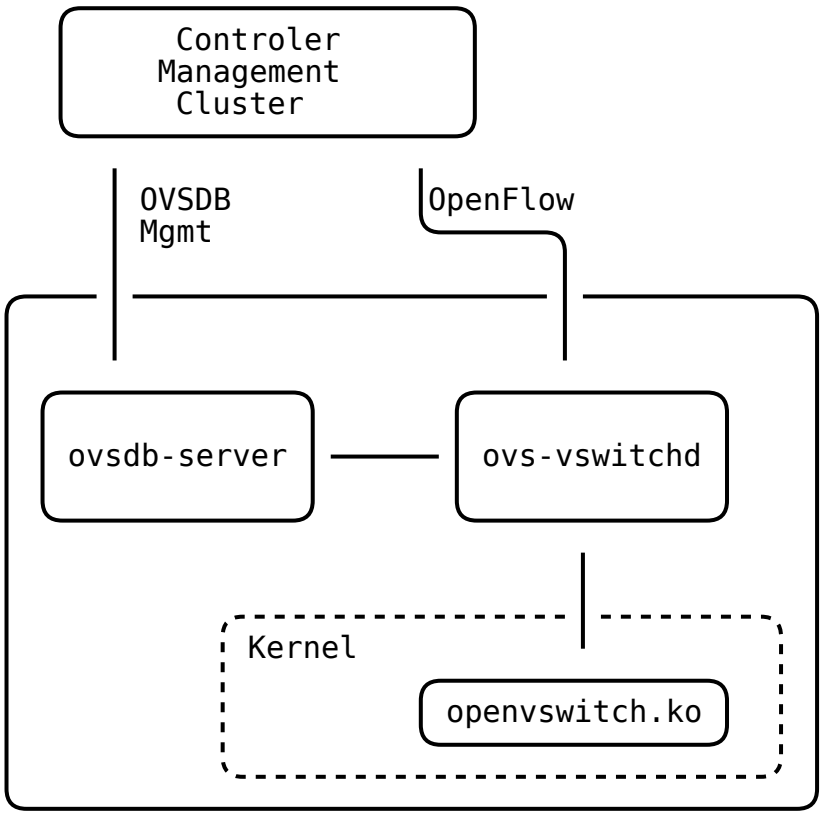
OVS: Lab 1 SDN sur un réseau virtuel

Objet	contrôleur SDN avec Openvswitch
Niveau requis	débutant, avisé
Débutant, à savoir	ovs-openflow-overview
Suivi	:DRAFT:

OpenVSwitch est un commutateur logiciel OpenFlow pouvant être contrôlé par un contrôleur.



Connexion OVSDB et Openflow

OVSDB est un protocole de gestion de base de données OVS qui définit le schéma de la base de données OVS et les spécifications pour la communication entre le contrôleur et le commutateur.

OPENFLOW est un protocole utilisé par le contrôleur SDN pour contrôler le commutateur OVS en renseignant sa base de données avec la configuration prévue.

Il est courant de le trouver installé sur un serveur distant (c'est pourquoi il est placé dans un “espace” différent dans le dessin) bien qu'il puisse être installé sur le même serveur.

Configuration du contrôleur

Comme le contrôleur doit gérer des commutateurs OVS auxquels des connexions OVSDB et Openflow seront établies, ont va mettre les connexions Openflow et OVSDB à l'écoute sur les ports 6653 et

6632, respectivement.

Interface de gestion OVSDB

L'interface de gestion OVSDB est utilisée pour effectuer la gestion des opérations de configuration sur l'instance OVS. Ilets pararamétré par l'utilisation de la commande `ovs-vsctl set-manager ptcp:<port tcp>:<ip ovsdb>`

Par exemple la commande suivante démarre met le serveur OVSDB à l'écoute sur le port TCP 6632 pour un openflow13:

```
$ ovs-vsctl set Bridge s1 protocols=OpenFlow13
$ ovs-vsctl set-manager ptcp:6640
```

Note: Le port 6640 est réservé pour OVSDB sur IANA.

Gestionnaire SDN Openflow

Comme OVS ne propose pas de contrôleur ou de gestionnaire SDN natif, on utilise une application ryu pour démarrer un contrôleur openflow.

```
$ ryu-manager ryu.app.simple_switch_13
```

```
loading app ryu.app.simple_switch_13
loading app ryu.controller.ofp_handler
instantiating app ryu.app.simple_switch_13 of SimpleSwitch13
instantiating app ryu.controller.ofp_handler of OFPHandler
```

On peut vérifier que les ports sont bien à l'écoute en lançant les commandes suivantes:

```
netstat -ntlpu
```

```
tcp        0      0 0.0.0.0:6653            0.0.0.0:*               LISTEN
28499/python3.6
tcp        0      0 0.0.0.0:3142            0.0.0.0:*               LISTEN
9787/apt-cacher-ng
tcp        0      0 0.0.0.0:6640            0.0.0.0:*               LISTEN
26902/ovsdb-server
```

Le commutateur OVS doit déjà être en cours d'exécution avant d'exécuter une commande OVS. Pour exécuter OVS utiliser la commande suivante avec les autorisations root:

```
service openvswitch start
```

Il également possible de vérifier son statut.

```
$ service openvswitch status

Redirecting to /bin/systemctl status  openvswitch.service
 openvswitch.service - Open vSwitch
   Loaded: loaded (/usr/lib/systemd/system/openvswitch.service; disabled;
vendor preset: disabled)
   Active: active (exited) since ven. 2019-04-19 23:56:32 CEST; 3 days ago
 Main PID: 22380 (code=exited, status=0/SUCCESS)
   CGroup: /system.slice/openvswitch.service

avril 19 23:56:32 master2016 systemd[1]: Starting Open vSwitch...
avril 19 23:56:32 master2016 systemd[1]: Started Open vSwitch.
```

