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

NGUYỄN THANH TÙNG – 23521744
TẠ QUỐC TUẤN - 23521728

BÁO CÁO ĐỒ ÁN CUỐI KỲ

GUACAMOLE

GIẢNG VIÊN HƯỚNG DẪN
VĂN THIÊN LUÂN

TP. HỒ CHÍ MINH, 2025

Mục lục

Hình ảnh

Bảng

I. CƠ SỞ LÝ THUYẾT

1.1. GIỚI THIỆU VỀ APACHE GUACAMOLE

Apache Guacamole là một **cổng truy cập máy tính từ xa mã nguồn mở và miễn phí**. Nó cho phép người dùng truy cập và điều khiển máy tính để bàn từ xa thông qua trình duyệt web mà không cần cài đặt bất kỳ phần mềm hoặc plugin bổ sung nào.

Dưới đây là một số đặc điểm chính của Apache Guacamole:

- **Truy cập không cần cài đặt (Clientless):** Người dùng có thể kết nối đến máy tính từ xa chỉ bằng trình duyệt web HTML5, loại bỏ nhu cầu cài đặt phần mềm client trên thiết bị của họ.
- **Hỗ trợ đa giao thức:** Guacamole hỗ trợ các giao thức phổ biến như **RDP** (Remote Desktop Protocol) cho Windows, **VNC** (Virtual Network Computing) cho Linux và các hệ thống khác, và **SSH** (Secure Shell) cho truy cập dòng lệnh.
- **Truy cập từ mọi nơi:** Vì là ứng dụng web, người dùng có thể truy cập máy tính của mình từ bất kỳ thiết bị nào có trình duyệt web và kết nối internet.
- **Quản lý tập trung:** Guacamole cho phép quản trị viên quản lý tập trung các kết nối và người dùng, cũng như cấu hình quyền truy cập.
- **Bảo mật:** Guacamole hỗ trợ các tính năng bảo mật như mã hóa phiên làm việc và tích hợp với các hệ thống xác thực khác nhau (ví dụ: LDAP, cơ sở dữ liệu SQL, xác thực đa yếu tố).
- **Khả năng mở rộng:** Kiến trúc của Guacamole cho phép mở rộng thông qua việc cân bằng tải giữa nhiều máy chủ Guacamole.
- **Tích hợp dễ dàng:** Guacamole cung cấp API cho phép tích hợp nó vào các ứng dụng và dịch vụ khác.

Kiến trúc cơ bản của Apache Guacamole bao gồm:

- **Ứng dụng web Guacamole:** Đây là giao diện người dùng mà người dùng tương tác thông qua trình duyệt web. Nó được viết bằng Java và JavaScript.
- **Máy chủ Guacamole (guacd):** Đây là một proxy daemon chạy ở backend. Nó nhận các yêu cầu kết nối từ ứng dụng web và thiết lập kết nối đến các máy chủ từ xa bằng các giao thức như RDP, VNC hoặc SSH. guacd chuyển đổi dữ liệu giữa giao thức gốc của máy chủ từ xa và giao thức Guacamole (một giao thức dựa trên văn bản được tối ưu hóa cho web).

1.2. MYSQL AUTHENTICATION BACKEND

MySQL Authentication Backend (hay còn gọi là MySQL Authentication Provider) của Apache Guacamole là một **phần mở rộng (extension) cho phép Guacamole sử dụng cơ sở dữ liệu MySQL (hoặc MariaDB) để quản lý thông tin xác thực người dùng, quyền truy cập và cấu hình kết nối.**

Thay vì sử dụng phương pháp xác thực mặc định dựa trên tệp **user-mapping.xml**, MySQL Authentication Backend lưu trữ tất cả thông tin này trong một cơ sở dữ liệu MySQL. Điều này mang lại nhiều lợi ích, đặc biệt là trong các môi trường triển khai lớn và phức tạp:

Các chức năng chính của MySQL Authentication Backend:

- **Quản lý người dùng:** Cho phép người dùng tạo, sửa đổi và xóa tài khoản người dùng trực tiếp trong cơ sở dữ liệu MySQL thông qua giao diện quản trị web của Guacamole (sau khi đã cài đặt và cấu hình backend này).
- **Xác thực người dùng:** Khi người dùng cố gắng đăng nhập vào Guacamole, backend này sẽ truy vấn cơ sở dữ liệu MySQL để xác minh tên người dùng và mật khẩu.

- **Quản lý quyền truy cập:** Người dùng có thể gán quyền truy cập cụ thể cho từng người dùng hoặc nhóm người dùng đối với các kết nối từ xa (ví dụ: RDP, VNC, SSH) đã được cấu hình trong Guacamole.
- **Quản lý kết nối:** Thông tin về các kết nối từ xa (hostname, port, giao thức, v.v.) cũng được lưu trữ trong cơ sở dữ liệu MySQL, cho phép quản lý tập trung.
- **Nhật ký hoạt động:** Backend này thường ghi lại các hoạt động đăng nhập, kết nối và ngắt kết nối vào cơ sở dữ liệu, giúp bạn theo dõi và kiểm tra.
- **Khả năng mở rộng:** Sử dụng cơ sở dữ liệu giúp Guacamole dễ dàng mở rộng quy mô khi số lượng người dùng và kết nối tăng lên.

