lagopus-book Documentation

Release 0.2.3

NIPPON TELEGRAPH AND TELEPHONE CORPORATION

March 15, 2016

CONTENTS

Contents:

CHAPTER

ONE

ABOUT THIS BOOK

This book is intended to be a door opener for newbies who wants to try Lagopus vswitch.

This book will first introduce you about Lagopus vswitch, guide you how to install and perform basic configuration.

For further information, you can check *Sample Configuration: Building Layer 2 switch using Ryu as an OpenFlow controller* to understand and try a bit more advanced configuration of Lagopus vswitch.

People already familiar with Lagopus vswitch can use *Lagopus vswitch command options and examples* as reference to command line options available, *Using Lagopus CLI (lagosh)* to check how to use CLI in detail, and *Lagopus vswitch datastore and DSL syntax* to get overview of DSL (domain-specific language) used in configuration files.

CHAPTER

TWO

INTRODUCTION TO LAGOPUS VSWITCH

Lagopus vswitch is a yet another OpenFlow 1.3 software switch with Intel DPDK. Lagopus vswitch supports many network protocols, such as MPLS and PBB, as well as standard protocols in datacenter. Lagopus vswitch provides high-performance packet processing. Since version 0.2.3, it supports Hybrid mode which Lagopus vswitch will act as Layer 2/3 switch without OpenFlow controller to explicitly configure flows.

All of the code is freely available under the Apache 2.0 license.

Lagopus vswitch is written by C, Python.

2.1 Supported hardware

Lagopus can run on Intel x86 servers.

- CPU: Intel Xeon E5, E3, Core i7,i5, i3, Atom
- NIC: Intel DPDK Supported NIC (#1)
- Memory: >= 2GB

#1: DPDK supported NIC is only required when using DPDK and not when running in raw socket mode.

Lagopus can also run on virtual machines (VMs) like Oracl VM VirtualBox, Linux KVM, VMware, Microsoft Hyper-V etc. How to setup Lagopus on *Appendix: Installing Lagopus vswitch on VirtualBox VM* and *Lagopus vswitch using KVM Virtual Machine* are also described in this document.

2.2 Support

Lagopus Official site is https://lagopus.github.io/. Any questions and suggestions could be sent to Lagopus mailing list, lagopus-devel@lists.sourceforge.net .

2.3 Development

Your contribution is very welcomed. Please submit your patch code using github “Pull Requests”. If you find any bug, let us know by github’s Issues page.

For further information about how to get assistance, submit issues and pull request, refer to Lagopus GitHub Wiki Page located here : https://github.com/lagopus/lagopus/wiki

CHAPTER

THREE

LAGOPUS VSWITCH INSTALLATION (RAW SOCKET)

This section describes how to install Lagopus on Linux installed bear-metal server and basic configuration using "raw socket". * If you are using "DPDK", refer to *Lagopus vswitch Installation (DPDK)*

Actual steps are confirmed on VirtualBox VM, which should be identical to the steps on bear-metal server

3.1 Software Versions

- Lagopus vswitch: Lagopus vswitch 0.2.3
- OS: Linux Ubuntu Server 14.04.3 LTS

3.2 Lagopus installation steps

- Install Linux to a bear-metal server or a VM.
- Install necessary packages.

```
$ sudo apt-get update
$ sudo apt-get install build-essential libexpat-dev libgmp-dev \
    libssl-dev libpcap-dev byacc flex git \
    python-dev python-pastedeploy python-paste python-twisted
```

- Download and extract Lagopus vswitch source code.

```
$ wget https://github.com/lagopus/lagopus/archive/v0.2.3.tar.gz
$ tar xvf v0.2.3.tar.gz
or
$ git clone https://github.com/lagopus/lagopus.git
$ cd lagopus
$ git tag -l
$ git checkout -b 0.2.3 refs/tags/v0.2.3
```

Note: * You can use `git checkout refs/tags/v0.2.3` if you want to checkout without creating branch.

- Compile Lagopus vswitch (raw socket)

```
$ cd lagopus-0.2.3
$ ./configure --disable-dpdk
$ make
```

- Install lagopus package

```
$ sudo make install
```

3.3 Setup Lagopus configuration file

Sample Lagopus configuration (DSL format) is "misc/examples/lagopus.dsl". `lagopus.dsl` file must be located at the same directory of the executable of lagopus vswitch or under `/usr/local/etc/lagopus/`.

- Copy sample configuration file under `/usr/local/etc/lagopus/`.

```
$ sudo mkdir /usr/local/etc/lagopus/
$ cd ~/lagopus-0.2.3
$ sudo cp misc/examples/lagopus.dsl /usr/local/etc/lagopus/lagopus.dsl
```

- Edit configuration file suited to your environment.
- Example:
  - OpenFlow controller is "127.0.0.1"
  - eth0 is management interface. (Thus does not appear in the configuration)
  - eth1, eth2 are used as Lagopus data port. (raw socket)

```
$ sudo vi /usr/local/etc/lagopus/lagopus.dsl
      channel channel01 create -dst-addr 127.0.0.1 -protocol tcp
      controller controller01 create -channel channel01 -role equal -connection-type
       ↪  main
      interface interface01 create -type ethernet-rawsock -device eth1
      interface interface02 create -type ethernet-rawsock -device eth2
      port port01 create -interface interface01
      port port02 create -interface interface02
      bridge bridge01 create -controller controller01 -port port01 1 -port port02 2
       ↪  -dpid 0x1
      bridge bridge01 enable
```

3.4 Running Lagopus vswitch

Enter `$ sudo lagopus` to run Lagopus vswitch. Enter `show version` command from lagosh to confirm it's running. Enter `stop` command from lagosh to stop Lagopus vswitch.

```
$ sudo lagopus
$ lagosh
Lagosh> show version
{
    "product-name": "Lagopus",
    "version": "0.2.2-release"
}
Lagosh> stop
Lagosh> show version
Socket connection refused.  Lagopus is not running?
Lagosh> exit
$
```

- For more configuration options, refer to *Lagopus vswitch command options and examples*

- For more information about lagosh, refer to *Using Lagopus CLI (lagosh)*

CHAPTER

FOUR

LAGOPUS VSWITCH INSTALLATION (DPDK)

This section describes how to install Lagopus on Linux installed bear-metal server and basic configuration using "DPDK: Data Plane Development Kit".

- If you are using "raw socket" instead of DPDK, refer to *Lagopus vswitch Installation (raw socket)*
- For more information about DPDK, check http://dpdk.org/

Actual steps are confirmed on VirtualBox VM, which should be identical to the steps on bear-metal server

4.1 Software Versions

- Lagopus vswitch: Lagopus vswitch 0.2.3
- OS: Linux Ubuntu Server 14.04.3 LTS

4.2 Lagopus installation steps

- Install Linux to a bear-metal server or a VM.
- Install necessary packages.

```
$ sudo apt-get update
$ sudo apt-get install build-essential libexpat-dev libgmp-dev \
    libssl-dev libpcap-dev byacc flex git \
    python-dev python-pastedeploy python-paste python-twisted
```

- Download Lagopus vswitch source code.

```
$ git clone https://github.com/lagopus/lagopus.git
$ cd lagopus
$ git tag -l
$ git checkout -b 0.2.3 refs/tags/v0.2.3
```

- Compile Lagopus vswitch (DPDK)

```
$ ./configure
$ make
```

Note:

- You can use `git checkout refs/tags/v0.2.3` if you want to checkout without creating branch.

- You would face below error if you run ./configure in a source tree you downloaded without using git (ex: downloading .tar.gz / .zip files)
- This is because running `git submodule update --init` in the script to download DPDK source tree requires .git file.
- Use `git clone` to download source tree as mentioned in steps above when using DPDK.

```
/home/<usr>/lagopus-0.2.3/mk/make_dpdk.sh /home/<usr>/lagopus-0.2.3 src/dpdk \
                "x86_64" "linuxapp" gcc
fatal: Not a git repository (or any of the parent directories): .git
make[1]: *** [dpdk] Error 1
make[1]: Leaving directory `/home/<usr>/lagopus-0.2.3'
make: *** [prerequisite] Error 2
configure: error: Prerequisite failure.
```

- Install lagopus package

```
$ sudo make install
```

4.3 Setup DPDK environment

4.3.1 Kernel module setup

- Load the kernel modules.
  - The kernel modules built are available in the src/dpdk/build/kmod directory.

```
$ sudo modprobe uio
$ cd lagopus
$ sudo insmod ./src/dpdk/build/kmod/igb_uio.ko
$ sudo insmod ./src/dpdk/build/kmod/rte_kni.ko
$ lsmod | egrep 'uio|kni'
rte_kni               282624  0
igb_uio                16384  0
uio                    20480  1 igb_uio
```

Note: rebooting OS will undo above. Repeate steps above after reboot.

4.3.2 Huge pages

Make hugepages available to DPDK. You can setup hugepage in two ways:

1. Manual configuration: Repeate steps after reboot if you select this.
2. Script configuration: Select this to keep it permanent after reboot.

Note: When configured manually (1), rebooting OS will undo the change.

1. Manual confguration (undone after reboot)

```
$ sudo sh -c "echo 256 >
 ↪  /sys/devices/system/node/node0/hugepages/hugepages-2048kB/nr_hugepages"
$ sudo mkdir -p /mnt/huge
$ sudo mount -t hugetlbfs nodev /mnt/huge
```

2. Script configuration (permanent after reboot)

- Add hugepages option of the linux kernel to reserve 256 pages of 2 MB.

```
$ sudo vi /etc/sysctl.conf
vm.nr_hugepages = 256
```

- Add the following line to /etc/fstab so that mount point can be made permanent across reboots.

```
$ sudo mkdir -p /mnt/huge
$ sudo vi /etc/fstab
nodev /mnt/huge hugetlbfs defaults 0 0
```

- Confirm HugePages are configured properly by below commands.

```
$ grep -i "HugePages" /proc/meminfo
AnonHugePages:         0 kB
HugePages_Total:     256
HugePages_Free:      256
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       2048 kB
$ mount | grep huge
nodev on /mnt/huge type hugetlbfs (rw)
```

4.3.3 NIC (Network Interface Card) assignment

Notes:

- You need to unbound NIC from kernel (ixgbe driver) before using it with DPDK.
- You will lose connection to the OS if you unbound NIC used for management (ex: ssh).

Steps:

- Check PCI ID of the NIC you want to use for DPDK. (eth1, eth2)

```
~/lagopus$ sudo ./src/dpdk/tools/dpdk_nic_bind.py --status

Network devices using DPDK-compatible driver
============================================
<none>

Network devices using kernel driver
===================================
0000:00:03.0 '82540EM Gigabit Ethernet Controller' if=eth0 drv=e1000 unused= *Active*
0000:00:08.0 '82545EM Gigabit Ethernet Controller (Copper)' if=eth1 drv=e1000 unused=
0000:00:09.0 '82545EM Gigabit Ethernet Controller (Copper)' if=eth2 drv=e1000 unused=

Other network devices
=====================
<none>
```

- Unbound NICs from ixgbe driver and registerd with igb_uio driver.

Replace `0000:00:08.0 0000:00:09.0` with PCI ID of your environment.

```
~/lagopus$ sudo ./src/dpdk/tools/dpdk_nic_bind.py --bind=igb_uio 0000:00:08.0
 ↪  0000:00:09.0
~/lagopus$ sudo ./src/dpdk/tools/dpdk_nic_bind.py --status
```

```
Network devices using DPDK-compatible driver
============================================
0000:00:08.0 '82545EM Gigabit Ethernet Controller (Copper)' drv=igb_uio unused=
0000:00:09.0 '82545EM Gigabit Ethernet Controller (Copper)' drv=igb_uio unused=

Network devices using kernel driver
===================================
0000:00:03.0 '82540EM Gigabit Ethernet Controller' if=eth0 drv=e1000 unused=igb_uio
 ↪  *Active*

Other network devices
=====================
<none>
```

4.4 Setup Lagopus configuration file

Sample Lagopus configuration (DSL format) is "misc/examples/lagopus.dsl". `lagopus.dsl` file must be located at the same directory of the executable of lagopus vswitch or under `/usr/local/etc/lagopus/`.

- Copy sample configuration file under `/usr/local/etc/lagopus/`.

```
$ sudo mkdir /usr/local/etc/lagopus/
$ cd ~/lagopus
$ sudo cp misc/examples/lagopus.dsl /usr/local/etc/lagopus/lagopus.dsl
```

- Edit configuration file suited to your environment.
- Example:
  - OpenFlow controller is "127.0.0.1"
  - eth0 is management interface. (Thus does not appear in the configuration)
  - eth1, eth2 are used as Lagopus data port. (DPDK)

```
$ sudo vi /usr/local/etc/lagopus/lagopus.dsl
channel channel01 create -dst-addr 127.0.0.1 -protocol tcp
controller controller01 create -channel channel01 -role equal -connection-type main
interface interface01 create -type ethernet-dpdk-phy -port-number 0
interface interface02 create -type ethernet-dpdk-phy -port-number 1
port port01 create -interface interface01
port port02 create -interface interface02
bridge bridge01 create -controller controller01 -port port01 1 -port port02 2 -dpid
 ↪  0x1
bridge bridge01 enable
```

4.5 Running / Stopping Lagopus vswitch

- Enter `$ sudo lagopus -d -- -c3 -n2 -- -p3` to run Lagopus vswitch.
- Or, enter `$ sudo lagopus -- -c3 -n2 -- -p3` to run Lagopus vswitch in background.
- Enter `show version` command from lagosh to confirm it's running.
- Enter `stop` command from lagosh to stop Lagopus vswitch.

```
$ sudo lagopus -- -c3 -n2 -- -p3
$ lagosh
      Lagosh> show version
      {
          "product-name": "Lagopus",
          "version": "0.2.2-release"
      }
Lagosh> stop
Lagosh> show version
Socket connection refused.  Lagopus is not running?
Lagosh> exit
$
```

- For more configuration options, refer to *Lagopus vswitch command options and examples*
- For more information about lagosh, refer to *Using Lagopus CLI (lagosh)*

Note:

- You need 2 or more processors (CPU cores) to run Lagopus vswitch using DPDK.
- Run `top` and enter `1` to see one (or more) core(s) running 100% in pooling mode.

```
top - 15:50:26 up 6 min,  1 user,  load average: 0.34, 0.13, 0.07
Tasks:  83 total,   2 running,  81 sleeping,   0 stopped,   0 zombie
%Cpu0  :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu1  :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:   3081312 total,   686804 used,  2394508 free,    20676 buffers
KiB Swap:  3143676 total,        0 used,  3143676 free.    78792 cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 1205 root      20   0 1037728  16264   6764 S 100.2  0.5   0:25.58 lagopus
```

CHAPTER

FIVE

LAGOPUS VSWITCH USING KVM VIRTUAL MACHINE

This section describes:

- How to install Lagopus on Virtual Machine (VM) based on KVM
- Example configuration to run Lagopus vswitch as Layer 2 switch using Ryu as an OpenFlow controller.
- “raw socket” will be used.

