# NX-OS BashシェルでのDocker Composeのインストール

## 内容



## 概要

このドキュメントでは、NX-OS Bashシェル内にDocker Composeパッケージをインストールする手順について説明します。

Cisco Nexus 3000および9000シリーズデバイスは、NX-OSリリース9.2(1)以降のBashシェルでDocker機能をサポートしています。 Docker Composeドキュメントで説明されているように、「Composeはマルチコンテナdockerアプリケーションを定義および実行するためのツールです。Docker Composeを使用すると、アプリケーション開発者は、「docker-compose.yml」という名前の1つのYAMLファイル内でアプリケーションを構成するすべてのサービスを定義できます。 次に、1つのコマンドを使用して、これらのサービスをすべて作成、構築、開始できます。さらに、Docker Composeコマンドスイートから、すべてのサービスを停止および監視できます。

Docker機能はNX-OS Bashシェル内でネイティブにサポートされていますが、Docker Composeは個別にインストールする必要があります。

## 前提条件

### 要件

このドキュメントでは、Cisco NexusデバイスでBashシェルを有効にする必要があります。Bashシェルを有効にする手順については、『Cisco Nexus 9000シリーズNX-OSプログラマビリティガイド』のBashの章の「Bashへのアクセス」セクションを参照してください。

このドキュメントでは、ドメインホスト名をIPアドレスに解決できるDNSクライアントとしてBashシェルを設定する必要があります。Bashシェル内でDNSサーバを設定する手順については、ドキュメントを参照してください。

### 使用するコンポーネント

このドキュメントの情報は、次のソフトウェアとハードウェアのバージョンに基づいています。

- NX-OSリリース9.2(1)以降のNexus 9000プラットフォーム
- NX-OSリリース9.2(1)以降のNexus 3000プラットフォーム

このドキュメントの情報は、特定のラボ環境にあるデバイスに基づいて作成されたものです。このドキュメントで使用するすべてのデバイスは、初期（デフォルト）設定の状態から起動しています。対象のネットワークが実稼働中である場合には、どのようなコマンドについても、その潜在的な影響について確実に理解しておく必要があります。

# HTTP/HTTPSプロキシの設定

環境でHTTPまたはHTTPSプロキシを使用する必要がある場合は、Docker Composeをインストールする前に、Bashシェルがこれらのプロキシを使用するように設定する必要があります。

`run bash sudo su -`コマンドを使用して、rootユーザーとしてBashシェルにます。

```
Nexus# run bash sudo su -
root@Nexus#whoami
root
```

## HTTP/HTTPSプロキシの一時的な設定

このセッションのHTTP/HTTPSプロキシを一時的に設定するには、`export`して、「http_proxy」および「https_proxy」環境変数を定義します。この例を次に示します。「proxy.example-domain.com」は仮想プロキシサーバのホスト名です。

```
root@Nexus#export http_proxy=http://proxy.example-domain.com:80/
root@Nexus#export https_proxy=https://proxy.example-domain.com:80/
```

次に示すように、環境変数が`echo $http_proxyecho $https_proxy`れていることを確認します。

```
root@Nexus#echo $http_proxy
http://proxy.example-domain.com:80/ root@Nexus#echo $https_proxy
https://proxy.example-domain.com:80/
```

これらの環境変数に割り当てられた値は、セッションが終了するとクリアされ、Bashシェルが入力されるたびに再設定が必要になります。次の例では、上記の設定が終了したBashセッションがNX-OSにプロンプトを返します。次に、環境変数がクリアされたBashシェルへの新しいセッションが作成されます。

```
root@Nexus#export http_proxy=http://proxy.example-domain.com:80/
root@Nexus#export https_proxy=https://proxy.example-domain.com:80/
root@Nexus#echo $http_proxy
http://proxy.example-domain.com:80/ root@Nexus#echo $https_proxy
https://proxy.example-domain.com:80/ root@Nexus#exit
Nexus# run bash sudo su -
root@Nexus#echo $http_proxy

root@Nexus#echo $https_proxy

root@Nexus#
```

