Instalar o Docker Compose no NX-OS Bash Shell

Contents

[Introduction](#)
[Prerequisites](#)
[Requirements](#)
[Componentes Utilizados](#)
[Configurando proxies HTTP/HTTPS](#)
[Configuração temporária de proxy HTTP/HTTPS](#)
[Configuração permanente de proxies HTTP/HTTPS](#)
[Instalando o Docker Compose](#)
[Verificando a funcionalidade de composição do Docker](#)
[Informações Relacionadas](#)

Introduction

Este documento descreve as etapas usadas para instalar o pacote Docker Compose no shell do NX-OS Bash.

Os dispositivos Cisco Nexus 3000 e 9000 Series suportam a funcionalidade Docker no shell Bash começando com NX-OS Versão 9.2(1). Conforme descrito pela documentação do Docker Compose, "Compose é uma ferramenta para definir e executar aplicativos Docker de vários contêineres." O Docker Compose permite que os desenvolvedores de aplicativos definam todos os serviços que constituem um aplicativo em um único arquivo YAML chamado "docker-compose.yml". Em seguida, com um único comando, todos esses serviços podem ser criados, criados e iniciados. Além disso, todos os serviços podem ser interrompidos e monitorados no conjunto de comandos do Docker Compose.

Embora a funcionalidade do Docker seja suportada nativamente no shell do NX-OS Bash, o Docker Compose deve ser instalado separadamente.

Prerequisites

Requirements

Este documento requer que o shell Bash esteja ativado em seu dispositivo Cisco Nexus. Consulte a seção "Access Bash" do capítulo Bash no Cisco Nexus 9000 Series NX-OS Programmability Guide para obter instruções sobre como habilitar o shell Bash.

Este documento exige que o shell Bash seja configurado como um cliente DNS capaz de resolver nomes de host de domínio para endereços IP. Consulte o documento para obter instruções sobre como configurar servidores DNS no shell Bash.

Componentes Utilizados

As informações neste documento são baseadas nestas versões de software e hardware:

- Plataforma do Nexus 9000 a partir do NX-OS versão 9.2(1)
- Plataforma Nexus 3000 a partir do NX-OS versão 9.2(1)

As informações apresentadas neste documento foram criadas a partir de dispositivos em um ambiente de laboratório específico. 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.

Configurando proxies HTTP/HTTPS

Se o ambiente exigir o uso de um proxy HTTP ou HTTPS, o shell Bash precisará ser configurado para usar esses proxies antes que o Docker Compose possa ser instalado.

Efetue login no shell Bash como o usuário raiz através do comando `run bash sudo su -`.

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

Configuração temporária de proxy HTTP/HTTPS

Para configurar temporariamente os proxies HTTP/HTTPS para esta sessão, use o comando `export` para definir as variáveis de ambiente "http_proxy" e "https_proxy". Um exemplo disso é mostrado abaixo, onde "proxy.example-domain.com" é o nome de host de um servidor proxy hipotético.

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

Confirme se as variáveis de ambiente estão configuradas conforme desejado usando os comandos `echo $http_proxy` e `echo $https_proxy`, como demonstrado abaixo:

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

Os valores atribuídos a essas variáveis de ambiente serão apagados quando a sessão for encerrada e precisarão ser reconfigurados toda vez que o shell Bash for inserido. No exemplo abaixo, a sessão Bash onde a configuração acima foi encerrada, retornando o prompt ao NX-OS. Em seguida, uma nova sessão para o shell Bash é criada, onde as variáveis de ambiente foram limpas.

```
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#
```

Configuração permanente de proxies HTTP/HTTPS

Para configurar permanentemente proxies HTTP/HTTPS para todas as sessões de um usuário específico que entra no shell Bash, as variáveis de ambiente "http_proxy" e "https_proxy" devem ser exportadas automaticamente sempre que qualquer usuário fizer login. Isso pode ser feito anexando comandos `export` ao arquivo .bash_profile localizado no diretório do usuário, que Bash carrega automaticamente quando esse usuário faz login no shell Bash. Um exemplo disso é mostrado abaixo, onde "proxy.example-domain.com" é o nome de host de um servidor proxy hipotético.

```
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 você quiser configurar proxies HTTP/HTTPS específicos para todas as sessões para todos os usuários que inserem o shell Bash, anexe esses comandos `de exportação` ao arquivo /etc/profile. Bash carrega automaticamente esse arquivo primeiro quando qualquer usuário faz login no shell Bash - como resultado, todos os usuários que fazem login no shell Bash terão seus proxies HTTP/HTTPS configurados de acordo.

Um exemplo disso é mostrado abaixo, onde "proxy.example-domain.com" é o nome de host de um servidor proxy hipotético. A conta de usuário "docker-admin" é então configurada com o tipo de shell Bash, que permite que a conta de usuário faça login diretamente no shell Bash ao acessar remotamente o dispositivo. O SSH é então usado para acessar o endereço IP mgmt0 (192.0.2.1) do dispositivo Nexus através do VRF de gerenciamento usando a conta de usuário docker-admin. O exemplo mostra que as variáveis de ambiente "http_proxy" e "https_proxy" foram definidas, mesmo quando uma nova conta de usuário foi registrada no 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/
```

