[토막] Network Namespace 간단한 구성

기준 : CentOS 7.6.1810
유저 : root 사용자

Network Namespace

가상화 또는 클러스터 시스템에서 자주 보이는 용어입니다.
앞서서 컨테이너를 사용하기 위한 격리 조건 중 하나로 언급이 되었었는데요.
논리적인 Network Space(공간)을 생성 후 네트워크 인터페이스, 라우팅, IP 라는 네트워크 요소들을 넣어
Host의 네트워크와 격리되는 공간을 말합니다. 다양한 환경의 네트워크들을 구분하고 통신하기 위해 활용됩니다.

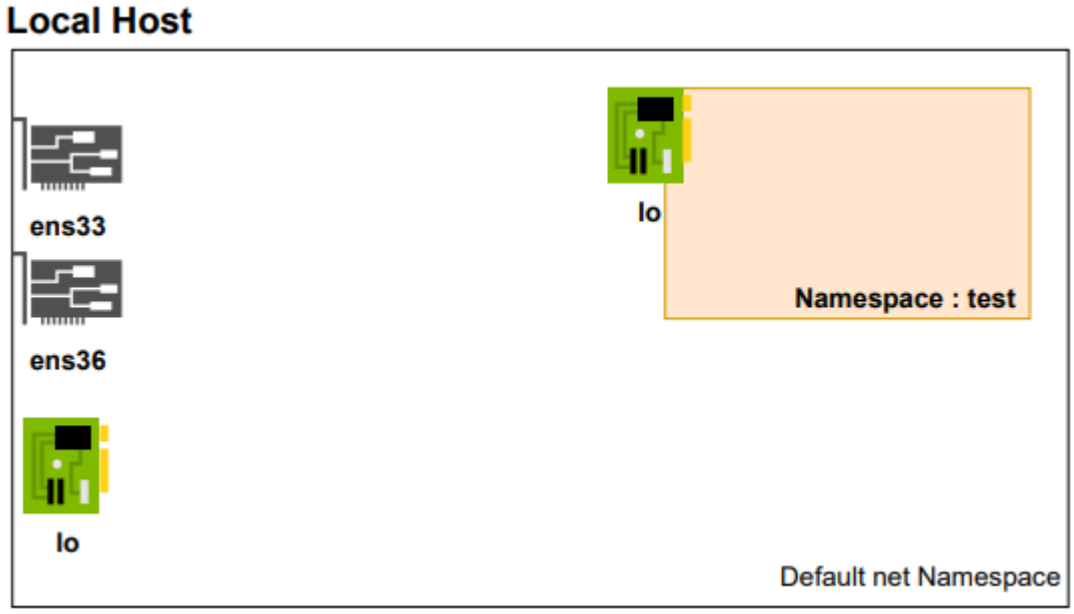
Default Network Namespace Check


```
# Host의 디폴트 Network Namespace 와 소유자를 확인
$ lsns -t net -o pid,uid,user,command
  PID   UID USER  COMMAND
    1     0 root  /sbin/init maybe-ubiquity
```

로컬 Host 의 디폴트 네트워크는 PID 1 (Init) 에서 실행되는 것을 확인할 수 있습니다.

일반적으로 우리가 보는 nic(예: eth0) 과 lo 인터페이스가 속한 기본 네임스페이스입니다.

