IOS XR でのシンプルなアプリケーションのホスティング

ここでは、さまざまな種類のアプリケーションのホスティングについて説明し、iPerfなどのシンプルなアプリケーションを IOS XR でネイティブに、またはサードパーティ製コンテナ内でホストする方法を示します。

- アプリケーション ホスティングのタイプ，1 ページ
- ネイティブ アプリケーションのホスティング：概要とワークフロー，2 ページ
- コンテナ アプリケーションのホスティング：概要とワークフロー，6 ページ

アプリケーション ホスティングのタイプ

IOS XR でのアプリケーションのホスティングは、2 つの形態で提供されます。

- **ネイティブ**：IOS XR が提供するコンテナ内でアプリケーションをホストできます。アプリケーションは、パッケージ マネージャとして RPM を使用するシスコ固有の Linux ディストリビューション（River Linux 7）で構築する必要があります。アプリケーションは、IOS XR のルート ファイル システムにあるライブラリを使用します。Chef や Puppet などの設定管理ツールを使用して、アプリケーションのインストールを自動化できます。
- **コンテナ**：IOS XR で独自のコンテナを作成し、そのコンテナ内でアプリケーションをホストします。アプリケーションは、Linux ディストリビューションを使用して開発できます。これは、IOS XR のルート ファイル システムが提供するものとは異なるシステム ライブラリを使用するアプリケーションに適しています。

アプリケーション ホスティングのタイプの選択

要件と次の基準に応じて、アプリケーション ホスティングのタイプを選択できます。

- **リソース**：ホストされているアプリケーションが消費するリソースの量を制御する必要がある場合は、制約事項を設定できるコンテナ モデルを選択する必要があります。ネイティブ

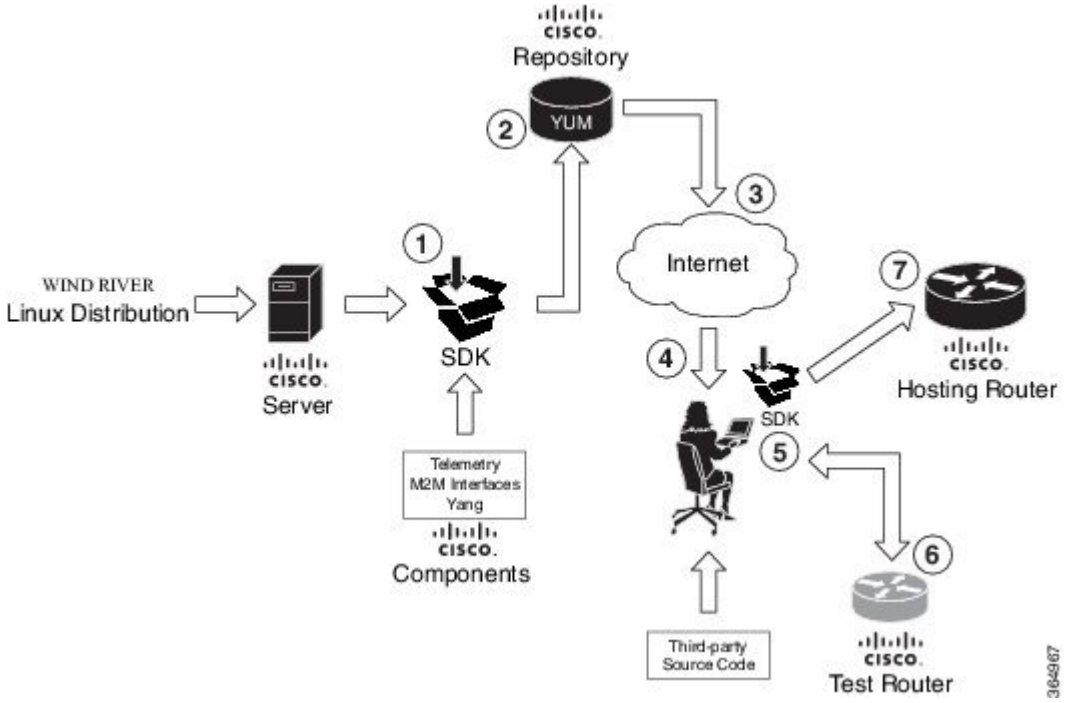
モデルでは、内部IOS XRプロセスと共有する割り当て済みのリソースを使用するアプリケーションのみを展開できます。

