Networking

Openstack-related topics - EVS

- There are two management components to EVS, which are configured on the physical host(s):-
  - **EVS Manager** - the administrative function that is used to configure and manage the EVS Controller.
  - **EVS Controller** - controls the configuration and management of the virtual switches.
- Client machines are called *EVS nodes*, and these are the zones whose VNICs are connected to the EVS.
- On the EVS nodes you would use the *zonecfg* and *dladm* commands to create EVS connections.
- On all the above systems, physical, logical domains and zones, you need to install the *evs* package:-

pkg install evs

- The EVS Manager and EVS Controller can be configured on separate systems, but in our example we will use just a single system to host both functions.
- Note that all communication between the various EVS nodes is carried out over secure SSH connections, and the configuration of these facilities will be explained.
- Ultimately, an EVS should ease an administrator's task of setting up secure and isolated networks across physical nodes in a datacentre.

Networking

EVS manager and Controller

- In the following example, we will use the following hosts:-
  - *squid* - this will be the EVS Manager and Controller.
  - *lotus* and *hoki* - Solaris 11 systems hosting zones.
- All these systems should have access to DNS and have a FQHN (Fully-Qualified Host Name).
- To test, we should *ping* between hosts using their respective FQHNs, for example:-

root@hoki:~# ***ping squid.fatrain.com***
squid.fatrain.com is alive

- The main tasks involved in setting up the EVS are as follows:-
  - Installing the required EVS packages.
  - Setting up SSH authentication.
  - Configuring the EVS Controller system *squid*.
  - Deploying the EVS across the VM hosts *lotus* and *hoki*.
  - Configuring the VM hosts to use EVS settings.
  - Testing the configuration.

Networking

EVS manager and Controller

- On *squid* (our EVS Manager and Controller), we need to install the required software:-

pkg install evs
pkg install rad-evs-controller

- Now restart the *rad:local* SMF service:-

svcadm restart rad:local

- Now on zone hosts *lotus* and *hoki*, install the *evs* package:-

pkg install evs

- When the *evs* package is installed an *evsuser* user account is created, which will be used to perform all EVS operations.
- For successful and secure communications between *evsuser* accounts on the above systems we need to configure SSH access.

Networking

EVS SSH authentication

- SSH authentication will be key-based thus eliminating any need to enter passwords.
- SSH provides this mechanism by adding keys in the *authorized_keys* file stored in the *.ssh* directory of the user account home directory.
- Note that the *evsuser* home directory is */var/user/evsuser.*
- The keys we will need are:-
    - VM hosts *root* user keys on the EVS controller, as some actions involve the *zoneadmd* daemon on the VM hosts, which runs as *root*.
    - EVS Controller *evsuser* keys on the VM hosts.
- To proceed:-
- Log in to the first VM host as *root*, in our case *lotus*, and generate the required SSH public key:-

lotus # ***ssh-keygen -t rsa***

```
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa): <rtn>
Created directory '/root/.ssh'.
Enter passphrase (empty for no passphrase):<rtn>
Enter same passphrase again:<rtn>
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
a8:a8:ef:cc:56:f3:89:b2:9b:00:73:ab:f2:90:3a:30 root@lotus
```

- As shown above, when prompted to enter a password, just hit *Return*.

Networking

EVS SSH authentication

lotus # ***ls - l /root/.ssh***

```
total 7
-rw-------   1 root     root        1675 Dec 11 15:41 id_rsa
-rw-r--r--   1 root     root         392 Dec 11 15:41 id_rsa.pub
```

* The *id_rsa.pub* contents needs to be appended to the *.ssh.authorized_hosts* file in the *evsuser* home directory on the EVS controller *squid.*
* As we do not yet have the *authorized_hosts* file on *squid*, we can simply copy it as follows:-

lotus# ***scp /root/.ssh/id_rsa.pub ***
root@squid.fatrain.com:/var/user/evsuser/.ssh/authorized_keys

```
The authenticity of host 'squid.fatrain.com (192.168.200.22)' can't be established.
RSA key fingerprint is d0:9d:3f:7b:eb:4e:90:89:9b:c1:92:91:2c:8f:1e:16.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'squid.fatrain.com,192.168.200.22' (RSA) to the list of
known hosts.
Password:********
id_rsa.pub          100% |*****************************|   392       00:00
```

* Enter the *root* password for *squid* when prompted.
* The *authenticity* messages may be repeated at various stages of these examples.

