サードパーティ製アプリケーション

サードパーティ製アプリケーションについて

サードパーティのアプリケーションは、Bash シェルで **dnf** コマンドを使用するか、NX-OS CLI を介してネイティブ ホストにインストールされます。

dnf install *rpm* コマンドを入力すると、Cisco **DNF** プラグインが実行されます。このプラグインは、RPM を表示されない場所にコピーします。スイッチのリロード時に、システムは RPM を再インストールします。

構成が `/etc` に置かれている場合、Linux プロセス、**incrond** は、ディレクトリで作成されたアーティファクトをモニターし、それらを表示されない場所にコピーし、この場所から `/etc` にコピーし直します。

注意事項と制約事項

サードパーティ製アプリケーションの RPM には、次の注意事項と制約事項があります。

- NX-OS 10.1(1) リリースには、NX-Linux（シスコ独自の Linux ディストリビューション）に基づく新しいオペレーティング システムと rootfs があるため、WRL5/WRL8 を使用して構築されたサードパーティ製 RPM は NX-Linux と互換性がない場合がありますそのため、サードパーティ製ソフトウェアが機能しない可能性があります。この場合、以前のリリースで使用されていたアプリケーションの古いバージョンを削除し、NX-Linux と互換性のある新しいソフトウェアに置き換えます。 にあります。

- 署名付き RPM のインストールに関するガイドラインと手順については、『Cisco Nexus 9000 Series NX-OS Software Upgrade and Downgrade Guide, Release 9.2(x)』を参照してください。リポジトリの数などです。
- サードパーティ製アプリケーションは、スイッチの起動時に開始されます。サードパーティ製アプリケーションは、その通信インターフェイスが起動する前、またはスイッチと通信ピアまたはサーバー間のルーティングが確立される前に起動される可能性があります。したがって、すべてのサードパーティ製アプリケーションは、通信障害が発生した場合に堅牢になるように作成し、アプリケーションは接続の確立を再試行する必要があります。通信障害が発生してもアプリケーションに復元力がない場合は、「ラッパー」アプリケーションを使用して、目的のアプリケーションを起動する前に通信ピアが到達可能であることを確認するか、必要に応じて目的のアプリケーションを再起動する必要があります。
- Cisco NX-OS リリース 10.2(3)F 以降、Python2 および依存 RPM は NX-OS から削除されます。ただし、Python2 および依存する RPM は、devhub サイトからパッケージグループ packagegroup-nxos-64-python-2-deprecated-rpms としてインストールできます。

Python2 および依存パッケージのインストール

次に、パッケージのインストールの完全なワークフローを示します。

```
switch# cat /etc/dnf/repos.d/open-nxos.repo
[open-nxos]
name=open-nxos
baseurl=https://devhub.cisco.com/artifactory/open-nxos/10.2.3/
enabled=1
gpgcheck=0
sslverify=0

dnf info packagegroup-nxos-64-python-2-deprecated-rpms
dnf install packagegroup-nxos-64-python-2-deprecated-rpms
The output of these cmds will be available post KR3F CCO.
```

サードパーティのネイティブ RPM/パッケージのインストール

パッケージのインストールの完全なワークフローは次のとおりです。

手順

エージェントが保存されているシスコのリポジトリを指すように、スイッチのリポジトリを設定します。

```
bash-4.2# cat /etc/dnf/repos.d/open-nxos.repo
[open-nxos]
name=open-nxos
```

```
enabled=1
gpgcheck=0
sslverify=0
```

CLI を使用してデジタル署名をインポートする手順については、『Cisco Nexus 9000 Series NX-OS Software Upgrade and Downgrade Guide, Release 9.2(x)』の「Using Install CLIs for Digital Signature Support」を参照してください。

