NetScaler CPX 12

Contents

아키텍처및트래픽흐름

June 11, 2018

2016 년 5 월 13 일

Docker 호스트에서 NetScaler CPX 인스턴스를프로비전하면 Docker 엔진이 CPX 인스턴스에가상인터페이스 eth0 을만듭니다. 이 eth0 인터페이스는 docker0 브리지의가상인터페이스 (veth*) 에직접연결됩니다. 또한 Docker 엔진은네트워크 172.17.0.0/16 에서 NetScaler CPX 인스턴스에 IP 주소를할당합니다.

CPX 인스턴스의기본게이트웨이는 docker0 브리지의 IP 주소이며, NetScaler CPX 인스턴스와의모든통신이 Docker 네트워크를통해이루어진다는의미입니다. docker0 브리지에서수신되는모든들어오는트래픽은 NetScaler CPX 인스턴스의 eth0 인터페이스에서수신되고 NetScaler CPX 패킷엔진에의해처리됩니다.

다음그림에서는 Docker 호스트에서 NetScaler CPX 인스턴스의아키텍처를보여줍니다.

NetScaler CPX 에서단일 **IP** 주소가작동하는방식

일반 NetScaler MPX 또는 VPX 장비가기능을수행하려면적어도 3 개의 IP 주소가필요합니다.

- NSIP(NetScaler IP) 주소라고하는관리 IP 주소
- 서버팜과통신하기위한 SNIP(서브넷 IP) 주소
- 클라이언트요청을받아들이기위한 VIP(가상서버 IP) 주소

NetScaler CPX 인스턴스가단일 IP 주소로작동하며, 이 IP 주소하나가관리뿐만아니라데이터트래픽에도사용됩니다.

프로비저닝중에 Docker 엔진에의해사설 IP 주소하나 (단일 IP 주소) 가 NetScaler CPX 인스턴스에할당됩니다. NetScaler 인스턴스의세가지 IP 기능은한 IP 주소로멀티플렉싱됩니다. 이단일 IP 주소는서로다른포트번호를사용하여 NSIP, SNIP 및 VIP 기능을합니다.

다음이미지에서는단일 IP 주소가어떻게 NSIP, SNIP 및 VIP 기능으로사용되는지보여줍니다.

NetScaler CPX 인스턴스에서시작된요청의트래픽흐름

Docker 에서는 NetScaler CPX 인스턴스에서시작된트래픽을 docker0 IP 주소로전달하는 IP 표및 NAT 규칙을암시적으로구성합니다.

다음그림에서는 NetScaler CPX 인스턴스에서시작된 ping 요청이대상에도달하는방식을보여줍니다.

이예제에서 ping 요청은 eth0 인터페이스의패킷엔진에의해원본 IP 주소를 NetScaler CPX IP 주소 (172.17.0.4) 로사용하여전송됩니다. 계속해서 Docker 호스트가 NAT(Network Address Translation) 를수행하여호스트 IP 주소 (192.68.x.x) 를원본 IP 주소로추가하고요청을대상 (216.58.x.x) 으로보냅니다. 목적지 IP 주소의응답은동일한경로를반대방향으로따릅니다. Docker 호스트가응답에대한 NAT 를수행하고응답을 eth0 인터페이스의 NetScaler CPX 인스턴스로전달합니다.

외부네트워크에서시작된요청의트래픽흐름

외부통신을사용하도록설정하려면 NetScaler CPX 를프로비전하는동안 Docker 가 80, 22 및필요한다른모든포트를노출하도록매개변수를설정해야합니다. 프로비전하는동안포트를노출하도록설정하지않은경우 Docker 호스트에서 NAT 규칙을구성하여이러한포트를사용할수있게만들어야합니다.

인터넷에서시작된클라이언트요청이 Docker 호스트에의해수신되고, Docker 호스트가 PAT(포트주소변환) 를수행하여공용 IP 주소및포트를 NetScaler CPX 인스턴스의단일 IP 주소및포트에매핑하고트래픽을인스턴스로전달합니다.

다음그림에서는 Docker 호스트가포트주소변환을수행하여트래픽을 NetScaler CPX 단일 IP 주소및포트로전달하는방식을보여줍니다.

이예제에서 Docker 호스트 IP 주소는 192.68.x.x 이고 NetScaler CPX 인스턴스의단일 IP 주소는 172.17.0.4 입니다. NetScaler CPX 인스턴스의 SSH 포트 22 는 Docker 호스트에서포트 1100 에매핑됩니다. 클라이언트의 SSH 요청은 IP 주소 192.68.x.x 의포트 1100 에서수신됩니다. Docker 호스트는포트주소변환을수행하여이주소및포트를단일 IP 주소 172.17.0.4 의포트 22 에매핑하고클라이언트요청을전달합니다.

NetScaler CPX 라이센스

June 11, 2018

2017 년 7 월 18 일

CPX 인스턴스에라이센스를부여하면 NetScaler CPX 의성능을향상시킬수있습니다. NetScaler MAS 를사용하여 CPX 라이센스를풀링할수있으며 NetScaler MAS 를라이센스서버로사용할수있습니다. Citrix 에서구입한 LAC(라이센스액세스코드)를사용하거나라이센스파일을업로드하여 GUI 를통해 MAS 에서라이센스를설치할수있습니다. 라이센스를 CPX 인스턴스에할당하려면 Configuration Jobs(구성작업) 또는 CPX 의 Nitro 를사용할수있습니다. 다중 vCPU 코어를사용하여 NetScaler CPX 를프로비전하는경우라이센스풀에서 CPX 라이센스가각코어에하나씩할당됩니다. 예를들어 vCPU 코어 4 개를사용하여 NetScaler CPX 를프로비전한경우라이센스 4 개가 NetScaler CPX 인스턴스에할당됩니다.

NetScaler MAS 에서 **CPX** 라이센스파일을설치하려면**:**

1. 웹브라우저에서 **NetScaler Management and Analytics System** 의 IP 주소를입력합니다 (예: `http://192.168.100.1`).
2. **User Name(**사용자이름**)** 및 **Password(**암호**)** 에서관리자자격증명을입력합니다.
3. **Networks(**네트워크**)** > **Licenses(**라이센스**)** 로이동합니다.
4. 세부정보창에서 **License Files(**라이센스파일**)** 로이동하고다음옵션중하나를선택합니다.
    - **Upload license files from a local computer(**로컬컴퓨터에서라이센스파일업로드**)** - 라이센스가이미로컬컴퓨터에있는경우 **Browse(**찾아보기**)** 를클릭하고라이센스할당에사용하려는라이센스파일 (.lic) 을선택합니다. **Finish(**마침**)** 를클릭합니다.

- **Use License Activation Code(**라이센스활성화코드사용**)** - Citrix 에서는구입한라이센스의 LAC 를전자메일로보냅니다. 텍스트상자에 LAC 를입력한다음 **Get Licenses(**라이센스가져오기**)** 를클릭합니다.

  > 참고**:**
  > 이옵션을선택한경우 NetScaler Management and Analytics System 이인터넷에연결되어있거나프록시서버를사용할수있어야합니다.

1. **Browse(**찾아보기**)** 를클릭하고라이센스할당에사용하려는라이센스파일 (.lic) 을선택합니다. **Finish(**마침**)** 를클릭합니다.

언제라도 License Settings(라이센스설정) 에서 NetScaler Management and Analytics System 에더많은라이센스를추가할수있습니다.

확인

Networks(네트워크**)** > **Licenses(**라이센스**)** > **CPX Licenses(CPX** 라이센스**)** 로이동하여 NetScaler MAS 에설치된라이센스를확인할수있습니다.

MAS Configuration Jobs(구성작업**)** 를사용하여 **NetScaler CPX** 인스턴스에라이센스할당

NetScaler MAS 를사용하여구성작업별로 NetScaler CPX 인스턴스에라이센스를할당할수있습니다.

작업을사용하여 **NetScaler CPX** 에라이센스를할당하려면**:**

1. 관리자자격증명을사용하여 NetScaler MAS 에로그온합니다.

2. **Networks(**네트워크**)** > **Configuration Jobs(**구성작업**)** 로이동한다음 **Create Job(**작업만들기**)** 을클릭합니다.

3. 필요한값을지정하고구성원본을선택한후아래이미지에표시된것처럼다음명령을입력합니다.

> 참고**:**
>
> 기본적으로 NetScaler MAS 에서라이센스서버포트번호는 27000 입니다.

4. 구성을실행하려는 NetScaler CPX 인스턴스를선택하고 **Next(**다음**)** 를클릭합니다.

5. 실행설정을지정하고 **Finish(**마침**)** 를클릭하여 NetScaler CPX 인스턴스에서명령을실행합니다. 구성을저장하여나중에실행하려면 **Save and Exit(**저장하고끝내기**)** 를클릭합니다.

작업을사용하여 **NetScaler CPX** 에서라이센스를제거하려면**:**

1. 관리자자격증명을사용하여 NetScaler MAS 에로그온합니다.

2. **Networks(**네트워크**)** > **Configuration Jobs(**구성작업**)** 로이동한다음 **Create Job(**작업만들기**)** 을클릭합니다.

3. 필요한값을지정하고구성원본을선택한후아래이미지에표시된것처럼다음명령을입력합니다.

4. 구성을실행하려는 NetScaler CPX 인스턴스를선택하고 **Next(**다음**)** 를클릭합니다.

5. 실행설정을지정하고 **Finish(**마침**)** 를클릭하여 NetScaler CPX 인스턴스에서명령을실행합니다. 구성을저장하여나중에실행하려면 **Save and Exit(**저장하고끝내기**)** 를클릭합니다.

Nitro API 를사용하여 **NetScaler CPX** 인스턴스라이센스할당

NetScaler Nitro API 를사용하여 NetScaler CPX 인스턴스에라이센스를할당할수있습니다.

NetScaler CPX 인스턴스에라이센스서버를추가하려면**:**

1. http://\<netscaler-ip-address\>/nitro/v1/config/nslicenseserver로이동합니다.

2. HTTP 메서드를 POST로설정합니다.

3. 요청헤더를추가합니다.

```
1     Cookie:NITRO_AUTH_TOKEN=<tokenvalue>
2     Content-Type:application/json
```

4. 요청페이로드를입력합니다.

```
1     {
2   ”nslicenseserver”:{
3   ”licenseserverip”:<String_value>,”servername”:<String_value>,”port”:<
       Double_value> }
4    }
```

5. 다음과같은응답을볼수있습니다.

```
  HTTP Status Code on Success: 201 Created

       HTTP Status Code on Failure: 4xx \<string\> (for general HTTP errors)
or 5xx \<string\> (for NetScaler-specific errors). The response payload
provides details of the error
```

NetScaler CPX 인스턴스의용량을설정하려면**:**

1. http://\<netscaler-ip-address\>/nitro/v1/config/nscapacity로이동합니다.

2. HTTP 메서드를 PUT로설정합니다.

3. 다음과같이요청헤더를추가합니다.

```
1     Cookie:NITRO_AUTH_TOKEN=<tokenvalue>
2     Content-Type:application/json
```

4. 다음과같이요청페이로드를입력합니다.

```
1     {
2   ”nscapacity”:{
3
4
5           ”platform”:<String_value>
6
```



```
7         }
8     }
```

5. 다음과같은응답을볼수있습니다.

```
  HTTP Status Code on Success: 200 OK

       HTTP Status Code on Failure: 4xx \<string\> (for general HTTP errors)
or 5xx \<string\> (for NetScaler-specific errors). The response payload
provides details of the error
```

NetScaler CPX 인스턴스라이센스에대한자세한내용을보려면 NetScaler 장비의 **Downloads(**다운로드**)** 섹션에서사용할수있는 NITRO API 설명서를다운로드하십시오.

Docker 에서 **NetScaler CPX** 인스턴스배포

June 13, 2018

2017 년 8 월 30 일

NetScaler CPX 인스턴스는 Docker Store 에서 Docker 이미지파일로사용할수있습니다. 인스턴스를배포하려면 Docker Store 에서 NetScaler CPX 이미지를다운로드한후 `docker run` 명령또는 Docker Compose 도구를사용하여인스턴스를배포합니다.

사전요구사항

확인할사항:

- Docker 호스트시스템에최소한다음구성요소가있습니다.
  - CPU 1 개
  - 2GB RAM

    > 참고**:** NetScaler CPX 인스턴스를시작하는데사용할처리엔진수를정의하여 NetScaler CPX 의성능을높일수있습니다. 처리엔진을추가할때마다 Docker 호스트에동일한수의 vCPU 와동일한양 (GB) 의메모리가포함되도록해야합니다. 예를들어처리엔진 4 개를추가하려는경우 Docker 호스트에 vCPU 4 개와 4GB 메모리가있어야합니다.
- Docker 호스트시스템에서 Linux Ubuntu 버전 14.04 이상이실행되고있습니다.
- Docker 버전 1.12 가호스트시스템에설치되어있습니다. Linux 에서 Docker 를설치하는것에대한자세한정보는다음을참조하십시오.https://docs.docker.com/engine/installation/ubuntulinux/
- Docker 호스트가인터넷에연결되어있습니다.

Docker Store 에서 **NetScaler CPX** 이미지다운로드

Docker Store 에서 NetScaler CPX Docker 이미지파일을다운로드한후 NetScaler CPX Docker 이미지를로드합니다. 이를위해 Docker 호스트에서다음명령을실행합니다.

```
docker pull store/citrix/netscalercpx:12.0-53.xx
```

NetScaler CPX Docker 이미지를다운로드한후다음명령을사용하여이미지세부정보를볼수있습니다.

```
    docker images
```

예를들면다음과같습니다.

```
    root@ubuntu:~# docker images


    REPOSITORY    TAG         IMAGE ID       CREATED    VIRTUAL SIZE


       cpx       12.0-53.xx  2e97aadf918b  43 hours ago  414 MB
```

이예제에서출력의다음필드에주의하십시오.

- REPOSITORY(저장소). 이미지가저장되어있는이름공간을지정합니다.
- TAG(태그). 설치되어있는 NetScaler CPX 이미지의버전을나타냅니다.
- IMAGE ID(이미지 ID). Docker 호스트에서이미지의고유 ID 를나타냅니다.

docker run 명령을사용하여 **NetScaler CPX** 인스턴스배포

호스트에로드한 NetScaler CPX Docker 이미지를사용하여호스트에서 Docker 컨테이너에 NetScaler CPX 인스턴스를설치할수있습니다. `docker run` 명령을사용하여기본 NetScaler CPX 구성으로 NetScaler CPX 인스턴스를설치합니다.

> 중요:
>
> https://www.citrix.com/products/netscaler-adc/cpx-express.html에서 NetScaler CPX Express 를 다운로드한경우 NetScaler CPX 인스턴스를배포하는동안 https://www.citrix.com/products/netscaler-adc/cpx-express.html에있는 EULA(최종사용자사용권계약) 를읽고이해한후이계약에동의해야합니다.

다음과같은 ***docker run*** 명령을사용하여 Docker 컨테이너에 NetScaler CPX 인스턴스를설치합니다.

```
docker run -dt -P --privileged=true  -net=host  -e NS_NETMODE=" HOST"
    -e CPX_CORES=<number of cores> --name <container_name> --ulimit core
    =-1 -e CPX_NW_DEV='<INTERFACES>' -e CPX_CONFIG=' {
  "YIELD" :" NO"  }
  '  -e LS_IP=<LS_IP_ADDRESS> -e LS_PORT=<LS_PORT> e PLATFORM=CP1000 -v
     <host_dir>:/cpx <REPOSITORY>:<TAG>
```



```
1 docker run -dt --privileged=true --net=host -e NS_NETMODE="HOST" -e
     CPX_NW_DEV='eth1 eth2' -e CPX_CORES=5  - e CPX_CONFIG='{
2  "YIELD":"No" }
3  ' -e LS_IP=10.102.38.134 -e PLATFORM=CP1000 -v /var/cpx:/cpx --name
      cpx_host cpx:12.0-51.xx
```

이예제에서는 NetScaler CPX Docker 이미지를기반으로 mycpx 라는컨테이너를만듭니다.

-P 매개변수는필수항목이며, NetScaler CPX Docker 이미지에의해컨테이너에서노출되는포트 (포트 80, 22, 443, 161/UDP) 를 Docker 호스트의사용자정의범위에서무작위로선택되는포트에매핑하도록 Docker 에지시합니다. 이는충돌을방지하기위함입니다. 이후에동일한 Docker 호스트에 NetScaler CPX 컨테이너를여러개만들경우에대비해서필요합니다. 포트매핑은동적으로이루어지며컨테이너를시작하거나다시시작할때마다설정됩니다. 포트는다음과같이사용됩니다.

- 80 은 HTTP 에사용됩니다.
- 443 은 HTTPS 에사용됩니다.
- 22 는 SSH 에사용됩니다.
- 161/UDP 는 SNMP 에사용됩니다.

정적포트매핑을원하는경우 -p 매개변수를사용하여수동으로설정합니다.

--privileged=**true** 옵션은승격된권한모드로컨테이너를실행하는데사용됩니다. 여러코어를사용하여 NetScaler CPX 를실행하는경우모든시스템권한을 NetScaler CPX 에부여해야합니다. 단일코어를사용하여 NetScaler CPX 를실행하려는경우이옵션대신 --cap-add=NET_ADMIN 옵션을사용하여전체네트워크권한으로 NetScaler CPX 컨테이너를실행할수있도록해야합니다.

**--net=host는표준 docker run 명령옵션으로, 컨테이너가호스트네트워크스택에서실행되며모든네트워크장치에액세스할수있도록지정합니다.

> 참고
>
> 브리지또는비네트워크에서 NetScaler CPX 를실행하는경우이옵션을무시합니다.

-e NS_NETMODE="HOST"는 NetScaler CPX 관련환경변수로서, 이를통해 NetScaler CPX 가호스트모드로시작되도록지정할수있습니다. NetScaler CPX 가호스트모드로시작되는경우 NetScaler CPX 에대한관리액세스를위해호스트컴퓨터에기본 iptable 규칙 4 개가구성됩니다. 이때다음포트가사용됩니다.

- 9995 - HTTP 용
- 9996 - HTTPS 용
- 9997 - SSH 용
- 9998 - SNMP 용

다른포트를지정하려는경우다음환경변수를사용할수있습니다.

- -e NS_HTTP_PORT=
- -e NS_HTTPS_PORT=
- -e NS_SSH_PORT=
- -e NS_SNMP_PORT=

> 참고
>
> 브리지또는비네트워크에서 NetScaler CPX 를실행하는경우이환경변수를무시합니다.

-e CPX_CORES는선택적인 NetScaler CPX 관련환경변수입니다. 이환경변수로 NetScaler CPX 컨테이너를시작하는데사용할처리엔진수를정의하여 NetScaler CPX 인스턴스의성능을높일수있습니다.

> 참고
>
> 처리엔진을추가할때마다 Docker 호스트에동일한수의 vCPU 와동일한양 (GB) 의메모리가포함되도록해야합니다. 예를들어처리엔진 4 개를추가하려는경우 Docker 호스트에 vCPU 4 개와 4GB 메모리가있어야합니다.

-e EULA = yes는필수 NetScaler CPX 관련환경변수로서 https://www.citrix.com/products/netscaler-adc/cpx-express.html에있는 EULA(최종사용자사용권계약) 를읽고이해했음을확인하는데필요합니다.

-e PLATFORM=CP1000 매개변수는 NetScaler CPX 라이센스유형을지정합니다.

호스트네트워크에서 Docker 를실행하는경우 -e CPX_NW_DEV 환경변수를사용하여 NetScaler CPX 컨테이너에전용네트워크인스턴스를할당할수있습니다. 네트워크인터페이스를공백으로구분하여정의해야합니다. 정의된네트워크인터페이스는 NetScaler CPX 컨테이너를제거할때까지 NetScaler CPX 컨테이너에유지됩니다. NetScaler CPX 컨테이너가프로비저닝되면할당된모든네트워크인터페이스가 NetScaler 네트워킹네임스페이스에추가됩니다.