Lợi ích khi sử dụng MySQL Authentication Backend:

- **Quản lý tập trung:** Tất cả thông tin xác thực và cấu hình được quản lý tại một nơi duy nhất, giúp việc quản trị trở nên dễ dàng và hiệu quả hơn.
- **Linh hoạt và mạnh mẽ:** Cơ sở dữ liệu MySQL cung cấp nhiều tính năng mạnh mẽ cho việc quản lý dữ liệu và quyền truy cập phức tạp.
- **Khả năng tích hợp:** Dễ dàng tích hợp với các hệ thống quản lý người dùng khác hoặc các ứng dụng khác có thể tương tác với cơ sở dữ liệu MySQL.
- **Khả năng mở rộng tốt:** Phù hợp với các môi trường có số lượng lớn người dùng và kết nối.
- **Giao diện quản trị:** Cung cấp giao diện web để quản lý người dùng, nhóm và kết nối một cách trực quan.

1.3. MYSQL REPLICATION

MySQL Replication là một **công nghệ tích hợp sẵn trong MySQL cho phép bạn sao chép dữ liệu từ một máy chủ MySQL (gọi là master hoặc primary) sang một hoặc nhiều máy chủ MySQL khác (gọi là slaves hoặc replicas) theo thời gian thực hoặc gần thời gian thực.**

Đây là một tính năng quan trọng và được sử dụng rộng rãi trong nhiều kiến trúc hệ thống để đạt được các mục tiêu sau:

Các mục tiêu chính của MySQL Replication:

- **Khả năng chịu lỗi (Fault Tolerance):** Nếu máy chủ master gặp sự cố, một trong các máy chủ slave có thể được nâng cấp lên thành master để tiếp tục phục vụ ứng dụng, giảm thiểu thời gian chết.
- **Cân bằng tải (Read Scalability):** Các thao tác đọc (SELECT) có thể được phân tán sang các máy chủ slave, giảm tải cho máy chủ master chuyên xử lý các thao tác ghi (INSERT, UPDATE, DELETE).
- **Sao lưu (Backup):** Máy chủ slave có thể được sử dụng để thực hiện sao lưu dữ liệu mà không ảnh hưởng đến hiệu suất của máy chủ master đang hoạt động.
- **Phân tích dữ liệu (Analytics):** Các truy vấn phân tích phức tạp có thể được chạy trên máy chủ slave để tránh ảnh hưởng đến hiệu suất của hệ thống giao dịch chính trên máy chủ master.
- **Phục hồi dữ liệu (Disaster Recovery):** Các máy chủ slave có thể được đặt ở các vị trí địa lý khác nhau để bảo vệ dữ liệu khỏi các sự kiện thảm họa cục bộ.
- **Kiểm thử và phát triển (Testing and Development):** Một bản sao dữ liệu trên máy chủ slave có thể được sử dụng cho mục đích kiểm thử và phát triển mà không gây rủi ro cho dữ liệu production.

1.4. SELF-SIGNED CERTIFICATE

Self-signed certificate (chứng chỉ tự ký) là một chứng chỉ số được tạo ra và ký bởi chính chủ sở hữu của nó, thay vì được một cơ quan cấp chứng chỉ (Certificate Authority - CA) đáng tin cậy xác thực.

1.4.1. Cách thức hoạt động:

- **Tạo và ký:** Khi bạn tạo self-signed certificate, bạn sử dụng khóa riêng của mình để ký chứng chỉ. Điều này có nghĩa là chứng chỉ không có "chuỗi tin cậy" từ một CA bên ngoài.
- **Mã hóa:** Mặc dù chứng chỉ này vẫn giúp mã hóa dữ liệu giữa client và server, nó không được trình duyệt hoặc hệ thống tin cậy tự động, vì không có CA đáng tin cậy xác nhận.

1.4.2. Ưu điểm

- **Miễn phí và nhanh chóng**: Bạn có thể tạo chứng chỉ này mà không phải trả phí hoặc chờ đợi xác thực từ CA.
- **Phù hợp cho môi trường phát triển hoặc nội bộ:** Trong các môi trường test, nội bộ hoặc khi triển khai dịch vụ nội bộ, self-signed certificate là đủ để đảm bảo mã hóa dữ liệu.

1.4.3. Nhược điểm

- **Cảnh báo từ trình duyệt:** Trình duyệt sẽ hiển thị cảnh báo "Not Secure" hoặc "Your connection is not private" vì chứng chỉ không được cấp bởi CA đáng tin cậy.
- **Thiếu tin cậy trong môi trường công cộng:** Đối với các ứng dụng hoặc trang web được truy cập từ bên ngoài, self-signed certificate có thể khiến người dùng lo ngại về tính bảo mật.