フルインストールログを使用した yum dnf を使用した RPM のインストールの例。

例：

```
bash-4.2# dnf install splunkforwarder
Loaded plugins: downloadonly, importpubkey, localrpmDB, patchaction, patching, protect-packages
Setting up Install Process
Resolving Dependencies
--> Running transaction check
---> Package splunkforwarder.x86_64 0:6.2.3-264376 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================================
 Package              Arch                 Version           Repository             Size
================================================================================================
Installing:
splunkforwarder       x86_64               6.2.3-264376      open-nxos              13 M

Transaction Summary
================================================================================================
Install       1 Package

Total size: 13 M
Installed size: 34 M
Is this ok [y/N]: y
Downloading Packages:
Running Transaction Check
Running Transaction Test
Transaction Test Succeeded
Running Transaction
  Installing : splunkforwarder-6.2.3-264376.x86_64
                                                                                1/1
complete

Installed:
  splunkforwarder.x86_64 0:6.2.3-264376


Complete!
bash-4.2#
```

パッケージが正常にインストールされたかどうかをスイッチに照会し、そのプロセスまたはサービスが稼働していることを確認する例。

例：

```
bash-4.2# dnf info splunkforwarder
Loaded plugins: downloadonly, importpubkey, localrpmDB, patchaction, patching, protect-packages
Fretta              |  951 B     00:00 ...
```

```
groups-repo          | 1.1 kB     00:00 ...
localdb              |  951 B     00:00 ...
patching             |  951 B     00:00 ...
thirdparty           |  951 B     00:00 ...
Installed Packages
Name         : splunkforwarder
Arch         : x86_64
Version      : 6.2.3
Release      : 264376
Size         : 34 M
Repo         : installed
From repo    : open-nxos
Summary      : SplunkForwarder
License      : Commercial
Description : The platform for machine data.
```

永続的なサードパーティ RPM

次に、永続的なサードパーティ RPM の背後にあるロジックを示します。

- ローカルリポジトリは、永続的なサードパーティ RPM 専用です。**dnf** /etc/yum/repos.d/ thirdparty.repo は /bootflash/.rpmstore/ thirdparty を指します。
- コマンドを入力するたびに、RPM のコピーが //bootflash/.rpmstore/ thirdparty に保存されます。**dnf install third-party.rpm**
- リブート中に、サードパーティ製リポジトリ内のすべてのRPMがスイッチに再インストールされます。
- /etc 構成ファイルの変更は、/bootflash/.rpmstore/config/etc の下に保持され、/etc での起動時に再生されます。
- /etc ディレクトリに作成されたスクリプトは、リロード後も保持されます。たとえば、/etc/init.d/ の下に作成されたサードパーティのサービススクリプトは、リロード中にアプリケーションを起動します。

（注） iptables のルールは、bash シェルで変更された場合、再起動後は保持されません。

変更した iptables を永続化するには、「」を参照してください。
リロード間で Iptable を永続化する

VSH からの RPM のインストール

パッケージの追加

NX-OS 機能 RPM は、VSH CLI を使用してインストールすることもできます。

手順の概要

1. **show install package**
2. **install add ?**
3. **install add** *rpm-packagename*

手順の詳細

手順

	コマンドまたはアクション	目的
ステップ 1	**show install package**	すでに存在するパッケージとバージョンを表示します。
ステップ 2	**install add ?**	サポートされている URI を決定します。
ステップ 3	**install add** *rpm-packagename*	The **install add** コマンドは、ローカル ストレージ デバイスまたは、ネットワーク サーバーへパッケージ ファイルをコピーします。

例

次に、Chef RPM をアクティブにする例を示します：

```
switch# show install package
switch# install add ?
WORD        Package name
bootflash:  Enter package uri
ftp:        Enter package uri
http:       Enter package uri
modflash:   Enter package uri
scp:        Enter package uri
sftp:       Enter package uri
tftp:       Enter package uri
usb1:       Enter package uri
usb2:       Enter package uri
volatile:   Enter package uri
switch# install add
bootflash:chef-12.0.0alpha.2+20150319234423.git.1608.b6eb10f-1.el5.x86_64.rpm
[####################] 100%
Install operation 314 completed successfully at Thu Aug  6 12:58:22 2015
```

次のタスク

パッケージをアクティブ化する準備ができたら、パッケージのアクティブ化 （6 ページ） に移動します。

（注） RPM パッケージの追加とアクティブ化は、次の 1 つのコマンドで実行できます。

```
switch#
install add bootflash:chef-12.0.0alpha.2+20150319234423.git.1608.b6eb10f-1.el5.x86_64.rpm
 activate
```

