Desjardins

Open vSwitch

Janvier 2014

Sylvain Lavoie
sylvain.lavoie@desjardins.com

Desjardins

COOPÉRER
POUR CRÉER L'AVENIR

Agenda

- Historique du projet
- Open vSwitch
- Les supporteurs
- Les fonctionnalités
- Son adoption dans le marché
- Architecture, les couches
- Démo

Historique du projet

- « Nicira Network Inc » fondé en 2007 (nicira.com) est le créateur des projets OpenFlow et Open vSwitch.

- Mai 2010 : Première version d'Open vSwitch qui est relâchée.

- Juillet 2012 : VMware achète « Nicira Network Inc » pour 1,2 milliard de dollars.
  - VMware s'engage à continuer le support de la version « Open Source » de la technologie qui ont permis aux solutions Nicira d'être un succès, incluant la solution « Open vSwitch » Réf. : http://www.vmware.com/ca/fr/company/news/releases/vmw-nicira-07-23-12.html#sthash.QCNQatfP.dpuf

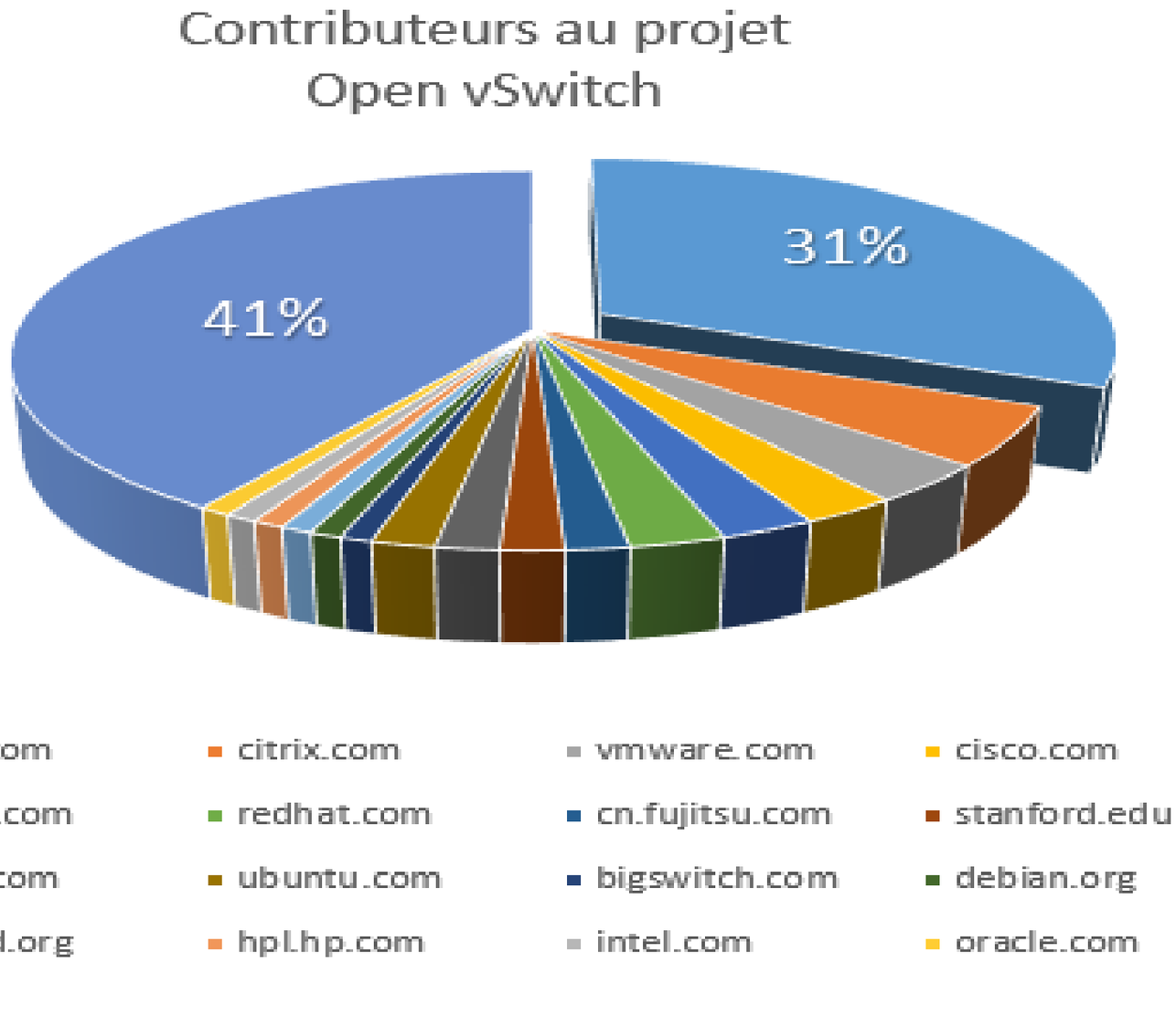
- Juin 2013 : VMware finalise son offre SDN « Software Defined Network » basée sur la technologie de « Nicira Network » pour « vCloud Hybrid Service ».
  - VMware s'oriente vers le marché du réseau virtuel pour les futures capacités réseau offertes par la firme.
  - Leurs solutions sont basées sur la technologie acquise avec l'achat de Nicira.
  - La vision de VMware est « tout ce que l'on peut faire, ou presque dans un réseau physique avec des boitiers et du Matériel, VMware peut le réaliser avec un logiciel contrôlé par des APIs ».

