# Cisco NX-OS での Docker の使用


- Cisco NX-OS での Docker について （1 ページ）
- Docker の注意事項と制限事項 （2 ページ）
- Cisco NX-OS 内で Docker コンテナを設定するための前提条件 （2 ページ）
- Docker デーモンの開始 （3 ページ）
- 自動的に起動するように Docker を構成する （3 ページ）
- Docker コンテナの開始: ホスト ネットワーク モデル （4 ページ）
- Docker コンテナの開始: ブリッジ型ネットワーク モデル （5 ページ）
- Docker コンテナでのブートフラッシュおよび揮発性パーティションのマウント （6 ページ）
- 拡張 ISSU スイッチオーバーでの Docker デーモンの永続性の有効化 （7 ページ）
- Cisco Nexus Platform Switches Switchover 時に Docker デーモンの永続性を有効にする （8 ページ）
- Docker ストレージ バックエンドのサイズ変更 （9 ページ）
- Docker デーモンの停止 （11 ページ）
- Docker コンテナ セキュリティ （12 ページ）
- Docker のトラブルシューティング （14 ページ）


## Cisco NX-OS での Docker について

Dockerは、すべての依存関係とライブラリと共にパッケージ化された、コンテナー内で安全に分離されたアプリケーションを実行する方法を提供します。Dockerの詳細にを表示するためにhttps://docs.docker.com/を参照してください。

Cisco NX-OS リリース 9.2（1）以降、スイッチ上の Cisco NX-OS 内で Docker を使用するためのサポートが追加されました。

スイッチに含まれる Docker のバージョンは CE 18.09.0 です。Docker デーモンはデフォルトでは実行されていません。手動で起動するか、スイッチの起動時に自動的に再起動するように設定する必要があります。

このセクションでは、スイッチ環境の特定のコンテキストで Docker を有効にして使用する方法について説明します。一般的な Docker の使用方法と機能の詳細については、https://docs.docker.com/ にある Docker のドキュメントを参照してください。

# Docker の注意事項と制限事項

次に、スイッチ上の Cisco NX-OS で Docker を使用するためのガイドラインと制限事項を示します。

- Docker でサードパーティの DHCPD サーバーを実行している場合、SVI と一緒に使用すると、クライアントに到達するオファーで問題が発生する可能性があります。可能な回避策は、ブロードキャスト応答を使用することです。
- Docker 機能は、少なくとも 8 GB のシステム RAM を備えた スイッチでサポートされています。

# Cisco NX-OS 内で Docker コンテナを設定するための前提条件

スイッチの Cisco NX-OS で Docker を使用するための前提条件は次のとおりです：

- ホスト Bash シェルを有効にします。スイッチの Cisco NX-OS で Docker を使用するには、ホスト Bash シェルのルート ユーザーである必要があります：

```
switch# configure terminal
    Enter configuration commands, one per line. End with CNTL/Z.
    switch(config)# feature bash-shell
```

- スイッチが HTTP プロキシ サーバを使用するネットワーク内にある場合、`http_proxy` と `https_proxy` 環境変数を `/etc/sysconfig/docker` に構成する必要があります。例：

```
export http_proxy=http://proxy.esl.cisco.com:8080
export https_proxy=http://proxy.esl.cisco.com:8080
```

- スイッチのクロックが正しく設定されていることを確認してください。そうしないと、次のエラー メッセージが表示される場合があります：

```
x509: certificate has expired or is not yet valid
```

- ドメイン名とネーム サーバがネットワークに対して適切に構成されていること、および `/etc/resolv.conf` ファイルに反映されていることを確認します：

```
switch# conf t
     Enter configuration commands, one per line. End with CNTL/Z.
     switch(config)# vrf context management
     switch(config-vrf)# ip domain-name ?
     WORD Enter the default domain (Max Size 64)

     switch(config-vrf)# ip name-server ?
```

```
    A.B.C.D Enter an IPv4 address
    A:B::C:D Enter an IPv6 address

root@switch# cat /etc/resolv.conf
domain cisco.com #bleed
nameserver 171.70.168.183 #bleed
root@switch#
```