Instalando o Docker Compose

Para instalar o Docker Compose, é necessário usar o utilitário `wget` para baixar a versão binária mais recente do Docker Compose e, em seguida, colocá-lo no diretório /usr/bin.

1. Determine a versão estável mais recente do Docker Compose disponível com as versões mais recentes disponíveis na página do Docker Compose GitHub. Localize o número da versão da versão estável mais recente na parte superior da página da Web. No momento dessa gravação, a última versão estável é 1.23.2.

2. Crie o URL para o binário Docker Compose substituindo `{última versão}` no URL abaixo pelo número da versão estável mais recente encontrado na etapa anterior:

https://github.com/docker/compose/releases/download/{última versão}/docker-compose-Linux-x86_64

Por exemplo, a URL para 1.23.2 no momento da gravação é a seguinte: https://github.com/docker/compose/releases/download/1.23.2/docker-compose-Linux-x86_64

3. Insira o shell Bash como raiz no prompt NX-OS com o comando `run bash sudo su -`, como demonstrado abaixo:

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

4. Se necessário, altere o contexto do namespace de rede do shell Bash para um namespace com DNS e conectividade com a Internet. Os namespaces de rede são logicamente idênticos aos VRFs do NX-OS. O exemplo abaixo demonstra como mudar para o contexto de namespace da rede de gerenciamento, que tem DNS e conectividade com a Internet neste ambiente específico.

```
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. Digite o seguinte comando, substituindo `{docker-url}` pelo URL criado na etapa anterior: `wget {docker-url} -O /usr/bin/docker-compose`. Um exemplo desse comando sendo executado é

mostrado abaixo, usando https://github.com/docker/compose/releases/download/1.23.2/docker-compose-Linux-x86_64 como o URL de substituição para `{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. Modifique as permissões do arquivo binário /usr/bin/docker-compose de modo que ele seja executável usando o comando `chmod +x /usr/bin/docker-compose`. Isso é demonstrado abaixo:

```
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
```

Verificando a funcionalidade de composição do Docker

Épossível verificar se o Docker Compose está instalado com êxito e funcionando criando e executando um pequeno arquivo docker-compose.yml. O exemplo abaixo passa por esse 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#
```

Caution: Certifique-se de que quando o comando `docker-compose` é executado, isso é feito no contexto de um namespace de rede que tenha DNS e conectividade com a Internet. Caso contrário, o Docker Compose não poderá obter as imagens solicitadas do Docker Hub.

Note: Para colocar um aplicativo Docker de vários contêineres iniciado pelo Docker Compose enquanto conectado à sessão Docker Compose, pressione a combinação de teclas "Ctrl+C".

Informações Relacionadas

- Documentação de instalação do Docker Compose
- Visão geral da documentação do Docker Compose
- Guia de programabilidade do Cisco Nexus 9000 Series NX-OS, versão 9.x
- Guia de programabilidade do Cisco Nexus 9000 Series NX-OS, versão 7.x
- Guia de programabilidade do Cisco Nexus 9000 Series NX-OS, versão 6.x
- Guia de programabilidade do Cisco Nexus 3000 Series NX-OS, versão 9.x
- Guia de programabilidade do Cisco Nexus 3000 Series NX-OS, versão 7.x
- Guia de programabilidade do Cisco Nexus 3000 Series NX-OS, versão 6.x
- Guia de programabilidade do Cisco Nexus 3500 Series NX-OS, versão 9.x
- Guia de programabilidade do Cisco Nexus 3500 Series NX-OS, versão 7.x
- Guia de programabilidade do Cisco Nexus 3500 Series NX-OS, versão 6.x
- Guia de programabilidade do Cisco Nexus 3600 Series NX-OS, versão 9.x
- Guia de programabilidade do Cisco Nexus 3600 Series NX-OS, versão 7.x
- Programabilidade e automação com o Cisco Open NX-OS