Open vSwitch

- « Open vSwitch » est une implémentation logicielle d'une switch ethernet. Concrètement il est constitué d'un service (ovs-vswitchd) et d'un module kernel (openvswitch_mod). Le service permet de commuter effectivement les paquets vers les bons ports virtuels, alors que le module kernel permet de capturer le trafic provenant des interfaces réseau, et d'y réinjecter le trafic légitime.

- La différence entre VMware « vNetwork distributed switch » et Cisco « Nexus 1000v », ce sont des vSwitch dont la gestion, la configuration et la surveillance sont effectuées avec une console centrale. « Open vswitch » n'est pas une vSwitch distribuée. La gestion est locale sur la machine physique, mais supporte la gestion à distance facilitant la tâche aux développeurs de plateforme de gestion de virtualisation/Cloud.

Les supporteurs



- En date de décembre 2013, plus de 150 personnes actives supportent son développement.
- De plus, il y a plus de 120 personnes additionnelles qui aident à sa réalisation.

Desjardins

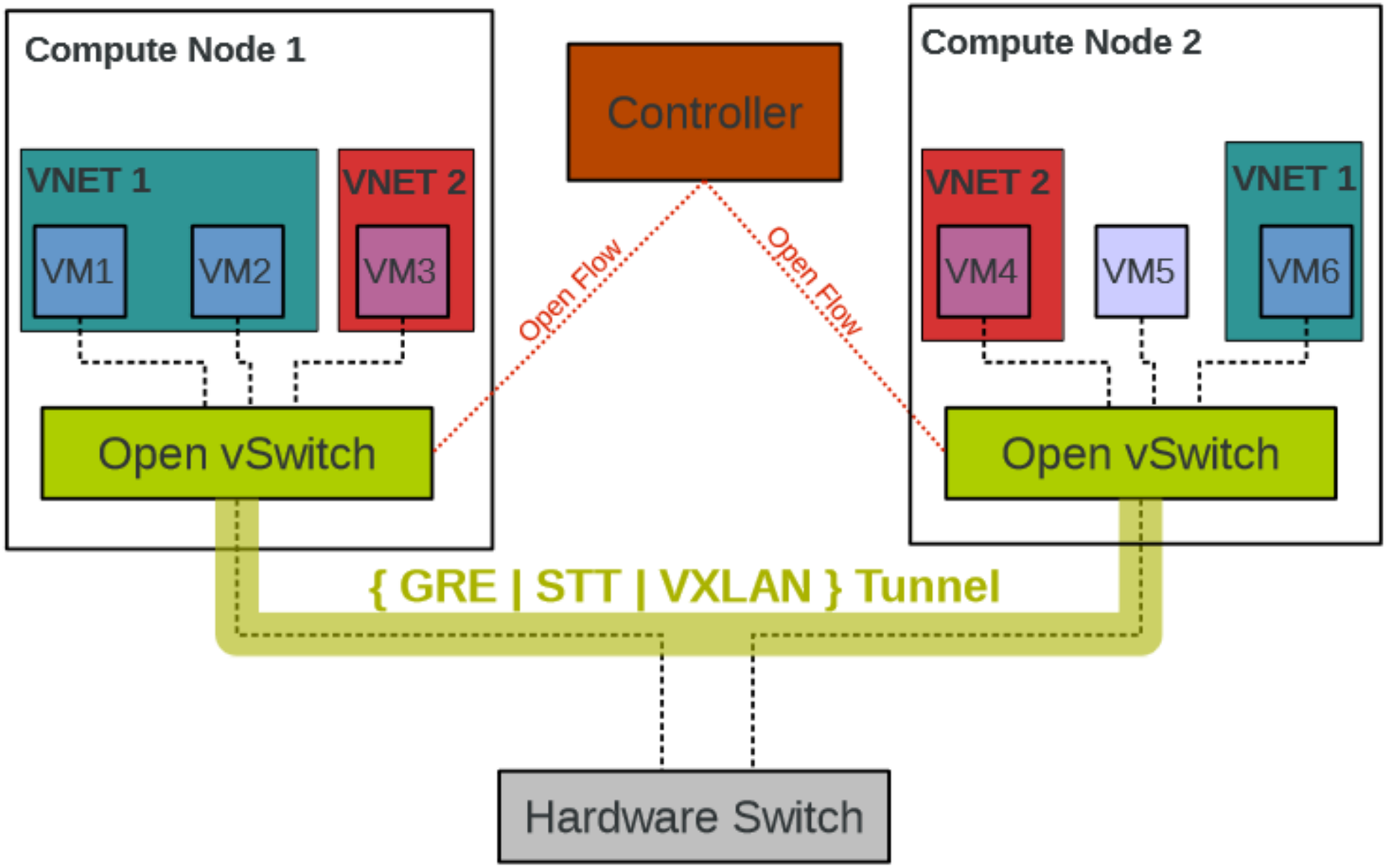
COOPÉRER
POUR CRÉER L'AVENIR

Les fonctionnalités

- « Open vSwitch » supporte beaucoup de fonctionnalités d'une switch L2 et même d'une switch L3. Voici la liste :
  - Visibility into inter-VM communication via NetFlow, sFlow(R), IPFIX, SPAN, RSPAN, and GRE-tunneled mirrors
  - LACP (IEEE 802.1AX-2008)
  - Standard 802.1Q VLAN model with trunking
  - BFD and 802.1ag link monitoring
  - STP (IEEE 802.1D-1998)
  - Fine-grained QoS control
  - Support for HFSC qdisc
  - Per VM interface traffic policing
  - NIC bonding with source-MAC load balancing, active backup, and L4 hashing
  - OpenFlow protocol support (including many extensions for virtualization)
  - IPv6 support
  - Multiple tunneling protocols (GRE, VXLAN, IPsec, GRE and VXLAN over IPsec)
  - Remote configuration protocol with C and Python bindings
  - Kernel and user-space forwarding engine options
  - Multi-table forwarding pipeline with flow-caching engine
  - Forwarding layer abstraction to ease porting to new software and hardware platforms

Les fonctionnalités
La fonction « tunneling »