> 참고
>
> 브리지또는비네트워크에서 NetScaler CPX 를실행하며 Docker 또는컨테이너네트워크를변경하는경우 (예: 컨테이너에대한다른네트워크연결구성또는기존네트워크제거) 업데이트된네트워크를사용하기위해 NetScaler CPX 컨테이너를다시시작해야합니다.

```
docker run -dt --privileged=true --net=host -e NS_NETMODE="HOST" -e
    EULA=yes -e CPX_NW_DEV='eth1 eth2' -e CPX_CORES=5 -e PLATFORM=CP1000
     --name cpx_host cpx:12.0-53.x
```

-e CPX_CONFIG는 NetScaler CPX 관련환경변수로서, 이를사용하여 NetScaler CPX 컨테이너의처리량을제어할수있습니다. NetScaler CPX 가처리할수신트래픽이들어오지않을경우이러한유휴시간동안 CPU 를양보하여처리량이낮아집니다. 이러한경우에 CPX_CONFIG 환경변수를사용하여 NetScaler CPX 컨테이너의처리량을제어할수있습니다. CPX_CONFIG 환경변수에 JSON 형식으로다음값을제공해야합니다.

- NetScaler CPX 컨테이너가유휴상황에서 CPU 를양보하도록하려면다음과같이정의합니다. { "YIELD" : "Yes" }
- NetScaler CPX 컨테이너가유휴상황에서 CPU 를양보하지않도록해높은처리량을얻으려면다음과같이정의합니다. { "YIELD" : "No" }

```
docker run -dt --privileged=true --net=host -e NS_NETMODE="HOST" -e
    EULA=yes -e CPX_CORES=5  - e CPX_CONFIG='{
 "YIELD":"No" }
 ' -e PLATFORM=CP1000 --name cpx_host cpx:12.0-51.x
```



```
docker run -dt --privileged=true --net=host -e NS_NETMODE="HOST" -e
   EULA=yes -e CPX_CORES=5  -e CPX_CONFIG='{
 "YIELD":"Yes" }
 ' -e PLATFORM=CP1000 --name cpx_host cpx:12.0-51.xx
```

-v 매개변수는선택적매개변수로서 NetScaler CPX 탑재디렉터리 /cpx의탑재지점을지정합니다. 탑재지점은호스트에서 /cpx 디렉터리를탑재하는디렉터리입니다. /cpx 디렉터리에는로그, 구성파일, SSL 인증서및코어덤프파일이저장됩니다. 예제에서탑재지점은 /var/cpx이고 NetScaler CPX 탑재디렉터리는 /cpx입니다.

라이센스를구입했거나평가라이센스가있는경우라이센스서버에라이센스를업로드하고 -e LS_IP=\<LS_IP_ADDRESS\> -e LS_PORT=\<LS_PORT\> 매개변수를사용하여 docker run 명령으로해당라이센스서버위치를지정할수있습니다. 이경우에는 EULA 에동의하지않아도됩니다.

```
docker run -dt --privileged=true --net=host -e NS_NETMODE="HOST" -e
   CPX_CORES=5  -e CPX_CONFIG='{
 "YIELD":"No" }
 ' -e LS_IP=10.102.38.134 -e PLATFORM=CP1000 --name cpx_host cpx
    :12.0-51.xx
```

여기서:

- <LS_IP_ADDRESS> 는라이센스서버의 IP 주소입니다.
- <LS_PORT> 는라이센스서버의포트입니다.

다음명령을사용하여시스템에서실행되고있는이미지와표준포트에매핑된포트를확인할수있습니다. docker ps

Docker Compose 를사용하여 NetScaler CPX 인스턴스배포

Docker Compose 도구를사용하여단일 NetScaler CPX 인스턴스또는다중 NetScaler CPX 인스턴스를프로비전할수있습니다. Compose 를사용하여 NetScaler CPX 인스턴스를프로비전하려면먼저 NetScaler CPX 이미지, NetScaler CPX 인스턴스용으로열려는포트및사용자의 NetScaler CPX 인스턴스권한을지정하여 Compose 파일을작성해야합니다.

> 중요
>
> 호스트에 Docker Compose 도구가설치되어있는지확인하십시오.

다중 **NetScaler CPX** 인스턴스를프로비전하려면**:**

1. 다음을참고하여 Compose 파일을작성합니다.

- **<service-name>** 은프로비전하려는서비스의이름입니다.
- **image:<repository>:<tag>** 는 NetScaler CPX 이미지의저장소와버전을의미합니다.
- **privileged: true** 는 NetScaler CPX 인스턴스에모든루트권한을제공합니다.
- **cap_add** 는 NetScaler CPX 인스턴스에네트워크권한을부여합니다.
- **<host_directory_path>** 는 NetScaler CPX 인스턴스에탑재할 Docker 호스트의디렉터리입니다.

- **<number_processing_engine>** 은 NetScaler CPX 인스턴스를시작하는데사용할처리엔진수입니다. 처리엔진을추가할때마다 Docker 호스트에동일한수의 vCPU 와동일한양 (GB) 의메모리가포함되도록해야합니다. 예를들어처리엔진 4 개를추가하려는경우 Docker 호스트에 vCPU 4 개와 4GB 메모리가있어야합니다.

일반적으로 Compose 파일은다음과유사한형식을따릅니다.

```
    <service-name>:
    container_name:
    image: <repository>:<tag>
    ports:
        - 22
        - 80
        - 443
        - 161/udp
        - 35021-35030
    tty: true
    cap_add:
        - NET_ADMIN
    ulimits:
    core: -1
    volumes:
        - <host_directory_path>:/cpx
    environment:
        - EULA=yes
        - CPX_CORES=<number_processing_engine>
        - CPX_CONFIG='{
"YIELD":"Yes" }
'
```

```
    CPX_0:
    container_name: CPX_0
    image: cpx:12.0-53.xx
    ports:
        -  443
        -  22
        -  80
        -  161/udp
        -  35021-35030
    tty: true
    cap_add:
        - NET_ADMIN
    ulimits:
        core: -1
    volumes:
        - /root/test:/cpx
```



```
17      environment:
18          -  CPX_CORES=2
19          -  EULA=yes
```

> 참고: 단일 NetScaler CPX 인스턴스를프로비전하고싶다면 Compose 파일에다음줄을추가해야합니다.
> `container_name:\<name_of_container\>`
> 다음명령을실행하여다중 NetScaler CPX 인스턴스를프로비전합니다.
> `docker-compose -f <compose_file_name> scale <service-name>=<number of instances> up -d docker-compose -f docker-compose.yml scale cpxlb=3 up -d`
> 단일 NetScaler CPX 인스턴스를프로비전하고싶다면다음명령을실행합니다. `docker-compose -f \<compose_file_name\> up -d`

NetScaler Management and Analytics System 에 NetScaler CPX 인스턴스추가

June 11, 2018

2017 년 4 월 28 일

NetScaler CPX 인스턴스를관리하고모니터링하려면 Docker 호스트에설치된이러한인스턴스를 NetScaler MAS 서버에추가해야합니다.

NetScaler MAS 서버를처음설정하거나나중에설정할때인스턴스를추가할수있습니다.

인스턴스를추가하려면각인스턴스의호스트이름또는 IP 주소를지정하거나 IP 주소범위를지정해야합니다. 그런다음 NetScaler MAS 에서인스턴스에액세스할때사용할수있는인스턴스프로필을지정해야합니다.

이인스턴스프로필에는 NetScaler MAS 에추가할인스턴스의사용자이름과암호가포함됩니다. 각인스턴스유형에대해기본프로필이제공됩니다. 예를들어 NetScaler 인스턴스의기본프로필은 ns-root-profile 입니다. 이프로필은기본 NetScaler 관리자자격증명으로정의됩니다. 인스턴스의기본관리자자격증명을변경한경우인스턴스에대한사용자지정인스턴스프로필을정의할수있습니다. 인스턴스가검색된후에인스턴스자격증명을변경한경우인스턴스프로필을편집하거나새프로필을만든다음인스턴스를다시검색해야합니다.

사전요구사항

다음을수행해야합니다.

- Citrix XenServer 에 NetScaler MAS 서버설치. 자세한내용은 NetScaler MAS 설명서를참조하십시오.
- Docker 호스트에 NetScaler CPX 인스턴스설치.

NetScaler CPX 인스턴스를 **NetScaler MAS** 에추가하려면**:**

1. 웹브라우저에서 **NetScaler Management and Analytics System** 의 IP 주소를입력합니다 (예: `http://192.168.100.1`).

2. **User Name(**사용자이름**)** 및 **Password(**암호**)** 필드에관리자자격증명을입력합니다. 기본관리자자격증명은 **nsroot** 및 **nsroot** 입니다.

3. **Infrastructure(**인프라**) > Instances(**인스턴스**)** 로이동합니다. **Instances(**인스턴스**)** 아래에서 **NetScaler CPX** 를선택하고 **Add(**추가**)** 를클릭합니다.

4. 다음옵션중하나를선택합니다.

- **Enter Device IP address(**장치 **IP** 주소입력**)** - 각인스턴스의호스트이름또는 IP 주소를지정하거나 IP 주소범위를지정합니다.

- **Import from file(**파일에서가져오기**)** - 로컬시스템에서추가할모든인스턴스의 IP 주소가포함된텍스트파일을업로드합니다.

5. **Profile Name(**프로필이름**)** 에서해당하는인스턴스프로필을선택하거나 **+** 아이콘을클릭하여새프로필을만듭니다.

6. 호스트의 **HTTP**, **HTTPS**, **SSH** 및 **SNMP** 포트세부정보를지정합니다. **Start Port(**시작포트**)** 및 **Number of ports(**포트수**)** 필드에호스트에서게시한포트범위를지정할수도있습니다. 또한 NetScaler MAS 서버에서 NetScaler CPX IP 주소에접속할수있는경우 **Routable(**라우팅가능**)** 확인란을선택합니다. 호스트를통해 NetScaler CPX IP 주소에접속할수있는경우 **Routable(**라우팅가능**)** 확인란을선택취소하고호스트의 IP 주소를지정합니다.

7. **OK(**확인**)** 를클릭하여 NetScaler MAS 에인스턴스를추가하는프로세스를시작합니다.

> 참고
>
> 인스턴스를다시검색하려는경우 **Infrastructure(인프라) > Instances(인스턴스) > NetScaler <Instance Type>(NetScaler)** 으로이동하고, 다시검색할인스턴스를선택한후, **Action(동작)** 드롭다운목록에서 **Rediscover(다시검색)** 를클릭합니다. 인스턴스 >

NetScaler CPX 구성

June 13, 2018

2017 년 4 월 28 일

Linux Docker 호스트를통해 CLI 프롬프트에액세스하거나 NetScaler Nitro API 를사용하여 NetScaler CPX 인스턴스를구성할수있습니다.

명령줄인터페이스를사용하여 **NetScaler CPX** 인스턴스구성

Docker 호스트에액세스하고다음그림과같이인스턴스의 SSH 프롬프트에로그온합니다. NetScaler CPX 인스턴스에로그온할수있는기본관리자자격증명은 root/linux 입니다.

인스턴스의명령줄프롬프트를사용하여 CLI 명령을실행하려면다음명령을입력합니다. **cli_script.sh** “< 명령 >”

예:

인스턴스프롬프트에서로그아웃하려면 **log out** 을입력합니다.

Nitro API 를사용하여 **NetScaler CPX** 인스턴스구성

NetScaler Nitro API 를사용하여 NetScaler CPX 인스턴스를구성할수있습니다.

웹브라우저에서 **Nitro API** 를사용하여 **NetScaler CPX** 인스턴스를구성하려면다음을입력합니다.

```
http://\<host_IP_address\>:\<port\>/nitro/v1/config/\<resource-type\
```

웹브라우저에서 **Nitro API** 를사용하여통계를검색하려면다음을입력합니다.

```
http://\<host_IP_address\>:\<port\>/nitro/v1/stat/\<resource-type\
```

Nitro API 사용에대한자세한내용은 REST Web Services(REST 웹서비스)를참조하십시오. NetScaler CPX 의경우 `netscaler-ip-address`가나오는위치에 `CPX IP address:port`를사용하십시오.

작업을사용하여 **NetScaler CPX** 인스턴스구성

NetScaler MAS 에서작업을만들고실행하여 NetScaler CPX 인스턴스를구성할수있습니다. 구성템플릿의구성을사용하고, 다른장치에서사용가능한구성을추출하고, 텍스트파일에저장된구성을사용할수있습니다. 또한다른인스턴스의구성유틸리티를사용하여수행한구성을기록할수도있습니다. 그러면 NetScaler MAS 에 NetScaler CPX 인스턴스에서사용할 CLI 명령이표시됩니다. 구성을선택한후구성을로드할 NetScaler CPX 인스턴스를선택하고변수값을지정한다음작업을실행해야합니다.

작업을사용하여 **NetScaler CPX** 인스턴스를구성하려면**:**

1. 관리자자격증명을사용하여 NetScaler MAS 에로그온합니다.
2. **Infrastructure(**인프라**) > Configuration Jobs(**구성작업**)** 로이동한다음 **Create Job(**작업만들기**)** 을클릭합니다.
3. 필요한값을지정하고구성원본을선택합니다. 실행하려는명령을입력할수도있습니다.
4. 구성을실행하려는 NetScaler CPX 인스턴스를선택하고 **Next(**다음**)** 를클릭합니다.
5. 실행설정을지정하고 Finish(마침) 를클릭하여 NetScaler CPX 인스턴스에서명령을실행합니다. 구성을저장하여나중에실행하려면 **Save and Exit(**저장하고끝내기**)** 를클릭합니다.

NetScaler CPX 인스턴스에서로그스트리밍구성

NetScaler CPX 인스턴스에서로그스트리밍을구성하여응용프로그램성능모니터링및분석에필요한웹페이지성능데이터, 흐름및사용자세션수준정보, 데이터베이스정보를수집할수있습니다. 이러한데이터레코드는 NetScaler MAS 로전송되고여기에서모든응용프로그램에대한실시간보고서와기록보고서를볼수있습니다.

NetScaler CPX 인스턴스에서로그스트리밍을구성하려면먼저 NetScaler CPX 인스턴스에서 AppFlow 기능과 ulfd 디먼을사용하도록설정해야합니다. ulfd 디먼을사용하도록설정할때실시간보고서와기록보고서를모니터링하려는 NetScaler MAS의 IP 주소를지정해야합니다. 그런다음 AppFlow 수집기, 동작, 정책을구성하고 AppFlow 정책을전역적으로바인딩해야합니다.

ulfd 디먼은모든흐름레코드를단일통합로깅형식으로 NetScaler MAS 로보냅니다.

NetScaler CPX 인스턴스의명령줄인터페이스를사용하거나 NetScaler MAS 의 Jobs(작업) 기능을사용하여로그스트리밍을구성할수있습니다.

레코드를모니터링하려면먼저 NetScaler CPX 인스턴스를 NetScaler MAS 에추가해야합니다. NetScaler MAS 에 NetScaler CPX 인스턴스를추가하는데대한자세한내용은 NetScaler Management and Analytics System 을사용하여 NetScaler CPX 인스턴스설치를참조하십시오.

NetScaler CPX 인스턴스에서로그스트리밍을구성하려면**:**

1. 다음명령을실행하여 AppFlow 기능을사용하도록설정합니다. `enable ns feature AppFlow`
2. 다음명령을사용하여 ulfd 디먼을사용하도록설정합니다. `set ns param -ulfd ENABLED -loggerip <NUMS_IP_Address>`
3. 다음명령을실행하여 AppFlow 수집기, 동작및정책을구성하고정책을전역적으로바인딩합니다.

```
1     add appflow collector <name> -IPAddress <ipaddress>‘
2
3     set appflow param -templateRefresh 3600 -httpUrl ENABLED -
         httpCookie ENABLED -httpReferer ENABLED -httpMethod ENABLED -
         httpHost ENABLED -httpUserAgent ENABLED -httpContentType ENABLED
          -httpAuthorization ENABLED -httpVia ENABLED -httpXForwardedFor
         ENABLED -httpLocation ENABLED -httpSetCookie ENABLED -
         httpSetCookie2 ENABLED -connectionChaining ENABLED -httpDomain
         Enabled
4
5     add appflow action <name> --collectors <string> ... [-
         clientSideMeasurements (Enabled|Disabled) ]
6
7     add appflow policy <name> true <action>
8
9     bind appflow global <policyName> <priority> [<
         gotoPriorityExpression [-type <type>]
```

> 참고**:**
> AppFlow 수집기를더미 IP 주소로구성해야합니다.

구성파일을사용하여 **NetScaler CPX** 구성

June 11, 2018

2018 년 3 월 21 일

NetScaler CPX 인스턴스를배포하는동안명령줄인터페이스 (`cli_script.sh`), NITRO API 또는 NetScaler MAS(Management and Analytics) 구성작업을사용하여 NetScaler CPX 를구성하는대신정적구성파일을사용하여 NetScaler CPX 를동적으로구성할수있습니다.

NetScaler CPX 컨테이너를배포하는동안정적구성파일을입력파일로제공할수있습니다. NetScaler CPX 컨테이너가시작하는동안컨테이너는정적구성파일에지정된구성을기반으로구성됩니다. 이구성에는 NetScaler CPX 컨테이너에서동적으로실행할수있는배시셸 (shell) 명령과 NetScaler 관련구성이포함됩니다.

정적구성파일의구조

위에서언급한바와같이 NetScaler CPX 는배포될때정적구성파일에지정된구성을기반으로구성됩니다.

정적구성파일은 `##NetScaler Commands`와 `##Shell Commands`라는두개의태그가포함된 `.conf` 파일입니다. `##NetScaler Commands` 태그에서는 NetScaler CPX 에서 NetScaler 관련구성을구성할수있도록모든 NetScaler 명령을추가해야합니다. `##Shell Commands` 태그에서는 NetScaler CPX 에서실행할셸명령을추가해야합니다.

NetScaler CPX 컨테이너를배포하는동안 NetScaler 명령과셸명령이구성파일에지정된순서로컨테이너에서실행됩니다.

> 중요:
>
> - 태그는구성파일에서여러번반복될수있습니다.
> - 태그는대/소문자가구분되지않습니다.
> - 컨테이너파일시스템에서구성파일이 `cpx.conf` 파일로 `/etc` 디렉터리에있어야합니다.
> - 구성파일에는설명도포함할수있습니다. 설명앞에 “#” 문자를추가해야합니다.
> - 구성파일로 NetScaler CPX 컨테이너를배포하는도중오류시나리오가발생하면컨테이너의 `ns.log` 파일에오류가기록됩니다.
> - NetScaler CPX 컨테이너를재부팅하면컨테이너에서구성파일이다시적용됩니다.

```
#NetScaler Commands

add lb vserver v1 http 1.1.1.1 80

add service s1 2.2.2.2 http 80

bind lb vserver v1 s1

#Shell Commands

```



```
touch /etc/a.txt

echo ”this is a” > /etc/a.txt

#NetScaler Commands

add lb vserver v2 http

#Shell Commands

echo ”this is a 1” >> /etc/a.txt

#NetScaler Commands

add lb vserver v3 http

#This is a test configuration file
```

NetScaler CPX 컨테이너를설치하고구성파일을기반으로 NetScaler CPX 컨테이터를동적으로구성하려면 `docker run` 명령의 – v 옵션을사용하여정적구성파일을탑재하십시오.

```
docker run -dt --privileged=true -e EULA=yes --ulimit core=-1 -v /tmp/
    cpx.conf:/etc/cpx.conf --name mycpx store/citrix/netscalercpx
    :12.0-56.20
```

Docker 로깅드라이버구성

June 11, 2018

2018 년 3 월 21 일

Docker 에는 “로깅드라이버” 라는로깅메커니즘이포함되어실행중인컨테이너로부터정보를받을수있도록도와줍니다. NetScaler CPX 컨테이너를구성하여 Docker 로깅드라이버로생성된로그를전달할수있습니다. Docker 로깅드라이버에대한자세한내용은 https://docs.docker.com/config/containers/logging/configure/를참조하십시오.

NetScaler CPX 컨테이너에서생성한모든로그는기본적으로 Docker 호스트의 `/cpx/log/ns.log` 파일에저장됩니다. docker run 명령을사용하여 NetScaler CPX 컨테이너를시작할때, 생성된모든로그를 `--log-driver` 옵션을사용하여 Docker 로깅드라이버로전달하도록구성할수있습니다. 로깅드라이버가구성가능한매개변수인경우 `--log-opt \<NAME \>=\<VALUE\>` 옵션을사용하여이를설정할수있습니다.

