

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG



BÁO CÁO ĐỒ QUẢN TRỊ MẠNG VÀ HỆ THỐNG

-&-

ĐỀ TÀI: TÌM HIỂU VÀ TRIỂN KHAI
CÔNG CỤ ẢO HOÁ KVM

Giảng viên hướng dẫn: **Đỗ Hoàng Hiển**

Thực hiện bởi Nhóm 6, gồm:

• LẠI QUAN THIÊN	22521385	Trưởng nhóm
• MAI NGUYỄN NAM PHƯƠNG	22521164	Thành viên
• LÊ MINH QUÂN	22521181	Thành viên
• HỒ DIỆP HUY	22520541	Thành viên
• ĐẶNG ĐỨC TÀI	22521270	Thành viên

Lớp: NT132.P12.ANTT

TP. Hồ Chí Minh, tháng 11 năm 2024

LỜI MỞ ĐẦU

Trong bối cảnh công nghệ thông tin ngày càng phát triển, ảo hóa (virtualization) đã trở thành một giải pháp quan trọng giúp tối ưu hóa việc sử dụng tài nguyên, cải thiện hiệu suất hệ thống và giảm chi phí vận hành. Trong số các công cụ ảo hóa hiện nay, KVM (Kernel-based Virtual Machine) được đánh giá là một trong những nền tảng mạnh mẽ và linh hoạt nhất, được tích hợp trực tiếp vào nhân Linux. KVM không chỉ hỗ trợ các môi trường ảo hóa đa dạng mà còn cung cấp hiệu suất vượt trội, đáp ứng nhu cầu của các doanh nghiệp, tổ chức và cá nhân.

Tuy nhiên, việc triển khai và quản lý hệ thống KVM đòi hỏi người dùng phải nắm vững kiến thức về cấu hình mạng, phân bổ tài nguyên và bảo mật. Điều này tạo ra không ít thách thức cho những người mới bắt đầu hoặc các đơn vị chưa có kinh nghiệm về ảo hóa. Với mục tiêu khai thác tiềm năng của KVM, chúng em đã lựa chọn đề tài "*Tìm hiểu và triển khai KVM*" để tìm hiểu sâu hơn về công cụ này, đồng thời cung cấp một hướng dẫn cụ thể giúp người dùng tiếp cận và triển khai KVM một cách dễ dàng.

Qua quá trình khảo sát và nghiên cứu, chúng em nhận thấy rằng việc áp dụng KVM trên nền tảng nhân Linux không chỉ mở rộng khả năng ảo hóa mà còn tạo điều kiện thuận lợi để phát triển các môi trường thử nghiệm an toàn, phục vụ mục đích học tập, nghiên cứu và kiểm thử bảo mật. Đề tài tập trung vào việc tìm hiểu lý thuyết về KVM, triển khai cài đặt hệ thống KVM và minh họa qua các ví dụ thực tế như tạo máy ảo, triển khai 1 server cơ sở dữ liệu MySQL, deploy 1 Web App lên server là máy ảo kvm, kết nối máy ảo bằng ssh.

Chúng em hy vọng rằng đề tài này sẽ mang lại những giá trị thiết thực, góp phần hỗ trợ cộng đồng IT và các nhà nghiên cứu trong việc khai thác tối đa lợi ích của KVM, đồng thời trang bị thêm kiến thức để sẵn sàng đối mặt với các thách thức trong thời đại số hóa.

LỜI CẢM ƠN

Trước tiên, chúng em xin gửi lời cảm ơn sâu sắc đến thầy Đỗ Hoàng Hiển - giảng viên hướng dẫn môn Quản trị mạng và Hệ thống, đã tận tình chỉ dẫn và chia sẻ những kinh nghiệm quý báu trong suốt quá trình thực hiện đề tài. Những ý kiến đóng góp và sự hướng dẫn của thầy đã giúp chúng em rất nhiều trong việc định hướng và hoàn thiện nội dung đồ án.

Chúng em cũng xin gửi lời cảm ơn chân thành đến các thành viên trong nhóm, những người đã không ngừng nỗ lực, phối hợp và đóng góp ý kiến để hoàn thành đề tài một cách tốt nhất. Dù gặp nhiều khó khăn và giới hạn về thời gian, nhưng nhờ sự đoàn kết và tinh thần trách nhiệm, chúng em đã vượt qua và đạt được mục tiêu đề ra.

Cuối cùng, chúng em xin bày tỏ lòng biết ơn đến những người đã hỗ trợ và động viên chúng em trong suốt quá trình thực hiện đề tài. Do giới hạn về kiến thức và thời gian, không thể tránh khỏi những thiếu sót trong đồ án, chúng em rất mong nhận được sự góp ý từ thầy và các bạn để hoàn thiện hơn.

TP. Hồ Chí Minh, tháng 11 năm 2024

Nhóm 6, lớp NT132.P12.ANTT

MỤC LỤC

CHƯƠNG 1. TỔNG QUAN ĐỀ TÀI	1
1.1. Khảo sát hiện trạng.....	1
1.1.1. Xu hướng.....	1
1.1.2. Vai trò của KVM	1
1.2. Phạm vi đề tài	1
1.2.1. Tìm hiểu lí thuyết	1
1.2.2. Thực hành triển khai các kịch bản.....	1
1.2.3. Thực hiện đánh giá	2
CHƯƠNG 2. CHI TIẾT ĐỀ TÀI	3
2.1. Giới thiệu về Kernel Virtual Machine (KVM)	3
2.2. Các khái niệm cơ bản.....	5
2.3. Các tính năng nổi bật của KVM	6
2.4. Ứng dụng của KVM.....	7
2.5. Các vấn đề xoay quanh KVM	8
CHƯƠNG 3. SO SÁNH HIỆU SUẤT GIỮA KVM VÀ VMWARE.....	10
3.1. Mục tiêu Benchmark.....	10
3.2. Thiết lập môi trường thí nghiệm.....	11
3.3. Kịch bản kiểm tra.....	12
3.4. Phân tích kết quả.....	13
3.4.1. Benchmark 1: Truy vấn cơ sở dữ liệu	13
3.4.2. Benchmark 2: Sử dụng SSH và build ứng dụng web.....	21
CHƯƠNG 4. MÔ HÌNH, KỊCH BẢN VÀ KẾT QUẢ TRIỂN KHAI KVM.....	26
4.1. Kịch bản 1: Tạo máy ảo thông qua GUI Virt Manager.....	26
4.1.1. Ngữ cảnh và mô hình	26

4.1.2. Triển khai.....	26
4.1.3. Kết quả	30
4.1.4. Ứng dụng.....	31
4.2. Kịch bản 2: Tạo máy ảo có User Data thông qua CLI Virt Manager.	33
4.2.1. Ngữ cảnh và mô hình	33
4.2.2. Triển khai.....	37
4.2.3. Kết quả	39
4.2.4. Ứng dụng.....	40
4.3. Kịch bản 3: SSH (Secure Shell) giữa 2 máy ảo trong KVM.....	42
4.3.1. Ngữ cảnh và mô hình	42
4.3.2. Triển khai.....	42
4.3.3. Kết quả	49
4.3.4. Ứng dụng.....	49
4.4. Kịch bản 4: Triển khai CSDL MySQL và sao lưu dữ liệu của CSDL đó.....	51
4.4.1. Ngữ cảnh và mô hình	51
4.4.2. Triển khai.....	51
4.4.3. Kết quả	59
4.4.4. Ứng dụng.....	60
4.5. Kịch bản 5: Xây dựng mạng nội bộ gồm Web Server, Data Server và các Client.	61
4.5.1. Ngữ cảnh và mô hình	61
4.5.2. Triển khai.....	61
4.5.3. Kết quả	74
4.5.4. Ứng dụng.....	75
CHƯƠNG 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	76
5.1. Kết luận	76

5.2. Hạn chế.....	77
5.3. Hướng phát triển trong tương lai	77
5.4. Lời kết.....	77
DANH MỤC TÀI LIỆU THAM KHẢO	79

MỤC LỤC HÌNH ẢNH

Hình 1. Biểu đồ so sánh thời gian truy vấn CSDL giữa KVM và VMWare.....	18
Hình 2. Biểu đồ so sánh hiệu suất CPU khi truy vấn CSDL giữa KVM và VMWare	19
Hình 3. Biểu đồ so sánh hiệu suất RAM khi truy vấn CSDL giữa KVM và VMWare	19
Hình 4. Biểu đồ so sánh thời gian khi truy vấn phức tạp CSDL giữa KVM và VMWare.....	20
Hình 5. Biểu đồ so sánh thời gian khi build WebApp giữa KVM và VMWare.....	23
Hình 6. Biểu đồ so sánh hiệu suất CPU khi build WebApp giữa KVM và VMWare	23
Hình 7. Biểu đồ so sánh hiệu suất RAM khi build WebApp giữa KVM và VMWare	24
Hình 8. Biểu đồ so sánh tốc độ Mạng gửi đi khi build WebApp giữa KVM và VMWare	24
Hình 9. Biểu đồ so sánh tốc độ Mạng nhận về khi build WebApp giữa KVM và VMWare .	24
Hình 10. Kết quả băm SHA256 file iso Ubuntu Server 22.04.....	27
Hình 11. Tạo máy ảo	28
Hình 12. Chọn file ISO cần cài	28
Hình 13. Cấu hình RAM và CPU cho máy ảo	29
Hình 14. Cấp phát dung lượng bộ nhớ cho máy ảo	29
Hình 15. Xác thực lại các thông số	30
Hình 16. Giao diện dòng lệnh của máy ảo Ubuntu Server 22.04 sau khi cài đặt xong	30
Hình 17. Đèn những thông tin cần thiết cho máy ảo	37
Hình 18. Quá trình nhập thông tin thành công, giao diện Virt Manager sẽ hiện lên để thực thi phần còn lại	38
Hình 19. File Seed.iso được tạo thành công.....	38
Hình 20. Giao diện dòng lệnh để cài đặt Ubuntu Server 22.04	39
Hình 21. Quá trình cài đặt thành công	39
Hình 22. Đăng nhập vào tài khoản máy ảo	40
Hình 23. Chạy thử lệnh update apt.....	40
Hình 24. Xác định địa chỉ IP của 2 máy ảo	43
Hình 25. Kết nối đến máy ảo kia.....	43
Hình 26. Kết nối thành công	44

Hình 27. Tiến hành Clone app về	45
Hình 28. Kiểm tra dự án trên máy ảo được SSH	46
Hình 29. Apply database	47
Hình 30. Khởi chạy Server.....	47
Hình 31. Kiểm tra xem web đã chạy được chưa	48
Hình 32. WebApp đã chạy ổn định trên máy được SSH, còn máy SSH đến thì không được	48
Hình 33. Giao diện của Web.....	49
Hình 34. Kiểm tra các file đã được ký đúng hay chưa.....	52
Hình 35. Cấu hình bind-address cho MySQL	53
Hình 36. Kích hoạt SSL/TLS cho MySQL	54
Hình 37. Phân quyền user trong MySQL	55
Hình 38. Kiểm tra user đã tồn tại chưa	55
Hình 39. Chuyển file từ Database Server sang Backup Server	56
Hình 40. Backup Server thực hiện kết nối vào CSDL	56
Hình 41. Backup Server có quyền thấy toàn bộ CSDL và bảng của Database Server	57
Hình 42. Các giá trị bên trong CSDL và bảng cũng được cập nhật theo thời gian thực.....	57
Hình 43. Kiểm thử kết quả	58
Hình 44. Tạo thư mục chứa mã nguồn	62
Hình 45. Sao chép mã nguồn vào thư mục /opt/myproject.....	62
Hình 46. Thiết lập môi trường ảo Python để quản lý các gói độc lập.....	62
Hình 47. Cài đặt các gói cần thiết	63
Hình 48. Cài đặt các gói cần thiết	63
Hình 49. Cấu hình MiddleWare	63
Hình 50. Cấu hình Static files và Media files	63
Hình 51. Thu thập tất cả static files của dự án	64
Hình 52. Kiểm tra phiên bản Nginx	64
Hình 53. Kích hoạt cấu hình Nginx.....	65
Hình 54. Xử lý tệp miền.....	65

Hình 55. Cấu hình tường lửa.....	66
Hình 56. Khởi chạy thử Server	66
Hình 57. Cấu hình tên miền nội bộ	66
Hình 58. Tạo chứng chỉ tự ký.....	67
Hình 59. Kiểm tra chứng chỉ.....	68
Hình 60. Web được khởi chạy thành công với giao thức HTTPS.....	68
Hình 61. Cấu hình bind-address của MySQL	70
Hình 62. Cấu hình settings.py của dự án.....	71
Hình 63. Cấu hình Database cho WebApp.....	71
Hình 64. Xem CSDL bên phía Web Server.....	72
Hình 65. Giao diện của Web với dữ liệu đã được load lên	73
Hình 66. Thao tác với CSDL thành công	73

CHƯƠNG 1. TỔNG QUAN ĐỀ TÀI

1.1. Khảo sát hiện trạng

1.1.1. Xu hướng

Ảo hóa là một công nghệ cốt lõi trong các trung tâm dữ liệu và hạ tầng điện toán đám mây. Vì thế xu hướng sử dụng ảo hóa hiện nay đang thay đổi cách các tổ chức và doanh nghiệp quản lý, triển khai, và tối ưu hóa tài nguyên công nghệ thông tin. Các công nghệ ảo hóa ngày càng phát triển để đáp ứng nhu cầu linh hoạt, hiệu quả, và bảo mật trong bối cảnh công nghệ thay đổi nhanh chóng. Các công nghệ phổ biến bao gồm VMware, Microsoft Hyper-V, Xen và KVM.

1.1.2. Vai trò của KVM

KVM, nhờ tính mã nguồn mở và tích hợp chặt chẽ với nhân Linux, đang chiếm một phần đáng kể trong lĩnh vực hạ tầng đám mây (được sử dụng rộng rãi trong OpenStack và Proxmox). Ngoài ra KVM còn nhận được sự hỗ trợ mạnh mẽ từ cộng đồng mã nguồn mở quốc tế và trong nước. Nhiều tài liệu hướng dẫn, diễn đàn thảo luận đã có sẵn để hỗ trợ người dùng triển khai và tối ưu hóa KVM.

1.2. Phạm vi đề tài

1.2.1. Tìm hiểu lí thuyết

- Hiểu về cơ chế hoạt động của KVM.
- Nắm vững các khái niệm cơ bản như ảo hóa phần cứng, hypervisor, và cách KVM tận dụng các công nghệ.

1.2.2. Thực hành triển khai các kịch bản

- Cấu hình và tạo máy ảo thông qua giao diện đồ họa Virtual Manager.
- Tạo máy ảo chứa User Data thông qua Command Line Application (CLI).
- SSH đến một máy ảo KVM trong mạng NAT để build một ứng dụng web.
- Sử dụng máy ảo KVM để deploy database và backup dữ liệu trong mạng NAT.
- Sử dụng máy ảo KVM deploy web application, trong đó có thêm 1 máy ảo Webserver và 1 máy ảo Database Server trong mạng NAT.

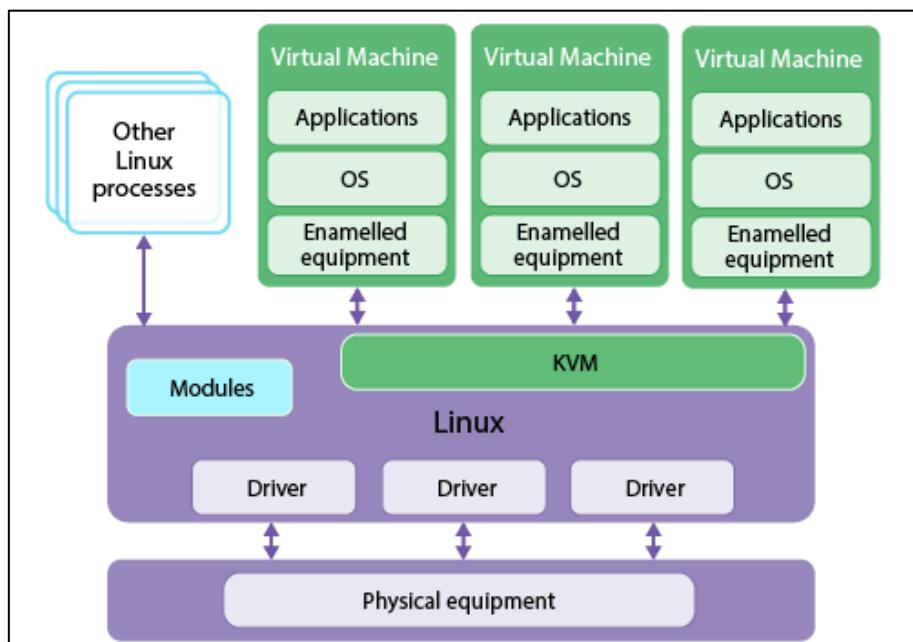
1.2.3. Thực hiện đánh giá

- Đánh giá hiệu suất của KVM
- So sánh hiệu suất KVM với VMWare
- Phân tích ưu, nhược điểm và các tính năng tiêu biểu của KVM.

CHƯƠNG 2. CHI TIẾT ĐỀ TÀI

2.1. Giới thiệu về Kernel Virtual Machine (KVM)

KVM (Kernel-based Virtual Machine) là một giải pháp ảo hóa mã nguồn mở, được tích hợp trực tiếp vào nhân Linux. Ra mắt vào năm 2007, KVM đã mở ra một kỷ nguyên mới trong lĩnh vực ảo hóa, cho phép hệ điều hành Linux hoạt động như một hypervisor (phần mềm giám sát máy ảo). KVM giúp chuyển đổi kernel Linux thành một nền tảng quản lý và triển khai máy ảo (Virtual Machines - VM), nơi mà mỗi máy ảo có thể hoạt động độc lập như một hệ thống hoàn chỉnh. Các máy ảo này có thể chia sẻ tài nguyên phần cứng như CPU, RAM, ổ cứng và mạng với các hệ điều hành khác chạy trên cùng một máy chủ vật lý, nhưng vẫn đảm bảo sự cô lập giữa chúng.



Hình ảnh Minh họa KVM

KVM được triển khai dưới dạng một module trong nhân Linux, tận dụng các tính năng ảo hóa phần cứng (Intel VT-x hoặc AMD-V) của CPU để cải thiện hiệu suất và độ ổn định. Thay vì xây dựng một hypervisor riêng biệt, KVM biến Linux thành một hypervisor loại 1 (Type 1 Hypervisor), cho phép ảo hóa được tích hợp ngay trong hệ điều hành.

- KVM cung cấp khả năng quản lý tài nguyên hiệu quả, đảm bảo việc phân bổ tài nguyên CPU, RAM và ổ đĩa cứng một cách công bằng giữa các máy ảo.

- Khi kết hợp với QEMU (Quick Emulator), KVM tận dụng tối đa các tính năng giả lập phần cứng, đồng thời cải thiện hiệu năng thông qua cơ chế tăng tốc phần cứng.

- Các công cụ như **libvirt** và **Virt-Manager** được sử dụng để quản lý, triển khai và giám sát các máy ảo một cách dễ dàng, từ môi trường dòng lệnh đến giao diện đồ họa.

Các ưu điểm của KVM so với các công cụ ảo hóa khác:

- Nhờ tích hợp trực tiếp vào nhân Linux và tận dụng công nghệ ảo hóa phần cứng, KVM đạt được hiệu suất gần tương đương với hệ thống vật lý (bare-metal).

- KVM được phát hành theo giấy phép GNU GPL, mang lại sự linh hoạt và khả năng tùy chỉnh cho các nhà phát triển và quản trị viên hệ thống mà không phải trả phí bản quyền.

- KVM hỗ trợ nhiều hệ điều hành như Linux, Windows, macOS, FreeBSD và Solaris, giúp người dùng triển khai linh hoạt nhiều loại máy ảo trên cùng một nền tảng.

- Là một phần của mã nguồn mở Linux, KVM được hưởng lợi từ tất cả các cải tiến, bản vá lỗi và cập nhật bảo mật của nhân Linux mà không cần nỗ lực phát triển thêm.

- Tài nguyên như RAM, CPU và ổ cứng được phân bổ cố định cho từng máy ảo, không bị chia sẻ hay ảnh hưởng bởi các máy ảo khác. Điều này đặc biệt quan trọng trong các môi trường như Virtual Private Server (VPS), giúp đảm bảo hiệu suất và độ ổn định cao hơn.

Yêu cầu phần cứng và phần mềm để triển khai KVM:

- Phần cứng:

+ CPU hỗ trợ công nghệ ảo hóa phần cứng (Intel VT-x hoặc AMD-V).

+ Bộ nhớ RAM và dung lượng ổ đĩa đủ lớn để đáp ứng nhu cầu của các máy ảo.

+ Card mạng hỗ trợ các tính năng như bridge hoặc NAT để cấu hình mạng ảo.

- Phần mềm:

+ Hệ điều hành Linux với kernel hỗ trợ KVM (các bản phân phối phổ biến như Ubuntu, CentOS, Debian, hoặc Fedora đều có tích hợp sẵn).

+ Các công cụ quản lý máy ảo, bao gồm:

- **QEMU:** Cung cấp khả năng giả lập phần cứng và tăng tốc hiệu năng.

- **Libvirt:** Thư viện và API để quản lý các tài nguyên ảo hóa.

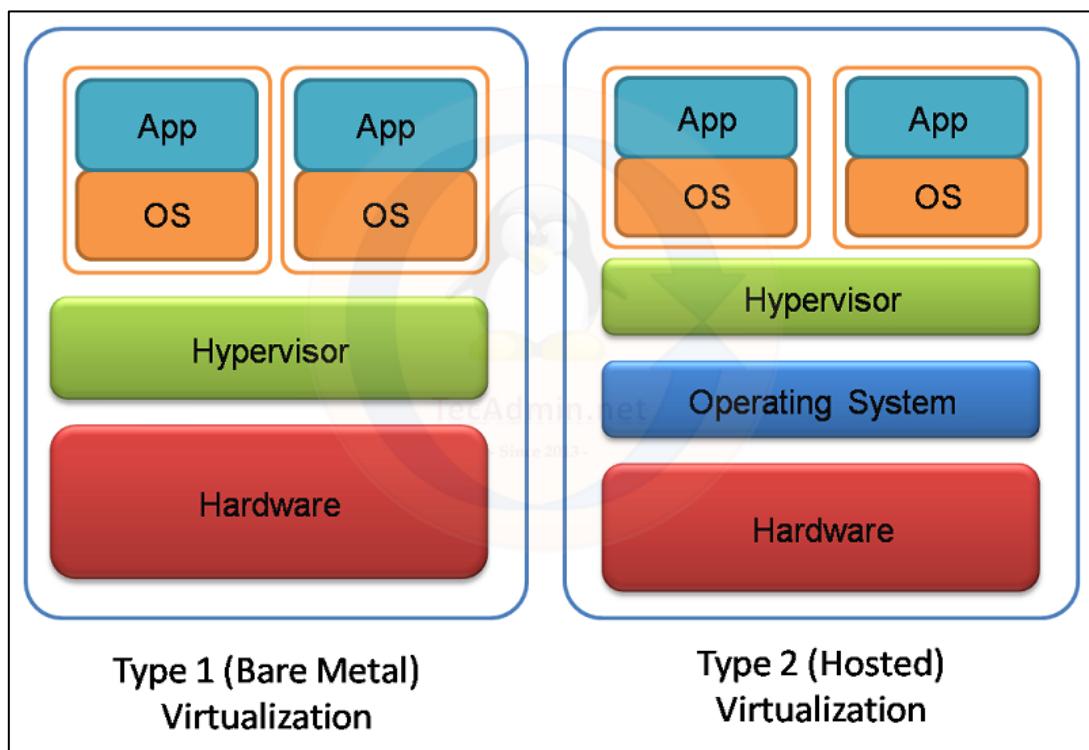
- **Virt-Manager:** Giao diện đồ họa giúp quản trị viên dễ dàng theo dõi và kiểm soát máy ảo.

2.2. Các khái niệm cơ bản

Áo hóa được chia thành hai loại chính dựa trên cách triển khai hypervisor:

- **Hypervisor Type 1 (Bare-Metal Hypervisor):** Hypervisor Type 1 chạy trực tiếp trên phần cứng vật lý của máy chủ mà không cần thông qua hệ điều hành trung gian. Loại này cung cấp hiệu suất cao nhờ giảm thiểu độ trễ và sự can thiệp từ các tầng phần mềm khác. KVM, mặc dù phụ thuộc vào nhân Linux, vẫn được xếp vào Type 1 do nó hoạt động như một module tích hợp trong kernel Linux. Một số ví dụ phổ biến khác của hypervisor Type 1 là VMware ESXi, Microsoft Hyper-V, và Xen.

- **Hypervisor Type 2 (Hosted Hypervisor):** Hypervisor Type 2 hoạt động trên một hệ điều hành chủ (host OS). Loại này dễ triển khai hơn nhưng có hiệu suất thấp hơn do phụ thuộc vào hệ điều hành trung gian. Đây thường là lựa chọn của cá nhân hoặc các tổ chức nhỏ, phù hợp để thử nghiệm hoặc phát triển. Các ví dụ điển hình bao gồm VMware Workstation, Oracle VirtualBox và Parallels Desktop.



Hypervisor Type 1 và Hypervisor Type 2

So sánh giữa KVM (Type 1) và VMware (Type 2):

- KVM (Type 1): KVM hoạt động như một hypervisor Type 1, tận dụng nhân Linux để quản lý các máy ảo. Điều này giúp KVM đạt được hiệu suất cao và khả năng cài đặt tài nguyên tốt. KVM thích hợp để triển khai trong môi trường doanh nghiệp hoặc đám mây với sự hỗ trợ mạnh mẽ từ các công cụ mã nguồn mở như Virt-Manager, libvirt và QEMU.

- VMware (Type 2): VMware Workstation là một hypervisor Type 2, được cài đặt trên một hệ điều hành chủ. VMware thường dễ sử dụng và phù hợp để thử nghiệm hoặc chạy các ứng dụng không yêu cầu hiệu suất cao. Tuy nhiên, hiệu suất của VMware thấp hơn KVM do phụ thuộc vào hệ điều hành trung gian và tài nguyên bị phân chia thêm một tầng.

Các thành phần chính của KVM:

- libvirt: Libvirt là một thư viện mã nguồn mở quản lý ảo hóa, cung cấp các API để tạo, quản lý và giám sát máy ảo. Libvirt hỗ trợ nhiều hypervisor, nhưng với KVM, nó đóng vai trò quan trọng trong việc quản lý tài nguyên. Công cụ dòng lệnh virsh đi kèm với libvirt giúp quản trị viên thực hiện các tác vụ như khởi động, dừng hoặc xóa máy ảo.

- Virt-Manager: Virt-Manager là giao diện đồ họa sử dụng libvirt để quản lý KVM. Công cụ này cung cấp khả năng giám sát, chỉnh sửa và quản lý các máy ảo một cách trực quan. Virt-Manager hỗ trợ các tính năng cao cấp như tạo snapshot, cấu hình mạng ảo, và theo dõi tài nguyên theo thời gian thực.

- QEMU: QEMU là công cụ giả lập phần cứng được sử dụng cùng với KVM. Khi kết hợp, QEMU tận dụng khả năng tăng tốc phần cứng từ KVM (Intel VT-x hoặc AMD-V), giúp các máy ảo hoạt động với hiệu suất gần như tương đương với hệ thống vật lý. QEMU chịu trách nhiệm giả lập các thiết bị phần cứng như CPU, ổ cứng, RAM và các thiết bị ngoại vi.

2.3. Các tính năng nổi bật của KVM

Ảo hóa tốt phần cứng, tối ưu cho việc tạo và quản lý nhiều máy ảo: KVM tận dụng khả năng ảo hóa phần cứng từ các bộ xử lý hỗ trợ VT-x (Intel) và AMD-V (AMD), cho phép tạo và quản lý các máy ảo với hiệu suất gần với phần cứng thật. Điều này giúp người dùng dễ dàng tạo và cấu hình máy ảo mà không cần sử dụng nhiều tài nguyên.