# Docker デーモンの開始

初めて Docker デーモンを開始すると、固定サイズのバックエンド ストレージ スペースがブートフラッシュの dockerpart と呼ばれるファイルに切り出され、次に /var/lib/docker にマウントされます。必要に応じて、Docker デーモンを初めて開始する前に /etc/sysconfig/docker を編集して、この領域のデフォルト サイズを調整できます。後で説明するように、必要に応じてこのストレージ スペースのサイズを変更することもできます。

Docker デーモンを開始するには：

**手順**

**ステップ 1** Bash を読み込み、スーパーユーザーになります。

```
switch# run bash sudo su -
```

**ステップ 2** Docker デーモンを起動します。

```
root@switch# service docker start
```

**ステップ 3** ステータスをチェックします。

```
root@switch# service docker status
dockerd (pid 3597) is running...
root@switch#
```

（注）
Docker デーモンを起動したら、ブートフラッシュ の dockerpart ファイルを削除したり、改ざんしたりしないでください。これは、docker の機能にとって重要であるからです。

```
switch# dir bootflash:dockerpart
2000000000 Mar 14 12:50:14 2018 dockerpart
```

# 自動的に起動するように Docker を構成する

スイッチの起動時に常に自動的に起動するように Docker デーモンを構成できます。

手順

**ステップ 1** Bash をロードしてスーパーユーザーになります。

```
switch# run bash sudo su -
```

**ステップ 2** chkconfig ユーティリティを使用して、Docker サービスを永続化します。

```
root@switch# chkconfig --add docker
root@n9k-2#
```

**ステップ 3** chkconfig ユーティリティを使用して、Docker サービスの設定を確認します。

```
root@switch# chkconfig --list | grep docker
docker 0:off 1:off 2:on 3:on 4:on 5:on 6:off
root@switch#
```

**ステップ 4** Docker が自動的に起動しないように構成を削除するには：

```
root@switch# chkconfig --del docker
root@switch# chkconfig --list | grep docker
root@switch#
```

# Docker コンテナの開始: ホスト ネットワーク モデル

Docker コンテナがデータ ポートと管理を含むすべてのホスト ネットワーク インターフェイスにアクセスできるようにする場合は、--network ホスト オプションを使用して Docker コンテナを起動します。コンテナ内のユーザーは、ip netns exec <net_namespace> <cmd> を使用して、/var/run/netns（Cisco NX-OS で設定されたさまざまな VRF に対応）でさまざまなネットワーク名前空間を切り替えることができます。

手順

**ステップ 1** Bash をロードしてスーパーユーザーになります。

```
switch# run bash sudo su -
```

**ステップ 2** Docker コンテナを開始します。

以下は、スイッチで Alpine Docker コンテナを起動し、すべてのネットワーク インターフェイスを表示する例です。コンテナは、デフォルトで管理ネットワークの名前空間で起動されます。

```
root@switch# docker run --name=alpinerun -v /var/run/netns:/var/run/netns:ro,rslave --rm --network
 host --cap-add SYS_ADMIN -it alpine
```

```
/ # apk --update add iproute2
fetch http://dl-cdn.alpinelinux.org/alpine/v3.7/main/x86_64/APKINDEX.tar.gz
fetch http://dl-cdn.alpinelinux.org/alpine/v3.7/community/x86_64/APKINDEX.tar.gz
(1/6) Installing libelf (0.8.13-r3)
(2/6) Installing libmnl (1.0.4-r0)
(3/6) Installing jansson (2.10-r0)
(4/6) Installing libnftnl-libs (1.0.8-r1)
(5/6) Installing iptables (1.6.1-r1)
(6/6) Installing iproute2 (4.13.0-r0)
Executing iproute2-4.13.0-r0.post-install
Executing busybox-1.27.2-r7.trigger
OK: 7 MiB in 17 packages
/ #
/ # ip netns list
management
default
/ #
/ # ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
inet 127.0.0.1/8 scope host lo
valid_lft forever preferred_lft forever
inet6 ::1/128 scope host
valid_lft forever preferred_lft forever
2: tunl0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN group default
link/ipip 0.0.0.0 brd 0.0.0.0
3: gre0@NONE: <NOARP> mtu 1476 qdisc noop state DOWN group default
link/gre 0.0.0.0 brd 0.0.0.0
...
/ #
/ # ip netns exec default ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
inet 127.0.0.1/16 scope host lo
valid_lft forever preferred_lft forever
2: dummy0: <BROADCAST,NOARP> mtu 1500 qdisc noop state DOWN group default
link/ether 42:0d:9b:3c:d4:62 brd ff:ff:ff:ff:ff:ff
3: tunl0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN group default
link/ipip 0.0.0.0 brd 0.0.0.0
...
```

