Install Docker Compose in NX-OS Bash Shell

Contents

Introduction

This document describes the steps used to install the Docker Compose package within the NX-OS Bash shell.

Cisco Nexus 3000 and 9000 series devices support Docker functionality within the Bash shell starting with NX-OS Release 9.2(1). As described by the Docker Compose documentation, "Compose is a tool for defining and running multi-container Docker applications." Docker Compose allows for application developers to define all of the services that constitute an application within a single YAML file named "docker-compose.yml". Then, with a single command, all of those services can be created, built, and started. Furthermore, all services can be stopped and monitored from within the Docker Compose suite of commands.

While Docker functionality is supported natively within the NX-OS Bash shell, Docker Compose must be installed separately.

Prerequisites

Requirements

This document requires the Bash shell to be enabled on your Cisco Nexus device. Refer to the "Accessing Bash" section of the Bash chapter in the Cisco Nexus 9000 Series NX-OS Programmability Guide for instructions to enable the Bash shell.

This document requires the Bash shell to be configured as a DNS client capable of resolving domain hostnames to IP addresses. Refer to the document for instructions to configure DNS servers within the Bash shell.

Components Used

The information in this document is based on these software and hardware versions:

- Nexus 9000 platform starting from NX-OS Release 9.2(1)
- Nexus 3000 platform starting from NX-OS Release 9.2(1)

The information in this document was created from devices in a specific lab environment. 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.

Configuring HTTP/HTTPS Proxies

If your environment requires the use of an HTTP or HTTPS proxy, the Bash shell will need to be configured to use these proxies before Docker Compose can be installed.

Log into the Bash shell as the root user through the `run bash sudo su -` command.

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

Temporarily Configuring HTTP/HTTPS Proxies

To temporarily configure HTTP/HTTPS proxies for this session, use the `export` command to define the "http_proxy" and "https_proxy" environment variables. An example of this is shown below, where "proxy.example-domain.com" is the hostname of a hypothetical proxy server.

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

Confirm that the environment variables are configured as desired using the `echo $http_proxy` and `echo $https_proxy` commands, as demonstrated below:

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

The values assigned to these environment variables will be cleared when the session is terminated, and will need to be reconfigured each time the Bash shell is entered. In the example below, the Bash session where the above configuration is exited, returning the prompt to NX-OS. Then, a new session to the Bash shell is created, where the environment variables have been cleared.

```
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#
```

Permanently Configuring HTTP/HTTPS Proxies

To permanently configure HTTP/HTTPS proxies for all sessions for a specific user that enters the Bash shell, the "http_proxy" and "https_proxy" environment variables must be exported automatically each time any user logs in. This can be accomplished by appending `export` commands to the .bash_profile file located in the user's directory, which Bash automatically loads when that user logs into the Bash shell. An example of this is shown below, where "proxy.example-domain.com" is the hostname of a hypothetical proxy server.

```
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/
```

If one wishes to configure specific HTTP/HTTPS proxies for all sessions for all users that enter the Bash shell, append these `export` commands to the /etc/profile file. Bash automatically loads this file first when any user logs into the Bash shell - as a result, all users that log into the Bash shell will have their HTTP/HTTPS proxies configured accordingly.

An example of this is shown below, where "proxy.example-domain.com" is the hostname of a hypothetical proxy server. The user account "docker-admin" is then configured with the Bash shelltype, which allows the user account to directly log into the Bash shell when remotely accessing the device. SSH is then used to access the mgmt0 IP address (192.0.2.1) of the Nexus device through the management VRF using the docker-admin user account. The example shows that the "http_proxy" and "https_proxy" environment variables have been set, even when a brand new user account has been logged into the Bash shell.

```
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/
```

Installing Docker Compose

To install Docker Compose, one must use the `wget` utility to download the latest binary release of Docker Compose, then place that binary in the /usr/bin directory.

1. Determine the latest stable version of Docker Compose available with the latest available releases on the Docker Compose GitHub page. Locate the version number for the latest stable release towards the top of the webpage. At the time of this writing, the latest stable release is 1.23.2.