Hiệu suất I/O được tối ưu hóa, phân bổ tài nguyên động tốt: KVM sử dụng công nghệ VirtIO để tối ưu hóa hiệu suất I/O cho máy ảo. Với VirtIO, tốc độ đọc/ghi ổ cứng và truy cập mạng được cải thiện, giúp việc triển khai và quản lý server cơ sở dữ liệu MySQL trên máy ảo KVM trở nên hiệu quả hơn.

Hỗ trợ mạng ảo hóa, tích hợp với các công cụ quản lý hệ thống: KVM cung cấp nhiều chế độ mạng ảo hóa, chẳng hạn như NAT, Bridged, và Host-only. Khi triển khai một ứng dụng web, ta có thể cấu hình máy ảo để kết nối với mạng nội bộ hoặc cung cấp địa chỉ IP công khai để người dùng bên ngoài có thể truy cập ứng dụng web, ngoài ra KVM còn có thể dễ dàng tích hợp với các công cụ giám sát và quản lý hệ thống như Prometheus, Nagios, hoặc Ansible, giúp quản lý và triển khai ứng dụng web trên máy ảo một cách hiệu quả và tự động hóa.

Bảo mật và linh hoạt: KVM cho phép các máy ảo kết nối qua SSH một cách an toàn, sử dụng mạng ảo hóa. Điều này cho phép quản trị viên truy cập từ xa để cấu hình, kiểm tra và quản lý tài nguyên mà không cần giao diện đồ họa. KVM còn cung cấp tính năng mạng ảo hóa mạnh mẽ, cho phép máy ảo kết nối với nhau và với mạng bên ngoài một cách dễ dàng, giúp người dùng có thể cấu hình và triển khai các máy chủ SSH mà không gặp nhiều khó khăn.

2.4. Ứng dụng của KVM

KVM đóng vai trò quan trọng trong việc xây dựng và quản lý các hệ thống mạng ảo hóa, đặc biệt là trong các trung tâm dữ liệu và môi trường doanh nghiệp.

1. Triển khai hạ tầng mạng ảo hóa:

- KVM cho phép tạo và quản lý nhiều máy ảo trên cùng một máy chủ vật lý, giúp tối ưu hóa việc sử dụng tài nguyên phần cứng.

- Các máy ảo này có thể được kết nối thông qua mạng ảo (virtual network), cho phép mô phỏng các cấu trúc mạng phức tạp mà không cần đầu tư thêm vào phần cứng vật lý.

- Nhờ tính năng cô lập tài nguyên của KVM, các mạng ảo đảm bảo tính bảo mật và ổn định, không bị ảnh hưởng bởi các máy ảo khác.

2. Hỗ trợ xây dựng môi trường phát triển và thử nghiệm:

- KVM tạo ra các máy ảo độc lập với các hệ điều hành khác nhau, giúp các nhà phát triển thử nghiệm phần mềm hoặc cấu hình mạng trong một môi trường an toàn và linh hoạt.

- Các snapshot trong KVM cho phép lưu lại trạng thái hiện tại của máy ảo, giúp dễ dàng khôi phục nếu xảy ra lỗi trong quá trình thử nghiệm.

3. Quản lý mạng ảo hóa nâng cao:

- KVM tích hợp với các công cụ như Open vSwitch hoặc các dịch vụ mạng SDN (Software Defined Networking) để tạo ra các mạng ảo hóa phức tạp hơn, phù hợp với nhu cầu của các doanh nghiệp lớn.

- Các tính năng như NAT (Network Address Translation), VLAN (Virtual LAN), và bridge networking trong KVM giúp dễ dàng triển khai các mạng ảo linh hoạt.

2.5. Các vấn đề xoay quanh KVM

1. Tính bảo mật của KVM: KVM được đánh giá cao về tính bảo mật nhờ khả năng cô lập tài nguyên và các biện pháp bảo vệ tích hợp trong nhân Linux.

- Các máy ảo chạy trên KVM được cách ly hoàn toàn khỏi nhau, đảm bảo rằng lỗi hoặc sự cố trên một máy ảo không ảnh hưởng đến các máy ảo khác hoặc hệ điều hành chủ.

- Tích hợp với SELinux và AppArmor: Cung cấp các chính sách bảo mật chi tiết để kiểm soát quyền truy cập của máy ảo vào tài nguyên hệ thống. Các chính sách này giúp ngăn chặn truy cập trái phép và giảm thiểu nguy cơ tấn công bảo mật.

- KVM tích hợp vTPM, cho phép các máy ảo sử dụng module TPM ảo để lưu trữ các thông tin bảo mật như khóa mã hóa, giúp tăng cường bảo mật dữ liệu.

- Như bất kỳ công nghệ nào khác, KVM cũng có thể bị khai thác nếu không được cấu hình đúng cách. Các lỗi bảo mật tiềm ẩn trong nhân Linux hoặc phần mềm quản lý (như QEMU) có thể trở thành mục tiêu tấn công.

2. Hiệu suất của KVM so với các giải pháp khác: KVM nổi bật với hiệu suất cao nhờ tích hợp trực tiếp vào nhân Linux và tận dụng công nghệ ảo hóa phần cứng (Intel VT-x, AMD-V).

- **Ưu điểm:** KVM mang lại hiệu suất gần như tương đương với hệ thống vật lý khi sử dụng các tính năng ảo hóa phần cứng.

- **Nhược điểm:** Hiệu suất của KVM phụ thuộc vào khả năng cấu hình của quản trị viên. Một hệ thống cấu hình kém có thể dẫn đến sự thiếu tối ưu trong việc phân bổ tài nguyên.

- **Khả năng mở rộng:** KVM được thiết kế để hoạt động tốt trong các môi trường quy mô lớn, từ trung tâm dữ liệu đến đám mây.

- **Khả năng mở rộng:** KVM hỗ trợ tính năng *live migration*, cho phép di chuyển máy ảo giữa các máy chủ mà không cần tắt máy ảo, đảm bảo tính liên tục của dịch vụ. Khả năng quản lý hàng nghìn máy ảo đồng thời giúp KVM trở thành một trong những lựa chọn hàng đầu cho các giải pháp đám mây lớn như OpenStack.

- **Tính linh hoạt:** KVM hỗ trợ nhiều hệ điều hành khác nhau như Linux, Windows, và các hệ điều hành Unix-like, mang lại sự linh hoạt trong việc triển khai. Khả năng tích hợp với các công cụ quản lý mã nguồn mở như Proxmox, OpenStack, và các công cụ lưu trữ mạng như Ceph, giúp KVM dễ dàng triển khai trong nhiều môi trường khác nhau.

- **Thách thức:** Việc mở rộng hạ tầng với KVM yêu cầu đội ngũ quản trị viên có kỹ năng chuyên sâu để đảm bảo hiệu suất và độ ổn định. Tích hợp với các công nghệ khác như container (Docker, Kubernetes) vẫn cần tối ưu hóa thêm để đạt hiệu quả cao nhất.

CHƯƠNG 3. SO SÁNH HIỆU SUẤT GIỮA KVM VÀ VMWARE

3.1. Mục tiêu Benchmark

Đánh giá hiệu năng giữa hai phương pháp ảo hóa: **KVM** trên hệ điều hành **Kali Linux 2024 (Dual-Boot)** và **VMware Workstation Pro** trên hệ điều hành **Windows 11 23H2**.

Thông qua đó ta có thể xác định khả năng tận dụng tài nguyên hệ thống trong từng môi trường, và rút ra nhận định về ưu và nhược điểm của hai phương pháp triển khai.

Các tiêu chí đo lường hiệu năng:

1. CPU:

- Đánh giá hiệu suất xử lý tác vụ nặng thông qua các phép toán phức tạp và khả năng xử lý đa nhiệm.

- So sánh thời gian hoàn thành tác vụ và hiệu năng khi chạy đồng thời nhiều tiến trình trên cả hai môi trường.

2. RAM:

- So sánh mức tiêu thụ RAM khi chạy các ứng dụng có cùng cấu hình.

- Đánh giá hiệu suất truy xuất bộ nhớ và khả năng xử lý khi RAM được sử dụng ở mức tối đa.

3. I/O Disk:

- So sánh tốc độ đọc/ghi dữ liệu trên ổ đĩa trong các kịch bản đọc/ghi tuần tự và ngẫu nhiên.

- Đo hiệu năng với các tác vụ thực tế như sao chép tệp lớn hoặc truy xuất dữ liệu từ cơ sở dữ liệu nhỏ.

3. I/O Network:

- Đánh giá hiệu suất truyền tải dữ liệu qua mạng trong môi trường LAN và WAN.

- So sánh băng thông, độ trễ, và mức độ ổn định của kết nối mạng giữa hai môi trường.

4. Thời gian khởi động: So sánh thời gian từ khi khởi động hệ điều hành đến khi môi trường KVM hoặc các dịch vụ cần thiết sẵn sàng hoạt động.

Kỳ vọng: Qua việc đo lường và phân tích các tiêu chí hiệu năng, nhóm sẽ đánh giá được ưu thế nổi bật của từng môi trường triển khai KVM (Dual-Boot và VMware). Từ đó,

đưa ra các khuyến nghị phù hợp, hỗ trợ cho việc lựa chọn môi trường triển khai tối ưu nhất tùy thuộc vào yêu cầu thực tế của hệ thống.

3.2. Thiết lập môi trường thí nghiệm

Cấu hình phần cứng: Thí nghiệm của nhóm được thực hiện trên cùng một máy laptop **Lenovo Gaming LOQ 2024**, với cấu hình chi tiết:

CPU	<u>Intel Core i5-12450HX</u> – 8 nhân (4P-core, 4E-core), 12 luồng
RAM	24GB DDR5
Lưu trữ	512GB SSD NVMe PCIe
GPU	<u>Nvidia RTX 3050 6GB</u>
Chuẩn Wifi	Wifi 6
CPU	<u>Intel Core i5-12450HX</u> – 8 nhân (4P-core, 4E-core), 12 luồng
Điều kiện thí nghiệm	Nhiệt độ phòng được duy trì ổn định ở mức 26°C. Sau mỗi kịch bản thí nghiệm, hệ thống được nghỉ 30 phút để hạ nhiệt.

Hệ điều hành:

Dual-Boot Kali Linux 2024.3	Windows 11 Home 23H2:
- Sử dụng hệ điều hành Kali Linux 2024.3 cùng với công cụ KVM làm môi trường ảo hóa. - Máy ảo được cài đặt để thí nghiệm là Ubuntu Server 22.04.	- Sử dụng hệ điều hành Windows 11 Home 23H2 với VMware Workstation Pro 17.5 làm môi trường ảo hóa. - Máy ảo được cài đặt để thí nghiệm là Ubuntu Server 22.04.

Cấu hình phân bổ tài nguyên:

- **Bên Windows 11:** Các máy ảo Ubuntu Server 22.04 được tạo trên **VMware Workstation** với cùng cấu hình: **1 Processor, 4 core CPU, 4GB RAM, 64GB Disk.** Còn các thông số còn lại để mặc định:

- **Bên Kali Linux 2024.3:** Các máy ảo Ubuntu Server 22.04 được tạo trên môi trường KVM với cùng cấu hình: **1 Processor, 4 core CPU, 4GB RAM, 64GB Disk.** Còn các thông số còn lại để mặc định:

- Điều kiện mạng: Tất cả các kịch bản benchmark đều sử dụng chung một kết nối Wifi 6, mạng NAT, băng thông tối đa **200Mbps**, đảm bảo các máy ảo truy cập mạng dưới cùng điều kiện.

3.3. Kịch bản kiểm tra

Để đánh giá hiệu suất trên hai môi trường triển khai (KVM và VMware), nhóm chúng em thực hiện 02 kịch bản kiểm tra như sau:

- Benchmark 1: Truy vấn cơ sở dữ liệu

+ **Mô tả sơ lược:** Thực hiện các truy vấn phức tạp trên cơ sở dữ liệu với số lượng từ **100.000 bản ghi** đến **1 triệu bản ghi**.

+ Mục tiêu:

- Đo thời gian xử lý truy vấn.
- Đánh giá tốc độ sao lưu dữ liệu và khả năng sử dụng tài nguyên (CPU, RAM) trong quá trình sao lưu.

- Benchmark 2: Hiệu suất mạng/CPU khi sử dụng SSH và build ứng dụng web

+ **Mô tả sơ lược:** Từ một máy khác trong cùng mạng, sử dụng SSH để kết nối đến máy ảo KVM. Thực hiện thao tác **clone một dự án webapp từ GitHub**, sau đó tiến hành build và kiểm tra hiệu suất CPU, Network và RAM trong quá trình này.

+ Mục tiêu:

- Đo thời gian hoàn thành quá trình clone và build.
- Đánh giá mức độ tiêu thụ tài nguyên và khả năng xử lý tác vụ liên quan đến mạng.

3.4. Phân tích kết quả

3.4.1. Benchmark 1: Truy vấn cơ sở dữ liệu

a. Mô tả:

Kịch bản được thiết kế để kiểm tra khả năng xử lý truy vấn cơ sở dữ liệu phức tạp trên máy ảo Ubuntu Server 22.04. Hai môi trường thử nghiệm bao gồm:

- KVM trên Linux: Máy ảo Ubuntu Server 22.04 chạy trên KVM được cài đặt trong hệ điều hành Kali Linux 2024.3.
- VMware trên Windows: Máy ảo Ubuntu Server 22.04 chạy trên VMware Workstation trong hệ điều hành Windows 11 Home 23H2.

Các thông số được đo lường bao gồm:

- CPU: Tỉ lệ sử dụng CPU trong quá trình thực thi.
- RAM: Tỉ lệ sử dụng RAM.
- Time: Thời gian hoàn thành truy vấn (tính bằng giây).

b. Giới thiệu về các bài test:

Về đối tượng:

- Thực hiện truy vấn trên 1 Database gồm 3 bảng là staffs (1 triệu bảng ghi), products (300 000 bảng ghi) và partner (100 000 bảng ghi).
- Trong đó, staffs.id sẽ là khoá chính của bảng sfatfs, đồng thời cũng được 2 bảng products (products.manager_id) và parner (parner.manager_id) tham chiếu đến.

Chi tiết cơ sở dữ liệu:

```
CREATE DATABASE my_company;

USE my_company;

CREATE TABLE staffs (
    id VARCHAR(50) PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    sex VARCHAR(10) NOT NULL,
    phone VARCHAR(20),
    address VARCHAR(255),
    birthday DATE,
```

```

join_date DATE,
position VARCHAR(100),
email VARCHAR(100) NOT NULL,
password VARCHAR(255) NOT NULL
);

CREATE TABLE products (
    id VARCHAR(50) PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    type VARCHAR(100),
    import_date DATE,
    price DECIMAL(10, 2),
    public_date DATE,
    manager_id VARCHAR(50),
    FOREIGN KEY (manager_id) REFERENCES staffs(ID)
);

CREATE TABLE partners (
    id VARCHAR(50) PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    address VARCHAR(255),
    join_date DATE,
    manager_id VARCHAR(50),
    FOREIGN KEY (manager_id) REFERENCES staffs(ID)
);

```

Về các bài test: gồm 11 bài, độ phức tạp tăng dần từ những câu truy vấn đơn giản như hiển thị bảng, cho đến các phép toán logic như giao, hợp, kết, điều kiện,... Dưới đây là 11 bài test:

- Test 1,2,3: Truy vấn đơn giản

```

select * from staffs;
select * from products;
select * from partner;

```

- Test 4: Liệt kê sản phẩm và quản lý của sản phẩm

```

SELECT products.name AS ProductName, products.type AS ProductType,
products.price,
        staffs.name AS ManagerName, staffs.position AS ManagerPosition
FROM products
JOIN staffs ON products.manager_id = staffs.id;

```

- Test 5: Liệt kê đối tác và người quản lý

```

Liệt kê đối tác và người quản lý
SELECT partners.name AS PartnerName, partners.address,
        staffs.name AS ManagerName, staffs.position AS ManagerPosition

```

```
FROM partners
JOIN staffs ON partners.manager_id = staffs.id;
CPU 42%, Ram: 85% (VMware khong dang ke), Disk 0%, time: 59.2
```

- Test 6: Thống kê số lượng sản phẩm và giá trị trung bình của từng loại sản phẩm

```
SELECT type, COUNT(*) AS ProductCount, AVG(price) AS AveragePrice
FROM products
GROUP BY type;
```

- Test 7: Tìm nhân viên quản lý nhiều sản phẩm nhất

```
SELECT staffs.name AS ManagerName, staffs.position, COUNT(products.id) AS
ProductCount
FROM staffs
JOIN products ON staffs.id = products.manager_id
GROUP BY staffs.id
ORDER BY ProductCount DESC
LIMIT 1;
```

- Test 8: Truy vấn tổng hợp cho báo cáo hàng tháng

```
SELECT products.name AS ProductName, products.import_date,
       staffs.name AS ManagerName, partners.name AS PartnerName
FROM products
LEFT JOIN staffs ON products.manager_id = staffs.id
LEFT JOIN partners ON staffs.id = partners.manager_id
WHERE MONTH(products.import_date) = MONTH(CURDATE())
      AND YEAR(products.import_date) = YEAR(CURDATE());
```

- **Test 9: Truy vấn tổng hợp với GROUP BY và COUNT =>** Truy vấn này sẽ tính số lượng sản phẩm và đối tác mà mỗi nhân viên (manager) quản lý. Nó sẽ giúp kiểm tra tốc độ tính toán tổng hợp.

```
SELECT
    s.id AS staff_id,
    COUNT(DISTINCT p.id) AS product_count,
    COUNT(DISTINCT pr.id) AS partner_count
FROM
    staffs s
LEFT JOIN
    products p ON p.manager_id = s.id
LEFT JOIN
    partners pr ON pr.manager_id = s.id
GROUP BY
    s.id
HAVING
    product_count > 0
ORDER BY
    product_count DESC
LIMIT 100;
```

- **Test 10: Truy vấn với điều kiện HAVING và tính toán theo nhóm =>** Truy vấn này tính số lượng đối tác và sản phẩm cho mỗi nhân viên, sau đó lọc ra những nhân viên có ít nhất 10 đối tác và sản phẩm.

```
SELECT
    s.id AS staff_id,
    s.name AS staff_name,
    COUNT(DISTINCT p.id) AS product_count,
    COUNT(DISTINCT pr.id) AS partner_count
FROM
    staffs s
LEFT JOIN
    products p ON p.manager_id = s.id
LEFT JOIN
    partners pr ON pr.manager_id = s.id
GROUP BY
    s.id
HAVING
    product_count >= 10 AND partner_count >= 10
ORDER BY
    product_count DESC, partner_count DESC
LIMIT 100;
```

- Test 11: Truy vấn tìm các nhân viên có số lượng đối tác tối thiểu và tối đa

```
SELECT
    s.id AS staff_id,
    s.name AS staff_name,
    COUNT(pr.id) AS partner_count
FROM
    staffs s
LEFT JOIN
    partners pr ON pr.manager_id = s.id
GROUP BY
    s.id
HAVING
    partner_count = (SELECT MIN(partner_count) FROM (SELECT COUNT(pr.id) AS
partner_count FROM staffs s LEFT JOIN partners pr ON pr.manager_id = s.id
GROUP BY s.id) AS subquery)
    OR partner_count = (SELECT MAX(partner_count) FROM (SELECT COUNT(pr.id)
AS partner_count FROM staffs s LEFT JOIN partners pr ON pr.manager_id = s.id
GROUP BY s.id) AS subquery)
ORDER BY
    partner_count DESC;
```

c. Kết quả kiểm tra

KVM			
	CPU	RAM	Time (s)
Test 1	73%	23%	0.5
Test 2	40%	20%	0.4
Test 3	70%	23%	0.1
Test 4	60%	20%	0.6
Test 5	70%	20%	1.8
Test 6	6%	20%	0.04
Test 7	6%	20%	0.81
Test 8	4%	20%	0.06
Test 9	10%	20%	4.47
Test 10	9%	20%	4.52
Test 11	80%	25%	8.2

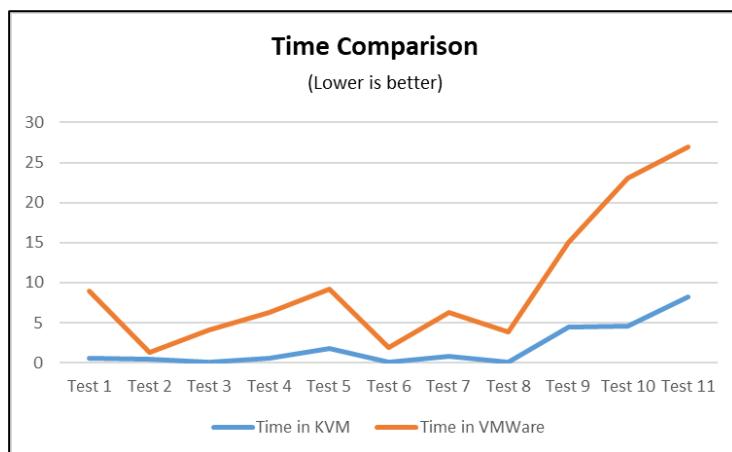
VMWare			
	CPU	RAM	Time (s)
Test 1	90%	82%	9
Test 2	50%	72%	1.34
Test 3	80%	62%	4.11
Test 4	75%	67%	6.33
Test 5	82%	72%	9.2
Test 6	14%	62%	1.84

Test 7	10%	72%	6.25
Test 8	17%	70%	3.88
Test 9	15%	67%	15
Test 10	14%	62%	23
Test 11	86%	71%	27

d. Phân tích kết quả

1. Thời gian xử lý (Time):

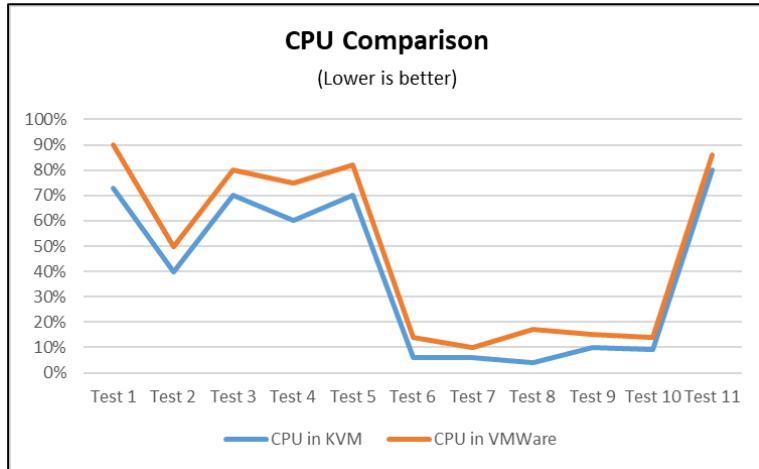
- KVM hoàn thành truy vấn nhanh hơn đáng kể trong tất cả các bài kiểm tra.
 - + Ví dụ: Test 1 mất **0.5 giây** trên KVM, nhưng trên VMWare 9 **giây**.
 - + Đặc biệt, Test 11 mất **8.2 giây** trên KVM, trong khi VMWare mất đến **27 giây**.
- Điều này cho thấy KVM có khả năng tận dụng phần cứng tốt hơn và không bị ảnh hưởng bởi lớp ảo hóa.



Hình 1. Biểu đồ so sánh thời gian truy vấn CSDL giữa KVM và VMWare

2. Hiệu suất CPU:

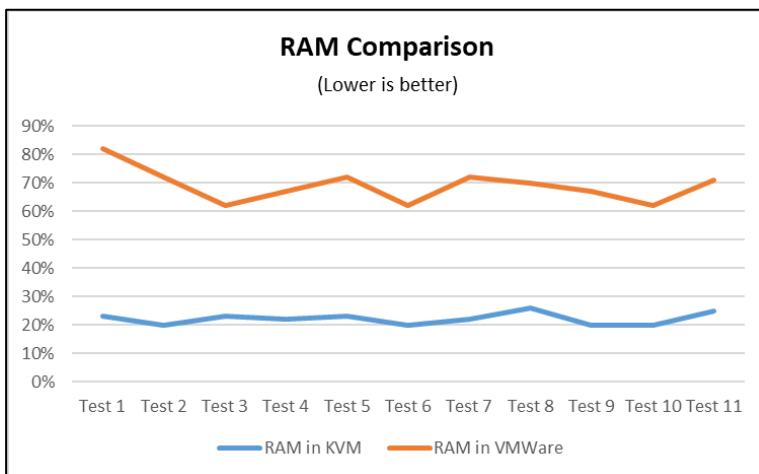
- + Trên **KVM**, CPU được sử dụng hiệu quả hơn. Ví dụ, Test 6 sử dụng **6% CPU**, trong khi trên VMWare cần **14% CPU** cho cùng bài kiểm tra.
- + VMWare thường có xu hướng sử dụng CPU cao hơn (do phải hỗ trợ lớp ảo hóa và hệ điều hành host).



Hình 2. Biểu đồ so sánh hiệu suất CPU khi truy vấn CSDL giữa KVM và VMWare

3. Hiệu suất RAM:

- + RAM trên DualBoot ổn định ở mức **20-26%** cho hầu hết các bài kiểm tra.
- + Trái lại, VMWare sử dụng lượng RAM lớn hơn khá nhiều, thường xuyên dao động ở ngưỡng **70% đến 85%**.

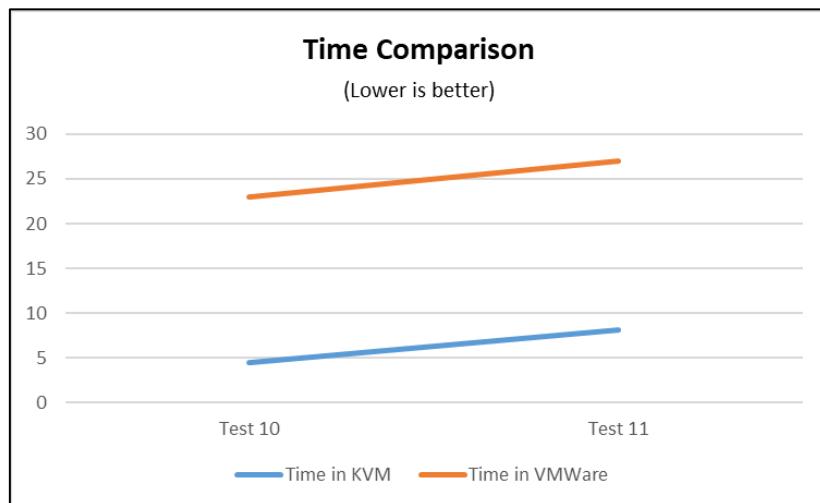


Hình 3. Biểu đồ so sánh hiệu suất RAM khi truy vấn CSDL giữa KVM và VMWare

4. Tăng trưởng thời gian ở các bài kiểm tra phức tạp

Khi truy vấn phức tạp hơn (Test 9 đến Test 11), **KVM** vẫn duy trì thời gian xử lý ổn định (dưới 10 giây), trong khi **VMWare** gặp hiện tượng gia tăng thời gian đột biến:

- Test 10: **4.52 giây** (KVM) vs **23 giây** (VMWare).
- Test 11: **8.2 giây** (KVM) vs **27 giây** (VMWare).



Hình 4. Biểu đồ so sánh thời gian khi truy vấn phức tạp CSDL giữa KVM và VMWare

e. Kết luận

- **KVM**: Là lựa chọn ưu việt cho các tác vụ xử lý nặng như truy vấn cơ sở dữ liệu phức tạp, nhờ khả năng tận dụng trực tiếp và hiệu quả tài nguyên phần cứng. Tuy nhiên, việc triển khai Dual-boot trên máy thật chạy Linux có độ phức tạp cao hơn, đòi hỏi kiến thức về hệ thống và khả năng xử lý sự cố.

- **VMware**: Dễ dàng cài đặt và sử dụng hơn trên Windows, phù hợp với các tác vụ không yêu cầu hiệu năng cao, nhưng tiêu tốn tài nguyên hơn do phụ thuộc vào lớp trung gian của phần mềm ảo hóa.

