**Tìm hiểu sâu về ML2/OVS Openstack**

1. Lý thuyết chung
2. Openstack Networking

Mạng Openstack cho phép người dùng tạo và quản lý các đối tượng mạng, chẳng hạn như mạng, mạng con và cổng mạng mà các dịch vụ OpenStack khác có thể sử dụng. Các plug-in có thể được triển khai để phù hợp với các thiết bị và phần mềm mạng khác nhau, mang lại sự linh hoạt cho kiến ​​trúc và triển khai Openstack.

Dịch vụ mạng, có tên mã là Neutron, cung cấp một API cho phép người dùng xác định kết nối và địa chỉ mạng trong đám mây. Dịch vụ Mạng cho phép người điều hành tận dụng các công nghệ mạng khác nhau để tăng sức mạnh cho mạng đám mây của họ. Dịch vụ Mạng cũng cung cấp API để định cấu hình và quản lý nhiều dịch vụ mạng khác nhau, từ chuyển tiếp L3 và dịch địa chỉ mạng (NAT) đến cân bằng tải, tường lửa vành đai và mạng riêng ảo.

Nó bao gồm các thành phần sau:

**Máy chủ API**

Openstack Networking API bao gồm hỗ trợ cho mạng Lớp 2 và Quản lý Địa chỉ IP (IPAM), cũng như phần mở rộng cho cấu trúc bộ định tuyến Lớp 3 cho phép định tuyến giữa các mạng Lớp 2 và cổng vào các mạng bên ngoài. Openstack Networking bao gồm một danh sách ngày càng tăng các plug-in cho phép khả năng tương tác với các công nghệ mạng nguồn mở và thương mại khác nhau, bao gồm bộ định tuyến, bộ chuyển mạch, bộ chuyển mạch ảo và bộ điều khiển mạng được xác định bằng phần mềm (SDN).

**Openstack Networking plug-in và agents**

Cắm và rút cổng mạng, tạo mạng hoặc mạng con và cung cấp địa chỉ IP. Plug-in và agents được chọn khác nhau tùy thuộc vào nhà cung cấp và công nghệ được sử dụng trong cloud. Điều quan trọng cần lưu ý là mỗi lần cài đặt chỉ có thể sử dụng một plug-in.

**Hàng đợi tin nhắn**

Chấp nhận và định tuyến các yêu cầu RPC giữa các agent để thực hiện các hoạt động API. Hàng đợi tin nhắn được sử dụng trong ML2 plug-in cho RPC giữa máy chủ neutron và các neutron agent chạy trên mỗi hypervisor, trong trình điều khiển cơ chế ML2 cho Open vSwitch và Linux bridge.

1. Về ý tưởng

Để cấu hình cấu trúc liên kết mạng, người dùng có thể tạo và cấu hình mạng và mạng con, đồng thời hướng dẫn các dịch vụ Openstack khác như Compute để gắn thiết bị ảo vào các cổng trên các mạng này. Openstack Compute là ứng dụng tiêu dùng nổi bật của Openstack Networking để cung cấp khả năng kết nối cho các phiên bản của nó. Đặc biệt, Openstack Networking hỗ trợ mỗi dự án có nhiều mạng riêng và cho phép các dự án (project) chọn sơ đồ địa chỉ IP của riêng họ, ngay cả khi các địa chỉ IP đó trùng với địa chỉ mà các dự án khác sử dụng. Có hai loại mạng lưới, dự án (project) và mạng lưới nhà cung cấp (provider network). Có thể chia sẻ bất kỳ loại mạng nào trong số các dự án như một phần của quá trình tạo mạng.

1. Provider network (Mạng nhà cung cấp)

Mạng nhà cung cấp cung cấp khả năng kết nối lớp 2 cho các phiên bản với hỗ trợ tùy chọn cho dịch vụ DHCP và metadata. Các mạng này kết nối hoặc ánh xạ tới các mạng lớp 2 hiện có trong trung tâm dữ liệu, thường sử dụng gắn thẻ VLAN (802.1q) để xác định và phân tách chúng.

Mạng nhà cung cấp thường cung cấp sự đơn giản, hiệu suất và độ tin cậy với chi phí linh hoạt. Theo mặc định, chỉ quản trị viên mới có thể tạo hoặc cập nhật mạng của nhà cung cấp vì chúng yêu cầu cấu hình cơ sở hạ tầng mạng vật lý. Có thể thay đổi để người dùng được phép tạo hoặc cập nhật mạng nhà cung cấp với các tham số sau policy.json:

* create\_network:provider:physical\_network
* update\_network:provider:physical\_network
* *Cảnh báo*

*Việc tạo và sửa đổi mạng nhà cung cấp cho phép sử dụng tài nguyên mạng vật lý, chẳng hạn như VLAN-s. Chỉ bật những thay đổi này cho các dự án đáng tin cậy.*

Ngoài ra, các mạng của nhà cung cấp chỉ xử lý kết nối lớp 2 cho các instance, do đó thiếu hỗ trợ cho các tính năng như bộ định tuyến và địa chỉ IP nổi (floating IP).

Trong nhiều trường hợp, nhà điều hành đã quen thuộc với kiến ​​trúc mạng ảo dựa trên cơ sở hạ tầng mạng vật lý cho lớp 2, lớp 3 hoặc các dịch vụ khác có thể triển khai liền mạch dịch vụ OpenStack Networking. Đặc biệt, các mạng của nhà cung cấp thu hút các nhà điều hành đang tìm cách chuyển đổi từ dịch vụ mạng Compute (nova-network) sang dịch vụ mạng Openstack. Theo thời gian, các nhà điều hành có thể xây dựng trên kiến ​​trúc tối thiểu này để kích hoạt nhiều tính năng mạng đám mây hơn.

Nói chung, các thành phần phần mềm Openstack Networking xử lý hoạt động của lớp 3 ảnh hưởng nhiều nhất đến hiệu suất và độ tin cậy. Để cải thiện, các mạng của nhà cung cấp chuyển các hoạt động của lớp 3 sang cơ sở hạ tầng mạng vật lý.

Trong một trường hợp sử dụng cụ thể, việc triển khai OpenStack nằm trong một môi trường hỗn hợp với máy chủ ảo hóa thông thường và bare-metal sử dụng cơ sở hạ tầng mạng vật lý lớn. Các ứng dụng chạy bên trong quá trình triển khai OpenStack có thể yêu cầu truy cập trực tiếp lớp 2, thường sử dụng VLAN, tới các ứng dụng bên ngoài quá trình triển khai.

1. Routed Provider network (Mạng nhà cung cấp được định tuyến)

Các mạng của nhà cung cấp được định tuyến cho phép kết nối lớp 3 giữa các instance. Các mạng này ánh xạ tới các mạng lớp 3 hiện có trong trung tâm dữ liệu. Cụ thể hơn, mạng ánh xạ tới các segment lớp 2, mỗi segment về cơ bản là một mạng nhà cung cấp. Mỗi cái có một cổng bộ định tuyến được gắn vào để định tuyến lưu lượng giữa chúng và bên ngoài. Dịch vụ mạng không cung cấp định tuyến.

Mạng nhà cung cấp được định tuyến cung cấp hiệu suất ở quy mô khó đạt được so với mạng nhà cung cấp đơn giản với chi phí kết nối lớp 2 được đảm bảo.

1. Self-service network (Mạng tự phục vụ)

Các mạng tự phục vụ chủ yếu cho phép các dự án chung (không có đặc quyền) quản lý mạng mà không cần đến quản trị viên. Các mạng này hoàn toàn ảo và yêu cầu các bộ định tuyến ảo tương tác với nhà cung cấp và các mạng bên ngoài như Internet. Các mạng tự phục vụ cũng thường cung cấp dịch vụ DHCP và metadata cho các instance.

Trong hầu hết các trường hợp, các mạng tự phục vụ sử dụng các giao thức overlay như VXLAN hoặc GRE vì chúng có thể hỗ trợ nhiều mạng hơn so với phân đoạn lớp 2 bằng cách sử dụng gắn thẻ VLAN (802.1q). Hơn nữa, Vlan thường yêu cầu cấu hình bổ sung của cơ sở hạ tầng mạng vật lý.

Các mạng tự phục vụ IPv4 thường sử dụng dải địa chỉ IP riêng (RFC1918) và tương tác với các mạng của nhà cung cấp thông qua NAT nguồn trên các bộ định tuyến ảo. Địa chỉ IP nổi cho phép truy cập vào các instance từ mạng của nhà cung cấp thông qua NAT đích trên bộ định tuyến ảo. Mạng tự phục vụ IPv6 luôn sử dụng dải địa chỉ IP công cộng và tương tác với mạng của nhà cung cấp thông qua bộ định tuyến ảo với các định tuyến tĩnh.