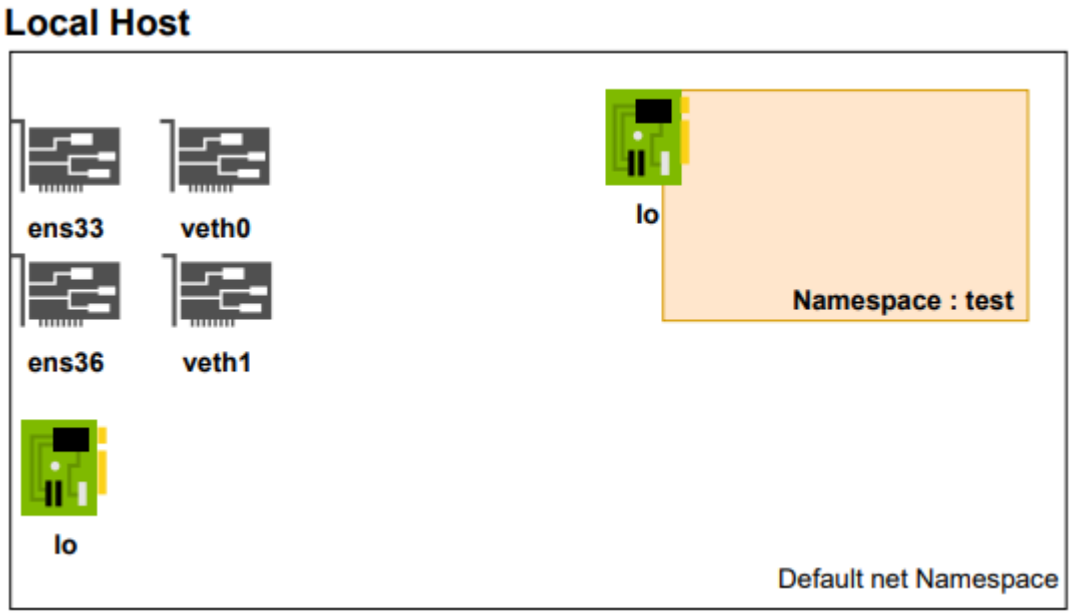
Create Network Namespace



새 네임스페이스를 생성하여, 별도의 격리된 네트워크 영역을 만들 수 있습니다.
새 네임스페이스는 lo 인터페이스만 가지고 있으며 다른 네임스페이스와 통신할 수 없는 영역입니다.

```
# test 라는 이름을 가진 Namespace 를 생성 후 리스트 확인
$ ip netns add test

$ ip netns
test

# Check
만들어진 네임스페이스를 사용하는 PID 가 없기 때문에 lsns 목록에서 보이지 않습니다
$ lsns -t net
PID USER TYPE COMMAND
   1 root net   /usr/lib/systemd/systemd --switched-root --system --deserialize 22
```