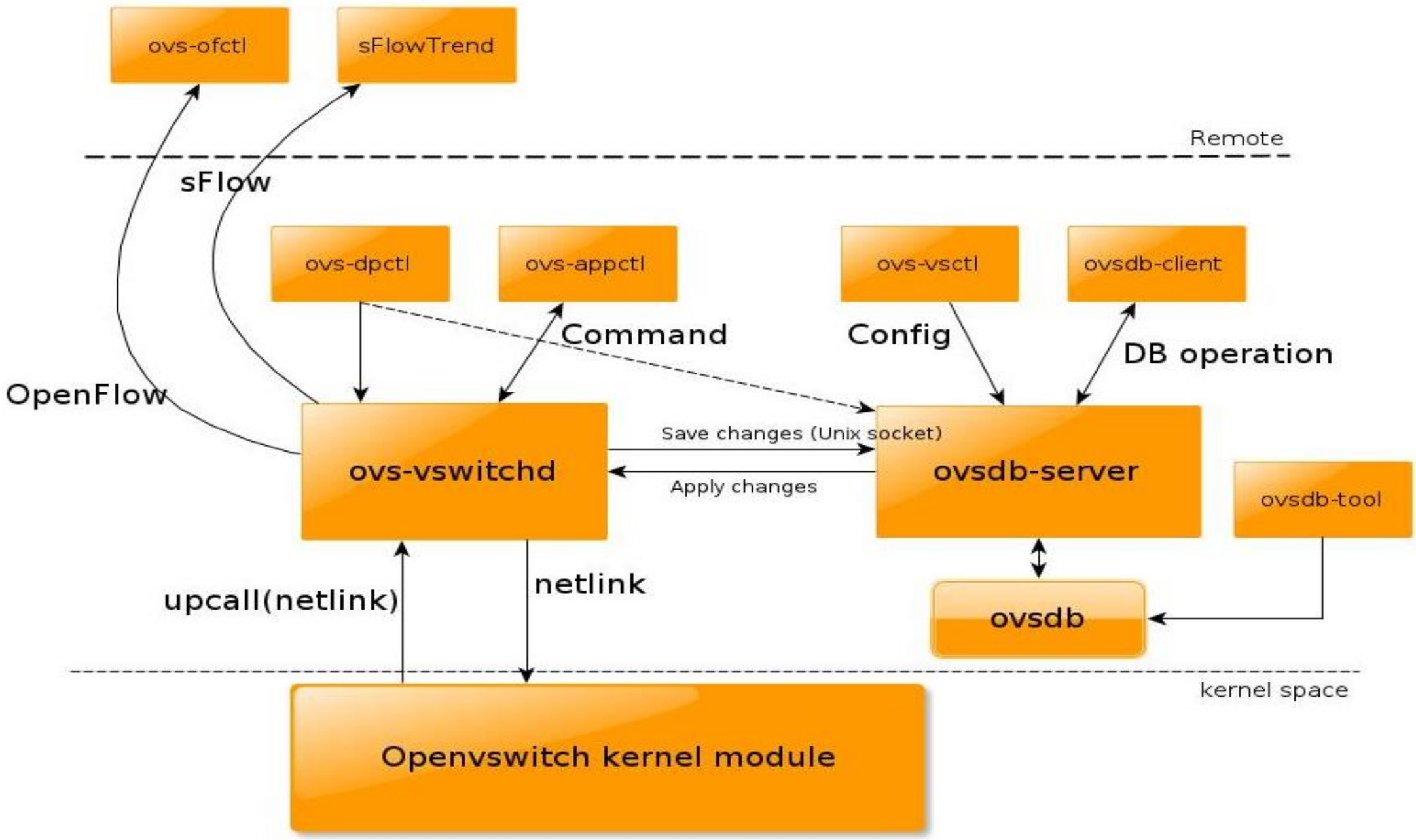
- Le « tunnel » fournit une isolation et réduit la dépendance des composants réseau physiques.



