アプリケーション ホスティング向け Linux

ここでは、アプリケーションのホスティングと、Cisco IOS XR オペレーティング システムでのアプリケーションのホスティングに使用する Linux 環境について説明します。

アプリケーション ホスティングの必要性

過去10年間、既存のツールチェーンとのシームレスな統合による運用上の俊敏性と効率をサポートするネットワーク オペレーティング システムが求められてきました。サービス プロバイダーは、短期の製品サイクル、俊敏なワークフロー、およびモジュール型ソフトウェアを求めていました。これらすべてを効率的に自動化することが可能です。以前の 32 ビット QNX バージョンに代わる 64 ビットの Cisco IOS XR はこれらのすべての要件を満たします。これは、アプリケーションや設定管理ツール、業界標準のゼロ タッチ プロビジョニング メカニズムの統合を簡略にする環境を提供することで実現します。64 ビットの IOS XR はサービス プロバイダー向け DevOps 形式のワークフローに一致し、アプリケーションをホストするデバイスの設定と運用を自動化するために使用できるオープンな内部データ ストレージ システムを備えています。

仮想環境への移行は加速しているものの、再利用可能で、ポータブルで、スケーラブルなアプリケーションを構築する必要性が高まっています。アプリケーションのホスティングによって、管理者には独自のツールやユーティリティを利用するためのプラットフォームが与えられます。Cisco IOS XR 6.0 は Linux ツールチェーンを使用して構築されたサードパーティ製の市販アプリケーションをサポートしています。ユーザは、シスコが提供するソフトウェア開発キットと相互にコンパイルされたカスタムアプリケーションを実行できます。アプリケーションホスティングは、ネイティブとコンテナという 2 つの形態で提供されます。ネットワーク デバイスでホストされるアプリケーションは、さまざまな用途に利用できます。これは、既存のツールのチェーンによる自動化から、設定管理のモニタリング、統合に及びます。

アプリケーションをデバイス上でホストできるようにするには、次の要件を満たす必要があります。

- アプリケーションの構築に適切なビルド環境
- デバイス外のデバイスおよびネットワークと対話するためのメカニズム

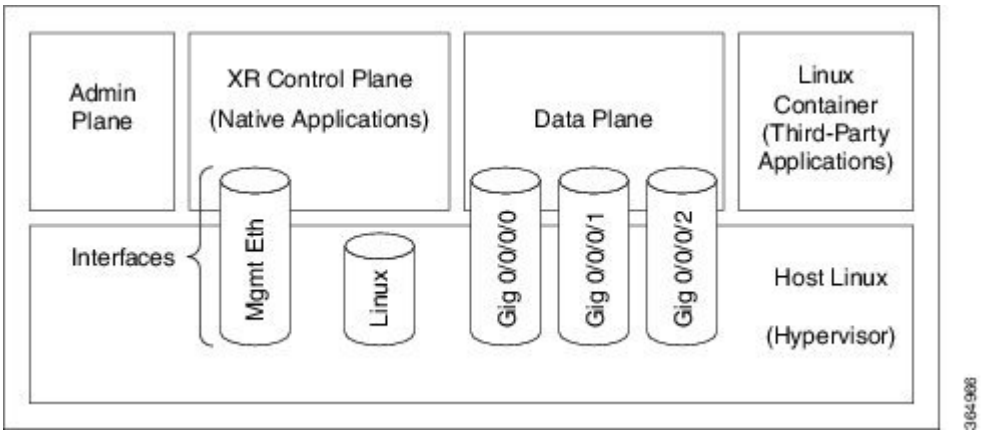
ネットワーク デバイスを、Chef や Puppet などの設定管理アプリケーションで管理すると、ネットワーク管理者は CLI のみに集中する作業から解放されます。アプリケーションが提供する抽象化により、アプリケーションがジョブを実行し、管理者は設計やその他の高レベルのタスクに集中できるようになりました。

アプリケーション ホスティングのアーキテクチャ

IOS XR は、ハイパーバイザを通じてアプリケーション ホスティング用の Linux コンテナを提供します。各コンテナによってユニークな機能が提供されます。64 ビットのホスト Linux（ハイパーバイザ）は Wind River Yocto のディストリビューションに基づいており、組み込み型システムで適切に動作します。ここでは、ホスト Linux 上で提供されるさまざまなコンテナを説明します。