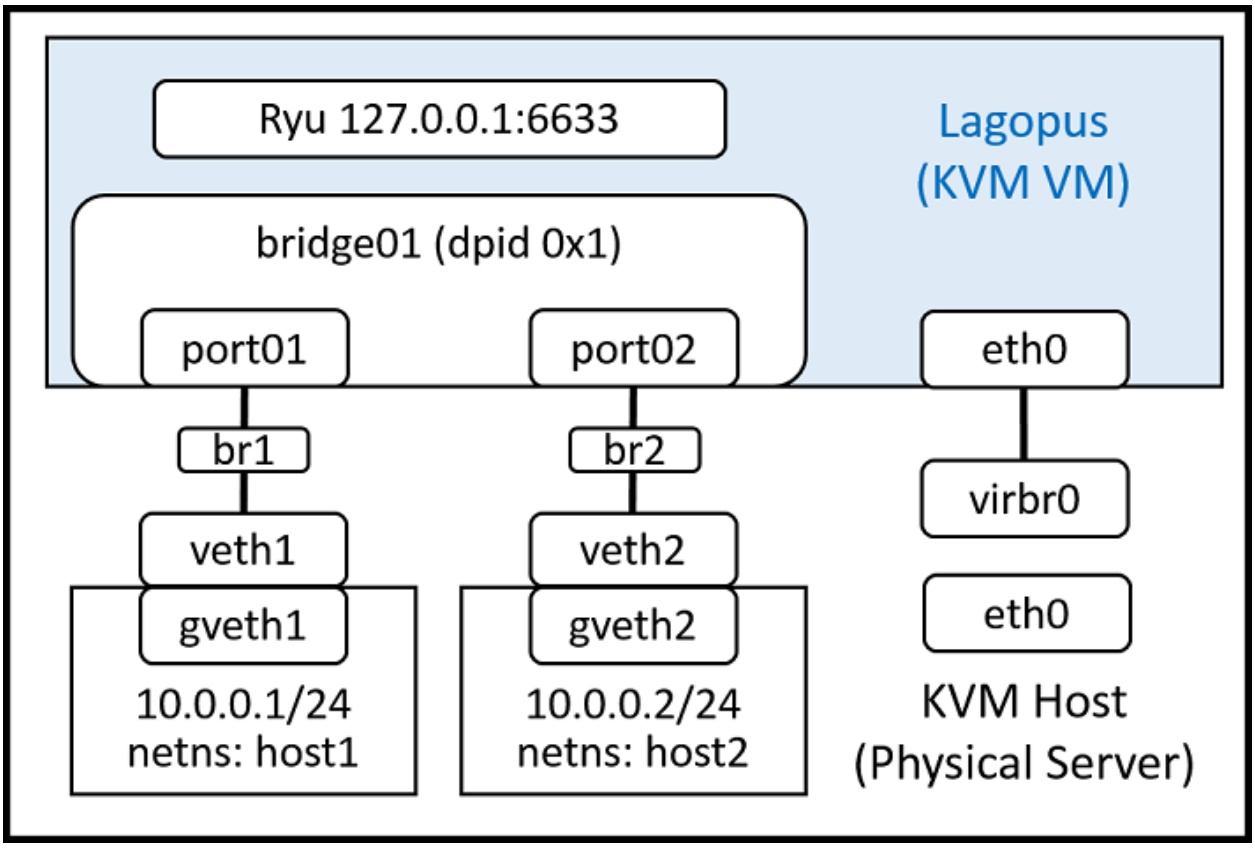
5.1 Software Versions

- Lagopus vswitch: Lagopus vswitch 0.2.3
- OS: Linux Ubuntu Server 14.04.3 LTS

5.2 Overall topology and client host setup

In this example, we will connect 2 client hosts to Lagopus vswitch bridge ports.

- KVM VM running Lagopus vswitch, client hosts, and bridges to connect them are all created on KVM host.
- Client hosts are created using network namespace (netns) for simplicity.
- You can also create and use KVM VM as client host instead of netns.



5.3 Setup client hosts and bridges on KVM host

- edit `/etc/network/interfaces` to setup vethpair, netns and bridge.
- reboot to apply the changes you just made.

  Note: `sudo /etc/init.d/networking restart` does not work on Ubuntu 14.04.

```
$ sudo vi /etc/network/interfaces
$ cat /etc/network/interfaces
auto veth1
iface veth1 inet manual
    pre-up ip link add veth1 type veth peer name gveth1
    pre-up ip netns add host1
    pre-up ip link set gveth1 netns host1
    pre-up ip netns exec host1 ip link set gveth1 up
    pre-up ip netns exec host1 ip addr add 10.0.0.1/24 brd + dev gveth1
    post-down ip link del veth1

auto veth2
iface veth2 inet manual
    pre-up ip link add veth2 type veth peer name gveth2
    pre-up ip netns add host2
    pre-up ip link set gveth2 netns host2
    pre-up ip netns exec host2 ip link set gveth2 up
    pre-up ip netns exec host2 ip addr add 10.0.0.2/24 brd + dev gveth2
    post-down ip link del veth2
```

```
auto br1 br2
iface br1 inet manual
    bridge_ports veth1
iface br2 inet manual
    bridge_ports veth2
```

- Confirm bridges are created and interfaces are connected.

```
$ brctl show
bridge name     bridge id               STP enabled     interfaces
br1             8000.a2b53bff4545       no              veth1
br2             8000.fe7797ec6779       no              veth2
virbr0          8000.000000000000       yes
```

- Confirm IP address is correctly configured on client hosts (gveth1/gveth2).

```
$ sudo ip netns exec host1 ip a
1: lo: <LOOPBACK> mtu 65536 qdisc noop state DOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
9: gveth1@if10: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast state
 ↪  LOWERLAYERDOWN group default qlen 1000
    link/ether c2:5f:77:74:3e:97 brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.1/24 brd 10.0.0.255 scope global gveth1
       valid_lft forever preferred_lft forever

$ sudo ip netns exec host2 ip a
1: lo: <LOOPBACK> mtu 65536 qdisc noop state DOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
11: gveth2@if12: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast state
 ↪  LOWERLAYERDOWN group default qlen 1000
    link/ether ca:6a:eb:34:3c:f0 brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.2/24 brd 10.0.0.255 scope global gveth2
       valid_lft forever preferred_lft forever
```

5.4 KVM installation and VM creation (Ubuntu)

Note: Refer to ubuntu documentation: KVM/Installation for details about installing and running KVM on Ubuntu.

5.4.1 KVM installation

- KVM requires CPU supporting Intel VT or AMD-V. Confirm it with below command.
- It's supported if result was 1 or more. Not supported if 0 (zero).

```
$ egrep -c '(vmx|svm)' /proc/cpuinfo
4
```

- Install necessary packages. (listed below)
- reboot and log in.
- Check installation was completed using `virsh` and `lsmod`.

```
$ sudo apt-get update
$ sudo apt-get install qemu-kvm libvirt-bin ubuntu-vm-builder bridge-utils virtinst
$ sudo reboot

... after reboot and log in...

$ virsh version
Compiled against library: libvirt 1.2.2
Using library: libvirt 1.2.2
Using API: QEMU 1.2.2
Running hypervisor: QEMU 2.0.0

$ lsmod | grep kvm
kvm_intel             163840  0
kvm                   507904  1 kvm_intel
```

5.4.2 VM creation using virt-install

There are two different options to create VM on headless server, `virt-install` and `vmbuilder`. The difference is `vmbuilder` is for creating Ubuntu-based guests, whereas virt-install lets you install other kinds of operationg systems, which are other Linux distributions and other OS like Windows, FreeBSD etc.

Note: There used to be a command called "Ubuntu-vm-builder", which is now a wrapper to vmbuilder. It's now only maintained for compatibility wih previous scripts, thus recommend using vmbuilder.

- Download ISO image of Ubuntu Server. (in `~/img`)

```
$ mkdir ~/img
$ cd ~/img
~/img$ wget
 ↪  http://ftp.riken.jp/Linux/ubuntu-releases/14.04.4/ubuntu-14.04.3-server-amd64.iso
```

- Create VM (use `--dry-run` to check without actually creating VM)

```
$ sudo virt-install \
--name lagopus \
--ram 4096 \
--disk path=/var/lib/libvirt/images/lagopus.img,size=30 \
--vcpus 2 \
--cpu host \
--os-type linux \
--os-variant ubuntutrusty \
--virt-type=kvm \
--noautoconsole \
-c ~/img/ubuntu-14.04.3-server-amd64.iso \
--graphics vnc,listen=0.0.0.0,password=vnc

Starting install...
Allocating 'lagopus.img'                                |  30 GB     00:00
Creating domain...                                      |    0 B     00:00
Domain installation still in progress. You can reconnect to
the console to complete the installation process.

(Confirm VM was created successfully)
$ virsh list
 Id    Name                           State
```

```
----------------------------------------------------
 4     lagopus                        running
```

- Continue OS (Ubuntu) installation on the VM.
  - Check vnc display number assigned to the newly created VM.
  - Connect to the VM via VNC. <kvm-host-ip>:<vnc-display>

```
$ virsh vncdisplay lagopus
:0
# VNC display number is "0" for above case.
# Continue installation by connecting via VNC.
```

- Start VM after OS installation.
- VM will shut down after installing OS and would not start automatically by default.

  Note: hostname of the VM was set as **lagopus-kvm** in this example.

```
$ virsh list --all
 Id    Name                           State
----------------------------------------------------
 -     lagopus                        shut off

$ virsh start lagopus
Domain lagopus started

$ virsh list --all
 Id    Name                           State
----------------------------------------------------
 4     lagopus                        running
```

5.4.3 Connecting to VM via SSH from KVM host (Usermode Networking)

By default, VM will have interface automatically connected to a bridge created local to KVM host. (ex: "virbr0")

This is called "Usermode Networking", where you can:

- Access external network from VM via host (NAT).
- Connect to VM from host via SSH.

To connect to VM from host via SSH:

- Connect to VM via VNC and confirm IP address assigned to it.
- `$ ssh <VM-usermode-interface-ip>` from KVM host.

5.4.4 Adding network interface to VM

- Confirm existing network interfaces on VM (named "lagopus").

```
$ virsh domiflist lagopus
Interface  Type       Source     Model       MAC
-------------------------------------------------------
-          network    default    virtio      52:54:00:6a:f2:f8
```

- Add two interfaces to VM. (use different MAC address)

- Script to generate MAC address.

```
$ vi genmac.sh
#!/bin/bash
# generate a random mac address for the qemu nic
printf '52:54:00:%02x:%02x:%02x\n' $((RANDOM%256)) $((RANDOM%256)) $((RANDOM%256))

example:
$ ./genmac.sh
52:54:00:f0:bd:b1
```

- virsh command to add interfaces to VM.

```
$ virsh attach-interface --domain lagopus --type bridge \
        --source br1 --model virtio \
        --mac 52:54:00:f0:bd:b1 --config --live
Interface attached successfully
$ virsh attach-interface --domain lagopus --type bridge \
        --source br2 --model virtio \
        --mac 52:54:00:0f:69:9c --config --live
Interface attached successfully
```

- Options:

```
--domain : name of the VM you wan to add interface
--source : bridge on KVM host to connect the interface
--type   : type of interface: network, bridge
--model  : NIC model: virtio, e1000 etc.
--mac    : MAC address of the interface
--config : flag to make it persistant after VM reboot
--live   : flag to apply change immediately on a running VM
```

- Confirm interfaces are created and connected to bridge “br1” and “br2”.

```
$ virsh domiflist lagopus
Interface  Type       Source     Model       MAC
-------------------------------------------------------
vnet0      network    default    virtio      52:54:00:6a:f2:f8
vnet1      bridge     br1        virtio      52:54:00:f0:bd:b1
vnet2      bridge     br2        virtio      52:54:00:0f:69:9c

$ brctl show
bridge name     bridge id               STP enabled     interfaces
br1             8000.a2b53bff4545       no              veth1
                                                        vnet1
br2             8000.fe54000f699c       no              veth2
                                                        vnet2
virbr0          8000.fe54006af2f8       yes             vnet0
```

- Connect to VM (via SSH)
- Confirm two interfaces (eth1/eth2) were added.

```
$ ssh <vm-ip-address>
<usr>@<vm-ip-address>'s password:

lagopus-kvm:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
```

```
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group
 ↪  default qlen 1000
    link/ether 52:54:00:6a:f2:f8 brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.224/24 brd 192.168.122.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::5054:ff:fe6a:f2f8/64 scope link
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether 52:54:00:f0:bd:b1 brd ff:ff:ff:ff:ff:ff
4: eth2: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether 52:54:00:0f:69:9c brd ff:ff:ff:ff:ff:ff

lagopus-kvm:~$ ethtool -i eth1
driver: virtio_net
version: 1.0.0
firmware-version:
bus-info: 0000:00:06.0
supports-statistics: no
supports-test: no
supports-eeprom-access: no
supports-register-dump: no
supports-priv-flags: no
```

- Make newly added interface (eth1/eth2) permanetly up during boot.

