OpenStack Networking - Netron

XLCloud-Networking_Needs_Quantum.pdf: continuar da página 9

Guias de Integração com Openflow:

Floodlight

Trema

Ryu

RestProxy Plugin

OpenVSwitch

Neutron

Serviço OpenStack que lida com rede (OpenStack Networking)

- é uma API disponível para os tenants
- oferece recursos dinamicamente
- criar e configurar redes virtuais

- Neutron é apenas uma camada de abstração que unifica as chamadas de configuração. Quem realmente faz o trabalho é o Neutron Plugin.
- Neutron controla a virtualização de rede (network services) da mesma forma que Nova controla a virtualização de servidores e instâncias.
- Neutron fornece uma API para gerenciamento de rede pelos projetos (tenants)
- Neutron pode definir topologias de rede complexas
- Neutron consegue lidar com diferentes tecnologias de virtualização de rede.

Implementação de referência

Neutron-PhysNet-Diagram.png
Retirada do OpenStack Admin Guide

- cloud controller host (Cloud Controller Node): quantum-server, nova-api, nova-scheduler, keystone, RabbitMQ, MySQL.
- network gateway host (Network Node): quantum-metadata-agent, quantum-dhcp-agent, quantum-l3-agent, quantum-*-plugin-agent
- conjunto de hypervisors para rodar VMs. Compute node: nova-compute, quantum-*-plugin-agent

cloud controller host e network gateway host podem ser combinados em uma máquina, porém é recomendado que exista um network gateway host separado para evitar restrições de CPU entre os encaminhamentos realizados pelo quantum-l3-agent e outros serviços OS.

- DataNetwork: comunicação de dados dentro da implementação de nuvem.
- Management Network: comunicações entre componentes OS.
- External Network: oferece acesso à internet à implementação OS.
- API Network: expõe todas as APIs do OS, incluindo a Netron API para os tenants. Pode ser a própria rede externa utilizando o recurso de subredes para separação.

Controller host config (OVS)

under-the-hood-scenario-1-ovs-network.png
Retirada do OpenStack Guide

Compute host config (OVS)

under-the-hood-scenario-2-ovs-compute.png
Retirada do OpenStack Guide

Networking API

Networking oferece uma API para definir interface entre os serviços OpenStack (e.g.: vNICs de máquinas virtuais com Nova). Os recursos alocados possuem dois modos: controle e operação
Abstrações que descrevem os recursos:
- Network: segmento L2 isolado (análogo a VLAN)
- Subnet: faixa de endereços IP (v4 ou v6) e estado de configuração associado.
- Port: ponto de conexão para um dispositivo. Também descreve configurações associadas da rede (como MAC e endereço IP usado na porta em questão).

Modo de operação:
- Rede: List, get, add, update, remove
- Porta: List, get, add, update, remove
- Periférico: Get, plug, unplug

Agentes

Agentes são softwares que devem rodar em cada nó que lida com pacotes de dados.

- plug-in agent (neutron-*-agent): roda em cada hypervisor para configuração local do vswitch.
- dhcp agent (neutron-dhcp-agent): provê serviço DHCP para as redes dos tenants.
- l3 agent (neutron-l3-agent): provê acesso externo a VMs de dentro das redes dos tentants. Porém, alguns plugins tem sua própria implementação desse recurso.
- l3 metering agent (neutron-metering-agent): provê medidas de tráfico das redes dos tenants.

Neutron Manager

- Gerenciador do Nova para o Neutron. Redireciona requisições relacionadas.
- Oferece outros serviços de rede (gerenciamento de IP, DHCP, NAT, ...)

Camadas do Neutron

1. presentation: configuration managers, operational APIs, ...
2. virtual network application: L2 switch, firewall, load balancer, router, ...
3. controller: floodlight, cisco overdrive, nicira OVP, ...
4. management: VxLan, STT, OpenFlow, SNMP, nvGRE, GRE, VLANs, NetConf, ...
5. devices: Nexus 5k, UCS, Open vSwitch, OF-enabled switches, Linux Bridge, ...

Neutron Plugin

Neutron Plugin são pedaços de código que implementam chamadas da API Neutron.

- composto por um controle centralizado e data model.
- Implementa todas as operações inclusas na "Network API v2.0".
- Traduz as operações da "Network API v2.0" em chamadas de reconfiguração de switches reais e virtuais.
- Pode ser compatível com mais de um tipo de switch (um plugin pode ter múltiplos drivers).
- Define se as mensagens devem ser síncronas ou assíncronas
- implementa ou faz interface com qualquer tipo de SDN (OpenFlow, Orchestration, Tunneling, Network Interface, virtualization, ‘legacy’, ...)