3.4.2. Benchmark 2: Sử dụng SSH và build ứng dụng web

a. Mô tả

Kịch bản được thiết kế để kiểm tra khả năng xử lý của hệ thống khi thực hiện kết nối SSH và build ứng dụng web trên Ubuntu Server 22.04. Hai môi trường thử nghiệm bao gồm:

- KVM trên Linux: Máy ảo Ubuntu Server 22.04 chạy trên KVM được cài đặt trong hệ điều hành Kali Linux 2024.3.

- VMware trên Windows: Máy ảo Ubuntu Server 22.04 chạy trên VMware Workstation trong hệ điều hành Windows 11 Home 23H2.

Các thông số được đo lường bao gồm:

- CPU: Tỷ lệ sử dụng CPU trong quá trình thực thi.
- RAM: Tỷ lệ sử dụng RAM.
- Networks (S-R): Băng thông gửi (Sent) và nhận (Received) dữ liệu trong quá trình xử lý.
- Time: Thời gian hoàn thành từng bước trong quá trình thử nghiệm (tính bằng giây).

b. Giới thiệu về các bài test

Về đối tượng: Ứng dụng được sử dụng trong kịch bản này là llama-chat, một ứng dụng web mã nguồn mở được clone từ [GitHub](#) (dung lượng: 4.17MB).

Về các bài test: Kịch bản được lặp lại 3 lần, mỗi lần bao gồm hai bài test:

- Test x.1: Cài đặt các dependency của ứng dụng thông qua lệnh npm install.
- Test x.2: Khởi chạy ứng dụng bằng lệnh npm run dev.

(với x là lần lặp thứ i với i ∈ (1,2,3))

c. Kết quả kiểm tra

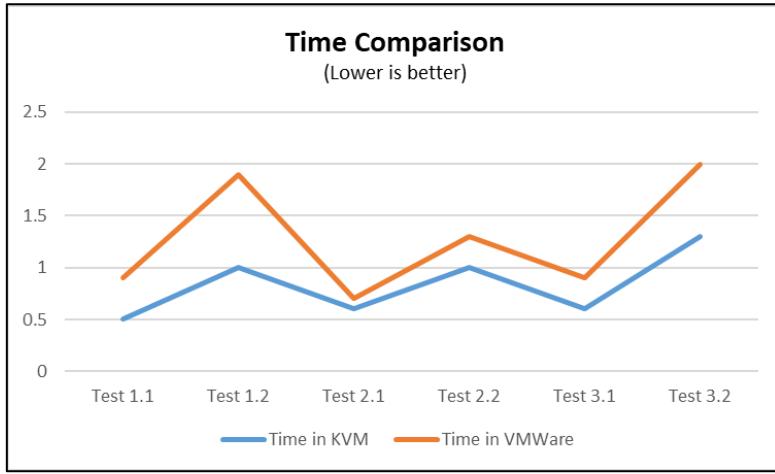
KVM					
	CPU	RAM	Networks (S-R)		Time (s)
Test 1.1	24%	20%	28Kib/s	34Kbi/s	0.5
Test 1.2	24%	23%	15Kbi/s	58Kbi/s	1
Test 2.1	24%	23%	36Kib/s	43Kbi/s	0.6
Test 2.2	26%	20%	36Kib/s	45Kbi/s	1
Test 3.1	26%	23%	28Kbi/s	32Kbi/s	0.6
Test 3.2	28%	24%	36Kib/s	80Kbi/s	1.3

VMWare					
	CPU	RAM	Networks (S-R)		Time (s)
Test 1.1	60%	52%	12Kib/s	24Kib//s	0.9
Test 1.2	40%	52%	5Kib/s	48Kbi/s	1.9
Test 2.1	54%	53%	12Kib/s	26Kbi/s	0.7
Test 2.2	44%	54%	6Kib/s	32Kbi/s	1.3
Test 3.1	51%	51%	19Kbi/s	20Kbi/s	0.9
Test 3.2	53%	52%	9Kib/s	60Kbi/s	2

d. Phân tích kết quả

1. Thời gian xử lý (Time):

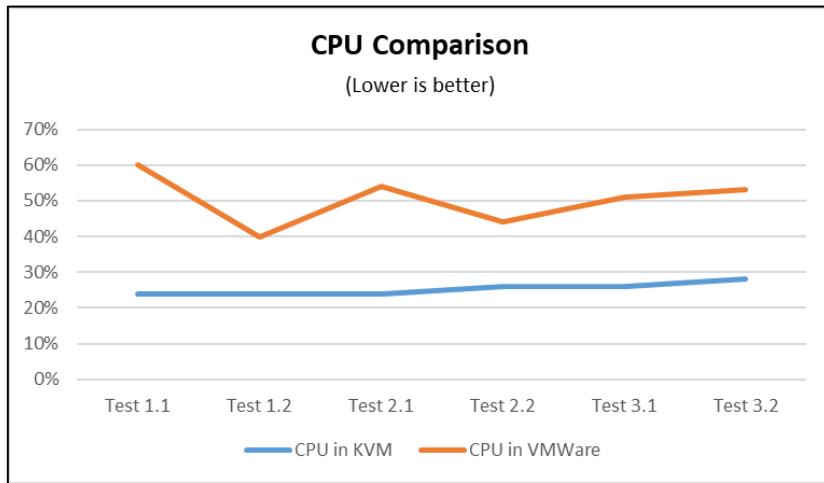
- + kvm: Thời gian hoàn thành nhanh hơn đáng kể, dao động từ **0.5 giây đến 1.3 giây**.
- + VMWare: Thời gian thực hiện lâu hơn, đặc biệt ở các bài test như 1.2 và 3.2, với thời gian lên tới **1.9 giây và 2.0 giây**.
- + Nhận xét: Hiệu suất thời gian của kvm tốt hơn do không bị ảnh hưởng bởi lớp ảo hóa.



Hình 5. Biểu đồ so sánh thời gian khi build WebApp giữa KVM và VMWare

2. Hiệu suất CPU:

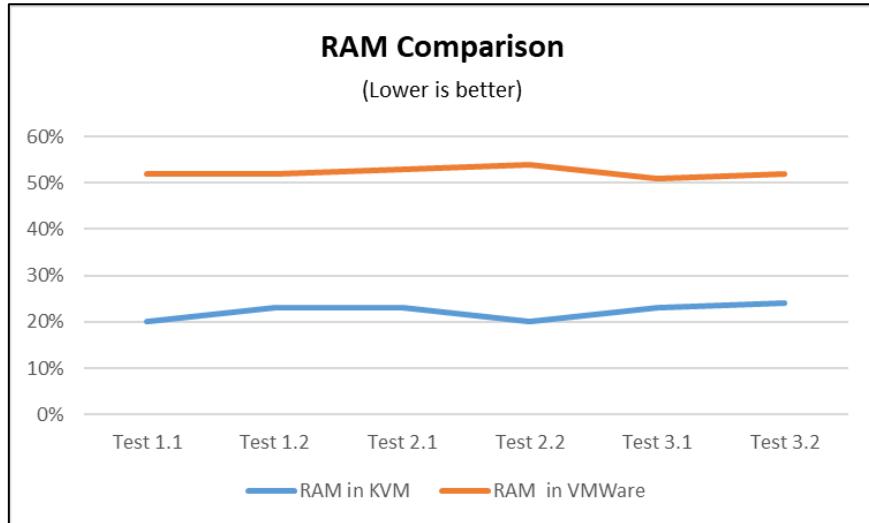
- + **KVM:** CPU hoạt động ổn định ở mức thấp, từ **24% đến 28%** trong tất cả các bài test.
- + **VMWare:** CPU bị sử dụng cao hơn đáng kể, dao động từ **40% đến 60%**.
- + **Nhận xét:** KVM quản lý tài nguyên CPU hiệu quả hơn, trong khi VMWare phải chia sẻ CPU với hệ điều hành host.



Hình 6. Biểu đồ so sánh hiệu suất CPU khi build WebApp giữa KVM và VMWare

3. Hiệu suất RAM:

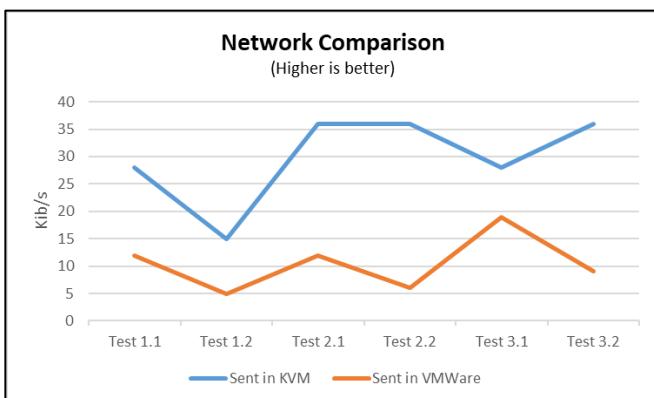
- + **KVM:** RAM sử dụng thấp, ổn định ở mức **20% đến 24%**.
- + **VMWare:** RAM tiêu thụ cao hơn, từ **51% đến 54%**.
- + **Nhận xét:** Lớp ảo hóa trong VMWare làm tăng nhu cầu RAM.



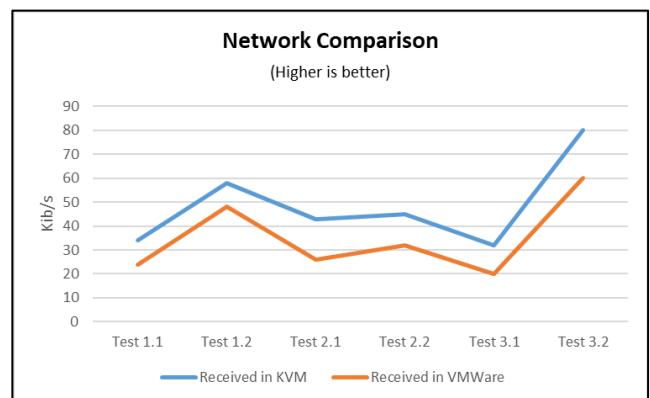
Hình 7. Biểu đồ so sánh hiệu suất RAM khi build WebApp giữa KVM và VMWare

4. Hiệu suất mạng (Networks):

- + **DualBoot:** Băng thông mạng cao và ổn định (Sent: **28-36Kib/s**, Received: **32-58Kib/s**).
- + **VMWare:** Băng thông mạng thấp hơn đáng kể (Sent: **5-19Kib/s**, Received: **20-60Kib/s**), dẫn đến việc tải và cài đặt các package lâu hơn.



Hình 8. Biểu đồ so sánh tốc độ Mạng gửi đi khi build WebApp giữa KVM và VMWare



Hình 9. Biểu đồ so sánh tốc độ Mạng nhận về khi build WebApp giữa KVM và VMWare

e. Kết luận:

Các bài test trong phần Benchmark này cho thấy rõ sự khác biệt giữa hiệu suất của DualBoot và VMWare khi thực hiện cài đặt và khởi chạy ứng dụng web.

- **KVM** tỏ ra vượt trội với hiệu suất CPU, RAM và mạng ổn định, giúp giảm thời gian thực hiện các tác vụ.

- **VMWare** mặc dù linh hoạt hơn trong việc quản lý nhiều môi trường, nhưng hiệu suất bị ảnh hưởng đáng kể bởi lớp ảo hóa.

- Đối với các tác vụ yêu cầu tốc độ và hiệu quả tài nguyên cao, **KVMS** là lựa chọn ưu tiên. Ngược lại, **VMWare** phù hợp với các nhu cầu thử nghiệm nhiều hệ điều hành hoặc môi trường khác nhau.

CHƯƠNG 4. MÔ HÌNH, KỊCH BẢN VÀ KẾT QUẢ TRIỂN KHAI KVM

4.1. Kịch bản 1: Tạo máy ảo thông qua GUI Virt Manager.

4.1.1. Ngữ cảnh và mô hình

Virt Manager (Virtual Machine Manager) là một công cụ giao diện đồ họa thân thiện, được sử dụng rộng rãi để quản lý máy ảo **trên các hệ điều hành nhân Linux**, đặc biệt với công nghệ KVM. Thay vì thao tác qua các lệnh dòng lệnh phức tạp, Virt Manager cung cấp một giao diện GUI trực quan, giúp người dùng dễ dàng tạo, cấu hình và quản lý máy ảo. Công cụ này hỗ trợ các chức năng nâng cao như quản lý tài nguyên hệ thống, kết nối mạng, và lưu trữ máy ảo.

Trong mô hình triển khai này, nhóm chúng em sử dụng một máy thật chạy **Kali Linux 2024.3** được cài đặt KVM làm môi trường ảo hóa. Trên máy thật đó, một máy ảo sử dụng hệ điều hành **Ubuntu Server 22.04** sẽ được triển khai và quản lý hoàn toàn thông qua Virt Manager. Mô hình này tận dụng sự tối ưu của KVM kết hợp với giao diện thân thiện của Virt Manager.

4.1.2. Triển khai

Bước 1: Chuẩn bị máy chủ vật lý

- Đảm bảo máy chủ vật lý đáp ứng các yêu cầu để chạy KVM và quản lý máy ảo.
- Yêu cầu về phần cứng
 - + Hỗ trợ ảo hóa: CPU phải hỗ trợ VT-x (Intel) hoặc AMD-V (AMD).
 - + RAM: Tối thiểu là 4 GB (trong trường hợp chạy nhiều máy ảo).
 - + Dung lượng đĩa: Tối thiểu 20GB cho mỗi file ISO của máy ảo.

Bước 2: Cài đặt KVM và các công cụ hỗ trợ

- Chúng ta cần cập nhật hệ thống để tải các gói dữ liệu về máy thông qua các lệnh:

```
sudo apt update && sudo apt upgrade
```

- Sau khi chúng ta cập nhật hệ thống, chúng ta sẽ tải KVM và Virt Manager về máy.
- Thêm người dùng vào nhóm quản lý KVM và khởi động lại hệ thống để thay đổi có tác dụng.

- Kiểm tra trạng thái hoạt động của KVM để đảm bảo hệ thống hoạt động bình thường.

Bước 3: Tải file ISO Ubuntu Server cho máy ảo từ trang chủ của Ubuntu

- Để có thể tạo được máy ảo, chúng ta cần có file ISO ứng với hệ điều hành mà chúng ta muốn tạo.

- Chúng ta sẽ truy cập vào trang của hệ điều hành và tải file ISO tương ứng về máy.

- Lưu ý: Để chắc chắn chúng ta tải đúng file của hệ điều hành cung cấp, chúng ta có thể thực hiện việc hash để so sánh với kết quả mà nhà phát hành đưa ra để tránh tình trạng có những mã độc được chèn vào file ISO nhằm mục đích xấu.

The screenshot shows a Windows Command Prompt window with the URL <https://releases.ubuntu.com/jammy/SHA256SUMS> in the address bar. Below the address bar, there are two lines of text representing SHA256 hashes:

```
bfd1cee02bc4f35db939e69b934ba49a39a378797ce9aee20f6e3e3e728fefbf *ubuntu-22.04.5-desktop-amd64.iso  
9bc6028870aef3f74f4e16b900008179e78b130e6b0b9a140635434a46aa98b0 *ubuntu-22.04.5-live-server-amd64.iso
```

Below the browser window, the Command Prompt window shows the command `certutil -hashfile ubuntu-22.04.5-live-server-amd64.iso SHA256` being run. The output shows the SHA256 hash of the local ISO file:

```
C:\Users\WanThinnn\Downloads\Programs>certutil -hashfile ubuntu-22.04.5-live-server-amd64.iso SHA256  
SHA256 hash of ubuntu-22.04.5-live-server-amd64.iso:  
9bc6028870aef3f74f4e16b900008179e78b130e6b0b9a140635434a46aa98b0  
CertUtil: -hashfile command completed successfully.  
C:\Users\WanThinnn\Downloads\Programs>
```

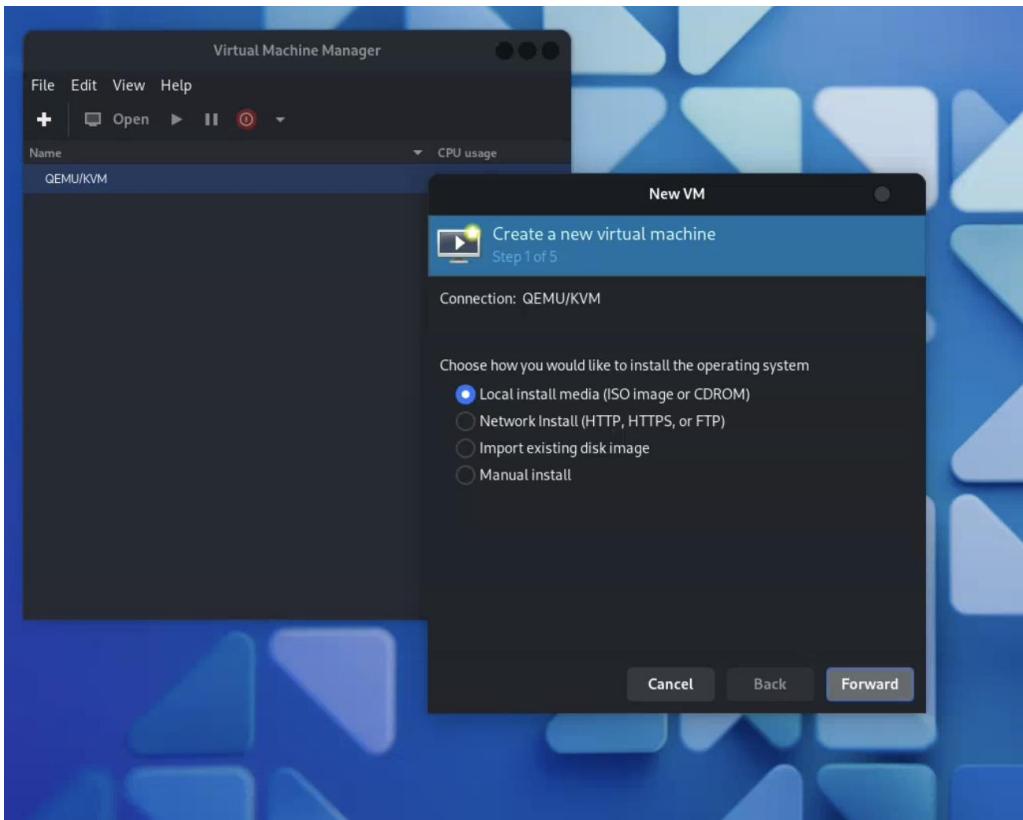
Hình 10. Kết quả băm SHA256 file iso Ubuntu Server 22.04

Bước 4: Tạo máy ảo mới bằng GUI của Virt Manager

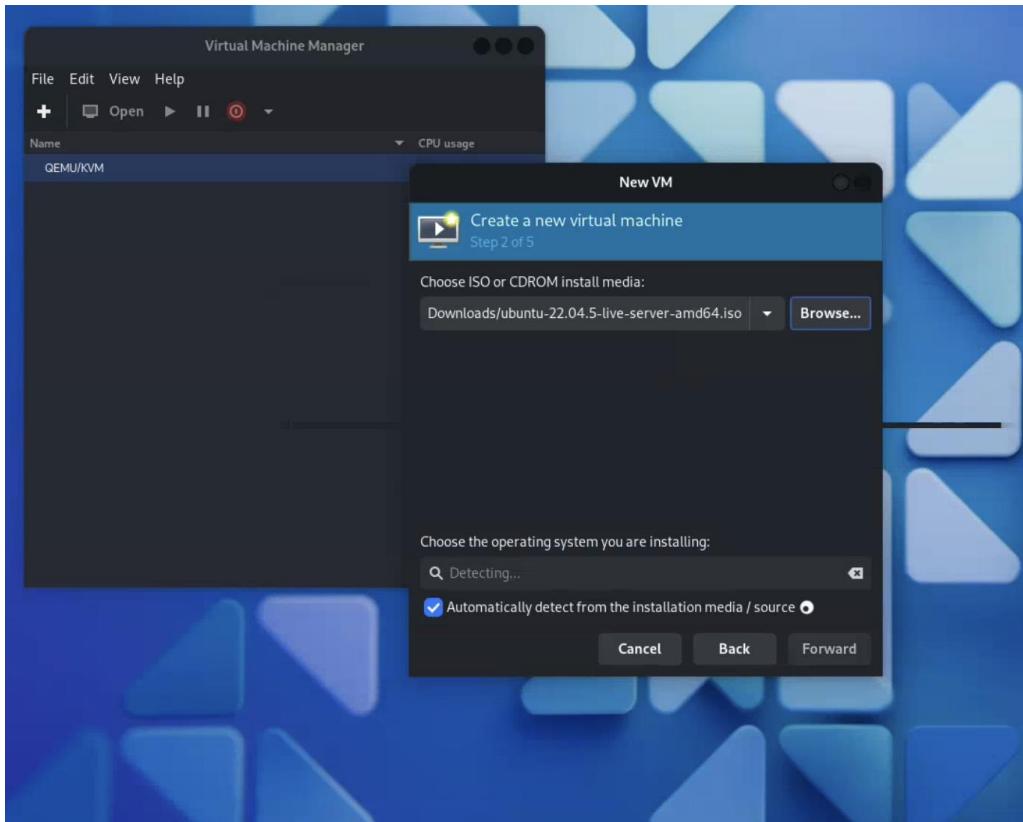
- Thông qua Terminal của hệ điều hành, chúng ta sẽ thực hiện mở Virt Manager thông qua câu lệnh:

```
sudo virt manager
```

- Tạo máy ảo mới bằng việc nhấn “**Create a new virtual machine**”, chúng ta sẽ chọn “Local install media ISO” để có thể import file ISO của chúng ta vừa tải vào.

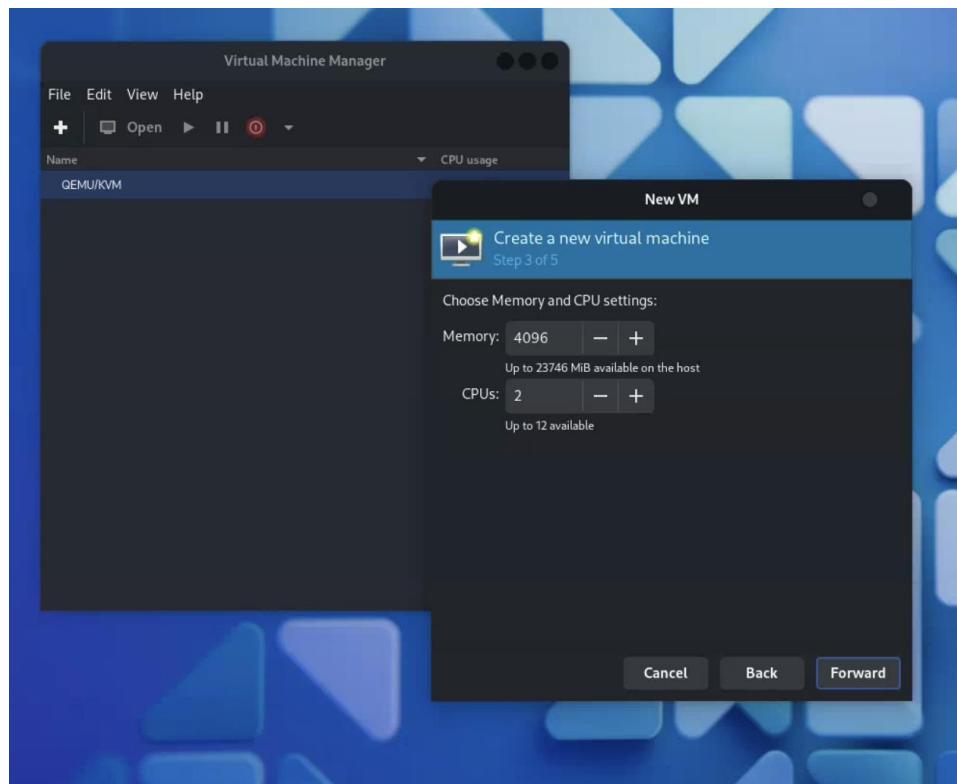


Hình 11. Tạo máy ảo

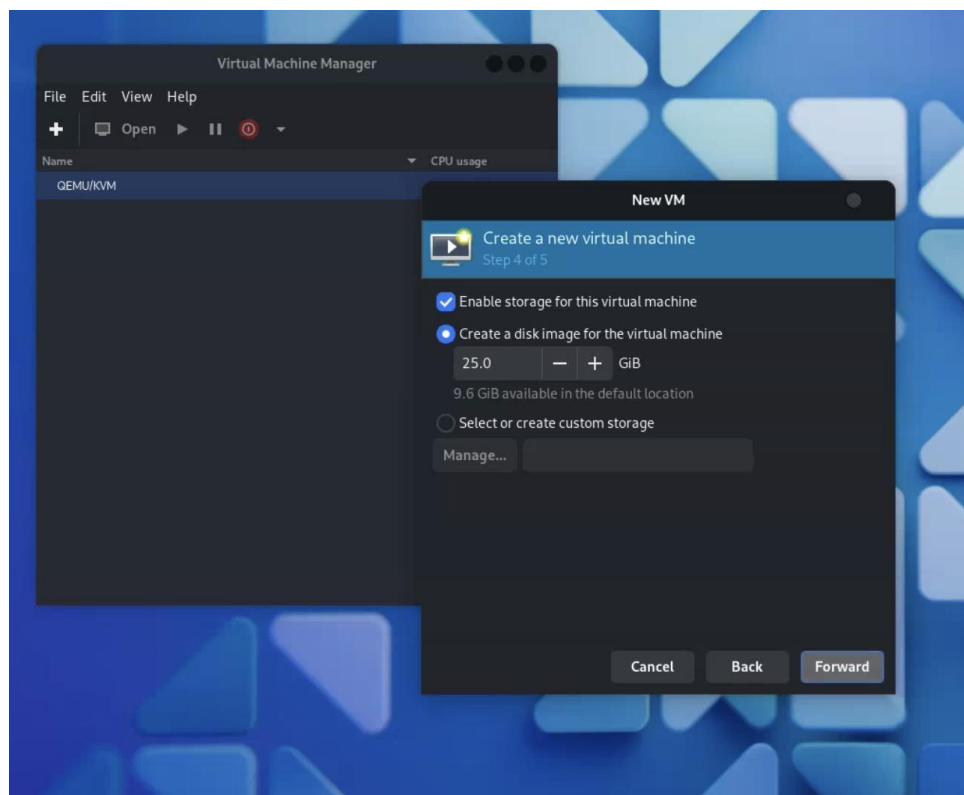


Hình 12. Chọn file ISO cài đặt

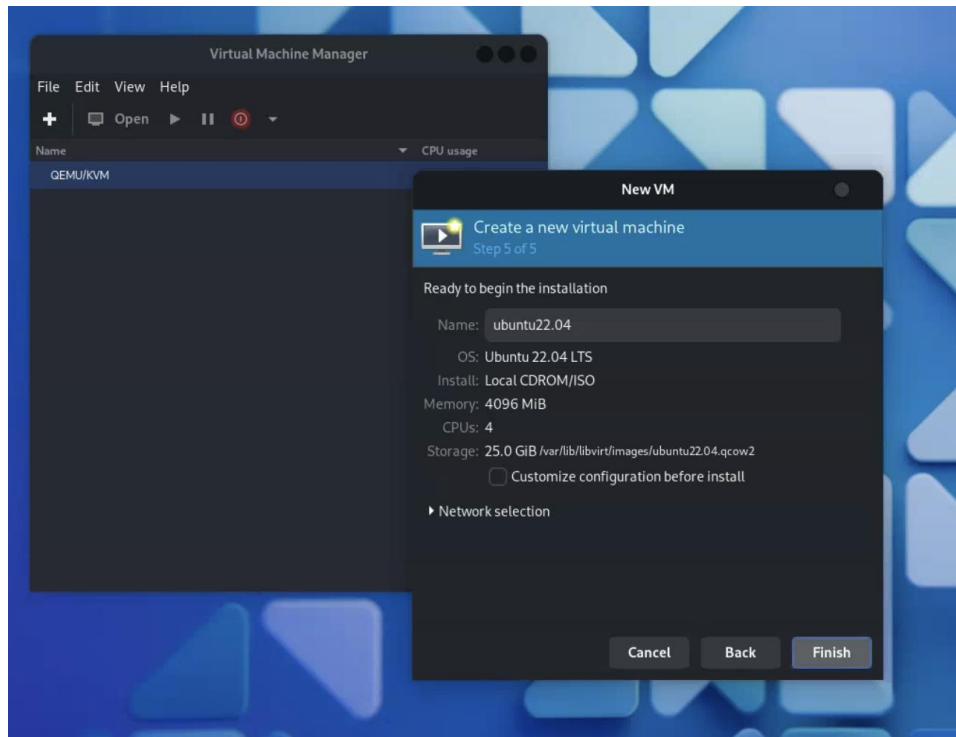
- Sau đó, chúng ta sẽ thực hiện những cấu hình cơ bản ban đầu cho máy ảo qua việc cấp cho chúng những CPU, RAM, Disk Storage, Networking tùy theo nhu cầu và mục đích của chúng ta.