Dịch vụ mạng triển khai các bộ định tuyến bằng cách sử dụng agent lớp 3 thường được đặt tại ít nhất một nút mạng. Trái ngược với các mạng của nhà cung cấp kết nối các instance với cơ sở hạ tầng mạng vật lý ở lớp 2, các mạng tự phục vụ phải đi qua agent lớp 3. Do đó, việc đăng ký quá mức hoặc lỗi của agent lớp 3 hoặc nút mạng có thể ảnh hưởng đến một số lượng đáng kể các mạng tự phục vụ và các instance sử dụng chúng. Cân nhắc triển khai một hoặc nhiều tính năng có tính sẵn sàng cao để tăng khả năng dự phòng và hiệu suất của các mạng tự phục vụ.

Người dùng tạo các mạng dự án để kết nối trong các dự án. Theo mặc định, chúng được cách ly hoàn toàn và không được chia sẻ với các dự án khác. OpenStack Networking hỗ trợ các loại công nghệ overlay và cách ly mạng sau đây.

**Phẳng (Flat)**

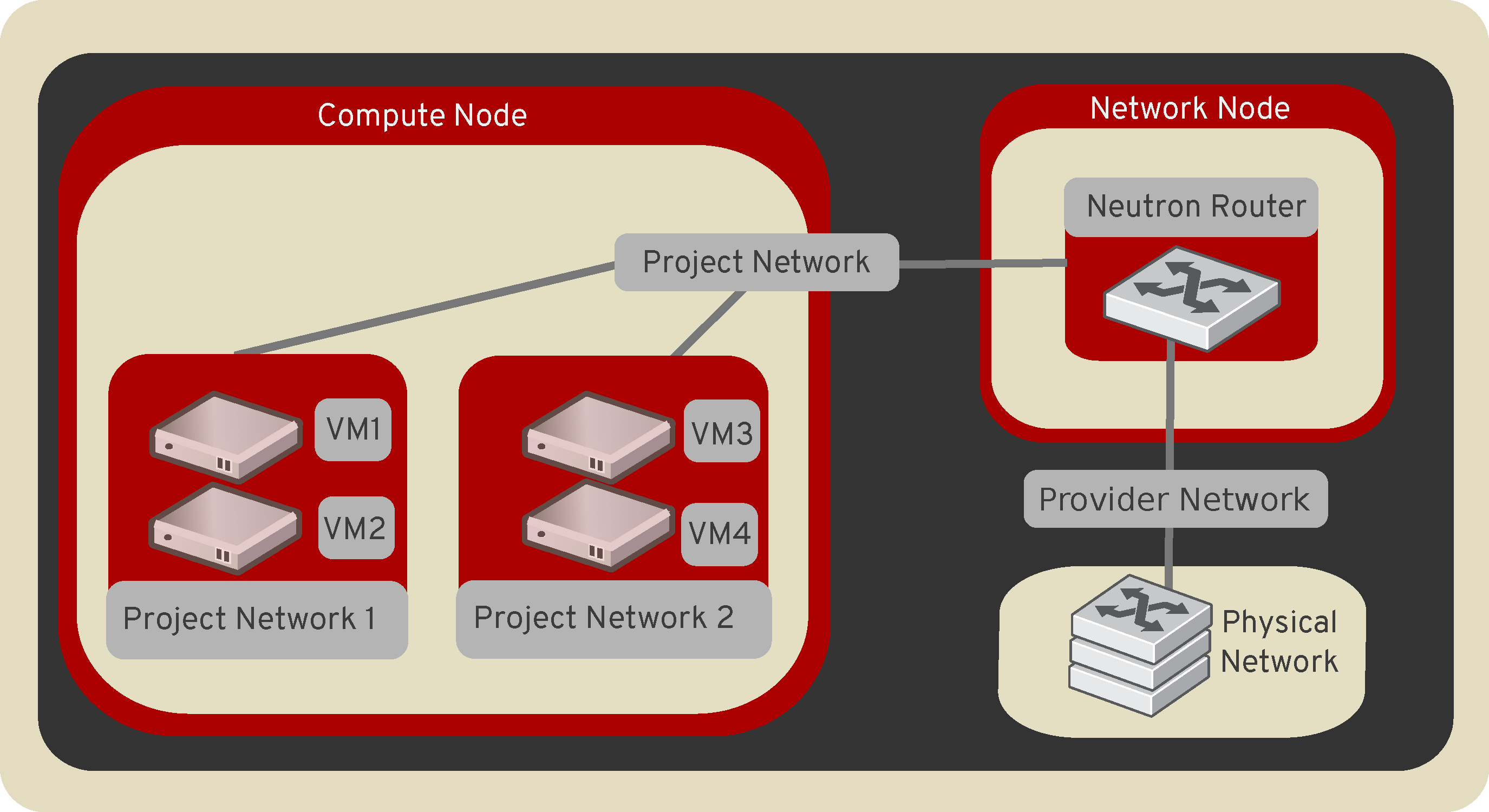
Tất cả các instance nằm trên cùng một mạng, mạng này cũng có thể được chia sẻ với các máy chủ. Không có sự gắn thẻ VLAN hoặc sự phân tách mạng nào khác diễn ra.

**VLAN**

Kết nối mạng cho phép người dùng tạo nhiều nhà cung cấp hoặc mạng dự án bằng cách sử dụng VLAN ID (được gắn thẻ 802.1Q) tương ứng với các VLAN có trong mạng vật lý. Điều này cho phép các phiên bản giao tiếp với nhau trên toàn môi trường. Họ cũng có thể giao tiếp với các máy chủ chuyên dụng, tường lửa, bộ cân bằng tải và cơ sở hạ tầng mạng khác trên cùng một VLAN lớp 2.

**GRE và VXLAN**

VXLAN và GRE là các giao thức đóng gói tạo ra các mạng lớp phủ để kích hoạt và kiểm soát giao tiếp giữa các compute instance. Cần có bộ định tuyến mạng để cho phép lưu lượng truy cập ra ngoài mạng dự án GRE hoặc VXLAN. Một bộ định tuyến cũng được yêu cầu để kết nối các mạng dự án được kết nối trực tiếp với các mạng bên ngoài, bao gồm cả Internet. Bộ định tuyến cung cấp khả năng kết nối với các instance trực tiếp từ mạng bên ngoài bằng địa chỉ IP nổi.

[](https://docs-openstack-org.translate.goog/neutron/rocky/_images/NetworkTypes.png?_x_tr_sl=en&_x_tr_tl=vi&_x_tr_hl=vi&_x_tr_pto=wapp)

1. Kiến trúc của mạng tự phục vụ

Các phần sau đây mô tả luồng lưu lượng mạng trong một số tình huống phổ biến. Lưu lượng mạng north-south di chuyển giữa một instance và mạng bên ngoài như Internet. Lưu lượng mạng theo hướng east-west di chuyển giữa các instance trên cùng một mạng hoặc các mạng khác nhau. Trong tất cả các tình huống, cơ sở hạ tầng mạng vật lý xử lý chuyển mạch và định tuyến giữa các mạng của nhà cung cấp và các mạng bên ngoài như Internet. Mỗi trường hợp tham chiếu một hoặc nhiều thành phần sau:

* Mạng nhà cung cấp (VLAN)
  + ID Vlan 101 (được gắn thẻ)
* Mạng tự phục vụ 1 (VXLAN)
  + ID VXlan (VNI) 101
* Mạng tự phục vụ 2 (VXLAN)
  + ID VXlan (VNI) 102
* Bộ định tuyến tự phục vụ
  + Gatewat mạng lưới nhà cung cấp
  + Giao diện trên mạng tự phục vụ 1
  + Giao diện trên mạng tự phục vụ 2
* Instance 1
* Instance 2

1. Không sử dụng DVR (Distributed Virtual Routing)
2. Network traffic flow: North -> South
   * + Instance với IP cố định:

Với instance có IP cố định, node mạng sẽ thực hiện SNAT cho traffic North -> South đi từ mạng tự phục vụ ra mạng bên ngoài (Internet).