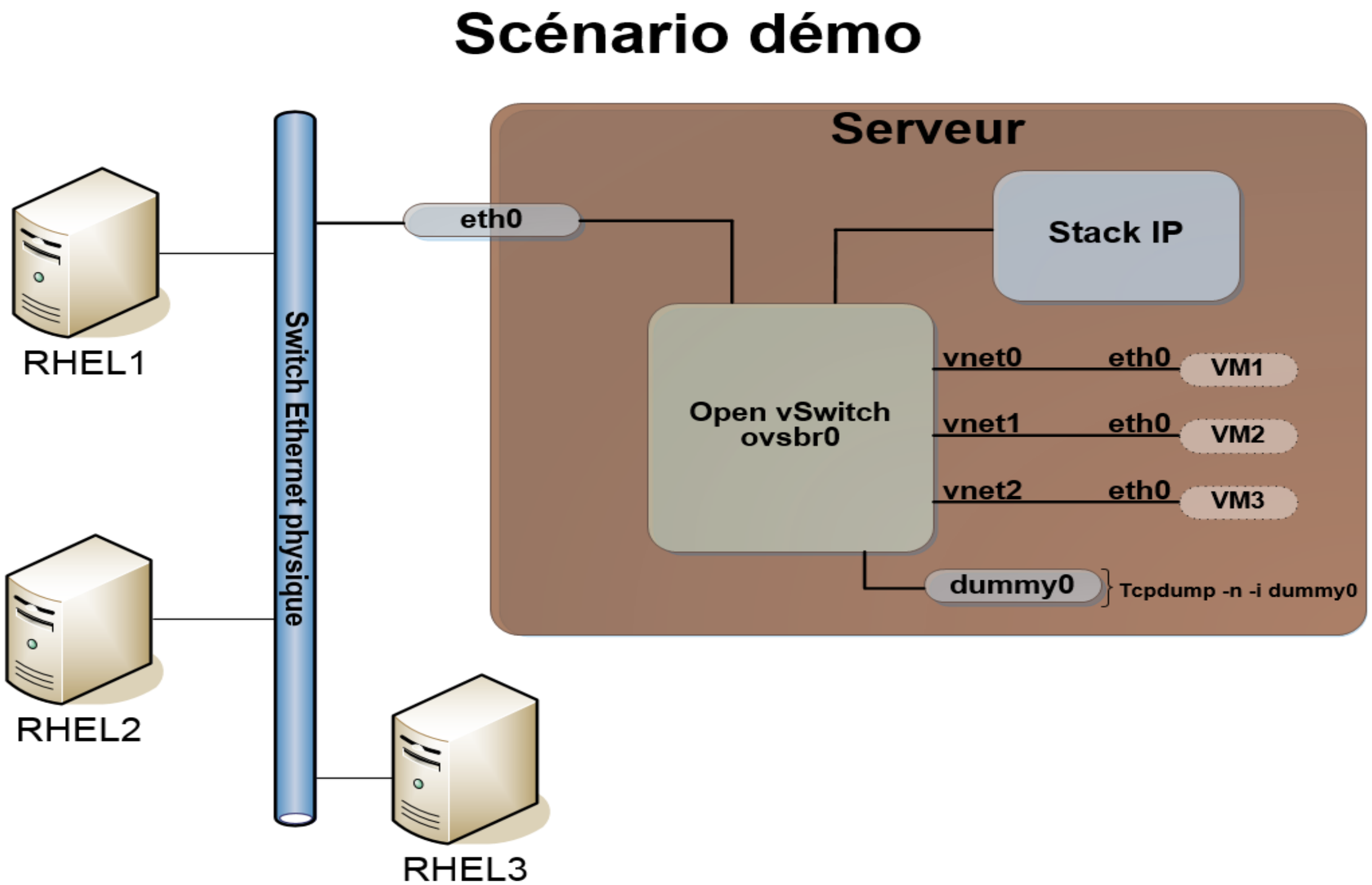
Son adoption dans le marché

- Plusieurs projets et/ou produits reposent sur la solution d’ « Open vSwitch ».
- Écrit en C, il est portable et en ce moment, il est porté sur les plateformes FreeBSD et Windows.
- Parmi ces consommateurs, nous y retrouvons en autre :
  - KVM, Xen Cloud Platform, Proxmox VE, VirtualBox
- Son module kernel « datapath » est maintenant intégré à Linux depuis 3.3. Les packages « linux based virtualization platform » sont disponibles pour les distributions comme Ubuntu, Debian, and Fedora.
- Il a également été intégré dans plusieurs solutions de gestion de virtualisation / nuage informatique comme :
  - OpenStack, openQRM, OpenNebula et oVirt

Architecture, les couches



Démo


Scénario démo
Serveur
eth0
Stack IP
Open vSwitch
ovsbr0
vnet0
eth0
VM1
vnet1
eth0
VM2
vnet2
eth0
VM3
dummy0
Tcpdump -n -i dummy0
RHEL1
Switch Ethernet physique
RHEL2
RHEL3