1.4.4. Khi nào nên dùng Self-signed Certificate?

- **Môi trường phát triển và thử nghiệm:** Để nhanh chóng thiết lập HTTPS cho máy chủ trong giai đoạn thử nghiệm.
- **Mạng nội bộ hoặc hệ thống nội bộ:** Khi bạn kiểm soát các máy khách và có thể cấu hình tin cậy chứng chỉ này bằng cách thêm vào Trusted Certificate Store.

1.5. HA – PROXY

1.5.1. Chức năng chính

- **Reverse Proxy:** HAProxy nhận các yêu cầu từ client và chuyển tiếp đến một hoặc nhiều máy chủ backend. Điều này giúp ẩn địa chỉ IP thực của máy chủ backend và tăng tính bảo mật.
- **Load Balancing:** Nó phân phối các yêu cầu đến các máy chủ backend dựa trên các thuật toán khác nhau (như round-robin, least connections, v.v.) để cân bằng tải và tăng hiệu năng cho hệ thống.
- **High Availability:** HAProxy có khả năng giám sát tình trạng của các máy chủ backend và tự động loại bỏ các máy chủ gặp sự cố ra khỏi danh sách xử lý, giúp hệ thống luôn sẵn sàng phục vụ.

1.5.2. Lợi ích khi sử dụng HA – Proxy

- **Tăng tính sẵn sàng (Availability):** Bằng cách chuyển hướng các yêu cầu đến các máy chủ backend khỏe mạnh, HAProxy giúp đảm bảo dịch vụ luôn hoạt động.
- **Cải thiện hiệu năng:** Phân phối tải đều giữa các máy chủ backend giúp giảm tình trạng quá tải và cải thiện tốc độ phản hồi.
- **Bảo mật:** HAProxy có thể kết thúc kết nối SSL/TLS (TLS termination) và sau đó chuyển tiếp yêu cầu đến backend qua HTTP hoặc HTTPS, giúp giảm tải công việc mã hóa trên máy chủ backend.

- **Cấu hình linh hoạt:** HAProxy cho phép cấu hình đa dạng với các tính năng nâng cao như giới hạn tốc độ, routing dựa trên nội dung, cân bằng tải theo trọng số, v.v.

1.5.3. Ứng dụng thực tế

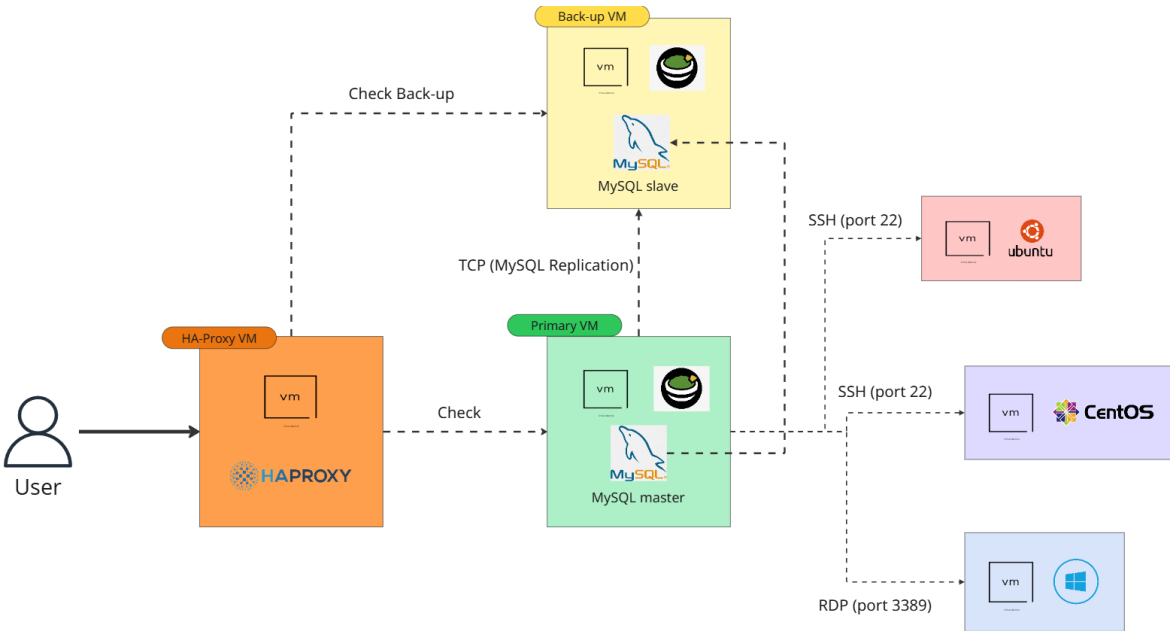
- **Websites có lượng truy cập lớn:** HAProxy giúp phân phối lưu lượng truy cập giữa nhiều máy chủ để tránh tình trạng quá tải.
- **Hệ thống microservices:** Nó có thể đóng vai trò là điểm giao tiếp trung tâm giữa các dịch vụ khác nhau, đảm bảo các yêu cầu được định tuyến đến đúng dịch vụ.
- **Môi trường có yêu cầu bảo mật cao:** Với tính năng TLS termination, HAProxy giúp giảm tải cho các máy chủ backend và cung cấp các biện pháp bảo mật bổ sung.

