SDN Fundamentals & Techniques

Chapter 4 - Demo 1 - ONOS and Linux namespace

Jing Yan (yanj3, jing.yan@aalto.fi) Feb 18th, 2020

Environment Description

I created ONOS VM and Mininet VM with the bridge network mode on my local physical machine, displaying ip info of ONOS VM, Mininet VM and my physical local machine as follows:

- Local Physical Machine:

```
yanjing@yanjingdeMacBook-Pro ~ % ifconfig en0 | grep inet
   inet 192.168.1.101
```

- Mininet VM based on VirtualBox:

```
root@mininet-vm:~# ifconfig -a | grep inet
       inet 192.168.1.2
```

- ONOS VM based on VirtualBox:

```
root@jingyan:/home/jingyan/onos# ifconfig -a | grep inet
       inet 192.168.1.3

In the 1st terminal:
root@jingyan:/home/jingyan/onos# bazel run onos-local clean debug

In the 2nd terminal:
root@jingyan:/home/jingyan/onos# ./tools/test/bin/onos localhost
```

Task 1.1

Create a linear topology of 5 switches and 5 hosts. In a linear topology, each switch has two connections with other switches except the first and the last ones. To do that you may write a shell script or a python-based code, your code must be well-commented and follow coding principles and naming convention (check the basic-net-ns.sh to get inspired).

Code and Execution

Note: The following code is not screenshot and can be seen in the attachment: chapter4_demo1/chapter4_demo1_task1.1.sh

```
#!/usr/bin/env bash
# Create host namespace h1-h5 and then print
echo "--- Create host namespace h1-h5 --- "
for i in 1 2 3 4 5
do
  ip netns add h$i
done
echo "--- Print host namespaces --- "
ip netns
# Create switch s1-s5 and then print
echo "--- Create switch s1-s5 --- "
for i in 1 2 3 4 5
do
  ovs-vsctl add-br s$i
  ovs-vsctl set bridge s$i protocols=OpenFlow13
done
echo "--- Print switches --- "
ovs-vsctl list-br
# Create links and then print
echo "--- Create links--- "
for i in 1 2 3 4 5
do
  ip link add h$i-eth0 type veth peer name s$i-eth1;
done
ip link add s2-eth2 type veth peer name s1-eth2
ip link add s3-eth2 type veth peer name s2-eth3
ip link add s4-eth2 type veth peer name s3-eth3
ip link add s5-eth2 type veth peer name s4-eth3
echo "--- Print links --- "
ip link show
# Move host ports into namespaces
echo "--- Move host ports into namespaces--- "
for i in 1 2 3 4 5
do
  ip link set h$i-eth0 netns h$i;
done
# Connect switch ports to OVS and then print
echo "--- Connect switch ports to OVS--- "
ovs-vsctl add-port s1 s1-eth1
ovs-vsctl add-port s1 s1-eth2
ovs-vsctl add-port s5 s5-eth1
ovs-vsctl add-port s5 s5-eth2
for i in 2 3 4
do
  for j in 1 2 3
  do
      ovs-vsctl add-port s$i s$i-eth$j
  done
done
for i in 1 2 3 4 5
do
  echo "--- Print ports of s$i--- "
```

```
  ovs-vsctl list-ports s$i
done
# Setup ONOS controller and then print
echo "--- Setup ONOS controller--- "
ovs-vsctl set-controller s1 tcp:192.168.1.3:9876
ovs-vsctl set-controller s2 tcp:192.168.1.3:9876
ovs-vsctl set-controller s3 tcp:192.168.1.3:9876
ovs-vsctl set-controller s4 tcp:192.168.1.3:9876
ovs-vsctl set-controller s5 tcp:192.168.1.3:9876
echo "--- Print OVS database contents--- "
ovs-vsctl show
# Setup networks for hosts and switches, and then print
echo "--- Setup networks for hosts and switches--- "
for i in 1 2 3 4 5
do
  ip netns exec h$i ifconfig h$i-eth0 10.0.0.$i/24
  ip netns exec h$i ifconfig lo up
  ip netns exec h$i ifconfig h$i-eth0 up
done
for i in 2 3 4
do
  for j in 1 2 3
  do
      ifconfig s$i-eth$j up
  done
done
ifconfig s1-eth1 up
ifconfig s1-eth2 up
ifconfig s5-eth1 up
ifconfig s5-eth2 up
for i in 1 2 3 4 5
do
  echo "--- Print ip info of h$i--- "
  ip netns exec h$i ifconfig h$i-eth0
done
```

```
# sh chapter4_demo1_task1.1.sh > chapter4_demo1_task1.1.log
```