パッケージのアクティブ化

始める前に

RPM は事前に追加しておく必要があります。

手順の概要

1. **show install inactive**
2. **install activate** *rpm-packagename*

手順の詳細

手順

	コマンドまたはアクション	目的
ステップ 1	**show install inactive**	追加されていても、アクティブ化されていないパッケージのリストを表示します。
ステップ 2	**install activate** *rpm-packagename*	パッケージをアクティブ化します。

例

次に、パッケージをアクティブ化する例を示します：

```
switch# show install inactive
Boot image:
        NXOS Image: bootflash:///yumcli6.bin

Inactive Packages:
        sysinfo-1.0.0-7.0.3.x86_64
Loaded plugins: downloadonly, importpubkey, localrpmDB, patchaction, patching,
              : protect-packages
Available Packages
chef.x86_64       12.0.0alpha.2+20150319234423.git.1608.b6eb10f-1.el5 thirdparty
eigrp.lib32_n9000 1.0.0-r0                                            groups-rep
o
```

```
sysinfo.x86_64    1.0.0-7.0.3                                              patching
switch# install activate chef-12.0-1.el5.x86_64.rpm
[####################] 100%
Install operation completed successfully at Thu Aug  6 12:46:53 2015
```

パッケージの非アクティブ化

手順の概要

1. **install deactivate** *package-name*

手順の詳細

手順

	コマンドまたはアクション	目的
ステップ 1	**install deactivate** *package-name*	RPM パッケージを非アクティブ化します。

例

次に、Chef RPM パッケージを非アクティブ化する例を示します。

```
switch# install deactivate chef
```

パッケージの削除

始める前に

パッケージを削除する前に非アクティブ化します。非アクティブ化された RPM パッケージのみ削除できます。

手順の概要

1. **install remove** *package-name*

手順の詳細

手順

	コマンドまたはアクション	目的
ステップ 1	**install remove** *package-name*	RPM パッケージを削除します。

例

次に、Chef RPM パッケージを削除する例を示します。

```
switch# install remove chef-12.0-1.el5.x86_64.rpm
```

インストール済みパッケージの表示

手順の概要

1. **show install packages**

手順の詳細

手順

	コマンドまたはアクション	目的
ステップ 1	**show install packages**	インストールされているパッケージのリストを表示します。

例

次に、インストールされているパッケージのリストを表示する例を示します。

```
switch# show install packages
```

詳細ログの表示

手順の概要

1. **show tech-support install**

手順の詳細

手順

	コマンドまたはアクション	目的
ステップ 1	**show tech-support install**	詳細ログを表示します。

例

次の例は、詳細ログを表示する方法を示しています。

```
switch# show tech-support install
```

パッケージのアップグレード

手順の概要

1. インストール 追加 *package-name* アクティベート アップグレード

手順の詳細

手順

	コマンドまたはアクション	目的
ステップ 1	インストール 追加 *package-name* アクティベート アップグレード	パッケージをアップグレードします。

例

次に、パッケージをアップグレードする例を示します：

```
switch# install add bootflash:bgp-1.0.1-r0.lib32_n9000.rpm activate ?
downgrade  Downgrade package
forced     Non-interactive
upgrade    Upgrade package
switch# install add bootflash:bgp-1.0.1-r0.lib32_n9000.rpm activate upgrade
[####################] 100%
Install operation completed successfully at Thu Aug  6 12:46:53 2015
```

パッケージのダウングレード

手順の概要

1. インストール 追加 *package-name* アクティベート ダウングレード

手順の詳細

手順

	コマンドまたはアクション	目的
ステップ 1	インストール 追加 *package-name* アクティベート ダウングレード	パッケージをダウングレードします。

例

次の例は、パッケージをダウングレードする方法を示しています。

```
switch# install add bootflash:bgp-1.0.1-r0.lib32_n9000.rpm activate ?
downgrade  Downgrade package
forced     Non-interactive
upgrade    Upgrade package
switch# install add bootflash:bgp-1.0.1-r0.lib32_n9000.rpm activate downgrade
[####################] 100%
Install operation completed successfully at Thu Aug  6 12:46:53 2015
```