Kịch bản thực hiện:

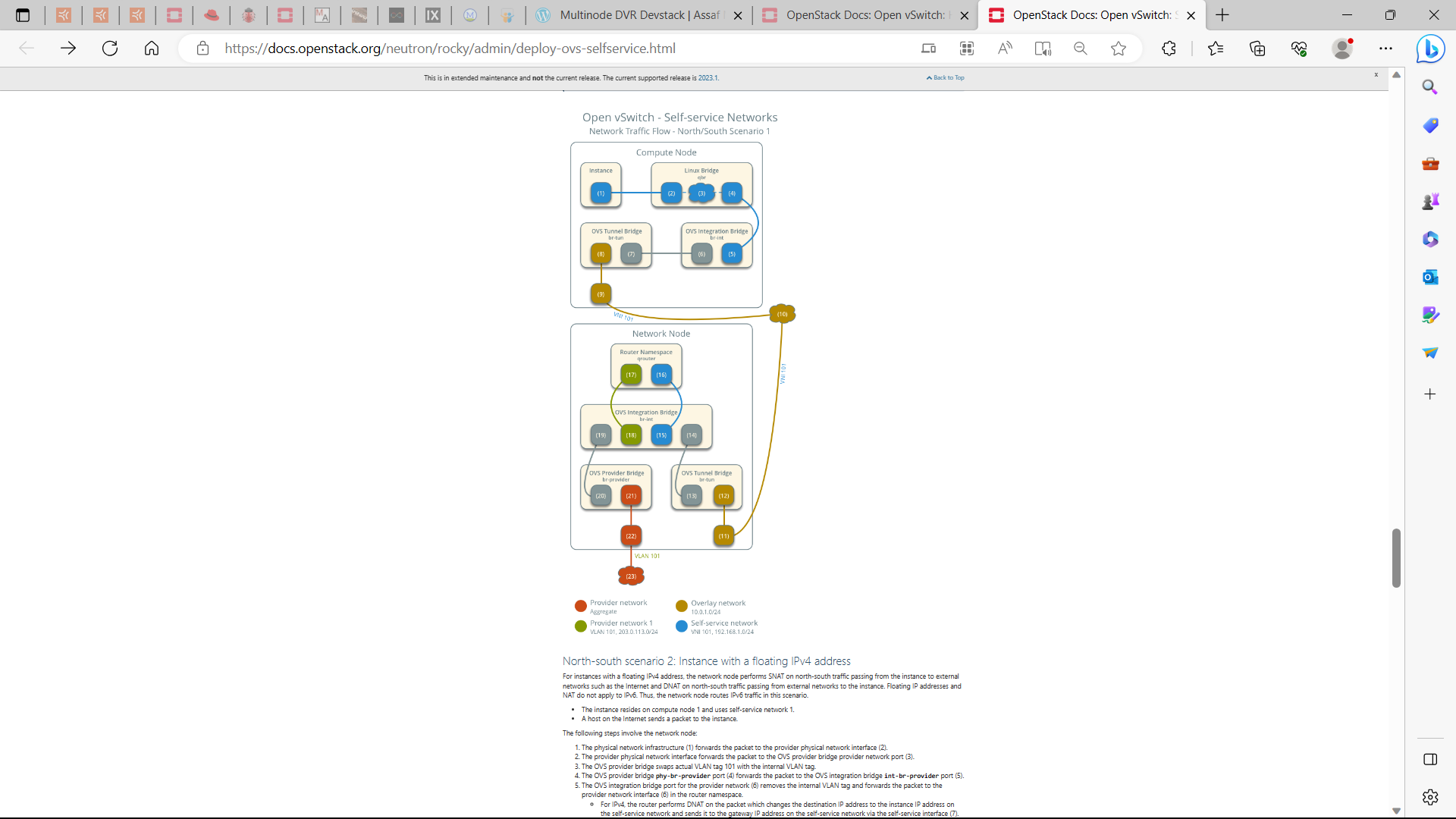
* Instance nằm trong compute node 1 và sử dụng mạng tự phục vụ 1.
* Instance gửi gói tin đến host nằm trên Internet

Các bước sau được thực hiện liên quan tới compute node 1:

1. Giao diện của instance (1) gửi gói tin đến cổng security group bridge (2) thông qua cặp **veth**
2. Quy tắc security group (3) trên bridge security group xử lý tường lửa và theo dõi kết nối cho gói.
3. Cổng security group bridge (4) chuyển tiếp gói đến cổng OVS integration bridge security group (5) qua cặp veth.
4. OVS integration bridge thêm thẻ VLAN nội bộ vào gói.
5. OVS integration bridge trao đổi thẻ VLAN bên trong để lấy ID đường hầm bên trong.
6. Cổng patch của OVS integration bridge (6) chuyển tiếp gói đến cổng patch OVS tunnel bridge (7).
7. OVS tunnel bridge (8) bọc gói bằng cách sử dụng VNI 101.
8. Giao diện vật lý mạng underlay (9) cho các mạng overlay chuyển tiếp gói đến nút mạng thông qua mạng overlay (10).

Các bước sau được thực hiện liên quan tới network node:

1. Giao diện vật lý mạng underlay (11) cho các mạng overlay chuyển tiếp gói đến OVS tunnel bridge (12).
2. OVS tunnel bridge mở gói và thêm ID đường hầm bên trong vào gói.
3. OVS tunnel bridge trao đổi ID đường hầm bên trong để lấy thẻ VLAN bên trong.
4. Cổng patch của OVS tunnel bridge (13) chuyển tiếp gói đến cổng patch của OVS integration bridge (14).
5. Cổng OVS integration bridge cho mạng tự phục vụ (15) loại bỏ thẻ VLAN bên trong và chuyển tiếp gói đến giao diện mạng tự phục vụ (16) trong không gian tên bộ định tuyến.
   * Đối với IPv4, bộ định tuyến thực hiện SNAT trên gói thay đổi địa chỉ IP nguồn thành địa chỉ IP của bộ định tuyến trên mạng nhà cung cấp và gửi nó đến địa chỉ IP gateway trên mạng nhà cung cấp thông qua giao diện gateway mạng nhà cung cấp (17).
   * Đối với IPv6, bộ định tuyến gửi gói đến địa chỉ IP chặng tiếp theo, thường là địa chỉ IP gateway trên mạng của nhà cung cấp, thông qua giao diện gateway nhà cung cấp (17).
6. Bộ định tuyến chuyển tiếp gói đến cổng OVS integration bridge cho mạng của nhà cung cấp (18).
7. OVS integration bridge thêm thẻ Vlan nội bộ vào gói.
8. Cổng patch của OVS integration bridge **int-br-provider**(19) chuyển tiếp gói đến cổng patch của OVS provider bridge **phy-br-provider** (20).
9. OVS provider bridge hoán đổi thẻ VLAN bên trong với thẻ VLAN thực tế 101.
10. Cổng mạng của nhà cung cấp OVS provider bridge  (21) chuyển tiếp gói đến giao diện mạng vật lý (22).
11. Giao diện mạng vật lý chuyển tiếp gói tới Internet thông qua cơ sở hạ tầng mạng vật lý (23).



* + - Instance với floating Ipv4:

Đối với các instance có địa chỉ floating Ipv4, nút mạng thực hiện SNAT trên lưu lượng truy cập north-south đi từ instance đến các mạng bên ngoài như Internet và DNAT trên lưu lượng truy cập north-south đi từ các mạng bên ngoài đến instance. Địa chỉ floating IP và NAT không áp dụng cho IPv6. Do đó, nút mạng định tuyến lưu lượng IPv6 trong trường hợp này.

* Instance nằm trên nút compute 1 và sử dụng mạng tự phục vụ 1.
* Một máy chủ lưu trữ trên Internet gửi một gói đến thể hiện.

Các bước sau liên quan đến nút mạng:

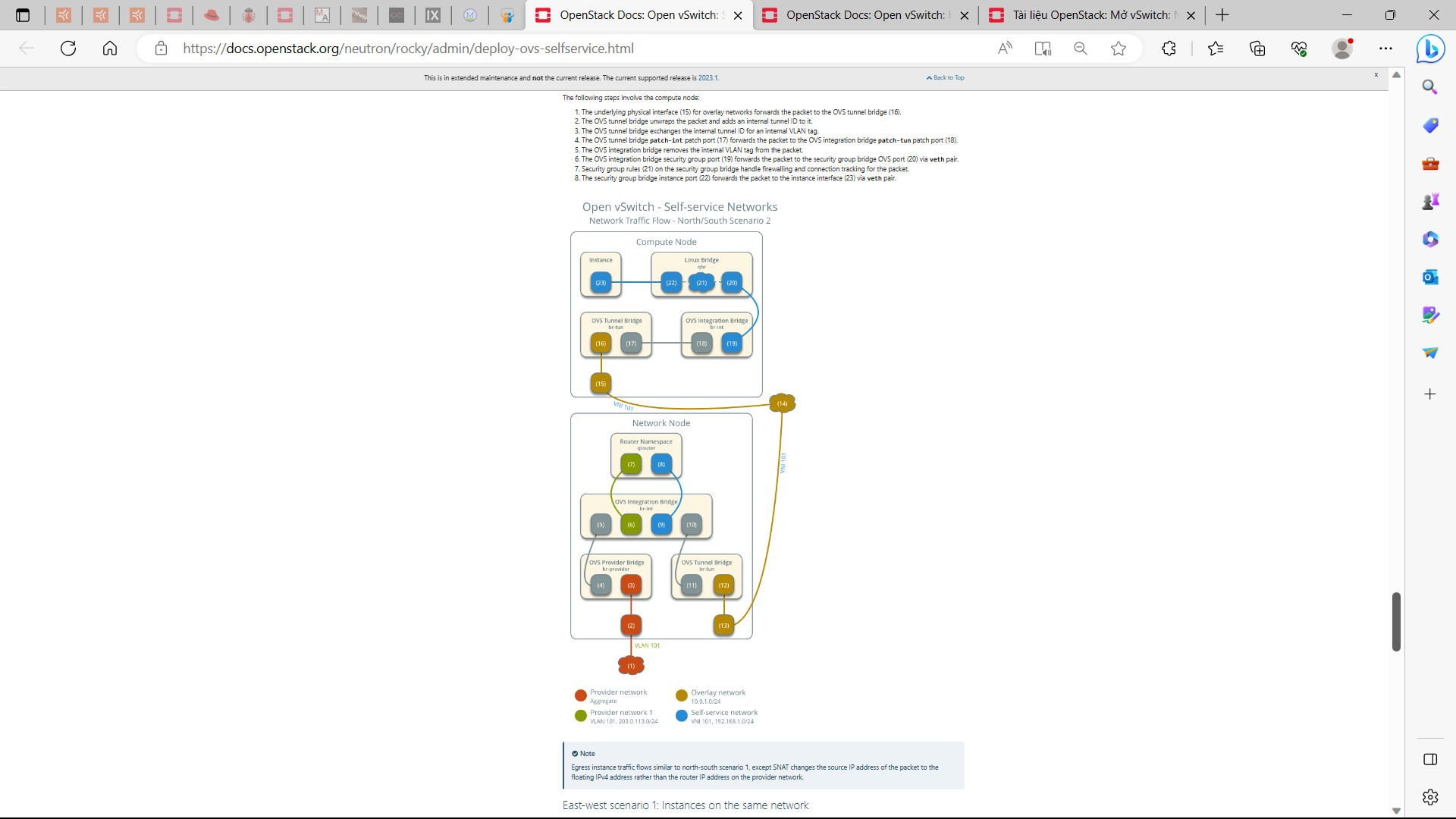
1. Cơ sở hạ tầng mạng vật lý (1) chuyển tiếp gói đến giao diện mạng vật lý của nhà cung cấp (2).
2. Giao diện mạng vật lý của nhà cung cấp chuyển tiếp gói đến cổng mạng của OVS provider bridge  (3).
3. OVS provider bridge hoán đổi thẻ VLAN thực tế 101 bằng thẻ VLAN bên trong.
4. Cổng OVS provider bridge  **phy-br-provider**(4) chuyển tiếp gói đến cổng OVS integration bridge OVS **int-br-provider**(5).
5. Cổng OVS integration bridge cho mạng nhà cung cấp (6) xóa thẻ Vlan bên trong và chuyển tiếp gói đến giao diện mạng nhà cung cấp (6) trong không gian tên bộ định tuyến.