# Docker コンテナの開始: ブリッジ型ネットワーク モデル

Dockerコンテナに外部ネットワーク接続（通常は管理インターフェースを介して）のみを許可し、特定のデータ ポートまたは他のスイッチ インターフェースへの可視性を必ずしも気にしない場合は、デフォルトの Docker ブリッジ ネットワーク モデルで Docker コンテナを開始できます。これは、ネットワーク名前空間の分離も提供するため、前のセクションで説明したホスト ネットワーキング モデルよりも安全です。

## 手順

**ステップ 1** Bash をロードしてスーパーユーザーになります。

```
switch# run bash sudo su -
```

**ステップ 2** Docker コンテナを開始します。

以下は、スイッチで Alpine Docker コンテナを開始し、iproute2 パッケージをインストールする例です。

```
root@switch# docker run -it --rm alpine
/ # apk --update add iproute2
fetch http://dl-cdn.alpinelinux.org/alpine/v3.7/main/x86_64/APKINDEX.tar.gz
fetch http://dl-cdn.alpinelinux.org/alpine/v3.7/community/x86_64/APKINDEX.tar.gz
(1/6) Installing libelf (0.8.13-r3)
(2/6) Installing libmnl (1.0.4-r0)
(3/6) Installing jansson (2.10-r0)
(4/6) Installing libnftnl-libs (1.0.8-r1)
(5/6) Installing iptables (1.6.1-r1)
(6/6) Installing iproute2 (4.13.0-r0)
Executing iproute2-4.13.0-r0.post-install
Executing busybox-1.27.2-r7.trigger
OK: 7 MiB in 17 packages
/ #
/ # ip netns list
/ #
```

**ステップ 3** ユーザー名前空間の分離を設定するかどうかを決定します。

ブリッジネットワーク モデルを使用するコンテナの場合、ユーザー名前空間の分離を設定して、セキュリティをさらに向上させることもできます。詳細については、「ユーザー[名前空間（namespace）]の分離による Docker コンテナの保護 （12 ページ）」を参照してください。

標準の Docker ポート オプションを使用して、sshd などのコンテナー内からサービスを公開できます。例：

```
root@switch# docker run -d -p 18877:22 --name sshd_container sshd_ubuntu
```

これにより、コンテナ内のポート 22 がスイッチのポート 18877 にマップされます。次の例に示すように、ポート 18877 を介してサービスに外部からアクセスできるようになりました。

```
root@ubuntu-vm# ssh root@ip_address -p 18887
```

# Dockerコンテナでのブートフラッシュおよび揮発性パーティションのマウント

Docker コンテナの run コマンドで -v /bootflash:/bootflash および -v /volatile:/volatile オプションを渡すことで、ブートフラッシュおよび揮発性パーティションを Docker コンテナに表示できます。これは、新しい NX-OS システム イメージをブートフラッシュにコピーするなど、コンテナ内のアプリケーションがホストと共有するファイルにアクセスする必要がある場合に役立ちます。

（注） この **-v** コマンド オプションを使用すると、任意のディレクトリをコンテナにマウントでき、NX-OS システムの動作に影響を与える可能性のある情報漏えいやその他のアクセスが発生する可能性があります。これを、NX-OS CLI を使用してすでにアクセス可能な /bootflash や /volatile などのリソースに制限します。

手順

**ステップ 1** Bash をロードしてスーパーユーザーになります。

```
switch# run bash sudo su -
```

**ステップ 2** Docker コンテナの実行コマンドに -v /bootflash:/bootflash および -v /volatile:/volatile オプションを渡します。

```
root@switch# docker run -v /bootflash:/bootflash -v /volatile:/volatile -it --rm alpine
/# ls /
bin        etc        media      root       srv        usr
bootflash  home       mnt        run        sys        var
dev        lib        proc       sbin       tmp        volatile
/ #
```