## HTTP/HTTPSプロキシの永続的な設定

Bashシェルに入る特定のユーザのすべてのセッションに対してHTTP/HTTPSプロキシを永続的に設定するには、ユーザがログインするたびに、「http_proxy」および「https_proxy」環境変数を自動的にエクスポートする必要があります。これは、ユーザのディレクトリにある.bash_profileファイルに`export`コマンドを追加することで実現できます。このファイルは、ユーザがBashシェルにログインすると、Bashによって自動的にロードされます。この例を次に示します。「proxy.example-domain.com」は仮想プロキシサーバのホスト名です。

```
root@Nexus#pwd
/root
root@Nexus#ls -al
total 28
drwxr-xr-x 5 root floppy 200 Dec 6 13:22 . drwxrwxr-t 62 root network-admin 1540 Nov 26 18:10 ..
-rw------- 1 root root 9486 Dec 6 13:22 .bash_history -rw-r--r-- 1 root floppy 703 Dec 6 13:22
.bash_profile drwx------ 3 root root 60 Nov 26 18:10 .config drwxr-xr-x 2 root root 60 Nov 26
18:11 .ncftp -rw------- 1 root root 0 Dec 5 14:37 .python-history -rw------- 1 root floppy 12
Nov 5 05:38 .rhosts drwxr-xr-x 2 root floppy 60 Nov 5 06:17 .ssh -rw------- 1 root root 5499 Dec
6 13:20 .viminfo root@Nexus#echo "export http_proxy=http://proxy.example-domain.com:80/" >>
.bash_profile
root@Nexus#echo "export https_proxy=https://proxy.example-domain.com:80/" >> .bash_profile
root@Nexus#cat .bash_profile | grep proxy
export http_proxy=http://proxy.example-domain.com:80/
export https_proxy=https://proxy.example-domain.com:80/
root@Nexus#exit
Nexus# run bash sudo su - root@Nexus#echo $http_proxy
http://proxy.example-domain.com:80/
root@Nexus#echo $https_proxy
https://proxy.example-domain.com:80/
```

Bashシェルを入力するすべてのユーザに対して、すべてのセッションに特定のHTTP/HTTPSプロキシを設定する場合は、これらの`export/etc/profile`す。Bashは、ユーザがBashシェルにログインすると、このファイルを自動的にロードします。その結果、Bashシェルにログインするすべてのユーザに、それに応じてHTTP/HTTPSプロキシが設定されます。

この例を次に示します。「proxy.example-domain.com」は仮想プロキシサーバのホスト名です。次に、ユーザアカウント「docker-admin」をBashシェルタイプで設定します。これにより、ユーザアカウントがデバイスにリモートアクセスするときに直接Bashシェルにログインできます。SSHを使用して、Docker-adminユーザアカウントを使用して、管理VRFを介してNexusデバイスのmgmt0のIPアドレス(192.0.2.1)にアクセスします。この例では、新しいユーザアカウントがBashシェルにログインしていても、「http_proxy」および「https_proxy」環境変数が設定されていることを示しています。

```
root@Nexus#echo "export http_proxy=http://proxy.example-domain.com:80/" >> /etc/profile
root@Nexus#echo "export https_proxy=https://proxy.example-domain.com:80/" >> /etc/profile
root@Nexus#cat /etc/profile | grep proxy
export http_proxy=http://proxy.example-domain.com:80/ export https_proxy=https://proxy.example-
domain.com:80/ root@Nexus#exit
Nexus# run bash sudo su -
root@Nexus#echo $http_proxy
http://proxy.example-domain.com:80/ root@Nexus#echo $https_proxy
https://proxy.example-domain.com:80/ root@Nexus#exit
Nexus# configure terminal
Nexus(config)# username docker-admin role dev-ops password example_password
Nexus(config)# username docker-admin shelltype bash
Nexus(config)# exit
```

```
Nexus# ssh docker-admin@192.0.2.1 vrf management
Password: -bash-4.3$ whoami
docker-admin
-bash-4.3$ echo $http_proxy
http://proxy.example-domain.com:80/ -bash-4.3$ echo $https_proxy
https://proxy.example-domain.com:80/
```