Hình 13. Cấu hình RAM và CPU cho máy ảo



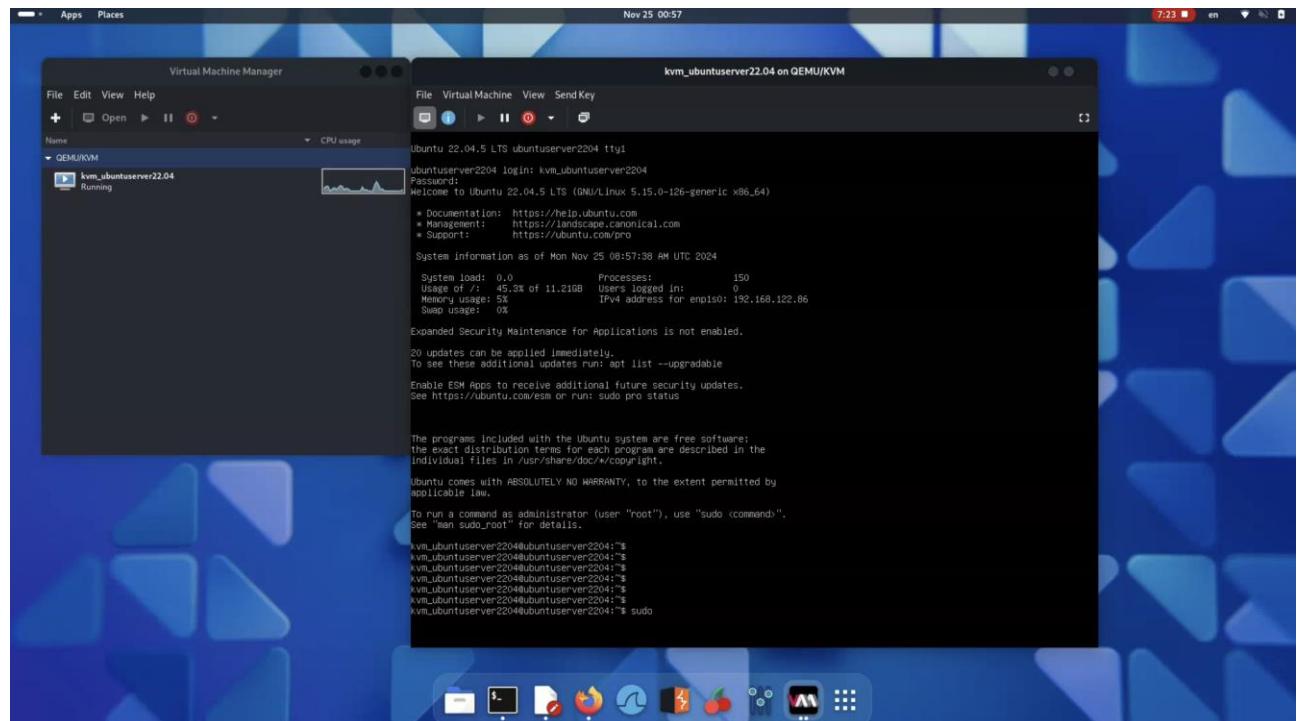
Hình 14. Cấp phát dung lượng bộ nhớ cho máy ảo



Hình 15. Xác thực lại các thông số

4.1.3. Kết quả

Khi hoàn tất các bước trên, chúng ta sẽ có một máy ảo chạy trên hệ điều hành Ubuntu Server 22.04:



Hình 16. Giao diện dòng lệnh của máy ảo Ubuntu Server 22.04 sau khi cài đặt xong

Qua đó, ta thấy rằng quá trình triển khai máy ảo Ubuntu Server 22.04 trên môi trường KVM + Virt Manager GUI khá đơn giản và dễ thực hiện, phù hợp cho cả những người mới bắt đầu tìm hiểu đến lĩnh vực này.

Ngoài ra, khi sử dụng thử máy ảo dựa trên KVM ta có thể nhận thấy được một vài điều sau:

- **Máy ảo hoạt động ổn định:** Sau khi khởi động, máy ảo có thể thực thi các tác vụ cơ bản và nâng cao mà không gặp lỗi.
- **Cấu hình đúng nhu cầu:** Tài nguyên hệ thống (CPU, RAM, và dung lượng đĩa) được phân bổ chính xác theo thông số đầu vào.
- **Kết nối mạng:** Máy ảo được ta thiết lập mạng NAT, có khả năng truy cập Internet và giao tiếp với máy thật hoàn toàn bình thường.
- **Quản lý máy ảo dễ dàng:** Virt Manager cung cấp giao diện trực quan giúp kiểm tra trạng thái, cấu hình, và truy cập máy ảo nhanh chóng.

4.1.4. Ứng dụng

1. Ứng dụng trong Phát triển Phần mềm

- Môi trường phát triển cách ly: Tạo môi trường riêng biệt cho việc phát triển phần mềm mà không làm ảnh hưởng đến hệ thống chính.
 - Thử nghiệm đa nền tảng: Virt Manager cho phép chạy nhiều hệ điều hành (Linux, Windows), giúp kiểm thử phần mềm trên các nền tảng khác nhau.
 - Phát triển và kiểm tra DevOps: Tạo môi trường CI/CD giả lập hoặc chạy thử các công cụ như Docker, Kubernetes, hoặc Jenkins.

2. Mô phỏng hạ tầng Network và IT

- Triển khai mạng ảo: Virt Manager hỗ trợ thiết lập các loại mạng (NAT, Bridge, Internal), phù hợp để mô phỏng các hệ thống mạng phức tạp.
 - Thử nghiệm cấu hình: Tạo các máy chủ ảo (VD: web server, database server) để kiểm tra trước khi triển khai trên hạ tầng thực tế.
 - Mô phỏng kết nối SSH và bảo mật: Giả lập kết nối từ xa giữa các máy ảo để thử nghiệm giao thức SSH, VPN, hoặc các cấu hình bảo mật.

3. Ứng dụng trong Học tập và Nghiên cứu

- Học hệ điều hành: Virt Manager cho phép cài đặt và trải nghiệm nhiều hệ điều hành để học tập, như Ubuntu, Windows, CentOS. Rất phù hợp với những bài Lab yêu cầu hiệu năng cao trong các chương trình học thuộc lĩnh vực CNTT.

- Thử nghiệm công nghệ mới: Dễ dàng triển khai các công nghệ mới như container, hypervisor, hoặc nền tảng phần mềm phức tạp mà không cần phần cứng chuyên dụng.

- Học bảo mật và pentest: Tạo môi trường an toàn để thực hiện các bài học về bảo mật, phân tích mã độc, hoặc thử nghiệm công cụ như Metasploit.

4. Ứng dụng trong Doanh nghiệp

- Ảo hóa máy chủ: Triển khai máy chủ dịch vụ (web, cơ sở dữ liệu, email) trong môi trường ảo hóa giúp giảm chi phí phần cứng.

- Tối ưu hóa tài nguyên: Quản lý tài nguyên CPU, RAM, và lưu trữ để phân bổ phù hợp giữa các máy ảo, giúp tận dụng tối đa phần cứng.

- Sao lưu và phục hồi: Virt Manager hỗ trợ snapshot, cho phép phục hồi hệ thống nhanh chóng khi gặp sự cố.

4.2. Kịch bản 2: Tạo máy ảo có User Data thông qua CLI Virt Manager.

4.2.1. Ngữ cảnh và mô hình

Trong kịch bản này, mục tiêu của nhóm là tạo một máy ảo KVM – Ubuntu Server 22.04 sử dụng **user-data** và **metadata** để tự động hóa cấu hình ban đầu thông qua giao diện dòng lệnh (CLI) với Virt Manager. Mô hình bao gồm:

- **Host chính:** Máy chủ chạy Kali Linux 2024.3 với KVM và libvirt đã được cấu hình sẵn.

- **Máy ảo:** Được tạo dựa trên 2 tệp ISO chứa thông tin tự động hóa (seed ISO) cùng với các tệp user-data và metadata, cung cấp các thông tin như tài khoản người dùng, hostname, và các thông số khác + tệp ISO ubuntu-server-2204 để cài máy ảo.

Dưới đây là đoạn code CLI để tạo máy ảo cùng với User Data:

```
#!/bin/bash

# Function to create the seed ISO
create_seed_iso() {
    echo "Creating a seed ISO file with userdata and metadata..."

    # Create a directory for userdata and metadata
    mkdir -p ~/Documents/NT132/kvm_userdata

    # Prompt for personal details
    read -p "Enter your full name: " FULL_NAME
    read -p "Enter Linux hostname (default: ubuntu-vm): " LINUX_HOSTNAME
    LINUX_HOSTNAME=${LINUX_HOSTNAME:-ubuntu-vm}
    read -p "Enter username for your VM (default: user): " VM_USERNAME
    VM_USERNAME=${VM_USERNAME:-user}

    # Prompt for password input and validate twice
    while true; do
        echo "Enter password for your VM: "
        read -s PASSWORD1
        echo "Re-enter password for your VM: "
        read -s PASSWORD2

        if [ "$PASSWORD1" == "$PASSWORD2" ]; then
            echo "Password confirmed."
            break
        else
            echo "Passwords do not match. Please try again."
        fi
    done
```

```

# Create the user-data file
cat <<EOF > ~/Documents/NT132/kvm_userdata/user-data
#cloud-config
autoinstall:
  version: 1
  identity:
    hostname: $LINUX_HOSTNAME
    username: $VM_USERNAME
    password: $(openssl passwd -6 "$PASSWORD2")
    realname: $FULL_NAME
  locale: en_US
  ssh:
    install-server: true
    allow-pw: true
EOF

# Create the meta-data file
cat <<EOF > ~/Documents/NT132/kvm_userdata/meta-data
instance-id: $LINUX_HOSTNAME
local-hostname: $LINUX_HOSTNAME
EOF

# Generate the ISO file containing userdata and metadata
genisoimage -output ~/Documents/NT132/kvm_userdata/seed.iso \
  -volid cidata -joliet -rock \
  ~/Documents/NT132/kvm_userdata/user-data ~/Documents/NT132/kvm_userdata/meta-
data

echo "ISO file with userdata and metadata has been created at
~/Documents/NT132/kvm_userdata/seed.iso."
}

# Function to create and start a virtual machine
create_vm() {
  # Prompt user for VM details
  read -p "Enter VM name (default: ubuntu-server-vm): " VM_NAME
  VM_NAME=${VM_NAME:-ubuntu-server-vm}

  read -p "Enter ISO file path: " ISO_PATH

  # Default seed ISO path
  SEED_ISO=${SEED_ISO:-$HOME/Documents/NT132/kvm_userdata/seed.iso}

  read -p "Enter disk size (default: 25G): " DISK_SIZE
  DISK_SIZE=${DISK_SIZE:-25G}

  read -p "Enter number of CPUs (default: 4): " CPU
  CPU=${CPU:-4}

  read -p "Enter RAM size in MB (default: 4096): " RAM

```

```

RAM=${RAM:-4096}

read -p "Enter network type (default: NAT): " NETWORK
NETWORK=${NETWORK:-default}

# Disk path is set automatically based on VM name
DISK_PATH="/var/lib/libvirt/images/$VM_NAME.qcow2"

read -p "Enter OS variant for VM: " osvariant
OS_VARIANT=$osvariant

# Create the virtual disk
echo "Creating virtual disk at $DISK_PATH with size $DISK_SIZE..."
sudo qemu-img create -f qcow2 $DISK_PATH $DISK_SIZE

# Start the VM installation
echo "Starting VM installation..."
sudo virt-install \
--name "$VM_NAME" \
--vcpus "$CPU" \
--memory "$RAM" \
--disk path="$DISK_PATH",format=qcow2 \
--cdrom "$ISO_PATH" \
--disk path="$SEED_ISO",device=cdrom \
--os-variant "$OS_VARIANT" \
--network network="$NETWORK" \
--boot cdrom,hd \
--noautoconsole

# Display message about GUI
sudo virt-manager --connect qemu:///system --show-domain-console $VM_NAME
}

# Main script execution
echo "Starting script to create a VM..."
create_seed_iso
create_vm

```

Giải thích mã nguồn trên:

Phần 1: Tạo Seed ISO (Hàm create_seed_iso) => Mục tiêu: Tạo file ISO chứa thông tin cấu hình tự động (user-data và meta-data) để dùng trong quá trình cài đặt máy ảo.

1. Tạo thư mục và nhập thông tin người dùng:

- Tạo thư mục ~/Documents/NT132/kvm_userdata để lưu dữ liệu.
- Thu thập các thông tin:

- **Tên đầy đủ** (FULL_NAME).
- **Hostname** cho Linux (mặc định: ubuntu-vm).
- **Tên người dùng** (mặc định: user).
- **Mật khẩu**, yêu cầu nhập hai lần để xác nhận.

2. Sinh tệp user-data:

- Tệp này dùng chuẩn **cloud-init** để tự động hóa cài đặt.
- Thông tin cấu hình bao gồm:
 - **Hostname, tên người dùng, mật khẩu băm** (SHA-512).
 - **Cài đặt SSH** (cho phép sử dụng mật khẩu để đăng nhập).

3. Sinh tệp meta-data: Chứa thông tin định danh máy ảo như:

- **instance-id**
- **local-hostname**.

4. Tạo file ISO: Sử dụng genisoimage để gói hai tệp user-data và meta-data vào file seed.iso.

Phần 2: Tạo Máy Ảo (Hàm create_vm) => Mục tiêu: Tạo và khởi động máy ảo với các thông số được chỉ định.

1. Thu thập thông tin máy ảo: Nhập các giá trị:

- **Tên máy ảo** (VM_NAME, mặc định: ubuntu-server-vm).
- **Đường dẫn ISO hệ điều hành** (ISO_PATH).
- **Dung lượng ổ đĩa** (mặc định: 25G).
- **Số CPU và RAM** (mặc định: 4 CPU, 4096MB).
- **Kiểu mạng** (mặc định: NAT).
- **Phiên bản hệ điều hành** (OS_VARIANT).

2. Tạo ổ đĩa ảo: Sử dụng qemu-img để tạo ổ đĩa ảo dạng QCOW2 tại /var/lib/libvirt/images/<VM_NAME>.qcow2.

3. Khởi tạo máy ảo:

- Sử dụng virt-install để cài đặt máy ảo với các thông số:

- Gắn ISO hệ điều hành và seed.iso để tự động hóa cài đặt.
- Gán tài nguyên phần cứng (CPU, RAM, ổ đĩa).
- Kết nối mạng và cấu hình boot (ưu tiên từ CDROM).

4. Mở Virt Manager: Tự động mở giao diện quản lý máy ảo qua virt-manager.

4.2.2. Triển khai

Trước khi chạy đoạn code trên, ta cần phải cấp quyền thực thi cho nó:

```
chmod +x create_vm.sh
```

Tiến hành chạy đoạn code và điền những thông tin cơ bản cho máy ảo:

```
wanthinnn@wanthinnn: ~/Documents
(wanthinnn@wanthinnn)-[~/Documents]
$ ./create_vm.sh
Starting script to create a VM...
Creating a seed ISO file with userdata and metadata...
Enter your full name: lai quan thien
Enter Linux hostname (default: ubuntu-vm): ubuntu-server-2204
Enter username for your VM (default: user): admin
Enter password for your VM:
Re-enter password for your VM:
Password confirmed.
I: -input-charset not specified, using utf-8 (detected in locale settings)
Total translation table size: 0
Total rockridge attributes bytes: 331
Total directory bytes: 0
Path table size(bytes): 10
Max brk space used 0
183 extents written (0 MB)
ISO file with userdata and metadata has been created at ~/Documents/NT132/kvm_us
erdata/seed.iso.
Enter VM name (default: ubuntu-server-vm): ubuntu-server-2204
Enter ISO file path: /home/wanthinnn/Downloads/ubuntu-22.04.5-live-server-amd64.
iso
Enter disk size (default: 25G): 64G
```

Hình 17. Điền những thông tin cần thiết cho máy ảo

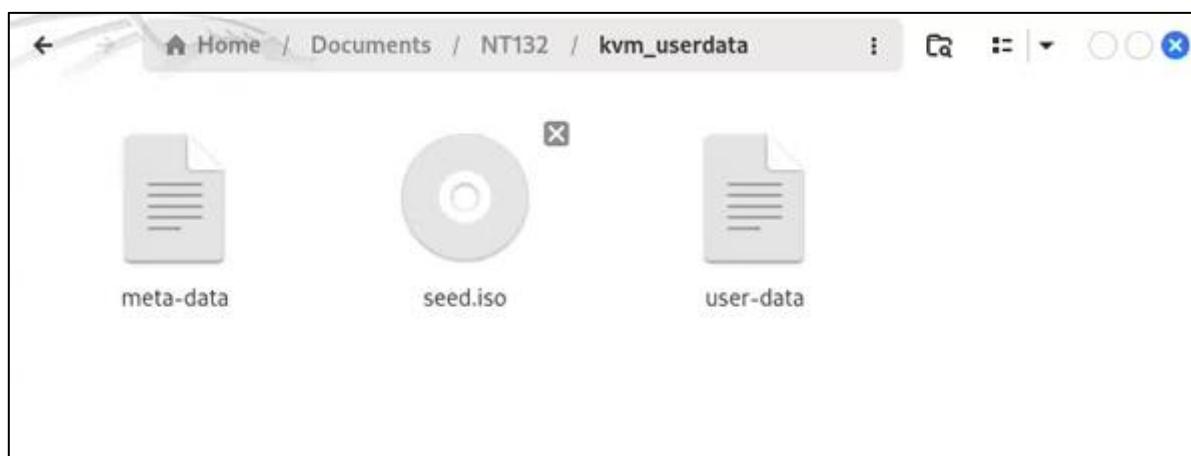
```
wanthinnn@wanthinnn:~/Documents
Enter VM name (default: ubuntu-server-vm): ubuntu-server-2204
Enter ISO file path: /home/wanthinnn/Downloads/ubuntu-22.04.5-live-server-amd64.iso
Enter disk size (default: 25G): 64G
Enter number of CPUs (default: 4): 8
Enter RAM size in MB (default: 4096): 8192
Enter network type (default: NAT):
Enter OS variant for VM: ubuntu22.04
Creating virtual disk at /var/lib/libvirt/images/ubuntu-server-2204.qcow2 with size 64G...
[sudo] password for wanthinnn:
Formatting '/var/lib/libvirt/images/ubuntu-server-2204.qcow2', fmt=qcow2 cluster_size=65536 extended_l2=off compression_type=zlib size=68719476736 lazy_refcount=off refcount_bits=16
Starting VM installation...

Starting install...
Creating domain... | 0 B 00:00
Domain is still running. Installation may be in progress.
You can reconnect to the console to complete the installation process.

(wanthinnn@wanthinnn)-[~/Documents]
```

Hình 18. Quá trình nhập thông tin thành công, giao diện Virt Manager sẽ hiện lên để thực thi phần còn lại

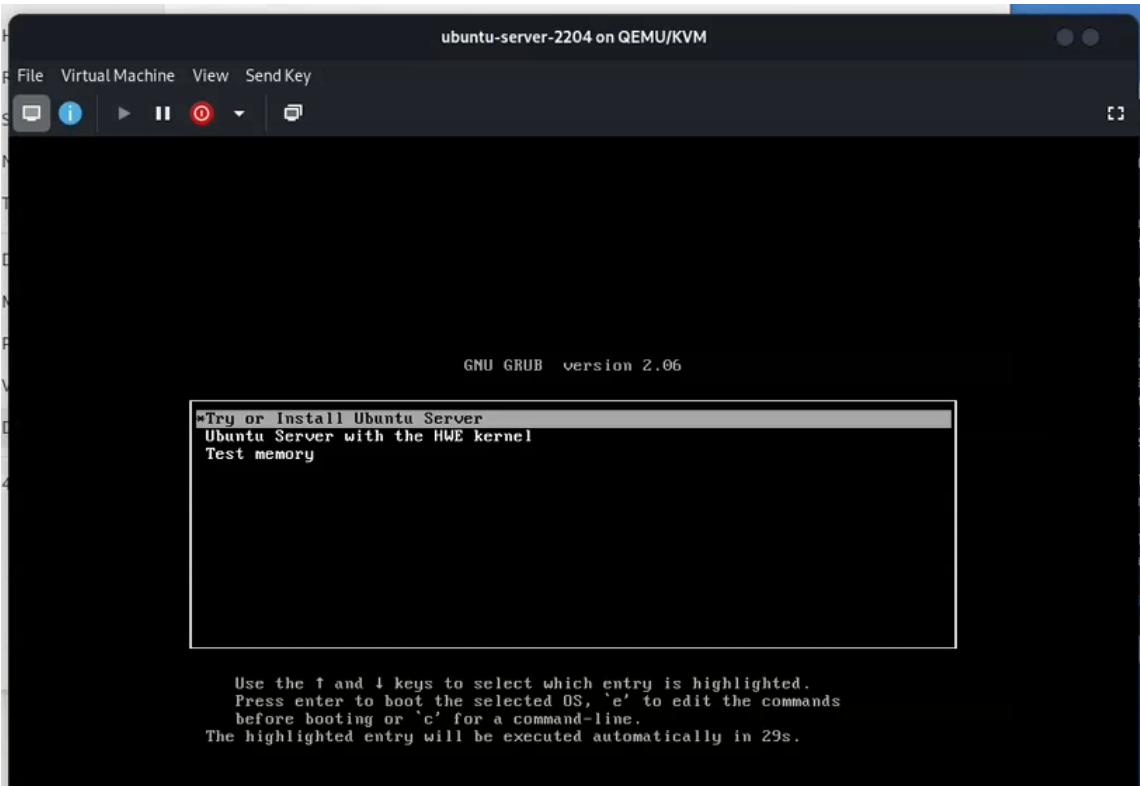
Sau khi ta điền những thông tin cơ bản cho máy ảo xong, đoạn code trên sẽ tạo ra 1 file seed.iso, file này chứa những thông tin tự động hóa cùng với các tệp user-data và metadata, cung cấp các thông tin như tài khoản người dùng, hostname, và các thông số khác. Ta có thể tận dụng file seed.iso này để tạo cho những máy ảo khác nhau nhằm tiết kiệm thời gian điền những thông tin cơ bản cho máy ảo mới.



Hình 19. File Seed.iso được tạo thành công

Nếu quá trình diễn ra suôn sẻ, không có lỗi gì, thì giao diện cài đặt máy ảo của Virt Manager sẽ hiện ra, lúc này ta chỉ việc nhấn Enter để cài luôn máy ảo Ubuntu Server và các

userdata mà ta đã nhập vào, không cần phải cấu hình gì thêm, quá trình diễn ra hoàn toàn tự động



Hình 20. Giao diện dòng lệnh để cài đặt Ubuntu Server 22.04

4.2.3. Kết quả

Quá trình cài đặt sẽ tồn từ 8-12 phút tùy thuộc vào cấu hình máy và tốc độ mạng

Sau khi hoàn tất, ta sẽ được giao diện dòng lệnh của Ubuntu Server như sau:

A screenshot of a terminal window showing the completed Ubuntu 22.04.5 LTS installation log. The log starts with "Ubuntu 22.04.5 LTS ubuntu-server-2204 tty1" and "Hint: Num Lock on". It then shows the "cloud-init" logs, which include the generation of SSH keys and host keys. The log concludes with "Cloud-init v. 24.2-0ubuntu1~22.04.1 finished at Mon, 25 Nov 2024 09:41:16 +0000. Datasource DataSourceNone. Up 13.02 seconds". The log is very long and contains many lines of technical details about the key generation process.

Hình 21. Quá trình cài đặt thành công

Tiến hành đăng nhập vào:

```
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAIEvE0+1T2+5squvuP2aEkngWuIvnNU1KE2oEgzNn+j root@ubuntu-server-2204
ssh-rsa AAAAB3NzaC1yc2EAAQABAAAQgD0Iuqv05dc1Jkb5x6av2R6Pmds6UjB196bf8uuTaLrIxgLCLXULydJ9bfIFLz1r7404c+9kajx1EeRfF9rpKUoBNOT
09x7tEgyUpd1RS08xtk7a9cB9knqLL00YXDRRKfcqHLSfdbmwsn7Njengz9IkpsdJGP103207wMqy0UgnLxZVGcmjgV/a3TTn+IAQMI+X1g/tbWahVv0nPL
inpPvys7puKo+xuC06E4MjhS2/4176M/eBImW9Xknlo/BakC1r6Tlss6QVp25r2KEY1y0D0m0xBV732gurR48fAEutJ1k02ivxe10k+w7FH0ep4dzJ0p0A14ehi
o2grK8B0HTpi37gro4TBVKLHS0CdtI+hco2skABpEZaP29U0ihmQ2qTUzU9s5uMxG/S+NxaLs10Wmp16mGeal2XmA7K8PFZeZguwuJsnBENNRXYH40Jpbgbu+8hb5j
EPtN6grRR/DvEaipJLssvnyk1PKhtV/LenDCG0= root@ubuntu-server-2204
-----END SSH HOST KEY KEYS-----
[ 13.02905] cloud-init[1554]: Cloud-init v. 24.2-0ubuntu1~22.04.1 finished at Mon, 25 Nov 2024 09:41:16 +0000. Datasource DataSourceNone. Up 13.02 seconds

Hint: Num Lock on

ubuntu-server-2204 login: admin
Password:
Welcome to Ubuntu 22.04 LTS (GNU/Linux 5.15.0-126-generic x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/pro

System information as of Mon Nov 25 09:41:24 AM UTC 2024

System load: 0.06 Processes: 191
Usage of /: 23.0% of 30.34GB Users logged in: 0
Memory usage: 3% IPv4 address for enp1s0: 192.168.122.252
Swap usage: 0%

Expanded Security Maintenance for Applications is not enabled.

20 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

admin@ubuntu-server-2204:~$
```

Hình 22. Đăng nhập vào tài khoản máy ảo

```
admin@ubuntu-server-2204:~$ sudo apt update
[sudo] password for admin:
Hit:1 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:2 http://archive.ubuntu.com/ubuntu jammy InRelease
Hit:3 http://archive.ubuntu.com/ubuntu jammy-updates InRelease
0% [Waiting for headers]
```

Hình 23. Chạy thử lệnh update apt

4.2.4. Ứng dụng

Việc sử dụng **User Data** trong quá trình triển khai máy ảo mang lại nhiều lợi ích thực tế trong các môi trường ảo hóa và phát triển, bao gồm:

1. Tự động hóa cấu hình ban đầu: Các thiết lập như hostname, username, password, và cài đặt SSH được cấu hình tự động trong quá trình khởi tạo. Điều này giảm thiểu công việc thủ công và rủi ro sai sót.

2. Tiện lợi trong quản lý hàng loạt máy ảo: Kịch bản phù hợp cho môi trường cần triển khai nhiều máy ảo với cấu hình tương tự, ví dụ: triển khai các máy chủ ứng dụng hoặc cơ sở dữ liệu.

3. Ứng dụng trong DevOps và CI/CD: Tích hợp User Data vào quy trình DevOps cho phép khởi tạo nhanh môi trường phát triển hoặc kiểm thử mà không cần can thiệp thủ công.