次の図に、アプリケーション ホスティングのアーキテクチャを示します。

図 *1*：アプリケーション ホスティングのアーキテクチャ



- **管理プレーン**：管理プレーンは IOS XR の起動時に最初に起動される Linux コンテナです。管理プレーンは、IOS XR コントロール プレーン コンテナのライフ サイクルを管理します、
- **XR コントロール プレーン**：アプリケーションは、64 ビット IOS XR コントロール プレーンでネイティブにホストされます。コントロール プレーンを通じて IOS XR の Linux bash シェルにアクセスできます。
- **データ プレーン**：データ プレーンは、モジュラ ルータ シャーシ内のライン カードのすべての機能を代用し、提供します。
- **サードパーティ製コンテナ**：サードパーティ製アプリケーションをホストするための独自の Linux コンテナ（LXC）を作成し、提供されている LC インターフェイスを使用します。

Linux コンテナ以外に、ホストの Linux では複数のインターフェイスが提供されます。

IOS XR でのアプリケーションのホスティング向け Linux

Linux は、システム管理者、開発者、およびネットワーク エンジニアが過去 20 ～ 30 年にわたって作成し、テストし、導入してきたアプリケーションやツールのエコシステム全体をサポートします。Linuxは、アプリケーションの有無を問わず、安定性、セキュリティ、拡張性、低コストのライセンス、特定のインフラストラクチャのニーズに合わせたアプリケーションのカスタマイズを実現する柔軟性により、サーバのホスティングに適しています。

自動化と統合の簡易性に重点を置くDevOps形式のワークフローへの注目が高まる中、ネットワーク デバイスは進化し、自動化プロセスをより簡単にする標準的なツールやアプリケーションをサポートする必要があります。標準化された共有ツールのチェーンはスピード、効率、コラボレーションを強化できます。

IOS XR は Yocto ベースの Wind River 7 Linux ディストリビューションから開発されています。OS は RPM ベースとなっており、組み込み型システムに最適です。

IOS XR によって、ボックス上での 64 ビット Linux アプリケーションのホスティングが可能となり、次の利点が得られます。

- 設定管理アプリケーションとのシームレスな統合
- ファイル システムへの容易なアクセス
- 操作の簡易性

IOS XRの シンプルなアプリケーションのホスティングについては、アプリケーション ホスティングのタイプを参照してください。

IOS XR の Linux シェルの概要

IOS XR の Linux アプリケーションをホストするには、XR の Linux シェルを理解している必要があります。

次のステップに従って、XR シェルを移動します。

1. Linux ボックスから SSH を使用して IOS XR コンソールにアクセスし、ログインします。

```
cisco@host:~$ ssh root@192.168.122.188
root@192.168.122.188's password:
RP/0/RP0/CPU0:ios#
```

   IOS XR プロンプトが表示されます。

2. IOS XR のイーサネット インターフェイスを表示します。

```
RP/0/0/CPU0:ios# show ipv4 interface brief
Wed Oct 28 18:45:56.168 IST

Interface                      IP-Address      Status                Protocol
Loopback0                      1.1.1.1/32      Up                    Up
GigabitEthernet0/0/0/0         10.1.1.1/24     Up                    Up
...


RP/0/RP0/CPU0:ios#show interfaces gigabitEthernet 0/0/0/0
```

```
Wed Oct 28 18:45:56.168 IST

GigabitEthernet0/0/0/0 is up, line protocol is up
Interface state transitions: 4
Hardware is GigabitEthernet, address is 5246.e8a3.3754 (bia
5246.e8a3.3754)
Internet address is 10.1.1.1/24
MTU 1514 bytes, BW 1000000 Kbit (Max: 1000000 Kbit)
reliability 255/255, txload 0/255, rxload 0/255
Encapsulation ARPA,
Duplex unknown, 1000Mb/s, link type is force-up
output flow control is off, input flow control is off
loopback not set,
Last link flapped 01:03:50
ARP type ARPA, ARP timeout 04:00:00
Last input 00:38:45, output 00:38:45
Last clearing of "show interface" counters never
5 minute input rate 0 bits/sec, 0 packets/sec
5 minute output rate 0 bits/sec, 0 packets/sec
12 packets input, 1260 bytes, 0 total input drops
0 drops for unrecognized upper-level protocol
Received 2 broadcast packets, 0 multicast packets
0 runts, 0 giants, 0 throttles, 0 parity
0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
12 packets output, 1224 bytes, 0 total output drops
Output 1 broadcast packets, 0 multicast packets
```