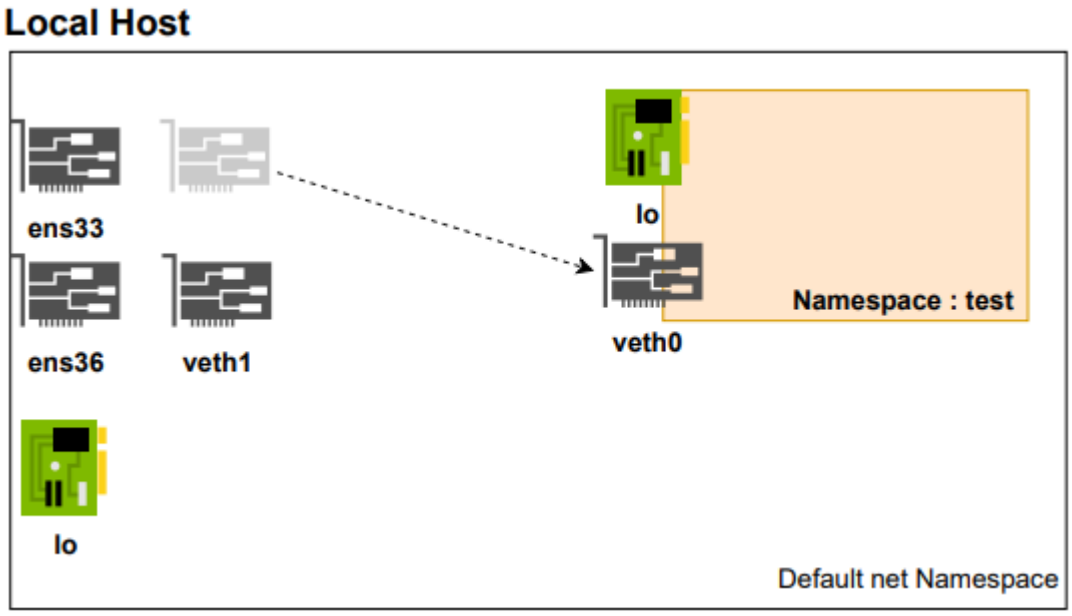
Namespace의 Network 연결 1 – 가상 인터페이스 생성



가상의 Network Namespace 간의 통신은 리눅스 커널에 존재하는 가상 인터페이스 기능인 veth 를 사용합니다.
veth 는 기본적으로 HOST <---> 다른 네임스페이스와의 연결 성립을 위한 한 쌍 단위로 구성됩니다.

```
# HOST에 가상 인터페이스를 추가 생성합니다. veth type 에 peer 로 pair 구성입니다.
$ ip link add veth0 type veth peer name veth1

# HOST에 veth0/veth1 이라는 2개의 가상 인터페이스가 생성되었습니다.
$ ip -br -c addr
lo               UNKNOWN        127.0.0.1/8
ens33            UP             211.239.150.48/23
ens36            UP             192.168.0.2/24
veth1@veth0      DOWN
veth0@veth1      DOWN
```

Namespace의 Network 연결 2 – 새 네임스페이스에 가상 인터페이스 할당



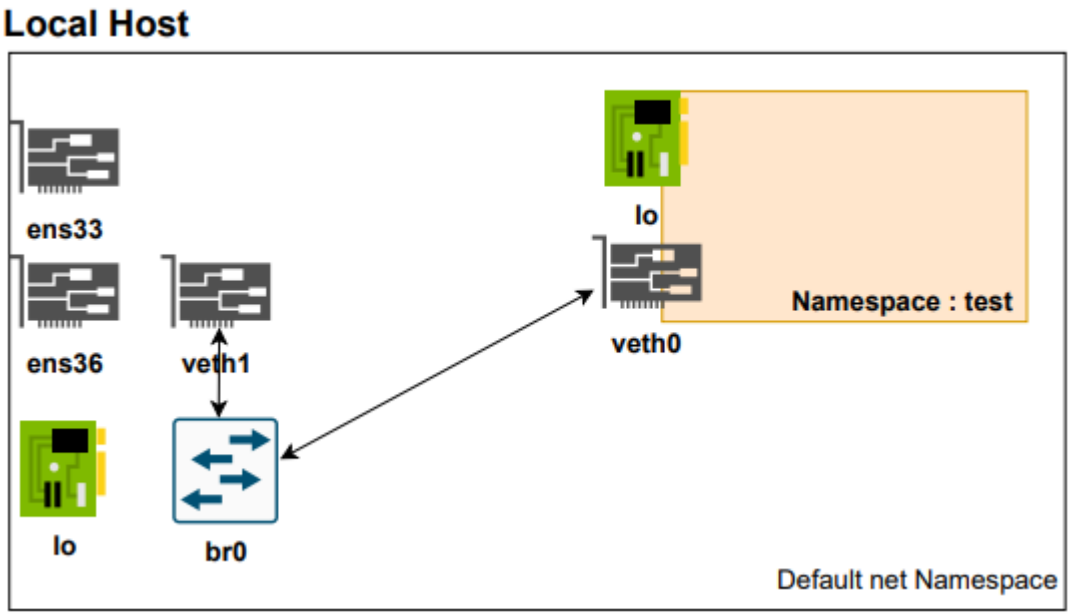
디폴트 영역에 통신을 위한 가상 인터페이스를 생성 후, 이제 새로 만든 test 네임스페이스에 달아 줄 수 있습니다.

```
# veth0 인터페이스를 test Namespace 에 Set
$ ip link set veth0 netns test

# 생성 후 HOST 에서 veth0 은 test namespace 에 장착했기 때문에 사라졌습니다.
$ ip -br -c addr
lo               UNKNOWN        127.0.0.1/8
ens33            UP             211.239.150.48/23
ens36            UP             192.168.0.2/24
veth1@if5        DOWN
```

test namespace 의 가상 인터페이스를 확인해 보면 veth0 이 장착된 것을 확인할 수 있습니다.

```
# test namespace 에 netns exec 옵션을 통한 인터페이스 조회 커맨드 실행
$ ip netns exec test ip -br addr
lo               DOWN
veth0@if4        DOWN
```

