Installa Docker Compose in NX-OS Bash Shell

Sommario

Introduzione

In questo documento viene descritto come installare il pacchetto Docker Compose all'interno della shell di NX-OS Bash.

I dispositivi Cisco Nexus serie 3000 e 9000 supportano la funzionalità Docker nella shell Bash a partire da NX-OS versione 9.2(1). Come descritto nella documentazione di Docker Compose, "Compose è uno strumento per la definizione e l'esecuzione di applicazioni Docker a più contenitori." Docker Compose consente agli sviluppatori di applicazioni di definire tutti i servizi che costituiscono un'applicazione all'interno di un singolo file YAML denominato "docker-compose.yml". Con un unico comando è quindi possibile creare, creare e avviare tutti questi servizi. Inoltre, tutti i servizi possono essere arrestati e monitorati dall'interno della suite di comandi Docker Compose.

Sebbene la funzionalità Docker sia supportata in modo nativo nella shell di NX-OS Bash, Docker Compose deve essere installato separatamente.

Prerequisiti

Requisiti

Questo documento richiede che la shell Bash sia abilitata sul dispositivo Cisco Nexus. Fare riferimento alla sezione "Accesso a Bash" del capitolo Bash nella Cisco Nexus 9000 NX-OS Programmability Guide per istruzioni su come abilitare la shell Bash.

Questo documento richiede che la shell Bash sia configurata come client DNS in grado di risolvere i nomi host dei domini in indirizzi IP. Per istruzioni sulla configurazione dei server DNS nella shell Bash, consultare il documento.

Componenti usati

Le informazioni fornite in questo documento si basano sulle seguenti versioni software e

hardware:

- Piattaforma Nexus 9000 a partire da NX-OS release 9.2(1)
- Piattaforma Nexus 3000 a partire da NX-OS versione 9.2(1)

Le informazioni discusse in questo documento fanno riferimento a dispositivi usati in uno specifico ambiente di emulazione. Su tutti i dispositivi menzionati nel documento la configurazione è stata ripristinata ai valori predefiniti. Se la rete è operativa, valutare attentamente eventuali conseguenze derivanti dall'uso dei comandi.

Configurazione dei proxy HTTP/HTTPS

Se l'ambiente richiede l'utilizzo di un proxy HTTP o HTTPS, è necessario configurare la shell Bash per l'utilizzo di tali proxy prima di poter installare Docker Compose.

Accedere alla shell Bash come utente root tramite il comando `run bash sudo su`.

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

Configurazione temporanea dei proxy HTTP/HTTPS

Per configurare temporaneamente i proxy HTTP/HTTPS per questa sessione, utilizzare il comando `export` per definire le variabili di ambiente "http_proxy" e "https_proxy". Di seguito è riportato un esempio di questo problema, dove "proxy.example-domain.com" è il nome host di un ipotetico server proxy.

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

Confermare che le variabili di ambiente siano configurate nel modo desiderato utilizzando i comandi `echo $http_proxy` ed `echo $https_proxy`, come mostrato di seguito:

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

I valori assegnati a queste variabili di ambiente verranno cancellati al termine della sessione e dovranno essere riconfigurati ogni volta che si immette la shell Bash. Nell'esempio riportato di seguito, la sessione Bash in cui è stata chiusa la configurazione precedente, restituendo il prompt a NX-OS. Viene quindi creata una nuova sessione per la shell Bash, in cui le variabili di ambiente sono state cancellate.

```
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#
```

Configurazione permanente dei proxy HTTP/HTTPS

Per configurare in modo permanente i proxy HTTP/HTTPS per tutte le sessioni di un utente specifico che accede alla shell Bash, è necessario esportare automaticamente le variabili di ambiente "http_proxy" e "https_proxy" ogni volta che un utente esegue l'accesso. A tale scopo, è possibile aggiungere i comandi di `esportazione` al file .bash_profile che si trova nella directory dell'utente e che viene caricato automaticamente da Bash quando l'utente accede alla shell Bash. Di seguito è riportato un esempio di questo problema, dove "proxy.example-domain.com" è il nome host di un ipotetico server proxy.

```
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/
```

Se si desidera configurare proxy HTTP/HTTPS specifici per tutte le sessioni per tutti gli utenti che immettono la shell Bash, aggiungere questi comandi di `esportazione` al file /etc/profile. Bash carica automaticamente questo file per primo quando un utente accede alla shell Bash. Di conseguenza, tutti gli utenti che accedono alla shell Bash avranno i proxy HTTP/HTTPS configurati di conseguenza.