  Note: you don't need to assign IP address to Lagopus ports.

```
lagopus-kvm:~$ sudo vi /etc/network/interfaces
# add below lines
auto eth1
iface eth1 inet manual
  up ip link set eth1 up
  down ip link set eth1 down

auto eth2
iface eth2 inet manual
  up ip link set eth2 up
  down ip link set eth2 down
```

- reboot or manually apply setting by `ifup` command.

  Note: `sudo /etc/init.d/networking restart` does not work on Ubuntu 14.04.

```
lagopus-kvm:~$ sudo ifup eth1
lagopus-kvm:~$ sudo ifup eth2
```

5.5 Lagopus installation steps on KVM VM

Proceed following steps inside the VM.

Note: all operations in this section are done INSIDE the VM

5.5.1 Installing Lagopus to VM (raw socket)

- Install necessary packages. (described below)

```
lagopus-kvm:~$ sudo apt-get install build-essential libexpat-dev libgmp-dev \
    libssl-dev libpcap-dev byacc flex git \
    python-dev python-pastedeploy python-paste python-twisted
```

- Download Lagopus vswitch source code.

```
lagopus-kvm:~$ git clone https://github.com/lagopus/lagopus.git
lagopus-kvm:~$ cd lagopus
lagopus-kvm:~$ git tag -l
lagopus-kvm:~$ git checkout -b 0.2.3 refs/tags/v0.2.3
```

- Compile and install Lagopus vswitch (raw socket)

```
lagopus-kvm:~$ ./configure --disable-dpdk
lagopus-kvm:~$ make
lagopus-kvm:~$ sudo make install
```

5.5.2 Setup Lagopus configuration

- Copy sample configuration file under /usr/local/etc/lagopus/.

```
lagopus-kvm:~$ sudo mkdir /usr/local/etc/lagopus/
lagopus-kvm:~$ cd ~/lagopus
lagopus-kvm:~$ sudo cp misc/examples/lagopus.dsl /usr/local/etc/lagopus/lagopus.dsl
```

- Edit lagopus.dsl
- Example:
  - OpenFlow controller is “127.0.0.1”
  - eth0 is management interface. (Thus does not appear in the configuration)
  - eth1, eth2 are used as Lagopus data port. (raw socket)

```
lagopus-kvm:~$ sudo vi /usr/local/etc/lagopus/lagopus.dsl
  channel channel01 create -dst-addr 127.0.0.1 -protocol tcp
  controller controller01 create -channel channel01 -role equal -connection-type main
  interface interface01 create -type ethernet-rawsock -device eth1
  interface interface02 create -type ethernet-rawsock -device eth2
  port port01 create -interface interface01
  port port02 create -interface interface02
  bridge bridge01 create -controller controller01 -port port01 1 -port port02 2 -dpid
   ↪  0x1
  bridge bridge01 enable
```

- Run Lagopus vswitch
- Confirm two interfaces are recognized.

```
lagopus-kvm:~$ sudo lagopus
lagopus-kvm:~$ lagosh
Lagosh> show version
{
```

```
    "product-name": "Lagopus",
    "version": "0.2.2-release"
}
Lagosh>
Lagosh> show interface
[
    {
        "name": "interface01",
        "rx-dropped": 0,
        "is-enabled": true,
        "tx-errors": 0,
        "rx-bytes": 0,
        "tx-packets": 0,
        "rx-packets": 0,
        "tx-bytes": 0,
        "rx-errors": 0,
        "tx-dropped": 0
    },
    {
        "name": "interface02",
        "rx-dropped": 0,
        "is-enabled": true,
        "tx-errors": 0,
        "rx-bytes": 0,
        "tx-packets": 0,
        "rx-packets": 0,
        "tx-bytes": 0,
        "rx-errors": 0,
        "tx-dropped": 0
    }
]
```

- Stop Lagopus vswitch and exit lagosh.

```
Lagosh> stop
Lagosh> exit
lagopus-kvm:~$
```

5.6 Building Layer 2 switch using Ryu as an OpenFlow controller

5.6.1 Setup Ryu Simple switch

- Install necessary packages.
- Use pip command and install Ryu.

```
lagopus-kvm:~$ sudo apt-get install python-setuptools python-pip python-dev \
  libxml2-dev libxslt-dev
lagopus-kvm:~$ sudo pip install ryu
```

- Run below only if required.

```
lagopus-kvm:~$ sudo pip install oslo.config
lagopus-kvm:~$ sudo pip install six --upgrade
```

- Run simple_switch Ryu application.

```
lagopus-kvm:~$ ryu-manager --verbose
 ↪  /usr/local/lib/python2.7/dist-packages/ryu/app/simple_switch_13.py
```

- Open new SSH connection and run Lagopus vswitch.

```
lagopus-kvm:~$ sudo lagopus
```

- Confirm Lagopus is connected to Ryu from ryu-manager console log.

```
connected socket:<eventlet.greenio.base.GreenSocket object at 0x7f9c71c1bb50>
 ↪  address:('127.0.0.1', 35705)
hello ev <ryu.controller.ofp_event.EventOFPHello object at 0x7f9c71bad1d0>
move onto config mode
EVENT ofp_event->SimpleSwitch13 EventOFPSwitchFeatures
switch features ev
 ↪  version=0x4,msg_type=0x6,msg_len=0x20,xid=0xb0597afd,OFPSwitchFeatures(auxiliary_id=0,capabiliti
move onto main mode
```

5.6.2 Confirm ping works via Lagopus vswitch

- On KVM host, ping between namespace host1/host2.

```
$ sudo ip netns exec host1 ping 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=11.5 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.745 ms
...
$ sudo ip netns exec host2 ping 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=0.770 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.798 ms
...
```

CHAPTER

SIX

LAGOPUS VSWITCH COMMAND OPTIONS AND EXAMPLES

Lagopus vswitch command options can be classified into 3 groups.

- Global options
- Intel DPDK options
- Datapath options

Each option groups should be ordered and separated by `--` (double dash) like below.

```
$ sudo lagopus [global options] [-- [dpdk options] [-- datapath options] ]

Example:
$ sudo lagopus -l lagopus.log -- -c3 -n1 -- -p7
                 <-  global  ->    <-dpdk->   <-datapath->
```

Examples of how to use options are described in *Lagopus command line examples* later in this section.

6.1 Global option

All mandatory options are marked (Mandatory). Others are optional.

- `-d` : Run in debug mode
- `--` : Run in daemon mode
- `-v | --version` : Show version
- `-h | -? | --help` : Show help
- `-l filename | --logfile filename` : Specify a log/trace file path [default: syslog]
- `-p filename | --pidfile filename` : Specify a pid file path [default: /var/run/lt-lagopus.pid]
- `-C filename | --config filename` : Specify a configuration file (DSL format) path

6.2 Intel DPDK Option

All mandatory options are marked (Mandatory). Others are optional.

- `-c` : (Mandatory)
  - Speicify which CPU cores are assigned to Lagopus with a hex bitmask of CPU port to use
  - Example: `-c9`

        * Only CPU core 0 and core 3 are assigned, then let bit0 and bit3 on.
        * 0x1001 (binary) = 9 (Hexadecimal)
- `-n` : (Mandatory)
  - Specify number of memory channel which CPU supports
  - Range: 1,2,4
  - Example: `-n2`
    * a CPU supports dual memory channlel
- `-m` :
  - Specify how much Hugepage memory is assigned to Lagopus in mega byte.
  - Maximum hugepage memory is assigned if no option is specified.
  - Example: `-m 512`
    * Use 512MB for Lagopus
- `--socket-mem` :
  - Specify how much Hugepage memory for each CPU socket is assigned to Lagopus in mega byte.
  - This option is exclusive with `-m` option

6.3 Datapath option

All mandatory options are marked (Mandatory). Others are optional.

6.3.1 Common option

- `--no-cache` :
  - Don't use flow cache [default: use flow cache]
- `--fifoness TYPE` :
  - Select packet ordering (FIFOness) mode [default: none]
  - `none` : FIFOness is disabled
  - `port` : FIFOness per each port.
  - `flow` : FIFOness per each flow.
  - Example: `--fifoness flow`
    * Specify flow-level FIFOness
- `--hashtype TYPE` :
  - Select key-value store type for flow cache [default: intel64]
  - `intel64` : Hash with Intel CRC32 and XOR (64bit)
  - `city64` : CityHash (64bit)
  - `murmur3` : MurmurHash3 (32bit)
  - Example: `--hashtype city64`

        * Use CityHash
- `--kvstype TYPE`:
  - Select key-value store type for flow cache [default: hashmap_nolock]
  - `hashmap_nolock` : Use hashmap without reader and writer lock
  - `hashmap` : Use hashmap with reader and writer lock
  - `ptree` : Use patricia tree
  - Example: `--kvstype ptree`
    * Use patricia tree for key-value store

6.3.2 CPU core and packet processing

Dataplane of Lagopus provides two options to assign CPU core and packet processing worker.

- Automatic assignment option
- Explicit assignment option

These options are exclusive.

Automatic assignment option

- `-p PORTMASK` : (Mandatory)
  - hexadecimal bitmask of ports to be used
  - Example: `-p3`
    * Use port 0 and port 1
- `-w NWORKERS`:
  - number of packet processing worker [default: assign as possible]
  - System assigns CPU core automatically.
  - Example: `-w8`
    * Use total 8 cores for data plane
- `--core-assign TYPE`:
  - Select automatically core assign policy [default: performance]
  - `performance` : Don't use HTT core. (e.g. use only 8 cores on 8C16T processor)
  - `balance` : Use HTT core (e.g. use 16 cores on 8C16T processor)
  - `minimum` : Use only one core.

Explicit assignment option

Current Lagopus limitation: youngest core number can not be specified among available CPU cores.

- `--rx "(PORT, QUEUE, CORE),(PORT, QUEUE, CORE),...,(PORT, QUEUE, CORE)"`:
  - List NIC RX assignment policy of NIC RX port, queue, and CPU core with the combination of (PORT, QUEUE, CORE) and ","

  - `PORT` : Port number [0 - n]
  - `QUEUE`: Queue number [default: 0]
  - `CORE` : CPU core number
  - Example: `--rx '(0,0,2),(1,0,3)'`
    * Assign port 0 to CPU core #2 and port 1 to CPU core #3
- `--tx "(PORT,CORE),(PORT,CORE),...,(PORT,CORE)"`:
  - List NIC TX assignment policy of NIC TX port and CPU core with the combination of (PORT, CORE) and ","
  - `PORT`: Port number [0 - n]
  - `CORE`: CPU core number
  - Example: `--tx '(0,4),(1,5)'`
    * Assign port 0 TX to CPU core #4 and port 1 TX to CPU core #5
- `--w "CORE, ..., CORE"`:
  - List of the CPU cores for packet processing with ","
  - Example: `--w '8,9,10,11,12,13,14,15'`
    * Assign CPU core 8 - 15 for packet processing

6.4 Lagopus command line examples

6.4.1 Simple run

- CPU core: 1 and 2
- Number of memory channel: 1
- NIC port: 0 and 1
- Packet processing workers are automatically assigned to CPU cores.
- Run in debug-mode

```
$ sudo lagopus -d -- -c3 -n1 -- -p3
```

6.4.2 Run with explicit assignment to achieve maximum performance

- CPU core: 1 - 7
  - core # 2 for I/O RX
  - core # 3 for I/O TX
  - core # 4, # 5, #6, # 7 for packet processing
- Memory channel: 2
- NIC port: 0 and 1
- Run in debug-mode

```
$ sudo lagopus -d -- -cfe -n2 -- --rx '(0,0,2),(1,0,2)' --tx '(0,3),(1,3)' --w 4,5,6,7
```

6.4.3 Run with packet-ordering in flow-level

- CPU core: 1 - 15
- Memory channel: 2
- NIC port: 0, 1, 2
- Worker assignment is automatic
- FIFOness option to ensure packet-ordering in flow-levle

```
$ sudo lagopus -- -cfffe -n2 -- -p7 --fifoness flow
```

CHAPTER

SEVEN

USING LAGOPUS CLI (LAGOSH)

7.1 Starting lagosh

Lagopus comes with CLI (Command Line Interface) named "lagosh".

- To start lagosh, simply type `lagosh` after starting lagopus.

```
$ sudo lagopus
$ lagosh
Lagosh>
```

- You would see error below if Lagopus is not running or did not start properly.

```
Lagosh> show version
Socket connection refused.  Lagopus is not running?
```

- Check log output (`-l <logfile>`), configuration files (`lagopus.dsl etc.`) and try running lagopus/lagosh again.

lagosh has two modes, operational and configuration.

- Prompt `Lagosh>` shows you are in operational mode.
- Prompt `Configure#` shows you are in configuration mode.
- Enter `configure` to move from operational mode to configuration mode.
- Enter `exit` or `quit` in configuration mode to exit configuration mode and move back to operational mode.
- Enter `exit` or `quit` in operational mode to exit lagosh and move back to Linux shell.

```
$ lagosh
Lagosh>
Lagosh> configure
Configure#
Configure# exit
Lagosh> exit
$
```

7.2 Using lagosh

Entering `?` or `help` will present available commands.

```
Lagosh> ?

Documented commands (type help <topic>):
========================================
configure  help  log  ping  show  ssh  stop  telnet  traceroute

Undocumented commands:
======================
EOF  exit  pager  quit  shell
```

Entering `help <topic>` or `? <topic>` will present brief explanation of the command (topic).

```
Lagosh> help show

        Show statistics.
        Usage
                show bridge
                show flow
                show mactable
                show interface
                show table
```

`[tab]` key is used to auto-complete commands and present available commands and options. For Example:

- Typing `s` followed by the `[tab]` key will display available commands starting with `s`.
- Typing `sh` followed by the `[tab]` key will complete to `show`.
- Typing `[tab]` key twice after `show[space]` will present available options for the `show` command.

```
Lagosh> s[tab]
show  ssh   stop
Lagosh> sh[tab]
Lagosh> show

Lagosh> show [tab][tab]
bridge      channel     controller  flow        group       interface   mactable
meter       port        version
```

Refer to below for more explanation of each commands.

