư mục gốc

1. \$ echo 'r ^([^\]) /tftpboot/\1' > /tftpboot/map-file
2. \$ echo 'r ^(/tftpboot/) /tftpboot/\2' >> /tftpboot/map-file

Cài đặt máy chủ TFTP và HTTP trong tệp cấu hình

1. /etc/ironic/ironic.conf
- 2.
3. [pxe]
4. tftp\_root=/tftpboot
5. tftp\_server=localhost
- 6.
7. [deploy]
8. http\_root=/httpboot
9. http\_url=http://localhost:80

Tùy vào lựa chọn sử dụng máy chủ tftp hay http, người dùng có thể lựa chọn cài đặt các bộ nạp khác nhau. Nếu muốn sử dụng máy chủ http, cần phải cài đặt và cấu hình bộ nạp iPXE như dưới đây:

```
$ apt-get install ipxe
```

```
$ cp /usr/lib/ipxe/{undionly.kpxe,ipxe.efi,snponly.efi}
```

Cho phép ghi đè cấu hình iPXE trong tệp cấu hình

```
1. /etc/ironic/ironic.conf
2.
3. [pxe]
4. ipxe_bootfile_name=undionly.kpxe
5. uefi_ipxe_bootfile_name=ipxe.efi
6. ipxe_config_template=$pybasedir/drivers/modules/ipxe_config.
  template
7.
8. [DEFAULT]
9. default_boot_interface=ipxe
10. enabled_boot_interfaces=ipxe,pxe
```

Khởi động lại dịch vụ ironic-conductor để ghi nhận thay đổi

```
$ sudo service ironic-conductor restart
```

## 7. Cấu hình trình điều khiển cho hệ thống

Dịch vụ Ironic ủy quyền quản lý phần cứng thực tế cho trình điều khiển. Trình điều khiển, còn được gọi là loại phần cứng, bao gồm các giao diện phần cứng, có chức năng xử lý một số khía cạnh của việc cung cấp máy chủ vật lý theo cách riêng biệt. Có các loại phần cứng dùng chung (ví dụ: redfish và ipmi) và các loại phần cứng riêng biệt (ví dụ: ilo và irmc).

Bước 1: Truy cập tệp cấu hình dịch vụ ironic

```
$ nano /etc/ironic/ironic.conf
```

Bước 2: Điều chỉnh các tham số như bên dưới để kích hoạt các trình điều khiển phục vụ cho việc triển khai các nốt máy chủ vật lý.

1. [DEFAULT]
2. enabled\_hardware\_types = ipmi
3. enabled\_boot\_interfaces = pxe
4. enabled\_console\_interfaces = ipmitool-socat,no-console
5. enabled\_deploy\_interfaces = direct
6. enabled\_inspect\_interfaces = inspector
7. enabled\_management\_interfaces = ipmitool
8. enabled\_network\_interfaces = flat
9. enabled\_power\_interfaces = ipmitool
10. enabled\_raid\_interfaces = agent
11. enabled\_storage\_interfaces = cinder,noop
12. enabled\_vendor\_interfaces = ipmitool,no-vendor

Bước 3: Tái khởi động ironic-conductor để ghi nhận các thay đổi

```
$ sudo service ironic-conductor restart
```

## 8. Các lỗi khác

### **Error: externally-managed-environment × This environment is externally managed**

Nguyên nhân: Môi trường thực thi của Python nằm ngoài quản lý của hệ điều hành Debian

Cách khắc phục: Cài đặt python3 và pipx

1. \$ **sudo** apt-get **install** python3
2. \$ **sudo** apt-get **install** pipx

Kích hoạt môi trường thực thi Python

1. \$ python3 -m venv .venv
2. \$ **source** .venv/bin/activate
3. \$ python3 -m pip **install** -r requirements.txt

Tạo file cấu hình để mở khóa thực thi python trực tiếp

1. \$ **sudo touch** ~/.config/pip/pip.conf
2. [global]
3. **break**-system-packages = **true**

### **Error: Unable to establish IPMI v2 / RMCP+ session**

Nguyên nhân: Hệ thống không tìm thấy module IPMI

Cách xử lý:

Sử dụng nmap để kiểm tra kết nối đến module IPMI

```
$ nmap -p 623 -sU -P0 <ipmi-address>
```