Test Code and Execution

Note: The following file is not screenshot can be seen in the attachment: chapter4_demo1/chapter4_demo1_task1.1_test.sh

```
#!/usr/bin/env bash
# Test networks
for i in 1 2 3 4 5
do
   for j in 1 2 3 4 5
   do
       echo "-----FROM h$i ping h$j-----"
       ip netns exec h$i ping -c1 10.0.0.$j
   done
done
```

```
# sh chapter4_demo1_task1.1_test.sh >> chapter4_demo1_task1.1.log
```

Result

From the “Code and Execution” and “Test Code and Execution” parts, all the results are redirected to “chapter4_demo1_task1.1.log” file. I will display part of the results here

Note: More info can seen in the attachment: chapter4_demo1/chapter4_demo1_task1.1.log

- All the namespaces (h1-h5) are created successfully
- All the switches (s1-s5) are created successfully
- All the links are created successfully, including the following and the inverse direction

    - s1-eth1@h1-eth0, s2-eth1@h2-eth0, s3-eth1@h3-eth0, s4-eth1@h4-eth0, s5-eth1@h5-eth0
    - s1-eth2@s2-eth2, s2-eth3@s3-eth2, s3-eth3@s4-eth2, s4-eth3@s5-eth2
- Ports of the switches created successfully
    - s1-eth1, s1-eth2, s2-eth1, s2-eth2, s2-eth3, s3-eth1, s3-eth2, s3-eth3, s4-eth1, s4-eth2, s4-eth3, s5-eth1, s5-eth2
- ONOS controller is connected successfully
    - 0ef0158d-89b3-4ce0-9a70-3cbf41ab89ff
    - Manager "tcp:192.168.1.3:9876"
    - is_connected: true
- Interfaces of hosts are added to the corresponding namespaces successfully and ips are assigned correctly
    - h1-eth0: 10.0.0.1/255.255.255.0
    - h2-eth0: 10.0.0.2/255.255.255.0
    - h3-eth0: 10.0.0.3/255.255.255.0
    - h4-eth0: 10.0.0.4/255.255.255.0
    - h5-eth0: 10.0.0.5/255.255.255.0
- Ping cmds succeeded among all hosts
    - h1 can ping h1, h2, h3, h4, h5 successfully
    - h2 can ping h1, h2, h3, h4, h5 successfully
    - h3 can ping h1, h2, h3, h4, h5 successfully
    - h4 can ping h1, h2, h3, h4, h5 successfully
    - h5 can ping h1, h2, h3, h4, h5 successfully

Task 1.2

Create a tree topology depth 3 fanout 2. In a tree topology, the depth refers to the number of layers in a tree and the fanout is the number of children each node has. For instance, in a tree where depth and fanout are equal to 2, the tree will have a node 0, i.e., switch 0, that has two other switches connected to it, switches 1 and 2.