II. MÔ HÌNH TRIỂN KHAI

2.1. MÔ HÌNH TỔNG QUAN

- 1 Guacamole Server chính
- 1 Guacamole backup
- 1 MySQL database (hoặc replication giữa 2 MySQL)
- 3 máy đích để kết nối: Ubuntu VM, CentOS VM, Windows VM
- 1 Máy ảo làm Reversed Proxy

2.2. SƠ ĐỒ MÔ HÌNH



Hình 1. Sơ đồ mô hình

III. CÀI ĐẶT, CẤU HÌNH

3.1. CÀI ĐẶT GUACAMOLE SERVER

- Cài guacd (Guacamole daemon): Guacd sẽ chịu trách nhiệm giao tiếp với guacamole-client thông qua giao thức Guacamole nội bộ.

- Cài Tomcat + deploy guacamole.war:

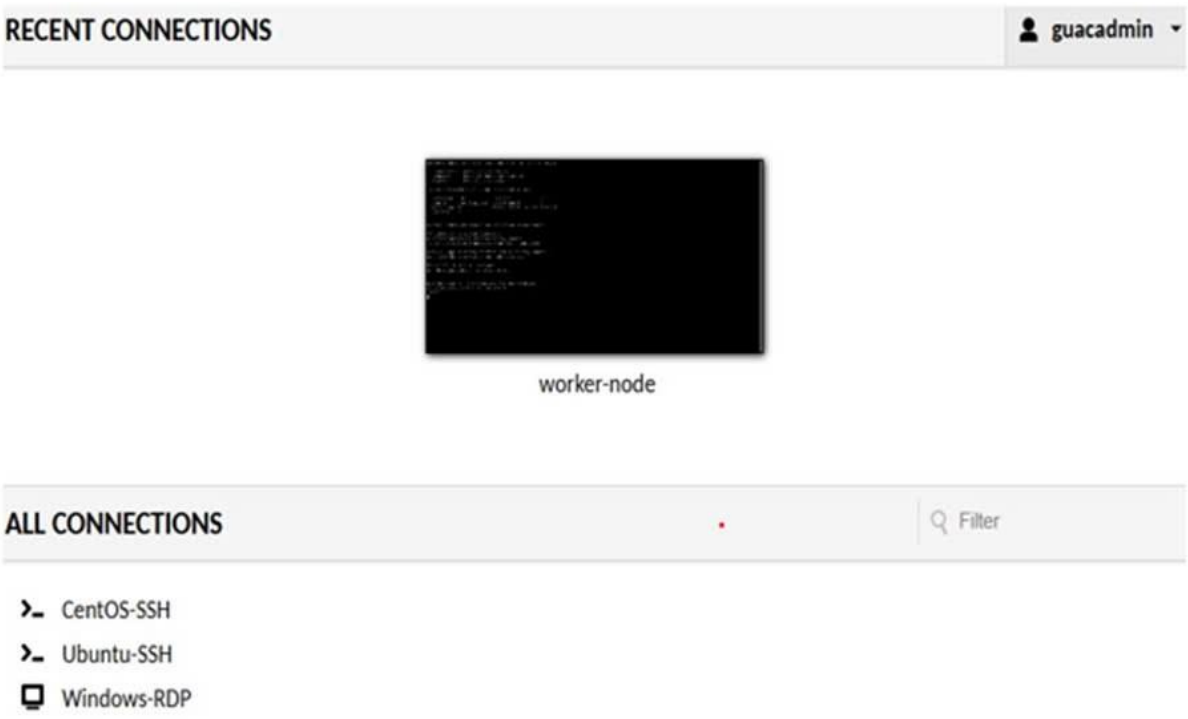
- Guacamole client được triển khai dưới dạng một ứng dụng web (guacamole.war) nên cần một servlet container như Tomcat để chạy. Đây cũng là nơi xử lý giao diện web và quản lý phiên làm việc của người dùng.

- File .war chứa toàn bộ phần giao diện người dùng (web client) của Guacamole. Sau khi triển khai vào Tomcat, người dùng có thể truy cập Guacamole qua trình duyệt.

- Cài thư viện giao thức (libssh, freerdp, v.v)

- Các thư viện này cho phép guacd thực hiện kết nối từ xa đến các máy chủ đích.

Hình 2. Giao diện Guacamole



Hình 3. Giao diện khi đăng nhập bằng tài khoản toàn quyền

3.2. CÀI ĐẶT VÀ CẤU HÌNH MYSQL

- Tạo database guacamole_db

- guacamole_db là cơ sở dữ liệu để lưu trữ toàn bộ thông tin liên quan đến người dùng, các kết nối, quyền truy cập,... giúp cho người dùng có thể dễ dàng quản lý, sao lưu và phục hồi.
- Guacamole server sẽ truy vấn vào database này khi người dùng đăng nhập hoặc mở kết nối từ xa.

- Import schema SQL (schema/*.sql)

- Các file schema chứa các câu lệnh SQL tạo cấu trúc bảng, ràng buộc, chỉ mục,…
- Guacamole cần phải có các bảng từ file schema để có thể hoạt động đúng ví dụ như: guacamole_user, guacamole_connection, guacamole_connection_permission, ...