Une fois qu'OVS est en cours d'exécution, on peut également exécuter ses commandes CLI. Par exemple,

```
$ ovs-vsctl show
f379007e-48a8-4fc7-a7d2-172725caef7a
    Manager "ptcp:6632"
    Bridge "ovs-br0"
        Controller "tcp:192.168.122.4:6633"
            is_connected: true
        Port "ovs-br0"
            Interface "ovs-br0"
                type: internal
        Port "virbr0"
            Interface "virbr0"
    ovs_version: "2.5.0"
```

Etablissement des connexions côté commutateur

Du côté commutateur OVS, la commande ci-dessous est exécutée pour établir une connexion OVSDB avec le contrôleur:

```
$ ovs-vsctl set-manager tcp:<controller-IP>:<port>
```

Pour confirmer que la connexion est établie, exécuter la commande ci-dessous qui indique que l'indicateur `is_connected` est défini sur `true`

```
$ ovs-vsctl show
```

```
f379007e-48a8-4fc7-a7d2-172725caef7a
    Manager "tcp:192.168.122.4:6640"
    Bridge "ovs-br0"
        Controller "tcp:192.168.122.4:6653"
            is_connected: true
        Port "ovs-br0"
            Interface "ovs-br0"
                type: internal
        Port "virbr0"
            Interface "virbr0"
    ovs_version: "2.5.0"
```

La connexion Openflow est établie sur un pont. On peut donc créer un pont sur OVS. On peut également créer un pont sur OVS en envoyant la configuration à OVS via une connexion OVSDB. Sur le pont dans OVS, la commande ci-dessous peut être exécutée pour connecter un pont à OVS.

```
ovs-vsctl set-controller <bridge name> tcp:<controller-IP>:6653
```

Des connexions OVSDB et Openflow ont été établies et peuvent être vérifiées en exécutant les commandes suivantes qui doivent confirmer l'état de connexion ESTABLISHED:

```
$ netstat -a | grep 6653
tcp        0      0 0.0.0.0:6653            0.0.0.0:*               LISTEN
tcp        0      0 192.168.122.4:50068     192.168.122.4:6653
ESTABLISHED
tcp        0      0 192.168.122.4:6653      192.168.122.4:50068
ESTABLISHED
```

```
$ netstat -a | grep 6640
tcp        0      0 0.0.0.0:6640            0.0.0.0:*               LISTEN
```

Les messages échangés entre le contrôleur et OVS peuvent être facilement capturés via le logiciel Wireshark. Lors de l'établissement d'une connexion, le contrôleur envoi une requête Openflow FEATURES_REQUEST et en réponse, reçoit FEATURES_REPLY d'OVS. Dans les messages FEATURES_REPLY, le contrôleur obtient l'ID de chemin de données (identifiant unique) du pont OVS qui se charge de la transmission (sur la base des règles Openflow configurées par OFL) du trafic entre les machines virtuelles connectées aux ports d'OVS.

Du côté OVS, la commande suivante peut être exécutée pour afficher les détails du pont.

```
ovs-ofctl show ovs-br0 -OOpenFlow13
```

```
OFPT_FEATURES_REPLY (OF1.3) (xid=0x2): dpid:0000325ac4b11e42
```

```
n_tables:254, n_buffers:256
capabilities: FLOW_STATS TABLE_STATS PORT_STATS GROUP_STATS QUEUE_STATS
OFPST_PORT_DESC reply (OF1.3) (xid=0x3):
 1(virbr0): addr:52:54:00:68:ec:b5
     config:     0
     state:      LINK_DOWN
     speed: 0 Mbps now, 0 Mbps max
 LOCAL(ovs-br0): addr:32:5a:c4:b1:1e:42
     config:     0
     state:      0
     speed: 0 Mbps now, 0 Mbps max
OFPT_GET_CONFIG_REPLY (OF1.3) (xid=0x5): frags=normal miss_send_len=0
```

Ici, on peut voir que FEATURE_REPLY affiche l'ID de chemin de données (au format hexadécimal) du pont sur OVS.

Suppression du contrôleur

Pour supprimer la connexion Openflow avec ODL, la commande ci-dessous peut être exécutée sur OVS:

```
$ ovs-vsctl del-controller <bridge name>
```

Pour supprimer la connexion OVSDB avec ODL, la commande ci-dessous peut être exécutée sur OVS:

```
$ ovs-vsctl del-manager
```

Une fois la connexion supprimée, l'indicateur `is_conncted` qui était `true` lors de l'établissement de la connexion disparaîtra de la sortie de la commande CLI `ovs-vsctl show` d'OVS.

Pour supprimer définitivement ovs-vswitchd Manager , procéder comme suit:

```
$ ovs-vsctl emer-reset
```

La commande **emer-reset** réinitialise la configuration dans un état propre, déconfigure les contrôleurs OpenFlow, les serveurs OVSDB et SSL, et supprime la mise en miroir des ports, fail_mode, configuration NetFlow, sFlow et IPFIX.

Attention: Cette commande supprime également toutes les autres clés de configuration de toutes les données,enregistrements de base, sauf `other-config: hwaddr` qui est préservé s'il est présent dans un enregistrement de Bridge. Les autres configurations de réseau sont laissées tel quel.

Débogage et dépannage

Lorsque OVS est en cours d'exécution, il génère un fichier journal OVSDB nommé 'ovsdb-server.log' et

un fichier journal Openflow nommé 'ovs-vswitchd.log' dans le chemin:

```
/var/log/openvswitch
```