在NX-OS Bash Shell中安裝Docker合成

目錄

簡介

本文檔介紹用於在NX-OS Bash外殼中安裝Docker合成軟體包的步驟。

從NX-OS版本9.2(1)開始，Cisco Nexus 3000和9000系列裝置支援Bash shell中的Docker功能。 如Docker Compose文檔所述，「Compose是用於定義和運行多容器Docker應用程式的工具。」Docker Compose允許應用程式開發人員在一個名為「docker-compose.yml」的YAML檔案中定義構成應用程式的所有服務。 然後，只需一個命令，即可建立、構建和啟動所有這些服務。此外，還可以從Docker Compose命令套件中停止並監控所有服務。

雖然NX-OS Bash外殼本身支援Docker功能，但必須單獨安裝Docker Compose。

必要條件

需求

本文檔要求在Cisco Nexus裝置上啟用Bash外殼。有關啟用Bash shell的說明，請參閱Cisco Nexus 9000系列NX-OS可程式設計性指南中Bash一章的「訪問Bash」一節。

本文檔要求將Bash shell配置為能夠將域名解析為IP地址的DNS客戶端。有關在Bash外殼中配置DNS伺服器的說明，請參閱文檔。

採用元件

本文中的資訊係根據以下軟體和硬體版本：

- 從NX-OS版本9.2(1)開始的Nexus 9000平台
- 從NX-OS版本9.2(1)開始的Nexus 3000平台

本文中的資訊是根據特定實驗室環境內的裝置所建立。文中使用到的所有裝置皆從已清除（預設）的組態來啟動。如果您的網路正在作用，請確保您已瞭解任何指令可能造成的影響。

配置HTTP/HTTPS代理

如果您的環境要求使用HTTP或HTTPS代理，則需要在安裝Docker Compose之前將Bash外殼配置為使用這些代理。

以root使用者身份通過`run bash sudo su -`命令登入Bash shell。

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

臨時配置HTTP/HTTPS代理

要為此會話臨時配置HTTP/HTTPS代理，請使用`export`命令定義「http_proxy」和「https_proxy」環境變數。下面顯示了一個示例，其中「proxy.example-domain.com」是假想代理伺服器的主機名。

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

使用`echo $http_proxy`和`echo $https_proxy`命令，確認已根據需要配置環境變數，如下所示：

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

分配給這些環境變數的值將在會話終止時清除，並且每次輸入Bash shell時都需要重新配置。在下面的示例中，退出上述配置的Bash會話，將提示返回到NX-OS。然後，將建立一個指向Bash shell的新會話，該會話中已經清除環境變數。

```
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#
```

永久配置HTTP/HTTPS代理

要為進入Bash外殼的特定使用者的所有會話永久配置HTTP/HTTPS代理，必須在任何使用者每次登入時自動匯出「http_proxy」和「https_proxy」環境變數。這可以通過向使用者目錄中的.bash_profile檔案附加`export`命令來實現，當使用者登入到Bash shell時，Bash會自動載入該檔案。下面顯示了一個示例，其中「proxy.example-domain.com」是假想代理伺服器的主機名。

```
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/
```

如果要為進入Bash外殼的所有使用者配置所有會話的特定HTTP/HTTPS代理，請將這些`export`命令附加到/etc/profile檔案。Bash會在任何使用者登入到Bash shell時首先自動載入此檔案 — 因此，所有登入到Bash shell的使用者都會相應地配置其HTTP/HTTPS代理。

下面顯示了一個示例，其中「proxy.example-domain.com」是假想代理伺服器的主機名。然後使用者帳戶「docker-admin」配置為Bash shelltype，這允許該使用者帳戶在遠端訪問裝置時直接登入Bash shell。然後使用SSH通過docker-admin使用者帳戶管理VRF訪問Nexus裝置的mgmt0 IP地址(192.0.2.1)。此示例顯示「http_proxy」和「https_proxy」環境變數已設定，即使全新使用者帳戶已登入到Bash shell也是如此。

```
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/
```

安裝Docker合成

要安裝Docker Compose，必須使用`wget`實用程式下載最新二進製版本的Docker Compose，然後將該二進位制檔案放在/usr/bin目錄中。

1.確定可用於「Docker合成GitHub」頁上最新可用版本的最新Docker合成穩定版本。在網頁頂端找到最新穩定版本的版本號。在撰寫本文時，最新的穩定版本是1.23.2。

2.使用上一步中發現的最新穩定版本的版本號替換以下URL中的`{latest-version}`，為Docker合成二進位制檔案定製URL:

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

例如，寫本文時1.23.2的URL如下：https://github.com/docker/compose/releases/download/1.23.2/docker-compose-Linux-x86_64

3.使用`run bash sudo su -`命令從NX-OS提示符輸入Bash shell作為root，如下所示：

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

4.如有必要，請將Bash shell的網路名稱空間上下文更改為具有DNS和Internet連線的名稱空間。網路名稱空間在邏輯上與NX-OS VRF相同。以下示例演示如何切換到管理網路名稱空間上下文，該上下文在此特定環境中具有DNS和Internet連線。

```
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.輸入以下命令，將`{docker-url}`替換為在上一步中建立的URL:`wget {docker-url} -O /usr/bin/docker-compose`。下面顯示了執行此命令的示例，將https://github.com/docker/compose/releases/download/1.23.2/docker-compose-Linux-x86_64用作`{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.修改/usr/bin/docker-compose二進位制檔案的許可權，以便使用`chmod +x /usr/bin/docker-compose`命令可以執行該檔案。如下所示：

```
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
```

驗證Docker撰寫功能

通過建立並運行小型docker-compose.yml檔案，可以驗證Docker Compose已成功安裝並且工作正常。以下示例將逐步完成此過程。

```
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：確保`docker-compose`命令在具有DNS和Internet連線的網路名稱空間的上下文中執行。否則，Docker合成將無法從Docker中心提取請求的映像。

附註：要停止連線到Docker Compose會話時由Docker Compose啟動的多容器Docker應用程式，請按"Ctrl+C"組合鍵。

相關資訊

- Docker撰寫安裝文檔
- Docker撰寫文檔概述
- Cisco Nexus 9000系列NX-OS可程式設計性指南9.x版
- Cisco Nexus 9000系列NX-OS可程式設計性指南7.x版
- Cisco Nexus 9000系列NX-OS可程式設計性指南6.x版
- Cisco Nexus 3000系列NX-OS可程式設計性指南9.x版
- Cisco Nexus 3000系列NX-OS可程式設計性指南7.x版
- Cisco Nexus 3000系列NX-OS可程式設計性指南6.x版
- Cisco Nexus 3500系列NX-OS可程式設計性指南9.x版
- Cisco Nexus 3500系列NX-OS可程式設計性指南7.x版
- Cisco Nexus 3500系列NX-OS可程式設計性指南6.x版
- Cisco Nexus 3600系列NX-OS可程式設計性指南9.x版
- Cisco Nexus 3600系列NX-OS可程式設計性指南7.x版
- Cisco Open NX-OS的可程式設計性和自動化