- **環境の選択**：ネイティブにホストされるアプリケーションは、IOS XRが提供するWind River 7 Linuxディストリビューションで構築する必要があります。アプリケーションの構築に使用するLinuxディストリビューションの選択肢を決定したら、コンテナ モデルを選択する必要があります。コンテナ モデルを使用してアプリケーションをホストすると、導入前にパッケージ化できます。

ネイティブ アプリケーションのホスティング：概要とワークフロー

ここでは、SDKシェルスクリプトを使用したネイティブアプリケーションのホスティングのワークフローを説明します。

図 *1*：*SDK* シェルス クリプトを使用したネイティブ アプリケーションのホスティング



1 SDKはYoctoプロジェクトのワークスペースを使用するWind River Linux 7（WRL7）ディストリビューションで構築されています。

IOS XRは、ベースの1つとしてYoctoプロジェクトからオープンな組み込み型カーネル インフラストラクチャを使用します。したがって、IOS XRには、すべてのシステム ライブラリと、アプリケーションをサポートするための基本的なツール チェーンが含まれています。その結果、IOS XRでネイティブにホストされるアプリケーションはWind Riverディストリビューショ

ン用に再構築する必要があります。また、テレメトリや YANG データ モデルなどの IOS XR の他の機能でそれらのアプリケーションをカスタマイズできます。

2 SDK はコンパイルされて、シスコの（YUM）リポジトリに保存されます。

（注） ネイティブ アプリケーションのホスティング環境は QEMU ハイパーバイザを使用するか、または SDK シェル スクリプトを実行するかのいずれかで構築されます。

Yellowdog Updated, Modified（YUM）は、Linux 用のオープン ソースのコマンドライン パッケージ管理ユーティリティであり、組み込み型のディストリビューション ソース ファイルの作成や Red Hat パケット マネージャ（RPM）などの適切な形式へのコンパイルが可能です。YUM では、自動のパケット更新が許可されます。YUM を使用することで、入手可能なパッケージをインストール、削除、更新、および表示することができます。

3 シスコのリポジトリは、インターネット上でのアプリケーション開発者によるアクセスを可能にします。

4 開発者は、SDK をダウンロードし、インストール用の SDK シェルス クリプトを実行します。詳細については、「SDK シェル スクリプトを使用した相互ビルド環境の作成」を参照してください。

5 開発者は IOS XR でホストされるアプリケーションを再構築します。詳細については、「ネイティブ RPM の構築」を参照してください。

6 開発者はテスト ルータで再構築されたアプリケーションをホストします。

7 開発者は、IOS XR を実行するホスティング ルータで再構築されたアプリケーションをホストします。

ネイティブ アプリケーションをホスティングするビルド環境の準備については、ビルド環境のセットアップを参照してください。

ネイティブ アプリケーションとしての iPerf の実行

ネイティブ アプリケーションのホスティングの例として、この項で説明するように、IOS XR に iPerf クライアントをネイティブにインストールして、別のルータにネイティブにインストールされた iPerf サーバとの接続を確認することができます。

トポロジ

次の図に、この例で使用されるトポロジを説明します。

図 *2*：ネイティブ アプリケーションとしての *iPerf*

iPerf サーバはルータ A に、iPerf クライアントはルータ B にインストールされています。どちらのインストールも IOS RX でネイティブに実行されます。iPerf クライアントは、IOS XR が提供するインターフェイスを通じて iPerf サーバと通信します。

前提条件

2 つのルータがトポロジに示すように設定されていることを確認します。

設定手順

ネイティブ アプリケーションとして iPerf を実行するには、次の手順を実行します。

1. ルータ A にログインし、XRNNS に入ります。

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

2. ルータ A の RPM として iPerf サーバをインストールします。

```
[xr-vm_node0_RP0_CPU0:~]$yum install
https://s3.amazonaws.com/alpha-builds/iperf-2.0.5.ios_xr6.x86_64.rpm
```

3. ステップ 1 と 2 を実行し、iPerf クライアントをルータ B にインストールします。

4. ルータ A の iPerf サーバのインストールを確認します。

```
[xr-vm_node0_RP0_CPU0:~]$ iperf -v


iperf version 2.0.5 (08 Jul 2010) pthreads
```

   同様に、ルータ B の iPerf クライアントのインストールを確認します。

5. ルータ A の Loopback0 インターフェイスを iPerf サーバにバインドし、iPerf サーバ インスタンスを起動します。

   次の例では、1.1.1.1 はルータ A の Loopback0 インターフェイスの割り当てられたアドレス、57730 は通信に使用されるポート番号です。

```
[xr-vm_node0_RP0_CPU0:~]$ iperf -s -B 1.1.1.1 -p 57730
Server listening on TCP port 57730
Binding to local address 1.1.1.1
TCP window size: 85.3 KByte (default)
```