- *lagosh common commands*
- *lagosh operational commands*
- *lagosh configuration commands*

7.3 lagosh common commands

This section describes details of commands available in both operational and configuration mode.

- `help`, `?`
  - List available commands with "help" or detailed help with "help cmd".
  - See *Using lagosh* for more details and examples.
- `quit`, `exit`, `EOF`
  - Entering one of these commands will move you out from current mode.

  - configuration mode -> operational mode -> Linux shell

```
Configure# EOF
Lagosh> EOF
$
```

- `pager`
  - default: `off`
  - turn on/off paging with command output has more line than the size of console.
  - Use `[space]` to scroll one page.
  - You can also use `[page up]`, `[page down]`, `[up]`, `[down]` keys to scroll up / down.
- `shell`
  - Execute Linux shell command.
  - For exmaple, you can open Lagopus configuration file, run bash and come back to the lagosh CLI like below:

```
Lagosh> shell vi /usr/local/etc/lagopus/lagopus.dsl
(vi will open "lagopus.dsl")
Lagosh>
Lagosh> shell bash
user@lagopus:~$ ls
lagopus-0.2.3  v0.2.3.tar.gz
user@lagopus:~$ exit
exit
Lagosh>
```

7.4 lagosh operational commands

This section describes details of each operational commands.

- `configure`
  - Enter configuration mode.
  - See *lagosh configuration commands* for details of configuration mode commands.

```
Lagosh> configure
Configure#
```

- `log`
  - Show log settings.

```
Lagosh> log
{
    "packetdump": [
        ""
    ],
    "ident": "lagopus",
    "debuglevel": 0
}
```

- `ping <target>`

  - Ping a remote target.
  - Argument <target> is mandatory, and could take IPv4 address or hostname.
  - Enter `Ctrl+c` to stop.

```
Lagosh> ping localhost
PING localhost (127.0.0.1) 56(84) bytes of data.
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0.018 ms
64 bytes from localhost (127.0.0.1): icmp_seq=2 ttl=64 time=0.020 ms
^C
--- localhost ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 0.018/0.019/0.020/0.001 ms
```

- `traceroute <target>`
  - Traceroute to remote target.
  - Argument <target> is mandatory, and could take IPv4/IPv6 address or hostname.

```
Lagosh> traceroute localhost
traceroute to localhost (127.0.0.1), 30 hops max, 60 byte packets
 1  localhost (127.0.0.1)  0.021 ms  0.006 ms  0.004 ms
Lagosh>
```

- Note: `traceroute` calles traceroute installed on OS and could return error similar to below if not installed. Exit lagosh and install traceroute in such case.

```
Lagosh> traceroute 127.0.0.1
sh: 1: traceroute: not found
```

- `ssh <target>`,`telnet <target>`
  - ssh or telnet to a remote target.
  - Attribute <target> is mandatory, and could take IPv4 address or hostname.

```
Lagosh> ssh localhost
user@localhost's password:
Welcome to Ubuntu 14.04.3 LTS (GNU/Linux 3.19.0-25-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

 System information disabled due to load higher than 1.0

Last login: Mon Feb  8 12:44:26 2016 from localhost
user@lagopus:~$ exit
logout
Connection to localhost closed.
Lagosh>
```

- `stop`
  - stop Lagopus.
  - There is no command to start Lagopus. Either exit lagosh or use `shell` command to start Lagopus again.

```
Lagosh> shell ps -eo uname,pid,args | grep lagopus
root      1279 lagopus
user      1304 sh -c ps -eo uname,pid,args | grep lagopus
```

```
user      1306 grep lagopus
Lagosh>
Lagosh> stop
Lagosh> shell ps -eo uname,pid,args | grep lagopus
user      1307 sh -c ps -eo uname,pid,args | grep lagopus
user      1309 grep lagopus
```

- `show <arg> [<arg> ...]`
  - Show statistics of objects specified by <arg> in JSON format.
  - Possible arguments:
    - `bridge`
    - `controller`
    - `group`
    - `mactable`
    - `port`
    - `channel`
    - `flow`
    - `interface`
    - `meter`
    - `version`
  - For some objects, you can specify members by name to limit output.
  - See below for an example using `show port`.

```
Lagosh> show port
[
    {
        "name": "port01",
        "state": [
            "link-down"
        ],
        "is-enabled": true,
        "supported-features": [],
        "config": [],
        "curr-features": []
    },
    {
        "name": "port02",
        "state": [
            "link-down"
        ],
        "is-enabled": true,
        "supported-features": [],
        "config": [],
        "curr-features": []
    }
]

Lagosh> show port port01
{
```

```
    "supported-features": [],
    "state": [
        "link-down"
    ],
    "config": [],
    "name": "port01",
    "curr-features": []
}

Lagosh> show port port01 state
[
    "link-down"
]
```

7.5 lagosh configuration commands

In lagosh configuration mode, you can view and edit Lagopus vswitch configurations.

> This feature is still under development.
> Please use this with special care and check for updates in new version.

- lagosh configuration mode is using **"git"** in background.
- Configure your git email and name before start using the configuration mode.

```
$ git config --global user.email "you@example.com"
$ git config --global user.name "Your Name"
```

- To enter configuration mode, enter `configure` in operational mode.

```
Lagosh> configure
Configure#
```

- Working files used by lagosh configuration mode are located under `$HOME/.lagopus.conf.d/`. Avoid directly editing files under this directory.
- The directory (`.lagopus.conf.d/`) will be created automatically when you first issue one of the following commands in configuration mode.
  - `show`
  - `edit`
  - `ls`

Typical configuration command lifecycle will be in below order.

1. `edit` : edit configuration
2. `diff` : check configuration diff from current one
3. `commit` : apply to running configuration
4. `save` : save to file to be permanent after restart

The remaining part of this section is reference to each configuration commands.

- `show`
  - Show configuration read from lagopus datastore.

```
Configure# show
log {
        syslog;
        ident lagopus;
        debuglevel 0;
        packetdump "";
}
datastore {
        addr 0.0.0.0;
        port 12345;
        protocol tcp;
        tls false;
}
agent {
        channelq-size 1000;
        channelq-max-batches 1000;
}

... snip: more configuration will continue ...
```

- `edit [<filename>]`
  - Launch text editor to edit candidate configuration.
  - If `<filename>` was not specified, file `lagopus.conf` will be opened. It will be created if it did not exist.
  - If you specify `<filename>`, the file specified will be opened. It will be created if it did not exist.
- `diff`
  - Show differences from previous configuration.
  - You an find modified mtu is highlighted in `diff` command output.

```
Configure# edit
[master 90c1232]
 1 file changed, 1 insertion(+), 1 deletion(-)
# modify mtu of interface01 from 1500 to 1000 and exit editor.

Configure# diff
diff --git a/lagopus.conf b/lagopus.conf
index 1a00103..90c5ee2 100644
--- a/lagopus.conf
+++ b/lagopus.conf
@@ -28,7 +28,7 @@ interface {
        interface01 {
                type ethernet-rawsock;
                device eth1;
-               mtu 1500;
+               mtu 1000;
                ip-addr 127.0.0.1;
        }
        interface02 {
```

- `history`
  - Show differences from previous configuration as git commit history.

```
Configure# history
commit 90c1232c7ef7407a7851bd71a4784e4e0e79c30b
Author: Your Name <you@example.com>
Date:   Mon Feb 8 18:08:36 2016 +0900

commit 03cc54c1b5e635cb1128e2fe6f2db1948093a0eb
Author: Your Name <you@example.com>
Date:   Mon Feb 8 18:08:11 2016 +0900
```

- `commit`
  - Apply configuration changes to Lagopus vswitch currently running.
  - `lagopus.conf.dsl` will be created and running configuration will be stored.
- `save`
  - Save current configuration to be permanent acress restart.
  - Configuration will be written to the file: `/usr/local/etc/lagopus/lagopus.dsl`, which is used during Lagopus startup by default.
  - Specifying filename is not supported yet.
- `ls`
  - List files under `$HOME/.lagopus.conf.d/`.
- `load <file>`
  - Load configuration file and commit immediately.
  - Currently raw DSL output is only supported.
- `set` (Not implemented yet)
  - Add or modify candidate configuration line.
- `unset` (Not implemented yet)
  - Remove candidate configuration line.

CHAPTER

EIGHT

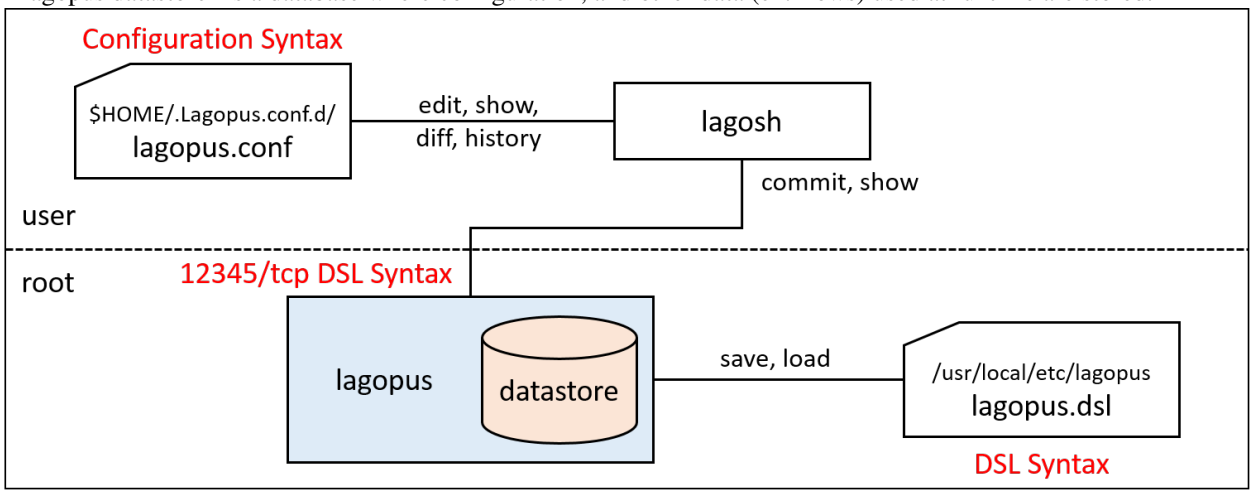
LAGOPUS VSWITCH DATASTORE AND DSL SYNTAX

This section describes how Lagopus vswitch stores configuration data in it's datastore at runtime, and explain each parameters used in DSL syntax configuration file which is loaded to datastore at boot time.

There are two types of syntax (format) used to describe configuration of Lagopus vswitch. Refer to *Configuration syntax types* for details about both syntax.

8.1 datastore overview

"Lagopus datastore" is a database where configuration, and other data (ex: flows) used at runtime are stored.



This diagram shows how datastore interacts with configuration files (`lagopus.conf`, `lagopus.dsl`) and `lagosh` command.

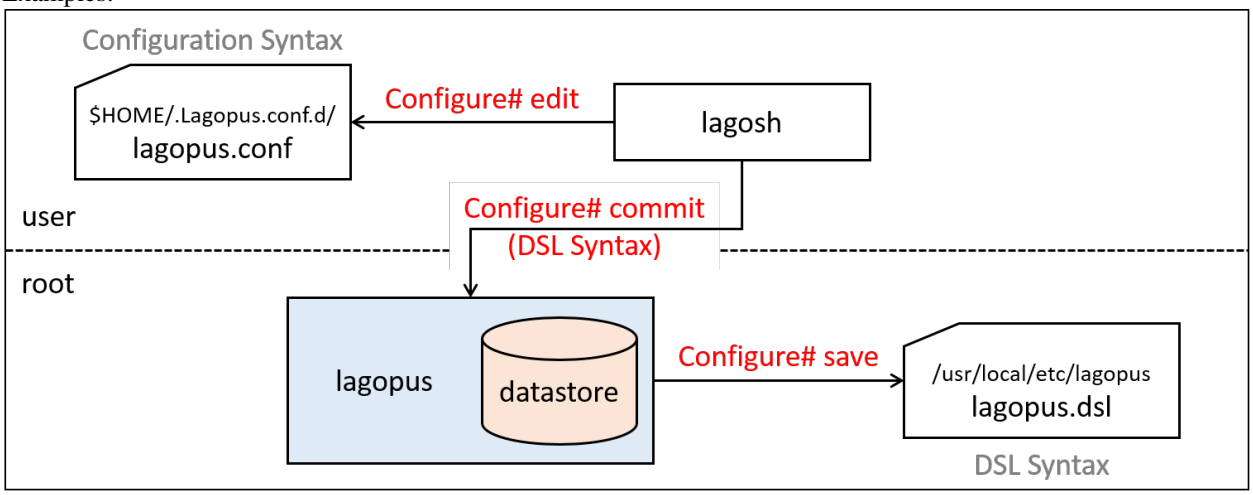
- `$HOME/.lagopus.conf.d/*.conf` (ex: `lagopus.conf`) are written in lagosh configuration syntax.
- `/usr/loca/etc/lagopus/*.dsl` (ex: `lagopus.dsl`) are written in DSL syntax.
- Since datastore can only read files in DSL syntax, lagosh will translate `*.conf` file to DSL syntax while communicating with lagopus datastore via TCP.
- lagopus will read `*.dsl` directly from the file at boot time.
  - Use `-C filename` or `--config filename` option to specify configuration file. (DSL format)
  - If not specified, lagopus will load `/usr/loca/etc/lagopus/lagopus.dsl`

8.1.1 lagosh operation and datastore

Instead of editing `*.dsl` file directly, you can edit/commit/save configuration using lagosh.

Refer to *Using Lagopus CLI (lagosh)* for detailed explanation about lagosh configuration commands.