4. Tăng tính bảo mật và đồng nhất: Mã hóa mật khẩu trực tiếp trong User Data giúp bảo mật thông tin người dùng, đồng thời đảm bảo tất cả các máy ảo được cấu hình nhất quán.

5. Triển khai máy ảo theo nhu cầu: Các tổ chức có thể nhanh chóng triển khai máy ảo phục vụ các bài thí nghiệm, mô phỏng hoặc đào tạo.

4.3. Kịch bản 3: SSH (Secure Shell) giữa 2 máy ảo trong KVM

4.3.1. Ngữ cảnh và mô hình

Trong kịch bản này, nhóm sẽ triển khai một mô hình mạng ảo hóa sử dụng KVM (Kernel-based Virtual Machine) để tạo ra **hai máy ảo Ubuntu Desktop (22.04.5 LTS)**. Mục đích là mô phỏng tình huống khi máy ảo VM 1 cần cài đặt một dự án từ GitHub bằng máy ảo khác (VM 2). Máy ảo VM 2 thông qua kết nối SSH, giúp truy cập đến máy ảo VM 1 và thực hiện các thao tác cần thiết.

Mô hình này sẽ cho phép thực hiện các thao tác mạng giữa các máy ảo trong môi trường KVM, đồng thời kiểm tra khả năng cấu hình và các tính năng của kết nối SSH giữa các máy ảo.

4.3.2. Triển khai

Bước 1: Tạo 2 máy ảo

- Cấu hình các thông số của 2 máy ảo.
- Cấu hình mạng cho 2 máy ảo.
- Quy ước thứ tự của 2 máy ảo Ubuntu như sau:
 - + Terminal **bên trái** là của **máy ảo 1 (có username là kvm2)**
 - + Terminal **bên phải** là của **máy ảo 2 (có username là huy)**

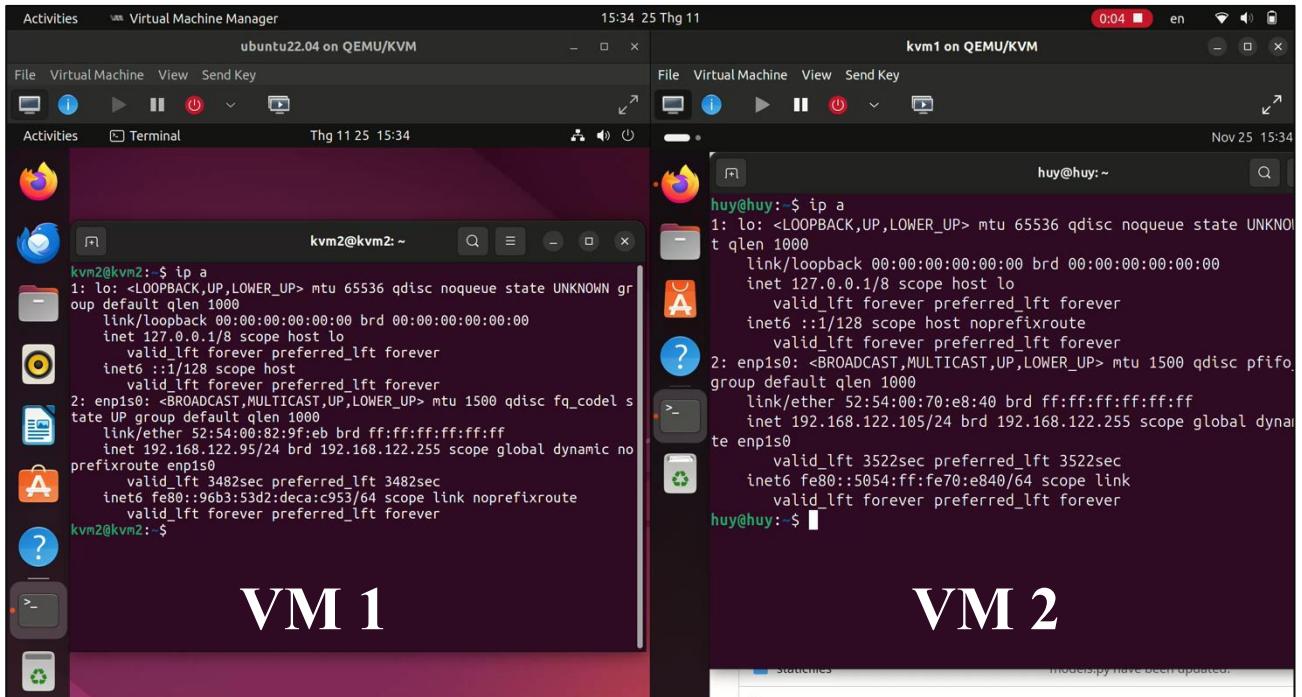
Bước 2: Khởi động máy ảo và cài đặt hệ điều hành Ubuntu Desktop 22.04

Bước 3: Cài đặt SSH trên 2 máy ảo để kết nối với nhau (nếu chưa cài đặt)

- Cài đặt OpenSSH Server để cho phép kết nối: sudo apt install openssh-server
- Khởi động và cấu hình OpenSSH Server để đảm bảo nó tự khởi động khi máy khởi động: sudo systemctl enable ssh.

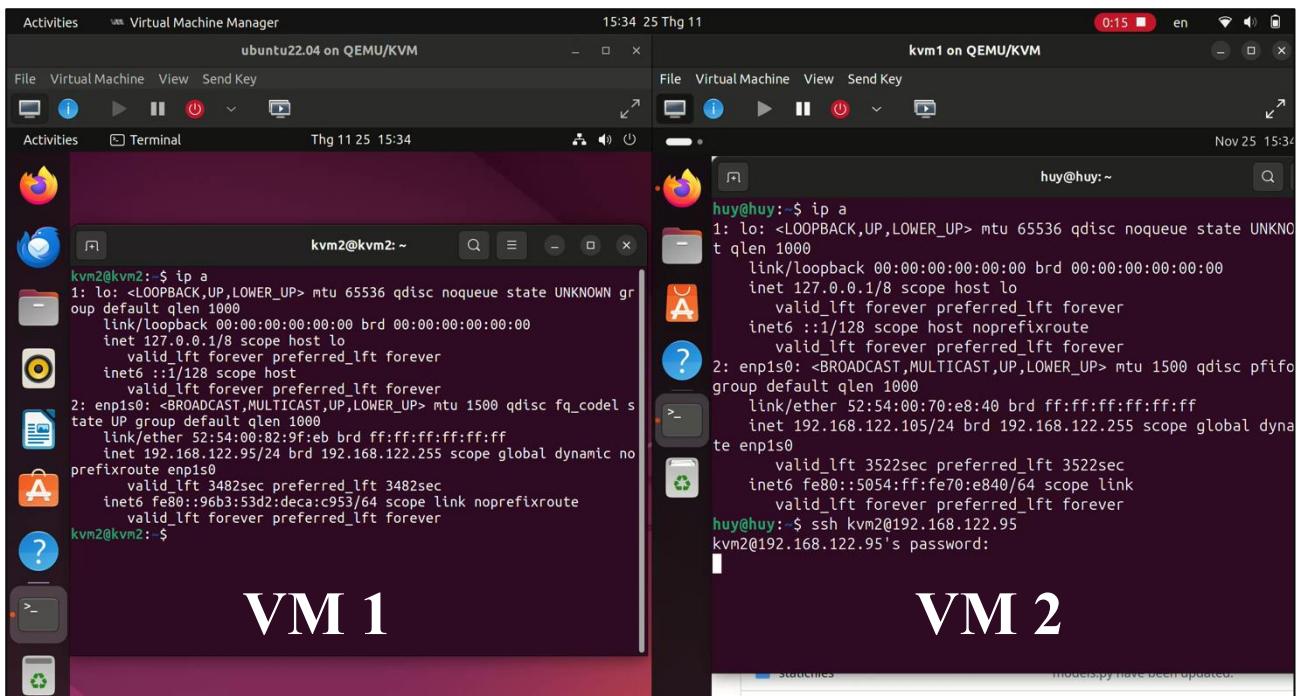
Bước 4: Kết nối máy ảo 2 đến máy ảo 1

- Xác định địa chỉ IP của 2 máy ảo



Hình 24. Xác định địa chỉ IP của 2 máy ảo

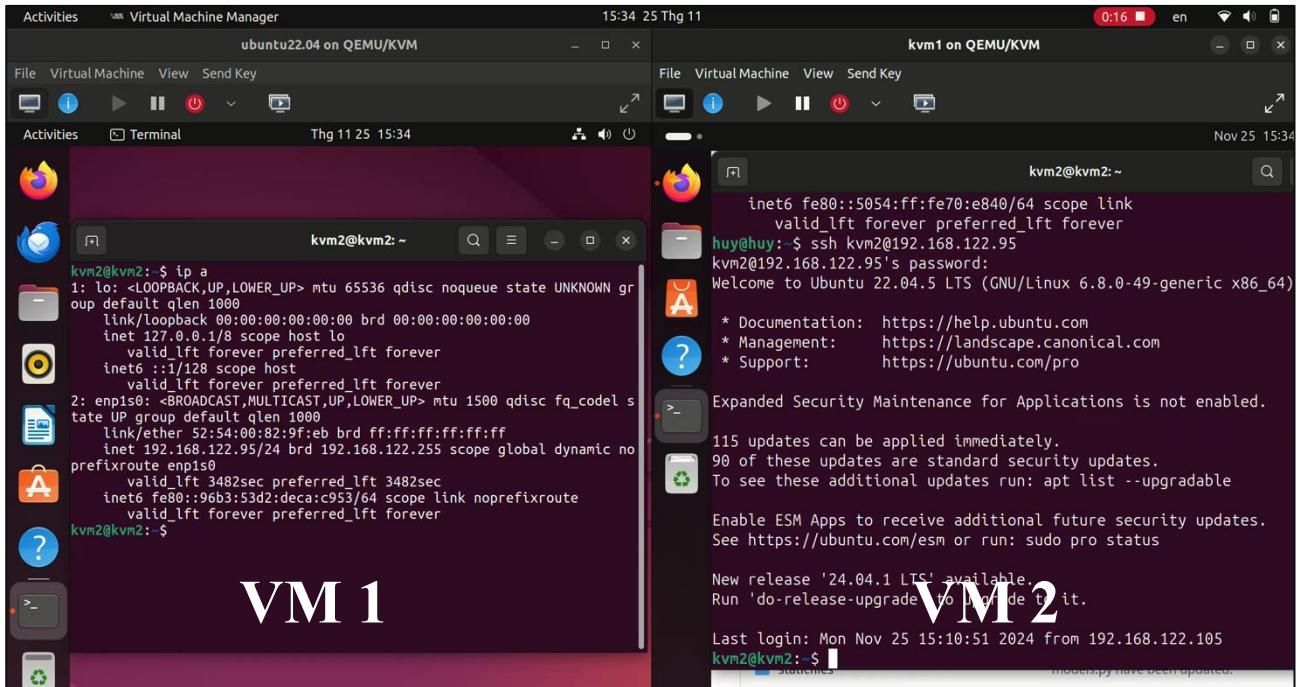
- Nhập mật khẩu của máy ảo 1 để bắt đầu kết nối



Hình 25. Kết nối đến máy ảo kia

Bước 6: Xác minh kết nối

- Sau khi kết nối thành công, trong giao diện cmd hoặc terminal của máy ảo 2 sẽ có command của máy ảo 1.



Hình 26. Kết nối thành công

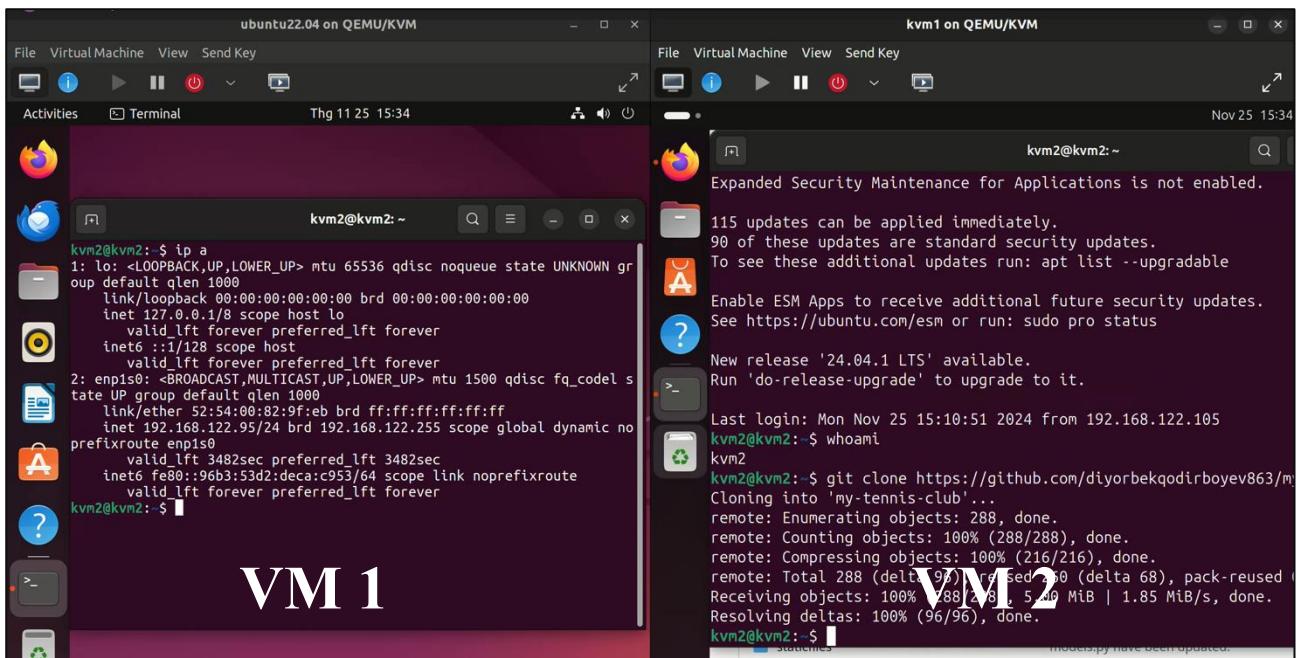
- Thực hiện các thao tác trên máy ảo 2 để điều khiển máy ảo 1 từ xa.

Bước 7: Thực hiện clone và build một dự án có trên Github

- Trong phần triển khai này nhóm sử dụng dự án WebApp Django:

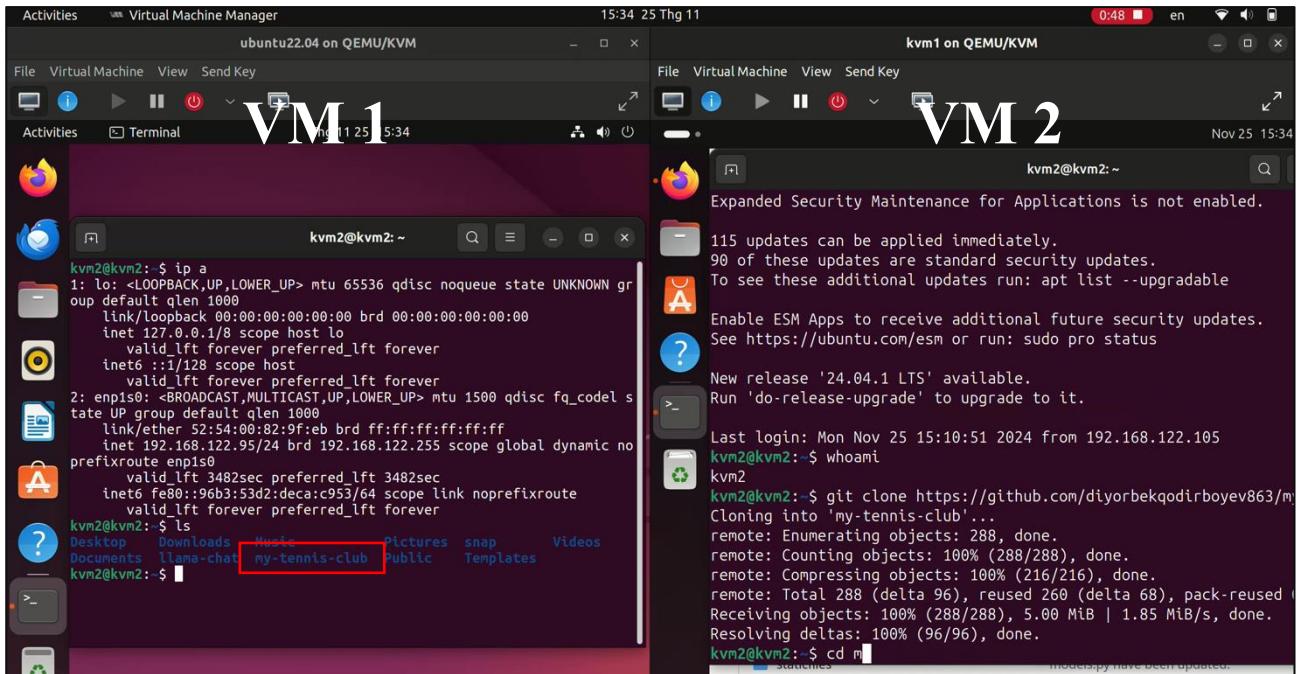
<https://github.com/diyorbekqodirboyev863/my-tennis-club.git>

- Tiến hành clone:



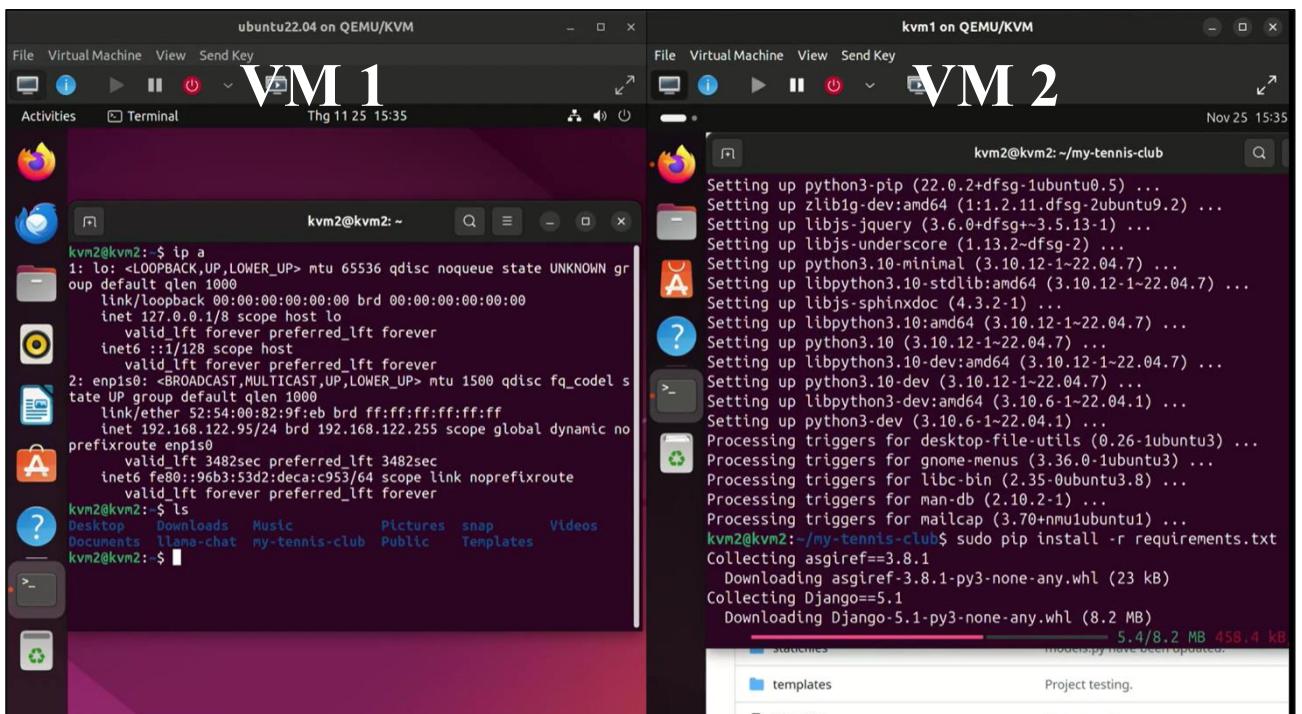
Hình 27. Tiến hành Clone app về

- Kiểm tra trên máy ảo 1 sẽ thấy dự án đã được cài đặt thông qua máy ảo 2.