# Docker Composeのインストール

Docker Composeをインストールするには、wgetユーティリティを使用してDocker Composeの最新のバイナリリリースをダウンロードし、そのバイナリを/usr/binディレクトリに配置する必要があります。

1. Docker Compose GitHubページで利用可能な最新のリリースと共に利用可能な最新の安定したバージョンを確認します。Webページの一番上に表示される最新の安定したリリースのバージョン番号を探します。このドキュメントの執筆時点では、最新の安定版リリースは1.23.2です。

2. DockerのURLを作成する次のURLの{latest-version}を、前のステップで見つかった最新の安定リリースのバージョン番号に置き換えて、バイナリを作成します。

https://github.com/docker/compose/releases/download/{latest-version}/docker-compose-Linux-x86_64

たとえば、このドキュメントの執筆時点の1.23.2のURLは次のとおりです。
https://github.com/docker/compose/releases/download/1.23.2/docker-compose-Linux-x86_64

3.次に示すように、run bash sudo su -コマンドを使用して、NX-OSプロンプトからrootとしてBashシェルを入力します。

```
Nexus# run bash sudo su -
root@Nexus#whoami
root
```

4.必要に応じて、Bashシェルのネットワーク名前空間コンテキストを、DNSおよびインターネット接続の名前空間に変更します。ネットワーク名前空間は、NX-OS VRFと論理的に同じです。次の例は、この特定の環境でDNSおよびインターネット接続を持つ管理ネットワーク名前空間コンテキストに切り替える方法を示しています。

```
root@Nexus#ip netns exec management bash
root@Nexus#ping cisco.com -c 5
PING cisco.com (72.163.4.161) 56(84) bytes of data. 64 bytes from www1.cisco.com (72.163.4.161):
icmp_seq=1 ttl=239 time=29.2 ms 64 bytes from www1.cisco.com (72.163.4.161): icmp_seq=2 ttl=239
time=29.3 ms 64 bytes from www1.cisco.com (72.163.4.161): icmp_seq=3 ttl=239 time=29.3 ms 64
bytes from www1.cisco.com (72.163.4.161): icmp_seq=4 ttl=239 time=29.2 ms 64 bytes from
www1.cisco.com (72.163.4.161): icmp_seq=5 ttl=239 time=29.2 ms --- cisco.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4000ms rtt min/avg/max/mdev =
29.272/29.299/29.347/0.218 ms
```