Démo - Hypervisor
1) Installation sur le Host

- Installation d'Open vSwitch sur Fedora 20 :

```
yum install openvswitch -y
```

- Activation des services :

```
systemctl enable openvswitch.service
systemctl start openvswitch.service
```

- Validation du service :

```
systemctl status openvswitch.service
```

Démo - Hypervisor
2) Configuration

- Désactiver le « NetworkManager » :

```
systemctl disable NetworkManager.service
```

- Activation du gestionnaire classique du réseau :

```
systemctl enable network.service
```

- Redémarrage du réseau :

```
systemctl restart network.service
```

- Valider l'état des services réseau :

```
systemctl is-enabled network.service
systemctl is-enabled NetworkManager.service
```

Démo - Hypervisor
3) Création d'un « bridge »

- Création d'un « bridge » ou d'une « switch » :

ovs-vsctl add-br ovsbr0 **!! Pour détruire del-br**

- Pour visualiser la configuration :

ovs-vsctl show

```
[root@ovshyper ~]# ovs-vsctl show
0e7a41bb-ee88-4d19-bc8a-83ff21e82067
    Bridge "ovsbr0"
        Port "ovsbr0"
            Interface "ovsbr0"
                type: internal
    ovs_version: "2.0.0"
```

- Affichage des interfaces réseau :

ifconfig

```
[root@ovshyper ~]# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.61  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::20c:29ff:fe56:cbf6  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:56:cb:f6  txqueuelen 1000  (Ethernet)
        RX packets 98  bytes 12235 (11.9 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 71  bytes 8233 (8.0 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 0  (Boucle locale)
        RX packets 36  bytes 15104 (14.7 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 36  bytes 15104 (14.7 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ovsbr0: flags=67<UP,BROADCAST,RUNNING>  mtu 1500
        inet6 fe80::d6:20ff:fe4c:7d35  prefixlen 64  scopeid 0x20<link>
        ether 66:6c:77:a2:f9:40  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8  bytes 648 (648.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Démo - Hypervisor
4) Assignation du port « eth0 »

- Affectation de l'interface « eth0 » à la « switch » :

ovs-vsctl add-port ovsbr0 eth0	**!! Pour détruire del-port**

- Pour voir la création :

ovs-vsctl show

```
[root@ovshyper cfgovs]# ifconfig eth0 0
[root@ovshyper cfgovs]# dhclient ovsbr0
[root@ovshyper cfgovs]# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::20c:29ff:fe56:cbf6  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:56:cb:f6  txqueuelen 1000  (Ethernet)
        RX packets 11714  bytes 2956851 (2.8 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 184  bytes 19676 (19.2 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 0  (Boucle locale)
        RX packets 199  bytes 69673 (68.0 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 199  bytes 69673 (68.0 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ovsbr0: flags=67<UP,BROADCAST,RUNNING>  mtu 1500
        inet 192.168.1.180  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::d6:20ff:fe4c:7d35  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:56:cb:f6  txqueuelen 0  (Ethernet)
        RX packets 11351  bytes 2913925 (2.7 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 15  bytes 1674 (1.6 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

```
[root@ovshyper ~]# ovs-vsctl add-port ovsbr0 eth0
[root@ovshyper ~]# ovs-vsctl show
0e7a41bb-ee88-4d19-bc8a-83ff21e82067
    Bridge "ovsbr0"
        Port "eth0"
            Interface "eth0"
        Port "ovsbr0"
            Interface "ovsbr0"
                type: internal
    ovs_version: "2.0.0"
[root@ovshyper ~]#
```

COOPÉRER
POUR CRÉER L'AVENIR

Démo - Hypervisor
5) Configuration ifcfg-ovsbr0

- Configuration de « ifcfg-ovsbr0 » sous /etc/sysconfig/network-scripts :

```
[root@ovshyper network-scripts]# cat ~/cfgovs/ifcfg-ovsbr0
DEVICE="ovsbr0"
NM_CONTROLLED="no"
BOOTPROTO="none"
IPADDR="192.168.1.61"
NETMASK="255.255.255.0"
DOMAIN=infosylvoie.inet
BROADCAST="192.168.1.255"
DNS1="192.168.1.224"
DNS2="192.168.1.1"
PEERDNS="yes"
GATEWAY="192.168.1.1"
DEFROUTE="yes"
IPV4_FAILURE_FATAL="yes"
IPV6INIT=no
ONBOOT="yes"
TYPE="OVSBridge"
DEVICETYPE="ovs"
HOTPLUG=no
```