# 拡張 ISSU スイッチオーバーでの Docker デーモンの永続性の有効化

Docker デーモンと実行中のコンテナの両方を拡張 ISSU スイッチオーバーで持続させることができます。これが可能なのは、バックエンドの Docker ストレージが存在するブートフラッシュが同じであり、アクティブ スーパーバイザとスタンバイ スーパーバイザの両方で共有されるためです。

Docker コンテナは、切り替え中に中断（再起動）されるため、継続的に実行されません。

手順

**ステップ 1** Bash をロードしてスーパーユーザーになります。

```
switch# run bash sudo su -
```

**ステップ 2** スイッチオーバーを開始する前に、chkconfig ユーティリティを使用して Docker サービスを永続化します。

```
root@switch# chkconfig --add docker
root@n9k-2#
```

**ステップ 3** スイッチオーバー後にコンテナが自動的に再起動されるように、`--restart without-stopped` オプションを使用してコンテナを起動します。

次の例では、Alpine コンテナを開始し、明示的に停止するか、Docker を再起動しない限り、常に再起動するように構成します。

```
root@switch# docker run -dit --restart unless-stopped alpine
root@n9k-2#
```

Docker コンテナは、切り替え中に中断（再起動）されるため、継続的に実行されません。

# Cisco Nexus Platform Switches Switchover 時に Docker デーモンの永続性を有効にする

Docker デーモンと実行中のコンテナの両方を、個別のブートフラッシュ パーティションを持つ2つの個別の物理スーパーバイザ間のスイッチオーバーで持続させることができます。ただし、Cisco Nexus スイッチの場合、両方のスーパーバイザのブートフラッシュ パーティションは物理的に分離されています。したがって、スイッチオーバーを実行する前に、`dockerpart` ファイルをスタンバイ スーパーバイザに手動でコピーする必要があります。

## 手順

**ステップ 1** Bash を読み込みしてスーパーユーザーになります。

```
switch# run bash sudo su -
```

**ステップ 2** スイッチオーバー後にコンテナが自動的に再起動されるように、`--restart without-stopped` オプションを使用してコンテナを起動します。

次の例では、Alpine コンテナを開始し、明示的に停止するか、Docker を再起動しない限り、常に再起動するように構成します。

```
root@switch# docker run -dit --restart unless-stopped alpine
root@n9k-2#
```

Docker コンテナは切り替え中に中断（再起動）されるため、継続的に実行されないことに注意してください。

**ステップ 3** スイッチオーバーを開始する前に、`chkconfig` ユーティリティを使用して Docker サービスを永続化します。

```
root@switch# chkconfig --add docker
root@n9k-2#
```

**ステップ 4** Docker バックエンド ストレージ パーティションを現用系からスタンバイ スーパーバイザ ブートフラッシュにコピーします。

```
root@switch# service docker stop
Stopping dockerd: dockerd shutdown

root@switch# cp /bootflash/dockerpart /bootflash_sup-remote/

root@switch# service docker start
```

# Docker ストレージ バックエンドのサイズ変更

Docker デーモンを起動または使用した後、必要に応じて Docker バックエンド ストレージ スペースのサイズを増やすことができます。

## 手順

**ステップ 1** Guest Shell を無効にします。

ゲスト シェルを無効にしないと、サイズ変更が妨げられる可能性があります。

```
switch# guestshell disable
You will not be able to access your guest shell if it is disabled. Are you sure you want to disable
 the guest shell? (y/n) [n] y
switch# 2018 Mar 15 17:16:55 switch %$ VDC-1 %$ %VMAN-2-ACTIVATION_STATE: Deactivating virtual
service 'guestshell+'
2018 Mar 15 17:16:57 switch %$ VDC-1 %$ %VMAN-2-ACTIVATION_STATE: Successfully deactivated virtual
 service 'guestshell+'
```