サードパーティ製アプリケーション

NX-OS

NX-API REST API オブジェクト モデルの仕様の詳細については、https://developer.cisco.com/docs/nx-api-dme-model-9-3-1-reference/ を参照してください。

DevOps 構成管理ツール

DevOps 構成管理ツールについては、次のリンクを参照してください。

- Ansible 2.0 リリース（Nexus サポート）、Ansible リリース インデックス
- Ansible NX-OS サンプル モジュール、Ansible NX-OS サンプル モジュール
- Puppet、Puppet Forge Cisco Puppet
- Cisco Puppet モジュール（Git）Cisco Network Puppet モジュール
- Chef、Chef Supermarket Cisco クックブック
- Cisco Chef クックブック（Git）Cisco Network Chef クックブック

V9K

ESX5.1/5.5、VirtualBox、Fusion、およびKVMの場合、仮想Nexus 9000スイッチをダウンロードするには、https://software.cisco.com/portal/pub/download/portal/select.html?&mdfid=286312239&flowid=81422&softwareid=282088129に移動します。

自動化ツールの教育コンテンツ

OpenNX-OSアーキテクチャと自動化に関する無料の書籍については、次を参照してください。http://www.cisco.com/c/dam/en/us/td/docs/switches/datacenter/nexus9000/sw/open_nxos/programmability/guide/Programmability_Open_NX-OS.pdf

collectd

collectdは、システムパフォーマンスの統計情報を定期的に収集し、RRDファイルなどの値を保存する複数の手段を提供するデーモンです。これらの統計情報を使用して、現在のパフォーマンスのボトルネック（パフォーマンス分析など）を見つけたり、将来のシステム負荷を予測したりできます（つまり、キャパシティプランニング）。

詳細については、https://collectd.org を参照してください。

Ganglia

Gangliaは、クラスタやグリッドなどのハイパフォーマンスコンピューティングシステム向けのスケーラブルな分散モニタリングシステムです。これは、クラスタのフェデレーションを対象とした階層設計に基づいています。データ表現のためのXML、コンパクトでポータブルなデータ転送のためのXDR、データストレージと可視化のためのRRDtoolなど、広く使用されているテクノロジーを活用します。設計されたデータ構造とアルゴリズムを使用して、ノードあたりのオーバーヘッドを低く抑え、同時実行性を高めます。この実装は堅牢であり、広範なオペレーティングシステムとプロセッサアーキテクチャに移植されており、現在、世界中の何千ものクラスタで使用されています。世界中の大学キャンパス間でクラスタをリンクするために使用されており、2000ノードのクラスタを処理するように拡張できます。

詳細については、http://ganglia.info を参照してください。

Iperf

Iperfは、TCPおよびUDPの最大帯域幅パフォーマンスを測定するためにNLANR/DASTによって開発されました。Iperfを使用すると、さまざまなパラメータとUDP特性を調整できます。Iperfは、帯域幅、遅延ジッターとデータグラム損失を報告します。

詳細については、http://sourceforge.net/projects/iperf/またはhttp://iperf.sourceforge.netを参照してください。

LLDP

リンク層検出プロトコル（LLDP）は、EDPやCDPなどの独自のリンク層プロトコルに代わるように設計された業界標準プロトコルです。LLDPの目的は、隣接するネットワークデバイスにリンク層通知を配信するための、ベンダー間互換性のあるメカニズムを提供することです。

詳細については、「https://vincentbernat.github.io/lldpd/index.html」を参照してください。

Nagios

Nalios は、Nadios Remote Plug-in Executor（NRPE）およびSSHまたはSSLトンネルを介して以下をモニターするオープンソースソフトウェアです。

- ICMP、SNMP、SSH、FTP、HTTP などによるネットワーク サービス
- CPU 負荷、ディスク使用率、システム ログなどのホスト リソース
- サーバー、スイッチ、アプリケーションのアラート サービス
- [サービス（Services）]

詳細については、「https://www.nagios.org/」を参照してください。

OpenSSH

OpenSSHは、盗聴、接続ハイジャック、およびその他の攻撃を排除するために、すべてのトラフィック（パスワードを含む）を暗号化するSSH接続ツールのオープンソース バージョンです。OpenSSHは、セキュアなトンネリング機能と複数の認証方式を提供し、すべてのSSHプロトコル バージョンをサポートします。