Démo - Hypervisor
6) Configuration ifcfg-eth0

- Configuration de « ifcfg-eth0 » sous /etc/sysconfig/network-scripts :

```
[root@ovshyper network-scripts]# cat ~/cfgovs/ifcfg-eth0
DEVICE="eth0"
NM_CONTROLLED="no"
ONBOOT="yes"
IPV6INIT=no
TYPE="OVSPort"
DEVICETYPE="ovs"
OVS_BRIDGE=ovsbr0
BOOTPROTO=none
HOTPLUG=no
[root@ovshyper network-scripts]#
```

Démo – QEMU/KVM
7) Configuration VM1

- Paramètres essentiels :

virsh edit vm1

```
<interface type='bridge'>
  <mac address='52:54:00:d4:68:e4'/>
  <source bridge='ovsbr0'/>
  <virtualport type='openvswitch'>
    <parameters interfaceid='7b10a428-e7c9-45dd-88eb-e1a080c41883'/>
  </virtualport>
  <model type='virtio'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0'/>
</interface>
```

Démo – QEMU/KVM
8) Interface réseau pour la VM1

- Qemu/kvm va automatiquement créer l'interface réseau :

ifconfig vnet0

```
[root@ovshyper ~]# ifconfig vnet0
vnet0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::fc54:ff:fed4:68e4  prefixlen 64  scopeid 0x20<link>
        ether fe:54:00:d4:68:e4  txqueuelen 500  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 138 overruns 0  carrier 0  collisions 0
```

- Lister la configuration d'Open vSwitch :

```
[root@ovshyper ~]# ovs-vsctl show
0e7a41bb-ee88-4d19-bc8a-83ff21e82067
    Bridge "ovsbr0"
        Port "eth0"
            Interface "eth0"
        Port "ovsbr0"
            Interface "ovsbr0"
                type: internal
        Port "vnet0"
            Interface "vnet0"
    ovs_version: "2.0.0"
```

Démo – QEMU/KVM
9) Interface réseau pour la VM1

- Afficher la « Forwarding Table » dans OVS :

ovs-appctl fdb/show ovsbr0

```
[root@ovshyper ~]# ovs-appctl fdb/show ovsbr0
 port  VLAN  MAC                Age
LOCAL     0  00:0c:29:56:cb:f6  230
    1     0  00:00:00:00:00:00   90
    1     0  00:1c:23:d6:18:b4   32
    1     0  00:0c:29:0f:48:7f   27
    1     0  00:0c:29:38:48:35   27
    1     0  00:0c:29:99:7e:a7   27
    1     0  7c:d1:c3:00:d0:24   27
    1     0  00:0c:29:3e:3c:bb   27
    1     0  00:0c:29:3d:e0:f4   27
    1     0  00:21:79:e2:ac:e7   25
    1     0  b8:e8:56:97:fa:eb   19
    1     0  00:08:9b:c9:d4:48    4
    1     0  00:08:9b:c9:d4:49    2
    1     0  00:0c:29:c3:65:2f    2
    3     0  52:54:00:d4:68:e4    2
    1     0  00:0c:29:ea:b9:8b    1
    1     0  00:1c:c4:4f:c3:63    1
    1     0  00:0c:29:13:1f:9a    1
    1     0  c0:c1:c0:42:7e:87    0
[root@ovshyper ~]# ifconfig vnet0
vnet0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::fc54:ff:fed4:68e4  prefixlen 64  scopeid 0x20<link>
        ether fe:54:00:d4:68:e4  txqueuelen 500  (Ethernet)
        RX packets 10  bytes 1236 (1.2 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1016  bytes 369082 (360.4 KiB)
        TX errors 0  dropped 28 overruns 0  carrier 0  collisions 0
```