**ステップ 2** Bash をロードしてスーパーユーザーになります。

```
switch# run bash sudo su -
```

**ステップ 3** 現在利用可能なストレージ容量に関する情報を取得します。

```
root@switch# df -kh /var/lib/docker
Filesystem Size Used Avail Use% Mounted on
/dev/loop12 1.9G 7.6M 1.8G 1% /var/lib/docker
root@n9k-2#
```

**ステップ 4** Docker デーモンを停止します。

```
root@switch# service docker stop
Stopping dockerd: dockerd shutdown
```

**ステップ 5** Docker バックエンド ストレージ スペース（`/bootflash/dockerpart`）の現在のサイズに関する情報を取得します。

```
root@switch# ls -l /bootflash/dockerpart
-rw-r--r-- 1 root root 2000000000 Mar 15 16:53 /bootflash/dockerpart
```

```
root@n9k-2#
```

**ステップ 6** Docker バックエンド ストレージ スペースのサイズを変更します。

たとえば、次のコマンドはサイズを 500 メガバイト増やします。

```
root@switch# truncate -s +500MB /bootflash/dockerpart
root@n9k-2#
```

**ステップ 7** Dockerバックエンドストレージスペースのサイズに関する最新情報を取得して、サイズ変更プロセスが正常に完了したことを確認します。

たとえば、次の出力は、Docker バックエンド ストレージのサイズが 500 メガバイト増加したことを確認します。

```
root@switch# ls -l /bootflash/dockerpart
-rw-r--r-- 1 root root 2500000000 Mar 15 16:54 /bootflash/dockerpart
root@n9k-2#
```

**ステップ 8** /bootflash/dockerpart のファイル システムのサイズを確認します。

```
root@switch# e2fsck -f /bootflash/dockerpart
e2fsck 1.42.9 (28-Dec-2013)
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
/bootflash/dockerpart: 528/122160 files (0.6% non-contiguous), 17794/488281 blocks
```

**ステップ 9** /bootflash/dockerpart のファイル システムのサイズを変更します。

```
root@switch# /sbin/resize2fs /bootflash/dockerpart
resize2fs 1.42.9 (28-Dec-2013)
Resizing the filesystem on /bootflash/dockerpart to 610351 (4k) blocks.
The filesystem on /bootflash/dockerpart is now 610351 blocks long.
```

**ステップ 10** /bootflash/dockerpart のファイル システムのサイズを再度チェックして、ファイル システムのサイズが正常に変更されたことを確認します。

```
root@switch# e2fsck -f /bootflash/dockerpart
e2fsck 1.42.9 (28-Dec-2013)
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
/bootflash/dockerpart: 528/154736 files (0.6% non-contiguous), 19838/610351 blocks
```

**ステップ 11** Daemon デーモンを再起動します。

```
root@switch# service docker start
Updating certificates in /etc/ssl/certs...
0 added, 0 removed; done.
Running hooks in /etc/ca-certificates/update.d...
done.
Starting dockerd with args '--debug=true':
```

**ステップ 12** 使用可能なストレージ領域の大きさを確認します。

```
root@switch# df -kh /var/lib/docker
Filesystem Size Used Avail Use% Mounted on
/dev/loop12 2.3G 7.6M 2.3G 1% /var/lib/docker
```

**ステップ 13** BASH シェルを終了します。

```
root@switch# exit
logout
switch#
```

**ステップ 14** Guest Shell を有効にします。

```
switch# guestshell enable

switch# 2018 Mar 15 17:12:53 switch %$ VDC-1 %$ %VMAN-2-ACTIVATION_STATE: Activating virtual
service 'guestshell+'
switch# 2018 Mar 15 17:13:18 switch %$ VDC-1 %$ %VMAN-2-ACTIVATION_STATE: Successfully activated
 virtual service 'guestshell+'
```

# Docker デーモンの停止

Dockerを今後使用しない場合は、このトピックの手順に従ってDockerデーモンを停止します。

### 手順

**ステップ 1** Bash をロードしてスーパーユーザーになります。

```
switch# run bash sudo su -
```

**ステップ 2** Docker デーモンを停止します。

```
root@switch# service docker stop
Stopping dockerd: dockerd shutdown
```

**ステップ 3** Docker デーモンが停止していることを確認します。

```
root@switch# service docker status
dockerd is stopped
root@switch#
```

（注）
必要に応じて、この時点でブートフラッシュの `dockerpart` ファイルを削除することもできます。

```
switch# delete bootflash:dockerpart
Do you want to delete "/dockerpart" ? (yes/no/abort) y
```

```
switch#
```