다음예제에서는로깅드라이버로 syslog 를사용하여 NetScaler CPX 컨테이너가생성된모든로그를전달하도록구성되었습니다.


```
docker run -dt --privileged=true --log-driver syslog --log-opt syslog-
    address=udp://10.106.102.190:514 -e EULA=yes --ulimit core=-1 --name
     test store/citrix/netscalercpx:12.0-56.20
```

이와비슷하게다음예제에서는로깅드라이버로 Splunk 를사용하여 NetScaler CPX 컨테이너가생성된모든로그를전달하도록구성되었습니다.

```
docker run -dt --privileged=true --log-driver=splunk --log-opt splunk-
    token=176FCEBF-4CF5-4EDF-91BC-703796522D20 --log-opt splunk-url=
    https://splunkhost:8088 -e EULA=yes --ulimit core=-1 --name test
    store/citrix/netscalercpx:12.0-56.20
```

NetScaler CPX 인스턴스업그레이드

June 11, 2018

2016 년 6 월 30 일

NetScaler CPX 인스턴스를종료하고최신버전을동일한탑재지점에설치한후이전인스턴스를삭제함으로써 NetScaler CPX 인스턴스를업그레이드할수있습니다. 탑재지점은호스트에서 **/cpx** 디렉터리를탑재하는디렉터리입니다.

예를들어기존 NetScaler CPX 인스턴스의 **/cpx** 디렉터리를호스트의 **/var/cpx** 디렉터리에탑재한경우아래에나온것처럼탑재지점은 **/var/cpx** 이고 NetScaler CPX 탑재디렉터리는 **/cpx** 입니다.

```
    root@ubuntu:~# docker run -dt -e EULA=yes --name mycpx -v /var/cpx
       :/cpx --ulimit core=-1 cpx:11.1-48.xx
```

사전요구사항

다음을확인하십시오.

- 기존 NetScaler CPX 인스턴스의 /cpx 디렉터리를탑재한호스트디렉터리의세부정보. docker inspect \<containerName\> 명령을사용하여호스트디렉터리에대한정보를표시할수있습니다. 여기서 \<containerName\>은 NetScaler CPX 컨테이너의이름입니다.

  이명령출력에볼륨을비롯한컨테이너구성에대한세부정보가표시됩니다. “**Mounts(**탑재**)**“항목의 “**Source(**소스**)**” 하위항목에호스트내해당호스트디렉터리의위치가표시됩니다.

- https://www.microloadbalancer.com/get-it-now에서최신 NetScaler CPX Docker 이미지파일을다운로드한후 NetScaler CPX Docker 이미지를로드합니다. 이미지를로드하려면 Docker 이미지파일을저장한디렉터리로이동합니다. docker load -i \<image_name\> 명령을사용해이미지를로드합니다. NetScaler CPX 이미지를로드하면 Docker 이미지를명령을입력해이미지에대한정보를표시할수있습니다.


```
root@ubuntu:~# docker load -i  cpx-12.0-41.10.gz

root@ubuntu:~# docker images

REPOSITORY   TAG          IMAGE ID         CREATED     VIRTUAL SIZE

cpx          12.0-41.10   2e97aadf918b  43 hours ago      414.5 MB
```

NetScaler CPX 인스턴스를업그레이드하려면

1. `docker stop \<containerName>\` 명령을입력하여기존 NetScaler CPX 인스턴스를중지합니다. 여기서 `\<containerName\>`은 NetScaler CPX 인스턴스의이름입니다.

```
root@ubuntu:~# docker stop mycpx

mycpx
```

2. `docker run` 명령을사용하여호스트에로드한 NetScaler CPX 이미지에서최신 NetScaler CPX 인스턴스를배포합니다. 기존 NetScaler CPX 인스턴스에사용한것과동일한탑재지점 (예: `/var/cpx:/cpx`) 에인스턴스를배포해야합니다.

```
root@ubuntu:~# docker run -dt -P -e CPX_CORES=1 --name latestcpx --
   ulimit core=-1 -e EULA=yes -v /var/cpx:/cpx --cap-add=NET_ADMIN cpx
   :12.0-41.10
```

`docker ps` 명령을입력하여배포된 NetScaler CPX 인스턴스가최신버전인지확인할수있습니다.

```
root@ubuntu:~# docker ps

CONTAINER ID        IMAGE             COMMAND             CREATED
                STATUS                          PORTS
                     NAMES

ead12ec4e965   cpx:12.0-41.10  "/bin/sh -c 'bash -C "   5 seconds
   ago       Up 5 seconds        22/tcp, 80/tcp, 443/tcp, 161/udp
     latestcpx
```

3. 올바른 NetScaler CPX 인스턴스가배포된것을확인한후에는 **docker rm** <containerName> 명령을입력하여이전인스턴스를삭제합니다.

```
root@ubuntu:~# docker rm mycpx

mycpx
```


NetScaler CPX 인스턴스에서와일드카드가상서버사용

June 11, 2018

2016 년 6 월 30 일

NetScaler 인스턴스를프로비전할경우 Docker 엔진에의해사설 IP 주소하나 (단일 IP 주소) 가 NetScaler CPX 인스턴스에할당됩니다. NetScaler 인스턴스의세가지 IP 기능은한 IP 주소로멀티플렉싱됩니다. 이단일 IP 주소는서로다른포트번호를사용하여 NSIP, SNIP 및 VIP 기능을합니다.

Docker 엔진에의해할당되는단일 IP 주소는동적입니다. 단일 IP 주소를사용하거나 127.0.0.1 IP 주소를사용하여 LB(부하분산) 또는 CS(콘텐츠스위칭) 가상서버를추가하십시오. 127.0.0.1 을사용하여만든가상서버를와일드카드가상서버라고합니다. 기본적으로와일드카드가상서버를만들경우 NetScaler CPX 가와일드카드가상서버의할당된 IP 주소를바꿉니다. 할당된 IP 주소는 127.0.0.1 이며 Docker 엔진에의해 NetScaler CPX 인스턴스에할당된 NSIP 로바뀝니다.

고가용성 NetScaler CPX 배포에서 NetScaler CPX 인스턴스중하나에서와일드카드가상서버를추가하고해당인스턴스의 **ns.conf** 파일을배포의다른 NetScaler CPX 인스턴스에복사하여 Docker 엔진에의해할당된단일 IP 주소를 NetScaler 인스턴스에지정하는대신배포의모든 NetScaler CPX 인스턴스에서 NetScaler 구성이일관되게하고배포의모든 NetScaler CPX 인스턴스에서단일 IP 주소를기반으로 LB 또는 CS 가상서버를만들수있습니다.

참고사항**:**

- 와일드카드가상서버에할당하는포트번호가배포의다른가상서버에서사용되고있지않은지확인하십시오.
- 와일드카드가상서버는 * 문자를지원하지않습니다.

와일드카드부하분산가상서버를만들려면명령프롬프트에서다음명령을입력합니다.

```
add lb vserver <name> <serviceType> 127.0.0.1 <port>

add lb vserver testlbvserver HTTP 127.0.0.1 30000
```

와일드카드콘텐츠스위칭가상서버를만들려면명령프롬프트에서다음명령을입력합니다.

```
add cs vserver <name> <serviceType> 127.0.0.1 <port>

add cs vserver testcsvserver HTTP 127.0.0.1 30000
```

횡적트래픽흐름을사용할수있게하는프록시로 **NetScaler CPX** 배포

June 11, 2018

2016 년 5 월 13 일

이배포에서는 NetScaler CPX 인스턴스가다중호스트에상주하는응용프로그램컨테이너간에서통신을사용할수있게하는프록시로작동합니다. NetScaler CPX 인스턴스는다중호스트에서응용프로그램과함께프로비전되며통신을위한최단경로를제공합니다.

다음이미지에서는 NetScaler CPX 인스턴스를통한두응용프로그램간의트래픽흐름을보여줍니다.

이이미지는응용프로그램 C 와응용프로그램 B 간의트래픽흐름과응용프로그램 A 와응용프로그램 B 간의트래픽흐름을보여줍니다. 호스트중하나에서응용프로그램 C 가 B 로요청을보내면응용프로그램 C 와같은호스트에있는 NetScaler CPX 컨테이너에서먼저요청이수신됩니다. 그런다음 NetScaler CPX 컨테이너가응용프로그램 B 와같은호스트에서호스트되는 NetScaler CPX 컨테이너로트래픽을전달하고다시응용프로그램 B 로트래픽이전달됩니다. 응용프로그램 A 가응용프로그램 B 로요청을보낼때에도유사한트래픽경로를따릅니다.

이예제에서글로벌 VIP 를통해인터넷에서응용프로그램으로트래픽이전달될수있도록 NetScaler MPX 도배포됩니다.NetScaler MPX 의트래픽은 NetScaler CPX 컨테이너에서수신된후응용프로그램컨테이너간에분산됩니다.

다음다이어그램은통신이실행되도록설정해야하는구성과함께이토폴로지를보여줍니다.

다음표에는이예제구성에서 NetScaler CPX 인스턴스에구성된 IP 주소및포트가나와있습니다.

이예제시나리오를구성하려면 Docker 호스트 3 개모두에서 NetScaler CPX 컨테이너를만들때 Linux 셸프롬프트에서다음명령을실행합니다.

```
docker run -dt -p 22 -p 80 -p 161/udp -p 30000-30002: 30000-30002 --
    ulimit core=-1 --privileged=truecpx:6.2
```

NetScaler MAS 의 Jobs(작업) 기능을사용하거나 NITRO API 를사용하여다음명령을실행합니다.

Docker 호스트 1 의 NetScaler CPX 인스턴스:

```
    add lb vserver VIP-A1 HTTP 172.17.0.2 30000
    add service svc-A1 10.102.29.100 HTTP 80
    bind lb vserver VIP-A1 svc-A1
    add lb vserver VIP-B1 HTTP 172.17.0.2 30001
    add service svc-B1 10.102.29.100 HTTP 90
    bind lb vserver VIP-B1 svc-B1
    add lb vserver VIP-C1 HTTP 172.17.0.2 30002
    add service svc-VIP-C2 10.102.29.105 HTTP 30002
    add service svc-VIP-C3 10.102.29.110 HTTP 30002
    bind lb vserver VIP-C1 svc-VIP-C2
    bind lb vserver VIP-C1 svc-VIP-C3
```

Docker 호스트 2 의 NetScaler CPX 인스턴스:

```
    add lb vserver VIP-A2 HTTP 172.17.0.3 30000
    add service svc-A2 10.102.29.105 HTTP 80
    bind lb vserver VIP-A2 svc-A2
    add lb vserver VIP-B2 HTTP 172.17.0.3 30001
    add service svc-VIP-B1 10.102.29.100 HTTP 30001
```



```
6       bind lb vserver VIP-B2 svc-VIP-B1
7       add lb vserver VIP-C2 HTTP 172.17.0.3 30002
8       add service svc-C2 10.102.29.105 HTTP 70
9       bind lb vserver VIP-C2 svc-C2
```

Docker 호스트 3 의 NetScaler CPX 인스턴스:

```
1       add lb vserver VIP-A3 HTTP 172.17.0.4 30000
2       add service svc-VIP-A1 10.102.29.100 HTTP 30000
3       add service svc-VIP-A2 10.102.29.105 HTTP 30000
4       bind lb vserver VIP-A3 svc-VIP-A1
5       bind lb vserver VIP-A3 svc-VIP-A2
6       add lb vserver VIP-B3 HTTP 172.17.0.4 30001
7       add service svc-VIP-B1 10.102.29.100 HTTP 30001
8       bind lb vserver VIP-B3 svc-VIP-B1
9       add lb vserver VIP-C3 HTTP 172.17.0.4 30002
10      add service svc-C3 10.102.29.110 HTTP 70
11      bind lb vserver VIP-C3 svc-C3
```

단일호스트네트워크에 **NetScaler CPX** 배포

June 11, 2018

2016 년 5 월 13 일

단일호스트네트워크에서 NetScaler CPX 인스턴스는동일한호스트에있는응용프로그램컨테이너간의프록시로작동합니다. 이를통해 NetScaler CPX 인스턴스는컨테이너기반응용프로그램에확장성과보안성을제공합니다. 또한성능을최적화하는동시에원격분석데이터에대한심층정보를제공합니다.

단일호스트네트워크에서클라이언트, 서버및 NetScaler CPX 인스턴스는동일한 Docker 호스트에서컨테이너로배포됩니다. 모든컨테이너는 docker0 브리지를통해연결됩니다.

이환경에서 NetScaler CPX 인스턴스는동일한 Docker 호스트에서컨테이너로프로비전된응용프로그램의프록시로작동합니다.

다음그림에서는단일호스트토폴로지를보여줍니다.

이예제에서웹앱컨테이너 (172.17.0.2) 는클라이언트이고두데이터베이스컨테이너인 DB1(172.17.0.10) 과 DB2(172.17.0.11) 는서버입니다. NetScaler CPX 컨테이너 (172.17.0.4) 는클라이언트와서버사이에위치하며프록시로작동합니다.

웹응용프로그램이 NetScaler CPX 를통해데이터베이스컨테이너와통신할수있게하려면먼저 NetScaler CPX 컨테이너에서두서버를나타내는두서비스를구성해야합니다. 그런다음 NetScaler CPX IP 주소와비표준 HTTP 포트 (예: 81) 를사용하여가상서버를구성합니다. 비표준포트를사용하는이유는 NetScaler CPX 가표준 HTTP 포트 80 을 NITRO 통신용으로예약하기때문입니다.

이토폴로지에서는클라이언트와서버가동일한네트워크에있기때문에 NAT 규칙을구성할필요가없습니다.

이시나리오를구성하려면 NetScaler MAS 의 Jobs(작업) 기능을사용하거나 NITRO API 를사용하여다음명령을실행합니다.

```
add service db1 HTTP 172.17.0.10 80
add service db2 HTTP 172.17.0.11 80
add lb vserver cpx-vip HTTP 172.17.0.4 81
bind lb vserver cpx-vip db1
bind lb vserver cpx-vip db2
```

다중호스트네트워크에 **NetScaler CPX** 배포

June 11, 2018

2016 년 5 월 13 일

다중호스트네트워크에서한 NetScaler CPX 인스턴스가데이터센터의프로덕션배포로구성될수있으며, 부하분산기능을제공합니다. 또한모니터링기능과분석데이터를제공할수있습니다.

다중호스트네트워크에서는 NetScaler CPX 인스턴스, 백엔드서버및클라이언트가서로다른호스트에배포됩니다. NetScaler CPX 인스턴스가컨테이너기반응용프로그램및서버집합과물리적서버의부하를분산시키는프로덕션배포에서다중호스트토폴로지를사용할수있습니다.

이섹션에서는세가지다중호스트토폴로지에대해설명합니다.

- 토폴로지 1: NetScaler CPX 및서버는동일한호스트, 클라이언트는다른네트워크
- 토폴로지 2: NetScaler CPX 와물리적서버및클라이언트
- 토폴로지 3: 서로다른호스트에서프로비전된 NetScaler CPX 및서버

토폴로지 **1: NetScaler CPX** 및백엔드서버는동일한호스트**,** 클라이언트는다른네트워크

이토폴로지에서 NetScaler CPX 인스턴스및데이터베이스서버는동일한 Docker 호스트에서프로비전되지만클라이언트트래픽은네트워크의다른위치에서시작됩니다. NetScaler CPX 인스턴스가컨테이너기반응용프로그램또는서버집합의부하를분산하는프로덕션배포에서이토폴로지를사용할수있습니다.

다음다이어그램은이토폴로지를보여줍니다.

이예제에서 NetScaler CPX 인스턴스 (172.17.0.4) 와두서버 DB1(172.17.0.10) 및 DB2(172.17.0.11) 는 IP 주소가 10.102.29.100 인동일한 Docker 호스트에서프로비전됩니다. 클라이언트는네트워크의다른위치에상주합니다.

인터넷에서시작된클라이언트요청은 NetScaler CPX 인스턴스에구성되어있는 VIP 에서수신되고, VIP 가두서버간에요청을분산시킵니다.

두가지방법을사용하여이토폴로지를구성할수있습니다.

방법 **1:** VIP 에대해추가 IP 주소및표준포트사용

1. NetScaler CPX IP 주소대신추가 IP 주소를사용하여 NetScaler CPX 컨테이너에서 VIP 를구성합니다. 이렇게하면 컨테이너에서표준포트 80 을사용하여클라이언트요청을수신할수있습니다.
2. Docker 호스트에대해추가 IP 주소를구성합니다.
3. Docker 호스트의추가 IP 주소에서수신된모든트래픽을 VIP 의추가 IP 주소로전달하는 NAT 규칙을구성합니다.
4. NetScaler CPX 인스턴스에서두서버를서비스로구성합니다.
5. 마지막으로, 서비스를 VIP 에바인딩합니다.

이예제구성에서 10.x.x.x 네트워크는공용네트워크를나타냅니다.

이예제시나리오를구성하려면 NetScaler MAS 의 Jobs(작업) 기능을사용하거나 NITRO API 를사용하여다음명령을실행합니다.

```
add service s1 172.17.0.10 HTTP 80
add service s2 172.17.0.11 HTTP 80
add lb vserver cpx-vip HTTP 172.17.4.100 80
bind lb vserver cpx-vip s1
bind lb vserver cpx-vip s2
```

Linux 셸프롬프트에서다음명령을실행하여 Docker 호스트에대한추가공용 IP 주소와 NAT 규칙을구성합니다.

```
ip addr add 10.102.29.103/24 dev eth0
iptables -t nat -A PREROUTING -p ip  -d 10.102.29.103 -j DNAT --to-
   destination 172.17.4.100
```

방법 **2:** VIP 에대해 NetScaler CPX IP 주소사용및포트매핑구성:

1. NetScaler CPX 인스턴스에서 VIP 와두서비스를구성합니다. VIP 와함께비표준포트 81 을사용합니다.
2. 서비스를 VIP 에바인딩합니다.
3. Docker 호스트의포트 50000 에서수신된모든트래픽을 VIP 및포트 81 로전달하는 NAT 규칙을구성합니다.

이예제시나리오를구성하려면 Docker 호스트 3 개모두에서 NetScaler CPX 컨테이너를만들때 Linux 셸프롬프트에서다음명령을실행합니다.

```
docker run -dt -p 22 -p 80 -p 161/udp -p 50000:81 --ulimit core=-1
   --privileged=true cpx:6.2
```

NetScaler CPX 인스턴스를프로비전한후 NetScaler MAS 의 Jobs(작업) 기능을사용하거나 NITRO API 를사용하여다음명령을실행합니다.

```
add service s1 172.17.0.10 http 80
add service s2 172.17.0.11 http 80
add lb vserver cpx-vip HTTP 172.17.0.4 81
bind lb vserver cpx-vip s1
bind lb vserver cpx-vip s2
```


> 참고:
>
> NetScaler CPX 인스턴스를프로비전할때포트매핑을구성하지않았다면 Linux 셸프롬프트에서다음명령을실행하여 NAT 규칙을구성합니다.
>
> **iptables -t nat -A PREROUTING -p tcp -m addrtype –dst-type LOCAL -m tcp –dport** 50000 -j DNAT - 대상 172.17.0.4:81

토폴로지 **2: NetScaler CPX** 와물리적서버및클라이언트

이토폴로지에서는 Docker 호스트에서 NetScaler CPX 인스턴스만프로비전됩니다. 클라이언트및서버는컨테이너기반이아니며네트워크의다른위치에상주합니다.

이환경에서물리적서버간에부하를분산하도록 NetScaler CPX 인스턴스를구성할수있습니다.

다음그림에서는이토폴로지를보여줍니다.

이예제에서 NetScaler CPX 컨테이너 (172.17.0.4) 는클라이언트와물리적서버사이에위치하며프록시로작동합니다. 서버 DB1(10.102.29.105) 및 DB2(10.102.29.110) 는네트워크에서 Docker 호스트외부에상주합니다. 클라이언트요청은인터넷에서시작되고 NetScaler CPX 에서수신되어두서버간에분산됩니다.

클라이언트와서버사이의통신이 NetScaler CPX 를통해이루어질수있도록 NetScaler CPX 컨테이너를만들때먼저포트매핑을구성해야합니다. 그런다음 NetScaler CPX 컨테이너에서두서버를나타내는두서비스를구성합니다. 마지막으로, NetScaler CPX IP 주소와 HTTP 포트 8080 에매핑된비표준포트를사용하여가상서버를구성합니다.

예제구성에서 10.x.x.x 네트워크는공용네트워크를나타냅니다.