* Đối với IPv4, bộ định tuyến thực hiện DNAT trên gói thay đổi địa chỉ IP đích thành địa chỉ IP instance trên mạng tự phục vụ và gửi nó đến địa chỉ IP gateway trên mạng tự phục vụ thông qua giao diện mạng tự phục vụ (7) .
* Đối với IPv6, bộ định tuyến gửi gói đến địa chỉ IP chặng tiếp theo, thường là địa chỉ IP gateway trên mạng tự phục vụ, thông qua giao diện mạng tự phục vụ (8).

1. Bộ định tuyến chuyển tiếp gói đến cổng OVS integration bridge cho mạng tự phục vụ (9).
2. OVS integration bridge thêm thẻ VLAN nội bộ vào gói.
3. OVS integration bridge trao đổi thẻ VLAN bên trong để lấy ID đường hầm bên trong.
4. Cổng patch OVS integration bridge **patch-tun**(10) chuyển tiếp gói đến cổng patch OVS tunnel bridge **patch-int** (11).
5. OVS tunnel bridge (12) bọc gói bằng cách sử dụng VNI 101.
6. Giao diện vật lý mạng underlay (13) cho các mạng overlay chuyển tiếp gói đến nút mạng thông qua mạng overlay (14).

Các bước sau liên quan đến nút tính toán:

1. Giao diện vật lý mạng underlay(15) cho các mạng overlay chuyển tiếp gói đến OVS tunnel bridge (16).
2. OVS tunnel bridge mở gói và thêm ID đường hầm bên trong vào gói.
3. OVS tunnel bridge trao đổi ID đường hầm bên trong để lấy thẻ VLAN bên trong.
4. Cổng patch OVS tunnel bridge **patch-int** (17) chuyển tiếp gói đến cổng patch OVS integration bridge **patch-tun** (18).
5. OVS integration bridge  loại bỏ thẻ VLAN bên trong khỏi gói.
6. Cổng OVS integration bridge (19) chuyển tiếp gói đến cổng nối OVS của security group bridge  (20) qua cặp **veth**.
7. Quy tắc security group (21) trên security group bridge  xử lý tường lửa và theo dõi kết nối cho gói.
8. Cổng security group bridge (22) chuyển tiếp gói đến giao diện instance (23) qua cặp **veth**.



1. Network traffic flow: East -> West
   * + Các instance nằm trên cùng 1 dải mạng:

Các instance có địa chỉ IPv4/IPv6 cố định hoặc địa chỉ floating IPv4 trên cùng một mạng giao tiếp trực tiếp giữa các nút compute chứa các instance đó.

Theo mặc định, giao thức VXLAN thiếu thông tin về vị trí mục tiêu và sử dụng multicast để khám phá vị trí đó. Sau khi tìm ra, nó lưu trữ vị trí trong cơ sở dữ liệu chuyển tiếp cục bộ. Trong các triển khai lớn, quy trình khám phá có thể tạo ra một lượng mạng đáng kể mà tất cả các nút phải xử lý. Để tăng hiệu quả, dịch vụ mạng sẽ dùng driver lớp 2 với cơ chế tự động điền vào cơ sở dữ liệu chuyển tiếp cho các giao diện VXLAN.

Kịch bản thực hiện:

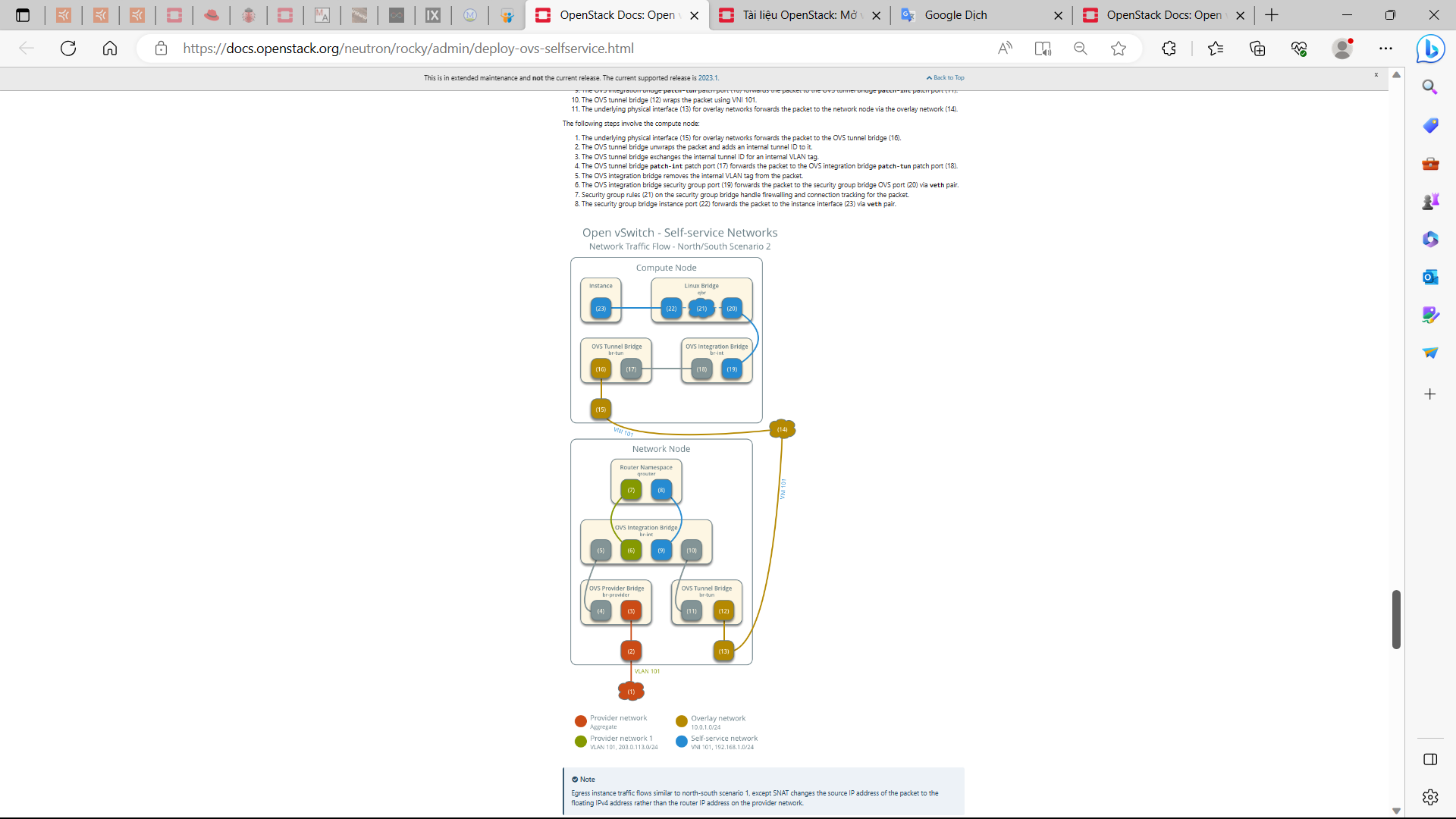
* Instance 1 nằm trên nút compute 1 và sử dụng mạng tự phục vụ 1.
* Instance 2 nằm trên nút compute 2 và sử dụng mạng tự phục vụ 1.
* Instance 1 gửi một gói đến instance 2.

