Installer Docker Composer dans l'interpréteur de commandes Bash de NX-OS

Contenu

Introduction

Ce document décrit les étapes utilisées pour installer le package Docker Compose dans le shell Bash de NX-OS.

Les périphériques des gammes Cisco Nexus 3000 et 9000 prennent en charge la fonctionnalité Docker dans le shell Bash à partir de la version 9.2(1) de NX-OS. Comme l'indique la documentation de la composition du Docker, « Compose est un outil permettant de définir et d'exécuter des applications de Docker à conteneurs multiples. » Docker Compose permet aux développeurs d'applications de définir tous les services qui constituent une application dans un seul fichier YAML nommé « docker-compose.yml ». Ensuite, avec une seule commande, tous ces services peuvent être créés, créés et démarrés. En outre, tous les services peuvent être arrêtés et surveillés à partir de la suite de commandes Docker Compose.

Bien que la fonctionnalité Docker soit prise en charge nativement dans le shell Bash de NX-OS, Docker Compose doit être installé séparément.

Conditions préalables

Conditions requises

Ce document nécessite que le shell Bash soit activé sur votre périphérique Cisco Nexus. Reportez-vous à la section « Accéder à Bash » du chapitre Bash du Guide de programmabilité NX-OS de la gamme Cisco Nexus 9000 pour obtenir des instructions sur l'activation du shell Bash.

Ce document nécessite que le shell Bash soit configuré en tant que client DNS capable de résoudre les noms d'hôte de domaine en adresses IP. Reportez-vous au document pour obtenir des instructions sur la configuration des serveurs DNS dans l'interpréteur de commandes Bash.

Components Used

Les informations contenues dans ce document sont basées sur les versions de matériel et de logiciel suivantes :

- Plate-forme Nexus 9000 à partir de la version 9.2(1) de NX-OS
- Plate-forme Nexus 3000 à partir de la version 9.2(1) de NX-OS

Les informations contenues dans ce document ont été créées à partir des périphériques dans un environnement de laboratoire spécifique. All of the devices used in this document started with a cleared (default) configuration. If your network is live, make sure that you understand the potential impact of any command.

Configuration des proxy HTTP/HTTPS

Si votre environnement nécessite l'utilisation d'un proxy HTTP ou HTTPS, le shell Bash doit être configuré pour utiliser ces proxys avant que Docker Compose puisse être installé.

Connectez-vous au shell Bash en tant qu'utilisateur racine via la commande `run bash sudo su -`.

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

Configuration temporaire des proxy HTTP/HTTPS

Pour configurer temporairement les proxy HTTP/HTTPS pour cette session, utilisez la commande `export` pour définir les variables d'environnement « http_proxy » et « https_proxy ». Un exemple de ceci est montré ci-dessous, où « proxy.example-domain.com » est le nom d'hôte d'un serveur proxy hypothétique.

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

Confirmez que les variables d'environnement sont configurées comme souhaité à l'aide des commandes `echo $http_proxy` et `echo $https_proxy`, comme illustré ci-dessous :

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

Les valeurs attribuées à ces variables d'environnement seront effacées lorsque la session sera terminée et devront être reconfigurées chaque fois que le shell Bash est entré. Dans l'exemple ci-dessous, la session Bash où la configuration ci-dessus est terminée, renvoyant l'invite à NX-OS. Ensuite, une nouvelle session du shell Bash est créée, où les variables d'environnement ont été effacées.

```
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#
```

Configuration permanente des proxy HTTP/HTTPS

Pour configurer de manière permanente les proxys HTTP/HTTPS pour toutes les sessions d'un utilisateur spécifique qui entre dans l'interpréteur de commandes Bash, les variables d'environnement « http_proxy » et « https_proxy » doivent être exportées automatiquement chaque fois qu'un utilisateur se connecte. Pour ce faire, vous pouvez ajouter des commandes `d'exportation` au fichier .bash_profile situé dans le répertoire de l'utilisateur, que Bash charge automatiquement lorsque l'utilisateur se connecte au shell Bash. Un exemple de ceci est montré ci-dessous, où « proxy.example-domain.com » est le nom d'hôte d'un serveur proxy hypothétique.

```
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/
```

Si vous souhaitez configurer des proxys HTTP/HTTPS spécifiques pour toutes les sessions pour tous les utilisateurs qui entrent dans l'interpréteur de commandes Bash, ajoutez ces commandes `d'exportation` au fichier /etc/profile. Bash charge automatiquement ce fichier en premier lorsqu'un utilisateur se connecte au shell Bash. Par conséquent, tous les utilisateurs qui se connectent au shell Bash auront leurs proxies HTTP/HTTPS configurés en conséquence.