- Apenas um plugin pode rodar de cada vez.
- não existe uma interface padrão para os drivers.
- "meta-plugin' que chama código de dois diferentes plugins. (proposto -- ver l2 modular plugin)
- plugins e drivers devem ser escritos como biblioteca para permitir o reuso.

Componentes:

Um plugin geralmente é formado por códigos para:

- armazenar informação sobre a configução lógica atual da rede (e.g., SQL database)
- determina e armazena informações sobre o 'mapeamento' entre modelo lógico e rede física (e.g. escolhendo uma VLAN para representar uma rede virtual)..
- comunica com um ou mais tipos de switches para configurá-los de acordo com o mapeamento. Isso pode ser na forma de um agente rodando no *hypervisor* ou de código que remotamente se conecta a um switch e o reconfigura.

Plugins disponíveis:

- Big Switch Plug-in
- Brocade Plug-in
- Cisco
- Cloudbase Hyper-V Plug-in
- Linux Bridge Plug-in
- Mellanox Plug-in
- Midonet Plug-in
- ML2 (Modular Layer 2) Plug-in
- NEC OpenFlow Plug-in
- Nicira NVP Plug-in
- Open vSwitch Plug-in
- PLUMgrid
- Ryu Plug-in

Observações

- Cria-se 'netork' e a associa auma 'sub-net'
- vif-plugging + switch
- vif-plugging deve ser mantido tão simples quanto o possível para evitar complicações do nova com detalhes da rede. (e.g., não se deve passar VLAN ID para o nova; deve-se apenas deixar que o nova crie o dispositivo básico e fazer com que o agente do plugin do neutron cuide de configurar a conexão para aquela vlan.
- data-forward node tipicamente tem uma interface de rede com um endereço IP para "management network" e outro para "data network".

Esquemas e relações

- nova-compute <> VIF driver <> AMPQ <> nova-network <> Neutron manager <> API client <> Neutron API <> Neutron plugin
- Neutron common:
  1. API e API Extension
  2. AuthN e AuthZ
  3. Backend
  4. Hypervisor(n)
     - VM (n) + eth(n) <> hypervisor's Network stack
     - Quantum plugin agent <> hypervisor's Network stack
- Neutron Plugin <> Neutron common
- Neutron Plugin <> Hypervisor(n) (quantum plugin agent)
- Neutron Plugin <> Network switch (quantum plugin agent)

Quantum Plugin BaseClass

- get_all_networks(self, tenant_id, **kwargs): returns a dictionary containing all <network_uuid, network_name> for the specified tenant.
- create_network(self, tenant_id, net_name, **kwargs): creates a new Virtual Network, and assigns it a symbolic name.
- delete_network(self, tenant_id, net_id): deletes the network with the specified network identifier belonging to the specified tenant.
- get_network_details(self, tenant_id, net_id): retrieves a list of all the remote vifs that are attached to the network.
- update_network(self, tenant_id, net_id, **kwargs): updates the attributes of a particular Virtual Network.
- get_all_ports(self, tenant_id, net_id, **kwargs): retrieves all port identifiers belonging to the specified Virtual Network.
- create_port(self, tenant_id, net_id, port_state=None, **kwargs): creates a port on the specified Virtual Network.
- update_port(self, tenant_id, net_id, port_id, **kwargs): updates the attributes of a specific port on the specified Virtual Network.
- delete_port(self, tenant_id, net_id, port_id): deletes a port on a specified Virtual Network, if the port contains a remote interface attachment, the remote interface is first un-plugged and then the port is deleted
- get_port_details(self, tenant_id, net_id, port_id): this method allows the user to retrieve a remote interface that is attached to this particular port
- plug_interface(self, tenant_id, net_id, port_id, remote_interface_id): attaches a remote interface to the specified port on the specified Virtual Network.
- unplug_interface(self, tenant_id, net_id, port_id): detaches a remote interface from the specified port on the specified Virtual Network.
- *subclasshook*(cls, klass): the *subclasshook* method is a class method that will be called (Already Implemented) everytime a class is tested using issubclass(klass, Plugin). In that case, it will check that every method marked with the abstractmethod decorator is provided by the plugin class

Dúvidas ou perguntas para orientação

- O que é e o que faz o Neutron Backend?
- Como é feita a autenticação?
- Quais são as funções da API?
- Como definir quais máquinas físicas fazem parte da núvem?