2. Craft the URL for the Docker Compose binary by replacing `{latest-version}` in the URL below with the version number of the latest stable release found in the previous step:

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

For example, the URL for 1.23.2 at the time of this writing is as follows: https://github.com/docker/compose/releases/download/1.23.2/docker-compose-Linux-x86_64

3. Enter the Bash shell as root from the NX-OS prompt with the `run bash sudo su -` command, as demonstrated below:

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

4. If necessary, change the network namespace context of the Bash shell to a namespace with DNS and Internet connectivity. Network namespaces are logically identical to NX-OS VRFs. The example below demonstrates how to switch to the management network namespace context, which has DNS and Internet connectivity in this specific environment.

```
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. Enter the following command, replacing `{docker-url}` with the URL created in the previous step: `wget {docker-url} -O /usr/bin/docker-compose`. An example of this command executing is shown below, using https://github.com/docker/compose/releases/download/1.23.2/docker-compose-Linux-x86_64 as the replacement URL for `{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. Modify the permissions of the /usr/bin/docker-compose binary file such that it is executable using the `chmod +x /usr/bin/docker-compose` command. This is demonstrated below:

```
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 <arg>...]
[options] [COMMAND] [ARGS...] docker-compose -h|--help Options: -f, --file FILE Specify an
alternate compose file (default: docker-compose.yml) -p, --project-name NAME Specify an
alternate project name (default: directory name) --verbose Show more output --log-level LEVEL
Set log level (DEBUG, INFO, WARNING, ERROR, CRITICAL) --no-ansi Do not print ANSI control
characters -v, --version Print version and exit -H, --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 the name specified in the client certificate --project-directory PATH Specify
an alternate working directory (default: the path of the Compose file) --compatibility If set,
Compose will attempt to convert deploy keys in v3 files to their non-Swarm equivalent Commands:
build Build or rebuild services bundle Generate a Docker bundle from the Compose file config
Validate and view the Compose file create Create services down Stop and remove containers,
networks, images, and volumes events Receive real time events from containers exec Execute a
command in a running container help Get help on a command images List images kill Kill
containers logs View output from containers pause Pause services port Print the public port for
a port binding ps List containers pull Pull service images push Push service images restart
Restart services rm Remove stopped containers run Run a one-off command scale Set number of
containers for a service start Start services stop Stop services top Display the running
processes unpause Unpause services up Create and start containers version Show the Docker-
Compose version information
```

Verifying Docker Compose Functionality

One can verify that Docker Compose is installed successfully and functioning by creating and running a small docker-compose.yml file. The example below steps through this process.

```
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: Ensure that when the `docker-compose` command is executed, it is done so within the context of a network namespace that has DNS and Internet connectivity. Otherwise, Docker Compose will not be able to pull requested images from Docker Hub.

Note: To stand down a multi-container Docker application started by Docker Compose while attached to the Docker Compose session, press the "Ctrl+C" key combination.

Related Information

- Docker Compose Installation Documentation
- Docker Compose Documentation Overview
- Cisco Nexus 9000 Series NX-OS Programmability Guide, Release 9.x
- Cisco Nexus 9000 Series NX-OS Programmability Guide, Release 7.x
- Cisco Nexus 9000 Series NX-OS Programmability Guide, Release 6.x
- Cisco Nexus 3000 Series NX-OS Programmability Guide, Release 9.x
- Cisco Nexus 3000 Series NX-OS Programmability Guide, Release 7.x
- Cisco Nexus 3000 Series NX-OS Programmability Guide, Release 6.x
- Cisco Nexus 3500 Series NX-OS Programmability Guide, Release 9.x
- Cisco Nexus 3500 Series NX-OS Programmability Guide, Release 7.x
- Cisco Nexus 3500 Series NX-OS Programmability Guide, Release 6.x
- Cisco Nexus 3600 Series NX-OS Programmability Guide, Release 9.x
- Cisco Nexus 3600 Series NX-OS Programmability Guide, Release 7.x
- Programmability and Automation with Cisco Open NX-OS