이예제시나리오를구성하려면 NetScaler CPX 컨테이너를만들때 Linux 셸프롬프트에서다음명령을실행합니다.

```
1     docker run -dt -p 22 -p 80 -p 161/udp -p 8080:8080 --ulimit core=-1
          --privileged=true cpx:6.2
```

그런다음 NetScaler MAS 의 Jobs(작업) 기능을사용하거나 NITRO API 를사용하여다음명령을실행합니다.

```
1     add service s1 HTTP 10.102.29.105 80
2     add service s2 HTTP 10.102.29.110 80
3     add lb vserver cpx-vip HTTP 172.17.0.4 8080
4     bind lb vserver cpx-vip s1
5     bind lb vserver cpx-vip s2
```

토폴로지 **3:** 서로다른호스트에서프로비전된 **NetScaler CPX** 및서버

이토폴로지에서 NetScaler CPX 인스턴스및데이터베이스서버는서로다른 Docker 호스트에서프로비전되고클라이언트트래픽은인터넷에서시작됩니다. NetScaler CPX 인스턴스가컨테이너기반응용프로그램또는서버집합의부하를분산하는프로덕션배포에서이토폴로지를사용할수있습니다.

다음다이어그램은이토폴로지를보여줍니다.

이예제에서 NetScaler CPX 인스턴스및서버 (DB1) 는 IP 주소가 10.102.29.100 인동일한 Docker 호스트에서프로비전됩니다. 다른네서버 (DB2, DB3, DB4 및 DB5) 는서로다른두 Docker 호스트 10.102.29.105 및 10.102.29.110 에서프로비전됩니다.

인터넷에서시작된클라이언트요청은 NetScaler CPX 인스턴스에서수신되고, 이인스턴스가다섯개서버간에요청을분산시킵니다. 이통신을사용하도록설정하려면다음을구성해야합니다.

1. NetScaler CPX 컨테이너를만들때포트매핑을설정합니다. 이예제에서는컨테이너의포트 8080 을호스트의포트 8080 으로전달해야한다는의미입니다. 클라이언트요청이호스트의포트 8080 에도달하면 CPX 컨테이너의포트 8080 으로매핑됩니다.
2. NetScaler CPX 인스턴스에서다섯개서버를서비스로구성합니다. 이러한서비스를설정하려면관련 Docker 호스트 IP 주소와매핑된포트의조합을사용해야합니다.
3. NetScaler CPX 인스턴스에서클라이언트요청을수신하는 VIP 를구성합니다. 이 VIP 는 NetScaler CPX IP 주소와호스트의포트 8080 에매핑된포트 8080 으로나타나야합니다.
4. 마지막으로, 서비스를 VIP 에바인딩합니다.

예제구성에서 10.x.x.x 네트워크는공용네트워크를나타냅니다.

이예제시나리오를구성하려면 NetScaler CPX 컨테이너를만들때 Linux 셸프롬프트에서다음명령을실행합니다.

```
docker run -dt -p 22 -p 80 -p 161/udp -p 8080:8080 --ulimit core=-1
    --privileged=true cpx:6.2
```

NetScaler MAS 의 Jobs(작업) 기능을사용하거나 NITRO API 를사용하여다음명령을실행합니다.

```
add service s1 10.102.29.100 HTTP 8081
add service s2 10.102.29.105 HTTP 8081
add service s3 10.102.29.105 HTTP 8082
add service s4 10.102.29.110 HTTP 8081
add service s5 10.102.29.110 HTTP 8082
add lb vserver cpx-vip HTTP 172.17.0.2 8080
bind lb vserver cpx-vip s1
bind lb vserver cpx-vip s2
bind lb vserver cpx-vip s3
bind lb vserver cpx-vip s4
bind lb vserver cpx-vip s5
```

네트워크에직접액세스하도록 **NetScaler CPX** 배포

June 11, 2018

2016 년 5 월 13 일

NetScaler CPX 인스턴스가네트워크에직접액세스하도록구성할수있습니다. 이시나리오에서들어오는트래픽은 NetScaler CPX VIP 에서직접수신됩니다.

이통신을사용하도록설정하려면먼저 docker0 브리지에서공용 IP 주소를구성해야합니다. 그런다음네트워크포트 eth0 에서공용 IP 주소를제거하고해당네트워크포트를 docker0 브리지에바인딩합니다.

두서비스를추가하여부하분산을구성한다음 NetScaler CPX 인스턴스에서네트워크공용 IP 주소를 VIP 로구성합니다. 클라이언트요청이 VIP 에서직접수신됩니다.

예제구성에서 10.x.x.x 네트워크는공용네트워크를나타냅니다.

이시나리오를구성하려면 Linux 셸프롬프트에서다음명령을실행합니다.

```
ip addr add 10.102.29.100/24 dev docker0; \
ip addr del 10.102.29.100/24 dev eth0; \
brctl addif docker0 eth0; \
ip route del default; \
ip route add default via 10.102.29.1 dev docker0
```

NetScaler MAS 의 Jobs(작업) 기능을사용하거나 NITRO API 를사용하여다음명령을실행합니다.

```
add service s1 172.17.0.8 http 80
add service s2 172.17.0.9 http 80
add lb vserver cpx-vip HTTP 10.102.29.102 80
bind lb vserver cpx-vip s1
bind lb vserver cpx-vip s2
```

Mesos 및 Marathon 환경에 NetScaler CPX 배포

June 11, 2018

2016 년 6 월 30 일

이배포에서는 Mesos 및 Marathon 환경을사용하여응용프로그램을시작하고규모를확장하거나축소할수있습니다. Mesos 를사용하면분산응용프로그램또는프레임워크에서리소스를격리하고공유할수있습니다. Marathon 은응용프로그램오케스트레이션프레임워크로, 이를통해응용프로그램을시작하고규모를확장하거나축소할수있습니다. Mesos 및 Marathon 에관한자세한내용은 https://docs.mesosphere.com/gettingstarted/overview/을참조하십시오.

Mesos 및 Marathon 환경에 NetScaler CPX 를배포하려면다음작업을완료해야합니다.

1. Mesos 및 Marathon 마스터-슬레이브클러스터를설정합니다. 모든응용프로그램및 NetScaler CPX 인스턴스를 Ubuntu 호스트에서구성해야하며이러한호스트를 Mesos 슬레이브로구성해야합니다. Ubuntu 호스트를 Mesos

마스터로구성하고 Mesos 마스터에 Marathon 을설치해야합니다. 자세한내용은 https://open.mesosphere.com/getting-started/install/을참조하십시오.

2. 모든 Mesos 슬레이브에서응용프로그램및 NetScaler CPX 인스턴스의이미지를저장해야합니다.
3. 그런다음 Marathon CLI 또는 Marathon GUI 를사용하여응용프로그램및 NetScaler CPX 인스턴스를시작할수있습니다. Marathon 에서는 NetScaler CPX 인스턴스를응용프로그램으로시작합니다. Marathon 에서응용프로그램을실행하는것에관한자세한내용은 https://mesosphere.github.io/marathon/docs/application-basics.html을참조하십시오.

다음그림에서는 Mesos 및 Marathon 환경의 NetScaler CPX 배포를보여줍니다.

이예제에서 UH1 은 Mesos 마스터로구성된 Ubuntu 호스트입니다. Ubuntu 호스트 UH2, UH3 및 UH4 는 Mesos 슬레이브로구성되었습니다. Marathon 은 Mesos 마스터에설치되었습니다. App1 및 App2 는 NetScaler CPX 로부하를분산할응용프로그램입니다. 응용프로그램을시작하기위해서 Mesos 마스터가 Mesos 슬레이브및기타리소스를응용프로그램에할당하고 Marathon 이할당된 Mesos 슬레이브에서응용프로그램을시작합니다. 이예제에서 CPX, App1 및 App2 는각각 Ubuntu 호스트 UH2, UH3 및 UH4 에서시작됩니다.

Marathon CLI 또는 Marathon GUI 를사용하여응용프로그램및 NetScaler CPX 인스턴스를시작할수있습니다.

Marathon CLI 를사용하여응용프로그램및 **NetScaler CPX** 인스턴스시작

응용프로그램또는 NetScaler CPX 인스턴스를시작하려면 JSON 스크립트를작성해야합니다. JSON 스크립트에는 ID, 인스턴스수, 컨테이너유형, 응용프로그램이미지파일이름, 포트매핑, 네트워크, 레이블및상태확인사양같은세부정보가포함되어야합니다. 그런다음 Mesos 마스터에서 JSON 스크립트를실행하여응용프로그램및 NetScaler CPX 인스턴스를시작해야합니다.

응용프로그램및 **NetScaler CPX** 인스턴스를시작하려면**:**

1. 시작하려는각응용프로그램및 NetScaler CPX 인스턴스에대한 JSON 스크립트를작성합니다.

   예를들어응용프로그램을시작하려면다음샘플스크립트에표시된것처럼 JSON 스크립트를작성합니다.

```
{

  "id": "web-backend",
  "cpus": 0.1,
  "mem": 100.0,
  "instances": 2,
  "container": {

    "type": "DOCKER",
    "docker": {

      "image": "nginx_backend:latest",
      "forcePullImage": false,
      "network": "BRIDGE",
      "portMappings": [
```



```
16              {
17    "containerPort": 80, "hostPort": 0, "servicePort": 20002, "protocol":
          "tcp"  }
18
19             ]
20           }
21
22         }
23   ,
24         "labels": {
25
26                 "NETSCALER_GROUP":"BACKEND"
27         }
28   ,
29         "healthChecks": [
30           {
31
32             "protocol": "HTTP",
33             "portIndex": 0,
34             "path": "/",
35             "gracePeriodSeconds": 5,
36             "intervalSeconds": 20,
37             "maxConsecutiveFailures": 3
38           }
39
40         ]
41       }
```

> 중요:
>
> 응용프로그램의경우레이블을 NETSCALER_CPX 로설정하지마십시오. 응용프로그램의경우레이블매개변수를 NETSCALER_CPX 로설정하면 NetScaler CPX 인스턴스에서응용프로그램을구성할수없습니다.

예를들어 NetScaler CPX 인스턴스를시작하려면다음샘플스크립트에표시된것처럼 JSON 스크립트를작성합니다.

```
1      {
2
3        "id": "cpx",
4        "cpus": 0.25,
5        "mem": 512,
6        "instances": 2,
7        "container": {
8
9          "type": "DOCKER",
10         "docker": {
11
```



```
12            "image": "cpx:10.5-53.535",
13            "network": "HOST",
14            "privileged": true,
15            "parameters": [
16              {
17   "key": "tty", "value": "true"  }
18  ,
19              {
20   "key": "env", "value": "NS_NETMODE=HOST"  }
21  ,
22              {
23   "key": "env", "value": "NETSCALER_GROUP=BACKEND"  }
24  ,
25              {
26   "key": "env", "value": "marathon_url=http://10.102.103.222:8080"  }
27
28            ]
29           }
30
31         }
32  ,
33       "labels": {
34
35                "NETSCALER_AS_APP": "true"
36         }
37
38
39       }
```

> 참고:
>
> Mesos 및 Marathon 환경에서는 NetScaler CPX 인스턴스를호스트네트워킹모드에서만실행하도록지원합니다. 따라서 JSON 스크립트에서네트워크매개변수의값을 HOST로설정해야합니다.

2. Mesos 마스터에서다음명령을실행하여 JSON 스크립트를실행합니다.

```
curl -X POST http://<Marathon_IP_Address>:<Marathon port>/v2/apps -d @<
    JSON_Script_Name>.json -H "Content-type: application/json"
```

또는다음 docker run 명령을사용하여 NetScaler CPX 인스턴스를시작할수있습니다.

```
docker run -dt --privileged=true --net=host -e NS_NETMODE="HOST" --
    ulimit core=-1 -e marathon_url= "http://<Marathon_IP_Address>:<
    Marathon port>" cpx:7
```

Marathon 에서인증을요구하는경우다음 docker run 명령을사용할수있습니다.

```
docker run -dt --privileged=true --net=host -e NS_NETMODE="HOST" --
    ulimit core=-1 marathon_url= "http://<Marathon_IP_Address>:<Marathon
     port>"  - e marathon_user=abcd  - e marathon_password=secret cpx:7
```

Marathon GUI 를사용하여응용프로그램및 **NetScaler CPX** 인스턴스시작

브라우저에 Marathon IP 주소및포트를입력하여 Marathon GUI 에액세스할수있습니다. 기본적으로 Marathon 포트는 8080 입니다.

응용프로그램및 **NetScaler CPX** 인스턴스를시작하려면**:**

1. Marathon GUI 에로그온합니다.
2. **Applications(**응용프로그램**)** 탭의왼쪽위창에서 **Create(**만들기**)** 를클릭합니다.
3. **New Application(**새응용프로그램**)** 화면의 **Docker container settings(Docker** 컨테이너설정**)**, **Environment variables(**환경변수**)**, **Labels(**레이블**)**, **Health checks(**상태확인**)** 및 **Optional settings(**옵션설정**)** 섹션에서매개변수를지정합니다.

> 참고**:**
>
> Mesos 및 Marathon 환경에서는 NetScaler CPX 인스턴스를호스트네트워킹모드에서만실행하도록지원합니다. 따라서 NetScaler CPX 인스턴스를시작하려면 Docker container settings(Docker 컨테이너설정) 섹션의 Network(네트워크) 목록에서 Host(호스트) 를선택합니다.

4. **+ Create(**만들기**)** 를클릭합니다.
5. 시작한응용프로그램이 **Applications(**응용프로그램**)** 아래에표시됩니다.

NetScaler MAS 를사용하여 **NetScaler CPX** 인스턴스와 **Mesos, Marathon, InfoBlox** 및 **Nuage** 통합

June 11, 2018

2016 년 6 월 30 일

네트워크관리자는 Marathon 스케줄러를사용하여 Mesos 클러스터에서응용프로그램을배포하고여러 NetScaler CPX 인스턴스를프로비전한다음응용프로그램에대한트래픽부하를분산할수있습니다. 배포환경에서 Nuage VSP 솔루션을사용하여각응용프로그램및 NetScaler CPX 인스턴스에네트워크연결을제공할수있으며 InfoBlox 솔루션을사용하여배포환경에 DNS 서비스를설정할수있습니다.

NetScaler MAS 를사용하여 NetScaler CPX 인스턴스및응용프로그램을관리하고, 모니터링하고, 시각화할수있습니다.

이배포환경에서다음작업을수행합니다.

1. Mesos 및 Marathon 마스터-슬레이브클러스터를설정합니다. 모든응용프로그램및 NetScaler CPX 인스턴스를 Ubuntu 호스트에서구성해야하며이러한호스트를 Mesos 슬레이브로구성해야합니다. Ubuntu 호스트를 Mesos 마스터로구성하고 Mesos 마스터에 Marathon 을설치해야합니다. 자세한내용은 https://open.mesosphere.com/getting-started/install/을참조하십시오.
2. 모든 Mesos 슬레이브에서응용프로그램및 NetScaler CPX 인스턴스의이미지를저장해야합니다.
3. 그런다음 Marathon CLI 또는 Marathon GUI 를사용하여응용프로그램및 NetScaler CPX 인스턴스를시작할 수있습니다. Marathon 에서는 NetScaler CPX 인스턴스를응용프로그램으로시작합니다. Marathon 에서응용프로그램을실행하는것에관한자세한내용은 https://mesosphere.github.io/marathon/docs/application-basics.html을참조하십시오.
4. Nuage VSP 솔루션에서 VIP 서브넷을설정합니다. VIP 서브넷은장래에모든 Mesos 응용프로그램에필요한 VIP 요구사항을충족할수있을만큼충분히커야합니다. VIP 서브넷은어떠한 Mesos 응용프로그램에서도사용되지않아야하며 VIP 서브넷에서 Nuage 끝점 (vport) 을만들지않아야합니다.
5. InfoBlox 솔루션에서 DNS 확인을위한 DNS 서버를설정합니다.
6. NetScaler MAS 에응용프로그램, Marathon 스케줄러, Nuage 및 InfoBlox 세부정보를등록합니다.
7. NetScaler CPX 인스턴스를 NetScaler MAS 에추가합니다.

참고사항**:**

1. NetScaler CPX IP 주소 (관리 IP 주소) 는 Mesos 클러스터내에서완전히라우팅가능해야합니다.
2. Mesos 슬레이브에서 NetScaler CPX 인스턴스를프로비전할때다음과같은세부정보를지정해야합니다.
   - NetScaler MAS 호스트이름또는 IP 주소
   - NetScaler MAS 관리자의사용자이름및암호
   - NetScaler CPX 인스턴스에서 DNS 확인용으로구성할 DNS 서버이름
3. 응용프로그램과 Nuage 서브넷사이에일대일매핑관계가있어야합니다.
4. NetScaler CPX 는 Mesos 슬레이브에서실행되고있는 Nuage Docker 모니터를사용하여네트워크에연결되어야합니다.
5. 연결된서브넷에 Marathon 응용프로그램작업 (응용프로그램의 VIP 주소에서 IP: 포트구성원) 만나타나야합니다.
6. Marathon 레이블중하나에 Marathon 응용프로그램포트를지정해야합니다.

Marathon GUI 를사용하여응용프로그램및 **NetScaler CPX** 인스턴스시작

브라우저에 Marathon IP 주소및포트를입력하여 Marathon GUI 에액세스할수있습니다. 기본적으로 Marathon 포트는 8080 입니다.

응용프로그램및 **NetScaler CPX** 인스턴스를시작하려면**:**

1. Marathon GUI 에로그온합니다.

2. **Applications(**응용프로그램**)** 탭의왼쪽위창에서 **Create(**만들기**)** 를클릭합니다.

3. **New Application(**새응용프로그램**)** 화면의 **Docker container settings(Docker** 컨테이너설정**)**, **Environment variables(**환경변수**)**, **Labels(**레이블**)**, **Health checks(**상태확인**)** 및 **Optional settings(**옵션설정**)** 섹션에서매개변수를지정합니다.

> 참고
>
> Mesos 및 Marathon 환경에서는 NetScaler CPX 인스턴스를호스트네트워킹모드에서만실행하도록지원합니다. 따라서 NetScaler CPX 인스턴스를시작하려면 **Docker container settings(Docker** 컨테이너설정**)** 섹션의 **Network(**네트워크**)** 목록에서 **Host(**호스트**)** 를선택합니다.

4. **+ Create(**만들기**)** 를클릭합니다.

5. 시작한응용프로그램이 **Applications(**응용프로그램**)** 아래에표시됩니다.

NetScaler MAS 에응용프로그램**, Marathon** 스케줄러**, Nuage** 및 **InfoBlox** 세부정보등록

NetScaler CPX 인스턴스를성공적으로배포하려면응용프로그램, Marathon 스케줄러, Nuage 및 InfoBlox 솔루션의세부정보를 NetScaler MAS 에등록해야합니다.

NetScaler CPX 인스턴스가 NetScaler MAS 연결을열수있도록네트워크를구성해야합니다. NetScaler CPX 컨테이너가시작되고 Nuage 환경에서관리 IP 주소를얻은후보안연결을사용하여 NetScaler MAS 에등록요청을보냅니다.

등록프로세스의일부로 NetScaler MAS 는 NetScaler CPX 인스턴스에대한정보와 NITRO REST API 를사용한구성을위해인스턴스에연결할때사용할 IP 주소또는포트를파악합니다. 그런다음 NetScaler MAS 는상태모니터링을시작합니다.