Examples:



- Start lagosh and enter configuration mode.

```
$ lagosh
Lagosh> configure
Configure#
```

- `Configure# edit` will create working file under `$HOME/.lagopus.conf.d/`
  - edit file in lagosh configuration syntax.
- `Configure# commit` will apply changes to lagopus running configuration (datastore).
- `Configure# save` will write running configuration to `/usr/local/etc/lagopus/lagopus.dsl` in DSL syntax.

8.2 Configuration syntax types

There are two types of syntax (format) used to describe configuration of Lagopus vswitch.

- configuration syntax
  - Used in `*.conf` files.
  - Used when editing via `lagosh`.
- DSL syntax
  - Used in `*.dsl` files.
  - Used by `lagopus` and it's `datastore`.

"configuration syntax" consists of `object-name`, `identifier`, and `attribute value` pair. Some object-name may not have identifier.

```
<object-name>
  [identifier] {
    <attribute value>;
    ...
  }
}

# example:
interface {
        interface01 {
                type ethernet-rawsock;
                device eth1;
                mtu 1500;
                ip-addr 127.0.0.1;
        }
}
```

In "DSL syntax", each line describes single configuration command.

```
<object-name> [identifier] [operation] <-attribute value> ...

# example:
interface interface01 create -type ethernet-rawsock -device eth1 -mtu 1500 -ip-addr
 ↪  127.0.0.1
```

Refer to *Configurable objects and DSL syntax* for details about DSL syntax.

8.2.1 Converting between `.conf` and `.dsl` syntax

By using `--dsl-encode` `--dsl-decode` option, you can convert between `.conf` and `.dsl` syntax.

Examples:

- `.conf` to `.dsl`

```
$ head .lagopus.conf.d/lagopus.conf
log {
        syslog;
        ident lagopus;
        debuglevel 0;
        packetdump "";
}
datastore {
        addr 0.0.0.0;
        port 12345;
        protocol tcp;

$ lagosh --dsl-encode .lagopus.conf.d/lagopus.conf
log -syslog -ident lagopus -debuglevel 0 -packetdump ""
datastore -addr 0.0.0.0 -port 12345 -protocol tcp -tls false
agent -channelq-size 1000 -channelq-max-batches 1000
... snip ...
```

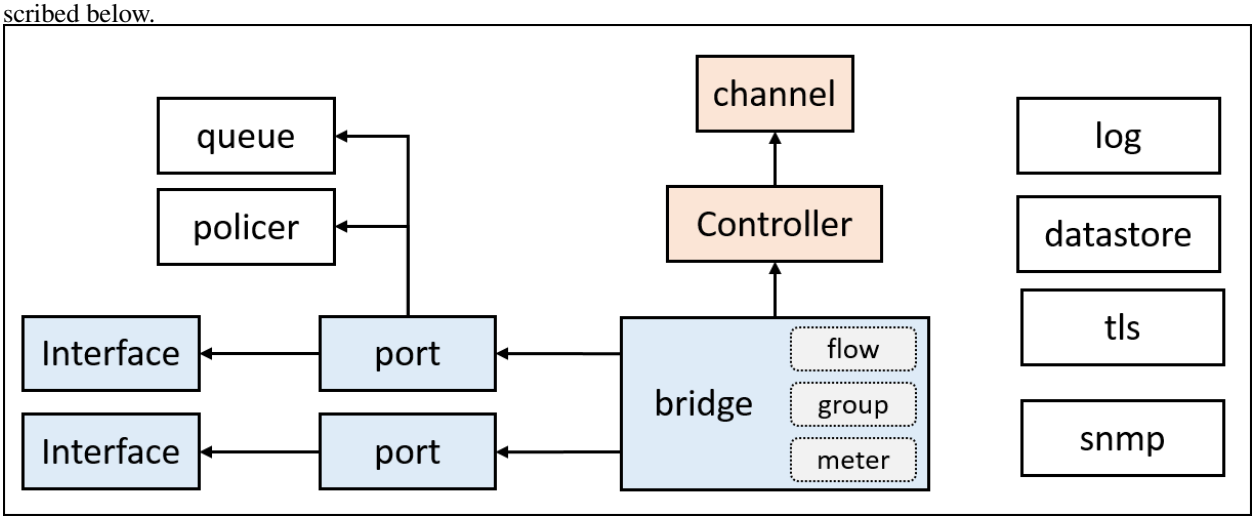
- `.dsl` to `.conf`

```
$ head /usr/local/etc/lagopus/lagopus.dsl
# all the log objects' attribute
```

```
log -syslog -ident lagopus -debuglevel 0
log -packetdump ""

# all the datastore objects' attribute
datastore -addr 0.0.0.0 -port 12345 -protocol tcp -tls false

# all the agent objects' attribute
agent -channelq-size 1000 -channelq-max-batches 1000

$ lagosh --dsl-decode /usr/local/etc/lagopus/lagopus.dsl
log {
        syslog;
        ident lagopus;
        debuglevel 0;
        packetdump "";
}
datastore {
        addr 0.0.0.0;
        port 12345;
        protocol tcp;
        tls false;
}
agent {
        channelq-size 1000;
        channelq-max-batches 1000;
}
... snip ...
```

8.3 Configurable objects and DSL syntax

This diagram shows overview of configurable objects of lagopus. Attributes of major configurable objects are described below.



8.3.1 log object

“log object” sets log output destination and level.

```
log <attribute value> ...

# example:
log -syslog -ident lagopus -debuglevel 0
log -file /home/user/lagopus.log -debuglevel 0
```

- `-syslog` Set log destination to syslog. takes no value.
- `-file` Set log destination to file.
  - You can set two lines each specifying `-syslog` and `-file`, but cannot use both attributes in one line.
- `-ident` Only with `-syslog`. char string ident used in syslog entry. default `lagopus`.
- `-debuglevel` Takes value of `0 ~ MAXIMUM_DBGLVL(UINT16_MAX)`. default `0`.
  - check `lagopus_msg_debug` in source code for usage of debuglevel.

8.3.2 datastore object

"datastore object" sets parameters to connect to lagopus datastore.

```
datastore <attribute value> ...
# example:
datastore -addr 0.0.0.0 -port 12345 -protocol tcp -tls false
```

- `-addr` address of the datastore process.
- `-port` tcp port the datastore process listens to.
- `-protocol` protocol used to connect to datastore. `tcp` or `tcp6`.
- `-tls` use TLS or not. `false` or `true`.

8.3.3 tls object

"tls object" sets parameters related to TLS.

```
tsl <attribute value> ...

# example:
tls -cert-file /usr/local/etc/lagopus/catls.pem -private-key
 ↪  /usr/local/etc/lagopus/key.pem -certificate-store /usr/local/etc/lagopus
 ↪  -trust-point-conf /usr/local/etc/lagopus/check.conf
```

- `-cert-file` location of cert file. (.pem)
- `-private-key` location of key file. (.pem)
- `-certificate-store` location to store certificates.
- `-trust-point-conf` location of trust-point-conf file. (.conf)

8.3.4 policer-action object

"policer-action object" sets type of policer action.

```
policer-action <policer-action-identifier> create <attribute value>

# example:
policer-action pa01 create -type discard
```

- `policer-action-identifier` Name to identify the policer-action
- `-type` Type of policer action. Currently type `discard` is only availble.

8.3.5 policer object

"policer object" sets policer parameters.

```
policer <policer-identifier> create <attribute value> ...

# example:
policer policer01 create -action pa01 -bandwidth-limit 10000 -burst-size-limit 11000
 ↪  -bandwidth-percent 20
```

- `policer-identifier` Name to identify the policer.
- `-action` Name of policer-action associated with the policer.
- `-bandwidth-limit`
- `-burst-size-limit`
- `-bandwidth-percent`

8.3.6 queue object

"queue object" sets parameters of queues used by OpenFlow `set-queue` action.

```
queue <queue-identifier> create <attribute value> ...

# example:
queue queue01 create -type two-rate -id 1 -priority 50
```

- `queue-identifier` Name to identify the queue.
- `-type` Type of queue. `single-rate` or `two-rate`.
- `-id` ID used in OpenFlow `set-queue` action.
- `-priority` Nonnegative integer, `0 ~ 65535(UINT16_MAX)`.
  - Packets will be scheduled on each port using weighted round robin based on ratio of priority of queues.
- `-color` Behavior based on color. `color-aware` or `color-blind`.
- Optional options for both `single-rate` and `two-rate`
  - `-committed-burst-size` CBS in bytes.
  - `-committed-information-rate` CIR in bps.
- Optional option for `single-rate`
  - `-excess-burst-size` EBS in bytes.
- Optional options for `two-rate`

  - `-peak-burst-size` PBS in bytes.
  - `-peak-information-rate` PIR in bps.

8.3.7 interface object

"interface object" sets parameters of interface.

```
interface <interface-identifier> create <attribute value> ...

# example:
interface interface01 create -type ethernet-rawsock -device eth1 -mtu 1500 -ip-addr
 ↪  127.0.0.1
interface interface01 create -type ethernet-dpdk-phy -port-number 0
```

- `interface-identifier` Name to identify the interface.
- `-type` Type of the interface. One of below.
  - `ethernet-dpdk-phy`
  - `ethernet-dpdk-vdev`
  - `ethernet-rawsock`
  - `gre`
  - `nvgre`
  - `vxlan`
  - `vhost-user`
- `-device` Name of the device associated with the interface. PCI ID for dpdk.
- `-port-number` DPDK port number. Only used by dpdk.
- `-mtu` MTU of the interface.
- `-ip-addr` IP address of the interface.

  Note: Either `-device` or `-port-number` should be specified per line, not both.

8.3.8 port object

"port object" sets port and interface assosication.

```
port <port-identifier> create <attribute value>

# example:
port port01 create -interface interface01
```

- `port-identifier` Name to identify the port.
- `-interface` interface-identifier assosiated with the port.
- `-policer` policer-identifier assosiated with the port.
- `-queue` queue-identifier assosiated with the port.

8.3.9 channel object

"channel object" sets parameters of channel used to communicate with OpenFlow controller.

```
channel <channel-identifier> create <attribute value> ...

# example:
channel channel01 create -dst-addr 127.0.0.1 -dst-port 6633 -local-addr 0.0.0.0
 ↪  -local-port 0 -protocol tcp
```

- `channel-identifier` Name to identify the channel
- `-dst-addr` IP address of the controller
- `-dst-port` tcp port number of the controller
- `-local-addr` source IP address used when connecting to controller
- `-local-port` tcp port used when connecting to controller. `0` = automatically assigned.
- `-protocol` protocol used when connecting to controller. `tcp` or `tls`.

8.3.10 controller object

"controller object" sets parameters related to OpenFlow controller.

```
controller <controller-identifier> create <attribute value> ...

# example:
controller controller01 create -channel channel01 -role equal -connection-type main
```

- `controller-identifier` Name to identify the controller.
- `-channel` channel-identifier used to connect to the controller.
- `-role` Role of the controller. `master`, `slave` or `equal`.
- `-connection-type` Controller connection type. `main` or `auxiliary`.

8.3.11 bridge object

"bridge object" sets parameters of bridge.

```
bridge <bridge-identifier> create <attribute value> ...

# example:
bridge bridge01 create -dpid 1 -controller controller01 -port port01 1 -port port02 2
 ↪  -port port03 3 -fail-mode standalone
```

- `bridge-identifier` Name to identify the bridge.
- `-dpid` Datapath ID. Nonnegative integer.
- `-controller` Name of controller (s) assosiated with the bridge.
- `-port <port-identifier> <openflow-port-id>` Port name and OpenFlow port ID pair(s) assoiated with the bridge.
  - `port-identifier` Name of the port defined in port object.

  - `openflow-port-id` Port ID to be used in OpenFlow protocol. Nonnegative integer.
- `-fail-mode` Mode when connection to controller was failed. `secure` or `standalone`.
- Other optional options.
  - `-flow-statistics`
  - `-group-statistics`
  - `-port-statistics`
  - `-queue-statistics`
  - `-table-statistics`
  - `-reassemble-ip-fragments`
  - `-max-buffered-packets`
  - `-max-ports`
  - `-max-tables`
  - `-max-flows`
  - `-block-looping-ports`
  - `-action-type`
  - `-instruction-type`
  - `-reserved-port-type`
  - `-group-type`
  - `-group-capability`
  - `-packet-inq-size`
  - `-packet-inq-max-batches`
  - `-up-streamq-size`
  - `-up-streamq-max-batches`
  - `-down-streamq-size`
  - `-down-streamq-max-batches`

CHAPTER

NINE

SAMPLE CONFIGURATION: BUILDING LAYER 2 SWITCH USING RYU AS AN OPENFLOW CONTROLLER

This section describes how to configure Lagopus vswitch as Layer 2 switch using Ryu as an OpenFlow controller.