Namespace의 Network 연결 3 - bridge 연결



HOST 와 test namespace 간에 veth0과 veth1 인터페이스가 각각 세팅되었지만 DOWN 상태로 아직 통신을 할 수는 없습니다.
논리적인 가상 인터페이스 간 통신을 위해서는 IP 할당이 필요한데
이번 글에서는 HOST 영역에 브릿지 인터페이스(bridge) 를 사용해 연결해 보겠습니다.
호스트의 네트워크를 복잡하게 만들지 않으려면 브릿지라는 가상 스위치에 네임스페이스를 연결하는 것이 좋습니다.

```
# Check
$ (명령어가 없다면) yum install -y bridge-utils-1.5-9.el7.x86_64
$ btctl show
bridge name      bridge id                STP enabled
interfaces
```

현재 브릿지가 없는 것을 확인했습니다. br0 이라는 이름으로 로컬 HOST에 생성을 합니다.

```
# Bridge Create && Check
$ ip link add br0 type bridge

$ brctl show
bridge name      bridge id                STP enabled
interfaces
br0              8000.000000000000       no

# 브릿지 생성 추가 확인
$ ip -br -c addr
lo               UNKNOWN        127.0.0.1/8
```

```
ens33            UP             211.239.150.48/23
ens36            UP             192.168.0.2/24
veth1@if5        DOWN
br0              DOWN
```

하지만 여전히 br0 과 vethx 간에 어떤 연결을 설정하지 않았기 때문에 통신할 수는 없습니다.
첫 단계로 신규 네임스페이스에 설정한 인터페이스와 호스트의 브릿지를 연결해 줍니다.

```
# HOST 에 존재하는 veth1 과 Host 의 br0 브릿지를 연결합니다.
$ ip link set veth1 master br0

# check 시 bridge 에 veth1 이 추가된 것을 확인할 수 있습니다.
$ brctl show
bridge name     bridge id               STP enabled     interfaces
br0             8000.46df623e69e4       no              veth1
```

가장 서두에도 말했지만, 네트워크의 통신 요소는 인터페이스, 라우팅, IP 입니다.
통신을 위해 두 번째 단계로 이제 IP 를 설정해야 합니다.
ifconfig 같이 별도 net-util 을 설정하는 것보다 ip 명령어를 사용하는 것이 수월합니다.

```
# netns exec 명령을 사용하여 test Namespace 의 veth0 인터페이스에 IP 를 할당하고 UP 합니다.
$ ip netns exec test ip addr add 10.10.10.2/24 dev veth0
$ ip netns exec test ip link set veth0 up

# 이제 host 의 veth1 인터페이스와 bridge 인터페이스도 up 해 줍니다.
$ ip link set br0 up
$ ip link set veth1 up

# UP check 시 이제 브릿지와 인터페이스가 모두 UP 된 것을 확인할 수 있습니다.
$ ip -br -c addr
lo               UNKNOWN        127.0.0.1/8
ens33            UP             211.239.150.48/23
ens36            UP             192.168.0.2/24
veth1@if5        UP
br0              UP

# test namespace 의 인터페이스 UP 도 확인됩니다.
```

```
$ ip netns exec test ip link
1: lo: <LOOPBACK> mtu 65536 qdisc noop state DOWN mode DEFAULT
group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
5: veth0@if4: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
noqueue state UP mode DEFAULT group default qlen 1000
     link/ether f2:1c:09:d4:47:fc brd ff:ff:ff:ff:ff:ff link-
netnsid 0
```

lo 인터페이스가 DOWN 이면 해당 네임스페이스의 내부 통신을 할 수가 없습니다. UP 이후 UNKNOWN 으로 바뀌면 됩니다.

```
$ ip netns exec test ip link set dev lo up
$ ip netns exec test ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state
UNKNOWN group default qlen 1000
```

Check

```
$ ip netns exec test ping 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.063 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.058 ms
```

Check 2

Host 의 브릿지와 신규 네임스페이스 간에 통신되는지 확인하려면 브릿지에 Gateway 와 같은 동일 대역 IP할당을 합니다
브릿지에 Routing 정보가 없으면, 다른 네임스페이스의 사설 인터페이스 ip를 찾아갈 수 없습니다.

브릿지에 통신할 네임스페이스와 같은 대역의 IP 를 할당하여 라우팅 정보를 넣습니다.