Các bước sau đây liên quan đến nút compute 1:

1. Giao diện instance 1 (1) chuyển tiếp gói đến cổng security group bridge nối với instance(2) thông qua cặp **veth**.
2. Quy tắc security group (3) trên security group bridge xử lý tường lửa và theo dõi kết nối cho gói.
3. Cổng security group bridge (4) chuyển tiếp gói đến cổng OVS integration bridge  (5) qua cặp **veth**.
4. OVS integration bridge  thêm thẻ VLAN nội bộ vào gói.
5. OVS integration bridge  trao đổi thẻ VLAN bên trong để lấy ID đường hầm bên trong.
6. Cổng patch OVS integration bridge  (6) chuyển tiếp gói đến cổng patch OVS tunnel bridge  (7).
7. OVS tunnel bridge  (8) bọc gói bằng cách sử dụng VNI 101.
8. Giao diện vật lý mạng underlay (9) cho các mạng overlay chuyển tiếp gói để tính toán nút 2 thông qua mạng overlay (10).

Các bước sau đây liên quan đến tính toán nút 2:

1. Giao diện vật lý cơ bản (11) cho các mạng overlay chuyển tiếp gói đến OVS tunnel bridge  (12).
2. OVS tunnel bridge  mở gói và thêm ID đường hầm bên trong vào gói.
3. OVS tunnel bridge  trao đổi ID đường hầm bên trong để lấy thẻ VLAN bên trong.
4. Cổng patch OVS tunnel bridge  **patch-int**(13) chuyển tiếp gói đến cổng patch OVS integration bridge  **patch-tun** (14).
5. OVS integration bridge  loại bỏ thẻ VLAN bên trong khỏi gói.
6. Cổng OVS integration bridge (15) chuyển tiếp gói đến cổng nối OVS của security group bridge (16) qua cặp **veth**.
7. Quy tắc security group (17) trên security group bridge xử lý tường lửa và theo dõi kết nối cho gói.
8. Cổng security group bridge (18) chuyển tiếp gói đến giao diện instance 2 (19) qua cặp **veth**.



* + - Các instance nằm trên khác dải mạng:

Các trường hợp sử dụng địa chỉ IPv4/IPv6 cố định hoặc địa chỉ IPv4 nổi giao tiếp qua bộ định tuyến trên nút mạng. Các mạng tự phục vụ phải nằm trên cùng một bộ định tuyến.

* Phiên bản 1 nằm trên nút điện toán 1 và sử dụng mạng tự phục vụ 1.
* Phiên bản 2 nằm trên nút điện toán 1 và sử dụng mạng tự phục vụ 2.
* Trường hợp 1 gửi một gói đến trường hợp 2.

*Ghi chú*

*Cả hai phiên bản nằm trên cùng một nút điện toán để minh họa cách VXLAN cho phép nhiều lớp phủ sử dụng cùng một mạng lớp 3.*

Các bước sau liên quan đến nút tính toán:

1. Giao diện instance (1) chuyển tiếp gói đến cổng security group bridge (2) thông qua cặp **veth**.
2. Quy tắc security group (3) trên security group bridge xử lý tường lửa và theo dõi kết nối cho gói.
3. Cổng nối OVS của security group bridge (4) chuyển tiếp gói đến cổng nối security group của OVS integration bridge  (5) qua cặp **veth**.
4. OVS integration bridge  thêm thẻ VLAN nội bộ vào gói.
5. OVS integration bridge  trao đổi thẻ VLAN bên trong để lấy ID đường hầm bên trong.
6. Cổng patch OVS integration bridge  **patch-tun** (6) chuyển tiếp gói đến cổng patch OVS tunnel bridge **patch-int** (7).
7. OVS tunnel bridge  (8) bọc gói bằng cách sử dụng VNI 101.
8. Giao diện vật lý mạng underlay (9) cho các mạng overlay chuyển tiếp gói đến nút mạng thông qua mạng overlay (10).

Các bước sau liên quan đến nút mạng:

1. Giao diện vật lý mạng underlay (11) cho các mạng lớp overlay chuyển tiếp gói đến OVS tunnel bridge  (12).
2. OVS tunnel bridge mở gói và thêm ID đường hầm bên trong vào gói.
3. OVS tunnel bridge trao đổi ID đường hầm bên trong để lấy thẻ VLAN bên trong.
4. Cổng patch OVS tunnel bridge **patch-int** (13) chuyển tiếp gói đến cổng patch OVS integration bridge **patch-tun** (14).
5. Cổng OVS integration bridge cho mạng tự phục vụ 1 (15) xóa thẻ VLAN bên trong và chuyển tiếp gói đến giao diện mạng tự phục vụ 1 (16) trong không gian tên bộ định tuyến.
6. Bộ định tuyến gửi gói đến địa chỉ IP chặng tiếp theo, thường là địa chỉ IP gateway trên mạng tự phục vụ 2, thông qua giao diện mạng tự phục vụ 2 (17).
7. Bộ định tuyến chuyển tiếp gói đến cổng OVS integration bridge cho mạng tự phục vụ 2 (18).
8. OVS integration bridge thêm thẻ Vlan nội bộ vào gói.
9. OVS integration bridge trao đổi thẻ VLAN bên trong để lấy ID đường hầm bên trong.
10. Cổng patch OVS integration bridge **patch-tun**(19) chuyển tiếp gói đến cổng patch OVS tunnel bridge **patch-int**(20).
11. OVS tunnel bridge (21) bọc gói bằng cách sử dụng VNI 102.
12. Giao diện vật lý mạng underlay (22) cho các mạng lớp overlay chuyển tiếp gói đến nút compute thông qua mạng lớp overlay (23).

Các bước sau liên quan đến nút tính toán:

1. Giao diện vật lý mạng underlay (24) cho các mạng lớp overlay chuyển tiếp gói đến OVS tunnel bridge  (25).
2. OVS tunnel bridge mở gói và thêm ID đường hầm bên trong vào gói.
3. OVS tunnel bridge trao đổi ID đường hầm bên trong để lấy thẻ VLAN bên trong.
4. Cổng patch OVS tunnel bridge  **patch-int** (26) chuyển tiếp gói đến cổng patch OVS integration bridge **patch-tun** (27).
5. OVS integration bridge loại bỏ thẻ VLAN bên trong khỏi gói.
6. Cổng nối security group của integration bridge (28) chuyển tiếp gói đến cổng nối OVS của security group bridge (29) qua vethcặp.
7. Quy tắc security group (30) trên security group bridge xử lý tường lửa và theo dõi kết nối cho gói.
8. Cổng nối instance của security group bridge (31) chuyển tiếp gói đến giao diện instance(32) qua cặp **veth**.
9. Tính khả dụng cao sử dụng VRRP
   * 1. VRRP là gì?

Virtual Redudant Routing Protocol: Là một giao thức chống lỗi giải quyết đề khi router là SPOF (Single Point of Failure) trong mạng. VRRP gán nhiệm vụ định tuyến cho một VRRP router từ tập các virtual router thông qua giao thức biểu quyết.

Các thuật ngữ liên quan tới VRRP:

**Virtual Router:** Bao gồm Master router và các Backup router. Host sử dụng virtual router là gateway mặc định.

**VRID:** Là ID của một virtual router, một nhóm các router vật lý có cùng VRID sẽ tạo thành một virtual router. Thường sẽ từ (0-255)

**Master Router:** Là router đảm nhiệm nhiệm vụ chuyển tiếp gói tin.

**Backup Router:** Là router sẽ thay thế Master router nếu Master router gặp lỗi.

**Virtual IP:** Địa chỉ IP của virtual router, một virtual router có thể có 1 hoặc nhiều địa chỉ IP

**IP address owner:** Là router có địa chỉ IP tại giao diện giống với Virtual IP

**Virtual MAC address:** Một Virtual router có một địa chỉ MAC ảo. Định dạng của địa chỉ MAC là 00-00-5E-00-01-{VRID}. Thông thường, virtual router sẽ phản hồi lại ARP request với địa chỉ MAC ảo, và chỉ khi được cấu hình, virtual router mới phản hồi bằng địa chỉ MAC thật trên giao diện.

**Priority:** VRRP xác định trạng thái của từng router trong Virtual Router dựa trên độ ưu tiên (priority). Biểu diễn từ 0-255, số biểu diễn càng lớn thì độ ưu tiên càng cao. VRRP cho phép cấu hình từ 1-254, độ ưu tiên 0 được dùng cho hệ thống khi router từ bỏ trí Master, và 255 được dùng để gán cho router là IP address owner. Vì vậy, nếu router là IP address owner, và đang hoạt động bình thường, router đó sẽ là Master router.