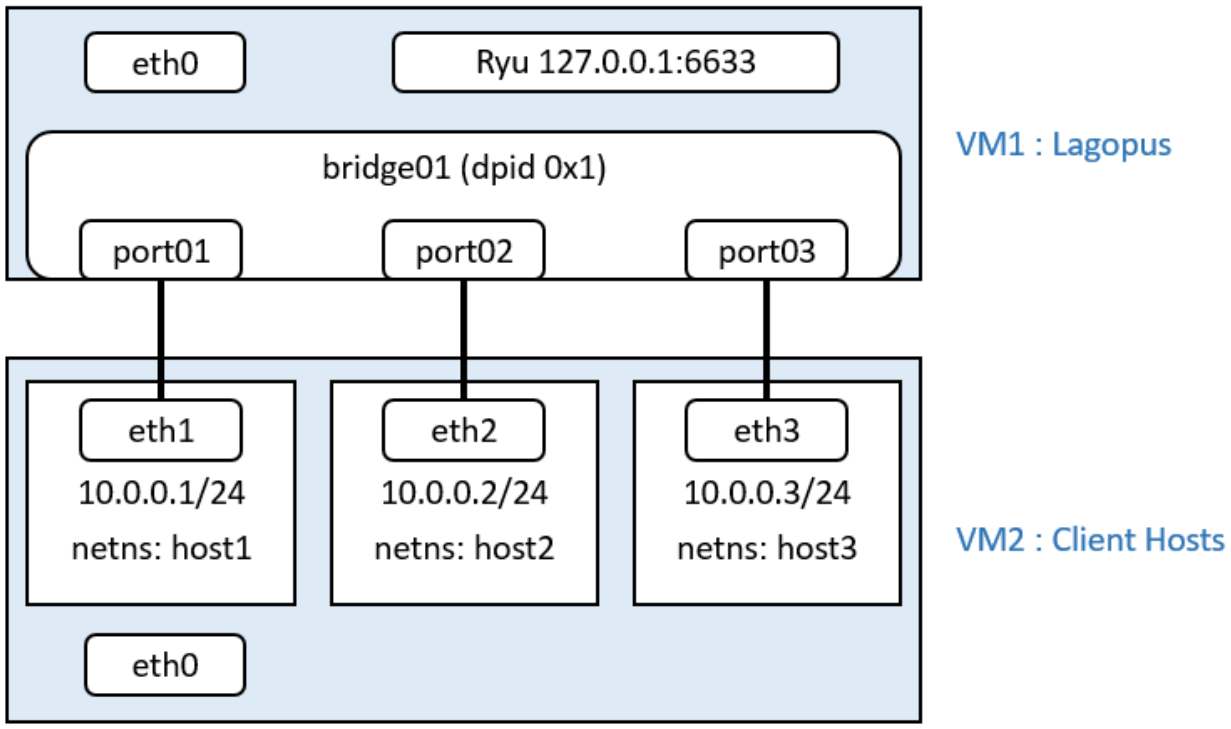
- For more information about Ryu, check https://osrg.github.io/ryu/

9.1 Software Versions

- Lagopus vswitch: Lagopus vswitch 0.2.3
- OS: Linux Ubuntu Server 14.04.3 LTS

9.2 Overall toplogy and client host setup

In this example, we will connect 3 client hosts to Lagopus vswitch bridge ports. You can use 3 physical hosts as client hosts. If you have a Linux host (VM) with 3 or more interfaces, you can either create 3 VMs or create network namespaces to simulate physical hosts.



In this example, we will use network namespace to simulate the client hosts on a single server. You would need at least 2 servers (VMs) for this configuration.

Note:

- Make sure NICs on Lagopus vswitch are running in promiscuous mode.
- eth1 on client host is connected to port01 of Lagopus vswitch, and same for eth2/port02, eth3/port03.
- Use below commands on client host to create 3 network namespaces.

```
host1:
$ sudo ip netns add host1
$ sudo ip link set eth1 netns host1
$ sudo ip netns exec host1 ip link set eth1 up
$ sudo ip netns exec host1 ip addr add 10.0.0.1/24 brd + dev eth1
host2:
$ sudo ip netns add host2
$ sudo ip link set eth2 netns host2
$ sudo ip netns exec host2 ip link set eth2 up
$ sudo ip netns exec host2 ip addr add 10.0.0.2/24 brd + dev eth2
host3:
$ sudo ip netns add host3
$ sudo ip link set eth3 netns host3
$ sudo ip netns exec host3 ip link set eth3 up
$ sudo ip netns exec host3 ip addr add 10.0.0.3/24 brd + dev eth3
```

You can confirm eth1 is only visible from host1 network namespace using below commands.

```
$ sudo ip netns exec host1 ip a
1: lo: <LOOPBACK> mtu 65536 qdisc noop state DOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group
 ↪  default qlen 1000
```

```
    link/ether 08:00:27:24:12:72 brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.1/24 brd 10.0.0.255 scope global eth1
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe24:1272/64 scope link
       valid_lft forever preferred_lft forever
$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group
 ↪  default qlen 1000
    link/ether 08:00:27:a4:17:d0 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fea4:17d0/64 scope link
       valid_lft forever preferred_lft forever
```

9.3 Lagopus vswitch Configuration (raw socket)

- Configure Lagopus vswitch by editing `lagopus.dsl`.
  - Change `-dst-addr 127.0.0.1` if you want to place Ryu on a host different from Lagopus vswitch.
- set interface up using `ip` command.

```
$ sudo vi /usr/local/etc/lagopus/lagopus.dsl
channel channel01 create -dst-addr 127.0.0.1 -protocol tcp
controller controller01 create -channel channel01 -role equal -connection-type main
interface interface01 create -type ethernet-rawsock -device eth1
interface interface02 create -type ethernet-rawsock -device eth2
interface interface03 create -type ethernet-rawsock -device eth3
port port01 create -interface interface01
port port02 create -interface interface02
port port03 create -interface interface03
bridge bridge01 create -controller controller01 -port port01 1 -port port02 2 -port
 ↪  port03 3 -dpid 0x1
bridge bridge01 enable
$ sudo ip link set eth1 up
$ sudo ip link set eth2 up
$ sudo ip link set eth3 up
```

9.4 Lagopus vswitch Configuration (DPDK)

- Configure Lagopus vswitch by editing `lagopus.dsl`.
  - Change `-dst-addr 127.0.0.1` if you want to place Ryu on a host different from Lagopus vswitch.

```
$ sudo vi /usr/local/etc/lagopus/lagopus.dsl
channel channel01 create -dst-addr 127.0.0.1 -protocol tcp
controller controller01 create -channel channel01 -role equal -connection-type main
interface interface01 create -type ethernet-dpdk-phy -port-number 0
```

```
interface interface02 create -type ethernet-dpdk-phy -port-number 1
interface interface03 create -type ethernet-dpdk-phy -port-number 2
port port01 create -interface interface01
port port02 create -interface interface02
port port03 create -interface interface03
bridge bridge01 create -controller controller01 -port port01 1 -port port02 2 -port
 ↪  port03 3 -dpid 0x1
bridge bridge01 enable
```

9.5 Install Ryu and run simple_switch Ryu application

- Install necessary packages
- Use pip command and install Ryu.

```
$ sudo apt-get install python-setuptools python-pip python-dev \
  libxml2-dev libxslt-dev
$ sudo pip install ryu
```

- Run below only if required.

```
$ sudo pip install oslo.config
$ sudo pip install six --upgrade
```

- Run simple_switch Ryu application.

```
$ ryu-manager --verbose
 ↪  /usr/local/lib/python2.7/dist-packages/ryu/app/simple_switch_13.py
```

9.6 Start Lagopus vswitch (raw socket)

- Open new terminal to start Lagopus vswitch.

```
$ sudo lagopus
```

9.7 Start Lagopus vswitch (DPDK)

- Follow instruction in *Lagopus vswitch Installation (DPDK)* to setup DPDK environment.
  - Load the kernel modules.
  - Set up Huge pages.
  - NIC assignment. (With an additional interface, eth3)
- Open new terminal to start Lagopus vswitch.

```
$ sudo lagopus -- -c3 -n1 -- -p7
```

9.8 Confirm simple switch is running

- On termal running ryu, below message will be shown once you start Lagopus vswitch and is connected to Ryu.

```
connected socket:<eventlet.greenio.base.GreenSocket object at 0x7f1de23b61d0>
 ↪  address:('127.0.0.1', 49928)
hello ev <ryu.controller.ofp_event.EventOFPHello object at 0x7f1de23b6810>
move onto config mode
EVENT ofp_event->SimpleSwitch13 EventOFPSwitchFeatures
switch features ev
 ↪  version=0x4,msg_type=0x6,msg_len=0x20,xid=0xc28ef426,OFPSwitchFeatures(auxiliary_id=0,capabiliti
move onto main mode
```

- Ping from host1 to host2 to confirm simple switch is running as expected

```
$ sudo ip netns exec host1 ping 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=10.9 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=2.73 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=2.04 ms
```

- Ryu will show below message when ping is successful.

```
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 1 08:00:27:61:87:73 ff:ff:ff:ff:ff:ff 1
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 1 08:00:27:21:2a:58 08:00:27:61:87:73 2
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 1 08:00:27:61:87:73 08:00:27:21:2a:58 1
```

- Confirm flow Ryu has created using `Lagosh> show flow`.

```
$ lagosh
      Lagosh> show flow
      [
          {
              "tables": [
                  {
                      "table": 0,
                      "flows": [
                          {
                              "stats": {
                                  "packet_count": 4,
                                  "byte_count": 354
                              },
                              "hard_timeout": 0,
                              "actions": [
                                  {
                                      "apply_actions": [
                                          {
                                              "output": 1
                                          }
                                      ]
                                  }
                              ],
                              "priority": 1,
                              "idle_timeout": 0,
                              "cookie": 0,
```

```
                            "dl_dst": "08:00:27:61:87:73",
                            "in_port": 2
                        },
                        {
                            "stats": {
                                "packet_count": 3,
                                "byte_count": 256
                            },
                            "hard_timeout": 0,
                            "actions": [
                                {
                                    "apply_actions": [
                                        {
                                            "output": 2
                                        }
                                    ]
                                }
                            ],
                            "priority": 1,
                            "idle_timeout": 0,
                            "cookie": 0,
                            "dl_dst": "08:00:27:21:2a:58",
                            "in_port": 1
                        },
                        {
                            "stats": {
                                "packet_count": 3,
                                "byte_count": 218
                            },
                            "hard_timeout": 0,
                            "actions": [
                                {
                                    "apply_actions": [
                                        {
                                            "output": "controller"
                                        }
                                    ]
                                }
                            ],
                            "priority": 0,
                            "idle_timeout": 0,
                            "cookie": 0
                        }
                    ]
                }
            ],
            "name": "bridge01",
            "is-enabled": true
        }
    ]
```

CHAPTER

TEN

APPENDIX: INSTALLING LAGOPUS VSWITCH ON VIRTUALBOX VM

This section describes how to install Lagopus on VirtualBox VM.

10.1 Software Versions

- Lagopus vswitch: Lagopus vswitch 0.2.3
- VirtualBox: 5.0.14r105127
- Host OS: Windows 10
- Guest OS: Linux Ubuntu Server 14.04.3 LTS

10.2 About VirtualBox

ORACLE VirtualBox is x86 and AMD64/Intel64 virtualization product which is freely available as Open Source Software under the terms of the GNU General Public License (GPL) version 2.

You could run VirtualBox on host OS running Windows, Linux, Macintosh, and Solaris.

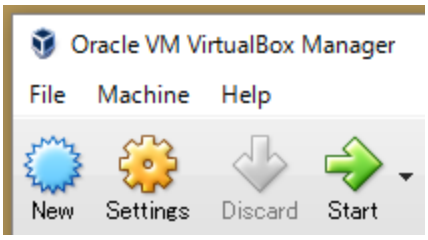
Refer to the VirtualBox web site For more information about VirtualBox.

10.3 Download and Install VirtualBox

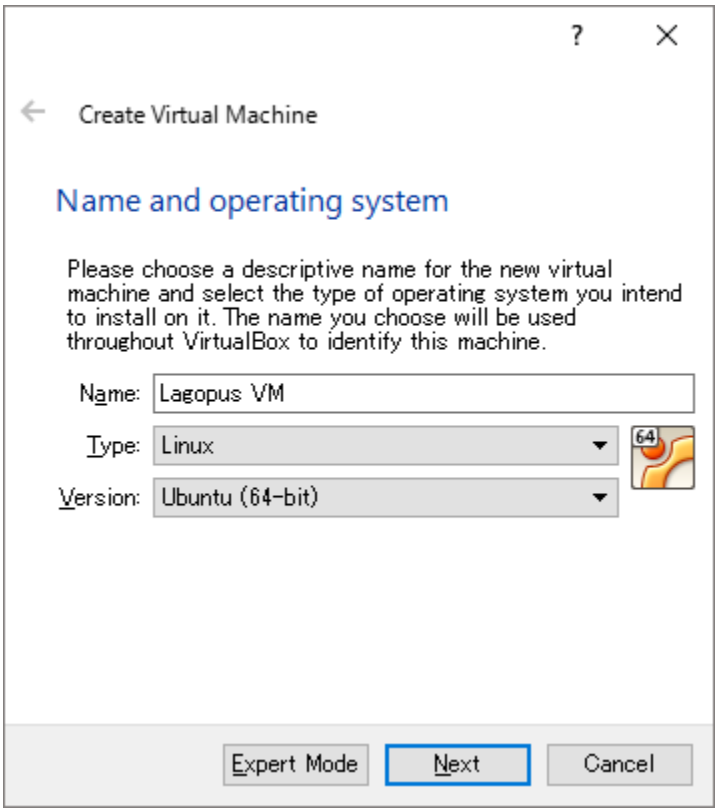
- Download VirtualBox installation image from: https://www.virtualbox.org/wiki/Downloads
- Install VirtualBox.