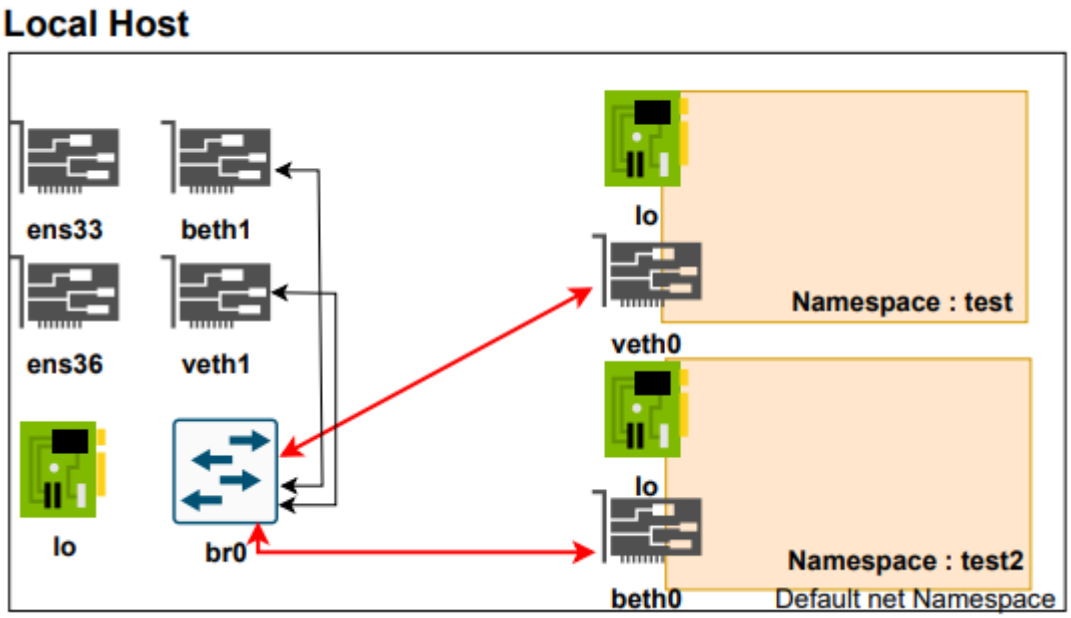
```
$ ip addr add 10.10.10.200/24 dev br0
```

test 네임스페이스의 veth0 인터페이스로 Ping 을 날려봅니다.

```
$ ping 10.10.10.2
 ping 10.10.10.2 -c 2
PING 10.10.10.2 (10.10.10.2) 56(84) bytes of data.
64 bytes from 10.10.10.2: icmp_seq=1 ttl=64 time=0.073 ms
64 bytes from 10.10.10.2: icmp_seq=2 ttl=64 time=0.071 ms

--- 10.10.10.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
```

```
rtt min/avg/max/mdev = 0.071/0.072/0.073/0.001 ms
```