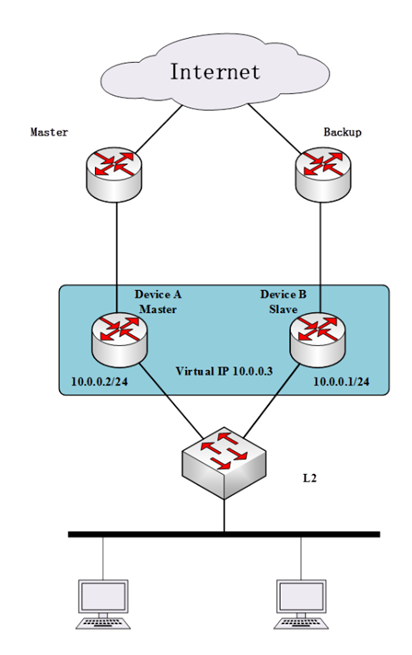
**Preemption method:** Mặc định, nếu Backup router hoạt động ở chế độ ưu tiên (preemption mode), nó sẽ so sánh priority của nó với priority nằm trong bản tin VRRP advertisemet. Nếu priority của nó lớn hơn của Master router, nó sẽ chủ động chiếm quyền và trở thành Master router; Nếu không, nó sẽ trở lại trạng thái Backup

**Non-preemption mode:** Nếu Backup router hoạt động ở chế độ không ưu tiên, miễn là Master router không gặp lỗi, Backup router sẽ không trở thành Master router kể cả khi được cấu hình có priority cao hơn.

**VRRP protocol packet:** Được đóng gói trong bản tin IP và gửi đến địa chỉ IP multicast được gán cho VRRP. Trong header của gói tin IP, địa chỉ nguồn là địa chỉ IP chính của giao diện gửi tin (không phải Virtual IP), địa chỉ đích là 224.0.0.18, TTL là 255, và protocol number là 112. Hiện tại VRRP có 2 phiên bản: VRRPv2 và VRRPv3. VRRPv2 chỉ có thể sử dụng trong mạng Ipv4, VRRP v3 có thể sử dụng cho cả Ipv4 và Ipv6A screenshot of a computer

Description automatically generated

**3 Trạng thái của router VRRP:** Trạng thái khởi tạo (Initialize), trạng thái active (Master), và trạng thái backup (Backup). Trong số chúng, chỉ có thiết bị ở trạng thái Master mới có thể chuyển tiếp các gói tin được gửi tới Virtual IP.



Cơ chế của VRRP:

* Thiết bị A và B sẽ tạo thành 1 nhóm VRRP, và Virtual IP (VIP) sẽ là 10.0.0.3/24. Thông qua việc biểu quyết, A sẽ là Master router, B sẽ là Backup router. Có thể có nhiều thiết bị trong 1 nhóm VRRP nhưng sẽ chỉ có 1 thiết bị là Master. Lưu ý: A và B có thể có địa chỉ IP riêng, và VIP có thể giống hoặc khác địa chỉ của A và B.
* Hiện tại, Router A đóng vai trò là Master router, cung cấp các dịch vụ định tuyến cho các máy trong mạng LAN, và Router B đóng vai trò là Backup router. Nhiệm vụ của B là nhận gói tin heartbeat được gửi bởi A theo một chu kỳ. Trong một khoảng thời gian quy định, nếu không có heartbeat nào được nhận từ A, tiến trình bầu cử sẽ được bắt đầu để bầu ra Master mới
* Các máy trong mạng LAN sẽ dùng virtual router là gateway mặc định. **Chúng chỉ biết địa chỉ của virtual router là 10.0.0.3, nhưng không biết địa chỉ chính xác của Master router và Backup router.** Các máy đó sẽ cài định tuyến mặc định cho hop tiếp theo là 10.0.0.3. Vì vậy, các máy chủ trong mạng sẽ giao tiếp với mạng khác qua virtual router. Nếu Master router gặp lỗi, Backup router sẽ chọn Master router mới qua bầu cử, và tiếp tục cung cấp khả năng định tuyến cho các máy chủ trong mạng. Do đó, các máy chủ có thể giao tiếp với mạng bên ngoài mà không bị gián đoạn.

Lợi ích của việc dùng VRRP:

* Hoạt động đơn giản: Không cần phải thay đổi tình hình mạng, cũng như không cần thực hiện cấu hình trên máy chủ. Chỉ cần cấu hình bằng một số lệnh trên các router liên quan để tạo backup cho next-hop gateway, và không ảnh hưởng tới các máy chủ.
* Quản lý mạng đơn giản: Trong mạng cục bộ (VD: Ethernet), với khả năng dùng multicast và broadcast, VRRP có thể cung cấp một link mặc định với độ tin cậy cao khi một thiết bị nào đó gặp lỗi, giúp tránh lỗi mạng một cách hiệu quả sau khi một link bị lỗi. Không cần phải thay đổi thông tin cấu hình như là giao thức định tuyến động và giao thức khám phá tuyến đường, và cũng như không phải sửa gateway mặc định trên máy chủ.
* Khả năng thích ứng mạnh: gói tin VRRP được đóng gói trong bản tin IP và hỗ trợ các giao thức tầng trên khác nhau
* Chi phí mạng nhỏ: VRRP chỉ định nghĩa một loại tin nhắn – VRRP notification message, và chỉ có Master router có thể gửi VRRP message.
  + 1. Keepalived?

Keepalived là một triển khai mã nguồn mở của VRRP. Đây là một giải pháp có tính sẵn sàng cao cho các dịch vụ WEB dựa trên giao thức VRRP, có thể được sử dụng để tránh các SPOF. Trong tính sẵn sàng định tuyến cao (HA) của neutron, bạn cần cài đặt keepalived.

Các thành phần của Keepalived:

* Keepalived được thiết kế theo modun, và với các modun khác nhau thì sẽ làm các chức năng khác nhau:
* Core: Lõi của keepalived, chịu trách nhiệm cho việc khởi động các tiến trình chính và duy trì file cấu hình global cho việc loading và sharing.
* Check: Thực hiện việc health check, bao gồm nhiều phương pháp kiểm tra tình trạng và các file cấu hình tương ứng.
* VRRPd: Tiến trình phụ dùng để triển khai VRRP

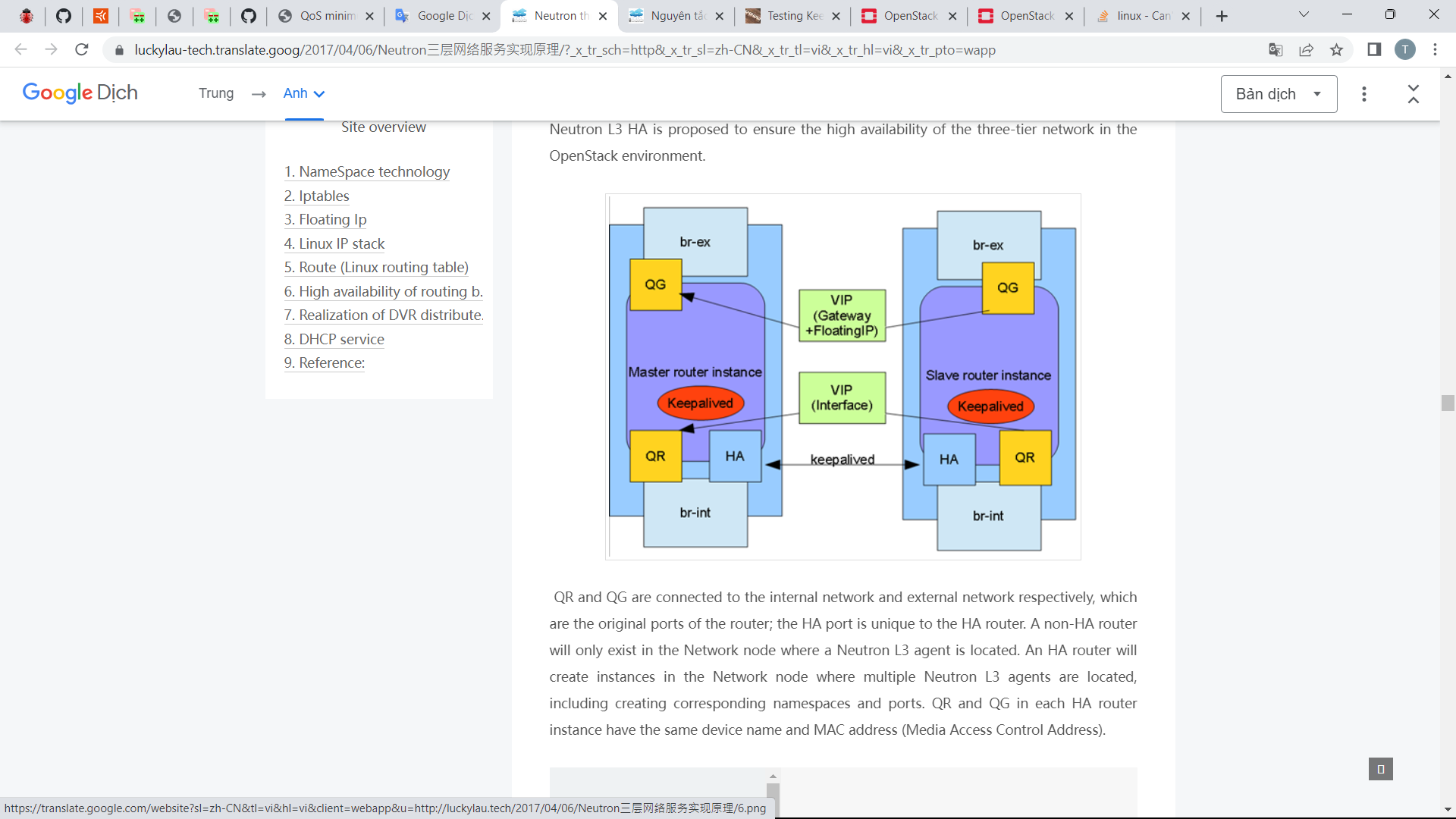
Tiến trình hoạt động của keepalived:

* Keepalived sử dụng 3 tiến trình: Watchdog là tiến trình điều khiển, và VRRP Stack và Checkers là hai tiến trình con.
* Watchdog sử dụng cơ chế heartbeat để đảm bảo tiến trình con đang chạy.
* Checkers: Chịu trách nhiệm phát hiện tình trạng của các máy chủ cho việc cân bằng tải.
* VRRP Stack: Thực hiện VRRP và cung cấp HA.

Tóm gọn lại Keepalived cho 2 chức năng chính:

* Cân bằng tải: health checking các máy chủ trong mô hình HA và các phương thức cân bằng tải xuống server Backend
* Tạo độ sẵn sàng cao: Chức năng VRRP đảm nhận quản lý khả năng chịu lỗi của cụm server (Failover) với Virtual IP.
  + 1. Cách neutron triển khai

Neutron L3 HA được đề xuất để đảm bảo tính sẵn sàng cao của mạng Openstack



QR và QG theo thứ tự được kết nối với mạng nội bộ và mạng bên ngoài, là các cổng ban đầu của router; port HA là duy nhất với HA router. Một router non-HA sẽ chỉ tồn tại trong một Network node có Neutron L3 agent. Một HA router sẽ tạo các phiên bản trên các Network node có chứa nhiều Neutron L3 agent, và cũng như tạo các namespaces và ports tương ứng. QR và QG trên từng các phiên bản HA router có cùng tên thiết bị và địa chỉ MAC.

#**Slave information**

root@netagent10038220:~# ip netns exec **qrouter-23d7334d-a55f-4b4b-9dfa-4d1ee4b3080b** ip addr

1: lo: <LOOPBACK,UP,LOWER\_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1

link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00

inet 127.0.0.1/8 scope host lo

valid\_lft forever preferred\_lft forever

inet6 ::1/128 scope host

valid\_lft forever preferred\_lft forever

26: **ha-557abd6c-48**: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 8950 qdisc noqueue state UNKNOWN group default qlen 1

link/ether fa:16:3e:3f:75:8d brd ff:ff:ff:ff:ff:ff

inet **169.254.192.2/18** brd 169.254.255.255 scope global ha-557abd6c-48

valid\_lft forever preferred\_lft forever

inet6 fe80::f816:3eff:fe3f:758d/64 scope link

valid\_lft forever preferred\_lft forever

29: **qr-fe09d52e-c6**: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 8950 qdisc noqueue state UNKNOWN group default qlen 1

link/ether fa:16:3e:58:13:ad brd ff:ff:ff:ff:ff:ff

32: **qg-21bed7aa-c6**: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc noqueue state UNKNOWN group default qlen 1

link/ether fa:16:3e:a9:28:e7 brd ff:ff:ff:ff:ff:ff

#**Master's message**

root@netagent10038219:~# ip netns exec **qrouter-23d7334d-a55f-4b4b-9dfa-4d1ee4b3080b** ip addr

1: lo: <LOOPBACK,UP,LOWER\_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1

link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00

inet 127.0.0.1/8 scope host lo

valid\_lft forever preferred\_lft forever

inet6 ::1/128 scope host

valid\_lft forever preferred\_lft forever

28: **ha-65bc01fe-af**: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 8950 qdisc noqueue state UNKNOWN group default qlen 1

link/ether fa:16:3e:1f:d2:bd brd ff:ff:ff:ff:ff:ff

inet 169.254.192.1/18 brd 169.254.255.255 scope global ha-65bc01fe-af

valid\_lft forever preferred\_lft forever

inet **169.254.0.1/24** scope global ha-65bc01fe-af

valid\_lft forever preferred\_lft forever

inet6 fe80::f816:3eff:fe1f:d2bd/64 scope link

valid\_lft forever preferred\_lft forever

30: **qr-fe09d52e-c6**: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 8950 qdisc noqueue state UNKNOWN group default qlen 1

link/ether fa:16:3e:58:13:ad brd ff:ff:ff:ff:ff:ff

inet 1.1.1.1/24 scope global qr-fe09d52e-c6

valid\_lft forever preferred\_lft forever

inet6 fe80::f816:3eff:fe58:13ad/64 scope link nodad

valid\_lft forever preferred\_lft forever

32: **qg-21bed7aa-c6**: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc noqueue state UNKNOWN group default qlen 1

link/ether fa:16:3e:a9:28:e7 brd ff:ff:ff:ff:ff:ff

inet 10.99.1.101/32 scope global qg-21bed7aa-c6

valid\_lft forever preferred\_lft forever

inet 10.99.1.102/32 scope global qg-21bed7aa-c6

valid\_lft forever preferred\_lft forever

inet 10.99.1.103/24 scope global qg-21bed7aa-c6

valid\_lft forever preferred\_lft forever

inet6 fe80::f816:3eff:fea9:28e7/64 scope link nodad

valid\_lft forever preferred\_lft forever

**Sự khác biệt giữa Slave và Master router là cổng mạng của Master router có địa chỉ IP. Những địa chỉ IP này chính là VIPs, chúng sẽ chỉ tồn tại trên Master router.** Từng router sẽ có HA port, và trên port sẽ có địa chỉ IP khác nhau. Các port này được dùng để cho giao tiếp VRRP broadcast. Ở trên mỗi router sẽ có chạy tiến trình keepalived. Keepalived là một tool phần mềm thực hiện VRRP. Neutron sử dụng keepalived để triển khai L3 HA

Trên Master, ta có thể kiểm tra bản tin VRRP broadcast

oot@netagent10038219:~# ip netns exec **qrouter-23d7334d-a55f-4b4b-9dfa-4d1ee4b3080b** tcpdump -i **ha-65bc01fe-af**

tcpdump: verbose output suppressed, use -v or -vv for full protocol decode

listening on ha-65bc01fe-af, link-type EN10MB (Ethernet), capture size 262144 bytes

05:01:11.336576 IP **169.254.192.1 > 224.0.0.18**: **VRRPv2, Advertisement**, **vrid 1**, **prio 50**, authtype none, **intvl 2s**, length 20

05:01:13.337818 IP **169.254.192.1 > 224.0.0.18**: **VRRPv2, Advertisement**, **vrid 1**, **prio 50**, authtype none, **intvl 2s**, length 20

05:01:15.339083 IP **169.254.192.1 > 224.0.0.18**: **VRRPv2, Advertisement**, **vrid 1**, **prio 50**, authtype none, **intvl 2s**, length 20

HA port gửi thông tin đến địa chỉ broadcast 224.0.0.18. Đây là địa chỉ được chỉ định trong giao thức VRRP. Nếu Master router lỗi, nó sẽ không thể gửi thông tin broadcast, dẫn đến các Slave router không nhận được bản tin broadcast trong khoảng một thời gian. Một trong các Slave router sẽ được chọn để trở thành Master router. Nó sẽ nhận được VIP để cung cấp mạng cho Openstack. Mặc dù router bị thay đổi, nhưng địa chỉ IP thì không, nên các máy chủ trong mạng sẽ không bị ảnh hưởng.

Vrid là id được tương ứng cho từng HA router, dưới từng tenant, từng router khác nhau sẽ tương ứng với các vrid khác nhau. Bởi vì vrid chỉ có 8 bits data, nên sẽ **chỉ có tối đa 255 HA routers được tạo ra trong 1 tenant**. prio là độ ưu tiên của router, hiện không thể cấu hình, và độ ưu tiên của các router là như nhau. authtype là phương pháp xác thực (Authentication) cho việc giao tiếp giữa các router, mặc định là none. Intvl là khoảng thời gian giữa những lần gói broadcast được gửi, mặc định là 2s, và có thể được cấu hình.