出力には、`GigabitEthernet0/0/0/0` インターフェイスの IP アドレスと MAC アドレスが表示されます。

3 **run** コマンドを入力し、IOS XR の Linux bash シェルを起動します。

また、bash プロンプトの表示時に IOS XR のバージョンも確認します。

```
RP/0/RP0/CPU0:ios# run
Wed Oct 28 18:45:56.168 IST

[xr-vm_node0_RP0_CPU0:~]$ uname -a
Linux xr-vm_node0_RP0_CPU0 3.10.19-WR7.0.0.2_standard #1 SMP Mon Jul 6
13:38:23 PDT 2015 x86_64 GNU/Linux
[xr-vm_node0_RP0_CPU0:~]$
```

（注） Linux bash セルを終了し、IOS XR コンソールを起動するには、**exit** コマンドを入力します。

```
[xr-vm_node0_RP0_CPU0:~]$exit
exit
RP/0/RP0/CPU0:ios#
```

4 **ifconfig** コマンドを実行してネットワーク インターフェイスを見つけます。

```
[xr-vm_node0_RP0_CPU0:~]$ifconfig
eth0      Link encap:Ethernet  HWaddr 52:46:12:7a:88:41
          inet6 addr: fe80::5046:12ff:fe7a:8841/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:8996  Metric:1
          RX packets:280 errors:0 dropped:0 overruns:0 frame:0
          TX packets:160 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:31235 (30.5 KiB)  TX bytes:20005 (19.5 KiB)

eth-vf0   Link encap:Ethernet  HWaddr 52:54:00:34:29:44
          inet addr:10.11.12.14  Bcast:10.11.12.255  Mask:255.255.255.0
          inet6 addr: fe80::5054:ff:fe34:2944/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:9000  Metric:1
          RX packets:19 errors:0 dropped:0 overruns:0 frame:0
          TX packets:13 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1566 (1.5 KiB)  TX bytes:1086 (1.0 KiB)
```

```
eth-vf1   Link encap:Ethernet  HWaddr 52:54:00:ee:f7:68
          inet6 addr: fe80::5054:ff:feee:f768/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:9000  Metric:1
          RX packets:326483 errors:0 dropped:3 overruns:0 frame:0
          TX packets:290174 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:24155455 (23.0 MiB)  TX bytes:215862857 (205.8 MiB)

eth-vf1.1794 Link encap:Ethernet  HWaddr 52:54:01:5c:55:8e
          inet6 addr: fe80::5054:1ff:fe5c:558e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:8996  Metric:1
          RX packets:10 errors:0 dropped:0 overruns:0 frame:0
          TX packets:13 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:728 (728.0 B)  TX bytes:1234 (1.2 KiB)

eth-vf1.3073 Link encap:Ethernet  HWaddr e2:3a:dd:0a:8c:06
          inet addr:192.0.0.4  Bcast:192.255.255.255  Mask:255.0.0.0
          inet6 addr: fe80::e03a:ddff:fe0a:8c06/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:8996  Metric:1
          RX packets:317735 errors:0 dropped:3560 overruns:0 frame:0
          TX packets:257881 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:18856325 (17.9 MiB)  TX bytes:204552163 (195.0 MiB)

eth-vf1.3074 Link encap:Ethernet  HWaddr 4e:41:50:00:10:01
          inet addr:172.0.16.1  Bcast:172.255.255.255  Mask:255.0.0.0
          inet6 addr: fe80::4c41:50ff:fe00:1001/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:8996  Metric:1
          RX packets:8712 errors:0 dropped:0 overruns:0 frame:0
          TX packets:32267 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:723388 (706.4 KiB)  TX bytes:11308374 (10.7 MiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:1635360 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1635360 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:182532711 (174.0 MiB)  TX bytes:182532711 (174.0 MiB)

tap123    Link encap:Ethernet  HWaddr c6:13:74:4b:dc:e3
          inet6 addr: fe80::c413:74ff:fe4b:dce3/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:13 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:0 (0.0 B)  TX bytes:998 (998.0 B)
```