NetScaler MAS 에세부정보를등록하려면

1. NetScaler MAS 에로그온합니다.
2. **Orchestration(**오케스트레이션**) > Container Orchestration(**컨테이너오케스트레이션**) > Mesos Configuration(Mesos** 구성**)** 으로이동한다음 **Add(**추가**)** 를클릭합니다.
3. **Application Settings(**응용프로그램설정**)** 에서다음매개변수를지정합니다.
   - **App Default Domain Suffix(**응용프로그램기본도메인접미사**)**. InfoBlox 에서구성되는응용프로그램의 DNS 이름을만드는데사용되는도메인접미사입니다.
   - **VIP Subnet Name(VIP** 서브넷이름**)**. NetScaler MAS 에서응용프로그램의 VIP 를할당하는데사용되는 Nuage 서브넷의이름입니다. 사전에 Nuage 시스템에서이서브넷을만든다음 NetScaler MAS 에서브넷을등록해야합니다.
4. **Marathon Scheduler Details(Marathon** 스케줄러세부정보**)** 에서 Marathon URL, 사용자이름및암호를지정합니다.
5. **Nuage Details(Nuage** 세부정보**)** 에서 VSD URL, 사용자이름, 암호및 VSD 엔터프라이즈 ID 를지정합니다.
6. **InfoBlox** 에서 InfoBlox URL, 사용자이름및암호를지정합니다.
7. **/etc/resolv.conf** 파일에 InfoBlox DNS IP 주소와도메인검색경로를추가합니다.
   참고**:** NetScaler Management and Analytics System 을다시시작할때 **/etc/resolv.conf** 파일에서 InfoBlox DNS IP 주소및도메인검색경로를유지하려면 **/mpsconfig/svm.conf** 파일에서다음항목을업데이트합니다.
   /mps/changenameserver.sh <DNS IP address1> <DNS IP address2>
   echo “search <domain name>” » /etc/resolv.conf
8. JSON 스크립트파일을사용하여 NetScaler CPX 인스턴스를시작합니다. 다음은샘플 JSON 스크립트파일입니다.
   참고**:** JSON 스크립트파일에다음줄이포함되어있는지확인하십시오.


```
"cmd": "cd /var/netscaler/bins/ ; sed -i 's/\## Creating NSPPE startup
 conf, Read By PE/\## Creating NSPPE startup conf, Read By PE\\\\\
necho \\"$NSIP $HOSTNAME\\"\>\> \\\\\\\\/etc\\\\\\\\/hosts/' ./docker\\
_startup.sh ; bash -C ./docker\\_startup.sh ; bash",
```

샘플 **JSON** 스크립트**:**

```
curl -X POST http://10.xx.xx.62:8080/v2/apps -d
   @cpx_nuage_custom_etchostcorrection.json -H "Content-type:
   application/json"

{

      "id": "cpx-host3",
      "cpus": 1,
      "mem": 1024,
      "instances": 2,
      "cmd" : "cd /var/netscaler/bins/ ; sed -i 's/# Creating NSPPE
     startup conf, Read By PE/# Creating NSPPE startup conf, Read By
     PE\necho "$NSIP $HOSTNAME" >> \/etc\/hosts/' ./docker_startup.
    sh ; bash -C ./docker_startup.sh ; bash",
      "constraints": [["hostname", "UNIQUE"]],
      "container": {

           "type": "DOCKER",
           "docker": {

                     "image":  "cpx:11.1.40.3",
                     "network": "NONE",
                     "privileged": true,
                     "parameters": [
                         {
 "key": "tty", "value": "true"  }

                     ]
                }

       }
,
      "env": {

       "NUAGE-ENTERPRISE": "<nuage_enterprise>",
       "NUAGE-DOMAIN" : "<nuage_domain>",
       "NUAGE-ZONE":"<nuage_zone>",
       "NUAGE-NETWORK":"<nuage_network>",
```

```
34              "NUAGE-USER":"<nuage_user>",
35          "NS_MGMT_SERVER":"<NMAS_server_IP>"
36              }
37  ,
38             "labels": {
39
40                          "NETSCALER_AS_APP": "true"
41              }
42
43      }
```

여기서:

- <nuage_enterprise> 는 Nuage 엔터프라이즈이름입니다.
- <nuage_domain> 은 Nuage 도메인이름입니다.
- <nuage_zone> 은 Nuage 영역이름입니다.
- <nuage_network> 는 Nuage 네트워크세부정보입니다.
- <nuage_user> 는 Nuage 관리사용자이름입니다.
- <NMAS_server_IP> 는 NetScaler MAS 서버 IP 주소입니다.

NetScaler CPX 를사용하여 Kubernetes 환경에서횡적트래픽부하분산

June 11, 2018

2017 년 10 월 5 일

NetScaler CPX 를 Kubernetes 클러스터에배포하여클러스터에서컨테이너화된응용프로그램의부하를분산할수있습니다. NetScaler CPX 는다음 Kubernetes 버전에서지원됩니다.

- 1.5.x
- 1.6.x

Kubernetes 에대한자세한내용은 http://kubernetes.io/docs/을참조하십시오.

기본적으로, Kubernetes 클러스터에 NetScaler CPX 를배포하면이는기본적인부하분산기능을제공하는 Kubernetes 의 kube-proxy 를대체합니다. kube-proxy 를 NetScaler CPX 로교체하면부하분산기능외에도 NetScaler MAS(Management and Analytics System) 를사용해다음과같은이점을얻을수있습니다.

- 클러스터에서응용프로그램환경에대한가시성확보
- 클러스터에서 NetScaler CPX 인스턴스관리및모니터링
- StyleBook 기능을사용하여응용프로그램의복잡한 NetScaler 구성을쉽게관리

NetScaler MAS 에대한자세한내용은 NetScaler Management and Analytics System 제품설명서를참조하십시오.

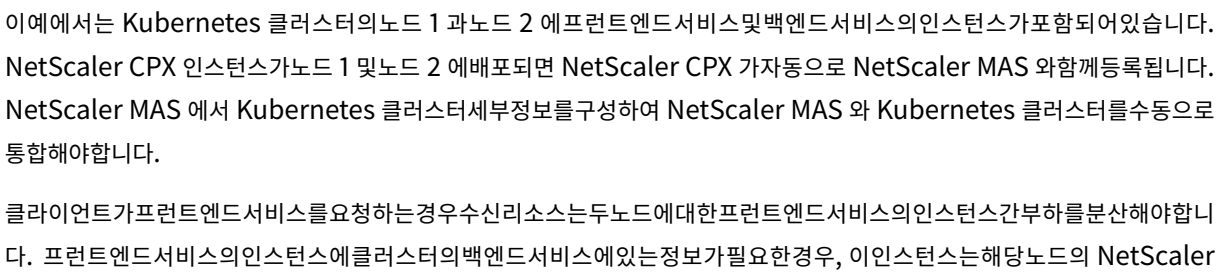
NetScaler CPX 가 Kubernetes 환경에서횡적트래픽흐름의부하를분산하는방법

Kubernetes 클러스터를배포한후에는 NetScaler MAS 에서 Kubernetes 환경세부정보를제공해 NetScaler MAS 와클러스터를통합해야합니다. NetScaler MAS 는서비스, 끝점, 수신규칙등 Kubernetes 리소스의변경사항을모니터링합니다.

Kubernetes 클러스터에 NetScaler CPX 인스턴스를배포하면 NetScaler CPX 인스턴스가자동으로 NetScaler MAS 에등록됩니다. 등록프로세스의일부로 NetScaler MAS 는 NetScaler CPX 인스턴스 IP 주소에대한정보와 NITRO REST API 를사용하여구성하기위해인스턴스에연결할포트를파악합니다.

NetScaler MAS 의 Stylebook 엔진은 NetScaler MAS 가서비스, 끝점, 수신규칙등 Kubernetes 에서수집하는모든정보를처리합니다. Stylebook 엔진은기존에프로비저닝된 Stylebook(`com.citrix.adc.stylebooks/1.0/cs-lb-mon`) 을사용하여부하분산에필요한가상서버및서비스그룹과같은 NetScaler 관련구성을생성하고 NetScaler CPX 인스턴스에해당구성을적용합니다. StyleBook 에대한자세한내용은 Stylebook을참조하십시오.

다음그림은 NetScaler CPX 가 Kubernetes 클러스터에서횡적트래픽흐름의부하를분산하는방법을보여줍니다.



이예에서는 Kubernetes 클러스터의노드 1 과노드 2 에프런트엔드서비스및백엔드서비스의인스턴스가포함되어있습니다. NetScaler CPX 인스턴스가노드 1 및노드 2 에배포되면 NetScaler CPX 가자동으로 NetScaler MAS 와함께등록됩니다. NetScaler MAS 에서 Kubernetes 클러스터세부정보를구성하여 NetScaler MAS 와 Kubernetes 클러스터를수동으로통합해야합니다.

클라이언트가프런트엔드서비스를요청하는경우수신리소스는두노드에대한프런트엔드서비스의인스턴스간부하를분산해야합니다. 프런트엔드서비스의인스턴스에클러스터의백엔드서비스에있는정보가필요한경우, 이인스턴스는해당노드의 NetScaler CPX 인스턴스에요청을보냅니다. 해당 NetScaler CPX 인스턴스는클러스터의백엔드서비스간요청의부하를분산함으로써횡적트래픽흐름을제공합니다.

Kubernetes 클러스터의노드에 NetScaler CPX 인스턴스배포

NetScaler CPX 인스턴스를 Kubernetes 클러스터의노드에 Kubernetes 포드로배포할수있습니다. NetScaler CPX 인스턴스는디먼세트또는매니페스트로배포될수있습니다.

- 디먼세트 – NetScaler CPX 인스턴스를디먼세트리소스로배포하면 NetScaler CPX 인스턴스를해당노드의포드로배포할수있고 NetScaler CPX 인스턴스가 Kubernetes 클러스터와결합된새노드에배포되도록보장할수도있습니다. 새노드가클러스터와결합하면디먼세트에지정된 NetScaler CPX 인스턴스가자동으로노드에설치됩니다.
- 매니페스트 – Kubernetes 매니페스트는 Kubernetes 개체배포와구성지침을포함한 YAML 또는 JSON 형식의파일입니다. NetScaler CPX 인스턴스의 Kubernetes 매니페스트를생성하고이를노드의특정디렉터리에배치할수있습니다. 각노드의 *kubelet* 이이디렉터리를모니터링하고매니페스터에지정된대로 NetScaler CPX 인스턴스인개체를만듭니다.

사전요구사항

이러한유형의배포에서는다음을확인하십시오.

- Kubernetes 클러스터를만듭니다. 자세한내용은 http://kubernetes.io/docs/를참조하십시오.

  > 참고:
  >
  > NetScaler CPX 는 Kubernetes v1.7 을지원합니다.

- 최신버전의 NetScaler CPX 를 Docker Store에서받아사용하십시오.
- NetScaler MAS 를 배포합니다. 지침은 http://docs.citrix.com/ko-kr/netscaler-mas/11-1/single-server-deployment.html을참조하십시오.
- NetScaler MAS 에 Kubernetes 클러스터를구성합니다. 지침은Kubernetes 환경의수신컨트롤러로 NetScaler MAS 사용을참조하십시오.
- NetScaler MAS 서버의지문을확인합니다. 지문을얻으려면다음을수행하십시오.

  1. NetScaler MAS 에로그온합니다.
  2. **System(**시스템**) > System Administration(**시스템관리**)** 으로이동하고 **View SSL Certificate(SSL** 인증서보기**)** 을클릭합니다.
  3. **Certificate Details(**인증서세부정보**)** 페이지의 **Fingerprint(**지문**)** 필드에서 NetScaler MAS 서버의지문을복사합니다.

NetScaler CPX 인스턴스를디먼세트로배포

디먼세트접근방법을통해 NetScaler CPX 인스턴스를노드의포드로배포할수있습니다. 그러면자동으로이인스턴스가 Kubernetes 클러스터를결합한각새노드에서포드로배포됩니다.

NetScaler CPX 인스턴스를디먼세트로배포하려면 YAML 파일또는 JSON 스크립트를작성해야합니다. 이파일또는스크립트는컨테이너유형, CPX 이미지파일이름, NetScaler MAS 서버 IP 주소및 NetScaler MAS 서버지문을지정합니다.

다음은샘플 YAML 파일입니다.

```
apiVersion: extensions/v1beta1

kind: DaemonSet

metadata:

  name: cpx

spec:

  template:

    metadata:

      name: cpx
```



```
16
17            labels:
18
19              app: cpx-daemon
20
21            annotations:
22
23              NETSCALER_AS_APP: "True"
24
25          spec:
26
27            hostNetwork: true
28
29            containers:
30
31              - name: cpx
32
33                image: "<repository>/cpx:12.0-64"
34
35                securityContext:
36
37                   privileged: true
38
39                env:
40
41                - name: "EULA"
42
43                  value: "yes"
44
45                - name: "NS_NETMODE"
46
47                  value: "HOST"
48
49                - name: "kubernetes_url"
50
51                  value: "https://10.217.212.231:6443"
52
53                - name: "NS_MGMT_SERVER"
54
55                  value: "10.217.212.226"
56
57                - name: "NS_MGMT_FINGER_PRINT"
58
59                  value: "74:EA:04:90:2C:FA:BF:7A:31:C9:52:64:D3:9C:BC:D3
          :O8:9F:9A:O4"
```



```
60
61                - name: ”NS_ROUTABLE”
62
63                  value: ”FALSE”
64
65                - name: ”KUBERNETES_TASK_ID”
66
67                  valueFrom:
68
69                     fieldRef:
70
71                        fieldPath: metadata.name
72
73                volumeMounts:
74
75                imagePullPolicy: Always
```

다음표는샘플디먼세트에서사용된섹션, 매개변수및환경변수에대해설명합니다.

섹션	매개변수	설명
컨테이너	name	NetScaler CPX 컨테이너의이름입니다.
	image	컨테이너생성을위한이미지를지정합니다.
SecurityContext	privileged: true	NetScaler CPX 컨테이너가권한있는모드에서실행되도록지정합니다.
	name: “EULA”	NetScaler CPX 관련환경변수로서 https://www.citrix.com/products/netscaler-adc/cpx-express.html에있는 EULA(최종사용자사용권계약) 를읽고이해했음을확인하는데필요합니다.

섹션	매개변수	설명
	name: “NS_NETMODE”	NetScaler CPX 인스턴스가호스트 모드에서시작되도록지정할수있게해주는 NetScaler CPX 관련환경변수입니다. 호스트모드에서시작된인스턴스는해당인스턴스의관리액세스를위해호스트컴퓨터에 4 개의기본 iptable 규칙을구성합니다. HTTP 의경우 9995, HTTPS 의경우 9996, SSH 의경우 9997, SNMP 의경우 9998 포트를사용합니다. 다른포트를지정하려면 -e NS_HTTP_PORT, -e NS_HTTPS_PORT, -e NS_SSH_PORT, -e NS_SNMP_PORT 와같은환경변수를사용할수도있습니다.
	name: “kubernetes_url”	Kubernetes URL 을지정하는 NetScaler CPX 관련환경변수입니다.
	name: “NS_MGMT_SERVER”	NetScaler MAS 서버 IP 주소를설명하는 NetScaler CPX 관련환경변수입니다. NetScaler CPX 인스턴스가배포되면이는자동으로이 IP 주소에 NetScaler MAS 서버와함께등록합니다.
	name: “NS_MGMT_FINGER_PRINT”	NetScaler MAS 지문을정의하는 NetScaler CPX 관련환경변수입니다.
	name: “NS_ROUTABLE”	NetScaler CPX 컨테이너가 non-IP-per-container 모드에서실행되는지아닌지를지정하는 NetScaler CPX 관련환경변수입니다. 값을 FALSE 로설정하십시오.
	name: “KUBERNETES_TASK_ID”	Kubernetes 클러스터에서 NetScaler CPX ID 를식별합니다.

섹션	매개변수	설명
imagePullPolicy		Kubernetes 가이미지를가져오는방법을지정합니다.

매니페스트를사용하여 **NetScaler CPX** 인스턴스배포

Kubernetes 매니페스트는 Kubernetes 개체배포와구성지침을포함한 YAML 또는 JSON 형식의파일입니다. NetScaler CPX 인스턴스의 Kubernetes 매니페스트를생성하고이를노드의특정디렉터리에배치할수있습니다. 각노드의 kubelet 이이 디렉터리를모니터링하고매니페스터에지정된대로 NetScaler CPX 인스턴스인개체를만듭니다.

다음은샘플매니페스트입니다.

```
apiVersion: v1

kind: Pod

metadata:

    name: cpx

annotations:

    NETSCALER_AS_APP: "True"

spec:

    hostNetwork: true

    containers:

        - name: cpx

          image: "<repository>/cpx:12.0-64"

    securityContext:

    privileged: true

    env:

        - name: "EULA"

```



```
31              value: "yes"
32
33              - name: "NS_NETMODE"
34
35              value: "HOST"
36
37              - name: "kubernetes_url"
38
39              value: "https://10.217.212.231:6443"
40
41              - name: "NS_MGMT_SERVER"
42
43              value: "10.217.212.226"
44
45              - name: "NS_MGMT_FINGER_PRINT"
46
47              value: "74:EA:04:90:2C:FA:BF:7A:31:C9:52:64:D3:9C:BC:D3:08
                    :9F:9A:04"
48
49              - name: "NS_ROUTABLE"
50
51              value: "FALSE"
52
53              - name: "KUBERNETES_TASK_ID"
54
55              valueFrom:
56
57                  fieldRef:
58
59                  fieldPath: metadata.name
60
61          imagePullPolicy: Always
```

다음표는샘플매니페스트에서사용된섹션, 매개변수및환경변수에대해설명합니다.

섹션	매개변수	설명
컨테이너	name	NetScaler CPX 컨테이너의이름입니다.
	image	컨테이너생성을위한이미지를지정합니다.
SecurityContext	privileged: true	NetScaler CPX 컨테이너가권한있는모드에서실행되도록지정합니다.

섹션	매개변수	설명
	name: “EULA”	NetScaler CPX 관련환경변수로서 https://www.citrix.com/products/netscaler-adc/cpx-express.html에있는 EULA(최종사용자사용권계약) 를읽고이해했음을확인하는데필요합니다.
	name: “NS_NETMODE”	NetScaler CPX 인스턴스가호스트모드에서시작되도록지정할수있게해주는 NetScaler CPX 관련환경변수입니다. 호스트모드에서시작된인스턴스는해당인스턴스의관리액세스를위해호스트컴퓨터에 4 개의기본 iptable 규칙을구성합니다. HTTP 의경우 9995, HTTPS 의경우 9996, SSH 의경우 9997, SNMP 의경우 9998 포트를사용합니다. 다른포트를지정하려면 -e NS_HTTP_PORT, -e NS_HTTPS_PORT, -e NS_SSH_PORT, -e NS_SNMP_PORT 와같은환경변수를사용할수도있습니다.
	name: “kubernetes_url”	Kubernetes URL 을지정하는 NetScaler CPX 관련환경변수입니다.
	name: “NS_MGMT_SERVER”	NetScaler MAS 서버 IP 주소를설명하는 NetScaler CPX 관련환경변수입니다. NetScaler CPX 인스턴스가배포되면이는자동으로이 IP 주소에 NetScaler MAS 서버와함께등록합니다.
	name: “NS_MGMT_FINGER_PRINT”	NetScaler MAS 지문을정의하는 NetScaler CPX 관련환경변수입니다.

섹션	매개변수	설명
	name: “NS_ROUTABLE”	NetScaler CPX 컨테이너가 non-IP-per-container 모드에서 실행되는지아닌지를지정하는 NetScaler CPX 관련환경변수입니다. 값을 FALSE 로설정하십시오.
	name: “KUBERNETES_TASK_ID”	Kubernetes 클러스터에서 NetScaler CPX ID 를식별합니다.
imagePullPolicy		Kubernetes 가이미지를가져오는방법을지정합니다.

NetScaler CPX 를사용하여 Kubernetes 환경에서수신트래픽부하분산

June 11, 2018

2017 년 10 월 5 일

Kubernetes 환경에서 Kubernetes 서비스의수신트래픽부하를분산하려면수신리소스와수신컨트롤러가필요합니다. 수신리소스는 Kubernetes 서비스의부하분산을구성할수있는 Kubernetes 리소스입니다. 부하분산장치는서비스를위해외부에연결할수있는 URL 을제공하여 Kubernetes 클러스터밖에있는클라이언트에서비스를노출시키고이러한 URL 로전송되는트래픽의부하를분산합니다. NetScaler CPX 는 Kubernetes 환경에서수신부하분산장치를사용하여 Kubernetes 클러스터외부의클라이언트에의해발생하는 Kubernetes 서비스에대한종적트래픽부하를분산할수있습니다.

수신컨트롤러에는 Kubernetes 와부하분산장치가통합되어있습니다. 이컨트롤러는 Kubernetes API 를통해수신리소스를모니터링하고확장, 업데이트롤링또는메타데이터변경으로인해서비스에어떠한변화가발생하는경우부하분산장치의구성을업데이트합니다. NetScaler MAS(Management and Analytics System) 는 Kubernetes 환경용 NetScaler 수신컨트롤러를포함하고있습니다. Kubernetes 클러스터에배포된 NetScaler CPX 인스턴스와수신컨트롤러를사용하여 Kubernetes 환경의수신트래픽을처리할수있습니다. NetScaler MAS 에대한자세한내용은 NetScaler Management and Analytics System 제품설명서를참조하십시오.