- Now test the we can log in to *squid* as *evsuser* without having to give a password:-

lotus # ***ssh evsuser@squid.fatrain.com***

```
Oracle Corporation      SunOS 5.11      11.3    November 2016
evsuser@squid:~$ exit
```

* Works OK

Networking

EVS SSH authentication

- Now repeat the exercise on the second VM host *hoki*, but this time append the keys to *authorized_hosts* on *squid*:-

```
hoki # cat /root/.ssh/id_rsa.pub | ssh root@squid.fatrain.com \
 'cat >> /var/user/evsuser/.ssh/authorized_keys'
```

- Now test the login:-

```
hoki # ssh evsuser@squid.fatrain.com
Last login: Sun Dec 11 16:03:10 2016 from lotus
Oracle Corporation      SunOS 5.11      11.3    November 2016
evsuser@squid:~$ exit
```

* Works OK.

- Next, generate an SSH key as *root* on the EVS controller itself (*squid*) and append the key the *evsuser authorized_keys* file, again on *squid*:-

```
squid # ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa): <rtn>
Enter passphrase (empty for no passphrase):<rtn>
Enter same passphrase again:<rtn>
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
1d:42:e9:e5:87:c0:b6:e1:58:f4:43:87:3b:18:f3:7f root@squid
squid:~# cat /root/.ssh/id_rsa.pub \
        >> /var/user/evsuser/.ssh/authorized_keys
squid # ssh evsuser@squid.fatrain.com
Last login: Sun Dec 11 16:45:16 2016 from hoki
Oracle Corporation      SunOS 5.11      11.3    November 2016
evsuser@squid:~$ exit
```

Networking

EVS SSH authentication

- Not quite finished yet...
- Now we need to perform similar tasks for the *evsuser* on *squid*:-
  - Login as the *evsuser* on *squid* and generate an SSH key.
  - Append said key to the *authorized_keys* file of the *evsuser* on *lotus, hoki* and *squid* itself.
- Proceed as follows on EVS controller *squid,* assuming you are logged in as *root*:-

```
squid # su  -  evsuser
evsuser@squid $ ssh-keygen  -t  rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/var/user/evsuser/.ssh/id_rsa): <rtn>
Enter passphrase (empty for no passphrase):<rtn>
Enter same passphrase again:<rtn>
Your identification has been saved in /var/user/evsuser/.ssh/id_rsa.
Your public key has been saved in /var/user/evsuser/.ssh/id_rsa.pub.
The key fingerprint is:
05:e1:52:b6:fc:a7:6f:15:8c:59:62:3b:12:fd:c0:e7 evsuser@squid

evsuser@squid $ scp /var/user/evsuser/.ssh/id_rsa.pub   \
 root@lotus:/var/user/evsuser/.ssh/authorized_keys
evsuser@squid $ scp /var/user/evsuser/.ssh/id_rsa.pub   \
 root@lotus:/var/user/evsuser/.ssh/authorized_keys
evsuser@squid $ cat  /var/user/evsuser/.ssh/id_rsa.pub   \
        >>  /var/user/evsuser/.ssh/authorized_keys
```

- You may get a *Permission denied* message from the last command; to rectify make sure the file is owned by *evsuser*.

Networking

EVS SSH authentication

- Once again, test the logins to ensure no passwords are required:-

```
evsuser@squid $ ssh  evsuser@lotus
Last login: Sun Dec 11 18:25:50 2016 from 192.168.200.22
Oracle Corporation      SunOS 5.11      11.3    November 2016
evsuser@lotus $ exit
Connection to lotus closed.
evsuser@squid $ ssh  evsuser@hoki
Oracle Corporation      SunOS 5.11      11.3    November 2016
evsuser@hoki $ exit
Connection to hoki closed.
evsuser@squid $ ssh  evsuser@squid
Last login: Sun Dec 11 18:29:07 2016 from squid
Oracle Corporation      SunOS 5.11      11.3    November 2016
evsuser@squid $ exit
Connection to squid closed.
```

  - Works OK!

- Just to summarise what you would need to do if adding a further VM host:-
  - On the VM host, create SSH key for *root*, and append to the *authorized_keys* file of the *evsuser* account on the EVS Controller.
  - Append the *evsuser* account key on the EVS controller to the *evsuser authorised_keys* file on the VM host.
- All authentication configuration is now completed, and we can proceed to configure the actual EVS facility.

Networking

EVS Controller configuration

- On all nodes in the EVS (both VM hosts and Controller) we need to set an EVS property which defines the controller:-

```
squid # evsadm  set-prop  -p  \
        controller=ssh://evsuser@squid.fatrain.com
```

  - In our example, we run the above command as *root* on *squid, lotus* and *hoki*.

- Display the property settings on each system like this:-

```
squid # evsadm  show-prop
PROPERTY        PERM    VALUE                           DEFAULT
controller      rw      ssh://evsuser@squid.fatrain.com --
```

- Now, on *squid*, we need to set the type of network topology, referred to as L2 topology:-

```
squid # evsadm  set-controlprop  -p  l2-type=flat
```

  - EVS also supports VLAN and VXLAN topology.

