Update!

Well, just when I thought I had it all figured out. I get kicked in the junk and slapped in the face!

Apparently OVS is the flakiest half baked useful idea/app out there. It does exactly what I want, and lets me down when I need it most. To top it off, anyone posting anything about how to configure aspects of the application, only post the brain dead easy parts. Everyone else? Well they use Debian/Ubuntu and phat chance gleaming any knowledge from them, because they all post the same damn things too, and none of it applies to Red Hat or its variants.

Anyway, my old write up is broken. OVS will lose the configuration after a reboot. Maybe not the first or second reboot. But count on it leaving you high-n-dry.

It took extreme digging in the documentation to find just a sliver of hope. In addition, it took building the latest version from source (then I created my own RPMs from that). But! After playing mystery config for a little bit, I "think" I landed on a working configuration. Any searches in regards to my configuration... well the so called documentation was zero help. So I just had to throw some configurations at my setup, and see what stuck, or try and piece together the crappy ass error messages.

Honestly, I like this new method better as it makes things a bit more tangible, and less dependent on the crap OVS database functions.

Below I will give some detail to what I did to make this work and I will also post my "updated" configurations.

I hope it helps someone!

Real Quick Like...

In my previous write up, I detailed out the KVM and the server interface configurations. The KVM configs are still good, but I did need to modify the server interface configs (network-scripts). The interface configurations is what makes this whole thing work.

First things first, I uninstalled and removed all of OVS from my server.

```
# REMOVE OVS
systemctl stop openvswitch && systemctl disable openvswitch
systemctl stop ovs-vswitchd && systemctl disable ovs-vswitchd
systemctl stop ovsdb-server && systemctl disable ovsdb-server


dnf remove openvswitch* *openvswitch -y


rm -f /etc/sysconfig/openvswitch
rm -fR /etc/openvswitch*
rm -fR /var/run/openvswitch*
rm -fR /var/log/openvswitch*
```

Then I pulled down the the latest copy of the OVS source from Github. The version was 3.2.90 at the time.

https://github.com/openvswitch/ovs

```
# CLONE OVS SOURCE
cd /opt
git clone https://github.com/openvswitch/ovs.git
```

The OVS documentation explained how to check for dependencies and it "should" resolve them. However, I did some cleanup and manually loaded all the dev tools and dependencies. Being burned by OVS already, I have trust issues *(you can call it spite or perhaps like I always say, "just learning from my mistakes")*.

Once that was done, I built the RPMs.

NOTE: If you want your RPM's to be compiled with DPDK, you will need to also go grab the DPDK version 22.x source and build that as well. Rocky 8.8 only has the 21.x packages available. I opted to not compile it in.

Open vSwitch with DPDK

```
# CLEANED UP OLD PACKAGES, DNF CACHE, AND INSTALL DEPS
dnf autoremove
dnf clean all
dnf makecache


# REMOVED OLD KERNELS
dnf remove --oldinstallonly --setopt installonly_limit=2 kernel -y


# REINSTALLED LATEST KERNEL AND MODULES
dnf reinstall kernel kernel-core kernel-modules -y


# INSTALLED DEV TOOLS AND BUILD PACKAGES
```

```
dnf install @'Development Tools' dnf-plugins-core rpm-build make automake -y

# INSTALLED DEPS
dnf install python3-six python3-sphinx libunwind-devel unbound-devel \
libpfm-devel python3-libpfm libcap-ng libcap-ng-devel libpcap-devel \
bpf* libbpf* libnuma* numa*

# REBOOTED
reboot
```

Let OVS check dependencies and then build the RPMs.

```
# OVS DEP CHECK AND RPM BUILD
cd /opt/ovs

./boot.sh
./configure --prefix=/usr --localstatedir=/var --sysconfdir=/etc

sed -e 's/@VERSION@/3.2.90/' rhel/openvswitch-fedora.spec.in > /tmp/ovs.spec
dnf builddep /tmp/ovs.spec
rm -f /tmp/ovs.spec

make rpm-fedora RPMBUILD_OPT="--with check"

ls -al rpm/rpmbuild/RPMS/x86_64/
```

Install OVS RPMs...

```
# INSTALL OVS
cd rpm/rpmbuild/RPMS/x86_64/

dnf install \
openvswitch-3.2.90-1.el8.x86_64.rpm \
network-scripts-openvswitch-3.2.90-1.el8.x86_64.rpm \
openvswitch-devel-3.2.90-1.el8.x86_64.rpm
```

This is all that is needed for OVS.