# Docker コンテナ セキュリティ

Docker コンテナのセキュリティに関する推奨事項は次のとおりです。

- 可能であれば、別のユーザー［名前空間（namespace）］で実行します。
- 可能であれば、別のネットワーク［名前空間（namespace）］で実行します。
- cgroup を使用して技術情報を制限します。既存の cgroup（ext_ser）が作成され、ホストされているアプリケーションを、プラットフォーム チームがスイッチで実行される追加のソフトウェアに対して妥当と見なしたものに制限します。Docker では、これを使用して、コンテナごとの技術情報を制限できます。
- 不要な POSIX 機能を追加しないでください。

## ユーザー[名前空間（namespace）]の分離による Docker コンテナの保護

ブリッジ ネットワーク モデルを使用するコンテナの場合、ユーザー名前空間の分離を設定して、セキュリティをさらに向上させることもできます。詳細については、「https://docs.docker.com/engine/security/userns-remap/」を参照してください。

### 手順

**ステップ 1** システムに dockremap グループがすでに存在するかどうかを確認します。

dockremap ユーザーは、デフォルトでシステムにすでに設定されている必要があります。dockremap グループがまだ存在しない場合は、次の手順に従って作成します。

a) 次のコマンドを入力して dockremap グループを作成します。

```
root@switch# groupadd dockremap -r
```

b) dockremap ユーザーを作成します（まだ存在していない場合）。

```
root@switch# useradd dockremap -r -g dockremap
```

c) dockremap グループと dockremap ユーザーが正常に作成されたことを確認します。

```
root@switch# id dockremap
uid=999(dockremap) gid=498(dockremap) groups=498(dockremap)
```

```
root@switch#
```

**ステップ 2** 再マップされた必要な ID と範囲を /etc/subuid と /etc/subgid に追加します。

例：

```
root@switch# echo "dockremap:123000:65536" >> /etc/subuid
root@switch# echo "dockremap:123000:65536" >> /etc/subgid
```

**ステップ 3** テキスト エディタを使用して、--userns-remap=default オプションを /etc/sysconfig/docker ファイルの other_args フィールドに追加します。

例：

```
other_args="-debug=true --userns-remap=default"
```

**ステップ 4** [サービス ドッカー [re]start（service docker [re]start）] を使用して、Docker デーモンを再起動するか、まだ実行されていない場合は起動します。

例：

```
root@switch# service docker [re]start
```

ユーザー名前空間の分離によるコンテナの構成と使用の詳細については、https://docs.docker.com/engine/security/userns-remap/で Docker のドキュメントを参照してください。

## cgroup パーティションの移動

サードパーティ サービスの cgroup パーティションは ext_ser で、CPU 使用率をコアあたり 25% に制限します。この ext_ser パーティションで Docker コンテナを実行することをお勧めします。

--cgroup-parent=/ext_ser/ オプションを指定せずに Docker コンテナを実行すると、最大 100% のホスト CPU アクセスが可能になり、Cisco NX-OS の通常の動作を妨げる可能性があります。

### 手順

**ステップ 1** Bash をロードしてスーパーユーザーになります。

```
switch# run bash sudo su -
```

**ステップ 2** ext_ser パーティションで Docker コンテナを実行します。

例：

```
root@switch# docker run --name=alpinerun -v /var/run/netns:/var/run/netns:ro,rslave --rm --network
 host --cgroup-parent=/ext_ser/ --cap-add SYS_ADMIN -it alpine
```

```
/ #
```

# Docker のトラブルシューティング

これらのトピックでは、Docker コンテナで発生する可能性のある問題について説明し、考えられる解決策を提供します。

## Docker の起動が機能不全になる

**[問題：（Problem:）]** Docker の起動に失敗し、次のようなエラー メッセージが表示されます：

```
switch# run bash
bash-4.3$ service docker start
Free bootflash: 39099 MB, total bootflash: 51771 MB
Carving docker bootflash storage: 2000 MB
2000+0 records in
2000+0 records out
2000000000 bytes (2.0 GB) copied, 22.3039 s, 89.7 MB/s
losetup: /dev/loop18: failed to set up loop device: Permission denied
mke2fs 1.42.9 (28-Dec-2013)
mkfs.ext4: Device size reported to be zero.  Invalid partition specified, or
        partition table wasn't reread after running fdisk, due to
        a modified partition being busy and in use.  You may need to reboot
        to re-read your partition table.

Failed to create docker volume
```