- Note that the underlying physical network infrastructure may need configuration if using VLAN or VXLAN topology, so to avoid complications we will use the *flat* topology.

- An example only if VLAN is required (not us):-

```
squid # evsadm  set-controlprop  -p  l2-type=vlan
squid # evsadm  set-controlprop  -p  vlan-range=300-400
```

  - Sets the VLAN id range to 300-400.
  - One id will be used for each EVS created.

Networking

EVS Controller configuration

- Next we need to specify the name of the data port that will be used by default, to connect to the physical network over which the EVS will be deployed:-

```
squid # evsadm  set-controlprop  -p  uplink-port=net0
```

- Now also for the VM hosts:-

```
squid # evsadm  set-controlprop  -h  lotus.fatrain.com  \
        -p  uplink-port=net0
squid # evsadm  set-controlprop  -h  hoki.fatrain.com  \
        -p  uplink-port=net0
```

- In our example, the above is unnecessary as all systems will use *net0*.
- For extra protection consider using network link aggregations on your physical systems, then the *uplink-port* property would be set to an aggregation name such as *aggr0*.
- To view our property settings:-

```
squid # evsadm  show-controlprop  -p  l2-type,uplink-port
PROPERTY        PERM    VALUE   DEFAULT         HOST
l2-type         rw      flat    vlan            --
uplink-port     rw      net0    --              --
uplink-port     rw      net0    --              lotus.fatrain.com
uplink-port     rw      net0    --              hoki.fatrain.com
```

Networking

EVS Controller configuration - create an EVS

- Before creating the EVS, note that it is possible to associate it with a *tenant* name.
- A tenant would act as a container for resources associated with that name, and for example, could be used to represent a client name.
- Let us imagine we are configuring an EVS to connect the VMs we host for our client (tenant) *Acebikes*
- To create the EVS:-

squid # ***evsadm create-evs -T acebikes AB***

  - *-T acebikes* defines the tenant name.
  - *AB* is the EVS name.

squid # ***evsadm show-evs***

EVS	TENANT	STATUS	NVPORTS	IPNETS	HOST
AB	acebikes	idle	0	--	--

  - EVS: The name of the EVS.
  - TENANT: The name of the owning tenant.
  - STATUS: Whether the switch is idle or busy. It is busy if it has at least one active connection.
  - NVPORTS: The number of VPorts associated with the switch.
  - IPNETS: The IP networks associated with the switch. Currently only one is allowed.
  - HOST: The list of hosts that the EVS spans across.

Networking

EVS IP Networks and vports

- Obviously there is still a fair amount of configuration to be carried out with the new EVS.
- The next step is to define an IP network on the EVS, and then some connection points called vports.
- The IP network and vports will have addresses defined, as we shall see.
- Having done this we can assign a vport to a zone in one of the VM hosts.
- To add an IP network to our *HB* EVS:-

```
squid # evsadm add-ipnet -T acebikes -p \
        subnet=192.168.20.0/24 AB/ipnet
```

  - Defines the network *AB/ipnet* with a network address of 192.168.20.0/24. (*AB* is the EVS name).
  - Associates the IP net with tenant *acebikes*.

Networking

EVS IP Networks and vports

- We now add vports to the ipnet:-

```
squid # evsadm add-vport -T acebikes AB/vport0
squid # evsadm add-vport -T acebikes AB/vport1
squid # evsadm add-vport -T acebikes AB/vport2

squid # evsadm
NAME    TENANT      STATUS      VNIC    IP              HOST
AB      acebikes    idle        --      ipnet           --
 vport0 --          free        --      192.168.20.2/24 --
 vport1 --          free        --      192.168.20.3/24 --
 vport2 --          free        --      192.168.20.4/24 --
```

- Note how the vports have IP addresses assigned to them.
- When these vports are assigned to VNICs within zones, the addresses will be inherited by the VNICs.
- If required, the administrator can manually configure the vport address, but only when adding a vport, for example:-

```
squid # evsadm add-vport -T acebikes \
        -p ipaddr=192.168.20.10 AB/vport4
```

  - The *ipaddr* property cannot be changed subsequent to vport creation.

Networking

EVS - Assign vports to zones.