詳細については、「http://www.openssh.com」を参照してください。

Quagga

Quaggaは、さまざまなルーティング プロトコルを実装するネットワーク ルーティング ソフトウェア スイートです。Quaggaデーモンは、ネットワーク アクセス可能CLI（「vty」という）を使用して構成できます。

（注） Quagga BGP のみが検証されています。

詳細については、「http://www.nongnu.org/quagga/」を参照してください。

スプランク

Splunk は、Web ベースのデータ収集、分析、およびモニタリング ツールであり、ユースケースに合わせて検索、可視化、および事前にパッケージ化されたコンテンツを備えています。raw

データは、Splunk Universal Forwarder を使用して Splunk サーバーに送信されます。ユニバーサルフォワーダは、リモートソースからの信頼性の高いセキュアなデータ収集を可能にし、そのデータをインデックス作成と統合のために Splunk Enterprise に転送します。数万のリモート システムに拡張でき、パフォーマンスへの影響を最小限に抑えながらテラバイト単位のデータを収集できます。

詳細については、http://www.splunk.com/en_us/download/universal-forwarder.html を参照してください。

tcollector

tcollectorは、ローカル コレクタからデータを収集し、そのデータをオープン時系列データベース（OpenTSDB）にプッシュするクライアント側プロセスです。

tcollector には次の機能があります。

- データ コレクタを実行し、データを照合します。
- 時系列データベース（TSD）への接続を管理します。
- コレクタに TSD コードを埋め込む必要がなくなります。
- 繰り返される値の重複を排除します。
- ワイヤ プロトコル作業を処理します。

詳細については、http://opentsdb.net/docs/build/html/user_guide/utilities/tcollector.html を参照してください。

tcpdump

tcpdump は、ネットワーク インターフェイス上の Boolean 式に一致するパケットの内容に関する説明を出力する CLI アプリケーションです。説明の前にタイムスタンプが表示されます。デフォルトでは、午前 0 時からの時間、分、秒、および小数点以下の秒として出力されます。

tcpdump は、次のフラグを使用して実行できます。

- -w：後で分析するためにパケット データをファイルに保存します。
- -r：ネットワーク インターフェイスからパケットを読み取るのではなく、保存されたパケット ファイルから読み取ります。
- -V：保存されたパケット ファイルのリストを読み取ります。

いずれの場合も、tcpdump は式にマッチするパケットだけを処理します。

詳細については、「http://www.tcpdump.org/manpages/tcpdump.1.html」を参照してください。

TShark

TSharkは、CLIのネットワークプロトコルアナライザです。Tsharkを使用すると、ライブネットワークからパケットデータをキャプチャしたり、以前に保存したキャプチャファイルからパケットを読み取ったりできます。これらのパケットのデコードされた形式を標準出力に出力するか、パケットをファイルに書き込むことができます。TSharkのネイティブ キャプチャ ファイル フォーマットは、pcapです。このフォーマットは、**tcpdump**と他のツールに使用されています。TSharkは、cap_net_adminファイル機能を削除した後、ゲストシェル内で使用できます。

```
setcap
 cap_net_raw=ep /sbin/dumpcap
```

（注） このコマンドは、ゲストシェル内で実行する必要があります。

詳細については、「https://www.wireshark.org/docs/man-pages/tshark.html」を参照してください。

サードパーティ製アプリケーション（TPA）のホスティング

Cisco NX-OSリリース10.6(2)以降、Cisco Nexus 9000スイッチは、スイッチ上のLinuxコンテナでのサードパーティアプリケーション（TPA）の直接的なホストをサポートしています。この機能により、管理者および自動化システムは、外部サーバーを必要とせずに、スイッチ上でコンテナ化されたアプリケーションをネイティブに展開、管理、およびモニタできます。

