NX-OS Bash Shell에서 Docker 작성 설치

목차

[소개](#)
[사전 요구 사항](#)
[요구 사항](#)
[사용되는 구성 요소](#)
[HTTP/HTTPS 프록시 구성](#)
[일시적으로 HTTP/HTTPS 프록시 구성](#)
[HTTP/HTTPS 프록시 영구 구성](#)
[Docker 작성 설치](#)
[Docker 작성 기능 확인](#)
[관련 정보](#)

소개

이 문서에서는 NX-OS Bash 셸 내에 Docker Compose 패키지를 설치하는 데 사용되는 단계에 대해 설명합니다.

Cisco Nexus 3000 및 9000 Series 장치는 NX-OS 릴리스 9.2(1)부터 Bash 셸 내에서 Docker 기능을 지원합니다. Docker 작성 문서에서 설명한 대로 "작성은 다중 컨테이너 Docker 응용 프로그램을 정의하고 실행하기 위한 도구입니다." Docker 작성을 사용하면 응용 프로그램 개발자가 "docker-compose.yml"이라는 단일 YAML 파일 내에서 응용 프로그램을 구성하는 모든 서비스를 정의할 수 있습니다. 그런 다음 하나의 명령으로 모든 서비스를 생성, 구축 및 시작할 수 있습니다.또한 Docker Compose 명령 모음 내에서 모든 서비스를 중지하고 모니터링할 수 있습니다.

Docker 기능은 NX-OS Bash 셸 내에서 기본적으로 지원되지만 Docker Compose는 별도로 설치해야 합니다.

사전 요구 사항

요구 사항

이 문서를 사용하려면 Cisco Nexus 디바이스에서 Bash 셸을 활성화해야 합니다.Bash 셸을 활성화하는 지침은 Cisco Nexus 9000 Series NX-OS Programmability Guide의 Bash 장의 "Accesing Bash" 섹션을 참조하십시오.

이 문서에서는 IP 주소에 대한 도메인 호스트 이름을 확인할 수 있는 DNS 클라이언트로 Bash 셸을 구성해야 합니다.Bash 셸 내에서 DNS 서버를 구성하는 방법은 문서를 참조하십시오.

사용되는 구성 요소

이 문서의 정보는 다음 소프트웨어 및 하드웨어 버전을 기반으로 합니다.

- NX-OS 릴리스 9.2(1)부터 시작하는 Nexus 9000 플랫폼

- NX-OS 릴리스 9.2(1)부터 시작하는 Nexus 3000 플랫폼

이 문서의 정보는 특정 랩 환경의 디바이스에서 생성되었습니다.이 문서에 사용된 모든 디바이스는 초기화된(기본) 컨피그레이션으로 시작되었습니다.현재 네트워크가 작동 중인 경우, 모든 명령어의 잠재적인 영향을 미리 숙지하시기 바랍니다.

HTTP/HTTPS 프록시 구성

환경에서 HTTP 또는 HTTPS 프록시를 사용해야 하는 경우 Docker 작성을 설치하기 전에 이러한 프록시를 사용하도록 Bash 셸을 구성해야 합니다.

`run bash sudo su -` 명령을 통해 루트 사용자로 Bash 셸에 로그인합니다.

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

일시적으로 HTTP/HTTPS 프록시 구성

이 세션에 대해 HTTP/HTTPS 프록시를 임시로 구성하려면 `export` 명령을 사용하여 "http_proxy" 및 "https_proxy" 환경 변수를 정의합니다.이 예제는 아래와 같습니다. 여기서 "proxy.example-domain.com"은 가상 프록시 서버의 호스트 이름입니다.

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

환경 변수가 `echo $http_proxy` 및 `echo $https_proxy` 명령을 사용하여 원하는 대로 구성되었는지 확인합니다.

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

이러한 환경 변수에 할당된 값은 세션이 종료될 때 지워지며 Bash 셸을 입력할 때마다 재구성해야 합니다.아래 예에서는 위 컨피그레이션이 종료되는 Bash 세션을 실행하여 프롬프트를 NX-OS로 반환합니다.그런 다음 환경 변수가 지워진 Bash 셸에 대한 새 세션이 생성됩니다.

```
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#
```

HTTP/HTTPS 프록시 영구 구성

Bash 셸을 입력하는 특정 사용자에 대해 모든 세션에 대해 HTTP/HTTPS 프록시를 영구적으로 구성하려면 사용자가 로그인할 때마다 "http_proxy" 및 "https_proxy" 환경 변수를 자동으로 내보내야

합니다.이 작업은 명령을 사용자 디렉토리에 있는 .bash_profile 파일에 추가하여 수행할 수 있습니다. 이 파일은 사용자가 Bash 셸에 로그인할 때 자동으로 로드됩니다.이 예제는 아래와 같습니다. 여기서 "proxy.example-domain.com"은 가상 프록시 서버의 호스트 이름입니다.

```
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/
```

Bash 셸을 입력하는 모든 사용자에 대해 모든 세션에 대해 특정 HTTP/HTTPS 프록시를 구성하려면 이러한 명령을 /etc/profile 파일에 추가합니다.Bash는 사용자가 Bash 셸에 로그인할 때 이 파일을 먼저 자동으로 로드합니다. 따라서 Bash 셸에 로그인하는 모든 사용자는 이에 따라 HTTP/HTTPS 프록시가 구성됩니다.