- Tạo 5 tài khoản người dùng

- Tạo các tài khoản để có thể kiểm tra tính năng phân quyền và truy cập đa người dùng.

```
mysql> select * from guacamole_entity;
+-------------+------------+-------+
| entity_id | name       | type |
+-----------+------------+------+
|           2 | admin      | USER |
|           1 | guacadmin  | USER |
|           3 | user1      | USER |
|           4 | user2      | USER |
|           5 | user3      | USER |
|           6 | user4      | USER |
+-------------+------------+-------+
6 rows in set (0.00 sec)
```

Hình 4. Các user hiện tại có trong cơ sở dữ liệu

- Tạo các kết nối (3 máy đích) và phân quyền cho từng user

```
MariaDB [guacamole_db]> SELECT connection_id, connection_name, protocol
    -> FROM guacamole_connection;
+---------------+-----------------+----------+
| connection_id | connection_name  | protocol |
+---------------+-----------------+----------+
|             1 | Ubuntu VM (SSH)  | ssh      |
|             2 | CentOS VM (SSH)  | ssh      |
|             3 | Windows VM (RDP) | rdp      |
+---------------+-----------------+----------+
```

Hình 5. Các kết nối được tạo

- Việc phân quyền cho từng user được thực hiện thông qua bảng guacamole_connection_permission trong cơ sở dữ liệu.
- Mỗi người dùng sẽ được cấp quyền cụ thể (READ, UPDATE, DELETE, ADMIN) đối với từng kết nối. Trong đó, quyền READ là

bắt buộc để người dùng có thể nhìn thấy và sử dụng kết nối từ xa thông qua giao diện web.

- Cơ chế phân quyền này giúp kiểm soát chặt chẽ ai được phép truy cập và thao tác với từng kết nối, đảm bảo tính linh hoạt và bảo mật trong hệ thống multi-user.

```
mysql> select * from guacamole_connection_permission;
+-----------+---------------+------------+
| entity_id | connection_id | permission |
+-----------+---------------+------------+
|         2 |             1 | READ       |
|         3 |             1 | READ       |
|         4 |             1 | READ       |
|         6 |             1 | READ       |
|         2 |             2 | READ       |
|         4 |             2 | READ       |
|         5 |             2 | READ       |
|         2 |             3 | READ       |
|         3 |             3 | READ       |
|         5 |             3 | READ       |
|         6 |             3 | READ       |
|         1 |             5 | READ       |
|         1 |             5 | UPDATE     |
|         1 |             5 | DELETE     |
|         1 |             5 | ADMINISTER |
+-----------+---------------+------------+
15 rows in set (0.00 sec)
```

Hình 6. Phân quyền cho từng user

3.3. CẤU HÌNH KẾT NỐI

- Để có thể cấu hình kết nối một cách dễ dàng, ta sẽ sử dụng Vagrant - một công cụ mã nguồn mở giúp dễ dàng tạo và quản lý các môi trường máy ảo (VM) để phục vụ việc phát triển phần mềm.

- Vagrant sẽ tự động hóa quá trình khởi tạo và cấu hình các máy ảo, đồng thời hỗ trợ thiết lập các giao thức kết nối từ xa như **SSH** (cho Linux) và **RDP** (cho Windows) một cách dễ dàng.

- Cụ thể, Vagrant được dùng để triển khai 3 máy ảo gồm:

- **Ubuntu VM** – sử dụng kết nối **SSH**
- **CentOS VM** – sử dụng kết nối **SSH**
- **Windows VM** – sử dụng kết nối **RDP**

- Việc cấu hình được thực hiện thông qua file Vagrantfile – là file cấu hình chính của Vagrant, trong đó:

- **Với máy Linux (Ubuntu/CentOS)**: Vagrant tự động tạo cặp khóa SSH và mở port 22 để kết nối từ xa.
- **Với máy Windows**: Vagrant bật giao thức RDP, gán port 3389 và tự động thiết lập user/password để truy cập.

- Sau khi khởi tạo máy ảo, các thông tin như IP, port, username và password được sử dụng để khai báo vào hệ thống quản lý từ xa của Apache Guacamole. Nhờ đó, người dùng có thể truy cập trực tiếp vào từng máy ảo qua trình duyệt mà không cần cài thêm phần mềm kết nối bên ngoài.

Trong đó:

- connection_id = 1: Có thể SSH vào các máy ảo Ubuntu
- connection_id = 2: Có thể SSH vào các máy ảo Centos
- connection_id = 3: Có thể RDP vào các máy ảo Windows

3.4. CẤU HÌNH GUACAMOLE BACKUP

- Đây là một máy ảo dùng để host Guacamole Server, với mục tiêu là back-up dữ liệu từ máy ảo thứ nhất trong trường hợp Guacamole Server đầu tiên gặp sự cố thì đây sẽ là máy thay thế.

- Máy ảo thứ hai có cấu hình giống như máy thứ nhất.