サードパーティ製アプリケーションのホスティングの利点

この機能は、複数の利点を提供します：

- カスタムおよびベンダーアプリケーションの柔軟なデプロイメント：Linuxベースのアプリケーション（DHCP/TFTPサーバー、syslogコレクタ、モニタリング ツールなど）をスイッチ上で直接コンテナ内で実行でき、ネットワークエッジで幅広い使用例をサポートします。
- 統合管理と自動化：TPAコンテナを構成、監視、管理するためのNX- OS CLIコマンドとプログラム可能なAPI（gNMIやgNOIなど）を提供し、自動化されたワークフローやバックエンド システムとの統合を可能にします。
- 運用の可視性とトラブルシューティング：リアルタイムのステータス、リソース使用率、および実行中のコンテナのログ取得を提供し、トラブルシューティングとパフォーマンス分析を簡素化します。
- 永続的で信頼性の高いアプリケーション運用：重要なアプリケーションをデバイスのアップグレードやリブート後に自動的に再起動するように設定でき、継続的で自動化されたサービス提供をサポートします。

- セキュアでコンプライアンスに準拠した展開のサポート：VRF 対応のデプロイメント、セキュアな通信、設定とログイン情報の永続ストレージなど、シスコのセキュリティと規則遵守のベストプラクティスに準拠しています。
- エッジとクラウドの統合向けに設計：オンボックス アプリケーション ホスティングとクラウドネイティブ管理が必要な Google Distributed Cloud Edge（GDCE）展開などのユースケースに対応します。

コンテナベースのサードパーティ アプリケーション（TPA）ホスティングの仕組み

Cisco NX-OS リリース 10.6(2) 以降、 Cisco Nexus 9000 スイッチは、統合された Docker エンジンを使用して、サードパーティ アプリケーション（TPA）をスイッチ上で Linux コンテナとして直接ホストできます。

この機能は、NX- OS オペレーティング システム、Docker エンジン、および AppManager 間の統合を活用して、TPA ホスティングのスイッチ上でのコンテナ ライフサイクル管理を提供します。

process_summary

次のコンポーネントは、コンテナベースの TPA ホスティングで重要な役割を果たします。

- NX- OS：コンテナ化されたアプリケーションの機能の有効化、構成、および監視を管理します。
- Docker エンジン：コンテナ ランタイム環境を提供し、 Linux アプリケーション コンテナのライフサイクルを管理します。
- AppManager：コンテナ制御、ライフサイクル アクション、およびステータス レポートのために NX- OSと Docker 間を調整します。

このプロセスにより、Linux コンテナベースのサードパーティ アプリケーションのセキュアで永続的な自動ホスティングがスイッチ上で直接実行できるようになります。

process_workflow

コンテナベースのサードパーティ製アプリケーション（TPA）をホストおよび管理するプロセスには、次の段階が含まれます。

1. 機能の有効化：TPA 機能を有効にすると、Docker エンジンが NX- OS 内のマネージド サービスとして起動します。
2. 内部通信：NX- OS と Docker エンジン間のすべての通信に対してループバック インターフェイスが構成され、コンテナのトラフィック分離が保証されます。
3. コンテナのストレージと起動：アプリケーション コンテナ イメージがスイッチに保存され、TPA 構成に従って Docker エンジンによって起動されます。
4. 永続性：永続ストレージ（ホストマウント ボリューム）により、コンテナ データとログイン情報が再起動やアップグレード後も存続します。

5. ライフサイクル管理：AppManagerおよび関連するNX-OSプロセスは、コンテナの開始、停止、再起動、およびステータスの監視を制御します。
6. モニタリングとロギング：NX-OSには、コンテナ ステータス、リソース使用率、およびDockerとコンテナからのログを取得するためのコマンドが用意されています。
7. 自動化とAPI：TPAホスティングは、自動化と外部管理のために、NX-OS CLIおよびプログラム可能なAPI（gNMI/gNOIなど）と統合されています。
8. セキュリティと分離： VRF認識型通信とUNIXドメイン ソケットを使用して、スイッチ内でのセキュアな分離操作を実現します。