**[考えられる原因：（Possible Cause:）]** root ユーザーではなく、管理ユーザーとして Bash を実行している可能性があります。

**[解決策：（Solution:）]** root ユーザーではなく、管理ユーザーとして Bash を実行しているかどうかを確認します。

```
bash-4.3$ whoami
admin
```

Bash を終了し、ルート ユーザーとして Bash を実行します：

```
bash-4.3$ exit
switch# run bash sudo su -
```

## ストレージが不足しているため、Docker が起動に失敗する

**問題**：ブートフラッシュ ストレージが不足しているため、Docker の起動に失敗し、次のようなエラー メッセージが表示されます。

```
root@switch# service docker start
Free bootflash: 790 MB, total bootflash: 3471 MB
```

```
Need at least 2000 MB free bootflash space for docker storage
```

**考えられる原因**：十分な空きブートフラッシュ ストレージがない可能性があります。

**解決策**：スペースを解放するか、必要に応じて `/etc/sysconfig/docker` の変数 *_dockerstrg* 値を調整してから、Docker デーモンを再起動します。

```
root@switch# cat /etc/sysconfig/docker
# Replace the below with your own docker storage backend boundary value (in MB)
# if desired.
boundary_dockerstrg=5000

# Replace the below with your own docker storage backend values (in MB) if
# desired. The smaller value applies to platforms with less than
# $boundary_dockerstrg total bootflash space, the larger value for more than
# $boundary_dockerstrg of total bootflash space.
small_dockerstrg=300
large_dockerstrg=2000
```

# Docker Hub からのイメージのプルの失敗（509 証明書失効 エラー メッセージ）

**問題**：システムが Docker ハブからイメージをプルできず、次のようなエラー メッセージが表示されます。

```
root@switch# docker pull alpine
Using default tag: latest
Error response from daemon: Get https://registry-1.docker.io/v2/: x509: certificate has
 expired or is not yet valid
```

**[考えられる原因：（Possible Cause:）]**システム クロックが正しく設定されていない可能性があります。

**[解決策：（Solution:）]**クロックが正しく設定されているかどうかを確認します。

```
root@n9k-2# sh clock
15:57:48.963 EST Thu Apr 25 2002
Time source is Hardware Calendar
```

必要に応じて、時計をリセットします：

```
root@n9k-2# clock set hh:mm:ss { day month | month day } year
```

例：

```
root@n9k-2# clock set 14:12:00 10 feb 2018
```

# Docker Hubからのイメージのプルの失敗（クライアントタイムアウトエラー メッセージ）

**問題**：システムが Docker ハブからイメージをプルできず、次のようなエラー メッセージが表示されます。

```
root@switch# docker pull alpine
Using default tag: latest
Error response from daemon: Get https://registry-1.docker.io/v2/: net/http: request
canceled while waiting for connection (Client.Timeout exceeded while awaiting headers)
```

**考えられる原因**：プロキシまたは DNS 設定が正しく設定されていない可能性があります。

**解決策**：プロキシ設定を確認し、必要に応じて修正してから、Docker デーモンを再起動します。

```
root@switch# cat /etc/sysconfig/docker | grep proxy
#export http_proxy=http://proxy.esl.cisco.com:8080
#export https_proxy=http://proxy.esl.cisco.com:8080
root@switch# service docker [re]start
```

DNS 設定を確認し、必要に応じて修正してから、Docker デーモンを再起動します。

```
root@switch# cat /etc/resolv.conf
domain cisco.com #bleed
nameserver 171.70.168.183 #bleed
root@switch# # conf t
     Enter configuration commands, one per line. End with CNTL/Z.
     switch(config)# vrf context management
     switch(config-vrf)# ip domain-name ?
     WORD Enter the default domain (Max Size 64)

     switch(config-vrf)# ip name-server ?
     A.B.C.D Enter an IPv4 address
     A:B::C:D Enter an IPv6 address
root@switch# service docker [re]start
```