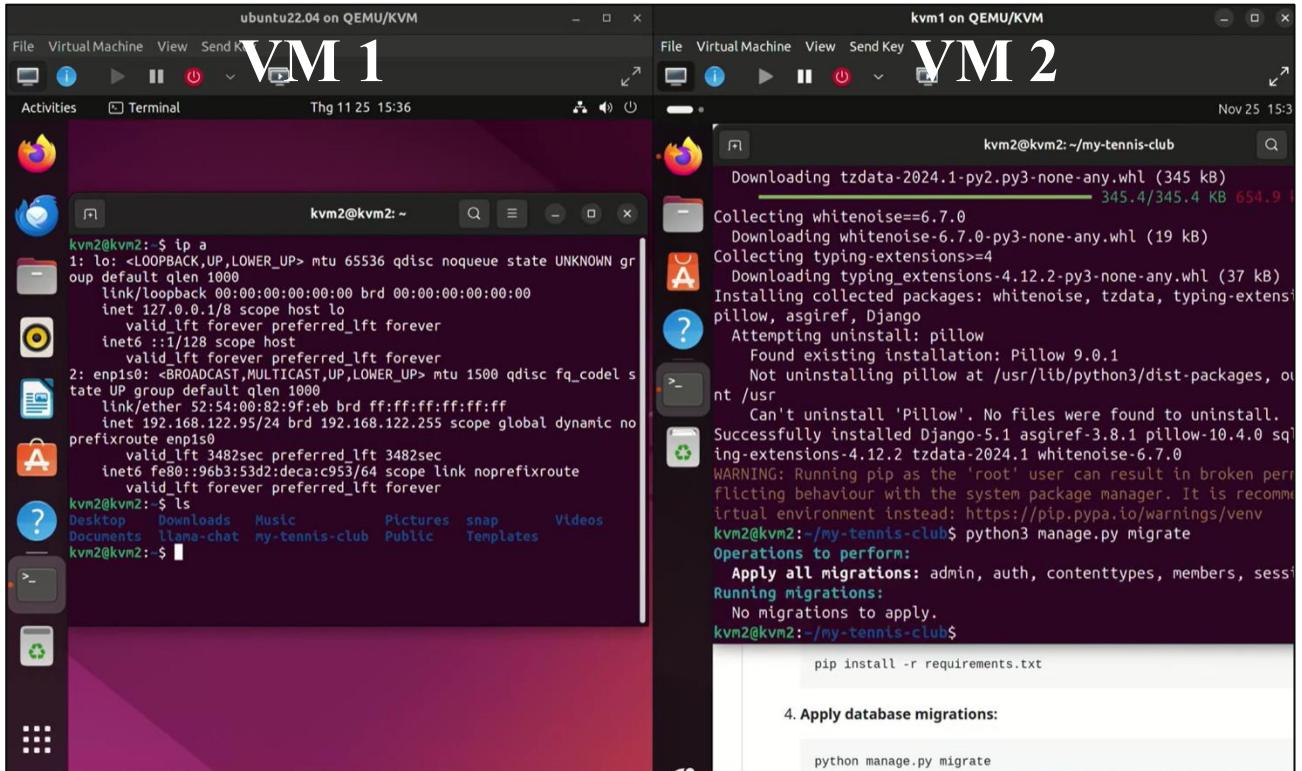


Hình 28. Kiểm tra dự án trên máy ảo được SSH

- Cài đặt các dependencies bằng lệnh: `pip install -r requirements.txt`

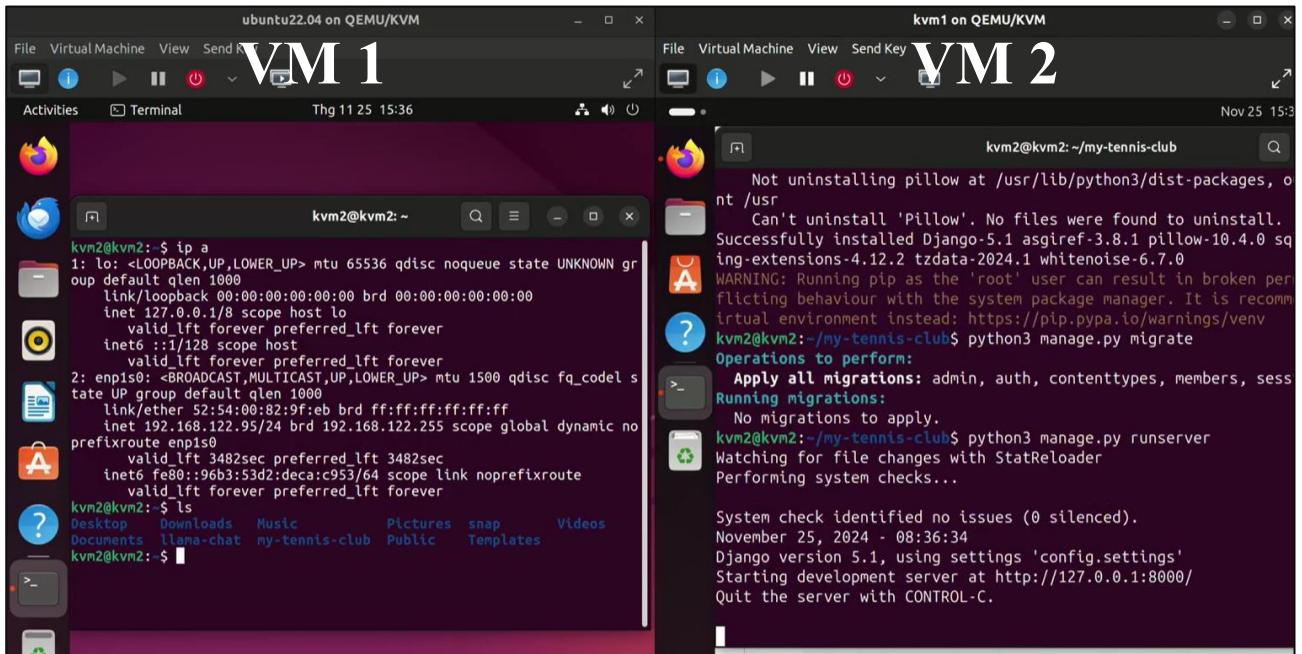


- Apply database migrations: `python manage.py migrate`



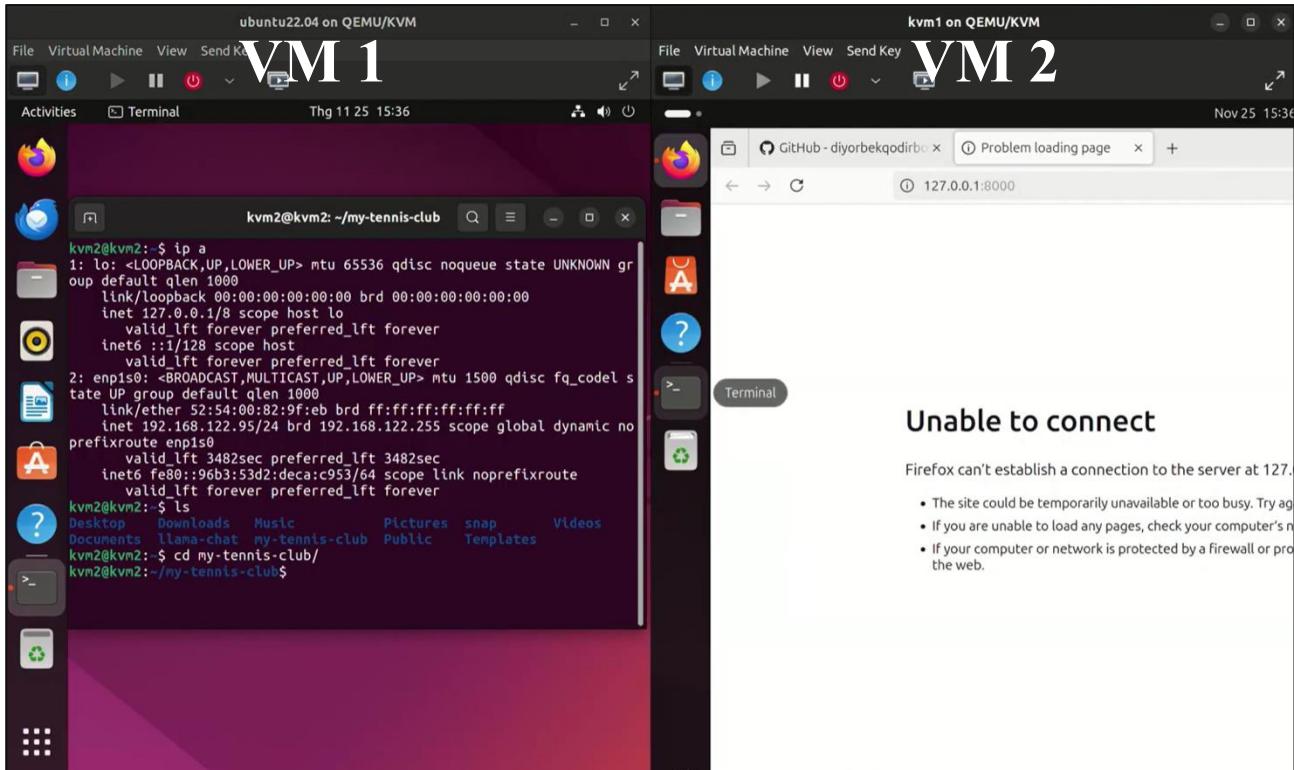
Hình 29. Apply database

- Chạy server bằng lệnh: `python manage.py runserver`



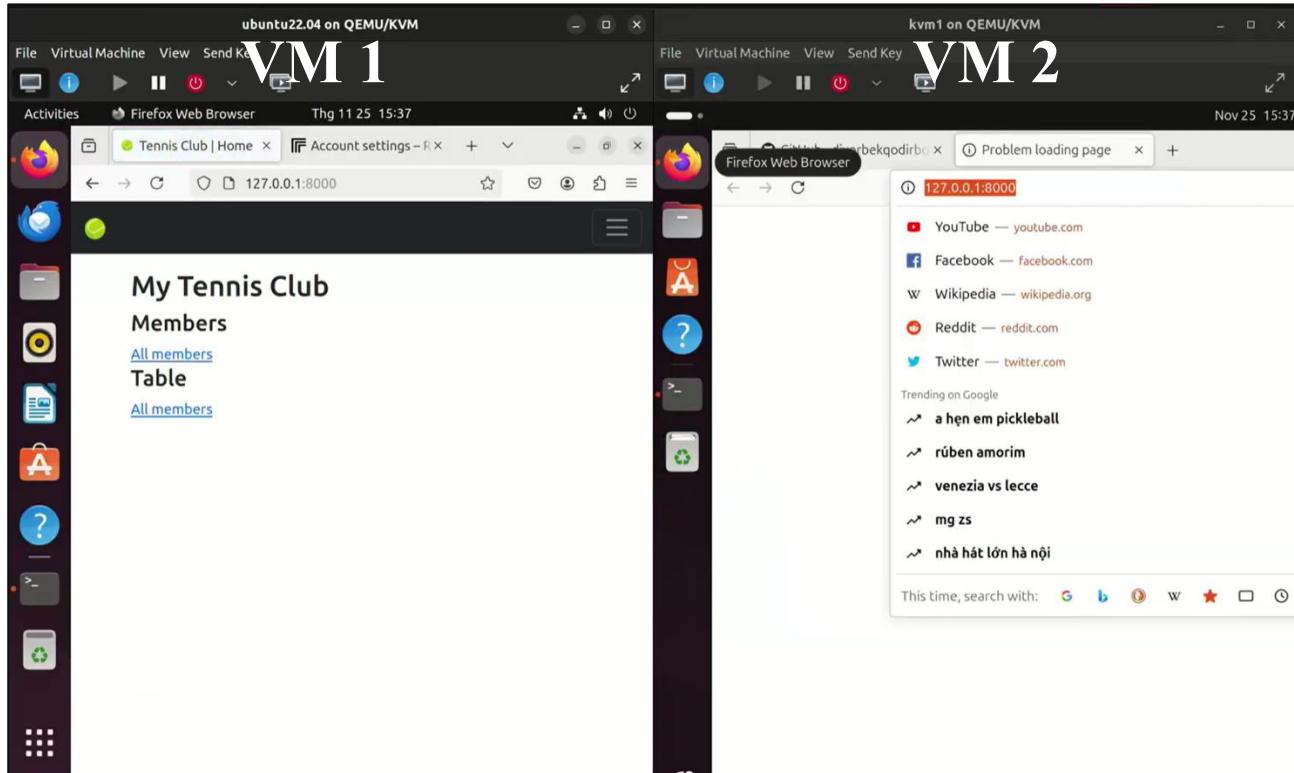
Hình 30. Khởi chạy Server

- Nhập ip http://127.0.0.1:8000/ vào web browser bất kì để xem dự án.

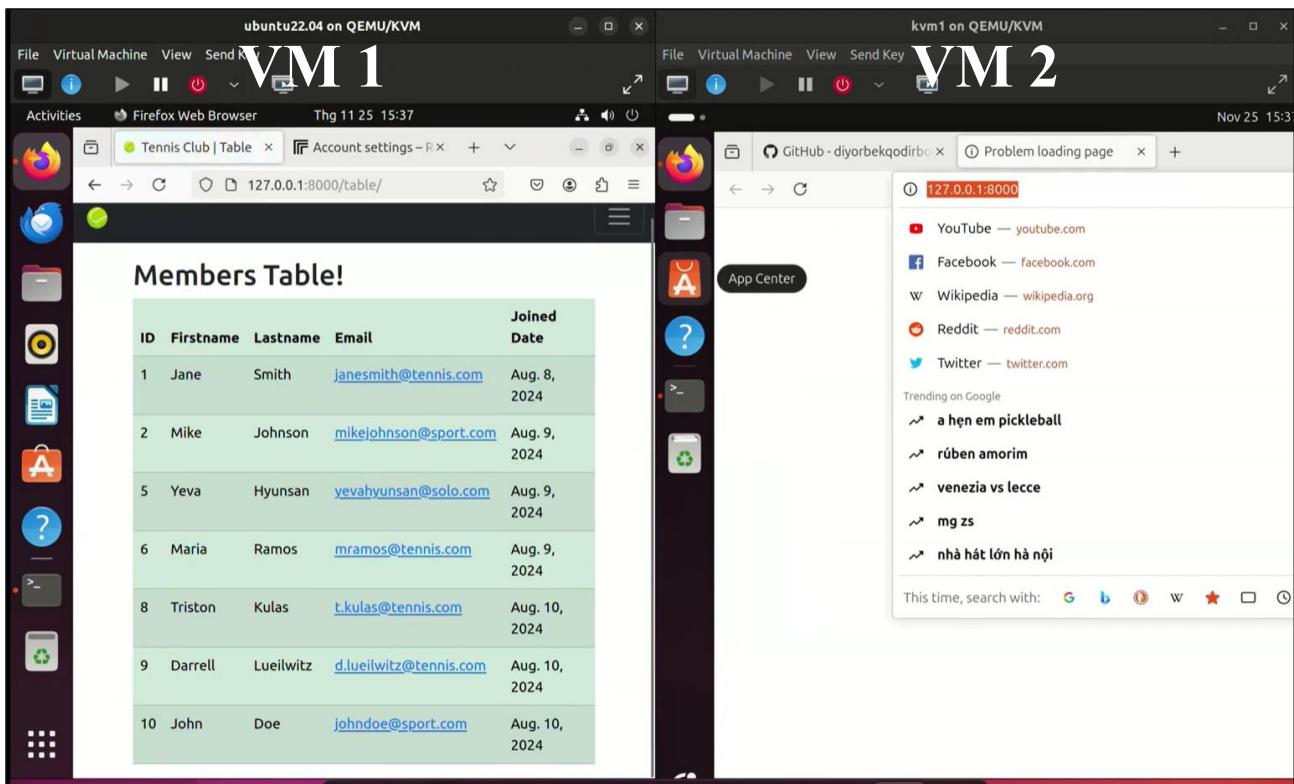


Hình 31. Kiểm tra xem web đã chạy được chưa

- Ta thấy máy ảo 2 không vào được dự án nhưng máy ảo 1 vào được:



Hình 32. WebApp đã chạy ổn định trên máy được SSH, còn máy SSH đến thì không được



Hình 33. Giao diện của Web

4.3.3. Kết quả

- Cả hai máy ảo có thể giao tiếp qua kết nối SSH mà không gặp phải vấn đề về mạng.
- Dự án GitHub đã được clone thành công và build trên máy ảo 1, thông qua việc SSH kết nối từ máy ảo 2.
 - Quá trình build và clone diễn ra suôn sẻ, không gặp phải lỗi trong việc cấu hình hay kết nối mạng.
 - Kết quả thực hiện các thao tác clone hay build dự án bằng cách SSH từ máy ảo 2 đến máy ảo 1 sẽ nằm trên máy ảo 2.

4.3.4. Ứng dụng

1. Mô phỏng môi trường phát triển phần mềm phân tán

- Các đội nhóm hoặc cá nhân có thể phối hợp làm việc trong môi trường phân tán, khi cần triển khai và quản lý dự án từ xa.
- Máy ảo 1 đóng vai trò tương tự như một máy quản lý hoặc máy developer, thực hiện kết nối SSH để điều khiển từ xa.

- Máy ảo 2 mô phỏng môi trường server nơi dự án được triển khai và xử lý.

2. Quy trình DevOps cơ bản

- Mô hình cho thấy một phần của chuỗi công việc trong CI/CD pipeline, cụ thể là việc tự động hóa thao tác clone và build dự án từ GitHub trên môi trường máy chủ từ xa.

- Việc sử dụng SSH từ máy ảo 1 để điều khiển máy ảo 2 có thể mở rộng thành các kịch bản tự động hóa phức tạp hơn như kiểm thử (testing), triển khai (deployment), hoặc theo dõi (monitoring).

3. Phân quyền và bảo mật trong hệ thống phân tán: Bằng cách sử dụng kết nối SSH giữa hai máy ảo, ta có thể kiểm tra khả năng phân quyền và đảm bảo an toàn dữ liệu giữa các thành phần của hệ thống. Điều này đặc biệt hữu ích khi mô phỏng các ứng dụng cần thao tác từ xa trên các server hoặc máy chủ khác nhau.

4. Tối ưu hóa tài nguyên máy chủ ảo hóa

- Việc sử dụng hai máy ảo để thực hiện công việc khác nhau (máy ảo 1 quản lý, máy ảo xử lý dự án) giúp tối ưu hóa tài nguyên trong môi trường ảo hóa.

- Máy ảo 1 không chịu tải nặng, chỉ đóng vai trò điều khiển, trong khi máy ảo 2 được sử dụng như môi trường build.

5. Kiểm tra tính khả thi của dự án trước khi triển khai thực tế: Mô hình này có thể được sử dụng để kiểm tra các bước triển khai, build và chạy dự án từ GitHub trong một môi trường an toàn trước khi đưa vào môi trường thực tế.

4.4. Kịch bản 4: Triển khai CSDL MySQL và sao lưu dữ liệu của CSDL đó.

4.4.1. Ngữ cảnh và mô hình

Trong kịch bản này, nhóm sẽ triển khai một mô hình gồm 2 máy chủ sử dụng KVM, trong đó:

- Một máy ảo đóng vai trò là **Database Server** được sử dụng để chứa CSDL MySQL, đồng thời đóng vai trò lưu trữ chính, cho phép Client thực hiện các thao tác CRUD (Create, Read, Update, Delete).

- Máy ảo còn lại sẽ có vai trò là **Backup Server**, sử dụng TLS/SSL để mã hóa kết nối giữa 2 máy, máy này chỉ có quyền Read trên CSDL và sẽ liên tục sao lưu bản sao của CSDL.

4.4.2. Triển khai

Bước 1: Tạo chứng chỉ SSL cho MySQL: ta thực hiện lần lượt 8 lệnh sau trên Database Server

- Tạo một khóa riêng tư 2048-bit cho CA và lưu vào file ca-key.pem

```
openssl genrsa -out ca-key.pem 2048
```

- Tạo chứng chỉ tự ký (self-signed certificate) cho CA, dùng khóa riêng tư ca-key.pem.

Chứng chỉ CA được lưu trong file ca-cert.pem

```
openssl req -new -x509 -nodes -days 365 \
-key ca-key.pem -out ca-cert.pem \
-subj "/C=VN/ST=HCM/L=HCM/O=UIT/OU=UIT/CN=MySQL-CA"
```

- Tạo một khóa riêng tư 2048-bit cho Database Server, lưu trong file server-key.pem

```
openssl genrsa -out server-key.pem 2048
```

- Tạo yêu cầu chứng chỉ (Certificate Signing Request - CSR) cho Database Server, sử dụng khóa riêng tư server-key.pem. CSR được lưu trong file server-req.pem

```
openssl req -new -key server-key.pem -out server-req.pem \
-subj "/C=VN/ST=HCM/L=HCM/O=UIT/OU=UIT/CN=192.168.122.44"
```

- Ký yêu cầu chứng chỉ của MySQL Server (server-req.pem) bằng CA (file ca-cert.pem và ca-key.pem). Chứng chỉ cho Database Server được lưu trong file server-cert.pem

```
openssl x509 -req -in server-req.pem -days 365 \
-CA ca-cert.pem -CAkey ca-key.pem -CAcreateserial \
-out server-cert.pem
```

- Tạo yêu cầu chứng chỉ (CSR) cho Backup Server, sử dụng khóa riêng tư client-key.pem. CSR được lưu trong file client-req.pem

```
openssl genrsa -out client-key.pem 2048
openssl req -new -key client-key.pem -out client-req.pem \
-subj "/C=VN/ST=HCM/L=HCM/O=UIT/OU=UIT/CN=namphuong"
```

- Ký yêu cầu chứng chỉ của Backup Server (client-req.pem) bằng CA (file ca-cert.pem và ca-key.pem). Chứng chỉ cho client được lưu trong file client-cert.pem

```
openssl x509 -req -in client-req.pem -days 365 \
-CA ca-cert.pem -CAkey ca-key.pem -CAcreateserial \
-out client-cert.pem
```

- Ngoài ra hãy thử kiểm tra xem các file đã được ký đúng hay chưa:

```
namphuong@npdb:/etc/mysql/ssl$ openssl verify -CAfile /etc/mysql/ssl/ca-cert.pem /etc/mysql/ssl/server-cert.pem
/etc/mysql/ssl/server-cert.pem: OK
namphuong@npdb:/etc/mysql/ssl$ openssl verify -CAfile /etc/mysql/ssl/ca-cert.pem /etc/mysql/ssl/client-cert.pem
/etc/mysql/ssl/client-cert.pem: OK
```

Hình 34. Kiểm tra các file đã được ký đúng hay chưa

Bước 2: Thêm quyền cho các chứng chỉ SSL vừa được tạo

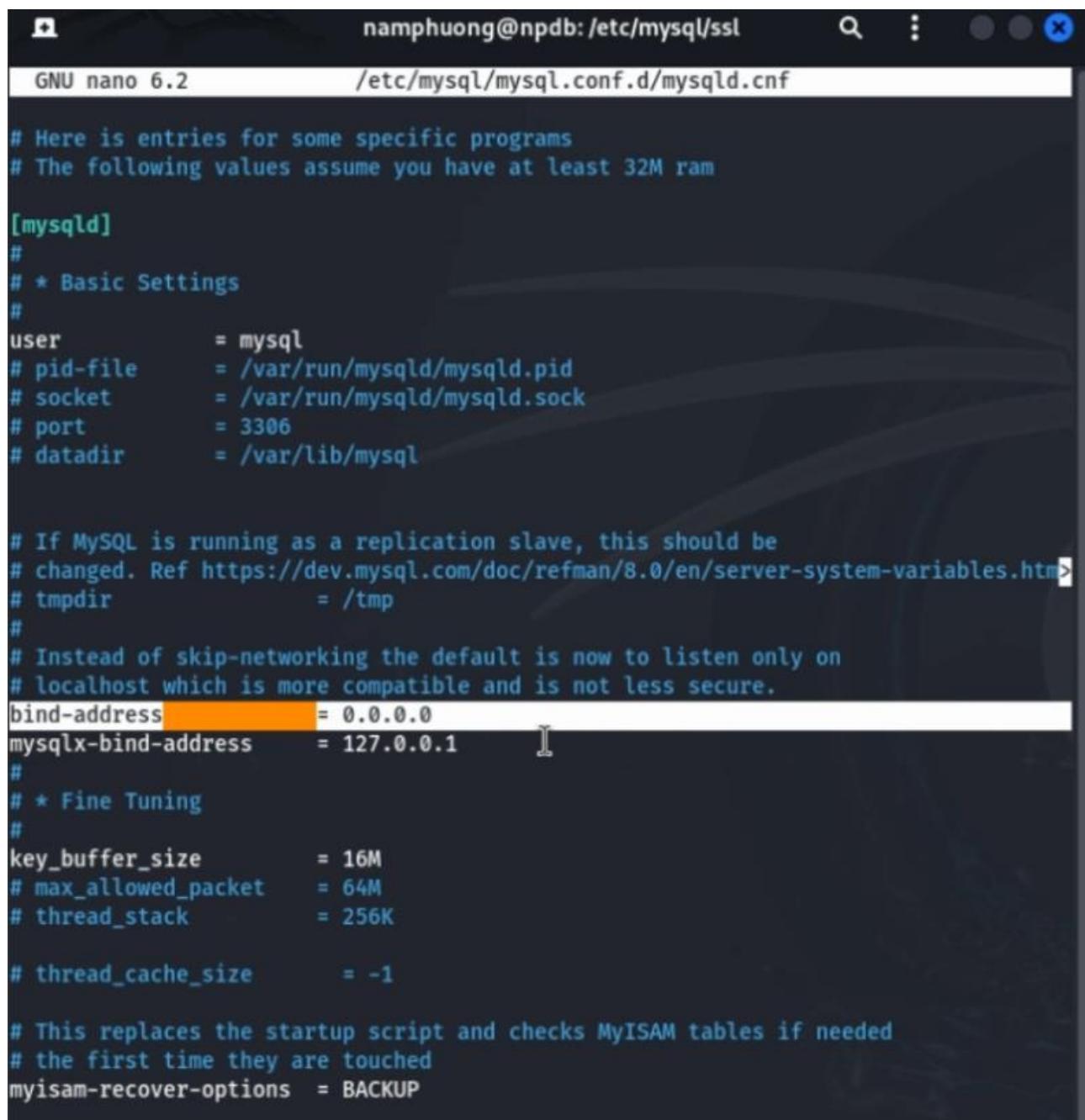
```
sudo chown mysql:mysql *.pem
sudo chmod 777 *.pem
```

Bước 3: Thực hiện cấu hình MySQL

- Mở file mysqld.cnf ở đường dẫn:

```
sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf
```

- Đầu tiên ta cần thực hiện thay đổi giá trị của bind-address thành 0.0.0.0, mục đích là để cho phép MySQL lắng nghe kết nối từ tất cả IP



```
namphuong@npdb: /etc/mysql/ssl
GNU nano 6.2          /etc/mysql/mysql.conf.d/mysqld.cnf

# Here is entries for some specific programs
# The following values assume you have at least 32M ram

[mysqld]
#
# * Basic Settings
#
user          = mysql
# pid-file     = /var/run/mysqld/mysqld.pid
# socket       = /var/run/mysqld/mysqld.sock
# port         = 3306
# datadir      = /var/lib/mysql

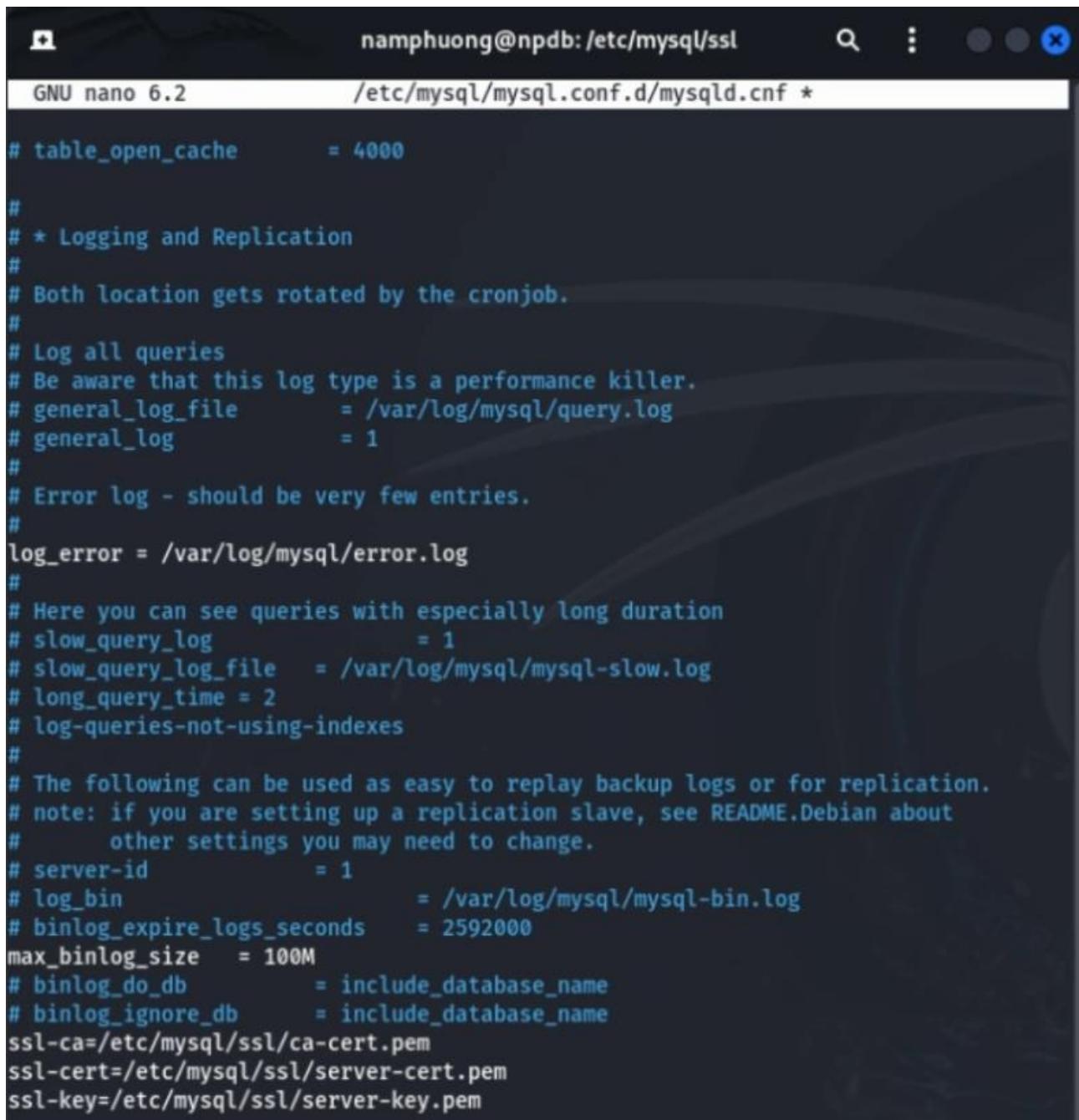
# If MySQL is running as a replication slave, this should be
# changed. Ref https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.htm
# tmpdir        = /tmp
#
# Instead of skip-networking the default is now to listen only on
# localhost which is more compatible and is not less secure.
bind-address    = 0.0.0.0
mysqlx-bind-address = 127.0.0.1
#
# * Fine Tuning
#
key_buffer_size   = 16M
# max_allowed_packet = 64M
# thread_stack      = 256K

# thread_cache_size    = -1

# This replaces the startup script and checks MyISAM tables if needed
# the first time they are touched
myisam-recover-options = BACKUP
```

Hình 35. Cấu hình bind-address cho MySQL

- Thêm 3 dòng sau vào file để kích hoạt và cấu hình SSL/TLS cho máy chủ MySQL



```
namphuong@npdb:/etc/mysql/ssl          :  X
GNU nano 6.2      /etc/mysql/mysql.conf.d/mysqld.cnf *

# table_open_cache      = 4000

#
# * Logging and Replication
#
# Both location gets rotated by the cronjob.
#
# Log all queries
# Be aware that this log type is a performance killer.
# general_log_file      = /var/log/mysql/query.log
# general_log            = 1
#
# Error log - should be very few entries.
#
log_error = /var/log/mysql/error.log
#
# Here you can see queries with especially long duration
# slow_query_log         = 1
# slow_query_log_file    = /var/log/mysql/mysql-slow.log
# long_query_time        = 2
# log-queries-not-using-indexes
#
# The following can be used as easy to replay backup logs or for replication.
# note: if you are setting up a replication slave, see README.Debian about
#       other settings you may need to change.
# server-id              = 1
# log_bin                 = /var/log/mysql/mysql-bin.log
# binlog_expire_logs_seconds = 2592000
max_binlog_size        = 100M
# binlog_do_db            = include_database_name
# binlog_ignore_db         = include_database_name
ssl-ca=/etc/mysql/ssl/ca-cert.pem
ssl-cert=/etc/mysql/ssl/server-cert.pem
ssl-key=/etc/mysql/ssl/server-key.pem
```

Hình 36. Kích hoạt SSL/TLS cho MySQL

- Thực hiện lệnh: `sudo systemctl restart mysql` để khởi động lại MySQL nhằm lưu những thay đổi của chúng ta

Bước 4: Tạo người dùng cho Backup Server chỉ cấp quyền thực hiện truy vấn

- Đăng nhập vào MySQL bằng người dùng root:

```
sudo mysql -u root -p
```

Thực hiện lần lượt các lệnh sau:

- Tạo một người dùng tên backup_user được phép kết nối từ bất cứ địa chỉ nào, có thể đăng nhập với mật khẩu 'namPhuong123@' và chỉ được kết nối bằng SSL

```
CREATE USER 'backup_user'@'%' IDENTIFIED BY 'namPhuong123@' REQUIRE SSL
```

- Gán quyền cho người dùng backup_user chỉ được thực hiện lệnh truy vấn trên tất cả CSDL và bảng

```
GRANT SELECT ON *.* TO 'backup_user'@'%';
```

```
FLUSH PRIVILEGES;
```

- Tải lại bảng quyền của MySQL ngay lập tức để các thay đổi vừa thực hiện có hiệu lực

```
mysql> CREATE USER 'backup_user'@'%' IDENTIFIED BY 'namPhuong123@' REQUIRE SSL;
Query OK, 0 rows affected (0.27 sec)

mysql> GRANT SELECT ON *.* TO 'backup_user'@'%';
Query OK, 0 rows affected (0.03 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.03 sec)
```

Hình 37. Phân quyền user trong MySQL

- Thực hiện xem lại những người dùng hiện có trong MySQL như sau:

```
mysql> SELECT User, Host FROM mysql.user;
+-----+-----+
| User      | Host   |
+-----+-----+
| backup_user | %     |
| debian-sys-maint | localhost |
| mysql.infoschema | localhost |
| mysql.session | localhost |
| mysql.sys    | localhost |
| root        | localhost |
+-----+-----+
6 rows in set (0.00 sec)
```

Hình 38. Kiểm tra user đã tồn tại chưa

Bước 5: Thực hiện lưu các chứng chỉ từ Database Server về Backup Server (có nhiều cách, ví dụ như tạo file rồi paste giá trị chứng chỉ, copy chứng chỉ từ ip...).