이 예제는 아래와 같습니다. 여기서 "proxy.example-domain.com"은 가상 프록시 서버의 호스트 이름입니다.그런 다음 사용자 어카운트 "docker-admin"이 Bash shelltype으로 구성됩니다. 이 경우 디바이스에 원격으로 액세스할 때 사용자 어카운트가 Bash 셸에 직접 로그인할 수 있습니다.그런 다음 docker-admin 사용자 계정을 사용하여 관리 VRF를 통해 Nexus 디바이스의 mgmt0 IP 주소(192.0.2.1)에 액세스하는 데 SSH를 사용합니다.이 예에서는 새로운 사용자 계정이 Bash 셸에 로그인된 경우에도 "http_proxy" 및 "https_proxy" 환경 변수가 설정되었음을 보여 줍니다.

```
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/
```

Docker 작성 설치

Docker Compose를 설치하려면 wget 유틸리티를 사용하여 Docker Compose의 최신 이진 릴리스를 다운로드한 다음 해당 이진 파일을 /usr/bin 디렉토리에 배치해야 합니다.

1. Docker Compose GitHub 페이지에서 사용 가능한 최신 릴리스와 함께 사용할 수 있는 Docker Compose의 최신 버전을 확인합니다.웹 페이지 맨 위에 있는 최신 안정적 릴리스의 버전 번호를 찾습니다.이 글을 쓸 때, 가장 최근의 안정적인 출시는 1.23.2.

2. 아래의 URL에서 `{latest-version}`을 이전 단계에서 찾은 최신 안정형 릴리스의 버전 번호로 대체하여 Docker 작성 바이너리의 URL을 작성합니다.

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

예를 들어, 이 작성 시 1.23.2의 URL은 다음과 같습니다
.https://github.com/docker/compose/releases/download/1.23.2/docker-compose-Linux-x86_64

3. 다음과 같이 run bash sudo `su -` 명령을 사용하여 NX-OS 프롬프트에서 Bash 셸을 root 입력합니다.

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

4. 필요한 경우 Bash 셸의 네트워크 네임스페이스 컨텍스트를 DNS 및 인터넷 연결을 사용하는 네임스페이스로 변경합니다.네트워크 네임스페이스는 NX-OS VRF와 논리적으로 동일합니다.아래 예는 이 특정 환경에서 DNS 및 인터넷 연결이 있는 관리 네트워크 네임스페이스 컨텍스트로 전환하는 방법을 보여줍니다.

```
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. 다음 명령을 입력하여 `{docker-url}`을 이전 단계에서 만든 URL로 바꿉니다.`wget {docker-url} -O /usr/bin/docker-compose`.다음은 이 명령을 실행하는 예입니다.
https://github.com/docker/compose/releases/download/1.23.2/docker-compose-Linux-x86_64을 `{docker-url}`의 대체 URL로 사용합니다.

```
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. /usr/bin/docker-compose 이진 파일의 사용 권한을 수정하여 `chmod +x /usr/bin/docker-compose` 명령을 사용하여 실행할 수 있도록 합니다.이 내용은 다음과 같습니다.

```
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
```

Docker 작성 기능 확인

작은 docker-compose.yml 파일을 만들고 실행하여 Docker Compose가 성공적으로 설치되었고 작동하는지 확인할 수 있습니다.아래 예에서는 이 프로세스를 단계별로 진행합니다.

```
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#
```

주의:docker-compose 명령이 실행될 때 DNS 및 인터넷 연결이 있는 네트워크 네임스페이스의 컨텍스트 내에서 실행되는지 확인합니다.그렇지 않으면 Docker 작성에서 요청한 이미지를 Docker 허브에서 가져올 수 없습니다.

참고: Docker 작성 세션에 연결된 상태에서 Docker 작성에서 시작한 다중 컨테이너 Docker 응용 프로그램을 중지하려면 "Ctrl+C" 키 조합을 누릅니다.

관련 정보

- Docker 작성 설치 설명서
- Docker 작성 문서 개요
- Cisco Nexus 9000 Series NX-OS 프로그래밍 가이드, 릴리스 9.x
- Cisco Nexus 9000 Series NX-OS 프로그래밍 가이드, 릴리스 7.x
- Cisco Nexus 9000 Series NX-OS 프로그래밍 가이드, 릴리스 6.x
- Cisco Nexus 3000 Series NX-OS 프로그래밍 가이드, 릴리스 9.x
- Cisco Nexus 3000 Series NX-OS 프로그래밍 가이드, 릴리스 7.x
- Cisco Nexus 3000 Series NX-OS 프로그래밍 가이드, 릴리스 6.x
- Cisco Nexus 3500 Series NX-OS 프로그래밍 가이드, 릴리스 9.x
- Cisco Nexus 3500 Series NX-OS 프로그래밍 가이드, 릴리스 7.x
- Cisco Nexus 3500 Series NX-OS 프로그래밍 가이드, 릴리스 6.x
- Cisco Nexus 3600 Series NX-OS 프로그래밍 가이드, 릴리스 9.x
- Cisco Nexus 3600 Series NX-OS 프로그래밍 가이드, 릴리스 7.x
- Cisco Open NX-OS를 통한 프로그래밍 기능 및 자동화