Namespace의 Network 연결 4 – 다른 새 네임스페이스 간의 통신 연결

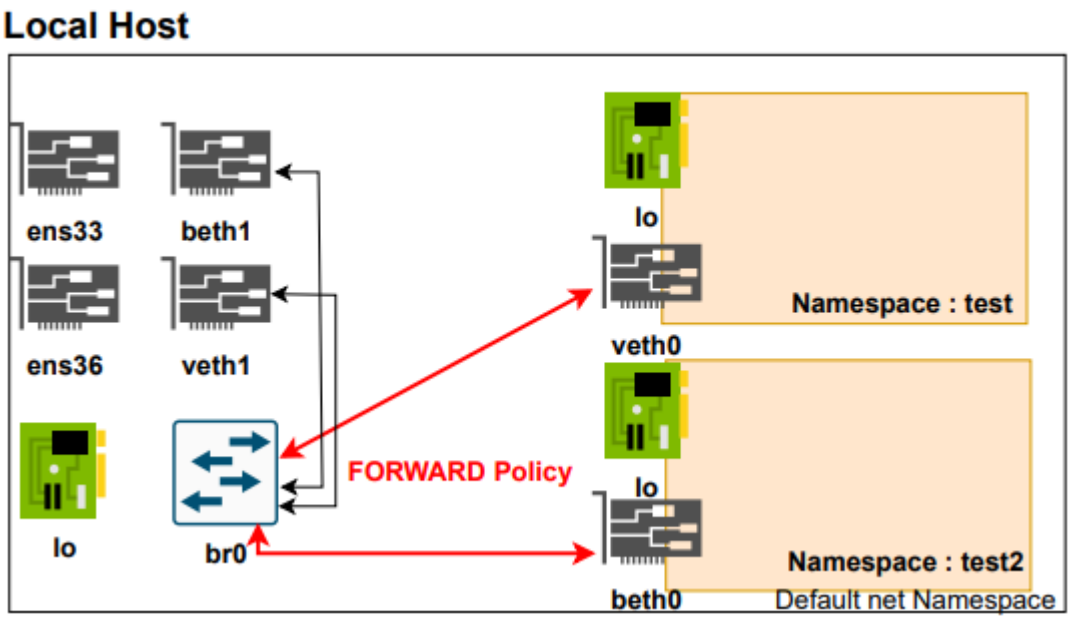


test namespace 브릿지에 추가되는 다른 네임스페이스와는 통신이 잘 되어야 합니다.
네트워크가 같다면 바로 되고, 다르다면 라우팅 설정을 추가로 해 주어야겠지요.

```
# test2 namespace 를 추가로 만들고 beth0/beth1 인터페이스를 생성 및 할당하고 브릿지에 연결합니다
# IP 세팅하고 test <---> test2 네임스페이스 간 Ping 시도시 잘 통신 됩니다.
ip netns add test2
ip link add beth0 type veth peer name beth1
ip link set beth0 netns test2
ip link set beth1 master br0
ip netns exec test2 ip addr add 10.10.10.3/24 dev beth0
ip netns exec test2 ip link set beth0 up
ip netns exec test2 ip link set dev lo up
ip link set beth1 up

# test2 namespace 확인
$ ip netns
test2 (id: 1)
test (id: 0)
```

```
$ ip -br -c addr
lo               UNKNOWN        127.0.0.1/8
ens33            UP             211.239.150.48/23
ens36            UP             192.168.0.2/24
veth1@if5        UP
br0              UP
beth1@if8        UP

$ brctl show
bridge name     bridge id               STP enabled     interfaces
br0             8000.2e0e64ccb0e5       no              beth1
                                                        veth1
# test namespace 의 veth0(10.10.10.2) 에 Ping 날려보기
ip netns exec test2 ping 10.10.10.2 -c 2
PING 10.10.10.2 (10.10.10.2) 56(84) bytes of data.
64 bytes from 10.10.10.2: icmp_seq=1 ttl=64 time=0.112 ms
64 bytes from 10.10.10.2: icmp_seq=2 ttl=64 time=0.076 ms

--- 10.10.10.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 0.076/0.094/0.112/0.018 ms
```

트러블슈팅



다른 네임스페이스와 통신이 안 된다면?
HOST 에 생성된 브릿지 인터페이스는 lo (내부통신) 이 아니기 때문에, Host 를

거쳐
다른 네트워크에 중계해 주려면 커널의 외부 포워딩 규칙을 따르기 때문입니다.
즉 NAT 구성 때와 비슷합니다. 커널 옵션에서 정의해 주어야 합니다.
ip4 트래픽에 대해 FORWARD 설정을 HOST 의 커널 옵션에 지정하여야 합니다.

```
# 먼저 HOST iptables FORWARD 정책을 확인하고 ACCEPT 로 변경합니다.
$ iptables -nL | grep -i forward
Chain FORWARD (policy DROP)

# 변경 후 체크
$ iptables --policy FORWARD ACCEPT
$ iptables -nL | grep -i forward
Chain FORWARD (policy ACCEPT)

$ service iptables save ( 또는 저장할 수 있는 OS별 옵션을 기입합니다 )

# 커널 옵션에서 ip4v.forward 활성화 합니다.
echo 1 > /proc/sys/net/ipv4/ip_forward
sysctl --system

# check
# 다시 통신 테스트를 해 봅니다.
```