Code and Execution

Note: The following file is not screenshot can be seen in the attachment: chapter4_demo1/chapter4_demo1_task1.2.sh

```
#!/usr/bin/env bash
# Create host namespace h1-h8 and then print
echo "--- Create host namespace h1-h8 --- "
for i in 1 2 3 4 5 6 7 8
do
   ip netns add h$i
done
echo "--- Print host namespaces --- "
ip netns

# Create switch s1-s8 and then print
echo "--- Create switch s1-s5 --- "
for i in 1 2 3 4 5 6 7
Do
   ovs-vsctl add-br s$i
   ovs-vsctl set bridge s$i protocols=OpenFlow13
```

```
done
echo "--- Print switches --- "
ovs-vsctl list-br

# Create links and then print
echo "--- Create links--- "
ip link add s1-eth1 type veth peer name s2-eth3
ip link add s1-eth2 type veth peer name s5-eth3
ip link add s2-eth1 type veth peer name s3-eth3
ip link add s2-eth2 type veth peer name s4-eth3
ip link add s5-eth1 type veth peer name s6-eth3
ip link add s5-eth2 type veth peer name s7-eth3

ip link add s3-eth1 type veth peer name h1-eth0
ip link add s3-eth2 type veth peer name h2-eth0
ip link add s4-eth1 type veth peer name h3-eth0
ip link add s4-eth2 type veth peer name h4-eth0
ip link add s6-eth1 type veth peer name h5-eth0
ip link add s6-eth2 type veth peer name h6-eth0
ip link add s7-eth1 type veth peer name h7-eth0
ip link add s7-eth2 type veth peer name h8-eth0
echo "--- Print links --- "
ip link show

# Move host ports into namespaces
echo "--- Move host ports into namespaces--- "
for i in 1 2 3 4 5 6 7 8
do
   ip link set h$i-eth0 netns h$i;
done

# Connect switch ports to OVS and then print
echo "--- Connect switch ports to OVS--- "
ovs-vsctl add-port s1 s1-eth1
ovs-vsctl add-port s1 s1-eth2

for i in 2 3 4 5 6 7
do
   for j in 1 2 3
   do
       ovs-vsctl add-port s$i s$i-eth$j
   done
done

for i in 1 2 3 4 5 6 7
do
   echo "--- Print ports of s$i--- "
   ovs-vsctl list-ports s$i
done

# Setup ONOS controller
echo "--- Setup ONOS controller--- "
ovs-vsctl set-controller s1 tcp:192.168.1.3:9876
ovs-vsctl set-controller s2 tcp:192.168.1.3:9876
ovs-vsctl set-controller s3 tcp:192.168.1.3:9876
ovs-vsctl set-controller s4 tcp:192.168.1.3:9876
ovs-vsctl set-controller s5 tcp:192.168.1.3:9876
ovs-vsctl set-controller s6 tcp:192.168.1.3:9876
ovs-vsctl set-controller s7 tcp:192.168.1.3:9876
echo "--- Print OVS database contents--- "
ovs-vsctl show

# Setup networks for hosts and switches, and then print
echo "--- Setup networks for hosts and switches--- "
for i in 1 2 3 4 5 6 7 8
do
   ip netns exec h$i ifconfig h$i-eth0 10.0.0.$i/24
   ip netns exec h$i ifconfig lo up
```

```
   ip netns exec h$i ifconfig h$i-eth0 up
done

ifconfig s1-eth1 up
ifconfig s1-eth2 up
for i in 2 3 4 5 6 7
do
   for j in 1 2 3
   do
       ifconfig s$i-eth$j up
   done
done

for i in 1 2 3 4 5 6 7 8
do
   echo "--- Print ip info of h$i--- "
   ip netns exec h$i ifconfig h$i-eth0
done
```

```
sh chapter4_demo1_task1.2.sh > chapter4_demo1_task1.2.log
```

Test Code and Execution

Note: The following file can be seen in the attachment: chapter4_demo1/chapter4_demo1_task1.2_test.sh

```
#!/usr/bin/env bash
# Test networks
for i in 1 2 3 4 5 6 7 8
do
   for j in 1 2 3 4 5 6 7 8
   do
       echo "-----FROM h$i ping h$j-----"
       ip netns exec h$i ping -c1 10.0.0.$j
   done
done
```

```
sh chapter4_demo1_task1.2_test.sh >> chapter4_demo1_task1.2.log
```

Result

From the “Code and Execution” and “Test Code and Execution” parts, all the results are redirected to “chapter4_demo1_task1.1.log” file. I will display part of the results here

Note: More info can seen in the attachment: chapter4_demo1/chapter4_demo1_task1.2.log

- All the namespaces (h1-h8) are created successfully
- All the switches (s1-s7) are created successfully
- All the links are created successfully, including the following and the inverse direction
    - s2-eth3@s1-eth1, s5-eth3@s1-eth2, s3-eth3@s2-eth1, s4-eth3@s2-eth2, s6-eth3@s5-eth1, s7-eth3@s5-eth2
    - s3-eth1@h1-eth0, s3-eth2@h2-eth0, s4-eth1@h3-eth0, s4-eth2@h4-eth0, s6-eth1@h5-eth0, s6-eth2@h6-eth0, s7-eth1@h7-eth0, s7-eth2@h8-eth0,
- Ports of the switches created successfully
    - s1-eth1, s1-eth2, s2-eth1, s2-eth2, s2-eth3, s3-eth1, s3-eth2, s3-eth3, s4-eth1, s4-eth2, s4-eth3, s5-eth1, s5-eth2, s5-eth3, s6-eth1, s6-eth2, s6-eth3, s7-eth1, s7-eth2, s7-eth3
- ONOS controller is connected successfully
    - 0ef0158d-89b3-4ce0-9a70-3cbf41ab89ff
    - Manager "tcp:192.168.1.3:9876"
    - is_connected: true