6 iPerf サーバに使用されているものと同じポート番号とルータ A の管理 IP アドレスを指定して、ルータ B の iPerf クライアント インスタンスを起動します。

次の例では、192.168.122.213 はルータ A の管理 IP アドレス、57730 は iPerf サーバへのアクセスに使用されるポート番号です。

```
[xr-vm_node0_RP0_CPU0:~]$ iperf -c 192.168.122.213 -p 57730
------------------------------------------------------------
Client connecting to 192.168.122.213, TCP port 57730
TCP window size: 85.0 KByte (default)
------------------------------------------------------------
[  3] local 192.168.122.1 port 46974 connected with 192.168.122.213 port 57730
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec   146 MBytes   122 Mbits/sec
```

TCP ではなく UDP を使用して iPerf サーバと通信するには、次のコマンドを使用します。

```
[xr-vm_node0_RP0_CPU0:~]$ iperf -c 192.168.122.213 -p 57730 -u
------------------------------------------------------------
Client connecting to 192.168.122.213, UDP port 57730
Sending 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 192.168.122.1 port 41466 connected with 192.168.122.213 port 57730
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec
[  3] Sent 893 datagrams
[  3] Server Report:
[  3]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec   0.233 ms    0/  893 (0%)
[root@hostB ~]#
```

7 ルータ B の iPerf クライアントから iPerf サーバに ping を実行します。

```
[xr-vm_node0_RP0_CPU0:~]$ /bin/ping 192.164.168.10
PING 192.164.168.10 (192.164.168.10) 56(84) bytes of data.
64 bytes from 192.164.168.10: icmp_seq=1 ttl=255 time=13.0 ms
64 bytes from 192.164.168.10: icmp_seq=2 ttl=255 time=2.14 ms
64 bytes from 192.164.168.10: icmp_seq=3 ttl=255 time=2.21 ms
```

ルータ B でホストされている iPerf クライアントはルータ A でホストされている iPerf サーバにアクセスします。

iPerf 用のネイティブ アプリケーションは正常にホスティングされています。

System V Init スクリプトを使用した iPerf の操作

iPerf のサーバ サービスまたはクライアント サービスは、IOS XR で自動的に起動、停止、または再起動できます。これを実現するには、次の例に示すように、System V（SysV）スクリプトを作成し、追加する必要があります。

1 iPerf サーバまたはクライアントを起動、停止、またはリロードする SysV スクリプトを作成します。

次の例では、iPerf サーバ用のスクリプトを作成します。また、iPerf クライアント用のスクリプトを作成するには、iPerf の設定手順のステップ 5 で説明したコマンドを使用します。

```
#!/bin/bash
#
# description: iPerf server
#
# Get function from functions library
. /etc/init.d/functions

# Start the service iperf
start() {
```

```
        iperf -s -B 1.1.1.1 -p 57730 & "Starting the iPerf Server: "
        /path/to/iperf &
        ### Create the lock file ###
        touch /var/lock/subsys/iperf
        success $"iPerf server startup"
        echo
}

# Restart the service iperf
stop() {
        iperf -s -B 1.1.1.1 -p 57730 & "Stopping the iPerf Server: "
        killproc iperf
        ### Delete the lock file ###
        rm -f /var/lock/subsys/iperf
        echo
}
### main logic ###
case "$1" in
  start)
        start
        ;;
  stop)
        stop
        ;;
  status)
        status iperf
        ;;
  restart|reload|condrestart)
        stop
        start
        ;;
  *)
        echo $"Usage: $0 {start|stop|restart|reload|status}"
        exit 1
esac
exit 0
```

2 iPerf サーバまたはクライアントをホストする IOS XR にスクリプトを追加します。

```
bash-4.3# chkconfig --add iperf
```