5.次のコマンドを入力し、{docker-url}したURLに置き換えます。wget {docker-url} -O /usr/bin/docker-compose。このコマンドを実行する例を次に示します。
https://github.com/docker/compose/releases/download/1.23.2/docker-compose-Linux-x86_64を{docker-url}URL。

```
root@Nexus#wget https://github.com/docker/compose/releases/download/1.23.2/docker-compose-Linux-
```

```
x86_64 -O /usr/bin/docker-compose
--2018-12-06 15:24:36-- https://github.com/docker/compose/releases/download/1.23.2/docker-
compose-Linux-x86_64 Resolving proxy.example-domain.com... 2001:DB8::1, 192.0.2.100 Connecting
to proxy.example-domain.com|2001:DB8::1|:80... failed: Cannot assign requested address.
Connecting to proxy.example-domain.com|192.0.2.100|:80... connected. Proxy request sent,
awaiting response... 302 Found Location: https://github-production-release-asset-
2e65be.s3.amazonaws.com/15045751/67742200-f31f-11e8-947e-bd56efcd8886?X-Amz-Algorithm=AWS4-HMAC-
SHA256&X-Amz-Credential=AKIAIWNJYAX4CSVEH53A%2F20181206%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-
Date=20181206T152526Z&X-Amz-Expires=300&X-Amz-
Signature=dfccfd5a32a908040fd8c18694d6d912616f644e7ab3564c6b4ce314a0adbbc7&X-Amz-
SignedHeaders=host&actor_id=0&response-content-disposition=attachment%3B%20filename%3Ddocker-
compose-Linux-x86_64&response-content-type=application%2Foctet-stream [following] --2018-12-06
15:24:36-- https://github-production-release-asset-2e65be.s3.amazonaws.com/15045751/67742200-
f31f-11e8-947e-bd56efcd8886?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-
Credential=AKIAIWNJYAX4CSVEH53A%2F20181206%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-
Date=20181206T152526Z&X-Amz-Expires=300&X-Amz-
Signature=dfccfd5a32a908040fd8c18694d6d912616f644e7ab3564c6b4ce314a0adbbc7&X-Amz-
SignedHeaders=host&actor_id=0&response-content-disposition=attachment%3B%20filename%3Ddocker-
compose-Linux-x86_64&response-content-type=application%2Foctet-stream Connecting to
proxy.example-domain.com|192.0.2.100|:80... connected. Proxy request sent, awaiting response...
200 OK Length: 11748168 (11M) [application/octet-stream] Saving to: ‚Äò/usr/bin/docker-
compose‚Äô /usr/bin/docker-compose
100%[=============================================================================================
==========================================>] 11.20M 6.44MB/s in 1.7s 2018-12-06 15:24:38 (6.44
MB/s) - ‚Äò/usr/bin/docker-compose‚Äô saved [11748168/11748168] root@Nexus#
```

6. /usr/bin/docker-composeバイナリファイルの権限を変更し、`chmod +x /usr/bin/docker-compose`す。次に例を示します。

```
root@Nexus#docker-compose
bash: /usr/bin/docker-compose: Permission denied
root@Nexus#chmod +x /usr/bin/docker-compose
root@Nexus#docker-compose
Define and run multi-container applications with Docker. Usage: docker-compose [-f --help--file
FILE Specify an alternate compose file--project-name NAME Specify an alternate project
namedirectory--verbose Show more output--log-level LEVEL Set log level (DEBUG, INFO, WARNING,
ERROR, CRITICAL)--no-ansi Do not print ANSI control characters--version Print version and exit--
host HOST Daemon socket to connect to--tls Use TLS; implied by --tlsverify--tlscacert CA_PATH
Trust certs signed only by this CA--tlscert CLIENT_CERT_PATH Path to TLS certificate file--
tlskey TLS_KEY_PATH Path to TLS key file--tlsverify Use TLS and verify the remote--skip-
hostname-check Don't check the daemon's hostname against theinthe--project-directory PATH
Specify an alternate working directorytheofthefile--compatibility If set, Compose will attempt
to convert
deploykeysinfilestoorafromthefileandthefilecreateandandtimefromacommandinarunningcontaineronacom
mandkillfromtheforaaonecommandnumberofforastartstoptheandstartversiontheversion
```