Démo – QEMU/KVM
10) Interface réseau pour la VM1

- Qemu/kvm va automatiquement créer l'interface réseau :

ifconfig vnet0

```
[root@ovshyper ~]# virsh console vm1
Connected to domain vm1
Escape character is ^]

CentOS release 6.5 (Final)
Kernel 2.6.32-431.1.2.0.1.el6.x86_64 on an x86_64

vm1.infosylvoie.inet login: root
Password:
Last login: Sun Jan 12 16:40:20 on ttyS0
[root@vm1 ~]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 52:54:00:D4:68:E4
          inet adr:192.168.1.70  Bcast:192.168.1.255  Masque:255.255.255.0
          adr inet6: fe80::5054:ff:fed4:68e4/64 Scope:Lien
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1269 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 lg file transmission:1000
          RX bytes:463576 (452.7 KiB)  TX bytes:824 (824.0 b)

[root@vm1 ~]#
```

Démo - Port Mirroring
11) Création interface réseau « dummy0 »

- Chargement du driver réseau Dummy :

```
modprobe dummy
```

- Activer l'interface réseau dummy :

```
ip link set up dummy0
```

- Valider la présence de l'interface réseau :

```
[root@ovshyper ~]# lsmod | grep dummy
[root@ovshyper ~]# modprobe dummy
[root@ovshyper ~]# lsmod | grep dummy
dummy                  12960  0
[root@ovshyper ~]# ip link set up dummy0
[root@ovshyper ~]# ifconfig dummy0
dummy0: flags=195<UP,BROADCAST,RUNNING,NOARP>  mtu 1500
        inet6 fe80::d486:2bff:fe6d:75d2  prefixlen 64  scopeid 0x20<link>
        ether d6:86:2b:6d:75:d2  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 3  bytes 210 (210.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

[root@ovshyper ~]#
```

Démo - Port Mirroring
12) Ajout d'un interface réseau dans ovs

- Ajout d'un port « dummy0 » dans la vSwitch:

ovs-vsctl add-port ovsbr0 dummy0

- Afficher la configuration de la vSwitch :

ovs-vsctl show

```
[root@ovshyper mirror]# ovs-vsctl add-port ovsbr0 dummy0
[root@ovshyper mirror]# ovs-vsctl show
0e7a41bb-ee88-4d19-bc8a-83ff21e82067
    Bridge "ovsbr0"
        Port "eth0"
            Interface "eth0"
        Port "ovsbr0"
            Interface "ovsbr0"
                type: internal
        Port "dummy0"
            Interface "dummy0"
    ovs_version: "2.0.0"
[root@ovshyper mirror]#
```

Démo – Port Mirroring
13) Configuration OVS

- Configuration du miroir de port dans la vSwitch:

```
ovs-vsctl \
-- --id=@p get port dummy0 \
-- --id=@m create mirror name=mirror0 \
-- add bridge ovsbr0 mirror @m \
-- set mirror mirror0 output_port=@p
```

```
[root@ovshyper mirror]# ./2.1_ovs_create_mirror.sh
+ ovs-vsctl -- --id=@p get port dummy0 -- --id=@m create mirror name=mirror0 --
add bridge ovsbr0 mirror @m -- set mirror mirror0 output_port=@p
a3f6f95e-99f7-4a97-9f12-c9e6e519d664
[root@ovshyper mirror]#
```

- Afficher le détail de la vSwitch :

ovs-vsctl list bridge ovsbr0

```
[root@ovshyper mirror]# ovs-vsctl list bridge ovsbr0
_uuid               : 46b29801-25b8-4a1e-8272-702f3fffbe3d
controller          : []
datapath_id         : "0000000c2956cbf6"
datapath_type       : ""
external_ids        : {}
fail_mode           : []
flood_vlans         : []
flow_tables         : {}
ipfix               : []
mirrors             : [a3f6f95e-99f7-4a97-9f12-c9e6e519d664]
name                : "ovsbr0"
```

Références

- Open vSwitch
- Introduction Open vSwitch
- Bon exemple en français
- Achat Nicira par VMware

Questions