- Interfaces of hosts are added to the corresponding namespaces successfully and ips are assigned correctly
    - h1-eth0: 10.0.0.1/255.255.255.0
    - h2-eth0: 10.0.0.2/255.255.255.0
    - h3-eth0: 10.0.0.3/255.255.255.0
    - h4-eth0: 10.0.0.4/255.255.255.0
    - h5-eth0: 10.0.0.5/255.255.255.0
    - h6-eth0: 10.0.0.6/255.255.255.0
    - h7-eth0: 10.0.0.7/255.255.255.0
    - h8-eth0: 10.0.0.8/255.255.255.0
- Ping cmds succeeded among all hosts
    - h1 can ping h1, h2, h3, h4, h5, h6, h7, h8 successfully
    - h2 can ping h1, h2, h3, h4, h5, h6, h7, h8 successfully
    - h3 can ping h1, h2, h3, h4, h5, h6, h7, h8 successfully
    - h4 can ping h1, h2, h3, h4, h5, h6, h7, h8 successfully
    - h5 can ping h1, h2, h3, h4, h5, h6, h7, h8 successfully
    - h6 can ping h1, h2, h3, h4, h5, h6, h7, h8 successfully
    - h7 can ping h1, h2, h3, h4, h5, h6, h7, h8 successfully
    - h8 can ping h1, h2, h3, h4, h5, h6, h7, h8 successfully

Task 1.3

Try to automate the precedent tasks to be able to create a topology given the type, i.e., tree or linear, and for each type its specifications (i.e., depth and fanout for the tree and number of switches for linear). It is worth noticing that you can use Python as it is preferred because of the available tools that allow the exploitation of networking concepts.

Automation and Network Test Code for Linear Topo and Execution

Note: The following file is not screenshot can be seen in the attachment: chapter4_demo1/chapter4_demo1_task1.3_linear.py

```
import os
from pyroute2 import netns
from pyroute2 import IPRoute

# Get access to the netlink socket
ip = IPRoute()

# Create namespaces based on the parameter and list hosts
def create_namespaces(host_num):
    for i in range(1, host_num+1):
        netns.create('h%s' % i)
    print(netns.listnetns())

# Create OVS based on the parameter and list OVS
def create_ovs(ovs_num):
    ovs_name_list = []
    for i in range(1, ovs_num+1):
        ovs_name_list.append('s%s' % i)
    for i in range(0, len(ovs_name_list)):
        cmd_create = 'ovs-vsctl add-br {}'.format(ovs_name_list[i])
```

```
                        cmd_set_pro    =    'ovs-vsctl    set    bridge    {}
protocols=OpenFlow13'.format(ovs_name_list[i])
        os.system(cmd_create)
        os.system(cmd_set_pro)
    os.system('ovs-vsctl list-br')

# Creat linear topo based on the parameter
def create_linear_topo(linear_host_num):
    global ip
    create_namespaces(linear_host_num)
    create_ovs(linear_host_num)

    # Create links for hosts and OVS
    for i in range(1, linear_host_num+1):
        os.system('ip link add h%s-eth0 type veth peer name s%s-eth1' % (i, i))

    # Create links for OVS and OVS
    ip.link('add', ifname='s1-eth2', peer='s2-eth2', kind='veth')
    for i in range(2, linear_host_num):
            ip.link('add',  ifname=  's%s-eth3'  %  i,  peer=  's%s-eth2'  %  (i+1),
kind='veth')

    # Move host ports into namespace
    for i in range(1, linear_host_num+1):
        os.system('ip link set h%s-eth0 netns h%s' % (i, i))

    # Connect switch ports to OVS
    os.system('ovs-vsctl add-port s1 s1-eth1')
    os.system('ovs-vsctl add-port s1 s1-eth2')
    os.system('ifconfig s1-eth1 up')
    os.system('ifconfig s1-eth2 up')
         os.system('ovs-vsctl   add-port   s%s   s%s-eth1'   %   (linear_host_num,
linear_host_num))
        os.system('ovs-vsctl   add-port   s%s   s%s-eth2'   %   (linear_host_num,
linear_host_num))
    os.system('ifconfig s%s-eth1 up' % (linear_host_num))
    os.system('ifconfig s%s-eth2 up' % (linear_host_num))
    for i in range(2, linear_host_num):
        os.system('ovs-vsctl add-port s%s s%s-eth1' % (i, i))
        os.system('ovs-vsctl add-port s%s s%s-eth2' % (i, i))
        os.system('ovs-vsctl add-port s%s s%s-eth3' % (i, i))
        os.system('ifconfig s%s-eth1 up' % i)
        os.system('ifconfig s%s-eth2 up' % i)
        os.system('ifconfig s%s-eth3 up' % i)

    # List OVS ports
    for i in range(1, linear_host_num+1):
        os.system('ovs-vsctl list-ports s%s' % i)

    # Setup ONOS
    for i in range(1, linear_host_num+1):
        os.system('ovs-vsctl set-controller s%s tcp:192.168.1.3:9876' % i)
    os.system('ovs-vsctl show')

    # Setup networks for hosts and switches, and then print
    for i in range(1, linear_host_num+1):
          os.system('ip netns exec h%s ifconfig h%s-eth0 10.0.0.%s/24' % (i, i,
i))
        os.system('ip netns exec h%s ifconfig lo up' % i)
        os.system('ip netns exec h%s ifconfig h%s-eth0 up' % (i, i))

# Test linear topo network based on the parameter
def test_linear_topo_network(linear_host_num):
    os.system('sleep 20')
    for i in range(1, linear_host_num+1):
        for j in range(1, linear_host_num+1):
            os.system("echo 'from h%s ping h%s'" % (i, j))
            os.system('ip netns exec h%s ping -c1 10.0.0.%s' % (i, j))
```

```
if __name__ == '__main__':
    linear_host_num = raw_input("Pls input switch/host num [2,254] for linear
topo: ")
   create_linear_topo(int(linear_host_num))
   test_linear_topo_network(int(linear_host_num))
```

```
# python chapter4_demo1_task1.3_linear.py
Pls input switch/host num [2,254] for linear topo: 7
```