Kubernetes 의수신리소스와컨트롤러에대한자세한내용은 수신리소스를참조하십시오.

NetScaler 수신컨트롤러의작동방법

Kubernetes 클러스터를배포한후에는 NetScaler MAS 에서 Kubernetes 환경세부정보를제공해 NetScaler MAS 와클러스터를통합해야합니다. NetScaler MAS 는서비스, 포드, 수신규칙등 Kubernetes 리소스의변경사항을모니터링합니다.

Kubernetes 클러스터에수신리소스로 NetScaler CPX 인스턴스를배포하면자동으로 NetScaler MAS 를함께등록합니다. 등록프로세스의일부로 NetScaler MAS 는 NITRO REST API 를사용하여 NetScaler 관련구성을구성하기위해인스턴스에

연결할포트와 NetScaler CPX 인스턴스 IP 주소에대한정보를파악합니다.

NetScaler MAS 의 Stylebook 엔진은 NetScaler MAS 가서비스, 포드, 수신규칙등 Kubernetes 에서수집하는모든정보를처리합니다. Stylebook 엔진은기존에프로비저닝된 Stylebook(***com.citrix.adc.stylebooks/1.0/cs-lb-mon***) 을사용하여부하분산에필요한가상서버, 서비스및서비스그룹과같은 NetScaler 구성을생성하고 NetScaler CPX 수신부하분산장치에해당구성을적용합니다. StyleBook 에대한자세한내용은 Stylebook을참조하십시오.

다음다이어그램은 Kubernetes 환경을보여주며, 수신트래픽을처리하기위해 Kubernetes 클러스터의 NetScaler CPX 수신리소스와통합된 NetScaler 수신컨트롤러가포함되어있습니다.

이예에서는 VIP(가상 IP) 주소를통해클러스터외부로부터 Kubernetes 서비스로들어오는트래픽의부하분산을위해 NetScaler CPX 컨테이너가배포되었습니다. NetScaler CPX 컨테이너는 A 및 B 서비스를구성하는여러 Kubernetes 포드간요청을분산시켜종적트래픽의부하를분산합니다.

> 중요
>
> NetScaler CPX 호스트 IP 주소를사용하여 NetScaler CPX 컨테이너로트래픽을보내기위해도메인 api.example.com 에대한 DNS 구성이구성되었습니다. 수신부하분산장치로여러개의 NetScaler CPX 컨테이너가구성된경우 DNS 방법을사용하여 NetScaler CPX 컨테이너에걸쳐수신트래픽이분산되도록하십시오.

NetScaler MAS 는 Kubernetes 클러스터에있는 NetScaler 장치를관리하고장치에대한풍부한분석을통해통찰력과문제해결을제공합니다. 또한 NetScaler 장치에서상세한트래픽통계를수집하여응용프로그램성능과보안에대한가시성을제공합니다.

Kubernetes 환경에서수신부하분산장치로 **NetScaler CPX** 배포

NetScaler CPX 는 Kubernetes 환경을위한수신부하분산장치로사용될수있습니다. 클러스터내노드에서 Kubernetes 포드로 NetScaler CPX 컨테이너를배포하거나, 클러스터외부의호스트가다른 Kubernetes 노드로서동일한오버레이네트워크에참여하는경우해당호스트에이컨테이너를배포할수있습니다.

사전요구사항

시작하기전에다음을수행하십시오.

- Kubernetes 클러스터를만듭니다. 자세한내용은 http://kubernetes.io/docs/을참조하십시오.
- NetScaler MAS 를배포합니다. NetScaler MAS 를배포하는방법에대한자세한내용은 http://docs.citrix.com/ko-kr/netscaler-mas/11-1/single-server-deployment.html를참조하십시오.
- NetScaler MAS 에 Kubernetes 클러스터를구성합니다. 지침은Kubernetes 환경의수신컨트롤러로 NetScaler MAS 사용.
- NetScaler MAS 서버의지문을확인합니다. 지문을얻으려면다음을수행하십시오.
    1. NetScaler MAS 에로그인합니다.
    2. **System(**시스템**) > System Administration(**시스템관리**)** 으로이동하고 **View SSL Certificate(SSL** 인증서보기**)** 을클릭합니다.
    3. **Certificate Details(**인증서세부정보**)** 페이지의 **Fingerprint(**지문**)** 필드에서 NetScaler MAS 서버의지문을복사합니다.

Kubernetes 클러스터외부에수신부하분산장치로 **NetScaler CPX** 배포

NetScaler CPX 는 Kubernetes 클러스터외부에수신부하분산장치로배포될수있습니다. 클러스터외부에있는호스트는다른 Kubernetes 노드로서동일한오버레이네트워크에참여해야합니다.

NetScaler CPX 를 **Kubernetes** 클러스터외부에있는호스트의수신부하분산장치로배포하려면**,**
호스트에서다음 ***docker run*** 명령을사용하여 Docker 컨테이너에 NetScaler CPX 인스턴스를배포합니다.

```
docker run -dt --privileged=true -p <port_number> -e NS_HTTP_PORT=<
    netscaler_HTTP_port> -e NS_HTTPS_PORT=<netscaler_HTTPS_port> -e EULA
    =yes -e NS_MGMT_SERVER=<MAS_IP_address> -e NS_MGMT_FINGER_PRINT="<
    MAS_finger_print>" -e NS_ROUTABLE=<True|False> -e NS_LB_ROLE=<
    lb_role> -e HOST=$HOSTNAME store/citrix/netscalercpx:12.0-53.6
```

예**:**

```
docker run -dt --privileged=true -p 5080:80 -p 5443:443 -p 80:5080 -e
    NS_HTTP_PORT=5080 -p 443:5443 -e NS_HTTPS_PORT=5443 -e EULA=yes -e
    NS_MGMT_SERVER=10.217.212.226 -e NS_MGMT_FINGER_PRINT="74:EA:04:90:2
    C:FA:BF:7A:31:C9:52:64:D3:9C:BC:D3:08:9F:9A:04" -e NS_ROUTABLE=FALSE
     -e NS_LB_ROLE=SERVER -e HOST=$HOSTNAME store/citrix/netscalercpx
    :12.0-53.6
```

이명령은 NetScaler CPX Docker 컨테이너를배포합니다. 다음표는 ***docker run*** 명령에서사용된다양한옵션과환경변수에 대한설명입니다.

옵션및 NetScaler CPX 관련환경변수	설명
-dt	NetScaler CPX 컨테이너가디먼실행되도록지정합니다.
–privileged=true	NetScaler CPX 컨테이너가권한있는모드에서실행되도록 지정합니다.
-p	NetScaler CPX 와호스트간포트를매핑합니다. 기본적으로 Kubernetes 수신개체는클러스터가포트 80 및 443 을 사용하여액세스하는것으로가정합니다.
-p 5080:80	컨테이너의포트 80 과호스트의포트 5080 을바인딩합니다.
-p 5443:443	컨테이너의포트 443 과호스트의포트 5443 을바인딩합니다.
-p 443:5443	컨테이너의포트 5443 과호스트의포트 443 을바인딩합니다.
-p 80:5080	컨테이너의포트 5080 과호스트의포트 80 을바인딩합니다.

옵션및 NetScaler CPX 관련환경변수	설명
-e NS_HTTP_PORT 또는 -e NS_HTTPS_PORT	NetScaler CPX 에대한관리액세스를위해사용자지정포트를할당할수있게해주는 NetScaler CPX 관련환경변수입니다. NetScaler MAS 는이러한포트를사용해 NetScaler CPX 에액세스합니다.
-e NS_MGMT_SERVER	NetScaler MAS 서버 IP 주소를정의할수있게해주는 NetScaler CPX 관련환경변수입니다. NetScaler CPX 가배포되면자동으로이 IP 주소를사용하여 NetScaler MAS 서버와함께등록합니다.
-e NS_MGMT_FINGER_PRINT	NetScaler MAS 지문을정의하는 NetScaler CPX 관련환경변수입니다.
-e NS_ROUTABLE=FALSE	NetScaler CPX 컨테이너가 non-IP-per-container 모드에서실행되도록지정하는 NetScaler CPX 관련환경변수입니다.
-e NS_LB_ROLE=SERVER	NetScaler CPX 컨테이너가수신리소스로사용되도록 NetScaler CPX 와 NetScaler MAS 에지정하는 NetScaler CPX 관련환경변수입니다.
-e HOST=$HOSTNAME	NetScaler MAS 가 NetScaler CPX 컨테이너에액세스하는데사용할수있는호스트이름을지정하는 NetScaler CPX 관련환경변수입니다. 호스트이름이 NetScaler MAS 에의해확인될수있도록하고, 그렇지않은경우 IP 주소를제공하십시오.

호스트에 NetScaler CPX 인스턴스를배포하면자동으로 NetScaler MAS(Management and Analytics System) 와 함께등록합니다. NetScaler MAS UI 에배포된 NetScaler CPX 인스턴스는 **Networks(**네트워크**) > Instances(**인스턴스**) > NetScaler CPX** 에서볼수있습니다.

Kubernetes 클러스터내에수신부하분산장치로 **NetScaler CPX** 배포

Kubernetes 클러스터내에수신부하분산장치로 NetScaler CPX 를배포하려면 Kubernetes 클러스터에있는노드에 Kubernetes 포드로배포하십시오.

Kubernetes 클러스터내에수신부하분산장치로 **NetScaler CPX** 배포

1. (선택사항) 클러스터의특정노드에 Kubernetes 포드로 NetScaler CPX 를배포하려는경우레이블을사용하여노드를지정할 수있습니다. Kubernetes 노드에레이블을지정하려면 ***kubectl*** 명령을사용합니다.

```
kubectl label nodes <node_IP_address> node-role=<label_name>
```


예:

```
kubectl label nodes 10.217.222.224 node-role=ingress
```

노드에레이블을지정했으면포드사양에레이블을지정하여포드를노드에배포할수있습니다.

2. NetScaler CPX 의포드사양을정의하여 NetScaler CPX 컨테이너를 Kubernetes 클러스터의포드로배포합니다. 포드 사양은 YAML 파일또는 JSON 스크립트로정의됩니다. YAML 파일또는 JSON 스크립트에는컨테이너유형, CPX 이미지파일 이름, NetScaler MAS 서버 IP 주소및 NetScaler MAS 서버지문이포함되어야합니다.

다음은 NetScaler CPX 의포드사양예입니다.

```
    apiVersion: v1
    kind: Pod
    metadata:
    name: cpx-ingress
    annotations:
        NETSCALER_AS_APP: "True"
    spec:
        containers:
            - name: cpx-ingress
              image: "cpx:12.0-41.16"
        securityContext:
            privileged: true
        env:
            - name: "EULA"
              value: "yes"
            - name: "NS_MGMT_SERVER"
              value: "10.217.212.226"
            - name: "NS_MGMT_FINGER_PRINT"
              value: "74:EA:04:90:2C:FA:BF:7A:31:C9:52:64:D3:9C:BC:D3:
                 O8:9F:9A:O4"
            - name: "NS_ROUTABLE"
              value: "FALSE"
            - name: "NS_HTTP_PORT"
              value: "5080"
            - name: "NS_HTTPS_PORT"
              value: "5443"
            - name: "NS_LB_ROLE"
              value: "SERVER"
            - name: "HOST"
              value: ""
            - name: "KUBERNETES_TASK_ID"
        valueFrom:
        fieldRef:
        fieldPath: metadata.name
            - name:"HOST"
```



```
    valueFrom:
        fieldRef:
            fieldPath: spec.nodeName
        ports:
            - containerPort: 80
              hostPort: 5080
            - containerPort: 443
              hostPort: 5443
            - containerPort: 5080
              hostPort: 80
            - containerPort: 5443
              hostPort: 443
        imagePullPolicy: Always
        nodeSelector:
            node-role: ingress
```

또는포드사양을정의하여 NetScaler CPX 를복제컨트롤러로배포할수있습니다. 이렇게하면 NetScaler CPX 가중단되는경우 Kubernetes 가클러스터에 NetScaler CPX 컨테이너를다시만듭니다. 다음은샘플사양입니다.

```
apiVersion: v1
kind: ReplicationController
metadata:
name: cpx-ingress
spec:
    replicas: 1
    selector:
        app: cpx-ingress-device
    template:
    metadata:
        name: cpx-ingress
    annotations:
        NETSCALER_AS_APP: "True"
    labels:
        app: cpx-ingress-device
    spec:
        containers:
            - name: cpx-ingress
            image: "cpx:12.0-41.16"
        securityContext:
            privileged: true
        env:
            - name: "EULA"
            value: "yes"
            - name: "NS_MGMT_SERVER"
            value: "10.217.212.226"
```



```
27                  - name: ”NS_MGMT_FINGER_PRINT”
28                  value: ”74:EA:04:90:2C:FA:BF:7A:31:C9:52:64:D3:9C:BC:D3
                        :O8:9F:9A:O4”
29                  - name: ”NS_ROUTABLE”
30                  value: ”FALSE”
31                  - name: ”NS_HTTP_PORT”
32                  value: ”5080”
33                  - name: ”NS_HTTPS_PORT”
34                  value: ”5443”
35                  - name: ”NS_LB_ROLE”
36                  value: ”SERVER”
37                  - name: ”HOST”
38                  value: ””
39                  - name: ”KUBERNETES_TASK_ID”
40          valueFrom:
41          fieldRef:
42          fieldPath: metadata.name
43              - name:”HOST”
44      valueFrom:
45          fieldRef:
46              fieldPath: spec.nodeName
47          ports:
48              - containerPort: 80
49              hostPort: 5080
50              - containerPort: 443
51              hostPort: 5443
52              - containerPort: 5080
53              hostPort: 80
54              - containerPort: 5443
55              hostPort: 443
56          imagePullPolicy: Always
57          nodeSelector:
58              node-role: ingress
```

다음표는위에나온예에서사용된섹션, 매개변수및환경변수에대해설명합니다.

섹션	매개변수	설명
containers	name	NetScaler CPX 컨테이너의이름입니다.
	image	컨테이너생성을위한이미지를지정합니다.
securityContext	privileged: true	NetScaler CPX 컨테이너가권한있는모드에서실행되도록지정합니다.

섹션	매개변수	설명
env	name: “EULA”	NetScaler CPX 관련환경변수로서 https://www.citrix.com/products/netscaler-adc/cpx-express.html에있는 EULA(최종사용자사용권계약) 를읽고이해했음을확인하는데필요합니다.
	name: “NS_MGMT_SERVER”	NetScaler MAS 서버 IP 주소를정의할수있게해주는 NetScaler CPX 환경변수입니다. NetScaler CPX 가배포되면자동으로이 IP 주소를사용하여 NetScaler MAS 서버와함께등록합니다.
	name: “NS_MGMT_FINGER_PRINT”	NetScaler MAS 지문을정의할수있게해주는 NetScaler CPX 환경변수입니다.
	name: “NS_ROUTABLE”	NetScaler CPX 컨테이너가 non-IP-per-container 모드에서실행할수있게해주는 NetScaler CPX 환경변수입니다. 값을 “FALSE”로설정하십시오.
	name: “NS_HTTP_PORT” 또는 name: “NS_HTTPS_PORT”	NetScaler CPX 에대한관리액세스를위해사용자지정포트를할당할수있게해주는 NetScaler CPX 관련환경변수입니다. NetScaler MAS 는이러한포트를사용해 NetScaler CPX 컨테이너에액세스합니다.
	name: “NS_LB_ROLE”	NetScaler CPX 컨테이너가수신리소스로사용되도록 NetScaler CPX 와 NetScaler MAS 에지정하는 NetScaler CPX 환경변수입니다.
	name: “HOST”	NetScaler CPX 컨테이너가실행중인노드의호스트이름입니다. 호스트이름을사용하여 NetScaler MAS 가 NetScaler CPX 컨테이너에액세스할수있습니다.

섹션	매개변수	설명
	name: “KUBERNETES_TASK_ID”	Kubernetes 클러스터에서 NetScaler CPX ID 를식별합니다.
	name: “HOST”	NetScaler CPX 컨테이너가실행중인노드의호스트이름입니다. 호스트이름을사용하여 NetScaler MAS 가 NetScaler CPX 에액세스할수있습니다.
포트	containerPort: 또는 hostPort:	NetScaler CPX 컨테이너와호스트간포트를매핑합니다. 기본적으로 Kubernetes 수신개체는클러스터가포트 80 및 443 에서액세스하는것으로가정합니다.
imagePullPolicy		Kubernetes 가이미지를가져오는방법을지정합니다.
nodeSelector	node-role:	포드를배포하려는노드의레이블입니다.

3. 다음명령을사용하여 NetScaler CPX 의포드사양을배포합니다.

```
kubectl create -f (fileName | scriptName)
```

예:

```
kubectl create -f sample.yaml
```

호스트에 NetScaler CPX 인스턴스를배포하면자동으로 NetScaler MAS(Management and Analytics System) 와함께등록합니다. NetScaler MAS UI 에배포된 NetScaler CPX 인스턴스는 **Networks(**네트워크**) > Instances(**인스턴스**) > NetScaler CPX** 에서볼수있습니다.

ConfigMaps 를사용하여 **Kubernetes** 에서 **NetScaler CPX** 구성

June 11, 2018

2018 년 3 월 23 일

ConfigMaps 를사용하여 Kubernetes 에서 NetScaler CPX 인스턴스를구성할수있습니다. ConfigMaps 를사용하면 NetScaler CPX 인스턴스를시작하는동안동적으로구성할수있습니다.

NetScaler CPX 인스턴스에서동적으로실행할배시셸 (shell) 명령과 NetScaler 관련구성을포함한 `cpx.conf` 구성파일을 만듭니다. 구성파일구조에는 `##NetScaler Commands`와 `##Shell Commands`라는두가지유형의태그가필요합니다. `##NetScaler Commands` 태그에서는 NetScaler CPX 인스턴스에서 NetScaler 관련구성을구성할수있도록모든 NetScaler 명령을추가해야합니다. `##Shell Commands` 태그에서는 NetScaler CPX 인스턴스에서실행할셸명령을추가해야합니다.

> 중요:
>
> - 태그는구성파일에서여러번반복될수있습니다.
> - 구성파일에는설명도포함할수있습니다. 설명앞에 "#" 기호를추가하십시오.
> - 태그는대/소문자가구분되지않습니다.
> - 구성파일로 NetScaler CPX 컨테이너를배포하는도중오류시나리오가발생하면 `ns.log` 파일에오류가기록됩니다.
> - NetScaler CPX 인스턴스가시작된후 ConfigMap 을변경하면 NetScaler CPX 인스턴스를다시시작해야업데이트된구성이적용됩니다.

다음은샘플구성파일입니다.

```
#NetScaler Commands
add lb vserver v1 http 1.1.1.1 80
add service s1 2.2.2.2 http 80
bind lb vserver v1 s1
#Shell Commands
touch /etc/a.txt
echo "this is a" > /etc/a.txt
#NetScaler Commands
add lb vserver v2 http
#Shell Commands
echo "this is a 1" >> /etc/a.txt
#NetScaler Commands
add lb vserver v3 http
```

구성파일을만들었으면 `kubectl create configmap` 명령을사용하여구성파일에서 ConfigMap 을만들어야합니다.

```
kubectl create configmap cpx-config --from-file=cpx.conf
```

위의예에서는구성파일 cpx.conf 를기반으로 `cpx-config`라는 ConfigMap 을만들수있습니다. 그런다음 NetScaler CPX 인스턴스를배포하는데사용한 YAML 파일에이 ConfigMap 을사용할수있습니다.