# Docker構成機能の確認

小さなdocker-compose.ymlファイルを作成して実行することにより、Docker Composeが正常にインストールされ、機能していることを確認できます。次の例では、このプロセスを順に説明します。

```
root@Nexus#mkdir docker-compose-example
root@Nexus#cd docker-compose-example/
root@Nexus#ls -al
total 0
drwxr-xr-x 2 root root    40 Dec  6 15:31 .
drwxr-xr-x 6 root floppy 260 Dec  6 15:31 ..
root@Nexus#vi docker-compose.yml
root@Nexus#cat docker-compose.yml
```

```
version: "3"
services:
  example_mongo:
    image: mongo:latest
    container_name: "example_mongo"
  example_alpine:
    image: alpine:latest
    container_name: "example_alpine"
root@Nexus#docker-compose up
Creating network "docker-compose-example_default" with the default driver
Pulling example_mongo (mongo:latest)...
latest: Pulling from library/mongo
7b8b6451c85f: Pull complete
ab4d1096d9ba: Pull complete
e6797d1788ac: Pull complete
e25c5c290bde: Pull complete
45aa1a4d5e06: Pull complete
b7e29f184242: Pull complete
ad78e42605af: Pull complete
1f4ac0b92a65: Pull complete
55880275f9fb: Pull complete
bd0396c9dcef: Pull complete
28bf9db38c03: Pull complete
3e954d14ae9b: Pull complete
cd245aa9c426: Pull complete
Creating example_mongo  ... done
Creating example_alpine ... done
Attaching to example_alpine, example_mongo
example_mongo     | 2018-12-06T15:36:18.710+0000 I CONTROL [main] Automatically disabling TLS
1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none' example_mongo | 2018-12-
06T15:36:18.717+0000 I CONTROL [initandlisten] MongoDB starting : pid=1 port=27017
dbpath=/data/db 64-bit host=c4f095f9adb0 example_mongo | 2018-12-06T15:36:18.717+0000 I CONTROL
[initandlisten] db version v4.0.4 example_mongo | 2018-12-06T15:36:18.717+0000 I CONTROL
[initandlisten] git version: f288a3bdf201007f3693c58e140056adf8b04839 example_mongo | 2018-12-
06T15:36:18.717+0000 I CONTROL [initandlisten] OpenSSL version: OpenSSL 1.0.2g 1 Mar 2016
example_mongo | 2018-12-06T15:36:18.717+0000 I CONTROL [initandlisten] allocator: tcmalloc
example_mongo | 2018-12-06T15:36:18.717+0000 I CONTROL [initandlisten] modules: none
example_mongo | 2018-12-06T15:36:18.717+0000 I CONTROL [initandlisten] build environment:
example_mongo | 2018-12-06T15:36:18.717+0000 I CONTROL [initandlisten] distmod: ubuntu1604
example_mongo | 2018-12-06T15:36:18.717+0000 I CONTROL [initandlisten] distarch: x86_64
example_mongo | 2018-12-06T15:36:18.717+0000 I CONTROL [initandlisten] target_arch: x86_64
example_mongo | 2018-12-06T15:36:18.717+0000 I CONTROL [initandlisten] options: { net: {
bindIpAll: true } } example_mongo | 2018-12-06T15:36:18.717+0000 I STORAGE [initandlisten]
example_mongo | 2018-12-06T15:36:18.717+0000 I STORAGE [initandlisten] ** WARNING: Using the XFS
filesystem is strongly recommended with the WiredTiger storage engine example_mongo | 2018-12-
06T15:36:18.717+0000 I STORAGE [initandlisten] ** See http://dochub.mongodb.org/core/prodnotes-
filesystem example_mongo | 2018-12-06T15:36:18.717+0000 I STORAGE [initandlisten]
wiredtiger_open config:
create,cache_size=31621M,session_max=20000,eviction=(threads_min=4,threads_max=4),config_base=fa
lse,statistics=(fast),log=(enabled=true,archive=true,path=journal,compressor=snappy),file_manage
r=(close_idle_time=100000),statistics_log=(wait=0),verbose=(recovery_progress), example_alpine
exited with code 0 example_mongo | 2018-12-06T15:36:19.722+0000 I STORAGE [initandlisten]
WiredTiger message [1544110579:722686][1:0x7f9d5de45a40], txn-recover: Set global recovery
timestamp: 0 example_mongo | 2018-12-06T15:36:19.745+0000 I RECOVERY [initandlisten] WiredTiger
recoveryTimestamp. Ts: Timestamp(0, 0) example_mongo | 2018-12-06T15:36:19.782+0000 I CONTROL
[initandlisten] example_mongo | 2018-12-06T15:36:19.782+0000 I CONTROL [initandlisten] **
WARNING: Access control is not enabled for the database. example_mongo | 2018-12-
06T15:36:19.782+0000 I CONTROL [initandlisten] ** Read and write access to data and
configuration is unrestricted. example_mongo | 2018-12-06T15:36:19.782+0000 I CONTROL
[initandlisten] example_mongo | 2018-12-06T15:36:19.783+0000 I STORAGE [initandlisten]
createCollection: admin.system.version with provided UUID: dc0b3249-576e-4546-9d97-de841f5c45c4
example_mongo | 2018-12-06T15:36:19.810+0000 I COMMAND [initandlisten] setting
featureCompatibilityVersion to 4.0 example_mongo | 2018-12-06T15:36:19.814+0000 I STORAGE
[initandlisten] createCollection: local.startup_log with generated UUID: 2f9820f5-11ad-480d-
```

```
a46c-c58222beb0ad example_mongo | 2018-12-06T15:36:19.841+0000 I FTDC [initandlisten]
Initializing full-time diagnostic data capture with directory '/data/db/diagnostic.data'
example_mongo | 2018-12-06T15:36:19.842+0000 I NETWORK [initandlisten] waiting for connections
on port 27017 example_mongo | 2018-12-06T15:36:19.842+0000 I STORAGE
[LogicalSessionCacheRefresh] createCollection: config.system.sessions with generated UUID:
d4aeac07-29fd-4208-9f83-394b4af648a2 example_mongo | 2018-12-06T15:36:19.885+0000 I INDEX
[LogicalSessionCacheRefresh] build index on: config.system.sessions properties: { v: 2, key: {
lastUse: 1 }, name: "lsidTTLIndex", ns: "config.system.sessions", expireAfterSeconds: 1800 }
example_mongo | 2018-12-06T15:36:19.885+0000 I INDEX [LogicalSessionCacheRefresh] building index
using bulk method; build may temporarily use up to 500 megabytes of RAM example_mongo | 2018-12-
06T15:36:19.886+0000 I INDEX [LogicalSessionCacheRefresh] build index done. scanned 0 total
records. 0 secs ^C
Gracefully stopping... (press Ctrl+C again to force)
Stopping example_mongo  ... done
root@Nexus#
```