3.5. CẤU HÌNH MYSQL REPLICATION

- Thiết lập Master – Slave:

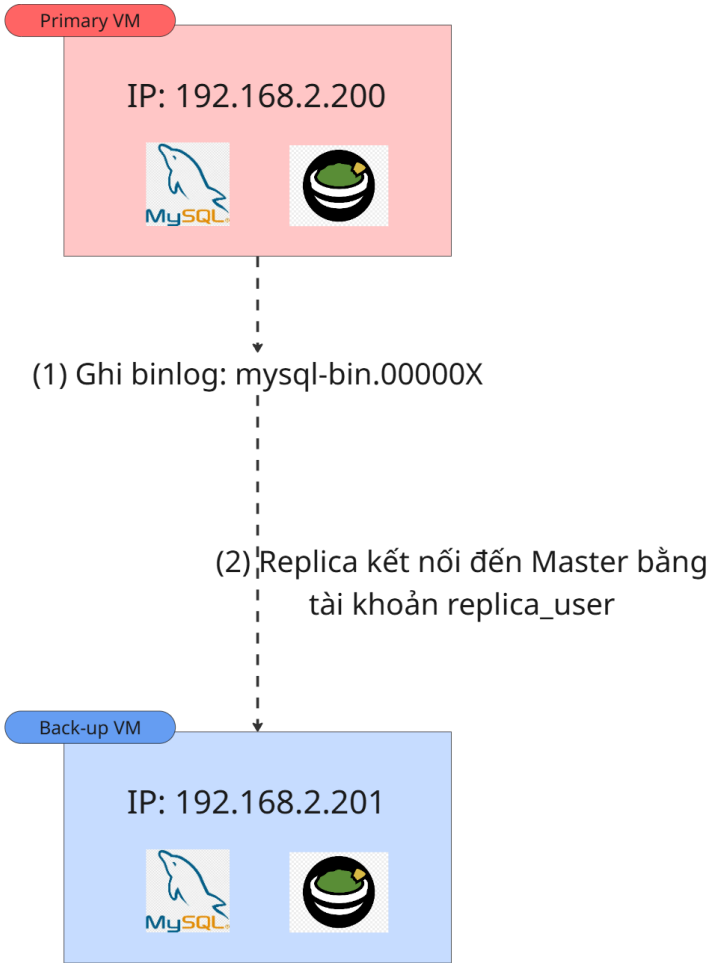
- Master sau mỗi lần có thầy đổi thì sẽ ghi vào phải **mysql-bin.00000X** thay đổi đó.
- Slave sẽ đăng nhập vào cơ sở dữ liệu của Master bằng tài khoản **replica_user** (hoặc người dùng tự đặt).
- Slave sẽ cập nhật dữ liệu theo file **mysql-bin.00000X.**

- Tự động đồng bộ hóa user/kết nối giữa 2 máy:

- Khi cấu hình, cần cung cấp cho Slave **bin file** hiện tại và vị trí của nó. Sau đó Slave sẽ đăng nhập vào cơ sở dữ liệu của Master và cập nhật đúng thông tin theo file **bin** hiện tại.

```
mysql> SHOW MASTER STATUS;  # Note File & Position
+------------------+----------+--------------+------------------+-------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB | Executed_Gtid_Set |
+------------------+----------+--------------+------------------+-------------------+
| mysql-bin.000002 |      933 | guacamole_db |                  |                   |
+------------------+----------+--------------+------------------+-------------------+
1 row in set (0.00 sec)
```

Hình 7. Ví dụ về file và vị trí hiện tại



Hình 8. Sơ đồ thực hiện replication

3.6. BẢO MẬT GUACAMOLE WEB VỚI SELF-SIGNED CERTIFICATE

3.6.1. Tạo Self-Signed Certificate và Private Key

- **Sử dụng OpenSSL**:

  Mở terminal và chạy lệnh sau để tạo chứng chỉ (certificate) và khóa riêng (private key). Ví dụ, tạo chứng chỉ có hiệu lực 365 ngày:

```
openssl req -new -x509 -days 365 -nodes -out guacamole.crt -keyout guacamole.key
```

Bảng 1. Lệnh tạo chứng chỉ và khóa riêng

Trong quá trình này, bạn sẽ được yêu cầu nhập các thông tin như quốc gia, tỉnh/thành phố, tên tổ chức, … (các thông tin này sẽ được nhúng vào certificate).

3.6.2. Tạo Keystore Dạng PKCS12

- **Chuyển đổi sang định dạng PKCS12**:

  Nếu Guacamole được triển khai trên Tomcat (thường là Java-based), bạn cần chuyển đổi certificate và key sang file keystore dạng PKCS12. Sử dụng lệnh:

```
openssl pkcs12 -export -in guacamole.crt -inkey guacamole.key -out guacamole.p12
-name tomcat
```

Bảng 2. Chuyển đổi certificate và key sang file keystore dạng PKCS12

3.6.3. Cấu Hình Tomcat Sử Dụng Self-Signed Certificate

- **Chỉnh sửa file cấu hình**:

  Mở file **server.xml** nằm trong thư mục **conf** của Tomcat.