Un exemple de ceci est montré ci-dessous, où « proxy.example-domain.com » est le nom d'hôte d'un serveur proxy hypothétique. Le compte d'utilisateur « docker-admin » est ensuite configuré avec le type de code Bash, qui permet au compte d'utilisateur de se connecter directement au shell Bash lors de l'accès à distance au périphérique. SSH est ensuite utilisé pour accéder à l'adresse IP de mgmt0 (192.0.2.1) du périphérique Nexus via le VRF de gestion à l'aide du compte d'utilisateur docker-admin. L'exemple montre que les variables d'environnement « http_proxy » et « https_proxy » ont été définies, même lorsqu'un nouveau compte d'utilisateur a été connecté au shell Bash.

```
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/
```

Installation de Docker Compose

Pour installer Docker Compose, vous devez utiliser l'utilitaire `wget` pour télécharger la dernière version binaire de Docker Compose, puis placer ce binaire dans le répertoire /usr/bin.

1. Déterminez la dernière version stable de Docker Compose disponible avec les dernières versions disponibles sur la page Docker Compose GitHub. Recherchez le numéro de version de la dernière version stable en haut de la page Web. Au moment de la rédaction de cet article, la dernière version stable est 1.23.2.

2. Créez l'URL du fichier binaire Docker Compose en remplaçant `{dernière-version}` dans l'URL ci-dessous par le numéro de version de la dernière version stable trouvée à l'étape précédente :

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

Par exemple, l'URL de 1.23.2 au moment de la rédaction du présent document est la suivante : https://github.com/docker/compose/releases/download/1.23.2/docker-compose-Linux-x86_64

3. Entrez le shell Bash en tant que racine à partir de l'invite NX-OS avec la commande `run bash sudo su -`, comme indiqué ci-dessous :

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

4. Si nécessaire, modifiez le contexte de l'espace de noms réseau de l'interpréteur de commandes Bash en espace de noms avec une connectivité DNS et Internet. Les espaces de noms réseau sont logiquement identiques aux VRF NX-OS. L'exemple ci-dessous montre comment basculer vers le contexte de l'espace de noms du réseau de gestion, qui dispose d'une connectivité DNS et Internet dans cet environnement spécifique.

```
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. Entrez la commande suivante, en remplaçant `{docker-url}` par l'URL créée à l'étape précédente : `wget {docker-url} -O /usr/bin/docker-compose`. Un exemple d'exécution de cette commande est présenté ci-dessous, en utilisant https://github.com/docker/compose/releases/download/1.23.2/docker-compose-Linux-x86_64 comme URL de remplacement pour `{docker-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. Modifiez les autorisations du fichier binaire /usr/bin/docker-compose de sorte qu'il soit exécutable à l'aide de la commande `chmod +x /usr/bin/docker-compose`. Ceci est démontré ci-dessous :

```
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 deploykeysinfilestoorafromthefileandthefilecreateandandtimefromacommandinarunningcontaineronacommandkillfromtheforaaonecommandnumberofforastartstoptheandstartversiontheversion
```

Vérification de la fonctionnalité de composition du Docker

Vous pouvez vérifier que Docker Compose est installé et fonctionne correctement en créant et en exécutant un petit fichier docker-compose.yml. L'exemple ci-dessous présente les étapes de ce

processus.

```
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#
```

Attention : Assurez-vous que lorsque la commande `docker-compose` est exécutée, elle le fait dans le contexte d'un espace de noms de réseau disposant d'une connectivité DNS et Internet. Sinon, Docker Compose ne pourra pas extraire les images demandées du Docker Hub.

Note: Pour arrêter une application Docker multi-conteneurs démarrée par Docker Compose lors de sa connexion à la session Docker Compose, appuyez sur la combinaison de touches Ctrl+C.

Informations connexes

- Documentation d'installation de la composition du Docker
- Vue d'ensemble de la documentation du composant Docker
- Guide de programmabilité NX-OS de la gamme Cisco Nexus 9000, version 9.x
- Guide de programmabilité NX-OS de la gamme Cisco Nexus 9000, version 7.x
- Guide de programmabilité NX-OS de la gamme Cisco Nexus 9000, version 6.x
- Guide de programmabilité NX-OS de la gamme Cisco Nexus 3000, version 9.x
- Guide de programmabilité NX-OS de la gamme Cisco Nexus 3000, version 7.x
- Guide de programmabilité NX-OS de la gamme Cisco Nexus 3000, version 6.x
- Guide de programmabilité NX-OS de la gamme Cisco Nexus 3500, version 9.x
- Guide de programmabilité NX-OS de la gamme Cisco Nexus 3500, version 7.x
- Guide de programmabilité NX-OS de la gamme Cisco Nexus 3500, version 6.x
- Guide de programmabilité NX-OS de la gamme Cisco Nexus 3600, version 9.x
- Guide de programmabilité NX-OS de la gamme Cisco Nexus 3600, version 7.x
- Programmabilité et automatisation avec Cisco Open NX-OS