I no longer plug my bridge or interface or port commands into OVS (using the `ovs-vsctl` module). I now plug all the pertinent config options into the network-scripts themselves.

Not once have I needed to give OVS any commands besides `ovs-vsctl show` just to check things.

I give my complete interface script setup on the "configurations page". So right now I am just going to show you a config for a single physical port, tied to a bridge, with a vlan passed through a trunk. Really once you grasp this, the rest of your interfaces are pretty simple.

- A "physical" server port needs tied to a "bridge" configuration. So we change the physical ports config like so...
  *(paying attention to the **`TYPE`**, **`DEVICETYPE`**, and **`OVS_BRIDGE`** options)*

```
# PHYSICAL PORT
# /etc/sysconfig/network-scripts/ifcfg-eth3

DEVICE=eth3
NAME=eth3
HWADDR=xx:xx:xx:xx:xx:xx
UUID=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
ONBOOT=yes
BOOTPROTO=none
TYPE=OVSPort
DEVICETYPE=ovs
OVS_BRIDGE=ovs3
HOTPLUG=no
```

- The bridge (ovs3 in this example) would now need configured as if it was the physical port in a basic setup (my "configurations page" has the basic setup too). But in this example I am going to configure it with the actual vlan information.
  *(now paying attention to the **`TYPE`**, **`DEVICETYPE`**, **`OVS_PORTS`** and **`OVS_EXTRA`** options)*

```
# BRIDGE INTERFACE
# /etc/sysconfig/network-scripts/ifcfg-ovs3

DEVICE=ovs3
NAME=ovs3
ONBOOT=yes
BOOTPROTO=none
TYPE=OVSBridge
DEVICETYPE=ovs
OVS_PORTS="ext0"
OVS_EXTRA="set port ovs3 trunk=310"
HOTPLUG=no
```

- Then we build the vlan interface. This interface would still be considered a "fake bridge", but because I am building out an actual interface script for it, OVS will not report it as a "fake bridge" in it's table (when you run the `ovs-vsctl` command "`ovs-vsctl list port ext0`"). This interface script informs the system of the name and the vlan that it is a member of. *(now paying attention to the **`TYPE`**, **`DEVICETYPE`**, **`OVS_BRIDGE`**, **`OVS_OPTIONS`** and **`OVS_EXTRA`** options)*

  ```
  # VLAN BRIDGE INTERFACE
  # /etc/sysconfig/network-scripts/ifcfg-ext0

  DEVICE=ext0
  NAME=ext0
  ONBOOT=yes
  BOOTPROTO=none
  TYPE=OVSIntPort
  DEVICETYPE=ovs
  OVS_BRIDGE=ovs3
  OVS_OPTIONS="tag=310"
  OVS_EXTRA="set Interface $DEVICE external-ids:iface-id=$DEVICE"
  HOTPLUG=no
  ```

That is it!

I can now restart my network or ifdown/up these scripts, and they will build the OVS configs. The major success I experienced with this method is, **IT WILL NOT BE DELETED OR FORGOTTEN WHEN I REBOOT!!!**

It took me a hot minute to cobble up this info and then another hotter minute to test out "`OVS_?`" options that have NO documentation. Wanna have some fun (guaranteed, anti fun)? Do a google search for "OVS_PORTS" and find anything, I mean anything what-so-ever that talks about OVS_PORTS. Then take it a step further, and correlate that search with anything that can be related to RHEL, CentOS, Rocky Linux, etc. etc.   You will find absolutely SQUAT!   **SQUAT!!**

The only information I was able to locate that related to RHEL based network script options at all was here (https://github.com/openvswitch/ovs/blob/master/rhel/README.RHEL.rst).

BONUS - You want some bonus anti fun? Go search the latest "official" OVS documentation for "ovs_port" (with or without the "s", with or without caps, use any variation of "OVS_"). OVS Official Documentation

What did you find???

Spoiler

SQUAT!!!

Alright, enough ranting.

Check out the configurations page for my "updated" scripts.

I hope this information helps someone!

Take care!

Revision #10
Created 2 October 2023 02:06:17 by Phatlix
Updated 3 October 2023 03:03:40 by Phatlix