- **Tìm Connector HTTPS**:

  Tìm phần cấu hình **<Connector>** dùng cho HTTPS (thường là cổng 8443). Nếu chưa có, bạn có thể thêm mới Connector.

- **Cập nhật thông số cho Connector**:

  Ví dụ về cấu hình Connector:

```
<Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
        maxThreads="150" SSLEnabled="true" scheme="https" secure="true"
        keystoreFile="/đường/dẫn/đến/guacamole.p12"
        keystoreType="PKCS12"
        keystorePass="mật_khẩu_của_bạn"
        clientAuth="false" sslProtocol="TLS"/>
```

Bảng 3. Cấu hình connector

Lưu ý:

- **keystoreFile:** Thay bằng đường dẫn đầy đủ tới file **guacamole.p12** vừa tạo.
- **keystorePass:** Nhập mật khẩu đã thiết lập khi xuất file PKCS12.
- **keystoreType:** Với file PKCS12, giá trị này là **PKCS12**.

3.6.4. Khởi Động Lại Tomcat

- Áp dụng thay đổi:

  Sau khi lưu file cấu hình, khởi động lại Tomcat để các thay đổi có hiệu lực.

```
sudo systemctl restart tomcat
```

Bảng 4. Khởi động lại tomcat

3.6.5. Kiểm Tra và Xác Nhận

- **Truy cập Guacamole qua HTTPS**:

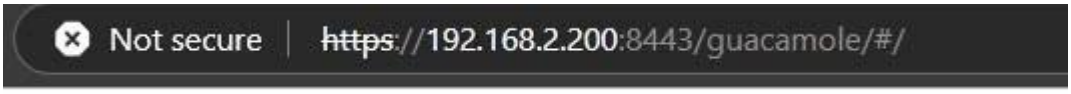
  Mở trình duyệt và nhập địa chỉ:

```
https://<địa_chỉ_máy_chủ>:8443/guacamole
```

Bảng 5. Truy cập vào Guacamole web

- **Chấp nhận cảnh báo**:

  Vì sử dụng self-signed certificate, trình duyệt sẽ cảnh báo về tính không tin cậy của chứng chỉ. Bạn cần xác nhận để tiếp tục (trong môi trường nội bộ hoặc dùng cho mục đích thử nghiệm).



Hình 9. Kết quả sau khi truy cập Guacamole web

3.7. CẤU HÌNH REVERSED PROXY CHO GIẢ ĐỊNH TRƯỞNG HỢP MẤT KẾT NỐI VỚI MÁY ẢO THỨ NHẤT, FAILOVER ĐẾN MÁY ẢO THỨ HAI

3.7.1. Mục tiêu

- Nhằm đảm bảo tính sẵn sàng cao (High Availability) cho hệ thống Guacamole, tiến hành cài đặt một máy ảo thứ ba chạy **HAProxy** để thực hiện chức năng **cân bằng tải và chuyển đổi tự động (auto-failover)** giữa hai máy chủ Guacamole: **Primary** và **Backup**.

3.7.2. Mô hình triển khai

- **VM1 (Primary Guacamole Server):** Chạy Guacamole Server với MySQL làm backend xác thực.
- **VM2 (Backup Guacamole Server):** Đồng bộ dữ liệu MySQL từ máy Primary bằng cơ chế **MySQL replication (read-only)**.
- **VM3 (HAProxy):** Đóng vai trò cổng truy cập duy nhất từ người dùng, chuyển tiếp kết nối tới Guacamole Primary, và nếu Primary không khả dụng, sẽ tự động chuyển sang máy Backup.

3.7.3. Cấu hình HAProxy

HAProxy được cấu hình ở chế độ HTTP hoặc TCP tùy thuộc vào cổng Guacamole sử dụng (mặc định là 8080). Ví dụ cấu hình đơn giản như sau:

```
frontend guacamole_frontend
    bind *:80
    default_backend guacamole_backend

backend guacamole_backend
    option httpchk GET /guacamole/
    http-check expect status 200
    server primary 192.168.56.101:8080 check
    server backup 192.168.56.102:8080 check backup
```