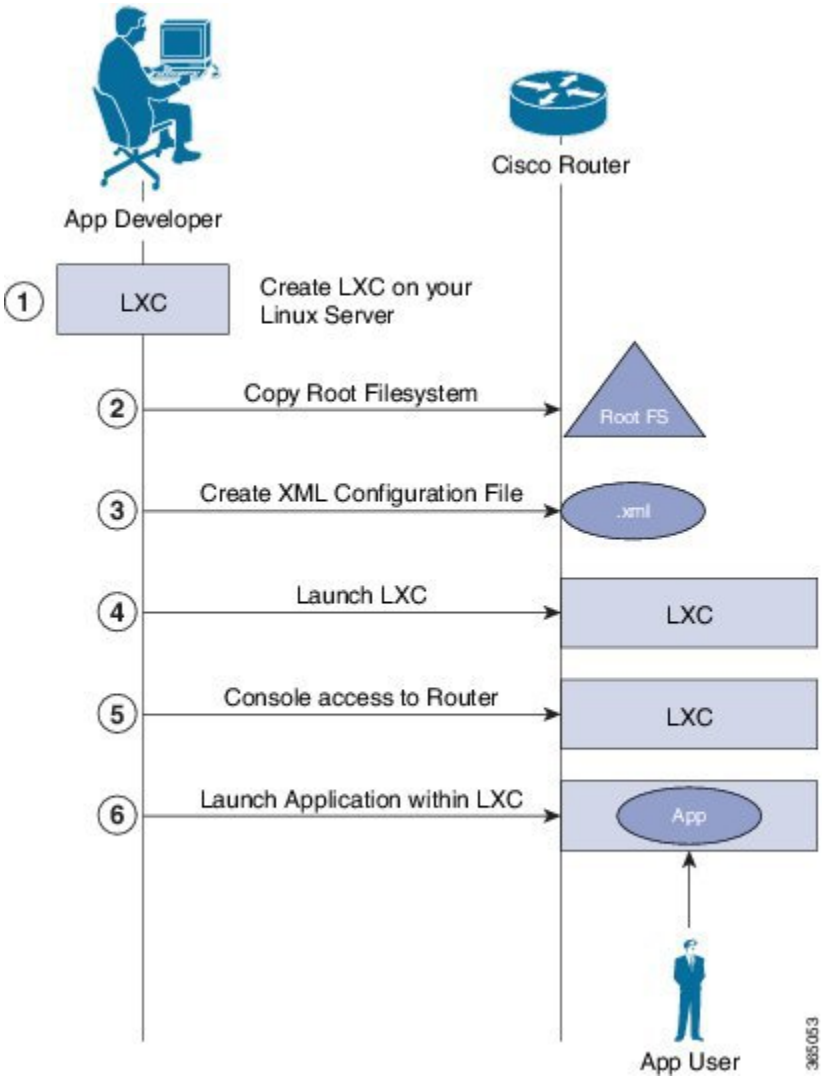
コンテナアプリケーションのホスティング：概要とワークフロー

ここでは、コンテナ アプリケーションのホスティングの概念を示し、そのワークフローについて説明します。

コンテナ アプリケーションのホスティングにより、アプリケーションが IOS XR の Linux コンテナ内の独自の環境とプロセス空間（名前空間）でホストできるようになります。アプリケーション開発者は、アプリケーション開発環境を完全に制御することができ、選択した Linux ディストリビューションを使用できます。アプリケーションは、IOS XR コントロール プレーン プロセスから隔離されていますが、XR GigE インターフェイスを通じて XR 外のネットワークに接続できます。また、アプリケーションは、IOS XR のローカル ファイル システムにも簡単にアクセスできます。

次の図に、アプリケーションをホストするための Linux コンテナを作成するワークフローを示します。完全な設定手順については、 コンテナ アプリケーションとしての iPerf の実行, （8 ページ）を参照してください。

図 *3*：コンテナ アプリケーションのホスティングのワークフロー



コンテナ アプリケーションのホスティングには 2 つのコンポーネントがあります。

- **Linux サーバ**：アプリケーションを開発し、Linux コンテナ（LXC）を起動状態にし、コンテナ環境を準備するために使用するサーバです。
- **ルータ**：実行するアプリケーションでコンテナをホストするために使用される 64 ビットの IOS XR を実行するルータです。

1 Linux サーバ上で、LXC を起動し、次の手順を実行します。

a コンテナ環境と必要なライブラリを準備します。

b LXC をシャットダウンします。

2 IOS XR を実行するルータに接続し、ルート ファイル システムをコピーします。

3 コンテナの設定ファイルを .xml 形式で作成します。このファイルは、コンテナ名、デフォルトの名前空間などのコンテナの属性を指定します。

（注） サードパーティ製のネットワークの名前空間（TPNNS）を指定すると、デフォルトで LXC が TPNNS で起動します。

4 ルータの LXC を起動します。

5 IOS XR コンソール アクセスによりルータの LXC にログインします。

6 アプリケーションを手動で起動するか、または LXC の起動時にアプリケーションが自動的に起動するように設定します。

Linux ボックスなどのコンテナを使用してユーザ用のアプリケーションをインストールしたり、ホスティングしたりできます。

コンテナ アプリケーションとしての iPerf の実行

コンテナアプリケーションのホスティングの例として、この項で説明するように、IOS XR の LXC 内に iPerf クライアントをインストールして、別のルータの LXC 内に インストールされた iPerf サーバとの接続を確認することができます。