10.4 Create VM

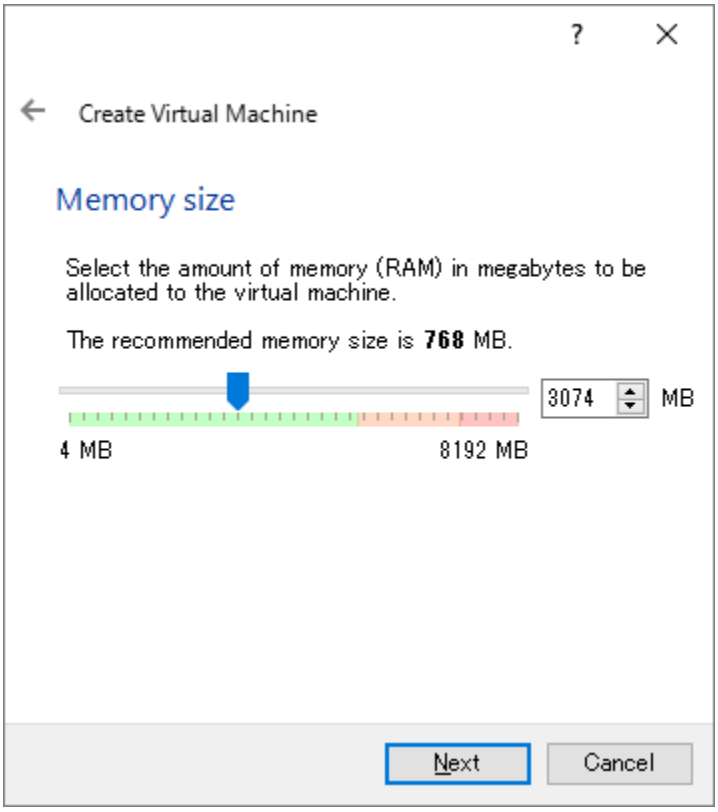
- Start “Oracle VM VirtualBox Manager” (Management GUI)
- Click “New”



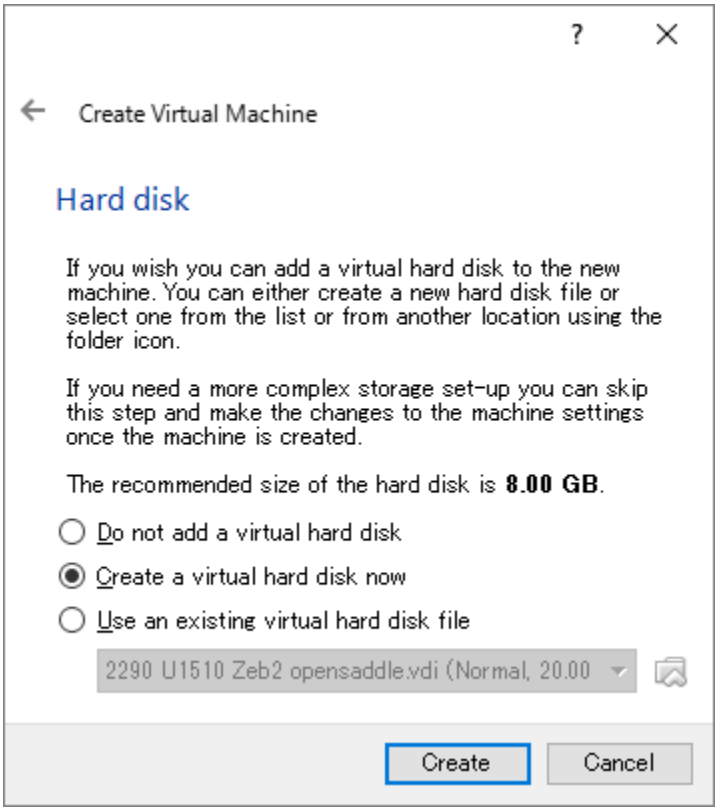
- Enter Name (=any), Type (=Linux), Version (=Ubuntu (64-bit)).



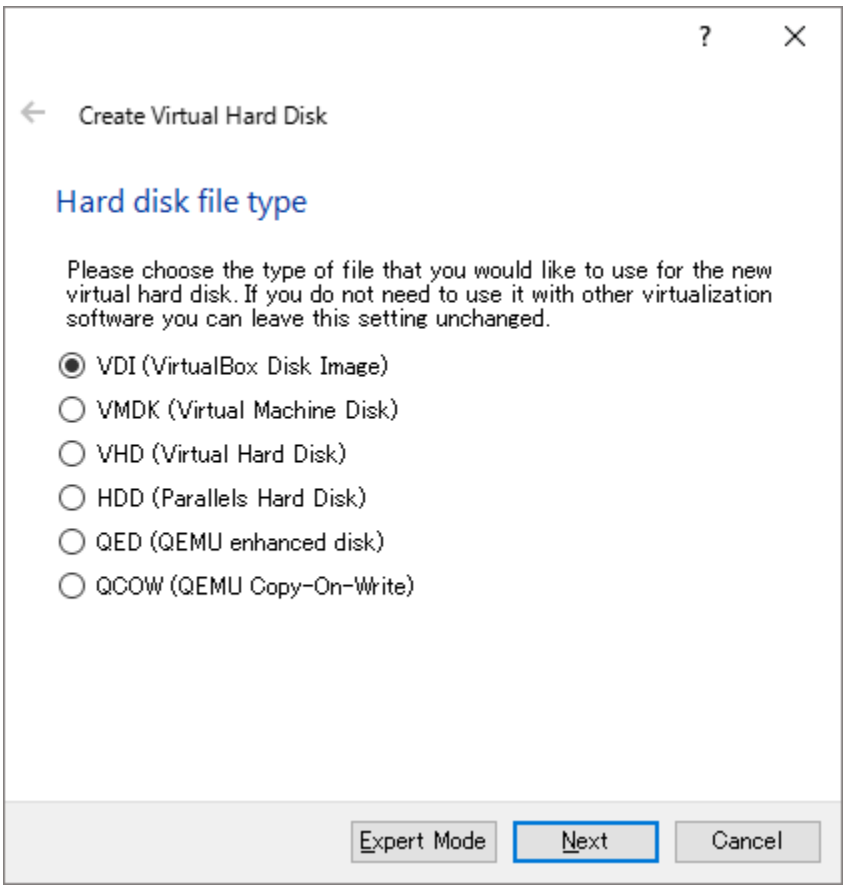
- Set Memroy Size (>=2GB recommended. 3GB in this example.)



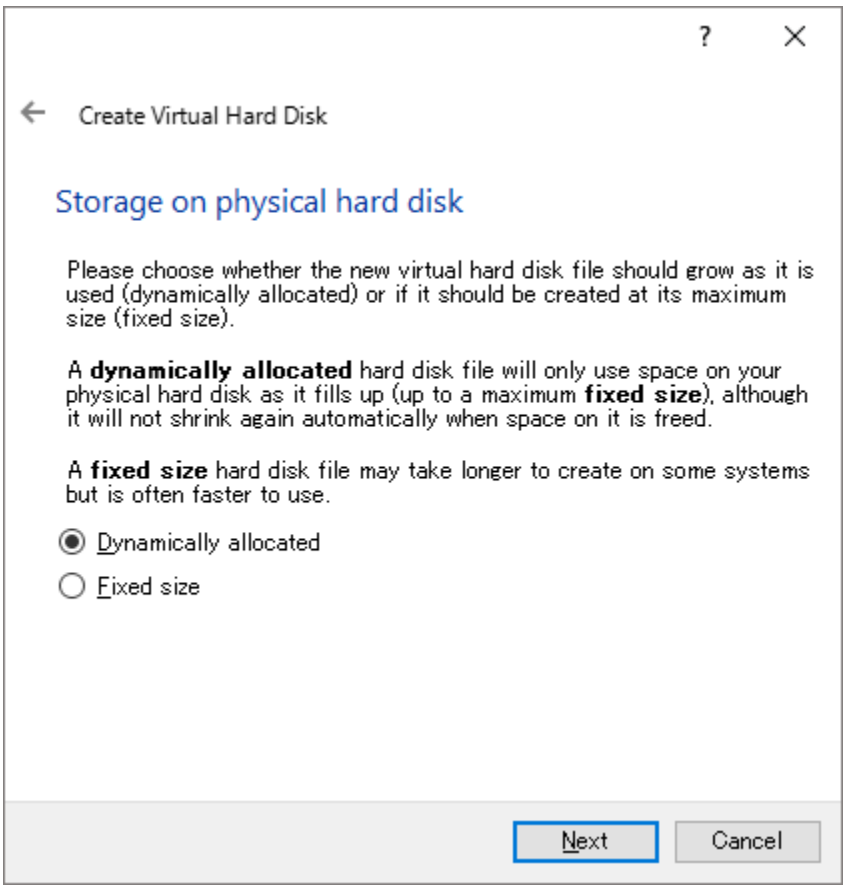
- Set Hard disk size (default = 8.00 GB)



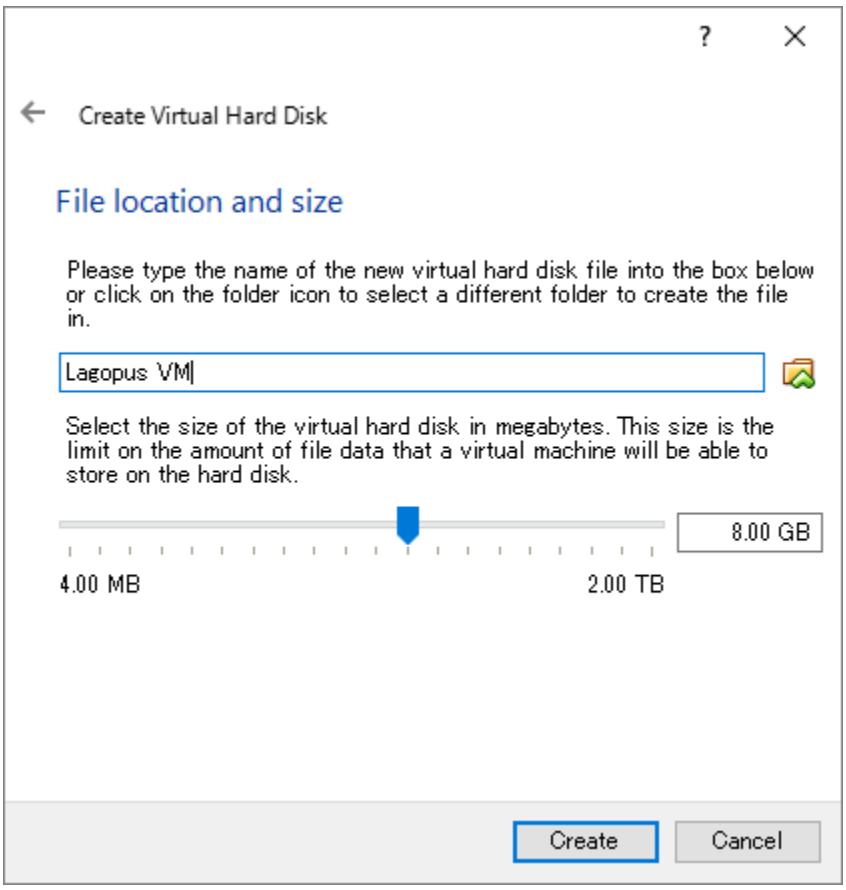
- Set Hard disk file type: default = VDI (VirtualBox Disk Image)



- Set Storage on physical hard disk: default = Dynamically allocated.



- Set File location and size: default folder name = Name of the VM you've specified.



- Click “Create” and VM will be created.