Result for Linear Topo Automation and Network Test

From the last part, I collected all the results to the “chapter4_demo1_task1.3_linear.log” file. I will display part of the results here.

Note: More info can seen in the attachment: chapter4_demo1/chapter4_demo1_task1.3_linear.log

```
# cat chapter4_demo1_task1.3_linear.log
# python chapter4_demo1_task1.3_linear.py
Pls input switch/host num [2,254] for linear topo: 7
['h7', 'h6', 'h5', 'h4', 'h3', 'h2', 'h1']
s1
s2
s3
s4
s5
s6
s7
s1-eth1
s1-eth2
s2-eth1
s2-eth2
s2-eth3
s3-eth1
s3-eth2
s3-eth3
s4-eth1
s4-eth2
s4-eth3
s5-eth1
s5-eth2
s5-eth3
s6-eth1
s6-eth2
s6-eth3
s7-eth1
S7-eth2
...
```

- All the namespaces (h1-h7) are created successfully
- All the switches (s1-s7) are created successfully
- All the links are created successfully, including hosts and switches, switches and switches
- Ports of the switches and hosts created successfully
- ONOS controller is connected successfully
    - 0ef0158d-89b3-4ce0-9a70-3cbf41ab89ff
    - Manager "tcp:192.168.1.3:9876"
    - is_connected: true
- Interfaces of hosts are added to the corresponding namespaces successfully and ips are assigned correctly
    - h1-eth0: 10.0.0.1/255.255.255.0

    - ...
- Ping cmds succeeded among all hosts
    - {h1-h7} can ping {h1-h7} successfully