- The next task is to assign the vports to existing and new zones, in place of the network interfaces to which they would normally be bound.
- Imagine that we have zone *bikesdb* running on VM host *lotus*, and we wish to replace the existing network *anet* (bound to *net0* on the VM host *lotus*) with a vport.
- As *root* on *lotus*, halt the zone-

lotus # ***zoneadm -z bikesdb halt***

- Now configure the new network details:-

lotus # ***zonecfg -z bikesdb***
Use 'create' to begin configuring a new zone.
zonecfg:bikesdb> ***create***
create: Using system default template 'SYSdefault'
zonecfg:bikesdb> ***set tenant=acebikes***
zonecfg:bikesdb> s***elect anet linkname=net0***
zonecfg:bikesdb:anet> ***set evs=AB***
zonecfg:bikesdb:anet> ***set vport=vport0***
zonecfg:bikesdb:anet> ***end***
zonecfg:bikesdb> ***commit***
zonecfg:bikesdb> ***exit***

- Now boot the zone and connect to the console:-

zoneadm -z bikesdb boot; zlogin -C bikesdb

Networking

EVS - Assign vports to zones.

- This is the original network configuration as seen from the zone, before the EVS was used:-

bikesdb # ***ipadm show-addr net0/v4***

ADDROBJ	TYPE	STATE	ADDR
net0/v4	static	ok	192.168.200.88/24

- Here it is again, now using the EVS:-

bikesdb # ***ipadm show-addr net0/v4***

ADDROBJ	TYPE	STATE	ADDR
net0/v4	inherited	ok	192.168.20.2/24

- Quite clearly the IP address is now controlled by the vport settings on the EVS controller.
- This means that further network settings with zones in an EVS-controlled VLAN are all centred on the EVS controller, and should make the administrators life a bit easier.
- For the most part this would simply involve add more vports.
- Note that a specific IP address can be defined when adding a vport, but addresses cannot be changed subsequently.
- More about zones later in this course.

Networking

EVS - Testing the connections

- Having connected the zones to the EVS, the connections across the EVS should be tested with *ping*:-

ping 192.168.20.3
192.168.20.3 is alive

- Unless specified when creating the vports, the default router IP is the first IP address in the range created with the IP network on the EVS, so in our example this would be *192.168.20.1*.
- To examine the current EVS status:-

evsadm

NAME	TENANT	STATUS	VNIC	IP	HOST	
AB	acebikes	busy		--	ipnet	woody, hamm
vport0	--	used	bikesweb/net0	192.168.20.2/24	woody	
vport1	--	used	bikesdb/net0	192.168.20.3/24	hamm	
vport2	--	used	bikesweb2/net0	192.168.20.4/24	woody	
vport4	--	free	--	192.168.20.10/24	--	

* Note that the last entry is vport4, which was created by *evsadm* with a specific IP address, as shown earlier in the notes.

Networking

EVS - Applying Properties

- Now that the EVS is in place, we can have a look at some on-going EVS administration and configuration options.
- Firstly, bandwidth can be defined for the EVS, for example:-

```
squid # evsadm set-evsprop -T acebikes -p maxbw=200 AB
squid # evsadm show-evsprop -p maxbw
EVS     TENANT PROPERTY  PERM VALUE   DEFAULT     POSSIBLE
AB      acebikes maxbw     rw   200       --          --
```

  - Sets a maximum bandwidth for tenant *acebikes* on EVS *AB* to *200* Mbps.
- Bandwidth can also be applied to a vport, along with other properties such as:-
  - **ipaddr** - an IP address; can only be specified when creating a vport.
  - **priority** - relative priority of the vport, which can be set to *low, medium* (defautl) or *high*. Affects internal processing priority only, not priority on the wire.
  - **macaddr** - a MAC address; can only be specified when creating a vport.

Networking

EVS - Applying Properties

- Example of displaying then applying vport bandwidth:-

```
squid # evsadm  show-vportprop  -p  maxbw  AB/vport0
NAME    TENANT    PROPERTY  PERM VALUE       EFFECTIVE POSSIBLE
AB/vport0  acebikes   maxbw       rw      --          200          --

squid # evsadm  set-vportprop   -T  acebikes  \
        -p  maxbw=20  AB/vport0

squid # evsadm  show-vportprop  -p maxbw  AB/vport0
NAME    TENANT    PROPERTY  PERM VALUE       EFFECTIVE POSSIBLE
AB/vport0  acebikes   maxbw       rw      20          20           --
```

- If vports do not get unassigned correctly, perhaps when removing them from a zone, they can be reset:-

```
squid # evsadm  reset-vport   -T  acebikes  AB/vport3
```

Networking

EVS - Removal of vports, ipnets and EVS

- To remove a vport:-

squid # ***evsadm remove-vport -T acebikes AB/vport4***

- To remove an ipnet:-

squid # ***evsadm remove-ipnet -T acebikes AA/ipnet10***

- To remove an EVS:-

squid # ***evsadm delete-evs -T acebikes AA***

- For further information on EVS see:-

https://docs.oracle.com/cd/E53394_01/html/E54790/gnrgr.html#scrolltoc

- There is also an excellent technical article here:-

https://community.oracle.com/docs/DOC-910242

- This has a full example of creating a two-EVS configuration.
- .. and don't forget the *evsadm* manual page!