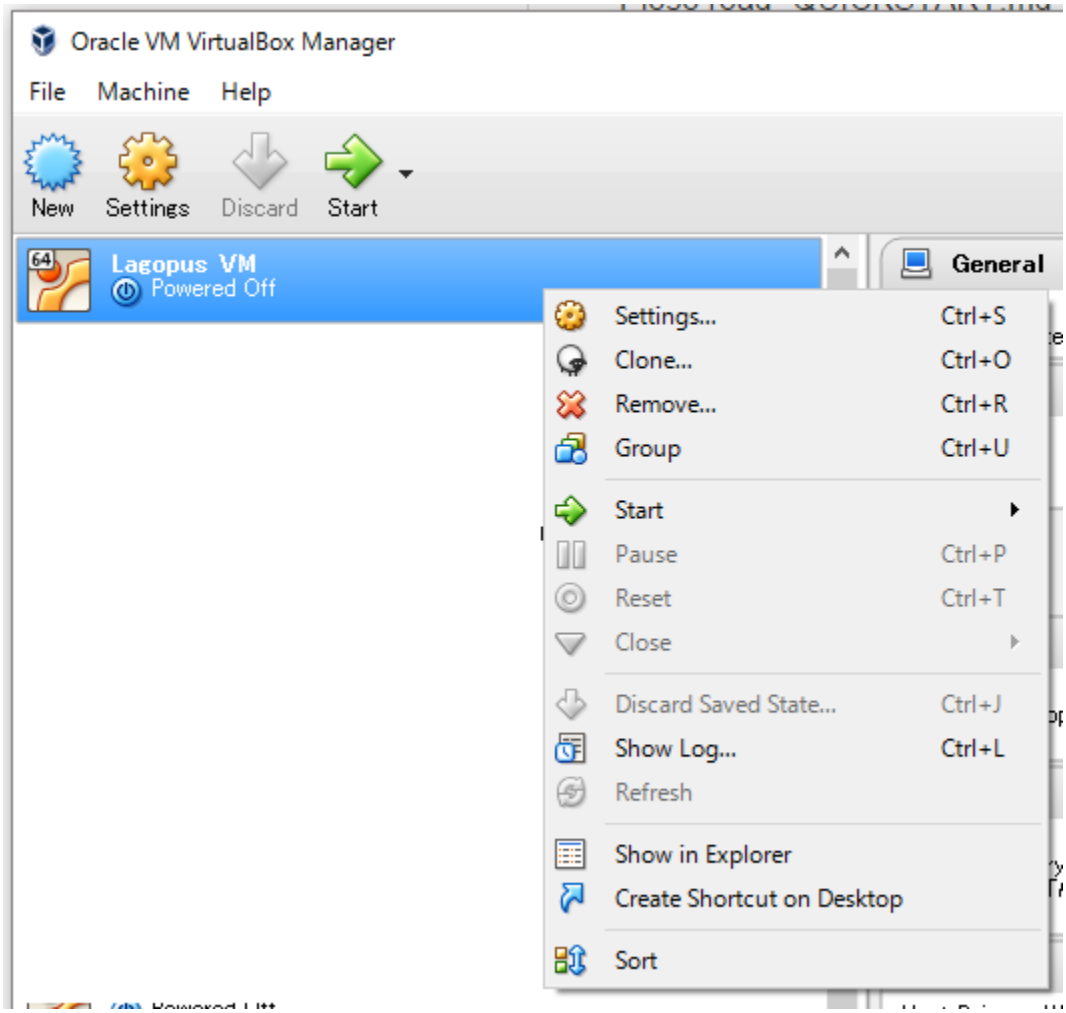
10.5 VM Network Setting

10.5.1 Add two (2) new adapters as data ports used by Lagopus.

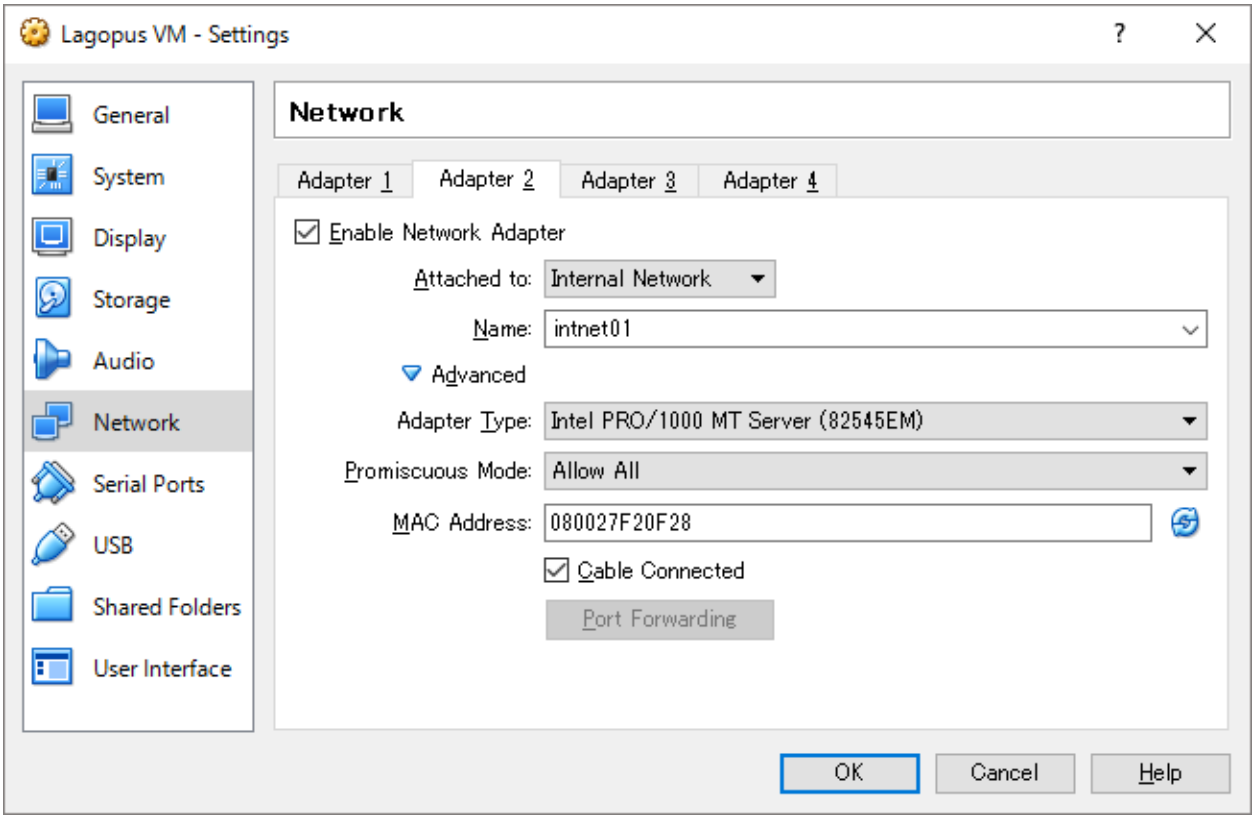
- Adapter 1 (available by default)
  - management port (Internet access, SSH from Host OS)
- Adapter 2/3 (new)
  - Lagopus data ports

Steps:

- Select VM you just created. Right click and select “Settings”



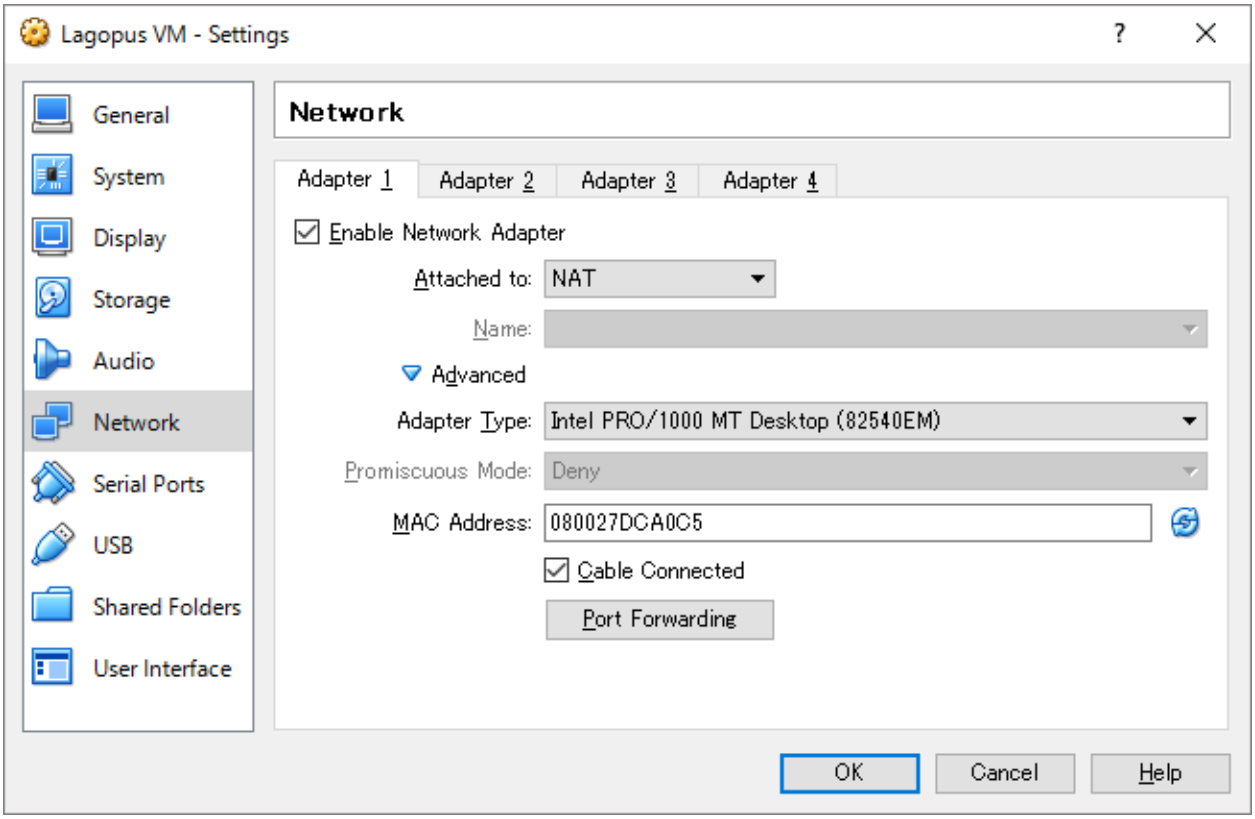
- Select “Network” -> “Adapter 2” and configure below. Do the same for “Adapter 3” as well.
  - Enable Network Adapter: enable
  - Attached to: Internal Network
  - Name: intnet02 (intnet03 for Adapter 3)
  - Advanced: Adapter Type: Intel PRO/1000 MT Server (82545EM)
  - Advanced: Promiscuous Mode: Allow All



Note: NIC supported by DPDK is listed here: http://dpdk.org/doc/nics

10.5.2 Configure "Port Forwarding" to enable SSH from host to guest.

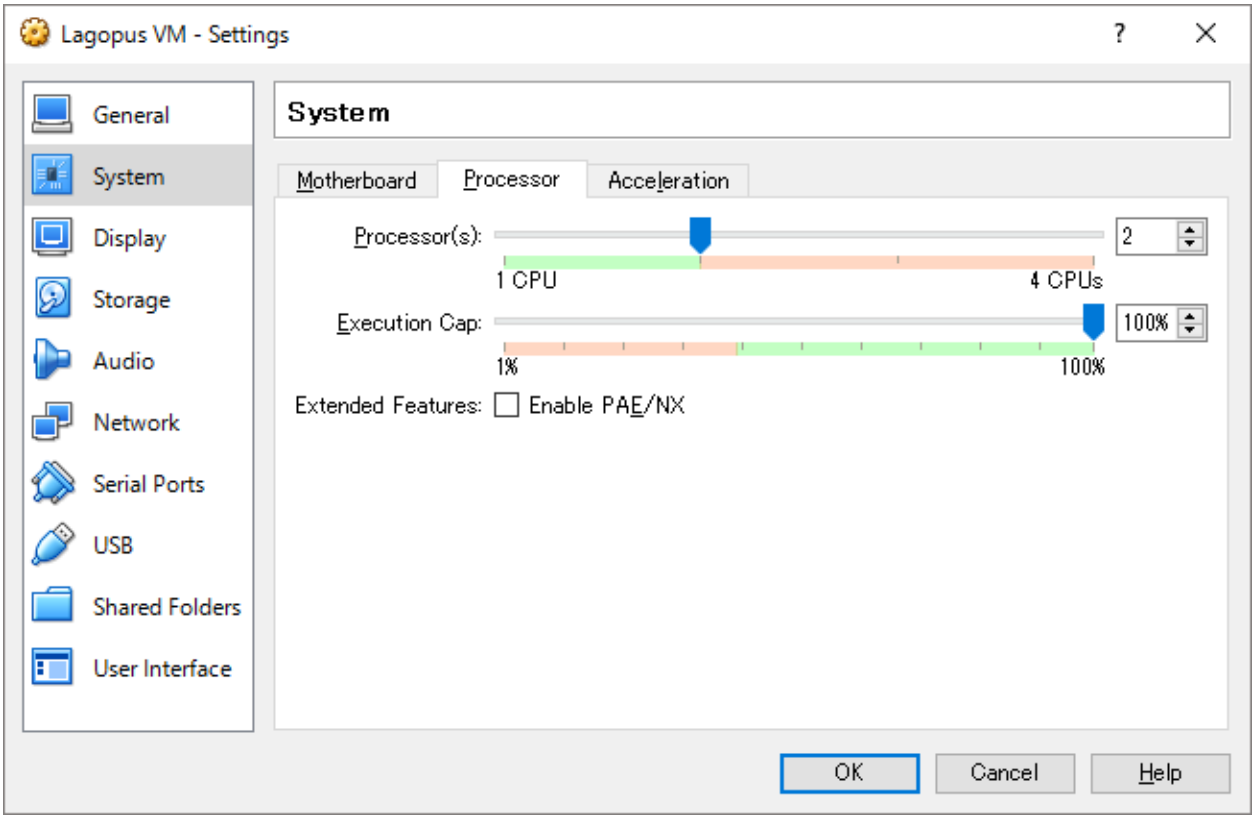
- Select "Network" -> "Adapter 1"
- Click "Advanced: Port Forwarding"



- Click “+” mark on right. Set below to newly created row.
  - Name: rule_name you like. ex: ssh
  - Host Port: port used on Host OS. ex: 2201
  - Guest Port: 22 (SSH)

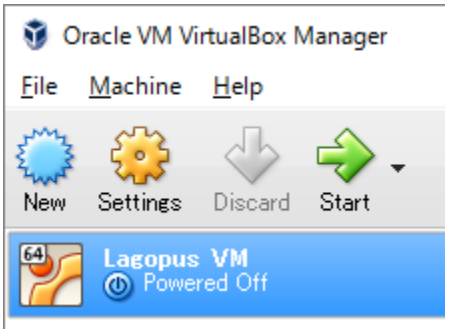
10.6 VM Processor (CPU) Setting

- Select “System” -> Processor” to modify CPU cores to be assigned to the VM.
- 2 or more CPU cores are required when using DPDK.

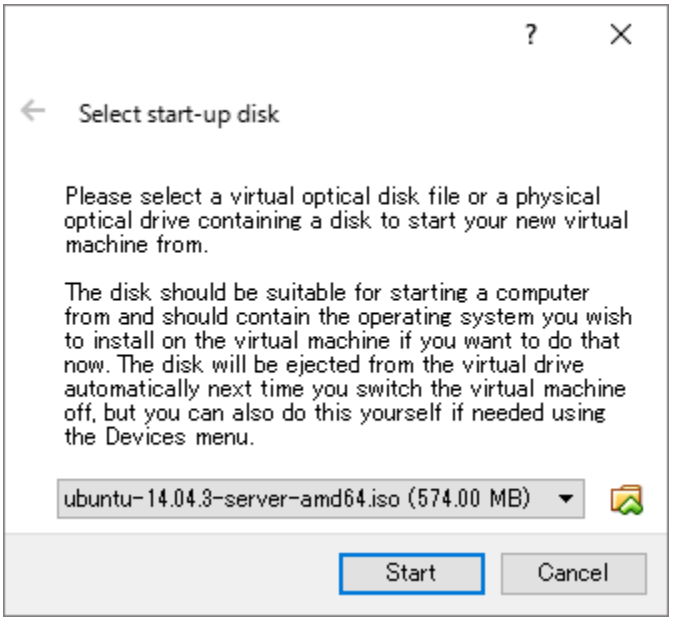


10.7 Install Guest OS (Linux / Ubuntu)

- Download Guest OS ISO:
  - Ubuntu Server 14.04.3 LTS (ubuntu-14.04.3-server-amd64.iso)
- Start VM by clicking "Start"



- At "Select start-up disk" diaglog, select ISO you just downloaded and click "Start"



- Follow Ubuntu install wizard. A few points to be noted are:
  - Select Primary Network Interface: eth0 (Adapter 1)
  - No automatic updates (for testing purpose to make package predictable)
  - Software installation: OpenSSH server

Once installed, SSH to localhost:<port> where <port> is the "Host port" you configured in "Port Forwarding".

10.8 Next Steps

Follow steps in *Lagopus vswitch Installation (raw socket)* or *Lagopus vswitch Installation (DPDK)* to continue Lagopus vswitch installation and configuration.