Automation and Network Test Code for Tree Topo and Execution

```
import  os
s_count = 0
h_count = 0

h_list = []
s_list = []

def run_command(c_str):
   print(c_str)
   os.system(c_str)

class host:
   def __init__(self,name):
       global h_list
       print("# Create host namespace h* and then print")
       run_command("ip netns add h%s" % name)
       self.name = "h%s" % name
       self.link_count = 0
       h_list.append(self)

def createNameSpaceAndHost():
   global h_count
   h_count = h_count+1
   return host(h_count)

class switch:
   def __init__(self,name):
       global s_list
       print("# Create switch s* and then print")
       run_command("ovs-vsctl add-br s%s" % name)
       self.name = "s%s" % name
       self.link_count = 0
       s_list.append(self)

def createSwitch():
   global s_count
   s_count = s_count + 1
   return switch(s_count)

def link(a, b):
   run_command("# Create link between %s and %s and then print" % (a.name, b.name))
   run_command("ip link add %s-eth%s type veth peer name %s-eth%s" % (a.name,
a.link_count, b.name, b.link_count))
   a.link_count = a.link_count + 1
   b.link_count = b.link_count + 1

def add(cur_node,stage,node_size):
```

```
    if stage == 0:
        for idx in range(node_size):
            host = createNameSpaceAndHost()
            link(cur_node,host)
    if stage > 0:
        for idx in range(node_size):
            s = createSwitch()
            cur = add(s,stage-1,node_size)
            link(cur_node,s)
        print(cur_node)

if __name__ == '__main__':
    tree_depth = raw_input("Pls input depth for tree topo: ")
    tree_fanout = raw_input("Pls input fanout for tree topo: ")
    s1 = createSwitch()
    add(s1, int(tree_depth) - 1, int(tree_fanout))

    print("# Move host ports into namespaces")
    for item in h_list:
        run_command("ip link set %s-eth0 netns %s"%(item.name,item.name))

    print("# Connect switch ports to OVS and then print")
    for item in s_list:
        for idx in range(item.link_count):
            run_command("ovs-vsctl add-port %s %s-eth%s"%(item.name,item.name,idx))

    print("# Setup ONOS controller")
    for item in s_list:
        run_command("ovs-vsctl set-controller %s tcp:192.168.1.3:9876"%item.name)

    print("# Setup networks for hosts and switches and then print")
    for idx in range(len(h_list)):
        run_command(" ip netns exec %s ifconfig %s-eth0
10.0.0.%s/24"%(h_list[idx].name,h_list[idx].name,idx+1))
        run_command(" ip netns exec %s ifconfig lo up"%(h_list[idx].name))
        run_command(" ip netns exec %s ifconfig %s-eth0
up"%(h_list[idx].name,h_list[idx].name))

    for item in s_list:
        for idx in range(item.link_count):
            run_command("ifconfig %s-eth%s up"%(item.name,idx))

    for item in h_list:
        print("# Print ip info of %s",item.name)
        run_command("ip netns exec %s ifconfig %s-eth0"%(item.name,item.name))

    os.system("sleep 20")
    print("# Test network")
    for i in range(1, len(h_list)+1):
        for j in range(1, len(h_list)+1):
            os.system("echo 'from h%s ping h%s'" % (i, j))
            os.system('ip netns exec h%s ping -c1 10.0.0.%s' % (i, j))
```

```
# python chapter4_demo1_task1.3_tree.py
Pls input depth for tree topo: 2
Pls input fanout for tree topo: 4
```

Result for Tree Topo Automation and Network Test

From the last part, I collected all the results to the “chapter4_demo1_task1.3_tree.log” file. I will display part of the results here.

Note: More info can seen in the attachment: chapter4_demo1/chapter4_demo1_task1.3_tree.log

```
# python chapter4_demo1_task1.3_tree.py
Pls input depth for tree topo: 2
Pls input fanout for tree topo: 4
# Create switch s* and then print
ovs-vsctl add-br s1
# Create switch s* and then print
ovs-vsctl add-br s2
# Create host namespace h* and then print
ip netns add h1
# Create link between s2 and h1 and then print
ip link add s2-eth0 type veth peer name h1-eth0
# Create host namespace h* and then print
ip netns add h2
# Create link between s2 and h2 and then print
ip link add s2-eth1 type veth peer name h2-eth0
# Create host namespace h* and then print
ip netns add h3
```

- All the namespaces (h1-h16) are created successfully
- All the switches (s1-s5) are created successfully
- All the links are created successfully, including hosts and switches, switches and switches
- Ports of the switches and hosts created successfully
- ONOS controller is connected successfully
    - 0ef0158d-89b3-4ce0-9a70-3cbf41ab89ff
    - Manager "tcp:192.168.1.3:9876"
    - is_connected: true
- Interfaces of hosts are added to the corresponding namespaces successfully and ips are assigned correctly
    - h1-eth0: 10.0.0.1/255.255.255.0
    - …
    - h16-eth0: 10.0.0.16/255.255.255.0
- Ping cmds succeeded among all hosts
    - {h1-h16} can ping {h1-h16} successfully