トポロジ

次の図で、この例で使用されるトポロジを説明します。

図 *4* : コンテナ アプリケーションとしての *iPerf*

iPerf サーバはルータ A に、iPerf クライアントはルータ B にインストールされています。どちらのインストールも 64 ビット IOS XR のコンテナ内で実行されます。iPerf クライアントは、IOS XR が提供するインターフェイスを通じて iPerf サーバと通信します。

前提条件

2 つのルータがトポロジに示すように設定されていることを確認します。

設定手順

アプリケーション コンテナとして iPerf を実行するには、次の手順を実行します。

1. ルータ A にログインし、XRNNS に入ります。

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

2. LXC を起動します。

```
[xr-vm_node0_RP0_CPU0:~]$virsh -c lxc+tcp://10.11.12.15:16509/ -e ^Q console demo1
```

3. 要求されたら、LXC にログインします。

```
Connected to domain demo
Escape character is ^Q
Kernel 3.14.23-WR7.0.0.2_standard on an x86_64

host login: Password:
```

4. ルータ A の LXC 内に iPerf サーバをインストールします。

```
[root@host ~]#apt-get install iperf
```

5. ステップ 1 ～ 4 を実行し、iPerf クライアントをルータ B にインストールします。

6. ルータ A の iPerf サーバのインストールを確認します。

```
[root@host ~]#iperf -v


iperf version 2.0.5 (08 Jul 2010) pthreads
```

   同様に、ルータ B の iPerf クライアントのインストールを確認します。

7. ルータ A の Loopback0 インターフェイスを iPerf サーバにバインドし、iPerf サーバ インスタンスを起動します。

   次の例では、1.1.1.1 はルータ A の Loopback0 インターフェイスの割り当てられたアドレス、57730 は通信に使用されるポート番号です。

```
[root@host ~]#iperf -s -B 1.1.1.1 -p 57730
Server listening on TCP port 57730
Binding to local address 1.1.1.1
TCP window size: 85.3 KByte (default)
```

8. iPerf サーバに使用されているものと同じポート番号とルータ A の管理 IP アドレスを指定して、ルータ B の iPerf クライアント インスタンスを起動します。

   次の例では、192.168.122.213 はルータ A の管理 IP アドレス、57730 は iPerf サーバへのアクセスに使用されるポート番号です。

```
[root@host ~]#iperf -c 192.168.122.213 -p 57730
------------------------------------------------------------
Client connecting to 192.168.122.213, TCP port 57730
TCP window size: 85.0 KByte (default)
------------------------------------------------------------
[  3] local 192.168.122.1 port 46974 connected with 192.168.122.213 port 57730
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec   146 MBytes   122 Mbits/sec
```

TCP ではなく UDP を使用して iPerf サーバと通信するには、次のコマンドを使用します。

```
[root@host ~]#iperf -c 192.168.122.213 -p 57730 -u
------------------------------------------------------------
Client connecting to 192.168.122.213, UDP port 57730
Sending 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 192.168.122.1 port 41466 connected with 192.168.122.213 port 57730
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec
[  3] Sent 893 datagrams
[  3] Server Report:
[  3]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec   0.233 ms    0/  893 (0%)
[root@hostB ~]#
```

9 ルータ B の iPerf クライアントから iPerf サーバに ping を実行します。

```
[root@host ~]#/bin/ping 192.164.168.10
PING 192.164.168.10 (192.164.168.10) 56(84) bytes of data.
64 bytes from 192.164.168.10: icmp_seq=1 ttl=255 time=13.0 ms
64 bytes from 192.164.168.10: icmp_seq=2 ttl=255 time=2.14 ms
64 bytes from 192.164.168.10: icmp_seq=3 ttl=255 time=2.21 ms
```

ルータ B でホストされている iPerf クライアントはルータ A でホストされている iPerf サーバにアクセスできます。

iPerf 用のコンテナ アプリケーションは正常にホスティングされています。コンテナ アプリケーションのホスティングの使用例については、IOS XR のコンテナ内でテレメトリ レシーバを実行するを参照してください。