出力には、IOS XR が使用する内部インターフェイス（`eth0` ～ `eth-vf1.3074`）が表示されます。これらのインターフェイスは、XR ネットワークの名前空間（XRNNS）にあり、IOS XR の外部のネットワークとやり取りすることはありません。IOS XR の外部のネットワークとやり取りするインターフェイスは、サードパーティ製ネットワークの名前空間（TPNNS）にあります。TPNNS については、IOS XR のサードパーティ製ネットワークの名前空間, （6 ページ）を参照してください。

Linux ネットワークの名前空間の概要

一般的な Linux OS は、ネットワーク インターフェイスと OS で共有されるルーティング テーブル エントリ一式を提供します。ネットワークの名前空間の導入により、Linux は 独立して機能する複数のインスタンスのネットワーク インターフェイスとルーティング テーブルを提供します。

（注） ネットワークの名前空間のサポートは、Linux OS のディストリビューションによって異なります。アプリケーションのホスティングに使用するディストリビューションがネットワークの名前空間をサポートしていることを確認します。

ネットワークの名前空間に移動するには、次のコマンドを使用します。

```
ip netns exec <namespace_name>
```

IOS XR のサードパーティ製ネットワークの名前空間

IOS XR の Linux シェルは、サードパーティ製アプリケーションと内部 XR プロセス間に必要な隔離を実装すると同時に、XR インターフェイスへの必要なアクセスをアプリケーションに提供するサードパーティ製ネットワークの名前空間を提供します。

次に、IOS XR で TPNNS を表示する例を示します。

1. IOS XR の bash シェルで TPNNS に移動します。

```
[XR-vm_node0_RP0_CPU0:~]$ ip netns exec tpnns bash
```

2. TPNNS インターフェイスを表示します。

```
[XR-vm_node0_RP0_CPU0:~]$ ifconfig
Gi0_0_0_0 Link encap:Ethernet  HWaddr 52:46:04:87:19:3c
          inet addr:192.164.168.10  Mask:255.255.255.0
          inet6 addr: fe80::5046:4ff:fe87:193c/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1514  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:210 (210.0 B)

Mg0_RP0_CPU0_0 Link encap:Ethernet  HWaddr 52:46:12:7a:88:41
          inet addr:192.168.122.197  Mask:255.255.255.0
          inet6 addr: fe80::5046:12ff:fe7a:8841/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1514  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:210 (210.0 B)

fwd_ew    Link encap:Ethernet  HWaddr 00:00:00:00:00:0b
          inet6 addr: fe80::200:ff:fe00:b/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:1 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:140 (140.0 B)

fwdintf   Link encap:Ethernet  HWaddr 00:00:00:00:00:0a
          inet6 addr: fe80::200:ff:fe00:a/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1482  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:1 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:140 (140.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
```

```
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo:0      Link encap:Local Loopback
          inet addr:1.1.1.1  Mask:255.255.255.255
          UP LOOPBACK RUNNING  MTU:1500  Metric:1
```

出力に表示されるインターフェイスはLinux環境でのIOS XRインターフェイスの複製です（同じMACアドレスとIPアドレスを備えています）。

- `Gi0_0_0_0` はIOS XR GigabitEthernet 0/0/0/0インターフェイスです。
- `Mg0_RP0_CPU0_0` はXRでの管理操作に使用するIOS XR管理インターフェイスです。
- `fwd_ew` はサードパーティ製アプリケーションとIOS XR間の通信（水平方向）に使用するインターフェイスです。
- `fwdintf` はIOS XRの外部のサードパーティ製アプリケーションとネットワーク間の通信に使用するインターフェイスです。
- `lo:0` は `fwdintf` インターフェイスを通じたサードパーティ製アプリケーションと外部ネットワーク間の通信に使用するIOS XR loopback0インターフェイスです。loopback0インターフェイスは、XRの外部の通信に使用できるように設定する必要があります。また、IOS XR の外部との通信の項で説明したように、アプリケーションは外部通信用のGigEインターフェイスも設定できます。