# スイッチのリロードまたはスイッチオーバーでDockerデーモンまたはコンテナが実行されない

**問題**：スイッチのリロードまたはスイッチオーバーを実行した後、Dockerデーモンまたはコンテナーが実行されません。

**考えられる原因**：Dockerデーモンが、スイッチのリロードまたはスイッチオーバーで持続するように構成されていない可能性があります。

**解決策**：Dockerデーモンが `chkconfig` コマンドを使用してスイッチのリロードまたはスイッチオーバーで持続するように構成されていることを確認してから、`--restart unless-stopped` オプションを使用して必要な Docker コンテナを開始します。たとえば、Alpine コンテナを開始するには：

```
root@switch# chkconfig --add docker
root@switch#
```

```
root@switch# chkconfig --list | grep docker
docker 0:off 1:off 2:on 3:on 4:on 5:on 6:off
root@switch# docker run -dit --restart unless-stopped alpine
```

# Docker ストレージ バックエンドのサイズ変更が失敗する

**問題**：Docker バックエンド ストレージのサイズを変更しようとして失敗しました。

**考えられる原因**：ゲスト シェルが無効になっていない可能性があります。

**解決策**：次のコマンドを使用して、ゲスト シェルが無効になっているかどうかを確認します。

```
root@switch# losetup -a | grep dockerpart
root@n9k-2#
```

ゲスト シェルが無効になっている場合、コマンドは出力を表示しません。

必要に応じて、次のコマンドを入力してゲスト シェルを無効にします。

```
switch# guestshell disable
```

それでも Docker バックエンド ストレージのサイズを変更できない場合は、`/bootflash/dockerpart` を削除し、`/etc/sysconfig/docker` の `[small_]large_dockerstrg` を調整してから、Docker を再度起動して、必要なサイズの新しい Docker パーティションを取得します。

# Docker コンテナがポートで着信トラフィックを受信しない

**問題**：Docker コンテナがポートで着信トラフィックを受信しません。

**考えられる原因**：Docker コンテナが kstack ポートではなく netstack ポートを使用している可能性があります。

**解決策**：Docker コンテナによって使用されるエフェメラル ポートが kstack の範囲内にあることを確認します。そうしないと、着信パケットがサービスのために netstack に送信され、ドロップされる可能性があります。

```
switch# show socket local-port-range
Kstack local port range (15001 - 58000)
Netstack local port range (58001 - 63535) and nat port range (63536 - 65535)
switch# conf t
Enter configuration commands, one per line. End with CNTL/Z.
switch(config)# sockets local-port-range <start_port> <end_port>
switch# run bash sudo su -
root@switch# cat /proc/sys/net/ipv4/ip_local_port_range
15001   58000
root@switch#
```

# Docker コンテナでデータ ポートと / または管理インターフェイスを表示できません

**問題**： Docker コンテナにデータ ポートまたは管理インターフェースが表示されません。

**解決方法**：

- `-v /var/run/netns:/var/run/netns:ro,rslave --network host` オプションを使用して、すべてのホスト名前空間がマップされたホスト ネットワーク名前空間で Docker コンテナが開始されていることを確認します。
- コンテナに入ると、デフォルトで管理ネットワークの名前空間に入ります。`ip netns` ユーティリティを使用して、データ ポート インターフェイスを持つデフォルト（`init`）ネットワーク名前空間に移動できます。`ip netns` ユーティリティは、dnf、apk などを使用してコンテナにインストールする必要がある場合があります。

# 一般的なトラブルシューティングのヒント

**[問題**：（**Problem:**）**]** 他のトラブルシューティング プロセスを使用しても解決されなかった Docker コンテナに関する他の問題があります。

**解決方法**：

- `/var/log/docker` で dockerd デバッグ出力を探して、何が問題なのかの手掛かりを見つけてください。
- スイッチに 8 GB 以上の RAM があることを確認します。Docker 機能は、RAM が 8 GB 未満のスイッチではサポートされていません。

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。