Definições

- AuthN: authentication
- AuthZ: authorization
- Backend: tecnologia que está por trás do plugin.
- Quantum API core: conjunto mínimo de operações
- driver: código que é capaz de conversar com um modelo particular ou uma família de switches.

Referências

Visitadas

- http://docs.openstack.org/api/openstack-network/2.0/
- http://www.slideshare.net/danwent/openstack-quantum-intro-os-meetup-32612
- OpenStack Quantum Networking update
- https://wiki.openstack.org/wiki/NeutronDevelopment
- http://docs.openstack.org/havana/config-reference/content/
- Networking plug-ins
- OpenStack Admin Guide - Networking
- OpenStack Networking API v2.0 Reference

Não visitadas

- http://www.ovirt.org/images/b/b3/Quantum_Ovirt_discussion-20111103.pdf
- http://events.csdn.net/OpenStack/Takashi Torii-OpenStack and OpenFlowSDN.pdf
- http://www.openstack.org/assets/presentation-media/HK-Openstack-Namespaces1-.pdf
- http://www.iaas.uni-stuttgart.de/lehre/vorlesung/aktuell/vorlesungen/smcc/materialien/SoftwareDefinedNetwork_20131024_v04.pdf
- http://www.cisco.com/en/US/prod/collateral/switches/ps9441/ps11541/data_sheet_c78-727737.pdf
- http://www.linux-kvm.org/wiki/images/7/7b/Kvm-forum-2013-openstack.pdf
- https://wiki.openstack.org/wiki/NeutronDevstack
- http://www.openstack.org/assets/presentation-media/HK-Openstack-Namespaces1-.pdf
- http://www.iaas.uni-stuttgart.de/lehre/vorlesung/aktuell/vorlesungen/smcc/materialien/SoftwareDefinedNetwork_20131024_v04.pdf
- http://www.cisco.com/en/US/prod/collateral/switches/ps9441/ps11541/data_sheet_c78-727737.pdf
- Quantum Plugins and Networking

Planejamento para o terceiro entregável

plan.png

Neutron e SDN (Em esboço ainda)

- Neutron:
  1. presentation: operational APIs
  2. virtual network application: L2 switch
  3. controller; Neutron pode fazer interface ou implementar essa camada.
  4. management: Neutron pode fazer interface ou implementar essa camada.
  5. devices: Neutron pode fazer interface ou implementar essa camada.

Neutron Server

quantum-server: processo principal do Netron. Expõe Netron API e encaminha as requisições de usuário para o plugin.

Agentes

Agentes enviam mensagens para o Quantum-Server via RPC (rabbitmq ou qpid) ou Netron API padrão (REST).

- plugin agent (quantum-*-agent): roda em cada hypervisor para realizar configuração local do vswitch. Dependente do plugin

escolhido.
- dhcp agent (quantum-dhcp-agent). serviço de DHCP para redes dos tentents. Usado quando as VMs devem receber IP de forma automatizada. Endereços via DHCP. Requer no mínimo dnsmasq 2.59. Configurado especificamente para o plugin utilizado.
- L3 agent (quantum-l3-agent). Encaminhamento L3/NAT para oferecer acessos as redes externas. Roteador que conecta as redes L2. Fica no Network Node. Usa: Linux IP Stack e IP tables.
  - Deve-se garantir que existam rotas entre o serviço de metadata do host e cada rede privada.
  - Nós que rodem quantum-l3-agent não devem ter endereços IP especificados manualmente para a NIC conectada a rede externa e sim uma faixa de IPs que poderá ser usada para roteamento.

```
$ sudo ovs-vsctrl add-br br-ext
$ sudo ovs-vsctrl add-port br-ext eth1
```

- não é possível executar ping para IPs fixos. Ao invés disso, usar: router network namespace (qrouter-<UUID>)
  ```
  $ ip netns exec qrouter-<UUID> ping <fixed ip>
  ```

Netron API

Serviço de redes virtual. Define conexões e endereços para os outros serviços.