Truy cập BIOS và kích hoạt IPMI over LAN

Menu Overview -> IDRAC SETTINGS/BMC -> User -> Apply



## 9. Tạo và cấu hình các máy chủ máy vật lý

Để khởi tạo một máy chủ vật lý hay còn gọi là node trong chế độ chạy độc lập của Ironic, người dùng có thể sử dụng cú pháp khai báo sau:

1. openstack baremetal node create \
2. --driver <Tên driver cho máy> \
3. --name <Tên của máy> \
4. --resource-class <Lớp tài nguyên> \
5. --property cpus=<Số vi xử lý> \
6. --property memory\_mb=<Dung lượng Ram> \
7. --property local\_gb=<Dung lượng Rom> \
8. --driver-info ipmi\_address=<Địa chỉ kết nối ipmi> \
9. --driver-info ipmi\_port=<Cổng ipmi> \
10. --driver-info ipmi\_username=<Tên đăng nhập ipmi> \
11. --driver-info ipmi\_password=<Mật khẩu ipmi> \
12. --deploy-interface <Giao diện triển khai> \
13. --network-interface <Giao diện mạng> \
14. --instance-info image\_source=<Nguồn lưu trữ hệ điều hành> \
15. --instance-info image\_checksum=<Mã lưu trữ hệ điều hành> \
16. --instance-info image\_type=<Kiểu phân vùng đĩa> \
17. --instance-info root\_gb=<Kích cỡ ổ đĩa> \
18. --driver-info deploy\_kernel=<Đường dẫn khởi động> \
19. --driver-info deploy\_ramdisk=<Đường dẫn initrd>

Ví dụ:

```
1.  openstack baremetal node create \  
2.  --driver ipmi \  
3.  --name test \  
4.  --resource-class baremetal \  
5.  --property cpus=2\  
6.  --property memory_mb=2048 \  
7.  --property local_gb=20 \  
8.  --driver-info ipmi_address=10.8.52.93 \  
9.  --driver-info ipmi_port=623 \  
10. --driver-info ipmi_username=ironic \  
11. --driver-info ipmi_password=ironic \  
12. --deploy-interface direct \  
13. --network-interface noop \  
14. --instance-info image_source=http://localhost:80/img.qcow2 \  
15. --instance-info  
image_checksum=d0131bc80bc15a941318edcfb \  
16. --instance-info image_type= partition \  
17. --instance-info root_gb=10g \  
18. --driver-info deploy_kernel=boot/vmlinuz-5.10.0-29-amd64 \  
19. --driver-info deploy_ramdisk=boot/initrd.img-5.10.0-29-  
amd64
```

## 10. Triển khai các máy chủ vật lý

Để triển khai máy chủ vật lý, có thể thực hiện theo các bước sau đây

Bước 1: Xác thực

```
$ openstack baremetal node validate <tên node>
```

```
root@ironic-controller:~# openstack baremetal node validate test
```

| Interface  | Result | Reason |
|------------|--------|--------|
| bios       | True   |        |
| boot       | True   |        |
| console    | True   |        |
| deploy     | True   |        |
| inspect    | True   |        |
| management | True   |        |
| network    | True   |        |
| power      | True   |        |
| raid       | True   |        |
| rescue     | True   |        |
| storage    | True   |        |

Hình 10.1 Xác thực tất cả các tham số đều chính xác

Bước 2: Quản lý, cung cấp phần cứng và tiến hành triển khai node

1. \$ openstack baremetal node manage test
2. \$ openstack baremetal node provide test
3. \$ openstack baremetal node deploy test

Bước 3: Kiểm tra trạng thái hoạt động của nút máy vừa triển khai

```
$ openstack baremetal node list
```

```
root@ironic-controller:~# openstack baremetal node list
```

| UUID                                 | Name      | Instance UUID | Power State | Provisioning State | Maintenance |
|--------------------------------------|-----------|---------------|-------------|--------------------|-------------|
| ec1dce8e-e01d-43e8-a336-6dd97d817126 | testing   | None          | None        | enroll             | False       |
| b802b977-103f-4fd0-a3ce-f321fe5daef4 | may1      | None          | None        | enroll             | False       |
| 645955b7-0fbb-4eb0-b559-209baa39b7bc | may3      | None          | None        | enroll             | False       |
| f971f76b-045b-41c4-a9bb-7cd787205ea7 | node      | None          | power on    | active             | False       |
| 5dd02580-bd06-4ca9-9ef8-c135fbdff49a | test-node | None          | power on    | deploy             | False       |
| 1ad90a89-5dd0-4261-8e73-8191b3ff2e8d | may4      | None          | power on    | deploy             | False       |
| a713c3b0-98cd-4cbf-9274-a4752c1a7624 | test      | None          | power on    | active             | False       |