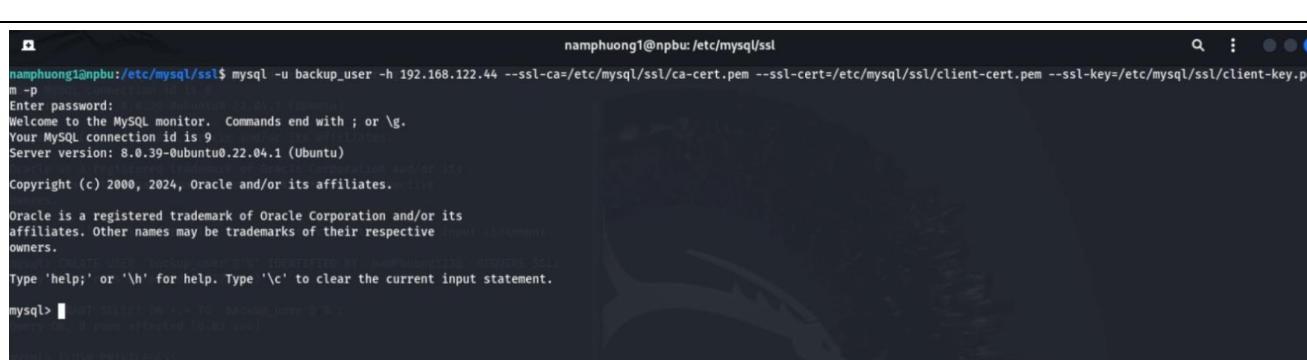
- Ở đây nhóm dùng lệnh scp trên linux để lấy file. Ngoài ra sau khi có file thì ta sẽ cấp quyền cho các file này

```
namphuong1@npbu:~$ sudo scp namphuong@192.168.122.44:/home/namphuong/{ca-cert.pem,client-cert.pem,client-key.pem} /etc/mysql/ssl/  
  
[sudo] password for namphuong1:  
namphuong@192.168.122.44's password:  
ca-cert.pem                                         100% 1298   101.6KB/s  00:00  
namphuong@192.168.122.44's password:  
Permission denied, please try again.  
namphuong@192.168.122.44's password:  
client-cert.pem                                     100% 1176   130.1KB/s  00:00  
namphuong@192.168.122.44's password:  
client-key.pem                                      100% 1704   197.4KB/s  00:00  
namphuong1@npbu:~$ cd /etc/mysql/ssl  
namphuong1@npbu:/etc/mysql/ssl$ ls  
ca-cert.pem  client-cert.pem  client-key.pem  
namphuong1@npbu:/etc/mysql/ssl$ sudo chown $(whoami):$(whoami) /etc/mysql/ssl/*.pem  
namphuong1@npbu:/etc/mysql/ssl$
```

Hình 39. Chuyển file từ Database Server sang Backup Server

Bước 6: Thực hiện kết nối vào MySQL bằng SSL/TLS trên Backup Server bằng cách:

```
mysql -u backup_user -h 192.168.122.44  
--ssl-ca=/etc/mysql/ssl/ca-cert.pem      --ssl-cert=/etc/mysql/ssl/client-cert.pem  
--ssl-key=/etc/mysql/ssl/client-key.pem -p
```



Hình 40. Backup Server thực hiện kết nối vào CSDL

- Có thể thấy bên Backup Server có quyền thấy toàn bộ CSDL và bảng của Database Server

```

namphuong@npdb:~$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 8.0.39-0ubuntu0.22.04.1 (Ubuntu)

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

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

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| my_company |
| mysql |
| performance_schema |
| sys |
| test |
+-----+
6 rows in set (0.00 sec)

mysql> select * from

```



```

namphuong1@nphuong1:~$ mysql -u backup_user -h 192.168.44.44
--ssl-ca=/etc/mysql/ssl/ca-cert.pem --ssl-cert=/etc/mysql/ssl/client-cert.pem --ssl-key=/etc/mysql/ssl/client-key.pem -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 8.0.39-0ubuntu0.22.04.1 (Ubuntu)

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

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

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| my_company |
| mysql |
| performance_schema |
| sys |
| test |
+-----+
6 rows in set (0.02 sec)

mysql>

```

Hình 41. Backup Server có quyền thấy toàn bộ CSDL và bảng của Database Server

- Để thấy các giá trị bên trong CSDL và bảng cũng được cập nhật cho bên Backup Server theo thời gian thực

```

namphuong@npdb:~$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 8.0.39-0ubuntu0.22.04.1 (Ubuntu)

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

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

mysql> use test;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
Database changed
mysql> select * from test;
+---+
| id |
+---+
| 1 |
| 10 |
| 2 |
| 3 |
| 4 |
| 5 |
| 7 |
| 9 |
+---+
8 rows in set (0.01 sec)

mysql> insert into test value '15';
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ''15'' at line 1
mysql> insert into test values '15';
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ''15'' at line 1
mysql> insert into test values (15);
Query OK, 1 row affected (0.05 sec)

mysql> select * from test;
+---+
| id |
+---+
| 1 |
| 10 |
| 15 |
| 2 |
| 3 |
| 4 |
| 5 |
| 7 |
| 9 |
+---+
9 rows in set (0.00 sec)

mysql>

```



```

namphuong1@nphuong1:~$ mysql -u backup_user -h 192.168.44.44
--ssl-ca=/etc/mysql/ssl/ca-cert.pem --ssl-cert=/etc/mysql/ssl/client-cert.pem --ssl-key=/etc/mysql/ssl/client-key.pem -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 8.0.39-0ubuntu0.22.04.1 (Ubuntu)

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

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

mysql> select * from test;
+---+
| id |
+---+
| 1 |
| 10 |
| 15 |
| 2 |
| 3 |
| 4 |
| 5 |
| 7 |
| 9 |
+---+
9 rows in set (0.00 sec)

mysql>

```

Hình 42. Các giá trị bên trong CSDL và bảng cũng được cập nhật theo thời gian thực

- Kiểm thử các lệnh khác trên Backup Server (không được phép như ta đã cấu hình)

The screenshot shows two terminal windows side-by-side. The left window is titled 'namphuong@npdb:' and the right window is titled 'namphuong1@npbu: /etc/mysql/ssl'. Both windows show MySQL command-line interface.

Left Terminal (namphuong@npdb):

```
mysql> select * from test;
+---+
| id |
+---+
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |
| 6 |
| 7 |
| 8 |
| 9 |
+---+
8 rows in set (0.01 sec)

mysql> insert into test value '15';
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ''15'' at line 1
mysql> insert into test values '15';
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ''15'' at line 1
mysql> insert into test values (15);
Query OK, 1 row affected (0.05 sec)

mysql> select * from test;
+---+
| id |
+---+
| 1 |
| 10 |
| 15 |
| 2 |
| 3 |
| 4 |
| 5 |
| 7 |
| 9 |
+---+
9 rows in set (0.00 sec)

mysql>
```

Right Terminal (namphuong1@npbu: /etc/mysql/ssl):

```
mysql> select * from test;
+---+
| id |
+---+
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |
| 6 |
| 7 |
| 8 |
| 9 |
+---+
8 rows in set (0.00 sec)

mysql> select * from test;
+---+
| id |
+---+
| 1 |
| 10 |
| 15 |
| 2 |
| 3 |
| 4 |
| 5 |
| 7 |
| 9 |
+---+
9 rows in set (0.00 sec)

mysql> insert into test values (20);
ERROR 1142 (42000): INSERT command denied to user 'backup_user'@'192.168.122.84' for table 'test'
mysql> delete from test where id = 15;
ERROR 1142 (42000): DELETE command denied to user 'backup_user'@'192.168.122.84' for table 'test'
mysql>
```

Hình 43. Kiểm thử kết quả

Bước 7: Tạo một file .sh có nội dung như sau:

- Thực hiện config các thông tin của Database Server
- Tạo thư mục để ta sao lưu dữ liệu
- Thực hiện truy vấn vào cơ sở dữ liệu để lấy dữ liệu cần lưu
- Ta sẽ lưu các file backup dưới dạng.zip cho đỡ tiêu tốn tài nguyên

The screenshot shows a terminal window titled 'namphuong1@npbu: ~' with a file named 'backup.sh' open in 'GNU nano 6.2'. The script contains the following code:

```
#!/bin/bash

# Thông tin kết nối cơ sở dữ liệu
DB_HOST="192.168.122.44"      # Ví dụ: 192.168.1.100
DB_PORT="3306"                  # Mặc định của MySQL là 3306
DB_NAME="test"
DB_USER="backup_user"
DB_PASSWORD="namPhuong123@"
BACKUP_DIR="/home/namphuong1/backup" # Thư mục để lưu sao lưu
TIMESTAMP=$(date +"%Y%m%d_%H%M")

# Tạo thư mục mới cho mỗi lần sao lưu
mkdir -p "$BACKUP_DIR/$TIMESTAMP"

# Lấy danh sách các bảng trong cơ sở dữ liệu
TABLES=$(mysql -h $DB_HOST -P $DB_PORT -u $DB_USER -p$DB_PASSWORD -D $DB_NAME >
          | sed 's/\t/,/g' > "$BACKUP_DIR/$TIMESTAMP/$TABLE.csv"
done

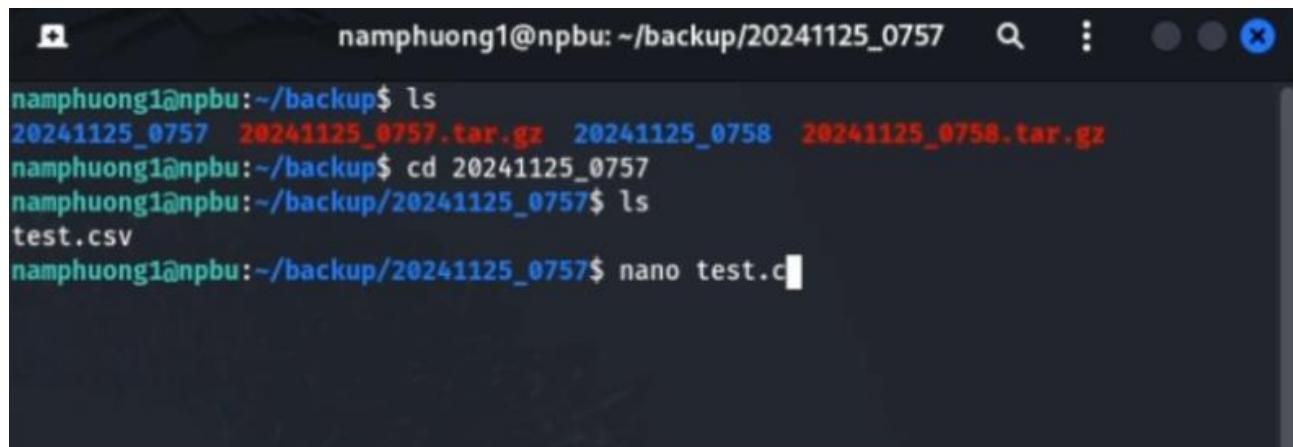
# Tùy chọn: Nén thư mục sao lưu để tiết kiệm dung lượng
tar -czf "$BACKUP_DIR/$TIMESTAMP.tar.gz" -C "$BACKUP_DIR" "$TIMESTAMP"
rm -rf "$BACKUP_DIR/$TIMESTAMP" # Xóa thư mục không nén
```

Bước 8: Vào crontab và thêm vào dòng sau:

- Crontab (viết tắt của "cron table") là một tập tin hoặc lệnh dùng để lên lịch thực hiện các tác vụ tự động trên hệ thống UNIX hoặc Linux. Các tác vụ này được gọi là cron jobs và được thực thi bởi cron daemon – một chương trình chạy nền chịu trách nhiệm quản lý và thực thi các công việc đã được lên lịch.

```
*/1 * * * * /home/namphuong1/backup.sh  
# Mỗi phút một lần, hệ thống sẽ tự động chạy script backup.sh
```

4.4.3. Kết quả



The screenshot shows a terminal window with the following session:

```
namphuong1@npbu: ~/backup/20241125_0757
namphuong1@npbu:~/backup$ ls
20241125_0757  20241125_0757.tar.gz  20241125_0758  20241125_0758.tar.gz
namphuong1@npbu:~/backup$ cd 20241125_0757
namphuong1@npbu:~/backup/20241125_0757$ ls
test.csv
namphuong1@npbu:~/backup/20241125_0757$ nano test.c
```

Từ phía Database Server:

- CSDL luôn sẵn sàng phục vụ client với các thao tác CRUD
- Backup server có thể kết nối và truy xuất dữ liệu mà không gặp lỗi quyền truy cập
- Kết nối giữa Database Server và các user (client hoặc backup server) được mã hóa, giảm thiểu nguy cơ bị tấn công trung gian (MITM)
- Không có người dùng trái phép nào truy cập được vào hệ thống, nhờ vào phân quyền và giới hạn IP
- Máy chủ hoạt động mà không bị quá tải hoặc bị ảnh hưởng bởi các tác vụ backup

Từ phía Backup Server:

- Backup server tự động lấy toàn bộ dữ liệu từ Database Server và lưu trữ dưới dạng .zip
- File backup được đặt trong thư mục định sẵn với tên rõ ràng và có thể phục hồi khi cần

- Các bản backup không bị thiếu dữ liệu hoặc bị hỏng nhờ thực hiện đúng kỹ thuật
- Backup server chỉ có quyền SELECT, bảo đảm không thể thay đổi hoặc xóa dữ liệu trên Database Server

4.4.4. *Ứng dụng*

Kịch bản triển khai cơ sở dữ liệu (CSDL) kết hợp với backup là một mô hình thiết yếu được ứng dụng rộng rãi trong nhiều lĩnh vực, nhằm đảm bảo dữ liệu luôn sẵn sàng, an toàn và có thể phục hồi khi cần thiết:

- 1. Đảm Bảo Tính Sẵn Sàng Của Dữ Liệu:** Đảm bảo dữ liệu quan trọng luôn sẵn sàng để phục vụ các hoạt động hàng ngày, ngay cả khi hệ thống gặp sự cố
- 2. Khắc Phục Sự Cố:** Hỗ trợ khôi phục dữ liệu nhanh chóng trong nhiều trường hợp
- 3. Tăng Cường Bảo Mật Dữ Liệu:** Bảo vệ dữ liệu khỏi các rủi ro bị xâm phạm hay mất mát trong quá trình xử lý
- 4. Hỗ Trợ Phân Tích Dữ Liệu:** Cho phép sử dụng dữ liệu backup để phân tích, báo cáo hoặc kiểm thử mà không ảnh hưởng đến CSDL chính
- 5. Phục Vụ Sao Lưu Dài Hạn:** Lưu giữ dữ liệu trong thời gian dài để tuân thủ các quy định hoặc cho các mục đích nghiên cứu

4.5. Kịch bản 5: Xây dựng mạng nội bộ gồm Web Server, Data Server và các Client.

4.5.1. Ngữ cảnh và mô hình

Trong thời đại số hóa và kết nối toàn cầu, nhu cầu triển khai các hệ thống mạng nội bộ hiệu quả, bảo mật và có khả năng mở rộng là một thách thức đối với các tổ chức và doanh nghiệp. Trong kịch bản này, mục tiêu của nhóm là xây dựng một mô hình mạng nội bộ, sử dụng công nghệ ảo hóa KVM (Kernel-based Virtual Machine) để đảm bảo tính linh hoạt, hiệu quả trong việc quản lý tài nguyên, và khả năng tích hợp với các hệ thống hiện có.

Mạng nội bộ bao gồm một hạ tầng máy chủ nội bộ mạnh mẽ, trong đó các máy chủ ảo được triển khai để thực hiện các nhiệm vụ riêng biệt. Các máy chủ này được triển khai trên nền tảng KVM, một giải pháp ảo hóa cấp độ kernel được tích hợp trong Linux, giúp tận dụng tối đa hiệu năng phần cứng. Hệ thống mạng nội bộ có ba thành phần chính:

- Web Server:** Một máy ảo KVM chạy hệ điều hành Ubuntu 20.04. Chịu trách nhiệm cung cấp dịch vụ web cho các ứng dụng nội bộ, đảm bảo truy cập nhanh chóng và bảo mật.
- Data Server:** Một máy ảo KVM chạy hệ điều hành Ubuntu Server 20.04 chứa cơ sở dữ liệu nội bộ, lưu trữ thông tin và phục vụ dữ liệu cho các ứng dụng web.
- Client:** Máy ảo KVM khác chạy hệ điều hành Ubuntu 20.04.06, có thể truy cập vào các website nội bộ để thực hiện công việc được giao.

4.5.2. Triển khai

*** Xây dựng Web Server**

Cài đặt KVM và tạo máy ảo để làm Web Server với cấu hình như sau:

OS	Ubuntu Desktop 20.04.06 (LTS)
CPU	8 Cores, 1 Processors
RAM	4GB
Disk	50GB
IP Address	192.168.122.203

Chuẩn bị trước khi triển khai Web Server:

- Tạo thư mục myproject để chứa mã nguồn:

```
cd /opt  
mkdir myproject
```

```
root@nhom6:/home/dducktai# cd /opt  
root@nhom6:/opt# ls  
myproject selfsigned.crt selfsigned.key  
root@nhom6:/opt#
```

Hình 44. Tạo thư mục chứa mã nguồn

- Sao chép mã nguồn vào thư mục /opt/myproject:

```
git clone https://github.com/dducktai/Project-NT132.P12.ANTT.git  
/opt/myproject
```

```
root@nhom6:/opt# cd myproject  
root@nhom6:/opt/myproject# ls  
config db.sqlite3 manage.py members README.md selfsigned.crt staticfiles venv  
convert.py LICENSE media Pipfile requirements.txt static templates
```

Hình 45. Sao chép mã nguồn vào thư mục /opt/myproject

- Thiết lập môi trường ảo Python để quản lý các gói độc lập:

```
python3 -m venv /opt/myproject/venv  
source /opt/myproject/venv/bin/activate
```

```
root@nhom6:/opt/myproject# ls  
config db.sqlite3 manage.py members README.md selfsigned.crt staticfiles venv  
convert.py LICENSE media Pipfile requirements.txt static templates  
root@nhom6:/opt/myproject# source /opt/myproject/venv/bin/activate
```

Hình 46. Thiết lập môi trường ảo Python để quản lý các gói độc lập

- Cài đặt các gói cần thiết, bao gồm cả Gunicorn để chạy ứng dụng và Whitenoise để phục vụ static files:

```
pip install -r /opt/myproject/requirements.txt  
pip install gunicorn  
pip install whitenoise
```

```
(venv) root@nhom6:/opt/myproject# pip install -r /opt/myproject/requirements.txt
Requirement already satisfied: asgiref==3.8.1 in /usr/lib/python3.8/site-packages (from -r /opt/myproject/requirements.txt (line 1)) (3.8.1)
Requirement already satisfied: Django==4.2.16 in /usr/lib/python3.8/site-packages (from -r /opt/myproject/requirements.txt (line 2)) (4.2.16)
Requirement already satisfied: pillow==10.4.0 in /usr/lib/python3.8/site-packages (from -r /opt/myproject/requirements.txt (line 3)) (10.4.0)
Requirement already satisfied: sqlparse==0.5.1 in /usr/lib/python3.8/site-packages (from -r /opt/myproject/requirements.txt (line 4)) (0.5.1)
Requirement already satisfied: tzdata==2024.1 in /usr/lib/python3.8/site-packages (from -r /opt/myproject/requirements.txt (line 5)) (2024.1)
Requirement already satisfied: whitenoise==6.7.0 in /usr/lib/python3.8/site-packages (from -r /opt/myproject/requirements.txt (line 6)) (6.7.0)
Requirement already satisfied: typing-extensions>=4; python_version < "3.11" in /usr/lib/python3.8/site-packages (from asgiref==3.8.1->-r /opt/myproject/requirements.txt (line 1)) (4.12.2)
Requirement already satisfied: backports.zoneinfo; python_version < "3.9" in /usr/lib/python3.8/site-packages (from Django==4.2.16->-r /opt/myproject/requirements.txt (line 2)) (0.2.1)
```

Hình 47. Cài đặt các gói cần thiết

```
(venv) root@nhom6:/opt/myproject# gunicorn --version
gunicorn (version 23.0.0)
(venv) root@nhom6:/opt/myproject# pip3 install whitenoise
Requirement already satisfied: whitenoise in /usr/lib/python3.8/site-packages (6.7.0)
```

Hình 48. Cài đặt các gói cần thiết

Cấu hình Django Project

- Sửa file settings.py:

```
nano /opt/myproject/config/settings.py
```

- Thêm vào MIDDLEWARE: Giúp Django sử dụng Whitenoise để quản lý static files, tăng tốc độ truy cập.

```
'django.contrib.messages.middleware.MessageMiddleware',
'django.middleware.clickjacking.XFrameOptionsMiddleware',
[Whitenoise.middleware.WhiteNoiseMiddleware],
```

Hình 49. Cấu hình MiddleWare

Cấu hình static files và media files

- Định nghĩa đường dẫn URL để truy cập các tệp tĩnh và tệp media. Sử dụng Whitenoise để nén các static files, tối ưu hóa hiệu suất.

```
STATIC_URL = 'static/'
STATICFILES_DIRS = [str(BASE_DIR.joinpath('static'))]
STATIC_ROOT = BASE_DIR / 'staticfiles'

MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')

STATICFILES_STORAGE = "whitenoise.storage.CompressedManifestStaticFilesStorage"
```

Hình 50. Cấu hình Static files và Media files

- Thu thập tất cả static files của dự án và lưu trữ vào thư mục STATIC_ROOT.

```
python manage.py collectstatic
```

```
(venv) root@nhom6:/opt/myproject# ls
config      db.sqlite3  manage.py  members  README.md      selfsigned.crt  staticfiles  venv
convert.py   LICENSE     media     Pipfile   requirements.txt  static  templates
```

Hình 51. Thu thập tất cả static files của dự án

Cấu hình Nginx:

- Cài đặt Nginx: để làm reverse proxy server

```
sudo apt update
sudo apt install nginx
```

```
(venv) root@nhom6:/opt/myproject# nginx -v
nginx version: nginx/1.18.0 (Ubuntu)
```

Hình 52. Kiểm tra phiên bản Nginx

- Tạo file cấu hình cho Nginx, chỉ định các cài đặt cần thiết để phục vụ ứng dụng:

```
sudo nano /etc/nginx/sites-available/myproject

#Nội dung file:
server {
    listen 80;
    server_name 192.168.122.203;

    location /static/ {
        alias /opt/myproject/staticfiles/;
    }

    location /media/ {
        alias /opt/myproject/media/;
    }

    location / {
        proxy_pass http:// 192.168.122.203:8000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }
}
```

- Kích hoạt cấu hình Nginx bằng cách tạo liên kết trong sites-enabled:

```
sudo ln -s /etc/nginx/sites-available/myproject /etc/nginx/sites-enabled/
```

```
(venv) root@nhom6:/opt/myproject# sudo nano /etc/nginx/sites-available/myproject
(venv) root@nhom6:/opt/myproject# cd /etc/nginx/sites-enabled
(venv) root@nhom6:/etc/nginx/sites-enabled# ls
default myproject
```

Hình 53. Kích hoạt cấu hình Nginx

- Chính sửa file cấu hình Nginx: Bỏ comment dòng “server_names_hash_bucket_size 64;”. Dòng này giúp Nginx xử lý các tên miền dài.

```
server_names_hash_bucket_size 64;

# server_tokens off;

server_names_hash_bucket_size 64;
# server_name_in_redirect off;

include /etc/nginx/mime.types;】
```

Hình 54. Xử lý tên miền

- Khởi động lại Nginx: Áp dụng các thay đổi trong cấu hình Nginx.

```
sudo service nginx restart
```

Cấu hình tường lửa:

- Cài đặt ufw:

```
sudo apt update
sudo apt-get install ufw
```

- Mở các cổng cần thiết cho server:

- + **8000**: Dùng để Gunicorn hoạt động.
- + **80**: Dùng cho HTTP.
- + **443**: Dùng cho HTTPS.
- + **3306**: Dùng cho MySQL

```
sudo ufw allow 8000
sudo ufw allow 80
sudo ufw allow 443
sudo ufw allow 3306
```

```
(venv) root@nhom6:/etc/nginx/sites-enabled# ufw status
Status: active

To                         Action    From
--                         ----     --
8000                        ALLOW     Anywhere
5353/udp                   ALLOW     Anywhere
443                         ALLOW     Anywhere
443/tcp                     ALLOW     Anywhere
3306                        ALLOW     Anywhere
3306/tcp                   ALLOW     Anywhere
8000 (v6)                  ALLOW     Anywhere (v6)
5353/udp (v6)              ALLOW     Anywhere (v6)
443 (v6)                    ALLOW     Anywhere (v6)
443/tcp (v6)               ALLOW     Anywhere (v6)
3306 (v6)                  ALLOW     Anywhere (v6)
3306/tcp (v6)              ALLOW     Anywhere (v6)
```

Hình 55. Cấu hình tường lửa

- Khởi động lại nginx.
- Chạy ứng dụng và kiểm tra:

```
cd /opt/myproject
gunicorn --bind 0.0.0.0:8000 config.wsgi
```

```
(venv) dducktai@nhom6:/opt/myproject$ gunicorn --bind 0.0.0.0:8000 config.wsgi
[2024-11-25 17:10:30 +0700] [6429] [INFO] Starting gunicorn 23.0.0
[2024-11-25 17:10:30 +0700] [6429] [INFO] Listening at: http://0.0.0.0:8000 (6429)
[2024-11-25 17:10:30 +0700] [6429] [INFO] Using worker: sync
[2024-11-25 17:10:30 +0700] [6431] [INFO] Booting worker with pid: 6431
```

Hình 56. Khởi chạy thử Server

Cấu hình tên miền nội bộ:

- Sửa file /etc/hosts: Định nghĩa tên miền nội bộ nhom6.local trỏ tới IP server.

```
sudo nano /etc/hosts
#Thêm dòng
192.168.122.203 nhom6.local
```

```
127.0.0.1 localhost
127.0.1.1 ubuntu
192.168.122.203 nhom6.local
```

Hình 57. Cấu hình tên miền nội bộ

- Cập nhật cấu hình Nginx:

```
sudo nano /etc/nginx/sites-available/myproject
#Thay IP bằng tên miền nhom6.local để sử dụng domain nội bộ.
```

```
#Nội dung file:
server {

    listen 80;
    server_name nhom6.local

    location /static/ {
        alias /opt/myproject/staticfiles/;
    }

    location /media/ {
        alias /opt/myproject/media/;
    }

    location / {
        proxy_pass http://nhom6.local:8000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }
}
```

- Khởi động lại Nginx.

Cấu hình HTTPS:

- Tạo chứng chỉ tự ký:

```
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout
selfsigned.key -out selfsigned.crt
```

```
(venv) dduckta1@nhom6:/opt$ ls
myproject  selfsigned.crt  selfsigned.key
```

Hình 58. Tạo chứng chỉ tự ký

- Thêm cấu hình HTTPS vào file cấu hình của nginx: Kích hoạt HTTPS cho server bằng chứng chỉ tự ký.

```
sudo nano /etc/nginx/sites-available/myproject
```

#Nội dung file:

```
server {

    listen 80;
    server_name nhom6.local

    ssl_certificate /opt/selfsigned.crt;
```

```

ssl_certificate_key /opt/selfsigned.key;

ssl_protocols TLSv1.2 TLSv1.3;
ssl_ciphers HIGH:!aNULL:!MD5;

location /static/ {
    alias /opt/myproject/staticfiles/;
}

location /media/ {
    alias /opt/myproject/media/;
}

location / {
    proxy_pass http://nhom6.local:8000;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
}
}

```

- Cập nhật kho CA: Thêm chứng chỉ vào hệ thống để các máy khác có thể tin cậy khi kết nối qua HTTPS.

```

sudo cp /opt/selfsigned.crt /usr/local/share/ca-certificates/
sudo update-ca-certificates

```

```

(venv) dducktai@nhom6:/opt$ cd /usr/local/share/ca-certificates/
(venv) dducktai@nhom6:/usr/local/share/ca-certificates$ ls
selfsigned.crt

```

Hình 59. Kiểm tra chứng chỉ

- Khởi động lại Nginx và chạy ứng dụng.