`kubectl get configmap` 명령을사용하여생성된 ConfigMap 을볼수있습니다.

```
root@node1:~/yaml## kubectl get configmap cpx-config -o yaml
```

샘플:

```
    apiVersion: v1
    data:
```



```
        cpx.conf: |
        #NetScaler Commands
            add lb vserver v1 http 1.1.1.1 80
            add service s1 2.2.2.2 http 80
            bind lb vserver v1 s1
        #Shell Commands
            touch /etc/a.txt
            echo "this is a" > /etc/a.txt
            echo "this is the file" >> /etc/a.txt
            ls >> /etc/a.txt
        #NetScaler Commands
            add lb vserver v2 http
        #Shell Commands
            echo "this is a 1" >> /etc/a.txt
        #NetScaler Commands
            add lb vserver v3 http
        #end of file
    kind: ConfigMap
    metadata:
        creationTimestamp: 2017-12-26T06:26:50Z
        name: cpx-config
        namespace: default
        resourceVersion: "8865149"
        selfLink: /api/v1/namespaces/default/configmaps/cpx-config
        uid: c1c7cb5b-ea05-11e7-914a-926745c10b02
```

아래에표시된것처럼생성한 ConfigMap(`cpx-config`) 을 NetScaler CPX 를배포하는데사용한 YAML 파일에서지정할 수있습니다.

```
        apiVersion: v1
    kind: Pod
    metadata:
        name: cpx-1
        labels:
            app: cpx-daemon
    annotations:
        NETSCALER_AS_APP: "True"
    spec:
        hostNetwork: true
        containers:
            - name: cpx
              image: "cpx:latest"
        securityContext:
            privileged: true
        volumeMounts:
```



```
17              - name: config-volume
18                mountPath: /cpx/conf
19          env:
20              - name: ”EULA”
21                value: ”yes”
22              - name: ”NS_NETMODE”
23                value: ”HOST”
24              - name: ”kubernetes_url”
25                value: ”https://10.106.76.31:6443”
26              - name: ”NS_MGMT_SERVER”
27                value: ”10.106.76.144”
28              - name: ”NS_MGMT_FINGER_PRINT”
29                value: ”A6:1F:7C:16:90:42:52:C9:0F:74:59:28:E7:A5:D6:D6:3
                      C:6D:DC:DE”
30              - name: ”NS_ROUTABLE”
31                value: ”FALSE”
32              - name: ”KUBERNETES_TASK_ID”
33          valueFrom:
34          fieldRef:
35          fieldPath: metadata.name
36          imagePullPolicy: Never
37          volumes:
38              - name: config-volume
39          configMap:
40              name: cpx-config
41          nodeSelector:
42              node: node1
```

NetScaler CPX 인스턴스가배포되고 ConfigMap 에지정된구성을시작하면 `cpx-config`가 NetScaler CPX 인스턴스에적용됩니다.

Google Compute Engine 에서 NetScaler CPX 프록시배포

June 11, 2018

2017 년 6 월 30 일

이배포가이드에서는엔터프라이즈네트워크내에서실행되는 NetScaler MAS 를사용하여 Google Cloud 의 GCE(Google Compute Engine) 에서 Docker 와함께 NetScaler CPX 를배포하는방법을설명합니다. 이배포에서는 GCE 에설치된 NetScaler CPX 가두백엔드서버의부하를분산하며, NetScaler MAS 가라이센스및분석솔루션을제공합니다.

NetScaler CPX 는컨테이너기반의프록시로서전체 L7 기능, SSL 오프로드, 여러프로토콜및 NITRO API 를지원합니다. NetScaler MAS 는관리, 라이센스및분석솔루션을제공합니다. NetScaler MAS 는라이센스서버로서온-프레미스또는클라우

드에서실행되는 NetScaler CPX 인스턴스에권한을부여합니다.

CPX 와 CPX Express 는동일한이미지입니다. Docker App Store(릴리스 11 또는 12) 의 CPX 이미지가 NetScaler MAS 를통해라이센스를부여받고설치되면정식 CPX 인스턴스가됩니다. 라이센스가없는경우 CPX 이미지는 CPX Express 인스턴스가되어 20Mbps 및 250 개 SSL 연결을지원합니다.

사전요구사항

- NetScaler CPX 전용 2GB 메모리및 vCPU 1 개
- GCE 에서사용가능한 Docker 오픈소스
- 온-프레미스에서실행되며인터넷또는 VPN 을통해 GCE 에연결된 NetScaler MAS

> 참고
>
> NetScaler MAS 를배포하는방법에대한자세한내용은 NetScaler MAS 배포를참조하십시오.

구성단계

이배포를구성하려면다음단계를수행해야합니다.

1. GCE VM 에 Docker 를설치합니다.
2. Docker 인스턴스와의원격 API 통신을구성합니다.
3. NetScaler CPX 이미지를설치합니다.
4. CPX 인스턴스를만듭니다.
5. NetScaler MAS 를통해 NetScaler CPX 에라이센스를부여합니다.
6. NetScaler CPX 에서부하분산서비스를구성하고구성을확인합니다.

a) NGINX 웹서버를설치합니다.

b) CPX 에부하분산을구성하고두웹서비스로부하가분산되는지확인합니다.

1 단계**: GCE VM** 에 **Docker** 설치

GCE 에서 Linux Ubuntu VM 을만듭니다. 그런후에다음예제에표시된명령을사용하여 VM 에 Docker 를설치합니다.

```
$ sudo curl  - ssl https://get.docker.com/ | sh
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
0 0 0 0 0 0 0 0 --:--:-- --:--:-- --:--:-- 0curl: (6) Could not resolve
    host: xn--ssl-1n0a
100 17409 100 17409 0 0 21510 0 --:--:-- --:--:-- --:--:-- 21492
```



```
apparmor is enabled in the kernel and apparmor utils were already
   installed
+ sudo -E sh -c apt-key add -
+ echo -----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1

mQINBFWln24BEADrBl5p99uKh8+rpvqJ48u4eTtjeXAWbslJotmC/CakbNSqOb9o
ddfzRvGVeJVERt/Q/mlvEqgnyTQy+e6oEYN2Y2kqXceUhXagThnqCoxcEJ3+KM4R
mYdoe/BJ/J/6rHOjq7Omk24z2qB3RU1uAv57iY5VGw5p45uZB4C4pNNsBJXoCvPn
TGAs/7IrekFZDDgVraPx/hdiwopQ8NltSfZCyu/jPpWFK28TR8yfVlzYFwibj5WK
dHM7ZTqlA1tHIG+agyPf3Rae0jPMsHR6q+arXVwMccyOi+ULU0z8mHUJ3iEMIrpT
X+80KaN/ZjibfsBOCjcfiJSB/acn4nxQQgNZigna32velafhQivsNREFeJpzENiG
HOoyC6qVeOgKrRiKxzymj0FIMLru/iFF5pSWcBQB7PYlt8J0G80lAcPr6VCiN+4c
NKv03SdvA69dCOj79PuO9IIvQsJXsSq96HB+TeEmmL+xSdpGtGdCJHHM1fDeCqkZ
hT+RtBGQL2SEdWjxbF43oQopocT8cHvyX6Zaltn0svoGs+wX3Z/H6/8P5anog43U
65c0A+64Jj00rNDr8j31izhtQMRo892kGeQAaaxg4Pz6HnS7hRC+cOMHUU4HA7iM
zHrouAdYeTZeZEQOA7SxtCME9ZnGwe2grxPXh/U/80WJGkzLFNcTKdv+rwARAQAB
tDdEb2NrZXIgUmVsZWFzZSBUb29sIChyZWxlYXNlZG9ja2VyKSA8ZG9ja2VyQGRv
Y2tlci5jb20+iQIcBBABCgAGBQJWw7vdAAoJEFyzYeVS+w0QHysP/i37m4SyoOCV
cnybl18vzwBEcp4VCRbXvHvOXty1gccVIV8/aJqNKgBV97lY3vrpOyiIeB8ETQeg
srxFE7t/Gz0rsLObqfLEHdmn5iBJRkhLfCpzjeOnyB3Z0IJB6UogO/msQVYe5CXJ
l6uwr0AmoiCBLrVlDAktxVh9RWch0l0KZRX2FpHu8h+uM0/zySqIidlYfLa3y5oH
scU+nGU1i6ImwDTD3ysZC5jp9aVfvUmcESyAb4vvdcAHR+bXhA/RW8QHeeMFliWw
7Z2jYHyuHmDnWG2yUrnCqAJTrWV+OfKRIzzJFBs4e88ru5h2ZIXdRepw/+COYj34
LyzxR2cxr2u/xvxwXCkSMe7F4KZAphD+1ws61FhnUMi/PERMYfTFuvPrCkq4gyBj
t3fFpZ2NR/fKW87QOeVcn1ivXl9id3MMs9KXJsg7QasT7mCsee2VIFsxrkFQ2jNp
D+JAERRn9Fj4ArHL5TbwkkFbZZvSi6fr5h2GbCAXIGhIXKnjjorPY/YDX6X8AaHO
W1zblWy/CFr6VFl963jrjJgag0G6tNtBZLrclZgWhOQpeZZ5Lbvz2ZA5CqRrfAVc
wPNW1fObFIRtqV6vuVluFOPCMAAnOnqR02w9t17iVQjO3oVN0mbQi9vjuExXh1Yo
ScVetiO6LSmlQfVEVRTqHLMgXyR/EMo7iQIcBBABCgAGBQJXSWBlAAoJEFyzYeVS
+w0QeH0QAI6btAfYwYPuAjfRUy9qlnPhZ+xt1rnwsUzsbmo8K3XTNh+l/R08nu0d
sczw30Q1wju28fh1N8ay223+69f0+yICaXqR18AbGgFGKX7vo0gfEVaxdItUN3eH
NydGFzmeOKbAlrxIMECnSTG/TkFVYO9Ntlv9vSN2BupmTagTRErxLZKnVsWRzp+X

-----END PGP PUBLIC KEY BLOCK-----

OK
+ sudo -E sh -c mkdir -p /etc/apt/sources.list.d
+ dpkg --print-architecture
+ sudo -E sh -c echo deb \[arch=amd64\] https://apt.dockerproject.org/
   repo ubuntu-yakkety main > /etc/apt/sources.list.d/docker.list
+ sudo -E sh -c sleep 3; apt-get update; apt-get install -y -q docker-
   engine
Hit:1 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety InRelease
Get:2 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety-updates
```

```
      InRelease [102 kB]
48 Get:3 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety-backports
      InRelease [102 kB]
49 Get:4 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety/restricted
      Sources [5,376 B]
50 Get:5 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety/multiverse
      Sources [181 kB]
51 Get:6 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety/universe
      Sources [8,044 kB]
52 Get:7 http://archive.canonical.com/ubuntu yakkety InRelease [11.5 kB]
53 Get:8 http://security.ubuntu.com/ubuntu yakkety-security InRelease [102
       kB]
54 Get:9 https://apt.dockerproject.org/repo ubuntu-yakkety InRelease [47.3
       kB]
55 Get:10 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety/main
      Sources [903 kB]
56 Get:11 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety-updates/
      restricted Sources [2,688 B]
57 Get:12 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety-updates/
      universe Sources [57.9 kB]
58 Get:13 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety-updates/
      multiverse Sources [3,172 B]
59 Get:14 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety-updates/
      main Sources [107 kB]
60 Get:15 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety-updates/
      main amd64 Packages [268 kB]
61 Get:16 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety-updates/
      main Translation-en [122 kB]
62 Get:17 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety-updates/
      universe amd64 Packages [164 kB]
63 Get:18 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety-updates/
      universe Translation-en [92.4 kB]
64 Get:19 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety-updates/
      multiverse amd64 Packages [4,840 B]
65 Get:20 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety-updates/
      multiverse Translation-en [2,708 B]
66 Get:21 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety-backports/
      universe Sources [2,468 B]
67 Get:22 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety-backports/
      main Sources [2,480 B]
68 Get:23 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety-backports/
      main amd64 Packages [3,500 B]
69 Get:24 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety-backports/
      universe amd64 Packages [3,820 B]
70 Get:25 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety-backports/
```



```
      universe Translation-en [1,592 B]
71 Get:26 http://archive.canonical.com/ubuntu yakkety/partner amd64
      Packages [2,480 B]
72 Get:27 http://security.ubuntu.com/ubuntu yakkety-security/main Sources
      [47.7 kB]
73 Get:28 https://apt.dockerproject.org/repo ubuntu-yakkety/main amd64
      Packages [2,453 B]
74 Get:29 http://security.ubuntu.com/ubuntu yakkety-security/universe
      Sources [20.7 kB]
75 Get:30 http://security.ubuntu.com/ubuntu yakkety-security/multiverse
      Sources [1,140 B]
76 Get:31 http://security.ubuntu.com/ubuntu yakkety-security/restricted
      Sources [2,292 B]
77 Get:32 http://security.ubuntu.com/ubuntu yakkety-security/main amd64
      Packages [150 kB]
78 Get:33 http://security.ubuntu.com/ubuntu yakkety-security/main
      Translation-en [68.0 kB]
79 Get:34 http://security.ubuntu.com/ubuntu yakkety-security/universe
      amd64 Packages [77.2 kB]
80 Get:35 http://security.ubuntu.com/ubuntu yakkety-security/universe
      Translation-en [47.3 kB]
81 Get:36 http://security.ubuntu.com/ubuntu yakkety-security/multiverse
      amd64 Packages [2,832 B]
82 Fetched 10.8 MB in 2s (4,206 kB/s)
83 Reading package lists... Done
84 Reading package lists...
85 Building dependency tree...
86 Reading state information...
87 The following additional packages will be installed:
88 aufs-tools cgroupfs-mount libltdl7
89 The following NEW packages will be installed:
90 aufs-tools cgroupfs-mount docker-engine libltdl7
91 0 upgraded, 4 newly installed, 0 to remove and 37 not upgraded.
92 Need to get 21.2 MB of archives.
93 After this operation, 111 MB of additional disk space will be used.
94 Get:1 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety/universe
      amd64 aufs-tools amd64 1:3.2+20130722-1.1ubuntu1 [92.9 kB]
95 Get:2 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety/universe
      amd64 cgroupfs-mount all 1.3 [5,778 B]
96 Get:3 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety/main amd64
      libltdl7 amd64 2.4.6-1 [38.6 kB]
97 Get:4 https://apt.dockerproject.org/repo ubuntu-yakkety/main amd64
      docker-engine amd64 17.05.0~ce-0~ubuntu-yakkety [21.1 MB]
98 Fetched 21.2 MB in 1s (19.8 MB/s)
99 Selecting previously unselected package aufs-tools.
```



```
(Reading database ... 63593 files and directories currently installed.)
Preparing to unpack .../aufs-tools_1%3a3.2+20130722-1.1ubuntu1_amd64.deb ...
Unpacking aufs-tools (1:3.2+20130722-1.1ubuntu1) ...
Selecting previously unselected package cgroupfs-mount.
Preparing to unpack .../cgroupfs-mount_1.3_all.deb ...
Unpacking cgroupfs-mount (1.3) ...
Selecting previously unselected package libltdl7:amd64.
Preparing to unpack .../libltdl7_2.4.6-1_amd64.deb ...
Unpacking libltdl7:amd64 (2.4.6-1) ...
Selecting previously unselected package docker-engine.
Preparing to unpack .../docker-engine_17.05.0~ce-0~ubuntu-yakkety_amd64.deb ...
Unpacking docker-engine (17.05.0~ce-0~ubuntu-yakkety) ...
Setting up aufs-tools (1:3.2+20130722-1.1ubuntu1) ...
Processing triggers for ureadahead (0.100.0-19) ...
Setting up cgroupfs-mount (1.3) ...
Processing triggers for libc-bin (2.24-3ubuntu2) ...
Processing triggers for systemd (231-9ubuntu4) ...
Setting up libltdl7:amd64 (2.4.6-1) ...
Processing triggers for man-db (2.7.5-1) ...
Setting up docker-engine (17.05.0~ce-0~ubuntu-yakkety) ...
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /lib/systemd/system/docker.service.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /lib/systemd/system/docker.socket.
Processing triggers for ureadahead (0.100.0-19) ...
Processing triggers for libc-bin (2.24-3ubuntu2) ...
Processing triggers for systemd (231-9ubuntu4) ...
+ sudo -E sh -c docker version
Client:
Version: 17.05.0-ce
API version: 1.29
Go version: go1.7.5
Git commit: 89658be
Built: Thu May 4 22:15:36 2017
OS/Arch: linux/amd64

Server:
Version: 17.05.0-ce
API version: 1.29 (minimum version 1.12)
Go version: go1.7.5
Git commit: 89658be
Built: Thu May 4 22:15:36 2017
OS/Arch: linux/amd64
```



```
Experimental: false

If you would like to use Docker as a non-root user, you should now consider
adding your user to the "docker" group with something like:

sudo usermod -aG docker albert_lee

Remember that you will have to log out and back in for this to take effect.

WARNING: Adding a user to the "docker" group will grant the ability to run
containers which can be used to obtain root privileges on the
docker host.
Refer to https://docs.docker.com/engine/security/security/#docker-daemon-attack-surface
for more information.

$

**$ sudo docker info**
Containers: 0
Running: 0
Paused: 0
Stopped: 0
Images: 0
Server Version: 17.05.0-ce
Storage Driver: aufs
Root Dir: /var/lib/docker/aufs
Backing Filesystem: extfs
Dirs: 0
Dirperm1 Supported: true
Logging Driver: json-file
Cgroup Driver: cgroupfs
Plugins:
Volume: local
Network: bridge host macvlan null overlay
Swarm: inactive
Runtimes: runc
Default Runtime: runc
Init Binary: docker-init
containerd version: 9048e5e50717ea4497b757314bad98ea3763c145
runc version: 9c2d8d184e5da67c95d601382adf14862e4f2228
init version: 949e6fa
```



```
Security Options:
apparmor
seccomp
Profile: default
Kernel Version: 4.8.0-51-generic
Operating System: Ubuntu 16.10
OSType: linux
Architecture: x86_64
CPUs: 1
Total Memory: 3.613GiB
Name: docker-7
ID: R5TW:VKXK:EKGR:GHWM:UNU4:LPJH:IQY5:X77G:NNRQ:HWBY:LIUD:4ELQ
Docker Root Dir: /var/lib/docker
Debug Mode (client): false
Debug Mode (server): false
Registry: https://index.docker.io/v1/
Experimental: false
Insecure Registries:
127.0.0.0/8
Live Restore Enabled: false

WARNING: No swap limit support
$

**$ sudo docker images**
REPOSITORY TAG IMAGE ID CREATED SIZE
$

**$ sudo docker ps**
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
$
```

2 단계**: Docker** 인스턴스와의원격 **API** 통신구성

Docker 인스턴스와의 API 통신을위해포트 4243 을엽니다. 이는 NetScaler MAS 가 Docker 인스턴스와통신하는데필요합니다.

```
$
$ **cd /etc/systemd/system**
$ **sudo vi docker-tcp.socket**
$

$ **cat docker-tcp.socket**
[Unit]
```



```
**Description=Docker Socket for the API
[Socket]
ListenStream=4243
BindIPv6Only=both
Service=docker.service
[Install]
WantedBy=sockets.target**
$

$ **sudo systemctl enable docker-tcp.socket**
Created symlink /etc/systemd/system/sockets.target.wants/docker-tcp.
   socket → /etc/systemd/system/docker-tcp.socket.
$ **sudo systemctl enable docker.socket**
$ **sudo systemctl stop docker**
$ **sudo systemctl start docker-tcp.socket**
$ **sudo systemctl start docker**
$ **sudo systemctl status docker**
● docker.service - Docker Application Container Engine
Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor
   preset: enabled)
Active: **active (running)** since Wed 2017-05-31 12:52:17 UTC; 2s ago
Docs: https://docs.docker.com
Main PID: 4133 (dockerd)
Tasks: 16 (limit: 4915)
Memory: 30.1M
CPU: 184ms
CGroup: /system.slice/docker.service
├─4133 /usr/bin/dockerd -H fd://
└─4137 docker-containerd -l unix:///var/run/docker/libcontainerd/docker
   -containerd.sock --metrics-interval=0 --start-timeout 2m -

May 31 12:52:17 docker-7 dockerd[4133]: time="2017-05-31T12
   :52:17.300890402Z" level=warning msg="Your kernel does not support
   cgroup rt peri
May 31 12:52:17 docker-7 dockerd[4133]: time="2017-05-31T12
   :52:17.301079754Z" level=warning msg="Your kernel does not support
   cgroup rt runt
May 31 12:52:17 docker-7 dockerd[4133]: time="2017-05-31T12
   :52:17.301681794Z" level=info msg="Loading containers: start."
May 31 12:52:17 docker-7 dockerd[4133]: time="2017-05-31T12
   :52:17.417539064Z" level=info msg="Default bridge (docker0) is
   assigned with an I
May 31 12:52:17 docker-7 dockerd[4133]: time="2017-05-31T12
   :52:17.465011600Z" level=info msg="Loading containers: done."
May 31 12:52:17 docker-7 dockerd[4133]: time="2017-05-31T12
```



```
    :52:17.484747909Z" level=info msg="Daemon has completed
    initialization"
May 31 12:52:17 docker-7 dockerd[4133]: time="2017-05-31T12
    :52:17.485119478Z" level=info msg="Docker daemon" commit=89658be
    graphdriver=aufs
May 31 12:52:17 docker-7 systemd[1]: Started Docker Application
    Container Engine.
May 31 12:52:17 docker-7 dockerd[4133]: time="2017-05-31T12
    :52:17.503832254Z" level=info msg="API listen on /var/run/docker.
    sock"
May 31 12:52:17 docker-7 dockerd[4133]: time="2017-05-31T12
    :52:17.504061522Z" level=info msg="API listen on [::]:4243"
$

(external)$ **curl 104.199.209.157:4243/version**
{
 "Version":"17.05.0-ce","ApiVersion":"1.29","MinAPIVersion":"1.12","
    GitCommit":"89658be","GoVersion":"go1.7.5","Os":"linux","Arch":"
    amd64","KernelVersion":"4.8.0-52-generic","BuildTime":"2017-05-04
    T22:15:36.071254972+00:00" }

(external)$
```

