安装码头工人在NX-OS打击Shell组成

Contents

Introduction

本文描述用于的步骤安装码头工人组成在NX-OS打击shell内的程序包。

在开始从NX-OS版本9.2(1)的打击shell内的Cisco连结3000和9000系列设备支持码头工人功能。如所描述由码头工人请组成文档，“Compose是定义和运行多容器码头工人应用程序的一个工具”。码头工人组成允许应用开发员定义构成在名为“码头工人compose.yml的”单个YAML文件内的一个应用程序的所有服务。然后，与单个命令，所有那些服务可以创建，被构件和开始。此外，所有服务可以被终止，并且监控从码头工人的内部请组成命令组。

当码头工人功能在NX-OS打击shell内时本地支持，必须分开安装码头工人Compose。

Prerequisites

Requirements

本文在您的Cisco连结设备要求打击shell被启用。请参见打击章节的“访问打击”部分在Cisco连结9000系列NX-OS可编程序性指南的关于指令对enable (event)打击shell。

本文要求打击shell被配置作为DNS客户端能够解决域主机名对IP地址。请参见本文关于指令配置在打击shell内的DNS服务器。

Components Used

本文档中的信息基于以下软件和硬件版本：

- 连结9000平台由NX-OS版本开始9.2(1)
- 连结3000平台由NX-OS版本开始9.2(1)

本文的信息从设备在特定实验室环境里被创建了。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.

配置HTTP/HTTPS代理

如果您的环境要求使用HTTP或HTTPS代理，将需要配置打击shell使用这些代理，在可以安装前码头工人Compose。

日志到打击shell里作为root用户通过`su -sudo`。

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

临时配置HTTP/HTTPS代理

临时地配置此会话的HTTP/HTTPS代理，请使用命令定义“http_proxy”和“https_proxy”环境变量。此的示例如下所示，其中“proxy.example-domain.com”是主机名-一个假定代理服务器。

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

确认环境变量被配置如期望的一样使用`$http_proxy`并且`$https_proxy`命令，如下所示：

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

被赋予到这些环境变量将清除值，当会话被终止和需要重新配置，每次打击shell被输入。在下面的示例中的，上述配置被退出的打击会话，返回提示到NX-OS。然后，对打击shell的一个新会话被创建，其中清除了环境变量。

```
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#
```

永久性配置HTTP/HTTPS代理

的一个特定用户的必须每次自动地导出永久性配置所有会话的HTTP/HTTPS代理输入打击shell，“http_proxy”和“https_proxy”环境变量所有用户登录。这可以通过添附对位于用户目录的.bash_profile文件的命令完成，自动地打击负荷，当该用户登录到打击shell时。此的示例如下所示，其中“proxy.example-domain.com”是主机名-一个假定代理服务器。

```
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/
```

如果一希望配置所有会话的特定HTTP/HTTPS代理输入打击shell的所有用户的，请添附这些命令对/etc/profile文件。打击首先自动地装载此文件，当所有用户登录到打击shell -结果，登录到打击shell的所有用户把他们的HTTP/HTTPS代理相应地被配置。

此的示例如下所示，其中“proxy.example-domain.com”是主机名-一个假定代理服务器。用户帐户“码头工人Admin”然后配置有打击shelltype，允许用户帐户直接地登录到打击shell，当远程存取资料设备。SSH然后用于通过管理VRF访问mgmt0 IP地址(192.0.2.1)使用码头工人Admin用户帐户的连结设备。示例表示，设置“http_proxy”和“https_proxy”环境变量，既使当一个全新的用户帐户登录到打击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/
```

安装码头工人请组成

要安装码头工人请组成，一个必须使用`wget`工具下载码头工人最新的二进制版本组成，然后安置该二进制在/usr/bin目录里。

1. 确定码头工人的最新的稳定的版本组成可用与在码头工人的最新的可用的版本组成GitHub页。找出最新的稳定版本的版本号往网页的顶层。在此文字时，最新的稳定版本是1.23.2。

2. 制作码头工人的URL通过替换组成二进制`{}`在下面URL用在上一步找到的最新的稳定版本的版本号：

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

例如， 1.23.2的URL在此文字时如下：https://github.com/docker/compose/releases/download/1.23.2/docker-compose-Linux-x86_64

3. 输入打击shell，从NX-OS提示的根与`sudo su -`，如下所示：

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

4. 如果需要，请更改打击shell的网络namespace上下文到与DNS和互联网连通性的namespace。网络名空间与NX-OS VRF是逻辑上相同的。下面的示例展示如何换成管理网络namespace上下文，有DNS和互联网连通性在此特定环境。

```
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. 输入以下命令，替换`{URL}`用被创建的URL在上一步：`wget {URL} - O /usr/bin/docker-compose`。示例此命令执行如下所示，使用https://github.com/docker/compose/releases/download/1.23.2/docker-compose-Linux-x86_64作为更换URL为`{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. 修改/usr/bin/docker-compose二进制文件的权限这样使用`chmod +x /usr/bin/docker-compose`命令是可执行的。这如下被展示：

```
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
```

验证码头工人请组成功能

一能验证创建和运行一个小的码头工人compose.yml文件成功安装码头工人Compose和作用。示例在步骤之下通过此进程。

```
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#
```

警告：保证，当码头工人命令被执行，在有DNS和互联网连通性的网络namespace的上下文内如此完成。否则，码头工人Compose不能拉从码头工人集线器的被请求的镜像。

Note:要退休多容器码头工人应用程序由码头工人开始了组成，当附有码头工人时组成会话，按“Ctrl+C”密钥组合。

Related Information

- 码头工人组成安装文档
- 码头工人组成文档概述
- Cisco连结9000系列NX-OS可编程序性指南，版本9.x
- Cisco连结9000系列NX-OS可编程序性指南，版本7.x
- Cisco连结9000系列NX-OS可编程序性指南，版本6.x
- Cisco连结3000系列NX-OS可编程序性指南，版本9.x
- Cisco连结3000系列NX-OS可编程序性指南，版本7.x
- Cisco连结3000系列NX-OS可编程序性指南，版本6.x
- Cisco连结3500系列NX-OS可编程序性指南，版本9.x
- Cisco连结3500系列NX-OS可编程序性指南，版本7.x
- Cisco连结3500系列NX-OS可编程序性指南，版本6.x
- Cisco连结3600系列NX-OS可编程序性指南，版本9.x
- Cisco连结3600系列NX-OS可编程序性指南，版本7.x
- 可编程序性和自动化与Cisco开放NX-OS