注意事項と制約事項

- NX-OSとDockerコンテナ間のトランスポート インターフェイスとして使用できるのは、ループバック インターフェイスのみです。このインターフェイスは、TPA通信用に明示的に設定する必要があります。
- 最適なパフォーマンスと信頼性を確保するために、アプリケーション コンテナを実行する場合は、少なくとも6つのCPUコアと64 GBのメモリを備えたスイッチ プラットフォームを使用することをお勧めします。
- 設定ファイル、ログイン情報、および永続的なアプリケーションデータを、専用ボリュームまたはホストマウントディレクトリに保存して、デバイスの再起動やアップグレード後もデータが保持されるようにします。
- 重要なアプリケーション コンテナは、システムのアップグレードまたはリブート後に自動的に再起動するように構成し、重要なサービスの継続的な運用を確保する必要があります。
- コンテナの高度なモニタリングとパフォーマンスの最適化は、現在の実装には含まれていません。ユーザーは、基本的なリソースおよびパフォーマンスの追跡にネイティブのDockerコマンドを使用できます。
- サポートされているDockerのバージョンは20.10.25-ceで、containerd v1.6.19です。

サードパーティ製アプリケーション（TPA）コンテナ ホスティングの構成

Cisco Nexus 9000スイッチでコンテナベースのサードパーティ アプリケーション（TPA）ホスティングを有効にし、設定するには、次の手順に従います。

Before you begin

Docker サービスの通信に使用するループバック インターフェイスを特定します。

手順

ステップ1 TPA機能を有効化します。

例：

```
feature dockerbox
```

このコマンドは、スイッチで Docker ベースのサードパーティ アプリケーション ホスティングを有効にします。

ステップ 2 TPA コンフィギュレーション モードに入ります。

例：

```
dockerbox
```

このコマンドにより、スイッチが TPA（Docker Box）構成モードになります。

ステップ 3 Docker サービス通信のトランスポート インターフェイスを構成します。

例：

```
transport-interface <loopback-interface>
```

Docker サービス通信では、ループバック インターフェイスのみがサポートされます。
loopback-interface は、選択したループバック インターフェイスで置き換えます。

コンテナベースの TPA ホスティングが有効になり、設定されました。スイッチでアプリケーション コンテナを展開および管理できるようになりました。

次のタスク

show dockerbox status および関連するコマンドで、設定を確認します。コマンド例と予想される出力については、検証のセクションを参照してください。

サードパーティ製アプリケーション（TPA）コンテナ ホスティングの確認

Cisco Nexus 9000 スイッチでのサードパーティ アプリケーション（TPA）コンテナ ホスティングのステータスと動作を確認するには、次の手順に従います。

これらの検証手順により、NX OS CLIコマンドを使用して、TPA ホスティングとそのコンテナの正常性と動作をチェックできます。

Before you begin

TPA ホスティングがスイッチで構成され、有効になっていることを確認します。

手順

ステップ 1 TPA ホスティングのステータスを表示します。

例：

```
show dockerbox status
```

このコマンドは、TPA（Docker）サービスの現在のステータスを表示します。

```
<example output placeholder>
```

ステップ 2 内部状態およびデバッグ情報を表示します。

例：

```
show system internal dockerbox
```

このコマンドは、TPA ホスティングの詳細な内部状態情報を提供します。

```
<example output placeholder>
```

ステップ 3 イベント履歴を表示します。

例：

```
show system internal dockerbox event-history [error|debug|msgs]
```

TPA ホスティングに関連するエラーログ、デバッグメッセージ、またはメッセージ履歴を表示するには、このコマンドを使用します。

```
<example output placeholder>
```

ステップ 4 コンテナ情報を表示します。

例：

```
show system internal dockerbox containers [brief|stats|logs]
```

このコマンドは、概要、リソース統計、ログを含む、実行中のコンテナに関する情報を表示します。

```
<example output placeholder>
```

ステップ 5 TPA ホスティング ログを表示します。

例：

```
show system internal dockerbox log [errors]
```

このコマンドは、TPA ホスティングの Docker サービス ログを表示します。 **errors** オプションを使用すると、エラー メッセージのみを表示できます。

```
<example output placeholder>
```

翻訳について
このドキュメントは、米国シスコ発行ドキュメントの参考和訳です。リンク情報につきましては、日本語版掲載時点で、英語版にアップデートがあり、リンク先のページが移動/変更されている場合がありますことをご了承ください。あくまでも参考和訳となりますので、正式な内容については米国サイトのドキュメントを参照ください。