3 단계**: NetScaler CPX** 이미지설치

Docker App Store 에서 NetScaler CPX 이미지를가져옵니다. CPX Express 와 CPX 의이미지는동일합니다. CPX 이미지가 NetScaler MAS 를통해라이센스를부여받고설치되면정식 CPX 인스턴스가되어 1Gbps 의성능을갖습니다. 라이센스가없는경우이미지는 CPX Express 인스턴스가되어 20Mbps 및 250 개 SSL 연결을지원합니다.

```
$ **sudo docker pull store/citrix/netscalercpx:12.0-41.16**
12.0-41.16: Pulling from store/citrix/netscalercpx
4e1f679e8ab4: Pull complete
a3ed95caeb02: Pull complete
2931a926d44b: Pull complete
362cd40c5745: Pull complete
d10118725a7a: Pull complete
1e570419a7e5: Pull complete
d19e06114233: Pull complete
d3230f008ffd: Pull complete
22bdb10a70ec: Pull complete
1a5183d7324d: Pull complete
241868d4ebff: Pull complete
3f963e7ae2fc: Pull complete
```



```
fd254cf1ea7c: Pull complete
33689c749176: Pull complete
59c27bad28f5: Pull complete
588f5003e10f: Pull complete
Digest: sha256:31
   a65cfa38833c747721c6fbc142faec6051e5f7b567d8b212d912b69b4f1ebe
Status: Downloaded newer image for store/citrix/netscalercpx:12.0-41.16
$

$ **sudo docker images**
REPOSITORY TAG IMAGE ID CREATED SIZE
store/citrix/netscalercpx 12.0-41.16 6fa57c38803f 3 weeks ago 415MB
$
```

4 단계**: CPX** 인스턴스만들기

Docker 호스트에 CPX 이미지를설치합니다. 아래예제에표시된것처럼특정서비스를위한포트를열고, NetScaler MAS 의 IP 주소를지정합니다.

```
bash-2.05b# **CHOST=${
 1:-localhost }
 **
bash-2.05b# **echo | openssl s_client -connect $CHOST:443 | openssl
   x509 -fingerprint -noout | cut -d'=' -f2**
depth=0 C = US, ST = California, L = San Jose, O = Citrix NetScaler, OU
    = Internal, CN = Test Only Cert
verify error:num=18:self signed certificate
verify return:1
depth=0 C = US, ST = California, L = San Jose, O = Citrix NetScaler, OU
    = Internal, CN = Test Only Cert
verify return:1
DONE
24:AA:8B:91:7B:72:5E:6E:C1:FD:86:FA:09:B6:42:49:FC:1E:86:A4
bash-2.05b#

$ **sudo docker run -dt -p 50000:88 -p 5080:80 -p 5022:22 -p 5443:443 -
   p 5163:161/udp -e NS_HTTP_PORT=5080 -e NS_HTTPS_PORT=5443 -e
   NS_SSH_PORT=5022 -e NS_SNMP_PORT=5163 -e EULA=yes -e LS_IP=xx.xx.xx.
   xx -e PLATFORM=CP1000 --privileged=true --ulimit core=-1 -e
   NS_MGMT_SERVER=xx.xx.xx.xx:xxxx -e NS_MGMT_FINGER_PRINT=24:AA:8B
   :91:7B:72:5E:6E:C1:FD:86:FA:09:B6:42:49:FC:1E:86:A4 --env
   NS_ROUTABLE=false --env HOST=104.199.209.157 store/citrix/
   netscalercpx:12.0-41.16**
44ca1c6c0907e17a10ffcb9ffe33cd3e9f71898d8812f816e714821870fa3538
```

```
$

$ **sudo docker ps**
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
44ca1c6c0907 store/citrix/netscalercpx:12.0-41.16 "/bin/sh -c 'bash ... " 19 seconds ago Up 17 seconds 0.0.0.0:5022->22/tcp, 0.0.0.0:5080->80/tcp, 0.0.0.0:50000->88/tcp, 0.0.0.0:5163->161/udp, 0.0.0.0:5443->443/tcp gifted_perlman
$

$ **ssh -p 5022 root@localhost**
root@localhost's password:
Welcome to nsoslx 1.0 (GNU/Linux 4.8.0-52-generic x86_64)

* Documentation: https://www.citrix.com/
Last login: Mon Jun 5 18:58:51 2017 from xx.xx.xx.xx
root@44ca1c6c0907:~#
root@44ca1c6c0907:~#
root@44ca1c6c0907:~# **cli_script.sh 'show ns ip'**
exec: show ns ip
Ipaddress Traffic Domain Type Mode Arp Icmp Vserver State
--------- -------------- ---- ---- --- ---- ------- ------
1) 172.17.0.2 0 NetScaler IP Active Enabled Enabled NA Enabled
2) 192.0.0.1 0 SNIP Active Enabled Enabled NA Enabled
Done
root@44ca1c6c0907:~# **cli_script.sh 'show licenseserver'**
exec: show licenseserver
1) ServerName: xx.xx.xx.xxPort: 27000 Status: 1 Grace: 0 Gptimeleft: 0
Done
root@44ca1c6c0907:~# cli_script.sh 'show capacity'
exec: show capacity
Actualbandwidth: 1000 Platform: CP1000 Unit: Mbps Maxbandwidth: 3000 Minbandwidth: 20 Instancecount: 0
Done
root@44ca1c6c0907:~#

$ **sudo iptables -t nat -L -n**
Chain PREROUTING (policy ACCEPT)
target prot opt source destination
DOCKER all -- 0.0.0.0/0 0.0.0.0/0 ADDRTYPE match dst-type LOCAL

Chain INPUT (policy ACCEPT)
target prot opt source destination

Chain OUTPUT (policy ACCEPT)
```

```
target prot opt source destination
DOCKER all -- 0.0.0.0/0 !127.0.0.0/8 ADDRTYPE match dst-type LOCAL

Chain POSTROUTING (policy ACCEPT)
target prot opt source destination
MASQUERADE all -- 172.17.0.0/16 0.0.0.0/0
MASQUERADE tcp -- 172.17.0.2 172.17.0.2 tcp dpt:443
MASQUERADE udp -- 172.17.0.2 172.17.0.2 udp dpt:161
MASQUERADE tcp -- 172.17.0.2 172.17.0.2 tcp dpt:88
MASQUERADE tcp -- 172.17.0.2 172.17.0.2 tcp dpt:80
MASQUERADE tcp -- 172.17.0.2 172.17.0.2 tcp dpt:22

Chain DOCKER (2 references)
target prot opt source destination
RETURN all -- 0.0.0.0/0 0.0.0.0/0
DNAT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:5443 to:172.17.0.2:443
DNAT udp -- 0.0.0.0/0 0.0.0.0/0 udp dpt:5163 to:172.17.0.2:161
DNAT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:50000 to:172.17.0.2:88
DNAT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:5080 to:172.17.0.2:80
DNAT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:5022 to:172.17.0.2:22
$
```

5 단계**: NetScaler MAS** 를통해 **NetScaler CPX** 에라이센스부여

NetScaler MAS 가온-프레미스에서실행되는경우 CPX 가 MAS 와통신하며정보를전송하는지확인할수있어야합니다. 다음이미지는 CPX 가 NetScaler MAS 에서라이센스를가져오는것을보여줍니다.

6 단계**: NetScaler CPX** 에서부하분산서비스를구성하고구성확인

먼저 Docker 호스트에 NGINX 웹서버를설치합니다. 그런다음두웹서버의부하를분산하도록 NetScaler CPX 에부하분산을구성하고이구성을테스트합니다.

NGINX 웹서버설치

다음예제에표시된명령을사용하여 NGINX 웹서버를설치합니다.

```
$ sudo docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
Digest: sha256:41
    ad9967ea448d7c2b203c699b429abe1ed5af331cd92533900c6d77490e0268
Status: Image is up to date for nginx:latest
```



```
6
7
8  **$ sudo docker run -d -p 81:80 nginx**
9  098a77974818f451c052ecd172080a7d45e446239479d9213cd4ea6a3678616f
10
11
12 **$ sudo docker run -d -p 82:80 nginx**
13 bbdac2920bb4085f70b588292697813e5975389dd546c0512daf45079798db65
14
15
16 **$ sudo iptables -t nat -L -n**
17 Chain PREROUTING (policy ACCEPT)
18 target prot opt source destination
19 DOCKER all -- 0.0.0.0/0 0.0.0.0/0 ADDRTYPE match dst-type LOCAL
20
21 Chain INPUT (policy ACCEPT)
22 target prot opt source destination
23
24 Chain OUTPUT (policy ACCEPT)
25 target prot opt source destination
26 DOCKER all -- 0.0.0.0/0 !127.0.0.0/8 ADDRTYPE match dst-type LOCAL
27
28 Chain POSTROUTING (policy ACCEPT)
29 target prot opt source destination
30 MASQUERADE all -- 172.17.0.0/16 0.0.0.0/0
31 MASQUERADE tcp -- 172.17.0.2 172.17.0.2 tcp dpt:443
32 MASQUERADE udp -- 172.17.0.2 172.17.0.2 udp dpt:161
33 MASQUERADE tcp -- 172.17.0.2 172.17.0.2 tcp dpt:88
34 MASQUERADE tcp -- 172.17.0.2 172.17.0.2 tcp dpt:80
35 MASQUERADE tcp -- 172.17.0.2 172.17.0.2 tcp dpt:22
36 MASQUERADE tcp -- 172.17.0.3 172.17.0.3 tcp dpt:80
37 MASQUERADE tcp -- 172.17.0.4 172.17.0.4 tcp dpt:80
38
39 Chain DOCKER (2 references)
40 target prot opt source destination
41 RETURN all -- 0.0.0.0/0 0.0.0.0/0
42 DNAT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:5443 to:172.17.0.2:443
43 DNAT udp -- 0.0.0.0/0 0.0.0.0/0 udp dpt:5163 to:172.17.0.2:161
44 DNAT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:50000 to:172.17.0.2:88
45 DNAT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:5080 to:172.17.0.2:80
46 DNAT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:5022 to:172.17.0.2:22
47 DNAT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:81 to:172.17.0.3:80
48 DNAT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:82 to:172.17.0.4:80
49 $
```

CPX 에부하분산을구성하고두웹서비스로부하가분산되는지확인

```
$ **ssh -p 5022 root@localhost**
root@localhost's password:
Welcome to nsoslx 1.0 (GNU/Linux 4.8.0-52-generic x86_64)

* Documentation: https://www.citrix.com/
Last login: Mon Jun 5 18:58:54 2017 from 172.17.0.1
root@44ca1c6c0907:~#
root@44ca1c6c0907:~#
root@44ca1c6c0907:~#
root@44ca1c6c0907:~#
root@44ca1c6c0907:~# **cli_script.sh "add service web1 172.17.0.3 HTTP
    80"**
exec: add service web1 172.17.0.3 HTTP 80
Done
root@44ca1c6c0907:~# **cli_script.sh "add service web2 172.17.0.4 HTTP
    80"**
exec: add service web2 172.17.0.4 HTTP 80
Done
root@44ca1c6c0907:~# **cli_script.sh "add lb vserver cpx-vip HTTP
    172.17.0.2 88"**
exec: add lb vserver cpx-vip HTTP 172.17.0.2 88
Done
root@44ca1c6c0907:~# **cli_script.sh "bind lb vserver cpx-vip web1"**
exec: bind lb vserver cpx-vip web1
Done
root@44ca1c6c0907:~# **cli_script.sh "bind lb vserver cpx-vip web2"**
exec: bind lb vserver cpx-vip web2
Done
root@44ca1c6c0907:~#

root@44ca1c6c0907:~# **cli_script.sh 'show lb vserver cpx-vip'**
exec: show lb vserver cpx-vip

cpx-vip (172.17.0.2:88) - HTTP Type: ADDRESS
State: UP
Last state change was at Mon Jun 5 19:01:49 2017
Time since last state change: 0 days, 00:00:42.620
Effective State: UP
Client Idle Timeout: 180 sec
Down state flush: ENABLED
Disable Primary Vserver On Down : DISABLED
Appflow logging: ENABLED
Port Rewrite : DISABLED
```

```
No. of Bound Services : 2 (Total) 2 (Active)
Configured Method: LEASTCONNECTION
Current Method: Round Robin, Reason: A new service is bound
    BackupMethod: ROUNDROBIN
Mode: IP
Persistence: NONE
Vserver IP and Port insertion: OFF
Push: DISABLED Push VServer:
Push Multi Clients: NO
Push Label Rule: none
L2Conn: OFF
Skip Persistency: None
Listen Policy: NONE
IcmpResponse: PASSIVE
RHIstate: PASSIVE
New Service Startup Request Rate: 0 PER_SECOND, Increment Interval: 0
Mac mode Retain Vlan: DISABLED
DBS_LB: DISABLED
Process Local: DISABLED
Traffic Domain: 0
TROFS Persistence honored: ENABLED
Retain Connections on Cluster: NO

2) web1 (172.17.0.3: 80) - HTTP State: UP Weight: 1
3) web2 (172.17.0.4: 80) - HTTP State: UP Weight: 1
Done
root@44ca1c6c0907:~#

(external)$ **curl 104.199.209.157:50000**
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
body {

width: 35em;
margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif;
 }

</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
```



```
<p>If you see this page, the nginx web server is successfully installed
    and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href=”http://nginx.org/”>nginx.org</a>.<br/>
Commercial support is available at
<a href=”http://nginx.com/”>nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
(external)$

![localized image](/en-us/netscaler-cpx/12/media/cpx-image-5.png)

(external)$ for i in {
 1..100 }
  ; **do curl http://104.199.209.157:50000 -o /dev/null ; done**

  % Total    % Received % Xferd  Average Speed   Time    Time     Time
    Current

                                 Dload  Upload   Total   Spent    Left
    Speed

100   612  100   612    0     0   1767      0 --:--:-- --:--:--
   --:--:--  1768

  % Total    % Received % Xferd  Average Speed   Time    Time     Time
    Current

                                 Dload  Upload   Total   Spent    Left
    Speed

100   612  100   612    0     0   1893      0 --:--:-- --:--:--
   --:--:--  1894

  % Total    % Received % Xferd  Average Speed   Time    Time     Time
    Current

                                 Dload  Upload   Total   Spent    Left
    Speed

100   612  100   612    0     0   1884      0 --:--:-- --:--:--
```



```
      --:--:--  1883
121
122   % Total    % Received % Xferd  Average Speed   Time    Time     Time
      Current
123
124                                  Dload  Upload   Total   Spent    Left
      Speed
125
126 100   612  100   612    0     0   1917      0 --:--:-- --:--:--
      --:--:--  1924
127
128   % Total    % Received % Xferd  Average Speed   Time    Time     Time
      Current
129
130                                  Dload  Upload   Total   Spent    Left
      Speed
131
132 100   612  100   612    0     0   1877      0 --:--:-- --:--:--
      --:--:--  1883
133
134   % Total    % Received % Xferd  Average Speed   Time    Time     Time
      Current
135
136                                  Dload  Upload   Total   Spent    Left
      Speed
137
138 100   612  100   612    0     0   1852      0 --:--:-- --:--:--
      --:--:--  1848
139
140   % Total    % Received % Xferd  Average Speed   Time    Time     Time
      Current
141
142                                  Dload  Upload   Total   Spent    Left
      Speed
143
144 100   612  100   612    0     0   1860      0 --:--:-- --:--:--
      --:--:--  1865
145
146   % Total    % Received % Xferd  Average Speed   Time    Time     Time
      Current
147
148                                  Dload  Upload   Total   Spent    Left
      Speed
149
150 100   612  100   612    0     0   1887      0 --:--:-- --:--:--
```



```
      --:--:--  1888
151
152   % Total    % Received % Xferd  Average Speed   Time    Time     Time
      Current
153
154                                  Dload  Upload   Total   Spent    Left
      Speed
155
156 100   612  100   612    0     0   1802      0 --:--:-- --:--:--
      --:--:--  1800
157
158   % Total    % Received % Xferd  Average Speed   Time    Time     Time
      Current
159
160                                  Dload  Upload   Total   Spent    Left
      Speed
161
162 100   612  100   612    0     0   1902      0 --:--:-- --:--:--
      --:--:--  1906
163
164   % Total    % Received % Xferd  Average Speed   Time    Time     Time
      Current
165
166                                  Dload  Upload   Total   Spent    Left
      Speed
167
168 100   612  100   612    0     0   1843      0 --:--:-- --:--:--
      --:--:--  1848
169
170
171
172   % Total    % Received % Xferd  Average Speed   Time    Time     Time
      Current
173
174                                  Dload  Upload   Total   Spent    Left
      Speed
175
176 100   612  100   612    0     0   1862      0 --:--:-- --:--:--
      --:--:--  1860
177
178   % Total    % Received % Xferd  Average Speed   Time    Time     Time
      Current
179
180                                  Dload  Upload   Total   Spent    Left
      Speed
```



```
100   612  100   612    0     0   1806      0 --:--:-- --:--:--
   --:--:--  1810

  % Total    % Received % Xferd  Average Speed   Time    Time     Time
    Current

                                 Dload  Upload   Total   Spent    Left
    Speed

100   612  100   612    0     0   1702      0 --:--:-- --:--:--
   --:--:--  1704

(external)$




root@44ca1c6c0907:~# **cli_script.sh 'stat lb vserver cpx-vip'**

exec: stat lb vserver cpx-vip


Virtual Server Summary

                      vsvrIP  port     Protocol        State   Health
   actSvcs

cpx-vip           172.17.0.2    88         HTTP           UP      100
           2


           inactSvcs

cpx-vip            0


Virtual Server Statistics

                                          Rate (/s)
   Total
```



```
219
220 Vserver hits                                          0
       101
221
222 Requests                                             0
       101
223
224 Responses                                            0
                  101
225
226 Request bytes                                        0
       8585
227
228 Response bytes                                       0
       85850
229
230 Total Packets rcvd                                   0
       708
231
232 Total Packets sent                                   0
       408
233
234 Current client connections                          --
        0
235
236 Current Client Est connections                      --
        0
237
238 Current server connections                          --
                         0
239
240 Current Persistence Sessions                        --
        0
241
242 Requests in surge queue                             --
        0
243
244 Requests in vserver's surgeQ                        --
        0
245
246 Requests in service's surgeQs                       --
        0
247
248 Spill Over Threshold                                --
        0
```



```
249
250 Spill Over Hits                                    --
        0
251
252 Labeled Connection                                 --
        0
253
254 Push Labeled Connection                            --
        0
255
256 Deferred Request                                    0
        0
257
258 Invalid Request/Response                           --
        0
259
260 Invalid Request/Response Dropped                   --
        0
261
262 Vserver Down Backup Hits                           --
        0
263
264 Current Multipath TCP sessions                     --
        0
265
266 Current Multipath TCP subflows                     --
        0
267
268 Apdex for client response times.                   --
       1.00
269
270 Average client TTLB                                --
        0
271
272 web1              172.17.0.3    80          HTTP           UP       51
            0/s
273
274 web2              172.17.0.4    80          HTTP           UP       50
            0/s
275
276 Done
277
278 root@44ca1c6c0907:~#
```


Locations
Corporate Headquarters | 851 Cypress Creek Road Fort Lauderdale, FL 33309, United States
Silicon Valley | 4988 Great America Parkway Santa Clara, CA 95054, United States