Hình 10.2 Danh sách trạng thái hoạt động của các nút

## 11. Chương trình tính toán song song đơn giản

Dưới đây là chương trình Python thực hiện phép tính nhân 2 ma trận và sử dụng MPI phục vụ tính toán song song

```
1.  from mpi4py import MPI # Import MPI module
2.  import numpy as np     # Import numpy module
3.  import time           # Import time module
4.
5.  # Định nghĩa hàm matrix_multiply với tham số là hai ma trận a và b
6.  def matrix_multiply(a, b):
7.      # Trả về kết quả của phép nhân hai ma trận a và b
8.      return np.dot(a, b)
9.
10. # Khởi tạo communicator cho các tiến trình sử dụng MPI
11. comm = MPI.COMM_WORLD
12.
13. # Lấy rank của tiến trình hiện tại
14. rank = comm.Get_rank()
15.
16. # Lấy tổng số lượng tiến trình
17. size = comm.Get_size()
18.
19. if rank == 0:
20.     # Nếu tiến trình hiện tại là tiến trình chính
21.
22.     # Tạo ma trận ngẫu nhiên a kích thước 1000x1000
23.     a = np.random.rand(1000, 1000)
24.
25.     # Tạo ma trận ngẫu nhiên b kích thước 1000x1000
26.     b = np.random.rand(1000, 1000)
27.
28.     # Phân chia ma trận a thành các phần nhỏ cho từng tiến trình
29.     chunks = np.array_split(a, size, axis=0)
30.
31. else:
32.     # Nếu tiến trình hiện tại không phải là tiến trình chính
33.
34.     # Gán a bằng None
35.     a = None
36.
37.     # Gán b bằng None
38.     b = None
```

```

39. # Gán chunks bằng None
40. chunks = None
41. # Phân phối các phần nhỏ của ma trận a đến tất cả các tiến trình
42. local_a = comm.scatter(chunks, root=0)
43.
44. # Broadcast ma trận b đến tất cả các tiến trình
45. b = comm.bcast(b, root=0)
46.
47. # Ghi lại thời gian bắt đầu
48. start_time = time.time()
49.
50. # Thực hiện phép nhân ma trận local_a với ma trận b
51. local_c = matrix_multiply(local_a, b)
52.
53. # Ghi lại thời gian kết thúc
54. end_time = time.time()
55.
56. # Thu thập kết quả từ tất cả các tiến trình về tiến trình chính
57. result = comm.gather(local_c, root=0)
58.
59. if rank == 0:
60.     # Nếu tiến trình hiện tại là tiến trình chính
61.
62.     # Kết hợp các kết quả thu thập được vào ma trận c
63.     c = np.vstack(result)
64.
65.     # In ra thông báo kết quả đã được tính
66.     print("Kết quả của phép nhân ma trận đã được tính.")
67.
68.     # In ra thời gian tính toán
69.     print(f"Thời gian tính toán: {end_time - start_time:.2f} giây")
70. else:
71.     # Nếu tiến trình hiện tại không phải là tiến trình chính
72.
73.     # In ra thời gian tính toán của từng tiến trình
74.     print(f"Thời gian tính toán của tiến trình {rank}: {end_time - start_time:.2f} giây")

```

Trong đó, chi tiết sử dụng của các câu lệnh trong thư viện MPI:

**'com = MPI.COMM\_WORLD'** tạo ra một communicator chứa tất cả các tiến trình.

**'rank = comm.Get\_rank'** và **'size = comm.Get\_size()'** lấy chỉ số và số lượng của tiến trình hiện tại.

**'com.scatter(chunks, root=0)'** phân phối các phần của ma trận 'a' đến tất cả các tiến trình.

**'com.bcast(b, root=0)'** gửi ma trận 'b' đến tất cả các tiến trình.

**'com.gather(local\_c, root=0)'** thu thập kết quả từ tất cả các tiến trình về tiến trình chính.