Di seguito è riportato un esempio di questo problema, dove "proxy.example-domain.com" è il nome host di un ipotetico server proxy. L'account utente "docker-admin" viene quindi configurato con il tipo di shell Bash, che consente all'account utente di accedere direttamente alla shell Bash quando si accede in remoto al dispositivo. SSH viene quindi utilizzato per accedere all'indirizzo IP mgmt0 (192.0.2.1) del dispositivo Nexus tramite il VRF di gestione utilizzando l'account utente docker-admin. Nell'esempio viene mostrato che le variabili di ambiente "http_proxy" e "https_proxy" sono state impostate, anche quando un nuovo account utente è stato registrato nella 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/
```

Installazione di Docker Compose

Per installare Docker Compose, è necessario utilizzare l'utilità `wget` per scaricare l'ultima versione binaria di Docker Compose, quindi posizionare il file binario nella directory /usr/bin.

1. Determinare l'ultima versione stabile di Docker Compose disponibile con le ultime versioni disponibili nella pagina Docker Compose GitHub. Individuare il numero di versione dell'ultima versione stabile nella parte superiore della pagina Web. Al momento della stesura di questo documento, l'ultima versione stabile è la 1.23.2.

2. Creare l'URL per il file binario di composizione Docker sostituendo `{last-version}` nell'URL seguente con il numero di versione dell'ultima versione stabile trovata nel passaggio precedente:

https://github.com/docker/compose/releases/download/{ultima versione}/docker-compose-Linux-x86_64

Ad esempio, l'URL per 1.23.2 al momento di questa scrittura è il seguente: https://github.com/docker/compose/releases/download/1.23.2/docker-compose-Linux-x86_64

3. Immettere la shell Bash come root dal prompt di NX-OS con il comando `run bash sudo -`, come illustrato di seguito:

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

4. Se necessario, modificare il contesto dello spazio dei nomi di rete della shell Bash in uno spazio dei nomi con connettività DNS e Internet. Gli spazi dei nomi di rete sono logicamente identici ai VRF di NX-OS. Nell'esempio seguente viene illustrato come passare al contesto dello spazio dei nomi della rete di gestione, che dispone di connettività DNS e Internet in questo ambiente specifico.

```
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. Immettere il comando seguente, sostituendo `{docker-url}` con l'URL creato nel passaggio precedente: `wget {docker-url} -O /usr/bin/docker-compose`. Di seguito è riportato un esempio di esecuzione del comando, utilizzando https://github.com/docker/compose/releases/download/1.23.2/docker-compose-Linux-x86_64 come URL sostitutivo per `{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. Modificare le autorizzazioni del file binario /usr/bin/docker-compose in modo che sia eseguibile utilizzando il comando `chmod +x /usr/bin/docker-compose`. Questa condizione viene dimostrata di seguito:

```
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
```

Verifica della funzionalità di composizione Docker

Èpossibile verificare che Docker Compose sia installato e funzioni correttamente creando ed eseguendo un file docker-compose.yml di piccole dimensioni. Nell'esempio seguente viene illustrato il processo.

```
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#
```

Attenzione: Verificare che il comando `docker-compose` venga eseguito nel contesto di uno spazio dei nomi di rete con connettività DNS e Internet. In caso contrario, Docker Compose non sarà in grado di estrarre le immagini richieste dall'hub Docker.

Nota: Per arrestare un'applicazione Docker a più contenitori avviata da Docker Compose mentre era collegata alla sessione di Docker Compose, premere la combinazione di tasti "Ctrl+C".

Informazioni correlate

- Documentazione sull'installazione di Docker Compose
- Panoramica della documentazione di Docker Compose
- Cisco Nexus serie 9000 NX-OS Programmability Guide, versione 9.x
- Cisco Nexus serie 9000 NX-OS Programmability Guide, versione 7.x
- Cisco Nexus serie 9000 NX-OS Programmability Guide, versione 6.x
- Cisco Nexus serie 3000 NX-OS Programmability Guide, versione 9.x
- Cisco Nexus serie 3000 NX-OS Programmability Guide, versione 7.x
- Cisco Nexus serie 3000 NX-OS Programmability Guide, versione 6.x
- Cisco Nexus serie 3500 NX-OS Programmability Guide, versione 9.x
- Cisco Nexus serie 3500 NX-OS Programmability Guide, versione 7.x
- Cisco Nexus serie 3500 NX-OS Programmability Guide, versione 6.x
- Cisco Nexus serie 3600 NX-OS Programmability Guide, versione 9.x
- Cisco Nexus serie 3600 NX-OS Programmability Guide, versione 7.x
- Programmabilità e automazione con Cisco Open NX-OS