Kết quả:



Hình 60. Web được khởi chạy thành công với giao thức HTTPS

*** Xây dựng Data Server:**

Như phần trên đã triển khai, chúng ta chỉ mới khởi chạy được WebApp, còn khi thao tác với cơ sở dữ liệu thì dữ liệu vẫn sẽ lưu vào file .sqlite3 theo mặc định của Django.

Vậy thì khá bất tiện khi mà chúng ta đang muốn triển khai 1 hệ thống web nội bộ theo như kịch bản. Do đó, nhóm chúng em sẽ tiếp tục tiến hành triển khai tiếp 1 server MySQL để lưu Cơ sở dữ liệu của Web này, gọi là Data Server

Tiếp tục tạo 1 máy ảo để làm Data Server với cấu hình như sau:

OS	Ubuntu Server 20.04 (LTS)
CPU	4 Cores, 1 Processors
RAM	4GB
Disk	64GB
IP Address	192.168.122.67

Cài đặt những thư viện cần thiết:

```
sudo apt install pkg-config python3-dev libmysqlclient-dev build-essential  
gcc python3-dev libmysqlclient-dev
```

Cấu hình MySQL và UFW

- Tại máy Data Server, ta đăng nhập vào MySQL và tạo 1 Cơ sở Dữ liệu:

```
mysql -u root -p  
CREATE DATABASE nt132_database;
```

- Tiếp theo, ta tạo 1 tài khoản để cho Web Django có thể kết nối đến CSDL có tên là adminDB và chỉ được phép truy cập từ địa chỉ IP 192.168.122.203, cùng với mật khẩu là 241203.

```
USE mysql;  
CREATE USER 'adminDB'@'192.168.122.203' IDENTIFIED BY '241203';
```

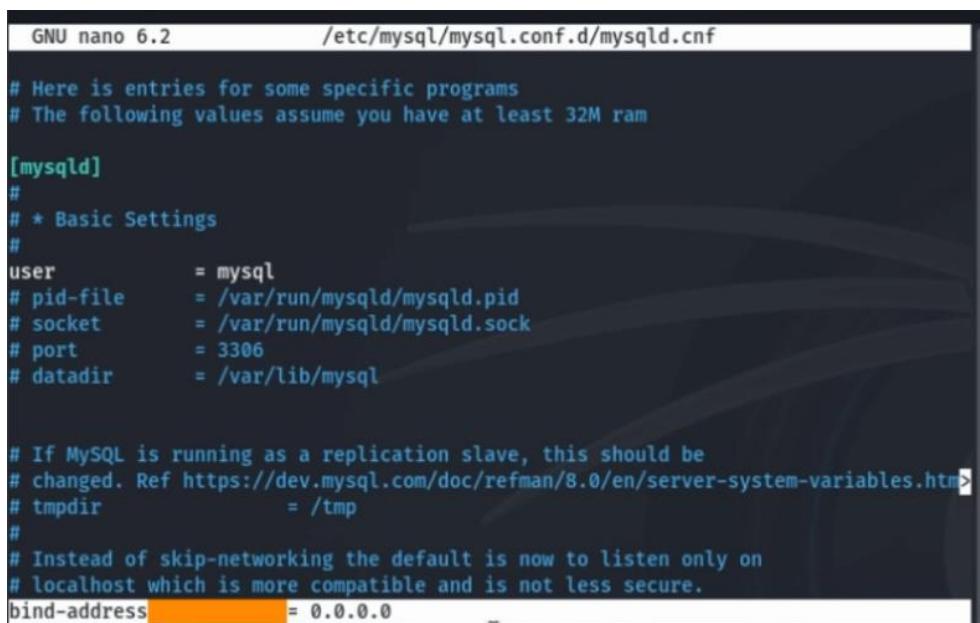
- Gán tất cả các quyền (bao gồm SELECT, INSERT, UPDATE, DELETE, CREATE, v.v.) trên tất cả các cơ sở dữ liệu (*.*) cho người dùng adminDB từ địa chỉ IP 192.168.122.203. Sau đó ta tải lại bảng quyền để các thay đổi được áp dụng ngay lập tức.

```
GRANT ALL PRIVILEGES ON * . * TO 'adminDB'@'192.168.122.203';
FLUSH PRIVILEGES;
```

- Tiếp theo, ta dừng MySQL trên máy Data Server để tiến hành thay đổi một vài thông số trong file mysqld.cnf:

```
sudo systemctl stop mysql
sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf
```

- Thay đổi bind-address = 127.0.0.1 thành 0.0.0.0 => Tùy chọn này trong file cấu hình mysqld.cnf xác định địa chỉ IP mà MySQL lắng nghe các kết nối. 0.0.0.0 có nghĩa là MySQL sẽ chấp nhận kết nối từ mọi địa chỉ IP (không giới hạn chỉ localhost hoặc địa chỉ cụ thể).



```
GNU nano 6.2          /etc/mysql/mysql.conf.d/mysqld.cnf

# Here is entries for some specific programs
# The following values assume you have at least 32M ram

[mysqld]
#
# * Basic Settings
#
user          = mysql
# pid-file     = /var/run/mysqld/mysqld.pid
# socket       = /var/run/mysqld/mysqld.sock
# port         = 3306
# datadir      = /var/lib/mysql

#
# If MySQL is running as a replication slave, this should be
# changed. Ref https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.htm
# tmpdir        = /tmp
#
# Instead of skip-networking the default is now to listen only on
# localhost which is more compatible and is not less secure.
bind-address  = 0.0.0.0
```

Hình 61. Cấu hình bind-address của MySQL

- Cho phép lưu lượng TCP qua cổng 3306, cổng mặc định của MySQL, thông qua tường lửa UFW (Uncomplicated Firewall) để đảm bảo rằng các máy khác trên mạng có thể kết nối với dịch vụ MySQL trên máy chủ này.

```
sudo ufw allow 3306/tcp
```

- Khởi động lại tường lửa UFW để áp dụng các thay đổi cấu hình vừa thực hiện, đồng thời khởi động dịch vụ MySQL để áp dụng các thay đổi đã thực hiện trong file cấu hình mysqld.cnf.

```
sudo systemctl restart ufw
sudo systemctl start mysql
```

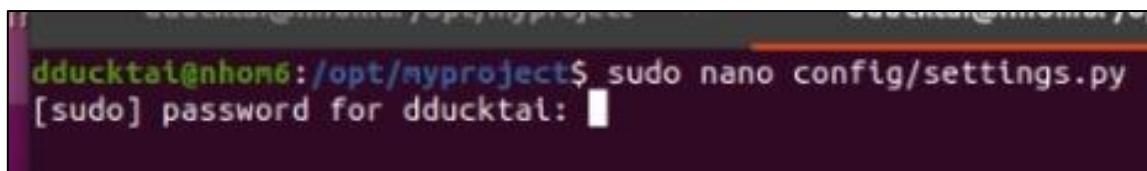
- Tiếp theo, ta tạm thời tắt Web Server đi, để tiến hành cấu hình CSDL trên máy này.

Cấu hình Database trên Django:

- Tại máy WebServer, ta đăng nhập vào CSDL của MySQL trên máy Data Server thông qua tài khoản adminDB đã được cấp (nhằm đảm bảo tài khoản đã được cấp thành công và hoạt động ổn định).

```
sudo mysql -u adminDB -p -h 192.168.122.67
```

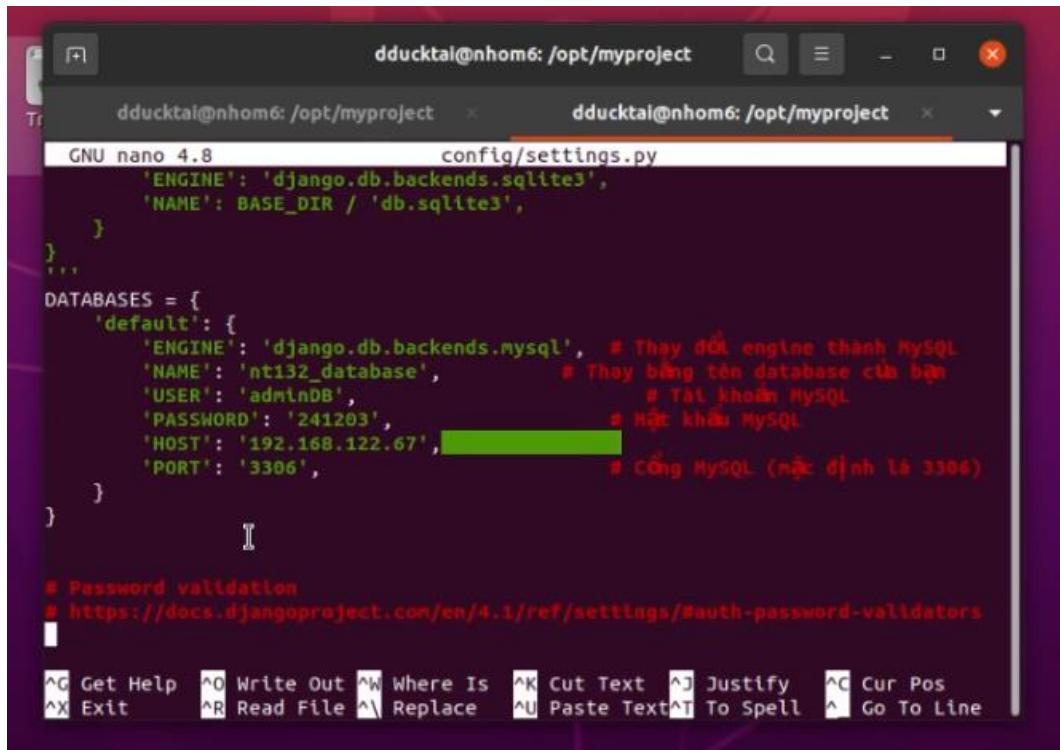
- Tiếp tục, Tại máy Web Server, ta tiến hành tùy chỉnh CSDL trong file settings.py của Django:



```
dducktai@nhom6:/opt/myproject$ sudo nano config/settings.py  
[sudo] password for dducktai: [REDACTED]
```

Hình 62. Cấu hình settings.py của dự án

- Ta cấu hình trong settings.py này kết nối Django với cơ sở dữ liệu MySQL bằng cách chỉ định loại cơ sở dữ liệu (ENGINE), tên cơ sở dữ liệu (NAME) là nt132_database, tài khoản MySQL (USER) là adminDB, mật khẩu (PASSWORD), địa chỉ IP của máy chủ (HOST) là 192.168.122.203, và cổng MySQL (PORT).



```
dducktai@nhom6:/opt/myproject$ nano config/settings.py  
# Thay đổi engine thành MySQL.  
# Thay bằng tên database của bạn.  
# Tài khoản MySQL.  
# Mật khẩu MySQL.  
# Cổng MySQL (mặc định là 3306).  
# Password validation  
# https://docs.djangoproject.com/en/4.1/ref/settings/#auth-password-validators
```

Hình 63. Cấu hình Database cho WebApp

- Cuối cùng, ta chạy tiếp lệnh sau: `python manage.py migrate` để thực thi các tệp migration => từ đó đồng bộ hóa cấu trúc cơ sở dữ liệu với các mô hình (models) đã được định nghĩa trong mã nguồn.

- Cụ thể, nó:

- + Tạo hoặc cập nhật các bảng trong cơ sở dữ liệu.
- + Áp dụng các thay đổi như thêm/sửa/xóa cột hoặc bảng.
- + Kích hoạt các tính năng cần thiết để ứng dụng Django làm việc với cơ sở dữ liệu.

- Sau bước này, ta đã hoàn thành việc triển khai 1 hệ thống Web nội bộ gồm Web Server và Data Server. Client thuộc nội bộ này đều có thể thao tác với CSDL một cách bình thường:

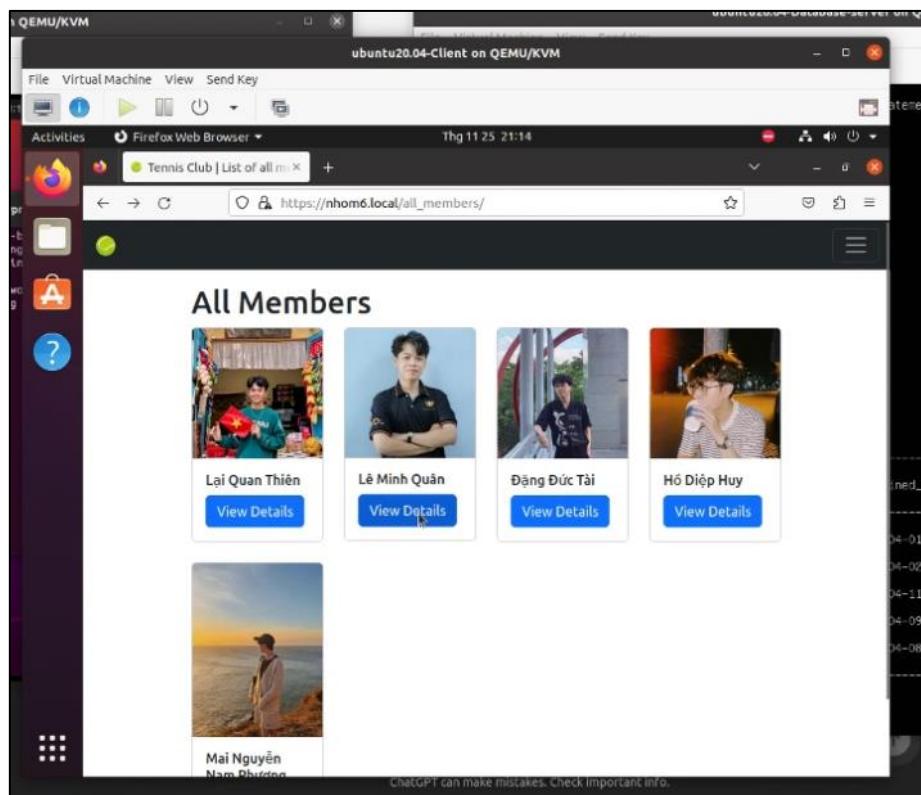
Kiểm tra:

- Bên phía Web Server, sử dụng tài khoản được cấp để xem CSDL:

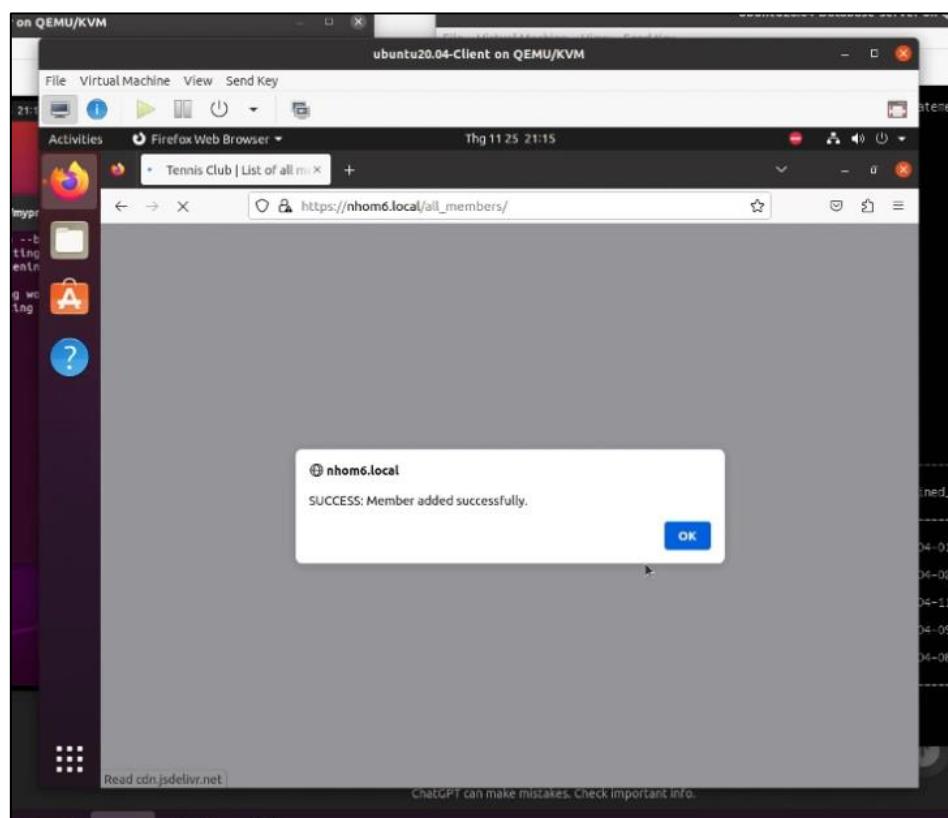
```
mysql> select * from members_member;
+----+-----+-----+-----+-----+-----+-----+
| id | firstname | lastname | email           | joined_date | slug          | bio      | phone_r |
| profile_image |
+----+-----+-----+-----+-----+-----+-----+
| 13 | Quan     | Le       | leminhquan@gmail.com | 2004-02-26  | quan-le      | UITer    | 0987654321
| members/Lê_Minhh_Quân11.jpg |
| 14 | Đặng Đức   | Tài      | dangductai@gmail.com | 2004-11-09  | ang-uc-tai    | UITer    | 0578564321
| members/ductai.jpg |
| 15 | Hồ Diệp   | Huy      | hodiephuy@gmail.com  | 2004-09-09  | ho-diep-huy   | UITer    | 0456398765
| members/HoHuy.jpg |
| 16 | Mai Nguyễn Nam | Phuong  | mainguyennamphuong@gmail.com | 2004-08-08  | mai-nguyen-nam-phuong | NP w TL | 0545458765
| members/namphuong.jpg |
+----+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

Hình 64. Xem CSDL bên phía Web Server

- Một máy Client thuộc mạng nội bộ này, có thể truy cập vào Web và xem dữ liệu được hiện lên, sau đó thực hiện thêm dữ liệu thành công:



Hình 65. Giao diện của Web với dữ liệu đã được load lên



Hình 66. Thao tác với CSDL thành công

4.5.3. Kết quả

Sau khi hoàn tất triển khai và cấu hình, mạng nội bộ cùng **Web Server** và **Data Server** đã hoạt động ổn định với những kết quả cụ thể:

- **Truy cập ứng dụng web:** Người dùng truy cập thành công vào ứng dụng thông qua tên miền nội bộ nhom6.local sử dụng giao thức HTTPS, đảm bảo bảo mật dữ liệu khi truyền qua mạng.

- **Data Server hoạt động ổn định:**

+ Cơ sở dữ liệu MySQL được triển khai và cấu hình chính xác. Các truy vấn từ ứng dụng web đến Data Server được thực hiện nhanh chóng và ổn định.

+ Cổng **3306** được mở để kết nối từ ứng dụng web nhưng vẫn đảm bảo an toàn nhờ cấu hình tường lửa nghiêm ngặt.

- **Tương thích giao tiếp:** Web Server (chạy Gunicorn và Nginx) và Data Server (MySQL) kết nối thông suốt, đáp ứng tốt các yêu cầu của ứng dụng.

- **Hiệu suất hệ thống:**

+ Web Server sử dụng **Gunicorn** và **Nginx**, phối hợp hiệu quả, giảm thiểu thời gian phản hồi.

+ Data Server xử lý tải truy vấn một cách ổn định, đảm bảo độ trễ thấp kể cả khi lượng dữ liệu tăng lên.

- **Tường lửa và bảo mật:** Cấu hình tường lửa UFW chỉ mở các cổng cần thiết (80, 443, 3306), giúp tăng cường bảo mật.

Kết quả cho thấy hệ thống Web Server và Data Server không chỉ đạt mục tiêu ban đầu mà còn sẵn sàng mở rộng hoặc tích hợp thêm các thành phần khác với hiệu năng và tính bảo mật vượt trội.

4.5.4. Ứng dụng

Hệ thống Web Server triển khai trong mạng nội bộ mang lại nhiều giá trị ứng dụng quan trọng trong thực tiễn, bao gồm:

1. Quản lý và chia sẻ tài nguyên nội bộ: Web Server đóng vai trò như một cổng thông tin tập trung, cung cấp các dịch vụ và tài nguyên nội bộ như tài liệu hướng dẫn, báo cáo, hoặc dữ liệu liên quan đến tổ chức. Nhờ đó, các thành viên trong mạng có thể truy cập dễ dàng và hiệu quả hơn.

2. Làm nền tảng phát triển và thử nghiệm: Đây là một môi trường lý tưởng cho việc phát triển và kiểm thử các ứng dụng web trước khi triển khai ra môi trường thực tế. Môi trường nội bộ này cho phép nhóm phát triển phát hiện và sửa lỗi một cách nhanh chóng, giảm thiểu rủi ro trong quá trình phát hành.

3. Đào tạo và học tập: Hệ thống giúp tạo môi trường học tập thực tế cho các cá nhân hoặc nhóm nghiên cứu về việc xây dựng và vận hành hạ tầng web. Các kịch bản cấu hình, bảo mật và tối ưu hóa đều có thể được thực hành ngay trên hệ thống này.

4. Nâng cao bảo mật dữ liệu: Việc triển khai trong mạng nội bộ hạn chế truy cập từ bên ngoài, đảm bảo an toàn cho dữ liệu nội bộ của tổ chức. Kết hợp với chứng chỉ SSL và tường lửa, hệ thống trở nên đáng tin cậy hơn trước các mối đe dọa.

=> Kết hợp với các thành phần khác trong mạng nội bộ như Data Server, hệ thống Web Server này không chỉ phục vụ nhu cầu hiện tại mà còn đáp ứng yêu cầu mở rộng, phát triển trong tương lai, trở thành một phần quan trọng trong hạ tầng CNTT của tổ chức.

CHƯƠNG 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1. Kết luận

Đề tài Tìm Hiểu Về Triển Khai Công Cụ Ảo Hóa KVM của môn học Quản trị mạng và Hệ thống đã giúp chúng em đã nắm được các khái niệm cơ bản về ảo hóa, sự khác biệt giữa ảo hóa Type 1 và Type 2, cũng có lại những kiến thức trong ngành CNTT như phần cứng, phần mềm.... Quá trình thực hiện đồ án môn học không chỉ giúp chúng em hiểu rõ về nguyên lý hoạt động của KVM mà còn mở rộng kiến thức về việc quản lý tài nguyên ảo hóa, bảo mật, và ứng dụng trong môi trường thực tiễn. Những kiến thức như quản lý CPU, RAM, lưu trữ và mạng trong hệ thống ảo hóa được ứng dụng triệt để, giúp tạo nên các kịch bản triển khai thực tiễn, từ xây dựng máy chủ dữ liệu đến thiết lập mạng nội bộ bảo mật.

Qua so sánh hiệu suất giữa KVM và VMWare, ta có thể thấy rằng trên cùng 1 phần cứng máy tính, KVM đã thể hiện ưu thế hơn trong việc tối ưu tài nguyên hệ thống và xử lý các tác vụ nặng. Đây là lợi thế rõ rệt giúp giảm độ trễ và tận dụng tối đa hiệu suất máy tính. Tuy nhiên, việc cài đặt dual-boot 1 hệ điều hành nhân Linux lên máy tính rồi sau đó cấu hình KVM thông qua các dòng lệnh Linux thì khá phức tạp, cần nhiều kỹ năng và kiến thức về IT hơn. Ngược lại, VMWare thì lại dễ sử dụng, linh hoạt trong môi trường đa nền tảng hơn (ta có thể cài VMWare lên cả macOS và Windows), phù hợp cho những ứng dụng yêu cầu giao diện trực quan, không đòi hỏi hiệu suất quá cao.

Đồ án của nhóm chúng em đã hoàn thành việc nghiên cứu và triển khai hệ thống ảo hóa KVM. Thông qua năm kịch bản thực tiễn, bao gồm tạo máy ảo GUI & CLI, thiết lập kết nối SSH, triển khai cơ sở dữ liệu MySQL và xây dựng Web Server kết hợp Data Server trong mạng nội bộ. Mỗi kịch bản không chỉ đạt được các mục tiêu kỹ thuật mà nhóm kỳ vọng mà còn cung cấp một nền tảng tốt để kiểm nghiệm hiệu suất và tính ứng dụng của KVM.

Ta thấy, KVM không chỉ đáp ứng các yêu cầu triển khai mà còn thể hiện tính ổn định, độ tin cậy và khả năng mở rộng. Khả năng kiểm soát tài nguyên, cùng các biện pháp bảo mật như tường lửa và mã hóa dữ liệu, đã đảm bảo an toàn trong môi trường mạng nội bộ. Hệ thống này là giải pháp lý tưởng để phục vụ mục đích thử nghiệm, phát triển, và triển khai hạ tầng ảo hóa cho doanh nghiệp lẫn tổ chức giáo dục.

Việc ứng dụng KVM trong thực tiễn đã khẳng định giá trị của nó trong các hoạt động công nghệ thông tin, từ quản lý tài nguyên nội bộ đến xây dựng các hệ thống phức tạp hơn, mang lại sự cân bằng giữa hiệu suất, bảo mật và khả năng sử dụng.

5.2. Hạn chế

Trong quá trình thực hiện đồ án, nhóm chúng em nhận thấy vẫn còn một số hạn chế:

- **Phân bổ thời gian chưa hợp lý:** Một số kịch bản được hoàn thiện muộn hơn so với kế hoạch được đề ra, chủ yếu liên quan đến vấn đề tối ưu hệ thống và tối ưu lại phương hướng thực hiện đồ án.

- **Phạm vi nghiên cứu giới hạn:** Chưa thể triển khai và so sánh toàn diện với các công nghệ ảo hóa khác do thời gian và nguồn lực hạn chế.

- **Kỹ năng thực hành:** Một số thao tác cấu hình ban đầu còn lúng túng, dẫn đến việc phải điều chỉnh nhiều lần.

Chúng em rút ra bài học quý báu từ những khuyết điểm này để cải thiện trong các dự án tương lai.

5.3. Hướng phát triển trong tương lai

Trong tương lai, nhóm chúng em sẽ cố gắng mở rộng việc tìm hiểu về các công nghệ ảo hóa tiên tiến khác, như Docker hoặc Kubernetes, để so sánh và tìm hiểu sự khác biệt với KVM. Ngoài ra, việc tích hợp các công nghệ bảo mật như IDS/IPS vào môi trường ảo hóa sẽ là một hướng quan trọng để đảm bảo an toàn hệ thống.

Nhóm cũng muốn tối ưu hóa hiệu năng của KVM trên nhiều loại phần cứng khác nhau, đồng thời triển khai các kịch bản thực tế như mô hình hybrid giữa KVM và nền tảng đám mây công cộng. Điều này giúp mở rộng ứng dụng của KVM trong các doanh nghiệp hoặc tổ chức lớn.

Thông qua các dự án tương lai, chúng em hy vọng tiếp tục trau dồi kỹ năng, đóng góp vào việc nâng cao chất lượng triển khai ảo hóa trong thực tiễn.

5.4. Lời kết

Chúng em xin gửi lời cảm ơn sâu sắc đến thầy Đỗ Hoàng Hiển, người đã đồng hành cùng chúng em trong suốt quá trình thực hiện đồ án. Thầy không chỉ truyền đạt kiến thức mà còn luôn sẵn lòng giải đáp mọi khó khăn, giúp chúng em hoàn thành công việc một cách tốt nhất.

Những bài học và sự tận tụy của thầy là nguồn động lực lớn lao cho chúng em trên con đường học tập và phát triển. Kính chúc thầy luôn mạnh khỏe, hạnh phúc và tiếp tục lan tỏa đam mê tri thức đến nhiều thế hệ sinh viên và luôn luôn đạt được những thành công trong sứ mệnh cao cả của mình.

----- HẾT -----

DANH MỤC TÀI LIỆU THAM KHẢO

1. Cisco. Cisco Catalyst 2960-S and 2960 Series Switches with LAN Lite Software
2. Cisco. ISR4321/K9 Datasheet
3. The Art of Service - Kernel Based Virtual Machine Publishing. Kernel Based Virtual Machine A Complete Guide - 2021 Edition
4. Khaleel Ahmad, Ahamed Shareef. Hands-On Kernel-Based Virtual Machine (KVM)
5. Cong Li (2017). Kernel-based Virtual Machine