- Network: segmento L2 isolado (análogo a uma VLAN).
- Subnet: bloco de endereços de IP (V4 ou V6) e estados de configuração associados.
- Port: ponto para conexção de dispositivos.

```
network <-- port
  |       |
   ---------- subnet
```

> *fixed_ips é atributo do port*
> *port.fixed_ips = {<IP1>:<subnet1>, <IP2>:<subnet2>, ...}*

Netron Plugins

Projeto original assume um modelo muito básico para realizar isolamento através de Linux VLANs e tabelas IP. Netron introduz o conceito de plugin, que é a
implementação de requisições lógicas de API em forma de um back-end acoplável. Padrão: Open vSwitch e Linux Bridge. Plugin deve estar em cada nó que usa
pacotes de dados.

Os plugins devem processar chamadas de API e gerenciar vSwitches.

OpenVSwitch: open vswitch, quantum-plugin-openvswitch-agent

```
$ sudo ovs-vsctrl add-br br-int
```

- br-int: integration bridge, nome padrão e fixo.

Netron Command Line Application

Requer: python-pyparsing, python-cliff, python-quantumclient

OpenStack Compute

Interagem com Netron através da API. Comunica com Netron API para connectar cada vNIC da VM em uma rede em particular.

Neutron Client

agent-*

dhcp-agent-*

ext-* (extensões)

floatingip-*

l3-agent-*

lb-* (loading balance)
  lb-pool-*
  lb-healthmonitor-*
  lb-member-*
  lb-vip-*

net-*

queue-*

router-*

security-*

subnet-*

quota-*

security-group-*

Exemplos - Netron API

```
$ quantum net-create net1
$ quantum net-create net2 --provider:network_type local
$ quantum net-create --tenant-id=<id> net1

$ quantum subnet-create net1 192.168.0.2/22 --name subnet1

$ quantum port-create net1 --fixed-ip ip_addr=192.168.2.40
$ quantum port-create net1
$ quantum port-list --fixed-ips
$ quantum port-list --tenant-id=<tenant id>
$ quantum port-list --device-id=<vm id>
```

- as redes já devem estar criadas para serem adicionadas na instanciação das VMs

```
$ nova boot --image=<img id> --flavor=<fl id> --nic net_id=<net id> <vm name>
```

- para ip fixo (port deve ser criado antes da instanciação)

```
$ nova boot --image=<img id> --flavor=<fl id> --nic port_id=<net id> <vm name>
```

Físico -> Virtual

1) eth0 (VM)
2) TAP VNET0 (hypervisor network interface card, VIF, vNIC)
3) LINUX BRIDGE qbrXXX (hub, multi NIC, virtual ou físico)
4) veth pair {qcbXXX, qvoXXX} (entra de um lado e sai do outro)

5) open switch bridge br-int (virtual switch, integration bridge,todos as VM estão conectadas a ele)
6) veth pair (int-br-eth1, phy-br-eth1)
7) physical network interface card eth1

Ryu

```
$ sudo pip install ryu
$ sudo apt-get install quantum-plugin-ryu-agent openvswitch-switch python-openvswitch openvswitch-datapath-dkms
```

- ovs_ryu_plugin.ini e quantum.conf devem ser copiados em cada nó que tenha agente ryu

Floodlight

Floodlight pode ser utilizado como o backend para o OpenStack Network utilizando um Plug-in para Neutron. O Neutron expõe um modelo de networking-as-a-service via API REST implementada no Floodlight. Dois componentes principais são utilizados para essa integração: um aplicativo Virtual Switch do Floodlight que implementa a Neutron API e o o RestProxy plugin que conecta o Neutron com o Floodlight.

Conexão entre FloodLight e Neutron em dois passos:

1. Rodar o REST Proxy Plug-in no neutron-server. Esse plugin foi projetado para rodar como parte do Neutro service.
2. Habilitar o Virtual Switch no Floodlight (antigo Virtual Network Filter). Esse é o módulo que implementa um isolamento de nível 2 (baseado em MAC) em redes openflow. Esse módulo vem com o Floodlight por padrão e não depende do Neutron ou do Openstack para ser executado.

Not to do

- Proxies ainda não implementados então não se deve executar:
  - nova-manage
  - nova para administrar recursos de rede

Files

Neutron-PhysNet-Diagram.png	76 KB	22 Jan 2014	Thiago Paiva
under-the-hood-scenario-2-ovs-compute.png	199 KB	22 Jan 2014	Thiago Paiva
under-the-hood-scenario-1-ovs-network.png	162 KB	22 Jan 2014	Thiago Paiva
plan.png	7.12 KB	23 Jan 2014	Thiago Paiva