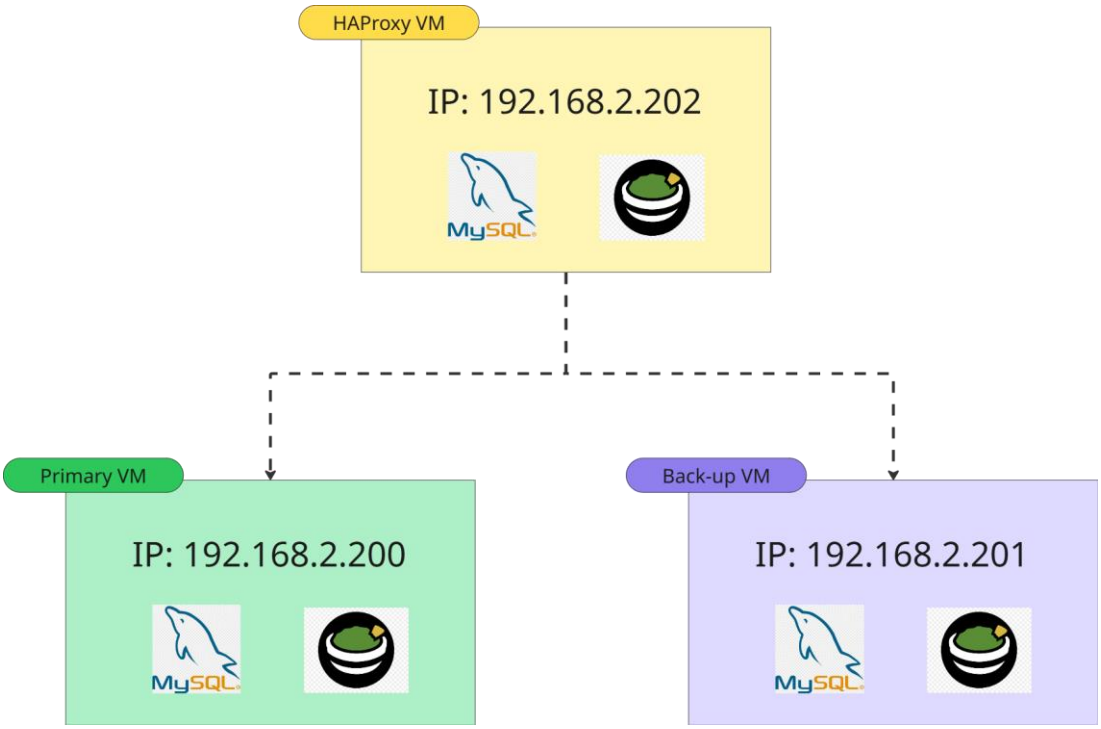
Bảng 6. Cấu hình HAProxy

Giải thích:

- **option httpchk**: kiểm tra đường dẫn /guacamole/ để xác định server còn hoạt động.
- **server primary**: là địa chỉ IP máy Guacamole chính.
- **server backup**: sẽ được sử dụng khi primary không phản hồi.

3.7.4. Kiểm thử và đánh giá

- Khi cả hai máy đều hoạt động: mọi yêu cầu được chuyển đến máy Primary.
- Khi máy Primary ngắt kết nối (dừng dịch vụ hoặc tắt máy): HAProxy tự động chuyển hướng đến máy Backup.
- Khi máy Primary hoạt động trở lại: HAProxy tiếp tục ưu tiên sử dụng Primary.



Hình 10. Sơ đồ kết nối giữa 3 máy

IV. KẾT QUẢ VÀ KẾT LUẬN.

Sau quá trình triển khai và cấu hình hệ thống Apache Guacamole trên môi trường ảo hóa, nhóm đã đạt được các kết quả chính sau:

- Cài đặt thành công Guacamole Server trên máy ảo master, sử dụng MySQL làm hệ thống xác thực người dùng.
- Tạo bản sao Guacamole trên máy backup, đồng bộ hóa dữ liệu người dùng và cấu hình kết nối thông qua MySQL Replication.
- Triển khai HAProxy trên máy thứ ba để đảm bảo tính sẵn sàng cao (high availability). Khi máy chủ chính không khả dụng, HAProxy sẽ tự động chuyển hướng truy cập sang máy chủ dự phòng.

Kết luận:

Hệ thống điều khiển máy từ xa thông qua Apache Guacamole đã được triển khai thành công với các tính năng bảo mật, dự phòng và khả năng mở rộng cao. Việc sử dụng MySQL để xác thực và đồng bộ hóa người dùng giữa các máy chủ giúp đảm bảo tính nhất quán và linh hoạt trong quản lý. Nhóm đánh giá rằng mô hình này hoàn toàn có thể áp dụng thực tế trong môi trường doanh nghiệp nhỏ hoặc trường học để hỗ trợ điều khiển máy từ xa an toàn và hiệu quả.

V. BẢNG CÔNG VIỆC

STT	MSSV	Tên	Nội dung công việc phụ trách	Tự đánh giá (theo thang điểm 10)
1	23521744	Nguyễn Thanh Tùng	- Cấu hình Guacamole cơ bản - Cấu hình MySQL với 5 tài khoản xác thực - Cấu hình SSH tới Ubuntu/Centos, RDP tới Windows, và phân quyền kết nối cho các user - Triển khai máy ảo Guacamole back-up và đồng bộ hóa dữ liệu với máy chính - Bảo mật Guacamole web với self-signed certificates - Triển khai Reversed Proxy, mô phỏng trường hợp failover tới máy back-up - Làm báo cáo	9/10
2	23521728	Tạ Quốc Tuấn	- Thiết kế sơ đồ hệ thống - Cấu hình Guacamole cơ bản - Cấu hình MySQL với 5 tài khoản xác thực	9/10

			- Cấu hình SSH tới Ubuntu/Centos, RDP tới Windows, và phân quyền kết nối cho các user - Làm báo cáo - Làm slide thuyết trình - Soạn câu hỏi Quiz	

Bảng 7. Bảng phân công công việc

VIDEO DEMO:

https://drive.google.com/file/d/1u58xzP_cpn4LLulgSzlikwqiG1Jcg7fN/view?usp=sharing