**注意**：`docker-compose`コマンドを実行するDNSそうしないと、Docker Composeは要求されたイメージをDocker Hubから取得できません。

**注**：Docker Composeが開始したマルチコンテナDockerアプリケーションをDocker Composeセッションに接続して停止するには、「Ctrl+C」キーの組み合わせを押します。

## 関連情報

- Docker Composeインストールドキュメント
- Docker Compose Documentationの概要
- Cisco Nexus 9000シリーズNX-OSプログラマビリティガイド、リリース9.x
- Cisco Nexus 9000シリーズNX-OSプログラマビリティガイド、リリース7.x
- Cisco Nexus 9000シリーズNX-OSプログラマビリティガイド、リリース6.x
- Cisco Nexus 3000シリーズNX-OSプログラマビリティガイド、リリース9.x
- Cisco Nexus 3000シリーズNX-OSプログラマビリティガイド、リリース7.x
- Cisco Nexus 3000シリーズNX-OSプログラマビリティガイド、リリース6.x
- Cisco Nexus 3500シリーズNX-OSプログラマビリティガイド、リリース9.x
- Cisco Nexus 3500シリーズNX-OSプログラマビリティガイド、リリース7.x
- Cisco Nexus 3500シリーズNX-OSプログラマビリティガイド、リリース6.x
- Cisco Nexus 3600シリーズNX-OSプログラマビリティガイド、リリース9.x
- Cisco Nexus 3600シリーズNX-OSプログラマビリティガイド、リリース7.x
- Cisco Open NX-OSによるプログラマビリティと自動化