（**no shut** コマンドで）イネーブルになっているすべてのインターフェイスがIOSXRのTPNNSに追加されます。

3 （任意）`fwd_ew` インターフェイスと `fwdintf` インターフェイスで使用されるIPアドレスを表示します。

```
[xr-vm_node0_RP0_CPU0:~]$ ip route
default dev fwdintf  scope link  src 1.1.1.1
8.8.8.8 dev fwd_ew  scope link
192.168.122.0/24 dev Mg0_RP0_CPU0_0  proto kernel  scope link  src 192.168.122.213
```

IOS XR でのサードパーティ製ネットワーク名前空間に移動する代替方法

IOS XR へのログイン時に **ip netns exec tpnns bash** コマンドを入力せずにTPNNSに直接移動するには、次に示すステップで説明する `sshd_tpnns` サービスを使用します。この手順には、サービスにアクセスするための非ルートユーザの作成が含まれています（ルートユーザはこのサービスにアクセスできません）。

（注） IOS XR で、インターフェイスをバインドするサービスを開始する前に、インターフェイスが設定され、起動され、動作可能であることを確認します。

インターフェイスが設定された後にのみサービスを開始するようにするには、サービス スクリプトに次の関数を含めます。

```
. /etc/init.d/tpnns-functions
tpnss_wait_until_ready
```

tpnss_wait_until_ready 関数を追加することによって、サービス スクリプトが 1 つ以上のインターフェイスが設定されるのを待ってから、サービスを開始するようになります。

1. （任意）リロード時に TPNNS サービスを自動的に開始するようにするには、`sshd_tpnns` サービスを追加し、そのサービスの存在を確認します。

```
bash-4.3# chkconfig --add sshd_tpnns
bash-4.3# chkconfig --list sshd_tpnns
sshd_tpnns      0:off   1:off   2:off   3:on    4:on    5:on    6:off
bash-4.3#
```

2. `sshd_tpnns` サービスを開始します。

```
bash-4.3# service sshd_tpnns start
Generating SSH1 RSA host key: [  OK  ]
Generating SSH2 RSA host key: [  OK  ]
Generating SSH2 DSA host key: [  OK  ]
  generating ssh ECDSA key...
Starting sshd: [  OK  ]

bash-4.3# service sshd_tpnns status
sshd (pid  6224) is running...
```

3. ステップ 1 で作成した非ルート ユーザとして `sshd_tpnns` セッションにログインします。

```
host@fe-ucs36:~$ ssh devops@192.168.122.222 -p 57722
devops@192.168.122.222's password:
Last login: Tue Sep  8 20:14:11 2015 from 192.168.122.1
XR-vm_node0_RP0_CPU0:~$
```

4. インターフェイスを表示して、TPNNS に移動していることを確認します。

```
[XR-vm_node0_RP0_CPU0:~]$ ifconfig
Gi0_0_0_0 Link encap:Ethernet  HWaddr 52:46:04:87:19:3c
          inet addr:192.164.168.10  Mask:255.255.255.0
          inet6 addr: fe80::5046:4ff:fe87:193c/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1514  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:210 (210.0 B)

Mg0_RP0_CPU0_0 Link encap:Ethernet  HWaddr 52:46:12:7a:88:41
          inet addr:192.168.122.197  Mask:255.255.255.0
          inet6 addr: fe80::5046:12ff:fe7a:8841/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1514  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:210 (210.0 B)

fwd_ew    Link encap:Ethernet  HWaddr 00:00:00:00:00:0b
          inet6 addr: fe80::200:ff:fe00:b/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:1 overruns:0 carrier:0
```

```
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:140 (140.0 B)

fwdintf   Link encap:Ethernet  HWaddr 00:00:00:00:00:0a
          inet6 addr: fe80::200:ff:fe00:a/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1482  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:1 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:140 (140.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo:0      Link encap:Local Loopback
          inet addr:1.1.1.1  Mask:255.255.255.255
          UP LOOPBACK RUNNING  MTU:1500  Metric:1
```