Khi HA router được tạo, Neutron L3 agent sẽ tạo các tiến trình keepalived cho từng router. Trước đó, file cấu hình keepalived sẽ được tạo cho các tiến trình sử dụng

root@**netagent10038219**:~# ps -aux | grep '23d7334d-a55f-4b4b-9dfa-4d1ee4b3080b'

root 4310 0.0 0.8 155892 73080 ? S Mar17 0:05 /usr/bin/python /usr/local/bin/neutron-keepalived-state-change --router\_id=23d7334d-a55f-4b4b-9dfa-4d1ee4b3080b --namespace=qrouter -23d7334d-a55f-4b4b-9dfa-4d1ee4b3080b --conf\_dir=/var/lib/neutron/ha\_confs/23d7334d-a55f-4b4b-9dfa-4d1ee4b3080b --monitor\_interface=ha-65bc01fe-af --monitor\_cidr=169. 254.0.1/24 --pid\_file=/var/lib/neutron/external/pids/23d7334d-a55f-4b4b-9dfa-4d1ee4b3080b.monitor.pid --state\_path=/var/lib/neutron --user=0 --group=0

root 4810 0.0 0.0 54252 564 ? Ss Mar17 0:59 keepalived -P -f **/var/lib/neutron/ha\_confs/**23d7334d-a55f-4b4b-9dfa-4d1ee4b3080b/keepalived.conf -p **/var/lib/neutron/ha\_confs/** 23d7334d-a55f-4b4b-9dfa-4d1ee4b3080b.pid -r **/var/lib/neutron/ha\_confs/**23d7334d-a55f-4b4b-9dfa-4d1ee4b3080b.pid-vrrp

root 4811 0.0 0.0 56496 3856 ? S Mar17 2:31 keepalived -P -f /var/lib/neutron/ha\_confs/23d7334d-a55f-4b4b-9dfa-4d1ee4b3080b/keepalived.conf -p /var/lib/neutron/ha\_confs/ 23d7334d-a55f-4b4b-9dfa-4d1ee4b3080b.pid -r /var/lib/neutron/ha\_confs/23d7334d-a55f-4b4b-9dfa-4d1ee4b3080b.pid-vrrp

root 6330 0.0 0.8 140156 71188 ? S Mar28 0:02 /usr/bin/python /usr/local/bin/neutron-ns-metadata-proxy --pid\_file=/var/lib/neutron/external/pids/23d7334d-a55f -4b4b-9dfa-4d1ee4b3080b.pid --metadata\_proxy\_socket=/var/lib/neutron/metadata\_proxy --router\_id=23d7334d-a55f-4b4b-9dfa-4d1ee4b3080b --state\_path=/var/lib/neutron --metadata\_port=9697 -- metadata\_proxy\_user=0 --metadata\_proxy\_group=0 --log-file=neutron-ns-metadata-proxy-23d7334d-a55f-4b4b-9dfa-4d1ee4b3080b.log --log-dir=/var/log/neutron

root 28706 0.0 0.0 13080 2544 pts/1 S+ 05:47 0:00 grep --color=auto 23d7334d-a55f-4b4b-9dfa-4d1ee4b3080b

root@netagent10038219:**/var/lib/neutron/ha\_confs/**23d7334d-a55f-4b4b-9dfa-4d1ee4b3080b# ls

keepalived.conf neutron-keepalived-state-change.log state

vrrp\_instance VR\_1 {

state BACKUP

interface **ha-65bc01fe-af**

virtual\_router\_id 1

priority 50

garp\_master\_delay 60

nopreempt

advert\_int 2

track\_interface {

ha-65bc01fe-af

}

virtual\_ipaddress {

169.254.0.1/24 dev ha-65bc01fe-af

}

virtual\_ipaddress\_excluded {

1.1.1.1/24 dev qr-fe09d52e-c6

10.99.1.101/32 dev qg-21bed7aa-c6

10.99.1.102/32 dev qg-21bed7aa-c6

10.99.1.103/24 dev qg-21bed7aa-c6

fe80::f816:3eff:fe58:13ad/64 dev qr-fe09d52e-c6 scope link

fe80::f816:3eff:fea9:28e7/64 dev qg-21bed7aa-c6 scope link

}

virtual\_routes {

0.0.0.0/0 via 10.99.1.1 dev qg-21bed7aa-c6

}

}

root@**netagent10038220**:~# ps -aux | grep '23d7334d-a55f-4b4b-9dfa-4d1ee4b3080b'

root 4330 0.0 0.8 155896 72840 ? S Mar17 0:04 /usr/bin/python /usr/local/bin/neutron-keepalived-state-change --router\_id=23d7334d-a55f-4b4b-9dfa-4d1ee4b3080b --namespace=qrouter -23d7334d-a55f-4b4b-9dfa-4d1ee4b3080b --conf\_dir=/var/lib/neutron/ha\_confs/23d7334d-a55f-4b4b-9dfa-4d1ee4b3080b --monitor\_interface=ha-557abd6c-48 --monitor\_cidr=169 .254.0.1/24 --pid\_file=/var/lib/neutron/external/pids/23d7334d-a55f-4b4b-9dfa-4d1ee4b3080b.monitor.pid --state\_path=/var/lib/neutron --user=0 --group=0

root 4882 0.0 0.0 54252 568 ? Ss Mar17 0:59 keepalived -P -f **/var/lib/neutron/ha\_confs/**23d7334d-a55f-4b4b-9dfa-4d1ee4b3080b/keepalived.conf -p **/var/lib/neutron/ha\_confs/** 23d7334d-a55f-4b4b-9dfa-4d1ee4b3080b.pid -r **/var/lib/neutron/ha\_confs/**23d7334d-a55f-4b4b-9dfa-4d1ee4b3080b.pid-vrrp

root 4883 0.0 0.0 56540 4304 ? S Mar17 1:46 keepalived -P -f /var/lib/neutron/ha\_confs/23d7334d-a55f-4b4b-9dfa-4d1ee4b3080b/keepalived.conf -p /var/lib/neutron/ha\_confs/ 23d7334d-a55f-4b4b-9dfa-4d1ee4b3080b.pid -r /var/lib/neutron/ha\_confs/23d7334d-a55f-4b4b-9dfa-4d1ee4b3080b.pid-vrrp

root 25801 0.0 0.0 13080 2548 pts/0 S+ 05:48 0:00 grep --color=auto 23d7334d-a55f-4b4b-9dfa-4d1ee4b3080b

root@netagent10038220**:/var/lib/neutron/ha\_confs**/23d7334d-a55f-4b4b-9dfa-4d1ee4b3080b# ls

keepalived.conf neutron-keepalived-state-change.log state

vrrp\_instance VR\_1 {

state BACKUP

interface **ha-557abd6c-48**

virtual\_router\_id 1

priority 50

garp\_master\_delay 60

nopreempt

advert\_int 2

track\_interface {

ha-557abd6c-48

}

virtual\_ipaddress {

169.254.0.1/24 dev ha-557abd6c-48

}

virtual\_ipaddress\_excluded {

1.1.1.1/24 dev qr-fe09d52e-c6

10.99.1.101/32 dev qg-21bed7aa-c6

10.99.1.102/32 dev qg-21bed7aa-c6

10.99.1.103/24 dev qg-21bed7aa-c6

fe80::f816:3eff:fe58:13ad/64 dev qr-fe09d52e-c6 scope link

fe80::f816:3eff:fea9:28e7/64 dev qg-21bed7aa-c6 scope link

}

virtual\_routes {

0.0.0.0/0 via 10.99.1.1 dev qg-21bed7aa-c6

}

}

Về tổng quan, để sử dụng HA router, cần phải thiết lập nhiều Neutron L3 agents trên các Network node khác nhau. Khi HA router được tạo, Neutron sẽ tạo các namespaces và ports tương ứng qua các L3 agents, do đó sẽ có nhiều phiên bản của HA router này. Cũng tại thời điểm đó, L3 agent sẽ tạo tiến trình keepalived và từng tiến trình keepalived có tất cả các thông tin của router (như hình trên). Theo cách này, mỗi Network node sẽ có một phiên bản của HA router và một tiến trình keepalived quản lý phiên bản router đó.

Khi mọi thứ đã sẵn sàng, các phiên bản của router sẽ tiến hành bầu chọn, phiên bản bầu chọn thắng sẽ trở thành Master, và những phiên bản còn lại sẽ là Slave. Bởi vì tất cả router đều có chung độ ưu tiên, kết quả của bầu chọn sẽ là ngẫu nhiên. Điều đó có nghĩa là, nếu nhiều HA routers được tạo, phiên bản Master của router sẽ được phân phối trên nhiều Network node

Sau quá trình bầu chọn, Master router sẽ đảm nhiệm việc cung cấp các dịch vụ mạng L3, đồng thời broadcast đến các Slave router khác để báo cáo về tình trạng của nó. Khi quá trình broadcast bị gián đoạn, Slave router sẽ tiến hành bầu chọn lại Master router mới để cung cấp các dịch vụ mạng L3, và đồng thời gửi broadcast để báo cáo tình trạng của nó.